Modeling multipath forwarding strategies in Information Centric Networks

Andrea Detti, Claudio Pisa, Nicola Blefari Melazzi
CNIT, University of Rome Tor Vergata, Italy.
andrea.detti@uniroma2.it, claudio.pisa@uniroma2.it, blefari@uniroma2.it

***Abstract*—A content can be replicated in more than one node in Information Centric Networks (ICNs). Thus, more than one path can be followed to reach the same content, and it is necessary to decide the interface(s) to be selected in every network node to forward content requests towards such multiple content containers. A multipath forwarding strategy defines how to perform this choice. We propose a general analytical model to evaluate the effect of multipath forwarding strategies on the performance of an ICN content delivery, whose congestion control follows a receiver driven, path-unaware, loss-based AIMD scheme. We use the model to understand the behavior of ICN multipath forwarding strategies proposed in the literature so far, and to devise and evaluate a novel strategy.**

***Index Terms*—Information Centric Networks, multipath forwarding, AIMD congestion control, analytical model, test-bed**

I. INTRODUCTION

The Internet today is more and more used as a container of information in which users can put content or from which they can get content. Correspondingly, networks must adopt efficient solutions to distribute contents, rather than to create host-to-host bit pipes. Efficient content distribution and dissemination systems exploit network strategies that try to jointly optimize communication, storage and computation resources. Content replication, caching, content routing, content adaptation are typical functionality of e.g. P2P applications, Content Delivery Networks and Information Centric Networks (ICNs).

ICN is an emerging network paradigm that puts the information delivery at the center of the network layer design. Whereas the current Internet model aims to create network pipes between hosts identified by addresses, ICN delivers to the users information (or contents) identified by names. A user expresses an interest for a content and the ICN functionality takes care of routing the content request towards the best source (be it the original one, a replica server, or an in-network cache) and of sending back to the user the requested data.

Content Centric Network (CCN) [9] is probably the best-known among the proposed ICN architectures [12]. A CCN includes routing-by-name, multicast delivery, receiver-driven congestion control and in-network caching functionality [1].

In-network caching and/or possible content replication results in the same content being available in multiple locations of an ICN. Thus, multipath solutions are very useful to speed up delivery and improve resilience [5]. A full multipath solution, either based on TCP/IP or ICN, usually includes: i) *path discovery*, ii) *congestion control*, and iii) *multipath forwarding* mechanisms. The path discovery makes involved nodes aware of the existence of multiple paths towards a given content. The congestion control regulates the data flow on the selected multiple paths. The multipath forwarding schedules traffic among available paths according to a given strategy; it can operate either on a per-packet basis or on a per-flow basis.

In this paper we propose an analytical model to evaluate the effect of per-packet multipath forwarding strategies on the performance of an ICN content delivery. We assume that the congestion control is regulated by a receiver-driven, loss-based Additive Increase Multiplicative Decrease (AIMD) scheme, which is not explicitly made aware of the underlying ICN multipath, and thus does not require modifications to current (e.g. NDN [3]) solutions.

Specifically, we consider the CCN architecture and use our model, validated by means of simulations, to compare the performance of literature multipath forwarding strategies [11] [6], understand why a strategy is better than another one, and devise a new strategy, named Fast Pipeline Filling.

II. BACKGROUND AND RELATED WORKS

Content Centric Network - CCN

A CCN addresses contents by using unique hierarchical names [9] (e.g. foo.com/doc1). Big contents are split into chunks, uniquely addressed by names that include the content name and the chunk number (e.g. foo.com/doc1/$CNx). To fetch a chunk, a receiver sends out an Interest message, which includes the chunk name. CCN nodes use a name-based Forwarding Information Base (FIB) to route-by-name Interest messages by using a prefix match logic. A FIB entry contains a name prefix (e.g. foo.com) and a list of *upstream* (inter)faces on which the Interest message can be forwarded towards available sources. When the upstream list contains more than one face, a multipath forwarding strategy singles out a forwarding face, or a set of them e.g. if replication is needed. During the Interest forwarding process, a CCN node leaves reverse path information <chunk name, *downstream* faces>in a Pending Interest Table (PIT). When an Interest

This work was performed in the context of the FP7/NICT EU-JAPAN GreenICN and the FP7 EU CONFINE projects.

