A Distributed Firewall for Multimedia Applications

Utz Roedig[1], Ralf Ackermann[1], Christoph Rensing[1], Ralf Steinmetz[1,2]

[1] Darmstadt University of Technology, Merckstr. 25, D-64283 Darmstadt, Germany
[2] GMD IPSI, Dolivostr. 15, 64293 Darmstadt, Germany
{Utz.Roedig, Ralf.Ackermann, Christoph.Rensing, Ralf.Steinmetz}
@KOM.tu-darmstadt.de

Abstract. Firewalls are a widely used security mechanism to provide access control and auditing at the border between "open" and private networks or administrative domains. As part of the network infrastructure they are strongly affected by the development and deployment of new communication paradigms and applications.Currently we experience a very fast rise in the use of multimedia applications. These differ in many aspects from "traditional applications", for example concerning bandwidth usage, dynamic protocol elements or multiple data flows for one application session. Corresponding firewall mechanisms and techniques did not change with the same dynamics though. Currently existing firewalls have problems supporting these new type of applications because to some extent they try to map the new characteristics to the manner of conventional applications which they are able to handle. We strongly believe that new application types require new firewall techniques and mechanisms. In this paper, we identify typical characteristics of multimedia applications that cause problems using traditional firewalls. Based on this analysis we deduce enhancements to existing firewalls that can be used to better adapt to a communication environment in which multimedia applications are used. We describe these enhancements in general, show a adequate systems architecture and present a implementation based on this design. The feasibility of that approach has been shown in the example scenario that we finally present.

1 Introduction

Today, security aspects have become more and more important and access control at the network border is considered essential. Therefore, most organizations replaced their basic internet routers by devices that perform additional packet-filtering. This option is cost effective, because most routers are able to perform packet-filtering tasks anyway and the functionality has just to be activated. For some institutions, however, this level of security may be insufficient. Packet filters usually just do a pattern matching on predefined fields within the packet (header) but do not pay attention on the semantics of the packets payload which represents application specific data. A more sophisticated method that can be used to control the communication over a network border are so called "stateful filters". They act like a filter, but they are also able to extract information from the application layer and most significant - may change their behavior according to what passed through them in advance.

For higher security users must consider additional options and have to augment packet filter security with proxying. A proxy server offers additional security because the session flow is retained, inspected and forwarded at the application layer. Many firewall vendors prefer stateful filters instead of proxies because that way it is easier to implement support for new protocols. Furthermore filters generally allow for better performance. However, there are strong claims that stateful filters are less secure than proxies [1]. To combine the advantages of all these firewall techniques [2] a mix of packet filters, stateful filters and proxies is often utilized. We call the combination of these elements as shown in Figure 1 *firewall system*.

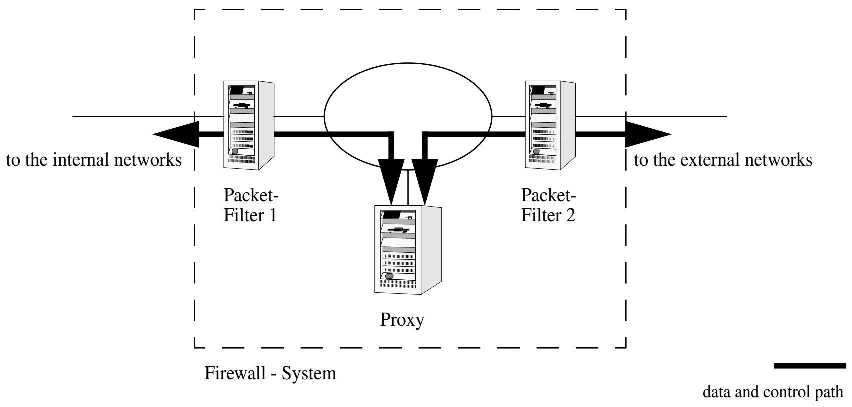

Figure 1: Firewall System

In this scenario, communication between the internal and the external network is only possible by passing data through both filters and the proxy. To enforce the hosts to not communicate directly but only via the proxy, the filters are configured in a way that only packets sent from or to the proxy server are forwarded. Applications that access a server in the other network have to be configured to use the proxy server either explicitly or by means of an element doing address or port translation. The area between the two filters is usually called *demilitarized zone* (DMZ) [3]. The described structure is often used for firewalls and is in general considered to provide a secure configuration to protect the internal network. Therefore, we use it as a reference scenario.

