Design and Evaluation of A High-Performance ATM Firewall Switch and Its Applications

Jun Xu and Mukesh Singhal
Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210
{jun,singhal}@cis.ohio-state.edu

July 17, 1998

Abstract: We present the design of a value-added ATM switch that is capable of performing packet-level (IP) filtering at the maximum throughput of 2.88 Gbps per port. This firewall switch nicely integrates the IP level security mechanisms into the hardware components of an ATM switch so that most of the filtering operation is performed in parallel with the normal cell processing and most of its cost is absorbed into the base cost of the ATM switch. The firewall switch employs a novel scheme to avoid or reduce the latency caused by filtering. We analyze in detail the performance of the firewall switch in terms of the throughput and latency, and address related design issues. Applications of our firewall switch as Internet and intranet security solutions are also discussed.

1 Introduction

A security hole of current ATM deployments is that an end-to-end ATM connection generally bypasses Internet and intranet firewalls if there are any. Terminating ATM connections at an intermediate packet filtering firewall for inspection is not a fix, as traditional router-based firewall will not be able to deliver the filtering throughput that is comparable to even low-end ATM rate of OC-3 (150 Mbps). The overhead caused by SAR (Segmentation and Reassembly) may further bring the performance down. To circumvent this problem, the ATM Forum favors the avoidance of packet filtering by exerting discretions at the connection establishment time. Based on this principle, two alternative access control schemes have been proposed. Smith [13] proposes that access control decisions be made at connection establishment time based on the higher layer information (e.g., source and destination IP addresses, ports, etc.) that is contained in the ATM *signaling message* as *information elements.* After the connection is established, the access control device "gets out of the way." Obviously, this is unacceptable because an intruder can always lie about the service he wants to access at the connection establishment time, and there is no way to check the contents of a connection once the connection is established. Pierson [12] tries to fix this problem by proposing that "a new SVC (Switched Virtual Connection) is requested when each new service is started." However, this requires that the ATM layer be notified whenever a new socket is opened, which entails considerable change to the whole TCP/IP stack and existing applications [4]. Even worse, this requires a new SVC for each transport layer flow, which may lead to the problem of *VC explosion.* Apparently call screening alone will not solve the problem of network-level access control in an ATM network. Therefore, new ATM firewall architecture that is capable of performing packet-level filtering at high speed is called for.

At the time of writing, StorageTek's ATLAS product is the only ATM packet filtering device available in the literature [6]. ATLAS is a line filter that scans an ATM physical link to perform the packet-level filtering at the rate of OC-3 (150 Mbps). Two performance boosting strategies are used in ATLAS. First, to avoid SAR, for each packet ATLAS only checks the first cell, which contains the IP header, protocol, TCP/UDP ports, and TCP flags (if applicable), to determine whether or not the packet is "safe." If the packet is considered safe, all following cells that belong to it are passed or otherwise dropped. Second, ATLAS uses a *policy cache architecture* [6] to dramatically speed up the process of deciding whether or not a packet header is safe. The core unit of this architecture is a cache block called *policy cache*. Each entry of the policy cache is a combination of VPI/VCI, source and destination IP addresses, and source and destination TCP/UDP ports that is considered "safe." When the first cell of a packet arrives, it is compared with each entry in the policy cache. If cache hit occurs, cells of the packet are forwarded. Otherwise, the first cell will go through a software screening process and all other cells of the packet will be buffered in a queue. If the cell is found unsafe, the whole packet is dropped. Otherwise, the packet is allowed to pass and an entry that contains the header pattern of the packet is added to the policy cache. The policy cache is implemented using CAM (Content Addressable Memory), which enables simultaneous searching of all memory locations to find a match with the pattern being searched for. Therefore, ATLAS can decide whether a packet header hits the policy cache very quickly.

However, the ATLAS product does have its limitations and drawbacks. First, it does not accept IP packets with IP option fields because IP option can be as large as 40 bytes and may "push" the TCP header to the second cell. Second, it is not friendly to those who have to manage and administer it because TCP/IP filtering rules for any newly-established PVC or SVC have to be manually configured. This is impractical in the future ATM networks where a large number of SVCs will be established on-the-fly in a short period of time.

In this paper, we present an implementation-ready hardware design of a value-added ATM switch that performs packet-level filtering at high speed (2.88 Gbps), and present its applications as Internet and intranet security solutions. Sections 2 and 3 presents respectively the logical and physical design of our firewall switch. In Section 4, we analyze the throughput and the latency of our firewall and explore related design issues. In Section 5, we present the applications of our firewall switch. Section 6 concludes the paper.

2 Logical Design of Our Firewall Switch

As shown in Fig. 1, three related security services, namely, *call screening service* (CSS), *packet filtering service* (PFS), and *firewall management service* (FMS), are implemented in our firewall switch. Their functions are described in detail in the following sections.

2.1 Call Screening Service (CSS)

When an ATM *signaling message* requesting a new SVC to be established arrives at the firewall, CSS decides whether it is secure to establish the connection and/or whether the new connection needs to be filtered based on the identity and authentication information [1] contained in the signaling message. If the connection is considered secure, the firewall switch will allow its traffic bypass packet filtering mechanisms. We will see in Section 3.1 that implementation of such a bypass is trivial. For simplicity of discussion, in the following we assume all connections that travel through the firewall switch need to be filtered. If the connection needs to be filtered, CSS will advise the PFS to filter the traffic in the connection and supply PFS with packet filtering rules as shown in Fig. 1. In a firewall switch, filtering rules for various connections may be different and may be generated on-the-fly as follows. First, the firewall switch may determine source and destination IP addresses/prefixes either from the higher layer IEs (information elements) contained in the authenticated signaling message or from the source and destination ATM addresses by querying a preconfigured ATM address to IP address/prefix mapping table. Then, from the intended services as indicated in the signaling message or in the security policy, the firewall switch may determine the source and the destination TCP/UDP ports allowed in the connection. Finally, these two pieces of information are glued together to generate a set of filtering rules. For example, if the source

[1] Authentication of the signaling message is described in detail in [14].

is a router for subnet 164.107.*.*, and the destination is a router for subnet 192.41.245.*, and the destination subnet only allows HTTP (port 80) and TELNET (port 23) from the source subnet, then the TCP/IP filtering rules are:

Src IP	Dest IP	Src Port	Dst Port	Protocol	Action
164.107.*.*	192.41.245.*	>1023	80	TCP	Allow
164.107.*.*	192.41.245.*	>1023	23	TCP	Allow

(All other packets are blocked)

Filtering rules for securing reverse traffic will be generated in the same fashion. A default set of TCP/IP filtering rules will be used by packet filtering service if CSS does not have enough information to generate such filtering rules on-the-fly. Such automatic configuration of TCP/IP filtering rules is vital in future ATM environments where a large number of SVC connections will be established within a short period of time. This relieves the network managers from worrying about setting up filtering rules for each SVC when it is established.

2.2 Packet Filtering Service (PFS)

PFS inspects the header of IP packets to block "unsafe" packets while allowing the "safe" packets to pass. We introduce a new scheme, called "Last Cell Hostage (LCH)," into our firewall switch to reduce the latency incurred by packet filtering. Two performance boosting schemes used in ATLAS, namely, filtering only on the first cell and hardware caching of the security policy (see Section 1.2), are also used in our firewall switch.

In ATLAS, if the first cell misses the policy cache, all cells of the packet have to be buffered while waiting for a slow software inspection process to finish (even when the last cell of the packet arrives later than the inspection decision). In contrast, with our LCH scheme, all cells of a packet except the last one is allowed to pass even if a cache miss occurs. Only the last cell is kept as "hostage" before the inspection is finished. If the inspection finds out that the packet is safe, the last cell is passed. Otherwise, it is dropped and is substituted with a pseudo last cell whose payload is generated randomly so that CRC failure of the whole packet at the receiver is guaranteed. This prevents the drop of one packet from affecting the following packet, which will otherwise be mixed with the previous packet and cause corruption. The introduction of LCH has two advantages:

- Even when a packet header misses the policy cache, if the packet is reasonably long, the software-based filtering process can be finished before the last cell arrives. This means that no latency is experienced even when a cache miss occurs. Section 4.2.2 will demonstrate LCH's effectiveness in avoiding or reducing latency caused by packet filtering.

