CoreLab: A Wide-Area Network Testbed for Emerging Network Services and Architectures

Akihiro Nakao[1], Ryota Ozaki[2], Yuji Nishida[2], Ping Du[2], Maoke Chen[2], Kiyohide Nakauchi[2], and Richard Potter[2]

[1]University of Tokyo
[2]National Institute of Information and Communications Technology (NICT)

Abstract

Wide-area network testbeds have become viable means for experimenting with new network services. We have recently developed and deployed a new wide-area network testbed called CoreLab that aims at supporting flexible computational execution environments compared to the existing testbeds as well as enabling virtual network connections among computational resources. CoreLab currently supports execution environments through multiple virtualization techniques in order to fulfill scalability, performance, isolation and flexibility requirements for various experiments. CoreLab also implements network namespace isolation and network virtualization to satisfy the necessity for avoiding cross-talks among emerging network services. This paper briefly summarizes the highlights of our recent development in the CoreLab project.

1 Introduction

Wide-area network testbeds have been regarded as crucial infrastructures for developing, deploying and experimenting with new, possibly disruptive network services and architectures for the future Internet. There have been a proliferating number of test-beds developed, such as PlanetLab [10], VINI [9], OneLab [5], Emulab [15], ORBIT [8, 14] and CoreLab [1] to provide *isolated*—without interference among one another—virtual execution environments (called *slivers*) for various experiments.

One of the most important considerations to make in designing such a network testbed is to carefully select the type of slivers to meet the requirements for various kinds of experiments and developers. We have identified the design principles for building a wide-area network testbed [11], such as *performance* of slivers, *scalability* in terms of the number of slivers, the level of *isolation* among slivers and *flexibility* of slivers in supporting a wide spectrum of network services. As a result, we have designed and developed CoreLab especially focusing on flexibility that has been largely ignored in the existing testbeds [11].

In our recent development of CoreLab, we have extended our design and implementation of the testbed in terms of (1) supporting more flexibility in slivers by enabling the unified architecture to support various kinds of virtualization techniques, (2) strengthening resource isolation both for computational and network resources, and (3) enabling federation with the other testbeds. We have also put forth implementing and executing network services on top of CoreLab to gain more experiences with our testbeds to improve them. This paper briefly summarizes the highlights of our recent enhancements of CoreLab.

2 Enhancement

2.1 Extended Flexibility

CoreLab has recently enabled the unified architecture to accommodate various kinds of virtualization techniques simultaneously, such as hosted virtualization approach, e.g., KVM [3] and resource containers, e.g., LXC [4], OpenVZ [7] etc. Since each virtualization technique has its pros and cons [11], it is important to be able to provide the execution environment through the virtualization technique that best satisfies the requirements for experiments.

2.2 Resource Isolation

CoreLab isolates resources, e.g., CPU, network bandwidth and memory among slivers in a fair manner. Resource consumption of the slivers is regularly monitored in order to prevent abuse and other malicious behaviors.

CPU and network bandwidth are fairly shared by time division through Linux facilities, such as Completely Fair Scheduler (CFS), token bucket filter and control groups (cgroups). Slivers and their management tasks are classified into groups so their resources are isolated from one another. Memory isolation is realized through exclusive space division and over-booking. The former does not allow memory sharing among slivers and achieves strong performance isolation, however, it provides less scalability in terms of the number of slivers. The latter allows over-commit of memory in each sliver to leverage unused memory by other slivers but therefore may cause performance degradation.

2.3 Network Namespace Isolation

Network namespace refers to a set of named entities in network stacks including network interfaces, IP addresses, ports, forwarding and routing tables, as well as the sockets. Network namespace isolation involves all of these while the IP addresses and ports are the most essential entities as they identify slivers.

CoreLab implements two methods for partitioning IP addresses and ports among slivers by (1) allocating private IP addresses to slivers using NAT and (2) sharing global IP addresses and allocating separate port ranges, given that each CoreLab node has only a limited number of IP addresses to share among slivers[1]. In the former method, each sliver is assigned to a private IP address and a specific range of port numbers, while a CoreLab node performs NAT and port-forwarding for the port range. The implementation includes a series of iptables destination NAT (DNAT) rules. The NAT solution has two short-comings: (1) it breaks the end-to-end semantics so that some applications become unavailable, and (2) the NAT module degrades communication performance.

