UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO DEL NÚCLEO DE UN SISTEMA TRANSACCIONAL DE MEDIOS DE PAGO ELECTRÓNICO BASADO EN SOFTWARE OPEN SOURCE

INFORME DE SUFICIENCIA

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PRESENTADO POR:

JOSE ANTONIO PEÑA CETRARO

PROMOCIÓN
1994-I

LIMA – PERU
2009

DISEÑO DEL NÚCLEO DE UN SISTEMA TRANSACCIONAL DE MEDIOS DE PAGO ELECTRÓNICO BASADO EN SOFTWARE OPEN SOURCE

Dedico este trabajo a mi esposa e hijos por ser
mi motivación para superarme día a día

SUMARIO

El presente trabajo describe el diseño de un sistema de procesamiento de información integrado por componentes de hardware, software y tecnologías de comunicación, orientado al procesamiento de transacciones de medios de pago (tarjetas de crédito, débito).

La motivación del presente trabajo es el poder demostrar que existen soluciones de código abierto suficientemente maduras para cumplir con requerimientos de una industria específica, en este caso la de medios de pago, y que podemos utilizar para bajar barreras de entrada y competir con nuevas tecnologías aportando valor al mercado.

Se describirá en qué consiste un sistema transaccional de medios de pago, definiendo los conceptos principales y los estándares involucrados. Luego, en base a estas definiciones, y requerimientos planteados se darán detalles de la implementación para un sistema de este tipo del lado de la entidad adquirente, dando una descripción de arquitectura, configuraciones de hardware apropiadas y configuraciones en el software del switch transaccional. Además, se configurará un esquema para simulación de la funcionalidad (generación y autorización de transacciones) y pruebas de rendimiento. Por último se hace una comparativa referencial de costos entre soluciones de tecnología propietaria y soluciones abiertas como la tratada en este informe.

ÍNDICE

PRÓLOGO

El objetivo general del presente informe de suficiencia es mostrar el diseño básico de un sistema distribuido basado en tecnologías de código abierto (open source) y/o tecnologías de uso extendido, para implementar una solución de medios de pago, que responda al mensaje de solicitud de autorización generadas por los dispositivos de captura de la transacción.

Los objetivos específicos son los siguientes:

- Demostrar que con software de código abierto y arquitecturas abiertas se puede implementar el núcleo de un sistema transaccional de medios de pago flexible y escalable, que pueda soportar los requerimientos y tendencias del negocio.
- Integrar diferentes componentes tecnológicos existentes en el medio para implementar esta solución
- Configurar una simulación del aplicativo switch transaccional

La metodología de diseño está basada en la adaptación y configuración del sistema JPOS, que es un conjunto de librerías java, liberados bajo licencia open-source, y que sirven para desarrollar e implementar un Switch Transaccional. .

Alcances y Limitaciones

El diseño presentado implementa sólo la funcionalidad básica de un switch transaccional de medios de pago, es decir el autorizar una transacción. La simulación de funcionalidad y rendimiento se realizará con equipamiento gama baja, como son laptops de uso personal, pero se buscará hacer una extrapolación a equipos de gama alta/media para sacar las conclusiones respectivas.

El informe se divide en cuatro capítulos. En el primer capítulo se revisarán los conceptos y procesos principales de los sistemas de medios de pago, así como se describirá el estándar ISO 8583 para intercambio de mensajes entre entidades.

En el segundo capítulo se presentará los componentes principales del diseño, en una arquitectura que cumple los requisitos de procesamiento en un entorno real.

En el tercer capitulo se presentará la arquitectura interna del software que implementa el núcleo o switch transaccional, que es uno de los componentes principales de la arquitectura vista en el segundo capítulo. Este software consiste en librerías Java y está

liberado bajo el concepto de código abierto (open source). Asimismo se configurará una simulación funcional y de rendimiento en un entorno de pruebas.

Por último, el cuarto capítulo se incluye las conclusiones del informe, incidiéndose en la demostración de que con tecnologías abiertas y plataformas de uso común, pueden implementarse soluciones, dentro de la industria de medios de pago, que antes estaban reservadas a tecnologías cerradas y de costos elevados.

Agradezco de manera especial a Iván Otarola y Eduardo Espejo, de Telefónica del Perú, por el apoyo prestado en la elaboración del presente informe, y a Marco Espinoza, autor del aplicativo de TPV Virtual utilizado en las simulaciones.

CAPITULO I

INTRODUCCIÓN A LOS SISTEMAS DE MEDIO DE PAGO

En este capítulo se hace una descripción detallada de un sistema transaccional de medios de pago, sus principales componentes, así como las entidades que interactúan y los procesos principales que se llevan a cabo y los estándares utilizados.

1.1 Definiciones

Para entender el entorno de los medios de pago electrónicos es necesario explicar una serie de conceptos, que son tratados a continuación.

1.1.1 Medios de Pago Electrónico

Sistema de procesamiento de información que permite la realización de pagos a través de medios electrónicos, es decir que permiten el acceso a distancia a una cuenta bancaria, considerándose en particular como tales las tarjetas de pago y el dinero electrónico, en cualquiera de las formas en que pueda presentarse el mismo, almacenado bien en una tarjeta inteligente o bien en una memoria de ordenador.

1.1.2 Actores de un sistema de medios de pago electrónico

a) **Tarjeta-habiente**: Es la persona que posee una tarjeta de pago, de crédito o débito, emitida por el Emisor, y realiza y paga las compras con ésta.
b) **Adquirente**: Es la entidad financiera que establece una relación con el comercio o tienda, procesando las transacciones con tarjeta y las autorizaciones de pago
c) **Emisor**: Es la entidad financiera que emite la tarjeta del cliente, extiende su crédito y es responsable de la facturación, recolección y servicio al consumidor.
d) **Comercio**: Es la entidad que vende productos, servicios o información y acepta el pago electrónico, que es gestionado por su entidad financiera (adquirente).
e) **Procesadora**: Es la entidad responsable técnicamente de la captura y procesamiento electrónico de la transacción de venta, y lo hace en nombre del Adquirente (es contratada por el Adquirente)
f) **Marca**: La "marca" de la tarjeta es el nombre de la sociedad de medios de pago Internacional a la que pertenece (Por ejemplo Visa o Mastercard). Esta puede

tener acuerdos con Organizaciones Internacionales de Medios de Pago (Por ejemplo Mastercard International).

En la Figura 1.1, se muestran las relaciones entre los actores descritos líneas arriba. Se resalta que normalmente el adquirente se relaciona con la marca, pero se puede dar el caso que para emisores locales, tener acuerdos directos, para evitar costos asociados al uso de procedimientos y tecnologías de las marcas.

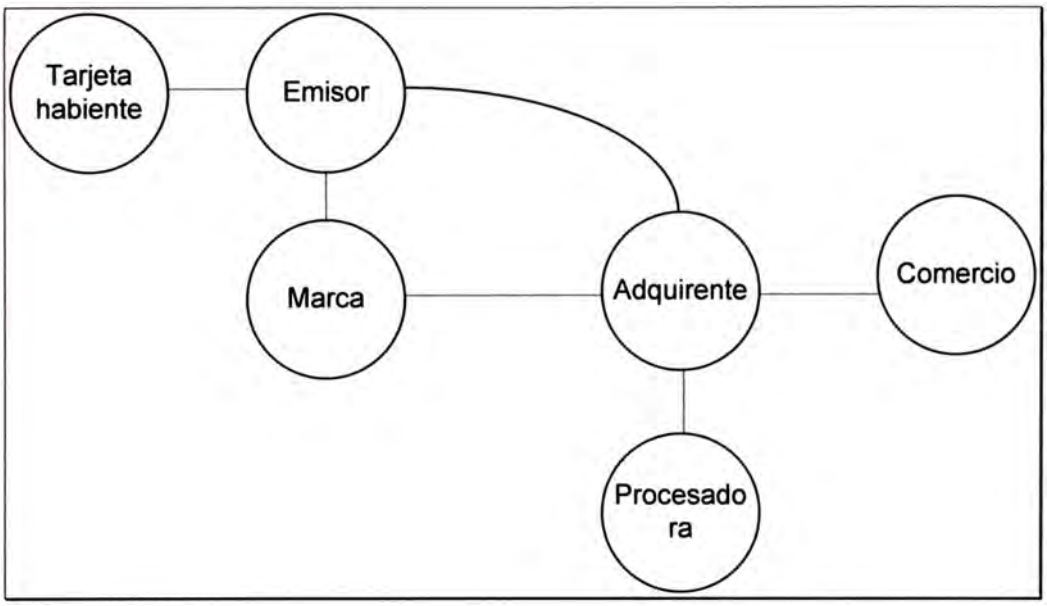

Figura 1.1.- Relaciones entre entidades

1.1.3 Elementos de un sistema de medios de pago electrónico

Para llevar a cabo el procesamiento electrónico de una transacción financiera de compra-venta, un sistema de medios de pago electrónico consistente en elementos de hardware y software distribuidos e interconectados entre sí por enlaces de comunicación.

a) **Terminal Punto de Venta** (TPV / POS): De sus siglas "Terminal de Punto de Venta" (Point of Sale terminal), dentro de los sistemas electrónicos de medios de pago, se refiere a un dispositivo electrónico que tienen como finalidad permitir la captura de información de la transacción (datos de la tarjeta de crédito o débito, datos de la venta, etc.), transmitirla hacia la entidad adquirente por algún enlace de comunicación y procesar la respuesta de ésta. Se instala en los establecimientos comerciales. Los TPV pueden ser clasificados por el tipo de tecnología que soportan, siendo éstas:
    - Enlace de comunicación: cable (telefónico o Ethernet) o wireless (WIFI, Bluetooth, GPRS, GSM)
    - Protocolo de comunicación: X.25, TCP/IP
    - Tipo de lectura: lectura de tarjetas de crédito/débito con banda magnética o con chip.

Actualmente los fabricantes de puntos de venta (Ingénico, Verizon, etc.) cuentan con sistemas wireless, que generalmente consisten en una estación base conectada a los sistemas del adquirente, por algún medio (conexión telefónica celular, conexión telefónica de línea fija, etc.). A esta estación base se conectan los puntos de venta de forma remota (wireless), a través de diferentes tecnologías (bluetooth, wifi). Otro modo de conexión es que el punto de venta wireless soporte una tarjeta GSM para comunicarse directamente con el proveedor de telefonía celular, y a través de éste llegue hacia los sistemas del adquirente. En la Figura 1.2 se muestra un ejemplo de un equipo wireless

Con este tipo de TPV, se facilitan muchos procesos de toma de pedidos, por ejemplo en restaurantes y grifos, donde se puede llevar el terminal junto al cliente, evitando desplazamientos incómodos de éste, haciendo que el uso de transacciones con tarjeta sea más seguro pues el cliente nunca pierde de vista la tarjeta, y por lo tanto, el riesgo de que alguien haga una copia de la información de la banda magnética se minimiza.

Figura 1.2.- TPV wireless, modelo Ingénico 8550

b) **Enlaces de comunicación**: Son los medios de conexión entre los equipos del sistema, se pueden utilizar diferentes medios como líneas telefónicas, líneas dedicadas locales, enlaces internacionales, generalmente son provistos por una Operadora de Comunicaciones.

c) **Network Access Control (NAC):** Elemento de red que sirve como punto de control para el ingreso de conexiones. En un NAC se utiliza un conjunto de protocolos para definir e implementar una política de seguridad para el acceso a nodos red, para dispositivos externos. En el caso de los sistemas de medios de pago, se utiliza para

validar el acceso de los TPVs a la red de adquirencia donde se encuentra el switch transaccional.

d) **Hardware Security Module (HSM):** Es un dispositivo de hardware, que ejecuta operaciones criptográficas, orientado a manejar llaves digitales, acelerar procesos criptográficos en términos de señales digitales por segundo, y proporcionar fuertes medidas de autenticación para el acceso de llaves críticas a servidores de aplicaciones. En el caso de los medios de pago, se utiliza para encriptar/desencriptar informaciones transmitidas en los mensajes, como los PIN (clave) de las transacciones de débito.

e) **Switch transaccional**: En el sentido general un switch transaccional es un sistema de hardware y software que recibe las transacciones de una entidad y las encamina a otra entidad, de acuerdo a las reglas definidas previamente. Se considera a este elemento como el núcleo de un sistema de medios de pago. Por ejemplo, dentro de un sistema de medios de pago, la entidad procesadora puede tener un switch transaccional para recibir las transacciones de los TPV, y encaminarlas hacia los sistemas de los sistemas de los emisores (autorizadores) que generan la autorización para que la transacción se complete.

f) **Sistemas back-office**: Se refieren a todos los sistemas que sirven como soporte para la explotación de la información generada por el switch, y también a los módulos que soportan los procesos que requiere el adquirente para configurar la operación del sistema (afiliaciones/desafiliaciones de comercios, pago a los comercios, emisiòn de información a los comercios, intercambio de información financiera con los emisores, instalación y mantenimiento de los TPV, etc.)

g) **Estándar ISO 8583**: Es un protocolo de comunicación de alto nivel, usado para transferir los mensajes financieros entre los elementos del sistema. Es el estándar de la Organización Internacional de Normalización (ISO). ISO 8583 define un formato de mensaje y un flujo de comunicación para que los diferentes sistemas puedan intercambiar estas transacciones. Si bien es cierto que, la norma ISO 8583 define un protocolo, generalmente éste no es usado en forma directa, sino que cada entidad o marca de tarjeta adapta el estándar según sus necesidades, utilizando los campos que determine convenientes para cubrir sus requerimientos.

1.2 Descripción de un sistema distribuido de medios de pago electrónico

A continuación se describen cómo interactúan, de manera general, los principales componentes de este sistema, definidos en la sección 1.1.2.

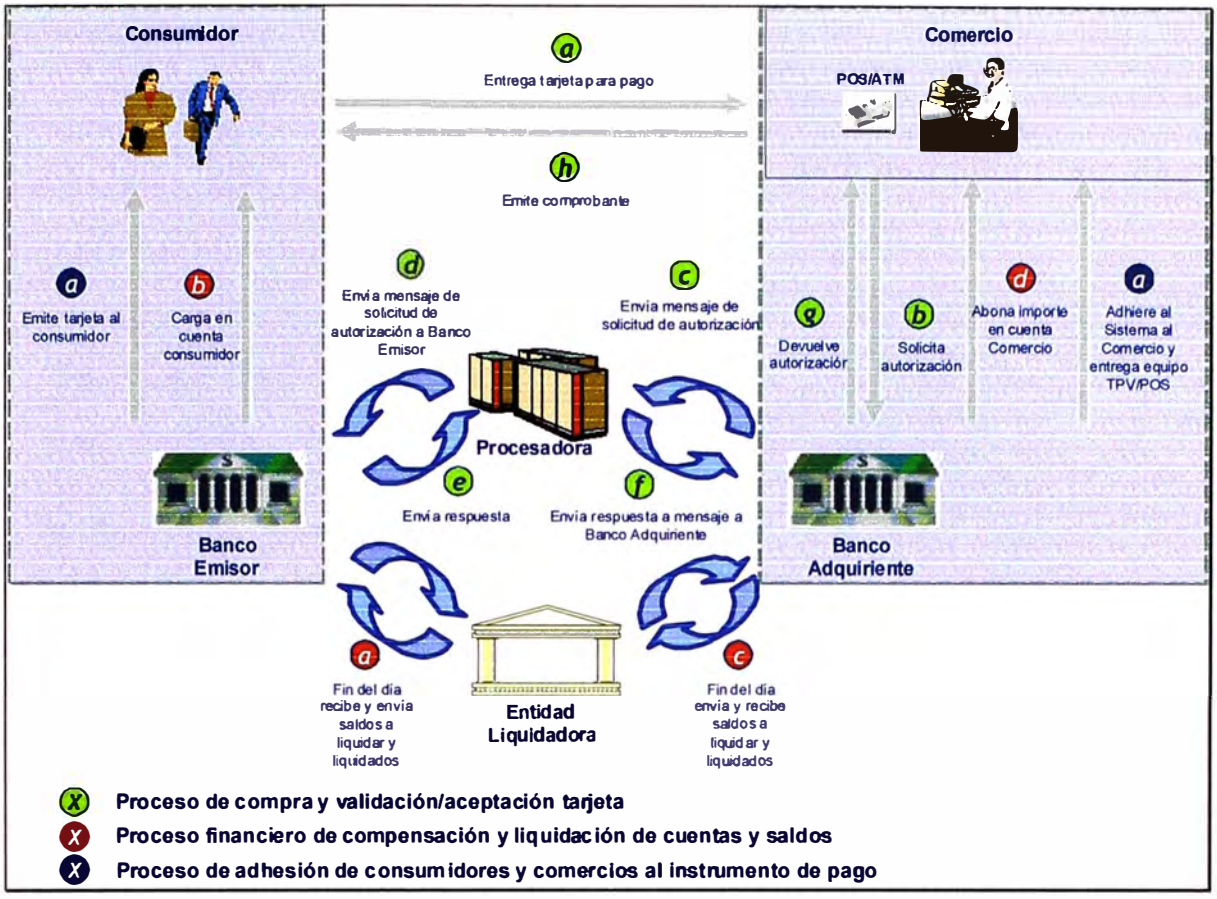

Figura 1.3.- Procesos principales en un sistema distribuido de medios de pago electrónico

Como se muestra en la Figura 1.3, un sistema de medios de pago tiene 3 procesos principales, que se describen a en las siguientes secciones.

1.2.1 Proceso de compra y validación de aceptación de tarjeta

La finalidad de este proceso es registrar la compra y obtener la autorización del emisor de la tarjeta.

a) El tarjeta-habiente entrega su tarjeta de crédito o débito al dependiente de la tienda para realizar el pago de la compra.

b) El dependiente realiza la transacción pasando la tarjeta por el TPV, el cual envía un mensaje electrónico de solicitud de autorización a la entidad Adquirente.

c) y d) La entidad Adquirente, por medio de la entidad Procesadora, envía un mensaje a la entidad Emisora correspondiente, de acuerdo a los datos del mensaje recibido del TPV, y espera la autorización de la transacción.

e) El Emisor autoriza la transacción de acuerdo a la disponibilidad de saldo de la cuenta del tarjeta-habiente, y responde al adquirente con un mensaje.

f) y g) El Adquirente recibe el mensaje y envía un nuevo mensaje al TPV con la autorización de la transacción.

h) El TPV recibe la autorización e imprime el voucher. El dependiente solicita la firma del tarjeta-habiente y da por concluida la compra.

