Coupling Discrete Time Events to Continuous Time in RMCAT

(a.k.a. The Anatomy of a RMCAT RTT)

and Reasonable Bounds on the Time Rate
of Change in Available Capacity

Michael A. Ramalho, Ph.D.

November 5, 2014

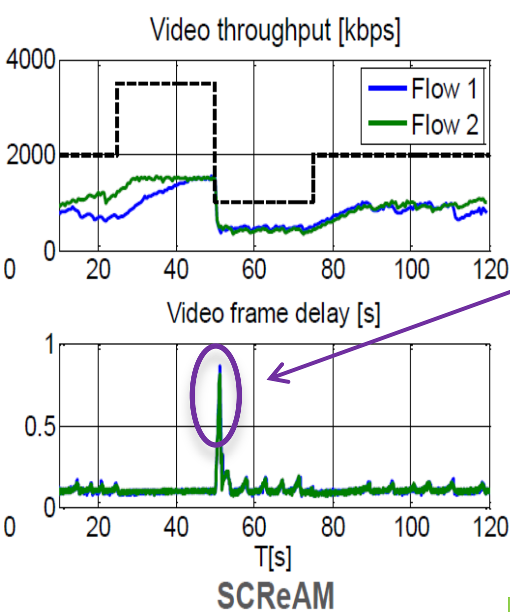
Talk Outline

1. Motivation (Prior Work On Test Plan Capacity Change Design)
2. RMCAT Discrete Time RTT Formalized
   - Fluid-flow (continuous-time) model and rigorous RMCAT RTT definition.
3. Infinitely Fast Capacity Change Downward
   - Unavoidable delay spike caused by infinitely fast capacity change
4. How Quickly ANY RMCAT Design Can Track Capacity Changes
   - Result is independent of algorithm type (“self-clocked” or “rate-based”).
5. Reasonable Assumptions on Time-Rate-of-Change of Capacity
   - Worse-case RTT defines “tracking responsiveness” (w/o predictive component).
   - Squelching mechanisms required (self-clocked schemes do this automatically).
   - TCP Dynamics as a function of their RTT.
   - A reasonable bound on RTCP feed back intervals.
6. Implications for Adaptation with Wireless (WiFi/LTE/etc).

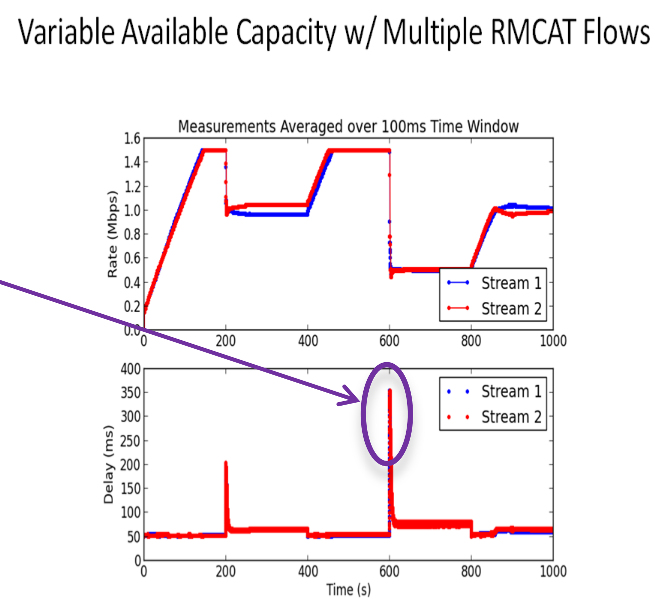
Motivation (Discussion at IETF 90)

Colin Perkins (Last Call on rmcat-cc-requirements):

- However, as I noted at IETF 90, I think the draft should also include a secondary requirement to **keep delay variation (jitter) down**, where possible, since larger delay variation needs larger receiver-side buffers to compensate, increasing overall latency.



