Cooperative Congestion Control in NDN

Adrien Thibaud*†, Julien Fasson*†, Fabrice Arnal‡, Renaud Sallantin‡, Emmanuel Dubois§ and Emmanuel Chaput*†
*TéSA Laboratory, firstname.name@tesa.prd.fr
†ENSEEIHT/IRIT/University of Toulouse, firstname.name@enseeiht.fr
‡Thales Alenia Space, firstname.name@thalesaleniaspace.com
§Centre National d'Études Spatiales, emmanuel.dubois@cnes.fr

***Abstract*—Named Data Networking (NDN), an Information-Centric Network (ICN) architecture, is based on caching, multipath and multi-producers retrieving. These properties provide new opportunities for a single user to increase its Quality of Experience (QoE). However, handling multiple flows, each of them having its own multiple paths, is more complex. To tackle this challenge, we highlight three main principles a solution should include. Nodes should cooperate, supervise their output queues and, eventually, wisely manage the multipath capacities of NDN. These three elements are the core of our proposition: Cooperative Congestion Control (CCC). More than a solution, CCC is proposed as a framework where each principle could be implemented in multiple ways. The ultimate objective is to fairly distribute the flows on the network and maximize QoE of users. We choose basic algorithms in order to evaluate the overall framework. We evaluate our solution with simulations and compare their results with a theoretical model.**

***Index Terms*—ICN, NDN, QoE, Max-min fairness, Multipath flows, Congestion Control**

I. Introduction

A new approach, the Information Centric Network paradigm (ICN), has recently emerged and defines a network layer that is content-oriented instead of location-oriented. Among the ICN architectures, Named Data Networking (NDN) [1] seems promising and increasingly draws the attention of the research community. NDN uses the mechanisms that made the success of HTTP, CDN (Content Delivery Network) and P2P (Peer-to-Peer network) directly as a basis for the network layer. NDN is a receiver-driven architecture. As in HTTP with its GET message, the user (or the consumer in the NDN terminology) sends an Interest message requesting a piece of data. Multiple producers (as in P2P) and multiple paths might be available and used, but everything is transparent from the end-users point of view. Furthermore, each forwarding node is able to opportunistically cache contents and extra nodes might be set to cache the most popular content at strategical places (as in CDN). NDN design also integrates security: a data packet is intrinsically secured and does not depend on the end points. It thus can be reused for multiple users, facilitating the caching and multicasting of contents.

The challenge of this architecture is to design a more compliant network with user needs through a data-oriented approach. We consider here a network where multipaths are common and that is potentially overloaded (i.e. the global consumers need is above the network capacity). Indeed, as pictured in Fig. 1, consumer A can retrieve its requested content through two different paths. But the network capacity is not able to satisfy consumers A and B simultaneously.

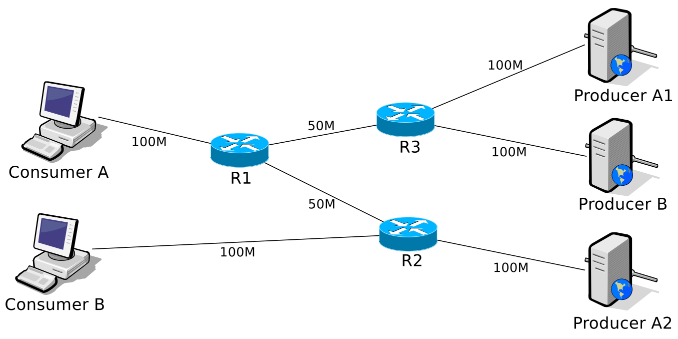


Fig. 1: Illustrative network topology

Given this context, our goal is to maximize the Quality of Experience (QoE) of all the users. In this paper, we only consider content-oriented applications that need a targeted rate. The QoE is evaluated through the received rate of the users. A user is increasingly satisfied with the rate he gets (capped by its personal objective rate). The competition of multiple users on the network rises a fairness problem. In our example in Fig. 1, the consumer A is in competition with the consumer B on the R1-R3 path, while the other path could have sufficient unused bandwidth to satisfy consumer A without bothering other users on the congested path. Ultimately, we search to distribute the flows of the users with a fairness concern, through all the paths available and with regards to the resources constraints.

