Triggers for Transport: a transport view

*

Sally Floyd

January, 2002

E2E Research Group

Past history in transport.

*

- Source Quench.
- Path MTU Discovery.
- Advice for Internet Subnetwork Designers:
Section 8.2: Recovery from Subnetwork Outages.
- L2 Trigger Bar BOF, March 2002
- Current practice.
- ...

Generic problems that triggers could cause?

$*$

- Irrelevant or misleading reports.
- Extra traffic.
- Deliberately false reports (e.g., DoS attacks).
- Traffic floods (e.g., DoS attacks).
- ???

So why are we talking about triggers again?

*

- We can learn from the past problems of Source Quench and Path MTU Discovery.

- The problems of irrelevant or false reports might be manageable.

- Explicit instead of implicit communication has its advantages.

(Along with disadvantages, e.g., being blocking by firewalls.)

- It is ok to question the tyranny of layering.

- At this stage, we are just *talking* about it.

The framework for these viewgraphs:

*

- What are the problems transport triggers might be proposed to solve?
- How would transport or applications use this information?
- How would transport or apps solve these problems without triggers?
- Are these problems important?
- What are the additional problems that triggers could introduce?

Not included in this set of viewgraphs:

*

- Possible mechanisms for triggers.
- Anything to do with routing.

Link back up?

∗

- What is the problem?

  – The transport protocol could have a backed-off retransmit timer, waiting for many seconds for it to expire.

  – TCP could have ssthresh backed off to one segment.

- How would transport use link-level information?

  – Transport could send a small probe packet before RTO timer expires.

  – TCP could increase ssthresh, for less conservative probing.

  – Transport could tell the application. Consensus (from Bernard Aboba) is that the link-up info is useful to some apps, but the link-down info is not useful.

- Are there transport-level solutions, without link-level info?
  - Occasional probing with tiny packets?
  - Probing with lower-than-best-effort packets?

- Are there link-level-only solutions?

Yep. The link keeps packets, sends them when the link comes back up. (This would work particularly well with TCP's Limited Transmit.)

- Trade-offs?
  - Explicit triggers would require less buffer space and bandwidth.
  - Explicit triggers are more complicated.
  - Explicit triggers could be blocked by firewalls.

- How important is this problem?

How much would the proposed solutions help?

Non-congestive loss for specific packets?

*

- What is the problem?

  – In the absence of specific info, TCP assumes that losses are from congestion, and reduces the congestion window.

- How would transport use link-level information?

  – ”Undo” the halving of the congestion window.

  – Decrease the sending rate slightly?

  – Notify the application?

  – Decrease the packet size?

- Are there transport-level solutions, without link-level info?

  – No. (There have been plenty of proposals...)

  – Transport could always play with changing packet sizes...

Non-congestive loss for specific packets, continued.

*

- Are there link-level-only solutions?
  – Yep. Link-level FEC, link-level retransmissions.

- How important is this problem?

How much would the proposed solutions help?

- References:
  – Explicit Transport Error Notification (ETEN), from BBN.

Link experiencing general non-congestive loss

$*$

- What is the problem?
  - Losses are a drag for the application.
  - Transport assumes that losses are due to congestion.

- How would transport use link-level information?
  - Notify the application?
  - Decrease the packet size?

- Are there transport-level solutions, without link-level info?
  - Transport could always play with changing packet sizes...
  - Heuristics for transport to infer that losses are from corruption?

- How important is this problem?

How much would the proposed solutions help?

More speculative possibilities for triggers

*

- Link going down.
- Link bandwidth increased.
- Link bandwidth decreased.

Extra viewgraphs:

*

Link going down

∗

- What is the problem?
  - The app doesn't use the remaining time well?
- How would transport use link-level information?
  - Transport could tell the application. What would the app do?
  - Transport: Wait longer before terminating connection?
  - SCTP's mobility mechanisms could use link-down information?
- Are there transport-level solutions, without link-level info?
  - Nope.
- How important is this problem?

How much would the proposed solutions help?

Link bandwidth increased

$*$

- What is the problem?

  – Transport doesn’t know to probe for newly-available bandwidth?

- How would transport use link-level information?

  – Transport could ask about available bandwidth, e.g., using an IP option like Quick-Start?

- Are there transport-level solutions, without link-level info?

  – Transport could use end-to-end mechanisms to infer bottleneck link bandwidth, and then could use something like Quick-Start.

- How important is this problem?

How much would the proposed solutions help?

Link bandwidth decreased

∗

- Are there transport-level solutions, without link-level info?

  – Transport will find out about the reduced available bandwidth after one round-trip time.

Advice for Internet Subnetwork Designers:

*

- draft-ietf-pilc-link-design-12.txt

Section 8.2: Recovery from Subnetwork Outages.
”The Internet protocols currently provide no standard way for a subnetwork to explicitly notify an upper layer protocol (e.g., TCP) that it is experiencing an outage rather than severe congestion.”
”The purpose of holding onto a packet during an outage, either in the subnetwork or at the IP layer, is so that its eventual delivery will implicitly notify TCP that the subnetwork is again operational.”

Implementation experience with Link Up and Link Down feedback

∗

- Some implementations already feed link-up and link-down info to the application at the same host.

Consensus (from Bernard Aboba) is that the link-up info is useful to some apps, but the link-down info is not useful.