Inferring and Controlling Congestion in CCN via the Pending Interest Table Occupancy

Amuda James Abu, Brahim Bensaou and Ahmed M. Abdelmoniem
The Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Email:{ajabu, brahim, amas}@cse.ust.hk

***Abstract*—Despite the improvements brought about by content-centric networking's (CCN) in-network caching and interest aggregation, congestion can still take place in such networks due to the dominance of non-reusable content, high cache churn, high delay variations, premature timeouts and interest retransmission. This becomes even more dramatic when multi-path routing is adopted. Identifying that at a given node in CCN, the Pending Interest Table (PIT) occupancy can give a good estimate of the data workload to arrive to the node in the near future, we propose in this paper a novel mechanism to control congestion in CCN based on this idea. Our mechanism uses the average occupancy of the PIT to estimate the anticipated data packet transmission queue length and sends explicit congestion notification signals to the content requesters to reduce their interest sending rates when such anticipated queue size exceeds a threshold. We demonstrate the effectiveness of our proposed mechanism via ns-3 simulation.**

I. Introduction

In-network caching and interest aggregation are among the key features of Content-centric networking (CCN) [1] and other Information-centric networking (ICN) proposals such as Named-Data Networking (NDN) [2]. With these features, a great deal of redundant traffic is filtered out from the network thus improving the performance of the network considerably. Nonetheless, due to various reasons, such as the dominance of non-reusable (one-timer) content in the network, the popularity distribution of traffic and the huge ratio of the universal content available to the cache sizes in the nodes, congestion can still take place in such networks. This motivates the design of a new clean-slate congestion control mechanism to take into account the unique characteristics of CCN/ICN.

A CCN node receives at most one data packet for every interest packet forwarded upstream. The data packet is then sent downstream at most one copy per interface. If the arrival rate of data packets exceeds the transmission capacity of the downlink, the queue begins to build up and eventually packets are dropped leading to timeout and retransmission of interest packets. To regulate the rate of arrival of new data chunks, one can regulate the rate of arrival of interests. A requester relies on timer expiration to infer congestion in the network however, this approach can be slow in reacting to congestion [3], [4]. To address this problem more effectively, intermediate nodes could be co-opted to notify requesters of congestion by explicitly sending a congestion control packet (aka NACK) [5]. However, sending NACKs downstream is not without drawbacks as it may require message prioritization and incurs additional overhead [6]. Approaches that do not require such prioritization and incur no extra overhead are desirable.

The Pending Interest Table (PIT) in CCN is a data structure that keeps track of all interests forwarded upstream and thus influences directly the occupancy of the data packet transmission buffer. In addition, it can represent a bottleneck for the network when the rate at which PIT entries are created is greater than the rate at which the entries are consumed [7], [8]. Therefore, congestion control mechanisms that are designed to efficiently manage the occupancy of the data packet transmission buffer should take into account the occupancy of the PIT as well. However, existing mechanisms [5], [9], [10], [11] for managing congestion in CCN/ICN do not consider the occupancy of the PIT, a vital information that can be used to estimate the anticipated occupancy of the data transmission buffer.

To this end, we propose in this paper a novel congestion control mechanism for CCN that uses the occupancy of the PIT as a good estimator of the additional data to be queued in the data packet queue in the near future. More specifically, our PIT-based congestion control mechanism takes into account the occupancy of the PIT in predicting the data buffer occupancy in the next RTT at each intermediate node. These latter, convey back explicit congestion notification to the requesters to slow down their interest sending rates whenever the anticipated queue size exceeds a threshold. To complement this, we also propose an interest rate controller that forces the requesters to reduce their interest sending rates upon the receipt of congestion notifications. We demonstrate the effectiveness of our proposed mechanism via simulation experiments in ns-3.

The rest of this paper is structured as follows: Our system description is given in Section II. We provide the rationale behind our congestion control protocol in Section III while Section IV entails the description of our proposed mechanism including its performance evaluation and analysis in Section V. We review related works in Section VI and draw our conclusion in Section VII.

II. System Description

In this section, we give a brief description of CCN and further explain how the occupancy of the PIT affects the data packet transmission queue.

DOI 10.1109/LCN.2016.72

A. *Content-centric networking*

Contents in CCN are divided into data chunks and each chunk is singly identified using a hierarchical naming convention [1]. Communication in CCN is initiated by the requesters of the data. More specifically, a receiver requests for chunks by sending interest packets. Each interest contains the identifier of the requested chunk. The identifier is used in routing an interest packet towards the location of the data chunk, then using the reverse path traversed by the interest packet, the chunk is returned to the requester [12]. To improve the network performance, every CCN node caches the data chunk to serve future request for the same chunk if needed.

B. *PIT occupancy and data transmission buffer*

In a CCN PIT, an entry is created for every newly arrived interest that experiences both cache and PIT misses before the interest is forwarded upstream [12], [1]. While the arrival of the requested data chunk consumes the pending interest from the PIT before the data chunk is forwarded downstream. In the CCN progressive deployment plans, it is suggested that the PIT and the cache be elevated from a router's buffer, as a result both the PIT and the cache are limited in size can become bottlenecks that affect the performance of the system. In particular, when the upstream network is congested, data chunks may take longer to return, leading to an increased holding time of each PIT entry in the PIT and possibly PIT blocking as discussed in our previous work [7]. PIT blocking can lead to an increased load on the PIT. In addition, when the PIT occupancy rate (i.e., number of PIT entries in the PIT per unit time) increases, it may lead to an increased workload on the the data buffer that exceeds the node's forwarding capacity. In this case congestion can take place in the data buffer.

