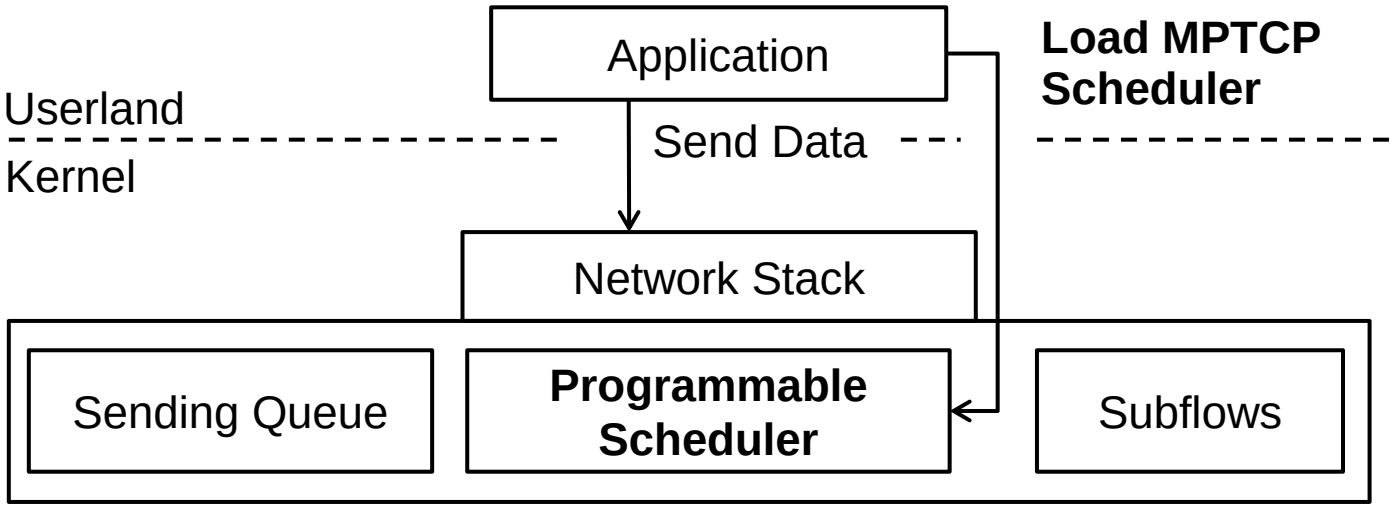
A Programming Model for Application-defined Multipath TCP Scheduling

ACM/IFIP/USENIX Middleware 2017
Alexander Frömmgen

TECHNISCHE UNIVERSITÄT DARMSTADT

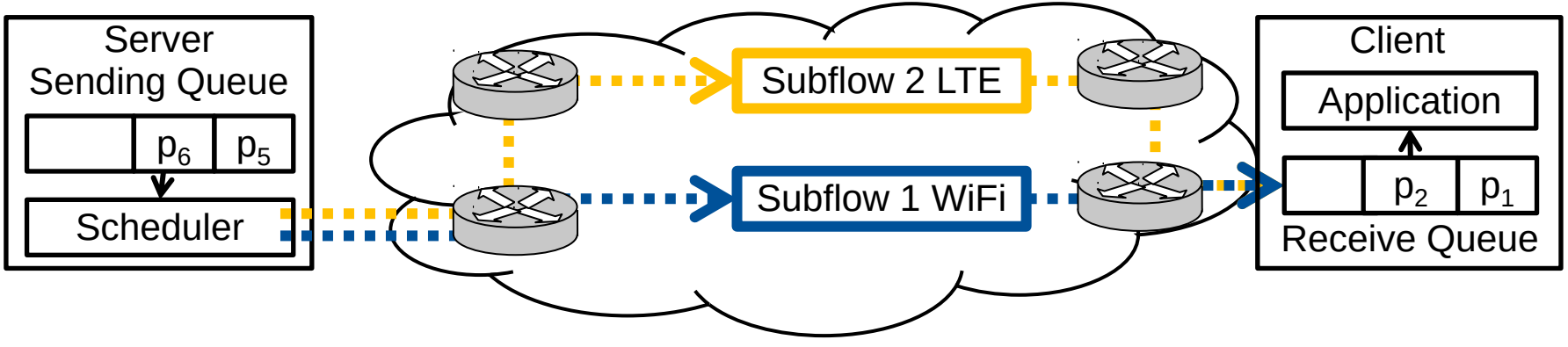
KOM – Multimedia Communications Lab

DFG Collaborative Research Centre 1053 – MAKI
Multi Mechanism Adaptation for the Future Internet



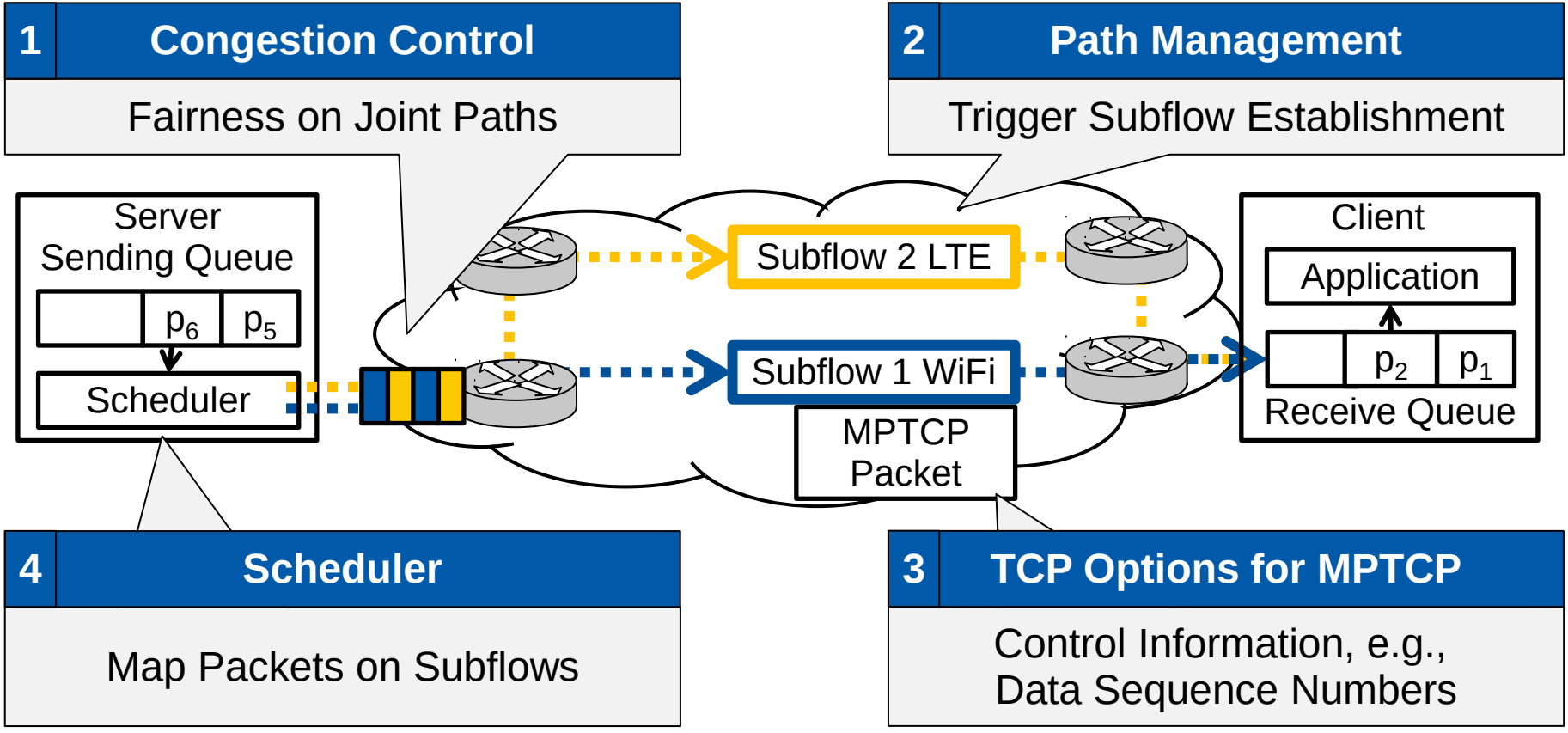
Alexander Frömmgen
Amr Rizk
Tobias Erbshäußer
Max Weller
Boris Koldehofe
Alejandro Buchmann
Ralf Steinmetz

