Scalable Network Support for Application Requirements with Forwarding on Gates

Florian Liers
Technical University of Ilmenau
Ilmenau, Germany
florian.liers@tu-ilmenau.de

Thomas Volkert
Technical University of Ilmenau
Ilmenau, Germany
thomas.volkert@tu-ilmenau.de

Andreas Mitschele-Thiel
Technical University of Ilmenau
Ilmenau, Germany
mitsch@tu-ilmenau.de

I. INTRODUCTION

The scalability of functions existing in a network is one of the most important challenges for large inter-networks. The quality of service (QoS) use-case with its IntServ and DiffServ solutions was the most prominent of the discussed features in the past. IntServ stores state information for each connection on all nodes along the path of the connection. The states are managed by using a signaling protocol, like e.g. RSVP. In inter-networks, routers suffer from the large amount of states needed for handling the high number of connections. In order to deal with this scalability problem, DiffServ maps several connections traversing an autonomous system (AS) to a smaller number of internal traffic classes. For example, all VoIP connections are mapped to a delay optimized traffic class. If this mapping is done, based on deep packet inspection, no connection oriented state information is needed. As drawback the application loses its possibility to inform the network about its specific needs. In special, that hampers the deployment of new applications because they are not considered in the mapping. This is important for the flexible and open end host concept which is the main driver of innovations for the Internet. By using signaling with a DiffServ approach, the network would be enabled to react to the specific needs of an application. But a scalability problem occurs at the ingress nodes of each AS. These nodes have to handle the signaling and store the necessary mapping states.

This scalability problem gets more severe in Future Internet architectures, providing functionality by dynamically constructing stacks based on functional blocks [1, 2]. The scalability problem arises in these systems not only for QoS but also for all other functions requiring states. In addition to classical network based functions, like multicast or mobility support, more application requirements will likely appear in the future. Examples are an integrated network-based virus scan for incoming data or or video re-coding for small mobile devices. Regarding the scalability challenges for a network, the main questions, which have to be answered, are where to place functions and the required state information, and how to re-use them for multiple connections.

This work was funded by the German Federal Ministry of Education and Research under the project G-Lab_FoG (support code 01BK0935).

The key contribution of this work is a demonstration of a scaling inter-network system, based on dynamic composition of functional blocks. New function instances are created, depending on application requirements. Furthermore, and more important, the system is able to re-use existing functions and their states for multiple connections in order to improve scalability. The simulator shows that the re-use is possible without per-connection state information on the hosts which provide the functions.

II. FORWARDING ON GATES

The demonstration uses the inter-network protocol "Forwarding on Gates" [3] (FoG). It uses index-based forwarding, which was adapted for networks consisting of functional blocks. The index-based forwarding concept [4, 5] separates forwarding from routing of packets by using lists of indices as routes. Such lists represent the decision of of the routing and are used by the forwarding to relay data without doing routing. In general, they are stored in the packet. Typically, the lists are calculated by the routing at the start of a communication. FoG differs from known systems by the following two main points:

a.) FoG is based on functional blocks. These blocks are called "gates" in the context of FoG. An index represents the next functional block to which the data has to be transferred to. Known systems, like Pathlet [5], encode only the next hops with these indices. Using functional blocks as the base for networking let the routing operate on graphs of blocks, too.

b.) FoG uses an incremental routing process. In contrast to source routing based systems, the routes in FoG are created incrementally by concatenating partial routes. During this process, FoG packets include only parts of the whole route. Therefore, the routing process is neither source routing nor hop-by-hop routing; it is in between these two extreme cases.

The first point enables functional blocks ("gates") to be part of the routing. Therefore, the routing can decide to reuse gates for multiple connections. In general, gates can be reused if their function does not depend on the communication data itself and if parameters used for the gates are identical. The second point enables scalable

implementations of the routing since each routing instance needs to calculate a partial route only in its known surrounding.

If FoG is used, each application is able to specify explicitly its requirements for the communication with peers, like packet ordering, encryption, automatic retransmission or the maximum transmission delay. Based on the location of a communication request and its parameters, the routing instances are contacted. They calculate a route through the graph of gates satisfying the requirements and reaching the destination. In addition, the creation of new function instances can be necessary. In the current demo, the placement of a function is done by an approach, which instantiates the function on the first node along the communication route, whose policy allows this.

