Multipath Congestion Control for Shared Bottleneck

Michio Honda (Keio University)

Yoshifumi Nishida (Keio University)

Lars Eggert (Nokia Research Center)

Pasi Sarolahti (Nokia Research Center)

Hideyuki Tokuda (Keio University)

21.May

PFLDNeT 2009 - Akihabara, Tokyo

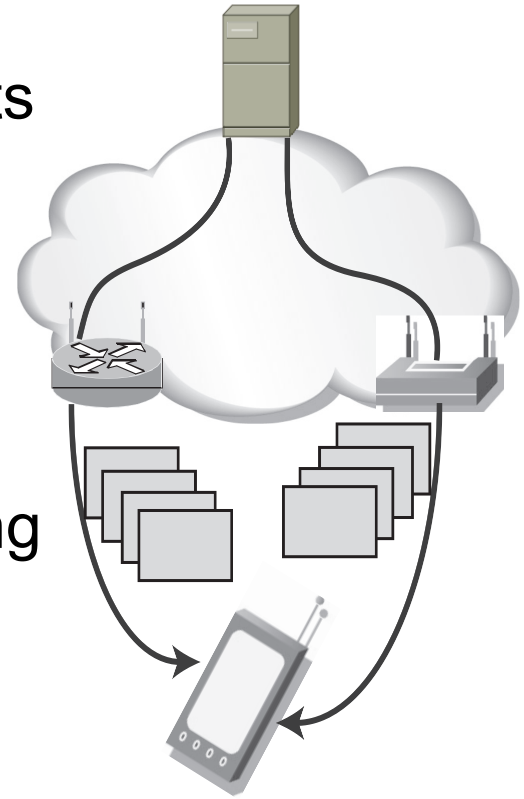
Keio University
1858
CALAMVS
GLADIO
FORTIOR

Outline

- Introduction
- Problem statements for multipath congestion control
- Approach
- Designing Multipath Congestion Control
- Experimental results
- Conclusion and ongoing work

Introduction

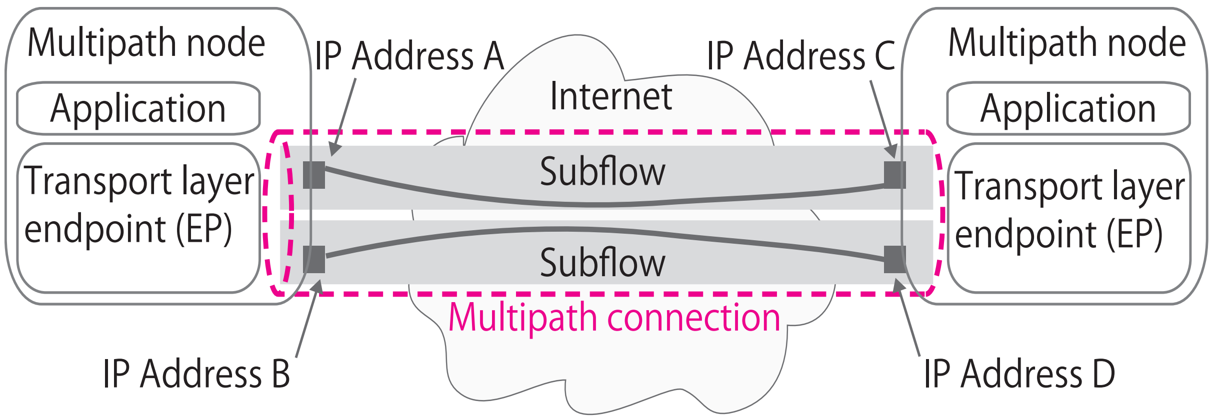
- Multiple paths between end-to-end hosts
  - Many hosts are equipped with multiple network interfaces
- Transmitting data over multiple paths
  - Increase resource allocation with improved reliability and load balancing

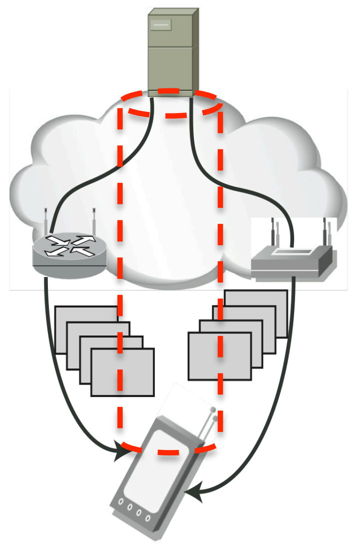


Keio University
CALAMVS GLADIO FORTIOR
1858

Multipath Transport Protocols

- Multipath connection
  - An entity over which applications communicate between transport layer endpoints (EP)
  - Provide the same communication primitive through the socket as well as general transport protocols (i.e., a reliable and ordered byte stream)
- Subflow
  - An entity over which the endpoint transmits a flow along a path

