Multipath TCP: Challenges & Opportunities

Matthieu Coudron

13 Feb. 2018, IIJ

Internet Initiative Japan

IIJ INNOVATION INSTITUTE

Outline

I. Multipath communications
   - Advantages
   - Challenges
II. Multipath TCP presentation
III. Augmented multipath TCP
IV. MPTCPNUMERICS: MPTCP window management
V. Conclusion

Multipath communications

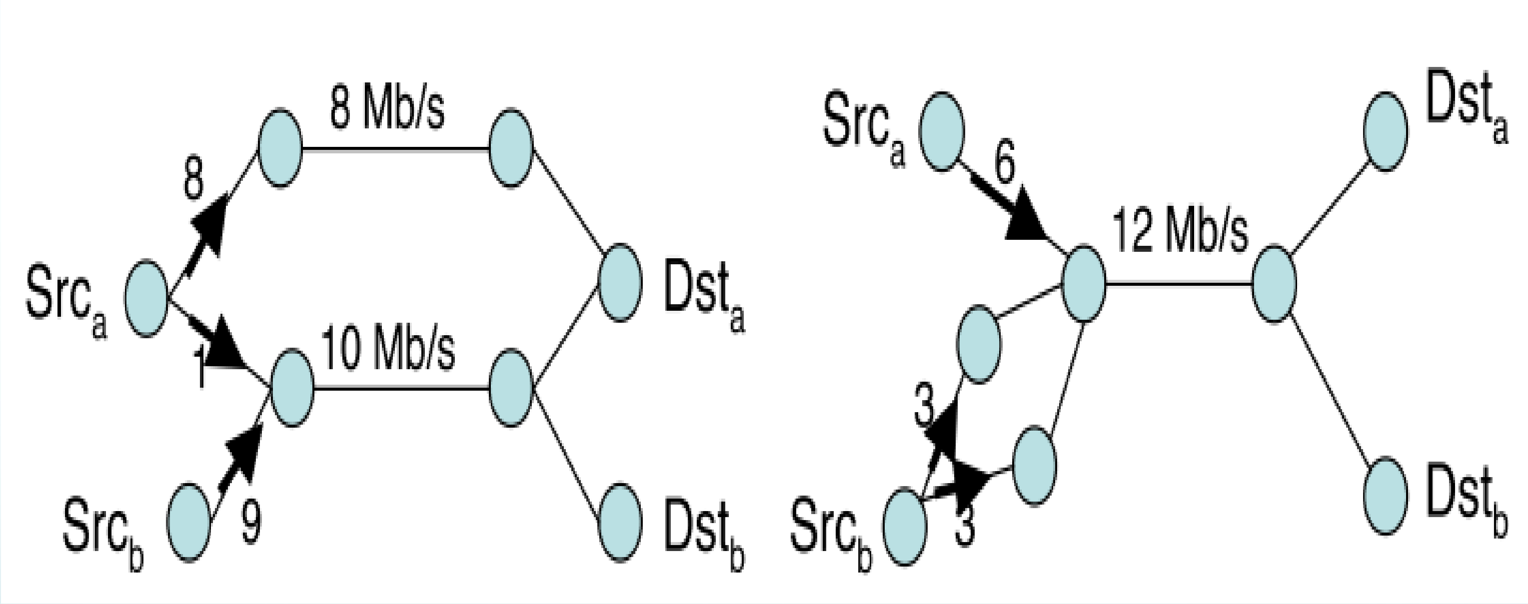
1. Motivations
2. Challenges

Motivations

- **Reliability**
  easy to retransmit on alternative paths
- **Fairness & Resource Pooling**
  make use of previously unused resources
- **Bandwidth aggregation**
  sum links' throughput
- **Confidentiality**
  harder to capture on several paths

Resource pooling

« Resource pooling means making a collection of networked resources behave as though they make up a single pooled resource »
D. Wishik et al., *The resource pooling principle*, CCR 2018



Multipath Challenges

- ✓ **Deployment concerns**
  - Deployment has to be incremental
  - Compatible with the existing infrastructure
- ✗ **Path management**
  - How many (disjoint) paths between hosts ?
- ✗ **Packet reordering**
  - How to deal with heterogeneous paths ?

Multipath TCP

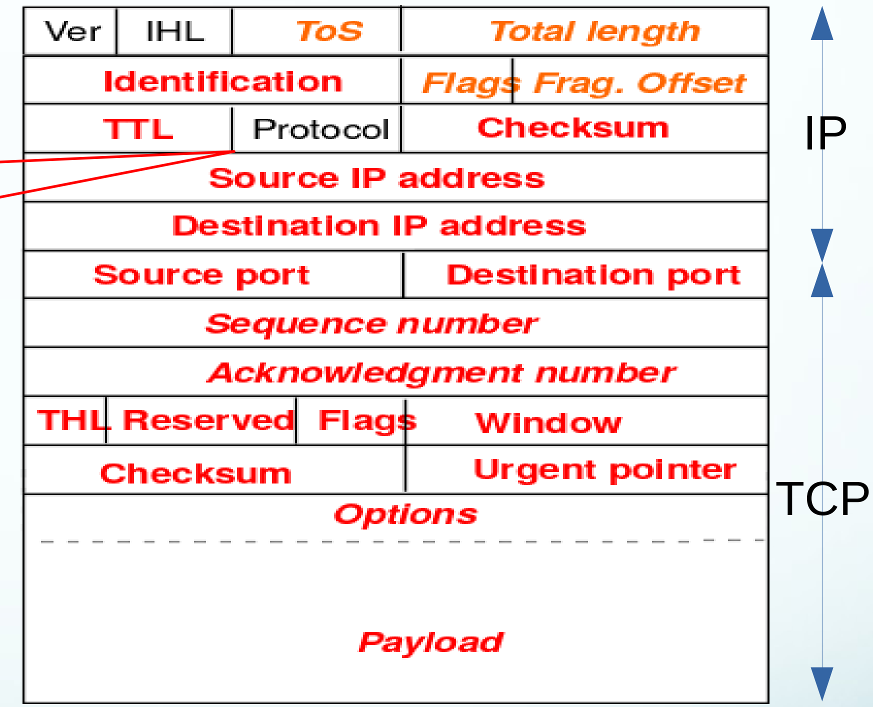
1. An ossified Internet
2. Introduction
3. Subflow management

An ossified Internet

- IP/TCP fields modified by middleboxes

Protocol field intact but some middleboxes drop packets with unknown protocol, i.e., different from TCP or UDP

- Solution(s) for a new protocol
  - Happy Eyeballs: Fallback on a safe protocol
  - Tunneling
  - Look like TCP (or UDP)



Source : O. Bonaventure Cloudnet 2012

MPTCP introduction

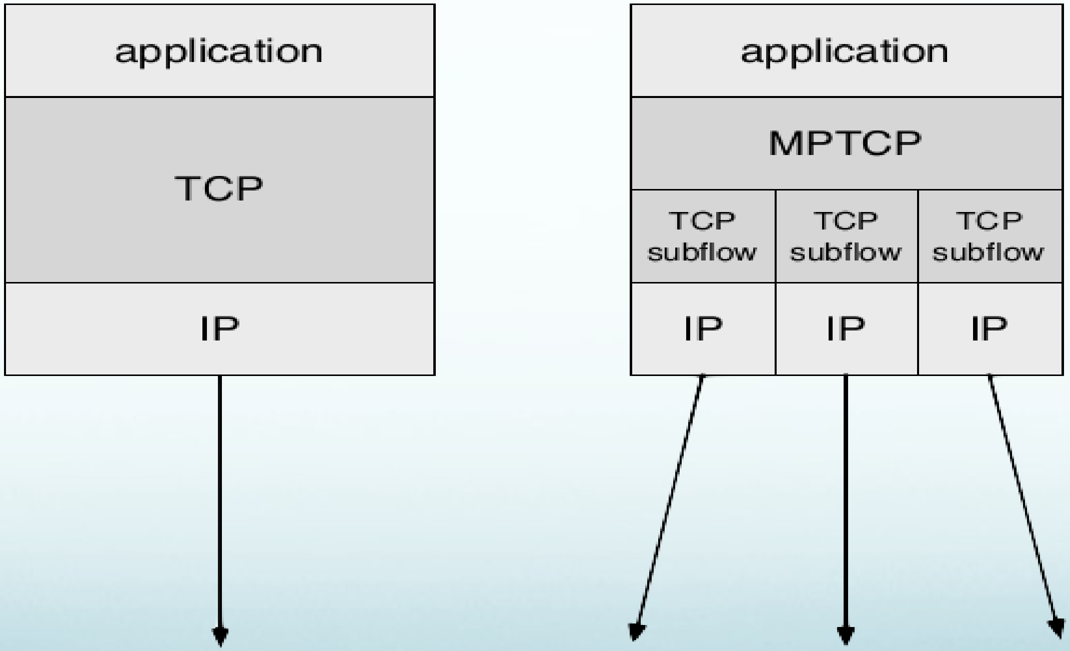
- Defined in RFC 6824 as a TCP extension
  - Emphasis on backwards compatibility
  - Works with most middleboxes
  - Congestion control fair to TCP

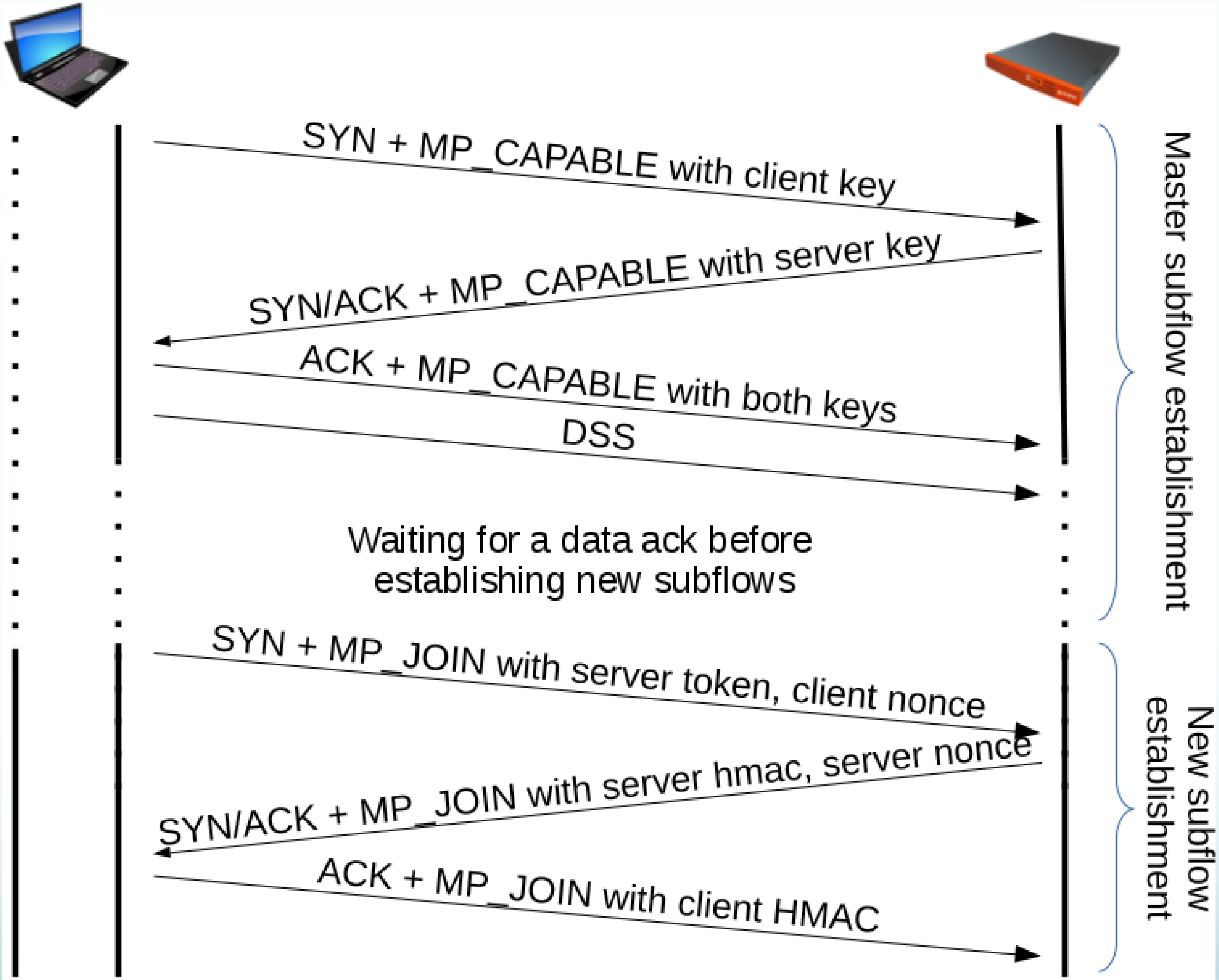
- Can send data **concurrently** on several subflows
  - Single data stream transmitted at **51.8 Gbit/s**.

