Cost-Performance Optimization of SSL-Based Secure Distributed Infrastructures

S. Bregni, *Senior Member, IEEE*, P. Giacomazzi, *Member, IEEE*, A. Poli, *Student Member, IEEE*

***Abstract*— Business-to-Business and Business-to-Customer transactions in Internet require secure communication, especially for web applications. The Secure Socket Layer (SSL) protocol is one of the most viable solutions to provide the required level of confidentiality, message integrity and endpoint authentication. The two main alternatives for providing SSL security are the *end-to-end* and the *accelerated* solutions, which enable different cost-performance tradeoffs, where performance is intended as the overall delay that the customer experiences to complete the transaction. The accelerated solution is enabled by special devices (SSL acceleration cards) placed in network nodes. In this paper, we propose an optimization algorithm, which designs the ICT infrastructure minimizing the total cost, given a target performance objective defined as the end-to-end delay for the completion of the distributed application tasks. We apply this method to evaluate the efficiency of SSL acceleration versus end-to-end SSL, in order to determine in what conditions SSL acceleration is convenient. Our algorithm performs joint optimization of computing and communication resources, whilst in literature hardware and network are typically optimized separately.**

***Keywords* — Communication system security, information systems, information technology, optimization methods.**

I. Introducción

LA SEGURIDAD del tráfico de Internet puede ser proporcionada utilizando varios protocolos: el protocolo *Secure Socket Layer* (SSL) [1], el protocolo *Transport layer Security* (TLS) [2] y, en la capa de red, *Internet Protocol Security* (IPSec) [3]. Todos estos protocolos y mecanismos proporcionan comunicaciones seguras entre las aplicaciones que se ejecutan sobre dominios públicos tales como la Internet. En este artículo, nos enfocamos en SSL, cuyos servicios de seguridad son usados alrededor del mundo mediante aplicaciones distribuidas, accesibles a través de una interfaz web. SSL es un protocolo que trabaja sobre el TCP. La capa de aplicación usa características del SSL para obtener la confidencialidad, la integridad del mensaje y la autenticación del terminal *(endpoint)* [4].

Los investigadores han estudiado el rendimiento del SSL, que por lo general es medido con el *overhead* del procesamiento de seguridad en los servidores web, como por ejemplo se estudió en [5]. Otros estudios se centran en algoritmos criptográficos [6] y proponen optimizaciones para acelerar las operaciones criptográficas. En [7], SSL se caracteriza detalladamente y se proporciona el tiempo consumido por los procesadores para la elaboración del SSL, en todas las fases de negociación del SSL.

Ya que el SSL consume recursos tanto de clientes como de servidores, el costo y el rendimiento son temas críticos en el empleo del SSL, por tanto, para garantizar los niveles de rendimiento requeridos de extremo a extremo *(end-to-end)* en la capa de aplicación, recursos adicionales deben ser suministrados para llevar a cabo las tareas relacionadas al SSL. Esto, a su vez, exige gastos de inversión adicionales. Por otro parte, hay varias opciones sobre como implementar servicios de seguridad SSL: actualmente, un método racional, general y cuantitativo no existe.

En este artículo, se proporciona un algoritmo de planificación orientado al rendimiento y al costo, realizado a nivel del sistema, de las infraestructuras SSL-aseguradas. Nuestro algoritmo está basado en un modelo matemático detallado tanto de rendimiento *end-to-end* como de costos en la infraestructura de Tecnología de Información y Comunicaciones *(Information and Communication Technology; ICT),* para la provisión de comunicaciones SSL-aseguradas sobre dominios público como la Internet o dominios privados como la red virtual privada de una empresa.

Dos opciones principales están disponibles para el empleo del SSL. En la primera, la conexión segura SSL puede ser proporcionada en un modo *end-to-end*, desde el cliente y desde el servidor de aplicación remota. En la segunda, el SSL puede ser acelerado por un dispositivo específico, el acelerador SSL (mirar por ejemplo [8][9]), el cual finaliza las conexiones SSL en lugar del cliente.

Nuestro método para seleccionar la estrategia óptima se basa en un modelo de toda la infraestructura tecnológica, incluidos hardware y componentes de red [11], el cual es usado para realizar un diseño óptimo de infraestructuras, minimizando los costos necesarios para satisfacer los requerimientos de rendimiento tanto de cálculo como de comunicación ([10]-[13]).

En la literatura, el diseño de infraestructuras usualmente se lleva a cabo dividiendo el problema en dos optimizaciones distintas una de hardware y una de red. El primer problema de optimización es como distribuir la carga total calculada de un sistema distribuido sobre múltiples máquinas, para minimizar costos de hardware ([12][14][15]). El segundo problema es donde ubicar las máquinas que deben intercambiar información de tal manera que se minimicen los costos de red ([16] - [18]). Estos dos problemas han sido estudiados separadamente en la literatura. Sin embargo, las decisiones de diseño sobre ambas alternativas están fuertemente interrelacionadas.

En general, se considera un escenario multi-sitio y se afrontan las siguientes opciones de diseño:

a) asignación de torres de servidores a sitios;
b) asignación de aplicaciones de servidor en torres de servidores compartidas;
c) asignación de dispositivos de aceleración SSL (llamadas tarjetas de aceleración SSL, o simplemente, tarjetas) para conectar los *routers* en la red;
d) dimensionado de todos los dispositivos.

Los autores agradecen a la Ingeniera María Lorena Zapata, Bogotá, Colombia, por traducir cuidadosamente al español el texto original.

S. Bregni, Politecnico di Milano, Milán, Italia, bregni@elet.polimi.it
P. Giacomazzi, Politecnico di Milano, Milán, Italia, giacomaz@elet.polimi.it
A. Poli, Politecnico di Milano, Milán, Italia, poli@elet.polimi.it

II. Aceleración SSL

A. El protocolo Capa de Conexión Segura (SSL)

En este artículo, nos concentramos en las aplicaciones HTTP y HTTPS. HTTPS es el protocolo HTTP usado por encima del protocolo SSL. Las comunicaciones SSL entre el cliente y el servidor se efectúan mediante sesiones SSL.

