

# Presenting C4 (Christian's CC Code)

Christian Huitema

CCWG Meeting at IETF 125, Shenzhen

March 18, 2026

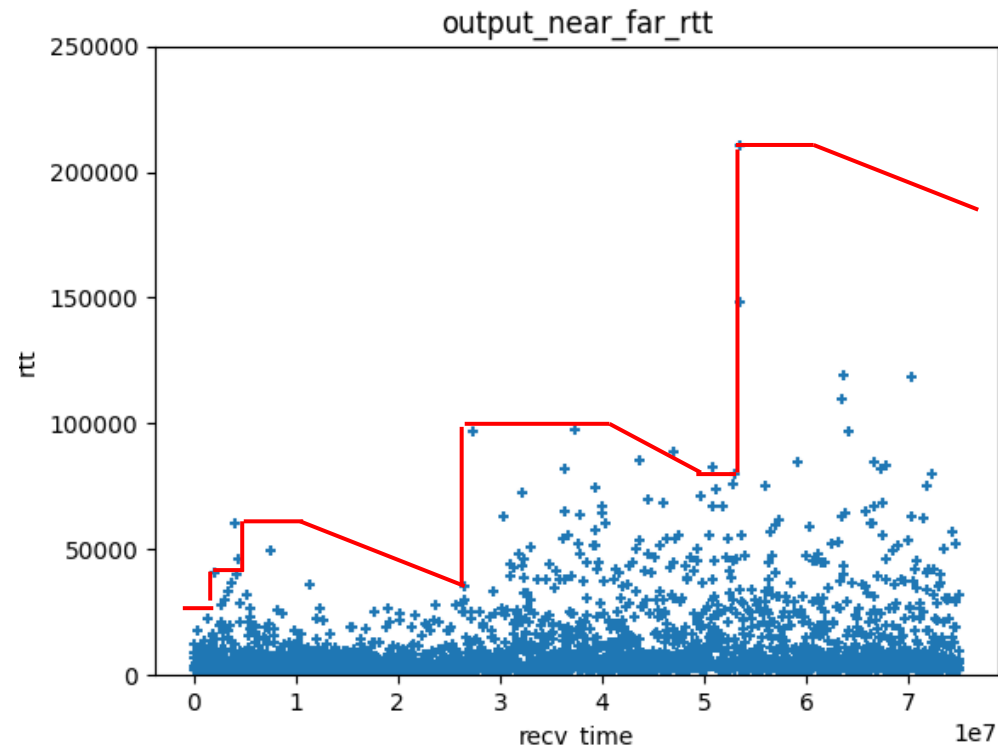
# Initial goals

- Works well with video
  - Good support for “application limited” flows
  - Stable throughput
  - No buffer-bloat
  - Low delays
- Work well with common access networks
  - Including questionable Wi-Fi networks
- Keep it simple

# C4, handling of congestion signals

Signal	Issues	Choices
ECN	<ul style="list-style-type: none"><li>• Not always there!</li></ul>	<ul style="list-style-type: none"><li>• React to ECN-CE rate</li><li>• Mix with other signals</li></ul>
Delays and timers	<ul style="list-style-type: none"><li>• Key link types have variable RTT or high jitter</li></ul>	<ul style="list-style-type: none"><li>• Track Max RTT</li><li>• Do not react to timer much</li></ul>
Packet loss	<ul style="list-style-type: none"><li>• Timers are not reliable</li><li>• Number of losses grows with data rate</li></ul>	<ul style="list-style-type: none"><li>• Assume FACK, do not react on PTO timers</li><li>• Track average loss rate, not individual losses</li></ul>
Data Rate	<ul style="list-style-type: none"><li>• Easy to get wrong in presence of delay jitter</li><li>• Easy to mistake reduce bandwidth vs. app limited</li></ul>	<ul style="list-style-type: none"><li>• Track Data Rate, use for pacing</li><li>• Filter measurements (see BBR)</li><li>• Do not rely on bandwidth limited estimation</li></ul>

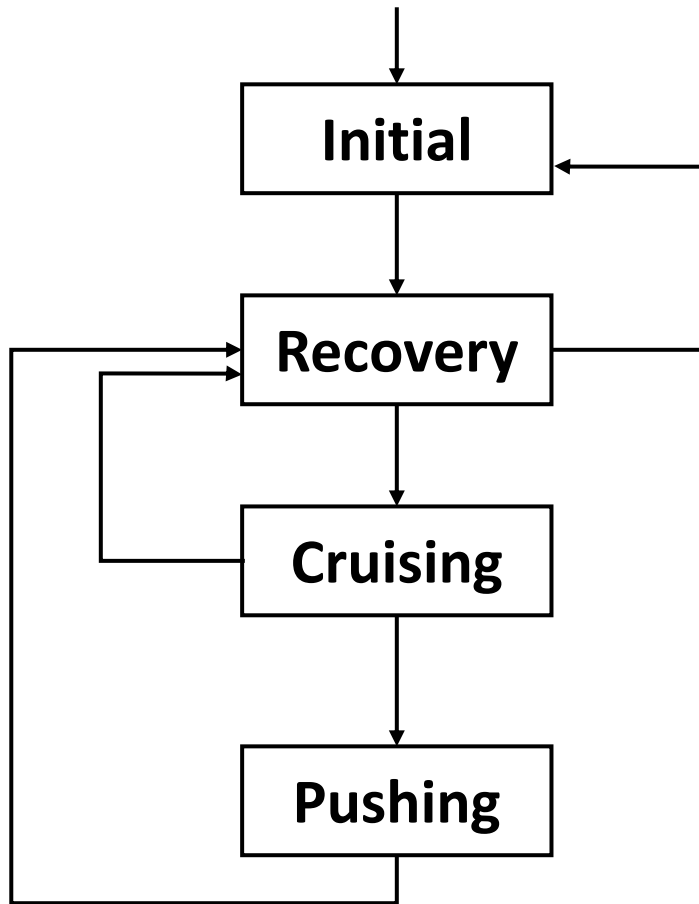
# C4 design choice: track max RTT, data rate



