A Comparative Analysis of Bloom Filter-based Routing Protocols for Information-Centric Networks

Ali Marandi*, Torsten Braun*, Kavé Salamatian† and Nikolaos Thomos‡
*University of Bern, Bern, Switzerland
Email:{marandi, braun@inf.unibe.ch}
†Université de Savoie, France
Email: kave.salamatian@univ-savoie.fr
‡University of Essex, Colchester, United Kingdom
Email: nthomos@essex.ac.uk

***Abstract*—Bloom filter-based routing protocols for Named Data Networking (NDN) aim at facilitating content discovery in NDN. In this paper, we compare the performance of two Bloom filter-based routing protocols, namely BFR and COBRA. BFR is a push-based routing protocol that works based on Bloom filter-based content advertisements, while COBRA is a pull-based routing protocol that operates based on route traces left from previously retrieved content objects, which are stored in Stable Bloom Filters. In this paper, we show that BFR outperforms COBRA in terms of average memory needed for storing routing updates, average round-trip delay, normalized communication overhead, total Interest communication overhead, and mean hit distance.**

I. Introduction

In Information-Centric Networking (ICN) [1], content objects are decoupled from locations in contrast to IP systems where users' requests have to be forwarded to servers' locations. The main difference between routing in IP and ICN architectures is that in IP architectures routing is location-oriented, while in ICN architectures routing is content-oriented. Content-oriented routing consists of routing users' requests according to their names. Thus, due to the importance of name-based routing, several works have studied this topic recently [2]–[5].

In IP systems, Bloom Filters (BFs) [6] have been used for different purposes, e.g., finding longest prefix matching IP addresses [7], exchanging summary caches [8], etc. Inspired by these works, researchers have proposed to use BFs for similar purposes in ICN as well [2], [4], [9]–[12]. In [9], a BF-based data structure called Prefix BF (PBF) is proposed to determine the size of longest prefix match quickly. SCAN [10] is a BF-based routing protocol for ICN, but it is not a fully content-oriented routing protocol, because it uses IP routing as a fall-back approach. SCAN uses BFs to represent the content objects' names passing through each interface. The main drawback of this approach is that the number of elements inserted to the BF associated with the interface increases with the number of content objects passing through each interface. This will result in all the BF bits to be set to 1. Thus, the BF will not represent the elements that have passed over the interface correctly. To cope with this problem, COntent-oriented intra-domain Bloom filter-based Routing Algorithm (COBRA) [4] proposed Stable Bloom Filters (SBFs) [13] to represent the content objects' names passing through each interface. SBFs need more storage overhead compared to standard BFs. However, they maintain only the names of the content objects that recently passed through each interface and discard the names of older content objects randomly. Another alternative is to use Bloom Filter-based Routing (BFR) [2], which is a routing protocol for ICN that allows origin servers to use BFs for content advertisements. In BFR, clients and routers benefit from BF-based content advertisements to populate the FIBs and to route the Interests over multiple paths towards the origin servers.

To the best of our knowledge, BFR and COBRA are the only BF-based routing protocols that have been proposed for ICN that are fully content-oriented and do not need any fall-back routing protocol as a complementary component of the routing process. BFR routes Interests according to push-based content advertisements, whereas COBRA routes Interests according to path traces left from previously retrieved content objects. COBRA is assumed to be a simple and efficient routing protocol because it does not require nodes to exchange routing information messages, which leads in reduced communication overhead [4], [14].

In this paper, we show that even if BFR requires nodes to exchange Bloom filter-based routing information, it still incurs much less communication overhead than COBRA, because in COBRA the nodes flood with Interests the network during the learning phase. Further, we show that, in COBRA, nodes need significantly more memory for storing routing information than BFR. This is problematic in case nodes have limited memory space. The experimental evaluation shows that BFR outperforms COBRA in terms of average round-trip delay and mean hit distance, among others.

The remainder of this paper is structured as follows. Sections II and III briefly describe BFR and COBRA routing protocols respectively. Then, in Section IV we analyze the performance of BFR and COBRA schemes. Finally, we conclude the paper in Section V.

II. Bloom Filter-based Routing (BFR)

In [2], we have proposed BFR as an intra-domain routing protocol for ICN. BFR allows the origin servers that provide content objects to represent and to advertise their available content objects using BFs. An advantage of using BFs for representing sets rather than regular lists is the lower complexity of query operations. The complexity of checking whether an element exists in a list that contains n elements is in the order of $O(n)$. However, to check whether an element exists in a BF, one only needs to give the element to all the k hash functions and check whether all the bits indexed by the outputs of the hash functions are set to 1. Thus, the complexity of a query operation for BFs is in the order of $O(k)$. For large sets, $k << n$, the complexity of the query operation for BFs is much less than the complexity of query operation for regular lists. When one uses BFs, false positive errors occur with probability p, but false negative errors are impossible to happen. Now, if n is the number of elements to be inserted into the BF, m is the size of the bit table, k is the number of hash functions, and p is the probability of false positive error, the values for m and k can be calculated as in [6]:

