Performance Evaluation of Simple Multipath OpenFlow Controller in a Ceph Distributed Storage System Environment

⚓ Introduction

Chawanat Nakasan, Kohei Ichikawa, Hajimu Iida

Many distributed systems require a large amount of network resources to function. To meet these demands, various measures were developed to use multiple network paths at the same time, known as *multipathing*.

Multipath TCP (MPTCP) stands out as a high-performance extension to TCP. *Simple Multipath OpenFlow Controller (smoc)*, designed to split MPTCP traffic into as many disjoint paths as possible and take advantage of interconnected topologies (right), was developed to recognize and route MPTCP traffic based on their logical grouping.

Network topologies in this work:
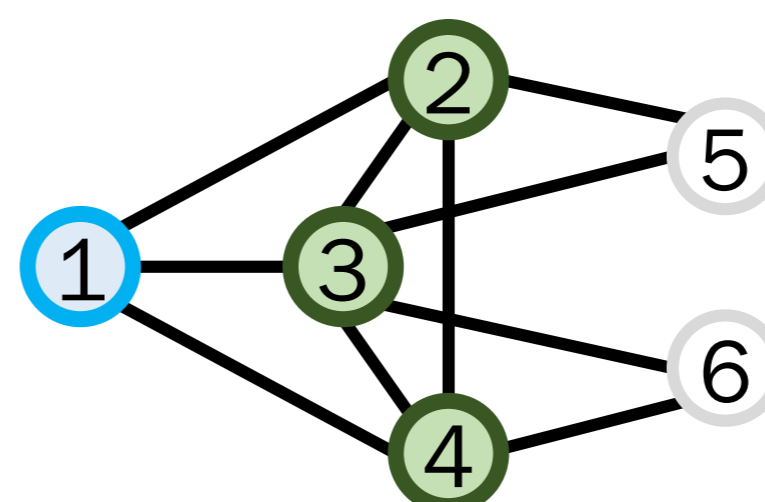
Metadata & Monitoring | Object Storage Device/Servers | Clients

Ceph distributed storage system installation with a metadata server, storage nodes, and clients. [DSS-TESTBED]

