Multi-Path TCP Performance Evaluation

18MCPC12

ACN Term Paper, Submission to Dr. P.Anupama

Advanced Computer Networks (ACN) Elective Course Work

SCIS, University of Hyderabad

April 15, 2019

1 Introduction

TCP/IP communication is currently restricted to a single path per connection, yet multiple paths often exist between peers. The simultaneous use of these multiple paths for a TCP/IP session would improve resource usage within the network and, thus, improve user experience through better resource utilization, better throughput and smoother reaction to failures.

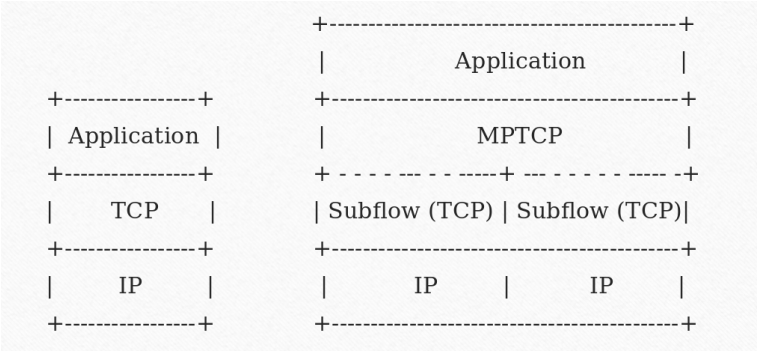

Figure 1: Comparision of standard TCP and MPTCP Protocol Stacks [1]

Multipath TCP (MPTCP) enables a transport connection to operate across multiple paths simultaneously over which an application can communicate between two hosts by setting up multiple paths ("subflows") and managing these subflows. MPTCP

operates at the transport layer and aims to be transparent to both higher and lower layers as shown in Figure 1.
An MPTCP connection begins similarly to a regular TCP connection. This is illustrated in Figure 2. Where an MPTCP connection is established between addresses A1 and B1 on Hosts A and B, respectively. If extra paths are available, additional TCP sessions (termed MPTCP "subflows") are created on these paths, and are combined with the existing session, which continues to appear as a single connection to the applications at both ends. The creation of the additional TCP session is illustrated between Address A2 on Host A and Address B1 on Host B. MPTCP identifies multiple paths by the presence of multiple addresses at hosts. Combinations of these multiple addresses equate to the additional paths. In the example, other potential paths that could be set up are $A1 <-> B2$ and $A2 <-> B2$. Although this additional session is shown as being initiated from A2, it could equally have been initiated from B1.The discovery and setup of additional subflows will be achieved through a path management method by specified path manager i.e. in this experiment its default path manager(Mesh) [1].

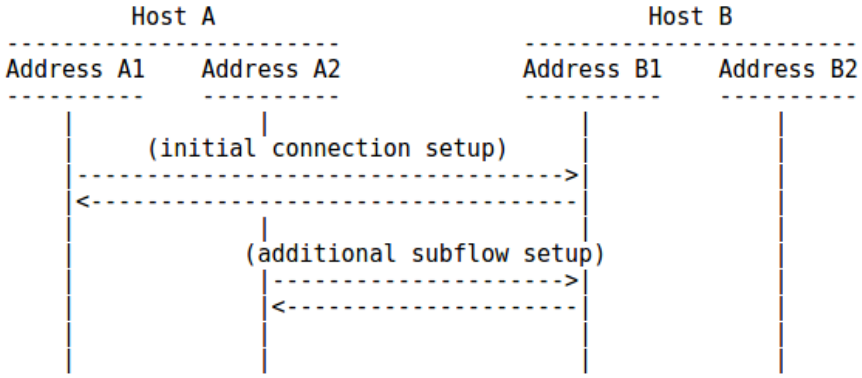

Figure 2: Example MPTCP Usage Scenario [1]

MPTCP Assumptions as in [1] are:

- It must be backwards-compatible with current, regular TCP, to increase its chances of deployment.
- It can be assumed that one or both hosts are multihomed and Multi addressed.

2 Experimental Setup

The aim of the experiment is to verify the behavior of the current implementation of the protocol Multipath TCP.

