Application-defined Multipath TCP Scheduling

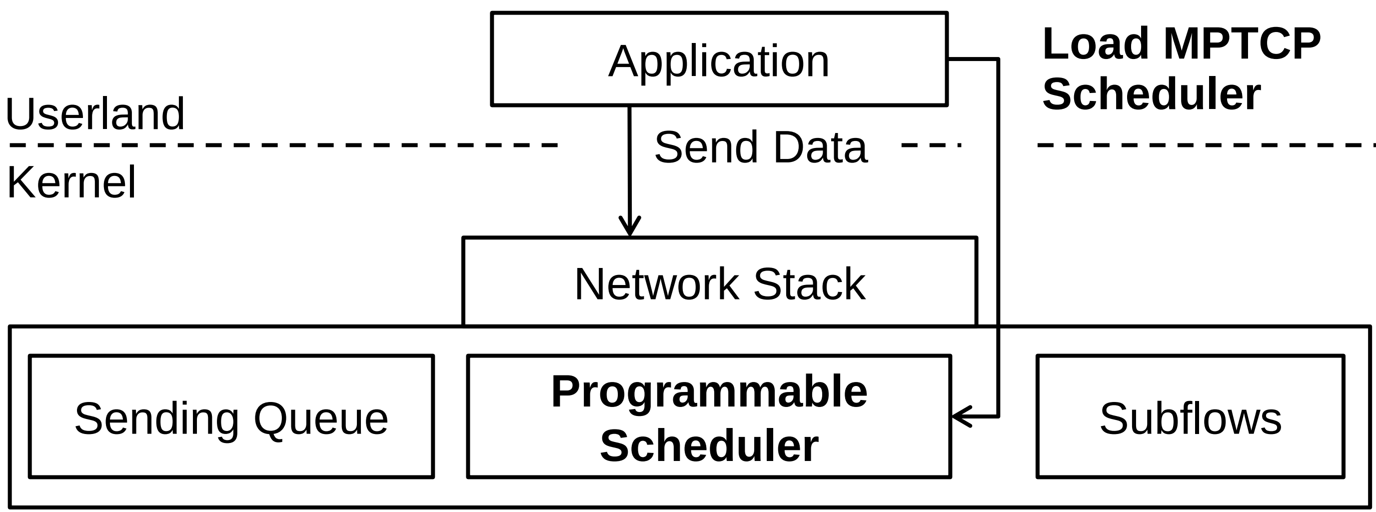
TECHNISCHE UNIVERSITÄT DARMSTADT

KOM – Multimedia Communications Lab

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

Frühjahrstreffen der GI Fachgruppen BS und KuVS

Amr Rizk



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

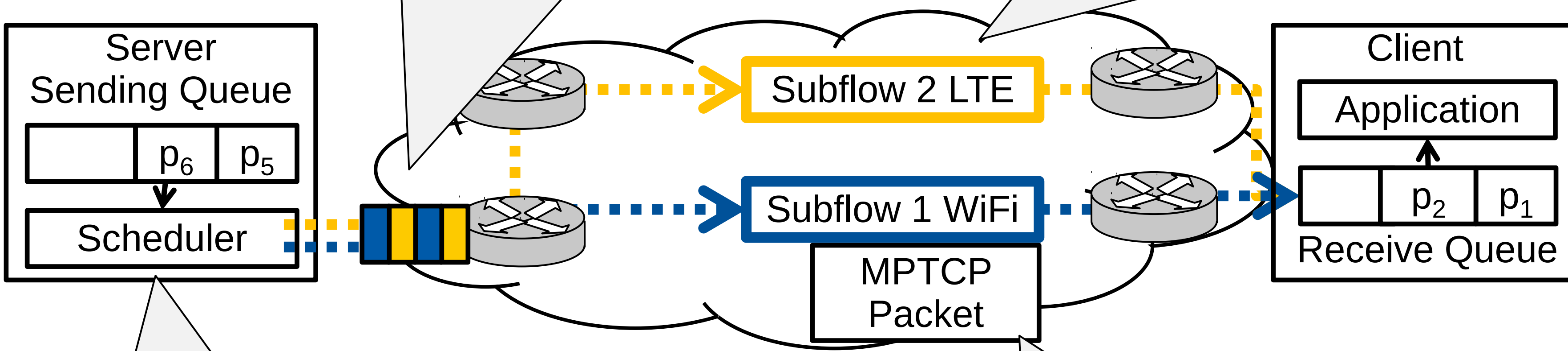
A version of this work was published at ACM/USENIX Middleware 2017

Multipath TCP in a Nutshell

TECHNISCHE UNIVERSITÄT DARMSTADT

1	Congestion Control
	Fairness on Joint Paths

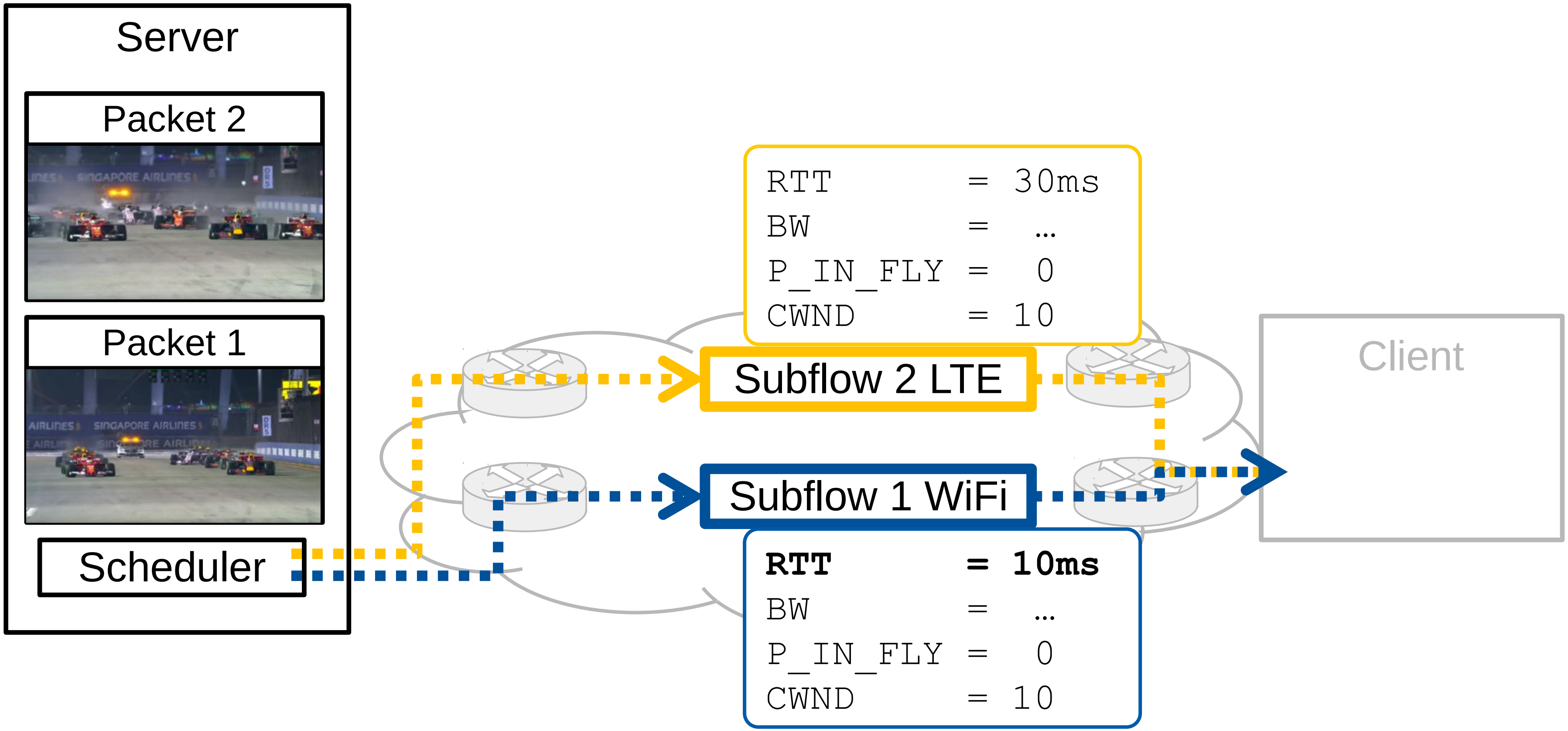
2	Path Management
	Trigger Subflow Establishment