---

[1] Another important ICN architecture is Named Data Networking (NDN) [3], which used CCNx (a Linux-based implementation of CCN) as its codebase, but as of 2013 it has forked a version whose differences however are not of importance in this paper.

reaches a node having the requested chunk, the node sends back the chunk within a Data message, which is routed on the downstream path by consuming the information previously left in the PITs. Traversed CCN nodes cache forwarded Data messages, so providing in-network caching functionality.

To download a whole content, a receiver fetches all the related chunks by sending out a sequence of Interest messages. For flow control purposes, the receiver exploits a receiver-driven approach, which consists in limiting the number of in-flight Interests through a congestion window (cwnd). The cwnd size may be constant or regulated by an Additive Increase Multiplicative Decrease (AIMD) control mechanism.

CCNx [1] is a Linux-based CCN implementation whose faces are UDP or TCP tunnels. The default multipath strategy implemented in the ccnd daemon (at least up to version 0.8.1) selects the fastest responding face, and performs experiments to determine if other faces can provide faster response. A similar approach is proposed in [13], but with face ranking based on data loss. In these cases only one path is used to fetch a given content; conversely, in this paper we focus on strategies that concurrently use all available paths. The default control mechanism, provided by the ccngetfile application, uses a constant congestion window.

ICN Multipath and Congestion Control

ICN multipath forwarding is carried out through specific network-layer strategies that usually operate on a per-packet (i.e. per-Interest) basis. In the CCN architecture, it is easily possible to extend the functionality of the PIT by adding to its main job of reverse routing also the monitoring of path performance parameters, such as number of pending Interests or round trip times. For instance, in [11] the authors propose a weighted round robin scheme among faces, whose weights are inversely proportional to the face round trip time; in [6] the weights are inversely proportional to the number of pending Interest messages; this strategy is claimed to be optimal to maximize user throughput and minimize overall network cost, in case of delay-based congestion control schemes (e.g. TCP Vegas).

Congestion control in ICN is an open issue and there is not, yet, a "standard" protocol. Out of order delivery may frequently happen in ICN, due to in-network caching and multipath. Thus, ICN receiver-driven congestion control schemes should not consider out of order delivery as a symptom of congestion, but rather infer congestion from other parameters such as increasing delay (delay-based congestion control) [6][5], and packet loss (loss-based congestion control) [4] [10]. Moreover, in presence of multipath, the congestion status of the different paths can be or not exposed by the network-layer to the congestion control algorithm. Clearly, having a per-path congestion information makes it possible to design more efficient congestion controls. For instance, in [5] authors add a route label to the Data packets that is a fingerprint of the traversed nodes, so that the receiver can estimate the RTT of the different paths and apply an RTT based congestion control. Without a per-path congestion information the receiver is only aware of a global congestion status, thus its reaction to the congestion is more coarse, but still effective in reducing congestion. For instance, in case of a loss-based AIMD without per-path congestion information, the loss on a single path halves the congestion window, thus reducing the traffic on all the paths, rather than only on the congested one. In this paper we assume a congestion control operating without per-path congestion information, leaving the other case for future work. However, we observe that both past (CCNx 0.1.0, 2009) and current (CCN 1.0 or NDN 0.3.1 2014 [3]) implementations do not provide any per-path congestion information, thus the case considered in this paper is compatible with current CCN/NDN specifications.

