RESTRUCTURING ENDPOINT CONGESTION CONTROL

Akshay Narayan, Frank Cangialosi, Deepti Raghavan, Prateesh Goyal,
Srinivas Narayana, Radhika Mittal*, Mohammad Alizadeh, Hari Balakrishnan

MIT CSAIL, *UC Berkeley

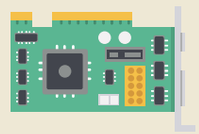
CONGESTION CONTROL

APPLICATION

Data

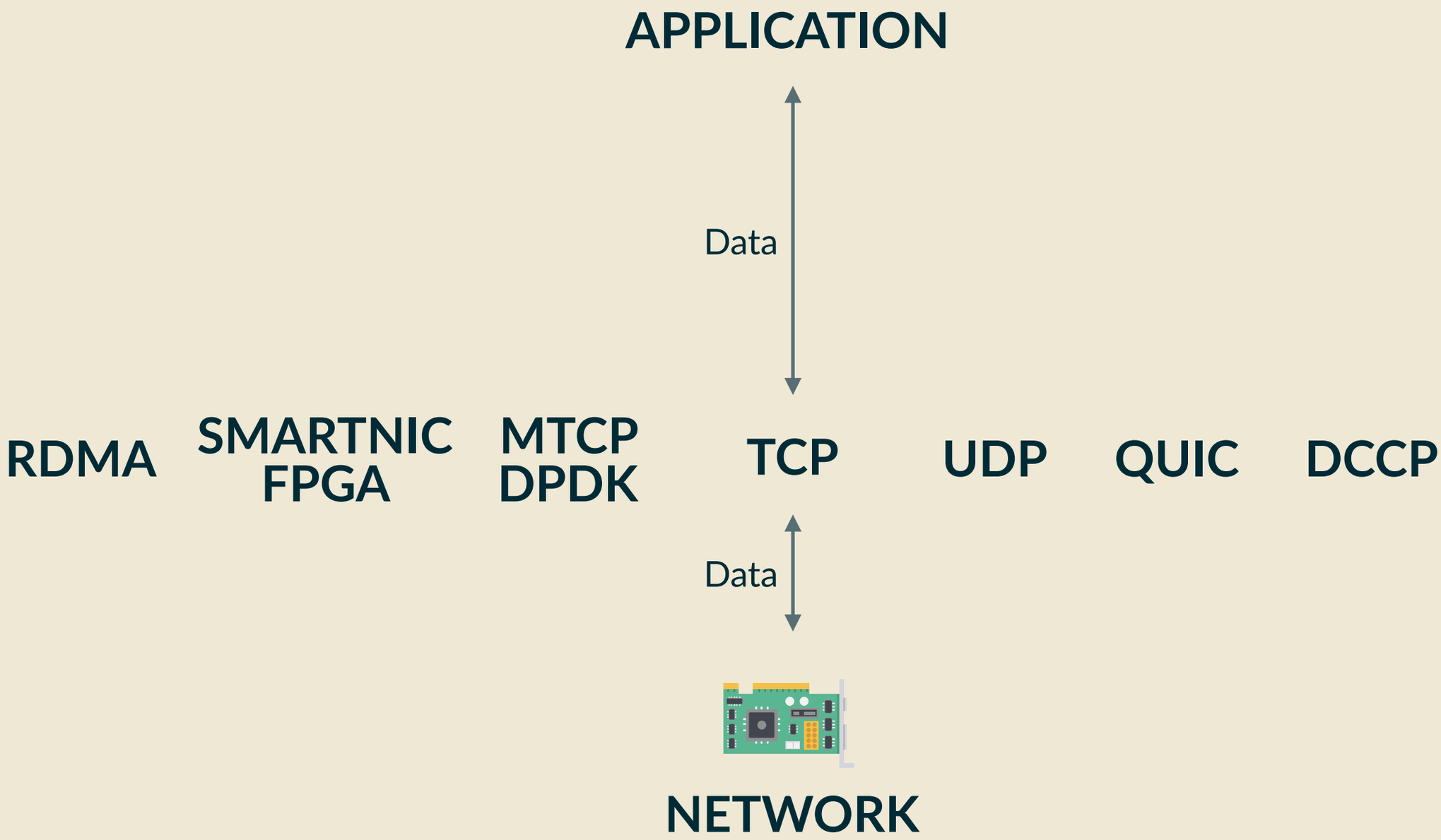
TCP

Data



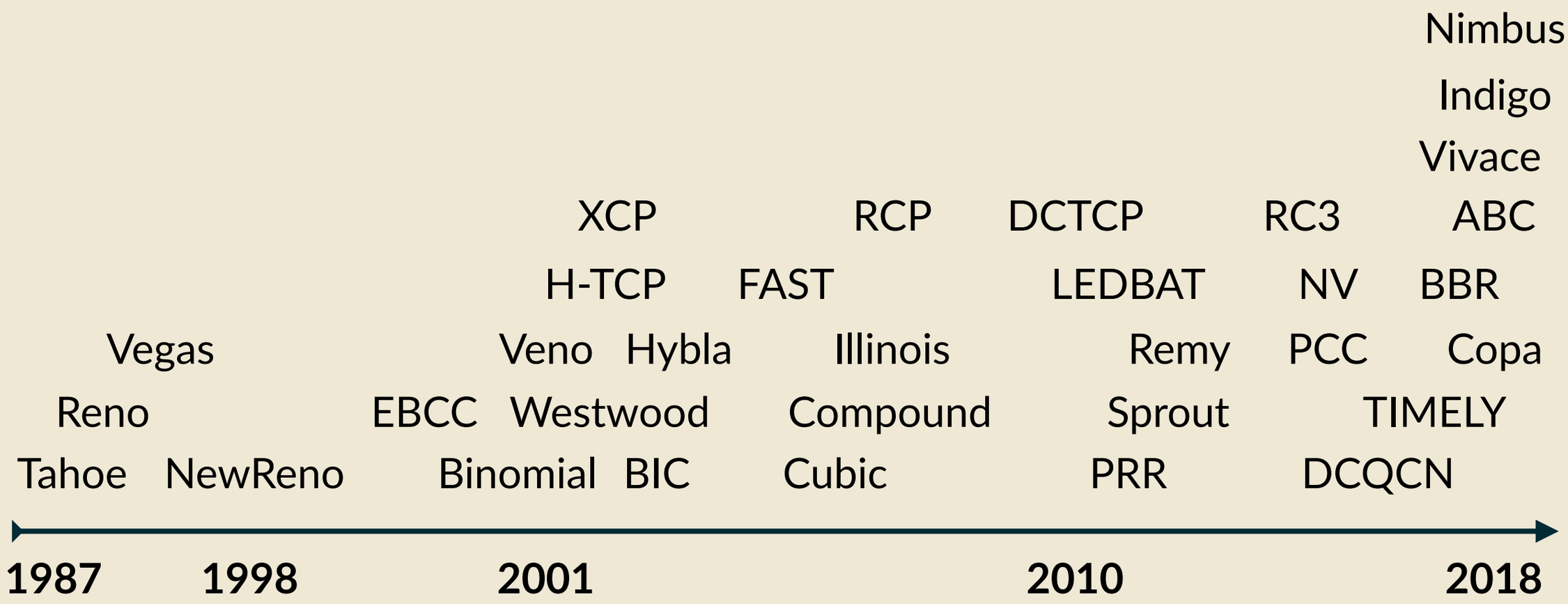
NETWORK

DATAPATHS


APPLICATION
Data
RDMA
SMARTNIC
FPGA
MTCP
DPDK
TCP
UDP
QUIC
DCCP
Data
NETWORK

NEW ALGORITHMS

Sprout (NSDI 2013): Bayesian forecasting
Remy (SIGCOMM 2013): Offline learning
PCC / PCC Vivace (NSDI 2015 / NSDI 2018): Online learning
Indigo (ATC 2018): Reinforcement learning
Nimbus (Arxiv): Fourier transforms