Todos los mensajes de petición de autorización y respuesta deben transcurrir en pocos segundos mientras se realiza la compra. Se dice que estos mensajes son On-line.

1.2.2 Proceso financiero de compensación y liquidación de cuentas y saldos

La finalidad de este proceso, que se realiza después del proceso de compra-venta, es realizar las transferencias financieras entre el tarjeta-habiente, emisor, adquirente y comercio.

a) La entidad Adquirente, por medio de la entidad Liquidadora, envía un archivo a cada Emisor con información de todas las transacciones del día o del periodo correspondiente.
b) El Emisor ejecuta los cargos a las cuentas de sus tarjeta-habientes
c) El Emisor responde otro archivo con la aceptación o rechazo de las transacciones y con los importes a transferir al adquirente para el pago a los comercios, cobrándose la comisión por cada transacción que le corresponde.
d) El Adquirente realiza la liquidación a los Comercios por las transacciones realizadas, cobrándose la comisión correspondiente y pactada con éstos.

1.2.3 Proceso adhesión de consumidores y comercios al instrumento de pago

a) La entidad Adquirente y las entidades Emisoras, afilian cada una por su lado a los Comercios y Tarjeta-Habientes respectivamente, acordando tasas y comisiones y condiciones del servicio.

1.3 El Estándar ISO 8583

ISO 8583 es el estándar de la Organización Internacional para la Normalización (ISO), para sistemas de intercambio de transacciones electrónicas realizadas por poseedores de tarjetas de crédito o débito.

Es importante notar que ISO 8583 no es en sí un protocolo sino un metaprotocolo que proporciona un conjunto de reglas para la definición de protocolos de transacciones financieras, de este modo, existen una variedad de protocolos basados en ISO 8583, como por ejemplo Base I, usado por VISA para autorizaciones de transacciones.

---

[1] Ver referencia bibliográfica [1]

1.3.1 Estructura de los Mensajes

Los mensajes ISO 8583 se componen de 3 partes:

a) **MTI (Message Type Indicator)**

Es un código numérico de 4 dígitos, que contiene información acerca del tipo de mensaje, que incluye la versión ISO utilizada, la clase del mensaje, la función del mensaje y el originador del mismo. Típicamente, los 4 campos del MTI son codificados en 2 bytes, siendo cada campo representado por medio byte o un nibble; por ejemplo un mensaje que empieza por 0x01 0x00 significa que el valor de la versión ISO es 0, la clase es 1 y la función y el originador son 0.

Versión:

Es el primer campo del mensaje, e indica la versión del estándar ISO a la que pertenece. El campo tiene los siguientes valores permitidos, mostrados en la Tabla N° 1.1.

Tabla N° 1.1.- Valores del campo Versión

Valor	Versión
0	1987
1	1993
2	2003
3-9	Reservado para uso futuro

Clase del Mensaje:

Información acerca de cómo es categorizado el mensaje. El campo tiene los siguientes valores permitidos, mostrados en la Tabla N° 1.2.

Tabla N° 1.2.- Valores del campo Clase de Mensaje

Valor	Clase de Mensaje
0	Reserved for ISO use
1	Authorization
2	Financial presentment
3	File action
4	Reversal/Chargeback
5	Reconciliation
6	Administrative
7	Fee collection
8	Network management
9	Reserved for ISO use

Función del Mensaje

Información acerca de la funcionalidad del mensaje. El exacto significado de estos valores depende de la versión o adaptación de ISO 8583 que se esté usando, como se muestra en la Tabla N° 1.3.

Tabla N° 1.3.- Valores del campo Función del Mensaje

Valor	Función del Mensaje
0	Request
1	Request response
2	Advice
3	Advice response
4	Notification
5	Notification acknowledgment
6	Instruction
7	Instruction acknowledgment
8,9	Reserved for ISO use

Origen del mensaje

Define el originador del mensaje, como se muestra en la Tabla N° 1.4

Tabla N° 1.4.- Valores del campo Origen del mensaje

Valor	Origen del Mensaje
0	Acquirer
1	Acquirer repeat
2	Card Issuer
3	Card Issuer repeat
4	Other originator
5	Other originator repeat
6-9	Reserved for ISO use

Como vemos, con estos 4 dígitos, un MTI describirá completamente qué es lo que un mensaje deberá hacer y cómo será transmitido a través de la red. No todas las implementaciones del ISO 8583 interpretan el significado de un MTI de la misma manera, pero algunos pocos MTI's son estándar, según se muestra en la Tabla N° 1.5.

Tabla N° 1.5.- Valores estándares del MTI

MTI	Significado	Uso
0100	Requerimiento de autorización	Requerimiento desde un terminal TPV para autorizar una compra de un tarjetahabiente
0120	Aviso de Autorización	Para transacciones manuales
0121	Aviso de Autorización Repetición	Si el aviso se vence por timeout
0200	Requerimiento Financiero del Comprador	Requerimiento de fondos, usualmente de un ATM
0220	Aviso Financiero del Comprador	Ejemplo: Checkout de un hotel
0221	Aviso Financiero del Comprador Repetición	Si el aviso se vence por timeout
0400	Requerimiento de Reverso del Comprador	Reversa o anulación de una transacción

0420	Aviso de Reverso del Comprador	Aviso de que se realizó un reverso
0421	Aviso de Reverso del Comprador Repetición	Si el reverso se vence por timeout
0800	Requerimiento de Manejo de Red	Echo test, logon, log off etc.
0820	Aviso de Manejo de Red	Keychange

b) Mapa de Bits

El mapa de bit es un campo o subcampo que indica que otros elementos (campos o subcampos) se encuentran en el mensaje. Un mensaje contendrá al menos un mapa de bits, llamado el *Mapa de Bits Primario* que indica que campos (Data Elements) del 1 al 64 están presentes. Puede existir un mapa de bits secundario, que indica que campos del 65 al 128 están presentes. De igual forma, un tercer bitmap puede usarse para indicar la presencia o ausencia de los campos del 129 al 192, aunque esos campos casi nunca se usan.

Un campo está presente cuando el bit correspondiente está en '1', ej. el byte 42x en binario es '0100 0010' lo que significa que los campos 2 y 7 están presentes en este mensaje. Con esto se obtiene ahorro en el espacio utilizado por la trama de datos, ya que sólo ocuparán espacio los campos presentes.

Para indicar la presencia del mapa de bits secundario, se utiliza el primer bit del mapa primario, si éste está encendido (valor '1') significa que a continuación del mapa de bits primario viene el mapa de bits secundario. Del mismo modo, para el tercer mapa de bits, se utiliza la primera posición del mapa de bits secundario.

El mapa de bits se puede transmitir como un dato binario de 8 bytes, o como un campo de 16 caracteres hexadecimales 0-9, A-F en el set de caracteres ASCII o EBCDIC.

c) Elemento de Datos

Los Elementos de Datos son los campos individuales que llevan la información propia de la transacción, algunos de los cuales pueden tener subcampos. Hay 128 campos definidos en el estándar ISO8583:1987, y 192 en posteriores versiones. La revisión de 1993 agregó nuevas definiciones, eliminó algunas pero sin embargo dejó el formato del mensaje sin cambios.

Mientras que cada Elemento de Datos tiene un significado y formato específico, el estándar también incluye algunos campos de propósito general y algunos especiales para sistemas o países, los cuales varían sustancialmente en su forma y uso de una implementación a otra.

Cada campo se describe en un formato estándar que define el contenido permitido del campo (numérico, binario, etc.) y el largo del campo (variable o fijo), de acuerdo a la Tabla N° 1.6.

Tabla N° 1.6.- Significado de los tipos de datos

Abreviatura	Significado
a	Alfanumérico, incluyendo espacios
n	Sólo valores alfanuméricos
s	Sólo caracteres especiales
an	Alfanumérico
as	Sólo caracteres alfanuméricos y especiales
ns	Sólo caracteres numéricos y especiales
ans	Caracteres alfabéticos, numéricos y especiales
b	Información binaria
z	Tracks 2 y 3 code set como se define en la ISO 4909 y ISO 7813.

Además, cada campo puede tener largo fijo o variable. Si es variable, el largo del campo será precedido por un indicador de largo, como se muestra en la Tabla N° 1.7.

Tabla N° 1.7.- Tipo de longitud de datos

Tipo	Significado
Fixed	Largo Fijo
LLVAR o (..xx)	Donde xx < 100, significa que los dos primeros dígitos indican el largo del campo
LLLVAR o (...xxx)	Donde xx < 1000, significa que los tres primeros dígitos indican el largo del campo
Un campo LLVAR o LLLVAR puede ser comprimido o ASCII dependiendo del formato del mensaje que puede ser ASCII o Comprimido.	Por ejemplo un campo LLVAR puede tener 1 o 2 bytes, si está comprimido el hexa '23x significa que hay 23 elementos, si es ascii, bytes '32x, '31x significa que hay 21 elementos. Un elemento depende del tipo de dato, si es numérico este estará comprimido, ej. largo 87 se representará por un byte '87x, si es ASCII serán dos bytos '38x y '37x. Los campos LLLVAR usan 2 o 3 bytes (dependiendo del tipo de mensaje) con un '0' adelante si es comprimido.

1.3.2 Ejemplo de Mensaje ISO 8583

Tenemos el siguiente Mensaje ISO8583:

```
0400202000000080000000000000000121292501
```

el que está divido así:

- MTI(4 dígitos) : 0040 (Requerimiento de reverso)
- Bitmap(8 bytes o 16 caracteres Hexadecimales) 20200000 00800000

Luego se separa esta información y se convierte en binario para obtener qué bit de la cadena es verdadero, teniendo los bits verdaderos presentes en el bitmap, se puede

conocer qué elementos de datos acompañan al mensaje. Se hace la separación y se convierte la información, quedaría lo mostrado a continuación:

```
20 = 0010 0000
20 = 0010 0000
00 = 0000 0000
00 = 0000 0000
00 = 0000 0000
80 = 1000 0000
00 = 0000 0000
00 = 0000 0000
```

De esto, los campos que están presentes en el mensaje son: 3, 11 y 41 (contando de izquierda a derecha las posiciones de los 1s al poner el mapa de bits como una tira binaria). Revisando los campos del estándar (ver ANEXO A), se tiene que, el campo 3, es un campo numérico de 6 posiciones, el campo 11 es un campo numérico de 6 posiciones, y el campo 41 es un campo ans, es decir que acepta caracteres alfanuméricos, numéricos y caracteres especiales y tiene una longitud de 8. Con esta información se obtienen los siguientes elementos:

- Campo 3 (Código Procesamiento)=000000
- Campo11 (Número de Rastreo del Sistema para auditoria)=000001
- Campo41 (Identificación de la terminal que acepta la tarjeta)=21292501

En este capítulo se han revisado las definiciones preliminares, conceptos y estándares que forman parte de un sistema de medios de pago. En el siguiente capítulo se mostrarán y describirán los elementos específicos de un sistema de medios de pago, con las características definidas para los objetivos de este informe.

CAPÍTULO II

ARQUITECTURA DE LA SOLUCIÓN

En este capitulo se dará el detalle de una configuración de hardware y software que soporte los requisitos de capacidad, rendimiento y disponibilidad que tienen estos sistemas en un entorno real de producción.

2.1 Requerimientos Técnicos

Actualmente existen 2 redes de adquirencia en el país, **Visanet**, que opera las transacciones con marca VISA y American Express, y la red de **MC Procesos**, que opera transacciones de las marcas MasterCard y marcas privadas (Ripley, Saga, Ace Home Center, etc.).

Como referencia la red de Visanet tiene la siguiente cobertura:

- Número de comercios afiliados: ~50,000 (ver http://www.visanet.com.pe/cgi-bin/scripts/dire_com.cgi)
- Número de TPVs en la red: ~55,000
- Número de transacciones mensuales: ~8 millones

Para cumplir con estos niveles de operación es necesario que el sistema cumpla con los siguientes requisitos:

- Número de transacciones por segundo: ~35 TPS
- Disponibilidad de hardware : 99.997 %

2.2 Alcance de la Solución

La solución tratada en este informe, se refiere a la implementación de un sistema distribuido de medios de pago, configurando el elemento del Switch (o Núcleo) Transaccional descrito en las secciones previas, en la función de Adquirente, por medio de software de código abierto y tecnologías de software y hardware de amplio uso, en contraste con soluciones especializadas y cerradas tanto de hardware/software (como Stratus/VOS y Tandem Himalaya por ejemplo) que hasta hace poco han sido dominantes en la industria de procesamiento de medios de pago.

Esta solución se basa en las siguientes tecnologías:

- Arquitectura Intel x86: es la arquitectura de computadores con mayor popularidad en el mercado, basada en un modelo CISC
- Sistema Windows Server: Como sistema operativo de los servidores que soportarán el switch transaccional y a las bases de datos. No se ahondará en los detalles de los sistemas operativos sino que se darán recomendaciones sobre que distribuciones a utilizar y detalles importantes de su configuración
- Switch Transaccional, implementado con librerías JPOS, que es una librería/framework de JAVA, para aplicaciones de misión crítica, que cumple con el estándar ISO8583, y que puede ser adaptada y extendida para procesar transacciones financieras.
- Base de Datos en cluster, donde el sistema guarda información histórica de las transacciones que han pasado por el Switch. Esta información es transferida luego a los sistemas de backoffice, para la ejecución de procesos posteriores como liquidación de cuentas e intercambio de comisiones entre las diversas entidades que participan del proceso. La configuración en cluster permite alta disponibilidad.

2.3 Flujo de datos del sistema como Adquirente

A continuación se describe técnicamente el flujo de los mensajes de proceso de las transacciones, dentro del sistema de adquirencia.

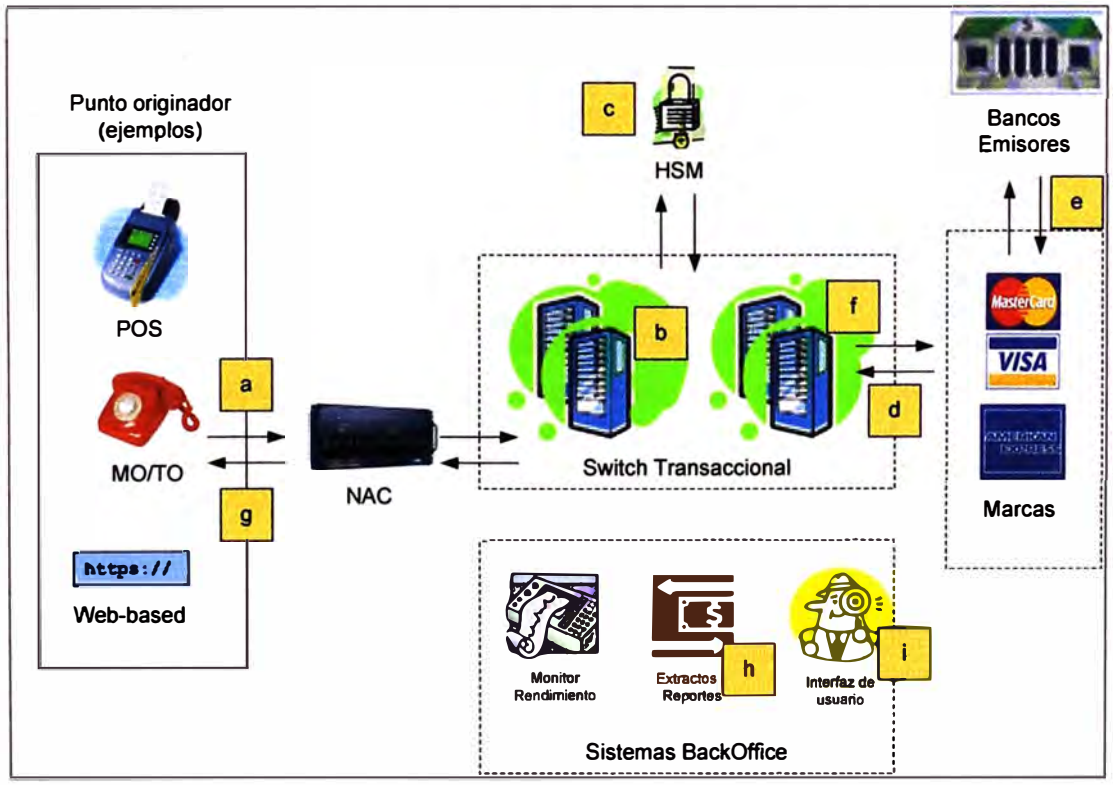

Figura 2.1.- Flujo de datos del sistema adquirente

a) Se reciben las transacciones en un mensaje de pedido de autorización, desde el punto originador, a través de un equipo NAC que brinda la protección contra accesos no autorizados hacia las redes internas del adquirente/procesador
b) El switch transaccional acepta el mensaje y lo procesa de acuerdo a las reglas de encaminamiento
c) Si es una transacción de débito, el switch envía el mensaje al HSM para desencriptar el PIN (clave de débito)
d) El mensaje se reenvía hacia el autorizador, formateado de acuerdo a las especificaciones de las marcas, a través de los sistemas gateway que las marcas proveen
e) Los sistemas de las marcas reenvían el mensaje hacia el emisor de la tarjeta para su autorización
f) Los mensajes son guardados en los logs de transacciones del switch
g) La respuesta es formateada y enviada hacia el originador de la transacción
h) Proceso posteriores del lado del adquirente, extraen la información de los logs del switch, para alimentar sistemas backoffice de reportes, liquidación, etc.
i) La información en los sistemas backoffice es proporcionada hacia los usuarios a través de aplicativos

2.4 Arquitectura de la Solución

A continuación se describe a detalle, la configuración hardware-software diseñada para cumplir los requisitos mencionados en los requisitos mencionados en el capítulo 2.1.

Esta configuración está implementada bajo los siguientes conceptos:

- Un Cisco Content Services Switch ('CSS') como "fronter" de dos o más servidores de aplicación usado para hacer un balance de carga de tráfico de manera automática.
- Servicios redundantes JPOS implementados en dos o más servidores de aplicación (configurados en modo multi-nodo). Cada servidor se dimensiona para ser capaz de soportar el 100% de las transacciones en cualquier momento.
- La base de datos es implementada en dos o más servidores como un "cluster virtual". La aplicación jPOS se conecta a la dirección del servidor virtual. La conmutación desde el servidor primario al secundario en caso de fallas, se da sin interrupción del servicio.
- La base de datos en sí está alojada en hardware como EMC Symmetrix, el cual es completamente tolerante a fallas de diseño de hardware con redundancia interna.

