mTunnel: a multicast tunneling system with a user based Quality-of-Service model[*]

Peter Parnes, Kåre Synnes and Dick Schefström

Luleå University of Technology,
Department of Computer Science/Centre for Distance-spanning Technology,
971 87 Luleå, Sweden,
{Peter.Parnes,Kare.Synnes,Dick.Schefstrom}@cdt.luth.se

Abstract. This paper presents a system, called mTunnel, for application level tunneling of multicast traffic in a lightweight manner, where the end-user is responsible for deciding which MBone-sessions and multicast groups to tunnel. mTunnel is primarily designed for easy deployment and easy-to-manage tunneling. Therefore it runs as an user application and does not need access to restricted system resources.
Information about currently tunneled sessions and control of mTunnel is provided through a Web-interface. To allow the user to easily start tunneling of announced MBone-sessions, mTunnel listens for announcements and presents this information through the Web-interface.
To save bandwidth, tunneled streams can be translated in four ways: audio streams can be recoded to an encoding that requires lower bandwidth, several audio streams can be mixed together, streams can be switched based on activity in another stream and streams can be scaled by dropping a certain percent of the traffic.

1 Introduction

The Multicast Backbone, MBone [6] has existed for several years and is slowly being deployed in production networks. When a host sends a packet on the MBone it does not send it to any specific host, but instead to a so called *multicast group*. The sending host does not need to know which hosts are members of this group, but that information is stored in and kept up to date by the network. Packets sent to a group are only sent to the parts of the Internet, where there are listeners. Local routers detect if there are any local listeners to a specific group using the *Internet Group Management Protocol - IGMP* [1]. IGMP is also used by hosts to signal that they want join or leave a multicast group.

To ease the development of the MBone it was first developed as a virtual network on the Internet consisting of tunnels between machines that acted as virtual routers, sending multicast packets encapsulated in unicast packets. The software used is called *MRouted* [4]. Today, multicast routing functionality is

[*] This work was supported by the Centre for Distance-spanning Technology (CDT), Luleå, Sweden

being built into standard Internet routers, but there is still no general support for multicast over point-to-point links (such as analog modems or ISDN-links).

MRouted uses the normal routing tables and does real multicast routing, which means that a lot of routing information is also sent over the tunnels. On low bandwidth links, this can cause a problem because much of the bandwidth is used for control traffic and not the data itself.

MRouted also use IGMP to detect members of multicast groups and decide on which groups to tunnel based on the group membership information. This means that if a user on the local network joins a group, traffic to and from that group will automatically be tunneled as long as the users tool is running. This automatic tunneling can be a problem on low bandwidth links, as local users do not see which groups are currently being tunneled and may make the total bandwidth requirement too high by joining to many groups. If this happens, all currently tunneled groups get affected as packets will be dropped at random.

Another reason for using an unicast tunnel is firewalls. At some company's it has proven to be very hard to convince the system administrators to open the companies firewall for multicast traffic.

mTunnel is an application for tunneling of multicast traffic based on user requests rather than based on IGMP. This means that users explicitly have to choose which groups to tunnel. mTunnel, also allows the users to specify how different groups should be prioritized, which is useful if a group of users are currently using mTunnel to tunnel an electronic meeting and do not want to be disturbed by another user that, for instance, want to watch the current NASA mission in Space[2]. mTunnel uses the World Wide Web as its main user interface.

The first design goal of mTunnel was to easily tunnel multicast traffic through non-multicast enabled parts of the Internet or within an intranet. The second design goal was to let the end users easily make the decision on which groups to tunnel. The end users should also be able to easily prioritize different parts of the traffic. The third design goal was that it should waste as little as possible of the available bandwidth on control data. The fourth and last design goal was that it should be as platform independent as possible.

The rest of this paper is divided into: Sect. 2 that presents the general architecture of mTunnel and its features, Sect. 3 that presents the user based Quality-of-Service model used by mTunnel, Sect. 4 that presents the implementation and current status and Sect. 5 that gives a summary and conclusion about the work.