CROSS PRODUCT OF SADNESS

RDMA

SMARTNIC
FPGA

MTCP
DPDK

TCP

UDP

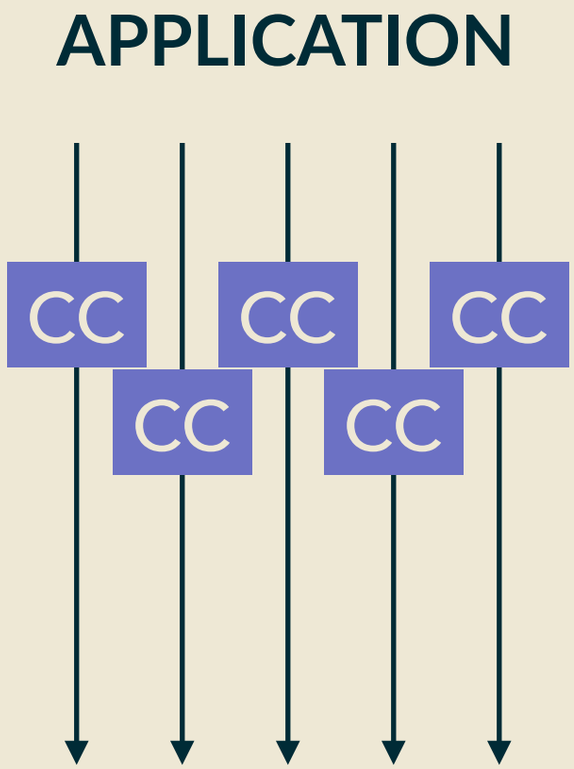
QUIC

DCCP

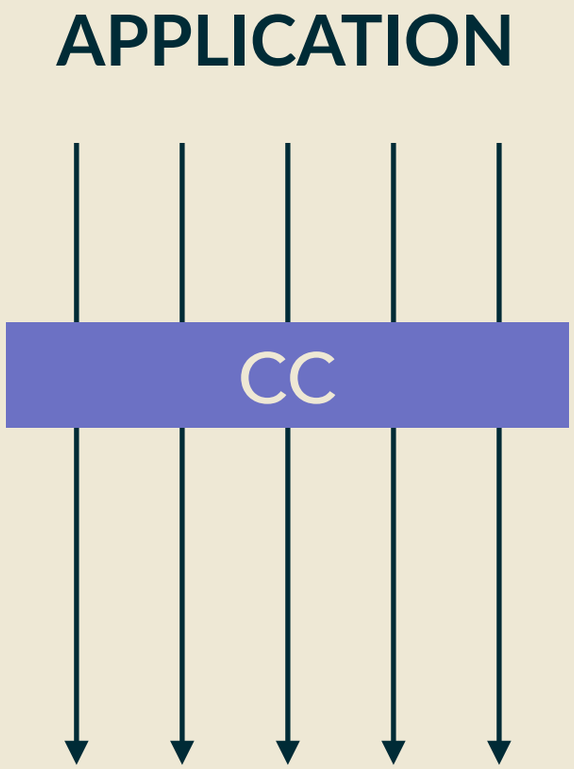
X

H-TCP Veno Hybla
TIMELY XCP Westwood
Compound Sprout EBCC BIC
Cubic PRR Binomial Nimbus
DCQCN Reno Vegas Indigo
Tahoe NewReno Vivace
RCP DCTCP RC3 ABC
FAST LEDBAT NV BBR
Illinois Remy PCC Copa