$CWND = Max\_RTT * rate$  (--) vs. bytes in flight (+)

- Measure rate – similar to BBR
- Set CWND large enough
  - $Max\_RTT * Rate$
- Pacing Rate as control:
  - $Bytes\_in\_flight = pacing\_rate * Max(max\_RTT, actual\_RTT)$
- Only measure max RTT when
  - $Pacing\_rate \leq Rate$
- Reduce rate if congestion
- Decay Max RTT over time

# C4 design choice: periodic probe



- Initial state
  - Initial evaluation
- Recovery
  - After congestion or pushing
  - Assess rate
- Cruising
  - Send at nominal rate
  - Assess max RTT
  - Detect congestion
- Pushing
  - Send X% higher than maximum rate

# How to fix the probing rate?

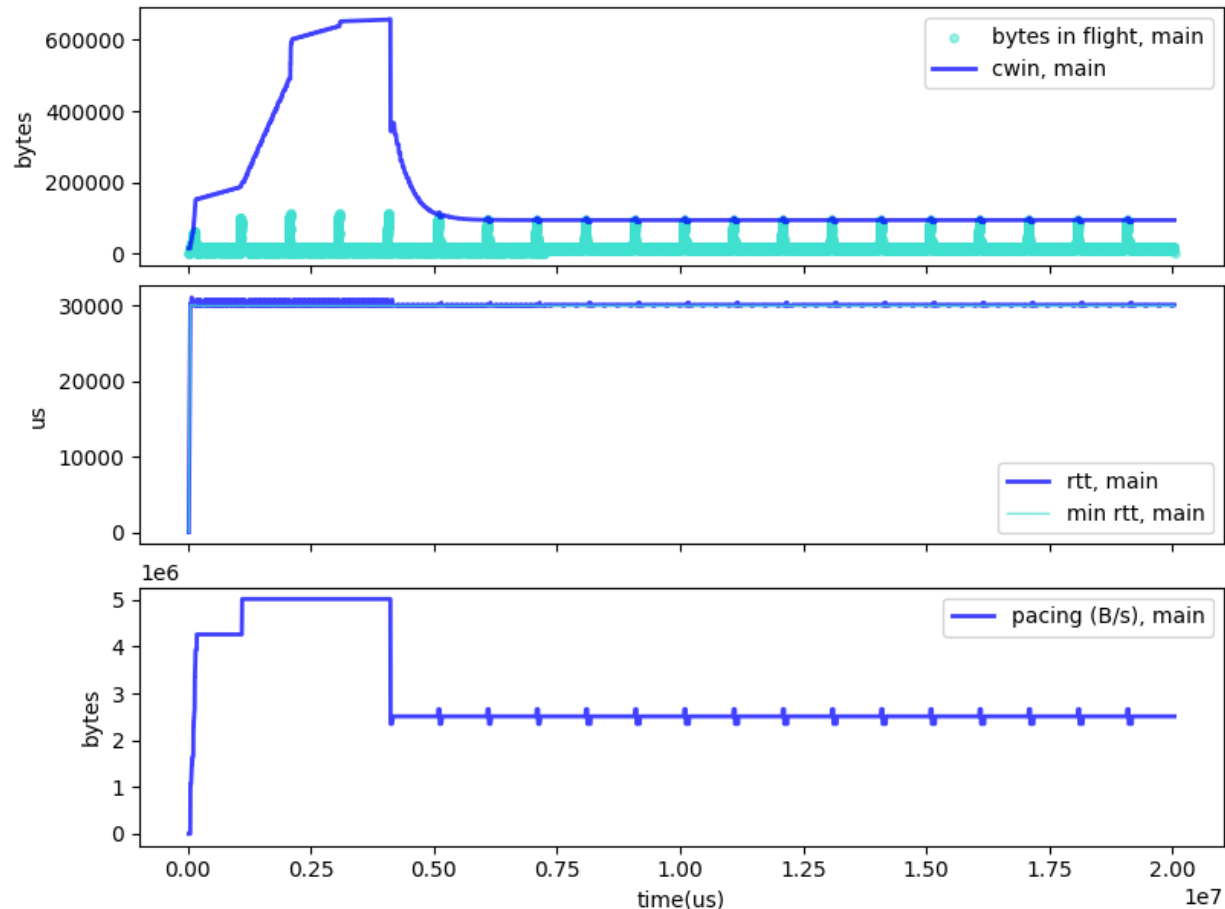
- Cycle: recovery, N cruising RTT, Pushing
  - When pushing: rate  $\ast = \alpha$
  - During recovery: measure observed delivery rate
  - Average growth rate:  $(\alpha - 1) / (1 + N + 1)$
- Tension: how to fix alpha?
  - Too small:
    - Slow reaction to “bandwidth change” events
  - Too big:
    - Packet loss and queues during Pushing
    - Very high CE marking by L4S during Pushing