$$m = -\frac{nln(p)}{(ln2)^2}, k = \frac{m}{n}ln2 \tag{1}$$

In the first phase of BFR, servers describe their content objects using BFs. Assume that server S_A produces a content object with name $/unibe.ch/images/fileName1$ at time instance t_1. This server immediately creates an empty BF and maps the name $/unibe.ch/images/fileName1$ as well as the name prefixes of this name into the BF. Right after BF calculation, server S_A should propagate the generated BF to inform all nodes in the network about its available content objects. For this purpose, we introduce a new Interest message type called Content Advertisement Interest (CAI). Fig. 1 shows the structure of a CAI message. CAI messages have a name prefix of type $/ContentAdvertisement/publisherID$, a nonce field for loop avoidance, a lifetime field, and a field with the required information for retrieving the BF. CAI messages do not demand any content object and they are only used in order to advertise the produced content objects at servers in the network. BFR respects all the design principles of NDN. At each node, we populate the FIB for name prefix $/ContentAdvertisement$ and add all the faces as the next hop faces for this name prefix. To permit broadcasting CAI messages, we consider the *multicast* forwarding strategy for name prefix $/ContentAdvertisement$.

In Fig. 2, when server S_A encapsulates its content advertisement BF in a CAI message called CAI_A with name prefix of type $/ContentAdvertisement/S_A$, server S_A broadcasts CAI_A to its neighborhood, i.e., router R_8. When router R_8 receives this message, it inserts this message in the Pending Interest Table (PIT) and creates a new PIT entry for name prefix $/ContentAdvertisement/S_A$. Further, router R_8 updates the face ID, over which it has received CAI_A, in a field called *in-record*. Fig. 3 shows the structure of an in-record field. In NDN, when an Interest message is stored in the PIT, the incoming face ID is updated in the in-record field. When router R_8 updates the information regarding the face over which it has received message CAI_A in the corresponding in-record,

/ContentAdvertisement/publisherID	BF information	Lifetime	Nonce

Fig. 1. CAI message format

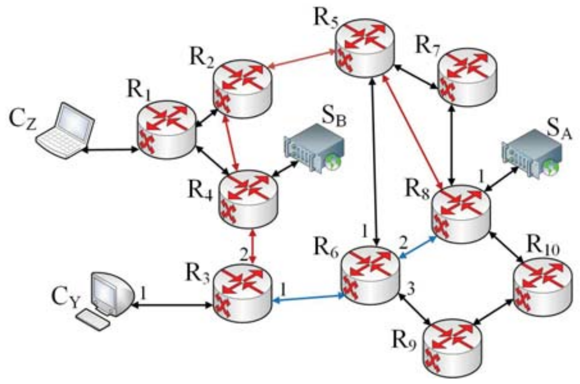

Fig. 2. A topology for describing BFR and COBRA

R_8 broadcasts this message to all the neighbors except to server S_A from which message CAI_A has been received. This broadcast process is repeated at all the other routers until all network nodes receive message CAI_A. Note that: 1) when a node receives message CAI_A, it stores this message in the PIT, and 2) since CAI messages are of Interest message type, the NDN Forwarding Daemon (NFD) [15] avoids loops and duplicates for these messages.

The same content advertisement process takes place at server S_B and message CAI_B is the CAI message propagated from this server. When all nodes receive and store messages CAI_A and CAI_B, they can route each Interest issued to demand a content object produced by server S_A or server S_B. Let us explain the Interest routing process again using Fig. 2. Assume that router R_1 receives an Interest from client C_Z demanding the first segment of a content object named $/unibe.ch/images/fileName1$, i.e., the Interest name is $/unibe.ch/images/fileName1/01$. To route this Interest at router R_1, the router first processes the Interest's name and eliminates the last name component, which is the sequence number 01 of fileName1. Then, router R_1 checks the resulting name against the BFs that exist in messages CAI_A and CAI_B. Assume that the content object named $/unibe.ch/images/fileName1$ is produced by server S_A. Thus, only the BF of CAI_A contains name $/unibe.ch/images/fileName1$. Therefore, since false negative errors do not happen with BFs, the BF of CAI_A correctly claims that it contains name $/unibe.ch/images/fileName1$. According to this positive feedback from the BF of message CAI_A, router R_1 creates a FIB entry for name $/unibe.ch/images/fileName1$ and lists all the face IDs over which message CAI_A has been received as the next hop faces for this FIB entry. Now, when the FIB is populated for name $/unibe.ch/images/fileName1$, router R_1 sends the Interest for name $/unibe.ch/images/fileName1/01$ over all the next hop faces mentioned in the FIB entry that is just populated. Note that BFR forwards each Interest over all the next hop faces benefiting from multi-path routing.