Therefore, the problem can be seen as a joint congestion and fairness problem. In classical IP networks, two methods are used to handle this issue: an end-to-end one and a hop-by-hop one. The best known example of the first method is TCP. The intelligence of this approach is located in the sender. The receiver only sends acknowledgment that are used by the sender to evaluate the network congestion level. The goal of the sender is to reach the best rate possible, possibly in an aggressive way. The second method is implemented through AQM solutions (Active Queue Management). For this approach, the intelligence is located in every node on the network. The nodes are able to detect when a congestion occurs and take action to avoid it. On the one hand, end-to-end methods cannot efficiently evaluate congestion phenomenon and tend to increase the rate of the user in order to increase their QoE, and on the other hand, hop-by-hop methods tend to decrease it in order to reduce the congestion. Even if those two kinds of method sometimes implement opposite actions, they actually cohabit in the Internet and have shown some efficiency. We thus believe that a NDN based solution should take the best of these two schemes. However, we think that they should be implemented in a single framework in order to be more consistent and thus, hopefully, effective. Furthermore, multipath retrieving is not natively handled in IP. MPTCP is the TCP solution for multipath, but it only works when one of the ends is multi-homed (with Fig. 1 example, MPTCP would not be feasible).

NDN provides the opportunity to manage multipath delivery on the network level and we are convinced that such a feature should be used in our framework. In this paper, we propose a new solution, Cooperative Congestion Control (CCC), that tackles these joint problems (congestion and fairness) in a more cooperative and effective way. CCC is based on a per node approach where the main elements are: a two ways cooperation between nodes, a local supervision of the output queues and a smart utilization of the multipath possibilities. Combined, these three principles can support a fair distribution of the flows.

The remaining sections of this paper are organized as follows. In Section II, we survey the related work. Then, Section III, we study the theoretical problem. Section IV presents the principles of CCC and how we implement it. In Section V, we present the evaluations of our solution. Lastly, the paper is concluded in Section VI.

II. Related Works

In term of congestion control, they are two main families of solutions: the end-to-end and the hop-by-hop type of solutions. A third and hybrid solution has quickly emerged since it allows to take advantage of NDN natural hop-by-hop processing while maintaining an end-to-end control like in IP with TCP-like protocols. All the solutions presented here are compared with our proposition in Sec. V.

Interest Control Protocol [2] (ICP) is the best known solution for congestion control, since it is an adaptation of TCP behavior for NDN. It is receiver-driven as well as an end-to-end solution. The consumer handles a congestion window for the content to be retrieved. The size of this window evolves using an Additive Increase Multiplicative Decrease mechanism (AIMD). When a Data packet arrives at the consumer, it additively increases the size of the congestion window. This additive increase corresponds to the congestion avoidance phase of legacy TCP: the congestion window increases of one in a round-trip time (RTT). When congestion is detected by a timeout, the consumer decreases the window multiplicatively (divided by two here). The Retransmission Timeout (RTO) in ICP is computed as the mean between the minimum and the maximum of the measured RTTs.

A Practical Congestion Control Scheme [3] (PCON) is an hybrid solution that defines how consumer sends its requests (end-to-end) and how nodes in the network handle Data and Interests (hop-by-hop). The end-to-end part is based on a congestion window at the consumer side whereas the hop-by-hop part is a forwarding strategy on each NDN nodes. We respectively name them PCON-CS and PCON-FS in this paper. PCON-FS uses the CoDel approach [4] to detect the congestion: the nodes measure the sojourn time of each packet in their queue. If the minimum detected over a given period is higher than a threshold, the interface is considered as congested. The interface marks Data packets rather than dropping them in order to trigger an explicit adaptation from the upstream nodes. With the marks it receives, the traffic is progressively load balanced on the other available paths. The PCON-CS consumer has the same behavior as ICP: a congestion avoidance phase and the RTO computation from RFC 6298 [5]. In addition, a marked Data is considered as a congestion notification and triggers a multiplicative decrease.