El Protocolo *Handshake* es responsable de la creación de sesión: este negocia parámetros criptográficos que son compartidos por varias conexiones, evitando las renegociaciones frecuentes. Esto permite a clientes y servidores identificarse mutuamente, negociar algoritmos de cifrado y de MAC e intercambiar en modo seguro la clave de sesión por cuenta del Protocolo de Registro SSL.

El Protocolo de Registro SSL proporciona servicios de integridad y privacidad. Este fragmenta los mensajes de entrada en bloques, comprime cada bloque, aplica un código de Autenticación de Mensaje *(Message Authentication Code; MAC)* y luego cifra el bloque usando una clave privada intercambiada por el Protocolo *Handshake*. Este agrega un encabezado al mensaje y lo envía a través del TCP. Los mensajes recibidos son decodificados, verificados, descomprimidos y reconstruidos, para después ser enviados a la capa de aplicación.

En [19], el tiempo de ejecución del SSL *handshake* ha sido medido en un host Xeon de 1.4 GHz (que tiene un *benchmark* SPECint2000 [20] $B = 516$). Los valores presentados son 2 ms para un SSL *handshake* resumido y 175 ms para un SSL *handshake* completo. Se puede ver que el navegador Microsoft Internet Explorer realiza una renegociación de sesión SSL, causando un nuevo SSL *handshake* completo [21] cada 120 s. El uso de la CPU de un servidor web, procesando páginas web transmitidas a 1 kB sobre un canal de SSL seguro (HTTPS), ha sido medida en [7]; el tiempo que la CPU emplea, es mostrado dividiéndolo en fases de generación de página (7.6%), codificación de página (el 2.4%) e intercambio de clave/ SSL *handshake* (90%). Se ha observado que el tiempo de generación y tiempo de codificación de una página son proporcionales al tamaño de la página.

B. Tarjetas de Aceleración SSL y Conmutadores de Contenido

Las operaciones SSL son ejecutadas típicamente por un proceso de servidor web. Mientras que un servidor HTTP escucha las nuevas conexiones en la puerta 80, el servidor SSL las escucha en la puerta 443. El proceso SSL usa la potencia de procesamiento de los servidores web, substrayendo recursos a otros procesos que se están ejecutando en los servidores y por lo tanto los tiempos de respuesta se incrementan.

El uso de tarjetas adaptadoras SSL permite mitigar las pesadas cargas de procesamiento del SSL. Las tarjetas adaptadoras SSL se colocan dentro de los servidores y se conectan en una ranura de la tarjeta madre. El servidor web continúa ejecutando un proceso SSL, pero un conjunto de funciones es delegado a la tarjeta adaptadora, de este modo la carga del servidor disminuye. Estas tarjetas soportan hasta 1000 conexiones nuevas por segundo [22].

Para disminuir la carga del servidor web, el SSL puede ser ejecutado por conmutadores de contenido, que son usados para equilibrar el tráfico inteligentemente entre servidores en centros de datos, basándose en la disponibilidad de contenido y la carga del servidor.

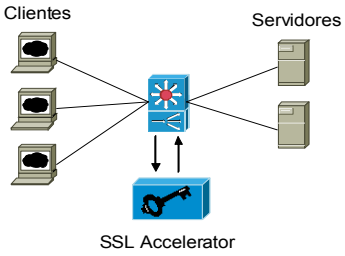

Figura 1. Conmutador de contenido con acelerador SSL integrado.

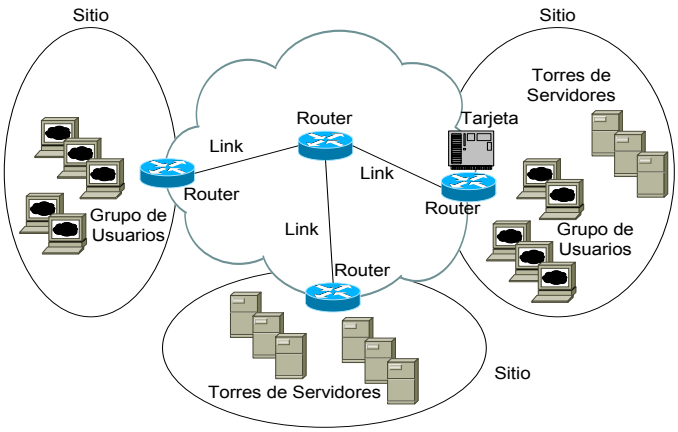

Figura 2. Modelo general de la infraestructura IT.

La configuración en la Fig. 1 centraliza las funciones SSL en un solo dispositivo. La necesidad de procesar muchas conexiones requiere aceleradores SSL en el conmutador de contenido [8]. El conmutador de contenido usa aceleradores SSL para descifrar el tráfico SSL de clientes a servidores y cifrarlo en la dirección contraria. Un conmutador de contenido también debería ser capaz de equilibrar la carga entre todos los servidores en la torre de servidores *(server farm),* estableciendo sesiones SSL.

III. El Problema de Optimización

Se considerará el modelo general de la infraestructura IT mostrado en la Fig. 2. Los usuarios y torres de servidores son localizados en sitios que son posiciones geográficamente separadas, las cuales alojan usuarios, torres de servidores, o ambos. Dentro de un sitio, los dispositivos son conectados a través de una red de área local de alta velocidad (LAN), la cual tiene mayor rendimiento y menor costo que el de una red geográfica que conecta sitios. Así, las LANs no son consideradas en la optimización de rendimiento y costo. Los sitios son conectados por una red geográfica elaborada con *link*s y *routers*. Los *routers* pueden estar o no equipados con una tarjeta para realizar la aceleración SSL.

A. Requisitos Tecnológicos

Se especifica la formalización de los requisitos para una organización de referencia con un conjunto de sitios S.

