Cost-Performance Optimization of SSL-Based Secure Distributed Infrastructures

Stefano Bregni*, *Senior Member, IEEE*, Paolo Giacomazzi, Alessandro Poli

Politecnico di Milano, Dept. of Electronics and Information, Piazza Leonardo Da Vinci 32, 20133 Milano, ITALY
Tel.: +39-02-2399.3503 – Fax: +39-02-2399.3413 – E-mail: {bregni, giacomaz, poli}@elet.polimi.it

***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.**

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

I. Introduction

The security of Internet traffic can be provided using various protocols: Secure Socket Layer (SSL) [1], Transport layer Security (TLS) [2] and, at the network layer, Internet Protocol Security (IPSec) [3]. All these protocols and mechanisms provide secure communications between applications that run on public domains such as Internet. In this paper, we focus on SSL, whose security services are used across the world by distributed applications accessible by a web interface. SSL is a protocol built on top of TCP. The application layer uses SSL features to obtain confidentiality, message integrity and endpoint authentication [4].

Researchers have studied the performance of SSL, usually measured as the overhead of secure processing in web servers, as studied for example in [5]. Other studies focus on cryptographic algorithms [6] and propose optimizations for accelerating crypto operations. In [7], SSL is profiled in detail and the time spent by processors for SSL elaboration is provided, for all the phases of SSL negotiation.

Since SSL consumes resources of both clients and servers, cost and performance are critical issues of SSL deployment as, in order to guarantee the target end-to-end performance levels at the application layer, additional resources must be supplied to carry out the SSL-related tasks. This, in turn, calls for additional investments costs. Moreover, there are several options on how to implement SSL security services: currently, a rational, general and quantitative method does not exist.

In this paper, we provide a cost-performance oriented planning algorithm, carried out at the system level, of SSL-secured infrastructures. Our algorithm is based on a detailed mathematical model of both end-to-end performance and costs of the Information and Communication Technology (ICT) infrastructure for the provision of SSL-secured communications over both public domains such as Internet or private domains such as the virtual private network of a company.

Two main options are available for the deployment of SSL. First, SSL secure connection can be provided in an end-to-end fashion, from the client and the remote application server. Alternatively, SSL can be accelerated by a specific device, the SSL accelerator (see for example [8][9]), that terminates SSL connections on behalf of the client.

Our method for the selection of the optimal strategy is based on a model of the overall technological infrastructure, comprising hardware and network components [11], which is used to carry out an optimal design of infrastructures by minimizing costs required to satisfy given performance requirements of both computing and communication ([10]–[13]).

In the literature, the design of infrastructures is usually carried out by splitting the problem into two distinct optimizations of hardware and network. The first optimization problem is how to distribute the overall computing load of a distributed system onto multiple machines, in order to minimize hardware costs ([12][14][15]). The second problem is where to locate machines that need to exchange information in order to minimize network costs ([16]–[18]). These two problems have been studied separately in the literature. However, design decisions on both alternatives are strongly interrelated.

Overall, we consider a multi-site scenario and address the following design choices: a) allocation of server farms to sites; b) allocation of server applications on shared server farms; c) allocation of SSL acceleration devices (called SSL acceleration cards, or simply cards, in the following sections) to network routers; d) dimensioning all devices.

II. SSL Acceleration

A. The Secure Socket Layer (SSL) Protocol

In this paper, we focus on HTTP and HTTPS applications. HTTPS is the HTTP protocol used on top of the SSL protocol. SSL communications between client and server are supported by SSL sessions, i.e. relations between client and server.

The Handshake Protocol is responsible for session creation: it negotiates cryptographic parameters to be shared by various connections, avoiding frequent renegotiations. It allows clients and servers to mutually authenticate, to negotiate ciphering and MAC algorithms and to securely exchange the session key for the SSL Record Protocol.

