UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DETERMINACIÓN DE INDICADORES DE LA PRODUCCIÓN DE SERVICIOS DE UNA RED ETHERNET UTILIZANDO UN MÓDULO DE CAPTURA DE PAQUETES IP BASADO EN FPGA

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PRESENTADO POR:

ANDRES MIJAIL LEIVA COCHACHIN

PROMOCIÓN
2009-II

LIMA-PERÚ
2012

A mi querida madre Judith
por su incondicional amor y comprensión.
A mi querido padre Alberto
por su constante apoyo para cumplir mis objetivos.
A mi querida universidad, la UNI,
por brindarme sólidos conocimientos.

SUMARIO

En la presente tesis se explica la metodología utilizada en el diseño e implementación de una solución integral (un módulo de captura de paquetes IP y una aplicación de software ejecutada en un servidor) que tiene el propósito de determinar los indicadores de la producción y utilización de servicios de una red Ethernet.

La solución es capaz de realizar la recopilación, evaluación y supervisión del contenido del tráfico IP para determinar las diferentes fuentes y destinos de cada servicio IP tanto dentro de la red como fuera de la misma, así como el nivel de tráfico significativo generado por cada servicio, para posteriormente generar reportes con los que una entidad, poseedora de una red privada, puede evaluar el interés y uso de los usuarios de sus diferentes servicios disponibles, determinar los principales servicios externos accedidos por usuarios internos, así como si tales servicios están siendo brindados adecuadamente. Con tal información la entidad puede determinar el nivel de utilización de sus servicios así como evaluar el comportamiento en el tiempo de sus usuarios buscando alcanzar sus objetivos o corregir comportamientos mediante la aplicación de mejores prácticas para mejorar sus prestaciones y conseguir el mayor beneficio para su entidad.

La solución integral consta de dos diseños

- El “Módulo de Captura”.- El cual es la solución de hardware basada en FPGA (Field Programmable Gate Array), encargada de capturar paquetes IP, realizar un preprocesamiento de la información capturada y enviar los resultados del preprocesamiento hacia un ordenador.
- La “Aplicación para Modelamiento y Reportes”.- La cual es una solución de software que utiliza el método desarrollado por Kuai Xu, Zhi-Li Zhang y Supratik Battacharyya y resumido en el documento “Profiling Internet Backbone Traffic: Behavior Models and Applications". Esta aplicación procesa la información proveniente del módulo de captura, y genera reportes estadísticos de indicadores de la producción de servicios.

ÍNDICE

INTRODUCCIÓN

La presente tesis es un proyecto de la especialidad en la cual se aplica conocimientos científicos y técnicos relacionados con la ingeniería electrónica. Considerando que el proyecto desarrollado es una solución integral que tiene como finalidad determinar los indicadores de la producción de servicios de una red Ethernet, es que la solución hace la medición, preparación y análisis de los datos, y utiliza técnicas estadísticas y la metodología presentada en el evento SIGCOMM'05 (Special Interest Group on Data Communication) para caracterizar el comportamiento de usuarios y servicios en una red Ethernet produciendo un modelo estructural que permite explicar el comportamiento del tráfico. Es necesario mencionar que SIGCOMM es un foro profesional, perteneciente a la ACM (Association for Computing Machinery), para tratar temas relacionados a redes de computadores y a comunicaciones.

Toda entidad que cuenta con una red IP privada con acceso a Internet envía y recibe a diario una gran cantidad de información a través de su enrutador principal (router). La solución presentada es necesaria debido al hecho de que la medición del nivel de tráfico, basado únicamente en la cantidad de paquetes por segundo que circula por el enrutador principal, sólo permite tener datos cuantitativos los cuales no permiten establecer el nivel de tráfico de cada uno de los servicios de red en la entidad. Dada estas y otras deficiencias, que se explican en la sección 1.3, es que el proyecto desarrollado se enfoca en estudiar dos aspectos principales:

- El nivel de tráfico saliente generado por cada uno de los servicios ofrecidos por una entidad hacia usuarios externos a la entidad.
- El nivel de tráfico entrante generado por servicios ubicados fuera del entorno de la entidad externos hacia usuarios dentro de la entidad.

El presente proyecto fue realizado, como tema de investigación complementaria en INICTEL-UNI, entre mayo y diciembre de 2011. El proyecto es una solución integral que consta de dos diseños:

- El “Módulo de Captura”, diseñado con un módulo de desarrollo con FPGA, y en el cual se ejecutan dos programas que fueron diseñados para: la captura de paquetes IP, y la extracción y envío de datos por UDP hacia un servidor central, utilizando una pila de protocolos TCP/IP.
- La “Aplicación para Modelamiento y Reportes”, que ejecuta tres rutinas que fueron

diseñadas para: la recepción y almacenamiento en una base de datos de la información enviada por el “Módulo de Captura”, obtener un modelo del comportamiento entre usuarios y servicios, y mostrar los resultados en una interfaz gráfica de usuario (tipo web).

El presente documento está organizado en seis capítulos principales y anexos que brindan información complementaria.

- Capítulo I “Planteamiento del problema de ingeniería”.- Capítulo introductorio en donde se explica el problema de ingeniería y se precisan los objetivos de la tesis. Así mismo en este capítulo se hace una evaluación de la problemática para la respectiva justificación del desarrollo del proyecto. En este capítulo también se establecen los alcances del proyecto desarrollado, y finalmente se presenta una síntesis de la tesis realizada.
- Capítulo II “Marco teórico”.- En el que se desarrollan cuatro tópicos principales: Sistemas embebidos con capacidad de conexión a redes Ethernet, estructura de paquetes que contienen aplicaciones basadas en IPv4, protocolos de intercambio de información de flujos de tráfico IP, e introducción a la teoría de la información (conceptos básicos de estadística descriptiva, medidas de tendencia central y de dispersión, probabilidad, entropía, entropía estandarizada).
- Capítulo III “Alternativas de Solución”.- Que consta de tres tópicos: Elementos necesarios, Alternativas para el módulo de captura de paquetes y Alternativas para la aplicación de software especializada.
- Capítulo IV “Diseño del módulo de captura de tráfico”.- Que consta de tres tópicos: diseño de la estructura de hardware del sistema, desarrollo del software de aplicación del sistema, y pruebas de validación.
- Capítulo V “Diseño de la aplicación para modelamiento y reportes”.- En el que se desarrolla cinco secciones: proceso de recolección de información de flujos, proceso de modelamiento estructural basado en la teoría de la información, definición de indicadores de producción de servicios, diseño de la aplicación de reportes, e integración de los procesos.
- Capítulo VI “Análisis y presentación de resultados”.- Se desarrollan cuatro ítems: descripción de la prueba, resultados de las pruebas de evaluación del sistema, análisis del impacto del sistema en la red de pruebas y estimación de costos.

Las tablas y figuras en las que no se indica su origen, son de autoría propia, es decir realizadas por el autor de la presente tesis.

CAPÍTULO I
PLANTEAMIENTO DEL PROBLEMA DE INGENIERÍA

En este capítulo consta de cinco secciones. Primero se explica el problema de ingeniería, luego se precisan los objetivos de la tesis. En la tercera sección se hace una evaluación de la problemática y en la cuarta se establecen los alcances del proyecto desarrollado, finalmente en la quinta sección se presenta una resumen del trabajo.

1.1 Descripción del problema

Necesidad, por parte de una entidad que posea una red privada con acceso a Internet, de recopilar, evaluar y supervisar el contenido y los diferentes destinos del mismo para cada servicio, así como el nivel de tráfico generado.

Con la información recopilada una entidad puede establecer quiénes son los usuarios potenciales, discriminando en la evaluación los que no lo son.

Por otro lado, con la información sobre el nivel de tráfico, una entidad puede obtener indicadores del rendimiento que le permiten percibir aquellos servicios que requieren ser innovados o modificados para mantener un nivel de tráfico adecuado a sus objetivos.

Con toda la información obtenida y su respectivo procesamiento, una entidad será capaz de evaluar el interés por parte de los usuarios para un servicio en particular, así como si estos servicios están siendo brindados adecuadamente.

Mientras una entidad genere mayores servicios de calidad y a su vez éstos mismos sean accedidos por un mayor número de clientes, la entidad logrará conseguir mayores beneficios.

1.2 Objetivos del trabajo

Determinar los indicadores de la producción de servicios de una red Ethernet utilizando un módulo de captura de paquetes IP basado en FPGA.

Este objetivo se logra caracterizando el comportamiento de usuarios y servicios en una red Ethernet, mediante una aplicación de software, para producir un modelo estructural que permita explicar el comportamiento del tráfico.

De lo expuesto, los objetivos secundarios de la tesis son:

- Diseñar una solución de hardware, basada en FPGA, que permita la captura de paquetes IP, realice un preprocesamiento y sus resultados los envíe hacia un ordenador. A esta solución se la denomina “Módulo de Captura”.
- Diseñar una solución de software, basado en un método presentado en el evento

SIGCOMM'05 [1], que procese la información proveniente del módulo de captura, y genere reportes estadísticos de indicadores de la producción de servicios. A esta solución se la denomina "Aplicación para Modelamiento y Reportes".

Nota:
SIGCOMM son las siglas de "Special Interest Group on Data Communication", Un foro profesional para la discusión de tópicos en el campo de las comunicaciones y redes de computadoras, lo que incluye diseño técnico e ingeniería, regulaciones y operaciones, y las implicaciones sociales referidas a redes de computadoras [2].

1.3 Evaluación del problema

Hoy en día las redes de datos y especialmente el internet, se ha convertido en una importante vía para el intercambio de datos de todo tipo. Estos datos pueden estar incluidos en correos electrónicos, páginas web, archivos de texto, audio, video, etc. Las ventajas que ofrece la utilización de las redes de datos son múltiples tales como la velocidad relativamente alta para el transporte de datos, la disponibilidad permanente, la ubicuidad en el acceso y muchas más.

Debido a estas y otras ventajas, en los últimos años se ha incrementado el uso de las tecnologías de información para el acceso a Internet en Perú. Este aumento se ha visto resaltado en el uso de la telefonía móvil. Esto se refleja en las últimas encuestas realizadas por el INEI la cual manifiesta que en el 2010 el uso de teléfonos celulares en el país fue de 73.1%, cifra mucho mayor comparada con la del 2004 que fue de 16.4% [3]. Este aumento se debe, principalmente, a que muchos usuarios utilizan las redes fijas y móviles para acceder a los servicios que muchas instituciones, empresas y entidades en general ofrecen en Internet tales como correo electrónico, compras en línea, almacenamiento de archivos, acceso remoto a terminales, traducción de nombres de dominio, videoconferencias, servicios de voz, autenticación, acceso a base de datos y muchos otros.

Toda entidad que cuenta con una red IP privada con acceso a Internet envía y recibe a diario una gran cantidad de información a través de su enrutador principal (router), el cual es un dispositivo de red que se ubica en el borde de la red interna y que se encarga de encaminar cada paquete IP hacia su destino, tanto hacia Internet como a la red interna de los datos provenientes del exterior.

El nivel de tráfico en la interfaz principal del enrutador podría aumentar o disminuir considerablemente según el nivel de utilización de la capacidad total de la red.

Usualmente, cada entidad cuenta con un conjunto de personas dedicadas a la gestión de toda su red. Una de sus funciones principales es precisamente medir el nivel de tráfico que entra y sale de la entidad por medio del enrutador principal, para generar alarmas en caso que dichos niveles se alejen de ciertos valores preestablecidos. Sin embargo, el hecho de medir el nivel de tráfico basado únicamente en la cantidad de paquetes por segundo que circula por el enrutador principal, sólo permite tener datos cuantitativos que

no reflejan claramente el nivel de tráfico de cada uno de los servicios de red en la entidad; lo que realmente se refleja es el nivel de tráfico de toda la red en general.

En tales cifras se refleja todas las conexiones establecidas y datos intercambiados tanto de usuarios internos hacia nodos externos como de usuarios externos a nodos internos. También se reflejan otros tipos de paquetes como ICMP (Internet Control Message Protocol), ARP (Address Resolution Protocol), STP (Spanning Tree Protocol), etc. los cuales no están relacionados con ningún servicio en particular.

Midiendo únicamente el tráfico en el enrutador principal no se obtiene un nivel de profundidad que permita conocer específicamente hacia qué usuarios externos o internos se dirige el nivel tráfico medido y con qué servicio está relacionado cada uno de las conexiones establecidas.

Dada estas deficiencias es que el proyecto desarrollado se enfoca a estudiar dos aspectos principales:

- El nivel de tráfico saliente generado por cada uno de los servicios ofrecidos por una entidad hacia usuarios externos a la entidad.
- El nivel de tráfico entrante obtenido por usuarios internos de servicios ubicados fuera del entorno de la entidad.

Para tener conocimiento de los niveles de tráfico mencionados, es necesario tener no sólo cifras cuantitativas sino también valores cualitativos que permitan identificar:

- Dos nodos, un nodo referido al usuario cliente y el otro al servidor donde se encuentra alojado un servicio en particular.
- Al servicio, de manera única, que está involucrado en la conexión.
- Si una conexión establecida corresponde a un servicio interno o externo a la entidad.

El conocimiento de estos valores cuantitativos y sus respectivos niveles de tráfico permite entender la eficiencia y rendimiento global con que opera cada uno de los servicios implementados. Igualmente, se puede establecer los usuarios potenciales de cada servicio, así como su ubicación (usuarios locales o externos). Asimismo, permite sintetizar cifras que revelen qué porcentaje de tráfico, relacionado a servicios, es cursado desde y hacia afuera de la red.

Obtener todas las cifras anteriormente descritas, produce excelentes beneficios para cualquier entidad en general que cuente con una red privada con acceso a internet. A continuación se describen algunos de los principales beneficios que brindan estas cifras:

- El administrador de la red puede entender el comportamiento de cada uno de los usuarios clientes de sus servicios. Se podría saber las horas de mayor demanda de cada servicio y qué usuarios son lo que generan mayor demanda en cada servicio.
- Permite determinar aquellos servicios que requieran ser innovados para mejorar la

aceptación por parte de los usuarios clientes. La innovación de un servicio permite el mejoramiento de su rendimiento y el aumento en el nivel de preferencia por parte de los usuarios.

- Con estas cifras la entidad podrá tomar las decisiones respectivas para innovar algunos servicios y así se pueda satisfacer la demanda global de los usuarios de manera óptima así como poder atraer nuevos clientes.
- El perfil de cada cliente potencial de un servicio podría conducir fácilmente a la clasificación del nivel de producción según su ubicación, es decir, se podría determinar hacia dónde se dirige la información producida y cuál es el sector de usuarios que más utilizan un servicio en particular.
- El monitoreo constante de estas cifras posibilita la síntesis de gráficos estadísticos que brindan una clara visión de la evolución en el tiempo (horario, diario, semanal, mensual, anual) de cada uno de los servicios en funcionamiento. Estos gráficos permiten realizar comparaciones de tiempos anteriores con tiempos actuales para saber si hubo mejora o no en la manera de operar del administrador en la implementación, innovación o modificación de un servicio en particular. Asimismo, el estudio de estos gráficos es de suma importancia ya que está íntimamente relacionado con las diversas utilidades que pueda generar cada servicio a lo largo del tiempo.

Según la problemática expuesta en esta sección, es que se justifica la realización del proyecto de determinación de indicadores de la producción de servicios de una red Ethernet utilizando un módulo de captura de paquetes IP basado en FPGA, como tema de tesis.

1.4 Alcance del trabajo

El presente proyecto fue realizado, como tema de investigación complementaria en INICTEL-UNI, entre mayo y diciembre de 2011. Para su desarrollo se estableció que el proyecto cumpliera con los siguientes alcances y limitaciones:

- Determinar el rendimiento global y efectivo de la producción de servicios en una red.
- Utilizar para el diseño del hardware (Módulo de Captura) el módulo de desarrollo FPGA adquirido sólo con fines de prueba, utilizándose un software de diseño con licencia de evaluación.
- Usar únicamente una sonda basada en FPGA para la captura de paquetes IP en un único enlace y leer los paquetes que circulan en ambas direcciones.
- Basar la captura de datos, en la inspección de las cabeceras de los paquetes sin analizar su carga útil, es decir, no tener en cuenta el tipo de aplicación a la que pertenece cada uno de los paquetes IP.
- Exportar los datos desde el “Módulo de Captura” al servidor de la “Aplicación para

Modelamiento y Reportes" de manera continua, mediante un protocolo de comunicación basado únicamente en UDP utilizando la arquitectura cliente-servidor y no utilizar el estándar IPFIX para exportar los flujos. Procesar los datos en el servidor cada 5 minutos.

- Desarrollar la "Aplicación para Modelamiento y Reportes" en lenguaje Java. Además almacenar la información en una base de datos MySQL instalada en el mismo servidor.
- Realizar las pruebas en un entorno de red LAN de la RAAP (Red Académica Peruana) en las instalaciones del INICTEL-UNI. Instalar la sonda basada en FPGA junto al enrutador principal de la red, con un ancho de banda de 2 Mbps (para la fecha de estudio).
- Basar la técnica de modelamiento en una metodología presentada en el SIGCOMM 2005 titulado: "Profiling Internet Backbone Traffic: Behavior Models and Applications", debido a que demostró ser un método eficiente que produce un modelo estructural compacto basado en la teoría de la información.
- Presentar la información sintetizada mediante tablas y gráficos estadísticos que representen solamente el nivel de tráfico significativo en la red de pruebas.

Nota: La RAAP (Red Académica Peruana) es una red que integra las universidades e institutos de investigación del país en una sola red (Figura 1.1), brindando una independencia de conexión con la red Internet actual (orientada al ámbito comercial).

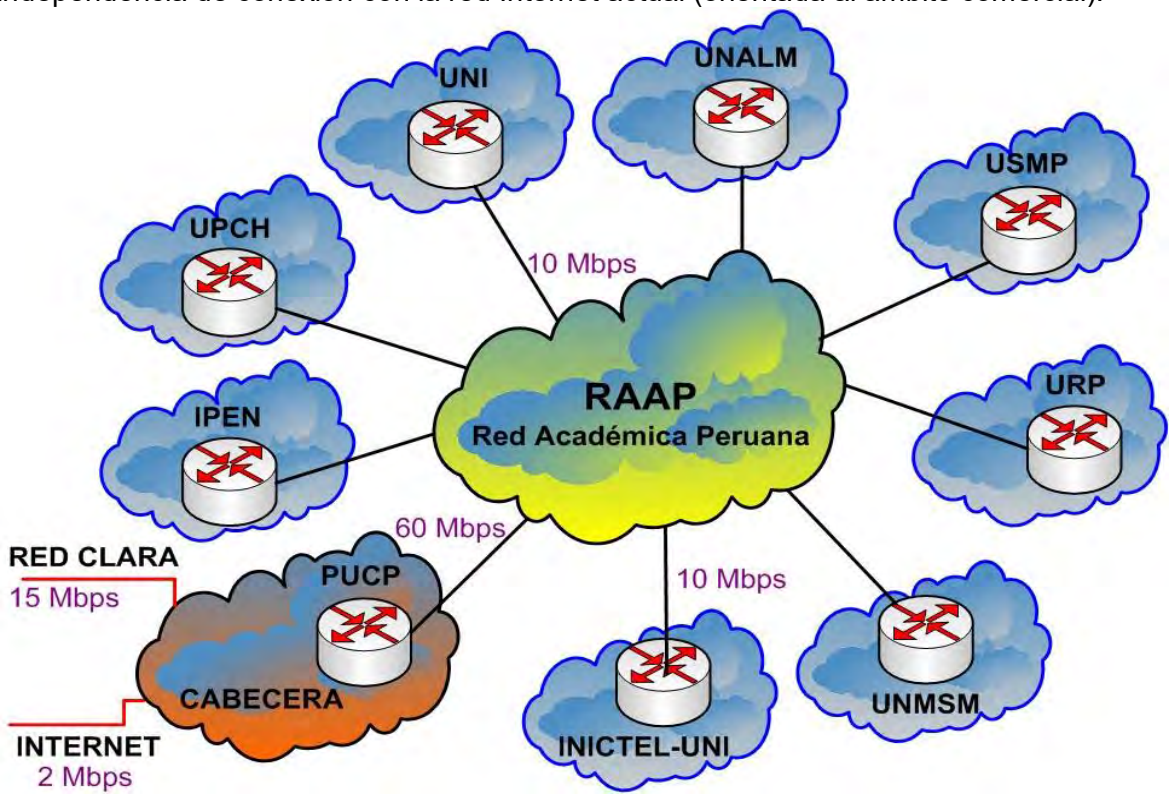


Figura 1.1 Topología de la RAAP (Fuente: Página oficial de la RAAP)

1.5 Síntesis del trabajo

En el presente trabajo se realiza el desarrollo de un proyecto de ingeniería que consta de un hardware denominado "Módulo de Captura" basada en FPGA, y en una aplicación, denominada "Aplicación para Modelamiento y Reportes" basada en una metodología de

modelamiento estructural, la cual es desarrollada en Java y se ejecuta en un servidor.

El objetivo del sistema desarrollado es capturar el tráfico de paquetes, desde y hacia Internet, en el enlace principal de una red de área local (LAN), que para los fines de experimentación es la LAN de INICTEL-UNI. Los paquetes extraídos son procesados en el mismo "Módulo de Captura" y la información obtenida es exportada al servidor para procesada por la "Aplicación para Modelamiento y Reportes" y presentar la información sintetizada en un interfaz web.

La Figura 1.2 muestra el escenario (una LAN) en donde se aplica el proyecto desarrollado (zona sombreada) que consta del "Módulo de Captura" y de la "Aplicación para Modelamiento y Reportes" (contenida en un servidor).

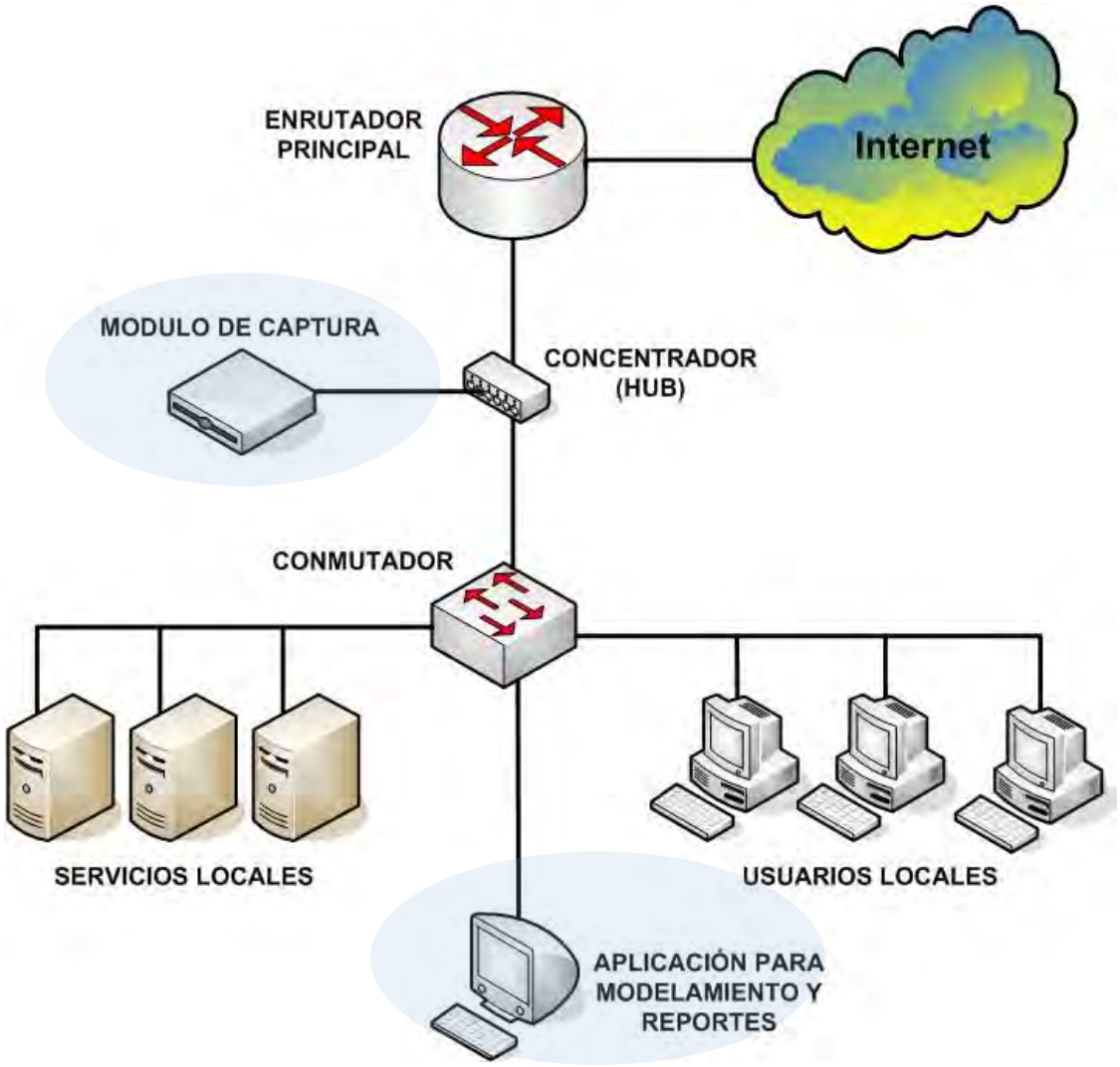


Figura 1.2 Esquema general del proyecto

Para los propósitos del proyecto, fue necesaria la inclusión de un concentrador (hub), el cual también es mostrado en la anterior figura, y que está situado entre el conmutador (switch) y el enrutador (router) con el que se accede a la Internet.

El diseño del Módulo de Captura consistió de lo siguiente:

- Diseño de un hardware con capacidad de conexión a redes Ethernet 100BASE-T utilizando un módulo de desarrollo con FPGA.
- Diseño de un programa de captura de paquetes IP utilizando una pila de protocolos TCP/IP, que se ejecuta en el módulo de desarrollo con FPGA.
- Diseño de un programa de extracción y envío de datos por UDP hacia un servidor central, utilizando una pila de protocolos TCP/IP, el cual se ejecuta en el módulo de desarrollo con FPGA.

El diseño de la Aplicación para Modelamiento y Reportes consistió de lo siguiente:

- Diseño de una rutina para la recepción y almacenamiento en una base de datos de la información enviada por el Módulo de Captura.
- Diseño de una rutina para obtener un modelo del comportamiento entre usuarios y servicios. Esta rutina consta de tres subrutinas, cada una con los siguientes objetivos:
  - Extracción de flujos de paquetes significativos utilizando la información enviada por el módulo de captura.
  - Distribución o clasificación de flujos significativos en clases de comportamiento.
  - Elaboración de un modelo del comportamiento basado en técnicas de modelamiento estructural.
- Diseño de una rutina para mostrar los resultados en una interfaz gráfica de usuario (tipo web), con la cual el usuario pueda determinar el rango de tiempo del estudio estadístico y observar los indicadores.

Cada diseño principal es desarrollado en capítulos separados. Los subdiseños son explicados en las subsecciones en sus correspondientes capítulos. Cada capítulo se inicia con el planteamiento de la solución (requerimientos, alternativas, etc.).

El trabajo concluye con el análisis y presentación de resultados, en la que se incluye la estimación de costos.

La tesis tiene un capítulo muy importante, en donde se desarrollan los conceptos teóricos básicos para la adecuada comprensión del trabajo realizado

CAPÍTULO II
MARCO TEÓRICO

La implementación de la solución alternativa que se propone en el presente trabajo de tesis, para la determinación de indicadores de la producción de servicios en una red Ethernet, tiene como punto de partida el estudio de las posibles formas de acceso a una red IP para poder capturar paquetes.

Debido a que el módulo de captura está basado en FPGA, se debe tener claro los conceptos básicos, la razón de ser y la utilidad de un sistema embebido. También es importante tener en cuenta el concepto y la estructura de un FPGA y la pila de protocolos específica que se utiliza.

Luego de obtener un paquete IP, es necesario conocer su estructura interna, el contenido, tamaño y significado de cada cabecera en las diferentes capas del modelo OSI. Teniendo un conocimiento sólido de la estructura se puede fácilmente identificar a que flujo pertenece cada paquete capturado para poder almacenar la información relevante en un registro de flujos en memoria.

La solución propuesta supone la utilización de paquetes IP versión 4 ya que es la versión que aún se sigue utilizando en redes comerciales y aunque posteriormente será cambiada a su versión 6, todavía se va a seguir utilizando por un tiempo.

Cabe resaltar que en la solución propuesta se ha utilizado una metodología innovadora para la exportación de información de flujos desde la sonda hasta el servidor central; por tal motivo, en este capítulo se realiza un breve estudio de los principales protocolos de exportación de información de flujos incidiendo en su arquitectura que sirve como base para la implementación de una topología de red que permite extraer dicha información y enviarla hacia un servidor remoto. Es por ello, el estudio general de los principales protocolos de exportación de información de flujos es importante para el entendimiento de la metodología utilizada.

Debido a que el proceso utilizado en el servidor central de procesamiento es un procedimiento específico basado en la teoría de la información, se explican algunos conceptos básicos sobre dicha teoría los cuales permiten entender tanto la implementación de la metodología utilizada en el servidor como los resultados obtenidos (indicadores de producción de servicios).

El presente capítulo está organizado en las siguientes secciones:

- Sistemas embebidos con capacidad de conexión a redes Ethernet.
- Estructura de paquetes que contienen aplicaciones basadas en IPv4.
- Protocolos de intercambio de información de flujos de tráfico IP.
- Introducción a la teoría de la información.

2.1 Sistemas embebidos con capacidad de conexión a redes Ethernet

En esta sección se brindan los conceptos básicos de un sistema embebido, sus aplicaciones y algunos ejemplos. Posteriormente, se explica el concepto de FPGA como un dispositivo de lógica programable muy utilizado hoy en día. Asimismo, se explica en forma breve el estándar Ethernet y la estructura de las interfaces Ethernet en los sistemas embebidos. Por último, se describe la pila de protocolos TCP/IP de código libre llamada LWIP, la cual es utilizada en algunos sistemas embebidos para poder comunicarse con otros dispositivos dentro de una red Ethernet.

2.1.1 Conceptos básicos de un sistema embebido

Un sistema embebido es un sistema de cómputo diseñado para realizar tareas dedicadas por lo tanto, sirven para cubrir necesidades específicas. Es diferente a otros tipos de sistemas de cómputo tales como computadoras personales o supercomputadores.

Un sistema embebido es un sistema de cómputo con mayores requerimientos de calidad y confiabilidad que otros tipos de sistemas de cómputo [4]. Algunos de ellos con mayores requerimientos que otros. Por ejemplo los dispositivos médicos o controladores de motores de aviones requieren mayores índices de calidad y confiabilidad que dispositivos controladores de televisores, celulares o videojuegos.

La Tabla 2.1, muestra algunos ejemplos de sistemas embebidos y el segmento de mercado al que pertenecen.

Tabla 2.1 Ejemplos de sistemas embebidos y sus mercados (Fuente: [4])

Mercado	Sistema Embebido
Automotor	Sistema de ignición
	Control del motor
	Sistema de frenado
Electrónica de consumo	Televisores analógicos y digitales
	Set-Top Boxes (DVDs, VCRs, etc.)
	PDAs
	Aplicaciones de cocina (refrigeradoras, tostadoras, hornos microondas, etc.)
	Juguetes y juegos
	Teléfonos, teléfonos celulares, pagers.
	Cámaras
	GPS (Global Positioning System)
Control industrial	Sistemas de control y robótica

Médico	Bombas de dosificación de medicamentos
	Máquinas de diálisis
	Dispositivos de prótesis
	Monitores cardiacos
Redes	Enrutadores
	Concentradores
	Gateways (Puertas de enlace)
Oficina	Máquinas de fax
	Fotocopias
	Impresoras
	Monitores
	Escáneres

En esta sección se desarrollan los siguientes aspectos: arquitectura de los sistemas embebidos, procesador embebido, memoria del sistema, entradas y salidas del sistema, interrupciones, buses del procesador, drivers de los dispositivos, sistemas operativos embebidos, middleware, aplicación de software.

a. Arquitectura de los sistemas embebidos

La arquitectura de un sistema embebido incluye lo siguiente:

- Elementos propios del sistema.
- Elementos externos que interactúan con el sistema.
- Las propiedades de cada elemento individual.
- Las relaciones entre los elementos.

Un sistema embebido típico es representado modularmente por el modelo referencial que se indica en la Figura 2.1.

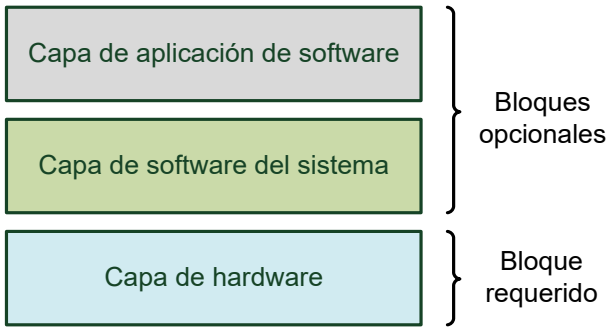


Figura 2.1 Modelo referencial de un sistema embebido (Fuente: [4])

El modelo indica que todo sistema embebido tiene como mínimo únicamente la capa de hardware. La otra opción es que implemente todas las capas (hardware, Software del sistema y aplicación de software). La capa de hardware contiene todos los principales componentes físicos mientras que las capas de software del sistema y aplicación de software contienen todo el software ubicado en el sistema y procesado por el sistema embebido.

La arquitectura de cualquier sistema embebido se puede entender fácilmente visualizando y agrupando cada uno de sus componentes según las capas del modelo

referencial.

b. Procesador embebido

El procesador es la principal unidad funcional de un sistema embebido. Es el responsable del procesamiento de instrucciones y datos. Un dispositivo electrónico contiene al menos un procesador master, actuando como dispositivo central de control y éste a su vez puede tener procesadores esclavos adicionales que son controlados por el procesador master.

Los procesadores esclavos pueden actuar como gestores de memoria, gestores de buses de datos o como dispositivos de entrada y salida. En la Figura 2.2 se muestra un diagrama de bloques de una arquitectura x86 a modo de ejemplo. De acuerdo a la figura, el procesador Atlas STPC es el procesador master y el controlador super I/O y el controlador Ethernet son procesadores esclavos.

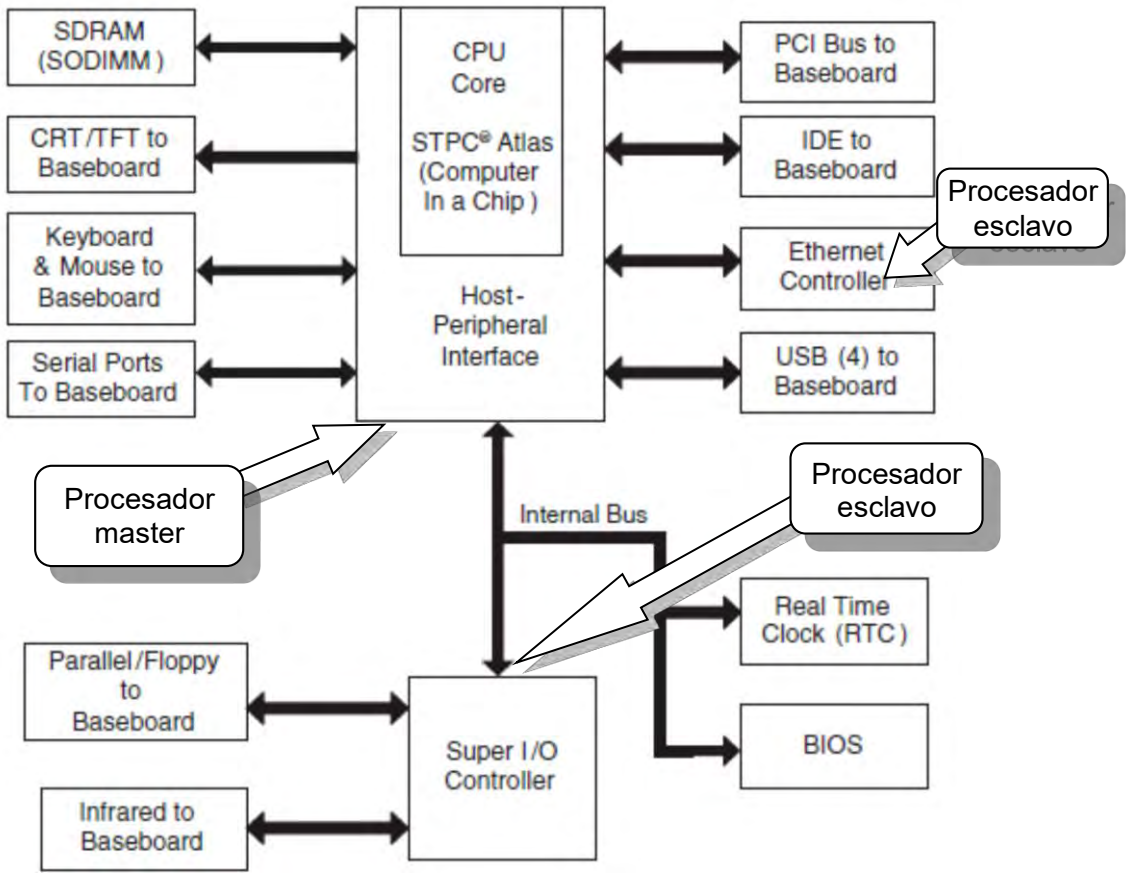


Figura 2.2 Diagrama de bloques de la tarjeta Ampro's Encore 400 (Fuente: [4])

Usualmente, el nivel de complejidad determina la clase de procesador master como cualquiera de las siguientes:

- Microprocesador: Si contiene un conjunto mínimo de memoria y componentes de entrada y salida integrados en el chip.
- Microcontrolador: Si contiene la mayor parte de memoria del sistema y componentes de entrada y salida integrados en el chip.

Los procesadores embebidos pueden ser clasificados en varios grupos según su arquitectura. Lo que los diferencia, es el conjunto de instrucciones de código máquina que los procesadores de una arquitectura específica pueden ejecutar.

El conjunto de instrucciones de una arquitectura de procesador se refiere como ISA (Instruction Set Architecture), el cual define lo siguiente:

- Las operaciones que pueden ser utilizadas por programadores para crear programas para una arquitectura en particular.
- Los operandos que son aceptados y procesados por la arquitectura.
- El almacenamiento.
- El modo de direccionamiento utilizado para acceder a los operandos y para procesarlos.
- El manejo de las interrupciones.

La arquitectura del conjunto de instrucciones es un factor determinante en la definición de las características importantes de un sistema embebido tales como el procesamiento, tiempo de diseño, funcionalidad disponible y costo.

Existen varios modelos diferentes de ISA cada una con sus propias definiciones. Los modelos ISA más comunes son: modelo específico de la aplicación, modelo de propósito general, modelo de nivel de instrucción en paralelo y otros modelos que son producto de las combinaciones de estos tres. A continuación se va a explicar únicamente el modelo ISA de propósito general.

Los modelos ISA de propósito general son implementados en procesadores destinados a ser utilizados en una amplia variedad de sistemas. Los tipos más comunes de arquitecturas ISA implementados en sistemas embebidos son:

- Modelo CISC (Complex Instruction Set Computing): Define operaciones complejas compuestas por varias instrucciones. Algunos ejemplos comunes de arquitecturas que implementan un modelo CISC son la familia de procesadores x86 de Intel y 68000 de Motorola/Freescale.
- Modelo RISC (Reduced Instruction Set Computing): A diferencia de CISC, el modelo RISC define lo siguiente:
  - Una arquitectura con un menor número de operaciones y más simples compuestas de un menor número de instrucciones.
  - Una arquitectura que tiene un número reducido de ciclos por operación. Varios procesadores RISC tienen operaciones de solamente un ciclo, mientras que en CISC las operaciones usualmente tienen múltiples ciclos.

Algunos ejemplos de arquitecturas que implementan un modelo RISC son ARM, PowerPC, SPARC y MIPS. La Figura 2.3 muestra un ejemplo de una implementación de arquitectura RISC.

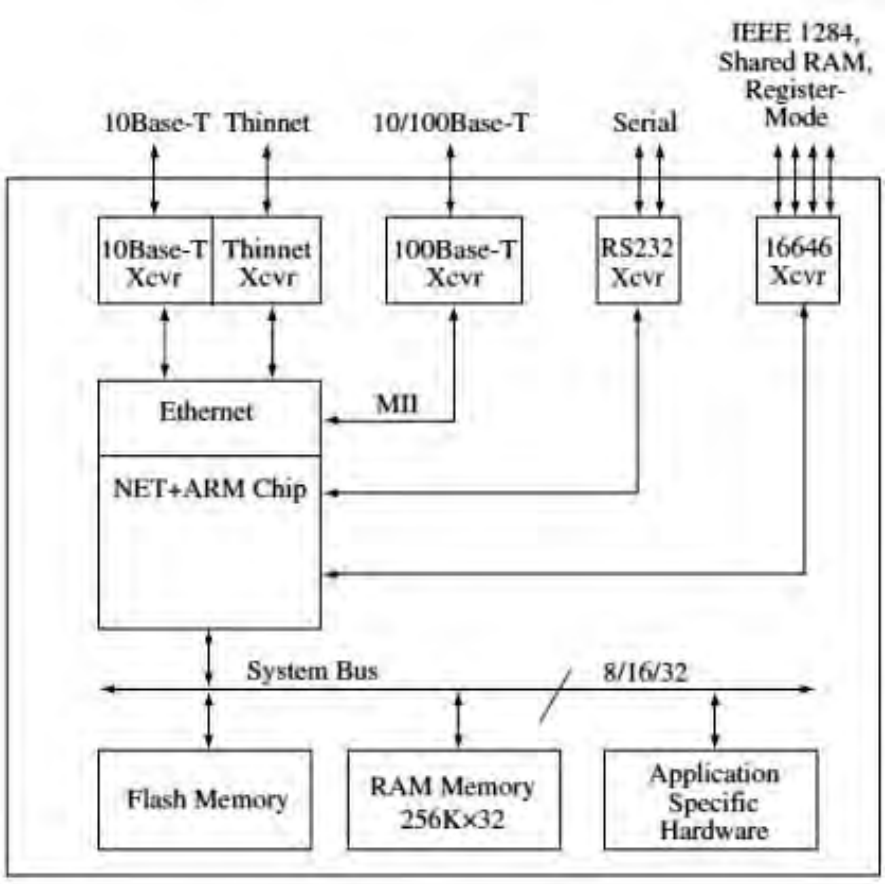


Figura 2.3 Ejemplo de implementación de una arquitectura RISC (Fuente: [4])

La mayoría de los diseños de hardware de procesadores están basados en los dos modelos de arquitectura mencionados a continuación:

- Arquitectura Von Neumann: Define un espacio simple de memoria para almacenar instrucciones y datos (Figura 2.4).

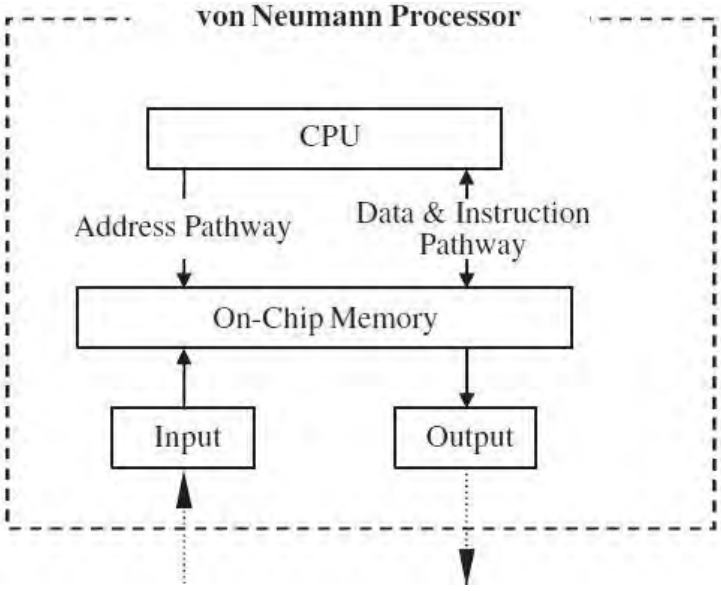


Figura 2.4 Arquitectura Von Neumann (Fuente: [4])

- Arquitectura Harvard: Es una variación de la arquitectura de Von Neumann. Define

espacios de memoria separados para instrucciones y datos. Tener buses de datos separados para instrucciones y datos permiten realizar búsquedas de instrucciones y transferencias de datos simultáneamente (Figura 2.5).

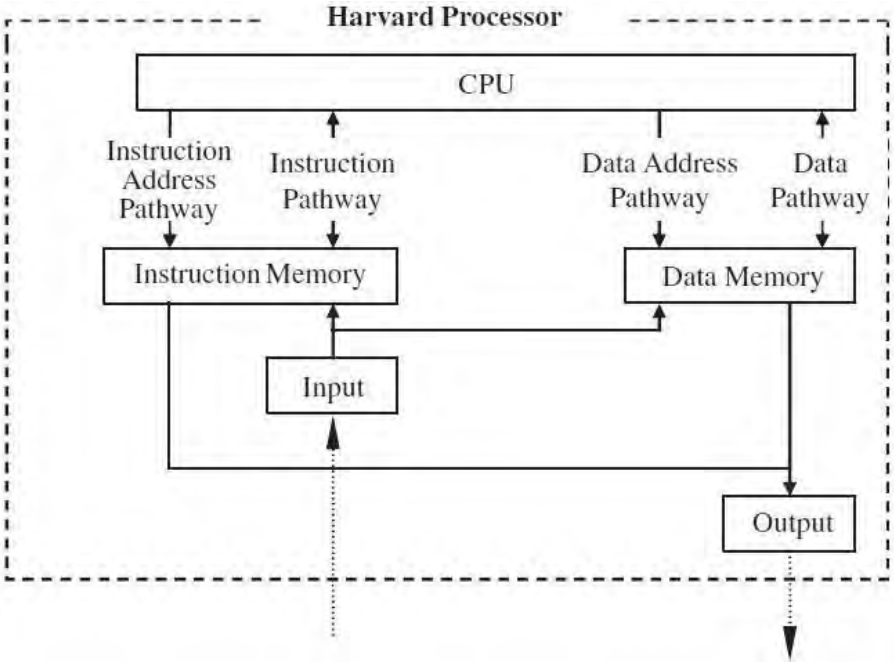


Figura 2.5 Arquitectura Von Neumann (Fuente: [4])

El CPU (Central Processing Unit) es la unidad central de procesamiento dentro de un procesador. Es el responsable de ejecutar el ciclo de búsqueda (fetch), decodificación y ejecución de instrucciones. Estos 3 procesos son referidos como pipeline de 3 estados. La Figura 2.6 ilustra la secuencia de estos 3 ciclos de un CPU.

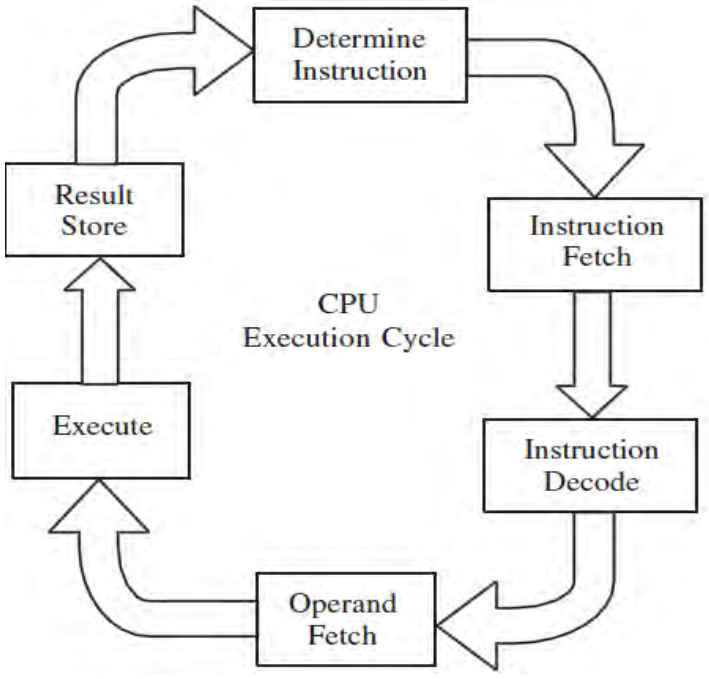


Figura 2.6 Ciclos fetch, decodificación y ejecución de un CPU (Fuente: [4])

Todos estos ciclos son implementados mediante la combinación de 4 componentes

principales, los cuales se describen a continuación.

a. ALU (Arithmetic Logic Unit)

Implementa operaciones matemáticas, lógicas y de comparación. El formato y tipos de operaciones implementados en el ALU dependen del modelo ISA. Este elemento es el responsable de aceptar múltiples operandos binarios y realizar cualquier operación lógica (AND, OR, NOT, etc.), matemática (suma, resta, multiplicación, etc.) y comparación (=, <, >, etc.) sobre dichos operandos. Es un circuito de lógica combinacional que puede tener una o más entradas y únicamente una salida.

b. Registros

Los registros son una combinación de flip-flops que pueden ser utilizados ya sea para almacenar datos temporalmente o para retardar señales. El conjunto de flip-flops puede ser activado individualmente o como un conjunto de ellos. Un registro de almacenamiento es una memoria interna programable utilizada para almacenar temporalmente, copiar y modificar operandos que son utilizados de manera inmediata o frecuentemente por el sistema. Un registro de cambio (shift register) retarda señales pasando dichas señales entre varios flip-flops internos con cada pulso de reloj.

El número de flip-flops en cada registro es utilizado para describir al procesador. Por ejemplos, un procesador de 32 bits tiene registros que tienen un ancho de 32 bits los cuales contienen 32 flip-flops, lo mismo sucede para procesadores de 16 bits, 8 bits, etc. El número de flip-flops también determina el ancho de los buses de datos utilizados en el sistema.

c. Unidad de control

La unidad de control es el principal responsable de generar señales de temporización, así como de controlar y coordinar la búsqueda (fetch), decodificación y ejecución de instrucciones en el CPU. Luego de que una instrucción ha sido buscada de la memoria y decodificada, la unidad de control determina la operación que el ALU debe realizar y a su vez selecciona y escribe señales apropiadas a cada unidad funcional dentro o fuera del CPU tales como la memoria, los registros, el ALU, etc.

d. Buses internos del CPU

Son los mecanismos que interconectan el CPU con otros componentes tales como el ALU, la unidad de control y los registros. Los buses son hilos que interconectan los diferentes componentes dentro del CPU. Cada hilo es dividido en funciones lógicas tales como:

- Datos (llevan datos entre los registros y el ALU).
- Direcciones (llevan las ubicaciones de los registros que contienen datos).
- Control (llevan señales de control entre registros, el ALU y la unidad de control).
- Otras funciones.

c. Memoria del sistema

El CPU accede a la memoria para obtener lo que necesita para procesar ya que en la memoria se almacenan todos los datos e instrucciones a ser ejecutadas por el sistema. Los sistemas embebidos tienen una jerarquía de memoria que consiste en una colección de diferentes tipos de memoria cada una con velocidades, tamaños y usos únicos.

Algunas de estas memorias pueden ser integradas dentro del procesador tales como registros, memorias de sólo lectura (ROM), algunos tipos de memorias de acceso aleatorio (RAM) y memorias caché de nivel 1. La Figura 2.7 ilustra la jerarquía de memoria en sistemas embebidos.

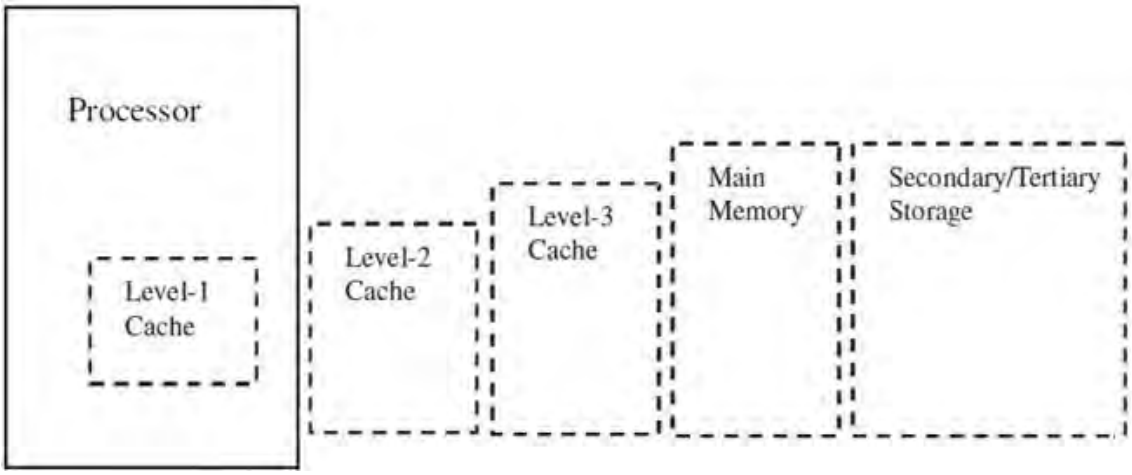


Figura 2.7 Jerarquía de memoria (Fuente: [4])

Los tipos de memoria se describen a continuación.

a. Memoria ROM (Read-Only Memory)

Es una memoria integrada en el procesador que contiene datos o instrucciones que permanecen incluso cuando no hay energía eléctrica en el sistema. Es considerada como una memoria no volátil o NVM (Non Volatile Memory). El contenido de la memoria ROM en el chip puede ser leído solamente por el sistema en donde se utiliza. Los tipos de memoria ROM más comunes son las siguientes:

- MROM (Mask ROM): Es una memoria ROM que contiene datos y que es grabada en forma permanente dentro del chip durante el proceso de fabricación y no puede ser modificada posteriormente.
- PROM (Programmable ROM): Usualmente llamadas OTP (One-Time Programmable), son un tipo de memorias ROM que pueden ser integradas dentro del chip y que son programables una sola vez por un programador de memorias PROM, es decir, puede ser programado fuera del proceso de fabricación.
- EPROM (Erasable Programmable ROM): Es una memoria ROM que puede ser integrada en el procesador y cuyo contenido puede ser borrado y reprogramado más de una vez. El número de veces de borrado y de reutilización que se pueden realizar depende del

procesador. El contenido de una memoria EPROM es escrito utilizando dispositivos especiales y es borrado ya sea de forma parcial o total utilizando otros dispositivos que emiten luz ultravioleta intensa en la ventana incorporada en el procesador.

- EEPROM (Electrically Erasable Programmable ROM): Es una memoria ROM que puede ser borrada y reprogramada más de una vez. El número de veces que se puede borrar y reutilizar depende del procesador. A diferencia de las memorias EPROM, el contenido puede ser escrito y borrado mientras el sistema embebido está en funcionamiento sin utilizar un dispositivo especial. El borrado debe hacerse en su totalidad a diferencia de las memorias EPROM las cuales pueden ser borradas parcialmente.
- Una variación más barata y más rápida que las memorias EEPROM es la memoria Flash. Las memorias EEPROM son escritas y borradas a nivel de bytes, sin embargo, las memorias Flash pueden ser escritas y borradas en bloques o sectores (grupos de bytes). Al igual que las EEPROM, las memorias Flash pueden ser borradas mientras el sistema embebido está en funcionamiento.

b. Memoria RAM (Random Access Memory)

Aludida usualmente como la memoria principal, es la memoria en la cual cualquier ubicación puede ser accedida directamente de manera aleatoria en vez de un acceso secuencial desde algún punto de inicio. Su contenido puede ser alterado más de una vez (el número de veces depende del hardware). El contenido de la memoria RAM es borrado si se pierde la energía eléctrica que la alimenta, esto significa que es una memoria volátil. Los dos tipos principales de memoria RAM son:

- RAM Estática ó SRAM (Static RAM): Compuesta por flip-flops basado en transistores. La carga en las memorias SRAM se mantiene hasta que elimina la energía o los datos son sobreescritos. Son más lentas que los registros pero más rápidas que las memorias DRAM. Consume menos energía que las DRAM ya que no requiere energía extra para su refresco. Debido a que son costosas son utilizadas en menores cantidades por ejemplo como memorias caché externas y memorias de video para el procesamiento de ciertos tipos de gráficos.
- RAM dinámica ó DRAM (Dynamic RAM): Compuesta por un arreglo de capacitores que no pueden mantener carga (datos). La carga se disipa gradualmente con el tiempo por tal motivo requieren un mecanismo de refresco para mantener la integridad de los datos. El mecanismo de lectura de una celda de la memoria descarga los capacitores. Usualmente un controlador de memoria administra la recarga y descarga de la DRAM. Son más lentas y más baratas que las memorias SRAM. Pueden almacenar más datos que una SRAM ya que su circuitería es más pequeña. Usualmente es la memoria principal y también es utilizado como memoria RAM de video y como memoria caché.

c. Memoria cache de nivel 1

Es el nivel de memoria entre el CPU y la memoria principal en la jerarquía de memoria (ver Figura 2.7). Puede ser integrada dentro o fuera del chip. La memoria caché dentro del procesador es aludida como caché de nivel 1 y las memorias SRAM son utilizadas como tales. Debido a que las memorias SRAM son veloces y costosas, los procesadores tienen una pequeña cantidad de caché ya sea en el chip o fuera del chip. Son utilizadas para almacenar un subconjunto de datos de la memoria principal los que son utilizados y accedidos con mayor frecuencia. Algunos procesadores tienen una caché tanto para instrucciones como para datos mientras que otros tienen dos cachés separadas dentro del chip para cada uno.

e. Gestión de la memoria dentro del chip

Dentro de un sistema pueden integrarse diferentes tipos de memorias las cuales tienen diferentes maneras de ser manejadas por el software que se ejecuta en el CPU especialmente su direccionamiento como direcciones virtuales ó lógicas y las direcciones físicas reales. Los gestores de memoria son circuitos integrados diseñados para gestionar estas cuestiones. Los dos tipos más comunes de gestores de memoria integrados en un procesador master son:

- MEMC (Memory Controllers): Implementa y proporciona interfaces a los diferentes tipos de memorias en el sistema tales como caché, SRAM y DRAM. Sincroniza los accesos a las memorias y verifica la integridad de los datos que están siendo transferidos. Estos controladores acceden directamente a la memoria utilizando su propia dirección física. Gestiona las consultas desde el procesador y accede a los bancos apropiados para esperar una respuesta y enviársela al procesador.
- MMU (Memory Management Unit): Se utiliza para traducir direcciones lógicas en direcciones físicas (mapeo de memoria) así como para manejar la seguridad de la memoria, controlar la caché, manejar el arbitraje del bus entre el CPU y la memoria y generar las excepciones adecuadas.

e. Organización de la memoria

La organización de la memoria incluye no solamente la jerarquía de la memoria de una plataforma en particular, sino también la organización interna de la memoria tal como la definición de las diferentes porciones de memoria que pueden o no pueden ser utilizadas así como la manera en que todos los diferentes tipos de memoria son organizados y accedidos por el resto del sistema. Por ejemplo, algunas arquitecturas dividen la memoria con la finalidad de que una porción almacene sólo instrucciones y otra porción sólo datos. El procesador master junto con el software trata a la memoria como un amplio arreglo unidimensional llamado Memory Map (Figura 2.8).

Address Offset	Register	Size
000	SIU module configuration register (SIUMCR)	32 bits
004	System Protection Control Register (SYPCR)	32 bits
008-00D	Reserved	6 bytes
00E	Software Service Register (SWSR)	16 bits
010	SIU Interrupt Pending Register (SIPEND)	32 bits
014	SIU Interrupt Mask Register (SIMASK)	32 bits
018	SIU Interrupt Edge/Level Register (SIEL)	32 bits
01C	SIU Interrupt Vector Register (SIVEC)	32 bits
020	Transfer Error Status Register (TESR)	32 bits
....	....	....

Figura 2.8 Ejemplo de registros dentro de un mapa de memoria (Fuente: [4])

Este mapa sirve para definir claramente las direcciones o conjunto de direcciones que son ocupadas por los diversos componentes del sistema. Se puede definir espacios de direcciones múltiples accesibles a solamente cierto tipo de información.

d. Entradas y salidas del sistema

Los componentes de entrada y salida (Figura 2.9) son los responsables de mover información hacia y desde otros componentes del procesador y desde y hacia cualquier memoria o dispositivo de entrada/salida fuera del procesador.

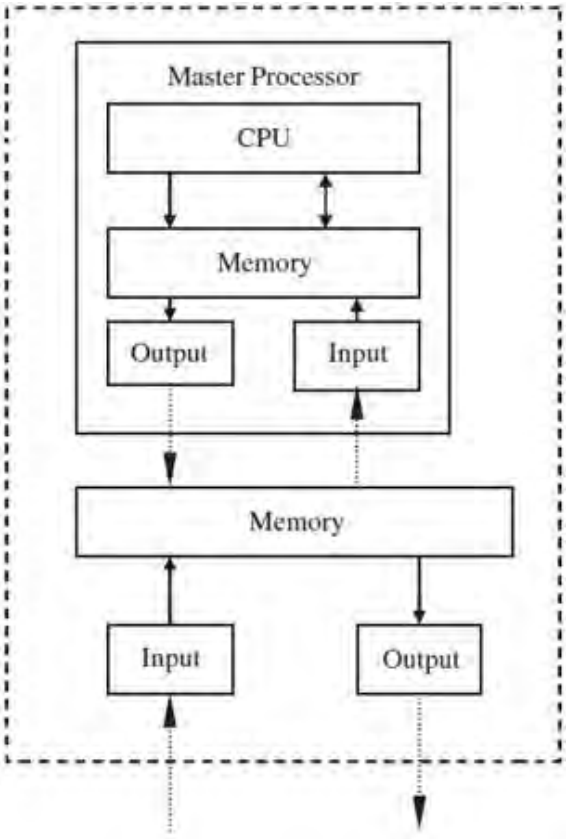


Figura 2.9 Diagrama de entrada/salida del procesador (Fuente: [4])

Las entradas y salidas del procesador pueden ser:

- Componentes de entrada que solamente brinda información al procesador.

- Componentes de salida que brinda información hacia fuera del procesador.
- Componentes que realiza ambas funciones.

Los dispositivos de entrada/salida pueden ser divididos en los siguientes grupos:

- Dispositivos de entrada: Transmite datos hacia los componentes de entrada/salida del sistema tales como un mouse, teclado o control remoto
- Dispositivos de salida: Reciben datos desde los componentes de entrada/salida del sistema y muestran los datos de diversas formas tales como imprimiendo en papel, en una pantalla o parpadeando un diodo LED.
- Dispositivos de entrada/salida: Pueden realizar ambas funciones y pueden ser conectados al sistema mediante medios de transmisión cableados o inalámbricos o pueden también estar ubicados dentro de la tarjeta donde se encuentra el sistema embebido. Un ejemplo es un dispositivo de red el cual transmite datos desde y hacia internet.

Un hardware de entrada/salida está compuesto de las combinaciones de alguno de las seis principales unidades lógicas siguientes:

- El medio de transmisión cableado o inalámbrico que conecta el dispositivo de entrada/salida con la tarjeta electrónica donde se encuentra el procesador.
- Un puerto de comunicación.
- Una interfaz de comunicación integrada o no en el procesador, el cual gestiona la comunicación de datos entre el CPU y el dispositivo de entrada/salida o el controlador de entrada/salida.
- Un controlador de entrada/salida el cual es un procesador esclavo que gestiona el dispositivo de entrada/salida.
- Buses de entrada/salida el cual consiste en la conexión entre el procesador master y el dispositivo de entrada/salida dentro de la tarjeta electrónica.
- El procesador master de entrada/salida integrado.

Las entradas y salidas del procesador pueden ser organizadas en diversas categorías según las funciones que realizan. Estas categorías incluyen:

- Redes y comunicaciones (capa física del modelo OSI.)
- Entrada (teclado, mouse, control remoto, voz, etc.)
- Gráficos y salida (pantallas táctiles, CRT, impresoras, diodos LEDs, etc.)
- Almacenamiento (controladores de discos ópticos, controladores de discos magnéticos, etc.)
- Depuración (BDM, JTAG, puerto serial, puerto paralelo, etc.)
- Tiempo real y diversos (temporizadores/contadores, conversores análogo a digital y digital a analógico, interruptores, etc.)

e. Interrupciones

Las interrupciones son señales disparadas por algún evento durante la ejecución de una instrucción por el procesador master. Puede ser iniciado de manera asíncrona por dispositivos externos, reinicios, fallas de energía eléctrica o iniciados de manera síncrona por llamadas al sistema o instrucciones ilegales. Estas señales causan que el procesador detenga la ejecución del actual flujo de instrucciones e inicie el proceso de manejo o procesamiento de la interrupción. Los tres principales tipos de instrucciones son:

- Interrupciones de software: Son disparadas internamente de manera explícita por algunas instrucciones dentro del flujo actual de instrucciones que están siendo ejecutadas por el procesador master.
- Interrupciones de hardware interno: Son iniciadas por algún evento que es resultado de un problema con el actual flujo de instrucciones que está siendo ejecutado por el procesador master. Esto se debe a limitaciones del hardware tales como operaciones matemáticas ilegales como la división por cero, depuración, instrucciones o códigos de operación no válidos, etc. Las interrupciones que son levantadas por algún evento interno del procesador master, tales como interrupciones de hardware interno o de software, se llaman excepciones o traps dependiendo del tipo de interrupción.
- Interrupciones de hardware externo: Son iniciadas por un hardware diferente al CPU tales como los buses de la tarjeta electrónica, dispositivos de entrada/salida, etc. Lo que dispara la interrupción está determinado por el software mediante los bits de ciertos registros que activan o desactivan fuentes de interrupciones en el código de inicialización del controlador del dispositivo.

El procesador master cuenta con pines o puertos de entrada llamados IRQ (Interrupt Request Level) para interrupciones que son disparadas por eventos externos. Estos puertos están conectados ya sea a un hardware intermediario comúnmente llamado controlador de interrupciones o directamente hacia otros componentes en la tarjeta electrónica.

Un controlador de interrupciones puede ser integrado en la tarjeta o dentro del procesador. Se encarga de arbitrar las transacciones de interrupciones conjuntamente con el software.

Para los sistemas sin controlador de interrupciones las líneas de petición de interrupción del dispositivo son conectadas directamente al procesador master y las transacciones son controladas por software y algunos componentes internos tales como registros, contadores, etc.

f. Buses del procesador

Los buses del procesador interconectan los principales componentes internos del procesador tales como el CPU, la memoria y las interfaces de entrada/salida. Llevan

señales entre los diferentes componentes. El ancho del bus es el número de bits que pueden ser transmitidos a la vez. La velocidad de cada bus es expresado en MHz el cual tiene un impacto en el rendimiento del procesador.

g. Drivers de los dispositivos

La mayoría de los dispositivos de hardware embebidos requieren algún tipo de software de inicialización y de gestión. El software que controla directamente un hardware es llamado driver del dispositivo.

Todo sistema embebido que requiere un software tiene por lo menos un driver en la capa de software del sistema. Los drivers de dispositivos son bibliotecas que inicializan el hardware y gestionan los accesos a las capas superiores de software. Los drivers son los enlaces entre el hardware y las capas de sistema operativo, middleware y aplicaciones.

En la Figura 2.10, se muestra en el modelo de capas de un sistema embebido resaltando la ubicación de la capa de driver de un dispositivo dentro de la capa de software del sistema.

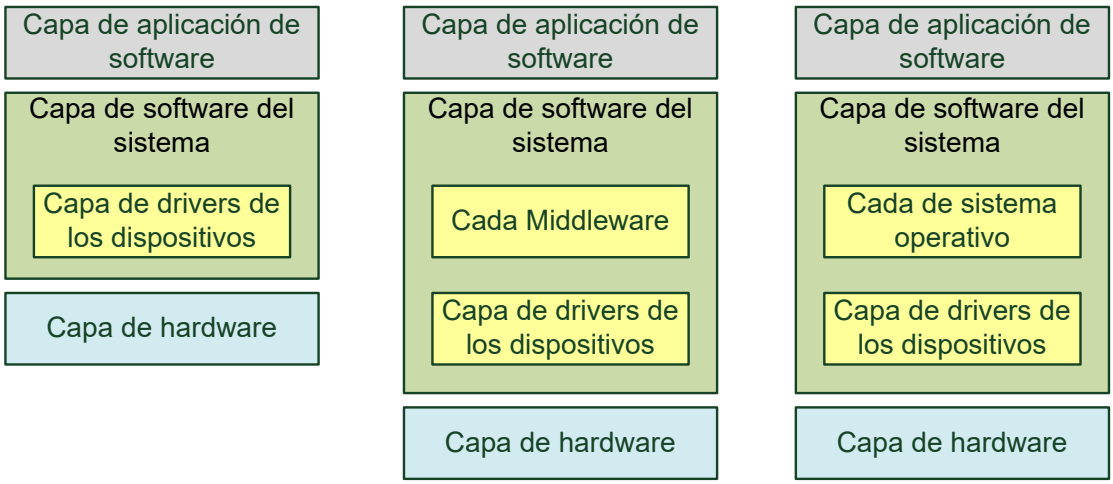


Figura 2.10 Modelos de sistemas embebidos con drivers de dispositivos

(Fuente: [4])

Los drivers son considerados como cualquiera de los 2 siguientes tipos:

- Drivers genéricos: Gestiona un hardware que está ubicado en la tarjeta electrónica y que no está integrado en el procesador master. Incluye código que inicializa y gestiona los accesos a los demás componentes principales de la tarjeta electrónica tales como los buses de la tarjeta como I2C, PCI, PCMIA, etc. las memorias externas al procesador como controladores de memoria, caché de nivel 2, memorias flash, etc. e interfaces de entrada/salida externas al procesador como Ethernet, RS232, pantalla, mouse, etc.
- Específicos según la arquitectura: Gestiona el hardware que está integrado en la arquitectura del procesador master tales como la memoria integrada, MMU integrados y unidades de punto flotante en hardware.

Todo driver cumple todas o algunas de las siguientes funciones:

- Inicializar el hardware al momento de encendido o reinicio.
- Detener el hardware configurando en su estado de apagado.
- Inhabilitar el hardware cuando está en funcionamiento.
- Obtener acceso restringido al hardware.
- Liberar o desbloquear el hardware.
- Leer datos del hardware.
- Escribir datos en el hardware.
- Instalar nuevo hardware cuando está en funcionamiento.
- Desinstalar o remover el hardware cuando está en funcionamiento.

h. Sistemas operativos embebidos

Un sistema operativo es una parte opcional de la pila de software de un sistema embebido. Pueden ser utilizados en cualquier arquitectura de procesador. Según se muestra en la Figura 2.11, un sistema operativo puede estar ubicado ya sea sobre el hardware, sobre la capa de driver de dispositivo o sobre un BSP (Board Support Package).

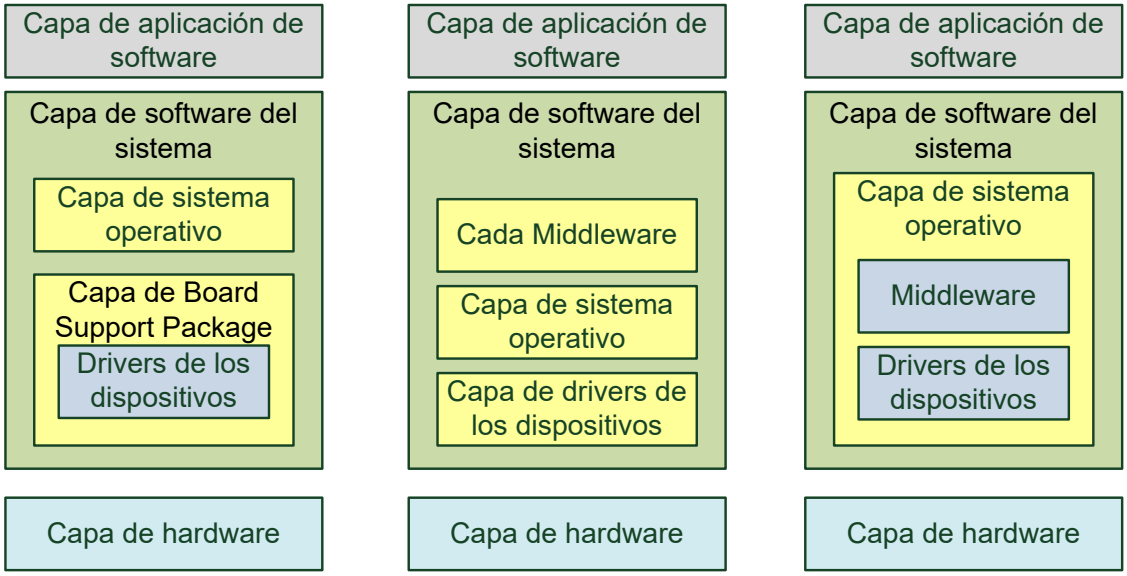


Figura 2.11 Sistemas operativos en el modelo de sistemas embebidos

(Fuente: [4])

El sistema operativo es un conjunto de bibliotecas de software las cuales sirven para dos propósitos principales en un sistema embebido:

- Proporcionan una capa de abstracción al software que se encuentra sobre el sistema operativo para que sea menos dependiente del hardware. Esto facilita el desarrollo de middleware y aplicaciones que se encuentran por encima del sistema operativo.
- Gestiona los diferentes recursos de hardware y software para asegurar que el sistema opere de manera eficiente y confiable.

Todos los sistemas operativos tienen por lo menos un kernel el cual es un componente que contiene la funcionalidad principal del sistema operativo. Contiene alguna o todas las características que se muestra en la Figura 2.12.

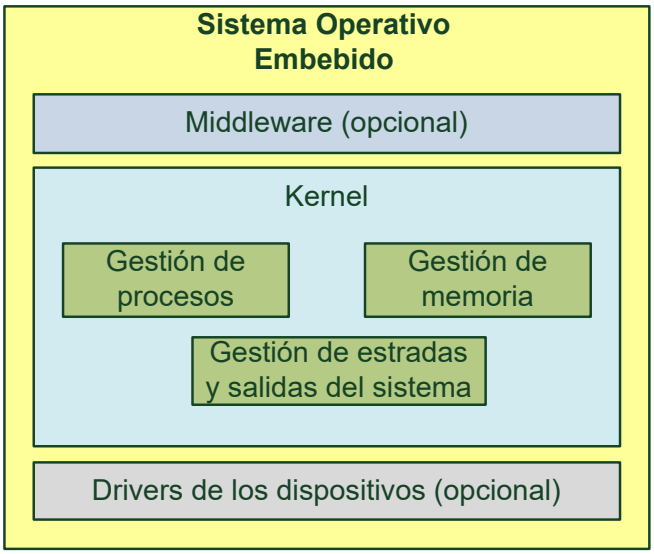


Figura 2.12 Modelo general de un sistema operativo (Fuente: [4])

- Gestión de procesos: Se gestiona la manera de observar otro software en el sistema embebido mediante procesos. Tiene funciones secundarias de gestión de interrupciones y detección de errores. Las múltiples interrupciones son gestionadas de manera eficiente con la finalidad de que los procesos que las generan son rastreados adecuadamente.
- Gestión de la memoria: El espacio de memoria del sistema embebido es compartido por todos los diferentes procesos por tal motivo, el acceso y la asignación de porciones de memoria son gestionados por el sistema operativo.
- Gestión de entrada/salida del sistema: Los dispositivos de entrada/salida son compartidos entre varios procesos y por lo tanto, las asignaciones y los accesos de un dispositivo de entrada/salida son gestionados por el sistema operativo.

Un programa es simplemente una secuencia de instrucciones pasiva y estática que puede representar los recursos de hardware y software. La ejecución de un programa es un evento activo y dinámico en el cual varias propiedades cambian relativamente con el tiempo y con las instrucciones que están siendo ejecutadas.

Un proceso o tarea es creado por un sistema operativo para encapsular toda la información que se relaciona con la ejecución de un programa. Un programa es solamente una parte de una tarea tal como se muestra en la Figura 2.13.

Los sistemas operativos embebidos gestionan todo el software utilizando tareas y según esta característica, los sistemas operativos pueden ser:

Figura 2.13 Ejemplo de una tarea de un sistema operativo (Fuente: [4])

- Unitasking (una sola tarea): Solamente puede existir una tarea al mismo tiempo. No requieren una gestión de tareas compleja.

- Multitasking (múltiples tareas): Puede existir múltiples tareas simultáneamente. Requieren una gestión de procesos compleja. Esto requiere que cada proceso permanezca independiente de los demás.

i. Middleware

Para un sistema embebido un middleware es un sistema de software que se ubica en el modelo de capas ya sea sobre la capa de drivers de dispositivos o sobre el sistema operativo tal como se muestra en la Figura 2.14 y puede estar o no estar incorporado dentro del sistema operativo.

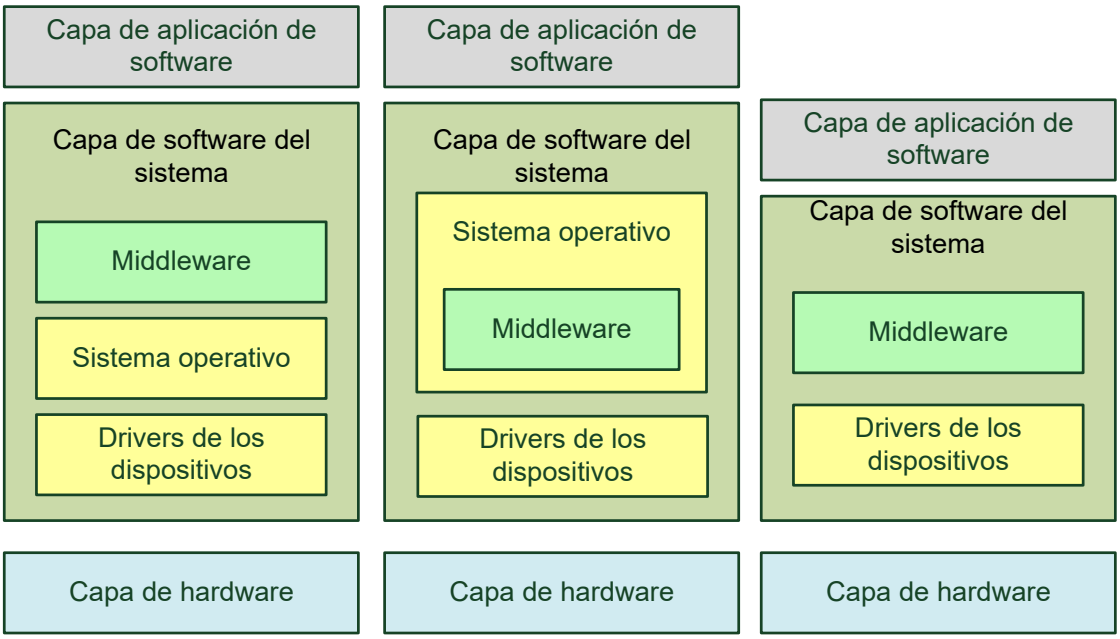


Figura 2.14 Ubicación del Middleware dentro del modelo de sistemas embebidos (Fuente: [1])

El Middleware es un software que arbitra entre la aplicación de software y el kernel o los drivers de dispositivos. Es una capa de abstracción utilizada en los sistemas embebidos con dos o más aplicaciones para proporcionar flexibilidad, seguridad, portabilidad, conectividad, intercomunicación y/o mecanismos de interoperatividad entre las aplicaciones. Sin embargo, la introducción de un middleware en el sistema genera un

impacto en la escalabilidad y el rendimiento y en general en todas las capas de un sistema embebido.

j. Aplicación de software

La aplicación de software de un sistema embebido se ubica sobre la capa de software del sistema tal como se muestra en la Figura 2.15. La aplicación es gestionada y ejecutada por dicha capa.

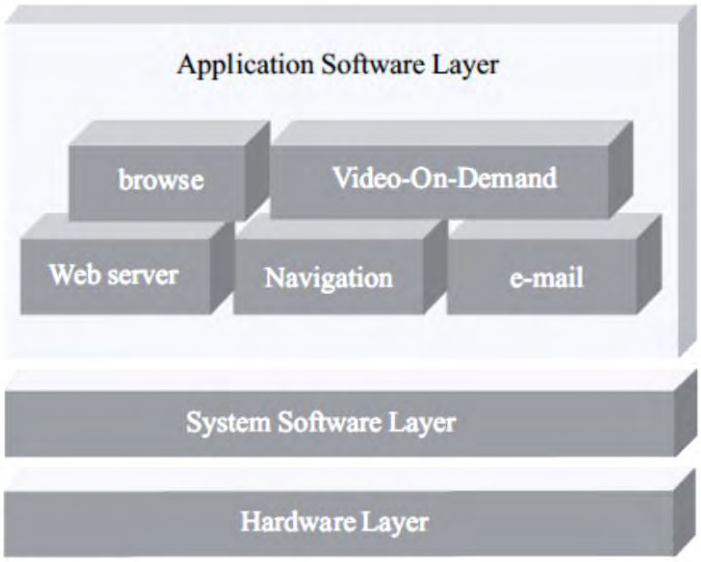


Figura 2.15 Capa de aplicación de software en el modelo de sistemas embebidos (Fuente: [4])

El software de aplicación define el tipo de sistema embebido ya que la funcionalidad de una aplicación representa el propósito del sistema embebido y realiza la mayoría de las interacciones con los usuarios del dispositivo.

Las aplicaciones embebidas pueden ser divididas según si son para un mercado específico las cuales son implementadas solamente en ciertos tipos de dispositivos o de propósito general las cuales pueden ser implementadas en diferentes tipos de dispositivos.

2.1.2 FPGA

Un FPGA (Field Programmable Gate Array) es un chip de silicio reprogramable. Utiliza bloques de lógica prefabricados (celdas lógicas) y recursos para ruteo programables (switches programables). Un FPGA puede ser configurado para implementar diferentes funcionalidades personalizadas en hardware. Asimismo puede realizar tareas de cómputo en software.

Esta subsección está organizada en los siguientes ítems: características de un FPGA, estructura Interna de una FPGA, celda lógica basada en tablas de búsqueda, macro celda, diseño en FPGA, y beneficios del uso de FPGA.

a. Características de un FPGA

Un FPGA Presenta las siguientes características:

- Son completamente reconfigurables y pueden cambiar de función de manera instantánea cuando se compila una diferente configuración de la circuitería interna.
- Combinan lo mejor de los ASIC (Application-specific Integrated Circuit) y de los sistemas basados en procesadores.
- Ofrecen velocidades temporizadas en hardware y alta fiabilidad.
- A diferencia de los procesadores, los FPGA pueden ejecutar operaciones de manera paralela, por lo que cada operación no necesita competir por los mismos recursos.
- Cada tarea o proceso es independiente, utiliza una sección dedicada del chip y puede ser ejecutado de manera autónoma sin afectar los otros bloques de lógica.

b. Estructura Interna de una FPGA

Un FPGA (Field Programmable Gate Array) es un dispositivo lógico que contiene un arreglo de dos dimensiones de celdas lógicas genéricas y switches programables [5]. Una celda lógica puede ser programada para realizar una función simple y los switches programables pueden ser personalizados para proporcionar interconexiones entre las celdas lógicas. La estructura conceptual de un FPGA se muestra en la Figura 2.16.

