BBR-Inspired Congestion Control for Data Fetching over NDN

Yi Hu, *Peraton Labs*, yhu@perspectalabs.com
Constantin Serban, *Peraton Labs*, cserban@perspectalabs.com
Lan Wang, *University of Memphis*, lanwang@memphis.edu
Alex Afanasyev, *Florida International University*, aa@cs.fiu.edu
Lixia Zhang, *UCLA*, lixia@cs.ucla.edu

***Abstract*—In this paper we explore congestion control solutions in a network running the Named Data Networking (NDN) protocol. Since requested data can be retrieved from either original producers or in-network caches, the end-to-end connection oriented TCP congestion control solutions, such as BBR (Bottleneck Bandwidth and Round-trip propagation time) do not work well for NDN transport, especially in wireless networks. We propose a BBR-guided congestion control, *BBR-CD*, for bulk data fetching by a group of users. *BBR-CD* applies RTT filtering and interest scheduling to improve BBR's efficiency in NDN. Our evaluation shows that *BBR-CD* achieves much higher application goodput compared with *ndncatchunks* which implements the TCP AIMD and CUBIC congestion control algorithms. Moreover, compared with one-flow-per-file *SIRC*, an NDN congestion control algorithm that paces interests based on inter-data-gap, *BBR-CD* provides better fairness and bandwidth utilization. Finally, *BBR-CD* outperforms a direct adoption of the original BBR to NDN, suggesting that our changes to BBR are effective.**

***Index Terms*—Named Data Networking, NDN, Congestion Control, Wireless Networks, Lossy Networks**

I. Introduction

In Named Data Networking (NDN) [1], [2], named, secured data is the centerpiece of communication, and consumers request desired data by name. All data items in NDN are cryptographically signed and encrypted if needed, hence providing inherent data security. Moreover, NDN provides efficient and robust data distribution using stateful data plane, request aggregation, and in-network caching. These features make NDN a much better fit than IP for disadvantaged wireless networks that are often infrastructure-less, ad hoc, and deployed on-demand anytime and anywhere. Our earlier work addressed how to signal data generations for consumers [3] in such networks, now we focus on the performance of the data delivery by applying congestion control to regulate interest transmissions and retransmissions in response to available resources and network dynamics. We focus on controlling the interest pipeline at data consumers, which can be used in conjunction with hop-by-hop interest traffic shaping.

Existing TCP congestion control mechanisms cannot be directly applied to NDN, as NDN's in-network caching invalidates the assumption of a TCP connection between two specific nodes for data delivery. More specifically, in-network caching can lead to large variations in the RTTs perceived at a consumer, making those congestion control algorithms that overly rely on precise RTT estimation unreliable. Furthermore, TCP's control-loop reacts to packet losses in the same way, regardless of the causes (e.g., collisions, errors, congestion, or node mobility), hence non-congestive losses result in network under-utilization.

We propose BBR-CD, a BBR [4] guided congestion control for data delivery in NDN. Similar to BBR, it uses the estimate of the bottleneck Bandwidth-Delay Product (BDP) to calculate the congestion window and NDN interest pacing rate, and periodically probes the available bandwidth by boosting and draining the inflight traffic to converge to a fair share of network resources among competing data flows. Simple BBR, however, does not work well when the measured RTTs have large variations depending on whether the data is retrieved from network caches or data sources. When multiple consumers fetch the same content simultaneously, the combined effects of interest losses and interest aggregation at routers make it impossible to get accurate RTT samples.

To address the above issues, we apply three modifications to improve BBR's efficiency in NDN: (1) we use average RTT, instead of the min RTT, to calculate BDP to reduce the impact of high variance in RTT estimation at the consumers; (2) we further apply RTT filtering to remove noise in RTT samples to eliminate effects such as data retrieval from local caches or added delays from interest timeouts due to wireless losses; and (3) we modified the conditions for transition to and from BBR's ProbeRTT state, in order to collect multiple RTT values to compute meaningful average RTT measures.