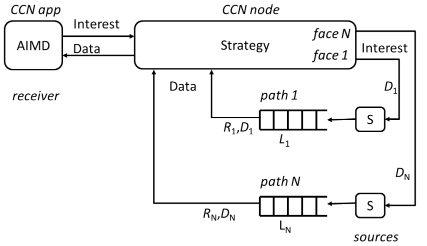

Fig. 1. Reference network model

III. MODEL OF THE AIMD RECEIVE-RATE WITH MULTIPATH FORWARDING

We propose an analytical model of the receive-rate of a loss-based AIMD congestion control in presence of a *generic* per-packet multipath forwarding strategy. The model exploits some simplifying approximations that, however, do not impair its accuracy, as we will show in section V.

Fig. 1 depicts the network model that we use for our analysis: a CCN application (e.g. ccngetfile [1]) is used to fetch a content, and integrates a loss-based AIMD congestion control. The Interest messages generated by the AIMD entity are sent to the underlaying CCN node multipath forwarding functionality, which implements a per-packet strategy. The strategy determines how to distribute the Interest messages on the N upstream paths. When an Interest reaches a source S (repository or in-network cache) at the end of the path, the source sends back the related Data message. The Data message retraces the path followed by the Interest message, in the downstream direction, reaches the CCN node and then the AIMD entity. To simplify the model, we assume that a source (be it a repository or an in-network cache) has all the content chunks, i.e. we are not modeling the case of in-network caches having a subset of content chunks only.

We also assume that congestion can occur on the downstream path only. Thus, we model the ith upstream path as a simple delay line, with a constant propagation delay equal to D_i seconds. In addition, similarly to [8], we model the ith downstream path with a fixed propagation delay equal to D_i seconds, with a FIFO buffer of size L_i Data packets, which is emptied with a rate of R_i Data messages per second. This simple queuing system is used to model the slowest link of the

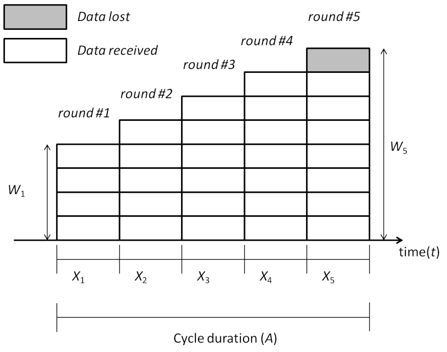

Fig. 2. Evolution of the AIMD congestion window

downstream path (its bottleneck), while the remaining links of the path are considered lossless.

We consider a receiver-driven congestion control mechanism that always maintains in the network W in-flight Interest messages, related to the next W missing chunks. A new Interest is immediately sent out after the reception of each Data, irrespective of whether the received Data is in order or not. The congestion window size W is controlled by an AIMD scheme, which increases it by one Interest per window W of Data received, and halves the window when a Data loss occurs. We assume that Data loss is detected immediately after the loss occurrence, i.e. we are not considering the possible detection delay e.g. due to time-out overestimation.

Let us consider the download of a given content. We consider a generic multipath forwarding strategy and define $P_i(H)$ as the *average* number of pending Interest messages injected by the considered strategy on the upstream face i, for $i = 1 \dots N$, when the strategy is handling a fixed number of H pending Interest of the considered content. We define the vector function $P(H)$ whose elements are $P_i(H)$.

$$\begin{aligned} & P(H) \longrightarrow \{P_i(H)\} \\ & \text{s.t.} \\ & \sum P_i(H) = H \\ & P_i(H) \geq 0 \; i = 1 \dots N \end{aligned} \tag{1}$$

In the following we will refer to $P(H)$ as the *sharing function* of the forwarding strategy. This characterization of a strategy is a key feature of our model, which makes it simple and general. In the following description of the model we consider $P(H)$ as a generic function; then in section IV we specialize the function $P(H)$ to specific strategies, to evaluate their performance. To make an example here, the sharing function of a strategy aimed at balancing the number of pending Interest messages across the upstreams faces is $P_i(H) = H/N$, for $i = 1 \dots N$.