Multipath-aware ICN Rate-based Congestion Control [6] (MIRCC) has similar goals than CCC. It targets to achieve a max-min fairness using the network at its full capacity. For this purpose, the authors propose an adaptation of RCP [7] for NDN with multipath compatibility. Their algorithm is partially distributed: every node computes two stamping rates (as in RCP but with a second for multipath support) and each consumer chooses the paths it wants to use and at which rate (with the help of the two stamping rate). As opposed to our solution, MIRCC puts a lot of intelligence in the consumers and less in the nodes.

III. Model

In this part, we search the theoretical optimal solution for our problem. For this purpose, we define a model where we compute the fairest flow distribution on the considered network. This will help us to analyze our simulation results. We split our approach in two steps. In the first step, we maximize the throughput of the users and thus their QoE. In the second step, we distribute this throughput fairly to satisfy as many users as possible. Indeed, we prefer an average satisfaction for each user than to have a few users with the maximum QoE and some users with the worst. These two parts represent one optimization problem each and are described in the following sub-sections.

A. Maximum Multi-Commodity Flow problem

The first part is an adaptation of the common Maximum Multi-Commodity Flow problem [8]. It can be formulated with the following objective:

$$\max \sum_k F^k \tag{1}$$

and the following constraints:

$$\sum_k f^k_{i,j} - \sum_k f^k_{j,i} = \begin{cases} F^k & i = s_k \\ 0 & i \neq s_k, t_{k,l} \\ -F^{k,l} & i = t_{k,l} \end{cases}, \tag{2}$$

$$\forall (i,j) \in E, \forall l = 1, \dots, L_k, \forall k = 1, \dots, K$$

$$\sum_k f^k_{i,j} \leq u_{i,j},\ \forall (i,j) \in E \tag{3}$$

where K is the number of competing flow on our network, F^k is the total size of the kth flow, L_k is the number of nodes providing the kth flow, E is the set of edges of the network, s_k is the node requesting the kth flow, $t_{k,l}$ is the lth node providing the kth flow, $F^{k,l}$ is the size of the lth path of the kth flow, $f^k_{i,j}$ is the size of the kth flow on the edge (i,j) (i and j are vertices of the network) and $u_{i,j}$ is the capacity of this same edge. (2) represents the flow conservation constraints while (3) represents the capacity constraints. Furthermore, we suppose that our users have explicit rate objectives:

$$F^k \leq F^k_{obj}, \forall k = 1, \ldots, K \tag{4}$$

where F^k_{obj} is the objective size of the kth flow.

The solution of this optimization problem gives the maximum aggregate throughput for all users and can be formulated as a new constraint:

$$\sum_k F^k = z^* \tag{5}$$

where z^* is the solution found. We thus define a new version of the optimization problem aiming to optimize fairness.

B. Fairness concern

Despite the fact that the aggregate throughput is maximized, the rate distribution might not be fair. Indeed, one user may have a very good throughput and penalizing another user at the same time. A fair share may decrease reasonably the QoE of this user and increase significantly the one of the unsatisfied users. It is therefore necessary to search for the fairest distribution, but without changing the optimal objective we found in the previous formulation. We decide to use the max-min fairness because it allows our users to get the maximum fair share possible without limiting them to get more if available. The second part of our model is then formulated with this new objective:

$$\max t \tag{6}$$

and these new constraints:

$$F^k - t \geq 0, \forall k = 1, .., K \tag{7}$$

where t represents the minimum of the F^k, through the Eq. (7). Solving the optimization problem presented by (6) as objective and (2), (3), (4), (5) and (7) as constraints gives one of the possible fairest distributions and the minimum size each flow should have in order to have an optimal distribution. Both optimization problems can be solved using the simplex algorithm.

