Virtual Routers: A Tool for Networking Research and Education

Florian Baumgartner, Torsten Braun, Eveline Kurt, Attila Weyland
Universität Bern, Neubrückstrasse 10, 3012 Bern
Phone/Fax: +41 31 631 4994/3261
[baumgart|braun|kurt|weyland]@iam.unibe.ch]

ABSTRACT

Virtual routers are software entities, i.e. user space processes, emulating IP routers on one or several (Linux) computers. Virtual routers can be used for both networking research and education. In contrast to simulation, virtual routers process packets in real-time and the virtual router code is similar to code in real routers. In the case of research, larger network test-beds can be built using a relatively small number of computers. New functionalities such as new queuing mechanisms are supported by a modular software architecture and can be tested in a rather safe environment compared to kernel space implementations. Virtual routers can also be used as a tool aiming to allow students to perform virtual experiments within a computer networks course. Students can create and experiment with arbitrary virtual IP network topologies. The web-based user interface allows students to interact remotely with the emulated routers, but simultaneously it is very similar to commonly available configuration interfaces of network devices in reality. This enables students to configure routers like in the real world but also to experiment in a much more robust and safe environment.

Categories and Subject Descriptors

C.2.6 Internetworking

General Terms

Management, Measurement, Performance, Design, Experimentation.

Keywords

Network Emulation, Performance Evaluation, Networking, Distance Learning

1. Introduction

The strengths and drawbacks of network simulators, like the ns network simulator [6], 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 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 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.

For the development of new network protocols, however, protocol implementations should be closer to real system environments and real-time packet processing is desirable in order to support a larger variety of performance evaluations. In particular, it would be advantageous if new functionalities can be tested in real environments with typical applications and network scenarios. Another requirement appears for networking education. Students should be able to perform practical networking experiments, but at least for early steps safe and robust laboratory environments are desirable.

Both of these application scenarios, i.e. research and education, are supported by a concept called virtual routers. Virtual routers emulate IP routers and can support flexibility and robustness by a modular software architecture and user space processing. We propose to use emulated routers not only for research and development, but also for distance learning allowing students to perform computer network experiments remotely and within a safe environment. This allows students to prepare themselves for later experiments that are performed remotely with real network devices [3]. Remote experiments are very popular in various areas such as nano-science [7], engineering [8][9], computer networks [10] etc.

This paper discusses related work in Section 2. It then introduces the implementation architecture of virtual routers in Section 3. Section 4 evaluates the performance of virtual routers using several different scenarios. Section 5 shows how virtual routers can be used for networking research and Section 6 shows how students can use virtual routers in a web-based networking course. Section 7 concludes the paper.

2. Related Work

The emulation of network devices or routers is not new. Emulab [17] is an ongoing project providing a large test-bed for the emulation of networks. The test-bed can be accessed remotely and provides mechanisms to use ns2 scripts [6] for the configuration of the test network. The Emulab approach is very flexible and powerful, but requires a large set of computers and network devices. Similar to the Emulab approach, several other approaches, even usually on a smaller scale, use single computers to emulate specific routers [4][5][16]. The Click modular router is a highly modular and well performing software router implementation, which allows the emulation of a very flexible, easy to configure router on a single computer.

To reduce the hardware resource consumption, some of these approaches support the partitioning of the hardware into multiple logically completely independent computers using virtual machines (e.g. VMware), allowing the execution of multiple routers on a single computer. Another approach is to virtualize a host's process network and file-system namespace [22]. In contrast virtual routers do not require partitioning the hardware platform, but are designed to be run in parallel in user space and to use high level communication channels for interconnections. While this requires a modification of applications, it also allows providing simpler interfaces.

There are other more specific approaches to emulate the typical behavior of a special network device (e.g. a WAN Router) or a specific link. NistNet [21] uses Linux kernel modules to emulate the behavior of a specific link and allows the emulation of WANs within a standard laboratory LAN.

Another approach to reduce the amount of required equipment has been proposed by Wang et al. [18]. They use an address mapping scheme in the kernel to route packets multiple times through the same host. This allows studying the impact of multiple hops on a single computer.

Also ns2 [6] supports the interconnection of real network devices with simulated environments and allows to route packets through a simulated network. However, since it is a simulator, ns2 has a completely different architecture and completely different interfaces than normal IP routers. Especially the completely different user interface is a drawback in the context of a course teaching the configuration of IP routers.

The Click [16] router provides a very flexible, modular software architecture, which consists of packet processing elements, placed in-line along a forwarding path. The user specifies the elements and their order on that forwarding path. A packet follows the forwarding path and is processed by each element on that path. Although the modular structure allows for a very fine-grained, highly detailed router configuration process, the system design (one Click router requires one dedicated running kernel, i.e. one computer), makes the Click router unsuitable for our distant learning environment, where we require cost-effective and resource preserving solutions for configuring multiple routers.

