Rethinking TCP-Friendly

Matt Mathis
Pittsburgh Supercomputing Center (PSC)
TSV AREA at IETF 74
25-Mar-2009

http://staff.psc.edu/mathis/unfriendly
http://staff.psc.edu/mathis/papers/TSVAREA74.pdf

The key points

- TCP-friendly paradigm is not enough (review)
  - Simple devices send uniform signals to all flows
  - All flows are mandated to have similar response
    - RFC 2581 AIMD, TFRC, etc
- The network should do capacity allocation
  - Aka “fairness”
- What happens if this idea is taken to the limit?
  - Protocols should not have to worry about fairness
  - Responsibly try to fill the network
  - The network allocates capacity
- What might be possible?
  - Consider the following non-standard CC.....

Relentless TCP

- Pure implementation of VJ packet conservation
  - Packets are sent in response to packets arriving
  - (Ideally) the only window reduction would be loss
  - Additive Increase only when:
    - Lossless RTT and
    - Flight size == cwnd

- Cool new property
  - TCP portion of the CC system has unity gain
    - e.g. If the queue is 20 packets too large, just drop 20 packets
    - If flow is using 1% too much capacity, drop 1% (for 1 RTT)
  - Claim: vastly easier queue management
    - Do not have to estimate TCP's response to a single loss

One minor problem

- It controls hard against “queue full”
- Must have a queue controller to limit occupancy

An example “baseline” queue controller

- Segregate flows
  - Assume some explicit scheduling policy
- In each (periodic) interval:
  - Monitor the minimum queue length
  - If it is above the set point,

    drop excess packets during the next interval
- The ideal periodic interval is 1 RTT
  - But it is not sensitive to huge miss-match
  - e.g. at 10RTT the average error is only 5 packets

Usage example: remote video upload

- Satellite link with TCP video upload + interactive
  - Assume 10 Mb/s, 600 ms RTT, 1500 B MTU
  - Pipe size is ~500 packets
  - Assume DSCP queuing or RR scheduling
- Relentless TCP + Baseline queue control (1s)
  - Expect one drop per 2 S (1 drop per 3 RTT)
    - About 1 per 1500 packets

Same example using standard TCP

- Optimal solution for 1 flow:
  - Need a loss when queue reaches 500 packets
  - Which is every 1000 RTT (due to delayed ACK)
  - Or once every ~700,000 packets (>10 minutes!)
  - 500 times lower loss rate than Relentless
  - More likely BER, etc prevents filling the link
  - THESE NUMBERS ARE ABSURD
- Work around by using multiple flows
  - Changes the peak queue size
  - An optimal queue controller must estimate:
    - Flow population and/or
    - Effective RTT

  (Note that Relentless/Baseline does not need this)

Claimed properties of Relentless CC

- Model: rate is proportional to 1/loss_rate
  - One loss every 3 RTTs
  - Vastly higher equilibrium loss rates than standard TCP
    - By roughly the window size in packets
      (e.g. 500 in the previous example)
- Not at all AIMD-friendly
  (Workaround: use Lower Effort (LE) service [RFC3662])
- Can not cause congestion collapse
  - Assuming the timeout behavior is unchanged
- Can replace MD in any AIMD style CC
  - Especially good with delay sensing algorithms
  - I chose to start with unmodified Reno to simplify the dialog, not because I think it is optimal

Limitations of the Implementation

- Hammers on SACK and the recovery code
  - Was easy to notice things that don't look quite right
- Hammers on the network
  - Interesting oddities in traces
- Flight size vs cwnd
  - Current philosophy is to limit bursts:
    - Pull cwnd down to flight size during recovery and other places
    - Running out of rwin or sender CPU causes cwnd reductions
    - Not ideal at large scale (1 Gb/s * 100 ms)
  - This philosophy comes from protecting other flows
    - But we want the network to do that
    - It would be philosophically consistent to send line rate bursts and let the network deliver as much as it can

More on Relentless TCP

- Publications
  - draft-mathis-iccrg-unfriendly-00.txt
  - Paper submitted to PFLDnet (May 20-22, Tokyo)
  - http://staff.psc.edu/mathis/relentless/
- Implementation
  - Trivial to install dlkm (Linux GPL)
    - Attached to (and separate from) the main relentless page
  - Overloads DSCP=LE to enable per connection
    - Otherwise stock Reno
- Questions?