2 The mTunnel application

mTunnel consists of four main parts, the *tunneler* which performs the tunneling itself, the *controller* that keeps both ends of the tunnel synchronized and updates control-clients, the *Web-interface* which present a user interface to mTunnel using the World Wide Web, and finally the *translator* which can translate streams in various ways.

[2] NASA currently multicasts most of their shuttle missions.

As with all tunnels it has two end-points (see Fig. 1), between which the traffic is tunneled. For that reason there must always be two copies of mTunnel running, one at each side of the tunnel.

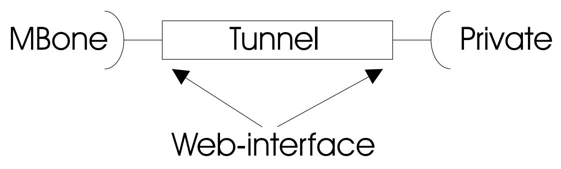

Fig. 1. A tunnel connecting a private network to the MBone.

mTunnel runs as a user process to allow for easy deployment of new and perhaps temporary tunnels.

Sessions

The term *session* is used as a name for one tunneled stream. A session consists of: a multicast group, a base port which is the lowest port number to tunnel, the number of consecutive ports[3] including the base port to tunnel, the Time To Live of the outgoing packets (see Sect. 2.6), and the name of the session.

Statistical information about the number of bytes and packets sent and received is also stored within a session.

A session can also include information about prioritization, if different from the default value (see Sect. 3).

2.1 The tunneler

The *tunneler*, is the part of mTunnel that does the actual tunneling. It listens on a number of multicast sockets (based on which groups to tunnel), encapsulates the incoming data, and sends it through the tunnel (see Fig. 2). The mTunnel on the other side, receives the encapsulated unicast packet, decapsulates it and resends it locally using multicast.

The tunneled data is sent over a unicast "connection" between the two end-points and all data is sent over the same port number. This allows for easy tunneling through firewalls (as the systems administrator only has to open one port in the firewall).

[3] Most multicast streams on the MBone use the Real-time Transfer Protocol [10], that uses more than one port, one for data and one for control.

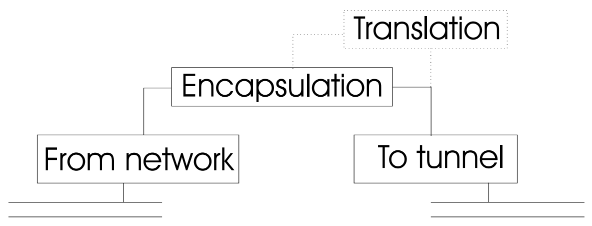

Fig. 2. Path of an incoming packet with optional translation.

Encapsulation of data

Each multicast packet is encapsulated by adding the multicast address of the group from which the packet was received, and the port, to the end of the packet (see Fig. 3). Because mTunnel runs as a user process, it does not have access to raw IP packets or the kernel buffers used to receive the packet by the operating system. Therefore, to minimize the number of copy-operations needed, the tunneler uses a memory buffer that is larger than needed to receive the incoming packets. This allows the tunneler to add the encapsulation information after the data in the same buffer as the packet was received in. On the other side of the tunnel, the same packet is recent without any extra copy-operation, by stripping of the earlier added encapsulation data.

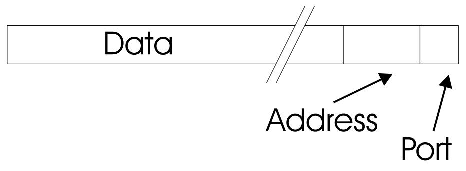

Fig. 3. An encapsulated packet where 'Data' is the original multicast packet.

Transmission loops

mTunnel is designed to connect a network or a single host that currently is isolated from the MBone, to the MBone. But, if both ends of a tunnel is started within the MBone, a transmission loop can occur if the tunneled MBone-sessions and TTL values are not chosen carefully. mTunnel therefore does not forward packets trough the tunnel if the sender matches the other end of the tunnel. Unfortunately, if two separate tunnels are deployed that together create a loop, packets will be forwarded over and over again.

