Exploiting Path Diversity with Multipath TCP

Savvas Zannettou (student) , Fragkiskos Papadopoulos, Michael Sirivianos
Cyprus University of Technology
sa.zannettou@edu.cut.ac.cy, {f.papadopoulos, michael.sirivianos}@cut.ac.cy

1 Introduction

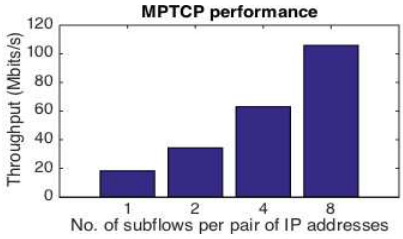
Recently, the IETF standardized a new transport layer solution called Multipath TCP (MPTCP) [2]. MPTCP takes advantage of multiple network interfaces simultaneously. It can achieve higher performance and robustness than regular TCP while maintaining compatibility with existing applications. To achieve this compatibility, MPTCP presents a regular TCP interface to applications but in fact it spreads the data through multiple flows called subflows.

An interesting aspect of MPTCP is subflow creation. The component that is responsible for the creation of subflows is called path manager. Currently, the MPTCP Linux Kernel implementation offers two path managers, *fullmesh* and *ndiffports* [1]. However, these path managers are unable to create more than one subflow between each pair of IP addresses when MPTCP uses multiple interfaces at the source and/or destination. This limitation prevents MPTCP from exploiting multiple network paths between the same pair of IP addresses.

In this work, we propose and evaluate a new path manager for MPTCP called M-fullmesh. M-fullmesh combines characteristics from *both* of the aforementioned path managers. Specifically, M-fullmesh utilizes all available pairs of IP addresses between two hosts (akin to fullmesh), and also gives the flexibility to change arbitrarily the number of subflows per pair of IP addresses (akin to ndiffports). The main idea behind M-fullmesh is that by creating more subflows and distributing them through different paths we can achieve higher network utilization, which can lead to significantly improved MPTCP performance and robustness.

2 Evaluation

We have implemented the M-fullmesh path manager in the existing MPTCP Linux Kernel, and used Mininet for our evaluation. Our topology consists of one source and one destination with two interfaces each; each interface has its own IP address. These hosts are connected through a core network with 8 different (not necessarily edge-disjoint) paths between each pair of source-destination IP addresses. The capacity of the links connecting the hosts to the core network is set sufficiently high so that these links are not bottlenecks. For the distribution of the subflows we used a custom-made Floodlight forwarding module that installs subflow routes in a round-robin fashion across the available paths. With this forwarding module, each subflow is distributed through a different path in the topology.



The figure shows the throughput of long-lived MPTCP connections. We observe that by increasing the number of subflows per pair of IP addresses we achieve significantly higher throughput, because we utilize more available network paths. Specifically, with 8 subflows per pair of IP addresses we utilize 32 paths in our topology, compared to only 4 paths when using 1 subflow per pair of IP addresses. This difference in path utilization corresponds to 83% improvement in terms of overall throughput. We intend to demonstrate M-fullmesh with a demo.

References

[1] *MPTCP Linux Kernel Configuration*. http://multipath-tcp.org/pmwiki.php/Users/ConfigureMPTCP.

[2] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *NSDI*, Berkeley, CA, USA, 2012.

Exploiting Path Diversity with Multipath TCP

Savvas Zannettou, Fragkiskos Papadopoulos, Michael Sirivianos
Cyprus University of Technology

Motivation

Simultaneously use all available interfaces and utilize as much as possible paths in the network using Multipath TCP (MPTCP)

- Create multiple subflows for each pair of Source/Destination IP addresses
- Distribute these subflows through different paths

→Better performance, network utilization and robustness

Existing MPTCP Limitation

Existing path managers are unable to create more than one subflow per pair of interfaces in multi-interface settings

fullmesh : Utilizes all pairs of Source/Destination IP addresses by creating only one subflow per pair
ndiffports: Enables the creation of multiple subflows between only one pair of Source/Destination IP addresses (other pairs, if any, remain unutilized)

