Distributed Emulation of IP Networks

Florian Baumgartner and Torsten Braun

Institute of Computer Science and Applied Mathematics
University of Berne
Neubrückstrasse 10, CH-3012 Berne, Switzerland

Abstract

For the development of communication services the setup of experimental networks with a sufficient size is a crucial element. Unfortunately the availability of the required equipment like routers and hosts is very limited and expensive. On the other hand simulations often lack interoperability to real systems and scalability. This paper presents an approach between these two alternatives and allows the establishment of test beds on a cluster of computers with full interoperability to the real world.

Key words: Distributed Emulation, Internet, Distance Learning, Performance Evaluation
PACS: 01.50.Ht, 01.50.My, 89.20.Ff, 89.20.Hh

1 Introduction

A usual problem in computer networks research is the demand for setting up test beds of sufficient size and complexity to evaluate new concepts or to have a suitable platform during the development of new algorithms and services.

To cope with these problems specific simulators to run the desired tests can be implemented and also more generic network simulators like ns [1] or OpNet [2] are available. Unfortunately all these network simulators have their drawbacks. Simulations often cover only a single aspect of the problem and ignore possible side effects. Especially for application oriented research simulators often lack complexity. Realistic traffic sources and sinks as well as fully functional protocol implementations are often missing.

Email addresses: baumgart@iam.unibe.ch (Florian Baumgartner), braun@iam.unibe.ch (Torsten Braun).

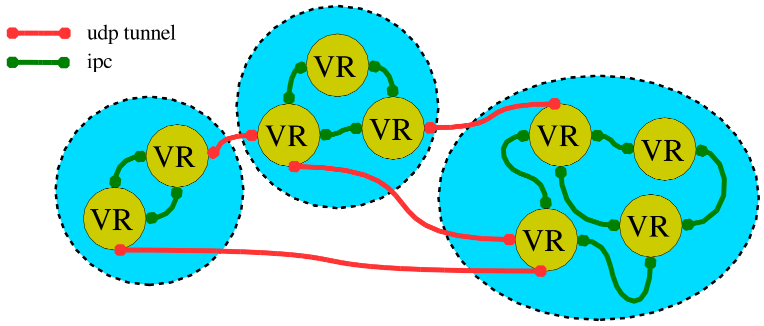

Fig. 1. Distributing Virtual Routers to hosts with connections by Unix IPC and UDP tunnels

We propose to combine real networks and emulated topologies. This allows to implement new functionalities on a real system and to emulate the network topology behind. Furthermore a reasonable emulation approach can also eliminate the need for kernel level implementations by offering a more convenient programming environment. Because of this, we propose an approach to combine real networks with emulated ones, using real devices wherever needed and emulating the rest.

2 Virtual Routers

The basic idea of combining real hardware with an emulated topology is shown in figure 1. Each computer can run multiple Virtual Routers (VRs). These VRs can be connected to other VRs running on the same or on an other computer. To integrate such a topology into a real network a VR can also be connected to the local computer by a so called softlink device, which emulates a network interface and allows to forward traffic from the real world to a VR. So it is possible to route traffic of a computer to and through an emulated topology. Since the number of connections between a VR and the real network is not limited one host can emulate a topology and act as multiple traffic sources and sinks simultaneously.

Figure 2 shows the design of a VR. The central forwarding mechanism works according to standard routing rules, but has been extended to allow routing decisions by source addresses, port numbers, protocol fields and Type of Service (ToS) values. The central forwarder is programmable and allows the processing of specific streams at higher layers, e.g. processing audio or video streams (e.g transcoding) or monitor traffic for debugging or control purposes [3].

