MIRCC: Multipath-aware ICN Rate-based Congestion Control

Milad Mahdian
Northeastern University
mmahdian@ece.neu.edu

Somaya Arianfar
Cisco Systems
sarianfa@cisco.com

Jim Gibson
Cisco Systems
gibson@cisco.com

Dave Oran
Cisco Systems
oran@cisco.com

ABSTRACT

Information-Centric Networking (ICN) designs have yet to settle on a definitive approach to congestion control. In TCP/IP, window-based schemes have been studied extensively and deployed on a grand scale, while rate-based alternatives like Rate Control Protocol (RCP) have been studied but not widely deployed or analyzed at scale in the wild. In that tradition, ICN congestion control work has to date concentrated on window-based approaches.

We posit that rate-based congestion control in fact has properties of interest, and ICN's well-known differences from IP, including receiver-driven flow control, stateful forwarding, symmetric paths, and lack of installed base justifies re-considering rate-based congestion control as a viable candidate for ICN.

This paper presents MIRCC, a rate-based, multipath-aware ICN congestion control approach inspired by but noticeably differing from RCP. We first explain our ICN-appropriate mechanisms for calculating and signaling flow rates. We then present MIRCC's algorithm for calculating per-link rates, and show its convergence advantages over RCP's algorithm. Finally, we explore several approaches for making our scheme multipath-aware, finding a novel solution to achieving fairness and network utilization. To characterize MIRCC's behavior, we use simulation results throughout.

Categories and Subject Descriptors

C.2 [**COMPUTER-COMMUNICATION NETWORKS**]: Architecture and Design; Network Protocols; Distributed Systems

Keywords

ICN; NDN; CCN; information-centric networks; congestion control; transport; multipath flows; flow fairness; RCP

1. INTRODUCTION

Information-Centric Networking [1] is a new network architecture that differs radically from IP. The question of how ICN networks achieve congestion control is one of many that must be resolved if ICN is to be successful.

ICN'16, September 26 - 28, 2016, Kyoto, Japan

ISBN 978-1-4503-4467-8/16/09...$15.00

DOI: http://dx.doi.org/10.1145/2984356.2984365

An initial question is whether ICN congestion control should be window-based or rate-based. Window-based congestion control, such as is used by TCP (e.g. [2]), allows a window of outstanding packets which grows and shrinks in response to the absence or presence of congestion indications. Rate-based congestion control, such as used by RCP [3], gives endpoints a sending rate that is increased or decreased based on the absence or presence of congestion. The relative characteristics of the two approaches are a standard topic in Networking textbooks.

The success of TCP/IP, the decades of deployment, and the extensive research into multiple generations of TCP congestion control schemes make a windowing approach a natural candidate for ICN congestion control (e.g. [4], [5], [6]) Nevertheless, this paper investigates a rate-based alternative for ICN congestion control. Several observations led us to explore this approach.

First, the differences between ICN and IP mean the best approaches may differ. In particular:

- ICN's Interest/Data exchange is receiver-driven. Thus, in an overload situation, congestion control must affect the behavior of the requester of the data, not the producer whose Data messages are the proximate cause of the overload.
- As a consequence of ICN's stateful forwarding, the Interest's upstream path and the corresponding Data message's downstream path are symmetric. This allows rate feedback state to aggregate well on the downstream path.
- ICN networks may use in-network caching, which affects where Data is pulled from, even on a message-by-message basis.
- Several interlocking ICN properties (the absence of address-based sessions, more robust delivery of Nacks, tolerance for out-of-order delivery, quick suppression of looping packets due to stateful forwarding) all support the possibility that ICN flows may be able to derive significant benefit from multipath.

Second, TCP congestion control is imperfect and still evolving in spite of considerable investment over decades, as shown by a random sample ([2], [7], [8], [9], [10]) among countless recent papers. Only a subset of this research is specific to window-based congestion control, but clearly this is not a solved problem.

Third, the overwhelming installed base of mature window-based TCP/IP flow control and the requirement to be friendly to these flows likely prevents the deployment of alternate schemes regardless of technical tradeoffs, which in turn prevents such alternate schemes from evolving and improving over time.

In summary, we believe, both because of the differences between ICN and IP and because RCP was a late entrant in IP congestion control, that rate-based congestion control for ICN is worth investigating. Given the multiple dimensions on which the two approaches differ and the complexity of algorithmic options, however, a short-term proof of superiority of either over the other is not feasible.

We identify a number of goals for an ICN congestion control scheme. It should provide some level of fairness between flows while allowing those flows to benefit from available network resources. (We target flow fairness, in spite of the arguments of [11].) Given a stable overall load, flows should receive a stable allocation of resources that maximizes network utilization while avoiding congestion and excess latency. In response to changes in load, the scheme should converge quickly to maintain those properties.

Allowing ICN flows to benefit fully from available network resources requires support for multipath. Coupling multipath with other objectives is of particular interest. When flows have multiple paths to their respective content, the tension between fairness and maximizing utilization is more complicated to resolve.

From a practical standpoint, we want a scheme that does not require forwarders to maintain long-lived per-flow state (in addition to the short-lived state in the Pending Interest Table).

Finally, we consider only distributed algorithms in the style of the current Internet, not centralized omniscient controllers.

In this paper, we propose *MIRCC* (Multipath-aware ICN Rate-based Congestion Control), a rate-based multipath-aware congestion control scheme for ICN networks. It is inspired by but significantly divergent from IP RCP. Two main contributions of MIRCC:

- better convergence time with less overshoot and oscillation than RCP.
- a multipath strategy allowing a flow to fully utilize the network resources along all the available paths to it, while maintaining fairness among competing flows regardless of number of paths that each flow has.

The main elements of *MIRCC* include an algorithm in each forwarder calculating Dual-Class rates for each link, protocol mechanisms to communicate rates and path identifiers to consumers in Data messages, an algorithm in each consumer to determine Interest sending rates for each class and to determine a sensible distribution of Interests across available paths, a mechanism for the consumer to hint at the desired path for an Interest, and forwarder support for interpreting the hint. Additionally, as a path discovery mechanism, the consumer can omit hints from Interests, in which case forwarders with multiple next hops will probabilistically distribute Interests across those next hops.

The remainder of this paper is structured as follows. After describing related work (Section 2), we first (Section 3) explain how the RCP-style end-to-end feedback loop operates in an ICN network, considering only single-path flows. The feedback loop is driven by $R(t)$, the per-link *stamping rate*: we explain MIRCC's per-link algorithm that iteratively updates $R(t)$ and show that, as flows start and finish, using the MIRCC algorithm (5) leads to better convergence time and less overshoot/oscillation than the classic RCP per-link algorithm (1) adapted to ICN.

We then generalize MIRCC to the multipath case (Section 4) to address this issue: How can we assure a flow a fair share of the network resources while maximizing network utilization? We describe MIRCC *Dual-Class Best-Subflow* scheme (Section 4.2), a novel multipath-aware rate-based congestion control scheme which allows flows to fully utilize network resources along their available paths while maintaining a fairness among competing flows, regardless of number of paths each flow has.

