Effective NDN Congestion Control Based on Queue Size Feedback

Sichen Song
University of California, Los Angeles
Los Angeles, CA, USA
songsichen@cs.ucla.edu

Lixia Zhang
University of California, Los Angeles
Los Angeles, CA, USA
lixia@cs.ucla.edu

ABSTRACT

Named data networking (NDN) can improve the consumer data retrieval throughput with its built-in multicast data delivery, in-network caching, and ability to support multi-path forwarding. However, their realization brings challenges. In this work, we first examine how multi-path forwarding and in-network caching can interfere with consumer measurements for congestion control. Based on the results, we propose a congestion control solution, NDN-QSF, that can work effectively in the presence of in-network caching. In NDN-QSF, forwarders estimate upstream bandwidth and use queue size as congestion feedback to inform downstream routers to limit interest transmission rates. We further adapt and extend NDN-QSF to enable routers to make informed multi-path forwarding decisions. We evaluated NDN-QSF through simulation experimentation and our results show that NDN-QSF can effectively control congestion by using queue size as congestion feedback and improve network throughput with multi-path forwarding.

CCS CONCEPTS

• **Networks** → **Intermediate nodes**;

KEYWORDS

Congestion Control, Named Data Networking, Multi-Path Forwarding

ACM Reference Format:
Sichen Song and Lixia Zhang. 2022. Effective NDN Congestion Control Based on Queue Size Feedback. In *9th ACM Conference on Information-Centric Networking (ICN '22), September 19–21, 2022, Osaka, Japan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3517212.3558088

1 INTRODUCTION

Data multicast, in-network caching, and multi-path forwarding are desired features that improve the performance of data object retrieval in computer networks. Data multicast and in-network caching reduce network traffic when multiple consumers request the same data, and multi-path forwarding explores and utilizes all available network resources to further improve throughput.

The above three desired features for data retrieval are enabled by networks running the Named Data Network (NDN) [14] protocol stack. NDN forwarders are stateful with built in data caches, and data is retrieved by names. Data consumers fetch content by sending interest packets carrying content names, which are forwarder towards data producers. If an interest hits a forwarder cache, it brings the matching data packet back. If multiple interests carry the same data name, they are aggregated. Data packets are delivered to consumers following the reverse path of the interest packets, creating a feedback loop. Therefore NDN automatically supports data multicast (through interest aggregation) and in-network caching. Its interest-data exchange feedback loop enables forwarders to detect and eliminate packet loops and make fine-grain measurement of data retrieval performance. Based on the measurement, forwarders can send interests to best paths and to multiple paths dynamically.

While caching, multicast data delivery, and multi-path forwarding reduce the contention of network resources, congestion is still possible. The existing TCP/IP congestion control algorithms operate on end-to-end connections following a single path. NDN interest packets can be satisfied by in-network caches and forwarded dynamically along different paths towards requested data locations. When data transmission no longer goes along a single path, the concepts such as "queuing delay" and "bandwidth-delay product" of a single path used in TCP/IP congestion control, are no longer applicable as interests can be satisfied at different nodes. Thus, a congestion control algorithm for NDN must consider the new features. Consequently, end-to-end congestion control widely used in IP is not a well fit for NDN, and a congestion control algorithm for NDN must consider its new features.

Congestion control and multi-path forwarding remain active research areas in NDN. This work aims to design a multi-path forwarding with congestion control for NDN forwarders that 1) utilizes multiple forwarding paths to maximize network throughput, 2) controls congestion, and has a low queuing delay. Both goals should be achieved in the presence of in-network caching and multi-path forwarding. In addition, we hope to explore whether these goals can be achieved *without* imitating flows in IP networks to establish an end-to-end single-path flow concept and adapt existing solutions in IP.

We take the first step to tackle the above challenges; thus, the results from our preliminary investigation are limited in scope. Our simulations focus on the scenarios with one or more consumers fetching the same data object, which can be large in size and segmented into multiple NDN data packets. Our evaluation metric measures consumer throughput and network congestion levels.

This work investigates the effect of multi-path forwarding and in-network caching on consumer congestion control measurements. It then presents a congestion control design for NDN, NDN-QSF, that uses queue size as congestion feedback and works in the presence of in-network caching. NDN-QSF is lastly extended to work under multi-path forwarding and make multi-path forwarding decisions.

ICN '22, September 19–21, 2022, Osaka, Japan

ACM ISBN 978-1-4503-9257-0/22/09.
https://doi.org/10.1145/3517212.3558088

Our main contributions can be highlighted as the follows.

- We use simulation study to investigate the effect of in-network caching and multi-path forwarding on end measurements and observe that these two features make it difficult for end consumer to accurately detect network congestion.
- We design a congestion control algorithm, NDN-QSF, that uses upstream queue size as congestion feedback and controls congestion hop by hop to effectively control congestion in NDN.
- To have multi-path forwarding decisions adapting to network loads, we extend NDN-QSF to make multi-path forwarding decisions based on upstream congestion feedback.
- We implement NDN-QSF as a forwarding strategy of NFD and evaluates it with ndnSIM simulation.

The rest of this paper is organized as the following: we discuss the related works in §2 and the effect of NDN features on consumer congestion control measurements in §3. We present the design of NDN-QSF in §4 . The NDN-QSF is extended to work with multi-path forwarding in §5 and evaluated in §6. We discuss the results in §7. Lastly we draw our conclusion in §8.

2 RELATED WORK

A number of works investigate NDN congestion control and multi-path forwarding. They share the same goal of building a effective feed back loop and differ in their network load measurements, congestion feedback, and how and where the reaction takes place.

Some works are based on the idea of distinguishing end-to-end single path flows and adapting existing solutions in IP networks. For example, In [4], Carofiglio *et al.* formulate the problem of dynamic multi-path forwarding and congestion control as a network utility maximization problem. Its solution shows that network utility can be maximized by having forwards make multi-path forwarding decisions and consumers control congestion using congestion window. Local measurements are enough for both forwarders and consumers. However, forwarding paths are marked on Data packets as feedback to enable consumers to measure queuing delay. Multi-path forwarding decisions are made based on balancing pending interest table (PIT) entries towards each upstream.

Another example is the multi-path aware rate-based congestion control (MIRCC)
[10], Mahdian *et al.* designed a congestion control and multi-path forwarding algorithm. The design is based on the idea that the bottleneck forwarder detects congestion and assigns a per-flow sending rate to each consumer, and the consumers send out interest of each flow following their assigned rate. Path marking on data packets is used to allow consumers to distinguish different flows. Consumers make multi-path forwarding decisions by including path hints in Interest packets that direct traffic to a specific forwarding path. The work points out that maximized throughput and max-min fairness may not be achieved together in some networks that have multi-path forwarding.

