MSRT: Multi-Source Request and Transmission in Content-Centric Networks

Qing Li*, Bin Gan*,Guangwu Hu*, Yong Jiang*, Qingmin Liao*, Mingwei Xu†
* Graduate School at Shenzhen, Tsinghua University, Shenzhen, China
† Tsinghua University, Beijing, China

***Abstract*—In Content-Centric Networks (CCN), multiple routers may cache the same content, which makes it possible to retrieve the content chunks in parallel. In this paper, we propose Multi-Source Request and Transmission mechanism (MSRT) for CCN. We develop a MinMax problem to compute the optimal solution to retrieve all the chunks from multiple sources in the shortest time. We prove that the problem is NP complete and thus design a fully polynomial-time approximation algorithm to solve this problem. However, the previous works on multipath congestion control cannot be directly employed in MSRT. Therefore, we then propose the Half eXplicit Congestion Protocol (HXCP) to control the request/transmission pace in MSRT. To demonstrate the performance of MSRT, we construct comprehensive experiments. The results show that 1) our scheme reduces the content transmission time to at most 80%; 2) our multipath congestion control scheme HXCP effectively avoids congestion, improves the throughput and guarantees the fairness in the multi-source/multipath scenario.**

I. Introduction

The current Internet architecture, designed nearly 40 years ago, is suitable for the end-to-end communication[1]. However, with the big success of the Internet, a large number of innovative applications are arising. Especially in the recent years, content distribution applications, e.g., video websites, become the main contributor of traffic in the Internet[2]. However, in such applications, users are more concerned about the content, rather than the location of the content[3]. At the same time, the current Internet routers cannot identify the forwarded content, which leads to multiple redundant transmissions of the same content in the network and reduces the network bandwidth utilization.

To solve the problem, a lot of Information Centric Network (ICN) architectures, such as Content Centric Network (CCN) [3], COMET[4], PSIRP/PURSUIT[5] and DONA[6], have been proposed. As one significant ICN architecture, CCN has attracted much attention academically. CCN shifts the network communication model from the current host-based packet delivery to data-driven content delivery by name-oriented routing and in-network caching. In CCN, a content object is assigned a unique name and may be divided into multiple chunks, which can be cached by CCN routers.

With the in-network caching and multiple content repositories in CCN, multiple routers and repositories may cache the same content. The multipath nature of CCN holds considerable promises for improving end-user performance and network resource utilization. However, the previous works [7–12] fail to take full advantage of this character.

In this paper, we propose the mechanism of Multi-Source Request and Transmission (MSRT) to improve the performance of CCN. We first design a multi-source request scheduling scheme for end users to quickly retrieve the content by a point-to-multipoint mode. In our scheme, the $Probe$ packet is employed to explore the chunk distribution status of the requested content. Based on the probed results, we develop a MinMax problem to retrieve data chunks of the content in the shortest time. We prove that the problem is NP complete. We thus design an efficient polynomial-time approximation algorithm to solve this problem.

However, the multi-source transmission model in our scheme only focuses on improving end-user performance, which may cause congestion in the network. While the current congestion control schemes cannot be directly used in MSRT, including the traditional TCP-based congestion control schemes [13, 14] and the explicit congestion control protocols such as eXplicit Control Protocol (XCP) [15]. Meanwhile, the current multipath congestion control protocols [16–18] cannot work efficiently either. Therefore, we propose an efficient multipath congestion control scheme for MSRT. Our multipath congestion control scheme is receiver-driven protocol and involved with explicit router participation. In our design, the scheme employs an Additive Increase and Multiplicative Decrease (AIMD) window control method. Specifically, the scheme performs Additive Increase window control at receiver side and Multiplicative Decrease window control with the cooperation of receivers and routers.

To demonstrate the performance of MSRT, we design and implement a CCN simulator platform. To show the performance of our multipath congestion control scheme, we evaluate the bandwidth utilization and fairness in different scenarios. The experiment results have successfully demonstrated the feasibility of the multi-source request scheduling scheme, which reduces the content transmission time to at most 80% compared with the traditional scheme in CCN. Besides, the simulations have confirmed the ability of our multipath congestion control scheme, which successfully stabilizes the network and improves the throughput in the multi-source/multipath environment. When there are multiple flows in the network, MSRT ensures fairness and provides 10% higher resource utilization than the existing transport protocols.

The remainder of the paper is organized as follows. Section II describes background and motivation. Section III proposes

multi-source request scheduling scheme. In Section IV, the multipath congestion control scheme is presented. Experiment results on the performance of the MSRT are presented in Section V. Section VI concludes the paper.

II. Background and Motivation

CCN operates using two main network primitives: *Interest* and *Data*, both identified by the content name. A user requests the content by the *Interest* message, which is forwarded towards the available content replica according to the name of the requested content. The requested content is then delivered back to the requester by the *Data* message, following the reverse path. There are three core components in the CCN router: Forwarding Information Base (FIB), Pending Interest Table (PIT), and Content Store (CS). FIB is used to forward the Interest message to the potential content source(s). PIT is used to mark where the request comes. CS is used to cache the responded content.

Some works have been devoted to complete CCN architecture, including CCN routing[7], forwarding[8], and caching [9–12]. Although these works improve the efficiency and stability of CCN, they make the transmission model of CCN become multipoint-to-multipoint. For an end user, the request model for content becomes point-to-multipoint. Thus, a multi-source request scheme is required to improve the end-user performance. Though multi-source mobile streaming (MS^2)[19] is proposed to solve the bottleneck issue of the Internet backbone with simultaneous multiple low streaming rate transmissions to mobile users, it does not consider redundant transmissions of popular contents. In MM^3C[20], an MS^2 architecture integrated with CCN for mobile multimedia streaming is proposed. However it is designed to reduce redundant transmissions in the network without considering the end-user performance. Thus, designing a multi-source request scheduling scheme for the end user to retrieve content in the shortest time is significantly important.

The point-to-multipoint transmission model for the end user in CCN makes the traditional congestion control protocols infeasible, including the traditional TCP-based congestion control protocols and the explicit congestion control schemes. The reasons can be concluded as follows:

- Out-of-order delivery cannot be used as an indication of network congestion. A packet might arrive out of order only because the content source of this packet is further from the requester. The retransmission timeout estimation [13] does not span multiple data sources.
- Feedback is not accurate in explicit congestion control protocols. When the requested *Data* packet arrives, multiple copies of the *Data* will be sent from the faces that *Interest*s arrived on. If the *Data* packet carries a feedback, every receiver will uniformly response to it, which is unreasonable as receivers are different.