1) *Sitios*. Un sitio $s_i \in S$ está definido como un conjunto de recursos tecnológicos conectados mediante una Red de Área Local (LAN). Para cada pareja de sitios (s_i, s_j), la distancia es $dist(s_i, s_j)$.
2) *Aplicación SSL*. Una aplicación SSL $a_i^{SSL} \in A^{SSL}$ es una

aplicación que suministra servicios de criptografía SSL, la cual está caracterizada por un tiempo de CPU $time_{CPU}(a_i^{SSL})$ requerido para cifrar un mensaje en un servidor de referencia de tipo $ref(a_i^{SSL}) \in ST$, tiempo de disco $time_{DISK}(a_i^{SSL})$, espacio de RAM $space_{RAM}(a_i^{SSL})$, y espacio de disco $space_{DISK}(a_i^{SSL})$.

3) *Aplicación del servidor*. Una aplicación del servidor (o proceso de aplicación) $a_i \in A$ está caracterizada por un tiempo de CPU $time_{CPU}(a_i)$, un tiempo de disco $time_{DISK}(a_i)$, un espacio de RAM $space_{RAM}(a_i)$, un espacio de disco $space_{DISK}(a_i)$, un numero de bits de una solicitud $req(a_i)$,un numero de bits de una respuesta $resp(a_i)$, una aplicación SSL $ssl(a_i) \in A^{SSL}$ sirviendo a la aplicación a_i.
4) *Grupo de usuarios*. Un grupo de usuarios $u_i \in U$ es un conjunto de $|u_i|$ usuarios con requerimientos de cálculo comunes, es decir, usan el mismo conjunto de aplicaciones. Cada grupo de usuarios está caracterizado por un conjunto de aplicaciones de servidor utilizadas, como especifica la matriz $\{\gamma_{ij}\}$ donde γ_{ij} =1 si la aplicación $a_i \in A$ es usada por u_i si no $\gamma_{ij} = 0$; el sitio site$(u_i) \in S$ donde el grupo está localizado, el tiempo de respuesta requerido $delay_{ij}$ y la frecuencia de las solicitudes $msg(u_i, a_j)$ del grupo de usuarios u_i para la aplicación a_j.

B. Recursos de Hardware

Los requisitos de cálculo de la organización pueden ser cumplidos con los recursos de hardware definidos en la siguiente forma.

1) Tipo de link. Un tipo de *link* $lt_k \in LT$ está caracterizado por un costo de servicio $cost(lt_k)$, la distancias mínima y máxima $distMin(lt_k)$ and $distMin(lt_k)$, y la capacidad $cap(lt_k)$.
2) Tipo de router. Un tipo de *router* $rt_k \in RT$ está caracterizado por un costo de adquisición $cost(rt_k)$, una velocidad de servicio $BPS(rt_k)$, una máxima velocidad de servicio del backplane $B(rt_k)$, un clasificador de coeficiente $KC(rt_k)$, y un coeficiente de enrutamiento $KR(rt_k)$.
3) Tipo de servidor. Cada tipo de servidor $st_k \in ST$ está caracterizado por un costo de adquisición $cost(st_k)$, un tamaño de RAM $ram(st_k)$, un espacio de disco $disk(st_k)$, un *benchmark* de rendimiento de la CPU $bench_{CPU}(st_k)$ (es decir, el ratio entre el tiempo de ejecución de una aplicación de referencia en una CPU de referencia y el tiempo de ejecución de la misma aplicación en la CPU del st_k) y un *benchmark* de rendimiento de disco $bench_{DISK}(st_k)$.
4) Tipo de tarjeta de aceleración SSL. Un tipo de tarjeta $ct_k \in CT$ está caracterizado por un costo de adquisición $cost(ct_k)$ y un *benchmark* de rendimiento de la CPU $bench_{CPU}(ct_k)$.
5) Torre de servidores. Una torre de servidores $sf_i \in SF$ es un conjunto de $|sf_i|$ servidores del mismo tipo $type(sf_i) \in ST$. El número máximo de servidores permitidos en las torres de servidores está representado con φ.
6) Router. Un *router* $r_i \in R$ es un ejemplo de recurso de tipo *router type*$(r_i) \in RT$ asignado a un sitio s_i.
7) Link. Un *link* $l_{ij} \in L$ es un ejemplo de recurso de tipo *link type*$(l_{ij}) \in LT$ asignado a una pareja de sitios (s_i, s_j).
8) Tarjeta de aceleración SSL. Una tarjeta $c_i \in C$ es un ejemplo de tipo tarjeta $type(c_i) \in CT$ asignada a un *router* c_i. Nótese que un *router* puede o no ser asignado a una tarjeta.

IV. Modelo De Optimización

A. Variables de decisión

Las alternativas de optimización están representadas mediante las siguientes variables de decisión.

1) Asignación de aplicaciones a torres de servidores: $x_{ij} = 1$ si la aplicación $a_i \in A$ es asignada a una torre de servidores sf_j sino $x_{ij} = 0$.
2) Asignación de torres de servidores a sitios: $y_{jk} = 1$ si una torre de servidores sf_j es asignada a un sitio s_k, sino $y_{jk} = 0$.
3) Asignación de tarjetas: $w_{lk} = 1$ si la tarjeta c_l está instalada en un *router* r_k, sino $w_{lk} = 0$.
4) Asignación de aplicaciones SSL a Torres: $z_{pj} = 1$ si la aplicación SSL $a_p \in A^{SSL}$ es asignada a una torre de servidores sf_j, sino $z_{pj} = 0$.

B. Tiempo de respuesta

Para cada pareja (u_i, a_j) en la cual $\gamma_{ij} = 1$, se deben identificar los caminos de enrutamiento directos e inversos desde el sitio del grupo de usuarios u_i, site$(u_i) \in S$, hacia los sitios de la torre de servidores donde la aplicación a_j está asignada. Estos caminos pueden ser identificados mediante algún algoritmo de camino mínimo. Nuestra herramienta implementa el algoritmo de camino mínimo de Dijkstra usando una métrica de 1 para cada *link*.

