COBRA: Lean Intra-domain Routing in NDN

Michele Tortelli*, Luigi Alfredo Grieco*, Gennaro Boggia* and Kostas Pentikousis†
*DEI, Politecnico di Bari, via Orabona 4, 70125 Bari, Italy,
{m.tortelli,a.grieco,g.boggia}@poliba.it.
†EICT GmbH, EUREF-Campus Haus 13, Torgauer Strasse 12-15, 10829 Berlin, Germany,
{k.pentikousis}@eict.de.

***Abstract*—Named Data Networking (NDN) is an emerging Information Centric Networking architecture based on hierarchical content names, in-network caching mechanisms, receiver-driven operations, and content-level security schema. NDN networking primitives and routing are based on content names and therefore efficient content discovery of permanent as well as temporarily available cached copies is a key problem to address. This paper examines current NDN approaches and proposes a fully distributed, content-driven, bloom filter-based intra-domain routing algorithm (COBRA), which outperforms previous solutions in this area. COBRA creates routes based on paths used previously for content retrieval, and maintains routing information up-to-date without the need for extensive signaling between nodes. We evaluate COBRA using simulation and compare its performance with other established routing strategies over the European research network GEANT topology as an example of a ndnSIM core network. Our results illustrate that COBRA can significantly reduce overhead with respect to flood-based routing while ensuring hit distances of the same order as when using Dijkstra's algorithm.**

I. Introduction

The NDN architecture is an Information Centric Networking (ICN) proposal [1] introducing content-driven networking operations to enhance and/or replace the current host-centric Internet model. In NDN, networking primitives are based on hierarchical names, whereby user requests, called *Interest* packets, specify hierarchical content names rather then IP addresses. As a consequence, routing and lookup operations inside a node are based on hierarchical names too.

From an architectural point of view, three main data structures are used in a NDN node. First, the Content Store (CS), where every node can cache contents it receives. Second, the Forwarding Information Base (FIB), which maintains information about the next-hop(s) to reach known content prefixes; note that more than one interface can be associated to a single prefix entry enabling native support for multicast communications. Third, the Pending Interest Table (PIT), which keeps track of the arrival faces associated to the *Interest* packets which have been previously forwarded. This architecture allows routing operations to be performed only on *Interest* packets; indeed, contents, called *Data* packets in NDN, can simply follow back the path to the requesters by exploiting the state left by the corresponding PIT entries across the network. Such a stateful forwarding plane [2] permits the aggregation of *Interest* packets and the identification of potential loops.

In NDN, due to the in-network caching capability, a name-based routing protocol should be able to both diffuse information about permanently stored named data and discover temporary copies that may be cached outside the ordinary paths towards original content providers. The distinction between these two tasks is evident in the literature related to name-based routing in NDN. For example, the Named-data Link State Routing Protocol (NLSR) [3] explicitly targets a design which uses only *Interest/Data* packets to populate NDN node FIBs with routes towards permanent content copies. On the other hand, other works (e.g., [4]–[7]) focus on how to efficiently forward content requests towards temporary cached copies by exploiting solely local information. Among these works, local information about both the presence of alternative copies and the "cost" to retrieve them, are gathered through different mechanisms. Chiocchetti et al., for example, present a distributed algorithm where each node calculates and maintains a per-content, per-interface Q-value, estimated using reinforcement learning and exploiting the fact that each upstream node, when returning the requested content to downstream nodes, can piggyback its estimation about the considered content. The authors in [4], [6], [7], instead, encourage the use of Bloom filters as a data structure to represent FIBs in a succinct form and to exchange information about content availability. Despite the difference in implementation details, Bloom filter-based updates are exchanged and aggregated inside nodes. We point out that this procedure could incur high signaling overhead due to the high refresh rates of the caches; unfortunately, this aspect has not been thoroughly investigated earlier. In addition, this line of work assumes an IP-based routing protocol as a primary or fall-back mechanism to the pure name-based routing one, or to be used to disseminate information about the location of permanent copies. This goes in contrast with one of the expected goals of NDN, which is limiting or eliminating, if possible, the dependency on IP-based communication models, [3]. Table I summarizes the main features of Bloom Filter-based proposals for NDN routing. Inspired from the work in [8], we introduce in this paper a new intra-domain routing algorithm for NDN, namely COntent-driven Bloom filter based Routing Algorithm (COBRA). In particular, we adopt the concept of leaving traces about retrieved contents along the downstream paths towards their requesters; nevertheless, the implementation details, as well as the complexity of the proposed algorithm, are