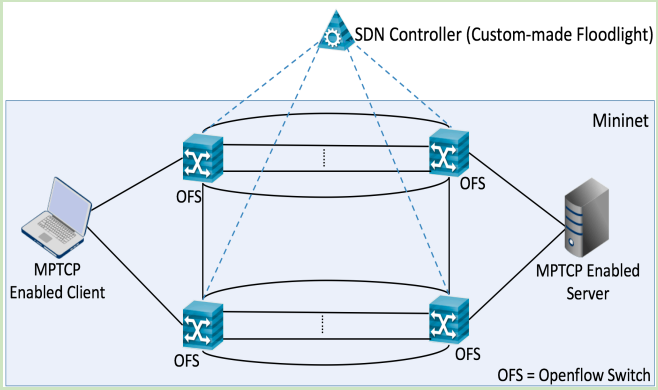
Our Approach: M-fullmesh Path Manager

- Combines characteristics from the fullmesh and ndiffports path managers
  →Utilizes all pairs of Source/Destination IP addresses (akin to fullmesh)
  →Provides the flexibility to change arbitrarily the number of subflows per pair of Source/Destination IP addresses (akin to ndiffports)
- We have implemented M-fullmesh in the MPTCP Linux Kernel

Unleashing the M-fullmesh potential

- Network topology must support multiple paths between each pair of Source/Destination IP addresses
- The routing mechanism should support the distribution of MPTCP subflows through different paths
  →Usage of Software Defined Networks paradigm
- Long-lived MPTCP connections

Evaluation Setup



Usage of Mininet
- Emulates an SDN environment
- Used OpenvSwitch to create a network with Openflow switches

Usage of a modified Floodlight SDN Controller

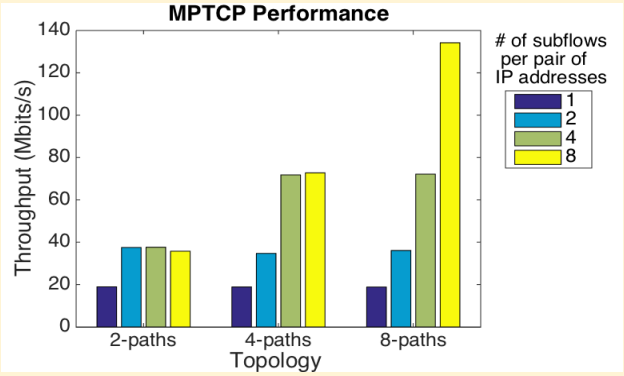
- Enables the distribution of subflows through different paths
  → Layer-4 routing
  → Round-robin selection of available routes
- The first packet of each subflow is redirected to the controller
- The matching rules, which are applied to the Openflow switches, contain the TCP port numbers for each subflow

Other setup settings

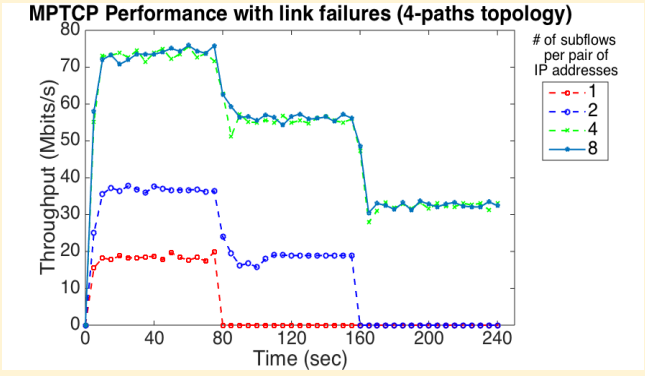
- The capacity of the links connecting the hosts to the core network is set sufficiently high so that these links are not bottlenecks
  →Bundle multiple paths existing in the core of the network

Results

Performance
- Measured MPTCP throughput of long-lived connections
- Topology with 2,4 and 8 paths between each pair of Source/ Destination IP Addresses



Robustness
- Measured MPTCP performance under multiple link failures
- Topology with 4 paths between each pair of Source/Destination IP addresses



*Link failures occur at 80 and 160 seconds