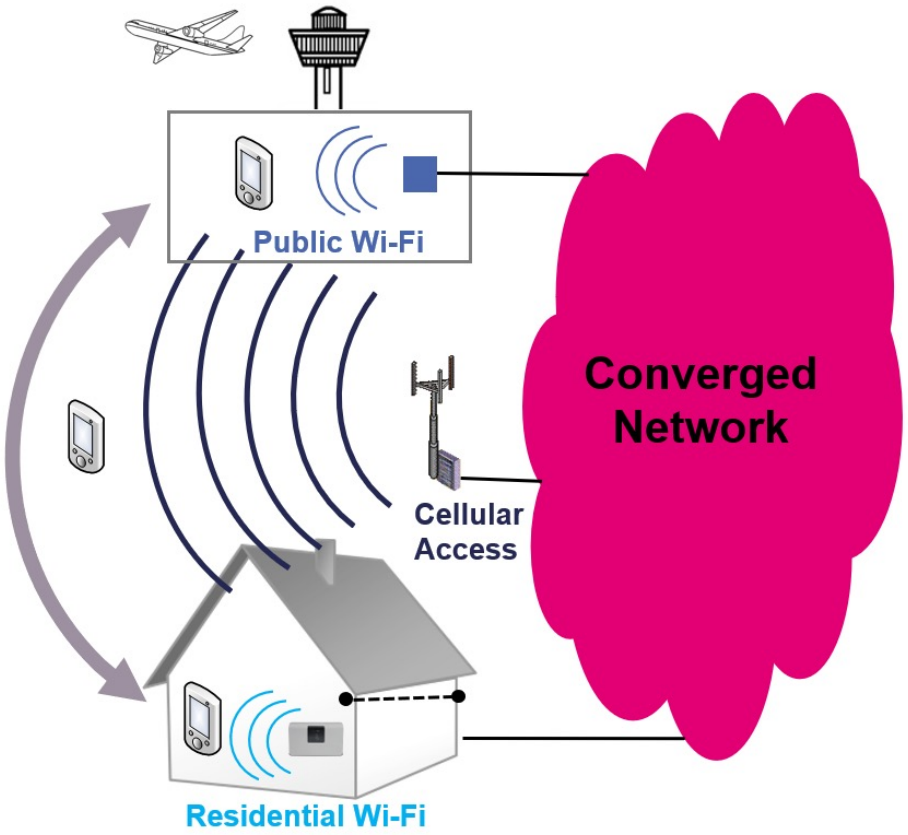
Accelerating Transport Layer Multipath Packet Scheduling for 5G-ATSSS

Fabian Brisch; Andreas J. Kassler; Jonathan Vestin; Marcus Pieska; Markus Amend

https://multipath-dccp.org

Motivation

- Mobile UE has a multitude of interfaces
  - WiFi, Cellular(3G,4G,5G), Satellite,…
- Current implementations only use one interface for traffic
- Using multiple networks concurrently can have several benefits
  - Increased consistency
  - Higher throughput
  - Higher utilization of low-cost interfaces