Under certain circumstances, however, the protection is consciously weakened - hazarding obvious security risks. This is usually done if such a practice seems to be the only applicable short-term way to use a certain service at all. Some non-representative examples are given below:

- Some applications complicate the use of a proxy since they were designed and implemented with end systems communicating directly in mind. So they do for instance transmit lower layer addresses as part of their application specific payload data again. In this case, the firewall filters may be configured to selectively pass data streams directly to the external net and vice versa. The same is necessary if no proxy is implemented (yet) for an application.
- Most firewall components that are used today are not designed to support multicast communication. To use applications which demand multicast the security policy of

the firewall must usually be weakened on a “all-or-nothing” basis. This conflicts with the intended policy to handle data streams at a firewall individually and with a fine-grained granularity. Approaches are currently made to remedy that drawback [4].

In this paper we deal with the ever increasing class of multimedia applications which lead to serious problems. For many of these applications, the firewall system has to be configured in a way that it does not provide the maximum possible protection for the internal network. In section 2 we describe characteristics of multimedia applications as well as the resulting problems. Based on these considerations, we present a new approach, the *Distributed Dynamic Firewall Architecture*, which may solve the specified problems, in section 3. Further, we describe our firewall system implementation based on the architectural ideas. To demonstrate the usability of it we show an example of an appropriate multimedia application in section 4. In section 5 we discuss related work and then conclude the paper.

2 Multimedia Applications and Firewalls

We concentrate on a special type of applications - multimedia applications. These use continuous media and discrete media data [5], with the continuous media being audio and/or video streams that demand a high throughput and compliance to real-time specifics like a bounded delay or jitter. The discrete media usually consists of control data streams for the audio and video data streams and additional information (e.g. meta data).

2.1 Communication Principles

In order to describe communication scenarios, we define the following terms to distinguish the granularity at which an application's data stream is considered. A *flow* is a single data stream, identified by a tuple of characteristic values (source address, source port, destination address, destination port, protocol number). The term *channel* is often used to describe a single data stream and will be treated synonymously throughout this paper. A *session* describes the association of multiple flows that together form an application's data stream.

The protocol behavior of most multimedia applications may be generalized in the following manner. A client typically connects to a server using an initial direct TCP or UDP “connection”, often called control channel. After control channel setup, the multimedia application opens one or more data channels to transmit audio or video data. The port numbers used for these data channels are dynamically negotiated on the control channel. Detailed examples for that behavior will be given in section 4.

Some multimedia applications do not only use a unicast data channel to send the requested data but make use of multicast mechanisms to reach several clients in a very effective manner. Additionally, if the intermediate network supports these option, the server and client could negotiate QoS parameters to ensure that time critical content can be transmitted through intermediate nodes as desired.

2.2 Characteristics

Multimedia applications differ from traditional applications in many characteristics. Especially the following issues cause problems in a network which is protected by a firewall:

- **Multiple flows for one logical session:** The audio and video content is in most cases sent using additional TCP or UDP flows separate from the control data. Sender and receiver are “connected” by several flows though there is only one logical session between them.
- **Dynamic behavior:** Many of the connection parameters are not fixed and therefore possibly unknown when the communication starts. As an example, the number of audio or video streams, the bandwidth needed to transmit the streams or the TCP and UDP ports used for the streams could be initially unknown. Intermediate systems like firewalls have to consider that fact and do probably have to adapt to this dynamic behavior.
- **Complex protocols:** The Protocols used to control multiple flows and the dynamic communication are usually more complex than those used for a static flow. Firewalls and especially proxies do observe the communication and may benefit from interpreting the semantics of the communication protocols.
- **Data rate and required throughput:** The increased data rate and data volume which characterizes multimedia applications demands higher network but also transfer node performance. Firewalls have to explicitly cope with this fact to prevent transmission bottlenecks.
- **QoS:** The transmission of continuous media typically requires a guaranteed quality (e.g. described in terms of bandwidth or delay). When using the current IETF IntServ approach these are insured by means of explicitly requesting and keeping resource reservations. A firewall as a network element should not hinder the corresponding protocols and could even actively take part in resource reservations. The same applies to the use of the DiffServ approach where QoS specific operations are especially performed at edge nodes which could coincident with firewall systems.
- **Multicast:** Many multimedia applications use multicast communication to transmit audio and video content. To ensure a handling of data streams at a fine granularity, the components of a firewall system have to individually relay multicast streams. Firewall systems therefore need additional mechanisms to determine which multicast groups should be relayed [4].

