Virtual Routers: A Tool for Emulating IP Routers

Florian Baumgartner[1], Torsten Braun[2] and Bharat Bhargava[1]
baumgart@cs.purdue.edu, braun@iam.unibe.ch, bb@cs.purdue.edu

[1]Center for Education and Research in Information Assurance and Security (CERIAS)
and Department of Computer Sciences, Purdue University

[2]Institute of Computer Science and Applied Mathematics
University of Bern, Switzerland

Abstract

Setting up experimental networks of a sufficient size is a crucial element for the development of communication services. Unfortunately, the required equipment, like routers and hosts, is expensive and its availability is limited. On the other hand, simulations often lack interoperability to real systems and scalability, which limits the scope and the validity of their results. Therefore, an intermediate approach between these two alternatives that allows for setting up testbeds on a cluster of computers is needed. This paper presents an intermediate approach based on the emulation of IP routers and evaluates the concept. In a first set of experiments the impact of various parameters on the packet delay was investigated, while further experiments compare the performance of Differentiated Services run on the network emulator with the results obtained by the well known network simulator ns.

1 Introduction

The strengths and drawbacks of network simulators, like the ns network simulator [9] or Opnet [10], in general lie in the use of a mathematical model to simulate a network, a node or a link between two nodes. Since there is no relationship between the real time and the time ns uses internally, ns can be used to run huge simulations with thousands of nodes and links. The simulation simply will last longer, but is nevertheless mathematically correct. The use of a such a model includes a rather abstract view of a network consisting of nodes and links. These properties are sufficient to simulate the traffic flow of thousands of nodes within a huge network. On the other hand, it is rather inconvenient if the components of the testbed have to closely imitate real devices, and an integration of an emulated network with real network components is favored. Such a combination of real devices like routers and end systems, with an emulated topology has several advantages for the set up of testbeds and the development of new concepts.

The idea to emulate network devices is not new. The Emulab project [3] uses several hundred machines to set up experimental networks. The experiments can be set up remotely, using scripts compatible with the ns network simulator. While Emulab is a rather brute force approach, Wang [11] proposes to apply an address mapping scheme and to forward packets repeatedly through the kernel of the same host. Even if this is rather complicated and does not provide a real emulation of an Internet router, it is especially capable to emulate the impact of multiple instances of a specific host on forwarded traffic. Another approach is to add special kernel components applying special characteristics of a WAN router, like certain link delays, to a single PC [8]. This allows to emulate a set of WAN routers within a laboratory LAN.

A network emulation allows the use of real end systems and applications. When using a simulator, these applications have to be redesigned, which usually is possible only if a proper description of the internals of an application is available. A network emulator connected to a real network can be used with any type of application without any changes to the application.

New traffic conditioning components can be implemented and evaluated in a convenient emulation environment. Writing of new kernel code, and especially its evaluation, is usually rather complicated and time consuming. Nevertheless in the emulation approach the new components can be evaluated directly with real applications.

Network emulation can also be used to test applications for real networks. Before setting up a set of IP routers or other network devices, similar scenarios may be set up using emulated devices. In contrast to simulations emulation

Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02)
0742-1303/02 $17.00 © 2002 IEEE

can not only be used for testing the set up with simulated sources but with real end systems also.

The front-end of an emulation of a single device can be like the one of a real network device. This might be a drawback during the set up of large topologies, yet it is a big advantage if a typical test network with a few dozens routers has to be set up. The use of a well known front-end also simplifies the handling of the emulator for new users and in environments with emulated and real devices.

Some of these tasks can also be accomplished with the ns network simulator. The simulator can be connected to real networks and real packets are forwarded through the simulator. On one hand, this allows for the use of real end systems, but it also limits the size of the simulated network, since now also the simulator has to process traffic in real time. Also, the simulator is still a kind of monolithic block, with a completely different user front-end than in the end system. Using a single program to simulate a network also complicates the modification of specific nodes, while a more distributed approach (as used for VRs) allows to replace or update single routers more easily.

Another disadvantage of the ns concept are the differences between the configurations of the simulator and real devices. For a complete emulation of an Internet router it is not sufficient to apply similar packet treatments, but also to provide similar front-ends and configuration concepts.