In the considered network model (fig. 1), the multipath forwarding strategy handles packets controlled by the AIMD entity, with a one-to-one relationship, thus the overall number of pending Interest H handled by the strategy is equal to the congestion window size W of the AIMD.

We model the evolution of the congestion window size in terms of "rounds". A round starts when the AIMD algorithm changes the value of the window size W and ends either when W Data messages are received or when they would have been received if loss did not happen (i.e. at the expiry of the ideal time-out related to the expected receipt of such messages). Thus, the window size remains constant during a round.

We define as "cycle" a sequence of rounds without losses following a lossy round and including the first following lossy round. For instance, in fig. 2 we have a cycle made up of five rounds. In the first round the congestion window W_1 is equal to 4 Interest messages. In the fifth round a Data loss occurs; thus, this is the last round of the cycle and the maximum congestion window reached during the cycle is $W_5 = 8$.

Since the network model does not consider random phenomena (e.g. random loss, delay, etc.), the congestion window behavior is periodic and formed by a sequence of equal cycles. Therefore, to evaluate average performance it is sufficient to compute average performance in a cycle.

To evaluate the average receive-rate Y in a cycle, we approximate the number of received Data messages with the number of sent Data messages, which is equal to the number of sent Interest messages T. Defining with A the duration of a cycle, the receive-rate Y can be written as:

$$Y = T/A \tag{2}$$

Let us now determine the value of T. During a cycle, the congestion window increases from a minimum value equal to $\lfloor W_{max}/2 \rfloor$ up to the maximum value W_{max}. The number T of Interest messages sent out by the AIMD entity in a cycle can be written as:

$$T = \sum_{k=\lfloor W_{max}/2 \rfloor}^{W_{max}} k \tag{3}$$

In a round, a Data loss occurs when the number of pending Interest injected in any path by the strategy is greater than the *pipeline capacity* of the path, i.e. the sum of the bandwidth-delay product $R_i \cdot (2 \cdot D_i)$ and of the buffer space (L_i). Therefore, the maximum congestion window W_{max} reached in a round can be evaluated by solving the following integer maximization:

$$\begin{aligned} & \max \quad W \\ & \text{s.t.} \quad P_i(W) \leq R_i \cdot (2 \cdot D_i) + L_i, \; i = 1, \dots, N. \end{aligned} \tag{4}$$

Let us now determine the duration A of a cycle. As shown in fig. 2, A is equal to the sum of the duration X_k of the rounds of the cycle, where k is the round index within the cycle, i.e.:

$$A = \sum_{k=1}^{W_{max} - \lfloor W_{max}/2 \rfloor} X_k \tag{5}$$

A round k lasts for the time needed to exchange a number of Data messages equal to the congestion window W_k of the AIMD during that round, which is equal to

$$W_k = \lfloor W_{max}/2 \rfloor + (k-1) \tag{6}$$

Defining as B_k the overall receive-rate in the round k, the duration X_k of round k can be written as:

$$X_k = W_k / B_k \tag{7}$$

Each path contributes to B_k. The contribution $B_{k,i}$ of the ith path is equal to the ratio between the number of in-flight Interest messages $P_i(W_k)$ on the path and the path round trip time RTT_i. Thus, we can write:

$$B_k = \sum_{i=1}^{N} B_{k,i} \tag{8}$$

$$B_{k,i} = \frac{P_i(W_k)}{RTT_i} \tag{9}$$

$$RTT_i = max\{2 \cdot D_i, P_i(W_k)/R_i\} \tag{10}$$

The above equation gives an approximation of RTT_i similar to the one used in [8]. Indeed, when the number of in-flights messages is lower than the bandwidth-delay product, the path performance are *delay-dominated* and RTT_i is equal to the propagation delay. Otherwise, the path performance are *bandwidth-dominated* and RTT_i is equal to the ratio between the number of in-flight message and the available rate R_i.