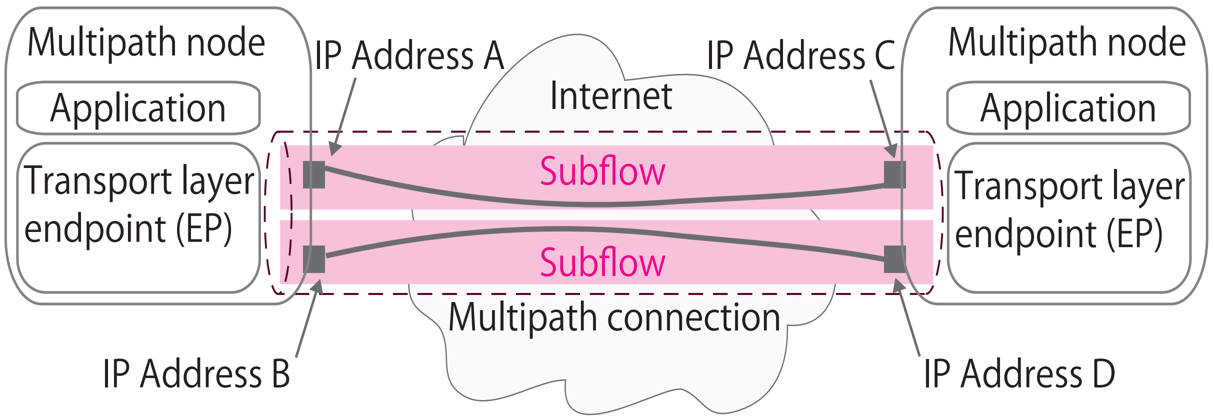


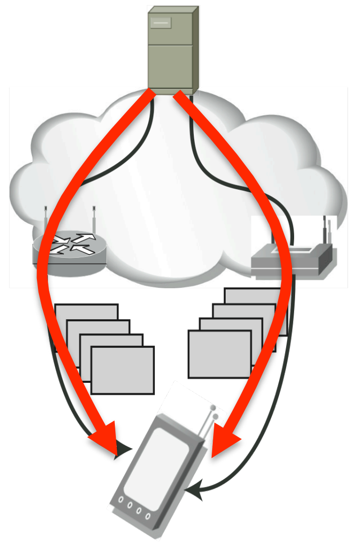


Keio University

Multipath Transport Protocols

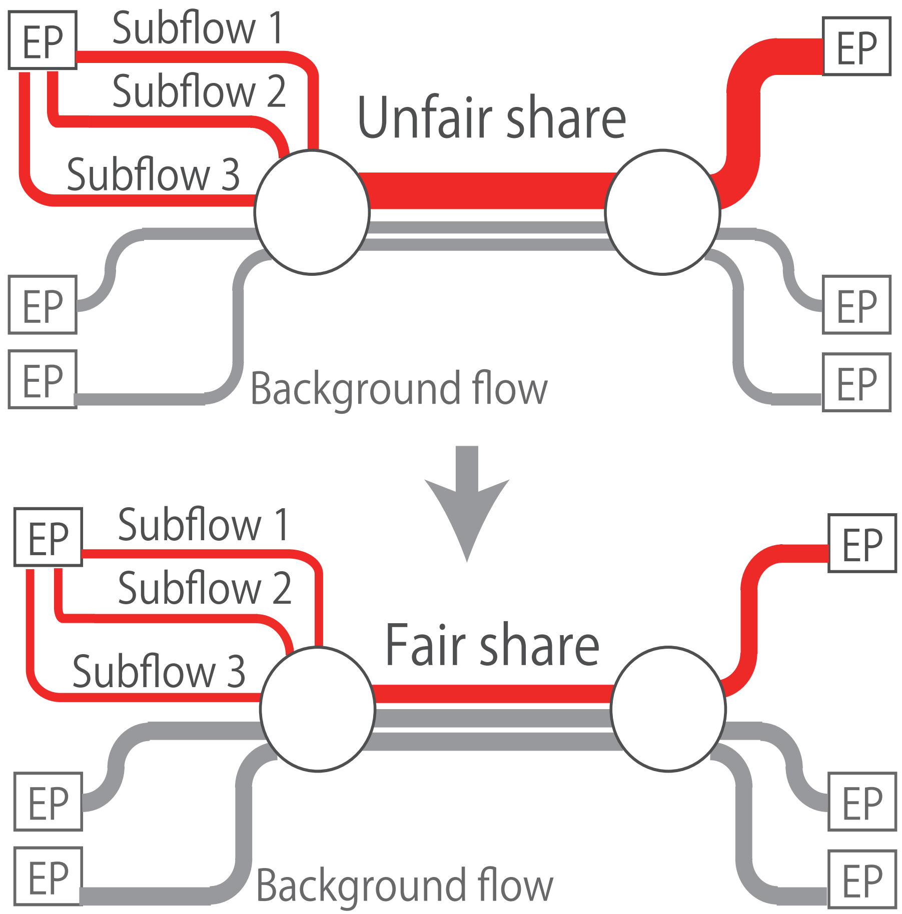
- Multipath connection
  - An entity over which applications communicate between transport layer endpoints (EP)
  - Provide the same communication primitive through the socket as well as general transport protocols (i.e., a reliable and ordered byte stream)
- Subflow
  - An entity over which the endpoint transmits a flow along a path

