Flow Balance & ICN Congestion Control:

draft-oran-icnrg-flowbalance-01

Dave Oran
Network Systems Research & Design

Outline

- Background on flow balance & its connection to congestion control
- Counting Interests vs. counting bytes
- Protocol Proposal

Flow Balance

- One Interest produces Exactly One Data
  - Can't just just inject Data packets
- Data messages bounded by L3 MTU (Not L2 MTU)
  - For NDN, this is in *theory* unbounded (MTU size is a TLV)
    - In practice, compiled in limit of implementation is 4K bytes
  - For CCNx, maximum data packet size is 64K bytes
- Fragmentation protocols have been defined to stay within link MTU

So, what's the problem?

- Small data objects inconvenient for some applications – natural object size much larger than Link MTU
  - e.g. video frames
- Applications with very small data objects
  - VoIP, Sensor Readings, etc.
- **With no info, conservative resource allocation assumes MTU-sized packets**
- Per-packet crypto overhead
  - e.g. signatures (unless you use Manifests)
  - Hashing small packets instead of larger ones

Why doesn't Fragmentation solve this?

- Possible Fragmentation schemes:
  - end-to-end
  - hop-by-hop with reassembly at every hop
  - hop-by-hop with cut-through of individual fragments
- …but this doesn't change flow balance
  - Buffer memory and link bandwidth still must be set aside for maximum-sized data objects to avoid congestion collapse under overload.

Design Considerations

- Some means required to allocate link bandwidth For Data messages
  - upper bound larger than a PMTU
  - and a lower bound lower than single link MTU
- Handle moderately sized objects (e.g. <64K bytes), not really big ones (≫100k bytes)
  - finding the right tradeoff between handling a large number of small data objects versus a single very large data object when allocating link and buffer resources becomes intractable.
- Since in many congestion control schemes, resources are allocated for returning Data based on arriving Interests, this information must be available in Interest messages.

Solution

- Simple – just add an *Expected Data Size* TLV to Interest messages
- Use this to calculate bandwidth allocation for bandwidth on the return hop instead of just counting all Interests equally.
- Except… it’s not so simple

Problem One – how to know size

- Sometimes easy:
  - For sensor and other Internet-of-Things applications: the data is instrument readings which have fixed known size.
  - In video streaming information is available ahead of time to clients in a *manifest* containing names of segments (or individual frames) of video and audio and their sizes.
  - VoIP uses vocoders that typically employ fixed-size audio frames. Therefore, their size is known either a priori, or via an initialization exchange at the start of an audio session.
- But sometimes not…
  - What if the consumer has to guess?
  - Need to consider both honest and malicious consumers

Problem Two: Data is too big

- Extra data could result in both unfair bandwidth allocation and data loss under congestion
- Three choices:
  1. Forward the data anyway, which is safe under non-congestion conditions, but unfair and possibly unstable when the output link is congested
  2. Forward the data when un-congested (e.g. by assessing output queue depth) but drop it when congested
  3. Always drop the data, as a way of "punishing" the requester for the mis-estimate.
- Need feedback though
  - Introduce a “MTU_TOO_LARGE” error code for Interest Return message in cases 2 & 3.

Problem Three: Data is too small

- Clearly no Congestion caused
  - but resources are inefficiently allocated because not all of the set-aside bandwidth for the returning data object gets used.
- Possible remediations
  - Ignore the problem
  - account for the usage according to the larger expected data size rather than actual returned data size (if forwarder does resource accounting)
  - Attempt to adjust congestion control parameters
    - Identify future Interests for the same object or closely related objects and allocate resources based on some retained state about the actual size of prior objects
    - Police consumer behavior and decrease the expected data size in one or more future Interests to compensate (TLV is a hop-by-hop header)
    - For small objects, do more optimistic resource allocation on the links on the presumption that there will be some "slack" due to clients overestimating data object size.

Problem Four: Interest Aggregation

- Perennial bugaboo since multiple Interests for same object can carry different parameters. Two cases to consider:
  1. Arriving interest carries expected data size smaller than any of the values associated with the PIT entry.
  2. Arriving interest carries an expected data size larger than any of the values associated with the PIT entry.
- Possible approaches:
  - Pick default based on link MTU of the face on which the Interest arrived and use that for all Interests lacking an expected data size. This is likely to be most compatible with simple interest counting which would rate limit all incoming interests equally
  - Configure some values for given Name prefixes that have known sizes. This may be appropriate for dedicated forwarders supporting single use cases, such as:
    - A forwarder handling IoT sensors sending very small Data packets
    - A forwarder handling real-time video with large average Data packets that exceed link MTU and are routinely fragmented
    - A forwarder doing voice trunking where the vocoders produce moderate sized packets, still much smaller than the link MTU

Problem Five: Malicious Actors

- Consumer intentionally over-estimates data size with the goal of preventing other users from using the bandwidth.
  - Deal with same way as for honest actors
- Consumer intentionally under-estimates data size with the goal having its Interest processed while the other aggregated interests are not processed, thereby causing T_MTU_TOO_LARGE errors and denying service to the other consumers. Mitigations:
  - (Simplest) Treat similarly to consumer mounting interest flooding attack
  - Remembers in the PIT entry not only the expected data size of the Interest it forwarded, but the maximum of the expected data size of the other Interests it aggregated. If a T_MTU_TOO_LARGE error comes back, instead of propagating it, treat as a transient error, drop the Interest Return, and re-forward the Interest using the maximum expected data size in the PIT (assuming it is is bigger). This recovers from the error, but the attacker can still cause an extra round trip to the producer or to an upstream forwarder with a copy of the data in its Content Store.

Proposed Encoding