Leveraging the Pending Interest Table Occupancy for Congestion Control in CCN

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—**Recent studies have shown that the occupancy of the pending interest table (PIT) in content-centric networks (CCN), named-data networks (NDN) and other information-centric networking (ICN) architectures has an undeniable effect not only on the network performance but also on the congestion level. Despite CCN's in-network caching and interest aggregation for improving network performance, congestion can still take place in such networks. Existing mechanisms for controlling traffic in CCN/ICN do not consider the occupancy of the PIT in managing network congestion in both interest and data paths. In this paper, we propose a novel and effective mechanism to control traffic in CCN/ICN. In particular, our mechanism uses the average occupancy of the PIT to estimate the anticipated data packet transmission queue length and sends explicit congestion notifications to the content requesters to slow down their interest sending rates when the anticipated queue size exceeds a threshold. We demonstrate the effectiveness of our proposed mechanism with little or no overhead via simulation experiments in ns-3 network simulator.**

I. Introduction

Content-Centric Networking (CCN) [1] and Named-Data Networking (NDN) [2] are among the most promising Information-Centric Networking proposals that have brought a change in the future Internet communication model from a host-centric to a content-centric approach. Two key features of CCN and NDN are: 1) ubiquitous in-network caching, where each data chunk that travels through a node in the network is cached in the node perhaps for future reuse, and 2) interest aggregation that avoids repeated forwarding upstream of content requests pending in the PIT. With these techniques, a great deal of redundant traffic is filtered out from the network. 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, which motivates the design of a new clean-slate congestion control mechanism that takes into account the characteristics of CCN/ICN.

A CCN node receives at most one data packet for every interest packet forwarded upstream. The data packet is then forwarded downstream at most one per interface. If the arrival rate of data packet exceeds the transmission capacity of the downlink, the queue begins to build up and eventually packets are dropped leading to retransmission of interest packets. To regulate the rate of arrival of new data chunk one can regulate the rate of arrival of interests. Interest rates can be regulated at the requesters [3], [4], [5] and/or at the intermediate nodes [6], [7], [8]. A requester relies on timer expiration to infer congestion in the network but this approach can be slow in reacting to congestion. To solve this problem, intermediate nodes can notify requesters of congestion by sending a congestion control packet (aka NACK). However, sending NACKs downstream is not without drawbacks as it may requires message prioritization and incur additional overhead. Approaches that do not require message prioritization and incur no overhead are desirable.

The Pending Interest Table (PIT) in CCN is a data structure that keeps track of all interests forwarded upstream and 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. 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. However, existing mechanisms [3], [4], [5], [6], [7], [8] 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 leverage the occupancy of the PIT to predict the data packet queue size.

In this paper, we propose an effective mechanism to manage congestion in CCN/ICN. The following are the contributions of this paper: First, we propose a novel PIT-based congestion control mechanism for CCN taking into account the occupancy of the PIT towards predicting the data buffer occupancy in the next RTT. Our proposal sends explicit congestion notifications to the content requesters to slow down their interest sending rates if the anticipated queue size exceeds a threshold. Second, we propose an interest rate controller for requesters that take advantage of the congestion notifications received from the network. We demonstrate the effectiveness of our proposed mechanism via simulation experiments in n-3-ndnSIM.

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 congestion control mechanism. The performance evaluation of our proposed mechanisms is given in Section V. We 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.

A. Content-centric networking

Contents in CCN are divided into data chunks and each chunk is singly identified using a hierarchical naming convention. 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 receiver. To improve the network performance, every CCN node caches the data chunk to serve future request for the same chunk if possible.

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. An arrival of the requested data chunk consumes the pending interest in the PIT before forwarding the data chunk downstream. We denote the average rate of creating PIT entries as λ and the average rate of consuming PIT entries as μ. Each interest pending in the PIT waits for an average time τ for data to return. We assume data always return and τ is less than the PIT entry lifetime.

In the CCN and NDN progressive deployment plans, it is suggested that the PIT be elevated from a router's buffer, as a result the PIT can become a bottleneck that affects the performance of the system. As time t increases and $\lambda > \mu$, the occupancy of the PIT grows until it reaches its maximum size or a target PIT occupancy after which newly arrived interests are blocked with a given probability p_b leading to their retransmission and further congestion of the PIT.

In the case of interest blocking at the PIT, the average rate at which PIT entries are created is therefore $(1 - p_b)\lambda$. If μ data packets arrives at the data buffer per unit time and data chunks worth B packets are transmitted per unit time where B is the data packet transmission link capacity. Congestion can happen in the data packet transmission buffer if $B < \mu$. The data packet queueing delay can be observed to increase and the queue may become full eventually leading to relatively long content download time, high packet drop rates and their retransmissions causing further congestion of both the PIT (downstream nodes' PIT) and the router's buffer.