Este producto soporta totalmente operaciones no-disruptivas (por ejemplo, cambios de configuración o upgrades de software sin interrupción de servicio).

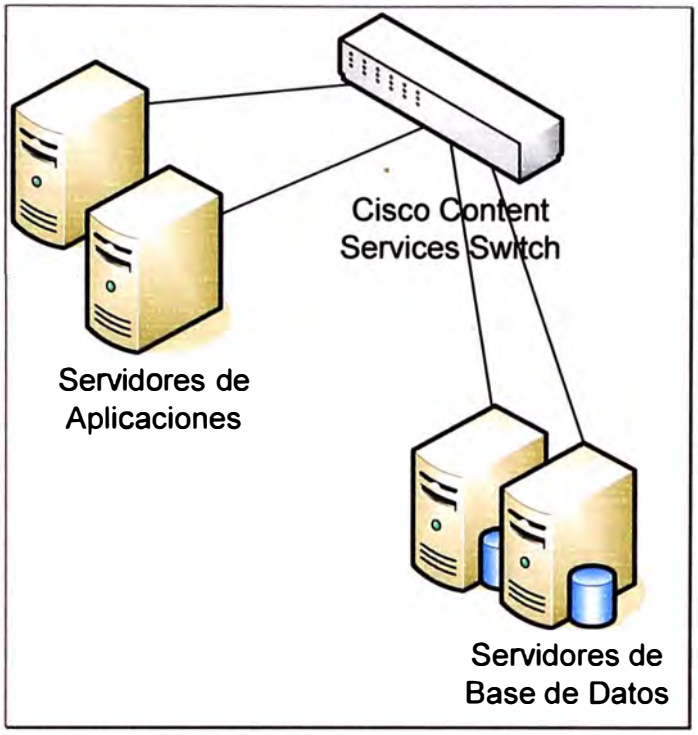

Figura 2.2.- Switch Transaccional

En la Figura 2.2 se muestra un esquema conceptual de la configuración de estos elementos

2.4.1 Configuración de Hardware

Estos requerimientos técnicos se pueden cumplir con una configuración de hardware similar a la mostrada el la Tabla N° 2.1.[2]

Tabla N° 2.1.- Configuración de Equipamiento recomendado en entorno de producción

Cantidad	Función	Característica	Configuración
1	**Conmutador de Servicios de Contenido**	CISCO CSS 11501	
2	**Servidor de Aplicaciones**	Procesador	8-way Intel Xeon
		Veloc. Procesador	3.2 GHz
		Memoria	8 GB
		Discos	RAID 1 (2 discos x73 GB) (para el sistema operativo y el contenedor de aplicaciones)
		Sistema Operativo	Windows Advanced Server 2003 Enterprise

[2] Ver referencia bibliográfica [2]

			Edition
		Fuente de Poder	Redundante
2	**Servidor de Base de Datos**	Procesador	4-way Intel Xeon Hyper-threaded para correr como 8 CPUs
		Veloc. Procesador	3.2 GHz
		Memoria	8 GB
		Discos	EMC Symmetrix fault tolerant storage, para almacenar la base de datos.
		Sistema Operativo	Windows Advanced Server 2003 Enterprise Edition
		Base de Datos	MS SQL Server 2005 Enterprise Edition, en configuración cluster
		Fuente de poder	Redundante

Los servidores de aplicación replicados proporcionan redundancia de hardware y la capacidad de instalar actualizaciones de software sin tiempos de parada de servicio.

Se podría modificar a RAID 5 (3 discos x 73 GB) si son necesarios ejecutar procesos extensivos de extracción de información

Esta configuración es usada en producción en la procesadora de medios de pago On-line Strategies (OLS) http://www.olsdallas.com/index.html en su proyecto de bandera OLS.Switch.

2.4.2 Rendimiento de la configuración

Andy Orrock, el actual Jefe de Operaciones (COO) de On-Line Strategies (OLS), publica en su blog site (http://www.andyorrock.com), datos sobre el rendimiento de la solución de procesamiento transaccional de medios de pago, en una configuración similar a la mostrada en la sección anterior.[3]

Los servidores de aplicación en esta configuración soportan 750,000 transacciones diarias, con menos de 8% de uso de CPU, medido sostenidamente por un periodo de media hora. El uso de CPU de los servidores de Base de Datos es menor a 5%. El número de transacciones por segundo pico (Peak TPS), observado en días de alto tráfico, es de 45 TPS sobre un periodo de 30 minutos

[3] Ver referencia bibliográfica [3] y [4]

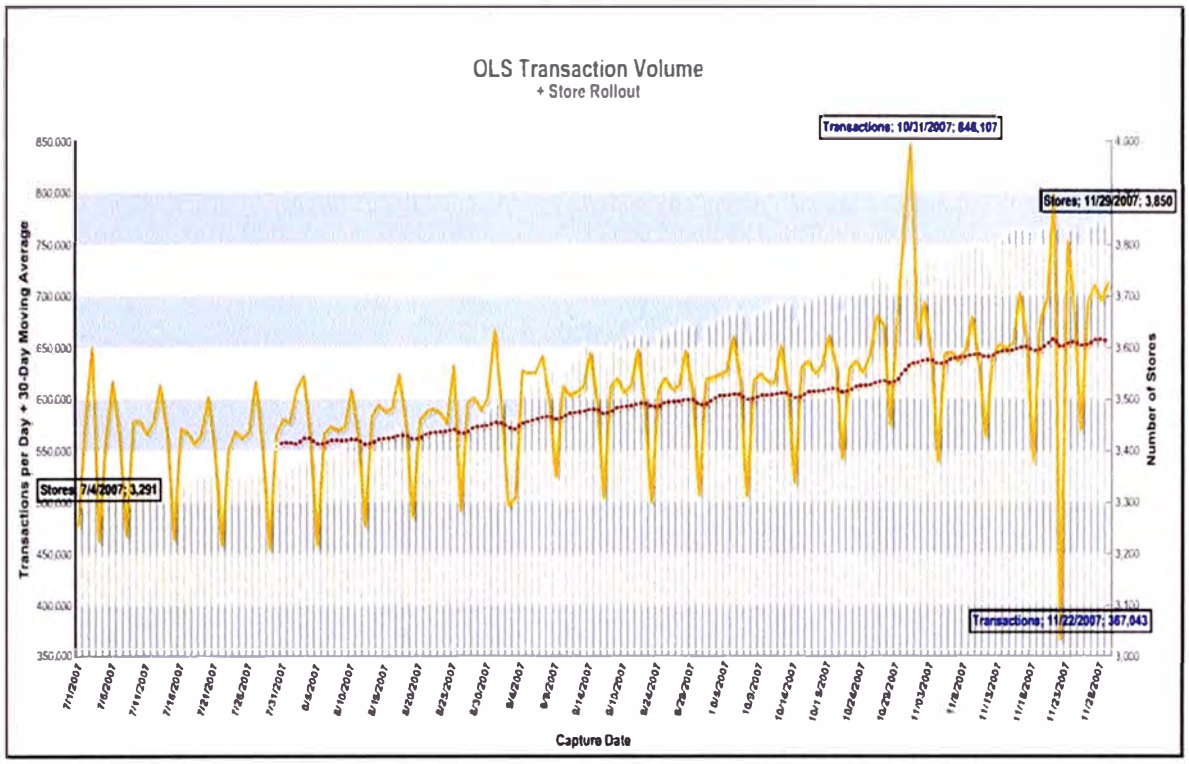

Figura 2.3.- Número de transacciones diarias

La forma de la gráfica de la Figura 2.3 con picos y valles, es una tendencia típica de los sistemas de medios de pago, y responde a las tendencias de los consumidores, donde los picos se dan los fines de semana, viernes y sábado, o antes de un feriado importante; y los valles se dan generalmente los domingos o los mismos días feriados.

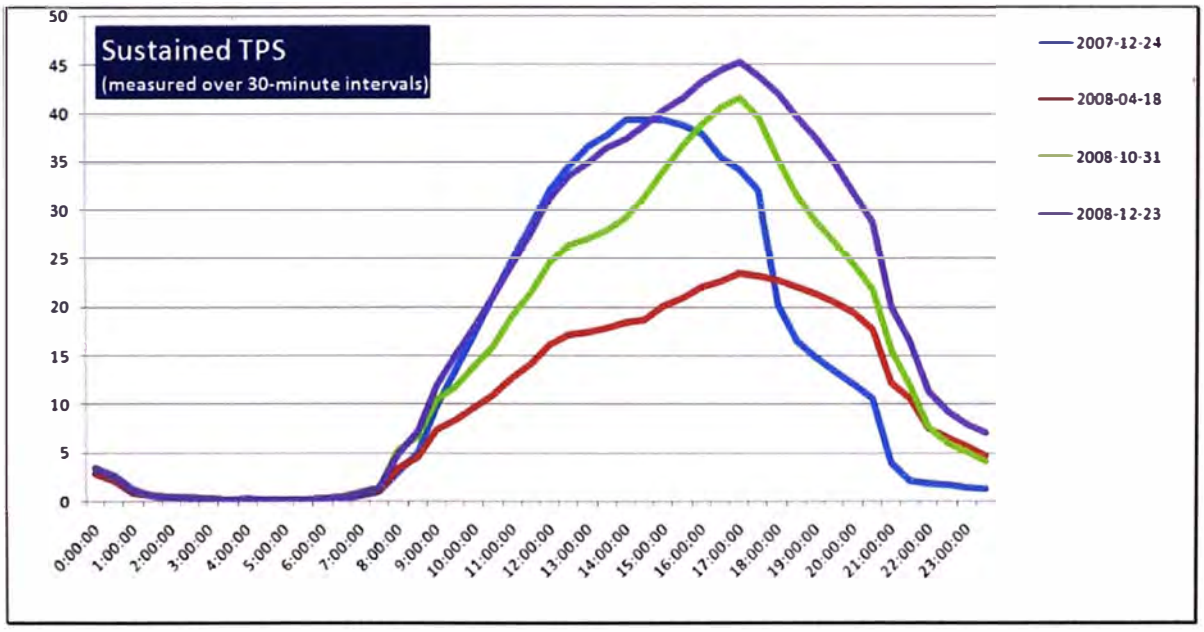

Figura 2.4.- TPS en días de alto tráfico

Del mismo modo se explica la forma de la gráfica de la Figura 2.4, donde los mayores consumos se dan en horas de la tarde, mostrando los picos entre las 14:00 y 18:00 horas.

En este capítulo se han definido los requerimientos técnicos de los sistemas de medios de pago y se ha mostrado una arquitectura que cumple con estos requerimientos. En el siguiente capítulo se detallará la solución basada en esta arquitectura y se darán los resultados de la ejecución de una simulación de funcionalidad y rendimiento.

CAPÍTULO III

IMPLEMENTACIÓN DE LA SOLUCIÓN

En este capitulo se describirán detalladamente los elementos de la arquitectura mostrada en el capítulo anterior, con las consideraciones necesarias para cumplir los requerimientos definidos y se hará una simulación de la funcionalidad y del rendimiento, configurando un sistema de pruebas para realizar transacciones de venta, mostrando los archivos log de esta simulación.

A continuación se describen los diversos elementos que forman parte de la arquitectura de la solución. Se tomará mayor importancia a la descripción del núcleo del sistema, es decir el software que implementa el Switch Transaccional.

3.1 Conmutador de Servicios de Contenido

Es un dispositivo de red que ejecuta la función de switch en las capas 4 a 7 del modelo OSI[4]. Un conmutador de servicios de contenido, no sólo proporciona una red con conectividad básica a través de puertos Ethernet y giga Ethernet, sino que principalmente dirige tráfico, dentro de un centro de datos, o a través de múltiples datacenters, basado en la información de capas 4 a 7, contenida dentro de las peticiones de usuario entrantes. Además de analizar totalmente las peticiones entrantes, este equipo está evaluando continuamente recursos disponibles de los servidores, tal que el flujo del tráfico es optimizado de acuerdo a las condiciones de carga actuales del centro de datos o website. A diferencia de un switch de capa 2-3, un switch de contenidos realiza sus decisiones de encaminamiento basados en términos como una URL definida, una cookie de contenido, el tipo de navegador y preferencias de lenguaje. El modelo a utilizar es un Cisco CSS 11501 Content Services Switch, mostrado en la Figura 3.1, provisto por Cisco Systems, y que tiene las características mostradas en la Tabla N° 3.1.

Cisco CSS 11501

Figura 3.1.- Conmutador de Servicios de Contenido

[4] Ver referencia bibliográfica [5]

Tabla N° 3.1.- Características Técnicas de Conmutador de Servicios de Contenido

	Características
Modelo	Cisco CSS11501S-C
Puertos Giga Ethernet	1
Puertos Ethernet	8
SSL y módulos de compresión	Sí, integrado
Características de redundancia	Active-active Layer 5 ASR Virtual internet protocol address (VIP) redundancy
Ancho de banda	6 Gbps
Opciones de almacenamiento	1GB memoria flash
Rendimiento	1400 SSL transacciones por segundo y 250 Mbps de encriptación masiva (ARC4). 500 Mbps capacidad de compresión debido al uso de tarjetas integradas con state-of-the-art chips de criptología y compresión.

3.2 Terminal de Punto de Venta (TPV)

La solución implementada puede soportar cualquier modelo de TPV que utilice ISO 8583 sobre TCP/IP como protocolo de comunicación con el Swtich de Transacciones, y que tenga capacidad para lectura de banda magnética.

Para la demostración de funcionalidad, sin embargo, se utilizará un TPV virtual, mostrado en Figura 3.2, desarrollado en software Visual Basic 6.0, realizado por la Dirección de Outsourcing y DataCenter de Telefónica del Perú, y que simulará transacciones de venta.

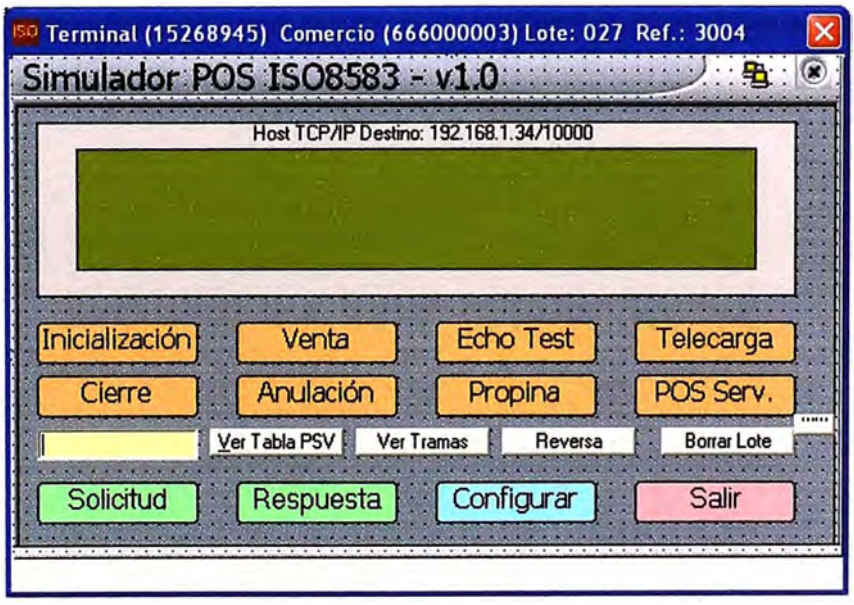

Figura 3.2 - Pantalla principal de simulador TPV

Las funcionalidades a utilizar son:

- **Inicialización**: envía una trama de inicialización al switch para realizar las demás transacciones.
- **Venta:** es el envío de la trama de venta en sí, que contiene la información de la misma.
- **Venta denegada:** se simulará una condición para denegar la venta.

Asimismo, para las revisiones de rendimiento, se utilizará una modificación de este TPV, para el envío de transacciones masivas, simulando varios TPV al mismo tiempo.

3.3 Clústers en SQL Server

Los clústeres de conmutación por error de SQL Server proporcionan alta disponibilidad para una instancia completa de SQL Server. [5] Los clústeres de conmutación por error de SQL Server se crean sobre los de Windows Server. Para crear un clúster de conmutación por error de SQL Server, primero debe crear el clúster de conmutación por error de Windows Server subyacente.

Un clúster de conmutación por error de SQL Server, denominado también instancia del clúster de conmutación por error, consta de los siguientes elementos:

Uno o varios nodos de clústeres de conmutación por error de Windows Server

Un grupo de recursos del clúster dedicado para el clúster de conmutación por error de SQL Server que contiene lo siguiente:

- Nombre de red para tener acceso al clúster de conmutación por error de SQL Server
- Direcciones IP
- Discos compartidos utilizados para la base de datos y el almacenamiento de registros de SQL Server
- Archivos DLL de recursos que controlan el comportamiento de conmutación por error de SQL Server
- Claves del Registro con punto de comprobación que se mantienen sincronizadas automáticamente en los nodos de clústeres de conmutación por error

Una instancia del clúster de conmutación por error de SQL Server aparece en la red como una única instancia de SQL Server en un solo equipo. Internamente, solamente uno de los nodos es propietario del grupo de recursos del clúster cada vez y es el encargado de atender todas las solicitudes del cliente de esa instancia del clúster de conmutación

---

[5] Ver referencia bibliográfica [6]

por error. En caso de se produzca un error (errores de hardware, errores del sistema operativo o errores de aplicación o servicio) o se realice una actualización planeada, la propiedad del grupo se mueve a otro nodo del clúster de conmutación por error. Este proceso se denomina conmutación por error. Gracias a la funcionalidad de clúster de conmutación por error de Windows Server, el clúster de conmutación por error de SQL Server proporciona alta disponibilidad a través de la redundancia en el nivel de instancia.

3.4 Switch Transaccional

Describimos el componente de software que provee la funcionalidad de switch transaccional, implementado sobre los 2 servidores descritos en la sección 2.4.1.