- LCH makes it easy to process IP packets with option. The technical challenge in filtering IP packets with option is that the decision may not be made until the second cell arrives. ATLAS does not process such packets because it would have to consider each of such a packet as missing the policy cache. When the percentage of such packets is high, the processing overhead would be unbearable in ATLAS. However, with LCH, only some fields of the first cell needs to be kept as the state information, and the software-based filtering process is postponed until the second cell arrives. Since all of the cells except the last one can be passed without delay, the IP packets with option can be processed almost as fast as those without.

Even though the LCH scheme only blocks the last cell of a packet while unconditionally allowing other cells to pass, it is by all means a safe approach. If an attacker sends a "bad" packet to the system it attacks, the packet will be discarded by the receiver because its last cell is "corrupted."

2.3 Firewall Management Service (FMS)

FMS controls and manages CSS and PFS, and provides user-friendly administration tools to network managers. These tools allow network managers to perform operations such as specifying or updating call screening rules, monitoring abnormal user activities, and disabling connections that violate the security policy. FMS logs two types of events forwarded from CSS and PFS. The first type of events includes unsafe signaling messages detected by CSS and unsafe packets detected by PFS. Such information will be used to analyze what type of intrusion is involved and how security policy needs to be updated in response to the intrusion. The other type of events includes the signaling and traffic profile information about each connection. Such information allows our firewall switch detect abnormal user activities that may indicate an intrusion attempt.

3 The Physical Design of Our Firewall Switch

Our firewall switch modifies various components of a normal ATM switch to implement the security services described in the last section. Conceptually, a normal ATM switch consists of five functional modules, namely, input module (IM), output module (OM), cell switching fabric (CSF), call admission control (CAC), and system management (SM), as shown in Fig. 2. IM extracts cells from the incoming SONET payload envelope, determines the destination port of each cell, and writes it to an internal tag appended to the cell. CSF routes each cell to its destination port indicated in the internal tag. CAC assembles the signaling cells into a signaling message and processes it. SM processes the OAM cells and performs switch management functions. All outgoing cells arrive at OM through CSF. OM processes and removes the internal tags appended to the cells and puts the cells into the SONET payload envelopes for transmission. Our firewall switch nicely integrates the IP level security mechanisms into these hardware components of an ATM switch so that most of the filtering operation is performed in parallel with the normal cell processing and most of its cost is absorbed into the base cost of the ATM switch. In our firewall switch, IM and OM are enhanced to implement packet filtering service, CAC is enhanced to implement call screening service, and SM is enhanced to implement firewall management service. Section 3.1 through 3.4 shows the modifications to IM, OM, CAC, and SM, respectively.

3.1 Input Module (IM)

In IM, translation of the VPI/VCI of each incoming cell to determine its output VPI/VCI/port is performed in a functional block called *cell translation*. In a firewall switch, this block is enhanced to perform filtering operations as well. Fig. 3 shows the overall picture of the cell translation block in a firewall switch. Area A depicts the header translation process in a normal switch. In flow 1, VPI/VCI of an incoming cell (stored in the cell buffer) is used as the key to search in the VP/VC table. The corresponding output VPI/VCI/port is fed to normal header translation logic (flow 2), which will in turn update the VPI/VCI field of the cell buffer (flow 3) and write the output port information to the internal tag (flow 4).

Area B shows the packet filtering function in a firewall switch. The VP/VC table is enhanced to contain a number of firewall-related fields. EXP-1st and EXP-2nd indicate whether the incoming cell is the first and second cell of the packet, respectively. PASS and LCH indicate whether the packet should be passed or put into LCH, respectively. SNO is the serial number assigned to each packet. SNO_LCH and OMCHK (Output Module Check) will be explained in Section 3.2. If the firewall switch needs to distinguish whether a connection needs to be filtered, a one-bit field can be added to VP/VC table for this purpose. These fields keep track of cell-level and packet-level state information regarding the connection. They (flow 6), as well as PTI (Payload Type Identifier) and fields in IP payload (flow 5) will be fed to the *cell-level filtering logic* (CFL). It will make decision whether the packet should be passed or put into LCH. It then updates the firewall-related fields in VP/VC (flow 7) and writes information to the firewall-related fields in internal tag (flow 10).

Fig. 4 shows the flowchart of the CFL. If the cell is a non-first cell of a packet, then decision on the packet has already been made and is recorded in the PASS (to pass the packet) and LCH (to put the cell into LCH), respectively. Otherwise ("EXP-1st?"), CFL sends this first cell to Express Packet Filtering Logic (EPFL) for further inspection (flow 8 in Fig. 3) and waits for the reply (flow 9 in Fig. 3). The decision is either to pass the packet or to put the packet into LCH, which is recorded in PASS or LCH field (set to TRUE), respectively. Besides, if the decision is to put the packet into LCH, OMCHK bit should be turned TRUE and SNO_LCH will record the serial number of this packet. We will explain their use in section 3.2. If the packet contains IP option, EXP-2nd is set TRUE to indicate that the second cell is expected for inspection. When such a cell arrives ("EXP-2nd?"), it will be forwarded to EPFL as well. When the last cell of a packet arrives ("Last cell?") as indicated by the PTI field in the cell header, the DROP, LCH, EXP-1st, and will be reset or updated to "welcome" the arrival of the next packet. The value of LCH, OMCHK, and SNO will be carried to OM by T-LCH, T-OMCHK, and T-SNO fields in the internal tag respectively.

EPFL employs a policy cache (invented by StorageTek) to boost the filtering performance. The policy cache stores the <VPI/VCI, src IP, dst IP, source Port, destination Port, Protocol> tuples which are found secure. When a cell is forwarded from CFL, it is checked against the cache block for a match. If a cache hit occurs, the decision "PASS" is sent to CFL. If a cache miss occurs, the decision "LCH" is sent to CFL and the cell is forwarded to Software Packet Filtering Module (SPFM) for further inspection (flow 11 in Fig. 3). SPFM checks (in software) the cell against packet filtering rules to see if it is secure. Once the final decision regarding whether the packet should be passed or dropped is available from SPFM (flow 12 in Fig. 3), EPFL will forward the decision to OM using internal cells or other internal communication mechanisms. Besides, a tuple that consists of the corresponding header fields of the packet will be added to the policy cache. For packets that contain IP option, the first cell may be stored as the state information if needed.

The throughput of CFL/EPFL can be estimated as follows. We assume that VP/VC table is implemented using 50ns off-the-shelf DRAM. With off-the-shelf CMOS technology, CFL can be implemented within one memory cycle because its logic is very simple. We will show in Section 4.2.1 that an IP packet today typically have 8 cells in length. We conservatively estimate the upper bound of the time that needs to be spent on such an average-length packet as follows. A read from VP/VC table is needed before processing each and every cell and a write to VP/VC table is needed after processing the first, the second (when the packet contains IP option), and the last cell. Processing the first cell also involves a lookup in a Content Addressable Memory module (the policy cache) in EPFL, which is conservatively estimated to be 150ns[2] Even without any pipelining, the first, second (possibly), and the last cell takes respectively a maximum of 6, 3, and 3 memory cycles to process respectively and each of the other cells takes 2 memory cycles. Therefore the average time spent on each cell is $(6+3+2+2+2+2+2+3)/8 = 2.75$ (memory cycles). At this rate, EPFL can process at least 7.27 millions cells per second, which is equivalent to 3.08 Gbps.

[2]Normal access time for current 1k-size CAM is between 50ns to 90ns. However, larger-size CAM (e.g., 64k) that may be needed by our firewall switch will have long access time due to propagation delay of OR gates used to concatenate small-size CAM chips into a large-size CAM.

We will see in Section 4.2.1 that this throughput is greater than the estimated throughput of the firewall switch, which is 2.88 Gbps. Thus, CFL/EPFL is not the bottleneck of our firewall switch.

3.2 Output Module (OM)

In a standard switch, OM processes the internal tag of each cell for housekeeping purposes and updates output VPI/VCI for multicast cells. OM also contains a VP/VC table, which contains fields for housekeeping information such as the number of cells transported in each connection and a field that stores output VPI/VCI for multicast connections.