Multipath TCP in a Nutshell

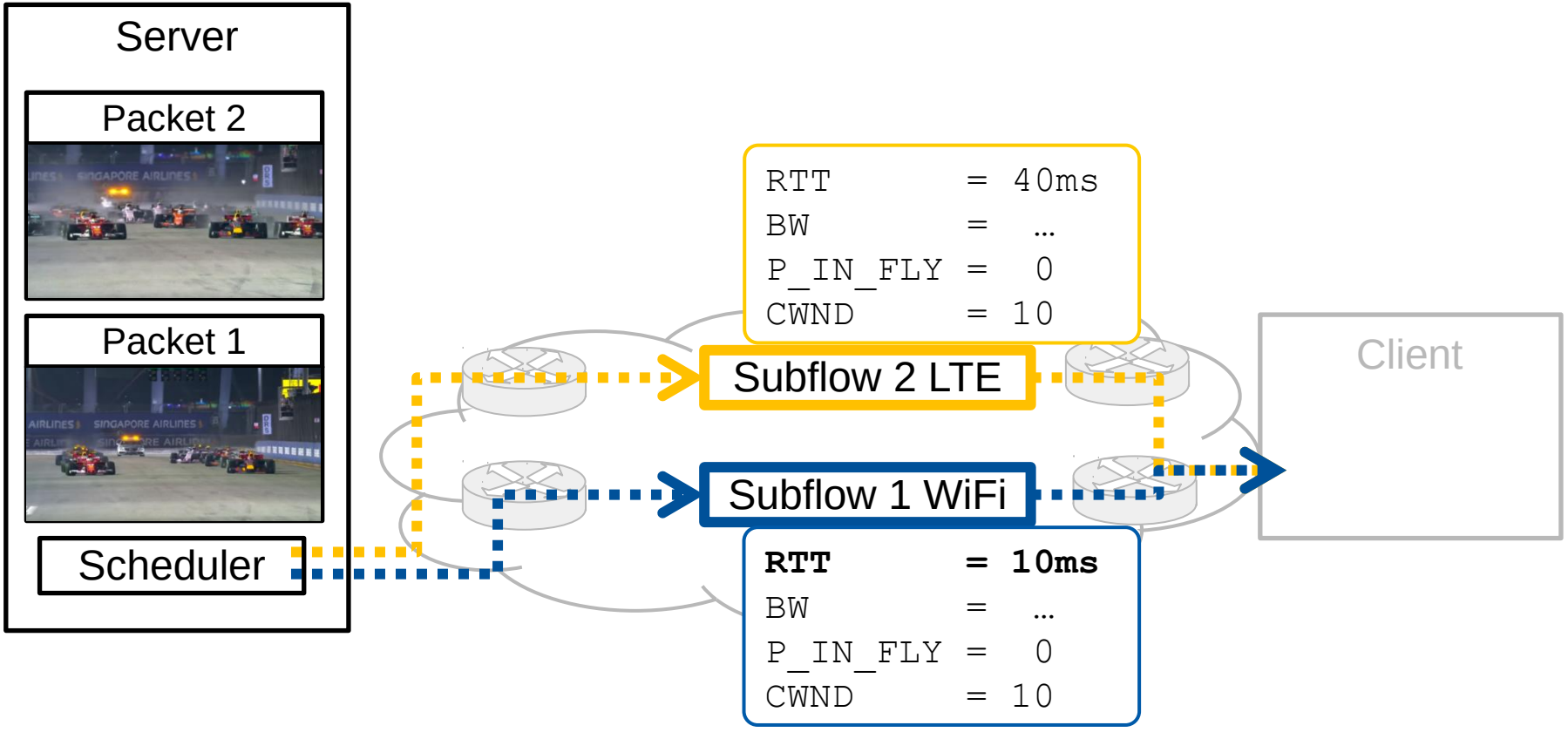


Multipath TCP in a Nutshell

TECHNISCHE UNIVERSITÄT DARMSTADT



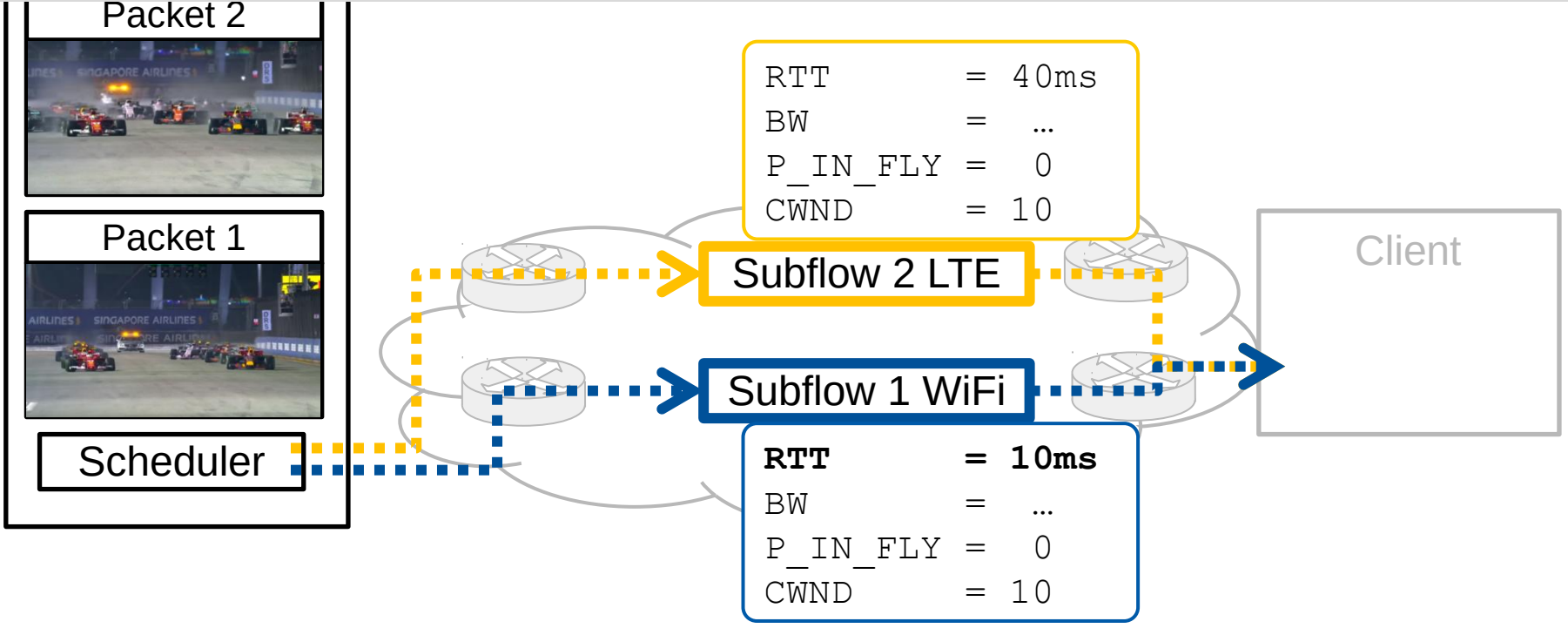
Multipath TCP Scheduling



Multipath TCP Scheduling

TECHNISCHE UNIVERSITÄT DARMSTADT

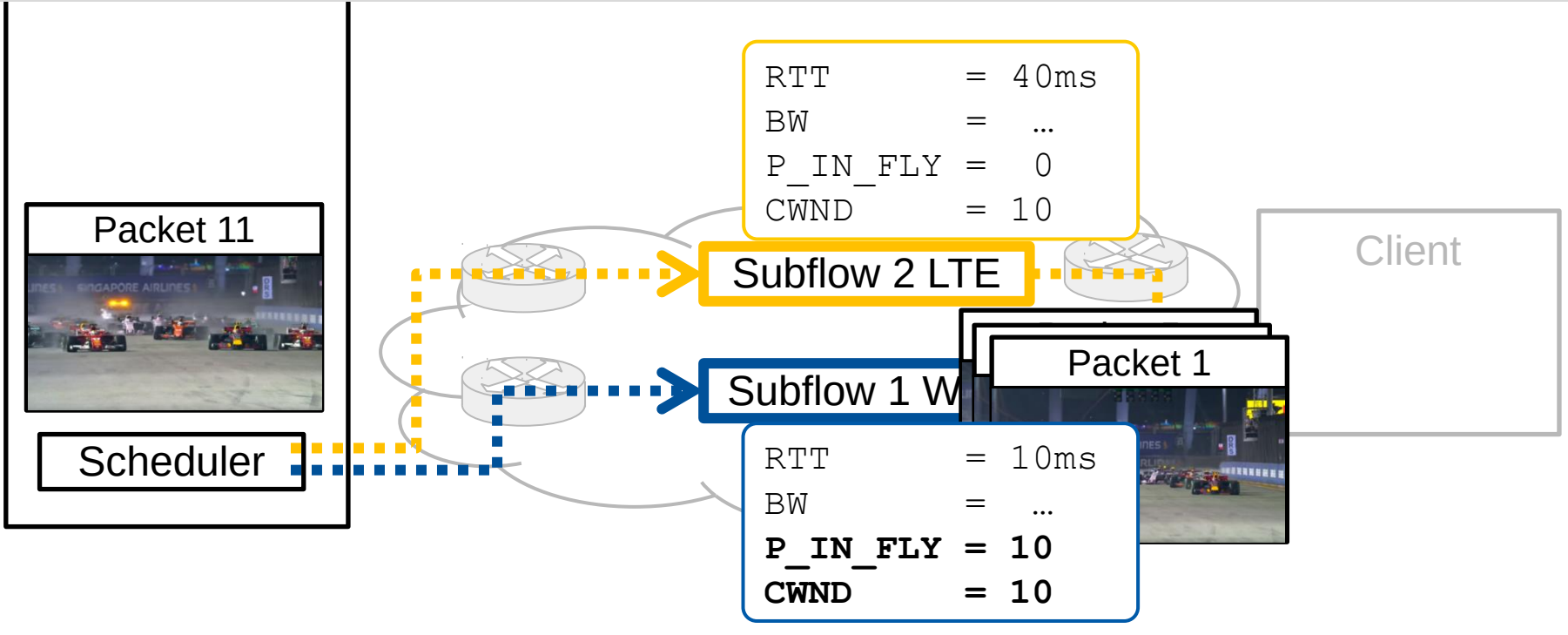
Intuition: Schedule on Subflow with minimum round-trip time (RTT).



Multipath TCP Scheduling

TECHNISCHE UNIVERSITÄT DARMSTADT

Schedule on Subflow with minimum round-trip time (RTT).