Large Delay Spikes

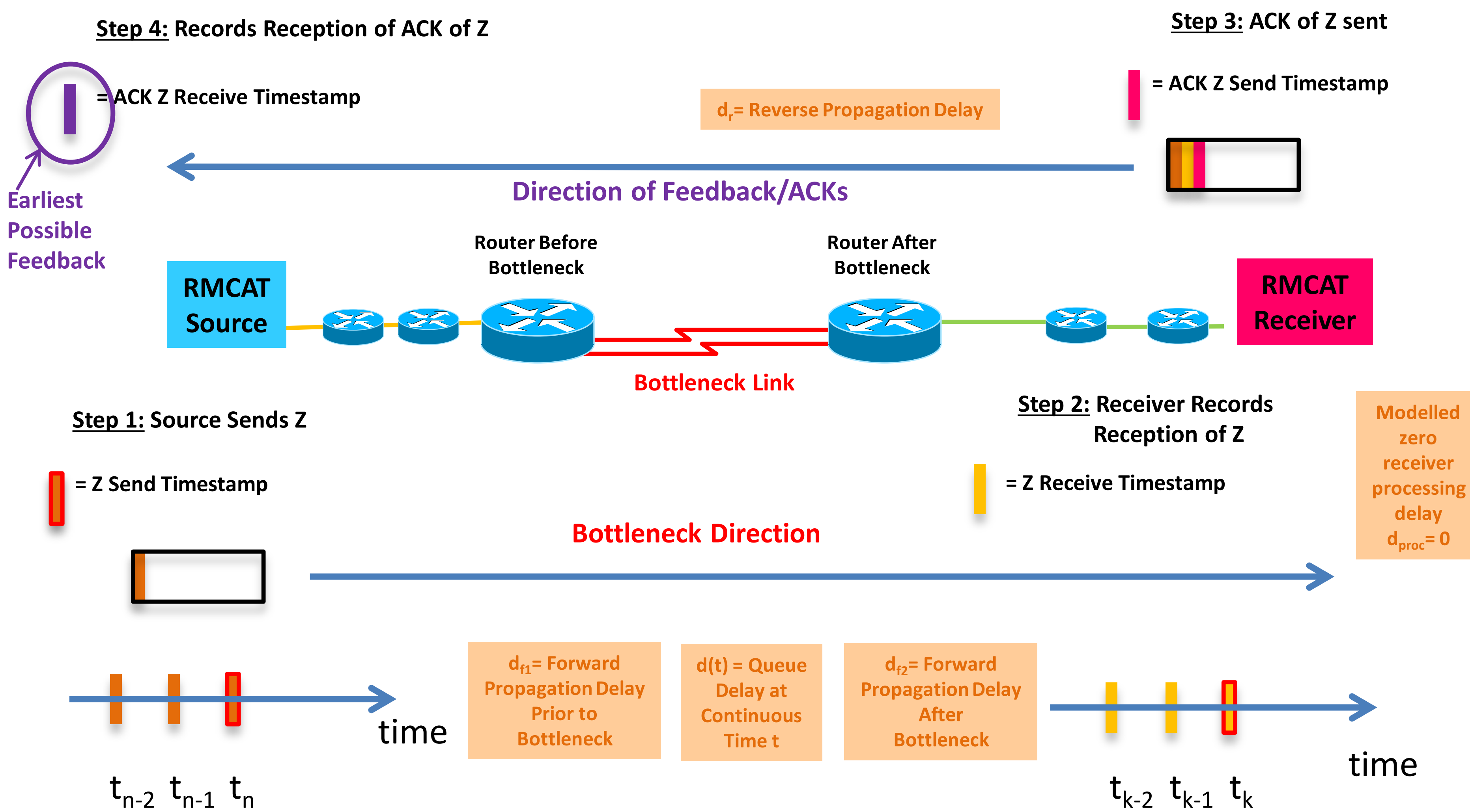


Colin (and others) believed these might be artifacts of the algorithms

NADAv2

A RMCAT Lab Discrete Time RTT (for Seq. No. = Z)

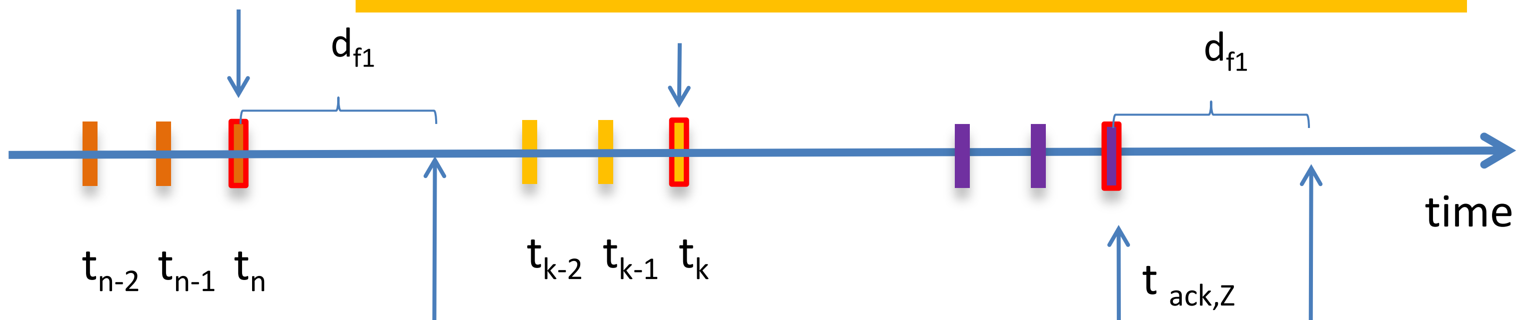
Note: Per-packet Feedback (no RTCP yet)



RMCAT LAB: Measurement Framework (Seq. No. = Z)

Sequence number Z sent.

Sequence number Z received (forward delay calculated here).



Sequence number Z-1 sent.

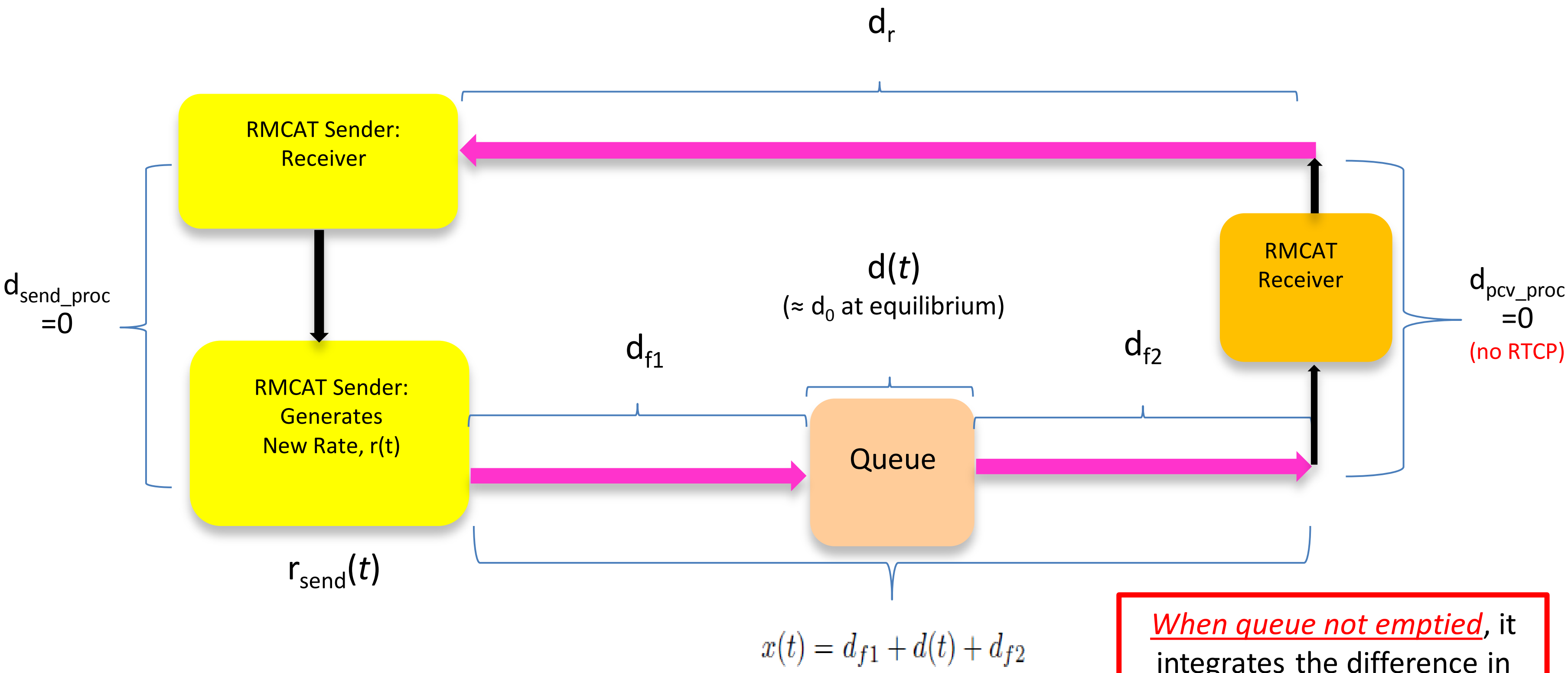
$t_{ack,Z}$ is the earliest time that the queue measurement is available at source (new rate change can occur here).