In this paper we describe problems caused by the first four aspects in more detail and derive a framework to cope with them.

2.3 Security Specifics

As described above, multimedia applications negotiate the number and the specifics (e.g. port numbers) of the data channels dynamically. Therefore, these applications require intermediate systems which are capable to adapt to the current communication situation. In our basic reference scenario, only the central component in the system, the proxy, has these capabilities. It is able to recognize the flow allocation commands of the applications based on the information transmitted via their control channels. According to an appropriate interpretation of these commands it relays the communication paths towards the communication endpoints. The filters at the border of the DMZ are not involved in this dynamic adaptation to the situation. So in a standard scenario they have to be configured to let all possible connections to and from the proxy pass through. That way they are used in an operation mode not providing their maximum protection functionality. From these considerations we deduce the first design criterion for our enhanced framework:

- **Criterion 1:** All components of a firewall system should be able to autonomously adapt or should allow other components to influence and change their individual configuration. In that case there must be mechanisms to coherently control the state of the whole firewall system according to the requirements of the current communication situation. To reach that target, firewall components should be able to dynamically pass information to others.

Another typical problem caused by firewall systems in a multimedia scenario is that of a certain performance penalty. In our reference scenario every packet is sent through two filters and one proxy. This reduces the performance and limits the amount of connections that can be sent through the system.

To increase performance, many firewall vendors use a stateful inspection machine instead of a proxy, which usually allows for a higher throughput. The proxy or stateful inspection machine just forwards the data channel packets without any processing. However, the general problem remains, all traffic is sent through three components. Performance-sensitive streams could be sent directly via both filters.

Additionally it is useful to achieve more flexibility by using dedicated components with special characteristics. These arguments lead to our second criterion.

- **Criterion 2:** Generally speaking there should be means of "routing" packet streams individually through the firewall system. As an example, control channels should be separated from the data channels and can then be treated differently and by different well-suited, dynamically loadable and extensible components.

3 DDFA Systems Architecture

We develop a Distributed Dynamic Firewall Architecture (DDFA), which follows the communication paradigma explained by our two criteria. The following sections describe its components, their internal functionality and interaction.

3.1 Functional Specifications

The first of our criteria requires that all components within the firewall system have to adapt to the current communication state to increase firewall security. This requirement can be met by two general approaches. First, all components can be enhanced with additional functionality so that they are able to analyze the semantics of the communication at all protocol layers. So both filters and the proxy have to be replaced by "stateful filters". In this case all components can adapt their configuration to the current communication state. Therefore, they use information that they have retreived themselves from the observed communication paths.

Second, the communication between the components can be enabled. By communicating, a component can distribute information (about a stream, or commands to adapt to a stream) to other components. In this case the configuration of all components can be adapted by themselves or by other components. Therefore, the components can use information that they retreive themselves from the observed communication paths (if this information is sufficent) or information that is retreived from another component and is distributed to them.

To choose between both approaches, the possible impact has to be considered. Our main goal is that the provided security of the overall system should be enhanced. Changes should not strength the system at a single point by weaking it at an other point. The first approach therefore has serious disadvantages. The number of complex and independent policy engines is increasing. It is difficult to set up and maintain all three policy engines in a consistent and secure way. By using the second approach, the complexity of the system also increases, but a central and consistent view of the policy engine is maintained.

