BLEST: Blocking Estimation-based MPTCP Scheduler for Heterogeneous Networks

Simone Ferlin, Ozgu Alay, Olivier Mehani and Roksana Boreli

[simula . research laboratory]

NICTA

Multipath TCP: What is it?

- TCP/IP is built around the notion of a *single connection* between hosts.
    - TCP connection: IP address + TCP port

Multipath TCP: What is it?

- TCP/IP is built around the notion of a *single connection* between hosts.
    - TCP connection: IP address + TCP port

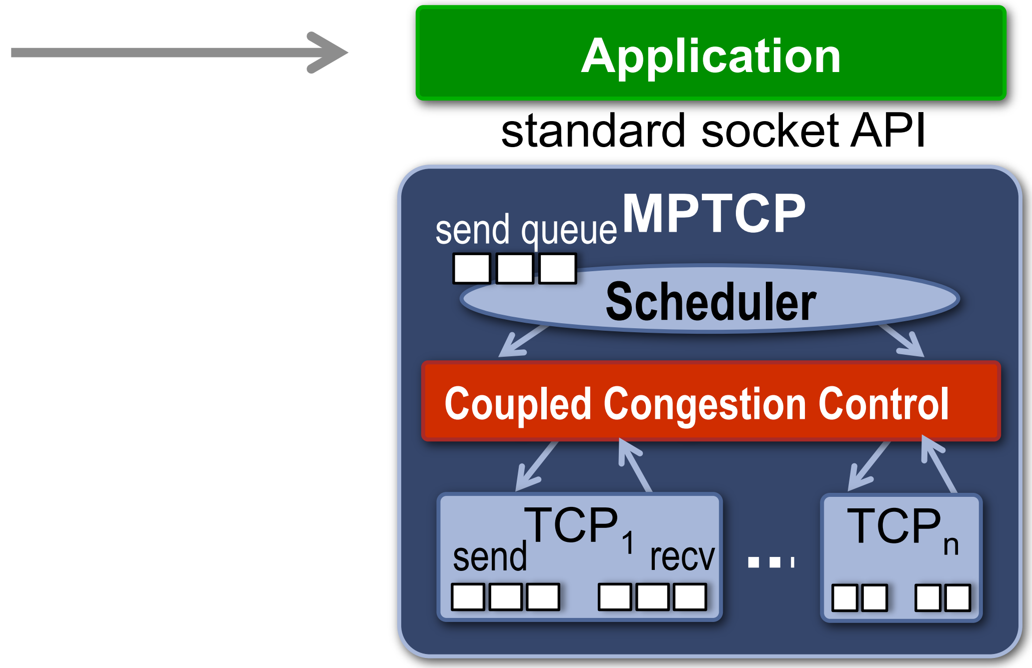
- MPTCP closes the gap between multipath networks and single-path transport.
    - Improved resource utilization (bandwidth aggregation) and reliability.

Multipath TCP: What is it?

- TCP/IP is built around the notion of a *single connection* between hosts.
  - TCP connection: IP address + TCP port

- MPTCP closes the gap between multipath networks and single-path transport.
  - Improved resource utilization (bandwidth aggregation) and reliability.

Application on top of TCP



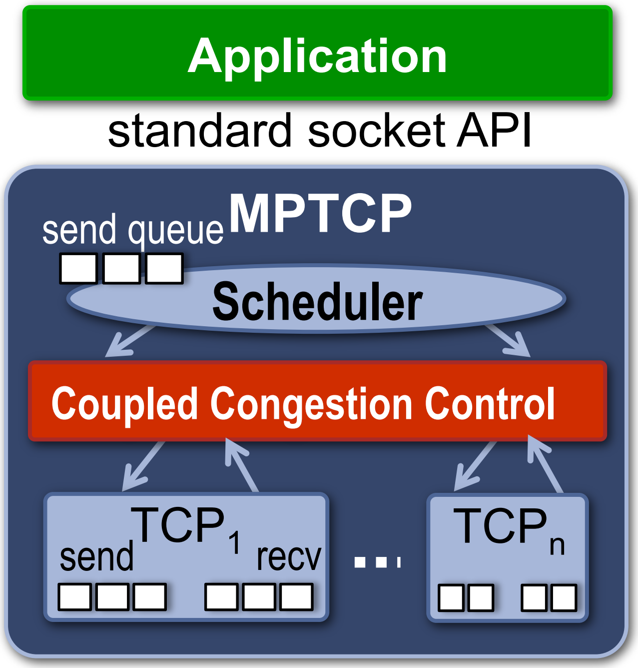
Multipath TCP: What is it?

- TCP/IP is built around the notion of a *single connection* between hosts.
  - TCP connection: IP address + TCP port

- MPTCP closes the gap between multipath networks and single-path transport.
  - Improved resource utilization (bandwidth aggregation) and reliability.

Application on top of TCP

Multipath Scheduler:
- Default: *minimum RTT*



Multipath TCP: What is it?

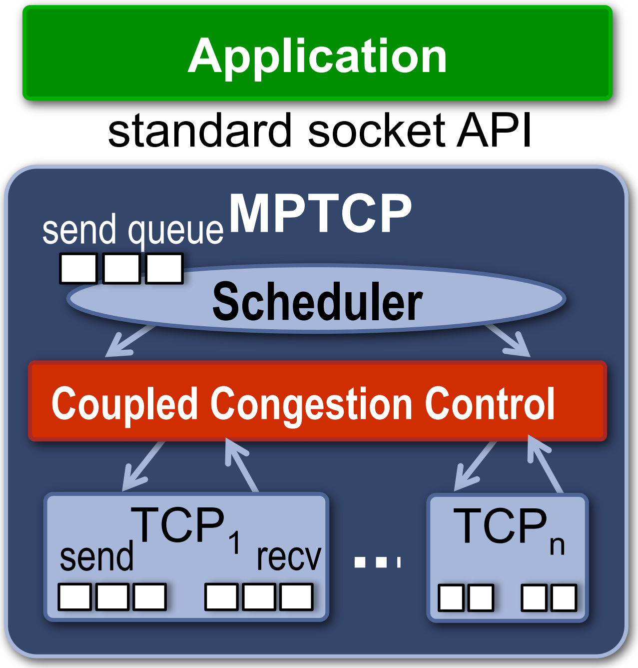
- TCP/IP is built around the notion of a *single connection* between hosts.
  - TCP connection: IP address + TCP port

- MPTCP closes the gap between multipath networks and single-path transport.
  - Improved resource utilization (bandwidth aggregation) and reliability.

Application on top of TCP

Multipath Scheduler:
- Default: *minimum RTT*

Coupled Congestion Control:
- LIA, OLIA and Balia: Reno-based



Multipath TCP: What is it?