The time the queue was actually sampled is $t_{queue,Z} = (t_n + d_{f1})$.

The ***effect*** of the rate change on the queue occurs d_{f1} ***after*** the source made the rate change (a full RTT after ***the queue*** was sampled)!

Let's define ***continuous time t*** for fluid-flow modelling of the rate adaption component. That is, let $(t - t_{OFFSET})$ represent times offset from time t.

RMCAT: Continuous Time Modeling for Control Loop



When queue not emptied, it integrates the difference in rates (1/**s** in Laplace Domain).

For small τ;

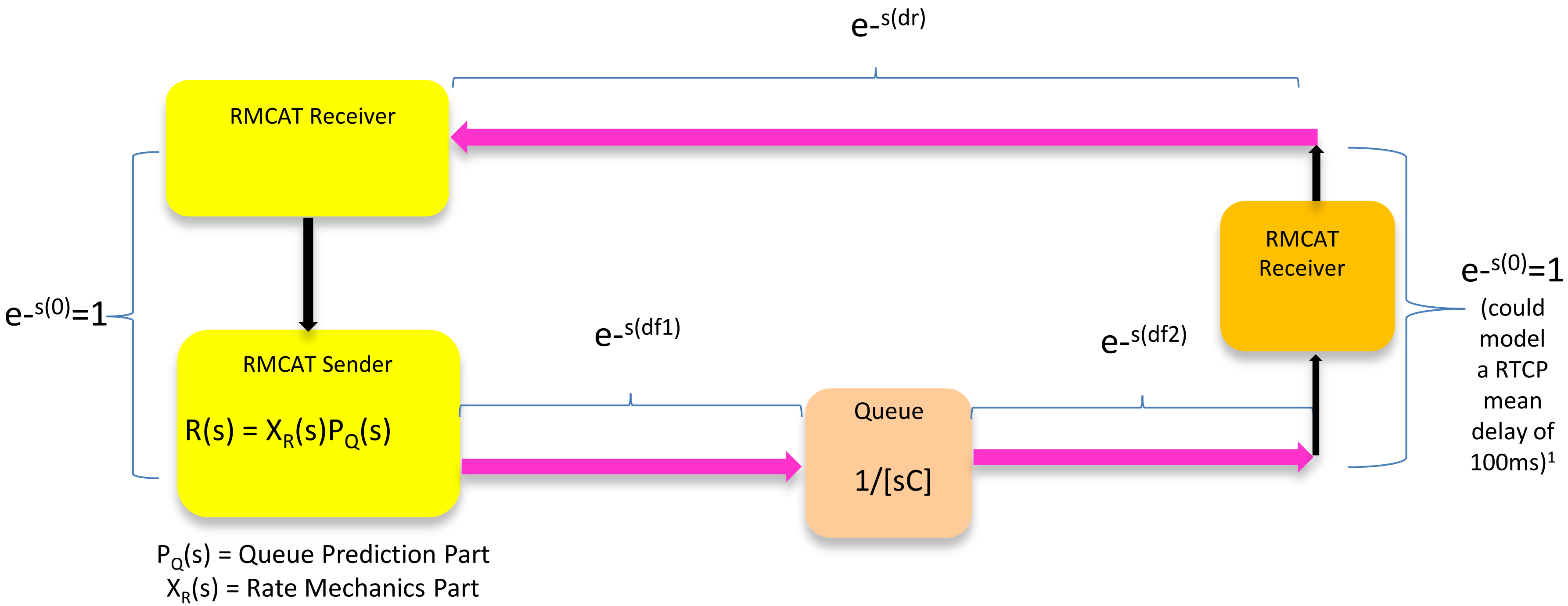
$$q(t) = \max [\, q(t - \tau) + (r_{at\_queue}(t) - C)\tau \,,\, 0 \,]$$

Queue increases/decreases as difference in rates

Also note:

- $r_{at\_queue}(t) = r_{send}(t - d_{f1})$
- $d(t) = q(t)/C$

Control Loop: Laplace Domain (linearize about equilibrium)