IV. Cooperative Congestion Control

A. Principles

The algorithm we propose, Cooperative Congestion Control (CCC), defines a cooperative way to distribute the concurrent flows on a network. It works on a per node approach and aims at avoiding the congestion and using the multipath capacity of NDN efficiently. Our purpose is to give each flow the QoS it needs while preserving the network resources. To do so we will use both pacing (i.e. delay between consecutive transmission within the same flow) and multipath (i.e. the use of different available paths for the same flow). Our solution is based on three main principles: (i) a cooperation between nodes that takes the form of the exchange of applications objectives and networks constraints, (ii) a queue supervision to estimate its state and (iii) a smart decision making upon those pieces of information to allocate bandwidth to flows.

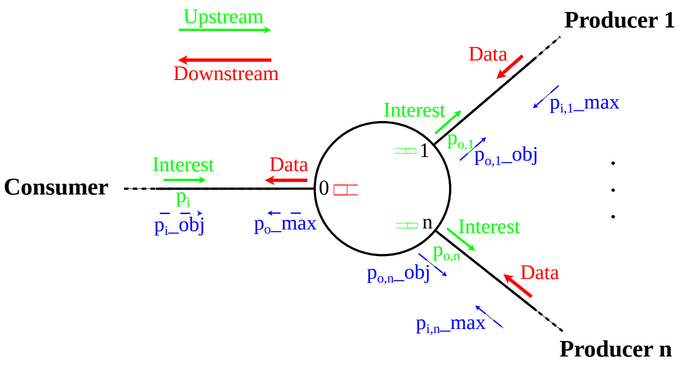


Fig. 2: Node context

B. Node behavior

Fig. 2 represents a NDN node with one flow. Interests arrive from the Consumer side and are distributed on the n available paths (Upstream). On the reverse direction, Data arrive from the Producers and are sent to the Consumer (Downstream). Our solution is based on the control of the interests pace (Interests per second) on the upstream, in order to offer the compliant downstream data rate. On the Fig. 2, Interests arrive at the incoming pace p_i on the interface 0 of our node. The $p_{o,k}$ ($k = 1, \ldots, n$) represents the outgoing pace of Interest through the path k. With the size of the Data packet of the flow, our algorithm is able to estimate the rate induced by the transmission pace of Interests.

For each flow it manages, a node receives a pacing objective, p_i_obj, from its neighbor on the consumer side and a constraint on this pacing, p_i_max, from its neighbors on each producer side. The objective represents the application needs and tends to increase the rate allocated to the flow. The constraint represents the network limitation and tends to restraint the rate allocated to the flow. This represents the cooperation part of the algorithm. Each node has to respect the pace constraint and tries to satisfy the pace objective it receives. Then, it has to send its own constraint and objectives.

A node receives an objective for a flow p_i_obj and can distribute it among the multiple output interfaces ($p_{o,k}\_obj, k = 1, \ldots, n$). Eq. 8 represents the *conservation of the objective* through a node. No objective is created anywhere on the network. Thus, at a global scale, the sum of the received objective by all the producers is equal to the initial objective emitted by the consumer.

$$p_i\_obj = \sum_{k=1}^{n} p_{o,k}\_obj \tag{8}$$

On the downstream, nodes receive n incoming constraints for a flow ($p_{i,k}\_max, k = 1, \dots, n$) and send their own outgoing constraint p_o_max such as :

$$p_o\_max \leq \sum_{k=1}^{n} p_{i,k}\_max \tag{9}$$

Eq. 9 represents the *aggregation of the incoming constraints*. It is not an equality because some local issue may reduce the outgoing constraint p_o_max. On our scheme, Consumers and Producers network layer also uses CCC. As a result, the application stack has to follow the constraints and creates objective that satisfies their user needs.