- Available in:
  - Linux
  - BSD
  - iOS7

MPTCP introduction

1. First acknowledges if destination is MPTCP compliant during the 3 way handshake
2. Then creates additional TCP subflows *according to path management mechanism*

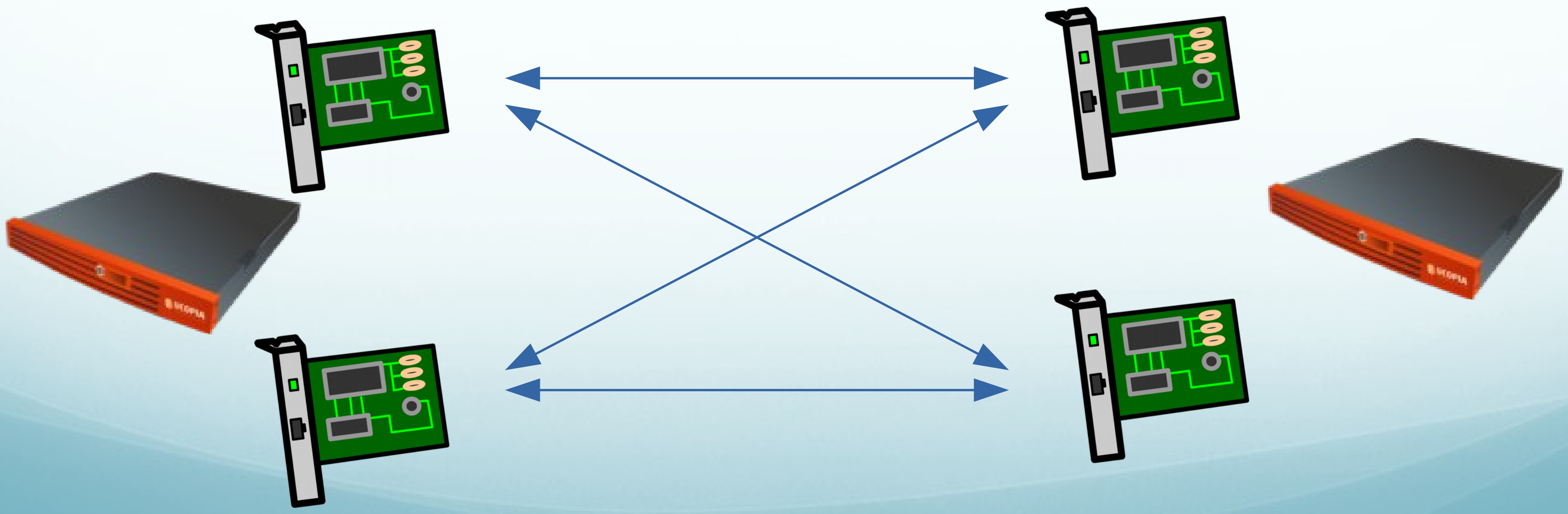




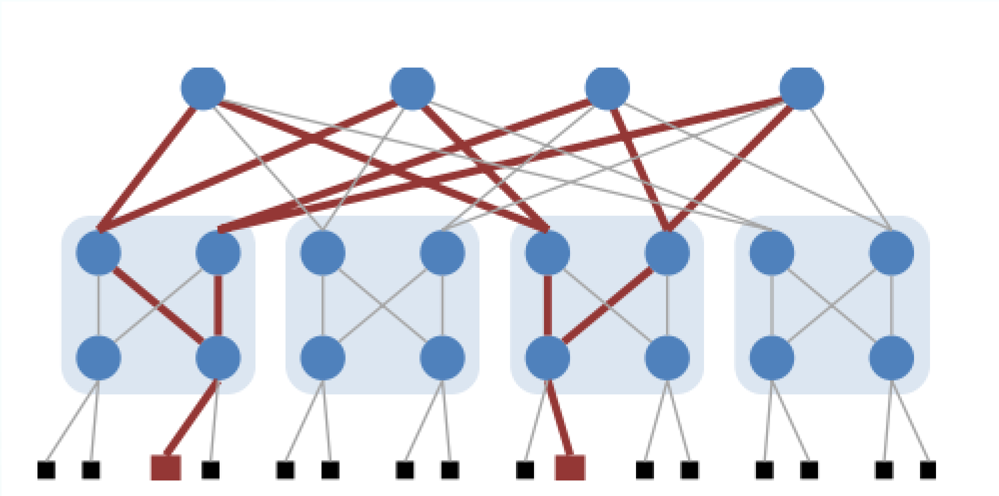
Token = Most significant 32 bits of the hash of the key

MPTCP path management

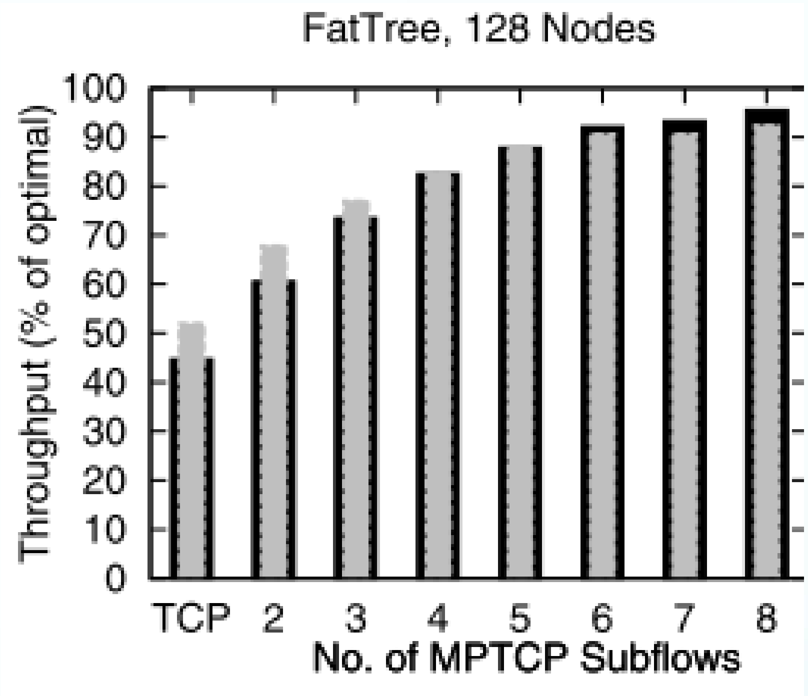
- RFC 6182 states path management should be « modular », i.e., policy-based
- Several subflows can originate from the same IP with different port numbers
- By default in linux 1 subflow per (IP_{source}, $IP_{destination}$)
- Example: 2 source IPs and 2 destination IPs => 2x2 = 4 subflows



Some (encouraging) results



Fat tree with K=4 pods



© Raiciu, et al. "Improving datacenter performance and robustness with multipathTCP", *ACM SIGCOMM* 2011.

Application sends data to MPTCP send buffer :

#MPTCP	Data ↓	Mapping
2	B	2 : 1
1	A	1 : 1

MPTCP scheduler

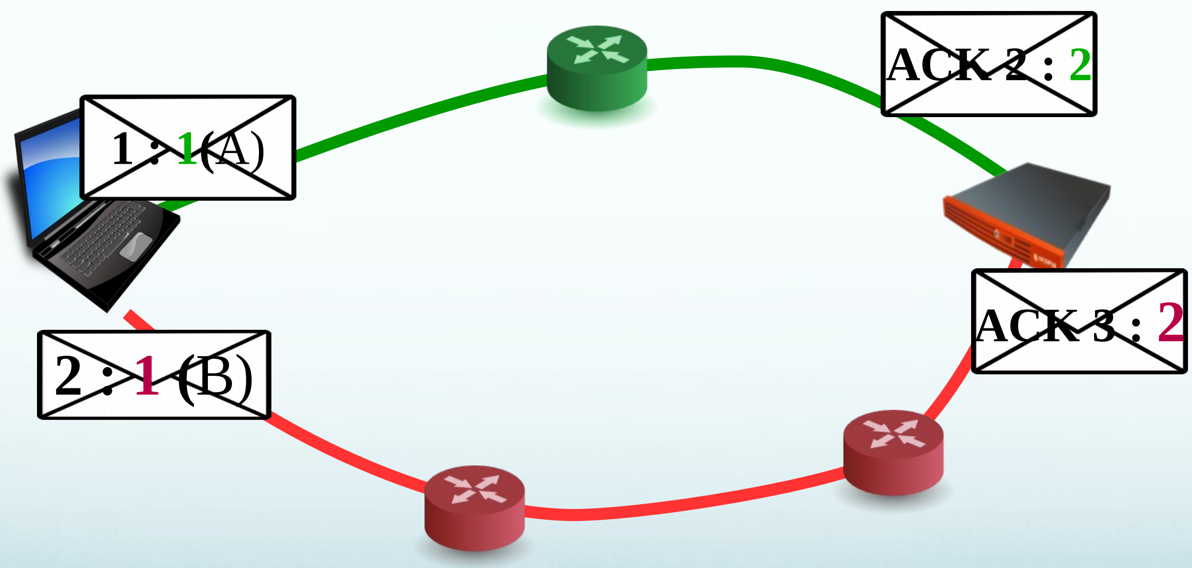
#TCP	DATA
1	A

#TCP	DATA
1	B

« AB » forwarded to application

B
A

MPTCP receive buffer



Application sends data to MPTCP send buffer :