2.1 Challenges encountered

We encountered three major challenges throughout the course of the project:

- Rebuilding the kernel and patching MPTCP: We eventually solved this by using a snapshot of the development of branch of MPTCP from May 17, 2014 with version 3.14 of the Linux kernel (using Ubuntu 14.04), and manually applying the patch to MPTCP. Mininet was chosen and scripts are written in Python.
- Making MPTCP work correctly
- MPTCP Flow control Analysis

2.2 Configuration Parameters

- Version == 0.89.x. To set a sysctl variable, just do: sysctl -w net.mptcp.[name of the variable]=[value]
- net.mptcp.mptcp_enabled Disable/Enable MPTCP on this machine. Possible values are 0 or 1. (default 1). Also enabled as part of setsockopt();
- net.mptcp.mptcp_checksum Disable/Enable the MPTCP checksum. As described in the draft, both sides (client and server) have to disable DSS-checksums. If one side has it enabled, DSS-checksums will be used. Possible values are 0 or 1. (default 1)
- net.mptcp.mptcp_syn_retries Specifies how often we retransmit a SYN with the MP_CAPABLE-option. After this, the SYN will not contain the MP_CAPABLE-option. This is to handle middleboxes that droping SYNs with unknown TCP options. Default is 3.
- net.mptcp.mptcp_path_manager, select for example the full-mesh path-manager. If you do not select a path-manager, the host will not trigger the creation of new subflows, nor advertise alternative IP-addresses through the ADD_ADDR-option.

- net.mptcp.mptcp_scheduler 'default': It will first send data on subflows with the lowest RTT until their congestion-window is full. Then, it will start transmitting on the subflows with the next higher RTT.

3 Result Analysis

3.1 Observations Through Wireshark

- MPTCP connection is established by using three-way handshake with TCP options. MP_CAPABLE option in the SYN packet indicates that the source can perform multipath TCP. If the destination also supports multipath TCP, then it replies with SYNACK packet, which also contains MP_CAPABLE option, after that source replies with ACK packet which again contains MP_CAPABLE option to confirm that the connection will use MPTCP shown in Figure 3.

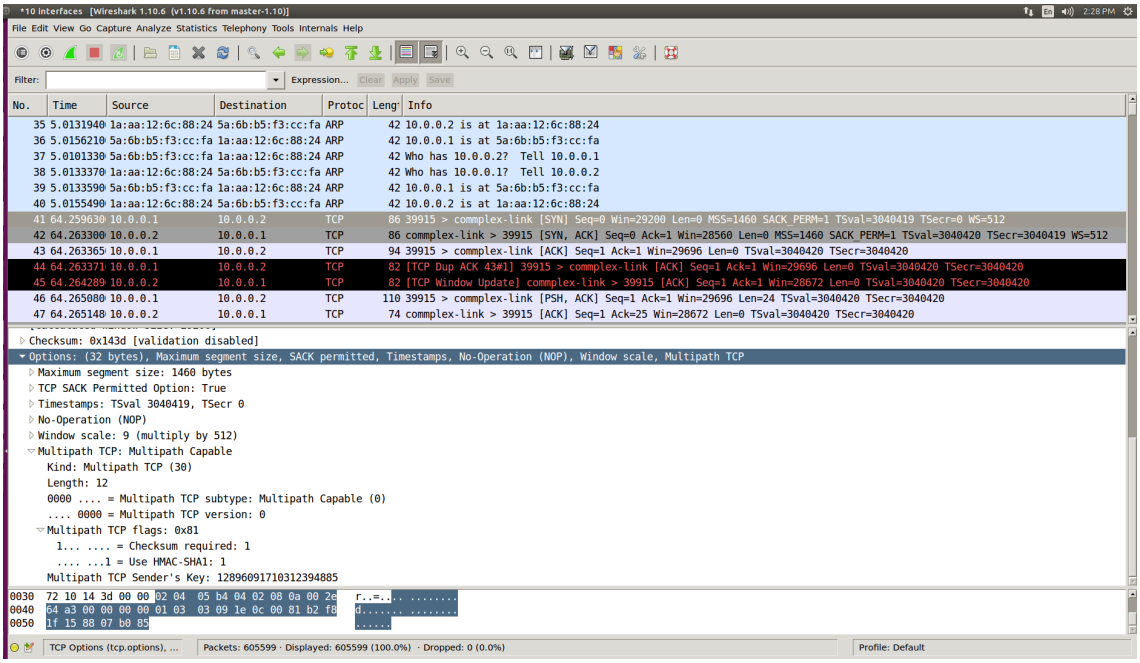

Figure 3: MPTCP SYN packet capture from Wireshark

- Once the connection established any of participants can create additional subflow if it finds a new path to the destination by sending a new SYN packet with MP_JOIN option and information on which MPTCP session to join. Each

sub-flow looks like a normal TCP connection. It has its own sequence numbers and congestion control which are different from other sub-flows.