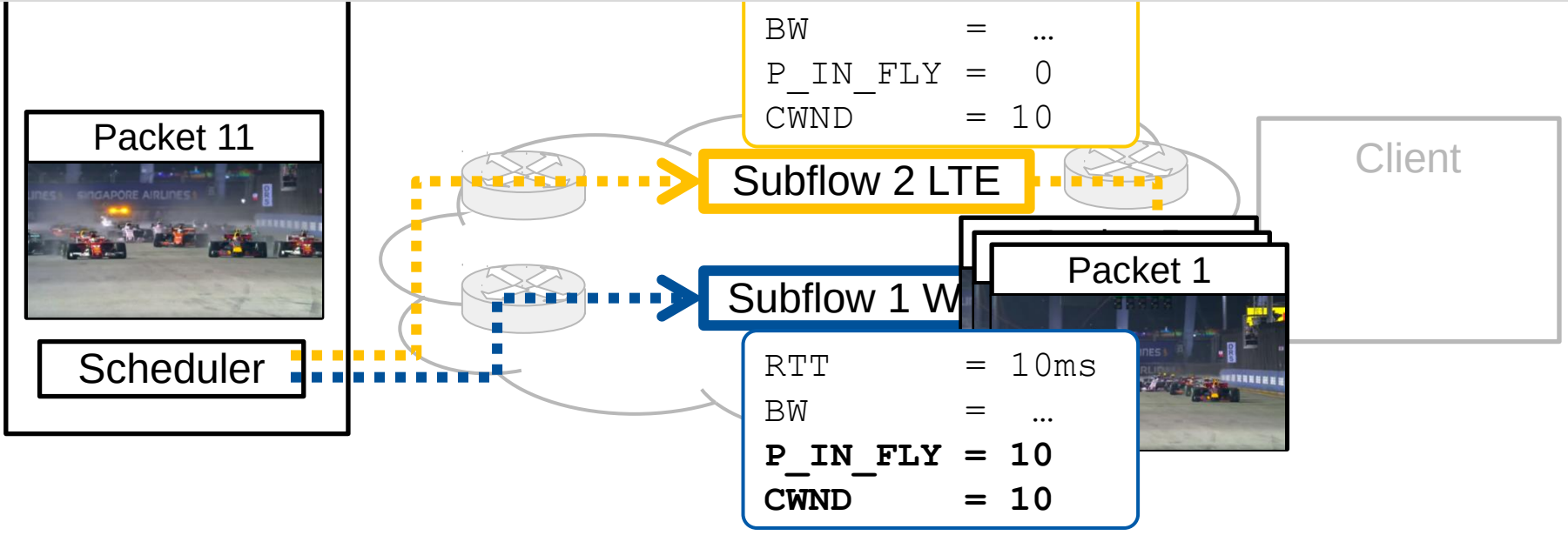


Multipath TCP Scheduling

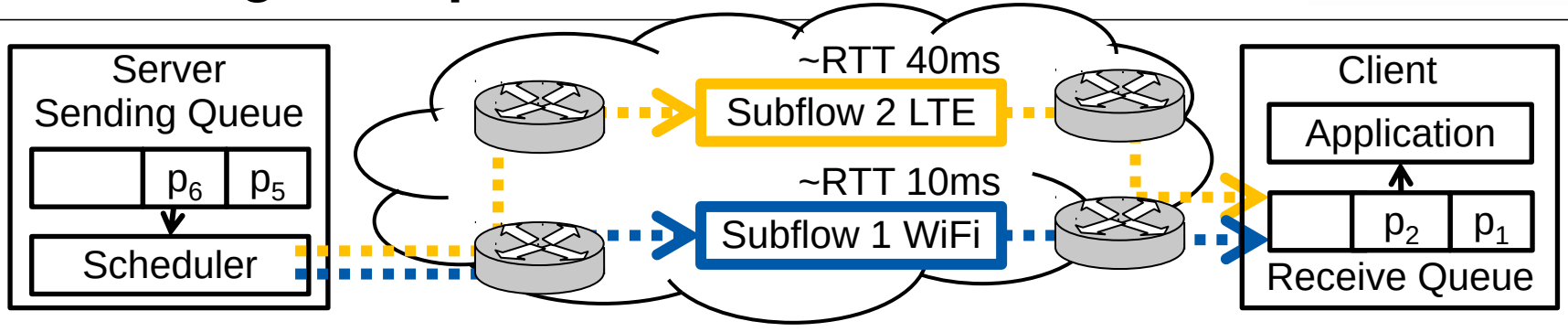
TECHNISCHE UNIVERSITÄT DARMSTADT

Schedule on Subflow with minimum round-trip time (RTT).

which is not saturated (Congestion window larger than packets in flight).

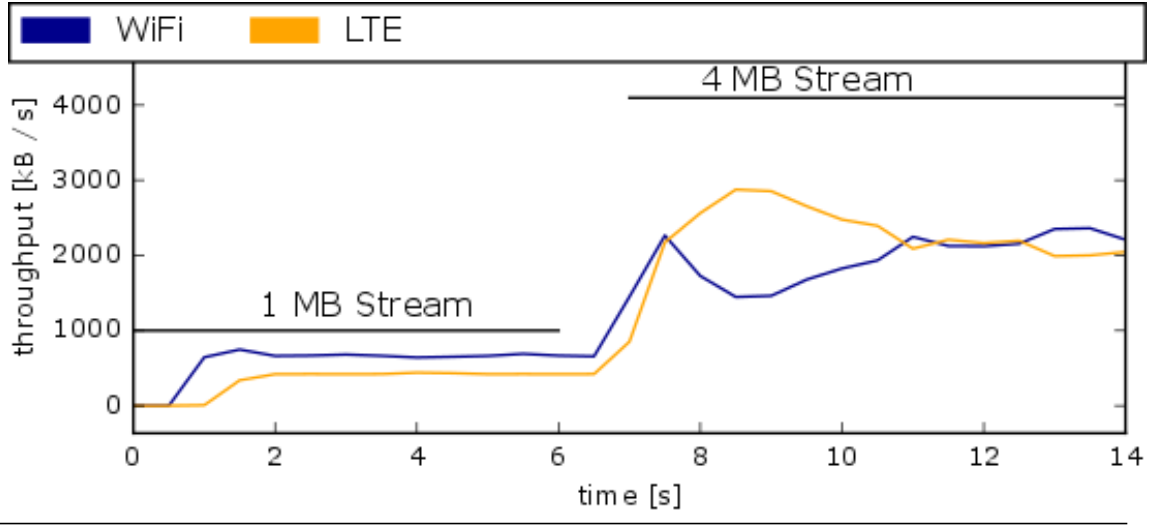


Multipath TCP Scheduling
Motivating Example

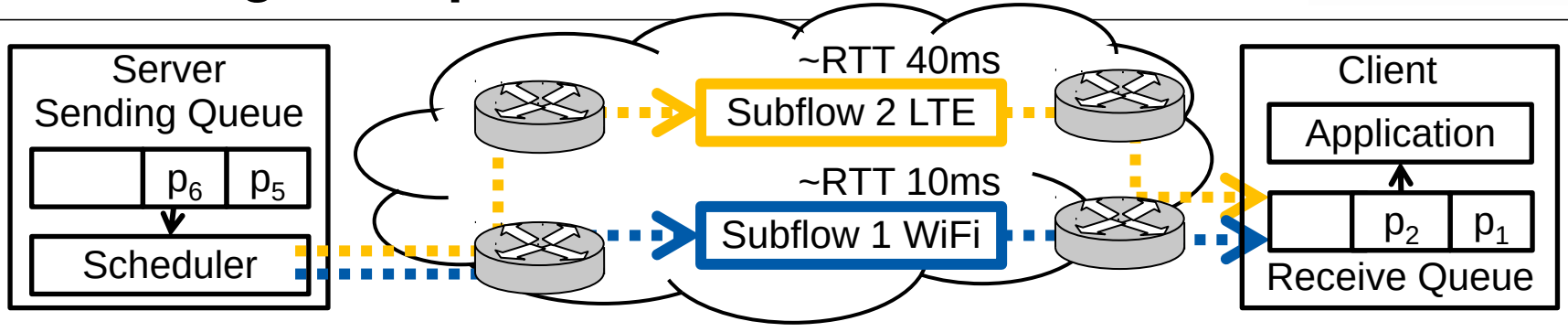


Constant Bitrate Stream

Home Network WiFi and LTE Germany

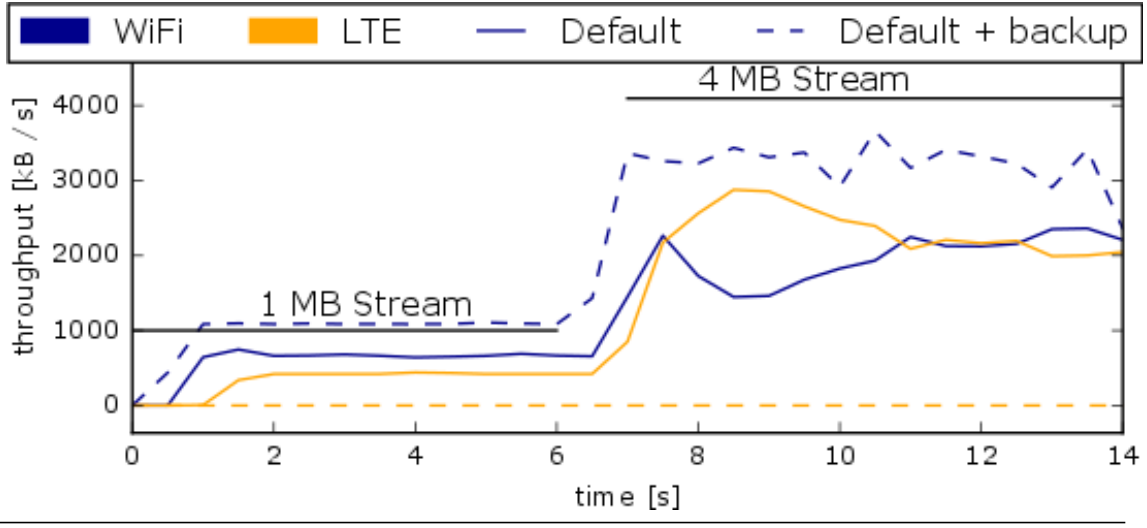


Multipath TCP Scheduling
Motivating Example



Constant Bitrate Stream

Home Network WiFi and LTE Germany



Multipath TCP Scheduler Overview

TECHNISCHE UNIVERSITÄT DARMSTADT

Name	Domain	Pref.	Available in Linux Implementation
Min RTT (default)	General Purpose	Binary	✓
Round-Robin	Academic	Binary	✓
Redundant	Thin Flows	Binary	✓
Compensate Loss	Short Datacenter Flows	?	No
Video and Energy		?	No
DASH Video		Yes	No
...			

Multipath TCP Scheduler Overview
Great Concepts without MPTCP Implementation

TECHNISCHE UNIVERSITÄT DARMSTADT

The reference FIFO scheduler. We compare our cross-layer scheduler and the optimal solution to the *reference FIFO* scheduler, which mimics the current implementation of MPTCP without any cross-layer scheduler. The video

The cross-layer scheduler can be implemented either at the transport layer or at the application layer. In the following of the paper, we consider an implementation at the application level because it is more convenient. Here are some details.

X. Corbillon, R. Aparicio-Pardo, N. Kuhn, G. Texier, and G. Simon. Cross-layer scheduler for video streaming over MPTCP. In ACM MMSys, 2016

independent subflows. This approach provides tran
applications, at the cost of requiring kernel upgra
space, RepFlow can be incorporated into MPTCP
multi-path support.