Our evaluation shows that BBR-CD achieves much higher goodput than *ndncatchunks* which implements the TCP AIMD and CUBIC congestion control algorithms. Moreover, compared with SIRC [5], a proposed NDN congestion control algorithm that paces interest sending rate based on inter-data-gap, BBR-CD provides better fairness and bandwidth utilization due to its BDP estimation and bandwidth probing. Finally, our results demonstrate that BBR-CD outperforms a direct adoption of the original BBR to NDN, suggesting that our changes to BBR are effective.

The rest of this paper is organized as follows. Section II presents the application model and its requirements. Section III discusses the design of the BBR-CD congestion control, and Section IV presents the evaluation scenario and the performance of BBR-CD. We discuss related work in Section V and conclude the paper in Section VI.

II. Application Model

We assume producers generate files at unpredictable intervals, and a group of consumers need to obtain each produced file as soon as possible. The producer and consumers are assumed to use a synchronization protocol, e.g., PLI-Sync [3], to notify all the consumers of the data generation, and the consumers starts fetching the file upon receiving the notification. Each file is segmented into multiple NDN data packets using the default NDN packet size of 8600B, and consumers retrieve each data packet by issuing an NDN interest packet. Intermediate NDN routers cache data packets passing by.

Figure 1 (top) shows a sample topology where two consumers can reach a data producer through a router in a wireless network. An NDN data packet can be fetched from either the original producer or the router cache, if an earlier interest has retrieved the data. Figure 1 (bottom) shows a topology where data consumers are connected to the producer via multiple routers. We assume that NDN routers perform opportunistic caching based on their resource availability with no coordination for cache management, and that the topology may change dynamically due to mobility or disconnections. Consequently, the consumers have no prior knowledge of which data packet is cached at which router.

There are multiple applications and use cases that fit the above model. One example is file generation and dissemination by an emergency response team, where maps, photos, and audio files need to be delivered over wireless networks, in real time, once generated. Other examples may consist of real time video generation and transmissions to dynamic groups of users over a variable network topology.

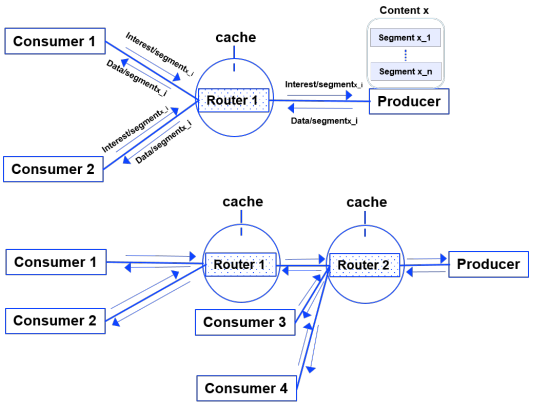

Fig. 1. NDN Data Delivery Transport

III. Congestion Control Description

A. Original Bandwidth-Delay Product Congestion Control

As a congestion control algorithm, BBR measures the bandwidth-delay product (*BDP*) of the network path over which a transport flow travels [4] by two parameters: the bottleneck bandwidth available to the transport flow (*BtlBw*), estimated from the maximum delivery rate; and the round-trip propagation delay of the path (*RTprop*), estimated from the the minimum round-trip delay. BBR aims to operate at a point where it can achieve both high throughput and low latency. To achieve high throughput, BBR tries to make the packet arrival rate match the bottleneck bandwidth available to the flow. To achieve low latency, BBR tries to keep the total data in flight along the path equal to the estimated *BDP* of the path (*BtlBw*×*RTprop*), avoiding the build up of queues. BBR maintains these conditions by controlling three parameters: *pacing rate*: the inter packet spacing at the time each packet is scheduled for transmission by a BBR sender; *send quantum*: the amount of data scheduled and transmitted together; and *congestion window (cwnd)*: the amount of data that can be in-flight at any moment for a given connection.