NEW CAPABILITIES

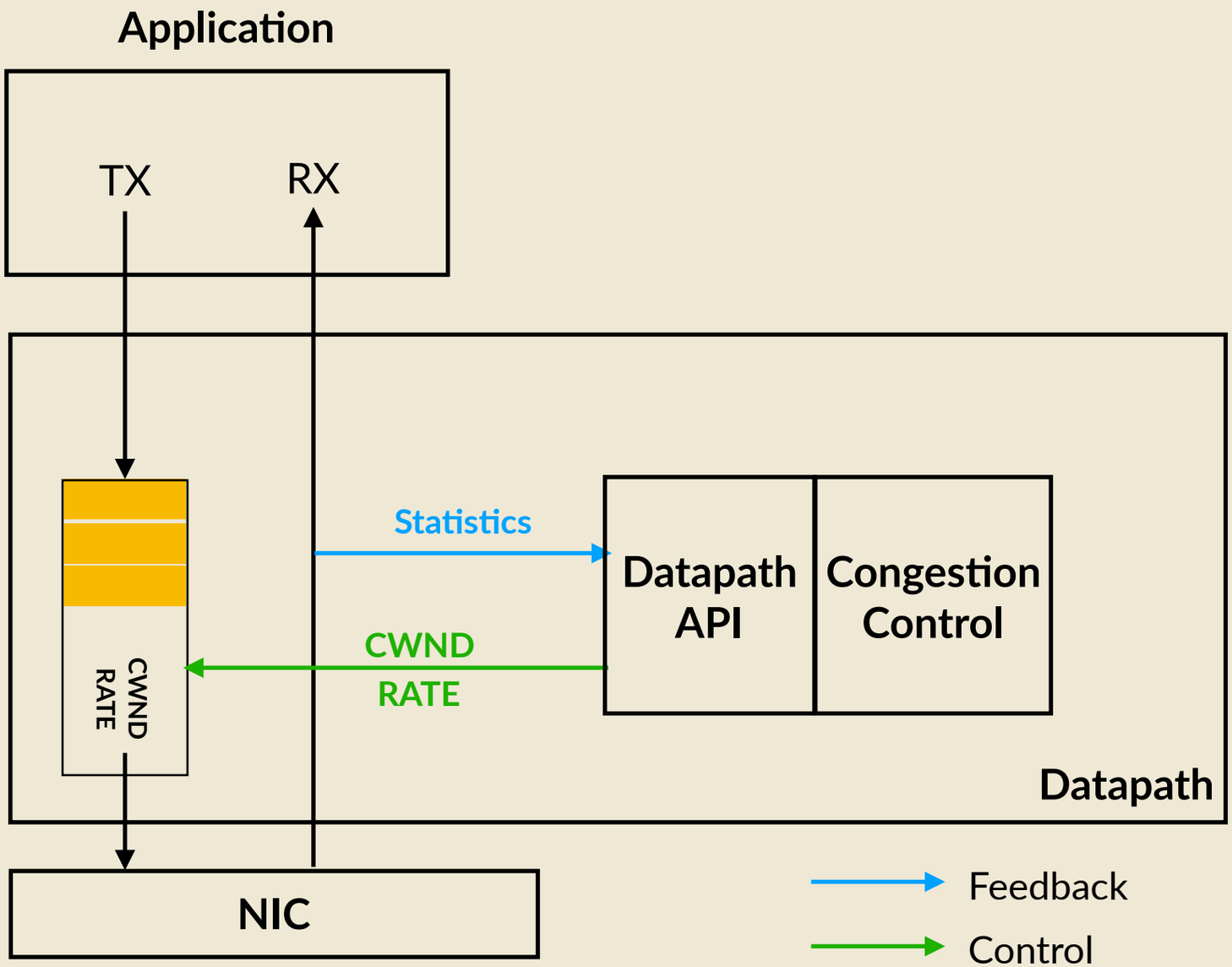


INDEPENDENT CC

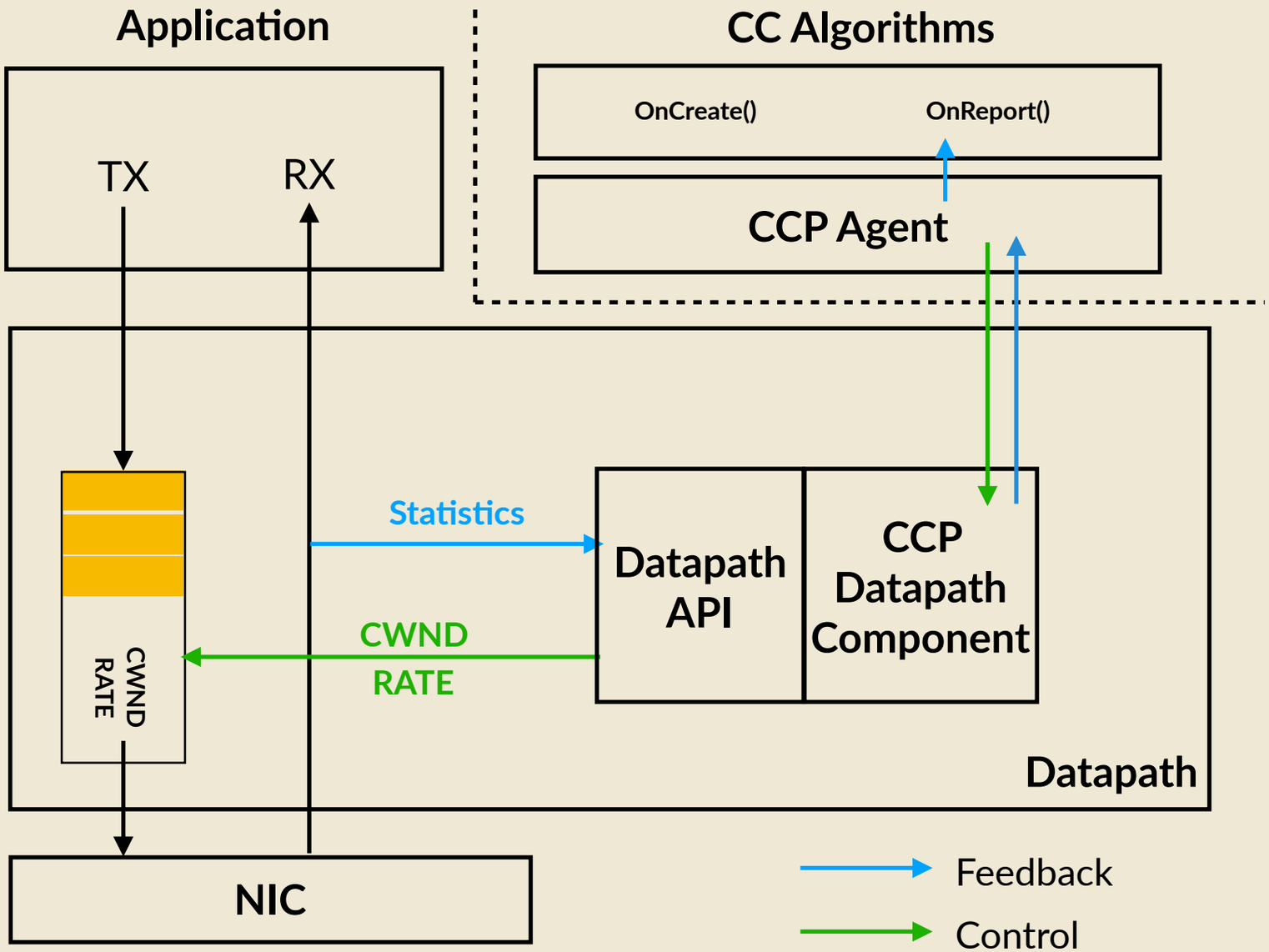


AGGREGATE CC