It is evident that RepFlow lends itself to many
tation choices. No matter the details, it is crucia
path diversity is utilized, i.e. the five-tuples of
and replicated flow have to be different (assumin

H. Xu and B. Li. RepFlow: Minimizing flow completio
replicated flows in data centers.
In IEEE INFOCOM, pages 1581–1589, 201

In future work, we plan to implement DAPS in FreeBSD's CMT-SCTP stack and in Linux implementation of MPTCP to evaluate the performance gain in realistic network conditions and address the related practical issues.

N. Kuhn, E. Lochin, A. Mifdaoui, G. Sarwar, O. Mehani, and R. Boreli. DAPS: intelligent delay-aware packet scheduling for multipath transport. In IEEE ICC, 2014.

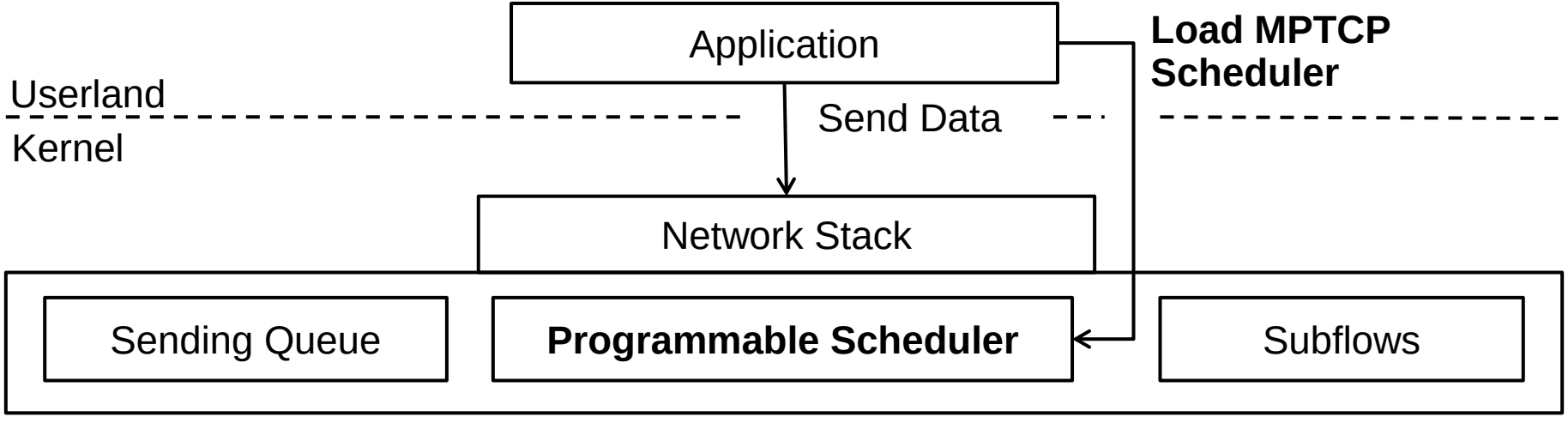
Research Questions

TECHNISCHE UNIVERSITÄT DARMSTADT

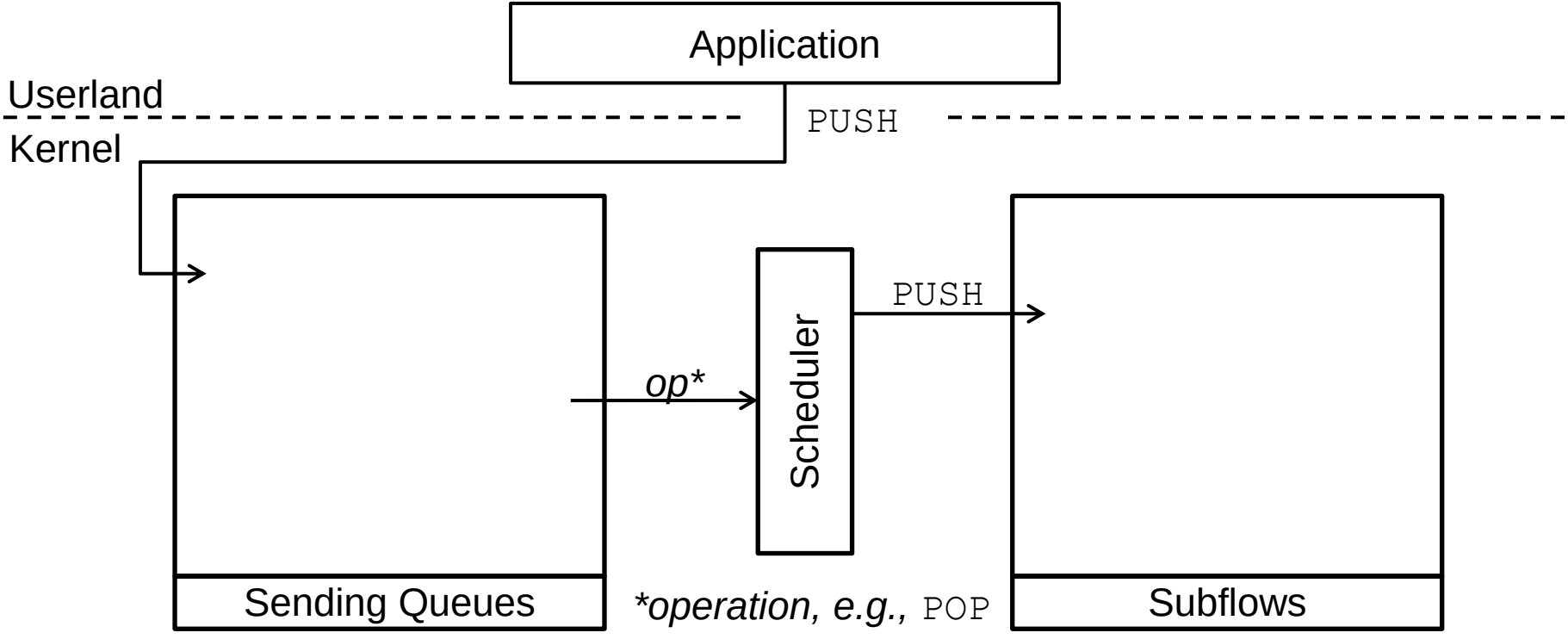
How can we systematically **specify** and **execute** MPTCP schedulers?

How can we **enable application-defined** MPTCP scheduling?

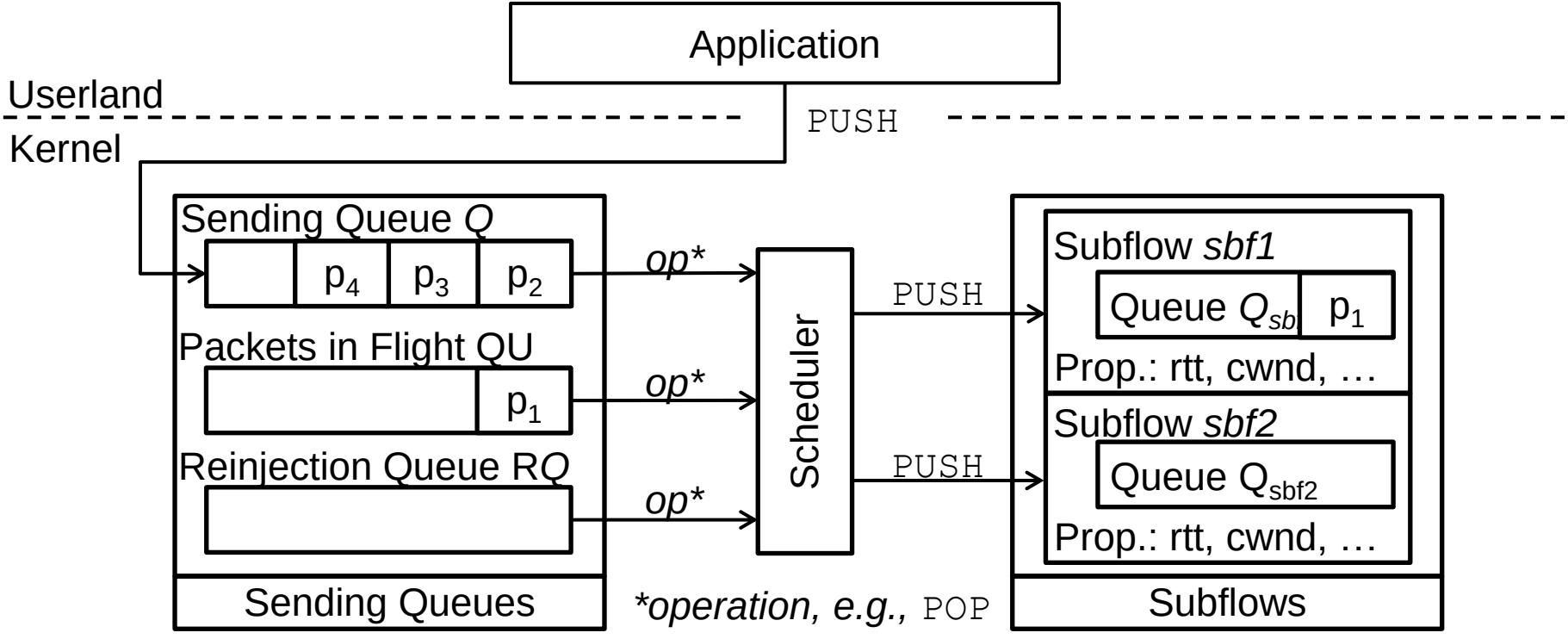
Part I: Towards a Programmable Scheduler in the Network Stack