There are also some congestion control protocols designed specifically for CCN. However, they generally fail to address the fact that the chunks may originate from different sources. ICP[21] and ICTP[22] are TCP-based congestion control protocols for CCN. They set the $Interest$ retransmission timer according to the RTT of received $Data$, which cannot be used as multiple sources exist in MSRT.

There are also a lot of works on multipath congestion control schemes in the traditional network. mTCP [16], MPTCP[17], [18]are transport protocols implemented based on TCP in traditional network. Like most of multipath transport proposals, [18] defines uncoupled congestion control on each path, implemented through separate and independently managed connections (subflows). Each subflow maintains a congestion window as in TCP and perform its own path congestion control. However, the main drawback of uncoupled multipath congestion control consists in inefficient/unfair control of shared bottlenecks.

Contug[23] maintains an independent retransmission timeout for each data source. However, it assumes that before the start of transmission, the receiver has to know the location of each data chunk, and the position will not be changed during transmission, which is generally impossible in CCN. CCTCP [1] is a scalable alternative for the existing proposals. It takes the multiple sources into account and keeps an individual timeout for each expected source. RAAQM [24] performs a per-route control of bottleneck queues along the paths at the receiver, which realizes a receiver-based window congestion control and RTT monitoring on each route distinguishing packets via a route label. Further proposals such as HoBHIS [25] and HR-ICP [26], require each CCN router to keep per-flow state, which strongly affects scalability and deployment in the core Internet routers.

Therefore, designing a novel and efficient multipath congestion control scheme for CCN to cooperate with multi-source request scheduling scheme is also a significant challenge.

III. Design of Multi-Source Request and Transmission Mechanism

A. CCN Extension

In this section, we describe how we extend the CCN architecture to support MSRT.

1) Naming Strategy: We extend the content name to the form of *P:L*, as in DONA [6]. *P* is the name of a content repository (the original content server). *L* is the globally unique identifier (label) of the content. Therefore, a content name is hierarchically structured as *RepositoryA:/youtube.com/videos/a.mpg*. An *Interest* packet with the name of *RepositoryA:/youtube.com/videos/a.mpg/01* should be forwarded to the content repository of *RepositoryA*.

The name can also be set as **:L*, where $*$ means any repository of *L*. An *Interest* packet with the name of **:L* implies that the *Interest* should be forwarded to all the repositories with the content of *L*.

2) Packet Type: We introduce two new packet types: *Probe* and *Information*. Both of them are identified by the content name as the same as *Interest* and *Data*. A user explores the chunk distribution status by broadcasting a *Probe* packet over the available faces. Each CCN node (a server or a CCN router)

R with the corresponding data chunks will send an *Information* packet back to the prober. The *Information* packet contains the chunk distribution status in *R*'s cache or store. Compared with the *Interest*, the name of the *Probe* should always be set as **:L*, implying the *Probe* should be forwarded to all the directions to search content *L*. The *Information* packet is always identified by *P:L*. When *R* is a CCN router, *P* is set according to the corresponding entry in the FIB of *R*. When *R* is a content repository, *P* is set to be *R*.

Besides, we extend the *Information* and *Data* packet header with a field of *FN* (Field of Node type), identifying whether the responding CCN node *R* is a CCN router or a repository. The four extended packets, *Interest*, *Data*, *Probe* and *Information* are shown in Figure 1.

3) FIB Extension: As shown in Figure 2, we extend the FIB entry with a type field of *FR* (Field of Repository), which helps to forward the *Interest* and *Probe* to content repositories. The content repositories should pro-actively advertise their existence via a CCN broadcast. For example, Repository R_A has the content of L (/youtbe.com/video/a.mpg). R_A advertises the information of R_A:L in the CCN network via a CCN broadcast. R_B does the same thing. As shown in Figure 2, a CCN router maintains the two entries of R_A:L and R_B:L in the FIB. Therefore, the CCN router can forward the corresponding *Interest* and *Probe* packets accordingly. Specially, the router should forward *Probe* packets with name $*$:L to all available faces towards the content L in the FIB.

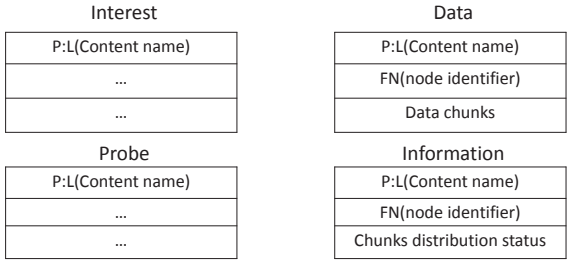


Figure 1. The four types of packets in MSRT

Forwarding Information Base (FIB)

FR(Repository)	Name	Forwarding face
R_A	youtube.com/videos/a.mpg	0
R_B	youtube.com/videos/a.mpg	1

Pending Interest Table (PIT)

FT(Type)	Lifetime	Name	Requesting face
Probe	10	youtube.com/videos/a.mpg	0
Interest	5	youtube.com/videos/a.mpg/01	0

Figure 2. FIB and PIT in MSRT

4) PIT Extension: As shown in Figure 2, we extend the PIT entry with a type field *FT* (Field of Type), which helps to distinguish *Interest* and *Probe*. When a CCN router receives a *Probe* for a content not in its cache, it forwards the *Probe* according to the FIB. When the corresponding *Information* packets travel back, they are forwarded according to the PIT. The PIT entry with *FT* of "Probe" is different from the entry with *FT* of "Interest". The "Probe" entry in the PIT will not be deleted with the reception of the *Information* packet. It will only be deleted when it is expired. Thus, the receiver can collect multiple *Information* packets from different data sources with one *Probe* packet.

B. Chunk Distribution Status Explore Method

1) CCN Router and Repository Functions: When a CCN router receives a *Probe* for a content not in its cache, it forwards the *Probe* according to the FIB and installs a PIT entry for the *Probe* packet. If the router has the content in its cache, it extracts repository field *FR* from the corresponding FIB entry and generates an *Information* packet of *P:L* where the *P* is set as to be *FR*. The *FN* field of the *Information* packet is set to be the CCN router.