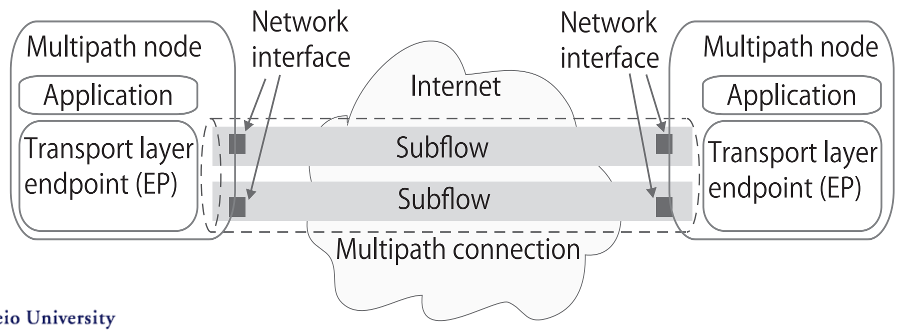




Problem Statement

- Existing multipath transport protocols adopt TCP's algorithm to each subflow (e.g., pTCP, mTCP, CMT)
- The endpoint of the multipath connection uses the shared bottleneck unfairly





Keio University

Approaching fair utilization of the shared bottleneck

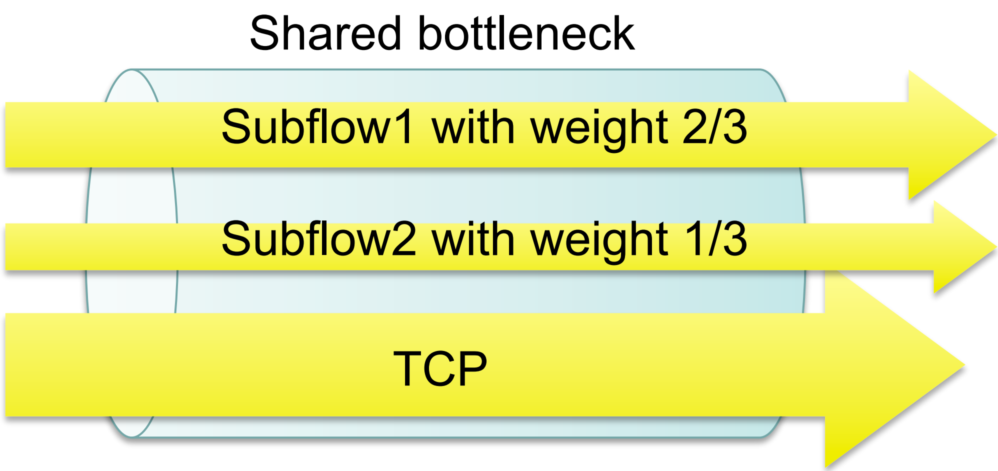
- How do we achieve TCP-friendly multipath connections?
- Aggregate congestion control approach (e.g., E-TCP, CM)
  - Share the congestion information between subflows
    - Don't work between subflows along different paths
    - Cause performance issue
- Shared bottleneck detection approach (e.g., mTCP)
  - Take time to detect shared bottleneck
- <u>Weighted congestion control approach</u>
  - Apply the weight to congestion control of subflows
    - Each subflow independently behaves based on its own congestion information (i.e., cwnd, RTT measurement)
    - Work even if each subflow traverses distinct paths

