A Practical Congestion Control Scheme for Named Data Networking

Conference Paper · September 2016

DOI: 10.1145/2984356.2984369

Klaus Schneider
The University of Arizona
5 PUBLICATIONS **7** CITATIONS
SEE PROFILE

Lixia Zhang
University of California, Los Angeles
347 PUBLICATIONS **27,377** CITATIONS
SEE PROFILE

A Practical Congestion Control Scheme for Named Data Networking

Klaus Schneider[1], Cheng Yi[2], Beichuan Zhang[1], Lixia Zhang[3]
[1]The University of Arizona, [2]Google, [3]UCLA
{klaus, bzhang}@cs.arizona.edu, cyi@google.com, lixia@cs.ucla.edu

ABSTRACT

Traditional congestion control mechanisms are designed for end-to-end connections and do not fit the Named Data Networking (NDN) architecture, in which content can be retrieved from multiple sources and through multiple paths. To fully exploit the NDN architecture, a congestion control scheme must consider the effects of in-network caching, multipath forwarding, and multicast data delivery. Moreover, the solution must not assume known link bandwidths or Data packet sizes, as these assumptions may not hold for overlay links, wireless links, or applications with varying Data packet sizes.

In this paper we propose *PCON*: a practical congestion control scheme to address the above issues. PCON detects congestion based on the CoDel AQM (by measuring packet queuing time), then signals it towards consumers by *explicitly marking* certain packets, so that downstream routers can divert traffic to alternative paths and consumers can reduce their Interest sending rates. Our simulations show that PCON's forwarding adaptation reaches a higher total throughput than existing work while maintaining similar RTT fairness. Moreover, PCON can adapt to the changing capacity of IP tunnels and wireless links, conditions that other hop-by-hop schemes do not consider.

1. INTRODUCTION

Named Data Networking [9] is an emerging network architecture that changes the network communication model from IP's host-to-host packet delivery to fetching data by names: consumers pull *Data packets* by sending out *Interest packets* to the network. This new model enables multicast data delivery, opportunistic in-network caching, and multipath forwarding [9, 25]. However, it also complicates network congestion control, as existing solutions cannot be directly applied.

Traditional TCP-like congestion control is based on established connections between two endpoints. The sender detects congestion by measuring round-trip time (RTT) or packet losses, then adjusts its sending rate accordingly. In NDN, however, the concept of end-to-end connections does not apply. Data chunks of the same content may be retrieved from different repositories or different caches along the paths towards these repositories. Since these different content sources result in varying retrieval delay, and the consumer cannot distinguish between them, traditional RTT-based timeouts become unreliable indicators of congestion.

ACM-ICN '16, September 26–28, 2016, Kyoto, Japan
© 2016 ACM. ISBN 978-1-4503-4467-8/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2984356.2984369

NDN's stateful forwarding plane enables routers to control congestion at each hop, by either dropping Interest packets or diverting them to alternative paths. However, most existing solutions in this direction (HBH Interest Shaping) [5, 22, 19, 10, 27, 26, 17, 11] assume known or predictable link bandwidths and Data chunk sizes, which limits their effectiveness in scenarios where these assumptions do not hold true. For example, the largest experimental NDN deployment, the NDN Testbed [2], runs over UDP tunnels where the underlying bandwidth is unknown and keeps changing; a video-on-demand provider may respond to congestion by adjusting the video quality (hence Data chunk size) dynamically.

This paper proposes PCON: a practical NDN congestion control scheme that does not assume known link bandwidth or Data chunk sizes. PCON routers detect congestion on their local links by using an active queue management (AQM) scheme extended from CoDel [14, 15]. When congestion is detected, a router signals this state to consumers and downstream routers by explicitly marking Data packets. The downstream routers then react by partially diverting subsequent Interests to alternative paths and/or passing the signal further downstream; consumers react by reducing their sending rate. Since in-network routers participate in avoiding congestion, we consider PCON a "hop-by-hop" congestion control scheme. However, it is fundamentally different from "Hop-by-hop Interest Shaping", as PCON routers *do not* reject Interests based on a prediction of how much link capacity the corresponding Data chunks will take up. Instead, they monitor the queue at their outgoing links, which allows to signal congestion early and to consider the available bandwidth implicitly.

We implement PCON in ndnSIM [12] and evaluate its performance against a delay-based [6] and a HBH Interest Shaping scheme [22] from the literature. We show that PCON prevents congestion in the case of varying RTTs, dynamic Data chunk sizes, IP tunnels, and wireless links. Moreover, PCON's forwarding adaptation (based on explicit congestion marks) achieves a higher throughput than the exiting solutions that are based on equalizing pending Interests. PCON is able to adapt to the changing available bandwidth of IP tunnels and wireless links, a case that is not considered by current HBH Interest Shaping proposals. We identify a few remaining issues as our future work, including the definition of "fairness" in data-centric networks and effective handling of unresponsive consumers.

This paper is organized as follows. Section 2 explains the design rationale of PCON. Section 3 describes PCON's design specifics and Section 4 evaluates its performance through simulations. Section 5 compares PCON with related work. Section 6 concludes the paper with an outlook to the future.

2. DESIGN RATIONALE

NDN's new communication model requires us to rethink the traditional congestion control mechanisms which have been shown to work well in TCP/IP. Here we elicit four groups of unique NDN characteristics and show how they lead to our design approach.

1) Multiple Paths and Endpoints.

As a data-centric architecture, NDN removes the concept of connections between two endpoints and enables *ubiquitous caching*, *native multicast*, and *multipath forwarding*. Since an NDN Interest may be answered by any cache along the way to the content repository, the consumer application cannot tell where the corresponding Data packet has originated from. Even if the knowledge about the origin was carried in the Data packet, the consumer would still not know where the next Data packet in the stream would come from (except using the solutions from [20, 4]). This limitation leads to the following three problems:

First, traditional TCP retransmission timers [8, 18] perform poorly, as they assume a normal RTT distribution *between two end nodes*. In NDN, however, data may be retrieved from multiple nodes or through multiple paths. Moreover, NDN routers aggregate Interests of the same name into one Pending Interest Table (PIT) entry and deliver the returning Data packet to all aggregated faces. Interests that arrive after the first will see a shorter retrieval delay (since the Data is already on its way). These three cases (multi-source, multipath, and PIT aggregation) can result in periods of short RTT measurements where the RTO setting (meanRTT + 4 * varRTT) falls too short, leading to a decreased sending rate and unnecessary retransmissions. One can avoid these false positive timeouts by setting a higher minimum RTO, but that also increases the time to detect packet losses, thus the reaction time to network congestion. PCON avoids this dilemma by combining a high minimum RTO with *explicit congestion signaling* for fast reaction.

Second, the optimal congestion window for the consumer (defined as the bandwidth-delay product – BDP) can change due to cache diversity: If one cache runs out of data, the next Interests will be answered by another cache (or the producer) further away. If the path to the new cache has a lower BDP, the high number of in-flight Interests may overflow the queue at the new bottleneck, before the consumer is able to adapt its sending rate. One may try to avoid such *cache switchovers* by using a caching policy that always retains complete content objects. However, this is not feasible when caches never see the full object in the first place (e.g., due to multipath splitting of traffic or consumers requesting only parts of a content object). We mitigate this problem by making buffers large enough to handle temporary traffic bursts, and by detecting and signaling congestion with the CoDel AQM scheme *before* the buffers reach their limit (see Section 4.2).

Third, a consumer can no longer use gaps in sequence numbers from received data packets (out-of-order delivery) as a sign of packet loss, hence congestion, as out-of-order arrivals could be caused by receiving data from different caches, or along multiple paths.

2) Pull-based Data Retrieval & HBH Flow Balance.