Since the implementation allows loading and unloading of additional components during runtime the memory requirements are rather low. Loading and unloading also allows the exchange of complete router modules without any

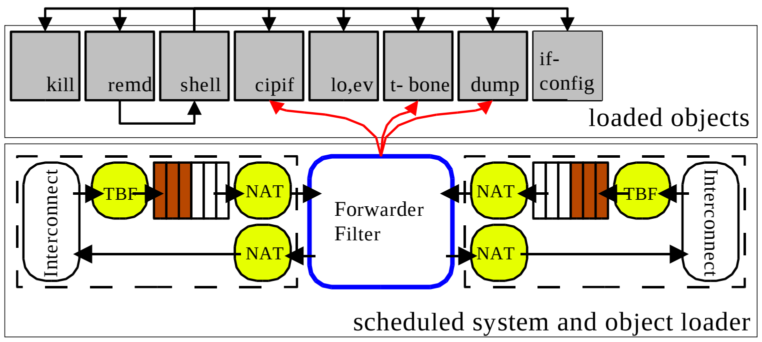

Fig. 2. The Architecture of a Virtual Router

interrupt of the emulated network. In addition to several configuration modules and commands (ev, lo, ifconfig, ...) a shell and a component allowing to control the router via telnet are available. The configuration command syntax is similar to a standard Unix system.

The main work regarding IP processing is assigned to the interface components. Figure 2 shows two of them. Each interface handles the connection to other interfaces. Three different connections types gave been implemented.

connections to a softlink device: This type of connection is used to enable the communication between a virtual router and a real network. The softlink Linux kernel module emulates an ethernet device and offers simple access for an application (e.g. a VR) to packets being forwarded to this emulated ethernet device by the Linux host. It has to be mentioned here, that there is no difference for the Linux system between a normal ethernet device and the emulated one. The VR uses this softlink device to read packets from the real network and also to send packets back to the network.

connections between VRs via FIFOs: This connection type is used to connect two VRs running on the same computer. It allows to forward packets from one VR to another and is simply based based on Unix pipes between two processes. As Unix pipes are simplex two pipes are used for each connection.

connections between VRs via tunnels: tunneling based on UDP is used to exchange apckets between VRs on different computers. IP packets to be forwarded to a VR are encapsulated within an UDP packet and sent to a specific port to the remote host. The opposite VR has to be configured to listen to this UDP port and read packets from there. Like any other connection it must be duplex, so the VR's interfaces on both sides of the connection have to receive and transmit UDP packets.

Received data is processed by an IP network address translation unit (NAT). This allows to force the routing of packets through an emulated topology by modifying the destination/source address pair within a VR. So it is possible to set up large networks on a single computer.

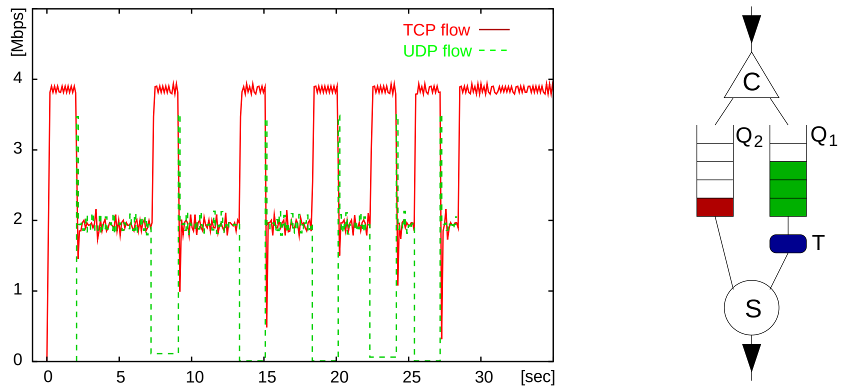

Fig. 3. Bandwidth Measurements: TCP flow protection through a VR by appropriate queuing system.

Data for transmission is also first processed by a NAT and then put to a queuing system. Since the VR is not bound to a physical bandwidth limit of a network, the speed of a VR interface has to be limited by a token bucket filter.

Because of it's flexibility the queuing system is one of the most complex parts of the interface. It consists of a set of components like queues, filters, shapers, schedulers, which can be combined during runtime. The current implementation offers a set of components like a generic classifier, a token bucket filter, a drop tail queue, a random early detection queue (RED) [4]), a weighted fair queuing (WFQ) scheduler, a simple round robin scheduler, and a priority round robin (PRR) scheduler. To allow the establishment of Differentiated Services networks some additional components were developed like a RED queue with three drop precedences (TRIO), a special marker for Differentiated Services and a Priority Weighted Fair Queuing (PWFQ) scheduler for the implementation of Expedited [5] and Assured Forwarding [6].

