

Bloom Filter-based Routing for Dominating Set-based Service-Centric Networks

Ali Marandi*, Vincent Hofer*, Mikael Gasparyan*, Torsten Braun*, and Nikolaos Thomos†

*University of Bern, Bern, Switzerland

†University of Essex, Colchester, United Kingdom

Abstract—A service-centric network requires a routing protocol to route service requests towards service providers. Routing operations can be divided into *intra-domain* and *inter-domain* routing. In the proposed approach, a so-called *supernode* is responsible for managing its own domain as well as for communicating with the supernodes of other domains to perform inter-domain routing. In order to appoint appropriate nodes as supernodes in the network topology, in this paper, we use Dominating Sets (DS) and Connected Dominating Sets (CDS). We propose fully distributed algorithms for constructing DS as well as CDS over the network topology. To prepare routing information, the nodes of each domain inform their supernodes about their available service names and resources (e.g., CPU, RAM). To this aim, the nodes use Bloom filters which reduce bandwidth and storage overhead. The performance evaluation shows that the required bandwidth overhead for DS and CDS construction algorithms increases with the topology size. The results also show that for large network topologies, CDS-based routing requires significantly less bandwidth overhead than both DS-based routing and Named Data Networking with multicast forwarding strategy. Finally, from the results we can observe that both DS-based and CDS-based routing have significantly lower service retrieval time than NDN multicast strategy.

I. INTRODUCTION

Information-Centric Networking (ICN) [1] proposes architectures to overcome the shortcomings of the current IP-based Internet architecture, such as location-dependent content retrieval, end-to-end path-based security, etc. Content-Centric Networking (CCN) [2] is the first distributed architecture proposed for ICN with fully content-oriented routing, content retrieval, and inherent security. Named Data Networking (NDN) [3] was proposed as a project that brings up CCN views into practice in terms of designing and deploying a fully name-based protocol stack. In CCN/NDN, content objects are identified by names. Two types of messages, namely *Interest* and *Data* messages are used for content discovery and content forwarding, respectively. When clients demand some data, they send Interest messages to request Data messages. Routers further forward Interest messages if they do not have the demanded Data message until these messages reach a router or a server that contains the corresponding Data message, i.e., the Data message with the matching name. Then, the Data message follows the reverse path the Interest message has travelled to reach the client that issued a request for it. In CCN, routers can store copies of passing by Data messages

so that the future Interests could be served by routers closer to the users. Basic components in CCN and NDN are Pending Interest Tables (PIT) to store Interest messages, Content Store (CS) tables to store the Data messages, and Forwarding Information Base (FIB) tables to maintain information about next hop face(s) for different name prefixes.

In [4], a framework called Service-Centric Networking (SCN) is proposed that extends CCN architecture so that it also supports services. Services could be manipulated content objects, e.g., transcoded audio/video, processed image, Google map directions, etc. Other examples of services could be storage resources provisions [4]. L-SCN [5] presents a layered routing architecture for SCN that leverages both intra-domain and inter-domain routing. To assure inter-domain and intra-domain communication, L-SCN uses so-called supernodes (SNs). Every node in a network topology is either an SN or a regular node. Each SN is aware of information available in its own domain, e.g., the provided services and available resources. For delivering the requested information, intra-domain and inter-domain routing is used. In the original L-SCN protocol, it is assumed that the network topology is clustered, but the protocol lacks of algorithms to cluster network nodes and to choose the SNs. To cope with this issue, in this paper, we present two fully distributed clustering algorithms browsing ideas from Dominating Set (DS) and Connected Dominating Set (CDS) concepts in graph theory [6]. Although the proposed clustering algorithms are designed to complement L-SCN, they are generic and can be used with other protocols that require formations of cluster of nodes. Once SNs are selected and nodes are clustered, SNs collect information about services and resources available in their domains so that they can route service requests. Nodes use Bloom Filters (BF) [7] to compress the sets of their available service names saving significant resources.