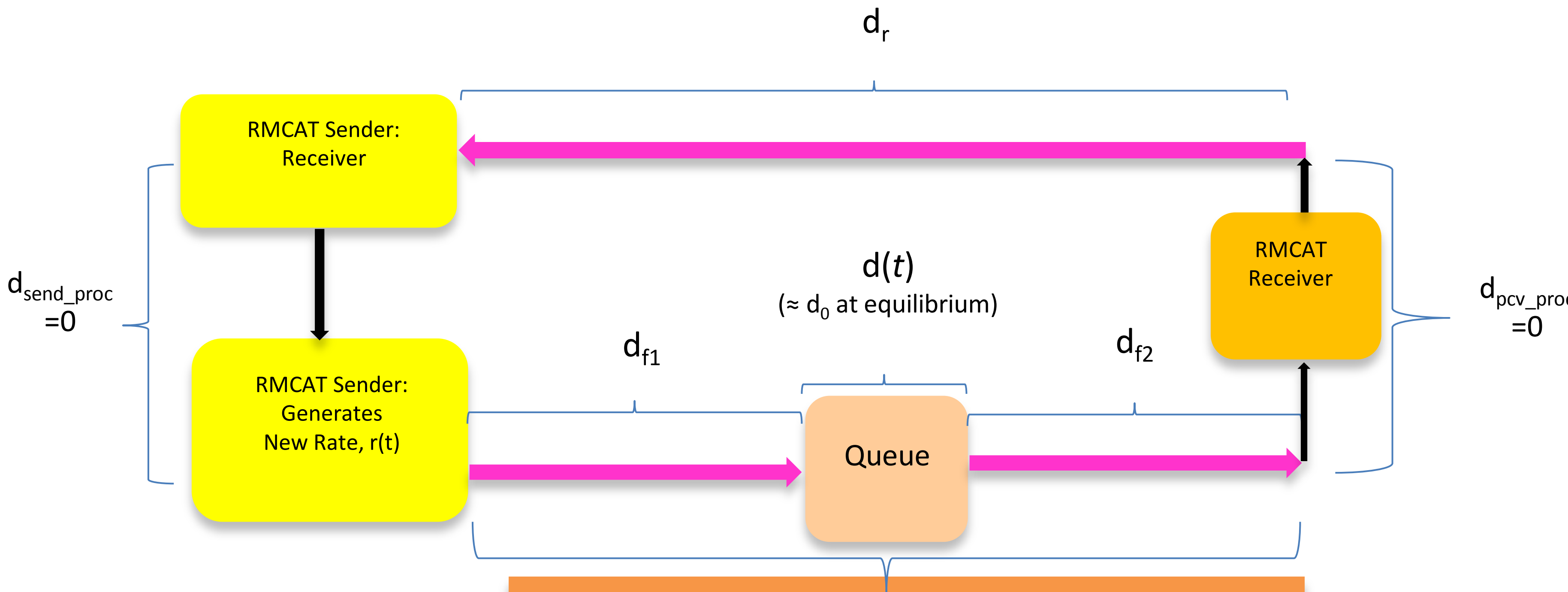


Key Control Loop Observations *At Equilibrium*:

- LTI System => Doesn't matter where prediction or rate control is (sender or receiver).
- Wherever it is, *it WILL take a minimum of a RTT[1] to make a difference at the queue*.

[1] – If the RTCP reporting interval is >> ($d_{f1}+d_{f2}+d_r$), it will dominate. If not, it will look like delay noise to individual delay samples.

So … What is the ***Discrete-Time*** RMCAT RTT Definition?



Forward Delay $= d_{f1} + \{ d(t) |_{t = t_{SAMP}} \} + d_{f2}$

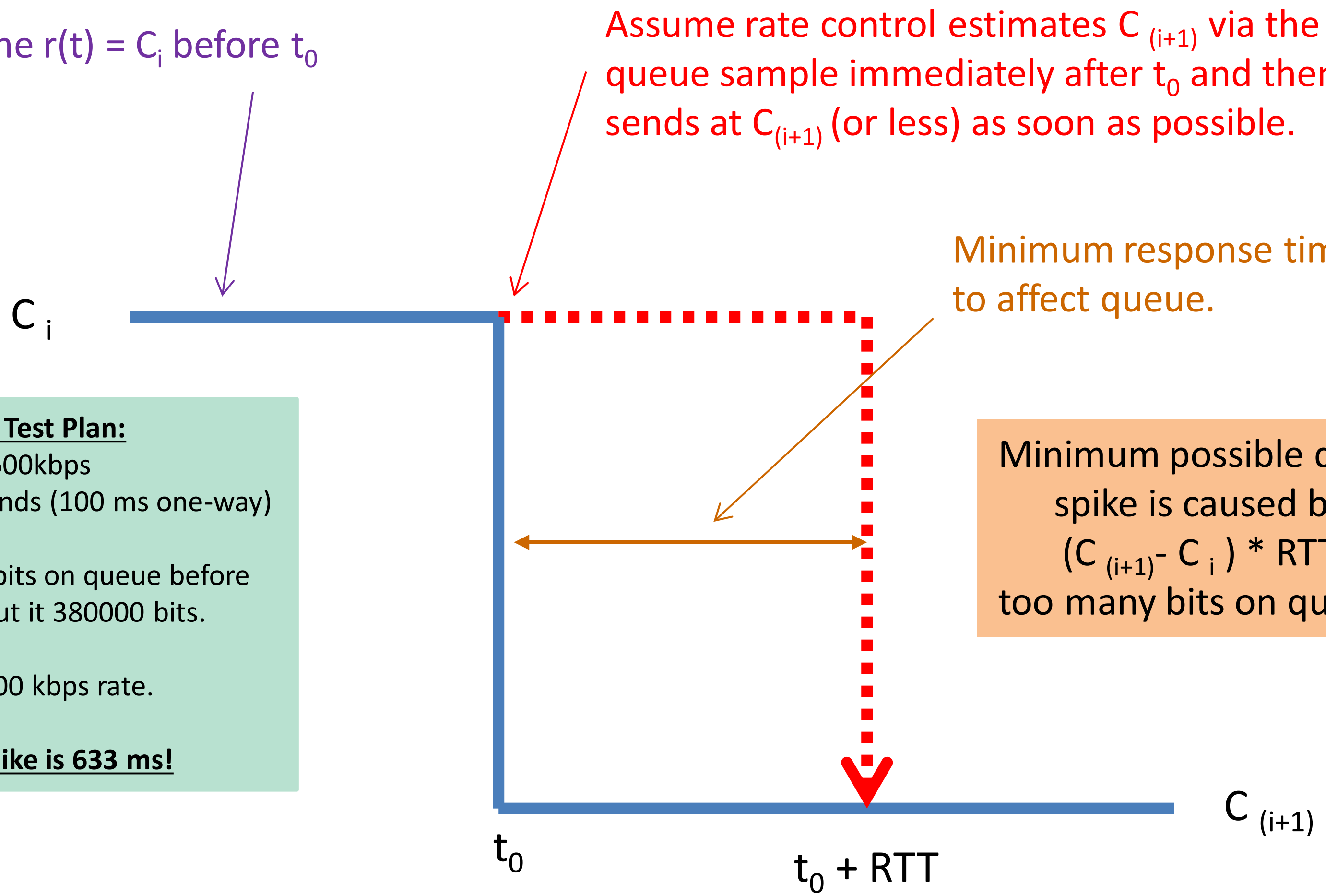
$$\text{RMCAT RTT } (t_i) \; |_{t_i = t_{ACK \text{ for seq Z received}}} = d_{f1} + \{ d(t) |_{t = t_{SAMP \text{ (seq no z)}}} \} + d_{f2} + d_r$$

$$\text{RMCAT RTT } (t_i) \; |_{t_i = t_{ACK \text{ for seq z received}}} = d_{f1} + d_{f2} + d_r + d(t - [d_x + d_{f2} + d_r]) \,,$$

$$\text{where } d_x = d(t) |_{t = t_{SAMP \text{ (seq no z)}}}$$

Queue Delay Variation During Downward Capacity Change

Fastest Possible Rate Adaptation Example (imprudently quick)



