RESTRUCTURING ENDPOINT CONGESTION CONTROL

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

MIT CSAIL, *UC Berkeley
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

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

APPLICATION

INDEPENDENT CC

APPLICATION

AGGREGATE CC
CURRENT DESIGN

![Diagram showing the current design of a networking system. The diagram includes components such as Application, TX, RX, NIC, Datapath, CWND, RATE, Feedback, and Congestion Control.]
CONGESTION CONTROL PLANE

Application

TX  RX

NIC

CC Algorithms

Datapath

CCP Agent

Datapath API

CCP Datapath Component

Datapath

OnCreate()  OnReport()

OnCreate()

OnReport()

Statistics

CWND

RATE

Feedback

Control
Demo
CONGESTION WINDOW DYNAMICS

Time (s)

Congestion Window (Pkts)

CCP

Kernel

96 Mbit/s, 20ms link RTT
WRITE-ONCE RUN-ANYWHERE

24 Mbit/s, 20ms link RTT
DESIGN: FAST AND SLOW PATH

- FAST PATH COMPONENT
  - CCP DATAPATH API
  - FAST PATH COMPONENT

- SLOW PATH COMPONENT
  - CCP ALGORITHM API
  - SLOW PATH COMPONENT

- Asynchronous
- Unconstrained
- Constrained
Event Handler  

```python
def OnReport(info):
```
ALGORITHM API

Event Handler    def OnReport(info):
State Update     cwnd += info.acked / cwnd;
def OnReport(info):
    datapath.update(['Cwnd', cwnd]);
    cwnd += info.acked / cwnd;
    datapath.update(['Cwnd', cwnd]);
DATAPATH PROGRAM

Synchronous component
(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))
(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

<table>
<thead>
<tr>
<th>Signal</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ack.{bytes,packets}_acked</td>
<td>(\Delta (\text{tcp_sock.bytes_acked}))</td>
</tr>
<tr>
<td>Ack.{bytes,packets}_misordered</td>
<td>(\Delta (\text{tcp_sock.sacked_out}))</td>
</tr>
<tr>
<td>Ack.ecn_{bytes,packets}</td>
<td>count in_ack_event(): CA_EVENT_ECE</td>
</tr>
<tr>
<td>Ack.lost_pkts_sample</td>
<td>rate_sample.losses</td>
</tr>
<tr>
<td>Ack.now</td>
<td>getnstimeofday()</td>
</tr>
<tr>
<td>Flow.was_timeout</td>
<td>set_state(): TCP_CA_LOSS</td>
</tr>
<tr>
<td>Flow.rtt_sample_us</td>
<td>rate_sample.rtt_us</td>
</tr>
<tr>
<td>Flow.{bytes,packets}_in_flight</td>
<td>tcp_packets_in_flight(tcp_sock)</td>
</tr>
<tr>
<td>Flow.rate_incoming</td>
<td>rate_sample.delivered / rate_sample.rcv_int_us</td>
</tr>
<tr>
<td>Flow.rate_outgoing</td>
<td>rate_sample.delivered / rate_sample.snd_int_us</td>
</tr>
</tbody>
</table>
**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](http://github.com/ccp-project)
Extra Slides
EBPF

- Event-driven semantics
- Explicit reporting model

Front-End (Language)

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))
```