Este componente se basa en una librería de clases Java, llamada JPOS, que sirve para construir un aplicativo con la funcionalidad de switch requerida. Sus componentes principales se describen en las secciones siguientes, de acuerdo a la Guía del Programador JPOS. [6]

3.4.1 ISOMsg: Manejo de mensajes ISO 8583

La representación interna del mensaje ISO 8583 se maneja usualmente con el objeto ISOMsg, el cual parte de una clase base llamada ISOComponent

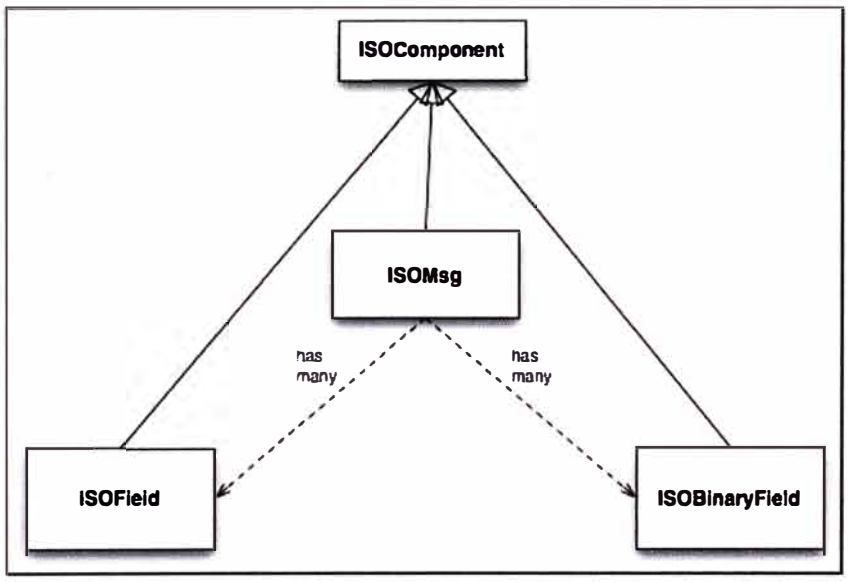

Figura 3.3.- Jerarquía de Clases de ISOMsg

El intercambio de mensajes relacionados con transacciones mercantiles son las bases de las redes financieras mundiales. Estos mensajes tienen múltiples significados y vienen en diferentes formatos, y en muchos casos, estos mensajes son transportados de una red a

---

[6] Ver referencia bibliográfica [7]

otra en diferentes protocolos. Estos protocolos y mensajes conforman una serie de normas que deben ser seguidos por una institución si desea participar exitosamente en esta actividad financiera.

Las transacciones pueden ser procesadas de acuerdo con la información almacenada en bases de datos SQL y pueden ser transportadas hacia otras redes en el mismo formato de mensaje que fueron recibidas o en uno totalmente diferente. Las transacciones pueden venir de muchas diferentes fuentes, terminales de punto de venta (TPV), cajeros automáticos (ATM), otras redes u otras computadoras como servidores de comercio electrónico para Internet, B2B, centros de llamadas, IVR, entre otros.

La jerarquía de clases de para el manejo de mensajes ISO 8583 se muestra en la Figura 3.3.

Ejemplo de uso

Los métodos de ISOComponent son los siguientes:

```
public abstract class ISOComponent implements Cloneable {
   public void set (ISOComponent c) throws ISOException;
   public void unset (int fldno) throws ISOException;
   public ISOComponent getComposite();
   public Object getKey() throws ISOException;
   public Object getValue() throws ISOException;
   public byte[] getBytes() throws ISOException;
   public int getMaxField();
   public Hashtable getChildren();
   public abstract void setFieldNumber (int fieldNumber);
   public abstract void setValue(Object obj) throws ISOException;
   public abstract byte[] pack() throws ISOException;
   public abstract int unpack(byte[] b) throws ISOException;
   public abstract void dump (PrintStream p, String indent);
   public abstract void pack (OutputStream out) throws IOException, ISOException;
   public abstract void unpack (InputStream in) throws IOException, ISOException;
}
```

ISOMsg extiende esta clase e implementa nuevos métodos para facilitar el manejo de los mensajes ISO, por ejemplo, para asignar valor a los campos de un mensaje:

```
public void set (int fieldNumber, String fieldValue);
```

El siguiente código puede ser usado para crear una representación interna de un mensaje ISO 0800.

```
import org.jpos.iso.*;
   ISOMsg m = new ISOMsg();
   m.set (new ISOField (0, "0800"));
   m.set (new ISOField (3, "000000"));
   m.set (new ISOField (11, "000001"));
   m.set (new ISOField (41, "29110001"));
   m.set (new ISOField (60, "jPOS 6"));
```

```
m.set (new ISOField (70, "301"));
```

3.4.2 ISOPackager: Empaquetamiento-desempaquetamiento de mensajes ISO8583

Como se ha visto en la sección 3.4.1, ISOComponent tiene dos métodos para manejar los mensajes ISO: pack y unpack. El método pack retorna una representación binaria del componente dado (que puede ser un campo o todo el mensaje), el método unpack hace lo contrario devolviendo el número de bytes consumidos.

Pero, JPOS tiene la clase ISOPackager que se puede utilizar en conjunto con ISOMsg y que flexibiliza el manejo de conversión de protocolos.

En ISOMsg existe el siguiente método:

```
public void setPackager (ISOPackager p);
```

por lo tanto a un ISOMsg se le puede asignar un ISOPackager dado, de tal manera:

```
ISOPackager customPackager = MyCustomPackager ();
ISOMsg m = new ISOMsg();
m.setMTI ("0800");
m.set (3, "000000");
m.set (11, "000001");
m.set (41, "29110001");
m.set (60, "jPOS 6");
m.set (70, "301");
m.setPackager (customPackager);
byte[] binaryImage = m.pack();
```

Lo que hace el código mostrado es empaquetar el mensaje ISO de acuerdo a las reglas definidas en una clase ISOPackager llamada customPackager (es un ejemplo).

JPOS viene con una serie de clases derivadas de una clase base ISOPackagerBase, para implementar el empaquetamiento-desempaquetamiento de diferentes protocolos basados en ISO 8583, que son utilizados por las entidades financieras a nivel mundial.

Entre estas clases están, por ejemplo:

- BASE24Packager: usado para sistemas Base24, una versión de ISO 8583 usada generalmente en sistemas Tandem (plataforma propietaria para switch transaccional)
- ISO87APackager: codificación ASCII de ISO 8583 v1987
- Base1Packager: usado en sistemas Base1, por VISA a nivel mundial
- XMLPackager, empaqueta y desempaqueta ISOMsgs utilizando formato XML

Con estas clases se puede implementar convertidores de protocolos, con un código similar al siguiente:

```
ISOPackager packagerA = ISO93APackager();
ISOPackager packagerB = ISO87APackager();
ISOMsg m = new ISOMsg();
m.setPackager (packagerA);
m.unpack (binaryImage);
m.setPackager (packagerB);
byte[] convertedBinaryImage = m.pack();
```

En este ejemplo se convierte el mensaje de formato ISO 8583 versión 1993 a versión 1987. Adicionalmente, JPOS implementa una clase "packager" genérica llamada GenericPackager, que puede ser configurada a través de un archivo XML, de la siguiente manera:

```
import org.jpos.iso.ISOPackager;
import org.jpos.iso.packager.GenericPackager;
import org.jpos.iso.packager.ISO87BPackager;
ISOPackager p = new GenericPackager ("iso87binary.xml");
```

En este ejemplo el archivo iso87binary.xml tiene la definición de los campos del mensaje. De esta manera se utiliza para configurar la simulación de la sección 3.6.

3.4.3 ISOChannel: Manejo del protocolo de Red

JPOS utiliza la interface llamada ISOChannel para encapsular detalles de los protocolos de comunicación de red. ISOChannel se utiliza para enviar y recibir objetos ISOMsg.

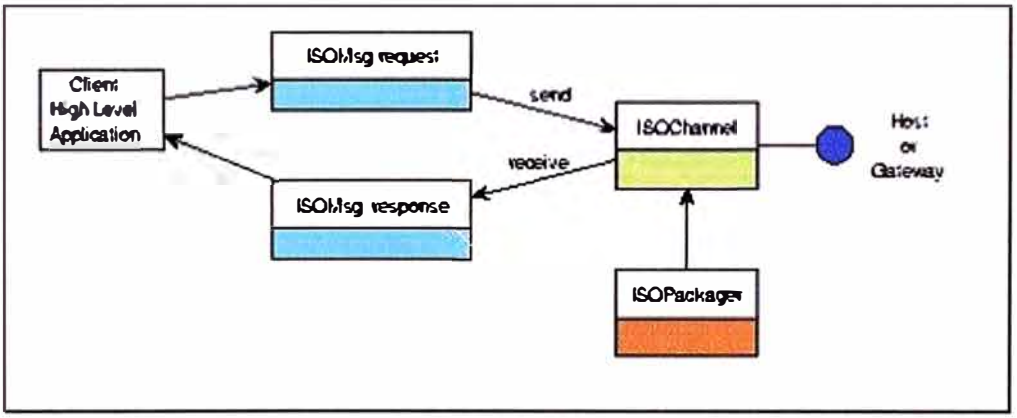

Figura 3.4.- Funcionamiento del ISOChannel

En la Figura 3.4 se muestran las relaciones de ISOChannel con otras clases JPOS. Los diversos elementos de un sistema de medios de pago, que se encuentran distribuidos, se comunican con el switch por diferentes protocolos de red (tcp/ip, x25, etc), y dentro de estos mismos, diversos protocolos de comunicación a nivel de aplicación, ISOChannel es capaz de usar diferentes protocolos de comunicación, a través de varias implementaciones ya construidas, algunas de las cuales se muestran en la Tabla N° 3.2.

Tabla N° 3.2.- Algunas implementaciones de ISOChannel

Nombre	Descripción
ASCIIChannel	4 bytes para la longitud del mensaje + datos de ISO8583
LogChannel	Puede ser usado para leer los repositorios log de JPOS (donde se almacena información de las transacciones procesadas), y enviarlas a otro canal
PADChannel	Usado para conectar redes X25
XMLChannel	Representa los mensajes ISO8583 en formato XML
BASE24Channel	Usado para conectarse con sistemas BASE24 sobre enlaces X.25
NACChannel	Usado para conectar con sistemas que implementan un punto de control de acceso (NAC por sus siglas en inglés)

Ejemplo de uso

Este ejemplo envía un mensaje 0800 a un servidor de echo local

```
import org.jpos.iso.*;
import org.jpos.util.*;
import org.jpos.iso.channel.*;
import org.jpos.iso.packager.*;
public class Test {
  public static void main (String[] args) throws Exception {
  Logger logger = new Logger();
  logger.addListener (new SimpleLogListener (System.out));
  ISOChannel channel = new ASCIIChannel (
      "localhost", 7, new ISO87APackager()
       );
   ((LogSource)channel).setLogger (logger, "test-channel");
   channel.connect ();
   ISOMsg m = new ISOMsg ();
    m.setMTI ("0800");
    m.set (3, "000000");
    m.set (41, "00000001");
    m.set (70, "301");
    channel.send (m);
    ISOMsg r = channel.receive ();
    channel.disconnect ();
  }
}
```

Si bien es cierto que este ejemplo funciona, JPOS implementa una facilidad llamada Q2, que es un ensamblador/configurador de componentes y que se explica más adelante en la sección 3.4.7.

3.4.4 ISOServer: Control de conexiones

ISOServer es un componente que escucha un determinado puerto de entrada por conexiones entrantes y pasa el control a una implementación ISOChannel subyacente. Una vez que una conexión es aceptada y es creada una instancia de ISOChannel, se crea un Thread, utilizado para ejecutar múltiples tareas en el procesador, a través de la clase ThreadPool. Luego estos mensajes son pasados a una implementación de ISORequestListener para su tratamiento.

En el siguiente ejemplo vemos el código de un servidor que escucha por el puerto 8000 mensajes ISO empaquetados en XML, y que responde aprobando la transacción en el campo 39 del mensaje ISO.

```
import java.io.*;
import org.jpos.iso.*;
import org.jpos.util.*;
import org.jpos.iso.channel.*;
import org.jpos.iso.packager.*;
public class Test implements ISORequestListener {
    public Test () {
    super();
}

public boolean process (ISOSource source, ISOMsg m) {
   try {
   m.setResponseMTI ();
   m.set (39, "00");
   source.send (m);
   } catch (ISOException e) {
       e.printStackTrace();
   } catch (IOException e) {
       e.printStackTrace();
   }
   return true;
}

public static void main (String[] args) throws Exception {
    Logger logger = new Logger ();
    logger.addListener (new SimpleLogListener (System.out));
    ServerChannel channel = new XMLChannel (new XMLPackager());
    ((LogSource)channel).setLogger (logger, "channel");
    ISOServer server = new ISOServer (8000, channel, null);
    server.setLogger (logger, "server");
    server.addISORequestListener (new Test ());
    new Thread (server).start ();
}
```

3.4.5 MUX: Manejo múltiples mensajes de entrada

La implementación de un adquirente a través de JPOS necesita recibir múltiples conexiones de los terminales TPV, a un mismo tiempo, y enrutar éstas a las entidades

emisoras a través de ISOChannel, como se ve en la Figura 3.5. Si bien es cierto se puede abrir una conexión socket para cada transacción, lo común es configurar una conexión socket (manejada por una sola instancia de ISOChannel), y multiplexarla. MUX es una interface, que funciona como un multiplexador de canales ISOChannel.

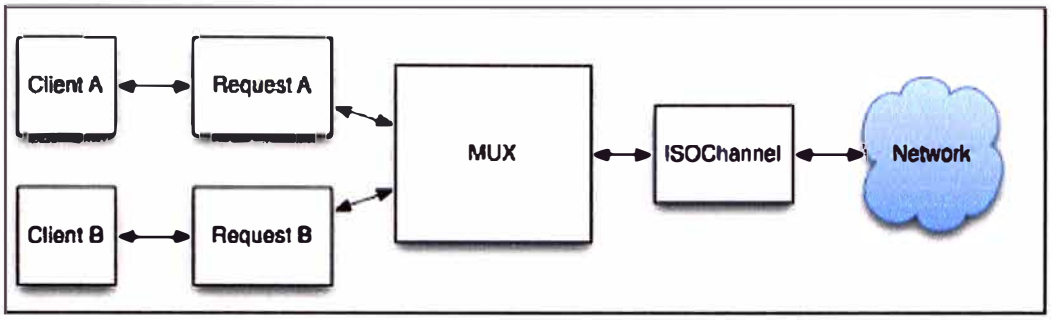

Figura 3.5.- Esquema de componente MUX

La interface MUX es la siguiente:

```
public interface MUX {
   public ISOMsg request (ISOMsg m, long timeout) throws ISOException;
   public void request
       (ISOMsg m, long timeout, ISOResponseListener r, Object handBack)
        throws ISOException;
   public boolean isConnected();
}
```

MUX tiene 2 implementaciones, ISOMux y QMUX, la última utilizada en el entorno Q2
El segundo método request, que implementa una manera asíncrona de llamado y pasa el control a un ISOResponseListener, sólo está implementado en QMUX, que se verá en la sección 3.4.7

3.4.6 JPOS Space: Manejo de colas

JPOS Space es un modelo de comunicación inter-componente, usado para transferir objetos del tipo ISOMsg entre los componentes del sistema JPOS. Se puede conceptualizar como un modelo de colas, donde un componente coloca un mensaje en una cola y otro componente lee la cola y captura el mensaje.
Se compone de 3 operaciones básicas:

- void out (Object key, Object value)
  Coloca un objeto dentro de la cola identificada por la llave (key). Si ya existe un objeto en la cola, el nuevo objeto es colocado al final de la misma.
- Object rd (Object key)
  Lee un objeto de la cola identificada por la llave. Se bloquea hasta que un objeto este presente.

- Object in (Object key)

  Igual que el anterior, pero borra el objeto de la cola una vez que lo lee.

La interface de JPOS Space es la siguiente:

```
public interface Space {
  public void out (Object key, Object value);
  public void out (Object key, Object value, long timeout);
  public Object in (Object key);
  public Object rd (Object key);
  public Object in (Object key, long timeout);
  public Object rd (Object key, long timeout);
  public Object inp (Object key);
  public Object rdp (Object key);
  public void push (Object key, Object value);
  public void push (Object key, Object value, long timeout);
}
```

Donde aparte de los tres métodos principales ya explicados se tienen los siguientes métodos útiles:

- void out (Object key, Object value, long timeout)

  Coloca un objeto en la cola utilizando un tiempo de expiración. La entrada es automáticamente removida cuando el tiempo vence.
- Object rd (Object key, long timeout)

  Espera un máximo de milisegundos para leer un objeto de la cola. Si el tiempo vence y no ha llegado ningún objeto a la cola retorna nulo.
- Object in (Object key, long timeout)

  Igual que el anterior, pero borra el objeto de la cola una vez que lo lee
- Object rdp (Object key)

  Lee un objeto de la cola si es que hay alguno. No se bloquea-
- Object inp (Object key)

  Igual que el anterior, pero borra el objeto de la cola una vez que lo lee
- void push (Object key, Object value)

  Igual que out, pero coloca el objeto al inicio de la cola
- void push (Object key, Object value, long timeout)

  Igual que el anterior pero con un tiempo de espera máximo

jPOS proporciona algunas implementaciones preconstruidas de colas usando Spaces:

- TSpace

  Implementación de colas en memoria
- JDBMSpace

Implementación de colas con persistencia, utilizando JDBM 2 (clase para acceso a base de datos) para implementar la persistencia

Ejemplo de uso: Coordinación entre clientes y servidores

Comunicar 2 procesos usando Spaces es simple, un proceso coloca objetos en la cola y el otro los extrae, pero si se tiene muchos procesos usando la misma cola, existe la alternativa de utilizar una clase para encapsular los mensajes, y así se puedan corresponder cada instancia cliente con su instancia servidor. Esta clase se llama TinySpace, y un cliente se podría codificar de la siguiente manera:

```
public ISOMsg process (ISOMsg m, long timeout) {
Space sp = SpaceFactory.getSpace ("tspace:myspace");
TinySpace ts = new TinySpace ();
ts.out ("Request", m);
sp.out ("QUEUE", ts);
return (ISOMsg) ts.in ("Response", timeout);
}
```

La parte servidor puede ser como sigue:

```
Space sp = SpaceFactory.getSpace ("tspace:myspace");
for (;;) {
Space ts = (Space) sp.in ("QUEUE");
ISOMsg m = (ISOMsg) ts.in ("Request");
...
... process request, prepare response
...
ts.out ("Response", m);
}
```

Como vemos se utiliza la cola "QUEUE" pero se encapsulan 2 tipos de mensajes ("request" y "response") utilizando la clase TinySpace.

3.4.7 Q2: Ensamblador de componentes

Q2 es un componente clave de la arquitectura de JPOS. Está implementando bajo el estándar Java JMX (ver Anexo C) que define una arquitectura de gestión para la administración/monitorización de aplicaciones y componentes basados en Java.