Other works do not rely on distinguishing end-to-end single-path flows. In PCON [11], Klaus*et al.* proposed a framework for congestion control and multi-path forwarding based on congestion marking. PCON forwarders monitor queuing delays on outgoing links to detect congestion and apply congestion marks on data packets traversing through the congested link. Congestion marks are used at consumers for congestion control and downstream forwarders for multi-path forwarding. Consumers run window-based congestion control and control the window size using rules similar to TCP AIMD [6] and Cubic [8]. Congestion marks on data packets will trigger a multiplicative window decrease. Multi-path forwarding is based on the idea of "balancing the frequency of congestion mark from all upstream." A "split ratio" is used to control the split of traffic among upstream. Whenever ECN is received from one upstream, the share of traffic to this upstream is reduced by a fixed amount. The split ratio stabilizes when upstream receives congestion marks at the same frequency.

The VIP framework [13] that jointly make caching and forwarding decisions to optimize network throughput also does not rely on distinguishing end-to-end flows. VIP uses queue size information as feedback to neighbors to detect congestion and make multi-path forwarding decisions. However, to learn the resource's popularity and optimize caching decisions, a virtual control plane is used to keep track of the queue size of each object without the effect of interest aggregation. The actual NDN forwarders use queue size from the virtual plane to route interests for different objects to different paths and evict cached objects. The work does not provide a method to control congestion. Interests for segments of the same object follow the same forwarding path.

These existing works inspires our design of NDN congestion control and multi-path forwarding. Similar to PCON and VIP, our design, NDN-QSF, sends queue size related information towards the consumer as congestion feedback. NDN-QSF also shares the idea of controlling rates directly and "using smart forwarders with dumb consumers" with MIRCC. Lastly, NDN-QSF is hop by hop similar to VIP and does not distinguishing end-to-end flows. The hop-by-hop design avoids the assumption that data can be satisfied by a relatively fixed endpoint in the network.

3 NDN AFFECTS CONSUMERS' LOCAL CONGESTION MEASUREMENTS

Many congestion control algorithms in TCP/IP, such as BBR [3] and Copa [2], rely on round trip time (RTT) and data arrival rate measurements of an end-to-end single-path connection to detect network congestion. Dynamic multi-path forwarding and in-network caching in NDN invalidate the assumption of end-to-end single-path connection and make it difficult to detect network congestion using local RTT and rate measurements.

3.1 Effect of Dynamic Multi-path Forwarding on Consumer Measurements

Consider the simple network topology in fig.1, where forwarder 1 makes multi-path forwarding decisions to split traffic between two paths with different delays. The consumer requests data at a constant 50mb/s rate. At time t, forwarder 1 will forward $r\%$ of traffic to replica 1 and the remaining to replica 2 where $r = \frac{1}{2} - 0.2 \times sin(t * 4) \times 100\%$. r changes over time to emulate forwarders' dynamic multi-path forwarding adjustments.

Our simulation result of the above scenario is illustrated in fig. 2. As shown in the bottom left figure, RTT samples from different forwarding paths are mixed, making the raw RTT samples hard

to interpret. Without distinguishing the two forwarding paths, a consumer can use an averaged RTT, which is affected by multi-path forwarding. The average RTT increases as more packets are directed to the path with a longer delay. The consumer throughput is also affected by multi-path forwarding adjustments. When the portion of traffic directed to the long delay path increases, the consumer measures a throughput below its interest sending rate.

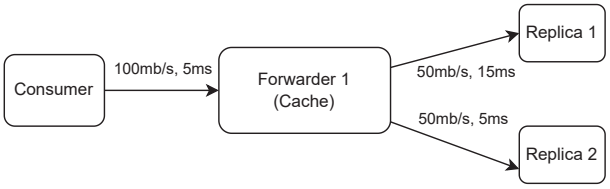

Figure 1: Network topology to simulate the effect of multi-path forwarding on consumer measurements

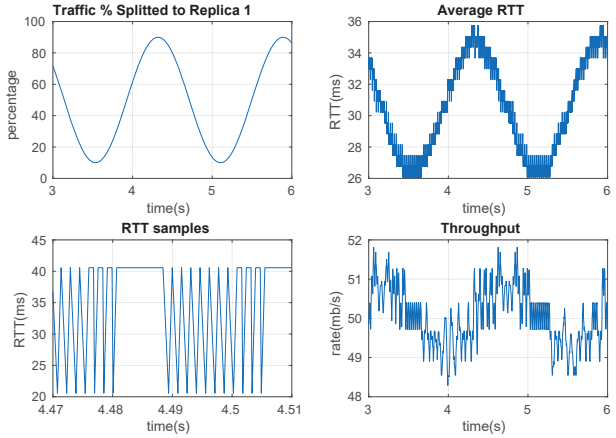

Figure 2: Effect of multi-path forwarding on Consumer Measurements

3.2 Effect of In-Network Caching on Consumer Measurements

Let us consider the network topology illustrated in fig.3. Both consumers request the segments of a large object with a fixed interest sending rate in sequential order. The size of each data packet is around 4500 bytes. Consumer 1 starts first to request data at 16mb/s. Consumer 2 starts half a second later and requests data at 20mb/s. Forwarder 1, which both consumers share, has a large cache. Consumer 2 will initially be satisfied by the cache and eventually drain it and be satisfied by the producer as it is requesting data faster than consumer 1.

The simulation result is shown in fig. 4. The top sub-figure shows the consumers' highest requested segment number at a given time. At around 3.55 seconds, the two lines intersect, indicating that consumer 2 has just passed consumer 1 in the progress of retrieving the object. The round trip time (RTT) and data arrival rate measured by consumer 2 in catching up with consumer 1 have three different stages. Initially, before 3.46 seconds, consumer 2 is satisfied by the cache. Thus, it measures the RTT to the cache, and its data receiving

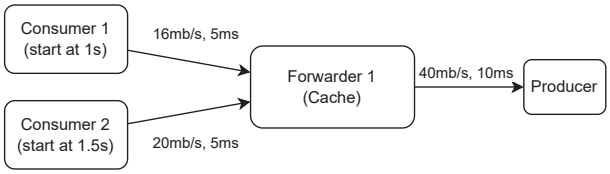

Figure 3: Network topology to simulate the effect of caching on consumer measurements