The SSL Record Protocol provides privacy and integrity services. It fragments input messages into blocks, compresses each block, applies a MAC (Message Authentication Code) code, and then encrypts the resulting block by using a private key exchanged by the Handshake Protocol. It adds a header to the message and sends it via TCP. Received messages are decrypted, verified, uncompressed, assembled back and then sent to the application layer.

In [19], the SSL handshake execution time has been measured on a 1.4 GHz Xeon host (with SPECint2000 [20] benchmark $B = 516$). Reported values are 2 ms for a resumed SSL handshake and 175 ms for a full SSL handshake. We note that the Microsoft Internet Explorer web browser performs a SSL session renegotiation, which causes a new full SSL handshake [21] every 120 s. The CPU utilization of a web server, processing 1 kB web pages transmitted over a secure SSL channel (HTTPS), has been measured in [7]; the CPU time expenditure has been reported broken into page generation (7.6%), page encryption (2.4%) and key exchange/SSL handshake (90%) phases. It has been observed that the page generation and page encryption times are proportional to page sizes.

B. SSL Acceleration Cards and Content Switches

SSL operations are typically executed by a web server process. While a HTTP server listens for new connections on port 80, a SSL server listens for connections on port 443. The SSL process uses web servers processing power, subtracting resources to the other processes running on the servers and consequently increasing response times.

The use of SSL adapter cards allows mitigating the heavy SSL processing load. SSL adapter cards are placed inside servers and plugged in a motherboard slot. Web server still run a SSL process, but a set of functions is delegated to the adapter card, thus decreasing server load. These cards supports up to 1000 new connections per second [22].

In order to lower the web server load, SSL can be executed by content switches, used to balance traffic intelligently among servers in data centres, based on content availability and server load.

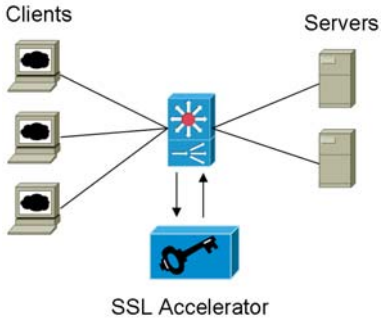

Fig. 1: Content Switch with integrated SSL accelerator.

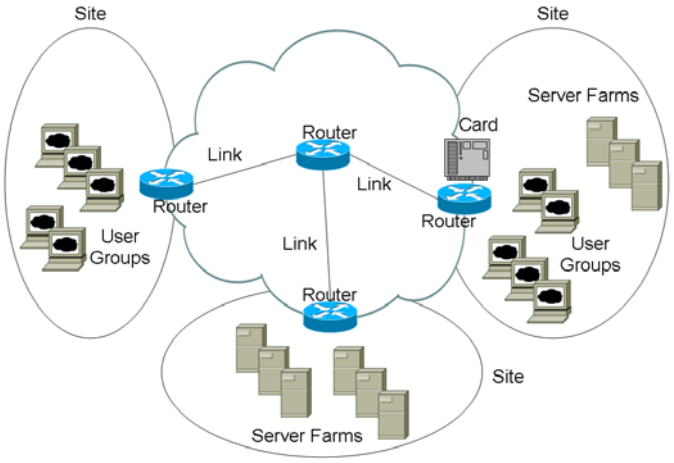

Fig. 2: General model of the IT infrastructure.

The configuration in Fig. 1 centralizes SSL functions in a single device. The need of processing many connections requires SSL accelerators in the content switch [8]. The content switch uses SSL accelerators to decrypt SSL traffic from clients to servers and encrypt it on the opposite way. A content switch should also be able to balance the load among all servers in the server farm, tracing SSL sessions.

III. The Optimization Problem

We consider the general model of IT infrastructure shown in Fig. 2. Users and server farms are located in sites defined as geographically separated locations accommodating either users, server farms, or both. Within a site, devices are connected through a high-speed local area network (LAN), which has greater performance and smaller cost than a geographic network connecting sites. Thus, LANs are not considered for cost-performance optimization. Sites are connected by a geographic network made of links and routers. Routers may or may not be equipped with a card performing SSL acceleration.

