Internet congestion control: TCP

- **1988:** "Congestion Avoidance and Control" (Jacobson/Karels)  
  Combined congestion/flow control for TCP

- **Goal:** stability - in equilibrium, no packet is sent into the network until an old packet leaves  
  - ack clocking, "conservation of packets" principle  
  - possible through window based stop&go - behaviour

- Superposition of stable systems = stable ->  
  network based on TCP with congestion control = stable

- Today, TCP dominates the Internet (WWW, ..)  
  recent backbone measurement: 98% TCP traffic
AIMD Background

- User 1 Allocation x1
- Fairness Line
- Efficiency Line
- User 2 Allocation x2
- AIMD Background
- MIMD
- AIAD
- AIMD
- Desirable
- Underload
- Overload
- Efficiency Line
- Fairness Line
Some reasons for TCP stability

• Exponential backoff:
  “For a transport endpoint embedded in a network of unknown topology and with an unknown, unknowable and constantly changing population of competing conversations, only one scheme has any hope of working - exponential backoff - but a proof of this is beyond the scope of this paper.”

• Conservation of packets:
  “The physics of flow predicts that systems with this property should be robust in the face of congestion."

• Additive Increase, Multiplicative Decrease:
  Not explicitely cited as a stability reason in the paper!
    - ...but in 1000's of other papers!
"Proofs" of TCP stability

- **AIMD:** Chiu/Jain: diagram + algebraic proof homogeneous RTT case

- steady-state TCP model: window size $\sim \frac{1.22}{\sqrt{p}}$
  
  ($p =$ packet loss)

- **Johari/Tan, Massoulié, ..:**
  - local stability, neglect details of TCP behaviour (fluid flow model, ..)
  - assumption:
    "queueing delays will eventually become small relative to propagation delays"

- **Steven Low:**
  - Duality model (based on utility function / F. Kelly, ..):
    Stability depends on delay, capacity, load and AQM !
Extended Use of Vector Diagrams

• Problem:
  - Stability analysis very complex
  - TCP-like mechanism design difficult

• Solution:
  - Extended use of vector diagrams!

• Analyze actual results (from simulation or real life measurements)

• Instead of just explaining a concept, design in the diagram space
  - Necessary simplifications may even be less dramatic!
How Stable is AIMD / async. RTT?

- Simple simulation (no queues, ..)
- RTT: 7 vs. 2
- AI=0.1, MD=0.5
- Simul. time=175
How is AIMD distorted in TCP?

- ns-2 simulator
- TCP Tahoe
- equal RTT
- 1 bottleneck link
Problems with TCP

- TCP over wireless: checksum error -> packet drop misinterpretation
- TCP over “long fat pipes”: large bandwidth\*delay product
  - long time to reach equilibrium, MD = dramatic!
- TCP reaches equilibrium, but not a stable point
  - fluctuations lead to regular packet drops & reduced throughput
  - fluctuations not feasible for streaming multimedia apps
- TCP Vegas: interpret delay as congestion ... but:

  similar delay, different available bandwidth!
Enhancement idea: ATM ABR

- TCP -> TCP/RED -> TCP/RED/ECN -> TCP/RED/Multilevel ECN -> ...

- Logical consequence: "ECN" with fine granularity (explicitly ask for congestion information)

- Routers know more about congestion. TCP-AI is like guessing and reacting when it is already too late.

- Idea related to ATM Available Bit Rate (ABR) service:
  - Resource management (RM) cells query the network for ECN flag & Explicit Rate information
  - Framework for sophisticated ER calculations -> 1000s of papers

True for all mechanisms which use binary feedback!

Often: Fairness through flow counting -> not scalable!
PTP: The Performance Transparency Protocol (draft-welzl-ptp-05.txt)

- "Generic" ECN / BECN - to carry traffic information (e.g. queue length, ..)

- Stateless & simple -> scalable!

- Only every 2nd router needed for full functionality
  - If less routers support it, results are an upper limit

- Available Bandwidth Determination:
  - nominal bandwidth ("ifSpeed") + 2* (address + traffic counter ("if(In/Out)Octets") + timestamp) = available bandwidth

- two modes: "forward packet stamping" / "direct reply" (not for available bandwidth (byte counters))
PTP: How does it work?

