A Congestion Control Mechanism for Wireless Links

Upperside Wi-Fi Voice 2004

Michael Welzl
http://www.welzl.at, michael.welzl@uibk.ac.at
Distributed and Parallel Systems Group
Institute of Computer Science
University of Innsbruck, Austria
Outline

- Problem identification
- The PTP signaling protocol
- The CADPC congestion control mechanism
- Simulation results
- Conclusion
Wi-Fi + noise: what happens?

- **802.11 link layer ARQ** - classify based on persistence:
  - low/high: retransmit for a while, then give up [RFC3366]
  - perfect: not reasonable for WiFi
    (neither unlimited buffer nor “BROKEN LINK” signal feasible)

- **Simple test:**
  - cheap off-the-shelf equipment, 2 positions at our institute
  - 1800 pings (15 minutes, 1 ping every 0.5 seconds), packet size = 100 byte

<table>
<thead>
<tr>
<th>Almost perfect environment</th>
<th>Very noisy environment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>rtt min/avg/max/mdev</strong></td>
<td><strong>rtt min/avg/max/mdev =</strong></td>
</tr>
<tr>
<td>1.941/3.139/6.152/0.424 ms</td>
<td>3.274/12.410/1022.437/46.585 ms</td>
</tr>
</tbody>
</table>
Wi-Fi + noise + TCP = poor performance!

- **TCP congestion control:**
  - introduced in the 80s, necessary to prevent congestion collapse

- **TCP interpretation of network feedback:**
  - single packet loss = sign of congestion
    - or
      - Explicit Congestion Notification (ECN) flag = 1 = sign of congestion
        - sender halves rate (Congestion Avoidance)
  - many packets lost in a row = sign of heavy congestion (timeout)
    - sender goes back to Slow Start
  - round-trip time estimate increases / decreases
    - longer RTT = more conservative (e.g., increase by \( \approx 1 \) segment / RTT)

**Wireless links:** misinterpretation of *packet loss* and *RTT estimate* ⇒ massive throughput degradation!
Overview of problems with TCP(-like) CC

• Special links (becoming common!):  
  - noisy (wireless) links  
  - “long fat pipes“ (large bandwidth*delay product)

• Stability issues  
  - Fluctuations lead to regular packet drops + reduced throughput  
    ⇒ problematic for streaming media  
  - Stability depends on delay, capacity, load, AQM

• Rate hard to control / trace / predict  
  - Load based charging difficult

• Main reason: binary congestion notification (E)CN  
  - when it occurs, it’s (almost) too late
The CADPC/PTP Solution

- Totally different CC model
  - *only* rely on rare explicit bandwidth (=traffic) signaling

- Assumptions:
  - extra forward signalling for CC = good idea (*≠ common belief*)
  - router support
  - mechanism must clearly outperform TCP to justify (even a little!) additional traffic
  - timeouts necessary for loss of signalling packets
  *rate should not depend on round trip time*

...Ignore packet loss and delay!

...yes it does :-(
The Performance Transparency Protocol (PTP)

- Basic idea: query routers for performance related information
  - designed to be as lightweight as possible

- Stateless + simple ⇒ scalable!
  - all calculations @ end nodes

- Only every 2nd router needed for full functionality

- PTP Available Bandwidth Determination:
  - packet queries for:
    - nominal bandwidth ("ifSpeed") + address + traffic counter ("if(In/Out)Octets") + timestamp
    - 2 consecutive packets: table of traffic + interval from all routers at receiver
    - receiver calculates available bandwidth at bottleneck, informs sender

- Designed for flexibility - two modes:
  - Forward Packet Stamping, Direct Reply (not for available bandwidth (byte counters))
Forward Packet Stamping: how it works
Forward Packet Stamping: how it works
Forward Packet Stamping: how it works
CADPC: a new CC mechanism

- "Congestion Avoidance with Distributed Proportional Control"
  fully distributed convergence to max-min fairness
  - each source increases/decreases the rate depending on its capacity share