A. Technology Requirements

This section specifies the formalization of requirements for a reference organization with a set of sites S.

1) Sites. A site $s_i \in S$ is defined as a set of technology resources connected by a Local Area Network (LAN). For each site pair (s_i, s_j), the distance is $dist(s_i, s_j)$.

2) SSL application. An SSL application $a_i^{SSL} \in A^{SSL}$ is an application providing SSL cryptography services. It is characterized by CPU time $time_{CPU}(a_i^{SSL})$ required to encrypt a message on a reference server of type $ref(a_i^{SSL}) \in ST$, disk time $time_{DISK}(a_i^{SSL})$, RAM space $space_{RAM}(a_i^{SSL})$, and disk space $space_{DISK}(a_i^{SSL})$.

3) Server application. A server application (or application process) $a_i \in A$ is characterized by CPU time $time_{CPU}(a_i)$, disk time $time_{DISK}(a_i)$, RAM space $space_{RAM}(a_i)$, disk space $space_{DISK}(a_i)$, number of bits of a request $req(a_i)$, number of bits of a response $resp(a_i)$, SSL application $ssl(a_i) \in A^{SSL}$ serving the application a_i.

4) User group. A user group $u_i \in U$ is a set of $|u_i|$ users with common computing requirements, i.e., using the same set of

applications. Each user group is characterized by the set of server applications used, as specified by the matrix $\{\gamma_{ij}\}$ where $\gamma_{ij} = 1$ if the application $a_i \in A$ is used by u_i else $\gamma_{ij} = 0$; the site $site(u_i) \in S$ where the group is located, the response time requirement $delay_{ij}$ and the frequency of requests $msg(u_i, a_j)$ of user group u_i for the application a_j.

B. Hardware Resources

The computing requirements of the organization can be satisfied by the hardware resources defined in the following.

1) Link type. A link type $lt_k \in LT$ is characterized by service cost $cost(lt_k)$, minimum and maximum distances $distMin(lt_k)$ and $distMin(lt_k)$, and capacity $cap(lt_k)$.

2) Router type. A router type $rt_k \in RT$ is characterized by acquisition cost $cost(rt_k)$, service rate $BPS(rt_k)$, maximum backplane service rate $B(rt_k)$, classifier coefficient $KC(rt_k)$, and routing coefficient $KR(rt_k)$.

3) Server type. Each server type $st_k \in ST$ is characterized by acquisition cost $cost(st_k)$, RAM size $ram(st_k)$, disk space $disk(st_k)$, CPU performance benchmark $bench_{CPU}(st_k)$ (i.e., the ratio of the execution time of a reference application on a reference CPU to the execution time of the same application on st_k's CPU) and disk performance benchmark $bench_{DISK}(st_k)$.

4) SSL acceleration card type. A card type $ct_k \in CT$ is characterized by acquisition cost $cost(ct_k)$ and CPU performance benchmark $bench_{CPU}(ct_k)$.

5) Server farm. A server farm $sf_i \in SF$ is a set of $|sf_i|$ servers of the same type $type(sf_i) \in ST$. The maximum number of servers allowed in server farms is referred to as φ.

6) Router. A router $r_i \in R$ is an instance of a router type $type(r_i) \in RT$ assigned to site s_i.

7) Link. A link $l_{ij} \in L$ is an instance of a link type $type(l_{ij}) \in LT$ assigned to a site pair (s_i, s_j).

8) SSL acceleration card. A card $c_i \in C$ is an instance of a card type $type(c_i) \in CT$ assigned to router c_i. Note that a router may or may not be assigned a card.

IV. The Optimization Model

A. Decision Variables

Optimization alternatives are represented by the following decision variables.

1) Allocation of applications to server farms: $x_{ij} = 1$ if the application $a_i \in A$ is placed on server farm sf_j else $x_{ij} = 0$.

