Network-resource Isolation for Virtualization Nodes[*]

Yasusi Kanada
Central Research Laboratory, Hitachi, Ltd.
Yasusi.Kanada.yq@hitachi.com

Kei Shiraishi
Network Solution Division, Hitachi, Ltd.
shiraishi_kei@itg.hitachi.co.jp

Akihiro Nakao
The University of Tokyo
nakao@iii.u-tokyo.ac.jp

***Abstract* – One key requirement for achieving network virtualization is resource isolation among slices (virtual networks), that is, to avoid interferences between slices of resources. This paper proposes two methods, per-slice shaping and per-link policing for network-resource isolation (NRI) in terms of bandwidth and delay. These methods use traffic shaping and traffic policing, which are widely-used traffic control methods for guaranteeing QoS. Per-slice shaping utilizes weighted fair queuing (WFQ) usually applied to a fine-grained flow such as a flow from a specific server application to a user. Since the WFQ for fine-grained flows requires many queues, it may not scale to a large number of slices with a large number of virtual nodes. Considering that the purpose of NRI is not thoroughly guaranteeing QoS but avoiding interferences between slices, we believe per-slice shaping suffices our objective. In contrast, per-link policing uses traffic policing per virtual link. It requires less resource and achieves less strict isolation between hundreds of slices. Our results show that both methods perform NRI well but the performance of the former is better in terms of delay. Accordingly, per-slice shaping is effective for delay-sensitive services while per-link policing may be sufficiently used for the other types of services.**

I. Introduction

In network virtualization, it is important to avoid resource interference, e.g., concerning communication bandwidth and delay, so that a single slice may not disrupt the whole infrastructure. To avoid this type of interference, a method for network-resource isolation (NRI) must be developed. Two types of traffic-control functions usually used for QoS guarantee, i.e., traffic shaping and traffic policing, can be used for NRI. Traffic shaping is suitable for services more sensitive to packet loss, but it is relatively expensive because a queue must be allocated to each virtual link (per link) to guarantee QoS. This requirement may limit the number of slices that can be allocated in the physical network. However, because the purpose of resource isolation is to avoid interference between slices (but not to guarantee QoS thoroughly), per-slice queue allocation, instead of per-link allocation, is sufficient. In contrast, traffic policing is a low-cost method for avoiding interference concerning bandwidth among slices that shares a queue, and it is suited for services more sensitive to delay but less sensitive to packet loss.

To implement NRI at a reasonable cost using these functions, two methods are proposed in this study: the per-slice shaping method that allocates a weighted fair queuing (WFQ) [Par 93] based queue to each slice and the per-link policing (per-virtual-link policing) method that allocates a WFQ based queue to each slice class (slice collection). Our final goal is to perform NRI among hundreds of slices with 1-kbps to 10-Gbps traffic using these methods.

II. Virtualization Platform, VNode, and Slice

When many users and systems share a limited amount of resources on computers or networks, virtualization technology creates the illusion that each user or system owns resources of their own. Concerning networks, WANs are virtualized by using VPNs. Nowadays, networks in data centers are virtualized using VLANs while servers are virtualized using VMs. In new-generation network research projects such as PlanetLab [Tur 07] or GENI [GEN 09], it is necessary to develop virtualization technology that makes it possible to build network environments where slices are isolated from one another so that they may develop new generation network protocols without disrupting the other slices.

Network virtualization is realized through slices and the *virtualization platform* that manages slices. A project called the Virtualization Node Project (VNP) has developed virtualization-platform architecture (see **Figure 1**) and a high-performance fully functional virtualization testbed [Nak 10b]. The major component in this architecture is *VNode* (virtualization node), which forwards packets on the platform. Each packet on the platform contains a virtualized packet on a slice. VNodes are connected by tunnels using a protocol such as GRE [Far 00]. Therefore, a slice with free topology can be created. Each VNode consists of the following three components.

- *Programmer* is a component that processes packets on the slices. Slice developers can program programmers.
- *Redirector* is a component that can forward or route packets on the virtualization platform and forward (redirect) packets from another VNode to a programmer or forward packets from a programmer to another VNode.
- *VNode Manager* is a software component that manages the VNode according to instructions from the Domain Controller (DC), which manages the platform domain.

