*I.J. Information Engineering and Electronic Business*, 2009, 1, 17-24
Published Online October 2009 in MECS (http://www.mecs-press.org/)

# A Virtualized Network Architecture for Improving QoS

Yanfeng Zhang
School of Information Science and Engineering
Northeastern University, Shenyang, China
Email: threewells14@gmail.com

Cuirong Wang and Yuan Gao
Department of Computer Engineering
Northeastern University at Qinhuangdao, Qinhuangdao, China
Email: {wangcr, gao}@mail.neuq.edu.cn

***Abstract*—Recently, researches have shown that today's best-effort Internet with "one fits all" principle comes to an impasse. Addressed to the different QoS requirements of today's network services, we propose a new network architecture based on network virtualization technology for improving Internet QoS. It divides a "thick" network into multiple "thin" virtual networks deploying on the same substrate, each is customized with a special designing goal and runs a customized protocol. Then the traffic with different QoS requirements is classified at the ingress router, and distributed to different virtual networks, which are the most suitable for carrying the special traffic. Also, we can deploy a service on multiple virtual networks, and making them working collaboratively to achieve a better QoS. To verify this idea, a prototype is implemented in LAN-scale network. By some simply designed experiments for comparison, we observe that, by use of our network architecture, service provider with specific QoS requirement can take its choice to choose appropriate virtual networks to achieve better QoS performance.**

***Index Terms*—network virtualization, QoS requirements, traffic differentiated**

## I. Introduction

The advent of broadband networking technology has dramatically increased the capacity of packet-switched networks from a few megabits per second to thousands of megabits per second. This increased data communication capacity allows new applications such as video conferencing and Internet telephony. These applications have diverse QoS requirements. Some require stringent end-to-end delay bounds; some require a minimal transmission rate; some require low loss rate; others may require high throughput. Our future network is QoS-based data networks. Aiming at satisfying these QoS requirements, researchers enhance today Internet's QoS system continuously, and a large number of protocols addressing to QoS problems were proposed. However, some of these solutions are complicated and hard to implement, some are at great cost of resources, others still do not achieve QoS guaranteed goal. It seems that today's Internet reaches an impasse [1].

Virtualization is a general strategy to resolve many problems in computer science. For an individual host, virtualization essentially lets a single computer do the job of multiple computers, by sharing the resources of a single computer across multiple environments. Virtual servers let you host multiple operating systems and multiple applications. While network virtualization as shown in Figure 1, is similar to computer virtualization, network resources, including bandwidth and router's CPU and memory being divided into slices.

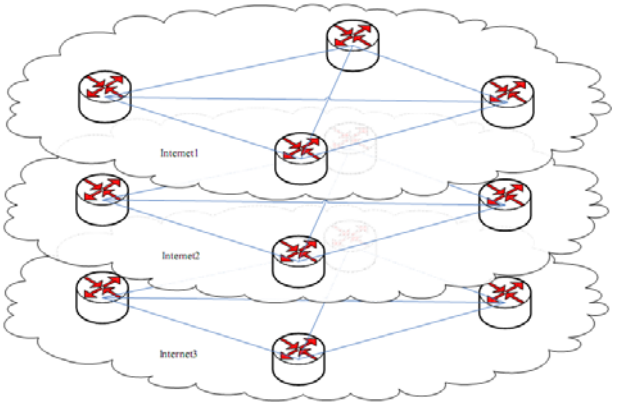


Figure1. Network virtualization

Virtualization is quite almighty. It not only makes one computer like many computers, but also makes many computers like a single computer. In the same way, virtualization makes a single network like many networks (e.g. DiffServ [2]), as well as makes many networks like a single network (e.g. multi-path transmission [3]). Indeed, virtualization is even much more meaningful to network than to a single computer, and numerous researches focused on network virtualization have proposed [4] [5] [6] [7] [8]. Addressed to end-to-end services, many troubles on today's Internet will be resolved by network virtualization.

In broad sense, router as a main network element is a particular computer, while network virtualization is just realized through router virtualization. We expect to virtualize router in a similar way as virtualzing a

Manuscript received Febuary 28, 2009; revised June 2, 2009; accepted July 25, 2009.

 

computer, for generating several routing processes that are separated from each other.

Our work is addressed to design a virtualized network architecture, and to propose the virtualized router architecture in high level. As shown in Figure 2, virtual networks provider provides virtual networks supermarket, where several virtual networks with specific designed goals are prepared for the service providers to choose. Service provider can lease a particular virtual network to deploy its service at a certain price. Also service provider can lease several virtual networks for backup use, which are running collaboratively for a service.

