Firewalls for ATM Networks

Uwe Ellermann Carsten Benecke
DFN-FWL[*]
University of Hamburg
{*Benecke,Ellermann*}*@fwl.dfn.de*

Abstract

There are many differences between ATM and todays most commonly used network technologies. New firewall architectures are required to exploit the advantages of ATM technology and to support the high throughput available in ATM networks.

This paper begins with a discussion of the impact of ATM on firewalls and then introduces the idea of parallelized firewalls, which may be used in order to achieve the high performance necessary for ATM networks.

1 Introduction

Firewalls are a widely used security mechanism in the Internet today. They are mostly used to provide access control and audit at the border between the public Internet and private networks, but are also used to secure critical subnets within private networks.

ATM is another somewhat newer trend in networking today. ATM provides a scalable high-speed network infrastructure, based on the concepts of fixed-length cells and virtual circuits. These conceptual differences to "legacy" networks[1] and the high throughput of ATM networks present both challenges and new opportunities for firewall concepts.

[*]This work was funded by the DFN-Verein (Association for the promotion of a German Research Network) and Deutsche Telekom under project number: DT10.

[1] In this article the term *"legacy" networks* is used for connectionless, shared medium based networks without resource reservation.

First published in: 'Proceedings: INFOSEC'COM 98', June 4th-5th, 1998, Paris, France

This paper discusses ATM specific topics of firewall design for ATM networks. General firewall issues such as security policies or implementation of firewalls are not discussed. Detailed discussions of these subjects can be found in [Chapman et al. 95, Cheswick et al. 94, Ellermann 94].

The following section gives a short introduction into ATM before discussing the consequences of using ATM in conjunction with firewalls. Different approaches to integrate packet screens into "Classical IP over ATM" networks are considered.

Section 3 presents performance measurements of the two most important firewall components: packet screens and proxy servers. It will be shown that the high processing requirements in both packet screens and proxy servers are the source of a severe throughput bottleneck of firewalls in ATM networks.

Parallel protocol processing is introduced in section 4; this is one promising solution to the need for increased firewall performance resulting from the high scalability of ATM networks. Several concepts for parallel firewalls are discussed.

The final section summarizes the results and closes with references to ongoing research.

2 ATM as a challenge for firewalls

Firewalls are widely deployed to protect critical subnetworks from public networks. While today firewalls are mostly used in networks not exceeding throughputs of 10 Mbit/s, most sites are currently upgrading to high-speed networks (HSN) like Fast-Ethernet, Gigabit-Ethernet or ATM. As firewalls are, by design, "choke-points", firewall performance is a major concern in HSNs. In addition to the performance requirements, ATM networks also introduce new networking concepts which require a revision of current firewall concepts. This section will discuss the differences between ATM and "legacy" networks and the resulting implications for firewalls. Performance measurements of firewalls in an ATM network are presented in section 3.

2.1 Overview of ATM

ATM is based on the concept of fixed-length cells and virtual circuits (VC). ATM cells are short compared to the variable-length of packets in "legacy" networks. Each cell has a payload of 48 bytes plus a header of 5 bytes. Unlike packets in "legacy" networks, ATM cells don't carry source or destination addresses in their headers. Instead a so called "virtual circuit" has to be established before any communication may occur between source and destination. Cells contain identifiers which allow cells to be associated with a virtual circuit. There are two different ways to set up a virtual circuit:

- "Permanent Virtual Circuits" (PVCs) are established by manual configuration in the end-systems and in all switches along the path. This solution is obviously restricted to rather small networks.
- "Switched Virtual Circuits" (SVCs) are initiated on demand.

The successful Internet and its protocols (IP, TCP, UDP etc.) will not be replaced by its ATM counterparts. Instead, ATM will be used as a fast medium to carry Internet protocols. ATM will primarily be used for building fast backbone networks. Only certain applications, such as videoconferencing, require advanced ATM features such as resource reservation.

Two widespread concepts are used to transmit IP traffic over ATM networks. "Classical IP and ARP over ATM" (CLIP) as defined by the IETF in RFC 1577 [Laubach 94] specifies an encapsulation format for IP datagrams and an ATMARP-server for the required mapping of IP addresses to ATM addresses. "LAN Emulation" (LANE) [ATM Forum 97] emulates "legacy" LANs and therefore supports not only IP but also other network layer protocols such as IPX. If only IP needs to be supported, CLIP provides better performance, as it introduces less overhead than LANE.

2.2 Implications of ATM on firewalls

ATM networks introduce four major challenges to firewalls:

Performance: Firewalls, as already stated, are a bottleneck by design. In order to increase security, all traffic is channeled through a small number of firewall systems. Current increase in workstation performance cannot cope with the easy scalability of ATM networks. 622 Mbit/s or even 1.2 Gbit/s can easily be achieved with ATM networks. Workstations cannot perform even simple filtering at these speeds.