TABLE I
BLOOM FILTER-BASED PROPOSALS FOR NAME-BASED ROUTING IN NDN

	IP Routing	Bloom Filter Type	Overhead	Performance Evaluation	Targeted ICN Architecture
Wang et al., [4]	Fall-back	Simple	Yes	None	General
Lee et al., [6]	Primary	Simple	Yes	Partial	General
Liu et al., [7]	Primary	Simple	Yes	None	General
COBRA	None	Stable Bloom Filter	No	Yes	NDN

very different from the ones which characterize the work in [8]. Indeed, in [8], information about retrieved contents are stored as tuples, containing multiple fields, indexed using content IDs; furthermore, timers are used to invalidate stale tuples, and best paths towards known seed copies are supposed to be already present in the forwarding information base of the nodes. In COBRA, instead, we propose to equip a NDN node with as many Stable Bloom Filters (SBFs) [9] as its interfaces; in this manner, the names of the retrieved contents can be hashed and inserted in the SBF associated to the incoming interface, thus leaving trace of this event in each of the nodes along the paths created by the forwarded *Interest* packets. This choice gives COBRA the characteristics of:

- *Simplicity*: COBRA does not introduce any other additional data structure or packet processing operation which could increase the computational overhead at a NDN node.
- *Efficiency*: COBRA is fully distributed, and does not require any update message exchanges between nodes in order to maintain the consistency of the collected information; indeed, it is locally guaranteed by the combination of the SBFs used in COBRA, and by an ad-hoc retransmission handling mechanism. In addition, COBRA fully exploits the benefits of using hierarchical names in NDN.
- *Autonomy*: COBRA does not require an IP-based routing protocol that acts as a fall-back mechanism or that is responsible to announce routes towards permanent content copies.

We assess COBRA performance using *ndnSIM* [10] and compare it with that of well-known routing strategies, such as flooding and Dijkstra's algorithm in scenarios based on the GEANT topology (http://www.geant.net/) using a content name catalog based on real-world traces [11]. To the best of our knowledge, this is the first performance evaluation using a discrete time simulator that embraces all facets of NDN and Bloom filters. Different simulation scenarios are considered and analyzed by varying the content popularity distribution, which is supposed to be Zipf-like.

Our results, presented in Sec. III, show that COBRA can dramatically reduce network overhead when compared to flooding-based routing, while ensuring virtually the same hit distance as the Dijkstra algorithm.

The remainder of this paper is organized as follows. Section II introduces the COBRA system design and Section III presents and discusses our performance evaluation results. Finally, Section IV concludes this paper and outlines future work items.

II. SYSTEM DESIGN

As mentioned in Sec. I, the COBRA operations are related to the use of Stable Bloom Filter (SBF) inside NDN nodes. For the sake of clarity, the key features of Bloom filters are briefly reported in the following. A Bloom Filter [12] is a probabilistic data structure conceived to efficiently perform membership queries on a large data set. It is a m-bits long vector, where, by using a group of k independent hash functions, the IDs of the elements in the set are mapped into. The mapping phase of an element x, for example, consists in setting to 1 all the bits at positions $h_1(x), h_2(x), \ldots, h_k(x)$, where $h_i(x)$, with $i = 1, \ldots, k$, is the i-th hash function of the element x. Doing this, a "footprint" is assigned to every element in the set. Then, to check the membership of an unknown element y, its footprint is computed by using the same hash functions used in the mapping phase. If even a single bit is found to be 0, y is claimed to be not in the set with a 100% probability; otherwise, if all the k bits are found to be 1, y is said to be in the set S, although with a fixed false positive probability p_{fp}. In fact, the probability of a false positive is the main price to pay using this space-efficient probabilistic data structure. This probability is heavily influenced by the length of the filter, and more precisely by the number of bits per element, m/n, where n is the cardinality of the set: the more the bits per element, the less the collisions in the mapping phase [12].