The local issue represents a misuse of the bandwidth by the node. For this problem, nodes set up a supervision of theirs downstream output interfaces (here interface 0 in Fig. 2). This allows nodes to estimate if congestion is happening or if unused bandwidth is remaining. Depending of the case, a node may reduce the constraints for each active flow or distribute the available bandwidth. Nodes have to respect Eq. 9 even when a constraint is increased, i.e. an outgoing constraint cannot be increased more than the aggregate incoming constraints. With this mechanism, NDN nodes try to reach the objective of each flow. Such a behavior may be unfair, so we need to implement a fairness algorithm. Furthermore, if all the active flows are satisfied and there is still some unused bandwidth, nodes are allowed to increase the outgoing constraints above their incoming objective. This permits the previous node to have more opportunities to balance its traffic.

Indeed, nodes have n output interfaces for the considered flow and they have to choose how to use them efficiently. Interests arrive at the incoming pace p_i and nodes have to determine the outgoing pace for all n output interfaces ($p_{o,k}, k = 1, \dots, n$). The only restrictions are the received constraints:

$$p_{o,k} \leq p_{i,k}\_max, k = 1, \dots, n \tag{10}$$

Finally, when an Interest arrives, nodes have to choose which output interface to use. Each output interface eventually paces Interests with their outgoing pace $p_{o,k}$.

C. Algorithm

The overall behavior of each node implementing is the following:

- the node receives pacing objectives (p_i_obj) and use an "Objective distribution" algorithm to compute the outgoing objectives ($p_{o,k}\_obj, \forall k$) ;
- it forwards Interest based on a pacing policy enforced by the "Load balancing" algorithm ;
- pacing is updated by a couple of "Pace increase" and "Pace decrease" algorithms ;
- these algorithms are triggered by a "Queue supervision" algorithm which is fed by a "Queue status" algorithm and a bandwidth estimation (Eq. 12).

As a consequence an effective CCC implementation consists in six sub-algorithms. We describe here a very straightforward implementation for each of them.

First, Interests carry the objective of the flow. When a node receives it, it does not know if it is the global objective of the consumer or if it is only a part of it. The only requirement of the node is the *conservation of the objective* (Eq. 8) and thus have to distribute it when multiple path are available:

Algorithm 1 Objective distribution

```
Require: $p_i\_obj = \sum_{k=1}^{n} p_{o,k}\_obj$
Ensure: $p_i\_obj = \sum_{k=1}^{n} p_{o,k}\_obj$
  for all output interface do
    Set the outgoing objective at an equal share of the received objective
  end for
```

For this first implementation, we decide to assign an equal part of the received objective to each output interface:

$$p_{o,k}\_obj = \frac{p_i\_obj}{n}, \forall k \tag{11}$$

With the objective distribution, the nodes try to encourage the use of multipath. But the load balancing is done by the pace distribution (Algo. 2) and driven by the received constraints.

Algorithm 2 Load balancing

```
Input: pace_to_distribute
  while pace_to_distribute is not null do
    Get an output interfaces not full ($p_{o,k} \leq p_{i,k\_max}$)
    Increase its output pace to its constraint (with respect to residual pace_to_distribute)
    Update the pace_to_distribute
  end while
```

It is used to distribute the incoming pace p_i and to distribute a potential overflow to an output interface (when a new constraint $p_{i,k}\_max$ is received and is below the current output pace $p_{o,k}$). Indeed, when a more restrictive constraint arrives, the Eq. 10 may not be respected. The output pace is then decreased to the new constraint and the overflow has to be distributed among the other output interface. If it is not possible, it is Eq. 9 that may not be respected. The outgoing constraint is then updated to respect the aggregation property.

For downstream and as for Interests, Data piggyback the constraint of its flow. The nodes have to do the *aggregation of the constraints*. They just have to ensure that the outgoing constraint respects the Eq. 9.

The nodes also set up a local supervision on each of their output interfaces (Algo. 3).

