Content-Based Routing Using Subpart Orthogonal Codes

Christine Jardak, Janne Riihijärvi and Petri Mähönen
Department of Wireless Networks, RWTH Aachen University
Kackertstrasse 9, D-52072 Aachen, Germany
Email: {cja,jar,pma}@mobnets.rwth-aachen.de

***Abstract*—We study the use of orthogonal codes as alternative to Bloom filters in content-based routing. In particular we explore the trade-off between centralized solutions using regular orthogonal codes and generating codes locally by hashing. We demonstrate that introduction of *subpart orthogonal codes* allows distributed operation while leading to shorter key lengths than Bloom filters with the same probability of data localization. We study the performance of sub-part orthogonal codes by both analytical means and in a simulation environment. Results show that the use of sub-part orthogonal codes reduces overhead without sacrificing performance in both tree and grid topologies.**

I. Introduction

In this article we propose a memory efficient and low error content-based routing (CBR) for localizing data stored in a network. In CBR each block of data is represented by a short bit sequence called a *key*. Each node then maintains a routing table in which entries hold keys of the blocks of data reachable through the given neighbor. The keys should of course be as short as possible and be stored in a compact manner to conserve memory and to reduce communication overhead. The classical approach for generating and storing keys for arbitrary data is the use of *Bloom filters* (BFs) [1] which use sequences of hash functions to map data blocks into bit sequences of predefined length. This mapping is done by using the outputs of the hash functions to indicate locations of bits to be set to ones, the rest of the bits being zero. The Bloom filter then consists of the logical AND of these sequences. Checking whether a particular data block is a member of the set, the key of that data block is compared to the value of the Bloom filter. If the corresponding bits set to one in the key are also set to one in the filter, membership is assumed. As keys are added to the filter, more and more bits are set to ones, and the probability of *false positive* increases.

We propose to replace the hash-based key-generation of Bloom filters by binary orthogonal codes (OCs) [2], which in the case of perfect key-distribution would avoid false positive errors as shown in our technical report [3]. The theory of OCs is well known in other fields of communication [4]. However, although using OCs for CBR is a tempting possibility, one has to analyze the problem of key distribution, since this is a critical part of the method if one wants to make it practically applicable. We first analyze the centralized key distribution model. While this could guarantee orthogonal and unique codes, its drawback is the centralization. Second, we introduce a completely new model based on a class of partially orthogonal codes, called *subpart orthogonal codes* (SOCs) which open a possibility for distributed algorithms. We compare both models with a centralized and a distributed usage of BFs. We show that the distributed model does not completely eliminate the probability of false positives but decreases it drastically in comparison with BFs. Moreover, when using SOCs, shorter key lengths are sufficient in order to provide the same probability of data localization as BFs.

The remainder of this paper is organized as follows. In Section II we discuss the related work for content-based routing using BFs. In Section III we describe the general idea of the CBR. Section IV is the main section which details our centralized model using OCs and the distributed one using SOCs. We proceed in calculating the probability of false positive for SOCs in Section V. Section VI describes the simulation settings, while in Section VII we develop a simulation model allowing us to validate the proposed solutions in more complex networks. As a result, we analyze our model with a tree and grid network topologies. We conclude the discussion in Section VIII.

II. Related work

Hashing has been among the most common ways used to represent data chunks. However, due to the tradeoff between the number of bits used per sequence and the probability of false positives, BFs have received more widespread attention in content-based routing. We briefly highlight some significant work on BFs. In [5]–[8] the authors use the probabilistic Bloom filter content-based routing without any extra trials for minimizing the probability of false positive or the length of the filter in use. Various features were introduced to BFs. Fan et *al* extended the bit Bloom filter with a sequence of counters [9] providing a dynamic management for the filter by adding or deleting elements. Mitzenmacher took advantage from the large number of zeros in a Bloom filter and proposed a compressing scheme while maintaining the same false positive rate [10].