The SBF [9] used in COBRA is a modified version of the classical Bloom filter. In particular, it is a counting Bloom filter, where every cell is composed by d bits, thus obtaining a counter; in this way, both insertions and deletions can be managed. At every insertion, P cells are selected at random, and their counters are decremented by one; then, the counters of the k cells associated with the footprint of the item are set to their maximum value, $M_{ax} = 2^d - 1$. The coexistence of these two actions at every insertion (i.e., decrement P random cells and set k cells to M_{ax}), is very important in the design of COBRA, and it is one of the main reasons that brought us use SBFs instead of other types of Bloom filter. Indeed, this property guarantees that the proportion between 0's and 1's is kept constant, thus making the SBF settled and able to reinforce only traces of effective contents that are retrievable through the

associated interfaces. The other advantage of a SBF is that, in the presence of a limited storage space, it is able to guarantee a lower false positive probability than the one provided by other Bloom filters [9]. In COBRA, the classical structure of a FIB is then replaced by the SBFs, which means that each network interface has its own SBF.

A. Protocol Description

COBRA operations are articulated in the following phases:

- *content-driven* construction of SBF-based FIBs;
- *interface ranking* based on Longest Prefix Match (LPM);
- *retransmission handling*, through which stale information are erased from SBFs and new routes are discovered;
- handling of both *link failure* and *link recovery*.

In what follows, the characteristics of each phase are described in detail, providing also practical examples that ease the explanation.

1) Content-driven construction of FIBs: when a *Data* packet is received, it is firstly forwarded back towards the requesting clients; then, the relative event is recorded in SBF of the incoming interface by hashing the entire name, as well as the prefixes obtained by progressively eliminating the last component of the name. That is, if the name of the retrieved content is */A/B/C/D*, then a footprint for */A/B/C/D*, */A/B/C*, */A/B*, and */A* is inserted in the SBF of the incoming interface. Doing this, the created path can be used to forward *Interest* packets for successive chunks or for other contents that share a prefix with the retrieved content, thus limiting flooded *Interest* packets. However, when no match is found in the SBFs of a node (e.g., during the network setup when requesting the first chunk of new contents), the received *Interest* will be sent in flooding. Because of scalability problems that could arise due to this characteristic, we conceive COBRA as an intra-domain protocol, leaving its inter-domain version as a future deployment. It is worth to remark that, in COBRA, the exploration of alternative paths through different interfaces as well as the ageing of the created routes are triggered by two events: the reception of a *Data* packet, which produces the decrement of p random cells of the SBF of the incoming interface, thus weakening information about oldest routes, and the reception of a retransmission.

A particular example is now given, taking the topology shown in Fig. 1 as a reference. Suppose that none of the nodes have information about content locations and that *Client*_X wants to fetch the content */CNN/US/News/Sport/Olympics.avi/s_0*, which is stored in *Repo_A*. At router *r*_0, upon the reception of the *Interest* sent by *Client*_X, a LPM is performed on the expressed name in order to find available information in its SBFs. In this particular case, where none of the nodes have valuable information inside their SBFs, the *Interest /CNN/US/News/Sport/Olympics.avi/s_0* is initially flooded throughout the network upon reaching repository *Repo_A*. Depending on the diffusion patterns of the *Interest* message, multiple copies of the requested content could be delivered towards *Client_X*, thus leaving, in the respective SBFs, as many traces of this event (i.e., the successful retrieval of the requested content) as the number of multiple paths to reach the seed copy. This operation is repeated in each node along the reverse path(s) towards *Client_X*. One possible scenario is represented by the case where information about the presence of */CNN/US/News/Sport/Olympics.avi/s_0* is left in interface 3 of *r_5*, interface 1 of *r_2*, interface 1 of *r_1* and interface 1 of *r_0* (i.e., the best path considering the number of traversed hops).