Face	Last nonce	Last renewed

Fig. 3. In-record information

As we explained before, we assume in this example that the content object with name $/unibe.ch/images/fileName1$ is produced by server S_A. When router R_4 receives the Interest for name $/unibe.ch/images/fileName1/01$ from router R_1, router R_4 checks the name $/unibe.ch/images/fileName1$ against the BF of messages CAI_A and CAI_B. Assume that the BF of CAI_B gives a false positive report. In such a case, router R_4 assumes that server S_B also provides the demanded content object and routes the Interest towards server S_B accordingly. In [2], we have shown that the impact of such false positive reports on the BFR operation is negligible. This is due to the fact that it is guaranteed that all the Interests will be routed towards the servers that provide the demanded content object, while in the worst case only a very small percentage of all the Interests (1.73% of all the Interests in the worst case) might also reach the server(s) that do not provide the demanded content object. Nevertheless, we have shown in [2] that all the Interests are satisfied in presence of false positive reports from BFs.

When a node detects a link failure, it knows that it should avoid sending Interest flows over the failed link. Therefore, with BFR a node that detects a link failure removes the corresponding face ID from the in-records of all the CAI messages that contain it. As a result, the node assumes that no BF has been received from the failed link and thus avoids sending any Interests over the failed link. Nevertheless, the node keeps a list of the CAI messages that their in-records were updated due to the link failure for the face ID associated with the failed link. When the link recovers, the node again inserts the corresponding face ID to the in-records from which this face ID has been previously removed. Thus, the node immediately uses the recovered link for forwarding Interests.

A server might move some content objects from its repository to the repository of another server. This process is called content migration. When content migration happens, clients and routers should be informed about this event to avoid wrong forwarding decisions. With BFR, the two servers that are involved in the content migration immediately propagate new CAI messages representing their new sets of available content objects. Further, these servers activate a flag called *discardOldAdverts* in the new CAI messages to inform clients and routers that the previous CAI messages received from these servers should be removed from PITs.

III. Content-driven, Bloom filter-based intra-domain Routing Algorithm (COBRA)

COBRA [4] is an intra-domain, BF-based routing protocol proposed for NDN. COBRA routes Interests according to route traces left from paths over which previous content objects have been retrieved. To store these route traces, COBRA uses SBFs. In the following, we briefly introduce SBFs and then we describe the operation of COBRA routing protocol.

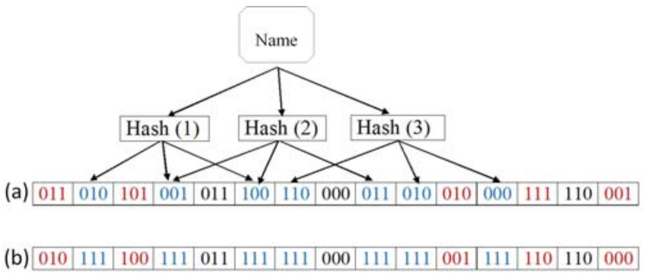
Name
Hash (1)
Hash (2)
Hash (3)
(a) 011 010 101 001 011 100 110 000 011 010 010 000 111 110 001
(b) 010 111 100 111 011 111 111 000 111 111 001 111 110 110 000

Fig. 4. SBF insertion with parameter set $\{m = 15, k = 3, d = 3, p = 5\}$

A. Stable Bloom Filter

In contrast to BF that is a table of bits, SBF is a table of counters. Assume that an SBF consists of n counters $SBF[1], .., SBF[n]$ and the length of each counter is d bits. Thus, the minimum and the maximum values for each counter are 0 and $2^d - 1$, respectively. SBF is a variant of BF, it also uses hash functions for insertion and query operations. Like in standard BF, SBF's table is also initialized by zero. However, the insertion operation for SBF differs from the insertion process of BF.

When one wants to insert an element into an SBF, it gives the element to the k hash functions and the k counters indexed by the outputs of the k hash functions are set to their maximum values ($2^d - 1$). Then, p counters are randomly selected and their values are decremented. Fig. 4 illustrates the insertion operation for an NDN name into an SBF with parameter set $\{m = 15, k = 3, d = 3, p = 5\}$. This figure also shows the states of the SBF before (i.e., Fig. 4(a)) and after (i.e., Fig. 4(b)) the insertion operation, respectively.

This insertion mechanism aims at keeping the elements that have been recently inserted into the SBF and removing elements that were previously inserted in the SBF randomly.

