Modeling of Modern Router Architectures Supporting Network Virtualization

Tobias Hossfeld
University of Würzburg
Department of Distributed Systems
Am Hubland
97074 Würzburg, Germany
hossfeld@informatik.uni-wuerzburg.de

Kenji Leibnitz
Osaka University, Graduate School of
Information Science and Technology
1-5 Yamadaoka, Suita, Osaka
565-0871, Japan
leibnitz@ist.osaka-u.ac.jp

Akihiro Nakao
The University of Tokyo, Interfaculty
Initiative in Information Studies
7-3-1 Hongo, Bunkyo-ku, Tokyo
113-0033, Japan
nakao@iii.u-tokyo.ac.jp

***Abstract*—For the emerging concept of *network virtualization* where a multitude of virtual networks (VNs) coexist, the main challenge is to isolate and control the resources allocated for individual virtual networks. Realizing such resource allocation requires deliberate design of the router enhanced specifically for handling a large number of virtual network flows. This paper discusses fundamental challenges on modeling the architecture of modern routers for the support of the virtual networks. Our intention is to discuss strategies for scheduling the traffic on each isolated VN and to provide an elementary framework for evaluating the performance of the router extended to classify incoming VN traffic. Furthermore, we also present simulation results and discuss challenges and future directions encountered in the implementation and deployment of VNs.**

I. Introduction

The challenges that are envisioned for future Internet services are leading to a paradigm shift in the design and operation of router technology with greater emphasis on individual services and their perceived end-to-end quality. Recently, there have been a growing number of new application-oriented overlay networks, which are logically constructed above the current IP infrastructure to provide services in a highly-distributed way, such as *peer-to-peer* (P2P) file sharing, or directory lookup with *distributed hash tables* (DHT). Since IP (Internet Protocol) layer routing is rather limited in its current form to simple forwarding of packets, the complex operation of the system lies here on application layer.

Network virtualization appears to be a promising approach to alleviating this problem by segregating the provisioning of services and applications from the physical infrastructure. The introduction of *Virtual Networks* (VNs) leads to an entirely new scenario, where more complex routers are required that can only provide mere lookup of destination addresses and forward packets to a designated output port. Instead, they consist of self-contained virtual machines on high-performance routers, which can provide many programmable and configurable options. Thus, network virtualization provides a means to shift the complexity from application layer to network layer, facilitating the easier and faster deployment from the viewpoint of the service providers, as well as providing the opportunities for new business models for services.

In order to support VNs, a redesign of the current router architecture is required to support much more complex functionalities [1]–[4]. These include for example new routing schemes like energy-efficient or semantic routing, packet filtering, address translation, brokering *quality of service* (QoS) or *quality of experience* (QoE) reservations or negotiating and policing *service level agreements* (SLAs). Each of these functionalities is expected to be fully configurable by the service provider directly at the router in accordance to the type of offered service. However, different services may also require different physical resources. For instance, filtering and address translation require CPU time, whereas delay-sensitive or quality-sensitive applications (e.g. multicast support for IPTV, VoIP) require an efficient scheduling of the flows to maintain bandwidth and delay constraints. Although the main idea behind VNs is to ideally completely isolate each network, they nevertheless must share the same common resources. This can introduce delays and jitter due to sharing CPU time or the waiting times in queues and processing times at the line cards.

In this paper our goal is to introduce a model for the scheduling and operation of a VN-supporting router from the viewpoint of its performance. We begin with a simplified router model as used in the conventional Internet infrastructure, which we then extend to incorporate also novel technology trends in router technology, such as multi-core processors. Due to the high complexity of such a VN-supporting router, we must provide simplified assumptions, which nevertheless yield the essential characterizes and key features in order to provide a framework for more sophisticated future studies.

The rest of this paper is organized as follows. Section II discusses some of the key issues on the concept of network virtualization. Section III presents an approximative model for a VN-supporting router, exhibits the key characteristics of handling multiple VNs and illustrates the need for the sophisticated router architecture. Further numerical examples comparing different scheduling strategies for exemplary cases of VNs with different requirements are provided in Section IV. The objective of this paper is not to provide exact quantitative values, but to rather show the qualitative tendencies that can be seen in the application of such technology. Finally, Section V concludes this paper with a summary of our study.