<u>Example in IETF RMCAT Test Plan:</u>
Capacity: 2500 kbps -> 600kbps
Assume RTT: 0.200 seconds (100 ms one-way)

Minimum ADDITIONAL bits on queue before we can do anything about it 380000 bits.

Queue must empty at 600 kbps rate.

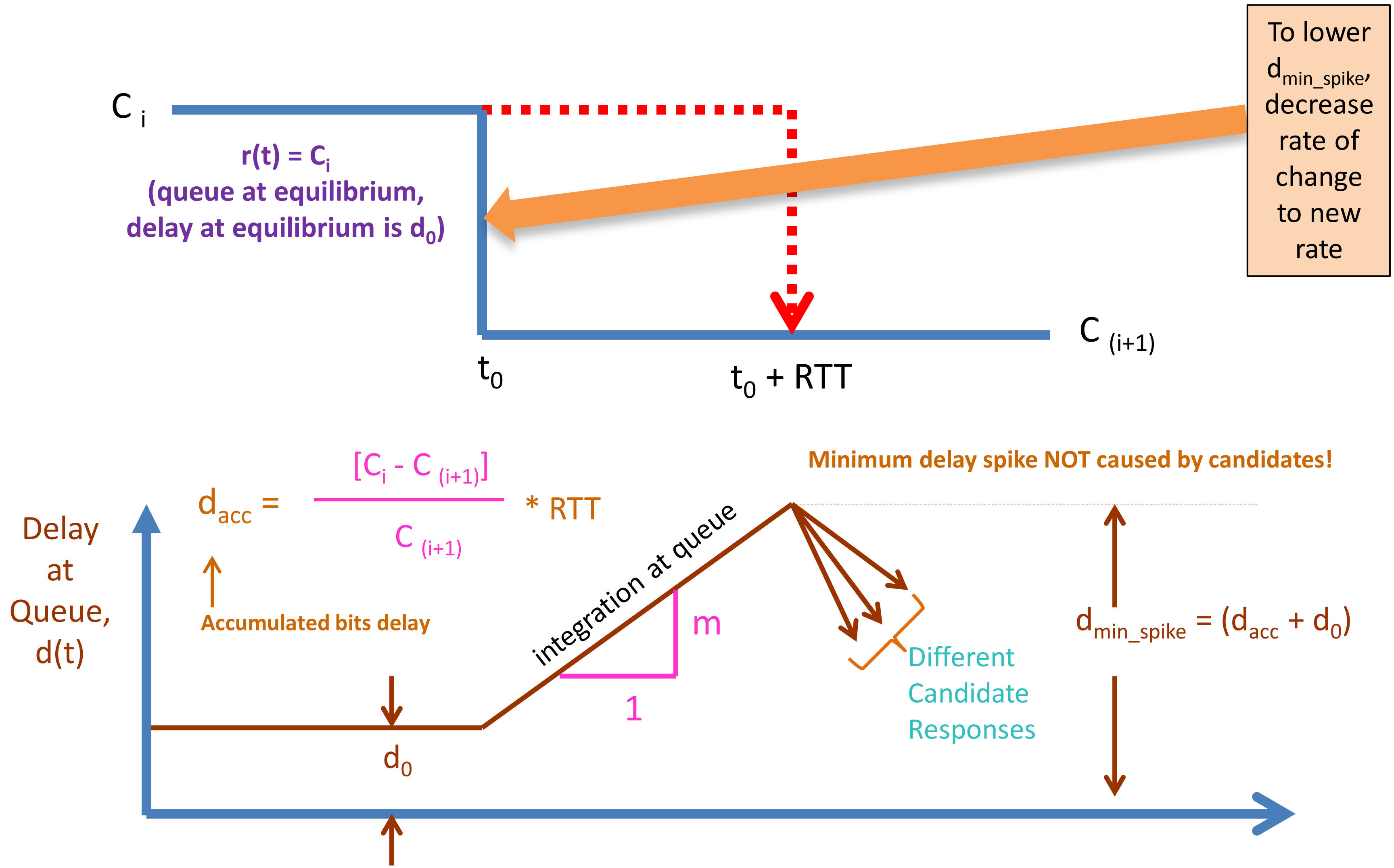
<u>Thus minimum delay spike is 633 ms!</u>

Minimum possible delay spike is caused by $(C_{(i+1)} - C_i) * RTT$ too many bits on queue.

Note: If we assumed one-way delay of 50 ms, 316 ms is minimum.

Queue Delay Variation During Downward Capacity Change

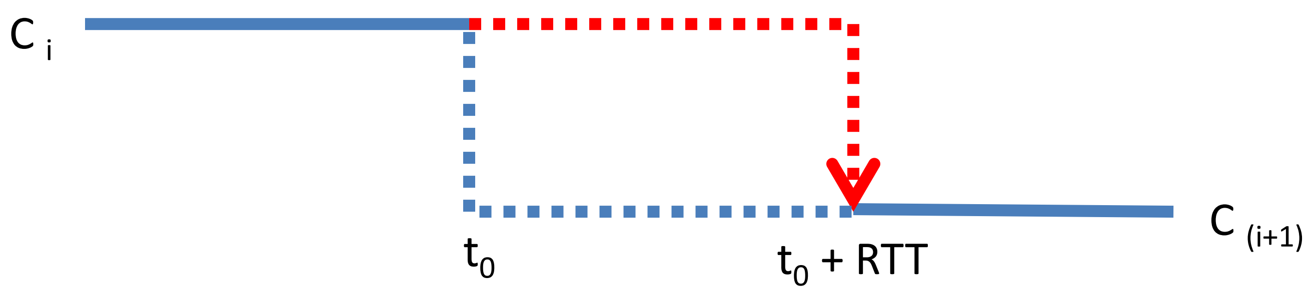
Fastest Possible Rate Adaptation Example (imprudently quick)