A system for simultaneous utilization of available network components by different users is presented by X-Bone [19][20]. It is intended for automated deployment, management and monitoring of IP overlay networks. The separation of the network components is achieved by introducing the overlays. Access control of users is done via certificates. Thus different configuration scenarios can be set up on the same network components to be used later during exercises by students. Also in this approach the network components must exist in reality to be usable. Additionally, the manual configuration of interfaces and routing tables does not scale well for networks with many components.

The existing approaches for emulation of network devices in the literature offer interesting and useful concepts for specific tasks, while the virtual routers aim to provide an extendable, cost-effective, and widely usable platform, which has already been used in active networking and QoS research as well as in hands-on courses.

3. Virtual Routers

3.1 Virtual Router Networks

Virtual routers are small entities (Unix user space processes) that are emulating single IP routers [11]. Links that are normally used to connect real routers are replaced by communication channels between these entities in order to create larger networks. Each virtual router runs as an independent process not interfering with other virtual routers and only exchanging packets by communication channels.

Figure 1 illustrates the interconnection of several virtual routers (VR) that are distributed to three different computers. The type of the communication channels between the virtual routers depend on whether they run on the same or on different computers. Virtual routers running on the same computer are interconnected by inter-process communication channels, while virtual routers running on different computers are interconnected by channels based on UDP tunnels. Up to 64 virtual routers have been created on a single computer for various research experiments [13].

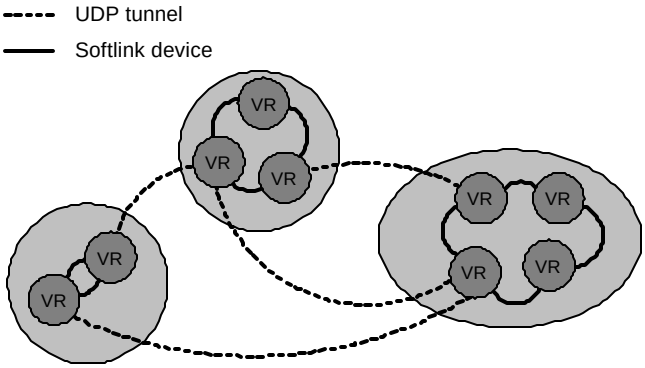

Figure 1 : Network of Virtual Routers

Virtual routers shall behave like real routers and therefore have to process traffic in real time. Like normal routers they have to receive packets, process, and forward them. Virtual routers can also be connected to real networks. This allows to route traffic generated by real end systems over a network consisting of both real and emulated sub-networks. For the interconnection of virtual routers to real networks, so-called softlink devices (Figure 2) have been implemented. A softlink device behaves like other network devices within a computer, e.g. an Ethernet device. However, packets transmitted over a softlink device are not transmitted over a real network but to a virtual router that is connected to the softlink device via file system I/O. In the other direction, packets from a virtual router can be sent to the IP stack on top of the softlink device. If in the scenario depicted in Figure 2 a web server transmits a packet to a remote web client (web browser), the packet is sent via the socket, TCP/IP stack, and the local softlink

device (sol0). Virtual router 1 receives the packet and forwards it via an UDP tunnel to virtual router 2. Virtual router 3 receives it via an inter-process communication channel from virtual router 2 and forwards the packet to virtual router 4, again via an UDP tunnel. Virtual router 4 delivers the packet to the softlink device at the receiving end system and the web client receives the packet from the socket it is connected to.

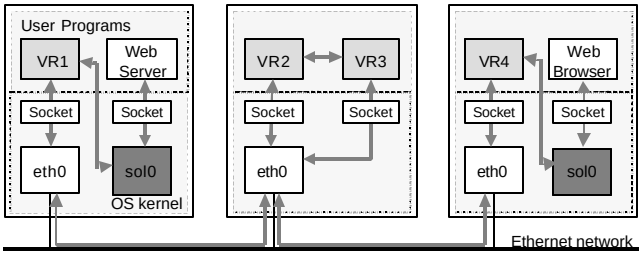

Figure 2 : Softlink Device

3.2 Implementation Architecture

Figure 3 shows the implementation architecture of a virtual router. The lower part consists of the core components required for packet forwarding and routing. IP packets are received and transmitted via interfaces (dashed boxes). Interfaces consist of several configurable subcomponents such as network address translators (NATs), queuing systems (e.g., traffic conditioners for Differentiated Services, token bucket filters, schedulers etc.), rate limiters, and interconnection handlers. Interconnection handlers interconnect a virtual router with other virtual routers or softlink devices. The packets received from an interface are processed by a programmable filter and the forwarding unit. Packets might be forwarded to other interfaces or to higher-layer components / protocols. A virtual router can be extended by dynamically loadable objects such as an active router extension (Python Based Active Router, PyBAR), a graphical user interface (command line interface), a traffic monitoring component (dump), and virtual network diagnosis utilities (ping, traceroute).

