Service-oriented Router Module Implementation on ns-3

(Poster Abstract)

Janaka Wijekoon
Nishi Laboratory, Keio University,
janaka@west.sd.keio.ac.jp

Rajitha Tennekoon
Nishi Laboratory, Keio University,
rajitha@west.sd.keio.ac.jp

Erwin Harahap
Nishi Laboratory, Keio University,
erwin2h@west.sd.keio.ac.jp

Hiroaki Nishi
Department of System Design
Keio University
west@sd.keio.ac.jp

ABSTRACT

A Service-oriented Router (SoR) is a router proposed to provide content-based services for next-generation networks by shifting the current Internet infrastructure to an information-based, open-innovation platform. Because the SoR is still under research and development, a simulation platform is required to simulate its functions. The ns-2-SoR implementation has several limitations including the fact that it cannot be used with generic traffic generators, transport protocols, and NetDevices. However, ns-3 infrastructural advancements provide an optimal environment to build the basics of an SoR, including Deep Packet Inspection, content-based services, and content-based packet redirection. In this paper, we describe the basic infrastructural implementation of an SoR on ns-3 and discuss its design. Our experiments show that the ns-3-SoR does not incur significant processing load, memory usage, or packet delivery latency on ns-3.

Categories and Subject Descriptors

C.2.6 [**Internetworking**]: Routers

General Terms

Design

Keywords

SoR, Service-oriented Router, ns-3, Network Simulation

1. INTRODUCTION

The router is a key device for interconnecting networks. A router can access all the information included in a packet stream, including packet header information, data link, network and transport layer information, and application information. Content or data can also be captured passively by a router. Our laboratory is studying Service-oriented Routers (SoR)[2, 3, 5, 6], to utilize the aforementioned advantages. An SoR is proposed to provide services to end users from the router itself, using special router APIs provided by our research team. Studies conducted in [5] are example applications implemented based on features of SoR to provide content based services.

There are several motivations behind this implementation of an SoR using an ns-3 simulator. As the SoR is still under research and development, a simulator is necessary to simulate the basic functions including Deep Packet Inspection (DPI), content-based packet redirection, and packet stream analysis. Therefore, we implemented the SoR in ns-2 [6] and the implementation was successfully used to simulate the SoR as a fine redirector for Content Delivery Networks [5]. The ns-2 SoR had several limitations due to the limitations of the ns-2 infrastructure. Because an ns-2 node has only one entry point to the node [1], the SoR module could not maintain proper tracking of its network interfaces for the data analysis and redirection. This severely limited the ns-2-SoR to support real-world applications after the DPI had been performed on the packets. Furthermore, as ns-2 supports only the ns-2 raw packet structure, it does not support payload handling. As a result, we had to propose a new packet structure to carry the user data in the simulation to simulate content-based data manipulation. Consequently, the SoR could not function with real-world packet formats.

The design of ns-3, however, is well suited for the SoR implementation. The ns-3 node was designed as a real world router and has network interfaces, for example, ns3::NetDevices. Moreover, the packet structure of ns-3 can also carry application data along with serialize and de-serialize functions that can be executed during packet propagation in the protocol stack [4]. These newly added features support our motivation to implement the SoR in a simulated real-world environment.

The rest of this paper is arranged as follows. We briefly explain the SoR and discuss the challenges of its integration to ns-3 in Section 2. Section 3 describes the architecture of the SoR. The experiments and test results are presented in Section 4. Section 5 concludes the paper and describes future work.