El tiempo de respuesta depende de la carga promedio de los dispositivos que son atravesados por las solicitudes enviadas mediante grupos de usuarios u_i hacia una aplicación a_j a lo largo de los caminos directos e inversos. Los dispositivos a lo largo de los caminos directos e inversos son representados mediante listas ordenadas

$$d_{ij}^{req} \left| \forall i \in \left[0, \left| d_{ij}^{req} \right| \right], d_{ij}^{req}[i] \in \left(L \cup R \cup C \cup SF \right) \right.$$
$$d_{ij}^{res} \left| \forall i \in \left[0, \left| d_{ij}^{res} \right| \right], d_{ij}^{res}[i] \in \left(L \cup R \cup C \right) \right. \quad (1).$$

donde *req* y *res* indican el camino directo e inverso, respectivamente. Cada lista incluye todos los dispositivos atravesados a lo largo del correspondiente camino (*links*, *routers*, tarjetas, y torres de servidores).

Sea $rtime_{ij}$ el tiempo de respuesta experimentado por la solicitud de un grupo de usuarios u_i hacia la aplicación a_j, incluyendo el retardo a lo largo de los caminos tanto directos como inversos. Se considera que el tiempo de respuesta a lo largo del camino directo incluye el tiempo de servicio real de la solicitud. El tiempo de respuesta se calcula como

$$rtime_{ij} = \sum_{n=1}^{\left| d_{ij}^{req} \right|} rtime_{ij}\left(d_{ij}^{req}[n] \right) + \sum_{n=1}^{\left| d_{ij}^{res} \right|} rtime_{ij}\left(d_{ij}^{res}[n] \right) \quad (2).$$

Las solicitudes de aplicación afectan a todos los dispositivos que se encuentran a lo largo de los caminos directos e inversos con la misma frecuencia $\lambda_{ij} = |u_i| \cdot msg(u_i, a_j)$.

La aplicación de aceleración SSL se realiza con la tarjeta en el *router* del sitio destino de la solicitud. La aceleración SSL para una solicitud determinada puede ser ejecutada solamente en el destino, de este modo la solicitud sigue sin ser cifrada hacia el sitio de destino. Si el *router* localizado en el sitio de la torre de servidores de destino no tiene una tarjeta, entonces la torre de servidores debe ejecutar la aplicación SSL.

El tiempo de respuesta de los dispositivos ya mencionados

(*link*s, *routers*, tarjetas, y torres de servidores) se calcula de acuerdo a los parámetros del dispositivo y a la cantidad de solicitudes que atraviesan un solo dispositivo. El tiempo de respuesta para cada solicitud se calcula como la suma del tiempo promedio dedicado por cada solicitud (o respuesta) en un conjunto de múltiples colas de tipo M/M/1 y los tiempos de servicio correspondientes (el tiempo de propagación es también considerado en los dispositivos di tipo *link*).

C. Función Objetivo

La función objetivo a minimizar es el costo total, *TC*, de los recursos tecnológicos seleccionados para satisfacer los requisitos en un plazo de tiempo indicado mediante la variable *years*.

$$TC = years \cdot \sum_{l_{ij} \in L} \left[\text{cost}\left(type\left(l_{ij} \right) \right) \cdot dist\left(s_i, s_j \right) \right] + \\ + \sum_{sf_i \in S \,\middle|\, \sum_{a_j \in A} x_{ji} \neq 0} \left[\text{cost}\left(type\left(sf_i \right) \right) \cdot \left| sf_i \right| \right] + \\ + \sum_{r_i \in R} \text{cost}\left(type\left(r_i \right) \right) + \sum_{c_i \in C \,\middle|\, \sum_{r_j \in R} w_{ij} = 1} \text{cost}\left(type\left(c_i \right) \right) \qquad (3).$$

D. Restricciones

Trece restricciones se han identificado en nuestro modelo:

1) cada aplicación debe ser asignada a una torre de servidores;
2) cada torre de servidores debe ser asignada a un sitio;
3) Las torres de servidores son solo permitidas en sitios para alojar torres de servidores;
4) a cada *router* se le debe asignar al máximo una tarjeta;
5) Todos los grupos de usuarios deben ser servidos con un tiempo de respuesta inferior a los requisitos de retardo de sus aplicaciones;
6) solo se consideran los recursos de tipo *link* que cumplan con las restricciones de distancia;
7, 8) solo se consideran los recursos de tipo servidor que cumplan con las restricciones de espacio de RAM y de disco;
9) existe un número máximo de servidores permitidos en las torres de servidores;
10, 11, 12, 13) existe una carga máxima (100%) en las torres de servidores, tarjetas, *routers* y *link*s.

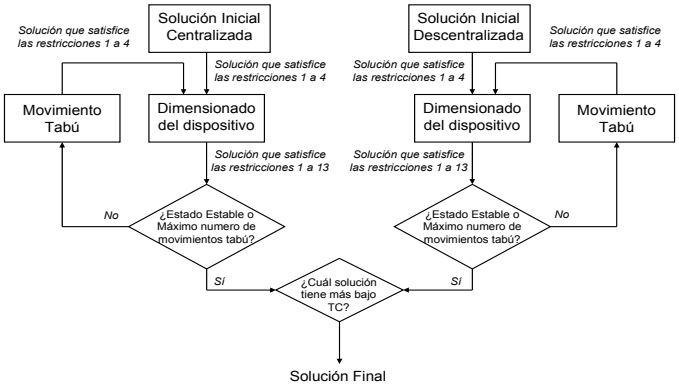

Figura 3. Diagrama de flujo del algoritmo de minimización de costo.

V. Algoritmo De Minimización De Costo

El algoritmo de minimización de costo, esquematizado en la Fig. 3, tiene como objetivo identificar la solución de costo mínimo que satisfaga los requisitos tecnológicos con los recursos tecnológicos correspondientes. El algoritmo se basa en una aproximación de búsqueda Tabú *(tabu-search; TS)* [23].