</Relentless>

Rethinking TCP-Friendly

- How can we move beyond “one-size-fits-all” CC?
- The TCP-friendly paradigm works pretty well
  - Although there are some well published problems
  - And it forbids Relentless TCP and other advances
- What might the Internet be like without it?
  - Can traffic management work at Internet scales?
  - How can we make the transition?
- The ID is intended to be a vision statement
  - Considering the good, bad and ugly
  - See draft-mathis-iccrg-unfriendly-00.txt

Goal: An alternate universe

- Routers control traffic (“allocate capacity”)
  - Segregate traffic
    - Send more losses to greedy flows
    - Shelter non-greedy flows
  - Think:
    - Fair Queuing (well not really...)
    - Approximate Fair Dropping (AFD)
    - RE-ECN
- TCP's goal is to keep the network busy
  - It is ok to be greedy (up to a point)
- Cool new property: Neither router behavior nor end-system behavior has to be standardized
  - ISPs can enforce their own “fairness” model
  - Allows TCPs to overcome adverse environments

Possible deployment scenario

1. Release Relentless TCP, LE marked
2. Other non-2581 protocols start using LE
3. LE definition is extended to include non-2581
4. ISPs implement or block LE service
   - If blocked, users complain
5. All ISPs eventually implement LE (and others?)
6. ISPs deploy traffic isolation for both services
   - Because they need better traffic controls
7. Requirement that non-AIMD use LE is relaxed

Moving the document forward

- Plan draft -01 before ICCRG/PFLDnet
  - May 20-22, Tokyo
  - (Might be a new -00)
- Please consider contributing text
  - Frame the discussion, not solve all problems
  - Best to summarize and reference existing documents
  - Want the main discussion to be crisp and clear
  - Fit the existing outline if you can
    - But do not hesitate to introduce new topics
- Volunteers?

We a talking about undoing 20+ years of IETF legacy. This is not a small change.

Backup Slides

The existing Internet "fairness" paradigm

1) Routers send independent signals to all flows
2) All flows have similar response to signals
3) This response is defined by AIMD [RFC2581]

- Modeled by $Rate = \frac{MSS}{RTT}\frac{0.7}{\sqrt{p}}$ [Mathis97]
- Defining TCP-friendly Rate Control (TFRC), etc

But there are “fairness” problems

- Non-responsive (UDP?) flows
- Applications that open many connections
- Flows with extremely different RTTs
  - TCP matches window size (short term window fair)
- Insufficient Active Queue Management (AQM)
  - RFC 2309
- Short term fair is not at all long term fair
- Defense from DOS attacks
- Many many more
  - See the ID
  - Please contribute if you are aware of more

ISP reaction

- Implement traffic controls at access routers
  - Throttle aggressive users and applications
  - Protect “normal” users and applications
- Over provision core routers
- Nearly universal for ISPs supporting home users

- Take this reaction to the limit....

Traffic Isolation is key

- Small flows are protected from greedy flows
  - Small means less than “short term rate share”
    - For whatever definition of “fair share” the ISP uses
  - Small flows don't see congestion signals sent to others
  - But may see 2nd order effects (e.g. jitter)

- This property has a useful corollary:
  - If the ISP can guarantee the threshold for small
  - The ISP can guarantee an SLA for small as well
    - Think of the instrumentation opportunities...

What are the scale limits to flow isolation?

- Historically congestion has been near the edges
  - Easy to do line rate classification and queuing
  - Currently supported in many products
  - Sometimes the “last mile” itself is sufficient isolation
    - Customer is teated as one flow
    - Can't do concurrent bulk and interactive
- Approximate Fair Dropping (AFD)
  - See [Pan SIGCOMM'03]
  - Shared (single) queue that emulates WFQ, etc
  - Much better scaling properties than separate queues
  - Is it good enough for the core?

Economic Models

- TCP friendly provides “short term window fair”
  - Rate is inversely proportional to the RTT
  - Natural incentive for users to seek near data
  - Natural incentive to deploy Content Distribution Nets
- AFD etc tends towards “short term rate fair”
  - Data rate might not depend on distance
  - Less incentive to seek near data or CDNs
  - Loss of what little locality we have
- Re-ECN might provide even better models
  - Bob Briscoe and I are trying to mind meld

Loss of implicit fairness

- Implicit fairness
  - Comes from uniform response to uniform signals
  - Means that you don't need to classify flows

Migration and Coexistence

- Consider a heterogeneous environment
  - Mixed network:
    - Drop tail, RED, or Flow Isolation with Baseline (FI)
    - With or without Lower Effort support
  - Traffic with various types of congestion control:
    - pure AIMD, pure Relentless, combined AIMD and Relentless
      - View Relentless as a generic non-AIMD CC
    - With or without LE marking on Relentless
- Which combinations have problems?
  - Combined AIMD and Relentless w/o LE or FI

    (Drop Tail or RED)
    - Relentless clobbers AIMD w/o controls in the network
    - Easy fix: segregate (or block) LE traffic

Problem cases, continued

- Pure Relentless with drop tail
  - Very likely RFC 2309 problems
  - But otherwise approximates short term window fair at 1/p
- Pure Relentless with RED
  - May be substantially under controlled and hit queue full
  - Also approximates short term window fair at 1/p
- Pure AIMD with Baseline FI
  - Sends signals too early
  - Limits performance to 75% of scheduled capacity