A combination of real networks and emulated topologies allows to implement new functionalities on a real system and emulate the network topology. Furthermore, a reasonable emulation approach can also eliminate the need for kernel level implementations by offering a more convenient programming environment. To capitalize on these benefits, we propose an approach that combines real systems with emulations, 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. The softlink device provides a special network device allowing a computer to forward packets to a Virtual Router. For the computer, the softlink device looks like a standard Ethernet interface and packets can be forwarded to that interface like to any other network interface. Of course a computer might use several of these softlink devices to set up network connections to several virtual topologies. This is why one host can emulate a topology

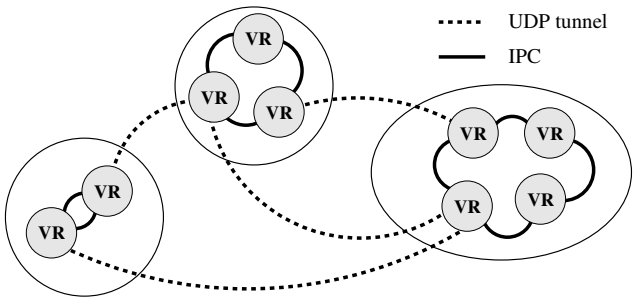

Figure 1. Distribution of Virtual Routers to a set of computers.

Table 1. Interface configuration and route setup with the VR command line interface.

```
MicroVR Shell (built 17-3-02-17-00)

> ifconfig add if0 10.1.1.1
interface if0 added (10.1.1.1 255.255.255.0)
> route add 10.1.1.0/24 if0
route 10.0.0.0 netmask 255.255.255.0 to if0 added
> route add default if0
route default to if0 added
> route
route 10.1.1.0/255.255.255.0 to if0(0)
route default to if0(0)
>
```

and act as multiple traffic sources and sinks simultaneously.

2.1 Virtual Router Architecture

Figure 2 shows the design of a VR. Virtual Routers run independently from the computer's network layer, providing their own routing and traffic conditioning components. The central forwarding mechanism uses 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. This simplifies the implementation of certain daemons, and is also a great facility to process streams during transmission (e.g. video or audio data) or to generate traffic dumps for debugging purposes.

The implementation allows loading and unloading of components during run time, without interrupting the network emulation. While this allows to update certain functions, unloading components at run time also minimizes the memory consumption of a standard Virtual Router.

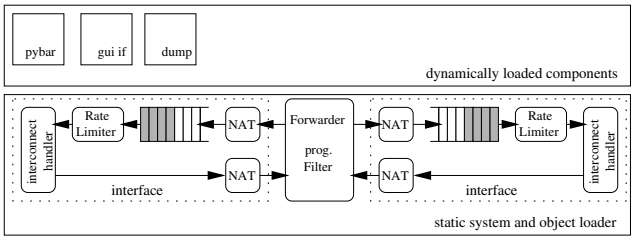

Figure 2. Virtual Router Architecture.

A Virtual Router provides a command line interface providing a command set comparable to a standard Unix system (ifconfig, route, etc.). Table 1 shows the commands required to set up an interface and to configure the appropriate routes. The command line can be used from the console but can also be accessed via telnet. Especially in larger Virtual Router topologies, this makes it easy to connect to a specific Virtual Router and change its configuration.

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

Connections to a softlink device: This type of a connection is used to allow for communication between a Virtual Router and a real network as can be seen in Figure 3. Since the softlink device looks like a standard network interface to the linux operating system, no differences between the virtual and a real network are visible for programs running on this computer.

Connections between VRs via IPC: This connection type is used to connect two VRs running on the same computer. It allows to forward packets from one VR to another. This type of connection is based on Unix IPC and forwards packets between two Virtual Routers running as separate processes.

Connections between VRs via UDP tunnels: Tunneling based on UDP is used to exchange packets between VRs on different computers. IP packets to be forwarded to a VR on another computer are encapsulated within an UDP packet and sent to a specific port to the remote host. The receiving VR has to be configured to listen to this UDP port and read packets from there. These connection have to be – as any other – duplex. Each VR has to listen to a port for UDP packets being received, and has to send packets to a port on a remote machine.

The connections between the Virtual Routers act as a kind of link layer for the Virtual Router network.

Received data are processed by an IP network address translation unit (NAT). As the destination/source address pair can be modified within a VR, hosts may forward packets to the emulated topology in fact addressing themselves without noticing this, thus allowing to set up large networks on a single computer. While no address translation is necessary in most configurations, this feature is useful to allow certain experiments also in single host setups.

