INFORM: a dynamic INterest FORwarding Mechanism for Information Centric Networking

Raffaele Chiocchetti, Diego Perino, Giovanna Carofiglio
Alcatel Lucent Bell Labs, Nozay, France
first.last@alcatel-lucent.com

Dario Rossi, Giuseppe Rossini
Telecom ParisTech, Paris, France
first.last@telecom-paristech.fr

ABSTRACT

Information Centric Networking is a new communication paradigm where network primitives are based on named-data rather than host identifiers. In ICN, data retrieval is triggered by user requests which are forwarded towards a copy of the desired content item. Data can be retrieved either from a server that permanently provides a content item, or from a temporary item copy opportunistically cached by an in-network node. As the availability of cached items dynamically varies over time, the request forwarding scheme should be adapted accordingly. In this paper we focus on dynamic request forwarding in ICN, and develop an approach, inspired by Q-routing framework, that we show to outperform algorithms currently available in the state of the art.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network communications, Packet-switching networks

General Terms

Algorithms, Design, Performance

Keywords

Information Centric Networking, Caching, Forwarding

1. INTRODUCTION

Recognizing that end users are often more interested in obtaining *content*, rather than merely being provided with *connectivity* among two addressable entities, a number of Information Centric Network (ICN) architectures (overviewed in [2]) have been proposed. While these proposals differ in a number of aspects (e.g., the way content is named, content resolution is addressed, etc.), they all provide name-based network layer primitives. In addition to current Internet functionalities, as packet forwarding or routing, name-based identifiers enable enhanced features which are natively supported by all proposals, as distributed in-network caching.

In ICN the data delivery process is typically pull-based, i.e. triggered by user requests which are forwarded towards a copy of the requested item. Request forwarding is driven by forwarding engines (e.g. FIBs) that are populated with reachability information about different content items. In presence of such a highly distributed caching infrastructure, item availability and location can vary over time because of temporary replicas spread across the network. Replica distribution is determined by several factors, as content popularity or caching policy, and the request forwarding scheme should be adapted accordingly.

As also reflected in the ICN literature [4, 8–11], two coexisting approaches can be exploited to adapt request forwarding: (i) on a long term, *control plane protocols* [9, 10] distribute item availability information across the network to reach permanent or long term item replicas; (ii) on a short term, *data plane forwarding schemes* [4,8,11] based on local information available at every node allows to quickly react to dynamic item availability.

This work adheres to the second family of approaches [4, 8, 11], proposing to complement NDN with a dynamic INterest FORwarding Mechanism. INFORM is an adaptive hop-by-hop forwarding algorithm that discovers routes towards temporary item replicas through exploration in the data plane, that can be exploited later on for subsequent requests for the same objects. This work yields several contributions beyond [4,11], notably: (i) we propose a distributed on-line request forwarding algorithm based on Q-routing, whereas [4] limitedly addressed simple heuristics; (ii) our solution is compared min-delay path forwarding [6] and the strategy layer proposed by the NDN project [11].

2. ICN BACKGROUND

As above introduced, previous research on ICN request forwarding progresses along two directions.

On the one hand, the definition of *routing protocols* in the control plane [9, 10], for the dissemination of FIB information addressing permanent and possibly temporary copies, when stable over time. Concerning permanent copies, some work focuses on the design of routing protocols supporting advertisement of name prefixes rather than IP address ranges. Yet, work in this area is still at preliminary stage and mostly limited to AS-level domain (e.g., OSPFn [9]), while the corresponding name-oriented protocol for inter-domain routing (e.g., BGPn) is still to appear. The addressing of

ICN'13, August 12, 2013, Hong Kong, China.

ACM 978-1-4503-2179-2/13/08 ...$15.00.

temporary replicas is realized in [10] through the encoding of cache content in compact bloom filters, periodically exchanged among neighboring caches. However, the scalability of such approach has not been assessed.

On the other hand, we have work focusing on data plane *forwarding strategies* [4,8,11] for the retrieval of more volatile temporary copies according to a predefined metric (e.g. the closest). The benefits in this case are the possibility to (i) efficiently locate temporary replicas without incurring the overhead of explicitly signaling (ii) tune the forwarding according to the instantaneous network state, which is simply impossible in case of proactive routing approaches.