4	Scheduler
	Map Packets to Subflows

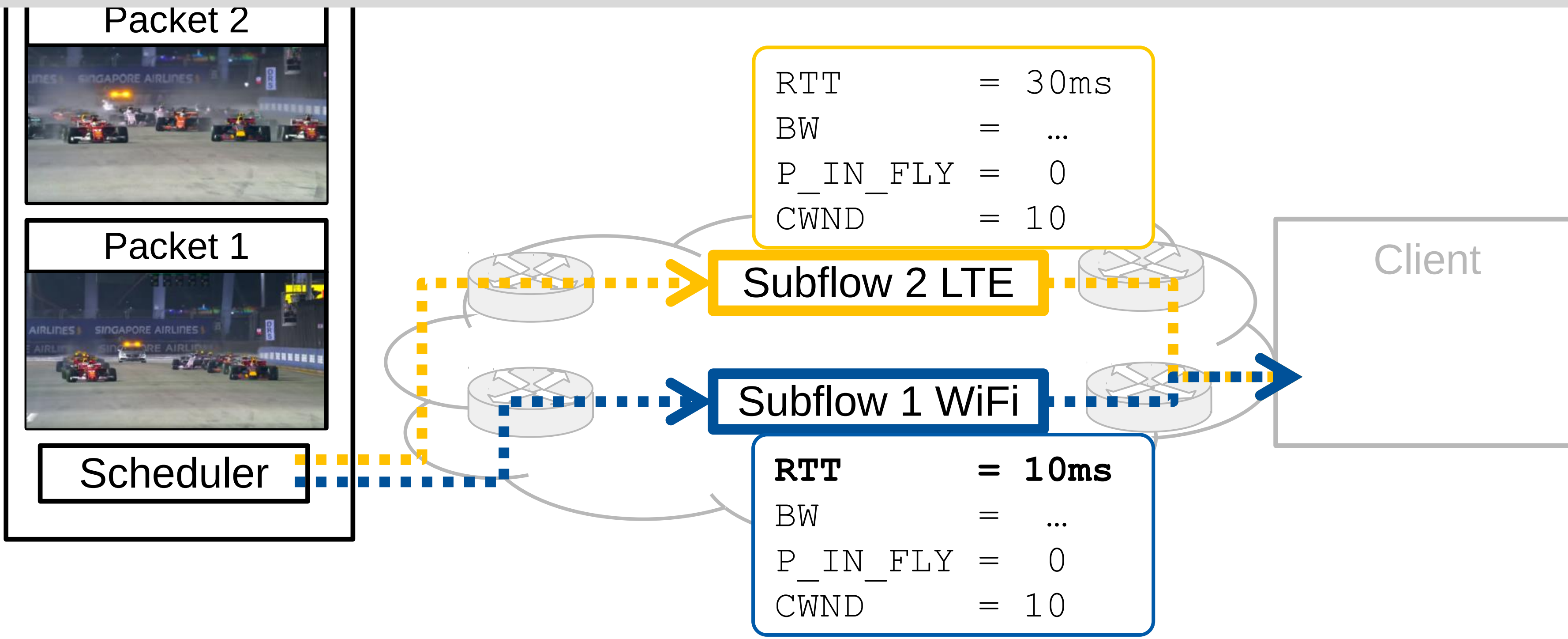
3	TCP Options for MPTCP
	Control Information, e.g., Data Sequence Numbers

Multipath TCP Scheduling



Multipath TCP Scheduling

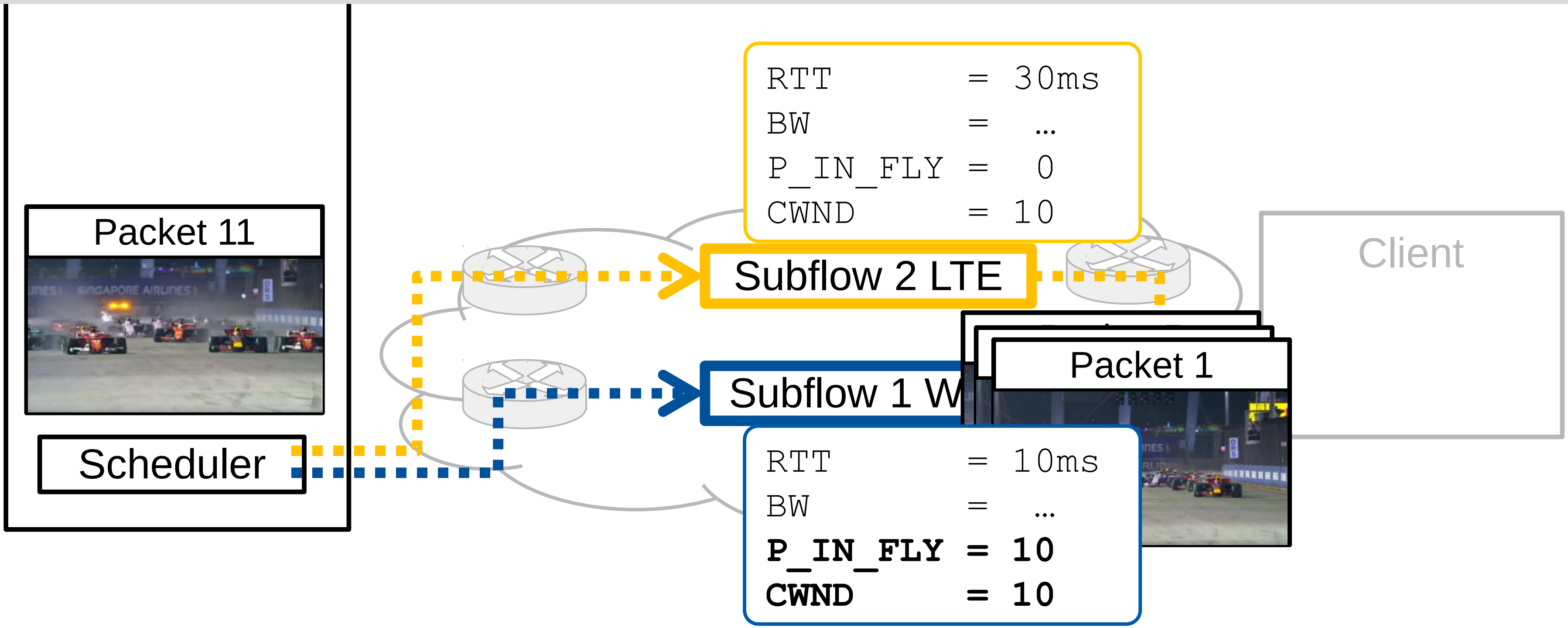
Intuition: Schedule on Subflow with minimum round-trip time (**minRTT**).



Images: https://www.youtube.com/watch?v=U2T1_VrTcoo

Multipath TCP Scheduling

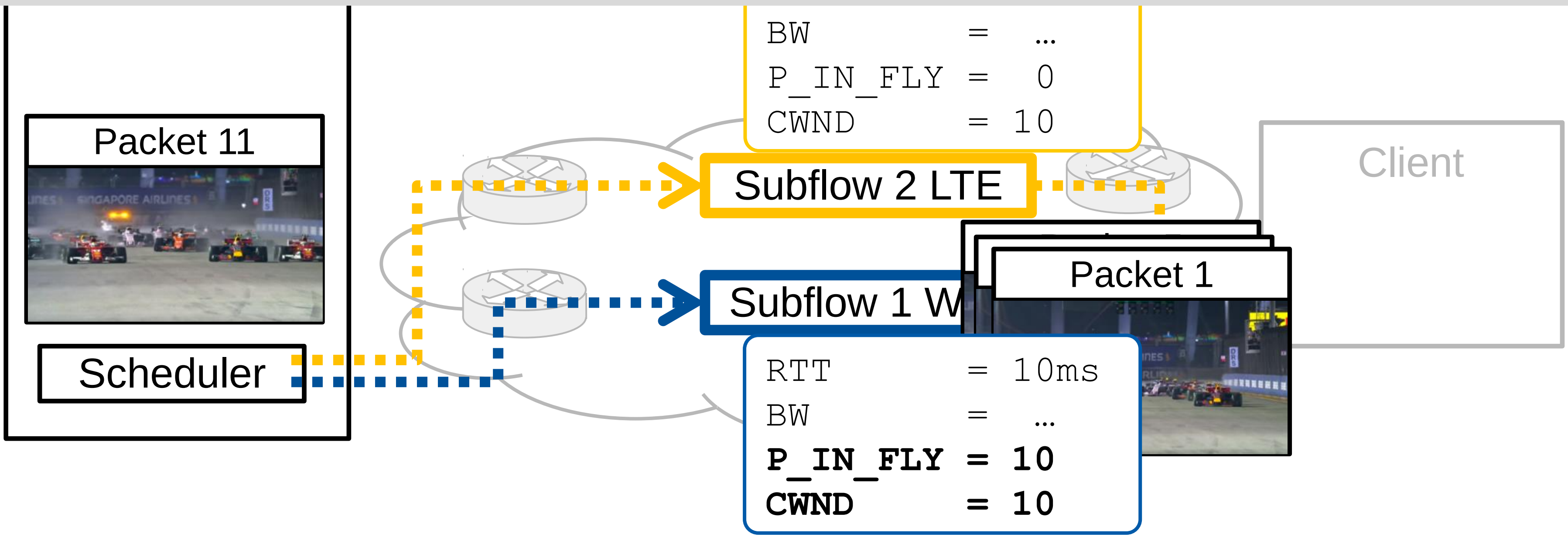
Schedule on Subflow with minimum round-trip time (minRTT).