We have an ICN forwarder under development that is tracking the various drafts under development in the IRTF ICN Research Group. In addition to its normal target environments, we have integrated our ICN forwarder with ns-3 for simulation purposes. The basic elements of this integration are an ns-3 "model" that adapts ns-3 to our ICN forwarding code, an ns-3 consumer application that implements the various endpoint behaviors given in the paper, and a simple producer application that responds to Interests with Data messages of configurable size. Throughout the paper, we have used this simulation environment to evaluate the various alternatives.

Key Definitions.

The concept of a *flow* and related terms are used throughout the paper, with the following specific meanings.

A *flow* is a set of Interests generated by a single client, sharing a prefix, and handled by one or more producers.

multipath is the situation where packets for a single flow take (or have the potential to take) more than one path.

subflow describes a subset of the Interests and Data messages that belong to the same flow and follow the same path. (Subflow is only useful in the presence of multipath.)

2. RELATED WORK

ICN Congestion Control.

In [12], Lei et al. propose adopting RCP for ICN networks. They illustrate the potential benefits of RCP for the ICN architecture, including high network utilization and max-min fairness and superior results compared to NDN-AIMD, an ICN window-based congestion control scheme. The authors demonstrate that, due to the symmetric and hop-by-hop routing of the NDN architecture, RCP is an attractive candidate to tackle the congestion control problem of such networks. As compared to this research, our paper proposes an alternate rate calculation algorithm with improved convergence time and less overshoot, and proposes an approach for managing multipath with rate-based congestion control.

In [4], Carofiglio et al. formulate the problem of joint multipath congestion control and request forwarding in ICN as a global optimization problem. By decomposing the problem into two subproblems of maximizing user throughput and minimizing overall network cost, they develop a receiver-driven window-based congestion control and a hop-by-hop dynamic request forwarding algorithm, respectively.

In [13], Wang et al. propose a simple hop-by-hop Interest shaping algorithm for NDN to avoid network congestion and achieve optimal network resource utilization. The proposed solution accounts for the interdependence between Interests and Contents in opposite directions and shares link bandwidth optimally without extra message exchange.

In [14], Yeh et al., propose VIP, a systematic framework for joint dynamic Interest request forwarding and cache placement, within the context of the NDN architecture. In VIP, a virtual control plane operates on the user demand rate for data objects in the network, and an actual plane handles Interest Packets and Data Packets. The authors develop jointly optimal and distributed dynamic forwarding and caching algorithms within the virtual plane. Later, a congestion control mechanism is added to the framework, which acts to maximize user utilities subject to network stability.

TCP/IP Congestion Control.

Many improvements to TCP congestion control have been proposed and adopted over the years. We do not attempt a comprehensive summary of general TCP congestion control state of the art, but highlight some areas particularly relevant to our work.

Explicit Congestion Notification (ECN) allows end-to-end notification of network congestion without dropping packets [15]. If ECN is successfully negotiated, an ECN-aware router with active queue management (AQM), when experiencing congestion, may set a mark in the IP header instead of dropping a packet in order to signal impending congestion. The receiver of the packet echoes the congestion indication to the sender, which reduces its transmission rate as if it detected a dropped packet.

MP-TCP [16] describes a window-based congestion control algorithm operating at the endpoints of a multipath TCP flow that increases (or at least does not decrease) flow throughput compared to a single path and is fair to other traffic.

In [3], Dukkipati proposed the Rate Control Protocol (RCP), a congestion control scheme intended for IP. RCP's main objective is to minimize flow completion times by allowing flows to rapidly determine their correct rate. Each packet carries a rate field in its header. A route maintains a flow rate, $R(t)$, on each link. On each packet, the router "stamps" the header rate with $R(t)$ unless the packet already carries a lower value. The sender thus learns the bottleneck rate of a flow's path in one round trip. As a new rate is delivered with each packet, routers update $R(t)$ so that senders will update their rate. The equation for $R(t)$ attempts to find a rate that fills links while keeping queue sizes close to zero.

The RCP rate update equation used by the authors differs from our MIRCC rate update equation, and we compare the two in detail. The RCP equation is

$$R(t) = R(t-T)\left(1 + \frac{\frac{T}{d}\left(\alpha(\eta C - y(t)) - \beta\frac{q(t)}{d}\right)}{C}\right) \quad (1)$$

where η is the target link utilization, d is a moving average of the RTT measured across all packets, T is the update interval ($T <= d$), $R(t-T)$ is the latest updated rate, C is the link capacity, $y(t)$ is the measured aggregate input traffic rate during the last update interval, $q(t)$ is the instantaneous queue size, and α, β are parameters chosen for stability and performance.

Although RCP, as compared to TCP and XCP [17], improves flow completion times, promotes high network utilization and provides max-min fairness [18], it has not seen much deployment due to a number of factors: 1) In IP, the packets with rate feedback do not necessarily traverse the same path as the packets with the rate stamp (asymmetric routing). 2) RCP may exhibit oscillatory behavior and lead to significant overshoot or undershoot during abrupt changes in traffic load, as described in [19]. 3) RCP is relatively slow in converging to the equilibrium point compared to XCP and PERC [20]. 4) RCP is not TCP-friendly, imposing a challenging compatibility issue in IP. 5) RCP does not address multipath forwarding.

PERC [20] is a proactive congestion control scheme inspired by message-passing algorithms, proposed for datacenters with speeds higher than 100 Gb/s. PERC explicitly computes, in a decentralized fashion, rates independent of congestion signals such as the incoming traffic. The authors show that for high speed networks (100 Gb/s and above), PERC improves convergence times by a factor of 7 compared to reactive explicit rate control protocols such as RCP. However, for lower speed networks (10 Gb/s) PERC's convergence time is higher than RCP's and other reactive protocols'.

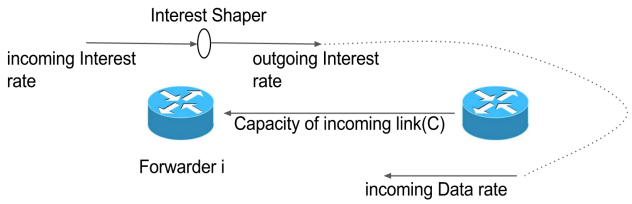


Figure 1: Modeling a forwarder under MIRCC

3. MIRCC FOR SINGLE-PATH FLOWS

This section discusses the application of MIRCC to a flow with a single path. Starting with the special case of single-path flows allows for simpler presentation of core elements of MIRCC before moving on to MIRCC's multipath Dual-Class Best-Subflow scheme, described in Section 4.

The heart of the system is the choice of control equation that the forwarders run, on each link, to calculate $R(t)$, the per-link *stamping rate*. This algorithm determines the overall distributed system's tradeoff between responsiveness and stability. A feedback loop operates to communicate rate information to consumers.