Our preliminary work [4,8] explored two cases where we either (i) assume FIB knowledge about multiple paths in the network leading toward permanent copies that can be directly exploited by the forwarding strategy [8]; or (ii) avoid to require any FIB knowledge (but can use if available) and rather perform a limited exploration of the network [4]. In particular [4] testified the potential of using the first few chunks of an object to explore the network in search for local temporary copies. Lastly, [11] introduces a dynamic approach in the NDN framework. Specifically, interfaces are periodically probed, gathering statistics for each of them: if, for a given content, an interface is estimated to be "better" than the currently exploited one, the forwarding plane switches to that interface.

While our proposal can be exploited by possibly several ICN architectures, we focus on the CCN/NDN approach initially proposed in [6]. Let us briefly introduce it here, using the CCN terminology, since a conceptual unifying framework is currently missing.

CCN clients request Data in a pull-based fashion sending *Interests* for named contents. Request are *forwarded* hop-by-hop toward a permanent copy of the requested Data: for each Interest, CCN nodes perform lookup for content names in a *Forwarding Information Base (FIB)*, that stores the set of interfaces through which any given content can be reached. As multiple paths are possibly stored for any given name, a *Strategy Layer* is responsible for the selection of one (or more) next hop interfaces among the set of possible ones. CCN nodes along this path may possess cached copies of the content of interest within their own *Content Store*: in this case, Interests do not need to reach the permanent copy stored at the repository, and the temporary copy in cache is directly sent back to the client along the reverse path. Indeed, Data travels back toward the requester following a trail of bread-crumbs, that are stored in a *Pending Interest Table (PIT)* at every network node.

3. HOP-BY-HOP DYNAMIC REQUEST FORWARDING

In this section, we describe the design of INFORM, our proposal for hop-by-hop dynamic request forwarding on the data plane. The goal of INFORM is twofold: (i) to discover paths to temporary copies of a content item, not addressed in routing tables and (ii) to forward requests for such content item towards the 'best' performing interface (according to a specified metric), while guaranteeing continuous Data delivery and limiting the network overhead.

3.1 A Reinforcement Learning framework

Our proposal for on-line dynamic request forwarding leverages previous work on reinforcement learning approaches to routing and extend them to the case of a cache network operating under CCN. Specifically, INFORM is inspired by the Q-routing algorithm [3], implementing a distributed version of Q-learning (cfr. [7] for a survey).

In the Q-routing algorithm each node builds its routing table learning the delivery times towards other nodes. This is achieved by means of a set of Q values stored by every node i for all possible destinations d, $Q_i(d,v) \forall v \in neighbours(i), d \in destinations$, where $Q_i(d,v)$ represents the delivery time of a packet from node i to node d if the packet is forwarded via node v.

The forwarding *action* taken by node i consists in selecting the interface to the neighboring node v towards a given destination d with the smallest Q value, $Q_i(d,v)$ (i.e. smallest delivery time). For every Data packet forwarded back through the neighboring node v, node i receives in response the best delivery time estimate of v, i.e. $\min_{k \in neigbours(v)} Q_v(d,k)$. The associated Q value at i is then updated as follows, $Q_i(d,v) = (1-\eta)Q_i(d,v) + \eta(\min_{k \in neigbours(v)} Q_v(f,k) + rtt_{i,v})$, where $rtt_{i,v}$ denotes the round trip delay between nodes i and v.

The Q-routing approach has been shown effective to improve performance with respect to basic shortest path forwarding [3] in presence of dynamic network conditions, and several improvements have been proposed to the original design in order to apply it to various network contexts. INFORM realizes distributed reinforcement learning at each network node with reward information exchange via Data piggyback.

INFORM is independently run by each node in the network, and works at content item granularity, i.e. at the granularity of a file. For a given file $f \in F$ a node i maintains a set of $Q_i(f,v) \forall v \in \mathcal{I}(i)$ values, where $\mathcal{I}(i)$ denotes the set of interfaces of node i[1]. Those values are computed and updated during an *exploration* phase, where a node probes the interfaces in order to learn the cost (reward) in terms of residual delay to the first hitting cache for file f associated with each of them. As explained before, the Q values update exploits the knowledge of the smallest Q value of the neighbors chosen for forwarding a given Interest, which is piggy-backed in the returning Data packet. Q values are then used during an *exploitation* phase to identify the best available interface where to forward Interest packets.