In addition to the lack of firewalls to transfer legitimate traffic at high speeds, various attacks can be performed much more efficiently in high-speed networks. This is especially true for various "denial of service" attacks, such as SYN-flooding and ICMP attacks resulting in packet storms. Audit files created during an attack can easily grow by some megabytes within minutes, preventing the machine collecting further audit data after all the available audit data storage space has been filled.

New requirements: ATM has a number of features not available in "legacy" networks, most notably, ATM supports various "quality of service" requirements; a certain bandwidth or a fixed maximum delay during transmission can be specified individually for virtual connections in ATM networks. No currently available firewall supports resource reservation in order to keep track of these quality of service requirements. This is currently an active research area.

New risks: ATM networks require a number of new protocols (e.g. PNNI – "Private Network-Network Interface" and ILMI – "Integrated Local Management Interface"). Even more services are necessary to support CLIP or LANE. The security implications of these protocols are not fully understood. Before firewalls can be integrated into such an environment, the risks associated with these new protocols must be identified; this requires extensive research (see [Benecke et al. 98]).

Technical problems: As already described, ATM networks differ from "legacy" networks in many ways. Most firewall concepts have implied assumptions about the underlying network. Application layer firewalls (proxies) are on a high level of abstraction and are therefore more loosely coupled with the underlying network.

Packet screens, on the other hand, are usually based on the assumptions that every packet sent contains complete address information and that also the services accessed can be identified in every packet. Both assumptions are no longer valid in ATM networks. These aspects are discussed in the following section.

2.3 Integration of packet screens into ATM networks

Packet screens filter packets based on information in the packet headers. In "legacy" networks the address information available allows packet screens to restrict access to certain IP addresses and to TCP or UDP services.

As ATM cells contain only 5 bytes of header and 48 bytes of payload, a packet screen operating on every cell has only very limited information available. IP datagrams, usually a few hundred bytes long,[2] must be "segmented" into multiple cells by the sender and "reassembled" at the destination.

Classical packet screens: As packet screens operate on IP datagrams, they have to reassemble IP datagrams from cells before filters can be applied.

[2] A "Maximum Transfer Unit" (MTU) of 9180 bytes is defined for CLIP in [Laubach 94]. An alternative MTU of up to 64 kbytes may be negotiated for a virtual circuit.

Datagrams, which are allowed to be forwarded, must be segmented into cells once again after filtering. Segmentation and reassembly is performed by hardware on the ATM interfaces and therefore does not increase the packet screens processing load. After cells are reassembled to IP datagrams, the further processing of these IP datagrams does not differ from packet screens used in “legacy” networks. The performance of this solution is analyzed in section 3.1.

Cell screens: The reassembly and segmentation in classical packet screens increases the transmission time, as all cells must arrive before the original datagram can be recovered and filters can be applied. The delay can be reduced, if the packet screen could extract the information required for filtering from cells, thus avoiding reassembly.

To understand how this could be implemented, a short description of the transmission mechanisms for IP datagrams over ATM networks is required. The CLIP protocol stack is shown in figure 1. First, a SNAP header [Heinanen 93] is prepended to an IP datagram. The SNAP header identifies the transmitted payload as IP. SNAP header (8 bytes) and IP datagram are then encapsulated in an AAL-5 frame (see [Peterson et al. 96]). The AAL-5 frame has a trailer of 8 bytes. It also contains a variable number of padding bytes to match the frame exactly into multiple 48 bytes cells. This AAL-5 frame is segmented into cells, where the last cell of the AAL-5 frame is marked.[3] All cells are then sent on the same virtual circuit across the ATM network. ATM guarantees the ordered delivery of cells.

A cell screen can identify the last cell of an AAL-5 frame. As all cells are delivered in order, the next cell will be the first cell of the next datagram. This first cell contains 8 bytes of SNAP header, 20 bytes of IP header and 20 bytes of TCP header. With the complete IP and TCP (alternatively UDP[4]) headers in the first cell, the packet screen has all information that is required for filtering. If the forwarding of the datagram is allowed by the filtering rules, the first cell and all following cells on the same virtual circuit are forwarded until the last cell of an AAL-5 frame is found. If the datagram must be blocked, the packet screen discards all cells up to and including the last cell of the AAL-5 frame.

A cell screen can be implemented mainly in hardware and installed between an external link and an internal switch.[5] But, as most parts of a cell screen

[3] The last cell of an AAL-5 frame is marked by setting a bit in the “payload type” field of the cell header.

[4] The UDP header (8 bytes) is shorter than the TCP header, so with UDP there is even room for 12 bytes of payload in the first cell.

[5] “StorageTek Network Systems” markets a product called “ATLAS” which implements this concept.