CURRENT DESIGN

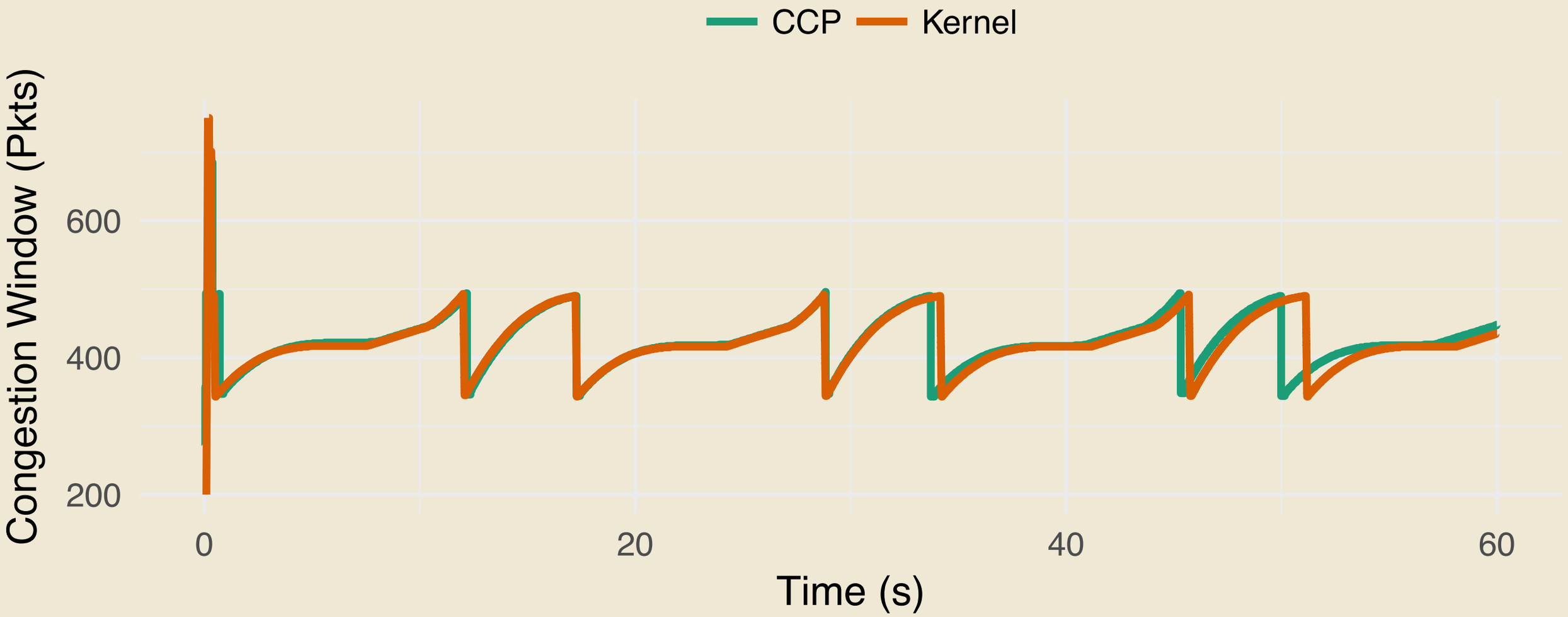


CONGESTION CONTROL PLANE



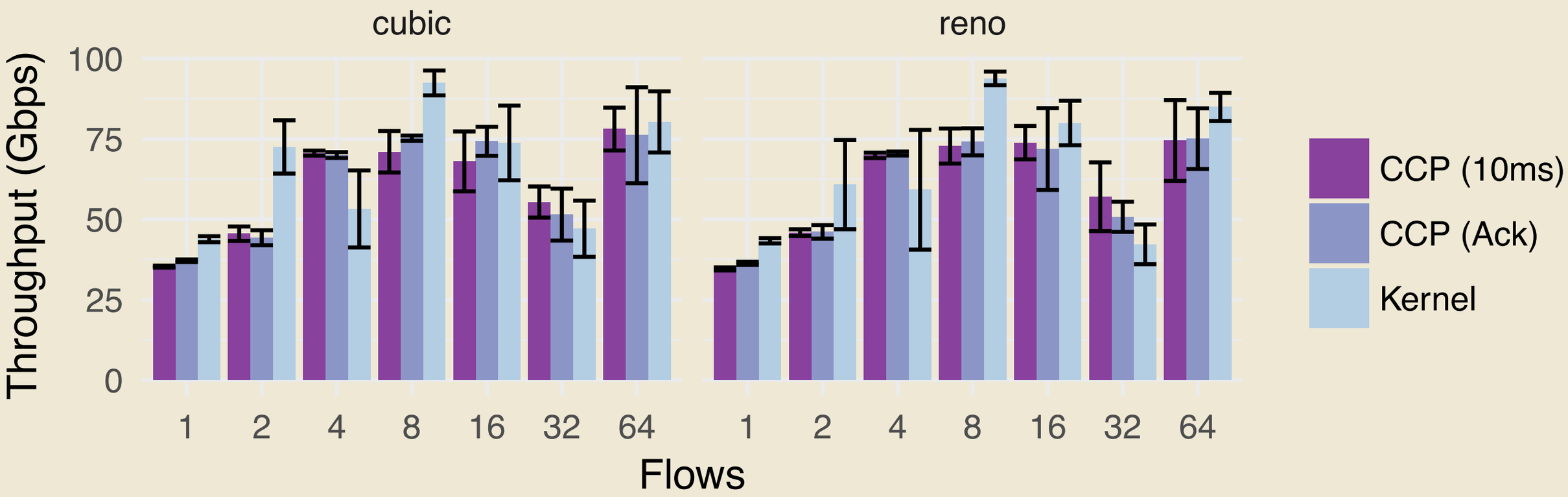