#MPTCP	Data ↓	Mapping
2	B	2 : 1

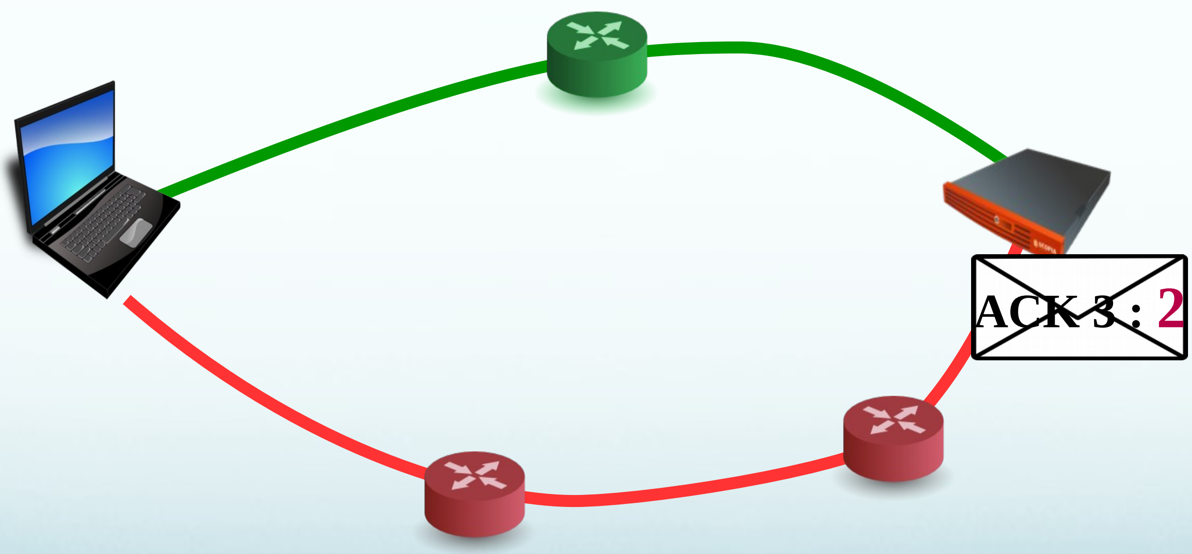
MPTCP scheduler

#TCP	DATA

#TCP	DATA
1	B

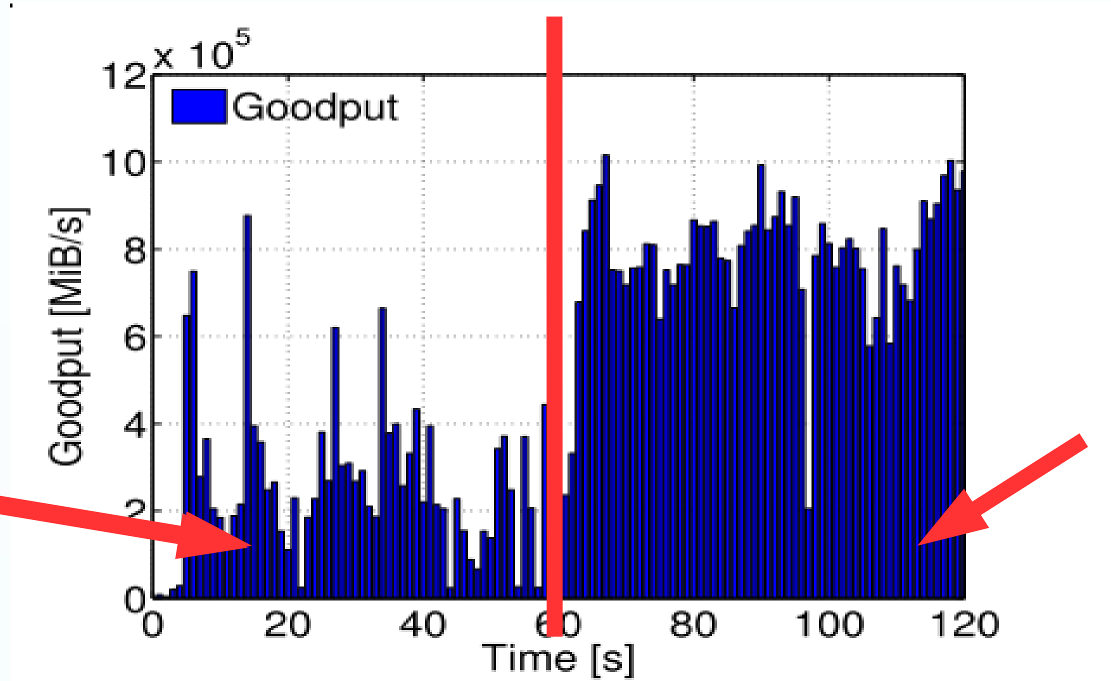
« AB » forwarded to application

MPTCP receive buffer



MPTCP can be worse then TCP

MPTCP with 2 subflows : (2G)



(c) Resulting MPTCP Goodput

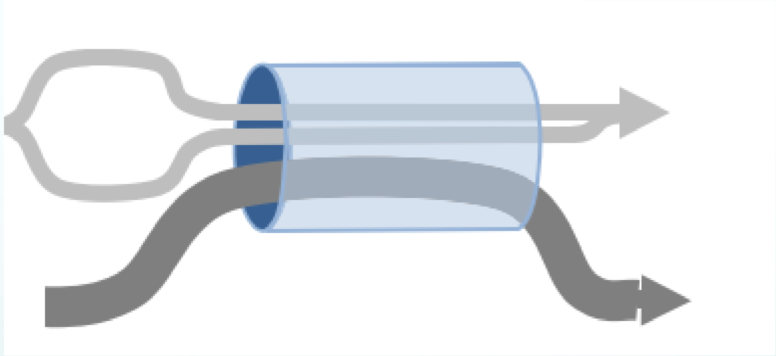
MPTCP with 1 subflow

Ferlin, S. et al. *« Multi-Path Transport over Heterogeneous Wireless Networks: Does it really pay off? »*

Fairness with legacy TCP flows

- A multipath flow should
  1) perform at least as well as a single-path flow would on the best of the paths available to it
  2) not take up any more capacity on any one of its paths than if it was a single path flow using only that route

- MPTCP congestion control kicks in during congestion avoidance
- window is shared between subflows



OLIA : Opportunistic Linked Increase Algorithm

- Losses handled like in TCP (Wr = Wr/2)
- For every ack ACK on flow r, add to w_r

$$\frac{w_r/\mathrm{rtt}_r^2}{(\sum_{p\in\mathcal{R}_u} w_p/\mathrm{rtt}_p)^2} + \frac{\alpha_r}{w_r}$$

- Where α is
  - > 0 if r belongs to best paths with small cwnd
  - < 0 if Wr has a big window while a better path exists with a smaller window
  - = 0 otherwise

Problem with short connections (e.g., <100KB)

- In a wifi/LTE setup, with an initial Wifi subflow, the transfer finishes before the LTE subflow could send data
- Yet LTE accounts for 61 % of the energy consumed

Nikravesh, A. et al. *« An in-depth understanding of multipath TCP on mobile devices »*, Mobicom 2016

Summary

- **Deployment incremental & backwards compatibility ok**
  - increased confidentiality can be seen as a threat
- **Path management**
  - How many (disjoint) paths between hosts ?
  - When to create/reset subflows ?
- **Packet reordering**
  - Increase buffer size but when possible or limit subflow usage

$$\gamma \geq 2 \sum_{i}^{N} BW_i * \max_i RTT_i$$

- **Pareto-optimality**
  - New solutions/protocols should be always better and for everyone

Path management problem: Augmented MPTCP

1. Goal & Overview
2. Presentation of LISP
3. Tesbed & Results

M. Coudron, S. Secci, G. Pujolle *« Cross-layer cooperation to boost multipath TCP performance in cloud networks »*, CloudNet 2013

Contribution overview

Objective

- Increase goodput between endusers and/or DataCenters via *disjoint* physical paths

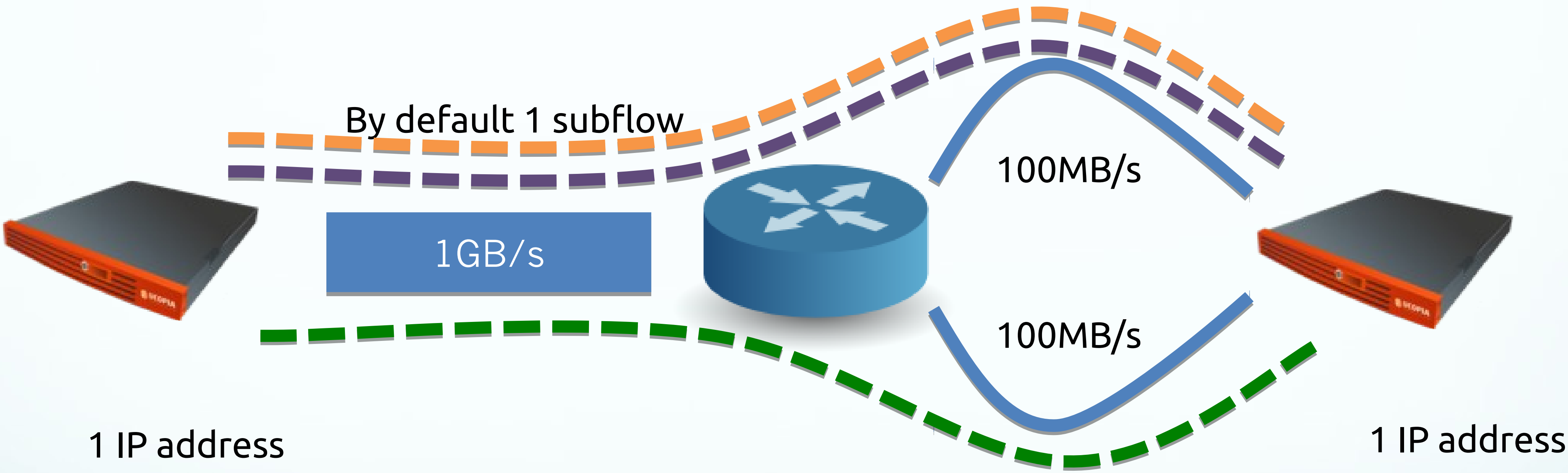
Problem

- Endhosts ignore the topology

Solution

- Ask a protocol that knows the answer ; for instance LISP => crosslayer solution
- (Assuming WAN segment is the bottleneck)

- How many subflows to create ?
- How to achieve proper forwarding ?



Wouldn't 2 subflows be better ?