Algorithm 3 Queue supervision

```
  Get queue status (Algo. 4)
  if congestion then
    Reduce the pace of each active flow (Algo. 5)
  else
    Evaluate available bandwidth (Eq. 12)
    Distribute it equally between active flow (Algo. 6)
  end if
```

They use a drop-based mechanism to detect congestion (Algo. 4): if some packets on the emission queue are dropped during the supervision period, the queue is considered as congested. The supervision is done periodically.

Algorithm 4 Queue status

```
if at least one packet dropped then
    Return True
end if
Return False
```

If a congestion is detected, the nodes need to decrease the constraint of its active flows. The outgoing constraint of each flow is then set to 90% of the current incoming pace of Interests (as described in Algo. 5). The overall estimation of the link use is thus reduce by ten percent.

Algorithm 5 Pace reduction

```
for each active flow do
    Reduce outgoing constraint by ten percent
end for
```

If no congestion is detected, the nodes compute the current estimation of the link use :

$$bandwidth\_estimation = \sum_{flowF} p_o^F\_max * data\_size^F \quad (12)$$

If the bandwidth estimation is below the capacity of the link, the nodes can distribute the available bandwidth with Algo. 6.

Algorithm 6 Pace augmentation

```
Input: available_bandwidth
Input: constraint of active flows
    for each active flow do
        Give an equal part of the available bandwidth
    end for
```

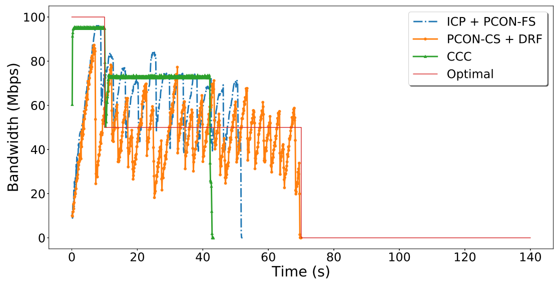
The single aspect of fairness of these algorithms is to manage each flow equally. Indeed, Algo. 5 and Algo. 6 have an obvious impact on the fairness of the flows distribution. But all six of these algorithms play a role in the fairness balance and with interdependence between each other. The relative simplicity of the proposed algorithms allows to reduce this interdependency and thus to analyze the performance the overall framework. In future works, we will study the use of advanced implementations in order to enhanced the fairness of our solution or propose multiple kinds of fairness.

V. Evaluations

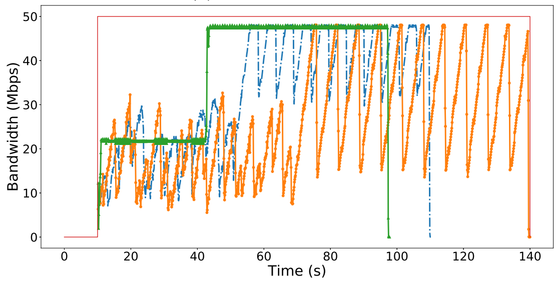
We have implemented CCC on the ndnSIM simulator [9]. All the codes and environment parameters are available[1] in order to be able to reproduce our results. We have chosen two scenarios to evaluate our solution.

The first scenario is illustrated on the Fig. 1. Both consumers have larger rate objectives than the network can offer. Each

[1]https://gitlab.tesa.prd.fr/athibaud/ccc-codebase



(a) Rate of flow A



(b) Rate of flow B

Fig. 3: Comparison between CCC, ICP and PCON

content size is 400 Mbytes and is split in chunks of 8400 bytes. The consumer A begins its request at $t = 0s$ while the consumer B begins at $t = 10s$. In this scenario, we compare our solution to ICP and PCON. As depicted in our previous work [10], we choose ICP as the end-to-end algorithm with PCON-FS as the hop-by-hop forwarding strategy and PCON-CS as the end-to-end algorithm with Dynamic Request Forwarding [11] (DRF) as the hop-by-hop forwarding strategy. The optimal fair distribution found with the model of Sec. III is also included. It represents the minimum rate each flow should reach in order to have a global max-min fairness.