JPOS tiene una serie de componentes que deben ser configurados en el momento de iniciar la aplicación, y luego interconectados y monitoreados. El módulo principal (main()) de una aplicación típica implementada con JPOS, debe comenzar leyendo algún tipo de archivo de configuración y luego inicializar algunos componentes, arrancar algunos procesos (threads), inicializar conexiones con base de datos, etc. Q2 es la manera estándar de inicializar todo lo requerido. Este es el ciclo de vida de Q2 (init/ start/ stop/ deploy) que se implementa por medio de QBeans que son MBeans (ver Anexo C). Q2 se encarga de registrarlos dentro de un MBeanServer.

JPOS contiene un módulo llamado Q2Mod que sirve para correr JPOS dentro del contenedor Q2, y contiene adaptadores que proporcionan a los componentes existentes de JPOS el ciclo de vida (init/ start/ stop/ destroy) que Q2 requiere. Algunos de estos adaptadores son:

- ChannelAdaptor: es usado para configurar implementaciones ISOChannel existentes
- QMUX: es una implementación de un MUX que reemplaza a ISOMUX (antigua implementación de MUX)
- QServer: es usado para configurar implementaciones ISOServer existentes

Cada uno será revisado a continuación.

ChannelAdaptor

Es usado para configurar un cliente ISOChannel estándar manteniéndolo conectado a un host remoto. Adicionalmente, la comunicación del ChannelAdaptor con otros componentes se implementa a través de una interfaz basada en Spaces, que brinda la funcionalidad de tener más de un canal conectado a un host dado, y también la habilidad de ejecutar procesos distribuidos por medio de múltiples canales en varias máquinas Q2 configuradas en cluster. El ChannelAdaptor tiene los atributos y propiedades configurables, mostrados en la Tabla N° 3.3.

Tabla N° 3.3.- Atributos y propiedades del ChannelAdaptor[7]

Tipo	Nombre	Descripción
Atributo	Class	Nombre de la clase implementada por ISOChannel
Atributo	Packager	Nombre de la clase implementada por ISOPackager
Atributo	Logger	Nombre de la clase Logger, opcional
Atributo	Header	Cabecera del canal, es opcional
Propiedad	Host	Dirección o nombre del host remoto
Propiedad	Port	Puerto del host remoto
Propiedad	Max-packet-lenght	Longitud máxima del paquete, opcional, por defecto 100,000
Propiedad	Timeout	Timeout a nivel del socket, en milisegundos

Un ejemplo de configuración se muestra a continuación y puede ser colocado en un archivo XML:

```
<channel-adaptor name='sample-channel-adaptor'
  class="org.jpos.q2.iso.ChannelAdaptor" logger="Q2">
 <channel class="org.jpos.iso.channel.ASCIIChannel" logger="Q2"
  packager="org.jpos.iso.packager.GenericPackager"
  header="ISO016000055">
  <property name="host" value="192.168.0.1" />
  <property name="port" value="1234" />
```

[7] Para una lista completa ver referencia bibliográfica [1]

```
  <property name="packager-config" value="cfg/iso87ascii.xml" />
  </channel>
<in>sample-send</in>
<out>sample-receive</out>
<reconnect-delay>5000</reconnect-delay>
<space>tspace:default</space>
</channel-adaptor>
```

Las etiquetas <in> y <out> se refieren al punto de vista del ChannelAdaptor, es decir <in> o "entrada" al ChannelAdaptor se refiere a información realmente enviada a través del ISOChannel subyacente.

QMUX

QMUX es una implementación de MUX, para multiplexar canales ISOChannel. Utiliza colas Space para comunicarse con los canales subyacentes, lo que le da nuevas habilidades como hacer multiplexación de canales para balance de carga y redundancia, y adicionalmente, como estos canales no necesitan estar corriendo en la misma máquina, se pueden utilizar implementaciones distribuidas. Una configuración de ejemplo de QMUX es la siguiente.

```
<mux class="org.jpos.q2.iso.QMUX" logger="Q2" name="mymux">
<in>sample-receive</in>
<out>sample-send</out>
</mux>
```

Para manejar mensajes de entrada que no se corresponden a mensajes de salida (por ejemplo respuestas tardías, mensajes de administración de red originados por un servidor remoto, etc.) se puede configurar una cola opcional "unhandled", y QMUX enviará a esa cola cualquier mensaje de entrada que no se corresponde con algún mensaje pendiente, luego se puede configurar un proceso separado para manejar esta cola.

La configuración es similar a la siguiente:

```
<mux class="org.jpos.q2.iso.QMUX" logger="Q2" name="mymux">
  <in>sample-receive</in>
   <out>sample-send</out>
   <unhandled>mymux.unhandled</unhandled>
</mux>
```

QMUX también puede soportar múltiples request listeners:

```
<mux class="org.jpos.q2.iso.QMUX" logger="Q2" name="mymux">
<in>sample-receive</in>
<out>sample-send</out>
<unhandled>mymux.unhandled</unhandled>
<request-listener class="my.request.listener" logger="Q2" realm="myrealm">
<property name="myproperty" value="abc" />
<property name="myotherproperty" value="xyz" />
```

```
<property file="cfg/myprop.cfg" /> <!-- properties taken from a file -->
</request-listener>
</mux>
```

En este esquema, para corresponder las respuestas con las peticiones originales, QMUX utiliza una llave por defecto que corresponde a los campos ISO: MTI, campo 41 (terminal ID) y campo 11 (Serial Trace Audit Number). Se puede configurar esta regla con la etiqueta <key>, como se muestra.

```
<mux ...>
...
<key>11, 37, 41, 42</key>
</mux>
```

QServer

QServer es un servicio que encapsula la clase ISOServer. Un ejemplo de configuración es el siguiente:

```
<server name="xml-server" class="org.jpos.q2.iso.QServer" logger="Q2">
<attr name="port" type="java.lang.Integer">8000</attr>
<!-- optional server-socket-factory element
<server-socket-factory class="org.jpos.iso.SunJSSESocketFactory" logger="Q2">
<property name="keystore" value="/path/to/your/keystore.jks" />
<property name="clientauth" value="true" />
<property name="serverauth" value="true" />
<property name="storepassword" value="secret" />
<property name="keypassword" value="secret" />
<property name="addEnabledCipherSuite" value="SSL_RSA_WITH_3DES_EDE_CBC_SHA" />
<property name="addEnabledCipherSuite" value="TLS_DHE_DSS_WITH_AES_256_CBC_SHA" />
</server-socket-factory>
-->
<channel name="xml.channel"
class="org.jpos.iso.channel.XMLChannel"
packager="org.jpos.iso.packager.XMLPackager" logger="Q2">
</channel>
<request-listener class="my.request.Listener" logger="Q2">
<property name="my-property" value="ABC" />
<property name="my-other-property" value="XYZ" />
</request-listener>
</server>
```

3.5 Diseño del núcleo transaccional utilizando Q2

El diseño básico de un núcleo transaccional, que sólo responda a los mensajes recibidos, aprobándolos, se hará configurando un QServer dentro del contenedor Q2.

3.5.1 Creación QServer

En el directorio **modules** del árbol de directorios de JPOS, se debe crear la siguiente estructura de directorios:

- Serversimulator
  - cfg
  - deploy

3.5.2 Archivo de configuración del QServer

Se crea el archivo XML **05_serversimulator.xml** en el directorio **deploy** con el siguiente contenido.

```
<?xml version="1.0" ?>
<server class="org.jpos.q2.iso.QServer" logger="Q2"
 name="simulator_10000">
 <attr name="port" type="java.lang.Integer">10000</attr>
 <channel class="org.jpos.iso.channel.NACChannel"
      logger="Q2"                    packager="org.jpos.iso.packager.GenericPackager"
header="0000000000">
      <property name="packager-config" value="cfg/iso87binary.xml" />
 </channel>
 <request-listener class="org.jpos.bsh.BSHRequestListener" logger="Q2">
  <property name="source" value="cfg/serversimulator.bsh" />
 </request-listener>
</server>
```

Las definiciones contenidas en este archivo son las siguientes:

Puerto TCP/IP : Se configura el Puerto 10000 para el servidor

```
<attr name="port" type="java.lang.Integer">10000</attr>
```

Channel Mode : Se utiliza un channel mode para redes con dispositivos NAC

```
<channel class="org.jpos.iso.channel.NACChannel"
```

Packager: Se utiliza el Generic Packager, por lo tanto se debe contar con el archivo de definición del Package. En este caso se utilizará el formato ISO 8583 binario del 87.

```
      logger="Q2"                    packager="org.jpos.iso.packager.GenericPackager"
header="0000000000">
      <property name="packager-config" value="cfg/iso87binary.xml" />
```

En el directorio config se coloca el archivo iso87binary.xml el cual define el formato del mensaje a utilizar. Ver Anexo B.

3.5.3 Programa autorizador local

En el directorio cfg se crea un programa bsh (bean shell), con la lógica para que el QServer resuelva la transacción. Tomar en cuenta lo siguiente:

- Los campos 37 y 38 se generan aleatoriamente.
- La coloca la condición de denegación cuando el campo de importe sea igual a 99.99

```
message.setResponseMTI ();

Random random = new Random (System.currentTimeMillis());
message.set (37, Integer.toString(Math.abs(random.nextInt()) % 1000000));
message.set (38, Integer.toString(Math.abs(random.nextInt()) % 1000000));

if ("000000009999".equals (message.getString (4)))
   message.set (39, "01");
else
   message.set (39, "00");

source.send (message);
```

3.6 Simulación

La simulación del switch transaccional utilizando JPOS, se hará utilizando la configuración del QServer realizada en la sección 3.5. Además, para generar las transacciones que reciba el QServer se utilizará el TPV virtual descrito en la sección 3.2. El QServer debe recibir las transacciones, generar un mensaje de respuesta y enviárselo de retorno al TPV.

3.6.1 Configuración de TPV Virtual

Se configurará el TPV Virtual para enviar transacciones de inicialización y de Venta normal.

Los campos ISO 8583 a enviar en la inicialización son:

- Mapa de Bits: 3, 11, 24, 41, 60
- TPDU: 6000030000
- Tipo Mensaje: 0800
- Campo 03 Código de Procesamiento (900000)
- Campo 11 Numero de Trace del Sistema (009971)
- Campo 24 Identificación Internacional de la Red (0023)
- Campo 41 Identificación del Terminal (15268945)
- Campo 60 Versión Software (37)

Los campos ISO 8583 a enviar en la venta son:

- Mapa de Bits: 2, 3, 4, 11, 14, 22, 24, 25, 41, 42, 49, 60, 62
- TPDU: 6000030000
- Tipo Mensaje: 0200
- Campo 02 Número de tarjeta (4241370900000330)
- Campo 03 Código de Procesamiento (000000)

- Campo 04 Importe (generado en la prueba)
- Campo 11 Numero de Trace del Sistema (generado en la prueba)
- Campo 14 Fecha de expiración (1210)
- Campo 22 Modo de ingreso POS (0012)
- Campo 24 Identificación Internacional de la Red (0023)
- Campo 25 Código de Condición POS (00)
- Campo 41 Identificación del Terminal (15268945)
- Campo 42 Identificación del Comercio (666000003)
- Campo 49 Código de Moneda (804)
- Campo 60 Versión Software (000000037)
- Campo 62 Referencia y Lote ()

3.6.2 Ejecución de simulación

a) Activación del switch.- El switch se puede ejecutar con el comando "java" desde el prompt del sistema operativo y desde el directorio de instalación del JPOS, del siguiente modo:

```
>java –jar jpos.jar
```

Aparecerá un mensaje en pantalla similar al siguiente:

```
[java] <log realm="Q2.system" at="Sat May 23 00:15:05 COT 2009.625">
[java]   <info>
[java]     Q2 started, deployDir=D:\Titulacion\Fuentes\jposee\build\deploy
[java]   </info>
[java] </log>
```

b) Ejecución de transacción de Inicialización.- Desde el TPV Virtual se ejecuta la opción Inicialización, la que envía el mensaje descrito en la sección 3.6.1 al Switch, el que a su vez retorna un mensaje de respuesta al TPV.

Las pantallas del aplicativo TPV Virtual se muestran a continuación, con las informaciones enviadas y luego recibidas como respuesta.

En la Figura 3.6, se muestra parte del mensaje, en formato ISO, enviado por el TPV al switch. Se pueden ver algunos de los campos importantes enviados en el mensaje, como son el campo MTI (tipo de mensaje); el campo 11, un número que identifica a la transacción; y el campo 41, que identifica al terminal que realizó la transacción.

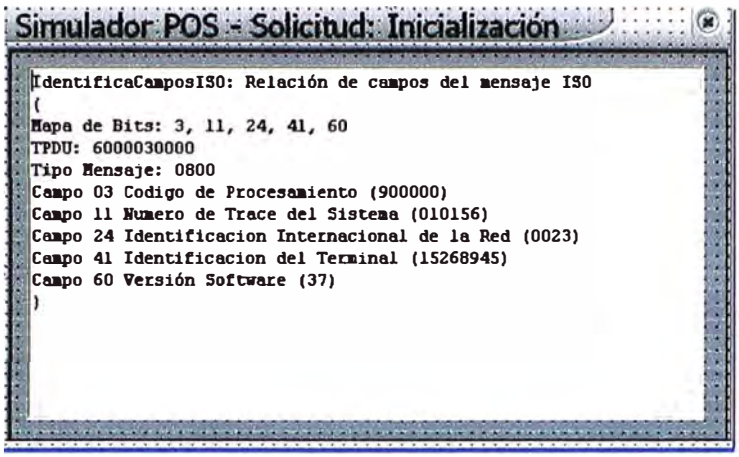

Figura 3.6.- Simulación Inicialización: Mensaje enviado por TPV

La Figura 3.7 muestra la pantalla principal del TPV virtual, con el mensaje de retorno en el campo 39.

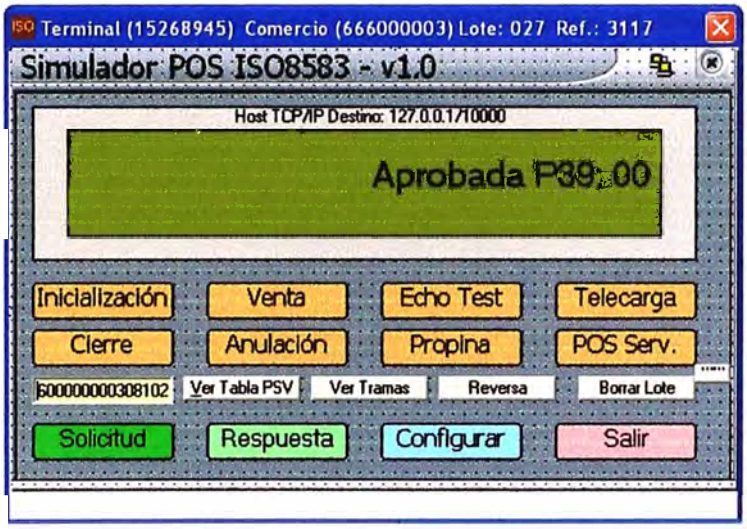

Figura 3.7.- Simulación Inicialización: Pantalla de TPV con campos de respuesta

La Figura 3.8 muestra la pantalla de Respuesta, del aplicativo TPV Virtual, con el mensaje de respuesta recibido del Switch. Se puede comprobar que el campo 39 (Código de respuesta) contiene el valor '00', que significa que es una transacción aprobada. Este campo contendrá otros valores diferentes a '00' si la transacción resultara denegada por el resolutor. Los diferentes valores indicarán el motivo por el que la transacción fue denegada o rechazada. A manera de ejemplo los diferentes motivos pueden ser validaciones en campos, clave de PIN incorrecta, crédito insuficiente, tarjeta caducada, etc.

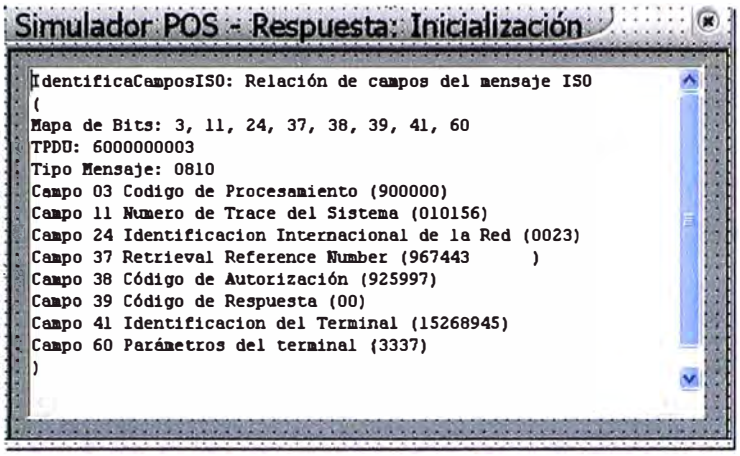

Figura 3.8.- Simulación Inicialización: Respuesta recibida del Switch

c) Ejecución de una transacción de venta.- Se activa la opción de venta en el menú del TPV Virtual. Aparece una ventana, que se muestra en la Figura 3.9, donde se ingresan los datos de una venta de 100 US$ y se activa una Venta Normal.

Figura 3.9.- Simulación Venta: Ingreso de datos en el TPV

En la Figura 3.10, se muestra parte del mensaje, en formato ISO, enviado por el TPV al switch. Se pueden ver algunos de los campos enviados en el mensaje.

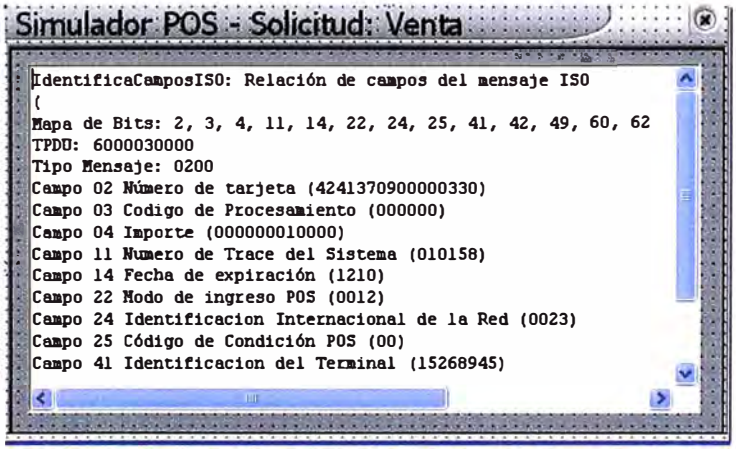

Figura 3.10.- Simulación Venta: Mensaje enviado por TPV

La Figura 3.11 muestra la pantalla principal del TPV virtual, con el mensaje de retorno en el campo 39. Para una venta además del campo 39 con el código de aprobación, se retornan los siguientes campos:

- campo 37 (retrieval reference number) , que es un número de referencia generalmente un secuencial para identificar el mensaje de respuesta del emisor,
- campo 38 (authorization identification response) que es un código de la autorización, que indica el motivo de la autorización.

