Integration of Multiple IP Domains in Low-cost and Security-oriented Small Networks

Satoshi Kodama
Research Center,
Research Institute for Science & Technology
Tokyo University of Science
Yamazaki 2641, Noda-shi, Chiba, 278-8510 Japan
skodama@rs.tus.ac.jp

Rei Nakagawa
Department of Information Sciences
Faculty of Science and Technology
Tokyo University of Science
Yamazaki 2641, Noda-shi, Chiba, 278-8510 Japan
j6316627@gmail.com

Abstract**—** This paper has two main aims. The first is to reduce hardware management-related issues when working with complex networks, such as a lack of flexibility and scalability with regard to global configuration changes. The second is to create a communication library that provides programmable network function virtualization, helping to achieve the first aim. Specifically, we target low-cost, security-oriented networks that use broadband routers as L3 switches. When such networks have multiple IP domains between the external gateway and internal networks, domains that are lower in the hierarchy become more secure, but at the cost of flexibility. Our proposed system architecture aims to increase communication flexibility in such networks without requiring complex encapsulation and achieves this goal via a practical method that does not change the packet size. Finally, we conduct tests using an experimental implementation of the system, demonstrating its scalability.

Keywords—Software Defined Network; Private Network; Network Management System

I. Introduction

Networks are increasingly being used in new and more varied ways, and applications are being created that do not necessarily conform to the OSI Reference Model. For example, as Fig. 1 shows, people use private networks consisting of separate IP domains managed by multiple broadband routers. This is a tree-structured network, where the top router acts as the default Internet gateway and each router (a leaf in the tree structure), creates its own independent IP domain on the LAN side. The independence of these domains provides robust security at low cost, but makes it difficult to communicate with other such networks via the gateway. This is because the default gateway is always in the top level; therefore, it cannot communicate with nodes lower down in neighboring networks easily.

There are two ways to resolve this issue. The first is to introduce a switch device that can perform virtual LAN (VLAN) functions; however, for small and inexpensive private networks, this method is too costly, and there are also concerns about the processing limitations of the hardware used as VLAN equipment [1]. The second approach is to add a proxy router to each network domain. We focus on this method, proposing a flexible and scalable communication system that does not require complicated mechanisms such as generic routing encapsulation [2].

The remainder of this paper is organized as follows. In Section II, we introduce the proposed system architecture, which uses a so-called conversion router, and present the components needed to implement the system in Sections III and IV. In Section V, we evaluate an experimental test system by creating a simple TCP connection and downloading files over HTTP from a server to a client. Finally, Section VI concludes the paper.

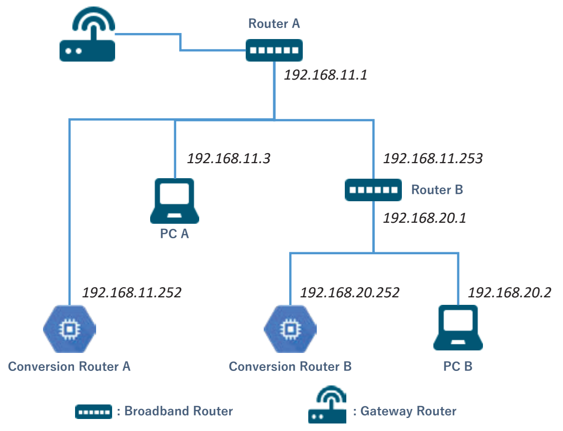


Fig. 1 Low-cost and security-oriented network

II. System Architecture

In this section, we present the proposed system architecture, which adds a conversion router to each target network, as Fig. 2 shows.

A. Usage scenarios

In the network shown in Fig. 2, two routers (Routers A and B) have set up separate IP domains in a private network. Router A is the default gateway for Router B. With such a network topology, PC A could not normally communicate with PC B, because PC A's communication packets would flow to the gateway (Router A), which has no knowledge of the IP addresses used by Router B.

To solve this problem, we add a conversion router to each network domain. This acts as a sub-router, complementing the flexibility and scalability of the gateway router. Each conversion router forwards the communication packets to a corresponding conversion router in the destination domain, which then forwards them to the original destination node. In this way, our proposed system resolves the problem and enables bidirectional communication between the domains.

B. Packet processing

The conversion routers process the packets as follows. The source router rewrites the destination address to that of the destination (or relay) conversion router. In addition, to enable the original destination address to be recovered, it embeds the lower two bytes of the destination address (i.e., the x.x part of an IP address y.y.x.x) in the urgent pointer entry in the TCP header (which is two bytes in size). Then, based on this embedded information, the destination conversion router recreates the original IP address and carries out any further processing needed to complete the transfer. These processing steps are executed by a communication library, described in Section III.

C. Advantages and issues

The main advantage of the proposed system is that we can delegate the router configuration that would otherwise be required to the conversion routers. This makes the system software more flexible and scalable, because complex processes, such as global changes to the conversion routers' configuration, are carried out independently of the other routers. In addition, the system does not require complex encapsulation, which is expected to improve performance and the usefulness of networks with isolated IP domains.