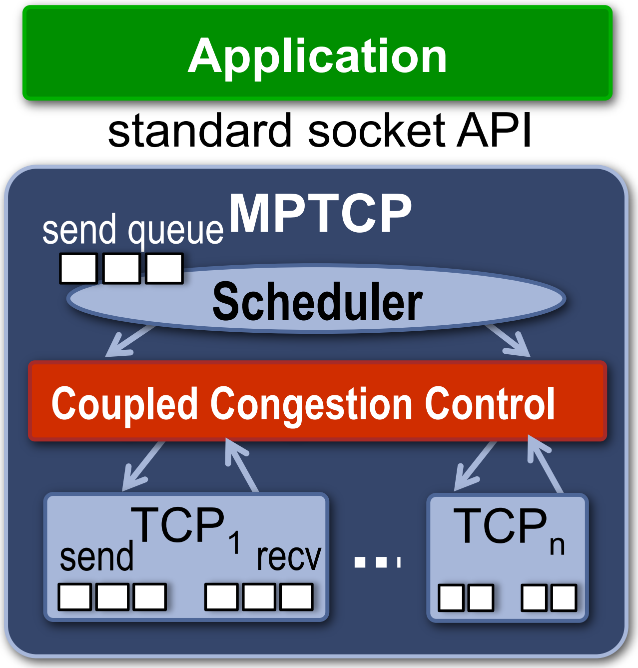
- TCP/IP is built around the notion of a *single connection* between hosts.
  - TCP connection: IP address + TCP port

- MPTCP closes the gap between multipath networks and single-path transport.
  - Improved resource utilization (bandwidth aggregation) and reliability.

Application on top of TCP

Multipath Scheduler:
- **Default: *minimum RTT***

Coupled Congestion Control:
- LIA, OLIA and Balia: Reno-based



Multipath TCP Scheduler: Overview

- MPTCP's default scheduler:
  - Minimum RTT (minRTT)
  - Not standardized nor specified at the IETF

Multipath TCP Scheduler: Overview

- MPTCP's default scheduler:
  - Minimum RTT (minRTT)
  - Not standardized nor specified at the IETF

The MPTCP scheduler first fills the window of the subflow with the lowest RTT (sRTT), then data is sent on the subflow with the next higher sRTT, …

Multipath TCP Scheduler: Overview

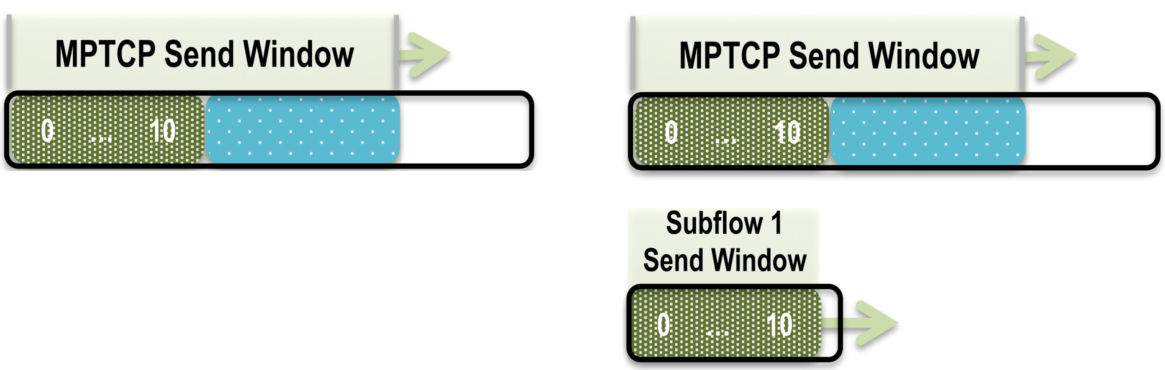
- MPTCP's default scheduler:
  - Minimum RTT (minRTT)
  - Not standardized nor specified at the IETF

The MPTCP scheduler first fills the window of the subflow with the lowest RTT (sRTT), then data is sent on the subflow with the next higher sRTT, …

Multipath TCP Scheduler: Overview

- MPTCP's default scheduler:
  - Minimum RTT (minRTT)
  - Not standardized nor specified at the IETF

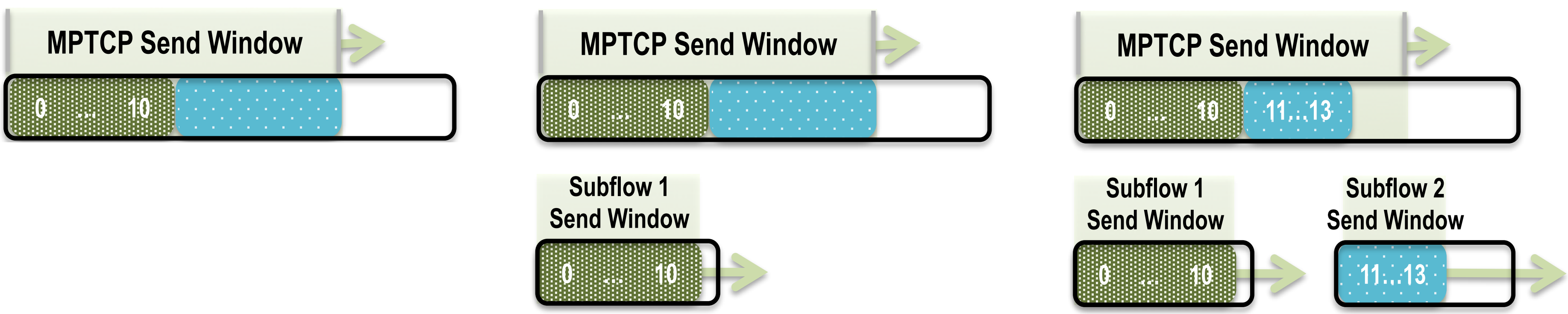
The MPTCP scheduler first fills the window of the subflow with the lowest RTT (sRTT), then data is sent on the subflow with the next higher sRTT, …



Multipath TCP Scheduler: Overview

- MPTCP's default scheduler:
  - Minimum RTT (minRTT)
  - Not standardized nor specified at the IETF

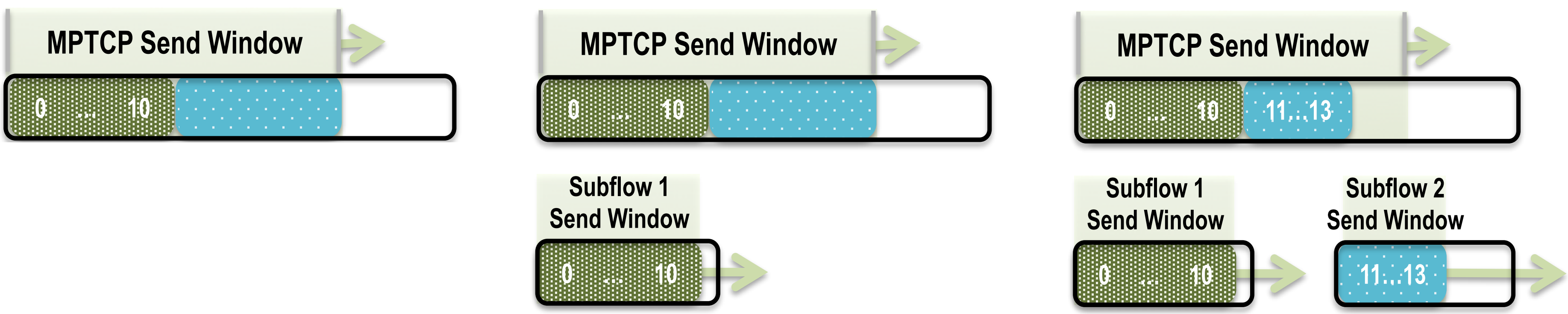
The MPTCP scheduler first fills the window of the subflow with the lowest RTT (sRTT), then data is sent on the subflow with the next higher sRTT, …