II. Concept of Network Virtualization

With the advances in microprocessor technology, *virtualization* has evolved into a viable option for managing, controlling, and isolating several virtual entities on a common shared physical resource. This concept is now increasingly also entering the field of networking, where virtual networks are formed that are composed of interconnected virtual devices sharing the same physical substrate network. Naturally the operation of such networks imposes much more sophisticated management routines for the access to the common resources, as the internal tasks of processing flows in routers becomes much more complicated than in conventional best-effort provisioning found in today's Internet.

Users perceive each virtual network as tunnel and are free to choose their topology that suits best to their demands, whereas also infrastructure providers benefit from such concept by not having to be forced to deploy all new functionalities to support certain services at each node. This task can be shifted to the service providers to manage and reprogram their network architectures offering the end users a service-specific end-to-end quality. An end user can therefore connect to multiple service providers, which offer exactly their custom-made services, without any interaction with the actual infrastructure. Thus, beside the technological benefits, the introduction of virtual networks would lead to entire different business and pricing models from economical viewpoint.

Virtual Networks are currently a much discussed topic in the research community for simultaneously operating new testbed protocols and services in an isolated environment. In this context, PlanetLab [5] is one of the most prominent examples. It was established in 2002 and as of January 2009 is currently consisting of 943 nodes at 466 sites worldwide [6]. Nodes in the PlanetLab network are usually Linux-driven PCs isolating services and applications in *slivers*, which are VMs running on the node, and which compose a *slice* for the entire virtual network. In order to maintain the integrity and stability among the slivers, an additional administrative mechanism is implemented: a resource container (based on Linux VServer) implements and isolates the virtual execution environment on the Linux kernel, whereas the task of the node manager and local administration are to manage the resources of all VMs and enforce the policies.

Among the major challenges we address in this work for the design and operation of VNs is the allocation and scheduling of the common substrate resources, especially focusing on isolation of resources allocated to each virtual network. Further key issues on VNs are summarized in [7]. Since each VN may have different requirements depending on the service it is offering, bandwidth management and queue scheduling are of essential importance. These involve scheduling mechanisms for CPU, memory, disk, network interface in each of VMs running on the host machine. An adaptive and dynamic resource allocation framework is proposed as DaVinci [8], where each substrate link periodically reassigns bandwidth shares between virtual links.

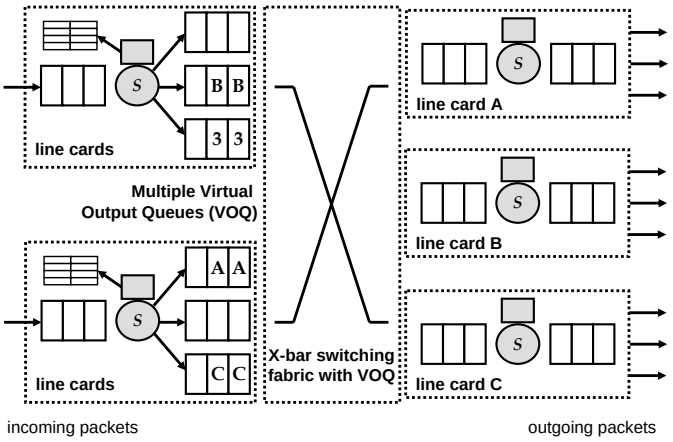


Fig. 1. A store-and-forward router implementing Virtual Output Queues

III. Modeling a Single VN-Supporting Router

In order to keep our following discussion tractable, we consider in this work a simple scenario consisting of $n = 3$ different virtual networks, which are each dedicated to certain applications like video, voice, or web traffic. The traffic profile of each application i is characterized by the size S_i of IP packets and their average bandwidth requirements R_i. The average number of users within VN i is denoted as N_i. The arrival of IP packets in VN i is modelled as a Poisson process caused by the superposition of the flows from several users. We use the parameter θ to adapt the system load and refer to it as the *normalized system load*. Then, the arrival rate of the Poisson process follows as $\lambda_i = \theta N_i R_i / S_i$.