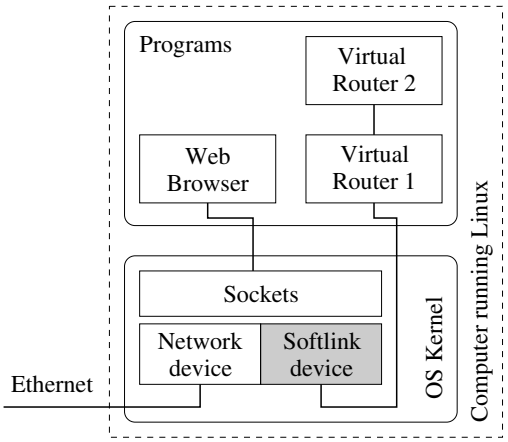

Figure 3. The softlink network device connects the Virtual Router to the computer's network.

Data for transmission is also first processed by a NAT and then passed to a queuing system. Since a VR is not bound by a physical bandwidth limit of a network, the speed of a VR interface has to be limited explicitly. The rate limit for each VR interface can be configured during run time, in order to vary the available interface speed.

3 Packet Delay

3.1 Link and Queuing Delay

A Virtual Router, in contrast to a network simulator, processes packets in real time. Therefore, changes to the processing speed or the load on the computer will directly affect the packet delay. Additionally, Virtual Routers provide no possibility to reduce a specific link delay below the "natural" limit. Mechanisms like specific queues can be applied to increase the link delay, but there is no possibility to decrease it below the value defined by the computer's processing power.

This is why it is important to keep link delays as small and as constant as possible. Table 2 provides an example of current delays on the Internet. The values were obtained by measuring the round trip time within the research networks of Germany (D), Switzerland (CH), two Virtual Router networks and in a Linux based Differentiated Services laboratory network at the University of Berne.

Table 2. Comparison of delays for different destination/networks.

	hops	delay [ms]	per hop [ms]	congestion
D	15	38	2.53	unknown
CH	8	4.5	0.56	small
VR	8	2.1	0.27	no
VR	16	4.3	0.27	no
Linux-DS	8	75.46	9.433	heavy

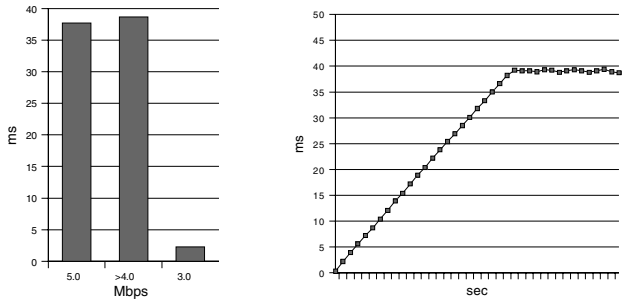

Figure 4. The graphs show the delays caused by the FIFO queue of a 4.0 Mbps link for different bandwidths.

Since the measurements within the Virtual Router networks were performed in an *unloaded* network (i.e., network with a low traffic) the delay was caused only by the link delay or the packet forwarding within the Virtual Routers. However, compared to the delay values of the other measurements in Table 2 the delays are reasonably small. The delay values for the heavy congested Linux network demonstrate the impact of queuing delay, causing obviously much more delay than the link layer.

Similar to the Linux routers, also the Virtual Router queuing system is the main cause of a packet's delay. To demonstrate that impact, Figure 4 shows the packet delay caused by the interface's FIFO queue for different sending rates. The interface speed was 4 Mbps. While the delay is very small as long as the queue stays empty, it increases drastically if the incoming bandwidth approaches or exceeds the interface capacity. The graph on the left hand side shows the considerable delays for 5 Mbps and for slightly more than 4 Mbps. In both cases the router's FIFO queue is full and causes a packet delay of approximately 40 milliseconds. The slightly higher delay for the rate of just more than 4 Mbps is due to some synchronization of the token bucket filter limiting the Virtual Router's bandwidth, and the token bucket filter limiting the sender's transmission rate to just more than 4 Mbps.

The right hand side graph in Figure 4 shows the increase of the queue length over time for the 4.x Mbps sender. Since only slightly more packets are sent than the queue is able to transmit, the length of the queue, and therefore the delay, increases slowly while the experiment proceeds. Once the maximum queue length is reached and the queue started to drop packets, the delay remains constant.