III. Protocol Design Rationale

One can imagine a congestion control mechanism similar to TCP congestion control where interest packets are substituted for Data and data chunk are instantiated for ACKs. Nevertheless in such system there is no distinction between congestion in the PIT and congestion in the router buffer. In addition caching would have a dramatic impact on TCP timeout due to the volatility of the RTT estimates. As a result, to take into account the peculiarities of CCN, it is not sufficient to control congestion in the traditional manner. We believe that for a CCN congestion control mechanism to be effective it must take into account the occupancy of the PIT, a key component of CCN that drives the number of data packets that can be retrieved from upstream nodes. Given the dependency of the data packet transmission buffer occupancy on the PIT occupancy, the current occupancy of the PIT can be used to estimate the anticipated occupancy of the buffer knowing the current buffer occupancy. Avoiding congestion before it takes place is always desirable in communication networks. Therefore, the PIT occupancy can be used to estimate the anticipated queue length of the data packet transmission buffer in the next RTT.

In addition, controlling congestion in a network at the traffic sources is considered a good design choice as adopted by some existing congestion control proposals for IP networks and CCN. However, such end-host proposals rely on a timer expiration or/and receipt of three duplicate ACKs (for TCP congestion control flavours) which may lead to a slow reaction to congestion in the network. Thanks to proposals such as TCP/AQM with ECN marking that signal congestion to end-hosts before timer expiration. A number of recently proposed congestion control protocols for CCN/ICN recommend to use congestion control packet (NACK) to signal congestion to consumers [5], [10]. Another proposal uses a data packet, excluding its payload, to signal congestion [13]. While we believe that it is desirable to explicitly notify CCN consumers of network congestion before timer expiration, explicitly signalling congestion to traffic sources should not incur additional network overhead or require message prioritization [6]. Similarly in our proposed congestion control protocol, instead of waiting for congestion to take place, routers signal imminent congestion to traffic sources, but differ from existing works by marking an option field in the data packet header without the need to exclude its payload. We consider this as a good design choice.

Contents in CCN/ICN are divided into chunks. The number of chunks per content depends on the size of the content as well as the chunk size. Contents with large number of chunks tend to occupy more entries in the PIT than content with small number of chunks. Suppose we adopt a random early drop mechanism to relieve congestion in the PIT, and need to erase an entry in order to accommodate a new request when the PIT is full or a target PIT occupancy has been reached, an entry for a content that occupies the largest number of PIT entries will be the best candidate to ensure fairness. To this end, we propose to use a mechanism to determine such elephant contents.

In summary, our protocol design is inspired by the well-proven TCP/AQM (e.g. RED) with ECN marking. However, it is bundled with additional functionalities such as:

i Penalizing contents that occupy a large number of entries in the PIT

ii Using PIT occupancy to estimate the anticipated occupancy of the router's packet transmission buffer in the next RTT.

iii Reaction to imminent congestion before it takes place.

Next, we present our proposed congestion control protocol for CCN detailing first how to achieve functionalities i–iii. We later give the full description of the protocol.

IV. Proposed Congestion Control Protocol

In the conventional TCP/IP network, congestion can happen in a transmission buffer if the transmission link capacity is a bottleneck in the network. However, with the evolution of content-centric networking featuring a pending interest table that keeps track of interests for yet-to-return data, the PIT also is a potential bottleneck thus limiting the number of interests that can be forwarded upstream and consequently the

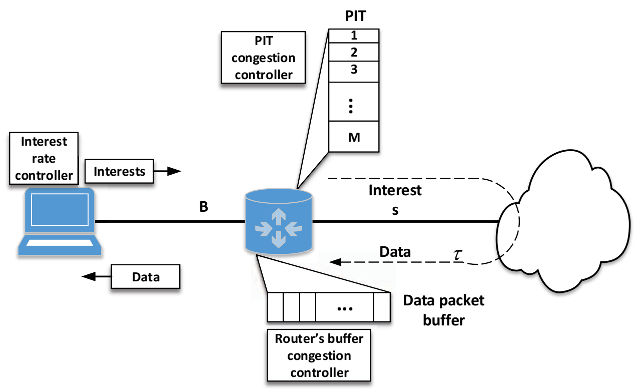


Fig. 1: A router's PIT and transmission buffer for data in a simple CCN network

number of data. Our proposed congestion control mechanism operates at two levels in a given CCN node: First, it controls congestion at the PIT, and secondly at the router's buffer. When congestion is detected either in the PIT or in the router's buffer, CCN receivers are notified before their retransmission timers expire. Upon receiving the notifications, they regulate their sending rates accordingly. See Fig. 1.

A. A modified data packet header for congestion notification

Each returned data packet has a new 1-bit field in its header to signal congestion to CCN receivers. A value of 0 means no congestion while a value of 1 implies imminent congestion. The 1-bit field is filled in by intermediate nodes depending on the current value of the field and the congestion status at the nodes. Note that no downstream node can modify the value in the field if its current value is 1.

When a CCN consumer receives a packet that is marked, it reacts accordingly by reducing its interest sending rate once per RTT.

B. Fairness controller

In a CCN PIT, a data chunk consumes at most one entry. Different contents can have different number of chunks. For example, given a fixed chunk size, there is a huge difference in the number of chunks of a 3GB movie download and a web request. Therefore, requests for the movie (elephant content) tend to occupy more entries in the PIT than for the web request (mice content). When congestion happens in the PIT, we believe that the elephant contents should be penalized more severely than the mice contents. To this end, we propose a mechanism that mark data packet with a probability when the PIT occupancy is between a minimum and maximum thresholds at a given node. Our mechanism follows a RED-like scheme to monitor the PIT occupancy and calculate the data packet marking probability $\hat{p}$.

C. Congestion notification via data packet marking

In section IV-A we propose to mark an Optional field in the header of the data packet to be forwarded downstream via a given interface if congestion is imminent either in the PIT or transmission buffer. With this technique, we avoid any additional traffic into the network and network resource wastage unlike the works in [5] and [13], respectively.

To notify a CCN receiver of congestion, we set the 1-bit Option field (OF) in a CCN data packet header to 1. It is set to 0 by default at the node that satisfies the request. Every downstream node that receives a data packet checks the 1-bit OF and infers congestion in the upstream network. If OF is 1 then the node can attempt other available outgoing faces in the forwarding information table (FIB). This request path diversification on congestion tends to offload upstream nodes in the future.