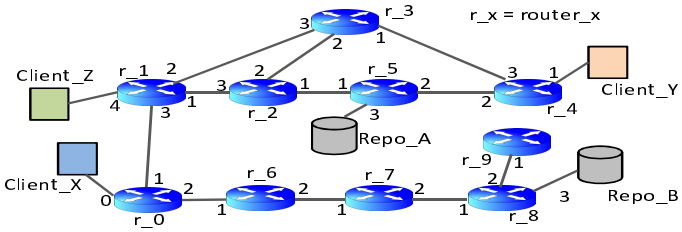


Fig. 1. Sample NDN Topology.

2) Interface Ranking: The collected information stored in the SBFs allow COBRA to perform an interface ranking at each lookup. The aim is to avoid flooding operations as much as possible, thus reducing the communication overhead. The routing cost used to rank the interfaces is correlated to the number of name components that a SBF provides a match for. In particular, when a node receives a new *Interest*, it firstly hashes the entire name of the content and it verifies if the related footprint is present in one of the SBFs associated to its interfaces. If so, the lowest routing cost is assigned to those interfaces. If all interfaces are ranked, the lookup will stop; otherwise, a new prefix is obtained by eliminating the last component of the received name, and its footprint is looked up in the remaining interfaces. The process stops either when all interfaces are ranked or when the number of components of the resulting prefix equals zero. In the latter case, if there are still some non-ranked interfaces, the highest routing cost is assigned to them. At the end of the lookup process, having created a FIB entry for the requested content containing all the interfaces with the respective routing costs, the *Interest* will be forwarded only towards the interface(s) with the lowest routing cost.

Referring to the case described above, once the trace for the first chunk */CNN/US/News/Sport/Olympics.avi/s_0* is created, *Client_X* can fetch the remaining chunks by exploiting the created path. In fact, the lookup operations for successive chunks, for example at *r_0*, will assign to interface 1 the lowest routing cost due to the match, in its SBF, with the name of the content (i.e., excluding the chunk part). Consequently, *Interest* packets will be forwarded only through it. The same considerations are valid for *r_1*, *r_2* and *r_5*.

3) Retransmission Handling: Permanent copies can be moved from one repository to another one, and temporarily cached copies can be evicted from a NDN node. As a consequence, information collected in the SBFs need to be updated in order to limit wrong forwarding decisions caused by stale information. To do so, in addition to the SBF feature of decreasing P random cells at each insertion, COBRA implements an ad-hoc retransmission handling mechanism, described in pseudocode 1, whose scope is twofold: erase stale information and increment the number of used interfaces only when needed (i.e., when a node detects a retransmission). In particular, the reception of a retransmission is considered as a signal of a wrong forwarding decision made in the past; in that case, the relative *Interest* will be forwarded both towards the interface(s) probed before, and towards the interface(s) that is (are) just after in the global rank. Afterwards, the footprint of only the entire name of the requested content will be erased from the SBF(s) associated to the interface(s) used before. The info related to its prefixes are left in the SBF to guarantee the possibility of retrieving other contents under the same prefixes. This incremental probing permits to invalidate wrong or old routes and to explore new paths. Still referring to Fig. 1, suppose that *Client_Z* wants to retrieve the same content retrieved by *Client_X* after a certain period of time, and that, during this time, the seed copy has been moved to *Repo_B*, and the copies cached in intermediate routers have been evicted due to the Least Recently Used (LRU) replacement policy implemented at each node. If *r_1* still has the information collected previously in the SBF of interface 1, it will rank its interfaces by giving the lowest routing cost to interface 1; the *Interest* expressed from *Client_Z* will be initially forwarded only towards *r_2*, that is, a wrong decision. *Client_Z*, then, will retransmit the same *Interest*. Afterwards, when *r_1* detects the reception of a retransmission from interface 2, it will perform the following operations: it will forward the received *Interest* toward both interface 1 and the interface(s) which is(are) a step below in the global rank (i.e., the interface(s) with a slightly higher routing cost), thus orienting the retransmitted *Interest* towards the right source. Then, it will erase the stale information present in the SBF of interface 1 by putting to zero all the k cells associated with the footprint of the content */CNN/US/News/Sport/Olympics.avi/s_0*.