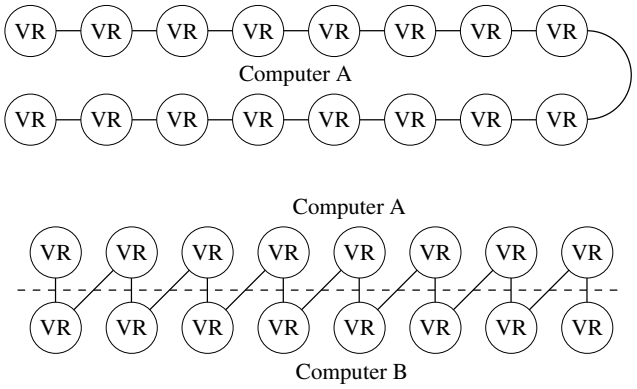

Figure 5. A chain of 16 Virtual Routers on a single computer and distributed to two computers.

The delay caused by the a queue can be easily calculated

$$d_{queue} = \frac{Q_{pkts} \cdot p_{avg}}{b_{bytes}}$$

where Q_{pkts} is the queue length in packets, p_{avg} the average size of transmitted packets in bytes, and b is the bandwidth of the outgoing link in bytes per second. For a queue length of 20 packets, an average packet size of 1000 bytes, and a link bandwidth of 4.0 Mbps, this results in a theoretical delay of ≈ 40 ms.

The experiments show comparable delays within real and Virtual Router networks and also demonstrate the impact of queuing delays. Even if the minimal link delay of Virtual Routers is determined by the processing speed, it is rather low and can easily be increased by additional buffering.

The delay measurements presented here give an overview and illustrate the delays caused by the connections (the link layer) between Virtual Routers, and emphasize the impact of queuing delay to packet forwarding.

3.2 Delay in Distributed and non-Distributed Topologies

Besides the general increase in the link delay due to additional communication overhead for certain connections, the current workload of the computer affects the forwarding delay. Therefore, the number of Virtual Routers involved in the processing of a specific packet might change the per hop delay.

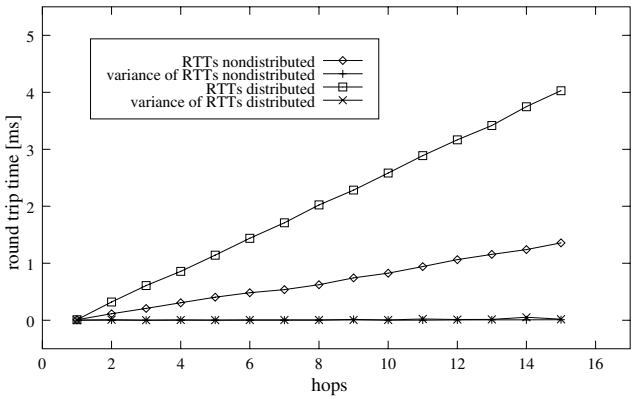

Figure 6. Round Trip Time within the Virtual Routers networks of Figure 5) without additional load.

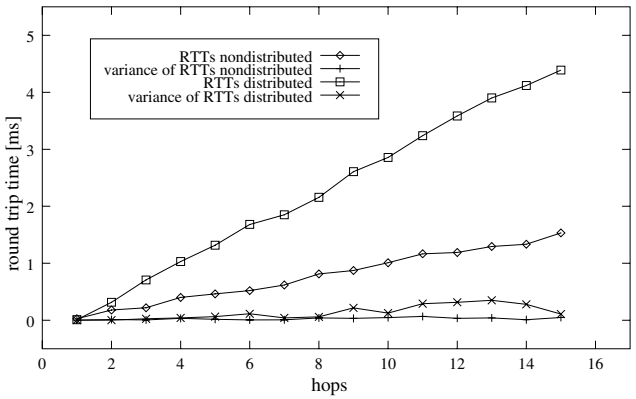

Figure 7. Round Trip Time through a Virtual Router network of Figure 5) with additional UDP traffic to add some processing load to the routers.

To measure this delay for an increasing number of routers and for different connections between Virtual Routers, a chain of 16 Virtual Routers was set up on one[1] and on two computers[2] as shown in Figure 5. To measure the round trip time (RTT), a number of pings was sent to different routers in these router chains.

To study the impact of heavier workloads, these experiments were performed for two scenarios. In the first set up, the Virtual Router network was unloaded, while in the second configuration an additional UDP traffic caused some additional processing load on the Virtual Routers. Since the UDP traffic causes additional load only, but should not add significant queuing delays to the echo request/echo reply packets, the additional traffic does not exceed 50% of the link bandwidth.

The graphs in Figures 6 and 7 show the average round trip times and variances for 100 pings for an increasing number of hops for the local and the distributed configuration. Additionally to the time, the graphs also display the according variances.