Although different approaches attempted to minimize the probability of false positive, they all meet in paying the price from the memory consumption. For example, Cohen and Matias propose the spectral Bloom filter [11] introducing a method based on the *recurring minimum* for enhancing the counter BFs. Keys are inserted in the filter as long as a single

minimum between the counters does not occur. If only a single minimum is indeed found, the rest of the keys are inserted in a secondary spectral Bloom filter, which results into doubling the size of memory in use. Moreover, in [12] the memory usage is exhausted by storing a cascading pipeline of secondary BFs for each main Bloom filter. i.e., the two pipelines $BF_A(0), \ldots, BF_A(n)$ and $BF_B(0), \ldots, BF_B(n)$ are used for the main BFs $BF_A(0)$ and $BF_B(0)$. The secondary BFs $BF_{A(i)}$ and $BF_{B(i)}$, for $i > 0$, contain elements that have caused a false positive error in $BF_{B(i-1)}$ and $BF_{A(i-1)}$, respectively, and are checked only when the verification of an element in the previous BFs resulted in a contradiction due to a false positive error. While the two latter approaches can be used with devices having no constraints on memory size, they do not scale to low-power communicating devices such as TelosB motes [13] having only 10 KB of RAM. Thus, our approach is designed to tackle the tradeoff between the probability of error and the memory consumption.

III. Content-based routing and Bloom filters

Content-based routing is a method for making forwarding decisions based on packet payload or a query for particular data without explicit use of node addresses. Once a new block of data has been stored, the storing node generates a key which identifies this block [14]. The key is then flooded to all nodes that are located less than a prescribed number of hops away. The nodes store the received key in their routing table consisting of a matrix of entries, with one column for each neighbor and with rows signifying the distance in number of hops to the node data is stored at. For a large number of data blocks, there might be several keys that correspond to blocks of data located the same number of hops away from a node. In order to keep the routing table compact, new keys are added bit-wise to the already stored keys. When retrieving the stored data, routing tables are used to forward the requests hop-by-hop to a storing node. A node that receives a request checks its routing table entries row at a time starting from the one corresponding to one-hop neighbors. The column of the routing table in which the key is found indicates then the neighbor node the query should be forwarded to. The key is conjectured to be in the entry with a certain probability of false positive.

One of the commonly used mechanisms to generate keys and store them are BFs [1]. Due to their flexibility in insertion and deletion of keys, we detail the use of counter BFs in content-based routing. For counter BFs checking if the key is contained in a routing table entry consists of bit-wise checking if the position of the 1s in the key match the positions of counters $\neq 0$ in an entry of the routing table. In case of a perfect match, the key is assumed to be stored in the entry. The keys in BFs are, however, generated from hash functions and are thus not necessarily orthogonal. This leads to possibility of spurious matches and causes queries to be forwarded to nodes that actually do not store the requested data with a certain probability of false positive [14]. Figure 1 depicts a BF content-based routing in a regular tree network. Each node in the network stores a block of data represented by its correspondent key. Node c maintains a four-column routing table, columns *c-e*, *c-f*, *c-g* and *c-a* consisting of 3 entries each. We take *K_1* as an example of a key to search. In our illustrative example we represent with solid green arrows on the tree, the route of the query sent from node a searching *K_1*. The dashed arrows represent the correct road leading to node *k* storing *K_1*. We show the sequential checking of the rows in the local routing table of node a, where *K_1* is matched in the third hop of column *a-c*. As a result, the query is directed towards node *c* where a false positive occurs leading into forwarding the query to node f.

IV. Orthogonal codes and Subpart orthogonal codes

In this section we proceed with the same spirit of the content-based routing. However, we introduce the use of orthogonal codes instead of Bloom filters in two different models: A centralized model and a distributed model.

A. Centralized model using OCs

Two codes are said to be orthogonal if two conditions are fulfilled: First, their respective normalized auto-correlation function is equal to unity 1 at a time-shifted zero. Second, their cross-correlation function is equal to zero for all time-shift values. Therefore, they have application in perfectly synchronized environments [4].