Model of the Scheduler Environment



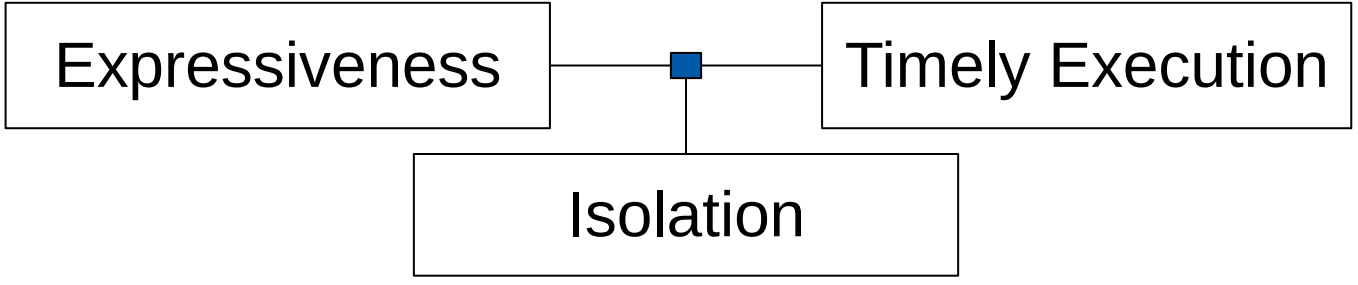
Model of the Scheduler Environment



Specifying Multipath TCP Schedulers

TECHNISCHE UNIVERSITÄT DARMSTADT

Requirements



Design Decisions

1. Modelled Elements as Entities: Set of Subflows with their properties, Queues of Packets with their Properties
2. Declarative Packet and Subflow Selection (Filter, Min, Max)
3. No Recursion, No Functions, Limited Loops
4. Variables with Single Assignment, Implicit and Static Type System
5. No, One, or Multiple Packets per Scheduler Execution

Systematically Specify MPTCP Schedulers Domain Specific Specification Language

TECHNISCHE UNIVERSITÄT DARMSTADT

Example: Preference-aware RTT-sensitive Scheduler

```
1  VAR sbfCandis = SUBFLOWS.FILTER(
2          sbf => sbf.CWND > sbf.SKBS_IN_FLIGHT + sbf.QUEUED
3                 AND !sbf.TSQ_THROTTLED AND !sbf.LOSSY);
4
5  VAR backSbf = sbfCandis.FILTER(
6                sbf => sbf.IS_BACKUP).MIN(sbf => sbf.RTT);
7  VAR nonBackSbf = sbfCandis.FILTER(
8                sbf => !sbf.IS_BACKUP).MIN(sbf => sbf.RTT);
9
10 IF (nonBackSbf.RTT_MS > 100 AND backSbf.RTT_MS < 80) {
11    backSbf.PUSH(Q.POP());
12 } ELSE {
13    nonBackSbf.PUSH(Q.POP());
14 }
```

Systematically Specify MPTCP Schedulers Domain Specific Specification Language

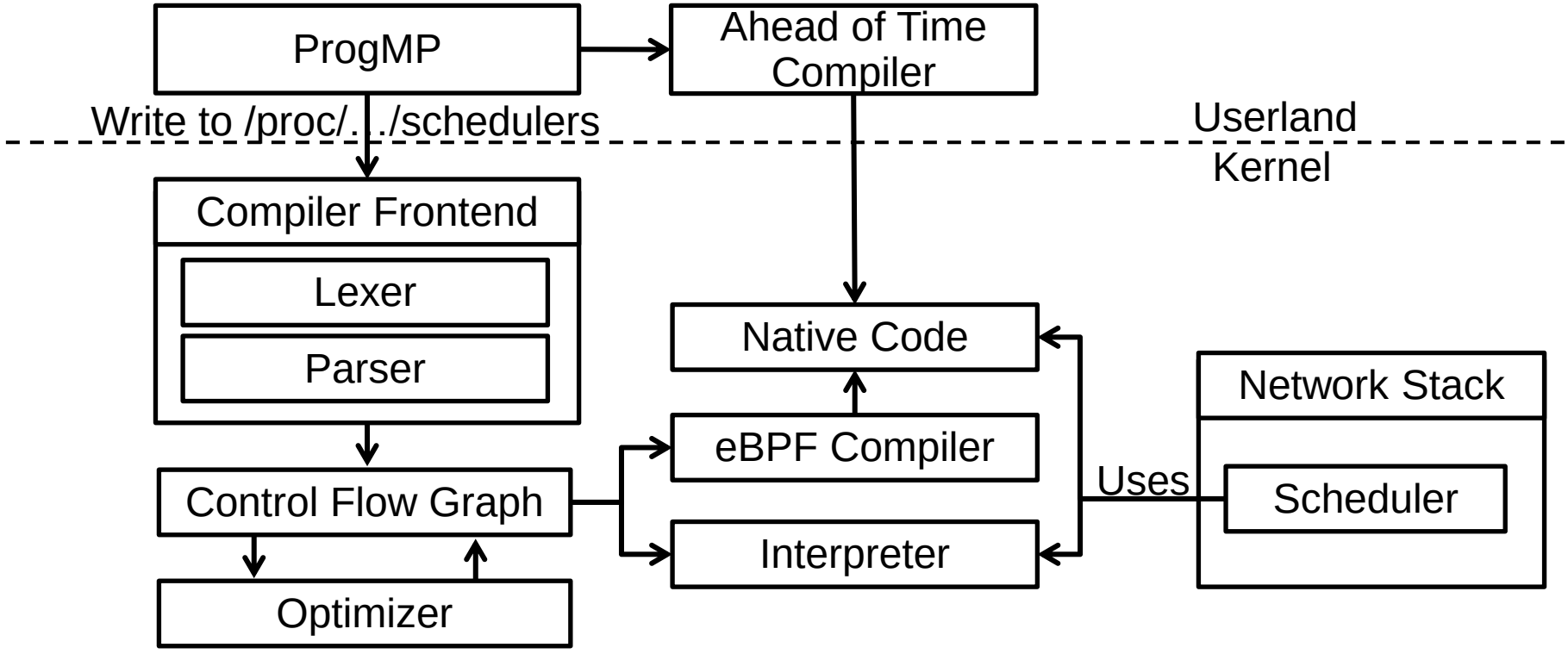
TECHNISCHE UNIVERSITÄT DARMSTADT

Example: Preference-aware RTT-sensitive Scheduler

```
1  VAR sbfCandis = SUBFLOWS.FILTER(
2          sbf => sbf.CWND > sbf.SKBS_IN_FLIGHT + sbf.QUEUED
3                 AND !sbf.TSQ_THROTTLED AND !sbf.LOSSY);
4
5  VAR backSbf = sbfCandis.FILTER(
6                sbf => sbf.IS_BACKUP).MIN(sbf => sbf.RTT);
7  VAR nonBackSbf = sbfCandis.FILTER(
8                sbf => !sbf.IS_BACKUP).MIN(sbf => sbf.RTT);
9
10 IF (nonBackSbf.RTT_MS > R1 AND backSbf.RTT_MS < R2) {
11    backSbf.PUSH(Q.POP());
12 } ELSE {
13    nonBackSbf.PUSH(Q.POP());
14 }
```

Systematically Specify and *Execute* MPTCP Schedulers

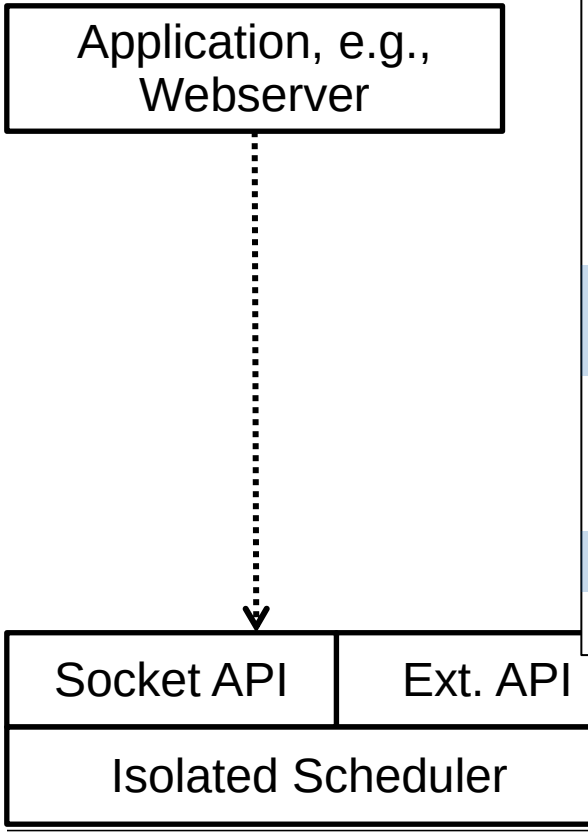
TECHNISCHE UNIVERSITÄT DARMSTADT



Specified schedulers are executable in the Linux Kernel

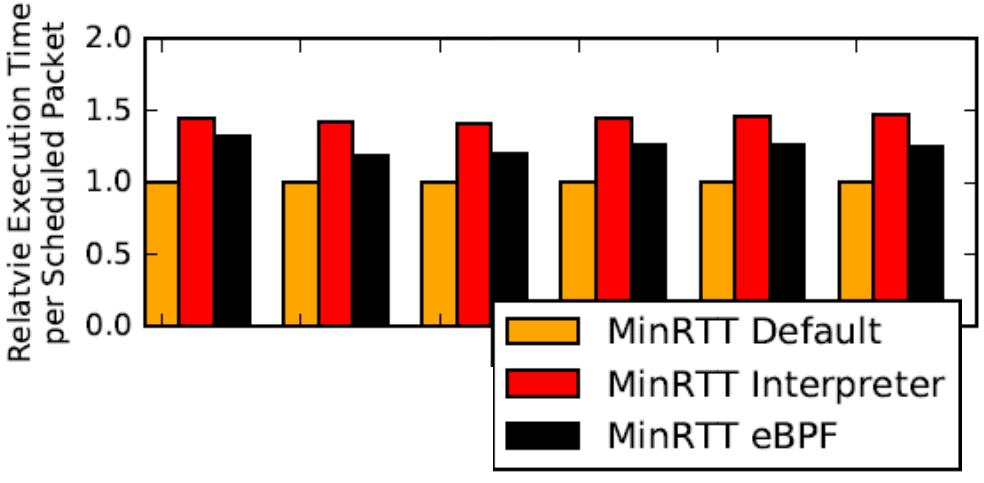
Application-aware Scheduling in Multi-Tenancy Cloud Environments