In the scenario with no additional workload (Figure 6), a straight linear correlation between the number of hops and the round trip time, with a very small variance, can be seen. The round trip times for the distributed case are higher than the RTT measured in the single computer set up, but also increase linearly. Obviously, Virtual Routers cause equal link delays during packet forwarding and packet transport for a certain set up.

In the distributed scenario, packets have to be additionally encapsulated into UDP packets, and forwarded over a LAN to the computer hosting the other Virtual Router. Additional packet processing and especially the transport over the LAN cause additional delay.

As can bee seen in Figure 7, the correlation between the hop count and the round trip time stays linear, even if the workload on the Virtual Routers (and of the whole computer) changes due to additional traffic. The round trip times are slightly increasing, caused by the need to transport additional packets. Even if the additional workload does not cause significant queue lengths, it causes single echo request or reply packets to be delayed increasing the average round trip time. Accordingly the graph in Figure 7 shows an increase in variance, albeit small.

This linear correlation between the number of hops and the round trip times for a certain connection type between the routers is important, since it shows that the individual Virtual Routers run rather independently from each other, causing similar link and packet forwarding delays and not interfering with other routers on the same computer.

Comparing the delays in the different set ups, the constant per hop delays for the different scenarios are discovered, even if the delay between the scenarios vary. Since Virtual Routers depend on the processing power, the workload of the computer and the speed of the local area network, an absolute guarantee for fixed delays is not possible. However, since the delays and their variances are small. this limitation is acceptable.

Furthermore additional buffering between the Virtual Routers can increase the delay between Virtual Routers to a specific value and might also be used to equalize certain variances.

[1] Dual processor 800 MHz Pentium III, running Linux.

[2] Dual processor 800 MHz Pentium III, and a single processor 400 MHz Pentium II, both running Linux.

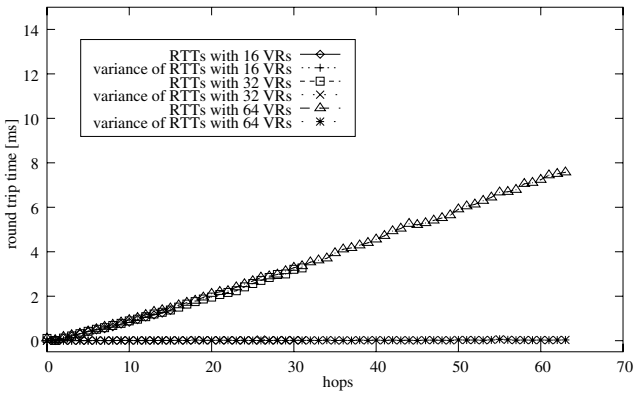

Figure 8. Identical Round Trip Times through a Virtual Router network for different topology sizes.

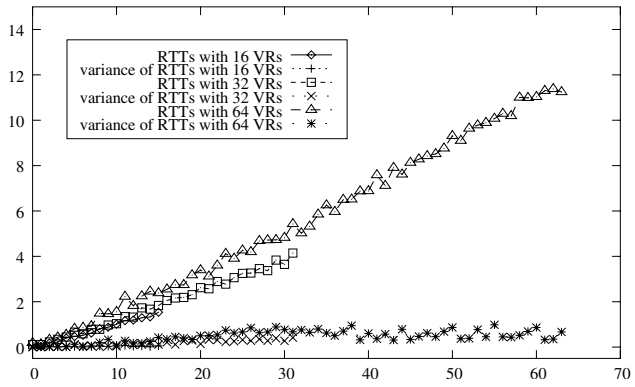

Figure 9. Round Trip Times through a Virtual Router network without additional load.

3.3 Topology Size and Delay

So far the described experiments included a constant number of Virtual Routers. To examine the impact of the topology size on the packet forwarding and link delay, similar router chains were set up but this time with chains of 16, 32 and 64 routers. As before, the round trip times were measured first within an unloaded environment. Only the single computer setup, was evaluated (all VR's on one computer).

Figure 8 shows the increase of the round trip time for the 16, the 32 and the 64 router chain including the variances. The three graphs showing the round trip times match perfectly (the graphs overlap) and show nearly no variances. Obviously, the number of running instances does not significantly affect the link and the packet forwarding delay.

Similarly to the previous experiments, now additional traffic was sent over the network and the round trip times were measured again. Figure 9 shows the. In contrast to the experiment before, the larger topology causes a bigger per hop delay than the 16- and the 32-router network. Additionally a bigger workload increases the variance of the round trip times. It is not surprising that the largest variances occur in the 64-router network, since the additional packets to be forwarded have a bigger impact.

