Steroid OpenFlow Service: A Glimpse of Evolving Customized Network Services in a Software Defined Future Internet

Aaron Rosen, Kuang-Ching Wang

Holcombe Department of Electrical and Computer Engineering  
Clemson University  
Clemson, SC, US

{arosen, kwang}@clemson.edu

**ABSTRACT**

In a software defined network (SDN), packet forwarding methods can be changed by software controllers on the fly to suit the needs of different traffic types. In addition to redirecting traffic's paths, such a network can apply customized “network services” to optimize distinct application performance objectives. This can be realized by inserting software agents in the forwarding paths between end-to-end applications to apply traffic optimization actions. Such “network service agents” can be deployed *explicitly* by request of the applications or *implicitly* as a transparent service by the network provider. To illustrate the concept, we developed a network service named Steroid OpenFlow Service (SOS) to optimize TCP based applications’ throughput performance. In an OpenFlow based SDN, the network detects the onset of long-range TCP flows, launches a pair of SOS agents (one near each end host), and redirects TCP traffic to the nearby SOS agents. The SOS agents then launches between themselves a parallel set of SCTP sockets to transport the received TCP traffic across the wide area network. Initial tests showed more than 10 times throughput enhancement as compared to a pure end-to-end TCP connection. The SOS paradigm is not limited to a specific set of protocols (e.g., TCP/SCTP) or a specific purpose (e.g., throughput enhancement); instead, it demonstrates the decoupling of end applications and the network’s choice of transport protocols. The paradigm reveals significant implications to future Internet architecture for cloud computing, mobile computing, and media/content delivery applications.

**Keywords**

Software Defined Networking, OpenFlow, Cloud Computing, Mobile Computing

# INTRODUCTION

Among the recent surge of efforts exploring solutions for speeding up innovations in Internet protocols, Software Defined Networking (SDN) emerges as one attractive approach to allow flexible implementation of novel protocols in the Internet switching infrastructure via software controllers. OpenFlow, a pioneering SDN solution, defines a standardized messaging interface for a software controller to communicate with and control the traffic forwarding rules of any number of Ethernet switches [1]. With OpenFlow, network providers can monitor and control the data plane of their packet switched network infrastructure in customizable layer fidelities via one or few centralized software controllers, referred to as OpenFlow controllers. In view of the fertile possibilities of OpenFlow and SDN, major industrial proponents including Google, Microsoft, Yahoo, Facebook, Verizon, Deutsch Telecom and others have recently formed the Open Networking Foundation to drive the standardization and realization of such solutions [2].

The advent of SDN breathes a paradigm shift in the service delivery architecture in Internet. In the past, the network is an application-agnostic platform on which services are deployed as end-to-end software in the application/service layer of the protocol stack. With SDN, the network gains visibility and agility to react to different traffic types differently on the fly via its software controller(s). In the National Science Foundation’s (NSF) Global Environment for Network Innovations (GENI) project [3], several researchers have been exploring novel network services that can be deployed in an OpenFlow network. In a nutshell, they envision a similar future network architecture where each switching device has either on-board or attached computing hosts and storage resources dedicated to running “network services.” Some of them have been showcased in the 9th GENI Engineering Conference [4, 5]: the SmartRE project deploys software agents at each switch to detect and remove redundancy in popular media traffic, the NetServ project proposes a common platform for hosting arbitrary network services (e.g., content distribution network agents, VoIP load control agents), and the GENI-Violin project uses software agents to log snapshots of network traffic for disaster diagnosis and recovery. In a similar paradigm, this paper presents Steroid OpenFlow Service (SOS) as a network service deployment approach. SOS addresses the unique aspect of decoupling application and network protocol choices with network service agents as the means of seamless and evolving network service optimization mechanisms.

In this paper, we present SOS in the context of end-to-end TCP application throughput optimization. SOS, however, has a number of eminent features that address key challenges of emerging Internet usage models such as cloud computing, mobile computing, and media/content delivery. Take mobile computing for example; today’s Internet relies on Mobile IP [6] or similar protocols to maintain seamless persistent connections for mobile devices moving across IP subnets. To address the lossy nature of mobile devices’ wireless last-hop connection, split-TCP [7] or similar protocols were proposed in the past to optimize mobile networking performance. None of these solutions, however, have seen mass usage due to difficulty in deploying, configuring, and invoking the specialized agents (home and foreign agents for Mobile IP; split-TCP agents at the last-hop gateway) needed by these protocols. SOS, on the other hand, provides a uniform and easily accessible platform for deploying such services in a SDN.

The rest of the paper is organized as follows. Section 2 describes the OpenFlow network model, the SOS architecture, and its associated discovery mechanism. Section 3 describes the specific implementation we have completed on the NSF GENI testbed across Clemson, Stanford, University of Wisconsin, and University of Washington. Section 4 discusses the implication of SOS for various emerging applications for the future Internet. The paper concludes in Section 5.

