Title

A Delay-Based Congestion-Control Protocol for Information-Centric Networks

Authors

Albalawi, A.A.
Garcia-Luna-Aceves, J.J.

Publication Date

2019

Peer reviewed

A Delay-Based Congestion-Control Protocol for Information-Centric Networks

Abdulazaz Ali Albalawi[§] and J.J. Garcia-Luna-Aceves[§†]
[§]Computer Science and Engineering Department, University of California, Santa Cruz, CA 95064
[†]PARC, Palo Alto, CA 94304
Email: aalbalaw@ucsc.edu, jj@soe.ucsc.edu

***Abstract*—The Delay-based Congestion-control Protocol (DCP) is presented. DCP is designed to detect and control congestion at the consumer end of a network based on an Information-Centric Networking (ICN) architecture without the use of round-trip time measurements or additional congestion signaling. DCP is based on a receiver-driven, window-based approach to congestion control using measurements of delays along the forwarding path from a producer or caching site to a consumer. DCP was implemented using the ndnSim simulator and compared to two other protocols, namely: an end-to-end protocol that behaves in the same way as TCP, and a pure implementation of the Low Extra Delay Background Transport (LEDBAT) algorithm as a congestion-control protocol for ICN's. The results clearly show that DCP outperforms LEDBAT when caching is involved. We evaluate the known latecomer-advantage problem that exists in LEDBAT in the context of an ICN architecture and address how DCP can overcome this issue.**

I. Introduction

This paper focuses on congestion control in the context of information-centric network (ICN) architectures in which content is disseminated using Interests issued by consumers and producers or caching sites respond to them with the content requested. Examples of Interest-based ICN architectures are Named Data Networking (NDN) [2] and Content-Centric Networking (CCN) [1]. An Interest requesting the desired content states the name of the content and other information that can be used for the retrieval of the content. Interests are forwarded towards those sites that advertise names that are the best matches to the names stated in the Interests. Data packets sent in response to Interests are sent following the reverse paths traversed by those Interests that are successfully delivered. The in-network caches used in an ICN allows Interests to be resolved from caching sites closest to the consumers along the paths to the nodes with the best-match names for the Interests.

The main approach used in the Internet to support reliable end-to-end communication and congestion control today is TCP, which relies on establishing end-to-end connections over which byte streams are sent reliably between two specific sites identified by their addresses and ports. Replacing end-to-end connections with Interest-based exchanges between a consumer and one or more other parties providing the requested content raises a number of challenges in the design of end-to-end mechanisms for reliability and congestion control. Section II provides a summary of related work on congestion-control protocols for ICN architectures addressing these challenges. Interestingly, most congestion-control protocols for ICN architectures rely on RTT estimates. The LEDBAT (Low Extra Delay Background Transfer) protocol [8] was recently introduced as the new transport protocol operating on UDP for BitTorrent, and has the distinctive feature that it is based on a delay-based congestion-control algorithm that uses queuing delay estimates over the forward path between senders and receivers of packets.

The contribution of this paper is the introduction of the Delay-based Congestion-control Protocol (DCP), an end-to-end congestion control protocol for ICN architectures based on queuing delay estimates similar to the one used in the LEDBAT algorithm. Section III presents the motivation for the design of DCP and Section IV describes its operation. Section V explains fairness issues among DCP consumers. Section VI evaluates the performance of DCP through simulations and provides a comparison between DCP and other congestion-control approaches, LEDBAT and an approach based on TCP in the context of an ICN architecture. Section VII provides our conclusions and an insight on our future work.

II. Related Work

Considerable research has been conducted in different areas of ICN architectures, such as content routing, caching policies, and congestion control. Several congestion-control schemes have been proposed in the context of ICN architectures and NDN in particular. The novelty of the schemes is that the receiver is in charge of controlling retransmissions and managing congestion, given that their design is based on the exchange of Interests and responses sent to them. Some of these proposals depend on network assistance to detect and control congestion in the network, either by shaping Interests or using congestion signaling.