Measurements of the round trip time for different VR numbers and under varying conditions give a good understanding of the relationship between a certain set up and the possible accuracy of the results. Of course a Virtual Router can not provide the predictive behavior a pure simulation can. On the other hand, processing speed and CPU workload also have an impact on real network devices and a less idealized and less predictive evaluation scenario might be advantageous and more realistic. However, the results show, that the behavior of Virtual Routers is realistic enough to make them a useful tool for experiments or an evaluation scenario during development.

4 Bandwidth Measurements

4.1 Traffic Conditioning

Because of its flexibility, the queuing system is one of the most complex parts of the interface. It consists of a pool of components like queues, filters, shapers, schedulers, which can be combined and configured during run time. 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 [7] and Assured Forwarding [5].

These queuing components can be used to set up Differentiated Services networks or to establish mechanisms to protect certain flows against others. The right hand side diagram in Figure 10 shows how several queuing components can be combined to set up a specific traffic conditioning system to favor congestion-avoiding TCP flows over aggressive UDP traffic.

In the setup from Figure 10, incoming packets are processed by a classifier C checking the packet's protocol id. TCP traffic is put to queue Q_2 while any other packets are put to queue Q_1. The token bucket filter T causes Q_1 to drop packets exceeding a certain packet rate. Finally the scheduler S reads packets from Q_2 directly. Packets from Q_1 are read via the token bucket filter.

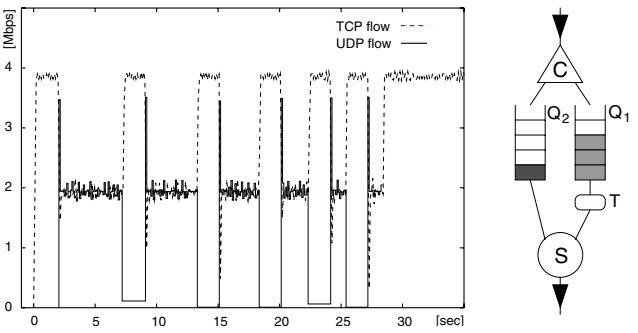

Figure 10. Throughput Measurements: TCP flow protection through a VR by an appropriate queuing system.

Table 3. Two Differentiated Services marker rules to mark packets from specific subnets with the Assured Forwarding Class 1 DSCP.

Differentiated Services Marker:	
—	
source:	10.42.11.0/255.255.255.0 dest: 0.0.0.0/0.0.0.0
service:	Assured Forwarding Class 1 with
	1.00 Mbps for low drop precedence
	0.50 Mbps for medium drop precedence
—	
source:	10.42.10.0/255.255.255.0 dest: 0.0.0.0/0.0.0.0
service:	Assured Forwarding Class 1 with
	2.00 Mbps for low drop precedence
	0.50 Mbps for medium drop precedence

The scheduler applies absolute priority round robin mode, favoring the queue with the token bucket filter which is configured for a bucket rate of 2 Mbps.

The graph in Figure 10 shows the achieved throughput values. The interface of the VR was limited to an overall bandwidth of 4 Mbps. The test starts with TCP traffic only, which is put to Q_2 and processed by the scheduler, achieving the full bandwidth of 4 Mbps. After a few seconds, a UDP source starts sending. The UDP packets are put to queue Q_1 and would suppress TCP's bandwidth completely, if the token bucket filter would not limit the rate of Q_1 to 2 Mbps. TCP gets a certain amount of bandwidth, even if the scheduler favors the queue Q_1, since Q_1 is limited by the token bucket filter. During the test the UDP source was switched on and off repeatedly, to visualize the impact of UDP on TCP.

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 also offers a suitable testbed for traffic measurements. Since the VR has to process packets in the real time the number of routers emulated on a host and the maximum allowed bandwidth are limited (even if the performance may be increased significantly by multiprocessor computers).

4.2 Virtual Routers and Differentiated Services

In the previous section the flexible traffic conditioning system was shown. Virtual Routers are capable especially to realize Differentiated Services networks.

Dependent on its setup, a Virtual Router can act as an intermediate or a border router or can be combined with real Differentiated Services networks. A Differentiated Services marker can be configured to mark certain flows with specific Differentiated Service Code Point (DSCP) values as shown in Table 3, allowing to set up first hop or ingress routers.

In this section results of experiments regarding Differentiated Services shall be presented and compared to similar experiments [2] [1] with the ns network simulator .