When comparing both approaches with respect to performance, the following facts have to be considered. The first method reduces system performance because the desired information must be extracted at least three times from the communication channels. The second method needs to extract the necessary information only once, but the extracted information has to be distributed to the other components which also reduces system performance. As shown, both approaches have an inferior performance than the initial system shown in figure 1, but therefore an increased security. We believe that the realization of our second criteria will outweight this performance drawback.

For security reasons, we decided to use the second approach. As described later, this decision also allows the realization of our second criteria. To fulfil the first criterion we specify requirements for the internal firewall communication subsystem as follows:

- The firewall components, e.g. filters or proxies have to be enhanced so that they are able to communicate with each other. That way they become enabled to receive missing information about the communication state from another component and may also act as an information source.
- All firewall components have to provide an interface, so that other components are able to manipulate their behaviour if this is necessary. This changes the overall system behaviour and must therefore be done in a secure manner.

The second of our criteria could be met in two ways. First, the “routing” could be performed by the components at the edges of the firewall system. This requires that these components are able to split and reassemble the flows of the multimedia sessions. Therefore these components have to support a redirect function e.g. by means of rewriting IP addresses. This redirect functions may act transparently and do not influence the behavior of the involved end systems. To perform the splitting and reassembling of the session flows, edge components have to communicate with components which observe the control channels. The edge components have to split and reassemble the session, the observing component (e.g. a proxy for the control channels) knows about the dependencies of the single flows and has to distribute this information to the filters.

Second, the routing could be performed by the proxy. The proxy handles the control flows of the multimedia session. Therefore this component is able to modify the data transmitted on the control channels. By modifying the negotiated ports on these channels, the proxy could inform the participating endsystems to send specific flows on diferent ways.

Both methods can be used to split the flows of a multimedia session. The first method does this transparent for the endsystems, while the second method does not. The first method is therefore practicable in all use cases, but more difficult to implement than the second method. To support all communication scenarios, and to be able to keep it simple where possible, we use both methods. These thoughts lead to the following firewall design requirements:

- The same design requirements as derived from the first criterion are also necessary to fulfil the second criterion. Flow information has to be exchanged between the components to split and reassemble the flows of the multimedia session.
- The components at the firewall systems edges must be able to split and reassemble packet streams individually according to specific logical sessions.

3.2 Systems Structure and Components

The internal communication requirements can be met by interconnecting each component with each other. Each of the boxes represents a firewall component with the necessary software enhancements for inter-component communication. A resulting system structure is shown in Figure 2a.

Such a design has some serious drawbacks though. There is a problem for the components to learn about the overall system state. Components have to find each other and keep track about the state of each component. Additionally we need to support the maximum number of possible communication relations. Those can not be assumed to be unique and homogenous. A component can easily become over-featured and therefore over-sized and difficult to implement on different systems. That directly leads to portability problems. The whole complex software enhancement has to be rewritten every time it is ported to another system type. To avoid these problems we use system structure as shown in Figure 2b.

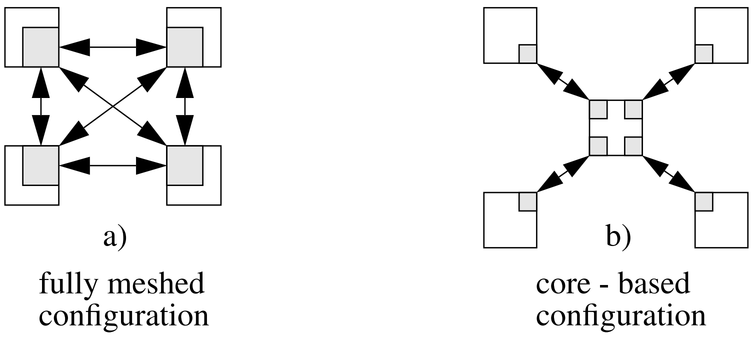

Figure 2: System Structure Alternatives

A central component (we further on call it DDFA-core) forms the central part of our firewall system. This component provides registration mechanisms which allows components to announce theire presence within the system. The core is then able to provide a location mechanism which enables the components to find each other.