3.1 Single-Path Overview

MIRCC assumes an ICN architecture and system such as provided by CCCNx [21] or NDN [22]. These architectures include separate Interests (initiated by consumers) and Data messages (sent by producers in response to Interests). These two types of messages are treated quite differently. A consumer wishing to retrieve a flow of Data objects starts by sending Interests at some initial rate determined by the MIRCC consumer logic. These are routed through the ICN network, with temporary state being created in the *Pending Interest Table* (PIT) at each forwarder. Upon receiving an Interest, a producer generates a Data message in response, and sends it back toward the consumer. Each Data message carries a rate value, initialized to *MaxRate* by the MIRCC producer logic. The forwarder at each hop forwards the Data message along the reverse path of the corresponding Interest, with the link for the reverse hop determined by the PIT entry.

The MIRCC component in each forwarder maintains a rate, $R(t)$, for each link. As the Data message traverses the path back to the consumer, MIRCC at each forwarder compares the $R(t)$ assigned to the local link from which it received the Data message and the rate in the received Data message. If the local $R(t)$ is smaller than the carried rate value, the forwarder updates the outgoing Data message's value with the local $R(t)$. Hence, the bottleneck forwarder for the packet's path is the one which last updated the packet's rate value, and the bottleneck rate is the corresponding link rate.

When the Data packet arrives at the consumer, the rate in the Data packet informs the consumer of the rate for the flow's path. The consumer uses this value to update its sending rate. The forwarder calculation of $R(t)$ at each link is smoothed, so the consumer immediately applies the received $R(t)$ value as its new sending rate without any further smoothing.

The per-link $R(t)$ calculation includes a feedback mechanism: if the link is overloaded, the flow stamping rate is reduced and if the link is underloaded, the flow stamping rate is increased.

ICN (unlike TCP) is receiver-controlled via Interests. The consumer varies the rate at which it sends Interests as the mechanism to control end-to-end load of both Interests and the (generally much larger) Data messages. As will become clear, this mechanism and others rely on both consumers and forwarders maintaining an esti-

mate of the ratio of Data message size to Interest size.

In addition to MIRCC's per-link $R(t)$ calculator/stamper, each forwarder is assumed to have a per-link Interest Shaper, e.g. as described in [13] (see figure 1). This resource control mechanism monitors traffic per-link and paces Interests per-link. This shaper, given its per-link functions, is a natural place for a software implementation to calculate MIRCC's $R(t)$.

The shapers at either end of the link, by pacing Interests as necessary, control returning Data message load and avoid Data message overflow on the reverse path. Note that the operation of this Interest shaper mechanism also acts to control bandwidth of the outgoing Interests and leaves enough headroom on the reverse path for the other end of the link to send its Interests.

As a consequence of the Interest shaper mechanism, Data messages should rarely be dropped: the Interest shaper books the reverse bandwidth for the Data message via the shaping. During an overload, Interests back up in the shaper until some are eventually not forwarded. In general, these overload Interests are not silently dropped, but are Nacked back towards the consumer, using the bandwidth implicitly "booked" in anticipation of a returning Data message. Receipt of a Nack informs the consumer that an Interest was not delivered due to congestion. Assuming all the data is required, the consumer will know to re-transmit a new Interest with the same name, without needing to wait for a timeout.

Under MIRCC, Nacks also carry $R(t)$. As the rate information is only over a partial path, the consumer only updates its flow rate if the received $R(t)$ is *smaller* that the current rate. Unlike with windowing schemes, the consumer in our rate-based scheme does not take any additional steps on receipt of Nacks, as the network can set $R(t)$ as low as it chooses.

Parameter	Definition
$R(t)$	The stamping rate computed at time t
$R_p(t)$	The stamping rate for the Dual-Class primary traffic class
$R_s(t)$	The stamping rate for the Dual-Class secondary traffic class
C	Capacity of incoming link
$C_s(t)$	Secondary class capacity of incoming link
$\hat{N}$	Equivalent number of flows with full rate
T	Period of rate calculation iteration
$q(t)$	Inflated instantaneous queue size
$y(t)$	Inflated incoming Interest rate during $[t-T, t)$
$d(t)$	Smoothed average RTT
$\beta(t)$	Self-tuned parameter chosen for stability
η	Target link utilization

Table 1: List of Parameters.

3.1.1 Equation for calculating $R(t)$

MIRCC at each forwarder maintains $R(t)$ for each outgoing link to which it sends Interests, and offers this rate to each passing flow. The classic RCP calculation, proposed in [3], is shown here as equation (1). However, as explained earlier, this calculation overshoots significantly when new flows arrive, leading to packet drops. In addition, its convergence time is relatively slow. To address this issue, we instead developed the following algorithm. In this equation, each forwarder computes $R(t)$ for each of its outgoing links as the aggregate of $base\_rate(t)$, calculated by (non-recursive) equation (3) to split the allowed link bandwidth among the passing flows, and $excess\_rate(t)$, calculated by (recursive) equation (4) to fill the extra available bandwidth with traffic equally. As with RCP [3], MIRCC does not need to count the actual number of flows.[1] RCP uses an equivalent number of flows with full rate, given by $C/R(t-T)$. Our simulations show that while this estimate sometimes works well, it often reacts to new flow arrivals relatively late, resulting in queue build-up and ultimately oscillation and Data packet drops. Hence, we have chosen a different estimate given by

$$\hat{N} = max(C, y(t))/R(t-T) \tag{2}$$

which takes into account $y(t)$ and, as we show in simulation, does not suffer from those problems.

Each forwarder then computes $base\_rate(t)$ as follows:

$$base\_rate(t) = \frac{\eta C - \beta(t)\frac{q(t)}{d(t)}}{\hat{N}} \tag{3}$$

where η is the fraction of physical link bandwidth that $R(t)$ will attempt to fill, C is the capacity of the incoming link as shown in Figure 1, $q(t)$ is the instantaneous queue size at the Interest Shaper inflated to the expected size of the corresponding reverse Data message, $d(t)$ is the average RTT, and $\hat{N}$ is the estimated number of flows. Further, $\beta(t)$ is a self-tuned positive parameter given by

$$\beta(t) = max\left(\beta', \frac{y(t) - y(t-T)}{y(t)}\right)$$

where β' is a constant parameter set to 0.1 by default, $y(t)$ is based on the measured incoming Interest rate arriving during the time interval $[t-T, t)$ to be transmitted on the outgoing link, but is inflated to the expected size of the corresponding reverse Data messages.[2] Note that both $y(t)$ and $q(t)$, although based on *Interest* rate, are intended to estimate and manage the incoming rate of the much larger *Data messages* from the upstream neighbor and prevent buildup in the upstream neighbor's Data message queue, which is why they are scaled up by the Data/Interest inflation factor. As $y(t)$ and $q(t)$ include Nacked Interests, $y(t)$ and $q(t)$ are both overestimates of the incoming rate and queue buildup of the *Data messages* at the upstream neighbor.

To estimate the inflation factor, each forwarder maintains, on each link, exponentially weighted moving averages of the size of output Interests and input Data messages. The ratio of these is the inflation factor.