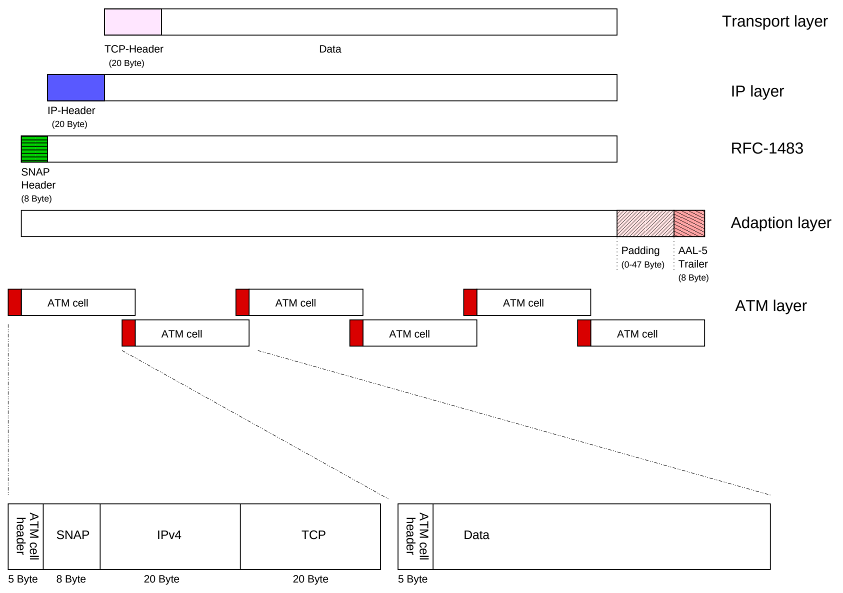

Figure 1: Cell Screen: CLIP protocol stack

are already required in ATM switches (forwarding of cells, recognition of the end of an AAL-5 frame and selective discard of cells[6]), the extension to support the missing cells screen features is a natural one.

A cell screen imposes a shorter delay, as screening can occur after the first cell has been received. The copy operations performed by classical packet screens, which move whole IP datagrams are also avoided; cell screens only have to copy the first cell for screening, subsequent cells can be forwarded or dropped efficiently by the switching hardware. The screening overhead for evaluation of the filter rules is, however, the same for cell screens and classical packet screens.

Signaling Screens: Most firewall concepts rely on a combination of one or more packet screens and one or more bastion hosts. The bastion hosts perform connection authentication on an application layer level. The packet screens function is to allow the proxy servers to communicate, while preventing all other communication. If another mechanism is available, which ensures that only this legitimate communication can take place, no packet screen is needed. For example a gateway firewall does not require a packet screen, as it is the only machine that is connected to both internal and external networks.

[6] The mechanism to discard all cells till the end of an AAL-5 frame is already implemented into switches for discarding useless cells after loss of a cell.

In ATM networks routing and forwarding are separate tasks.[7] All routing decisions are made during the setup of a virtual circuit. All data sent is forwarded along this virtual circuit. The end systems of a virtual circuit (sender and receiver) can be identified during connection setup before any data is sent. By specifying rules which define which circuits may be setup between which end systems, all traffic can be forced to be processed by a bastion host before it enters a network on the other side of the firewall.

Current ATM switches already support a simple filtering language; its structure is similar to the filter rules of packet screens in routers. Rules can be defined in an ATM switch to expressively allow or deny the establishment of virtual circuits to a list of ATM addresses.[8] It requires only moderate effort to define rules, that forbid the establishment of virtual circuits between internal and external end systems except for the bastion host (see [Benecke et al. 98]). The filter rules only have to be examined during the setup of a new virtual circuit, there is no impact on the performance of the following communication. Obviously the bastion host must be powerful enough to support the high bandwidth available or the traffic must be distributed among several parallel bastion hosts as discussed in section 4.2.

3 Performance of firewalls in ATM networks

An ATM test-network was setup for performance measurements of different firewall concepts in high-speed networks.[9] The following discussion summarizes the results of performance measurements for the two most important firewall components – packet screens and proxy servers. A detailed analysis of firewall performance in ATM networks can be found in [Ellermann et al. 98].

3.1 Performance of packet screens

A workstation equipped with two ATM interfaces was used as a packet screen for the performance measurements. The software “IP-filter” (version 3.2)[10]

[7] In IP networks routing decisions are made for every IP datagram based on the destination IP address contained in the header of each datagram. Following IP datagrams may travel different paths across the network.

[8] IP addresses, TCP or UDP port numbers are not available during the setup of a virtual circuit. For that reason signaling screens cannot filter on IP addresses used or ports accessed.

[9] The network consists of six Sun Ultra Sparc 1/140 (Solaris 2.6) equipped with Sun 155 Mbit/s ATM interfaces (SunATM 2.1) and two Cisco Lightstream 1010 ATM switches.