D. PIT congestion controller

Given a PIT of size M we define two PIT occupancy thresholds similar to RED, ρ_{min} and ρ_{max}. A CCN node updates its current PIT occupancy $M(t)$ on every entry creation and deletion in the PIT and the average PIT occupancy $\tilde{M}$ using exponential moving average.

$$\tilde{M} = (1-\omega)\tilde{M} + \omega M(t) \tag{1}$$

The detailed algorithm is shown in Algorithms 1.

On the arrival of an interest packet at a CCN node via interface j and the requested data chunk is found in cache, the interest is consumed by the node. On a cache miss, the PIT is checked for pending interest with the same name. If found, we do a normal CCN packet processing. If no matching entry is found in the PIT then a PIT entry is created and the interest is forwarded upstream. In this case we update $M(t)$ and $\tilde{M}$.

On the arrival of a data packet while $\tilde{M} \geq \rho_{max}$ we mark the data packet for sure. If the average PIT occupancy is between ρ_{min} and ρ_{max}, we use RED-like scheme to compute the marking probability $\hat{p}$ and mark the data packet with the computed probability. Similarly, we update $M(t)$ and $\tilde{M}$.

E. Router's buffer congestion controller

Suppose an interface j has a transmission buffer and at any time t the buffer occupancy is observed to be $Q_j(t)$ packets. Similarly, the PIT occupancy on the average is denoted as $\tilde{M}$ as in Equation 1. Given the PIT occupancy, we estimate the anticipated occupancy of the transmission buffer of interface j in the next τ seconds as $\tilde{Q}_j(t+\tau)$

$$\tilde{Q}_j(t+\tau) = Q_j(t) + \kappa_j \tilde{M}; \tag{2}$$

where $\tilde{Q}_j(t+\tau)$ is defined as the anticipated number of packets in the buffer. Note that $\tilde{Q}_j(t+\tau)$ is a first order estimator of the persistent queue length as it ignores both the packets transmitted in interval of time τ and the extra PIT traffic that would arrive during τ from downstream nodes. Also, for each node we have κ defined as a set of ratios of the number of interests received from a given interface to the total number of interests received from all interfaces. $\kappa = \{\kappa_1, \ldots, \kappa_f\}$ where f is the number of transmission interfaces at a node. κ_j indexed at j is the ratio of interests received from interface j to the total interests received from all f interfaces.

Having estimated the anticipated number of packets in the buffer $\tilde{Q}_j(t+\tau)$ for interface j, we then compare $\tilde{Q}_j(t+\tau)$

Algorithm 1: PIT congestion controller

```
Input: $M$, $\rho_{min}$, $\rho_{max}$
On Interest Arrival: via interface $j$
if cache hit or PIT hit then
    normal CCN processing
else
    Create PIT entry
    Forward interest upstream
    $M = M + 1$
    $\tilde{M} = (1-\omega)\tilde{M} + \omega M$
end
On PIT entry timeout:
$M = M - 1$
$\tilde{M} = (1-\omega)\tilde{M} + \omega M$
normal CCN processing
On arrival of Data:
if PIT hit then
    $M = M - 1$
    $\tilde{M} = (1-\omega)\tilde{M} + \omega M$
    if ($\tilde{M} \geq \rho_{max}$) then
        Mark data packet with probability
        equal to 1
    else if ($\rho_{min} \leq M \leq \rho_{max}$) then
        Compute $\hat{p}$ using RED
        Mark data packet with probability $\hat{p}$
    end
end
normal CCN processing
```

to a threshold $\gamma_Q^j B_j$ and take decisions based on the outcome of the comparison, where B_j is the buffer size of interface j. If $\tilde{Q}_j(t+\tau) >= \gamma_Q^j B_j$ then the flag $flagI_j$ associated with interface j is set, otherwise $flagI_j$ is unset. Every data packet that is transmitted via interface j is marked if $flagI_j$ is set. Conversely, no data packet is marked if $flagI_j$ is unset. See Algorithm 2 for details.

F. Receiver's congestion controller

At the receiver, when a CCN consumer receives a data packet, the CCN receiver controller checks the 1-bit OF; if its value is 1 then congestion is inferred otherwise a congestion-free network is assumed.

Given a data content C divided into chunks, a CCN consumer generates an interest packet for each chunk. The interest is sent out to fetch the requested chunk from the network (caching nodes and/or content producer). To avoid overloading the network with interest packets and consequently data packet, we propose to have an interest controller residing on every node that generates interests for contents. The controller includes a transport mechanism for sending interests. It uses a TCP-like congestion control mechanism to regulate the amount of interests that can be sent upstream. Details of the interest controller are shown in Algorithm 3.

To avoid waiting indefinitely for data, the controller in the CCN router maintains a timer for every interest sent. The controller can infer congestion upon the expiration of this timer, in which case, the interest window is cut by half. Note that congestion window never goes below 1 packet.

Algorithm 2: Router's buffer congestion controller

```
Input: $\tilde{M}$, $Q_j$, $\gamma_Q$, $\boldsymbol{\kappa}$
On Interest Arrival: via interface $j$
if cache hit then
    normal CCN processing
    if $flagI_j$ then
        Mark data packet via interface $j$
    end
else
    if PIT hit then
        normal CCN processing
    else
        normal CCN processing
        update $\boldsymbol{\kappa}$
        $\tilde{Q}_j(t+\tau) = Q_j(t) + \kappa_j \tilde{M}$
        if $\tilde{Q}(t+\tau) >= \gamma_Q^j B_j$ then
            set $flagI_j$
        else
            unset $flagI_j$
        end
    end
end
On arrival of Data:
if PIT hit then
    normal CCN processing
    update $\boldsymbol{\kappa}$
    if $flagI_j$ then
        Mark outgoing data packet via
        interface $j$
    end
else
    normal CCN processing
end
```

Due to high delay variability caused by in-network caching in CCN, TCP-like RTO estimation has been shown to cause unnecessary retransmission [4], [3], [14] thus degrading network performance. In view of this, we follow a different approach in estimating the RTO. Our approach is informed by our previous work in [7] that records the maximum RTT observed over a given period of time. See [7] for the detailed algorithm. In addition, we set the RTO to be slightly larger than the observed maximum value to avoid unnecessary retransmission.