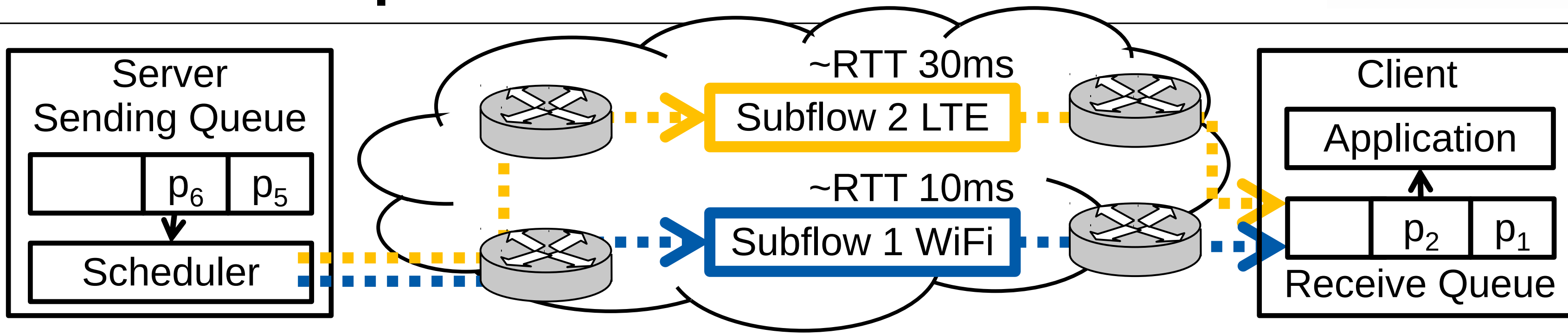
Multipath TCP Scheduling

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

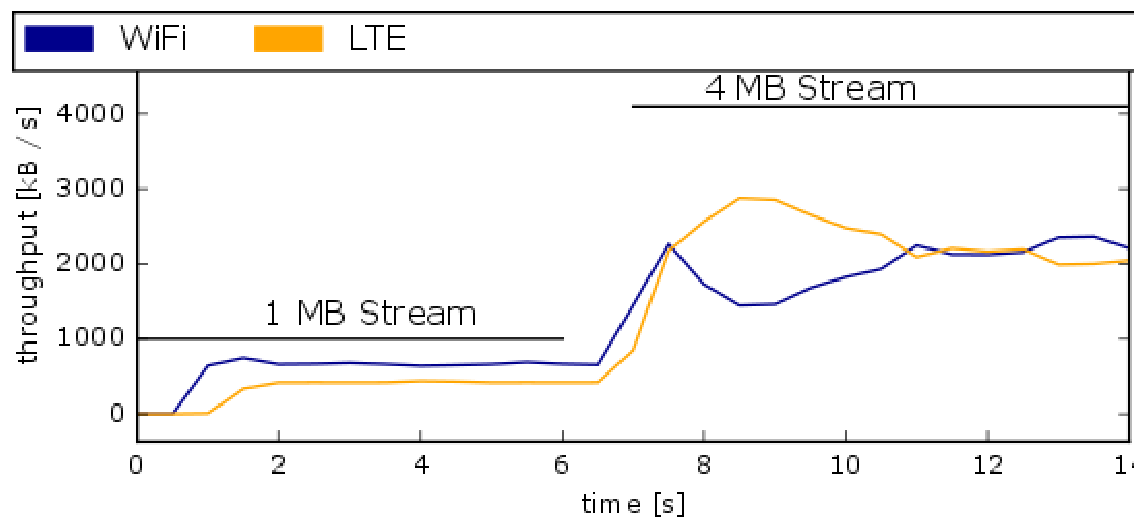
which is not saturated (Congestion window > packets in flight).



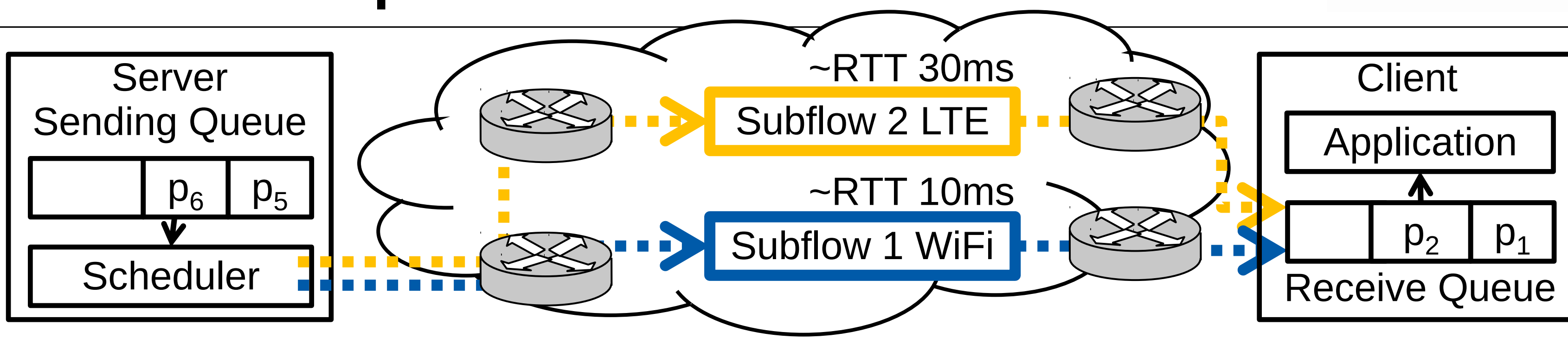
Multipath TCP Scheduling
Real-world Experiment



Constant Bitrate Stream
Home Network WiFi and LTE Germany

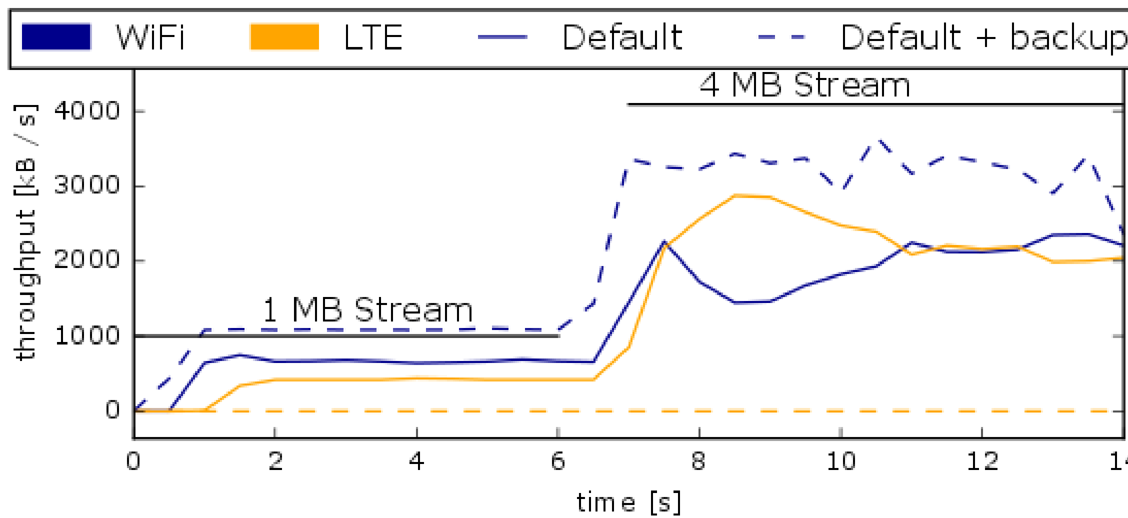


Multipath TCP Scheduling
Real-world Experiment



Constant Bitrate Stream

Home Network WiFi and LTE Germany



Multipath TCP Scheduler Overview

Name	Domain	Preferences	Available in Linux (Implementation)
Min RTT (**Default**)	General Purpose	Binary	✓
Round-Robin	General Purpose	Binary	✓
Redundant	Thin Flows	Binary	✓
Compensate Loss	Short Datacenter Flows	?	**No**
Energy preserving		?	**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

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.

independent subflows. This approach provides tra
applications, at the cost of requiring kernel upgr
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 cruci
path diversity is utilized, i.e. the five-tuples of
and replicated flow have to be different (assumi