NDN consumers send out Interests to retrieve Data packets. Since Data packets always follow the "breadcrumbs" of their corresponding Interests, to control congestion over a link from nodes B to A, A must control the rate of Interests it sends to B. In theory, NDN's *Hop-by-Hop Flow Balance* (over each link, one Interest packet brings back exactly one Data packet [1, 9]) can be used to control the link load from B to A. In practice, however, that would require a-priori knowledge of the link capacity, which may not be available for IP tunnels or wireless links. Instead of trying to estimate the link bandwidth or expected load, PCON uses the packet queuing time [14] and local link loss detection (Section 3.5) as more reliable indicators of congestion.

3) Diverse Deployment Scenarios.

NDN must work in a diverse range of deployment scenarios, including *IP-Overlay* links and *wireless* links. One example of IP-Overlay deployment is the NDN Testbed [2], where NDN can run over either TCP or UDP tunnels. TCP tunnels provide reliable delivery (similar to a wired link) with the trade-off of increasing the total data retrieval delay, which can be unnecessary and even undesirable for loss-tolerant and delay-sensitive applications (like audio/video conferencing). Since NDN needs to work for all applications, the Testbed currently runs UDP tunnels. However, UDP tunnels may drop packets without notification and their capacity is quite unpredictable; it may vary continuously depending on the competing cross-traffic in the IP underlay network.

Another common NDN deployment scenario is wireless links. As with IP overlay links, the achievable bandwidth of a wireless link is unknown and can be highly variable; it depends on the wireless interferences, wireless traffic load levels which result in packet collisions, and some other factors.

PCON considers both wireless and IP-Overlay networks by detecting silent drops on these networks and signaling them back with explicit NACKs (see Section 3.5).

4) Flow Semantics and Fairness.

Fairness is an important topic for NDN transport control. However, the existing concept of "per-flow fairness" does not directly apply to NDN. In IP networks, a flow refers to traffic between *two endpoints*, often identified by the 5-tuple of src/dst IP address, src/dst port, and the transport protocol. In NDN, packets may not flow between two fixed endpoints. One can define an NDN *content flow* by requests for a specific *content object* from one or multiple consumers [16]. To identify such a flow, one needs a global agreement on the definition of a content object. For example, for a layer-encoded video, would the whole video be considered a single content object or, as some consumers may fetch a sub-layer only, each sub-layer a separate one? Even assuming we can agree on the definition of contents, "per-content fairness" remains difficult, as some NDN routers may "see" only a subset of a content flow when other chunks are answered by prior cache hits or sent elsewhere by multipath forwarding.

Furthermore, NDN's multicast data delivery creates a new trade-off between efficiency and fairness. A router does not know whether an Interest is generated by a single consumer, or represents an aggregation of Interests from multiple consumers further downstream. The latter case (retrieving data for multiple consumers) is arguably more important, but a simple "per-content flow fairness" cannot give it a higher preference. One could account for the higher demand in a virtual control plane [24], which however incurs more complexity and state overhead.

We evaluate the per-consumer fairness of PCON in Section 4.4. However, the above consideration suggests that more work is required to define the fairness metric in an NDN network (e.g. whether fairness should be based on contents or on consumers). Thus, we leave the topic of fairness to future work.

3. SYSTEM DESIGN

Following these considerations, we design PCON with the principles of *1) early congestion detection by adopting an AQM*, *2) explicit congestion signaling*, *3) exploiting multipath forwarding*, and *4) spe-*

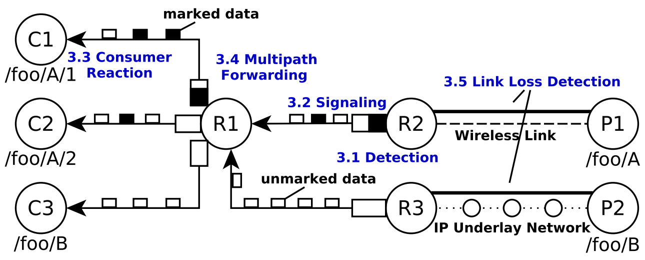


Figure 1: System Architecture

cial consideration of IP-overlay and wireless links. As shown in Figure 1, PCON consists of five components:

- *Congestion Detection* (3.1): Each node detects congestion locally by monitoring its outgoing queues.
- *Congestion Signaling* (3.2): After detecting congestion, nodes mark Data packets to inform downstream routers and consumers.
- *Consumer Rate Adjustment* (3.3): End consumers react to congestion signals by adjusting their Interest sending rate.
- *Multipath Forwarding* (3.4): Downstream routers react to congestion signals by adjusting their traffic split ratio.
- *Local Link Loss Detection* (3.5): On wireless and IP overlay links we locally detect packet loss and signal it back using NACKs.

3.1 Congestion Detection

As Nichols and Jacobson have pointed out [14, 15], the most reliable place to detect congestion is at the link where it occurs. Thus, we detect congestion by monitoring the outgoing queues of each router, instead of trying to infer congestion at the consumer by monitoring packet loss or round-trip time (as discussed earlier, these techniques are less reliable in NDN than in IP networks). To detect congestion on router queues, we use a well-studied AQM mechanism, CoDel [14, 15], as it allows routers with large buffers to both absorb traffic bursts and to keep their queues small, by detecting congestion *before* the buffer overflows. CoDel measures the queuing delay ("sojourn time") of each packet on its outgoing links. If the minimum sojourn time over a time period (default: 100ms) exceeds a threshold (default: 5ms), it considers this link as congested. CoDel's use of queuing delay instead of queue size fits our design assumption of unknown link capacity: queuing delay implicitly considers the link bandwidth (for a fixed queue size, higher bandwidth leads to lower queuing delay).

PCON monitors congestion in both the downstream (Data) and the upstream (Interest) direction. Interests can cause congestion when they are larger than Data packets, on asymmetric links, or when there is cross-traffic in the upstream direction. After detecting congestion in the Interest direction (like R1–R2 in Figure 1), R1 will *mark the PIT entry* of that Interest, and consider the corresponding Data for congestion signaling.

In addition to monitoring outgoing queues, we use two alternative ways of detecting congestion. The first is used in scenarios where the state of the router queues is inaccessible, as in UDP tunnels. Here we infer the congestion level by monitoring packet loss at the local link (see Section. 3.5). The second addresses another problem that can be considered "congestion": A PIT that is growing too large and threatens to overflow the router's memory (which is analogous to a TCP sender overflowing the receiver buffer). Just like congestion due to insufficient link bandwidth, the state of overgrowing PIT can be signaled downstream to reduce the number of incoming Interests. However, in this paper, we focus on bandwidth-related congestion and leave the detection of overgrowing PIT for future work.

3.2 Congestion Signaling

After a router has detected that one of its outgoing links is congested, it signals this state to the consumers and to all routers along the path. Signaling is only necessary in the downstream direction, since only downstream routers can reduce the amount of traffic that the signaling router will receive. Thus, we signal congestion by marking NDN *Data packets*, but not Interests.

We signal congestion similar to CoDel's use of ECN marking: The first packet is marked when entering the *congested* state and later packets are marked in an increasing "marking interval" (similar to CoDel's drop spacing); the marking interval starts at 1.1 * the CoDel interval (110 ms) [15] and is decreased in inverse proportion to the square root of number of marks, in order to achieve a linear decrease in the consumers sending rate. On congested upstream links, we use the same marking interval for setting the flag in the PIT entry, which then marks the associated Data packets.

Due to NDN's built-in multicast data delivery, one marked Data packet may branch out to reach multiple consumers, which will all slow down their rate. This is exactly the desired behavior, because if any of the consumers fails to slow down, the congested link will not see a reduction in rate at all (it would just see less PIT aggregation).