Depending on the caching policy, some protocols [4], [3] assume that content chunks might be served by different sources. Accordingly, they rely on measurements of a single round-trip time (RTT) as it is done in TCP can give the wrong indications of congestion in the network, because content could be served from multiple sources over multiple paths to the same consumer. Furthermore, keeping multiple RTT's for multiple sources adds complexity on the receiver side. In addition, an out-of-order delivery can cause duplicate acknowledgments for the case of TCP, which can be used as an indication of congestion in the network. However, based

on the caching policy implemented in an ICN, data packets can be delivered out-of-order without having anything to do with congestion. Protocols that follow a hop-by-hop approach require the network router's assistance to detect and control congestion in the network. Some of these protocols control congestion by dropping Interests using Interest shapers to force a timeout at the consumer end which can be costly. Surprisingly, most of these schemes are based on using RTT estimates as the primary indication for congestion in the network.

ConTug [4] and ICP [3] are among the first proposed transport protocols for ICN's. They are window-based protocols that mimic the AIMD (additive increase multiplicative decrease) operation of TCP and assume that content chunks could be served from multiple sources. In ConTug, the retransmission timeout is set to the maximum RTT that is measured during a session. This approach can lead to a long timeout value for the Interest in a scenario in which a content chunk is served from the original producer or a remote cache, while the rest of the chunks are served from a nearby cache. If one of the chunks of the nearby caches is lost, the consumer has to wait for a long time before a retransmission timeout (RTO) is triggered and an Interest for the lost chunk is retransmitted. The same scenario can be applied to the way ICP sets the timeout value, because the timeout value is set to the mean of the maximum and minimum RTT that was measured throughout the session.

Most of the congestion control protocols proposed in the context of NDN follow a hop-by-hop approach taking advantage of NDN's stateful forwarding plane. Wang et al. [5] propose a hop-by-hop Interest shaper for congestion control. Each router shape Interests going upstream in order to control the rate of returning data thus avoiding congestion in downstream links. However, the author assumes that data and Interest size are fixed and link capacity is known. This approach is improved in [10] by incorporating congestion signaling using NACK packets, which is also proposed in [11]. Recently, Schneider et al. [13] propose PCON a congestion-control scheme fro ICN's that performs congestion signaling to consumers by marking returning data packets. The protocol does not require routers to have knowledge about the link capacity, and does not require Interest and data packets sizes to be fixed. The protocol detects congestion in the network by measuring packet queuing time over an interval and compares it to a target using CoDel AQM [12]. One of the main differences between PCON and other hop-by-hop protocols (e.g., the one in [5]) is that PCON does not reject Interests when it detects congestion but instead signals congestion back to consumers by explicitly marking returning data packets.

ICP and ContTug use RTT estimates to determine congestion in the network and follow Karn's algorithm [6], which states that a RTT estimate for a retransmitted packet cannot be used in the RTT estimation. The disadvantage of this approach is that during the period of congestion (when a packet is lost) no estimates can be made. This leads to inaccurate estimates of RTT during congestion, which may cause the consumer to retransmit prematurely or after undue delays.

III. Motivation

One of the main data structures used in NDN is the Pending Interest Table (PIT). The PIT is used to aggregate faces of Interests of the same data name. Once a router receives the data packet in response to a given Interest, it delivers it to all aggregated faces from which an Interest for the same content was received.

Estimates regarding the increase or decrease of RTT should not be used as primary indications of congestion in the network. Using RTT estimates is not efficient to determine whether congestion is increasing or decreasing along the interest-path or data-path. Fig. 1 shows an example of two Interests sent by the same consumer and their corresponding data packets. Using only RTT measurements could lead to an incorrect conclusion of congestion developing in the data-path for the second packet. However, the true cause for the increased RTT for the second data packet is congestion along the interest-path, not the data-path. Such congestion could be caused by a congestion at the PIT of one of the relaying routers, for example.

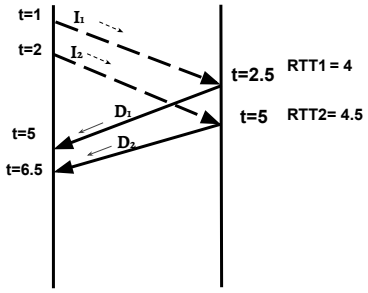


Fig. 1: Example of RTT ambiguity

Some of the congestion control protocols proposed for NDN use packet losses or Explicit Congestion Notifications (ECN) as indications of congestion in the end-to-end path. These protocols allow the consumer to increase its sending window until a loss occurs or a ECN mark is received. However, this approach leads to delayed reactions to congestion, given that a packet loss or an ECN notification at the bottleneck is triggered *after* an increase in the queuing delay occurs. Hence, consumers should use the increase in queuing delay along the data-path as an early signal of congestion, instead of waiting for packet losses or ECN notifications.