We now consider packets arriving at a certain router. Following the discussion in [9], we assume the popular store-and-forward router architecture with *Virtual Output Queues* (VOQ) at the input links, see Fig. 1. The essential components of the router are the *line cards* and a *switching fabric* controlled by a centralized scheduler. Each line card controls an input link and an output link. In the VOQ strategy each input link maintains a separate queue for each output link.

When a packet arrives at a router, i.e., at the input link of a line card, its destination address is looked up in the forwarding table. To be more specific, the packet completely leaves the input link and is first stored in the corresponding output queue of the line card's memory. Due to the VOQ concept, each input link has a separate FIFO queue dedicated to each output link. When the packet gets served in the output queue, it is passed to its output interface by the switching fabric and handed to the output link scheduler. Here, the packet may be queued before being forwarded to the output link. The speed of processing corresponds to the bandwidth of the output link.

A. Simple Router Model

Hohn et al. [9] propose in their paper a simple model for such a store-and-forward router and validate its accuracy and applicability using measurements of traffic at a gateway router of the Sprint IP backbone network. Based on their

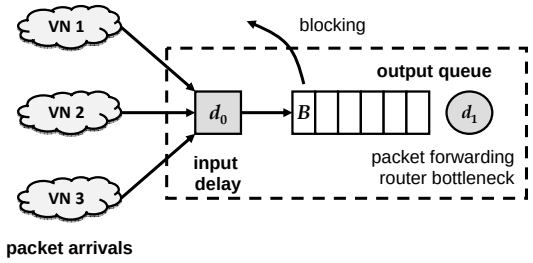


Fig. 2. Simple model with minimum delay due to input buffer at router

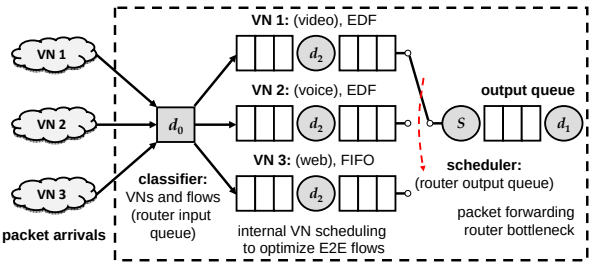


Fig. 3. Model of a complex router supporting various VNs

observations, they propose a two-stage model. First, each packet experiences a minimum delay $d_0(s)$ which depends on the size s of the packet and reflects the store-part of the router. Then, the packets enter the FIFO output queue and are processed with the capacity of the output link when they reach the head of the queue.

In this two-stage model, the order of packet arrivals at the input link is preserved. The output link is assumed to be the bottleneck of the router, as in practice the switching fabric is generally overprovisioned and therefore very little queueing is expected at the input queues. If a packet enters the system and has to wait for longer than $d_0(s)$ in the output queue, then the effect of this minimum delay can be simply neglected.

For the sake of completeness, the equations of the two-stage system model is reviewed briefly according to [9]. Let $U(t)$ be the amount of unfinished work at time t, which is the time it would take until the system is completely empty, given that there are no further packet arrivals. When a packet j of size s_j arrives at time t_j, then it is immediately inserted at the end of the FIFO queue, if the unfinished work $U(t_j)$ is larger than the initial delay $d_0(s_j)$, i.e., $U(t_j) > d_0(s_j)$.