Demo

CONGESTION WINDOW DYNAMICS

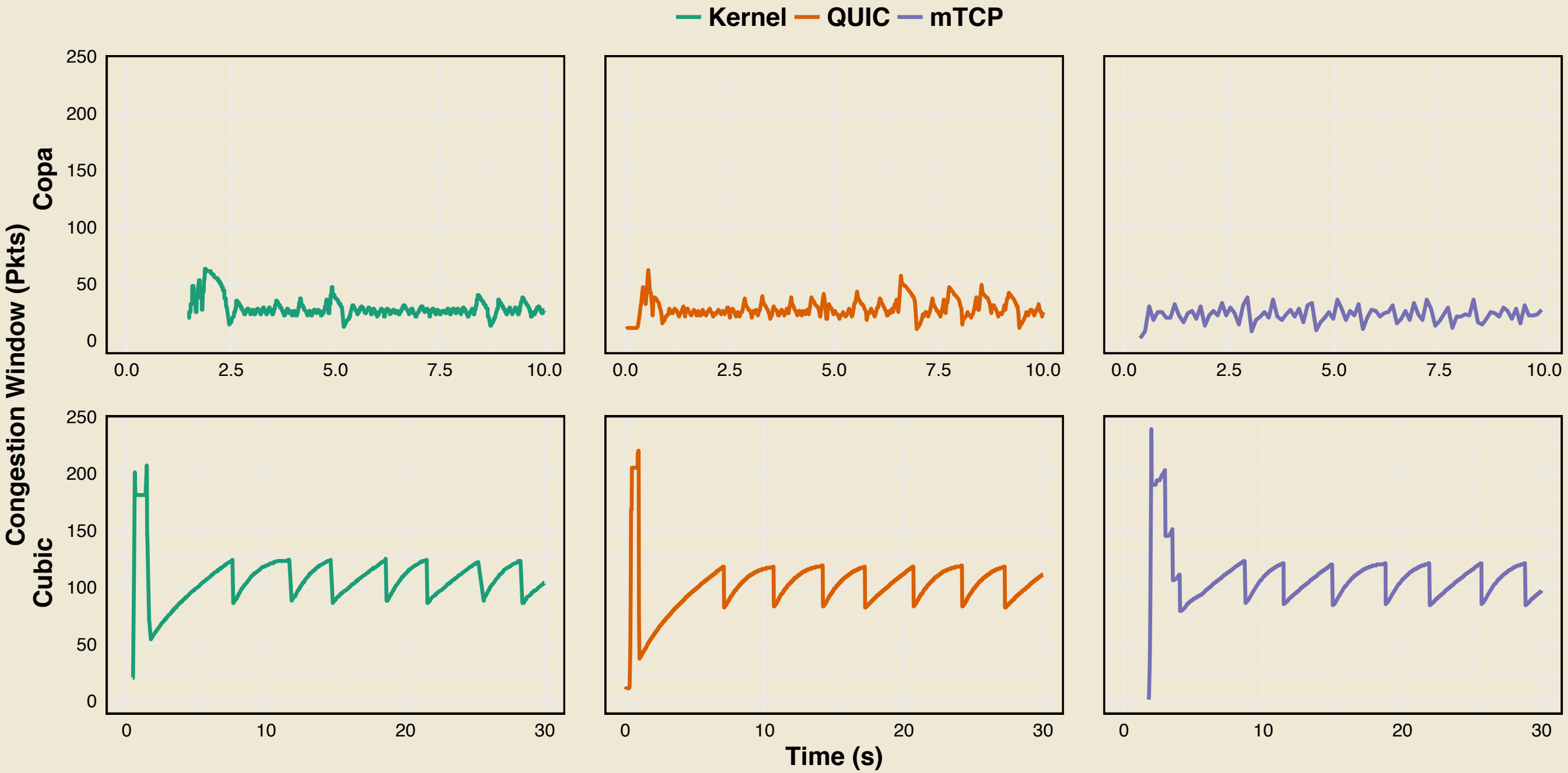


96 Mbit/s, 20ms link RTT

OVERHEAD