Upon receiving a data packet with the OF bit marked, the controller infers congestion in the network and decreases its sending rate accordingly not more than once in an RTT. We adopt AIMD to increase and decrease the interest sending rate at the consumer. Our approach is different in the value of the multiplicative decrease factor β when a CCN consumer receives a marked packet. We set $\beta = 0.1$ instead of 0.5. This is because a data packet is marked as a result of imminent congestion (congestion has not taken place, just anticipated). Other values between 0.1 and 0.5 are plausible but too close to 0.5 could impair the network performance. Also negative powers of 2 such as (0.125) are more interesting to avoid floating point operations.

Algorithm 3: CCN consumer interest controller

```
1 Initialization: $W = W_{init} = 1$
  Input: $\beta$
  /* Similar to TCP congestion control
     algorithm except in decreasing the
     congestion window                        */
2 On Data:
3 if Data successfully received and data packet marked then
4     extract the value for $\omega$
      /* Once in 1 RTT                        */
5     $W = W - \beta W$
6 end
7 On Timeout:
8 $W = \max(W_{init},\ 0.5W)$
9 retransmit timeout interest
```

G. Complexity analysis

In our scheme, we propose to use an option field in the data packet header to explicitly notify content requesters of congestion. With this approach, no extra overhead cost is introduced. In the proposed scheme, the requesters check the Option field of the received data if it is marked, resulting in an $\mathcal{O}(1)$ operation.

Our PIT congestion controller maintains three variables, M, $\tilde{M}$ and $\tilde{\rho}$, see Algorithm 1, for all interest packets received. The variables are updated each time an entry is created in the PIT. Note that in a normal CCN PIT operation, an interest aggregation or PIT entry creation is done for every interest packet that arrives at the PIT. Updating the three variables is an $\mathcal{O}(1)$ operation.

In Algorithm 2, our router's buffer congestion controller maintains a fixed number of variables similar to Algorithm 1 except for κ and $flagI_j$ whose space complexity depends on the number of transmission interfaces through which interests have been received. Note that most routers in CCN would probably have a number of transmission interfaces that is at most 2 or 3 order of magnitude. Maintaining as many variables would not pose any significant memory efficiency problem in the router. In addition, we update κ and $flagI_j$ for each interest arrival at the PIT. Given that both κ and $flagI_j$ are indexed by the interface ID obtained when an interest arrives at the PIT, updating the two variables is also an $\mathcal{O}(1)$ operation.

Finally, from the code perspective, the implementation of our proposed scheme may seem to be not efficient based on typical IP routers. In fact, these routers cannot even match the compute power demand of CCN itself which has led to the design of specialized routers (e.g., by Cisco and Huawei) for CCN. As discussed above, in such routers our proposed scheme should not pose any efficiency issue.

V. Performance Evaluation and Analysis

This section presents a performance evaluation and analysis of our proposed mechanism via simulation in ns-3. We also demonstrate the benefits of a network assisted congestion control mechanism by a comparative performance study of our proposed mechanism to a *TCP*[1] without congestion feedback from intermediate nodes in the network.

A. Simulation description

To study the impact of congestion on the performance of our proposed scheme in a single shared bottleneck, we consider a dumbbell topology consisting of 6 nodes. Two access routers $ar1$ and $ar2$ receive requests for contents from many users (10,000 users). We believe this number of users is large enough to evaluate our scheme. Users' content requests follow a Zipf probability distribution with the skewness parameter α. In the case of cache misses, all chunk requests from $ar1$ and $ar2$ can only be satisfied by content producers $p1$ and $p2$, respectively. Two core routers, $cr1$ and $cr2$ connect $ar1$ to $p1$ and $ar2$ to $p2$. We believe that this topology is simple and sufficient to capture the impact of caching, aggregation and PIT occupancy on the performance of our proposed mechanism. We do not consider complex topologies and bidirectional flows of interest packets because the arrivals of interest packets on one direction are capable of causing congestion in the reverse path.

By default $ar1$ and $ar2$ have relatively small cache sizes and thus requests are not satisfied by these nodes. Requests are forwarded using shortest path routing and content request arrivals at each access router follow a Poisson process with mean arrival rate λ. Requests for the same content have the same interest lifetime. We use ProWGen [15] to generate our workloads. ProWGen is a widely used synthetic web proxy workload generation tool for web requests in the Internet.

Note that large queueing delay and packet drops are two manifestations of congestion in a network. They can affect negatively the time it takes a content requester to fetch a given content from the network, the rate of packet retransmission and chunk delay variance (jitter). To see the extent to which our scheme improves content requesters' quality of experience while keeping the network in a normally operating and stable state we evaluate the performance of our scheme by considering the following metrics:

- *Content download time* defined as the time from when the first request is sent to the time the last chunk for the content is received.
- *squared coefficient of variation* (SCV) of the chunk delay per content;
- *number of interest retransmission* (every 100s);
- *data chunk packet loss rate*;
- *number of entries in the PIT* indicating the number of pending interest entries in the PIT;
- *queue length at the bottleneck link.*

Node $cr1$ receives interests from two interfaces leading to entries in the PIT. As such, we present results for the number of entries in the PIT for only $cr1$.

[1]Similar to a TCP congestion control that relies on timeout to infer congestion in the network. RTO estimation and reaction to timeout is the same as in our proposed mechanism

B. Simulation setup

Our proposed congestion control mechanism was implemented in ns-3 network simulator using the ndn/ccn ns-3 modules [2]. To use the workload generated by ProwGen in ns-3, we developed a custom application module. The scenario described in Section V-A was simulated in ns-3.