Fig. 3 shows the results of this scenario. Sub-figure 3a represents the rate of the flow A. When the flow is alone, it should theoretically reach 100Mbps as pictured by the optimal curve. However, the rate obtained by CCC is below the optimal curve. Indeed, 100Mbps is the level 2 link capacity while we trace the level 3 rate. Therefore, the difference is due to the overhead of the layer 2. At $t = 10s$, the second flow begins and the max-min fair rate (the optimal curve) drops to 50Mbps for each flow. Our solution is very stable in comparison to ICP and PCON. Indeed, our consumer never has to blindly increase its Interests transmission rate since the network explicitly notifies the procedure to follow. The convergence time is also very short: approximately one supervision period to reallocate the bandwidth and a few supervision periods to decrease the congested link below its capacity. Therefore, the completion time of both consumers is lower with our solution. It successfully uses the network resources with no oscillation and with a fast convergence when new flows appear or old flows disappear. In term of fairness, our solution has the same performance than the combination "ICP + PCON-FS" and the "PCON-CS + DRF". It achieves a local, per link, max-min fairness: during the competition, both flows get half the bandwidth of the congestion link (R1-R3).

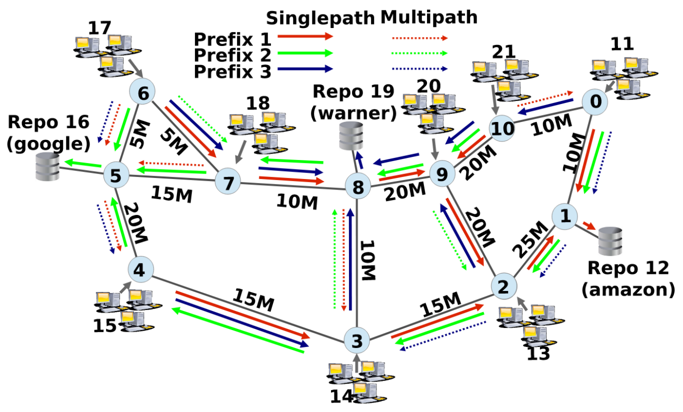


Fig. 4: Topology abilene [11]

The global fairness is not reached since the flow B is above the max-min fair rate computed by our model.

The second scenario is done on the Abilene topology of Fig. 4 as used in [11]. This topology represents a bigger and more realistic network. The figure shows how the routing tables are filled. We conduct the same experimentation as in [6] and compare it to the results with MIRCC. Consumer arrival time follows a Poisson process with mean 3 seconds. It is randomly set on the nodes 11, 13, 14, 15, 17, 18, 20 or 21 and requests randomly a content among /amazon, /google and /warner. Each content size is 9Mbytes and is chunked in 5000 pieces.

Fig. 5 presents the completion times of MIRCC and CCC solutions. Our solution does not handle requests coming from multiple interfaces yet. This is why there is no result for four consumers (C20 for /google, C18 for /amazon, C17 for /amazon and C11 for /warner). We plan to add this feature in future work. However, our proposition successfully retrieves each content faster that the MIRCC solution. This scenario also demonstrates that CCC is functional on a large scale simulation.

VI. Conclusions

In this paper, we have described the problem of multi-flow fair distribution on a limited network and its theoretical model. Then, we have proposed a solution, Cooperative Congestion Control, based on NDN that tackles this difficult challenge through a comprehensive set of algorithms. It works in a distributed way: consumers, producers and nodes exchange objectives and constraints for each flow, measure local utilization of each link and use these pieces of information to allocate fairly the rate of the active flows. The implementation of each algorithm can be changed independently and we have first defined simple algorithms in order to demonstrate the efficiency of our design. Our solution successfully uses the multipath capacity of NDN and maximizes the rate and QoE of the users. Our local queue supervision ensures a rapid re-allocation of flows. It provides a fast convergence time when a flow appears and disappears. Simulation results show that we can achieve local link max-min fairness: each active flow gets a fair share of the link bandwidth. This is the case for each link. As future works, we plan to study the performance of different