Then, the amount of unfinished work is increased immediately after queueing the packet at time t_j^+ by the amount $d_1(s_j) = s_j/c_1$ corresponding to the time it takes to process the packet with the capacity c_1 of the output link. Formally, this can be described as

$$U(t_j) > d_0(s_j) \Rightarrow U(t_j^+) = U(t_j) + d_1(s_j).$$

The sojourn time T_j of the packet is then simply the sum of its waiting time, i.e., the amount of unfinished work at time t_j, and its processing time is $U(t_j) > d_0(s_j) \Rightarrow T_j = U(t_j^+)$. If the remaining unfinished work is larger than the delay $d_0(s_{j+1})$ of the next incoming packet $j+1$, we obtain

$$U(t_{j+1}) > d_0(s_{j+1}) \Rightarrow T_{j+1} = T_j - (t_{j+1} - t_j) + d_0(s_{j+1})$$

since the unfinished work is reduced by the amount $t_{j+1} - t_j$, i.e., $U(t_{j+1}) = U(t_j^+) - (t_{j+1} - t_j)$.

If however a packet k arrives at time t_k with a delay $d_0(s_k)$ larger than the unfinished work, the sojourn time for this packet satisfies $U(t_k) \leq d_0(s_k) \Rightarrow T_k = d_0(s_k) + d_1(s_k)$ and the unfinished work accordingly can be derived as

$$U(t_k) \leq d_0(s_k) \Rightarrow U(t_k^+) = d_0(s_k) + d_1(s_k).$$

Figure 2 shows the simple model with the minimal delay d_0 due to the input buffer at the router. Note that in practical routers there is a limited buffer capacity B, however, usually the buffers are well dimensioned and in fact actually overprovisioned [10]. Therefore, the approximation of using infinite buffer sizes can be safely assumed as suggested by [9].

B. Complex VN-Supporting Router

In the previous section we considered the model of a simple router as found in the contemporary Internet infrastructure. However, recent trends from router vendors have shown that high-performance routers with multi-core processors are being developed that are capable of operating well in an environment supporting VNs [11], [12]. Furthermore, sophisticated scheduling and management techniques are studied to improve the performance of off-the-shelf hardware to act as sophisticated routers [13], as in the OpenFlow project [14]. Let us now extend the previously discussed simple router model to provide a generalized framework for modeling more complex multi-core routers. We follow in general the structure as proposed by the VERA architecture [1], however, with our focus more on the general modeling aspects rather than on specific details of flow or packet classification or implementations.

We now consider an extension of the model we have discussed in the previous section and which is shown in Fig. 2. Again, the router receives packets on its input line cards which may originate from traffic of different VNs having different traffic characteristics. While, in the simple model the minimal delay d_0 only corresponds to the processing time of the incoming packets at the line cards, we also include now in the extended model that d_0 contains the delay from the *packet classifier*, which separates the packets from individual flows to its corresponding VNs. Since classification can be performed at a very high processing speed due to the CPU capabilities, it can be included in the approximation within the d_0 delay.

Figure 3 shows for example 3 VNs, which operate each in an isolated environment and each VN has an input and output buffer, as well as VN-specific processing delay d_2. This delay may be different depending on the functionality of processing required at each VN as previously discussed. Depending on the application-specific requirements, each VN may also work with different packet sizes and scheduling disciplines, see Fig. 3, where the VNs for video and voice traffic use EDF (earliest deadline first) scheduling, and the best-effort web

traffic uses FIFO. Note, however, that these are only arbitrary examples shown here and many further combinations with different processing times d_2 may be possible.

In the ideal case of having a complete separation of VNs, each output port could be switched over an own optical wavelength, providing perfect isolation among the VNs. However, in the currently predominant case of using electrical line cards, the traffic must be again multiplexed to use the common output port. This is performed by a scheduler that inserts the packets into the router output queue. We assume here also that the delay due to this scheduler can be neglected due to its computational speed. Finally, the bottleneck lies again at the output queue of the router, where we have in analogy to the model in Section III-A the delay at the output line card of d_1.

IV. Numerical Results for Example Scenarios

We now discuss simulation results from example scenarios with $n = 3$ different VNs, each dedicated to video, voice, and web traffic, respectively. First we consider the simple router model and illustrate the influence of the delays on the sojourn time of packets by approximations using $M/G/1/\infty$ queues. Then, we discuss simulation results of the simple model and our proposed model in the presence of the different traffic types from each VN to illustrate the general behavior of each system and characterize their performance.