- Only depends on old rate, smoothness factor and traffic
  - does not depend on packet loss or RTT
    - Feedback packets can be delayed ⇒ scalable
    - reasonable choice: 4 x RTT

- Rate based, simple control

- Smooth convergence to a steady rate
CADPC synchronous case fluid analysis

- Final formula per user:

\[ x(t+1) = x(t)a(1-x(t)-\text{traffic})+x(t) \]

- Converges to: \( \frac{n}{n+1} \)

- Continuous-time version of synchronous case (\( \text{traffic}=nx \)):

logistic growth \[ x'(t) = x(t)a(1-x(t)/c) \quad [c = 1/(1+n)] \]

asymptotically stable equilibrium point: \( c \)
Some simulation results

Many more can be found in:

<table>
<thead>
<tr>
<th>Topology</th>
<th>Dumbbell</th>
<th>4 RTTs</th>
<th>0.5</th>
<th>1000 bytes</th>
<th>10 Mbit/s</th>
<th>1000 Mbit/s</th>
<th>50 ms each</th>
<th>Drop-tail</th>
<th>Long-term: 160 seconds</th>
<th>0 seconds</th>
<th>Greedy, long-lived</th>
<th>TCP Reno</th>
<th>50 packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>CADPC update frequency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CADPC smoothness factor a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packet Sizes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bottleneck Link Bandwidth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bandwidth of all other links</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Link delay</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Queueing discipline</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All flows start after</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bottleneck Queue Length</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CADPC vs. TCP
Smoothness

**10 x TFRC**

Bandwidth (byte/s) x 10^3

- Flow 0
- Flow 1
- Flow 2
- Flow 3
- Flow 4
- Flow 5
- Flow 6
- Flow 7
- Flow 8
- Flow 9

Time (s)

**10 x CADPC**

Bandwidth (byte/s) x 10^3

- Flow 0
- Flow 1
- Flow 2
- Flow 3
- Flow 4
- Flow 5
- Flow 6
- Flow 7
- Flow 8
- Flow 9

Time (s)
Startup enhancement
Heterogeneous RTTs + Robustness
CADPC vs. TCP-friendly CC. mechanisms

- Throughput
- Avg. Queue Length
- Loss
- Fairness

Graphs showing comparisons between CADPC and TCP-friendly CC. mechanisms with varying numbers of flows.
**CADPC vs. 3 TCP(+ECN) flavors**

![Graphs showing throughput, loss, average queue length, and fairness index for CADPC vs. 3 TCP(+ECN) flavors.](image-url)
CADPC Advantages

- high utilisation
- close to zero loss
- small bottleneck queue length
- very smooth rate
- fully distributed precise max-min-fairness
- rapid response to bandwidth changes (e.g. from routing)
- provable asymptotic stability (synchronous RTTs, fluid model)

some say it’s impossible :(
CADPC Advantages /2

- Useful for asymmetric links
- Useful for noisy (wireless) links + "long fat pipes"
- Useful for QoS and load-based charging

Disadvantages

- Requires router support
- Requires traffic isolation because...
  - not TCP-friendly
  - slowly responsive: bad results with web traffic
**CADPC/PTP + Wi-Fi Voice**

- **Wi-Fi**
  - Packet loss from link noise
    - misinterpreted as congestion by TCP (halve rate)
    - ignored by CADPC/PTP (only relies on byte counters)
  - Increased RTT estimate from link layer ARQ
    - more conservative behavior in TCP (less throughput)
    - ignored by CADPC/PTP (convergence RTT independent)

- **VoIP**
  - long-lived flows: a requirement for CADPC
  - QoS requirements: inherently supported by CADPC
  - small packets: make CADPC work even better (more precisely)
Link noise: simulation example

1 TCP vs. 1 CADPC flow

Remember: CADPC converges to $n/(n+1)$!

(easy to change for known max. no of flows)
The End …

- Publications
- CADPC+PTP ns code
- PTP Linux code (router kernel patch + end system implementation)
- Future updates

http://www.welzl.at/ptp

Note: I am eager for projects; ⇒ contact me!