Plug-n-Serve: Load-Balancing Web Traffic using OpenFlow

Nikhil Handigol*, Srinivasan Seetharaman†, Nick McKeown*, Ramesh Johari*
* Stanford University, Palo Alto, CA USA
† Deutsche Telekom R&D Lab, Los Altos, CA USA

ABSTRACT

Effective load-balancing systems for services hosted in unstructured networks need to take into account both the congestion of the network and the load on the servers. In this demonstration, we illustrate a comprehensive load-balancing solution that works well for such networks. The system we showcase, called *Plug-n-Serve*, tries to minimize response time by controlling the load on the network and the servers using customized flow routing.

The demonstration shows how the Plug-n-Serve system works within our deployment in the CS building at Stanford University. Besides the base behavior, we show the effect of dynamically adding and removing computing resources to the system, increasing the request arrival rate, altering the CPU or network load of each request, and changing load-balancing algorithms.

Categories and Subject Descriptors: C.2.2 – Computer Systems Organization [**Computer-Communication Networks**]: Network Architecture and Design; C.4 – Computer Systems Organization [**Performance of Systems**]

General Terms:
Management, Design, Experimentation

Keywords:
Load balancing, OpenFlow, Architecture, Unstructured

1. MOTIVATION

It is common for a large web sites to balance load over many HTTP servers, and there exist commercial products to do this [1, 2]. Load-balancing may be oblivious (e.g., spreading the requests equally over all servers, without regard for their load), or stateful (e.g., sending requests to the least-loaded server). In a data-center or a dedicated web-hosting service, the HTTP servers are connected by a regular, over-provisioned network; the load-balancer usually does not consider the network state when load-balancing across servers.

However, this simplistic scenario does not hold for unstructured networks, such as enterprise and campus networks, that are not custom-built for running server farms. In such unstructured networks, the substantial background traffic and the potential topological biases can significantly affect the performance of network-oblivious load-balancing (our baseline), and inflate the *response time* (defined as the duration from issuing the HTTP request to the complete receipt of the response).

In our work, we ask the question: "If we simply add many HTTP servers to our own enterprise or campus network, regardless of its topology, what is the best way to balance load so as to minimize the client response time?". In particular, we take into account both the congestion of the network and the load on the servers, and, then, control the load on the network and the servers to try to minimize response time.

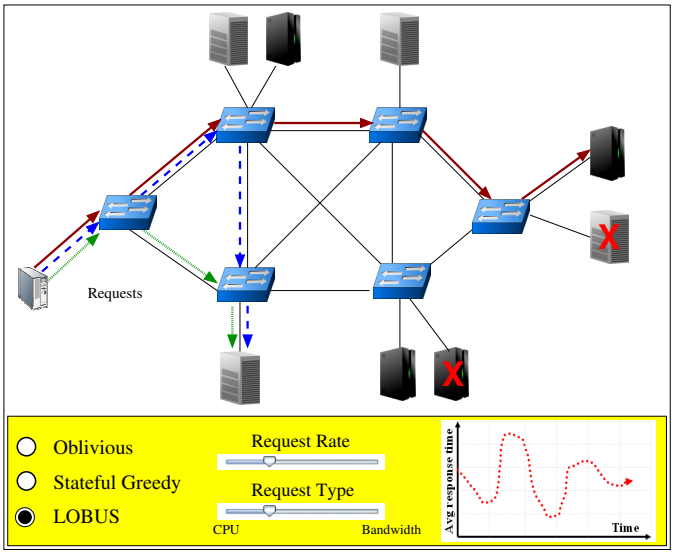

Figure 1: The frontend GUI shows the up-to-date state of the system and allows the user to control various parameters. The state of each server can be toggled by clicking. OpenFlow channels requests to different servers, or through different paths to the same server.

We demonstrate a load-balancer, called *Plug-n-Serve*, that load-balances over arbitrary unstructured networks, and tries to minimize the average response time. The system allows operators to increase the capacity of the web service by simply plugging in computing resources and switches in an arbitrary manner. In response, the Plug-n-Serve controller implementing an integrated optimization algorithm we developed, called *LOBUS* (LOad-Balancing over UnStructured networks), automatically expands its view of the network, and appropriately shares the load over the added devices.

2. DEMONSTRATION

Our demonstration presents Plug-n-Serve in action in the Gates Computer Science Building at Stanford University.

Web servers are randomly spread across an OpenFlow network [4]. The network, which is also used by production traffic, consists of a heterogeneous mix of commercial switches from Cisco, HP and NEC in addition to the NetFPGA-based ones. The OpenFlow switches [6] are remotely controlled by a NOX-based Plug-n-Serve controller that runs on a separate PC [3, 5]. HTTP-based client requests are generated by several PCs within the Gates network.