Not necessarily... need to follow different physical paths

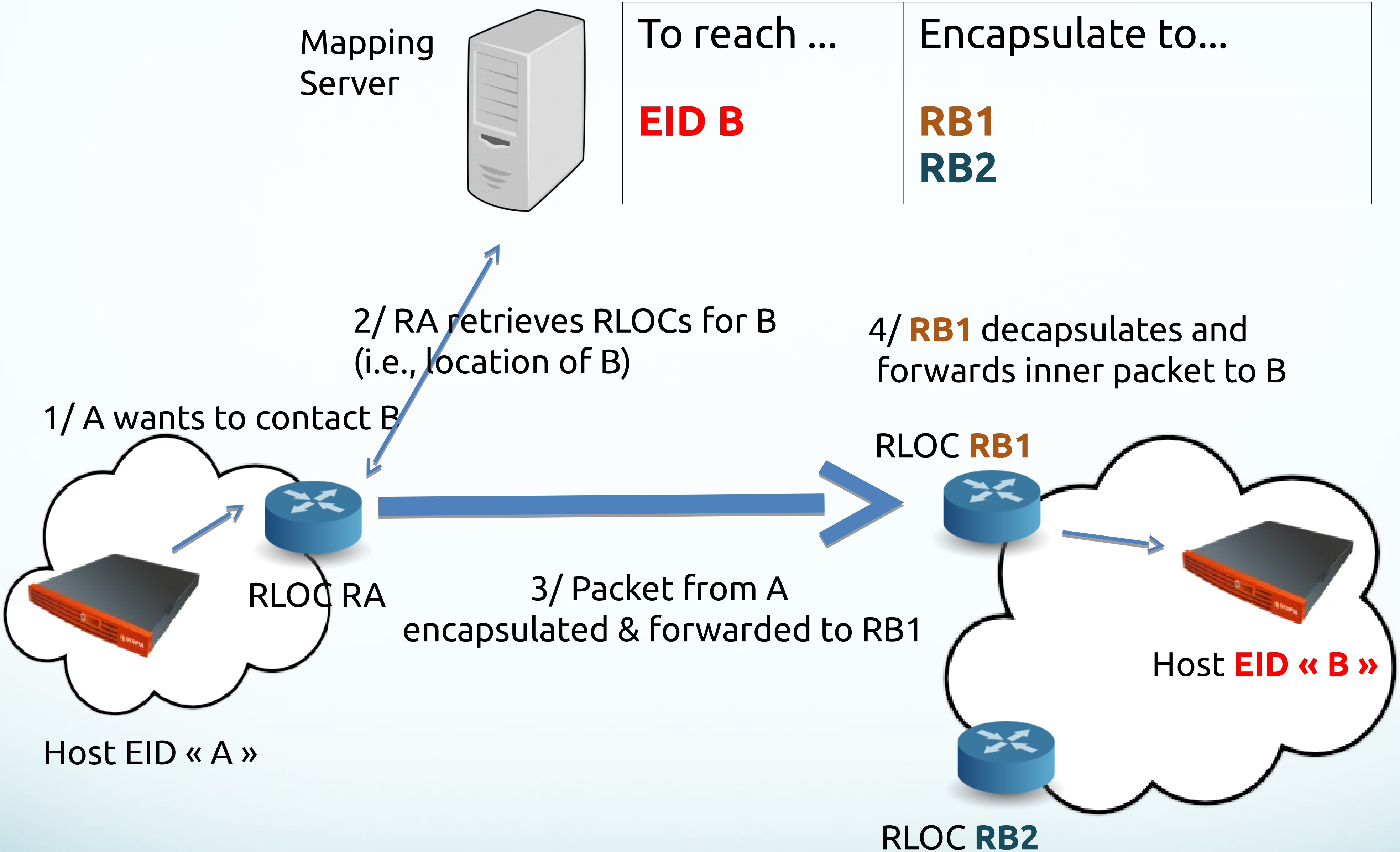
Architecture Overview

- Use LISP protocol to enhance MPTCP
  - LISP can give edge path diversity information
  - LISP can enable multipath WAN forwarding
- Enforce per subflow forwarding (SDN-*like*)
  - Based on TCP ports in our case

Location/Identifier Separation Protocol (LISP)

- Defined in RFC 6830
  - Tunneling protocol between edge routers
  - Allows us to get the WAN path diversity
- IPs classified in two groups
  - **Endpoint IDentifier** (EID)
  - **Routing Locators** (RLOCs)
- EID associated to RLOC(s) via a mapping system control-plane

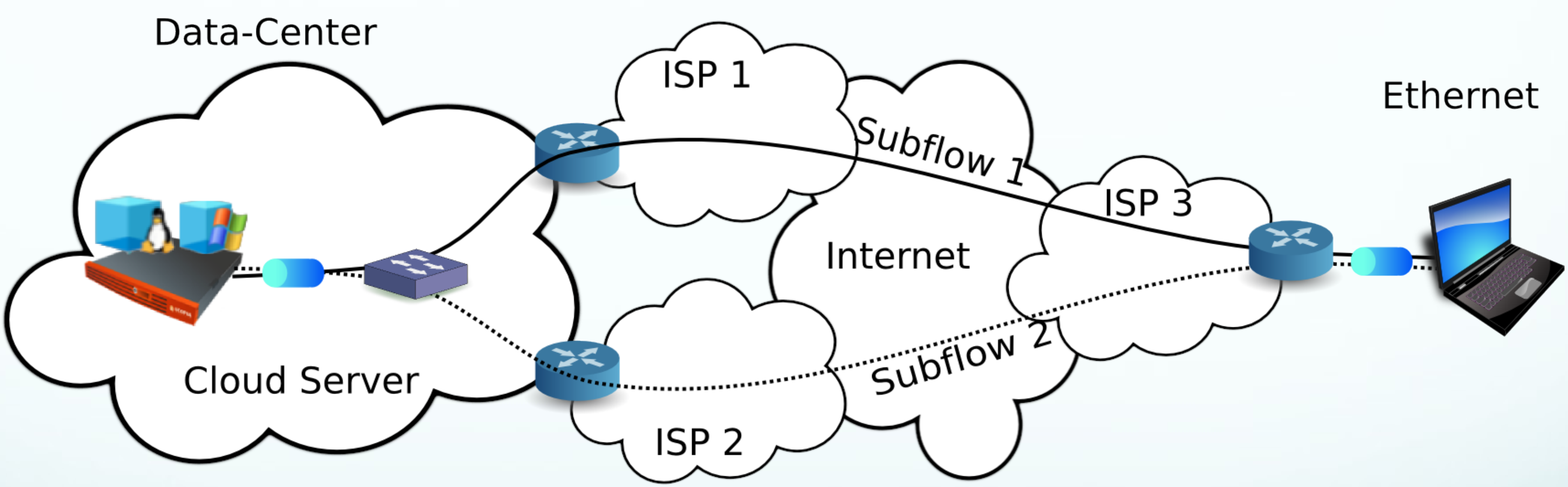
To reach …	Encapsulate to…
EID B	**RB1** **RB2**



Our testbed

Lisp-Lab
Building Today the Internet of Tomorrow

nu@ge



Our guess: Number of WAN paths = Number of RLOCs

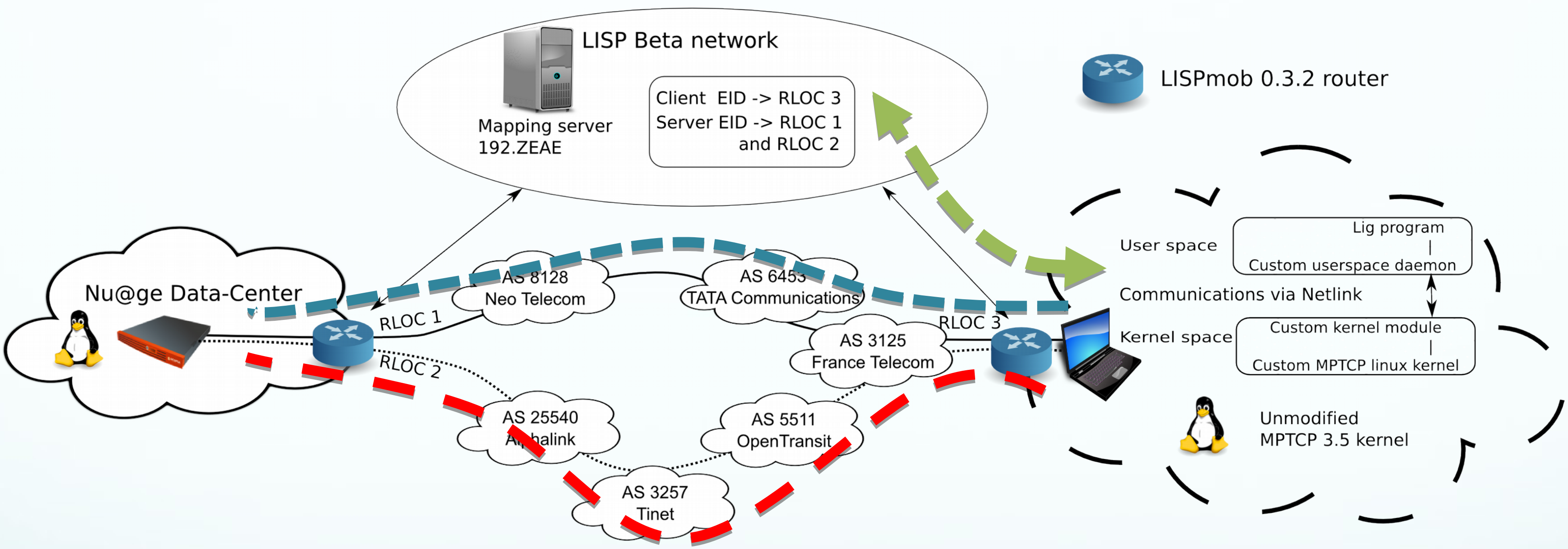
1/ First subflow established

2/ Retrieves number of RLOCs

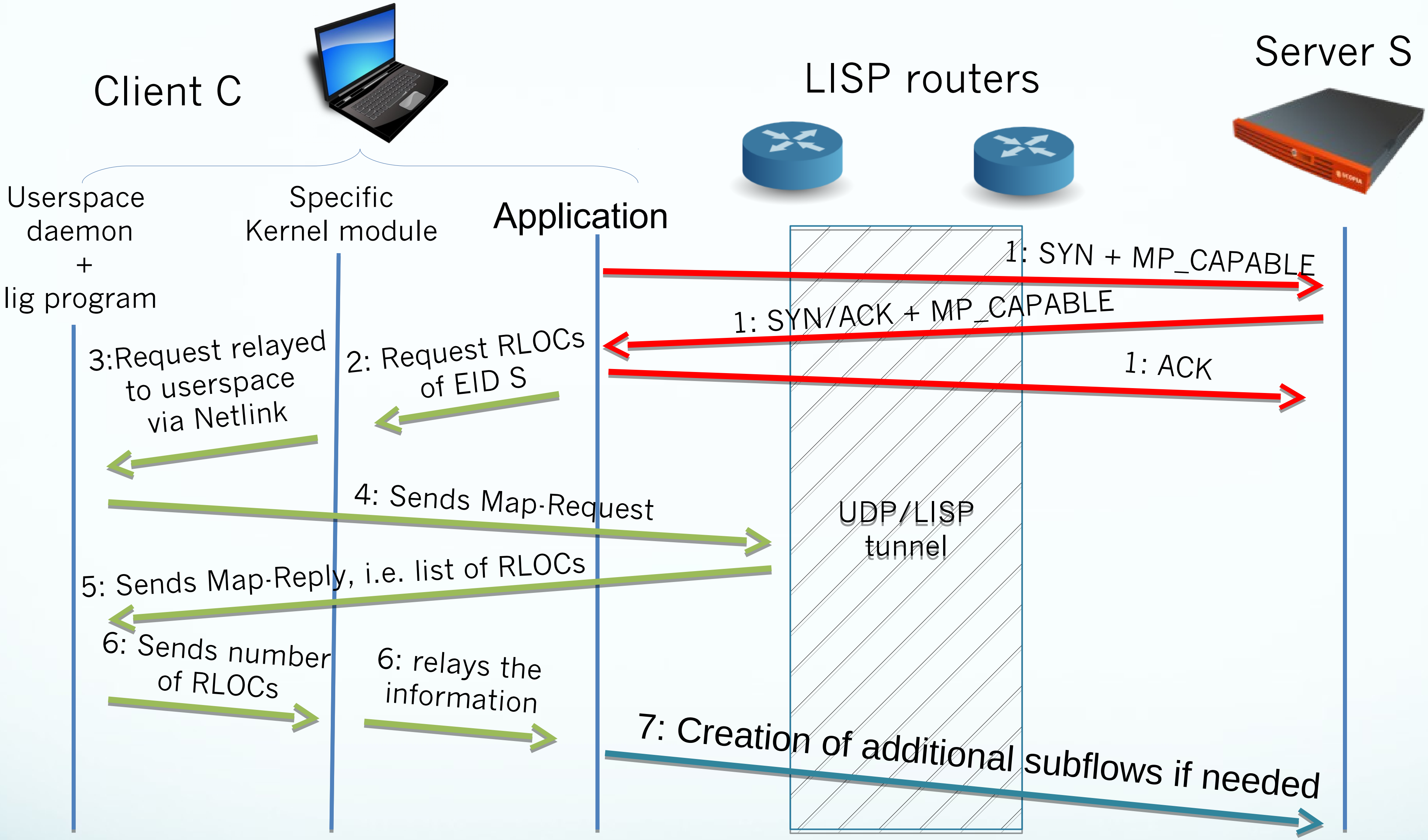
3/ Creation of 2^{nd} subflow with Specific source port number