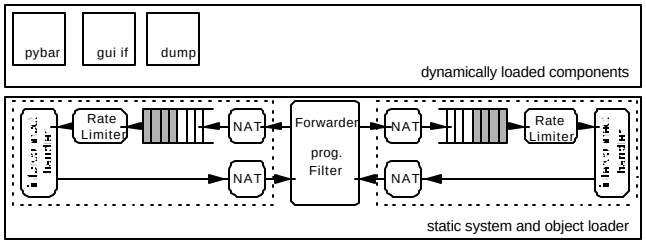

Figure 3 : Virtual Router Implementation Architecture

The central unit of each Virtual Router is an event scheduler, where most of the router's components are registered and which takes care of scheduling events in a fair way.

A general problem of running multiple routers on a single computer is the undesired synchronization between different processes. Initial tests showed oscillation effects between different routers, caused by interferences between the operating system's process scheduler and the event schedulers of the virtual routers. It was possible to reduce this effect by increasing the frequency of context switches, performed by the kernel's process scheduler. The most significant improvement to avoid these side effects however, was the modification of the router's event scheduler. Instead of processing events in a fixed order, the processing order of almost simultaneous events was randomized, which eliminated the oscillation effects.

Besides handling the interconnections between virtual routers, the queuing system is another important part of the virtual router interface. It consists of a set of components such as queues, filters, shapers, schedulers etc. All these components are implemented within a framework of C++ classes, which allows the easy implementation of new types of queues or new schedulers. These components can be loaded, combined and configured during run time. This allows replacing the complete queuing system of a virtual router during runtime without the need to shut down a running network emulation process. In particular, this has been used by the active networking mechanisms, but is also useful for the rapid prototyping of packet schedulers. 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), a weighted fair queuing scheduler, a simple round robin scheduler, and a priority round robin scheduler. To allow the establishment of Differentiated Services networks some additional components have been developed like a RED queue with three drop precedence values, a special marker for Differentiated Services, and a Priority Weighted Fair Queuing scheduler for the implementation of Expedited [14] and Assured Forwarding [15].

3.3 Application Programming Interface (API)

Virtual routers provide a high degree of flexibility: Interface components can be created and modified dynamically, higher layer objects can be loaded dynamically, filters are programmable, and routing tables can be adapted. To support this flexibility, virtual routers support different configuration interfaces by API channels. The configuration program such as a graphical user interface, has to establish an API channel to the virtual router and can exchange configuration messages over this duplex channel. The communication is based on virtual router control blocks (VCRBs) and virtual router response blocks (VRRBs). A virtual router receiving a configuration command within a VCRB, parses the control block, executes the configuration command, and returns the configuration result via a VRRB. Note that the configuration program and the virtual router that communicate via an API channel can run on different computers. Currently implemented commands support adding and deleting virtual router interfaces, retrieving interface information, changing interface characteristics and queuing systems. In addition routing table entries, loadable objects, filters, and protocol stacks can be added, deleted and read. Also, IP packets can be delivered for further processing to the virtual router.

4. Virtual Router Performance

4.1 Link and Queuing Delay

In contrast to a network simulator a virtual router processes packets in real time. Therefore, changes to the processing speed or the load on the computer will directly affect the packet delay. Special 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 constant as possible. Figure 4 provides an example of measured 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 Bern.

	hops	RTT [ms]	per hop [ms]	congestion
D	15	38	2.53	unknown
D	15	32	2.13	unknown
CH	8	4.5	0.56	Small
VR	8	2.1	0.26	No
VR	16	4.3	0.29	No
Linux-DS	8	75.46	9.433	Heavy

Figure 4 : Delays in different networks

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 and the packet forwarding within the virtual routers. However, compared to the delay values of other measurements shown in Figure 4 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.

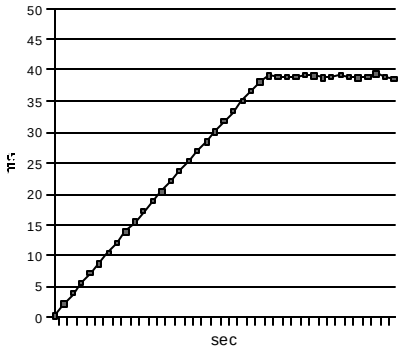

Figure 5 : Delay of a virtual router FIFO queue using a 4 Mbps link

Similar to the Linux routers, also the virtual router queuing system is the main reason for packet delays. To demonstrate that impact, Figure 5 shows the packet delay caused by the FIFO queue of the virtual router interface for different transmission 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 increases or exceeds the interface capacity. Since only slightly more packets are sent than the queue is able to process, the length of the queue - and therefore the delay - increases slowly when the experiment proceeds. Once the maximum queue length has been reached and the queue started to drop packets, the delay remains constant.

4.2 Impact of Processing Overhead to Delay

Besides the general increase in the link delay due to additional communication overhead, the current workload of the computer affects the forwarding delay. Therefore, the number of virtual routers involved in processing a packet might change the per hop delay. To measure this delay for an increasing number of routers and for different connections between virtual routers, a chain of 16 virtual routers has been set up on one (dual processor 800 MHz Pentium III computer running Linux) and on two computers (dual processor 800 MHz Pentium III computer plus a single processor 400 MHz Pentium II, both running Linux) as shown in Figure 6. Ping has been used for the measurement of the round trip times to different routers.