2) Allocation of server farms to sites: $y_{jk} = 1$ if the server farm sf_j is placed on site s_k else $y_{jk} = 0$.

3) Allocation of cards: $w_{lk} = 1$ if card c_l is installed on router r_k else $w_{lk} = 0$.

4) Allocation of SSL applications to farms: $z_{pj} = 1$ if the SSL application $a_p \in A^{SSL}$ is placed on server farm sf_j else $z_{pj} = 0$.

B. Response Time

For each such pair (u_i, a_j) that $\gamma_{ij} = 1$, we must identify the direct and reverse *routing paths* from the site of user group u_i, $site(u_i) \in S$, to the site of the server farm where application a_j is allocated. These paths can be identified by means of any shortest path algorithm. Our tool implements the Dijkstra shortest path algorithm using a metric of 1 for each link.

The response time depends on the average load of the devices crossed by the requests sent by user group u_i to application a_j along direct and reverse paths. The devices along direct and reverse paths are represented by way of ordered lists

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

where *req* and *res* indicate the direct and reverse path, respectively. Each list includes all devices crossed along the corresponding path (links, routers, cards, and server farms).

Let $rtime_{ij}$ be the response time experienced by the request from user group u_i to application a_j, including the delay along both the direct and reverse paths. The response time along the direct path is supposed to include the actual service time of the request. Response time is calculated as

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

Application requests hit all devices along the direct and reverse paths with the same frequency $\lambda_{ij} = |u_i| \cdot msg(u_i, a_j)$.

The SSL acceleration application is performed by the card in the router of the destination site of the request. SSL acceleration for a given request cannot be executed in sites different from the destination of the request, as the request would proceed unencrypted to the destination site. If the router placed in the destination server farm site has not a card, then the server farm must execute the SSL application.

The response time for devices (viz. links, routers, cards, and server farms) is computed according to device parameters and the amount of requests crossing single devices. The response time for each request is computed as the sum of the average time spent by each request (or response) in a set of multiple class M/M/1 queues and the corresponding service times (the propagation time is also considered in link devices).

C. Objective function

The objective function to be minimized is the total cost, *TC*, of the technology resources selected to satisfy requirements over a time horizon indicated as *years*.

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

D. Constraints

Thirteen constraints have been identified in our model:

1) each application must be allocated on one server farm;
2) each server farm must be allocated on one site;

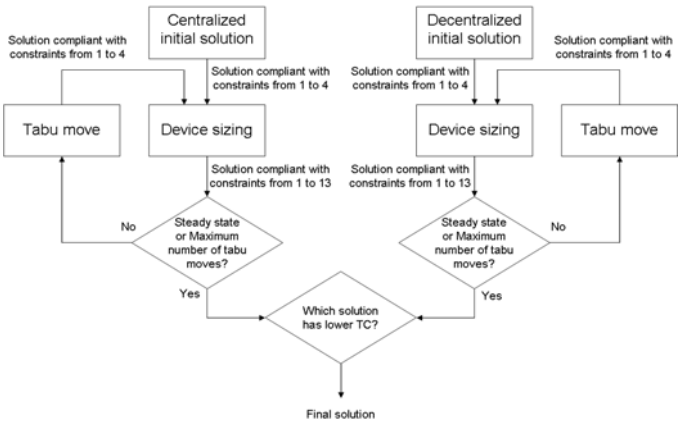

Fig. 3: Flow diagram of the cost minimization algorithm.

3) server farms only on sites allowed to host server farms;
4) each router must be assigned at most one card;
5) all user groups must be served with a response time lower than their application delay requirements;
6) only link types that meet distance constraints;
7, 8) only server types that meet RAM and disk space constraints;
9) on the maximum number of servers allowed in server farms;
10, 11, 12, 13) on the maximum load on server farms, cards, routers and links (100%).

V. Cost Minimization Algorithm

The cost minimization algorithm, outlined in Fig. 3, aims at identifying the minimum-cost solution that satisfies technology requirements with corresponding technology resources. The algorithm is based on the tabu-search (TS) approach [23].