Talk Outline

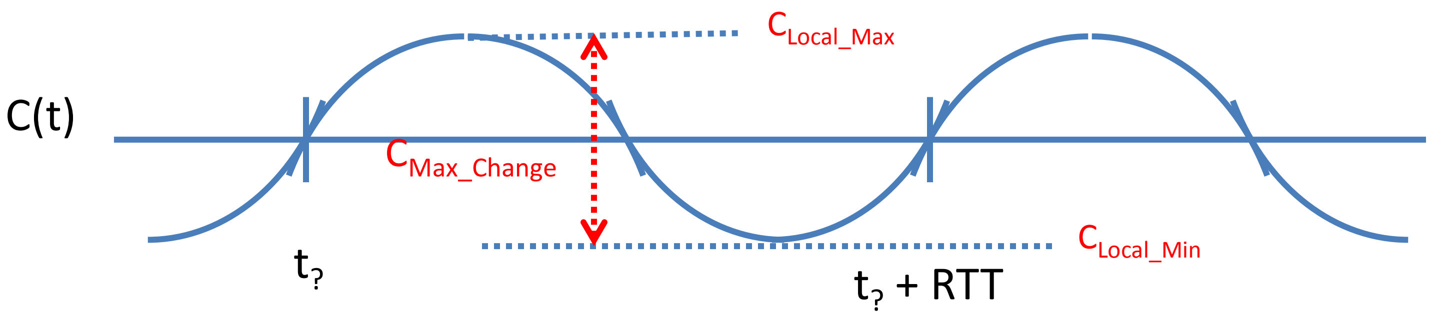
1. Motivation (Prior Work On Test Plan Capacity Change Design)
2. RMCAT Discrete Time RTT Formalized
   - Fluid-flow (continuous-time) model and rigorous RMCAT RTT definition.
3. Infinitely Fast Capacity Change Downward
   - Unavoidable delay spike caused by infinitely fast capacity change
4. How Quickly ANY RMCAT Design Can Track Capacity Changes
   - Result is independent of algorithm type ("self-clocked" or "rate-based").
5. Reasonable Assumptions on Time-Rate-of-Change of Capacity
   - Worse-case RTT defines "tracking responsiveness" (w/o predictive component).
   - Squelching mechanisms required (self-clocked schemes do this automatically).
   - TCP Dynamics as a function of their RTT.
   - A reasonable bound on RTCP feed back intervals.
6. Implications for Adaptation with Wireless (WiFi/LTE/etc).

How Quickly Can RMCAT Track Available Capacity Changes? Answer: It Depends on the RTT, Duh!



- The quickest time *__a RMCAT flow__* can possibly “measure” (and “react to”) changes in capacity is bounded by *__ITS__* round-trip time.
- Thus the quickest time it can influence it’s contribution to the queue after a change in capacity is thus bounded by *__ITS__* round-trip time.
- *__Corollary:__* RMCAT flows with different RTTs can react to changes on different time scales (which correspond to their individual RTTs). Just like TCP!
- A reasonable ASSUMPTION for the *__fastest__* time-rate-of-change in available capacity is one which could be tracked (i.e., measured and reacted to) by a RMCAT flow with *__an assumed worst-case RTT__* and *__RTCP interval__*.

A reasonable assumption bound for RMCAT time rate of change in available capacity



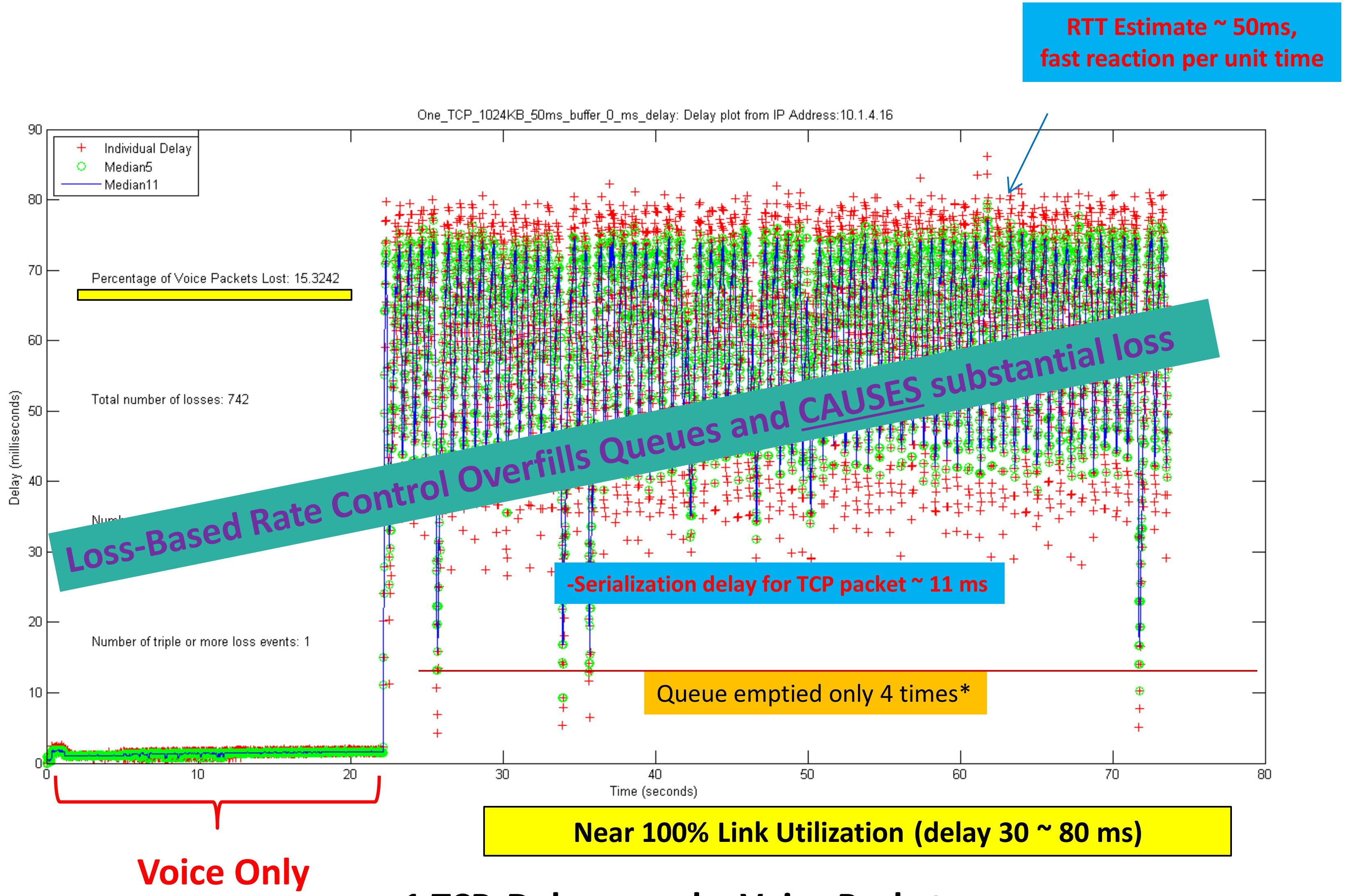
- Assume worst-case RMCAT RTT of ~250 ms (¼ sec).
- Highest capacity change "frequency" that is actionable is $f_{MAX} \approx 4$ Hz.
  - Capacity changes faster than this CANNOT be "seen/sensed/measured" and then "actionable" by a RMCAT flow with the "worst-case RTT".[1]
- Ditto for ACK-based, "self-clocked", protocols like TCP, SCReAM too!
  - TCP can't know to stop within this time either; as they only stop after their ACKs stop.
  - TCP thus "pounds the queue" causing gross overflow events on long RTT connections too!
- Corollary: "Fast Response" is a matter of the ***worst-case capacity change assumption*** - not a fundamental property of "self-clocked" protocols.[2]