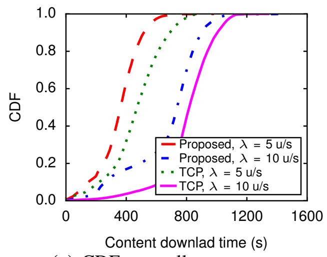
Each of the links connecting $ar1$ to $cr1$ and $ar2$ to $cr1$ has a capacity of 1Gbps and a propagation delay of 10ms while for $cr2$ to $p1$ and $cr2$ to $p2$ it has a capacity of 1Gbps and a propagation delay of 20ms. To make the link connecting $cr1$ to $cr2$ a shared bottleneck link we set the link capacity and propagation delay as 0.5Gbps and 70ms, respectively. The size of each returned data chunk is 1500 bytes. To make each of the links busy 100% of the time when congestion happens we set buffer sizes in each node to the bandwidth delay product and the default size of the PIT is 20000 entries. To see how the size of PIT impact the network performance we consider a PIT size less than the buffer size. We set $\gamma_Q = 1.0$, $\rho_{min} = 0.75P$ and $\rho_{max} = 0.95P$ where P is the PIT size. For the value of $\hat{M}$ to capture the current changes in the occupancy of the PIT we set $\omega = 0.125$.

For our workload, we use $\alpha = 0.95$ (default). We also consider $\alpha = 0.65$. These values are within the recommended values in the literature [16]. There are 60% one-timers and 40% unique contents. The mean file size for contents is 7000 chunks. We generated 10,000 content requests (not at the chunk level). Note that the number of data chunks per content depends on the file size of the content. For in-network caching, we use LRU replacement policy using the default ubiquitous caching policy of CCN and the cache sizes considered in each node are 1%, 0.25% and 0.025% of the content universe . In this work we set the interest lifetime and the PIT entry lifetime to the estimated RTO.

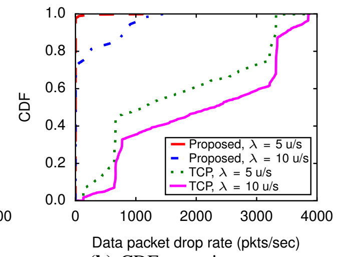
We ran each simulation 100 times. For lightly loaded networks we set $\lambda = 5$ users per second while for heavily loaded networks $\lambda = 10$ users per second. Simulations ran for the entire time to download all the contents (10,000). We present the CDF over all the contents and over simulation time. We hereafter refer to our proposed interest rate controller without congestion feedback from the network as *TCP* and with congestion feedback from the network as *Proposed*.

C. Impact of traffic load

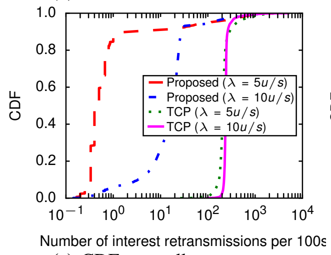
Fig. 2 shows the distributions of the content download time Fig. 2a, data chunk drop rates Fig. 2b, number of interest retransmissions per 100s Fig. 2c, SCV of the chunk delay Fig. 2d, bottleneck queue occupancy Fig. 2e and PIT occupancy Fig. 2f with different average arrival rate of users at each of the access routers in the absence of in-network caching. The results shown in Fig. 2a reveal that about 60% of the contents have finished their downloads in less than 400s using Proposed while the same percentage of contents requires about 500s using TCP in the presence of a lightly loaded network. This is because of the higher data chunk drop rates experienced by TCP than the drop rates experienced by Proposed as shown in Fig. 2b, leading to more



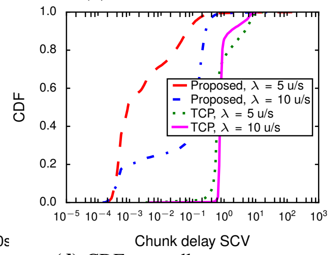
(a) CDF over all contents



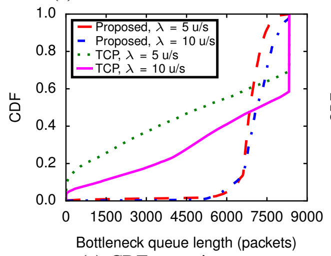
(b) CDF over time



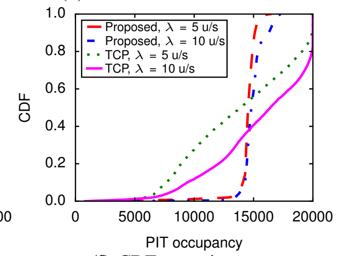
(c) CDF over all contents



(d) CDF over all contents



(e) CDF over time



(f) CDF over time

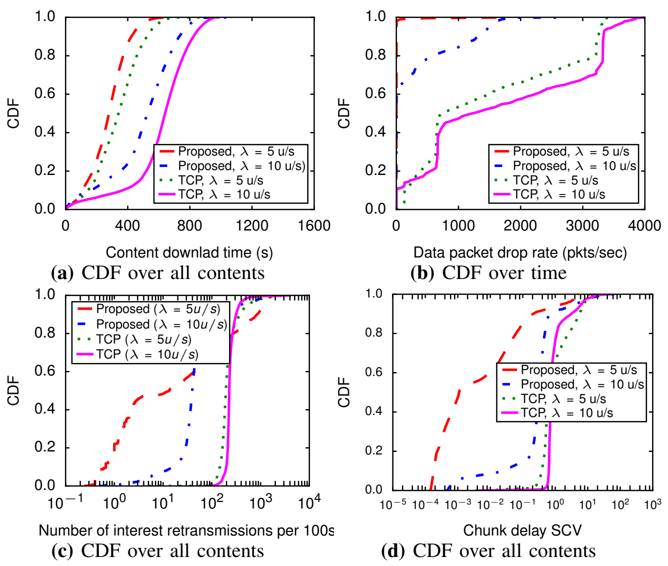
Fig. 2: Impact of network traffic load without caching

interest retransmissions by majority of the content requesters, see Fig. 2c.