PRAGMA EXPERIMENTAL NETWORK TESTBED (PRAGMA-ENT)
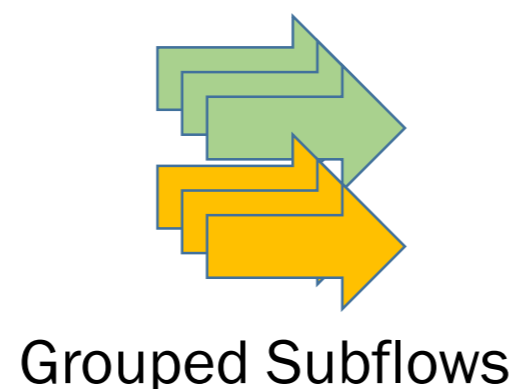*A Resource for the PRAGMA Community*
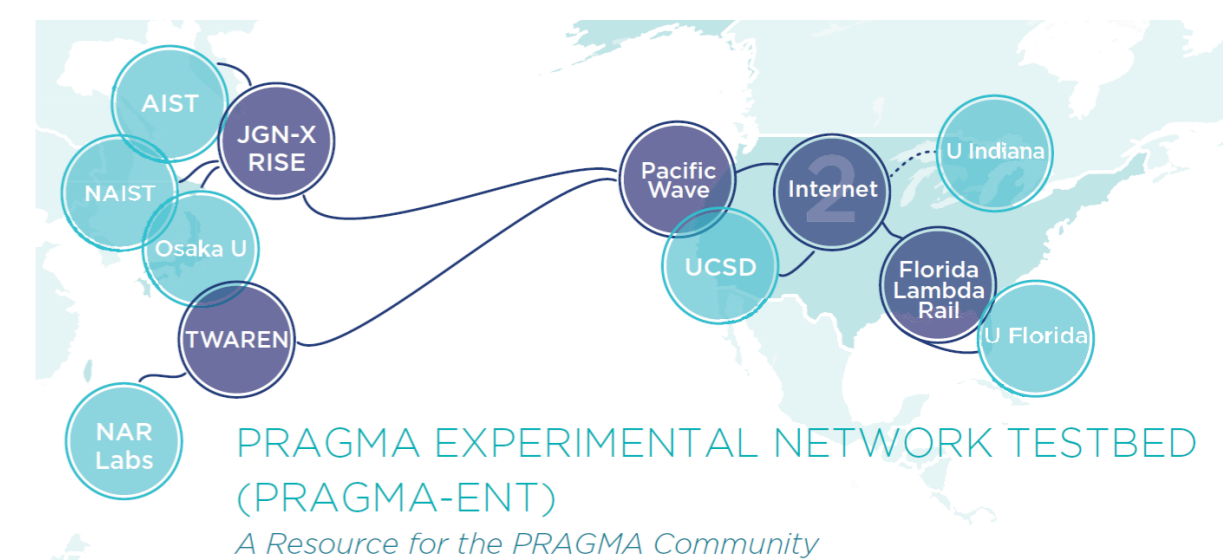
Wide-area network testbed operated by Pacific Rim Applications and Grid Middleware Assembly (PRAGMA) featuring large interconnected networks.
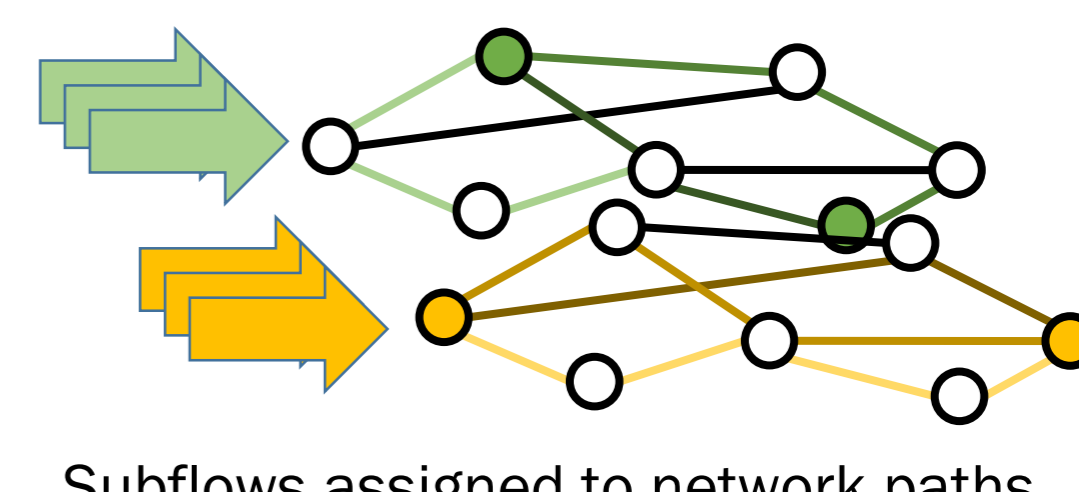[PRAGMA-ENT]