1 – If other information was known (e.g., form), predictive components could react quicker.

2 - Rebuttal to: http://conferences.sigcomm.org/sigcomm/2014/doc/slides/150.pdf

TCP Dynamics as a Function of the RTT

1 TCP: Voice Delay, 50 ms BE Queue

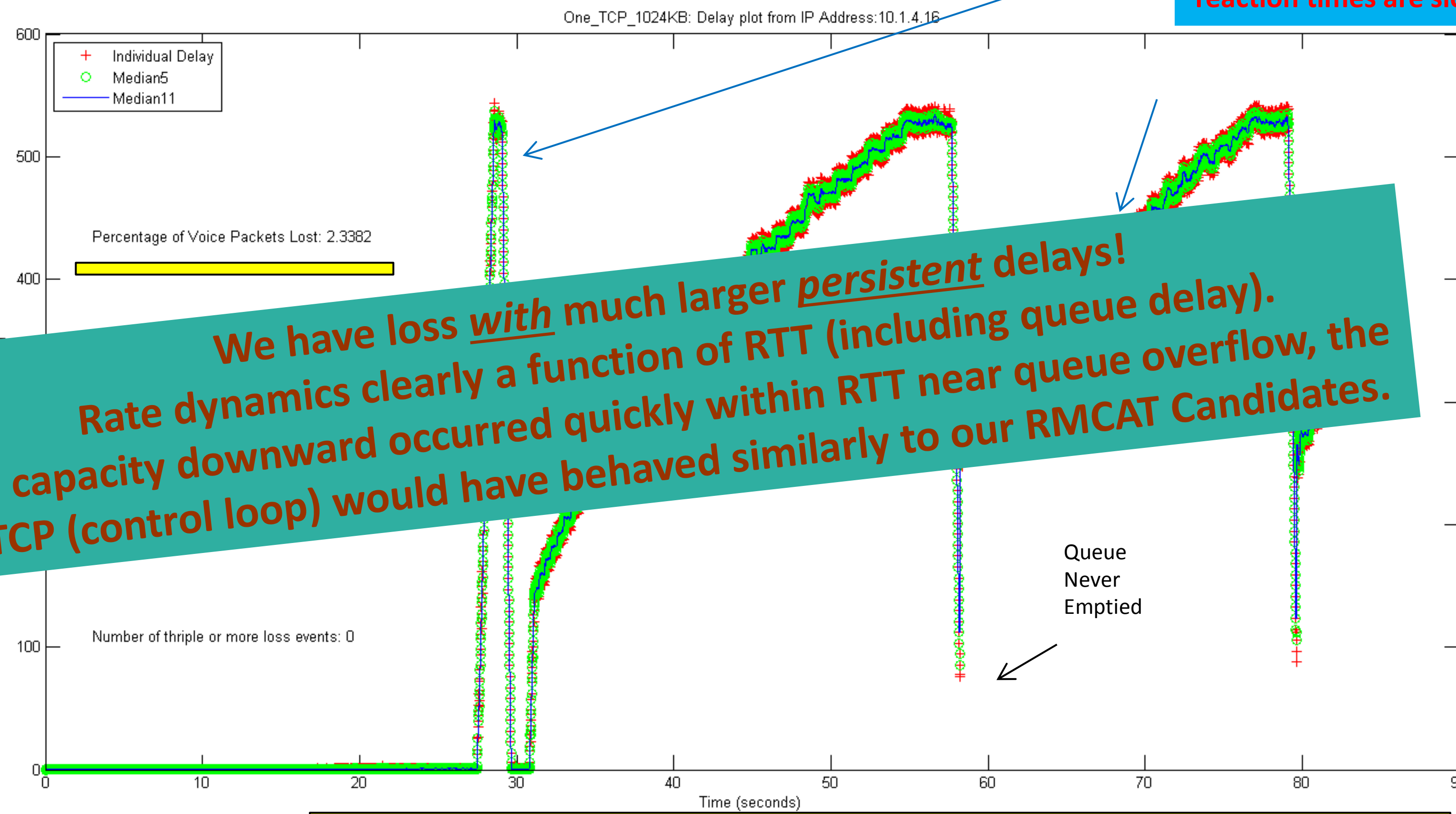


1 TCP, Delay seen by Voice Packets

TCP Dynamics as a Function of the RTT

1 TCP: Voice Delay, 500 ms BE Queue

RTT Estimate now includes queuing delay … subsequent rate increase reaction times are slowed.