If mTunnel suspects that a loop has occurred (the packet rate through the tunnel suddenly raises dramatically), it sends out a special probe-packet and waits for the probe-packet to be received again. If the probe-packet *is* received all current tunneling is stoped and users of the system are notified by the Web-interface. If the probe-packet *is not* received, the process is repeated a couple of times, because the probe-packet could have been lost on the way due the to best-effort nature of UDP-packets.

2.2 The controller

The *controller*, is the part of mTunnel that keeps the two endpoints synchronized. That is, if a new session is added on one end of the tunnel, the controller sends a message to the other end telling it to add the same sessions. The controllers communicate with each other over a TCP-connection for reliable messaging. If one end of the tunnel is restarted, the other end automatically updates the first end with information about current sessions. The controller also allows for connections from clients who wants a simpler and faster (than through the Web-interface) way of communicating with the server.

Using the controller each session can be controlled in various ways:

- **Pause/Continue**: The tunneler can be instructed to temporarily pause tunneling of a session. If the pause lasts for more than one minute the tunneler also leaves the multicast group by sending an IGMP leave message for that session, which makes traffic to that group stop. This has the advantage of the traffic to the end of the tunnel connected to the MBone lowers, but unfortunately this also makes the delay longer when the tunneling of the session is continued and no one else is listening to the same session (since the tunneler must rejoin the group and there is always an initial delay before the first packets are received).
- **Priority**: The priority for the session can be changed (see Sect. 3).
- **Translation**: The tunneler can be instructed to start a translator. A translator can recode a stream between different encodings; mix several streams into one stream; switch the traffic, meaning that only packets from a certain source is forwarded based on another stream (for instance, video packets are only tunneled for the member that currently is speaking; and streams can be scaled, meaning that parts of the traffic is just dropped (see Sect. 2.4).

2.3 The Web-interface

As one of the design goals was to allow for easy deployment, there is a small and minimal Web-server built into mTunnel. The alternative was to use a separate Web-server and the CGI-interface [7], but the built in Web-server allows for faster access than through a CGI-program (it also allows for a simpler implementation).

The Web-interface lets the user watch information about current sessions and statistics about the total number of packets and bytes sent through the tunnel. It also allows the user to control the current sessions and create new sessions.

Creation of a new tunnel session

Tunneling of new sessions can be specified in two ways: 1) manually entered or 2) chosen from a list of earlier announced MBone-sessions:

1. **Manually:** The user first chooses how many sessions to start (normally one session per available media). Second, the user specifies a name that is common for all the new sessions, the TTL with which outgoing packets should be sent, and for each session the user specifies the media-type, the multicast group, the base port and the number of ports to tunnel.
2. **Chosen:** The user first chooses a MBone-session to tunnel from a list over announced sessions. The information about announced sessions is gathered using a built in version of the multicast Session Directory, mSD [8], which is an application for displaying information about currently announced MBone-sessions. Second, the user chooses which sessions to create (an announced MBone-session can include several different media which results is several tunnel sessions).

2.4 The translator

The last part of mTunnel is the *translator*, that translates the traffic in various ways. Currently the translator includes a *recoder*, that translates between different encodings; a *mixer*, that mixes several streams into one stream; a *switch*, which only forwards packets from a certain source based on another stream (for instance, video packets are only tunneled for the member that currently is speaking); and a *scaler*, that rescales the traffic by dropping packets in predefined ways.

- The *recoder*, parses incoming traffic and translates it to another format. Currently, translation from PCM to GSM is supported, which for a normal MBone audio session would result in a reduction in bandwidth of about 75% (from 78 to 17 Kbps). LPC support is currently under development and would result in a reduction of about 88% if translating from PCM.
- The *mixer*, mixes several streams into a single stream. This allows for a bandwidth reduction, if several sources are active at the same time, but has the drawback of that the final receiver can not choose only to play out data from one sender. Currently mixing of PCM and GSM is supported.
- The *switch*, selects which packets to forward through the tunnel based on the activity in another stream, e.g. it can be instructed to only forward video packets from the source that currently is also sending audio packets.
- The *scaler*, rescales traffic by dropping certain parts of a traffic flow. This is currently only applicable on video traffic where the scaler drops packets randomly or drops complete frames depending on the type of traffic. This rescaling is possible due to the nature of video encodings on the MBone today, which are designed to tolerate packet-loss.