A. Lower and Upper Bounds of Packet Sojourn Time

We compare the simple router model with an $M/G/1/\infty$ queueing model to find lower and upper bounds on the sojourn time. For the lower bound, we ignore the delays of the packets before being queued due to the input line cards. Thus, we only consider the processing time of packets in the output queue. The average service time $\mathrm{E}[H]$ of a packet is the ratio of the average packet size $\mathrm{E}[S]$ and the processing capacity c_1.

$$\mathrm{E}[H] = \mathrm{E}[S/c_1] = \sum_{i=1}^{n} \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j} \mathrm{E}[S_i/c_1] \qquad (1)$$

The average waiting time $\mathrm{E}[W]$ of a packet takes into account the second-order moment $\mathrm{E}[H^2]$ of the service time which is computed in a similar way to Eqn. (1). It is given as

$$\mathrm{E}[W] = \frac{\lambda \mathrm{E}[H^2]}{2(1-\rho)} \qquad (2)$$

with the system load $\rho = \lambda \mathrm{E}[H]$ of the $M/G/1/\infty$ system. The mean packet sojourn time $\mathrm{E}[D]$ simply follows as

$$\mathrm{E}[D] = \mathrm{E}[W] + \mathrm{E}[H] \ . \qquad (3)$$

For the upper bound of the sojourn time, we consider a $M/G/1/\infty$ waiting queue with service time $H = d_0(S) + d_1(S)$. The formal derivation of the upper bound of the mean sojourn times follows similar to Eqns. (1-3).

Figure 4 shows the average sojourn times of packets through the router for different normalized load factors θ and compared to the upper and lower bound values. It can be seen that the lower bounded system significantly underestimates the current sojourn time of packets, since the additional delay before a packet is queued heavily influences the system's behavior. On the other hand, the upper bounded model overestimates heavily the actual model in higher load scenarios, as the influence of the additional delay d_0 of packets diminishes when the packets are waiting for longer than d_0 in the output queue of the router.

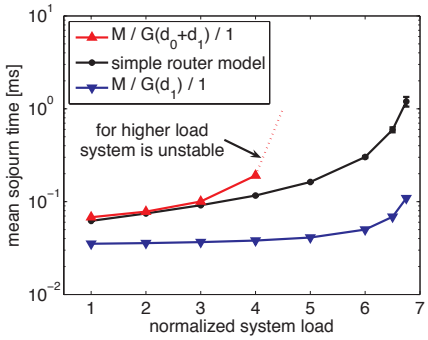


Fig. 4. Estimating of the mean packet sojourn time in simple router model

TABLE I
Simulation Settings for the Simple Router Model

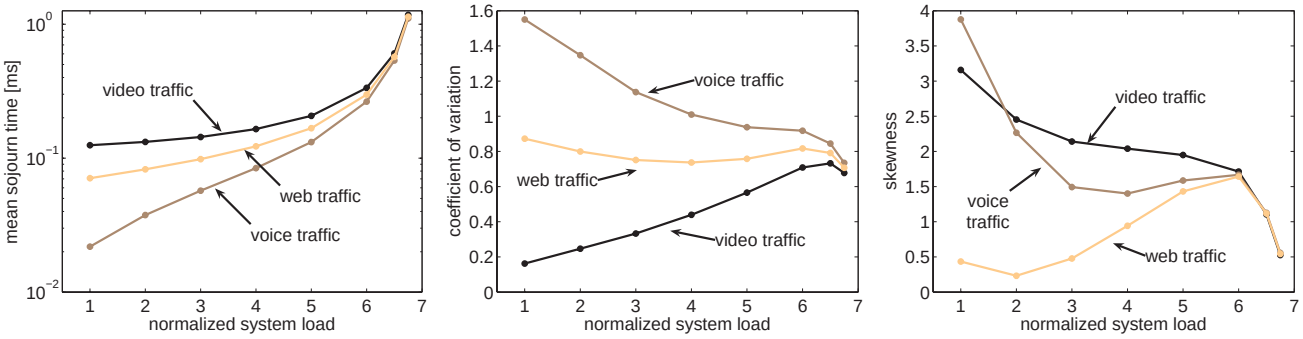
	VN 1	VN 2	VN 3
type of traffic	video	voice	best effort
number of users N_i	10	50	100
packet size S_i [kb]	12	0.9	6.16
bandwidth per user R_i [kbps]	1000	30	1