When a content repository receives a *Probe*, it generates an *Information* packet with *P:L* where *P* is the repository itself. In this case, the *FN* field of the *Information* packet is also set to be the repository itself. The *Information* packet is forwarded back to the request by the PIT entries along the path of the *Probe* packet. The PIT entry whose type field *FT* is "Probe" will only be deleted when it is expired.

2) User Functions: In CDSEM, before requesting the content, the user first sends a *Probe* packet with the name of **:L* to explore the corresponding chunk distribution status.

When the corresponding *Information* packets arrive, the user extracts the chunk distribution status of the content in each data source. The *Information* packet records the content repository and the node identifier *FN* of the data source. Then the user sends the *Interest*s to get different chunks from different content sources. Each communication between the user and data source is controlled by a separate sliding window. In CDSEM, we use *FN* to identify different sliding windows. After *Data* packets arrive, we update the round trip time (RTT) of the corresponding data source.

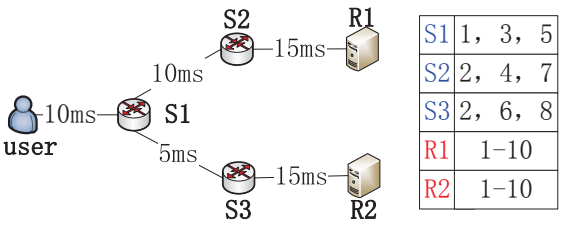
An example of CDSEM is shown in Figure 3 and Figure 4. There is a content file that is divided into ten chunks. On the path from the user to content repositories, there are some data chunks cached in routers (as shown in Figure 3). The specific process of CDSEM is shown in Figure 4.

It is necessary to mention that the chunk distribution status of the content dynamically changes, influenced by caching policies, changing traffic, and cache sizes. Therefore, we periodically send the *Probe* to explore the latest chunk distribution status of the content.

C. Multi-Source Request Schedule

In this section, we formalize the problem of the multi-source request schedule as a MinMax problem.

We first take the topology and the content distribution of Figure 3 as an example. A content file contains 10 equal-size data chunks. There are five data sources S_1, S_2, S_3, R_1 and R_2 for the content. S_1 has data chunks $\{1, 3, 5\}$, S_2 has data chunks $\{2, 4, 7\}$, S_3 has data chunks $\{2, 6, 8\}$, and R_1



S1	1, 3, 5
S2	2, 4, 7
S3	2, 6, 8
R1	1–10
R2	1–10

Figure 3. A topology of chunk distribution status

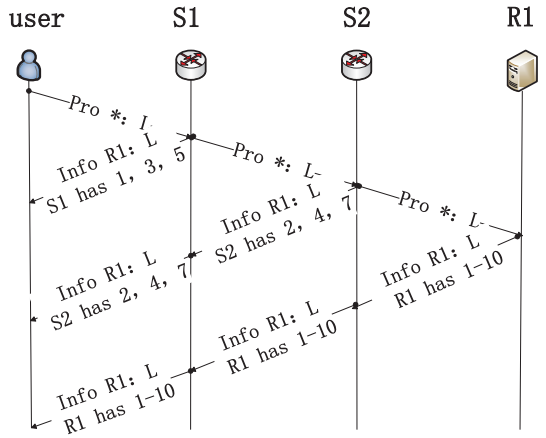


Figure 4. An example for CDSEM

and R_2 have whole content. The RTTs from the user to each data source are respectively rtt_1, rtt_2, rtt_3, rtt_4 and rtt_5. The window sizes for the data sources are respectively win_1, win_2, win_3, win_4 and win_5. The problem is how to retrieve the whole content from the five data sources in the shortest time. The more general case could be modelled as a MinMax problem.

Based on the results of CDSEM, we obtain window sizes, RTT values and chunk distribution status of all the data sources. We accordingly develop a MinMax problem to retrieve the content in the shortest time. We prove it to be NP-complete. Then, we design a heuristic algorithm and a fully polynomial time approximation algorithm to solve the problem.

1) Optimal Content Retrieval Problem: A content C is composed of a number of data chunks with the same size M. C=$\{c_1, c_2, ... c_p\}$, where c_i is a chunk and p is number of chunks in C. There are a set of content sources S=$\{s_1, s_2, ... s_q\}$ where s_k is a content source and q is the source number. Each source s_k has a corresponding RTT rtt_k and a window size win_k to the end user. s_k also denotes the set of chunks it stores. For example, s_k=$\{c_{k1}, c_{k2}, ...c_{kn}\}$ means the content source s_k has the chunks, c_{k1}, c_{k2}, ... c_{kn} in its storage.

An available retrieval solution is to find a set of chunk group CG=$\{cg_1, cg_2, ... cg_q\}$, where $cg_i \subseteq s_i$, $\cup_{1\le i\le q} cg_i = C$ and $cg_i \cap cg_j \doteq \emptyset\ (\forall i \neq j)$. cg_i means the user retrieves the chunks in cg_i from s_i. Let x_{ij} denote whether c_i is retrieved from s_j in the solution. If $c_i \in cg_j$, $x_{ij} = 1$; else, $x_{ij} = 0$. $T = max\{rtt_j + \frac{M \times rtt_j}{win_j} \sum_{i=1}^{p} x_{ij},\ j \in [1, 2, ..., q]\}$ is the total retrieval time for the retrieval solution. Our objective is to find an optimal available solution with the shortest total retrieval time. The **optimal content retrieval problem** is formalized as follows.

$$T = min \quad max_{j\in[1,2,...,q]}\{rtt_j + \frac{M \times rtt_j}{win_j} \sum_{i=1}^{p} x_{ij},\}$$

$$s.t : \sum_{j=1}^{q} x_{ij} = 1, i \in [1, 2, ..., p] \tag{1}$$

$$x_{ij} = \begin{cases} 1, & c_i \in cg_j \quad and \quad c_i \in s_j \\ 0, & otherwise \end{cases}$$

The constraint $\sum_{j=1}^{q} x_{ij} = 1$ indicates that data chunk c_i only belongs to one chunk group. If data chunk c_i belongs to chunk group cg_j, $x_{ij} = 1$, otherwise $x_{ij} = 0$. An optimal solution of this problem corresponds to an optimal way to retrieve the content in the shortest time.