The different modes of the translator can of course be stacked after each other, e.g. an audio session can be both mixed and recoded.

2.5 Security issues

There are three main aspects of security in mTunnel: access to the interface, access to the tunneled data and firewalls.

Access to the Web-interface and the controller can either be public or based on simple user/password authentication. If the access is public, anyone that has access to the Web-interface can get information about current sessions and start new sessions, but only the host that created a session can alternate it (that is pause, continue, stop and/or change the priority).

When tunneling data over public networks, there is sometimes a need to hinder vicious persons from reading the tunneled data. This is solved using encryption of the tunneled traffic.

A third concern regarding security, is how to tunnel data through firewalls where it has proven administratively hard to open the firewall for regular multicast traffic. Here, mTunnel makes it easier to tunnel traffic as all tunneled data always have the same connection, i.e. its source and destination host and port is always the same. By this, it is easier to convince security administrators to open the firewall based on that connection pattern.

2.6 Time To Live

When packets are sent on the MBone today, their reach is limited by a so called *Time To Live (TTL)* value. For instance, if a user wants to send multicast packets to the local network only, he sends them with a TTL of 1.

In the current standard version of the sockets interface[4] under Unix and Windows, there is no way for a user application to get information about the TTL of an incoming packet. Due to the socket interface, and the fact that mTunnel runs as a user-application, mTunnel can only forward packets based on the TTL value specified when the session was created. Unfortunately, this means that if a user sends traffic in an announced MBone-session with a lower TTL than the announced TTL, the local packets will be "amplified" and retransmitted with a higher TTL than intended by the original sender.

The only available solution to this problem today is to make the users of mTunnel aware of the problem.

2.7 MBone Session Announcements

mTunnel includes an option for tunneling of MBone session announcements. These announcements are multicasted using the Session Announcement Protocol [5] over a known group-address and port-number. When mTunnel is started, it can be instructed to automatically tunnel these announcements. This allows for users on the side that is normally not connected to get information about current MBone-sessions. This SAP-tunneler, can also be instructed to only tunnel information about MBone-sessions that are currently tunneled. It can also

[4] The way an application speaks with the operating system and the network.

create new announcements for sessions that are manually entered (that is, they are not announced in the normal MBone manner).

To allow for faster update, known MBone-sessions are cached locally and are read from the cache and re-announced when mTunnel is restarted.

3 User based Quality-of-Service and session prioritization

The MRouted application (as presented in Sect. 1) connects networks to the MBone and makes them "true" members of the MBone, by tunneling all requested traffic. The decision on which sessions to tunnel, is based on requests made by the multicast aware applications that the users on the other side of the tunnel starts.

This model works very good if the bandwidth is not limited, as several sessions can be tunneled at the same time. But, if the bandwidth is limited (like over analog modems or ISDN-links) the users have to quit their MBone applications to stop a session from being tunneled (i.e. if an application "wants" multicast traffic for a special multicast group, usually the only way is to quit the application to stop the traffic). Also, if several users share the same narrow link, it might complicated or even impossible to coordinate which sessions to tunnel (several users join different sessions at the same time).

mTunnel instead uses a *user based Quality-of-Service model* where the local users explicitly have to choose which sessions to tunnel. This has several advantages, such as: making the end users aware of other currently tunneled sessions and removing the need for users to quit their MBone tools to stop the tunneling of specific multicast groups.

Another disadvantage with using MRouted for tunneling over dial-up links is that it uses the same link continuously for exchanging router information, even if no actual traffic is currently being tunneled. This means that links that have automatic set-up and tear-down will be kept dialed up as long as the MRouted program is running. mTunnel does not exchange this router information, as it only tunnels multicast traffic and is not a full router implementation (other multicast routers will not see the mTunnel as a router, but only as simple end-host in the network).

3.1 Prioritization

If the total currently needed bandwidth exceeds the available bandwidth, the tunneler can throw away packets (not send them through the tunnel) based on user defined priorities.

