Computer Communications 000 (2015) 1–10

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Providing explicit congestion control and multi-homing support for content-centric networking transport

Feixiong Zhang [a,*], Yanyong Zhang [a], Alex Reznik [b], Hang Liu [c], Chen Qian [d], Chenren Xu [e]

[a] WINLAB, Rutgers University, 671 Route 1 South, North Brunswick, NJ 08902-3390, USA
[b] InterDigital Communication LLC, 781 3rd Ave, King of Prussia, PA 19406, USA
[c] The Catholic University of America, 620 Michigan Ave. N.E., Washington, DC 20064, USA
[d] University of Kentucky, Lexington,Kentucky 40506, USA
[e] Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

ARTICLE INFO

Article history:
Available online xxx

Keywords:
Transport protocol
Explicit congestion control
Multi-homing
Content-centric network

ABSTRACT

Content-centric networking (CCN) adopts a receiver-driven, hop-by-hop transport approach that facilitates in-network caching, which in turn leads to multiple sources and multiple paths for transferring content. In such a case, keeping a single round trip time (RTT) estimator for a multi-path flow is insufficient as each path may experience different round trip times. To solve this problem, it has been proposed to use multiple RTT estimators to predict network condition. In this paper, we examine an alternative approach to this problem, CHoPCoP, which utilizes explicit congestion control to cope with the multiple-source, multiple-path situation. In addition, a parallelized version of CHoPCoP, pCHoPCoP, is designed to support bandwidth aggregation on multi-homed hosts. We have implemented CHoPCoP/pCHoPCoP on the ORBIT testbed and conducted experiments under various network and traffic settings. The evaluation shows that CHoPCoP can effectively deal with congestion in the multipath environment, and pCHoPCoP can maximize the network utilization for a multi-homed host.

1. Introduction

During the last decade, content retrieval has dominated the Internet usage. To address the challenges posed for content retrieval, content-centric networking (CCN) [13,17] has been proposed. Being a significant shift in the network design philosophy, CCN is centered on named content instead of host addresses. Routing towards a content is based on the content name instead of the host address, and data retrieval is initiated by issuing *Interest* at the content receiver. Compared to application-layer overlay solutions such as Content Distribution Networks (CDN) and Peer-to-peer systems (P2P), CCN holds the promise of providing a more efficient and cost-effective solution to content dissemination.

CCN's unique characteristics introduce new design challenges for the underlying transport protocol. First, CCN is naturally receiver-driven, since the content receiver needs to issue an Interest first in order to request a Data chunk. Second, hop-by-hop transfer is desired for CCN transport, because content files can be cached along the route to improve throughput. Moreover, since a specific content is often widely disseminated and cached in the network, a CCN flow may have multiple sources – i.e., one content chunk originates from source A, while the next chunk might originate from source B. Such multi-source/multi-path transfer in CCN makes congestion estimation based on a single RTT value fall short. These features of CCN are sufficiently distinct from a traditional end-to-end host-based model that a new transport approach is called for.

Recently transport protocol design for CCN has received attention in the research community and several proposals have been published [2–5,19,21,23]. In order to deal with the challenge caused by the multiple-source, multiple-path transfer, a recent study proposes the use of multiple RTT estimators at the receivers to gauge network congestion of each path. Additionally, recent studies also suggest the routers adopt a hop-by-hop Interest shaping scheme to actively prevent network congestion.

In this paper, we propose a new transport scheme, called the Chunk-switched Hop Pull Control Protocol (CHoPCoP), which has the following design elements:

- *Random early marking (REM).* REM uses explicit congestion signaling instead of RTT-based congestion prediction. Router detects congestion by monitoring the size of outgoing data queue. It then explicitly marks data packets to notify receivers when the

* Corresponding author. Tel.: +1 7327031188.
E-mail addresses: feixiong@winlab.rutgers.edu (F. Zhang), yyzhang@winlab.rutgers.edu (Y. Zhang), alex.reznik@interdigital.com (A. Reznik), liuh@cua.edu (H. Liu), qian@cs.uky.edu (C. Qian), chenren@cs.cmu.edu (C. Xu).

http://dx.doi.org/10.1016/j.comcom.2015.06.019
0140-3664/

network is congested. To our knowledge, this is the first paper that describes, analyzes and evaluates explicit congestion control for CCN.

- *Fair share interest shaping (FISP).* CHoPCoP router decides whether to forward an Interest immediately or delay it temporarily based upon the available queue sizes and the flow demands. FISP is triggered to delay Interests when REM can't effectively prevent congestion, for instance in the absence of cooperation from receivers, thus actively protects the router from congestion. FISP also realizes fair bandwidth sharing among flows by fair scheduling of multiple Interest queues at an interface.
- *AIMD-based receiver interest control (RIC).* The receiver adjusts its Interest window based on the AIMD (Additive Increase Multiplicative Decrease) mechanism. Here, receiver detects congestion mostly by marked packets from upstream routers.

We further extend CHoPCoP to a parallelized version (pCHoPCoP) for multi-homing scenarios in which the receiver has multiple active network interfaces. Here, the pCHoPCoP receiver consists of a pCHoPCoP engine and multiple CHoPCoP controllers, one for each network interface; and transport control functions are distributed among these components.

We have implemented CHoPCoP/pCHoPCoP using the Click Modular Router [14] and evaluated its performance on the ORBIT testbed [18]. We conduct experiments over various network and traffic settings and compare our protocol with several existing ones. Our evaluation shows that (i) explicit congestion control provides congestion detection timely and correctly in a multi-source/multi-path setting, and significantly alleviates detection errors due to source/path change, thus improves network stability and efficiency; (ii) our FISP scheme ensures fair sharing of network resources among different flows, it also provides protection against misbehaving receivers while still makes the most of network resources; (iii) our receiver Interest control scheme guarantees full bandwidth utilization while reacts to REM signal to avoid saturating the network; (iv) the pCHoPCoP receiver can utilize all its interfaces and realize bandwidth aggregation over each of them, thus maximize the network utilization for the multi-homed host.

The rest of the paper is organized as follows. Section 2 describes the overall features of CCN and the related work on transport control in CCN. A detailed description of CHoPCoP and pCHoPCoP is given in Sections 3 and 4. Our implementation is presented in Section 5. Section 6 presents ORBIT-based evaluation results, and concluding remarks are given in Section 7.

2. CCN and transport control

In this section, we give an overview of CCN's basic features [13] and discuss the related work on transport control.

2.1. CCN background

There are two types of CCN packets: *Interest* and *Data*. A receiver asks for content by issuing an Interest packet and the corresponding Data chunk is returned in response to that Interest. The receiver can thus control the progression of content retrieval by adapting the Interest sending rate.

CCN packet forwarding is performed using three main data structures: forwarding information base (FIB), content store (CS) and pending Interest table (PIT)—FIB is used to forward Interest packets toward the data source, CS is the cache memory to store passing Data chunks, and PIT keeps track of forwarded Interest packets so that returned chunks can be sent to its receiver.

When an Interest packet is received from interface *f*, the router performs the following operations: (1) checks CS and returns a copy through *f* if cache hits; (2) otherwise, conducts a PIT lookup to verify if an entry for the same content name already exists. If so, appends *f* to the entry and discards the Interest; (3) if not, creates a new PIT entry and forwards the Interest through the interface indicated by FIB.

When a Data chunk is received, the router forwards the chunk to all interfaces specified by the corresponding PIT entry and then removes that entry. The router may also choose to cache the chunk in CS, if appropriate.