The system enhancement for each firewall component are split in two parts.

- The system-dependent part is located on the component itself.
- The system-independent part is located within the core component.

That way a special additional adaption layer is inserted, which makes it easier to tailor the software. It allows to use different types of components providing the same functionality. The core can also be used to maintain tasks which control and organize the interaction of all interconnected components.

4 DDFA Prototype

The functional specifications and the derived system structure has been used to improve the standard firewall scenario shown in figure 1. The enhanced version of the standard scenario is shown in figure 3.

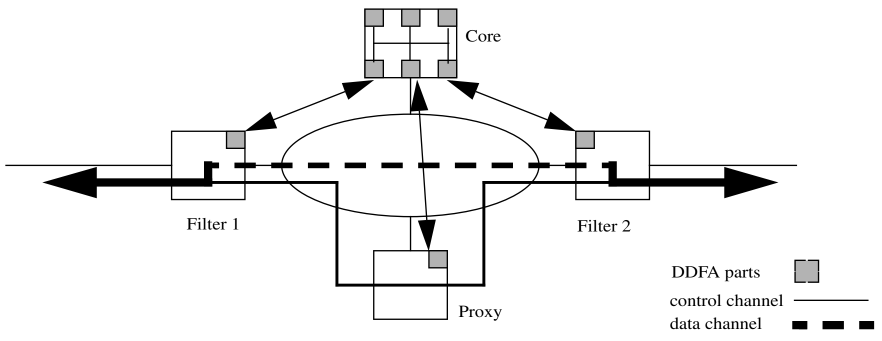

Figure 3: Standard Scenario with DDFA

This DDFA prototype system consists of two filters (FreeBSD 3.2 with IP-Filter 3.2.10) and a proxy host (FreeBSD 3.2 with a self implemented H.323 Proxy) and a core host. Therefore the prototype only supports IP-Telephony apllications based on the H.323 protocol.

4.1 Core Component

The DDFA core forms the main component of our system. The system independent parts of the software enhancements for the firewall components are also located there. The following tasks are fulfilled by the core component:

- **Location mechanism:** The components which participate in the firewall system use the core as a central contact point. The core registers the components and publishes their presence so that they can be addressed by other components.
- **Communication:** The core provides a general communication mechanism, so that the components can interact with each other. This communication is coordinated by

the core, so that requests are submitted and computed in a strict sequential order and an atomic way.

- **Authentication:** The interaction between the components can be controled and restricted. The core provides access control mechanisms and thus decides which components are allowed to participate in the system.
- **Protocol specific features:** Tasks that are usually located at separate firewall components can dynamically be loaded into the core. This is an implementation detail and allows for a higher systems throughput through the efficient use of a single address space.
- **Control tasks:** System startup and individual control functions such as clean-up of component specific data structures are also located at the DDFA core.
- **System independent parts:** The core provides mechanisms to load the system independent parts of the connected components.

The design of our system is modular to simplify enhancement of system functionality. The design is flexible to adapt to different firewall policies. The internal mechanisms and programming details are described in [14].

4.2 Component Adapters

We now show two different adapters to integrate firewall components into the DDFA. One adapter is used to integrate a filter, the other to integrate an IP-Telephony (based on the H.323 protocol family) proxy. Every adapter consists of a system dependent part, installed on the firewall component itself, and a system independent part which is loaded as described into the core.

IP-Filter Adapter:

An IP-Filter adapter is used to integrate “IP-Filter” packet filters hosted on a FreeBSD operating system into the DDFA system. The system independent part provides a generic interface within the DDFA-core to access filters in a standardized manor. Other components can reconfigure the filter, redirect streams or request flow information from the filter by using this interface. The system dependent part is hosted on the filter machines. This part is system dependent because it has to communicate with the filter software, the operating system and the network interfaces on the filter host. It translates the standardized commands from the core into the specific language used for the particular filter. Therefore, parts of this system dependent component must be rewritten in most cases when the target operating system or filter software is changed. System dependent and system independent part are connected via a secured TCP link [14].