Increasing the average arrival rate to 10 users per second can lead to more congestion as evident by the increase in download time from 400s to about 800s for downloading the same amount of contents using Proposed compared to TCP which achieves about 900s to download the same amount of contents. In Fig. 2d, we observe that the SCV for chunk delay is less than 1 for all the contents using Proposed unlike TCP that achieves about 20% of the contents experiencing SCV ≥ 1. This suggests that, using Proposed, the distribution of the chunk delays experienced for downloading a content are considered low-variance.

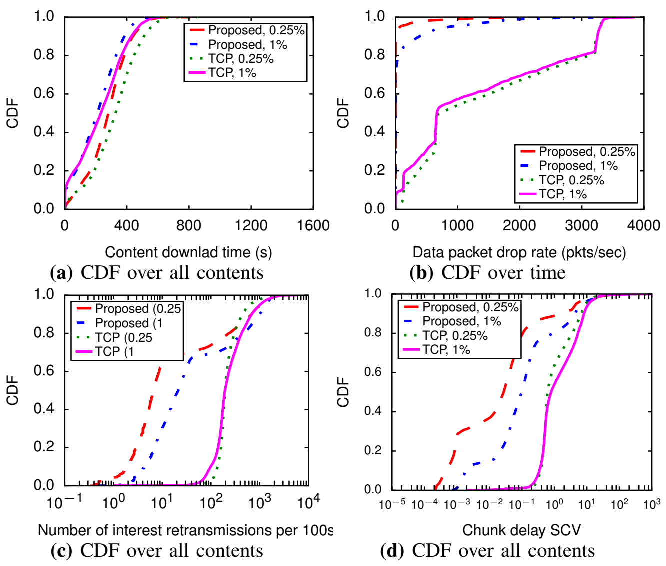
The improved performance achieved by Proposed over TCP is partly due to the bottleneck buffer and the PIT not becoming full in a significant fraction of the time unlike TCP in which the buffer is full in about 30% of the time as shown in Fig. 2e and the PIT in about 20% of the time as shown in Fig. 2f, respectively. Next, we show how the presence of in-network caching can affect the performance of our proposed congestion mechanism.

With in-network caching, we observe a decrease in the content download time from 400s to 300s to fetch about 60% of the contents using Proposed in Fig. 3a. TCP also benefits from caching as the content download time is observed to decrease from about 900s to 700s to download 60% of contents. However, a less percentage of the contents



(a) CDF over all contents **(b)** CDF over time **(c)** CDF over all contents **(d)** CDF over all contents

Fig. 3: Impact of network traffic load with caching



(a) CDF over all contents **(b)** CDF over time **(c)** CDF over all contents **(d)** CDF over all contents

Fig. 4: Impact of caching

experience nearly zero packet drop (Fig. 3b), few number of interest retransmission (Fig. 3c) and less-than-one SCV (Fig. 3d) than when there is no caching using Proposed. This is because some of the chunk requests are satisfied from in-network caches thus reducing the number of hops which in turn increases the rate at which the congestion window is increased especially in slowstart phase causing more chunk drops, interest retransmission and higher chunk delay variance.

We do not observe significant performance improvements w.r.t. the bottleneck buffer and PIT occupancies. The results are not included due to page limit.

D. *Impact of caching*

In this section we vary the cache size per node in the network when the average user arrival rate is 5 users/s. We consider cache sizes of 1%, 0.25% and 0.025% of the universe contents and Fig. 4 shows the distributions of the content download time (Fig. 4a), data chunk drop rate (Fig. 4b), number of interest retransmissions (Fig. 4c) and SCV of chunk delay (Fig. 4d). We do not include results for 0.025% of universe content as no significant performance gain over the scenario without caching was observed.

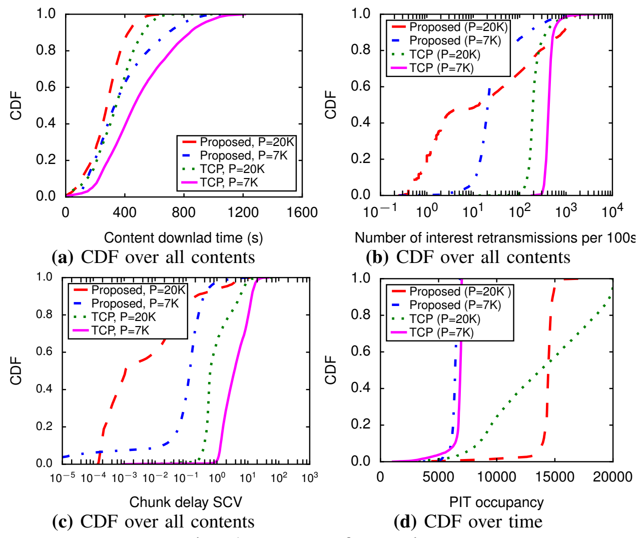
Fig. 4a presents results showing that both Proposed and TCP benefit from increased cache size as evident by downloading about 40% of the contents in less than 200s with 1% of universe contents cache size compared to downloading the same amount of contents in about 250s with 0.25%. This is achieved at the expense of a reduced percentage of contents (from 90% to 80% in Fig. 4b) that experience nearly zero data chunk drop rate, (from 60% to 25% in Fig. 4c) about 9 interest retransmissions every 100s, (from about 80% to 50% in Fig. 4d) about 0.1 SCV) using Proposed. TCP also degrades in performance but it is less significant. The improved download time is due to the fact that more requests are satisfied from in-network caches as the size of the cache increases.

E. *Impact of PIT size*

To see how caching influence the impact of PIT size on the performance of our proposed congestion control mechanism, we present in Fig. 5 the distributions of the content download time (see Fig. 5a) and number of interest retransmission every 100s (see Fig. 5b). As caching always strives to improve performance especially in terms of content download time and being one of the key features of CCN/ICN, we do not include results for the scenario without caching due to page limit.