B. COBRA Operation

COBRA equips nodes with SBFs. Each node maintains as many SBFs as it has interfaces. When a content object travels over a path towards a client, each node located on this path receives the content object over an interface and stores the name of the content object as well as all the name prefixes of it into the SBF associated with the interface. This leads to storing a trace of the retrieval path for the content object at all nodes located on the retrieval path of the content object. In the beginning of the network operation, all SBFs of all nodes are empty because no content objects has been retrieved yet. Therefore, nodes do not have route traces stored in SBFs. Thus, nodes end up in flooding all the Interests until the SBFs learn the route traces. This phase is called *learning phase*. Flooding Interests during the learning phase does not scale well with increasing size of the content universe. In Fig. 2, assume that at time instance t_1, client C_Y issues an Interest to demand a content object segment named $/unibe.ch/images/fileName1/0$, which is stored at server S_A. When the SBFs are empty, both client C_Y and router R_3, as well as the other routers have to flood the Interest until it reaches server S_A. Then, this server sends the demanded Data packet backwards to client C_Y. Since the Interest has been flooded, the corresponding Data packet comes back over different paths (e.g., path $S_A - R_8 - R_6 - R_3 - C_Y$ (blue in Fig. 2), path $S_A - R_8 - R_5 - R_2 - R_4 - R_3 - C_Y$ (red n Fig. 2)). Therefore, for router R_8, name $/unibe.ch/images/fileName1$ and all its name prefixes are inserted into the SBF of interface 1. But for router R_6, the same name prefixes are inserted into the SBFs of interfaces 1, 2, and 3. This is because router R_8 receives the Data packet only from server S_A, while router R_6 receives the Data packet from routers R_8, R_5, and R_9. After storing the route traces for name $/unibe.ch/images/fileName1/0$ and its name prefixes, client C_Y and all the routers that stored these name prefixes in their SBFs, do not need to flood anymore the Interests that come for the subsequent segments of the same file name.

When SBFs are not empty for a name prefix, another phase of COBRA routing called *interface ranking* takes place. Let us explain this phase with the help of the topology in Fig. 2. Let us assume that at time instance t_2, when the route traces for name $/unibe.ch/images/fileName1$ are stored in SBFs, client C_Y issues Interest I_1 to demand the subsequent segment of the same file name, i.e., $/unibe.ch/images/fileName1/1$. Client C_Y checks the full name against the SBF of interface 1. The SBF of interface 1 does not contain the full name. Thus, client C_Y increases the routing cost (initialized by zero) and eliminates the last name component. Then, client C_Y checks the resulting name against the SBF of interface 1. Since the SBF of this interface contains the name, client C_Y assigns the current routing cost in the FIB to interface 1. Client C_Y does not have any more interfaces. Thus, it forwards the Interest over interface 1 towards router R_3. When router R_3 receives the Interest, it checks the full name against the SBFs of all the interfaces except the incoming one. Since the SBFs of interfaces 1 and 2 do not contain the full name, router R_3 increases the routing cost (initialized by zero) and eliminates the last name component. Then, router R_3 checks the resulting name against the SBFs of interfaces 1 and 2. Both of these SBFs contain the name. Thus, interfaces 1 and 2 are ranked with the current routing cost. The same process continues at the other routers until one of the following conditions happens: 1) all the interfaces of the router are ranked; 2) the last component of the Interest is eliminated and still one or more interfaces are not ranked. In the latter case, the router assigns the maximum routing cost to those not yet ranked interfaces.

To deal with link failures, COBRA permits nodes to reset the SBF(s) associated with a failed link upon detecting a link failure, i.e., setting all the bits of the SBF(s) to zero. Following this reset strategy, the nodes that are directly connected to the failed link know that no content object is reachable through the failed link. Thus, they avoid to forward any Interests over the failed link. When a link recovery is detected, COBRA allows nodes to set the values of all the counters of the associated SBF(s) to the maximum value. This strategy encourages the nodes that are directly connected to the recovered link to forward all Interests through the recovered link. The forwarding through the recovered link is temporarily. When new content object names are inserted into the SBFs associated with the newly recovered link, the route traces will be corrected in the SBFs.

If content migration takes place, the route traces stored in SBFs should be corrected, because the location of permanent copies of the migrated content objects has changed and the temporary cached copies might be evicted due to a caching policy, e.g, LRU. With COBRA, clients and routers are not explicitly informed about a content migration event. However, they consider Interest retransmissions as indications of wrong forwarding decisions made in the past. Thus, when a node observes an Interest retransmission event, it retransmits the Interest not only over the interface with the smallest routing cost, but also over the interface(s) with higher routing cost(s) to increase the probability that the retransmitted Interest reaches the right server.

IV. Performance analysis of BFR and COBRA

In this Section, we compare the performance of BFR and COBRA using our BFR and COBRA implementations in ndnSIM [15].

