Thoughts on Quality of Service for NDN/CCN-style ICN protocol architectures

Dave Oran
Network Systems Research & Design

NETWORK SYSTEMS RESEARCH & DESIGN

September 24, 2018
ICNRG Interim, Cambridge MA USA

My view of QoS

- ***NOT*** Quality of Experience (QoS actually means something technically)
- Control the allocation of resources in network elements to achieve ***managed unfairness*** of the use of those resources
  - Corollary: you cannot use QoS to create or increase resource capacity!
- Helpful in a fairly narrow range of network conditions:
  - If your resources are lightly loaded, you don't need it
  - If your resources are heavily oversubscribed, it doesn't save you
  - Failures can rapidly shift your state from the first above to the second
- History has shown QoS is needed even if not widely deployed
- QoS that works across mutually suspicious domains is an unsolved problem, which is why you don't see it on the open Internet
- QoS ≠ billing
  - (and I don't discuss how you figure out who pays for what QoS, or how you maintain enough state to generate a bill in this talk)

NETWORK SYSTEMS RESEARCH & DESIGN

What can we control to achieve QoS in ICN?

Network element resources

- ***Link** capacity*
- **Cache** capacity
- **Router memory** usage
- **Router Forwarding** capacity

Two fundamental things to specify:

- How do you create equivalence classes (aka flows) of traffic to which different QoS treatments are applied?
- What are the possible treatments and how are those mapped to the resource allocation algorithms?

How does this relate to QoS in TCP/IP?

Network element resources for IP

- **Link** capacity
- ~~**Cache** capacity~~
  - No caching at L3/L4 in TCP/IP
- ~~**Router memory** usage~~
  - Stateless forwarding pushes all memory considerations to be simply link buffering, and hence covered by Link capacity above
- **Router Forwarding** capacity
  - including replication hardware/software for multicast

Three fundamental things have been specified for IP:

- **Equivalence classes:** subset+prefix match on IP 5-tuple {SA,DA,SP,DP,PT}
- **Diffserv treatments:** (very) small number of globally-agreed traffic classes
- **Intserv treatments:** per-flow parameterized *Controlled Load* and *Guaranteed* service classes

Network Systems Research & Design

Why is ICN Different? Can we do Better? Part 1

- ***Hierarchical Names are a much richer basis for specifying equivalence classes than IP 5-tuples***
  - *QoS not pre-bound to topology since names are non-topological, unlike IP addresses*
- ***Intserv requires flow signaling with state O(#flows)***
  - *ICN, even worst case, requires state O(#active interest/data exchanges)*
- ***Diffserv limits traffic treatments to a few bits stolen from the ToS field of IP***
  - *Greenfield possibilities for more powerful treatment options in ICN*
- ***IP has three forwarding semantics, with different QoS needs (Unicast, Anycast, Multicast)***
  - *Pull-based model of ICN avoids thorny multicast QoS problems that IP has*
  - *Multi-destination/multi-path forwarding for ICN changes resource allocation needs in a fairly deep way*

NETWORK SYSTEMS RESEARCH & DESIGN

Why is ICN Different? Can we do Better? Part 2

- ***IP treats all endpoints as open-loop packet sources***
  - *NDN/CCN has strong asymmetry between producers and consumers as packet sources*
- ***IP has no caching***
  - *ICN needs ways to allocate cache resources*
  - *Treatments to control caching operation are unlikely to look much like treatments used to control link resources*
- ***Stateless forwarding and asymmetric routing in IP limits available state/feedback to manage link resources***
  - *NDN/CCN forwarding allows all link resource allocation to occur as part of Interest forwarding, potentially simplifying things considerably.*
  - *With symmetric routing, producers have no control over the paths data packets traverse*

NETWORK SYSTEMS RESEARCH & DESIGN

A strawman set of principles

Warning: I have now transitioned to opinion mode

1. Define equivalence classes (aka flows) using the name hierarchy rather than an independent traffic class definition
   - Either prefix-based (EC3) or explicit name component based (ECNT)
2. Put **consumers** in control of Link and Forwarding resource allocation
   - Do **ALL** link and forwarding (both memory and CPU) resource allocations based on Interest arrivals – schedule the reverse link direction ahead of time for carrying the matching data
3. Put **producers** in control of cache resources
   - Consumers don't care if anything is cached, at least not directly
   - **Producers** want to reduce their load and serve consumers with fewest resources
   - Some controls are already there (expiration, hold time, etc)
   - Use same equivalence class mechanism for cache resource partitioning
     - E.g. can group cache evictions by equivalence class
4. Re-think how to specify traffic treatments – don't just copy Diffserv
   - We have explicit latency control with Interest Lifetime, can we tighten this up to really manage latency-sensitive traffic? Can we play with this hop-by-hop?
   - Consider anticipatory allocation for reverse traffic (e.g. phone-home interaction styles)

Fire away!