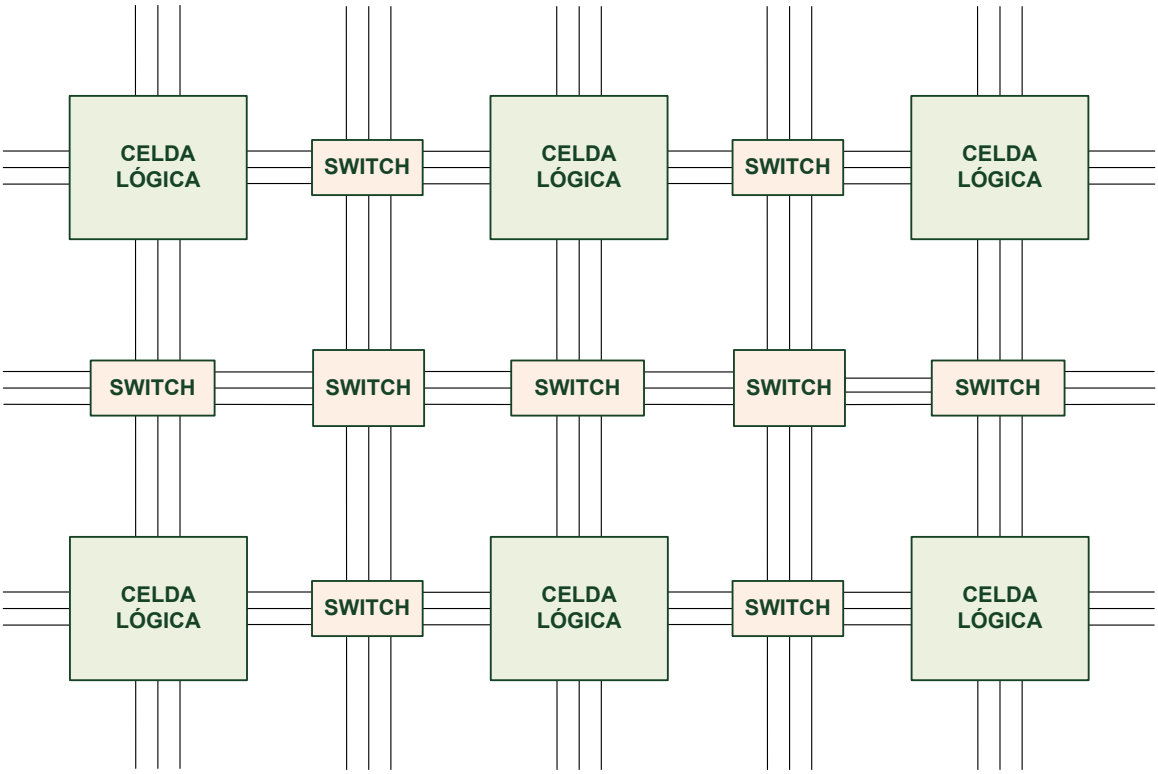
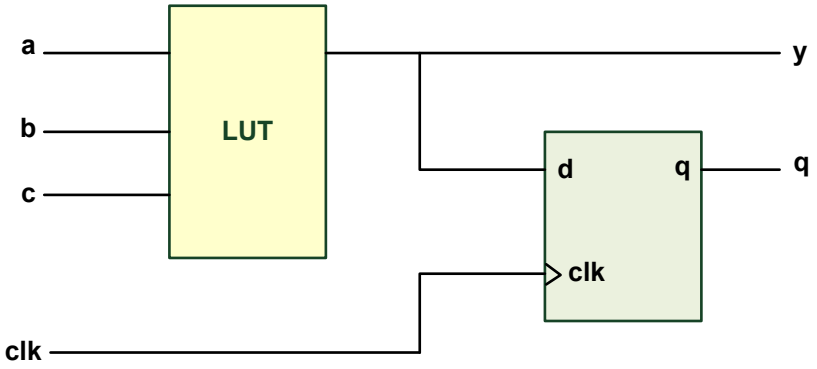


Figura 2.16 Estructura conceptual de un dispositivo FPGA (Fuente: [5])

c. Celda lógica basada en Tablas de búsqueda

Una celda lógica contiene un pequeño circuito combinacional configurable con un flips-flop tipo D. La mayoría de las implementaciones de un circuito combinacional configurable es con LUT (Look-up Table). Un LUT puede ser considerado como una memoria pequeña de 2^nx1. Se utiliza para implementar cualquier función combinacional de n entradas. En la

Figura 2.17, se muestra un diagrama conceptual de una celda lógica basada en un LUT de 3 entradas. Se muestra además un ejemplo de la implementación de la función lógica “a ⊕ b ⊕ c”. La salida del LUT puede ser almacenada en el flip-flop tipo D, el cual puede ser utilizado para implementar circuitos secuenciales.



a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Figura 2.17 Ejemplo de una celda lógica basada en un LUT de 3 entradas (Fuente: [5])

d. Macro celda

La mayoría de los dispositivos FPGA contienen macro celdas o macro bloques los cuales son diseñados y fabricados a nivel de transistor y sus funcionalidades complementan las celdas lógicas genéricas. Las macro celdas comúnmente utilizadas incluyen bloques de memoria, multiplicadores combinacionales, circuitos de gestión de reloj y circuitos de interfaces de entrada y salida. Los dispositivos FPGA avanzados incorporan uno o más núcleos de procesador prefabricados.

e. Diseño en FPGA

Un diseño personalizado puede ser implementado especificando la función de cada celda lógica y estableciendo las conexiones de cada switch programable. Una vez que la fase de diseño y síntesis es completada, se utiliza un cable adaptador para descargar el diseño al dispositivo FPGA y obtener el circuito personalizado.

f. Beneficios del uso de FPGA

Un FGPA presenta las siguientes ventajas y beneficios [6]:

- Alto Rendimiento: Los FPGAs exceden la potencia de cómputo de los DSP (Digital Signal Processing) ya que usualmente realizan ejecución de procesos en forma paralela y por lo tanto ejecutan más procesos en cada ciclo de reloj.
- Rápido desarrollo de prototipos: La tecnología de las FPGAs ofrece alta flexibilidad y rápidos tiempos de desarrollo de prototipos lo que agiliza el tiempo de salida al mercado. Se dispone de varias herramientas de software de alto nivel para el diseño de FPGA con varios niveles de abstracción las cuales disminuyen la curva de aprendizaje. Usualmente, estas herramientas incluyen núcleos IP (funciones prediseñadas) que agilizan el proceso

de diseño.

- Alta fiabilidad: Los circuitos de un FPGA son una implementación segura para la ejecución de un programa. Usualmente, los sistemas basados en procesadores implican varios niveles de abstracción que ayudan a programar las tareas y compartir los recursos entre los múltiples procesos. El núcleo de un procesador sólo puede ejecutar una instrucción a la vez y se presenta el riesgo de que las tareas se obstruyan entre sí. A diferencia de los sistemas basados en procesadores, los FPGAs no necesitan sistemas operativos lo que aumenta la fiabilidad de estos dispositivos con la ejecución paralela y un hardware preciso para cada tarea.
- Fácil mantenimiento a largo plazo: Los FPGA son capaces de mantenerse al tanto de las modificaciones a futuro que pudieran ser necesarias. Se pueden implementar mejoras funcionales durante el desarrollo del sistema sin necesidad de invertir tiempo rediseñando el hardware o modificando el diseño de la tarjeta de circuito impreso.

2.1.3 Descripción de la pila de protocolos LWIP

En esta sección se describe el stack TCP/IP llamado lwIP el cual es suficientemente pequeño para ser utilizado en sistemas con limitados recursos computacionales y de memoria como los sistemas embebidos [7].

a. Definición

LwIP es una implementación ligera del conjunto de protocolos TCP/IP el cual es utilizado en varios productos comerciales. Esta implementación ha sido embebida en diferentes plataformas y puede ser ejecutado con un sistema operativo o sin él. El enfoque de esta implementación es reducir el uso de la memoria RAM teniendo a la vez todo el conjunto de protocolos TCP/IP. Esta característica hace que lwIP sea apropiado para ser implementado en sistemas embebidos con decenas de kilobytes de memoria RAM libre y alrededor de 40 kilobytes de código en memoria ROM.

Las principales características de lwIP son las siguientes:

- Incorpora el protocolo IP (Internet Protocol) incluyendo el reenvío de paquetes sobre múltiples interfaces de red.
- Incorpora el protocolo ICMP (Internet Control Message Protocol) para el mantenimiento y depuración de la conexión de red del sistema.
- Incorpora el protocolo UDP (User Datagram Protocol) para el envío y recepción de datagramas.
- Incorpora el protocolo TCP (Transmission Control Protocol) con control de congestión y estimación del RTT (Round Trip Time).

LwIP consiste de las siguientes partes:

- Implementación de los protocolos TCP/IP (IP, ICMP, UDP y TCP).

- Capa de emulación de sistema operativo.
- Subsistema de gestión de buffer y memoria.
- Funciones para la interfaz de red.
- Funciones para el cálculo del checksum de Internet.

b. Capa de emulación de sistema operativo

Las llamadas a funciones específicas del sistema operativo y las estructuras de datos no son utilizadas directamente en el código. La capa de emulación proporciona una interfaz uniforme a servicios del sistema operativo tales como:

- Temporizadores utilizados para el protocolo TCP.
- Mecanismo de sincronización de procesos mediante semáforos.
- Mecanismos de transferencia de mensajes.

Cuando se desea embeber la pila lwIP en cualquier sistema operativo, sólo es necesario contar con una implementación de la capa de emulación para el respectivo sistema operativo.

c. Subsistema de gestión de buffer y memoria

Este subsistema se encarga de alojar buffers de tamaños variables. Asimismo, se encarga de evitar copias de datos dejando el contenido de datos en buffers que residen en memoria que no es gestionada por el subsistema de red.

c.1. Buffer de paquete

Un buffer de paquete es una representación interna de un paquete en lwIP. Se representa por una estructura llamada “pbuf” la cual tiene soporte para permitir que los datos del paquete residan tanto en memoria dinámica como en memoria estática. Los buffers pueden ser enlazados en una lista llamada cadena de buffers de paquetes lo cual permite que un paquete pueda abarcar varios buffers de paquete. Se tienen tres tipos de buffers los cuales son descritos en la Tabla 2.2.

Tabla 2.2 Tipos de buffers en lwIP (Fuente: [7])

Nombre del Buffer	Descripción
PBUF_RAM	El dato del paquete es almacenado en memoria gestionada por el subsistema de buffers de paquetes. Almacena las cabeceras que son antepuestas a los datos. Son encadenadas a un buffer PBUF_ROM. Utilizado cuando una aplicación envía datos que son generados de forma dinámica.
PBUF_ROM	El dato del paquete es almacenado en memoria que no es gestionada por el subsistema de buffers de paquetes. Son utilizados cuando una aplicación envía datos que están localizados en memoria gestionada por la aplicación. Utilizados cuando el dato está localizado

	en memoria ROM.
PBUF_POOL	Consiste en buffers de tamaño fijo. Utilizado por controladores de dispositivos de red debido a que la asignación de este buffer es rápida y por tal motivo es utilizado en un manejador de interrupciones.

Los paquetes entrantes son almacenados en buffers del tipo PBUF_POOL y los paquetes salientes son almacenados en buffers del tipo PBUF_ROM o PBUF_RAM.

La estructura interna de un buffer de paquete consiste de los siguientes elementos:

- El campo next es un puntero al siguiente buffer en el caso de un buffer en cadena.
- El puntero “payload” apunta al inicio de los datos en el buffer.
- El campo “len” contiene la longitud de los datos contenidos en el buffer.
- El campo “tot_len” contiene la suma de la longitud del actual buffer y todas las longitudes de todos los campos “len” en un buffer en cadena.
- El campo “flags” indica el tipo de buffer.
- El campo “ref” contiene un contador referencial.

La Figura 2.18 muestra la estructura interna de un buffer de paquete en lwIP.

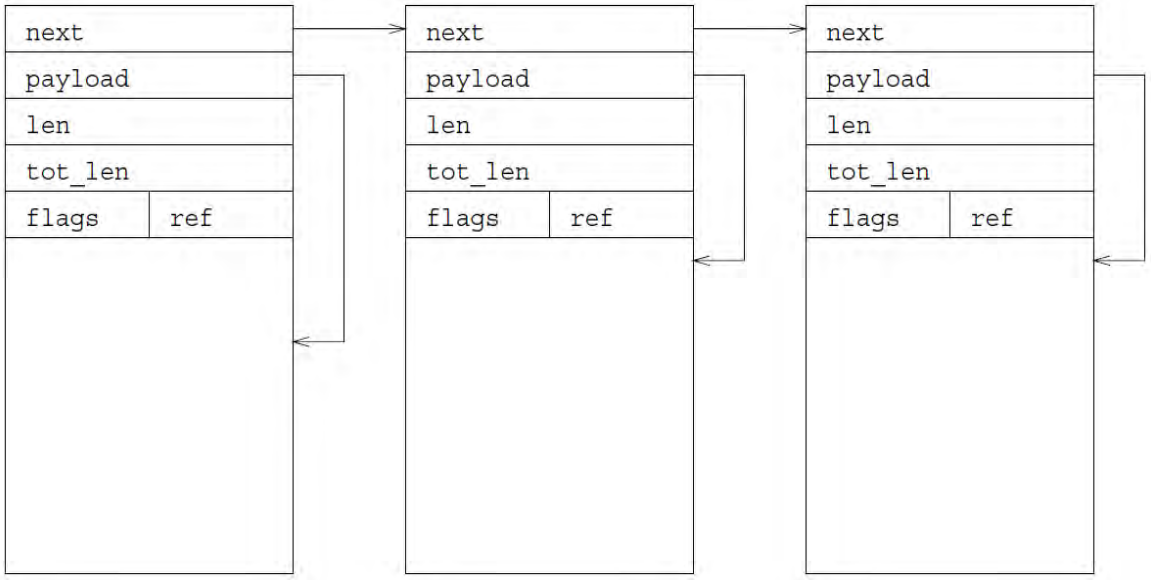


Figura 2.18 Estructura interna de buffers de paquetes (Fuente: [7])

El tamaño total de la estructura de un buffer de paquete depende del tamaño de un puntero en la arquitectura del procesador del sistema.

c.2. Gestión de memoria

La gestión de memoria maneja las asignaciones y no asignaciones de regiones contiguas de memorias y puede reducir el tamaño de un bloque de memoria previamente asignado. El gestor de memoria utiliza una porción dedicada del total de memoria del sistema.

d. Interfaz de red

En lwIP los drivers para la interfaz física de red se representan por estructuras de interfaz de red llamadas “netif”. La Figura 2.19 ilustra el código que define a una estructura de interfaz de red.

```
struct netif {
  struct netif *next;
  char name[2];
  int num;
  struct ip_addr ip_addr;
  struct ip_addr netmask;
  struct ip_addr gw;
  void (* input)(struct pbuf *p, struct netif *inp);
  int (* output)(struct netif *netif, struct pbuf *p,
                 struct ip_addr *ipaddr);
  void *state;
};
```

Figura 2.19 Estructura de una interfaz de red en lwIP (Fuente: [7])

- El campo “name” es utilizado para almacenar un nombre a la interfaz de red.
- El campo “num” es utilizado para distinguir diferentes interfaces del mismo tipo.
- Las direcciones ip_addr (dirección IP), netmask (máscara de subred) y gw (gateway) son utilizadas por la capa IP cuando envía y recibe paquetes.
- El puntero “input” apunta a la función del driver del dispositivo la cual es llamada cuando se recibe un paquete.
- El puntero “output” apunta a la función del driver del dispositivo la cual es llamada para transmitir un paquete hacia la red física. Esta función es llamada por la capa IP cuando un paquete va a ser enviado.
- El puntero “state” apunta al estado específico del driver del dispositivo de interfaz de red y es establecido por el driver del dispositivo.

e. Procesamiento de IP

LwIP solamente implementa la funcionalidad básica del protocolo IP. Puede enviar, recibir y reenviar paquetes pero no se puede enviar o recibir paquetes IP fragmentados ni manejar paquetes con opciones IP.

e.1 Recepción de paquetes:

En la recepción, el sistema sigue los siguientes pasos:

1. El driver del dispositivo de red llama a la función “ip_input”.
2. Se revisa el campo versión de IP y el campo longitud de cabecera.
3. Se calcula y se revisa la cabecera “checksum”.
4. Se compara la dirección IP destino con las direcciones IP de las interfaces de red para determinar si el paquete tiene como destino el host local. Si coincide alguna dirección, se utiliza el campo “protocolo” para decidir el protocolo de nivel más alto al cual debe ser

enviado el paquete.

e.2 Envío de paquetes

Un paquete saliente es manejado por la función “ip_output” el cual utiliza la función “ip_route” para encontrar la interfaz de red apropiada por la cual se debe transmitir el paquete. Cuando se determina la interfaz de salida, el paquete es pasado a la función “ip_output_if”. En esta función se cargan todos los campos de la cabecera IP y se calcula el “checksum” de la cabecera IP. La dirección IP del paquete de la interfaz de red saliente es utilizada como la dirección IP origen del paquete. Si no se encuentra una interfaz de red apropiada para enviar los paquetes, se utiliza una interfaz de red por defecto.

e.3 Reenvío de paquetes

Un paquete es reenviado en caso de que ninguna dirección IP de un paquete saliente no coincide con ninguna dirección IP de las interfaces de red. El reenvío es realizado por la función “ip_forward” la cual disminuye el campo TTL y si alcanza el valor de 0, se envía un mensaje de error de ICMP al host que envió el paquete y el paquete es descartado. Si no alcanza el valor de 0, el paquete es reenviado a la interfaz de red apropiada.

f. Procesamiento UDP en lwIP

El protocolo UDP es utilizado para demultiplexar paquetes entre diferentes procesos. El estado de cada sesión UDP es conservado en una estructura PCB (Process Control Block) mostrada en la Figura 2.20. Estas estructuras se guardan en una lista enlazada y son utilizadas para encontrar coincidencias cuando se recibe un datagrama UDP.

```
struct udp_pcb {
  struct udp_pcb *next;
  struct ip_addr local_ip, dest_ip;
  u16_t local_port, dest_port;
  u8_t flags;
  u16_t chksum_len;
  void (* recv)(void *arg, struct udp_pcb *pcb, struct pbuf *p);
  void *recv_arg;
};
```

Figura 2.20 Estructura PCB de UDP en lwIP (Fuente: [7])

Para enviar datos, el programa de la aplicación llama a la función “udp_send” la cual a su vez llama a la función “udp_output”. Dentro de esta función se realiza el “checksum” y se cargan los campos de la cabecera UDP. Finalmente, el paquete es pasado a la función “ip_output_if” para su transmisión. La Figura 2.21 muestra un diagrama de flujo en el cual se puede observar las funciones que son llamadas tanto en la recepción como en la transmisión de paquetes UDP.

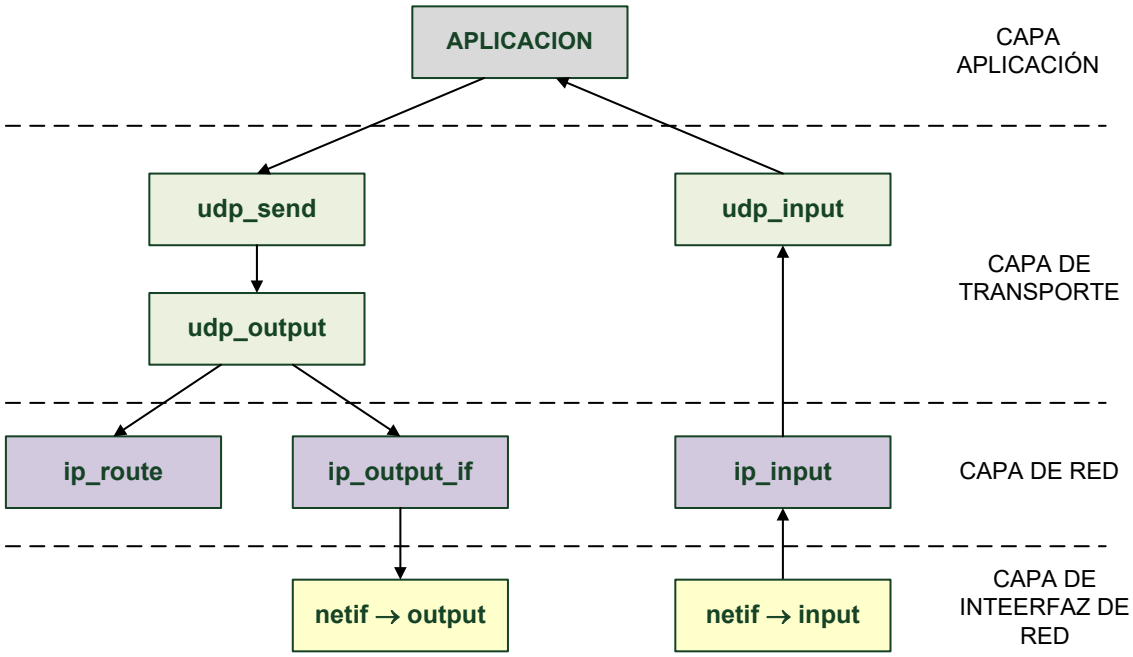


Figura 2.21 Procesamiento del protocolo UDP en lwIP (Fuente: [7])

2.2 Estructura de paquetes que contienen aplicaciones basadas en IPv4

En esta sección se desarrollan los siguientes tópicos: Conceptos básicos de redes IP, la trama Ethernet, protocolo IP, modelo Cliente-Servidor, números de Puerto, protocolo de capa transporte TCP, protocolo de capa transporte UDP.

2.2.1 Conceptos básicos de redes IP

Una red de computadoras es una colección interconectada de computadoras autónomas que sirve para compartir información, compartir recursos y ofrecer servicios.

a. Clasificación de las redes según su tecnología de transmisión

Es la siguiente [8]:

- Redes Broadcast: Existe un solo canal de comunicación compartido por todas las computadoras. Un paquete enviado por un nodo es recibido por los demás.
- Redes Point-to-point: Existen muchas conexiones entre pares individuales de computadoras. Los paquetes pueden atravesar computadoras intermedias por lo cual se requiere el enrutamiento para dirigirlos

b. Clasificación de las redes según su tamaño

Es la siguiente [8]:

- Redes LAN (Local Area Networks): Son redes de propiedad privada dentro de un solo edificio o campus de hasta unos kilómetros de extensión. Las redes LAN usualmente operan a velocidades de 10 a 100 Mbps, tienen bajo retardo (décimas de microsegundos) y experimentan pocos errores.
- Redes MAN (Metropolitan Area network): Es una versión más grande de una red LAN y normalmente se basa en una tecnología similar. Puede abarcar un grupo de oficinas

corporativas cercanas o una ciudad y podría ser privada o pública.

- Redes WAN (Wide Area Network): Es una red que se extiende sobre un área geográfica extensa. Los hosts están conectados por una subred la cual tiene dos componentes distintos: las líneas de transmisión (circuitos, canales o troncales) y los elementos de conmutación (nodos conmutadores de paquetes o routers).
- Internet: Es una red de redes vinculadas por gateways los cuales con computadores que pueden traducir entre formatos incompatibles.

c. Jerarquías de protocolos

La mayor parte de las redes se organiza en una pila de niveles donde cada nivel ofrece servicios a los niveles superiores y oculta la implantación de estos servicios. El nivel “n” de una computadora se comunica con el nivel “n” de la otra computadora. Las reglas y convenciones que controlan esta conversación se denomina protocolo de nivel “n”. El nivel “n” de una computadora no puede transferir los datos directamente al nivel “n” de la otra ya que la información es pasada (mediante primitivas de servicio) a los niveles inferiores hasta llegar al nivel 1 que corresponde al medio físico [8].

d. Modelos de redes [9]

Un modelo de red es una representación del funcionamiento de una red Existen dos tipos básicos de modelos de redes: los modelos de protocolo y los modelos de referencia.

e. Modelo de referencia OSI

El modelo de referencia OSI (Open System Interconnection) es el modelo de referencia de redes más conocido. Se utiliza para el diseño de redes de datos, especificaciones de funcionamiento y resolución de problemas. Proporciona una referencia común para mantener consistencia en todos los tipos de protocolos y servicios de red. Proporciona una amplia lista de funciones y servicios dentro de cada capa. Asimismo, este modelo describe la interacción de cada capa con las capas superiores e inferiores [9]. Este modelo presenta 7 capas las cuales son las siguientes:

1. La capa física: Es la capa de red más básica. Describe los medios mecánicos, eléctricos, funcionales y de procedimientos de las conexiones físicas que sirven para la transmisión de bits.
2. La capa de enlace de datos: Los protocolos de la capa de enlace de datos describen los métodos para intercambiar tramas de datos entre dispositivos que comparten un medio.
3. La capa de red: Proporciona servicios para el intercambio de datos en la red entre dispositivos finales.
4. La capa de transporte: Define los servicios para la segmentación, transferencia y reensamblado de datos para las comunicaciones entre dispositivos finales.

5. La capa de sesión: Proporciona servicios a la capa de presentación para la organización de su diálogo y la administración del intercambio de datos.
6. La capa de presentación: Proporciona una representación común para los datos que son transferidos entre los servicios de la capa de aplicación.
7. La capa de aplicación: Proporciona los medios para la conectividad de extremo a extremo entre usuarios que utilizan la red.

f. Modelo TCP/IP

Es el primer modelo de protocolo en capas también conocido como el modelo de Internet. La arquitectura de la suite de protocolos TCP/IP sigue la estructura de este modelo. Los estándares y los protocolos TCP/IP se definen en un conjunto de documentos denominados RFC (Request for Coments) los cuales contienen las especificaciones formales de los protocolos de comunicación de datos y los recursos que describen el uso de los protocolos. Este modelo presenta 4 capas (Figura 2.22) las cuales son las siguientes [9]:

1. La capa de acceso a la red: Se encarga de controlar los dispositivos de hardware y los medios que forman la red.
2. La capa de Internet: Se encarga de determinar la mejor ruta a través de la red.
3. La capa de transporte: Se encarga de admitir la comunicación entre distintos dispositivos de distintas redes.
4. La capa de aplicación: Se encarga de representar los datos para el usuario y del control de codificación y de diálogo.

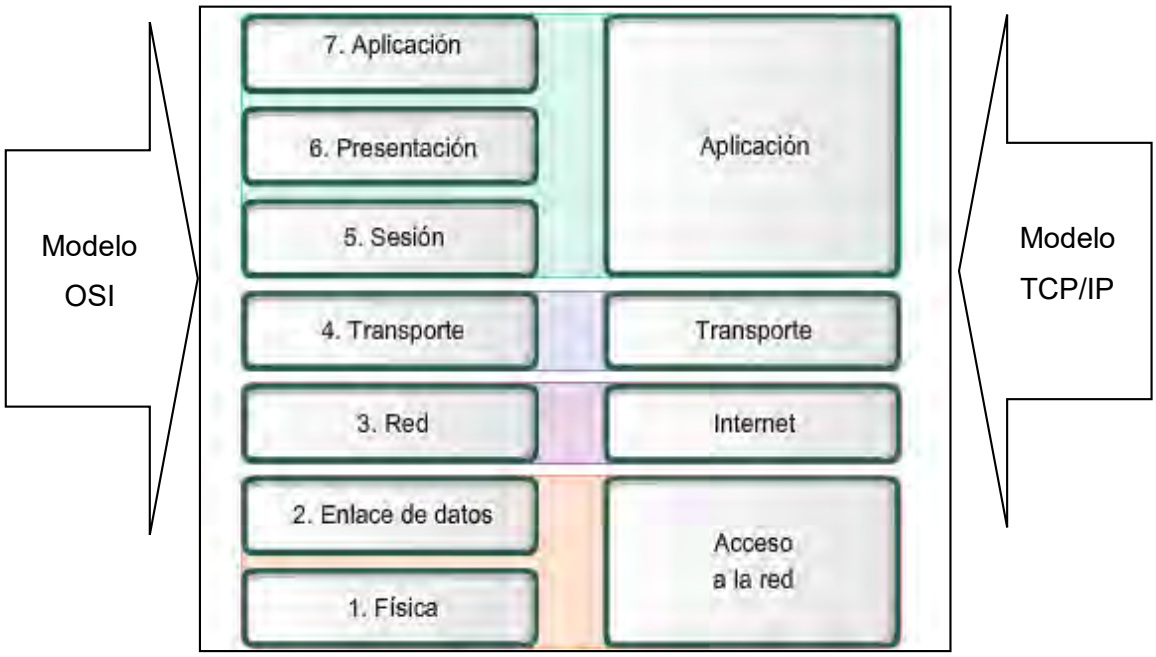


Figura 2.22 Capas del a) modelo de referencia OSI, y del b) modelo TCP/IP (Fuente: Ibídem)

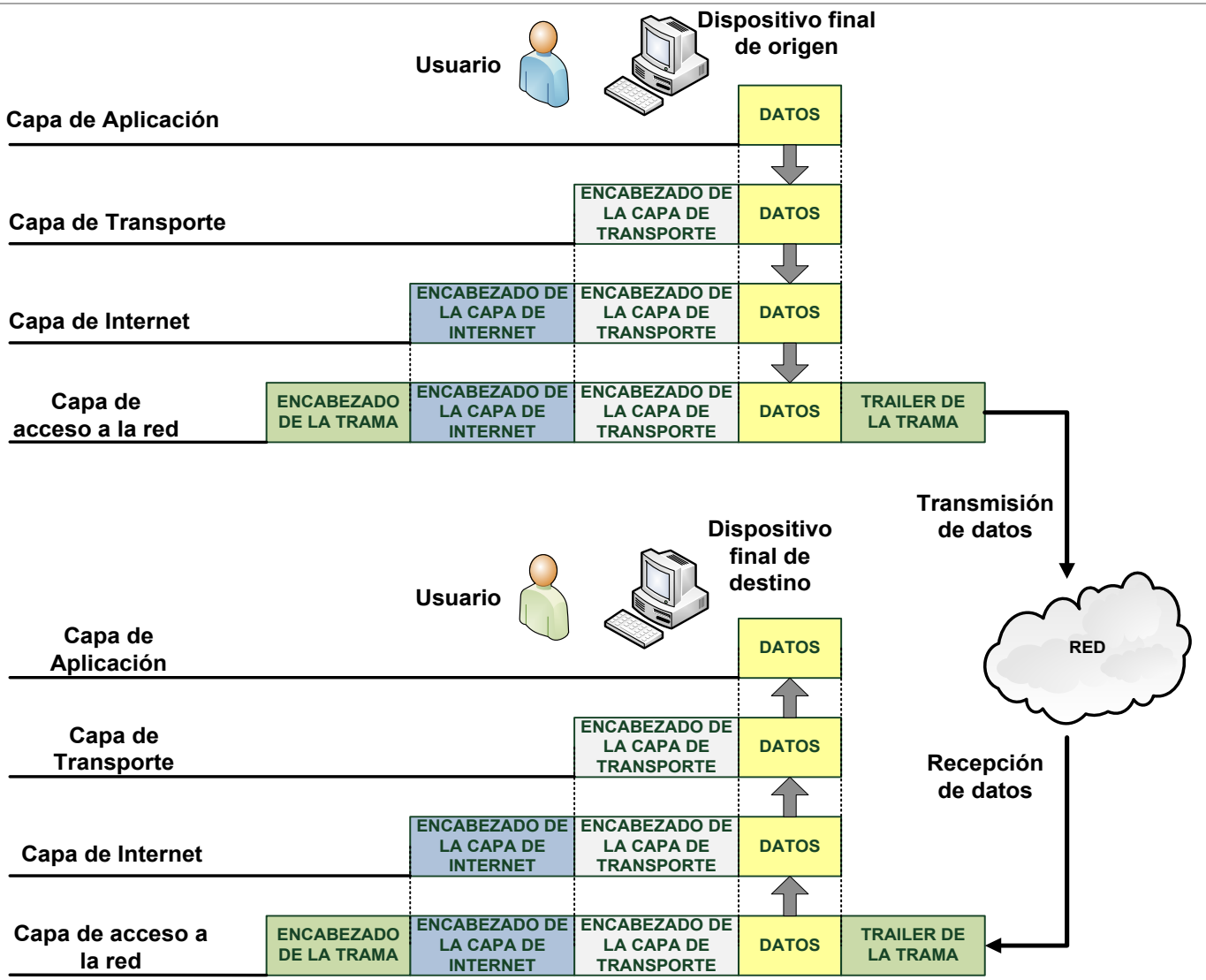


Figura 2.23 Proceso de comunicación entre 2 dispositivos finales (Fuente: [9])

g. Proceso de comunicación

La suite de protocolos TCP/IP se implementan tanto en el host emisor como en el receptor. Estos protocolos interactúan para proporcionar la entrega de aplicaciones de extremo a extremo a través de la red.

La Figura 2.23 (página anterior) ilustra el proceso de comunicación entre dos host finales resaltando el proceso de encapsulamiento y desencapsulamiento en cada capa del modelo TCP/IP.

El proceso completo de comunicación se resume en los siguientes pasos [9]:

- Creación de datos en la capa de aplicación en el dispositivo final de origen.
- Segmentación y encapsulación de datos cuando pasan por la pila de protocolos en el dispositivo final de origen.
- Generación de los datos sobre el medio en la capa de acceso a la red en el dispositivo final de origen.
- Transporte de los datos a través de la red.
- Recepción de los datos en la capa de acceso a la red del dispositivo final de destino.
- Desencapsulación y reestructuración de los datos cuando pasan por la pila de protocolos en el dispositivo final de destino.
- Transferencia de los datos a la aplicación de destino en la capa de aplicación del dispositivo final de destino.

2.2.2 La trama Ethernet

Ethernet un estándar de redes LAN para computadoras con acceso al medio por contienda CSMA/CD (Acceso Múltiple por Detección de Portadora con Detección de Colisiones). Este estándar define las características de cableado y señalización de nivel físico y los formatos de tramas de datos del nivel de enlace de datos del modelo OSI, por tal motivo, Ethernet opera tanto en la capa de enlace de datos como en la capa física del modelo OSI tal como se muestra en la Figura 2.24 [10].

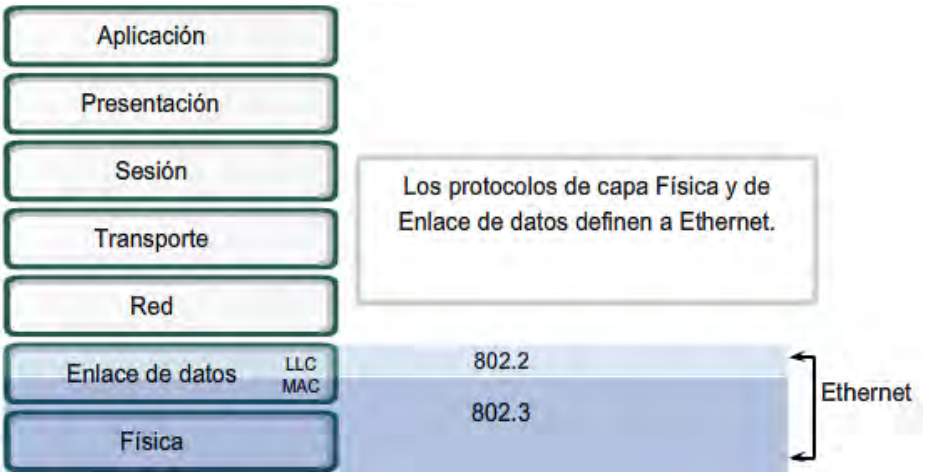


Figura 2.23 Capas donde opera el estándar Ethernet (Fuente: Ibídem)

La estructura de la trama Ethernet agrega encabezados y un tráiler a la unidad de datos de protocolo de capa 3 para encapsular el mensaje que se envía. Existen dos estilos de tramas de Ethernet el IEEE 802.3 (original) y el IEEE 802.3 revisado (Ethernet) cuya diferencia más significativa entre ambos es la agregación de un delimitador de inicio de trama (SFD) y un pequeño cambio en el campo Tipo que incluye la Longitud, tal como se muestra en la Figura 2.24 en la cual se compara ambos estilos.

IEEE 802.3 (original)						
7	1	6	6	2	46 a 1500	4
Preámbulo	Delimitador de inicio de trama	Dirección MAC de destino	Dirección MAC de origen	Longitud/Tipo	Encabezados y datos	Secuencia de verificación de trama

IEEE 802.3 revisado (Ethernet)					
8	6	6	2	46	4
Preámbulo	Dirección MAC de destino	Dirección MAC de origen	Tipo	Datos	Secuencia de verificación de trama

Figura 2.24 Diferencia entre los dos estilos de tramas Ethernet (Fuente: Ibídem)

El estándar Ethernet original define el tamaño mínimo de trama en 64 bytes y el tamaño máximo de trama en 1518 bytes. Este tamaño incluye todos los bytes de los campos Dirección MAC de destino, Dirección MAC de origen, Longitud/Tipo, Encabezado y datos y Secuencia de verificación de trama. Los campos Preámbulo y Delimitador de inicio de trama no se incluyen en la descripción del tamaño de una trama, tal como se muestra en la Figura 2.25.

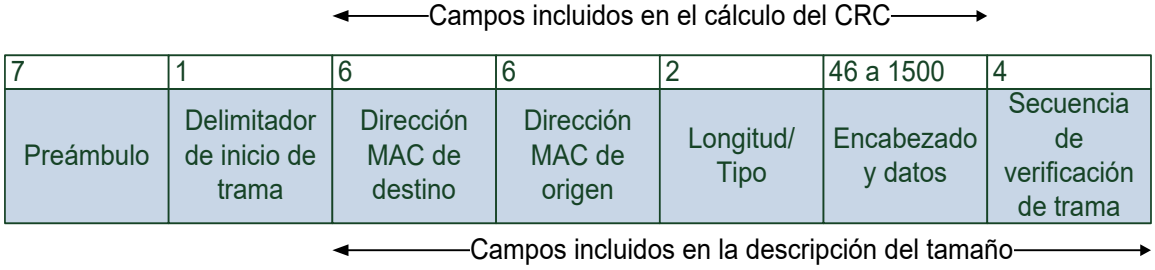


Figura 2.25 Formato de la cabecera Ethernet (Fuente: Ibídem)

A continuación se describen cada uno de los campos de la trama Ethernet original.

- Campo Preámbulo (7 bytes) y campo Delimitador de inicio de trama (SDF) (1 byte): Ambos campos se utilizan para la sincronización entre el emisor y el receptor. Los primeros bytes le indican al receptor que se prepare para recibir una trama nueva.
- Campo Dirección MAC de destino (6 bytes): Identifica al receptor. La Capa 2 utiliza esta dirección para ayudar a los dispositivos a determinar si la trama viene dirigida a ellos. La dirección de la trama se compara con la dirección MAC del dispositivo. Si coinciden, el dispositivo acepta la trama.
- Campo Dirección MAC de origen (6 bytes): Identifica la NIC (Network Interface Card) o

interfaz que origina la trama.

- Campo Longitud/Tipo (2 bytes): Indica la longitud exacta del campo Datos de la trama. Si el objetivo del campo es designar un tipo, el campo Tipo sirve para determinar qué protocolo de capa superior está presente. Si el valor de los dos octetos de este campo es equivalente a 1536 decimal o mayor, los contenidos del campo Datos se codifican según el protocolo indicado.
- Campos Datos y Relleno (de 46 a 1500 bytes): Contienen los datos encapsulados de una capa superior (Capa 3 genérica). Usualmente, los datos encapsulados de capa 3 corresponden a un paquete IPv4. Todas las tramas deben tener al menos 64 bytes de longitud. Si se encapsula un paquete pequeño, el campo relleno se utiliza para aumentar el tamaño de la trama hasta alcanzar el tamaño mínimo de 64 bytes.
- Campo Secuencia de verificación de trama (4 bytes): Este campo se denomina FCS (Frame Check Sequence) el cual se utiliza para detectar errores en la trama Ethernet. Para ellos, se calcula el valor de CRC (Cyclic Redundancy Check) cuyo resultado es incluido en el campo FCS. El receptor recibe la trama y calcula el CRC para detectar errores. Si ambos valores de CRC coinciden, significa que no se produjo ningún error. Si ambos cálculos no coinciden significa que los datos fueron modificados por diversos motivos y la trama es descartada.

2.2.3 Protocolo IP

El Protocolo de Internet (IPv4 e IPv6) es el protocolo de transporte de datos de la capa 3 más ampliamente utilizado. Fue diseñado como un protocolo con bajo costo. Provee sólo las funciones necesarias para enviar un paquete desde un origen a un destino a través de un sistema interconectado de redes. No rastrea ni gestiona el flujo de paquetes ya que estas funciones son realizadas por protocolos de otras capas [11].

Principales características del protocolo IPv4:

- Es un protocolo sin conexión ya que no establece una conexión previa antes de enviar los paquetes de datos.
- Es un protocolo de máximo esfuerzo (no confiable) ya que no se usan encabezados o mecanismos para garantizar la entrega de paquetes.
- Es un protocolo que no depende del medio físico donde se transportan los datos. cualquier paquete IP puede ser comunicado ya sea eléctricamente por cable, por medio de señales ópticas por fibra, o sin cables por las señales de radio.

El tamaño máximo de un paquete que cada medio diferente puede transportar se denomina MTU (Maximum Transfer Unit). La capa de red determina el tamaño de cada paquete según el MTU del medio cuyo valor es entregado por la capa de enlace de datos. En algunos casos, un dispositivo intermediario necesita dividir un paquete cuando se lo

envía desde un medio con MTU más grande que otro. Este proceso se denomina fragmentación de paquetes.

El protocolo IPv4 encapsula o empaqueta el datagrama o segmento de la capa de Transporte para que la red pueda entregarlo a su host de destino. Durante este proceso, la unidad de datos de protocolo de la Capa de transporte encapsulada, permanece sin cambios.

La Figura 2.26 muestra el formato de la cabecera IPv4 con cada uno de sus campos.

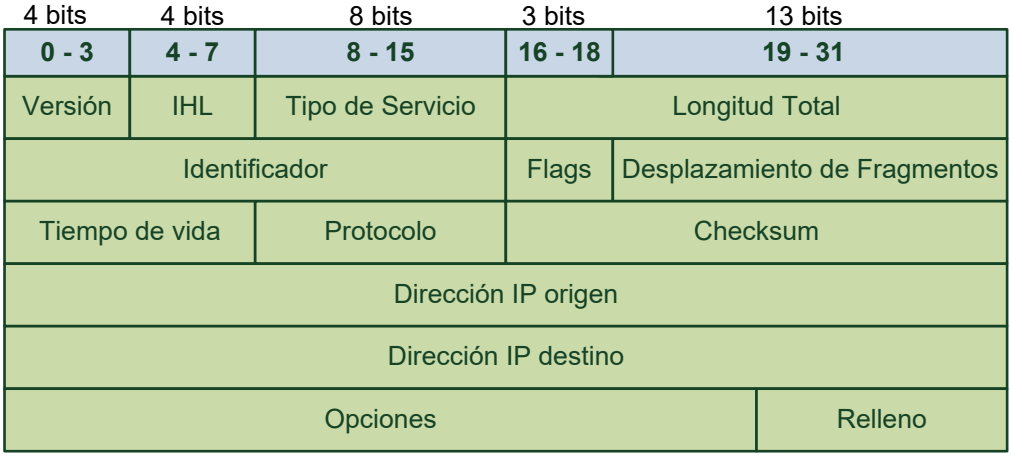


Figura 2.26 Formato de la cabecera IPv4 (Fuente: Ibídem)

- Campo Versión (4 bits): Contiene el número de la versión del protocolo IP, en este caso su valor es 4.
- Campo Tamaño de cabecera IHL (Internet Header Length) (4 bits): Indica el tamaño de la cabecera IP del paquete expresado en palabras de 32 bits. Su valor mínimo es 5 y su valor máximo es 15.
- Campo Tipo de servicio (8 bits): Contiene un valor binario de 8 bits que se usa para determinar la prioridad de cada paquete. Permite la aplicación de un mecanismo de Calidad del Servicio (QoS) a paquetes de alta prioridad tales como aquellos que llevan datos de voz en telefonía.
- Campo Longitud total (16 bits): Indica el tamaño completo del paquete expresado en bytes, incluyendo el encabezado y los datos.
- Campo Identificador (16 bits): Utilizado para identificar los fragmentos de un datagrama unos de otros en un paquete fragmentado.
- Campo Flags o Señalizador (3 bits): Se utiliza para especificar valores referentes a la fragmentación de paquetes. El bit 2 es reservado y debe ser 0. El bit 1 es el señalizador de no fragmentar, si su valor es 1, el paquete no se debe fragmentar. El bit 0 es el señalizador de más fragmentos, si su valor es 1 indica que existen más fragmentos del paquete.
- Campo Desplazamiento de fragmentos (13 bits): Identifica el orden en el cual ubicar el fragmento de un paquete fragmentado durante la reconstrucción del mismo.

- Campo Tiempo de vida TTL (Time to live) (8 bits): Indica el tiempo remanente de "vida" del paquete. El valor TTL disminuye al menos en uno cada vez que el paquete es procesado por un router. Cuando el valor se vuelve cero, el router descarta o elimina el paquete y es eliminado del flujo de datos de la red.
- Campo Protocolo (8 bits): Indica el tipo de carga que transporta el paquete. Este campo permite pasar los datos al protocolo apropiado de la capa superior. Para el protocolo ICMP el valor es 01, para el protocolo TCP el valor es 06 y para el protocolo UDP es 17.
- Campo Checksum (16 bits): Se utiliza para controlar errores de la cabecera del paquete. Su cálculo consiste en sumar en complemento a 1 cada palabra de 16 bits de la cabecera considerando el valor de 0 para el campo Checksum y haciendo el complemento a 1 del valor resultante.
- Campo Dirección IP origen (32 bits): Representa la dirección de host de capa de red de origen del paquete.
- Campo Dirección IP destino (32 bits): Representa la dirección de host de capa de red de destino del paquete.
- Campo Opciones: Puede contener un número indeterminado de opciones para proveer otros servicios que son rara vez utilizados.
- Campo Relleno: Es utilizado opcionalmente para asegurar que el tamaño en bits de la cabecera IP sea un múltiplo de 32. El valor utilizado en este campo es el 0.

2.2.4 Modelo Cliente-Servidor

La mayoría de las aplicaciones de red fueron creadas asumiendo que un lado es el cliente y el otro el servidor [13]. El propósito de una aplicación de red es que el servidor proporcione algún servicio a los clientes. Los servidores se pueden clasificar en dos clases los iterativos y concurrentes, cuyo modo de procesar las consultas son descritas en la Tabla 2.3.

Tabla 2.3 Tipos de servidores en el modelo Cliente-Servidor (Fuente: Ibídem)

Servidores iterativos	Servidores concurrentes
- Esperan la llegada de la consulta de un cliente. - Procesan la consulta del cliente. - Envían la respuesta de vuelta al cliente que envió la consulta. - Vuelve a realizar el primer paso.	- Esperan la llegada de la consulta de un cliente. - Inician un nuevo servidor para manejar la consulta del cliente. Esto implica la creación de un nuevo proceso, tarea o hilo de ejecución dependiendo del sistema operativo. - El nuevo servidor procesa la consulta del cliente y su función termina cuando finaliza el proceso de la consulta. - Vuelve a realizar el primer paso.

Generalmente los servidores TCP son concurrentes y los servidores UDP son

iterativos.

2.2.5 Números de puerto

Los protocolos TCP y UDP identifican las aplicaciones utilizando números de puerto de 16 bits [12]. Existen en total 65535 números de puerto posibles. Los números son administrados por la IANA (Internet Assigned Numbers Authority) y se dividen en 3 categorías mostradas en la Tabla 2.4.

Tabla 2.4 Clasificación de los números de puerto (Fuente: Ibídem)

	Puertos bien conocidos	**Puertos registrados**	**Puertos dinámicos o privados**
Rango	0 - 1023	1024 - 49151	49152 - 65535
Descripción	Están reservados para servicios y aplicaciones conocidas, entre ellas HTTP, POP3/SMTP y Telnet.	Son asignados a procesos o a aplicaciones individuales del usuario.	Son conocidos como puertos efímeros y suelen asignarse de manera dinámica a aplicaciones de un cliente cuando se inicia una conexión.

En los sistemas Unix los números de puerto bien conocidos están contenidos en el archivo “/etc/services”. En el Anexo C se muestra la lista de números de puerto bien conocidos la cual es extraída de dicho archivo.

2.2.6 Protocolo de capa transporte TCP

TCP (Transport Control Protocol) es un protocolo orientado a la conexión, descrito en la RFC 793. Proporciona un flujo de bytes confiable de extremo a extremo sobre la capa de red que es no confiable. Para utilizar los servicios de este protocolo, el transmisor y el receptor tienen que crear puntos terminales de la conexión llamados sockets. La dirección de un socket es la dirección IP del host y un número de 16 bits llamado número de puerto. Una conexión es identificada con las direcciones de socket de cada extremo. TCP también proporciona mecanismos de control de flujo [12].

TCP también provee mecanismos para el control del flujo los cuales contribuyen con la confiabilidad de la transmisión ajustando la tasa efectiva de flujo de datos entre los dos servicios de la sesión TCP. En TCP se mandan datos en segmentos cuyo tamaño está limitado por tamaño máximo de un paquete IP y el MTU de cada red. Cuando el transmisor advierte que se recibió la cantidad de datos especificados en los segmentos, puede continuar enviando más datos para esta sesión.

Algunas de las aplicaciones que utilizan TCP son los exploradores Web, correo electrónico y transferencia de archivos. La Figura 2.27 muestra el formato de la cabecera TCP con cada uno de sus campos.

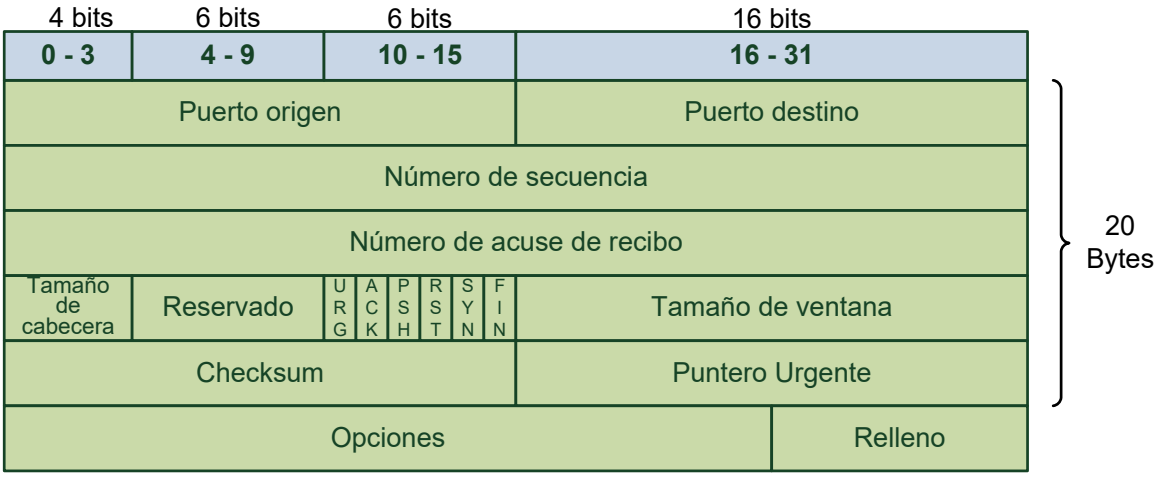


Figura 2.27 Formato de la cabecera TCP (Fuente: Ibídem)

- Campo puerto de origen (16 bits): Contiene el número de puerto origen.
- Campo puerto de destino (16 bits): Contiene el número de puerto destino.
- Campo Número de secuencia (32 bits): Indica el número de secuencia del primer octeto del segmento TCP.
- Campo Número de acuse de recibo (32 bits): Indica el próximo número de secuencia esperado por el receptor.
- Campo Tamaño de la cabecera TCP (4 bits): Indica el número de palabras de 32 bits que tiene la cabecera TCP.
- Campo Reservado (6 bits): Reservado para uso futuro. Su valor debe ser 0.
- Campo Flags o Señalizadores (6 bits): Son los siguientes:
  - URG: Puntero urgente. Indica que el segmento contiene datos urgentes.
  - ACK: Indica que hay un número de acuse en el campo de acuse.
  - PSH: Función push. Indica que el receptor no debe almacenar los datos antes de entregarlos.
  - RST: Reset. Indica el reinicio de la conexión.
  - SYN: Sirve para sincronizar los números de secuencia.
  - FIN. Indica que el emisor no tiene más datos para enviar.
- Campo Tamaño de la Ventana (16 bits): Indica el número de octetos o bytes que el receptor del segmento TCP está dispuesto a aceptar.
- Campo Checksum (16 bits): Indica el complemento a uno de la suma en complemento a uno de todas las palabras de 16 bits en la cabecera TCP. Durante el cálculo del checksum, este campo es reemplazado por ceros. El cálculo del checksum también incluye una pseudocabecera que contiene las direcciones IP origen y destino, el campo protocolo y la longitud del paquete TCP.
- Campo Puntero Urgente (16 bits): Apunta al número de secuencia del siguiente byte de

datos urgentes. Este campo es interpretado solamente si el bit URG está establecido en 1.

- Campo Opciones: Información opcional.

- Campo Relleno: Se utiliza opcionalmente para asegurar que el tamaño en bits de la cabecera TCP sea un múltiplo de 32. El valor utilizado en este campo es el 0.

2.2.7 Protocolo de capa transporte UDP

UDP es un protocolo simple de la capa transporte, sin conexión, descrito en la RFC 768. Cuenta con la ventaja de proporcionar la entrega de datos sin utilizar muchos recursos. Las unidades de datos del protocolo UDP se llaman datagramas los cuales son enviados como "mejor intento". UDP Genera mucho menos sobrecarga que TCP, ya que no es orientado a la conexión y no cuenta con los sofisticados mecanismos de retransmisión, secuenciación y control del flujo de TCP [12].

Entre los protocolos principales de la capa de Aplicación que utilizan UDP se incluyen SNMP, DHCP, RIP, TFTP, DNS, etc. Asimismo, UDP se utiliza mayormente en aplicaciones de streaming de video y en VoIP (Voz sobre IP).

La Figura 2.28 muestra el formato de la cabecera UDP con cada uno de sus campos.

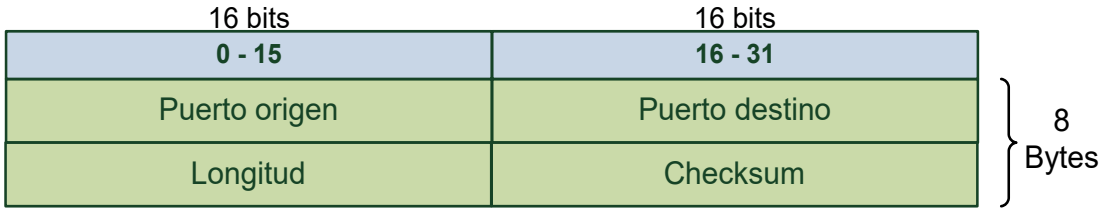


Figura 2.28 Formato de la cabecera UDP (Fuente: Ibídem)

- Campo Puerto origen (16 bits): Contiene el número de puerto origen. Es opcional.

- Campo puerto destino (16 bits): Contiene el número de puerto destino.

- Campo Longitud (16 bits): Indica el tamaño total del datagrama UDP incluyendo los datos. El valor mínimo es de 8 bytes.

- Campo Checksum (16 bits): Indica el complemento a uno de la suma en complemento a uno de todas las palabras de 16 bits que abarca una pseudocabecera IP, la cabecera UDP, los datos y ceros hasta completar un múltiplo de 16 bits. Durante el cálculo del checksum, este campo es reemplazado por ceros. La pseudocabecera IP contiene las direcciones IP origen y destino, el campo protocolo y la longitud del paquete UDP.

2.3 Protocolos de intercambio de información de flujos de tráfico IP

Existen varias aplicaciones que requieren mediciones de tráfico IP basado en flujos. Tales mediciones pueden ser realizadas por un router durante el proceso de reenvío de paquetes, por un firewall o por una sonda de medición de tráfico conectada ya sea a un enlace o a un puerto de monitoreo. Existen protocolos diseñados exclusivamente para la exportación de información de flujos, sin embargo, el proceso de exportación no es

realizado únicamente por el protocolo, también requiere un sistema de procesos que operan de manera coordinada. Los protocolos de exportación de información de flujos tales como Netflow o IPFIX tienen ciertos requerimientos los cuales son especificados en RFCs y son resumidos en esta sección.

2.3.1 Definiciones relacionadas

En esta sección se desarrollan los siguientes temas: definición de flujo, punto de observación, registro de flujo, proceso de medición, proceso de exportación, y proceso de recolección. Estos conceptos provienen del documento RFC 3917. “Requirements for IP Flow Information Export” [47].

a. Definición de flujo

Un flujo está definido como un conjunto de paquetes IP atravesando un punto de observación en la red durante un cierto intervalo de tiempo. Todos los paquetes de un flujo particular tienen un conjunto de propiedades comunes.

Las propiedades comunes pueden ser las siguientes:

1. Uno o más campos de cabeceras de los paquetes.
2. Una o más características propias de los paquetes.
3. Uno o más campos derivados del tratamiento del paquete.

Un paquete pertenece a un flujo si satisface completamente todas las propiedades definidas para un flujo.

b. Punto de observación

Un punto de observación es la ubicación en la red donde se pueden observar los paquetes IP. Algunos ejemplos de puntos de observación son los siguientes:

- Un enlace en la cual se tiene instalada una sonda.
- Un medio compartido tal como una red LAN basada en Ethernet.
- Algún puerto de un router.
- Un conjunto de interfaces físicas o lógicas de un router.

c. Registro de flujo

Un registro de flujo contiene información sobre un flujo específico que fue medido en un punto de observación. Asimismo, contiene mediciones tales como el número total de bytes y de paquetes del flujo y propiedades características tal como la dirección IP origen.

d. Proceso de medición

Este proceso genera los registros de flujos. La entrada al proceso son las cabeceras de los paquetes. Este proceso consiste en un conjunto de funciones que incluyen las siguientes:

- Captura de las cabeceras de los paquetes.
- Colocar una marca de tiempo (fecha y hora).

- Muestreo.
- Clasificación.
- Mantenimiento de los registros de flujos. Esta función incluye:

- Creación de nuevos registros.
- Actualización de los registros que ya se tienen.
- Cálculo de estadísticas de flujos.
- Derivación de otras propiedades de los flujos.
- Detección de la expiración de los flujos.
- Transferencia de los registros de flujos al proceso de exportación.
- Borrado de los registros de flujos.

La Figura 2.29 muestra la secuencia de aplicación de cada función. La función de muestreo no se ilustra ya que esta función podría aplicarse antes de cualquier función.

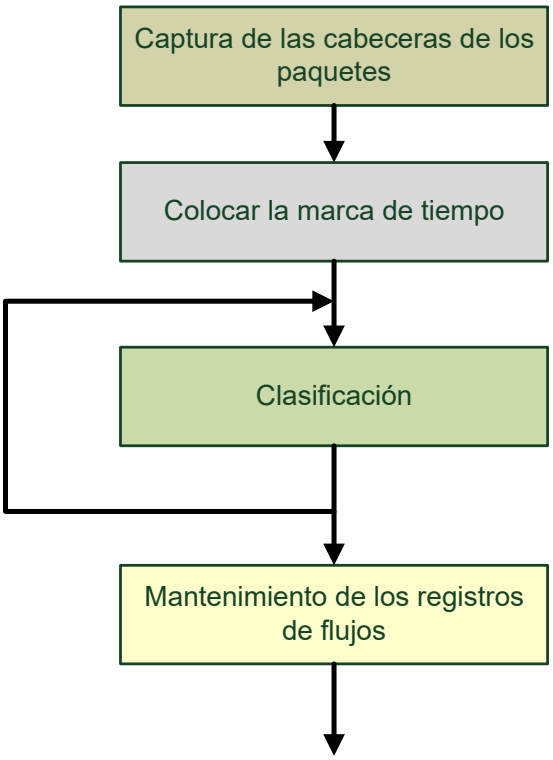


Figura 2.29 Secuencia de las funciones del proceso de medición (Fuente: Ibídem)

e. Proceso de exportación

El proceso de exportación se encarga de enviar los registros de flujos a uno o más procesos de recolección.

Los registros de flujos son generados por uno o más procesos de medición.

f. Proceso de recolección

El proceso de recolección se encarga de recibir los registros de flujos de uno o más procesos de exportación. Este proceso podría almacenar los registros de flujos recibidos y procesarlos.

2.3.2 Aplicaciones que requieren mediciones de tráfico IP basado en flujos

Existen muchas aplicaciones que requieren el uso de mediciones de tráfico IP basado en flujos. A continuación se describen las aplicaciones que son consideradas de importancia en las redes actuales [47].

a. Contabilización basada en el uso: Varios modelos de negocios para la entrega se servicios IP están actualmente en investigación. La contabilización de un servicio puede estar basada ya sea en el tiempo de uso o en el volumen de la información.

b. Perfiles del tráfico: Los protocolos de exportación de información de flujos ayudan en el proceso de caracterización de flujos IP. Usualmente, en dicho proceso, se utiliza un modelo que representa parámetros claves en los flujos tales como la duración del flujo, el volumen y el tiempo. Esto permite realizar la planificación de la red, dimensionamiento de la red, análisis de tendencias, desarrollo de modelo de negocios y otras actividades. La información necesaria para tener un perfil del tráfico es la distribución de servicios y protocolos utilizados en la red, la cantidad de paquetes de un tipo específico y las características específicas de cada flujo.

c. Ingeniería de tráfico: La ingeniería de tráfico consiste en medir, modelar, caracterizar y controlar la red. Su objetivo es la optimización del uso de recursos de red y del rendimiento del tráfico. Los parámetros necesarios para realizar ingeniería de tráfico son la utilización de un enlace, la carga entre nodos de red específicos, número y tamaño de puntos de entrada y salida de flujos activos e información de enrutamiento.

d. Detección de ataques e intrusiones: La información de flujos cumple un rol fundamental en la detección de violaciones de seguridad. El monitoreo de flujos permite la detección de situaciones inusuales en el caso de un ataque de denegación de servicio. El análisis de los flujos puede ser útil para obtener información de los flujos atacantes y para la planificación de una estrategia de defensa.

e. Monitoreo de QoS (Quality of Service): El monitoreo de QoS es una medición pasiva de los parámetros de calidad para flujos IP. Se utiliza el tráfico existente en la red para el análisis de la calidad de servicio. Un claro ejemplo es la validación de los parámetros negociados en la especificación del nivel de servicio. Para ello, se requiere la correlación de datos provenientes de múltiples puntos de observación.

2.3.3 Protocolos NetFlow e IPFIX

Netflow e IPFIX son los protocolos de exportación de información de flujos más utilizados en la actualidad. A continuación se describe brevemente cada uno de ellos.

- Netflow [14]: Es un protocolo desarrollado por Cisco Systems para la recolección de información del tráfico de red. Netflow proporciona valiosa información tal como los usuarios y las aplicaciones en la red, los tiempos de máxima utilización de la red e

información del tráfico de enrutamiento. Netflow versión 9 es la última versión del protocolo, la cual implementa un método flexible y extensible para registrar los datos. Esta versión es la base del nuevo estándar IPFIX.

- IPFIX (IP Flow Information Export) [15]: Es un estándar propuesto por el IETF (Internet Engineering Task Force) para la exportación de información de flujos en redes IP. Es el protocolo sucesor de Cisco Netflow versión 9 del cual está basado. Sus principales características es su definición flexible de un flujo de red y la descripción de formatos de registros mediante el uso de plantillas basadas en un modelo de información extensible y bien definido. Esto lo hace muy útil para la gestión de redes en niveles superiores a la capa de red y de transporte.

2.4 Introducción a la teoría de la información

Esta sección inicia presentando algunos conceptos básicos de estadística descriptiva. Luego se presentan algunas medidas de tendencia central y de dispersión de datos. Posteriormente se exponen algunos conceptos relacionados con la probabilidad. Por último, se brindan los conceptos básicos de la teoría de la información tales como la entropía y entropía estandarizada [16] [17] [18] [19] [20] [1].

2.4.1 Conceptos básicos de estadística descriptiva

Son los siguientes:

a. Observación: Es la información o característica que se registra de cada objeto observado o unidad observada.

b. Variables cualitativas: Son aquellas que clasifican los objetos observados en categorías. Con estas variables se puede contar número de casos, comparar entre categorías, pero no se pueden realizar operaciones matemáticas. Dentro de este tipo de variables se distingue las variables nominales las cuales están asociadas a nombres como por ejemplo marca de auto, religión, estado civil, etc. y las variables ordinales las cuales tienen asociado un orden como por ejemplo nivel educacional, nivel socioeconómico, etc.

c. Variables cuantitativas: Son aquellas que tienen valores numéricos que representan medidas o frecuencias. Se pueden realizar operaciones matemáticas con estas variables. Dentro de este tipo de variables se distingue las variables cuantitativas discretas y continuas. Una variable se denomina discreta si solamente puede tomar un número finito o contable de valores. Una variable se denomina continua si puede tomar un número infinito de valores en un intervalo dado.

d. Experimento aleatorio: Es toda acción cuyo resultado no puede predecirse con certeza, por ejemplo el lanzamiento de un dado; el resultado de lanzar un dado no puede predecirse con certeza ya que es igual de probable obtener cualquiera de los 6 resultados posibles.

e. Espacio muestral: Es el conjunto de resultados posibles o sucesos elementales de un

experimento aleatorio. Puede ser de 2 tipos:

- Espacio muestral finito: Cuando se conoce cuantos resultados posibles existen.
- Espacio muestral infinito: Cuando se tiene infinitos resultados posibles.

f. Evento o suceso: Es un subconjunto del espacio muestral, es decir, es un conjunto de resultados posibles de un experimento aleatorio.

g. Variable aleatoria: Es una variable estadística cuyos valores son obtenidos de mediciones realizadas en cualquier experimento aleatorio. De manera formal, es una función que asigna eventos a números reales. El valor de una variable aleatoria no es fijo y puede tomar diferentes valores como resultado de algún experimento aún no realizado. Una variable aleatoria X es una función real definida en el espacio muestral Ω asociado a un experimento aleatorio. Dado un experimento aleatorio, con espacio muestral Ω, una variable aleatoria X es una función cuyo dominio es el espacio muestral y el rango son números reales, es decir, X: $\Omega \rightarrow \mathbb{R}$.

2.4.2 Medidas de tendencia central y de dispersión

Son las siguientes:

a. Media aritmética: Es una medida de tendencia central de un conjunto de datos de una muestra. Es el valor resultante que se obtiene al dividir la sumatoria de un conjunto de datos sobre el número total de datos. Se utiliza para el tratamiento de datos cuantitativos. Sea X un conjunto de datos que contiene n muestras, la media aritmética denotada por $\overline{X}$ se define por la siguiente expresión:

$$\overline{X}=\frac{\sum_{i=1}^{i=n} x_i}{n} \qquad (2.1)$$

b. Varianza: Es una medida de dispersión absoluta de un conjunto de datos de una muestra. Se define como el promedio del cuadrado de las distancias entre cada observación y la media aritmética de un conjunto de observaciones. Sea X un conjunto de datos que contiene n muestras y cuya media aritmética es $\overline{X}$, la varianza de X denotada por σ^2 se define por la siguiente expresión:

$$\sigma^2 = \frac{\sum_{i=1}^{i=n}(x_i - \overline{X})^2}{n} = \frac{\sum_{i=1}^{i=n} x_i^2}{n} - \overline{X}^2 \qquad (2.2)$$

c. Desviación estándar: Es una medida de dispersión absoluta de un conjunto de datos de una muestra. Es una medida cuadrática que informa sobre la media de distancias que tienen los datos respecto a su media aritmética. Se define como la raíz cuadrada de la varianza. Sea X un conjunto de datos que contiene n muestras y cuya media aritmética es $\overline{X}$, la desviación estándar de X denotada por σ se define por la siguiente expresión:

$$\sigma=\sqrt{\sigma^2}=\sqrt{\frac{\sum_{i=1}^{i=n}(x_i-\overline{X})^2}{n}}=\sqrt{\frac{\sum_{i=1}^{i=n}x_i^2}{n}-\overline{X}^2} \quad (2.3)$$

d. Coeficiente de variación de Pearson: Es una medida de dispersión relativa de un conjunto de datos de una muestra. Se define como el cociente entre la desviación estándar y el valor absoluto de la media aritmética. Sea X un conjunto de datos que contiene n muestras y cuya media aritmética es $\overline{X}$, el coeficiente de X denotado por CV se define por la siguiente expresión:

$$CV=\frac{\sigma}{\overline{X}} \quad (2.4)$$

Esta medida representa el número de veces que la desviación estándar contiene a la media aritmética. A mayor valor, mayor es la dispersión de los datos y la media tiene menor representatividad. Su valor es usualmente menor que 1, pero en algunas distribuciones de probabilidad puede ser uno o mayor que uno.

2.4.3 Probabilidad

La probabilidad es un número que cuantifica la frecuencia con que se obtiene un resultado luego de realizar un experimento aleatorio. La probabilidad de ocurrencia un evento o suceso A se define como:

$$P(A)=\frac{\text{Número de casos favorables}}{\text{Número de casos posibles}} \quad (2.5)$$

Donde:

$0 \le P(A) \le 1$

Si P(A)=0, entonces A es un evento imposible.

Si P(A)=1, entonces A es un evento seguro.

Se define como probabilidad condicional a la probabilidad de que ocurra un evento A dado la verificación de ocurrencia de un evento B. Se denota como P(A|B):

$$P(A|B)=\frac{P(A\cap B)}{P(B)} \quad (2.6)$$

Si los eventos A y B son excluyentes se tiene: $P(A\cap B)=0$ y $P(A|B)=0$. Por otro lado, si B es un evento seguro, se tiene: $P(B)=1$ y $P(A|B)=P(A\cap B)$.

Se define como distribución de probabilidad a una función que asigna a cada evento o suceso aleatorio la probabilidad de ocurrencia de dicho suceso. La distribución de probabilidad se define como el conjunto de probabilidades de todos los sucesos. Si X es una variable aleatoria discreta que toma los valores $x_1, x_2,\ldots, x_n$, la función o distribución de probabilidad de la variable X es la función que asigna a cada valor x_i la probabilidad de que X tome el valor x_i, es decir:

$$F(x_i)=P(X=x_i) \quad (2.7)$$

2.4.4 Entropía

La entropía es una medida de la incertidumbre de una variable aleatoria [1]. Sea X una variable aleatoria discreta que puede tomar N_X valores diferentes.

Sea m el número de observaciones de la variable X. Se induce una distribución de probabilidad empírica de X, de la siguiente naturaleza $p(x_i)=m_i/m$, tal que $x_i \in X$, donde m_i es la frecuencia del número de veces que se observa a la variable aleatoria X tomando el valor x_i. La entropía de la variable aleatoria X se denota como H(X) y está definida por la siguiente ecuación:

$$H(X) = -\sum_{x_i \in X} p(x_i).\log_2 p(x_i) \tag{2.8}$$

La entropía es expresada en bits. Por convención se tiene que $0.\log_2 0 = 0$, debido a que $x.\log_2 x \to 0$ cuando $x \to 0$ y a que la agregación de términos de probabilidad 0 no cambia el valor de la entropía.

La entropía mide la variedad de la observación de los valores de X. El rango de valores que puede tomar la entropía es el siguiente:

$$\begin{gathered} 0 \le H(X) \le H_{max}(X) \\ H_{max}(X) = \log_2[\min\{N_X, m\}] \end{gathered} \tag{2.9}$$

El máximo número de posibles de posibles valores únicos que la variable aleatoria X puede tomar en m observaciones es $2^{H_{max}(X)}$. Se asume que $m \ge 2$ y $N_X \ge 2$, de lo contrario, no tendría sentido emplear el concepto de variedad de la observación.

2.4.5 Entropía estandarizada

Es una medida especial utilizada en la metodología descrita en la referencia [1]. La metodología también utiliza el término de incertidumbre relativa como equivalente. Esta medida proporciona un índice de la variedad o uniformidad sin tener en cuenta el número de valores diferentes "N_X" que puede tomar una variable aleatoria o el número de observaciones "m". Esta medida se denota como RU (Relative Uncertainty) y está definida por la siguiente expresión:

$$RU(X) = \frac{H(X)}{H_{max}(X)} = \frac{H(X)}{\log_2[\min\{N_X, m\}]} \tag{2.10}$$

Si $RU(X) = 0$, indica que todas las observaciones de X son las mismas, es decir, $p(x) = 1$ y por lo tanto, no existe variedad de la observación.

Generalizando, sea A un subconjunto de los valores observados en X, es decir, A es el conjunto de valores observados.

Se considera como medidas generales de la uniformidad en los valores observados de X la entropía condicional denotada por H(X|A) y la respectiva incertidumbre relativa condicional denotada por RU(X|A), definidas por las siguientes expresiones:

$$H(X|A) = H(X) \tag{2.11}$$

$$RU(X|A) = \frac{H(X)}{\log_2|A|} \tag{2.12}$$

$$H_{max}(X|A)=\log_2|A| \tag{2.13}$$

El valor de |A| es la cantidad de valores del conjunto A, es decir, el número total de valores que se observaron.

Si $RU(X|A) \approx 1$, entonces significa que los valores observados de X (conjunto A) están cerca de estar uniformemente distribuidos y por lo tanto son casi indistinguibles unos de otros.

Si $RU(X|A) \ll 1$, entonces significa que los valores observados de X (conjunto A) está sesgado, es decir, contiene unos pocos valores que son observados con mayor frecuencia que el resto.

CAPÍTULO III
ALTERNATIVAS DE SOLUCIÓN

En este capítulo consta de tres secciones. Primero se presentan los elementos necesarios para la implementación de una solución integral al problema de ingeniería planteado. La segunda sección describe brevemente cada alternativa existente para el primer elemento, luego se indica la alternativa escogida y las razones de su elección. La última sección describe para el segundo elemento, luego se indica la alternativa escogida y las razones de su elección.

3.1 Elementos necesarios

Para obtener indicadores de la producción de servicios de una red Ethernet existen diversas alternativas. La gran mayoría de ellas requiere el uso de dos elementos sustanciales descritos a continuación:

- Un módulo de captura de tráfico: Este dispositivo se encarga de realizar la lectura y procesamiento de paquetes IP en un enlace de red específico.
- Una aplicación para modelamiento y reportes: Es un programa de software que se encarga de recibir información de la sonda, almacenarla, procesarla y generar reportes estadísticos.

Para cada elemento existen diversas alternativas que se describen brevemente en las siguientes secciones.

3.2 Alternativas para el módulo de captura de tráfico

Para su implementación se tienen las siguientes alternativas: Utilización de un router o switch, utilización de un dispositivo hardware dedicado o utilización de una PC con un software. Los aspectos más importantes de cada uno se exponen a continuación.

a. Utilización de un router o switch

Existen sondas de captura de tráfico en software incorporados en routers y/o switches. Asimismo, los sistemas operativos de diversos routers modernos cuentan con soporte de protocolos de exportación de flujos como Netflow o IPFIX.

Estos protocolos junto con el módulo de software del sistema operativo implementan la funcionalidad completa de una sonda de captura de paquetes IP dentro del dispositivo. En la Figura 3.1 se muestra un escenario básico para esta solución utilizando el protocolo Netflow de Cisco.

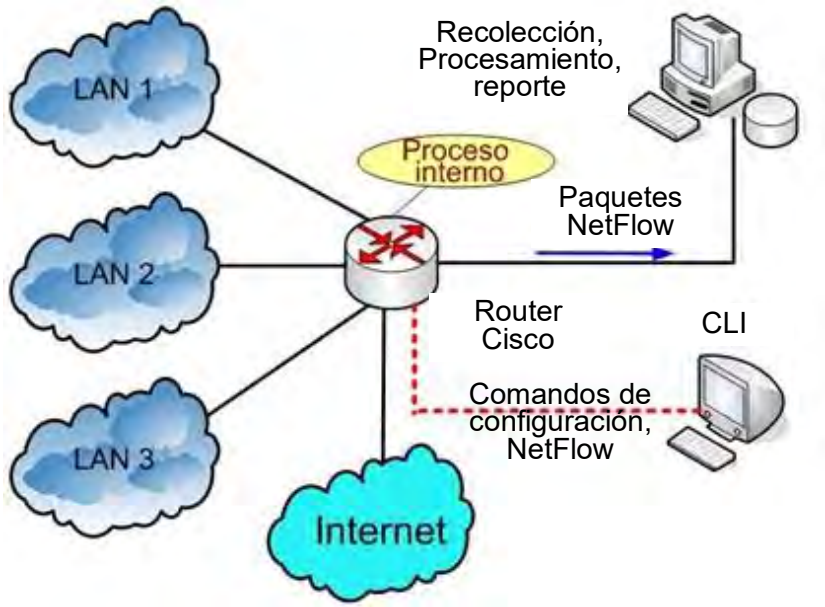


Figura 3.1 Aplicación de un router Cisco con soporte Netflow

En este caso no es necesario utilizar un dispositivo para derivar el tráfico ya que el usuario debe únicamente configurar el equipo (CLI) para activar el protocolo e indicar el dispositivo destino de la información extraída. El router o switch realiza un proceso interno para extraer información del tráfico de una de sus interfaces, almacenarla en una memoria y luego exportarla hacia un equipo remoto.

Existen algunos modelos de enrutadores que incorporan esta funcionalidad, asimismo, cada fabricante utiliza un protocolo diferente. Netflow es un protocolo propietario de Cisco y aunque no es el único, es el más popular ya que está incorporado tanto en enrutadores Cisco como en algunos equipos de otros fabricantes.

Esta solución, implica el uso de un proceso recolector ubicado en un servidor remoto que se encarga de recibir la información para que sea procesada.

b. Utilización de un dispositivo hardware dedicado

Existen dispositivos de hardware dedicado, diseñados solo para realizar la función de captura de paquetes, realizar el pre-procesamiento y la exportación de registro de flujos utilizando protocolos como Netflow o IPFIX. Entre las disponibles se puede mencionar a nBox [21], tarjetas de captura DAG Endace [22], NinjaBox [23], tarjetas COMBO [24], etc.

El rendimiento de este tipo de dispositivos es mucho mayor que cualquier otra solución de captura. Los dispositivos mencionados son capaces de capturar paquetes sin pérdida en enlaces de red con velocidades superiores a 1 Gbps.

La desventaja es su elevado costo y además no se encuentran disponibles directamente en el mercado local. La mayoría de ellos utilizan solamente interfaces de fibra óptica. En la Figura 3.2 se muestran algunos modelos de los dispositivos de hardware mencionados.

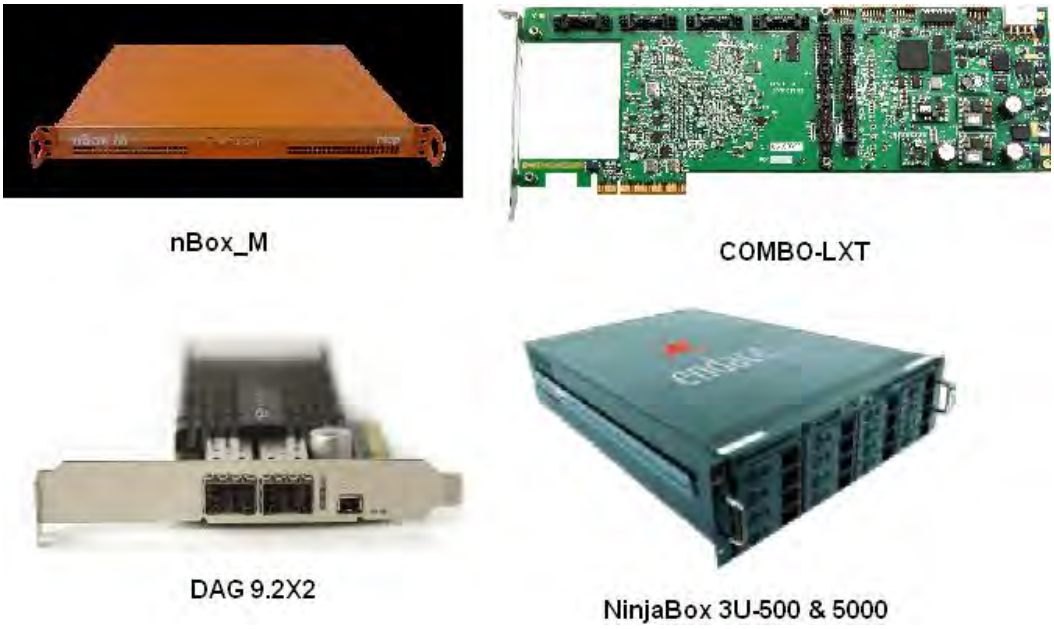


Figura 3.2 Algunos modelos de hardware dedicado

c. Utilización de una PC con un software

Consiste en utilizar una PC con una tarjeta de red convencional (NIC – Network Interface Card) y un software que maneje un protocolo de exportación de flujos.

Existen programas con soporte Netflow v5, v9 e IPFIX de código libre tales como softflowd [25], fprobe-ng [26], fprobe [27] y nProbe [28]. Mayores alcances sobre estos se pueden ver en el Anexo D. Estos programas se instalan en una PC y al igual que un router, se debe activar el proceso de captura y exportación de flujos. El rendimiento de esta solución depende de las características de la PC, de la tarjeta de red y del sistema operativo.

d. Selección de alternativa

Las características ideales de un módulo de captura de paquetes implica que sea portable, ligero y fácil de transportar para que pueda ser ubicado fácilmente en cualquier enlace de red, además de ser flexible en el diseño para poder realizar modificaciones futuras.

Debido a que el propósito del estudio, fue el de diseñar un hardware dedicado, usando como módulo para soporte de diseño a un dispositivo hardware basado en FPGA, se eligió utilizar un dispositivo hardware dedicado como solución a la captura de paquetes. A continuación se explican los motivos por los cuales se opta por esta solución de desarrollo:

- Un hardware dedicado es fácil de implementar en cualquier enlace de red ya que es portable y ligero para su transporte.
- A diferencia de una PC, un hardware dedicado usualmente requiere menores rutinas de mantenimiento. Esto es una importante ventaja sobre todo cuando se tiene un gran número de dispositivos desplegados en distintos puntos de la red.