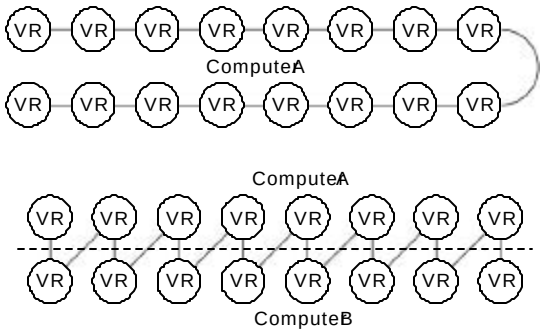

Figure 6 : 16 virtual routers on one computer (upper part) and on two computers (lower part)

To study the impact of heavier load, the experiments have been performed for two load scenarios. In the first load scenario, the virtual router network was unloaded, while in the second load scenario additional UDP traffic caused some additional processing load on the virtual routers.

Figure 7 and Figure 8 show the average round trip times and variances for 100 pings for an increasing number of hops using the virtual router configurations on one and two computers. In the scenario without additional workload (Figure 7), 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 round trip times measured in the single computer scenario (non distributed), but also increase linearly. Obviously, virtual routers cause equal link delays during packet forwarding and packet transport. In the distributed scenario using two computers, packets have to be additionally encapsulated into UDP packets and forwarded over a local area network to the computer hosting the other virtual routers. Additional packet processing and especially the transport over the local area network cause higher delays.

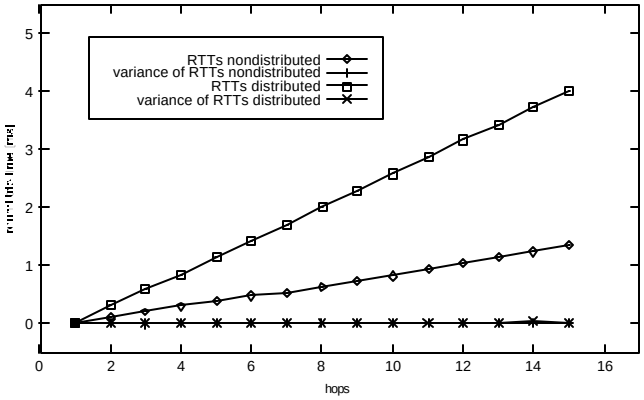

Figure 7 : round trip times in an unloaded virtual router network

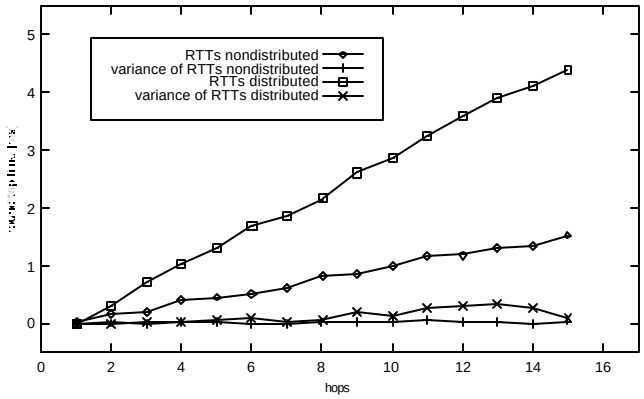

Figure 8 : round trip times in a loaded virtual router network

As it can be seen in Figure 8, the correlation between the hop count and the round trip time is nearly 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 compared to Figure 7, because of the need to transport additional packets. In addition, Figure 8 shows a small increase in variance. The linear correlation between the number of hops and the round trip times 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 scenarios, constant per hop delays for the different scenarios are discovered, even if the absolute delay values between the scenarios vary. Since virtual router delays 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 compensate variances.

To examine the impact of the topology size on packet forwarding and link delay, similar router chains were established but with 16, 32 and 64 routers. As before, the round trip times were measured first using an unloaded environment. Only the single computer scenario with running all virtual routers on one computer has been evaluated. Figure 9 shows increasing round trip times and variances for the 16, the 32 and the 64 virtual router chain. The three graphs showing the round trip times match perfectly (the graphs overlap) and show nearly no variances. Obviously, the number of running entities does not significantly affect the link and the packet forwarding delay. Similarly to the previous experiments, now additional traffic has been sent over the network and the round trip times were measured again. Figure 10 shows the measured delays and variance values for the different chains. In contrast to the experiment described above, the larger topology causes a longer per hop delay than networks with 16 and 32 virtual routers. Moreover, larger workload increases the variance of the round trip times. It is not surprising that the largest variances occur in the network with 64 virtual routers, since the additional packets to be forwarded have a stronger impact.