B. Performance of the Simple Model with Multiple VNs

We now investigate the performance of both router models as discussed in Section III. In particular, we consider the case of having $n = 3$ VNs, which are each dedicated to applications with different traffic characteristics. The following results only serve the purpose of investigating the basic behavior of each system and should be therefore regarded as toy model, since we are only considering a single router. In order to fully evaluate the performance of the VN architecture, we need to consider end-to-end flows with several interconnected routers.

The traffic profile of the considered VNs is shown in Table I. Each application i has different number of users N_i, packet sizes S_i, and bandwith R_i required per user. The arrival rates λ_i are computed over the number of users and capacity per user, i.e., $\lambda_i = \theta\, N_i\, R_i/S_i$ for $i = 1, 2, 3$, with the normalized load factor θ. The input delays d_0 are set depending on the packet size for each class as $d_{0,i} = S_i/c_0$, where $c_0 = 300\,\mathrm{Mbps}$ is the processing capacity at the input links and the delays at the output buffer are $d_{1,i} = S_i/c_1$ with $c_1 = 150\,\mathrm{Mbps}$. The total buffer size is set as $B = 3.75\,\mathrm{Mbit}$. We performed 10 simulation runs with 10^4 packets generated per simulation, and averaged over the mean values, coefficient of variation, and skewness of the packet delay distribution.

Figure 5 shows the results from the simulation runs for each VN in terms of the statistics of the packet sojourn time distribution as function of the system load. In Fig. 5(a), the mean sojourn time is plotted over the normalized system load θ. We can see that all curves show a similar behavior. When the system is only lightly loaded, the mean sojourn time is



(a) Mean sojourn time (b) Coefficient of variation (c) Skewness

Fig. 5. Influence of system load on packet sojourn time for simple model

mostly influenced by the packet size of each traffic type with voice traffic having a shorter sojourn time than video or best effort traffic. However, once the load increases, the system will become less able to handle all traffic and the average sojourn time increases exponentially. Under highly loaded conditions ($\theta > 5$) the sojourn time of the different traffic classes is nearly same. It should be noted that for the considered scenarios with $\theta \leq 7$, the system is not overloaded. In particular, no packets get lost due to the limited buffer size B.

In Fig. 5(b), we show the coefficient of variation for all three traffic types and can see that video traffic has the lowest variance and voice traffic the highest value due to their packet sizes. As the load increases, the coefficient of variation of the packet sojourn time distribution becomes the same for all traffic types and is actually reduced since now all packets must wait in the queue. Figure 5(c) indicates always a positive skewness. In the case of best effort web traffic there is almost no skewness in the sojourn time distribution, whereas the other traffic classes have initially a much higher skewness in lightly loaded systems. Once the system has to carry more high load, however, all traffic classes show nearly the same skewness.

In summary, Fig. 5 illustrates the need for isolation when we are dealing with multiple different traffic types competing for the same resources. As load increases, the influence from the individual characteristics of each traffic type diminishes and jitter is reduced since all packets, even small ones, are queued. This results in an exponential increase in packet delays.