2. INTEGRATION OF SOR TO NS-3 NODE

Simutools 2014, March 17-19, Lisbon, Portugal

DOI 10.4108/icst.simutools.2014.254690

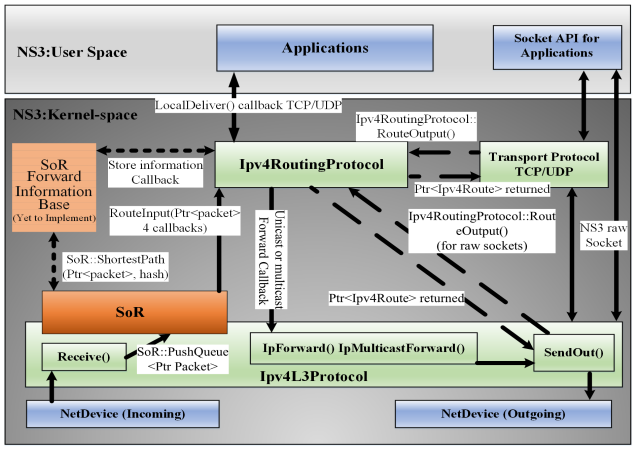

Figure 1: Integration of SoR to the ns3::Node

In this section, we briefly discuss the SoR and ns-3 individually. Then, we discuss the integration of these two modules.

2.1 Network Simulator 3 (ns-3)

The network simulator 3 (ns-3) [4] is a discrete-event simulator that is proposed to eventually replace the ns-2 simulator. Nodes in the ns-3 simulator are connected to each other by means of NetDevices communicating over their respective channels. A node can have multiple NetDevices as its interfaces. Nodes can operate with or without an IP stack such as a Layer 2 switch or an Ethernet bridge. Furthermore, the packets in ns-3 can be serialized or desterilized to/from actual packet formats as they traverse through the network stack. This makes ns-3 well suited for real-world integration. Moreover, ns-3 has IPv4/IPv6 addressing schemes available for network specifications, something which ns-2 lacked and thus performed implicitly.

2.2 Enable the ns3-SoR

The implemented ns-3-SoR supports two basic features of the SoR [3, 2]: 1) capture packet at its interfaces; 2) perform DPI to analyze the packet information including application layer data. Therefore, we placed the SoR module as shown in Figure 1. This allows the proposed SoR to capture raw packets before they go through the protocol stack. Hence, it allows the SoR to have total control of the packet streams. *<node name> -> EnableSoR(1);* command was used to enable the ns3::Node for the SoR. Once the packet arrives at the ns3::Node from the receiving ns3::NetDevice, *Node::ReceiveFromDevice()* function inserts the packet into the SoR::packet queue. Then, the SoR module retrieves the packet from the queue and analyzes the packet for the five basic properties of a network packet that are used to distinguish the packet streams in the network. These are sender IP, sender port, destination IP, destination port, and the protocol of the packet. Therefore, the implemented SoR is able to capture the packet and identify the five-tuples shown in Figure 2.

2.3 Technical Challenges of the Integration

The primary challenge was to implement the SoR so that it could capture and analyze the raw network packets before being altered by the protocol stack. Therefore, we implemented the packet entry point of the SoR module as the same packet entry point of the ns3::Node. As a result, all the packets first go through the SoR::Analyzer module. This method presented the challenge of using the IP and Transport layer headers in the Datalink and Physical layers. This was because the ns3::networkmodule does not have access to the ns3::Internetmodule headers [4]. Therefore, we made several changes to the Waf compiler to get access to the ns3::Internet module including the ns3::Ipv4header in the ns3::network module in order to perform the DPI in the SoR.

```
1: At time 2s client sent 20 bytes to 5.6.7.8 port 9
This is from the Node:1 UDP packet 10.0.1.1:49153 --> 5.6.7.8:9
This is from the Node:2 UDP packet 10.0.1.1:49153 --> 192.168.16.1:9
This is from the Node:3 UDP packet 10.0.1.1:49153 --> 192.168.16.1:9
At time 2.0063s server received 20 bytes from 10.0.1.1 port 49153 and
At time 2.0063s server sent 32 bytes to 10.0.1.1 port 49153
This is from the Node:2 UDP packet 10.0.3.1:9 --> 10.0.1.1:49153
This is from the Node:1 UDP packet 10.0.3.1:9 --> 10.0.1.1:49153
This is from the Node:0 UDP packet 10.0.3.1:9 --> 10.0.1.1:49153
At time 2.0126s client received 20 bytes from 10.0.3.1 port 9 and the
```