2.2. Related work on transport control

There has been some work on transport schemes proposed for CCN. Some propose a RTT-based congestion control approach, in which the receiver maintains a single round trip time (RTT) estimator (for instance ICP [2] and ICTP [21]) or multiple RTT estimations [3,4], one for each route, to predict network status. Multiple RTT based scheme works well for multipath transfer in CCN, however, it adds lots of complexity to the receiver and heavily relies on the accuracy of timing. Our scheme, on the other hand, utilizes explicit congestion signaling from network to effectively notify the receiver about network condition. Compared with multiple RTT based scheme, our approach leads to a much simpler receiver design and is not restricted to limitations of timers which have long been criticized [8,10,20,25].

In [5,19], authors propose quota-based hop-by-hop Interest shaping scheme to actively control traffic volume. They assign a quota (in terms of the number of pending Interests) to each flow, and if the number of pending Interests for a flow exceeds the quota, that flow's Interests will be delayed or dropped. The quota-based Interest shaping requires assigning an appropriate quota value for each flow, which is challenging in practice, if not impossible; and it can be rather inefficient under dynamic traffic setting. Our fair share Interest shaping scheme also considers about fairness, however, resources are shared by all flows and Interest from a flow is delayed temporarily only when shared resources become limited and the corresponding flow unfairly consumes resources.

In addition, several transport layer protocols [11,12,16] are proposed to realize bandwidth aggregation for multi-homed hosts in the Internet. They propose an end-to-end transport approach in which either the sender [12,16] or the receiver [11] maintains multiple transport pipes/subconnections, one for each interface. We extend CHoPCoP to a parallelized version by borrowing some of their mechanisms. To our best knowledge, pCHoPCoP is the first transport protocol that supports multi-homing in CCN.

3. CHoPCoP design

In this section, we describe the design of the CHoPCoP protocol in detail. CHoPCoP consists of the following three functional modules: (1) explicit congestion signaling by random early marking; (2) fair share Interest shaping; and (3) AIMD based receiver Interest control. Fig. 1 illustrates the functional modules of CHoPCoP content provider, router, and receiver. In our router model, the memory allocated for buffering packets at interfaces is separated from that used for caching (CS). Each interface has separate inbound buffers/queues and outbound buffers/queues for Interest and Data.

CCN data chunk is large in size [13] because of extra fields in the packet (e.g. signature). Large chunk can cause fragmentation, which may drastically harm throughput since the loss of a single fragment will cause retransmission of the whole chunk, like IPv4 fragmentation. Similar to two-level content segmentation in [21], we thus propose that a chunk is segmented to multiple small packets before the content provider sends it out and the receiver aggregates received packets into a complete chunk. If certain data packets in a chunk are lost, the receiver will issue a specific Interest packet containing the offsets of lost data packets within that data chunk, causing only those lost data packets to be retransmitted.

Please cite this article as: F. Zhang et al., Providing explicit congestion control and multi-homing support for content-centric networking transport, Computer Communications (2015), http://dx.doi.org/10.1016/j.comcom.2015.06.019

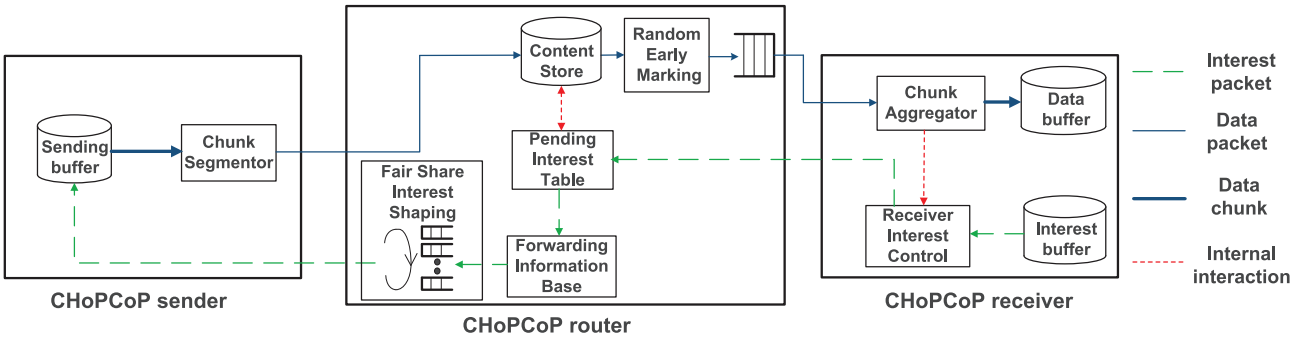


Fig. 1. Functional modules of CHoPCoP content provider, router and receiver (we omit inbound queues in router for simplicity).

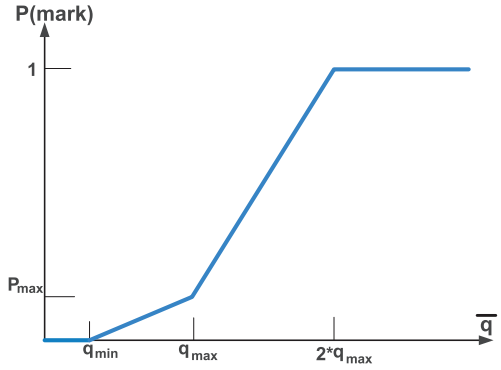


Fig. 2. Mark probability function for REM.

3.1. Explicit congestion signaling

Since content flow in CCN may have multiple sources and multiple paths, using a single RTT estimation can not work well. Although using multiple RTT estimations [3] is proposed to address the issue, we take a different approach that each intermediate router estimates its congestion level and then notifies the receiver of the congestion event. Upon reception of such a notification, a properly functioning CHoPCoP receiver slows down the Interest issuance rate using the method presented in Section 3.3.

The technique we propose to achieve explicit congestion signaling is referred to as *Random Early Marking* (REM), similar to mechanism in RED [10] and ECN [8]. In REM, before a router forwards a data packet through interface f, it samples the occupancy of the corresponding outbound data queue, denoted by q. The router then smooths the sampling by calculating the moving average of queue occupancy $\bar{q}$ as

$$\bar{q} = (1-\mu)\bar{q} + \mu q, \tag{1}$$

where $0 < \mu < 1$ is a design parameter which sets the weight of the current sampling. In this way, we can avoid the adverse impact caused by temporary increases in the data queue. The router marks the packet with a probability $P = \frac{\bar{q}-q_{min}}{q_{max}-q_{min}} P_{max}$ if the value of $\bar{q}$ is between q_{min} and q_{max}, and with a probability $P = P_{max} + \frac{\bar{q}-q_{max}}{q_{max}}(1 - P_{max})$ if $\bar{q}$ is between q_{max} and $2 * q_{max}$ (shown in Fig. 2). If the queue occupancy is above the threshold $2 * q_{max}$, the router always marks the packet. Such gentle variation of packet marking probability is proved to make explicit congestion control much more robust to the setting of parameters [9].

REM predicts network congestion much more accurately than single RTT estimation in a multi-source environment. It actively notifies the receiver prior to congestion taking place, thus keeps the network stable. REM enjoys other additional benefits same as RED/ECN, including the avoidance of global synchronization and bias against bursty traffic, and the guarantee of statistical fairness [10]. Compared to multiple RTT estimation proposal [3], REM vastly simplifies congestion detection at receiver and is free from limitations of timer [8,10,20,25]. Compared with RED/ECN, in REM the congestion signal is delivered to the receiver and directly used for controlling rate instead of being reflected back to the sender, thus causing less delay.

3.2. Fair share interest shaping (FISP)

A lightweight flow-aware network paradigm is well-adapted to CCN and can bring significant advantages [19]. This requires realizing per-flow fair bandwidth sharing at router and overload control even when the receiver is non-cooperative.