The next step in the scheme is to allow the flows not passing any saturated links to fill the extra available bandwidth with traffic equally. On each link, the incremental excess rate is given by

$$excess\_rate(t) = R(t-T) - y(t)/\hat{N} \tag{4}$$

To avoid high-frequency oscillation, an exponential weighted moving average (EWMA) is applied to both $base\_rate(t)$ in (3) and $excess\_rate(t)$ in (4). In our simulations, the EWMA α is 0.5.[3] Combining these elements gives us the stamping rate, i.e.,

[1] Although N is described as the number of flows, $\hat{N}$ is a derived value based on the observed change in link traffic that follows a change in $R(t)$. Flows that are not elastic or are bottlenecked on links do not respond to changes in $R(t)$. The total count of flows using the link in a given period is thus neither needed nor wanted. Hence we use the description "full-rate-equivalent flows".

[2] The "incoming Interests" making up $y(t)$ are incoming to the link's output Interest shaper, not Interests arriving on the link. We cannot accurately describe $y(t)$ as "outgoing Interests" because $y(t)$ includes Interests that are not transmitted in $[t-T, t)$, either because the shaper Nacks or queues them, and excludes Interests transmitted in $[t-T, t)$ that were queued in previous periods.

[3] i.e. $\mu(t) = (1-\alpha)\mu(t-T) + \alpha x(t)$

$$R(t) = base\_rate\_ewma(t) + excess\_rate\_ewma(t) \quad (5)$$

3.1.2 *Forwarder Determination of T and $d(t)$*

T is the period for an iteration of the $R(t)$ calculation. For instance, $R(t-T)$ is the flow stamping value used in the interval $[t-T, t)$ and $R(t)$ is the stamping value in the interval $[t, t+T)$. Traffic statistics are collected in the interval $(t-T, t)$ for use when calculating $R(t)$.

In general, T should be as short as possible, and no greater than RTT. The limitation on reducing T is that traffic measurements become lumpy and misleading as $T \Rightarrow 0$, tending towards either 0 or C (link rate).

$d(t)$ is the EWMA of the per-packet RTT.

For single-path flows, the consumer's main job is to send Interests at a rate consistent with the $R(t)$ received for the flow.

Recall, however, that $R(t)$ is the rate in bits of the Data messages that the consumer should be receiving, not the bit rate of Interests the consumer is sending. The consumer must therefore *deflate* the path's $R(t)$ by the Data/Interest inflation factor to determine the Interest rate. The consumer using $R(t)$ (like the forwarder calculating $R(t)$) must therefore have some estimate of Interest-to-Data inflation. For some applications (e.g. constant bit rate video flows), this is known to the consumer *a priori*. For other applications, the data sizes may be known by the producer ahead of time and communicated from producer to consumer using a *Manifest*.

3.2 Comparison to RCP

Comparing RCP's algorithm to MIRCC's algorithm.

As shown in our simulations and also reported in [19] and [20], RCP's algorithm has a relatively slow convergence time and a tendency to overshoot or undershoot during abrupt changes in traffic load. This is due to the way that rate is calculated in RCP, given in equation (1). The rate calculation in RCP is done in a recursive manner, thereby taking some time to converge to the equilibrium point. Therefore its reaction to abrupt changes in traffic load becomes slow, resulting in overshoot and undershoot. By using the rate calculation given in equations (3) - (5), MIRCC removes this recursive element. Further, equations (4) and (5) reduce the dependency of the algorithm outcome on sudden changes in traffic and act as a safeguard against overshooting and undershooting.

To compare the performance of MIRCC's and RCP's algorithms, we have adapted the IP RCP algorithm to ICN and compared the two algorithms in the simple ICN scenario depicted in Figure 2. In this linear network topology with 4 nodes, node 0 is the requesting node, and Node 3 is the producer. Each link has a capacity of 1Mbps. There are 4 consumer applications installed on Node 0. Consumer 2 is a long flow working as the background traffic. The other three consumers start sending packages at time 83 seconds and compete for bandwidth. As can be seen by comparing figures 3a and 3b, MIRCC converges to the new rate much faster than RCP. Also, the traffic rate in MIRCC shows less oscillation. It is worth noting that in both of these simulations we have used the Interest Shaper at each forwarder. The presence of the Interest Shaper has reduced the oscillatory behavior of the RCP scheme greatly.

Also note that MIRCC and RCP have similar network utilization and flow completion time properties. This is easy to see, as equations (3)-(5) reach the same equilibrium point as equation (1).

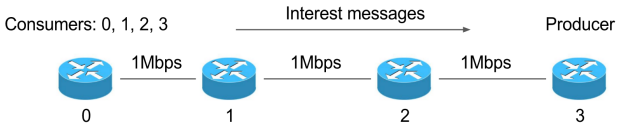
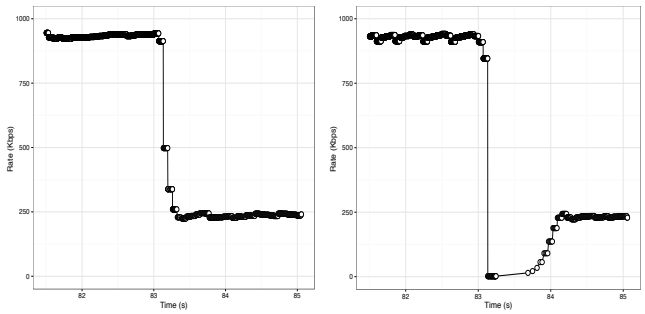


Figure 2: Linear topology with 4 nodes.



(a) Rate convergence in MIRCC. (b) Rate convergence in RCP.

Figure 3: Rate convergence comparison between (a) MIRCC and (b) RCP's update algorithm as adapted to ICN.

Comparing RCP-like schemes under IP and ICN.

Another RCP comparison of interest is between the control loops for IP RCP and for ICN MIRCC. In IP RCP, the rate is stamped in sent traffic, received by the destination, and echoed back to the sender in the resultant ACK. In the event of a link becoming overloaded, the feedback path is at least half a round trip (if the overloaded link is very close to the receiver). Regardless of which link is overloaded, the sender will originate an RTT's worth of traffic between the forwarder detecting the link overload and the sender's receipt of a rate that has been updated based on the overload.

In an ICN network, due to symmetric paths of Interest and Data, the updated rate can be stamped directly in returning Data messages rather than needing to be stamped in Interests and echoed back by the receiver. The feedback time to a given sender is thus the wait for the next Data message plus at most half a round trip (if the overloaded link is very close to the receiver). The sender thus originates at most a half round trip's worth of traffic (and very little if the overloaded link is close to the sender) between detection of the overload and the sender's receipt of a rate reflecting the overload.

4. MIRCC FOR MULTI-PATH FLOWS

An RCP-style rate-based congestion control algorithm such as MIRCC is relatively straightforward for single-path flows. The calculations at each link are designed to be simple. The sending endpoint simply reacts to the returned rate. The producer endpoint in ICN has no responsibilities.

The existence of multiple paths with different rates introduces several new challenges.

- Given multiple paths for a flow, each with their own rate, at what rate should the consumer send the Interests for a flow? What per-flow fairness does this imply? What portion of the Interests should follow each path?
- By what mechanisms are the multiple paths and their rates discovered? By what mechanisms are Interests directed onto the appropriate paths?

For this paper, the producer at the upstream end of each path is assumed capable of supplying the entire flow. The effects of opportunistic caching of some subset of the Data messages in a flow are not considered at this time.