This behavior limits of course the number of virtual routers, which can be emulated on a single host. The maximum number not only depends on the processing power but also on the traffic load within the emulated network, the bandwidth of virtual router interfaces, the complexity of their queuing systems and of course on the required accuracy of the results. The concrete number of virtual routers differs significantly with the type of experiment. A higher link bandwidth between the virtual routers or more load on the emulated network will reduce this number. Therefore it is important to design experiments in a way that allows controlling the impact of the shared platform the virtual routers are running on.

Although, virtual routers can not provide the perfect predictive behavior as a pure simulation can do, processing speed and CPU load also might have an impact on real network devices. Therefore, a less idealized and less predictive evaluation scenario might be more realistic. In any case, the results show, that the behavior of virtual routers is realistic enough to use them a useful tool for research or educational experiments or for development purposes.

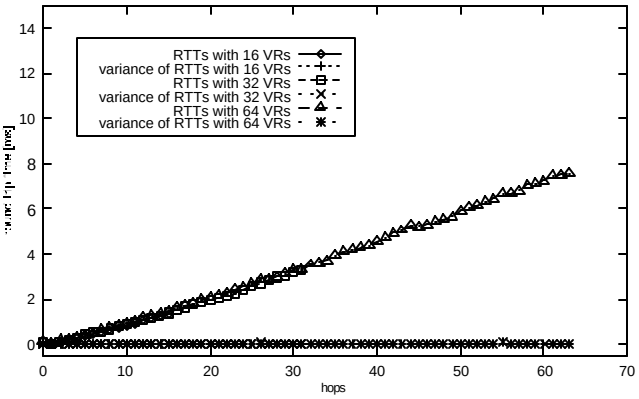

Figure 9 : round trip times with different numbers of virtual router entities in an unloaded network

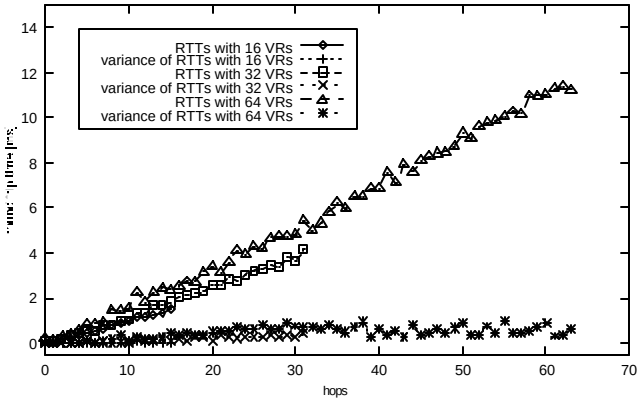

Figure 10 : round trip times with different numbers of virtual router entities in a loaded network

5. Virtual Routers for Networking Research

5.1 Evaluation of Queuing Components

The queuing components of a virtual router can be used to configure Differentiated Services networks or to establish mechanisms to protect certain flows against others. Figure 11 shows how several queuing components can be combined to set up a specific traffic conditioning system to protect TCP flows against aggressive UDP traffic. Incoming packets are processed by a classifier checking the packet's protocol identifier. TCP traffic is put to queue 1 while any other packets are put to queue 2. The token bucket filter causes queue 1 to drop packets when exceeding a certain packet rate. The scheduler reads packets from queue 2 directly. The scheduler applies absolute priority for queue 1, which is configured with a token bucket rate of 2 Mbps. Figure 11 shows the achieved throughput values. The interface of the virtual router was limited to a total bandwidth of 4 Mbps. The test starts with TCP traffic only, after a few seconds, a UDP source starts transmission. During the test the UDP source was switched on and off repeatedly in order to visualize the impact of UDP on TCP. The UDP packets would have suppressed the TCP flows bandwidth completely without the sophisticated queuing system. Due to the queuing system, however, the TCP flows get at least 2 Mbps bandwidth.

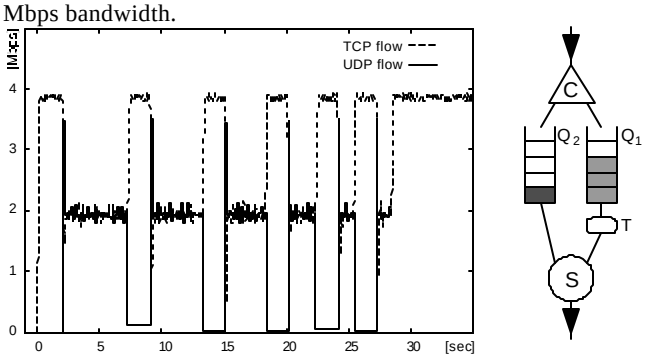

Figure 11 : Protection of TCP flows by the virtual router queuing system

Although the original idea of virtual routers was to provide a platform for the development and evaluation of mechanisms such as network management and Quality of Service routing, the architecture also offers a suitable test-bed for traffic measurements. In [13] we have shown that performance evaluation results achieved with virtual routers are quite similar to results obtained using simulation with ns.