That said, however, the packet location where the routing information is embedded is potentially problematic. The use of the urgent pointer area assumes we are communicating over TCP using IPv4 and imposes a two-byte limit. Therefore, we need to think carefully about the system's future scalability, because this information embedding approach assumes the IP addresses lie within small networks, all of which use the same address range. One of the leading alternative embedding candidates is the MAC address, which is attracting attention as a location for embedding new information since it can be rewritten relatively freely [7].

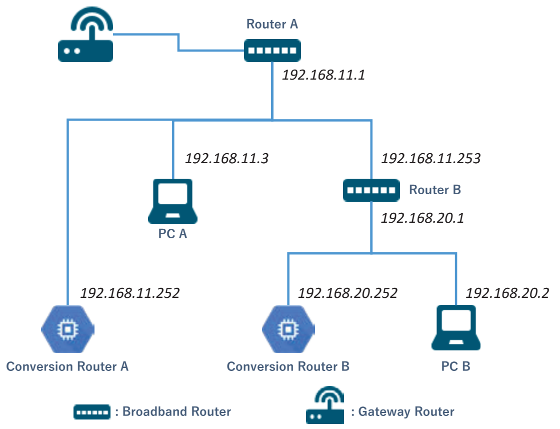


Fig. 2 Proposed system architecture

III. COMMUNICATION WORKSHOP

We have also designed a communication library for the conversion routers to use to process the communication packets, as Fig. 3 shows.

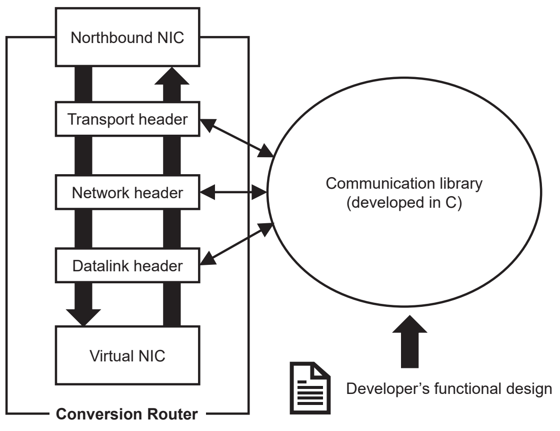


Fig. 3 Conversion router's communication library

Before developing this library, we studied communication library implementations that enable the communication headers (such as Ethernet, IP, or TCP headers) to be rewritten [3, 4]. In terms of its packet processing functions, our library is similar to OpenFlow's data plane [5]. In addition, it provides simple and powerful network function virtualization [6] that can be arbitrarily defined by developers using the C language. In our system, the communication library processes packets bound for a virtual NIC (vNIC). This is assigned the conversion router's IP address, and only the packets associated with that address are streamed to user space by the Linux kernel. Then, the library branches on the packets' protocol headers, rewriting them as discussed in Section II.

IV. Scalability

The proposed system's scalability depends on the design of the distributed management system. As the network size grows, the overhead due to the conversion routers needing to identify each other increases. In the target network in Fig. 1, at higher levels in the network's tree structure, more terminals are accessible without needing to go through the conversion router. Taking advantage of this property, we install an authentication and management server for the conversion routers in the top level of the structure. This server manages the information required to integrate the conversion routers and periodically delivers it to each router in the form of a routing table, enabling the routers to be centrally managed and providing the scalability needed to handle increased network sizes.

V. Experimental Evaluation

In this section, we demonstrate the proposed architecture using a simple experiment.

A. *Aim*

In this experiment, we created a TCP connection and carried out an HTTP download using the "wget" command, using the network shown in Fig. 2 with PC A as the client and PC B as the server. Then, we verified the connection and performance of the PC in the lower domain, which would not otherwise be able to communicate with the rest of the network. Note that no management server was used for this test.

B. *Configuration*

Table. 1 lists the equipment used for the experiment. For the sake of simplicity, the client (PC A) and Conversion Router A made the following assumptions. Conversion Router A assumed that PC A planned to connect to PC B, while PC A began the experiment by trying to connect to Conversion Router A. This then started to relay the connection from PC A to PC B and carried out the previously-described processing steps. In addition, to enable interconnection between Routers A and B and Conversion Routers A and B, all traffic to the WAN-side IP addresses of Routers A and B was redirected to the addresses of Conversion Routers A and B Set it.