Pseudocode 1 Retransmission Handling

```
Input: Retransmitted Interest, PIT and FIB Entry
Initialization
    Probed_Int ← PIT.getOutgoing()                    ▷ # of Interfaces probed previously
    Actual_Cost ← max(Routing_Cost) of Probed_Int
    Interfaces ← Node Interfaces
if Probed_Int = Interfaces then
    Retransmit to all Interfaces
else
    for each interface if not probed do
        if Routing_Cost_If_i <= Actual_Cost + 1 then
            New_Interfaces ← Probed_Int + If_i         ▷ Incremental Probing
        else
            break
        end if
    end for
    Transmit to New_Interfaces
end if

Erase Stale Information

Footprint ← Hash(Interest Name)
for each Probed_Int do
    if SBF_If_i contains Footprint then
        Erase Footprint from SBF_If_i
    end if
end for
```

4) Handling of Link Failure and Link Recovery: As described below, COBRA adopts two different countermeasures when a link failure is detected and/or link recovery is accomplished. When a node detects a *failed link*, it resets the SBF associated to that interface, thus effectively stopping the forwarding of successive *Interest* packets through it. The retransmission handling mechanism will, then, allow the discovery of alternative paths, as explained in the previous subsection.

In case of a *link recovery*, instead, the nodes which are directly connected to it and which detect such event, will set to '1' the cells of the two SBFs associated with the interfaces of that link. As a consequence, those SBFs will provide a match for the entire name of all the requested contents, thus forcing the recovered link to be used as a primary path, or as an additional path to the one(s) established before the link recovery was accomplished. In this way, new or alternative paths can be discovered. As mentioned before, after a certain period of time, the mechanisms of insertion and deletion will settle the SBFs, thus leaving only traces of effective contents that are retrievable through those interfaces.

III. Performance Evaluation

We use *ndnSIM* [10] to evaluate the performance of COBRA. We introduced new ad-hoc functionalities in order to reproduce COBRA behavior and the basic operations of a SBF (i.e., setup, lookup of hierarchical names inside SBF-based FIBs, retransmission handling, and so on).

COBRA is compared with three other routing approaches: *Flooding*, where every node forwards received *Interest* packets towards all its interfaces except for incoming one; *Best-Route with Caching*, where the Dijkstra algorithm is applied to calculate best paths (in terms of the least number of hops) to reach every permanent content copy; *Best-Route without Caching*, which is the same as before, but disabling in-network caching capabilities (i.e., *Interest* packets expressed by clients are satisfied only by repositories which store the seed copies).

We adopt the GEANT network topology as the core network for our studies. At each core router, a variable number of attached clients, ranging form 3 to 6, is considered, thus obtaining a total number of nodes equal to 95; the obtained topology is, then, in line with the ones used in other works in the ICN field, as [13], in terms of type and number of nodes. The number of repositories which store permanent content copies, as well as the points of attachment to the core network, is randomly varied in each of the simulated runs. In all our tests we also simulate events of link failure and link recovery, in order to evaluate the responsiveness of COBRA in searching alternative paths. In particular, in each scenario we schedule the

failure, as well as the recovery, of particular links at precise time instants, in order to compare the behavior of all evaluated routing strategies under the same conditions.