BBR operates a state machine with four states. A new connection enters the *Startup* state. In this state, BBR exponentially increases the sending rate until the flow reaches the bottleneck bandwidth, and then it enters the *Drain* state to reduce the number of queued-up packets. After the Drain state achieves 1 *BDP*, BBR enters the steady state, *ProbeBW* state, where it periodically raises the number of in-flight packets to probe whether a higher bandwidth is available. At a much coarser time granularity (e.g., 10sec), BBR goes to the *ProbeRTT* state, where it reduces the number of packets on-the-fly to a small value (e.g., 4 packets) to reduce the queues in the network to measure the minimum RTT.

B. BBR-CD Overview

BBR-CD adopts the four states in the state machine defined in the BBR algorithm but adapts the state machine to provide an NDN consumer-driven congestion control scheme. A BBR-CD flow represents the interaction between one consumer and either the data producer or network caches serving the data. Therefore, the BBR-CD flow model represents the set of overlapping paths of different lengths taken by the interests issued by a consumer, and the corresponding data items propagating back to the consumer.

Similar to BBR, BBR-CD uses *cwnd* and *pacing rate* to control the amount of traffic in-flight and regulate the inter-packet spacing time. More specifically, $cwnd = cwnd\_gain * estimatedBDP + quanta$, where *cwnd_gain* is a scaling factor to the estimated *BDP*, and its value is selected by the BBR state machine. BBR-CD follows BBR to use a quanta term, which is set to $3 * NDN\_packet\_size$. The scheduler in BBR-CD calculates the next interest sending time after a data arrival or an interest timeout event, as $now() + NDN\_packet\_size/pacing\_rate$, where $pacing\_rate = pacing\_gain * estimatedBtlBW$. Similar to *cwnd_gain*, *pacing_gain* is a scaling factor to the estimated bottleneck bandwidth and its value is set by the BBR state machine. BBR-CD sets the NDN data packet size to be 8600B to amortize the per-packet security overhead.

An important difference between BBR and BBR-CD is that, rather than using the minimum round trip time for *RTprop*, BBR-CD uses the estimated average of the round trip time over a time window for *RTprop*. For a single flow i at time T, BBR-CD calculates *RTprop* using the average of all samples

for data packets of the flow within a time window W_i as $\widehat{RTprop_i} = avg(RTT_t^i), \quad \forall t \in [T - W_i, T]$ (all samples within the time window have the same weight). The length of the window W_i may depend on the expected frequency of path changes and other dynamics. BBR uses a fixed value of 10 seconds, while BBR-CD sets the time window to be β times the latest estimated average RTT. Empirically, we set β to 8 for evaluations in IEEE 802.11n networks. The running average RTT alone cannot sufficiently reflect the dynamics of BDP in data retrieval, so BBR-CD periodically probes whether the available BtlBW has changed and measures the magnitude of the change in average RTT to guide interest transmission. We will elaborate on the probing procedure below in III-C.

BBR-CD measures the RTT as the time interval between sending an interest and receiving the corresponding data, excluding the cases when data is fetched from the local cache,[1] which do not reflect network conditions, and the cases when data is returned after interest retransmissions. It is important to note that the RTT of a single interest-data exchange may not accurately reflect the true RTT between a consumer and a producer (even without caching), because in a multi-consumer scenario, identifying exactly which interest retrieves a data back can be challenging. For example, a consumer's interest may be lost upstream at an intermediate router, but an interest from another consumer for the same data segment arrives before the lost interest times out, and fetches the data, which is sent by the intermediate router to both consumers. Thus, the first consumer's perceived RTT includes the true RTT value plus a time gap between the loss of the first interest and the arrival of the second interest. In consideration of this ambiguity in RTT measurements, BBR-CD sets the interest retransmission timer to be the sum of RTprop and a cushion equal to multiple RTT standard deviations. Specifically, BBR-CD has a Loss Detector Module which resends an interest that was sent at time T for flow i if the corresponding data has not been received by $T+\widehat{RTprop_i}+\alpha*\delta_i$, where δ_i is the standard deviation of all RTTs over time window W_i for data flow i, and α is a scaling factor to cover possible variation in RTTs, e.g., due to different data retrieval path lengths. Empirically we set $\alpha = 10$. After an interest times out, BBR-CD resends the interest, and will not include the RTT of the corresponding data in the BDP estimation.