In the centralized model, we propose to index each block of data with a code such that all the codes in use are orthogonal. Unlike the idea of counter BFs [9], an entry of the routing table is a sequence of m counters. We focus on the usage of counter sequences instead of bits because they offer the possibility of removing a key from an entry of the routing table. A new key of length m bits is inserted in an entry by incrementing or decrementing [1] the correspondent counters of the entry. Testing the membership of a key in an entry follows the principle of code division multiple access [2]. The bits of the key are multiplied one-to-one with the correspondent counters of the entry. The values of the resulting m counters are then summed together. If the output of the sum is equal to m, the key is conjectured to exists in the entry with a probability of false positive equal to 0. The Use of OCs eliminates the occurrence of a false positive error as long as the number of original chunks of data is smaller than or equal to the key length. The drawback of this approach is however the code/data association which has to be unique. This mechanism requires a centralized procedure, for example some type of shared knowledge such as clock synchronization [4], predefined look-up tables or logical division of nodes into clusters.

B. Distributed model using SOCs

In this section, we detail a new distributed method to associate keys automatically and almost uniquely by introducing a new concept of subpart orthogonal codes.

[1] Decrement is possible since orthogonal codes are sequences of -1 and 1

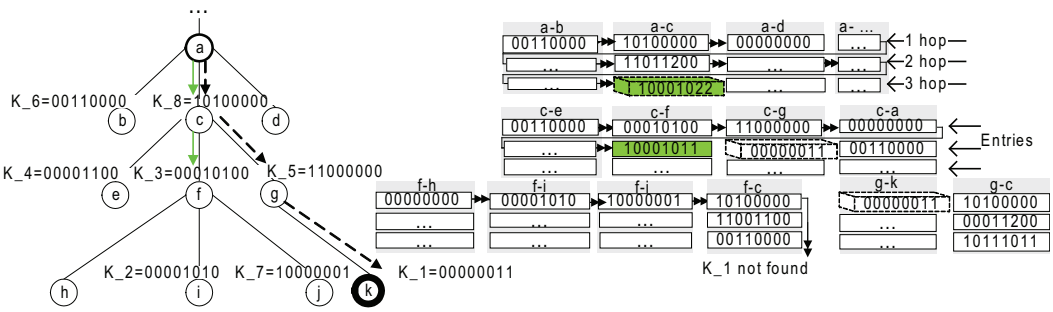
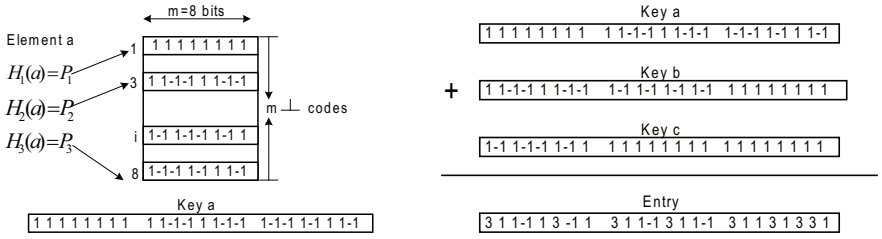


Fig. 1. Content-based routing on a regular tree topology using counter BFs: We show the effect of the probability of error in deviating the routing from the correct way towards the storing node. The 3-D dashed cells in the routing tables lead to the correct route towards locating *K_1*, while the green ones represent the mistaken entries that were chosen by the content-based routing.



(a) Creating SOC. (b) Forming an entry.

Fig. 2. The creation of a subpart orthogonal code of element a. Figure 2(a) the hash functions $H_1(a), H_2(a)$ and $H_3(a)$ result into indexes 1, 3 and 8. The corresponding orthogonal codes have a lenght of $m = 8$ bits are concatenated to form *Key a*. Figure 2(b) shows the insertion of element a, b and c in the same entry. After creating the corresponding *Key a*, *Key b* and *Key c* we execute a bit-to-bit addition of the three keys. The result is $n = 24$ counters that form the entry.