- Once an MPTCP connection has been initiated over one interface, MPTCP will announce the other available IP addresses to the remote host by using the ADD_ADDR option and subflows will be established over these interfaces.
- A New sub-flow can come up if a new path is available and can vanish if one of the paths becomes unavailable.
- Since path manager is set as full mesh, two interfaces from onside are sending connection establishment requests to other side two interfaces and vice versa. Thats is 4 connections are possible between client and server with 2-interfaces each.
- Also mptcp_syn_retries is 3 seconds (default). For every 3 seconds a new connection establishment is observed for each existing flow.
- MPTCP using default scheduler. It will first send data on subflows with the lowest RTT until their congestion-window is full. Then, it will start transmitting on the subflows with the next higher RTT.
- However Congestion control was not clear with references experimental and theoretical standard references [1], [2] and [3] in my observation. So **future work** can be focused on this area.

3.2 Test scenarios

In order to investigate MPTCP behaviors, we have deployed three different scenarios:

1. 1-host iperf traffic:(a) Client and server, each of them is equipped two Ethernet interfaces as shown in Fig. 2. They create an MPTCP connection of 2 paths. (b) single interface mptcp client to server. (c) TCP client to server
2. 4-hosts iperf traffic: (a) 4-Clients and server, each of them is equipped two Ethernet interfaces. They create an MPTCP connection of 2 paths and and sending iperf traffic simultaneously from each client. (b) single interface mptcp clients to server simultaneously. (c) TCP clients to server simultaneously.

3. 4-hosts iperf traffic to 4-interfaces server: Clients each of them is equipped two Ethernet interfaces where as server with 4-interfaces. They create an MPTCP connection of 4X2 paths.

All presented measurements were performed in the uplink direction. That is on the client is actual data flow captured over all used interfaces. For generating dataow, we used the TCP protocol (Iperf with 2GB traffic). The results obtained from the iperf were then shown in the figures as graphs to measure the performance of all above mentioned three scenarios. Each individual experiment is tested 10 times and calculated as mean value. The basic topology for all the experiments is as shown in Figure 4. All links are configured with same bandwidth (1Gbps) throughout the network topology.