BBR-CD tracks the delivery rate to estimate the bottleneck bandwidth. When a data item arrives at a consumer at time T, BBR-CD measures the data's RTT and the delivery rate of data. The delivery rate for the interval between interest sent and data received events is the ratio between the amount of data delivered and the time elapsed, i.e., $deliveryRate_T^i = \Delta_{delivered_i}/\Delta_T$. This rate computation excludes the cases of interest being retransmitted, therefore the resulting rate must be less than or equal to the available bottleneck rate for the data flow. Similar to the average RTT value, BBR-CD calculates the BtlBw as the unweighted average of all delivery rates sampled over a time window of flow i to estimate the effective bandwidth, $\widehat{BtlBw_i} = avg(deliveryRate_t^i), \quad \forall t \in [T - W_i, T]$.

[1]This can happen when NDN applications are restarted without restarting the local NFD, for example.

C. BBR-CD Control Algorithms

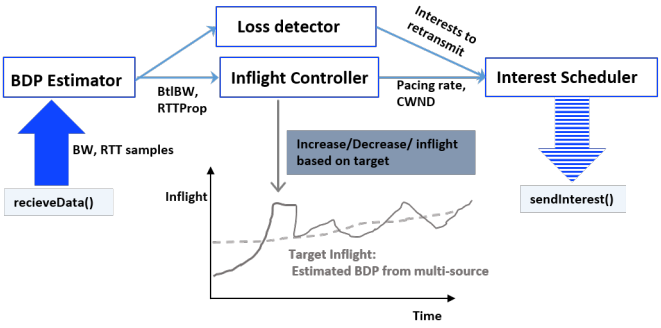

Fig. 2. BBR-CD Congestion Control Algorithm

BBR-CD congestion control consists of four components: *BDP Estimator*, *Loss Detector*, *Inflight Controller*, and *Interest Scheduler* as shown in Figure 2. The BDP Estimator records the timestamp and status of the delivered data when an interest is sent, and calculates $\widehat{RTprop_i}$ and $\widehat{BtlBw_i}$ samples when data packets are received. It updates the time window records and feeds the estimated $\widehat{RTprop_i}$ and $\widehat{BtlBw_i}$ to the Inflight Controller and the Loss Detector.

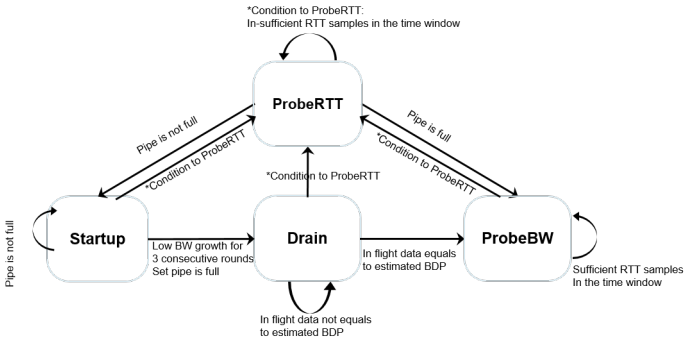

Fig. 3. BBR-CD State Transition

Figure 3 shows an overview of the state transitions among the Startup, Drain, ProbeBW, and ProbeRTT state in the Inflight Controller.We made a few modifications to the original BBR state machine.