C. Complex Multi-Core Router to Support VNs

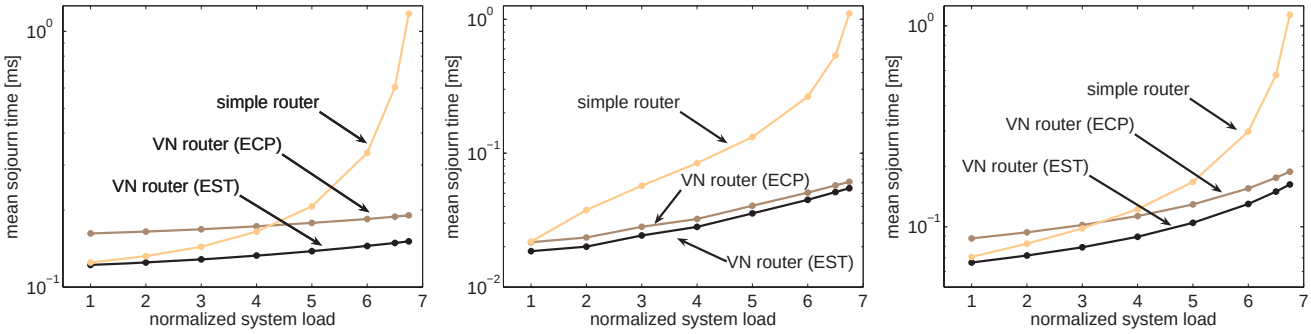
In order to provide a fair comparison between the multi-core and simple router model, we need to consider both systems under nearly equal conditions. Since we now have the processing delay d_1' at the output queue in the simple model split up into 2 parts (d_1 and d_2) in the new model, the parameter settings for comparing the packet delays in both systems must be set accordingly.

For the *Equal Service Time (EST)* settings, we assume the same overall delays for the simple router and the complex router case. In both scenarios, we keep the same d_0, however, we now have $d_1' = d_1 + d_2$, where d_1' refers to the values we used for the simple router. The overall delays introduced by the processing within the system, with the exception of queuing delays, remain the same. On the other hand, for the *Equal Capacity (ECP)* settings, we again consider that the input delays d_0 remain the same, however, the processing capacity at the output queue is being chosen appropriately. For c_1' corresponding to the processing capacity of the simple router model, we now have $d_1 = 1/c_1' = 1/c_1 + 1/(n\,c_2)$. Thus, the overall processing capacity at the output buffer remains same.

The scheduler operates with a prioritization ratio of 3:2:1 between video, voice, and web traffic, however, further experiments not included in this paper indicated also a very similar behavior in the case of equal prioritization. The same number of simulation runs and packets per simulation were performed as in Section IV-B.

1) Mean Sojourn Time of Packets: We first investigate the mean sojourn time of packets by comparing each traffic class independently for both parameter settings EST and ECP with the results for the simple router. Figure 6 shows the results for video, voice, and web traffic, respectively. The isolation of each traffic type and the inclusion of scheduling results in a drastic reduction of the mean sojourn time for both parameter settings EST and ECP compared to the simple router case. The EST parameters result always in a smaller mean packet delay than ECP, but both curves do not show the exponential increase that can be observed in the simple router case. This is mainly caused through an implicit prioritization by scheduling. The graphs in Fig. 6 show that by separating the processing and scheduling of the different traffic classes, it is possible to achieve significantly smaller packet delays than the model with only a single processing unit, especially at high load.

2) Standard Deviation of Packet Delays: Since the mean values of the packet sojourn time remain very small, the jitter in the packet delays can be better observed in the standard deviation than in the coefficient of variation, as we did in the case of simple router. In Fig. 7, the standard deviation of the packet delays for all traffic types and parameter settings EST and ECP are shown together with the corresponding curve for the simple router model. Again we can see that the complex router model with isolation significantly outperforms the simple router model and the complex router model is more robust when dealing with high load scenarios.



(a) Video traffic (b) Voice traffic (c) Web traffic

Fig. 6. Mean sojourn time of packets in complex router model

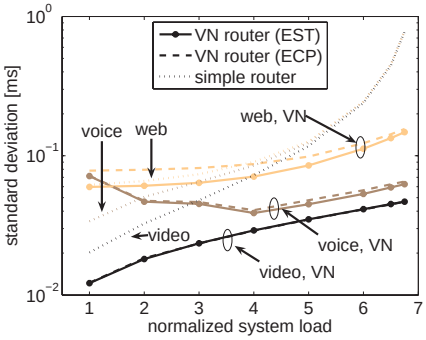


Fig. 7. Standard deviation for simple router and VN router

V. Conclusion and Future Directions

In this paper, we provided a queuing theoretical framework for modeling and evaluating the performance of a multi-core VN router. We compared our model through simulations with a monolithic router without dedicated support for Virtual Networks and showed basic characteristics for scenarios with 3 different VNs accommodating different traffic types. In general, the isolation of VNs provides benefits by reducing delay and jitter.