- El rendimiento de un hardware dedicado es superior en comparación con las otras soluciones, ya que el dispositivo realiza únicamente la función para la cual fue diseñada.
- La programación de la funcionalidad de un FPGA es flexible ya que permite que este dispositivo pueda ser programado muchas veces. Permite realizar correcciones de la funcionalidad de una manera rápida y sencilla.
- El hardware dedicado no es vulnerable a ataques de software malicioso o malintencionado, debido a que no depende de un sistema operativo.
- Un factor determinante, es que en INICTEL-UNI se tiene a disposición una tarjeta de desarrollo con FPGA modelo ML505 de fabricante XILINX (ver Anexo A.1 para mayores detalles) para poder realizar las pruebas. El hecho de contar con una tarjeta de desarrollo es una ventaja ya que permite realizar un diseño personalizado y utilizar solamente los recursos de hardware necesarios para la implementación del módulo de captura.

3.3 Alternativas para la aplicación para modelamiento y reportes

Existen varias alternativas de software tanto gratuitas como comerciales que permiten obtener datos y cifras relacionadas a patrones del tráfico en una red específica. Se citan solamente algunas de ellas, las que son consideradas como las más populares en su categoría. Dentro de la evaluación de alternativas se incluye la herramienta “ntop”, “Netflow Analyzer”, “Certifica Metrix”, “Google Analytics” y el desarrollo de un software a medida.

a. Utilización de la herramienta “ntop”

Ntop [29] es una sonda de tráfico que muestra la utilización de la red. Es portable ya que puede ser ejecutado en cualquier plataforma Unix o Windows. Esta herramienta funciona como un servidor web embebido donde presenta información del tráfico y el estado de la red. Es fácil de utilizar y es apropiado para el monitoreo de varios tipos de redes. Su administración en mediante interfaz web y el uso de recursos de CPU y memoria depende del tamaño de la red y del tráfico. Las funcionalidades que incorpora esta herramienta son las siguientes:

- Ordena el tráfico de red de según varios protocolos.
- Muestra el tráfico de la red de acuerdo a ciertos criterios.
- Presenta estadísticas del tráfico.
- Almacena en disco las estadísticas del tráfico en formato RRD.
- Identifica la identidad de usuarios de computadoras.
- Identifica el sistema operativo de los hosts de manera pasiva.
- Muestra la distribución del tráfico IP entre los varios protocolos.
- Analiza el tráfico IP y lo ordena según las direcciones de origen y destino.
- Muestra una matriz de subred del tráfico IP. (quien está hablando con quien).
- Funciona como colector NetFlow o SFlow para los flujos generados por routers y

Switches.

b. Utilización de la herramienta "Netflow Analyzer"

NetFlow Analyzer [30] es una herramienta basada en Web que sirve para el análisis y control de la red y del tráfico. Es un recopilador, analizador y motor de informes que utiliza los protocolos NetFlow, sFlow, JFLow y otros similares. Son casi 4000 empresas que utilizan NetFlow Analyzer para obtener una visibilidad profunda del tráfico de sus redes y sus patrones. Muestra el comportamiento de la red en tiempo real y el impacto del tráfico sobre la salud general de la red. Las funcionalidades que incorpora esta herramienta son las siguientes:

- Analiza el tráfico de la red con granularidad (nivel de detalle) de un minuto. Muestra detalles del tráfico entrante, saliente, velocidad, volumen de paquetes y utilización del ancho de banda.
- Genera informes de manera automática donde se puede ver el tráfico, la aplicación, el origen, el destino y otras características más. Estos informes pueden ser exportados con formato PDF, CVS o ser enviados por correo electrónico. También permite programar los informes para que se generen de manera periódica.
- Permite crear perfiles de alerta que pueden ser notificados por correo electrónico.
- Permite conocer el rendimiento de la red.
- Evalúa el impacto de las diferentes aplicaciones analizando el ancho de banda consumido por cada aplicación y la distribución de protocolos.
- Valida las políticas de calidad de servicio y sus efectos.
- Controla y optimiza el ancho de banda.
- Detecta tráfico WAN no autorizado.
- Facilita la solución de incidentes en la red.

c. Utilización de "Certifica Metrix"

"Certifica Metric" [31] es una solución integral que permite mejorar la efectividad de una empresa, registrando el comportamiento que los usuarios tienen dentro de un sitio web por medio del monitoreo permanente. "Certifica" brinda un servicio de medición y certificación de audiencia para Internet mediante un software especial el cual determina indicadores de tráfico de los sitios web asociados. Esta solución recoge los siguientes índices:

- Indicadores de tráfico: visitantes únicos, páginas, minutos, sesiones.
- Comportamiento y hábitos del visitante: procedencia de sitios y/o países.
- Palabras claves utilizadas para llegar al sitio web.
- Rutas más frecuentes dentro del sitio web.
- Variaciones considerables del tráfico.
- ISPs (Internet Service Providers) más utilizados por los visitantes.

- Navegadores utilizados por los visitantes.
- Análisis de evolución y ventas.

d. Utilización de "Google Analytics"

Google Analytics es una solución comercial de análisis web para empresas. Proporciona valiosa información sobre el tráfico de sitios web y la eficacia del plan de marketing. Contiene funciones potentes, flexibles y fáciles de utilizar las cuales permiten observar y analizar el tráfico. Google Analytics ayuda al diseño de anuncios personalizados, a la mejora de iniciativas de marketing y a crear sitios web que generen más ganancias [32].

Google Analytics es un producto independiente de Google que muestra información sobre visitantes de una página web tales como [33]:

- Cuántos visitantes recibe un sitio web.
- De dónde provienen las visitas.
- Qué páginas observan.
- Cuánto tiempo navegan por el sitio web.
- A dónde se dirigen cuando abandonan el sitio web.

Toda esta información permite identificar dónde se está perdiendo clientes y dónde realizar acciones eficaces para atraer más clientes.

e. Desarrollo de un software a medida

Esta solución implica la creación de una aplicación de software que cumpla exactamente con los objetivos planteados y que sea ejecutado en un hardware adecuado. Esto implica la programación de un software portable que reciba los paquetes exportados por el módulo de captura y descifre la información contenida en ellos. Para cumplir los objetivos, este software debe procesar esta información según el procedimiento descrito en [1] y obtener los resultados e indicadores para cada servicio, detallados en dicha referencia, tales como:

- Número total de flujos.
- Número total y promedio de Bytes.
- Número total y promedio de paquetes.
- Coeficientes de variación del número total de bytes y de paquetes de los flujos.

Además, es necesario que el software deba contener una base de datos donde se almacenan los resultados y una interfaz web en el cual se presentan los indicadores de producción de servicios a los usuarios interesados.

f. Selección de alternativa

Cada una de las alternativas anteriormente descritas, poseen sus propias ventajas y desventajas. A continuación se explica las razones por las cuales se elige la

implementación de un software a medida como opción para el proyecto.

- La programación de un software a medida es el único medio por el cual es posible procesar información de paquetes según la metodología descrita en la referencia [1]. Las otras opciones procesan a información del tráfico según su propio criterio el cual, algunos por su naturaleza comercial, no se puede determinar fácilmente el método que utilizan.
- Un software personalizado puede ser adecuado para descifrar correctamente cualquier formato particular de carga útil en los paquetes que exporta el módulo de captura hacia el servidor. En las otras alternativas se utilizan protocolos de exportación de flujos como Netflow o IPFIX (ver sección 2.3) para recibir información, sin embargo, estos protocolos utilizan un formato estandarizado el cual no se considera en la solución del proyecto.
- Un software personalizado posibilita no solamente implementar una metodología particular, también permite extender su funcionalidad sin limitaciones. Asimismo la elección de un lenguaje de programación adecuado facilita la interacción con distintos motores de base de datos tales como MySQL, PostgreSQL, Oracle, etc.

CAPÍTULO IV
DISEÑO DEL MÓDULO DE CAPTURA DE TRÁFICO

Este capítulo se enfoca a exponer el diseño del Módulo de Captura de Tráfico. Se presentan las secciones correspondientes al diseño del Módulo de Captura de Tráfico (4.1-diseño de la estructura de hardware del sistema, 4.2-desarrollo del software de aplicación del sistema). El capítulo concluye con la presentación de las pruebas de validación del diseño (4.3).

El diseño integral del módulo de captura está compuesto de dos partes claramente diferenciadas. La primera parte consiste en el diseño de la estructura interna del FPGA que viene incorporado en la tarjeta de desarrollo ML505 de Xilinx (ver sección A.1 de Anexo A) la cual fue utilizada como plataforma de hardware para implementar la funcionalidad del módulo de captura. La segunda parte consiste en el diseño de la aplicación de software del sistema embebido (ver sección 2.1.1) la cual se ejecuta sobre el hardware diseñado en la primera parte.

4.1 Diseño de la estructura de hardware del sistema

En esta sección se exponen los detalles relacionados al diseño del hardware del módulo de captura en un FPGA de Xilinx. La información está organizada de la siguiente manera:

- Descripción de la herramienta de software utilizada, la EDK 11.3 de Xilinx.
- Configuración básica del hardware.
- Ajustes en el diseño del hardware con procesador.
- Descripción del sistema diseñado.
- Exportación a SDK (Software Development Kit).

4.1.1 Herramienta EDK 11.3 de Xilinx

Para el diseño del módulo de captura se utilizó la herramienta de software EDK [35] versión 11.3 tanto para el diseño del hardware como del software. Esta herramienta es propiedad de Xilinx y tiene la ventaja de facilitar el proceso de diseño de un sistema embebido con procesador (SoC-System on Chip) basado en FPGA. La herramienta EDK tiene los siguientes componentes:

- XPS (Xilinx Platform Studio): Herramienta de software utilizado para el diseño de la porción de hardware del sistema.
- SDK (Software Development Kit): Entorno de desarrollo integrado complementario a XPS,

el cual es utilizado para la creación y verificación de aplicaciones de software embebido.

- IP cores [37] para el procesador MicroBlaze de Xilinx.
- Drivers y bibliotecas para el desarrollo de software embebido.
- Depurador y compilador de C/C++ con licencia GNU diseñado exclusivamente para los procesadores MicroBlaze (ver sección A.3 de Anexo A) y PowerPC.
- Documentación y Ejemplos de proyectos.

La herramienta XPS de EDK pone a disposición del diseñador un conjunto de IP cores (bloques de hardware que cumplen una función específica) diseñados adecuadamente por Xilinx para ser implementados en un modelo específico de FPGA que viene incluido en un determinado modelo de tarjeta de desarrollo del mismo fabricante, en este caso Xilinx. Asimismo, XPS permite estructurar, de manera personalizada, la topología interna del FPGA conectando cada IP core que se desea utilizar en el diseño con los buses de datos internos.

Cabe resaltar que cada IP core incluido en la herramienta EDK es propiedad de Xilinx y la herramienta XPS no permite observar el código VHDL de cada uno; sin embargo, es posible adquirir productos de Xilinx (software e IP cores) sin cargos únicamente para fines de evaluación. Para el diseño del módulo de captura se adquirió una licencia de evaluación la cual no supone realizar pago alguno para el uso de los productos de software (EDK) de Xilinx por el lapso de 30 días y también para la evaluación de IP cores por 120 días [40].

El proceso de diseño de un sistema embebido basado en un FPGA utilizando la herramienta EDK de Xilinx, se resume en el diagrama de bloques mostrado en la Figura 4.1, la cual fue extraída de la documentación del fabricante.

Como se muestra en dicha figura, EDK también integra la herramienta ISE [38] (Integrated Software Environment) de Xilinx para el diseño de IP cores utilizando un lenguaje de descripción de hardware ó HDL (Hardware Description Language) tales como VHDL o Verilog. Asimismo realiza la síntesis, simulación, implementación y descarga del diseño en el FPGA. Los cuadros que están en color verde son los pasos que se efectuaron para el diseño del módulo de captura.

Los IP cores diseñados por Xilinx, permiten reducir de manera considerable el tiempo del proceso de diseño del hardware debido a que se tienen disponibles un conjunto de bloques funcionales o IP cores de uso común dentro de la herramienta XPS. Además, esta herramienta permite personalizar algunos parámetros de hardware esenciales de cada IP core para permitir que el diseño sea más personalizado.

Para el diseño del módulo de captura se han utilizado determinados IP cores que la herramienta XPS pone a disposición, por tal motivo, no se realizó el diseño de ningún bloque funcional en particular y no fue necesario realizar los pasos de los cuadros que se

indican en color naranja. Los IP cores ya están diseñados por Xilinx para ser implementados en el FPGA de la tarjeta de desarrollo, que en este caso fue el modelo ML505 (Ver sección A.1 de Anexo A).

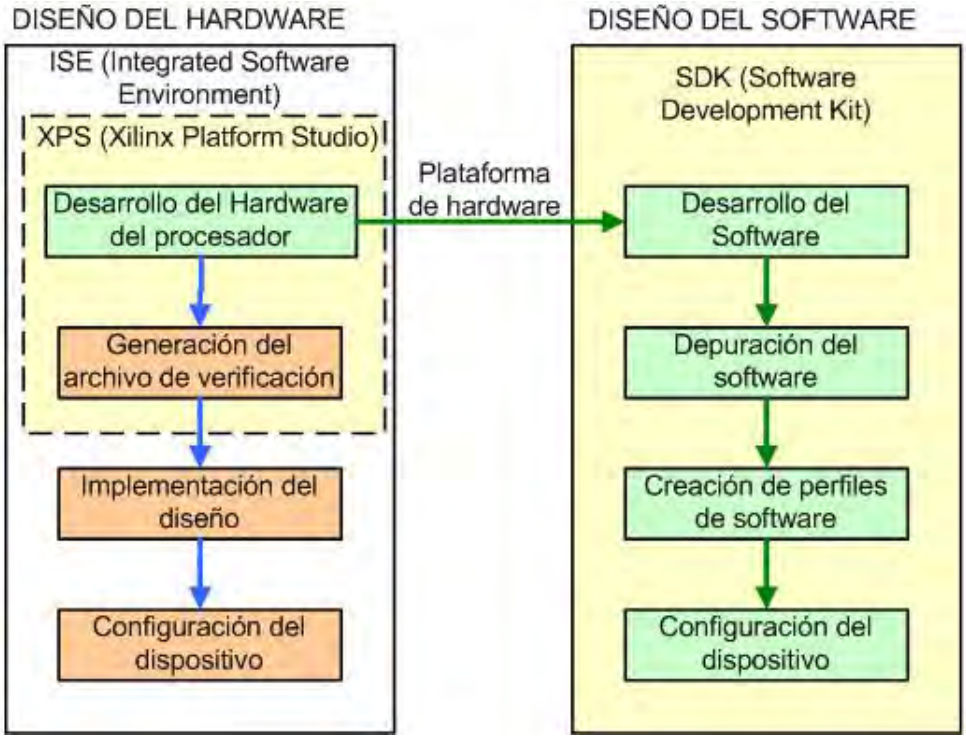


Figura 4.1 Proceso básico de diseño de un sistema embebido en EDK (Fuente: [36])

Para tener una visión más detallada sobre el proceso de diseño de un sistema embebido utilizando la herramienta EDK, se debe consultar la referencia indicada.

4.1.2 Configuración básica del hardware

Para iniciar el proceso de diseño del hardware la herramienta XPS cuenta con un asistente llamado BSB (Base System Builder) que automatiza las tareas comunes de configuración de la plataforma básica de hardware y software en los diseños de sistemas embebidos con procesador.

Después de ejecutar el asistente, se cuenta con un proyecto que contiene todos los elementos básicos necesarios para construir un sistema personalizado y de mayor complejidad [36]. De manera concreta, este asistente permite realizar lo siguiente:

- Crear un nuevo archivo de proyecto.
- Escoger la tarjeta de desarrollo que se va a utilizar en el diseño.
- Seleccionar y configurar el procesador a utilizar (MicroBlaze en este caso).
- Seleccionar y configurar interfaces de entrada/salida.
- Agregar periféricos internos.
- Configurar un software ejemplo.
- Generar un reporte con un resumen del sistema.

La principal ventaja de utilizar el BSB, es que reconoce los componentes de hardware con que cuenta el FPGA así como los componentes de hardware de la tarjeta de desarrollo seleccionada y proporciona las opciones de configuración básicas para cada componente seleccionado para el diseño.

A continuación se detalla cada uno de los pasos del BSB, los componentes seleccionados (IP cores) y sus configuraciones básicas realizadas con el asistente BSB de la herramienta XPS.

1. Selección de tarjeta de desarrollo:

Se escoge el vendedor de la tarjeta de desarrollo así como el modelo (Tabla 4.1).

Tabla 4.1 Selección de tarjeta de desarrollo (Fuente: BSB)

Parámetro	Valor
Vendedor de la tarjeta	Xilinx
Nombre de la tarjeta	Virtex 5 ML505 Evaluation Platform
Revisión de la tarjeta	1

2. Configuración del sistema:

Se escoge un sistema con procesador único (Tabla 4.2).

Tabla 4.2 Configuración del sistema (Fuente: BSB)

Parámetro	Valor
Tipo de sistema	Single-Processor System

3. Configuración del procesador:

Se escogió y se configuró el procesador que se va a utilizar, en este caso se eligió a MicroBlaze ya que es la única opción disponible para el modelo de FPGA Virtex 5 de la tarjeta utilizada (Tabla 4.3).

Tabla 4.3 Configuración del procesador (Fuente: BSB)

Parámetro	Valor
Tipo de procesador	MicroBlaze
Frecuencia de reloj del sistema	125 MHz
Memoria local	8 KB
Habilitar Unidad de punto flotante	Si

4. Configuración de periféricos:

En este paso se escogió y se configuró cada IP core que estará incluido en el sistema. A continuación se muestran los componentes elegidos y sus configuraciones. (Ver Tabla 4.4 - Configuración de periféricos).

Tabla 4.4 Configuración de periféricos (Fuente: BSB)

Nombre de la Instancia	Nombre del IP Core	Parámetros		Descripción
DDR2_SDRAM	mpmc	No hay parámetros de configuración en BSB para esta instancia		Controlador de memoria multipuerto para la lectura y escritura de la memoria RAM externa y para la implementación de la caché en esta memoria.
Hard_Ethernet_MAC	xps_ll_temac	Uso de DMA (Direct Memory Access)	Si	Bloque de controlador del MAC Ethernet embebido en el FPGA Virtex 5. Este IP core permite que el sistema tenga acceso a una red Ethernet.
		Uso de interrupción	Si	
RS232_Uart_1	xps_uartlite	Velocidad de transmisión	9600 bits/s	UART (Universal Asyncronous Receiver Transmitter) utilizado para imprimir caracteres por el Puerto serial del módulo.
		Cantidad de bits de datos	8 bits	
		Paridad	No habilitado	
		Uso de interrupción	Si	
dlmb_cntrl	lmb_bram_if_cntrl	No hay parámetros de configuración en BSB para esta instancia		Bloques controladores de la memoria local interna. Permite que el procesador MicroBlaze pueda tener acceso a bloques de memoria interna (para datos e instrucciones) que son más rápidos que la memoria principal externa.
ilmb_cntrl	lmb_bram_if_cntrl	No hay parámetros de configuración en BSB para esta instancia		
xps_timer_0	xps_timer	Ancho del contador	32 bits	Bloque de hardware con 2 timers (contador y temporizador a la vez).
		Modo de configuración	2 timers	
		Uso de interrupción	Si	

5. Configuración de la memoria caché:

Se habilitó el uso de la memoria caché tanto de instrucciones como de datos. Se configuró su tamaño al valor de 4KB. Asimismo se indicó que estará ubicada en la memoria

RAM externa tal como indica la Tabla 4.5.

Tabla 4.5 Configuración de la memoria caché (Fuente: BSB)

Parámetro	Valor
Tamaño de la memoria caché de instrucciones	4KB
Memoria caché de instrucciones	DDR2_SDRAM
Tamaño de la memoria caché de datos	4KB
Memoria caché de datos	DDR2_SDRAM

6. Configuración de las aplicaciones ejemplo:

Se habilitó el uso de aplicaciones ejemplo que sirven para verificar en primera instancia el correcto funcionamiento de los periféricos y de las memorias.

Tabla 4.6 Configuración de las aplicaciones ejemplo (Fuente: BSB)

Aplicación		Valor opcional
Entrada y salida estándar		RS232_Uart_1
Test de memoria		TestApp_Memory_microblaze_0
	Instrucciones	ilmb_cntlr
	Datos	dlmb_cntlr
Test de Periféricos		TestApp_Peripheral_microblaze_0
	Instrucciones	DDR2_SDRAM

7. Resumen del sistema:

El asistente BSB muestra al final un resumen con cada uno de los IP cores elegidos y los rangos de direcciones de memoria virtual que maneja el procesador para el control de cada uno de ellos, tal como se muestra en la Tabla 4.7.

Tabla 4.7 Resumen del sistema (Fuente: BSB)

Nombre del IP core	Nombre de la Instancia	Dirección base	Dirección alta
Processor 1	microblaze_0		
mpmc	DDR2_SDRAM	0x50000000	0x5FFFFFFF
xps_ll_temac	Hard_Ethernet_MAC	0x81C00000	0x81C0FFFF
xps_uarlite	RS232_Uart_1	0x84000000	0x8400FFFF
lmb_bram_if_cntlr	dlmb_cntlr	0x00000000	0x00001FFF
lmb_bram_if_cntlr	ilmb_cntlr	0x00000000	0x00001FFF
xps_timer	xps_timer_0	0x83C00000	0x83C0FFFF

4.1.3 Ajustes en el diseño del hardware con procesador

El BSB genera un conjunto de archivos de diseño en lenguaje VHDL [39] que describe el hardware del sistema del módulo de desarrollo con cada componente integrado y con parámetros de configuración por defecto. Sin embargo, para poder optimizar su funcionalidad, la herramienta XPS permite realizar ciertos ajustes o configuraciones personalizadas a todas las instancias generadas por el BSB. La ventaja de utilizar esta herramienta es que el diseñador puede realizar los ajustes mediante una interfaz en modo

gráfico sin necesidad de realizar los ajustes directamente en el código VHDL.

A continuación se indica cada una de las instancias que fueron modificadas y los ajustes que se realizaron a cada una.

a. Ajustes en la instancia del procesador MicroBlaze: microblaze_0

Para mejorar la rapidez del software de aplicación que es ejecutada por el procesador, se realizan ciertos ajustes en la instancia del procesador MicroBlaze.

- Se habilita el barrel shifter: Esto mejora dramáticamente el desempeño de una aplicación, pero incrementa el tamaño del procesador. Al ser activado, el compilador utilizará automáticamente las instrucciones relacionadas a este componente. Un barrel shifter es un registro de cambio con la rotación de bits. Los bits son cambiados un número deseado de posiciones de bits en un solo ciclo de reloj [41].
- Se habilita el divisor de enteros: Es un circuito hardware que mejora el desempeño de las aplicaciones que realizan división de enteros. Al igual que el barrel shifter, este hardware incrementa el tamaño del procesador. Al ser activado, el compilador utiliza automáticamente las instrucciones relacionadas a este componente.
- Se habilita el comparador de patrones: Se habilitan 2 instrucciones de comparación de patrones que mejoran el rendimiento de operaciones relacionadas a la comparación de patrones y cadenas de texto. Al ser activado, el compilador utiliza automáticamente las instrucciones relacionadas a este componente.
- Se habilita las excepciones generadas por la unidad de punto flotante: El FPU (Floating Point Unit) arroja excepciones para las condiciones de underflow, overflow, división por cero u operaciones ilegales. El diseñador proporciona su propio manejador de excepciones.
- Se habilita las excepciones de división por cero: Se arroja una excepción en caso de que el divisor en una división es cero. El diseñador proporciona su propio manejador de excepciones.
- Se habilita las excepciones del bus PLB tanto en datos como en instrucciones: Se arroja una excepción si es que existe un error o un límite de tiempo alcanzado en el bus PLB. El diseñador proporciona su propio manejador de excepciones.

Las demás opciones de configuración se dejan por defecto ya que se considera que su variación no afecta de manera sustancial el rendimiento de la ejecución del programa que se ejecuta en el procesador MicroBlaze.

b. Ajustes en la instancia Hard_Ethernet_MAC

Este es el componente de mayor importancia, ya que implementa la capa de enlace de datos del modelo OSI [42] y es la interfaz por donde el procesador envía y recibe los paquetes. El ajuste que se realiza para tener el mayor rendimiento posible y de acuerdo

con las opciones que se tienen disponibles en el XPS, es incrementar el tamaño de los buffers de transmisión, así como los de recepción, al valor máximo disponible de 32768 Bytes. Esta es la única modificación que se realiza. Cabe resaltar que este componente es un hardware que ya viene embebido dentro del FPGA utilizado. En este caso, el IP core utilizado para este hardware sirve como interfaz entre el bus del sistema y el mismo hardware. Este componente es ajustado ya que posee interfaces que permiten que algunos parámetros del mismo sean configurados. (Ver sección A.4 del Anexo A).

El tipo de interfaz física para este componente es GMII (Gigabit Media Independent Interface) el cual permite utilizar velocidades de 10/100/1000 Mbps. No se habilita la función de cálculo del checksum de TCP y UDP en hardware tanto en la transmisión como en la recepción con la finalidad de ahorrar recursos del FPGA y ciclos de reloj.

c. Ajustes en el controlador de interrupciones

Por defecto el BSB crea una instancia llamada xps_intc_0 [43] el cual corresponde al controlador de interrupciones del sistema. Esta instancia se encarga de expandir el número de entradas de interrupciones disponibles al CPU y opcionalmente provee un esquema de prioridad para las interrupciones que puedan generar cada dispositivo que tenga esta capacidad. El BSB genera por defecto una lista con un nivel de prioridad para la atención de las peticiones de interrupción de cada dispositivo. Para el diseño, se modifica dicha lista para quedar así:

Tabla 4.8 Nivel de prioridad para las interrupciones (Fuente: Software XPS)

Pin conectado	Prioridad
fpga_0_Hard_Ethernet_MAC_PHY_MII_INT_pin	1
Hard_Ethernet_MAC_TemacIntc0_Irpt	2
DDR2_SDRAM_SDMA2_Tx_IntOut	3
DDR2_SDRAM_SDMA2_Rx_IntOut	4
xps_timer_0_Interrupt	5
RS232_Uart_1_Interrupt	6

Se puede observar que la primera prioridad la tiene el pin de entrada de interrupción proveniente del dispositivo MAC de capa física que es un circuito integrado externo al FPGA que se encarga de recibir y transmitir los paquetes desde y hacia la red. Es un circuito integrado transceptor que implementa la capa física Ethernet [44].

Se le otorga la mayor prioridad a esta interfaz ya que la función principal del sistema es capturar y enviar paquetes todo el tiempo. Con esta configuración se reduce la probabilidad que algunos paquetes sean descartados.

4.1.4 Descripción del sistema diseñado

La Figura 4.2 muestra el archivo de imagen generado por la herramienta XPS el cual representa de una manera más concreta como está diseñado el sistema.

Esta herramienta realiza de manera automática la conexión de todas las instancias a

los buses del sistema: bus de memoria local de datos e instrucciones LMB (Local Memory Bus) [45] y el bus del procesador PLB (Processor Local Bus) [46].

Tiene la opción de generar una imagen con un diagrama de bloques, lo que permite representar de manera gráfica (Figura 4.2) la ubicación del procesador MicroBlaze, las memorias internas, los periféricos, el controlador de la memoria, el generador de reloj, el controlador de interrupciones y los buses que interconectan cada IP core del sistema.

La herramienta XPS contiene 3 pestañas “Bus Interfaces”, “Ports” y “Addresses”. En cada una de ellas se muestran cada IP core del sistema detallando las conexiones con los buses, sus puertos y el mapeo de direcciones con el procesador respectivamente.

La Figura 4.3 muestra la pestaña “Bus Interfaces”, en la cual el diseñador configura de manera explícita el bus al cual va conectado cada componente. En la parte izquierda se muestra un esquema gráfico indicándose cada uno de los buses con diferente color y las conexiones con cada IP core del diseño.

La Figura 4.4 muestra la pestaña “Ports” en la cual se puede configurar de manera explícita las conexiones internas (entre IP cores) así como las conexiones externas (puertos externos del FPGA). La Figura 4.5 muestra en mayor detalle los pines del FPGA (puertos externos) generados automáticamente por el BSB y también se muestra los nombres de cada uno de los pines que son utilizados como referencia en los archivos de diseño. Asimismo, se muestra la dirección de cada pin, los que son de entrada, de salida o ambos. Como es lógico se muestran pines de conexión con el puerto serial, la memoria externa, el MAC Ethernet externo y el generador de reloj del sistema.

La Figura 4.6 muestra a cada IP core del diseño con su respectivo rango de direcciones virtuales que el procesador utiliza para interactuar con cada uno. También el tamaño de cada rango, la interfaz del bus, el nombre del bus y el tipo y versión de de IP core.

Luego de haber realizado el diseño del sistema, el software XPS sintetiza, implementa y genera el archivo de programación para programar el FPGA. Como resultado de estos procesos la plataforma XPS genera archivos conteniendo diversa información relacionada al proyecto.

XPS también genera varios reportes donde se resume las cantidades y porcentajes de utilización de recursos del FPGA del diseño. Asimismo, se genera un reporte conciso donde se resume cada componente de todo el diseño en formato HTML. Ambos reportes se muestran en el Anexo B (Reportes del diseño del módulo de captura generados por la herramienta XPS).

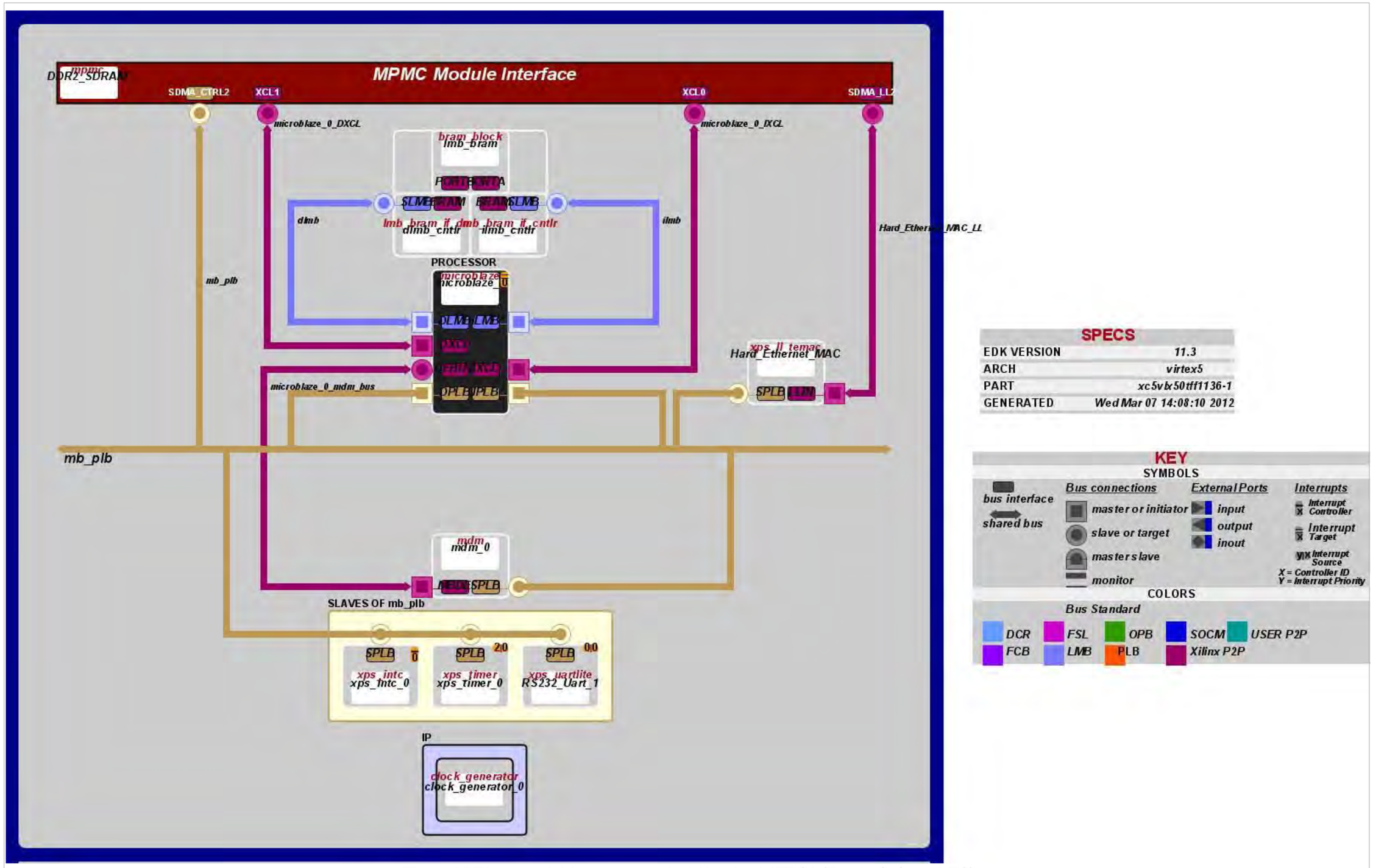


Figura 4.2 Diagrama de bloques generado del interior del FPGA (Fuente: Software XPS)

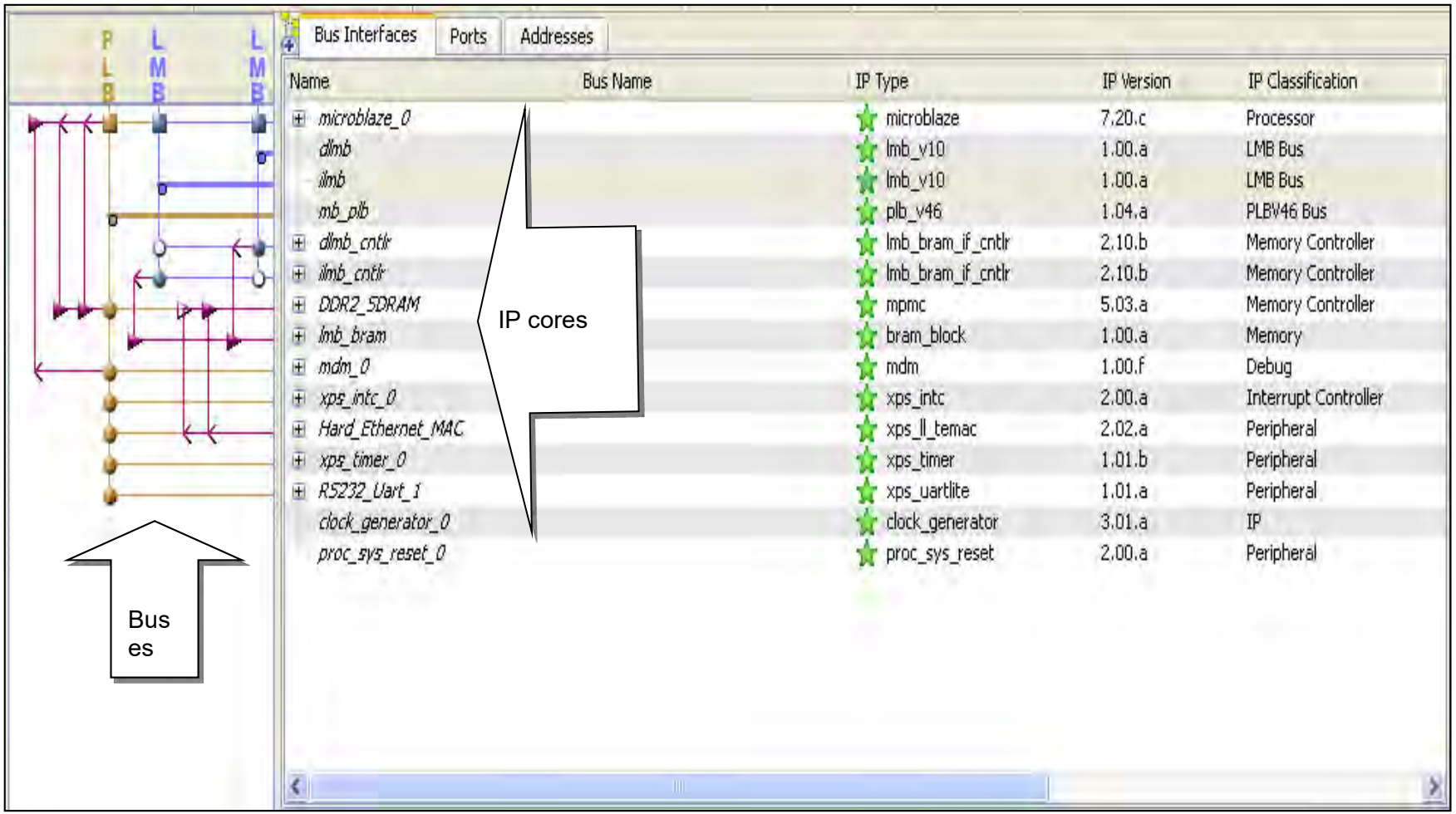


Figura 4.3 Pestaña de configuración de las conexiones a los buses (Fuente: Software XPS)

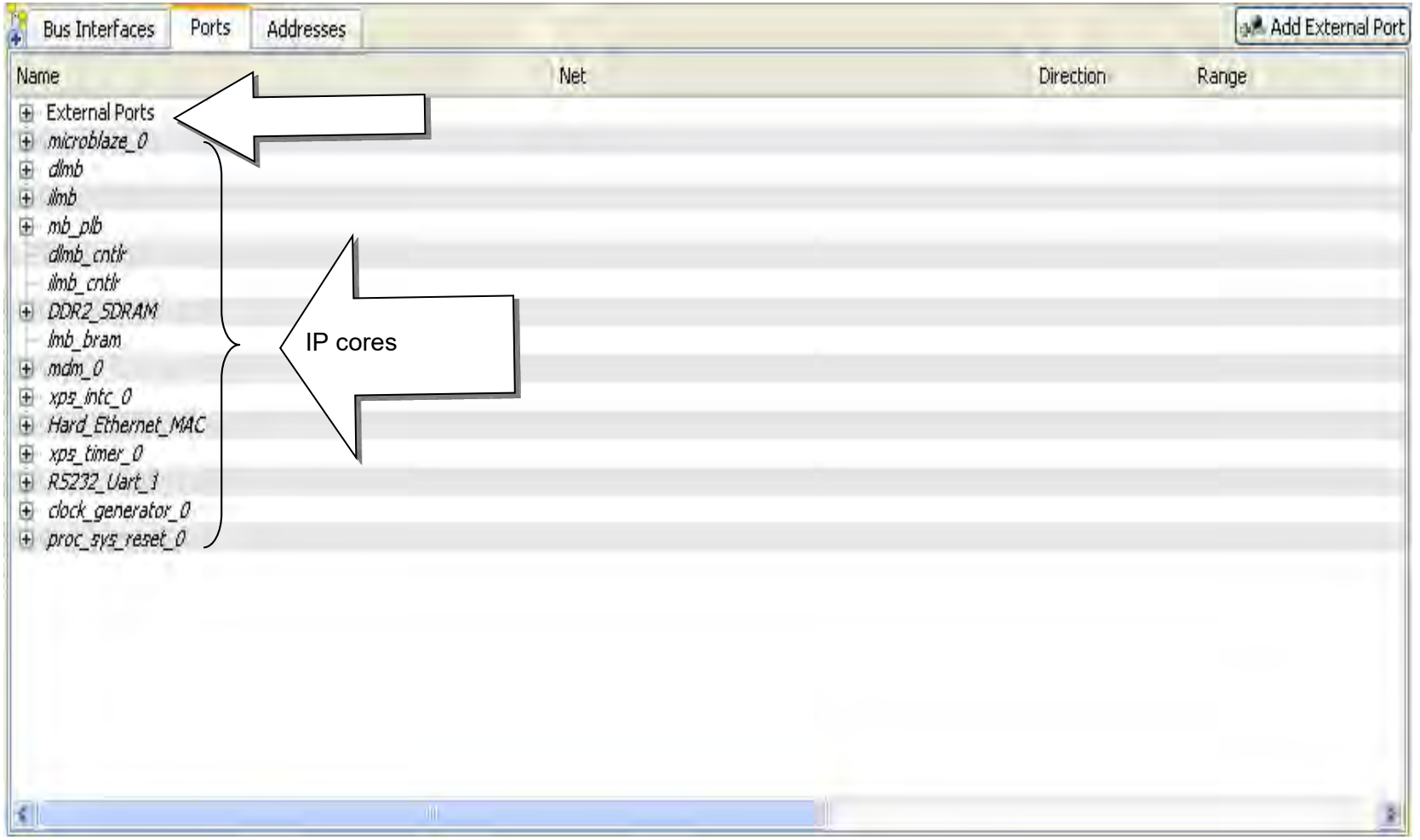


Figura 4.4 Pestaña de configuración de puertos de cada IP core (Fuente: Software XPS)

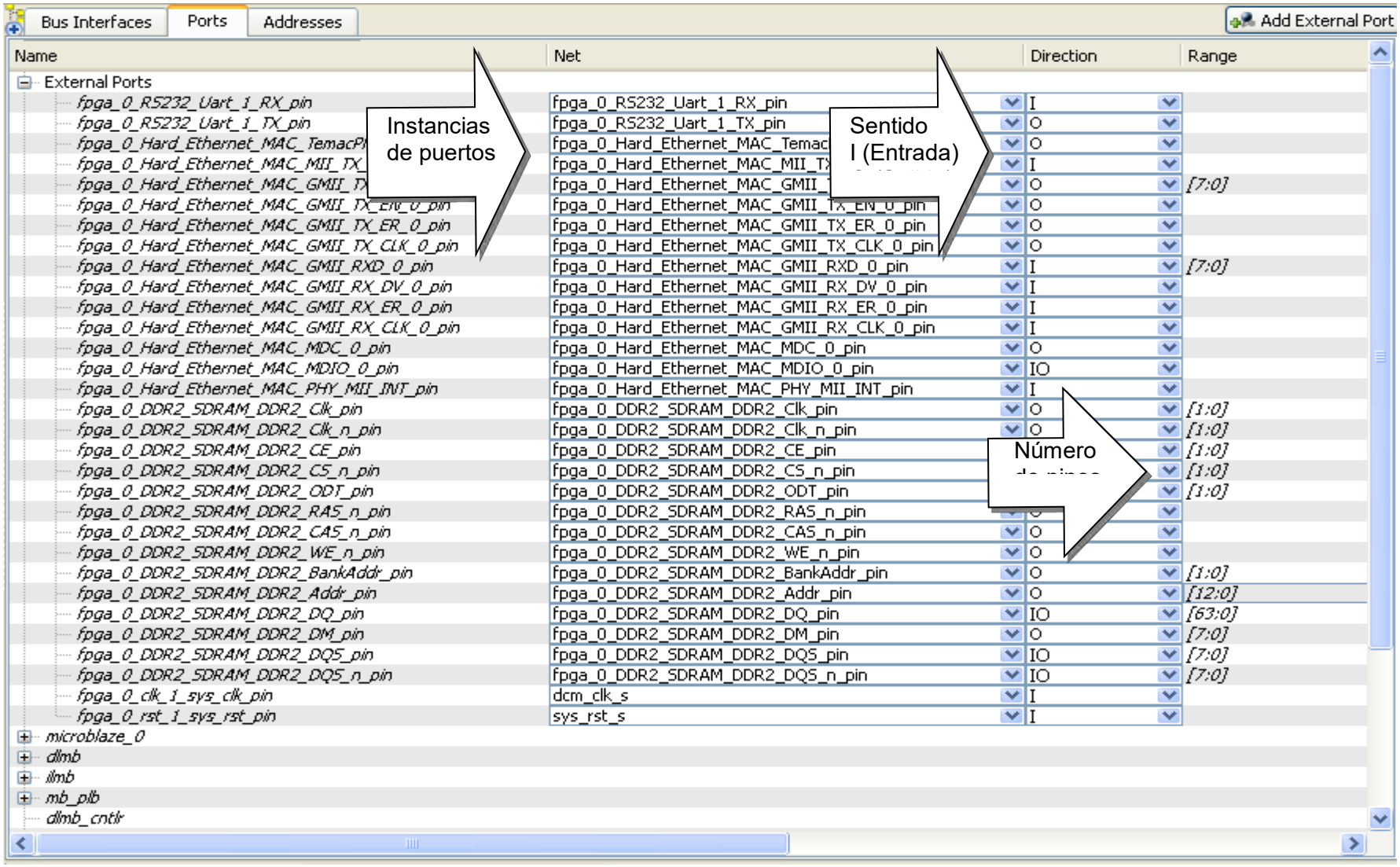


Figura 4.5 Puertos externos del FPGA (Fuente: Software XPS)

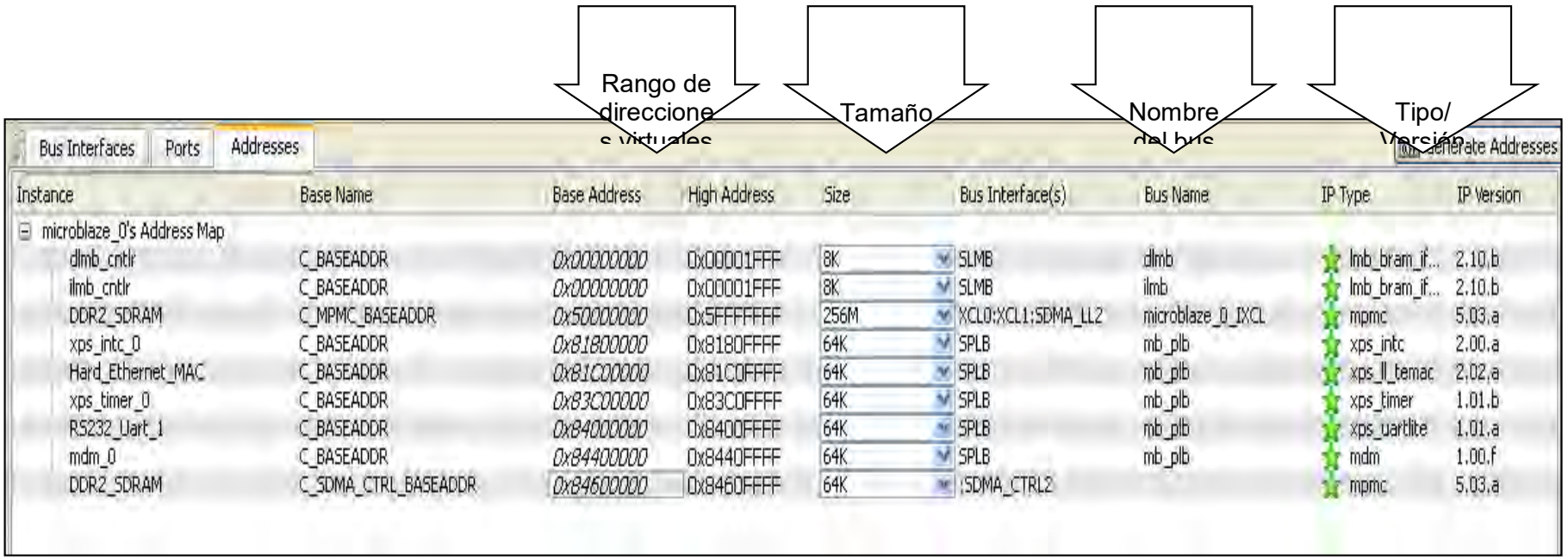


Figura 4.6 Direcciones virtuales de cada IP core para el procesador MicroBlaze (Fuente: Software XPS)

4.1.5 Exportación a SDK

Luego del diseño de la plataforma de hardware, se procede a diseñar el software que será ejecutado en ella. Para esto es necesario primero exportar desde XPS la información del hardware obteniéndose así un archivo de formato XML (Extensible Markup Language), así como una serie de archivos, los que luego serán accedidos por la plataforma SDK.

4.2 Desarrollo del software de aplicación del sistema

Esta sección está organizada de la siguiente manera:

- Definición del formato de un registro de información de flujo.
- Proceso de extracción de información de las cabeceras de cada paquete.
- Proceso de agregación de paquetes en flujos.
- Proceso de exportación de información de flujos hacia un servidor externo.
- Resumen de la aplicación de software.

Para lograr los objetivos la solución de hardware necesita extraer la información que se muestra en la Tabla 4.9.

Tabla 4.9 Información a extraer

Información	Propósito
Dirección IP origen y dirección IP destino	Conocer la procedencia del cliente o del servidor que tiene asignada esta dirección.
Puerto origen y puerto de destino	Identificar la aplicación que está formando parte del proceso de comunicación entre el cliente y el servidor.
Protocolo	Determinar el protocolo de nivel superior a la capa de red. Para el caso son solo dos posibilidades: TCP y UDP.
Número de bytes	Determinar la cantidad de información en volumen que es intercambiada entre el cliente y el servidor.
Cantidad de paquetes	Medir el impacto de los paquetes de un flujo particular en el tráfico que se va a analizar.

Este capítulo está orientado al diseño de la solución de hardware o “Módulo de Captura”´. Se planteó como objetivo (sección 1.2) que esta solución capture paquetes IP, realice un preprocesamiento y sus resultados sean enviados hacia un ordenador.

Para cumplir los objetivos y requerimientos mencionados, se establece que el módulo de captura realice básicamente tres funciones principales:

1. Extracción de información de las cabeceras de cada paquete.
2. Agregación de paquetes en flujos.
3. Exportación de información de flujos hacia un servidor externo.

Para cada una de las funciones mencionadas, se ha empleado una metodología idónea para la programación del software que se ejecuta sobre el hardware previamente diseñado. Preliminarmente al desarrollo de las funciones principales, se define el registro de información de flujo y su formato, el cual es necesario almacenar información de cada flujo en el tráfico. Además se explica en forma concisa cada una de las metodologías empleadas

así como los respectivos algoritmos. Todos los pasos explicados en esta sección son implementados en la plataforma de software SDK.

4.2.1 Definición del formato de un registro de información de flujo

Antes de iniciar con el proceso de captura y extracción de información, se debe definir de manera clara los datos específicos (información de cabeceras) que se pretenden extraer de cada paquete. Por tal motivo en esta subsección se define cuales son estos datos a extraer así como el formato que va a tener un registro de información de flujo que será almacenado en un registro temporal.

En el presente trabajo de tesis, se propone un formato específico para el registro de información de un flujo en particular que se observa en cualquier nodo o enlace de red. Este formato contiene algunos de los campos que están incluidos en la mayoría de los protocolos de exportación de flujos existentes actualmente (ver sección 2.3).

Se concibió el formato mostrando en la siguiente figura porque contiene únicamente la información necesaria y suficiente que requiere el servidor central de procesamiento para obtener la información estadística referente a la producción de servicios. El formato propuesto es mostrado en la Figura 4.7.

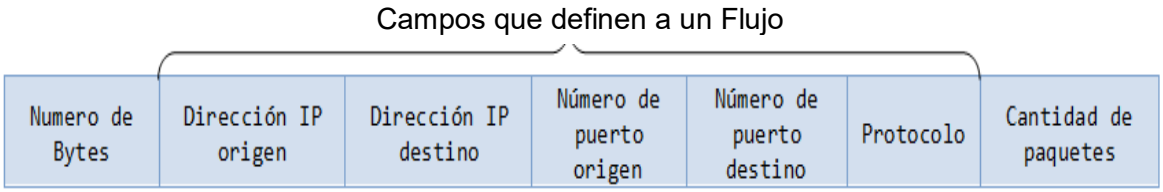


Figura 4.7 Formato del registro de información de flujo

A continuación se muestra la definición de flujo utilizado en la presente Tesis, la cual fue extraída de la RFC 3917 [47]: “Un flujo es definido como un conjunto de paquetes IP atravesando un punto de observación en la red durante cierto intervalo de tiempo. Todos los paquetes que pertenecen a un flujo en particular tienen un conjunto de propiedades comunes”.

Las propiedades comunes que se consideran en la presente tesis son cinco campos de cabeceras de paquetes, los mismos que se muestran en la Figura 4.7 resaltados por la etiqueta “Campos que definen a un Flujo”.

En el presente trabajo de tesis se aplica la metodología de análisis de tráfico de paquetes IP basado en flujos debido a que [47] [50]:

- Existen diversas aplicaciones que requieren análisis de tráfico IP basado en flujos.
- El volumen de datos que se analiza es mucho menor en comparación con el análisis basado en paquetes.
- Las redes actualmente crecen en volumen y complejidad y requieren un método de análisis de tráfico simplificado.

- Es apropiado para redes de alta velocidad.
- Permite efectuar diferentes niveles de análisis del tráfico.
- Facilita el análisis del comportamiento de la red, de las aplicaciones y de los hosts.
- Facilita el análisis de la seguridad y del rendimiento de la red.

Los demás campos del formato propuesto (Número de bytes y cantidad de paquetes) sirven para contabilizar el acumulado de la cantidad de información (en bytes) y de la cantidad de paquetes del conjunto que pertenecen a un flujo en particular. Considerando todos estos campos se satisface los requerimientos del módulo de captura relacionados al tipo de información a capturar.

A continuación se van a detallar cada uno de los procesos involucrados en el software de aplicación del módulo de captura.

4.2.2 Proceso de extracción de información de las cabeceras de cada paquete

Este proceso consiste en la identificación automática de los campos de cabeceras requeridos de un paquete IP y el almacenamiento temporal de tales campos utilizando el formato propuesto previamente.

La metodología empleada para la implementación de este proceso consiste principalmente en el estudio, análisis y modificación de la pila de protocolos LWIP (Ver sección 2.1). Se va a explicar el proceso sólo para un paquete entrante ya que para los subsiguientes el proceso es el mismo. El proceso aplicado a un paquete entrante se resume de manera simple en la Figura 4.8.

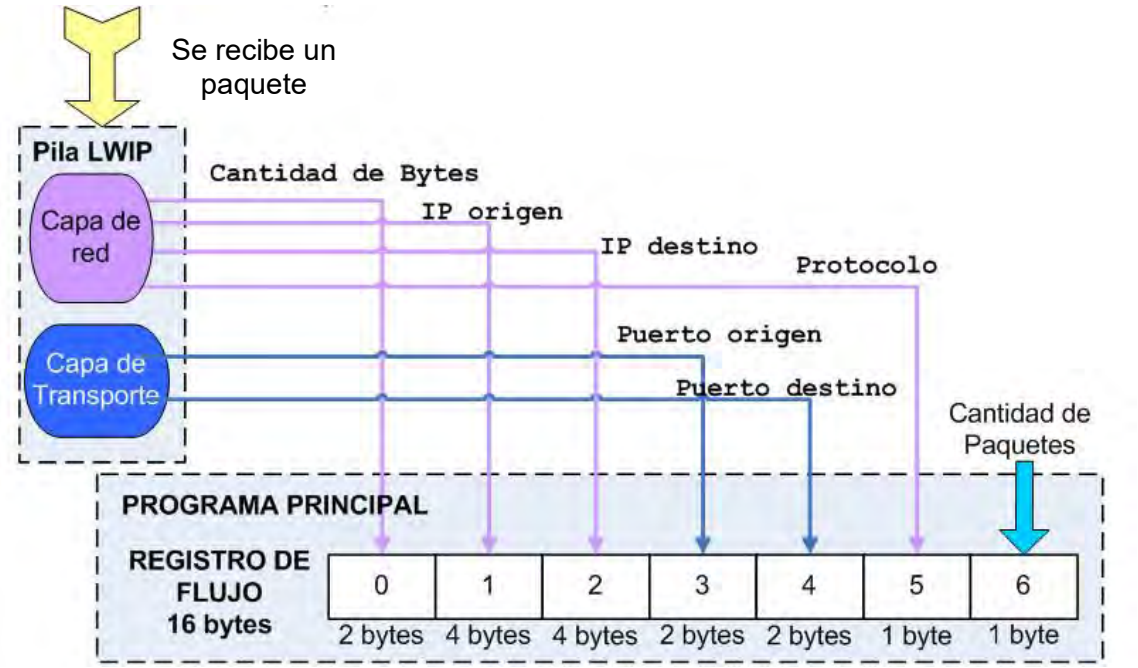


Figura 4.8 Proceso de extracción de información de cabeceras

De manera más concreta, la metodología consiste en la ubicación de la fase de identificación y procesamiento de cada uno de los campos requeridos dentro del código de

programa de la pila de protocolos LWIP. Se realiza una modificación del código (agregación de código adicional) en cada una de estas fases. El código agregado se encarga de almacenar cada campo en una variable temporal, los cuales posteriormente son ordenados por el programa principal según el formato propuesto para el registro de información de flujo.

A continuación se brinda mayores detalles concernientes a la extracción de cada uno de los campos de un paquete entrante. Esta subsección está organizada de la siguiente manera:

- Definición de variables de registro de flujo.
- Extracción y almacenamiento de la cantidad de bytes.
- Extracción y almacenamiento de las direcciones IP origen y destino y del campo protocolo.
- Extracción y almacenamiento de los puertos origen y destino.

a. Definición de variables de registro de flujo

En la estructura del código de la pila de protocolos LWIP se define una estructura llamada “netif” [48] la cual representa al driver del dispositivo de interfaz de red del módulo hardware.

Debido a que en casi todo el código de la pila LWIP se tienen funciones que hacen referencia a esta estructura y/o la utilizan como una variable de entrada, las variables miembro de la estructura “netif” son accesibles en varias ubicaciones del código. Por tal motivo, se agregó las siguientes variables en la definición de esta estructura (Tabla 4.10).

Tabla 4.10 Variables miembro agregadas a la estructura “netif”

Nombre de Variable	Tipo de Variable	Tamaño de la variable	Descripción
longitud	unsigned short	2 bytes (0 → 65535)	Variable donde se almacena la longitud del paquete en bytes.
ip_origen	unsigned char	Arreglo de 4 caracteres de 1 byte cada uno	Arreglo donde se almacena la dirección IP origen del paquete.
ip-destino	unsigned char	Arreglo de 4 caracteres de 8 bytes cada uno	Arreglo donde se almacena la dirección IP destino del paquete.
puerto_origen	unsigned short	2 bytes (0 → 65535)	Variable donde se almacena el número de puerto origen del paquete entrante.
puerto_destino	unsigned short	2 bytes (0 → 65535)	Variable donde se almacena el número de puerto destino del paquete entrante.
protocolo	unsigned char	1 byte	Variable donde se almacena el número de protocolo de capa transporte.

		(TCP=0x06, UDP=0x11)	

Cada vez que el paquete atraviesa la pila de protocolos LWIP capa por capa, el código agregado en ciertas ubicaciones de la pila se encarga de almacenar la información de cabecera en la respectiva variable miembro de la estructura “netif” indicada en la tabla anterior.

Cabe resaltar que dentro del proceso de análisis del campo “tipo” de la cabecera Ethernet, se modificó el código de tal proceso para que los paquetes de tipo ARP (Address Resolution Protocol) y PPPoE (Point-to-Point Protocol over Ethernet) sean omitidos del análisis. Solamente se realiza el análisis para los paquetes cuyo valor del campo “tipo” sea 0x0800 los cuales corresponden a paquetes que contienen la cabecera del protocolo IPv4.

b. Extracción y almacenamiento de la cantidad de bytes

Para el presente trabajo de tesis, se considera el cálculo del tamaño del paquete desde la cabecera de capa de red hacia las capas superiores sin considerar la cabecera Ethernet (datagrama IP únicamente). Se asume esta consideración debido a que puede haber confusión en el cálculo, ya que puede haber presencia de tramas VLAN cuyo tamaño varía en 4 bytes en comparación con las tramas Ethernet estándar (Ver sección A.4 en Anexo A). Además, si se considera el campo “length” de la trama ethernet en vez de “type”, el cual no se observa en la red de pruebas, es posible que el paquete tenga datos en el campo “pad” los cuales, por definición, no están considerados en el campo “length” tal como se indica en la tabla de descripción de campos de una trama Ethernet en el mismo anexo.

La metodología empleada para obtener la cantidad de bytes del paquete según la consideración mencionada, se basa en la lectura del campo “Longitud total” de 2 bytes de la cabecera IP debido a que este campo proporciona la longitud total del datagrama IP medido en bytes incluyendo los bytes del encabezado IP y los datos (capas superiores) (Ver sección 2.2).

Para poder extraer la información de este campo se agrega código dentro de la estructura de LWIP. La pila LWIP contiene una función llamada “ip_input” que es llamada por el driver del dispositivo de interfaz para procesar cada paquete IP entrante. Esta función realiza una revisión básica de los campos de la cabecera IP del paquete tales como la versión, el tamaño del paquete, el cálculo y revisión del campo “checksum” entre otros campos [48].

Nota:

Se asume que la pila de protocolos LWIP no va a recibir paquetes fragmentados ni paquetes con el campo “opciones” ya que éstos serán descartados porque LWIP no tiene soporte. Además, el porcentaje de paquetes fragmentados observados en la red de pruebas es prácticamente nulo.

La función “ip_input” recibe los siguientes parámetros de entrada:

- **Buffer de paquete**: Es un espacio de memoria donde se encuentra el paquete entrante que va a ser analizado.
- **Estructura “netif”:** Es la estructura que contiene la información de configuración de la interfaz de red. Además contiene todas las variables que fueron agregadas (Tabla 4.10) para almacenar la información que se necesita extraer.

Dentro del proceso de revisión del campo “longitud total”, el código que se agrega se encarga de almacenar el contenido de tal campo en la variable “longitud” de la estructura “netif”. Esta tarea se realiza de tal manera debido a que se tiene acceso a las variables miembro de dicha estructura dentro de la función “ip_input”, porque ésta acepta como uno de sus parámetros de entrada a dicha estructura.

La Figura 4.9 ilustra la metodología empleada utilizando un diagrama de bloques representando a la función “ip_input” y sus 2 parámetros de entrada. El código agregado está representado por la flecha de color azul.

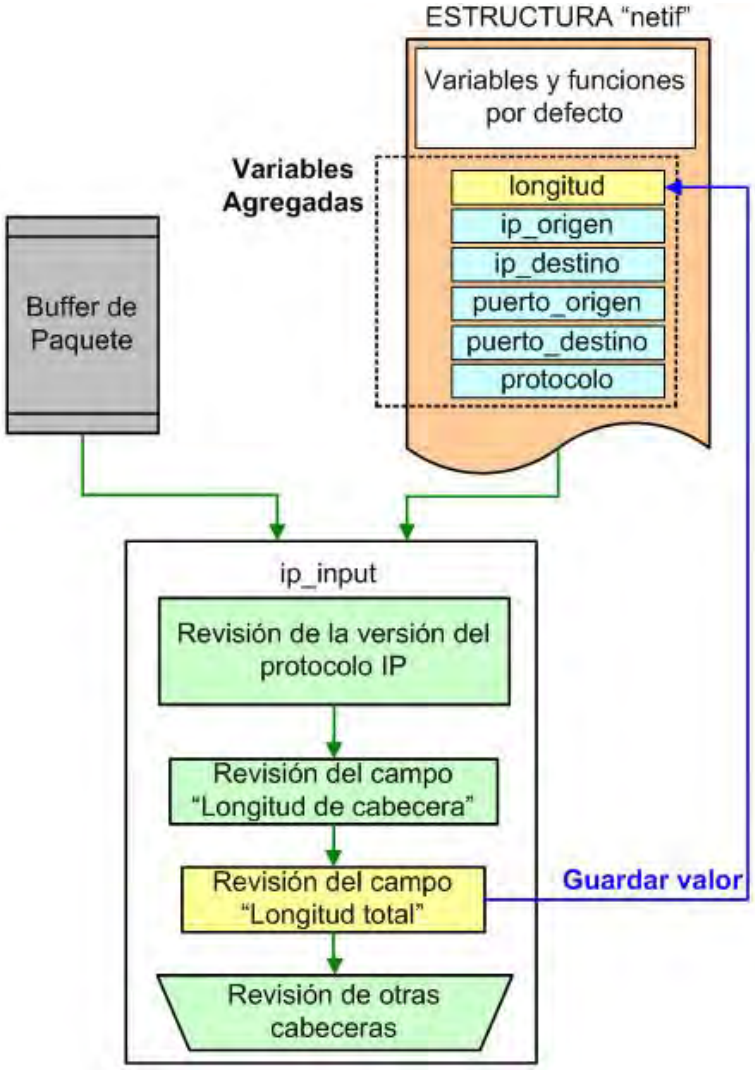


Figura 4.9 Proceso de extracción de la longitud total de un paquete IP

c. Extracción y almacenamiento de las direcciones IP origen y destino y del campo protocolo

La metodología empleada para obtener las direcciones IP origen y destino, se basa en la lectura directa de cada uno de los campos, de 4 bytes de longitud, que tienen el mismo nombre en la cabecera IP del paquete entrante.

Para poder extraer la información de estos campos se agrega código dentro de la estructura de LWIP, específicamente dentro de la función "ip_input" en la cual existen varios subprocesos, los cuales fueron identificados durante el análisis:

- Revisión de la versión del protocolo IP (4 bits): Se verifica que el campo de versión del protocolo IP tenga el valor 4 (versión del protocolo IP actualmente utilizada). En caso contrario se descarta el paquete.
- Revisión del campo "Longitud de cabecera" (4 bits).
- Revisión del campo "Longitud total" (2 bytes): Se revisa la longitud total del paquete, si excede el valor del tamaño del buffer de paquete, el paquete es descartado.
- Verificación del campo "Checksum" (4 bytes): Se realiza el cálculo del checksum sobre la cabecera IP y si no coincide con el campo, el paquete es descartado.
- Comparación de la dirección IP destino entrante (unicast o broadcast) con la que se tiene configurada en la estructura "netif" (dirección IP del sistema).
- Verificación del campo de dirección IP origen del paquete entrante: Se verifica si ésta dirección es de tipo multicast o broadcast y si es así la descarta, ya que según Douglas E. Comer [49] las direcciones multicast o de multidifusión IP solamente pueden aparecer como direcciones de destino al igual que las direcciones broadcast.
- Revisión del campo protocolo: El proceso de extracción de las direcciones IP origen y destino se realiza en este subproceso. Los posibles valores que acepta la pila LWIP para este campo son: UDP (0x11), TCP (0x06), ICMP (0x01). La Tabla 4.11 muestra multiplexación del flujo del programa.

Tabla 4.11 Multiplexación del flujo de programa

Campo de protocolo	¿Dirección IP destino = Dirección IP local?	Acción
ICMP (0x01)	Si	El paquete es procesado por la respectiva función de la pila LWIP relacionado con el protocolo ICMP ("icmp_input").[1]
	No	Se descarta el paquete
TCP (0x06)	Si	El paquete es procesado por la respectiva función por defecto que implementa la capa de transporte en TCP ("tcp_input").
	No	El paquete es procesado por una nueva función agregada en la pila LWIP llamada "tcp_input_0".[2]
UDP (0x11)	Si	El paquete es procesado por la respectiva función por defecto que implementa la capa de transporte en UDP ("udp_input")
	No	El paquete es procesado por una nueva función agregada en la pila LWIP llamada "udp_input_0"[2]

Nota:

[1] Esto se deja como tal ya que ayuda a la depuración de algunos errores de conexión que pueden presentarse durante las pruebas del módulo de captura. Sin embargo, la información de este paquete no es almacenado en las variables de la estructura “netif” ya que no corresponden a transacciones de tipo cliente-servidor.

[2] En cualquiera de estos 2 casos se agrega código para almacenar la dirección IP origen, la dirección IP destino y el valor del campo “protocolo” en las variables “ip_origen”, “ip_destino” y “protocolo” de la estructura “netif” respectivamente.

Los paquetes que no utilizan cabeceras TCP ni UDP en su estructura, no son considerados en el análisis debido a que no forman parte de una transacción de tipo cliente-servidor y por lo tanto no están relacionados con la producción o uso de servicios de red.

Como se indica en la Figura 4.10, solamente se extrae información de aquellos paquetes cuyo contenido del campo “protocolo” sea 0x06 para paquetes TCP y 0x11 para paquetes UDP sin importar si el paquete entrante tiene o no como destino el sistema local (módulo de captura).

La Figura 4.10 ilustra la metodología empleada utilizando un diagrama de bloques. Se representa a la función “ip_input” junto con sus 2 parámetros de entrada así como el detalle de cada subproceso de ésta función. A lo largo de la función se observan varias sentencias condicionales que de no cumplirse podrían descartar el paquete. El código de estas sentencias no es modificado ya que es parte de la pila LWIP y sirve para prevenir que paquetes malformados o con errores sean analizados.

La revisión del campo protocolo es realizada utilizando una sentencia de selección. Para el caso de ICMP el código es modificado para que se analice el contenido del paquete cuando su destino sea el sistema local. Esto permite determinar si existe conexión entre el módulo y los demás host de la red. Para ello se ejecuta el comando “ping” desde cualquier host de la red hacia el módulo. Si el destino de un paquete ICMP no es el sistema local, el paquete es descartado.

Si el paquete entrante es TCP se analiza si tiene como destino el módulo de captura, en caso afirmativo el paquete se llama a la función “tcp_input” la cual corresponde a la función de procesamiento de paquetes TCP por defecto de la pila LWIP y la cual procesa de modo normal el paquete. En caso contrario, se llama a la función “tcp_input_0” la cual fue creada y agregada para que únicamente se encargue de extraer la información de números de puerto y descartar el paquete una vez finalizado. De manera análoga se realiza el tratamiento de los paquetes UDP entrantes. En ambos casos, se extrae la información de las direcciones IP de origen y destino así como el número de protocolo respectivo. En la Figura 4.10, el código agregado está representado por las flechas de color azul (caso TCP) y naranja (caso UDP).

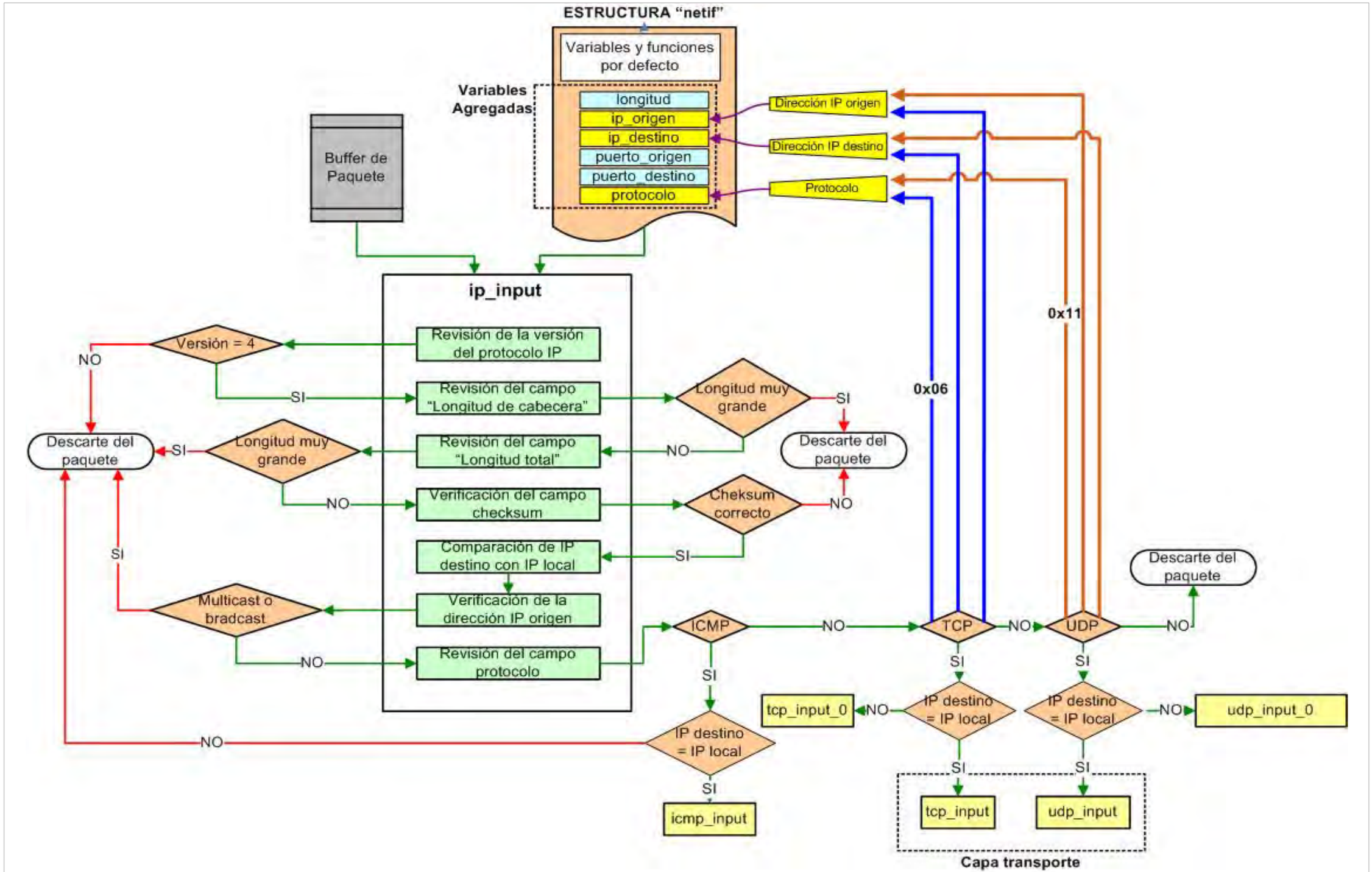


Figura 4.10 Proceso de extracción de la dirección IP de origen y destino y del campo protocolo de un paquete IP

d. Extracción y almacenamiento de los puertos origen y destino

La metodología empleada, para extraer la información de los puertos origen y destino del paquete entrante, se basa en la lectura directa del contenido de los campos “puerto origen” y “puerto destino” de la cabecera TCP o UDP, según sea el protocolo de capa transporte. Para poder extraer la información de estos campos se agrega código en la estructura de LWIP dentro de la función de procesado de cada capa de transporte respectiva.

Para el caso del protocolo TCP se agrega código dentro de la función por defecto “tcp_input” y en la nueva función creada denominada “tcp_input_0”. La Tabla 4.12 muestra una comparación de ambas funciones mostrando los principales subprocesos incorporados en cada una de ellas, a las que se les ha agregado la tarea de almacenar los números de puerto en las respectivas variables “puerto_origen” y “puerto_destino” de la estructura “netif”.

Tabla 4.12 Comparación de las funciones que procesan paquetes TCP

<table>
<tr><th></th><th>tcp_input
(función por defecto)</th><th>tcp_input_0
(función agregada)</th></tr>
<tr><td>Propósito</td><td>Realizar el procesamiento inicial de TCP</td><td>Realizar el procesamiento inicial de TCP</td></tr>
<tr><td>Caso de uso</td><td>IP destino = IP local</td><td>IP destino ≠ IP local</td></tr>
<tr><td rowspan="7">Subprocesos</td><td>Verificar el campo checksum</td><td rowspan="2">Verificar el tamaño de la cabecera TCP.</td></tr>
<tr><td>Verificar el tamaño de la cabecera TCP.</td></tr>
<tr><td>Leer la cabecera de puerto origen y almacenar su valor en la variable “puerto_origen” de la estructura “netif”.</td><td>Leer la cabecera de puerto origen y almacenar su valor en la variable “puerto_origen” de la estructura “netif”.</td></tr>
<tr><td>Leer la cabecera de puerto destino y almacenar su valor en la variable “puerto_destino” de la estructura “netif”.</td><td>Leer la cabecera de puerto destino y almacenar su valor en la variable “puerto_destino” de la estructura “netif”.</td></tr>
<tr><td>Lectura de otros campos de la cabecera.</td><td rowspan="2">Descartar el paquete</td></tr>
<tr><td>Demultiplexado del segmento TCP entre los PCBs (Process Control Block) activos y pasar el correspondiente PCB a la máquina de estado finito TCP. Ver referencia [48].</td></tr>
</table>

Para el caso del protocolo UDP se agrega código dentro de la función por defecto “udp_input” y en la nueva función agregada “udp_input_0”. La Tabla 4.13 muestra una comparación de ambas funciones mostrando los principales subprocesos incorporados en cada una de ellas, a las que, de igual manera que para TCP, se les ha agregado la tarea de almacenar los números de puerto en las respectivas variables “puerto_origen” y

"puerto_destino" de la estructura "netif".

Tabla 4.13 Comparación de las funciones que procesan paquetes UDP

	udp_input (función por defecto)	**udp_input_0 (función agregada)**
Propósito	Realizar el procesamiento del datagrama UDP	Realizar el procesamiento del datagrama UDP
Caso de uso	IP destino = IP local	IP destino ≠ IP local
Subprocesos	Verificar el tamaño de la cabecera IP + cabecera UDP.	Verificar el tamaño de la cabecera IP + cabecera UDP.
	Leer la cabecera de puerto origen y almacenar su valor en la variable "puerto_origen" de la estructura "netif".	Leer la cabecera de puerto origen y almacenar su valor en la variable "puerto_origen" de la estructura "netif".
	Leer la cabecera de puerto destino y almacenar su valor en la variable "puerto_destino" de la estructura "netif".	Leer la cabecera de puerto destino y almacenar su valor en la variable "puerto_destino" de la estructura "netif".
	Verificar el campo checksum.	Descartar el paquete
	Demultiplexado del datagrama UDP entre los PCBs (Process Control Block) activos y pasar el correspondiente PCB al proceso de recepción respectivo. Ver referencia [48].	

Se observa en ambas tablas de que en caso de que el paquete no tiene como destino el sistema local (módulo de captura), éste es procesado por la nueva función agregada y dentro de ella es descartado después de que la información de números de puerto es almacenada. Si el destino del paquete es el sistema local, éste es procesado por la función por defecto dentro de la cual sólo se agrega código para el almacenamiento de la información de números de puerto pero el resto del proceso continúa tal como fue diseñado.

4.2.3 Proceso de agregación de paquetes en flujos

El proceso de agregación de paquetes en flujos consiste en el almacenamiento de información correspondiente a cada flujo del tráfico utilizando la información extraída de cada paquete entrante al módulo de captura.

Este proceso está basado en la compresión de una cantidad significativa de paquetes pertenecientes a un flujo en particular lo cual facilita la realización de un posterior análisis del tráfico a nivel de flujos.

La Figura 4.11 muestra un diagrama de bloques del proceso de agregación de paquetes en flujos indicando los subprocesos. Asimismo, se muestran los demás procesos que también forman parte del diseño.

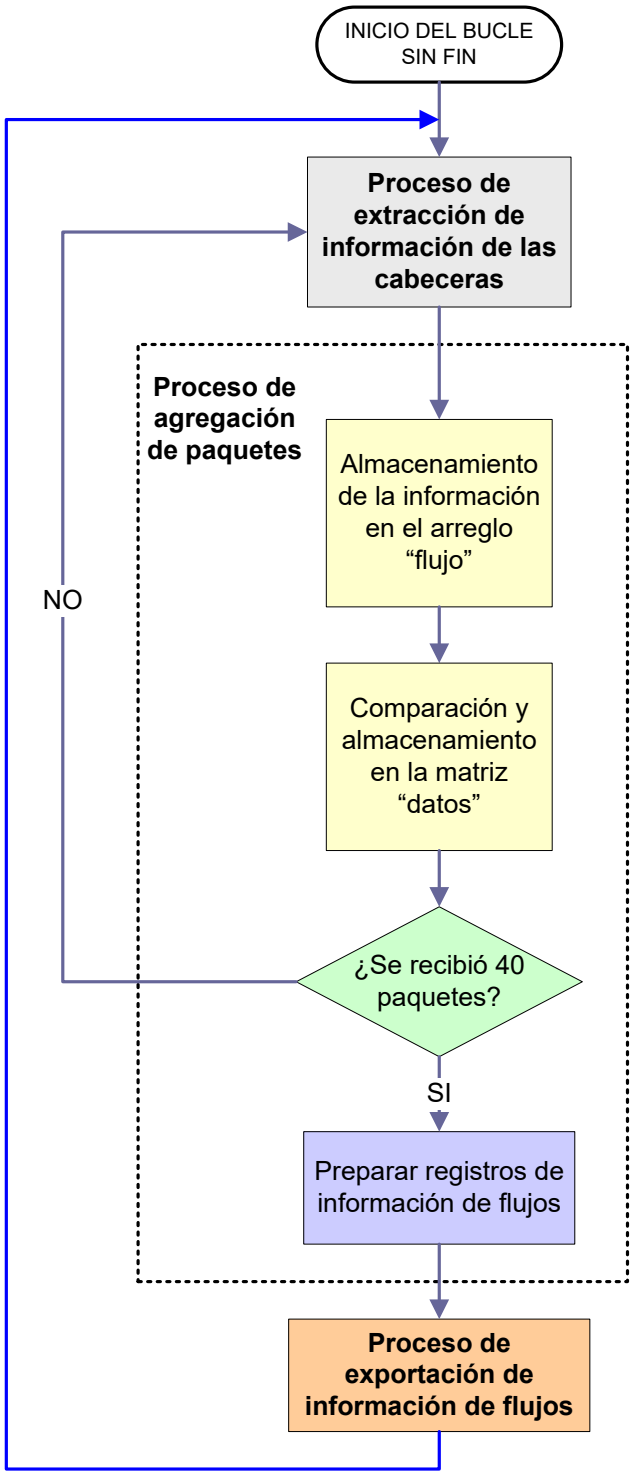


Figura 4.11 Diagrama de flujo genérico del proceso de agregación de paquetes en flujos

Al igual que los procesos anteriores, se desarrolló una metodología, la que consta de los pasos que se describen a continuación. Los cuatro primeros pasos son ilustrados mediante la Figura 4.12 "Proceso de agregación de paquetes en flujos".

1. En el programa principal se ejecuta la función "xemacif_input" dentro de un bucle sin salida. Esta función se encarga de pasar los paquetes recibidos desde la cola de recepción hacia la pila de protocolos LWIP [51].