Lisp-Lab
Building Today the Internet of Tomorrow

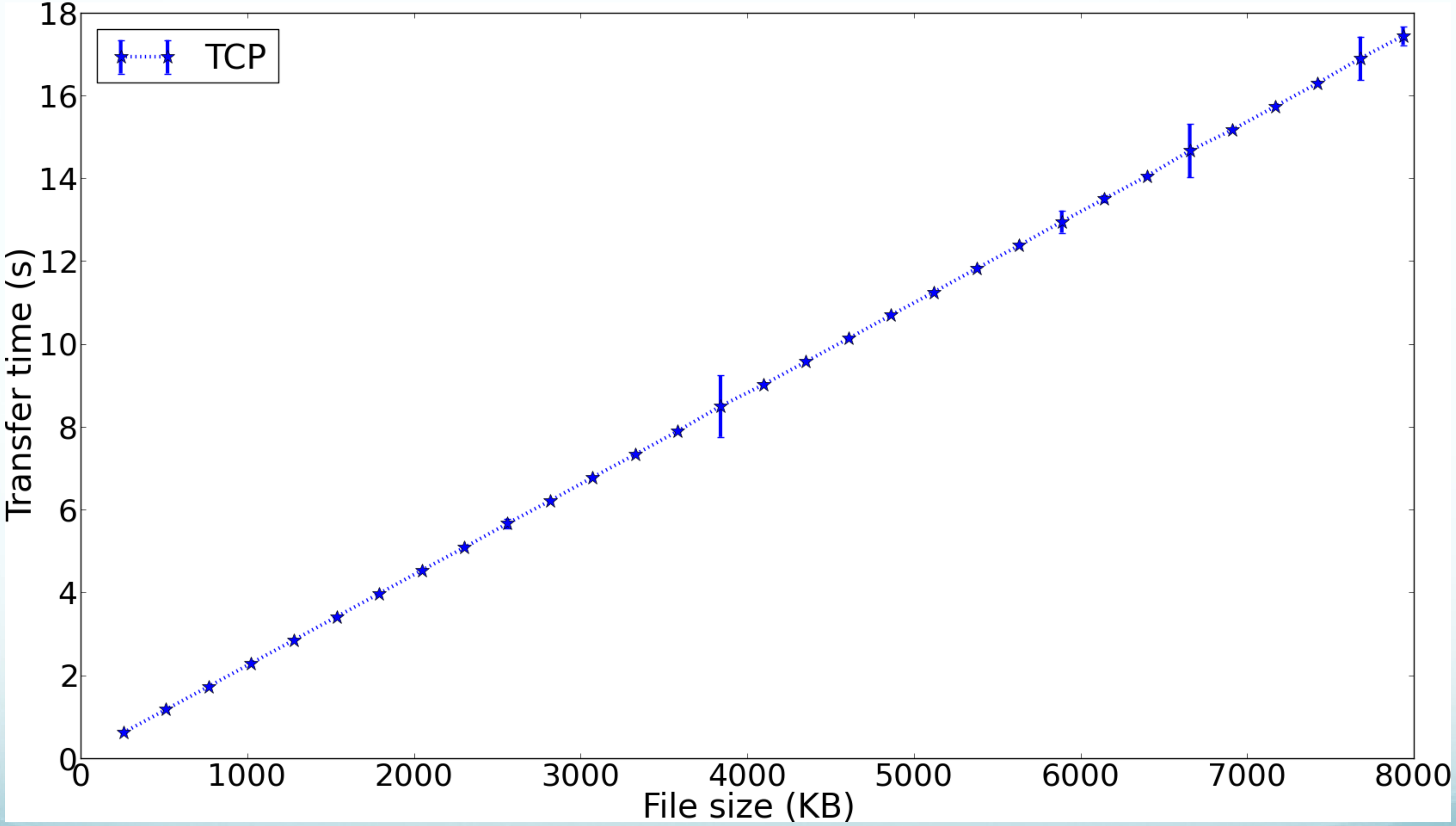
nu@ge



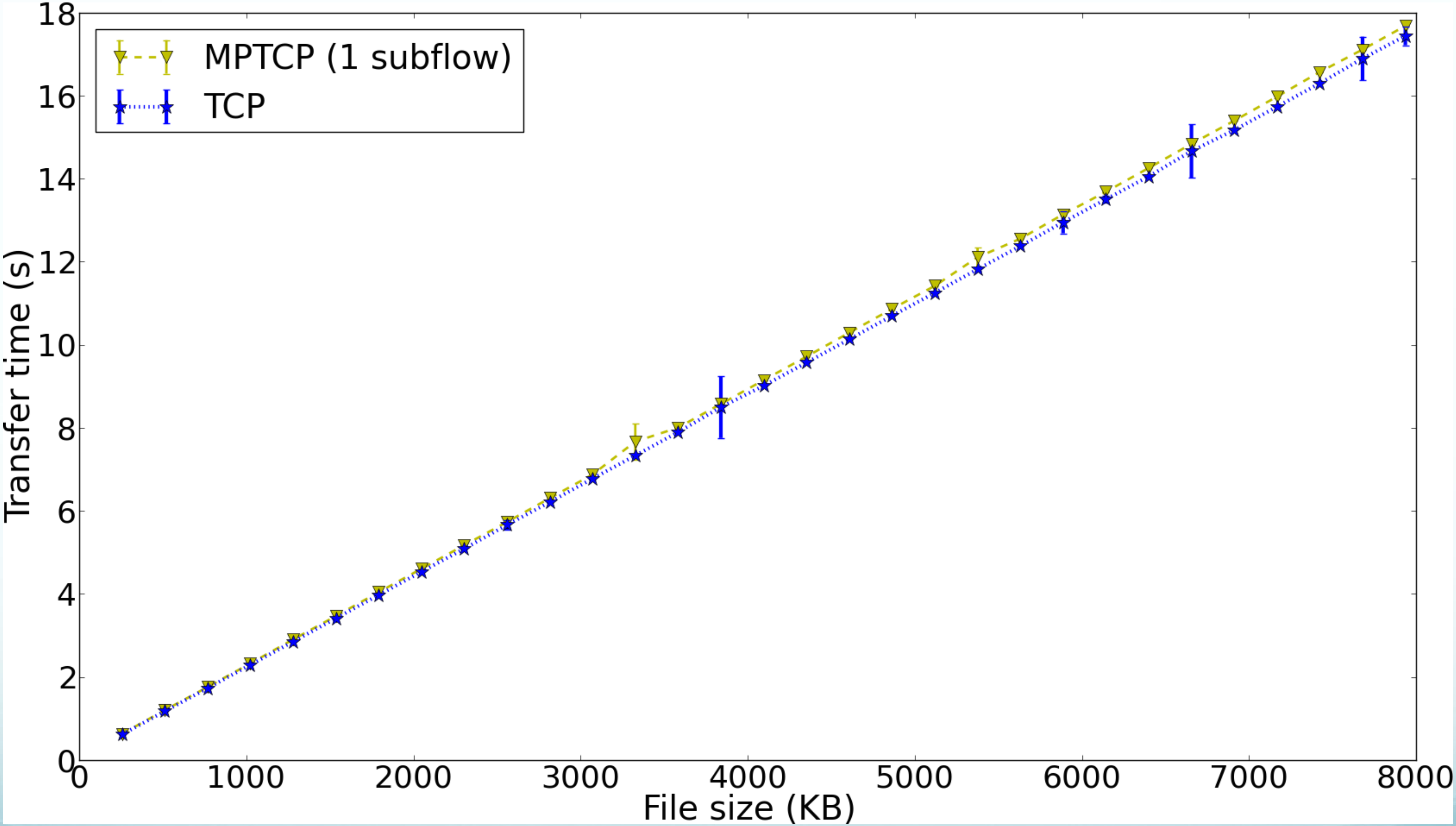
(ΣPortNumbers) %2 = **0** and **1** (or **1** and **0**)


Client C
LISP routers
Server S
Userspace daemon + lig program
Specific Kernel module
Application
1: SYN + MP_CAPABLE
1: SYN/ACK + MP_CAPABLE
1: ACK
2: Request RLOCs of EID S
3:Request relayed to userspace via Netlink
4: Sends Map-Request
UDP/LISP tunnel
5: Sends Map-Reply, i.e. list of RLOCs
6: Sends number of RLOCs
6: relays the information
7: Creation of additional subflows if needed

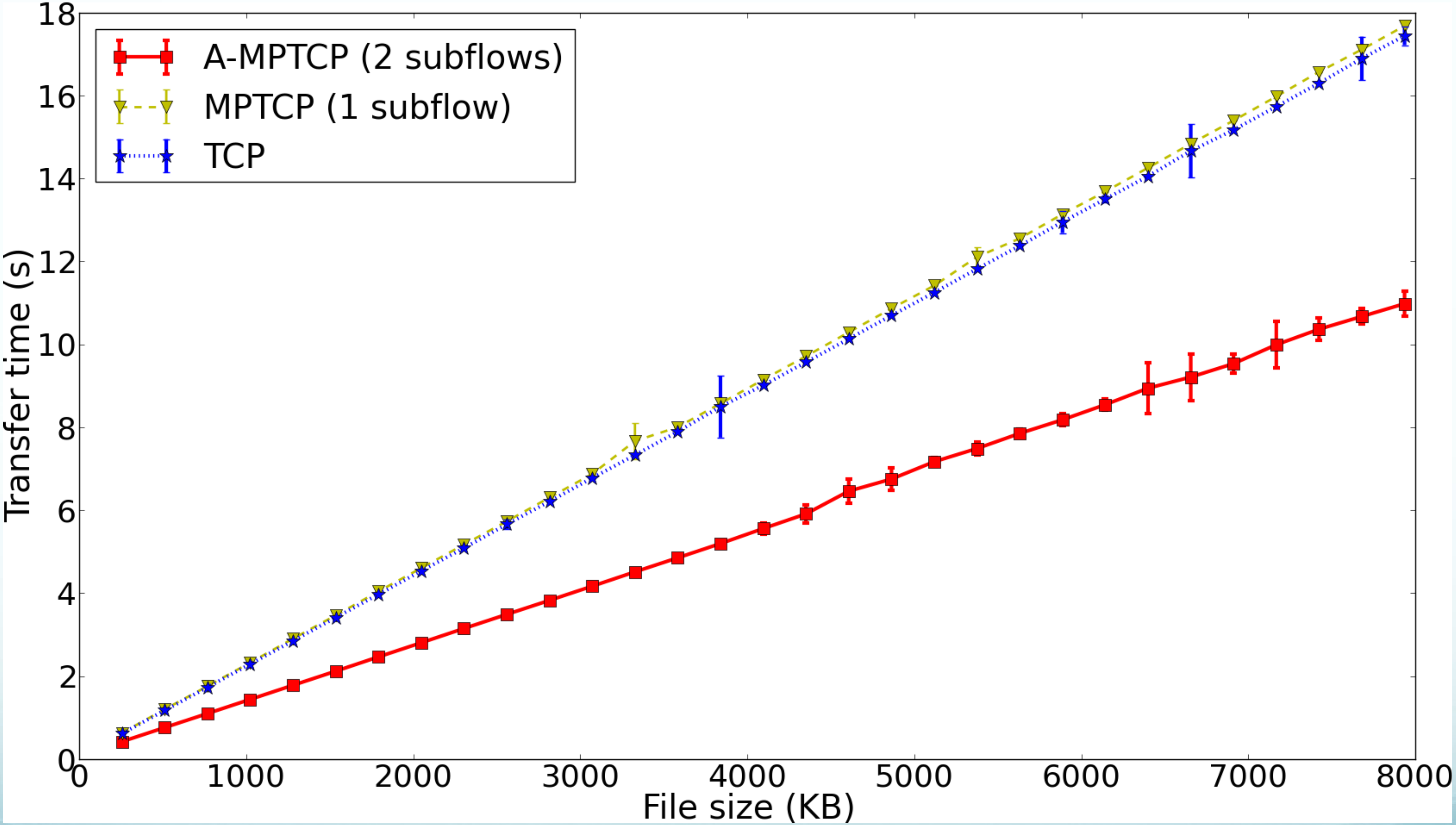
Transfer time (lower is better)