Given a PIT size that is larger than the bottleneck buffer size and using Proposed, Fig. 5a shows that about 60% of the contents have finished their downloads in about 300s. Only about 45% of the contents have completed their downloads within the same period of time if the PIT size is reduced from 20K to 7K entries (less than the buffer size). Similar performance behaviour is observed with TCP. Only about 45% of the contents have completed their downloads for P=20K while only about 20% of the contents have finished their download for P=7K in the same amount of time with TCP. However, thanks to our proposed PIT congestion control mechanism with about 50% of the contents achieving at most 20 retransmissions in every 100s unlike TCP with the same percentage of contents achieving 110 and 130 interest retransmissions every 100s for P=20K and P=7K, respectively. Fig. 5d shows that our mechanism is capable of avoiding the PIT becomes full for a significant fraction of time.

About 10% of the contents experience more than 100 interest retransmissions every 100s with Proposed, see Fig. 5b. Again this is due to increase rate of increasing the congestion window caused by caching. No significant performance difference is observed in the SCV. Due to delay variability caused by caching, about less than 5% of the contents experience SCV more than 1, making the distribution of the delay experienced by the requested chunks to be of high-variance. See Fig. 5c.



(a) CDF over all contents **(b)** CDF over all contents **(c)** CDF over all contents **(d)** CDF over time

Fig. 5: Impact of PIT size

F. *Impact of Zipf's skewness parameter*

In this section we consider two different values of α, 0.65 (many popular contents) and 0.95 (few number of very popular contents) and results from simulations are presented in Fig. 6.

Fig. 6a shows the improved content download time achieved with different values of α, with or without caching using Proposed or TCP. For Proposed, we observe about 90% and 70% of the contents complete their downloads in 400s, if $\alpha = 0.95$ while it is about 75% and 50% that complete their download in the same amount of time if $\alpha = 0.65$, in the presence (see Fig. 6b) and absence (see Fig. 6a) of caching, respectively. TCP achieves smaller percentage of contents that finish their download in the same amount of time (70% and 35% if $\alpha = 0.95$ and 45% and 30% if $\alpha = 0.65$ in the presence and absence of caching, respectively).

Interest aggregation can improve performance w.r.t. of the number of interest retransmissions every 100s. With Proposed in Fig. 6c about 90% of the contents experience a single interest retransmission for $\alpha = 0.95$ compared to about 10% that experience the same number of retransmission for $\alpha = 0.65$. Caching increases the rate at which consumer's congestion window increased as some of the chunk requests do not traverse the full perimeter of the network. This has the potential of increasing the number of retransmission leading to about 15% of the contents experiencing 1 interest retransmission and negligible for $\alpha = 0.65$, see Fig. 6d.

Larger values of α tend to benefit more from caching than lower values. This is evident as the SCV of 0.01 experienced by 75% of the contents in the absence of caching, Fig. 6f, is observed by about 60% of the contents in the presence of caching when $\alpha = 0.95$, see Fig. 6e using Proposed. The inverse is the case when $\alpha = 0.65$. No significant performance difference is observed in the two values of α considered.

VI. Related works

Key features of CCN/NDN such as interest aggregation, and universal in-network caching and dynamic eviction of contents pose new challenges to existing techniques for controlling and managing network congestion in today's IP networks. These have motivated several works on congestion/flow control in CCN/ICN at receiver nodes [3], [4], [6] and at intermediate nodes [9], [13], [11], [10], [5].

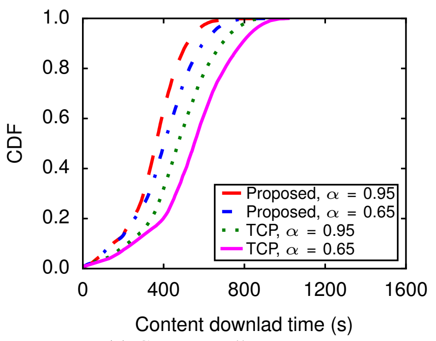
Proposals at intermediate (cum receiver) nodes bear a resemblance to our work as these aim to regulate (directly or indirectly) the rate of forwarding interests upstream. For instance, Rozhnova et. al. propose an interest rate shaper at an intermediate node based on the current queue occupancy and available bandwidth[11]. The performance of the proposed mechanism may degrade under busty traffic resulting in the queue exceeding the target. Wang et. al. consider a bi-directional approach for controlling congestion in CCN/NDN [10]. The authors model the congestion control problem as an optimization problem and propose an algorithm for shaping the interest rate using the network load (interests) and available bandwidth. Although the proposed mechanism effectively controls congestion with zero packet loss but requires parameter tuning to stabilize the queue.

In [13], Oueslati *et. al.* propose a flow aware traffic control at intermediate routers based on per-flow fair share of bandwidth using deficit round robin scheduling algorithm. The technique discards Interest(s) that belongs to a flow that has exceeded its share. To inform a CCN receiver of a congestion in the network, a DDR scheduler only rejects the payload and the header packet is sent downstream instead of discarding the entire packet. Downstream routers treat the truncated packet as a normal data packet. When it arrives at the receiver, packet loss is implied and the receiver adjusts its Interest rate accordingly (AIMD) and re-expresses the interest packet(s). Dropping interests and cutting data packet payload to signal congestion at intermediate nodes may not be the best solution as network resources might have been wasted.

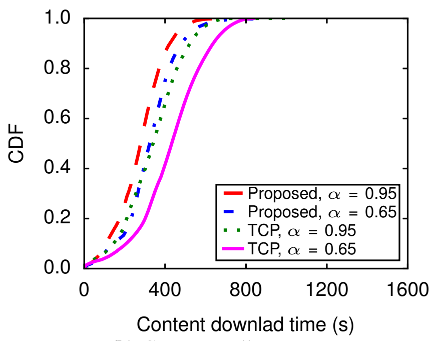
In addition, other proposals make use of NACK packets to signal congestion [5]. In [9], a credit counter is used to keep track of the number of data packet that a flow is allowed to transmit. All the existing works do not consider using the PIT occupancy to anticipate the data packet transmission queue occupancy. They always resort to dropping interest/data packets. A more desirable solution is to notify CCN receivers before congestion takes place. Our work has filled this gap with results demonstrating its effectiveness while taking into account the characteristics of CCN/ICN.