We assess the performance of the proposed routing protocols over various real topologies [8], [9] and observe that: 1) the bandwidth overhead required to construct DS and CDS increases with the topology size, 2) the proposed CDS construction algorithm requires more bandwidth overhead than our DS construction algorithm, 3) for large topologies, the proposed CDS-based routing protocol requires drastically less bandwidth overhead to route service requests than both our DS-based routing protocol and vanilla NDN with multicast strategy [10], and 4) we observe that the CDS-based routing

TABLE I: Acronyms and their definitions.

Definition	Acronym
Bloom Filter	BF
Bloom Filter-based Routing [11]	BFR
Clustering Data	CD
Clustering Initializing Interest	CII
Connected Dominating Set	CDS
Dominating Set	DS
Forwarding Information Base	FIB
Minimum Connected Dominating Set	MCDS
Minimum Dominating Set	MDS
Super Node	SN
Supernode Connection Data	SNCD
Supernode Connection Interest	SNCI
Supernode Selection Interest	SSI
Supernode Selection Data	SSD
Service and Resource Availability Data	SRAD
Service and Resource Availability Interest	SRAI
Service Data	SD
Service Provision Refusal	SPR
Service Request	SR

protocol achieves slightly better service retrieval time than DS-based routing, while both DS-based and CDS-based routing protocols have much less service retrieval time compared to NDN with multicast strategy.

II. PRELIMINARIES AND RELATED WORKS

In this section, we briefly introduce some necessary background information.

A. Bloom Filters

A Bloom Filter (BF) [7] is a data structure that is used to compactly represent a set. BFs have been used in networking [11]–[13] for drastically reducing the bandwidth required to communicate sets. A BF uses a bit vector as well as hash functions to represent a set. The hash functions are used for insert and search operations. Using BFs, network nodes require significantly less storage resources to store sets [11]. When one uses BFs, false positive reports are possible, while false negative reports cannot happen. There is a trade-off among the probability of false positive reports p , length of the bit vector m , the number of hash functions k , and the number of inserted elements n , which is shown in (1)

$$m = -\frac{n \ln(p)}{(\ln(2))^2}, k = \frac{m}{n} \ln(2). \quad (1)$$

BF-based routing have been studied in several works [11], [13], [14]. Push-based BFR [11] uses BFs to advertise content object names in the network. In this protocol, the required bandwidth and storage overhead for propagating and storing BFs grows linearly with the number of available content object names [13]. To cope with this issue, pull-based BFR is proposed as an alternative protocol, which uses BFs to only advertise the names of requested content objects [13]. This strategy reduces significantly the required bandwidth and storage overhead for BF-based content advertisements.

B. Dominating Sets

For a graph $G = \langle V, E \rangle$, where V is the set of vertices and E is the set of edges, a set $D \subseteq V$ is called a *Dominating Set* (DS) provided that each vertex of V is either an element

of set D or it is directly connected to an element of set D . The elements of a DS are called dominators and the direct neighbors of dominators are called dominated nodes. The DS with the minimum possible cardinality is called Minimum DS (MDS). Given a DS $C \subseteq D$, if any vertex $v_i \in C$ can reach any other vertex $v_j \in C$ using a path that does not leave set C , the latter set is called *Connected Dominating Set*. A CDS with the minimum possible number of elements is called MCDS. In Fig. 1, we provide examples of DS, MDS, CDS, and MCDS over a graph. The determination of MDS and MCDS are NP-hard problems [15].