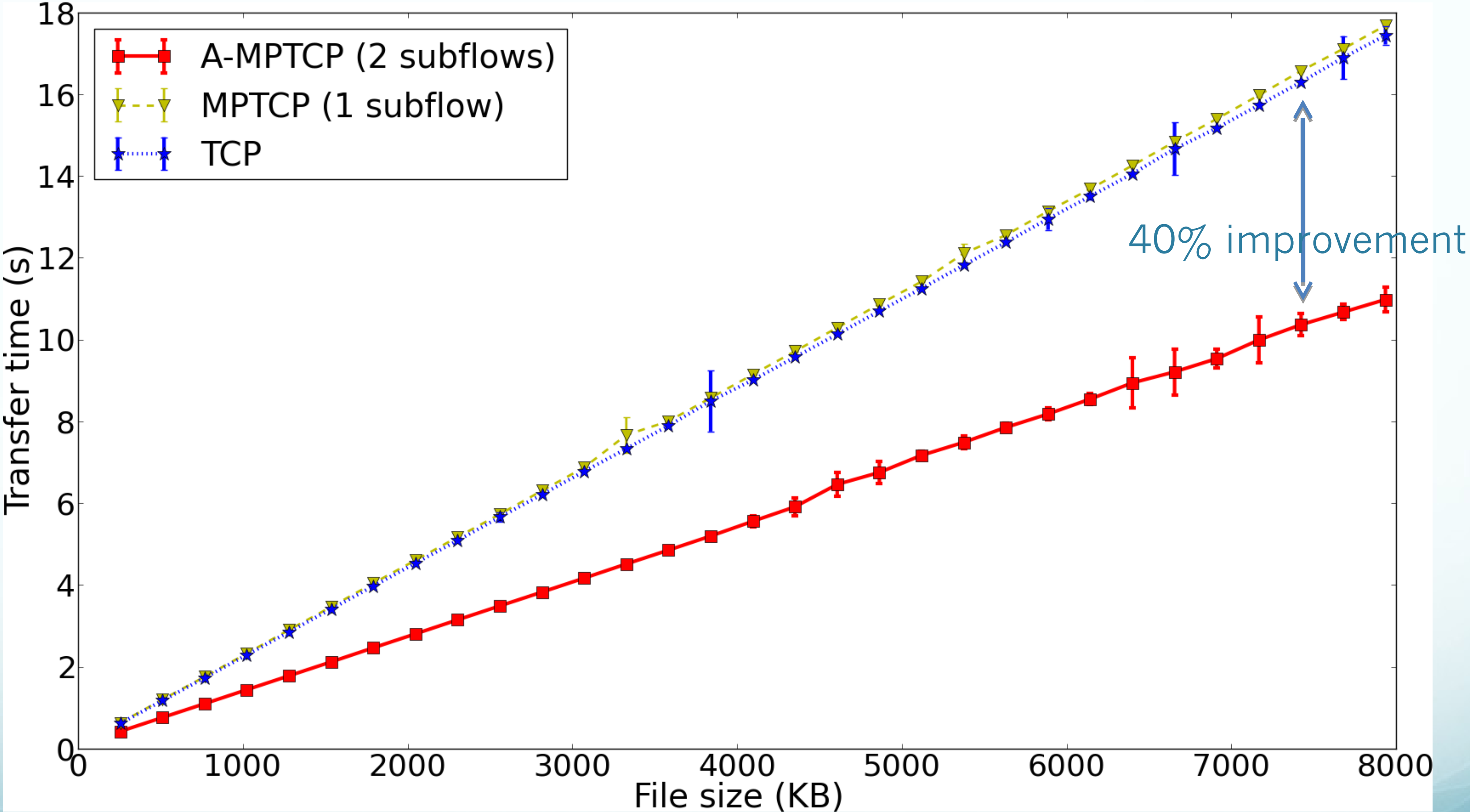
Transfer time (lower is better)



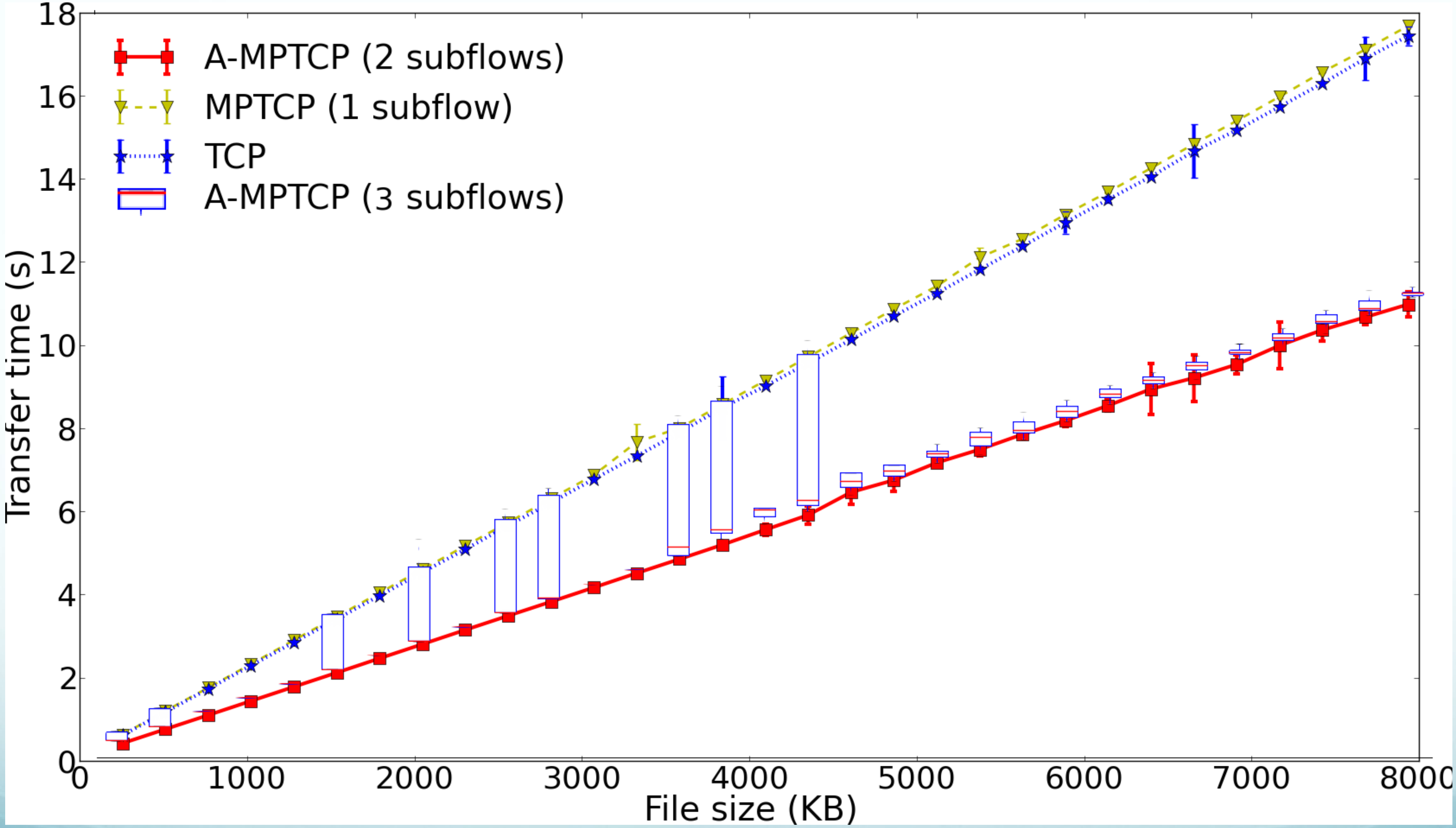
Transfer time (20 iterations)



Results on 20 iterations



3 subflows



A-MPTCP (2 subflows)
MPTCP (1 subflow)
TCP
A-MPTCP (3 subflows)
Transfer time (s)
File size (KB)
0
2
4
6
8
10
12
14
16
18
0
1000
2000
3000
4000
5000
6000
7000
8000

Summary

- A-MPTCP increases throughput in certain conditions
  - enough WAN paths given by LISP
  - WAN path is the bottleneck

- Too many subflows can hurt the goodput
- Crosslayer architectures are complex to deploy

- Perspective
  - Detect when using subflow would hurt

MPTCPNUMERICS : Window management for MPTCP

1. Overview
2. Presentation of LISP
3. Simulation Results

M. Coudron, D. Duy, S. Secci *« On buffer and window management for MPTCP »*, NoF 2016

Contribution overview

Objective

- Throughput is but one metric, users may want to trade (some) goodput for better confidentiality