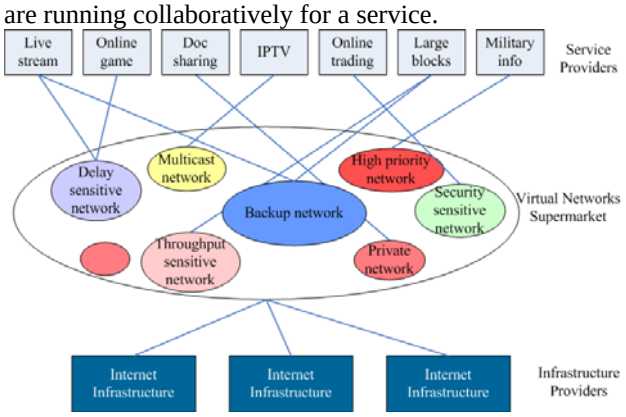


Figure 2. Virtual Networks Supermarket

In addition, on behalf of verifying this idea, this paper presents a prototype on local VINI [9], which is a LAN scale virtual networks testbed imposing OpenVZ [10] to split network resource, and implement these functions by using Click modular router [11]. We also would like to bring forward some novel commercial and operation models based on virtualized networks to discuss our proposal's application prospect.

In the following of this paper, section II gives related work. Section III presents the QoS challenges of the current Internet and shows how multiple virtual networks are suitable for improving network QoS. Some design details of this network architecture are introduced in section IV. Section V implements a prototype of this idea and several related experiments for verifying are provided in section VI. Section VII concludes.

## II. Related Work

Recently, network virtualization has been a hot research topic in research communities and commercial organizations. For research perspective, it can be used for building experimental platforms that run multiple virtual networks in parallel [4] [5] [6]. While for commercial perspective, researchers have proposed that future networks can run multiple virtual networks on a shared physical infrastructure; each virtual network is logically separated and can be customized for particular traffic class, with the substrate providing separate resources for each virtual network. Each application or service can run its own protocols without disturbance from traffic on other virtual networks. And many researchers have already focused on how to build and run an efficient virtualized network infrastructure [12] [13] [14] [15] [16].

Network virtualization brings us more chances to improve today's Internet. The authors of CABO architecture [6] firstly argue that the future Internet should support two separate entities: infrastructure providers (who manage the physical infrastructure) and service providers (who deploy network protocols and offer end-to-end services). CABO decouples service from infrastructure, it pushes this design to its logical conclusion by allowing service providers to offer a wide range of end-to-end services and new network architectures. CABO enables better network services and gives more robust management and operations. Cabernet [17] based on CABO, introduces a connectivity layer, which uses virtual links purchased from infrastructure providers to run virtual networks with the necessary geographic footprint, reliability and performance for the service providers. While in this paper, we advise introducing virtual networks supermarket for service providers to choose.

## III. Improving Network QoS

Many services, such as video conferencing, media streaming, online gaming, Voice over IP (VoIP), and online trading have stringent requirements on network QoS. Unfortunately, today's Internet does not provide satisfactory QoS to end users. There are still some instabilities that threat QoS.

### A. QoS Challenges of Current Internet

**Single network carrying diverse traffic:** The current Internet carries many different types of services, including voice, video, streaming music, web pages and email, many of the proposed QoS mechanisms that allowed these services to co-exist were both complex and failed to scale to meet the demands of the Internet. The "one fits all" principle is now not suitable for network/service diversification. Some services need low delay QoS mechanism while some services need high throughput QoS mechanism, and their QoS requirements may be conflict with each other. Hence, it is very hard for the researchers to propose an all-satisfied QoS mechanism to deal with the problem.

**Unpredictable network events:** Network events (e.g. link failure and router lapse/power down) may occur anytime. Although current routing protocols (interior gateway protocols) are answerable for this urgency, both distance vector algorithms (RIP) and link state algorithms (OSPF and IS-IS) have mechanisms to keep away from the unavailable node or path. But it may take intolerable time to wait for network re-convergence, this time is determined by router's configuration and network size. For the network service, it may make a long time disconnection and severe performance degradation. These network events are unpredictable, and every network element is under the danger.

**Temporal traffic congestion:** As resource allocation of current Internet may be illegitimate, a part of the Internet that may be congested, even if other parts are idle. In addition, numerous measurement studies have shown that today's P2P and video traffic often create traffic burst,

 

which congests Internet seriously [18]. More specifically, a transmission path may traverse the congested part, making it more congested. While To a high-QoS required service, it cannot tolerate packets losses due to network congestion.

*B. Multiple Customized Virtual Networks Opportunities*

While we have several QoS challenges in a single network, there are several characteristics of multiple networks given to solve them:

**Multiple networks for carrying diverse traffic:** We have multiple virtual networks available for carrying different kinds of traffic, with different QoS mechanisms deployed on. These virtual networks give us the opportunity of classifying traffic and distributing it to different direction or networks, which is customized for transmitting some kind of traffic. This work includes designing different QoS mechanisms, including routing protocols, transmission protocols and so on. Through this customization, we can achieve the goal of better QoS and flexible, scalable Internet.

**Backup network for fault tolerance:** In engineering, fault-tolerant design, also known as fail-safe design, is a design that enables a system to continue operation, possibly at a reduced level (also known as graceful degradation), rather than failing completely, when some part of the system fails. While multiple virtual networks can provide network redundancy, that means having backup network which automatically "kick in" should the main virtual network breakdown. For end user, when error occurs, they just enjoy a graceful degradation rather than stopped service. In the future fragile virtual network deployment, building fault tolerant system allows our nonstop reliable service.