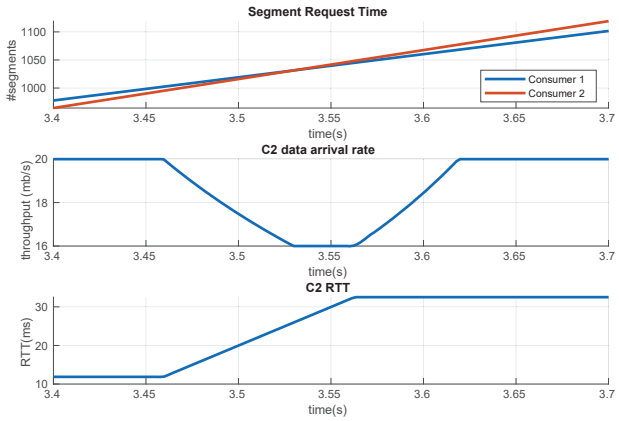

Figure 4: Effect of Caching on Consumer Measurements

rate is the same as the interest sending rate. Eventually, after 3.61 seconds, interests from consumer 2 are forwarded to the producer. Consumer 2 measures RTT to the producer while its data arrival rate is the same as interest sending rate. However, between these two stages, there is a "middle stage" where consumer 2 receives data at 16mb/s, which equals to the data request rate of consumer 1. The RTT consumer 2 measures gradually increase from around 10ms to around 30ms.

The effect of interest aggregation is closely related to the rate and delay consumer 2 measured in the "middle stage". By the time interest of consumer 2 reaches the forwarder, the corresponding data is not in the cache, but consumer 1's interest for the same segment is pending at the forwarder. Consequently, the interest from consumer 2 will be aggregated, and the data requested by consumer 1's interest will be multicasted to both consumers. The RTT measurements at consumer 2 in this stage depend on how long the interest for the same segment from consumer 1 has been pending by the time consumer 2's interest reaches the forwarder, which is related to the progress difference between the consumers. As consumer 2 has a faster interest sending rate, its progress in retrieving the large object will gradually catch up with consumer 1's progress. Meanwhile, RTT measured by consumer 2 gradually increases. Also, the data arrival rate measured by consumer 2 in this stage is the data arrival rate of consumer 1 as consumer 2 is receiving multicasted data requested by consumer 1.

3.3 False Detection of Congestion when Misinterpreting Measurements

If one adopts the same interpretation used in IP's end-to-end congestion control algorithms, one will misinterpret measurements under

multi-path forwarding and in-network caching as network congestion. When more traffic is directed to a path with longer RTT under multi-path forwarding or in the"middle stage" under in-network caching, consumer measurements can be misinterpreted as a congestion event: the RTT gradually increases as if queuing delay is growing, and the interest sending rate is faster than the data receiving rate, suggesting that traffic is being buffered. However, reacting to this false detection of congestion results in under-utilization.

The above simple illustration of the multi-path forwarding and in-network caching's effect on the end consumer measurement shows that NDN consumers cannot reliably detect congestion based on their local RTT and data arrival rate measurements. Instead, we need to use the information the forwarders can provide to infer congestion. Inspired by the PCON [11] design for NDN congestion control, we decided to use forwarders' local queuing information as the congestion signal downstream.

4 EFFECTIVE CONGESTION CONTROL IN THE PRESENCE OF CACHING

With the idea of using queuing information provided by forwarders as congestion feedback, we design a congestion control solution for NDN forwarders, NDN-QSF, whose forwarding is illustrated in fig. 5. The proposed forwarding process is based on the existing NDN forwarding pipeline with the addition of per-name prefix interest queuing, rate limiting, measurements, and congestion feedback modules. In the design, interest and data queue size at each forwarder is sent to the downstream as congestion feedback. Each forwarder keeps a per-upstream estimation of the available bandwidth towards the producer. All interest packets in the same name prefix received from downstream are put into an interest queue. A rate limit for a name prefix towards an upstream is calculated from the upstream's estimated bandwidth and queue size of the specific name prefix. When the upstream congestion feedback shows a large queue, one should lower the rate limit to drain the upstream queue. The forwarding strategy then forwards interests in the interest queue towards the upstream while ensuring compliance with the rate limit.

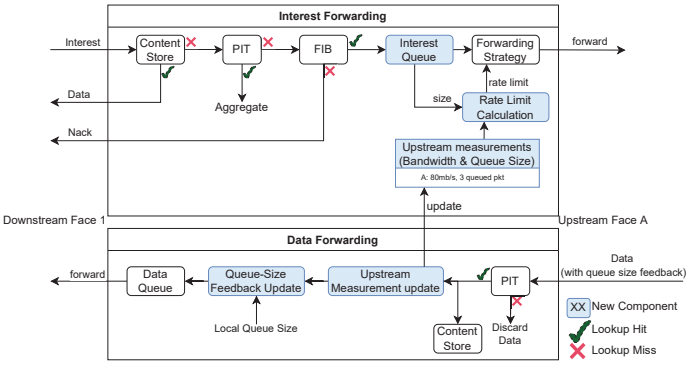

Figure 5: Forwarding Pipeline of Congestion Control Design

4.1 Queues and Queue Size Feedback

As motivated in §3, queue size feedback is passed downstream as congestion signal. The feedback is transmitted by piggybacking on every data packet. Queue size feedback is on a per name prefix bases. Its value is based on both the interest queue and data queue size of a specific name prefix. As mentioned above, each forwarder has an interest input queue for each name prefix (FIB entry). The interest queue buffers interest packets to comply with rate limits for the upstream. Each forwarder also has a data packet queue for each downstream link. The data queue buffers data packets to be sent towards each downstream.

The value of the queue size feedback is determined by interest queue and data queue sizes at the current forwarder, as well as whether the interest packet corresponding to the data packet hit the cache or was aggregated. When an interest did not hit the cache or was aggregated, its corresponding data packet will have the congestion feedback value as the larger size between the interest queue and the data queue for the downstream the data packet is forwarded. Suppose cached data was returned, or a data is multicasted to a downstream in response to an aggregated interest, the queue size feedback marked on the data packet will be the data queue size for the downstream that the data will be forwarded.

As the interest queue and data queue have different dequeuing speeds, taking the larger value does not have a special meaning. It provides a means to downstream forwarders for reacting to the larger queue size at this hop.

As queue size feedback is piggybacked on every data packet, the forwarder keeps a minimum interest sending rate to keep an up-to-date view of the upstream queue size.