The concept of network virtualization has emerged in test-bed research such as PlanetLab, VINI [15], G-Lab [16], and GENI [17] to design, develop, and deploy innovative network services where each experiment is conducted in an execution environment built on top of a virtual network with both computational and network resources isolated from those of the others. However, just as ARPANET started as a test-bed decades ago has grown into the current Internet, an indispensable social communication infrastructure, we foresee that test-beds of today may eventually form a foundation of a future network architecture. For example, the concept of network virtualization developed in the test-beds may allow us to embed a role of test-beds inside the network architecture. We could experimentally run our newly invented disruptive technologies on the resources completely isolated from those for the production network. This would free us from the ossification of the Internet reported in [18] and facilitate building a continuously evolvable network architecture, where multiple independent architectures coexist in isolated virtual networks. One could migrate to any successful architecture from outdated or defective ones if such a concept becomes the norm.

References

[1] S. Karlin and L. Peterson, "VERA: an extensible router architecture," *Computer Networks*, vol. 38, February 2002.

[2] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, and T. Schooley, "Evaluating Xen for router virtualization," in *Proc. of ICCCN'07*, pp. 1256–1261, 2007.

[3] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, F. Huici, and L. Mathy, "Fairness issues in software virtual routers," in *Proc. of ACM PRESTO '08*, (New York, NY, USA), pp. 33–38, ACM, 2008.

[4] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, F. Huici, and L. Mathy, "Towards high performance virtual routers on commodity hardware," in *Proc. of ACM CoNEXT '08*, 2008.

[5] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *IEEE Computer*, vol. 38, pp. 34–41, April 2005.

[6] PlanetLab Consortium, "http://www.planet-lab.org/."

[7] N. Chowdhury and R. Boutaba, "Network virtualization: State of the art and research challenges," *IEEE Communications Magazine*, April 2009.

[8] J. He, R. Zhang-Shen, Y. Li, C.-Y. Lee, J. Rexford, and M. Chiang, "DaVinci: Dynamically adaptive virtual networks for a customized internet," in *Proc. of ACM CoNEXT '08*, (Madrid, Spain), Dec. 2008.

[9] N. Hohn, D. Veitch, K. Papagiannaki, and C. Diot, "Bridging router performance and queuing theory," *ACM SIGMETRICS Perf. Eval. Rev.*, vol. 32, pp. 355–366, June 2004.

[10] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," *ACM SIGCOMM Comp. Commun. Rev.*, vol. 34, pp. 281–292, Oct. 2004.

[11] G. Liao, D. Guo, L. Bhuyan, and S. R. King, "Software techniques to improve virtualized I/O performance on multi-core systems," in *Proc. of ACM/IEEE ANCS'08*, (San Jose, CA), November 2008.

[12] Y. Qi, Z. Zhou, B. Yang, F. He, Y. Xue, and J. Li, "Towards effective network algorithms on multi-core network processors," in *Proc. of ACM/IEEE ANCS'08*, (San Jose, CA), November 2008.

[13] Q. Ye and M. H. MacGregor, "Adaptive scheduling to maximize nic throughput in a cots router," in *Proc. of ACM/IEEE ANCS'08*, (San Jose, CA), November 2008.

[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," whitepaper, OpenFlow Consortium, 2008.

[15] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In vini veritas: realistic and controlled network experimentation," in *Proc. of ACM SIGCOMM '06*, (New York, NY, USA), pp. 3–14, ACM, 2006.

[16] German Lab Project, "http://www.german-lab.org."

[17] GENI, "http://www.geni.net/."

[18] National Research Council, "Looking over the frence at networks: A neighbor's view of networking research." National Academy Press, 2001. Washington DC.