In our demonstration, we show the performance of LOBUS and compare it with that of oblivious and stateful load-balancing approaches. Figure 1 shows a snapshot of the frontend GUI provided by Plug-n-Serve. It captures three main aspects of the system, namely:

- Up-to-date state of the overall system: The load on the servers, and the congestion of the network links and switches.
- The average response time for the requests as a time-series.
- The effect of dynamically adding or removing servers and network links on response time.

The GUI allows us to vary the request arrival rate. We can also vary the work brought by each new request, based on whether the request adds more load to the CPU (e.g. computation intensive requests) or to the network (e.g. for high bandwidth data from the servers, such as video). Lastly, the GUI allows us to change the load-balancing algorithm.

3. DESIGN AND IMPLEMENTATION

All servers used in the demonstration are assigned the same IP alias. When a request arrives for the server IP address, the controller decides which server to route it to and the path it should take. Once a flow has been allocated to a server, all the packets in the flow are forwarded at line-rate to that server by the datapath of one or more deployed switches.

To implement this, Plug-n-Serve does the following:

- It determines the current state of the network and the servers, including the network topology, network congestion, and load on the servers.
- It chooses the appropriate server to direct requests to, and controls the path taken by packets in the network, so as to minimize the response time.

We use the newly proposed OpenFlow architecture to *measure* the state of the network, and to directly *control* the paths taken by new HTTP requests. OpenFlow is an open routing platform which provides vendor-independent means to control the way switches route traffic. The Plug-n-Serve controller is capable of managing a large network of switches and servers.

To allocate web requests, Plug-n-Serve relies on the following three functional units, as illustrated in Figure 2:

- **Flow Manager**: This module is an OpenFlow controller that manages and routes flows based on the specific load-balancing algorithm chosen. This controller also handles the necessary Layer 2 protocols (viz., DHCP, ARP, STP). The LOBUS algorithm is implemented in this module.

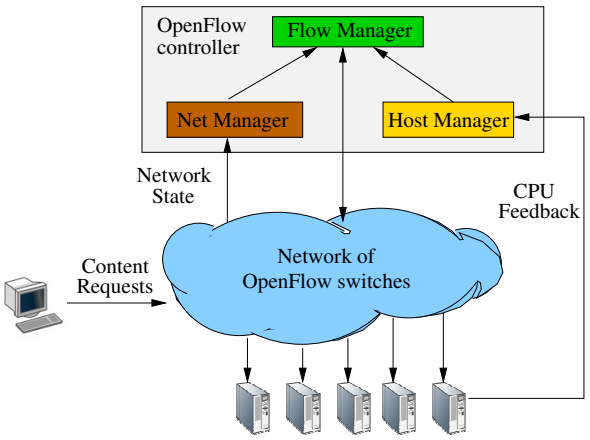

Figure 2: The main control logic of Plug-n-Serve, implemented as an OpenFlow controller, consists of three functional units.

- **Net Manager**: This module is responsible for probing the network, and keeping track of the network topology and its utilization levels. It queries switches periodically to get link usage and monitors the latency experienced by packets traversing the links.
- **Host Manager**: This component monitors the state and load at individual servers in the system, and reports it to the Flow Manager. It also detects new servers plugged into the load-balancer system.

4. CONCLUDING REMARKS

We showcase Plug-n-Serve, an Open-Flow based server load-balancing system that effectively reduces response time of web services in unstructured networks built with cheap commodity hardware. Using OpenFlow to keep track of state and to control the routes allows the system to be easily reconfigured; the network operator, thus, can add or remove capacity by turning hosts on or off, and add or remove path diversity by turning switches on or off. The demonstration, with its many tunable options, provides insights into the effectiveness of the LOBUS algorithm we developed. Furthermore, it allows us to identify and understand scenarios where it is beneficial to have information of the network topology and the level of background load on the resources.

5. REFERENCES

[1] Foundry ServerIron Load Balancer. `http://www.foundrynet.com/products/webswitches/serveriron/`.
[2] Microsoft's Network Load Balancing. `http://technet.microsoft.com/en-us/library/bb742455.aspx`.
[3] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: Towards an operating system for networks. In *ACM SIGCOMM Computer Communication Review*, July 2008.
[4] 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.
[5] NOX - An OpenFlow Controller. `http://www.noxrepo.org`.
[6] The OpenFlow Switch Consortium. `http://www.openflowswitch.org`.