4.2 Estimating Upstream Bandwidth

Each forwarder estimates the bandwidth through each upstream towards the producer. This bandwidth is estimated using each upstream's data arrival rate and queue size feedback. The forwarder measures upstream data arrival rate using a sliding window that keeps the arrival time of recent data packets, as shown in fig. 6. By dividing the number of inter-data arrival time gaps inside the measurement window by the period of data arrival in the measurement window, the data arrival rate from each upstream can be estimated. Forwarders also keep a smoothed upstream queue size by applying a windowed average to the queue size feedback from each upstream.

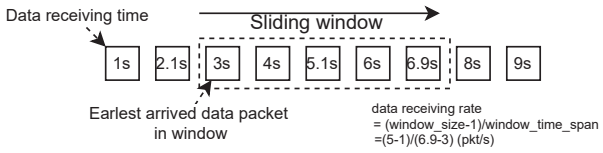

Figure 6: Data arrival rate measurement based on sliding window

With the queue size and data arrival rate information of each upstream, the bandwidth of an upstream is estimated as the data arrival rate of the upstream when its queue has built up. Sending interests to an upstream faster than the estimated bandwidth further builds up the upstream queue. Based on this idea, the following two rules are used for bandwidth estimation of the upstream:

(1) if queue size > QUEUE_THRESHOLD, set the estimated bandwidth to the current data arrival rate.
(2) if data arrival rate > estimated bandwidth, set bandwidth to the current data arrival rate.

In the first rule, the QUEUE_THRESHOLD parameter is used as a threshold on the recorded upstream queue size to determine if the upstream is congested. We find that a value of 4 works well in our simulations. The second rule makes corrections when an upstream with no congestion has an underestimated bandwidth.

4.3 Rate-Limit Calculation

A forwarder makes periodic adjustments to its rate limits towards its upstream. The ROUND_LENGTH parameter is used to define the period of adjustment. A small random jittering is added to the adjustment period to avoid the synchronization of different forwarders. In our experiments, we use network RTT as the ROUND_LENGTH. The algorithm for rate limit adjustments is illustrated in alg. 1.

```
initialization;
for each adjustment round do
    u ←findBestUpstreamFromFib(namePrefix);
    /* congestion control                          */
    if queueSize(u) > QUEUE_THRESHOLD × 2 then
        rate_limit[u] ← bandwidth[u] × MD_FACTOR;
    else
        rate_limit[u] ← bandwidth[u];
    end
    rate_limit[u] ← max(rate_limit[u] , MIN_RATE);
    /* rate probing                                */
    if queueSize(u) < QUEUE_THRESHOLD then
        rate_limit[u] ← rate_limit[u] ×1.05
    end
    sleep(ROUND_LENGTH+random());
end
```

Algorithm 1: Rate-Limit Adjustments

4.3.1 Congestion Control. Congestion is controlled hop-by-hop by adjustment of the rate limits. Forwarders learn the queue size of each upstream from congestion feedback and check if the queue size is above a threshold. If so, the rate limit towards the upstream will be set to a portion of its estimated bandwidth to drain out the upstream queue.

The QUEUE_THRESHOLD parameter represents the expected amount of steady-state queue size. Its selection trades off between bandwidth utilization and queuing delay. A larger value enables persistent queuing to improve throughput at the cost of increased queuing delay. This value is shared in the entire network. We find a value of 5 works well in most of our evaluations.

The MD_FACTOR parameter determines how much bandwidth the forwarder gives up to drain the upstream queue. A Larger value allows faster reaction at the cost of potential overreaction that harms the throughput. As each round, the upstream queue is drained by roughly $bandwidth \times round\_length \times (1 - MD\_FACTOR)$, the MD_FACTOR is set based on queue threshold and adjustment round length. A larger value works better for long adjustment round and low queuing environments.

When the congestion control is triggered, the rate limit of interests towards the congested upstream is reduced. Thus interests from downstream may start to queue up at this forwarder, causing downstream to measure a large queue. The downstream will consequently start to control congestion. This process can propagate to consumers, eventually reducing the interest sending rate and avoiding congestion.

4.3.2 Rate Probing. Rate probing is introduced to allow forwarders to correct their underestimated upstream bandwidth by sending to an upstream with a rate higher than its estimated bandwidth. It is done by increasing the rate limit towards an upstream if the queue size at the upstream is below a threshold. This requirement rules out the case where the upstream already has a queue built up. When upstream is already congested, the current data arrival rate should be an up-to-date estimation of its bandwidth.

Once a forwarder starts rate probing by increasing the rate limit, its input queue will be drained out quickly, and its downstream is likely to measure a low upstream queuing and consequently also starts probing and increases the rate limit. This upstream to downstream propagation of rate probing continues towards the consumer, in which consumer should conclude it can increase interest sending speed. The increased sending speed at consumers results in increased traffic along the path being probed, thus increasing bandwidth estimations along the path.

4.4 Consumer Congestion Control

As the forwarders run hop-by-hop buffering and congestion control, the consumer congestion control algorithm can control interest transmission based on the queue size of the first-hop forwarder to keep a persistent queue at the first hop forwarder. As this work focuses on the design of forwarders, we use a naive consumer that keeps a large queue at the first hop forwarder in our evaluation.

4.5 Providing Fairness with Fair-queueing

NDN breaks the IP's concept of end to end flows and thus as discussed in PCON [11], the fairness in an NDN network is yet to be well defined. We limited the scope of this work not to dive deep into fairness in NDN traffic. However, we argue that fair queuing can be a useful primitive to use together with NDN-QSF to provide fairness. We validate this claim with the simulation in §6.6.

5 MULTI-PATH FORWARDING AT FORWARDERS

Similar to the feature of in-network caching, dynamic multi-path forwarding in NDN makes it infeasible for consumers to detect congestion using local measurements accurately. We believe that NDN-QSF that uses "upstream queue size feedback" is effective under multi-path forwarding. In addition, we find that the bandwidth estimated for the congestion control algorithm can be used to guide traffic splitting among multiple paths. Thus, we extend our congestion control algorithm to make multi-path forwarding decisions to improve network resource utilization and consumer throughput.

The updated design follows the same forwarding pipeline illustrated in fig. 5 with modified rate-limit calculation and forwarding strategy. With traffic forwarded to multiple paths, the bandwidth of every upstream is estimated, and a rate-limit of every upstream is calculated to split interest packets towards multiple paths while controlling congestion.