Theorem 1. *The optimal content retrieval problem is NP-complete.*

We prove Theorem 1 by a reduction from the 3-dimensional matching problem, which is known to be NP-complete:

Instance 1. *Disjoint sets $A = \{a_1, ..., a_n\}$, $B = \{b_1, ..., b_n\}$, $C = \{c_1, ..., c_n\}$ and a family $H = \{T_1, ..., T_m\}$ of triples with $|\ T_i \cap A\ | = |\ T_i \cap B\ | = |\ T_i \cap C\ | = 1$ for i=1,...,m.*

Question 1. *Does H contain a matching, i.e., a subfamily H' for which $|H'| = n$ and $\cup_{T_i \in H'} T_i = A \cup B \cup C$.*

Proof of Theorem 1. Given an instance of the above 3-dimensional matching problem, we construct an instance of the multi-source request problem with m data sources and $3n + l(m - n)$ data chunks when $m > n$, $l > 1$ and m, n, l are integers. We define the $3n$ chunks which in $A \cup B \cup C$ as element chunk and the $l(m - n)$ chunks which is not in $A \cup B \cup C$ as dummy chunk. Data source s_i caches or stores all the $l(m-n)$ dummy chunks and all the $3n$ element chunks for $i = 1, ..., n$. And the other $m - n$ data sources cache or store all the dummy chunks. Data sources s_i for $i = 1, ..., n$ can transmit each chunk in one time unit and data sources for $i = n + 1, ..., m$ can transmit each chunk in $\frac{3}{l}$ time unit. It is quite simple to show that there is a minimal time at most 3 for the multi-source request problem if and only if there is a 3-dimensional matching. In the schedule, we request element chunks a_x, b_y and c_z which correspond to the triple $T_i = \{a_x, b_y, c_z\}$ from data source s_i for $i = 1, ..., n$, and request all the $l(m - n)$ dummy chunks from data sources s_i for $i = n + 1, ..., m$ for each data source transmits l dummy chunks. Thus, a 3-Dimensional matching problem can be reduced to a multi-source request problem where $rtt_j = 1$ for $j = 1, ..., n$, $rtt_j = \frac{3}{l}$ for $j = n + 1, ..., m$, and win_j=M for $j = 1, ..., m$. □

2) Minimum rEtrieval Time Algorithm: In this section, we propose a heuristic algorithm which is the Minimum rEtrieval Time Algorithm (META) to solve the MinMax problem described in Equation 1. The main idea of the algorithm META is to calculate the retrieval time for the data chunks one by one sequentially. In each round, we choose the data source with the minimum retrieval time for the current chunk.

The procedure of META is described as follows. We set up a data structure named Retrieval Time (RT) which is a $1 \times q$ matrix and records the current retrieval time at each data source. In stage i for data chunk c_i for $i = 1, ..., p$, we calculate the expected retrieval time for the data chunk c_i when it is assigned to each available data sources, and choose the data source which has the minimum expected retrieval time to request and send the Interest to it. Then, we update the retrieval time of the chosen data source in RT. The pseudo-code of the META is shown in Algorithm 1. In this algorithm, in each stage i for the data chunk c_i, we choose to request it from to the available data source with the minimum retrieval time. Therefore, the time complexity of META is $O(pq)$.

Algorithm 1 Minimum rEtrieval Time Algorithm

Input: Data chunks C={c_1, c_2, ... c_p}, Data sources $s_j, j \in [1, 2, ..., q]$, stores data chunks s_j={c_{j1}, c_{j2}, ... c_{jn}} and each data source s_j, has a RTT rtt_j and a window size win_j;
Output: CG={cg_1, cg_2, ... cg_q} and the minimal time T_{opt}.

```
Initialize the RT = (rtt_1, rtt_2, ..., rtt_q);
for i = 1 to p do
    min = +∞, index = 0
    for j = 1 to q do
        if c_i ∈ s_j and min > RT[j] + (M×rtt_j)/win_j then
            min = RT[j] + (M×rtt_j)/win_j, index = j;
        end if
    end for
    cg_index = cg_index ⋃ c_i;
end for
T_opt = max_{j∈[1,q]} RT[j]);
Return T_opt and CG;
```

3) Binary Search based on Hungarian Algorithm: In this section, we propose another Binary Search based on Hungarian Algorithm (BSHA) to get a better solution for the optimal content retrieval problem. BSHA can get a better solution than META and the time complexity is still polynomial, $O(p^3 log_2(1/\varepsilon))$, where p is the number of data chunks and ε is a constant.

Binary search is known as a half-interval search algorithm. It halves the feasible region to check with each iteration. Locating an item (or determining its absence) takes logarithmic time.Hungarian algorithm is a classical algorithm used to solve the maximum bipartite graph matching problem in polynomial time. In this section, we solve the optimal content retrieval problem by Binary Search based on Hungarian Algorithm (BSHA).

In each step of the BSHA, the algorithm has a feasible interval [a, b] where the optimal retrieval time T_{opt} is in the interval, and a, b are constants. At the beginning of BSHA, we initialize $a = 0$ and b as the time of a max feasible solution T_{fea} to retrieve all the chunks. Then, we start the binary search to half the interval by finding a feasible solution for the retrieval time of $(a + b)/2$. If we find a feasible solution for it, we set b to $(a + b)/2$, otherwise we set a to $(a + b)/2$. The optimal retrieval time is still in the interval. We employ the Hungarian algorithm to compute whether there is a feasible solution in a specific time or not.

The process of finding a feasible solution for a specific time can be described as follows: first, we can request at most $d_j = \lfloor \frac{win_j \times (T - rtt_j)}{M \times rtt_j} \rfloor$ data chunks from each data source s_j, $j = 1, ..., q$. Thus we allocate d_j resource units to data source s_j. We have $\sum_{j=1}^{q} d_j$ resource units in all. Then, we construct a bipartite graph $G = (V, E)$ where the vertices V can be divided into two disjoint sets A and B where A is the set of data chunks and B is the set of resource units. The set E denotes the edges between data chunks and resource units, indicating that the resource units can be allocated to the corresponding data chunks. Computing whether there is a feasible solution for a specific time or not is equivalent to find a max matching whose edges number is equal to the number of data chunks for the bipartite graph. Hungarian algorithm is a very effective algorithm to seek the max matching. The algorithm BSHA would end with a constant $\varepsilon > (b - a)$ and we choose b as the approximate result of the optimal retrieval time and the max matching for the bipartite graph as the final chunk groups.

