VIOLIN: Virtual Internetworking on Overlay Infrastructure

Xuxian Jiang and Dongyan Xu

Purdue University, West Lafayette, IN 47907, USA
{jiangx, dxu}@cs.purdue.edu

Abstract. We propose a novel application-level virtual network architecture called VIOLIN (Virtual Internetworking on OverLay INfrastructure). VIOLINs are isolated virtual networks created on top of an overlay infrastructure (e.g., PlanetLab). Entities in a VIOLIN include virtual end-hosts, routers, and switches implemented by software and hosted by physical overlay hosts. Novel features of VIOLIN include: (1) a VIOLIN is a "virtual world" with its own IP address space. All its computation and communications are strictly confined within the VIOLIN. (2) VIOLIN entities can be created, deleted, or migrated on-demand. (3) Value-added network services not widely deployed in the real Internet can be provided in a VIOLIN. We have designed and implemented a prototype of VIOLIN in PlanetLab.

1 Introduction

Current Internet only provides basic network services such as IP unicast. In recent years, overlay networks have emerged as application-level realization of value-added network services, such as anycast, multicast, reliable multicast, and active networking. While highly practical and effective, overlays have the following problems: (1) Application functions and network services are often closely coupled in an overlay, making the development and management of overlays complicated. (2) The development of overlay network services is mainly individual efforts, leading to few standards and reusable protocols. Meanwhile, advanced network services [1][2][3][4][5] have been developed but not widely deployed. (3) It is hard to *isolate* an overlay from the rest of the Internet, making it easy for a compromised overlay node to attack other Internet hosts.

In this paper, we propose a novel virtual network architecture called VIOLIN (Virtual Internetworking on OverLay INfrastructure), motivated by recent advances in virtual machine technologies [6][7]. The idea is to create virtual isolated network environments on top of an overlay infrastructure. A VIOLIN[1] consists of virtual routers, LANs, and end-hosts, all being software entities hosted by overlay hosts. The key difference between VIOLIN and application-level overlay is

[1] With a slight abuse of terms, VIOLIN stands for either the virtual network technique or one such virtual network.

J. Cao et al. (Eds.): ISPA 2004, LNCS 3358, pp. 937–946, 2004.

that VIOLIN re-introduces *system(OS)-enforced* boundary between applications and network services. As a result, a VIOLIN becomes an "advanced Internet" running value-added network-level protocols for routing, transport, and management.

The novel features of VIOLIN include: (1) Each VIOLIN is a "virtual world" with its own IP address space. All its computation and communications are strictly confined within the VIOLIN. (2) All VIOLIN entities are software-based, leading to high flexibility by allowing on-demand addition/deletion/migration/configuration. (3) Value-added network services not widely deployed in the real Internet can be provided in a VIOLIN. (4) Legacy applications can run in a VIOLIN without modification, while new applications can leverage the advanced network services provided in VIOLIN.

We expect VIOLIN to be a useful complement to application-level overlays. First, VIOLIN can be used to create testbeds for network-level experiments. Such a testbed contains more realistic network entities and topology, and provides researchers with more convenience in experiment setup and configuration. Second, VIOLIN can be used to create a service-oriented (virtual) IP network with advanced network services such as IP multicast and anycast, which will benefit distributed applications such as video conferencing, on-line community, and peer selection.

We have designed and implemented a prototype of VIOLIN in PlanetLab. A number of distributed applications have also been deployed in VIOLIN. The rest of the paper is organized as follows. Section 2 provides an overview of VIOLIN. Section 3 justifies the design of VIOLIN and its benefit to distributed applications. Section 4 describes the implementation and ongoing research problems of VIOLIN. Section 5 presents preliminary performance measurements in PlanetLab. Section 6 compares VIOLIN with related works. Finally, section 7 concludes this paper and outlines our ongoing work.