In CHoPCoP, we propose a fair share Interest shaping scheme, FISP in short, to address these issues. As seen in Fig. 1, multiple Interest queues are allocated in an interface, one for each active content flow, identified by the content name. An active content flow corresponds to a FIB entry that has at least one PIT entry in active, thus flow information can be easily extracted from FIB and PIT. A modified DRR [22] is further used for fair scheduling among different queues: the deficit counter of a queue here is decreased by the size of the corresponding Data after servicing an Interest. To deduct such size value, we can use the segmentation information from the Interest (e.g. in the CCNx prototype [6]), or define a standard field as suggested in [5]. The analysis in [15] shows that the number of active flows that need scheduling remains in hundreds even though there may be tens of thousands of flows in progress, thus demonstrates that fair sharing is scalable.

Moreover, FISP is triggered to delay certain Interests when a router's data queue continues to grow even though REM has marked outgoing data packets, which is particularly helpful when the receiver is non-cooperative. FISP realizes this using an algorithm that consists of the following four phases.

In the first phase, FISP checks whether delaying Interests should take place. To do so, FISP periodically counts the total queue requirement at an interface, Q, as

$$Q = q_d + \gamma q_i, \tag{2}$$

where q_d quantifies the occupancy of outgoing data queue which is directly extracted from REM, q_i quantifies buffer resources needed by data chunks that will arrive at a response to outstanding Interests, and γ is a weight parameter. q_i is implemented along with PIT. It is increased when an Interest corresponding to the interface is sent upstream and decreased when data comes back. The queue requirement Q is then smoothed using a moving average method. If the value of Q exceeds a preset threshold value, Q_{min}, the router starts Interest shaping for this interface.

In the second phase, FISP determines which flow's Interests should be delayed by looking at each flow's queue requirement. We use Q_j to denote flow j's queue requirement calculated similar to Eq. (2). If Q_j exceeds its *fair share*, i.e., $Q_j \geq \frac{Q_{min}}{n}$ (n is the number of flows the interface currently has), then flow j's packets are delayed

Please cite this article as: F. Zhang et al., Providing explicit congestion control and multi-homing support for content-centric networking transport, Computer Communications (2015), http://dx.doi.org/10.1016/j.comcom.2015.06.019

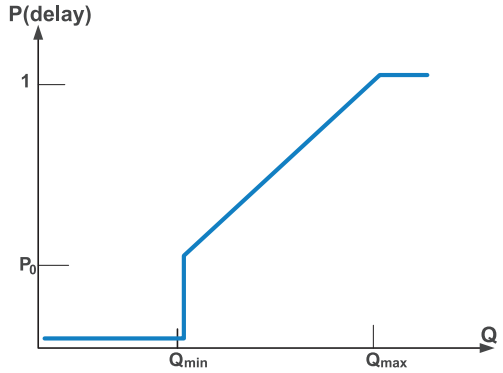


Fig. 3. Delay probability function for FISP.

with a probability $P = \frac{Q-Q_{min}}{Q_{max}-Q_{min}} + P_0$ as pictorially shown in Fig. 3. The Interests that are delayed are sent to a delay queue instead of being sent upstream and will re-enter outgoing Interest queues after a certain interval. The delayed Interests will not be counted towards the queue requirement for the interface.

In the third phase, we consider the overly-congested scenario. Once the queue requirement Q exceeds another threshold, Q_{max}, then any incoming Interest will be delayed.

In the fourth phase, if the router finds that the queue requirement, Q, falls below $Q_{release}$, then it will release all the delayed Interests to the outgoing Interest queues. We note that the relationship between the three threshold values is $Q_{release} < Q_{min} < Q_{max}$. Moreover, Q_{min} in FISP is larger than $2 * q_{max}$ in REM, and the delay probability starts from P_0 instead of 0, ensuring FISP is triggered to delay Interests after REM, when router queue continues to accumulate even after packet marking.

FISP actively protects the network from congestion even with non-cooperative receivers. Compared with quota-based Interest shaping [5,19], FISP is more efficient in resource utilization, while it also ensures fairness among different flows. Moreover, the whole protocol is kept simple since the delaying algorithm in FISP is not triggered in normal condition.

3.3. AIMD based receiver interest control (RIC)

In CHoPCoP, each receiver maintains an Interest window that keeps track of pending Interests. The size of this window determines the Interest rate from the receiver, which in turn impacts the traffic volume in the network. We need to keep the receiver Interest window large enough to enjoy the available bandwidth, while not saturating network capacity. Here, we use the AIMD (Additive Increase Multiplicative Decrease) mechanism to manage receiver Interest control (RIC). Specifically, the receiver adjusts its Interest window proportional to congestion level implied in explicit congestion signaling.

RIC consists of two phases, namely, the slow start phase and the congestion avoidance phase. The slow start phase begins when the receiver sends out the first Interest for a given content. In this phase, the Interest window rapidly increases to utilize the network capacity by incrementing the window size by one every time it receives a complete data chunk. After the window size, W, reaches a threshold, or when the network is congested, the receiver starts the congestion avoidance phase. Here, the receiver window is either increased at a much slower rate, or decreases, depending upon whether the congestion is detected. Before congestion is detected, every time after the receiver receives W data chunks, we increase the window size by α where $\alpha < W$. After congestion is detected, however, the window size decreases to βW where $\beta < 1$. The values of α and β determine how aggressive the user is in tracking available bandwidth, as has been well studied in TCP [7].

In CHoPCoP, the receiver detects congestion either when it has a timeout or receives marked packets. We use different β values in these situations. In the former case, we simply use a fixed value, β_0, and in the latter, we calculate β as

$$\beta = \beta_2 - \frac{(\beta_2 - \beta_1)N_{marked}}{N}, \tag{3}$$

where N_{marked} and N denote the number of marked packets and the total number of packets in a chunk respectively.

Eq. (3) shows that the receiver reacts to REM in proportion to the extent of congestion, not only its presence. Note that in normal condition the receiver detects congestion through receiving REM notification; timeout only takes place when REM fails.

Whenever a timeout occurs, the receiver needs to retransmit the Interest. RIC retrieves the offsets of lost data packets within the data chunk from the chunk aggregator and sends out the Interest with the offset information. The timeout value is calculated as:

$$R\bar{T}T = \sigma R\bar{T}T + (1-\sigma)RTT, \tag{4}$$

$$TimeOut = \delta R\bar{T}T, \tag{5}$$

Where RTT denotes the current RTT sample value, while $R\bar{T}T$ denotes the moving average of RTT value. Note that we would set the timeout parameter relatively large because REM is adopted and timer is not critical any more.

4. pCHoPCoP: parallel CHoPCoP

CHoPCoP is designed for the data transmission scenario in which the receiver has only one active interface for fetching content. For multi-homed hosts, the receiver could have multiple active interfaces fetching content from several content providers, thus an extension for CHoPCoP is needed which is called pCHoPCoP. A multipoint-to-point pCHoPCoP connection is regarded as multiple CHoPCoP running paralleled to download a content from several interfaces. We present its overall architecture and function elements here.

4.1. Design framework

Fig. 4 shows the overall architecture of pCHoPCoP. The sender and router work in the same way as CHoPCoP. However, for pCHoPCoP receiver, the transport layer consists of two functional units: the pCHoPCoP engine and the CHoPCoP controller. The pCHoPCoP engine controls the cooperation and progression of different CHoPCoP controllers. The application pushes Interest packets to the pCHoPCoP engine, and then pCHoPCoP engine distributes these Interest packets to different CHoPCoP controllers. A CHoPCoP controller receives data chunks from its corresponding network interface, and then sends them to the pCHoPCoP engine; finally the application pumps data from pCHoPCoP engine.