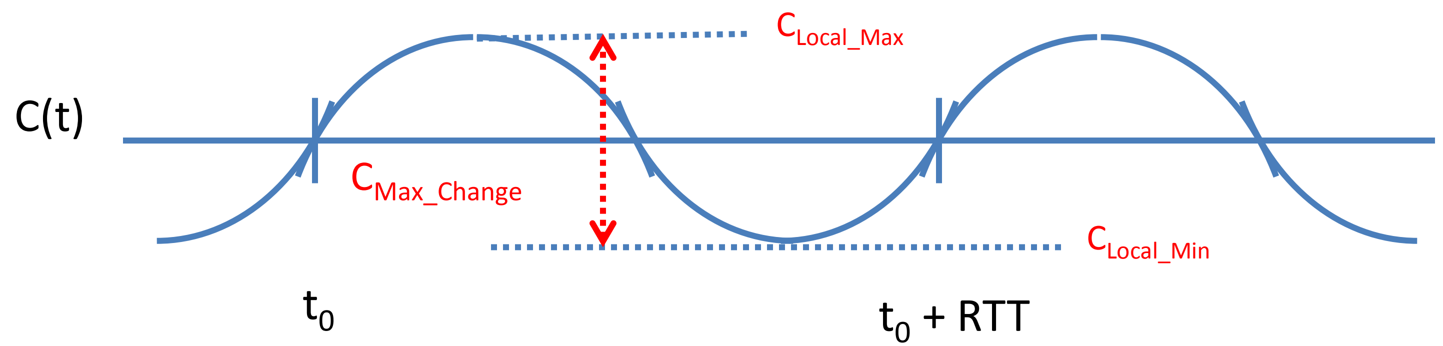


We have loss *with* much larger *persistent* delays!
Rate dynamics clearly a function of RTT (including queue delay).
If capacity downward occurred quickly within RTT near queue overflow, the TCP (control loop) would have behaved similarly to our RMCAT Candidates.

100% Link Utilization (delay 100 ~ 500 ms)

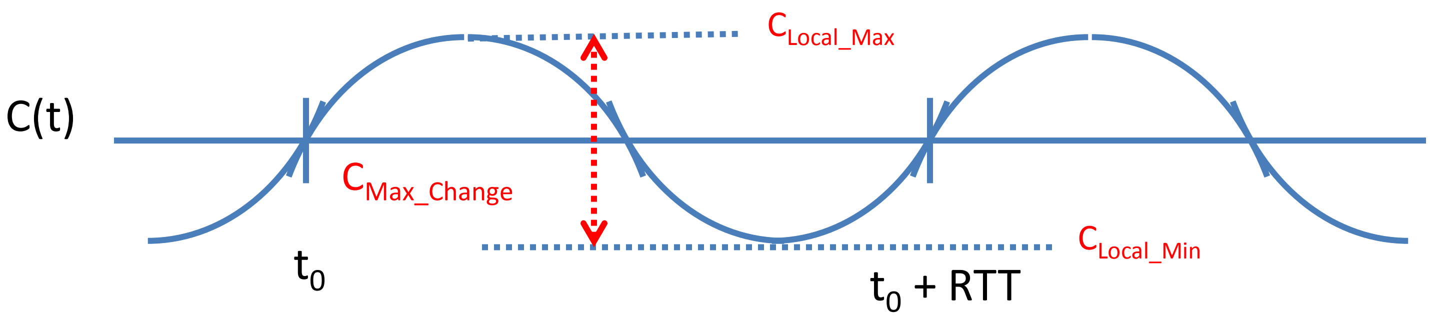
1 TCP, Delay seen by Voice Packets

ACK-Gated RMCAT Proposals (a la Ericsson)



- On long-RTT connections, ACK-gated protocols will also hit the queue too hard and build up excess delay. They will "stop" quickly – but that is a matter of degree (by comparison to how rate-based designs) – not because of some more beneficial property of ACK-gated protocols.*
- Once a worst-case RTT assumption has been made, it imposes a theoretical constraint on how quickly *ANY RMCAT FLOW* can adapt to it.
  - Thus imposing a "hidden assumption" on the time-rate-of-change of capacity ANY RMCAT DESIGN on *the worst-case RTT can adapt to*.
- This, in turn, imposes a minimum RTCP spacing constraint:
  - For 250 ms RTT, < π/2 spacing (@4 Hz) implies $T_{RTCP} \leq \sim 62.5$ ms.

Implications for WiFi/LTE/Wireless Adaptation



Repeated/paraphrased from last slide:

On long-RTT connections, ACK-gated protocols will also hit the queue too hard and build up excess delay. They will "stop" quickly – but that is a matter of degree (by comparison to rate-based approaches) – not because of some more beneficial property of "self-clocked" protocols over rate-controlled protocols.

Summary:

- We will need to develop better "squelching conditions" in future enhancements to our present RMCAT designs (i.e., when feedback stops and/or becomes irregular).
- *Complete squelch* is only prudent when there is *no impairment in feedback path*.
- Wireless challenges now become *a known second-order problem*; unless we want to limit RTT (not possible) or go to per-packet feedback (non RTCP approaches).

Summary

1. **RMCAT Discrete Time RTT Formalized**
   - Fluid-flow (continuous-time) model and rigorous RMCAT RTT definition.
2. **How Quick Can ANY RMCAT Design Can Track Capacity Changes**
   - Result is independent of algorithm type ("self-clocked" or "rate-based").
3. **Reasonable Assumptions on Time-Rate-of-Change of Capacity**
   - Worse-case RTT defines "tracking responsiveness" (w/o predictive component).
   - Like TCP, dynamics of RMCAT solution will be a function of their RTT.
   - The feedback intervals and flow RTT will determine capacity tracking ability.
   - Bounding feedback intervals and worst-case RTT effectively bounds best-case tracking.
4. **Implications for Adaptation with Wireless (WiFi/LTE/etc).**
   - Squelching mechanisms required (self-clocked schemes do this automatically).