IP-Telephony Proxy Adapter:

An IP-Telephony-Proxy adapter can be used to make H.323 proxy functionality available within the DDFA system. The system independent part provides an interface, which allows other components to modify the proxy behavior or to receive informations about the flows processed by the proxy. In addition, the proxy uses this interface to communicate with other components. This proxy is described in [15].

4.3 Communication example in Detail

A representative example of a Multimedia Application supported by our system is Microsoft NetMeeting . It is used for multimedia conferencing and is based on the H.323 protocol suite [9]. Figure 4 shows the communication mechanism between two H.323 based IP-telephony clients. In this figure the caller (later assumed to be on the internal network behind the firewall) initiates an IP-phone call to the called (later assumed to be on the external network) client. Only audio streams will be considered between both clients. If video streams between the clients are used too, two additional streams (RTCP and RTP) in each direction are added.

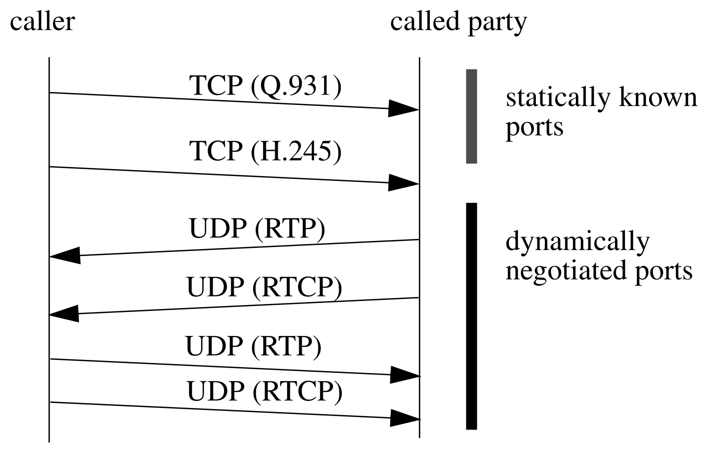

Figure 4: H.323 communication

In a H.323 session two TCP control channels are utilized. The first control channel is employed for call setup and uses the Q.931 protocol. The ports used for the second control channel are dynamicaly negotiated on the Q.931 channel. After the second (H.245) control channel is established, this channel is used between the clients to establish the audio and video streams between both clients. The ports used for these flows are negotiated dynamically on the H.245 control channel.

To show the difference between a standard firewall system (shown in Figure 1) and our extended DDFA system (shown in Figure3), we will explain how both systems handle a H.323 application. The H.323 Application is handled by a system shown in Figure 1 in the following manner:

1. Boundary conditions:

- The calling H.323 client has to support a proxy. The application has to know that it must route the call over the proxy.
- Filter 1 must pass the initial Q.931 (TCP, destination port 1720) connection from the caller to the proxy, Filter 2 to must pass the initial Q.931 (TCP, destination port 1720) connection from the proxy to the called client.

- Filter 1 must pass the H.245 (TCP, destination port greater than 1023) connection from the caller to the proxy, Filter 2 to must pass the H.245 (TCP, destination port greater than 1023) connection from the proxy to the called client.
- Filter1 and Filter 2 must be configured to allow all UDP flows with a destination port greater 1023 to and from the proxy.

2. Communication:

- The calling host connects to the proxy via TCP at port 1720. The proxy then relays this connection to the destination host. This Q.931 connection is now survied by the proxy.
- The proxy recognizes that the H.245 port is negotiated on the Q.931 control channel. The proxy also relays and observes this control channel between both endpoints.
- The proxy recognizes the negotiation of the audio channels. By modifying the submitted ports and IP-addresses the proxy gets these audio streams and relays them between both endpoints. The bulk data transfered between the clients are relayed by the proxy.

If the application is handled by our modified firewall system, the boundary conditions and the communication mechanisms change. The system configuration and the session flows are shown in Figure 3.

1. Boundary conditions:

- Filter 1 must pass the initial Q.931 (TCP, destination port 1720) connection from the caller to the proxy. In addition, all TCP connection attempts, made from the internal client to the external client on port 1720 should be redirected to the proxy. In this case, the application does not need to support a proxy, the proxy is transparently inserted in the communication path.