- Forward packet stamping:

```
Sender  
S-a  
9>8!

"Init 9"

Router  
1  
A-b  
7=7

2  
B-?-c

3  
C-r  
6>5!

Receiver

TTL: 8
TTL-C: 9
Request: Bwinfo

TTL: 7
TTL-C: 7
a Bwinfo
A Bwinfo

TTL: 6
TTL-C: 6
a Bwinfo
A Bwinfo
B Bwinfo

TTL: 4
TTL-C: 4
a Bwinfo
A Bwinfo
B Bwinfo
c Bwinfo
C Bwinfo

9-4-5=0, complete
Bwinfo table

Upper case: outgoing interface

Lower case: incoming interface

- The receiver can tell if the information is complete (DS Count, TTL)```
PTP: How does it work? /2

• Direct reply:
  - Routers compare values; if requirements cannot be met...
    • the values are updated, the packet is redirected to the sender
  - Similar to RSVP reservation setup
  - Does not work with available bandwidth (traffic counter)
  - Advantageous for satellite links!

• Forward packet stamping
  satellite usage scenario:

1st ground station acts as a receiver - only relies on PTP
Design algorithm

• find useful (closely related) ATM ABR mechanism

• start with simplifications, then expand the model

• A new mechanism must work for 2 users, equal RTT
  – simple analysis similar to Chiu/Jain (diagram + math)

• it must also work for heterogeneous RTTs
  – simulate using a simple diagram based simulator
  – analyze using extended vector diagram analysis

• it must also work for more users and more realistic scenarios
  – simulate with ns
The ATM ABR best match: CAPC

- “Congestion Avoidance with Proportional Control” (A. Barnhart 1994)

- Uses load factor LF: Input Rate IR / Target Rate R0
  - RO e.g. 95% of nominal bandwidth, \( d = 1 - LF \) (available bandwidth)

- "As long as the incoming rate is greater than RO, the desired rate, ERS will diminish at a rate that is proportional to the amount by which RO is exceeded. Conversely, whenever the incoming rate is less than RO, ERS will increase."

- for each new cell entering the queue:
  LF\(\leq 1\): \( ERX = \min(ERU, 1 + d*Rup) \) ... else \( ERX = \max(ERF, 1 + d*Rdn) \)
  \( ERS = ERS*ERX \)
  - constants: Rup, Rdn define the speed of rate increase / decrease,
    ERU, ERF = upper / lower bound
  - different default values for LAN and WAN!

hint for RTT dependance!
Conversion for packet nets: CADPC

- "Congestion Avoidance with Distributed Proportional Control"
- Only ask for current load, do calculations at sender

- **Fairness:** sender 1 RTT = x * sender 2 RTT
  - sender 1 needs to calculate ERS x times -> $ERS = ERS \cdot (ERX^x)$

- **Result in Chiu-Jain-diagram-simulator:**
  - max-min fairness approximated for limited rtt variations
    (trade-off between Rup, Rdn and allowed RTT's)
  - not good enough, but gives a hint that weighing the rate changes with
    the user's properties can lead to max-min fairness!
**CADPC Design**

- **Idea:**
  - relate user's current rate to the state of the system! \(^{(\text{also in LDA+})}\)
  - Thought: in the Chiu-Jain-diagram, if the rate increase is indirectly proportional to the user's current rate, the rates will equalize.

- \( \text{erx} = 1 + \text{rup} \times (1.0 - \text{myRate/traffic}) \)
  - does not work! e.g. synchronous case \(\rightarrow\) \(\text{myRate/traffic} = \) constant!

- **Solution:**
  - \( \text{erx} = 1 + \text{rup} \times (1.0 - \text{myRate/d}) \)
  - relationship between user's rate and available rate keeps changing!

- **Enhancement:**
  - dependence on \(\text{rup}\) not desirable; rate changes should be proportional to the current load \(\rightarrow\) use \(\text{d}\) instead of \(\text{rup}\)!
CADPC vector diagram analysis
CADPC synchronous case analysis

- **Final formula per user:**
  \( d = \text{traffic} / r_0; \)
  \( \text{erx} = 1 + d \times (1.0 - \text{myRate}/d); \)
  \( \text{ers} = \text{ers} \times \text{erx}; \)

- **Combined:**
  \( x_i(t+1) = x_i(t) \times \left( 1 + d \times \left( 1 - \frac{x_i(t)}{\sum_{j=1}^{n} x_j(t)} \right) \right) \)

- **After some straightforward derivations:**
  \( x_i(t+1) = x_i(t) \left( r_0 + 1 - r_0 x_i(t) - \sum_{j=1}^{n} x_j(t) \right) \)

(Special form of logistic equation => stable!)
CADPC synchronous case analysis /2

- Equilibrium: assume $x(t+1) = x(t)$
- leads to: $x(t) = \frac{r_0}{n+r_0}$
- traffic (n users): $n \times x(t) = \frac{n \times r_0}{n+r_0}$

Convergence of equilibrium with increasing number of users:

$\text{r}_0 = 1$
ns simulation: 25 TCP / 25 CADPC

not simultaneously!

Bandwidth (byte / s)

Time (s)

single bottleneck (dumbbell)
Results

• Implementation: $r_0$ normalized to 1 $\rightarrow$ calc $\rightarrow$ de-normalize
• 1 PTP packet every 4 RTTs, no other acks!
  - rate indeed converges to $\frac{n}{n+1}$

• No packet loss

• Very smooth rate, rapid convergence

• Not in the picture:
  - rapid convergence to almost perfect fairness
  - bg traffic: rapid backoff and recovery
CADPC advantages

• Better stability than TCP
  - smooth rate advantageous for streaming media apps

• No problems with wireless links (no packet loss interpretation)

• Rare feedback - good in environments with long delay
  - rapid convergence & reaction - good in environments with a high bw*delay product

• Rate calculation independent of RTT => independent of position
  - scalable! if PTP = x% of generated traffic \( n \), PTP scales \( O(n) \)

• Only (rare) PTP packets necessary to calculate rate
  - Satellite environments:
    do receiver's calculations at sat. base station and give earlier feedback
  - easier to differentiate pricing
  - easier to implement metering => traffic shaping, policing, admission control, ..
Deployment plans

• Problem: PTP needs router support
  - CADPC needs complete path information (every 2nd router)

• Possibilities:
  - QoS in a DiffServ class (QoS “in the small”):
    “we offer QoS & provide router support, you use CADPC and get a good result”
  - If CADPC works with non-greedy senders: edge2edge PTP signaling (TCP over CADPC)
    PTP supported traffic engineering
  - CADPC <=> TCP translation at edge routers?
Future work

- **More ns simulations**
  - CADPC vs. AIMD in vector diagram simulator: **CADPC is much less aggressive**
  - compare with TCP-friendly binary mechanisms
  - compare with other ER mechanisms PCP, ALS

- **Extension to proportional fairness?**

- **CADPC implementation**
  - PTP already available for Linux
  - compare with TCP, TFRC, RAP, ...
  - evaluate QoS