In the following, we detail the different phases of our algorithm as well as the operations performed upon reception of Interest and Data packets.

3.2 Exploration and exploitation phases

Figure 1 summarizes the transition among different phases and the behavior of our algorithm in each phase. The algorithm starts by initializing the set of $Q_i(f,v)$, $\forall v \in \mathcal{I}(i)$ values, when an Interest for a given Data packet of file f is received and corresponding Q values are not available.

A first exploration phase then starts. The goal of such initial exploration phase is to compute Q values for the different interfaces, while guaranteeing the delivery of the requested

[1]Remark that Q values are associated to a given content item but not to a single content download. Specifically, they associated to Interest/Data packets belonging to multiple (parallel or subsequent) downloads of the same content item.

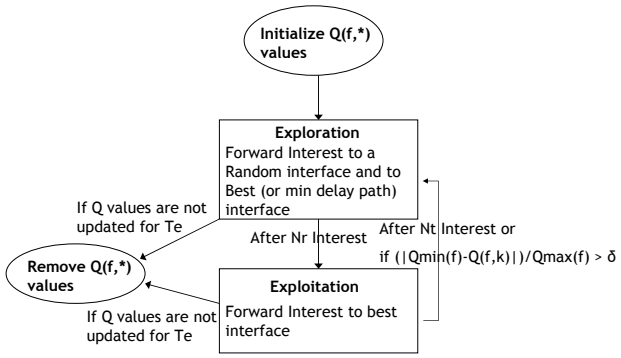


Figure 1: Transition between phases

Data packets. To this end, a node randomly selects an interface to forward an incoming Interest, and, at the same time, it forwards the Interest over the shortest path (in terms of delay) towards a permanent copy of the file.

Observation. Interest forwarding over shortest path is required as random interface selection alone does not guarantee that the requested Data packet is delivered. Indeed, we assume that an Interest contains a TTL value that is decremented at every hop, and the Interest random walk in the network may end without encountering any matching Data. Only one random interface per Interest is chosen in order to limit the overhead of this probing phase.

The exploration phase lasts N_r chunks, after which the interface k providing the minimum delay is identified, i.e. $k : Q(f,k) = \min_{v \in \mathcal{I}(i)} Q(f,v)$, and its Q value is stored, i.e. $Q_{min}(f) = Q(f,k)$.

After the exploration phase, the *exploitation* phase starts. The goal of this phase is to exploit the information about rewards associated with each interface collected during the exploration phase. Thus, an Interest is forwarded over the best interface k only; remark that the corresponding $Q(f,k)$ is the only one being updated during this phase. The algorithm remains in exploitation phase until $\frac{|Q_{min}(f)-Q(f,k)|}{Q_{min}(f)} > \delta$ or for N_t chunks at most. The first condition indicates that system state has changed and Q values have then to be updated. The second condition is also required as, despite the Q value of the best interface is not significantly changed, the state of the other interfaces may have changed.

After the exploitation phase, the algorithm returns in exploration. As previously mentioned this is required to deal with dynamic item availability, and to update Q values accordingly. Differently form the first exploration phase, during all the subsequent explorations an Interest is forwarded towards a randomly selected interface, and, at the same time, towards the previously determined best interface k rather than shortest path interface. At the end of this exploration phase, the new interface k' providing the minimum delay is identified, i.e. $k' : Q(f,k') = min_{v \in \mathcal{I}(i)} Q(f,v)$, the minimum Q value is updated, i.e. $Q_{min}(f) = Q(f,k')$, and the algorithm returns in exploitation phase.

Finally, Q values associated with a given file are deleted when they are not updated for T_e time units, i.e. no interests for file f is forwarded by the node for T_e time units.

```
if requested Data is present in the cache then
    forward Data packet through j with Q_min(f);
else
    if request is present in the PIT then
        add interface j to list of requesting interfaces;
    else
        create a new entry in the PIT;
        if Exploration phase then
            i=select random interface;
            if First exploration then
                k=select shortest path interface;
            else
                k=select best interface
                k : Q(f,k) = min_{v∈I(i)} Q(f,v);
            end
            forward Interest through i,k;
        else
            k=select best interface
            k : Q(f,k) = min_{v∈I(i)} Q(f,v);
            forward Interest through k;
        end
    end
end
```

Algorithm 1: Operations upon the reception of an Interest packet from interface j.