As time t increases the average number of PIT entries created per unit time and the average number of data packets waiting in the queue are $(1 - p_b)\lambda$ and $\max(0,\ \mu - B)$, respectively. Intuitively, we can control congestion at the PIT as well as at the router's buffer by increasing p_b at intermediate nodes or decreasing λ at receiver nodes. In this paper, we follow the latter approach.

III. Protocol Design Rationale

In Section II we show that there are two levels of congestion in CCN (PIT and router's buffer). This section presents the rationale behind the design of our congestion control protocol.

Due to the peculiarity of CCN and other ICN proposals, it is not sufficient to control congestion only at the router's packet transmission buffer. 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 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 [9], [7]. Another proposal uses a data packet, excluding its payload, to signal congestion [10]. 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 [5]. Similarly in our proposed congestion control protocol, routers signal to traffic sources of imminent congestion, 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 and other ICN proposals 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 need to erase an entry in order to accommodate a new request when the PIT is full or a target PIT occupancy is reached, an entry for content that occupies the largest number of PIT entries will be the best candidate. To this end, we propose to use a mechanism that determines such elephant content.

In summary, Our protocol design is informed 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 most of the entries in the PIT

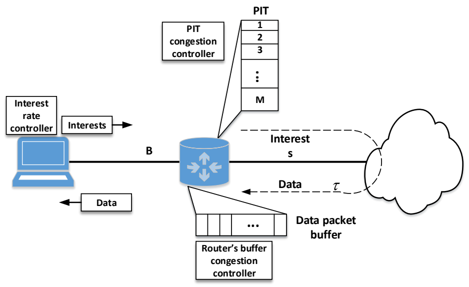

Fig. 1: A router's PIT and transmission buffer for data in a simple CCN network

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.

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 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 Option 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 [9] and [10], 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).

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 f_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 data packet and if $\tilde{M} \geq \rho_{max}$ 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 occupancy of the buffer 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 for each node we have κ defined as

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
```

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
```

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 $\tilde{Q}_j(t + \tau)$ 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 $flag_j$ associated with interface j is set, otherwise $flag_j$ is unset. Every data packet that is transmitted via interface j is marked if $flag_j$ is set. Conversely, no data packet is marked if $flag_j$ is unset. See Algorithm 2 for details.

F. Receiver's congestion controller

We present in this section the mechanism of our CCN receiver controller. When a CCN consumer receives a data packet it 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 forever for data to arrive, the controller maintains a timer for every interest sent. The controller can infer congestion upon the expiration of the timer. Upon interest timeout, the interest window is cut by half. Note that congestion window never goes below 1 packet.

Due to high delay variability caused by in-network caching in CCN, TCP-like RTO estimation has been shown to cause unnecessary retransmission [3], [4], [11] 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 [12] that records the maximum RTT observed over a given period of time. See [12] for the detailed algorithm. In addition, We set the RTO to a bit above the observed maximum value to avoid unnecessary retransmission.

Upon receiving a data packet and the OF is marked as "1", the controller infers congestion in the network and decreases its sending rate accordingly not more than once in 1 RTT. We borrow the idea from 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 resulted 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.

Algorithm 3: CCN consumer interest controller

```
1 Initialization: $W = W_{init} = 1$
  Input: $\beta$
  /* triggered by the consumer application or other
     events such as interest timeout                    */
  /* 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
```

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 network assisted congestion control 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

We consider a dumbbell topology consisting of 6 nodes. Two access routers $ar1$ and $ar2$ receive requests for contents from many users. Users' content requests follow a Zipf probability distribution with the skewness parameter α. In the case cache misses, all chunk requests from $ar1$ and $ar2$ can only be satisfied by only 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 routers follow a Poisson process with mean arrival rate λ. Requests for the same content have the same interest lifetime. We use proWGen [13] to generate our workloads.

In this performance evaluation, we consider the following metrics: *content download time* (i.e. the time from when the first request is sent to the time the last chunk for the content is received); average chunk delay per content; squared coefficient of variation (SCV) of the chunk delay per content; number of interest retransmission (at different intervals); data packet loss rate; link utilization; number of entries in the PIT (i.e., the number of pending interest entries in the PIT); queue length at the bottleneck link.

[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. The link connecting $cr1$ to $cr2$ is the bottleneck with 0.5Gbps and 70ms delay. The size of each returned data chunk is 1500 bytes. Buffer sizes in each node are set 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 [14]. 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 to the estimated RTO.

Simulations are run several times and the average values are presented for a lightly loaded network ($\lambda = 5$ users per second) and a heavily loaded network ($\lambda = 10$ users per second).

C. Impact of traffic load

1) Without in-network caching:

2) With in-network caching:

D. Impact of caching

E. Impact of PIT size

F. Impact of Zipf's skewness parameter

G. Simulation results

In this section we discuss the performance of PCCC in terms of the data packet transmission queue of the bottleneck link.