In a firewall switch, OM is involved in implementing the LCH scheme. When the LCH scheme needs to be applied to a packet, IM will indicate that in the T-LCH field of the internal tag. Decision on that packet will also be forwarded to OM as soon as it is available. The responsibility of OM is to keep the last cell of the packet as "hostage" until the decision on the packet arrives. To accomplish this function, VP/VC table is enhanced to include two new fields. One is LCH_Cell, which points to the buffer that buffers the last cell of an LCH packet and other cells that are blocked due to the LCH cell. The other is SEL_NO, which stores the serial number of the last decision received (PASS or DROP). Also, there is a decision queue that stores the decisions forwarded from IMs, yet to be applied to LCH packets at OM.

The following processes are involved in implementing the LCH scheme. The first process is invoked when the last cell of an LCH packet arrives. The second process is invoked when other packets that follow the LCH packet in the same connection arrive. The third process is invoked when a decision on an LCH packet arrives.

1. OM identifies the last cell of an LCH packet from the LCH bit in the internal tag and the PTI field in the cell header. When such a cell is identified, OM will check whether the decision on the packet has arrived (SEL_NO $\geq$ T-SEL_NO?). If the decision is available, it is retrieved from the decision queue and is applied to the cell.

2. While the decision on the LCH packet is pending, packets that follow the LCH packet in the same connection should also be buffered in order to preserve the cell order. This is

taken care of by the T-OMCHK bit in the internal tag of a cell. IM will turn T-OMCHK bit on once an LCH decision has been made to a packet in that connection. When OM detects such a cell, OM will check whether there is a cell buffered in LCH_Cell. If there is not any, the cells should be allowed to pass. Otherwise the cells need to be queued up.

3. When the decision on a packet arrives, OM puts it in the decision queue and updates the SEL_NO field with the serial number contained in the decision message. Also, cells queued up in that connection will now be processed by OM.

The overhead of the first and third process can be absorbed into standard housekeeping process of OM. The overhead incurred by the second process is equivalent to the processing of multicast cells because both of them involve a VPI/VCI table lookup.

As each OMCHK cell results in as much overhead as a multicast cell, it is desirable to make the number of such cells as small as possible. In the ideal situation, once a queue for OMCHK cells in a VP/VC becomes empty, the OMCHK bit is cleared in the corresponding VP/VC entry in IM. This is accomplished by OMCHK Clearance Protocol, which works as follows.

Once a queue becomes empty, an internal cell is sent back to IM, which contains the VPI/VCI of the connection and the serial number of the last packet that is put into LCH and has been processed by OM (recorded in SEL_NO). When IM receives the internal cell, it checks whether the serial number matches the SEL_LCH field in the enhanced VP/VC table. If it matches, which means that no new LCH packet has been sent to OM after that one, IM can safely turn OMCHK bit off. Otherwise, this message is ignored.

3.3 Call Admission Control (CAC)

CAC in a standard switch processes and routes signaling messages. In a firewall switch, CAC implements the call screening service as discussed in Section 2.1. It compares the identities of the communicating parties contained in the signaling message with call screening rules to decide whether the connection is allowed to be established and/or whether the connection needs to be filtered. If the connection needs to be filtered, CAC will set up an entry in the VP/VC table for the new connection and initiate the firewall-related fields in the VP/VC table. It will also generate the packet filtering rules to be executed by Software Packet Filtering Module. CAC may be assisted by fast cryptographic hardware to quickly authenticate the digital signature contained in the signaling message if applicable.

3.4 System Management (SM)

In a standard switch, SM handles all the management plane functions including fault management, performance management, configuration management, accounting management, security management, and traffic management [3]. In a firewall switch, it manages firewall functions that need to be performed by the switch, which involves the following tasks:

- Maintaining firewall-related *managed objects* such as call screening rules.
- Executing the commands sent from network management station such as updating call screening rules.
- Monitoring the performance of packet filtering, such as the throughput and average latency of filtering operation at each port.

4 Performance of The Firewall Switch

In this section, we study two important performance metrics of our firewall switch: throughput and latency. Related design issues such as the size of the policy cache and its replacement policy will also be addressed. In Section 4.1, we first present the statistical model and mathematical analysis for calculating the throughput and the latency of the firewall switch. In Section 4.2, data from real-world measurements and simulation are plugged in the expressions to obtain the numerical results.

4.1 Performance Analysis

4.1.1 The Throughput of the Firewall Switch

Given a certain traffic load, the throughput of the firewall switch, THR, can be expressed as

$$THR = \frac{STHR * NC}{MR} \tag{1}$$

where MR is the miss ratio of the policy cache under the traffic load, NC is the average number of cells that a packet in the traffic load consists of, and STHR is the throughput of software packet filtering module (SPFM) in IM. NC can be directly measured from the traffic sample and STHR can be estimated using the throughput data of traditional IP firewalls. MR, however, merits more discussion.

MR is closely related to the concept of *flow*. A *flow* is a unidirectional stream of IP packets with common tuple <source-IP-address, source-port, destination-IP-address, destination-port, IP-protocol> in the header. When the first packet of a flow arrives at the firewall switch, obviously it will miss the policy cache and will be inspected in SPFM. If it is found secure, a cache entry that contains the tuple will be added to the policy cache. If the cache entry is not evicted from the policy cache before the last packet of the flow arrives, there is exactly one cache miss for all the packets inside the flow. If this is true for each and every flow, the theoretically minimum miss ratio 1/PF is achieved, where PF is the average number of packets per flow and can be directly measured from the traffic load. However, this theoretical limit can only be achieved

- if the firewall has infinite amount of policy cache, or
- if the firewall has "decent amount of cache" (explained in the next section) and employs an optimal cache replacement policy (needs to predict the future).

Yet, neither of which is realistic. In the following sections, we show that a miss ratio of only about 10% higher than the theoretical minimum can be achieved with "decent amount of cache" and simple FIFO cache replacement policy.

4.1.2 On the Size of Policy Cache

In this section, we explain what is "decent amount of cache" and what happens if the cache size is less than that. Given a certain traffic load, we define F(t) as the number of *active flows* at time t during an observation period [0, T]. A flow is *active*[3] at time t if the first packet of the flow arrives before t and the last packet of the flow arrives after t. $MAXF = \max_{0 \le t \le T} F(t)$ is the maximum number of active flows during the observation period [0, T]. Let CS denote the size of policy cache in our firewall switch. If $CS \ge MAXF$, with an optimal cache replacement policy, the theoretically minimum miss ratio of 1/PF can be achieved. The reason is that whenever the first packet of a flow arrives, since the active number of flows at that moment is no more than MAXF, there must be one flow that is no longer active and whose header is present in the policy cache. Clearly, this entry can be evicted without causing cache miss.

On the other hand, if $CS < MAXF$, we say that a *cache overflow* happens at time t when $CS < F(t)$. At such a time, entries that cache the headers of active flows have to be evicted. Assume that a non-optimal cache replacement such as FIFO is used, MR(MAXF) is the miss ratio achieved when the cache size is MAXF and MR(CS) is the miss ratio achieved when the cache size is CS. We define the overflow function $OF(t) = \max(F(t) - CS, 0)$, which is equal to the number of uncached active flows due to *cache overflow* at time t. We define overflow index (OI) as $OI = \frac{\int_0^T OF(t)dt}{\int_0^T F(t)dt}$. OI can be viewed as the percentage of uncached active flows through [0, T]. Then OI * 100% of the flows have a miss ratio of 100% and the rest have a miss ratio of MR(MAXF). Therefore, when there are only CS cache entries, the miss ratio is

$$MR(CS) = OI * 100\% + (1 - OI) * MR(MAXF) \tag{2}$$

We will see in Section 4.2.1 that MR(MAXF) is no more than 10% with FIFO replacement policy given a traffic load similar to today's Internet traffic. Therefore, OI is a significant factor in this equation. For example, if OI is 10%, the miss ratio MR(CS) is at least two times of MR(MAXF), which means that the throughput with CS cache entries is only half of the throughput with MAXF cache entries. Therefore, we define "decent amount of cache"

[3]NLANR's definition of active flow takes into consideration the 64-second timeout.

as a size larger than MAXF. Additionally, even larger cache size helps reduce cache miss ratio if a suboptimal cache replacement policy is used.

4.1.3 The Latency of the Firewall Switch

In this section, we demonstrate the effectiveness of LCH in avoiding and reducing latency caused by packet-level filtering. Let us consider the following scenario. Assume that the total volume of the link is C (cells/sec), we would like to investigate how a packet on a connection of average bandwidth B (cells/sec) will be delayed assuming that it misses the policy cache. The following three metrics are studied:

- $Pr(DE_{withLCH} = 0)$, which is the probability that no latency is experienced by the packet. Such probability is always 0 if LCH is not used because packet filtering in software takes a while.