[10] `ftp://coombs.anu.edu.au/pub/net/ip-filter/`

used on the packet screen allows the specification of filter rules. The tool "Netperf"[11] repeats `write` calls on an already opened TCP connection for 10 seconds. The throughput is calculated by the amount of data transferred.

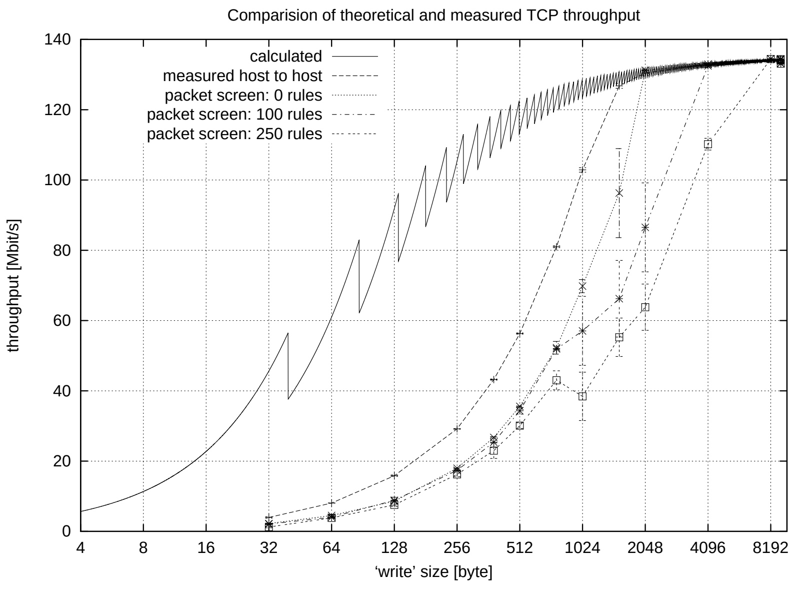

Figure 2: Throughput over a packet screen

The figure 2 shows the achieved throughputs for three selected filter configurations with 0, 100 and 250 rules. As expected the performance depends primarily on the number of filter rules configured. The calculated theoretical maximum throughput can only be reached with `write` calls longer than 2048 bytes. The reason for the sharp drop of throughput for shorter `write` calls is the limited packet throughput of the packet screen. In an OC-3c ATM network (155 Mbit/s) almost 180,000 datagrams per second are necessary in order to achieve the calculated theoretical maximum throughput with message sizes below 40 bytes. While the workstations in our environment were able to generate about 16,000 datagrams per second, the tested packet screen reaches only about 8.000 datagrams per second. This value will be further reduced by adding more filter rules. As typical message sizes rarely exceed 500 bytes the expected throughput of the packet screen in a real environment will be limited to 30-40 Mbit/s. In order to reach the calculated theoretical maximum throughput of 120 Mbit/s for this message size, the packet throughput of the packet screen must be four times higher. A packet throughput of approximately 30,000 datagrams per second is necessary to reach 120 Mbit/s with a message size of 500 bytes. These results show that the actual data throughput is not a bottleneck, the packet throughput of the packet screen limits the maximum throughput instead.

[11] `ftp://ftp.cup.hp.com/dist/networking/benchmarks/netperf/`

3.2 Performance of proxy servers

Proxy servers control connections at application level. The processing of the transferred data by an application process obviously requires more resources than a check of datagrams at a packet screen. Nevertheless most firewall concepts are based on proxy servers as better security can be achieved by doing access control on the application level.

Despite the higher processing overhead it is also possible with proxy servers to achieve a maximum throughput of 134 Mbit/s. But this throughput can only be achieved for transfers of large quantities of data in large datagrams. For more important smaller quantities of data, for instance the transfer of a HTML page, the connection establishment time dominates the time required for transferring the data. The connection establishment time to a server via a standard proxy server in a LAN environment was measured to be about 0.03 seconds. The time required to transfer a message of 16 kbytes is magnitudes lower. For that reason the time to open a connection and transfer a message of 16 or 32 kbytes will take about 0.03 seconds, regardless of the length of the message.

The data throughput and the increase in connection establishment time designate a more user-oriented view on proxy server performance. The number of parallel connections is however just as important. First results show that dependent on the type of proxy server less than 100 active connections can be processed at the same time on a proxy server. The actual number of parallel connections experienced in high-speed networks can be much higher. Also the rising complexity of proxy servers (integration of virus scanner, encryption etc.) will require a distribution among several bastion hosts.

4 Concepts for Parallel Firewalls

The throughput measurements for packet screens and proxy servers have shown that the performance of these classical firewall concepts is not sufficient for high-speed networks. The packet throughput of workstation-based packet screens is too low to result in an adequate throughput for typical IP datagram sizes. The high processing overhead of application layer firewalls such as proxy servers result in low maximum throughputs, so that proxy servers are perceived as a bottleneck for communication. In the following section we start with an overview of parallel protocol processing in order to introduce parallel firewall concepts later on.