2. Communication:

- The calling client tries to connect to the called client via TCP on port 1720 to set up the Q.931 connection. Filter 1 redirects this request to the H.323 proxy. The proxy then asks via itscore connection the Filter 1 component about the state of this flow. As result the proxy gets the information that this connection was primary made to the destination client. The proxy now uses the core connection to inform filter 2 that he will connect the destination client on port 1720 via TCP. Filter 2 adjusts his configuration to allow this connection. The proxy now connects to the destination client and relays and observes the Q.931 connection between both clients.
- By observing the Q.931 connection, the proxy recognizes the negotiation of the H.245 connection. To negotiate the H.245 connection, the destination client passes information about the target IP-address and port for the H.245 connection to the calling client. This information is modified by the proxy, so that it will receive the connection request for the H.245 connection. Before the proxy passes the modified message to the calling client, it uses the core connection to inform Filter 1 about the connection that will be made to the proxy. Filter 1 adjusts his filter rules.
- The proxy now receives the H.245 connection request. It informs the Filter 2 via the core about the outgoing connection to the destination client. Filter 2 adjusts its filtering rules. The proxy now connects to the destination client and relays and observes the H.245 connection.

- Next the audio data streams are negotiated on the H.245 channel. The proxy observes this communication and informs both filters about the negotiated streams. The filters then change the filter rules according toinformation passed by the proxy. The bulk data are now sent directly between the clients. These data do not have to be relayed by the proxy.
- When the communication is going to be finished, special messages are sent on the control channels. This is recognized by the proxy, and it distributes this information to the filters, so that all previous opened paths within the filter rules could be closed. When the communication is finished, the system configuration is again in the state described in the section “Boundary conditions”.

4.4 Security Concerns

As we described, the main difference between the standard firewall and our DDFA system is the initial configuration. In the standard system several predefined “holes” within the filter configurations are necessary because an adaption of the system during the communication is not possible. The DDFA System does not need these predefined holes, because the system can open and close the actual necessary paths on all components during the communication. The DDFA System, therefore, allows a more secure operation, regarding the filter configuration, than the standard system.

Finally we have to consider if the overall DDFA System is also more secure then the standard system. The overall system is more complex then the standard one, but a central and consitent view of the policy engine is maintained. An administrator of the firewall system will not recognize the difference between configuring the central component in the standard scenario or configuring the DDFA system. Therefore configuration errors could be possible with equal probability in both systems. Because the internal design and implementation of the DDFA system is secure, this system provides a higher security level then the standard one.

4.5 Performance Concerns

As shown, in the DDFA system, the bulk data (audio and video flows) are sent directly via both filters between the endpoints. In the standard scenario, the audio and video streams are additionally processed by the proxy located between both edge filters. Therefore our DDFA system has the following performance advantages:

- The bulk data are only processed by filters. By avoiding the usage of proxies for the data flows, performance is increased [13].
- The bulk data are only processed by two components. By reducing the amount of hops, the performance is increased.

As mentioned, the distribution of flow information within the DDFA system leads to an inferior performance. This performance reduction only affects the control channels of the multimedia session. Therefore, theoretically, the usage of the DDFA system leads to a slower session setup (and session tear down), because of the delay on the control channels. Subjective we could not recognize this delay during our first tests. The theoretically proof proposition according the DDFA performance has to be verified. Therefore measurements have to be done.

5 Related work

As the increased use of multimedia applications not only in the research community but also in commercial environments generates an increasing demand for adequate secure and yet performant solutions - there is a lot of further research activity on that topic. An approach to support the requirements of high data rates is described in [10]. The authors propose parallel firewalls to support high performance networks. In their approach, the connections are dynamically distributed to different proxies because the proxies represent the bottlenecks of firewall systems. The distribution is done by one or several packet filters at the edge of the system via network address translation. This approach allows for scalability, yet at significant costs since the data streams are still routed through a proxy, which is not necessary in our implementation.