# C4 design Choice: probing cascade

Probe Level	Rate increase	Nb. Cruising
0	1/32	1
1	1/16	4
2	1/4	1
3	1/4	1

- After Initial, start at level 1
- If rate measurement increased by at least  $\frac{1}{2}$  of probing rate increase, success
- If success, move to next level
- If success and level was 4:
  - Go back to initial
- If no success:
  - If level 0: stay there.
  - Else excess CE mark: move to level 0
  - Else if level 1: stay there
  - Else move down 1 level

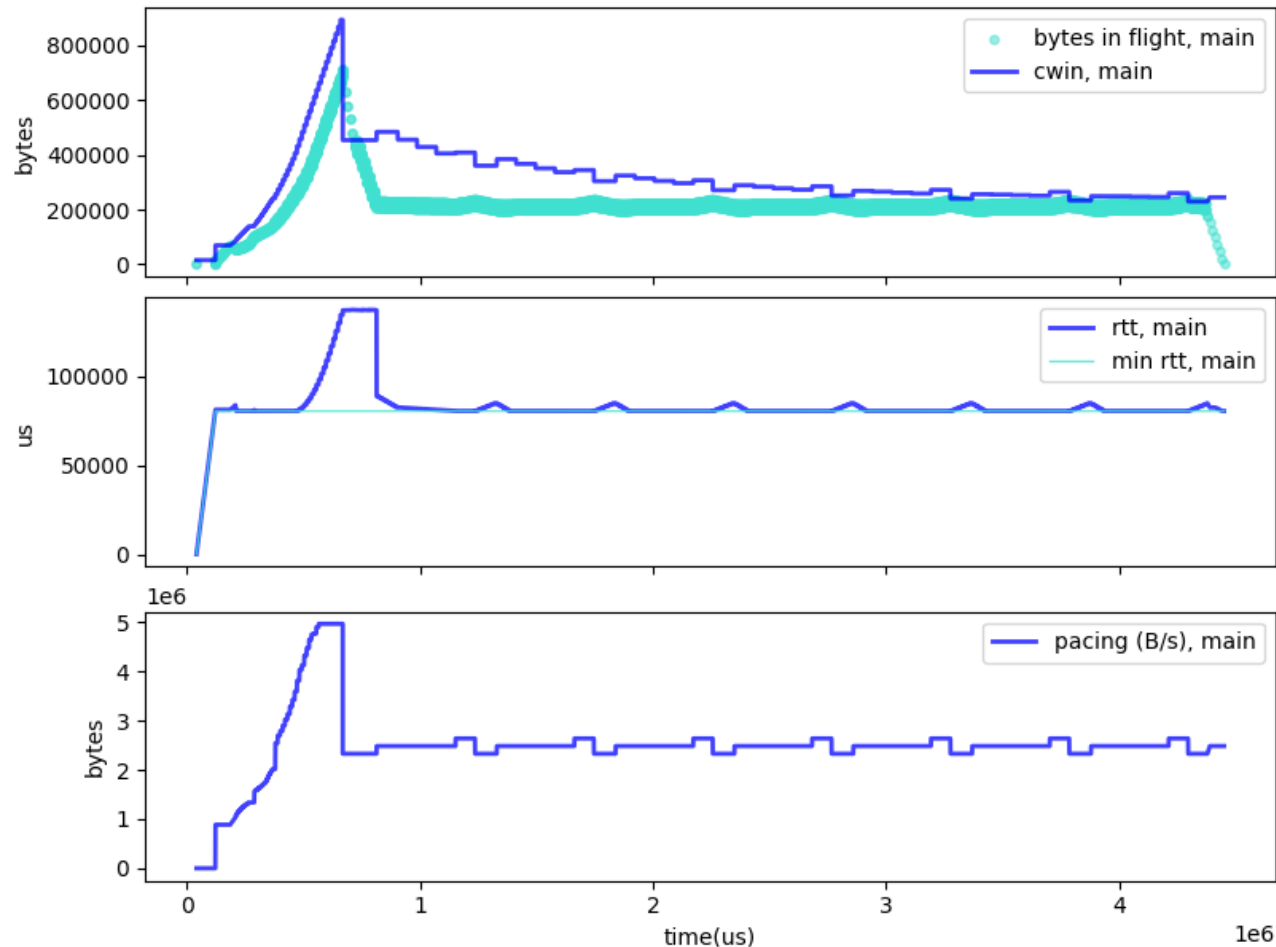
# Example: multimedia connection



- Media test, 20 seconds of audio and video
- RTT remains pegged at 30ms
- Data rate matches path
- Everything is just fine...