a) Startup: A data flow enters the Startup state where it increases its cwnd to reach a constant gain factor, which is the ratio of inflight data size over the estimated BDP. We set $highGain = 2.885$, $cwnd\_gain = highGain$, and $pacing\_gain = highGain$. We initialize $cwnd$ to 1 for generality. The Startup state performs an exponential search of the available bandwidth, doubling the sending rate each round to find *BtlBw* in $O(\log_2(BDP))$ round trips. The controller estimates whether the pipe is fully utilized by looking for a plateau in the *BtlBw* tracked by the BDP estimator. If the

controller observes that three consecutive rounds of doubling interest sending rate result in little data rate increase ($<25\%$) of *BtlBw*, it concludes that the full capacity of the available *BtlBw* is reached. It then moves from the Startup state to the Drain state to drain likely inflated queues.

b) Drain: In the Drain state, the controller switches to use a gain factor well below 1.0, i.e., $drainGain = 1/highGain$ and $pacing\_gain = drainGain$, to drain any queue created during the Startup state, while $cwnd\_gain = highGain$ to keep the $cwnd$. When the inflight data size matches the estimated BDP from the Startup state, the flow moves from the Drain state to the ProbeBW state.

c) ProbeBW: The ProbeBW state is a flow's steady state, probing for bandwidth changes using gain cycling to set the value of $pacing\_gain$, in which the gain factor of a flow cycles through a sequence of values. We adopt BBR's 8-phase gain cycle with the following gain factor values: $[5/4, 3/4, 1, 1, 1, 1, 1, 1]$. Each phase normally lasts for roughly one RTT so that a flow periodically raises or lowers inflight data size to probe for *BtlBw* samples. $cwnd\_gain$ is set to 1 and stays constant in this state.

d) ProbeRTT: BBR-CD's implementation of ProbeRTT differs from that in BBR. Instead of entering the ProbeRTT state when the minimum RTT has not decreased for a period of time, the flow enters the ProbeRTT state when not enough RTT samples are collected over the given time window W_i, which means the flow i either encountered interest timeouts or the data retrieval path is disrupted (e.g., due to producer/consumer mobility). In either case, flow i decreases the interest sending rate to k per second, a low enough value that should not cause congestion; we empirically set k to 4. BBR-CD leaves ProbeRTT when there are sufficient RTT samples collected over time window W_i to enter either Startup or ProbeBW, depending on it estimate of whether the pipe was filled already. We use a threshold value (empirically set to 4 in our experiment) to decide whether sufficient samples are collected or not.

The Loss Detector periodically checks whether pending interests are timed out (lost). The Interest Scheduler paces the interest transmission based on the *cwnd* and *pacing rate* output from the Inflight controller. When a data segment is received or a pending interest is timed out, the interest scheduler sets the next send time to be packet size divided by pacing rate, where pacing rate is computed as $\widehat{BtlBw_i} * pacing\_gain$.

IV. Performance Evaluation

A. Experiment Settings

We implemented the BBR-CD in software using multiple versions of NFD and ndn-cxx library (0.6.2, 0.6.6, and 0.7.1), and ported the stack on different platforms (Ubuntu 16.04, Ubuntu 18.04, Android 7.2, and Android 8.0). The platforms are deployed in up to 100 devices, connected by a variety of wireless networks. The evaluation described in this paper was performed in a hybrid environment, consisting of virtual machines hosting the NDN software stack and applications, and an ns-3 simulation-in-the-loop network model simulating network environment. We focused on evaluating networks consisting of one mobile device (Topology 1, Figure 4 left) or groups of ten mobile devices (Topology 2, Figure 4 right), an AP, and a producer. Each mobile device as a consumer is connected via WiFi 802.11n to an Access Point (AP), and the AP is further connected via another WiFi 802.11n network to the content producer. Note that the APs are routers running NFD. Each AP is configured to operate at a HtMCS6 rate of 50Mbps used by one consumer in Topology 1, and shared by ten consumers in Topology 2. The link between the AP and the producer is also 50Mbps. Due to the poor performance of existing Wifi broadcast/multicast implementations in most devices, we set up each consumer and the producer to use an UDP unicast face towards their router(s). As a result, while each consumer hosts an NFD with a local data cache, this cache is not shared by other consumers, since their interests are only sent to the AP and upstream.