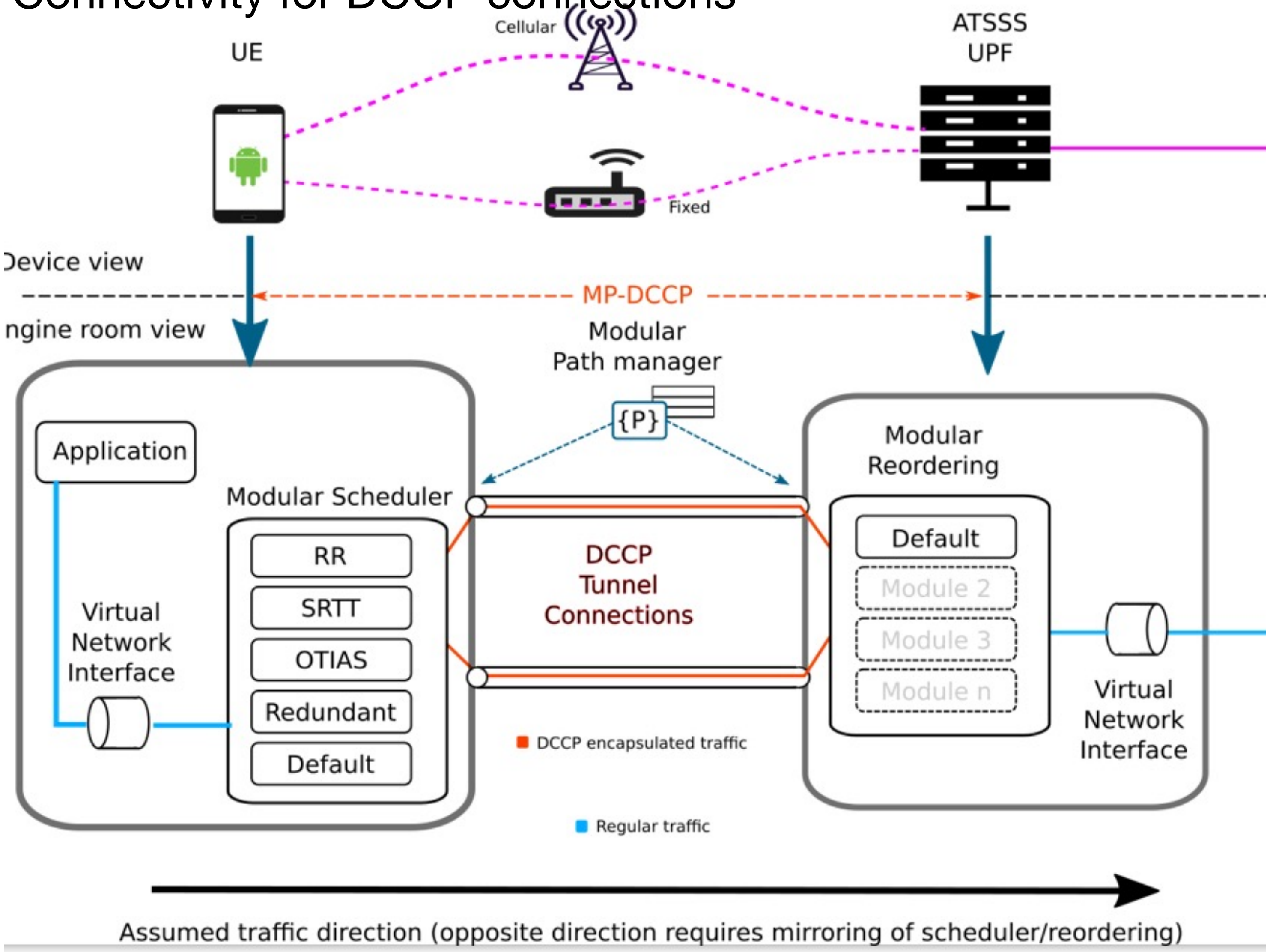


Challenge

- „Preffered“ + secondary path
- Preffered Path Throughput inconsistent and unpredictable, but „lower cost“ for user/network operator
- Aim:
  - Minimize Congestion, Maximize Share of preferred path
  - Keep P4 Implementation possible

MP-DCCP Framework as Enabler for 5GATSSS

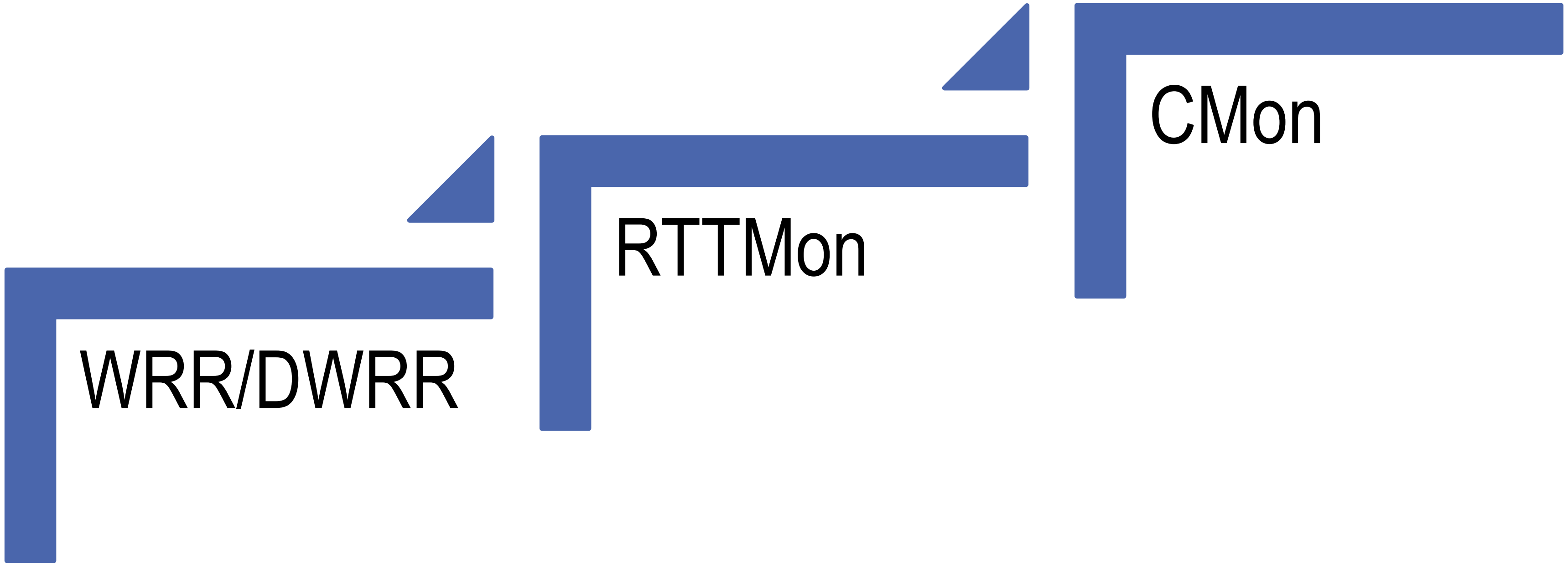
- Enable Multipath Connectivity for DCCP connections