- $EXP[DE_{withLCH}]$, which is the average latency experienced by the packet if LCH is used.

- $EXP[DE_{w/oLCH}]$, which is the average latency experienced by the packet if LCH is not used.

We assume that cells of different connections are fully interleaved. During each cell slot, the probability λ that an incoming cell belongs to the connection can be expressed as $\lambda = B/C$. Let T denote the inter-arrival time (in the unit of cell slots) between two consecutive cells in the connection, then $Pr[T = i] = \lambda(1-\lambda)^{i-1}: \quad i = 1, 2, 3, ..., \lambda = B/C$. When λ is small, this discrete geometric distribution can be approximated as the exponential distribution $Pr(T > t) = e^{-\lambda t}$.[9] Suppose the first cell of the packet arrives at time 0, the *response time* of the SPFM is R cell time, and N is the number of cells in the connection that arrive within time window (0, R], then N is a random variable with Poisson distribution $Pr[N = k] = \frac{(\lambda R)^k e^{-\lambda R}}{k!}$. Then no latency is experienced by the packet if no more than L - 2 cells arrive within time window (0, R]. Therefore,

$$Pr[DE_{withLCH} = 0] = Pr[N \leq L-2] = \sum_{k=0}^{L-2} \frac{(\lambda R)^k e^{-\lambda R}}{k!} \tag{3}$$

We define the latency experienced by a packet as the maximum among the latency experienced by individual cells of the packet. With LCH, the cell that was delayed most is the last cell; without LCH, the cell that was delayed most is the first cell since it has to wait for the last cell to arrive. Therefore

$$DE_{withLCH} = \begin{cases} R - \hat{T} & : \quad R \geq \hat{T} \\ 0 & : \quad R < \hat{T} \end{cases} \qquad DE_{w/oLCH} = \begin{cases} R & : \quad R \geq \hat{T} \\ \hat{T} & : \quad R < \hat{T} \end{cases} \tag{4}$$

where $\hat{T}$ is the arrival time of the L^{th} cell assuming the arrival time of the first cell is 0. The density function of $\hat{T}$ is

$$f_{\hat{T}}(t) = f_{ER_{L-1,\lambda}}(t) = \frac{\lambda^{L-1} t^{L-2} e^{-\lambda t}}{(L-2)!} \tag{5}$$

where $ER_{n,\lambda}$ is the Erlang random variable[9]. Therefore,

$$EXP[DE_{withLCH}] = \int_0^R (R-t) f_{\hat{T}}(t)dt = R(1 - \sum_{k=0}^{L-2} \frac{(\lambda R)^k e^{-\lambda R}}{k!}) - \frac{L-1}{\lambda}(1 - \sum_{k=0}^{L-1} \frac{(\lambda R)^k e^{-\lambda R}}{k!}) \tag{6}$$

$$EXP[DE_{w/oLCH}] = \int_0^R R f_{\hat{T}}(t)dt + \int_R^\infty t f_{\hat{T}}(t)dt = R(1 - \sum_{k=0}^{L-2} \frac{(\lambda R)^k e^{-\lambda R}}{k!}) + \frac{L-1}{\lambda}(\sum_{k=0}^{L-1} \frac{(\lambda R)^k e^{-\lambda R}}{k!}) \tag{7}$$

4.2 The Numerical Results

In this section, we calculate numerical results of the firewall switch performance by plugging in the parameters measured from Internet traffic [8]. The performance analysis of our firewall switch owes thanks to NLANR's (National Laboratory for Applied Network Research) Internet traffic raw trace and statistics data (available online). Though their data were collected for studying unrelated issues such as TCP implementation behavior, routing dynamics, and web caching, it is also quite useful for the quantitative estimate of the throughput and latency of our firewall switch. NLANR logged and collected statistics on traffic parameters transferred through the Internet backbone, among which PF (average number of packets in a flow) and PL (the average length of a packet in bytes) will be used in our analysis. Since only less than 1% of the traffic is non end-to-end (mostly traffic between backbone routers), it is very representative of the traffic that flows between the Internet and an enterprise network. Additionally, since more than 87% of the traffic is HTTP, FTP, SMTP, and NNTP, it is also very representative of the traffic inside an enterprise intranet. Therefore, the traffic statistics is suitable for analyzing the firewall switch performance for both Internet and intranet use.

4.2.1 The Throughput of the Firewall Switch

We are now ready to calculate THR using (1). According to NLANR's traffic statistics, $PL \approx 343(bytes)$ and $PF \approx 14.92(packets/flow)$. Then calculated from PL, $NC \approx 8(cells/packet)$ based on AAL 5 encapsulation. A survey on the performance of the state of the art software-based IP firewall technology [5, 10, 11] shows that $STHR \approx 50000(packets/sec)$. What is still missing from (1) is MR. As discussed in Section 4.1.1, the theoretically minimum MR is 1/PF and the real value of MR depends on the size of the policy cache and the cache replacement policy being used. We conducted a simulation on the NLANR's Internet trace data to examine the miss ratio of LRU (as a reference) and FIFO. With $CS \approx 1.2 * MAXF$, LRU's miss ratio is only 4% higher than the theoretically minimum miss ratio, and FIFO's miss ratio is about 10% higher. Since implementing FIFO (virtually no cost) involves much less cost than implementing LRU, it is chosen as the cache replacement policy. Therefore, $MR \approx 0.0737$ considering the 10% degradation due to the use of FIFO cache replacement policy. According to (1):

$$THR = \frac{STHR * NC}{MR} = \frac{50000 * 8}{0.0737} = 5.43 * 10^6(cells/sec) = 2.88Gbps \tag{8}$$

Therefore, our firewall switch can achieve a maximum throughput of 2.88 Gbps based on the current Internet traffic characteristics. As discussed in Section 3.1, CFL/EPFL will not prevent that throughput from being achieved.

4.2.2 The Latency of the Firewall Switch

In Section 4.1.3, we derived the expressions to calculate three metrics. In this section, we plug in real-life data to get the numerical results. We first need to calculate R, which is used in (6) and (7). Let ATR be the actual throughput of the firewall switch, since software packet filtering module (SPFM) is the system bottleneck, the system utilization of SPFM is $\rho =$ ATR/THR. The response time R, which is the interval between the time a packet (the first cell) that misses the policy cache is sent to the SPFM and the time when the inspection finishes, can be expressed as (assuming M/M/1) R = PT/(1 - ρ), where PT = 1/THR is the processing time of a packet at SPFM.

There are two types of traffic which may miss the policy cache. The first type is the first packet of a TCP flow. In this case, it is either a 44-byte-long SYN packet (inverse direction) or a 40-byte-long dataless ACK (reverse direction), which consists of only two cells. The second type is the first packet of a UDP flow or the non-first packet of a TCP flow (miss the policy cache due to cache replacement). The average length of such packets should be roughly equal to the average packet length of packets, which is 8 cells long. In both cases, we discuss the latency caused by policy cache miss under heavy (75%), moderate (50%), and light load (25%).

Fig. 5 shows $Pr(DE_{withLCH} = 0)$ experienced by the first type of packet (two-cell-long) that misses the policy under the aforementioned three types of system loads. It shows that LCH is effective in avoiding latency when the system is lightly loaded, or when the system is moderately loaded and the bandwidth of the connection is small (no more T3 rate). Fig. 6, 7, and 8 compares $EXP[DE_{withLCH}]$ with $EXP[DE_{w/oLCH}]$ under heavy, moderate, and light loads, respectively. We can see that LCH is effective in avoiding or reducing latency when the system is lightly or moderately loaded, or when the system is heavily loaded and the bandwidth of the connection is small. At the very least, LCH guarantees that the latency is less than R, which is around 80 μs under heavy load. As this type of packet is the first and second packet of a TCP connection, most probably the application on top of the connection was in the initialization stage (not in the middle of a high-resolution video conference session), and a small delay like this is generally tolerable.

