cwnd and ssthresh are overloaded

- cwnd carries both long term and short term state
  - Long term state sometimes gets saved in ssthresh
- ssthresh carries queue size estimate and (temp) cwnd
- Poorly defined interactions between:
  - Application stalls and congestion control
  - Application stalls and loss recovery
  - Reordering and congestion avoidance
  - Other unanticipated concurrent events
  - ...
Laminar: Two separate subsystems

- Pure congestion control
  - New state variable: CCwin
  - Target quantity of data to be sent during each RTT
  - Carries state between successive RTTs
  - Not concerned with timing details, bursts etc

- Transmission scheduling
  - Packet conservation self clock (mostly)
  - Primary state is implicit, computed on every ACK
  - Variables: pipe (3517), total_pipe and DeliveredData
  - Controls exactly when to transmit
  - Tries to follow CCwin
  - Little or no explicit long term state
  - Includes slowstart, burst suppression, (future) pacing
Variables

- **CCwin**: (Target) Congestion Control window
- **pipe**: From 3517, data which has been sent but not ACKed or SACKed
- **DeliveredData**: Quantity of newly delivered data reported by this ACK (see PRR)
- **total_pipe = pipe + DeliveredData + SndBank**: This is all circulating data
- **SndCnt**: permission to send computed from the current ACK

Note that the above 4 are recomputed on every ACK

- **SndBank**: accumulated SndCnt to permit TSO etc
Default (Reno) Congestion Control

On startup:
    \( CC\text{win} = MAX\_WIN \)

On ACK if not application limited:
    \( CC\text{win} += MSS*\text{MSS}/CC\text{win} \quad \text{// in Bytes} \)

On congestion:
    if \( CC\text{win} == MAX\_WIN \)
        \( CC\text{win} = \text{total\_pipe}/2 \quad \text{// Fraction depends on delayed ACK and ABC} \)
    \( CC\text{win} = CC\text{win}/2 \)

Except on first loss, \( CC\text{win} \) does not depend on pipe!
Default transmission scheduling

```
sndcnt = DeliveredData  // Default is constant window
if total_pipe > CCwin:
    // Proportional Rate Reduction
    sndcnt = (PRR calculation)
if total_pipe < CCwin:
    // Implicit slowstart
    sndcnt = DeliveredData+MIN(DeliveredData, ABClimit)

SndBank += sndcnt
while (SndBank && TSO_ok())
    SndBank -= transmitData()
```
Algorithm updates

• Draft describes default Laminar versions of:
  o Congestion Avoidance (Reno)
  o Restart after idle
  o Congestion Window Validation
  o Pacing (generic)
  o RTO and F-RTO
  o Undo (generic)
  o Control Block Interdependence
  o Non-SACK TCP

• However there are many opportunities for improvement
Technical summary

- Today cwnd does both CC and transmission scheduling
  - Which are often in conflict
  - Every algorithm has to avoid compromising other uses
- Many pairs of functions interact poorly:
  - Congestion control and loss recovery
  - Application stalls and loss recovery
  - Pacing and CC
  - CC and restart after idle
  - etc
- Laminar separates CC and transmission scheduling
  - They become independent
  - Can evolve separately
  - No "cross subsystem" interactions
TCPM Issues

- Laminar removes ssthresh and cwnd
  - Updates or obsoletes approximately 60 RFC's
  - Interim plan: organize draft parallel to existing docs
- Most algorithm changes are straightforward
  - TCPM style standards (re)design
  - A few details have no precedent or otherwise call for significant redesign: Move to ICCRG?
- At what level (time?) does TCPM want to get involved?
  - Best if original authors redesign their own algorithms
Backup Slides
Fluid model Congestion Control

On every ACK:  // Including during recovery
   \[ CC_{\text{win}} += \text{MAX}(\text{DeliveredData}, \text{ABClimit}) \times \frac{\text{MSS}}{CC_{\text{win}}} \]

On retransmission:
   oldCC = CC_{\text{win}}
   \text{if} (CC_{\text{win}} == \text{MAX}_{\text{WIN}}):
      \[ CC_{\text{win}} = \text{initialCCestimate}(\text{total\_pipe}) \]
   \[ CC_{\text{win}} = CC_{\text{win}}/2 \]
   undoDelta = oldCC - CC_{\text{win}}

Undo:
   \[ CC_{\text{win}} = \text{MIN}(CC_{\text{win}}+\text{undoDelta}, \text{MAX}_{\text{WIN}}) \]
   undoDelta = 0

Insensitive to reordering and spurious retransmissions!