One example of the signaling mechanism is shown in Figure 1. Consumer C3 retrieves data from P2 (under the prefix /foo/B). Since there is no congestion on the path, C3 only receives unmarked Data packets. C2 retrieves data from P1 (/foo/A/2) and receives congestion marks from R2, since the link between R2 and R1 is congested. C1 also retrieves data from P1, but under a different prefix (/foo/A/1). In addition to receiving congestion marks from R2, C1 also receives them from R1, since the link R1–C1 is congested too. This allows C1 to adjust to the bottleneck at R1, while C2 adjusts to its bottleneck at R2.

3.3 Consumer Rate Adjustment

Once the congestion marks reach the consumer, the consumer has to decide on how to adapt its rate. We use a congestion window (specifying the maximum number of in-flight Interest packets) which is increased on unmarked Data packets and decreased on marked ones, NACKs, and timeouts. Data packets act as *selective acknowledgements* of Interest packets, meaning that each Data packet acknowledges one corresponding Interest. Selective ACKs require some thought in our design, since NACKs/timeouts often occur in bursts (on buffer overflows). If we perform a multiplicative decrease on each packet loss in a burst, the congestion window is reduced too drastically (often back to 1) after a single congestion event. We prevent this over-adjustment by applying the "TCP SACK-based Conservative Loss Recovery Algorithm"[3] (see Algorithm 1): The consumer performs at most one window decrease per RTT.

We don't (need to) apply the "Conservative Loss Adaptation Algorithm" to marked packets, since the CoDel marking already avoids bursts of marks. Moreover, CoDel marking adapts to the congestion level (marks become more frequent when congestion persists) while the loss adaptation algorithm does not (it always allows at most one window decrease per RTT); thus disabling it allows the consumer to react faster to highly congested links (see Section 4.3). We still use the conservative loss adaptation for timeouts (resulting from buffer overflows) and NACKs resulting from buffer overflows in the IP underlay network (see Section 3.5).

With the conservative loss adaptation and CoDel marking in place, PCON can implement a number of classic loss-based TCP algorithms like Taheo, Reno, New Reno, HTCP, HSTCP, BIC, and CUBIC. The only difference to traditional TCP is that a window decrease is triggered not only by timeouts, but also by packet marks (similar to TCP ECN marks) and NACKs. After experimenting with TCP Reno,

Algorithm 1 NDN Conservative Window Adaptation[1]

```
1:  function OnData(dataPkt, seq)
2:      if m_highData ≤ seq then
3:          m_highData ← seq;
4:      end if
5:      if dataPkt.isMarked() then
6:          windowDecrease();
7:      else
8:          windowIncrease();
9:      end if
10: end function
11:
12: function OnTimeout(seq) // Same for OnNack()
13:     // Only one window decrease per RTT
14:     if $m\_highData > m\_recPoint$ then
15:         m_recPoint ← m_highInterest;
16:         windowDecrease();
17:     end if
18: end function
```

we switched to TCP BIC [23], because (similar to TCP CUBIC) it performs better on high BDP networks.

When router buffers are large enough, PCON removes the traditional sources of packet drops: *1) Router buffer overflows* (it prevents them by explicitly signaling congestion), *2) Drops by the AQM scheme* (it marks packets instead), and *3) Drops by the "link"*, like wireless or UDP tunnel (it either recovers the loss by in-network retransmission or sends explicit NACKs – see Section 3.5). Therefore, the only sources of packet loss should be malfunctioning (host or link failure) or misconfiguration. This provides a solution for the high RTT variance caused by caching and multipath: We can use higher RTOs to avoid false positive timeouts and still achieve fast retransmissions, as packet loss is either avoided or explicitly signaled.

3.4 Multipath Forwarding

The scheme described so far will effectively prevent congestion when used with single-path routing, as it adapts the consumers' sending rate to the available network bandwidth. Below, we describe how PCON routers use the NDN forwarding layer to shift traffic away from congested paths.

We design our multipath forwarding strategy with the objective to *maximize end-user throughput* while keeping the network cost (as measured by path length) as low as possible. We assume that the strategy has access to a ranked list of path lengths, set by the routing protocol. The optimal strategy decision at each hop depends on the demanded rate (determined by the capacity of the downstream links and the consumer) relative to the capacity of the outgoing links. If the demand can be satisfied by the shortest path, traffic should not be split on other paths. If the demand exceeds the capacity of the shortest path, it should be incrementally distributed on the next-shortest path(s). If the demand exceeds the sum of the capacity of all available links, it should be split in a way to fully take up the bandwidth, irrespective of path length.

Instead of adjusting the forwarding ratio to pending Interests or round-trip time like related work (see Section 4.4), we design our forwarding adaptation based on *received congestion marks*. For each FIB prefix, PCON maintains a forwarding percentage of each face,

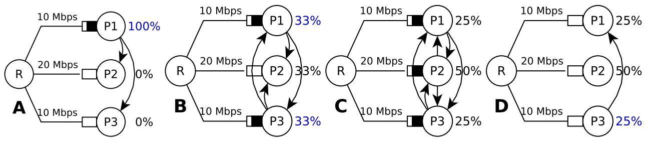


Figure 2: Multipath Forwarding Example

which is initialized to 100% for the shortest path (given by the routing protocol) and 0% for all other paths. Then for each marked Data packet, it reduces the forwarding percentage of the incoming face, while at the same time increasing the forwarding percentage of all other faces to an equal amount:

$reduction = fwPerc(F) * \frac{CHANGE\_PERC}{f(Distance)}$

$fwPerc(F) -= reduction$

$fwPerc(\bar{F}) += \frac{reduction}{NUM\_FACES-1}$

fwPerc(F) is the forwarding percentage of the current face and *CHANGE_PERC* is a fixed parameter, which makes a trade-off between how fast it adjusts the forwarding ratio to the optimal value (higher is better) and how much it oscillates around that value (lower is better). Through experiments, we found a value of 1%-3% to work well for a number of different bandwidths. *f(Distance)* is a factor to adjust the forwarding ratio more strongly the closer the adjusting router is to the congested link. We discuss it later in this section.

For a new FIB prefix, our forwarding scheme achieves the goal of staying on the shortest path until the demand exceeds the capacity of that path (only then will congestion marks arrive and change the forwarding ratio). It also achieves the goal of taking up all available network bandwidth (converging to the optimal split ratio) when the demand is high. However, when the demand falls again later in the lifetime of a FIB entry, we need a mechanism to shift traffic back to the shortest path (which is now underutilized). We detect free link capacity by not receiving any congestion signals for at least a "marking interval" (110 ms) and then shift a fixed percentage of traffic from the longest to the shortest path (e.g. 10% of CHANGE_PERC). After some time, this will move the full traffic back to the shortest path, if its capacity is large enough (if not, the arriving congestion marks will shift the traffic to other paths again).

We show an example of our forwarding adaptation in Figure 2. First, R sends traffic on the shortest path towards P1. If the rate stays below 10 Mbps, traffic will stay on that path. If not, R will see congestion marks from P1 and starts shifting traffic to other links (part A). Once the traffic is split equally on all links, R will see marks from both P1 and P3 (part B), and will reduce both their ratios, thus increasing the one of P2. In the final state (part C), R has reached the optimal split ratio and receives marks from all 3 routers. The split ratio will oscillate around this equilibrium and react to changes of the unbound capacity of the links, as seen by the received congestion marks. If the demanded rate falls below 40 Mbps (part D), none of the links will see congestion marks, and R will shift traffic from the longest path (P3) back to the shortest path (P1).