Keio University

CALAMVS GLADIO FORTIOR 1858

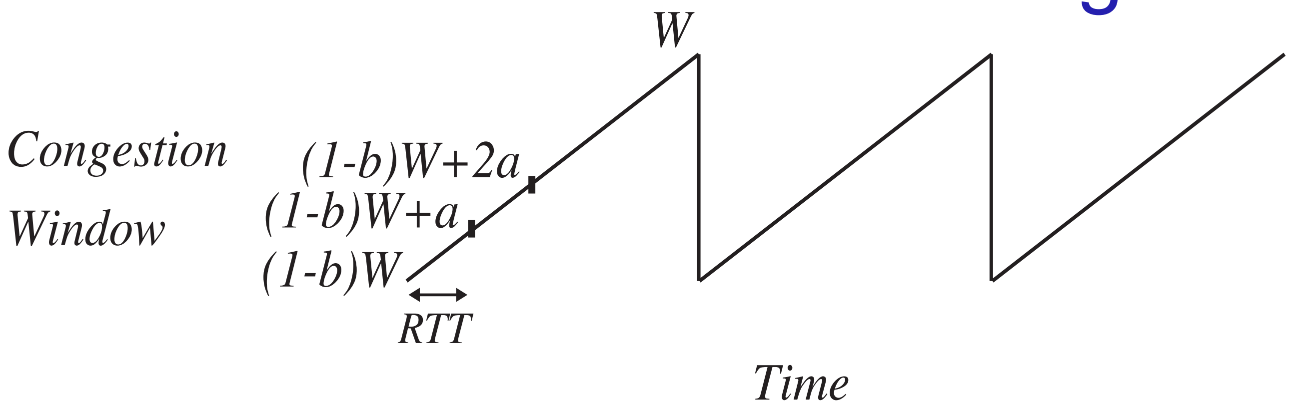
Approaching fair utilization of the shared bottleneck

- The sum of the throughput of subflows should be equal with TCP at the shared bottleneck
- We define the weight of TCP is 1, so maintain the sum of weight of subflows to 1 in the multipath connection
  - One subflow with the weight *D* achieves *D* times throughput TCP



Keio University

Applying the AIMD parameters for each subflow based on the weight



Window size of *AIMD(a, b)*

- Based on the weight of the subflow (*D*), we determine its AIMD parameter (additive increase parameter "*a*" and multiple decrease parameter "*b*")