DS and CDS find applications in several networking problems, e.g., creating a virtual backbone for routing in ad-hoc networks [15]–[19]. As finding an MCDS is NP-hard, the work in [15] proposed to approximate MCDS's, by finding dominating sets slightly larger than a dominating set with the fewest possible number of nodes. This set is used as virtual backbones for wireless ad-hoc networks in a unit-disk graph to alleviate broadcast storms. The authors of [15] present a distributed algorithm to approximate MCDS, which first finds the maximal independent set and then uses a Steiner tree to connect the vertices in this set. The work in [16] proposes another distributed algorithm that does not depend on spanning trees. This algorithm maintains the same approximation ratio after topology changes, but needs a leader node to operate. If a leader node is not given, leader election should take place which adds time and message complexity to the algorithm. In [17], a fully localized and distributed algorithm called r-CDS is proposed, which does neither require to build a tree nor to select a leader. The work in [18] examines distributed algorithms proposed for MCDS approximations and presents distributed construction of an approximate MCDS, for unit disk graphs.

The work in [19] uses DS's to provide a solution for collaborative caching in ICN. It leverages DS's for efficient collaborative caching to reduce caching redundancy in the network. It combines routing and caching strategies with a CDS. In [19], the most popular content objects are cached at the core routers. When CDS is used for the core routers routing becomes simpler, as every core router is directly connected to at least another core router, and, therefore, routing can be done solely through core routers. To the best of our knowledge, the work in [19] is the only that makes use of DS's to provide a solution for ICN-based networks, but it uses a centralized clustering algorithm in the sense that the network topology as well as the number of neighbors for each node are known a priori. The main drawback of [19] is that even if the CDS construction methodology can operate easily for small topologies, it does not scale with the size of the network topology.

The above-mentioned works in [15]–[18] are for wireless ad-hoc networks, modeled as unit disk graphs. Differently from these works, in this paper, we focus on using DS and CDS for intra-domain and inter-domain routing and propose fully distributed algorithms to construct these sets for any arbitrary network topology.

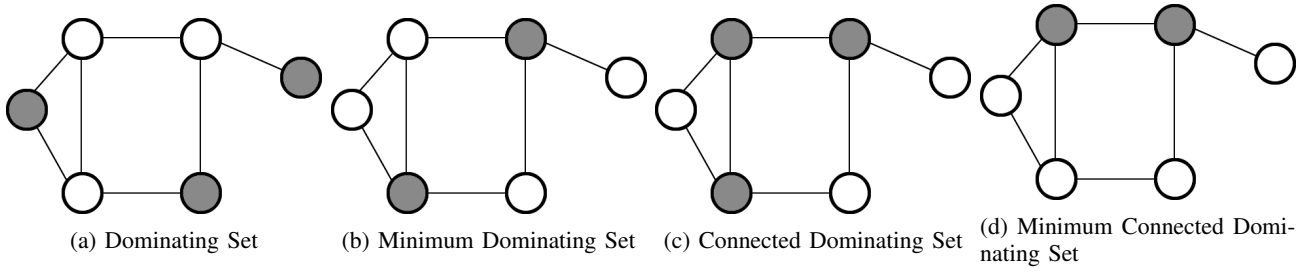


Fig. 1: Different Dominating Sets; Dominator nodes are grey, Dominated nodes are white