Because of the presence of PIT's, an Interest other than the one that creates the PIT entry experiences a shorter retrieval delay than the first Interest. This can cause a consumer to erroneously increase its Interest window size, because a shorter RTT is used as an indication of more bandwidth being available. To address such a problem, consumers can determine if data packets are being served from a close cache or a distance producer by using measurements of delays along the data-path instead RTT. The motivation in the design of DCP is to create an approach that can be used as a building block for future congestion-control protocols for NDN that avoid replicating

the same design and performance limitations in TCP. An ideal congestion control protocol for NDN architecture should be able achieve the following goals: (a) To utilize end-to-end available bandwidth and knowledge of whether content is being served from from a close cache or a distance producer; (b) to maintain low queuing delays for data packets in a way that does not affect the application performance; and (c) to use the bottleneck fairly link by yielding to other network flows that share the same link.

IV. DCP Design

A. Measuring Packets Delay

In NDN, consumers can calculate the one-way delays incurred by data packets in their own system time by using timestamps returned from the producer or middle nodes (caches) in the data-packet headers. A producer or middle nodes stamps the transmission time of the data packet in its header so that the consumer can calculate the one-way delay of data packet. The consumer then keeps both the transmission time and arrival time of a data packet. Based on these time stamps the consumer can calculate the one-way delay for the data packet as follows: $delay_{\mathrm{d}} = A_{\mathrm{d}} - T_{\mathrm{d}}$, where A is the arrival time for a packet and T is the transmission time of that packet. We note that aggregating Interests does not affect the calculation of the one-way delays packets in DCP even though all aggregated Interests for a certain data packet receive the same transmission time as the one given to the first Interest. This is because consumers in DCP are only interested in measuring the relative delay among data packets as an indication that content is being served from a new data site.

B. Congestion Control

To meet our first design goal, DCP uses delay measurements along the data-path to estimate the queuing delay. Whenever the estimated queuing delay is less than a predetermined threshold, the consumer infers that the network is not yet congested and increases its sending rate by increasing the congestion window (interest window) to utilize any bandwidth capacity in the network. When the estimated queuing delay becomes greater than the predetermined threshold, the consumer decreases its sending rate as a response to potential congestion in the network.

The approach used for congestion detection in DCP is shown in Algorithm 1. The algorithm is similar to the one used in LEDBAT [14] with slight changes. We use the same or similar terminology as in LEDBAT, where TARGET is the maximum queuing delay that the consumer may introduce in the network. To ensure fairness between multiple DCP flows, the TARGET value must be the same. off-target is a normalized value representing the difference between the current measured queuing delay and the predetermined TARGET. Notice that off-target can be positive or negative; a negative off-target will result in a reduction in the congestion window as the queuing delay is exceeding the application delay threshold.

C. Measuring Base Delay

A consumer using DCP can detect congestion in the network by measuring data packets queuing delay and compare it to the TARGET. However, it is important to distinguish the queuing delay from the rest of the end-to-end delay components as stated by the LEDBAT protocol. The end-to-end delay is the time taken for a packet to be transmitted across a network from source to destination, and can be decomposed into nodal processing delay, transmission delay, propagation delay, and queuing delay. Except for the queuing delay, the rest of the end-to-end delay components are considered constant.

Separating the queuing delay component from the rest of the end-to-end delay enables the consumer to detect any change in the queuing delay in the network. The minimum delay that packets experience along either the data-path or interest-path is referred to as the base delay. The base delay is considered to be the sum of all of the constant delay components since queuing delay is always an additive element of the end-to-end delay. Calculating the base delay in DCP is done with a similar approach to that used in LEDBAT. It works by maintaining a list of one-way delay minimal over a time interval. Then only the the minimum value of the minimal delays in the list is used as the approximate base delay for this path. A longer time interval results in a higher likelihood the true base delay is detected. However, a shorter time interval allows the base delay to respond quickly to route changes. We set the time interval to be 60 seconds as in LEDBAT's original algorithm and the length of the base delay list to be 6.