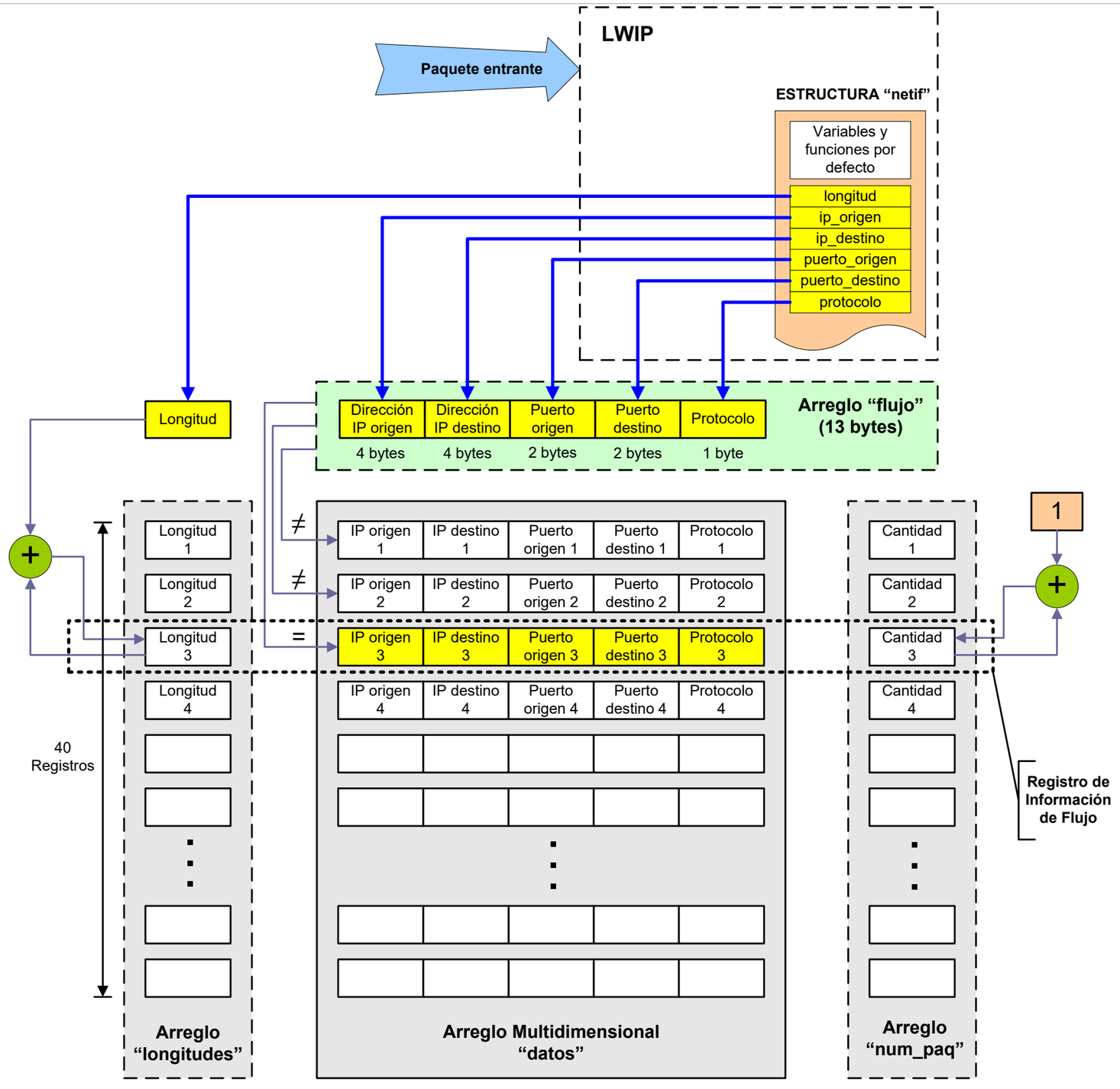


Figura 4.12 Proceso de agregación de paquetes en flujos.

2. La función “xemacif_input” devuelve un valor que es utilizado para detectar si el paquete entrante es de tipo TCP o UDP. En caso afirmativo, se realiza la lectura de los valores almacenados en las variables agregadas de la estructura “netif”. En caso de que el paquete entrante no sea de tipo TCP ni UDP, el programa principal no realiza tarea alguna. En este último caso el paquete entrante no es de interés para el análisis, por eso, las variables de la estructura “netif” no fueron actualizadas por la pila LWIP y por lo tanto, no se debe procesar la información actual almacenada en ellas.

3. Las variables leídas de la estructura “netif” (“iporigen”, “ipdestino”, “puerto_origen”, “puerto_destino” y “protocolo”) son concatenadas en una sola variable llamada “flujo”, la cual es una cadena de texto de 13 caracteres. El orden de la concatenación de estas variables es según se indica el formato propuesto en la sección 4.2.1.

4. El valor de la nueva variable “flujo” es utilizado según cualquiera de los 2 casos siguientes:

- **Caso 1**.- Si es la primera vez que se recibe un paquete, el valor de la variable “flujo” es almacenado en la primera ubicación (primera fila) de un arreglo multidimensional llamado “datos” de tamaño 40x13 bytes (40 registros de 13 bytes cada uno) la cual sirve de almacenamiento temporal para esta variable. Se define el arreglo “longitudes” (40 números) para el almacenamiento temporal de la suma de la longitud de los paquetes y el arreglo “num_paq” (40 números) para el almacenamiento temporal del número total de paquetes de cada flujo analizado. En la primera ubicación del arreglo “longitudes” se almacena el valor de la variable “longitud” de la estructura “netif” y en la primera ubicación del arreglo “num_paq” se almacena el valor 1.

- **Caso 2**.- Si ya se tiene registros de flujos en el arreglo “datos”, el valor de la variable “flujo” es comparado con cada registro no vacío del arreglo. Si no coincide con ninguno, éste es agregado en la ubicación siguiente a la del último registro no vacío del arreglo. Se almacena el valor de la variable “longitud” de la estructura “netif” en la misma ubicación en el arreglo “longitudes” y se almacena el valor de 1 en la misma ubicación en el arreglo “num_paq”. Si coincide con alguno de los registros, sólo se suma el valor de la variable “longitud” de la estructura “netif” al valor almacenado en la misma ubicación en el arreglo “longitudes” y se suma una unidad al valor almacenado en la misma ubicación en el arreglo “num_paq”.

5. Cuando el programa detecta que ya fueron procesados exactamente 40 paquetes de tipo TCP y/o UDP en total, los valores almacenados en los arreglos “datos”, “longitudes” y “num_paq” son concatenados en un arreglo unidimensional de 640 caracteres (40 registros de 16 bytes cada uno) llamado “matriz_de_datos” (Figura 4.13). Este arreglo sirve de almacenamiento temporal para los registros de información de flujos.

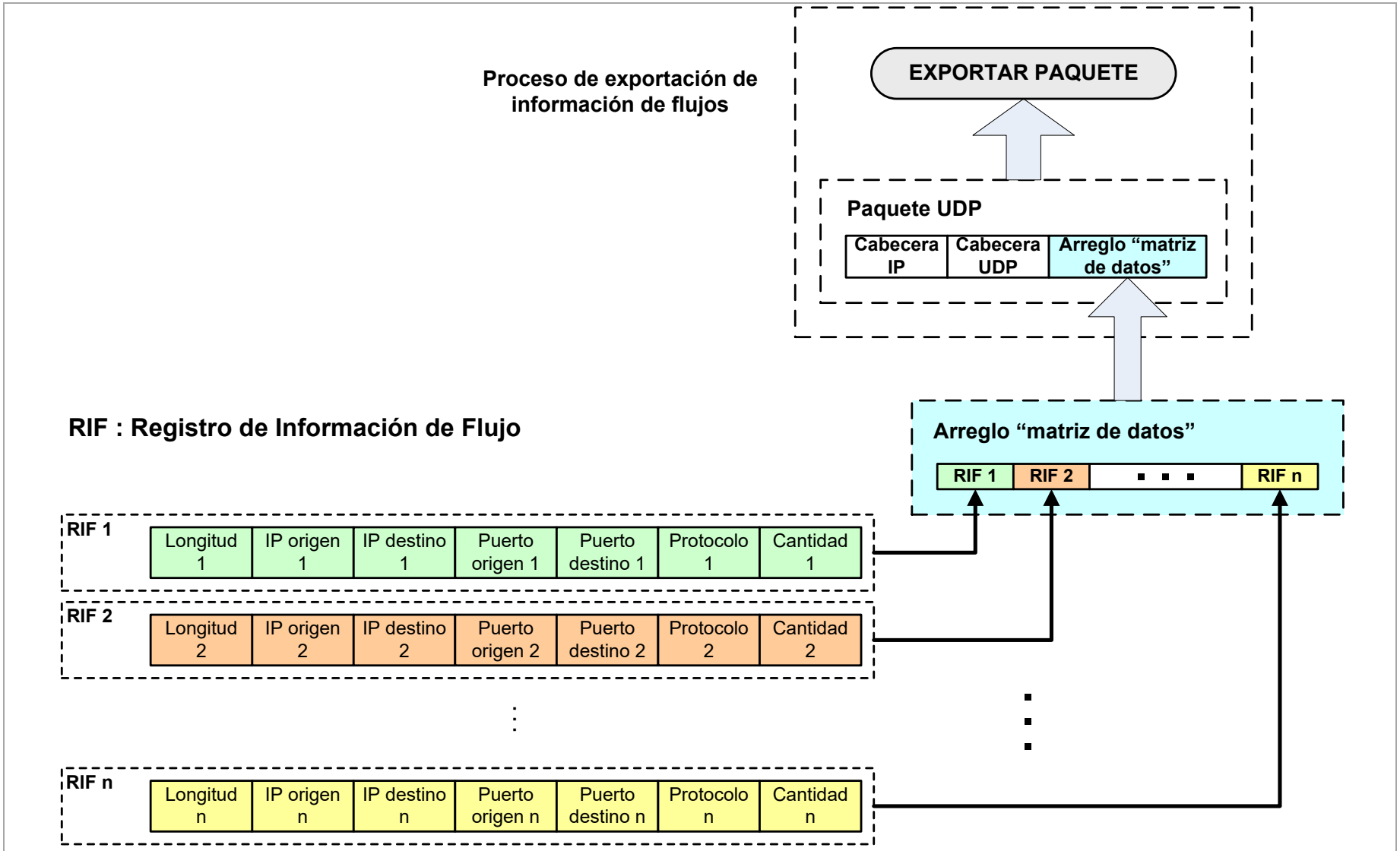


Figura 4.13 Configuración del arreglo "matriz_de_datos".

El orden de concatenamiento lo define el formato propuesto en la sección 4.2.1. Se almacenan los registros completos unos tras otros hasta completar la cantidad total de registros no vacíos de los 3 arreglos (el cual debe ser el mismo para los 3 arreglos). La cantidad máxima de registros de información de flujos posibles es de 40 registros.

6. La información de flujos almacenada en el arreglo “matriz_de_datos” es transferida a una función que se encarga de preparar un nuevo paquete que será exportado hacia el servidor remoto donde se implementa la “aplicación para modelamiento y reportes”.

La configuración del arreglo “matriz_de_datos” es realizada utilizando “n” registros de información de flujo (n = Total de flujos identificados en 40 paquetes). Este arreglo es colocado dentro de la carga útil de un paquete UDP el cual será exportado hacia un servidor remoto tal como indica la Figura 4.13.

La razón por la cual se el número máximo de registros de información de flujos es 40, se explica a continuación.

Para estructurar el paquete que será exportado, es necesario limitar su tamaño para que no supere el valor de MTU (Maximum Transfer Unit) de la red física, el cual es de 1500 bytes para redes Ethernet [49].

Condición 1:

Según el formato definido, el tamaño de un registro de información de flujo es de 16 bytes. El máximo tamaño del paquete a exportar se presenta cuando todos los registros son diferentes, es decir, cuando cada paquete entrante representa un flujo diferente cada uno. Por lo tanto, en este caso particular, el tamaño de la carga útil del paquete UDP que se exporta queda planteado mediante la expresión: Tcarga_max= Tregistro x Npaquetes, en donde Tcarga es el tamaño máximo de carga útil a exportar, Tregistro es el tamaño de cada registro (16 bytes), y Npaquetes es el número de paquetes entrantes. Quedando Tcarga_max= 16 x Npaquetes

A su vez, el máximo tamaño de la carga útil de un paquete UDP se presenta cuando el tamaño del datagrama IP es igual al MTU de la infraestructura física Ethernet, lo que se expresa de la siguiente manera: Tcarga_max= MTU – Tcabecera_red –Tcabecera_UDP, en donde Tcabecera es el tamaño de cabecera de la capa de red (20 bytes) y Tcabecera_UDP es el tamaño de la cabecera de la capa de transporte UDP (8 bytes), quedando Tcarga_max= 1500-20-8 =1472 bytes.

El tamaño máximo de la carga útil que se exporta, en el peor de los casos, no debe superar el tamaño máximo de la carga útil de un paquete UDP en una red Ethernet. En consecuencia, el número máximo de registros a exportar en un solo paquete UDP queda limitado según se indica en la siguiente expresión: 16 x Npaquetes $\leq$ 1472 bytes, o su equivalente: Npaquetes $\leq$ 92.

Según esta condición, se puede exportar hasta 92 registros de información de flujo diferentes, de 16 bytes cada uno, en un solo paquete UDP sin que sea fragmentado. Si se intenta enviar más de 92, el paquete tendría que ser fragmentado antes de ser enviado y reensamblado en el servidor destino.

Condición 2:

Otra consideración a tener en cuenta es el máximo valor que puede almacenar el campo “Número de Bytes” del formato propuesto para un registro de información de flujo. Como el tamaño de este campo es de 2 bytes (16 bits), es posible almacenar una suma total de longitudes de cada paquete entrante desde 0 hasta 2^{16}-1=65535 bytes.

El peor caso, que hace que se desborde este campo, se presenta cuando todos los paquetes entrantes pertenecen a un mismo flujo y, al mismo tiempo, el tamaño de cada paquete entrante (sin contar la cabecera Ethernet) es igual al tamaño máximo posible de 1500 bytes (MTU para redes Ethernet). Para evitar el desbordamiento se plantea la siguiente expresión la cual refleja esta situación: 1500 x Npaquetes ≤ 65535, lo que es equivalente a Npaquetes ≤ 43.69.

Conclusión:

Debido a que se analizaron los peores casos y para cumplir las condiciones planteadas, se elije el valor de 40 para el número de paquetes entrantes cuyos registros de información de flujo serán exportados en un solo paquete UDP hacia un servidor remoto.

4.2.4 Proceso de exportación de información de flujos hacia un servidor externo

Este proceso consiste en el envío de paquetes UDP hacia un servidor remoto conteniendo los registros de información de flujo sintetizados en el proceso anterior (arreglo “matriz_de_datos”).

La transacción UDP entre el módulo de captura y el servidor o PC remoto es de tipo cliente-servidor. Para el proceso se considera al módulo de captura como cliente el cual inicia de manera activa la conexión y la PC remota es el servidor el cual está en estado de escucha pasiva de conexiones UDP entrantes todo el tiempo. Si el módulo de captura conoce la dirección IP y el número de puerto del servidor remoto, éste puede enviar paquetes UDP directamente hacia el servidor ya que con estos datos se puede establecer una comunicación entre procesos a través de un socket [52] entre el cliente (módulo de captura) y el servidor (PC remoto).

La metodología utilizada para la implementación de este proceso está basada en el análisis de las funciones propias de la pila LWIP destinadas para generar y manipular transacciones UDP. Las características de tales funciones se encuentran en las referencias [53] y [54].

Este proceso está caracterizado por la implementación de 2 funciones las cuales son

invocadas para su ejecución desde el programa principal. Dentro de estas funciones se utilizan las mencionadas funciones de LWIP. Las características de cada una de ellas se muestran en la Tabla 4.14.

Tabla 4.14 Funciones relacionadas con el proceso de exportación

Nombre de la Función	Características
iniciar_udp	- Crea una nueva estructura PCB (Process Control Block) la cual define una sesión UDP [48]. Se utiliza la función “udp_new”. - Vincula la sesión UDP con la dirección IP local del sistema y un número de puerto local elegido al azar. Se utiliza la función “udp_bind”. - Configura la dirección IP y el puerto del servidor remoto a donde se va a enviar el paquete UDP dentro del PCB. Se utiliza la función “udp_connect”.
enviar_dato_udp	- Recibe 2 parámetros de entrada, el tamaño de la carga útil y el arreglo de caracteres a enviar. - Asigna memoria para un buffer de paquete de tipo PBUF_POOL (ver sección 2.1) de tamaño igual al primer parámetro. Se utiliza la función “pbuf_alloc”. - Carga este buffer con los datos contenidos en el arreglo de caracteres recibido como parámetro. - Envía el buffer de paquete hacia su destino (servidor remoto). Se utiliza la función “udp_send”. - Elimina el buffer de paquete de la memoria con la función “pbuf_free”.

A continuación se muestran dos diagramas de bloques. La Figura 4.14 ilustra la función “iniciar_udp”, en donde se muestra cada subproceso con los datos utilizados en cada uno de ellos. Por otro lado, la Figura 4.15 ilustra la función “enviar_dato_udp” en donde se ven los parámetros de entrada y los subprocesos donde son utilizados.

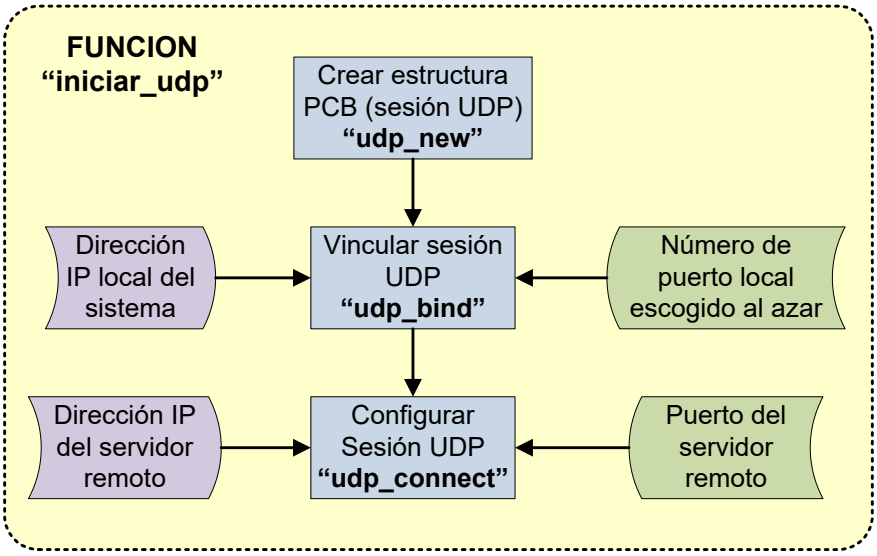


Figura 4.14 Inicialización del proceso de exportación

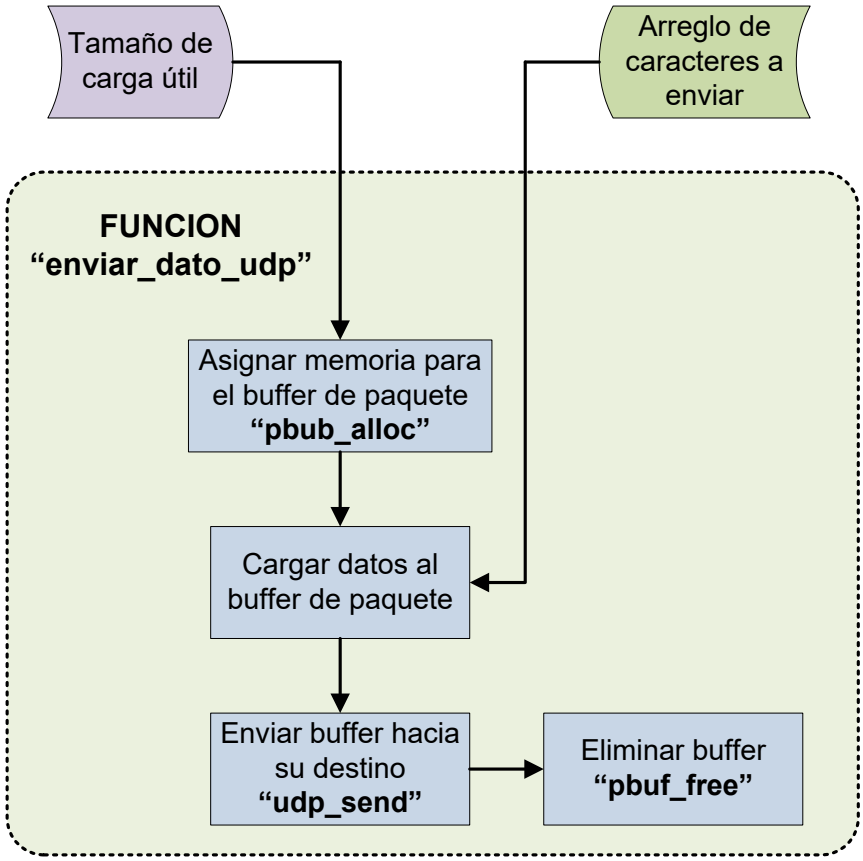


Figura 4.15 Proceso de exportación de información de flujos

4.2.5 Resumen de la aplicación de software

En esta sección se describe el programa principal el cual se encarga de realizar las configuraciones iniciales del sistema y controlar el flujo del hilo de ejecución de la aplicación de software.

Al final de la sección se muestran las configuraciones de la plataforma de software incluyendo el compilador del software, la salida estándar del sistema, la configuración de parámetros de la pila LWIP, las instancias de los controladores de cada IP core y la configuración de la memoria para generar un script enlazador.

Todos los gráficos mostrados provienen del diseño realizado sobre la herramienta SDK (Software Development Kit) de Xilinx.

a. Descripción del programa principal

En esta sección se describe un esquema del programa principal en el cual se explican las inicializaciones del programa y el flujo de los 3 procesos detallados anteriormente.

La Figura 4.16 ilustra un diagrama de bloques que muestra la secuencia de eventos y procesos del programa principal del módulo de captura. Después de la figura se explica cada paso.

En los recuadros azules se muestran los procedimientos de inicialización, previos al inicio del bucle que contiene los procesos principales.

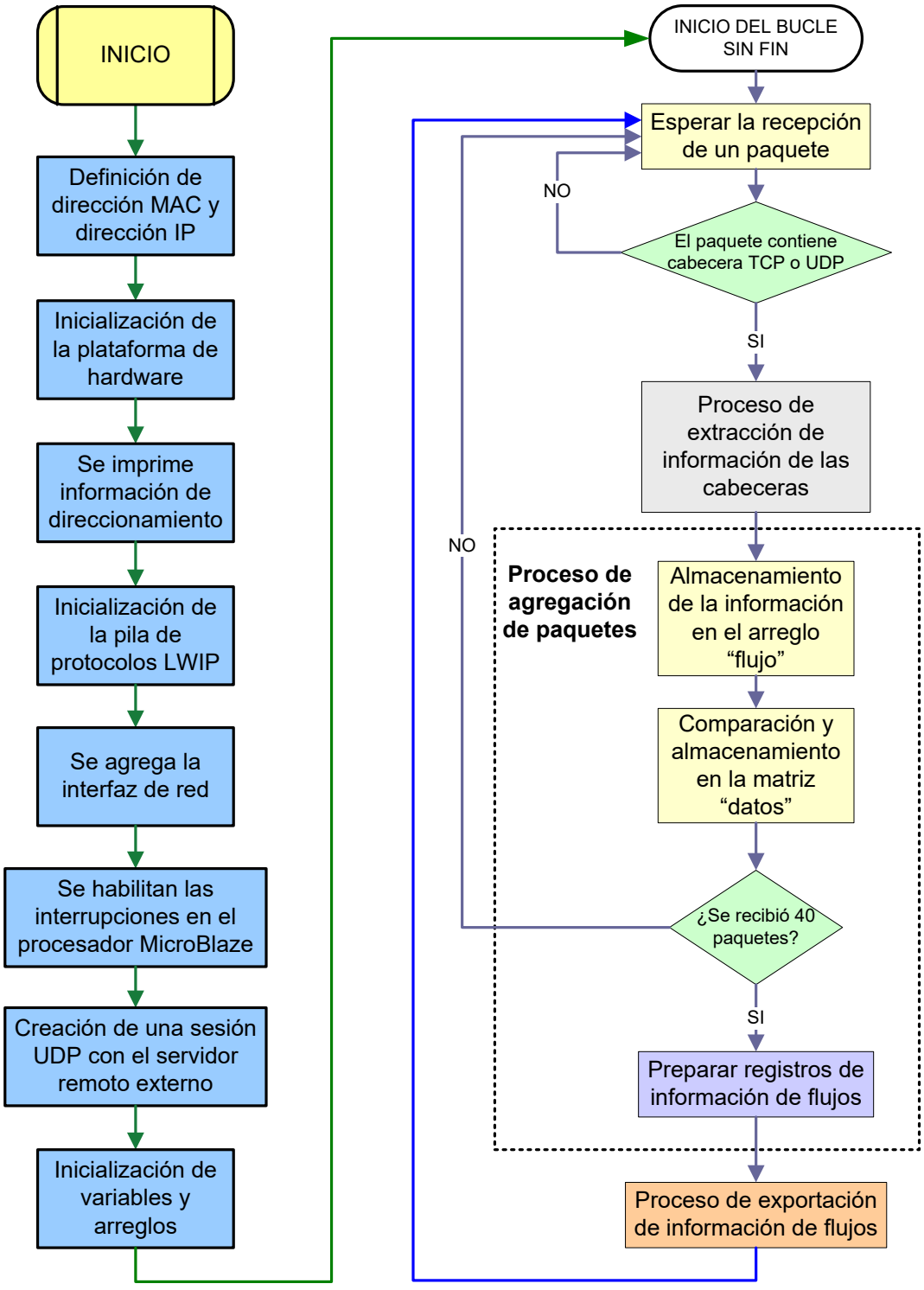


Figura 4.16 Diagrama de bloques del programa principal del software del módulo de captura

- El programa inicia con la definición de la dirección MAC y la dirección IP del módulo de captura. Se almacenan dichos valores en sus respectivas variables.

- Se llama a la función "init_platform" la cual inicializa la plataforma de hardware. Esta función incluye la habilitación de las memorias caché y la inicialización del controlador de

interrupciones.

- Se imprime en la salida estándar (puerto serial) una cabecera con un título. Luego se imprime la dirección IP, la máscara de subred y la puerta de enlace del módulo de captura. Igualmente se imprime la velocidad de la interfaz física de red. En este caso la interfaz está configurada para que la negociación de la velocidad sea automática.
- Se llama a la función “lwip_init” la cual inicializa la pila de protocolos LWIP según las configuraciones de sus parámetros.
- Se agrega una estructura de interfaz de red llamada “netif” a la lista de interfaces de red del sistema. Se establece como la interfaz de red por defecto a dicha estructura.
- Se habilitan las interrupciones en el procesador MicroBlaze.
- Se habilita la estructura de interfaz de red actual “netif” previamente establecida.
- Se llama a la función “iniciar_udp” la cual crea una sesión UDP con el servidor remoto configurando la dirección IP y el número de puerto del servidor colector de los paquetes que exporta el módulo de captura.
- Se inicializan los arreglos y variables que se utilizan en los 3 procesos principales y se ingresa a un bucle sin fin en el cual residen dichos procesos.
- Se recibe un paquete y se verifica si contiene cabeceras TCP o UDP. En caso afirmativo se continúa con el programa, en caso negativo el paquete es descartado y se espera la recepción del siguiente paquete.
- Se lee la información necesaria del paquete y se almacenan en variables locales.
- Se realiza el proceso de agregación de paquetes en flujos
- Si ya se han recibido 40 paquetes, se prepara un arreglo conteniendo registros de información de flujos de los 40 paquetes analizados. En caso contrario se espera la recepción del siguiente paquete.
- Se envía el arreglo a la función “enviar_dato_udp” la cual se encarga de enviar un paquete UDP hacia el servidor remoto con los registros de información de flujos como cara útil. Terminado esto, se espera la recepción del siguiente paquete.

b. Configuraciones de la plataforma de software

En esta sección se explica la configuración de la plataforma de software para lo cual se hacen uso de los siguientes elementos, que son detallados en las siguientes páginas:

- Figura 4.17, Configuración de la plataforma de software.
- Figura 4.18, Configuración del sistema operativo.
- Figura 4.19, Configuración de la pila LWIP.
- Figura 4.20, Configuración de los drivers.
- Figura 4.21, Configuración de parámetros para la generación de un script enlazador.
- Tabla 4.15, Secciones que son asignadas por el script enlazador.

En la Figura 4.17 se muestra la configuración de la plataforma de software. El compilador utilizado es “mb-gcc”, el cual es un compilador cruzado de programas escritos en lenguaje C y C++ para el procesador MicroBlaze derivado del compilador estándar “gcc” (GNU Compiler Collection) [36].

El archivador utilizado “mb-ar” crea, modifica y extrae ficheros objeto de archivos de bibliotecas de programa [36]. Se utiliza el sistema operativo “Standalone versión 2.00.a”, el cual funciona con un solo hilo de ejecución y proporciona acceso las capas de software más bajas [55]. Igualmente, se activa la biblioteca lwip130 tal como se muestra en la parte inferior.

La Figura 4.18 muestra la configuración del sistema operativo mencionado. Se designa a RS232_Uart_1 (instancia de la interfaz del puerto serial) como el periférico para la salida estándar del sistema. Las demás opciones se encuentran deshabilitadas.

La Figura 4.19 muestra la configuración de las bibliotecas de sistema. En este caso sólo se utiliza la biblioteca de la pila LWIP. Se detallan todos los parámetros configurables de LWIP junto con su descripción. Los parámetros corresponden a las opciones del adaptador de red, las opciones de memoria de LWIP, las opciones de los buffers de paquetes, las opciones del protocolo ARP, las opciones de los protocolos IP, ICMP, UDP, TCP y DHCP, la habilitación de estadísticas y la habilitación de la opción de depuración de la pila LWIP.

La Figura 4.20 muestra la configuración de los controladores de los IP cores los cuales son generados automáticamente por la plataforma XPS al momento de exportar el diseño de hardware a SDK. Se muestra cada controlador (driver) junto con su versión y la instancia de hardware respectiva.

La Figura 4.21 muestra las opciones de configuración para la generación automática del programa enlazador del software llamado “linker script” el cual tiene 2 funciones [36]:

- Dividir el programa en diferentes bloques de memoria.
- Describir el mapeo entre todas las secciones en todos los archivos de entrada de objetos a las secciones de salida dentro del archivo ejecutable. La salida es mapeada a las diferentes memorias que cuenta el sistema.

En la Figura 4.21 también se muestran todas las secciones de entrada que son asignadas por el “linker script” del procesador MicroBlaze. Se observa que las secciones de código, datos, heap y stack son asignadas a la memoria RAM DDR2 SDRAM externa, mientras que las secciones de arranque y los vectores de hardware son asignados automáticamente a la memoria local de instrucciones (bloques de RAM internos).

La Tabla 4.15 muestra los nombres y la descripción de cada una de las secciones mostradas en la Figura 4.21.

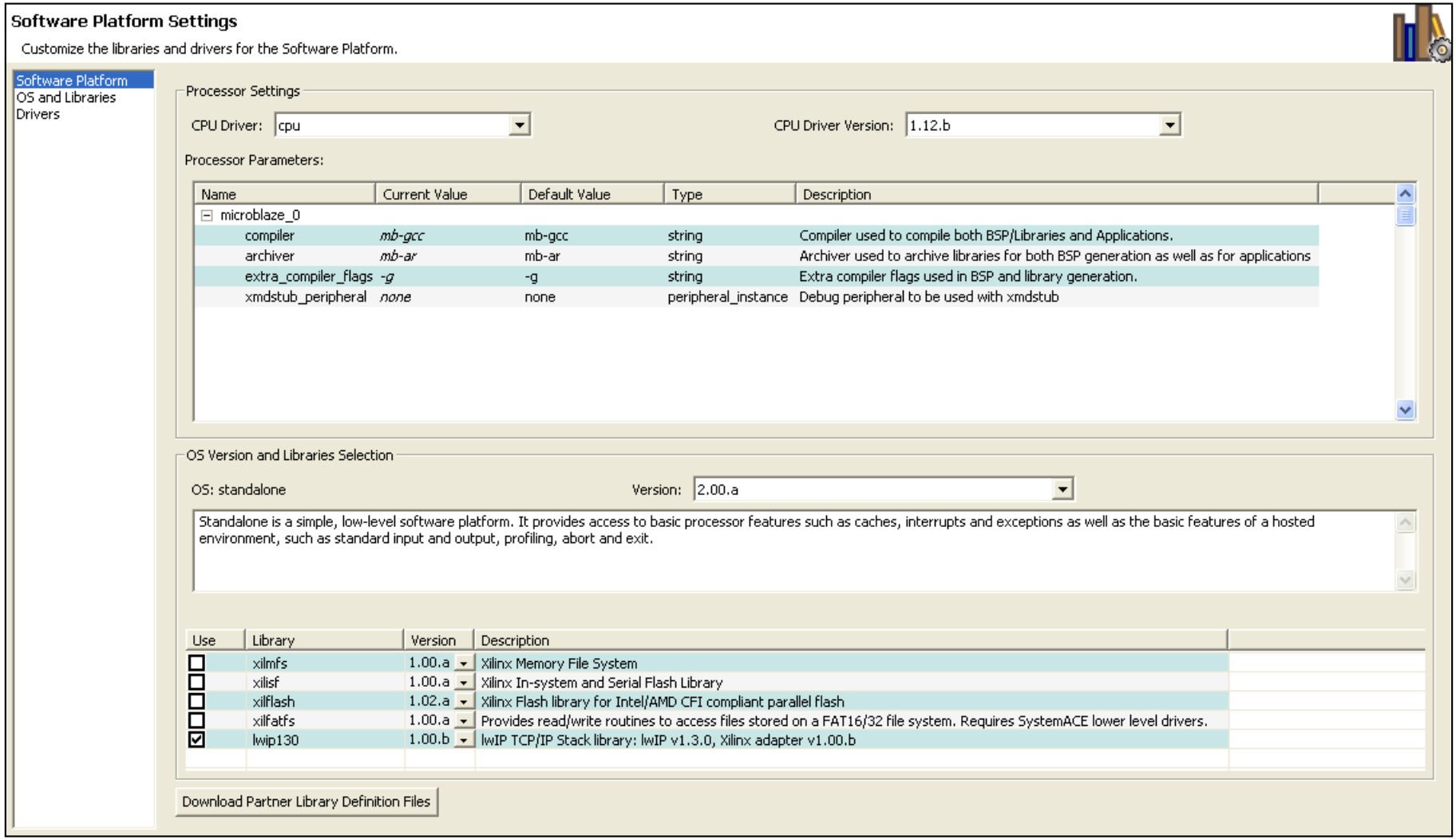


Figura 4.17 Configuración de la plataforma de software (Fuente: SDK)

Operating System Configuration: standalone version 2.00.a

Name	Current Value	Default Value	Type	Description
standalone				
stdin	*RS232_Uart_1*	none	peripheral_instance	stdin peripheral
stdout	*RS232_Uart_1*	none	peripheral_instance	stdout peripheral
enable_sw_intrusive_profiling	*false*	false	bool	Enable S/W Intrusive Profiling on Hardware Targets
profile_timer	*none*	none	peripheral_instance	Specify the Timer to use for Profiling. For PowerPC system, specify none to use PIT timer
microblaze_exceptions	*false*	false	bool	Enable MicroBlaze Exceptions
predecode_fpu_exceptions	*false*	false	bool	(MicroBlaze) Predecode FPU exceptions and save operand info before invoking user registered exception handler.

Figura 4.18 Configuración del sistema operativo (Fuente: SDK)

Libraries Configuration:

Name	Current Value	Default Value	Type	Description
lwip130				
api_mode	*RAW_API*	RAW_API	enum	Mode of operation for lwIP (RAW API/Sockets API)
socket_mode_thread_prio	*1*	1	int	Priority of threads in socket mode
temac_adapter_options	true	true	bool	Settings for xps_ll_temac lwIP adapter
n_tx_descriptors	*64*	64	int	Number of TX Buffer Descriptors to be used in SDMA mode
n_rx_descriptors	*64*	64	int	Number of RX Buffer Descriptors to be used in SDMA mode
n_tx_coalesce	*1*	1	int	Setting for TX Interrupt coalescing
n_rx_coalesce	*1*	1	int	Setting for RX Interrupt coalescing
tcp_rx_checksum_offlc	*false*	false	bool	Offload TCP Receive checksum calculation (hardware support required)
tcp_tx_checksum_offlc	*false*	false	bool	Offload TCP Transmit checksum calculation (hardware support required)
phy_link_speed	*CONFIG_LINKSPEED_AUTODETECT*	CONFIG_LINKSPEED_AUTODETECT	enum	link speed as negotiated by the PHY
temac_use_jumbo_fra	*false*	false	bool	use jumbo frames
lwip_memory_options				Options controlling lwIP memory usage
mem_size	*131072*	131072	int	Size of the heap memory (bytes).
memp_n_pbuf	*16*	16	int	Number of memp struct pbufs. Set this high if application sends lot of data out of ROM
memp_n_udp_pcb	*4*	4	int	Number of active UDP PCBs. One per active UDP connection
memp_n_tcp_pcb	*32*	32	int	Number of active TCP PCBs. One per active TCP connection
memp_n_tcp_pcb_liste	*8*	8	int	Number of listening TCP connections
memp_n_tcp_seg	*256*	256	int	Number of simultaneously queued TCP segments
memp_n_sys_timeout	*8*	8	int	Number of simultaneously active timeouts
memp_num_netbuf	*8*	8	int	Number of struct netbufs (socket mode only)
memp_num_netconn	*16*	16	int	Number of struct netconns (socket mode only)
memp_num_api_msg	*16*	16	int	Number of api msg structures (socket mode only)
memp_num_tcpip_msg	*64*	64	int	Number of tcpip msg structures (socket mode only)
pbuf_options	true	true	bool	Pbuf Options
pbuf_pool_size	*256*	256	int	Number of buffers in pbuf pool.
pbuf_pool_bufsize	*1700*	1700	int	Size of each pbuf in pbuf pool.
pbuf_link_hlen	16	16	int	Number of bytes that should be allocated for a link level header.
arp_options	true	true	bool	ARP Options
arp_table_size	*10*	10	int	Number of active hardware address IP address pairs cached.
arp_queueing	*1*	1	int	If enabled outgoing packets are queued during hardware address resolution.
lwip_ip_options	true	true	bool	IP Options
ip_forward	*0*	0	int	Enable forwarding IP packets across network interfaces.
ip_options	*0*	0	int	1 = IP options are allowed (but not parsed). 0 = packets with IP options are dropped
ip_reassembly	*1*	1	int	Reassemble incoming fragmented IP packets
ip_frag	*1*	1	int	Fragment outgoing IP packets if their size exceeds MTU
ip_reass_bufsize	*5760*	5760	int	Reassembly Buffer size
ip_frag_max_mtu	*1500*	1500	int	Assumed max MTU on any interface for IP frag buffer
ip_default_ttl	*255*	255	int	Global default TTL used by transport layers
icmp_options	true	true	bool	ICMP Options
icmp_ttl	*255*	255	int	ICMP TTL value
udp_options	true	true	bool	Is UDP required ?
lwip_udp	*true*	true	bool	Is UDP required ?
udp_ttl	*255*	255	int	UDP TTL value
tcp_options	true	true	bool	Is TCP required ?
lwip_tcp	*true*	true	bool	Is TCP required ?
tcp_wnd	*2048*	2048	int	TCP Window (bytes)
tcp_snd_buf	*8192*	8192	int	TCP sender buffer space (bytes)
tcp_mss	*1460*	1460	int	TCP Maximum segment size (bytes)
tcp_ttl	*255*	255	int	TCP TTL value
tcp_maxrtx	*12*	12	int	TCP Maximum retransmission value
tcp_synmaxrtx	*4*	4	int	TCP Maximum SYN retransmission value
tcp_queue_ooseq	*1*	1	int	Should TCP queue segments arriving out of order. Set to 0 if your device is low on memory
dhcp_options	true	true	bool	Is DHCP required?
lwip_dhcp	*false*	false	bool	Is DHCP required?
dhcp_does_arp_check	*false*	false	bool	ARP check on offered addresses?
stats_options	true	true	bool	Turn on lwIP statistics?
lwip_stats	*false*	false	bool	Turn on lwIP statistics?
debug_options	true	true	bool	Turn on lwIP Debug?
lwip_debug	*false*	false	bool	Turn on lwIP Debug?
ip_debug	*false*	false	bool	Debug IP layer
tcp_debug	*false*	false	bool	Debug TCP layer
netif_debug	*false*	false	bool	Debug network interface layer
sys_debug	*false*	false	bool	Debug sys arch layer
pbuf_debug	*false*	false	bool	Debug pbuf layer

Figura 4.19 Configuración de la pila LWIP (Fuente: SDK)

Configuration for Drivers:

Peripheral	HW Version	Instance	Driver	Version
mpmc	5.03.a	DDR2_SDRAM	*mpmc*	*3.01.a*
xps_ll_temac	2.02.a	Hard_Ethernet_MAC	*lltemac*	*2.00.a*
xps_uartlite	1.01.a	RS232_Uart_1	*uartlite*	*1.14.a*
lmb_bram_if_cntlr	2.10.b	dlmb_cntlr	*bram*	*1.00.a*
lmb_bram_if_cntlr	2.10.b	ilmb_cntlr	*bram*	*1.00.a*
mdm	1.00.f	mdm_0	*uartlite*	*1.14.a*
xps_intc	2.00.a	xps_intc_0	*intc*	*1.11.a*
xps_timer	1.01.b	xps_timer_0	*tmrctr*	*1.11.a*

Figura 4.20 Configuración de los drivers (Fuente: SDK)

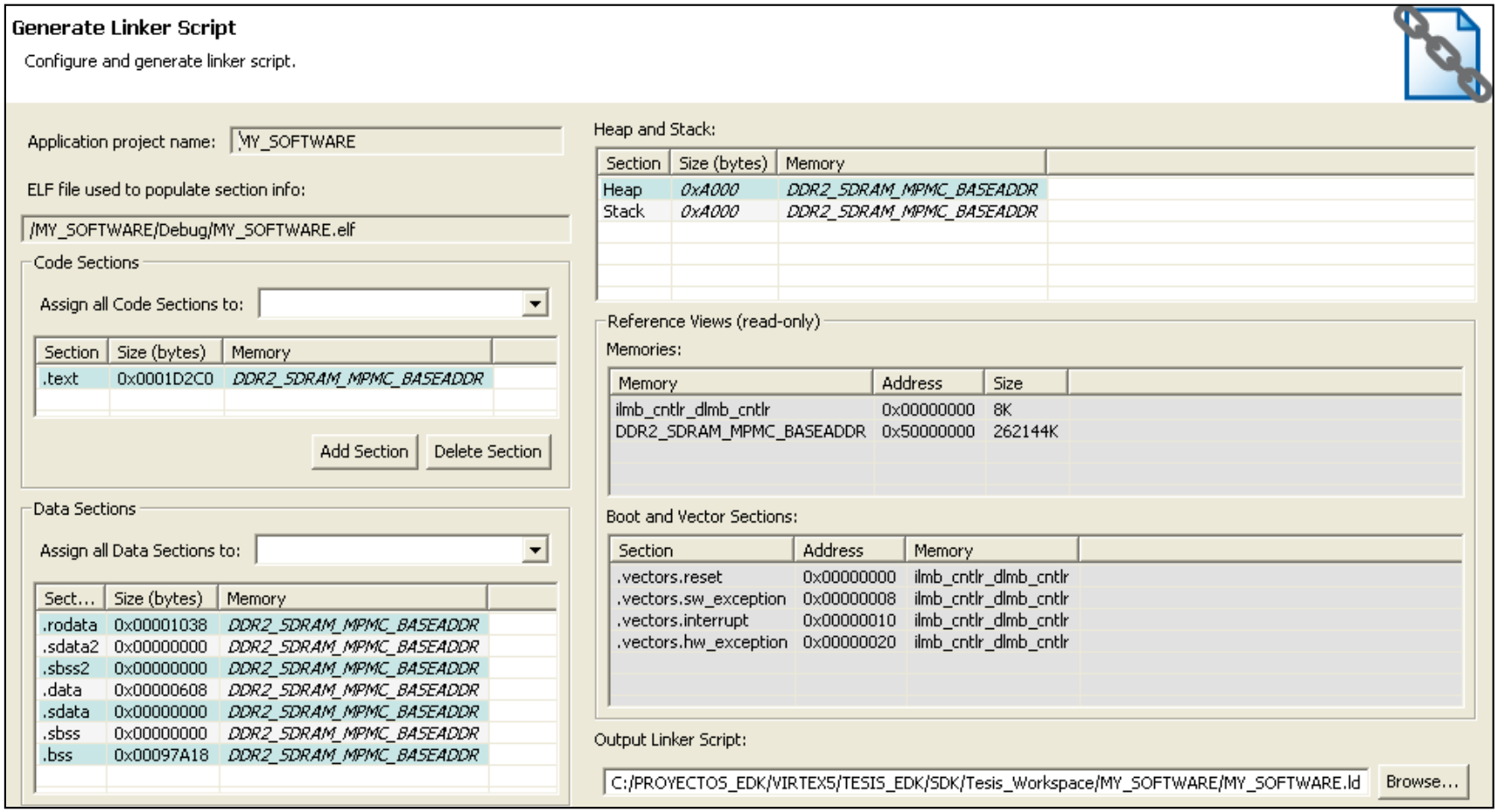


Figura 4.21 Configuración de parámetros para la generación de un script enlazador (Fuente: SDK)

Tabla 4.15 Secciones que son asignadas por el script enlazador (Fuente: [36])

Sección	Descripción
.vectors.reset	Vector o dirección de reinicio.
.vectors.sw_exception	Vector o dirección de excepción de software.
.vectors.interrupt	Vector o dirección de interrupción de hardware.
.vectors.hw_exception	Vector o dirección de excepción de hardware.
.text	Instrucciones de programa desde el código en funciones de alto nivel y sentencias globales de lenguaje ensamblador.
.rodata	Variables de sólo lectura.
.sdata2	Variables pequeñas estáticas y globales de sólo lectura con valores iniciales.
.data	Variables pequeñas estáticas y globales con valores iniciales. Inicializadas a cero por el código de arranque.
.sdata	Variables pequeñas estáticas y globales con valores iniciales.
.sbss2	Variables pequeñas estáticas y globales de sólo lectura sin valores iniciales. Inicializadas a cero por el código de arranque.
.sbss	Variables pequeñas estáticas y globales sin valores iniciales. Inicializadas a cero por el código de arranque.
.bss	Variables estáticas y globales sin valores iniciales. Inicializadas a cero por el código de arranque.
.heap	Sección de memoria definida para heap.
.stack	Sección de memoria definida para stack.

4.3 Pruebas de validación

Para comprobar el funcionamiento correcto, tanto del hardware como del software, se han realizado 2 tipos de pruebas:

- Pruebas para medir los tiempos de ejecución de los principales subprocesos del programa simulando la situación que genera mayor tiempo de procesamiento.
- Pruebas para comprobar el funcionamiento correcto de los 3 procesos del módulo de captura simulando la recepción de diferentes flujos de paquetes y comparando los resultados obtenidos con los esperados.

En las siguientes secciones se desarrolla cada una de estas pruebas explicando la metodología empleada para su realización y mostrando los resultados obtenidos.

4.3.1 Mediciones de tiempos de ejecución

El tiempo de ejecución (runtime) es el tiempo que demora un proceso en ser ejecutado por un sistema de cómputo. Para el módulo de captura esta medida es muy importante ya que permite determinar de manera indirecta la máxima cantidad de paquetes por segundo que el módulo es capaz de procesar en tiempo real.

La Figura 4.22 muestra un esquema de la metodología empleada para la medición de tiempos de ejecución de cada subproceso en el software del módulo de captura. Se muestra un color diferente para cada subproceso.

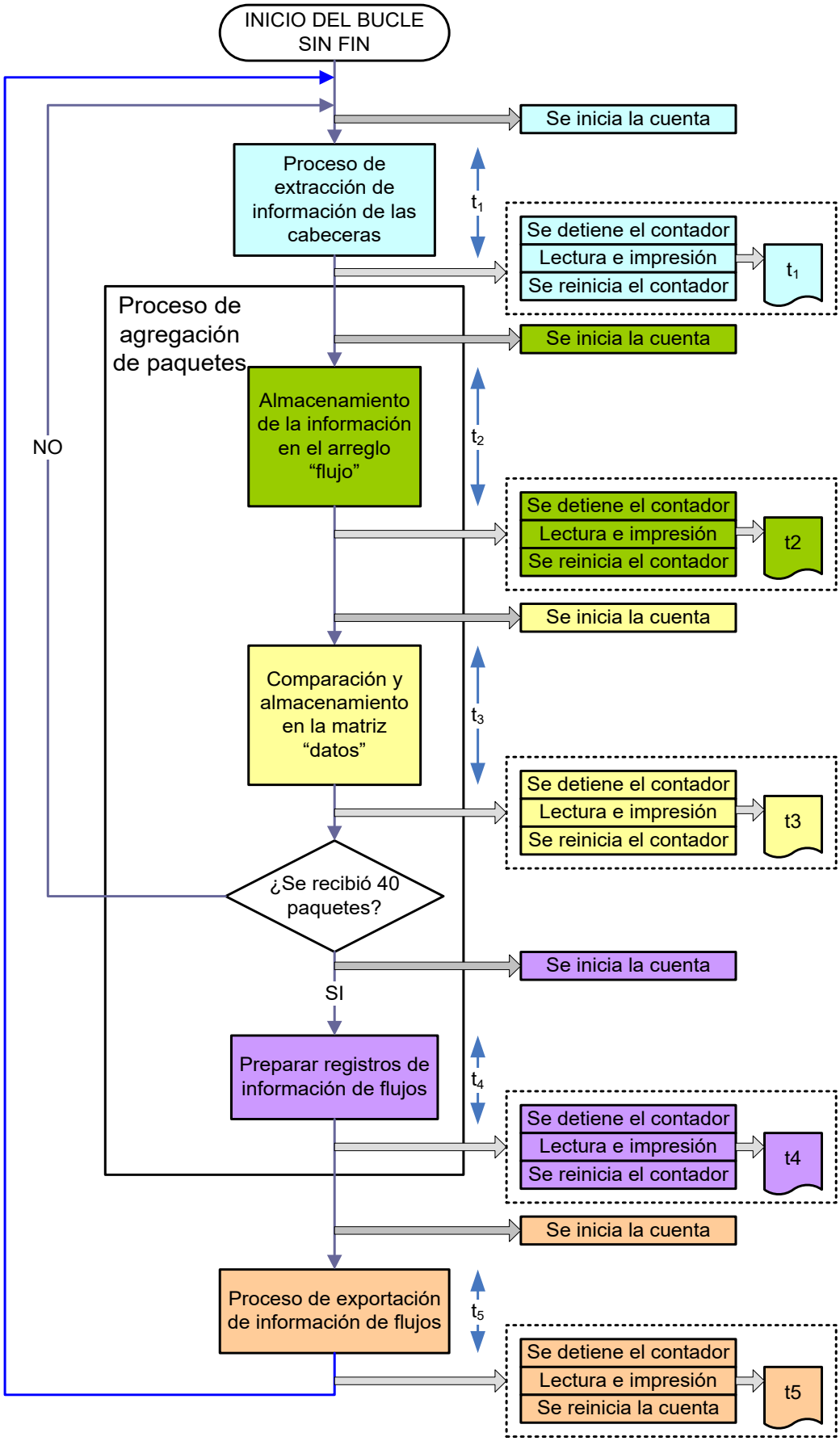


Figura 4.22 Metodología para la medición de tiempos de ejecución

Para medir el tiempo de ejecución del procesamiento de paquetes, según la metodología implementada en el diseño realizado, se utiliza el IP core “xps_timer_0” [56], el cual realiza la función de temporizador y contador utilizando un registro de 32 bits para almacenar las cuentas. Este hardware es inicializado, configurado y controlado desde el programa principal. Para su control se utilizan las funciones “XTmrCtr_Start” para iniciar la cuenta del contador, “XTmrCtr_GetValue” para obtener el valor del registro de 32 bits (cuenta del contador), “XTmrCtr_Stop” para detener la cuenta del contador y “XTmrCtr_Reset” para establecer la cuenta a un valor de reinicio predefinido.

La metodología empleada para la medición se basa en el control del temporizador hardware dentro del programa principal utilizando las funciones mencionadas. La medición de cada subproceso se realiza como se indica a continuación.

1. Se coloca la función “XTmrCtr_Start” justo antes del inicio del subproceso en el código del programa.
2. Se detiene el contador colocando la función “XTmrCtr_Stop” inmediatamente después de la culminación del subproceso.
3. Se realiza la lectura del temporizador con la función “XTmrCtr_GetValue” y se imprime su valor en el puerto serial (salida estándar del sistema).
4. Se utiliza la función “XTmrCtr_Reset” para reiniciar la cuenta a 0.
5. Durante la ejecución del programa se repite automáticamente los 4 pasos anteriores para cada paquete entrante.

Con esta configuración, el módulo registra la cuenta del temporizador para cada uno de los 40 paquetes entrantes en el respectivo subproceso. Al finalizar, se divide el valor de cada cuenta obtenida por la frecuencia del procesador en Hz (125 MHz = 125 000 000 Hz) para obtener los tiempos de ejecución en segundos.

Las mediciones de los tiempos de ejecución son realizadas de manera independiente en cada uno de los siguientes procesos:

- Proceso de extracción de información de cabeceras (tiempo de ejecución=t_1).
- Proceso de agregación de paquetes, que consta de los siguientes subprocesos:
  - Subproceso de almacenamiento de la información en el arreglo “flujo”. (tiempo de ejecución = t_2).
  - Subproceso de comparación y almacenamiento en la matriz “datos”.
(Tiempo de ejecución = t_3).
  - Subproceso de preparación de registros de información de flujos.
(Tiempo de ejecución = t_4).
- Proceso de exportación de información de flujos (tiempo de ejecución=t_5).

Se implementa un escenario experimental como se indica en la Figura 4.23, en el cual

se utiliza el generador de paquetes sendIP [57]. Se genera paquetes simulando la situación más crítica (40 paquetes entrantes pertenecientes a diferentes flujos), lo cual, según la naturaleza del software diseñado, ésta situación produce el máximo tiempo de procesamiento posible, debido a la ejecución de un gran número de comparaciones.

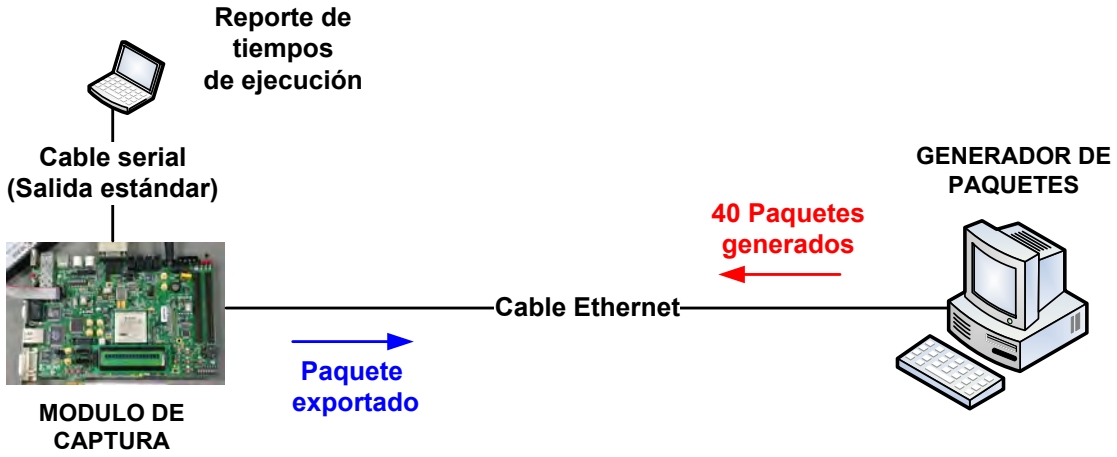


Figura 4.23 Escenario de pruebas para la medición de tiempos de ejecución

En la Tabla 4.16 se muestran los resultados obtenidos de las mediciones de tiempo de ejecución realizadas para cada subproceso. Las mediciones del tiempo de ejecución de cada subproceso fueron realizadas una sola vez y de manera independiente. Asimismo, se muestra una columna con los tiempos de ejecución totales para cada paquete y una fila con los tiempos de ejecución totales para cada subproceso.

De la tabla, se observa que el tiempo de ejecución total para 40 paquetes entrantes, en el peor de los casos, es 24.53 ms. De aquí, se deduce que el módulo es capaz de procesar 40 x 1000 / 24.53 = 1630 paquetes en 1 segundo.

Tabla 4.16 Tiempos de ejecución obtenidos en cada subproceso.

numero de paquete	Proceso de extracción	Proceso de Agregación			proceso de exportación	total por paquete (µs)
		Almacenamiento inicial	Comparación	Preparación de registros		
	t_1(µs)	t_2(µs)	t_3(µs)	t_4(µs)	t_5(µs)	
1	226.72	127.39	125.20	-	-	479.31
2	155.10	124.58	126.27	-	-	405.95
3	154.90	125.16	127.47	-	-	407.53
4	154.74	124.78	131.15	-	-	410.66
5	154.74	124.82	132.13	-	-	411.69
6	147.57	125.00	133.16	-	-	405.73
7	147.35	123.96	135.34	-	-	406.66
8	147.42	123.54	137.69	-	-	408.64
9	151.47	123.06	139.65	-	-	414.18
10	146.94	123.56	141.21	-	-	411.71
11	147.40	124.06	143.10	-	-	414.56
12	146.41	123.51	145.08	-	-	415.00
13	147.56	122.54	146.49	-	-	416.59

14	149.22	122.86	148.69	-	-	420.78
15	152.08	124.26	150.82	-	-	427.17
16	147.61	123.28	152.29	-	-	423.18
17	147.66	123.16	153.90	-	-	424.73
18	146.58	123.52	155.37	-	-	425.46
19	147.19	123.28	158.10	-	-	428.58
20	146.84	123.16	159.22	-	-	429.22
21	147.22	123.42	160.67	-	-	431.30
22	152.44	123.17	163.09	-	-	438.70
23	148.61	124.15	164.64	-	-	437.40
24	146.84	123.44	166.76	-	-	437.04
25	147.50	123.49	168.76	-	-	439.74
26	147.50	124.06	170.50	-	-	442.07
27	147.31	124.76	171.46	-	-	443.53
28	148.46	123.56	174.14	-	-	446.16
29	151.42	123.28	175.46	-	-	450.16
30	148.65	124.19	177.84	-	-	450.68
31	147.42	123.31	180.08	-	-	450.81
32	146.70	122.54	182.15	-	-	451.40
33	147.24	124.76	183.52	-	-	455.52
34	147.03	123.52	184.86	-	-	455.41
35	151.03	123.16	186.06	-	-	460.25
36	147.13	123.74	189.21	-	-	460.08
37	148.11	125.22	190.74	-	-	464.07
38	147.35	124.06	191.76	-	-	463.18
39	147.71	124.25	194.24	-	-	466.20
40	146.46	122.96	196.42	6 539.13	595.16	7 600.12
Total por sub-proceso	6 027.62	4 954.54	6 414.69	6 539.13	595.16	24 531.14
		17 908.36				

La Figura 4.24 muestra la tendencia de los tiempos de ejecución de algunos subprocesos. Se observa una tendencia constante para el tiempo de ejecución del proceso de extracción de información de cabecera, igualmente para el subproceso de almacenamiento. Sin embargo, para el subproceso de comparación, se observa un crecimiento lineal conforme llegan más paquetes. Esto se debe a que como cada paquete entrante pertenece a flujos diferentes (situación del peor caso), cada vez que un nuevo paquete ingresa se realizan comparaciones con toda la información previamente almacenada hasta el final de la última fila no vacía de la matriz “datos”.

La Figura 4.25 muestra los porcentajes del tiempo total de ejecución de cada subproceso. Se observa que el subproceso que consume la mayor cantidad de tiempo es el subproceso de preparación de registros de información de flujos (26.7% del tiempo total). La Figura 4.26 muestra los porcentajes del tiempo total de ejecución de cada proceso. Se observa que el proceso que consume la mayor cantidad de tiempo es el proceso de agregación (73% del tiempo total).

Este resultado era de esperarse ya que este proceso abarca 3 subprocesos los cuales,

según la gráfica anterior, consumen gran cantidad del tiempo total.

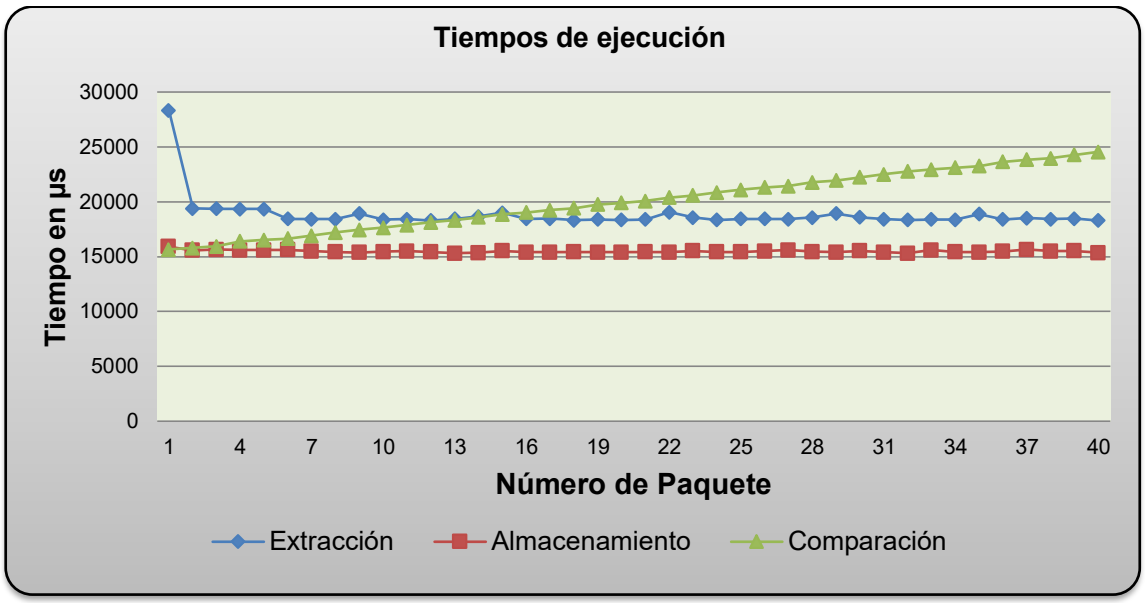


Figura 4.24 Evolución de los tiempos de ejecución

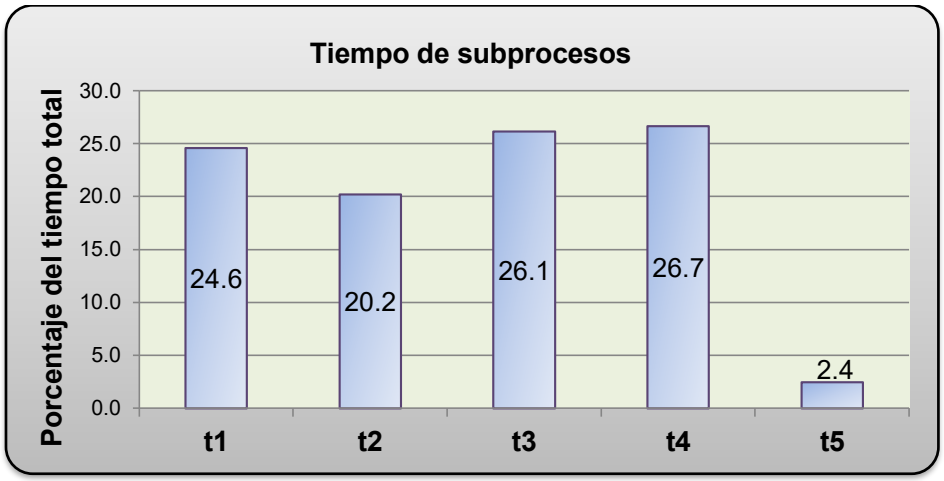


Figura 4.25 Porcentaje del tiempo de ejecución de cada subproceso

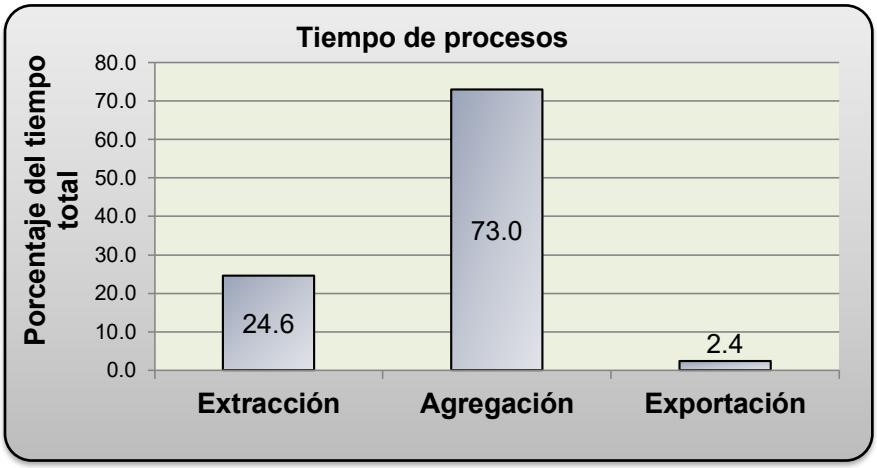


Figura 4.26 Porcentaje del tiempo de ejecución de cada subproceso

Como se había mencionado, de la prueba realizada se deduce que el módulo de

captura es capaz de procesar 1630 paquetes en 1 segundo, en el peor de los casos, entonces, la mayor cantidad posible de tráfico que se puede ser procesado se puede determinar asumiendo que cada paquete entrante tiene una longitud de 1500 bytes (MTU de Ethernet). En este caso particular, el máximo tráfico posible podría llegar a tener el valor de 1500 x 8 x 1630 = 19560000 bps ó 19.56 Mbps.

4.3.2 Pruebas de verificación del correcto funcionamiento

El módulo de captura realiza en total tres procesos cuya función en conjunto consiste en recibir paquetes IP, procesar según la metodología descrita y exportar un solo paquete UDP conteniendo la información de los flujos según el formato definido anteriormente. El proceso se repite cada vez que se recibe 40 paquetes en la interfaz física de red.

Para poder comprobar el correcto funcionamiento, según la metodología descrita, se realizó una prueba. El escenario de prueba consiste en un generador de paquetes y el módulo de captura ambos conectados por un cable de red tal como se muestra en la Figura 4.27. Se utiliza el generador de paquetes packETH [58], el cual genera dos series de paquetes bien definidas y los envía directamente hacia el módulo de captura. Éste recibe los paquetes, extrae la información, agrega los paquetes en flujos y envía los registros de información de flujos de vuelta al generador.

Se realiza una comparación entre el resultado de aplicar manualmente el proceso del módulo de captura a la serie de paquetes generados (serie conocida) y la información que exporta el módulo de captura, la cual es leída en el generador utilizando el analizador de paquetes Wireshark [59]. Se comprueba la correcta funcionalidad si ambos resultados son idénticos.

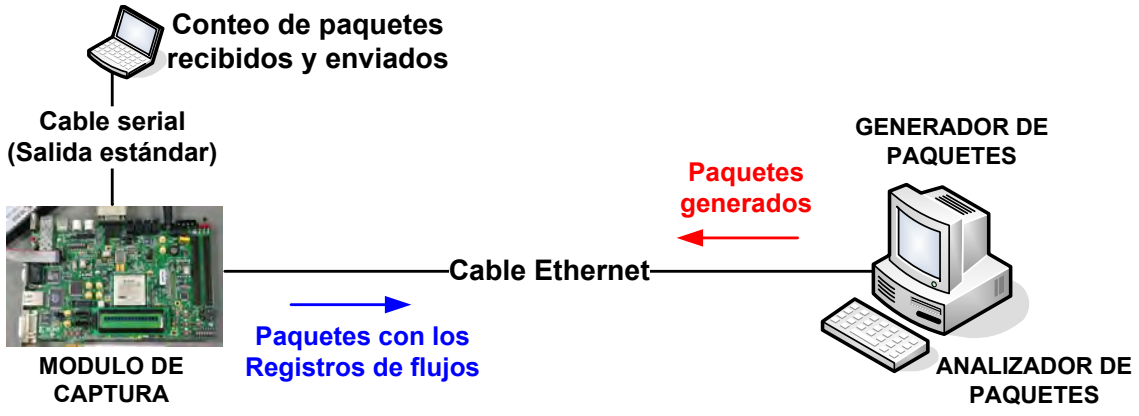


Figura 4.27 Escenario de pruebas para la verificación del funcionamiento del módulo de captura

La serie de paquetes consiste en un conjunto de 800 paquetes UDP de 1024 bytes de carga útil, cada uno enviado con un intervalo de 0.05 segundos entre cada paquete. El tamaño total de cada paquete es de 1052 bytes sin considerar la cabecera de la capa de

enlace de datos.

Se utiliza 10 tipos de paquetes diferentes, cada uno representa un flujo en particular ya que tienen diferentes combinaciones de valores en los campos dirección IP origen y destino, puerto origen y destino y protocolo. Se envía cada tipo de paquete uno tras otro hasta completar 80 ciclos para completar los 800 paquetes. Los 10 tipos de paquetes generados presentan las características descritas en la Tabla 4.17.

Tabla 4.17 Características de los paquetes generados para la verificación del correcto funcionamiento

Nombre	IP origen	IP destino	Puerto origen	Puerto destino	Protocolo	Tamaño (bytes)
paquete1	192.168.1.71	192.168.1.81	1234	5678	UDP	1052
paquete2	192.168.1.72	192.168.1.82	4321	8765	UDP	1052
paquete3	192.168.1.73	192.168.1.83	1234	5678	UDP	1052
paquete4	192.168.1.74	192.168.1.84	4321	8765	UDP	1052
paquete5	192.168.1.75	192.168.1.85	1234	5678	UDP	1052
paquete6	192.168.1.76	192.168.1.86	4321	8765	UDP	1052
paquete7	192.168.1.77	192.168.1.87	1234	5678	UDP	1052
paquete8	192.168.1.78	192.168.1.88	4321	8765	UDP	1052
paquete9	192.168.1.79	192.168.1.89	1234	5678	UDP	1052
paquete10	192.168.1.80	192.168.1.90	4321	8765	UDP	1052

La cantidad de información total enviada es de 841600 bytes. Como cada paquete es generado cada 0.05 segundos, el tiempo total aproximado (sin considerar el tiempo de generación de cada paquete) es de 0.05 x 800 = 40 segundos. La velocidad de paquetes aproximada es de 841600 x 8 / 40 = 168320 bits/s = 168.32 Kbps.

Se agregó código al programa para que imprima la cantidad total de paquetes entrantes al sistema, la cantidad total de paquetes enviados o exportados hacia un servidor remoto y la cantidad total de paquetes rechazados o descartados. La Figura 4.28 muestra la salida del puerto serial del módulo de captura. En ella se observa la cantidad de paquetes entrantes, enviados y rechazados.

Según la figura, el módulo recibió 806 paquetes, es decir, 6 paquetes adicionales a los que fueron generados. Estos paquetes corresponden a consultas ARP las cuales no forman parte del análisis. Como se generó 800 paquetes y por cada 40 paquetes entrantes el módulo envía un paquete con registros de información de flujos, se espera que el módulo envíe 800/40 = 20 paquetes de vuelta hacia el generador. Según lo que se puede apreciar en la figura referida, la cantidad de paquetes enviados fue de 20, igual al valor esperado.

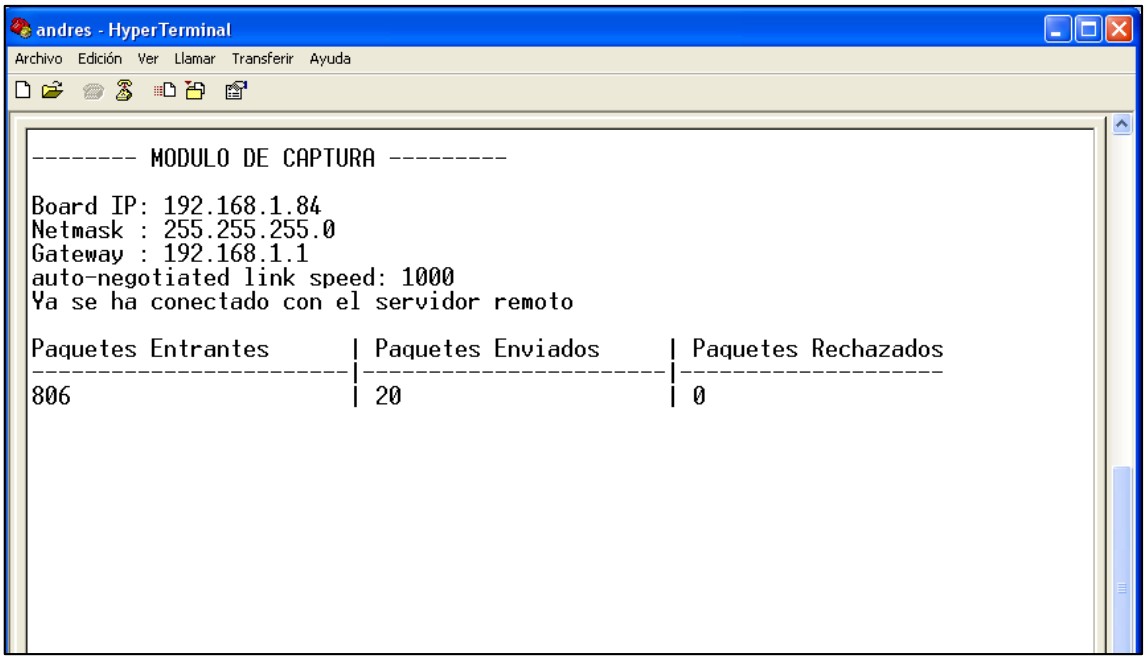


Figura 4.28 Salida del puerto serial del módulo de captura (Fuente: Hyperterminal)

La Tabla 4.18 muestra la traducción de la carga útil de uno de los 20 paquetes (Figura 4.29) mostrando cada elemento separado por el carácter "|". Se puede verificar que éste paquete contiene el resultado del proceso de 40 paquetes de los cuales se tienen 4 paquetes de cada uno de los 10 tipos y conteniendo exactamente la información de flujo de los paquetes generados según se indica en la Tabla 4.17.

Tabla 4.18 Traducción de la carga útil de paquete exportado por el módulo de captura

Paquete	->	Registros de información de flujo
Paquete 1	->	4208|192.168.1.71|192.168.1.81|1234|5678|UDP|4
Paquete 2	->	4208|192.168.1.72|192.168.1.82|4321|8765|UDP|4
Paquete 3	->	4208|192.168.1.73|192.168.1.83|1234|5678|UDP|4
Paquete 4	->	4208|192.168.1.74|192.168.1.84|4321|8765|UDP|4
Paquete 5	->	4208|192.168.1.75|192.168.1.85|1234|5678|UDP|4
Paquete 6	->	4208|192.168.1.76|192.168.1.86|4321|8765|UDP|4
Paquete 7	->	4208|192.168.1.77|192.168.1.87|1234|5678|UDP|4
Paquete 8	->	4208|192.168.1.78|192.168.1.88|4321|8765|UDP|4
Paquete 9	->	4208|192.168.1.79|192.168.1.89|1234|5678|UDP|4
Paquete 10	->	4208|192.168.1.80|192.168.1.90|4321|8765|UDP|4

La Figura 4.29 muestra la captura de los 20 paquetes enviados por el módulo de captura y recibidos por la PC donde está alojado el generador de paquetes. Se utilizó las direcciones IP 192.168.1.70 y 192.168.1.84 para el generador y para el módulo de captura respectivamente.

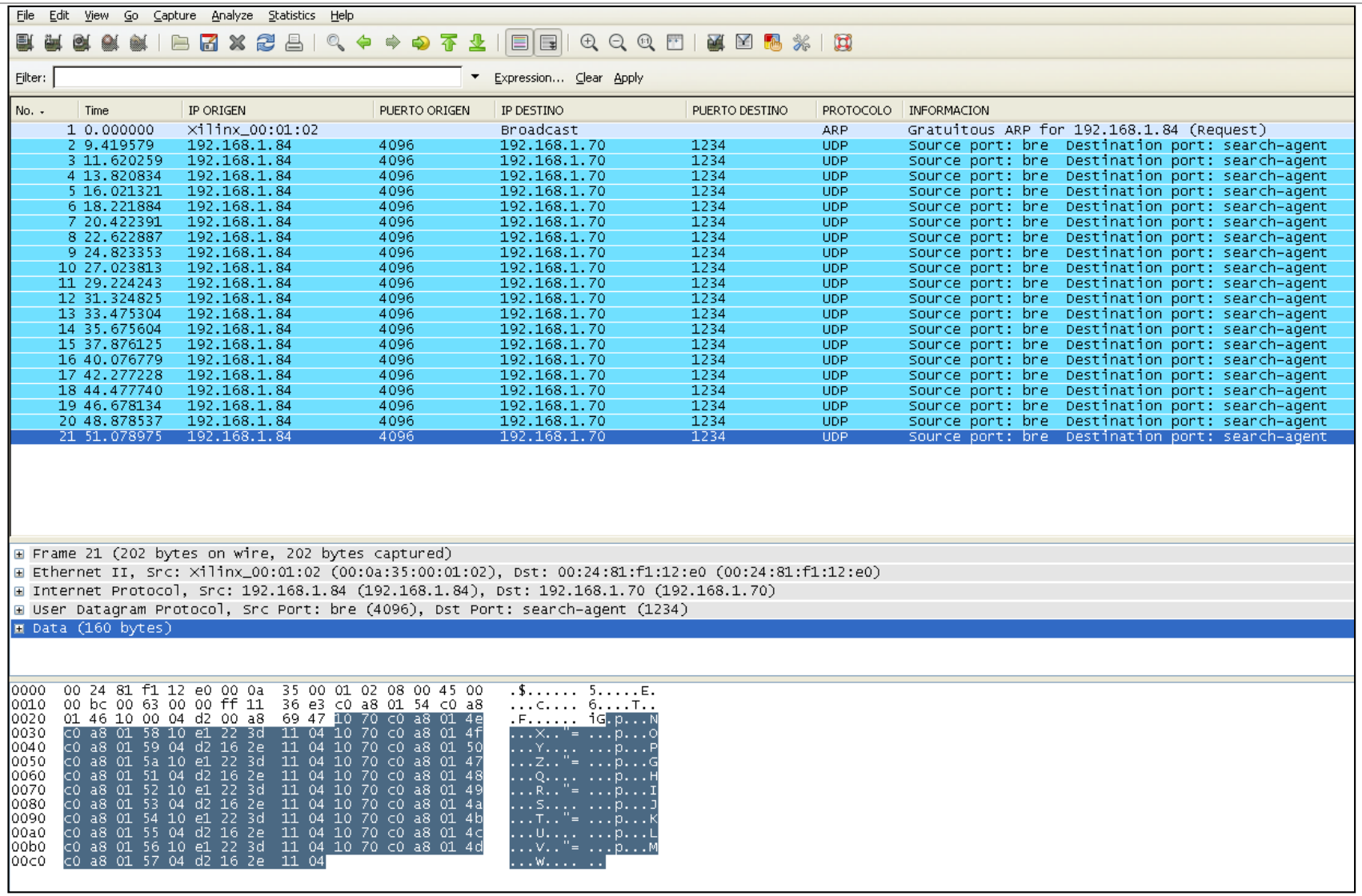


Figura 4.29 Paquetes enviados por el módulo de captura

CAPÍTULO V
DISEÑO DE LA APLICACIÓN PARA MODELAMIENTO Y REPORTES

Este capítulo se enfoca a exponer el diseño de la aplicación para modelamiento y reportes la cual consta de las siguientes partes: un proceso de recolección de información de flujos, un proceso de modelamiento estructural y una aplicación para la presentación de reportes. Dentro de este capítulo se desarrolla lo siguiente:

- Se explica el proceso de recolección de información de flujos.
- Se explica el proceso de modelamiento estructural basado en la teoría de la información.
- Se definen indicadores de producción de servicios, basado en los resultados de los procesos anteriores.
- Se presenta diseño de la aplicación de reportes, donde se presentan los indicadores al usuario cliente.
- Se explica la integración de todos los elementos y procesos involucrados que forman parte de la aplicación para modelamiento y reportes.

5.1 Proceso de recolección de información de flujos

El primer proceso que se ejecuta en el servidor es el proceso de recolección de información de flujos, el cual consiste en la recepción y almacenamiento de la información de los paquetes exportados por el módulo de captura.

El proceso inicia con la creación de un socket UDP dejando el puerto 1234 en escucha de cualquier conexión entrante a través de ese puerto (modo servidor). Al iniciar el módulo de captura, éste se conecta al servidor remoto (modo cliente) mediante el socket UDP creado por el servidor y al detectar el puerto abierto, éste envía la información de flujos hacia ese puerto del servidor.

Luego, el proceso espera a que el reloj del sistema marque el inicio de un intervalo de 5 minutos. Se considera este inicio cuando el reloj marque por primera vez un número de minutos múltiplo de cinco. Cuando ocurre esto, se inicia el proceso de recolección propiamente dicho.

El proceso de recolección se inicia con la recepción de paquetes. Cuando se detecta un paquete en el socket preestablecido, su contenido es analizado para extraer la información de los registros de flujos que fueron generados por el módulo de captura. Si la longitud de los datos del paquete no es múltiplo de 16 bytes (tamaño de un registro

de flujo), entonces se considera que el paquete está malformado y no se realiza su análisis.

Se ilustra cada uno de los pasos de este proceso mediante el diagrama de flujo mostrado en la Figura 5.1.