In the latter method, we remove NAT and let all the slivers share the public IP address(es) of a CoreLab node, but isolate their network activities through static separation of port ranges assigned to slivers. CoreLab implements the port-range separation through either a Layer-2 translator, e.g., ebtables [2], or a full-fledged programmable bridge, e.g., Open vSwitch [6]. Port-space isolation with Open vSwitch can achieve 600Mbps between two slivers over two nodes connected with Gigabit Ethernet, while the same performance for NAT solution is only 250 Mbps.

2.4 Federation

Recently, the Future Internet initiatives such as FIRE (EC FP7) and GENI (NSF) and its related research projects rely on *slice-based* experimental facilities where a *slice* represents a collection of slivers. In order to mutually extend the footprint of these testbeds, federation among slice-based facilities such as CoreLab, PlanetLab, Emulab, and VINI, is strongly required.

CoreLab is designed to support federation based on Slice-based Facility Architecture (SFA) [12] that defines the abstraction of a set of computational and network resources, global identifiers (GID) for objects, resource specifications (RSpec), and high-level interfaces of objects. SFA thus enables sharing resources even if facilities are managed by different control entities.

3 Applications

We have demonstrated various types of network service applications on CoreLab, such as (1) building IPv6 routing overlays on Layer-2 network virtualization, (2) enabling advanced routing enabled through *Open vSwitch in a slice*, both of which are difficult to implement at scale in the existing testbeds such as PlanetLab and VINI, and (3) implementing an infrastructure called Mobitopolo [13] for developing, debugging, and deploying distributed applications. In the last example, we demonstrate yet another testbed built on top of the CoreLab testbed. Mobitopolo provides user-mode Linux virtual machines that are connected with point-to-point Layer-2 tunnels and can be saved to snapshots. Distributed applications on Mobitopolo can be migrated live to different locations while all the (virtual) network connections among them can survive through the live WAN migration. Interesting network services await us to be developed, deployed and experimented with in CoreLab.

[1]Usually each CoreLab node has only one or two addresses (interfaces) in the current deployment.

4 Conclusion

This paper briefly summarizes the highlights of our recent enhancement of CoreLab. In our recent development of CoreLab, we have extended its design and implementation in terms of flexibility, resource isolation and network namespace isolation, and federation. We are striving to enhance CoreLab further in near future and also planning to enable innovative network services on top of it for future Internet.

References

[1] CoreLab Testbed. http://www.corelab.jp.

[2] Ebtables.

[3] Kvm. http://kvm.qumranet.com/kvmwiki.

[4] LXC. http://lxc.sourceforge.net/.

[5] OneLab. http://www.onelab.eu/.

[6] OpenVSwitch. http://openvswitch.org/.

[7] OpenVZ. http://wiki.openvz.org/Main_Page.

[8] ORBIT. http://www.orbit-lab.org/.

[9] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In vini veritas: realistic and controlled network experimentation. *SIGCOMM Comput. Commun. Rev.*, 36(4):3–14, 2006.

[10] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, 2003.

[11] A. Nakao, R. Ozaki, and Y. Nishida. Corelab: An emerging network testbed employing hosted virtual machine monitor. In *3rd International Workshop on Real Overlays and Distributed Systems (ACM ROADS Workshop 2008)*, Madrid, Spain, December 2008.

[12] L. Peterson, S. Sevinc, J. Lepreau, R. Ricci, J. Wroclawski, T. Faber, S. Schwab, and S. Baker. Slice-Based Facility Architecture. Technical Report http://svn.planet-lab.org/attachment/wiki/GeniWrapper/sfa.pdf, February 2009.

[13] R. Potter and A. Nakao. Mobitopolo: a portable infrastructure to facilitate flexible deployment and migration of distributed applications with virtual topologies. In *VISA '09: Proceedings of the 1st ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*, pages 19–28, 2009.

[14] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols. *Wireless Communications and Networking Conference, 2005 IEEE*, 3:1664–1669 Vol. 3, 2005.

[15] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):255–270, 2002.