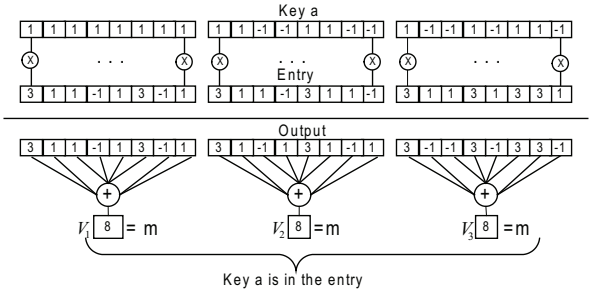


Fig. 3. The membership verification of *Key a* in the entry. We execute a scalar multiplication between the subparts of *Key a* and the corresponding ones for the entry. For each subpart of the output we sum the values of the counters. Since $V_1 = m, V_2 = m$ and $V_3 = m$ *Key a* is certified to be in the entry

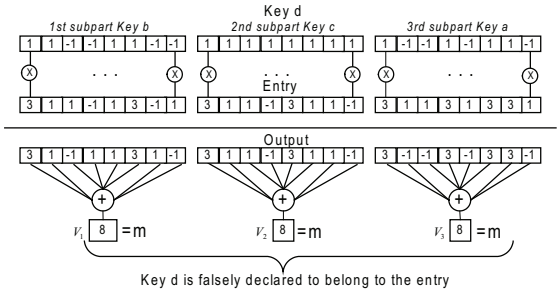


Fig. 4. Example of a false positive error. We verify the membership of *Key d* in the entry. *Key d* is formed from the 1^{st} *subpart Key b*, 2^{nd} *subpart Key c* and 3^{nd} *subpart Key a*. The verification results in $V_1 = m, V_2 = m$ and $V_3 = m$ although *d* is not a member of the entry.

1) Creating SOCs: Subpart-orthogonal codes of length m bits are formed by concatenating p partitions of n bits each, where $n \times p = m$, $p \geqslant 1$. We define a family of SOCs as a set of keys having the same value of *p* and *n*. Because of the way they are constructed, keys of the same family are not necessarily orthogonal. However, the individual partitions of a key are mutually orthogonal and are also orthogonal to the partitions of any other key in the same family. The construction of the SOCs is as follows. Assume an array of n orthogonal codes of length n bits, and then choose p independent hash functions, $H_1, H_2, \ldots, H_p$, each with a range $1, \ldots, p$. For a block of data a, the orthogonal codes at positions $H_1(a), H_2(a), \ldots, H_p(a)$ are concatenated to form the corresponding subpart orthogonal code. Figure 2(a) depicts an example of creating a 24-bit subpart orthogonal code identifying element a. The key has 3 partitions of 8 bits each.

2) Forming an entry: An entry of a routing table is formed from m counters initially all set to 0. In order to insert multiple data blocks to the entry, we compute the key of each block. We add all the keys bit-wise, Figure 2(b) showing an example of inserting three elements a, b and c into the same entry.

3) Verification: In order to verify whether element a belongs to the entry or equivalently its corresponding key *Key a*, we multiply each subpart of the key with the corresponding ones of the entry. This generates an output of p subparts, each

with m counters. We finally sum up the counter values of the subparts which yields a set of p values V_1, V_2,..., V_p. If $V_1 = m$, $V_2 = m$,... and $V_p = m$ then the key is declared to be in the entry. Since other keys in the entry may have the same subparts, some false positive errors occur. In Figure 3, we illustrate the query of element a in the entry. The verification method results in $V_1 = 8$, $V_2 = 8$ and $V_3 = 8$. Since all of the three values are different from zero, we declare that element a is in the entry although there exists a certain probability that we are wrong.

4) False positive error: This error, also called *false drop*, occurs when an element is falsely declared to belong to the entry. This error occurs if all the subparts p of the queried key are found in their correspondent positions in some other keys in the entry. This yields $V_1 = m$, $V_2 = m$,... and $V_p = m$ even if the searched key is not in the entry. Figure 4 depicts an example of a false positive error. Having a query for element d, the verification of *Key d* leads to the faulty conclusion that the element is a member of the entry.