A. Simulation Settings

To evaluate the performance of BFR and COBRA, we use two topologies: 1) the GEANT network topology depicted in Fig. 5, and 2) a 10×10 grid topology depicted in Fig. 6. We use the GEANT topology for performance comparison because BFR and COBRA used this topology as an example network [2], [4]. The GEANT topology is a tree-based topology. In the GEANT topology, we attach the clients and origin servers randomly in each simulation to the GEANT core routers. We connect a different number of clients (between three to six nodes) to each randomly selected GEANT router. We randomly place five origin servers in each simulation and we have 56 clients connected to GEANT core routers. Therefore, the tested topology consists of 101 nodes. We also use in this paper the grid topology to assess the performance of both BFR and COBRA when the topology is more connected. In the grid topology, we attach the end-points, i.e., the 5 origin servers and the 25 clients randomly in each simulation to the grid routers. Thus, we obtain in total 131 nodes. We use a dataset of URLs extracted from real traces of HTTP requests [16]. We consider a content universe, i.e., the set of produced files at origin servers, that includes $100,000$ file names in total. Each file has 100 segments. Thus, we generate 10^7 unique segments. Each node has limited cache space and can store up to $10,000$ segments in its CS. The content popularity follows the Zipf-Mandelbrot law. Equation (2) shows the probability distribution for this law, where M is the cardinality of the content universe and α is the Zipf's power parameter, which is the skewness of the popularity function (the larger α is, the smaller is the cardinality of the set of popular content objects.)

$$P(x = i) = \frac{1/i^{\alpha}}{\sum_{j=1}^{M} 1/j^{\alpha}} \qquad (2)$$

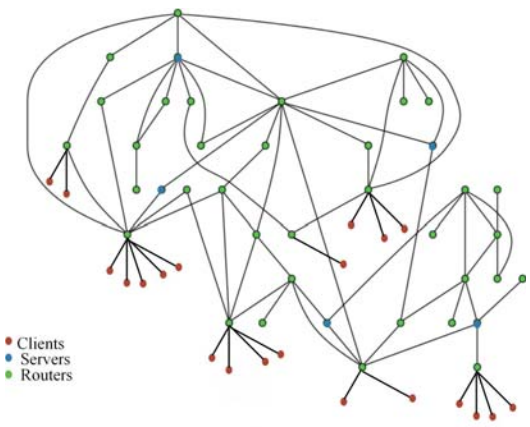

Fig. 5. Geant topology and attached clients and servers

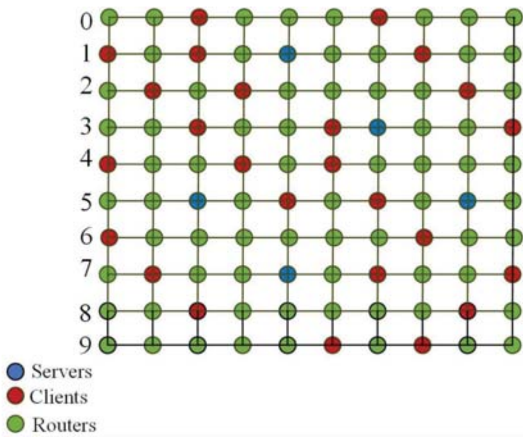

Fig. 6. Grid topology and attached clients and servers

For evaluating the performance of BFR and COBRA, we consider values of α in range $[0.8, 1.4]$. We ran ten simulations, each lasted for $100,000$ seconds. We report the average measured values over these simulations. The mean values have 95% confidence intervals. For BFs, we use the parameters $N = 1000$ and $p_{fpp} = 0.0638$, where N denotes the number of inserted elements in the BF and p_{fpp} denotes the false positive errors' probability. Therefore, the size of each advertised BF is roughly 716 bytes for advertising 1000 URLs. For SBFs, we use parameter set $\{N = 10^6, p_{fpp} = 0.0638, d = 3\}$ to create the SBFs at nodes. Therefore, we obtain SBFs of size $m = 2.05$ Mbytes with $k = 4$ hash functions.

B. Results

Below, we compare the performance of BFR and COBRA with respect to the following metrics: 1) Average memory needed for storing routing information, 2) Average round-trip delay, 3) Normalized communication overhead, 4) Total communication overhead for Interests, and 5) Mean hit distance.

1) Average memory needed for storing routing information: Table I compares BFR and COBRA in terms of the average memory space that each node needs to store routing information. In BFR, routing information consists of the content advertisement information that servers propagate. Thus, the storage overhead for routing information is not related with the structure of the topology. However, in COBRA, routing information consists of the route traces stored in the SBFs. In COBRA, each node stores as many SBFs as the number of its interfaces. Therefore, the number of SBFs is directly proportional to the number of links. Thus, the more connected the topology is, the higher is the storage overhead COBRA requires to store SBFs. To better understand this, we compare the average memory needed for storing SBFs in COBRA for GEANT and grid topologies in Table I. For COBRA, columns 2 and 3 of Table I show that in the grid topology each node needs approximately 2.36 times more memory to store SBFs than in the GEANT topology. Nevertheless, we observe a significant difference between the values of columns 2 and 4. This means that even if the GEANT topology is used, BFR needs much less memory space for storing routing information than COBRA. Therefore, when nodes have restricted memory capacity (e.g., sensors in IoT scenarios with constrained nodes), it is more appropriate to use BFR than COBRA.