Even if the experiments do not provide a direct comparison between Virtual Routers and ns (since partially different bandwidths were used), they are capable to demonstrate a similar behavior of Virtual Routers and ns for similar scenarios.

The experiments cover aspects like the sharing of bandwidth between TCP and UDP flows in an assured service environment. Assured service allows to assign packet a low-drop precedence and, therefore, provide streams or their certain shares with a higher priority. The current proposal for Assured Forwarding [5] defines three drop precedences. However, the experiments presented here use two different drop precedences to allow a simpler comparison of the results.

Three experiments shall be presented: the sharing of bandwidth between TCP flows within a well provisioned network, assured and achieved bandwidths of UDP flows during heavy congestion, and the capability of Differentiated Services to protect congestion-avoiding TCP flows against an aggressive UDP traffic.

Figure 11 shows the results of two similar experiments regarding the sharing between TCP flows with different assured bandwidths.

In both configurations TCP flows only were transmitted. Packets of each TCP flow up to a certain rate are marked with a lower drop precedence. Both graphs show the assured bandwidths and the throughputs achieved by each flow. Since in both experiments less bandwidth is assigned to the assured service than the network is capable to transmit, each flow achieves a higher throughput than marked with low drop precedence. Even if the link bandwidths in both experiments differ, the distribution of the available bandwidth is similar: bandwidth exceeding the assured packet rate is shared equally among the flows. Both

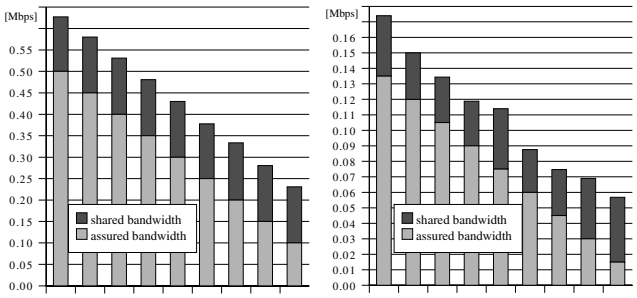

Figure 11. Bandwidths of TCP flows in a Virtual Router (left) and a ns setup (right). The graphs show the assured bandwidths and the throughputs exceeding the assured ones.

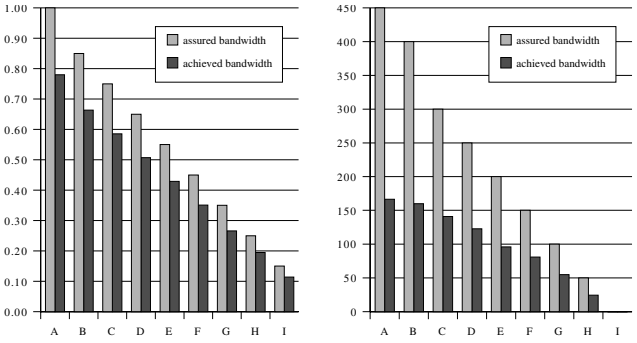

Figure 12. UDP flows with different assured bandwidths in a badly provisioned Virtual Router network (left) and in a similar ns experiment (right).

ns and Virtual Routers show these equal distribution of unallocated bandwidth to TCP flows.

The next examined scenario is the impact of a badly provisioned network on a set of flows. The concept of Differentiated Services relies on a properly provisioned network. Therefore, no more traffic with a low drop precedence shall enter a network than can be handled within that domain. If a higher packet rate was allowed to pass the border routers, the ISP would not be able to provide the negotiated service. Figure 12 shows the impact of such a badly provisioned network on a set of UDP flows.

Since the amount of assured bandwidth exceeds the networks capacity, all packets exceeding the assured bandwidths are dropped. The remaining parts of the flows share the bandwidth proportionally to their assured bandwidth.

One goal of Differentiated Services is to protect congestion-avoiding TCP flows against an aggressive traffic like UDP [2] [6]. Since TCP reacts to packet loss by decreasing its packet rate, without a special protection any

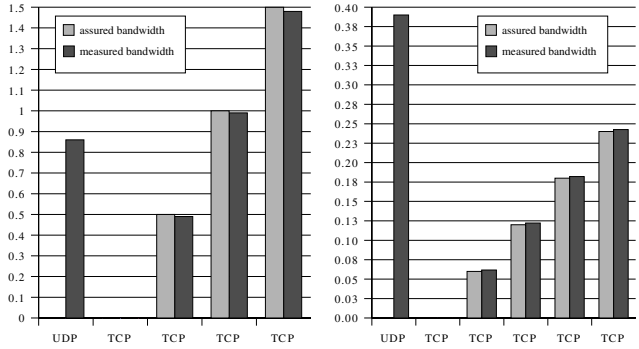

Figure 13. Protection of TCP against aggressive UDP traffic in a Virtual Router (left) and a ns simulation scenario (right).

aggressive data source can cause TCP to cease data transmission. By assigning a low-drop precedence to the packets of TCP flows only, TCP can be assigned a certain share of the available bandwidth.