One remaining question is the dynamics of multiple routers adjusting their forwarding ratio along the path of the marked Data packets: with multipath forwarding, routers that are close to the content repository see less congestion marks than routers that are close to the consumer, as the latter see marks from many more links. To avoid over-adjustment at these routers and to address congestion closer to where it occurs, we adjust the forwarding ratio less at routers far away from the congested link: $f(Distance) \geq 1$. The factor f(Distance) can be based on the number of hops between the router and the congestion point, on the number of routers that already have adjusted their ratio earlier (closer to the congested link), or on the

[1]m_highInterest denotes the highest sequence number of an Interest that the consumer has sent so far; m_highData denotes the highest sequence number of any received Data. m_highInterest, m_highData, and m_recPoint are all initialized to 0. Note that these sequence numbers are only used at the consumer and are independent from the Interest name; an application may choose an arbitrary naming scheme as long as it locally keeps track of the sequence in which packets are sent and received.

Algorithm 2 Router Signaling and Forwarding Adaptation

```
 1: function OnData(dataPkt, incomingFace, fibEntry, pitEntry)
 2:     if dataPkt.isMarked() then
 3:         reduceFwPerc(fibEntry, incomingFace, CHANGE_PERC);
 4:     end if
 5:     // For each DownstreamFace
 6:     for dsFace IN pitEntry do
 7:         if dataPkt.isMarked() OR dsFace.isCong() OR pitMarked then
 8:             MarkOutgoingData();
 9:         end if
10:     end for
11: end function
12:
13: function OnNack(incomingFace, fibEntry)
14:     // Only one reduction per CoDel Interval
15:     if time::NOW >= lastUpdate + CODEL_INTERVAL then
16:         reduceFwPerc(fibEntry, incomingFace, CHANGE_PERC);
17:         lastUpdate = time::NOW;
18:     end if
19: end function
```

number of alternative links on the path. For our evaluation, we use a simple linear reduction (f(Distance) = HopCount to congested link), and leave the optimal setting of f(Distance) for future work.

A related question is the dynamics between the router forwarding adaptation and the consumer rate adaptation. For PCON, the consumer reacts more strongly to each congestion mark (e.g., by halfing the cwnd) than routers, because the consumer can also react to Data packets as sign of available bandwidth, while routers only react to the much rarer congestion marks. However, this topic may need further study.

Another addition to the forwarding scheme is to consider the trade-offs in using paths with a very low forwarding ratio. There are costs associated with using multipath forwarding instead of single-path forwarding: Adding a new path can lead to higher RTTs, more out-of-order delivery, or even packet drops. These costs are offset by the additional bandwidth that the new path contributes. However, if the split ratio of the path is very low (e.g. less than 5%) the additional bandwidth might not be worth the cost. Thus, the strategy can set a *lower threshold* for each link, below which it will disable the link, but maintain its forwarding percentage and send probing traffic to adapt the percentage to congestion marks. Should the percentage exceed the threshold, the strategy will enable the path again.

3.5 Local Link Loss Detection

The design described so far assumes that the only cause for packet loss in the network is router buffer overflow, which is the case for wired links with low bit error rate, but not for IP tunnels or wireless links. NDN routers connected by UDP tunnels experience packet losses when the IP underlay network has congestion or routing problems. Since we can't access the router queues inside the UDP tunnel, to the NDN layer, a UDP tunnel looks like a link that may drop packets unpredictably, often in bursts. Wireless links behave in a similar way: interference, multipath propagation, or disconnections can all result in packet losses. The wireless link layer may use FEC codes and retransmissions to mitigate those losses, but does not guarantee reliable delivery; after exceeding a preset retransmission limit, it silently drops the packet.

Since silent packet drops are costly in NDN (see Section 2), we avoid them by implementing a shim layer (like [21]) on top of loss-prone NDN link abstractions (which should not be confused with the OSI data link layer, as NDN can run on top of different layers of the protocol stack). The only functionality this shim layer needs to implement is to reliably indicate whether a given packet was lost on a certain "link". It may report packets as lost which weren't lost (false positive), but it may never fail to report a packet that was actually lost (false negative), as consumer and PIT timeouts are more costly than unnecessary retransmissions. We call this notion *reliable loss detection*, which is different from reliable packet delivery. We achieve reliable loss detection by using *positive acknowledgments* for each frame (Interest or Data packet) and detecting losses by observing gaps in sequence numbers or timeouts of the ACKs. This gives the sender a definite report of packet loss in only a small multiple of the link propagation delay (when accepting the chance of false negatives). In contrast, with negative link ACKs the receiver is responsible for detecting the loss, and if the NACK is lost, the sender might never know about the original link loss.

The knowledge of packet loss is signaled back to the consumer to allow a faster and more reliable retransmission, compared to relying on timeouts. The signaling works slightly different depending on whether the loss happens on a wireless or IP overlay link: For IP overlays, the loss can be signaled back to the consumer with a *marked NACK*, where the reaction is the same as to a marked Data packet. However, since these NACKs might happen in bursts, the consumer needs to apply the conservative loss adaptation algorithm, that is, only reduce the cwnd once per RTT. Routers in the network reduce their forwarding ratio of these links and need to apply a similar mechanism: they only perform one reduction of the forwarding percentage per CoDel interval (100ms), which is an estimate of the upper threshold of consumer RTTs in the network (see Algorithm 2).

For wireless links, the mechanism is simpler: the loss is signaled with an *unmarked NACK* and only triggers a consumer retransmission, but no change in sending rate or forwarding ratio. The queues at the end of each wireless link are still able to effectively signal congestion (by marking Data packets), should it occur.

Although this reliable link loss detection is more complex and less effective for controlling congestion levels than measuring queuing delay, we show in Section 4.6 that it still allows the consumers to adapt to congestion in the IP underlay network and to recover wireless packet losses.

An addition to the link adaptation layer is to locally retransmit a packet a certain number of times before reporting the loss to the network layer [21], which can improve the performance by reducing the number of consumer retransmissions. However, the more important task of signaling the congestion level in the underlay network to both routers and consumers is achieved even without in-network retransmissions.

3.6 Highly Congested Links

In some cases, PCON cannot prevent congestion on a certain link: the congestion may be caused by unresponsive consumers, IP cross-traffic, software failure, overflowing PIT entries, or hardware limitations. Even though we cannot avoid congestion in these cases, we can still avoid using these "highly congested" links by disabling them in the forwarding strategy. There are multiple ways to detect the highly congested state; for example, when the queue keeps persisting after many intervals of congestion signaling, or when the router memory approaches its limits. Whenever a router enters the highly congested state, we signal it by marking *all* outgoing Data packets with two additional bits: one that indicates this state ("highCong"), and one that indicates whether a highly congested link is a direct neighbor or not ("highCongLocal"). Marking all packets allows us to know whether the state persists in the adjacent router, which isn't possible with the spread-out CoDel marks.

Whenever a router receives a packet with the high congested state or detects this state on its own upstream links, the router disables this path for Interest forwarding. After disabling the link, a router checks the FIB entry for alternative links, and if it doesn't find one,