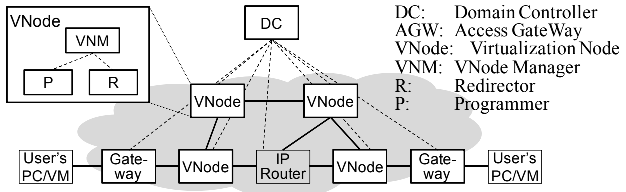


Figure 1. Physical structure of virtualization platform

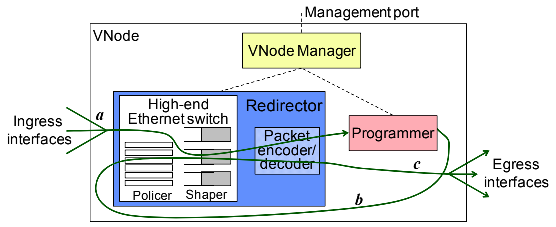


Figure 2. Policing and shaping in a VNode

Figure 2 shows a more detailed internal structure of a VNode. There exists a pair of packet encoder and decoder between the redirector and the programmer. The decoder converts the VNode-external data format to the internal format and the encoder converts vice versa. Traffic that comes to the ingress interfaces (to the left) follows the curved line and passes through the shaper, the decoder, the programmer, the policer, the shaper again, and the encoder, and is outputted through the egress interfaces (on the right).

In the model developed by VNP, a slice consists of the two types of components [Nak 10a].

- *Node Sliver* represents computational resources that exist in a VNode (in a programmer). It is used for node control or protocol processing with an arbitrary packet format. It is generated by slicing physical computational resources.
- *Link Sliver* represents resources of a virtual link that connects two node slivers. It is generated by slicing physical network resources such as bandwidth.

[*] This paper is an extended version of a poster paper [Kan 12].

III. Resource Isolation and Specification Method

A. Resource Isolation

The user of a slice should be able to use the slice without being affected by the behavior of other slices created on the same virtualization platform. For this purpose, methods for isolation must be developed, and the slice must be isolated from other slices. Isolation can be classified as the following two types.

- *Resource isolation* is a function that enables a slice to use the required resources to provide the expected performance even when congestion occurs on the physical network or on other slices.
- *Security isolation* is a function that avoids disruptions and intrusions against a given slice caused by adversaries.

This paper focuses on resource isolation.

The virtualization platform has three types of resources: network, computational, and storage. Three types of resource isolation can thus be classified:

- Network-resource isolation (NRI)
- Computational-resource isolation (CRI)
- Storage-resource isolation (SRI)

Although a programmer must isolate slices of traffic which share a network interface card (NIC) or processor to guarantee NRI, redirectors play the most important role because they measure and control traffic between VNodes.

Traffic control for NRI is required in every VNode because every node sliver may generate excessive traffic. NRI can be implemented by using QoS functions of the VNodes. It is assumed that each VNode has bandwidth-control functions, especially shaping and policing. However, the priority-based control function may also be used for NRI. For example, if a VNode contains a router, NRI may be implemented by using the DiffServ function of the router.

B. Specification of Network-resource Isolation

It is necessary to specify bandwidth and or the maximum burst size in defining a link-sliver with NRI, as is specified in virtual leased lines. Several examples of link-sliver definitions, which are parts of slice definitions, are shown below. The first example (in **Figure 3**a) is a definition without a bandwidth specification, that is, a best-effort link-sliver. The link sliver has two virtual ports, i.e., end points: port0 and port1. A link sliver without resource specification is similar to a best-effort path, thus, is subject to disruptions caused by the other slices and non-virtualized traffic. The second example (in Figure 3b) is a definition with a simple bandwidth specification. The bandwidth is 30 Mbps, and the maximum burst size is 10 kB. In the third example (in Figure 3c), bandwidth and burst-size parameters are specified for each direction. The bandwidth from port0 to port1 is 30 Mbps, and that of reverse direction is 20 Mbps.

```
<linkSliver type="link" subtype="GRE" name="LinkSliver1">
  <vports><vport name="port0" /><vport name="port1" />
  </vports></linkSliver>
```

(a) Example without a bandwidth specification

```
<linkSliver type="link" subtype="GRE" name="LinkSliver1">
  <vports><vport name="port0" /><vport name="port1" />
  </vports>
  <resources>
    <resource key="bandwidth" value="30M" />
    <resource key="burstSize" value="10k" /></resources>
</linkSliver>
```