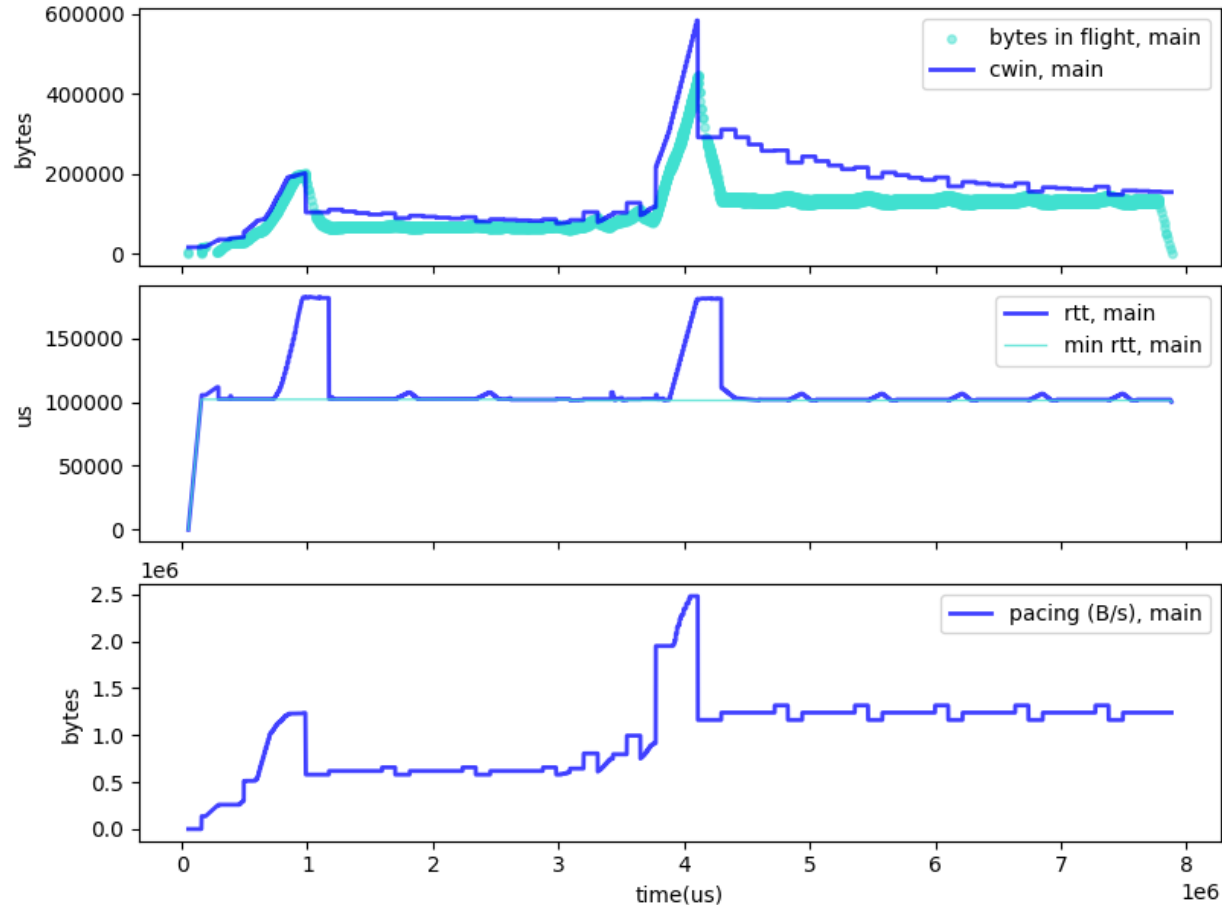
***Yes, CWIN gets too big during Initial. Working on it.***

# Example: short connection using ECN



- Short download on 20 Mbps path, 80ms RTT
- L4S style AQM
- C4 announces ECT1 (L4S support)
- Works exactly as expected, low delays after Initial phase.
- Remains at probe level 0

# Example of cascade: low and up test



- At T=3s, the bandwidth increases from 5 Mbps to 10 Mbps.
- After the next probe succeeds, the cascade kicks in and the new capacity is evaluated.
- The connection stabilizes back to almost no queuing.