Algorithm 1 DCP: Estimating Queuing Delay

```
1: function on initialization()
2:       base_delay= +INFINITY
3:
4: function ONDATA(dataPkt)
5:       dataPkt ← current_delay
6:       base_delay=min(base_delay, current_delay)
7:       if (current_delay > base_delay)
8:             queuing_delay=(current_delay-base_delay)
9:             off_target=(Target-queuing_delay)/TARGET
10:      else
11:            queuing_delay=(base_delay - current_delay)
12:            off_target=(Target+queuing_delay)/TARGET
13:
14: function updateCWND(off_target)
15: end function
```

Algorithm 2 DCP: Window Update

```
1: function on initialization()
2:       flightsize= 0
3:       CWND= INIT_CWND
4:
5: function updateCWNDI(off_target)
6:       CWND+=(off_target*threshold_)/CWND
7:       max_allowed_CWND=flightsize +1
8:       CWND=min(CWNDI, max_allowed_CWND)
9:       CWND=max(CWNDI, MIN_CWND)
10: end function
```

D. Measuring Current Delay

The consumer calculates the current delay for every data packet. It is clear that the current delay value can fluctuate from one data packet to another, which affects extracting the actual delay estimate. To eliminate any outliers, LEDBAT

suggest the use of filters. One suggestion was to use Exponentially Weighted Moving Average (EWMA) of the current delay. Another suggestion was to use a NULL filter that does not filter at all. However, in our implementation of DCP the consumer maintains a fixed size list of the current delay that gets updated for every new value it calculates. The length of the current delay list is 5 and only the minimum value is used to estimate the queuing delay.

E. Measuring Queuing Delay

The queuing delay estimation is simply the difference between an end-to-end delay measurement and the current estimate of base delay. The consumers measure the queue delay for every data packet received after updating the base delay and the current delay using Algorithm 1. If the queuing delay of each packet exceeds the TARGET delay, the consumer will consider the network is congested and reduce the congestion window size according to Algorithm 2. However, if the queuing delay is less than or equal to the delay target, the consumer increases the size of the congestion window. DCP uses a linear controller for increasing and decreasing the consumer congestion window, allowing the protocol to recover quickly from bad delay samples. The increase and decrease of the congestion window is proportional to the difference between the current queuing delay estimate and the TARGET delay. The congestion window in DCP refers to the number of Interests a consumer is allowed to send.

As the queuing delay gets closer to the TARGET delay, the window growth slows down. To compete with concurrent TCP flows fairly, the LEDBAT algorithm maximizes the congestion window growth by 1 per RTT. This happens when the queuing delay estimate is zero. The same approach is used in DCP in the congestion avoidance state, where the maximum window growth is only 1 per RTT as seen in line 7 in Algorithm 2. Also, in LEDBAT, the sender may maintain its congestion window (cwnd) in bytes or in packets. The decrease in the congestion window in DCP happens when the queuing delay estimate exceeds the application's TARGET delay. The decrease in the window is proportional to the difference between the TARGET delay and the current queuing delay. This allows the consumer to decrease its sending rate as a response to potential congestion in the network unlike loss-based congestion control protocol. However, data loss can still occur in DCP. A data loss is considered an indication of strong congestion in the network.

The TARGET delay determines how much queuing delay is tolerated in the network. For example, if the value of the target is set to 0, it means that no delay is tolerated in the network. This may lead to underutilization of the available capacity in the network, because the bottleneck queue will always be empty. A high TARGET delay might increase congestion in the bottleneck and cause packets to be dropped. Therefore, the choice of the TARGET delay is a trade off between these two cases. The LEDBAT algorithm uses a TARGET value of 100 ms or less [14]. The TARGET value is set to 50 ms in our current version of DCP. The dynamic selection of the TARGET value is left for future work.

F. Timeout

If the timeout interval for Interests is too small, unnecessary retransmissions of Interests occur. On the other hand, a timeout interval that is too large results in the consumer retransmitting Interests too late when a data packet or an Interest is dropped. LEDBAT uses a congestion timeout (CTO) as a strong indication of heavy congestion in the bottleneck when a packet get dropped. The LEDBAT CTO approach is based on retransmission timeout (RTO) in TCP.

Even though consumers in DCP are able to indicate congestion in the network before a data packet is dropped at the bottleneck queue, DCP uses CTO as an indication of strong congestion in the network. When an Interest or data packet is dropped, a CTO event is triggered and the consumer reducing its congestion window to 1. The calculation of the CTO is based on the RTT estimation, which is the time it takes to send an Interest and receive the data packet for it.