Estos campos generalmente los retorna el emisor o autorizador, pero en esta simulación se han generado aleatoriamente.

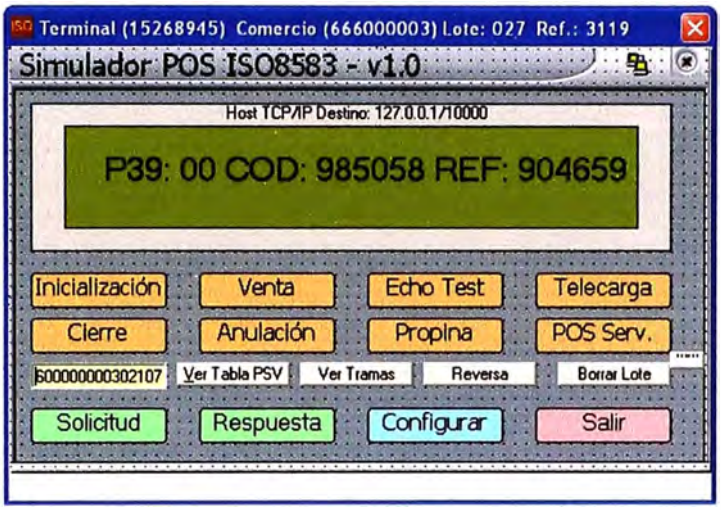

Figura 3.11.- Simulación Venta: Pantalla de TPV con campos de respuesta

La Figura 3.12 muestra la pantalla de Respuesta, con el mensaje de respuesta recibido del Switch. Se puede comprobar que el campo 39 (Código de respuesta) contiene el valor '00', que significa que es una transacción aprobada.

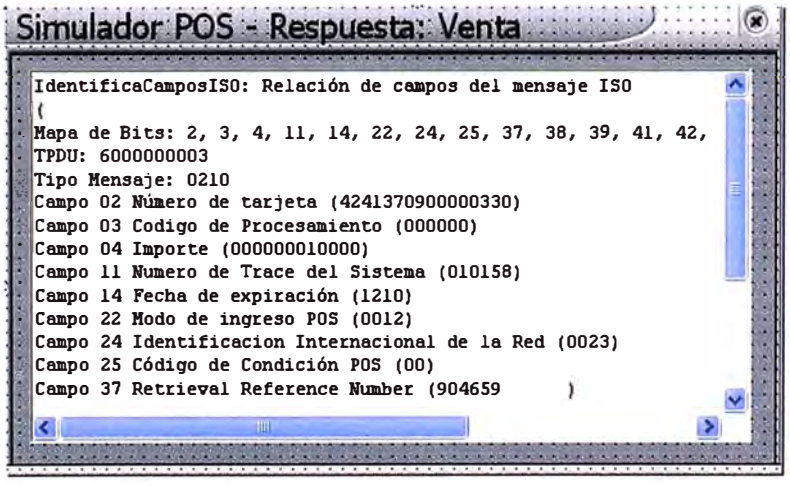

Figura 3.12.- Simulación Venta: Respuesta recibida del switch

En la Figura 3.13 se muestra el archivo de log de las transacciones, generados en el switch, y que contiene toda la información sobre los mensajes enviados y recibidos. Este archivo se encuentra en el directorio log/ del árbol de directorios de la aplicación.

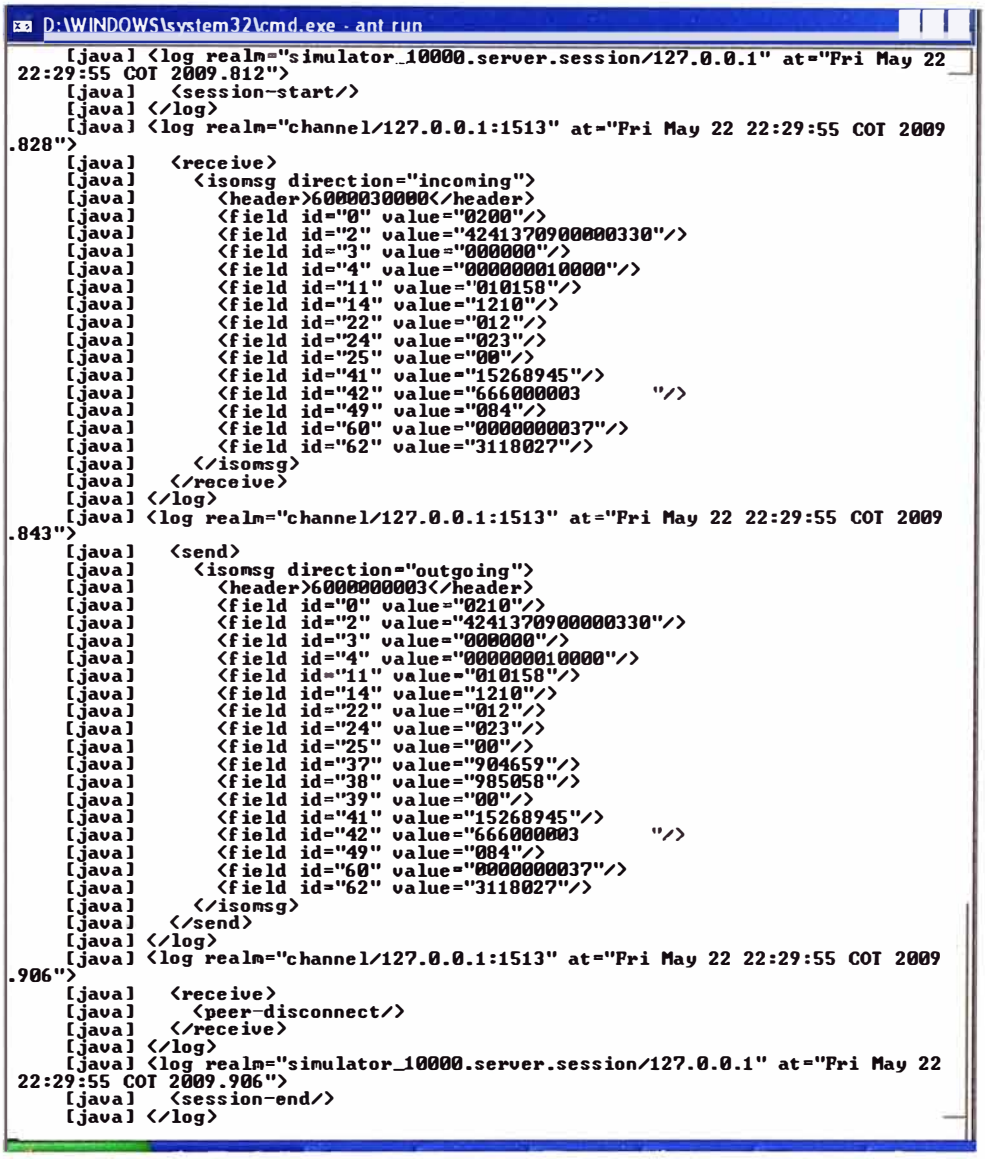

Figura 3.13.- Log de transacciones del switch

d) Ejecución de una transacción de venta, con condición para denegación.- Se activa la opción de venta en el menú del TPV Virtual. Se ingresan los datos de una venta de 99.99 US$. La lógica del servidor está programada para simular un rechazo cuando el importe de la transacción es 99.99. Se devuelve el valor 01 en el campo 39.

A continuación, en las siguientes figuras, se muestran las mismas pantallas mostradas en la sección anterior, para este caso:

Figura 3.14.- Simulación Venta Denegada: Ingreso de datos en el TPV

Como se muestra, en la Figura 3.14, se ingresa el importe de 99.99 para simular la denegación. Luego vemos en la Figura 3.15, parte del mensaje, en formato ISO, enviado por el TPV al switch. Se pueden ver algunos de los campos enviados en el mensaje.

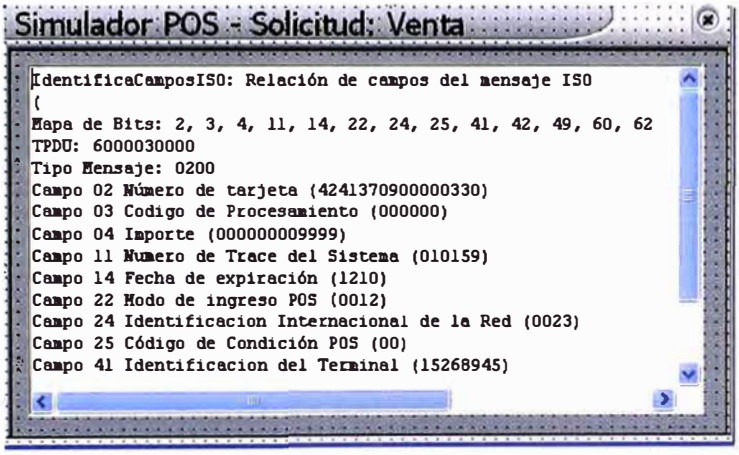

Figura 3.15.- Simulación de Venta Denegada: Mensaje enviado por TPV

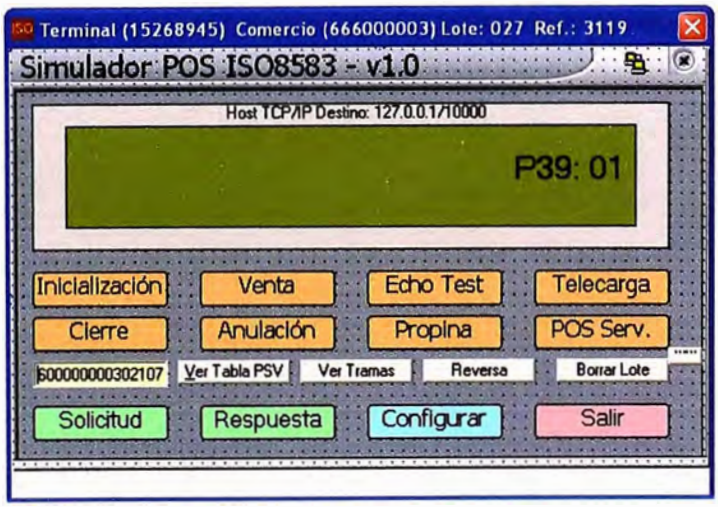

Figura 3.16.- Simulación de Venta Denegada: Pantalla de TPV con campos de respuesta

La Figura 3.16 se muestra la pantalla de Respuesta, con el mensaje de respuesta recibido del Switch. Se puede comprobar que el campo 39 (Código de respuesta) contiene el valor '01', que significa que es la transacción ha sido denegada. Asimismo se muestra en la Figura 3.17 la ventana del TPV virtual con el mensaje de respuesta.

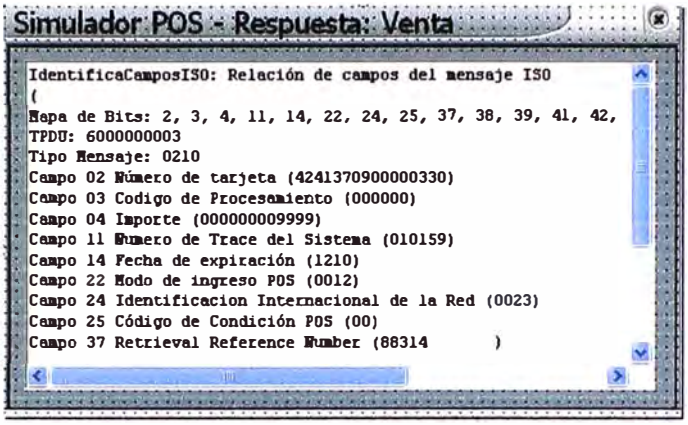

Figura 3.17.- Simulación de Venta Denegada: Respuesta recibida del switch

Por último, en Figura 3.18 se muestra el archivo de log de las transacciones.

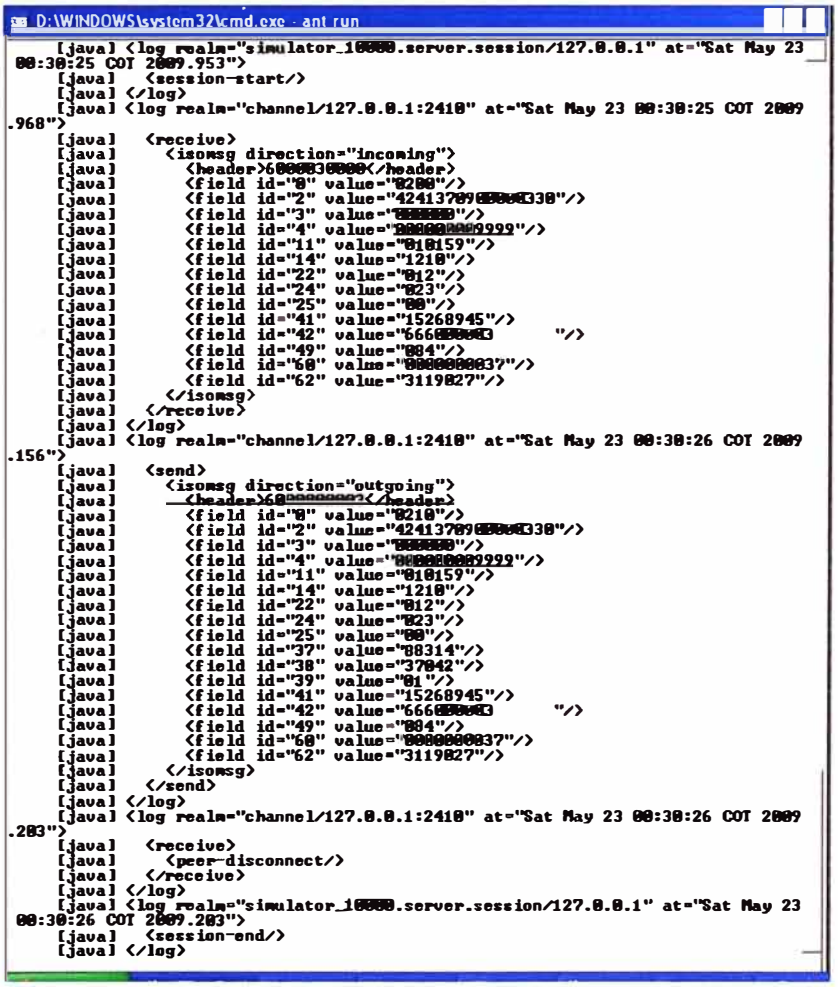

Figura 3.18.- Log de transacciones del switch

3.7 Medición de desempeño de la simulación

Se realizó una medición de desempeño usando un programa basado en el TPV Virtual, para pruebas masivas, configurado para el envío de ráfagas de transacciones hacia el servidor. El switch para esta simulación ejecuta las siguientes tareas:

- Recibe la transacción
- Desempaqueta el mensaje
- Construye el mensaje de respuesta
- Almacena la transacción en un log
- Envía la respuesta

En un entorno real, el switch puede ejecutar, además, las siguientes validaciones: Validar el comercio, validar el terminal, validar comercios contra listas blancas o negras (listas de comercios con riesgos).

Se configuraron 9 terminales, para ser activados por el simulador cada uno lanzando 2,000 transacciones en ráfaga. La pantalla de configuración se muestra en la Figura 3.19.

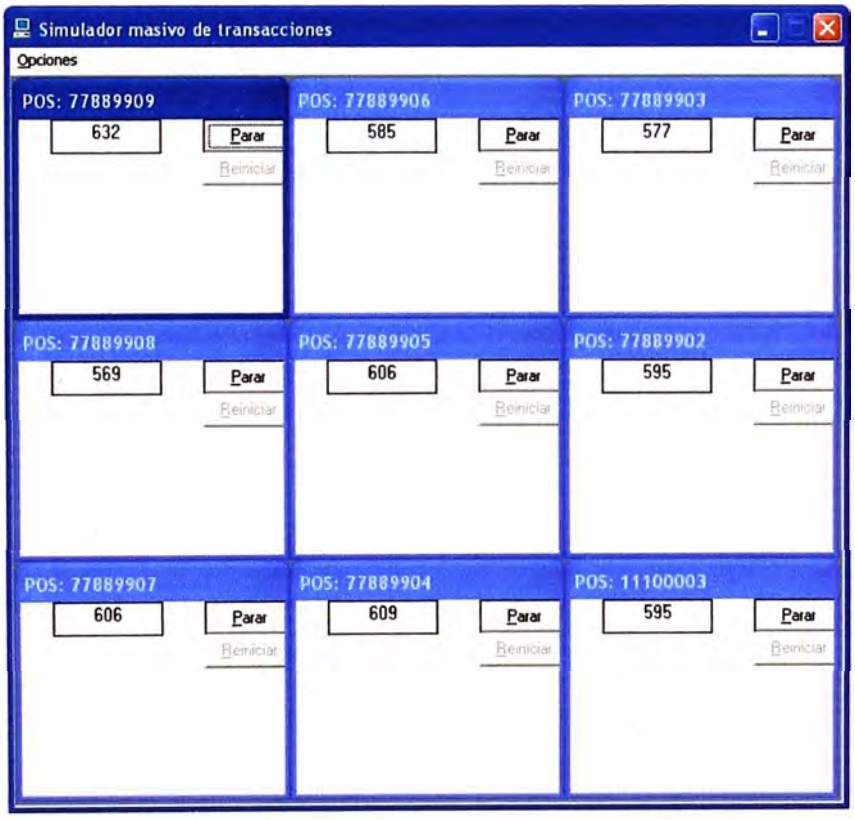

Figura 3.19.- Pantalla del TPV Virtual para simulaciones masivas

La configuración del servidor donde se instaló JPOS es la siguiente.