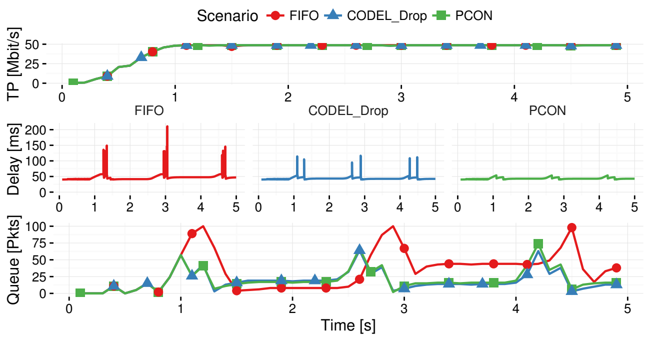


Figure 3: Consumer Rate Adjustment and AQM

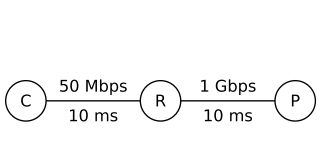


Figure 4: Topo 4.1

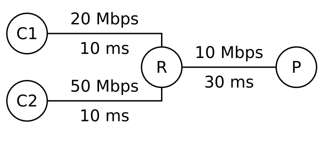


Figure 5: Topo 4.2–4.3

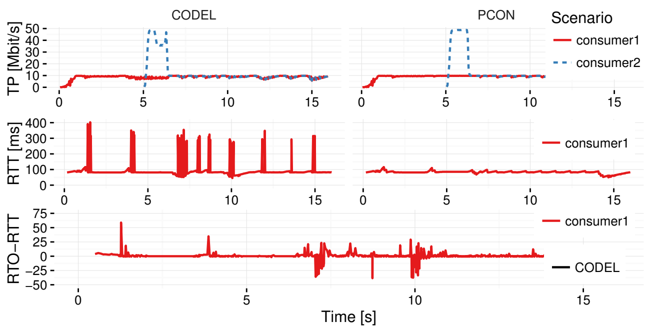


Figure 6: Caching and Multicast

it propagates the highCong state further downstream. It also resets the highCongLocal bit, so that the next further downstream router knows that this state only applies to given FIB prefix, but not to the whole link it is connected to. The second downstream router then disables the path only for this FIB entry. Note that, when no other option is available, a router still forwards Interests on disabled paths, and expects prior downstream routers to choose better alternatives.

One could use NACKs to signal the highCong state, but our marking scheme has the benefit that marked packets *still contain the data*, and thus reduce retransmissions. They can be used in the case where the upstream is still able to deliver data, but wants to strongly discourage its use. In Section 4.5, we show how the highly congested state improves performance in the case of non-reactive cross-traffic.

4. EVALUATION

We evaluate PCON using ndnSIM 2.1 [12] in various scenarios to demonstrate the effectiveness of its decisions. In two of those scenarios (4.4 and 4.6), we compare PCON against related work from the literature: one delay-based scheme and one based on HBH Interest shaping.

4.1 Consumer Rate Adjustment and Router Queue Management

We first evaluate the effects of using an AQM scheme (CoDel) and explicit congestion signals. We use a linear 3-node topology (Figure 4) with a single consumer and no caching. We compare three different cases:

- **FIFO:** A naive implementation of TCP BIC with FIFO queues and the consumer using the conservative loss adaptation.
- **CODEL_Drop:** Routers use CoDel dropping queues instead of drop-tail FIFO; consumers *don't* use conservative loss adaptation.
- **PCON:** As CODEL_Drop plus explicit congestion signaling.

In all scenarios, we use the TCP BIC window adaptation and the traditional TCP RTO estimation (which performs properly, since there is no cache diversity or multipath). BIC uses a "slow start" phase with a pre-configured "maximum increment" that specifies how much the cwnd maximally increases per RTT (here: 16 Interests). After a loss event, it uses a "binary search" to rapidly take up available bandwidth, similar to the initial slow start phase (see [23] for the details). A loss event is triggered either by a timeout (FIFO, CODEL) or by a marked Data packet (PCON).

The results are shown in Figure 3. We found that even in this simple scenario, using timeouts from drop-tail queues (FIFO) is problematic: they always need to completely fill the queue, potentially lead to persistent queues, and increase the delay of retransmissions (as discussed in Section 2, we can't use out-of-order delivery like TCP "fast retransmit" in NDN). CoDel and PCON have the benefit that they react before the queue reaches its limit (here 100 packets). Raising the queue size to 1000 packets would have no impact on CoDel/PCON, but heavily increase the queuing delay for FIFO queues. Moreover, PCON's explicit marking scheme achieves the same throughput as the others, but avoids packet drops and thus retransmission delays (spikes in the Delay plot indicate retransmissions).

4.2 Caching and Multicast

Next, we evaluate the effect of caching and RTO timer settings. We use two consumer nodes retrieving the same data (see Figure 5), with C2 starting at a later time point (5s). Moreover, we increase the delay of the bottleneck link (between R and P) to 30ms, to emulate a situation where consumers have access to a close, high-bandwidth cache and a far away lower-bandwidth repository. We compare PCON against CoDel dropping with traditional TCP timers at the consumer (RTO = meanRTT + 4*varRTT).

The results are shown in Figure 6. First, C1 is the only consumer and adapts its sending rate to the bottleneck bandwidth of link R–P, namely 10 Mbps. At 5s, C2 starts and initially gets all the data from the cache at R (with a higher bandwidth of 50 Mbps). At 7s, C2 exhausts the cache and joins C1 in retrieving data from P. This is where the problems of TCP timers begin to show: Both consumers retrieve the same content, thus both occasionally get data from the cache (R) with an RTT of around 20 ms. This smaller RTT reduces the RTO, often to a value smaller than the RTT of the following packets (see "RTO - RTT"), leading to *unnecessary retransmissions*. PCON avoids this problem by setting a fixed higher RTO ("RTO - RTT" is always positive) and using explicit marks for the window adaptation: It achieves a faster congestion reaction (marked packets reach the consumer faster than the RTO) and doesn't cause any packet drops or retransmissions.

4.3 Changing Data Packet Size and Cross-Traffic

Next, we evaluate the effect of sudden changes in Data packet size and available link bandwidth (caused by unresponsive cross-traffic). We use the same topology as in the last measurement (Figure 5), but only C1 retrieves regular Data packets; C2 is used only for cross-traffic. We look at 4 different scenarios:

- **Payload:** At 5s, the Data payload size is suddenly increased to 50x its original value (from 100 Bytes to 5KB). At 15s, it is reduced back to the original value.

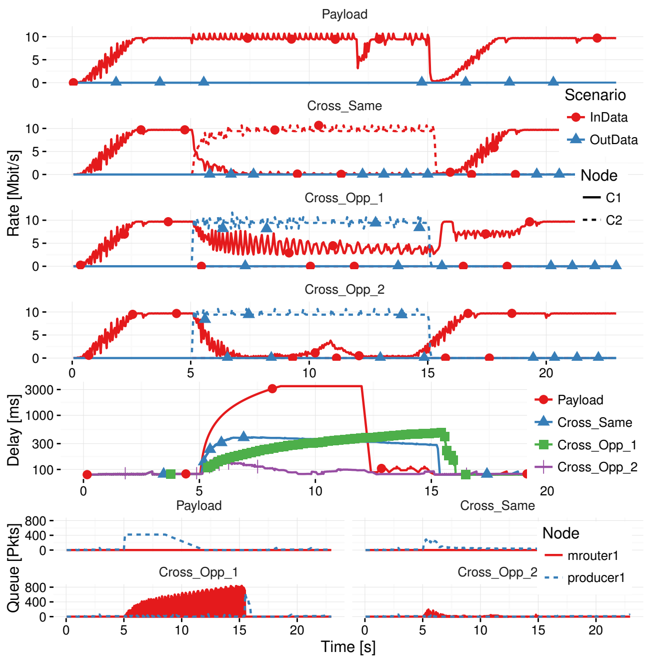


Figure 7: Changing Data Packet Size and Cross-Traffic

- **Cross_Same:** C2 requests an unresponsive, fixed rate of cross-traffic with 98% of the available bottleneck link bandwidth (reducing the available bandwidth to 1/50 of the original). At 15s, the cross-traffic stops.
- **Cross_Opp_1:** The same amount of cross-traffic as in Cross_Same, but now in the opposite direction (C2 sends data towards P), resulting in congestion on the Interest path. First, we ignore this congestion and only consider congestion on the Data path (marked packets).
- **Cross_Opp_2:** Same as Cross_Opp_1, but now the routers react to congested Interest packets by marking the PIT entry and then marking the returning Data packet.

The results (Figure 7) show that when the Data chunk size suddenly increases (at 5s), the router queues will fill up and the consumer will experience a temporary higher delay, proportional to the change in Data size (here: 50x the RTT of 80 ms, that is, 4 seconds). PCON then quickly reduces the cwnd, so that the queue starts draining and comes back to the normal value at about 12s. At 15s, after the Data size returns to the original 100 Bytes, the consumer rate drops, but the consumer quickly adapts the cwnd to again take up the full bandwidth (at about 18s).