Dos soluciones iniciales son identificadas: Una solución totalmente centralizada asignando todas las aplicaciones a una torre de servidores y una solución totalmente descentralizada asignando cada aplicación a una torre de servidores distinta. Ambas soluciones cumplen con las restricciones 1, 2, 3, 4.

Después, la fase del dimensionado del dispositivo *(device-sizing phase)* identifica un conjunto de recursos tecnológicos que satisface todos los requisitos 1-13, incluyendo los retardos de aplicación, y calcula el correspondiente costo total *TC*. Los movimientos Tabú se realizan con el fin de reducir el TC hasta que las soluciones de costo alcancen un estado de equilibrio para un numero predefinido de movimientos o hasta que un número máximo de movimientos sea alcanzado. La fase de dimensionado del dispositivo es repetida después de cada movimiento tabú. La solución final seleccionada será la de menor costo entre aquellas obtenidas en las soluciones iniciales centralizadas y descentralizadas.

En nuestra implementación de la búsqueda tabú, El vecindario de una solución se explora mediante la ejecución de cuatro tipos de movimientos: *Desplazamiento de aplicación*, *desplazamiento de la torre de servidores*, *inserción de la tarjeta*, y *extracción de la tarjeta*. *Un desplazamiento de aplicación* extrae una aplicación de servidor de una torre de servidores y la asigna a una torre de servidores distinta. Un *desplazamiento de la torre de servidores* extrae una torre de servidores de un sitio y la localiza en un sitio distinto. Una *inserción de tarjeta* agrega una tarjeta a un *router*, mientras que una *extracción de la tarjeta* extrae una tarjeta de un *router*. Los movimientos Tabú deben satisfacer las restricciones 1, 2, 3, y 4. La ejecución de un movimiento cambia la configuración de las variables de decisión *x*, *y*, *w*. Esta configuración es una entrada a la fase del dimensionado del dispositivo del algoritmo de minimización de costos que se describe en la Fig. 3.

La fase del dimensionado del dispositivo pretende identificar el conjunto de recursos tecnológicos de mínimo costo que cumplan los requisitos dentro de la configuración actual de las variables de decisión *x*, *y*, *w*. El dimensionado se realiza en dos pasos: Un primer dimensionado seguido por una secuencia de mejoras y desmejoras.

El primer dimensionado se asigna a cada dispositivo el tipo de costo mínimo que cumpla con los requisitos de acuerdo a las reglas generales comúnmente usadas. De acuerdo a estas reglas generales, los tipos de dispositivos son escogidos de tal forma que la carga máxima de todas las colas sea inferior al 60% [24].

Dada una configuración de las variables de decisión *x*, *y*, *w* y la asignación del *type()* para cada dispositivo (como fue calculado en el primer paso del dimensionado), entonces, el $rtime_{ij}$ para cada pareja (u_i, a_j) puede ser determinado. Un puntaje (*score*) de configuración se calcula para medir la distancia entre el tiempo de respuesta real y máximo:

$$score = \sum_{i,j \mid \gamma_{ij}=1} \max\left\{ \left(rtime_{ij} - delay_{ij} \right), 0 \right\} \qquad (4).$$

Si *score* = 0, la configuración actual cumple con los restricciones de retardo, sino es necesaria una mejora del dispositivo. Si *score* > 0, las parejas con un tiempo de respuesta superior al máximo retardo son dirigidas, considerando la mejora de los respectivos dispositivos a lo largo de los caminos de la solicitud. Una mejora reemplaza el tipo de dispositivo actual con el tipo de más bajo costo que tiene un costo mayor que el tipo actual. Nótese que un recurso más costoso y de tipo torre de servidores puede ser obtenido, ya sea usando un tipo servidor distinto o cambiando el número total de servidores en la torre.

El algoritmo realiza una serie de mejoras, cada una de ellas reduce el puntaje de configuración, hasta que el *score* = 0 es alcanzado. Las mejoras que ofrecen más grande reducción del puntaje se llevan a cabo en primer lugar. Todas las configuraciones intermedias son factibles ya que cumplen con las restricciones 1─4.

Cuando una solución con un *score* = 0 es encontrada, el espacio de configuración es explorado por medio de una serie de mejoras y desmejoras para identificar el conjunto de dispositivos de costo mínimo. Las desmejoras del dispositivo traen soluciones a un costo total más bajo. Una serie de desmejoras y una serie de mejoras son realizadas en forma iterativa hasta que un número máximo predeterminado de cambios de configuración sea alcanzado. La mejor solución encontrada con todas las mejoras y desmejoras y con *score* = 0 es seleccionada como la salida de la fase del dimensionado del dispositivo.

VI. Verificación Empírica

Las pruebas empíricas se han llevado a cabo utilizando un prototipo de herramienta que se desarrolló implementando el algoritmo de costo mínimo. Esta herramienta incluye una base de datos de recursos tecnológicos comerciales y los datos de costos relacionados.

Tipos de servidores han sido analizados por los sitios web de cuatro vendedores: Dell, HP, IBM, and ION Computers. Los datos de rendimiento se han recogido de 460 servidores. La capacidad de cálculo ha sido medida mediante SpecInt2000 [20]. Se ha supuesto que una torre de servidores está compuesta por un máximo de $\varphi = 30$ servidores. De este modo, en nuestra herramienta se pueden elegir entre 13800 torres de servidores distintas.

La base de datos de los *routers* se basa en productos Cisco. Se consideran dos tipos de *router*, que tienen datos de costo y rendimiento suministrados por Cisco.

Un tipo de tarjeta acelerador SSL está disponible, con parámetros establecidos para que sea capaz de realizar hasta 1000 *handshake*s/s, es decir, un valor común para los dispositivos disponibles en el mercado [22]. El costo de la tarjeta se ha considerado inicialmente con un valor de 50.000 dólares.

Los datos de costo de los *link*s se han tomado de los documentos de precios oficiales de las líneas arrendadas digitales en Italia [25].