G. Synchronization

No synchronization between the consumer and the producer or middle nodes is required in DCP if data are retrieved from the same source. This is because measurement errors in delays due to clock offset are canceled out by their difference in the queuing delay estimate.

Even though clock offset errors have no impact on LEDBAT, cock offsets may be critical in an ICN architecture, because Interests may be resolved from caching sites. For example, if two consumers connected to the same router with caching want to retrieve the same data and consumer b starts few seconds after consumer a, consumer b initial requests will be satisfied by the cache at the nearby router. If consumer b receives all the cached content, it joins consumer a in retrieving the data from the producer, This could lead to inconsistent calculation of the one-way delays of data packets if the cache and the producer have different clock offsets. Thus, causing clock offset to show up as a linearly changing error in a time estimate instead of fixed error. Such inconsistency is limited to the amount of delay history maintained in the base delay estimator used by DCP for measuring the one-way delays for data packet. However, a more suitable solution is needed as explained next.

Measurement errors in delays in an ICN can be caused by different clock offsets of different caching sites. This translates into a difference of the clock rate from its true rate and can be seen as a clock skew problem similar to the the skew of LEDBAT's one-way delay estimate. One possible consequence of this issue in DCP occurs when data are retrieved from a new data site with a clock offset that is greater than the previous site clock offset. This results in a low estimate of the queuing delay; however, it also lead to a lower base delay measurement, so it does not lead to unfairness for the consumer. Another possible consequence of the issue happens when data are being retrieved from a new data site with a clock offset that is less than the previous site clock offset. This results in a high

estimate of the queuing delay. This increase can reduce the throughput of a flow, because the base delay will only keep the minimal samples over a time interval. A high change in the relative delay of data packets could be used by the consumer to infer that such change in the delay is due to clock drift in the clock offset, which means data is being be retrieved from a new source. The sender then can compensate by recalculating its base delay estimate.

LEDBAT suggested a correction mechanism for the synchronization problem and related to clock skew. Even though clock skew is usually very small, we believe more research is needed to define the synchronization issue in an ICN architecture. We leave the topic of researching and testing different solutions to the issue as part of our future work.

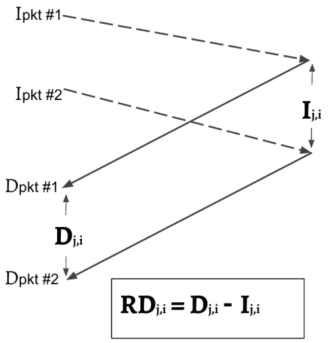


Fig. 2: Transmission of two data packets and the corresponding relative-delay measurements

V. Fairness among DCP Consumers

A. Latecomer's Advantage

LEDBAT's algorithm suffers from the "late-comer advantage," where late flows arriving at a non-empty bottleneck increase their sending window more aggressively than already existing flows due to wrong estimates of the base delay. For example, If one flow arrives at an empty bottleneck it will be able measure the correct base delay of the network, leading to a correct estimate of the queuing delay. However, if a new flow arrives at the bottleneck, it will calculate the queue delay of the first flow as part of its base delay. Leading to wrong estimate of the queuing delay, thus a higher target delay than the first flow. This will result in aggressive increase of the window size of the second flow causing the first flow to back off as it sees the true queuing delay of the bottleneck increase. Eventually, the second flow will start using the entire bottleneck's capacity while shutting the first flow out. There have been several solutions to the late-comer advantage in LEDBAT [9][15]. DCP solution to the late-comer advantage is based on LEDBAT's original approach, by using TCP like slow-start.

A solution to the late-comer advantage is to introduce TCP slow-start at the beginning of DCP flows. The goal is to drain the queue empty by filling it which will likely induces losses on already existing flows, allowing the new comer to measure the correct base delay. Such a mechanism cannot ensure fairness without causing packet drops in other flows. Although such drops are limited, this suggests that a more suitable solution is needed. We leave the topic of evaluating different fairness solution in DCP for our future work. The slow-start mechanism in DCP is similar to the one in TCP. DCP uses it for new consumers joining the network to ensure correct measurement of the base delay. During the slow-start state, consumers double their sending window for every round-trip time until a timeout event is triggered. After a timeout event is triggered, consumers enters the congestion avoidance state. Consumers in the congestion avoidance state follow the the original algorithm in LEDBAT, where the maximum increase of the window size is 1 per RTT.

