Path Selection for Multiple Paths (stepping back)

(Path Selection for Multiple Paths ~~in QUIC~~)

Spencer Dawkins

Background

- Multipath transport protocols have been “a thing” in TSV for a while
  - Standards-track extensions for SCTP, TCP, proposals for DCCP, QUIC ...
  - There are interesting technical questions, both engineering and research
  - Multipath sequencing, loss detection, congestion control, reordering …
- One less-technical question - what are applications trying to do?
  - Lots of use cases have been put forward
  - Not a lot of commonality between use cases
- Path Selection for Multiple Paths ~~In QUIC~~ (ignore “in QUIC” for now)
  - ***“How many path selection strategies are enough?”***

“Thank you” to Christian Huitema for that question!

What I see, so far

- At least 10 path selection strategies described in QUIC WG, so far
  - Divided into “switching between paths” and “splitting between paths”
- The number of desired strategies is growing over time
  - It’s not obvious there is an upper bound on the number of strategies
- "Goldilocks and the Three Bears" story: “too big”/“too small/“just right”
  - “Minimum RTT”, “RTT threshold”, “equivalent RTT”, …
  - Arbitrary combinations not just possible - that’s already happening

Rather than continuing to add path selection strategies,
can we identify “building blocks”
and use them to support new strategies?

Background - Identified Path Selection Strategies

Active-Standby	RTT Equivalence
Latency Versus Bandwidth	Priority-based
Bandwidth Aggregation/Load Balancing	Redundant
Minimum RTT Difference	Control Plane Versus Data Plane
Round-Trip-Time Thresholds	Combinations of Strategies

“Identified” = “Identified So Far” 😵😵😵😵

What I'd like to do

- Assume multiple active paths will become more realistic over time
  - Not **just "**use wifi when you've got wifi, and 5G when you don't"
  - Even in 3GPP - 5G, EPS/4G, public wifi, private wifi …
  - Goal for "Path splitting" beyond utilizing all available bandwidth
- Trim down the 10 (or so) strategies as much as possible
  - I think we could maybe eliminate two or three
  - Is "latency versus bandwidth" real? No one **chooses** high latency …
- Identify "building blocks" to assemble and support new strategies

Why I’d like to do that

- Some implementers really can handle multipath just fine today
  - Either granular control or bandwidth aggregation
  - I think we’re still at the beginning of making multipath broadly usable
- I’d like to make multipath easier to use for more applications
  - “If multiple paths are available, why wouldn’t you use them?”
  - But “using multiple paths” means different things for different use cases
- I’d like to support multipath without requiring constant library upgrades
  - “How many selection strategies do you put in a library?”
  - “How many libraries do you ship with an application?”
  - ***At least some strategies get standardized - more maintenance work***

Answers I’m hoping for

- Is this research?
- Is this interesting?
- Is this doable?
- Do you want to help?

We can talk here, and in Gather, and on panrg@irtf.org,
and on Github (SpencerDawkins/quic-multipath-selection)

Thank you all!

And please, Make Good Choices