***Yes, CWIN, Initial...***

# C4 design choice: multiplicative increase

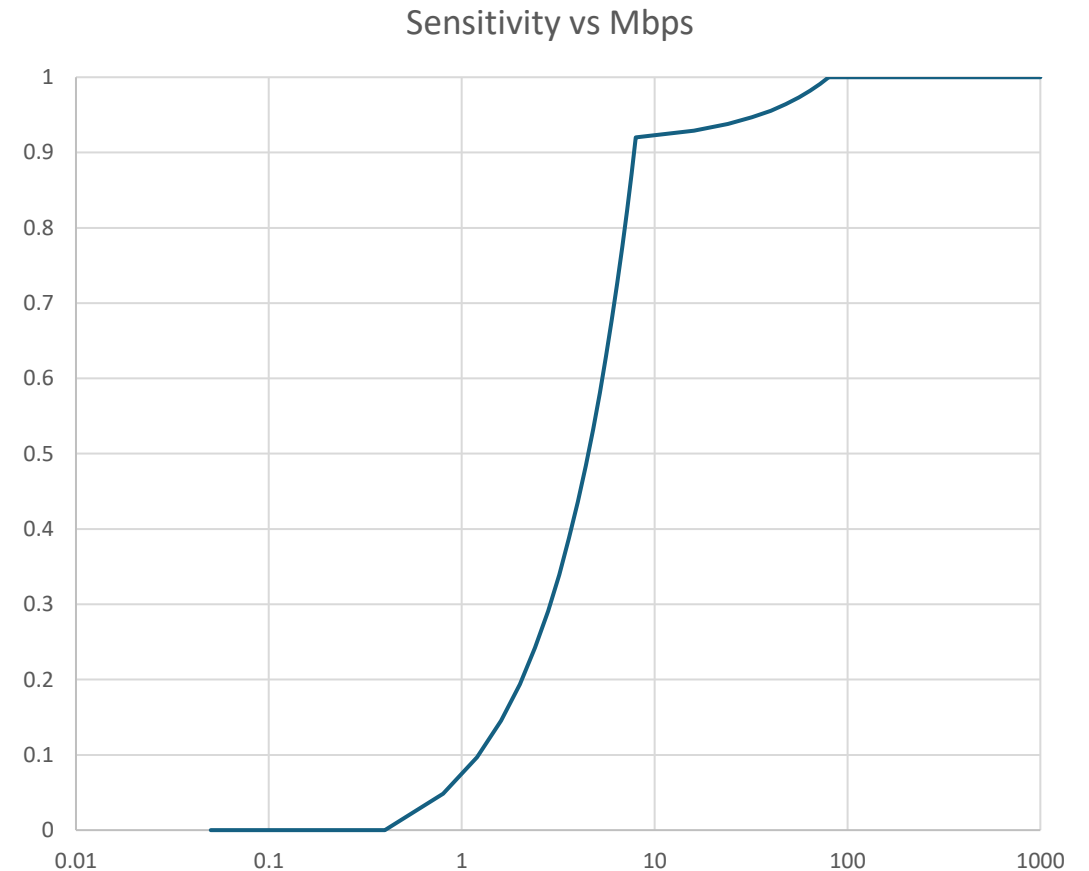
- Classic congestion control use additional increase (AIMD)
  - high rate flow increase slower than low rate flow
  - Leads to asymptotic “fair sharing” equilibrium
  - Also leads to scaling issues
    - See RFC 3649, High Speed TCP for Large Congestion Windows, December 2003
- Cubic also has scaling issue
  - $K = \text{cubic\_root}(W\_max * (1 - \text{beta\_cubic}) / C)$
  - K is a time factor, increases as  $\text{cubic\_root}(W\_max)$
- MIMD increase scales independently of CWND, data rate
  - But is it stable? Is it fair?

# Stability and fairness issue with multiplicative increase

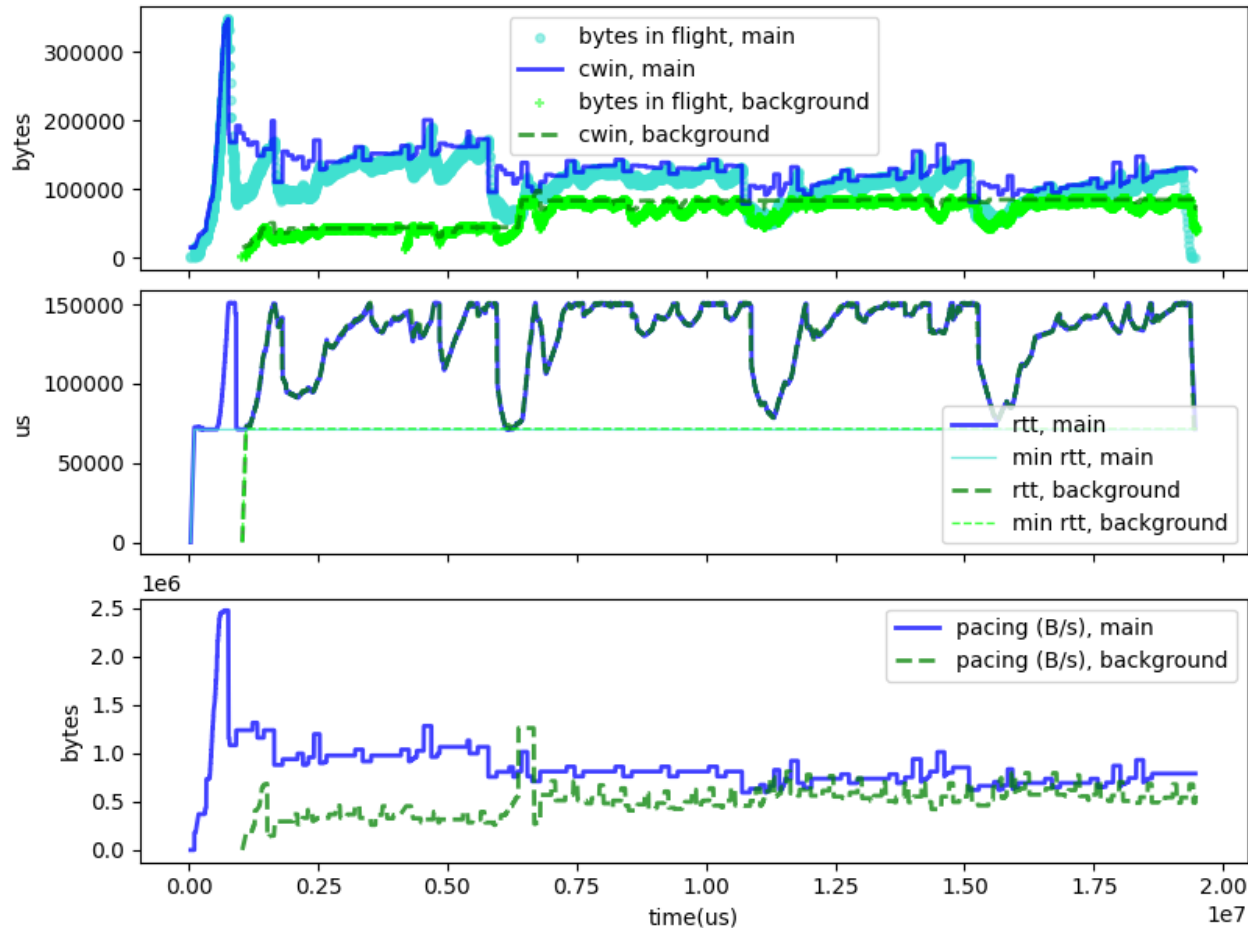
- Competing flows all see the same congestion signals (e.g., ECN rate)
  - On congestion, all decrease by same rate , e.g.  $\text{rate} \rightarrow \text{rate} * \alpha$
  - Absence of congestion, all increase by same rate, e.g.,  $\text{rate} \leftarrow \text{rate} * \beta$
- Result:
  - Share of each flow depends on initial conditions
  - Graphs tend to look like random walks