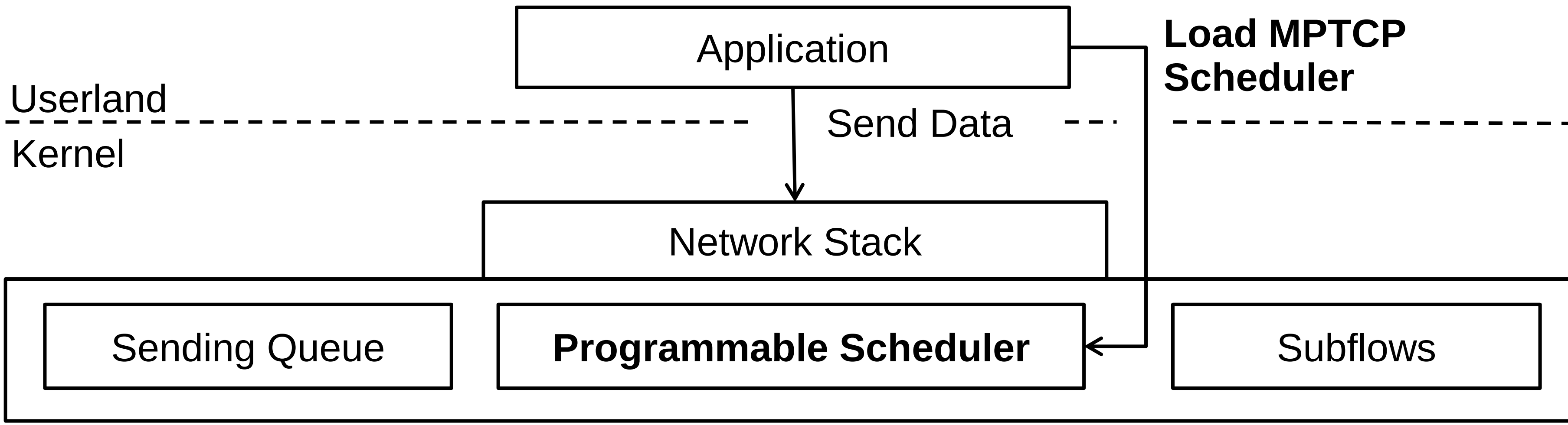
H. Xu and B. Li. RepFlow: Minimizing flow completio
replicated flows in data centers.
In IEEE INFOCOM, 2014

Goals (of this talk)

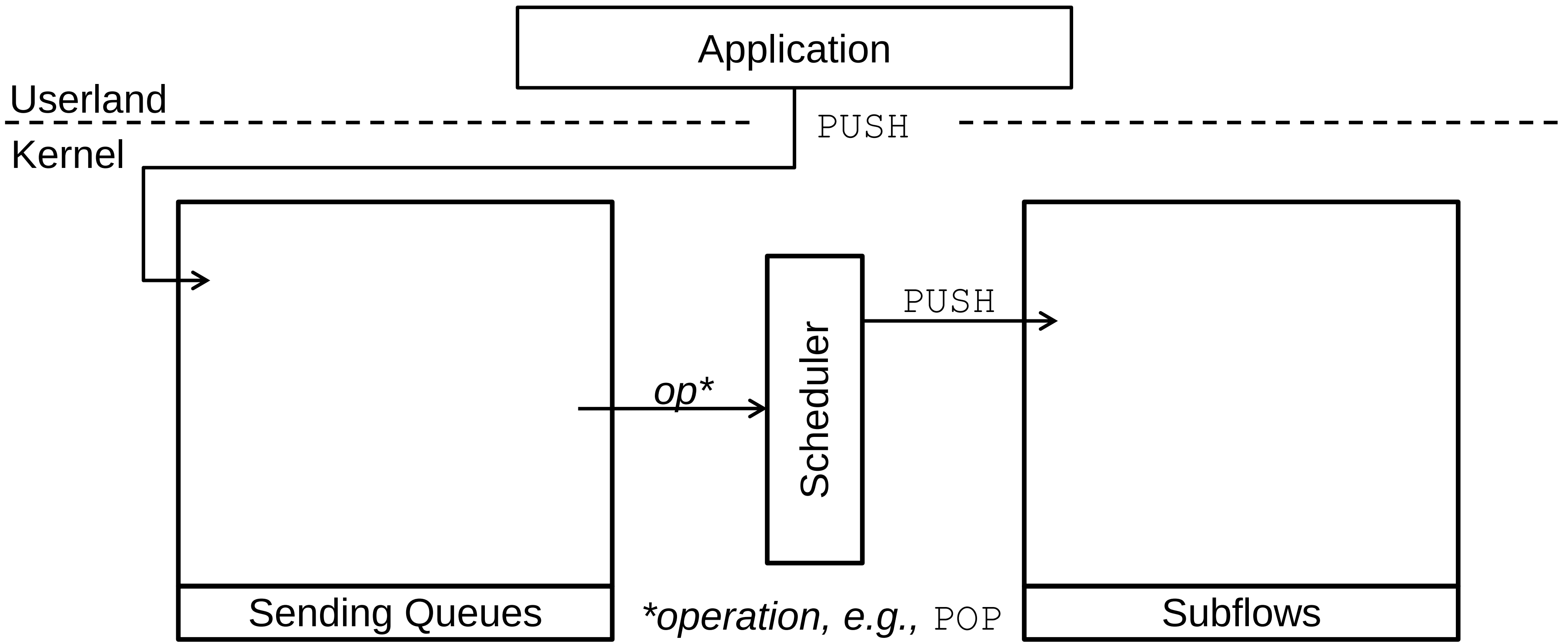
Systematically **specify** and **execute** MPTCP schedulers

Enable application-defined MPTCP scheduling

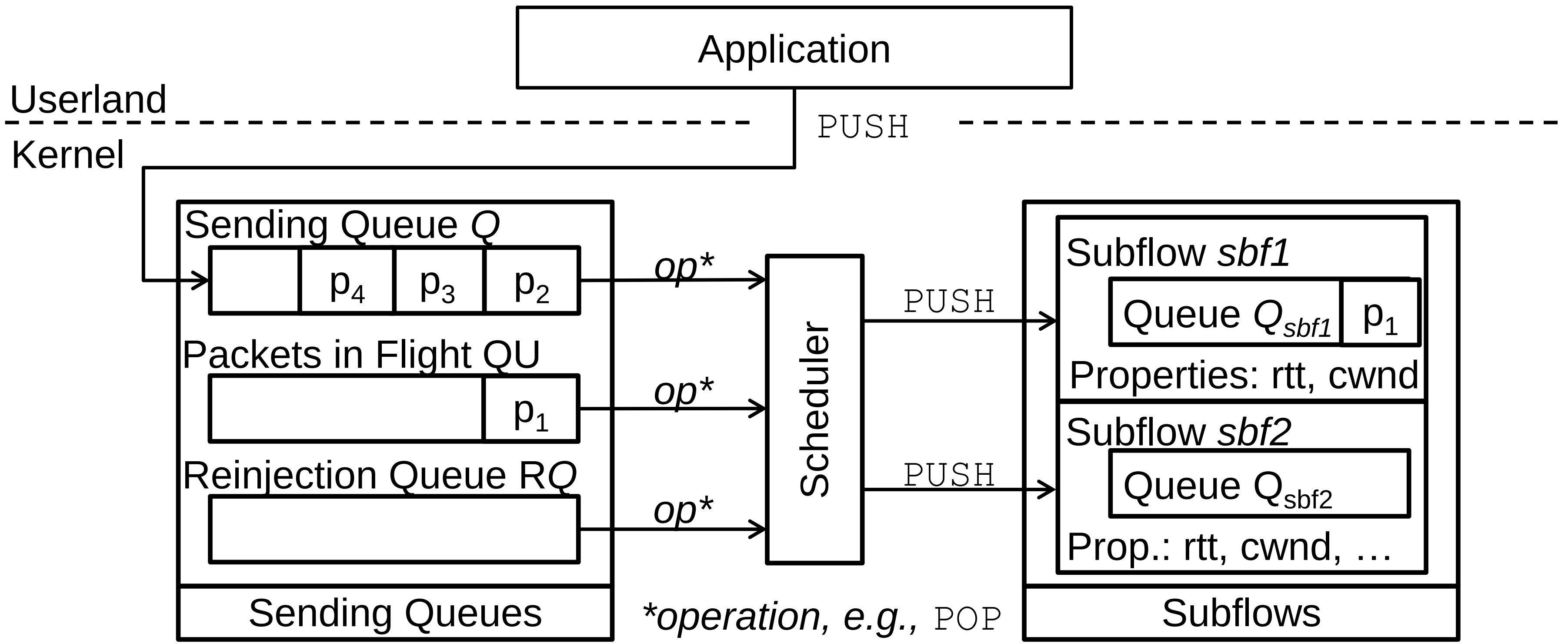
Part I: Towards a Programmable Scheduler in the Network Stack