III. DEMONSTRATION

Our demonstrator illustrates the two mentioned key features in a live scenario as depicted in Figure 1. Our use case is live video streaming and uses three different requirement sets. In general, the stream is transferred in a UDP-like fashion. The application defines additional non-functional and functional requirements, the network has to satisfy.

Basically, our demonstrator consists of two applications, developed by us. The first one is an IP-only application with graphical user interface and video preview. It is responsible for video grabbing (either from a webcam or from a local video file) and delivers this video stream to the second application. Within this second application, the FoG concept and needed management functions are implemented. The software is able to simulate a FoG based network and its communication in real-time. It has an additional interface to real (IP) networks. The interface is used to convert the video stream from IP to FoG. Within the software, a FoG based video viewer shows the received stream. Via several verbose graphical outputs it is possible to observe all activities in the FoG network. Based on this, our software shows the key features of FoG.

First, the video display is requesting the video from the video source just via best effort, like in today's Internet. This is marked by "1." in Figure 1. The received video is displayed as a separate widget within the software.

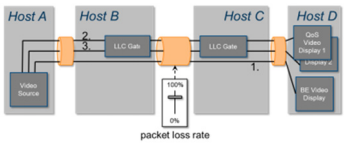

Figure 1: scenario overview

Second, the video is transmitted with the explicit application requirement "maximum loss rate limited to explicit value". In this case special gates ("LLC Gates") are automatically created by the network to implement transmission error detection and retransmission of packets. For demonstrating purposes, the packet loss rate can be adjusted by a graphical control element in the software. This is shown in Figure 1 as slider below the link between host B and C. With the help of the slider the packet loss rate can be set dynamically. The variation of packet loss will lead to obvious differences in the video transmission quality of case one and two. Coding artifacts and presentation stops occur in the video presentation of case one. In case two the video has only an additional delay which is caused by the "LLC Gates" and their applied retransmission of packets. The different handling of both transmissions based on the requirements shows the first contribution.

Third, an additional video transmission is started with the same requirements as in the previous one. The network will now re-use the created gates from the second scenario. This includes all QoS enabled functional blocks for unicast transmission as well as video encoding and decoding blocks. The whole process of finding and using existing reusable blocks will be visible to the audience in the graphical display showing the network topology.

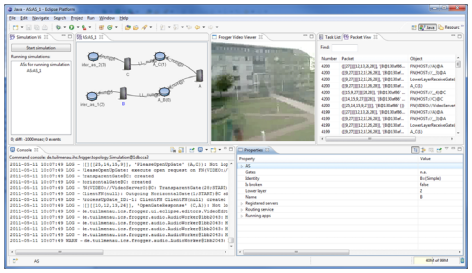

Figure 2: demonstrator screenshot

The graphical user interface of the FoG prototype, depicted in Figure 2, enables the user to observe every process in the network graphically. It consist of (from the top left to the bottom right in Figure 2) an AS overview, a network topology presentation, several video stream outputs, a packet view, a debug console with different log levels and a property view which is able to show properties and attributes of the currently selected item of the network. For example, this can be a physical host or link, a logical function block or a running application.

IV. REFERENCES

[1] D. Martin, L. Volker, M. Zitterbart: A flexible framework for Future Future Internet design, assessment, and operation, Computer Networks, Volume 55, Issue 4, Special Issue on Architectures and Protocols for the Future Internet, March 2011.

[2] B. Reuther, D. Henrici: A model for service-oriented communication systems, Journal of Systems Architecture, 2008.

[3] F. Liers, T. Volkert, A. Mitschele-Thiel: A Flexible Abstraction for the Future Internet, 8th Würzburg Workshop on IP (EuroView), Germany, Würzburg, August 2008.

[4] D. R. Cheriton: Sirpent: a high-performance internetworking approach. In proceedings of ACM SIGCOMM '89: Symposium proceedings on Communications architectures & protocols, 1989.

[5] P. B. Godfey, I. Ganichev, S. Shenker, I. Stoica: Pathlet Routing, In proceedings of SIGCOMM 2009, August 2009.