# C4 design choice: fair decrease, sensitivity

- Competing flows all see the same congestion signals (e.g., ECN rate)
- Absence of congestion:
  - $\text{Rate} += \text{rate} * \alpha$
- Congestion:
  - $\text{Rate} -= \text{rate} * \beta * \frac{(1 + \text{sensitivity})}{2}$

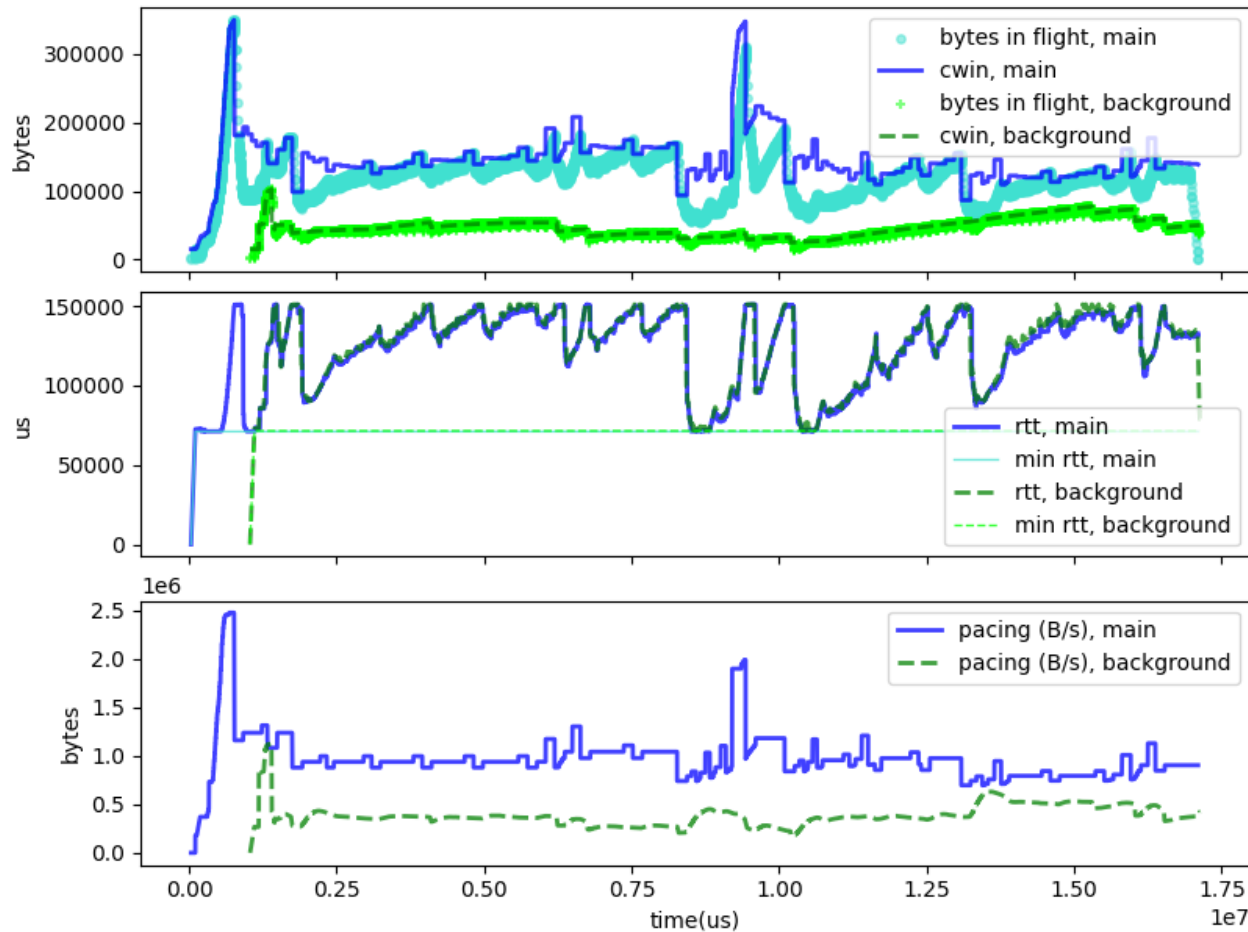


# Competition between C4 and BBR 2



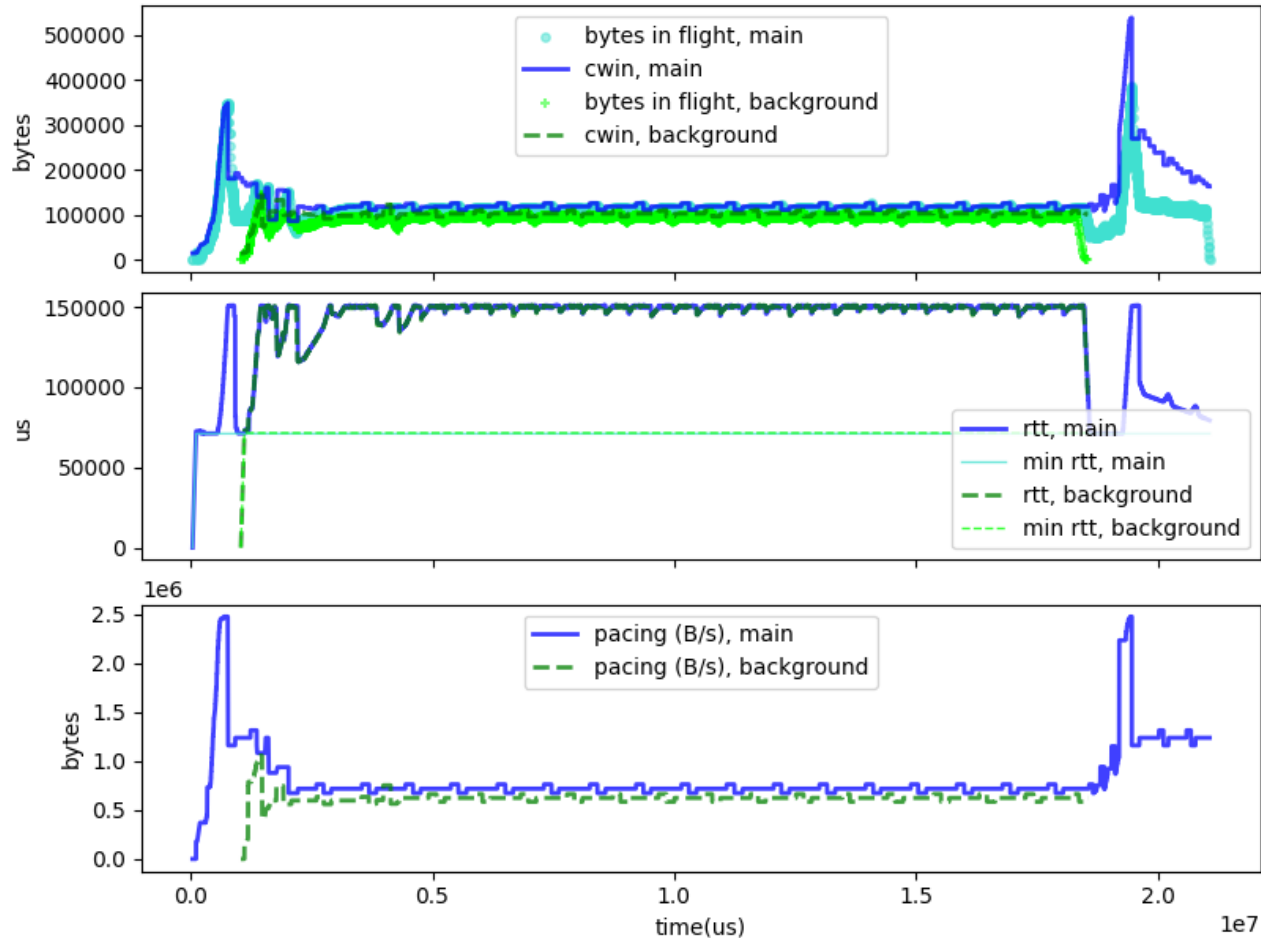
- The path has a 10Mbps data rate and 70ms RTT.
- Sharing between C4 (blue) and BBR (green),
- Pacing rates converge to similar values after initial stabilization
- RTT peaks to almost 2 times min RTT.

# Competition between C4 and Cubic



- The path has a 10Mbps data rate and 70ms RTT.
- Sharing between C4 (blue) and Cubic (green),
- Converges to about  $2/3^{\text{rd}}$  for C4,  $1/3^{\text{rd}}$  for Cubic
- RTT peaks to almost 2 times min RTT, more during some high growth periods.