4.1 Multipath Fairness vs. Utilization

Two multipath flow rate concepts are used throughout the paper:

- The *best-subflow rate* is the largest rate of any of the flow's subflows. Having all flows sending at their best subflow rate achieves max-min fairness between flows, by an extension of the argument that single-flow RCP achieves max-min fairness between flows. Network utilization is not maximized by this approach.

- The *total per-subflow rate* is the sum of the rates of all the flow's subflows. Sending at this rate (distributing Interests onto each subflow at its rate) maximizes network utilization, by an extension of the argument that single-flow RCP maximizes network utilization. (Each subflow is simply treated as a separate single-path flow.) Max-min fairness between flows is not achieved.

Some examples motivate why sending at the best-subflow rate does not maximize network utilization, and why sending at the total per-subflow rate does not achieve max-min fairness.

In figure 4, consumer A has two 4-hop paths to producer A, where the two paths share l_a, l_g. Suppose further that many other consumers, not shown, are using these links such that $R(t)$ is only minimally affected by the presence of an additional consumer.

In both scenarios shown, the bottleneck links for the two subflows are l_c, l_d and the rates returned to the consumers in Data messages traversing the two subflows are thus 10Mbps. A key difference between the scenarios is that, in scenario 2, the minimum cut is less than the sum of the subflows, i.e. $l_g < l_c + l_d$.

Consumer A might request at the best-subflow rate, i.e. 10Mbps. In this case however, the consumer is not taking advantage of multipath and network utilization is not maximized.

Alternatively, consumer A might request data at the total per-subflow rate, i.e. $\sum_{subflow} R = 20Mbps$, to take advantage of multipath. In scenario 2, sending at 20Mbps means taking more than the 15Mbps single flow rate on R_g. Not only might that seem unfair (e.g. to consumer E, which will send at 15Mbps on its single path), but consumer A has no way of distinguishing the two scenarios to treat them differently, since the returned per-subflow stamping rates are identical.

Alternatively, in scenario 2, an omniscient consumer A, if it had a way to understand the entire topology, might send at 15Mbps, taking a fair share of the bottleneck link l_g. This would be fair to consumer E and maximize utilization of the bottleneck link. But MIRCC is a distributed mechanism rather than an omniscient SDN-style mechanism.

In figure 5, assume that only the consumers shown are present, so an additional consumer has a large effect on $R(t)$. If only Consumer 2 is running, its single path is bottlenecked on l_c and its rate is 9Mbps. If only Consumer 1 is running, it has 2 paths but l_a is the bottleneck link, with the 15Mbps of traffic split between l_b and l_c. (Multiple distribution between those two links are possible.)

With Consumers 1 and 2 both running, l_a has 2 flows/3 subflows, l_c has 2 flows/2 subflows and l_b has 1 flow/1 subflow. With only two flows, the decision of how to distribute Consumer 1's traffic between l_b and l_c has a big effect on the stamping rates on those links, the link responsible for the lowest path stamping rate can change between l_a and l_b/l_c. Absent the MIRCC mechanisms,

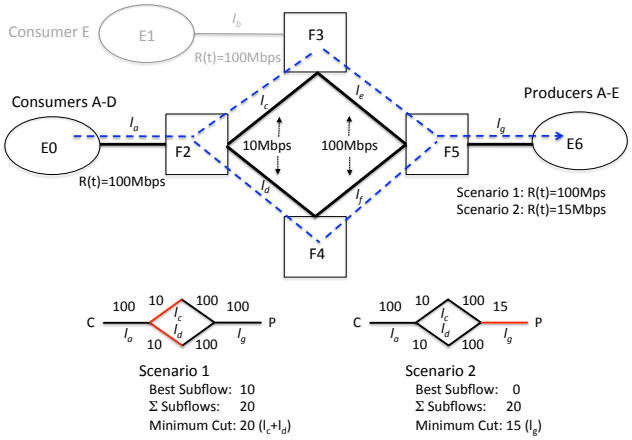


Figure 4: Diamond Topology

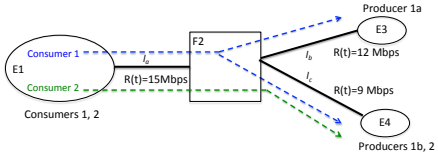


Figure 5: Slingshot Topology

oscillations in bottleneck link can lead to instability in total traffic, per-link traffic, and per-flow traffic.

4.2 Dual-Class Best-Subflow Multipath Rate Management

MIRCC's Dual-Class Best-Subflow scheme has the following main elements.

The Dual-Class scheme includes primary and secondary traffic classes. Each Interest is marked with a class. For each link, the owning ICN forwarder maintains per-class rates: $R_p(t)$ and $R_s(t)$, a generalization of $R(t)$ as described in section 3. Primary and secondary rates are both stamped in each Data message, which thus carries its path's lowest value for each rate.

The forwarder's maintenance of $R_p(t)$ for each link is very similar to the maintenance of single-path $R(t)$ in section 3. (In the absence of secondary class traffic, the two are identical.) However $y_p(t)$ reflects only primary class Interests, with consequent effect on dependent parameters such as $\hat{N}$. ($q(t)$ includes all the queued Interests, regardless of class, for both primary and secondary rate calculation.)

$$\hat{N}_p = max(C, y_p(t))/R_p(t-T) \tag{6}$$

$$base\_rate_p(t) = \frac{\eta C - \beta_p(t)\frac{q(t)}{d(t)}}{\hat{N}_p} \tag{7}$$

$$\beta_p(t) = max\left(\beta', \frac{y_p(t) - y_p(t-T)}{y_p(t)}\right) \tag{8}$$

$$excess\_rate_p(t) = R_p(t-T) - y_p(t)/\hat{N}_p \tag{9}$$

$$R_p(t) = base\_rate\_ewma_p(t) + excess\_rate\_ewma_p(t) \tag{10}$$

A MIRCC consumer sends primary class Interests at the flow's best-subflow rate. Thus, primary class traffic is max-min fair between flows, but does not maximize network utilization.

The purpose of the secondary class is to "soak up" bandwidth not used by primary class traffic. This is achieved by three key differences between the primary and secondary classes.

First, the capacity term in the algorithm for $R_s(t)$ is $C_s(t)$ rather than C. The link's secondary capacity consists of what is not used by primary traffic.

$$C_s(t) = max(C - y_p(t), 0) \tag{11}$$

Also, $y(t)$ is replaced by $y_s(t)$, which reflects only secondary class traffic.

Second, the per-link shaper under load, drops queued secondary class Interests to make room for new primary class Interests.