Model of the Scheduler Environment



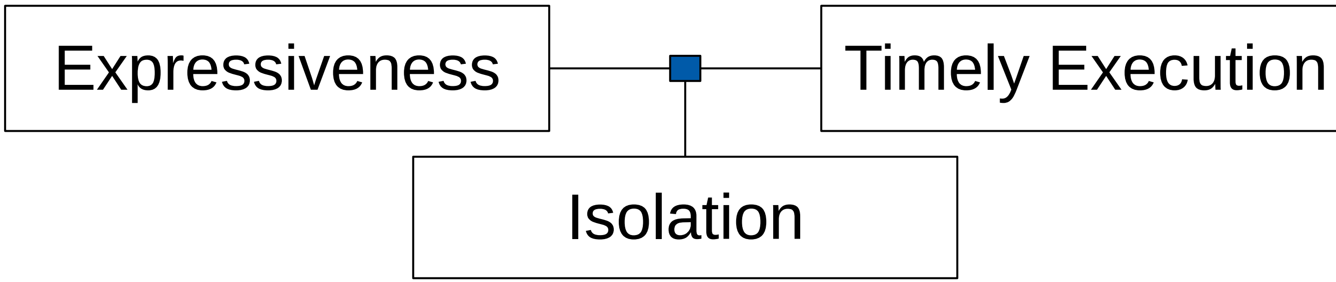
Model of the Scheduler Environment



Specifying Multipath TCP Schedulers

TECHNISCHE UNIVERSITÄT DARMSTADT

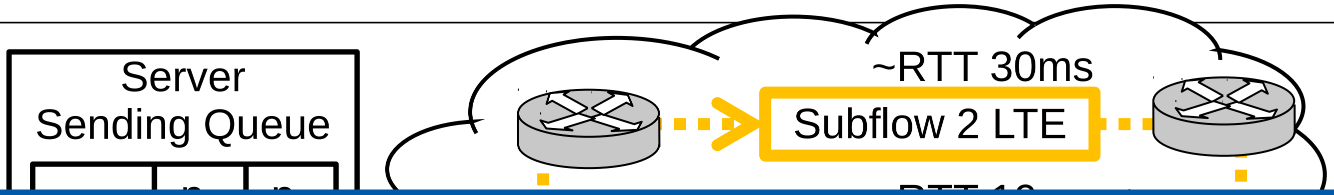
Requirements



Design Decisions

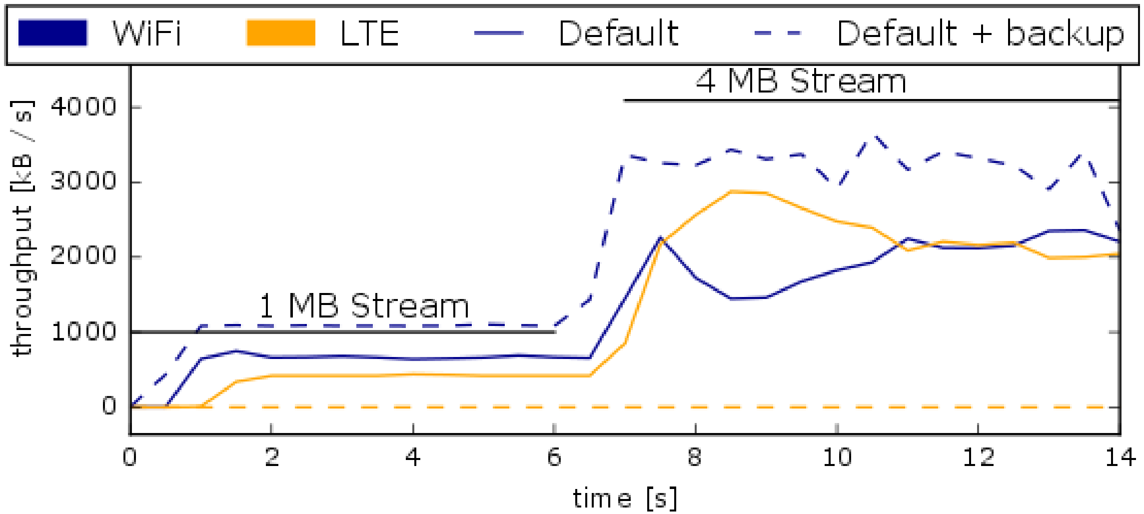
1. Modelled Elements as Entities: Set of Subflows with their properties, Queues of Packets with their Properties
2. Declarative Packet and Subflow Selection (Filter, Min, Max)
3. No Recursion, No Functions, Limited Loops
4. Variables with Single Assignment, Implicit and Static Type System
5. No, One,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

Back to the Motivating Example



How to use the Backup Flow?

Constant Bitrate Stream
Home Network WiFi and LTE Germany



Systematically Specify MPTCP Schedulers
Domain Specific Specification Language

TECHNISCHE UNIVERSITÄT DARMSTADT

Example: Preference-aware **RTT-sensitive** Scheduler

```
VAR sbfCandis = SUBFLOWS.FILTER(
        sbf => sbf.CWND > sbf.SKBS_IN_FLIGHT + sbf.QUEUED
               AND !sbf.TSQ_THROTTLED AND !sbf.LOSSY);

VAR backSbf = sbfCandis.FILTER(
              sbf => sbf.IS_BACKUP).MIN(sbf => sbf.RTT);
VAR nonBackSbf = sbfCandis.FILTER(
              sbf => !sbf.IS_BACKUP).MIN(sbf => sbf.RTT);

IF (nonBackSbf.RTT_MS > 100 AND backSbf.RTT_MS < 80) {
   backSbf.PUSH(Q.POP());
} ELSE {
   nonBackSbf.PUSH(Q.POP());
}
```

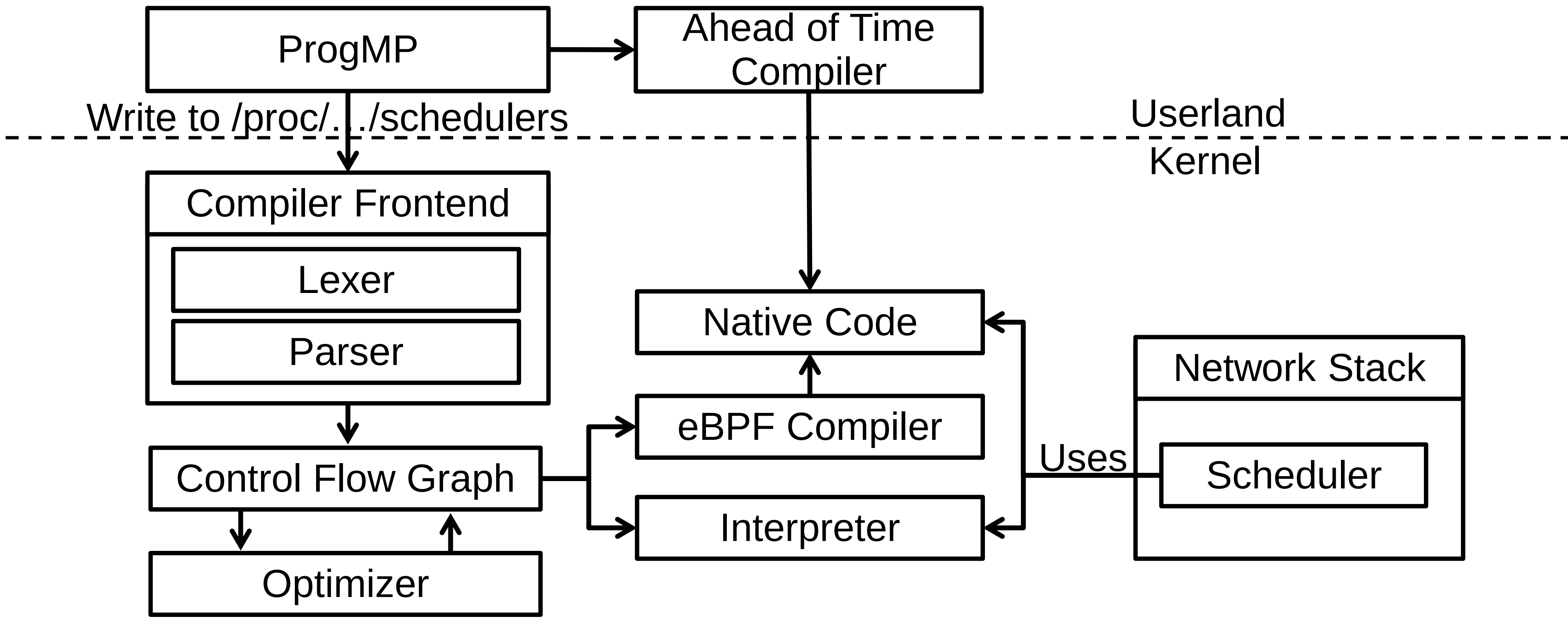
Systematically Specify MPTCP Schedulers
Domain Specific Specification Language