Using the SOCKS protocol, specified by the Authenticated Firewall Traversal Working Group [11] of the IETF, a client that wishes to establish a connection to an object that is reachable only via a firewall must open a TCP connection to the SOCKS server system and has to authenticate at the server. The SOCKS server evaluates the request and if that proceeded successfully - establishes the appropriate connection directly. This approach has some major disadvantages. The implementation of the SOCKS protocol typically involves the re-compilation or re-linkage of TCP-based client applications to use the appropriate encapsulation routines in the SOCKS library. Often this is not possible. Also it can be used only for communication between known partners, which restrict its usability.

The PIX firewall system developed by Cisco [12] is based on a combination of stateful filters and proxies. Their approach is to authenticate a user at a proxy and to build up the initial connection. If this is successful all session flows are directly passed-through between the two parties while maintaining control of the session state. This architecture reaches a high throughput but there is a limited possibility to configure additional components (e.g. packet filters) dynamically to adapt the whole firewall system. We consider our approach at least as comparable and even more flexible for emerging new multimedia protocols.

6 Summary and Outlook

We presented a distributed firewall architecture and implementation, which is targeted to solve the problem of efficiently supporting multimedia applications in a secure manner. The main idea of our approach is, to treat a firewall system as a distributed architecture of specialized components and to dynamically adapt all these as well as their interactions to the current communication situation.

By implementing a prototype we showed the general usability of our approach. In a representative example scenario we described how the data channels are directly passed through the system, whereas the control flow is handled by a proxy. The prototype system determines which connections are allowed by using IP-, TCP- and UDP-filter lists. Communication paths through the firewall system are opened on demand and only when they are really needed. The proxy approach allows us to also implement support for a user specific authentication which will definitely be needed in a production environment.

Based on our implementation we actually measure the performance of the system, using the utilities and methods presented in [13] in order to compare with other approaches. Finally we plan to add features to both the systems architecture and the user interface to improve the usability of the system. In a future implementation step CORBA usage in the core system will be evaluated. Thereby we intend to transparently distribute the functionality of the core component over several hosts in order to increase the resilience of the system and its overall performance.

7 References

[1] Network Associates: Application Gateways and Stateful Inspection, http://www.avolio.com/apgw+spf.html

[2] Chapman, D.B.: Building Internet Firewalls, O'Reilly, Cambridge, 1995

[3] Cheswick, W.R., Bellovin S.M.: Firewalls and Internet Security, Addison Wesley, 1994

[4] Finlayson, R.: IP Multicast and Firewalls, Internet Draf draft-ietf-mboned-mcast-firewall-02.txt, 1998

[5] Steinmetz, R., Nahrstedt, C.: Multimedia: Computing, Communications & Applications, Prentice-Hall, 1995

[6] Comer, D.E.: Internetworking with TCP/IP, Volume I, 2nd Edition, Prentice Hall, 1991

[7] Reed, D.: IP-Filter, http://coombs.anu.edu.au/~avalon/

[8] Progressive Networks: Real Audio, http://www.real.com/

[9] ITU: ITU-T Recommendation H.323, Packet-Based Multimedia Communications Systems, 1998

[10]Ellermann, U., Benecke, C.: Parallel Firewalls: Scalable solutions for High-speed Networks [German], DFN-CERT Workshop Sicherheit in vernetzten Systemen, Hamburg 1998

[11]Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., Jones, L.: SOCKS Protocol Version 5, RFC 1928, 1996

[12]Cisco: Cisco's PIX Firewall Series and Stateful Firewall Security, White Paper, 1997

[13]Ellermann, U., Benecke, C.: Tools for measuring the Performance of Proxies [German], published in MMB-Arbeitsgespräche: “Leistungs-, Zuverlässigkeits- und Verläßlichkeitsbewertung von Kommunikationsnetzen und verteilten Systemen”, Hamburg 1998

[14]Utz Roedig, Ralf Ackermann, Christoph Rensing, and Ralf Steinmetz. DDFA Concept. Technical Report KOM-TR-1999-04, KOM, December 1999

[15]Utz Roedig, Ralf Ackermann and Ralf Steinmetz. Evaluating and Improving Firewalls for IP-Telephony Environments. The 1st IP Telephony Workshop, Berlin 2000