TABLE I
AVERAGE MEMORY NEEDED FOR STORING ROUTING INFORMATION AT EACH NODE WITH DIFFERENT FALSE POSITIVE ERROR RATES.

p_{fpp} (%)	Average allocated memory per node $(KBytes)$		
	COBRA (GEANT)	COBRA (Grid)	BFR (Any topology)
28.30%	6235.392	14725.121	32.072
23.70%	7111.619	16783.365	36.579
16.00%	9052.344	21370.883	46.561
6.38%	13594.010	32880.641	69.921
2.29%	18655.232	44042.242	95.954

2) Average round-trip delay: Figs. 7a and 8a compare BFR and COBRA in terms of average round-trip delay, i.e., the average delay between the time when clients send Interests and the time they retrieve the demanded content objects for GEANT and grid topologies, respectively. From Figs. 7a and 8a, we observe that BFR outperforms COBRA without link failures. To further understand the performance of all the schemes, we schedule three link failures at time instants $5,000$ s, $15,000$ s, and $25,000$ s. These links recover at time instants $10,000$ s, $20,000$ s, and $30,000$ s, respectively. In presence of link failures, both BFR and COBRA routing protocols avoid sending an Interest through the path over which a link has failed. However, BFR forwards the Interest over the rest of the paths towards the server that provides the demanded content object, while with COBRA, nodes do not always benefit from all the paths towards the demanded content objects. We see in Fig. 8a a smaller impact of link failures on BFR's performance in terms of average round-trip delay than in Fig. 7a. This means that BFR is more resilient to link failures when the topology is more connected. We also examine the performance of BFR and COBRA with content migrations. We schedule a random number of content migration events (between 2 to 4 content migration events) between randomly selected servers at random time instants. We observe from Figs. 7a and 8a that with BFR, content migration events have a slight impact on the average round-trip delay. The reason is that, when a content migration happens and BFR is used, servers immediately propagate new

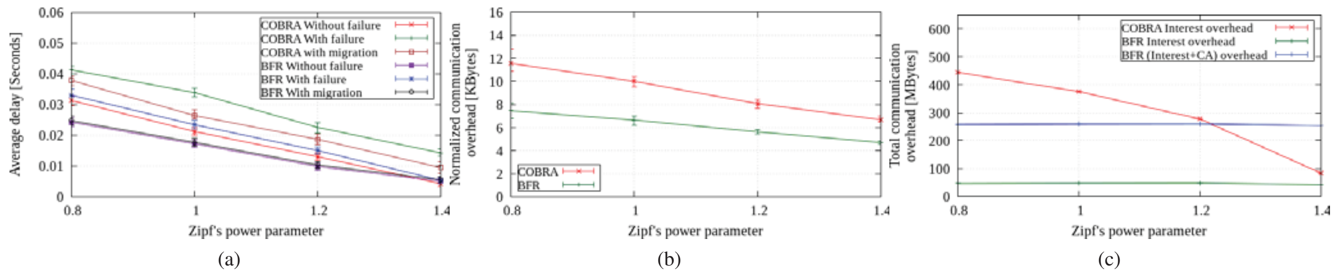

Fig. 7. Results for GEANT topology for different values of Zipf's power parameter: (a) Average round-trip delay; (b) Normalized communication overhead; (c) Total Interest communication overhead.

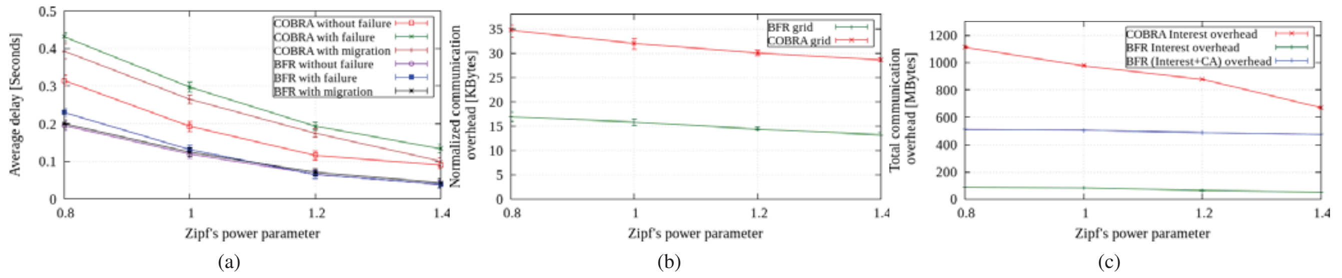

Fig. 8. Results for GRID topology for different values of Zipf's power parameter: (a) Average round-trip delay; (b) Normalized communication overhead; (c) Total Interest communication overhead.