By default, all sessions have the same priority, but using the controller and the Web-interface, the priorities can be changed. Priorities for one or several sessions can also temporally be locked, meaning that no other session can get a higher priority than that/those sessions. This is useful if an important electronic meeting is conducted over the tunnel and the participants do not want to be disturbed by another user who wants to watch some other MBone-session.

Priorities, can also be configured in mTunnel, based on a number of different variables in the session: the media-type, the multicast address and port, the used bandwidth, the name and the description. This allows for advanced selection of priority schemes, that enables a user to participate in sessions even if the total needed bandwidth is not available.

4 Implementation and Status

The current prototype is implemented in the platform independent Java language (version 1.1), except some parts of the audio recoder that is implemented in C for efficiency reasons. The audio recode functionality is currently only available on Sun/Solaris, but the rest has been tested to work under both Unix and Windows95/NT4.

mTunnel is currently being used in three different ways: to connect different parts of a large software company's intranet, to connect computers at users homes, and to connect industry networks to the MBone.

More information about the current version and status of mTunnel can be found at [9].

4.1 Further issues

Important issues currently not addressed within the mTunnel development includes the ideas of *header compression* [2, 3], which together with stream based flow labels would result in a reduction of the bandwidth requirement. This means that not the full header would be transmitted through the tunnel, but common parts between packets in a stream would only transmitted with regular intervals and removed from the packets inbetween.

The encryption of the security part is currently completely missing and will be implemented in the future.

Another issue that requires further examination is compression of data sent over the tunnel. Several tunneled packets could be grouped together and then be compressed before sent through the tunnel.

The translator should also be extended to include a larger variety of encodings.

5 Summary and Conclusions

This paper presents a system for allowing users to easily connect to the MBone infrastructure and to connect different isolated multicast capable networks.

mTunnel allows users easy access to information about current tunneled sessions through a Web-interface, which also allows for easy configuration of existing and future sessions.

mTunnel does not start tunneling of MBone-sessions based on current multicast group activity, but instead makes the user responsible for deciding which

MBone-sessions to tunnel. This allows for a *user based Quality-of-Service model* where service decisions are left to the user.

To save bandwidth, data streams can be translated in four different ways: audio can be recoded to an encoding that requires lower bandwidth, several simultaneous audio streams can be mixed into a single stream, streams can be switched based on another stream (for instance, video packets could be forwarded, based on which audio packets are currently being received), and streams can be scaled by dropping certain parts of the traffic.

The system is currently being used to connect different parts of a large software company's intranet, to connect computers at users homes to the MBone and to connect industry networks to the MBone.

The usage of mTunnel has shown and proven that it is useful and that there is a need for this kind of applications.

Acknowledgments

Thanks to Mattias Mattsson, Fredrik Johansson, Håkan Lennestål, Johnny Widén and Ulrika Wiss, CDT, for interesting comments, encouragement and feedback.

This work was done within Esprit project 20598 MATES, which is supported by the Information technology part of the 4:th Framework Program of the European Union. Support was also provided by the Centre for Distance-spanning Technology (CDT).

References

1. S. Deering. Internet Group Management Protocol - IGMP. IETF RFC1112.
2. M. Degermark, M. Engan, B. Nordgren, and S. Pink. Low-loss TCP/IP header compression for wireless networks. In *Proceedings from MobiCom*, 1996.
3. M. Degermark and S. Pink. Soft state header compression for wireless networks. In *Proceedings from 6th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, 1996.
4. B. Fenner. MRouted. <URL:ftp://ftp.parc.xerox.com/pub/net-research/ipmulti />.
5. M. Handley. Session Announcement Protocol - SAP. work in progress.
6. V. Kumar. The MBone information web. <URL:http://www.mbone.com/>.
7. NCSA. Common Gateway Interface - CGI. <URL:http://hoohoo.ncsa.uiuc.edu/ cgi/>.
8. P. Parnes. The multicast Session Directory - mSD. <URL:http://www.cdt. luth.se/~peppar/progs/mSD/>.
9. P. Parnes. The multicast Tunnel system - mTunnel. <URL:http://www.cdt. luth.se/~peppar/progs/mTunnel/>.
10. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. IETF RFC1889.

This article was processed using the LaTeX macro package with LLNCS style