**Prepared multiple paths:** Most routing protocols select only one path between two ends. Researchers have proposed that it would be more efficient and robust if routers could flexibly divide traffic over multiple paths [19] [20], and it is useful in a variety of contents including improving security, reacting to failures, and balancing load. Actually, most of today Internet's challenges can be solved by introducing multipath routing. Unfortunately, much of the existing multipath opportunities in the Internet are not currently exploited, due to challenges in scalability and incentive compatibility. In the control plane, exchanging path-level or link-level information will consume extra bandwidth and processing resources, also computing multiple paths requires more computational power. In the data plane, the forwarding tables at the source (though not necessarily the intermediate routers) will have multiple entries for each destination, thus consuming more memories, while multiple virtual networks have inherent solution to these problems. Providing multiple networks means providing multiple paths, each virtual network have a path from source to destination. Since the link weight setting always be different in different virtual networks, the paths from source to destination in different networks always be physical disjoint too, along the other path traffic can bypass the error hardware node.

**Separate resources:** When different types of traffic coexist over the same network substrate, each virtual network could control a subset of resources at each node and link (e.g. CPU, memory, and bandwidth). When a certain virtual network is congested, resources allocated on that virtual network are not enough, but rich resources available on the other virtual networks. Hence, on account of resources are separated, we can make a efficient use of resources over the other virtual networks, balancing traffic load between multiple virtual networks.

## IV. Architecture Description

In this section, we introduce our architecture. In subsection A we describe the architecture from network view and in section B we describe the design of router in detail. Subsection C proposes the characteristics of this architecture.

*A. From Network View*

From network view, network virtualization divides network into a set of virtual networks as shown in Figure 1, with each virtual network in charge of a specific traffic. This partition means network resources isolation, including any network element. Each virtual network correspond to a certain type of QoS class is designed to satisfy this QoS requirement. This target-clear design is mainly implemented on router by various QoS control mechanisms in different virtual routers.

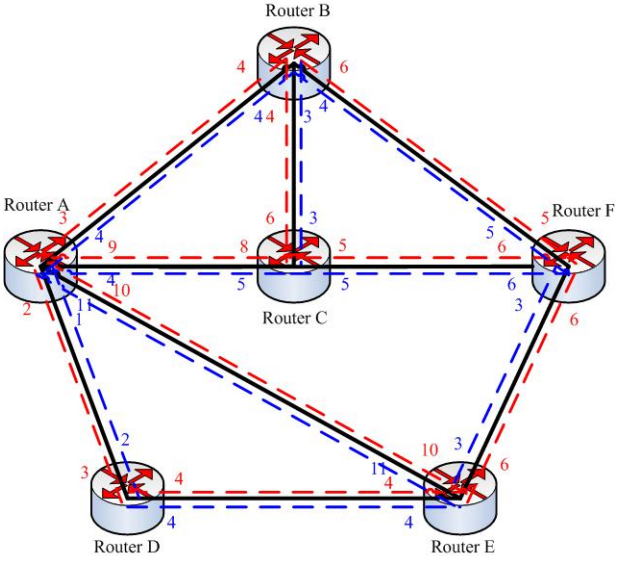


Figure 3. Different link cost settings in different virtual networks

Another issue from network view is routing. QoS routing needs to identify end-to-end paths where there are enough available resources to guarantee performance requirements in terms of metrics as loss, delay, call blocking, numbers of hops, reliability, as well as bandwidth optimization [21]. However, due to multiple networks coexist, QoS routing can be implemented in an elementary and simple way. We set different costs in different virtual networks in terms of QoS requirements, and then using shortest path routing based on this differentiated service cost. Compared with shortest-path routing, our new routing method exploits signaling traffic,

 

but compared with QoS routing, it saves complex resources computation thanks to the benefits of network virtualization.

A routing example is given in Figure 3, red links and routers' partial resources construct a delay-sensitive network, and blue links and routers' partial resources construct a loss-sensitive network. Link costs are labeled in Figure 3, just like OSPF, but instead of giving real link costs, virtual link costs are given. We can distribute a certain type of packets, which are distinguished by some embedded codes in packet header, to the appropriate virtual network. The delay-sensitive packets pass through router A with its destination prefix of router F. In terms of OSPF in delay-sensitive network (red links), packets will be routed along the path A-B-F, with cost 9. By contrast, loss-sensitive packets passing through router A with the same destination router F, will be routed along the path A-D-E-F in terms of loss-sensitive network (blue links) OSPF link cost settings.

*B. Router Design*

As the most crucial component in Internet, router carries out many vital works. As mentioned above, router takes charge of forwarding packet based on its destination address by querying forwarding table. We would like to forward a packet on not only destination address but also on its QoS requirement. Each router is composed of a set of virtual routers, taking responsibility of forwarding packets labeled different QoS requirement types. The provision of a single class of QoS-controlled service requires coordinated use of admission control, traffic access control, packet scheduling, buffer management, and so on. These series of QoS control mechanisms based on QoS requirement, are provided in a specific virtual router. For instance, for real-time service, this delay-sensitive traffic will go through the exact virtual router that is designed for real-time flows, in which admission control mechanism, packet scheduling mechanism and other QoS control mechanisms adapted for real-time traffic are settled. A schematic diagram illustrates the forwarding process in a virtual router in Figure 4.