B. Latecomer's Disadvantage

The aggregation of Interests does not cause unfairness between consumers trying retrieve the same data content in DCP. Even though a new consumer with aggregated Interests will experience less delay than the consumer with initial Interest request. For example, consider two consumers connected to the same router, operating with the same link speed, and wanting to retrieve the same data. Assume that Consumer 2 starting after Consumer 1 before any of the Interests from Consumer 1are satisfied. The Interests from Consumer 2 Interests are aggregated at the router, which could result in Consumer 2 having a low estimate of the queuing delay, because some of the data packets for Consumer 2 may experience less delay than data packets for Consumer 1. However, aggregated Interests of Consumer 2 will also result in a smaller base delay than the base delay for Consumer 1. The same happens when caching takes place, as Consumer 2 experiences a low base delay compared to Consumer 1.

However, there could be a case where a specific caching scenario may cause underutilization of the link's capacity. Using our previous example, if the initial requests by Consumer 2 were satisfied by a close cache resulting from prior requests by consumer 1, Consumer 2 joins Consumer 1 in retrieving the data from the producer after exhausting the cache hits. If consumer 1 leaves the network, consumer 2 Interests will no longer be aggregated and instead they have to traverse all the way to the producer or a more further away cache. This scenario will cause underutilization of the link's capacity for consumer 2. This is because consumer 2 base delay measurements were affected by the close caching site. We identify this is as the "latecomer's disadvantage".

Consumers in DCP overcome this issue by resetting their base delay list and calculating a new base delays whenever they identify contents are being retrieved from a new site. An immediate calculation of the relative delay can help determine whenever contents are being retrieved by a new site. To determine the change in the relative delay value of data packets, we use $RD_{\mathrm{j,i}} = \Delta D_{\mathrm{j,i}} - \Delta I_{\mathrm{j,i}}$, where RD$j, i$ represents the delay experienced by packet j with respect to packet i.

Fig. 2 shows the arrival of two Interests 1 and 2 at the provider and the arrival of two data packets corresponding to Interests 1 and 2 at the consumer. The relative delay value fluctuates from one data packet to another due to congestion in the network. Therefore, it is important to distinguish between a change in the relative delay of data packets due to congestion in the network rather than being the result of data being retrieved from a new site. Because of this fluctuation, any specific relative-delay value could be atypical, and hence an average of the relative delay gives a better estimate of the delay between packets in the network. The mean absolute deviation (MAD) is the average of the absolute deviations between each relative delay value and the average delay. Calculating MAD helps deal with relative delays with negative value. The MAD value is measured for the last 10 relative delays samples and is updated with every new value. Upon obtaining a new value of the relative delay, DCP measures how far the new relative delay is from the average relative delay. If it is higher than the MAD value, then the consumer infers that content is being received from a new site. To ensure fairness between this consumer and existing DCP flows, a consumer should resume slow start whenever the base delay history is reset. as we mentioned earlier. Also, since the value of the relative delay is affected by packet dropout at the bottleneck queue, the relative delay is not measured during slow start.

VI. Performance

We evaluate the performance of DCP under different scenarios using simulation using ndnSIM [7].

A. Basic Bottleneck Configuration

The first experiment shows the protocol performance over a simple network consisting of a single consumer and a single producer scenario. We compare DCP against an end-to-end protocol that behaves in the same way as TCP, where consumers can only infer congestion via a retransmission timeout and use AIMD window control to avoid congestion, such a mechanism is used by most end-to-end protocols in ICN. The simulations last for 60s, where the consumer will be issuing Interests non stop as long as its remitted by it congestion control protocol. The topology of the network is a singe path of four nodes with a single consumer at one end sending Interests for a content served at the other end. The size of the data packet of the content is fixed at 1024 Byte with Interest size of 24 Byte. The router's queue is set to be equal 60 packets. The bandwidth delay product of the network is around 133 packets including the size of the queue. Since ndnsim ignores the processing delay per packet, the only constant end-to-end delay in the network would be the propagation delay and the transmission delay. In our test we would be using a TARGET delay of 50 ms, based on the network's parameters, this will allow enough packets to be build up in the queue without being dropped.