$$a = \frac{3b}{2-b} D^2$$

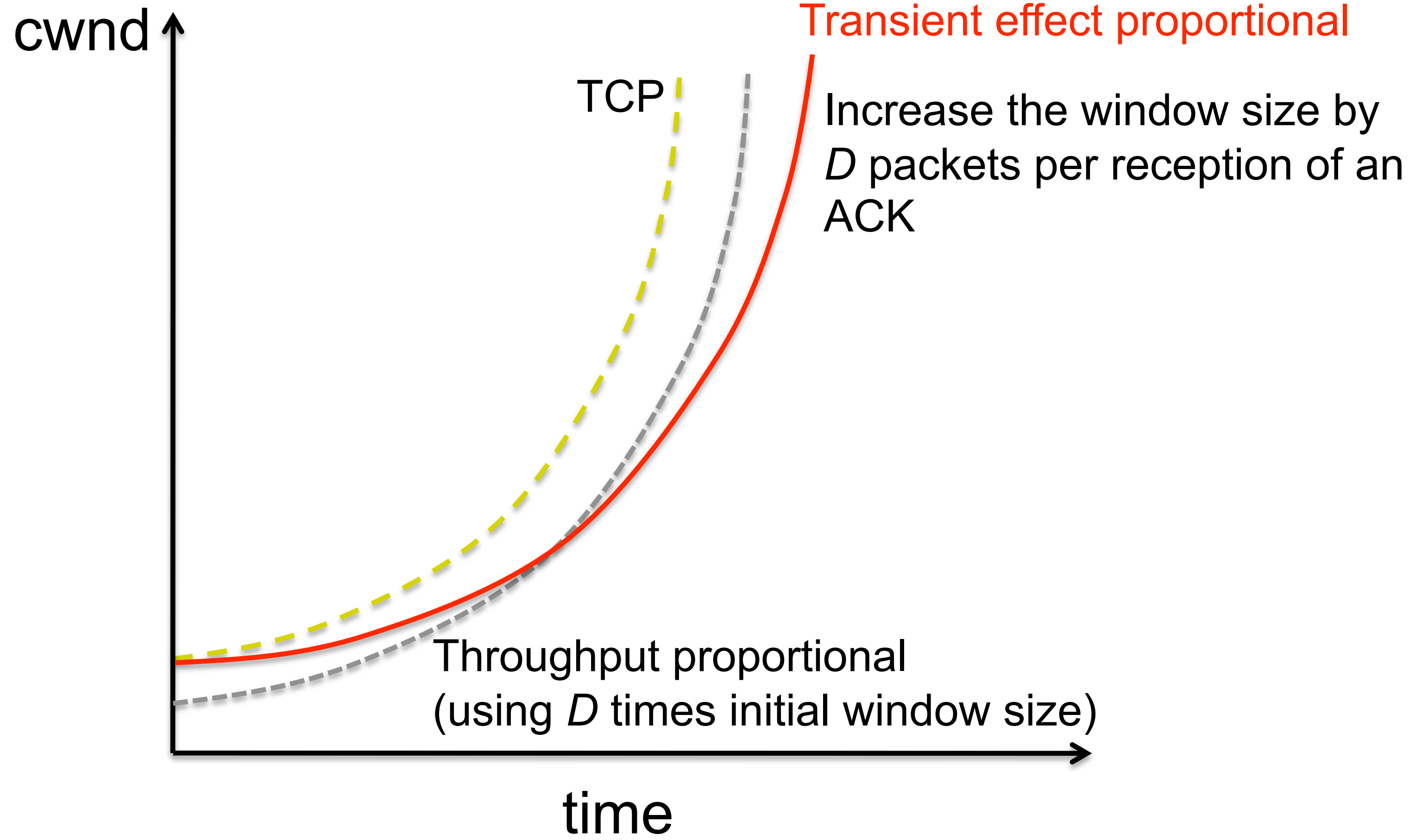
- We adopt AIMD(D^2, 1/2) for *D* times throughput compared to TCP (using AIMD(1, 1/2))
  - based on the response function and simulation results (MulTCP and PA-MulTCP cannot fit $D<1$)

Keio University

CALAMVS GLADIO FORTIOR 1858

Slow-start behavior of subflows

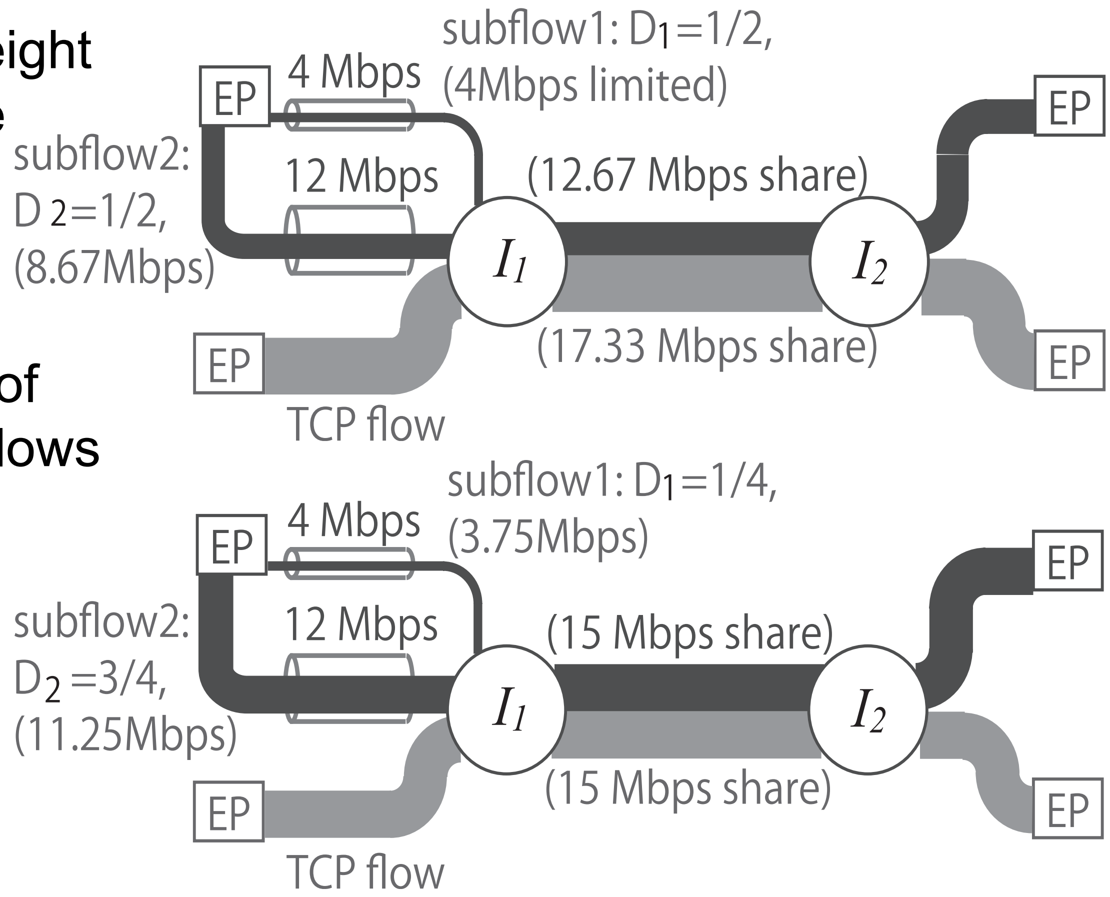
- We use conservative increase behavior with the same window size of TCP at the beginning of the transmission and after RTO