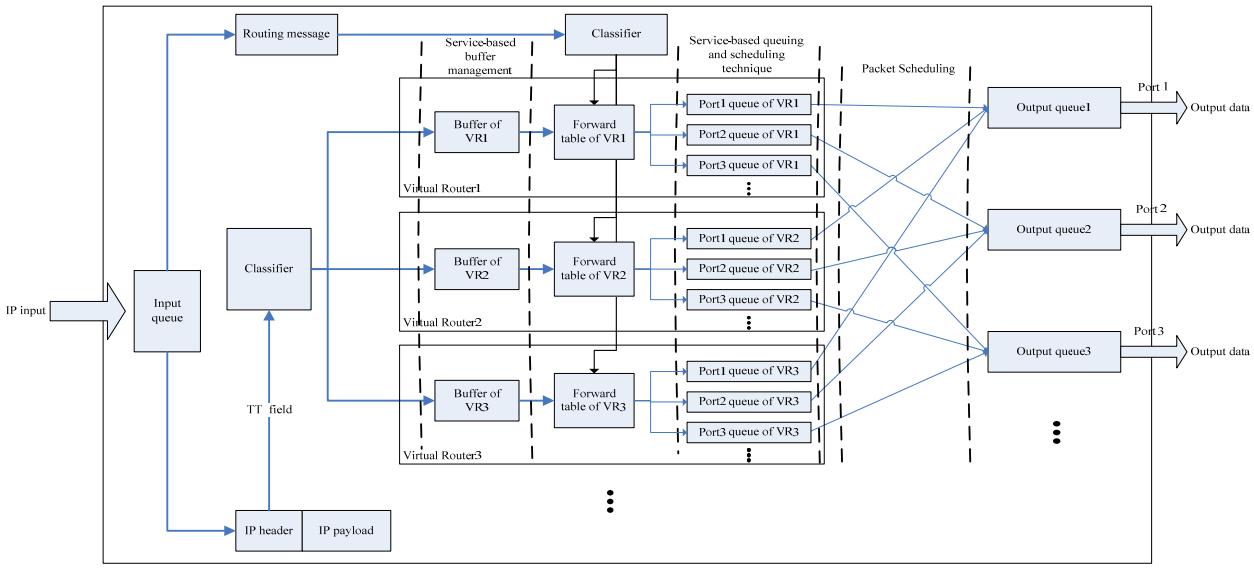


Figure 4. Router architecture design

On receipt of a packet, the packet is first directed to the packet classifier, which reads its traffic type field (TT) in the packet header to determine which virtual router should be the receiver.

After classified to different virtual routers, packet is sent to a series of corresponding QoS control processes encapsulated in a specific virtual router. Buffer management, which is selected based on its relating QoS requirement of virtual router, is the first QoS control process in virtual router. As we know, buffer management is the strategy for deciding when and how to discard packets to avoid network congestion. It is also related to delay metric in end-to-end performance measurement, which gives us much higher probability to satisfy proper QoS requirement.

Then, IP destination address in the packet header is read and be proceeded to determine the network prefix by using the address mask. It then uses the network prefix to look up the outgoing port/line from its forward table, which is the QoS oriented shortest path route to the destination network. Forwarding table is generated by routing messages from the virtual routers located on other real routers in the same virtual network. Before entering into virtual routers, these messages are identified to determine which virtual router should be their receivers. In other words, routing message of service type X can only affects virtual router X's forwarding table.

On finishing matching process in forwarding table, packet is sent to a certain port, each port has a queue correspond to it. Since there are packets having priority over others, even though they are the same QoS requirement type, fair packet queuing and scheduling is of the essence. Yet, this QoS control mechanism will be implemented much simpler based on network virtualization, for there is a packet differentiating process in advance (i.e. packet classifier module).

 

Outside virtual router, another packet scheduling procedure exists. We recommend that, based on disposal in virtual router, this packet scheduling process should be simple, just considering importance proportion of these QoS requirements of specific port. And this is related to QoS oriented OSPF settings mentioned above, router with lower virtual link cost is that apt to forward the corresponding packets (i.e., with higher importance proportion of this QoS requirement). Finally, packet is sent outside.

### *C. Advantages Analysis*

In summary, this kind of router design is:

- Clear for work assignment. We assign each type of QoS control mechanisms to a certain virtual router, and each router is specialized the similar work, which looks much more decent.
- Scalable. Scalability seems the common benefit brought from virtualization. We can add or remove any special routing process addressed to a certain QoS requirement, by adding or deleting a virtual router commodiously. With the prevalence of programmable router, virtualization embedded in router is a feasible technique and scheme.

Conducive to achieve robust network. By the help of virtualization, if a virtual router taking charge of service X, due to poor design, is in the danger of work disorder, and even to the edge of breakdown. This disorder virtual router will not affect other virtual routers, this making sure the network running well, in avoidance of network performance degradation.