Multipath TCP Scheduler: Overview

- MPTCP's default scheduler:
  - Minimum RTT (minRTT)
  - Not standardized nor specified at the IETF

The MPTCP scheduler first fills the window of the subflow with the lowest RTT (sRTT), then data is sent on the subflow with the next higher sRTT, …

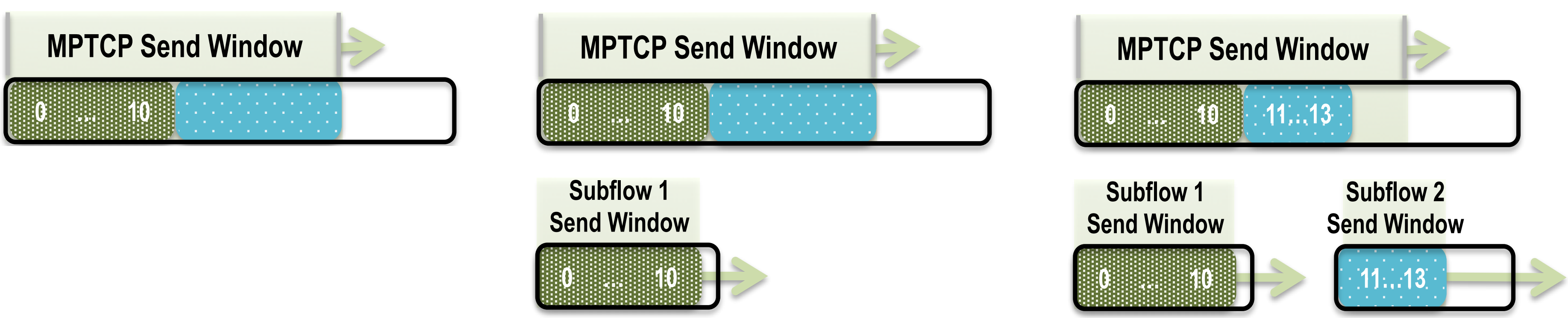


MPTCP's default scheduler *always* fills up the CWND of all available subflows in ascending RTT order.

Multipath TCP Scheduler: Overview

- MPTCP's default scheduler:
  - Minimum RTT (minRTT)
  - Not standardized nor specified at the IETF

The MPTCP scheduler first fills the window of the subflow with the lowest RTT (sRTT), then data is sent on the subflow with the next higher sRTT, …

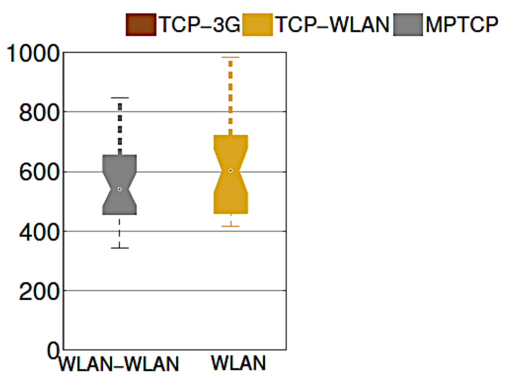


MPTCP's default scheduler *always* fills up the CWND of all available subflows in ascending RTT order.

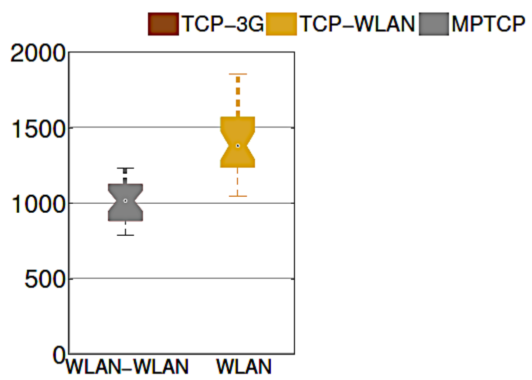
- This is harmful if the subflows have distinct RTT: HoL

Multipath TCP Scheduler: WLAN+WLAN

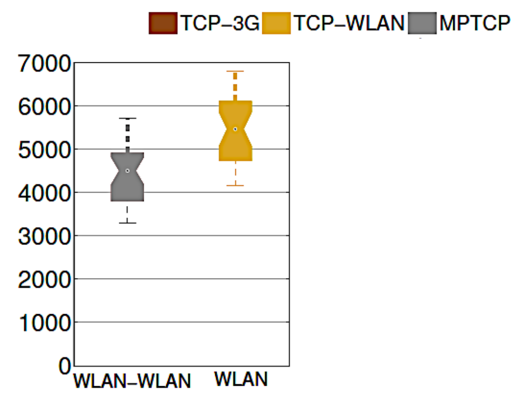
- Metric: **Download Time**
- Traffic: **Web download, 6 concurrent connections, 3 web sites:**
  - Wikipedia: 15 objects, 72 kiB
  - Amazon: 54 objects, 1 MiB
  - Huffington Post: 138 objects, 3.994 MiB
- Setup: CORE emulation with synthetic (UDP, TCP) background traffic



(a) Download time, Wikipedia



(b) Download time, Amazon

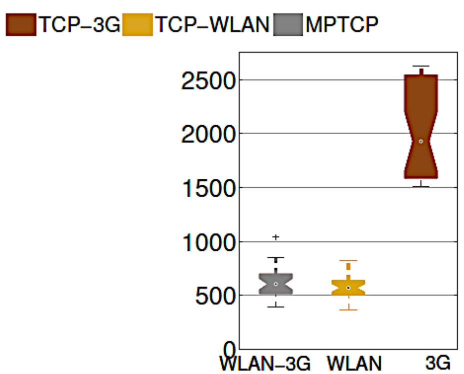


(c) Download time, Huffington Post

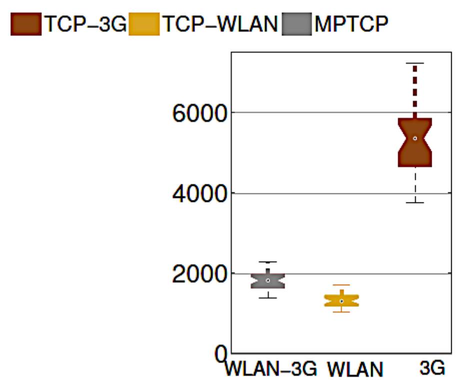
MPTCP in WLAN+WLAN provides gains with larger the downloads.