Problems

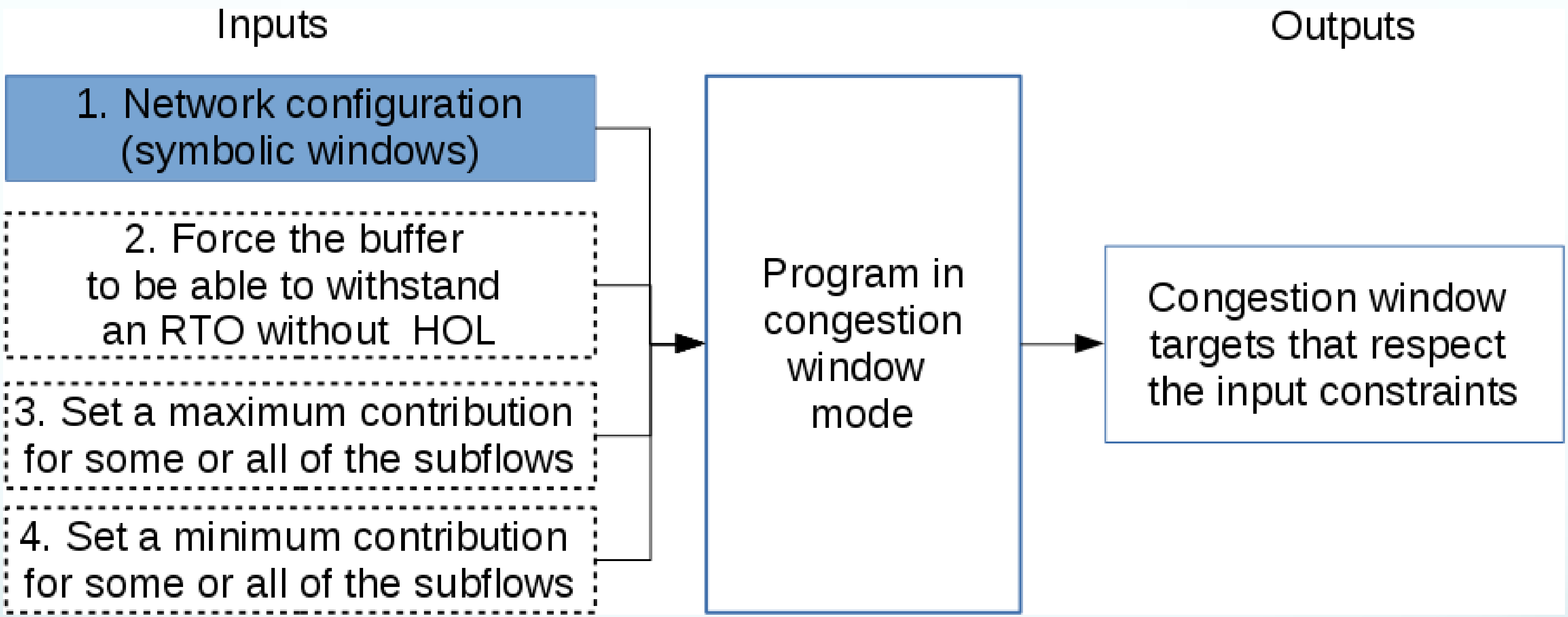
- MPTCP advertised window is shared between subflows so an efficient subflow might starve others
- goodput-only approaches can lead to discard less efficient paths but these paths may present an interest for the user

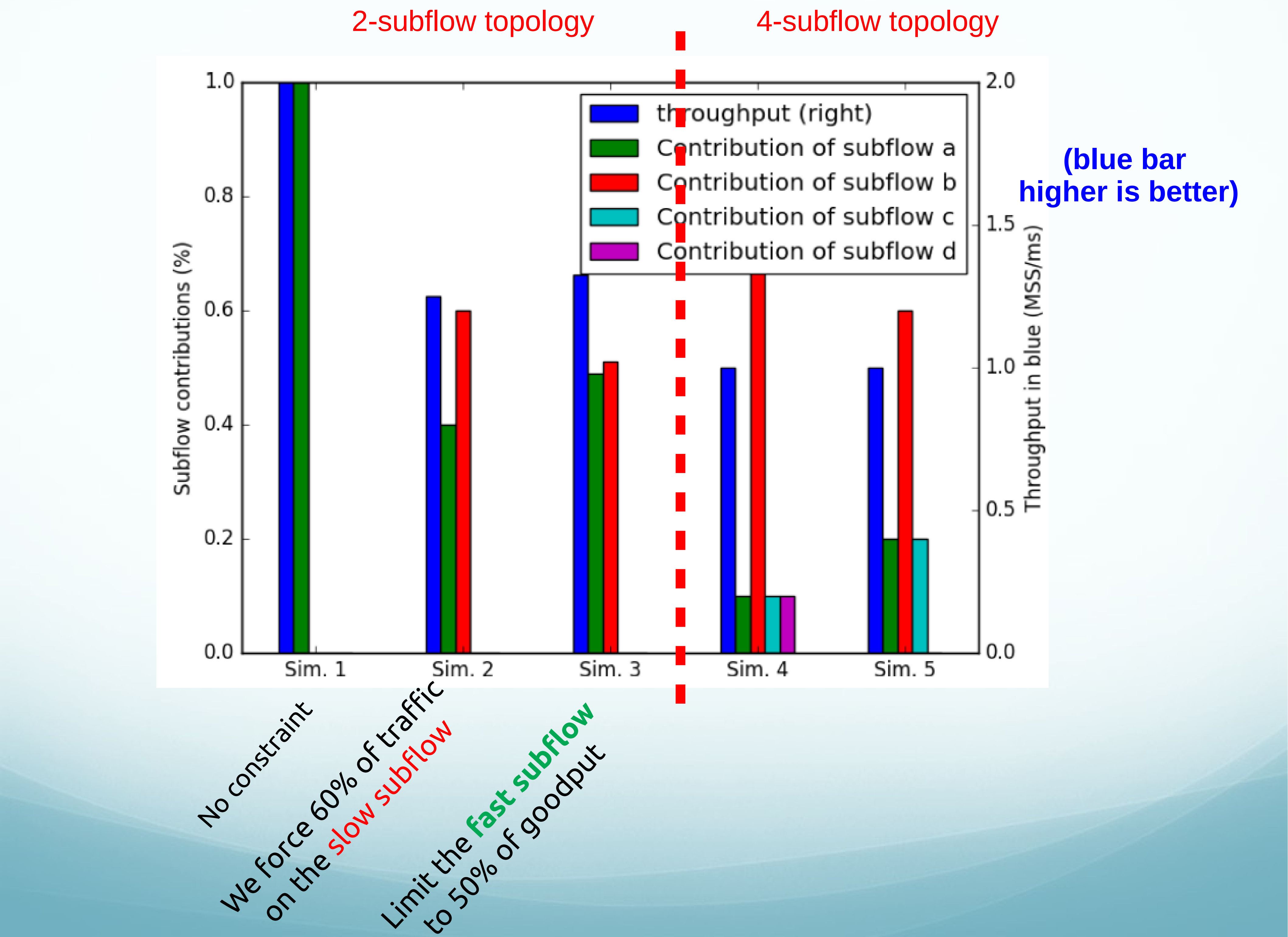
Solution

- Cap the congestion windows on best paths to ensure free buffer space for less performing subflows

MPTCPNUMERICS: an event based simulator

- Custom discrete time simulator
- Accepts as input
  - A topology configuration file (with RTTs, buffer size)
  - Constraints on subflow contributions ; e.g.,
  - « the Wifi subflow should contribute to at least (/no more than) 50 % of the goodput»
- to give congestion window targets under constraints


Inputs
Outputs
1. Network configuration (symbolic windows)
2. Force the buffer to be able to withstand an RTO without HOL
3. Set a maximum contribution for some or all of the subflows
4. Set a minimum contribution for some or all of the subflows
Program in congestion window mode
Congestion window targets that respect the input constraints


2-subflow topology
4-subflow topology
throughput (right)
Contribution of subflow a
Contribution of subflow b
Contribution of subflow c
Contribution of subflow d
Subflow contributions (%)
Throughput in blue (MSS/ms)
1.0
0.8
0.6
0.4
0.2
0.0
2.0
1.5
1.0
0.5
0.0
Sim. 1
Sim. 2
Sim. 3
Sim. 4
Sim. 5
(blue bar higher is better)
No constraint
We force 60% of traffic on the slow subflow
Limit the fast subflow to 50% of goodput

Perspectives

- Go beyond the “throughput” objective: throughput aggregation is not the only metric of interest to users
  - Latency sensitive applications
  - Higher confidentiality at the expense of throughput
  - Monetary costs related to interfaces
  -
- Our tool may help the user choose strategies such as « giving up 20 % of the foreseen throughput for an increase in path diversity of 50 % »

Conclusion

Conclusion

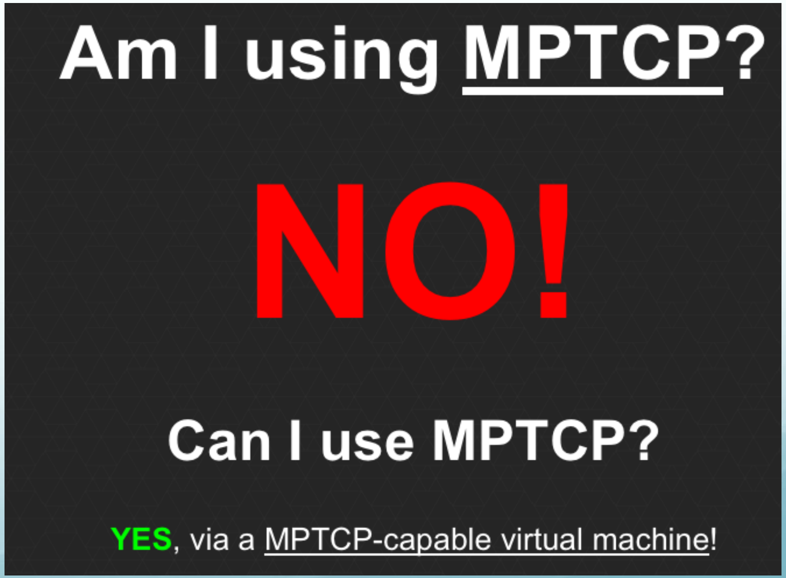
- MPTCP is viable today but throughput can be worse than TCP on heterogeneous paths
- Throughput aggregation is the main area of study
- But users may tradeoff throughput for
  - Energy/financiary economies
  - Latency improvements (packet duplication/Network coding
  - Higher confidentiality
- Knowing the application traffic profile would help
  - TAPS (Transport Area Protocol Services) proposes an abstraction to do just that

Source code available at
http://github.com/lip6-mptcp

coudron@iij.ad.jp

Want to try MPTCP ?