TECHNISCHE UNIVERSITÄT DARMSTADT

Example: Preference-aware **RTT-sensitive** Scheduler

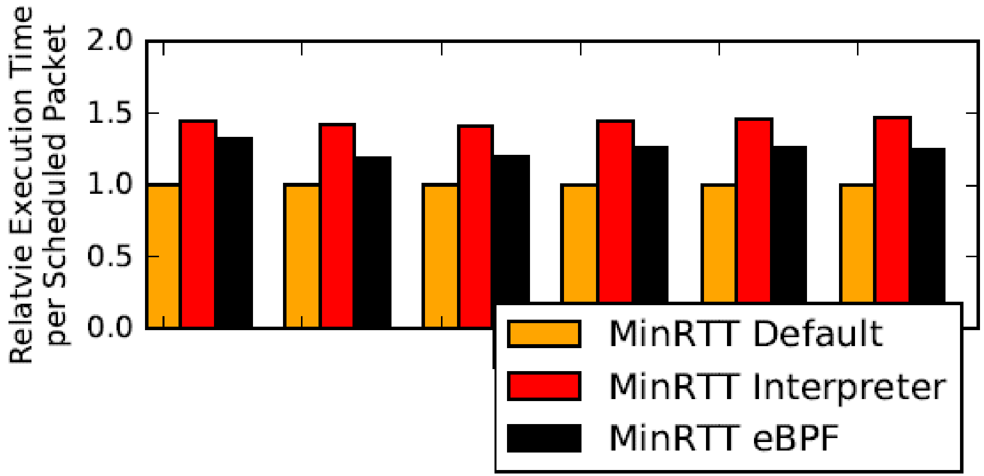
```
VAR sbfCandis = SUBFLOWS.FILTER(
        sbf => sbf.CWND > sbf.SKBS_IN_FLIGHT + sbf.QUEUED
               AND !sbf.TSQ_THROTTLED AND !sbf.LOSSY);

VAR backSbf = sbfCandis.FILTER(
              sbf => sbf.IS_BACKUP).MIN(sbf => sbf.RTT);
VAR nonBackSbf = sbfCandis.FILTER(
              sbf => !sbf.IS_BACKUP).MIN(sbf => sbf.RTT);

IF (nonBackSbf.RTT_MS > R1 AND backSbf.RTT_MS < R2) {
   backSbf.PUSH(Q.POP());
} ELSE {
   nonBackSbf.PUSH(Q.POP());
}
```