In order to verify these advantages of this architecture, we tend to illustrate through a service instance that can be deployed on local VINI [9].

## V. Prototype Implementation on Local VINI

As a proof of concept, we implemented a prototype based on local VINI, synthesizing many components created by the networking research and open source communities. And we would like to introduce some details of local VINI first.

VINI [4] is a virtual network infrastructure that allows network researchers to evaluate their protocols and services in a realistic environment that also provides a high degree of control over network conditions. VINI allows researchers to deploy and evaluate their ideas with real routing software, traffic loads, and network events. To provide researchers flexibility in designing their experiments, VINI supports simultaneous experiments with arbitrary network topologies on a shared physical infrastructure. While local VINI is a LAN-scale VINI including several linux machines, it uses OpenVZ [10] to isolate resources between virtual networks, and supports multiple non-interferential virtual networks running simultaneously, with each network running a Quagga [22] routing instance.

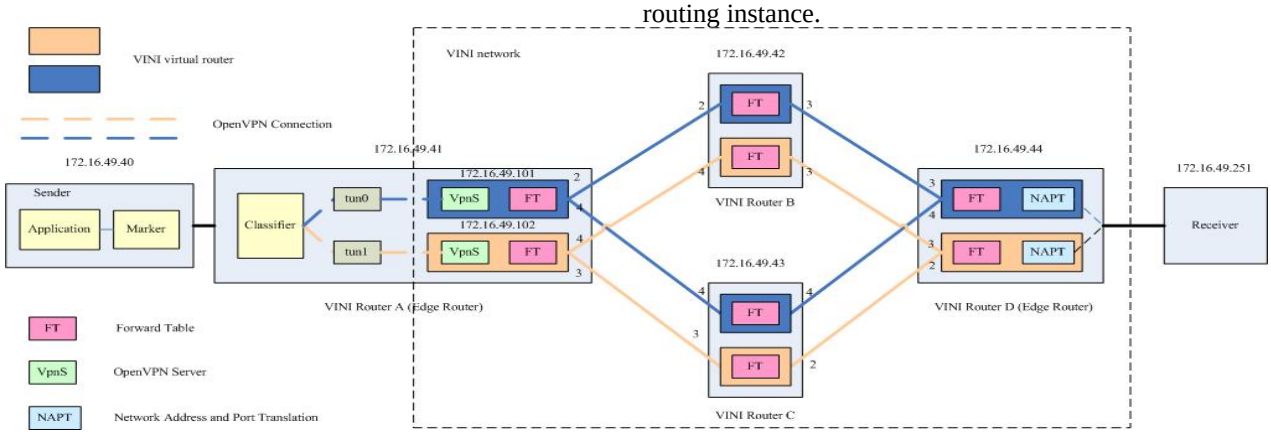


Figure 5. Prototype implemented in local VINI

As described above, the routing instance takes regard of application's QoS requirement, and routes packet according to OSPF protocol in different virtual networks (i.e., multiple OSPF instances coexist in different virtual networks). Philosophy is easy to perceive, but implementation should always be an annoying work. In this paper, we seize on VINI, and in virtual network, we can implement different QoS control mechanisms using Click [11]. We implement the prototype by a set of modifications of local VINI, which is shown in Figure 5.

The prototype is composed of three parts, the sender (i.e. the host with address 172.16.49.40), the local VINI network (i.e. VINI router A, B, C, D are included), and the receiver (i.e. host 172.16.49.251). VINI network is composed of four real/hardware nodes, and each with two virtual nodes. Four blue virtual nodes belong to a VINI virtual network, while another four orange virtual nodes belong to another VINI virtual network. Note that, the virtual links' OSPF costs in different virtual network are set differently, by which we can implement different QoS policies on different sets of virtual nodes to achieve QoS-differentiated goal.

The first step in defining a QoS policy is to identify the traffic that is to be treated differently. This is realized by classifying and marking. Only after traffic is positively identified can policies be applied to it. At the sender side, we designed a marker to mark traffic from applications before sent out. In addition, to illustrate the problem but not to give complete mechanisms, we simply give two different QoS requirements traffic, loss-sensitive and

 

delay-sensitive, with different codes. Click tool is used to mark IP packet's TT field. We set sender host's default gateway as VINI router A (i.e. host 172.16.49.41), which also works as an edge router. This enables every packet to receiver (172.16.49.251) could get through VINI network.

We create two VINI virtual routers on VINI router A (edge router), where two *openvpn* [23] servers in both virtual routers are running waiting for *openvpn* connections. Afterward, two *openvpn* connections (i.e. from host 172.16.49.41 to 172.16.49.101 and from 172.16.49.41 to 172.16.49.102, 172.16.49.101 and 172.16.49.102 are VINI virtual nodes IP addresses) are established, at the same time two *tun* devices are set up, tun0 for host 101 and tun1 for host 102 respectively. Classifier is used to classify traffic to different VINI virtual networks through different *tun* devices. We direct the first type of traffic to 172.16.49.101 through tun0 device, and direct the second type of traffic to 172.16.49.102 through tun1 device.