5.2 Active Networking

Another application area for virtual routers is the development of new networking mechanisms and protocols. In particular, we have used the open and extensible platform for implementing an active network architecture and testing that in a virtual router based test-bed. The active router platform developed is based on the Python language and is called Python Based Active Router (PyBAR) [12]. The PyBAR architecture is based on the standard Python virtual machine and can be connected to several network nodes, e.g. Linux routers but also to virtual routers. In addition to a rather thin NodeOS (platform adaptor) layer written in C++, the system consists of a set of native or interpreted library and extension modules and a central core written in Python to execute received code. Received packets are forwarded either to a specific service handler, provided by an extension module, or are processed by the core. The NodeOS provides communication facilities for the PyBAR core and the extension modules. The PyBAR core and extension modules can be loaded dynamically as an object according to Figure 3. The NodeOS provides several interfaces to the routers based on Python's ability to use native code. This allows the addition of new functions like traffic conditioning, encapsulation or monitoring components directly to the IP routers kernel. In case of running PyBAR on top of a virtual router, PyBAR can take advantage from the large set of virtual router traffic conditioning functions and the flexibility to configure arbitrary queuing components at the virtual router interfaces. Figure 13 shows examples for commands provided by the platform adaptor interface. These commands can be called by a program executed by PyBAR.

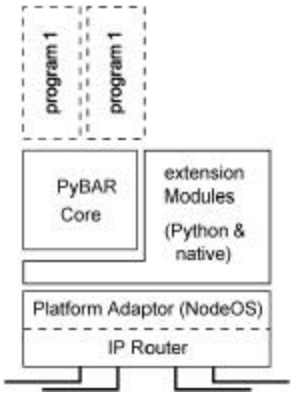

Figure 12 : Python Based Active Router

command	Description
version	returns information about the platform the system is currently running on
getCaps	returns capabilities of the Internet router, the PyBAR is attached to
attach	connects the PyBAR system to Internet routing device
getInterfaces	returns a list of interfaces
getRoutingtable	returns a list of routes
addRoute	adds a routing rule to routing table
delRoute	deletes a route from the routing table
getQcomps	query information about the traffic conditioning system
addQcomp	add a queuing component
linkQcomp	connect two queuing components
queryQcomp	query information for a specific component
configQcomp	configure a component
configTC	configuration of the traffic conditioning components

Figure 13 : Platform adaptor interface for virtual routers

6. Virtual Routers for Distance Learning

6.1 Overview

Virtual routers offer a platform for rapid development, prototyping and testing of new communication subsystems but also can serve as a platform for distance learning. Virtual routers not only help to keep the costs for building large experimentation networks very low but also offer a robust environment for performing network device configuration exercises. In order to be able to use virtual routers for web-based distance learning courses, it was required to extend virtual routers by an appropriate web-based user interface. The goal was to offer students an environment in which they could conveniently create or import their own network topologies, perform the required interface and routing table configurations in order to get the network running, and to perform tests whether the router configurations have been correct.

The work described hereafter has been performed in the Swiss Virtual Campus [1] program, which supports a series of projects in order to develop learning material for distance learning over the Internet. Within the project Virtual Internet and Telecommunications Laboratory of Switzerland [2], a set of modules has been developed and tested that allow students to perform practical exercises remotely from any Internet workstation instead of being present in a laboratory room. Typically more specialized modules follow introductive ones. A recommended order of modules is provided and helps the student to successfully advance in her learning experience.

The different modules being developed by various project partners can be classified as remote and virtual exercises. In the case of remote exercises, students work with real devices that are located in a university's laboratory room. Students control and configure the behaviour of the devices using web technologies from any workstation connected to the Internet. Students working on remote exercises need certain knowledge and experience level since potential mistakes during network device configuration can cause significant error states. These might cause that the devices will not be accessible over the Internet or in the worst case must be reset manually. Therefore, there is a need for students with a lower knowledge level to gain the experiences they need for performing such advanced modules in a more smooth way.

For this reason, a second class of modules is required, which are called virtual modules. In the case of virtual modules, the experimentation environment does not exist in reality, but it is emulated or simulated. This allows offering a much safer and robust experimentation environment. Students can make errors without the need to manually reset any devices.

The virtual routers have been embedded in a virtual exercise module, which introduces the students to IP networks, addressing and routing in particular. At this early learning stage the student does not need a fully featured, expensive commercial router. Thanks to its modularity we can just use the core components of the virtual router (addressing and forwarding) together with the virtual network diagnosis utilities ping and traceroute to obtain a powerful exercise platform for students.

The user interface to the virtual routers has been adapted to match the routers in reality more closely and thus optimally prepare the student for the more specialized modules to follow later in the course. This interface is accessible via a web browser and offers the possibility to create and modify a network as well as select and configure routers. The student lays out her network by placing and interconnecting routers on a design board (Figure 14). After she completed the layout, the appropriate resources (virtual routers, communication channels) need to be allocated before the configuration and testing can take place. The student configures a router via a command line interface, where the command line input and the command / reply history (browsable) have been separated. This command line interface gives the student a very close to reality experience when configuring the routers of her network (Figure 15). The student mainly will use ifconfig or route commands to setup the interfaces of the router (e.g., assigning IP addresses and network masks) and to configure the static routing table of each router. To validate her network configuration, the student can use the traceroute and ping commands.