Reordering

- Compensate overall packet scrambling introduced by different path latencies
  →Supports Splitting

Scheduler Implementation in P4

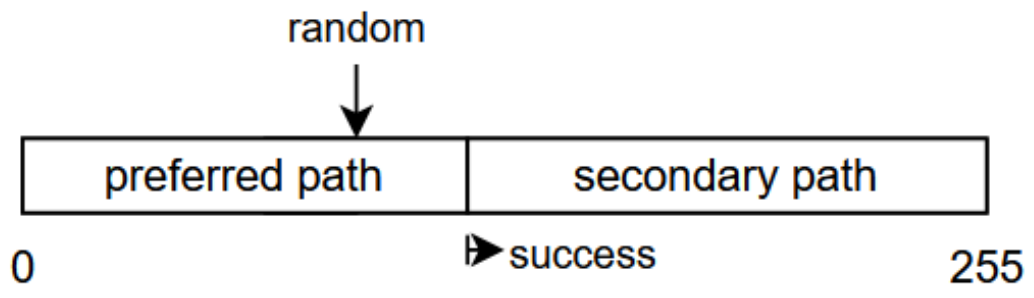


- Rising Complexity of schedulers(tunnel state, scheduling decision)
- Questions
  - Performance Impact?
  - Maximum Complexity?

(Dynamic) Weighted Round Robin Scheduling

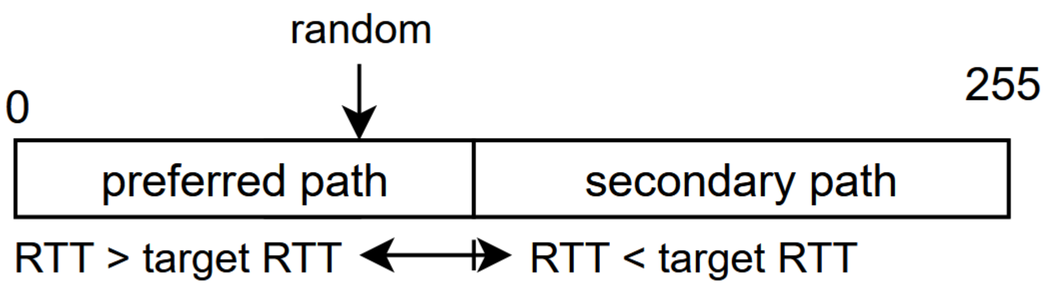
- Compact Weight representation (single 8-bit register)
- Staticor dynamic weights
- Generated random determines path that packet gets scheduled on

- Very lightweight implementation, no additional mutexing necessary
- Weight adjustment on transmittion success/failure