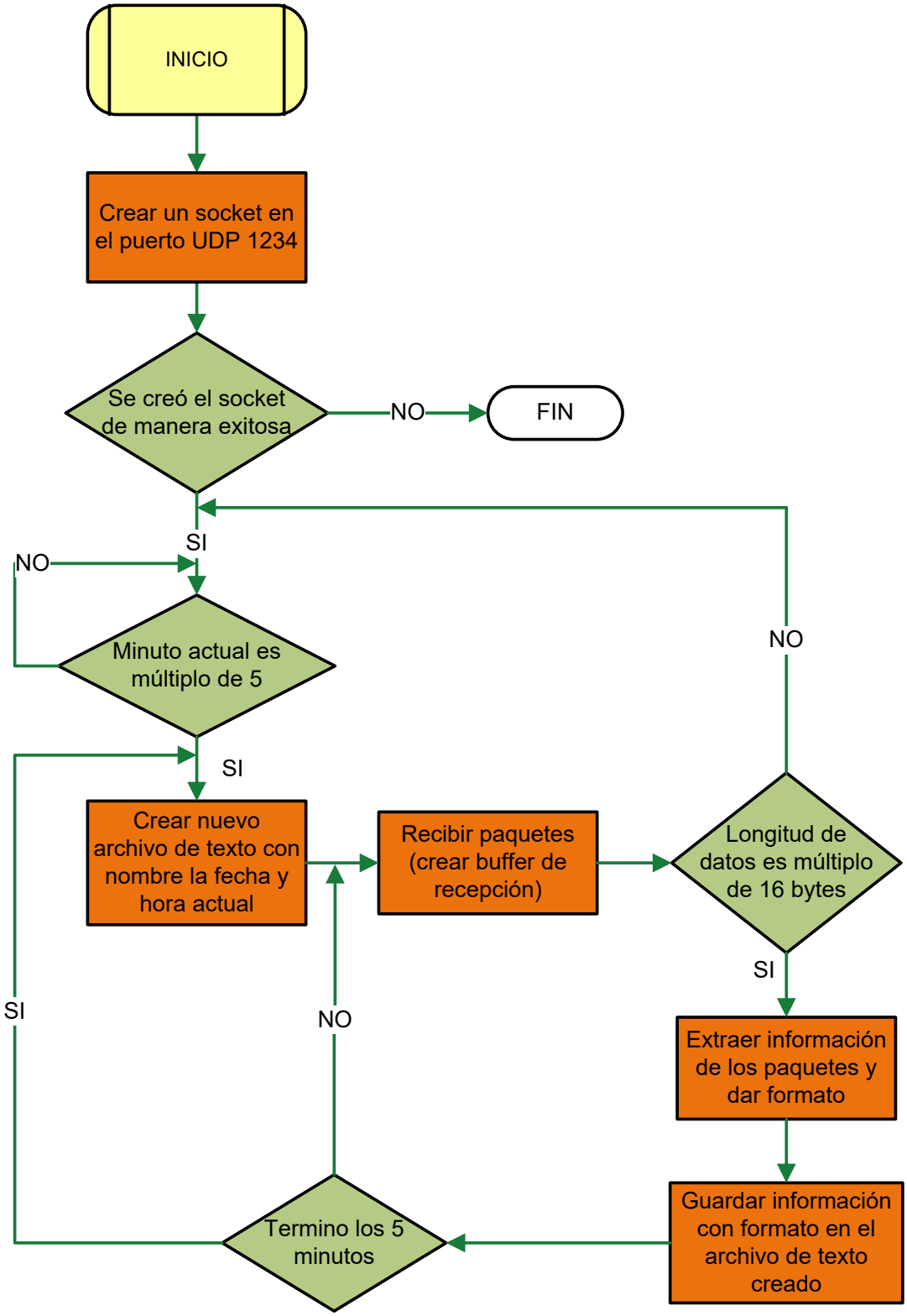


Figura 5.1 Diagrama de flujo del proceso de recolección de información de flujo

Si el tamaño de los datos del paquete cumple esta condición, cada campo de un

registro es almacenado en variables separadas que luego son concatenadas utilizando el carácter separador de pipe (“|”). Esta concatenación es almacenada en una variable tipo cadena de texto. Se repite este proceso para los registros de flujos subsiguientes concatenando el último resultado al resultado anterior.

Al finalizar el análisis de todo el paquete, se obtiene una variable de cadena de texto conteniendo cada registro de información de flujo con el formato establecido (paquete con formato). Por último el contenido de esta variable es almacenado en un archivo de texto dentro del servidor cuyo nombre indica la fecha y hora del inicio de la captura del primer paquete.

Este mismo proceso se realiza para los subsiguientes paquetes recolectados hasta que se alcance los 5 minutos de recolección. Llegado este tiempo, el servidor sigue realizando el mismo proceso a los subsiguientes paquetes pero almacenando los resultados en otro archivo de texto con un nombre diferente. El nombre del archivo de texto indica la fecha y hora de inicio del nuevo intervalo de 5 minutos.

La Figura 5.2 muestra un esquema del proceso de recolección de información de flujos tal como se ha descrito.

5.2 Proceso de modelamiento estructural basado en la teoría de la información

Este es el proceso principal de la aplicación para modelamiento y reportes ya que implementa la metodología detallada en la referencia [1]. Esta metodología utiliza conceptos de la teoría de la información para obtener un modelo compacto del tráfico observado. La implementación de los procesos que se describen en esta sección está basada en dicha referencia.

Según la referencia, se definen cuatro dimensiones (IP origen, IP destino, puerto origen y puerto destino). El modelamiento es realizado para cada una de las dimensiones, con lo cual se obtienen cuatro perspectivas diferentes del tráfico observado. Para cada una de las dimensiones se ejecutan los subprocesos indicados en la Figura 5.3.

La información almacenada en los archivos de texto del proceso anterior se utiliza como entrada para el proceso de modelamiento. En el sistema operativo del servidor se implementa una tarea programada que ejecuta el programa de los tres subprocesos cada cinco minutos.

En el inicio de dicho programa existen instrucciones que se encargan de revisar la presencia del archivo de texto con la información de flujos más reciente. Si existe tal archivo, se ejecutan los subprocesos, en caso contrario, se omite la ejecución y después de cinco minutos se vuelve a verificar la presencia del archivo de texto correspondiente a los siguientes cinco minutos. En las restantes subsecciones se detalla cada uno de los subprocesos mencionados.

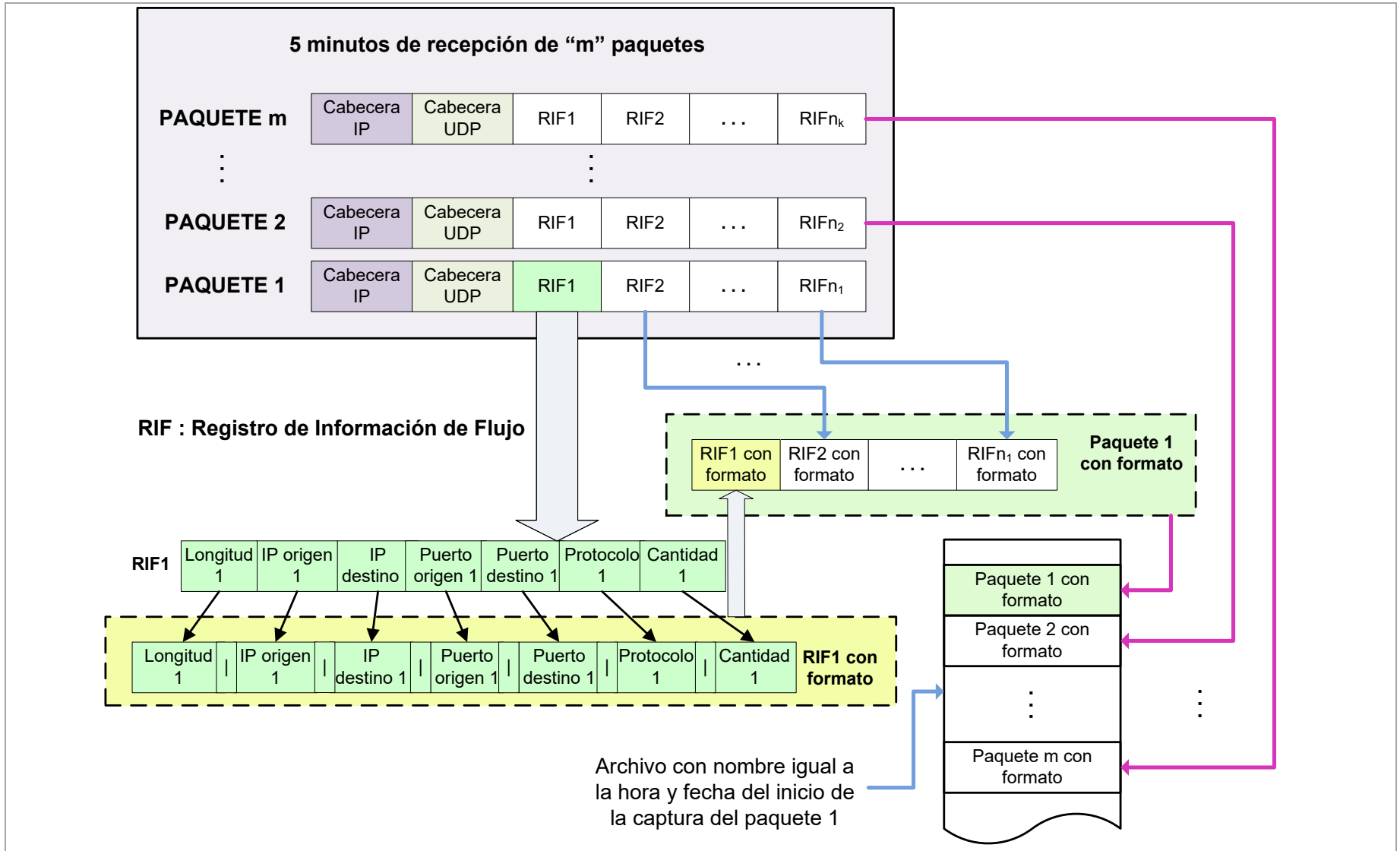


Figura 5.2 Esquema del proceso de recolección de información de flujos

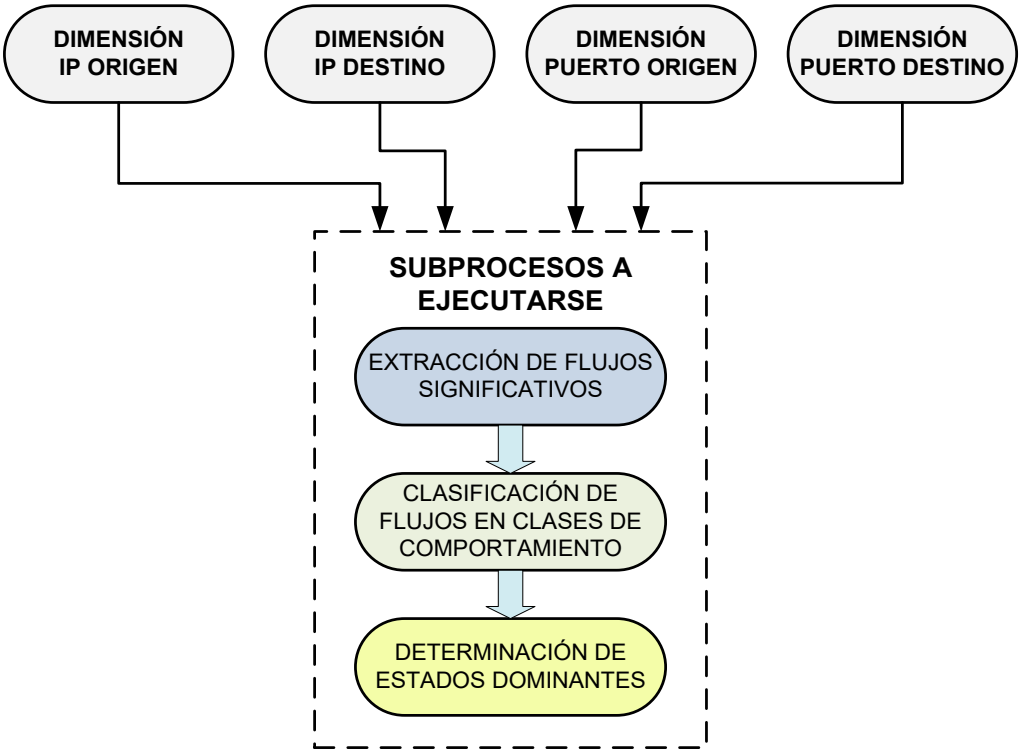


Figura 5.3 Dimensiones y subprocesos involucrados en el proceso de modelamiento

5.2.1 Extracción de flujos significativos

En este subproceso y en los siguientes se va a utilizar el término cluster el cual se refiere a un conjunto de flujos o registros de información de flujos que tienen el mismo valor en el campo de una determinada dimensión. En la Figura 5.4 se muestra un ejemplo particular que refleja el concepto de un cluster como un conjunto de flujos que tienen el valor de la dirección IP origen en común.

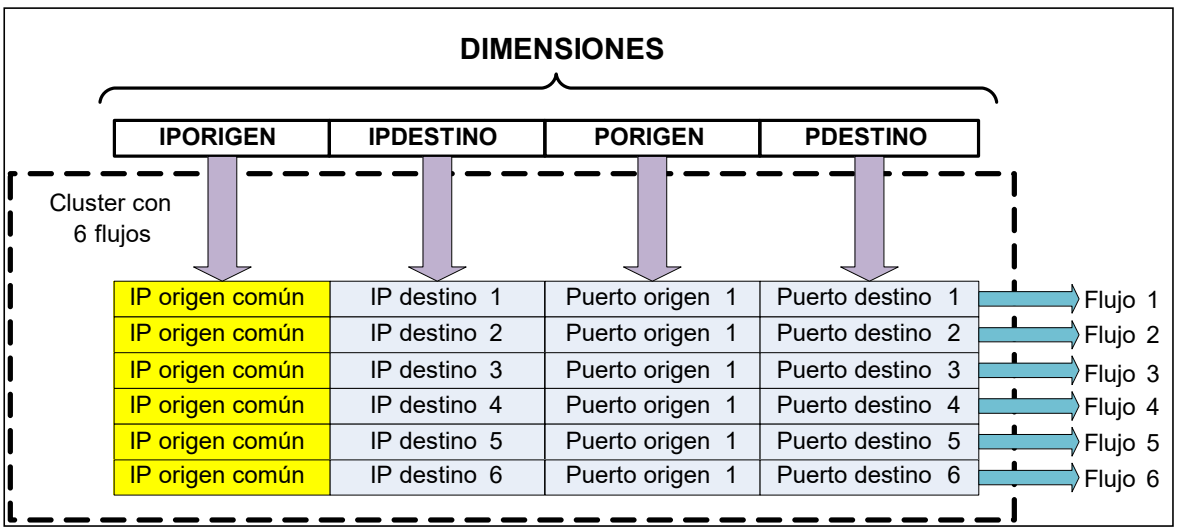


Figura 5.4 Ejemplo de un cluster de la dimensión IPORIGEN

Este subproceso realiza la eliminación de aquellos clusters que no son relevantes (significativos) en comparación con los demás. El criterio utilizado para la extracción de

flujos significativos está basado en el concepto de incertidumbre relativa (ver sección 2.4 “Teoría de la información”). A continuación se va a explicar la metodología empelada la cual es extraída de la referencia [1].

Dada una dimensión denotada por “X” y un intervalo de tiempo de 5 minutos denotado por “T”. Sea “m” el número total de flujos observados en dicho intervalo de tiempo. Se tiene el conjunto A= (a_1, a_2,…, a_n) que contiene los diferentes valores que toma la dimensión X (por ejemplo diferentes valores de direcciones IP origen) en cada uno de los flujos observados. Cada valor a_i del conjunto A define o identifica a un cluster particular de la dimensión X. La distribución de probabilidad P_A en X está dada por la ecuación 4.1:

$$P_i:=PA(a_i)=m_i/m \quad (5.1)$$

Donde m_i es el número total de flujos que toma el valor a_i de la dimensión X o, equivalentemente, m_i es el número total de flujos del cluster a_i.

La incertidumbre relativa de la dimensión X dado el conjunto observado A mide el grado de uniformidad en los valores observados en A, por lo tanto, se plantea las siguientes condiciones:

- Si $RU(P_A):=RU(X|A) > \beta= 0.9$, entonces los valores observados están más cerca de estar uniformemente distribuidos y por consiguiente son casi indistinguibles unos de otros.
- Si $RU(P_A):=RU(X|A) \leq \beta= 0.9$ entonces probablemente existen valores en A que resaltan del resto de valores en el conjunto.

El subconjunto S del conjunto A contiene los valores más significativos de A, si es que S es el subconjunto más pequeño de A que cumple con las siguientes condiciones:

- La probabilidad de cualquier valor en S es mayor que la de los valores restantes.
- La distribución de probabilidad condicional en el conjunto de valores restantes, denotado, por R:= A-S está cerca de ser uniformemente distribuida, es decir que $RU(P_R):=RU(X|R) > \beta= 0.9$.

Se deduce, de manera intuitiva, que el subconjunto S contiene los valores o clusters más significativos del conjunto observado A mientras que los valores restantes son casi indistinguibles unos de otros.

Para obtener el conjunto de valores del subconjunto S, según la metodología descrita, se realiza el procedimiento ilustrado en la Figura 5.5. El proceso es análogo para las demás dimensiones.

Este subproceso (extracción de flujos significativos) almacena la información de los clusters significativos en un arreglo multidimensional. Este arreglo multidimensional es utilizado por el siguiente subproceso (clasificación de flujos significativos en clases de comportamiento).

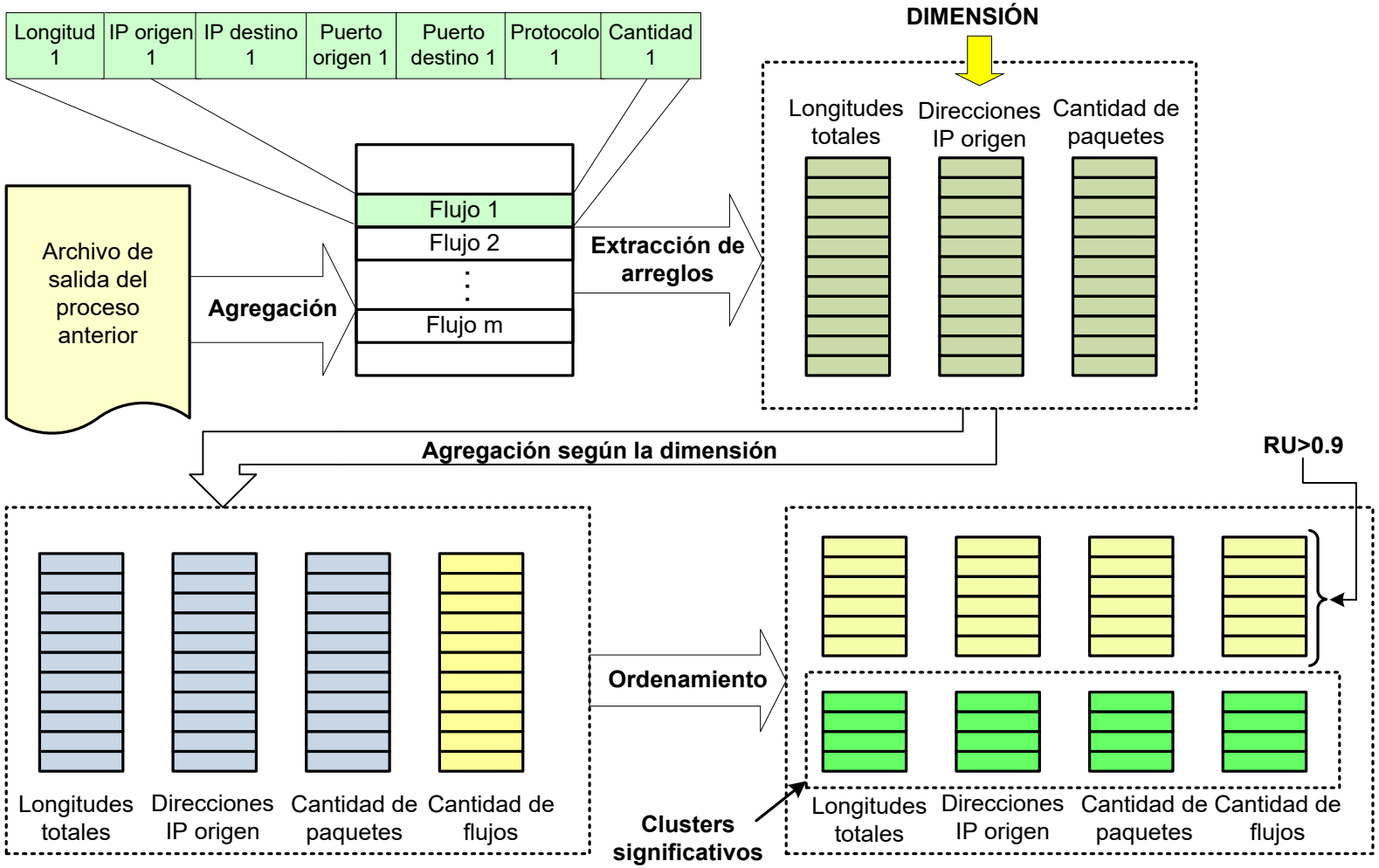


Figura 5.5 Procedimiento para obtener clusters significativos de la dimensión IPORIGEN

La Figura 5.6 muestra un ejemplo en donde se descartan cinco de un total de 8 utilizando esta metodología.

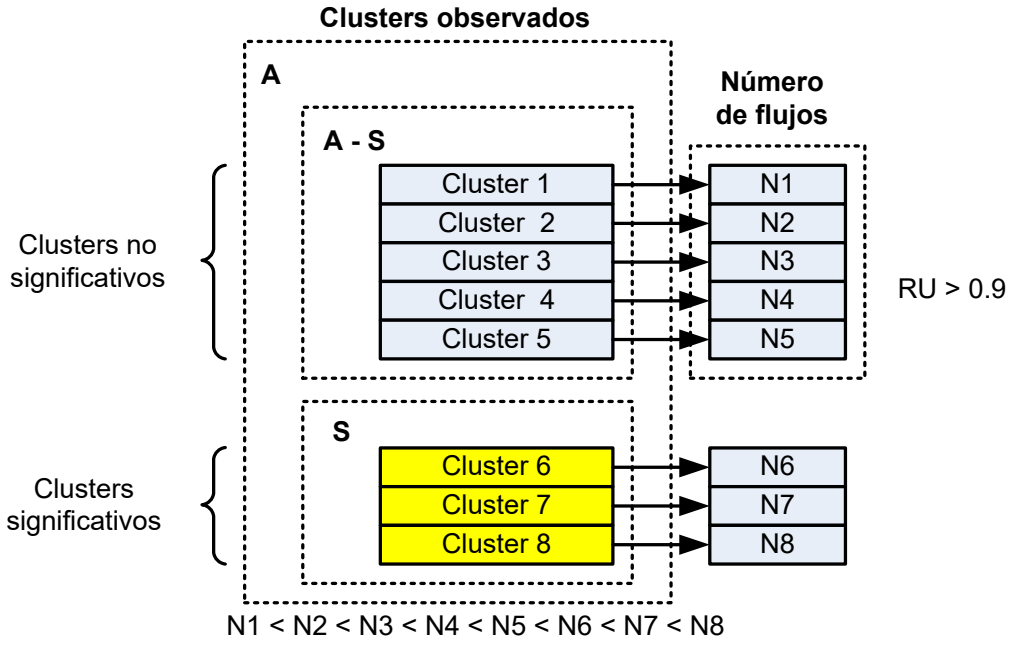


Figura 5.6 Extracción de 3 clusters significativos de un total de 8

La metodología general es descrita a continuación (ver Figura 5.5):

1. Se utiliza el archivo de texto generado por el proceso anterior como entrada para este subproceso. El archivo contiene la información de flujos del tráfico en los últimos cinco minutos.
2. La información de flujos en el archivo es agregada para obtener un arreglo multidimensional cuyas filas contienen información de flujos diferentes. Esta agregación consiste en sumar las longitudes y cantidades de paquetes de las filas que contienen informaciones de flujo comunes.
3. Se extraen tres arreglos uno conteniendo las longitudes totales, otro conteniendo sólo los valores de la dimensión específica y el último conteniendo la cantidad total de paquetes.
4. Se agrega la información según los valores comunes en el arreglo de las dimensiones. Este proceso consiste en sumar las longitudes totales y la cantidad total de paquetes de las ubicaciones donde aparecen valores de dimensión comunes. Asimismo se crea un nuevo arreglo que contiene la cantidad total de flujos contenida en cada valor de la dimensión.
5. Los valores del arreglo cantidad total de flujos son ordenados de menor a mayor. Asimismo, los valores de los demás arreglos son reubicados según el orden de este último. En caso de coincidencia de 2 ó más valores de cantidad de flujo, se ordena según el valor del campo cantidad de paquetes. Si aún así hay coincidencia se procede a ordenar según el valor del campo longitud.

6. Se calcula la incertidumbre relativa de los valores del arreglo cantidad total de flujos. Si no se supera el valor de 0.9, se elimina el último elemento (el mayor valor) y se vuelve a calcular la incertidumbre relativa con los valores restantes. El proceso se repite hasta que el cálculo de la incertidumbre relativa de los valores restantes supere el valor de 0.9 por primera vez.
7. Los valores que fueron eliminados en este último proceso, son los números de flujos de los clusters relevantes o significativos.

5.2.2 Clasificación de flujos significativos en clases de comportamiento

Este subproceso realiza la clasificación de cada cluster (conjunto de flujos) en 27 clases de comportamiento definidas según ciertas condiciones. Se aplica un enfoque de la teoría de la información para caracterizar el comportamiento de los clusters o flujos significativos.

La metodología empleada conlleva a un esquema natural de clasificación del comportamiento el cual agrupa cada cluster en 27 clases con distintos patrones de comportamiento.

Cada flujo o registro de flujo de un cluster extraído en el subproceso anterior comparten un mismo valor en el campo de la dimensión que se está analizando (dimensión fija); sin embargo, cada flujo en el cluster puede tomar cualquier valor posible en las otras tres dimensiones (dimensiones libres). Por consiguiente, cada flujo en un cluster induce una distribución de probabilidad en cada una de las tres dimensiones libres y se puede definir la incertidumbre relativa (RU) de los clusters de tales dimensiones.

Para cada cluster significativo se denota a las dimensiones libres como X, Y y Z. El orden de las notaciones se define según la convención establecida en la Tabla 5.1.

Tabla 5.1 Convención para la notación de las dimensiones libres (Fuente: [1])

Dimensión fija	Dimensiones Libres		
	X	Y	Z
IPORIGEN	PORIGEN	PDESTINO	IPDESTINO
IPDESTINO	PORIGEN	PDESTINO	IPORIGEN
PORIGEN	PDESTINO	IPORIGEN	IPDESTINO
PDESTINO	PORIGEN	IPORIGEN	IPDESTINO

Cada cluster es caracterizado por un vector de incertidumbres relativas correspondiente a cada una de las dimensiones libres. El vector es denotado por [RU_X, RU_Y, RU_Z] donde a su vez, cada RU se divide en 3 categorías según su valor.

Cada categoría está etiquetada por los números 0 (bajo), 1 (medio) y 2 (alto) según el criterio mostrado en la Tabla 5.2.

Tabla 5.2 Condiciones para el etiquetado de las incertidumbres relativas (Fuente: [1])

CATEGORÍA	CONDICIÓN
L=0 (bajo)	$0 \leq RU \leq \zeta$
L=1 (medio)	$\zeta < RU < 1 - \zeta$
L=2 (alto)	$1-\zeta \leq RU \leq 1$

El valor del parámetro ζ depende de la dimensión libre. Si la dimensión libre es IPORIGEN o IPDESTINO, entonces ζ=0.3. Si la dimensión libre es PORIGEN o PDESTINO, entonces ζ=0.2. Este proceso de etiquetado clasifica los clusters en 27 clases de comportamiento posibles representados por un vector de etiquetas según se muestra a continuación la ecuación 4.2:

$$[L(RU_X), L(RUy), L(RUz)] \in \{0,1,2\}^3 \quad (5.2)$$

Para facilitar la referencia de cada clase de comportamiento, el vector de etiquetas vector es tratado como un número entero llamado identificador de clase según la ecuación 4.3: Cálculo del identificador de clase

$$\text{Identificador de clase} = 3^2L(RU_X) + 3L(RUy) + L(RU_X) \in \{0,1,2,3,4\ldots,26\} \quad (5.3)$$

Cada clase de comportamiento se refiere como BC_N, (Behavior Class N) donde "N" es el identificador de clase. Los clusters extraídos utilizando las diferentes dimensiones fijas tienen sus propias clases de comportamiento y cada uno tiene significado e interpretación diferente. En este documento se denota como IPORIGEN BC_N. IPDESTINO BC_N, PORIGEN BC_N y PDESTINO BC_N a las clases de comportamiento en las respectivas dimensiones.

Para clasificar los clusters o flujos significativos, según la metodología descrita, se realiza el procedimiento que es ilustrado en la Figura 5.7, el cual es utilizado para la clasificación de un solo cluster significativo de la dimensión IPORIGEN. El proceso es análogo para los demás clusters de la misma dimensión así como para los clusters de otras dimensiones.

1. Se utiliza el arreglo multidimensional generado por el subproceso anterior como entrada para este subproceso. Este arreglo contiene los valores de la dimensión fija de los clusters, las sumas de longitudes, la cantidad total de paquetes y la cantidad total de flujos de cada cluster significativo del tráfico observado en los últimos 5 minutos.
2. Cada valor que representan un cluster significativo (valor de la dimensión fija) de este arreglo multidimensional es comparado con cada línea del arreglo que contiene todos los flujos extraídos (obtenido durante el proceso anterior). Por cada cluster se genera 4 arreglos cada uno conteniendo los valores de las 4 dimensiones de cada uno de los flujos de cada cluster significativo.

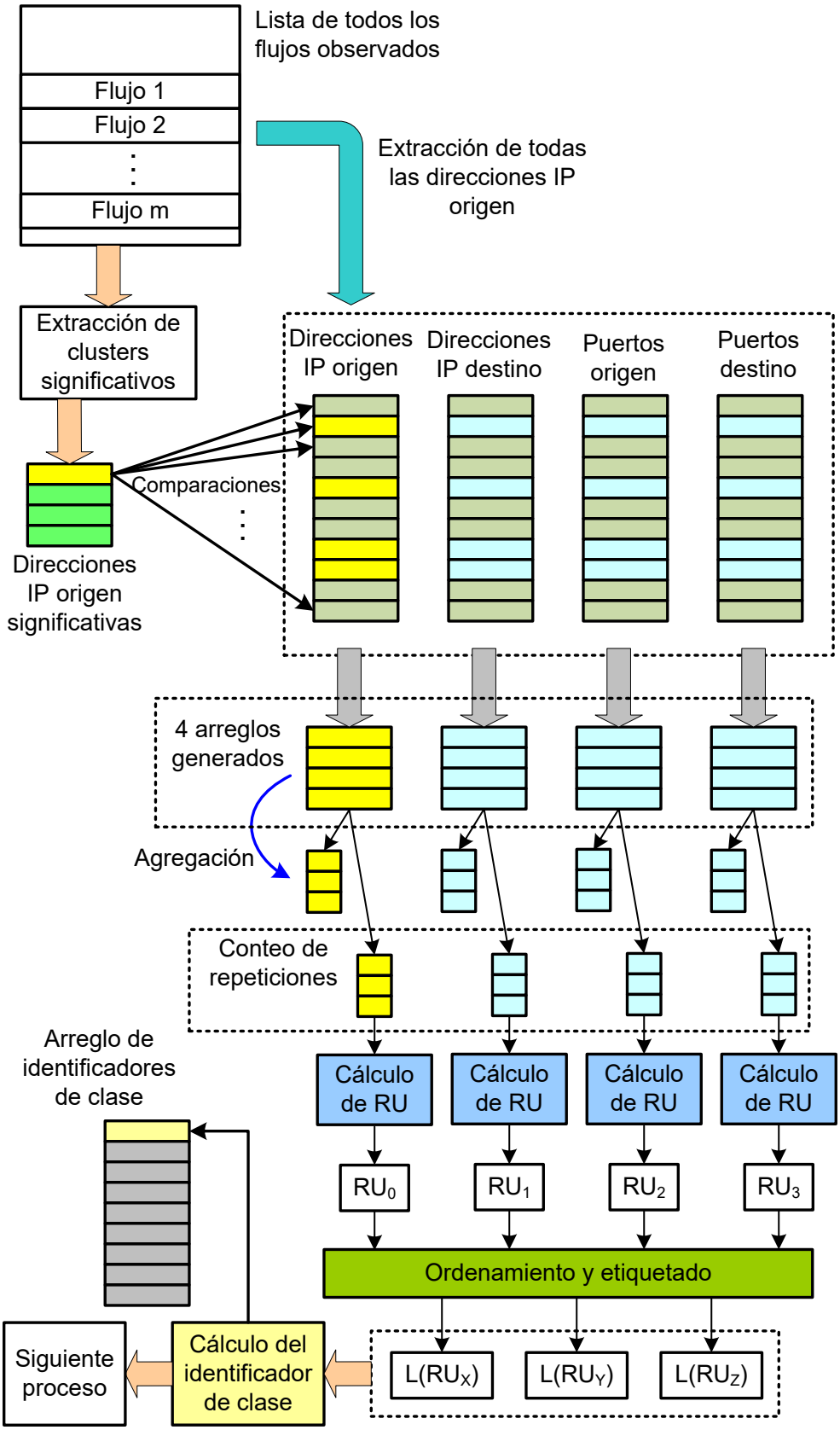


Figura 5.7 Clasificación de un cluster significativo de la dimensión IPORIGEN

3. La información de cada arreglo es agregada y se genera un nuevo arreglo temporal por cada arreglo conteniendo el número de repeticiones de la información. Se calcula el RU de

cada dimensión utilizando el respectivo arreglo temporal.

4. Se determina el número de etiqueta para cada valor de RU según el criterio que se indica en la Tabla 5.2 y utilizando el respectivo valor del parámetro ζ según la dimensión a la que pertenece cada arreglo.

5. Se calcula el identificador de clase conforme al orden de las dimensiones libres indicado en la Tabla 5.1 y según la ecuación 5.3. Para ello, se utiliza una sentencia de selección para la dimensión fija. El identificador de clase es almacenado en un arreglo de números enteros.

6. El programa continúa con la determinación de los estados dominantes para el presente cluster bajo análisis. Luego se procesa de la misma manera el siguiente cluster de la lista y así sucesivamente.

Se introducen tres métricas que describen tres características temporales de las clases de comportamiento a lo largo del tiempo. Tales métricas se muestran en la Tabla 5.3.

Tabla 5.3 Métricas que describen las características temporales de las clases de comportamiento (Fuente: [1])

Métrica	Descripción	Definición matemática
Popularidad Π_i	Número de intervalos de tiempo en promedio en donde se observa la aparición de una clase de comportamiento.	$\Pi_i = \frac{O_i}{T}$ $0 \leq \Pi_i \leq 1$
Tamaño promedio Σ_i	Número promedio de clusters que pertenecen a una clase de comportamiento particular en cada intervalo de tiempo donde se observa la clase.	$\Sigma_i = \sum_{j=1}^{j=T} \frac{C_{ij}}{O_i} = \frac{1}{O_i} \sum_{j=1}^{j=T} C_{ij}$
Volatilidad Ψ_i	Mide la tendencia de que una clase de comportamiento contenga diferentes clusters todo el tiempo, es decir, si los mismos clusters vuelven a aparecer con el tiempo o si tienden a aparecer nuevos clusters.	$\Psi_i = \frac{U_i}{\sum_{j=1}^{j=T} C_{ij}} = \frac{U_i}{\Sigma_i O_i}$ $0 \leq \Psi_i \leq 1$

El significado de cada operador en las definiciones matemáticas se muestra a continuación:

- i .- identificador de clase de comportamiento.
- T .- Número de intervalos de tiempo de 5 minutos.
- C_{ij} .- Número de clusters observados de la clase BC_i en el intervalo de tiempo j.
- O_i .- Número de intervalos de tiempo donde $C_{ij} \neq 0$ para la clase BC_i. ($O_i \leq T$).
- U_i .- Número de clusters únicos (que no se repiten) de la clase BC_i observados en todo el periodo de tiempo. Estos clusters aparecen una sola vez en cualquier y sólo en un único

intervalo de tiempo j de todos los “T” intervalos de tiempo del análisis.

Estas métricas, se calculan utilizando la información almacenada en la base de datos durante varios intervalos de tiempo de 5 minutos. Los detalles del mecanismo empleado para su cálculo se detallan en la sección 4.5 “Diseño de la interfaz de reportes gráficos”.

5.2.3 Determinación de estados dominantes

Este subproceso consiste en obtener un modelo del tráfico observado utilizando la técnica de análisis de estados dominantes para caracterizar la interacción entre clusters de una dimensión específica. Esta técnica se basa en conceptos de modelamiento estructural [64] y análisis de reconstrucción [65] de la teoría de sistemas.

El objetivo del análisis de estados dominantes es explorar la interacción o dependencia entre las dimensiones libres identificando subconjuntos de valores más simples llamados estados dominantes de un cluster o modelos estructurales según la referencia [64] los cuales representan o aproximan la naturaleza de los flujos que constituyen un cluster.

Los estados dominantes presentan las siguientes características:

- Reproducen la distribución de probabilidad original de las dimensiones libres de un cluster con exactitud razonable.
- Proporciona un resumen compacto de cada cluster de una dimensión.
- Proporcionan una interpretación aceptable del comportamiento de los clusters desde 4 perspectivas diferentes definidas por la dimensión que se analiza.
- Permiten predecir, con una exactitud razonable, el tipo habitual de transacciones existentes en la red y como se comportan cada una de ellas.

Para extraer los estados dominantes de un cluster, primero se determinan valores sustanciales de cada dimensión libre según el siguiente procedimiento:

1. Se realiza un reordenamiento de las dimensiones libres del cluster según su incertidumbre relativa de menor a mayor. En caso de empate en las incertidumbres relativas, la dimensión X precede a Y ó Z y la dimensión Y precede a Z. Las dimensiones reordenadas se denotan por “A”, “B” y “C” tal que $RU_A \leq RU_B \leq RU_C$.
2. Se determinan los valores sustanciales de la dimensión “A”. Se calcula la probabilidad marginal de cada valor “a” de la dimensión “A”. Si la probabilidad marginal de “a” supera un umbral, se considera a “a” como valor sustancial, es decir :

$$p(a)=\sum_{b\in B}\sum_{c\in C} p(a,b,c)\geq\delta=0.2 \qquad (5.4)$$

δ: Valor umbral para la selección de valores sustanciales, se utiliza el valor 0.2.

En caso de que no exista ningún valor sustancial “a” el proceso se detiene.

3. Se explora la dependencia entre las dimensiones “A” y “B” calculando la probabilidad condicional de observación de un valor “b_j” en la dimensión “B” dado cada valor sustancial

"a_i" de la dimensión "A" previamente determinado, es decir:

$$p(b_j|a_i)=\sum_{b\in B}\frac{p(a_i,b_j,c)}{a}\geq\delta=0.2 \tag{5.5}$$

En caso de que no exista ningún valor sustancial "b_j" el proceso se detiene.

4. Por último, se calcula la probabilidad condicional de observación de un valor "c_k" en la dimensión "C" dado cada combinación de valores sustanciales "a_i" de la dimensión "A" y "b_j" de la dimensión "B" previamente determinados, es decir $p(c_k \mid a_i, b_j) \geq \delta$.

En caso de que exista o no exista algún valor sustancial "c_k" el proceso se detiene.

5. El proceso total termina cuando se han determinado todos los posibles valores sustanciales a_i de la dimensión "A", todos los posibles valores sustanciales b_j de la dimensión "B" dado cada a_j y todos los posibles valores sustanciales c_k de la dimensión "C" dado a_i y b_j.

La Figura 5.8 muestra un diagrama que representa de manera general el proceso de determinación de valores sustanciales de un cluster de cualquier dimensión.

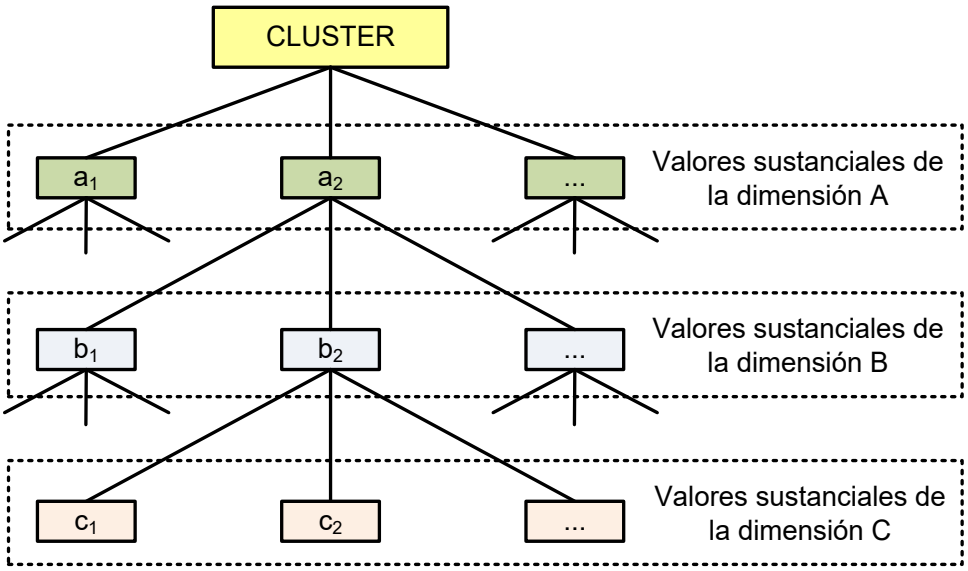


Figura 5.8 Estados dominantes de un cluster (Fuente: [1])

Los estados dominantes se determinan por la combinación de valores sustanciales de diferentes dimensiones.

Las cuatro formas posibles de estados dominantes que pueden existir, dependiendo de la existencia de valores sustanciales en las dimensiones A, B y C se muestran en la Tabla 5.4.

El símbolo "*" en la tabla indica valores aleatorios o arbitrarios en la dimensión libre respectiva, es decir, ningún elemento cumple la condición de estado dominante.

Tabla 5.4 Formas posibles de estados dominantes

Forma del estado dominante	Situación
$a_i \rightarrow (* , *)$	Existe un valor sustancial "a_i" en la dimensión "A". No existen valores sustanciales en la dimensión "B" ni en la "C".
$a_i \rightarrow b_j \rightarrow *$	Existe un valor sustancial "b_j" de la dimensión "B" dado el valor sustancial "a_i" de la dimensión "A". No existen valores sustanciales en la dimensión "C".
$a_i \rightarrow b_j \rightarrow c_k$	Existe un valor sustancial "c_k" de la dimensión "C" dado el valor sustancial "a_i" de la dimensión "A" y "b_j" de la dimensión "B".

En la Figura 5.9 se ilustra el proceso de reordenamiento de las dimensiones libres de un cluster con valor "IP1" de la dimensión fija IPORIGEN. El proceso se repite para los demás clusters de dicha dimensión. Según el esquema, para iniciar el proceso de determinación de estados dominantes no es necesaria la ejecución del proceso de clasificación de clusters significativos.

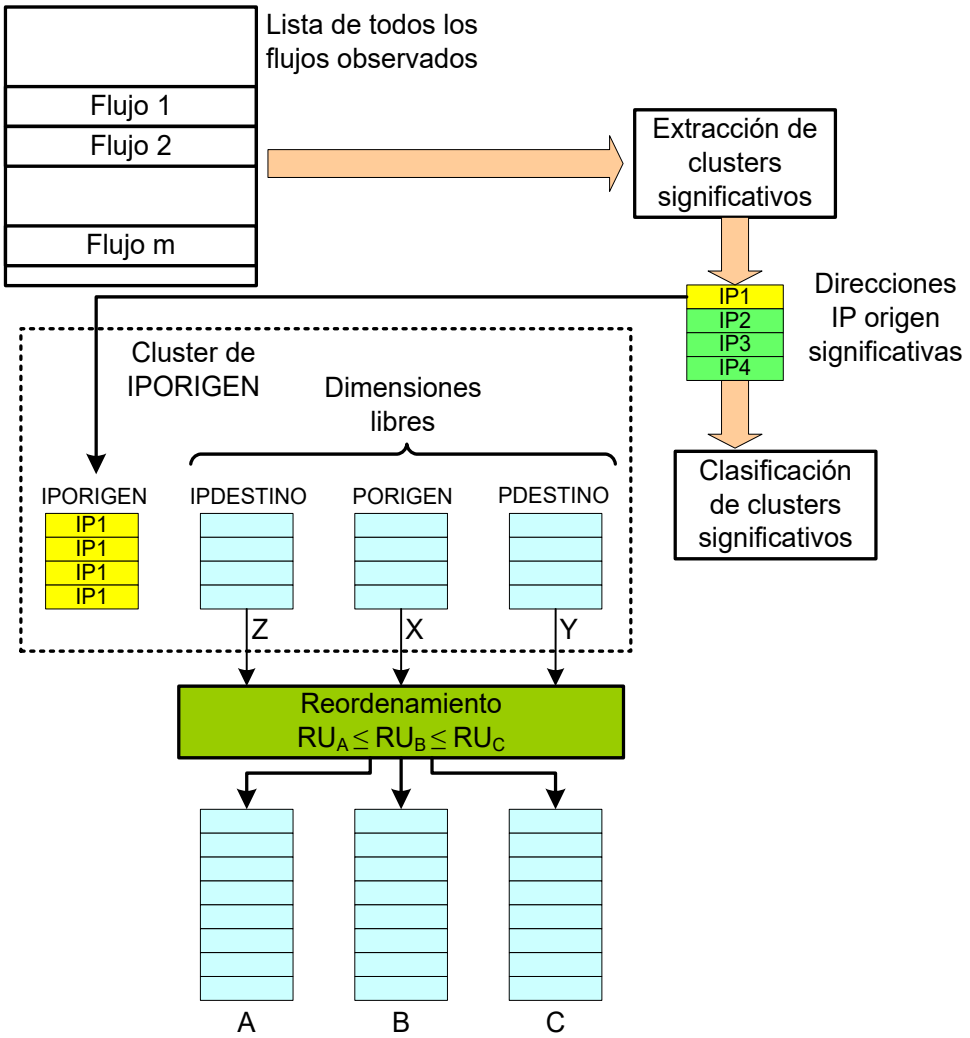


Figura 5.9 Proceso de reordenamiento de las dimensiones libres de un cluster

La Figura 5.10 ilustra en detalle el proceso de determinación de estados dominantes para un cluster significativo de una dimensión fija cualesquiera.

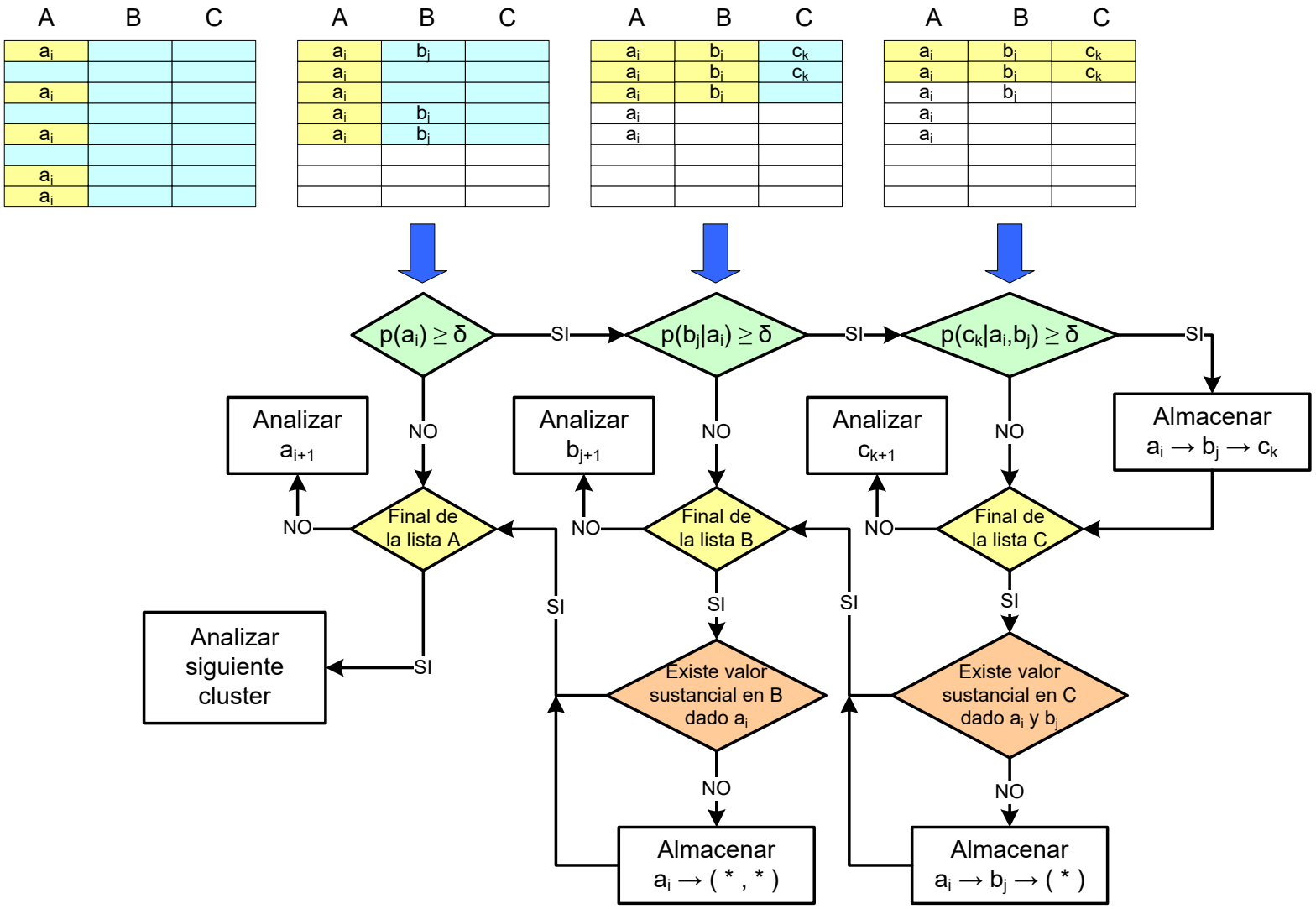


Figura 5.10 Proceso de determinación de estados dominantes de un cluster de la dimensión IPORIGEN

En la figura anterior se muestra el proceso de agregación en cada dimensión libre el cual permite obtener las probabilidades condicionales de cada elemento de la dimensión. Según se cumple o no las condiciones de estados dominantes, se almacenan dichos estados o modelos estructurales en una base de datos. Este proceso no tiene una secuencia determinada, es decir, pueden existir bloques de la figura que no son ejecutados. Esto depende de la naturaleza de los elementos y si cumplen o no las condiciones establecidas.

Al final del proceso, se obtiene una tabla con los estados dominantes o modelos estructurales de cada cluster junto con su clase de comportamiento almacenados en una base de datos MySQL. Según la referencia [1], una tabla de estados dominantes presenta 2 características importantes:

- Los clusters dentro de una clase de comportamiento tienen formas casi idénticas de modelos estructurales.
- El modelo estructural de un cluster presenta un resumen compacto de todos los flujos que lo constituyen revelando sólo la información esencial del cluster.

Dependiendo de la dimensión fija y de la clase de comportamiento de los estados dominantes o modelos estructurales de los clusters se escogen aquellos relacionados con la entrega o recepción de un servicio de red. Los detalles acerca de esto se desarrollan en la siguiente sección.

5.3 Definición de indicadores de producción de servicios

Para determinar indicadores de producción de servicios es necesario identificar, filtrar y contabilizar aquellos flujos relevantes del tráfico que están relacionados con transacciones de tipo cliente-servidor. El modelo obtenido en el proceso anterior permite identificar aquellos clusters o conjunto de flujos que son relevantes en comparación con el resto, así como el comportamiento de cada uno de éstos.

Según la metodología utilizada detallada en [1], la gran mayoría de clusters significativos presentan 3 perfiles de comportamiento canónicos según los 4 aspectos mencionados a continuación:

- Las clases de comportamiento a los que pertenecen.
- Características temporales de clusters individuales (frecuencia y estabilidad).
- Estados dominantes.
- El tamaño promedio de un flujo en términos de cantidad de bytes y de paquetes y sus variaciones.

Los 3 perfiles de comportamiento canónicos se muestran en la Tabla 5.5. De la tabla, se filtran las 8 clases de comportamiento que tienen el perfil canónico de servidores o servicios y se utilizan las métricas relacionados a estas clases como indicadores de

producción de servicios en una red.

Tabla 5.5 Perfiles de comportamiento canónico (Fuente: [1])

Perfil canónico	Dimensión	Clase de comportamiento	Ejemplos
Servidores o Servicios	IPORIGEN	$BC_{6,7,8}$	Web, DNS, email.
	IPDESTINO	$BC_{18,19,20}$	
	PORIGEN	BC_{23}	Tráfico de servicio agregado.
	PDESTINO	BC_{25}	
Heavy hitter hosts	IPORIGEN	$BC_{18,19}$	Rastreadores, proxies web y NAT boxes.
	IPDESTINO	$BC_{6,7}$	
Escaneos o exploits	IPORIGEN	$BC_{2,20}$	Exploits y escaners.
	IPDESTINO	$BC_{2,8}$	Blancos de escaneos.
	PDESTINO	$BC_{2,5,20,23}$	Tráfico de exploits agregados.

Desde la perspectiva de las dimensiones IPORIGEN e IPDESTINO, se tienen 3 clases de comportamiento de cada una que presentan el perfil canónico de servidores o servicios. La Tabla 5.6 muestra cada una de estas clases con las etiquetas de sus dimensiones libres y la interpretación de cada una.

Tabla 5.6 Clases de comportamiento de las dimensiones IPORIGEN e IPDESTINO que presentan el perfil canónico de servidores o servicios.

Dimensión fija	Clase de comportamiento	Dimensiones libres	Etiqueta	Interpretación
IPORIGEN	BC_6	PORIGEN	0	Uno o muy pocos números de puertos origen (puerto de algún servicio).
		PDESTINO	2	Variedad de números de puertos destino (puertos destino de clientes).
		IPDESTINO	0	Uno o muy pocas direcciones IP destino (poco número de clientes).
	BC_7	PORIGEN	0	Uno o muy pocos números de puertos origen (puerto de algún servicio).
		PDESTINO	2	Variedad de números de puertos destino (puertos destino de clientes).
		IPDESTINO	1	Variedad regular de direcciones IP destino (regular número de clientes).
	BC_8	PORIGEN	0	Uno o muy pocos números de puertos origen (puerto de algún servicio).
		PDESTINO	2	Variedad de números de puertos destino (puertos destino de clientes).
		IPDESTINO	2	Variedad de direcciones IP destino (gran número de clientes).
IPDESTINO	BC_{18}	PORIGEN	2	Variedad de números de puertos origen (puerto origen de clientes).
		PDESTINO	0	Uno o muy pocos números de puertos destino (puertos de servicios).
		IPORIGEN	0	Uno o muy pocas direcciones IP origen (poco número de clientes).
	BC_{19}	PORIGEN	2	Variedad de números de puertos origen (puerto origen de clientes).

		PDESTINO	0	Uno o muy pocos números de puertos destino (puertos de servicios).
		IPORIGEN	1	Variedad regular de direcciones IP origen (regular número de clientes).
	BC_{20}	PORIGEN	2	Variedad de números de puertos origen (puerto origen de clientes).
		PDESTINO	0	Uno o muy pocos números de puertos destino (puertos de servicios).
		IPORIGEN	2	Variedad de direcciones IP origen (gran número de clientes).

Estas clases de comportamiento representan patrones de comportamiento de un servidor comunicándose ya sea con unos pocos, varios o un gran número de clientes. Según pruebas realizadas en enlaces de red experimentales, tal como se indica en la referencia [1], los clusters asociados con este comportamiento tienden a tener una frecuencia relativamente alta y casi todos son estables ya que vuelven a aparecer en la misma clase o en clases similares. El tamaño promedio tanto en paquetes como en bytes muestran alta variabilidad, es decir, cada cluster consiste de flujos de tamaños diferentes.

Desde la perspectiva de las dimensiones PORIGEN y PDESTINO, se tiene una clase de comportamiento de cada una que representa el perfil canónico de servicios. La Tabla 5.7 muestra cada una de estas clases con las etiquetas de sus dimensiones libres y la interpretación de cada una.

Tabla 5.7 Clases de comportamiento de las dimensiones PORIGEN y PDESTINO que presentan el perfil canónico de servidores o servicios.

Dimensión fija	Clase de comportamiento	Dimensiones libres	Etiqueta	Interpretación
PORIGEN	BC_{23}	PDESTINO	2	Variedad de números de puertos destino (puertos destino de clientes).
		IPORIGEN	1	Variedad regular de direcciones IP origen (regular número de servicios).
		IPDESTINO	2	Variedad de direcciones IP destino (gran número de clientes).
PDESTINO	BC_{25}	PORIGEN	2	Variedad de números de puertos origen (puerto origen de clientes).
		IPORIGEN	2	Variedad de direcciones IP origen (gran número de clientes).
		IPDESTINO	1	Variedad regular de direcciones IP destino (regular número de servicios).

Estas clases representan el comportamiento agregado de un número relativamente pequeño de servidores comunicándose con un número mucho mayor de clientes en un número de puerto de un servicio específico.

Según la referencia [1], los clusters o grupos de flujos de cada clase de comportamiento tienen ciertas similitudes y diferencias entre sí. Por tal motivo, se definen indicadores que

son utilizados para distinguir subclases dentro de una clase de comportamiento. Para ello, se considera el tamaño promedio de los clusters tanto en bytes como en cantidad de paquetes y su variabilidad.

Cada flujo de un cluster se denota por f_i tal que $1 \leq i \leq m$, donde el valor de "m" el número total de flujos en el cluster. Sea PKT_i el número total de paquetes y BT_i el número total de bytes dentro del flujo f_i. En la Tabla 5.8 se definen 4 métricas para la diferenciación de los clusters dentro de las clases de comportamiento.

Tabla 5.8 Métricas para la diferenciación de clusters (Fuente: [1])

	Paquetes	**Bytes**
Número promedio	$\mu(PKT)=\sum_{i=1}^{i=m}\frac{PKT_i}{m}$	$\mu(BT)=\sum_{i=1}^{i=m}\frac{BT_i}{m}$
Coeficiente de variación	$CV(PKT)=\frac{\sigma(PKT)}{\mu(PKT)}$ Donde: $\sigma(PKT)=\sqrt{\frac{\sum_{i=1}^{i=m}(PKT_i-\mu(PKT))^2}{m}}$ $\sigma(PKT)$: Desviación estándar del número de paquetes.	$CV(BT)=\frac{\sigma(BT)}{\mu(BT)}$ Donde: $\sigma(BT)=\sqrt{\frac{\sum_{i=1}^{i=m}(BT_i-\mu(BT))^2}{m}}$ $\sigma(BT)$: Desviación estándar del número de bytes

Estos indicadores y las métricas relacionadas a las características temporales de las clases de comportamiento (ver Tabla 5.3 "Características Temporales") son considerados como indicadores de la producción de servicios de una red Ethernet.

Las métricas de popularidad, tamaño promedio y volatilidad caracterizan a cada clase de comportamiento que presenta el perfil canónico de servicios, mientras que las métricas de número promedio y coeficiente de variación del número total de paquetes y número total de bytes caracterizan a cada cluster dentro de cada clase de comportamiento que presenta el perfil canónico de servicios. Por consiguiente, se tienen indicadores tanto para las clases de comportamiento como para cada cluster dentro de cada clase.

La Tabla 5.9 resume cada uno de los indicadores previamente expuestos junto con una breve descripción relacionado a los objetivos de cada uno.

Tabla 5.9 Indicadores de producción de servicios

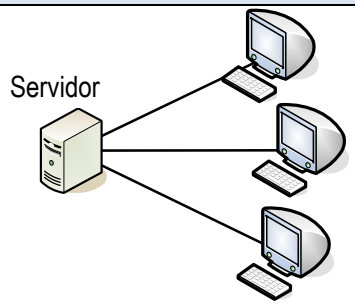
Indicador	**Referencia**	**Descripción**
Popularidad	Clase de comportamiento	Indica la frecuencia de aparición de la clase de comportamiento. Un valor alto (cercano a 1) indica que se están observando clusters de la clase la mayor parte del tiempo.
Tamaño promedio	Clase de comportamiento	Indica la cantidad promedio de clusters que pertenecen a la clase. Un valor alto indica que aparecen un mayor número

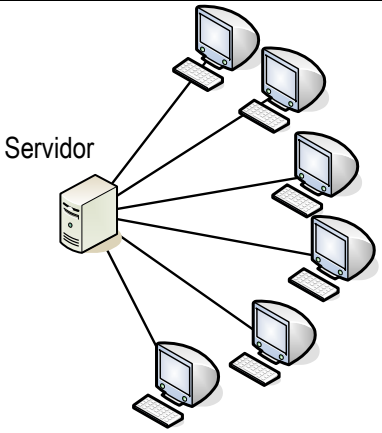
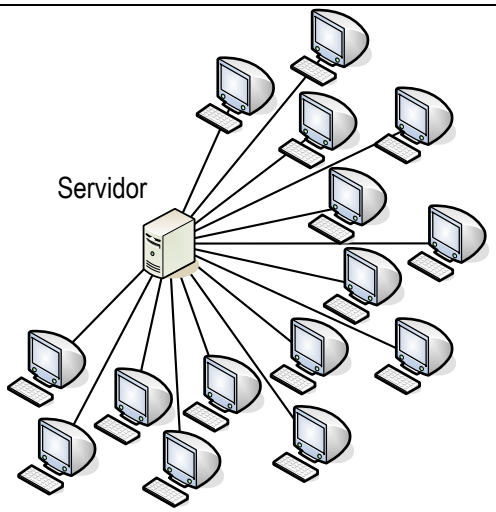
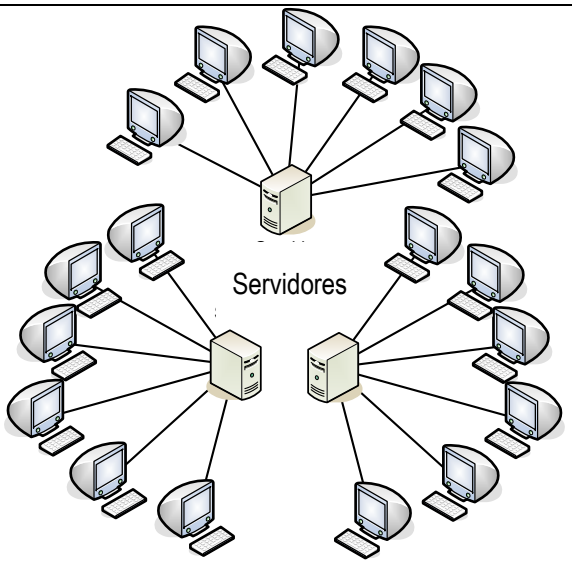
		de clusters pertenecientes a la clase en los intervalos de tiempo donde aparece la clase.
Volatilidad	Clase de comportamiento	Indica la tendencia de la clase a contener diferentes clusters todo el tiempo. Un valor alto (cercano a 1) indica que aparecen nuevos clusters de la clase la mayor parte del tiempo. Un valor bajo (cercano a 0) indica que aparecen los mismos clusters de la clase la mayor parte del tiempo.
Número promedio de paquetes	Conjunto de flujos (Cluster)	Indica la cantidad promedio de paquetes en cada uno de los flujos del cluster. Un valor alto indica que se observan muchos paquetes en los flujos del cluster.
Número promedio de bytes	Conjunto de flujos (Cluster)	Indica la cantidad promedio de bytes en cada uno de los flujos del cluster. Un valor alto indica que se observan muchos bytes (información) en los flujos del cluster.
Coeficiente de variación de paquetes	Conjunto de flujos (Cluster)	Indica la dispersión del número de paquetes respecto a su valor promedio. Un valor alto indica que la cantidad de paquetes en los flujos del cluster son diferentes y varían demasiado. Un valor bajo indica que las cantidades de paquetes se asemejan y están cercanas a su valor promedio.
Coeficiente de variación de bytes	Conjunto de flujos (Cluster)	Indica la dispersión del número de bytes respecto a su valor promedio. Un valor alto indica que la cantidad de bytes en los flujos del cluster son diferentes y varían demasiado. Un valor bajo indica que las cantidades de bytes se asemejan y están cercanas a su valor promedio.

Las interpretaciones mostradas en la Tabla 5.6, indican que las clases IPORIGEN BC_6, BC_7 y BC_8 presentan simetría con las clases IPDESTINO BC_{18}, BC_{19} y BC_{20} respectivamente. Igualmente las interpretaciones mostradas en la Tabla 5.7, indican que la clase PORIGEN BC_{23} presenta simetría con la clase PDESTINO BC_{25}.

Debido a la simetría observada, para el cálculo de los indicadores de producción de servicios, se considera únicamente las clases mostradas en la Tabla 5.10.

Tabla 5.10. Clases de comportamiento consideradas para el cálculo de los indicadores de producción de servicios

Clase de comportamiento	Descripción	Gráfico
IPORIGEN BC_6	Representa el comportamiento de un servidor comunicándose con pocos clientes. La dirección IP del servidor es identificada por la dirección IP que identifica al cluster de la clase.	

IPORIGEN BC_7	Representa el comportamiento de un servidor comunicándose con regular número de clientes. La dirección IP del servidor es identificada por la dirección IP que identifica al cluster de la clase.	
IPORIGEN BC_8	Representa el comportamiento de un servidor comunicándose con un gran número de clientes. La dirección IP del servidor es identificada por la dirección IP que identifica al cluster de la clase.	
PORIGEN BC_{23}	Representa el comportamiento agregado de un número regular de servidores alojando el mismo servicio (mismo número de puerto) y comunicándose con un número mucho mayor de clientes. El servicio específico es identificado por el puerto origen que identifica al cluster de la clase.	

5.4 Diseño de la aplicación de reportes

En esta sección se describe la estructura de la base de datos y el desarrollo de la interfaz, ambas implementadas en el servidor. En este servidor se almacenan 3 proyectos principales escritos en lenguaje JAVA los cuales son:

- Extracción: Implementa el proceso de recolección de información de flujos (Sección 4.2). Se encarga de recibir los datos provenientes del módulo de captura y almacenarlos en archivos de texto simples.
- Análisis: Implementa el proceso de modelamiento estructural basado en la teoría de la información. Contiene los subprocesos de extracción de flujos significativos, clasificación de los flujos significativos en clases de comportamiento y determinación de estados dominantes. (Sección 4.3).
- Reporte: Implementa la interfaz de visualización de reportes que contienen la información de indicadores de producción de servicios.

Los programas de estos proyectos interactúan con una base de datos MySQL, la cual se encarga de almacenar los resultados del proceso de modelamiento estructural basado en la teoría de la información. Luego, estos resultados son utilizados por la interfaz de reportes la cual realiza un procesamiento de dicha información y muestra los resultados en una interfaz web.

Primero, se demuestra en detalle el diseño de la estructura de la base de datos, detallando cada uno de sus tablas componentes. Luego se detalla el diseño de la interfaz de reportes demostrando la secuencia del procesamiento de la información y su apariencia en el entorno de aplicación web.

5.4.1 Diseño de la estructura de la base de datos

La base de datos utiliza un motor de base de datos MySQL versión 5.1.61. Tiene una estructura plana que consta de 13 tablas independientes entre sí. Todas las tablas tienen un campo llamado “fecha” la cual almacena la fecha de ingreso de una línea de datos a la tabla. La Figura 5.11 ilustra el modelado simple de la base de datos realizado con la herramienta de software “DeZign for Databases”. Se observa que no existen relaciones entre los campos ya que la base de datos es plana y cada una de las 13 tablas son independientes.

En las tablas siguientes (Tabla 5.11 a Tabla 5.14), se describen cada una de las tablas de la base de datos de la Figura 5.11, mostrando cada uno de sus campos y una breve descripción sobre la información que se almacena en cada tabla.

La Tabla 5.11, muestra el nombre, la descripción y los campos de la tabla “resumen_total” la cual almacena cifras totales observadas en cada intervalo de 5 minutos. Los nombres de los campos reflejan la naturaleza de la información almacenada.

La Tabla 5.12, muestra los nombres, las descripciones y los campos de las tablas de resumen de cada una de las 4 dimensiones. En ellas se almacenan los resultados del proceso de extracción de flujos significativos (sección 4.3.1) de cada dimensión en cada intervalo de 5 minutos, es decir, se acumula las cantidades totales de bytes, paquetes y

flujos de los clusters significativos y no significativos observados del tráfico analizado. Los campos mostrados son los mismos para cada una de las 4 tablas mostradas.

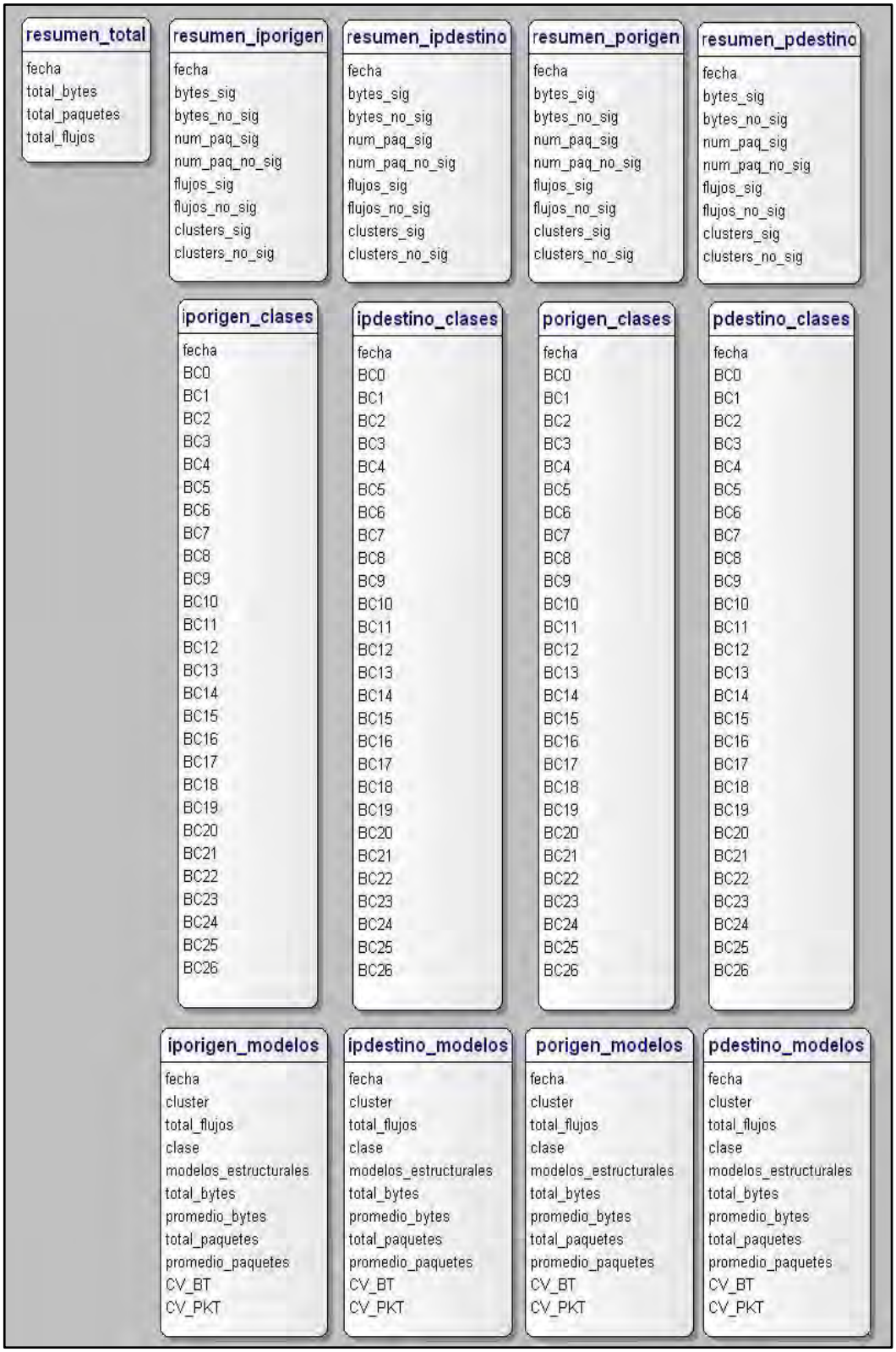


Figura 5.11 Modelado de la base de datos del servidor (Fuente: Software de modelado “DeZign for Databases”)

Tabla 5.11 Nombre y descripción de la tabla de resumen total

Nombre de Tabla	Descripción	Nombre de los campos
resumen_total	Almacena la cantidad total de bytes, paquetes y flujos observados en cada intervalo de tiempo de 5 minutos.	-fecha -total_bytes -total_paquetes -total_flujos

Tabla 5.12 Nombre y descripción de las tablas de resumen de cada dimensión

Nombre de Tabla	Descripción	Nombre de los campos
resumen_iporigen	Almacena la cantidad de bytes, paquetes, flujos y clusters significativos y no significativos observados en cada intervalo de 5 minutos. Estos datos son extraídos del tráfico utilizando la perspectiva de la dimensión IPORIGEN.	-fecha -bytes_sig -bytes_no_sig -num_paq_sig -num_paq_no_sig -flujos_sig -flujos_no_sig -clusters_sig -clusters_no_sig
resumen_ipdestino	Almacena la cantidad de bytes, paquetes, flujos y clusters significativos y no significativos observados en cada intervalo de 5 minutos. Estos datos son extraídos del tráfico utilizando la perspectiva de la dimensión IPDESTINO.	
resumen_porigen	Almacena la cantidad de bytes, paquetes, flujos y clusters significativos y no significativos observados en cada intervalo de 5 minutos. Estos datos son extraídos del tráfico utilizando la perspectiva de la dimensión PORIGEN.	
resumen_pdestino	Almacena la cantidad de bytes, paquetes, flujos y clusters significativos y no significativos observados en cada intervalo de 5 minutos. Estos datos son extraídos del tráfico utilizando la perspectiva de la dimensión PDESTINO.	

La Tabla 5.13, muestra los nombres, la descripción y los campos de las tablas referentes a las clases de comportamiento de cada una de las 4 dimensiones. En ellas se almacenan los resultados del proceso de clasificación de flujos significativos en clases de comportamiento (sección 4.3.2) de cada dimensión en cada intervalo de 5 minutos. Se almacenan las cantidades totales de clusters significativos observados en cada una de las 27 clases.

En total existen 28 campos, 1 de fecha y 27 campos BCN donde “N” es el identificador de clase. Los campos mostrados son los mismos para cada una de las 4 tablas mostradas (iporigen_clases, ipdestino_clases, porigen_clases, pdestino_clases).

La Tabla 5.14, muestra los nombres, la descripción y los campos de las tablas referentes a los modelos estructurales de cada una de las 4 dimensiones. En ellas se almacenan los resultados del proceso de determinación de estados dominantes (sección 4.3.3) de cada dimensión en cada intervalo de 5 minutos. Los campos mostrados son los mismos para cada una de las 4 tablas mostradas.

Tabla 5.13. Nombre y descripción de cada una de las tablas referentes a las clases de comportamiento

Nombre de Tabla	Descripción	Campos
iporigen_clases	Almacena la cantidad total de clusters observados en cada clase y en cada intervalo de 5 minutos utilizando la perspectiva de la dimensión IPORIGEN.	-fecha -BCN ; 0≤N≤26 (27 campos BCN)
ipdestino_clases	Almacena la cantidad total de clusters observados en cada clase y en cada intervalo de 5 minutos utilizando la perspectiva de la dimensión IPDESTINO.	
porigen_clases	Almacena la cantidad total de clusters observados en cada clase y en cada intervalo de 5 minutos utilizando la perspectiva de la dimensión PORIGEN.	
pdestino_clases	Almacena la cantidad total de clusters observados en cada clase y en cada intervalo de 5 minutos utilizando la perspectiva de la dimensión PDESTINO.	

Tabla 5.14 Nombre y descripción de cada una de las tablas referentes a los modelos estructurales

Nombre de Tabla	Descripción	Campos
iporigen_modelos	Almacena los modelos estructurales, el total de flujos y la clase de comportamiento de cada cluster significativo de la dimensión IPORIGEN observado en cada intervalo de 5 minutos. Asimismo, almacena las cantidades totales, promedios y coeficientes de variación del número de bytes y de paquetes de cada modelo estructural.	-fecha -cluster -total_flujos -clase -modelos_estructurales -total_bytes -promedio_bytes -total_paquetes -promedio_paquetes -CV_BT -CV_PKT
ipdestino_modelos	Almacena los modelos estructurales, el total de flujos y la clase de comportamiento de cada cluster significativo de la dimensión IPDESTINO observado en cada intervalo de 5 minutos. Asimismo, almacena las cantidades totales, promedios y coeficientes de variación del número de bytes y de paquetes de cada modelo estructural.	
porigen_modelos	Almacena los modelos estructurales, el total de flujos y la clase de comportamiento de cada cluster significativo de la dimensión PORIGEN observado en cada intervalo de 5 minutos. Asimismo, almacena las cantidades totales, promedios y coeficientes de variación del número de bytes y de paquetes de cada modelo estructural.	
pdestino_modelos	Almacena los modelos estructurales, el total de flujos y la clase de comportamiento de cada cluster significativo de la dimensión PDESTINO observado en cada intervalo de 5 minutos.	

	Asimismo, almacena las cantidades totales, promedios y coeficientes de variación del número de bytes y de paquetes de cada modelo estructural.	

5.4.2 Diseño de la interfaz de reportes

La interfaz de reportes está basada en un servidor web Apache Tomcat versión 6.0. Se utiliza el servidor tomcat debido a que es un software de código abierto y tiene la característica de ser una plataforma para el desarrollo y despliegue de aplicaciones y servicios web [63]. Tomcat implementa Servlets y JSP (Java Servers Pages) los cuales están especificados en el JCP (Java Community Process).

El diseño de la interfaz de reportes y de los demás procesos es realizado utilizando la herramienta NetBeans versión 7.0.1. En esta herramienta se crean los servlets y las páginas web que son desplegados en el servidor web Tomcat.

El esquema general del proceso de consulta de datos se ilustra en la Figura 5.12. El cliente solicita información al servidor que recibe la consulta y la procesa. Los datos de la consulta son transferidos a un servlet el cual realiza consultas a la base de datos. El servlet procesa los datos obtenidos y genera una página web con la información solicitada por el cliente. Por último, el servidor Tomcat envía la página web generada hacia el cliente.

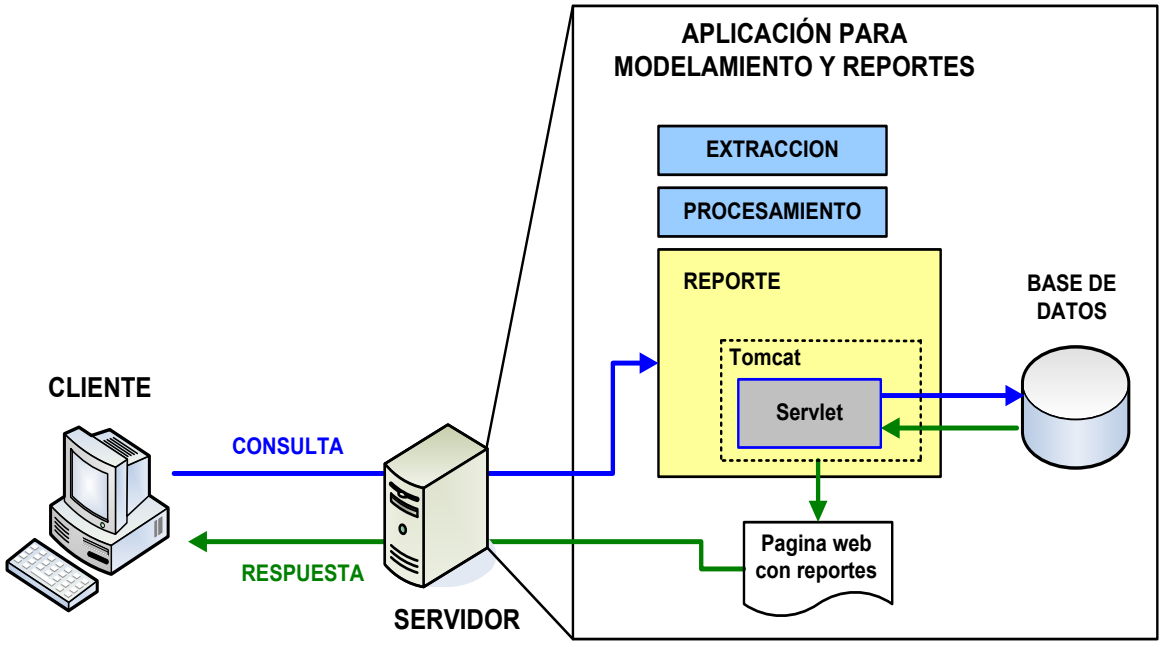


Figura 5.12 Proceso de consulta de datos de un cliente hacia el servidor

Los bloques “extracción” y “procesamiento” mostrados en la figura anterior son ejecutados cada 5 minutos. Los resultados del bloque “procesamiento” son almacenados en la base de datos. Estos resultados son utilizados por el bloque “reportes” para procesar las consultas de los clientes. A continuación se detalla las características de la página web principal y los servlets de la interfaz de reportes.

a. Página web principal

La Figura 5.13 muestra la apariencia de la página web principal. Se muestran 3 columnas, la primera para la selección de la fecha inicial y la fecha final que sirve para filtrar la información de la base de datos según la fecha ingresada.

En la segunda columna, se muestran 18 botones, cada uno enlazado con un servlet diferente. Al ser presionado un botón, el servlet respectivo realiza consultas a la base de datos, procesa la información y genera una página web con la información solicitada la cual es mostrada en una ventana del navegador web. Las características de cada servlet se detallan más adelante.

En la tercera columna, se muestra un cuadro donde se selecciona el número de puerto de servicios conocidos y otro cuadro donde se selecciona la ubicación del servicio a analizar, es decir, si el servicio está ubicado dentro de la red local (on-net) o si está ubicado fuera de la red local (off-net). La información de la tercera columna es utilizada solamente en los 4 primeros botones los cuales muestran información relevante respecto a los indicadores de la producción de servicios. Los demás botones muestran información secundaria que sirve para analizar los resultados de cada etapa del procesamiento.

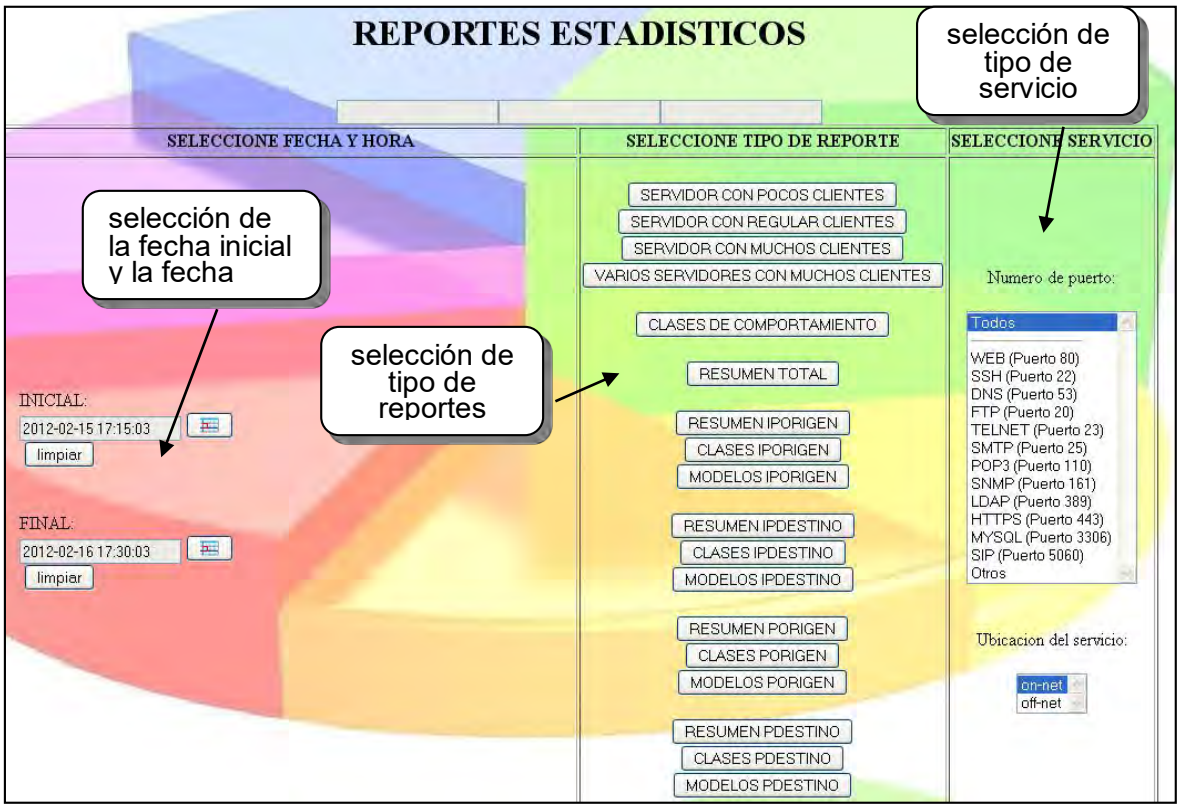


Figura 5.13 Página web principal del servidor web de reportes

A continuación se describen cada uno de los servlets mencionados. Se ilustran mediante diagramas de flujo y se muestran los resultados enviados al cliente.

b. Descripción de los servlets

En el diseño de la interfaz de reportes se utiliza 18 servlets cuya funcionalidad y resultados son descritos en detalle en esta sección. Para la explicación de cada uno de los servlets, estos son clasificados en 7 grupos.

b.1 Servlets de clases que presentan el perfil canónico de servidores

Estos servlets corresponden al procesamiento de clusters de las 3 clases de comportamiento que presentan el perfil canónico de servidores relacionados a la dimensión IPORIGEN (Tabla 5.5 “Perfiles de comportamiento canónico”).