Figure 2: Five Tuple recognition of ns3-SoR

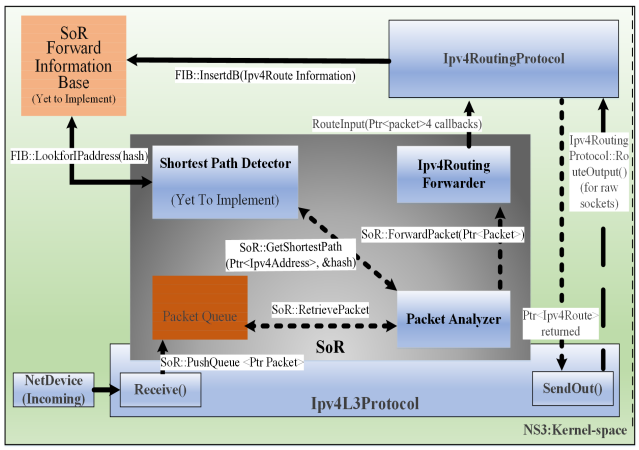

Figure 3: ns3-SoR Architecture

3. ARCHITECTURE OF THE NS3-SOR

As shown in the Figure 3, the ns-3-SoR consists of a packet buffer, packet analyzer, shortest path detector (not discussed in this paper), SoR forwarding information base (not discussed in this paper), and Ipv4RoutingForwarder. Once a packet arrives at the ns3::Receiver, the receiver function calls the SoR::PacketQueue to store the packet in the SoR buffer. Then, the SoR::PacketAnalyzer module takes the packet and performs the DPI for the five-tuples. Finally, it displays the packet content in the standard display. Furthermore, as shown in the Figure 2, the SoR packet analyzer module has the capability of altering the packet header information if necessary. Once a packet has been analyzed, if the packet needs a change of header, the SoR analyzer module does it according to the hard coded rules in the SoR analyzer. Then, the packet is passed to the ns3::Ipv4Routing Protocol for further routing. The ns3:: Ipv4Routing Protocol handles the packet according to the basic ns-3 packet forwarding processes. Since the SoR routing protocol and neighbor management are under development, this paper does not provide information about the routing.