Third, a MIRCC consumer sends secondary class Interests at the flow's total-subflow secondary rate. Thus, secondary class traffic maximizes network utilization without having any particular per-flow fairness property.[4]

The combined fairness of the primary and secondary class traffic is therefore a mixture, but with a core of per-flow max-min fair traffic sent in the (preferred) primary class.

$$\hat{N}_s = max\{max(C_s(t), y_s(t))/R(t-T), 1\} \tag{12}$$

$$base\_rate_s(t) = \frac{\eta C_s(t) - \beta_s(t)\frac{q(t)}{d(t)}}{\hat{N}_s} \tag{13}$$

$$\beta_s(t) = max\left(\beta', \frac{y_s(t) - y_s(t-T)}{y_s(t)}\right) \tag{14}$$

$$excess\_rate_s(t) = R_s(t-T) - y_s(t)/\hat{N}_s \tag{15}$$

$$R_s(t) = base\_rate\_ewma_s(t) + excess\_rate\_ewma_s(t) \tag{16}$$

4.3 Multipath Path Management

MIRCC management of multiple paths includes a significant role for the consumer endpoint. Recall that MIRCC is a distributed algorithm that does not rely on an omniscient SDN-style controller. Even with long-lived per-flow state (which MIRCC specifically tries to avoid), no node other than the endpoint has the full view of the available paths that is needed to achieve fairness between flows. Thus, the end-user is assigned this enhanced role in MIRCC. The end-user also has the job of distributing Interests onto subflow paths.

For secondary rate Interests, the consumer endpoint sending at the total per-subflow secondary rate simply distributes the Interests onto each subflow based on its secondary rate.

For the primary class traffic, although sent at the rate of the best subflow, MIRCC distributes the traffic across the available subflow paths rather than sending all the traffic on the best-subflow path. This balancing achieves smoother evolution of traffic on each path and link by avoiding situations where small changes in a subflow's bandwidth can cause all the primary class traffic to switch onto and off of that subflow.

We choose a distribution proportional to the primary "stamping" rate on each path, subject to the constraint that no path and no link carry more than their stamping rate.

[4]Note that secondary class traffic has *per-subflow* fairness.

Having specified the desired behavior and identified that the consumer endpoint's transmission strategy achieves those behaviors, the next question is what mechanisms MIRCC uses to produce those results. In particular,

- How is it determined that multiple paths exist?
- How is traffic distributed onto paths?
- Each subflow must be identifiable by the consumer, so that the consumer knows the rates of each path and the changes in the best path and best path rate over time.
- To avoid overloading any lower-rate path, the consumer must be able to steer Interests onto paths.
- To discover paths, initially and over time, the consumer must send some subset of Interests without a steering hint, and the network must do some probabilistic multipath forwarding that leads to discovery of useful paths. (Recall that only a single initial message for any given path is needed to discover the path's rate.) For Interests without steering hints, the forwarders must apply some probabilistic multipath forwarding strategy to allow paths to be discovered.

4.3.1 Path Identification

By definition each subflow has a separate path, so identifying subflows is equivalent to distinguishing their paths. The *pathId* is signaled in a Data message header TLV and is calculated hop-by-hop, by hashing node+link identifier at each hop with the partial pathId received in the incoming Data message header. The Data message is sent downstream with an updated TLV.

4.3.2 Directing Traffic onto a Specific Subflow Path

The consumer, when distributing Interests to subflows, includes subflow-specific hints in the Interest that the network uses to steer the Interest along the subflow's path. The following mechanisms have been considered. The network of course can act on these hints as it chooses.

Bloom Filter Steering Hints: One approach is to use "Bloom filter steering" as in [23]. The Data message carries a header TLV in which is constructed a Bloom filter of the node+link identifier chosen at each split point. The consumer includes this Bloom filter in future Interests for the corresponding subflow. At split points, the forwarder looks up, in the received Bloom Filter, the node+link for each *nexthop* in the FIB entry. A hit indicates which nexthop should be used.

(For our simulations, steering is currently done with a modification of this scheme that was easier to implement, but is not appropriate for a real network. Each forwarder tracks, in its FIB, the upstream (received) partial *pathhash*, upstream face, and downstream (sent) partial pathhash. For Interests with a hint, the hash is looked up in the FIB entry's list of downstream partial hashes and the Interest is sent to the corresponding upstream interface, with the corresponding upstream hash. This simulation shortcut preserves the key properties of the Bloom Filter approach, though never misses.)

Random Value Steering Hints: An alternate approach is for the consumer to include a "random value steering" header TLV in Interests. The forwarder at each split point hashes this to select the next hop. The Data message, in addition to carrying a separate pathId, reflects back the steering hint received in the message. The consumer can probe for different paths by using new random numbers, and can detect path changes if old random numbers return new paths.

4.3.3 *Path Discovery*

Consumers sending Interests for a new flow do not yet have a list of paths for that flow. Therefore, initial Interests must be sent without a steering hint. A forwarder receiving a hintless Interest that matches a FIB entry with multiple next hops makes a probabilistic decision among the next hops. The MIRCC forwarder weights the next hops by the bandwidth of the outgoing links, though other schemes are possible.

To allow for the possibility of newly available paths, MIRCC consumers send a subset of Interests without steering hints even after discovering an initial set of paths. Currently, consumers send 3% of Interests without a steering hint.

4.3.4 *Sample Cases*

Three scenarios give a flavor of the operation of the Dual-Class Best-Subflow scheme.

First, consider a scenario with many elastic flows. The secondary rates on the various paths will tend to be 0. Consumers will thus send at the best-subflow rate with a per-flow max-min fair outcome.

A second scenario is where the links are used by a single flow. Figure 4, scenario 2 is a useful example topology. The best-subflow rate will be 10Mbps and traffic will be split equally with 5Mbps of primary class traffic on each path. The secondary rate on each path will initially be approximately 5Mbps for a total of 10Mbps. The flow converges to sending 5Mbps of secondary rate traffic, 2.5Mbps on each path, filling l_g, the flow's bottleneck link.

Third, with a small number of flows sharing the links, each flow will get its best-subflow rate of primary traffic, and the secondary rate capacity will be shared between the flows. Flows with more paths will get more of the secondary rate traffic.

4.4 Multipath Simulation Results

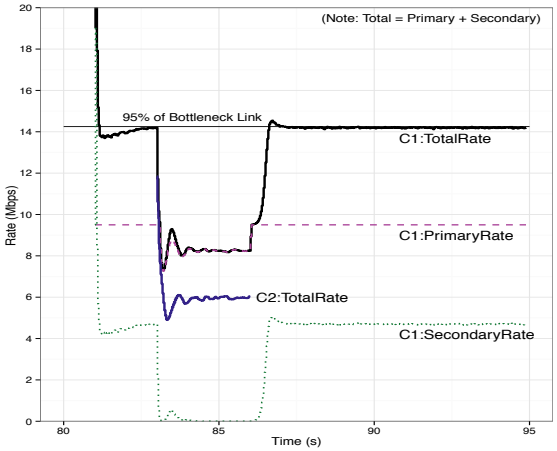


Figure 6: Dual-Class Best-Subflow Rate Management: 2 Consumers, Slingshot Topology, RTT=~25ms

Using the simulator, we ran many simulations of these multipath schemes and present representative results. In summary, MIRCC uses two rate classes to achieve fairness and good network utilization. Additionally, MIRCC converges to sensible behavior even with a small number of flows.