This queuing system can be used to set up Differentiated Services networks or to establish mechanisms to protect certain flows against other ones. Figure 3 shows a setup which favors a congestion avoiding TCP flow over aggressive UDP traffic. Without an appropriate queuing system the UDP traffic would suppress TCP traffic completely. The queuing system (right diagram) is set up to send any traffic except TCP to queue Q_1 which is limited to 2 Mbps by the token bucket filter T. TCP packets are put to queue Q_2. The scheduler S is configured to favor the non TCP packets of queue Q_1, which is limited to 2 Mbps by T. So the queue Q_2 will get the difference between the available link bandwidth of 4 Mbps and the maximum bandwidth of Q_1/T of 2 Mbps.

The graph on figure 3 shows the behavior of UDP and TCP bandwidths. The UDP traffic is sent in bursts of decreasing intervals, reducing the TCP flow to the guaranteed amount of 2 Mbps. Without any UDP traffic the TCP flow gets the full link bandwidth of 4 Mbps.

3 Distribution of VRs

The idea of an VR is to emulate a single router, not an end system. Real end systems are thought to be used as traffic sources and sinks.

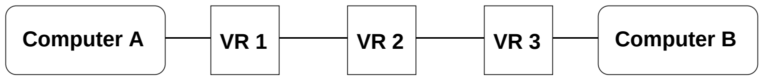

Fig. 4. Two end systems connected over three Virtual Routers

The diagram above shows a typical simple setup of two computers (A,B) connected via three VRs (1,2,3), allowing it to send traffic (UDP, TCP, ...) from computer A to computer B and vice versa. To allow a connection to a computer's network layer (see section 2) a softlink device has to be used. This requires a VR running on the same computer, on which the softlink device is installed. Nevertheless several setups are possible.

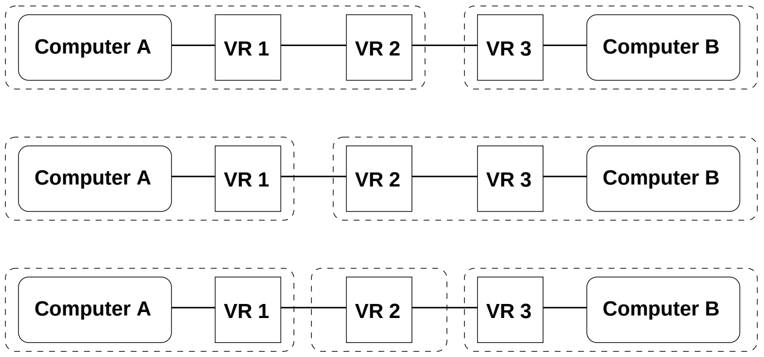

Fig. 5. Distributing VRs to end systems

Diagram 5 shows how the VRs may be distributed. On each host, which has to be connected to a VR via Softlink device a VR has to be started. VR 2 might be placed either on computer A or B or on an additional computer between A and B.

A concrete setup would depend on the available processing power of the computers and the bandwidth of the network, the computers are connected with.

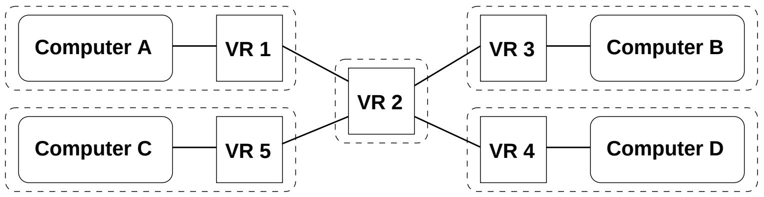

Fig. 6. Increasing the number of end systems

The idea of separate end systems for each traffic source or sink would cause a significant demand for computers in a topology of a reasonable size as can be seen on figure 6. Fortunately at least under Unix operating systems the IP addresses of a host are bound to the network interfaces (or logical interfaces) within that host. As mentioned in section 2 the Softlink device is an emulation of a normal network interface and like other interfaces it it owns an IP address. As it is possible to create several softlink devices on a computer (up to 256), a computer may appear at different points in a topology as source and sink as can be seen on figure 7.