Fig. 9 shows $Pr(DE_{withLCH} = 0)$ experienced by the second type of packet (eight-cell-long) that misses the policy under the three types of system loads. Fig. 10, 11, and 12 compares $EXP[DE_{withLCH}]$ with $EXP[DE_{w/oLCH}]$ under heavy, moderate, and light loads, respectively (Note that the Y-axis is drawn on a logarithm scale). We see that LCH not only yields a high probability of avoiding latency, but also significantly reduces the average latency in all cases. Actually, it is this type of packets that most needs such an improvement for two reasons. First, without LCH, packets as long as eight cells cause a significant latency when the bandwidth of the connection is small (e.g., more than 400μs for a 10 Mbps connection in Fig. 10) while LCH keeps the latency lower than R at all times. Second, such a packet may either carry a frame in a high-resolution video conference application (TCP) or a file block transferred using NFS (UDP). Delay of such packets is obviously undesirable and LCH effectively avoids and reduces latency in such cases.

5 Applications

An obvious use of the firewall switch is to place it at the "chokepoint" between an enterprise ATM network and a public ATM WAN to secure all traffic that goes between the two networks. In the following we show a novel use of our firewall switch to fix a security hole of MPOA (Multi-Protocol Over ATM) [2] that has long been left unsolved. In MPOA, *cut-through connections* generally bypasses firewall routers if there are any, which may lead to security breaches [7]. The only solution so far proposed came from the ATM Forum. In this proposal, the first several packets between two end systems, which according to MPOA's flow detection process must travel through the *default route*[4], will get filtered by the intermediate MPOA servers. Only when these packets are found secure will the cut-through connection be allowed to be established. This approach is insecure because a malicious end system can always send "benign packets" in the beginning. Once the cut-through connection is established, it can do anything it pleases over that connection.

If there were a physical chokepoint inside the ATM network such that all cut-through connections must travel through it, then placing our firewall switch at that point would be a perfect firewall solution. However, generally there won't be such a physical chokepoint because a security domain (typically an IP subnet defined over a virtual LAN (VLAN)) can span multiple *edge devices* and an *edge device* can have ports that belong to different security domains. Instead, our firewall solution for MPOA is based on our novel concept of "logical chokepoint". In this approach, a *connection classification* protocol is executed between an edge device and an MPOA server to decide whether a cut-through connection is considered secure. If the connection is found insecure, it will be forced to travel through a firewall switch, which is deployed in the "middle" of an ATM network. In this way, participating

[4] *MPOA trigger* then needs to be used cautiously.

edge devices act together to form a protection shield against internal security breaches. The *econnection classification* protocol is nicely embedded into the MPOA address resolution protocol as follows. The MPOA *resolution request* that the source *edge device* sends to the *MPOA server* will contain the IP addresses of the source and the destination end systems, and the authentication information of the source end system such as a digital signature if applicable. MPOA server will decide whether the connection is secure based on these information. The decision will be included in the MPOA *resolution reply* sent from the MPOA server to the edge device. Forcing a connection to travel through a firewall switch can be implemented as follows. The source edge device specifies in the signaling message the ATM address of the firewall switch as the final destination. The true destination is encoded into a user-to-user *information element* or *generic identifier transport information element* [1]. When this signaling message is forwarded to the firewall switch, the firewall switch knows the game being played and will retrieve the final destination from the corresponding information element and use it as the new destination to forward the signaling message. Our technical report [15] presents this scheme in detail. We demonstrate in [15] that the only significant cost of the scheme is the firewall switch, which is amortized over huge number of end systems it is able to protect. We also show in [15] that the proposed scheme has no negative performance impact on the edge devices and MPOA servers.

6 Summary

We designed a high-performance firewall switch that operates at the maximum throughput of 2.88 Gbps. Our firewall switch nicely integrates the IP level security mechanisms into the hardware components of an ATM switch so that most of the filtering operation is performed in parallel with the normal cell processing and most of its cost is absorbed into the base cost of the ATM switch. The proposed switch employs LCH scheme to avoid or reduce latency caused by packet filtering. We conducted an in-depth performance analysis of the firewall switch in terms of throughput and latency, and addressed related design issues. Finally, we proposed a novel use of our firewall switch to fix a security hole in MPOA that has been left unsolved.

References

[1] ATM Forum – "Private Network-Network Interface Specification Version 1.0 (PNNI 1.0)", March 1996.

[2] ATM Forum, Technical Committee, Multi-Protocol Over ATM Version 1.0, Feb. 1997.

[3] T. Chen, and S. Liu, ATM Switching Systems, Artech House, Boston, MA, 1995.

[4] J. Hughes, and A. Guha, "Requirements for Secure Packet-Level Access over ATM", ATM Forum/95-1126.

[5] Keylabs Inc., "Test Final Report – Firewall Shootout Networld+Interop", May 1998, URL: http://www.keylabs.com.

[6] B. Kowalski, StorageTek Corp., "ATLAS Policy Cache Architecture", StorageTek Corp., MN, 1997, Available URL: http://www.network.com/.

[7] D. Minoli, and A. Alles, LAN, ATM, and LAN Emulation Technologies, Artech House, Inc., Norwood, MA, 1996.

[8] National Laboratory for Applied Network Research (NLANR), "FIX-West statistics data summaries", Available URL: http://www.nlanr.net/.

[9] R. Nelson, Probability, Stochastic Processes, and Queueing Theory, Springer-Verlag, NY, 1995.

[10] D. Nessett, and P. Humenn, "The Multilayer Firewall", *1998 Network and Distributed System Security Symposium* (NDSS'98), San Diego, CA, 1998.

[11] D. Newman, H. Holzbaur, and K. Bishop, "Firewalls: Don't Get Burned", *Data Communications Magazine* online, March 1997, URL: http://www.datacomm.com.

[12] L. Pierson, and T. Tarman, "Requirements for Security Signalling", ATM Forum/95-0137.

[13] T. Smith, and J. Stidd, "Requirements and Methodology for Authenticated Signalling", ATM Forum/94-1213.

[14] T. Tarman, "Phase I ATM Security Specification", ATM Forum BTD-SECURITY-01.13, July 1997.

[15] J. Xu, and M. Singhal, "A Firewalling Scheme for Securing MPOA-based corporate Intranets," OSU CIS Technical Report (OSU-CISRC-6/98-TR19), June 1998.

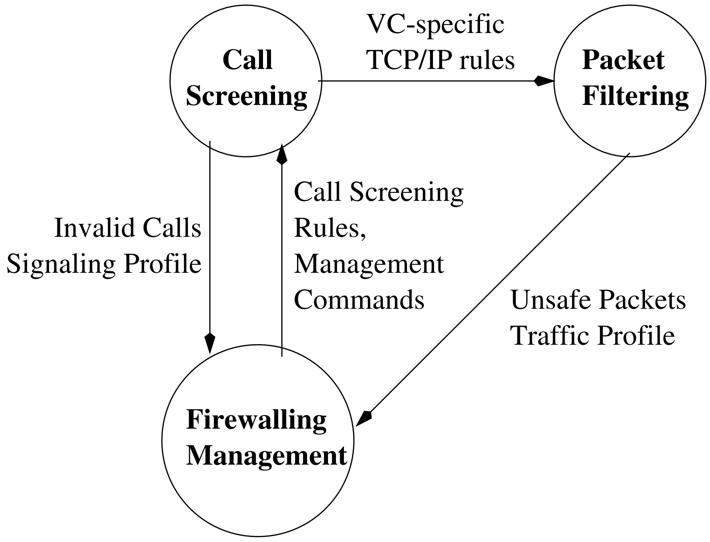

Figure 1: Logical design of our firewall switch

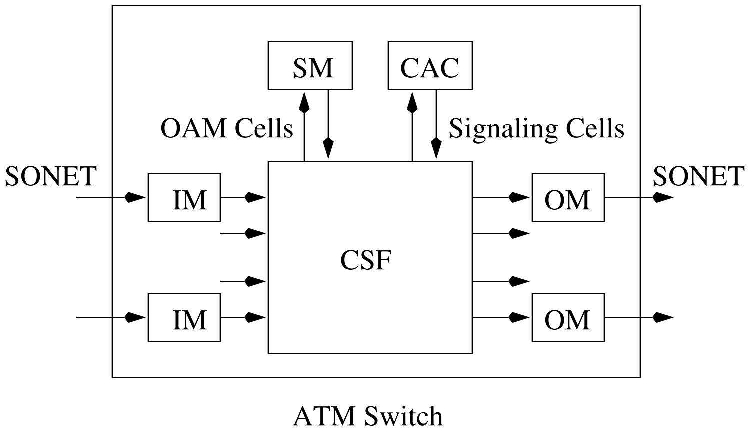

Figure 2: The conceptual organization of an ATM switch[3]