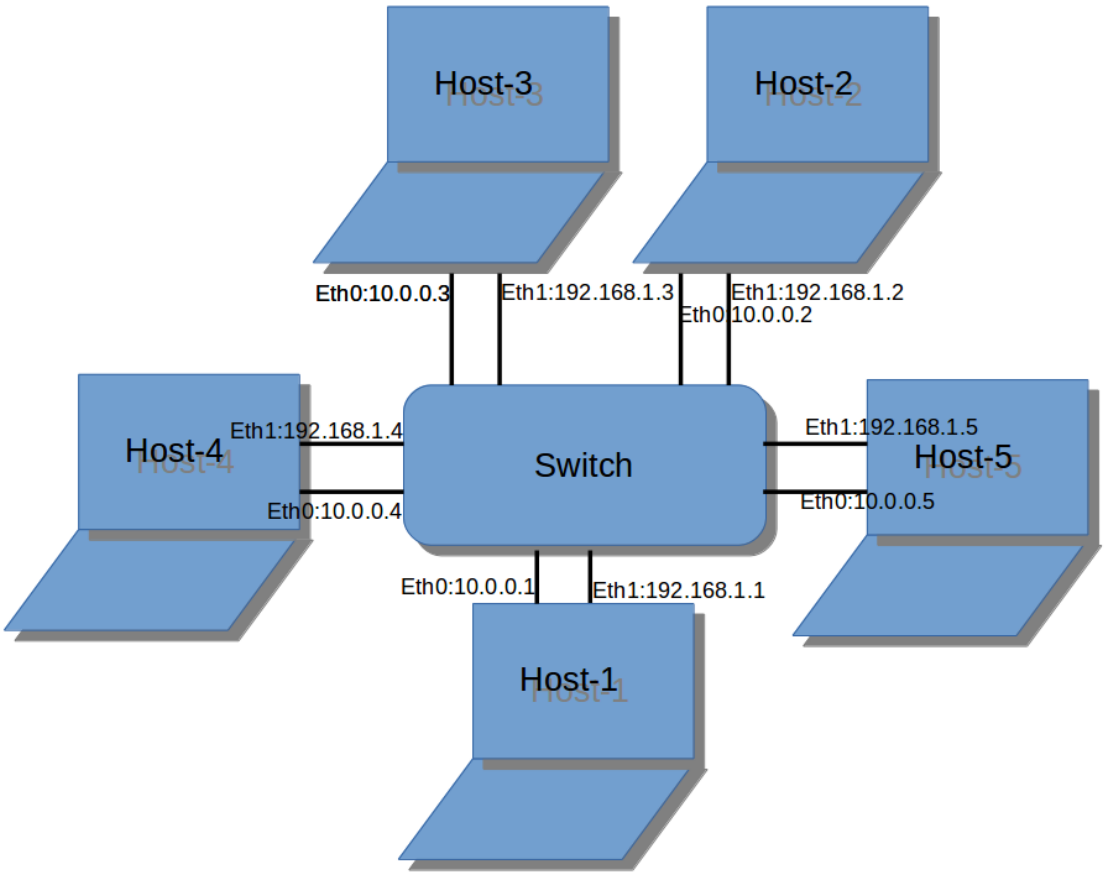

Figure 4: Network Topology: Host-2 (server) and Host-1, Host-3, Host-4, Host-5 (clients) are connected through a switch

In **scenario 1**, The MPTCP connection over two Ethernet paths outperforms

the single Ethernet TCP connection. MPTCP throughput is 1.2 Gbit/s while TCP throughput is only 1286Mbit/s as shown in Figure 5 observed from iperf output. Moreover, the load sharing of the two MPTCP paths is quite equitable with The first path takes 50.38% (1.0608GB/2.1GB) connection load while the second one spends 49.62% (1.0445/2.1GB)remaining load as observed using ifstat tool statistics. We also recognize that both two paths throughput are so even without abnormal peaks. Therefore, we can benefit from MPTCP in the homogeneous context where all paths have similar characteristics in terms of bandwidth and delay.
Single flow MPTCP performing slightly better than TCP. This reason can be congestion control algorithm used by MPTCP is better than normal TCP. Also number of packets sent by MPTCP is slightly higher than TCP which was observed using ifstat tool.

In **Scenario - 2**, We increased to 4 hosts sending traffic simultaneously to the server. Measurement graphs are shown in Figure 7, Figure 8, Figure 9 and Figure 10. As specified in scenario 1, MPTCP performs better than the single Ethernet TCP connection. Single flow MPTCP performing slightly better than TCP in scenario 1. But in scenario 2, we observed 3-sets of mean throughput in which sometimes TCP mean throughput is slightly better than MPTCP.
Any multipath congestion control algorithm must meet three different goals. The first goals is, MPTCP would have throughput same as standard TCP. The second, a multipath sub-flow should not take capacity more than single path TCP would on the same link, this ensure that multipath sub-flow will not harm other flows. The third goal is balancing the congestion; the multipath flow should move as much as traffic to the least congested paths. However, if each multipath flow sends more traffic through its least congested path, the network traffic will move away from congested paths resulting in improvements in robustness and overall network throughput.
With third goal, we can clearly observe MPTCP congestion control algorithm better performs by shifting to the lease congested path as in [3]. Hence MPTCP single flow has better scope to achieve higher throughput and transfer speed. Figure 11, Figure 12, and Figure 13 shows the time taken to transfer 2GB traffic using over scenario 1, 2 and 3. Which again shows better performance MPTCP 2-flows, MPTCP-1 flow and TCP in order. Which is also observed using ifstat tool as the number of packets transferred over period are smarr in the same order (MPTCP-2 flow, MPTCP-1 flow and then TCP) .

Scenario - 3 measurement graphs are in Figure 6 and Figure 13. Both bandwidth and time to finish iperf 2GB transfer of 4-interfaces are better than 2-interfaces. With bandwidth aggregation and better congestion control algorithm of MPTCP, The flows are multiplexed from client side to server side. As shown in Figure 6, the first flow which is initiated first getting better throughput and finished faster than other flows because first flow finishes handshake first and get scheduled to send traffic first before even other flows will finish their handshake procedure. Also MPTCP default scheduler sending first flow's data since this flow has the lowest RTT as observed during experiment with ping statistics as part of script.