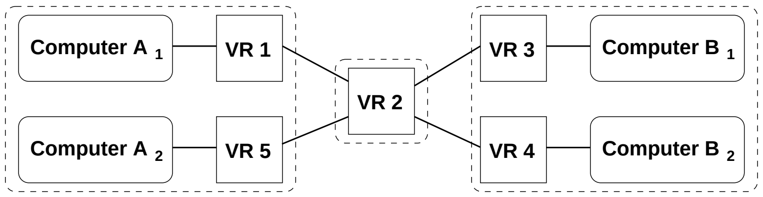

Fig. 7. using multiple softlink devices

4 Traffic Measurements

Although the original idea of a Virtual Router was to offer a platform for the development and evaluation of distributed mechanisms like network management and Quality of Service routing, the architecture offers also a suitable testbed for traffic measurements. Since the VR has to process packets in real time the number of routers emulated on a host and the maximum allowed bandwidth is of course limited. The performance may be increased significantly by multiprocessor computers.

4.1 Distribution and Packet Delay

The most problematic issue during network emulation in a distributed environment is the time a packet is delayed during forwarding and transmission. Since the packet forwarding requires real processing and is not just simulated, the lower delay bound per hop is limited by the available processing power as well as by the available bandwidth and delay of the underlying network.

To measure the delay for an increasing number of routers a chain of 16 VRs was

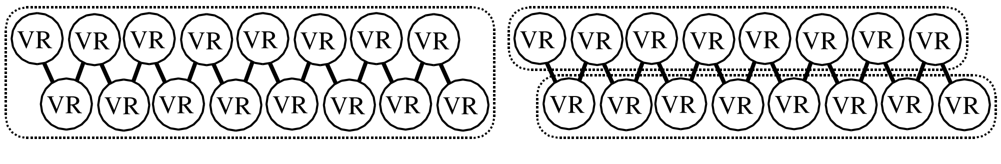

Fig. 8. 16 VRs on on one and distributed to onetwo hosts

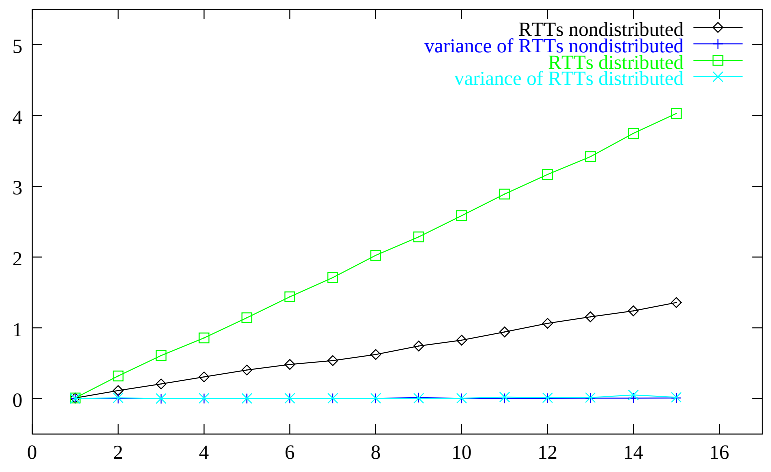

Fig. 9. RTTs in an unloaded VR network

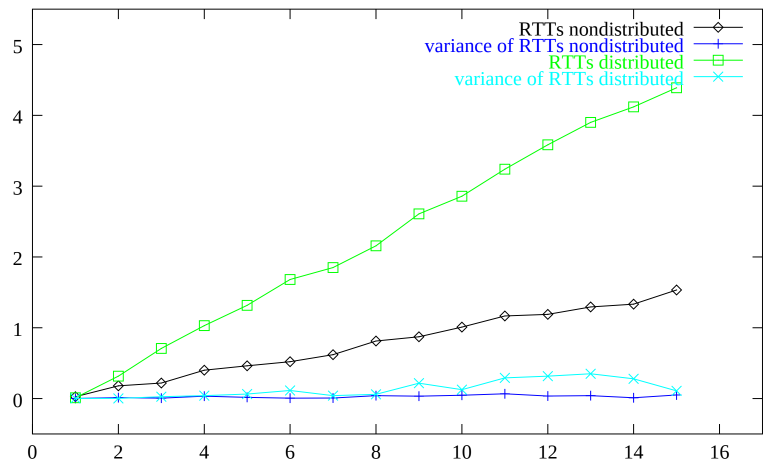

Fig. 10. RTTs in a a loaded VR network

set up on one[1] and on two computers[2] (see Figure 8). To measure the round trip time (RTT) a number of pings have been sent to different routers in this router chains. These experiments have been performed on an unloaded network and with additional bursty UDP traffic up to 50 % of the link bandwidth.