Keio University
CALAMVS GLADIO FORTIOR 1858

How do we use spare bandwidth of disjoint links?

- Disjoint links can have different spare bandwidth
- We have to adjust the weight of subflows to bypass the limitation of spare bandwidth
- Detect spare bandwidth limitation by comparison of throughput between subflows



Keio University
CALAMVS GLADIO FORTIOR 1858

Detection of spare bandwidth limitation

- Comparison of each subflow based on the value which has deducted the effect of the weight and RTT

$$T_{wr} = \frac{RTT}{weight} T_{measured}$$

- We reduce the weight of the subflow with the smallest *Twr*
  - At the same time increase the weight of the highest *Twr*
- We change the weight of subflow with more outstanding weight more conservatively

$$D^{dec}_{new} = (D^{dec}_{cur})^2$$

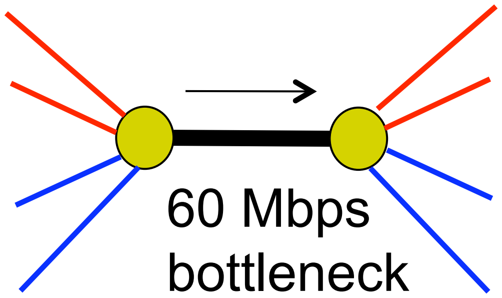
  - Maintain aggressiveness of subflows achieving better throughput