A continuación, se muestran evidencias empíricas de los ahorros de costo proporcionados por las tarjetas que soportan el servicio de aceleración SSL. El análisis ha sido realizado comparando los costos de las soluciones de optimización obtenidas con y sin tarjetas. Los costos han sido minimizados sobre un período de tres años.

El análisis se centra en un escenario donde las torres de servidores se ven obligadas a ser ubicadas en un sitio llamado Proveedor de Servicio de Aplicación (*Application Service Provider; ASP*), como se muestra en la Fig. 4. Las aplicaciones requeridas por el usuario son solicitudes de generación de pagina HTTP (aplicación a_1) o HTTPS (aplicación a_2). Las aplicaciones del servidor calculan y envían las páginas web de comercio electrónico con un tamaño de 36 kB. Los parámetros de la aplicación del servidor y las aplicaciones SSL se han calculado utilizando los datos empíricos aplicados a este escenario en particular, que han sido presentados en la Sección II.A.

Un total de 6.000 usuarios activos han sido considerados. Cada uno de ellos tiene una sesión web activa y envía una solicitud de página cada 50 s. La mitad de los usuarios requiere una conexión segura mediante el protocolo SSL. La Fig. 4 muestra la topología del escenario considerado y los grupos de usuarios correspondientes. Dos nodos POP están conectados a un nodo *"core"* y dos sitios de usuario están conectados a cada nodo pop. La distancia entre el POP y los sitios de usuario es de 25 km, mientras que la distancia entre el "core" y los sitios POP es de 50 km. Un grupo de usuario se asigna a cada sitio de usuario. Todas las torres de servidores deben ser colocados en el sitio ASP ($serv(s_8) = 1$).

El algoritmo de minimización de costo se aplica variando las restricciones de retardo. En cada simulación, el mismo valor de $delay_{ij}$ se utiliza para todos los grupos de usuarios y las aplicaciones. Los dos casos, con o sin tarjetas de aceleración SSL, han sido comparados.

En la Fig. 5, el costo total TC se grafica en función del retardo promedio *end-to-end*. Los costos son significativamente más altos cuando los retardos requeridos son más estrictos, un 30% más altos a medida que disminuye el tiempo de respuesta máximo de 0,40 s a 0,20 s.

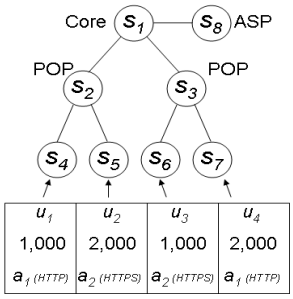

Figura 4. Topología del escenario de prueba.

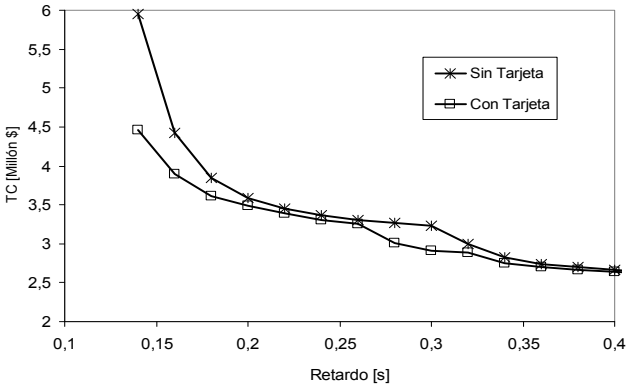

Figura 5. Costo total vs. retardo promedio *end-to-end* requerido.

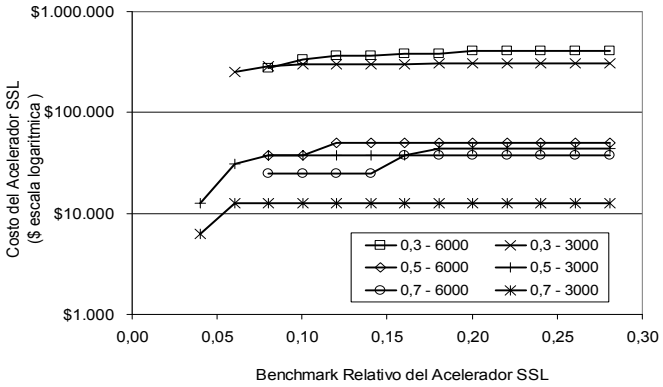

Figura 6. El compromiso entre el Costo y el *Benchmark* de la CPU de las tarjetas SSL.

De los resultados presentados en la Fig. 5, se puede ver que la aceleración SSL es conveniente sólo si el requisito de retardo es inferior a 0,4 s. Los ahorros de costo permitidos por las soluciones con las tarjetas de aceleración SSL comparadas con las soluciones sin tarjetas, son más altos cuando el retardo máximo permitido es menor: por ejemplo, los ahorros son del 25% con un retardo de 0,14 s y 2% con un retardo de 0,34 s. Un extenso análisis con un gran conjunto de distintos parámetros del sistema han demostrado que, en general, la aceleración SSL resulta conveniente cuando se tiene un rendimiento de retardo *end-to-end* estricto.

Además, se analizó el máximo ratio costo/rendimiento de las tarjetas de aceleración SSL, por lo cual hace conveniente la adopción de aceleración SSL. En la Fig. 6, el costo máximo utilizando el beneficioso del empleo de la aceleración SSL se grafica en función del *benchmark* de la CPU de la tarjeta de aceleración SSL. El *benchmark* relativo de las tarjetas se define como el ratio entre el *benchmark* de las tarjetas utilizadas en la optimización y el *benchmark* de un acelerador estándar SSL usado como una referencia para nuestro estudio, que es capaz de realizar hasta 1000 *handshake*s/s. En la Fig. 6, las curvas indican los objetivos de retardo igual a 0,3 s, 0,5 s y 0,7 s, a 6.000 o 3.000 usuarios (en este último caso, el número de usuarios en los grupos de usuarios se ha reducido a la mitad).