(b) Example with a bandwidth specification

```
<linkSliver type="link" subtype="GRE" name="LinkSliver2">
  <vports><vport name="port0" /><vport name="port1" />
  </vports>
  <resources from="port0">
    <resource key="bandwidth" value="30M" /></resources>
  <resources from="port1">
    <resource key="bandwidth" value="20M" /></resources>
</linkSliver>
```

(c) Example with direction-dependent bandwidth specification

Figure 3. Link-sliver definition examples

IV. Methods of Network-Resource Isolation

A. Two Traffic-Control Functions —Shaping and Policing

Two major traffic control functions which are usually used for guaranteeing QoS can be used for NRI. *Shaping* queues packets and limits and schedules the egress traffic by a certain scheduling algorithm such as the leaky-bucket algorithm [Par 93]. If the queue is filled, the packet output may be delayed, and packets may be dropped. *Policing* measures network traffic, usually by using the token-bucket algorithm [Par 93] without accumulating packets, and that drops packets when the bandwidth or the burst size exceeds a predefined limit. Both shaping and policing can be applied to link slivers, slices, or classes of slices. The cost of policing is much lower than shaping.

Shaping may perform better than policing in NRI; namely, it can avoid interference in both bandwidth and in delay. However, it is expensive in high-end nodes because a specialized hardware is required. Therefore, a method for NRI using fewer queues and less-complicated scheduler should be developed. In a shaping-based method, a queue must be allocated to each link sliver to guarantee the bandwidth, and this allocation causes high packet-scheduling overhead. In addition, it may be impossible to allocate a queue to each link sliver because there may be insufficient number of queues in the hardware interface. As a result, the scalability may be sacrificed; namely, the number of slices that can be allocated in the platform may be limited. It is therefore preferable to allocate less queues.

Two methods can be used to reduce the number of queues and the complexity of the scheduler. One is *per-slice shaping* method that uses shaping for isolating slices but allocates a queue to each slice. The other is *per-link policing* method that uses policing for isolating slices.

B. Per-slice shaping

Per-slice shaping, or shaping-based isolation, isolates slices by queuing traffic per slice, instead of queuing it per link, to reduce shaping cost. Policing is not necessary for NRI in this method; packets are not intentionally dropped then. Per-slice shaping can be specified by the following expression.

```
<resource key="performanceIsolation"
          value="shaping" />
```

As explained in the previous subsection, shaping per link is required to guarantee QoS. However, because the purpose of resource isolation is to avoid interference between slices, but not to guarantee QoS thoroughly, we believe per-slice queue allocation, instead of per-link allocation, suffices our objective. It is not necessarily required to avoid interference between flows *within* a slice. For this reason, a queue is allocated to each slice. Although per-slice shaping cannot avoid interference between link slivers of a slice, the flows of different slices can be isolated almost completely by this method. In our prototype, both input packets and output packets of programmers are queued in a per-slice WFQ.

By using this method, the bandwidth used for each slice is guaranteed, so interference between slices is avoided. Because each slice uses a separate queue, not only interference of bandwidth but also correlation of delay between slices can mostly be avoided. Therefore, this method is suited for strict real-time services. The number of link slivers connected to a node sliver is considered to be typically three to five, so the number of queues can be reduced by 70 to 80%.

C. Per-link Policing

The per-link policing method, or the policing-based isolation method, isolates slices by policing traffic per link sliver. Traffic is shaped per slice class or per link-sliver class in this method; namely, two or more slices using per-link policing can share a queue to reduce shaping cost compared to per-slice shaping. The slice classes or link-sliver classes can be defined according to QoS or any other purpose. This isolation may be less strict than that of per-slice shaping. However, per-link bandwidth can be assured in this method.

Per-link policing can be specified by the following expression in the link-sliver definition.

```
    <resource key="performanceIsolation"
              value="policing" />
```

In per-link policing, an ingress network interface with a traffic-policing function is used. Each interface has packet filters. If a filter is applied to a link sliver, it can control the bandwidth per link sliver.

In this method, policing can be used for guaranteeing bandwidth of link slivers that share a shaper. If the total bandwidth is guaranteed by shaping, the bandwidth is properly divided to the link-slivers by using policing and each bandwidth is guaranteed. Hundreds or thousands of slices may share a queue in per-link policing. It is therefore reasonable to implement NRI using per-link policing. Packets are, however, easily dropped by policing, so it is usually not preferable to use policing everywhere.