5.1 Rate-Limit Calcuation

```
initialization;
for each adjustment round do
    /* congestion control                                    */
    for each upstream u do
        if queueSize(u) > QUEUE_THRESHOLD × 2 then
            rate_limit[u] ← bandwidth[u] × MD_FACTOR;
        else
            rate_limit[u] ← bandwidth[u];
        end
        rate_limit[u] ← max(rate_limit[u] , MIN_RATE);
    end
    /* rate probing                                          */
    if queueSize(upstream2probe) < QUEUE_THRESHOLD
     then
        rate_limit[u] ← rate_limit[u] ×1.05
    end
    if queueSize(upstream2probe) ≥ QUEUE_THRESHOLD
     then
        upstream2probe ← selectUpstream2Probe();
    end
    /* Avoiding Unnecessary Contentions                      */
    if queueSize(upstream2probe) < QUEUE_THRESHOLD
     && input_queue_size ≤ QUEUE_THRESHOLD then
        for each upstream u with
         queueSize(u)>QUEUE_THRESHOLD do
            rate_limit[u] ← rate_limit[u] − STEP ;
            rate_limit[upstream2probe] ←
             rate_limit[upstream2probe] + STEP
        end
    end
    sleep(ROUND_LENGTH+random());
end
```

Algorithm 2: Rate-Limit Adjustments with Multi-Path Forwarding

The rate-limit calculation we designed previously for in-network caching is modified slightly. Congestion control is now applied on every upstream. On the other hand, rate probing is restricted to be applied to one upstream in an adjustment round. That is because we want to avoid adjusting the interest transmission rate to multiple upstreams when adjusting the interest transmission to one upstream can satisfy all downstream traffic. A forwarder probes an upstream by increasing its rate limit while keeping the rate limit to all other upstream. This rate limit increase is kept until the probed upstream has a queue size above a threshold. When the upstream being probed has a queue built up, another upstream that does not have a large queue will be selected to be probed in the following adjustment rounds.

5.1.1 Avoiding Unnecessary Contention. Maximizing consumer throughput in an NDN network with a single consumer and multiple forwarding paths is a maximum flow problem. A difficult case of this problem is illustrated in fig. 7. In this network, both *Node 1* and *Node 2* need to make multi-path forwarding decisions to maximize network throughput. The maximum network throughput is 120mb/s, and the only way to achieve it is to have *Node 1* equally split traffic to all its three upstreams while having *Node 2* direct all its traffic to *Node 5*. That's because the *Node 1* to *Node 2* link can at most forward data at 40mb/s. Thus *Node 2* sending any traffic to *Node 3* or *Node 4* results in underutilizing the *Node 2-Node 5* link.

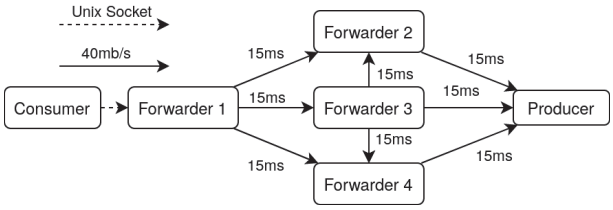

Figure 7: Scenario where unnecessary contention reduces bandwidth utilization

However, the multi-path forwarding algorithm described so far does not have a mechanism to stop *Node 2* from sending to *Node 3* and *Node 4*. Instead, if *Node 2* probes its link to *Node 3* in the beginning, it will keep sending to Node3 based on the bandwidth it probed while not fully utilizing its path through *Node 5* link.

To address this issue, a new rule, which is inspired by the backward edges in the Ford Fulkerson algorithm [7], is added: when the input queue is small, which indicates downstream traffic is below the total upstream bandwidth, reduce the rate limit towards congested upstream while increasing the limits to uncongested ones.

Intuitively, this new rule is trying to avoid unnecessary contention: when downstream interest sending rate is below the total upstream bandwidth, traffic should be prioritized to uncongested upstream instead of creating additional contention on the congested ones.

Using the same example, if *Node 2* is sending to *Node 3*, as *Node 1* will also direct traffic to *Node 3*, the *Node 3-Node 5* link will be congested and consequently *Node 3* will start build up input queue and push back to *Node 2*. At this time, *Node 2* knows its upstream to *Node 3* has large queue while the link to *Node 5* does not. Thus, *Node 2* will reduce its traffic through *Node 3* and increase its traffic through *Node 5*. Consequently *Node 1* can send at higher rate to *Node 3* and the network moves towards an increased throughput.

5.2 Forwarding Strategy

The forwarding strategy is in charge of splitting interests from the input interest queue to different upstream forwarders while complying with the rate-limit of each upstream. We implemented the forwarding strategy by using the rate limit of each upstream as the rate to dequeue and forward from the incoming interest queue.

6 EVALUATION

The ndnSim [1] simulator is used to evaluate NDN-QSF. A variety of network environments are simulated to thoroughly evaluate throughput, delay, and stability. Queuing delay in the simulations are calculated by dividing interest queue size by sending rate of interest packets. Data packets have a size of around 4500 bytes. To better illustrate steady state behavior, y-axis in the figures are clipped and some large queuing delay during the system start up is not captured.

6.1 Congestion Control in the Presence of Caching

NDN-QSF is first evaluated in the presence of in-network caching with the simulation scenario illustrated in fig.8. The three consumers start at different times and request the segments of a large object in sequential order. A shared bottleneck is on the path between each consumer and the producer. When in-network caching and data-multicast are effectively utilized, each consumer can retrieve data at the bottleneck bandwidth of 60mb/s. [1] The simulation result is illustrated in fig.9.

Consumer 1 started first and gradually increased its throughput. Its data is admitted to the in-network caches. Consumer 2 and consumer 3 started later and achieved 70mb/s throughput initially when being satisfied by cache. They can catch up with consumer 1 utilizing the cache and eventually make the same progress as consumer 1. The system converges to a steady-state in 2 seconds. In steady-state, every consumer achieves 60mb/s optimal throughput, indicating that the data multicast and caching are effectively used. In the steady-state, the queuing delay at each hop is controlled to around 3ms, which is much lower than the round trip time of a single hop. Overall, the simulation shows that congestion is effectively controlled while caching and data-multicast are utilized with our design.

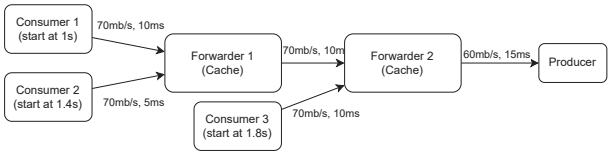

Figure 8: Scenario 1: Shared caches

6.2 Congestion Control in the Presence of Multi-Path Forwarding

We then evaluate the effectiveness of multi-path forwarding decisions made by NDN-QSF. We use a scenario where a single consumer fetches data through multiple forwarding paths. As shown in fig. 10, two forwarders make multi-path forwarding decisions to retrieve the data from three replicas. Note that the path to replica 2 and replica 3 have very different RTTs and bottleneck bandwidth. We intend to use the differences to represent a heterogeneous network environment. The simulation result is shown in fig. 11. The result shows that the algorithm converges in around 12 seconds. After the algorithm converges, the traffic is split at each forwarder according to upstream bandwidth. The 150mb/s multi-path bandwidth is fully utilized in steady-state, and a small queue is persistently kept at each hop.