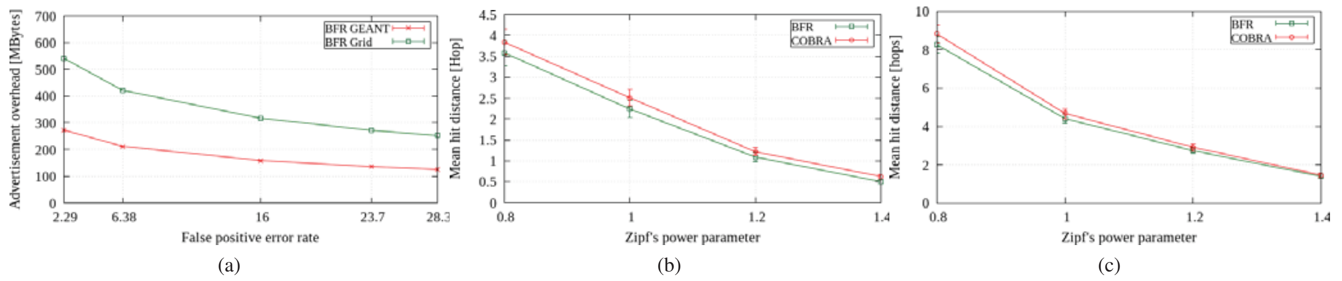

Fig. 9. Results for different values of Zipf's power parameter: (a) A comparison of content advertisement communication overhead vs. false positive probability for the grid and GEANT topologies; (b) Mean hit distance for GEANT topology; (c) Mean hit distance for the grid topology.

CAI messages to inform clients and routers about this event, thus clients and routers update routes once they receive new CAI messages. However, with COBRA, clients and routers are unaware of content migration events until they detect Interest retransmissions. Therefore, for COBRA, we see from Figs. 7a and 8a a much higher impact of content migration events on the average round-trip delay.

3) Normalized communication overhead: Figs. 7b and 8b compare results for normalized communication overhead, i.e., the total bandwidth used to forward all Interests and Data packets divided by the number of retrieved Data packets, for GEANT and grid topologies, respectively. We observe from these figures that BFR needs much less communication overhead to retrieve a Data packet than COBRA. The reason is that during the learning phase SBFs are empty and COBRA needs to flood the Interests, which incurs significant communication overhead. On the other hand, BFR nodes do not flood the Interests and nodes forward each Interest only over the paths en-route to the server(s) that provide the demanded content object. Thus, BFR shows much less normalized communication overhead compared to COBRA. For BFR, when the grid topology is used, Fig. 8b shows that the normalized communication overhead for all values of the Zipf's power parameter are approximately twice than when GEANT topology is used. The reason is that the grid topology is more connected than the GEANT topology and BFR uses multi-path content discovery. Thus, each Interest is forwarded over more links when the grid topology is used. Nevertheless, when the grid topology is used, Fig. 8b shows much higher normalized communication overhead for COBRA than when the GEANT topology is used. This is due to the Interest floodings in the learning phase of COBRA that entails much higher communication overhead when a grid-like topology is used.

4) Total communication overhead for Interests: Figs. 7c and 8c compare BFR and COBRA in terms of total communication overhead needed for sending Interests for different values of the Zipf's power parameter for GEANT and grid topologies, respectively. From these figures, it is clear that the total communication overhead for sending Interests is much higher for COBRA than for BFR. This is because COBRA needs to flood Interests during the learning phase when SBFs are empty. Nevertheless, the gap between the curves of COBRA Interest overhead and BFR Interest overhead is much bigger in Fig. 8c than in Fig. 7c, because the number of links are much higher in the grid topology than GEANT topology. Thus, it leads to much higher communication overhead for forwarding the Interests, specifically during the learning phase when COBRA floods the Interests. For COBRA, if we increase the value of Zipf's power parameter, we observe that the total communication overhead decreases significantly. The reason is that by increasing the value of Zipf's power parameter, the cardinality of the set of popular content objects decreases. Thus, the number of Interest floodings during the learning phase also decreases significantly. In Figs. 7c and 8c, for BFR, we see results in terms of *(Interest+CA) overhead*, i.e., the sum of the values of total communication overhead needed for sending Interests and the total communication overhead needed for propagating content advertisements in BFR. We still observe a big gap between the curves of COBRA Interest overhead and BFR (Interest+CA) overhead in both Figs. 7c and 8c. In Fig. 7c, we observe this gap at least if $\alpha = 0.8$ or $\alpha = 1$. If $\alpha = 1.4$, Fig. 7c shows that COBRA needs much less communication overhead for flooding Interests during the learning phase, because the number of popular content objects is much smaller and the topology is tree-like, thus having less number of links. Fig. 9a compares the communication overhead that BFR requires to propagate all the content advertisements in the grid and GEANT topologies, for different values of false positive probability. In Fig. 8c, we observe a bigger gap between the curves of COBRA Interest overhead and BFR (Interest+CA) overhead than the gap between these curves in Fig. 7c. This is due to the higher impact of Interest floodings in COBRA during the learning phase, in a more connected topology.