Two initial solutions are identified first: a fully centralized solution allocating all applications on one server farm and a fully decentralized solution allocating each application on a separate server farm. They both meet constraints 1, 2, 3, 4.

Then, the device-sizing phase identifies a set of technology resources meeting all requirements 1─13, including application delays, and calculates the corresponding total cost *TC*. Tabu moves are made to reduce TC until either the cost of solutions reaches a steady state for a predefined number of moves or a maximum number of moves is reached. The device sizing phase is repeated after each tabu move. The final solution is selected as the one with lowest cost among those obtained from the centralized and decentralized initial solutions.

In our tabu-search implementation, the neighbourhood of a solution is explored by executing four types of moves: *application displacement*, *server farm displacement*, *card insertion*, and *card removal*. An *application displacement* removes a server application from a server farm and allocates it on a different server farm. A *server farm displacement* removes a server farm from a site and allocates it on a different site. A *card insertion* adds a card to a router, while a *card removal* removes a card from a router. Tabu moves must satisfy constraints 1, 2, 3, and 4. The execution of a move changes the configuration of decision variables *x*, *y*, *w*. This configuration is an input to the *device sizing* phase of the cost minimization algorithm described in Fig. 3.

The device sizing phase aims at identifying the minimum-cost set of technology resources satisfying requirements within the current configuration of decision variables *x*, *y*, *w*. Sizing is performed in two steps: a first sizing followed by a sequence of upgrades and downgrades.

The first sizing assigns to each device the minimum-cost type that satisfies requirements according to commonly used rules of thumb. According to these rules of thumb, device types are chosen in such a way that the maximum load of all queues is lower than 60% [24].

Given a configuration of decision variables *x*, *y*, *w* and the *type()* assignment for each device – as computed in the first sizing step – the $rtime_{ij}$ for each (u_i, a_j) pair can be determined. A configuration score is then computed to measure the distance between actual and maximum response time:

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

If $score = 0$, the current configuration meets delay constraints, else a device upgrade is needed. If $score > 0$, pairs with a response time higher than maximum delay are addressed, by considering for upgrade the devices along corresponding request paths. An upgrade replaces the current type of a device with the lowest-cost type that has a cost greater than that of the current type. Note that a more costly server farm type can be obtained, either by using a different server type or by changing the total number of servers in the farm.

The algorithm performs a series of upgrades, each of them lowering the configuration score, until $score = 0$ is reached. Upgrades that deliver a higher score reduction are performed first. All intermediate configurations are feasible since they also comply with constraints 1─4.

When a solution with $score = 0$ is found, the configuration space is explored by means of a series of downgrades and upgrades in order to identify the minimum-cost set of devices. Device downgrades bring to solutions with lower total cost. A series of downgrades and series of upgrades are iteratively performed until a given maximum number of configuration changes is reached. The best solution found with all upgrades and downgrades with $score = 0$ is selected as the output of the device sizing phase.

VI. Empirical Verification

Empirical tests have been carried out, using a prototype tool we developed implementing the cost minimization algorithm. This tool includes a database of commercial technological resources and related cost data.

Server types have been surveyed from the web sites of four main vendors: Dell, HP, IBM, and ION Computers. Performance data have been collected for 460 servers. Computing capacity is benchmarked by means of SpecInt2000 [20]. A server farm is supposed to be composed by at most $\varphi = 30$ servers. Thus, our tool can choose among 13800 different server farms.

The database of routers is based on Cisco products. Two types of routers are considered, with cost and performance data provided by Cisco.

An SSL accelerator card type is available, with parameters set to make it able to perform up to 1000 handshakes/s, i.e. a

common value for market available devices [22]. The cost of the card has been initially considered equal to 50,000 USD.

Cost data of links have been collected from the official pricing documents of digital leased lines in Italy [25].