IV. MULTIPATH FORWARDING STRATEGIES

In this section we present five strategies: two rather generic ones, namely Pending Interest Equalization (PE) and RTT Equalization (RE); the strategy proposed in [11] (UG); the strategy proposed in [6] (CF); and our own, Fast Pipeline Filling (FPF). For each strategy we model the sharing function $P(H)$ (eq. 1), which enables to analytically compute the receive-rate Y by means of eq. 2. These strategies monitor the characteristics of the path (e.g. number of pending Interest, RTT, etc.) for each content.

A. Pending Interest Equalization (PE)

The goal of this strategy is to balance the number of pending Interest on the different N paths. For each received Interest the strategy chooses the face with the lowest number of pending Interest messages; in case of equality, a random face is chosen. The sharing function $P(H)$ can be readily written as

$$P_i(H) = H/N \tag{11}$$

B. Round Trip Time Equalization (RE)

The goal of this strategy is to equalize the round trip time observed on the different faces. For each received Interest, the strategy chooses the face with the lowest RTT. In doing so, the RTTs tend to be equalized since increasing the number of pending Interests on a path, the path RTT increases too or remains constant (see eq. 10). From another point of view, this strategy could also be seen as a greedy approach to minimize the RTT. Since the RTT is not a linear function (eq. 10), it is not easy to evaluate the sharing function $P(H)$ of the strategy with a closed formula. For this reason, we resort to the recursive algorithm 1 below, in which at each step the face with the lowest RTT is selected.

Algorithm 1 Computation of $P(H)$ for RTT Equalization

```
procedure RTT($d, r, pi$)
    return $max\{2 \cdot d, pi/r\}$
end procedure

procedure P(H)
    $P_i(H) = 0$ for $i = 1 \ldots N$
    $S = 1 \ldots N$
    for $x = 1 \ldots H$ do
        Select $i \in S$ s.t. $RTT(D_i, R_i, P_i(H))$ is min
        $P_i(H) = P_i(H) + 1$
    end for
    return $\{P_i(H)\}$
end procedure
```

C. Strategy of [11] (UG)

This strategy distributes incoming Interests among faces by using a weighted round robin logic, with the weight z_i of face i being inversely proportional to that face round trip time RTT_i, i.e.

$$z_i = \frac{1}{RTT_i \cdot \sum_{j=1}^{N} RTT_j^{-1}} \tag{12}$$

Rather surprisingly, we found that the sharing function $P(H)$ of this RTT-based strategy is equal to the one of the pending Interest equalization strategy. Indeed, during a round k the receive-rate $B_{k,i}$ of path i is the one reported in eq. 9. Since the UG strategy consists of a weighted round robin scheme, $B_{k,i}$ is also equal to the overall rate B_k multiplied by the weight z_i. Consequently we can write the following equations:

$$\begin{aligned} &\frac{P_i(H)}{RTT_i} = \left(\sum_{j=1}^{N} \frac{P_j(H)}{RTT_j} \right) \cdot z_i \text{ for } i = 1 \ldots N \\ &\sum_{i=1}^{N} P_i(H) = H \end{aligned} \tag{13}$$

The solution of these equations is simply $P_i(H) = P_j(H) = H/N$ for any i, j, i.e. the sharing function of the pending Interest equalization (PE) strategy. This implies that the two strategies will result in the same receive-rate, even though the PE strategy has a simpler implementation since it does not require to estimate the RTT.