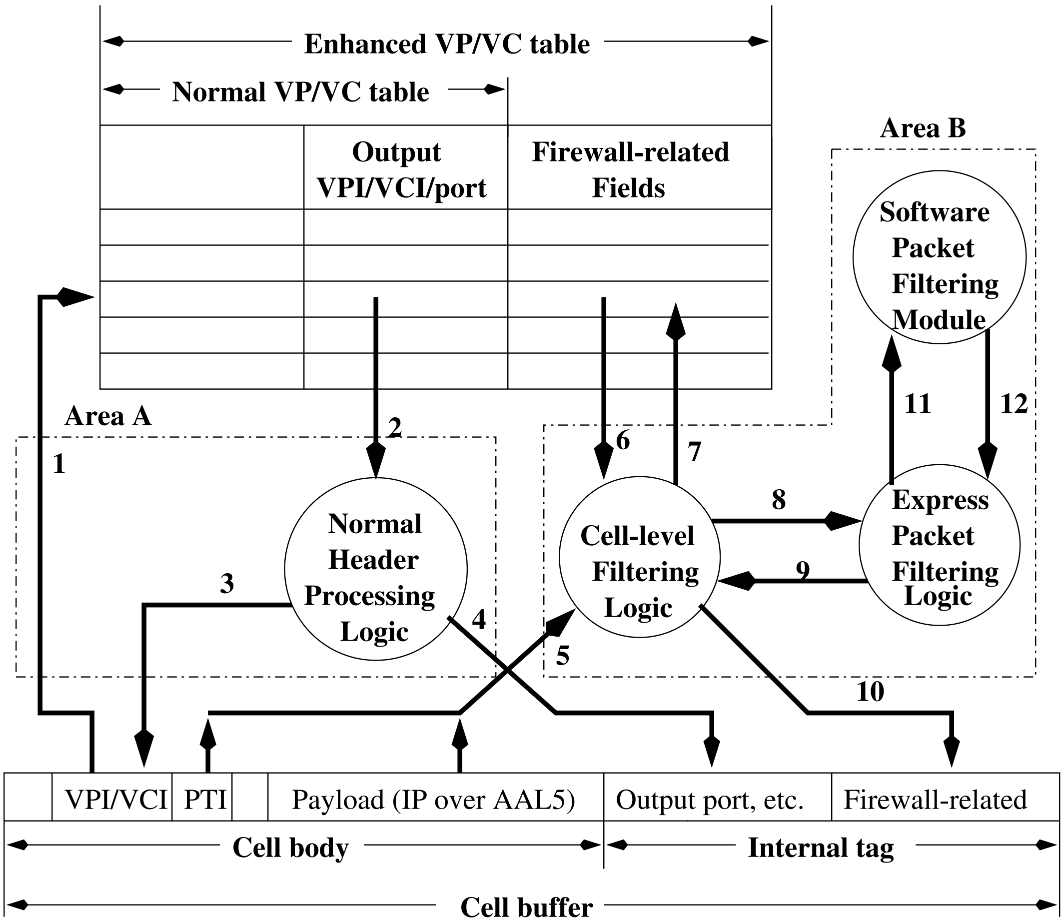

Figure 3: The header translation module in our firewall switch

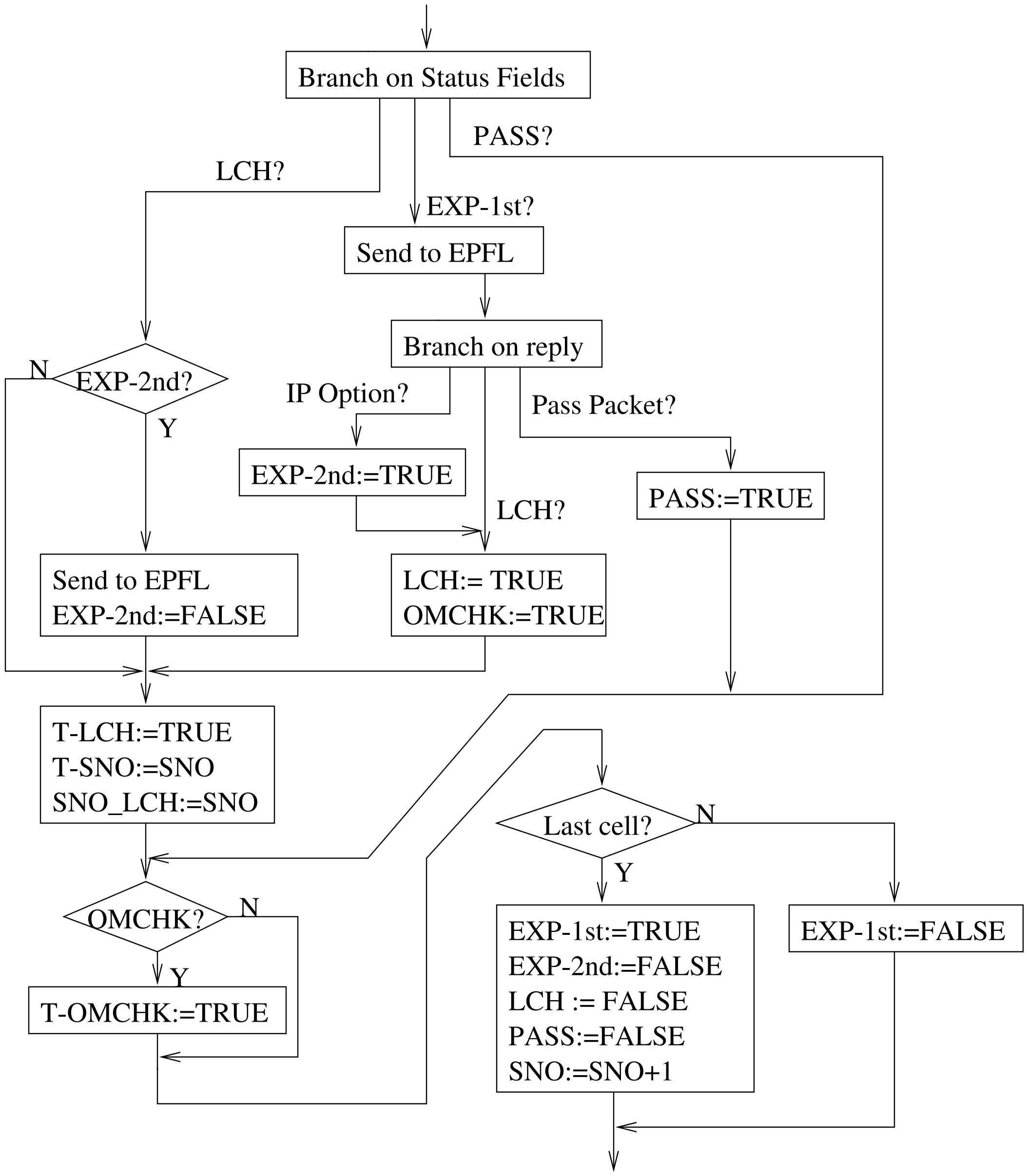

Figure 4: The flowchart of cell-level filtering logic

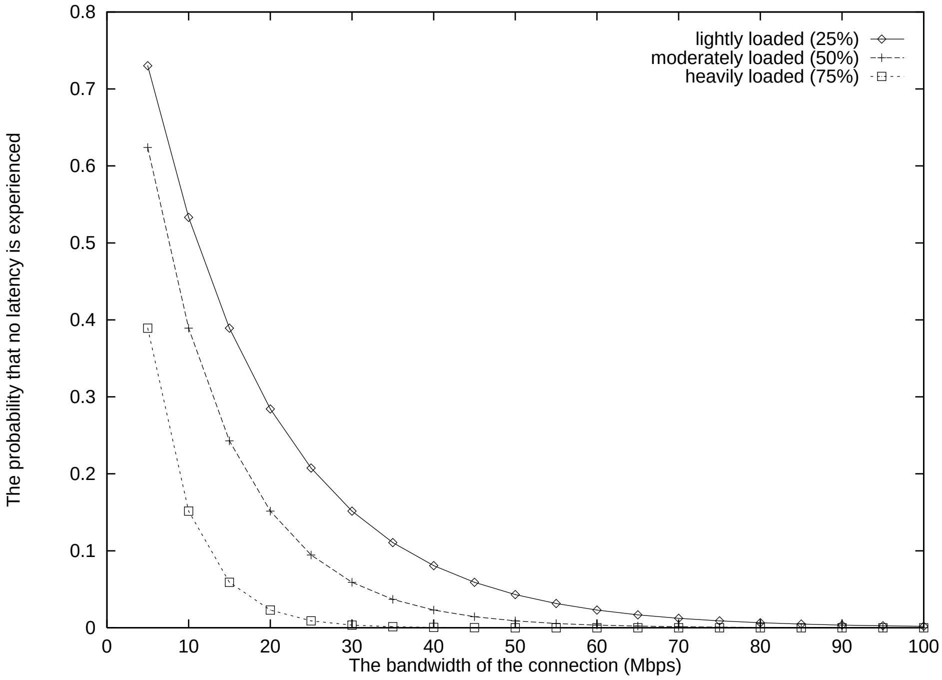

Figure 5: The probability that no latency is experienced by a 2-cell-long packet under different loads