We create a content name catalog to test COBRA based on Uniform Resource Locators (URLs) of real traces [11], thus obtaining 10^5 unique contents. Arguably, we do not aim to emulate here the content catalog corresponding to the entire Internet. Instead, in this first evaluation we aim to establish the merits and potential of COBRA. It is part of our ongoing work to further the simulation scenario with a larger content catalog. That said, we do point out that recent work (e.g., [5]) often uses this size of content catalogs. Furthermore, it is worth to remark that the limited content catalog cardinality, along with the aforementioned scalability problems that could arise when dealing with flooded *Interest* packets, represent the two main reasons that limit the field of applicability of COBRA at an intra-domain level so far. Studying its feasibility at an inter-domain scale, is one of the central goals of our future research. The maximum name length of 28 fields (as in the classical URL scheme, with field names delimited by "/") is used. In NDN, named data are divided into chunks; to reflect this characteristic, contents have a size geometrically distributed, with an average number of 100 chunks (similar to what used in [13]), that is, we have a total number of unique chunks in the order of 10^7. Given that *ndnSIM* does not currently support fragmentation, the payload of each chunk is fixed in order to not exceed the Ethernet MTU. A Zipf-like probability distribution (i.e., $P(X = i) = \frac{1/i^{\alpha}}{\sum_{j=1}^{M} 1/j^{\alpha}}$, where M is the catalog cardinality) is used to characterize the popularity of the contents. The larger α is the smaller the number of contents representing the majority of client requests. The comparison of COBRA with routing protocols mentioned above is done using different values of α in the interval [0.8, 1.4]. The cache size of each node can store up to 10^4 chunks [14].

In the performance evaluation presented herein, we use the parameter set $\{M = 10^6, p_{fp} = 0.05, d = 3\}$ to dimension the SBFs of each node. Using the known optimal formula [12], we obtain a SBF of $m = 3.3$ Mbyte and a number of hash functions equal to $k = 7$. However, this formula represents a conservative sizing rule for a SBF [9]. In fact, the rationale of a SBF is to guarantee as much as possible the desired false positive probability given a limited space; that is, when the memory space is a constraint, by finding the optimal set of parameters (e.g., d and k), the SBF is able to guarantee a lower false positive probability than the one provided by other Bloom filters given that limited space [9].

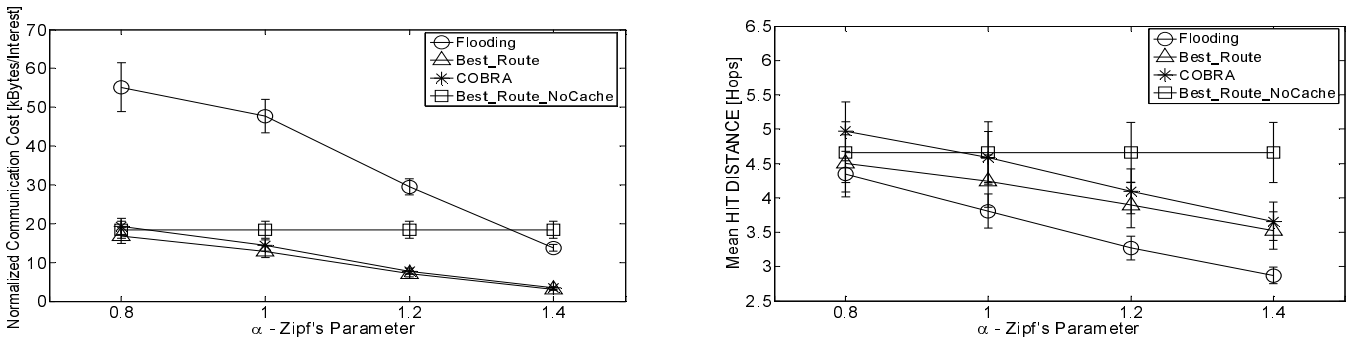
A. Results

We use the following metrics: the *Hit Distance*, i.e., the number of hops that an *Interest* has to travel to find the desired *Data*, and the *Normalized Communication Cost*, which is the traffic generated to retrieve a chunk requested by a client (considering both *Interest* and *Data* packets). Fig. 2 illustrates the results for each evaluated metric, for each routing approach and for each value of α, averaged over six runs (with each run lasting 28 hours as simulation time); the mean values are reported with their 95% confidence intervals.