Los nombres de cada uno de los servlets, las clases y el comportamiento que describen se muestran en la Tabla 5.15.

Tabla 5.15 Servlets de clases con perfil canónico de servidores

Nombre del servlet	Clase de comportamiento	Descripción del comportamiento
iporigen_bc6_html	IPORIGEN BC6	Servidor comunicándose con pocos clientes.
iporigen_bc7_html	IPORIGEN BC7	Servidor comunicándose con regular número de clientes.
iporigen_bc8_html	IPORIGEN BC8	Servidor comunicándose con un gran número de clientes.

El funcionamiento de cada uno de estos servlets es similar, por tal motivo se realiza una única descripción para los tres. Estos son los parámetros de entrada o variables que son ingresadas por el usuario desde la página web principal:

- Variables “fecha_inicial” y “fecha_final”: Indica la fecha inicial y final del análisis respecitvamente en formato “yyyy-MM-dd HH:mm:ss”, es decir, “año-mes-día hora-minutos-segundos”.
- Variable “puerto”: Si el cliente selecciona la opción “Todos” (Analizar todos los número de puerto), el valor del número de puerto es 0. Si selecciona la opción “Otros” (Analizar otros números de puerto diferentes a los mostrados en la interfaz web principal), el número de puerto es 1. Si selecciona cualquier otra opción particular, el valor del número de puerto coincide con el número de puerto del servicio elegido.
- Variable “loc_rem”: Si el cliente selecciona la opción “on-net”, el valor de esta variable es igual a la cadena: “local”. Si el cliente selecciona la opción “off-net”, el valor de esta variable es igual a la cadena: “remoto”.

El proceso que realiza el servlet se resume en los siguientes pasos:

1. Se consulta en la base de datos todos valores de los campos de la tabla “iporigen_modelos” menos el campo “fecha”.
2. Se filtran todos los registros cuyos campos “fecha” estén comprendidos entre los valores de las variables “fecha_inicial” y “fecha_final”.
3. Se filtran todos los registros cuyos campos “clase” sean iguales a 6, 7 ó 8 (Clase de

comportamiento IPORIGEN 6, 7 ó 8) según el nombre del servlet.

4. Se filtran los campos “clusters” de modo que solamente incluyan direcciones IP locales o remotas según indica el valor de la variable “loc_rem”.

5. Se filtran los valores del campo “modelos_estructurales” de modo que solamente se incluyan aquellos modelos que presenten cadenas de texto que indiquen el número puerto origen igual a los valores que se indica por medio de la variable “puerto”.

6. La información es ordenada, en primer lugar según los valores del campo “cluster” (IP origen) y en segundo lugar según los valores del campo “modelos_estructurales”.

7. La información en el arreglo multidimensional es sintetizada. Existen algunos registros cuyos campos “cluster” y “modelos_estructurales” se repiten en distintos intervalos de tiempo de 5 minutos. Únicamente estos registros son sintetizados en un solo registro. En el nuevo registro, los valores de los campos comunes permanecen sin cambio. Los valores de los demás campos son sintetizados de la siguiente manera: Los valores de los campos “total_flujos”, “total_bytes” y “total_paquetes” se suman y el resultado es almacenado en el campo respectivo del nuevo registro. Los valores de los campos “promedio_bytes” y “promedio_paquetes” del nuevo registro se calculan dividiendo la suma total de bytes o de paquetes entre la suma total de flujos. Se calcula el valor mínimo y máximo de los campos “CV_BT” y “CV_PKT”.

8. Se realiza el conteo de los registros que fueron sintetizados y este valor se agrega al final del nuevo registro. Este valor indica con qué frecuencia (número de intervalos de 5 minutos) se observa la aparición del cluster en todo el intervalo de tiempo que comprende el análisis.

9. Se procede de igual manera con los demás registros que requieren ser sintetizados.

10. Los nuevos registros sintetizados y los demás registros que no fueron sintetizados son impresos en una tabla dentro de una página web. Los campos o columnas de dicha tabla se muestra en la Tabla 5.16. Los valores de todos los registros sintetizados son almacenados en un nuevo arreglo multidimensional.

11. En el nuevo arreglo, existen algunos registros cuyos valores de la columna “IP del servidor” (campo “cluster” en la base de datos) se repiten debido a que algunos registros contienen más de un modelo estructural. Estos registros son sintetizados en un solo registro utilizando la misma metodología explicada en el paso anterior.

12. Al igual como se hizo anteriormente, se realiza el conteo de los registros que fueron sintetizados y este valor se agrega al final del nuevo registro. Este valor indica la cantidad de modelos estructurales diferentes que se observa en cada cluster en todo el intervalo de tiempo que comprende el análisis.

13. Se procede de igual manera con los demás registros que requieren ser sintetizados.

14. Los nuevos registros sintetizados y los demás registros que no fueron sintetizados son colocados en una tabla en donde son ordenados, en primer lugar de mayor a menor según el número total de paquetes, en segundo lugar según el número total de bytes y en tercer lugar según la cantidad total de flujos. El arreglo ordenado es impreso en una tabla dentro de la misma página web. Los campos o columnas de dicha tabla se muestran en la Tabla 5.17. Se agrega la columna “orden” y la columna “nombre de dominio” cuyo contenido se detalla en dicha tabla.
15. Los valores de todos los nuevos registros sintetizados son almacenados nuevamente en un nuevo arreglo multidimensional.
16. Se genera una tabla resumen que contiene únicamente un registro el cual es producto de la síntesis de los registros de la tabla anterior utilizando la misma metodología explicada anteriormente para la síntesis. Los campos o columnas de dicha tabla se muestra en la Tabla 5.18. Se agregan las columnas “Clase de comportamiento”, “Servicio” y “Total servidores” cuyo contenido se detalla en dicha tabla.

Tabla 5.16 Campos de la tabla resumen que contiene los registros sintetizados extraídos directamente de la base de datos

Campo de la tabla	Descripción
IP del servidor	Dirección IP que representa a un servicio (dirección IP de un servidor).
Modelo Estructural	Modelo estructural correspondiente a la dirección IP del servidor.
Total Flujos	Suma de las cantidades totales de flujos con la dirección IP origen y modelo estructural respectivo observados en cada intervalo de 5 minutos.
Total Bytes	Suma de la cantidad total de bytes de aquellos flujos que contienen la dirección IP origen y modelo estructural respectivo observados en cada intervalo de 5 minutos.
Promedio Bytes	Cantidad promedio de bytes con respecto a la cantidad total de flujos.
CV_BT mínimo	Valor mínimo del coeficiente de variación de bytes de todos los intervalos de 5 minutos.
CV_BT máximo	Valor máximo del coeficiente de variación de bytes de todos los intervalos de 5 minutos.
Total Paquetes	Suma de la cantidad total de paquetes de aquellos flujos que contienen la dirección IP origen y modelo estructural respectivo observados en cada intervalo de 5 minutos.
Promedio Paquetes	Cantidad promedio de paquetes con respecto a la cantidad total de flujos.
CV_PKT mínimo	Valor mínimo del coeficiente de variación de paquetes de todos los intervalos de 5 minutos.
CV_PKT máximo	Valor máximo del coeficiente de variación de paquetes de todos los intervalos de 5 minutos.

Frecuencia	Cantidad de intervalos de 5 minutos donde aparecen aquellos flujos que contienen la dirección IP origen y modelo estructural respectivo.

Tabla 5.17 Campos de la tabla resumen que contiene los registros sintetizados según la dirección IP del servidor

Campo de la tabla	Descripción
Orden	Indica la numeración de cada registro.
IP del servidor	Dirección IP que representa a un servicio (dirección IP de un servidor).
Nombre de dominio	Nombre de dominio público correspondiente a la dirección IP del servidor.
Aquí se ubican los campos: Total Flujos, Total Bytes, Promedio Bytes, CV_BT mínimo, CV_BT máximo, Total Paquetes, Promedio Paquetes, CV_PKT mínimo, CV_PKT máximo y Frecuencia. La descripción de cada uno de estos campos se encuentra en la Tabla 5.16.	
Cantidad de modelos estructurales	Cantidad total de modelos estructurales diferentes correspondiente a la IP del servidor.

Tabla 5.18 Campos de la tabla que contiene el resumen

Campo de la tabla	Descripción
Clase de comportamiento	Clase de comportamiento actual: IPORIGEN BC6, BC7 ó BC8.
Servicio	Nombre del servicio que está siendo analizado.
Total servidores	Cantidad total de servidores diferentes detectados en el análisis y que ofrecen el respectivo servicio.
Aquí se ubican los campos: Total Flujos, Total Bytes, Promedio Bytes, CV_BT mínimo, CV_BT máximo, Total Paquetes, Promedio Paquetes, CV_PKT mínimo, CV_PKT máximo, Frecuencia y Cantidad de modelos estructurales. La descripción de cada uno de estos campos se encuentra en la Tabla 5.16.	

b.2 Servlet "porigen_bc23_html"

Este servlet corresponde al procesamiento de clusters de la clase de comportamiento PORIGEN 23 la cual presenta el perfil canónico de servidores (Tabla 5.5 "Perfiles de comportamiento canónico"). Esta clase representa el comportamiento de varios servidores que ofrecen el mismo servicio a muchos clientes diferentes.

Los parámetros de entrada son los mismos que los servlets anteriores. Asimismo, el proceso también es similar, sin embargo, los 5 primeros pasos son diferentes. Tales pasos se detallan a continuación:

1. Se consulta en la base de datos todos valores de los campos de la tabla "porigen_modelos" menos el campo "fecha".
2. Se filtran todos los registros cuyos campos "fecha" estén comprendidos entre los valores de las variables "fecha_inicial" y "fecha_final".
3. Se filtran todos los registros cuyos campos "clase" sea igual a 23 (Clase de

comportamiento PORIGEN 23).

4. Se filtran los campos “clusters” de modo que solamente se incluyan los números puerto origen igual a los valores que se indica por medio de la variable “puerto”.

5. Se filtran los valores del campo “modelos_estructurales” de modo que solamente incluyan aquellos modelos que presenten cadenas de texto que indiquen direcciones solamente IP locales o remotas según indica el valor de la variable “loc_rem”.

6. Los demás pasos son similares al de los servlets anteriores. La estructura de cada una de las tablas también es la misma.

b.3 Servlet “comportamiento”

Este servlet muestra las 3 métricas (Tabla 5.3) que describen las características temporales de aquellas clases de comportamiento que presentan el perfil canónico de servidores. Estas clases son las mismas que se analizaron en los servlets anteriores (IPORIGEN BC6, BC7, BC8 y PORIGEN BC23).

Los parámetros de entrada o variables que son ingresadas por el usuario desde la página web principal son “fecha_inicial” y “fecha_final”, que indican la fecha inicial y final del análisis respectivamente en formato “yyyy-MM-dd HH:mm:ss”, es decir, “año-mes-día hora-minutos-segundos”.

El proceso para el cálculo de la métrica “popularidad” se resume en los siguientes pasos:

1. Se consulta en la base de datos todos valores de los campos “BC6”, “BC7” y “BC8” de la tabla “iporigen_clases” y los valores del campo “BC23” de la tabla “porigen_clases”.

2. Se filtran todos los registros cuyos campos “fecha” estén comprendidos entre los valores de las variables “fecha_inicial” y “fecha_final”.

3. Se contabiliza el número de registros diferentes de cero de los campos “BC6”, “BC7” y “BC8” de la tabla “iporigen_clases” y del campo “BC23” de la tabla “iporigen_clases” y se divide entre el número total de registros de cada campo. Con esto, se obtiene la métrica “popularidad” de cada clase de comportamiento.

El proceso para el cálculo de la métrica “tamaño promedio” se resume en los siguientes pasos:

1. Se consulta en la base de datos todos valores de los campos “BC6”, “BC7” y “BC8” de la tabla “iporigen_clases” y los valores del campo “BC23” de la tabla “porigen_clases”.

2. Se filtran todos los registros cuyos campos “fecha” estén comprendidos entre los valores de las variables “fecha_inicial” y “fecha_final”.

3. Se suman todos los valores de los registros de los campos “BC6”, “BC7” y “BC8” de la tabla “iporigen_clases” y del campo “BC23” de la tabla “iporigen_clases” y se divide entre el número total de registros diferentes de cero de cada campo. Con esto, se obtiene la

métrica “tamaño promedio” de cada clase de comportamiento.

El proceso para el cálculo de la métrica “volatilidad” se resume en los siguientes pasos:

1. Se consulta en la base de datos todos valores de los campos “fecha”, “cluster” y “clase” de la tabla “iporigen_modelos”.
2. Se filtran todos los registros cuyos campos “fecha” estén comprendidos entre los valores de las variables “fecha_inicial” y “fecha_final”.
3. Para calcular la “volatilidad” de la clase IPORIGEN BC6, se filtran los registros cuyo contenido del campo “clase” sea el valor 6.
4. Se contabiliza el número total de clusters que aparecen solamente una sola vez en todo el intervalo de tiempo del análisis. Este valor se divide entre la suma de todos los valores del campo “BC6” de la tabla “iporigen_clases”, filtrados por fecha. El resultado corresponde a la volatilidad de la clase IPORIGEN BC6.
5. Se realiza un procedimiento análogo para calcular la popularidad de las clases IPORIGEN BC7 y BC8.
6. Para la clase de comportamiento PORIGEN BC23 se realiza un procedimiento análogo al anterior. Esta vez, se consultan los registros del campo “BC23” de la tabla “porigen_clases” y los registros “fecha”, “cluster” y “clase” de la tabla “porigen_modelos”.

El servlet genera una tabla con los datos numéricos de cada una de las métricas y 3 gráficos de barras para cada una de las métricas. La Figura 5.14 muestra un ejemplo de la tabla generada, la cual está contenida en un archivo pdf generado.

CLASES CON CON EL PERFIL CANONICO DE SERVICIOS

PROPIEDADES TEMPORALES

Formato de datos contenido

CLASE	Popularidad	Tamaño Promedio	Volatilidad
	Mide la cantidad de intervalos de 5 minutos en promedio donde se observaron clusters en la clase	Mide el numero promedio de clusters que pertenecen a la clase en cada intervalo de 5 minutos donde se observa la clase	Mide la tendencia de la clase de contener diferentes clusters todo el tiempo
IPORIGEN BC6	1,0000	7,7958	0,1123
IPORIGEN BC7	0,8028	1,2629	0,0614
IPORIGEN BC8	0,1799	1,1154	0,0862
PORIGEN BC23	0,0000	0,0000	0,0000

Figura 5.14 Ejemplo de una tabla de métricas

La Figura 5.15 muestra un ejemplo de la gráfica de barras de la métrica popularidad

generada por el servlet. La forma de la gráfica es similar para las demás métricas.

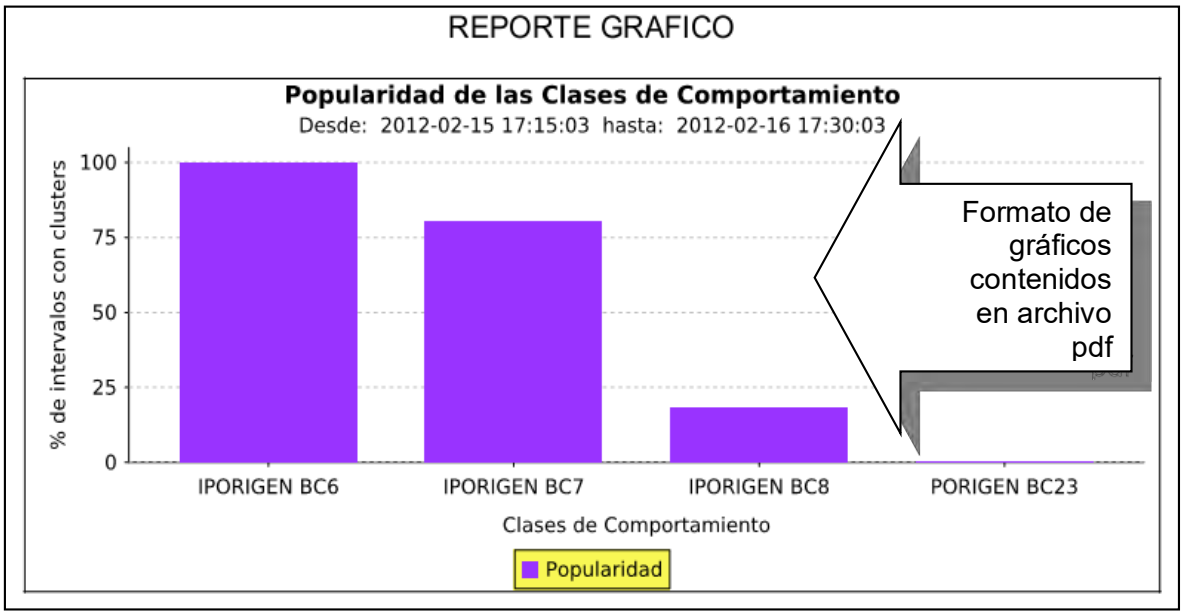


Figura 5.15 Ejemplo de un gráfico de barras de la métrica "popularidad"

b.4 Servlet "resumen_total"

Este servlet muestra 3 gráficos: número total de bytes, de paquetes y de flujos versus el tiempo. Se grafican puntos en cada intervalo de 5 minutos unidos con líneas rectas en cada gráfico. El valor de cada punto es extraído directamente del respectivo campo de la tabla "resumen_total" con los valores filtrados según la fecha de inicio y final. La Figura 5.16 muestra un ejemplo de una gráfica generada por el servlet "resumen_total" correspondiente a la cantidad total de bytes.

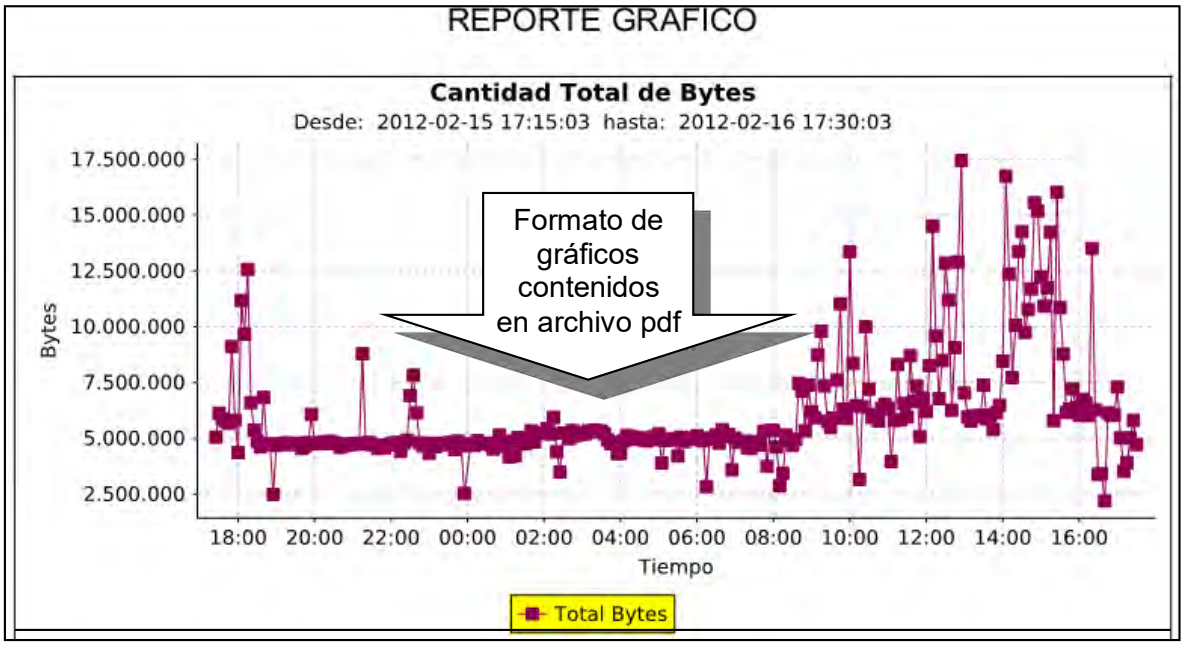


Figura 5.16 Ejemplo de la gráfica temporal de la cantidad total de bytes

Asimismo, el servlet genera 3 tablas que muestran las cifras totales, valores mínimo,

promedio, máximo, desviación estándar y varianza del número total de bytes, paquetes y flujos. También se muestra la cantidad total de registros (equivalente al número de intervalos de 5 minutos) consultados a la base de datos y el tiempo de análisis. Al final se presentan cifras aproximadas del nivel de tráfico a nivel de bits, bytes y paquetes. En la Figura 5.17 se ilustra un ejemplo para las 3 tablas resumen.

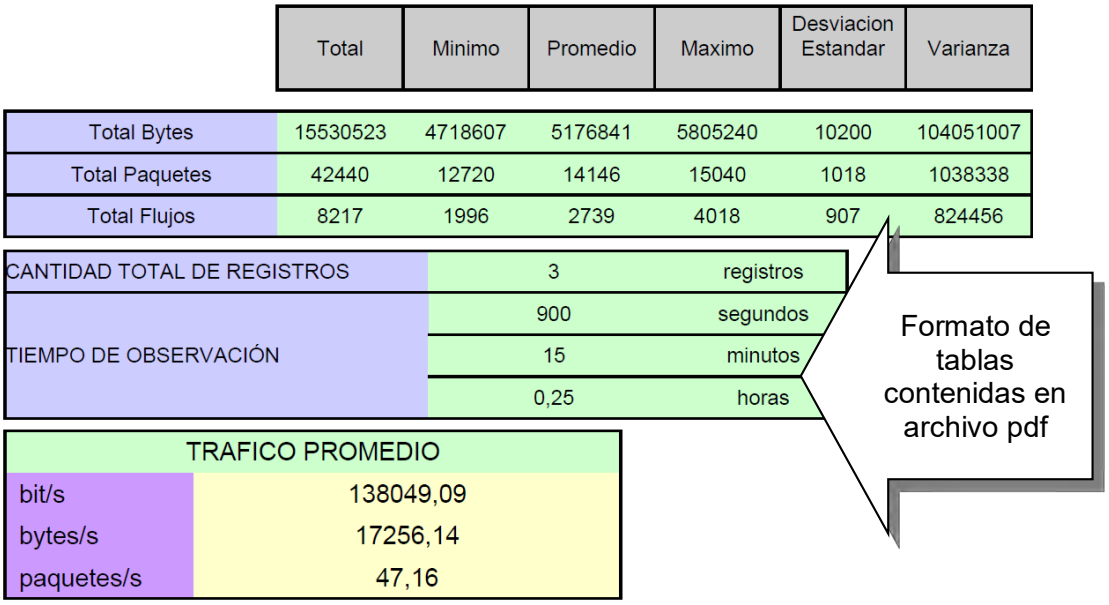


Figura 4.17 Ejemplo de tablas de resumen del servlet “resumen_total”

b.5 Servlets que muestran el resumen de cada dimensión

Estos servlets muestran los resultados del proceso de extracción de clusters significativos en cada dimensión. La Tabla 5.19 muestra el nombre de cada uno de los servlets junto con una breve descripción.

Tabla 5.19 Servlets que muestran el resumen de cada dimensión

Nombre del servlet	Descripción del Servlet
resumen_iporigen	Muestra el resumen de la cantidad de información significativa y no significativa desde la perspectiva de la dimensión IPORIGEN.
resumen_ipdestino	Muestra el resumen de la cantidad de información significativa y no significativa desde la perspectiva de la dimensión IPDESTINO.
resumen_porigen	Muestra el resumen de la cantidad de información significativa y no significativa desde la perspectiva de la dimensión PORIGEN.
resumen_pdestino	Muestra el resumen de la cantidad de información significativa y no significativa desde la perspectiva de la dimensión PDESTINO.

Cada servlet genera 4 gráficos de series temporales, cada uno conteniendo la siguiente información:

- Número total de bytes significativos y no significativos versus el tiempo.

- Número total de paquetes significativos y no significativos versus el tiempo.
- Número total de flujos significativos y no significativos versus el tiempo.
- Número total de clusters significativos y no significativos versus el tiempo.

Se grafican puntos en cada intervalo de 5 minutos unidos con líneas. El valor de cada punto es extraído directamente del respectivo campo de la tabla “resumen_iporigen” con los valores filtrados según la fecha de inicio y final para la dimensión IPORIGEN. Para las demás dimensiones el procedimiento es análogo. La Figura 5.18 muestra un ejemplo de una gráfica generada por el servlet “resumen_iporigen” correspondiente a la cantidad total de bytes significativos y no significativos.

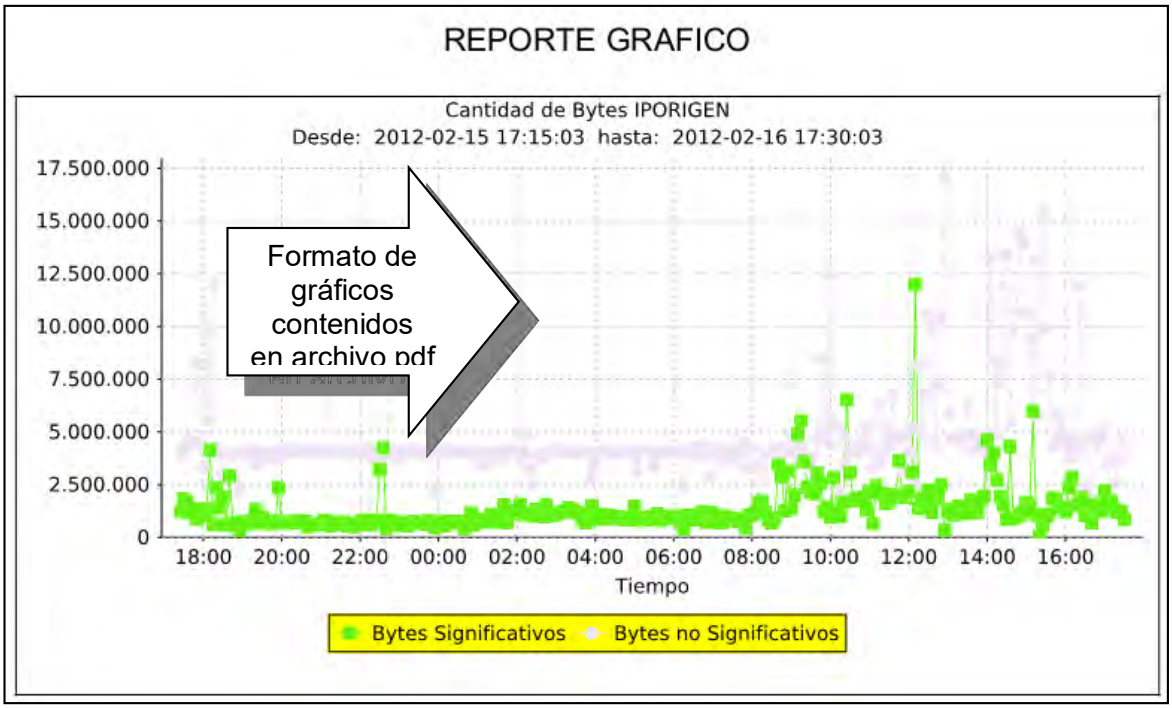


Figura 5.18 Ejemplo de una gráfica temporal de la cantidad total de bytes significativos y no significativos

Asimismo, cada servlet genera gráficas circulares con las cantidades totales significativa y no significativa de bytes, paquetes, flujos y clusters. La Figura 5.19 muestra un ejemplo de las gráficas circulares del reporte del resumen de la dimensión IPORIGEN generadas por el servlet.

También, el servlet genera una tabla que muestra las cifras totales, valores mínimo, promedio, máximo, desviación estándar y varianza del número total de bytes, paquetes, flujos y clusters tanto significativos como no significativos. En la Figura 5.20 se ilustra un ejemplo de la tabla de resumen para la dimensión IPORIGEN.

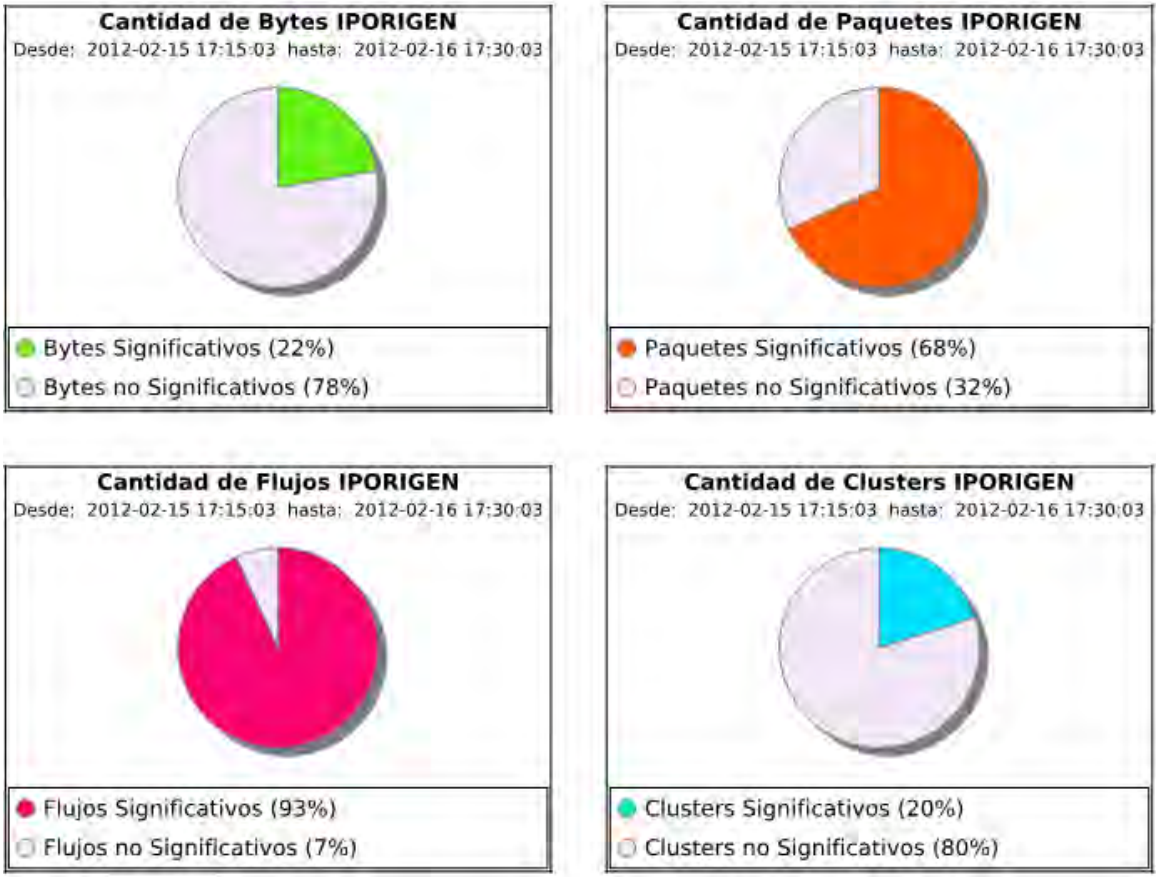


Figura 5.20 Ejemplo de gráficas circulares del reporte del resumen de dimensión IPORIGEN

	Total	Minimo	Promedio	Maximo	Desviacion Estandar	Varianza
Bytes Significativos	392475655	58950	1358047	11981780	0	-5451257
Bytes no Significativos	136559908	1263384	4725256	17094194	2792	7796533
Paquetes Significativos	2648584	659	9164	13922	1909	
Paquetes no Significativos	1241895	1033	4297	11717	15	
Flujos Significativos	488968	245	1691	8646	8	
Flujos no Significativos	36116	35	124	390	81	
Clusters Significativos	6316	11	21	72	0	0
Clusters no Significativos	25117	27	86	261	52	2720

Formato de tablas contenidas en archivo pdf

Figura 5.21 Ejemplo de tablas de resumen del servlet “resumen_iporigen”

b.6 Servlets que muestran las clases de comportamiento de cada dimensión

Estos servlets muestran las métricas producto de la clasificación de clusters significativos en clases de comportamiento para cada dimensión. La Tabla 5.20 muestra el nombre de cada uno de los servlets junto con una breve descripción.

Tabla 5.20 Servlets que muestran las clases de comportamiento de cada dimensión

Nombre del servlet	Descripción del Servlet
iporigen_clases	Muestra las métricas de cada una de las 27 clases de comportamiento de los clusters significativos de la dimensión IPORIGEN.
ipdestino_clases	Muestra las métricas de cada una de las 27 clases de comportamiento de los clusters significativos de la dimensión IPDESTINO.
porigen_clases	Muestra las métricas de cada una de las 27 clases de comportamiento de los clusters significativos de la dimensión PORIGEN.
pdestino_clases	Muestra las métricas de cada una de las 27 clases de comportamiento de los clusters significativos de la dimensión PDESTINO.

Cada uno de los 4 servlets muestra las 3 métricas (Tabla 5.3) que describen las características temporales de cada una de las 27 clases de comportamiento. Los parámetros de entrada y la descripción de la metodología utilizada para el cálculo de las métricas “popularidad”, “tamaño promedio”, “volatilidad” es exactamente la descrita para el Servlet “comportamiento” (ver sección b.3).

El servlet genera una tabla con los datos numéricos de cada una de las métricas y 3 gráficos de barras para cada una de las métricas. La Figura 5.22 muestra un ejemplo de la tabla generada.

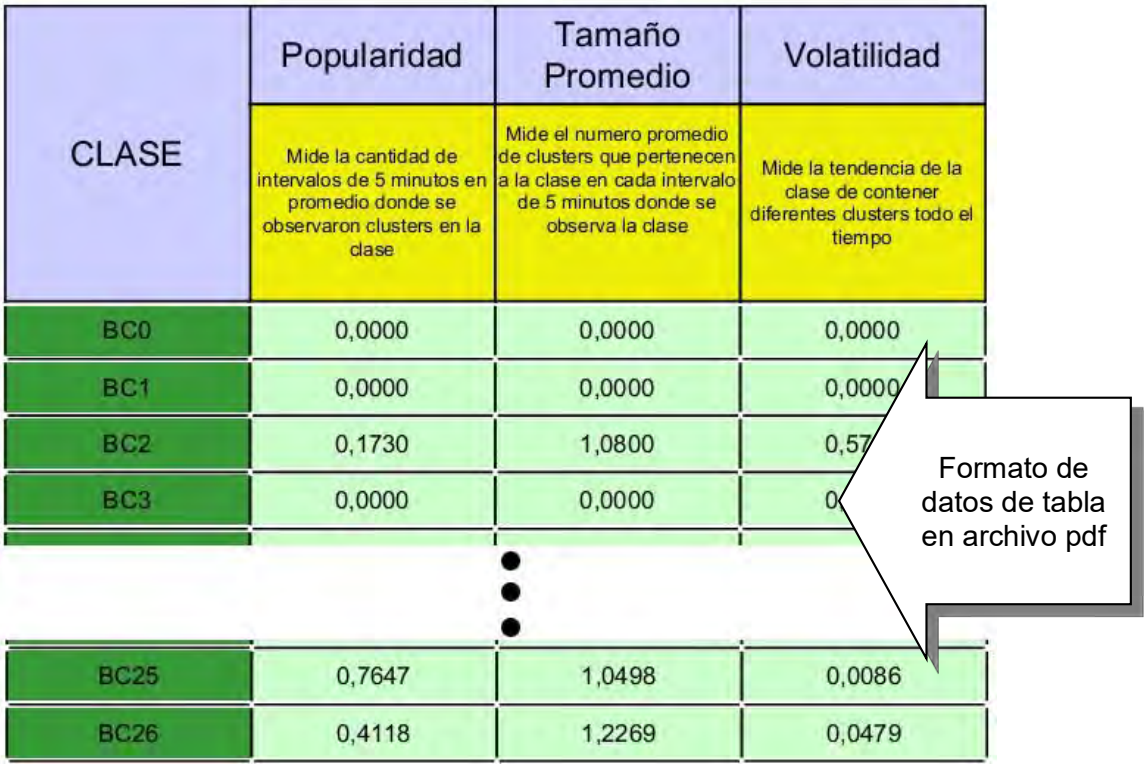


Figura 5.22 Ejemplo de una tabla de métricas de las 27 clases de comportamiento de la dimensión IPORIGEN

La Figura 5.23 muestra un ejemplo de la gráfica de barras de la métrica popularidad generada por el servlet. La forma de la gráfica es similar para las demás métricas.

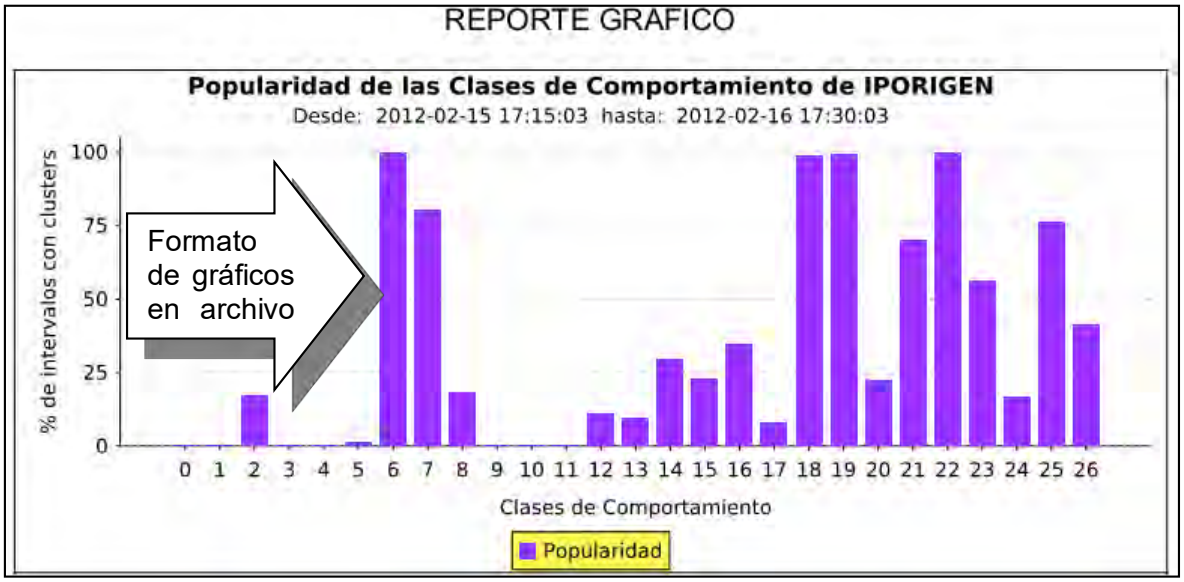


Figura 5.23 Ejemplo de un gráfico de barras de la métrica "popularidad"

b.7 Servlets que muestran los modelos estructurales de cada dimensión

Estos servlets muestran los resultados del proceso de modelamiento estructural basado en la teoría de la información en cada dimensión. La Tabla 5.21 muestra el nombre de cada uno de los servlets junto con una breve descripción.

Tabla 5.21 Servlets que muestran los modelos estructurales de cada dimensión

Nombre del servlet	Descripción del Servlet
iporigen_modelos	Muestra las clases de comportamiento, los modelos estructurales, cantidades totales y promedios de flujos, bytes y paquetes así como los coeficientes de variación de bytes y de paquetes de todos los clusters significativos de la dimensión IPORIGEN.
ipdestino_ modelos	Muestra las clases de comportamiento, los modelos estructurales, cantidades totales y promedios de flujos, bytes y paquetes así como los coeficientes de variación de bytes y de paquetes de todos los clusters significativos de la dimensión IPDESTINO.
porigen_ modelos	Muestra las clases de comportamiento, los modelos estructurales, cantidades totales y promedios de flujos, bytes y paquetes así como los coeficientes de variación de bytes y de paquetes de todos los clusters significativos de la dimensión PORIGEN.
pdestino_ modelos	Muestra las clases de comportamiento, los modelos estructurales, cantidades totales y promedios de flujos, bytes y paquetes así como los coeficientes de variación de bytes y de paquetes de todos los clusters significativos de la dimensión PDESTINO.

MODELOS ESTRUCTURALES IPORIGEN

CLUSTER	CLASE	MODELO ESTRUCTURAL	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_BT	CV_PKT
DIRECCION IP 1	6	porigen(80) -> ipdestino (DIRECCION IP 25) -> *	5	1296	259.2	7	1.4	1.69	0.57
DIRECCION IP 2	6	porigen(53) -> ipdestino (DIRECCION IP 26) -> *	5	810	162.0	10	2.0	0.63	0.63
DIRECCION IP 3	6	porigen(80) -> ipdestino (DIRECCION IP 27) -> *	5	155684	31136.8	119	23.8	0.82	0.74
DIRECCION IP 4	15	ipdestino (DIRECCION IP 28) -> (* , *)	6	7656	1276.0	44	7.3		
DIRECCION IP 5	15	ipdestino (DIRECCION IP 29) -> (* , *)	6	11079	1846.5	191			
DIRECCION IP 6	6	porigen(80) -> ipdestino (DIRECCION IP 30) -> *	7	26004	3714.86	45			
DIRECCION IP 7	8	porigen(80) -> (* , *)	10	440	44.0	10			
DIRECCION IP 8	22	(* , * , *)	11	117968	10724.36	2110			
DIRECCION IP 9	16	(* , * , *)	13	28337	2179.77	145	11		
DIRECCION IP 10	22	(* , * , *)	14	28246	2017.57	282	20.14		
DIRECCION IP 11	6	porigen(53) -> ipdestino (DIRECCION IP 31) -> *	22	4709	214.05	36	1.64	0.64	0.95
DIRECCION IP 12	6	porigen(53) -> (* , *)	22	11695	531.59	73	3.32	1.61	1.31
DIRECCION IP 13	14	(* , * , *)	22	65851	2993.23	657	29.86	2.91	3.64
DIRECCION IP 14	6	porigen(53) -> ipdestino (DIRECCION IP 32) -> *	24	2104	87.67	26	1.08	0.25	0.26
DIRECCION IP 15	6	porigen(80) -> ipdestino (DIRECCION IP 33) -> *	24	19849	827.04	125	5.21	0.03	0.11
DIRECCION IP 16	6	porigen(53) -> ipdestino (DIRECCION IP 34) -> *	25	2186	87.44	27	1.08	0.25	0.25
DIRECCION IP 17	6	porigen(861) -> ipdestino (DIRECCION IP 35) -> *	27	11148	412.89	155	5.74	0.18	0.17
DIRECCION IP 18	6	porigen(8081) -> ipdestino (DIRECCION IP 36) -> *	28	41159	1469.96	167	5.96	2.48	0.83
DIRECCION IP 19	19	(* , * , *)	29	3591	123.83	54	1.86	1.24	1.19
DIRECCION IP 20	18	pdestino(80) -> ipdestino (DIRECCION IP 37) -> *	32	28451	889.09	150	4.69	0.3	0.35
DIRECCION IP 21	22	(* , * , *)	49	55227	1127.08	281	5.73	1.34	0.71
DIRECCION IP 22	19	(* , * , *)	92	25002	271.76	519	5.64	1.99	1.23
DIRECCION IP 23	22	(* , * , *)	108	189913	1758.45	584	5.41	2.38	1.44
DIRECCION IP 24	6	porigen(53) -> (* , *)	150	59335	395.57	429	2.86	0.53	0.55

Captura de pantalla de tabla generada, contenida en archivo pdf

Figura 5.24 Ejemplo de una tabla generada por el servlet “iporigen_modelos”

La información mostrada por los servlets es presentada mediante una tabla que contiene los mismos campos de la tabla de la base de datos “iporigen_modelos” (modelos de la dimensión IPORIGEN) sin mostrar el campo “fecha”.

Los valores de la tabla consisten únicamente en la transferencia de información de la base de datos en la página web generada por el servlet. Para las demás dimensiones se consulta la tabla que corresponde a la dimensión.

La Figura 5.24 (página anterior) muestra parte de una tabla generada correspondiente a los modelos estructurales de la dimensión IPORIGEN.

5.5 Integración de los procesos

En esta sección se presenta un resumen donde se describe cómo se integran todos los elementos y procesos involucrados que forman parte de la aplicación para modelamiento y reportes.

En el servidor se almacenan los 3 proyectos escritos en JAVA llamados “Extracción”, “Análisis” y “Reporte” tal como se había mencionado al inicio de la sección 4.5. Cada uno de estos proyectos implementa un proceso. En la Tabla 5.22 (“Descripción de las funciones de cada proyecto de la aplicación para modelamiento y reportes”) se resume la función de cada uno de estos proyectos.

La ejecución del programa principal del proyecto “EXTRACCIÓN” inicia la recepción de paquetes del módulo de captura. Si este proyecto no es ejecutado, los paquetes que envía el módulo no son procesados.

El programa principal del proyecto “ANÁLISIS” es programado para que el sistema operativo (Linux distribución Ubuntu 10.04) lo ejecute cada 5 minutos. Para esto, se utiliza el programa “cron” el cual es propio del sistema operativo y se encarga de ejecutar programas en intervalos regulares de tiempo.

El programa realiza el proceso propiamente dicho solamente si encuentra archivos de texto generados por el proyecto “EXTRACCIÓN” y cuyo nombre sea igual a la fecha y hora actual. En caso contrario no realiza el proceso y el proyecto es ejecutado en los próximos 5 minutos.

El proyecto “REPORTE” es desplegado de manera fija en el servidor web Tomcat y se ejecuta el servlet respectivo cada vez que un cliente solicita la entrega de algún reporte. Para que un cliente pueda observar los reportes, necesita acceder a la siguiente dirección: http://IP_SERVIDOR_TOMCAT:8080/REPORTE desde un navegador web.

La Figura 5.25 ilustra los procesos y elementos que forman parte de la aplicación para modelamiento y reportes.

La misma figura muestra la forma de integración de estos elementos y procesos según se describe en la presente sección.

Tabla 5.22 Descripción de las funciones de cada proyecto de la aplicación para modelamiento y reportes

Nombre de proyecto	Funciones
EXTRACCIÓN	-Recibe los paquetes UDP provenientes del módulo de captura. -Interpreta el contenido de los paquetes. -Almacena el contenido de los paquetes en archivos de texto.
ANÁLISIS	-Revisa la existencia de archivos de texto recientes generados por el proceso de anterior. -Sintetiza los registros repetidos de flujos y genera una lista única con flujos diferentes. -Se almacena la respectiva información obtenida de la lista de flujos generada en la tabla “resumen_total” de la base de datos. -Realiza el proceso de extracción de flujos significativos de la dimensión IPORIGEN. -Almacena la información generada del proceso en cada uno de los campos de la tabla “resumen_iporigen” de la base de datos. -Realiza el proceso de clasificación de flujos significativos en clases de comportamiento de la dimensión IPORIGEN. -Almacena la información generada del proceso en cada uno de los campos de la tabla “iporigen_clases” de la base de datos. -Realiza el proceso de determinación de estados dominantes de la dimensión IPORIGEN. -Almacena la información generada del proceso en cada uno de los campos de la tabla “iporigen_modelos” de la base de datos. -Realiza el mismo procedimiento para las demás dimensiones.
REPORTE	-Recibe peticiones de clientes externos en el puerto TCP 8080 (puerto por defecto del servidor web Tomcat). -Presenta al cliente la página web principal para que seleccione un reporte en particular mediante botones. -Procesa la aplicación implementada en el respectivo servlet relacionado con el botón presionado por el cliente. -Presenta al cliente el reporte ya sea en formato HTML o PDF según el tipo de reporte seleccionado.

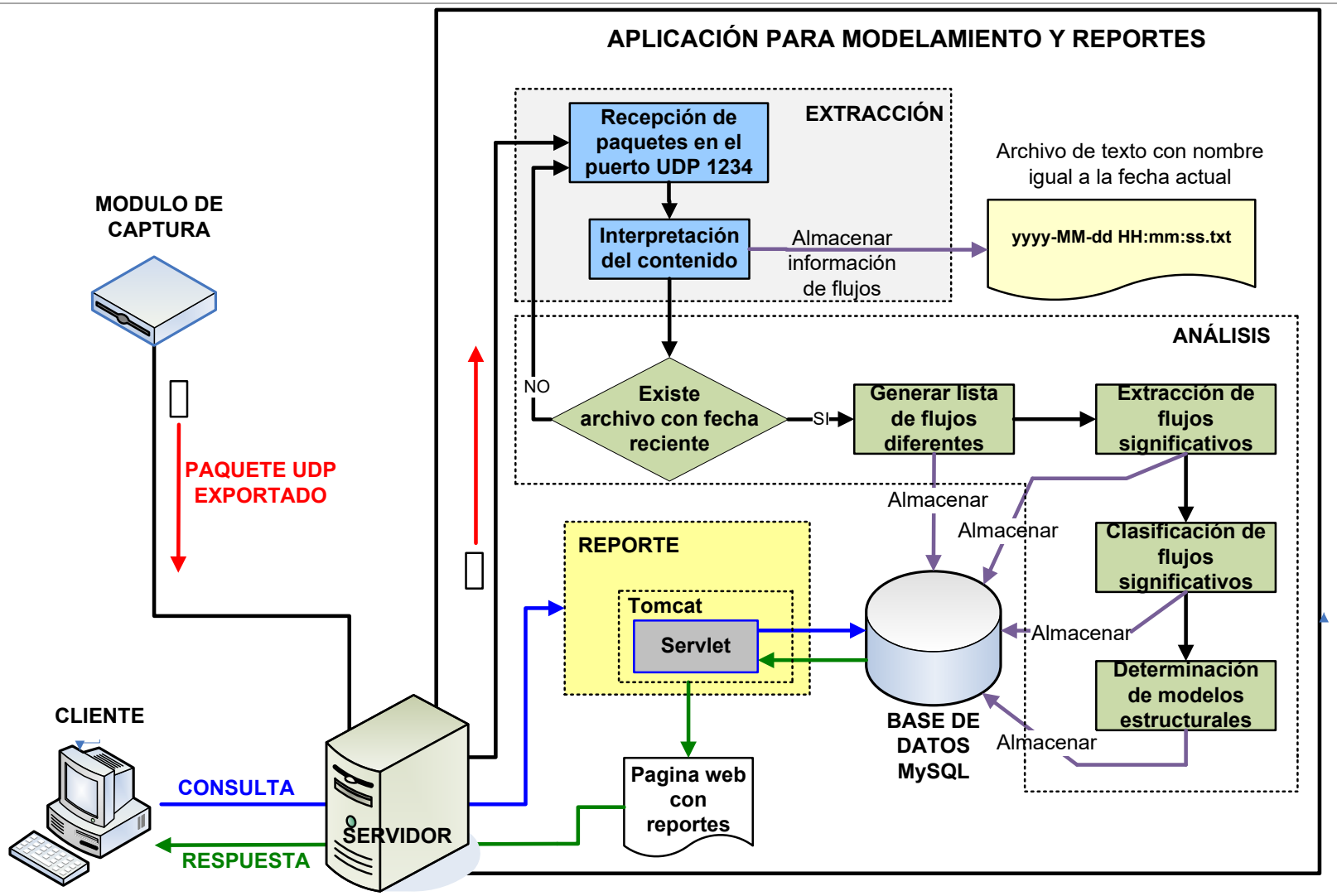


Figura 5.25 Procesos y elementos que forman parte de la aplicación para modelamiento y reportes

CAPÍTULO VI
ANÁLISIS Y PRESENTACIÓN DE RESULTADOS

En este capítulo se presentan e interpretan los resultados obtenidos de la prueba realizada en escenario real. Se realiza una breve descripción de la prueba, se muestran los indicadores de tráfico obtenidos durante el lapso de 4 días y 8 horas y se realiza un análisis e interpretación de tales indicadores. Finalmente, se hace un análisis del impacto que produce el sistema diseñado en la red de pruebas, y se hace la estimación de costos.

6.1 Descripción de la prueba

Las pruebas fueron realizadas en la red LAN del INICTEL-UNI correspondiente a la Red Académica Peruana RAAP la cual cuenta con acceso a internet de 2 Mbps, según el esquema mostrado en la Figura 6.1. Los equipos agregados a la red se muestran en color amarillo.

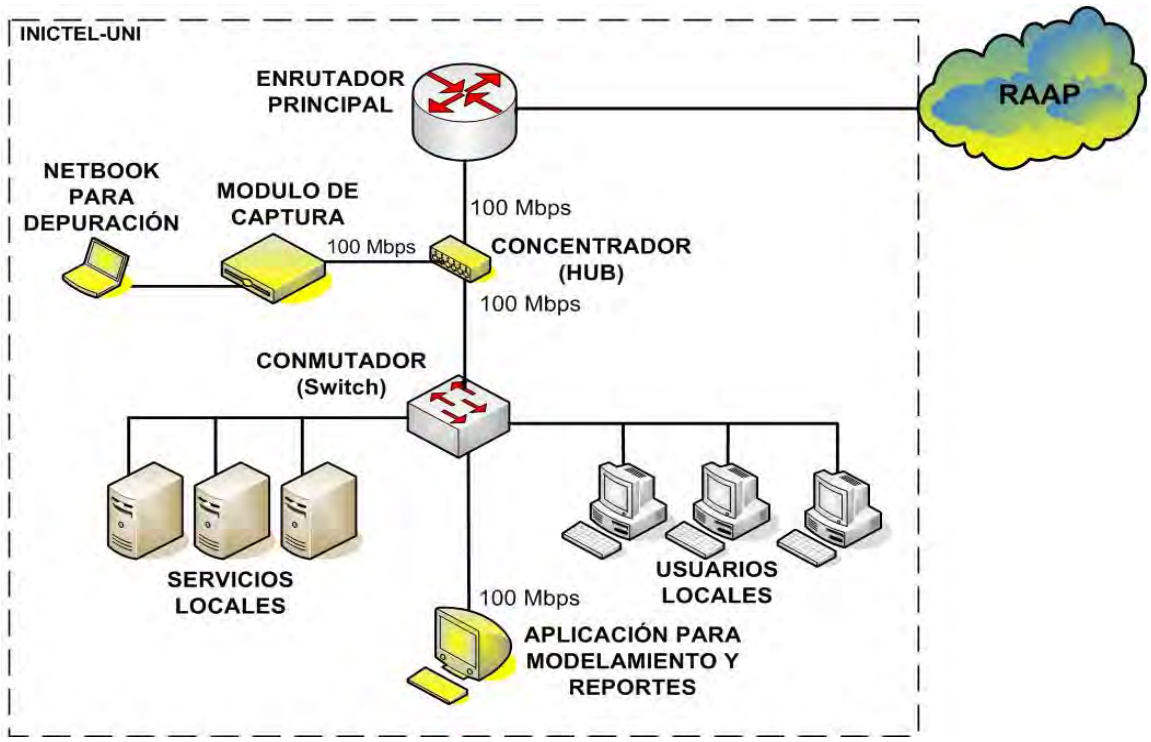


Figura 6.1 Esquema general del escenario de pruebas

Los equipos empleados fueron los siguientes:

- 1 hub marca 3com dual speed 10/100 Mbps.
- Una tarjeta de desarrollo modelo ML505 de Xilinx que implementa la funcionalidad del módulo de captura.

- Una netbook marca HP para la observación de errores del módulo de captura.
- Una Workstation marca HP que contiene la aplicación para modelamiento y reportes.

En la topología de la red del INICTEL-UNI, normalmente existe una conexión entre el enrutador principal y un switch. Sin embargo, para las pruebas se coloca el hub en medio de esta conexión utilizando dos de sus puertos. Un tercer puerto del hub es conectado hacia el módulo de captura. Los 3 puertos del hub operan a 100 Mbps.

El sistema se dejó en funcionamiento durante un lapso de 4 días y 8 horas. Pasado ese tiempo, el sistema se detuvo y se observaron los resultados mostrados en la siguiente sección.

6.2 Resultados de las pruebas de evaluación del sistema

En esta sección se muestran los indicadores de producción de servicios IP, las métricas de las clases de comportamiento y mediciones de tráfico y de retardo.

6.2.1 Indicadores de producción de servicios IP

En esta sección se muestran los resultados de las pruebas de evaluación correspondiente a servicios tanto internos como externos. Los servicios mostrados corresponden únicamente a los que se descubrió que tienen mayor demanda, los cuales son: web (puerto 80), SSH (puerto 22), DNS (puerto 53), SNMP (puerto 161), HTTPS (puerto 443) y SIP (puerto 5060). Las subsiguientes figuras contienen tablas con los modelos estructurales de la dimensión IPORIGEN BC6 de cada servicio mencionado. Estos modelos corresponden a aquellos servicios que cuentan con pocos clientes.

Servicios web locales.- La segunda tabla de la Figura 6.2 muestra las direcciones IP locales que cuentan con un servicio web y que fueron las más visitadas por clientes externos. La primera dirección IP local mostrada en la segunda tabla tiene alojado un servicio web que es el más visitado por 4 clientes potenciales cuyas direcciones IP se muestran en la primera tabla en la columna de modelos estructurales como “ipdestino”. La segunda tabla muestra que se detectó en total 12254 paquetes de flujos significativos. La última tabla muestra un resumen con las cifras de todos los servidores web locales. En total se detectaron 8 servidores que entregan servicios web a usuarios ubicados fuera de la institución.

Servicios SSH locales.- La segunda tabla de la Figura 6.3 muestra las direcciones IP locales que tienen activado el servicio SSH y cuyos servidores fueron los más accedidos por usuarios externos. La primera dirección IP local mostrada en la segunda tabla, corresponde a un servidor que tiene activado el servicio SSH y es el más accedido por 5 usuarios potenciales cuyas direcciones IP se muestran en la primera tabla en la columna de modelos estructurales como “ipdestino”. La segunda tabla muestra que se detectó en total 8487 paquetes de flujos significativos. La última tabla muestra un resumen con las

cifras de todos los servidores SSH locales. En total se detectaron 4 servidores que tienen activado el servicio SSH y que fueron accedidos por usuarios ubicados fuera de la institución.

Con esta información, se tiene identificado las direcciones IP de los usuarios que acceden remotamente a los servidores de la institución ya sea para tener el control total de algunos servicios locales, para realizar cambios de configuración o para utilizar los recursos computacionales de los servidores.

Servicios DNS locales.- La segunda tabla de la Figura 6.4 muestra una dirección IP local que cuenta con el servicio DNS y que fue visitada por un único cliente potencial externo cuya dirección IP se muestra en la primera tabla en la columna de modelos estructurales como “ipdestino”.

La segunda tabla muestra que se detectó en total sólo 5 paquetes de flujos significativos. La última tabla muestra un resumen con las cifras de todos los servidores DNS locales que para este caso el contenido es el mismo de la segunda tabla. De las cifras mostradas se deriva que los servicios DNS de la institución no son los más preferidos por usuarios externos. Esto sugiere la aplicación de mejoras para este servicio.

Nota: En los resultados, no se observan servicios SNMP locales.

Servicios HTTPS locales.- La segunda tabla de la Figura 6.5 muestra las direcciones IP locales que cuentan con un servicio HTTPS y que fueron las más visitadas por clientes externos. La primera dirección IP local mostrada en la segunda tabla tiene alojado un servicio web que es el más visitado por únicamente un cliente potencial cuya dirección IP se muestra en la primera tabla en la columna de modelos estructurales como “ipdestino”.

La segunda tabla muestra que se detectó en total 1578 paquetes de flujos significativos. La última tabla muestra un resumen con las cifras de todos los servidores HTTPS locales. En total se detectaron sólo 2 servidores que entregan servicios HTTPS a usuarios ubicados fuera de la institución. Estos resultados sugiere la aplicación de mejoras para este servicio.

Servicios SIP locales.- La segunda tabla de la Figura 6.6 muestra una dirección IP local que cuenta con el servicio SIP y que fue visitada por 3 usuarios potenciales externos cuyas direcciones IP se muestra en la primera tabla en la columna de modelos estructurales como “ipdestino”.

La segunda tabla muestra que se detectó en total 983 paquetes de flujos significativos. La última tabla muestra un resumen con las cifras de todos los servidores SIP locales que para este caso el contenido es el mismo de la segunda tabla. De las cifras mostradas se deriva que los servicios SIP de la institución no son los más preferidos por usuarios externos. Esto sugiere la aplicación de mejoras para este servicio.

MODELOS ESTRUCTURALES IPORIGEN BC6

PUERTO: 80 SERVICIO: WEB
SERVICIOS LOCALES
Desde: 2012-05-07 08:50:03 hasta: 2012-05-11 17:20:03

DETALLE:

IP DEL SERVIDOR	MODELO ESTRUCTURAL	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	Frecuencia
IP LOCAL 1	porigen(80) -> (* , *)	64	1795628	28056,69	1,28	2,15	1520	23,75	1,09	1,93	3
IP LOCAL 1	porigen(80) -> ipdestino(IP EXTERNA 1) -> *	4	502720	125680	1,73	1,73	351	87,75	1,65	1,65	1
IP LOCAL 1	porigen(80) -> ipdestino(IP EXTERNA 2) -> *	6	27760	4626,67	1,18	1,18	42	7	0,46	0,46	1
IP LOCAL 1	porigen(80) -> ipdestino(IP EXTERNA 3) -> *	13	292427	22494,38	0,77	0,77	239	18,38	0,57	0,57	1

RESUMEN ORDENADO SEGUN CANTIDAD TOTAL DE PAQUETES:

ORDEN	IP DEL SERVIDOR	NOMBRE DE DOMINIO	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS ESTRUCTURALES
1	IP LOCAL 1		420	14808271	35257,79	0,37	2,15	12254	29,18	0,27	1,93	23	21
2	IP LOCAL 2		97	1068906	11019,65	0,3	2,01	2362	24,35	0,56	1,7	2	2
3	IP LOCAL 3		12	5801	483,42	0,15	0,29	58	4,83	0,29	0,45	2	2
4	IP LOCAL 4		6	5532	922	0,09	0,09	30	5	0,31	0,31	1	1
5	IP LOCAL 5		6	5017	836,17	0,18	0,18	30	5	0,37	0,37	1	1
6	IP LOCAL 6		6	4866	811	0,14	0,14	30	5	0,45	0,45	1	1
7	IP LOCAL 7		6	4500	750	0,15	0,15	30	5	0,45	0,45	1	1
8	IP LOCAL 8		6	3552	592	0,2	0,2	30	5	0,45	0,45	1	1

RESUMEN:

CLASE DE COMPORTAMIENTO	SERVICIO	TOTAL SERVIDORES	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS
IPORIGEN BC6	WEB	8	559	15906445	28455,18	0,09	2,15	14824	26,52	0,27	1,93	32	30

Figura 6.2 Servicios web locales que cuentan con pocos clientes

MODELOS ESTRUCTURALES IPORIGEN BC6

PUERTO: 22 SERVICIO: SSH
SERVICIOS LOCALES
Desde: 2012-05-07 08:50:03 hasta: 2012-05-11 17:20:03

DETALLE:

IP DEL SERVIDOR	MODELO ESTRUCTURAL	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	Frecuencia
IP LOCAL 3	porigen(22) -> ipdestino(**(IP EXTERNA 5) -> ***	53	126212	2381,36	0,09	0,16	728	13,74	0,21	0,26	2
IP LOCAL 2	porigen(22) -> ipdestino(**(IP EXTERNA 5) -> ***	73	174248	2386,96	0,06	0,08	1018	13,95	0,18	0,18	3
IP LOCAL 1	porigen(22) -> ipdestino(**(IP EXTERNA 1) -> ***	10	26432	2643,2	0,14	0,14	145	14,5	0,08	0,08	1
IP LOCAL 1	porigen(22) -> ipdestino(**(IP EXTERNA 2) -> ***	29	67676	2333,66	0,11	0,11	410	14,14	0,19	0,19	1
IP LOCAL 1	porigen(22) -> ipdestino(**(IP EXTERNA 3) -> ***	17	35040	2061,18	0,34	0,34	214	12,59	0,34	0,34	1
IP LOCAL 1	porigen(22) -> ipdestino **(IP EXTERNA 4) -> ***	330	747816	2266,11	0,07	0,19	4521	13,7	0,17	0,23	7
IP LOCAL 1	porigen(22) -> ipdestino(**(IP EXTERNA 5) -> ***	233	529228	2271,36	0,06	0,21	3197	13,72	0,16	0,28	6
IP LOCAL 4	porigen(22) -> ipdestino(**(IP EXTERNA 5) -> ***	12	26553	2212,75	0,29	0,29	157	13,08	0,3	0,3	1

RESUMEN ORDENADO SEGUN CANTIDAD TOTAL DE PAQUETES:

ORDEN	IP DEL SERVIDOR	NOMBRE DE DOMINIO	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS ESTRUCTURALES
1	IP LOCAL 1		619	1406192	2271,72	0,06	0,34	8487	13,71	0,08	0,34	16	5
2	IP LOCAL 2		73	174248	2386,96	0,06	0,08	1018	13,95	0,18	0,18	3	1
3	IP LOCAL 3		53	126212	2381,36	0,09	0,16	728	13,74	0,21	0,26	2	1
4	IP LOCAL 4		12	26553	2212,75	0,29	0,29	157	13,08	0,3	0,3	1	1

RESUMEN:

CLASE DE COMPORTAMIENTO	SERVICIO	TOTAL SERVIDORES	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS
IPORIGEN BC6	SSH	4	757	1733205	2289,57	0,06	0,34	10390	13,73	0,08	0,34	22	8

Figura 6.3 Servicios SSH locales que cuentan con pocos clientes

MODELOS ESTRUCTURALES IPORIGEN BC6

PUERTO: 53 SERVICIO: DNS
SERVICIOS LOCALES
Desde: 2012-05-07 08:50:03 hasta: 2012-05-11 17:20:03

DETALLE:

IP DEL SERVIDOR	MODELO ESTRUCTURAL	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	Frecuencia
IP LOCAL 1	porigen(53) -> (IP EXTERNA 1) -> *	5	273	54,6	0,4	0,4	5	1	0	0	1

RESUMEN ORDENADO SEGUN CANTIDAD TOTAL DE PAQUETES:

ORDEN	IP DEL SERVIDOR	NOMBRE DE DOMINIO	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS ESTRUCTURALES
1	IP LOCAL 1		5	273	54,6	0,4	0,4	5	1	0	0	1	1

RESUMEN:

CLASE DE COMPORTAMIENTO	SERVICIO	TOTAL SERVIDORES	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS
IPORIGEN BC6	DNS	1	5	273	54,6	0,4	0,4	5	1	0	0	1	1

Figura 6.4 Servicios DNS locales que cuentan con pocos clientes

MODELOS ESTRUCTURALES IPORIGEN BC6

PUERTO: 443 SERVICIO: HTTPS
SERVICIOS LOCALES
Desde: 2012-05-07 08:50:03 hasta: 2012-05-11 17:20:03

DETALLE:

IP DEL SERVIDOR	MODELO ESTRUCTURAL	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	Frecuencia
IP LOCAL 2	porigen(443) -> ipdestino (IP EXTERNA 1) -> *	63	307159	4875,54	0,22	2,48	553	8,78	0,1	1,66	3
IP LOCAL 1	porigen(443) -> ipdestino (IP EXTERNA 1) -> *	232	654585	2821,49	0,12	0,4	1578	6,8	0,07	0,12	9

RESUMEN ORDENADO SEGUN CANTIDAD TOTAL DE PAQUETES:

ORDEN	IP DEL SERVIDOR	NOMBRE DE DOMINIO	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS ESTRUCTURALES
1	IP LOCAL 1		232	654585	2821,49	0,12	0,4	1578	6,8	0,07	0,12	9	1
2	IP LOCAL 2		63	307159	4875,54	0,22	2,48	553	8,78	0,1	1,66	3	1

RESUMEN:

CLASE DE COMPORTAMIENTO	SERVICIO	TOTAL SERVIDORES	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS
IPORIGEN BC6	HTTPS	2	295	961744	3260,15	0,12	2,48	2131	7,22	0,07	1,66	12	2

Figura 6.5 Servicios HTTPS locales que cuentan con pocos clientes

MODELOS ESTRUCTURALES IPORIGEN BC6

PUERTO: 5060 SERVICIO: SIP
SERVICIOS LOCALES
Desde: 2012-05-07 08:50:03 hasta: 2012-05-11 17:20:03

DETALLE:

IP DEL SERVIDOR	MODELO ESTRUCTURAL	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	Frecuencia
IP LOCAL 1	porigen(5060) -> ipdestino **(IP EXTERNA 1) -> ***	4	85993	21498,25	1,34	1,34	143	35,75	1,34	1,34	1
IP LOCAL 1	porigen(5060) -> ipdestino **(IP EXTERNA 1) -> ***	12	420868	35072,33	0,93	0,97	689	57,42	0,93	0,96	6
IP LOCAL 1	porigen(5060) -> ipdestino **(IP EXTERNA 3) -> ***	3	91173	30391	0,97	0,97	151	50,33	0,97	0,97	1

RESUMEN ORDENADO SEGUN CANTIDAD TOTAL DE PAQUETES:

ORDEN	IP DEL SERVIDOR	NOMBRE DE DOMINIO	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS ESTRUCTURALES
1	IP LOCAL 1		19	598034	31475,47	0,93	1,34	983	51,74	0,93	1,34	8	3

RESUMEN:

CLASE DE COMPORTAMIENTO	SERVICIO	TOTAL SERVIDORES	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS
IPORIGEN BC6	SIP	1	19	598034	31475,47	0,93	1,34	983	51,74	0,93	1,34	8	3

Figura 6.6 Servicios SIP locales que cuentan con pocos clientes

En las Figuras 6.7 y 6.8 se ilustran algunas cifras contenidas en las tablas de resumen de cada servicio local. De los gráficos, se deduce que el número de servidores y la cantidad total de paquetes los servicios web y SSH locales son los más utilizados por usuarios externos. Los demás servicios requieren ser mejorados para elevar las cifras de sus indicadores.

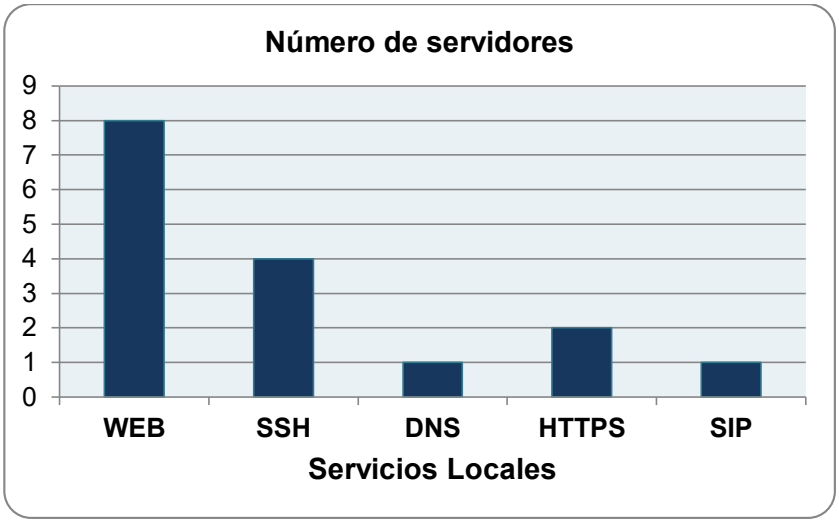


Figura 6.7 Número de servidores de servicios locales

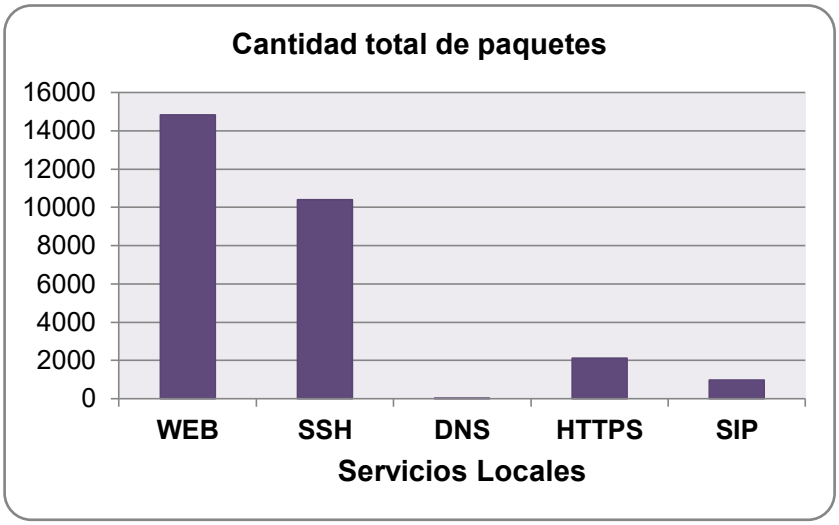


Figura 6.8 Cantidad total de paquetes de servicios locales

Las Figuras 6.9 y 6.10 muestran los valores máximos y mínimos de los coeficientes de variación de bytes y de paquetes respectivamente. Los resultados indican que durante la entrega de servicios HTTPS, WEB y SIP se presentan cantidades de paquetes diferentes y cuyas cantidades de bytes (tamaño de paquetes) difieren la mayor parte del tiempo. En contraste, la entrega de los servicios SSH y DNS se presentan con cantidades de bytes y de paquetes no muy variables. Esto indica que estos servicios entregan una cantidad de información constante durante el tiempo de entrega de dichos servicios. Estos servicios tienden a formar parte de rutinas diarias de los usuarios locales.

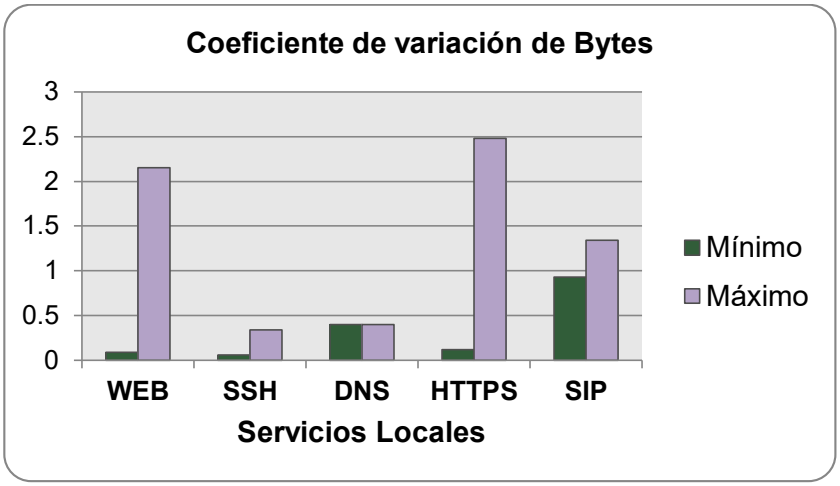


Figura 6.9 Coeficiente de variación de Bytes de servicios locales

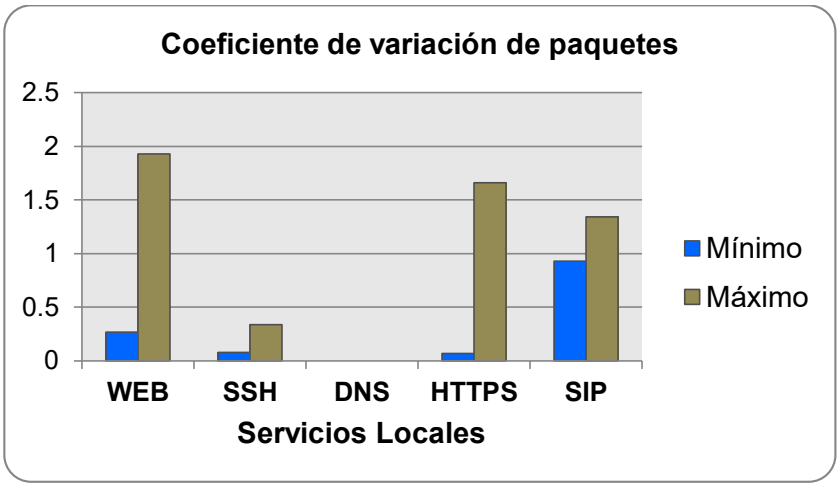


Figura 6.10 Coeficiente de variación de paquetes de servicios locales

Servicios web remotos.- La primera tabla de la Figura 6.11 muestra solamente las 10 principales direcciones IP remotas que cuentan con un servicio web y que fueron las más visitadas por clientes dentro de la institución. La primera dirección IP cuyo nombre de dominio es “proxy_server.dbd.puc-rio.br” tiene alojado un servicio web que es visitado frecuentemente por un cliente potencial dentro de la institución cuya dirección IP se muestra en la primera tabla, segunda columna como “ipdestino”. La segunda tabla muestra que se detectó en total 616177 paquetes de flujos significativos que tienen como origen dicho servidor y como destino los clientes locales. La última tabla muestra un resumen con las cifras de todos los servidores web remotos. En total se detectaron 1237 servidores que entregan servicios web a usuarios ubicados dentro de la institución.

Nota: En la primera tabla de la Figura 6.11, se muestran únicamente los modelos estructurales que corresponden al servidor web más visitado.

Servicios DNS remotos.- La primera tabla de la Figura 6.12 muestra solamente las 10 principales direcciones IP remotas que cuentan con un servicio DNS y que fueron las más visitadas por clientes dentro de la institución. La primera dirección IP cuyo nombre de

dominio es “ns3.unired.edu.pe” (DNS de telefónica) tiene alojado un servicio DNS que es visitado frecuentemente por diversos clientes potenciales dentro de la institución según se muestra en la primera tabla, segunda columna como “(*,*)” lo cual indica diferentes direcciones IP destino locales con diferentes números de puerto. La segunda tabla muestra que se detectó en total 661882 paquetes de flujos significativos que tienen como origen dicho servidor y como destino los clientes locales. La última tabla muestra un resumen con las cifras de todos los servidores DNS remotos. En total se detectaron 52 servidores que entregan servicios DNS a usuarios ubicados dentro de la institución.

Nota: En la primera tabla de la Figura 6.12, se muestran únicamente los modelos estructurales que corresponden al servidor DNS más accedido.

Servicios SNMP remotos.- La primera tabla de la Figura 6.13 muestra las direcciones IP remotas que cuentan con un servicio SNMP (llamado comúnmente agente SNMP) y que fueron los más accedidos por usuarios dentro de la institución. La primera dirección IP tiene activado el agente SNMP el cual es accedido frecuentemente por dos clientes potenciales dentro de la institución cuyas direcciones IP se muestran en la primera tabla, segunda columna como “ipdestino”. La segunda tabla muestra que se detectó en total 41076 paquetes de flujos significativos que tienen como origen dicho servidor y como destino los clientes locales. La última tabla muestra un resumen con las cifras de todos los servicios SNMP remotos. En total se detectaron 10 servidores externos que están funcionando como agentes SNMP y que están siendo monitoreados por dos servidores (clientes) ubicados dentro de la institución.

Nota: En la primera tabla de la Figura 6.13, se muestran únicamente los modelos estructurales que corresponden al servidor SNMP más accedido.

Servicios HTTPS remotos.- La primera tabla de la Figura 6.14 muestra solamente las 10 principales direcciones IP remotas que cuentan con un servicio HTTPS y que fueron las más visitadas por clientes dentro de la institución. La primera dirección IP cuyo nombre de dominio es “vb-in-f139.1e100.net” (servidor de Google) tiene alojado un servicio HTTPS que es visitado frecuentemente por diversos clientes potenciales dentro de la institución especialmente por aquellos cuyas direcciones IP se muestran en la primera tabla, segunda columna como “ipdestino”. La segunda tabla muestra que se detectó en total 124377 paquetes de flujos significativos que tienen como origen dicho servidor y como destino los clientes locales. La última tabla muestra un resumen con las cifras de todos los servidores HTTPS remotos. En total se detectaron 277 servidores que entregan servicios HTTPS a usuarios ubicados dentro de la institución.

Nota: En la primera tabla de la Figura 6.14, se muestran únicamente los modelos estructurales que corresponden al servidor web más visitado.

Nota: En los resultados, no se observan servicios SSH ni SIP remotos.