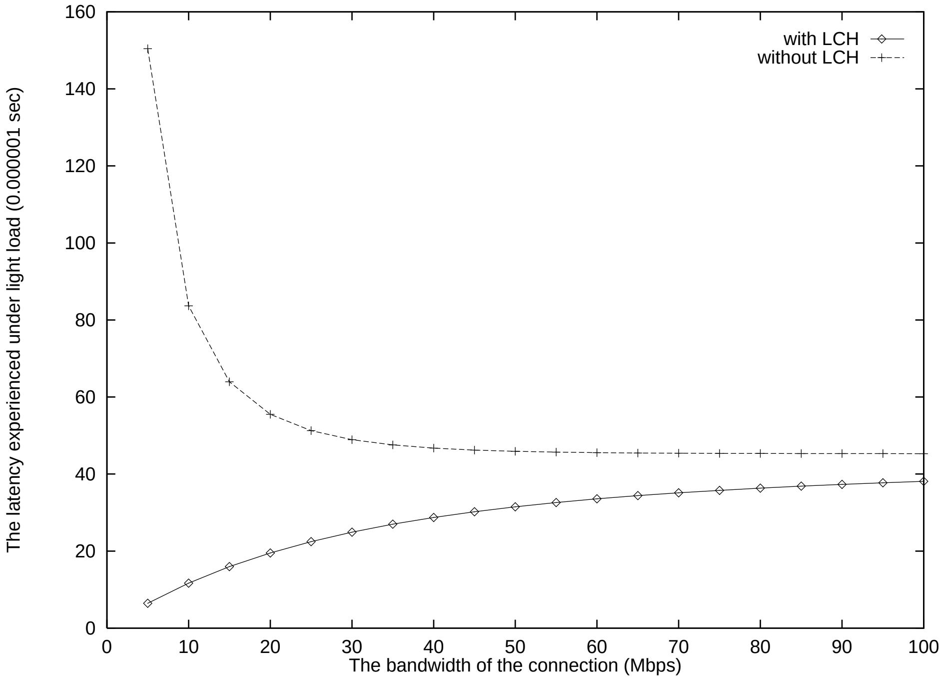

Figure 6: The latency experienced by a 2-cell-long packet under light load (25%)

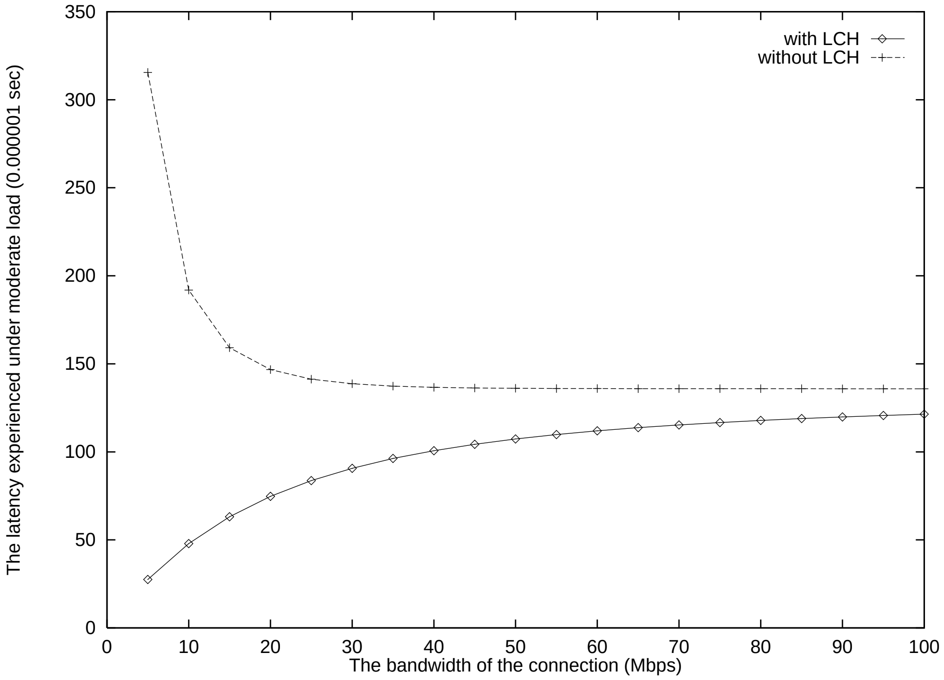

Figure 7: The latency experienced by a 2-cell-long packet under moderate (50%) load