# NETWORK MODEL

## OpenFlow Network

OpenFlow is an open standard that provides a way for researchers to test out new ideas within a campus’s network. OpenFlow is based on an Ethernet switch with an internal flow table, and provides a standardized interface in order to add and remove flow entries [8]. An OpenFlow flow entry can match a packet against L2, L3, or L4 fields, in addition to the port number on which the packet arrived. After a packet is matched, an action must be specified in order to handle the packet. The action can range from something simple like drop the packet or output the packet on certain port(s). There are also actions supported that allow one to modify the L2, L3, and L4 fields within a packet.

In an OpenFlow network, the OpenFlow switch will talk to a controller over a secure channel. When a packet goes into an OpenFlow switch, the switch’s flow table is consulted to see if there is a flow entry that matches the packet. If there is, the packet follows the action associated with the flow entry. Otherwise, the packet is sent to the controller over the secure channel, and the controller then informs the switch what to do with the packet. The controller can install a flow rule here so the next time packets of this type are seen, it will no longer need to ask the controller what to do. Another option is the controller could handle each one of these packets individually.

## SOS Architecture

The figure below shows a high level description of how SOS works from a network view.

First, at 1 in the figure above, the client initiates a TCP connection to the Server. Then at 2, the packet gets to the first OpenFlow switch and is forwarded to the controller. At 3, the controller looks at the source and destination address of the packet to determine if there is an SOS agent near the endpoints. If there is, the controller goes ahead and pushes down flow entries into the OpenFlow switches at both endpoints. This is done so that the controller does not need to be asked again about any of these packets. During this time, the controller also sends a special routing message to the agents telling them who they should create connections to. At 4, the TCP connection from the client is redirected to the first SOS agent due to the flow entry installed from 3. Then at 5, the agent fires up a number of parallel SCTP connections to the agent nearest the end host (using the routing information received from the controller). Finally at 6, the end SOS agent completes the TCP connection to the server. During this whole process, the client and server have no idea this has happened because the agents do packet rewrite to set packet headers back to their original state.

## SOS Service Discovery

In order for all of this to work dynamically, the agents need to advertise themselves so that a controller can learn about them. The agents do this by sending a special discovery packet out to the network periodically. The reason this is implemented this way is because SOS is a service that can be provided to multiple controllers and users. Using a piece of software called FlowVisor, it allows one to slice up a network, allowing researchers to have their own slice of a network to experiment [9]. Using FlowVisor, experiments that want to discover agents on the network simply need to include the flowspace for the discovery packets. This allows multiple people to use the SOS service.

# GENI TESTBED IMPLEMENTATION

## Experiment Topology and Setup

In order to test out this idea, the GENI OpenFlow testbed was used. This testbed consists of completely OpenFlow enabled switches, which are deployed along several paths of the National LambdaRail and Internet2 backbones [10]. In this experiment, four campuses participated: Clemson, Stanford, Wisconsin, and Washington University. This testbed was a great resource because of the far distance between client and server and the large pipes across the backbone.

## SOS Agent Implementation

The figure below shows the state map of how the SOS agent is implemented.

## Performance Measurement

In order to measure the performance of the SOS service, iperf was used to measure the throughput between two end points. The results are displayed in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Source | Destination | SOS (Mbps) | TCP (Mbps) |
| Clemson | Wisconsin | 18.8 | 1.21 |
| Clemson | Stanford | 38.7 | 3.55 |
| Stanford | Wisconsin | 6.41 | .591 |
| Stanford | Washington | \*TBD | \*TBD |
| Washington | Clemson | \*TBD | \*TBD |
| Washington | Wisconsin | \*TBD | \*TBD |

\*Note: we are still waiting for the network to be connected in Washington to gather these results.

As it can be noted from the results, SOS is always able to out perform TCP across the long links that were tested. This is largely due to how TCP (and SCTP) handle congestion control. Since TCP (and SCTP) increase their sending window size for each successful round trip transmission, when the RTT is large, it takes longer for the connection to grow. Since SOS uses multiple connections, it allows the sending window size to grow x number of connections faster. In addition to that, when a packet is dropped in TCP (and SCTP), the sending window size is halved. Since SOS uses multiple connections when a dropped packet occurs, only that connection is affected. Therefore, only a fraction of the total sending window size is lost, as opposed to half of the total sending window size.

## Lessons Learned

One lesson that was learned during the development process was that OpenFlow enabled hardware switches are unable to modify L3 and L4 fields of a packet at line rate. This is because these switches do not yet support this feature in hardware. In order to solve this problem, our agents are equipped with a software OpenFlow switch (openVswitch). Using this, packets can be modified at near line rate. The reason for this is because the hardware switches CPU we were using were quite slow. As a note, the PCs running the agent software were old Pentium 4 commodity PCs and the agent software barely created a load on these PCs.