The cross-traffic in the same direction has a similar effect, but creates less delay (300 ms; note the exponential scale). In both scenarios, the consumer quickly reacts to packet marks and reduces its sending rate. If the Interest lifetime (and RTO) is long enough, this works without packet loss; otherwise some packets may be lost (and retransmitted). Nonetheless the adaptation works the same way.

The scenario of cross-traffic in the opposite direction shows why it is necessary to consider the congestion status of Interest packets: If we don't (Cross_Opp_1), the queuing delay at R1 (and thus the consumer RTT) rises indefinitely and the first sign of congestion will be a packet loss due to router buffer overflow. If we consider congestion on the Interest path (Cross_Opp_2), we avoid the queue buildup, at least when the cross-traffic stays below the available bottleneck link bandwidth. For a solution when the cross-traffic exceeds the available bandwidth, see Section 4.5.

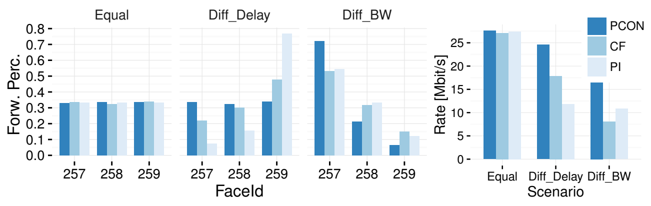
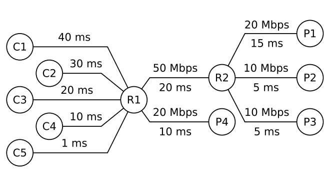


Figure 8: Multipath Forwarding: PCON vs. PI vs. CF [6]



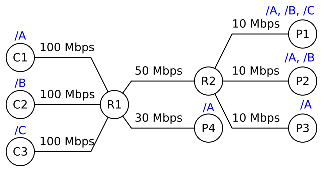


Figure 9: Topo 4.4 **Figure 10: Topo 4.5**

4.4 Multipath Forwarding

Next, we compare PCON's multipath forwarding adaptation (based on congestion marks) against alternatives from the literature. Starting from Detti et al.'s review [7], we compare PCON against pending Interest adaptation, as all of the alternatives make stronger assumptions: "Fast Pipeline Filling" [7] assumes knowledge of the pipeline capacity and "RTT Equalization" has the problem of potentially mixing RTT measurements from different "flows" or different cache hits, leading to a suboptimal forwarding ratio.

PI-based schemes don't make these assumptions and the information about pending Interests is readily available at the forwarding layer. Thus, we consider two of these schemes: *PI*, which chooses the face with the lowest number of pending Interests, and *CF* [6], which uses a weighted round-robin distribution: $weight(face, prefix) \leftarrow 1/avgPI(face, prefix)$, where "avgPI" is the exponential moving average of the number of pending Interests.

In each measurement, we waited until the forwarding split ratio has stabilized and then take its average. We use three scenarios in a topology composed of the nodes R2, P1, P2, and P3 in Figure 9 (where R2 acts as the consumer):

- **Equal:** 3 paths with an equal bandwidth of 10Mbit/s and equal RTT of 20ms.
- **Diff_Delay:** 3 paths with an equal bandwidth of 10Mbit/s, but a different RTT of 100ms (face 257), 50ms (face 258), and 10 ms (face 259).
- **Diff_BW:** 3 paths with an equal delay of 20ms, but different bandwidth of 22.5Mbps, 6 Mbps, and 1.5Mbps. Here we expect a different split ratio of 75%–20%–5%.

The results (Figure 8) show that PI-based distribution (CF and PI) works well when the bandwidth and delay of all available paths is the same: it achieves the optimal split of 33%–33%–33%. However, when delays differ, both of them prefer the lower-delay path at a cost of total throughput (CF achieves a slightly better split ratio and higher bandwidth than PI). When the bandwidth differs, both CF and PI bias against high-bandwidth paths (instead of the optimal ratio of 75%, they only achieves about 54%), again reducing the total throughput. These findings confirm Nguyen et al.'s analysis [13] that PI-based forwarding biases against paths with longer RTT or higher bandwidth. Because PCON responds directly to the congestion state of the involved routers, it achieves the expected split ratio in all three cases, which results in a significantly higher overall throughput in the cases of different delay and different bandwidth.

Next, we compare the full PCON scheme against an alternative

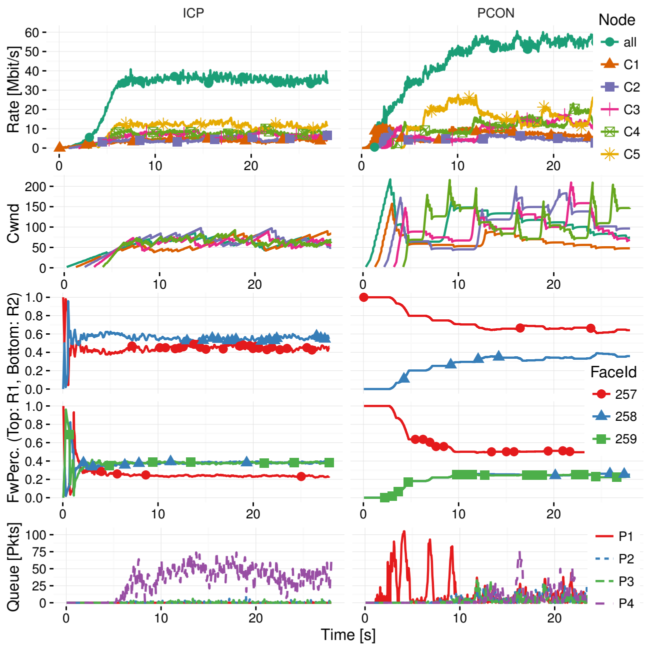


Figure 11: Multipath Forwarding: ICP (left) vs. PCON (right)

	ICP		PCON	
	RTT [ms]	Rate [Mbps]	RTT [ms]	Rate [Mbps]
C1	128.31	4.48	132.08	5.53
C2	107.83	5.16	112.37	6.77
C3	88.28	6.34	92.26	9.32
C4	68.01	7.86	72.19	12.69
C5	48.21	12.20	52.46	20.52
All	78.08	7.21	79.23	10.96

Table 1: Mean RTT and Rate per Consumer

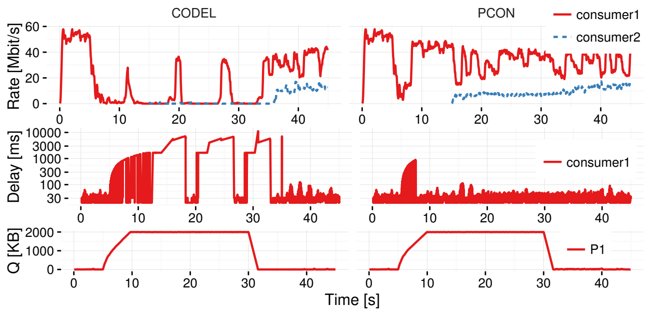


Figure 12: Highly Congested Links

from the literature. Namely, Carofiglio et al.'s work of delay-based congestion control, together with route-labeling, Remote Adaptive Active Queue Management (RAAQM), and their multipath forwarding strategy (CF) [6].