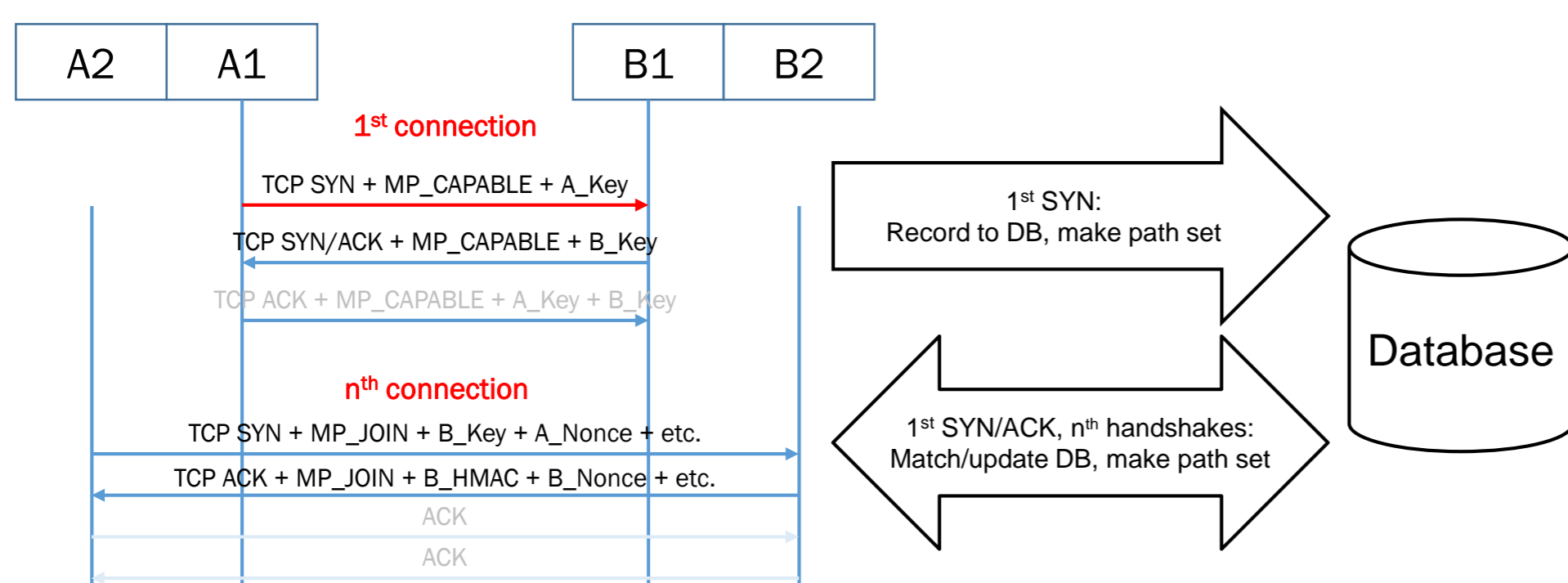## ⚓ Two steps for efficient multipath routing
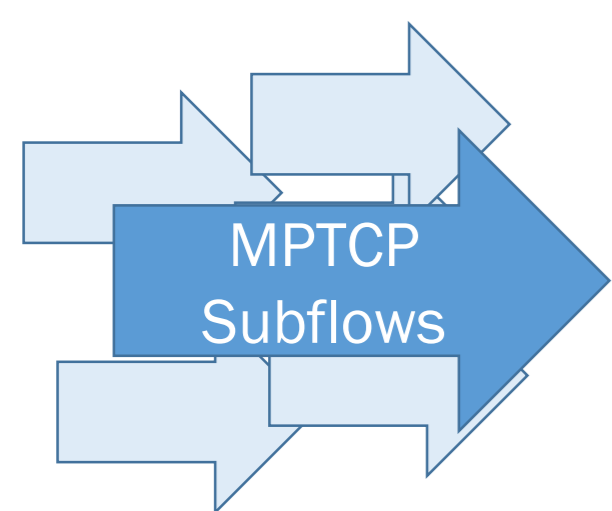
MPTCP Subflows

Step 1: Identify Subflow Groups

Grouped Subflows

**Step 2:** Generate Path Sets

Subflows assigned to network paths

The controller doesn't know which MPTCP connection each subflow belongs to. First 2 packets of each subflow are read to figure out this relationship.

| A2 | A1 | | B1 | B2 |

1st connection

TCP SYN + MP_CAPABLE + A_Key
TCP SYN/ACK + MP_CAPABLE + B_Key
TCP ACK + MP_CAPABLE + A_Key + B_Key

1st SYN: Record to DB, make path set

nth connection

TCP SYN + MP_JOIN + B_Key + A_Nonce + etc.
TCP ACK + MP_JOIN + B_HMAC + B_Nonce + etc.
ACK
ACK

1st SYN/ACK, nth handshakes: Match/update DB, make path set

Database

Visual representation of MPTCP handshake process and how data is handled

To **generate a path set: Primary Path (PP)** is the shortest path. **Alternative Paths (APs)** = all cycle-free paths (excluding PP) sorted ascending by edges shared with the PP then path cost. **Path set =** [PP] + APs. Bandwidth guaranteed to be equal or better than single-path routing.