Analyzing Fig. 2.(a) (illustrating the normalized communication cost), it is evident one of the advantages of COBRA. Although it is fully distributed and content-driven, it dramatically reduces communication cost w.r.t. *Flooding*, but with comparable performance w.r.t. the *Best-Route*. This represents a remarkable gain, especially since COBRA does not have any a priori information about topology or content locations, and it does not require signaling messages between nodes, as is the case with *Best-Route*. Fig. 2.(a) also shows that the overhead introduced by a *Flooding* strategy is always higher with respect to the other approaches, except for the case $\alpha = 1.4$, where it is smaller than the overhead associated to the *Best-Route without Caching*. This is because the client requests are directed to a very restricted set of popular contents, which are not likely to be evicted from the caches of the nodes. As a consequence, the forwarded *Interest* packets are satisfied within very few hops from the requesters. In addition, it is worthwhile to note that a forwarding strategy based on the knowledge of the best paths towards every content involves the use of some routing protocols to calculate and update those paths. One possible way to obtain the best paths is the use of IP routing protocols, which must be adopted as fall-back or primary mechanisms to be run in addition to name-based routing. Adapting classical IP routing techniques in a content centric network requires significant effort, and it could lead to potential problems, such as scalability issues at a world scale [3], or non-negligible signaling overhead even at the intra-domain level. This represents another advantage that characterizes COBRA with respect to *Best-Route*.

For what concerns the mean hit distance, shown in Fig. 2.(b), the closeness between the COBRA and *Best-Route* performance is obvious. In fact, it can be seen that the difference between the relative mean hit distances is no more than 0.5 hops. Furthermore, only in the case $\alpha = 0.8$, COBRA presents a path stretch slightly grater than the *Best-Route without caching*; in the remaining cases, the COBRA path stretch is always smaller.

We evaluate the performance of COBRA in the presence of topology changes; due to lack of space, herein only some representative results are reported. In particular, the temporal evolution of the hit distance is shown in Fig. 3, when $\alpha = 1$. In this case, a link failure is scheduled at 10000 s and 30000 s, whereas the corresponding recoveries are scheduled at 20000 s and 40000 s. Fig. 3 confirms the responsiveness of COBRA in the presence of link failure and link recovery. In fact, detection of a link failure resets the SBFs associated to interfaces of the failed link; thus, *Interest* packets are no longer forwarded toward those interfaces, and alternative paths are discovered via the retransmission handling mechanism. The detection of link recovery, instead, or the addition of a new link as well, is



(a) (b)

Fig. 2. Evaluated Metrics for several values of α: (a) Normalized Communication Cost; (b) Mean Hit Distance.

followed by setting all cells of the respective SBFs associated to interfaces of that link. This allows, during a limited period of time after the detection, the recovered link to be used as an additional path to forward the *Interest* packets through, because the respective SBFs provides a match for the entire name of all the requested contents in that period. After that, the mechanisms of insertion and deletion will settle the SBFs, accounting only for effective contents retrievable through that interface. This permits COBRA to react to link failure and recovery and to restore paths established before the failure, as it can be seen from the mean hit distance reported in Fig. 3.