Multipath TCP Scheduler: 3G+WLAN

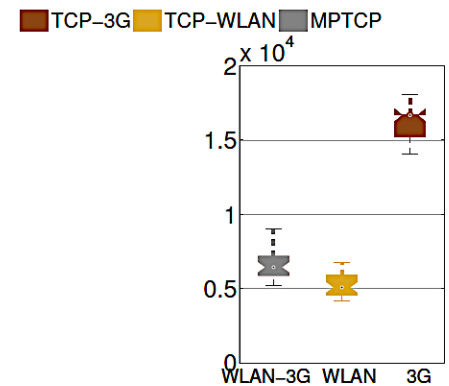
- Metric: **Download Time**
- Traffic: **Web download, 6 concurrent connections, 3 web sites:**
  - Wikipedia: 15 objects, 72 kiB
  - Amazon: 54 objects, 1 MiB
  - Huffington Post: 138 objects, 3.994 MiB
- Setup: CORE emulation with synthetic (UDP, TCP) background traffic



(a) Download time, Wikipedia



(b) Download time, Amazon



(c) Download time, Huffington Post

MPTCP in 3G+WLAN provides **marginal or no gain** with heterogeneity.

Multipath TCP Scheduler: References

Related work:

- **Delay-Aware Packet Scheduler (DAPS)**
  N. Kuhn, E. Lochin, A. Mifdaoui, G. Sarwar, O. Mehani, and R. Boreli,
  DAPS: Intelligent delay-aware packet scheduling for multipath transport
- **Out-of-order Transmission for In-order Arrival Scheduler (OTIAS)**
  F. Yang, Q. Wang, and P. Amer, Out-of-order transmission for in-order arrival
  scheduling policy for multipath TCP

Multipath TCP Scheduler: References

Related work:

- **Delay-Aware Packet Scheduler (DAPS)**
  N. Kuhn, E. Lochin, A. Mifdaoui, G. Sarwar, O. Mehani, and R. Boreli,
  DAPS: Intelligent delay-aware packet scheduling for multipath transport
- **Out-of-order Transmission for In-order Arrival Scheduler (OTIAS)**
  F. Yang, Q. Wang, and P. Amer, Out-of-order transmission for in-order arrival scheduling policy for multipath TCP

Both DAPS and OTIAS not extensively evaluated against MPTCP's default:

- Different test scenarios
- Did not consider different traffic classes (web transfers, CBR or bulk)

Multipath TCP Scheduler: References

Related work:

- **Delay-Aware Packet Scheduler (DAPS)**
  N. Kuhn, E. Lochin, A. Mifdaoui, G. Sarwar, O. Mehani, and R. Boreli,
  DAPS: Intelligent delay-aware packet scheduling for multipath transport
- **Out-of-order Transmission for In-order Arrival Scheduler (OTIAS)**
  F. Yang, Q. Wang, and P. Amer, Out-of-order transmission for in-order arrival scheduling policy for multipath TCP

Both DAPS and OTIAS not extensively evaluated against MPTCP's default:

- Different test scenarios
- Did not consider different traffic classes (web transfers, CBR or bulk)

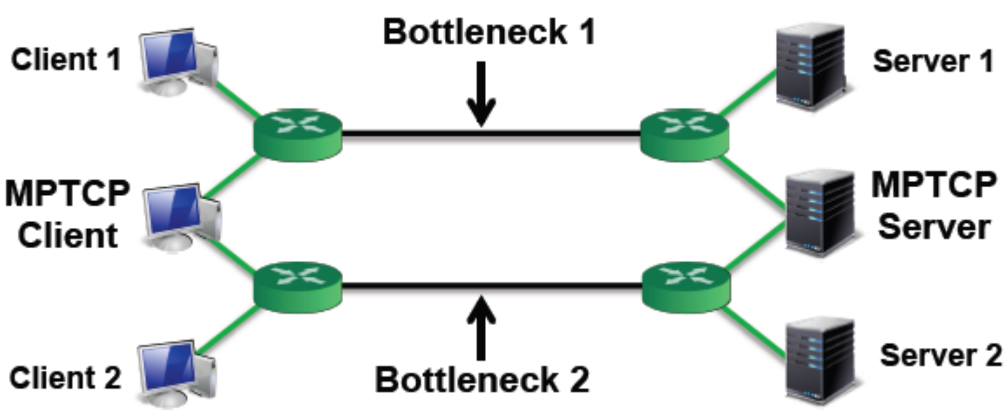
We implement DAPS and OTIAS in the Linux kernel:

- Systematically evaluate their performance – emulation and real-network;
- Address implementation aspects;
- Propose BLEST - Blocking Estimation-based MPTCP Scheduler

Measurement Setup: Emulation

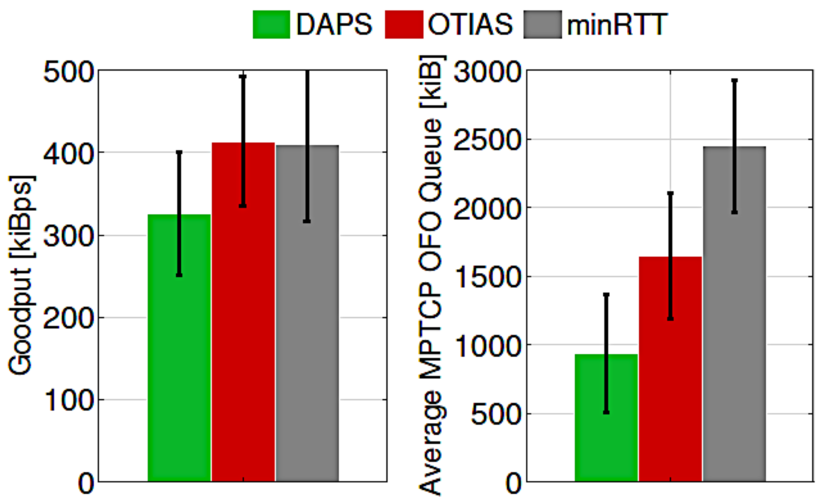
Setup:

- CORE emulation with a synthetic mix background traffic:
  - UDP on/off and TCP rate-limited and bulk flows with distinct RTTs
- Bottleneck settings:
  - WLAN: 25 Mbps, 25 ms, Loss=0.5 to 1%, Bottleneck queue: 100 p
  - 3G: 5 Mbps, 65 ms, Loss=0%, Bottleneck queue: 3750 p
- Socket buffer size*:
  - WLAN+WLAN: 1024 KiB/2048 KiB
  - 3G+WLAN: 1024 KiB/2048 KiB