3.3 Interest and Data packet

Algorithm 1 details the operations performed by our algorithm upon reception of an Interest packet. If the requested Data packet is present in cache, then it is send over the interface that requested it. The minimum value $Q_{min}(f)$ is also added to the Data packet, as it will be used by downstream nodes to update their Q values. Otherwise if the request is not present in the PIT and the algorithm is in exploration phase, the Interest is forwarded over a random interface, and over the best interface (or over the interface towards the shortest path in terms of delay in case of first exploration). Finally, if the algorithm is in exploitation phase, the Interest packet is forwarded towards the best interface k only.

Algorithm 2 details the operations performed by our algorithm upon reception of a Data packet. After storing the Data packet in the cache, the Q value associated with the incoming interface of the considered file are updated. Finally, the list of requesting interfaces is looked up from the PIT, the Data packet is forwarded towards the *interested* interfaces and the PIT entry is removed.

```
cache.store(data);
Q_i(d,f).update(
    (1−η)Q_i(d,f) + η(min_{k∈I(y)} Q_y(f,k) + rtt_{i,y}));
foreach interface ∈ PIT.entry[data] do
    data.insert (Q_min(f)); //Q-value piggybacking;
    data.forward(interface);
end
erase (PIT.entry);
```

Algorithm 2: Operations upon the reception of a Data packet on interface j from neighbor y.

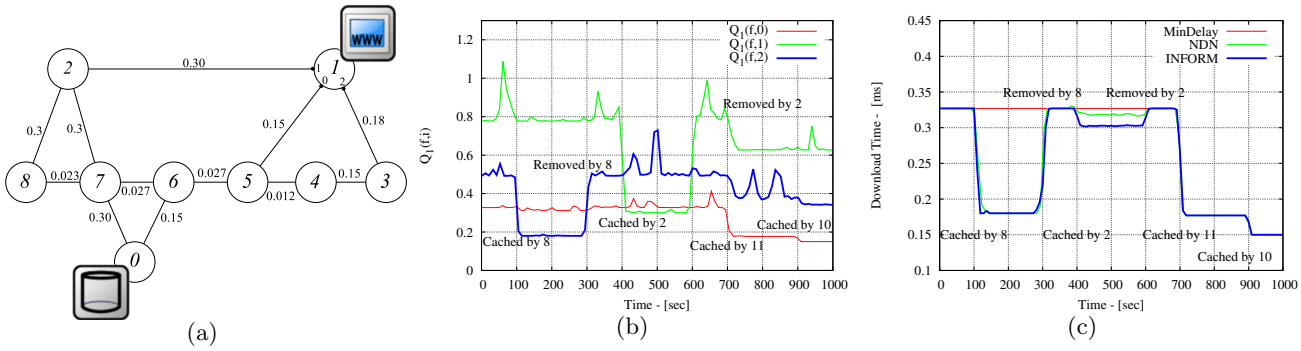


Figure 2: Toy case scenario: (a) Network topology with link delays [ms]. (b) $Q_1(\cdot,\cdot)$-values evolution over the time.(b) Data packet download time evolution over time.

3.4 Toy case example

For a better understanding of our algorithm we now present a toy case scenario. We consider the simple network topology reported in Fig. 2(a), and we assume there is only one client connected to node 1, and one server connected to node 0 . We consider a single content item composed of 100 Data packets that is repeatedly requested by the client and permanently stored by the server. We assume there are no caches in the network, and we dynamically place the content item across different nodes. The dynamic placement pattern is showed in Figg. 2(b)-2(c), where we report node 1 Q values and the Data packet delivery time evolution over time respectively.

First, we observe our algorithm is able to detect and react to dynamic item availability. Indeed, when content placement is modified, Q values associated with different interfaces are updated and the download time evolves consequently. Second, we notice the correctness of computed Q values after convergence. Indeed, converged Q values are equal to the minimum delays between node 1 and the closest item replica in terms of delay via a given interface. Third, the interface over which Interest packets are forwarded during the exploration phase, always corresponds to the optimal one under the given content item placement and network delay. As the choice of the parameters is critical for the performance of INFORM, we consider this issue in the next Section.