A pCHoPCoP connection consists of multiple CHoPCoP pipes.[1] Transport control functions related to per-pipe characteristics are put at each CHoPCoP controller while other functions pertaining to aggregate connection are managed at the pCHoPCoP engine. Here, reliability and buffer management pertain to the aggregate connection, and are handled by pCHoPCoP engine. In addition, flow control also lies at pCHoPCoP engine. On the other hand, congestion control, being per-pipe functionality, is still handled by individual CHoPCoP controllers. Therefore, while an individual CHoPCoP controller controls how much data it can request along its path, the pCHoPCoP engine controls what data to request through each CHoPCoP controller.

pCHoPCoP achieves bandwidth aggregation by effectively scheduling transmissions (Interests) to each CHoPCoP pipe. Briefly,

[1] The data transmission path formed by a CHoPCoP controller is denoted as CHoPCoP pipe.

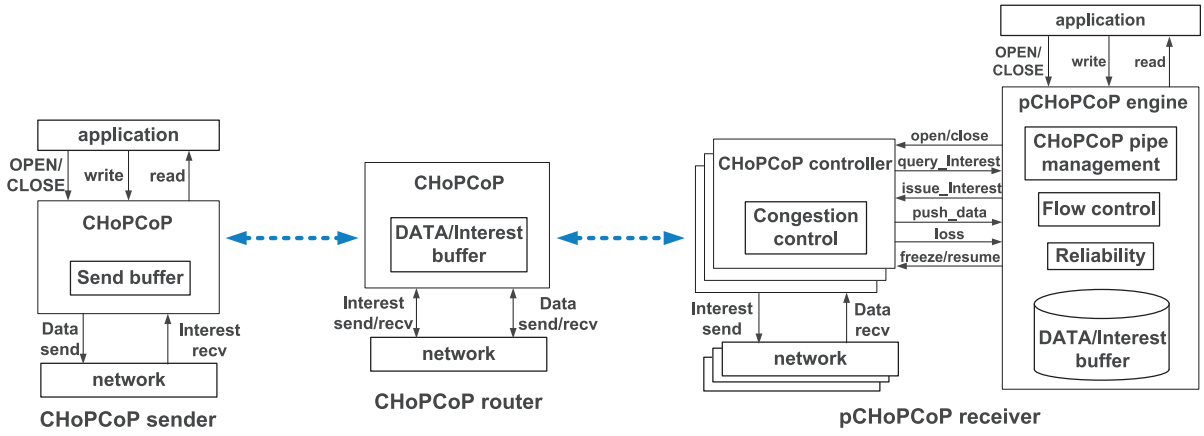


Fig. 4. pCHoPCoP protocol architecture. The pCHoPCoP receiver consists of a pCHoPCoP engine and multiple CHoPCoP controllers.

CHoPCoP controller calls for Interests from pCHoPCoP engine when there is space in its congestion window and pCHoPCoP engine assigns Interests to the CHoPCoP controller based on the pending Interest space. The individual CHoPCoP controller is responsible for loss detection and reports any loss detected to pCHoPCoP engine such that the corresponding Interest will be reassigned to another CHoPCoP controller that has space in its window.

4.2. Protocol details

The pCHoPCoP engine maintains two key data structures for performing effective Interest packet scheduling: the pending Interest space and the pipe state table.

Pending Interest space: The data structure keeps the ranges of chunk IDs for data yet to be requested, which includes the chunk IDs of data that need to be re-requested, and chunk IDs greater than the highest chunk ID requested so far.

Pipe state table: The data structure keeps track of which CHoPCoP pipes are active and which are not. As the available memory of the pCHoPCoP receiver is limited, the pCHoPCoP engine needs to consider whether it has enough space in the receiving buffer or not when a CHoPCoP controller calls for an Interest. The pCHoPCoP engine will thus freeze or reactivate a CHoPCoP pipe accordingly.

As seen in Fig. 4, four functions act as the interface between the application and the pCHoPCoP engine.

OPEN()/CLOSE(): The application opens a pCHoPCoP socket by using the OPEN() call and terminates the pCHoPCoP socket by the CLOSE() call.

Write()/read(): The application publishes its ranges of chunk IDs for data to be requested to the pCHoPCoP engine using the write() call and fetches data from the receiving buffer of the pCHoPCoP engine using the read() call.

In addition, eight functions act as the interface between the pCHoPCoP engine and CHoPCoP controllers.

Open()/close(): The pCHoPCoP engine creates a CHoPCoP controller by using the open() call and then the CHoPCoP controller starts the communication through its corresponding interface. The pCHoPCoP engine removes a CHoPCoP pipe by sending a close() call to the corresponding CHoPCoP controller. The pipe state table will be updated for these two operations.

Query_Interest()/issue_Interest()/push_data(): When a CHoPCoP controller has space in its congestion window, it queries the pCHoPCoP engine for Interest through the query_Interest() call. The pCHoPCoP engine then distributes a suitable Interest packet to the CHoPCoP controller using the issue_Interest() call. When a CHoPCoP controller gets a data chunk, it sends it to the pCHoPCoP engine by using the push_data() call.

Freeze()/resume(): When the pCHoPCoP engine does not have enough buffer, it uses freeze() call to freeze some CHoPCoP pipes and updates the pipe state table. When enough buffer becomes available again, the pCHoPCoP engine uses resume() call to reactivate some sleeping pipes according to the pipe state table.

Loss(): When a CHoPCoP controller detects a loss, it uses the loss() call to inform the pCHoPCoP engine. The pCHoPCoP engine then inserts the chunk ID in the pending Interest space, thus the Interest will be retransmitted later.

5. Implementation

We have implemented a complete network stack for our protocol CHoPCoP/pCHoPCoP as a user-level daemon, using the Click Modular Router. Some parameter settings of the protocol are listed below:

REM (random early marking): Parameters of the packet marking probability function is set as follows: $q_{min} = 0.1C$, $q_{max} = 0.3C$, $P_{max} = 0.002$, where C denotes the queue capacity. For REM smoothing, we set $\mu = 0.05$ in Eq. (1). The parameter setting is based on discussions in RED [10]. Note that the value of P_{max} is rather small because a data chunk is regarded as marked at the receiver as long as any of its segmented packets gets marked.

FISP (fair-share interest shaping): We set $\gamma = 0.6$ in Eq. (2). For the delay probability function, $Q_{release} = 0.2C$, $Q_{min} = 0.6C$, $Q_{max} = 0.9C$, $P_0 = 0.3$, where C denotes the buffer capacity for the interface.

Additional description of the implementation can be found in [24].

6. Evaluation

In this section, we describe our evaluation effort of CHoPCoP/pCHoPCoP on the ORBIT testbed [18]. Our evaluation focuses on CHoPCoP's capability in stabilizing network condition and adapting to CCN's multipath nature, and pCHoPCoP's capability in achieving aggregated bandwidth on multi-homed hosts. For such purpose, we conduct detailed experiments on several simplified but representative network topologies and compare CHoPCoP with existing CCN transport protocols that utilize a single RTT estimation (i.e. ICP [2], HR-ICP [5] and ICTP [21]) and protocols that use quota-based hop-by-hop Interest shaping (i.e. HbH in HR-ICP). We don't compare CHoPCoP with multiple RTT estimation proposals here since our scheme can be an alternative to those proposals in controlling multipath transfer. Due to lack of space, we omit some of CHoPCoP's evaluation which can be found in [24].