Algorithm 2 Binary Search based on Hungarian Algorithm

Input: C={$c_1, c_2, ...c_p$}, s_j={$c_{j1}, c_{j2}, ...c_{jn}$}, $j \in [1, 2, ..., q]$
Output: CG={cg_1, cg_2, ... cg_q} and the minimal time T_{opt}.

```
Initialize a = 0, b (a random feasible time), Constant ε;
while ε < b − a do
    T = (b + a)/2;
    for j=1 to q do
        d_j = ⌊(win_j × (T − rtt_j))/(M × rtt_j)⌋;
        RU_j={rs_jk | k ∈ [1, d_j]};
    end for
    Construct a bipartite graph G = (A, B, E), A = {c_1, c_2, ..., c_p}, B = {RU_1, RU2, ..., RUq}. If there is an edge between node c_i and node ru_jk, (k ∈ [1, d_j]) for resource units of data source s_j, e[c_i][ru_jk] = 1.
    Judge whether there is a feasible solution for T using Hungarian Algorithm.
    if there is feasible solution array pred then
        b = (b + a)/2;
    else
        a = (b + a)/2;
    end if
end while
Construct CG according to pred.
for j = 1 to q do
    for k = 1 to d_j do
        cg_j = cg_j ⋃ c_{pred[ru_jk]};
    end for
end for
Return T_opt=b and CG;
```

Algorithm 3 Hungarian Algorithm

```
Initialize all = 0.
for i=1 to p do
    unlabel all nodes in B.
    set pred[ru_jk]=0 for j ∈ [1,m], k ∈ [1,d_j].
    if find(i) then
        all=all+1.
    end if
end for
if all < p then
    there is a feasible solution for T.
else
    there is not a feasible solution for T.
end if
Return array pred.
Function find(x)
for j = 1 to q do
    for k = 1 to d_j do
        if e[c_x][ru_jk] == 1 and ru_jk is unlabeled then
            label node ru_jk
            if pred[ru_jk]==0 or find(pred[ru_jk]) then
                pred[ru_jk]=x.
                return true
            end if
        end if
    end for
end for
return false
end Function
```

IV. Multi-path Congestion Control

The multi-source transmission model in MSRT makes the method of traditional congestion control invalid. Therefore, we propose Half eXplicit Congestion Control Protocol (HXCP), which is receiver-driven and involved with explicit router participation protocol.

A. Interest Control Overview in HXCP

Data are requested via *Interest* (one *Interest* per *Data* packet) in the order of the multi-source request scheduling, according to a window-based Additive Increase Multiplicative Decrease (AIMD) approach. As shown in Figure 5, in our design, the AIMD controller is separated into two parts: additive-increase *Interest* window controller in the receiver and multiplicative-decrease *Interest* window controller in CCN routers. CCN routers send feedback messages to the receiver and the receiver adjusts its window size accordingly.

Now we give an overview of HXCP. The congestion window, win, is kept at the user side and defines the maximum bytes of un-responded *Interest*s the user is allowed to send in a round trip time. win is increased by $\frac{\eta}{win}$ upon each normal *Data* packet reception whose feedback is positives. This corresponds to an increment of η (bytes of one *Interest* by default) each time a complete window of Interest is acknowledged by *Data* reception.

In HXCP, the CCN router can also send feedbacks to the receiver, as it is one end of the communication. Receivers maintain their round trip times, window sizes, and inter-Interest times. They share these information with CCN routers via a congestion header in packets. The CCN Routers monitor the input packets rate to each of their output queues. Based on the difference between the link bandwidth and its input packets rate, the CCN router determines whether to decrease the window sizes of the corresponding flows. The router achieves this by annotating the congestion header of *Data* packets. Feedback is divided between flows based on their actual throughput and RTT so that the system guarantees fairness. A more congested CCN router later in the path can overwrite the feedback. When the feedback reaches the receiver, the receiver updates its window size win accordingly.

B. Function Description of HXCP

1) The Congestion Header: Each HXCP packet carries a congestion header $< win, rtt, feedback >$, which is used to transport the flow states and feedbacks between CCN routers and the receiver. The field win is the receiver's current window size to the data source and the rtt is the round trip time from the receiver to the data source. These parameters are set by the user and never modified during transmission. As for the feedback field, CCN routers along the path modify this field to directly control the congestion window in receiver.

2) Functions of Receivers: The receiver is responsible for maintaining three parameters for each data source in the congestion header. Upon *Data* arrival, the receiver checks the congestion feedback in *Data* packet. If it is an increasing feedback, the receiver increases win by $\frac{\eta}{win}$. Otherwise, the receiver adjusts the window size accordingly. In addition, the receiver updates the round trip time that is in correspondence with the data source whose identifier is the same as the node identifier field in the *Data* packet header.

3) Functions of CCN Routers: This part of content is the same as the router functions in eXplicit Congestion Control Protocol. The difference between them is that the CCN router may be the data source. Thus, the CCN router may match the *Interest* and respond it with the *Data* packet. The CCN router is also responsible for coping the congestion header in the *Interest* into the *Data* packet. The CCN router calculates and sends the feedback to the receiver by annotating the congestion header of *Data* packets.

V. Evaluation and Analysis

In this section, we evaluate MSRT by comprehensive experiments. Our evaluation focuses on the capacity of multi-source request mechanism and multi-path congestion control mechanism in CCN. We evaluate the makespan (the content retrieval time) of multi-source request mechanism with our two different algorithms META and BSHA, compared with the original approach in CCN. We also evaluate the execution time of two algorithms META and BSHA.

Additionally, we evaluate HXCP's capability in stabilizing network condition. We conduct detailed experiments on several simplified but representative network topologies and compare HXCP with existing CCN transport protocol RAAQM [24] and MPTCP [27].