TECHNISCHE UNIVERSITÄT DARMSTADT

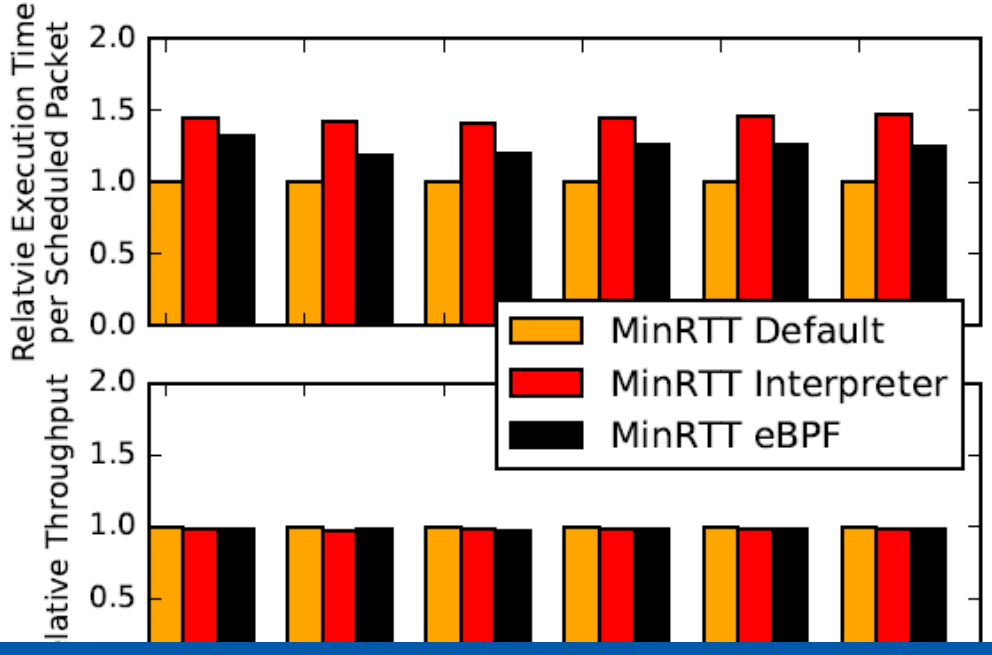



```
 1  import socket
 2  from progmp import ProgMp
 3  s = socket.socket(socket.AF_INET, socket.
 4  s.connect(("10.0.0.2", 8080))
 5
 6  try:
 7    ProgMp.loadScheduler("python_api_exampl
 8    ProgMp.setScheduler(s, "python_api_exam
 9  except:
10    print "Scheduler loading error."
11
12  ProgMp.setRegister(s, Progmp.R1(), 50)
13  s.send("Multipath is awesome!")
```

Abstraction vs. Overhead



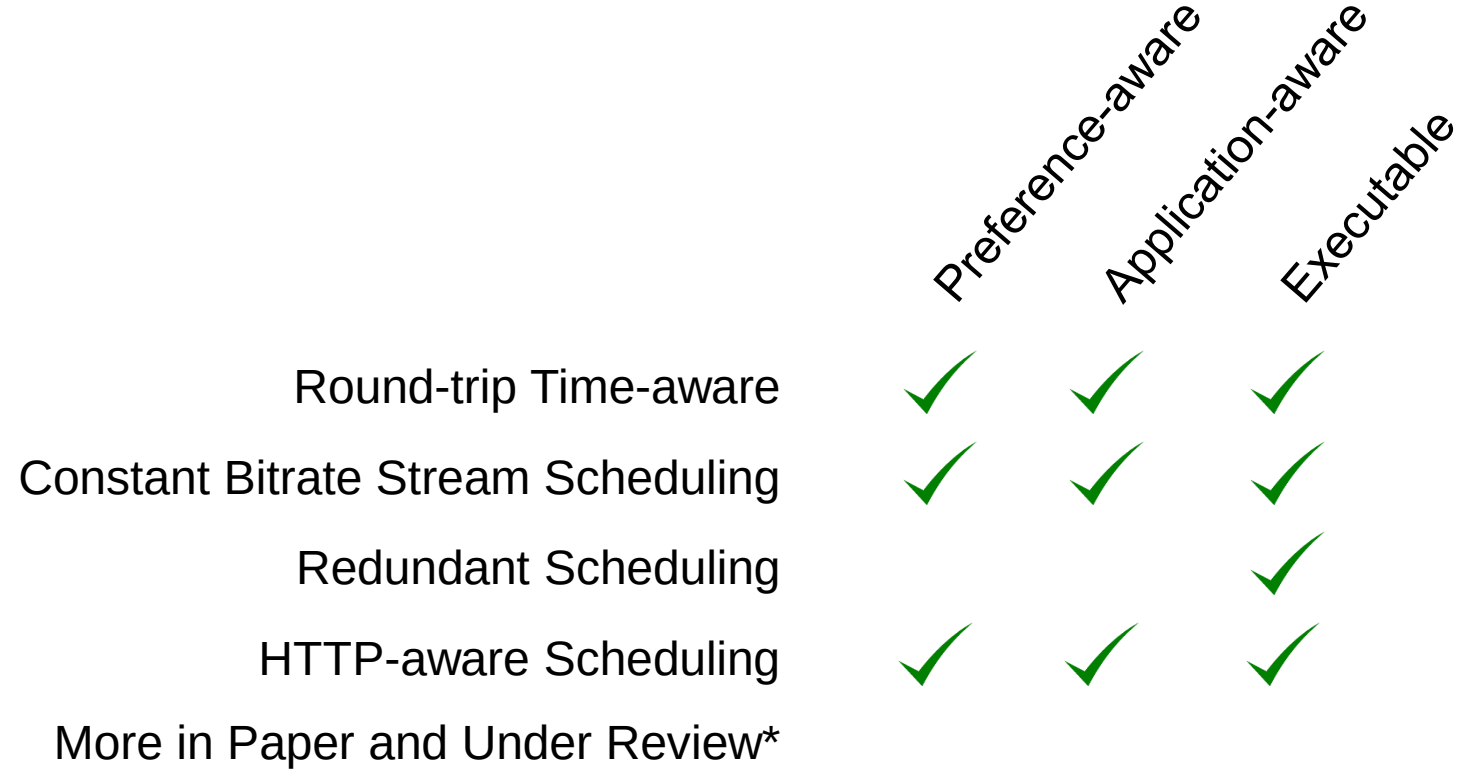
Abstraction vs. Overhead



The runtime environment induces a small overhead, which is acceptable for most application scenarios.

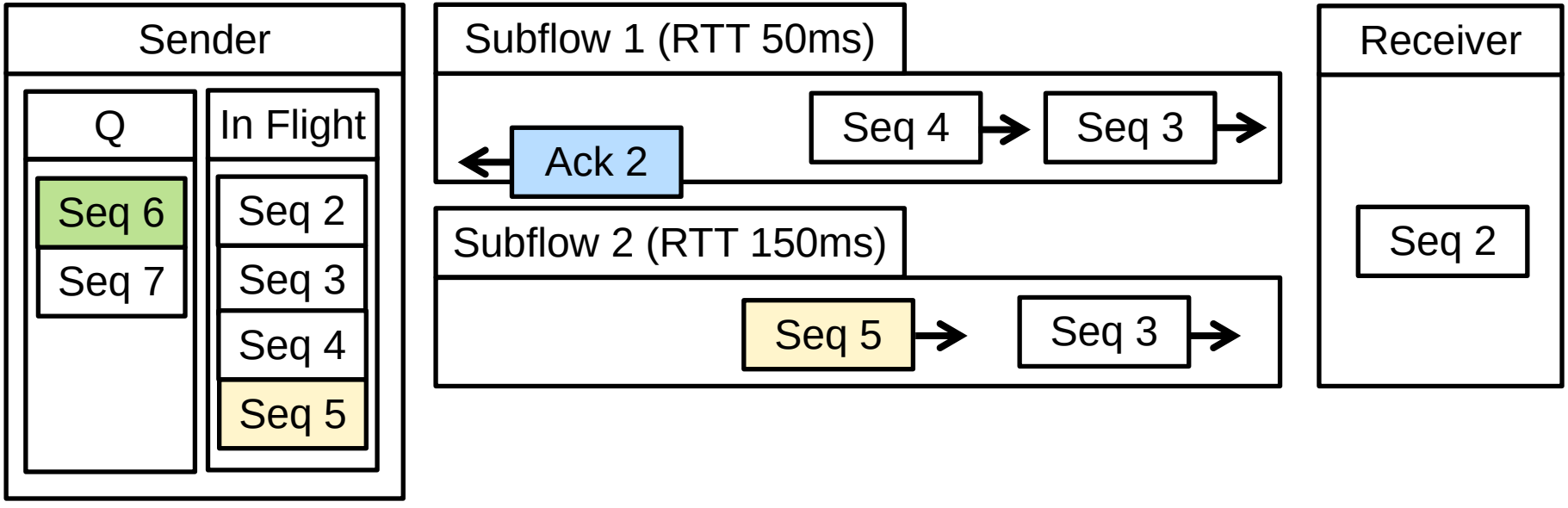
Part II:
Design of Novel Multipath TCP Schedulers

TECHNISCHE UNIVERSITÄT DARMSTADT



*** Multipath TCP Scheduling for Thin Streams: Active Probing and One-way Delay-awarness** by Alexander Frömmgen, Jens Heuschkel and Boris Koldehofe, IEE ICC 2018 (to appear).

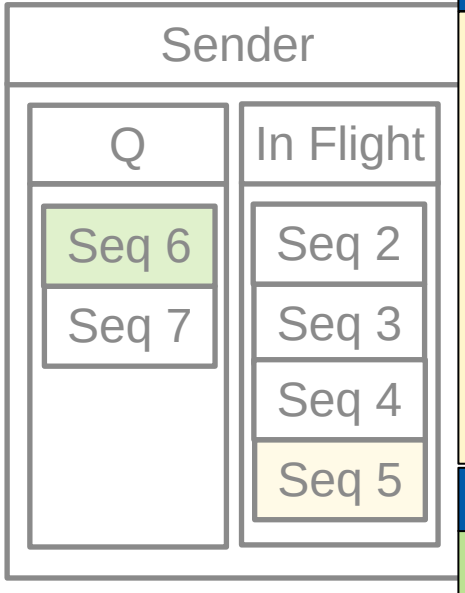
A Close Look at Redundant Schedulers



Should we send the fresh packet Seq 6 or the old packet Seq 5 when the acknowledgemen Ack 2 arrives at the sender?