Finally, Fig. 2(c) shows INFORM outperforms min-delay path forwarding as it can discover off-path temporary content replicas when available, i.e. between 100 and 300s and 400 and 600s. NDN forwarding schemes [11] can also discover the temporary replica available at node 8 between 100 and 300s with a slightly longer convergence time than INFORM. Differently, the temporary replica at node 2 available between 400 and 600s is discovered but is not exploited to download all chunks, i.e. some packets are downloaded from node 2 while other from the server. In Sec. 4 we deeply compare the performance of the three algorithms under different simulation settings.

4. EVALUATION

We evaluate the performance of INFORM by means of packet-level simulations. For our analysis we extend the ccnSim simulator [1] to support INFORM, and the dynamic forwarding policy proposed by the NDN project [11].

We model the network topology as an Erdos-Renyi graph $G(n,\rho)$, where n is the number of nodes and ρ is the probability that a link connecting two nodes does exist. We assume b among the n nodes are border routers where users are connected to, and $s \leq n$ content servers are connected to distinct nodes of the network. We further assume every node is equipped with a cache of size $c\%$ of the content catalog and implements the Least Recently Used (LRU) replacement policy. Unless otherwise specified we assume $n = 22$, $b = 8$, $s = 1$, $\rho = 0.3$, and $c = 15\%$.

The placement of border routers and servers are randomly generated, results are averaged over multiple simulation runs, and we do not consider the cache warm up period. Users generate content requests according to a Poisson process of intensity $\lambda = 1$ req/s per border router. The motivation behind the Poisson assumption comes from the observation that Internet traffic is well modeled at session level by a Poisson process [5].We consider a catalog of 10^5 content items whose popularity is Zipf distributed with $\alpha = 1$. We assume each content item is composed of 100 independent Data packets that are permanently stored at servers s. Finally, we populate nodes' FIB with next hop information for the min-delay path towards one of the permanent content item copy.

4.1 Parameter tuning

In this section, we investigate the impact of INFORM parameters on its performance. We expect convergence time and stability to be primarily affected by the learning rate η, which determines the speed at which INFORM adjusts the forwarding policy to dynamic item availability. Similarly, accuracy of the best interface estimation is primarily affected by the duration of the exploration phase N_r. Results in the following are averaged over multiple nodes and simulation runs.

Fig. 3(a) shows the convergence time of INFORM Q values as a function of the learning rate for three objects having different popularity. On the one hand, we observe that convergence time decreases as the learning rate increases and is independent from file popularity. This stems from the fact that the weight of the last delay estimation increases with the learning rate, resulting in a faster adaptation of the forwarding policy to item availability.

On the other hand, a high learning rate leads also to undesirable oscillations in delay estimations. Fig. 3(b) shows the standard deviation of Q values after convergence increases

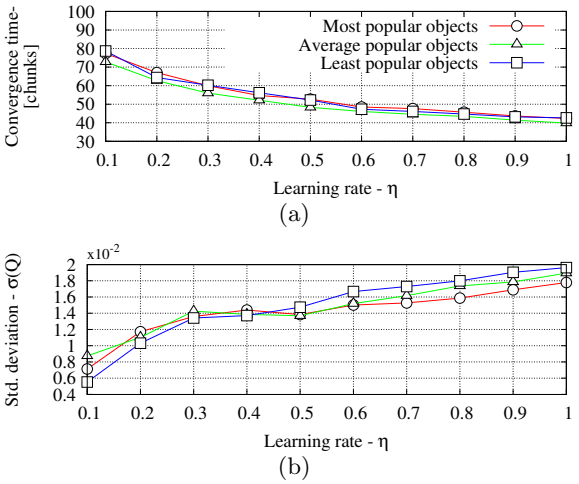


Figure 3: Parameter tuning. Convergence time as a function of the learning rate (a); Q value standard deviation as a function of the learning rate (b).