4.1 Parallel Protocol Processing

It turned out that a typical workstation is unable to provide the available throughput of a high-speed network at transport level or application level due to a bottleneck in the protocol processing in higher layer protocols [Zitterbart 91] such as TCP or IP. For this reason many parallel processing approaches have been suggested [Woodside 91].

While the *static methods* all introduce some kind of pipelining in the protocol stack which differ in the achievable granularity of parallel processing, the *dynamic methods* handle either incoming packets (*packet parallelism* [Goldberg 93]) or whole connections (*connection parallelism*). For a discussion of advantages and disadvantages of these methods see [Benecke 96].

While packet parallelism fits very well for parallel packet screens (see section 4.3) the connection parallelism is the better choice for concepts for parallel application level firewalls.

In the following sections we will discuss how dynamic methods may be used to improve the performance of firewalls in high-speed networks. Examples will be used to discuss the advantages and disadvantages of different approaches.

4.2 Parallel Bastion Hosts

As most proxy servers support TCP based services and TCP is a connection-oriented protocol the connection parallelism is a straightforward choice for parallel application level firewalls. The load that has to be distributed among parallel processes is the accumulated number of parallel connections a proxy server has to handle. We will now discuss different basic approaches for distributing the load. This will lead to parallel application level firewalls.[12]

4.2.1 Static Distribution of Connections

The "load" denotes the number of open connections to a proxy server. If the load has to be shared we need solutions for *distributing these connections.*

The easiest way to distribute the load is to provide a separate proxy server for each service (e.g. HTTP, FTP,...) that has to be supported. As traffic is *statically mapped* to dedicated proxy servers, measurements have to show which proxy servers can be mapped together on a single processor (e.g. bastion

[12] In the discussion of application level firewalls *bastion hosts* are used as an example. In most cases these concepts are also applicable to gateway firewalls.

host) and which proxy servers should be mapped onto separate processors or hosts.

By distributing the proxy servers among different hosts the security can also be improved. If an intruder succeeds in attacking one bastion host, he still has no access to other proxy servers. If on the other hand all proxy servers are concentrated on a single bastion host, all these services can be used by an intruder who succeeds in attacking this bastion host to proceed attacking the guarded net. The major disadvantage of this solution is the static mapping of all connections to a certain service to a dedicated proxy server on a dedicated processor. If the current traffic differs from the expected traffic (e.g. more FTP requests than HTTP requests) the forejudged mapping may be inefficient. This may lead to situation where a single bastion host is under heavy load while other parallel bastion hosts are idle.

4.2.2 Dynamic Distribution of Connections

The throughput can be improved by replicating a proxy server on multiple processors, so that connections can be *dynamically mapped* to replicated proxy servers.

Example: Round-Robin DNS

A well known example for dynamically distributed connections is a “Round Robin” extension to DNS [Brisco 95]. All names of replicated WWW servers which shall share the connections are registered with a `CNAME` for the virtual “WWW” server. After each lookup the DNS server rotates the list of `CNAME`s. The next client requesting the name of the “WWW” server will receive a different answer from the DNS server and the connections to the “WWW” server will be distributed among the parallel servers.

We have a dynamic distribution of connections, but we still can not make sure that this load is balanced as the distributing process does not get any feedback about the load of the parallel proxy servers.

Example: Distribution by “meta” Proxy

The distribution of connections to the proxy servers can be improved by a “meta” proxy. All requests are sent to this “meta” proxy which chooses one of the parallel proxy servers to process the request. As the parallel proxy servers may send status information to the meta proxy this choice can be made load dependent (e.g. the proxy server with the lowest load gets the request).

The main problem of the dynamic distribution is to find an inexpensive (fast) algorithm that distributes the incoming connections among the parallel bastion host. This distribution can either be *centralized* or *decentralized.*

4.2.3 Centralized vs. Decentralized Distribution

A *centralized distribution* may be realized by a single (meta) proxy which distributes all incoming connections among the pool of proxy servers which actually serve the requests. The advantage of this solution is that the meta proxy may gather status and load statistics from the proxy servers that enables a fair and balanced distribution of incoming connections. The meta proxy may also be able to redirect requests if it detects an intrusion or failure of a bastion host. On the other hand the meta proxy has to handle all incoming connections. It must be fast enough so that the distribution of connections is not a bottleneck itself.

Another way to improve the throughput of centralized distribution is to make the decision in the kernel (e.g. on the network (IP) layer). The low throughput of proxy servers results from the fact that the proxy servers are application level processes which receive the request via one connection and forward it via another one. If we just want to distribute the incoming connections a kernel level process could forward the datagrams to the bastion hosts. This mechanism is a special case of “Network Address Translation” (NAT).