1. Install the MPTCP kernel (Debian/Ubuntu)
   - http://multipath-tcp.org
2. Reboot
3. Go to www.amiusingmptcp.de

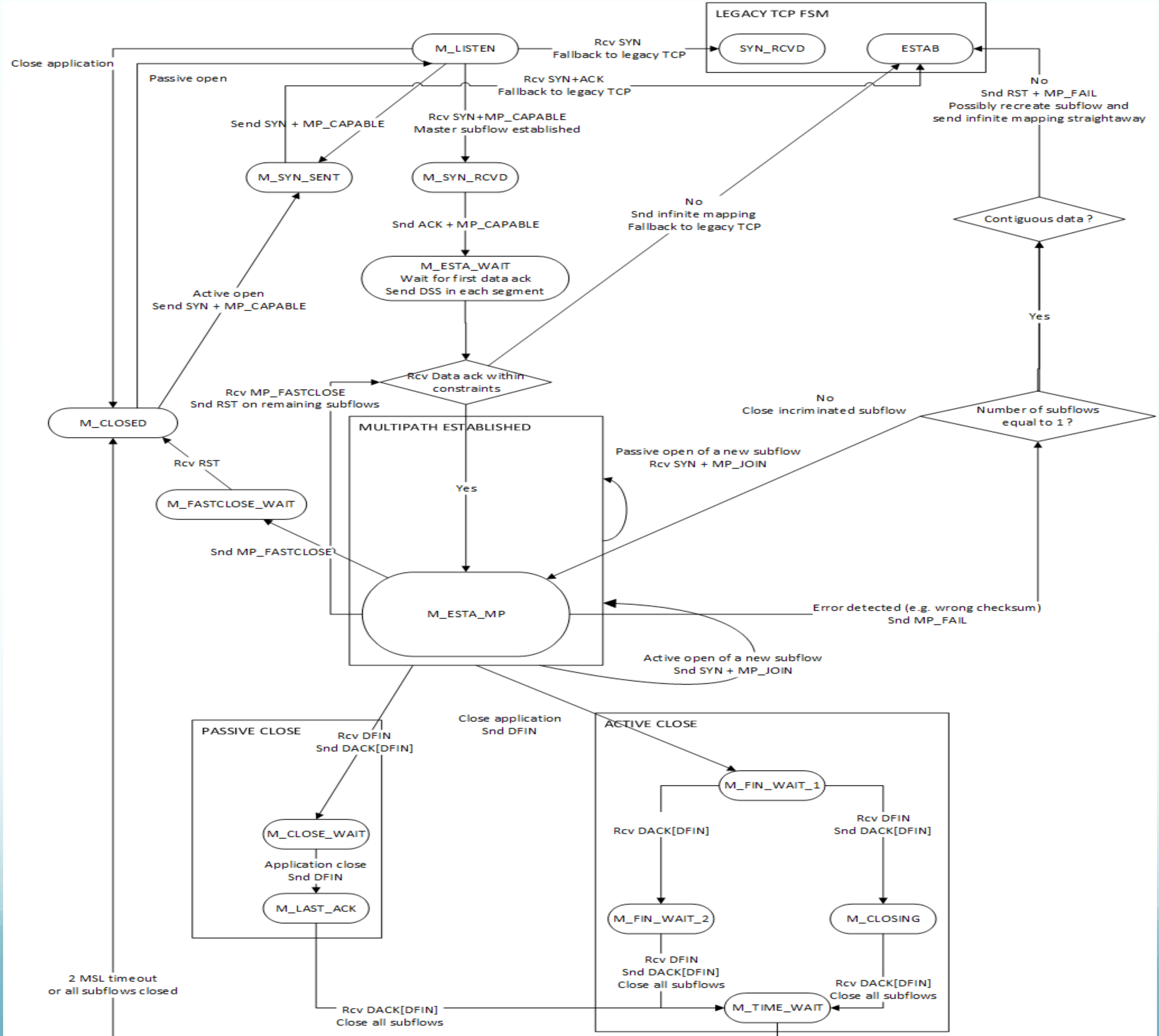


```
    Window size value: 909
    [Calculated window size: 116352]
    [Window size scaling factor: 128]
  ▶ Checksum: 0x6e8b [unchecked, not all data available]
    Urgent pointer: 0
  ▼ Options: (32 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps, Multipath TCP
    ▶ No-Operation (NOP)
    ▶ No-Operation (NOP)
    ▶ Timestamps: TSval 1389505775, TSecr 21868216
    ▼ Multipath TCP: Data Sequence Signal
        Kind: Multipath TCP (30)
        Length: 20
        0010 .... = Multipath TCP subtype: Data Sequence Signal (2)
      ▼ Multipath TCP flags: 0x05
          ...0 .... = DATA_FIN: 0
          .... 0... = Data Sequence Number is 8 octets: 0
          .... .1.. = Data Sequence Number, Subflow Sequence Number, Data-level Length, Checksum present:
          .... ..0. = Data ACK is 8 octets: 0
          .... ...1 = Data ACK is present: 1
        Original MPTCP Data ACK (32 bits): 2020799161
        [Multipath TCP Data ACK: 376 (Relative)]
        Data Sequence Number: 1671049916  (32bits version)
        Subflow Sequence Number: 467
        Data-level Length: 245
        [Data Sequence Number: 467 (Relative)]
  ▶ [SEQ/ACK analysis]
  ▼ [MPTCP analysis]
      [Master flow: master is tcp stream 0]
      [Stream index: 0]
      [TCP subflow stream id(s): 2 1 0]
      [Segment Data Sequence Number start: 1671049916 (64bits)]
      [Segment Data Sequence Number end: 1671050160 (64bits)]
      1671049915 found in packet 16 (current frame=19)
      Application latency: 0.297393000 seconds
```

Peer reviewed communications

- *« Augmented MPTCP communications », ICNP 2013*
- *« Crosslayer cooperation to boost MPTCP performance in the cloud», ICNP 2013*
- *« Differentiated pacing on multiple paths to improve one-way delay estimations », IM2015*
- *« On buffer and window management for MPTCP », NoF 2016*
- *« Per node clocks to simulate time desynchronization in networks », WNS3 2016*
- *« Multipath transmission for the Internet: a survey », IEEE Communications Surveys & Tutorial vol. 18 N°4 Déc. 2016*
- *« Multipath TCP in NS-3 : implementation evaluation », under major revision, Computer Networks*

MPTCP State Machine



LEGACY TCP FSM
SYN_RCVD
ESTAB
M_LISTEN
Close application
Passive open
Rcv SYN
Fallback to legacy TCP
Rcv SYN+ACK
Fallback to legacy TCP
No
Snd RST + MP_FAIL
Possibly recreate subflow and
send infinite mapping straightaway
Send SYN + MP_CAPABLE
Rcv SYN+MP_CAPABLE
Master subflow established
M_SYN_SENT
M_SYN_RCVD
No
Snd infinite mapping
Fallback to legacy TCP
Contiguous data ?
Snd ACK + MP_CAPABLE
M_ESTA_WAIT
Wait for first data ack
Send DSS in each segment
Active open
Send SYN + MP_CAPABLE
Yes
Rcv Data ack within
constraints
Rcv MP_FASTCLOSE
Snd RST on remaining subflows
No
Close incriminated subflow
Number of subflows
equal to 1 ?
M_CLOSED
MULTIPATH ESTABLISHED
Passive open of a new subflow
Rcv SYN + MP_JOIN
Rcv RST
Yes
M_FASTCLOSE_WAIT
Snd MP_FASTCLOSE
M_ESTA_MP
Error detected (e.g. wrong checksum)
Snd MP_FAIL
Active open of a new subflow
Snd SYN + MP_JOIN
Close application
Snd DFIN
PASSIVE CLOSE
Rcv DFIN
Snd DACK[DFIN]
ACTIVE CLOSE
M_FIN_WAIT_1
Rcv DACK[DFIN]
Rcv DFIN
Snd DACK[DFIN]
M_CLOSE_WAIT
Application close
Snd DFIN
M_LAST_ACK
M_FIN_WAIT_2
M_CLOSING
Rcv DFIN
Snd DACK[DFIN]
Close all subflows
2 MSL timeout
or all subflows closed
Rcv DACK[DFIN]
Close all subflows
M_TIME_WAIT
Rcv DACK[DFIN]
Close all subflows

Buffer mode

To handle a fast retransmit scenario

$$\gamma \geq 2 \sum_{i}^{N} BW_i * \max_i RTT_i$$

To handle a Retransmission Time-Out

$$\gamma \geq \sum_{i}^{N} BW_i * \max_i RTO_i$$

Standard recommendation

Such recommandation are for bulk transfer :

- No recommendation for other traffic patterns or for lossless scenarios

Buffer requirements Formulated as an Integer Linear Programming (ILP) model :

Objective : minimize buffer size gamma

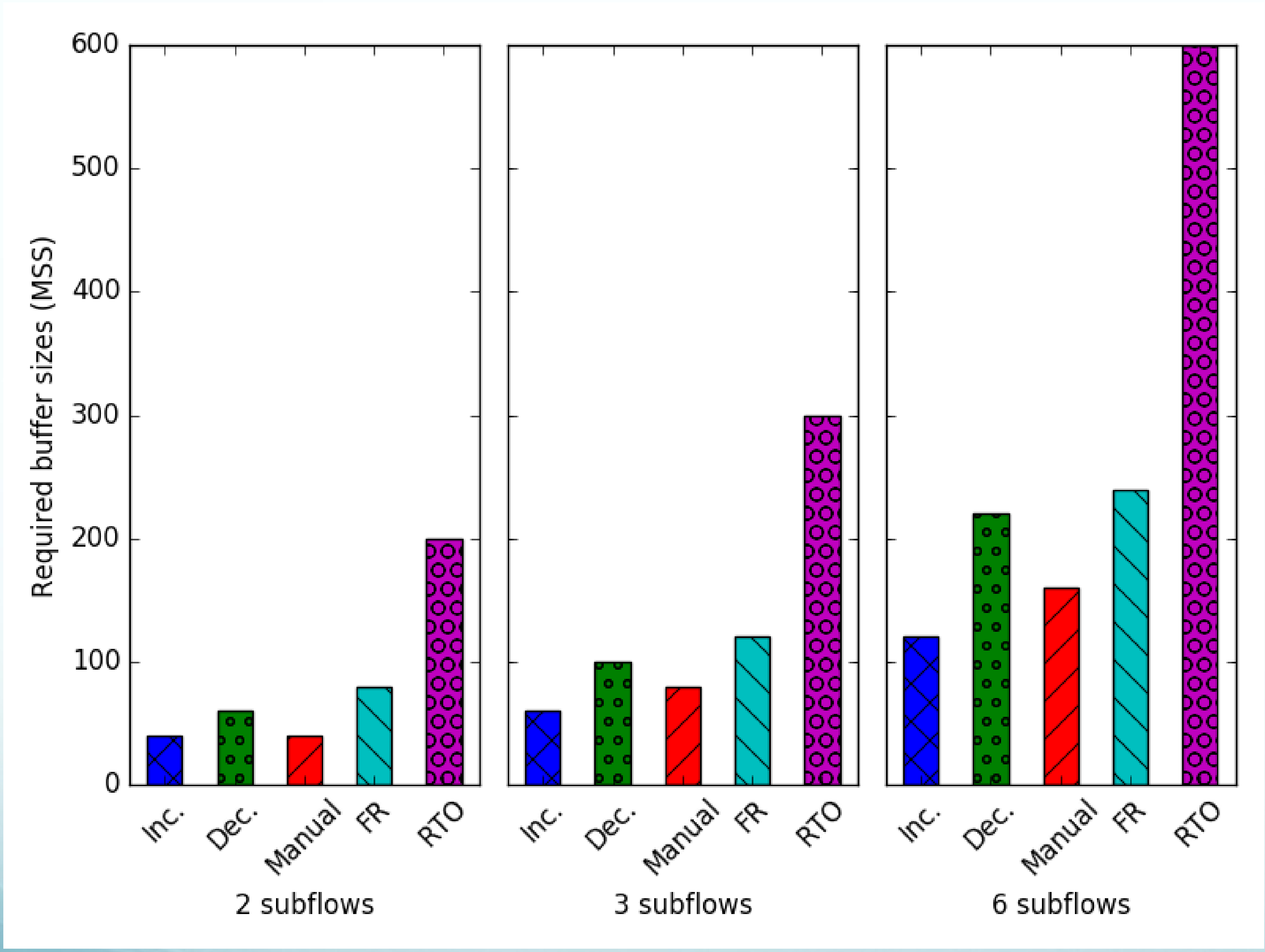
Constraints : TCP flow control

$$\min \gamma$$

$$\forall i \in [0; N] \ \forall p_t^i \in P^i$$

$$mss_i \cdot cwnd_i^{max} \leq \gamma - \sum_{j=1}^{N} \sum_{p \in P^j(t)} 1^{fly}(p) \cdot mss_j \cdot cwnd_j^{max}$$

Influence of scheduling in buffer requirements



Inc/Dec/Manual
Are different
scheduling
Strategies

FR=Fast Retransmit
RTO=Retransmission
Timeout

Same RTT/window
Different fowd