RTT Monitoring Scheduling

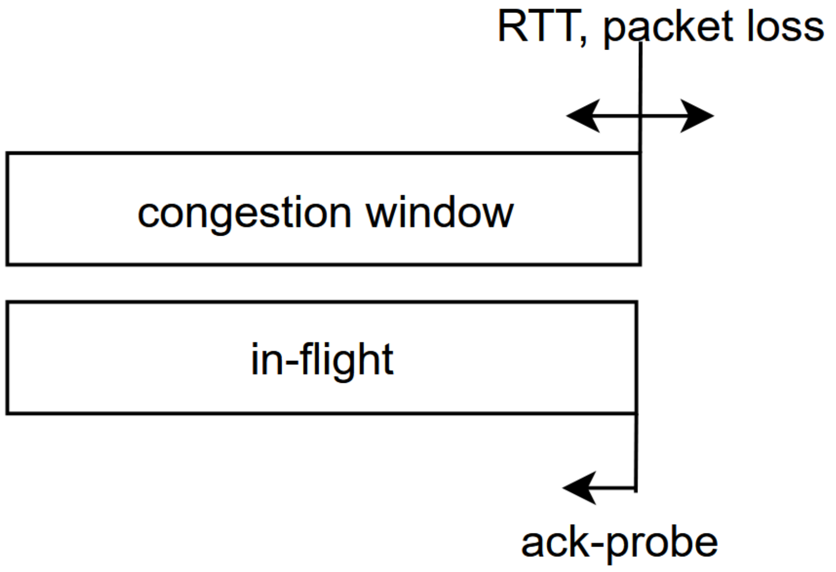
- Weight based scheduling
- Preffered path weight decrease on RTT increase over target RTT
- Regular weight adjustment
- Need for Mutexing
- Complex calculation of Timings



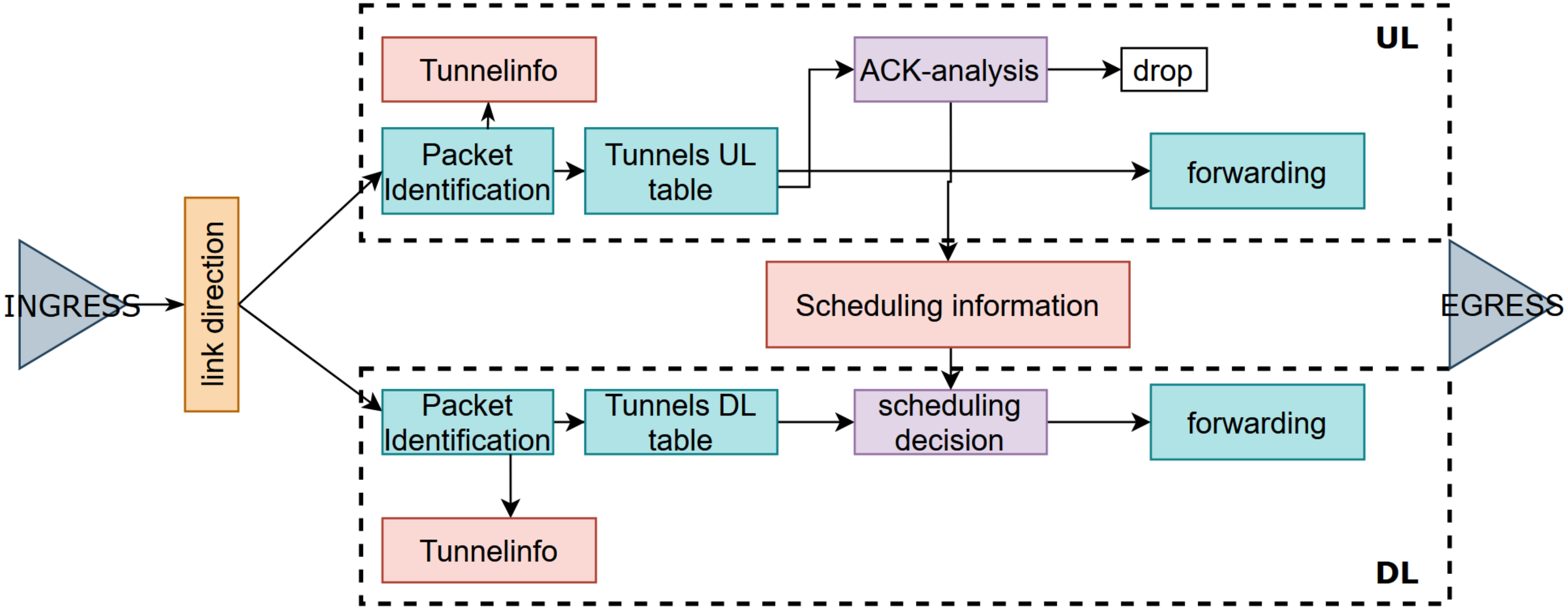

```
targetRTT = (minRTT * 1500) >> 10;
if(targetRTT == minRTT){
    targetRTT = targetRTT + 2;
}
if(shortenedTimestamp > lastScheduling + targetRTT){
    bit<6> currentWeight;
    path_weights.read(currentWeight,meta.index);
    if(currentRTT > targetRTT && currentWeight > 0){
        currentWeight = currentWeight - 1;
    }
    if(currentRTT <= targetRTT && currentWeight < 62){
        currentWeight = currentWeight + 1;
```