D. Strategy of [6] (CF)

This strategy distributes incoming Interests on faces by using a weighted round robin logic, with the weight z_i of face i being inversely proportional to its number of pending Interest messages P_i, i.e.

$$z_i = \frac{1}{P_i \cdot \sum_{j=1}^{N} P_j^{-1}} \tag{14}$$

The sharing function can be computed by using eq. 14 in eqs. 13. After some simple algebra eqs. 13 can be written as:

$$\frac{P_i(H)}{\sqrt{R}TT_i} = \frac{P_j(H)}{\sqrt{R}TT_j} \text{ for any } i, j \tag{15}$$

$$\sum_{i=1}^{N} P_i(H) = H \tag{16}$$

Eq. 15 shows that the sharing function $P(H)$ of the CF strategy resembles the one of a weighted PE strategy, whose weights are the square root of the round trip times. Thus, paths with higher round trip time will have more pending Interests with respect to the PE strategy. To evaluate the sharing function $P(H)$, we resort to an approximated method given by the following iterative algorithm 2. At each iteration step, the face with the lowest $P_i(H)/\sqrt{RTT_i}$ is selected.

Algorithm 2 Computation of $P(H)$ for CF strategy

```
procedure P(H)
    $P_i(H) = 0$ for $i = 1 \ldots N$
    $S = 1 \ldots N$
    for $x = 1 \ldots H$ do
        Select $i \in S$ s.t.
        $P_i(H)/\sqrt{RTT(D_i, R_i, P_i(H))}$ is min
        $P_i(H) = P_i(H) + 1$
    end for
    return $\{P_i(H)\}$
end procedure
```

E. Fast Pipeline Filling (FPF)

Our FPF strategy has been motivated by insights enabled by our model. Its goal is to completely fill the pipeline capacity of the different paths, and to achieve this saturation condition as fast as possible. In doing so, the value W_{max} reached by the congestion window during a cycle is the maximum possible one, the cycle duration A is the shortest possible one, and this choice maximizes the receive-rate of eq. 2. For each received Interest, the FPF strategy identifies the set S of faces whose number of pending Interest messages is lower than the related pipeline capacity (C_i). Within this set, the strategy selects the face with the lowest RTT. The sharing function $P(H)$ can be computed by means of algorithm 3.

Algorithm 3 Computation of $P(H)$ for FPF

```
procedure P(H)
    $P_i(H) = 0$ for $i = 1 \ldots N$
    $C_i = 2 \cdot R_i \cdot D_i + L_i$
    for $x = 1 \ldots H$ do
        Form the set $S$ of face indexes $i$ s.t. $P_i(H) < C_i$
        Select $i \in S$ s.t. $RTT(D_i, R_i, P_i(H))$ is min
        $P_i(H) = P_i(H) + 1$
    end for
    return $\{P_i(H)\}$
end procedure
```

V. ANALYTICAL AND SIMULATION RESULTS

To assess the validity of the analytical model, we developed an event-driven Matlab simulator reproducing the scenario reported in fig. 1; then we carried out a set of tests considering two paths. The first path has a delay $D_1 = 20ms$, a queue length $L_1 = 20$ Data messages and a transmission rate $R_1 =$ 10 Mbps. The length of a Data message is 4876 bytes, 4096 bytes of payload and 780 bytes of CCN/UDP/IP overhead. This value has been taken from CCNx measurements. Then, we varied one at a time the delay D_2 and the transmission rate R_2 of the second path, while keeping the unvaried parameter equal to the one of path 1. We compared the receive-rates of the considered multipath forwarding strategies. We point out that the aim of the comparison is not to judge which is the best strategy, considering also that our scenario is very simple and these strategies do not have all the same objectives. The aim of the comparison is to show how the model can be used to gain insights on the behavior of the receive-rate in presence of a multipath forwarding strategy.

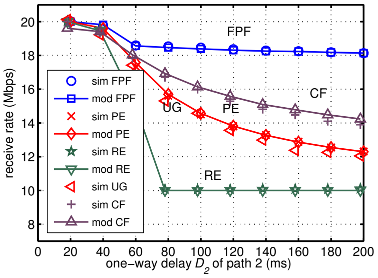

Fig. 3. Receive-rate Y for two paths versus D_2, with $D_1 = 20$ ms, $R_1 = R_2 = 10$ Mbps, $L_1 = L_2 = 20$ Data messages

Fig. 3 reports the receive-rate versus the delay of the second path. We observe that model (mod) and simulation (sim) results are very close to each other, thus confirming the validity of the model in the scenario of fig. 1.