6.1. A single-source, single-destination benchmark scenario

We first conduct several benchmark experiments using a three node baseline topology, shown in Fig. 5. Here, the source node (A)

Please cite this article as: F. Zhang et al., Providing explicit congestion control and multi-homing support for content-centric networking transport, Computer Communications (2015), http://dx.doi.org/10.1016/j.comcom.2015.06.019

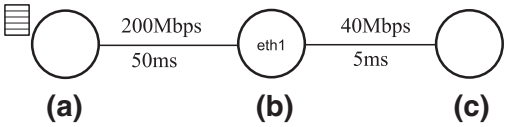


Fig. 5. A three node baseline topology, consisting of a source (A), a router (B) and a receiver (C).

is connected to the router (B) via a long Internet connection (with a 200 Mbps bandwidth and a 50 ms latency), and the router is connected to the receiver (C) via a local Internet access (with a 40 Mbps bandwidth and a 5 ms latency). We set the outbound data queue size of the router's eth1 interface to be 40 Mbps × 300 ms = 1500 KB according to the bandwidth-delay product rule [1]. We consider a 320 MB content file that consists of 10,000 chunks in total, each chunk 32 KB in size. There is only one flow in this setting.

6.1.1. The effectiveness of REM and RIC

First, we compare CHoPCoP with existing protocols: ICTP [21], ICP [2], and HR-ICP [5]. Here, the receiver is cooperative and slows down Interest issuing when marked packets are observed, so the main components that are effective in CHoPCoP are REM and RIC. We will show that CHoPCoP stabilizes the network condition and achieves higher throughput.

Fig. 6(a)–(d) summarizes the results, showing how the receiver window size, the transient receiving data rate, the number of timeout instances, and the router queue size, change over time. The results show that CHoPCoP significantly outperforms the others. The receiver side Interest window is much smoother, with an average size of 23.27 and a standard variance of 2.57 (Fig. 6(a)); the receiving data rate is much higher (also smoother), with 39.91 Mbps at the steady state (Fig. 6(b)); and no timeout is observed at the receiver (Fig. 6(c)). Specifically, the throughput[2] of CHoPCoP is 85.75 and 13.69% higher than that of ICP/HR-ICP and ICTP respectively. This is because it effectively smooths the outgoing data queue at the router (with an average of 235.8 KB and standard variance of 116.5), as shown in Fig. 6(d).

The inferior performance of ICTP/ICP/HR-ICP can be explained below. In ICTP, after the receiver window reaches a certain value (~22 in our experiment), the network capacity is fully utilized; thus the receiving Interest rate remains the same even if the window keeps increasing. However, the increasing window results in a large queue at the router, which may easily cause congestion. On the other hand, in ICP/HR-ICP, a relatively small timeout setting causes a large number of false timeouts and thus poor throughput.

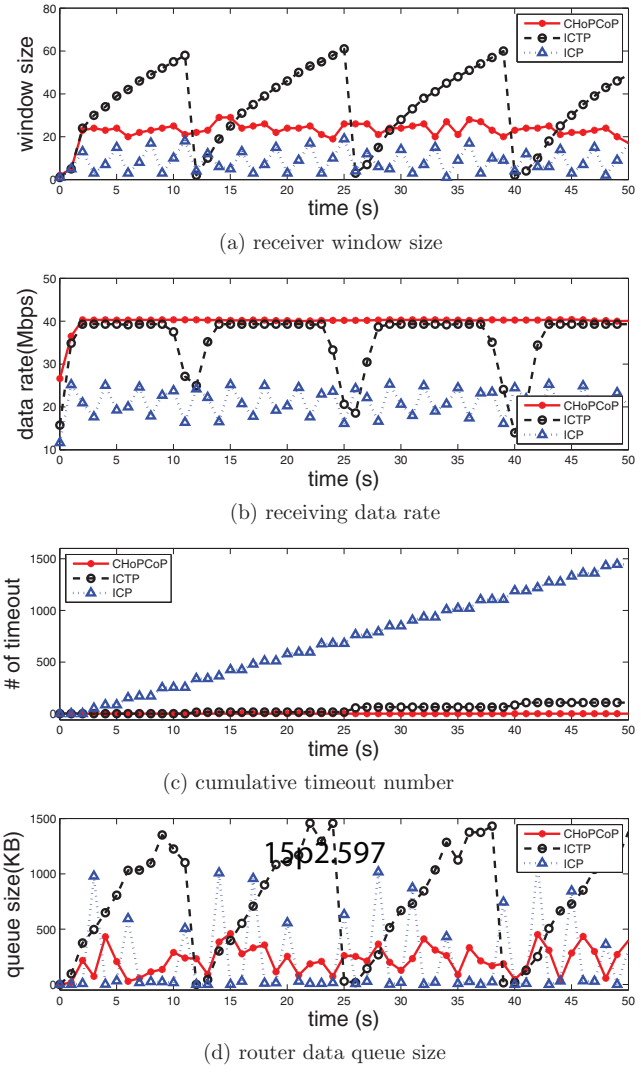
6.1.2. The effectiveness of fair share interest shaping (FISP)

Next, we consider a non-cooperative receiver who issues Interests at a constant Interest rate (CIR), even when the router has signaled congestion through packet marking. In this case, FISP is triggered to actively delay Interest in order to mitigate congestion since the receiver does not respond to REM signaling.

Here, the router's outgoing data queue is 1500 KB in size, and we have $Q_{max} = 0.9C = 1350$ KB. We consider CIR of 140, 160, and 200 Interests per second in each run, requesting a 320 MB file. We show the router queue size with and without FISP in Fig. 7, and the delivery ratio and throughput in Table 1.

The results show that with FISP, the router's outgoing data queue can be kept at ~1050 KB when CIR > $\frac{40\,\text{Mbps}}{32\,\text{KB}} \approx 150$ Interests per second. Without FISP, router queue overflows and the router keeps congested. On the other hand, FISP's Interest shaping incurs the accumulation of Interest packets at the router which might exceed the capacity of the internal delay queue and cause Interest dropping.

[2] We use the term throughput for the average data delivery rate, while the term receiving rate for instant data delivery rate.



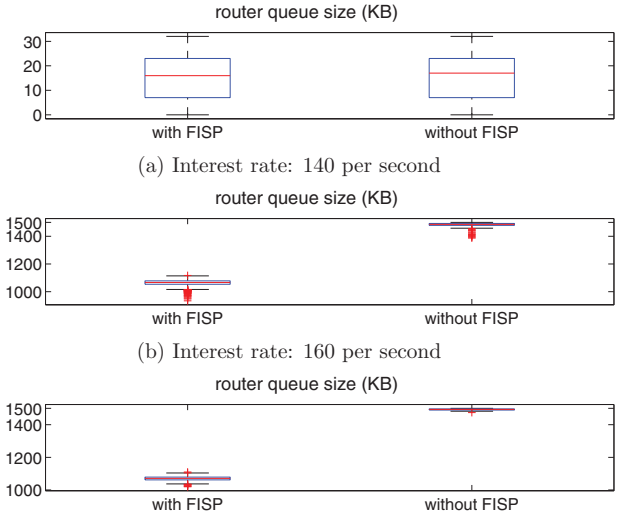
(a) receiver window size

(b) receiving data rate

(c) cumulative timeout number

(d) router data queue size

Fig. 6. The performance results for a 3-node baseline topology. ICP and HR-ICP behave very similarly, we thus use the ICP curve to represent both schemes.



(a) Interest rate: 140 per second

(b) Interest rate: 160 per second

(c) Interest rate: 200 per second

Fig. 7. Boxplot of router queue size with and without FISP under different CIR. At a low CIR, both schemes perform similarly. At a high CIR, FISP nicely controls the router queue size, and thus avoids queue overflow. Without FISP, the router queue overflows when the CIR is above 160 Interests per second.