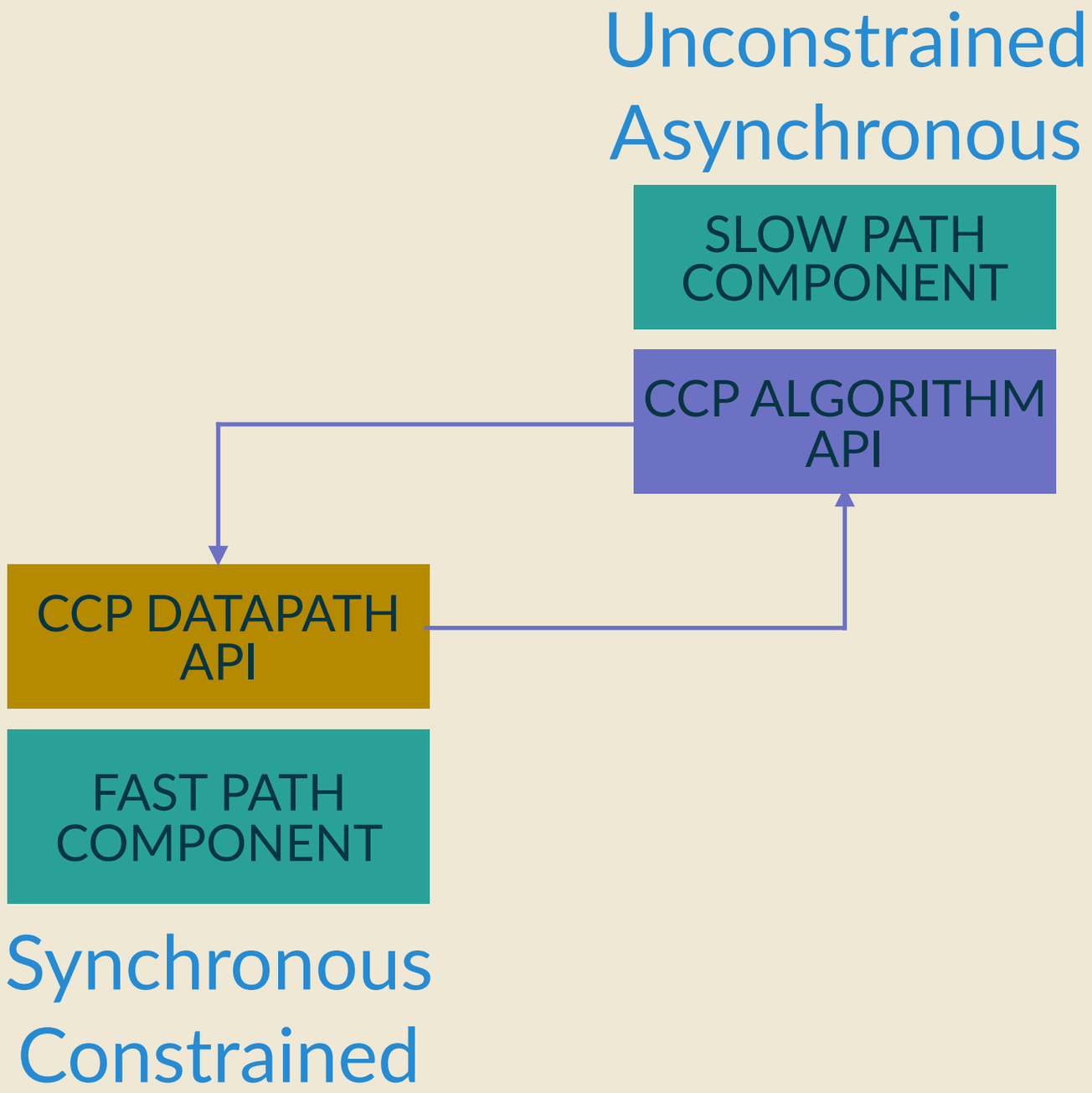

cubic
reno
Throughput (Gbps)
100
75
50
25
0
1
2
4
8
16
32
64
Flows
CCP (10ms)
CCP (Ack)
Kernel