V. Probability of false positive errors for SOCs

We now calculate the probability of false positive errors in an entry *I* of the routing table. Denote by n' the number of distinct blocks of data in the entry and by N_{dat} the number of original data blocks. We consider SOCs with p partitions where each partition is an m bit orthogonal code. We denote by Ev_{fp} the event that a false positive error occurs when verifying an element a in the entry. This event is the consequence of two other events. The first event denoted Ev_{nu}, is the event that the subparts of the *Key a* are not unique but are similar to some of the correspondent subparts of other keys that have been assigned to other blocks of data. The second denoted Ev_{sp}, is that of finding similar subparts of *Key a* in the correspondent subparts of keys that are in the entry. The probability to commit a false positive error can now be expressed as $\mathbb{P}(Ev_{fp}) = \mathbb{P}(Ev_{nu} \cap Ev_{sp})$. The event Ev_{sp} is the combination of p other events. The first event $Ev_{sp\text{-}1}$, occurs when the first partition of *Key a* corresponds to the first partition of at least one other key in the entry. Events $Ev_{sp\text{-}2}$,..., $Ev_{sp\text{-}p}$ are defined similarly. This yields $\mathbb{P}(Ev_{fp}) = \mathbb{P}(Ev_{nu} \cap Ev_{sp\text{-}1}, Ev_{sp\text{-}2}, \ldots, Ev_{sp\text{-}p})$. Since keys in the entry can also have none unique subparts, this leads to a dependency of both events Ev_{nu} and $Ev_{sp\text{-}i}$, $\mathbb{P}(Ev_{fp}) = \mathbb{P}(Ev_{nu})\mathbb{P}(Ev_{nu} \mid Ev_{sp\text{-}1})\mathbb{P}(Ev_{nu} \mid Ev_{sp\text{-}2}) \ldots \times \mathbb{P}(Ev_{nu} \mid Ev_{sp\text{-}p})$. The probability that the key of an element in the entry has its i^{th} partition similar to any other partition of the N_{dat} keys of is $\mathbb{P}(Ev_{nu} \mid Ev_{sp\text{-}i}) = \frac{n'}{N_{dat}}$. We denote by N_{ss} the number of data blocks whose keys have at least one subpart shared between more than one key and by N_{us} the number of data blocks whose keys have a unique subparts, this yield $\mathbb{P}(Ev_{nu}) = \frac{N_{ss}}{N_{dat}} = 1 - \frac{N_{us}}{N_{dat}}$. The probability that an orthogonal code from the initial set of m codes is chosen exactly once, is the probability that it is not chosen for $N_{dat} - 1$ blocks of data and is selected only by one block of data. Due to the independence of both events, the probability that an orthogonal code is chosen exactly once is $\left(\frac{1}{m}(1 - \frac{1}{m})^{N_{dat}-1}\right)N_{dat}$. Summing over all the m orthogonal codes, the expected number of orthogonal codes that are chosen exactly once is $N_{us} = m\left[\frac{1}{m}\left(1 - \frac{1}{m}\right)^{N_{dat}-1}\right]N_{dat}$. Finally by substituting the value of N_{us} into $\mathbb{P}(Ev_{nu})$ we obtain $\mathbb{P}(Ev_{nu}) = 1 - \left(1 - \frac{1}{m}\right)^{N_{dat}-1}$. Hereafter, by replacing the values of $\mathbb{P}(Ev_{nu})$ and $\mathbb{P}(Ev_{nu} \mid Ev_{sp\text{-}i})$ in $\mathbb{P}(Ev_{fp})$, the probability that a false positive occurs when verifying an element in an entry becomes

$$\mathbb{P}(Ev_{fp}) = \left[\left(1 - \left(1 - \frac{1}{m}\right)^{N_{dat}-1}\right)\frac{n'}{N_{dat}}\right]^p. \quad (1)$$