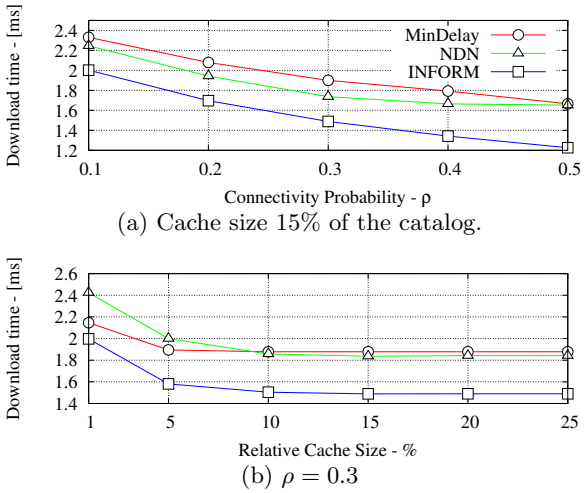


Figure 4: Mean Download time as a function of the network connectivity (a) and cache size (b).

with the learning rate and is independent from file popularity. As standard deviation captures the stability of delay estimations, we observe INFORM stability decreases as the learning rate increases. Nevertheless, the maximum standard deviation is 2%, that we argue to have a limited effect on the interface selection process.

4.2 Performance comparison

We now evaluate the performance of INFORM and compare it with simple min-delay path forwarding and the dynamic forwarding scheme proposed by the NDN project [11]. We consider three main metrics: i) the *Data packet delivery time*, which represents the time elapsing between a client expression of a Interest for a given packet, and the reception of the corresponding Data packet; ii) the *Data load*, defined as the average number of Data packets flowing through the network in one time unit; iii) the *Interest load*, defined as the average number of Interest packets flowing through the network in one time unit. The first metric allows us to quantify the performance as perceived by the end-users, while the second and third metrics quantify the network traffic cost.

We set the learning rate to $\eta = 0.7$, the duration of the exploration phase to $N_r = 50$ chunks, and the duration of the exploitation phase to $N_r = 100$ chunks. For the NDN forwarding algorithm parameters, we performed several simulations with different settings, and we present the best obtained results.

Fig. 4(a) reports the average Data packet delivery time as a function of the network connectivity (i.e., probability ρ that any two nodes are connected), which determines the number of available paths between clients and servers. Clearly, delivery time decreases as the network connectivity increases: as the number of links in the network increases, the distance between clients and servers is reduced, with a consequence decrease in the delivery time.

We also observe INFORM provides the smallest delivery time among the three algorithms for all connectivity values. Specifically, it provides a performance improvement between 18-33% with respect to simple min-delay path forwarding, and between 10-33% with respect to the NDN forwarding strategy. The performance gap increases with the connectivity, testifying that INFORM can better exploit an increasing number of paths.

Fig. 4(b) shows the average Data packet delivery time as function of the cache size. We observe the delivery time sharply decreases as the cache size increases until additional storage does not provide any additional benefits. We also observe INFORM outperforms other algorithms for all cache sizes providing an improvement of 22-25% with respect to the NDN forwarding strategy and 5-26% with respect to min-delay path forwarding.

Figg. 5(a)-(b) report the Data load as a function of the network connectivity and cache size respectively. INFORM generates less Data packets than the NDN forwarding scheme, but clearly more than min-delay path forwarding.

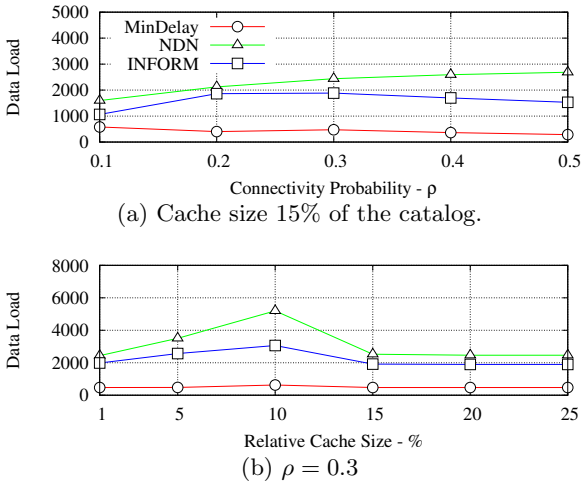
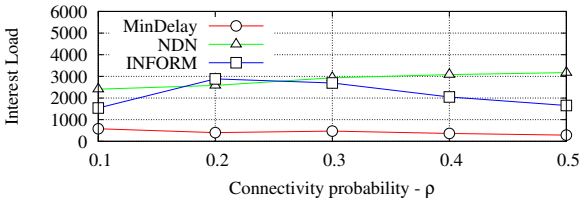
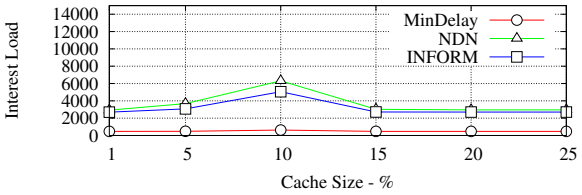