[1]In the case where one consumer has a local bottleneck not shared with others (for example, in a modified fig. 8 where bandwidth between consumer 1 and forwarder 1 is 30mb/s), the consumer will fetch at a slower speed compared to others. During the data retrieval, the progress differences between the slower consumer and others increase over time. Eventually, the cache may not benefit the slower consumer due to large progress differences. This challenge is out of the scope of this work but is investigated in [5] and [9].

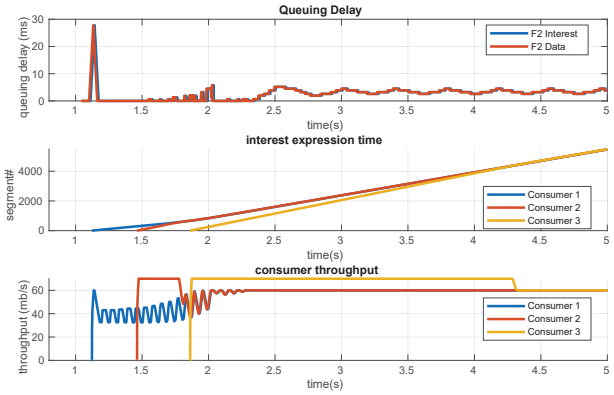

Figure 9: Scenario 1 simulation results

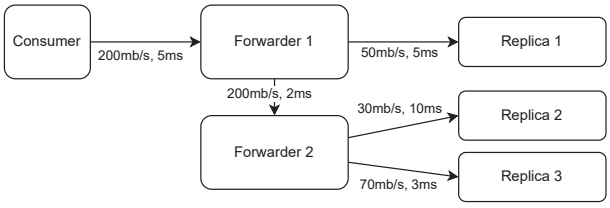

Figure 10: Scenario 2: Simple multi-path topology

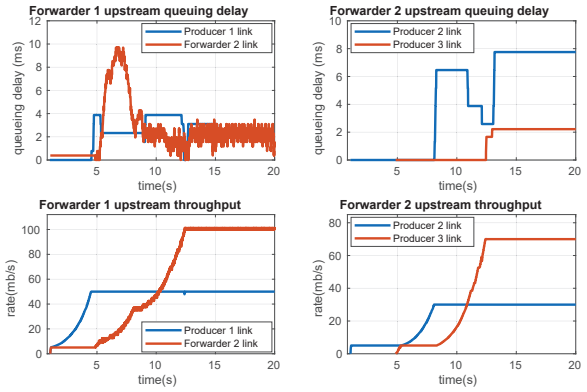

Figure 11: Scenario 2 simulation results

6.3 Multiple Forwarding Paths Sharing the Same Bottleneck Link

Multiple forwarding paths may end up merging into the same bottleneck link. We evaluate our design in this setting as illustrated in fig.12. While there are three different forwarding paths for forwarder 1 to choose from, all of them share the same bottleneck link of 150mb/s.

The simulation result is shown in fig. 13. The system converged to full bottleneck bandwidth utilization in 8 seconds, and queuing delay was effectively controlled. After start-up, forwarder 1 probed its link to forwarder 2 and consequently directed almost all traffic through that link. However, the traffic splitting between the three alternative paths changes slowly afterward. At 60 seconds, around half of the traffic is directed to forwarder 4. The changing traffic splitting is due to forwarder 1 not knowing that all the three paths are merged to the same bottleneck link. The probing of an alternative path when the bottleneck is fully utilized caused the change of traffic splitting. We believe slow "drifting" of the traffic splitting when multiple forwarding paths merge to the same bottleneck link is inherent in our design since forwarders do not know about the shared nodes in the end-to-end forwarding path used. Fortunately, while the multi-path forwarding decision is not perfectly stable, link capacity is still effectively utilized, and queuing delay is controlled.

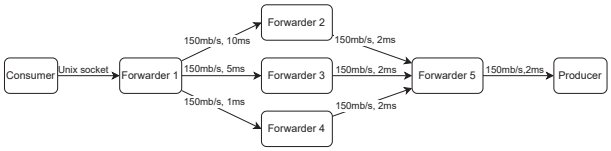

Figure 12: Scenario 3: Multiple paths merging to use the same bottleneck

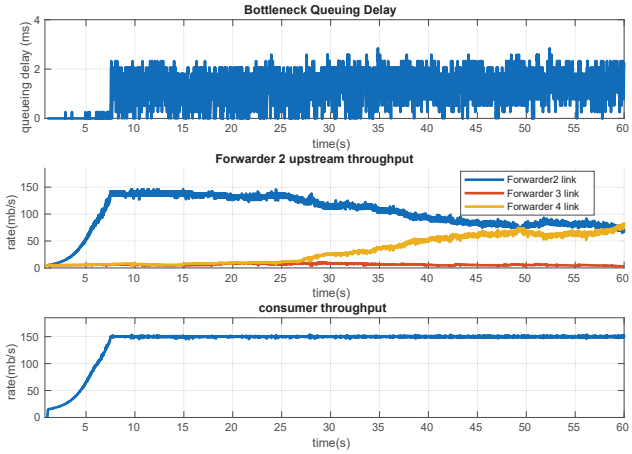

Figure 13: Scenario 3 simulation results

6.4 Throughput Maximization of Multi-Path Forwarding

The fourth scenario used for evaluation is the topology we discussed in §5.1.1 as illustrated in fig. 7. In this topology, the consumer has a high-bandwidth low-delay link to its local forwarder. The only way to achieve maximum throughput is to have forwarder 1 equally split the traffic to the three available paths while forwarder 3 must send all its traffic through to the producer. In this optimal setting, the network throughput can be at most 120mb/s.

The simulation result is shown in fig. 14. In the throughput figure, it can be seen that at 7 seconds, forwarder 3 is still sending to forwarder 2 and forwarder 4. This multi-path forwarding decision does not maximize network throughput. Then, the rate-limit calculation module tries to move traffic to the middle path using the rule of "avoiding unnecessary contention". Consequently, forwarder 3 gradually gives up sending towards forwarder 2 and forwarder 4, increasing the link's throughput towards the producer. As a result, forwarder 1 can direct more traffic toward the path in the top and bottom. Eventually, the network throughput is close to optimal.