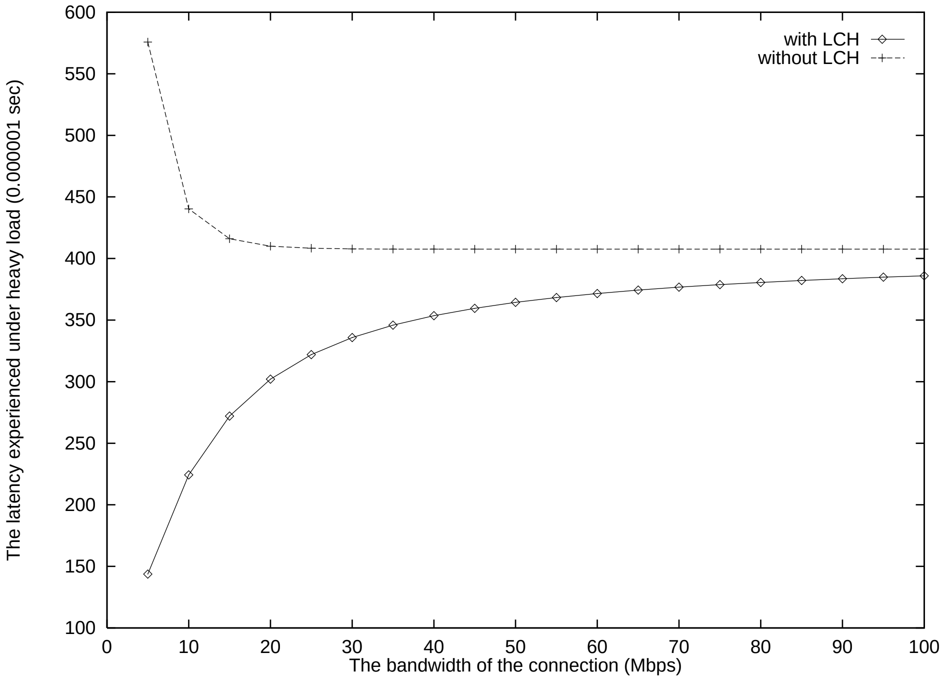

Figure 8: The latency experienced by a 2-cell-long packet under heavy (75%) load

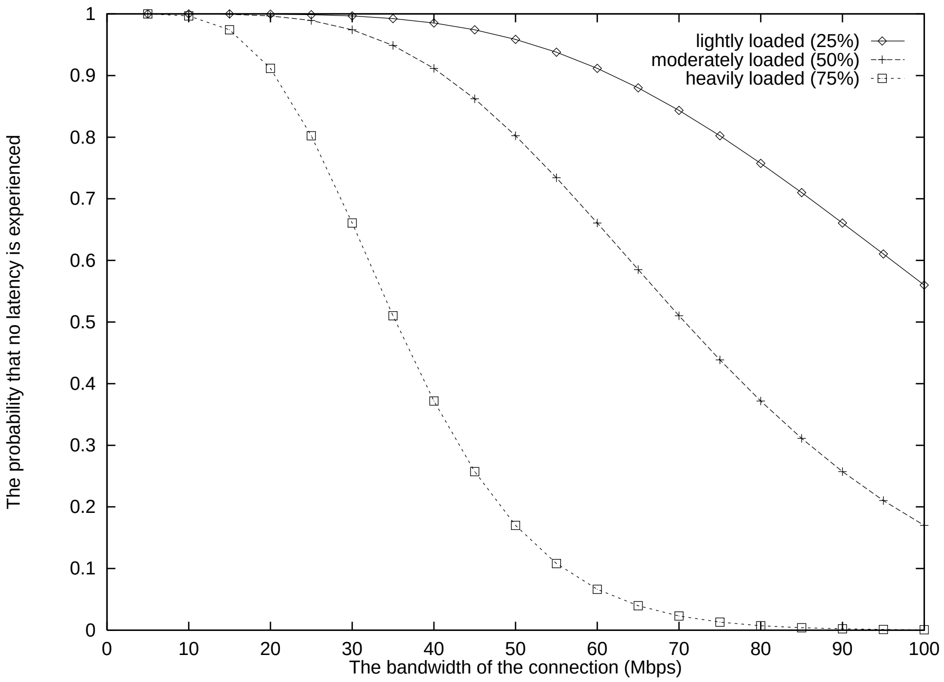

Figure 9: The probability that no latency is experienced by a 8-cell-long packet under different loads

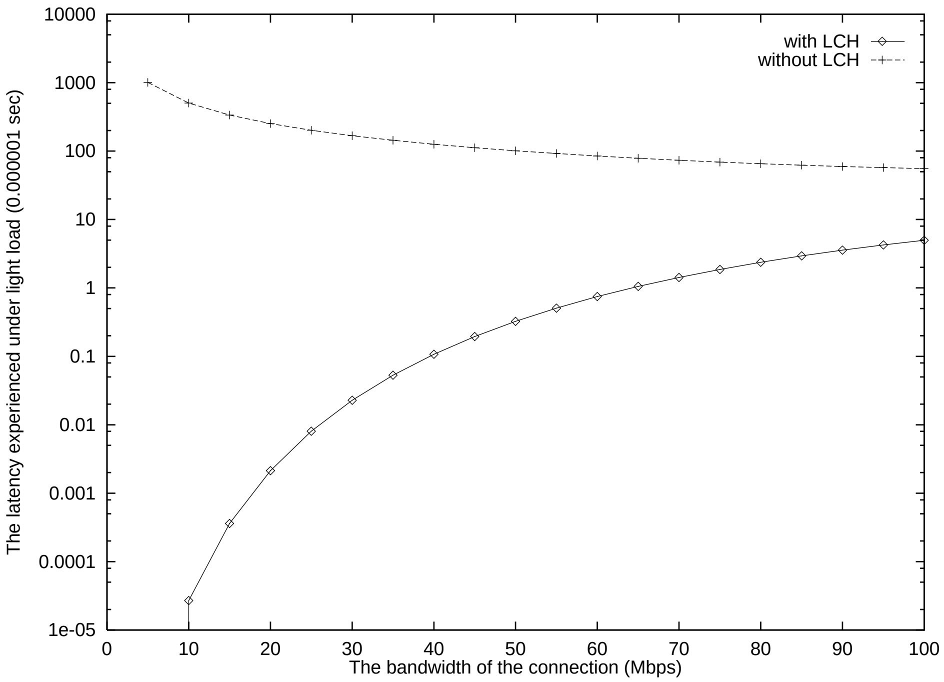

Figure 10: The latency experienced a 8-cell-long packet (the Y-axis is drawn on a logarithm scale) under light load (25%)

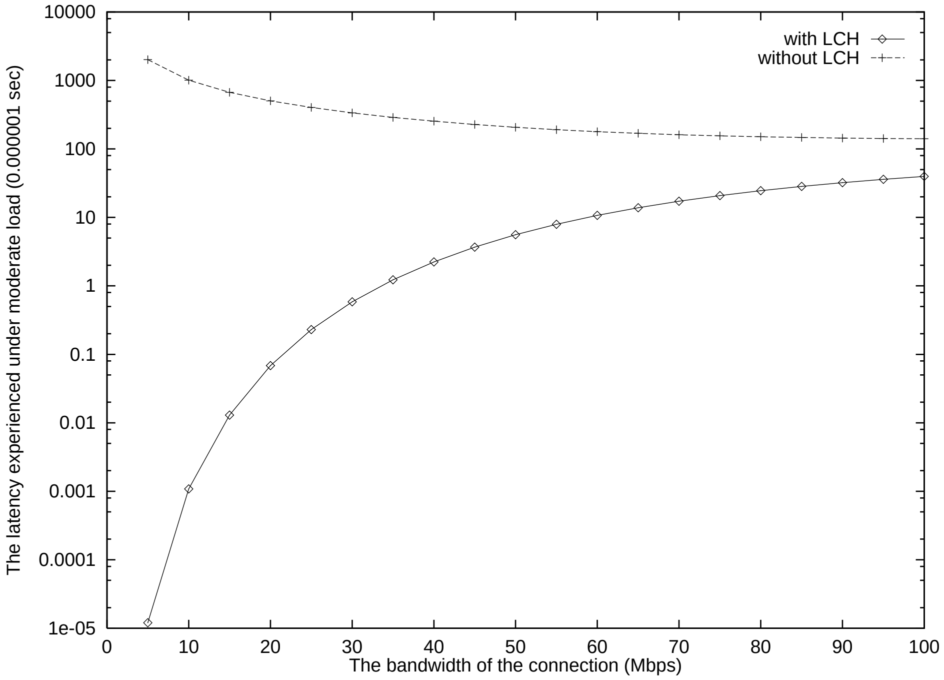

Figure 11: The latency experienced by a 8-cell-long packet (the Y-axis is drawn on a logarithm scale) under moderate load (50%)

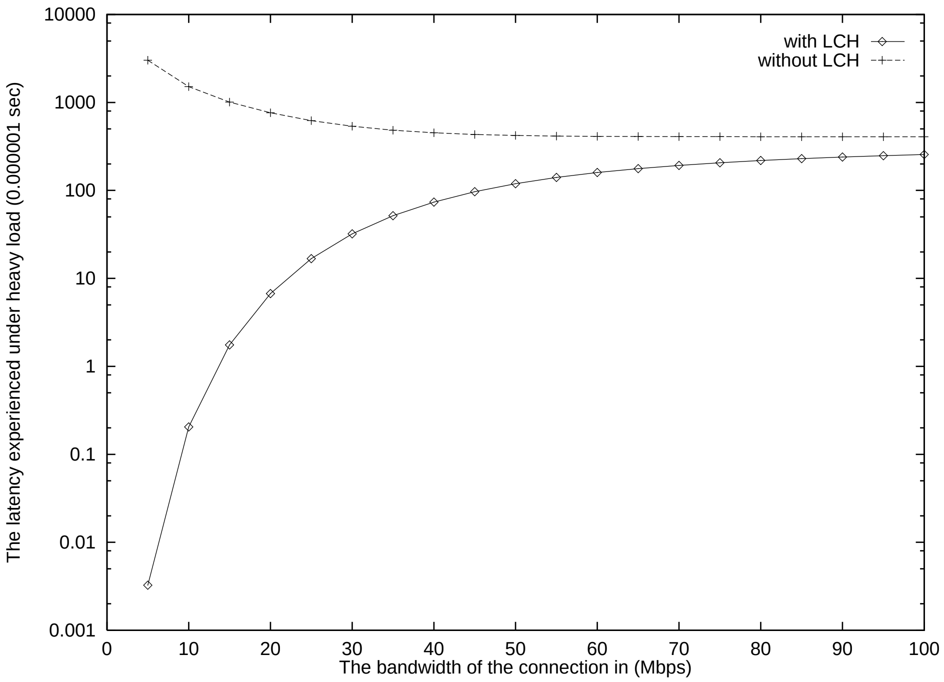

Figure 12: The latency experienced by a 8-cell-long packet (the Y-axis is drawn on a logarithm scale) under heavy load (75%)