La Fig. 6 muestra que las curvas de costo contra *benchmark* decrecen rápidamente o incluso no tienen puntos por debajo de un cierto valor de *benchmark*. Este comportamiento conlleva a la necesidad de un mínimo *benchmark* para la CPU del acelerador SSL con el fin de alcanzar el objetivo de retardo. Cuando el mínimo valor de benchmark se encuentra con seguridad por encima del valor requerido, el costo máximo de la tarjeta, para obtener una aceleración SSL económicamente conveniente es casi constante.

Por ejemplo, con 6.000 usuarios y un objetivo de retardo de 0.3 s, se necesita un *benchmark* de alrededor de 7200 (es decir, 0.08 multiplicado por el *benchmark* de referencia del acelerador SSL igual a 90.300), estando dispuesto a pagar por ello al máximo unos 406.300 dólares. Por otra parte, con un objetivo de retardo más alto, 0.7 s, se estará dispuesto a pagar por ello al máximo alrededor de 37.500 dólares. Estaremos interesados en *benchmark*s más altos, sólo si, quisiéramos tener un sistema escalable, es decir, por ejemplo, si se pronostica un crecimiento significativo del número de usuarios en un futuro cercano.

La Fig. 6 muestra que existe un amplio espacio para la reducción de costo. Con un *benchmark* relativo igual a 1, se podría obtener un ahorro de costo, incluso con un costo mayor de más de 40 veces que el costo real de las tarjetas. Obviamente, el costo máximo relativo de las tarjetas que permite la reducción del costo total, disminuye a medida que disminuye el *benchmark* de tarjetas. Sin embargo, incluso para los *benchmark*s relativamente pequeños, las tarjetas son todavía económicamente ventajosas.

VII. Conclusiones

En este trabajo, se ha estudiado el problema de la optimización conjunta costo-rendimiento de los servicios *end-to-end* basados en SSL. Se ha llevado a cabo esta compleja tarea personalizando una nueva metodología para la optimización conjunta de los costos de hardware y comunicación, dadas las restricciones de rendimiento *end-to-end*.

La implementación concreta de nuestra metodología consiste en una herramienta de optimización que implementa un mecanismo de búsqueda tabú basado en un modelo matemático preciso y complejo de los recursos del sistema, la arquitectura, la configuración y los costos. Nuestra herramienta de optimización dimensiona todos los recursos del sistema (incluidos los servidores y *link*s de comunicación) con el fin de disponer de un sistema que cumpla con los requisitos de retardo *end-to-end* a costo mínimo.

Luego, se aplicó nuestra herramienta de optimización en escenarios y topologías de sistemas más complejos, a fin de comprobar si la aceleración SSL llevada a cabo por dispositivos especiales dedicados, que están actualmente disponibles en el mercado, puede ser más conveniente que los servicios SSL *end-to-end*. Hemos encontrado un compromiso entre el aumento de los costos de los servidores en la solución *end-to-end* (la carga del servidor se incrementa por las tareas relacionadas con SSL) y el aumento de los costos para los aceleradores SSL (los servidores más pequeños puede ser elegidos, ya que las tareas relacionadas con SSL son realizadas por los aceleradores.

Se examinaron escenarios con múltiples sitios y miles de usuarios. Se encontró que la aceleración SSL es económicamente conveniente solo si el objetivo de retardo requerido *end-to-end* es estricto. Por el contrario, si el retardo *end-to-end* de la capa de aplicación que el usuario puede tolerar es grande, la aceleración SSL no es conveniente. Siendo más detallados, se encontró que con más límites estrictos de retardo *end-to-end* (es decir, del orden de 100-200 ms), el sistema de menor costo construido con aceleradores SSL muestra una ventaja en costo del orden de 15-20%, en comparación con el mismo sistema diseñado con SSL *end-to-end*, es decir, sin aceleradores.

También se investigó el equilibrio entre costo y rendimiento de los aceleradores SSL y se indicaron las condiciones bajo las cuales ellos son convenientes.

Referencias

[1] A. O. Freier, P. Karlton, P. C. Kocher, "The SSL protocol version 3.0," IETF draft, 1996.
[2] T. Dierks, C. Allen, "The TLS Protocol Version 1.0". Available: http://www.ietf.org/rfc/rfc2246.txt, 1999.
[3] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol," RFC 2401, Nov 1998.
[4] L. D. Bisel, "The Role of SSL in Cybersecurity," *IT PROFESSIONAL*, pp. 22-25, 2007.