Per-link policing is more scalable than per-link shaping; that is, there may be thousands or more slices even when a filter is created for each link sliver. Policing can avoid bandwidth interference, although policing cannot avoid interference or correlation concerning delay completely. Therefore, this method is not suited for strict real-time services, but is sufficient for most other types of services, i.e., non-real-time services such as most TCP-based services or weak real-time services.

D. Combination of Per-slice Shaping and Per-link Policing

Per-slice shaping and per-link policing may be combined; slices can be isolated by using both shaping per slice and policing per link-sliver. The method can be called *the per-slice shaping method with (per-link) policing*. This isolation is as strict as that of per-slice shaping, and per-link bandwidth can also be assured. However, packets may be dropped and the shaping cost is higher than per-link policing. This method can be specified by the following expression.

```
    <resource key="performanceIsolation"
              value="shapingAndPolicing" />
```

E. Management of Network-resource Isolation

The DC accepts a slice definition from the slice developer, and it sends link-sliver-creation requests to the VNodes. The DC manages the total bandwidth of each VNode and rejects the slice definition if it contains network-resource requests that cannot be satisfied.

V. Implementation

Three methods for NRI, i.e., per-link policing, per-slice shaping without policing, and per-slice shaping with policing (i.e., the combined method), have been implemented in our VNode prototype. The structure of the VNode and the redirector and the methods for allocating shaping queues are described in this section.

A. Structure of Redirector

The redirector, namely, the packet-forwarding part of a VNode, plays the most important role of the NRI in the VNode. It consists of three components: redirector body (RB), redirector manager (RM), and service module cards (SMCs) (see **Figure 4**). A layer-3 (L3) switch is used for the RB. An SMC is an add-on card, with a network

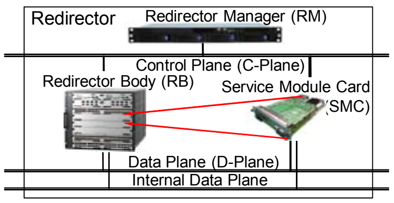


Figure 4. Structure of redirector

processor, installed in the redirector. The RM manages the redirector according to a management request from the VNode manager through an XML-RPC based API. The RM configures and manages RB and SMC through command-line interfaces (CLIs). Link slivers are implemented using GRE tunnels that allow any IP and non-IP protocols. An SMC is used for packet encoding and decoding. The GRE encoder and decoder on the SMC can convert packets at almost 10 Gbps if the packet size is large enough.

B. Allocation of Shaping Queues

Because the SMC may cause bottleneck, packets are queued and scheduled just before the SMC (i.e., packet encoder/decoder as shown in Figure 2). The scheduler is a part of 10-Gbit Ethernet interface between the RB and the SMC. The interface has four WFQs per port as well as queues with high priority and queues with low priority. They are allocated in the following way (see **Figure 5**).

- *Per-slice shaping queues*: Three WFQs are used for link slivers with per-slice shaping with/without policing. The number of per-slice shaping slices is at most three.
- *Per-class shaping queue*: There is only one link-sliver class for per-class-shaping (i.e., per-link-policing) in this implementation. Therefore, one WFQ is used for link slivers with per-link policing.
- *Best-effort queue*: One queue with lower priority is used for best-effort link-slivers. The number of best-effort slices is not limited.

Because best-effort traffic has lower priority than isolated traffic, it may cause starvation; namely, no bandwidth may be allocated to best-effort traffic. To avoid starvation, isolated traffic must always be policed in this implementation; that is, per-slice shaping without policing must be inhibited. In this implementation, only three slices with per-slice shaping can be created, but more slices with per-link policing can be created as described in Section IV E.