The per-hop delay measured at forwarder 1 and forwarder 3 shows that the queuing is effectively controlled in steady-state. The average queuing delay at each hop is around 5ms, which is much less than the round trip time of each hop.

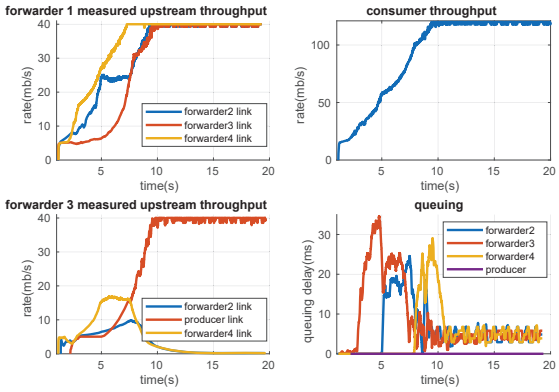

Figure 14: Throughput maximization evaluation results

6.5 Congestion Control under both Caching and Multi-Path Forwarding

A simple topology with two consumers requesting the same object, as illustrated in fig. 15 is used to evaluate our design under both in-network caching and multi-path forwarding. Both consumers are requesting the same segmented object in order. Consumer 1 started first, and consumer 2 started 1.5 seconds later. As forwarder 1 has a large cache, it is expected that consumer 2 will start to be satisfied by the cache in forwarder 1. The high bandwidth between consumer 2 and the cache allows it to catch up with consumer 1. Ideally, both consumers will eventually make the same progress requesting the object, and data multicast is used to improve consumer throughput.

The simulation result is shown in fig. 16. The multi-path forwarding converged to fully utilize upstream bandwidth in one second. The queuing delay is stable and low in the steady-state. From

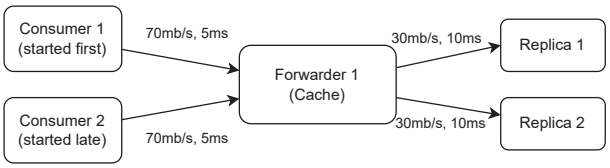

Figure 15: Scenario 5: Simple topology with caching

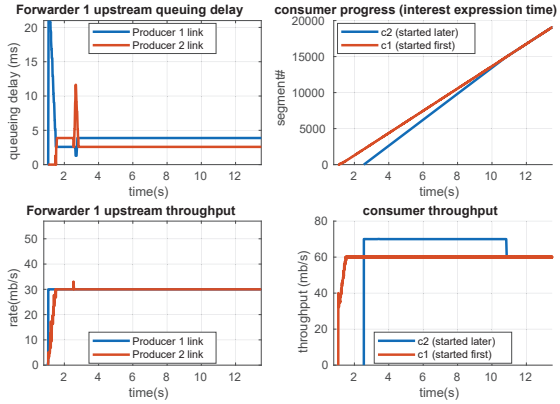

Figure 16: Scenario 5 simulation results

the consumers' perspective, it can be seen that the later started consumer 2 can catch up with consumer 1 by utilizing its high bandwidth to cache. When satisfied by a cache, the forwarder's queue size feedback to consumer 2 does not include the input queue size. Consequently, consumer 2 can send faster than consumer 1, whose queue size feedback is affected by the input queue of the forwarder. After consumer 2 catches up, both consumers retrieve data at 60mb/s, which is the total multi-path bandwidth towards the data object. During this time, data multicast is effectively used.

6.6 Multiple Consumers with Fair Queuing

We use a scenario where multiple consumers fetch different objects to validate that fair-queueing can be used with NDN-QSF to provide fairness. As illustrated in fig. 17, in the scenario there are three consumers fetching different objects. The consumers start and end at different times to emulate new flows joining and leaving the network. Their traffic shares a single bottleneck between the forwarders. The bottleneck is congested when forwarder 2 is queuing up large data packets towards the consumers. Thus we added a fair queuing mechanism in forwarder 2's data forwarding pipeline. Data from each name prefix are put in separate queues with the same fair-queueing weight. This fair queuing setup provides a per-name prefix fairness.

The simulation result is shown in fig. 18. The result shows that the system is resilient to flows' joining and leaving. After new flows join at 5 seconds and 15 seconds, the system can quickly converge to a steady state with low queuing delay, near full bandwidth utilization, and per-name prefix fairness with respect to both throughput and queuing delay.

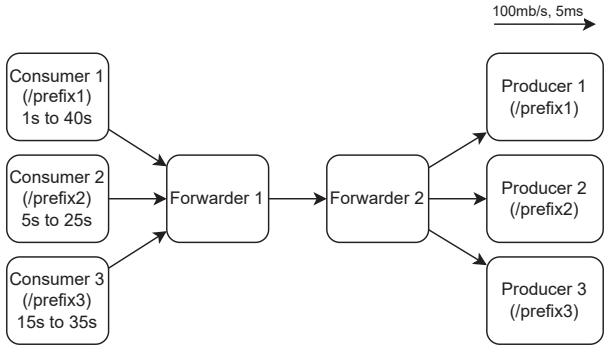

Figure 17: Scenario 6: Dumbbell topology with three consumers fetching different objects

This result validates the idea that fair queuing can be used together with NDN-QSF design to form a more complete congestion control and multi-path forwarding solution in NDN.

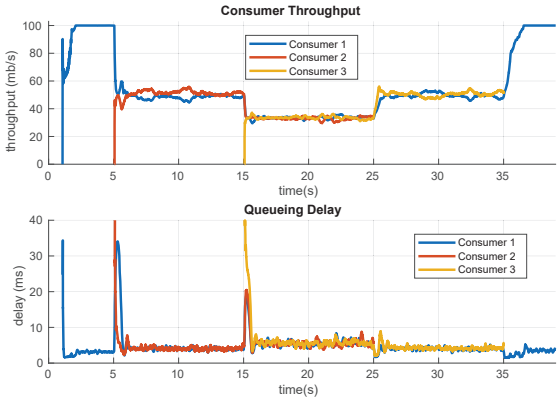

Figure 18: Scenario 6 simulation results

7 DISCUSSION

NDN Congestion Control Requires Network Feedback. NDN's in-network caching and multi-path forwarding help scale data delivery and maximize overall network throughput. An earlier work attempted to make end consumers' measurement "resilient to RTT variations" [12]. However, investigations show that both mechanisms affect consumers' RTT measurements, making the RTT variations no longer a reliable indication of network congestion status. Consequently, the well-established point-to-point and end-to-end congestion control solutions can no longer be adopted in NDN networks. It motivates us to use forwarders' measurements as feedback for congestion detection. As we discussed in §2, although different solutions use different congestion feedback, they share the use of forwarders' local measurements of network load, such as queue size and PIT table size.