The graphs of the figures 9 and 10 show RTTs of pings for an increasing number of hops for the local and the distributed setup. The left diagram shows the results without additional traffic in the network, on the right diagram additional UDP traffic has been sent over the VRs. All graphs show the expected linear correlation between the number of hops and the RTTs. The RTTs in the distributed emulation are much higher than in the single computer setup. Since each VR has been connected to two VRs on the other computer, the packets suffer additional delay by the UDP tunnels being used to connect VRs on different computers. The changing load of the UDP tunnels causes also a

[1] dual processor 800 MHz PentiumIII, running Linux

[2] dual processor 800 MHz PentiumIII and a single processor 400 MHz PentiumII, both running Linux

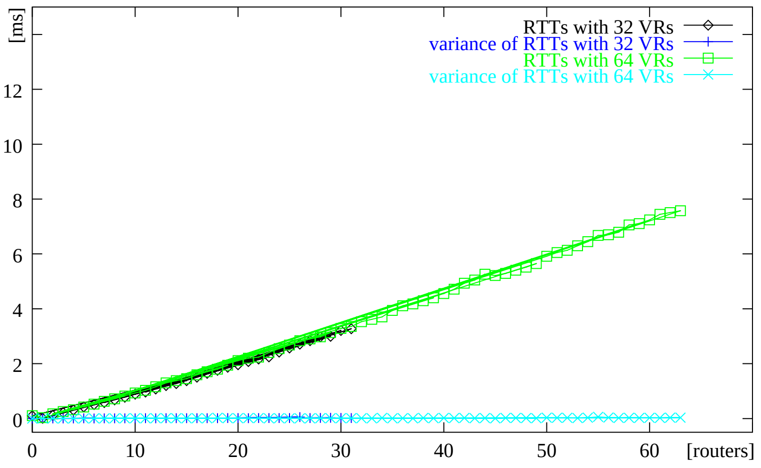

Fig. 11. RTTs with different numbers of VR entities in an unloaded network

higher RTT variance.

4.2 Topology Size and Packet Delay

Another interesting question is the impact of the number VR entities running on a computer to the RTTs. For this experiment 32 and 64 routers were set up in a chain on a single computer, in order to evaluate the influence of a distributed emulation. Figures 11 shows that as long as there is no additional traffic on the computer the RTTs with 32 VRs correspond to the values with 64 routers.

The situation looks different, when additional traffic is sent over the router chain as it can be seen on figure 12. The RTTs increase and also the variance of RTTs are significantly higher. On the other hand the graph still shows an linear increase of RTTs with the number of hops, the ICMP packets have to pass.

The measurement of RTTs gives a good impression about the impact of distribution to packet delay. Another important issue is the reaction of more delay and packet loss sensible flows to a VR topology. To answer this question we evaluate the behaviour of TCP flows transmitted through a topology emulation.