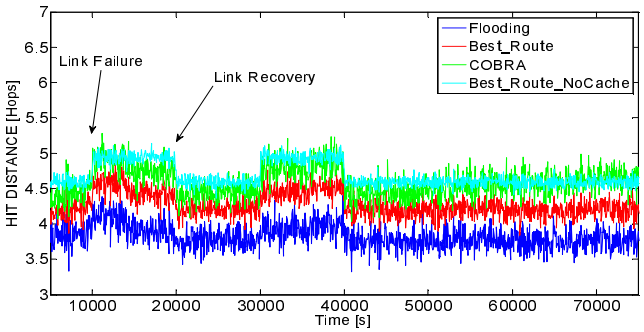


Fig. 3. Mean Hit Distance with Link Failure and Link Recovery [$\alpha = 1$].

IV. Conclusion and Future Work

In this work we proposed the COBRA algorithm as an intra-domain, content-driven, fully distributed Bloom Filter based routing protocol for NDN. It is a name-based routing protocol which aims at being as simple and efficient as possible: no signaling messages exchanged between NDN nodes; no additional data structures introduced inside nodes; no further packet processing operations required; no auxiliary routing protocols needed; the benefits of hierarchical names completely exploited. Results of our extensive comparative performance evaluation show that COBRA significantly reduces the communication cost with respect to Flooding ensuring, at the same time, about the same hit distance as the Dijkstra algorithm.

As future work we plan to expand the evaluation of COBRA by using different catalogs, topologies as well as scenarios considered in IRTF ICNRG [15], and by comparing it with other proposals for NDN name-based routing. We also plan to further improve the interface ranking algorithm by firstly quantifying its complexity and by limiting the number of *Interest* packets that are forwarded when multiple interfaces are selected.

References

[1] B. Ahlgren et al., "A survey of information-centric networking," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 26–36, Jul. 2012.

[2] C. Yi et al., *A Case for Stateful Forwarding Plane*, PARC Technical Report NDN-0002, Jul. 2012.

[3] A. Hoque et al., "NLSR: Named-data Link State Routing Protocol," in *Proc. ACM SIGCOMM ICN Workshop*, Hong Kong, China, Aug. 2013.

[4] Y. Wang et al., "Advertising cached contents in the control plane: Necessity and feasibility," in *Proc. IEEE INFOCOM WKSHPS*, Orlando, (FL), USA, Mar. 2012.

[5] D. Chiocchetti et al., "INFORM: a dynamic INterest FORwarding Mechanism for Information Centric Networking," in *Proc. ACM SIGCOMM ICN Workshop*, Hong Kong, China, Aug. 2013.

[6] M. Lee et al., "SCAN: Scalable Content Routing for Content-Aware Networking." in *Proc. IEEE ICC*, Kyoto, Japan, Jun. 2011.

[7] H. Liu et al., "A multi-level DHT routing framework with aggregation," in *Proc. ACM ICN Workshop*, Helsinki, Finland, Aug. 2012.

[8] E. J. Rosensweig and J. Kurose, "Breadcrumbs: Efficient, best-effort content location in cache networks," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009.

[9] F. Deng and D. Rafiei, "Approximately detecting duplicates for streaming data using stable bloom filters," in *Proc. of ACM SIGMOD*, Chicago (IL), USA, Jun. 2006.

[10] A. Afanasyev et al., *ndnSIM: NDN simulator for NS-3*, PARC Technical Report NDN-0005, Oct. 2012.

[11] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: evidence and possible causes," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 835–846, Dec. 1997.

[12] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, pp. 636–646, 2002.

[13] R. Chiocchetti et al., "Exploit the known or explore the unknown?: Hamlet-like doubts in ICN," in *Proc. ACM ICN Workshop*, Helsinki, Finland, Aug. 2012.

[14] D. Rossi and G. Rossini, "On sizing CCN content stores by exploiting topological information," in *Proc. IEEE NOMEN Workshop*, Orlando (FL), USA, Mar. 2012.

[15] K. Pentikousis et al., "Information-centric Networking: Baseline Scenarios," Internet Draft, Oct. 2013, Expires: April 24, 2014. [Online]. Available: http://www.ietf.org/id/draft-irtf-icnrg-scenarios-01.txt