Modelo:	Laptop HP-Compaq 6710b
CPU:	Intel Core 2 Duo, T8300 a 2.4 GHz
RAM:	2 GB
Sistema Operativo:	Windows XP Service Pack 2
Disco Duro:	160 GB

La configuración del cliente donde se instaló el TPV Virtual

Modelo:	Laptop Lenovo 3000 C200
CPU:	Intel Core 2 Duo, T5600 a 1.8 GHz
RAM	1.5 GB
Sistema Operativo:	Windows XP Service Pack 3
Disco Duro:	120GB

Ambos conectados por una red 802.11g a 54 Mbps

La utilización de los procesadores, en el servidor switch, estuvo en el orden del 45% durante la prueba, como se ve en la siguiente Figura 3.20.

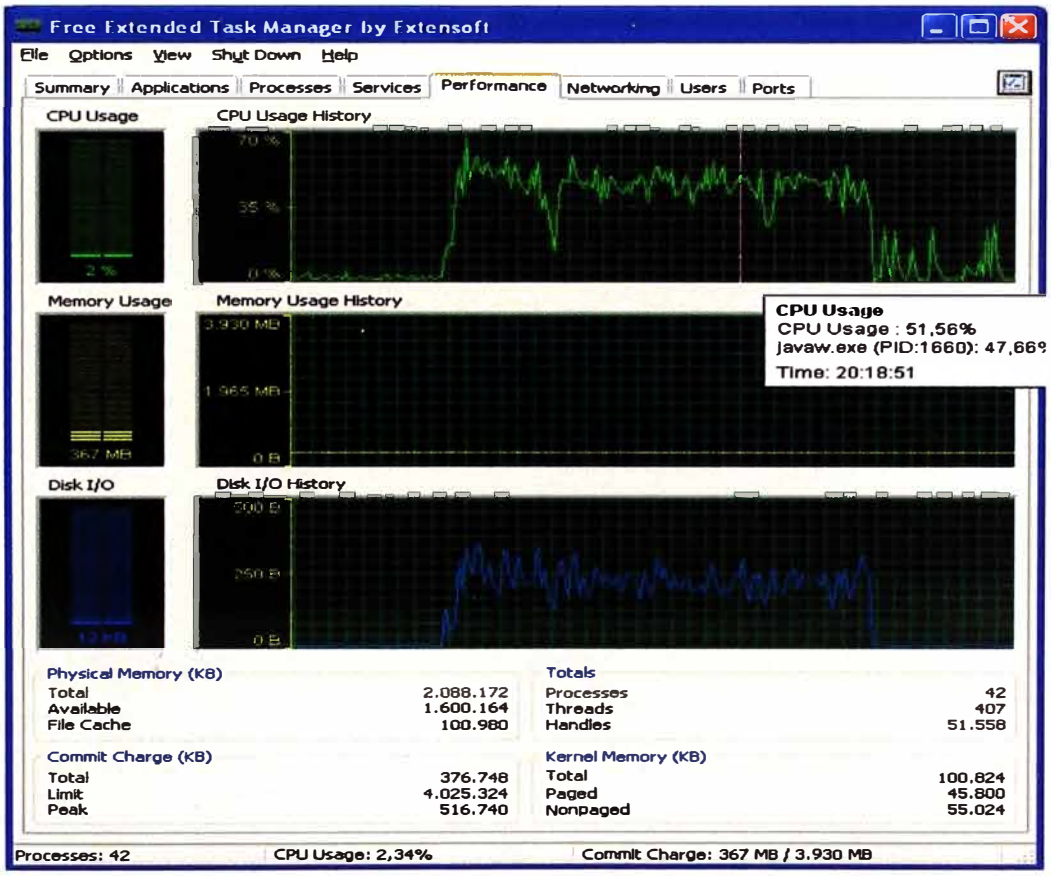

Figura 3.20.- Uso de recursos en Switch de prueba

La utilización de la red fue mínima, estuvo por el orden del 1% con picos no mayores al 5%, como se puede ver en la Figura 3.21.

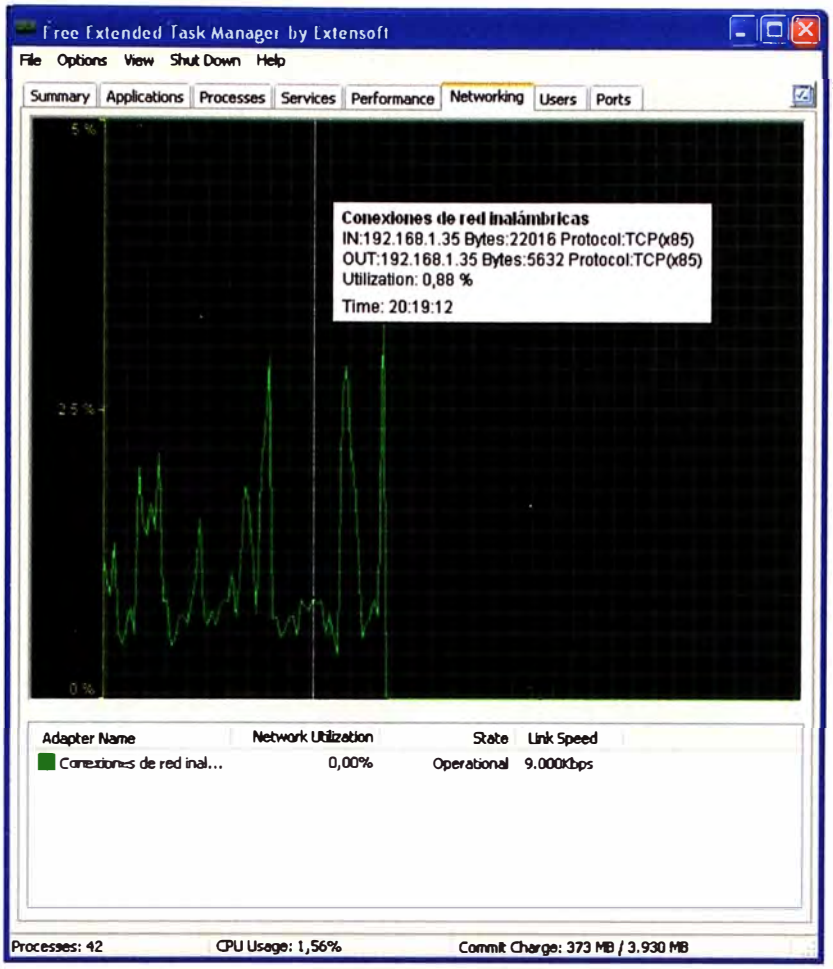

Figura 3.21.- Utilización de red en Switch de prueba

Se calculó el TPS (número de transacciones por segundo) con los resultados, mostrados en la Tabla N° 3.4.

Tabla N° 3.4.- Resultados de la prueba de desempeño

Parámetro	**Descripción**
Tiempo total de la prueba	120 segundos
Hora inicial	20:17:38
Hora final	20:19:37
Nro de transacciones realizadas	8,876
Promedio TPS	**74**
TPS máximo	122
TPS mínimo	41
Desviación estándar	16.25

En la Figura 3.22 se puede ver la evolución de los TPS durante la prueba.

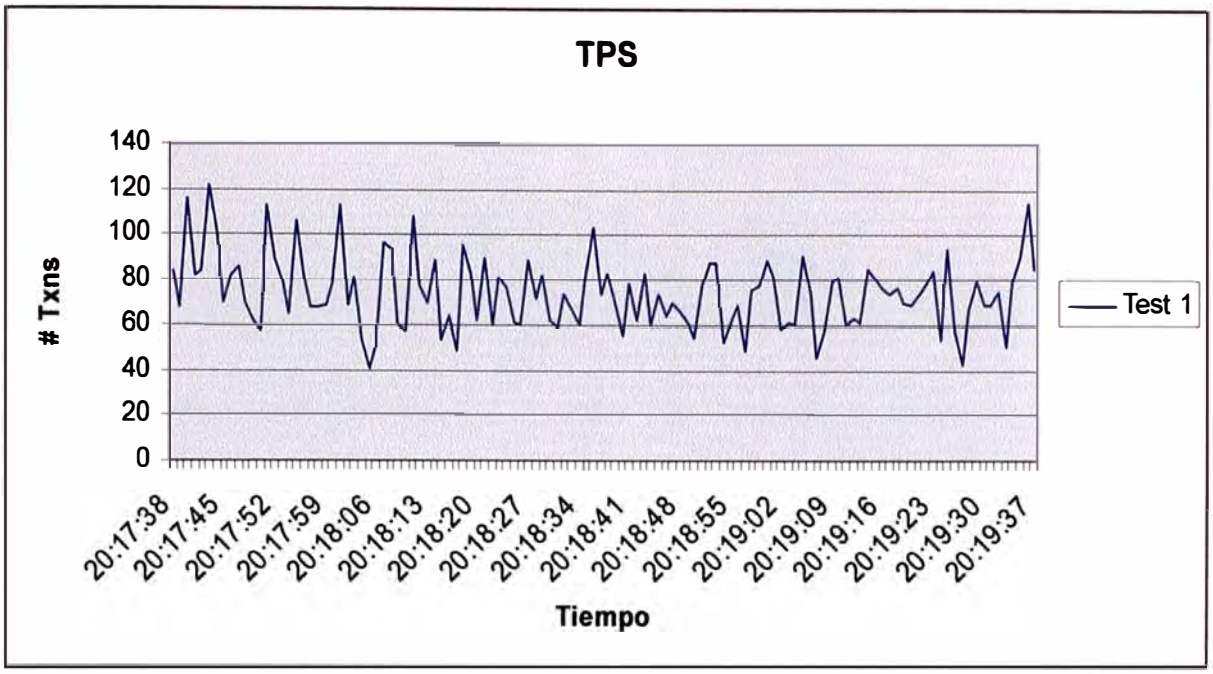

Figura 3.22.- Evolución de TPS durante simulación

Estos resultados son referenciales debido al entorno empleado en la prueba. Se ha buscado poder comparar estos resultados con los de la plataforma usada en producción, mostrada en la sección 2.4.1 Configuración de Hardware. Se ha usado, para este fin, la publicación de SPEC (Standard Performance Evaluation Corporation), que es un organismo que publica comparativas de rendimiento de procesadores, encontrando lo mostrado en la Tabla N° 3.5[8].

Tabla N° 3.5.- Rendimiento SPECint de procesadores equivalentes

Equipos	SPECint_base2006
Hewlett-Packard Company ProLiant DL160 G5p (3.20 GHz, Intel Xeon X5482) (similar a servidor en producción)	26.2
Hewlett-Packard Company HP dv2000 (Intel Core 2 Duo T7100) (similar a equipo usado en prueba)	11.8

Podemos ver en este resultado, que el CPU de producción tiene un índice SPEC que es 2.2 veces mayor que el SPEC del CPU de prueba. Si extrapolamos este valor a la plataforma de producción podemos ver que podría soportar 160 TPS en promedio. Esto es mucho mayor que el requerimiento de TPS que tiene en la actualidad.

[8] Se ha utilizado el indicador SPECint, al considerarse operaciones en números enteros. Ver referencia bibliográfica [8] y [9]

De esta forma se valida, de manera aproximada, que la plataforma presentada en la sección 2.4.1 Configuración de Hardware soporta los requerimientos de un entorno real de producción.

3.8 Comparación de costos frente a una solución propietaria

Se tomará como referencia una plataforma Stratus/VOS con software de switch transaccional PRICEnet, desarrollado por Sermepa, utilizado principalmente en España

Tabla N° 3.6.- Comparativa de costos entre plataformas

Plataforma Propietaria	**Costos de Instalación y Licencias**
Plataforma Stratus ftServer Serie V modelo V100 1 Intel Xeon 3.6 Ghz Arquitectura Fibra Canal 62 GB de disco	~ 250,000 US$
Software Pricenet	~ 500,000 US$
Total Plataforma Propietaria	**~ 750,000 US$**
Plataforma Abierta	**Costos de Instalación y Licencias**
CISCO CSS 11501	8,500 US$
2 Dell Poweredge 6850 *Quad* 3.66*ghz Xeon* + Raid / 5x 73gb	2,800 US$
2 HP Proliant ML380 XEON 3.2GHZ 144GB	2,000 US$
EMC Symmetrix 3930 Triple Bay Storage	20,000 US$
4 Licencias Windows Advanced Server 2003	2,600 US$
2 Licencias MS SQL Server	4,000 US$
Licencia comercial JPOS	6,750 US$
Total Plataforma Abierta	**46,650 US$**

Los datos mostrados en la Tabla N° 3.6 son referenciales, su objetivo es mostrar órdenes de magnitud. El costo de instalación de una plataforma abierta para soluciones de este tipo, puede ser 10 a 15 veces menor al de una plataforma propietaria.

A lo largo de este capítulo se ha descrito la arquitectura interna del software que implementa el núcleo o switch transaccional, se ha implementado un servidor con la funcionalidad básica lo que nos ha servido para realizar una simulación funcional y de rendimiento en un entorno de pruebas, validando de manera aproximada que esta arquitectura cumple los requerimientos de un entorno de producción.

CONCLUSIONES

1) Como conclusión de este trabajo se demuestra que el avance de la tecnología hace que una problemática que requería de soluciones especializadas y caras, pueda ser resuelta por tecnologías ya extendidas y comunes, en cuanto a la plataforma hardware, cuyas prestaciones han venido cumpliendo la ley de Moore[9]. Para lograr cumplir requerimientos de alta disponibilidad, con hardware que no tiene estas características, se utilizan equipos en arreglos de cluster.
2) En cuanto al software, se demuestra que el modelo de desarrollo de código abierto u Open Source, puede dar como resultados aplicativos complejos tanto horizontales (como ya se ha demostrado con sistemas operativos, Linux, o herramientas ofimáticas) como verticales, aplicables a una industria específica, como es el caso de JPOS, base de la solución presentada en este informe.
3) Se ha visto además, que una solución de este tipo puede cumplir con los requerimientos tecnológicos que exigen la industria y el mercado, con costos muchos menores a los de plataformas propietarias.
4) Todo esto representa una oportunidad, en países como el nuestro, de añadir valor a soluciones ya probadas en otros entornos, adaptándolas al nuestro o modificándolas para nuevos usos no pensados, con lo que se abre el camino al progreso.

[9] La **Ley de Moore** expresa que aproximadamente cada 18 meses se duplica el número de transistores en un circuito integrado. Se trata de una ley empírica, formulada por el co-fundador de Intel, Gordon E. Moore el 19 de abril de 1965, cuyo cumplimiento se ha podido constatar hasta hoy. (Wikipedia)

ANEXO A

Listado de campos ISO 8583 versión 1987

Se muestra la definición de los campos del estándar ISO 8583 versión 1987.

ISO-Defined Data Elements		
Data Element	**Type**	**Usage**
1	b 64	Bit Map Extended
2	n ..19	Primary account number (PAN)
3	n 6	Processing code
4	n 12	Amount, transaction
5	n 12	Amount, Settlement
6	n 12	Amount, cardholder billing
7	n 10	Transmission date & time
8	n 8	Amount, Cardholder billing fee
9	n 8	Conversion rate, Settlement
10	n 8	Conversion rate, cardholder billing
11	n 6	Systems trace audit number
12	n 6	Time, Local transaction
13	n 4	Date, Local transaction
14	n 4	Date, Expiration
15	n 4	Date, Settlement
16	n 4	Date, conversion
17	n 4	Date, capture
18	n 4	Merchant type
19	n 3	Acquiring institution country code
20	n 3	PAN Extended, country code
21	n 3	Forwarding institution. country code
22	n 3	Point of service entry mode
23	n 3	Application PAN number
24	n 3	Function code(ISO 8583:1993)/Network International identifier (?)
25	n 2	Point of service condition code
26	n 2	Point of service capture code
27	n 1	Authorizing identification response length
28	n 8	Amount, transaction fee
29	n 8	Amount. settlement fee
30	n 8	Amount, transaction processing fee
31	n 8	Amount, settlement processing fee
32	n ..11	Acquiring institution identification code
33	n ..11	Forwarding institution identification code
34	n ..28	Primary account number, extended
35	z ..37	Track 2 data
36	n ...104	Track 3 data
37	an 12	Retrieval reference number
38	an 6	Authorization identification response
39	an 2	Response code
40	an 3	Service restriction code
41	ans 8	Card acceptor terminal identification
42	ans 15	Card acceptor identification code
43	ans 40	Card acceptor name/location
44	an ..25	Additional response data

45	an ..76	Track 1 Data
46	an ...999	Additional data - ISO
47	an ...999	Additional data - National
48	an ...999	Additional data - Private
49	a 3	Currency code, transaction
50	an 3	Currency code, settlement
51	a 3	Currency code, cardholder billing
52	b 16	Personal Identification number data
53	n 18	Security related control information
54	an 120	Additional amounts
55	ans ...999	Reserved ISO
56	ans ...999	Reserved ISO
57	ans ...999	Reserved National
58	ans ...999	Reserved National
59	ans ...999	Reserved for national use
60	an .7	Advice/reason code (private reserved)
61	ans ...999	Reserved Private
62	ans ...999	Reserved Private
63	ans ...999	Reserved Private
64	b 16	Message authentication code (MAC)
65	b 16	Bit map, tertiary
66	n 1	Settlement code
67	n 2	Extended payment code
68	n 3	Receiving institution country code
69	n 3	Settlement institution county code
70	n 3	Network management Information code
71	n 4	Message number
72	ans ...999	Data record (ISO 8583:1993)/n 4 Message number, last(?)
73	n 6	Date, Action
74	n 10	Credits, number
75	n 10	Credits, reversal number
76	n 10	Debits, number
77	n 10	Debits, reversal number
78	n 10	Transfer number
79	n 10	Transfer, reversal number
80	n 10	Inquiries number
81	n 10	Authorizations, number
82	n 12	Credits, processing fee amount
83	n 12	Credits, transaction fee amount
84	n 12	Debits, processing fee amount
85	n 12	Debits, transaction fee amount
86	n 15	Credits, amount
87	n 15	Credits, reversal amount
88	n 15	Debits, amount
89	n 15	Debits, reversal amount
90	n 42	Original data elements
91	an 1	File update code
92	n 2	File security code
93	n 5	Response indicator
94	an 7	Service indicator
95	an 42	Replacement amounts