4.3 TCP Behaviour

TCP applies congestion control mechanisms to avoid packet loss by adapting to the available bandwidth. Because of that TCP is much more sensible to

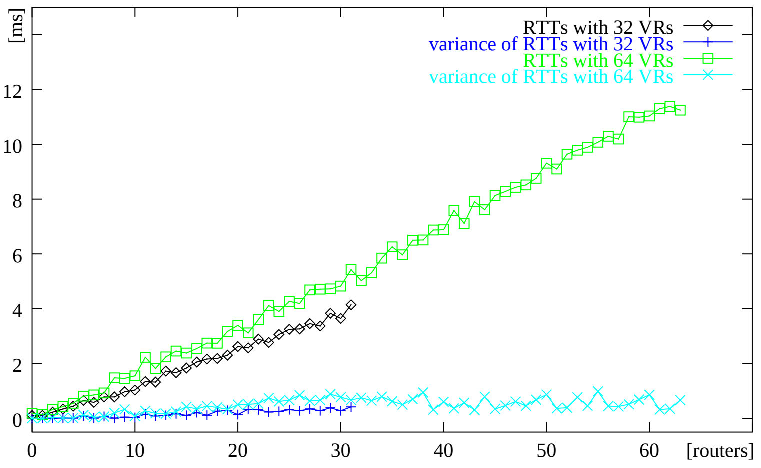

Fig. 12. RTTs with different numbers of VR entities in a loaded net

changes of delay and packet loss, than protocols such as UDP are. Figure 13 shows the results of an experiment, in which a best effort TCP flow has been sent through the distributed topology of 64 routers. Each router has been set up with a drop tail queue for each network interface and an interface speed of 1 Mbps. The diagram shows the according queue length of the first, the second and the last VR in the chain. Additionally, the bandwidth of the TCP flow and the RTT through the topology, which has been measured by sending pings to the TCP's destination is presented.

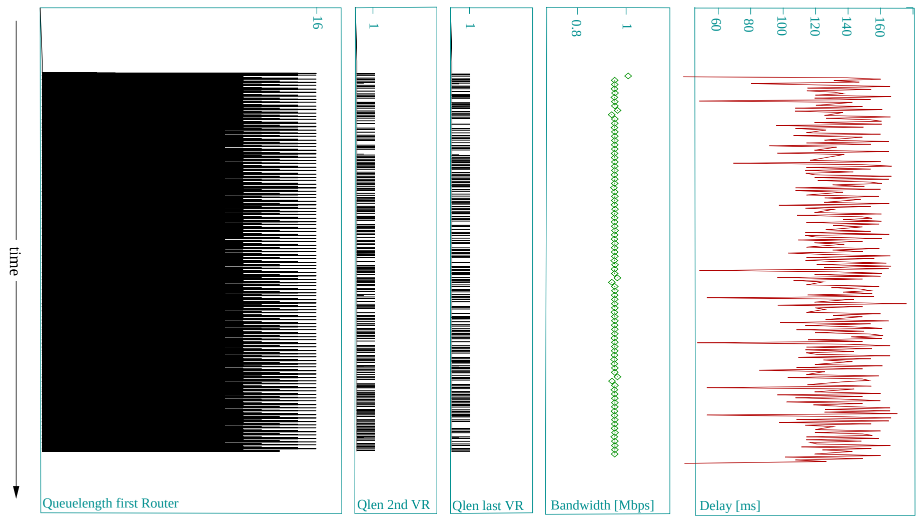

Fig. 13. Queuelengths, Bandwidth and Delay in a chain of eight VRs, transmitting a TCP flow

The graphs on figure 13 show that the first queue is filled up with packets, while the queues of the following routers are empty or store only one packet. Since the first VR is a bottleneck for the TCP flow, the droptail queue of this VR is filled until packets are dropped and TCP reduces it's transmission rate to $\leq$ 1 Mbps. The following routers are capable to transport this amount of traffic, and therefore their queues remain empty. The filled up queue also increases the delay of the packets, as can be seen on the right diagram. The variances of the queue lengths and the delay are caused by the congestio avoiding mechanism of TCP.