[5] K. Kant, R. Iyer, P. Mohapatra, "Architectural impact of secure socket layer on internet servers", *Proc. of International Conference on Computer Design 2000,* Austin, Texas, USA, Sept. 2000, pp. 7-14.
[6] L. Wu, C. Weaver, T. Austin, "CryptoManiac: a fast flexible architecture for secure communication", *ACM SIGARCH Computer Architecture News*, vol. 29, pp. 110-119, 2001.
[7] L. Zhao, R. Iyer, S. Makineni, L. Bhuyan, "Anatomy and performance of SSL processing", *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software, 2005. ISPASS 2005*, Austin, Texas, USA, March 2005, pp. 197-206.
[8] Cisco Systems, Inc., "Introduction to Secure Sockets Layer," White Paper, 2002.
[9] M. Khalil-Hani, V. P. Nambiar, M. N. Marsono, "Hardware Acceleration of OpenSSL Cryptographic Functions for High-Performance Internet Security", *Proc. of 2010 International Conference on Intelligent Systems, Modelling and Simulation (ISMS 2010)*, Liverpool, UK, 27-29 Jan. 2010.
[10] A. Roy-Chowdhury, J. S. Baras, "Performance-Aware Security of Unicast Communication in Hybrid Satellite Networks", *Proc. of IEEE ICC '09*, Dresden, Germany, 14-18 June 2009.
[11] D. A. Menasce, V. A. F. Almeida, *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning*, Prentice Hall, 2000.
[12] H. K. Jain, "A comprehensive model for the design of distributed computer systems", *IEEE Trans. Software Eng.*, vol.13, pp. 1092-1104, 1987.
[13] J. E. Blyler, G. A. Ray, "What's Size Got to do with it?: Understanding Computer Rightsizing", Wiley-IEEE Press, 1997.
[14] B. Gavish, H. Pirkul, "Computer and Database Location in Distributed Computer Systems", *IEEE Trans. on Computers*, vol. 35, 1986, pp. 583-590,
[15] A. Aue, M. Breu, "Distributed Information Systems: An Advanced Methodology", *IEEE Trans. Software Eng.*, vol. 20, pp. 594-605, 1994.
[16] L. W. Clarke, G. Anandalingam, "An integrated system for designing minimum cost survivable telecommunications networks", *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 26, pp. 856-862, 1996.
[17] J. Lui, M. Chan, "An efficient partitioning algorithm for distributed virtual environment systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, pp. 193-211, 2002.
[18] R. Subbu, A. C. Sanderson, "Network-based distributed planning using coevolutionary agents: architecture and evaluation," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 34, pp. 257-269, 2004.
[19] J. Guitart, V. Beltran, D. Carrera, J. Torres, E. Ayguadé, "Characterizing secure dynamic web applications scalability", *Proc. of 19th IEEE International Parallel and Distributed Processing Symposium*, Denver, CO, USA, April 2005.
[20] Standard Performance Evaluation Corporation. Available: http://www.spec.org/, 2010.
[21] http://support.microsoft.com/kb/265369/.
[22] W. Chou, "Inside SSL: Accelerating Secure Transactions," *IT PROFESSIONAL*, pp. 37-41, 2002.
[23] F. Glover, M. Laguna, *Tabu Search*, Kluwer Academic Publishers, 1997.
[24] D. Ardagna, C. Francalanci, M. Trubian, "Joint Optimization of Hardware and Network Costs for Distributed Computer Systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol.38, no.2, pp.470-484, 2008.
[25] Telecom Italia, "Collegamenti Diretti Retail", Available: http://www.wholesale-telecomitalia.it/, 2010.

Stefano Bregni (M'93-SM'99) es Profesor Asociado del Politecnico di Milano del curso de redes de telecomunicaciones. Nació en Milán, Italia, en 1965. En el 1990 se graduó en ingeniería de telecomunicaciones en el Politecnico di Milano. Desde 1991 trabaja en las áreas de SDH y sincronización de redes, con especial atención en medida de estabilidad de frequencia del reloj, primero con SIRTI S.p.A. y, posteriormente con CEFRIEL. En 1999, se unió al Politecnico di Milano como Profesor Asistente titular.

Ha sido Miembro Sénior de IEEE desde 1999. Desde 2004, ha sido Conferencista Distinguido de la IEEE Communications Society, donde desempeña las siguientes posiciones oficiales: Vocal de la Junta de Gobernadores (2010 a 2012), Director de Educación (2008-2011), Presidente del Comité Técnico de Transmisión, Acceso y Sistemas Ópticos (TAOS) (2008-2009; Vicepresidente 2002-2003, 2006-2007; Secretario 2004-2005) y Vocal del comité Globecom/ICC Technical Content (2007-2010). Es o ha sido Vice-Presidente del Programa Técnico de IEEE GLOBECOM 2012, Presidente de los Simposios GLOBECOM 2009 y Presidente de Simposio de otras ocho conferencias ICC/GLOBECOM. También fue Vicepresidente Técnico de la conferencia IEEE Optical Network Design and Modelling 2005. Editor del IEEE Global Communications Newletter y Editor Asociado de la revista IEEE Communications Surveys and Tutorials) Fue Conferencista tutor en cuatro conferencias IEEE ICC y GLOBECOM. Sirvió en comités ETSI y UIT-T de sincronización de red digital.

Es autor de alrededor de 70 artículos, la mayoría en conferencias y revistas de la IEEE, y de los libros *Synchronization of Digital Telecommunications Networks* (Chichester, UK: John Wiley & Sons, 2002; traducido y publicado en ruso por MIR, 2003) y Sistemas de Trasmisión PDH y Multiplexacion SDH (*Sistemi di trasmissione PDH e SDH - Multiplazione*. Milano, Italia: McGraw-Hill, 2004). En la actualidad sus temas de investigación se enfocan principalmente en modelado de tráfico y redes ópticas.

Paolo Giacomazzi (M'07) recibió el grado de maestría en ingeniería electrónica en el Politécnico de Milano en el año 1990 y el grado de especialización de CEFRIEL, Milán, en 1990. Se desempeñó como Teniente del Cuerpo de Ingenieros del Ejército Italiano desde 1990 hasta 1992. Desde abril 1992 hasta octubre 1998 ha estado como profesor asistente de redes de telecomunicaciones en el Politecnico di Milano. En 1995 ha sido investigador visitante del Centro de Comunicaciones Inalámbricas de la Universidad de Mississippi, Oxford, MS. De noviembre 1998 a octubre 2001, ha sido profesor asociado de comunicaciones eléctricas y redes de telecomunicaciones en la Universidad de Messina. Desde noviembre de 2001, es profesor asociado de redes de telecomunicaciones en el Politecnico di Milano. Desde 1997, es un Editor Asociado de la revista IEEE Network y actualmente se desempeña como Editor Asociado en el libro revisión de características de la revista IEEE Network.

Alessandro Poli (S'07) nació en Cremona, Italia, en 1981. Recibió el título de pregrado y Máster en ciencias de Ingeniería de Computadores del Politecnico di Milano, Italia. En 2010 obtuvo el título de doctorado en Ingeniería de la Información del Departamento de Electrónica e Información del Politecnico di Milano. Sus intereses de investigación incluyen redes de videostreaming peer-to-peer, diseño de redes inalámbricas metropolitanas, optimización de la infraestructura tecnológica de información y comunicaciones, y descubrimiento de servicios semánticos.