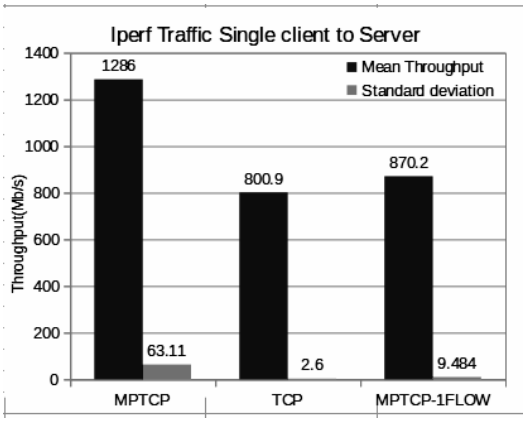

Figure 5: 1-host iperf traffic

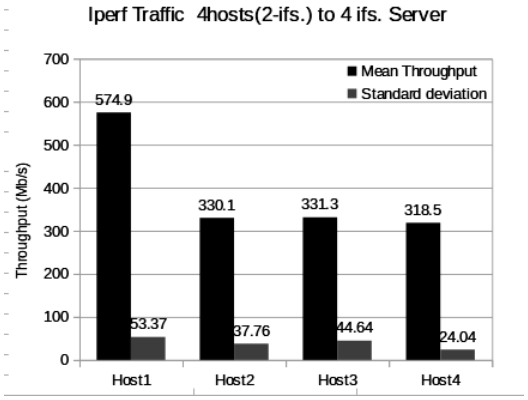

Figure 6: 4-hosts traffic to 4-ifs server

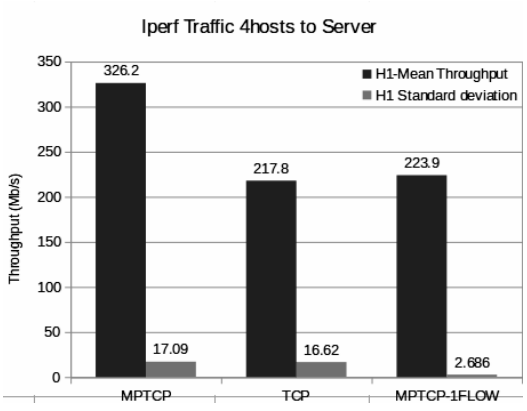

Figure 7: 4-hosts traffic - Host1 Results

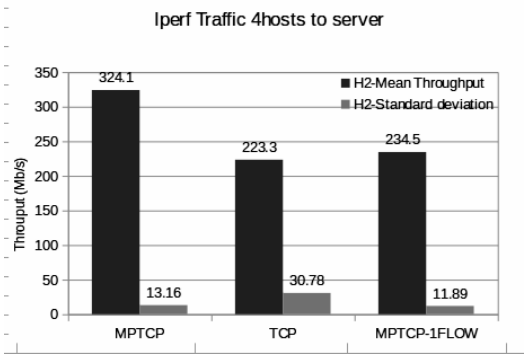

Figure 8: 4-hosts traffic - Host2 Results

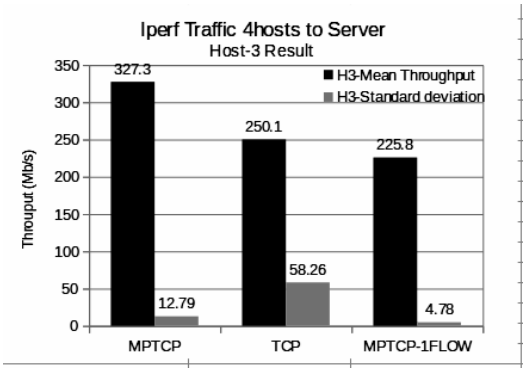

Figure 9: 4-hosts traffic - Host3 Results

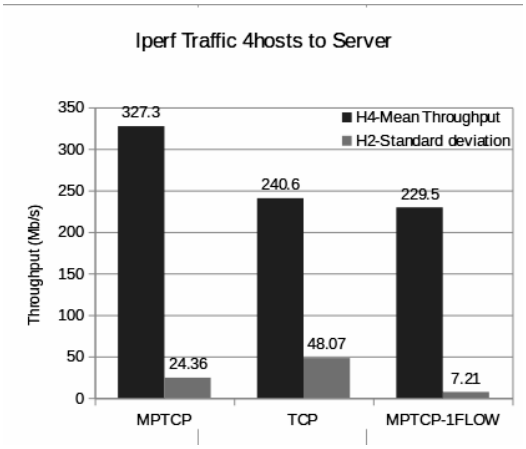

Figure 10: 4-hosts traffic - Host4 Results

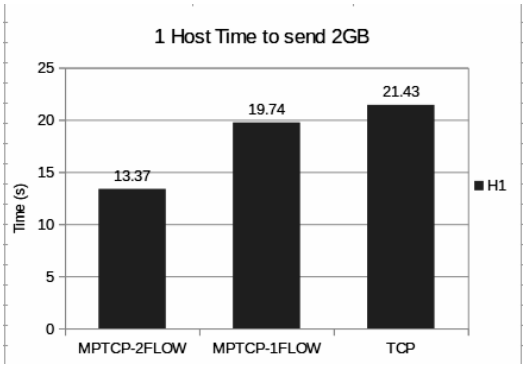

Figure 11: 1-host time to send 2GB

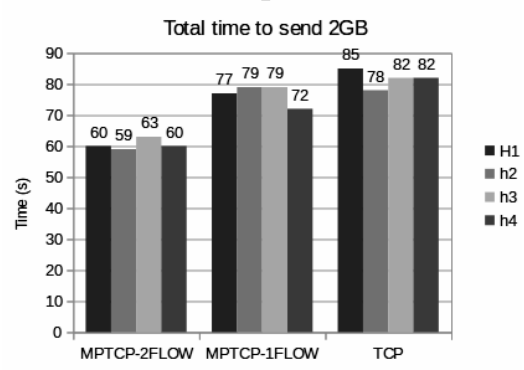

Figure 12: Time for 4-hosts simultaneously sending 2GB

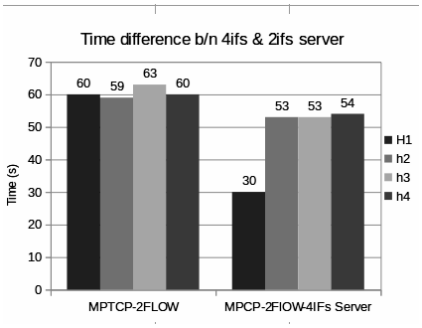

Figure 13: Time for 4-hosts simultaneously sending 2GB on 2-ifs & 4-ifs server

4 Summary

In summary, The following three questions are answered through this experiment results and analysis:

1. Why MPTCP has higher throughput than TCP? (Ans: every single connection through diverse path increases amount of available bandwidth. By bandwidth aggregation over multiple paths, and the resilience by switching traffic upon path failure helps MPTCP achieving higher throughput).