The same setup was used to study the bandwidth sharing between two best effort tcp flows. Boths flows have been sent through the emulated topology. Figure 14 shows the bandwidths of the flows. Flow 1 started a little bit earlier, until it has to reduce its transmission rate due to congestion.

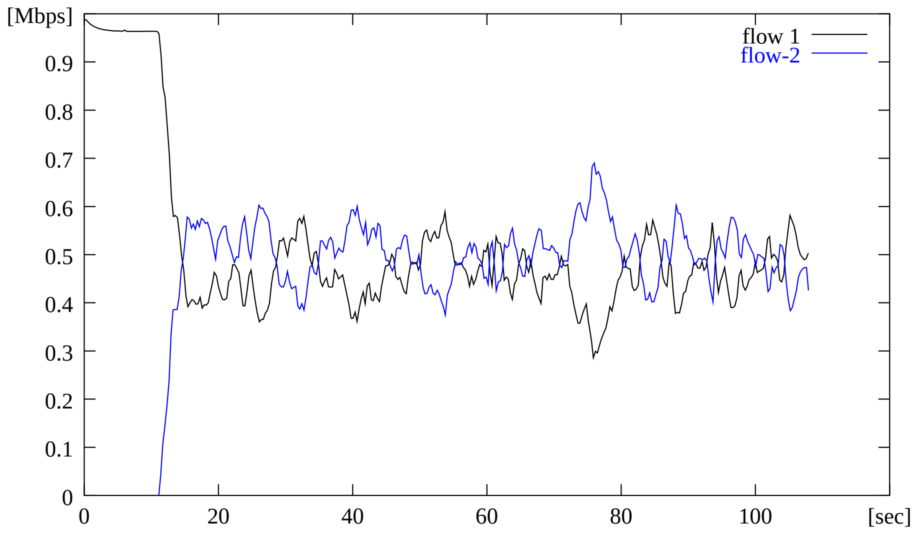

Fig. 14. queuelengths, bandwidth and delay in a chain of VRs, carrying a TCP flow

5 Summary and Outlook

The approach to emulate network topologies for the development and evaluation of network technologies has proven to be very useful. Especially when an environment for the rapid implementation of a new idea is needed the concept of virtual routers shows it strengths. The possibility to integrate real hosts/routers into a larger emulated scenario and the more convenient programming environment are powerful tools during implementation and evaluation.

Currently ongoing work includes the development of a graphical configuration tool to interactively setup virtual topologies and the support for intelligent network management devices.

Acknowledgements

The work described in this paper is part of the work done in the project 'Quality of Service support for the Internet Based on Intelligent Network Elements' funded by the Swiss National Science Foundation (project no 2100-055789.98/1). Parts of the implementation platform were funded by the SNF R Equip project no. 2160-53299.98/1.

References

[1] Ucb/lbnl/vint network simulator - ns (version 2). URL: http://www-mash.CS.Berkeley.EDU/ns/.

[2] Opnet modeler. URL: http://www.mil3.com.

[3] Manuel Gunter and Torsten Braun. Service delivery control with mobile code. *IFIP Conference on Intelligence in Networks (SmartNet)*, September 2000.

[4] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, August 1993.

[5] Van Jacobson, K. Nichols, and K. Poduri. An expedited forwarding phb. Request for Comments 2598, June 1999.

[6] Juha Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured forwarding phb group. Request for Comments 2597, June 1999.

[7] Florian Baumgartner and Torsten Braun. Virtual routers: A novel approach for qos performance evaluation. In Jon Crowcroft, James Roberts, and Smirnov Mikhail, editors, *Quality of Future Internet Services, First COST 263 International Workshop. QofIS*, Lecture Notes in Computer Science, pages 336–347, Berlin, Germany, September 2000. Springer. ISBN 3-540-41076-7.

[8] Florian Baumgartner and Torsten Braun. Quality of service and active networking on virtual router topologies. In Hiroshi Yasuda, editor, *Active Networks, Second International Working Conference, IWAN*, Lecture Notes in Computer Science, pages 211–224, Tokio, Japan, October 2000. Springer. ISBN 3-540-41179-8.