VI. Setup of simulations

The aim of our simulations is to validate and compare the centralized model using OCs and the distributed model using SOCs with BFs. For a fair comparison, we consider both centralized and distributed approaches for BFs denoted CBFs and DBFs, respectively. In the centralized approach the simulation tries several numbers k of hash functions and chooses the optimal one that minimizes the probability of false positive in the network. The distributed approach is the traditional one for content-based routing using BFs. We calculate the highest number of keys inserted in a Bloom filter. This corresponds to the number of keys inserted in any entry of the last row in the routing table. The number of hash functions is computed from $k = \ln(2)\frac{m}{n}$, where authors in [1] denote by n the length of the filter in bits and by m the number of members inserted in the filter. While this method minimizes the probability of error for the BFs in the last rows of the routing table, it is not necessarily minimizing it for BFs in lower rows containing fewer keys [15].

We validate our proposed models on a regular tree topology and a grid topology assuming the shortest path between the nodes. Complex topologies are subject of evaluation in our future work. We exclude the border effects which are not the major issue for this article. We consider a grid topology of 37×37 nodes, resulting into a network of 1369 nodes and take $\frac{1}{4}$ of them as a neighborhood fixing by that $h = 13$ hops. Unlike the grid topology, we consider a tree of degree 4 and depth 5, resulting into a network of 1365 nodes and take a neighborhood of $h = 4$ covering $\frac{1}{4}$ of the nodes in the network. Each node in the network stores maximum of one block of data. In order to increase the probability of localizing a data block, we treat the case of having several replicas from the original data blocks that are randomly distributed in the network. We consider 128 original blocks of data and vary the number of replicas reaching maximum the value of 10. Original blocks of data as well as their replicas are stored in randomly selected nodes. We study the behavior of the probability of data localization as a function of the key length using various lengths of BFs from 128 up to 896 bits. Conceptually, for SOCs, we fix the length of a subpart to 128 bits and vary the number of partitions from 1 to 7 resulting in the same key lengths.

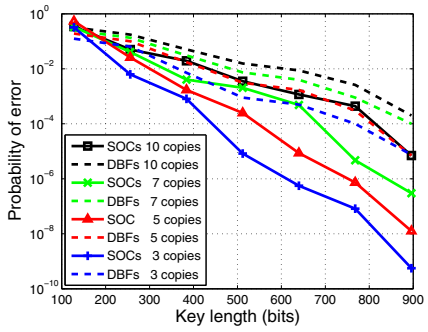


Fig. 5. The probability of false positive as a function of SOCs lengths.

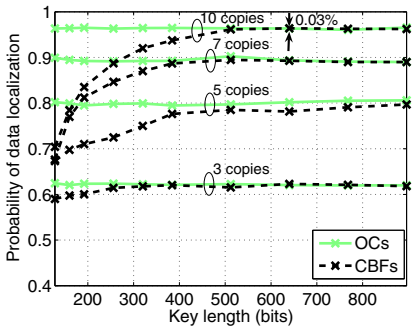


Fig. 7. The case of centralized approaches for OCs and BFs on the grid.

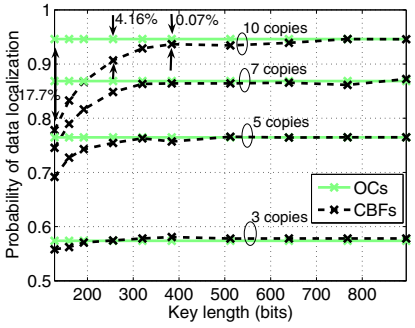


Fig. 6. The case of centralized approaches for OCs and BFs on the tree.

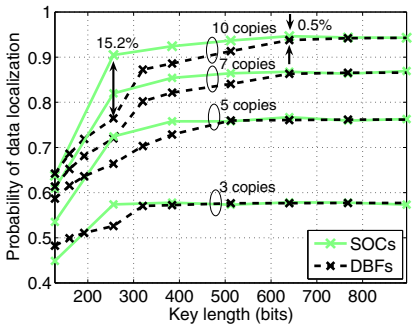


Fig. 8. The case of distributed approaches for OCs and BFs on the tree.