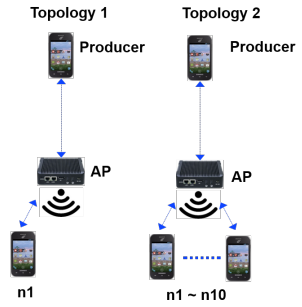

Fig. 4. Evaluation topologies

We compare BBR-CD with three congestion control schemes: 1) *ndncatchunks* [6], 2) SIRC [5], and 3) the original BBR directly ported to NDN, without our modifications.

ndncatchunks provides three types of interest pipelines in data fetching: 1) fixed window size, 2) adjusting congestion window size using the TCP AIMD algorithm, and 3) adjusting congestion window size according to TCP CUBIC algorithm. We compared BBR-CD with *ndncatchunks* using both AIMD and CUBIC, and found that AIMD outperforms CUBIC in our topologies and application, so we present the AIMD results.

SIRC follows AIMD initially until 10 inter-data gap samples are collected, it then regulates interest sending based on inter-data gap samples. Since no public implementation of SIRC is available, we implemented SIRC based on Section III in [5]. SIRC assumes that each returned data packet carries the ID of the node from which the data is fetched and establishes per path flow. However, we implemented per file flow as the standard NFD implementation does not attach node ID to the data packet from a cache. An SIRC flow is controlled by the algorithm described in [5].

We also implemented the original BBR [7] in NDN, using the minimum RTT sample over 10-sec time window as *RTprop* and the maximum delivery rate over $10 * RTprop$ time as *maxBtlBW*, as per the specs. The original BBR also follows state transition rules specified in [7]. The starting window size is set to 1 for all schemes.

We use *goodput*, defined as the application data size divided by the delivery time, as the main metric to measure the effectiveness of the congestion control scheme. We let each consumer fetch a 20MB file from the producer.[2] The delivery time is the time duration from when the first interest is sent until the time when the last data segment is received by a consumer. We tested the cases of both one consumer fetching and ten consumers simultaneously fetching, and the goodput result is the average over 10 rounds with the whisker showing the min and max values of the goodput among the 10 rounds.

B. Performance in Single/Multiple Consumers

Figure 5 shows that, for both topologies, BBR-CD's goodput is higher than the other three schemes[3]. Interest pacing is an important factor contributing to this performance gap. Figure 6 compares the pacing interval between successive interests of the four schemes in the single-consumer topology. BBR-CD has the smoothest interest transmission, where the pacing interval falls mostly between 1 and 10 msec. SIRC's interest pacing based on inter-data-gap shows slightly worse performance than BBR-CD in the single-consumer case, with the pacing interval falling mostly between 0.1 msec and 10's of msec. BBR-original's interest pacing uses the collected min RTT and max delivery rate, which spread the pacing interval wider than those of BBR-CD and SIRC, with periods of no data received. Ndncatchunk AIMD does not pace interest transmission, thus the intervals between its interests range from 1 msec to more than 10 sec.

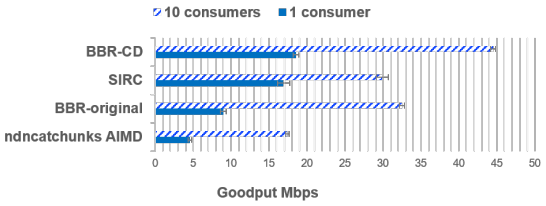

Fig. 5. Goodput in Topology 1 and 2 without Added Losses

C. Performance in Lossy Settings

We introduced packet loss in topology 1 and 2 using the receiver-loss model as follows: 3% at each consumer, 3% at the gateway router, and 4% at the producer, resulting in a roundtrip loss of approximately 10%. Figure 7 shows that BBR-CD outperforms the other three schemes in goodput. In addition to the pacing factor mentioned earlier, superfluous interest retransmissions contributed to the performance gap in this case. Analysis of the logs show that, about 50% of total interests transmitted by *ndncatchunks* AIMD were retransmissions, but only about 10% of total interests transmitted by BBR-CD and BBR-original were retransmissions. AIMD cannot correctly identify whether the variation of RTTs is due to congestion or packet losses, *ndncatchunks* shows repetitions of data bursts