Host A | Primary Path (2 hops) | Host B

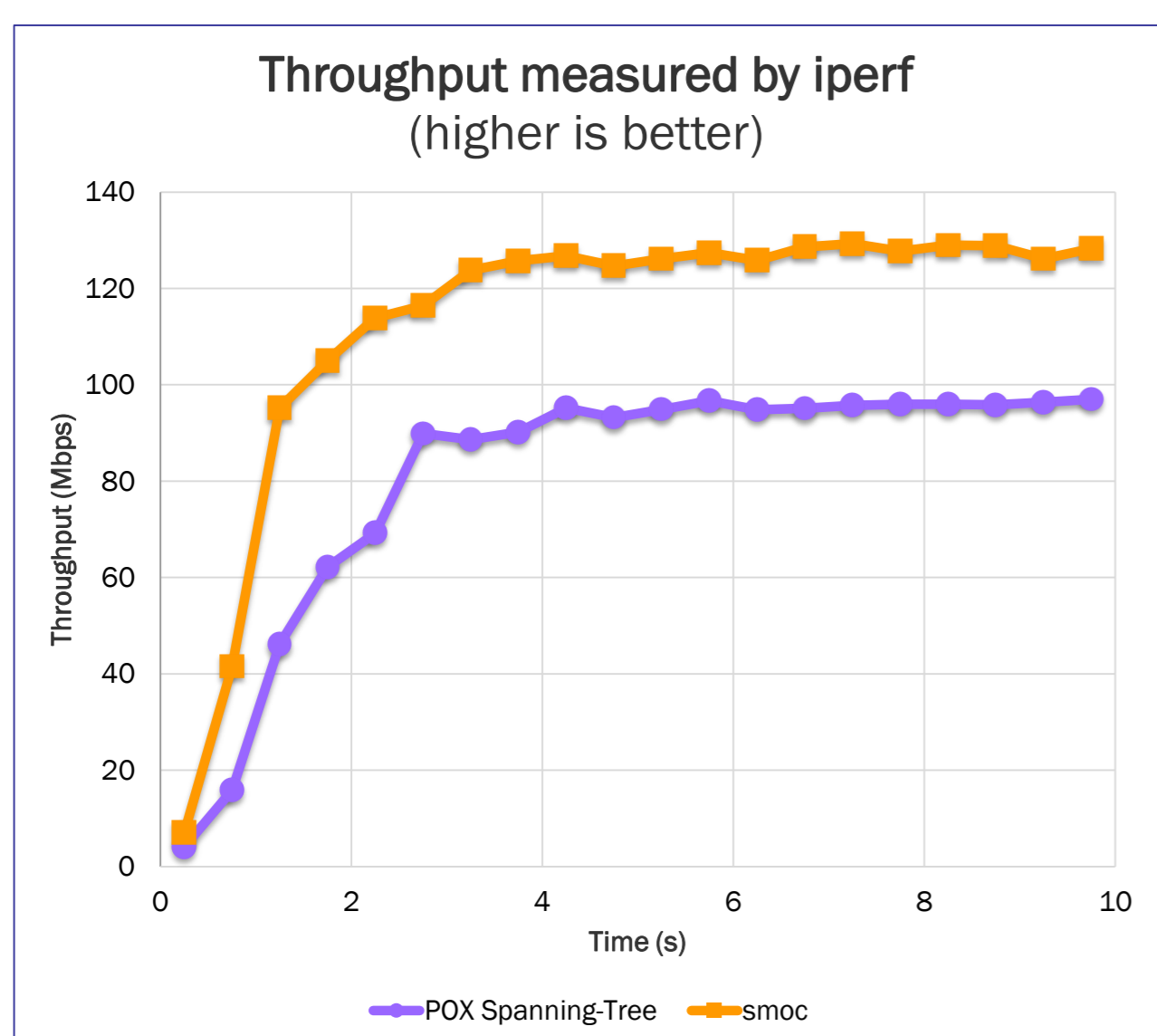AltPath #2 (1 share, length 3)
AltPath #1 (0 shares, length 3)
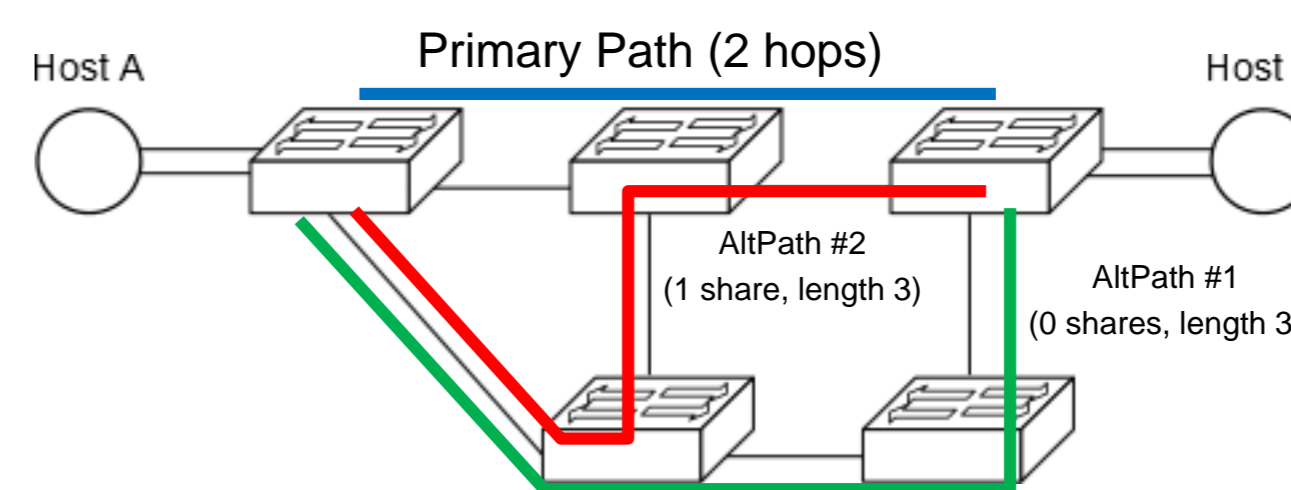
Partial example of algorithm results