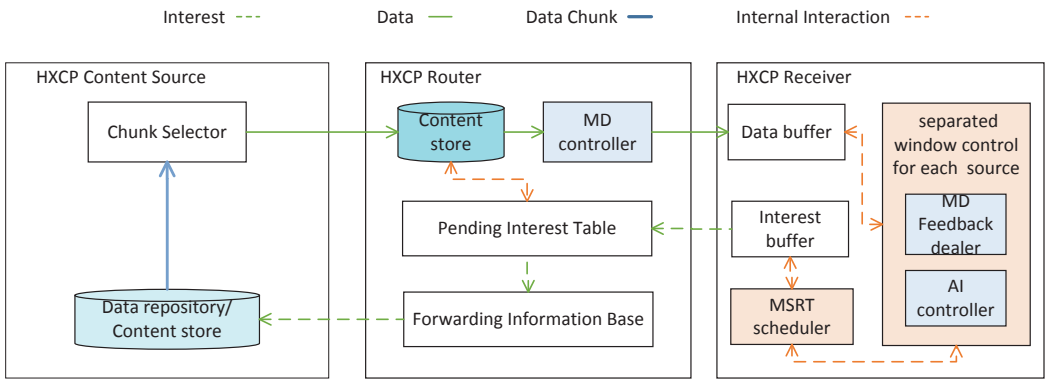


Figure 5. Functional modules of HXCP content source, router and receiver.

Our CCN simulation platform provides a network environment to support CCN protocol evaluation. The platform runs at two servers of Huawei RH2288 with the cpu of E5-2600 v2, the memory of 8GB DDR3., as shown in Figure 6.

Figure 6. The servers of simulation platform.

A. Multi-source Request Scheme Simulation

We first evaluate mechanism of MSRT in a four-layer binary tree. The number of data sources is set to be four or eight. The number of chunks is set from 100 to 900.

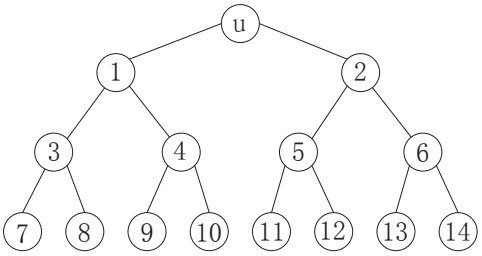


Figure 7. An experiment topology for multi-source request scheduling scheme with CDSEM

1) Makespan: We quantify the makespan of the multi-source mechanism with META and multi-source mechanism with BSHA, comparing with retrieving content from a single source. We set up a non-cooperative receiver that generates *Interest*s at a constant *Interest* rate (CIR), even when the router has sent congestion feedback message. We set CIR as 200 *Interest*s per second in each run, requesting a content file. As shown in Figure 8, the makespan with multi-source request is better than retrieving content without multi-source request. When the chunk number increases, the makespan performance improves significantly. The reason is that the probe is more worthwhile when the content is larger. there is an extra time to explore the chunk distribution status of content to be requested. When the number of the data chunks is large, the transmission time reduces greatly and the time overhead of CDSEM can be covered or even ignored.

As for the multi-source request with META and BSHA, the experiment results show that BSHA can obtain better results for multi-source request.

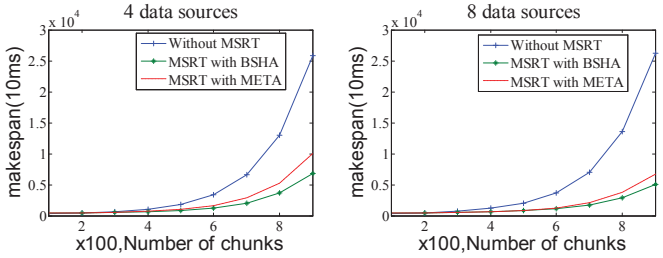


Figure 8. Makespan of MSRT (with BSHA or META) compared with single-source request without MSRT

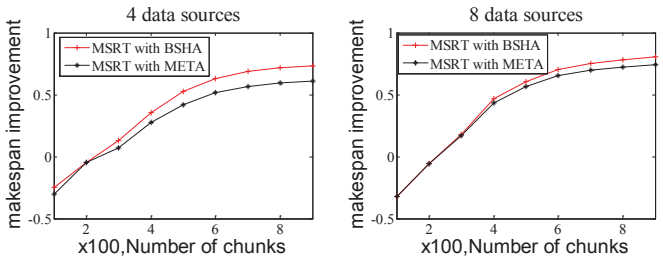


Figure 9. Makespan improvement of MSRT

2) Algorithm Execution Time: In this section, we present the experiment results of Algorithm META and BSHA with

4/8 data sources and different number of data chunks (100, 200, 400, 800). The results are shown in Table I. The experiment results confirm that the execution time of META is much smaller than BSHA, which is consistent with our analysis. Besides, when the problem scale increases, the execution time of BSHA increases sharply.

Table I
ALGORITHM EXECUTION TIME (MS)

Source x Chunk	META	BSHA	S x C	META	BSHA
4x100	1.45	21.32	8x100	3.85	93.73
4x200	3.29	87.92	8x200	7.84	204.75
4x400	7.45	475.84	8x400	17.31	1776.72
4x800	16.48	5042.75	8x800	22.65	5423.26

The results of makespan and execution time of MSRT with META and BSHA reveal that we should choose different algorithms depending on different network scenarios. Specifically, the META algorithm should be chosen when the receiver's computing capacity is relatively low, the requested content is larger with more data chunks and stored in more data sources. Otherwise, BSHA should be chosen for MSRT.

B. Multi-path Congestion Control Scheme Simulation

1) A Single-Receiver, Two-Source Benchmark Scenario: We conduct several benchmark experiments using a four-node topology, shown in Figure 10. Here, the receiver node *A* is connected to the router *B* via a long Internet connection (with 100Mbps bandwidth and 50ms latency), and the router is connected to the sources *C* and *D* via a local Internet access (with a 40Mbps bandwidth and a 5ms latency). We set the outbound data queue size of the router to be 3000KB. We consider a 320MB content file with 10,000 chunks in total. Each chunk is 32KB in size, stored in source *C* and source *D*.

First, we compare HXCP with existing protocols: RAAQM and MPTCP in Figure 11, which shows how the receiver window size and the transient receiving data rate change over time. The results show that HXCP significantly outperforms RAAQM and has the same throughput as MPTCP. However, HXCP is much smoother at receiver side, with an average size of 44.43; the receiving data rate is much higher (also smoother), with 78.52Mbps at the steady state. The throughput of HXCP is 8.6% and 110.2% higher than that of MPTCP and RAAQM respectively.