MODELOS ESTRUCTURALES IPORIGEN BC6

PUERTO: 80 SERVICIO: WEB
SERVICIOS REMOTOS
Desde: 2012-05-07 08:50:03 hasta: 2012-05-11 17:20:03

DETALLE:

IP EXTERNA 11	porigen(80) -> ipdestino **(IP LOCAL 1) -> ***	3	45582	15194	0,72	0,72	62	20,67	0,51	0,51	1
IP EXTERNA 12	porigen(80) -> ipdestino **(IP LOCAL 2) -> ***	30	540419	18013,97	0,95	0,95	652	21,73	0,75	0,75	1
IP EXTERNA 1	porigen(80) -> ipdestino **(IP LOCAL 3) -> ***	17223	570860758	33145,26	0,27	8,23	616177	35,78	0,14	8,09	367
IP EXTERNA 13	porigen(80) -> ipdestino **(IP LOCAL 4) -> ***	23	1015577	44155,52	1,03	1,28	980	42,61	0,94	1,04	3

RESUMEN ORDENADO SEGUN CANTIDAD TOTAL DE PAQUETES:

ORDEN	IP DEL SERVIDOR	NOMBRE DE DOMINIO	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS ESTRUCTURALES
1	**IP EXTERNA 1**	proxy_server.dbd.puc-rio.br. www3.dbd.puc-rio.br.	17223	570860758	33145,26	0,27	8,23	616177	35,78	0,14	8,09	367	1
2	**IP EXTERNA 2**		148	53594667	362126,13	0,91	3,3	39904	269,62	0,89	3,21	10	1
3	**IP EXTERNA 3**	www.crpradio.com.pe.	5098	10164335	1993,79	0	2,31	26528	5,2	0	1,66	910	2
4	**IP EXTERNA 4**	www.ginga.org.pe.	1384	17105181	12359,23	0,09	3,79	23274	16,82	0,08	2,11	71	3
5	**IP EXTERNA 5**	gator2.hostgator.com.	15	31945326	2129688,4	1,83	2,17	22602	1506,8	1,83	2,14	2	1
6	**IP EXTERNA 6**	ec2-174-129-1-195.compute-1.amazonaws.com.	73	25072293	343456,07	0,58	2,61	18343	251,27	0,56	2,12	11	1
7	**IP EXTERNA 7**	ubuntu.c3sl.ufpr.br.	3	21474017	7158005,67	1,19	1,19	15137	5045,67	1,19	1,19	1	1
8	**IP EXTERNA 8**		13	20325199	1563476,85	1,39	1,56	14374	1105,69	1,39	1,56	2	1
9	**IP EXTERNA 9**	server.streaminginternacional.info.	42	19794588	471299,71	0,4	1,93	14143	336,74	0,4	1,91	4	1
10	**IP EXTERNA 10**		302	6377033	21116	0,62	2,21	11845	39,22	0,91	2,79	24	1

RESUMEN:

CLASE DE COMPORTAMIENTO	SERVICIO	TOTAL SERVIDORES	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS
IPORIGEN BC6	WEB	1237	50907	1366034136	26833,92	0	9,7	1363070	26,78	0	8,16	4057	1381

Figura 6.11 Servicios web remotos que cuentan con pocos clientes

MODELOS ESTRUCTURALES IPORIGEN BC6

PUERTO: 53 SERVICIO: DNS
SERVICIOS REMOTOS
Desde: 2012-05-07 08:50:03 hasta: 2012-05-11 17:20:03

DETALLE:

IP EXTERNA 11	porigen(53) -> ipdestino **(IP LOCAL 1) -> ***	4	1445	361,25	0,55	0,55	5	1,25	0,35	0,35	1
IP EXTERNA 1	porigen(53) -> (* , *)	273609	80448535	294,03	0,29	1,33	595122	2,18	0,28	1,13	1230
IP EXTERNA 12	porigen(53) -> (* , *)	280	52403	187,15	0,33	0,92	404	1,44	0,26	0,91	15
IP EXTERNA 13	porigen(53) -> ipdestino **(IP LOCAL 2) -> ***	44	8814	200,32	0,55	0,79	66	1,5	0,49	0,69	3

RESUMEN ORDENADO SEGUN CANTIDAD TOTAL DE PAQUETES:

ORDEN	IP DEL SERVIDOR	NOMBRE DE DOMINIO	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS ESTRUCTURALES
1	**IP EXTERNA 1**	ns3.unired.net.pe.	273609	80448535	294,03	0,29	1,33	595122	2,18	0,28	1,13	1230	1
2	**IP EXTERNA 2**	google-public-dns-a.google.com.	27611	5720895	207,2	0	2,03	41317	1,5	0	1,67	492	2
3	**IP EXTERNA 3**	cachewas.tdp.net.pe.	10209	1907992	186,89	0	1,89	13382	1,31	0	1,58	342	5
4	**IP EXTERNA 4**	resolver1.opendns.com.	4790	561877	117,3	0	0,73	6937	1,45	0	0,73	746	2
5	**IP EXTERNA 5**	cachesis.tdp.net.pe.	1169	249859	213,74	0	1,64	1825	1,56	0	1,46	81	4
6	**IP EXTERNA 6**	ns4.unired.net.pe.	501	82523	164,72	0	0,99	707	1,41	0	0,91	40	3
7	**IP EXTERNA 7**	ns0.wikimedia.org.	434	73153	168,56	0,34	0,41	535	1,23	0,34	0,4	3	2
8	**IP EXTERNA 8**	ns1.wikimedia.org.	414	70605	170,54	0,31	0,42	518	1,25	0,34	0,42	3	2
9	**IP EXTERNA 9**	ns2.wikimedia.org.	405	66933	165,27	0,3	0,38	490	1,21	0,29	0,38	3	2
10	**IP EXTERNA 10**	ns3.google.com.	74	14782	199,76	0,14	0,84	115	1,55	0	0,61	9	4

RESUMEN:

CLASE DE COMPORTAMIENTO	SERVICIO	TOTAL SERVIDORES	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS
IPORIGEN BC6	DNS	52	319848	89635158	280,24	0	2,03	661882	2,07	0	1,67	3036	91

Figura 6.12 Servicios DNS remotos que cuentan con pocos clientes

MODELOS ESTRUCTURALES IPORIGEN BC6

PUERTO: 161 SERVICIO: SNMP
SERVICIOS REMOTOS
Desde: 2012-05-07 08:50:03 hasta: 2012-05-11 17:20:03

DETALLE:

IP EXTERNA 2	porigen(161) -> (* , *)	102	63159	619,21	0,17	2,44	789	7,74	0,29	2,47	8
IP EXTERNA 2	porigen(161) -> ipdestino(IP LOCAL 1) -> *	3	1791	597	0,97	0,97	23	7,67	0,97	0,97	1
IP EXTERNA 1	porigen(161) -> ipdestino(IP LOCAL 2) -> *	321	413743	1288,92	0,09	0,63	2496	7,78	0,09	0,65	57
IP EXTERNA 1	porigen(161) -> ipdestino(IP LOCAL 3) -> *	4486	6098115	1359,37	0,02	0,8	36793	8,2	0,03	0,82	842
IP EXTERNA 3	porigen(161) -> ipdestino(IP LOCAL 4) -> *	20	11361	568,05	0,03	0,53	70	3,5	0	0,6	7
IP EXTERNA 3	porigen(161) -> ipdestino(IP LOCAL 5) -> *	89	43682	490,81	0,15	0,79	269	3,02	0,2	0,8	27

RESUMEN ORDENADO SEGUN CANTIDAD TOTAL DE PAQUETES:

ORDEN	IP DEL SERVIDOR	NOMBRE DE DOMINIO	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS ESTRUCTURALES
1	IP EXTERNA 1		4807	6511858	1354,66	0,02	0,8	39289	8,17	0,03	0,82	899	2
2	IP EXTERNA 2		105	64950	618,57	0,17	2,44	812	7,73	0,29	2,47	9	2
3	IP EXTERNA 3		109	55043	504,98	0,03	0,79	339	3,11	0	0,8	34	2
4	IP EXTERNA 4		73	23229	318,21	0,12	0,64	279	3,82	0,09	0,67	14	2
5	IP EXTERNA 5		65	22551	346,94	0,25	0,59	244	3,75	0,16	0,54	12	2
6	IP EXTERNA 6		14	5621	401,5	0,16	0,61	54	3,86	0,11	0,37	6	1
7	IP EXTERNA 7		21	3227	153,67	0,26	0,64	35	1,67	0	0,6	7	1
8	IP EXTERNA 8		3	1438	479,33	0,36	0,36	12	4	0,41	0,41	1	1
9	IP EXTERNA 9		2	724	362	0,79	0,79	6	3	0,67	0,67	1	1
10	IP EXTERNA 10		2	694	347	0,31	0,31	6	3	0	0	1	1

RESUMEN:

CLASE DE COMPORTAMIENTO	SERVICIO	TOTAL SERVIDORES	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS
IPORIGEN BC6	SNMP	10	5201	6689335	1286,16	0,02	2,44	41076	7,9	0	2,47	984	15

Figura 6.13 Servicios SNMP remotos que cuentan con pocos clientes

MODELOS ESTRUCTURALES IPORIGEN BC6

PUERTO: 443 SERVICIO: HTTPS
SERVICIOS REMOTOS
Desde: 2012-05-07 08:50:03 hasta: 2012-05-11 17:20:03

DETALLE:

IP EXTERNA 5	porigen(443) -> ipdestino**(IP LOCAL 1) -> ***	43	405628	9433,21	0	1,31	491	11,42	0	1,12	7
IP EXTERNA 1	porigen(443) -> (* , *)	9	13769	1529,89	1,35	1,35	44	4,89	0,73	0,73	1
IP EXTERNA 1	porigen(443) -> ipdestino**(IP LOCAL 2) -> ***	8	641764	80220,5	0,92	0,92	601	75,12	0,84	0,84	1
IP EXTERNA 1	porigen(443) -> ipdestino**(IP LOCAL 3) -> ***	29	7698385	265461,55	2,45	3,28	6333	218,38	2,43	3,2	3
IP EXTERNA 12	porigen(443) -> (* , *)	9	28571	3174,56	1,73	1,73	81	9	0,97	0,97	1
IP EXTERNA 13	porigen(443) -> ipdestino**(IP LOCAL 4) -> ***	4	2035	508,75	0,18	0,18	25	6,25	0,17	0,17	1

RESUMEN ORDENADO SEGUN CANTIDAD TOTAL DE PAQUETES:

ORDEN	IP DEL SERVIDOR	NOMBRE DE DOMINIO	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS ESTRUCTURALES
1	**IP EXTERNA 1**	vb-in-f139.1e100.net.	46	8353918	181606,91	0,92	3,28	6978	151,7	0,73	3,2	5	3
2	**IP EXTERNA 2**	vb-in-f18.1e100.net.	304	3060445	10067,25	0,47	2,68	6408	21,08	0,15	2,34	41	4
3	**IP EXTERNA 3**	ec2-107-20-134-222.compute-1.amazonaws.com.	8	7731457	966432,12	1,5	1,5	5183	647,88	1,5	1,5	1	1
4	**IP EXTERNA 4**	vb-in-f19.1e100.net.	180	2473679	13742,66	0,33	1,78	4245	23,58	0,2	1,67	30	3
5	**IP EXTERNA 5**	vb-in-f132.1e100.net.	548	2525551	4608,67	0	2,47	4178	7,62	0	2,12	53	3
6	**IP EXTERNA 6**	vb-in-f17.1e100.net.	216	1967243	9107,61	0,31	2,7	4075	18,87	0,2	2,13	32	4
7	**IP EXTERNA 8**	lga15s35-in-f22.1e100.net.	182	2322224	12759,47	0,38	2,96	3961	21,76	0,22	2,55	28	1
8	**IP EXTERNA 9**	vb-in-f83.1e100.net.	211	1707905	8094,34	0,18	2,31	3760	17,82	0,15	1,55	38	4
9	**IP EXTERNA 10**	lga15s35-in-f21.1e100.net.	174	1079257	6202,63	0,39	3,14	3043	17,49	0,27	1,72	42	1
10	**IP EXTERNA 11**	h79-138-25-25.static.se.alltele.net.	98	2675083	27296,77	1,06	4,14	2568	26,2	0,87	2,26	5	1

RESUMEN:

CLASE DE COMPORTAMIENTO	SERVICIO	TOTAL SERVIDORES	TOTAL FLUJOS	TOTAL BYTES	PROMEDIO BYTES	CV_BT minimo	CV_BT maximo	TOTAL PAQUETES	PROMEDIO PAQUETES	CV_PKT minimo	CV_PKT maximo	FRECUENCIA	CANTIDAD DE MODELOS
IPORIGEN BC6	HTTPS	277	6057	98650978	16287,1	0	9,86	124377	20,53	0	4,04	845	321

Figura 6.14 Servicios HTTPS remotos que cuentan con pocos clientes

En las Figuras 6.15 y 6.16 se ilustran algunas cifras contenidas en las tablas de resumen de cada servicio remoto. Del primer gráfico, se deduce que los servicios web y HTTPS externos son los más frecuentados por los usuarios de la institución. Asimismo, se detectó mayor cantidad de paquetes por parte de los servicios web y DNS externos. Los demás servicios son utilizados en menor cantidad por usuarios de la institución. Los detalles sobre las direcciones IP y nombres de dominio de los servidores web, DNS, SNMP y HTTPS más frecuentados, se muestran desde la Figura 6.11 hasta la 6.14.

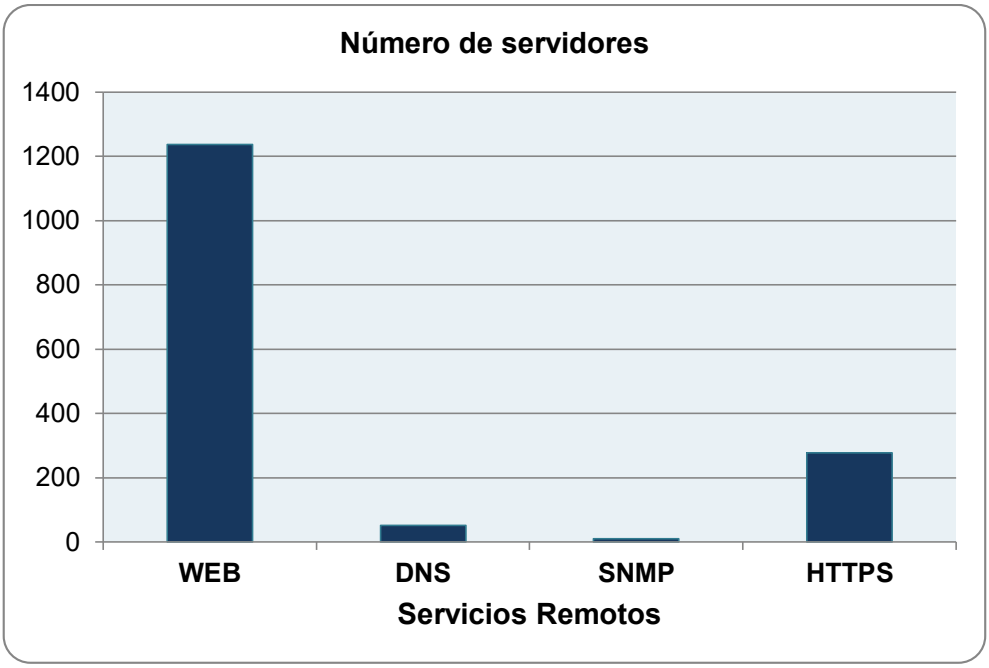


Figura 6.15 Número de servidores de servicios remotos

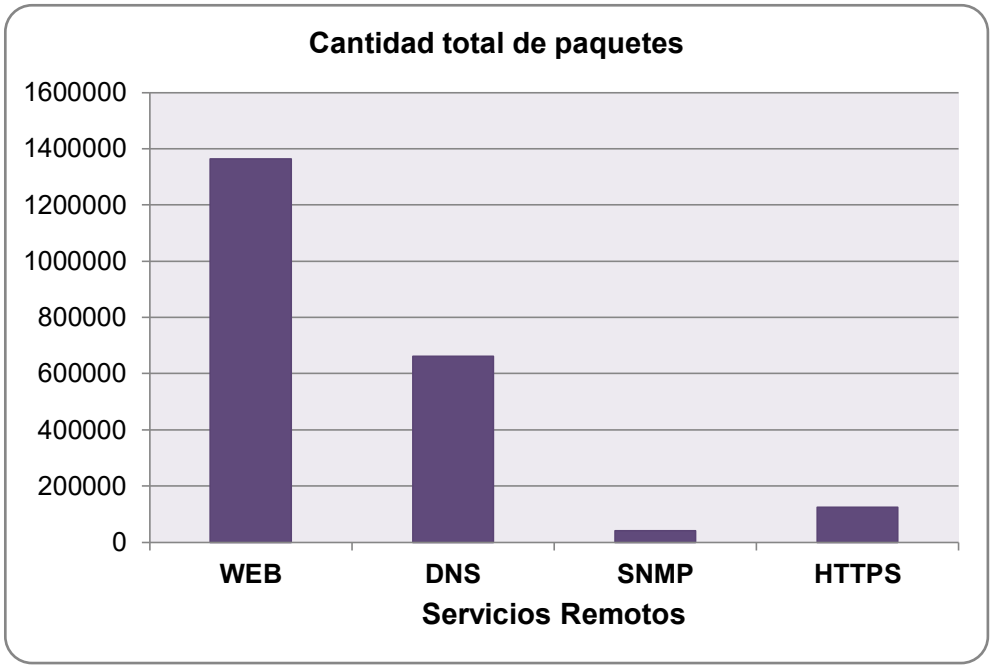


Figura 6.16 Cantidad total de paquetes de servicios remotos

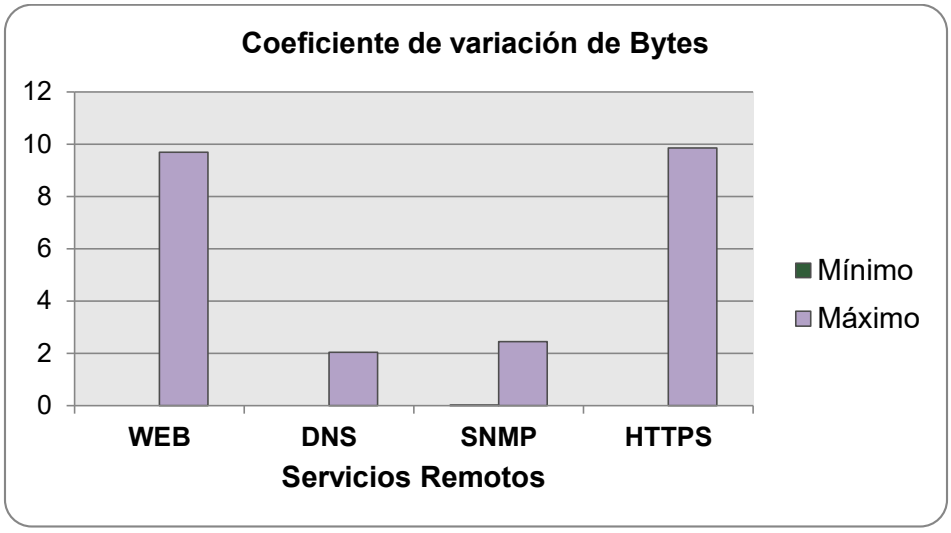


Figura 6.17 Coeficiente de variación de Bytes y de paquetes de servicios remotos

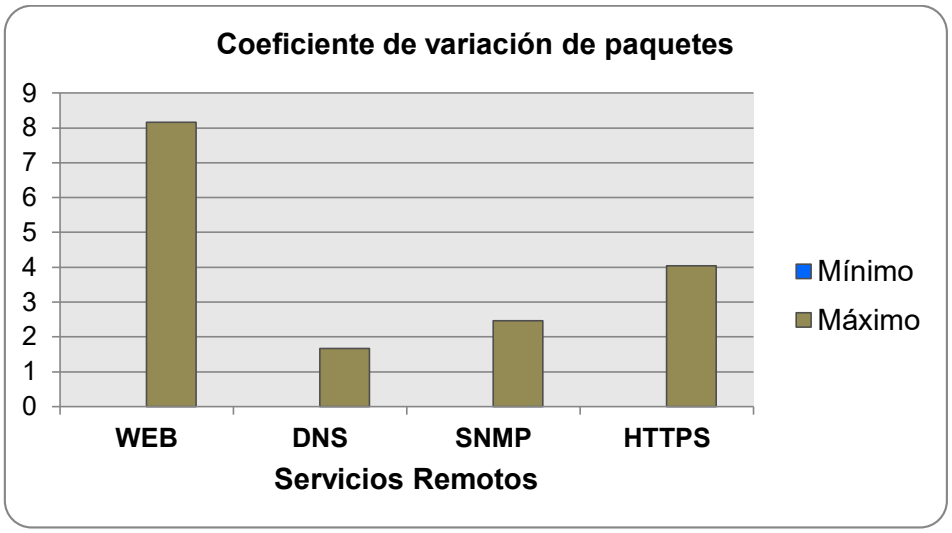


Figura 6.18 Coeficiente de variación de Bytes y de paquetes de servicios remotos

Las Figuras 6.17 y 6.18 muestran los valores máximos y mínimos de los coeficientes de variación de bytes y de paquetes. Los resultados indican que durante la recepción de servicios WEB y HTTPS se presentan cantidades de paquetes diferentes y cuyas cantidades de bytes (tamaño de paquetes) difieren la mayor parte del tiempo. En contraste, la recepción de los servicios DNS y SNMP se presentan con cantidades de bytes y de paquetes no muy variables. Esto indica que estos servicios entregan información cuya cantidad tiende a ser constante durante el tiempo de recepción de dichos servicios. Son servicios locales que tienden a formar parte de rutinas diarias de los usuarios externos que lo reciben.

La Figura 6.19 muestra un gráfico comparativo de la cantidad total de paquetes de servicios web locales y remotos. Se observa una marcada diferencia de ambas cifras ya que los servicios web remotos representan el 98.92 % del tráfico web significativo, mientras que los servicios web locales representan únicamente el 1.08 % de dicho tráfico.

Estos últimos resultados revelan que la institución consume mucho más de lo que produce en cuanto a servicios web. Similar situación se observa en los demás servicios.

La Figura 6.20 muestra un gráfico comparativo de la cantidad total de paquetes de todos los servicios locales y remotos. Se observa una marcada diferencia de ambas cifras ya que todos servicios remotos representan el 91.63 % del tráfico web significativo, mientras que los servicios web locales representan únicamente el 8.37 % de dicho tráfico.

En La Figura 6.21 se muestra un gráfico comparativo de la cantidad de información expresada en MB. Se observa la misma tendencia ya que los servicios remotos representan el 94.6 %, mientras que los servicios locales el 5.4 % del tráfico significativo. Una vez más, estos resultados revelan que la institución consume mucho más de lo que produce respecto a todos los servicios observados.

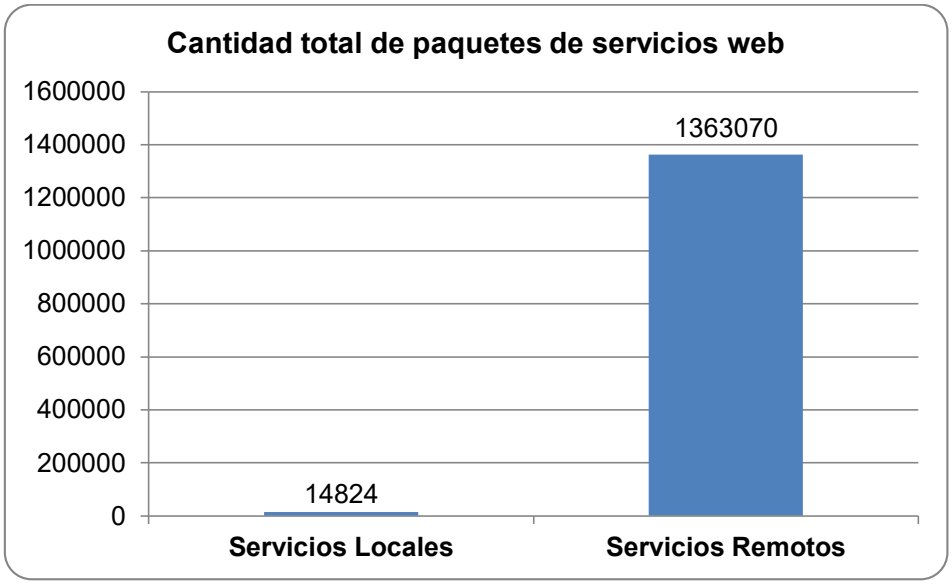


Figura 6.19 Cantidades totales de paquetes de servicios web locales y remotos

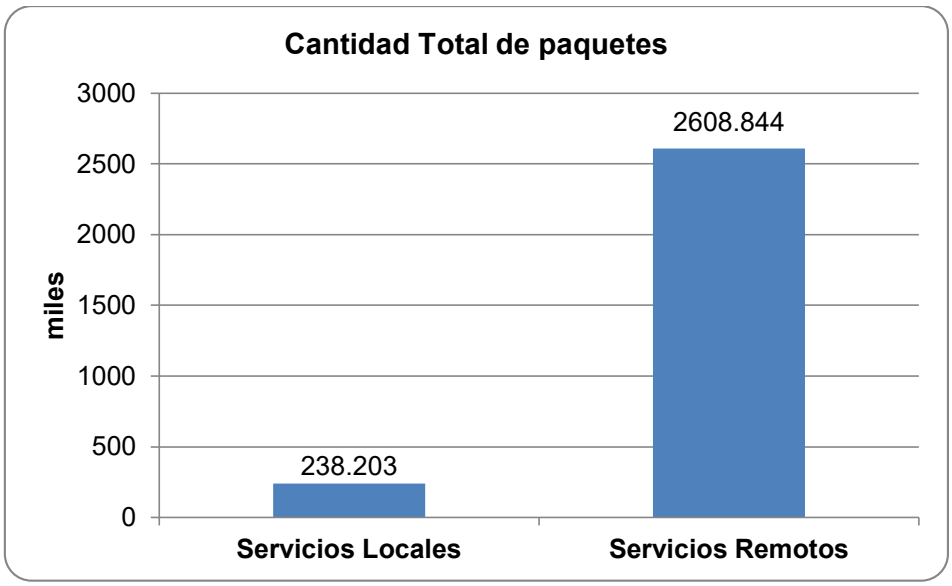


Figura 6.20 Cantidades totales de paquetes de servicios locales y remotos

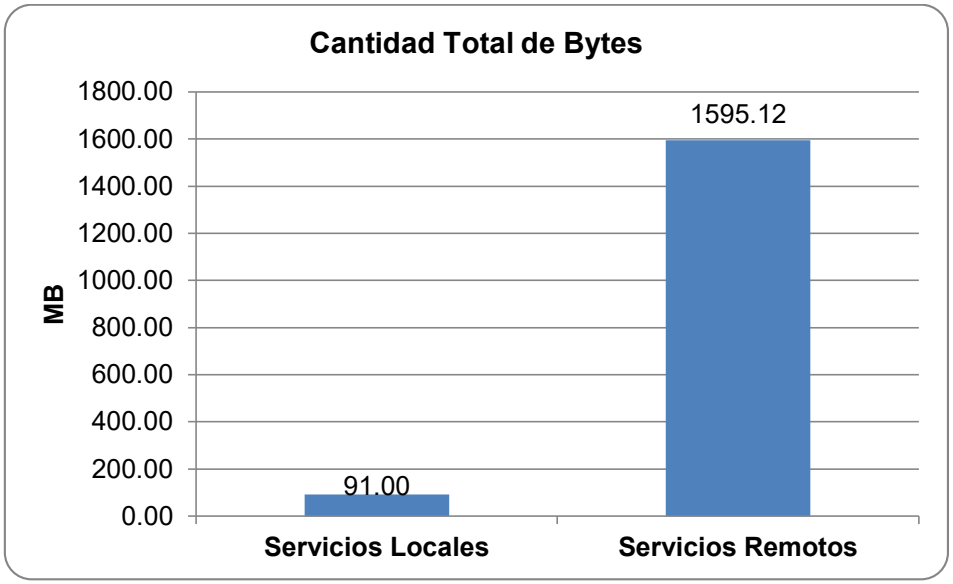


Figura 6.21 Cantidades totales de Bytes de servicios locales y remotos

6.2.2 Métricas de las clases de comportamiento

La clase de comportamiento que más predominó durante las pruebas es la IPORIGEN BC6 la cual representa el comportamiento de un servidor comunicándose con un número reducido de clientes.

Este hecho implica que existe poca actividad relevante en la institución tanto hacia servicios externos como internos. Esto se refleja claramente en el reporte de métricas (popularidad, número promedio y volatilidad) de las 4 clases de comportamiento que presentan el perfil de servidores mostrados en los tres gráficos de la Figura 6.22 “Reporte de métricas de las clases de comportamiento”.

Dichas gráficas muestran que no se observaron flujos de la clase PORIGEN BC23, lo cual revela que no existe tráfico agregado hacia el puerto de un servicio en particular. No se muestran los detalles de los resultados de las clases de comportamiento IPORIGEN BC7 y BC8 debido a que se detectó muy pocos flujos que pertenecen a tales clases de comportamiento.

La Figura 6.23 “Gráfica del tráfico durante el tiempo que duró la prueba”, muestra el reporte del tráfico durante cada intervalo de 5 minutos. Esta gráfica es generada por la herramienta de monitoreo “cacti” la cual monitorea la cantidad de paquetes totales entrantes y salientes que atraviesan dicha interfaz.

La Figura 6.24 “Reporte de la cantidad total de bytes durante el tiempo que duró la prueba”, muestra una gráfica, generada por la aplicación de reportes del sistema en la cual se muestra la cantidad total de bytes observados por el módulo de captura durante cada intervalo de 5 minutos. Ambas gráficas muestran semejanzas ya que en ellas se observan claramente los picos de tráfico en horas de máxima demanda en cada uno de los 5 días que duró la prueba. Esto comprueba el correcto funcionamiento del sistema.

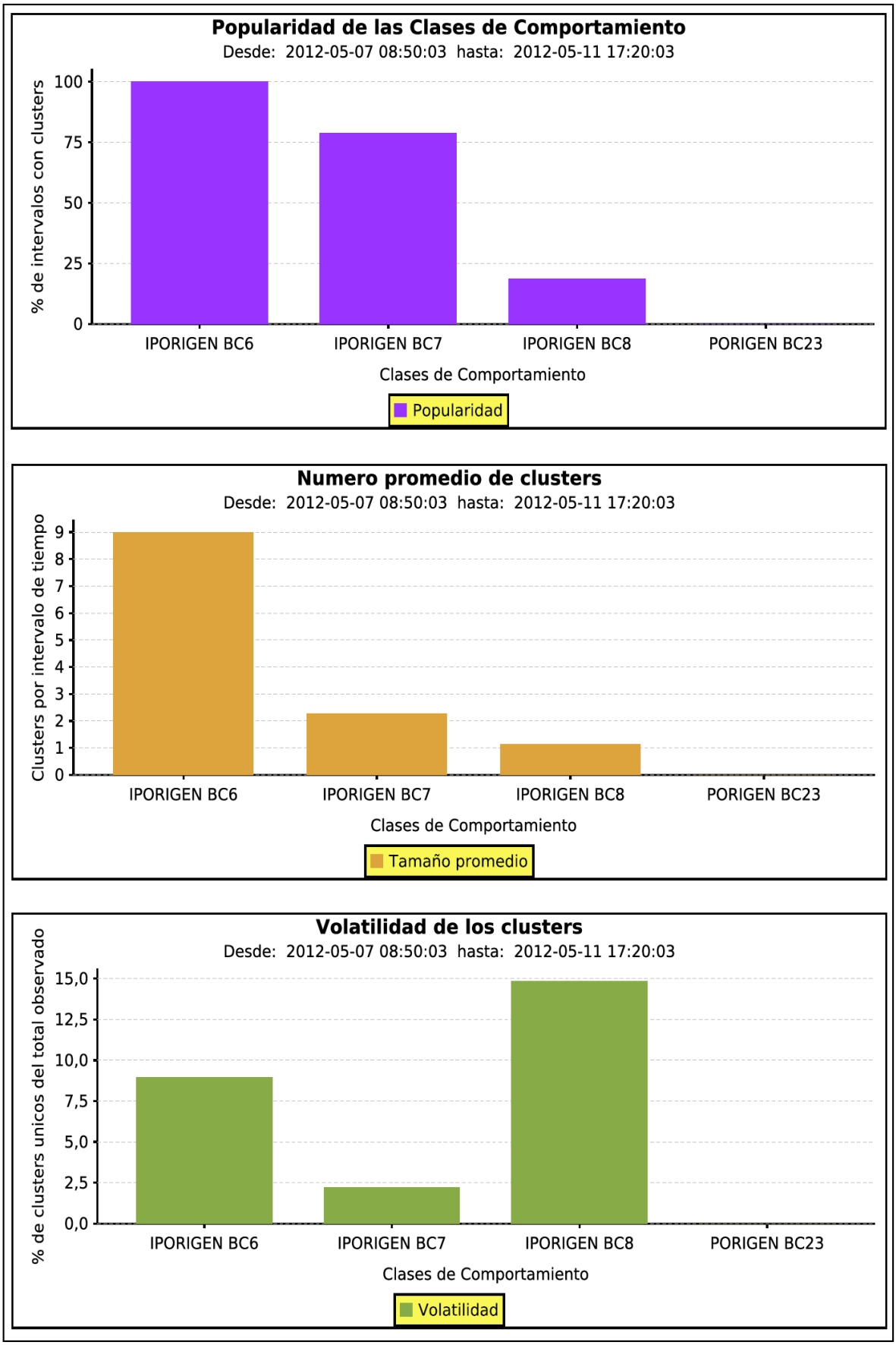


Figura 6.22 Reporte de métricas de las clases de comportamiento

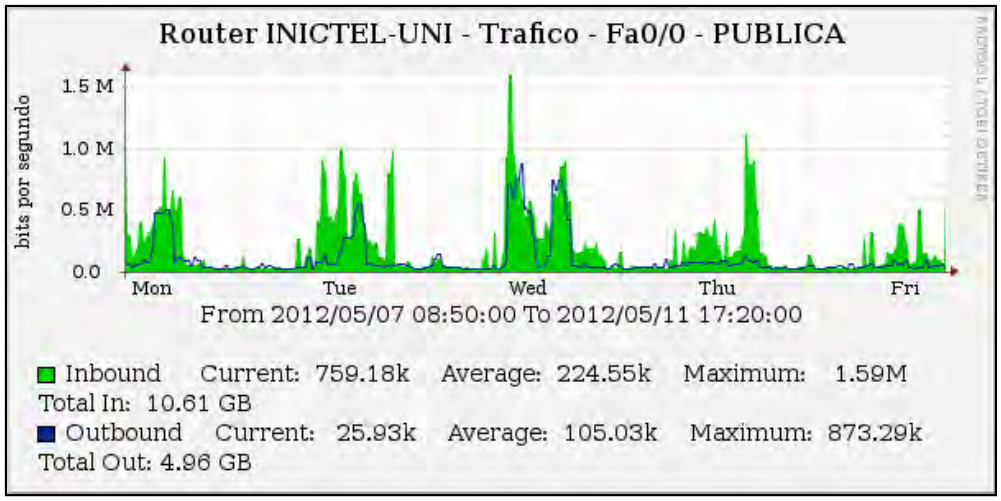


Figura 6.23 Gráfica del tráfico durante el tiempo que duró la prueba (Fuente: “cacti”)

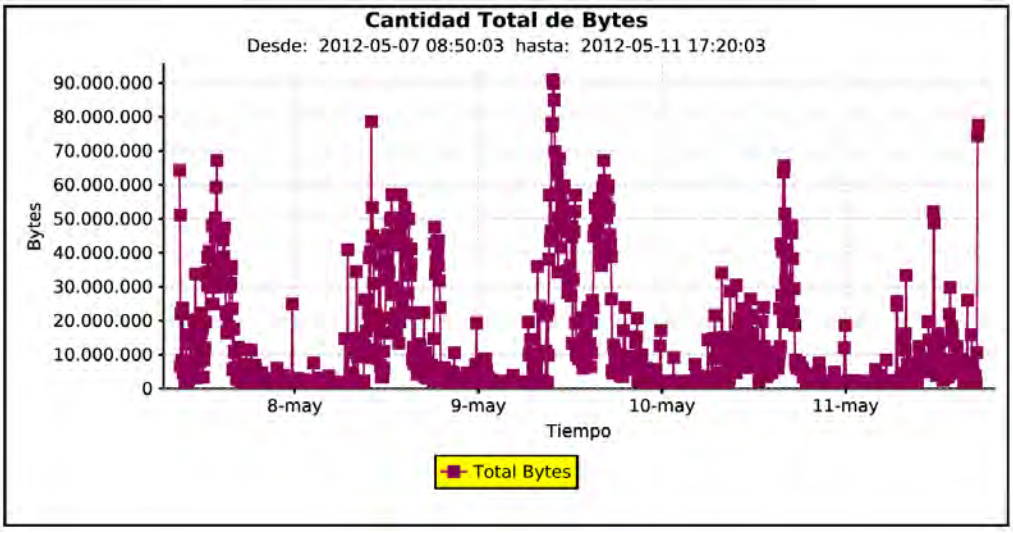


Figura 6.24 Reporte de la cantidad total de bytes durante el tiempo que duró la prueba

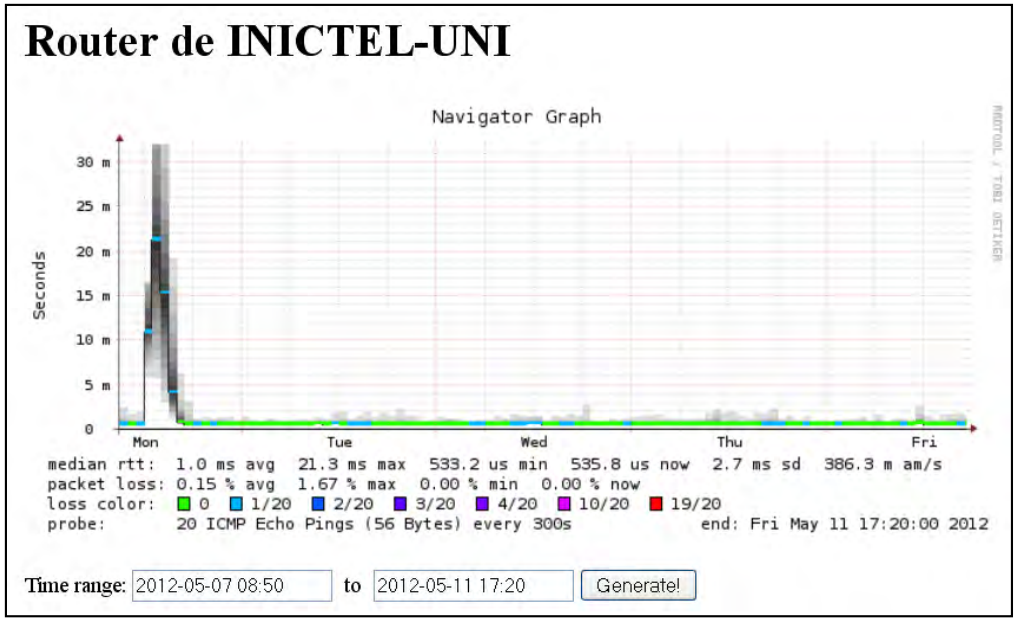


Figura 6.25 Retardo entre el router principal y el servidor (Fuente: “Smokeping”)

La Figura 6.25 “Retardo entre el router principal y el servidor” ilustra una gráfica generada por la herramienta de monitoreo “Smokeping”, la cual muestra el tiempo RTT (Round-trip time) entre el servidor y el router principal cada 5 minutos.

Se observa un tiempo de ida y vuelta promedio de 1 ms, lo cual indica que un paquete enviado desde el router hacia el servidor tiene un retardo promedio de 500 µs considerando el tiempo de ida igual al tiempo de vuelta. Este valor aproxima el retardo producido por la red durante la exportación de la información de flujos desde el módulo de captura hacia el servidor.

6.3 Análisis del impacto del sistema en la red de pruebas

Según la naturaleza del sistema implementado, se considera que los motivos principales que causan una disminución del rendimiento en la red de pruebas son los siguientes:

a. El uso de un Hub o concentrador para la derivación del tráfico: Debido a la naturaleza del hub, se observaron colisiones en el enlace donde fue colocado. La luz indicadora de colisiones parpadeaba la mayor parte del tiempo. A pesar de este hecho, se produjeron retardos que no fueron percibidos por los usuarios internos y los servicios funcionaron de manera normal.

b. La interrupción del enlace troncal al momento de colocar el hub: Durante la instalación del hub en el enlace troncal, fue necesario interrumpir el acceso a internet a los usuarios de la red interna. El lapso de interrupción tuvo un tiempo máximo de 5 segundos. Esta interrupción tuvo un impacto minúsculo sobre los clientes de estos servicios.

c. El tráfico generado por el módulo de captura: Según la naturaleza del sistema diseñado, el conjunto de paquetes generado por el módulo de captura es enviado hacia el servidor central de procesamiento a través de la misma red que está siendo analizada (red LAN de la RAAP en INICTEL-UNI). Por tal motivo, existe un pequeño incremento del tráfico en el sentido entrante a la red. La cantidad máxima de paquetes que son enviados hacia el servidor es 40 veces menor que la cantidad de paquetes que detecta el módulo de captura. Por cada “40n” paquetes recibidos en el módulo o entrantes a la red LAN, se envían “n” paquetes como máximo hacia el servidor central de procesamiento y como el módulo está ubicado en la cabecera de la red, se perciben en total “41n” paquetes entrantes a la red LAN.

6.4 Estimación de costos

En esta sección se citan todos los elementos que fueron utilizados para el desarrollo del proyecto. Asimismo se detallan los costos referenciales de cada uno según los precios ofrecidos por proveedores locales. Asimismo, se omite los costos de algunos elementos que debido a su naturaleza, están libres de costos para fines de realizar únicamente una

maqueta de pruebas como en este caso.

6.4.1 Elementos necesarios para la implementación del proyecto

En esta sección se detallan los elementos tanto de hardware como de software que fueron utilizados para el diseño, pruebas de validación y la puesta en funcionamiento del proyecto considerando la utilización de un único módulo de captura y un único servidor central de procesamiento.

Los dispositivos listados a continuación (Tabla 6.1) son los que fueron utilizados para la implementación de la prueba experimental. Asimismo, se consideran los elementos de software utilizados para el diseño del módulo de captura, diseño del servidor central de procesamiento y otros.

Tabla 6.1 Elementos utilizados en el proyecto de tesis

	Elementos de hardware	Elementos de software
Módulo de captura	Tarjeta de desarrollo modelo ML505 de Xilinx. 1 memoria compact flash de 128 MB. 1 grabador de memorias compact flash. 1 cable USB para la descarga de programas en el módulo. 1 CPU, 1 monitor, 1 teclado y 1 mouse compatibles.	Licencia para el uso de Xilinx ISE versión 11.3. Licencia para el uso de Xilinx EDK versión 11.3. Licencia para el uso del sistema operativo Windows XP Profesional SP3.
Servidor central de procesamiento	1 servidor HP Workstation. 1 monitor, 1 teclado y 1 mouse del servidor HP.	Sistema Operativo Linux, distribución Ubuntu versión 10.04.4 LTS. Plataforma Java Development Kit (JDK) versión 1.6. Plataforma de desarrollo de software Netbeans version 7.0.1. Servidor de aplicaciones web Apache Tomcat versión 6.0. Software generador de reportes iReport versión 3.7.6. Motor de base de datos MySQL versión 5.1.61.
Otros	1 hub marca 3COM de 2 velocidades 10/100Mbps. 1 laptop HP. 3 Cables RJ45 cat. 5E de 1m. 1 cable adaptador USB-serie marca Trendnet TU-S9. 1 cable serial con conectores hembra en ambos terminales.	Generador de paquetes packETH versión 1.7. Generador de paquetes sendIP versión 2.5.

6.4.2 Evaluación de precios de cada elemento

En esta sección se indican los precios referenciales de cada elemento mencionado en la sección anterior. Las cifras mostradas en la Tabla 6.2 son de proveedores locales y ya incluyen el costo de importación y el IGV.

Tabla 6.2 Lista de precios de cada elemento utilizado en el proyecto

Tipo de elemento	Cant.	Descripción	Observaciones	Costo Total S/.
Hardware	1	Tarjeta de desarrollo modelo ML505 de Xilinx + Memoria compact flash de 128 MB + Cable USB de Xilinx para la descarga de programas en el módulo + Cable serial con conectores hembra en ambos terminales.		6640.00
	1	Grabador de memorias compact flash.		350.00
	1	CPU, monitor, teclado y mouse compatibles		2400.00
	1	Servidor HP Workstation + 1 monitor, 1 teclado y 1 mouse del servidor HP.		4000.00
	1	hub o concentrador marca 3COM de dos velocidades 10/100 Mbps.		580.00
	1	Laptop HP (opcional).		-
	3	Cables RJ45 categoría 5E de 1m.		6.00
	1	Cable adaptador USB-serial marca Trendnet TU-S9.		65.00
Software	1	Licencia para el uso de Xilinx ISE(1) versión 11.3.	Licencia de evaluación del software por 30 días únicamente para fines de pruebas.	-
	1	Licencia para el uso de Xilinx EDK(2) versión 11.3.	Licencia de evaluación del software por 30 días únicamente para fines de pruebas.	-
	1	Licencia para el uso del sistema operativo Windows XP Profesional SP3.		530.00
	1	Sistema Operativo Linux, distribución Ubuntu versión 10.04.4 LTS.	Software Libre	-
	1	Plataforma Java Development Kit (JDK) versión 1.6.	Software Libre	-
	1	Plataforma de desarrollo de software Netbeans versión 7.0.1.	Software Libre	-
	1	Servidor de aplicaciones web Apache Tomcat versión 6.0.	Software Libre	-
	1	Software generador de reportes iReport versión 3.7.6.	Software Libre	-
	1	Motor de base de datos MySQL versión 5.1.61.	Software Libre	-
	1	Generador de paquetes packETH versión 1.7.	Software Libre	-
	1	Generador de paquetes sendIP versión 2.5.	Software Libre	-
Total S/.				14571.00

(1) El software ISE (Integrated Software Environment) de propiedad de Xilinx, incluye herramientas de software para el diseño del FPGA. El tipo de licencia utilizada es

únicamente para la evaluación del software por 30 días (para realizar las pruebas). Este tipo de licencia no tiene costo.

(2) El software EDK (Embedded Development Kit) de propiedad de Xilinx, incluye herramientas de software para el diseño de hardware y software embebido en FPGA tales como XPS (Xilinx Platform Studio) y SDK (Software Development Kit) utilizadas en el proyecto. El tipo de licencia utilizada es únicamente para la evaluación del software por 30 días (para realizar las pruebas). Este tipo de licencia no tiene costo.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. El utilizar un modulo de FPGA permitió asignar un procesador con un solo hilo de ejecución para el procesamiento de los paquetes. Además se hizo uso eficiente del stack TCP/IP de la plataforma de hardware utilizada. Esto permite tener un diseño eficiente, flexible y de rápido tiempo de desarrollo.
2. El análisis del tráfico de red basado en flujos permitió obtener información detallada sobre los patrones característicos del tráfico de la red. Con esto, se obtiene un mayor nivel de observación de toda la red en su conjunto y en especial facilita el estudio de su comportamiento en el tiempo.
3. La técnica utilizada para el modelamiento estructural basado en la teoría de la información, permitió obtener un resumen compacto de los flujos de paquetes que circulan hacia dentro y hacia fuera de la red.
4. Las tablas mostradas como resultado de las pruebas realizadas permitieron determinar aquellos servicios IP locales más utilizados por usuarios externos y asimismo permitió identificar aquellos usuarios externos que utilizan cada uno de los servicios de la institución en un determinado intervalo de tiempo. Utilizando esta información, se puede determinar a largo plazo la tendencia en el uso de cada servicio así como la ubicación de los principales usuarios de cada servicio.
5. Los indicadores de la producción de servicios, obtenidos como producto de las pruebas realizadas, revelan claramente la situación actual de los servicios IP en la red de pruebas.
6. Los indicadores permitieron identificar aquellos servicios que tienen un mayor número de clientes, los cuales, requieren contar mayores niveles de ancho de banda y menores niveles de retardo. Igualmente, se pudo identificar aquellos servicios con un menor número de usuarios que el resto, los cuales requieren ser mejorados o modernizados para poder incrementar el número de usuarios en el tiempo.

Recomendaciones

1. En el enlace troncal de una red de tamaño considerable (más de 100 dispositivos finales) el nivel de tráfico es usualmente mayor. Por tal motivo, el módulo de captura diseñado puede ser replicado adecuadamente en distintos puntos troncales de la red. Con

esto, se puede obtener indicadores de producción de servicios de redes más grandes sin importar el nivel del tráfico ni el número de usuarios de la red, siempre y cuando en cada punto de observación no se exceda la capacidad de captura del módulo. Este hecho confirma el nivel de aplicación para redes extendidas del sistema diseñado.

2. Se sugiere establecer una conexión redundante entre el servidor colector de información de flujos y el módulo de captura para mejorar la disponibilidad de la captura de información del tráfico.

3. Para elevar aún más la velocidad de procesamiento del módulo de captura se recomienda realizar el diseño de la pila TCP/IP y los procesos involucrados utilizando únicamente los recursos de hardware del FPGA. De este modo se podría utilizar el módulo de captura en enlaces de red de mayor velocidad.

4. Los resultados obtenidos facilitan la toma de decisiones para la aplicación de mecanismos que permitan elevar el nivel del tráfico de servicios locales. Estos mecanismos pueden incluir la asignación de mayor ancho de banda a ciertos servicios para facilitar el rápido acceso a los usuarios y así, poder aumentar la preferencia de tales servicios.

5. La técnica utilizada para el modelamiento estructural basado en la teoría de la información, puede ser empleada para complementar la funcionalidad del sistema. El estudio de otros patrones de comportamiento del tráfico pueden detectar comportamientos anómalos de usuarios y servicios. Se debe seguir trabajando en complementar la herramienta para que se puedan detectar ataques y proteger los servicios mediante mecanismos de identificación de atacantes y descartar las direcciones IP de estas fuentes como podría ser el caso de ataques de botnets, denegación de servicio entre otros.

ANEXO A
MÓDULO DE DESARROLLO UTILIZADO EN EL PROYECTO

ANEXO A MÓDULO DE DESARROLLO UTILIZADO EN EL PROYECTO

En esta sección se va a describir la arquitectura del módulo de desarrollo utilizado para el diseño del hardware de captura de paquetes. También se va a brindar las principales características del modelo de FPGA así como de otros componentes de hardware incorporados en la tarjeta. Este anexo se divide en las siguientes secciones:

- Tarjeta de desarrollo ML505 de Xilinx.
- FPGA Virtex 5 de Xilinx.
- Procesador Softcore MicroBlaze de Xilinx.
- Características del Hardware Ethernet MAC embebido.
- Componentes externos al FPGA.

A.1 Tarjeta de desarrollo ML505 de Xilinx

Para el desarrollo e implementación se utilizó una tarjeta de desarrollo con FPGA modelo ML505 de Xilinx. Es una tarjeta de evaluación de propósito general que tiene incorporado un FPGA de la familia Virtex-5 y además cuenta con diversas interfaces de hardware que sirven para la comunicación con dispositivos externos. La tarjeta presenta los siguientes componentes [66]:

- 1 FPGA Virtex-5 de Xilinx modelo XC5VLX50T-1FFG1136.
- 2 memorias Flash PROM de Xilinx modelo XCF32P. Cada memoria es de 32Mb y sirve para almacenar grandes configuraciones de dispositivos.
- 1 chip controlador marca System ACE de Xilinx, para la configuración de una memoria CompactFlash con conector tipo I.
- 1 CPLD XC95144XL para la interconexión de circuitos integrados (glue logic).
- 1 Módulo tipo SODIMM para memoria DDR2 de 256 MB y de 64 bits de ancho.
- 1 chip programable para la generación del reloj del sistema, 1 zócalo para un oscilador de reloj de 3.3V y 2 pares diferenciales de conectores SMA (Subminiature versión A) para un reloj externo.
- 8 Interruptores con encapsulamiento DIP (Dual In-line Package), 8 diodos LEDs, pulsadores y un codificador rotatorio de propósito general.
- 1 Cabecera de expansión con 32 terminales de entrada/salida, 16 pares diferenciales LVDS, 14 entradas/salidas disponibles compartidas con los botones, diodos LEDs, energía eléctrica de la tarjeta, cadena de expansión JTAG (Joint Test Action Group) y expansión de un bus IIC (Inter-Integrated Circuit).
- 1 codec AC97 (Audio códec 97) de audio estéreo con enchufes line-in, line-out, de audífono de 50 mW, para entrada de micrófono, para audio digital S/PDIF (Sony/Philips Digital Interface Format) y un transductor de audio piezoeléctrico.
- 1 puerto serial RS-232 con conector tipo DB9 y una cabecera para un segundo puerto

serial.

- 1 pantalla LCD de 2 líneas de 16 caracteres cada una.
- 1 memoria EEPROM IIC de 8Kb.
- 2 Conectores tipo PS/2 para teclado y mouse.
- 1 entrada de video tipo VGA y una salida de video tipo DVI (Digital Video Interface con un adaptador DVI a VGA.
- 1 memoria síncrona ZBT SRAM (Zero-Bus Turnaround SRAM) de 9Mb en un bus de datos de 32 bits con 4 bits de paridad.
- 1 memoria flash lineal tipo StrataFlash marca Intel P30 de 32 MB.
- 1 Memoria Flash SPI (Serial Peripheral Interface) de 2 MB.
- Transceptor Ethernet de 3 velocidades 10/100/1000 y un conector RJ45 con soporte para interfaces Ethernet de capa física tipo MII, GMII, RGMII y SGMII.
- 1 chip de interfaz USB con puertos tipo Host y Periférico.
- 1 batería de litio recargable para mantener las claves de cifrado del FPGA.
- 1 puerto de configuración JTAG para su uso con cable paralelo III, cable paralelo IV, o cable USB de descarga.
- 1 suministro de energía incorporado para todos los voltajes necesarios.
- 1 chip de monitoreo de voltaje y temperatura con controlador de ventilador.
- 1 adaptador de corriente alterna 5V, 6A.
- 1 diodo LED indicador de poder.
- Interfaces físicas Ethernet de tipo MII (Media Independent Interface), GMII (Gigabit MII), RGMII (Reduced GMII), y SGMII (Serial GMII).
- 1 Módulo SFP (Small Form-factor Pluggable) para 1000Base-X para transceptor GTP/GTX.
- Entrada y salida diferencial con conectores SMA para transceptores GTP/GTX.
- SGMII para transceptor GTP/GTX.
- PCI Express x1 (Peripherial Component Interface) para transceptor GTP/GTX.
- SATA (Serial Advanced Technology Attachment) con cable loopback para transceptor GTP/GTX.
- Síntesis de reloj de circuitos integrados para transceptor GTP/GTX.
- Puerto MICTOR (Matched Impedance Connector).
- Puerto de depuración BDM (Background Debug Mode).

En la Figura A.1, se presenta a continuación el diagrama de bloques para una tarjeta modelo ML50x. La letra x indica que este diagrama es válido para los modelos ML505, ML506 y ML507. Las Figuras A.2 muestran una fotografía real de la tarjeta indicando cada parte con números. La leyenda se muestra posteriormente (Tabla A.1).

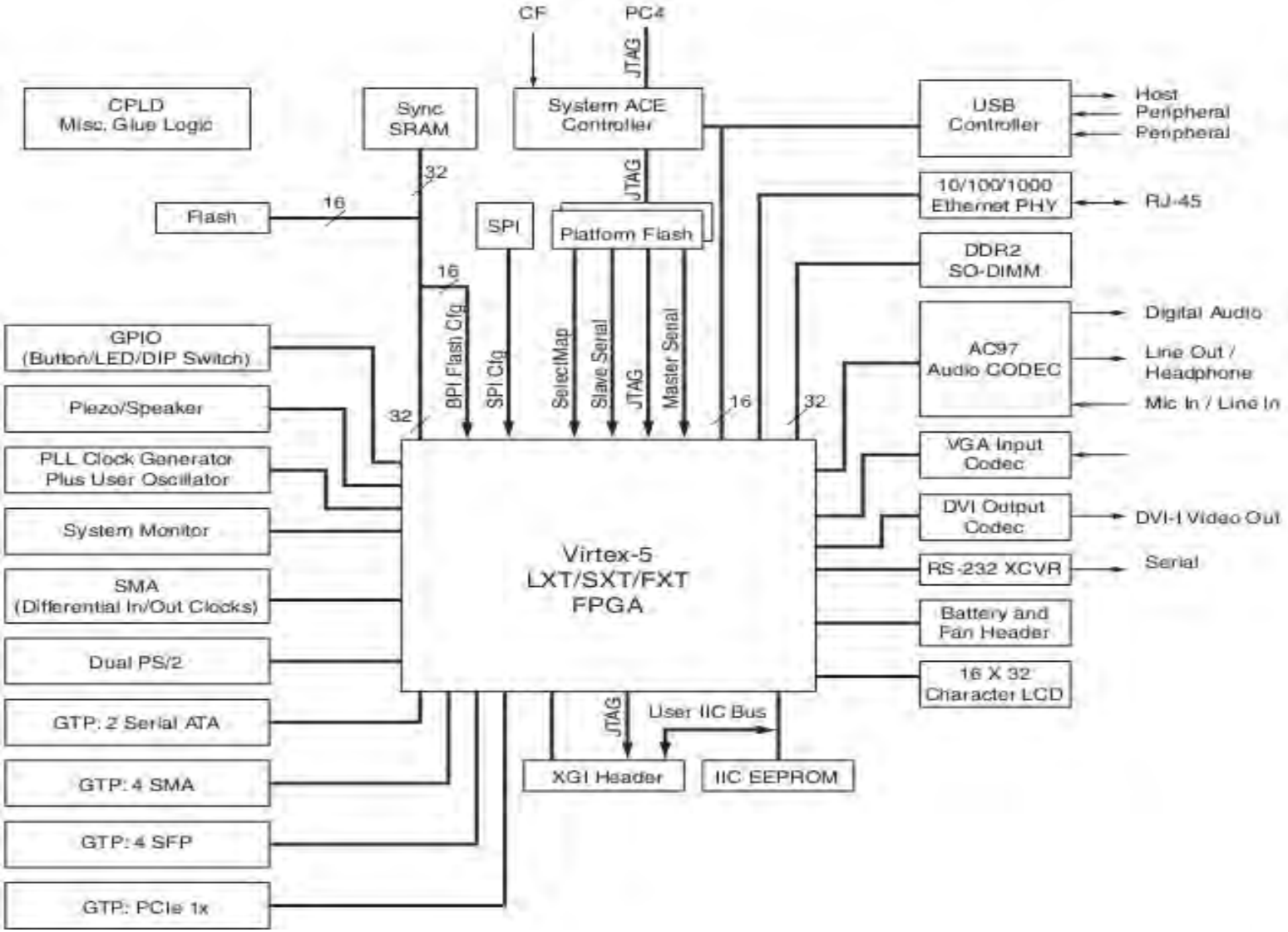


Figura A.1 Diagrama de bloques de la tarjeta de evaluación ML50x (Fuente: [66])

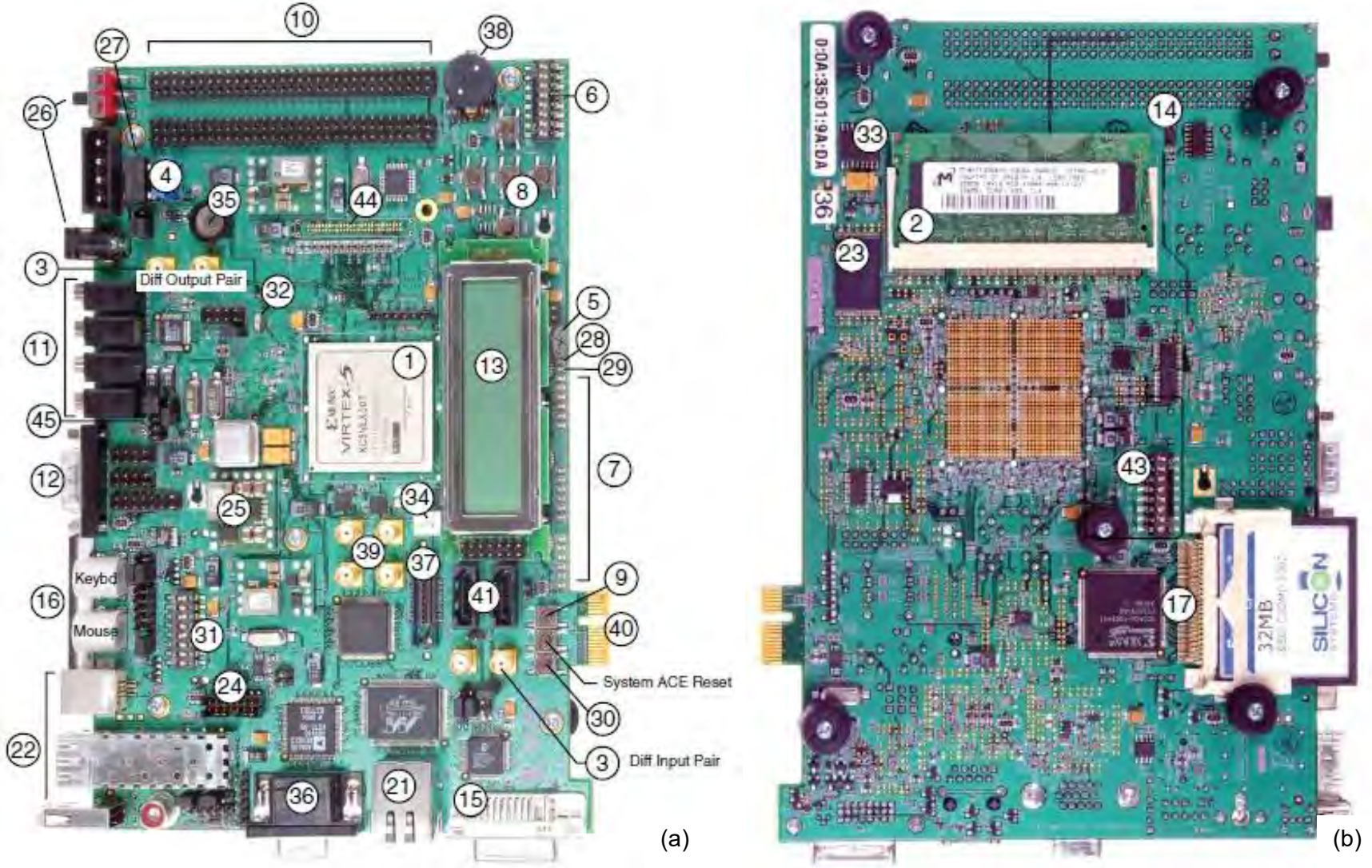


Figura A.2 Componentes de la tarjeta ML505, (a) parte frontal y (b) parte posterior (Fuente: [66])

Tabla A.1 Leyenda de tarjeta (Fuente: [66])

1	FPGA Virtex-5 modelo XC5VLX50T-1FFG1136.	**24**	Puerto de configuración JTAG.
2	DDR2 SODIMM.	**25**	Suministro de poder en la tarjeta.
3	Entrada/salida de reloj diferencial con conectores SMA.	**26**	Adaptador de corriente alterna, enchufe y conmutador de poder.
4	Osciladores.	**27**	Diodo LED indicador de poder.
5	Ajuste del brillo y contraste del LCD.	**28**	Diodo LED DONE.
6	Interruptores de propósito general (Activo en alto).	**29**	Diodo LED INIT.
7	Diodos LEDs de usuario y de error (Activo en alto).	**30**	Interruptor de programa.
8	Pulsadores (Activo en alto).	**31**	Interruptores tipo DIP para el modo y para la dirección de configuración.
9	Botón de reinicio del CPU (Activo en bajo).	**32**	Batería de clave de cifrado.
10	Cabeceras de expansión XGI.	**33**	Memoria Flash SPI (Serial Peripheral Interface).
11	Codec de audio estéreo AC97.	**34**	Controlador del ventilador IIC y monitor de voltaje y temperatura.
12	Puerto serial RS-232.	**35**	Transductor piezoeléctrico.
13	LCD de 2 líneas de 16 caracteres cada una.	**34**	Codec de entrada de video VGA.
14	Bus IIC con una EEPROM de 8Kb.	**37**	Puerto MICTOR (Matched Impedance ConnecTOR) para rastreo y depuración.
15	Conector DVI.	**38**	Codificador rotatorio.
16	Puertos PS/2 para mouse y teclado.	**39**	Entrada y salida diferencial del transceptor GTP/GTX con conectores SMA (SubMiniature versión A).
17	Conector de CompactFlash y System ACE (Advanced Configuration Environment).	**40**	Interface PCI (Peripheral Component Interconect) express x1.
18	Memoria síncrona ZBT SRAM (Zero-Bus Turnaround SRAM).	**41**	Conector SATA (Serial Advanced Technology Attachment).
19	Memoria flash lineal.	**42**	Conector SFP (Small Form-factor Pluggable).
20	CPLD XC95144XL de Xilinx.	**43**	Circuitos de generación de reloj del transceptor GTP/GTX.
21	Interfaz Ethernet de capa física de 3 velocidades 10/100/1000.	**44**	Soft touch landing pad de Agilent.
22	Controlador USB con puertos Host y Periférico.	**45**	Monitor del sistema.
23	Dispositivos de almacenamiento de configuración flash PROM modelo XCF32P de Xilinx		

El FPGA de la tarjeta puede ser configurado utilizando información proveniente de cualquiera de los siguientes dispositivos:

- Cable de descarga de Xilinx utilizando el modo JTAG: Se utiliza un cable USB que conecta una computadora y la tarjeta. En este caso, la tarjeta se programa con instrucciones provenientes de una PC.
- Controlador System ACE utilizando el modo JTAG: Se utiliza una memoria CompactFlash externa que almacena la información del diseño. Este controlador se encarga de interpretar dicha información y de programar la tarjeta según el contenido de la memoria externa.
- 2 memorias PROM Flash: Se utiliza el contenido de las memorias PROM Flash para programar la tarjeta.
- 1 memoria flash lineal: Se utiliza el contenido de la memoria flash lineal para programar la tarjeta.
- 1 memoria flash SPI: Se utiliza el contenido de la memoria flash SPI para programar la tarjeta.

En la tarjeta existe una cadena de elementos llamada cadena JTAG el cual sirve para configurar las dos memorias PROM flash, el CPLD, para depurar el diseño hardware y software y/o para programar la tarjeta, ya sea utilizando una memoria CompactFlash o directamente desde una PC. La Figura A.3 muestra una representación de esta cadena.

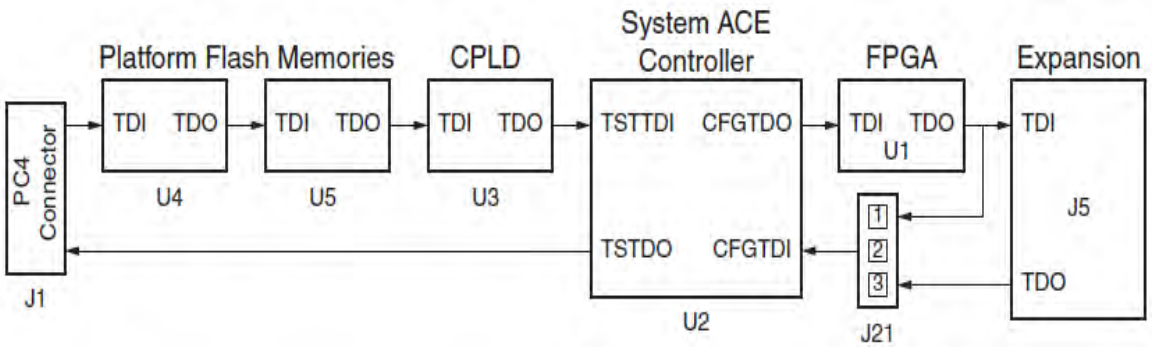


Figura A.3 Cadena JTAG de la tarjeta ML505 de Xilinx (Fuente: [66])

La cadena inicia en el conector PC4 (Parallel Cable IV) y pasa por las 2 memorias PROM flash, el CPLD, el controlador System ACE, el FPGA de la tarjeta y una extensión opcional de la cadena hacia una tarjeta de expansión. El jumper J21 determina si la cadena JTAG debe ser extendida a una tarjeta de expansión.

Para poder seleccionar el dispositivo a utilizar para la configuración, la tarjeta cuenta con 8 interruptores los cuales cumplen las siguientes funciones:

- En el diseño se utilizó el controlador de System ACE y una tarjeta CompactFlash que soporta hasta ocho imágenes de configuración, las cuales pueden ser seleccionadas según el estado de los interruptores 1, 2 y 3.
- Los interruptores 4,5 y 6 sirven para escoger uno de los 8 modos de configuración

posibles según el dispositivo que se utiliza para su configuración. La Tabla A.2 detalla cada uno de estos modos y la configuración de interruptores para cada uno.

Tabla A.2 Modos de configuración de la tarjeta ML505 de Xilinx (Fuente: [66])

Mode[2:0]	Mode
000	Master Serial (Platform Flash PROM, hasta 4 configuraciones)
001	SPI (una configuración)
010	BPI Up (Parallel NOR Flash, hasta 4 configuraciones)
011	BPI Down (Parallel NOR Flash, hasta 4 configuraciones)
100	Master SelectMAP (Platform Flash PROM, hasta 4 configuraciones)
101	JTAG (PC4, System ACE hasta 8 configuraciones)
110	Slave SelectMAP (Platform Flash PROM, hasta 4 configuraciones)
111	Slave Serial (Platform Flash PROM, hasta 4 configuraciones)

Para el diseño del módulo de captura se utilizó el controlador System ACE, el cual se encarga de interactuar con una memoria compact flash externa. En esta memoria se almacena archivos de extensión ".ace" que contienen información de configuración que sirve para programar la funcionalidad del FPGA. Según la Tabla A.2, para el diseño el estado de los interruptores es "101" correspondiente al modo JTAG.

- El interruptor 7 sirve para habilitar la memoria PROM flash de reserva. Esta función aún no está implementada ya que está destinada para usos futuros.
- El interruptor 8 sirve para habilitar o deshabilitar la configuración de la tarjeta por medio del controlador System ACE utilizando una memoria Compact Flash externa. Para el diseño este interruptor debe estar en el estado "habilitado".

La Tabla A.3 muestra en detalle la función de cada uno de los 8 interruptores de configuración.

Tabla A.3 Interruptores de configuración de la tarjeta ML505 de Xilinx (Fuente: [66])

Switch (SW3)	Función
1	bit 2 de configuración de dirección de imagen (más significativo)
2	bit 1 de configuración de dirección de imagen
3	bit 0 de configuración de dirección de imagen (menos significativo)
4	bit 2 de configuración de modo (más significativo)
5	bit 1 de configuración de modo
6	bit 0 de configuración de modo (menos significativo)
7	Memoria Flash PROM de reserva (On=Habilitado, Off= Deshabilitado).
8	Configuración del System ACE (On=Habilitado, Off= Deshabilitado).

El controlador System ACE programa el FPGA siempre y cuando exista una tarjeta CompactFlash insertada en una ranura en la parte trasera de la tarjeta. Además, se debe

de escoger correctamente una de las ocho imágenes (archivos de extensión “ace”) almacenadas en la memoria CompactFlash.

Luego se debe verificar que esté habilitada la configuración de la tarjeta por medio del controlador System ACE y se debe seleccionar el modo respectivo según se indica en las tablas. La tarjeta cuenta con un botón que sirve especialmente para reiniciar la programación del FPGA cuando se utiliza el controlador System ACE para su configuración.

Mayores detalles respecto a otros aspectos relacionados con la tarjeta de desarrollo ML505 de Xilinx, se encuentran en la referencia [66].

A.2 FPGA Virtex 5 de Xilinx

La información proporcionada en esta sección fue extraída de la referencia [67]. Aquí se otorga las características más importantes del FPGA Virtex-5 relacionadas únicamente con el diseño del módulo de captura ya que este dispositivo cuenta con más características que no son detalladas en esta sección.

A.2.1 Características generales

La tarjeta de desarrollo utilizada en el diseño cuenta con un FPGA de la familia Virtex-5 de Xilinx modelo XC5VLX50T. Esta familia de FPGAs utiliza la segunda generación de la arquitectura basada en columnas ASMBL (Advanced Silicon Modular Block) [68].

Los FPGA de la familia Virtex 5 contienen varios bloques de IP hardware incluyendo bloques de RAM/FIFOs de 36 Kbit, 25 x 18 DSP slices, tecnología SelectIO, bloques de interfaz de fuente síncorna ChipSync, funcionalidad de monitor del sistema, DCM (Digital Clock Manager) y generadores de reloj de PLL (Phase Locked Loop) y opciones de configuración avanzadas.

También tiene características que dependen de la plataforma donde se encuentre tales como bloques transceptores seriales de alta velocidad para una mejor conectividad serial, bloques compatibles con PCI Express, Ethernet MAC (Media Access Controllers) de 3 modos.

El FPGA Virtex 5 ofrece la mejor solución para satisfacer las necesidades de diseñadores de lógica programable, diseñadores de DSP de alto rendimiento y diseñadores de sistemas embebidos con DSP, microprocesador de hardware o en software y capacidades de conectividad.

Las siguientes tablas (A.4 y A.5) muestran cada uno de los componentes con que cuenta el FPGA modelo XC5VLX50T utilizado en el diseño así como la cantidad de cada componente.

Tabla A.4 Componentes en el FPGA Virtex 5 modelo XC5VLX50T (Fuente: [67])

Configurable Logic Blocks (CLBs)			DSP48E Slices[2]	Block RAM Blocks			CMTs[4]
Array (Row x Col)	Virtex-5 Slices[1]	Max Distributed RAM (Kb)		18 Kb[3]	36 Kb	Max (Kb)	
120 x 30	7200	480	48	120	60	2160	6

Tabla A.5 (Continuación) Componentes en el FPGA Virtex 5 modelo XC5VLX50T

PowerPC Processor Blocks	Endpoint Blocks for PCI Express	Ethernet MACs[5]	Max RocketIO Transceivers[6]		Total I/O Banks[8]	Max User I/O[7]
			GTP	GTX		
N/A	1	4	12	N/A	15	480

Nota:

1. Cada slice del FPGA virtex 5 contiene 4 LUTs (Look Up Tables) y 4 flip-flops.
2. Cada slice DSP48E contiene 25x18 multiplicador, un sumador y un acumulador.
3. Los bloques RAM son de 36 Kbits de tamaño. Cada bloque puede ser utilizado como 2 bloques independientes de 18 Kbits cada uno.
4. Cada CMT (Clock Management Tile) contiene 2 DCM (Digital Clock Manager) y 1 PLL (Phase Looked Loop).
5. Esta tabla muestra cada Ethernet MAC por separado en cada dispositivo.
6. Los transceptores GTP RocketIO están diseñados para funcionar desde 100 Mbps a 3.75 Gbps. Los transceptores GTX están diseñados para funcionar desde 150 Mbps hasta 6.5 Gbps.
7. Este número no incluye los transceptores RocketIO.
8. Incluye la configuración del banco 0.

A.2.2 Descripción de la arquitectura del FPGA Virtex 5

Un FPGA de la familia Virtex 5 consta de arreglos de compuertas programables con varios elementos configurables e IP cores [69] embebidos optimizados para el diseño de sistemas de alta densidad y rendimiento. La familia de FPGA Virtex 5 posee los siguientes elementos internos:

a. Bloques de entrada/salida:

Provee la interfaz entre los pines del FPGA y la lógica configurable interna. Soporta varios estándares de entrada/salida como:

- LVTTL (Low Voltage Transistor-Transistor Logic).

- LVCMOS (Low Voltage Complementary Metal Oxide Semiconductor) (3.3V, 2.5V, 1.8V, 1.5V, y 1.2V).
- PCI (33 y 66 MHz).
- PCI-X
- GTL (Gunning Transceiver Logic) y GTLP (GTL Plus).
- HSTL (High-Speed Transceiver Logic) de 1.5V and 1.8V (Clase I, II, III, y IV).
- HSTL de 1.2V (Clase 1)
- SSTL (Stub Series Terminated Logic) de 1.8V and 2.5V (Clase I y II).

Asimismo, cada bloque de entrada/salida soporta los siguientes estándares de señalización diferencial:

- LVDS (Low Voltage Differential Signaling) y LVDS extendida (sólo 2.5V).
- BLVDS (Bus LVDS).
- ULVDS (Ultra LVDS).
- Hypertransport.
- HSTL diferencial de 1.5V and 1.8V (Clase I y II).
- SSTL diferencial de 1.8V and 2.5V (Clase I y II).
- RSDS (Reduced Swing Differential Signaling) de 2.5V punto a punto.

Cada bloque puede ser conectado a la lógica ChipSync [70] para el mejoramiento de la interconexión de fuente-síncrona. Esta interconexión incluye alineación por bit (tanto en señales entrantes como salientes), serializador/des-serializador de datos (SerDes), divisores de reloj y recursos dedicados de entrada/salida y de reloj.

Cada bloque puede ser utilizado como cualquiera de las siguientes funciones:

- Operación de LVDS simple o diferencial.
- Bloque de entrada con SDR (Single Data Rate) opcional o registro DDR (Double Data Rate).
- Bloque de salida con SDR opcional o registro DDR.
- Bloque bidireccional.
- Circuito de alineación por bit.
- Entrada/salida dedicada con recursos de reloj.
- Serializador/des-serializador de datos embebido.

b. CLBs (Configurable Logic Blocks)

Son los elementos de lógica básicos de los FPGAs de Xilinx. Proporcionan lógica combinacional y secuencial así como memoria distribuida y capacidades de registro de cambio SRL32. Cada uno de estos bloques está compuesto de 2 slices idénticos cada uno conteniendo lo siguiente:

- 4 generadores de funciones: Configurables como LUTs (Look Up Tables) de 6 entradas

o LUTs de 5 entradas y doble salida.

- 4 elementos de almacenamiento: Pueden ser configurados ya sea como flip-flops tipo D disparados por flanco o latches sensibles al nivel.
- Compuertas lógicas aritméticas.
- Grandes Multiplexores.

Cada bloque se conecta a una matriz de conmutadores para acceder a recursos de ruteo. Para mayores detalles sobre los CLBs se puede consultar la referencia [71].

c. Módulos de bloques de RAM

Estos módulos proporcionan memorias RAM de 36 Kbit de doble puerto los cuales son puestos en cascada para formar bloques de memoria más grandes. Además contienen lógica de FIFO (First Input First Output) programable opcional para aumentar la utilización del dispositivo.

Cada bloque de RAM también puede ser configurado como 2 bloques de RAM de 18 Kbit de doble puerto proporcionando mayor granularidad para diseños que necesiten bloques de RAM más pequeños.

Cada uno de los 2 puertos de cada bloque es totalmente síncrono e independiente y ofrece 3 modos de “lectura durante escritura”. Opcionalmente pueden funcionar como: registros de pipeline, circuito de control de reloj, FIFO embebido, ECC (Error Correcting Code) y byte de habilitación de escritura. Para mayores detalles sobre los CLBs se puede consultar la referencia [71].

d. Slices DSP48E embebidos en cascada

Cada Slice contiene 25x18 multiplicadores de complemento a 2 y un sumador/restador/acumulador de 48 bits que permite la implementación de algoritmos de alta velocidad de procesamiento digital de señales DSP (Digital Signal Processing). Además, cada slice puede ser utilizado para realizar funciones lógicas bit a bit.

e. Bloques CMT (Clock Management Tile)

Proporcionan la mayor flexibilidad y el más alto rendimiento de reloj para FPGAs. Cada bloque contiene 2 bloques DCM (Digital Clock Manager) con auto-calibración digital y un bloque PLL (Phase-locked loop) con auto-calibración analógica para la compensación del retardo en la distribución de la señal de reloj, multiplicación/división de reloj, granularidad fina/gruesa del desplazamiento de fase de la señal de reloj y para el filtro de jitter de la señal de reloj entrante.

Los bloques DCM y PLL pueden ser utilizados de manera independiente o en cascada. Se dispone de hasta 6 bloques CMT que proporcionan hasta 18 elementos generadores de reloj. Cada bloque DCM puede ser utilizado para eliminar el retardo de distribución para generar señales de reloj alineadas (sin skew).

Asimismo un DCM provee de desplazamientos de fase de 90°, 180° y 270°. Igualmente, proporciona una salida de reloj con síntesis de frecuencia flexible igual a una fracción o múltiplo entero de la frecuencia de la señal de reloj entrante.

f. Bloques terminales integrados para PCI Express

Cada bloque Proporciona la funcionalidad para terminales PCI Express x1, x4 ó x8. Se puede implementar un terminal PCI Express completo con la mínima utilización de lógica del FPGA utilizando estos bloques junto con los transceptores RocketIO.

Los dispositivos Virtex-5 de tipo LXT, SXT, TXT y FXT contienen hasta 4 bloques terminales integrados. Estos bloques implementan la capa de transacción, capa de enlace de datos y funciones de capa física. Los bloques cumplen con la especificación PCI Express Base 1.1.

g. Bloques de Ethernet MAC de 3 modos (10/100/1000 Mbps)

Los dispositivos Virtex 5 tipo LXT, SXT, TXT y FXT contienen hasta 8 Ethernet MACs embebidos, 2 por cada bloque Ethernet MAC. Cada bloque tiene las siguientes características:

- Está diseñado para la especificación IEEE 802.3-2002 [72].
- Conformidad UNH (University of New Hampshire) revisada.
- Interfaz RGMII/GMII con interfaz SGMII o SelectIO cuando se utiliza con transceptores RocketIO.
- Transmisión en half dúplex o full dúplex.
- Soporte de tramas Jumbo Ethernet.
- Soporte de 1000 Base-X PCS/PMA cuando se utiliza con transceptores RocketIO.
- Conexión a microprocesadores por medio del bus DCR (Device Control Register).

h. Transceptores GTP RocketIO

Estos transceptores poseen las siguientes características:

- Son capaces de ejecutar desde 100 Mbps hasta 3.75 Gbps.
- Recuperación total de datos y reloj.
- Soporte de camino de datos de 8/16 bits o 10/20 bits.
- Codificación/decodificación 8B/10B opcional basado en FPGA.
- Buffer FIFO/elastic integrado.
- Soporte de “Channel Bonding” y corrección de reloj.
- Generación/revisión de CRC de 32 bits embebido.
- Función de detector de coma o detección A1/A2 integrada.
- Pre-énfasis programable (ecualización de transmisor AKA).
- Oscilación de salida del transmisor programable.
- Ecualización del receptor programable.

- Terminación del receptor programable.
- Tiene soporte embebido para señalización OOB (Out of band) en Serial ATA y detección de señal eléctrica libre.
- Generador/revisor de PRBS (Pseudo Random Binary Sequence) embebido.

Cada componente del FPGA está unido con otro mediante un GRM (General Routing Matrix) el cual es una arreglo de conmutadores ubicados entre cada uno de los componentes.

Cada elemento programable esta unido a la matriz de conmutadores para que éste pueda realizar conexiones múltiples al GRM. Las conexiones de rutas están optimizadas para soportar interconexiones de CLBs con el menor número de saltos.

La reducción de saltos incrementa el rendimiento del diseño después de la etapa PAR (Place and route). Todos los elementos programables son controlados por valores contenidos en elementos de almacenamiento estáticos. Estos valores son cargados dentro del FPGA durante la configuración y también pueden ser recargados para cambiar las funciones de los elementos programables.

A.3 Procesador Softcore MicroBlaze de Xilinx.

MicroBlaze™ es un procesador soft-core RISC de 32 bits de arquitectura hardvard con un conjunto de instrucciones optimizadas para aplicaciones embebidas. Utilizando este procesador se tiene la completa flexibilidad para seleccionar la combinación de periféricos, memoria e interfaces produciendo el sistema exacto que el diseñador necesita al menor costo posible en un único FPGA [73].