There are two virtual networks in our prototype, and we scheme that blue virtual nodes are for loss-sensitive traffic and that with orange nodes are for delay-sensitive traffic. Two virtual routers coexist in every host, and they forward designated traffic distinctly. For example, router B is more suitable for forwarding loss-sensitive traffic, so its OSPF cost is lower than router C in blue virtual network. Likewise, router C is more suitable for forwarding delay-sensitive traffic, so its OSPF cost is lower than router B in orange virtual network.

On the egress router in VINI network, VINI router D with IP address 172.16.49.44 as shown in Fig. 4, VINI uses Click to implement NAPT (network address and port translation) for exchanging packets with external hosts that have not opted-in, and packets destined for an external host will be forwarded to these egress points (i.e. two virtual nodes with NAPT function in router D). This involves rewriting the source IP address of the packet to the egress node's public IP address, and rewriting the source port to an available local port. Therefore, no matter what this traffic's QoS requirement is, it will reach the destination host 172.16.49.251. Note that, for end users, they do not know the exact path their packets passing through. Only marking QoS requirement type should end users do, leaving QoS oriented routing system to determine the best routing path for their peculiar traffic on local VINI.

## VI. Experiments

In this section, we run two experiments. The first one is to demonstrate that our prototype can support QoS differentiated function. The second one is to test our prototype's react ability to network state's dynamic change. They are both simply and easily implemented.

### *A. QoS requirement differentiated function*

As we know, there are many QoS control mechanisms, only random early detection (RED) queues are embedded on each virtual node. In order to adapt to either loss-sensitive traffic or delay-sensitive traffic, different parameters of RED are set. For loss-sensitive packet, we expect longer queue, while for delay-sensitive packets, shorter queue is preferred.

We firstly run a series of experiments with different RED parameters to pick up a suited RED configuration for loss-sensitive traffic and a suited RED configuration for delay-sensitive traffic. Each packet is set a constant length, 256 bytes, and 450 packets per second, total 10000 packets. This RED queue is implemented on each VINI virtual node along the transmission path.

TABLE I. Packet losses and delay over different networks

| min | max | p | lost | delay |
|---|---|---|---|---|
| 2 | 10 | 0.02 | 4.5% | 9.2 ms |
| 10 | 80 | 0.02 | 0.49% | 118.3 ms |

Table I lists a set of RED settings in different network and the corresponding lost packets as well as transmission delays. Clearly, for different QoS requirements, loss-sensitive or delay-sensitive, our prototype can carries the two types of traffic over different networks to achieve better QoS. For loss-sensitive traffic, we lose 0.49% packets approximately, and get 118.3ms average delay, while for delay-sensitive traffic, we lose 4.5% packets, and get 9.2ms average delay. Besides, we use *tcpdump* at host 172.16.49.42 and at host 172.16.49.43 to detect traffic flow, different types of traffic assuredly get through different paths. In this way, we can realize QoS oriented routing It is notable that, RED buffer management is only one of various QoS control mechanisms, other mechanisms should be added in our prototype as a future work.

### *B. Multiple Networks Prepared for Reaction to Dynamic Network State Change*

We verify our prototype through running an example service, live streaming service over multiple virtual networks. And next, a comparison between live streaming over our network prototype and over traditional Internet is studied.

Live streaming falls into two general types, constant bitrate source (e.g., CBR voice streaming) and variable bitrate source (e.g., VBR video streaming). To evaluate our proposal's performance, they have distinct senses. Hence, we will study the CBR source case and the VBR source case separately. For the CBR source, we can generate the constant rate traffic primitively by using a timer, while for the VBR source, we generate video packets in terms of real video data trace file. We introduce the movie *StarWarsIV*'s trace file and set a part of the trace file as VBR streaming source, whose mean rate is 320Kbps and peak rate is 620Kbps, with frame rate 25Fps. For CBR source, constant rate of 320Kbps is set, with the same frame rate 25Fps.

Basically, two virtual networks are set with the topologies as shown in Figure 6. Both are interior networks running separated OSPF instances (the link weight settings as shown in Figure 6). In each virtual network, there are two available paths from VINI's ingress point to egress point. And $P_{1,1}$ and $P_{1,2}$ represent the first path over network1 and the second path over

 

network1 respectively, while similarly $P_{2,1}$ and $P_{2,2}$ represent the first path over network2 and the second path over network2 respectively.

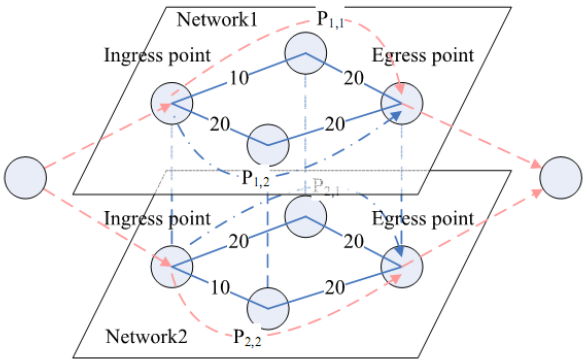


Figure 6. The topology settings of two virtual networks for live streaming experiment

In order to emulate fluctuated network conditions comprehensively, we fluctuate both of the network conditions every other 40s by using *NIST Net*, a network condition shift schedule is designed as Table II shows. Network bandwidths (each path) in both networks will change to the specified value at the specified time. Note that, the first path in network1 (i.e., $P_{1,1}$) is down at 120s and up at 160s. Finally, the experiment terminates at 200s.