Table 1
FISP leads to better data delivery ratio and throughput.

CIR	Data delivery ratio		Throughput	
	With FISP	Without FISP	With FISP	Without FISP
140	100%	100%	36.31 Mbps	36.36 Mbps
160	100%	5.86%	37.44 Mbps	2.4 Mbps
200	100%	1.91%	37.44 Mbps	977.9 Kbps

Table 2
Results under different chunk size. Each packet is 1 KB in size.

Chunk size (pkts per chunk)	Average receiving Rate (Mbps)	Average interest Window size	Average router Queue size (KB)
2	10.08	106.61	0.71
8	34.40	91.60	84.84
32	36.93	21.92	197.69
128	33.77	6.23	384.56
512	24.30	1.76	420.26

Table 3
Results under different REM smoothing parameter. Chunk size is 32 KB.

Smoothing Parameter μ	Average receiving Rate (Mbps)	Average interest Window size	Average router Queue size (KB)
0.01	37.02	23.04	228.81
0.05	36.93	21.92	197.69
0.25	36.94	21.87	193.59

Table 4
Results under different parameter (q_{min}, q_{max}) in the REM mark probability function. Chunk size is 32 KB, $P_{max} = 0.002$. Note that q_{max} should be smaller than router buffer capacity C.

(q_{min}, q_{max})	Average receiving Rate (Mbps)	Average interest Window size	Average router Queue size (KB)
$(0.01C, 0.05C)$	29.47	13.49	30.69
$(0.05C, 0.2C)$	36.43	19.51	122.79
$(0.1C, 0.3C)$	36.93	21.92	197.69
$(0.2C, 0.4C)$	37.09	25.12	296.24
$(0.3C, 0.5C)$	37.10	29.21	420.56
$(0.8C, 0.9C)$	37.10	76.57	959.93

We set the capacity of the internal delay queue to be 5000 packets, and additional experiments show that FISP can handle a maximum CIR of around 300 without Interest dropping.

6.1.3. The sensitivity of CHoPCoP parameters

Here, we evaluate how CHoPCoP's performance is impacted by different parameter settings. In particular, we investigate the sensitivity of CHoPCoP parameters including the chunk size, REM smoothing parameter μ in Eq. (1), REM's marking probability function in Fig. 2, FISP's weight parameter in Eq. (2), and FISP's delay probability function in Fig. 3.

Table 2 shows that having a chunk size of 32 packets gives the best performance. If the chunk size is too large, the control granularity is not fine enough to have timely response to network dynamics; If the chunk size is too small, the control overhead becomes more pronounced.

Table 3 shows that REM is insensitive to the value of μ, making the calculation algorithm of router's average queue occupancy easier to configure in an actual network.

Tables 4 and 5 show the impact of using different REM marking probability function parameters. When (q_{min}, q_{max}) is too small, packets will be marked unnecessarily; thus router queue size keeps small, and the receiving rate at the receiver does not reach optimal. The system throughput is optimized when parameter (q_{min}, q_{max}) increases to the value $(0.05C, 0.2C)$. After that point, the receiving rate keeps almost the same although the receiver's Interest window size and the router's queue size increases along with the parameter. In Table 5, the average receiving rate is insensitive to the parameter P_{max}. These two tables show that the setting of REM mark probability function is relatively easy to configure since the throughput is not sensitive to it.

Table 5
Results under different parameter P_{max} in the REM mark probability function. Chunk size is 32 KB, $q_{min} = 0.1C$, and $q_{max} = 0.3C$.

P_{max}	Average receiving Rate (Mbps)	Average interest Window size	Average router Queue size (KB)
0.001	36.97	22.96	225.63
0.002	36.93	21.92	197.69
0.02	36.29	19.54	130.08
0.1	35.95	18.39	95.04
0.2	35.78	18.03	85.58

Table 6
Results under different weight parameter γ in the FISP's queue requirement equation.

Weight Parameter γ	Average receiving Rate (Mbps)	Data delivery Ratio	Average router Queue size (KB)
0.4	27.86	99.7%	1221.7
0.5	37.44	100%	1108.6
0.6	37.44	100%	1053.8
0.8	37.44	100%	959.17
1.5	37.44	100%	635.3

Table 7
Results under different parameter (Q_{min}, Q_{max}) in the FISP delay probability function. Chunk size is 32 KB, $P_0 = 0.3$, and $Q_{release} = 0.2C$.

(Q_{min}, Q_{max})	Average receiving Rate (Mbps)	Data delivery Ratio	Average router Queue size (KB)
$(0.4C, 0.7C)$	37.44	100%	764.78
$(0.6C, 0.9C)$	37.44	100%	1053.8
$(0.8C, 1C)$	37.44	100%	1211.0

We next look at the impact of different parameters in FISP. Here the receiver issues Interests at a constant Interest rate of 160 Interests per second without responding to REM signal. As seen in Section 6.1.2, the router queue would overflow without Interest control in such Interest rate.

The weight parameter γ in Eq. (2) determines FISP'S sensitivity in responding to future chunk arrival in the near future, thus highly affects whether FISP can effectively control the aggressive traffic. As seen in Table 6, when γ is too small, FISP is not sensitive to the size of pending Interest, which causes router queue overflow and packet loss, thus a low throughput; when γ is too large, FISP keeps the router's average queue size very small, which might unnecessarily delay Interests and be in conflict with REM's setting. There, an appropriate value of γ is very important here, but FISP can work well in a relatively large range of γ value as shown in the table.

Tables 7 and 8 illustrate the impact of using different FISP delay probability function parameters. These two tables show that FISP can keep the router from congestion and realize high throughput at different (Q_{min}, Q_{max}) and P_0, which indicates the insensitivity of throughput to FISP delay probability function parameters.

Please cite this article as: F. Zhang et al., Providing explicit congestion control and multi-homing support for content-centric networking transport, Computer Communications (2015), http://dx.doi.org/10.1016/j.comcom.2015.06.019

Table 8
Results under different parameter P_0 in the FISP delay probability function. Chunk size is 32 KB, $Q_{release} = 0.2C$, $Q_{min} = 0.6C$, and $Q_{max} = 0.9C$.

P_0	Average receiving Rate (Mbps)	Data delivery Ratio	Average router Queue size (KB)
0.1	37.44	100%	1062.2
0.3	37.44	100%	1053.8
0.5	37.44	100%	1048.9

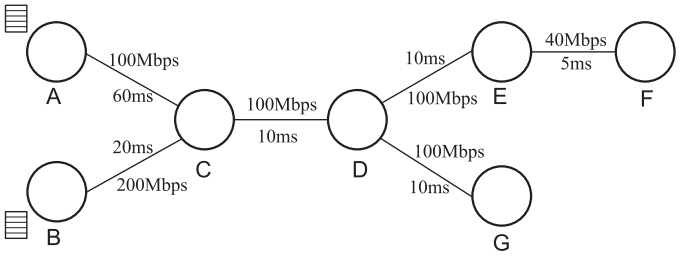


Fig. 8. A larger-scale topology with two sources (A and B) and two receivers (F and G). Link EF is the bottleneck link between A/B and F, while link CD is the bottleneck between A/B and G.

Table 9
Throughput comparison among different transport protocols.

Transport Protocol	Throughput (F) (Mbps)	Throughput (G) (Mbps)	Total (Mbps)
CHoPCoP	36.86	59.44	96.30
ICTP	34.12	13.71	47.83
ICP	7.85	5.90	13.75
HR-ICP	7.68	6.05	13.73

6.2. A multi-source, multi-path scenario

We further look at a larger-scale network topology that has 7 nodes and 2 flows to validate CHoPCoP's capabilities. The topology and detailed link parameters are shown in Fig. 8.