96	an 8	Message security code
97	n 16	Amount, net settlement
98	ans 25	Payee
99	n ..11	Settlement institution identification code
100	n ..11	Receiving institution identification code
101	ans 17	File name
102	ans ..28	Account identification 1
103	ans ..28	Account identification 2
104	ans ...100	Transaction description
105	ans ...999	Reserved for ISO use
106	ans ...999	Reserved for ISO use
107	ans ...999	Reserved for ISO use
108	ans ...999	Reserved for ISO use
109	ans ...999	Reserved for ISO use
110	ans ...999	Reserved for ISO use
111	ans ...999	Reserved for ISO use
112	ans ...999	Reserved for national use
113	n ..11	Authorizing agent institution id code
114	ans ...999	Reserved for national use
115	ans ...999	Reserved for national use
116	ans ...999	Reserved for national use
117	ans ...999	Reserved for national use
118	ans ...999	Reserved for national use
119	ans ...999	Reserved for national use
120	ans ...999	Reserved for private use
121	ans ...999	Reserved for private use
122	ans ...999	Reserved for private use
123	ans ...999	Reserved for private use
124	ans ...255	Info Text
125	ans ..50	Network management information
126	ans .6	Issuer trace id
127	ans ...999	Reserved for private use
128	b 16	Message Authentication code

Donde, por ejemplo:

Field Definition	Meaning
n6	Fixed length field of six digits
n.6	LLVAR numeric field of up to 6 digits in length
a..11	LLVAR alphanumeric field of up to 11 characters in length
b...999	LLLVAR binary field of up to 999 bytes in length

ANEXO B

Archivo iso87binary.xml utilizado en la simulación

Este es el archivo de definición de la trama ISO utilizada en la simulación de transacciones de la sección 3.6 Simulación

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
 <!DOCTYPE isopackager (View Source for full doctype...)>
- <!– ISO 8583:1987 (ASCII) field descriptions for GenericPackager –>
- <isopackager>
 <isofield id="0" length="4" name="MESSAGE TYPE INDICATOR" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="1" length="16" name="BIT MAP" class="org.jpos.iso.IFA_BITMAP" />
 <isofield id="2" length="19" name="PAN - PRIMARY ACCOUNT NUMBER" class="org.jpos.iso.IFA_LLNUM" />
 <isofield id="3" length="6" name="PROCESSING CODE" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="4" length="12" name="AMOUNT, TRANSACTION" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="5" length="12" name="AMOUNT, SETTLEMENT" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="6" length="12" name="AMOUNT, CARDHOLDER BILLING" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="7" length="10" name="TRANSMISSION DATE AND TIME" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="8" length="8" name="AMOUNT, CARDHOLDER BILLING FEE" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="9" length="8" name="CONVERSION RATE, SETTLEMENT" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="10" length="8" name="CONVERSION RATE, CARDHOLDER BILLING" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="11" length="6" name="SYSTEM TRACE AUDIT NUMBER" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="12" length="6" name="TIME, LOCAL TRANSACTION" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="13" length="4" name="DATE, LOCAL TRANSACTION" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="14" length="4" name="DATE, EXPIRATION" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="15" length="4" name="DATE, SETTLEMENT" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="16" length="4" name="DATE, CONVERSION" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="17" length="4" name="DATE, CAPTURE" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="18" length="4" name="MERCHANTS TYPE" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="19" length="3" name="ACQUIRING INSTITUTION COUNTRY CODE" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="20" length="3" name="PAN EXTENDED COUNTRY CODE" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="21" length="3" name="FORWARDING INSTITUTION COUNTRY CODE" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="22" length="3" name="POINT OF SERVICE ENTRY MODE" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="23" length="3" name="CARD SEQUENCE NUMBER" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="24" length="3" name="NETWORK INTERNATIONAL IDENTIFIEER" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="25" length="2" name="POINT OF SERVICE CONDITION CODE" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="26" length="2" name="POINT OF SERVICE PIN CAPTURE CODE" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="27" length="1" name="AUTHORIZATION IDENTIFICATION RESP LEN" class="org.jpos.iso.IFA_NUMERIC" />
 <isofield id="28" length="9" name="AMOUNT, TRANSACTION FEE" class="org.jpos.iso.IFA_AMOUNT" />
 <isofield id="29" length="9" name="AMOUNT, SETTLEMENT FEE" class="org.jpos.iso.IFA_AMOUNT" />
 <isofield id="30" length="9" name="AMOUNT, TRANSACTION PROCESSING FEE" class="org.jpos.iso.IFA_AMOUNT" />
 <isofield id="31" length="9" name="AMOUNT, SETTLEMENT PROCESSING FEE" class="org.jpos.iso.IFA_AMOUNT" />
 <isofield id="32" length="11" name="ACQUIRING INSTITUTION IDENT CODE" class="org.jpos.iso.IFA_LLNUM" />
 <isofield id="33" length="11" name="FORWARDING INSTITUTION IDENT CODE" class="org.jpos.iso.IFA_LLNUM" />
 <isofield id="34" length="28" name="PAN EXTENDED" class="org.jpos.iso.IFA_LLCHAR" />
 <isofield id="35" length="37" name="TRACK 2 DATA" class="org.jpos.iso.IFA_LLNUM" />
 <isofield id="36" length="104" name="TRACK 3 DATA" class="org.jpos.iso.IFA_LLLCHAR" />
 <isofield id="37" length="12" name="RETRIEVAL REFERENCE NUMBER" class="org.jpos.iso.IF_CHAR" />
 <isofield id="38" length="6" name="AUTHORIZATION IDENTIFICATION RESPONSE" class="org.jpos.iso.IF_CHAR" />
 <isofield id="39" length="2" name="RESPONSE CODE" class="org.jpos.iso.IF_CHAR" />
 <isofield id="40" length="3" name="SERVICE RESTRICTION CODE" class="org.jpos.iso.IF_CHAR" />
```

```
<isofield id="41" length="8" name="CARD ACCEPTOR TERMINAL IDENTIFICACION" class="org.jpos.iso.IF_CHAR" />
<isofield id="42" length="15" name="CARD ACCEPTOR IDENTIFICATION CODE" class="org.jpos.iso.IF_CHAR" />
<isofield id="43" length="40" name="CARD ACCEPTOR NAME/LOCATION" class="org.jpos.iso.IF_CHAR" />
<isofield id="44" length="25" name="ADITIONAL RESPONSE DATA" class="org.jpos.iso.IFA_LLCHAR" />
<isofield id="45" length="76" name="TRACK 1 DATA" class="org.jpos.iso.IFA_LLCHAR" />
<isofield id="46" length="999" name="ADITIONAL DATA - ISO" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="47" length="999" name="ADITIONAL DATA - NATIONAL" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="48" length="999" name="ADITIONAL DATA - PRIVATE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="49" length="3" name="CURRENCY CODE, TRANSACTION" class="org.jpos.iso.IF_CHAR" />
<isofield id="50" length="3" name="CURRENCY CODE, SETTLEMENT" class="org.jpos.iso.IF_CHAR" />
<isofield id="51" length="3" name="CURRENCY CODE, CARDHOLDER BILLING" class="org.jpos.iso.IF_CHAR" />
<isofield id="52" length="8" name="PIN DATA" class="org.jpos.iso.IFA_BINARY" />
<isofield id="53" length="16" name="SECURITY RELATED CONTROL INFORMATION" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="54" length="120" name="ADDITIONAL AMOUNTS" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="55" length="999" name="RESERVED ISO" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="56" length="999" name="RESERVED ISO" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="57" length="999" name="RESERVED NATIONAL" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="58" length="999" name="RESERVED NATIONAL" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="59" length="999" name="RESERVED NATIONAL" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="60" length="999" name="RESERVED PRIVATE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="61" length="999" name="RESERVED PRIVATE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="62" length="999" name="RESERVED PRIVATE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="63" length="999" name="RESERVED PRIVATE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="64" length="8" name="MESSAGE AUTHENTICATION CODE FIELD" class="org.jpos.iso.IFA_BINARY" />
<isofield id="65" length="1" name="BITMAP, EXTENDED" class="org.jpos.iso.IFA_BINARY" />
<isofield id="66" length="1" name="SETTLEMENT CODE" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="67" length="2" name="EXTENDED PAYMENT CODE" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="68" length="3" name="RECEIVING INSTITUTION COUNTRY CODE" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="69" length="3" name="SETTLEMENT INSTITUTION COUNTRY CODE" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="70" length="3" name="NETWORK MANAGEMENT INFORMATION CODE" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="71" length="4" name="MESSAGE NUMBER" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="72" length="4" name="MESSAGE NUMBER LAST" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="73" length="6" name="DATE ACTION" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="74" length="10" name="CREDITS NUMBER" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="75" length="10" name="CREDITS REVERSAL NUMBER" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="76" length="10" name="DEBITS NUMBER" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="77" length="10" name="DEBITS REVERSAL NUMBER" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="78" length="10" name="TRANSFER NUMBER" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="79" length="10" name="TRANSFER REVERSAL NUMBER" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="80" length="10" name="INQUIRIES NUMBER" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="81" length="10" name="AUTHORIZATION NUMBER" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="82" length="12" name="CREDITS, PROCESSING FEE AMOUNT" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="83" length="12" name="CREDITS, TRANSACTION FEE AMOUNT" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="84" length="12" name="DEBITS, PROCESSING FEE AMOUNT" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="85" length="12" name="DEBITS, TRANSACTION FEE AMOUNT" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="86" length="16" name="CREDITS, AMOUNT" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="87" length="16" name="CREDITS, REVERSAL AMOUNT" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="88" length="16" name="DEBITS, AMOUNT" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="89" length="16" name="DEBITS, REVERSAL AMOUNT" class="org.jpos.iso.IFA_NUMERIC" />
<isofield id="90" length="42" name="ORIGINAL DATA ELEMENTS" class="org.jpos.iso.IFA_NUMERIC" />
```

```
<isofield id="91" length="1" name="FILE UPDATE CODE" class="org.jpos.iso.IF_CHAR" />
<isofield id="92" length="2" name="FILE SECURITY CODE" class="org.jpos.iso.IF_CHAR" />
<isofield id="93" length="6" name="RESPONSE INDICATOR" class="org.jpos.iso.IF_CHAR" />
<isofield id="94" length="7" name="SERVICE INDICATOR" class="org.jpos.iso.IF_CHAR" />
<isofield id="95" length="42" name="REPLACEMENT AMOUNTS" class="org.jpos.iso.IF_CHAR" />
<isofield id="96" length="16" name="MESSAGE SECURITY CODE" class="org.jpos.iso.IFA_BINARY" />
<isofield id="97" length="17" name="AMOUNT, NET SETTLEMENT" class="org.jpos.iso.IFA_AMOUNT" />
<isofield id="98" length="25" name="PAYEE" class="org.jpos.iso.IF_CHAR" />
<isofield id="99" length="11" name="SETTLEMENT INSTITUTION IDENT CODE" class="org.jpos.iso.IFA_LLNUM" />
<isofield id="100" length="11" name="RECEIVING INSTITUTION IDENT CODE" class="org.jpos.iso.IFA_LLNUM" />
<isofield id="101" length="17" name="FILE NAME" class="org.jpos.iso.IFA_LLCHAR" />
<isofield id="102" length="28" name="ACCOUNT IDENTIFICATION 1" class="org.jpos.iso.IFA_LLCHAR" />
<isofield id="103" length="28" name="ACCOUNT IDENTIFICATION 2" class="org.jpos.iso.IFA_LLCHAR" />
<isofield id="104" length="100" name="TRANSACTION DESCRIPTION" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="105" length="999" name="RESERVED ISO USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="106" length="999" name="RESERVED ISO USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="107" length="999" name="RESERVED ISO USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="108" length="999" name="RESERVED ISO USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="109" length="999" name="RESERVED ISO USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="110" length="999" name="RESERVED ISO USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="111" length="999" name="RESERVED ISO USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="112" length="999" name="RESERVED NATIONAL USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="113" length="999" name="RESERVED NATIONAL USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="114" length="999" name="RESERVED NATIONAL USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="115" length="999" name="RESERVED NATIONAL USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="116" length="999" name="RESERVED NATIONAL USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="117" length="999" name="RESERVED NATIONAL USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="118" length="999" name="RESERVED NATIONAL USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="119" length="999" name="RESERVED NATIONAL USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="120" length="999" name="RESERVED PRIVATE USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="121" length="999" name="RESERVED PRIVATE USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="122" length="999" name="RESERVED PRIVATE USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="123" length="999" name="RESERVED PRIVATE USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="124" length="999" name="RESERVED PRIVATE USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="125" length="999" name="RESERVED PRIVATE USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="126" length="999" name="RESERVED PRIVATE USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="127" length="999" name="RESERVED PRIVATE USE" class="org.jpos.iso.IFA_LLLCHAR" />
<isofield id="128" length="8" name="MAC 2" class="org.jpos.iso.IFA_BINARY" />
</isopackager>
```

ANEXO C

Java Management Extensions

Java Management eXtensions, JMX es la tecnología que define una arquitectura de gestión, la API (Application Programming Interface), los patrones de diseño, y los servicios para la monitorización/administración de aplicaciones basadas en Java. Su versión 1.2 ha sido añadida al J2SE en su versión 5.0.

Arquitectura JMX

La arquitectura JMX es un modelo de tres capas. El nivel de instrumentación lo definen los requisitos para implementar recursos a manejar por JMX. Puede ser cualquier entidad, como aplicaciones, componentes o dispositivos. El nivel de agente es el encargado de controlar las entidades de la capa de instrumentación. El nivel de gestión o adaptación es el encargado de adaptar las entidades externas que interactúan a nivel de agente.

Niveles

- Nivel de Instrumentación
- Nivel de Agente
- Nivel de Gestión o adaptación

Nos permite implementar una gestión fácil e instantánea para los objetos Java. En la arquitectura, los recursos se gestionan mediante MBeans. Los MBeans son objetos java similares conceptualmente a los javaBeans y son los encargados de representar cada una de las entidades. Un modo sencillo de ver los MBeans es pensar que son aquellas aplicaciones que se encargan de monitorizar otras entidades.

Cualquier entidad que necesite ser gestionada, en un futuro, puede ser implementada por medio de MBeans. Estos nos brindan la instrumentación de los recursos gestionados de forma estándar y para ser incorporados en cualquier agente JMX. Pueden ser cargados o eliminados dinámicamente según sea necesario, lo que nos brinda una gran flexibilidad.

Tipos de MBeans

- Standard MBean

  La forma más sencilla de instrumentar un recurso. Es similar al modelo de componentes de JavaBeans y con una interface de gestión estática. Posee métodos para obtener y asignar valores a los atributos que se definen en el objeto implementado. Los métodos que expone el objeto son determinados en el momento de la compilación del mismo y no pueden ser modificados en tiempo de ejecución.

- Dynamic MBean

Posee métodos genéricos para obtener, asignar valores e invocar callback. Los métodos toman entre sus argumentos el nombre del atributo cuyo valor se desea obtener, el nombre del atributo y el valor que se le quiere asignar.

- Model MBean

Constituye un MBean dinámico, genérico y configurable que se utiliza para instrumentar recursos en tiempo de ejecución.

- Open MBean

Este tipo de MBean permite utilizar objetos gestionados que son descubiertos en tiempo de ejecución.

Modelo de Capa

- Nivel de Agente

Está representado por el Servidor de MBean donde los diferentes MBean se registran, gestiona un conjunto de agentes representado por los MBeans. El Servidor MBeans implementa la interfaz MBeanServer que interactúa con los MBean.

Los mensajes emitidos por los MBeans pueden ser enviados mediante al gestor o tratadas dentro del agente.

- Nivel de Gestión

Nos permite implementar adaptadores en el Servidor JMX hacia otros protocolos de gestión e implementar MBeans que interaccionen con otros protocolos de gestión exponiendo sus atributos como un recurso gestionable. Los adaptadores de protocolos y conectores permiten que las aplicaciones de gestión accedan al Servidor JMX y manipule los MBeans que lo forman. Crean un puente entre las tecnologías existentes y las futuras. JMX incluye entre sus API, una API para la gestión SNMP y una API para la gestión WBEM..

- Nivel de Implementación

Un recurso es cualquier entidad ya sea de software o de hardware que se pueda gestionar a través de la red, siendo accedido a través de su interfaz de gestión. La implementación de un recurso significa crear un objeto Java que represente la interfase de gestión y que sigue el modelo de componentes de JavaBeans. Llamamos a estos objetos MBean.

Servidor MBean

El Servidor constituye un entorno que permite a los gestores interactuar con los MBeans, entre los que se encuentra:

- Localización

- Filtrado por nombre o expresión
- Monitorización de atributos
- Descubrir otros agentes
- Establecer relaciones entre MBeans
- Carga dinámica
- Programación de tareas
- Definir jerarquías de agentes

Adaptadores

Las aplicaciones de gestión acceden a los MBeans remotamente a través del Servidor MBean. También el acceso puede realizarse a través de los conectores de adaptadores de protocolos.

Las diferentes aplicaciones de gestión utilizan los conectores para interactuar con los agentes. La biblioteca JDMK que nos brinda soporta a conectores para RMI, HTTP y HTTPS.

Mientras que los adaptadores de protocolos se utilizan cuando se quiere integrar el recurso representado por MBean en un entorno de gestión ya existente. JDMK ofrece un conjunto de herramientas que facilitan la creación de agentes SNMP y la integración de la gestión SNMP en la arquitectura JMX.

BIBLIOGRAFÍA

[1] Protocolo ISO 8583: http://es.wikipedia.org/wiki/ISO_8583

[2] Datos de configuración de hardware:
http://www.andyorrock.com/2008/12/how-do-you-measure-uptime-in-a-payment-switch.html

[3] Performance: TPS
http://www.andyorrock.com/2008/12/2008-holiday-run-up-analysis-part-ii.html

[4] Performance: Número de transacciones y comercios por día
http://www.andyorrock.com/paymentsystems/2007/10/can-jpos-handle.html

[5] Content Switching: http://newsroom.cisco.com/dlls/FAQ-CSS_11500.pdf

[6] Clusters de conmutación por error: http://msdn.microsoft.com/es-es/library/ms189134.aspx

[7] Alejandro Revilla, "JPOS Programmer's Guide" versión 1.6.1,
Organización JPOS - Uruguay, 2008
http://www.jpos.org/products/proguide

[8] Resultados SPEC-CInt2006 de equipo HP ProLiant DL160 G5p (3.20 GHz, Intel Xeon X5482)
http://www.spec.org/cpu2006/results/res2008q4/cpu2006-20081013-05592.html

[9] Resultados SPEC-CInt2006 de equipo HP dv2000 (Intel Core 2 Duo T7100)
http://www.spec.org/cpu2006/results/res2007q3/cpu2006-20070723-01523.html