TABLE II. NETWORK CONDITION SHIFT SCHEDULE

| Time(s) | 0-30 | 30-60 | 60-90 | 90-120 | 120-150 |
|---|---|---|---|---|---|
| Bw1(Kbps) | 600 | 200 | 400 | $P_{1,1}$ down | $P_{1,1}$ up |
| Bw2(Kbps) | 200 | 600 | 400 | --- | --- |

Ideally, our prototype is self-adaptive to response different network conditions. We verify these self-adapting functions under the fluctuated network environment presented as Table II.

We first use CBR source to generate traffic and transmit it over our prototype. Figure 7 illustrates the traffic rates over the first network and the second network, with total rate 320Kbps and initial split ratio of 1:1. The available bandwidth of network1 is higher than that of network2 during first 30s, but neither of them is capable of carrying all traffic by itself, so the traffic is allocated more on network1. It takes several iterations to reach a steady state. At time 30s, the network bandwidths over two networks turn to inversion, and as we see, the traffic rates over two networks get inversed too. But it takes longer time to reach a stable state, and the decreasing/increasing trend is more and more lenitive. At 60s, the bandwidths of two networks are set even, followed by two traffic rates becoming even. When timeline gets to 90s, both network bandwidths are set bigger, but the first path in network1 $P_{1,1}$ (i.e., the transmission path of network1) is breakdown. Due to it will take several seconds for network1's OSPF routing protocol to find another path, all the traffic should be allocated to the second network, and after finding the other path at time 105s, network1 will join to share the traffic load.

Indeed, using CBR source states the problem more clearly than using VBR source, but we also give more reality to the common view by introducing VBR source. In Figure 8, some key points can still be observed. Before 30s, traffic rate over network1 stands above that over network2, while after about 30s traffic rate over network1 under that over network2. Afterward, the rates over network1 and network2 become near each other at 60s-90s. And obviously, we can observe that no traffic allocated on network1 from nearly 95s to 105s. After that, the situation becomes intangible, but the overall trend seems two networks share the traffic equally.

Undoubtedly, our prototype benefits us to a certain extent, it senses network conditions and adapt traffic distribution policy with a definite purpose, but we prefer the benefits quantitatively described. Packet loss is a significant metric to evaluate the reliability of service, a comparison among packet losses only on network1, only on network2, and on our prototype is presented, each network states are configured as Table II.

TABLE III. CBR LOST RATE COMPARISON

| Time (s) | | 0-30 | 30-60 | 60-90 | 90-120 | 120-150 |
|---|---|---|---|---|---|---|
| uni-net1 (%) | total | 0.69 | 62.89 | 42.06 | 75.75 | 38.10 |
| uni-net2 (%) | total | 60.33 | 2.97 | 37.78 | 38.84 | 41.23 |
| multi-net (%) | net1 | 0 | 0.65 | 0 | 7.43 | 0 |
| | net2 | 0.44 | 0 | 0.11 | 8.41 | 0 |
| | total | 0.44 | 0.65 | 0.11 | 15.85 | 0 |

TABLE IV. VBR LOST RATE COMPARISON

| Time (s) | | 0-30 | 30-60 | 60-90 | 90-120 | 120-150 |
|---|---|---|---|---|---|---|
| uni-net1 (%) | total | 8.00 | 53.11 | 28.19 | 62.20 | 29.85 |
| uni-net2 (%) | total | 50.30 | 15.50 | 26.52 | 30.00 | 29.71 |
| multi-net (%) | net1 | 1.13 | 4.04 | 0.61 | 10.02 | 0.49 |
| | net2 | 3.27 | 2.06 | 0.69 | 12.24 | 0.61 |
| | total | 4.40 | 6.06 | 1.30 | 22.25 | 1.10 |

Table III shows CBR streaming's packet losses in different time intervals over traditional Internet and our proposed network architecture. By transmitting over uni-network (uni-net1 means only over network1 and uni-net2 means only over network2), we can observe that a mass of packets are lost and the losses basically conforms the trend of bandwidth transformation. While imposing multi-net (i.e., our scheme), the situation ameliorates optimistically. The total losses on network1 and network2 descend transparently compared with uni-network. Additionally, the streaming service is not affected by a single network breakdown, it will not interrupt unless all the networks collapse. Table IV presents the VBR streaming's packet losses in a similar way. Albeit the losses are more than CBR streaming resulted from traffic burst, the overall benefits can be also obtained greatly.

The results of these streaming service experiments show that our prototype is competent for carrying our idea. In the case of multiple networks, deploying service

 

on several virtual networks can reduce packet losses significantly, and offers us much more reliability.