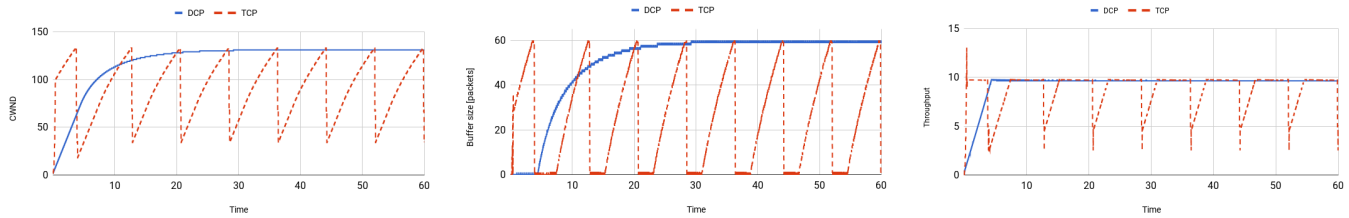
Fig. 3b shows the size of the bottleneck's buffer for the whole duration of the simulation. Packets dropout is the only way to detect congestion in the network for TCP. Therefore, a consumer keeps increasing its sending rate, until it cause a buffer overflow which will result in a timeout event to be triggered. Thus, a consumer reduces its sending rate by halving is congestion window. This process drains the queue empty for few seconds until it get filled again and the same cycle repeat. However, DCP (slow-start disabled) allows the consumer to increase its congestion window as long as the queuing delay is less than the TARGET delay. As soon as queue builds, the consumer slow down its sending rate. Once the queue delay reaches the TARGET delay in both protocols, the consumer halts the window growth. This demonstrates the main strength of DCP as a congestion control for ICN by transmitting Interests at the bandwidth of the connection, without congesting the network and without overflowing the bottleneck's queue. The congestion window reaches a steady state value of 133 Interests as it can be seen in Fig. 3a. DCP maintains this steady state value for the duration of the connection with no occurrence of timeout since slow-start was disabled. Fig. 3c shows the throughput for both protocols for the duration of the simulation. With a bottleneck link of speed 10 Mbps the consumer in DCP was able to almost utilize the link's capacity with an average throughput around 9.3 Mbps. As a result of the multiple packets dropout in the TCP based protocol, the consumer was only able to achieve an average throughput of 8.7 Mbps.

B. Caching

The topology of the network used to evaluate performance with caching consists of two consumers retrieving the same data. However, consumer 1 stops after retrieving around 100 data packets. Since caching is used, a consumer 2 initial requests are satisfied by the closest cache. The cache starts retrieving the data from the producer after consumer 2 exhausts its capacity. Fig. 4 shows the increase of the congestion window for both LEDBAT and DCP for consumer 2. It is clear that a pure implementation of LEDBAT in ICN architecture suffer from the "latecomer disadvantage". This is because consumer 2 initial requests were satisfied by a close cache, this resulted in a small base delay. Therefore, once consumer 2 start retrieving the data from the producer it will experience a high queuing delay even thought the network is empty. However, DCP (slow-start disabled) overcome this issue by using relative delay measurements of data packets. Consumer 2 conclude that an increase in the relative delay value if due to data being retrieved from a new distance caching site or producer. Resulting in consumer 2 resetting its base delay history.

C. Multi-Flow Scenario

We evaluate LEDBAT's "latecomer advantage"in NDN and how DCP with slow-start overcome this issue. The topology used in the scenario consists of two consumers (Consumer 1 and Consumer 2) and two producers (Producer 1 and Producer 2). The two consumers request different content files of a fixed data packet size of 1024 Bytes, each one is hosted by one of the producers. The queue size is set to 120 to allow packets not to be dropped due to buffer overflow. To simulate the



(a) Window Size **(b)** Buffer Size **(c)** Throughput

Fig. 3: Single-flow scenario

latecomer advantage in ICN, the second consumer should start sending Interests when the queue is starting to build up and the first consumer begin to experience an increase in current delay. Based on our network parameter, that should be when the window size of the first consumer is greater than 73, which is equal to the bandwidth delay product of the network. This way a later consumer account the queuing delay of the first consumer as its base delay measurement, leading to a higher target delay. Which lead to unfairness as the first consumer will have to back off as it can sense true increase in the queue delay of the bottleneck. The simulation for this scenario runs for 60 seconds.