A Close Look at Redundant Schedulers

TECHNISCHE UNIVERSITÄT DARMSTADT



Should we send the

Prefer Per Subflow Fresh Packets

```
1 FOREACH(VAR sbf IN sbfCandidates) {
2   VAR skb = QU.FILTER(s => !s.SENT_ON(sbf).TOP
3   IF(skb != NULL) {
4     sbf.PUSH(skb);
5   } ELSE {
6     sbf.PUSH(Q.POP());
7   }
8 }
```

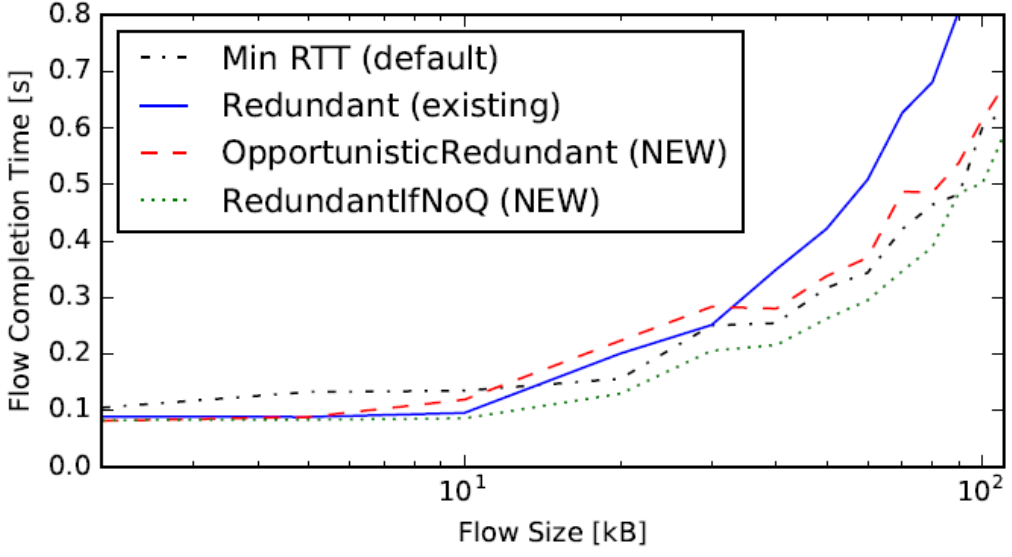
Prefer Global Fresh Packets

```
1 IF(!sbfCandidates.EMPTY) {
2   FOREACH(VAR sbf IN sbfCandidates) {
3     sbf.PUSH(Q.TOP);
```

ProgMP enables rapid specification and evaluation of schedulers.

A Close Look at Redundant Schedulers

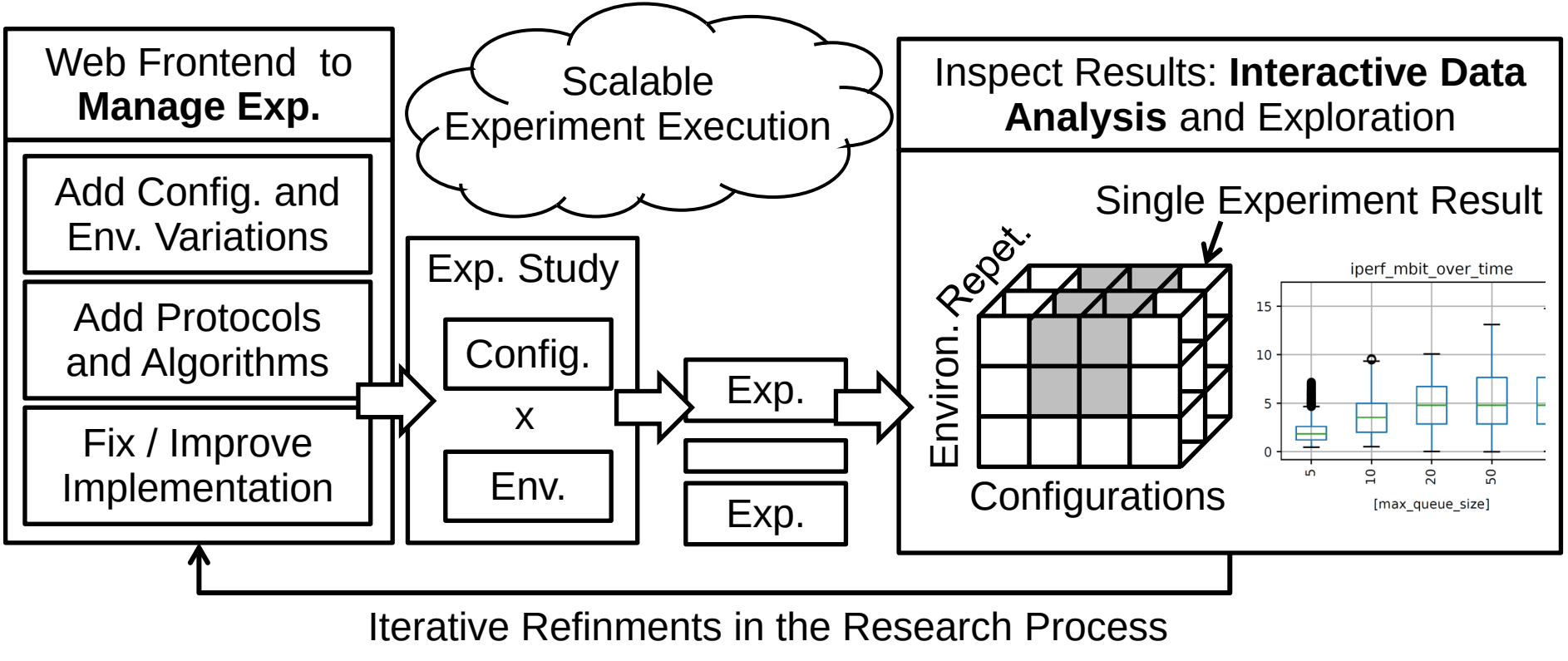
TECHNISCHE UNIVERSITÄT DARMSTADT



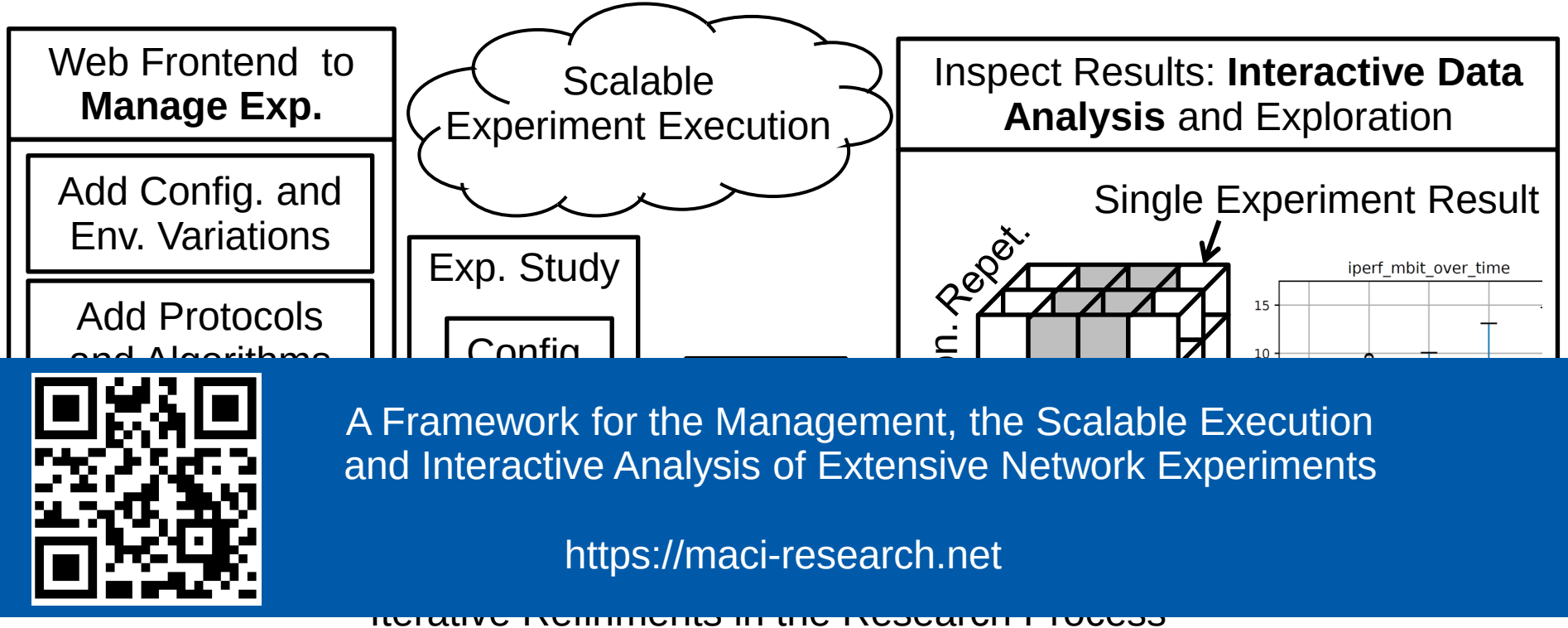
Better

ProgMP enables novel redundant schedulers,
which outperform established approaches.

Side Note: How can we systematically compare and evaluate scheduler design decisions?



Side Note: How can we systematically compare and evaluate scheduler design decisions?