Systematically Specify and *Execute* MPTCP Schedulers

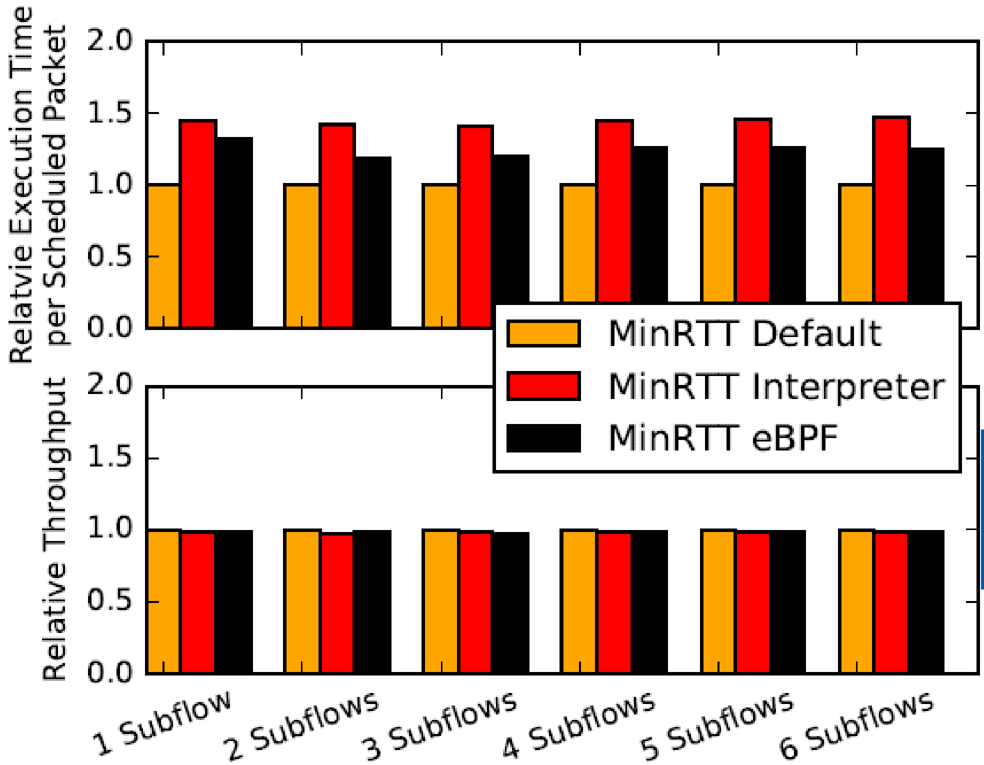


Specified schedulers are executable in the Linux Kernel

Abstraction vs. Overhead



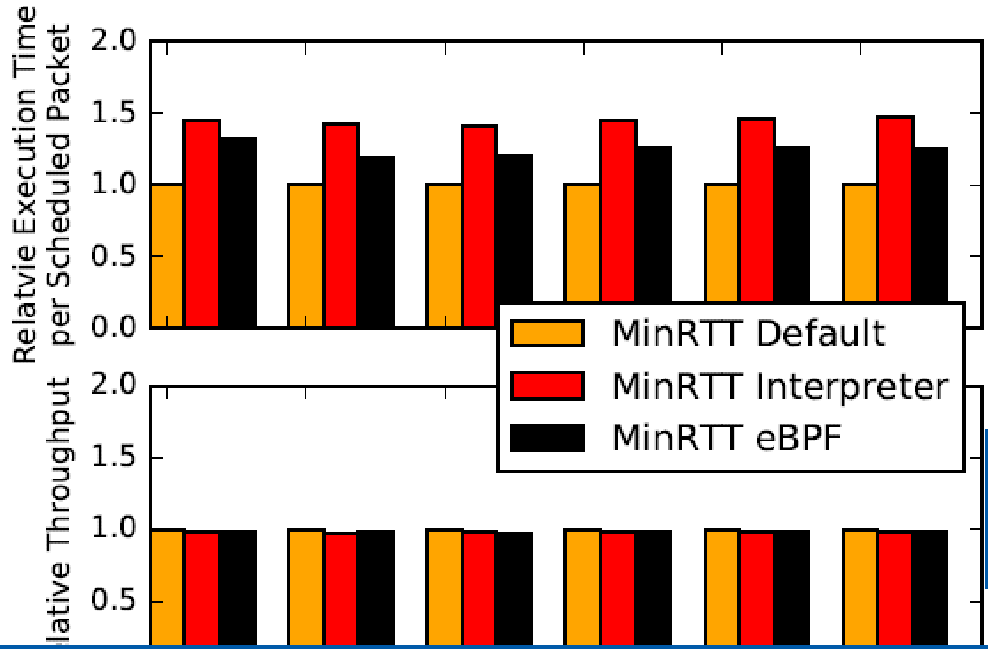
Abstraction vs. Overhead



link saturation
at 1 Gbit/s

Abstraction vs. Overhead

TECHNISCHE UNIVERSITÄT DARMSTADT



link saturation
at 1 Gbit/s

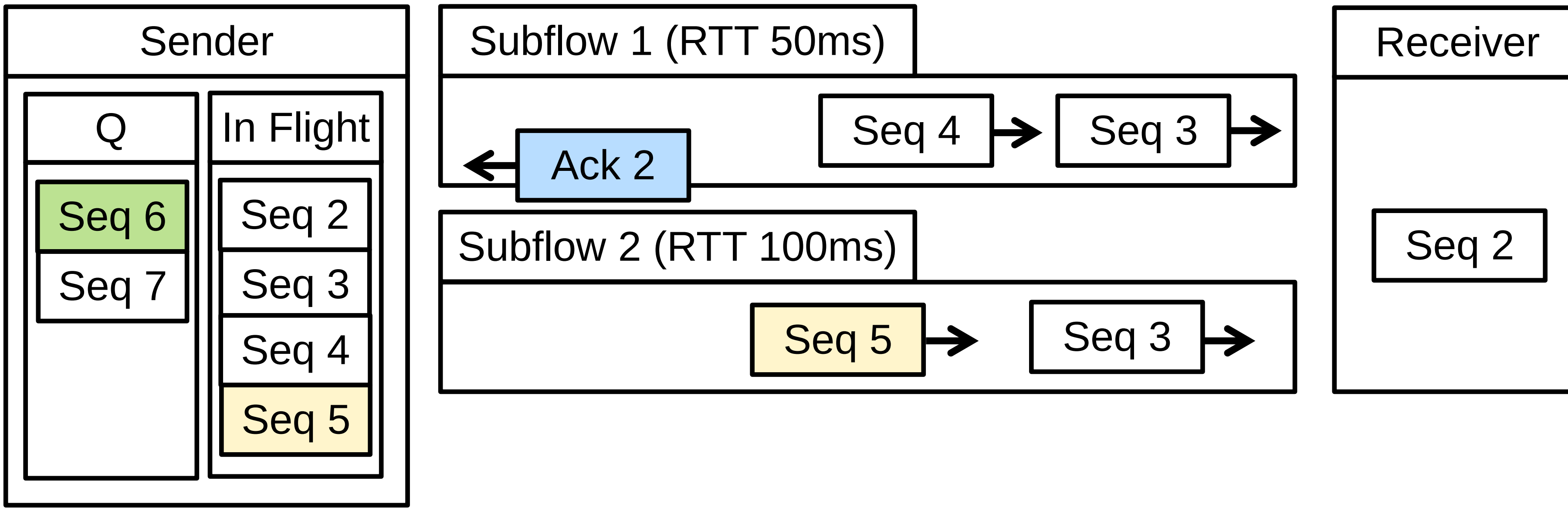
The runtime environment induces an overhead, which is acceptable for most application scenarios.

Part II: Design of Novel Multipath TCP Schedulers

TECHNISCHE UNIVERSITÄT DARMSTADT

	Preference-aware	Application-aware	Executable
Round-trip Time-aware	✓	✓	✓
Constant Bitrate Stream Scheduling	✓	✓	✓
Redundant Scheduling			✓
HTTP/2-aware Scheduling	✓	✓	✓
More in Paper and Under Review			

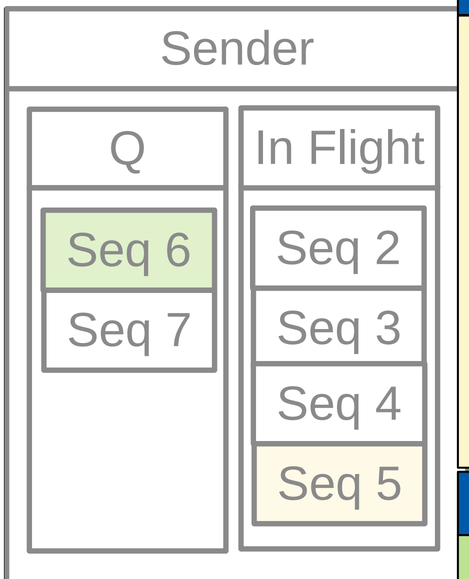
A Close Look at Redundant Schedulers



When the acknowledgement Ack 2 arrives at the sender:

→ should we send the fresh packet Seq 6 or the old packet Seq 5

A Close Look at Redundant Schedulers



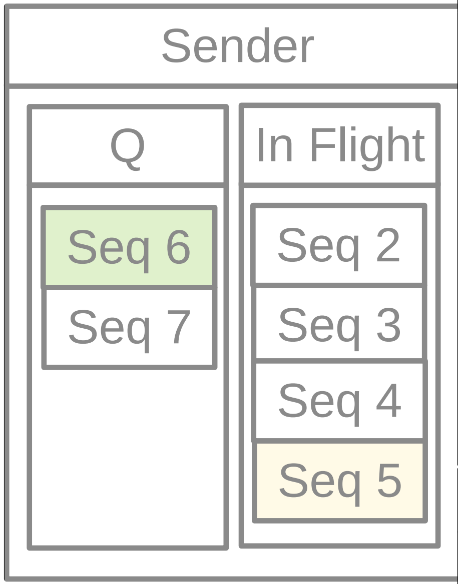
Prefer Per Subflow Fresh Packets

```
FOREACH(VAR sbf IN sbfCandidates) {
  VAR skb = QU.FILTER(s => !s.SENT_ON(sbf).TO
  IF(skb != NULL) {
    sbf.PUSH(skb);
  } ELSE {
    sbf.PUSH(Q.POP());
  }
}
```

Prefer Global Fresh Packets

```
IF(!sbfCandidates.EMPTY) {
  FOREACH(VAR sbf IN sbfCandidates) {
    sbf.PUSH(Q.TOP);
  }
  DROP(Q.POP());
}
```

A Close Look at Redundant Schedulers



Prefer Per Subflow Fresh Packets

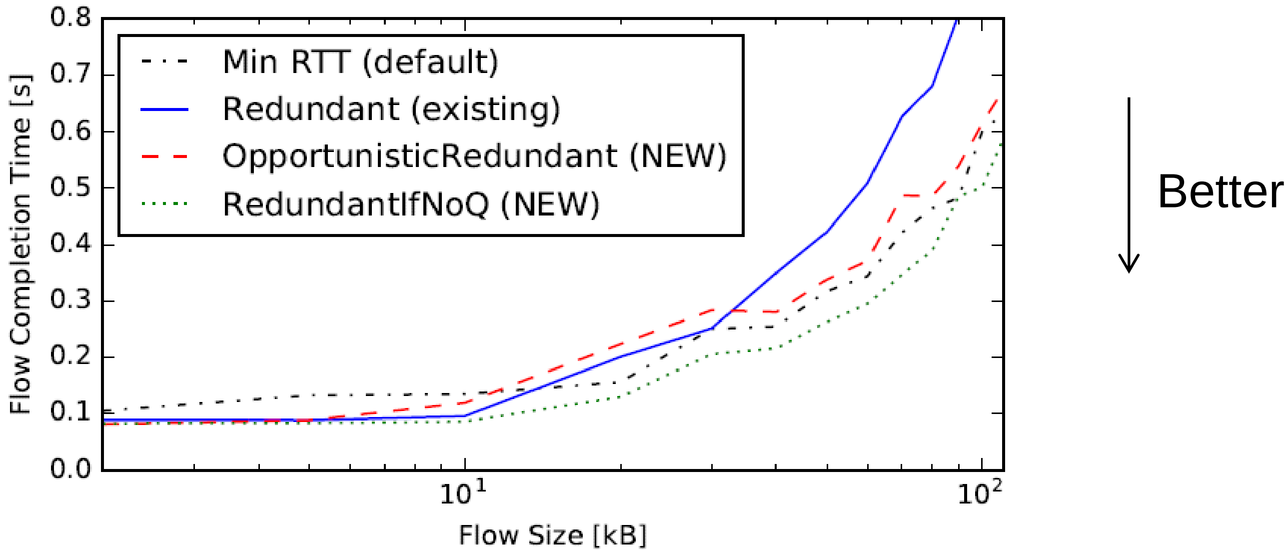
```
FOREACH(VAR sbf IN sbfCandidates) {
  VAR skb = QU.FILTER(s => !s.SENT_ON(sbf).TO
  IF(skb != NULL) {
    sbf.PUSH(skb);
  } ELSE {
    sbf.PUSH(Q.POP());
  }
}
```

Prefer Global Fresh Packets

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

ProgMP enables rapid specification and evaluation of schedulers

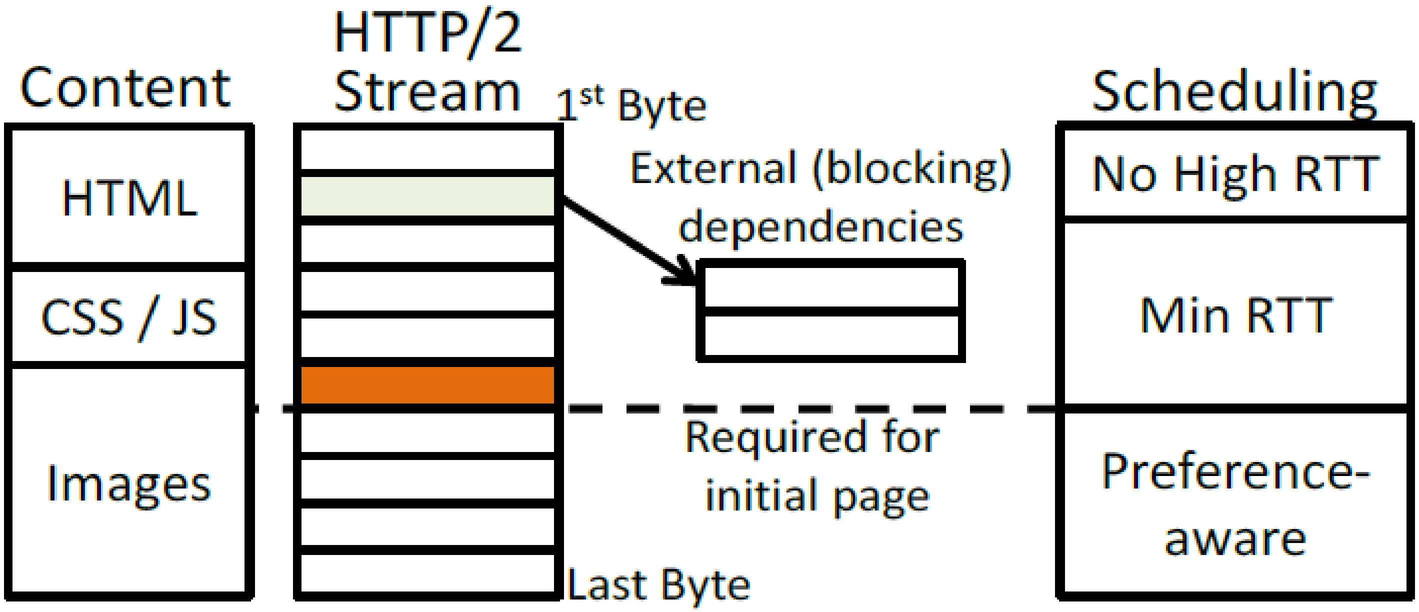
A Close Look at Redundant Schedulers



ProgMP enables novel redundant schedulers, which outperform established approaches.