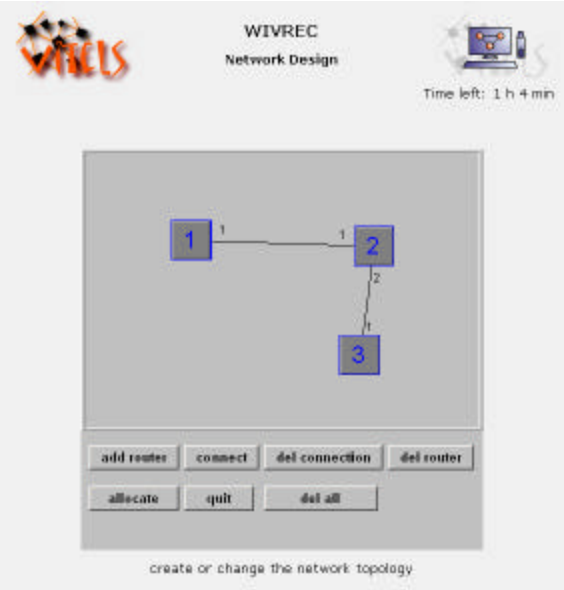

Figure 14 : Create/Change Network Applet

Figure 15 : Configuration Applet

6.2 Implementation Architecture

This subsection describes the implementation architecture of the web-based virtual router configuration interface [23]. Figure 16 shows two computers: a client where the student has launched a web browser and a server including a web server, a Java program called administrator, and the virtual routers. The student navigates through dynamic web pages generated using PHP and downloads/executes Java applets that are embedded in the web pages. The different tasks (i.e. network design, router selection and router configuration) have been realized in different applets. The student switches between them according to the current task she performs. These applets share common data such as the network configuration. Due to Java security restrictions, the data can not be saved on the client computer. Therefore, they are transmitted to the administrator program which stores the data and provides it for subsequent applets. The Java applets on the client computer then open TCP connections to the administrator program running on the server and exchange request / response messages. The most important commands are for saving data to files at the server, allocation of virtual router resources, retrieval of virtual network topology data, and closing a session.

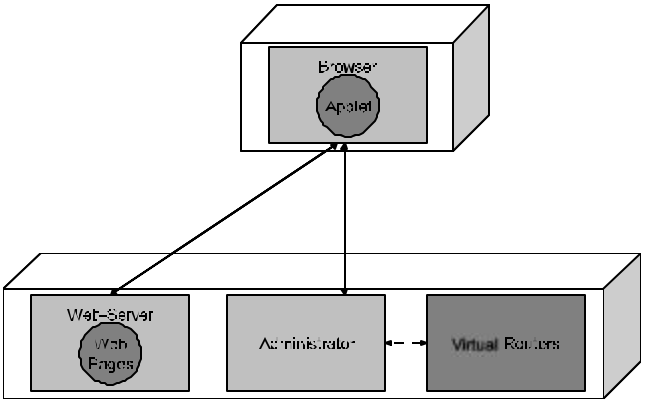

Figure 16 : Implementation Architecture

The administrator program is the interface between the applet running on the client and the virtual routers. It shares common data with the component for dynamic web page creation. Moreover, it receives commands from the applet and translates it into appropriate configuration API calls to the designated virtual router. However, there is no 1:1 relationship between commands issued by the student and commands sent to virtual routers. As depicted in Figure 17, a command from the applet triggers a sequence of virtual router API commands. Figure 17 shows the example of deleting a virtual router interface via the configuration applet. After receiving this user command, the administrator program causes the virtual router to disconnect the specified interface and then to delete it.

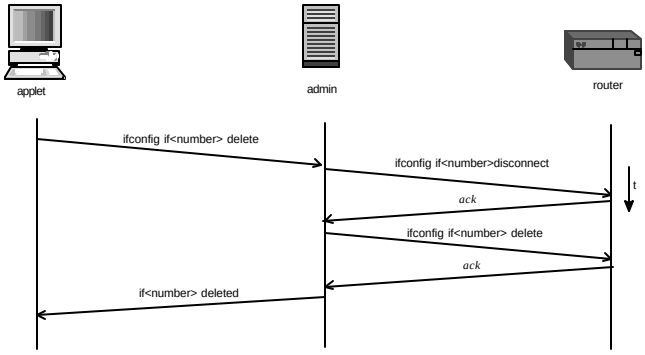

Figure 17 : Message Exchange between Client, Server, and Virtual Routers

Since the virtual exercise module allows configuring emulated routers via a web-based command line interface, the student desiring to perform the exercise only needs a web browser on her client computer. With these minimal requirements, the course module can be easily accessed by all students and can be deployed

with low costs. The course module has been tested within a regular computer network laboratory course and will be further improved based on the student's feedback.