Keio University

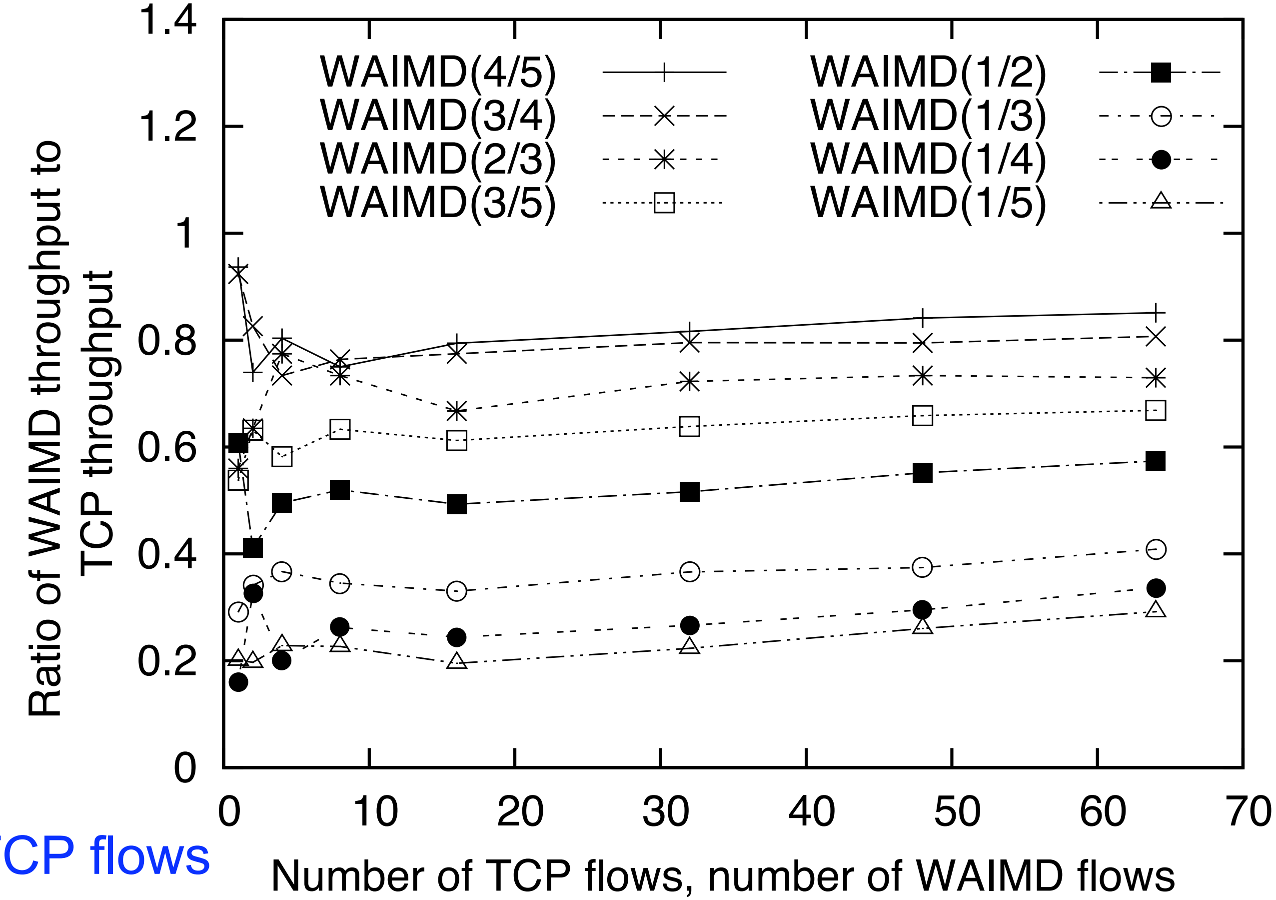
CALAMVS GLADIO FORTIOR 1858

Experimental results (Weighted AIMD flows v.s. TCP flows)