6.2.1. Throughput comparison

We first compare the throughput of CHoPCoP with existing protocols: ICTP, ICP and HR-ICP. Here, receiver F and G simultaneously request two different content files C_1 and C_2 (320M each) respectively. Content C_1 is located at source A, while content C_2 is randomly distributed at the two sources.

Table 9 summarizes the throughput of two receivers under different schemes during the period when both flows are active. CHoPCoP achieves a performance gain of 103% over ICTP, and a factor of 6 over ICP/HR-ICP.

6.2.2. A closer look at CHoPCoP

Next, we take a closer look at how CHoPCoP behaves in such a larger-scale network with multiple flows. In this experiment, receiver F requests a 240 MB content file C_1 from time 0, while G requests a 320 MB content file C_2 from time 20 s. Again, C_1 is at source A while C_2 is randomly distributed at the two sources. Here, we have two content flows, C_1 and C_2, the latter with two sources.

Fig. 9 shows how the receiving rate for both flows changes with time. From time 0 to time 20 s, flow C_1 is the only flow in the network, achieving a throughput of 36.86 Mbps, close to the bandwidth of its bottleneck link EF (40 Mbps). After flow C_2 starts at 20 s, the two flows share the network resources. In this case, their total throughput is limited by the bandwidth of link CD (100 Mbps). As a result, flow C_1's throughput remains around 36.86 Mbps (because of its bottleneck link), while C_2's throughput is 59.44 Mbps. After C_1 ends at 50 s, C_2 owns the resources exclusively, achieving a throughput of 93.65 Mbps, which is close to the bottleneck link bandwidth of 100 Mbps.

This detailed breakdown shows that CHoPCoP achieves efficient resource utilization when flows with different bottleneck links dynamically join and leave the network.

6.3. Multi-homing scenarios

Finally, we evaluate pCHoPCoP's functionality of bandwidth aggregation using three multi-homing network topologies in which the receiver has two interfaces, as seen in Fig. 10. In these scenarios, a 320 MB content is located at both node A and B, and the receiver node E has two interfaces, *int1* and *int2*. Node E requests the content from both of its interfaces, resulting in two CHoPCoP pipes, CHoPCoP1 and CHoPCoP2. Note that *int2* in topology (c) is a Wi-Fi interface.

We plot the throughput and the window size of the two CHoPCoP pipes and aggregated pCHoPCoP, as seen in Figs. 11 and 12. For multi-homing topology (a), since CHoPCoP1 and CHoPCoP2 are symmetric in terms of bandwidth and delay, the throughput and receiver window dynamics are almost the same for these two pipes; for topology (b), since the bottleneck for CHoPCoP2 is link DE with 80 Mbps bandwidth, CHoPCoP2 has a higher throughput and larger receiver window than CHoPCoP1; for topology (c), because of the Wi-Fi link at CHoPCoP2, it has a lower throughput and smaller receiver window than CHoPCoP1. In summary, the results show that while each CHoPCoP controller can fully utilize network capacity over its pipe and keep the network from congested by maintaining a relatively stable receiver window size, the pCHoPCoP engine aggregates the utility of all its corresponding CHoPCoP controllers, thus resulting in much higher overall throughput at the receiver and less time for retrieving the content.

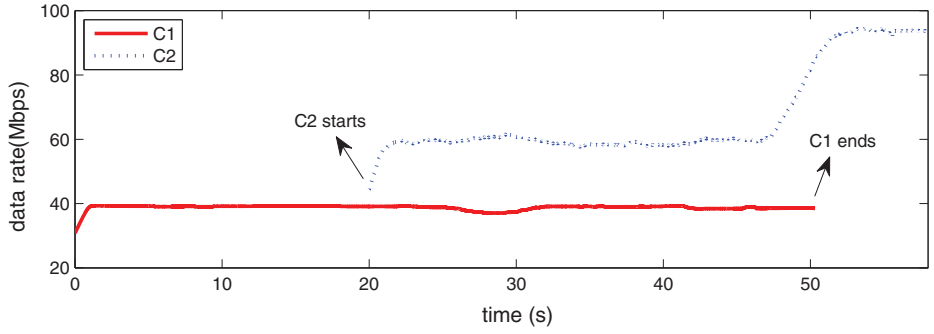


Fig. 9. The receiving rate for the two receivers. F initiates content flow C_1 (with source A) while G initiates content flow C_2 with two sources A and B.

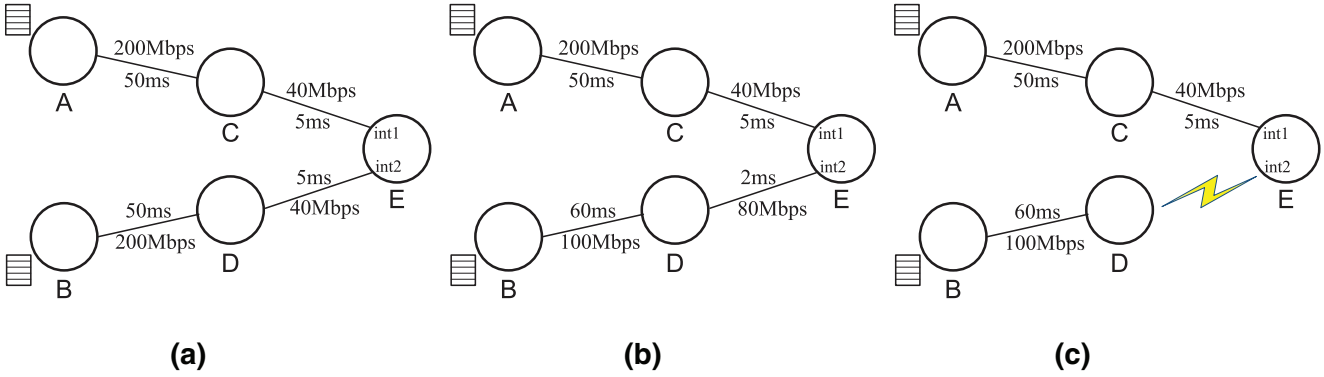


Fig. 10. Three multi-homing topologies. Each consists of two sources (A and B), and the receiver E requests content from them using both of its interfaces.

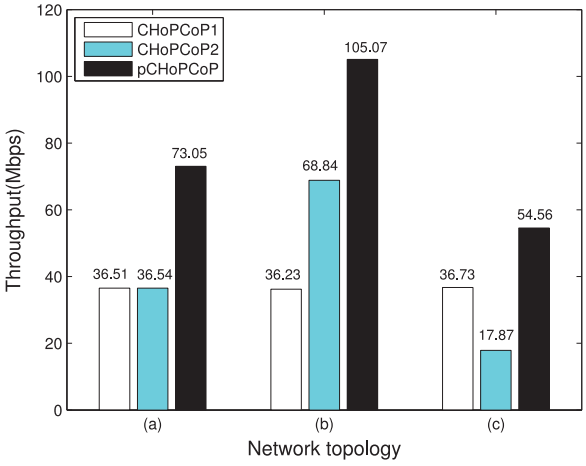


Fig. 11. The throughput of the two CHoPCoP pipes and the aggregated pCHoPCoP at the three multi-homing topologies.