Figure 5: Data load as function of the network connectivity (a) and cache size (b).



(a) Cache size 15% of the catalog.



(b) $\rho = 0.3$

Figure 6: Interest load as function of the network connectivity (a) and cache size (b).

As the network connectivity or cache size increases the Data load decreases for min-delay path forwarding. Differently, for INFORM and NDN, the Data load first increases and then decreases as the connectivity and cache size increase. Additional links in the network or storage capacity increase the availability of temporary item replicas that are discovered by such algorithms: it follows requests are forwarded through longer paths to reach those replicas resulting in a higher Data Load. After a certain threshold, the load decreases as more links or additional storage capacity do not increase item availability but only shorten the distance between clients and item copies thus reducing the load on the network.

We observe a similar trend in Figg. 6(a)-(b,) where the Interest load as a function of the network connectivity and cache size is reported. The Interest Load has a less significant impact on the global network traffic than Data Load, as Interest packets are much smaller in size than Data packets. Nevertheless, the amount of Interest packet may have a significant impact on processing and memory cost at network routers. Indeed, Interests require routers to perform lookup operations and generate additional state to be stored.

5. CONCLUSION

In this paper, we have presented INFORM a dynamic INterest FORwarding Mechanism for ICN. INFORM is designed to discover temporary copies of content items not addressed in routing tables and to forward requests over `best' performing interface at every hop. We have shown INFORM is able to detect and react to dynamic item availability, and to forward requests towards the `best' available copy. By means of simulations we have shown INFORM outperforms simple min-delay forwarding, and state of the art NDN dynamic forwarding scheme. Our current and future work include an analytical modeling of INFORM, and a practical system design to enable Interest forwarding with INFORM at high speed.

Acknowledgements

This work presented in this paper has been partially carried out at LINCS (`http://www.lincs.fr`), and partially funded by the French national research agency (ANR), CONNECT project, under grant number ANR-10-VERS-001.

6. REFERENCES

[1] ccnsim homepage. `http://www.telecom-paristech.fr/~drossi/ccnSim`.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26 –36, 2012.

[3] J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems 6*, pages 671–678. Morgan Kaufmann, 1994.

[4] R. Chiocchetti, D. Rossi, G. Rossini, G. Carofiglio, and D. Perino. Exploit the known or explore the unknown?: hamlet-like doubts in icn. In *ACM SIGCOMM ICN*, 2012.

[5] Edward Chlebus and Jordy Brazier. Nonstationary poisson modeling of web browsing session arrivals. *Information Processing Letters*, 102(5):187 – 190, 2007.

[6] V. Jacobson, D. Smetters, J. Thornton, M. F. Plass, N. Briggs, and R. Braynard. Networking named content. In *ACM CoNEXT*, 2009.

[7] Littman M. Moore A.W. Pack Kaelbling, L. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[8] G. Rossini and D. Rossi. Evaluating ccn multi-path interest forwarding strategies. *Computer Communications*, 2013.

[9] L. Wang, A. K. M. Hoque, Cheng Yi, A. Alyyan, and B. Zhang. OSPFN: An OSPF based routing protocol for named data networking. Technical report, March 2012.

[10] Y. Wang and K. Lee. Advertising cached contents in the control plane: Necessity and feasibility. In *IEEE INFOCOM, NOMEN Workshop*, 2012.

[11] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang. A case for stateful forwarding plane. *Computer Communications*, 2013.