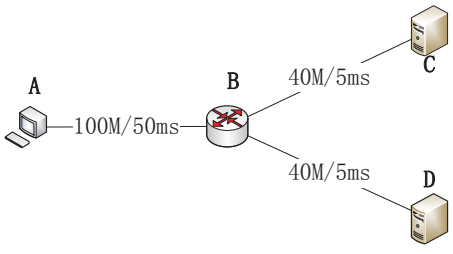
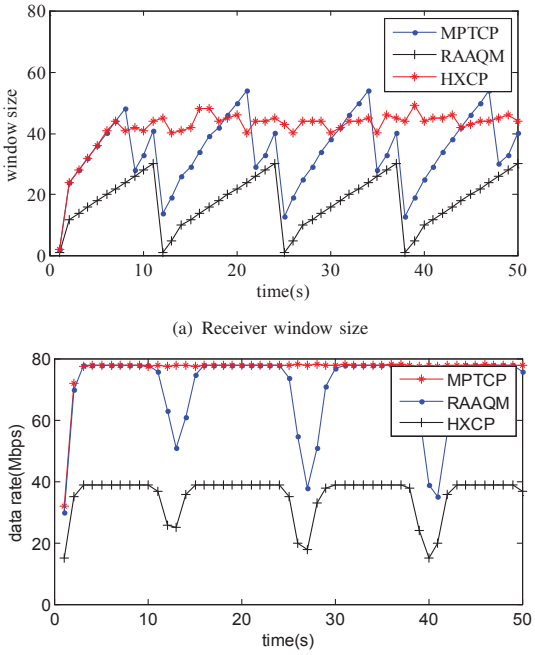


Figure 10. A topology with two sources and one requester.



(a) Receiver window size

(b) Receiving data rate

Figure 11. The receiver window size and receiving rate.

C. A Multi-Flow Scenario

We further evaluate how HXCP behaves when multiple flows exist. We conduct the experiments on the topology in Figure 12, which consists of two receivers (A and B). These two receivers requesting different content files $f1$ and $f2$ resulting in two flows in the topology. File $f1$ is hosted by E, and $f2$ is hosted by F and G.

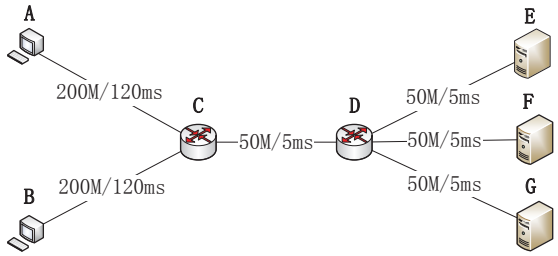
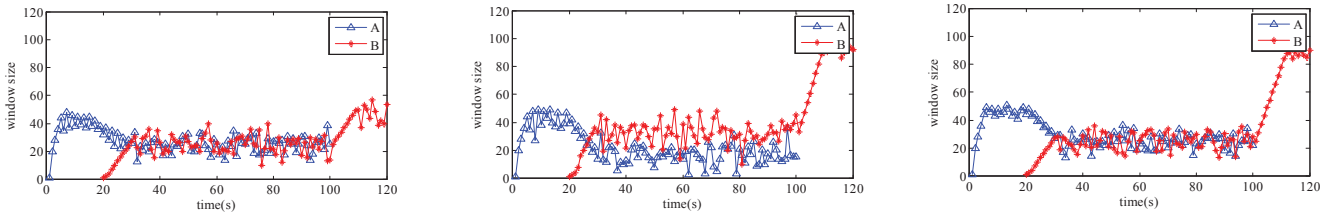


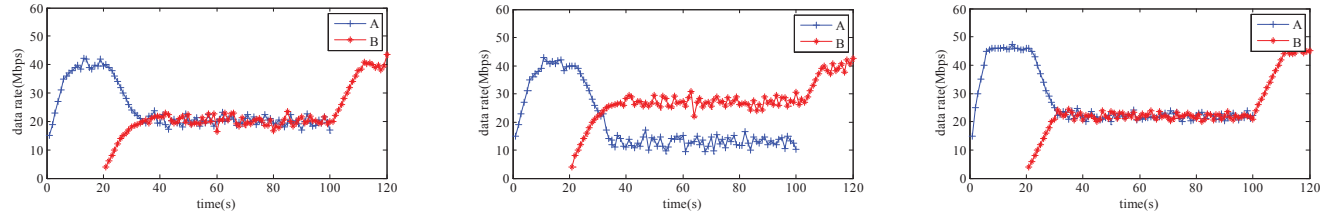
Figure 12. Multiple flow topology where A requests file $f1$ from source E and B requests file $f2$ from source F and source G.

1) Fairness between the Flows: First, we compare HXCP with existing protocols: RAAQM, MPTCP. We will show that HXCP is fair in sharing network resource in CCN. We show the performance of the three protocols when adapting to the number of flows in the network. For this purpose, in the topology in Figure 12, we have receiver A requests content file f1 from the source E and receiver B requests content file f2 from source F and source G. A starts its request at time 0 and stops at time 100 while B starts at time 20. Figures 13(a)-(c) show the results of the receiver window size and



(a) Receiver window size for RAAQM (b) Receiver window for MPTCP (c) Receiver window size for HXCP

Figure 13. The receiver window size of RAAQM, MPTCP and HXCP.



(a) Receiving data rate for RAAQM (b) Receiving data rate for MPTCP (c) Receiving data rate for HXCP

Figure 14. Receiving rates of RAAQM, MPTCP and HXCP.

Figures 14(a)-(c) show the results of receiving data rate of each receiver.

Table II summarizes the throughput of two receivers under different schemes during the period when both flows are active.

Table II
THROUGHPUT IN MULTI-FLOW SCENARIO ($Mbps$).

Protocol	Throughput(A)	Throughput(B)	Total
RAAQM	20.53	20.53	41.06
MPTCP	13.76	26.44	41.20
HXCP	22.52	22. 52	45.04

Before B starts, there is only one flow, initiated by A, and as a result, this flow owns the resources exclusively. The results show that HXCP outperforms the RAAQM and MPTCP in throughput. Specifically ,the throughput of HXCP for receiver A is about 45.53Mbps at the steady state and is 11% higher than that of RAAQM and MPTCP. Additionally, the window of the HXCP is much smoother than RAAQM and MPTCP, which is consistent with the results in the single-receiver, two-source benchmark scenario. RAAQM and MPTCP have similar results with each other because there is only one receiver and one source.