1) Queue size: Fig. 2 shows the data packet transmission buffer occupancy for both PCCC and interest packet rate shaper plus NACK (simply NACK) over time. We observe that our proposed mechanism do not cause spikes in the queue unlike NACK. With PCCC, the instantaneous queue length does not go beyond the set threshold of about 50 packets. Normally, PCCC should not reach the queue target given that we use the anticipated queue length of the buffer. However this depends on the propagation delay including other delays experienced by the marked data packets forwarded to

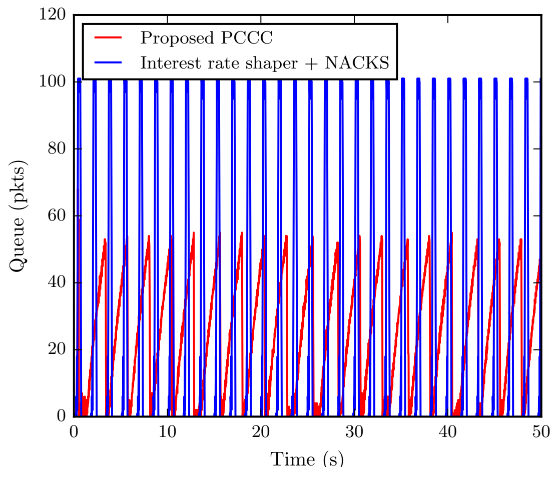

Fig. 2: Queue size at the bottleneck link

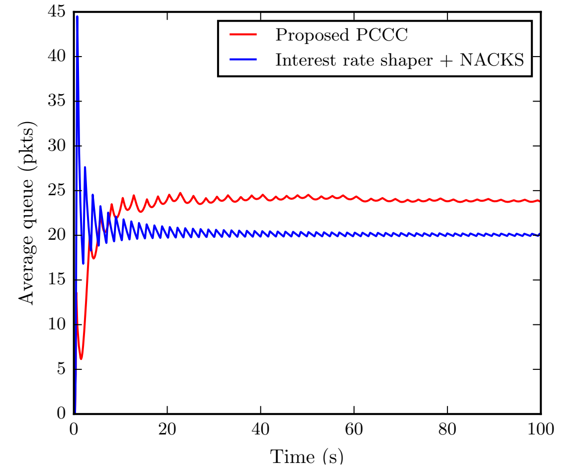

Fig. 3: Moving average queue size at the bottleneck link

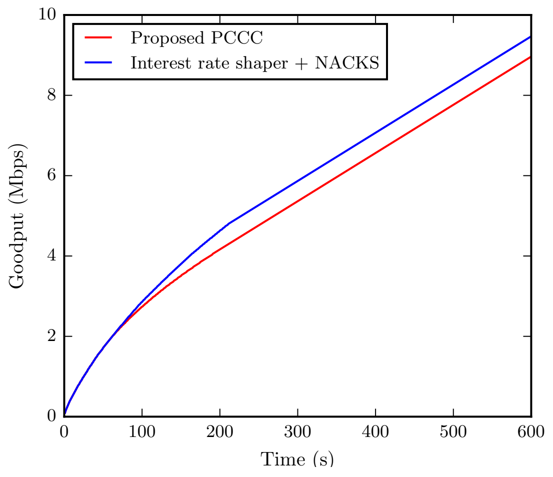

Fig. 4: Data packet goodput at one of the receiver node

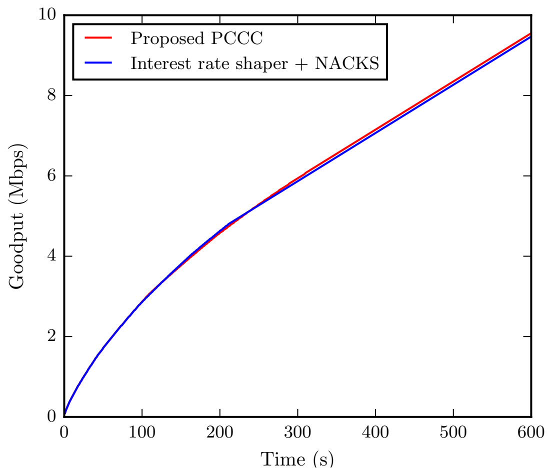

Fig. 5: Data packet goodput at one of the receiver node

downstream nodes. Note that we present results for the first 50 seconds for visibility.

We also present in Fig. 3 the moving average of the queue length as simulation time advances for both PCCC and NACK. We observe that both mechanisms converge over time.