We use a larger topology (Figure 9) with 5 consumer apps, all having a high-speed access link (100 Mbps), but different delays (1ms to 40ms). They start at different times (from second 0 to 4) and retrieve different files under the same prefix. All consumers start with the path C–R1–R2–P1 until the forwarding ratio is adapted (based on congestion marks for PCON; based on pending Interests for ICP). After a discussion with the authors, we choose the following ICP parameters, which work well in this scenario: pMin=0.00001, dropFactor=0.005, beta=0.9. We report 3 main findings (Figure 11):

First, ICP is faster in adjusting the forwarding percentage to the final value. This is because the number of pending Interests is available almost immediately, but it takes some time (here: about 2.5 seconds) for the congestion marks to arrive.

Second, as shown earlier, PCON arrives at a split ratio that results in a rate that is much closer to the full network bandwidth (60 Mbps) than ICP. ICP, on the other hand, tends to choose paths with a lower link delay. This leads to a different load at the router queues: ICP only creates noticable queuing delay on the bottleneck of the shortest path (P4); PCON distributes the queuing delay evenly among all bottlenecks (R1 and R2 aren't shown in Figure 11, because they don't experience any queuing delay). Morever, ICP results in a shorter end-to-end delay at the consumers (see Table 1). The average per-packet RTT of all consumers is very close (78ms vs. 79ms), since PCON sends a higher percentage of traffic to the lower-delay consumers.

Third, we see that both ICP and PCON result in a consumer rate that is indirectly proportional to the difference in RTT. Both schemes are close to *linear RTT fairness*, meaning that the throughput ratio is roughly linear to the RTT ratio. ICP achieves a moderately fairer split: its throughput ratio of C5 to C1 is 2.72 instead of 3.71 for PCON (with a RTT ratio of 2.66 and 2.52). If the RTTs of all consumers are equal, both schemes achieve the same fair (meaning equal) split of the available bandwidth.

A minor difference is the way in which the congestion window is adapted: While PCON uses TCP BIC with an initial slow start phase, the current implementation of ICP uses regular AIMD (like TCP Reno) without a slow start. This difference in the window adaptation mechanism does not affect the other results.

4.5 Highly Congested Links

The next scenario investigates PCON's reaction to fully congested links and confirms the rationale behind signaling the "highly congested state" (**PCON**) over only using 1-Bit congestion marks (**CODEL**). We use the topology in Figure 10, where each link has a one-way delay of 5ms. There are three consumers with different start times: At the beginning, C1 requests traffic from prefix /A (served by all producers). At 5 seconds, C3 requests traffic from prefix /C (served only by P1) with a high fixed rate and doesn't adapt to congestion marks. Thus, C3 quickly overflows the outgoing queue of P1 (see "Q" in Figure 12) and makes the link practically useless for any other traffic. At 15 seconds, C2 starts, requests traffic under prefix /C, and like C1, reacts to congestion marks. At 30 seconds, C3 stops and the link to P1 becomes usable again.

The results (Figure 12) show that 1-bit marking is unable to quickly disable and completely avoid the congested link. With the spaced-out packet marks, R2 has no explicit knowledge about whether the queue at R3 is overflowing or not. Thus, R2 sends a small proportion of the traffic (under prefix /A and /B) to P1. These packets result in a burst of timeouts and quickly reduce the congestion window of both consumers (C2 can't even go much higher than 1). Moreover, even after R2 has reduced forwarding percentage towards P1 (face 257) to 0%, every time another path receives a congestion mark it will increase again (here to 0.5%) and lead to more timeouts.

PCON's signaling of the highly congested state avoids that problem: At about 7 seconds, it completely disables the use of the interface towards P1 and only re-enables it at about 32 seconds, when the queue has drained. Moreover, since this state can be saved *per-face* at R2 (and per-prefix at R1), the traffic of R2 immediately avoids using P1, even though it is under a previously unseen prefix. This wouldn't be possible without the third bit (highCongLocal), which indicates

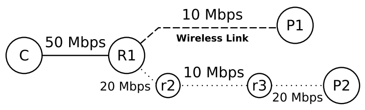


Figure 13: Topo 4.6

that the congested link is directly connected. After the cross-traffic disappears at 30s, both consumers quickly take up and share the newly available bandwidth.

4.6 IP Overlay and Wireless Links

Lastly, we evaluate the performance of PCON against the concept of Hop-by-hop Interest Shaping. In our topology (Figure 13), router R1 is connected to P1 over a wireless link and to P2 over a UDP tunnel. We simulate the wireless link as one that randomly drops packets with a probability of 1%. The routers inside the UDP tunnel (r2 and r3) use a drop-tail FIFO queue with a maximum size of 250 KB (about 250 Data packets). We show the queue size of r3, but don't control any behavior of the underlay routers.

For comparison, we implement a simplified version of HBH Interest Shaping based on Wang et al.'s work [22]. The shaper reads out the fixed link capacity from the network interface and uses a moving average estimation of the Data chunk size to compute the *Interest shaping rate*. Whenever the incoming Interest rate exceeds the shaping rate, the excessive Interests are put into an internal queue (not shown in the Figure) which uses the CoDel logic to signal back congestion via NACKs. We look at 3 different scenarios:

- **Shaper_Ideal:** The shaper somehow knows the available link capacity in the underlay network (10 Mbps) and uses it as parameter.
- **Shaper_Overlay** The shaper at R1 uses its local link bandwidth (20 Mbps) as shaping parameter, which is higher than the available (but unknown) capacity in the IP-underlay network (10 Mbps).
- **PCON:** Our design as described earlier. We detect losses with the link adaptation layer and signal them back via NACKs. NACKs created by the overlay tunnel contain congestion marks (and thus reduce the cwnd), while NACKs from the WiFi link only trigger a retransmission, but no window decrease. The consumer considers a burst of NACKs during one RTT as *one* loss event. Moreover, the router's forwarding ratio adapts to NACKs at most once per CoDel Interval (100 ms).

All three scenarios start out with the optimal split ratio of 0.5. PCON adjusts it based on the received congestion marks and NACKs (which in this case is unnecessary), while the Interest shaper keeps the optimal value.

The results are shown in Figure 14. First, we see that in the ideal case (known link capacity and fixed Data chunk size), Interest shaping achieves its goal of keeping router queues small, reducing Data retrieval delay, and avoiding timeouts. However, when the shaper assumes a bandwidth that is too high (second column), we see high queuing delay and periodic bursts of timeouts ("T/Os" shows the cumulative timeouts over time). When the shaper bandwidth is set too low (not depicted), we see a predictable lower throughput (when the shaper assumes a link capacity of 5 Mbps, the maximum consumer throughput will be 5 Mbps).

PCON results in a higher queuing and consumer delay than the ideal shaping scenario, because it detects congestion based on losses in the underlay network and isn't able to fully drain the underlay router's queues. However, it is able to control the consumer rate and completely avoid timeouts, even without knowing the bottleneck link bandwidth. Moreover, PCON's forwarding strategy converges to the optimal split ratio, even in the case where congestion signals result

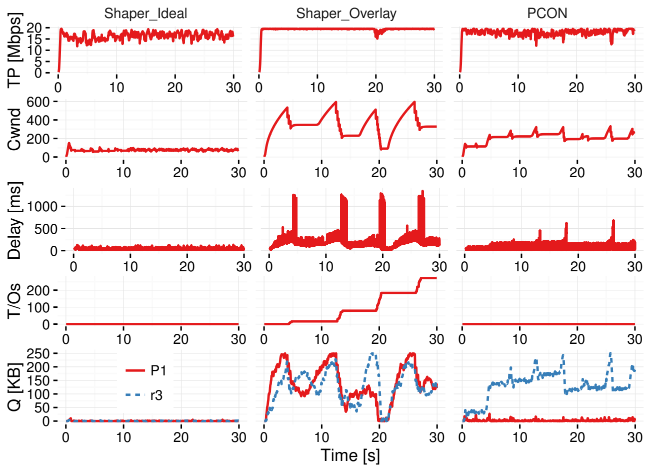


Figure 14: IP Overlay & Wireless Links

from a mix of bursty NACKs and non-bursty CoDel markings.