VII. Conclusion

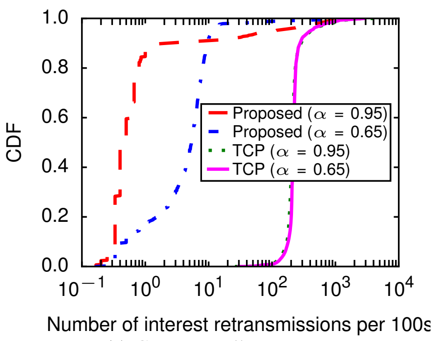
This paper has presented a novel congestion control mechanism for CCN and other ICN architectures. Our proposed method leverage the occupancy of the PIT to predict the future queue length of the data packet transmission buffer. In this technique, receivers are notified of imminent congestion before the queue reaches its target or before the buffer overflows. CCN consumers that receive congestion notifications slow



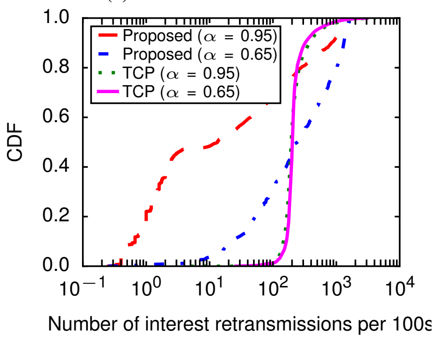
(a) CDF over all contents



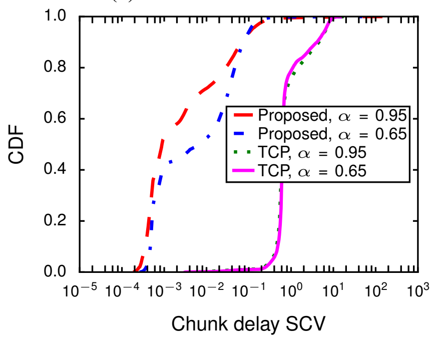
(b) CDF over all content



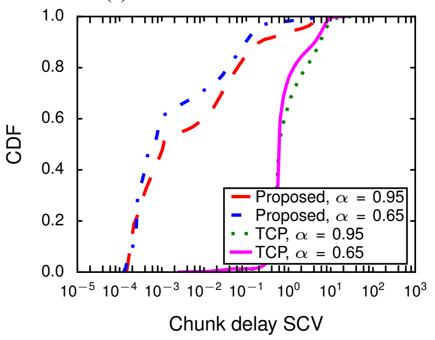
(c) CDF over all contents



(d) CDF over all contents



(e) CDF over all contents



(f) CDF over all contents

Fig. 6: Impact of Zipf's skewness parameter α. a, c and e represent the absence of caching while b, d, and f represent the presence of caching

down their sending rates. Simulation results show that our proposed method is effective in controlling congestion in the network. Comparative analysis results show the benefits of explicit congestion notification. None of the recently proposed congestion control mechanisms for CCN is available in ns-3 making our comparative analysis not to include any of these existing works. As a future work, we plan to implement one or two of the existing congestion control algorithms for CCN in ns-3 and compare with our proposed algorithm. In addition, we plan to carry out further analysis and more exposition of our proposed method in the presence of burst traffic.

ACKNOWLEDGMENT

This work was in part supported by the Hong Kong PHD Fellowship Scheme (PF11-02174).

REFERENCES

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," *Communication of the ACM*, vol. 55, pp. 117–124, Jan. 2012.

[2] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, , and B. Zhang, "Named data networking," Tech. Rep. NDN-0019, Revision 1, NDN, Apr. 2014.

[3] G. Carofiglio, M. Gallo, and L. Muscariello, "Icp: Design and evaluation of an interest control protocol for content-centric networking," in *Proceedings of IEEE INFOCOM Workshop, NOMEN*, Mar. 2012.

[4] L. Saino, C. Cocora, and G. Pavlou, "Cctcp: A scalable receiver-driven congestion control protocol for content centric networking," in *Proceedings of the IEEE ICC*, (Budapest, Hungary), June 2013.

[5] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications*, vol. 36, no. 7, pp. 779–791, 2013.

[6] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, and S. Wang, "Optimal multipath congestion control and request forwarding in information-centric networks," in *Proceedings of the 21st IEEE International Conference on Network Protocols*, Oct. 2013.

[7] A. J. Abu, B. Bensaou, and J. M. Wang, "Interest packets retransmission in lossy ccn networks and its impact on network performance," in *Proceedings of the 1st ACM ICN*, pp. 167–176, 2014.

[8] A. J. Abu and B. Bensaou, "Modelling the pending interest table occupancy in ccn with interest timeout and retransmission," in *Proceedings of the 40th IEEE LCN*, pp. 246–254, 2015.

[9] G. Carofiglio, M. Gallo, and L. Muscariello, "Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks," in *Proceedings of the 2nd ICN ACM SIGCOMM workshop*, 2012.

[10] 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," in *Proceedings of the 3rd ICN ACM SIGCOMM workshop*, pp. 55–60, 2013.

[11] N. Rozhnova and S. Fdida, "An effective hop-by-hop interest shaping mechanism for ccn communications," in *Proceedings of IEEE INFOCOM Workshop , NOMEN*, pp. 322–327, 2012.

[12] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[13] S. Oueslati, J. Roberts, and N. Sbihi, "Flow-aware traffic control for a content-centric network," in *Proceedings IEEE INFOCOM*, pp. 2417–2425, Mar. 2012.

[14] D. Saucez, L. A. Grieco, and C. Barakat, "Aimd and ccn: Past and novel acronyms working together in the future internet," in *Proceedings of the 2012 ACM Workshop on Capacity Sharing*, (New York, NY, USA), pp. 21–26, ACM, 2012.

[15] M. Busari and C. Williamson, "Prowgen: a synthetic workload generation tool for simulation evaluation of web proxy caches," *Computer Networks*, vol. 38, no. 6, pp. 779 – 794, 2002.

[16] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable icn," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pp. 147–158, 2013.