Example: Packet Screen as a Central Distributer

An example for a centralized distribution at the network level is a combination of packet screen and parallel bastion hosts.

Modern packet screens are able to map one IP address onto another (“Network Address Translation” (NAT)) while they are filtering datagrams. This can be used to distribute the datagrams to parallel bastion hosts depending on the load of the proxy servers (see figure 3).

Measurements for a packet screen with enabled NAT show that the performance impact of NAT is almost equal to the impact of 10-20 filter rules. Depending on the type of traffic and the processing overhead of the proxy servers there is a risk that the packet screen used for distribution may become the bottleneck in this setup.

There are, however, some points to be considered. First all datagrams of one connection have to be forwarded to the same proxy server. Secondly there may also exist inter-connection dependencies. For example the data stream and the control stream of an FTP session should be mapped to the same

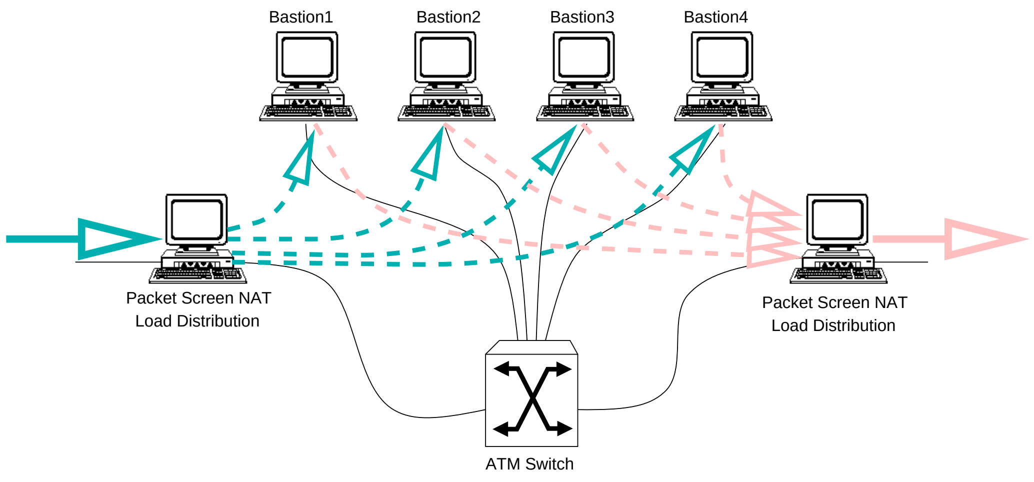

Figure 3: Distribution by “Network Address Translation” (NAT)

proxy server. This may be implemented by forwarding all connections of a client (represented by its IP address) to the same proxy server. On the other hand mapping all incoming connections of the same client onto the same proxy server may be too restrictive in some environments. For each kind of proxy server the context information that have to be shared in order to resolve inter-connection dependencies must be specified. The propagation of context information enables a higher degree of parallelism among the proxy servers.

Distribution over “native” ATM

The previous setup (figure 3) may be improved by using native ATM connections between the packet screens and the parallel bastion hosts. All parallel bastion hosts respond to the same IP address. The packet screen does not need to use time consuming NAT transformation on every IP datagram, it acts as a router and simply forwards an IP datagram over one of the native ATM connections to a bastion host instead. The route for an IP datagram cannot be a function of the destination IP address because the IP addresses of the bastion hosts are all the same. The path of a datagram is calculated by the load distribution algorithm. The use of the same IP address for multiple systems usually has disastrous effects on the network. These problems do not occur in the described setup as the bastions can only communicate through the packet screen. Further studies have to show quantitative aspects of performance improvements of this approach.

A *decentralized distribution* may be realized by the parallel proxy servers themselves. Each proxy server inspects all incoming connections and decides whether it is responsible for this connection.

Example: Distributed connection response

The decentralized algorithm to decide whether or not to serve a connection may be realized as a function of the first TCP segment (indicated by the SYN flag). This method requires that every packet can be received by all proxy servers similar to the concept of the parallel packet screen (see section 4.3). New connections can be distributed either randomly, load dependent, or dependent on information inside the datagram.

Another example for a decentralized distribution depends on the cooperation between client and server:

Example: Redirecting HTTP requests

The HTTP protocol [Fielding et al. 97] enables servers to redirect requests by sending the clients an alternative URL. This feature can be used by parallel servers to balance their load. Whenever a server under heavy load receives a request it may decide to redirect the request to a replicated server. An obvious problem with this approach is a spoofing of redirect messages which increases the risk of "man in middle" attacks.