2 VIOLIN Overview

The concept of VIOLIN is illustrated in Figure 1. The low-level plane is the real IP network; the mid-level plane is an overlay infrastructure such as PlanetLab; and the top-level plane shows one VIOLIN created on the overlay infrastructure. All entities in the VIOLIN are hosted by overlay hosts; and there are three types of entities like in the real network: end-host, LAN, and router.

- A *virtual end-host (vHost)* is a virtual machine running in a physical overlay host. Meanwhile, it is possible that one physical overlay host supports multiple vHosts belonging to different VIOLINs.
- A *virtual LAN (vLAN)* is constructed by creating one *virtual switch* (*vSwitch*, not shown in Figure 1) that connects multiple vHosts.
- A *virtual router (vRouter)* is also a virtual machine with multiple *virtual NICs (vNICs)*. A vRouter interconnects two or more vLANs.

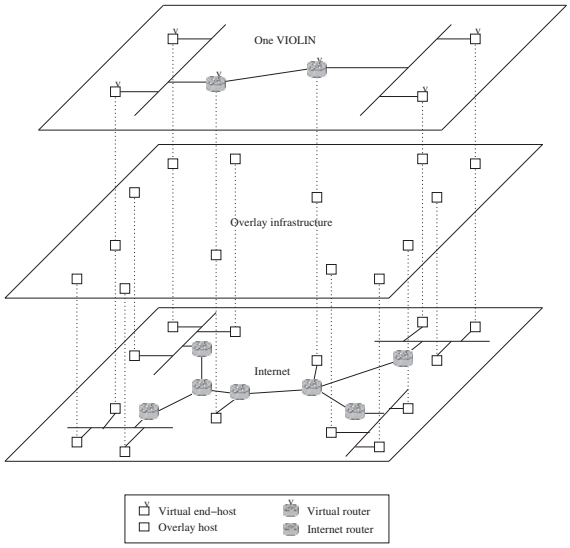

Fig. 1. VIOLIN, overlay infrastructure, and underlying IP network

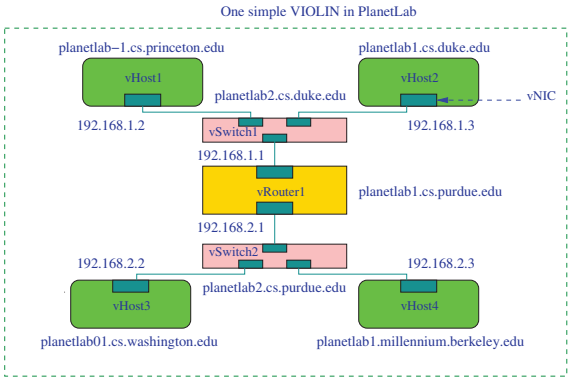

Fig. 2. A VIOLIN in PlanetLab (with names of physical PlanetLab hosts and virtual IP addresses)

Figure 2 shows a simple VIOLIN we create in PlanetLab. Two vLANs are interconnected by one vRouter (vRouter1 hosted by *planetlab1.cs.purdue.edu*):

One vLAN comprises vHost1, vHost2, and vSwitch1; while the other one comprises vHost3, vHost4, and vSwitch2. The links between these entities emulate cables in the real world. The IP address space of the VIOLIN is completely independent. Therefore, it can safely overlap the address space of another VIOLIN or the real Internet.

3 VIOLIN Design Justification

In this section, we make the case for VIOLIN and describe how applications (including network experiments) can benefit from VIOLIN.

3.1 Virtualization and Isolation

Analogous with the relation between virtual machine and its host machine, VIOLIN involves network virtualization and leads to isolation between a VIOLIN and the underlying IP network. Virtualization makes it possible to run unmodified Internet protocols in VIOLINs. Furthermore, entities in a VIOLIN are decoupled from the underlying Internet. For example, if we perform *traceroute* from vHost1 (hosted by planetlab-1.cs.princeton.edu) to vHost3 (hosted by planetlab01.cs.washington.edu) in Figure 2, we will only see vRouter1 as the intermediate router and the hop count is two, although the PlanetLab hosts at Princeton and at UW are many more hops apart in the actual Internet. More interestingly, it is potentially feasible to repeat such virtualization *recursively*: a level-n VIOLIN can be created on a level-$(n-1)$ VIOLIN, with level-0 being the real Internet.

