Aster*x: Load-Balancing Web Traffic over Wide-Area Networks

Nikhil Handigol*, Srini Seetharaman†, Mario Flajslik*, Aaron Gember‡,
Nick McKeown*, Guru Parulkar*, Aditya Akella‡, Nick Feamster◇,
Russ Clark◇, Arvind Krishnamurthy×, Vjekoslav Brajkovic×, Tom Anderson×

* Stanford University, † Deutsche Telekom R&D Lab USA,
‡ University of Wisconsin-Madison, ◇ Georgia Tech, × University of Washington

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 whitepaper, we propose a comprehensive load-balancing solution that works well for such networks. The system we propose, called *Aster*x*, tries to minimize response time by controlling the load on the network and the servers using customized flow routing.

We hope to use the GENI infrastructure to understand how Aster*x works within a combined network and computation slice spanning multiple campuses across the country; the GENI infrastructure is well-suited for this purpose because of the distributed and realistic nature of the computing and networking substrate. Besides the base behavior, we plan to investigate 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.

Keywords:
Load balancing, OpenFlow, Architecture, Unstructured

1. MOTIVATION

It is common for 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 wide-area 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 host a service across many HTTP servers spanning multiple campus networks, what is the best way to balance load so as to minimize the client response time?". In particular, we take into account the congestion of the network, the location, and the load on the servers, and, then, control the load on the network and the servers to try to minimize response time.

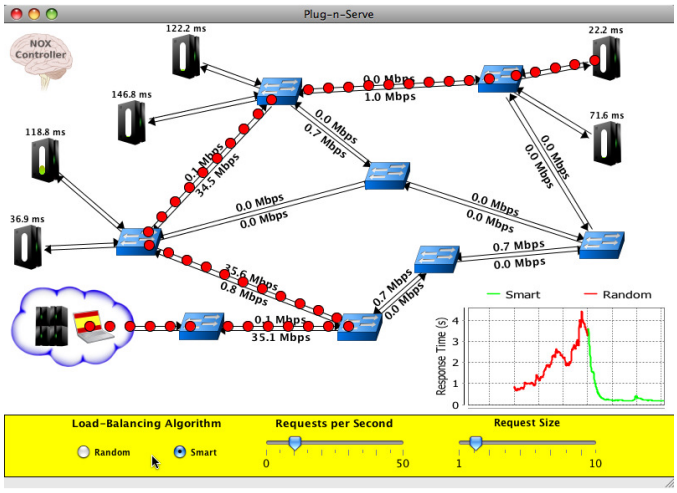

Figure 1: The frontend GUI shows the up-to-date state of the system and allows the user to control various parameters. The state of each server can be toggled by clicking. OpenFlow channels requests to different servers, or through different paths to the same server.

We propose a load-balancer, called *Aster*x*, that balances load 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 Aster*x controller implementing an integrated optimization algorithm we developed, automatically expands its view of the network, and appropriately shares the load over the added devices.

2. DEMONSTRATION

We plan to demonstrate Aster*x in action within a GENI slice spanning at least 3 university campuses across the country. Web servers are hosted over PlanetLab nodes interconnected by an OpenFlow network [14, 16]. The OpenFlow slice is remotely controlled by a NOX-based Aster*x controller that runs on a separate PC [10, 15]. HTTP-based client requests are generated by several clients from multiple locations.

In our demonstration, we will show the performance of our

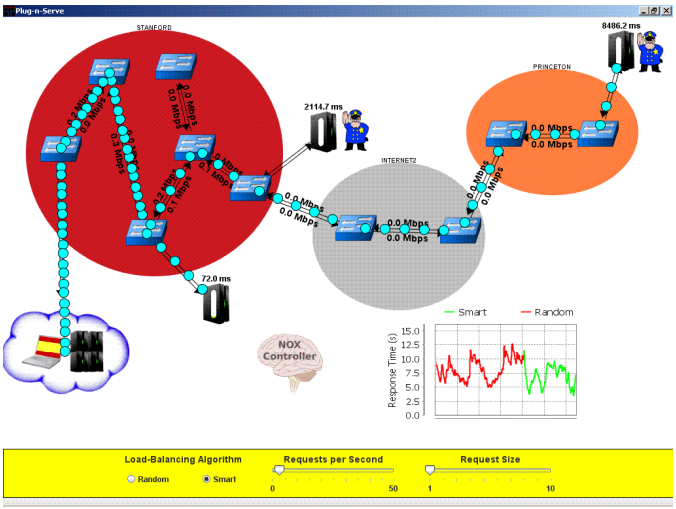

Figure 2: A screenshot of the frontend GUI of the Plug-n-Serve demonstration at GEC-6, 2009.

algorithm and compare it with that of oblivious and stateful load-balancing approaches. Figure 1 shows a snapshot of the frontend GUI provided by Aster*x. 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. WHAT'S NEW?

An older version of Aster*x, called Plug-n-Serve, has been demonstrated in the past at SIGCOMM, 2009 [11] (Fig. 1) and GEC-6, 2009 (Fig. 2). The previous demos showcased the Aster*x architecture and its performance in a local area network. While the proposed demo/experiment builds on the two successful previous demos, it will differ significantly from them in the following ways:

- **The GENI infrastructure** - Aster*x will run on a GENI slice comprising OpenFlow-based network elements and PlanetLab-based computation elements. The demo will highlight the benefits of a nationwide integrated testbed like GENI, as it brings together the networking and computation substrate under a single experiment.
- **Scale** - It will run over a slice spanning at least 3 university campuses across the United States. This will be at a much larger scale than the previous ones.
- **Client diversity** - It will also showcase clients using the load-balanced web-service from multiple locations in the country.
- **Algorithms** - It will also evaluate the novel algorithms developed for load-balancing traffic in wide-area networks.