For the first scenario, we use the slingshot topology of figure 5, with a long-lived Consumer talking to Producer 1 and a shorter-lived Consumer talking to Producer 2. In figure 6, we show the rates for both Consumers over time for MIRCC's dual-class best-subflow schemes.

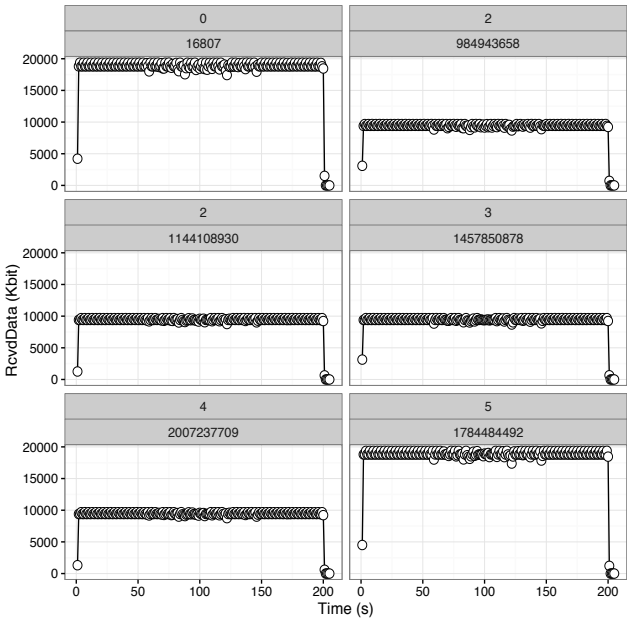


Figure 7: Diamond Topology - Scenario 1, Link Utilization vs. Simulation Time(s): each plot shows the link utilization of incoming links for each node. (Above each graph, the two boxes contain a Node ID from Figure 4 and a random link ID.)

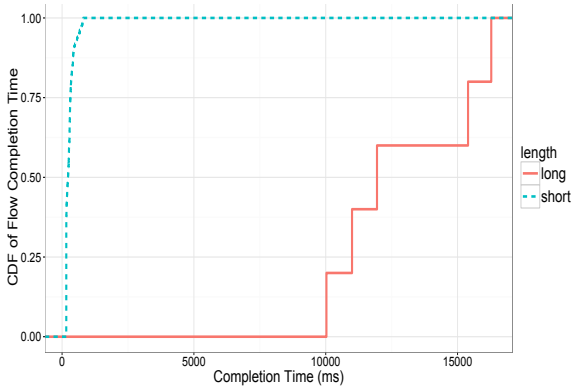
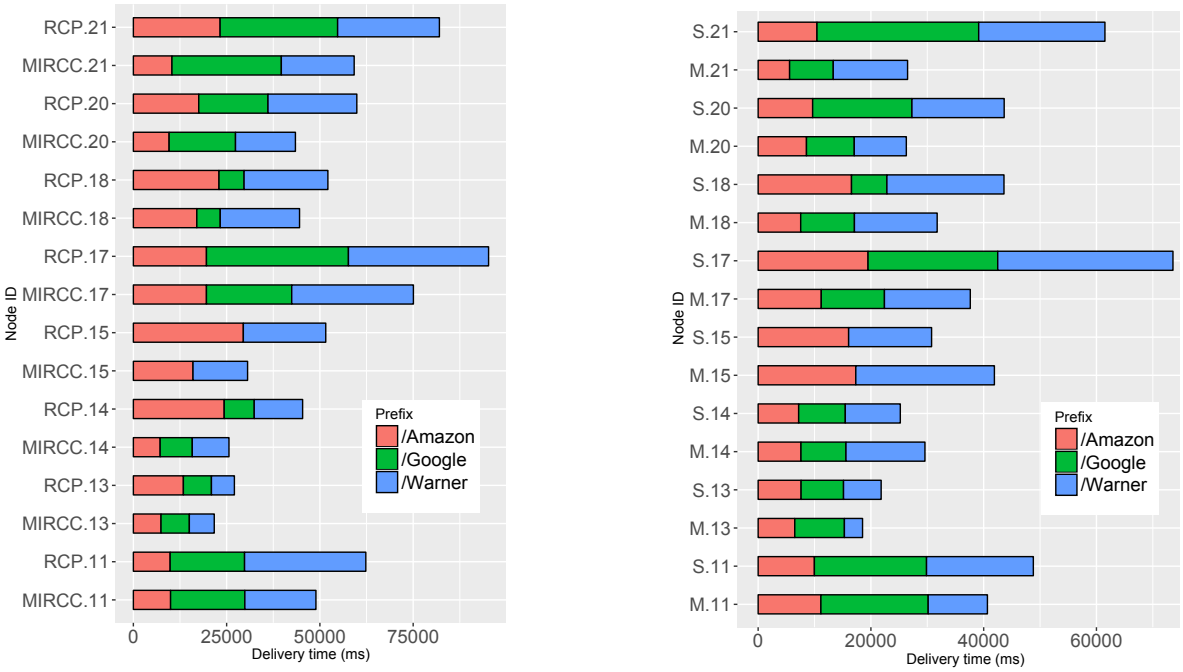


Figure 8: Diamond Topology- Scenario 1, CDF of Flows' Completion Times vs. completion time (ms).

Initially, Consumer 1's multipath flow is the only flow in the topology. Consumer 1 gets the full bandwidth of l_a (15Mbps derated by η). As shown by the dashed graphs, the primary class component is 12Mbps (corresponding to the best subflow through l_b) and the secondary rate component is 3Mbps (l_a's remaining available bandwidth).

Once Consumer 2 starts up, MIRCC's rate calculations and consumer responses evolve to a new convergence point that fills l_a, with no excess bandwidth available. Consumer 1 bottlenecks on l_a and Consumer 2 bottlenecks on l_c, the shared 9Mbps second hop. It happens that the Consumers' use of l_a converge to a ratio of just under 2:1. (If Producer 2 is moved to E3 so that the 12Mbps l_b is the shared second hop, the Consumers both bottleneck on l_a, which is shared evenly at 1:1.) When Consumer 2 finishes, the situation reverts to the starting use.

For the second scenario, we use the Diamond topology - Sce-



(a) MIRCC vs. RCP.

(b) MIRCC single path vs. multipath.

Figure 9: Abilene Topology: average delivery time(ms) at each node; (a) MIRCC vs. RCP: Single path only. (b) MIRCC effect of enabling multipath; S.i denotes node i for singlepath. M.i denotes node i when multipath is enabled.