Table. 1 Specification of equipment

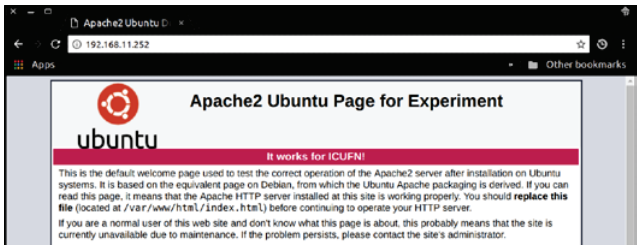
Network Device	Specification
Router A, B	Buffalo WXR-1750DHP/Y
Conversion Router A, B	Raspberry Pi 3 running CentOS 7 CPU: 1.2GHz 64-bit quad-core Memory: 1Gbyte
PC A, B	Intel NUC running Ubuntu 16.04 CPU: Intel Core i7-7567U Memory: 16Gbytes

C. *Scenario*

First, PC A began to communicate with Conversion Router A using the "wget" command. Upon receiving the connection request from PC A, Conversion Router A embedded the lower two bytes of PC B's IP address in the urgent pointer region of the TCP header, set the source and destination IP addresses to those of Conversion Router A and Router B, respectively, and relayed the communication. Upon confirming the connection request from Conversion Router A, Conversion Router B read the lower two IP address bytes stored in the urgent pointer and used them to overwrite the lower two bytes of the actual destination IP address. Then, after setting the source address to that of Conversion Router B, it relayed the request to PC B. The response from PC B was then relayed back to PC A by following the same process in reverse, with PC B's IP address embedded in the urgent pointer by the conversion router, and then retained as the original source address.

D. *Result*

Fig. 4 shows that PC A was able to obtain an HTML document from PC B, while Table. 2 and Fig. 5 show averages of both the download time and the communication throughput when obtaining files of different sizes (1 MB, 10 MB, 50 MB, 100 MB, 500 MB, and 1 GB) 10 times from PC B over HTTP.



Apache2 Ubuntu Page for Experiment

It works for ICUFN!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Fig. 4 HTML document acquired by PC A

Table. 2 Download Time

File Size (bytes)	Download Time (s)
1M	0.1
10M	1.3
50M	6.0
100M	12.0
500M	57.0
1G	116.0

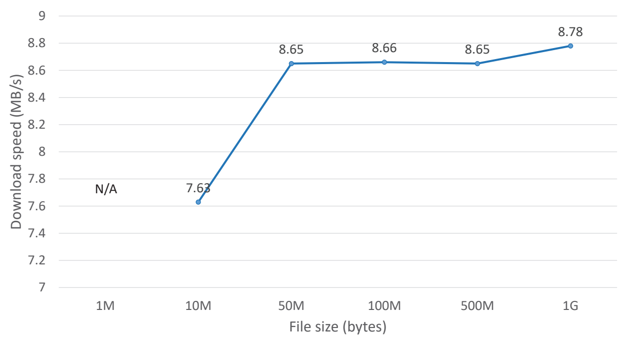


Fig. 5 Communication throughput

The IP address in Fig. 4 shows the TCP connection request from PC A to Conversion Router A that served to trigger the experiment. In Fig. 5, the download speed for the 1 MB file is given as "N/A" because the transfer happened too quickly to measure accurately, and the throughput is lower than expected for the 1 MB file for the same reason. For the other files, we measured an average throughput of about 8.7 MB/s, which indicates the real system performance in this experiment.

VI. Conclusion

In this paper, we have proposed a system for connecting multiple IP domains in small networks based around standard routers in a low-cost, security-oriented manner. Our system involves adding a conversion router to each domain, running flexible and scalable software that complements the other routers. The conversion routers operate using a communication library that enables programmable network function virtualization based on the developer's functional design and only requires simple packet rewriting (which does not change the packet size) based on the urgent pointer.

Our system has the advantage that the system software flexible and scalable because it carries out the complex process of requesting global configuration changes to the conversion routers independently of the other routers.

In future work, we plan to tackle issues related to scalability as the network size grows. First, there is the issue of how to maintain relationships between multiple clients on the same LAN that connect to the same server, and handle the corresponding traffic. Second, we will consider the increasing delays that are expected as the network structures get deeper and more complex. Finally, we will look for ways to optimize the structure of the mechanism for relaying connection requests on the client side.

References

[1] Satoshi Kodama, Rei Nakagawa, Toshimitsu Tanouchi, Shinya Kameyama, "Management system by using embedded packet for hierarchical local area network," IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2016.

[2] D. Farinacci, T. Li, Procket Networks, S. Hnaks, Enron Communications, D. Meyer, Cisco Systems, P. Traina, Juniper Networks, "Generic Routing Encapsulation (GRE), " RFC 2784, 2000.

[3] Satoshi Kodama, Rei Nakagawa and Toshimitsu Tanouchi, "Proposal of the Virtualized Control system for the integrated management of multiple services," IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), 2017.

[4] Satoshi Kodama, Rei Nakagawa and Toshimitsu Tanouchi, "A Research on the Integrated Virtual Platform for Managing Multiple Services," WSEAS TRANSACTIONS ON INFORMATION SCIENCE AND APPLICATIONS, vol. 14, Art. #12, pp. 102-111, 2017.

[5] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker and Jonathan Turner "OpenFlow: enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, vol. 38, pp. 69-74, 2008.

[6] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck and Raouf Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," IEEE Communications Surveys & Tutorials, vol.18(1), pp. 236-262, 2018.

[7] Peter Kietzmann, Cenk Gündoğan, Thomas C. Schmidt, Oliver Hahm and Matthias Wählisch "The need for a name to MAC address mapping in NDN: towards quantifying the resource gain", 4th ACM Conference on Information-Centric Networking, 2017.