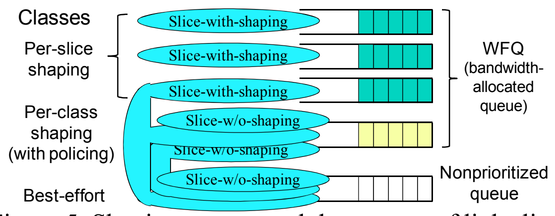


Figure 5. Shaping queues and three types of link slivers

VI. Evaluation

In this section, we show the evaluation result of the round-trip delay, jitter, and packet-drop ratio of four types of slow-path link slivers, i.e., per-slice shaping with policing and without policing, per-link policing, and best-effort. We conduct measurements on a simple network of two VNodes with slow-path node slivers (i.e., Linux virtual machines).[1]

Table 1 lists the measurement results for round-trip delay, jitter, and packet-drop ratio when the bandwidth of the foreground traffic is 90 Mbps except the congestion-less case (the bottom line). Both the averages and standard deviations are shown. It is clear that both packet-drop (bandwidth) and delay interferences are mostly suppressed; there are only small differences between the cases except the no-isolation case. The delay is slightly larger in the case of the per-link policing (1.60) than in the case of the other two methods and the congestion-less case (1.31). This is

[1] All the node slivers in a VNode are allocated to the same PC server in the programmer, thus, nodes slivers are sharing the same 1-Gbps interface. To keep the traffic acceptable to the node slivers, the total traffic in a VNode is shaped to 900 Mbps by the shaper just before the SMC.

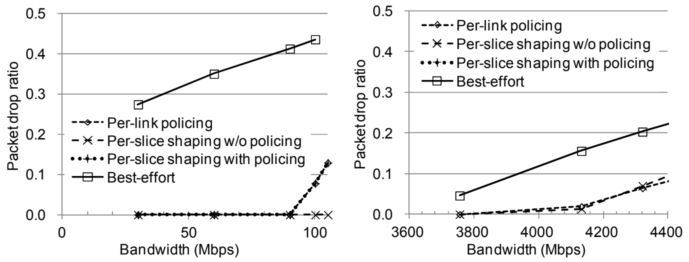
probably because the queue is shared by two flows and the average queue length is larger. However, there are no other significant differences between these isolation methods.

The performance parameters were also measured for different bandwidths, but there were no significant differences between them, except packet-drop ratio shown in **Figure 6**a. The packet-drop ratio rapidly increases as the bandwidth approaches 100 Mbps, i.e., the specified limit, for the link slivers with per-link policing and with per-slice shaping with policing (which show almost no difference). However, a link sliver with per-slice shaping *without* policing can take bandwidth up to 100 Mbps or more because it can preempt bandwidth of the best-effort slice.

Table 1. NRI performance of 100-Mbps link-sliver (actual traffic bandwidth = 90 Mbps)

Isolation type	Delay (mS)		Jitter (mS)		Drop ratio	
	Average	Std dev	Average	Std dev	Average	Std dev
Per-link policing	1.60	0.12	0.10	0.01	0	0
Per-slice shaping w/o policing	1.30	0.08	0.11	0.02	0	0
Per-slice shaping w policing	1.33	0.10	0.10	0.01	0	0
No isolation	12.08	4.28	0.12	0.01	0.41	0.05
(Congestion-less)*	1.31	0.15	0.12	0.02	0	0

*The bandwidths of both the foreground and background traffic are much lower and the isolation types do not cause almost any differences in this case.



(a) 100-Mbps slow-path link sliver (b) 4.55-Gbps fast-path link sliver

Figure 6. Packet-drop measurement results

We have also measured a foreground slice with higher-bandwidth (4.55 Gbps) link slivers and a fast-path node sliver (using a network processor) with two background slices. The packet size is 1350 B. To keep the traffic acceptable to the node sliver, the total traffic is shaped to 4.0 Gbps by the shaper. The result in Figure 6b shows there is no significant increase in packet drop ($< 4 \times 10^{-5}$) both in per-slice shaping and per-link policing when the bandwidth of the foreground traffic is 4.0 Gbps or less. The result below 3.7 Gbps is omitted from the figure because the shaping bandwidth (4 Gbps) is not filled.

These results show that the per-slice shaping with/without policing is effective for delay-sensitive services while per-link policing may be sufficiently used for the other types of services. Per-slice shaping without policing performs better in terms of packet drop. However, practically, policing is required to avoid starving best-effort slices at least in this implementation. Therefore, per-slice shaping with (per-link or per-slice) policing is suitable for delay-sensitive services.