- Throughput proportion of weighted AIMD (weight < 1) flows compared to TCP



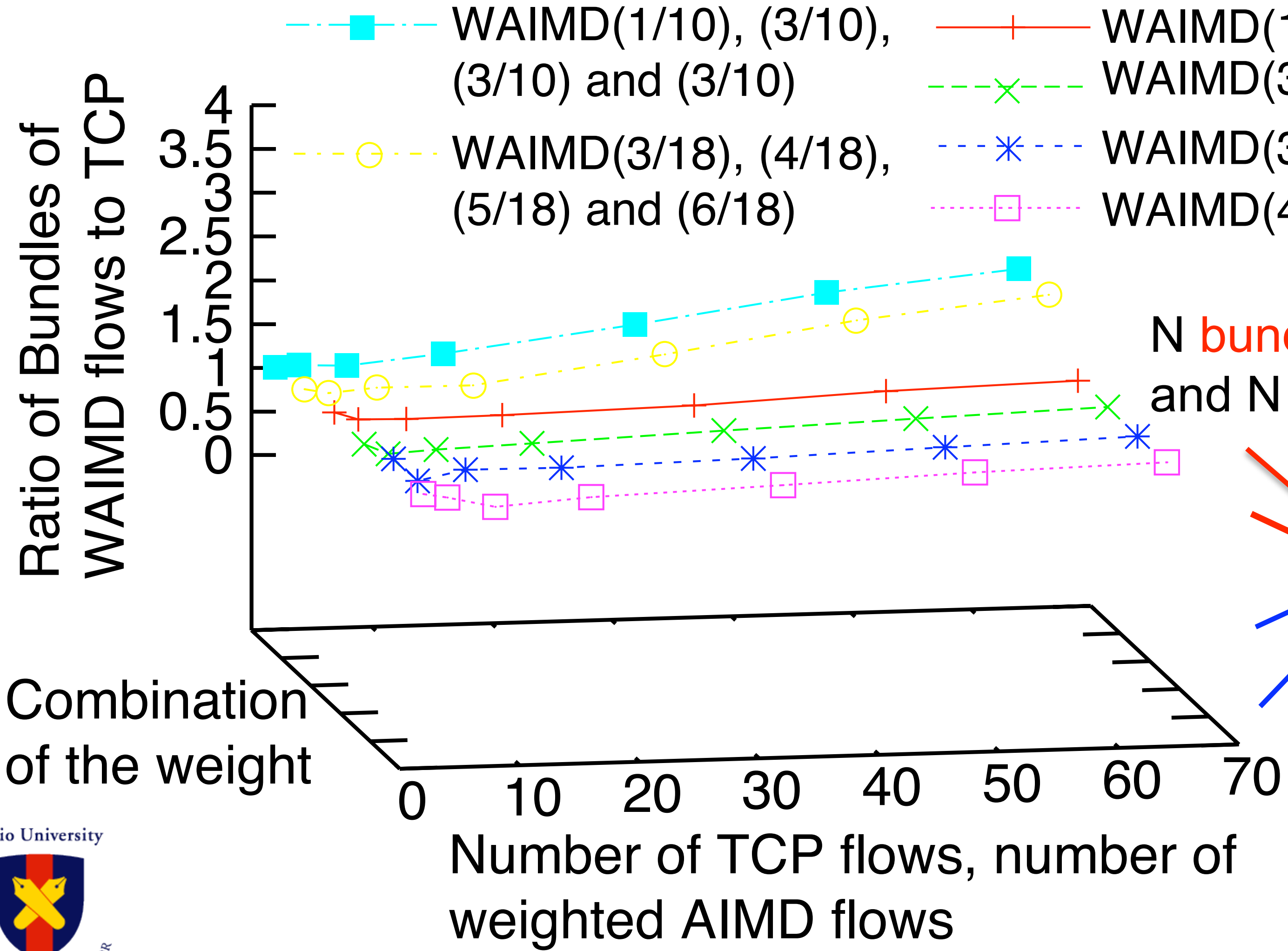
N WAIMD flows
and N standard TCP flows



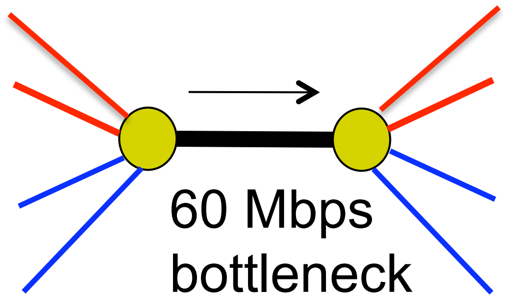
Keio University

Experimental results (Bundles of WAIMD flows v.s. TCP flows)

- Comparison between aggregate of WAIMD flows and TCP

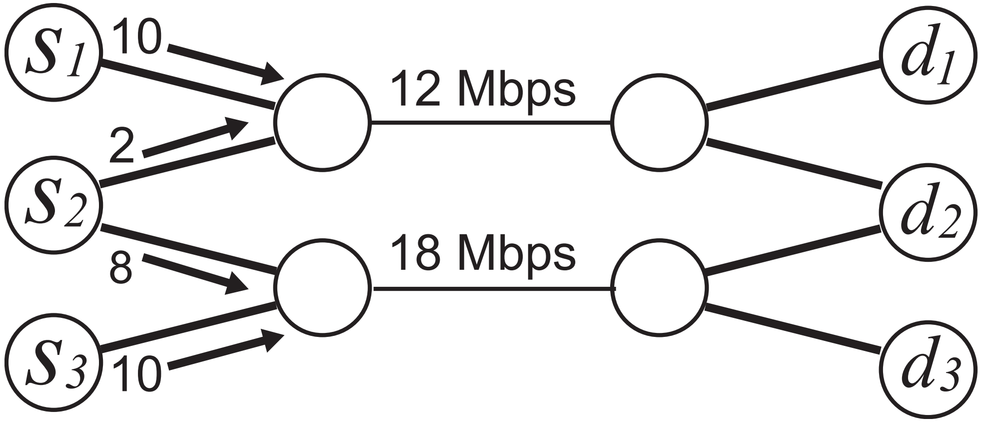


N bundles of WAIMD flows and N standard TCP flows



Keio University
CALAMVS GLADIO FORTIOR 1858

Behavior on disjoint bottlenecks



- Our algorithm converges to equal resource allocation between endpoints across bottlenecks, similarly to Kelly's and Key's resource pooling (but equal window allocation)
  - Discussion: Should we achieve an equal resource allocation for per-flow fairness? or per-connection?

Keio University

CALAMVS GLADIO FORTIOR 1858

Conclusion and Ongoing work

- Conclusion
  - Our scheme achieves TCP-friendliness of multipath communication for coexistence of TCP and multipath transport protocols
    - Weighted congestion control approach
  - We find out that our scheme achieves TCP friendliness of the bundle of multiple subflows through experiments
- Ongoing work
  - Evaluation and optimization of convergence speed and stability
  - Investigation for the other fairness metric (e.g., proportional fairness, cost fairness)

Keio University

CALAMVS GLADIO FORTIOR 1858