Congestion Monitoring Scheduling

- Extend RTTMon with Congestion Window Monitoring
- Not Weight-based
- Additional tunnel state variables (in-flight packets, probing)
- Detects tunnel outages (no relying on received ACKs for tunnel state)
- Extendable to multiple tunnels



Dataplane Pipeline with integrated Scheduler



Scheduling Information

- Central set of registers maintaining tunnel state
- Information used by Schedulers
  - Current RTT
  - Minimum RTT
  - In-flight packets (highest sequence - highest ack)
  - Interval for next scheduling change
- Data gathered on ACK receive

- Array Index per UE and Tunnel
- MP-DCCP Option for RTT measurement

Implementation – Key Challenges

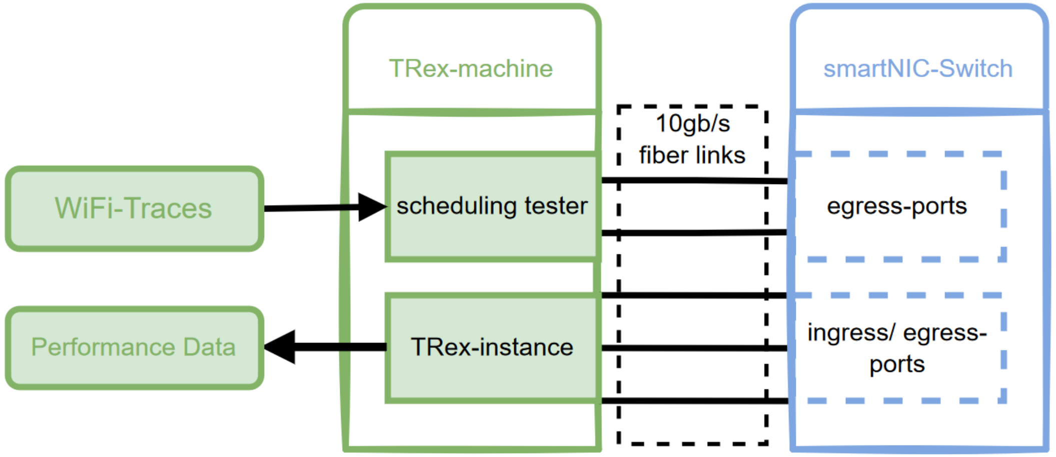
- P4/smartNIC limit available constructs, necessitating some „workarounds“
  - No loops, everything must be done during packet handling
  - Limited division support
    - P4 no support*
    - MicroC only for 32-bit

```
scaledTimestamp = meta.intrinsic_metadata.ingress_global_timestamp;
scaledTimestamp = scaledTimestamp >> 15;
scaledTimestamp = scaledTimestamp - (scaledTimestamp>>2);    //17
scaledTimestamp = scaledTimestamp - (scaledTimestamp>>2);    //19
scaledTimestamp = scaledTimestamp - (scaledTimestamp>>1);    //20
bit<32> shortenedTimestamp = scaledTimestamp[31:0];
```

  - No real number support
    - Ratios and factors must be approximated as integers
  - Mutex support only in MicroC