Network isolation is with respect to (1) administration: the owner of a VIOLIN has full administrator privilege - but *only* within this VIOLIN; (2) address space and protocol: the IP address spaces of two VIOLINs can safely overlap and the versions and implementations of their network protocols can be different - for example, one running IPv4 while the other running IPv6; (3) attack and fault impact: any attack or fault in one VIOLIN will not affect the rest of the Internet; (4) resources: *if* the underlying overlay infrastructure provides QoS support [8][9], VIOLIN will be able to achieve resource isolation for local resources (such as CPU and memory [10]) of VIOLIN entities and for network bandwidth between them.

Benefit to Applications. System-level virtualization and isolation provide a confined and dedicated environment for untrusted distributed applications and risky network experiments. From another perspective, applications requiring strong confidentiality can use VIOLIN to prevent both internal information leakage and external attacks.

3.2 System-Enforced Layering

Contrary to application-level overlays, VIOLIN enforces strong layering in order to disentangle application functions and network services. In addition, OS-enforced layering provides better protections to network services after the application level software is compromised. We note that layering itself does *not* incur more performance overhead compared with application-level overlays. We also note that layering is between application and network functions, *not* between network protocols. In fact, VIOLIN can be used as a testbed for the *protocol heap* architecture [11].

Benefit to Applications. Application developers will be able to focus on application functions rather than network services, leading to clean design and easy

implementation. In addition, legacy applications can run in a VIOLIN without modification and re-compilation.

3.3 Network Service Provisioning

VIOLIN provides a new opportunity to deploy and evaluate advanced network services. There exist a large number of well-designed network protocols that are not yet widely deployed. Examples include IP multicast, scalable reliable multicast [2][4], IP anycast [3], and active networking [1][5]. There are also protocols that are still in the initial stage of incremental deployment (e.g., IPv6). VIOLIN is a platform to make these protocols a (virtual) reality.

Benefit to Applications. VIOLIN allows applications to take full advantage of value-added network services. For example, in a VIOLIN capable of IP multicast, applications such as publish-subscribe, layered media broadcast can be more conveniently developed than in the real Internet. We further envision the emergence of *service-oriented* VIOLINs, each with high-performance vRouters and vSwitches deployed at strategic locations (for example, vRouters close to Internet routing centers, vSwitches close to domain gateways), so that clients can connect to the VIOLIN to access its advanced network services.

3.4 Easy Reconfigurability

Based on all-software virtualization techniques, VIOLIN achieves easy reconfigurability. Different from a physical network, vRouters, vSwitches, and vHosts can be added, removed, or migrated dynamically. Also, vNICs can be dynamically added to or removed from vHosts or vRouters; and the number of ports supported by a vSwitch is no longer a hardware constraint.

Benefit to Applications. The easy reconfigurability and hot vNIC plug-and-play capability of VIOLIN is especially useful to handle the dynamic load and/or membership of distributed applications. Not only can a VIOLIN be created/torn down on-demand for an application, its scale and topology can also be adjusted in a demand-driven fashion. For example, during a multicast session, a new vLAN can be dynamically grafted on a vRouter to accommodate more participants.

4 VIOLIN Implementation

4.1 Virtual Machine

All VIOLIN entities are implemented as virtual machines (VMs) in overlay hosts. We adopt User-Mode Linux (UML) [12] as the VM technology. UML allows most Linux-based applications to run on top of it without any modification. Based on *ptrace* mechanism, UML - the *guest OS* for a virtual machine, performs system call redirection and signal handling to emulate a real OS. More specifically, the guest OS will be notified when an application running in the virtual machine issues a system call, the guest OS will then redirect the system call into its own implementation and nullify the original call. One important feature of UML is

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.