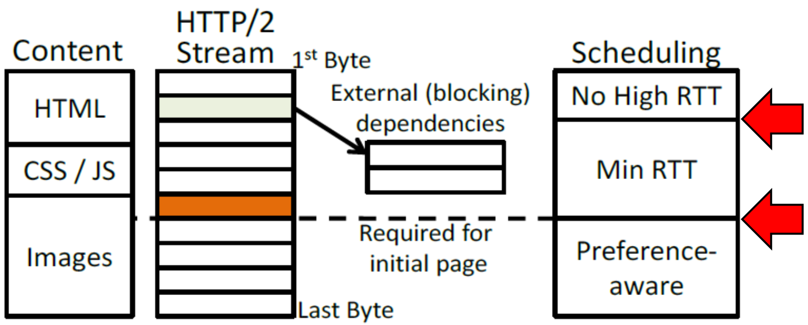
HTTP/2-aware Scheduling

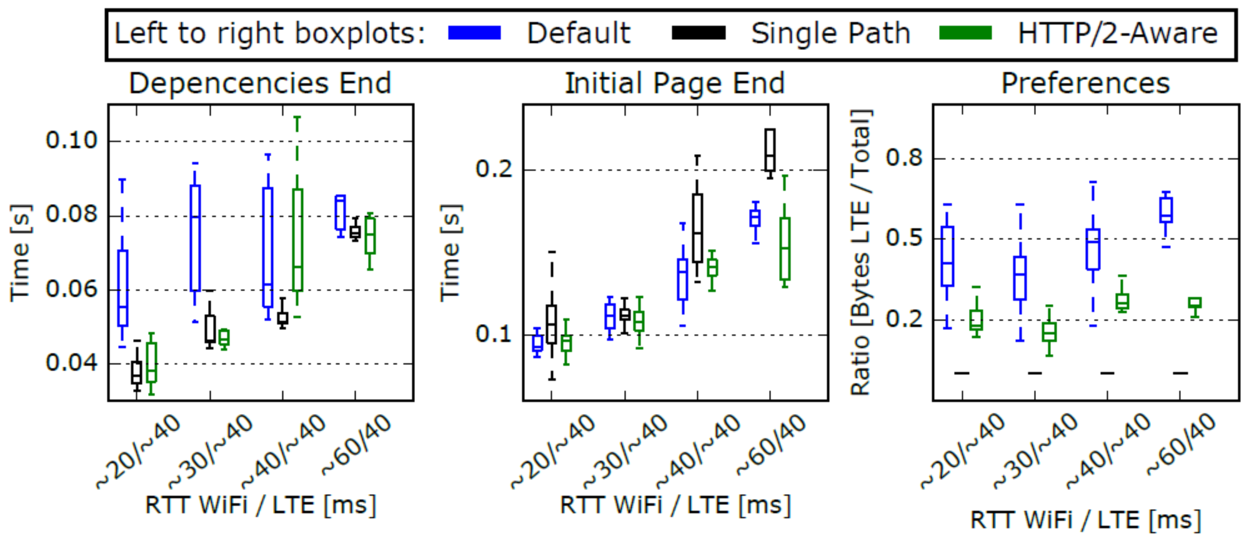
TECHNISCHE UNIVERSITÄT DARMSTADT



ProgMP enables HTTP/2-aware Scheduling.

HTTP/2-aware Scheduling





Conclusion

We presented **a programming model for Multipath TCP scheduling**

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

We **proposed** and **evaluated** novel MPTCP schedulers

- RTT-aware scheduler
- Constant bitrate schedulers
- Flavors of redundant schedulers
- HTTP/2-aware scheduler
- …

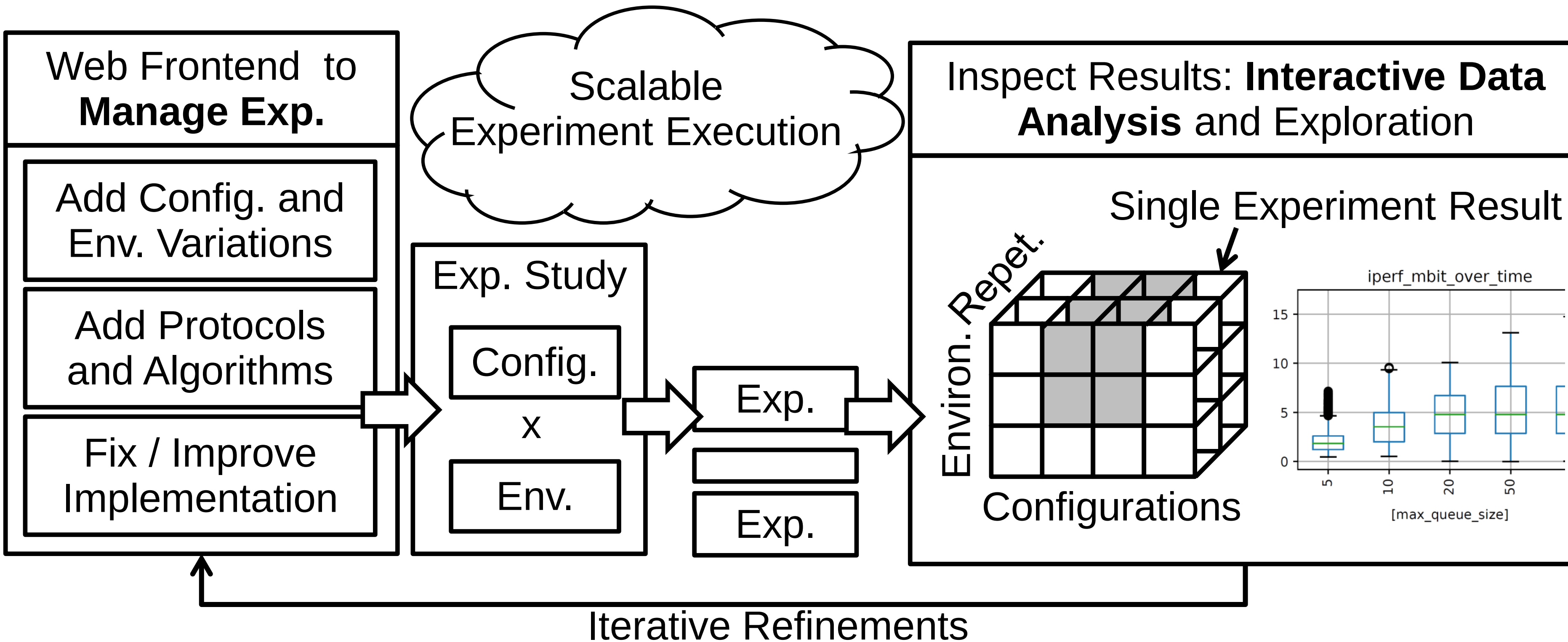
Conclusion

https://progmp.net

We pre

We **pro**

How can we systematically compare and evaluate scheduler design decisions?



How can we systematically compare and evaluate scheduler design decisions?

Web Frontend to **Manage Exp.**

Add Config. and Env. Variations

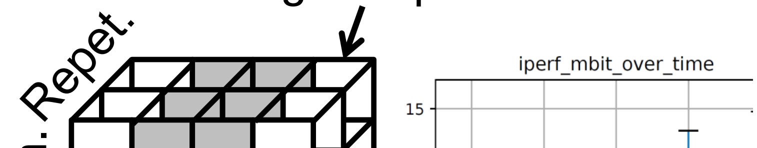
Add Protocols

Scalable Experiment Execution

Exp. Study

Inspect Results: **Interactive Data Analysis** and Exploration

Single Experiment Result



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

https://maci-research.net

Iterative Refinements