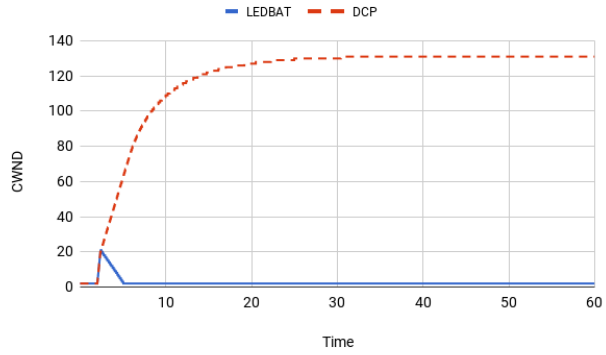


Fig. 4: DCP and LEDBAT performance under caching

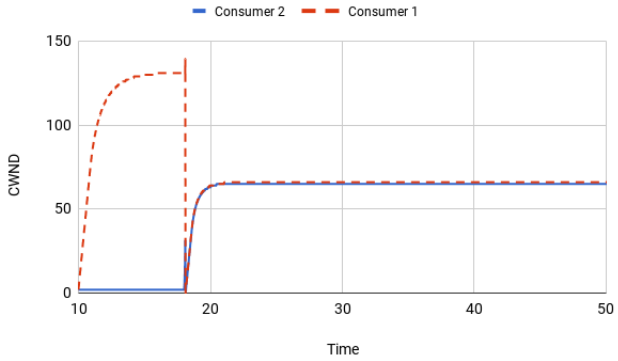


Fig. 5: DCP performance under latecomer advantage

Fig. 5 shows the results for DCP with slow-start enabled. Once Consumer 2 joins the network it will go into the slow-start state. During this state Consumer 2 double its sending window for every round-trip time until a timeout event is triggered. Such a mechanism drains the queue empty, allowing Consumer 2 to measure the correct base delay and hence achieving fairness between the two consumers.

ACKNOWLEDGMENTS

This work was supported in part by the Jack Baskin Chair of Computer Engineering at UC Santa Cruz.

VII. CONCLUSION

We introduced DCP, which adopts the congestion estimation algorithm of LEDBAT in the context of an ICN. DCP is based on a receiver driven, window-based approach to congestion control using measurements of delays along the forwarding path. We analyzed the known "latecomer advantage" problem that exists in LEDBAT for ICN and how slow-start in DCP can enable fairness. We also analyzed a new issue raised by LEDBAT algorithm in ICN due to caching we call it "latecomer disadvantage" and showed how the use of relative measurements delay in DCP allow consumers to overcome this problem.

REFERENCES

[1] Community ICN,https://wiki.fd.io/view/Cicn, March 2, 2005.
[2] Named Data Networking, https://named-data.net.
[3] G. Carofiglio, et al. ICP: Design and Evaluation of an Interest Control Protocol for Content-Centric Networking. In IEEE NOMEN 12, 2012.
[4] S. Arianfar, et al. "Contug: A receiverdriven transport protocol for content-centric networks," in IEEE ICNP 2010 (Poster session), 2010.
[5] N. Rozhnova, et al. "An Improved Hop-by-hop Interest Shaper for Congestion Control in Named Data Networking," ICN 2013
[6] P. Karn and C. Partridge. Improving round-trip time estimates in reliable transport protocols. In Computer Communication Review, volume 17 No. 5, pages 2 – 7, August 1987.
[7] A. Afanasyev, et al. ndnSIM: NDN simulator for NS-3. Technical Report NDN-0005. (NDNsim)
[8] D. Rossi, C. Testa, S. Valenti, and L. Muscariello, LEDBAT: The New BitTorrent Congestion Control Protocol. IEEE ICCCN '10, Aug. 2010.
[9] D. Rossi, et al. News from The Internet Congestion Control World. ArXiv e-prints, Aug. 2009.
[10] Y. Wang Caching, Routing and Congestion Control in a Future Information-Centric Internet
[11] C. Yi, et al., A Case for Stateful Forwarding Plane. Technical Report NDN-0002, July 2012.
[12] S. Braun, et al. An empirical study of receiver-based AIMD flow-control for CCN. IEEE ICCCN, 2013
[13] K. Schneider, et al. A Practical Congestion Control Scheme for Named Data Networking In ICN , 2016
[14] S. Shalunov. Low Extra Delay Background Transport (LEDBAT).IETF Draft, Mar. 2009.
[15] G. Carofiglio, et al., The Quest for LEDBAT Fairness. Jun, 2010.