[2]The AP's content store size is set to the default value of 65,536 packets, which can hold the entire 20MB file.

[3]The goodput in the 10 consumer case is summed over all consumers.

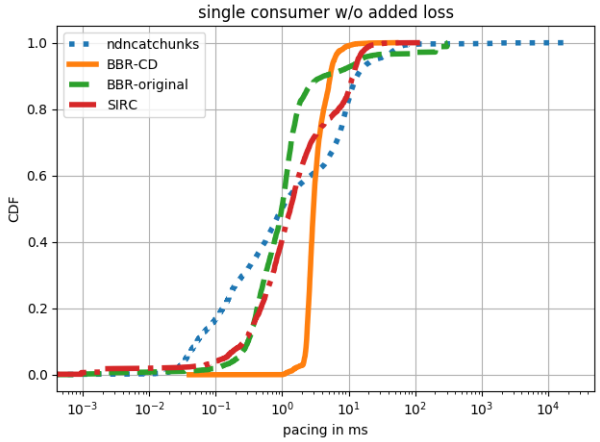

Fig. 6. CDF of Interest Pacing Interval in Topology 1 without Added Losses

and starvation periods. In contrast, the other three schemes are able to sustain the data retrieval without starvation.

Fig. 7. Goodput in Topology 1 and 2 with Added Losses

D. Performance with Intermittent Connection

We tested the disconnection and reconnection of the producer during a data transfer in Topology 2, and we measured the delivery time increase ratio caused by the producer disconnection. For example, across the ten consumers, BBR-CD's mean delivery time for 20MB data increased from 37.8 sec to 50.6 sec (discounting the disconnected period) when the producer was disconnected 32 sec during the delivery process, the mean delivery increase ratio was $(50.6 - 37.8)/32 = 0.4$. An increase ratio below 1 means that the consumers were fetching data from the AP cache when the producer was disconnected, and the smaller this ratio the better. Table I compares the

TABLE I
DELIVERY INCREASE RATIO WITH PRODUCER DISCONNECTION

-	Min	Mean	Max
BBR-CD	0.625	0.4	0.33
catchunks	0.91	0.86	0.812

delivery time increase ratio between *ndncatchunks* AIMD and BBR-CD. We can see that all three delivery time metrics for the NDN congestion control schemes are below 1. However, BBR-CD has much lower increase ratios than *ndncatchunks* because (a) it does interest pacing to slow down the sending

rate when the available bandwidth is reduced (as a result of the producer being disconnected), and (b) it prioritizes the transmission of the initial interests over retransmissions for better in-network cache utilization.

V. Related Work

A number of NDN congestion control schemes have been developed [8] which can be classified into three categories: consumer-based control, hop-by-hop control, and hybrid methods.

One natural way to perform congestion control is by controlling the interest sending rate at the consumer side (e.g., [5], [9], [10]), which matches NDN's receiver-driven "Pull" based communication model. BBR-CD is also consumer-based control, but it differs from previous work in this category in that it handles multiple dynamic paths introduced by in-network caching, without assuming that the consumer has prior knowledge of which packet will be served by which source or which path. In [9] and [10], a consumer maintains an RTO for each data source or each transfer path, and uses AIMD to adjust the cwnd associated with an RTO. Such schemes assume the consumer can retrieve data source or path information from NDN's forwarding layer. This assumption is impractical for the dynamic paths introduced by in-network caching. In addition, the AIMD based window adjustment mechanism does not control the consumer's pacing rate. In SIRC [5], a consumer paces the interests based on inter-data gaps. When data arrives more frequently, the interests are pumped out more quickly. In contrast, BBR-CD adjusts pacing rates of interests by estimating the BDP, which is a general form of the inter-data gap in SIRC. The main difference is, in steady state, BBR-CD periodically probes the BW and RTT, rather than being totally reactive to the data arrival rate as in SIRC. Such probing contributes to the fairness among competing content flows.