* Socket buffer: 16 MiB for bulk transfers to evaluate aggregation

Multipath TCP: Bulk Traffic



(a) 3G+WLAN

Metric: Goodput and average OFO queue size

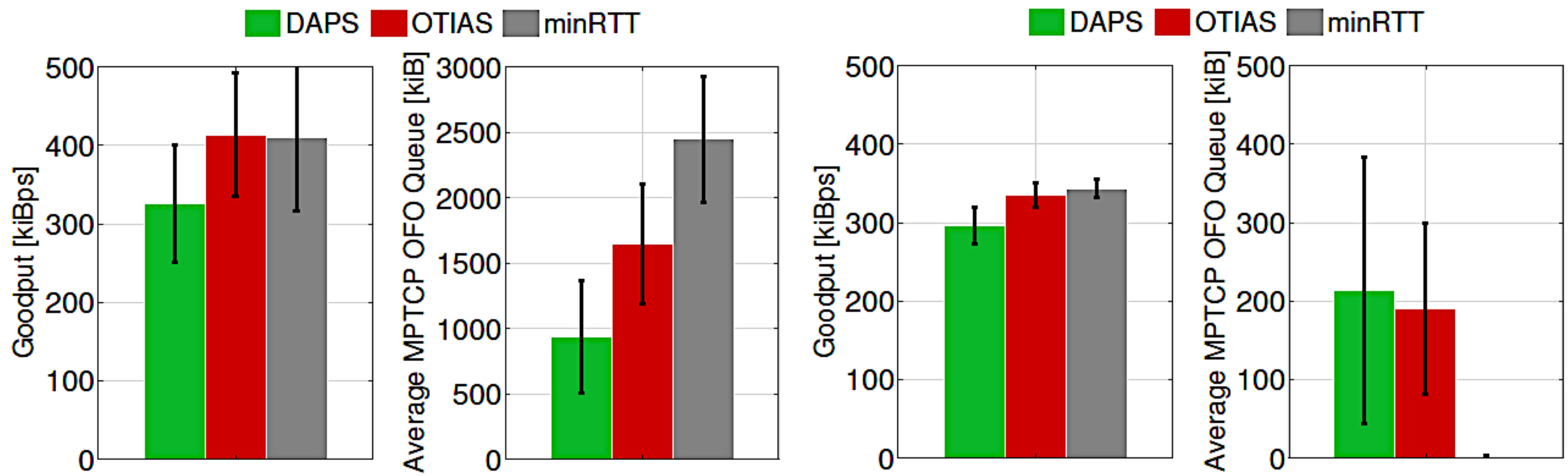
OTIAS:

- **3G+WLAN: The estimation can *discard* the 3G path**

DAPS:

- **3G+WLAN: The estimation never discards a subflow that can send**

Multipath TCP: Bulk Traffic



(a) 3G+WLAN (b) WLAN+WLAN

Metric: Goodput and average OFO queue size

OTIAS:

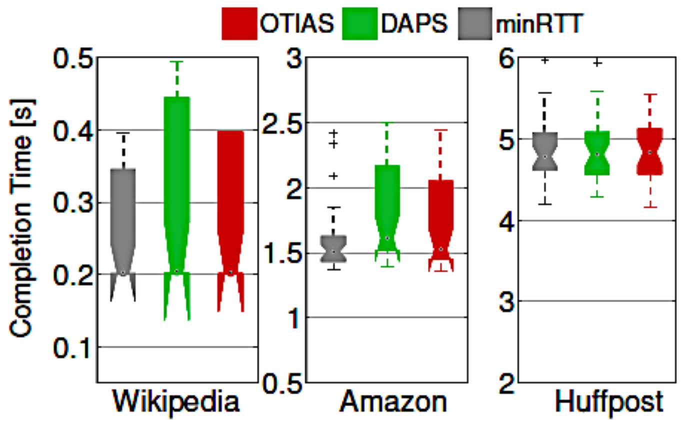
- 3G+WLAN: The estimation can *discard* the 3G path
- **WLAN+WLAN: It lacks retransmission and it builds up send queues**

DAPS:

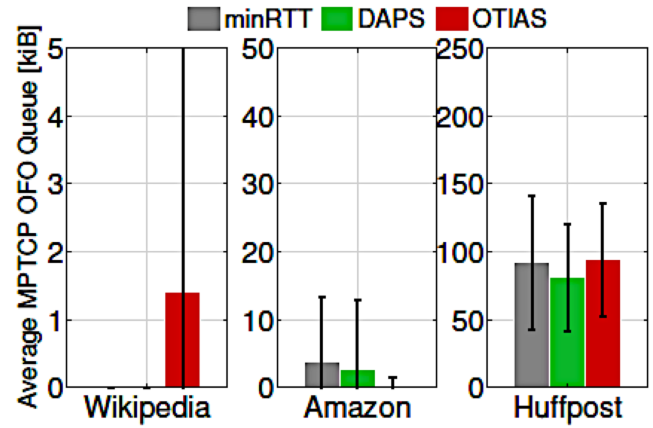
- 3G+WLAN: The estimation never discards a subflow that can send
- **WLAN+WLAN: It has retransmission, but it reacts after *schedule runs***

Multipath TCP: Web Transfers

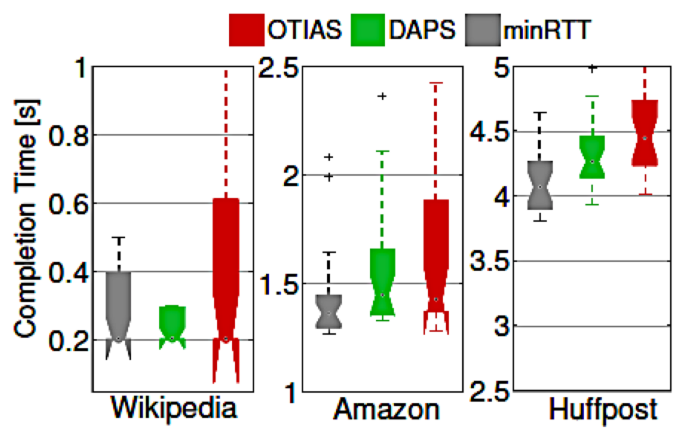
Metric: Completion time and average OFO queue size



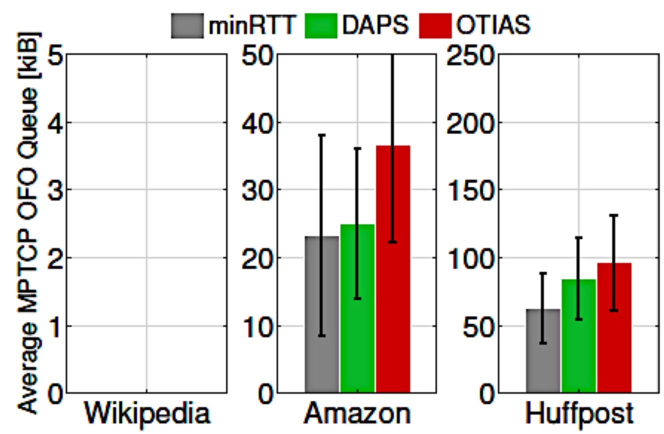
(a) 3G+WLAN, Completion time



(b) 3G+WLAN, OFO queue



(c) WLAN+WLAN, Completion time



(d) WLAN+WLAN, OFO queue

DAPS vs OTIAS

OTIAS:

- Decisions on a per-packet basis, reacting fast with the network's current state
- It builds up queues on the subflows with lowest RTTs, regardless of their CWND, not restricting the scheduler if the CWND is full
- It assumes symmetric forward delays (OWD = RTT/2)
- It does not apply scheduler reinjections (retransmissions)

DAPS:

- DAPS is more complex, it requires more memory at run-time
- It builds *schedules runs* being unable to react to network changes
- It uses *all* subflows available, even if a certain subflow is weak
- It does not apply scheduler reinjections (retransmissions)

Recapitulating: minRTT

- For each new segment, the minRTT, chooses the subflow with lowest RTT among all subflows with window space

Recapitulating: minRTT

- For each new segment, the minRTT, chooses the subflow with lowest RTT among all subflows with window space.

- If MPTCP detects a *full* send window, it retransmits the segment blocking the fastest subflow and *penalises* the slow subflow, halving its CWND
  - This mechanism is called *penalisation and retransmission*
    - Raiciu, Costin, et al. How hard can it be? designing and implementing a deployable multipath TCP

Recapitulating: minRTT

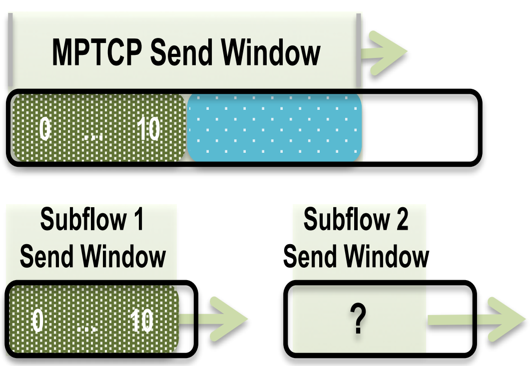
- For each new segment, the minRTT, chooses the subflow with lowest RTT among all subflows with window space.

- If MPTCP detects a *full* send window, it retransmits the segment blocking the fastest subflow and *penalises* the slow subflow, halving its CWND
  - This mechanism is called *penalisation and retransmission*
    - Raiciu, Costin, et al. How hard can it be? designing and implementing a deployable multipath TCP

  - The idea it to reduce the contribution of the slow subflow, keeping its CWND artificially low, *reacting* on receive window limitation
    - In other words, after a *penalisation* the CWND of the slow subflow will start growing again, until blocking reoccurs.

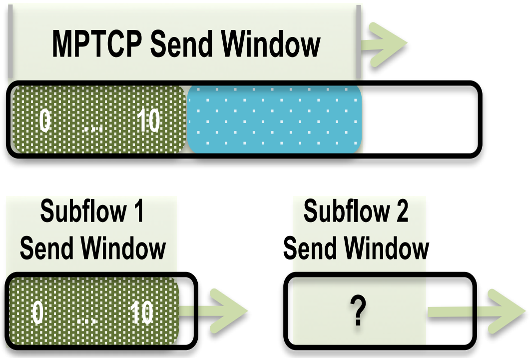
Blocking Estimation: BLEST

- Segments 0 …10 are in flight on subflow 1, the one with lowest delay



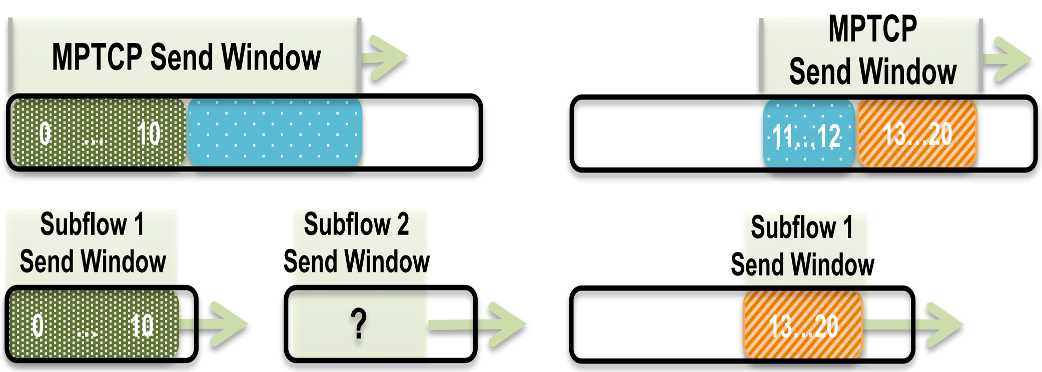
Blocking Estimation: BLEST

- Segments 0 …10 are in flight on subflow 1, the one with lowest delay
- It is uncertain how many segments to sent on subflow 2, which has a higher delay



Blocking Estimation: BLEST

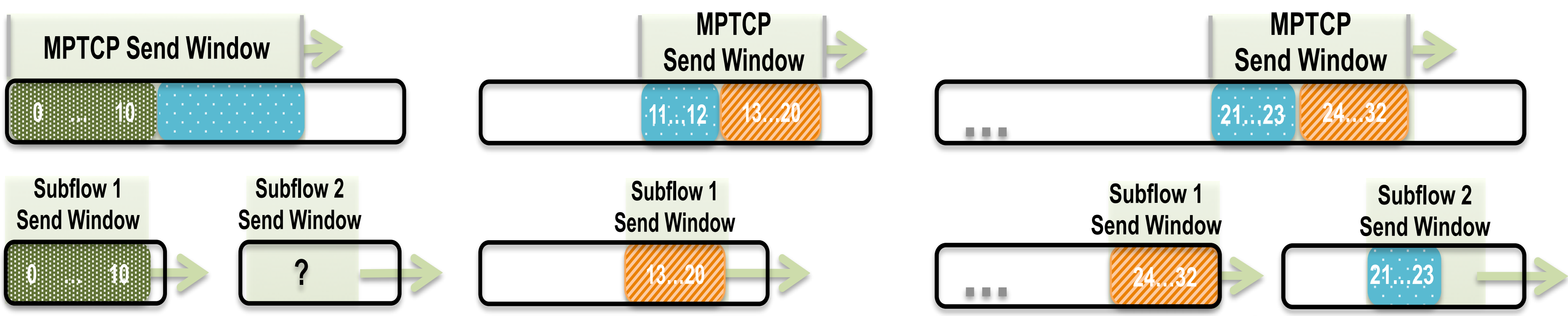
- Segments 0 …10 are in flight on subflow 1, the one with lowest delay
- It is uncertain how many segments to sent on subflow 2, which has a higher delay
- While subflow 2's window could accommodate more data, only segments 11…12 are allocated, due to BLEST's blocking prediction*