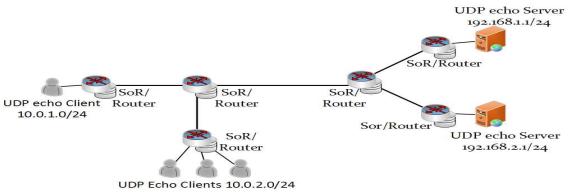

Figure 4: Simulation Topology

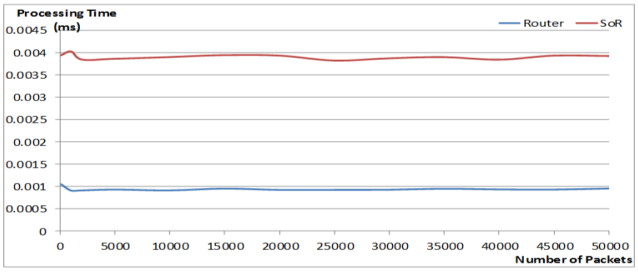

Figure 5: Average Processing Time of both SoR and NS3::Node

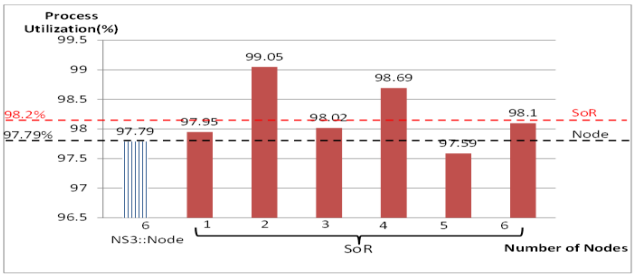

Figure 6: Processor Consumption for 50,000 packets

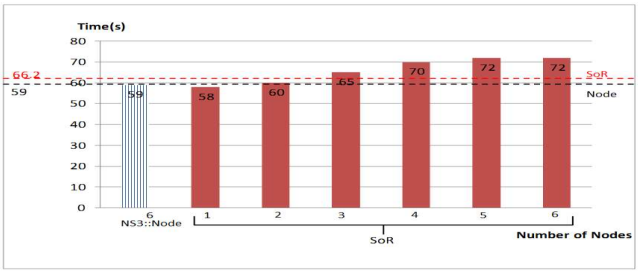

Figure 7: Total Simulation Run time for 50,000 packets

4. SIMULATION AND TEST RESULTS

The simulation topology shown in the Figure 4 was implemented on a computer with an Intel i7 3.4 GHz processor with 8 GB of memory. All links were configured as 5 Mbps P2P with 2 ms link delay. The six NS3::nodes were capable of functioning as both SoR and regular NS3::Nodes and the data traffic was handled using a UDP echo client and servers. First, we initiated every UDP echo client to send 5,000 to 50,000 packets to the UDP echo server and measured the average processing delay of both the SoR and ns3::Node. As shown in the Figure 5, on the average, the SoR consumed 4μs for the processing while the ns3::Node consumed only 1μs. The SoR processing time was 3μs higher compared to the NS3::Node because the SoR performed DPI to the packets to determine the 5-tuples. Next, we set all the clients to send 50,000 packets to the server and we measured the average processor utilization for six ns3::Nodes and compared that with the average processor utilization for six SoRs. As shown in Figure 6, the six ns3::Nodes utilized 97.79% of the processor while the six SoRs utilized 98.2% of the processor - only 0.41% higher than the ns3::Nodes. This is a very small increase considering the SoR implements packet analysis.

According to the Figure 7, the total simulation time, when the nodes were configured as SoRs, was on average 7s higher compared with the scenario where the nodes were configured as ns3::Nodes.

5. CONCLUSION

The implementation of the basic SoR functions on the ns-3 simulator successfully executed with a minimum overhead to both computer and ns-3 simulator. Experimental test results suggest that adding the SoR to ns-3 simulator can be done without significant processing time and process consumption. The improved flexibility comes at a cost of increased total execution time. For the future development of this study, we will focus on implementing a routing protocol to handle ns3::Ipv4L3routing in the SoR network topology and a content-based shortest-path detection algorithm.

6. ACKNOWLEDGMENTS

This work was partially supported by funds for integrated promotion of social system reform and research and development, MEXT, Japan, by Low Carbon Technology Research and Development Program for "Practical Study on Energy Management to Reduce CO2 emissions from University Campuses" from Ministry of the Environment, Japan and by MEXT/JSPS KAKENHI Grant (B) Number 25280033.

7. REFERENCES

[1] Kevin Fall and Kannan Varadhan. The ns manual, http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf. Accessed 11/2013.

[2] K. Inoue, D. Akashi, M. Koibuchi, H. Kawashima, and H. Nish. Semantic router using data stream to enrich services. In *International Conference on Future Internet Technologies (CFI08)*, June 2008.

[3] H. Nishi. Service-oriented backbone router for future internet, http://openinter.net/wp-content/uploads/2012/03/p2.pdf, Accessed 02/2014.

[4] ns 3 Manual. http://www.nsnam.org/docs/manual/html/index.html. Accessed 11/2013.

[5] J. Wijekoon, E. Harahap, and H. Nishi. Sor based request routing for future cdn. In *6th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1 – 5, October 2012.

[6] J Wijekoon, E. Harahap, and H. Nishi. Service-oriented router simulation module implementation in ns2 simulator. *The 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013)*, 19:478 – 485, 2013.