Safe Congestion Control

draft-mathis-tsvwg-safecc-02
Matt Mathis - IETF 116
Freelance (in collaboration with MLab)

This is very early work

- Ultimate goal: robust tests for CC safety
  - Disallow behaviors that might harm other Internet users
    - Discourage behaviors that cause self harm or user surprises
  - Ideally any CCA that passes all tests should be unconditionally safe to deploy
  - Might eventually become RFC5033bisbis
- This is a “working draft” intended for expert readers
  - No Background or tutorial explanations
  - Mostly extra terse
- Quite likely that some text or ideas will spin out into other docs
  - RFC5033bis
  - Recommendations (requirements?) for upper layers

Input needed

- Coauthors/collaborators
- Is the Safe CC coverage complete?
  - Have I overlooked some pathologies or misbehaviors?
- Is my prior art complete?
  - I have mostly missed a decade of the IETF progress
  - What important ideas/developments/documents have I missed?
  - **What important failed (unpublished) ideas have I missed?**
- Where should this work proceed?
  - Congress; iccrg; tsvwg; etc?

Concept of “under adverse conditions”

- Linguistic shorthand
  - Generally statements of monotonicity over all network conditions
    - Simple concept
    - Complicated to say precisely
    - Brutal to repeat everywhere it is needed
- Imagine testing across the “entire” parameter space
  - Bandwidth, RTT, queue space, cross traffic, random loss, etc
    - Many orders of magnitude in all dimensions
  - For all starting conditions and all incremental changes the stated property must hold

Four most important (or challenging) criteria

- Freedom from Congestion Collapse
- Freedom from Regenerative Congestion
- Upper bound on steady state loss
- Freedom from starvation

Freedom from Congestion Collapse

- Overhead/payload ratio must not increase under adverse conditions
  - Problem discovered in 1986-87 Internet collapses
  - Jacoboson88 provided a solution
- Often failures can be discovered by thought experiments on designs
- Well understood in the TSV area (and our documents)

Freedom from Congestion Collapse

- Overhead/payload ratio must not increase under adverse conditions
  - Problem discovered in 1986-87 Internet collapses
  - Jacoboson88 provided a solution
- Often failures can be discovered by thought experiments on designs
- Well understood in the TSV area (and our documents)
- Libraries and applications often fail badly
  - Pervasive use of starting over on failures (not saving partial data)
- Application designers often think:
  - “TCP will protect the network from congestion collapse”
  - They do not consider congestion collapse to be their problem

Apply Congestion Collapse tests to the entire stack

- Application bench tests
  - Run a fixed “Unit of Application work”
  - Vary network parameters across entire space
  - Flag conditions that cause increased overhead
- Can “easily” fix egregious failures
  - E.g. restart from partial data
- However none can be totally fixed
  - Signalling (e.g. SYN and SSL) must be repeated
  - Unread data in receiver’s resequencing queue must be repeated
- We can’t use MUST

Material vs Non-material

- RFC2119 language is too “absolute”
  - These are strongly suggested criteria
- Is a “violation” important?
  - The term “material” comes from US legal (court) language
- Current draft language for all criteria
  - SHOULD but MUST document exceptions
- Also need non-absolute language for “requirements”
  - Currently using “criteria”

Freedom from Regenerative Congestion

- Adverse conditions must cause increased presented load
  - Definitions are tricky here, because loss must cause additional (re)transmissions
  - However the retransmission and all future transmissions must be delayed
- Again, TSV does pretty well
- Applications less so
  - Spreading the load across additional channels with different flow-tuples

Upper bound on steady state loss

- Goal is to protect all protocols, not just other transports
  - DNS, SYN exchanges and all other single packet exchanges are particularly exposed
    - Often rely on simple RTO without prior RTT measurement
- Current draft says 2%
  - Reno and CUBIC with SACK are way out of conformance
    - 25% or 33% loss on contrived networks (Somebody test this please)
    - Unacceptably high for widespread use
  - I would rather say 0.1%
    - Probably unrealistically low
- We will need a published, well thought out justification for final text
  - Probably experimental results and a model in a separate paper

Freedom from starvation

- Large flows must not starve small and starting flows
  - The distinction between small and large must self scale
  - Must apply for all mixed traffic, with multiple CCAs
  - This may create a weak form of fairness implicit in balancing “small” vs “large
  - Efficiency (filling arbitrary networks) is explicitly NOT a goal
    - Efficiency has been proven to conflict with freedom from starvation [Arun2022SigComm]
- More important than Fairness or Efficiency on most networks
- Some criteria are easy
  - Forbid CCAs from needlessly maintaining persistent full queues
  - This may eventually become grounds for banning Reno equivalent CCAs
- Much more research is needed
  - This might also require a separate paper

Currently 13 criteria listed in the draft

- I only covered the most interesting ones
- Several others are “interesting” as well
- Read draft-mathis-tsvwg-safecc

Looking forward

- Who will help?
- Which WG(s)?
- Side tasks:
  - Research on methods to test “under adverse conditions”
  - Draft on “Congestion Control Requirement for Applications”
  - Research on plausible “Upper Bounds for Steady State Loss”
  - Research on “Freedom from Starvation”
  - Research on several lessor criteria
- Important side point: different criteria can have differing maturities
  - Can start applying some of the criteria before others are ready