In the following, we provide empirical evidence of the cost savings granted by cards supporting the SSL acceleration service. Analysis has been performed by comparing the costs of optimization solutions obtained with and without cards. Costs have been minimized over a three-year period.

Analysis focuses on a scenario where server farms are forced to be placed in one site called Application Service Provider (ASP), as shown in Fig. 4. The applications required by the user are HTTP (application a_1) or HTTPS (application a_2) page generation requests. The server applications compute and send e-commerce web pages with size 36 kB. Application parameters of server and SSL applications have been computed by using the empirical data presented in Section II.A applied to this particular scenario.

A total number of 6,000 active users have been considered. Each of them has an active web session and send a page request every 50 s. Half of the users require a secure connection by SSL protocol. Fig. 4 shows the topology of the considered scenario and the corresponding user groups. Two POP nodes are connected to a core node and two user sites are connected to each pop node. The distance between POP and user sites is 25 km, while the distance between Core and POP sites is 50 km. One user group is assigned to each user site. All server farms must be placed in the ASP site ($serv(s_8) = 1$).

The cost minimization algorithm is applied with varying delay constraints. In each simulation, the same $delay_{ij}$ value is used for all user groups and applications. Two cases, with or without the SSL acceleration cards, have been compared.

In Fig. 5, the total cost *TC* is plotted as a function of the required average end-to-end delay. Costs are significantly higher when delay requirements are tighter, about 30% higher as the maximum response time decreases from 0.40 s to 0.20 s.

From the results reported in Fig. 5, we notice that SSL acceleration is convenient only if the delay requirement is below 0.4 s. Cost savings allowed by solutions with SSL acceleration cards, compared to solutions without cards, are higher when the maximum allowed delay is lower: for instance, they are equivalent to 25% with delay 0.14 s and to 2% with delay 0.34 s. An extensive analysis for a large set of different system parameters has shown that, in general, SSL acceleration is convenient with a strict end-to-end delay performance.

Furthermore, we analyzed the maximum cost/performance ratio of SSL acceleration cards, which makes convenient adopting SSL acceleration. In Fig. 6, the maximum cost making advantageous employing SSL acceleration is plotted as a function of the CPU benchmark of the SSL acceleration card. The relative benchmark of cards is defined as the ratio of the benchmark of cards used in the optimization to the benchmark of a standard SSL accelerator, used as a reference for our study, able to perform up to 1000 handshakes/s. In Fig. 6, the curves refer to delay targets equal to 0.3 s, 0.5 s and 0.7 s, and to 6,000 or 3,000 users (in the latter case, the number of users in the user groups has been halved).

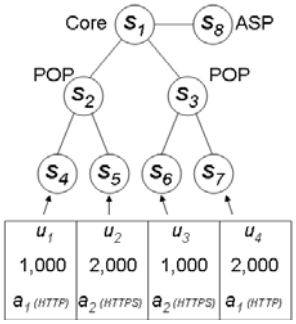

Fig. 4: Topology of the test scenario.

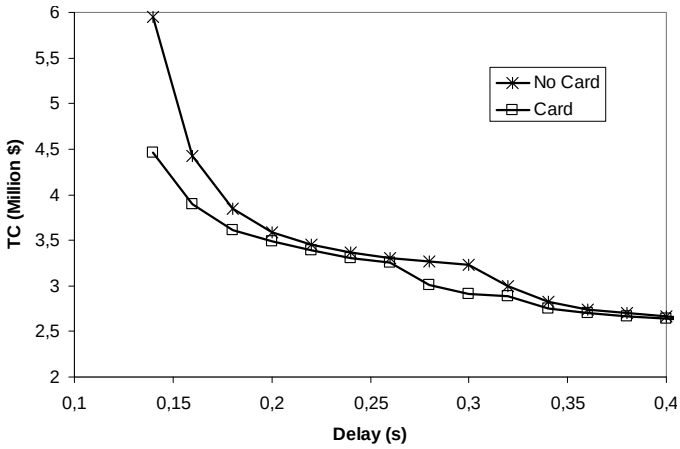

Fig. 5: Total cost in the scenario vs. the required average end-to-end delay.