Focus on Fetching Data, no point-to-point Flow. We designed our congestion control solution without establishing the equivalent of an end-to-end single-path flow as in IP networks that other designs

did [4, 10]. NDN-QSF does not mark forwarding paths on data packets or let consumers explicitly establish end-to-end flows that are forwarded along a single path by marking "flow ids" in interest packets. Evaluation of our design shows that congestion control and multi-path forwarding can be effective without differentiating end-to-end single path flows.

Understanding the Cost. Because a router fetches data in different name prefixes and from different producers, NDN-QSF lets each router keep an Interest queue for each name prefix, measure the data arrival rate to estimate upstream capacity and inform downstream routers of the observed max queue size. Performing the above tasks add additional per name prefix state to an NDN router. Thus the amount of memory required for our solution is proportional to the number of name prefixes, which may result in the scalability limitation of our design when a network has a large number of name prefixes. This issue deserves further investigation.

Fairness and Congestion Control. Our design of NDN-QSF is yet to consider the fairness issue. As pointed out in [11], different from congestion control in an IP network, fairness in NDN needs a new definition, per-name prefix fairness versus per consumer fairness. Once this definition is settled, we believe our approach can be extended to support fairness. Fair queuing can be used to provide fairness across different consumers and objects. Per-name prefix fairness, if it is desired, can be achieved by fair queuing on the interest queues of different name prefixes. Per-consumer fairness would be more complicated to achieve, as it requires downstream routes to inform upstream about the number of consumers whose request is forwarded to the upstream. The upstream can then separately queue incoming interests from different downstream and fair-queue them using the reported number of consumers from each downstream as weight. One proximation to simplify the implementation could be letting each router perform per-downstream face fairness.

Open Questions and Future work. This work leaves several questions open and worth to be worked on in the future. Firstly, we have only performed a limited evaluation of NDN-QSF. The experiments we used are based on relatively simple network topologies and lack noise or varying background traffic. We also did not perform a comparison with other NDN congestion control or multi-path forwarding algorithms to show the advantages and disadvantages of our proposed solution. Second, we have not made detailed investigation into the effects and selection methods of some parameters (e.g., queue threshold). We hope to clearly identify trade-offs associated with these parameters and provide suggestions for their selection in the future. Finally, as mentioned earlier, NDN-QSF does not provide fairness between different consumers or different name prefixes. We only validates that fair queuing can be used with NDN-QSF in §6.6. We hope to introduce fairness-related primitives into NDN-QSF in the future and experiment with them to build a more complete solution.

8 CONCLUSION

In this work, we investigate the effect of multi-path forwarding and in-network caching on congestion control and show that it's difficult for consumers to detect network congestion based on their local measurements in NDN. In response to this issue we propose to use forwarder's local queue size as congestion feedback to downstream. We designed a congestion control solution for NDN, NDN-QSF, that works in the presence of in-network caching. The congestion control design is hop-by-hop with forwarders' local queue size as congestion feedback to downstream. The forwarders measure the available bandwidth through its upstream link and limits interest transmission rate based on estimated bandwidth and congestion feedback of the upstream. NDN-QSF is then applied to multi-path forwarding environment and extended to make multi-path forwarding decisions. Through extensive simulation, we show that the proposed design effectively controls congestion while utilizing the bandwidth of multiple forwarding paths.

Our initial design of NDN-QSF identified multiple remaining issues to be addressed in future study. The first is to experiment with NDN-QSF in much larger setting with bigger topologies and many more consumers and data producers. The second one is to address potential scalability challenge because NDN-QSF builds Interest queues on a per producer basis. Yet another one is to define, and achieve, fairness in situations where multiple consumers fetching multiple objects.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable comments which helped us improve the paper's quality. This work is partially supported by the National Science Foundation under award 2019012.

This material is partially based upon work supported by the AFRL under contract No. FA9453-21-C-0554. The views expressed are those of the authors and do not reflect the official guidance or position of the United States Government, the Department of Defense or of the United States Air Force.

REFERENCES

[1] Alexander Afanasyev, Ilya Moiseenko, Lixia Zhang, et al. 2012. ndnSIM: NDN simulator for NS-3. (2012).

[2] Venkat Arun and Hari Balakrishnan. 2018. Copa: Practical Delay-Based Congestion Control for the Internet. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. 329–342.

[3] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue* 14, 5 (2016), 20–53.

[4] Giovanna Carofiglio, Massimo Gallo, and Luca Muscariello. 2016. Optimal multipath congestion control and request forwarding in information-centric networks: Protocol design and experimentation. *Computer Networks* 110 (2016), 104–117.

[5] Jiachen Chen, Mayutan Arumaithurai, Xiaoming Fu, and KK Ramakrishnan. 2016. SAID: A control protocol for scalable and adaptive information dissemination in ICN. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. 11–20.

[6] Dah-Ming Chiu and Raj Jain. 1989. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN systems* 17, 1 (1989), 1–14.

[7] Lester Randolph Ford and Delbert Ray Fulkerson. 2015. Flows in networks. In *Flows in Networks*. Princeton university press.

[8] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS operating systems review* 42, 5 (2008), 64–74.

[9] Stratis Ioannidis and Edmund Yeh. 2018. Jointly optimal routing and caching for arbitrary network topologies. *IEEE Journal on Selected Areas in Communications* 36, 6 (2018), 1258–1275.

[10] Milad Mahdian, Somaya Arianfar, Jim Gibson, and Dave Oran. 2016. MIRCC: Multipath-aware ICN rate-based congestion control. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. 1–10.

[11] Klaus Schneider, Cheng Yi, Beichuan Zhang, and Lixia Zhang. 2016. A practical congestion control scheme for named data networking. In *Proceedings of the 3rd*

ACM Conference on Information-Centric Networking. 21–30.

[12] Sichen Song and Lixia Zhang. 2021. Exploring Rate-Based Congestion Control in NDN. In *Proceedings of the 8th ACM Conference on Information-Centric Networking (ICN '21)*. Association for Computing Machinery, New York, NY, USA, 141âĂŞ143. https://doi.org/10.1145/3460417.3483379

[13] Edmund Yeh, Tracey Ho, Ying Cui, Michael Burd, Ran Liu, and Derek Leong. 2014. VIP: A framework for joint dynamic forwarding and caching in named data networks. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*. 117–126.

[14] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, KC Claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. 2014. Named data networking. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 66–73.