## VII. Conclusion

This paper propose a virtualized network architecture for improving QoS, which differentiates traffic with different QoS requirements and carries different traffic on customized virtual networks. It also can impose multiple networks to serve for a single service. A prototype is also presented for verifying this idea. And then, by running several experiments on it, we argue that our proposed virtualized network architecture is competent for future Internet architecture. At the same time, we will enhance our prototype with more functions as our future work.

## References

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet Impasse through Virtualization," *Computer*, vol. 38, no. 4, pp. 34-41, 2005.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differenteated Services", *RFC 2475*, 1998.

[3] J. He, and J. Rexford, "Toward Internet-Wide Multipath Routing", *Network*, vol. 22, no. 2, pp. 16-21, 2008.

[4] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI veritas: realistic and controlled network exerimentation", *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp: 3-14, 2006.

[5] "Geni design principles," *Computer*, vol. 39, no. 9, pp. 102-105, 2006.

[6] N. Feamster, L. Gao, and J. Rexford, "How to lease the internet in your spare time," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 61-64, 2007.

[7] E. Keller, R. Lee, and J. Rexford, "Accountability in hosted virtual networks," in *Proceedings of ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures (VISA)*, New York, NY, USA: ACM, 2009.

[8] Y. Wang, E. Keller, B. Biskeborn, J. Van der Merwe, and J. Rexford, "Virtual routers on the move: live router migration as a network-management primitive," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 231-242, 2008.

[9] Y. Zhang, H. Wang, C. Wang, and Y. Gao, "Design and implementation of a network testbed based on virtualization," in *CTC '08: Proceedings of the 2008 China Testing Conference*, 2008.

[10] (2009) Containers virtualization open source project: Openvz, [Online]. Available: http://wiki.openvz.org

[11] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263-297, 2000.

[12] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17-29, 2008.

[13] E. Keller and E. Green, "Virtualizing the data plane through source code merging," in *PRESTO' 08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*. New York, NY, USA: ACM, 2008, pp. 9-14.

[14] M. Caesar and J. Rexford, "Building bug-tolerant routers with virtualization," in *PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*. New York, NY, USA: ACM, 2008, pp. 51-56.

[15] J. Fu and J. Rexford, "Efficient ip-address lookup with a shared forwarding table for multiple virtual routers," in *CoNEXT '08: Proceedings of the 2008 ACM CoNEXT conference*. New York, NY, USA: ACM, 2008.

[16] 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 *CoNEXT '08: Proceedings of the 2008 ACM CoNEXT conference*. New York, NY, USA: ACM, 2008.

[17] Y. Zhu, R. Zhang-Shen, S. Rangarajan, and J. Rexford, "Cabernet: connectivity architecture for better network services," in *CoNEXT '08: Proceedings of the 2008 ACM CoNEXT conference.* New York, NY, USA: ACM, 2008.

[18] S. Ohzahata and K. Kawashima, "A study on traffic characteristics evaluation for a pure p2p application," in *PDP '08: Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network Based Processing (PDP 2008)*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 483-490.

[19] D. Jurca and P. Frossard, "Media flow rate allocation in multipath networks," *IEEE Trans. On Multimedia*, vol. 9, no. 6, pp. 1227-1240, Oct. 2007.

[20] P. Frossard, J. C. de Martin, and M. R. Civanlar, "Media streaming with network diversity," in *Proc. IEEE*, no. 1, Jan. 2008, pp. 39-53.

[21] G. Apostolopoulos, R. Guerin, S. Kamat, T. Orda, S.K. Tripathi, "Intradomain QoS routing in IP networks: a feasibility and cost/benefit analysis", *Network*, vol. 13, no. 5, pp. 42-54.

[22] (2009) Quagga routing suite. [Online]. Available: http://www.quagga.net

[23] (2009) The opensource vpn : Openvpn. [Online]. Available: http://openvpn.net

**Yanfeng Zhang** received the Master degree and Bachelor degree in computer science from 2008 and 2005 respectively.

He is currently a Ph.D. student in Computer Application Techniques in school of Information Science and Engineering, Northeastern University, China. And he is doing research work in Next Generation Internet Technologies Lab in Northeastern University at Qinhuangdao. His research interests include network virtualization, and data center networking. He was working in NEUSoft Research Center as an intern software engineer between March 2006 and July 2007.

Mr. Zhang is an IEEE student member, including IEEE Communication Society member and IEEE Computer Society member.

**Cuirong Wang** received the Ph.D. degree in Software and Theory from Northeastern University, China in 2004.

She is a professor in Department of Computer Engineering in Northeastern University at Qinhuangdao, China. Her research interests include routing protocol, network security and sensor networks.

Prof. Wang is a senior member of China Computer Federation and a member of China Fault-Tolerance Committee.

**Yuan Gao** is a professor in Department of Computer Engineering in Northeastern University, China. His research interests include network virtualization, network security, and video streaming.

Prof. Gao is a senior member of China Computer Federation and a member of China Fault-Tolerance Committee.

 