* Here, minRTT would allocate as much data as fits into subflow 2's window given its CWND

Blocking Estimation: BLEST

- Segments 0 …10 are in flight on subflow 1, the one with lowest delay
- It is uncertain how many segments to sent on subflow 2, which has a higher delay
- While subflow 2's window could accommodate more data, only segments 11…12 are allocated, due to BLEST's blocking prediction*
- Then, subflow 1 can advance with segments 13…20, because 0…10 were acked



* Here, minRTT would allocate as much data as fits into subflow 2's window given its CWND

Blocking Estimation: BLEST

HoL-blocking would occur if the fast subflow (F) cannot send due to lack of space in the send window because of the slow subflow (S)

- Therefore, BLEST estimates the amount of data X that will be sent on F during RTT_S , and check whether this fits into MPTCP's send window

Blocking Estimation: BLEST

HoL-blocking would occur if the fast subflow (F) cannot send due to lack of space in the send window because of the slow subflow (S)

- Therefore, BLEST estimates the amount of data X that will be sent on F during RTT_S , and check whether this fits into MPTCP's send window

$$rtts = RTT_S / RTT_F$$
$$X = MSS_F \cdot (CWND + (rtts - 1)/2) \cdot rtts$$

- If $X \times \lambda > |M| - MSS_S \cdot (inflight_S + 1)$ the next segment will not be sent on S. Instead, the scheduler will wait for F to become available
  - While minRTT will always opt to use an available subflow, BLEST is able to skip a subflow, reducing the number of retransmissions triggered

Blocking Estimation: BLEST

HoL-blocking would occur if the fast subflow (F) cannot send due to lack of space in the send window because of the slow subflow (S)

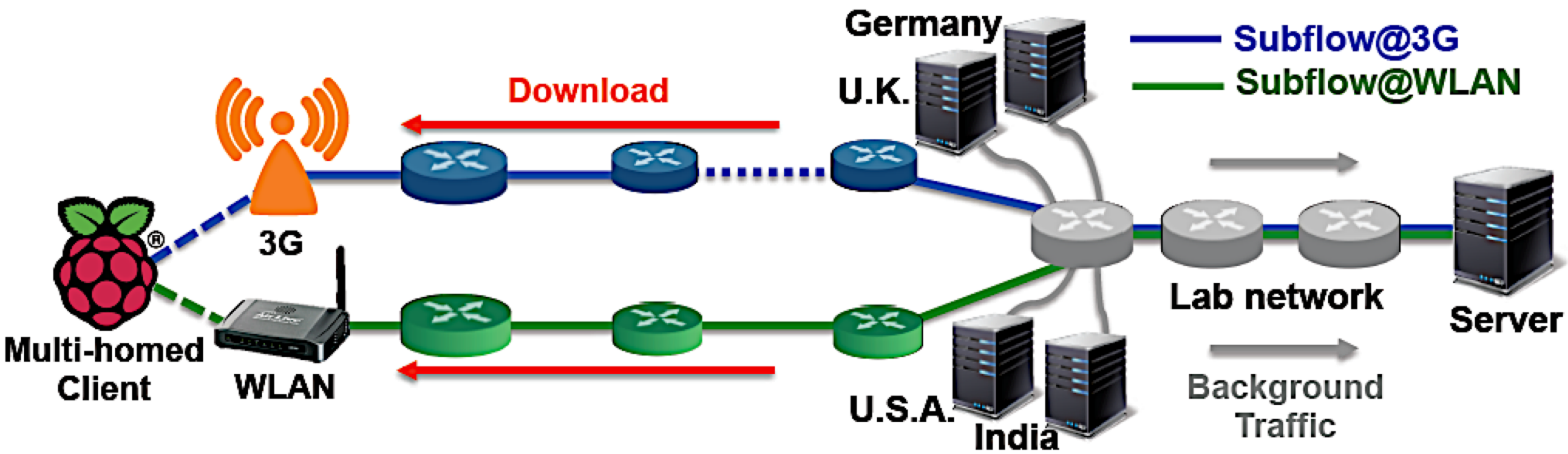
- Therefore, BLEST estimates the amount of data X that will be sent on F during RTT_S , and check whether this fits into MPTCP's send window

$$rtts = RTT_S / RTT_F$$
$$X = MSS_F \cdot (CWND + (rtts - 1)/2) \cdot rtts$$

- If $X \times \lambda > |M| - MSS_S \cdot (inflight_S + 1)$ the next segment will not be sent on S. Instead, the scheduler will wait for F to become available
  - While minRTT will always opt to use an available subflow, BLEST is able to skip a subflow, reducing the number of retransmissions triggered

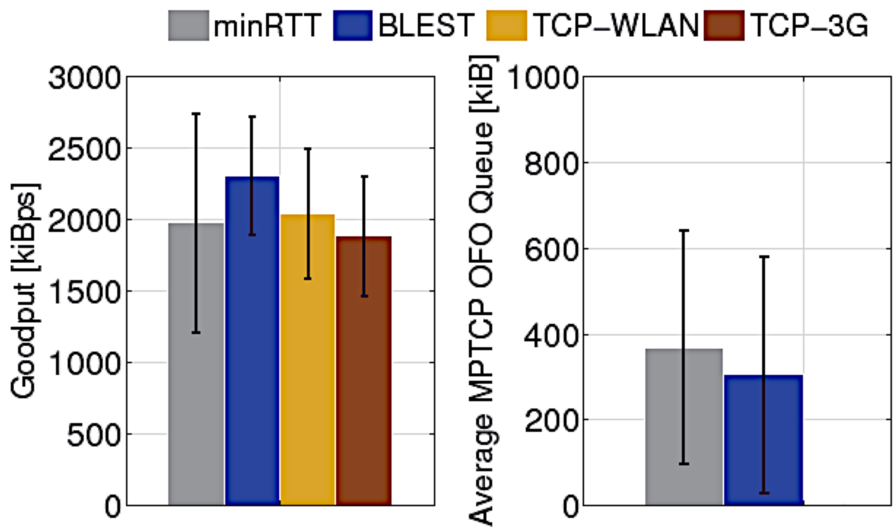
- When the scheduler calculates X inaccurately, the λ is introduced to correct the estimates and it is dynamically adapted: λ is initially set to 1

Real-Network Evaluation



- vms from five commercial cloud service providers (2x in Europe, 1x in North America and 2x in Asia) connected via 100 Mbps links
- Lab network connected to a research gigabit network
- Background traffic composed of UDP on/off and TCP flows against the server machine

Multipath TCP: Bulk Traffic



(a) 3G-WLAN

Metric: Goodput, retransmissions by *penalisation and retransmission* and average OFO queue size