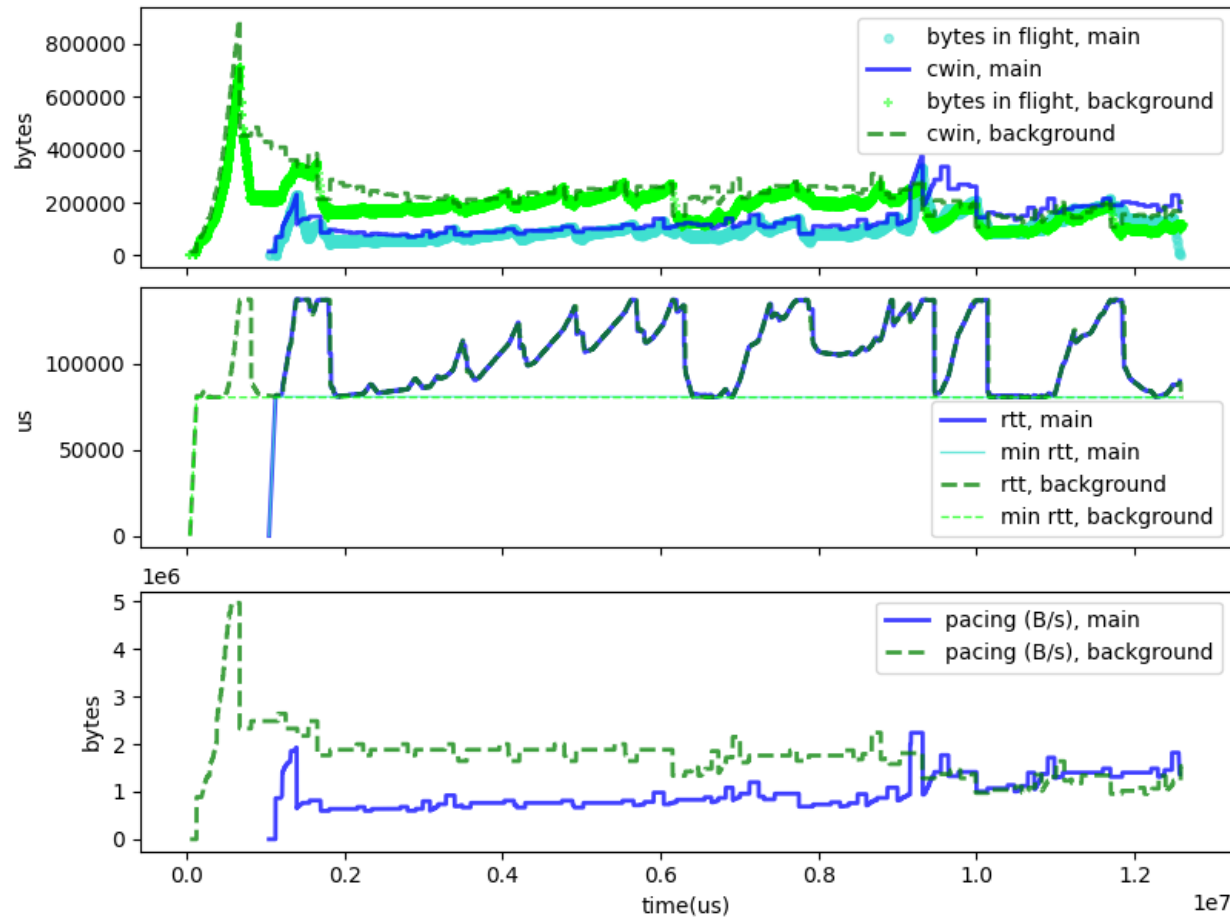
# Competition between C4 and C4



- The path has a 10Mbps data rate and 70ms RTT.
- Sharing between C4 (blue) and second C4 (green),
- Pacing rates converge to similar values after initial stabilization
- RTT peaks to almost 2 times min RTT.
- Returns to low values while competition ceases

*Trim the data  
rate a bit?*

# Competition between C4 and C4 with ECN



- The path has a 20 Mbps data rate and 80 ms RTT.
- Sharing between C4 (blue) and second C4 (green).
- L4S style AQM
- First comer advantage disappears after 8 to 9 sec
- RTT remains low-ish after initial phase

***Work to do on  
L4S simulation!***

# Work in progress, C4 next steps

- Management of congestion during probe
  - The other connections will back-off
  - The probing connection will keep its nominal rate
  - Maybe not fair, maybe fix that
- Try to fix the Initial phase, avoid the big delay spike.
- Manage what happens if rate evaluation is too high.
- Better simulation:
  - Reports results of the 100 tests in each trial
  - Further test the L4S simulator

# Documents

- [draft-huitema-ccwg-c4-spec](#)
  - Just the spec, no discussion.
  - Can you implement that? Give feedback?
- [draft-huitema-ccwg-c4-design](#)
  - Everything we discussed in this talk and more.
  - Something we miss? Suggestions?
- [draft-huitema-ccwg-c4-test](#)
  - 32 test scenarios, tested using the `picoquic\_ns` simulator
  - Validate proposed changes by testing each scenario 100 times.
- <https://github.com/private-octopus/c4>