5) Mean hit distance: In Figs. 9b and 9c, we show the performance of BFR and COBRA in terms of mean hit distance, i.e., the mean path length an Interest requires traveling to reach the demanded content object, for GEANT and grid topologies, respectively. The first transmission of each Interest in the network has to reach the server that provides the demanded content object. However, the subsequent transmissions of the Interest can be retrieved from routers' caches. When the value of Zipf's α increases, a smaller number of content objects is popular, which are mostly cached at routers closer to the clients. Thus, we observe smaller values of mean hit distance for bigger values of α. We see from Figs. 9b and 9c smaller values of mean hit distance for BFR than for COBRA, which does not exceed 0.45 hops. Fig. 9c shows much larger mean hit distance than Fig. 9b. The reason is that the initial distance between the clients and servers is much longer in the grid topology than in the GEANT topology.

V. Conclusions

In this paper, we compared the performance of BFR and COBRA routing protocols. BFR works based on proactive content advertisements from servers, while COBRA routes Interests according to the route traces left from previously retrieved content objects. Neither BFR nor COBRA use any IP-based routing protocol as primary or fall-back routing mechanism, meaning that both BFR and COBRA are fully content-oriented routing policies. BFR outperforms COBRA in terms of average memory needed for storing routing information, average round-trip delay, normalized communication overhead, total Interest communication overhead, and mean hit distance. In future, we plan to study the scalability of BF-based routing protocols for ICN and to design scalable BF-based routing protocols.

References

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of Information-Centric Networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, Jul. 2012.

[2] A. Marandi, T. Braun, K. Salamatian, and N. Thomos, "BFR: A Bloom filter-based routing approach for Information-Centric Networks," in *Proc. of the 16th International IFIP Networking Conference*, Jun. 2017, pp. 1–9.

[3] A. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "NLSR: named-data link state routing protocol," in *Proc. of the 3rd ACM SIGCOMM workshop on Information-centric networking*, Aug. 2013, pp. 15–20.

[4] M. Tortelli, L. A. Grieco, G. Boggia, and K. Pentikousisy, "COBRA: Lean intra-domain routing in NDN," in *Proc. of the IEEE 11th Consumer Communications and Networking Conference (CCNC)*, Jan. 2014, pp. 839–844.

[5] H. Liu, X. De Foy, and D. Zhang, "A multi-level DHT routing framework with aggregation," in *Proc. of the ACM 2nd edition of the ICN workshop on Information-centric networking*, Aug 2012, pp. 43–48.

[6] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[7] S. Dharmapurikar, P. Krishnamurthy, and D. E. Taylor, "Longest prefix matching using Bloom filters," in *Proc. of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, Aug. 2003, pp. 201–212.

[8] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking (TON)*, vol. 8, no. 3, pp. 281–293, Jun. 2000.

[9] M. Varvello, D. Perino, and J. Esteban, "Caesar: A content router for high speed forwarding," in *Proc. of the Second Edition of the ICN Workshop on Information-centric Networking*, Oct. 2012, pp. 73–78.

[10] M. Lee, K. Cho, K. Park, T. T. Kwon, and Y. Choi, "SCAN: Scalable content routing for content-aware networking," in *Proc. of the IEEE Int. Conf. on Communications (ICC)*, Jun. 2011, pp. 1–5.

[11] Y. Wang, K. Lee, B. Venkataraman, R. L. Shamanna, I. Rhee, and S. Yang, "Advertising cached contents in the control plane: Necessity and feasibility," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Mar. 2012, pp. 286–291.

[12] A. W. Kazi, "Prefetching Bloom filters to control flooding in content-centric networks," in *Proc. of the ACM CoNEXT Student Workshop*, Nov. 2010, p. 22.

[13] F. Deng and D. Rafiei, "Approximately detecting duplicates for streaming data using Stable Bloom Filters," in *Proc. of the ACM SIGMOD international conference on Management of data*, Jun. 2006, pp. 25–36.

[14] C. Yi, J. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "On the role of routing in Named Data Networking," in *Proc. of the 1st international conference on Information-Centric Networking*, Sep. 2014, pp. 27–36.

[15] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2.0: A new version of the NDN simulator for NS-3," Tech. Rep., Jan. 2015.

[16] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: evidence and possible causes," *IEEE/ACM Transactions on Networking (TON)*, vol. 5, no. 6, pp. 835–846, Dec. 1997.