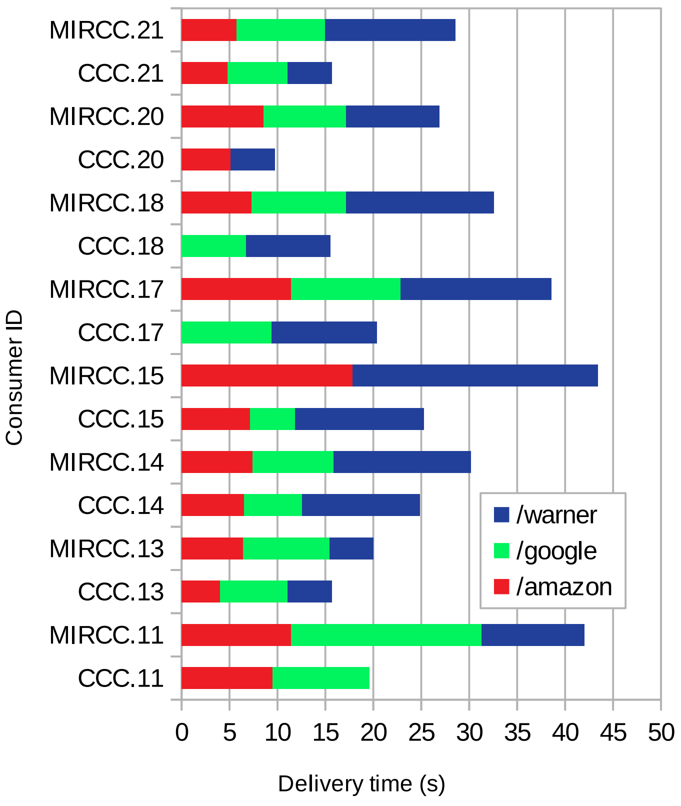


Fig. 5: CCC/MIRCC comparison on the Abilene topology

implementations for CCC sub-algorithms. A key issue is the fairness of the system. We have shown that CCC is fair at the link level. A global fairness is a much more challenging objective and involves several of these algorithms.

References

[1] L. Zhan and *al.* Named data networking. *SIGCOMM Comput. Commun. Rev.*, 44(3):66–73, July 2014.

[2] G. Carofiglio, M. Gallo, and L. Muscariello. Icp: Design and evaluation of an interest control protocol for content-centric networking. In *2012 Proceedings IEEE INFOCOM Workshops*, pages 304–309, March 2012.

[3] K. Schneider, C. Yi, B. Zhang, and L. Zhang. A practical congestion control scheme for named data networking. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ACM-ICN '16, pages 21–30, New York, NY, USA, 2016. ACM.

[4] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar. Controlled Delay Active Queue Management. RFC 8289, January 2018.

[5] M. Sargent, J. Chu, Dr. V. Paxson, and M. Allman. Computing TCP's Retransmission Timer. RFC 6298, June 2011.

[6] Milad Mahdian, Somaya Arianfar, Jim Gibson, and Dave Oran. Mircc: Multipath-aware icn rate-based congestion control. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ACM-ICN '16, pages 1–10, New York, NY, USA, 2016. ACM.

[7] Nandita Dukkipati. *Rate Control Protocol (Rcp): Congestion Control to Make Flows Complete Quickly*. PhD thesis, Stanford, CA, USA, 2008. AAI3292347.

[8] T. C. Hu. Multi-commodity network flows. *Operations Research*, 11(3):344–360, 1963.

[9] S. Mastorakis and *al.* ndnSIM 2: An updated NDN simulator for NS-3. Technical Report NDN-0028, Revision 2, NDN, November 2016.

[10] Adrien Thibaud, Julien Fasson, Fabrice Arnal, Renaud Sallantin, Emmanuel Dubois, and Emmanuel Chaput. An analysis of NDN Congestion Control challenges. working paper or preprint, https://hal.archives-ouvertes.fr/hal-02314169, October 2019.

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