WRITE-ONCE RUN-ANYWHERE



24 Mbit/s, 20ms link RTT

DESIGN: FAST AND SLOW PATH



ALGORITHM API

Event Handler `def OnReport(info):`

ALGORITHM API

Event Handler `def OnReport(info):`

State Update `cwnd += info.acked / cwnd;`

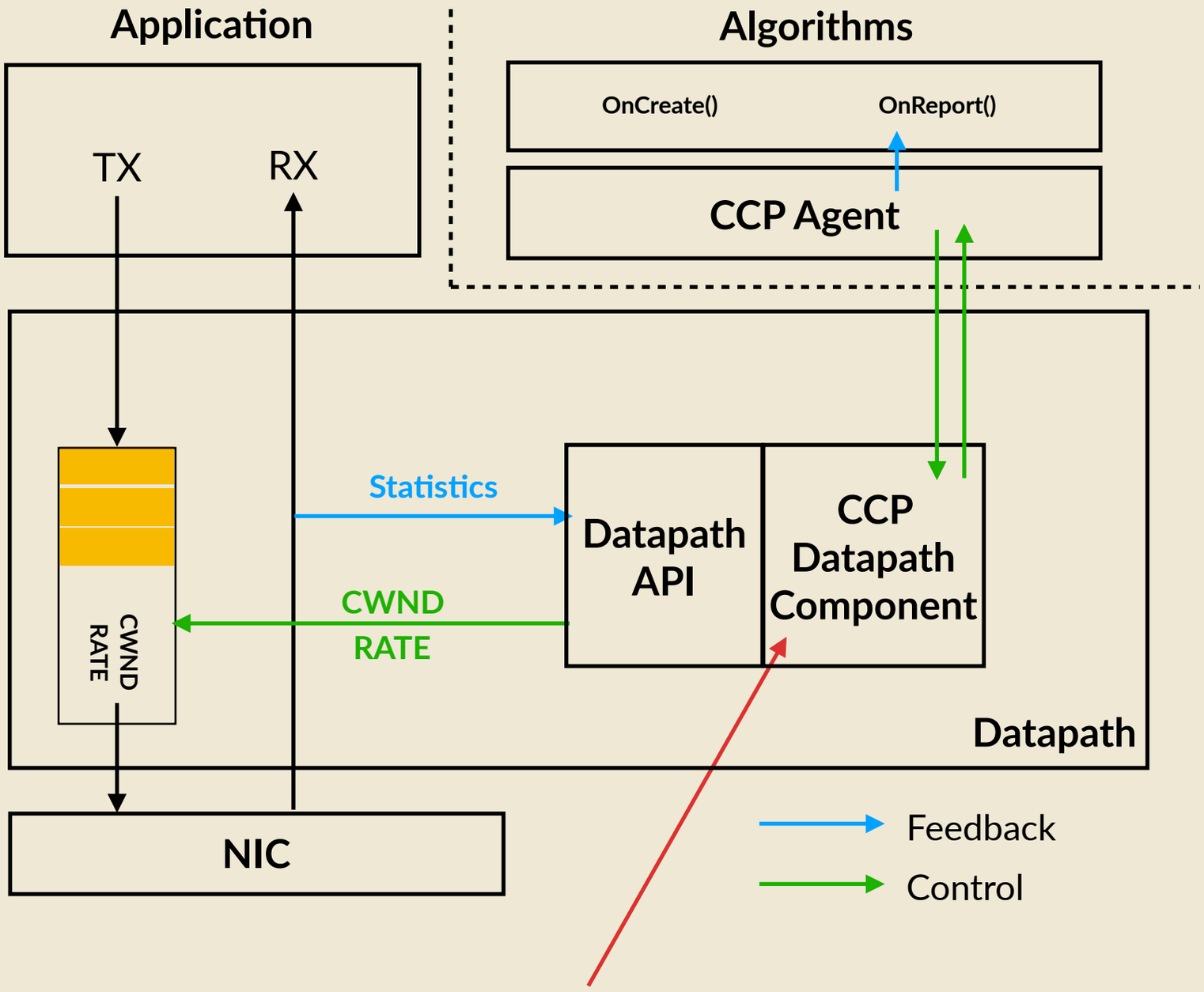
ALGORITHM API

Event Handler `def OnReport(info):`

State Update `cwnd += info.acked / cwnd;`

Decision `datapath.update([“Cwnd”, cwnd]);`

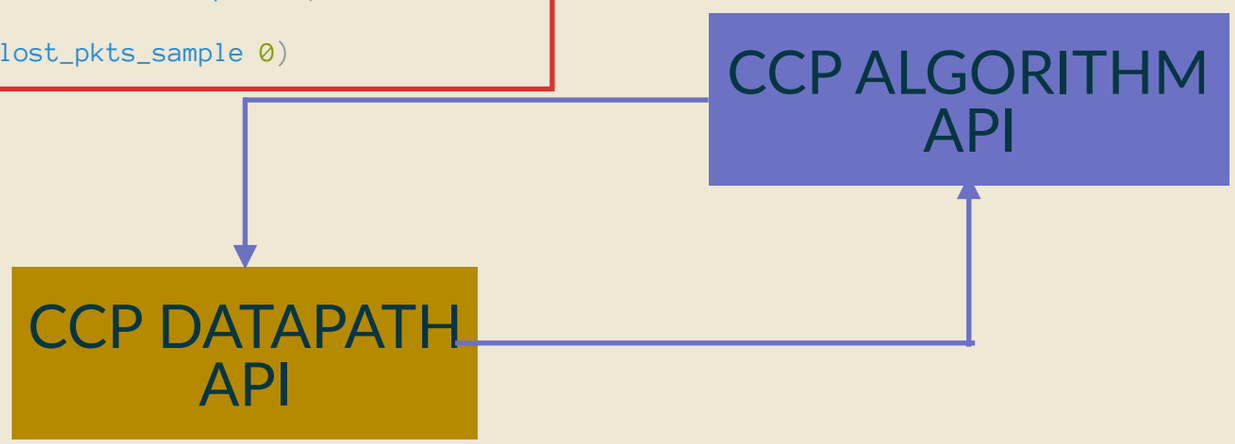
DATAPATH PROGRAM



Synchronous component

DATAPATH PROGRAMS

```
(def (Report (acked 0)))
(when true
  (:= Report.acked (+ Report.acked
Ack.bytes_acked))
  (fallthrough))
(when (> Micros Flow.rtt_sample_us)
  (report))
(when (> Flow.lost_pkts_sample 0)
```



DATAPATH PROGRAMS

Congestion Avoidance

```
(def (Report (acked 0)))
(when true
  (:= Report.acked (+ Report.acked Ack.bytes_acked))
  (fallthrough))
(when (> Micros Flow.rtt_sample_us)
  (report))
(when (> Flow.lost_pkts_sample 0)
  (report))
```

DATAPATH PROGRAMS

Slow Start

```
(def (Report (acked 0)))
(when true
  (:= Report.acked (+ Report.acked Ack.bytes_acked))
  (:= Cwnd (+ Cwnd Report.acked))
  (fallthrough))
(when (> Flow.lost_pkts_sample 0)