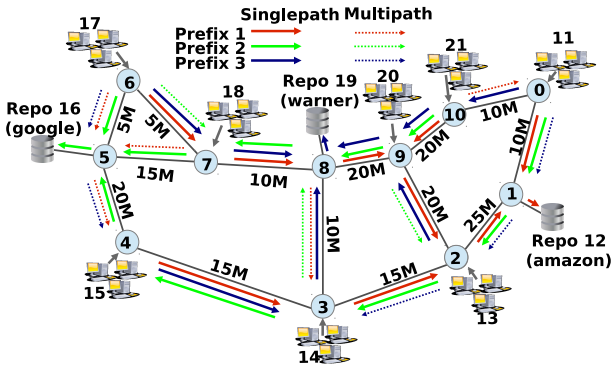


Figure 10: Abilene topology [4].

nario 1 as depicted in Figure 4 except that only consumers A-D are present. Flow arrival time follows a Poisson process with mean 0.5 seconds. We have two types of consumers chosen randomly upon arrival to be either a short flow (1000 messages) with probability 0.99, or a long one (50,000 messages) with probability 0.01. Long flows are considered to be the background traffic as the short flows come and go. Figures 7 and 8 are plotted to demonstrate the network link utilization and the Cumulative Distribution Function (CDF) of the flows' completion time, respectively. Figure 7 shows high and stable link utilization even though messages are arriving and leaving quite often. (On average, a new flow joins in every 0.5 seconds and leaves after receiving all of its messages.) Further, simulations do not exhibit any dropped Interest or Data messages.

In Figure 8, the CDF of the flows' completion time has been plotted, showing that MIRCC, unlike TCP-like schemes, does not discriminate against short flows.

In the third scenario, we consider the performance of MIRCC in the Abilene topology in figure 10 as used in [4]. This scenario features a more realistic set of nodes, dynamic consumer arrival time and varying flow size. Three repositories, at nodes 12, 16, 19, each store content under a given name prefix (/Amazon, /Google, /Warner, respectively). Each content is chunked in 5000 Data messages of size 1.8 kbytes each. The size of each Interest message is 300 bytes. Request arrival time follows a Poisson distribution with mean 3 seconds. The consumer is randomly assigned to one of nodes 11, 13, 14, 15, 17, 18, 20, 21 with a prefix among the aforementioned prefixes. We are mainly interested in 1) comparing MIRCC to RCP (with multipath forwarding disabled for fair comparison) and 2) demonstrating the performance of MIRCC when multipath mechanism is enabled. As shown in figure 10, we can modify the topology to enable or disable multipath forwarding. In Figure 9a, we show that MIRCC beats RCP in terms of average delivery time at every node. This further verifies that our rate calculation scheme does a better job than RCP's algorithm of converging to the adjusted rate. In figure 9b, M.i denotes node i when multipath is enabled. S.i denotes node i when multipath is disabled. As can be seen, the multipath solution significantly reduces delivery times, except at nodes 14 and 15, which experience a slightly higher latency due to an increase in the number of flows on links (3, 4), (4, 5) sharing available bandwidth fairly. Still, the overall advantage of using multipath is significant.

5. CONCLUSIONS

In this paper, we proposed MIRCC, a rate-based multipath-aware congestion control scheme for ICN networks, which is inspired by but significantly divergent from RCP. We first considered single-path flows and developed a non-recursive rate-calculation algorithm

which achieves max-min fairness, high link utilization and short flow completion time. The proposed algorithm, as shown through extensive simulation, has much better convergence time and less overshoot and oscillation than classic RCP. We then focused on multipath routing/forwarding and tackled the problem of coupling multipath routing/forwarding with objectives such as fairness, congestion control and network utilization. We showed that assuring full network utilization while maintaining fairness among competing flows regardless of number of paths each one has is a challenging problem. We designed a hybrid scheme with dual-class rate management, in which each flow has two rate levels: the primary rate, which is managed to ensure a level of max-min fairness between all flows, and the secondary rate, which is managed to consume remaining bandwidth to achieve full link utilization.

Possible further topics, among many, include the handling of UDP-tunneled links between devices, studying the effect of wider RTT spreads among competing consumers, and accommodating the effects of opportunistic caching by forwarders, particularly, following the *resource pooling principle* [24].

Acknowledgements

Many thanks to our shepherd, Ioannis Psaras, and to the anonymous reviewers, for comments that greatly improved the content and presentation of this material.

6. REFERENCES

[1] 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 '09*. New York, NY, USA: ACM, 2009, pp. 1–12.

[2] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008. [Online]. Available: http://doi.acm.org/10.1145/1400097.1400105

[3] N. Dukkipati, "Rate control protocol (rcp): Congestion control to make flows complete quickly," Ph.D. dissertation, Stanford University, Stanford, CA, USA, 2008, aAI3292347.

[4] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, and S. Wang, "Optimal multipath congestion control and request forwarding in information-centric networks," in *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, Oct 2013, pp. 1–10.

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

[6] L. Saino, C. Cocora, and G. Pavlou, "Cctcp: A scalable receiver-driven congestion control protocol for content centric networking," in *2013 IEEE International Conference on Communications (ICC)*, June 2013, pp. 3775–3780.

[7] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *Proceedings of the ACM SIGCOMM 2010 Conference*, New York, NY, USA, 2010, pp. 63–74. [Online]. Available: http://doi.acm.org/10.1145/1851182.1851192

[8] R. Scheffenegger, B. J. Briscoe, and M. KÃijhlewind, "Problem Statement and Requirements for Increased Accuracy in Explicit Congestion Notification (ECN) Feedback," RFC 7560, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7560.txt

[9] M. KÃijhlewind and R. Scheffenegger, "TCP Modifications for Congestion Exposure (ConEx)," RFC 7786, May 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7786.txt

[10] R. Gonzalez, J. Pradilla, M. Esteve, and C. E. Palau, "Hybrid delay-based congestion control for multipath tcp," in *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, April 2016, pp. 1–6.

[11] B. Briscoe, "Flow rate fairness: Dismantling a religion," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 63–74, Mar. 2007.

[12] K. Lei, C. Hou, L. Li, and K. Xu, "A rcp-based congestion control protocol in named data networking," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2015 International Conference on*, Sept 2015, pp. 538–541.

[13] Y. Wang, N. Rozhnova, A. Narayanan, D. Oran, and I. Rhee, "An improved hop-by-hop interest shaper for congestion control in named data networking," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, New York, NY, USA, 2013, pp. 55–60.

[14] E. M. Yeh, T. Ho, Y. Cui, R. Liu, M. Burd, and D. Leong, "Forwarding, caching and congestion control in named data networks," *CoRR*, vol. abs/1310.5569, 2013.

[15] S. Floyd, "Tcp and explicit congestion notification," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 5, pp. 8–23, 1994.

[16] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath tcp," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, Berkeley, CA, USA, 2011, pp. 99–112.

[17] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 89–102, 2002.

[18] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 59–62, 2006.

[19] Y. Zhang, S. Jain, and D. Loguinov, "Towards experimental evaluation of explicit congestion control," *Computer Networks*, vol. 53, no. 7, pp. 1027–1039, 2009.

[20] L. Jose, L. Yan, M. Alizadeh, G. Varghese, N. McKeown, and S. Katti, "High speed networks need proactive congestion control," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*. ACM, 2015, p. 14.

[21] The ccnx project. [Online]. Available: http://www.ccnx.org

[22] Named data networking. [Online]. Available: https://named-data.net

[23] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander, "Lipsin: Line speed publish/subscribe inter-networking," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, New York, NY, USA, 2009, pp. 195–206.

[24] I. Psaras, L. Saino, and G. Pavlou, "Revisiting resource pooling: The case for in-network resource sharing," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. ACM, 2014, p. 24.

.