Example: Transparent redirecting connections

Another disadvantage of the solution for HTTP is the need for an explicit cooperation between client and server. This cooperation may be hidden by providing a transparent `connect()` system call in a shared library that replaces the standard library function. The new `connect()` system call tries to open a connection. A server may accept the connection or supply the address of an alternative server. As this redirection is hidden by the `connect()` call, there is no need to change the client software. Another advantage is that it is a generic solution. It works for all TCP based applications that use the library replacement. Note that this solution can be build on the simple protocol used by SOCKS [Leech et al. 96].

As a redirection of a connection increases the connection setup time, the tradeoff between increased throughput and connection setup time has to be taken into account. A redirection is usually only worth the increased setup time, if a large amount of data has to be transferred. For small quantities of data it is more efficient to process the connection without redirection and notify the client to use an alternate proxy server for subsequent connections. A prototype that uses the described transparent redirection for a distributed load balancing is currently developed.

4.3 Parallel packet screening

Measurements for the performance of packet screens (figure 3.1) have shown that a typical workstation is able to perform screening in an 155 Mbit/s ATM network for large packet sizes only. Unfortunately the average packet size in the Internet is much smaller. The packet throughput of the investigated packet screen has to be increased about four times to reach acceptable throughputs with smaller packet sizes.

Parallel packet screens based on the paradigm of “packet parallelism” (see section 4.1) provide a scalable solution for high-speed networks. “Packet parallelism” fits very well for parallel packet screens that do not care about connection contexts. As there is no need to update any connection contexts, any packet of any connection may be screened in parallel. Of course this is the case for connectionless (UDP) traffic anyway.

A distributed decision about which packet screen is responsible for the filtering has to be made. The additional costs for this decision must be very low compared to the total costs of filtering. Every packet screen in the parallel setup inspects every packet and immediately discards packets that another screen is responsible for. The decision which packet screen is responsible for a packet may be a function of information elements in the packets. For example the hash value of the IP checksum may be used as an index to the packet screen that has to examine the packet. All other packet screens may discard the packet. Because the IP checksum is likely to differ for successive packets this algorithm should assure an almost balanced distribution.

Example: Broadcast LAN implementation

The implementation is very simple for broadcast LANs such as Fast-Ethernet, Gigabit-Ethernet, or FDDI. In the case of Fast-Ethernet the parallel packet screens can be placed between two Hubs[13]. The Hubs ensure that all packets are distributed to all parallel packet screens (figure 4).

Example: ATM Implementation

As ATM networks are connection-oriented the required broadcast functionality must be emulated. An ATM switch can be used for a very efficient implementation. It is possible to configure a point-to-multipoint connection so that the switch copies all incoming cells to multiple outgoing virtual channels. This mechanism can be used to assure that all packet screens in an ATM network receive the incoming packets.

[13] A Hub is a multi-port repeater that replaces the shared medium coax wire.

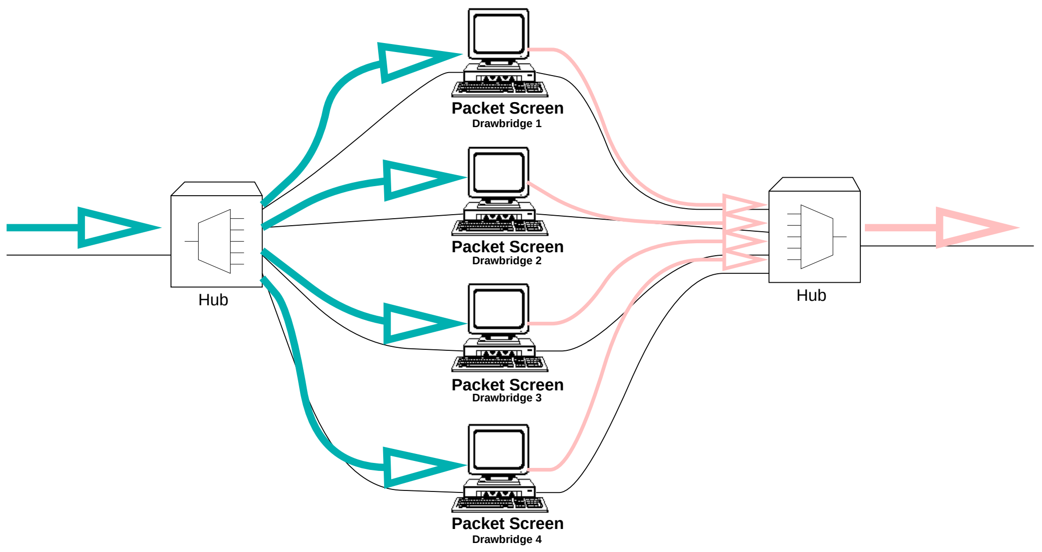

Figure 4: Parallel Packet Screen (Fast-Ethernet/Gigabit-Ethernet)

A problem arises by the increased possibility of failure due to the parallel packet screens. By monitoring or status propagation one should make sure that a failure is detectable so that the distributed filtering algorithm may be adjusted to the new number of parallel packet screens. On the other hand with failure detection there is no longer a single point of failure. If one of the packet screens breaks down the others take over.