7. Summary and Conclusions

The paper described the concept of virtual routers, which can be used to emulate multiple routers on a single computer. The virtual routers as the basic underlying components have been successfully used for several research purposes, in particular for research projects in the area of Quality-of-Service management and monitoring as well as active networking. Virtual routers are also very useful in order to provide safe, robust but realistic experimentation environments for students in a networking class. Students can prepare themselves for later experiments with real network devices.

8. Acknowledgements

The work described in this paper has been partially supported by the Swiss National Science Foundation project 2100-055789.98.

9. References

[1] Swiss Virtual Campus, www.swissvirtualcampus.ch, October 2003

[2] Virtual Internet and Telecommunications Laboratory of Switzerland, www.vitels.ch , October 2003

[3] M. Steinemann, T. Jampen, S. Zimmerli, T. Braun: Architectural Issues of a Remote Network Laboratory, Networked Learning 2002 (NL 2002), Berlin, May 1-4, 2002

[4] B. White, J. Lepreau, S. Guruprasad: Lowering the Barrier to Wireless and Mobile Experimentation, First Workshop on Hot Topics in Networks (HotNets-I), 28-29 October 2002, Princeton, New Jersey, USA

[5] Yongguang Zhang and Wei Li: An Integrated Environment for Testing Mobile Ad-Hoc Networks, Third ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2002, Lausanne, July 9-11, 2002

[6] Network Simulator, www.isi.edu/nsnam/ns, October 2003

[7] M. Guggisberg, P. Fornaro, T. Gyalog and H. Burkhart: An Interdisciplinary Virtual Laboratory on Nanoscience, Electronic Notes in Future Generation Computer Systems, Elsevier, Vol. 1 (2001)

[8] D. Bühler, W. Küchlin, G. Gruhler, G. Nusser: The Virtual Automation Lab - Web-based Teaching of Automation Engineering Concepts, 7th Annual IEEE International Conference on the Engineering of Computer Based Systems, Edinburgh, April 2000

[9] A. Böhne, N. Faltin, B. Wagner: Self-directed Learning and Tutorial Assistance in a Remote Laboratory, Interactive Computer Aided Learning Conference, September 25-27, 2002, Villach, Austria

[10] R. Sontag: Berufsbegleitend lernen: Informations- und Kommunikationssysteme, it+ti: Informationstechnik und Technische Informatik, Oldenbourg Verlag, 3/2001, pp. 167

[11] F. Baumgartner, T. Braun: Virtual Routers: A Novel Approach for QoS Performance Evaluation, QofIS'2000, September 25-26, 2000, Berlin, Germany

[12] F. Baumgartner, T. Braun, B. Bhargava: Design and Implementation of a Python-Based Active Network Platform for Network Management and Control, IFIP TC6 4th International Working Conference (IWAN 2002), Zürich, December 2002

[13] F. Baumgartner, T. Braun, B. Bhargava: Virtual Routers: A Tool for Emulating IP Routers, 27th Annual IEEE Conference on Local Computer Networks, Tampa, November 6-8, 2002

[14] B. Davie et al.: An Expedited Forwarding PHB, RFC 3246, March 2002

[15] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski: Assured Forwarding PHB Group, RFC 2597, June 1999

[16] E. Kohler, R.Morris, B.Chen, J.Jannotti and M.F. Kaashoek: The Click Modular Router, ACM Transactions on Computer Systems, Vol 18, No. 3, page 263-297, August 2000

[17] B.White, J.Lepreau, L.Stoller, R.Ricci, S.Guruprasadm M.Newboldm M.Hiber, C.Barb and A.Joglekar: An Integrated Experimental Environment for Distributed Systems and Networks, Proceedings of the Fifth Symposium on Operating Systems Design and Implementation, USENIX Association, pages 255-270, Dec 2002

[18] S.Y.Wang and H.T.Kung: A simple methodology for constructing extensible and high-fidelity TCP/IP network simulators. In IEEE Infocom, March 1999

[19] J.Touch: Dynamic Internet Overlay Deployment and Management using the X-bone. In Proceedings of ICNP, pages 59-68,2000

[20] J.Touch and S.Hotz: The X-bone. Third Global Internet Mini Conference in conjunction with Globecom'98, Sydney, Australia, November 1998.

[21] Nistnet http://snad.ncsl.nist.gov/itg/nistnet, October 2003

[22] S. Guruprasad, L.Stoller, M.Hibler and J. Lepreau: Scaling Network Emulation with Multiplexed Virtual Routers, In Final Program and Poster Abtracts of Sigcomm, August 2003

[23] F. Baumgartner, T. Braun, E. Kurt, M Steinemann and A. Weyland: Implementation of a Distance Learning Module Based on Emulated Routers, in Proceedings of the 13. ITG/GI-Fachtagung Kommunikation in verteilten Systemen (KiVS 2003), Leipzig, Germany, March 25-28, 2003, pp. 71-80