Este procesador es altamente configurable ya que permite seleccionar un conjunto específico de características requeridas para diferentes diseños. El conjunto de características fijas del procesador incluye [74]:

- 32 registros de propósito general de 32 bits.
- Instrucciones de 32 bits con 3 operandos y 2 modos de direccionamiento.
- Bus de direcciones de 32 bits.
- Pipeline de sucesión única.

Además de estas características fijas, el procesador MicroBlaze está parametrizado para permitir la selección de funcionalidades adicionales. Para el diseño se utilizó la última versión de MicroBlaze (v7.20) incluida en la herramienta EDK 11.3.

La Tabla A.6 proporciona una visión general de todas las características configurables de algunas versiones de MicroBlaze. La columna sombreada corresponde a la versión utilizada en el diseño del módulo de captura.

Tabla A.6 Características configurables del procesador MicroBlaze (Fuente: [74])

Característica	v7.20
Estados de pipeline	3/5
Interfaz de datos OPB (On-chip Peripheral Bus)	opcional
Interfaz de instrucciones OPB (On-chip Peripheral Bus)	opcional
Interfaz de datos LMB (Local Memory Bus)	opcional
Interfaz de instrucciones LMB (Local Memory Bus)	opcional
Barrel shifter en hardware	opcional
Divider en hardware	opcional
Lógica de depuración en hardware	opcional
Interfaces FSL (Fast Simplex Link)	0-7
Intrucciones set y clear de estado de máquina	opcional
Caché de instrucciones sobre interfaz IXCL (Instruction Xilinx Caché Link)	opcional
Caché de datos sobre interfaz DXCL (Data Xilinx Caché Link)	opcional
Línea caché de 4 ó 8 palabras en XCL (Xilinx Caché Link)	opcional
Soporte de excepciones en hardware	opcional
Instrucciones de comparación de patrones	opcional
Unidad de punto flotante FPU (Floating Point Unit)	opcional
Desactivación del hardware multiplicador	opcional
Lectura de depuración de hardware ESR (Exception Status Register) y EAR (Exception Address Register)	Si
Registro de versión del procesador PVR (Processor Version Register)	opcional
Área o velocidad optimizada	opcional
Multiplicador de hardware con resultado de 64 bits	opcional
Memoria caché LUT	opcional
Interfaz de datos de PLB (Processor Local Bus)	opcional
Interfaz de Instrucciones de PLB (Processor Local Bus)	opcional
Instrucciones de conversión de punto flotante y raíz cuadrada	opcional
Unidad de gestión de memoria MMU (Memory Management Unit)	opcional
Instrucciones de FSL extendida	opcional
Uso de XCL para todos los accesos de memoria de caché de instrucciones	opcional
Uso de XCL para todos los accesos de memoria de caché de datos	opcional
Uso de política "write-back" para el caché de datos	opcional
Protocolo IXCL para caché de instrucciones	opcional
Protocolo DXCL para caché de datos	opcional

La información proporcionada a continuación fue extraída de la referencia [75].

A.3.1 Generalidades de MicroBlaze

MicroBlaze es un procesador virtual que está construido combinando bloques de código llamados “cores” dentro del FPGA. Es un procesador de 32 bits de arquitectura RISC optimizado para la implementación en FPGAs de Xilinx con buses de 32 bits de datos e instrucciones separados para ejecutar programas y acceder a datos desde una memoria externa o interna al mismo tiempo.

Tiene 32 registros de propósito general, un ALU (Arithmetic Logic Unit) y 2 niveles de interrupción. Este diseño fijo puede ser configurado con características más avanzadas para adecuar a las necesidades exactas de una aplicación embebida tales como: barrel

shifter, divider, multiplier, FPU (Floating Point Unit), caches de datos e instrucciones, manejo de excepciones, lógica de depuración, interfaces FSL (Fast Simplex Link) y otras.

A.3.2 Pipeline de MicroBlaze

El procesador MicroBlaze posee pipeline paralelo dividido en 3 estados: Fetch, Decodificación y ejecución. En general cada estado demora un ciclo de reloj para completarse, en consecuencia, el procesador demora 3 ciclos de reloj para que una instrucción sea ejecutada. Cada estado está activo en cada ciclo de reloj por lo tanto se pueden ejecutar simultáneamente 3 instrucciones.

Este procesador implementa un buffer antes del estado fetch que reduce el impacto de la latencia en memoria para instrucciones de múltiples ciclos. Una vez que se reanuda una ejecución el estado fetch puede cargar nuevas instrucciones directamente desde el buffer de instrucciones ubicado antes del estado fetch en vez de tener que esperar la culminación de un acceso a memoria. Este buffer es parte de las características fijas del MicroBlaze y no se permite su configuración.

A.3.3 Interfaces de bus del procesador MicroBlaze

MicroBlaze está organizado como una arquitectura Harvard con interfaces de bus separadas para accesos de datos y accesos de instrucciones. EL procesador no separa entre acceso de datos de entrada/salida y memoria, en su lugar, utiliza memoria mapeada de entrada/salida. Tiene hasta 3 interfaces para accesos a memoria:

- LMB (Local Memory Bus): Proporciona accesos de ciclo único a los bloques de RAM (BRAM) en el chip.
- OPB (On-chip Peripheral Bus) de IBM: Provee conexión para periféricos dentro del chip y fuera del chip y la memoria.
- XCL (Xilinx Caché Link): Está destinado para utilizarse con controladores de memoria externos. Soporta lasta 8 puertos FSL (Fast Simplex Link) el cual es una interfaz punto a punto que conecta aceleradores de hardware desarrollados por el usuario (coprocesadores) al procesador MicroBlaze para acelerar algoritmos de tiempos críticos.

A.3.4 Instrucciones del MicroBlaze

Todas las instrucciones son de 32 bits de ancho y están definidas como:

- Tipo A: Hasta 2 registros fuente y un registro de destino para la operación.
- Tipo B: Un registro fuente, un operando inmediato de 16 bits y un único registro de destino para la operación.

Las instrucciones se clasifican en las siguientes categorías funcionales: aritméticas, lógicas, ramificación y carga/almacenamiento de datos desde/hacia la memoria. Los datos en memoria deben ser traídos al procesador y deben ser ubicados en registros de propósito general para poder realizar cualquier operación.

Las interfaces tanto de instrucciones como de datos son de 32 bits de ancho y utiliza el formato Big-Endian de bits invertidos para representar datos. MicroBlaze soporta accesos a la memoria mediante una palabra de 32 bits, una mitad de palabra de 16 bits y un bytes. Todas las instrucciones deben ser palabras alineadas. Las siguientes figuras muestran la organización de los bits y bytes para cada uno de los 3 tipos de datos mencionados utilizando el formato Big-Endian.

Byte address	n	n+1	n+2	n+3
Byte label	0	1	2	3
Byte significance	MSByte			LSByte
Bit label	0			31
Bit significance	MSBit			LSBit

Figura A.4 Representación de una palabra de 32 bits en MicroBlaze (Fuente: [74])

Byte address	n	n+1
Byte label	0	1
Byte significance	MSByte	LSByte
Bit label	0	15
Bit significance	MSBit	LSBit

Figura A.5 Representación de media palabra de 16 bits en MicroBlaze (Fuente: Ibídem)

Byte address	n	
Bit label	0	7
Bit significance	MSBit	LSBit

Figura A.6 Representación de un byte de 8 bits en MicroBlaze (Fuente: Ibídem)

A.3.5 Registros de propósito especial del MicroBlaze

El procesador MicroBlaze tiene registros de propósito especial tales como:

- Contador de programa (PC Program Counter): Se puede leer pero no se puede escribir en él.

- Registro de estado de máquina (MSR Machine Status Register): Sirve para indicar los estados del procesador como por ejemplo la indicación del carry aritmético, un error de división por cero, un error del FSL (Fast Simplex Link), habilitar o deshabilitar interrupciones, etc.

- Registro de dirección de excepción (EAR Exception Address Register): Almacena la dirección de carga/almacenamiento que causa una excepción.

- Registro de estado de excepción (ESR Exception Status Register): Indica el tipo de excepción ocurrida.
- Registro de estado del punto flotante (FSR Floating Point Unit): Indica eventos en una operación de punto flotante tales como una operación inválida, error de división por cero, overflow, underflow o error de operando sin normalizar.

A.3.6 Interrupciones del procesador MicroBlaze

El procesador MicroBlaze soporta únicamente interrupciones de fuentes externas, de aquellas conectadas al puerto de entrada de interrupciones. Si se requiere múltiples interrupciones, se debe utilizar un controlador de interrupciones para manejar los requerimientos de múltiples interrupciones al procesador.

Se tiene un controlador de interrupciones disponible para su uso en la herramienta de software EDK (Embedded Development Kit). El procesador reacciona a interrupciones solamente si el bit IE (Interrupt Enable) en el registro de estado de máquina MSR (Machine Status Register) está establecido a 1.

Durante una interrupción, la instrucción en el estado de ejecución se va a completar, mientras la instrucción en el estado de decodificación se sustituye por una rama en el vector de interrupción (dirección 0x10).

La dirección de retorno de interrupción es cargada automáticamente en el registro de propósito general R14. Además, el procesador deshabilita interrupciones futuras estableciendo a 0 el bit IE en el MSR. Este bit es automáticamente de nuevo establecido a 1 cuando se ejecuta la instrucción RTID (Return form Interrupt). Esta información fue extraída de la referencia [74]. Mayores detalles sobre las instrucciones relacionadas a las interrupciones se encuentran en dicha referencia.

A.3.7 Software de diseño

El software que controla el procesador MicroBlaze debe ser escrito en lenguaje C/C++, el cual, es el preferido por muchos diseñadores y además es el formato que se utiliza en la herramienta de software Xilinx EDK (Embedded Development kit). En esta herramienta se tiene incorporado compiladores para generar el código de máquina necesario para el funcionamiento del procesador.

El procesador es inservible por sí mismo sin dispositivos periféricos conectados. EDK contiene un gran número de periféricos comúnmente utilizados. Se puede crear diferentes tipos de sistemas con estos periféricos incluso el diseñador puede crear sus propios diseños personalizados para implementar una funcionalidad que no está disponible en las bibliotecas de periféricos de la herramienta EDK y poder incorporar dicho diseño en el sistema embebido basado en el procesador MicroBlaze. Los detalles sobre cómo utilizar la herramienta EDK se encuentra en la referencia [76].

A.4 Hardware MAC Ethernet embebido de 3 modos.

Toda la información proporcionada en esta sección fue extraída de la referencia [77]. La familia de dispositivos de FPGA Virtex 5 contiene pares de MACs Ethernet embebidos que son configurables de manera independiente para satisfacer todas las necesidades comunes de conectividad a redes Ethernet.

a. Características principales del MAC Ethernet

Este hardware presenta las siguientes características:

- MAC Ethernet de 10/100/1000 Mb/s totalmente integrado en el FPGA Virtex 5.
- Diseñado para la especificación IEEE 802.3-2002.
- Operación full-duplex configurable en 10/100/1000 Mb/s.
- Operación half-duplex configurable en 10/100 Mb/s.
- Interfaz MDIO (Management Data Input/Output) para manejar objetos en la capa física.
- Salidas de vector estadístico accesible por el usuario.
- Soporte de tramas VLAN (Virtual Local Area Network).
- Espacio entre tramas (IFG Interframe gap) configurable en operación full-duplex.
- Campo FCS (Frame Check Sequence) configurable tanto en transmisión como en recepción.
- Auto relleno de cabecera Ethernet en transmisión y extracción del relleno en la recepción.
- Es configurado y monitoreado mediante la interfaz host.
- Interfaz de bus seleccionable por hardware ya sea el bus DCR (Device Control Register) o el bus de host genérico.
- Control de flujo configurable mediante tramas de control PAUSE generadas por el MAC Ethernet. Se habilita de modo simétrico o asimétrico.
- Soporte configurable de tramas Jumbo de cualquier tamaño.
- Filtro de recepción de direcciones configurable para direcciones generales, unicast y broadcast.
- Soporte de la interfaz MII (Media Independent Interface), GMII (Gigabit MII) y RGMII (Reduced GMII).
- Sub capas de capa física PCS (Physical Coding Sublayer) y PMA (Physical Medium Attachment) incluidas para su uso con transceptores seriales RocketIO para proporcionar una completa implementación del estándar 1000BASE-X.
- Soporte de la interfaz SGMII (Serial GMII) mediante las interfaces de los transceptores seriales RocketIO y la capa física externa de cobre para la operación en full-duplex.

La Figura A.7 muestra la relación entre el modelo de referencia OSI y el MAC Ethernet embebido según la especificación IEEE 802.3. Las capas en color gris muestran la funcionalidad implementada en el MAC Ethernet. Asimismo, se muestra la ubicación en la

arquitectura de las interfaces físicas soportadas.

La sub capa MAC es responsable de los protocolos de enmarcado Ethernet y de la detección de errores de estas tramas. La MAC es independiente del dispositivo de capa física y puede conectarse a cualquier tipo de estos dispositivos. La sub capa MAC Control proporciona manipulación de control de flujo en tiempo real de la sub capa MAC. El MAC Ethernet proporciona ambas sub capas en todos los modos de operación.

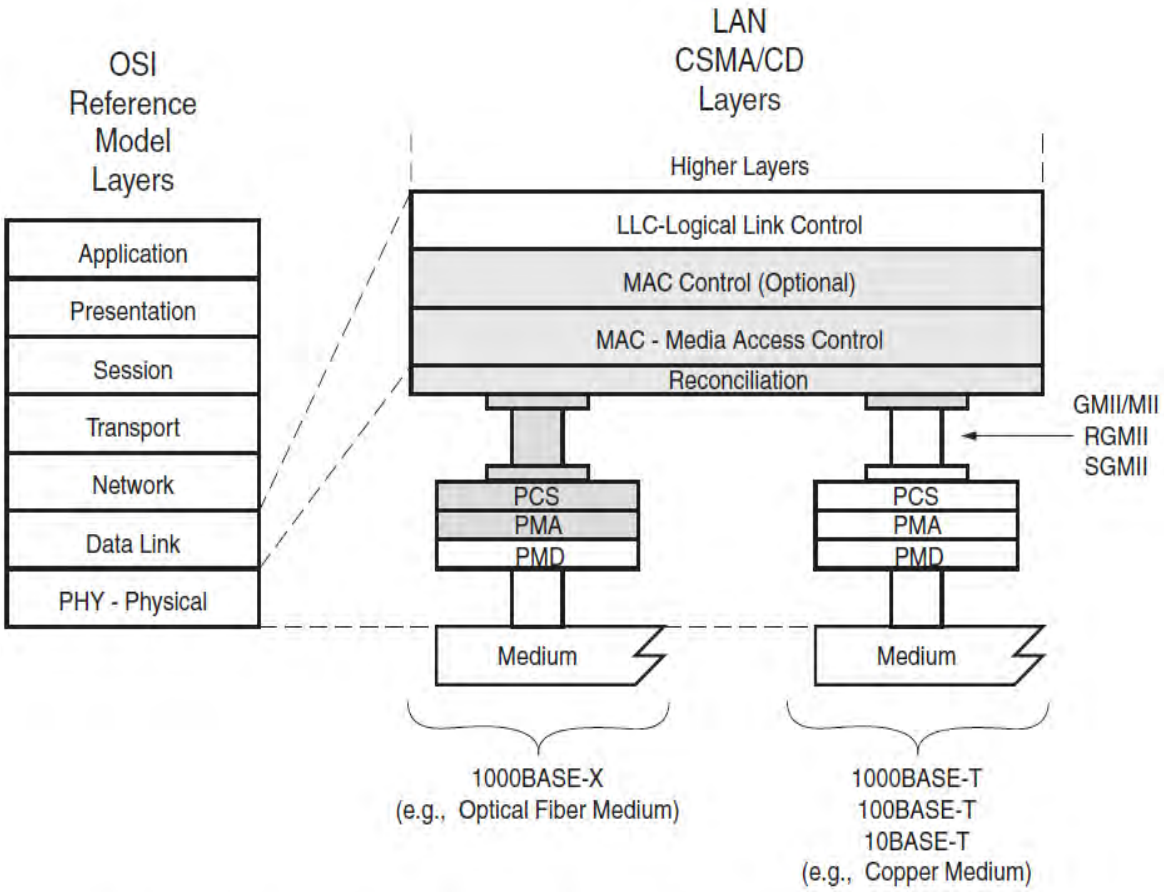


Figura A.7 Modelo de la especificación IEEE 802.3-2002 (Fuente: [77])

La capa física está compuesta por la combinación de las sub capas PCS (Physical Coding Sublayer), PMA (Physical Medium Attachment) y PMD (Physical Medium Dependent). El MAC Ethernet soporta la conexión con los siguientes tipos de interfaces físicas:

- BASE-T: Proporciona un enlace entre el MAC y los medios de cobre. Esta funcionalidad no está implementada en el MAC Ethernet de 3 modos que está embebida en el FPGA Virtex 5. Sin embargo, se encuentra disponible en el mercado dispositivos de capa física BASE-T que pueden conectarse al MAC Ethernet utilizando interfaces GMII/MII, RGMII ó SGMII.
- BASE-X: Proporciona un enlace entre el MAC y los medios de fibra óptica. El MAC Ethernet de 3 modos que está embebida en el FPGA Virtex 5 soporta el estándar BASE-X

de 1 Gbps. Las sub capas PCS y PMA del estándar 1000BASE-X se puede utilizar conectando el MAC Ethernet a un transceptor serial RocketIO. Se puede conectar un transceptor óptico directamente al transceptor serial RocketIO para completar la funcionalidad de la sub capa PMD.

b. Características Generales del estándar IEEE 802.3-2002

En la Figura A.8 se muestra un dato Ethernet encapsulado en una trama. Los campos y los bytes dentro de la trama son transmitidos de izquierda a derecha. El MAC Ethernet puede manejar tramas Jumbo Ethernet donde el campo de datos puede ser mucho más grande en tamaño que 1500 bytes.

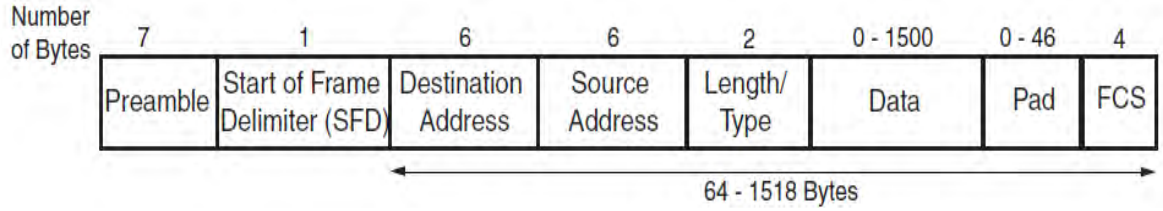


Figura A.8 Formato de una trama Ethernet estándar (Fuente: [77])

El MAC Ethernet también acepta tramas de una VLAN (Virtual Local Area Network). El formato de una trama VLAN es mostrado en la Figura A.9. Como se observa si la trama es de tipo VLAN, el MAC Ethernet acepta 4 bytes adicionales.

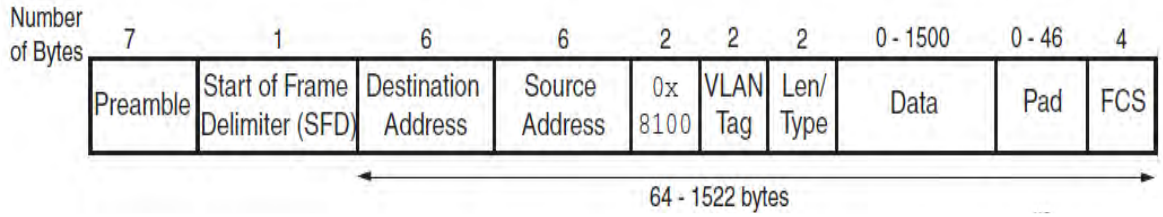


Figura A.9 Formato de una trama Ethernet VLAN (Fuente: [77])

El MAC Ethernet puede enviar o recibir tramas Ethernet PAUSE de control de flujo. El formato de esta trama se diferencia de una trama Ethernet estándar tal como se muestra en la Figura A.10.

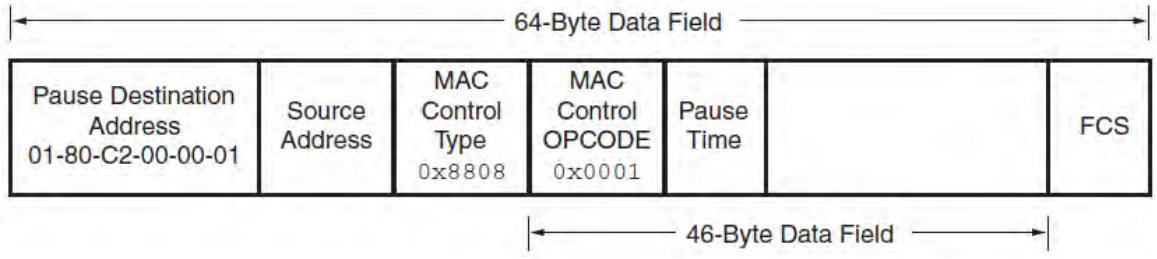


Figura B.10 Ejemplo de una trama Ethernet PAUSE (Fuente: [77])

Se explica en la Tabla B.7 la función de cada uno de los campos en una trama Ethernet.

Tabla A.7 Descripción de los campos de una trama Ethernet (Fuente: [1])

Campo	Descripción

Preamble (Preámbulo)	Para una transmisión este campo es insertado automáticamente por el MAC Ethernet. Contiene 7 bytes con el patrón 0x55 transmitidos de izquierda a derecha. Este campo es utilizado para sincronización. El MAC Ethernet despoja este campo durante la recepción de un paquete y también puede recibir tramas sin este campo siempre que el campo “Start of Frame Delimiter” esté disponible.
Start of Frame Delimiter (SDF)	Este campo marca el inicio de una trama y debe contener el patrón 0xD5. Para la transmisión, este campo es insertado automáticamente por el MAC Ethernet. Para la recepción, este campo es despojado de la trama entrante.
Destination Address	El MAC Ethernet soporta la transmisión y recepción de paquetes unicast, multicast y broadcast. El bit menos significativo de este campo determina si la dirección es unicast (0) o si es multicast (1). Una dirección broadcast corresponde cuando todos los bits de este campo son 1.
Source Address	Este campo debe ser proporcionado por el programa cliente. La dirección unicast para el MAC Ethernet es utilizada como dirección fuente cuando el MAC Ethernet crea una trama PAUSE.
Length/Type	El valor de este campo determina si debe ser interpretado como "length" (longitud) o como "type" (tipo) tal como se define en el estándar IEEE 802.3. Un valor decimal mayor o igual que 1536=0x0600 es interpretado como un campo "tipo" por el MAC Ethernet.
	El valor en este campo representa el número de bytes en el campo "data" (sin incluir el campo "pad") cuando se usa como campo "length".
	Un valor de 0x8100 en este campo indica que esta trama es de tipo VLAN y un valor de 0x8808 indica que es una trama Ethernet PAUSE.
Data	Este campo puede variar desde 0 hasta 1500 bytes en longitud para una trama normal. El MAC Ethernet puede manejar tramas Jumbo de cualquier longitud.
Pad	Este campo puede varias desde 0 hasta 46 bytes en longitud. Este campo es utilizado para asegurar que la longitud de la trama sea de al menos 64 bytes en longitud (sin considerar el campo preámbulo ni el SDF), el cual se requiere para una operación exitosa del método de acceso CSMA/CD. Los valores en este campo son utilizados para realizar el cálculo del campo FCS pero no son incluidos en el campo "length" cuando es utilizado. La longitud de este campo junto con el campo "Data" debe ser de al menos 46 bytes. Si la longitud del campo "Data" es 0, este campo tiene 46 bytes. Si el campo "Data" tiene 46 bytes o más de longitud, este campo tiene 0 bytes.
Frame Check Sequence (FCS)	El valor de este campo es calculado sobre los campos "destination address, "source address", "length/type", "data" y "pad" utilizando CRC (Cyclic Redundancy Check) de 32 bits tal como se define en el estándar IEEE 802,3-2002. Para una transmisión, este campo es insertado ya sea de manera automática por el MAC Ethernet o por el cliente. Para la recepción, el valor de este campo es verificado en cada trama entrante. Si se recibe un valor incorrecto en este campo, el MAC Ethernet le indica al cliente que se recibió una trama incorrecta.

Las tramas son transmitidas sobre el medio con un gap (espacio de tiempo) entre tramas de 96 veces el tiempo de un bit (9.6 µs para 10 Mb/s, 0.96 µs para 100 Mb/s y 96 ns para 1 Gbps) tal como se especifica en el estándar IEEE 802.3. Estos valores son

mínimos y pueden incrementarse como resultado de una disminución del rendimiento. Los procesos de transmisión de tramas son diferentes tanto para sistemas half-dúplex como para full-dúplex.

En un sistema half-duplex el proceso es como se indica a continuación:

- El MAC Ethernet monitorea el medio vigilando la señal portadora CRS (Carrier Sense Signal) desde el dispositivo de capa física externo, incluso si es que no se tiene nada para transmitir. Siempre que el medio esté ocupado (CRS=1), el MAC Ethernet posterga la transmisión de la trama retardando cualquier transmisión pendiente.
- Después del último bit de la trama en curso (cambio de la señal CRS de 1 a 0), el MAC Ethernet inicia la temporización del intervalo entre tramas (gap).
- El MAC Ethernet reinicia el temporizador del intervalo entre tramas si la señal portadora CRS pasa a 1 (medio ocupado) durante el periodo definido como IFG1 (Interframe gap part 1). El estándar IEEE 802.3 establece que el periodo IFG1 debe ser 2/3 del intervalo de temporización del intervalo entre tramas (64 veces el tiempo de un bit) pero puede ser más pequeño y tan mínimo como 0. El propósito de esta opción es brindar soporte a una posible breve falla de la señal portadora durante una colisión.
- El Ethernet MAC no reinicia el temporizador del intervalo entre tramas si la señal portadora pasa a 1 durante el periodo definido como IFG2 (Interframe gap part 2) para asegurar un acceso equitativo al bus. El estándar IEEE 802.3 establece que el periodo IFG2 debe ser el último 1/3 del intervalo de temporización del intervalo entre tramas.

Si después de iniciada una transmisión, se produce una colisión con el mensaje de otra estación (COL=1), cada estación continua transmitiendo por un periodo de tiempo adicional predefinido (tiempo de 32 bits para 10/100 Mb/s) para asegurar la propagación de la colisión en todo el sistema. La estación queda en silencio por una cantidad de tiempo aleatoria (backoff) antes de intentar transmitir de nuevo.

Una estación puede experimentar una colisión durante el inicio de su transmisión (dentro de la ventana de colisión) antes de que su transmisión haya tenido tiempo de ser propagada a todas las estaciones en el bus. Después de que la ventana de colisión haya pasado, otra estación transmisora puede adquirir el bus sin problemas. De este modo, se evitan posteriores colisiones (después de haberse terminado la ventana de colisión) porque se asume que todas las demás estaciones han detectado la transmisión.

En un sistema full-dúplex existe una conexión punto a punto dedicada entre 2 dispositivos Ethernet capaces de transmitir y recibir simultáneamente sin posibilidad de colisiones. En este caso el MAC Ethernet no utiliza la señal portadora CRS (Carrier sense signal) desde el dispositivo de capa física externo porque el medio no es compartido y el MAC Ethernet solamente necesita monitorear su propia transmisión. Después de que el

último bit de una trama Ethernet es transmitido, el MAC Ethernet inicia el temporizador del intervalo entre tramas y posterga cualquier transmisión hasta que el temporizador alcance 96 veces el tiempo de un bit.

c. Arquitectura del MAC Ethernet

La Figura B.11 muestra un diagrama de bloques del MAC Ethernet embebido en un FPGA Virtex 5.

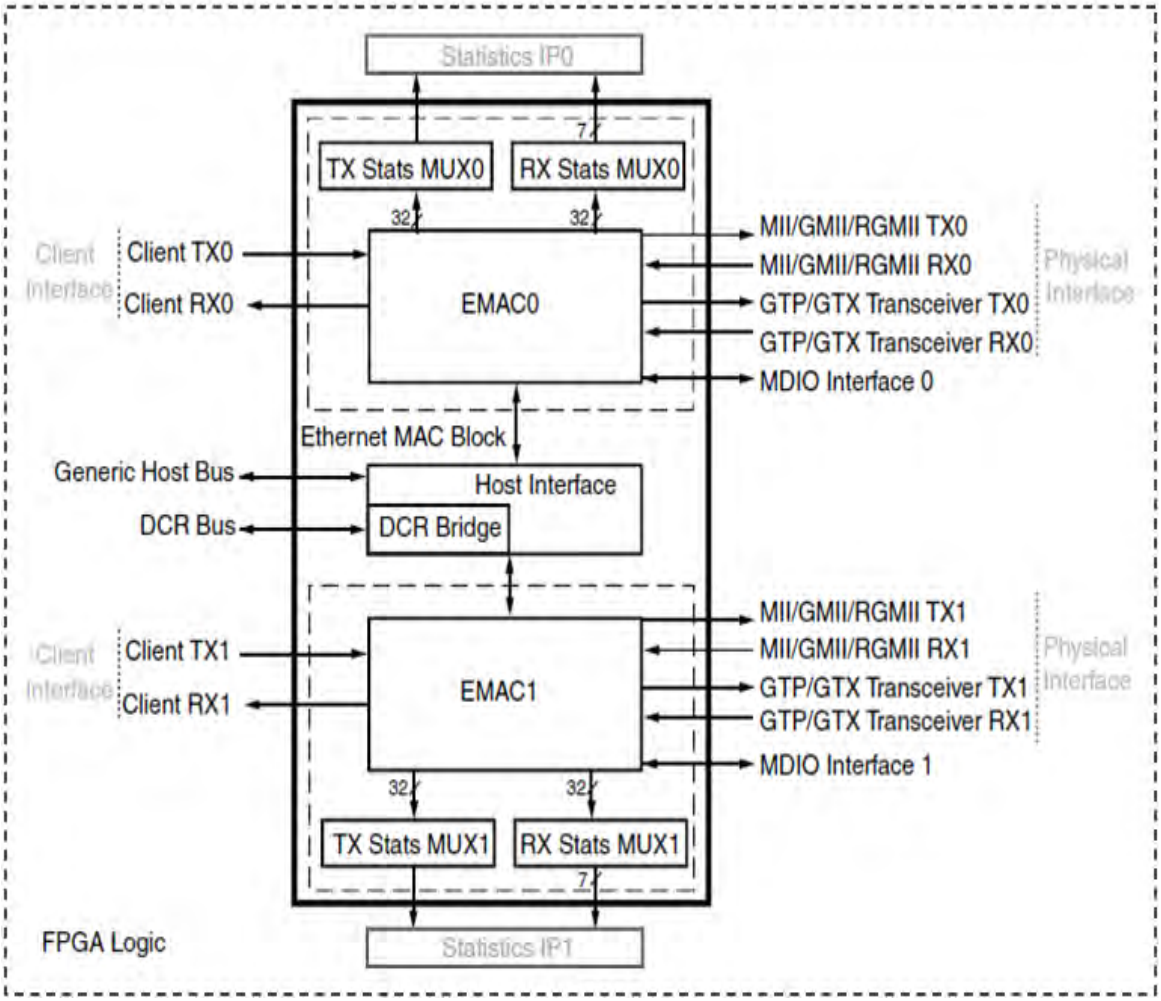


Figura B.11 MAC Ethernet embebido (Fuente: [77])

El MAC Ethernet embebido contiene 2 bloques iguales MAC Ethernet compartiendo una interfaz de host única. Cada bloque MAC Ethernet cuenta con un camino de datos de transmisión y recepción independientes con capacidad full-dúplex.

La interfaz de host proporciona el acceso a los registros de configuración de ambos MACs Ethernet. Se puede acceder a esta interfaz utilizando ya sea el bus de host genérico o el bus DCR (Device Control Register) mediante el puente DCR.

La interfaz física de cada MAC Ethernet puede ser configurada como MII, GMII, RGMII, SGMII ó 1000BASE-X. La Figura A.11 muestra todos los puertos posibles para esta interfaz. Cada MAC Ethernet puede ser configurado con diferentes interfaces físicas.

Cada MAC Ethernet tiene opcionalmente una interfaz MDIO (Management Data I/O) que permite el acceso a los registros de administración de un dispositivo de capa física externo o el acceso a los registros de administración de la interfaz física dentro del MAC Ethernet (solamente cuando está configurado en los modos 1000BASE-X ó SGMII).

Cada MAC Ethernet arroja vectores estadísticos que contienen información sobre las tramas Ethernet vistas durante la transmisión o recepción. Se puede diseñar e implementar los módulos externos “Statistics IP0” y/o “Statistics IP1” en el FPGA para acumular todas las estadísticas tanto de transmisión como de recepción de cada uno.

La Figura A.12 muestra un diagrama de bloques de un único MAC Ethernet representado por las siglas EMAC# (Ethernet MAC #). El símbolo “#” indica que se puede tratar tanto del primer bloque (número 0) como del segundo bloque (número 1).

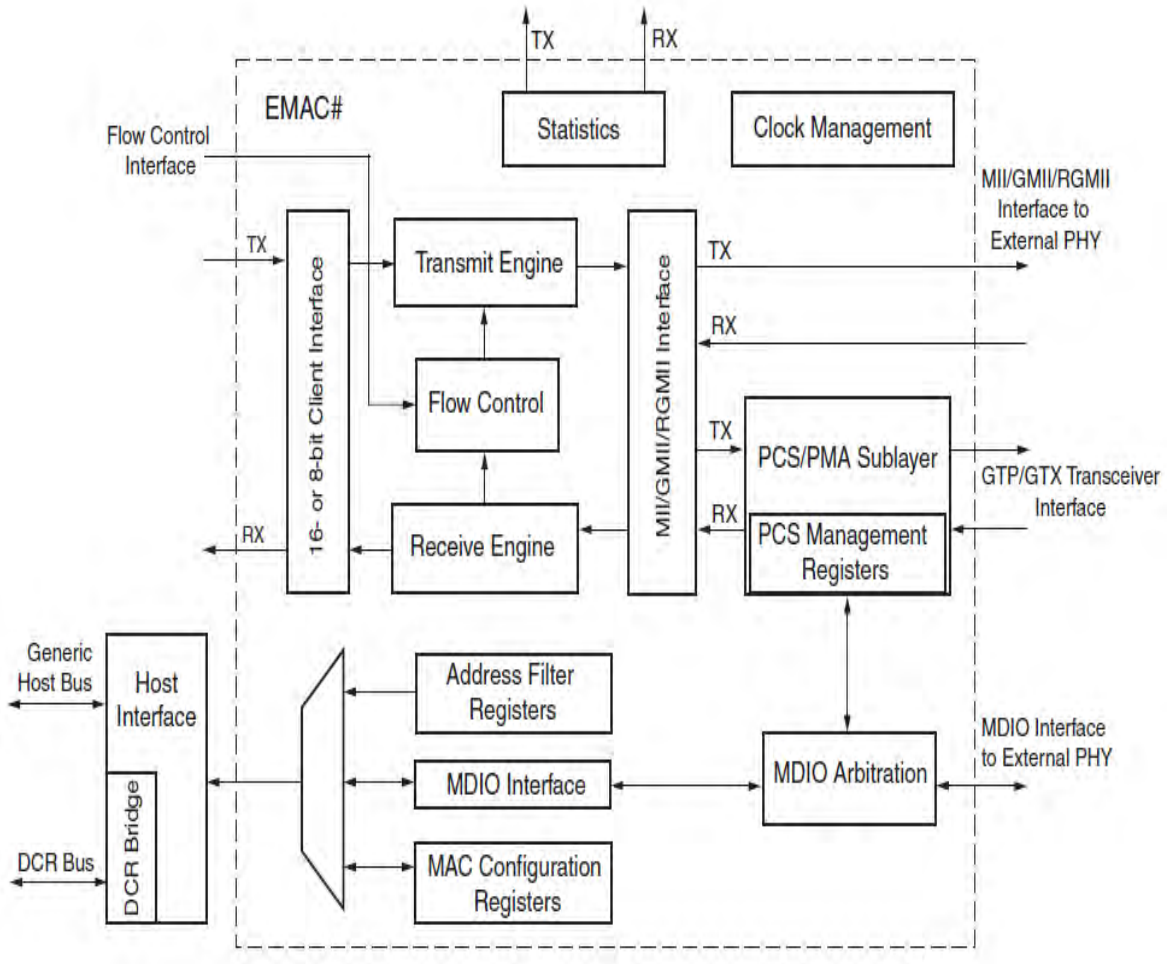


Figura A.12 Diagrama de bloques de un único MAC Ethernet embebido ([77])

La interfaz del cliente conecta las interfaces de usuario de transmisión y recepción a las máquinas de transmisión y recepción del MAC Ethernet respectivamente. La interfaz de control de flujo permite que la lógica del cliente pueda forzar a la capa física a suspender el envío de tramas hasta que el cliente sea capaz de aceptar más. El filtro de direcciones

permite aceptar o rechazar tramas entrantes en el camino de datos de recepción. La interfaz de host compartida permite tener acceso a los registros de configuración del MAC y a los registros de filtro de direcciones.

La interfaz de host puede también iniciar transacciones mediante la interfaz MDIO para tener acceso a los registros de administración de la sub capa PCS dentro del MAC Ethernet cuando está configurado en los modos 1000BASE-X o SGMII. En este mismo modo de configuración, es posible que un dispositivo externo también tenga acceso a los registros de administración de la sub capa PCS dentro del MAC Ethernet cuando se tiene habilitada la interfaz MDIO y deshabilitada la interfaz de host. La interfaz de host también puede iniciar transacciones mediante la interfaz MDIO para acceder a un dispositivo de capa física externo.

Cuando la interfaz física del MAC Ethernet está configurada en los modos MII, GMII ó RGMII las señales de salida se conectan a un dispositivo de capa física externo. Cuando la interfaz física está configurada en los modos 1000BASE-X o SGMII, el bloque de las sub capas PCS/PMA es habilitado y sus interfaces son directamente conectadas a un transceptor serial RocketIO.

El módulo de gestión de reloj configura automáticamente la salida de los relojes a la frecuencia correcta basado en la velocidad interna del MAC Ethernet (10 Mb/s, 100 Mb/s ó 1000 Mb/s) y la configuración del modo (GMII, MII, RGMII, SGMII y 1000BASE-X).

Para mayores detalles sobre cada uno de los bloques y otros detalles adicionales se deben consultar los respectivos capítulos de la referencia [77].

ANEXO B
REPORTES DEL DISEÑO DEL MÓDULO DE CAPTURA GENERADOS POR LA HERRAMIENTA XPS

ANEXO B REPORTES DEL DISEÑO DEL MÓDULO DE CAPTURA GENERADOS POR LA HERRAMIENTA XPS

En este anexo se muestran los reportes del diseño del módulo de captura generados por la herramienta XPS, organizados en 7 tablas (B.2 a B.8):

La Tabla B.1 resume el significado de los campos de las tablas referidas.

Tabla B.1 Significado de los campos de las tablas

Campos de Tabla "Estado del proyecto"	
Project File	Indica el archivo principal del proyecto del sistema hardware.
Module Name	Indica el nombre del módulo el cual en este caso es el nombre por defecto "system".
Product Version	Indica la versión de la herramienta ED
Implementation State	Indica el estado actual del proyecto. Como el proyecto ya fue terminado, se indica que ya se ha generado el archivo de programa del sistema.
Campos de Tabla "Archivos de reportes"	
Report Name	Indica los nombres los archivos de reporte de cada uno de los procesos de diseño generados por la herramienta XPS.
Generated	Indica la fecha y hora de generación de cada reporte.
Errors	Indica el número de errores en cada archivo de reporte.
Warnings	Indica en número de mensajes de advertencias en cada archivo de reporte.
Infos	Indica el número de anotaciones que genera XPS con respecto a cada reporte.
Campos de Tabla "Resumen del proceso de síntesis"	
Report	Indica los nombres de los archivos de reporte de cada uno de los elementos de hardware del diseño generados por la herramienta XPS durante el proceso de síntesis.
Generated	Indica la fecha y hora de generación de cada reporte.
Flip Flops Used	Indica el número de flip flops utilizados en la construcción del respectivo elemento hardware.
LUTs Used	Indica el número de LUTs (Look up tables) utilizados en la construcción del respectivo elemento hardware.
BRAMS Used	Indica el número de BRAMS (bloques de memoria RAM) utilizados en la construcción del respectivo elemento hardware.
Errors	Indica en número de errores de síntesis en cada archivo de reporte.
Campos de Tabla "Resumen de la utilización de recursos del FPGA"	
Slice Logic Utilization	Indica los nombres de cada recurso utilizado del FPGA como los registros, LUTs, IOBs (bloques de entrada/salida), y BRAMS (Bloques de RAM). Asimismo, se indica en detalle los diferentes usos que se ha dado a cada uno de estos elementos y el uso de los elementos internos de cada recurso.
Total Memory used	Indica los nombres de cada elemento de memoria utilizado del

(KB)	FPGA, Asimismo, se indica en detalle los diferentes usos que se ha dado a cada uno de estos elementos.
Used	Indica el número de elementos utilizados de cada recurso.
Available	Indica el número total de elementos disponibles en el FPGA de cada recurso.
Utilization	Indica el porcentaje de utilización del total de elementos disponibles en el FPGA de cada recurso.
Notes	Indica algunas anotaciones que genera XPS con respecto a cada recurso del FPGA.
Campos de Tabla "Lista de reportes relacionados al rendimiento del sistema"	
Final Timing Score	Indica el resumen general de los tiempos de Setup, Hold y Component Switching Limit.
Routing results	Indica un enlace hacia el reporte del proceso de ruteo en la cual se muestran las todas las señales que fueron ruteadas.
Timing Constraints	Indica un enlace hacia el reporte que muestra todas restricciones de tiempo del sistema.
Pinout Data	Indica un enlace hacia el reporte que detalla la función de cada uno de los pines del FPGA.
Clock Data	Indica un enlace hacia el reporte donde se detallan todas las señales de reloj del sistema.
Campos de Tabla "Reporte del uso de recursos de reloj del sistema"	
Clock Net	Indica las señales de reloj de cada recurso de reloj utilizado por cada elemento del sistema.
Resource	Indica el nombre de cada recurso de reloj utilizado por cada elemento del sistema.
Locked	Indica si el recurso de reloj está bloqueado o no.
Fanout	Indica el número de fan-out (número máximo posible de entradas a puertos) del respectivo recurso de reloj.
Net Skew (ns)	Indica el tiempo de skew (máxima diferencia de retardo entre la fuente de reloj y el recurso de reloj del elemento) en nanosegundos de la respectiva señal de reloj.
Max Delay (ns)"	Indica el máximo retardo de propagación (retardo en atravesar un circuito lógico más retardo en la ruta de la señal) en nanosegundos de la respectiva señal de reloj.
Campos de Tabla "Lista de reportes detallados del diseño"	
Report Name	Indica el nombre de cada reporte.
Status	Indica si el reporte pertenece al proyecto actual.
Generated	Indica la fecha y hora de generación de cada reporte.
Errors	Indica el número de errores en cada archivo de reporte.
Warnings	Indica en número de mensajes de advertencias en cada archivo de reporte.
Infos	Indica el número de anotaciones que genera XPS con respecto a cada reporte.

Tabla B.2 Estado del proyecto

Project Status (04/12/2012 - 17:26:31)			
Project File:	system.xmp	**Implementation State:**	Programming File Generated
Module Name:	system	• **Errors:**	
Product Version:	EDK 11.3	• **Warnings:**	

Tabla B.3 Archivos de reportes

XPS Reports				
Report Name	**Generated**	**Errors**	**Warnings**	**Infos**
Platgen Log File	mié 7. mar 14:47:57 2012	0	19 Warnings (1 new, 0 filtered)	168 Infos (108 new, 0 filtered)
Libgen Log File	mié 7. mar 15:04:44 2012	0	0	1 Info (1 new, 0 filtered)
Simgen Log File				
BitInit Log File	mié 7. mar 15:04:50 2012	0	0	39 Infos (39 new, 0 filtered)
System Log File	mar 3. abr 17:51:56 2012			

Tabla B.4 Resumen del proceso de síntesis

XPS Synthesis Summary					
Report	**Generated**	**Flip Flops Used**	**LUTs Used**	**BRAMS Used**	**Errors**
system	mié 7. mar 14:49:02 2012	10083	9430	34	0
xps_intc_0_wrapper	mié 7. mar 14:47:27 2012	166	118		0
proc_sys_reset_0_wrapper	mié 7. mar 14:47:13 2012	67	51		0
mdm_0_wrapper	mié 7. mar 14:47:07 2012	119	112		0
clock_generator_0_wrapper	mié 7. mar 14:46:58 2012		1		0
xps_timer_0_wrapper	mié 7. mar 14:46:50 2012	358	287		0
hard_ethernet_mac_wrapper	mié 7. mar 14:46:33 2012	1292	1110		0
hard_ethernet_mac_wrapper_fifo_generator_v4_3_3_fifo_generator_v4_3_xst_1	mié 7. mar 14:46:12 2012	123	283		0
hard_ethernet_mac_wrapper_blk_mem_gen_v2_7_blk_mem_gen_v2_7_xst_1	mié 7. mar 14:45:57 2012	4	154		0
hard_ethernet_mac_wrapper_fifo_generator_v4_3_4_fifo_generator_v4_3_xst_1	mié 7. mar 14:45:35 2012	71	44		0
hard_ethernet_mac_wrapper_fifo_generator_v4_3_2_fifo_generator_v4_3_xst_1	mié 7. mar	85	208		0

	14:45:21 2012				
hard_ethernet_mac_wrapper_fifo_generator_v4_3_1_fifo_generator_v4_3_xst_1	mié 7. mar 14:44:56 2012	51	84		0
ddr2_sdram_wrapper	mié 7. mar 14:44:15 2012	4982	3934	11	0
rs232_uart_1_wrapper	mié 7. mar 14:40:04 2012	147	130		0
lmb_bram_wrapper	mié 7. mar 14:39:48 2012			2	0
ilmb_cntlr_wrapper	mié 7. mar 14:39:41 2012	2	6		0
dlmb_cntlr_wrapper	mié 7. mar 14:39:36 2012	2	6		0
dlmb_wrapper	mié 7. mar 14:39:30 2012	1	1		0
ilmb_wrapper	mié 7. mar 14:39:24 2012	1	1		0
mb_plb_wrapper	mié 7. mar 14:39:18 2012	153	342		0
microblaze_0_wrapper	mié 7. mar 14:39:01 2012	2459	2558	4	0

Tabla B.5 Resumen de la utilización de recursos del FPGA

Device Utilization Summary				
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	9,360	28,800	32%	
Number used as Flip Flops	9,356			
Number used as Latch-thrus	4			
Number of Slice LUTs	8,794	28,800	30%	
Number used as logic	8,354	28,800	29%	
Number using O6 output only	7,560			
Number using O5 output only	273			
Number using O5 and O6	521			
Number used as Memory	403	7,680	5%	
Number used as Dual Port RAM	248			
Number using O6 output only	12			
Number using O5 output only	36			
Number using O5 and O6	200			
Number used as Single Port RAM	32			
Number using O6 output only	32			
Number used as Shift Register	123			
Number using O6 output only	122			
Number using O5 output only	1			
Number used as exclusive route-thru	37			

Number of route-thrus	331			
Number using O6 output only	306			
Number using O5 output only	23			
Number using O5 and O6	2			
Number of occupied Slices	4,660	7,200	64%	
Number of LUT Flip Flop pairs used	13,005			
Number with an unused Flip Flop	3,645	13,005	28%	
Number with an unused LUT	4,211	13,005	32%	
Number of fully used LUT-FF pairs	5,149	13,005	39%	
Number of unique control sets	1,116			
Number of slice register sites lost to control set restrictions	2,573	28,800	8%	
Number of bonded IOBs	147	480	30%	
Number of LOCed IOBs	147	147	100%	
IOB Flip Flops	292			
Number of BlockRAM/FIFO	44	60	73%	
Number using BlockRAM only	44			
Number of 36k BlockRAM used	27			
Number of 18k BlockRAM used	34			
Total Memory used (KB)	1,584	2,160	73%	
Number of BUFG/BUFGCTRLs	8	32	25%	
Number used as BUFGs	7			
Number used as BUFGCTRLs	1			

Number of IDELAYCTRLs	5	16	31%	
Number of BSCANs	1	4	25%	
Number of BUFIOs	8	56	14%	
Number of DSP48Es	5	48	10%	
Number of PLL_ADVs	1	6	16%	
Number of TEMACs	1	2	50%	
Average Fanout of Non-Clock Nets	3.63			

Tabla B.6 Lista de reportes relacionados al rendimiento del sistema

Performance Summary			
Final Timing Score:	0 (Setup: 0, Hold: 0, Component Switching Limit: 0)	**Pinout Data:**	Pinout Report
Routing Results:	All Signals Completely Routed	**Clock Data:**	Clock Report
Timing Constraints:	All Constraints Met		

Tabla B.7 Reporte del uso de recursos de reloj del sistema

Clock Report					
Clock Net	**Resource**	**Locked**	**Fanout**	**Net Skew(ns)**	**Max Delay(ns)**
clk_125_0000MHzPLL0	BUFGCTRL_X0Y0	No	3431	0.426	1.934
clk_62_5000MHzPLL0	BUFGCTRL_X0Y3	No	460	0.359	1.869
Hard_Ethernet_MAC/Tx					
ClientClk_0	BUFGCTRL_X0Y31	No	39	0.320	1.908
clk_125_0000MHz90PLL					
0	BUFGCTRL_X0Y4	No	161	0.231	1.841
Hard_Ethernet_MAC/Rx					
ClientClk_0	BUFGCTRL_X0Y30	No	134	0.287	1.912
mdm_0/Dbg_Clk_1	BUFGCTRL_X0Y29	No	67	0.221	1.730
clk_200_0000MHz	BUFGCTRL_X0Y1	No	8	0.187	1.692
DDR2_SDRAM/DDR2_SDRA					
M/mpmc_core_0/gen_v5					
_ddr2_phy.mpmc_phy_i					
f_0/u_phy_io_0/delay					
ed_dqs<0>	IO Clk	No	18	0.095	0.419
DDR2_SDRAM/DDR2_SDRA					
M/mpmc_core_0/gen_v5					
_ddr2_phy.mpmc_phy_i					
f_0/u_phy_io_0/delay					

ed_dqs<1>	IO Clk	No	18	0.083	0.380
DDR2_SDRAM/DDR2_SDRA					
M/mpmc_core_0/gen_v5					
_ddr2_phy.mpmc_phy_i					
f_0/u_phy_io_0/delay					
ed_dqs<2>	IO Clk	No	18	0.101	0.425
DDR2_SDRAM/DDR2_SDRA					
M/mpmc_core_0/gen_v5					
_ddr2_phy.mpmc_phy_i					
f_0/u_phy_io_0/delay					
ed_dqs<3>	IO Clk	No	18	0.107	0.404
DDR2_SDRAM/DDR2_SDRA					
M/mpmc_core_0/gen_v5					
_ddr2_phy.mpmc_phy_i					
f_0/u_phy_io_0/delay					
ed_dqs<5>	IO Clk	No	18	0.101	0.425
DDR2_SDRAM/DDR2_SDRA					
M/mpmc_core_0/gen_v5					
_ddr2_phy.mpmc_phy_i					
f_0/u_phy_io_0/delay					
ed_dqs<4>	IO Clk	No	18	0.101	0.425
DDR2_SDRAM/DDR2_SDRA					
M/mpmc_core_0/gen_v5					

_ddr2_phy.mpmc_phy_i					
f_0/u_phy_io_0/delay					
ed_dqs<6>	IO Clk	No	18	0.096	0.393
DDR2_SDRAM/DDR2_SDRA					
M/mpmc_core_0/gen_v5					
_ddr2_phy.mpmc_phy_i					
f_0/u_phy_io_0/delay					
ed_dqs<7>	IO Clk	No	18	0.101	0.425
mdm_0/Dbg_Update_1	Local		19	2.253	2.974
RS232_Uart_1_Interru					
pt	Local		1	0.000	0.479
fpga_0_Hard_Ethernet					
_MAC_MII_TX_CLK_0_pi					
n_IBUFG	Local		2	0.000	3.288

Tabla B.8 Lista de reportes detallados del diseño

Detailed Reports					
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Translation Report	Current	mié 7. mar 14:49:53 2012	0	141 Warnings (141 new, 0 filtered)	4 Infos (4 new, 0 filtered)
Map Report	Current	mié 7. mar 14:56:42 2012			
Place and Route Report	Current	mié 7. mar 14:59:58 2012	0	6 Warnings (6 new, 0 filtered)	4 Infos (4 new, 0 filtered)
Post-PAR Static Timing Report	Current	mié 7. mar 15:01:00 2012	0	6 Warnings (6 new, 0 filtered)	3 Infos (3 new, 0 filtered)
Bitgen Report	Current	mié 7. mar 15:02:24 2012			

ANEXO C
NÚMEROS DE PUERTO CONOCIDOS Y REGISTRADOS

ANEXO C NÚMEROS DE PUERTO CONOCIDOS Y REGISTRADOS

La siguiente tabla muestra el contenido parcial del archivo “/etc/services” propio de sistemas Unix en el cual se especifica los números de puerto conocidos y registrados tanto para TCP como para UDP. Esta lista está basada en la referencia [78].

Tabla C.1 Lista números de puertos conocidos (Fuente: [78])

Nombre del servicio	Número de puerto y protocolo
tcpmux	1/tcp
echo	7/tcp
echo	7/udp
discard	9/tcp
discard	9/udp
systat	11/tcp
daytime	13/tcp
daytime	13/udp
netstat	15/tcp
qotd	17/tcp
msp	18/tcp
msp	18/udp
chargen	19/tcp
chargen	19/udp
ftp-data	20/tcp
ftp	21/tcp
fsp	21/udp
ssh	22/tcp
ssh	22/udp
telnet	23/tcp
smtp	25/tcp
time	37/tcp
time	37/udp
rlp	39/udp
nameserver	42/tcp
whois	43/tcp
tacacs	49/tcp
tacacs	49/udp
re-mail-ck	50/tcp
re-mail-ck	50/udp
domain	53/tcp
domain	53/udp
mtp	57/tcp
tacacs-ds	65/tcp
tacacs-ds	65/udp
bootps	67/tcp
bootps	67/udp
bootpc	68/tcp
bootpc	68/udp
tftp	69/udp
gopher	70/tcp
gopher	70/udp
rje	77/tcp

Nombre del servicio	Número de puerto y protocolo
finger	79/tcp
www	80/tcp
www	80/udp
link	87/tcp
kerberos	88/tcp
kerberos	88/udp
supdup	95/tcp
hostnames	101/tcp
iso-tsap	102/tcp
acr-nema	104/tcp
acr-nema	104/udp
csnet-ns	105/tcp
csnet-ns	105/udp
rtelnet	107/tcp
rtelnet	107/udp
pop2	109/tcp
pop2	109/udp
pop3	110/tcp
pop3	110/udp
sunrpc	111/tcp
sunrpc	111/udp
auth	113/tcp
sftp	115/tcp
uucp-path	117/tcp
nntp	119/tcp
ntp	123/tcp
ntp	123/udp
pwdgen	129/tcp
pwdgen	129/udp
loc-srv	135/tcp
loc-srv	135/udp
netbios-ns	137/tcp
netbios-ns	137/udp
netbios-dgm	138/tcp
netbios-dgm	138/udp
netbios-ssn	139/tcp
netbios-ssn	139/udp
imap2	143/tcp
imap2	143/udp
snmp	161/tcp
snmp	161/udp
snmp-trap	162/tcp
snmp-trap	162/udp
cmip-man	163/tcp
cmip-man	163/udp
cmip-agent	164/tcp
cmip-agent	164/udp
mailq	174/tcp
mailq	174/udp
xdmcp	177/tcp

Nombre del servicio	Número de puerto y protocolo
xdmcp	177/udp
nextstep	178/tcp
nextstep	178/udp
bgp	179/tcp
bgp	179/udp
prospero	191/tcp
prospero	191/udp
irc	194/tcp
irc	194/udp
smux	199/tcp
smux	199/udp
at-rtmp	201/tcp
at-rtmp	201/udp
at-nbp	202/tcp
at-nbp	202/udp
at-echo	204/tcp
at-echo	204/udp
at-zis	206/tcp
at-zis	206/udp
qmtp	209/tcp
qmtp	209/udp
z3950	210/tcp
z3950	210/udp
ipx	213/tcp
ipx	213/udp
imap3	220/tcp
imap3	220/udp
pawserv	345/tcp
pawserv	345/udp
zserv	346/tcp
zserv	346/udp
fatserv	347/tcp
fatserv	347/udp
rpc2portmap	369/tcp
rpc2portmap	369/udp
codaauth2	370/tcp
codaauth2	370/udp
clearcase	371/tcp
clearcase	371/udp
ulistserv	372/tcp
ulistserv	372/udp
ldap	389/tcp
ldap	389/udp
imsp	406/tcp
imsp	406/udp
https	443/tcp
https	443/udp
snpp	444/tcp
snpp	444/udp
microsoft-ds	445/tcp

Nombre del servicio	Número de puerto y protocolo
microsoft-ds	445/udp
kpasswd	464/tcp
kpasswd	464/udp
saft	487/tcp
saft	487/udp
isakmp	500/tcp
isakmp	500/udp
rtsp	554/tcp
rtsp	554/udp
nqs	607/tcp
nqs	607/udp
npmp-local	610/tcp
npmp-local	610/udp
npmp-gui	611/tcp
npmp-gui	611/udp
hmmp-ind	612/tcp
hmmp-ind	612/udp
qmqp	628/tcp
qmqp	628/udp
ipp	631/tcp
ipp	631/udp

Tabla C.2 Lista números de puertos registrados (Fuente: [78])

Nombre del servicio	Número de puerto y protocolo
socks	1080/tcp
socks	1080/udp
proofd	1093/tcp
proofd	1093/udp
rootd	1094/tcp
rootd	1094/udp
openvpn	1194/tcp
openvpn	1194/udp
rmiregistry	1099/tcp
rmiregistry	1099/udp
kazaa	1214/tcp
kazaa	1214/udp
nessus	1241/tcp
nessus	1241/udp
lotusnote	1352/tcp
lotusnote	1352/udp
ms-sql-s	1433/tcp
ms-sql-s	1433/udp
ms-sql-m	1434/tcp
ms-sql-m	1434/udp
ingreslock	1524/tcp
ingreslock	1524/udp
prospero-np	1525/tcp
prospero-np	1525/udp
datametrics	1645/tcp

Nombre del servicio	Número de puerto y protocolo
datametrics	1645/udp
sa-msg-port	1646/tcp
sa-msg-port	1646/udp
kermit	1649/tcp
kermit	1649/udp
l2f	1701/tcp
l2f	1701/udp
radius	1812/tcp
radius	1812/udp
radius-acct	1813/tcp
radius-acct	1813/udp
msnp	1863/tcp
msnp	1863/udp
unix-status	1957/tcp
log-server	1958/tcp
remoteping	1959/tcp
cisco-sccp	2000/tcp
cisco-sccp	2000/udp
search	2010/tcp
pipe_server	2010/tcp
nfs	2049/tcp
nfs	2049/udp
gnunet	2086/tcp
gnunet	2086/udp
rtcm-sc104	2101/tcp
rtcm-sc104	2101/udp
gsigatekeeper	2119/tcp
gsigatekeeper	2119/udp
gris	2135/tcp
gris	2135/udp
cvspserver	2401/tcp
cvspserver	2401/udp
venus	2430/tcp
venus	2430/udp
venus-se	2431/tcp
venus-se	2431/udp
codasrv	2432/tcp
codasrv	2432/udp
codasrv-se	2433/tcp
codasrv-se	2433/udp
mon	2583/tcp
mon	2583/udp
dict	2628/tcp
dict	2628/udp
gsiftp	2811/tcp
gsiftp	2811/udp
gpsd	2947/tcp
gpsd	2947/udp
gds_db	3050/tcp
gds_db	3050/udp

Nombre del servicio	Número de puerto y protocolo
icpv2	3130/tcp
icpv2	3130/udp
mysql	3306/tcp
mysql	3306/udp
nut	3493/tcp
nut	3493/udp
distcc	3632/tcp
distcc	3632/udp
daap	3689/tcp
daap	3689/udp
svn	3690/tcp
svn	3690/udp
suucp	4031/tcp
suucp	4031/udp
sysrqd	4094/tcp
sysrqd	4094/udp
remctl	4373/tcp
remctl	4373/udp
iax	4569/tcp
iax	4569/udp
radmin-port	4899/tcp
radmin-port	4899/udp
rfe	5002/udp
rfe	5002/tcp
mmcc	5050/tcp
mmcc	5050/udp
sip	5060/tcp
sip	5060/udp
sip-tls	5061/tcp
sip-tls	5061/udp
aol	5190/tcp
aol	5190/udp
xmpp-client	5222/tcp
xmpp-client	5222/udp
xmpp-server	5269/tcp
xmpp-server	5269/udp
cfengine	5308/tcp
cfengine	5308/udp
mdns	5353/tcp
mdns	5353/udp
postgresql	5432/tcp
postgresql	5432/udp
freeciv	5556/tcp
freeciv	5556/udp
amqp	5672/tcp
amqp	5672/udp
amqp	5672/sctp
ggz	5688/tcp
ggz	5688/udp
x11	6000/tcp

Nombre del servicio	Número de puerto y protocolo
x11	6000/udp
x11-1	6001/tcp
x11-1	6001/udp
x11-2	6002/tcp
x11-2	6002/udp
x11-3	6003/tcp
x11-3	6003/udp
x11-4	6004/tcp
x11-4	6004/udp
x11-5	6005/tcp
x11-5	6005/udp
x11-6	6006/tcp
x11-6	6006/udp
x11-7	6007/tcp
x11-7	6007/udp
gnutella-svc	6346/tcp
gnutella-svc	6346/udp
gnutella-rtr	6347/tcp
gnutella-rtr	6347/udp
sge_qmaster	6444/tcp
sge_qmaster	6444/udp
sge_execd	6445/tcp
sge_execd	6445/udp
afs3-fileserver	7000/tcp
afs3-fileserver	7000/udp
afs3-callback	7001/tcp
afs3-callback	7001/udp
afs3-prserver	7002/tcp
afs3-prserver	7002/udp
afs3-vlserver	7003/tcp
afs3-vlserver	7003/udp
afs3-kaserver	7004/tcp
afs3-kaserver	7004/udp
afs3-volser	7005/tcp
afs3-volser	7005/udp
afs3-errors	7006/tcp
afs3-errors	7006/udp
afs3-bos	7007/tcp
afs3-bos	7007/udp
afs3-update	7008/tcp
afs3-update	7008/udp
afs3-rmtsys	7009/tcp
afs3-rmtsys	7009/udp
font-service	7100/tcp
font-service	7100/udp
http-alt	8080/tcp
http-alt	8080/udp
bacula-dir	9101/tcp
bacula-dir	9101/udp
bacula-fd	9102/tcp

Nombre del servicio	Número de puerto y protocolo
bacula-fd	9102/udp
bacula-sd	9103/tcp
bacula-sd	9103/udp
xmms2	9667/tcp
xmms2	9667/udp
amanda	10080/tcp
amanda	10080/udp
hkp	11371/tcp
hkp	11371/udp
bprd	13720/tcp
bprd	13720/udp
bpdbm	13721/tcp
bpdbm	13721/udp
bpjava-msvc	13722/tcp
bpjava-msvc	13722/udp
vnetd	13724/tcp
vnetd	13724/udp
bpcd	13782/tcp
bpcd	13782/udp
vopied	13783/tcp
vopied	13783/udp
wnn6	22273/tcp
wnn6	22273/udp

ANEXO D
PROGRAMAS CON SOPORTE NETFLOW E IPFIX

ANEXO D PROGRAMAS CON SOPORTE NETFLOW E IPFIX

Las ventajas que ofrece la utilización de un protocolo de exportación de flujos son las siguientes:

- Netflow e IPFIX son protocolos estándar y por tal motivo muchos dispositivos de red como routers y switches, especialmente los últimos modelos, soportan Netflow.
- Si bien es cierto Netflow es un protocolo desarrollado por Cisco, muchos fabricantes diferentes a Cisco incorporan este protocolo en muchos modelos de sus equipos.
- Existen muchas herramientas de software que recolectan paquetes Netflow y que realizan el análisis de la información.

Las desventajas que ofrece la utilización de un protocolo de exportación de flujos son las siguientes:

- Utiliza recursos del equipo de red. Aunque esta característica no afecta sustancialmente el rendimiento de estos equipos en condiciones normales, se debe tomar en consideración el uso de CPU y memoria que se requiere.
- Es necesario tener acceso a los equipos de red para realizar la respectiva configuración. Se debe considerar este hecho ya que hay situaciones en el que el acceso a estos equipos está restringido a un grupo de personas destinado para ello y muchas veces si se desea realizar una implementación de este tipo, es necesario solicitar el permiso correspondiente.

Algunas desventajas del uso de un hardware dedicado son las siguientes:

- Debido a su tamaño reducido y su poco peso, el uso de un dispositivo hardware dedicado permite ubicarlo en cualquier parte de la red y poder moverlo fácilmente. Asimismo, se pueden utilizar varios de estos equipos en diferentes puntos para tener una mayor visión de la red.
- Son difíciles de ser víctimas de un ataque por parte de agentes externos ya que usualmente la mayoría utiliza un sistema operativo como Linux embebido que no son blancos de los hackers.
- Usualmente no almacena la información en un disco ya que toda la información es enviada a la central de procesamiento.
- Muchos dispositivos hardware cuentan con aceleradores para la lectura de paquetes cuyo rendimiento supera a la de una tarjeta de red convencional. Muchos de ellos pueden leer paquetes a velocidades mayores a 1Gbps sin descarte de paquetes.
- La mayoría tiene la capacidad de colocar a cada paquete capturado una estampa de tiempo con un alto nivel de exactitud.

Algunas desventajas del uso de un hardware dedicado son las siguientes:

- Algunos dispositivos hardware tienen un costo relativamente alto especialmente aquellos que soportan velocidades de lectura mayores a los 10Gbps.

- Usualmente estos dispositivos deben de realizar la lectura de paquetes junto con un concentrador, conmutador o TAP para poder hacer la derivación del tráfico hacia uno de sus puertos.
- Se requiere tener conocimiento de las opciones de configuración propias del dispositivo; esto va a depender del fabricante.
- Softflowd: Software libre desarrollado para Linux y OpenBSD. Es un analizador de tráfico de red basado en flujos que soporta el protocolo Netflow de Cisco. Rastrea el flujo de tráfico ya sea utilizando directamente la interfaz de red en modo promiscuo o indirectamente mediante la lectura de un archivo de captura de paquetes de formato pcap. Soporta las versiones 1, 5 y 9 de Netflow y es compatible con IPv6. Puede rastrear flujos de IPv6 y a la vez exportar datagramas por IPv6. En las estadísticas que exporta se incluyen valores mínimos, máximos, promedios y totales de bytes y paquetes por cada protocolo (TCP, UDP, etc.).
- fprobe-ng: Es un software libre basado en la utilidad libpcap. Recolecta tráfico de red y envía paquetes Netflow a un colector específico. Soporta las versiones 1, 5 y 7 de Netflow. Utiliza un método sencillo de detección de paquetes IP en comparación con otras herramientas similares. Se pueden emplear la sintaxis de filtros de tcpdump para mejorar su funcionalidad.
- fprobe-ulog: Similar a fprobe-ng pero utilice un método distinto para la captura de información. Está basado en Netfilter para identificar con cadenas de iptables el tráfico que se desea analizar. Es más compleja su instalación ya requiere tener compatibilidad con el módulo ULOG del kernel de Linux.
- nProbe: Es una herramienta de “ntop” que funciona como sonda. Está disponible como una aplicación de software independiente o también como sistema embebido. Se encuentra disponible para sistemas Unix, MacOS X, Solaris, Windows y para sistemas embebidos. Soporta IPv4 e IPv6 y es configurable por el usuario. Está diseñado para ejecutarse en entornos de recursos limitados ya que el tamaño del software es menos que 100 KB. Soporta la versión 5 y 9 de Netflow y también IPFIX. Asimismo cuenta con soporte nativo de PF_RING para alta velocidad de generación de flujos.

ANEXO E
GLOSARIO DE TÉRMINOS

ANEXO E GLOSARIO DE TÉRMINOS

ACE	Advanced Configuration Environment
ACM	Association for Computing Machinery
ALU	Arithmetic Logic Unit
API	Application Program Interface
ARP	Address Resolution Protocol
ASIC	Application-specific Integrated Circuit
BDM	Background Debug Mode
BRAM	Block Read Only Memory
BSB	Base System Builder
BSP	Board Support Package
CISC	Complex Instruction Set Computing
CLB	Configurable Logic Block
CLI	Comand Line Interface
CPLD	Complex Programmable logic Device
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense Multiple Access with Colision Detect
CVS	Comma Separated Values
DDR2	Double Data Rate
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
DNS	Domain Name System
DRAM	Dynamic Read Only Memory
DSP	Digital Signal Processing
EDK	Embedded Developmetn Kit
EEPROM	Electrically Erasable Programmable Read Only Memory
EPROM	Erasable Programmable Read Only Memory
FCS	Frame Check Sequence
FIFO	First In, First Out
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
GCC	GNU Compiler Collection
GMII	Gigabit Media Independent Interface
GNU	GNU is not Unix
HDL	Hardware Description Language

HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IDE	Integrated Development Environment
IFG	Interframe gap
IGV	Impuesto General a las ventas
IHL	Internet Header Length
INEI	Instituto Nacional de Estadística e Informática
INICTEL	Instituto Nacional de Investigación y Capacitación en Telecomunicaciones
IP	Internet Protocol.
IPFIX	Internet Protocol Flow Information Export
IPTV	Internet Protocol Television
IPv4	Internet Protocol version 4
IRQL	Interrupt Request Level
ISA	Industry Estándar Architecture
ISE	Integrated Software Environment
ISP	Internet Service Provider
JCP	Java Community Process
JDK	Java Development kit
JSF	JavaServer Faces
JSP	JavaServer Pages
JTAG	Joint Test Action Group
LAN	Local Area Network
LED	Light-Emitting Diode
LMB	Local Memory Bus
LTS	Long Term Support
LUT	Look-up Table
LWIP	Lightweight Internet Protocol
MAC	Media Access Control
MAN	Metropolitan Area network
MEMC	Memory Controller
MMU	Mmemory Management Unit
MROM	Mask Read Only Memory
MTU	Maximum Transfer Unit
NIC	Network Interface Card

NVM	Non Volatile Memory
OSI	Open Systems Interconnection
OTP	One-Time Programmable
PC	Personal Computer
PC4	Parallel Cable IV
PCB	Process Control Block
PCS	Physical Coding Sublayer
PDF	Portable Document Format
PLB	Processor Local Bus
PLL	Phase Looked Loop
PMA	Physical Medium Attachment
PMD	Physical Medium Dependent
POP3	Post Office Protocol
PROM	Programmable Read Only Memory
QoS	Quality of Service
RAAP	Red Académica Peruana
RAM	Random Access Memory
RFC	Request for Comments
RIP	Routing Information Protocol
RISC	Reduced Instruction Set Computing
ROM	Read Only Memory
RRD	Round Robin Database
RTT	Round Trip Time
RU	Relative Uncertainty
SCTP	Stream Control Transmission Protocol
SDF	Start Frame Delimiter
SDK	Software Development Kit
SIGCOMM	Special Interest Group on Data Communication
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SoC	System on a Chip
SODIMM	Small Outline DUAL In-line Memory Module
SPAN	Switched Port Analyzer
SRAM	Static Read Only Memory
SSH	Secure Shell
STP	Spanning Tree Protocol

SIP	Session Initiation Protocol
TAP	Test Access Point
TCP	Transfer Transport Protocol
TFTP	Trivial File Transfer Protocol
TTL	Time to Live
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VLAN	Virtual Local Area Network
VoIP	Voz sobre Protocolo de Internet
WAN	Wide Area network
XML	Extensible Markup Language
XPS	Xilinx Platform Studio

BIBLIOGRAFÍA

[1] Kuai Xu, et al, "Profiling Internet BackboneTraffic: Behavior Models and Applications", Evento SIGCOMM'05, 21–26 de agosto de 2005, Philadelphia, Pennsylvania, USA. http://conferences.sigcomm.org/sigcomm/2005/paper-XuZha.pdf.

[2] SIGCOMM, Página web institucional, “About”, http://www.sigcomm.org/about.

[3] Ketterling, Hans-Peter A.., “Introduction to Digital Professional Mobile Radio”, Artech House, 2004, ISBN 1580531733.

[4] Tammy Noergaard, “Embedded Systems Architecture. A Comprehensive Guide for Engineers and Programmers”. Ed. Newnes, 2005.

[5] Pong P. Chu, “FPGA Prototyping by VHDL Examples”. Xilinx SpartanTM-3 Version.

[6] National Instruments. “Introducción a la Tecnología FPGA: Los Cinco Beneficios Principales”, http://zone.ni.com/devzone/cda/tut/p/id/8259.

[7] Adam Dunkels, “Design and Implementation of the lwIP TCP/IP Stack”. Swedish Institute of Computer Science.

[8] Redes de Computadoras. Tercera Edición. Andrew S. Tanenbaum. Pearson Education. 1997.

[9] Cisco " CCNA Exploration 4.0. Aspectos Básicos de Networking. Capítulo 2

[10] Op. Cit, Capítulo 9.

[11] Op. Cit, Capítulo 5.

[12] Op. Cit, Capítulo 4.

[13] W. Richard Stevens. TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, 1994, ISBN 0-201-63346-9

[14] Cisco. “Cisco IOS NetFlow”. http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html

[15] Brian Trammell and Elisa Boschi, ETH Zurich, “An Introduction to IP Flow Information Export (IPFIX)”. http://dl.comsoc.org/livepubs/ci1/public/2011/apr/trammell.html

[16] Variables aleatorias. Funciones de distribución. http://centros5.pntic.mec.es/ies.valle.de.leiva/Departamentos/Matematicas/matcienciassociales2/A91_Funciones%20de%20distribucion.pdf

[17] Universidad TALCA, “Estadística Descriptiva” http://dta.utalca.cl/estadistica/ejercicios/obtener/descriptiva/Estadistica%20Descriptiva.pdf

[18] Tuveras, “Medidas Descriptivas”, http://www.tuveras.com/estadistica/estadistica02.htm

[19] Universidad Autónoma de Madrid, “Probabilidad, Definiciones”,

http://www.uam.es/personal_pdi/psicologia/carmenx/EsquemaTema15.pdf

[20] Thomas M. Cover, Joy A. Thomas, “Elements of Information Theory”. Second Edition.

[21] ntop, nBox “An Embedded NetFlow v5/v9/IPFIX Probe (IPv4, IPv6, MPLS)” http://www.ntop.org/products/nbox/

[22] Endace, “Tarjetas de Captura”, http://www.endace.com/endace-dag-high-speed-packet-capture-cards.html.

[23] Endace,, “NinjaBox Datasheet” http://www.altima.co.jp/products/endace/download/EDM12_71v2_NinjaBox_Datasheet_A4SCN.pdf.

[24] Liberouter, “Programmable Hardware”, http://www.liberouter.org/hardware.php?flag=2.

[25] Mindrot, “Softflowd”, http://www.mindrot.org/projects/softflowd/ .

[26] Lopsa, “fprobe-ng”, https://lopsa.org/node/1732 .

[27] SourForge, “fprobe”, http://sourceforge.net/projects/fprobe/.

[28] ntop, “nProbe™ v6, An Extensible NetFlow v5/v9/IPFIX GPL Probe for IPv4/v6” http://www.ntop.org/products/nprobe/.

[29] ntop, “Traffic analysis with NetFlow™ and sFlow™ support”. http://www.ntop.org/products/ntop/

[30] ManageEngine, “NetFlow Analyzer”, http://www.manageengine.com/products/netflow/spanish/index.html

[31] Interactive advertising bureau (IAB Perú), “Acerca de Certifica”, http://www.iabperu.com/ia_certifica.aspx?men=1&sub=3

[32] Página web oficial de Google Analytics. http://www.google.com/analytics/

[33] Acerca de Google Analytics. http://support.google.com/adwords/bin/answer.py?hl=es-419&hlrm=es-419&answer=2404031

[34] Adam Dunkels, “Design and Implementation of the lwIP TCP/IP Stack”, febrero 2001. http://www.sics.se/~adam/lwip/doc/lwip.pdf

[35] EDK Concepts, Tools, and Techniques. A Hands-On Guide to Effective Embedded System Design. UG683 EDK 11 http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/edk_ctt.pdf

[36] Embedded System Tools Reference Manual. Embedded Development Kit, EDK 11.1. UG 111, EDK 11.1 http://china.xilinx.com/support/documentation/sw_manuals/xilinx11/est_rm.pdf

[37] What Is an IP Core? http://www.wisegeek.com/what-is-an-ip-core.htm

[38] ISE WebPACK Design Software http://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.htm

[39] Matpic, “Hardware Description Language”. http://www.matpic.com/esp/vhdl/vhdl.html

[40] ISE Design Suite 11: Installation, Licensing, and Installation, Release Notes. UG631 (v 11.5) March 16, 2010, página 46. http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/irn.pdf

[41] Implementing Barrel Shifters Using Multipliers. Xilinx XAPP195 (v1.1) August 17,

2004.
http://china.origin.xilinx.com/support/documentation/application_notes/xapp195.pdf

[42] Redes de Computadoras. Capa de Enlace de Datos.
http://soporteeducativo.media.officelive.com/Documents/Capa_de_enlace_de_datos.pdfQqwqw

[43] DS572 April 19, 2010. Logicore IP XPS Interrupt Controller (v2.01a).
http://www.xilinx.com/support/documentation/ip_documentation/xps_intc.pdf

[44] Marvel Alaska 88EIIII. Single-Port Gigabit Ethernet Transceiver.
http://www.xilinx.com/products/boards/ml505/datasheets/M88E1111.pdf

[45] DS445 April 24, 2009. Local Memory Bus (LMB) V10 (v1.00a).
http://www.xilinx.com/support/documentation/ip_documentation/lmb_v10.pdf

[46] DS531 April 24, 2009. Processor Local Bus (PLB) v4.6 (v1.04a). Product Specification.
http://www.xilinx.com/support/documentation/ip_documentation/ds531.pdf

[47] RFC 3917. “Requirements for IP Flow Information Export” (IPFIX).http://www.ietf.org/rfc/rfc3917.txt

[48] Adam Dunkels . Design and Implementation of the LWIP TCP/IP Stack. Swedish Institute of Computer Science. February 20, 2001.
http://www.es.sdu.edu.cn/project/doc/Design%20and%20Implementation%20of%20the%20lwIP%20tcpIP%20stack.pdf

[49] Douglas E. Comer, “Redes Globales de Información con Internet y TCP/IP Principios básicos, protocolos y arquitectura” Página 294.

[50] Muraleedharan N,”Flow based Traffic Analysis”. C-DAC Bangalore Electronics City
http://www.iitg.ernet.in/cse/ISEA/isea_PPT/ISEA_02_09/Presenataion_Flow_Base_Analysis.pdf

[51] Xilinx. lwIP 1.3.0 Library (v1.00.b). UG 650 April 15, 2009.

[52] Comunicación mediante sockets.http://sopa.dis.ulpgc.es/progsis/material-didactico-teorico/tema7_1transporpagina.pdf

[53] Raw TCP/IP interface for lwIP. http://lwip-avr.googlecode.com/svn-history/r2/trunk/lwip/doc/rawapi.txt

[54] LWIP Wiki. Raw/UDP. http://lwip.wikia.com/wiki/Raw/UDP

[55] Xilinx, “OS and Libraries Document Collection”. UG 643 December 2, 2009.
http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/oslib_rm.pdf

[56] LogiCORE IP XPS Timer/Counter (v1.02a). DS573 April 19, 2010.
http://www.xilinx.com/support/documentation/ip_documentation/xps_timer.pdf.

[57] Freecode. SendIP. http://freecode.com/projects/sendip.

[58] packETH. http://packeth.sourceforge.net/.

[59] Wireshark. http://www.wireshark.org/about.html

[60] Ediciones de MySQL. http://www.mysql.com/products/

[61] Características de NetBeans. http://netbeans.org/features/index.html

[62] About the Java Technology.
http://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html

[63] Apache Tomcat 6.0. http://tomcat.apache.org/tomcat-6.0-doc/index.html

[64] K. Krippendorff, “Information Theory: Structural models for qualitative data”. Sage Publications, 1986.

[65] R. Cavallo and G. Klir, “Reconstructability analysis of multi-dimensional relations: A theoretical basis for computer-aided determination of aceptable systems models”, International Journal of General Systems, vol. 5, pp. 143-171, 1979.

[66] Xilinx, “ML505/ML506/ML507 Evaluation Platform, User Guide. UG347 (v3.1)” November 10, 2008. http://www.xilinx.com/support/documentation/boards_and_kits/ug347.pdf

[67] Virtex-5 Family Overview, Product Specification. DS100 (v5.0) February 6, 2009. http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf.

[68] Xilinx, “Programmable Logic Design- Quick start Handbook”, ASMBL Architecture, pag.29.

[69] IP core (intellectual property core). http://whatis.techtarget.com/definition/0,,sid9_gci759036,00.html

[70] http://www.xilinx.com/support/documentation/application_notes/xapp707.pdf

[71] Virtex-5 FPGA User Guide, UG190 (v5.0) June 19, 2009.

[72] IEEE Std 802.3-2002. http://cs.eou.edu/CSMM/twelch/802.3-2002_part1.pdf

[73] MicroBlaze Soft Processor Core. http://www.xilinx.com/tools/microblaze.htm

[74] MicroBlaze Processor Reference Guide. EDK 11.4 UG081 (v10.3). http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/mb_ref_guide.pdf

[75] MicroBlaze Tutorial Creating a Simple Embedded System and Adding Custom Peripherals Using EDK Software Tools. Rod Jesman, Fernando Martinez Vallina, Jafar Saniie. Embedded Computing and Signal Processing Laboratory – Illinois Institute of Technology. http://ecasp.ece.iit.edu/mbtutorial.pdf

[76] EDK Concepts, Tools, and Techniques. A Hands-On Guide to Effective Embedded System Design. UG683 EDK 11. http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/edk_ctt.pdf Qw

[77] Virtex-5 FPGA Embedded Tri-Mode Ethernet MAC User Guide. UG 194 (v1.8) April 28, 2009. http://www.xilinx.com/support/documentation/user_guides/ug194.pdf

[78] Service Name and Transport Protocol Port Number Registry. Last Updated 2012-04-30, http://www.iana.org/assignments/port-numbers.