Figure 13 shows this effect in a Virtual Router network and an ns network. In parallel a set of TCP and UDP flows are transported through a Differentiated Services network. The graph displays the assured and the measured bandwidths. For the UDP traffic, only the overall bandwidth is displayed. While in both graphs the TCP flows achieve the assured bandwidths, the UDP flow consumes all the bandwidth not allocated to the TCP flows. The impact of aggressive traffic to TCP without protection by a lower-drop precedences is clearly demonstrated by the first TCP flow, which got no assured bandwidth. In both experiments, null throughput was measured due to the suppression by aggressive UDP traffic. .

5 Summary and Conclusions

Virtual Routers were developed to allow for the simple and quick setup of test networks for development purposes. When using Virtual Routers, an appropriate evaluation network is available without any need for expensive hardware. Each Virtual Router can be configured by a command line interface similar to those in common operating systems. Virtual Routers allow to set up large emulated network topologies on a small number of computers.

Virtual Routers can be connected to a real network and process real traffic. Therefore, CPU speed, workload and the way of distributing Virtual Routers to a set of computers have a direct impact on the accuracy of emulation results. The experiments measured packet delays and their variances under various conditions. The results for the tested networks (up to 64 Virtual Routers on a single computer), show a constant packet forwarding delay for each

router. Additional traffic increases this delay only slightly and causes mainly an increase of the measured variances. Distributing the network emulation to two computers simply adds a certain constant overhead to the per hop delay, since packets have to be exchanged between the Virtual Routers on two different hosts.

Results of evaluating Differentiated Services with the ns network simulator were compared to a Differentiated Services network set up with Virtual Routers. The experiments show that the behavior of TCP and UDP flows supported by Differentiated Services on Virtual Routers is similar to that obtained by the ns network simulator. In all tested scenarios the flows showed a similar distribution of resources among the competing flows. The capability of Assured Forwarding to prioritize certain kinds of traffic and even to protect congestion-avoiding TCP flows against aggressive UDP was shown with both tools.

The results of the experiments show that Virtual Routers provide a convenient tool for the setup of flexible testbeds consisting of several dozens of nodes. While they produce results similar to network simulations, they provide a more realistic environment. The similarity to real routers, combined with capability to set up a virtual network on even a single computer, is beneficial for both evaluation and development.

Acknowledgments

The work described in this paper is a part of the work done at the University of Berne 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) and the SNF R'Equip project no. 2160-53299.98/1. This work was also supported by the NSF grant EIA 0103676.

References

[1] F. Baumgartner and T. Braun. Evaluierung von Assured Service für das Internet. In R. Steinmetz, editor, *Kommunikation in Verteilten Systemen (KiVS)*, Lecture Notes in Computer Science, pages 72–85, Darmstadt, Germany, 1999. Springer. ISBN 3-540-65597-2.

[2] F. Baumgartner and T. Braun. Fairness of assured service. In H. Szczerbicka, editor, *Modelling and Simulation, 13th European Simulation Conference*, volume 1, pages 390–397, Warsaw, 1999. ISBN 1-56555-171-0.

[3] Emulab.net, University of Utah, School of Computing, http://www.emulab.net, June 2002.

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

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

[6] J. Ibanez and K. Nichols. Preliminary simulation evaluation of assured service. Internet Draft `draft-ibanez--diffserv-assured-evald-00.txt`, August 1998.

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

[8] NIST net, National Institute of Standards and Technology, http://snad.ncsl.nist.gov/itg/nistnet, June 2002.

[9] The network simulator - ns-2, Information Science Institute, University of Southern California, http://www.isi.edu/nsnam/ns, June 2002.

[10] Opnet modeler, OPNET Technologies, Inc., http://www.mil3.com/products/modeler/home.html, June 2002.

[11] S. Wang and H. Kung. A simple methodology for constructing extensible and high-fidelity tcp/ip network simulators. In *IEEE INFOCOM*, March 1999.