4. RESEARCH IMPLICATIONS

The Aster*x project has research implications much beyond the demonstration itself. Over the past decade there has been a change in the way content is hosted and served in the Internet. To meet growing demand, and minimize access time, service is replicated across multiple servers in possibly multiple locations. Content Distribution Networks (CDN), such as those run by Akamai [3], Limelight [13], and Amazon [4] serve content from thousands of servers around the world. Some large network operators have built their own CDNs to generate revenues from content customers [5, 7, 12]. Service replication is not limited to just simple web-content. Novel services like Google DNS [9] (designed to accelerate browsing experience) replicate and serve DNS records from many servers. Protocols such as OPES and ESI [6, 8] are used to allow application servers to cache generated content. Large cloud service providers, such as Google and Yahoo, use a global network of datacenters to ensure low latency for their clients. Replicated content hosting and delivery and load-balancing over the wide-area is, therefore, an important area of revenue and research. We are likely to see a number of projects and publications on both novel systems and algorithms for wide-area load-balancing in the near future. Aster*x is one such proposed project that falls in this category.

Successful evaluation and adoption of the load-balancing systems and algorithms faces two major hurdles:

- **Infrastructure** - Experiments to evaluate the performance of the algorithms need to be run on "real" wide-area networks. While it is possible to connect multiple small network islands via tunneled communication channels (a non-trivial effort in itself), the fact remains that these are not "real" wide-area networks and the performance numbers cannot be realistic.
- **Reconfigurable topologies** - The experiments will also need to show that the algorithms work on a large variety of topologies. Building a large number of wide-area topologies is almost infeasible today - both economically and practically. While this problem has been solved to some extent for distributed systems with PlanetLab [17] and Emulab [18], there exists no equivalent solution for networking systems today.

The GENI infrastructure helps systems overcome both the above hurdles. In fact, we believe the GENI infrastructure is necessary for the performance evaluation of systems like Aster*x. With experiments on the GENI infrastructure, systems like Aster*x can make a strong case for themselves, leading not only to research publications but also faster adoption and evolution.

5. DESIGN AND IMPLEMENTATION

All servers in Aster*x 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

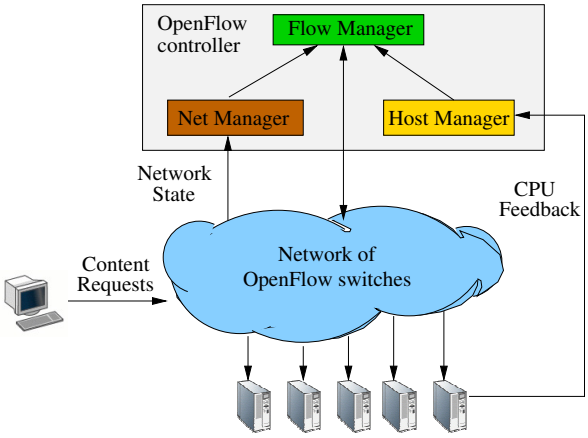

Figure 3: The main control logic of Aster*x, implemented as an OpenFlow controller, consists of three functional units.

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, Aster*x 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 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 Aster*x controller is capable of managing a large network of switches and servers.

To allocate web requests, Aster*x relies on the following three functional units, as illustrated in Figure 3:

- **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 load-balancing algorithm is implemented in this module.
- **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.

6. CONCLUDING REMARKS

We propose Aster*x, a 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.

We plan to deploy Aster*x, with its many tunable options, within a GENI slice to obtain insights into the effectiveness of the load-balancing algorithm we developed. Furthermore, this trial deployment will allow 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. The GENI infrastructure, with its computing and networking resources, serves as a crucial platform for experimentation.

7. 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] Akamai Content Distribution Service. `www.akamai.com`.
[4] Amazon tees up content delivery service. `http://news.cnet.com/8301-1023_3-10045350-93.html`.
[5] AT&T Announces New Digital Media Solutions Portfolio. `http://www.att.com/gen/press-room?pid=4800&cdvn=news&newsarticleid=25853`.
[6] A. Beck, M. Hofmann, H. Orman, R. Penno, and A. Terzis. Requirements for open pluggable edge services (opes). RFC 3836, 2004.
[7] T-Mobile parent company launches CDN. `http://www.networkworld.com/news/2009/012609-deutsche-telecom-cdn-services.html`.
[8] Edge server includes (esi) language specification. `http://www.w3.org/TR/esi-lang`.
[9] Introducing Google Public DNS. `http://googleblog.blogspot.com/2009/12/introducing-google-public-dns.html`.
[10] 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.
[11] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari. Plug-n-Serve: Load-Balancing Web Traffic using OpenFlow. In *ACM SIGCOMM Demo*, August 2009.
[12] Level3 Content Delivery Network. `http://www.level3.com/index.cfm?pageID=36`.
[13] Limelight Networks - Content Delivery Network. `www.limelightnetworks.com`.
[14] 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.
[15] NOX - An OpenFlow Controller. `http://www.noxrepo.org`.
[16] The OpenFlow Switch Consortium. `http://www.openflowswitch.org`.
[17] An open platform for developing, deploying, and accessing planetary-scale services. `http://www.planet-lab.org/`.
[18] 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. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, Dec. 2002. USENIX Association.