  (report))
```

CONGESTION SIGNALS

Signal	Definition
Ack.{bytes,packets}_acked	Δ(tcp_sock.bytes_acked)
Ack.{bytes,packets}_misordered	Δ(tcp_sock.sacked_out)
Ack.ecn_{bytes,packets}	count in_ack_event(): CA_EVENT_ECE
Ack.lost_pkts_sample	rate_sample.losses
Ack.now	getnstimeofday()
Flow.was_timeout	set_state(): TCP_CA_LOSS
Flow.rtt_sample_us	rate_sample.rtt_us
Flow.{bytes,packets}_in_flight	tcp_packets_in_flight(tcp_sock)
Flow.rate_incoming	rate_sample.delivered / rate_sample.rcv_int_us
Flow.rate_outgoing	rate_sample.delivered / rate_sample.snd_int_us

NEXT STEPS

- Distribute CCP in-tree
- Hardening for scale
- More algorithms!
- Cluster aggregation CCP

CURRENT STATUS

- Datapaths (libccp):
  - Linux TCP
  - QUIC
  - mTCP/DPDK
- CCP Agent (portus)

github.com/ccp-project

Extra Slides

EBPF

Front-End
(Language)

- Event-driven semantics
- Explicit reporting model

Back-End
(Datapath)

- Congestion control enforcement
- Direct access to socket state

```
(def (Report (acked 0)))
(when true
  (:= Report.acked (+ Report.acked Ack.bytes_acked))
  (:= Cwnd (+ Cwnd Report.acked))
  (fallthrough))
(when (> Flow.lost_pkts_sample 0)
  (report))
```