5 Conclusions

This paper discussed the impact of ATM on firewalls. The functional differences between ATM and "legacy" networks means that classical firewall concepts (for "legacy" networks) cannot be applied to ATM technology without modification. These differences have a greater impact on packet screens than application level firewalls, as they are more dependent on the underlying network than application level firewalls. Three different approaches for packet screens were introduced: classical packet screens, cells screens and signaling screens.

The highly scalable ATM technology raises throughput problems for both packet screens and proxy servers. This emphasizes the necessity for the parallel firewall concepts we have introduced, which overcome the throughput bottleneck. Parallel firewalls can provide scalable solutions for upcoming high-speed networks.

Prototypes of parallel firewalls will be developed in further research. Other important aspects which will have strong functional and performance impact on firewalls are the development of “native” ATM firewalls and the integration of cryptographic mechanisms into firewall concepts.

References

[ATM Forum 97] John D. Keene, Editor. “LAN Emulation Over ATM Version 2 – LUNI Specification”. *ATM Forum* (AF-LANE-0084.000), July 1997.

[Benecke 96] Carsten Benecke. “Entwurf und Realisierung eines parallelen Analysewerkzeuges für Ethernet-basierte Netze”. *Universität Hamburg*, February 1996.

[Benecke et al. 98] Carsten Benecke, Uwe Ellermann. “Securing ‘Classical IP over ATM Networks’”. *Proceedings of 7th USENIX Security Symposium, January 26th-29th, 1998, San Antonio, Texas*, January 1998. (`http://www.fwl.dfn.de/eng/team/cb/eng_natm/`)

[Brisco 95] Thomas P. Brisco. “DNS Support for Load Balancing”. *RFC 1794*, April 1995. (`ftp://ds.internic.net/rfc/rfc1794.txt`)

[Chapman et al. 95] D. Brent Chapman, Elizabeth D. Zwicky. “Building Internet Firewalls”. O’Reilly & Associates, September 1995.

[Cheswick et al. 94] William R. Cheswick, Steven M. Bellovin. “Firewalls and Internet Security: Repelling the wily Hacker”. Addison-Wesley, 1994.

[Ellermann 94] Uwe Ellermann. “Firewalls: Isolations und Audittechniken zum Schutz von lokalen Computer-Netzen”. *DFN-Bericht Nr. 76*, September 1994.

[Ellermann et al. 98] Uwe Ellermann, Carsten Benecke. “Firewalls für Hochgeschwindigkeitsnetze”. *Deutscher Internet Kongress 1998*, May 5th-6th, 1998, Frankfurt, 1998.
(`http://www.fwl.dfn.de/team/ue/fw/fire-hsn/`)

[Fielding et al. 97] Roy T. Fielding, Jim Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Tim Berners-Lee. “Hypertext Transfer Protocol – HTTP/1.1”. *RFC 2068*, January 1997.
(`ftp://ds.internic.net/rfc/rfc2068.txt`)

[Goldberg 93] Goldberg, M.W. and Neufeld, G.W. and Ito, M.R. “A Parallel Approach to OSI Connection-Oriented Protocols”. *In: Pehrson, B. and Gunningberg, P. and Pink, S. (eds.) Protocols for High Speed Networks, III, pp. 219–232*, North-Holland 1993

[Heinanen 93] Juha Heinanen. "Multiprotocol Encapsulation over ATM Adaptation Layer 5". *RFC 1483*, July 1993.
(ftp://ds.internic.net/rfc/rfc1483.txt)

[Laubach 94] Mark Laubach. "Classical IP and ARP over ATM". *RFC 1577*, January 1994. (ftp://ds.internic.net/rfc/rfc1577.txt)

[Leech et al. 96] Marcus Leech, Matt Ganis, Ying-Da Lee, Ron Kuris, David Koblas, LaMont Jones. "SOCKS Protocol Version 5". *RFC 1928*, March 1996. (ftp://ds.internic.net/rfc/rfc1928.txt)

[Peterson et al. 96] Larry L. Peterson, Bruce S. Davie. "Computer Networks: A Systems Approach". Morgan Kaufman Publishers, 1996.

[Woodside 91] Woodside, C.M. and Franks, G. "Alternative Software Architectures for Parallel Protocol Execution with Synchronous IPC". *IEEE/ACM Transactions on Networking, Vol. 1, No. 2, pp. 178–186*, April 1993

[Zitterbart 91] Zitterbart, M. "Funktionale Parallelität in transportorientierten Kommunikationsprotokollen". *Fortschritt-Berichte, VDI Reihe 10, Nummer 183*, VDI-Verlag 1991