smoc was implemented based on this method using POX Framework. It was tested in PRAGMA-ENT and a VM-based Ceph installation at Nara Institute of Science and Technology [DSS-TESTBED].
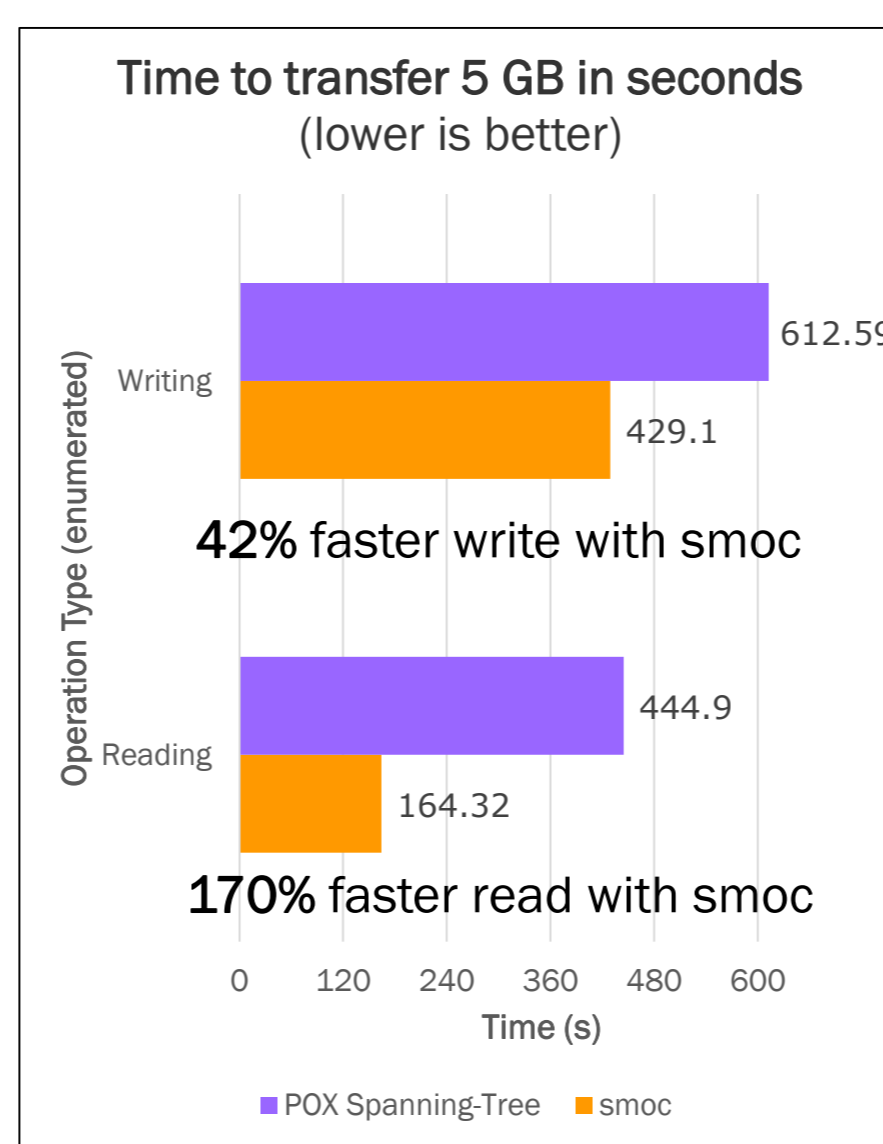
⚓ Evaluation Results

For large files, smoc (with MPTCP) provides a better throughput compared to the standard POX Spanning-Tree Controller.

Wide-area Network Performance [PRAGMA-ENT]

Throughput measured by iperf (higher is better)

Throughput (Mbps)

POX Spanning-Tree — smoc

Time (s)

Distributed Storage Performance [DSS-TESTBED]

Time to transfer 5 GB in seconds (lower is better)

Operation Type (enumerated)

Writing: 612.59 / 429.1
42% faster write with smoc

Reading: 444.9 / 164.32
170% faster read with smoc

POX Spanning-Tree | smoc

smoc provides **better throughput** compared to POX, and also shows **flexibility and response to changes** in network conditions.

POX | smoc

Write

Read