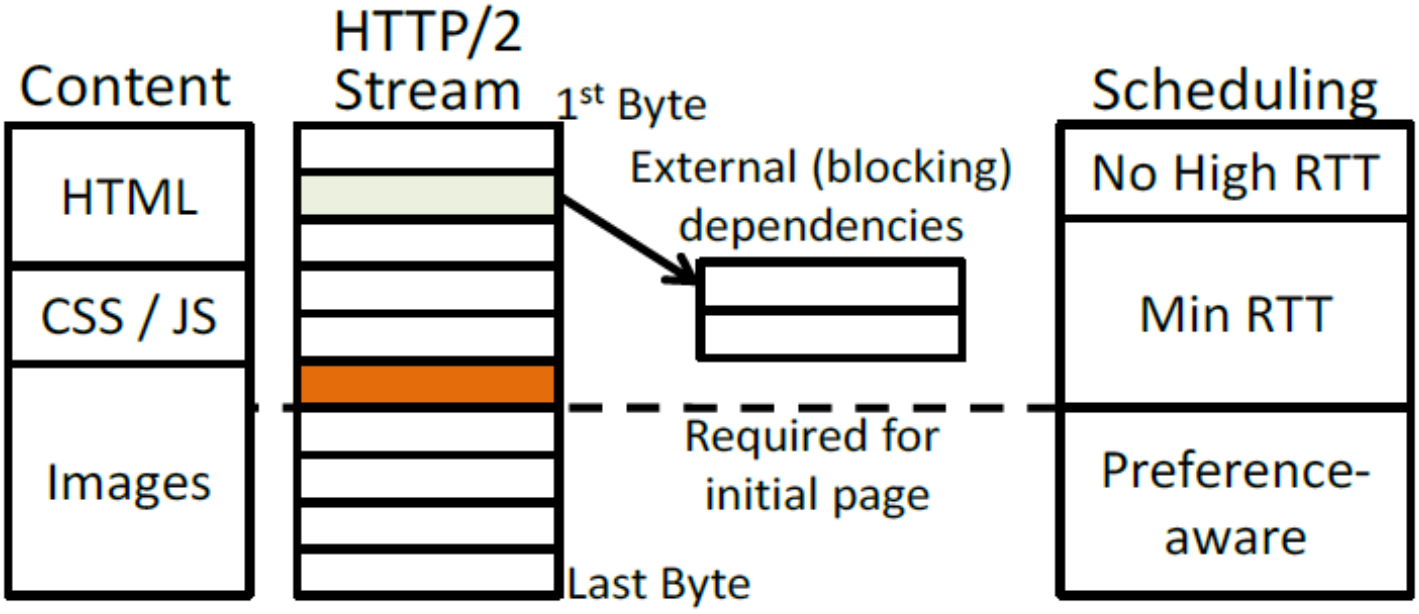


A Framework for the Management, the Scalable Execution and Interactive Analysis of Extensive Network Experiments

https://maci-research.net

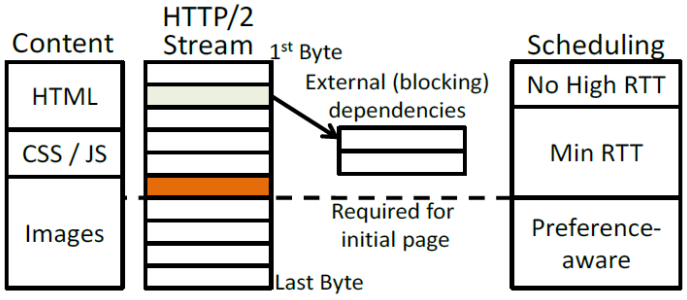
Iterative Refinements in the Research Process

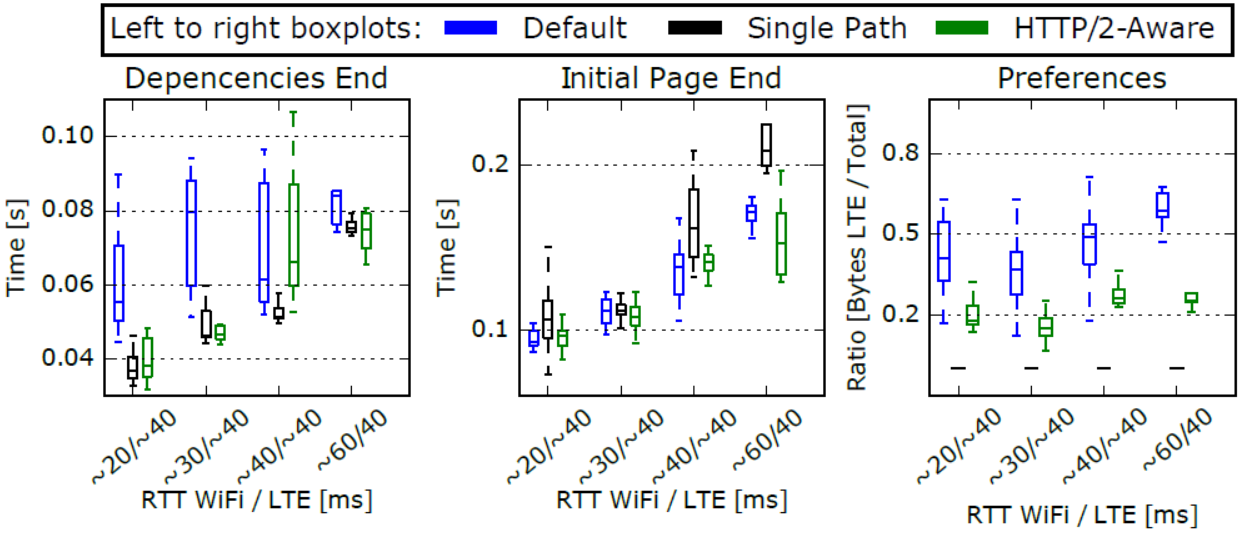
HTTP/2-aware Scheduling



ProgMP enables HTTP/2-aware Scheduling.

HTTP/2-aware Scheduling





Note that the large variance is partly caused by a high variance of the underlying real world environment.

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

We presented **the first programming model for Multipath TCP scheduling**.

- **Specification** and **execution** of MPTCP schedulers
- **Application-defined** MPTCP scheduling

We **proposed** and **evaluated** sophisticated novel schedulers.

- RTT-aware scheduler
- Constant bitrate schedulers
- Flavors of redundant schedulers
- HTTP-aware scheduler
- …

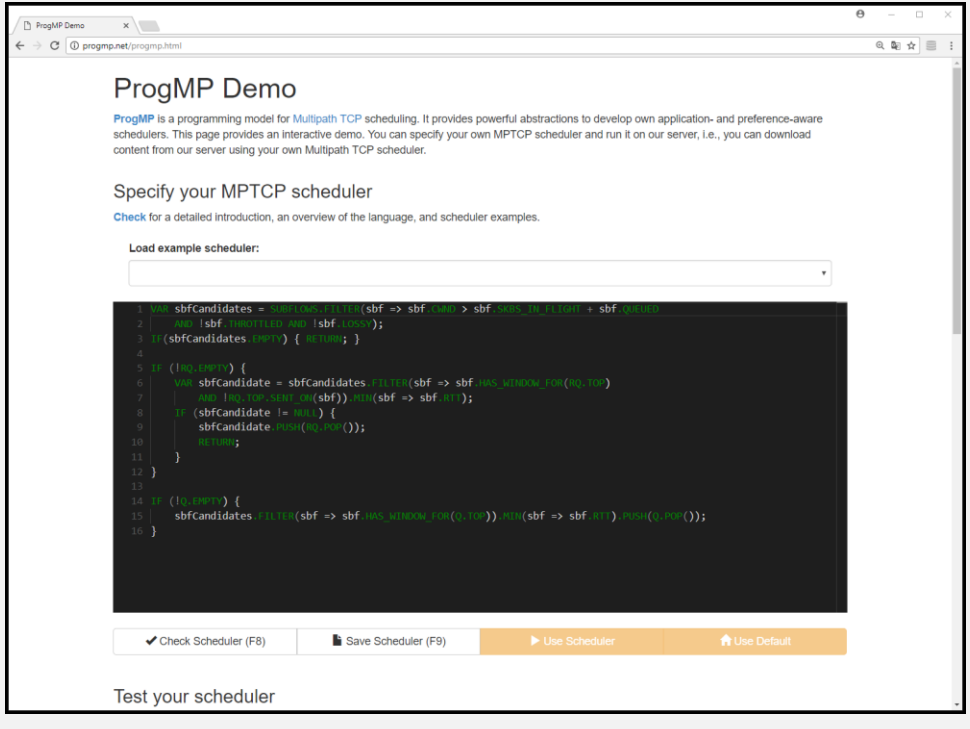
Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

We pre

We **pro**

Try it! https://progmp.net



Questions

TECHNISCHE UNIVERSITÄT DARMSTADT

Multipath TCP in iOS 11

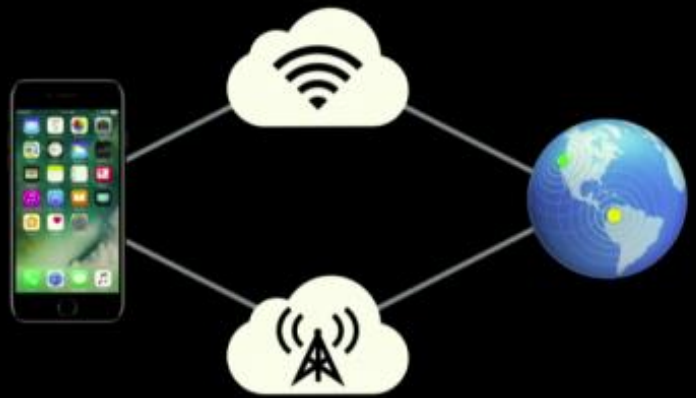
Interactive Mode

NEW

Low latency for low-volume interactive flows

Wi-Fi and cellular

Available in an upcoming Beta



http://www.tessares.net/highlights-from-advances-in-networking-part-1/

Multipath TCP Scheduler Overview

TECHNISCHE UNIVERSITÄT DARMSTADT

Configure the scheduler:

We have a modular scheduler infrastructure. At compile-time you can select the schedulers that should be compiled in.

At run-time you can select one of the compiled schedulers through the sysctl net.mptcp.mptcp_scheduler. You have the choice between:

- 'default': This scheduler is the default one. It will first send data on subflows with the lowest RTT until their congestion-window is full. Then, it will start transmitting on the subflows with the next higher RTT.
- 'roundrobin': This scheduler will transmit traffic in a round-robin fashion. It is configurable, how many consecutive segments should be sent with the tunable "num_segments" in the sysfs (default 1). Additionally, you can set the boolean tunable "cwnd_limited", to specify whether the scheduler tries to fill the congestion window on all subflows (true) or whether it prefers to leave open space in the congestion window (false) to achieve real round-robin (even if the subflows have very different capacities) (defaults to true). In case you are unsure, never ever enable this scheduler. Its performance is bad and it is only interesting for academic/testing purposes. The default scheduler is the best known up to today.
- 'redundant': This scheduler will try to transmit the traffic on all available subflows in a redundant way. It is useful when one wants to achieve the lowest possible latency by sacrificing the bandwidth.

New in v0.92: Alternatively, you can select the scheduler through the socket-option MPTCP_SCHEDULER (defined as 43) by doing:

Enabling per Application-aware Scheduling in Multi-Tenancy Cloud Environments

TECHNISCHE UNIVERSITÄT DARMSTADT