VII. Simulation Results

In this section, we present the simulation results of the probability of false positive for SOCs as a function of the key length. We show as well the enhancement in the probability of data localization for the centralized and distributed models using OCs in comparison with the respective ones using BFs.

A. Probability of error for SOCs

In Figure 5 we show the probability of false positive error as a function of the key length. For keys longer than 128 bits the probability of false positive for DBFs is significantly higher than the SOCs one. Due to their orthogonal partitions, SOCs are indeed subject to much less positive errors than BFs. The length 128 bits is a special case for SOCs due to the fact that they have a single partition. This results into several data blocks having similar keys which drastically increase the probability of error. For both BFs and SOCs, a solution for minimizing the probability of false positive consists of enlarging the size of the keys. For SOCs, this results into a higher number of partitions p which exponentially reduces the false positive error given by (1). For DBFs, this results into increasing the number of hash functions k which also reduces the probability of false positive. However, our analysis shows that beyond one partition SOCs are more efficient than DBFs. For instance, in order to fulfill a requirement of probability of false positive of 0.05, Figure 5 shows that in the case of 10 replicas, SOCs need a key length of 256 bits while DBFs reaches this value for a key length of 384 bits.

B. Centralized approaches

Applying the centralized approach to the tree network results in Figure 6. We highlight the case of 10 replicas for a key length of 128 bits and note that the probability of data localization for OCs is 17.7% higher than the one for CBFs. After doubling the key length, the gap between OCs and CBFs decreases to 4.16%. For a key length of 384 bits, the gap between CBFs and OCs is reduced to 0.07%. Since the size of each entry matches the size of the keys, this solution also linearly increases the memory usage of each node. In a grid topology, nodes are connected through multiple paths increasing the probability of data localization. This phenomenon can be seen from Figure 7 showing a 0.8 probability of data localization for OCs in a grid topology when distributing 5 replicas for each block of data. On the other hand, using the same number of replicas for a tree topology the probability of data localization using OCs reaches a value of only 0.77 as depicted in Figure 6. Moreover, multiple paths affect as well the slope of the probability of data localization for CBFs. For the grid topology i.e., for 10 replicas, the CBFs curve manages to follow the OCs one after a key length of 640 bits while it follows it quickly after a key length of 384 bits for the tree.

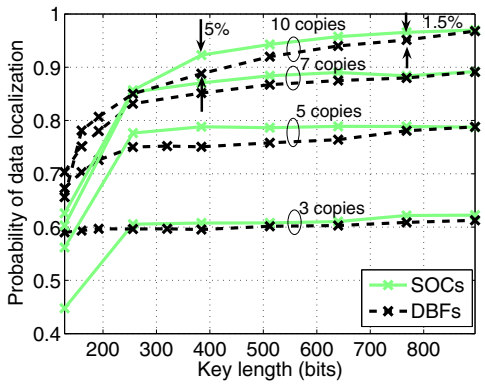


Fig. 9. The case of distributed approaches for Ocs and BFs on the grid

C. Distributed approaches

Applying the distributed approach to the tree network results in Figure 8 which verifies our expectations that the probability of data localization for SOCs is significantly better than the one for DBFs. As an example, in the case of 10 replicas, for a key length of 128 bits, the probability of data localization for DBFs and SOCs are equal. This is a particular case for SOCs where the keys comprise only one partition $p = 1$. Therefore, they experience a high number of false positive errors limiting the probability of data localization. After doubling the key length (256 bits), the number of SOC partitions increases to 2 enhancing the probability of data localization with a factor of 15.23% in comparison with DBFs. For $p \geq 2$ the gap starts decreasing reaching 0.5% of difference for a key length of 640 bits. The results for the grid topology in Figure 9 show similar behavior. As an example, in the case of 10 replicas and for a key length of 384 bits, the probability of data localization for SOCs is 5% larger than the one for DBFs. The gap starts decreasing reaching 1.5% of difference for a key length of 786. In this approach as well the grid topology has a higher probability of data localization due to the multiplicity of paths. i.e., having 10 replicas and a key length of 896 bits, the probability of data localization reaches 0.97 for the grid while it is 0.94 for the tree.