Our implementation of HBH Interest shaping is arguably simplistic, as it doesn't even try to predict the real bottleneck bandwidth and instead uses the incorrect value of the local link capacity. However, the related work we know of ([5, 22, 19, 10, 27, 26, 17, 11]) doesn't consider the possibility of unknown link bandwidth, and leaves the behavior in this case unspecified. Moreover, the task of explicitly estimating the bandwidth of an IP tunnel or wireless link is non-trivial. We would like to see more work in this area, but as for now, we consider our implementation of HBH Interest shaping as representative of the state of the art.

5. RELATED WORK

Over the last couple years there have been a number of proposals for congestion control in NDN. Van Jacobson's original paper states that "flow balance is maintained at each hop" [9], but leaves the details of congestion control to future work. Other researchers soon recognized that NDN's caching and multipath forwarding makes traditional congestion detection by RTT measurement or packet losses inadequate, and proposed to use explicit congestion notification in the form of a truncated Data packet [16] or a NACK [25].

Carofiglio et al. [6] proposed a delay-based congestion control scheme, which triggers an Interest window decrease based on a sample of past RTT measurements. To overcome the problem of mixing RTT measurements from different packet sources, they introduced the concept of *route-labels*, tags in each Data packet that indicate which node the packet has originated from and which path it took. However, the practicality and scalability of route-labels has been questioned [13], as the number of potential routes increases exponentially with the number of nodes in the network.

Moreover, even with route-labels, a consumer still does not know from where the next Data packet would come, making it difficult to set a correct timeout value. As an alternative, [20, 4] have proposed mechanisms to predict the sources of the next Data packet based on markings of previously returned Data packets. However, the reliability of these schemes in real networks (where the content of each cache might change rapidly) remains an open question.

Another approach to NDN congestion control is *Hop-by-Hop (HBH) Interest Shaping* [5, 22, 19, 10, 27, 26, 17, 11], based on the assumption that a router can control the amount of returning data on its links by controlling the Interest sending rate in the opposite direction. As shown in Section 4.6, these schemes require explicit knowledge of the available link bandwidth, which limits their effectiveness for wireless networks and IP tunnels.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we adopt CoDel's congestion estimation and use NDN's stateful hop-by-hop forwarding plane to design PCON, a congestion control scheme that works in diverse scenarios (including WiFi links and IP-Overlays), without making strong assumptions about the network. Architectural differences between IP and NDN communication prevent a naive, direct adoption of existing TCP/IP based mechanisms. PCON considers these differences and thus doesn't rely on end consumer timeouts, RTT measurements, or out-of-order packet arrivals. Different from most related work, we remove the following assumptions in our design: 1) We do not require in-network routers to know the capacity of their links, 2) we do not assume that Data packet sizes are known or predictable, 3) we do not rely on in-network flow differentiation, 4) we do not determine the exact origin (cache) of a Data packet (either via route-labels [6] or previous Data packets [20, 4]), and 5) we do not assume any specific caching policy at routers.

Instead, PCON uses explicit hop-by-hop control and integrates an AQM scheme into the congestion detection. This explicit signaling allows for a novel forwarding adaptation based on congestion marks, which improves throughput over PI-based alternatives.

We identify certain topics for future work: 1) A definition of fairness in a data dissemination network, that is, whether fairness should be measured on per-consumer or per-content basis. 2) An effective mitigation against unresponsive consumers, with the goal of minimizing their impact on the rest of the traffic. 3) Determining the dynamics of congestion signals that are used to both adjust the forwarding ratio of multiple routers on the path, and also to adjust the sending rates of multiple consumers at the end.

Acknowledgements

This work is partially supported by the National Science Foundation under Grant No. 1345142, 1345318, and 1551057. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors would like to thank Luca Muscariello and Giovanna Carofiglio for helpful feedback on the evaluation and Alex Afanasyev for providing support regarding ndnSIM.

7. REFERENCES

[1] Ndn protocol design principles, accessed apr. 2016. http://named-data.net/project/ndn-design-principles/.

[2] Ndn testbed. http://named-data.net/ndn-testbed/.

[3] E. Blanton, K. Fall, and M. Allman. A conservative sack-based loss recovery algorithm for tcp. *RFC 3517*, 2003.

[4] S. Braun, M. Monti, M. Sifalakis, and C. Tschudin. An empirical study of receiver-based aimd flow-control for ccn. In *IEEE ICCCN*, 2013.

[5] G. Carofiglio, M. Gallo, and L. Muscariello. Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks. In *ACM ICN workshop*, 2012.

[6] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, and S. Wang. Optimal multipath congestion control and request forwarding in information-centric networks. In *ICNP*, 2013.

[7] A. Detti, C. Pisa, and N. Blefari Melazzi. Modeling multipath forwarding strategies in information centric networks. In *IEEE INFOCOM WKSHPS*, 2015.

[8] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM CCR*, 1988.

[9] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *ACM CoNEXT*, 2009.

[10] K. Lei, C. Hou, L. Li, and K. Xu. A rcp-based congestion control protocol in named data networking. In *CyberC*, 2015.

[11] C. Li, T. Huang, R. Xie, H. Zhang, J. Liu, and Y. Liu. A novel multi-path traffic control mech. in ndn. In *IEEE ICT*, 2015.

[12] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang. ndnSIM 2.0: A new version of the NDN simulator for NS-3. Technical Report NDN-0028, NDN, January 2015.

[13] D. Nguyen, M. Fukushima, K. Sugiyama, and A. Tagami. Efficient multipath forwarding and congestion control without route-labeling in ccn. In *IEEE ICCW*, 2015.

[14] K. Nichols and V. Jacobson. Controlling queue delay. *ACM Communications*, 2012.

[15] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar. Controlled delay active queue management: draft-ietf-aqm-codel-03. *RFC draft*, 2016.

[16] S. Oueslati, J. Roberts, and N. Sbihi. Flow-aware traffic control for a content-centric network. In *IEEE INFOCOM*, 2012.

[17] H. Park, H. Jang, and T. Kwon. Popularity-based congestion control in named data networking. In *IEEE ICUFN*, 2014.

[18] V. Paxson, M. Allman, J. Chu, and M. Sargent. Computing tcp's retransmission timer. Technical report, RFC 2988, 2000.

[19] N. Rozhnova and S. Fdida. An extended hop-by-hop interest shaping mechanism for ccn. In *IEEE GLOBECOM*, 2014.

[20] L. Saino, C. Cocora, and G. Pavlou. Cctcp: A scalable receiver-driven congestion control protocol for ccn. In *IEEE ICC*, 2013.

[21] S. Vusirikala, S. Mastorakis, A. Afanasyev, and L. Zhang. A best effort link layer reliability scheme. Technical report, NDN TR41, 2016.

[22] Y. Wang, N. Rozhnova, A. Narayanan, D. Oran, and I. Rhee. An improved hop-by-hop interest shaper for congestion control in named data networking. *ACM SIGCOMM CCR*, 2013.

[23] L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control (bic) for fast long-distance networks. In *IEEE INFOCOM*, 2004.

[24] E. Yeh, T. Ho, Y. Cui, M. Burd, R. Liu, and D. Leong. Vip: A framework for joint dynamic forwarding and caching in named data networks. In *ACM ICN*, 2014.

[25] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang. A case for stateful forwarding plane. *Elsevier Computer Communications*, 2013.

[26] F. Zhang, Y. Zhang, A. Reznik, H. Liu, C. Qian, and C. Xu. A transport protocol for content-centric networking with explicit congestion control. In *IEEE ICCCN*, 2014.

[27] J. Zhou, Q. Wu, Z. Li, M. A. Kaafar, and G. Xie. A proactive transport mechanism with explicit congestion notification for ndn. In *IEEE ICC*, 2015.