After B starts there is two flows, and the two flows shares the resources. The results reveal that HXCP and RAAQM can share the bandwidth for the two receivers in fair where each of the receiver gets almost the same data rate. However, the receiving data rate of HXCP is a little higher (also smoother), at the steady state than that of RAAQM. And the results also indicate the unfairness of MPTCP where the receiving data rate of B is about two times of A.

The inferior performance of RAAQM and MPTCP can be explained below. In RAAQM, it does not exploit the multi source request for receiver B which makes that it treats receiver A and B equally. In MPTCP,it makes use of the two path for receiver B and one path for receiver A which makes it to allocate different bandwidth for receiver A and B.

After A stops, there is only one flow that has two sources. The results are very similar to the results of single receiver and two sources scenario shown in Figures 11(a)-(b).

VI. CONCLUSION

With the in-network cache and multiple repositories in CCN, the same content may be cached or stored at multiple locations, which makes it possible to retrieve content chunks in parallel. In this paper, we present the Multi-Source Request and Transmission (MSRT) mechanism for end users to retrieve content in the shortest time. In MSRT, the *Probe* packet is employed to explore the chunk distribution status of the content to be requested. Based on the probed results, we formalize an optimization problem to compute the optimal solution to retrieve data chunks in the shortest time. We proved the problem to be NP complete and provide the efficient approximation algorithm.

The existing multi-source request scheduling mechanism makes the previous multipath congestion protocols invalid, Therefore, we propose our own new and efficient multipath congestion control mechanism HXCP to cooperate with multi-source request scheme of MSRT. HXCP is receiver-driven and involved with explicit router participation, which performs separated AIMD window control for each data source.

VII. ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China under grant No. 61402255, the R&D Program of Shenzhen under grant No. JCYJ20150630170146830, No. ZDSYS20140509172959989.

References

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

[2] C. V. N. I. Cisco, "Global mobile data traffic forecast update, 2013–2018," *white paper*, 2014.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of CoNEXT*, Rome, Italy, Dec. 2009.

[4] W. K. Chai, N. Wang, I. Psaras, G. Pavlou, C. Wang, G. G. De Blas, F. J. Ramon-Salguero, L. Liang, S. Spirou, A. Beben *et al.*, "Curling: Content-ubiquitous resolution and delivery infrastructure for next-generation services," *IEEE Communications Magazine*, vol. 49, no. 3, pp. 112–120, 2011.

[5] N. Fotiou, D. Trossen, and G. C. Polyzos, "Illustrating a publish-subscribe internet architecture," *Telecommunication Systems*, vol. 51, no. 4, pp. 233–245, 2012.

[6] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *Proceedings of SIGCOMM*, Kyoto, Japan, Aug. 2007.

[7] A. K. M. M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "NLSR: Named-data link state routing protocol," in *Proceedings of ACM SIGCOMM Workshop on ICN*, Hong Kong, Aug. 2013.

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

[9] S. Guo, H. Xie, and G. Shi, "Collaborative forwarding and caching in content centric networks," in *Proceedings of IFIP NETWORKING*, Prague, Czech Republic, May 2012.

[10] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in information-centric networking," in *Proceedings of ICCCN*, Shanghai, China, Aug. 2014.

[11] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of ACM SIGCOMM Workshop*, Helsinki, Finland, Aug. 2012.

[12] I. Psaras, R. G. Clegg, R. Landa, W. K. Chai, and G. Pavlou, "Modelling and evaluation of ccn-caching trees," in *Proceedings of IFIP NETWORKING*, Valencia, Spain, May 2011.

[13] S. Floyd and T. Henderson, "The newreno modification to tcp's fast recovery algorithm," IETF Draft, 1999.

[14] V. Jacobson, "Congestion avoidance and control," in *Proceedings of ACM SIGCOMM*, Stanford, USA, Aug. 1988.

[15] D. Katabi, "Decoupling congestion control and bandwidth allocation policy with application to high bandwidth-delay product networks," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.

[16] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths," in *Proceedings of ATEC*, Boston, USA, Mar. 2004.

[17] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath tcp," in *Proceedings of NSDI*, Boston, USA, Mar. 2011.

[18] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, "Multi-path tcp: A joint congestion control and routing scheme to exploit path diversity in the internet," *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 1260–1271, 2006.

[19] T. Taleb and K. Hashimoto, "MS2: A new real-time multi-source mobile-streaming architecture," *IEEE Transactions on Broadcasting*, vol. 57, no. 3, pp. 662–673, 2011.

[20] D. O. Mau, T. Taleb, and M. Chen, "MM3C: Multi-source mobile streaming in cache-enabled content-centric networks," in *Proceedings of GLOBECOM*, San Diego, USA, Dec. 2013.

[21] G. Carofiglio, M. Gallo, and L. Muscariello, "Icp: Design and evaluation of an interest control protocol for content-centric networking," in *Proceedings of Computer Communications Workshops (INFOCOM WKSHPS)*, Orlando,USA, Apr. 2012.

[22] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, and N. Blefari-Melazzi, "Transport-layer issues in information centric networks," in *Proceedings of ACM SIGCOMM Workshop on ICN*, Helsinki, Finland, Aug. 2012.

[23] S. Arianfar, P. Nikander, L. Eggert, and J. Ott, "Contug: A receiver-driven transport protocol for content-centric networks," in *Proceedings of ICNP*, Kyoto, Japan, Oct. 2010.

[24] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papali, "Multipath congestion control in content-centric networks," in *Proceedings of Computer Communications Workshops (INFOCOM WKSHPS)*, Turin, Italy, Apr. 2013.

[25] N. Rozhnova and S. Fdida, "An effective hop-by-hop interest shaping mechanism for ccn communications," in *Proceedings of Computer Communications Workshops (INFOCOM WKSHPS)*, Orlando,USA, Apr. 2012.

[26] G. Carofiglio, M. Gallo, and L. Muscariello, "Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks," *SIGCOMM Comput. Commun. Rev.*

[27] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, "Multi-path tcp: A joint congestion control and routing scheme to exploit path diversity in the internet," *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 1260–1271, Dec. 2006.