Another NDN congestion control approach is hop-by-hop control, e.g., [11]. The basic principle of hop-by-hop control is each NDN node detects congestion by monitoring its Pending Interest Table size or interest/data arrival rate, and adjusts its interest forwarding rate to control the returning data rate accordingly. Hop-by-hop control schemes can benefit from the cooperation from consumers such as initializing interest sending rate properly.

Hybrid schemes, e.g., [12], [13] and [14], enhance consumer-based control with explicit information from the network. More specifically, consumers perform congestion control with feedback from intermediate nodes to converge to fairness for all competing flows. BBR-CD can be used in conjunction with a hop-by-hop control, i.e., a consumer estimates the BDP to reflect the intermediate nodes' traffic reshaping, further improving the performance.

VI. Conclusion

The results shown in this paper demonstrate that BBR-CD can provide higher goodput compared with TCP-based consumer side NDN congestion control schemes and inter data gap-based consumer side congestion control. This paper demonstrates that a BBR-based scheme can perform well in wireless scenarios with significant losses, disconnections, and consumer-induced dynamics. Among the topics not covered in this paper are the impact of complex network topologies, cache settings, and interest paths, as well as more complex interactions between multiple consumers and the network. Also not explored are schemes that may combine BBR guided consumer driven congestion control with traffic shaping at NDN routers. These topics are left for future research.

References

[1] V. Jacobson, D. K. Smetters, J. Thronton, M. F. Plass, N. H. Briggs, and R. Braynard, "Network Named Content," *CoNEXT*, 2009.

[2] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *ACM Computer Communication Reviews*, Jun. 2014. [Online]. Available: http://dx.doi.org/10.1145/2656877.2656887

[3] Y. Hu, C. Serban, L. Wang, A. Afanasyev, and L. Zhang, "Pli-sync: Prefetch loss-insensitive sync for ndngroup streaming," in *Proceedings of IEEE International Conference on Communications*, 2021.

[4] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time," vol. 14, no. 5, 2016.

[5] M. Amadeo, A. Molinaro, C. Campolo, M. Sifalakis, and C. Tschudin, "Transport layer design for named data wireless networking," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2014.

[6] K. Schneider, C. Yi, B. Zhang, and L. Zhang, "A practical congestion control scheme for named data networking," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, 2016.

[7] N. Cardwell, Y. Cheng, S. H. Yeganeh, and V. Jacobson, "BBR Congestion Control," Internet Engineering Task Force, Tech. Rep. draft-cardwell-iccrg-bbr-congestion-control-00, 2017. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-cardwell-iccrg-bbr-congestion-control-00

[8] Y. Ren, J. Li, S. Shi, L. Li, G. Wang, and B. Zhang, "Congestion control in named data networking – a survey," *Computer Communications*, vol. 86, 2016.

[9] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papali, "Multipath congestion control in content-centric networks," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2013.

[10] L. Saino, C. Cocora, and G. Pavlou, "Cctcp: A scalable receiver-driven congestion control protocol for content centric networking," in *2013 IEEE International Conference on Communications (ICC)*.

[11] 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," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, 2013.

[12] 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 *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, 2014.

[13] G. Carofiglio, M. Gallo, and L. Muscariello, "Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks," vol. 42, no. 4, 2012.

[14] K. Schneider, C. Yi, B. Zhang, and L. Zhang, "A practical congestion control scheme for named data networking," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, 2016, pp. 21–30.