VIII. Conclusions

We have introduced two models for content-based routing making use of orthogonal codes. The first model is centralized and associates a unique orthogonal code for each block of data eliminating with that the occurrence of a false positive error. The second model is distributed and generates a new type of keys constituted of subpart orthogonal codes. These subpart orthogonal codes are not explicitly and uniquely orthogonal among them but are approximately orthogonal. We provide an analytical calculation for the probability of false positive generated for the latter model. Simulations have validated the accuracy of our work, and confirm that the probability of false positive error for our distributed model is lower than the one generated from BFs. We show the validity of these conclusions on a grid and on a tree topology considering as well various replicas from each block of data. Our results show a higher probability of localizing a block data in the networks as well as the use of shorter key lengths minimizing by that the demand on memory space.

Acknowledgment

We would like to thank the European Union (Projects UbiSec&Sens and WASP), for providing partial financial support for our work. We acknowledge also financial and logistic support from DFG (Deutsche Forschungsgemeinschaft) and RWTH Aachen University through UMIC Research Centre.

References

[1] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," in *Proc. Communications of the ACM*, vol. 13, July 1970, pp. 422–426.

[2] M. Rupf and J. L. Massey, "Optimum sequence multisets for synchronous code-division multiple-access channels," *IEEE Transactions on Information Theory*, vol. 40, pp. 1261–1266, july 1994.

[3] C. Jardak et al., "Data Retrieval Based on Orthogonal Keys in Wireless Sensor Networks." [Online]. Available: http://www.mobnets.rwth-aachen.de/fileadmin/_temp_/OC-vs-BF-technical-report.pdf

[4] A. Goldsmith, *Wireless Communications*. Cambridge Univ-press, 2005.

[5] C. Jardak et al., "Distributed Information Storage and Collection for WSNs," in *Proc. of the fourth IEEE International Conference on Mobile Ad-hoc and Sensor Systems(MASS)*, Italy-Pisa, October 2007.

[6] Y. P. G. Koloniaria and E. Pitoura, "Content-Based Overlay Networks for XML Peers Based on Multi-level Bloom Filters," in *Proc. of the 1998 ACM SIGMOD International Conference on the Management of Data*, Washington-USA, September 1998, pp. 254–256.

[7] S. C. Rhea et al., "Probabilistic location and Routing," in *Proc. of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 3, New York-USA, June 2002, pp. 1248–1257.

[8] T. Repantis and V. Kalogeraki, "Data Dissemination in Mobile Peer-to-Peer Networks," in *Proc. of the 6th International Conference on Mobile Data Management (ACM MDM)*, Ayia Napa, May 2005, pp. 211–219.

[9] L. Fan et al., "Summary cache: A scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking*, vol. 8:3, pp. 281–293, 2000.

[10] M. Mitzenmacher, "Compressed Bloom Filters," in *Proc. of the 20th annual ACM symposium of Principles of distributed computing*, Rhode Island, 2001, pp. 144–150.

[11] S. Cohen and Y. Matias, "Spectral Bloom Filters," in *Proc. of the 2003 ACM SIGMOD International Conference on the Management of Data*, Washington-USA, June 2003, pp. 241–252.

[12] B. Chazelle et al., "The Bloomier Filter: An Efficient Structure for Static Support Lookup Tables," in *Proc. The fifteenth annual ACM-SIAM symposium on Discrete Algorithms*, 2004, pp. 30–39.

[13] J. Polastre et al., "Telos: Enabling Ultra-Low Power Wireless Research," in *Proc. of IPSN'05*, CA-USA, Apr. 2005, pp. 364–369.

[14] A. Brodrei et al., "Network Applications of Bloom Filters: A survey," *IEEE Transactions on Communications*, vol. 30, pp. 91–99, May 2004.

[15] C. Jardak et al., "Analyzing the Optimal Use of Bloom Filters in Wireless Sensor Networks Storing Replicas," in *Proc. The IEEE Wireless Communications and Networking Conference (WCNC)*, Hungary, Sep. 2009.