In case of homogeneous paths ($D_2 = D_1 = 20ms$) all strategies provide the same performance. In average, they equally share the load on both paths, and this is an optimal result for the receive-rate, in case of symmetric paths.

As the delay D_2 increases, the FPF strategy shows the best performance, since it is able to quickly fill the capacity of the pipelines of both paths. As a consequence the AIMD congestion window reaches the highest possible value between data loss events and the receive-rate performance is the best one. This behavior is shown in fig. 4, which reports the evolution of the congestion window for $D_2 = 120$ ms, for the PE and FPF strategies. The capacity of the pipeline of paths 1 and 2 is $C_1 = 2 \cdot R_1 \cdot D_1 \approx 30$ and $C_2 \approx 81$ Data messages, respectively. As the congestion window increases up to 30, the FPF strategy injects all messages on path 1, which has the lowest RTT. When the window is greater than 30, the FPF strategy maintains the number of in-flight Interests on path 1 equal to 30 and starts to inject additional in-flight Interest messages in path 2. A first loss occurs when the congestion window becomes greater than the sum of the pipeline capacity of the two paths, i.e. $C_1 + C_2 = 111$. After the first loss, the congestion window drops to 55 and then restarts its growth. The FPF strategy maintains path 1 filled with 30 in-flight Interest messages, other messages are injected in path 2 and a new loss occurs when the congestion window reaches again the value 111. Fig. 3 confirms the findings anticipated in section IV-C: the PE and UG strategies have the same (average) performance. Their sharing function, which equalizes the number of in-flight Interests, makes the smallest pipeline a limiting

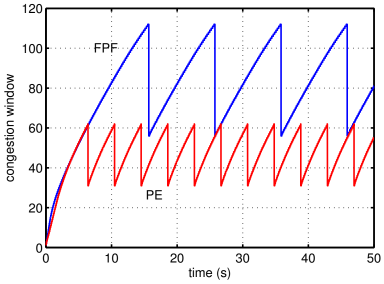

Fig. 4. Congestion window versus time in case of two paths, with $D_1 = 20$ ms, $D_2 = 120$ ms, $R_1 = R_2 = 10$ Mbps, $L_1 = L_2 = 20$ Data messages

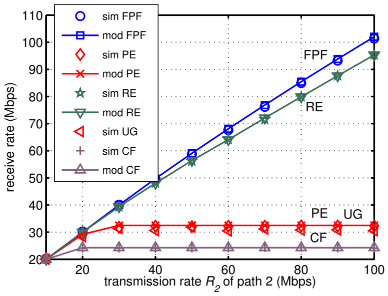

Fig. 5. Receive-rate Y for two paths versus R_2, with $D_1 = D_2 = 20$ ms, $R_1 = 10$ Mbps, $L_1 = L_2 = 20$ Data messages

factor of the AIMD growth. Indeed, when the smallest pipeline is filled with in-flight messages a drop occurs, even if the pipelines of the other paths are partially available. This partial exploitation of pipelines explains the lower performance with respect to the FPF strategy. For instance, in the scenario of fig. 4, as the congestion window increases, the PE strategy equally distributes the in-flight Interest messages between the two paths. Consequently, when the congestion window reaches the value 62, there are 31 in-flight Interest messages on path 1, this value is above the capacity of the pipeline of path 1 and a first loss occurs. After this first loss, the congestion window drops to 31 and restarts its growth; the PE strategy equally distributes in-flight Interests between the two paths and a new loss occurs when the congestion window reaches again 62.