VII. Related Work

Conventional resource-management methods and QoS-guarantee methods can be applied to solving the resource-isolation problem in virtual networks. Only a few studies focus on resource isolation. In "spawning networks" in Genesis [Cam 99], individual queues are created for each virtual network even when a child virtual network is spawned from a parent virtual network and each child operates in isolation from other virtual networks. However, the implementation detail is not described in their paper. Bilal and Feamster [Bil 10] describes a hardware implementation concerning NRI. However, they used only one queue for all the slices. They used virtualized-network specific methods to implement a VN by using multiple queues. Soltesz et al. [Sol 07] discusses isolation (i.e., fault isolation and resource isolation) using a virtual mechanism.

VIII. Conclusion

Two methods for NRI for virtualization networks are proposed in this paper: per-slice shaping and per-link policing. The per-slice shaping enables NRI with 70–80% less queues compared to the per-link shaping. The per-link policing enables less strict isolation between hundreds of slices using only one queue. These methods have enabled virtualization platforms scalable both in number of slices and in number of programmers or node slivers. The evaluations show that both methods perform NRI well but the performance of the former is better in terms of delay. Accordingly, per-slice shaping with/without policing is effective for delay-sensitive services while per-link policing may be sufficiently used for the other types of services.

Future work includes addressing the following two issues. First, in certain situations, redirector and programmer in a VNode must cooperate to guarantee network-resource isolation. Therefore, a method of this cooperation must be developed. Second, the number of slices with per-slice shaping should be increased from three to hundreds.

Acknowledgments

We thank Atsushi Takahara and Noriyuki Takahashi from NTT for designing and implementing NRI management functions in the Domain Controller and VNode Manager. We thank Takeo Hayashi and Akihiro Motoki from NEC for discussing on the design and for their technical support on the measurements of NRI for VNodes. We also thank Ken'ichi Abiru from Fujitsu Laboratories, Makoto Kitani from ALAXALA Networks, Yasushi Kasugai from Hitachi, and other members of the VNP for their help and comments on the design of NRI. Part of research results described in this paper is an outcome of the Advanced Network Virtualization Platform (Project A) funded by National Institute of Information and Communications Technology.

References

[Bil 10] Anwer, B. and Feamster, N., "Building a Fast, Virtualized Data Plane with Programmable Hardware", *Computer Communication Review*, Vol. 40, pp. 75–82, January 2010. Also in *ACM SIGCOMM VISA'09*, August 2009.

[Cam 99] Campbell, A. T., Vicente, J., and Villela, D. A., "Virtuosity: Performing Virtual Network Resource Management", *7th Int'l Workshop on Quality of Service* (*IWQoS'99*), pp. 65–76, 1999.

[Far 00] Farinacci, D., Li, T., Hanks, S., Meyer, D., and Traina, P., "Generic Routing Encapsulation (GRE)", RFC 2784, IETF, March 2000.

[GEN 09] The GENI Project, "Lifecycle of a GENI Experiment", GENI-SE-SY-TS-UC-LC-01.2, April 2009, http://groups.geni.net/geni/-attachment/wiki/ExperimentLifecycleDocument/ExperimentLifeCycle-v01.2.pdf?format=raw .

[Kan 12] Kanada, Y., Shiraishi, K., and Nakao, A., "Network-resource Isolation for Virtualization Nodes", *COMSNETS 2012*, January 2012.

[Nak 10a] Nakao, A., "Network Virtualization as Foundation for Enabling New Network Architectures and Applications, *IEICE Trans. Commun.*, Vol. E93-B, No. 3, pp. 454–457, March 2010.

[Nak 10b] Nakao, A., "Virtual Node Project — Virtualization Technology for Building New-Generation Networks", *NICT News*, No. 393, pp. 1–6, Jun 2010.

[Par 93] Parekh, A. K. and Gallager, R. G., "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Single-node Case", *IEEE/ACM Transactions on Networking*, vol. 1, June 1993.

[Sol 07] Soltesz, S., Pötzl, H., Fiuczynski, M. E., Bavier, A., and Peterson, P., "Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors", *2nd ACM SIGOPS/EuroSys European Conference on Computer Systems* (*EuroSys'07*), Vol. 41, No. 3, pp. 275–287, June 2007.

[Tur 07] Turner, J., Crowley, P., Dehart, J., Freestone, A., Heller, B., Kuhms, F., Kumar, S., Lockwood, J., Lu, J.,Wilson, M., Wiseman, C., and Zar, D., "Supercharging PlanetLab — High Performance, Multi-Application, Overlay Network Platform", *ACM SIGCOMM Computer Communication Review*, Vol. 37, No. 4, pp. 85–96, October 2007.