Performance Evaluation using Testbed

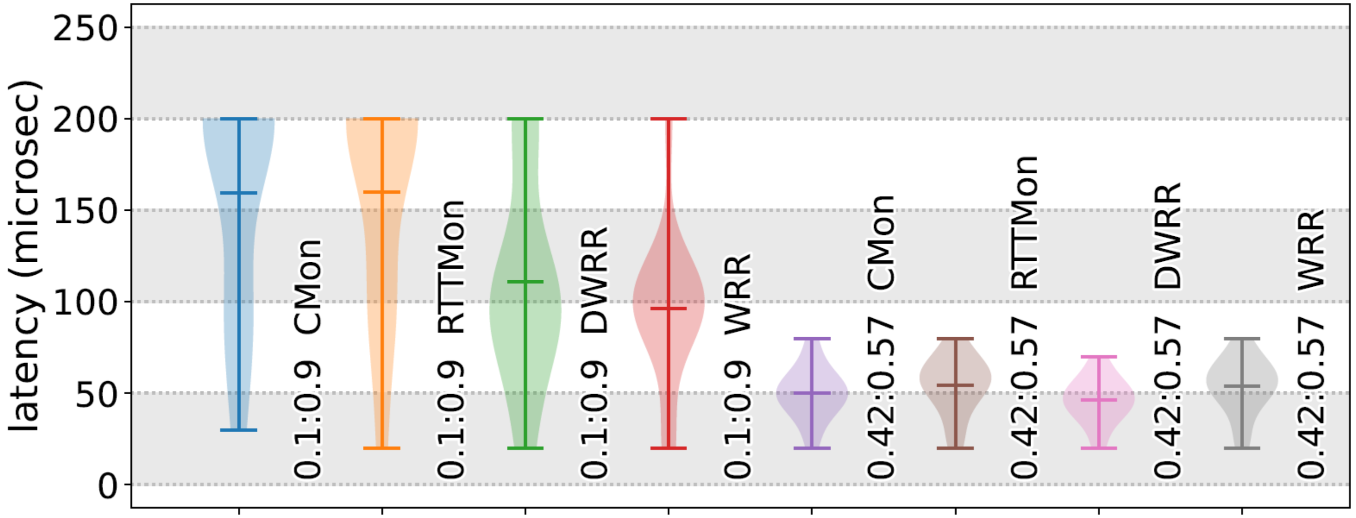
- Trex Traffic Generator
  - 20-core Intel Xeon Silver processor at 3.0GHz, 8 GB RAM. Intel X710 10G SFP+ NIC
  - Trex version v2.93 using DPDK version 21.02.0-rc1, running on Ubuntu 20.04.3 LTS using Linux Kernel 5.4.0-91-generic
- NFP machine (DUT)
  - 20-core Intel Xeon Silver processor at 3.0GHz, 8 GB RAM
  - Netronome Agilio CX NFP4000 processor, 60 (48 PPC, 12 FPC) Processing Cores @800 MHz, 8 threads each 2x40G port split into 8x10 G → 5 microsec base RTT
- Different test cases
  - Scheduling Validation: WiFi-traces for preferred path
  - Performance: Typical 5G - traffic composition



Performance Results

10 kUE
64-1400 B packets
Throughput in mpps

Test	WRR	DWRR	RTTMon	CMon
1:0(512B)	3.0	3.0	2.9	3.0
1:0(1024B)	2.9	2.9	2.8	3.0
0:1(512B)	5.3	5.4	4.5	4.6
0.42:0.57	4.2	4.1	3.7	3.8
0.4:0.6	4.3	4.2	3.8	3.9
0.25:0.75	4.7	4.5	4.1	4.2
0.1:0.9	4.7	4.6	4.2	4.2



Conclusions & Outlook

- P4 supports simple scheduling algorithms
  - Simple Packet analysis, saving tunnel state
  - Challenges due to limited functionality of language and hardware targets
- Scheduler Complexity has limited impact on proxy performance
  - Packet cloning has major performance impact on smartNIC platform

- Integration between user space and data-plane?
- Even more complex schedulers?


thank you
danke
謝謝
ngiyabonga
شكراً جزيلاً
teşekkür ederim
gracias
merci
obrigado
dziękuję
bedankt
спасибо
Баярлалаа
рахмат
mersi
barka
welalin
tack
spas
kia ora
dank je
misaotra
matondo
paldies
grazzi
mahalo
tapadh leat
faafetai lava
vinaka
blagodaram
спасибі
kiitos
dankie
dhanyavad
nanni
nandri
hvala
maururu
köszönöm
enkosi
bayarlalaa
gràcie
akun dankon
aciú
хвала
asante
manana
obrigada
tenki
djiere dieuf
tau
mochchakkeram
murakoze
дякую
mamnun
chokrane
go raibh maith agat
chnorakaloutioun
grafias ago
gràcies
sulpáy
taiku
sukriya
kop khun krap
sobodi
dėkuji
sagolun
arigatô
takk
dakujem
trugarez
mèsi
ありがとう
tanemirt
grazie
didi madloba
najis tuke
rahmet
terima kasih
kam sah hamnida
rahmat
diolch
dhanyavadagalu
shukriya
merce
мерси
তোমাকে ধন্যবাদ
감사합니다
xièxie
ευχαριστώ