2. When is the second flow starts? (Ans: As soon as MPTCP 3-way handshake finishes, all the available flows are established even before data transmission of first flow. This is observed using Wireshark capture and the theory as in RFC [1]).
3. How server's bandwidth is divided among 4-clients? (Ans: Our experiments show that nal bandwidth is sum of sub ows bandwidth).

Hence, The presented measurements demonstrate capability of Multipath TCP to effectively utilize multiple network paths and provide increased availability, bandwidth and resilience to link failures.

Acknowledgments

The author is grateful to course instructor Dr.P.Anupama for clarifications, MTech. project student M.Thushar for helping with Mininet and MCA project students - T. Sachin and C.Rahul for helping in MPTCP kernel module installation. This work is presented as a term paper in Advanced Computer Networks (ACN) course as part of my PhD. coursework.

References

[1] https://tools.ietf.org/html/rfc6824

[2] Multipath TCP resources:http://www.multipath-tcp.org.

[3] D. Wischik, C. Raiciu, A. Greenhalgh, andM. Handley, Design, implementation and evaluation of congestion control for multipath tcp,in Proceedings of the 8th USENIX Conference On Networked Systems Design and Implementation, ser. NSDI11. Berkeley, CA, USA: USENIX Association, 2011, pp. 88. [Online]. Available:http://dl.acm.org/citation.cfm?id=1972457.1972468