	Scheduler	Traffic	Retrans. Packets
3G+WLAN	minRTT	Bulk	33.42
	BLEST		21.3

Multipath TCP: Web Transfers

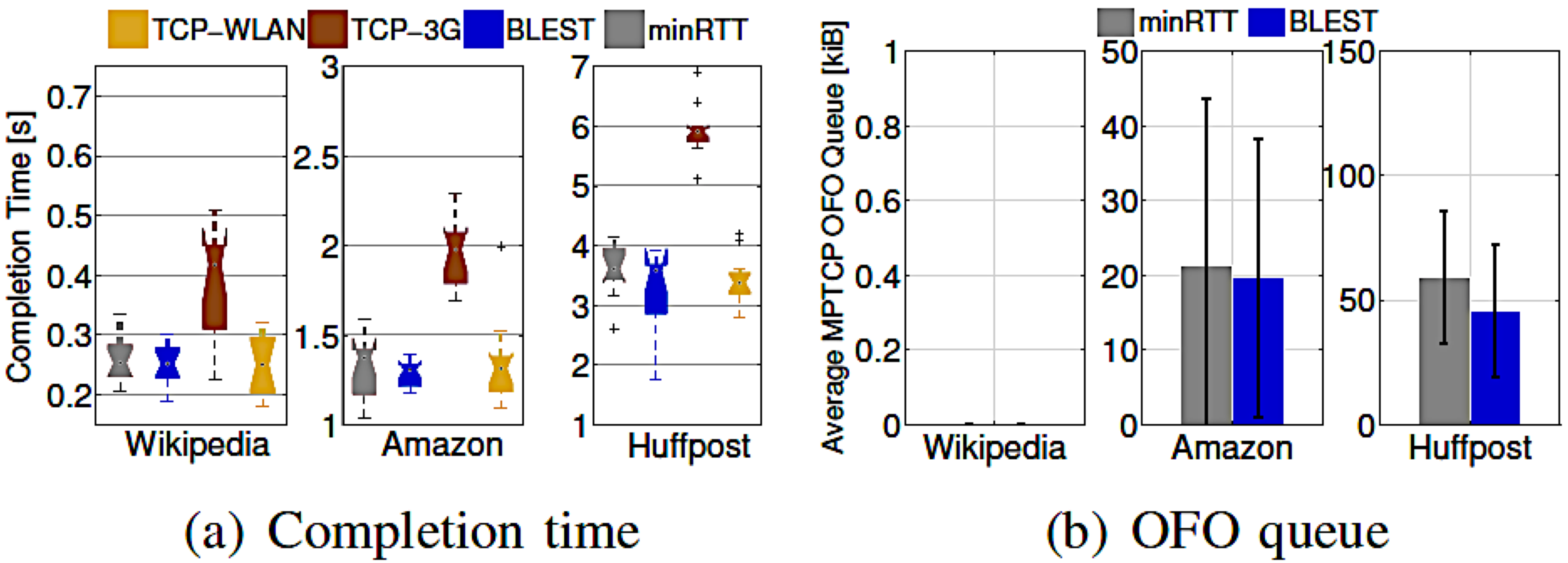


Figure 14. Web Traffic in **3G-WLAN** in real experiments

MPTCP's performance with BLEST is closer to that of the WLAN path, performing only 3% worse than TCP on the best path (WLAN).

Conclusion

Path heterogeneity is rather the rule than the exception with MPTCP

- Path heterogeneity results in HoL-blocking at the receiver undermining MPTCP's overall performance
- To overcome path heterogeneity, MPTCP follows a *reactive* approach, the *penalisation and retransmission* mechanism

Conclusion

Path heterogeneity is rather the rule than the exception with MPTCP

- Path heterogeneity results in HoL-blocking at the receiver undermining MPTCP's overall performance
- To overcome path heterogeneity, MPTCP follows a *reactive* approach, the *penalisation and retransmission* mechanism

We highlighted the limitations of such reactive approach systematically evaluating different applications in both WLAN+WLAN and 3G+WLAN scenarios with emulations and real-network experiments

- We implemented and compared minRTT, DAPS and OTIAS
- With BLEST applications increased goodput, lowered completion time, reduced retransmissions and reduced receiver buffer size

Questions?

The implementation is open source and available for other researchers:

https://bitbucket.org/blest_mptcp/nicta_mptcp

http://ferlin.io --- ferlin@simula.no

Backup Slides

DAPS

Algorithm 1 DAPS [6]

$S_{max} \leftarrow 0$
for $P_i \in \{P_1, P_2, ..., P_n\}$ **do**
  $SEQ_{P_i} \leftarrow InitializeVector()$
end for
for $P_i \in \{O_1, O_2, ..., O_{\sum_i \in 1,2,...,n \frac{lcm(D_i)}{D_i}}\}$ **do**
  $SEQ_{P_i} \leftarrow Append(SEQ_{P_i}[S_{max} + 1, S_{max} + C_i]$
end for
$t \leftarrow 0$
while $t < \mathrm{lcm}(D_i \in \{D_1, D_2, ..., D_n\})$ **do**
  for $P_i \in \{P_1, P_2, ..., P_n\}$ **do**
    if $t \equiv 0 \ (mod\ D_i)$ **then**
      $Transmit(P_i, SEQ_{P_i}[\frac{t}{D_i}])$
      $S_{max} \leftarrow S_{max} + C_i$
    end if
  end for
  $t \leftarrow t + 1$
end while

Where:

- $\{D_1, D_2, ..., D_n\}$ paths' respective forward delays
- $\{P_1, P_2, ..., P_n\}$ is the set of paths
- SEQ_{P_i} packets' seq. numbers to be transmitted on P_i

A schedule is created to span the least common multiple (LCM) of the forward delays

DAPS creates a schedule for the distribution of future segments for a scheduling run and follows this schedule until it is completed

OTIAS

Algorithm 2 OTIAS [8]

for each available subflow j **do**
  $\texttt{pkt\_can\_be\_sent}_j = cwnd_j - unacked_j$
  $\texttt{RTT\_to\_wait}_i^j = \left\lfloor \frac{\texttt{not\_yet\_sent}_j - \texttt{pkt\_can\_be\_sent}_j}{cwnd_j} \right\rfloor$
  $T_i^j = (\texttt{RTT\_to\_wait}_i^j + 0.5) \times srtt_j$
  if $T_i^j < min_T$ **then**
    $min_T = T_i^j$
    $\texttt{selected\_subflow} = j$
  end if
end for

When asked to schedule a new segment, OTIAS estimates its arrival time, and chooses the subflow with the shortest time.

- OTIAS decides which subflow to use on a per-packet basis, however, it builds up queues in the subflows, which can be detrimental
- If a segment that had been sent is blocking the connection, queued packets will linger at the sender more than assumed, disturbing the created schedule
- No retransmission approach: If a send queue exists for a subflow, as that segment would have to wait in the queue before being retransmitted