7. Conclusion

In this paper, we present the design, implementation and evaluation of CHoPCoP/pCHoPCoP, a CCN transport protocol. In addition to being receiver driven and hop-by-hop transport, CHoPCoP utilizes explicit congestion signaling to tackle with CCN's multipath nature. We also propose fair share Interest shaping scheme to provide bandwidth sharing among different flows. Moreover, we extend CHoPCoP to a parallelized version, pCHoPCoP, that enables the receiver to utilize all of its network interfaces for bandwidth aggregation.

We have implemented the complete protocol stack using the Click Modular Router, and evaluated its performance on the ORBIT testbed. Our experimental results show that explicit congestion signaling, when coupled with our AIMD-based receiver Interest control, can successfully stabilize the router queues and improve the throughput in a multi-source/multi-path environment. When there are multiple flows in the network, our fair share Interest shaping scheme ensures fairness and provides more efficient resource utilization than earlier quota-based Interest shaping algorithms. Finally, using three multi-homing topologies, we show that pCHoPCoP can effectively realize bandwidth aggregation over all available network interfaces/pipes for the receiver.

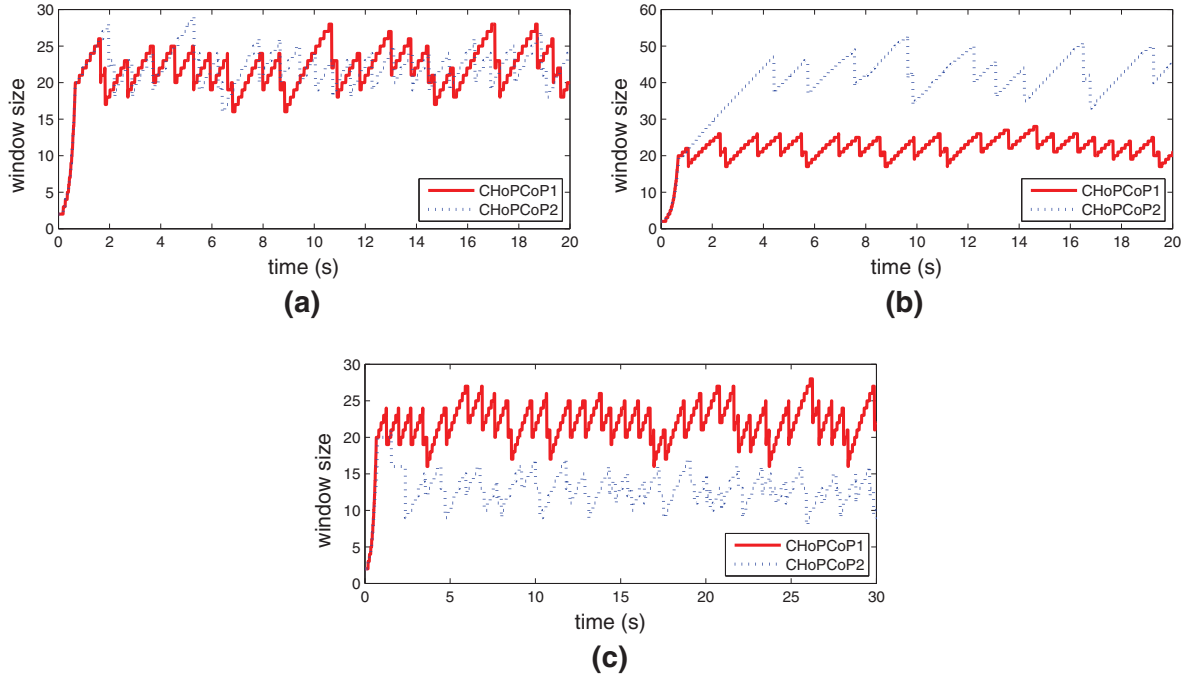


Fig. 12. The receiver window size of the two CHoPCoP pipes at the three multi-homing topologies.

Please cite this article as: F. Zhang et al., Providing explicit congestion control and multi-homing support for content-centric networking transport, Computer Communications (2015), http://dx.doi.org/10.1016/j.comcom.2015.06.019

References

[1] G. Appenzeller, I. Keslassy, N. McKeown, Sizing router buffers, in: Proceedings of the ACM SIGCOMM, 2004.
[2] G. Carofiglio, M. Gallo, L. Muscariello, ICP: Design and evaluation of an interest control protocol for content-centric networking, in: Proceedings of the IEEE INFOCOM workshops, 2012.
[3] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, Multipath congestion control in content-centric networks, in: Proceedings of the IEEE INFOCOM workshops, 2013a.
[4] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, S. Wang, Optimal multipath congestion control and request forwarding in information-centric networks, in: Proceedings of the IEEE ICNP, 2013b.
[5] G. Carofiglio, M. Gallo, L. Muscariello, Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks, in: Proceedings of the ACM ICN, 2012.
[6] CCNx opensource project, (2015) (https://www.ccnx.org/).
[7] D.-M. Chiu, R. Jain, Analysis of the increase and decrease algorithms for congestion avoidance in computer networks, Comput. Netw. ISDN Syst. 17 (1) (1989) 1–14.
[8] S. Floyd, TCP and explicit congestion notification, ACM SIGCOMM Comput. Commun. Rev. 24 (5) (1994) 8–23.
[9] S. Floyd, Recommendation on using the "gentle_" variant of red (2000).
[10] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Trans. Netw. 1 (4) (1993) 397–413.
[11] H.-Y. Hsieh, K.-H. Kim, Y. Zhu, R. Sivakumar, A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces, in: Proceedings of the 9th Annual International Conference on Mobile computing and networking, ACM, 2003, pp. 1–15.
[12] H.-Y. Hsieh, R. Sivakumar, A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts, Wirel. Netw. 11 (1–2) (2005) 99–114.
[13] V. Jacobson, et al., Networking named content, in: Proceedings of the ACM CoNEXT, 2009.
[14] E. Kohler, et al., The click modular router, ACM Trans. Comput. Syst. 18 (3) (2000) 263–297.
[15] A. Kortebi, L. Muscariello, S. Oueslati, J. Roberts, Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing, in: Proceedings of the ACM SIGMETRICS, 2005.
[16] L. Magalhaes, R. Kravets, Transport level mechanisms for bandwidth aggregation on mobile hosts, in: Proceedings of the Ninth International Conference on Network Protocols, 2001, IEEE, 2001, pp. 165–171.
[17] Named data networking project, (2015) (http://www.named-data.net/).
[18] ORBIT testbed, (2015) (http://www.orbit-lab.org/).
[19] S. Oueslati, J. Roberts, N. Sbihi, Flow-aware traffic control for a content-centric network, in: Proceedings of the IEEE INFOCOM, 2012.
[20] I. Psaras, V. Tsaoussidis, Why TCP timers (still) don't work well, Comput. Netw. 51 (8) (2007) 2033–2048.
[21] S. Salsano, et al., Transport-layer issues in information centric networks, in: Proceedings of the ACM ICN, 2012.
[22] M. Shreedhar, G. Varghese, Efficient fair queueing using deficit round robin, in: Proceedings of the ACM SIGCOMM, 1995.
[23] Cheng Yi, et al., A case for stateful forwarding plane, Elsevier Comput. Commun. 36 (7) (2013) 779–791.
[24] F. Zhang, Y. Zhang, A. Reznik, H. Liu, C. Qian, C. Xu, A transport protocol for content-centric networking with explicit congestion control, in: Proceedings of the IEEE ICCCN, 2014.
[25] L. Zhang, Why TCP timers don't work well, in: Proceedings of the ACM SIGCOMM, 1986.

Please cite this article as: F. Zhang et al., Providing explicit congestion control and multi-homing support for content-centric networking transport, Computer Communications (2015), http://dx.doi.org/10.1016/j.comcom.2015.06.019