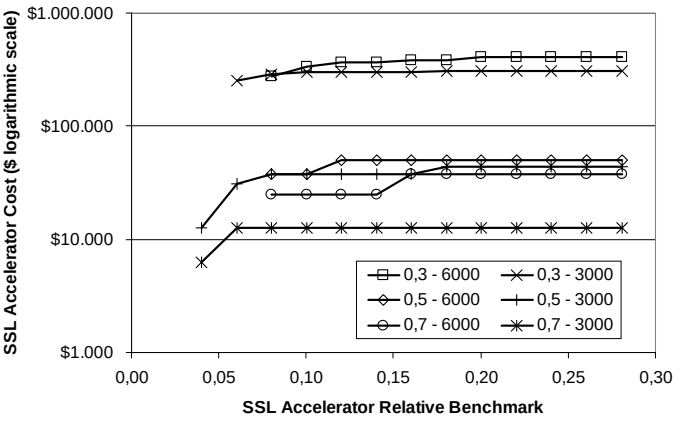

Fig. 6: Cost vs. CPU benchmark trade-off of SSL cards.

The graph in Fig. 6 shows that the cost/benchmark curves exhibit a sharp knee shape or even have no points below a certain benchmark value. This behaviour corresponds to the need of a minimum SSL accelerator CPU benchmark to meet the delay target. When the minimum benchmark value is safely above the required value, the maximum card cost to obtain an economically convenient SSL acceleration is nearly constant.

For example, with 6,000 users and delay target 0.3 s, we need a benchmark of roughly 7,200 (i.e., 0.08 multiplied by the reference SSL accelerator benchmark equal to 90,300), being willing to pay for it at most about 406,300$. On the other hand, with higher delay target 0.7 s, we will be willing to pay

for it at most about 37,500$. We are interested in higher benchmarks only if we would like to have a scalable system, that is, for example, if we forecast a significant growth of the number of users in the near future.

The Fig. 6 shows that there is an ample space for cost reduction. With a relative benchmark equal to 1, we could obtain cost saving even with a cost more than 40 times greater than the actual cost of cards. Obviously, the maximum relative cost of cards enabling the reduction of total cost decreases as the card benchmark diminishes. However, even for relatively small benchmarks, cards are still economically advantageous.

VII. Conclusions

In this work, we have studied the problem of the joint cost-performance optimization of end-to-end SSL-based services. We have carried out this complex task by customizing a novel methodology for the joint optimization of hardware and communication costs, given end-to-end performance constraints.

The concrete implementation of our methodology consists in an optimization tool implementing a tabu-search mechanism based on a precise and complex mathematical model of the system resources, architecture, configuration and costs. Our optimization tool sizes all the system resources (including servers and communication links) in order to have a system complying with the end-to-end delay requirements with minimum cost.

Then, we applied our optimization tool in more complex scenarios and system topologies, in order to check whether SSL acceleration carried out by special dedicated devices, currently available on the market, can be more convenient than end-to-end SSL services. We found a trade-off between increasing costs for servers in the end-to-end solution (server load is increased by SSL-related tasks) and increasing costs for SSL accelerators (smaller servers can be chosen as SSL-related tasks are performed by accelerators).

We examined scenarios with multiple sites and thousands of users. We found that SSL acceleration is economically convenient only if the end-to-end target delay requirement is strict. On the contrary, if the end-to-end application-layer delay that the user can tolerate is large, SSL acceleration is not convenient. More in detail, we have found that with rather strict end-to-end delay bounds (i.e., on the order of 100-200 ms), the smallest-cost system constructed with SSL accelerators exhibits a cost advantage on the order of 15%-20%, compared to the same system designed with end-to-end SSL, i.e. without accelerators.

We also investigated the cost-performance trade-off of SSL accelerators and reported the conditions under which they are convenient.

References

[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.