# IMPLICATIONS ON FUTURE INTERNET SERVICES ARCHITECTURE

## Cloud Computing

Cloud computing has emerged as the mainstream model for future Internet based computing and storage. Via Internet based resource allocation services, end users from individuals to enterprises can flexibly request the needed computing and storage resources on demand for their applications. State-of-the-art cloud computing platforms (e.g., Amazon EC2, VMware vCloud, Eucalyptus, etc.) allocate static computing and storage instances upon request. While these services provide some capabilities of IP subnet customization for the allocated hosts, it is still challenging to support seamless live process migration with identical network setups. This is critical for scenarios such as data center disaster recovery at a different site or intra-/inter-site process migration for load balancing purposes. With SOS decoupling the network from the edge, various agent solutions can be devised to seamlessly mask network failures from computing hosts, or vice versa, host migrations from the network and remote peers.

## Mobile Computing

Mobile computing is predicted to become the dominant mode of Internet access in the near future (2 billion wired servers/PCs vs. 10 billion mobile data devices by 2020 [11]). To add to this prediction, the automotive industry is moving swiftly to connect cars to the Internet [12]. To date, supporting seamless Internet connection for mobile devices translates to significant handover overheads, delicate mobile IP agent configurations on base stations, and inferior end-to-end performance. With SOS, a wider variety of innovated agent implementations can be developed and deployed in Internet to support and enhance seamless mobile connection performance, thereby enabling more mobile applications to leverage the abundant cloud resources on Internet. With SDN and SOS, mobile end users also have the opportunity to reveal their mobility preference, projection, and history to *proactively* provision network forwarding services to achieve zero latency across link and network handovers.

## Content/Media Delivery

Content/media delivery has already been an important mode of Internet usage for years. Plenty of solutions, such as cache proxies, content distribution networks, and quality of service provisioning techniques have been studied and deployed. Deployment of such services in the network, however, is predominantly a network provider decision and involves significant man power to set up and maintain. This led to little or no room for customizing application and/or user specific connection service levels. With the SOS architecture, novel network services are envisioned to be openly deployed as new SOS agents through a standardized procedure. This procedure allows new applications to be released in conjunction with their customized SOS agents that are aware of application specific performance requirements. In turn, the agents optimize the network forwarding strategies accordingly over the SOS paradigm.

# CONCLUSIONS AND FUTURE WORK

In the future, it is planned to implement an OpenFlow controller that is able to make use of multiple paths in a network. This should allow for an even greater increase in end to end throughput. It is also planed to investigate the different traffic types that this service would benefit.

With SDN being deployed more widespread across the Internet, it allows for easier deployment and development of such services as the Steroid OpenFlow Service described in this paper. These services deployed would allow for enhancements to be added seamlessly to a network, which in turn world improve end users experiences.

# ACKNOWLEDGEMEMTS

The presented work is supported in part by a National Science Foundation grant CNS-0944089. The authors would also like to express their thanks to the technical supports provided by the Clemson Computing and Information Technology (CCIT) and the GENI OpenFlow project teams at Stanford University, University of Wisconsin, University of Washington, and the GENI Project Office at BBN Technologies.

1. **REFERENCES**
2. The OpenFlow Consortium, http://openflow.org/, last accessed in Jul. 2011.
3. The Open Networking Foundation, http://www.opennetworkingfoundation.org/, last accessed in Jul. 2011.
4. GENI: Exploring Networks of the Future, http://www.geni.net/, last accessed in Jul. 2011.
5. The 9th GENI Engineering Conference, http://groups.geni.net/geni/wiki/Gec9Agenda, last accessed in Jul. 2011.
6. GEC9 demos, http://groups.geni.net/geni/attachment/wiki/Gec9Agenda/WednesdayMorning.pptx, last accessed in Jul. 2011.
7. IETF, “RFC 3775: IP Mobility Support for IPv6,” http://www.ietf.org/rfc/rfc3775.txt, Jan. 2004.
8. H. Balakrishnan, et al., “Improving TCP/IP Performance over Wireless Networks,” in Proceedings of ACM MobiCom’95, pp.2-11, 1995.
9. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J.Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2): 69-74, April 2008.
10. Openflow, http://flowvisor.org. Last accessed in Jul. 2011.
11. Explore Networks of the future, http://geni.net. Last accessed Jul. 2011
12. The MobilityFirst Project, http://www.nets-fia.net/Meetings/Nov10/MobilityFirst\_FIAmtg\_summary\_11.10.pdf, last accessed in Jul. 2011.
13. Networked Vehicle Association, http:// www.networkedvehicle.org/. Last accessed in Jul. 2011.