Fig. 3 shows that the performance of the CF strategy is in-between FPF and PE/UG. Since the CF strategy behaves as a weighted PE whose weights are the square root of RTTs (see section IV-D), it maintains a greater value of in-flight Interests on path 2 (whose RTT is greater), with respect to the PE/UG strategy. This allows AIMD to reach a greater value of the congestion window between losses, i.e. to achieve a greater receive-rate. However, loss typically occurs on path 1 before having saturated the pipeline capacity of path 2; thus, the performance of the CF strategy is lower than that of FPF. The RE strategy results in the worst performance in terms of receive-rate. As D_2 increases, it tends to waste the second path since its RTT is greater than the one of path 1; thus, the receive-rate decreases to the rate of path 1, i.e. 10 Mbps.

Fig. 5 shows the receive-rate vs. the rate R_2 of path 2. The FPF strategy provides the best performance. The RE strategy performs rather well since it favors path 2, which has the greater rate and, consequently, a lower RTT, due to its smaller queening delay (see eq. 10). The PE and UG performance are limited to the small pipeline capacity of path 1 and the achieved rate is roughly two times the rate of path 1, i.e. 20 Mbps. For the CF strategy this scenario is clearly critical, since it tends to use the slower path 1 even more than the PE and UG approaches, since path 1 has an higher RTT.

VI. CONCLUSIONS

The main aim of this paper is to propose a model useful to understand the performance of ICN multipath forwarding strategies, when using a path-unaware AIMD congestion control, that is in line with current ICN architectures in which the network layer does not provide path-related information to higher layers. A by-product of the paper is that a good forwarding strategy to maximize the receive-rate should control the pending Interests injected in the different paths so as to fill the capacity of their pipelines. This is the rationale followed by the FPF strategy. Future work will consider path-aware congestion control schemes and random losses in the network model of fig. 1, devising a strategy suitable also for wireless environments, and also other performance maximization figures, e.g. delay. Finally, we remark that in [7] we carry out an experimental campaign by using the PlanetLab test-bed to evaluate CCNx-based implementations of literature strategies and of our proposal, in realistic network settings. All the related software is freely available [2].

REFERENCES

[1] CCNx project. http://www.ccnx.org.
[2] M. path sw. https://www.dropbox.com/s/hoi9lyi8yyq6j8o/MPSoftware.zip.
[3] Named Data Networking (NDN) project. http://named-data.net.
[4] S. Braun, M. Monti, M. Sifalakis, and C. Tschudin. An empirical study of receiver-based aimd flow-control strategies for ccn. In *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*. IEEE, 2013.
[5] G. Carofiglio, M. Gallo, L. Muscariello, and M. Papalini. Multipath congestion control in content-centric networks. In *INFOCOM NOMEN Workshop, 2012 IEEE Conference*. IEEE, 2013.
[6] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, and S. Wang. Optimal multipath congestion control and request forwarding in information-centric networks. In *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, 2013.
[7] A. Detti, L. Pisa, and N. Blefari Melazzi. Multipath forwarding strategies in information centric networks with aimd congestion control. *arXiv.org, http://arxiv.org/abs/1412.2204*, 2014.
[8] V. Jacobson. Congestion avoidance and control. 18(4):314–329, 1988.
[9] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *5th international conference on Emerging networking experiments and technologies*. ACM, 2009.
[10] L. Saino, C. Cocora, and G. Pavlou. CCTCP: A scalable receiver-driven congestion control protocol for content centric networking. In *IEEE International Conference on Communications (ICC)*. IEEE, 2013.
[11] A. Udugama and C. Goerg. Adaptation of multi-path forwarding strategies for ccn to operate under mobility. In *Presented at CCNxCon2013*. Parc, 2013.
[12] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, and G. Polyzos. A survey of information-centric networking research. *Communications Surveys Tutorials, IEEE*, PP(99):1–26, 2013.
[13] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang. A case for stateful forwarding plane. *Computer Communications*, 36(7):779–791, 2013.