2) Data packet goodput: Data packet goodput defines the number of data chunks that a receiver receives successfully per unit time. Fig. 4 shows the instantaneous goodput as time advances. Both PCCC and NACK have good performance but PCCC experiences a slightly lower goodput after 100 seconds. This is due to the differences in the interest rate controller algorithm at the receiver as PCCC stays in slowstart phase once and the rest in congestion avoidance phase while NACK remains in slowstart throughout except when the congestion window is set to the initial window size of 1 packet.

Fig. 4 shows the goodput when the target queue length is 50 packets. We change the target queue length to 150 packets and observe that PCCC has the similar performance to NACK.

VI. Related works

Congestion control problem in CCN and other ICN proposals has received a great deal of effort in recent years. New features of CCN/NDN such as multi-source content distribution, stateless CCN senders (no-acknowledgement of successfully delivered data), hop-by-hop communication (intermediate nodes processing of CCN packets beyond layer 3), interest aggregation, and universal in-network caching and dynamic eviction of contents (out-of-order arrival of data chunk), 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 at receiver nodes [4], [3], [15], [5], [16], [17], [18], [19] and at intermediate nodes [6], [20], [10], [8], [7], [9].

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. 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. 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 [10], 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. The proposed technique maintains information about the active number of interests. In fact, the authors do not consider anticipating the queue occupancy after a time period of an average data response time. Interests that will be forwarded at a rate exceeding the current fair rate are discarded. Each flow currently in the DRR *ActiveList* is associated with a counter and all counters are incremented by one quantum each time the DRR completes a cycle. The counter is decreased by the size of the requested data packet each time an Interest packet is sent to the CCN FIB. Upon receiving an Interest and the counter is less than the decrease factor the Interest is dropped. To inform a CCN receiver of a congestion in the network, a DDR scheduler only rejects the payload and send the header packet 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 resources of the network downstream upstream, respectively, might have been wasted.

In addition, other proposals make use of NACK packets to signal congestion [9]. In [6], a credit counter is used to keep track of the number of data packet that a flow is allowed to transmit. The proposed work uses information from the PIT to know the number of outgoing flows. All the existing works do not consider using the PIT occupancy to anticipate the data packet transmission queue occupancy and always resort to dropping interest/data packets. A more desirable solution is to notify CCN receivers before congestion takes place. Our work is aimed to propose effective congestion control mechanism that take into account the characteristics of CCN.

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 down their sending rates. Simulation results show that our proposed method is effective in controlling congestion in the network. Comparative analysis results show the benefit of explicit congestion notification. Additional analysis of our proposed PIT-based congestion control mechanism is still needed. As a future work, we plan to carry out further analysis and more exposition of our proposed method in the presence of burst traffic.

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. http://named-data.net/wp-content/uploads/2014/04/tr-ndn-0019-ndn.pdf.

[3] 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.

[4] 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 on Emerging Design Choices in Name-Oriented Networking*, (Orlando, FL), pp. 304–309, Mar. 2012.

[5] 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*, (Gottingen, Germany), Oct. 2013.

[6] 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 ACM SIGCOMM workshop on Information-centric networking*, (New York, NY, USA), pp. 37–42, ACM, 2012.

[7] 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 ACM SIGCOMM workshop on Information-centric networking*, pp. 55–60, 2013.

[8] N. Rozhnova and S. Fdida, “An effective hop-by-hop interest shaping mechanism for ccn communications,” in *Proceedings of IEEE INFOCOM Workshop on Emerging Design Choices in Name-Oriented Networking*, pp. 322–327, 2012.

[9] 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.

[10] 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.

[11] 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.

[12] 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 International Conference on Information-centric Networking*, pp. 167–176, 2014.

[13] 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.

[14] 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.

[15] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papalini, “Multipath congestion control in content-centric networks,” in *Proceedings of IEEE INFOCOM NOMEN*, (Turin, Italy), Apr. 2013.

[16] D. Byun, B.-J. B. Lee, and M.-W. Jang, “Adaptive flow control via interest aggregation in ccn,” in *Proceedings of the IEEE ICC*, (Budapest, Hungary), pp. 2331–2335, June 2013.

[17] A. Somaya, N. Pekka, E. Lars, and O. Jorg, “Contug: A receiver-driven transport protocol for content-centric networks,” in *Proceedings of the IEEE International Conference on Network Protocols (Poster session)*, 2010.

[18] S. Braun, M. Monti, M. Sifalakis, and C. Tschudin, “An empirical study of receiver-based aimd flow-control strategies for ccn,” in *Proceedings of the International Conference on Computer Communication and Networks*, 2013.

[19] F. Bjurefors, P. Gunningberg, C. Rohner, and S. Tavakoli, “Congestion avoidance in a data-centric opportunistic network,” in *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, pp. 32–37, Aug. 2011.

[20] T. Janaszka, D. Bursztynowski, and M. Dzida, “On popularity-based load balancing in content networks,” in *2012 24th International Teletraffic Congress (ITC 24)*, (Krakow), pp. 1–8, Sept. 2012.