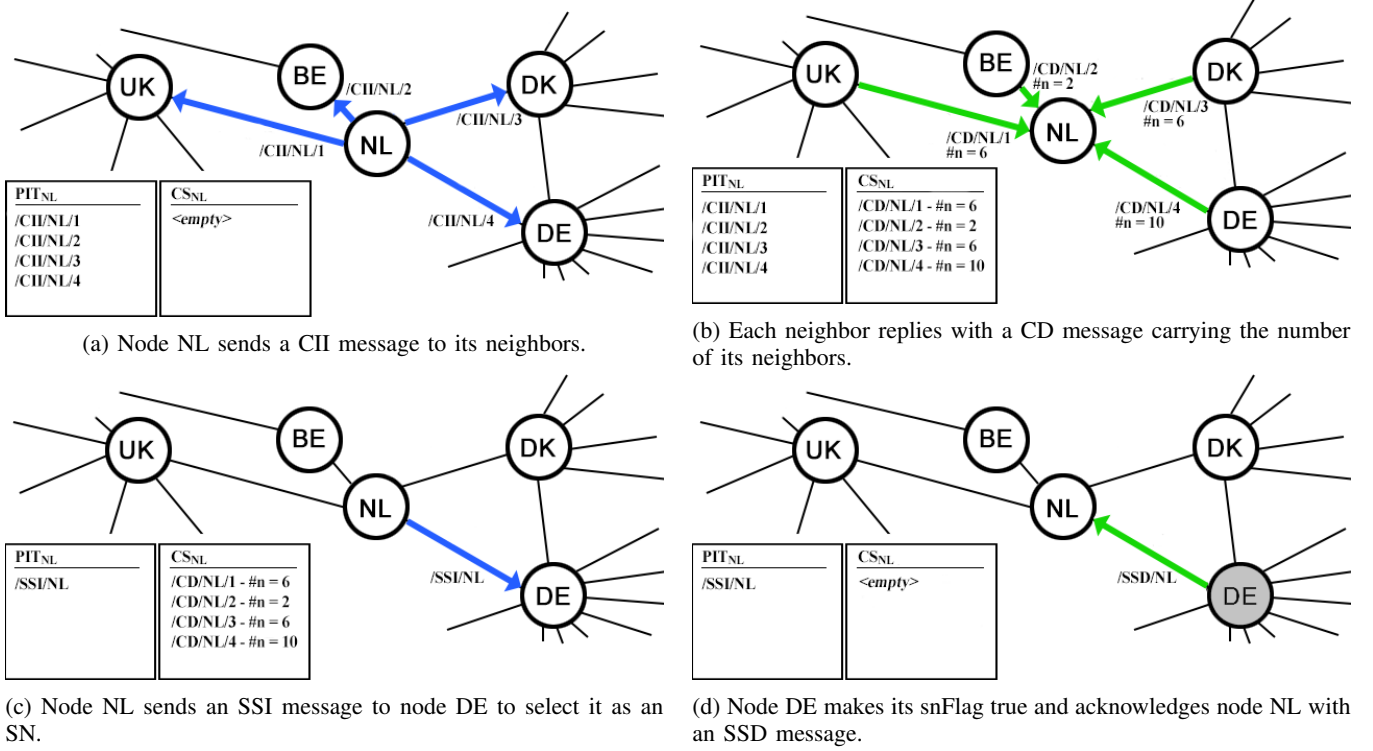


Fig. 2: The clustering algorithm described with a part of the GEANT network

III. CLUSTERING NETWORK NODES

In this paper, we first propose an algorithm for constructing a DS. Then, we suggest an algorithm for converting a DS to a CDS. In the following, we present our algorithms for DS as well as CDS construction to cluster the nodes in the network topology.

A. Dominating Set Construction

Our DS construction algorithm requires the nodes to store the following parameters:

- **snFlag**: a flag that indicates whether a node is an SN or not.
- **snFaceId**: the face ID of the face over which the SN of the associated domain can be reached.

To describe our algorithm, we use a part of the GEANT topology [8] illustrated in Fig. 2. In this topology, nodes represent core routers located at an European country. To

show the message exchanges needed for constructing a DS, we focus on the node with label *NL* that represents Netherlands in Fig. 2. Interest messages are shown with blue arrows and Data messages with green arrows. Although we only show the messages sent by *NL*, the algorithm runs in parallel for each node.

- 1) The algorithm starts with node *NL* sending a Clustering Initializing Interest (CII) message with name */CII/NL/faceID* over each of its faces to ask each of its neighbors to send back the number of its neighbors (Fig. 2a). The neighbors of node *NL* store the received CII messages in their PIT tables.
- 2) In Fig. 2b, we show that when *NL*'s neighbors receive a CII message, they respond with a Data message called Clustering Data (CD) message with the same name containing the number of its neighbors as well as its node ID. When the CD message arrives at the node that issued the corresponding CII message, it is stored in the CS

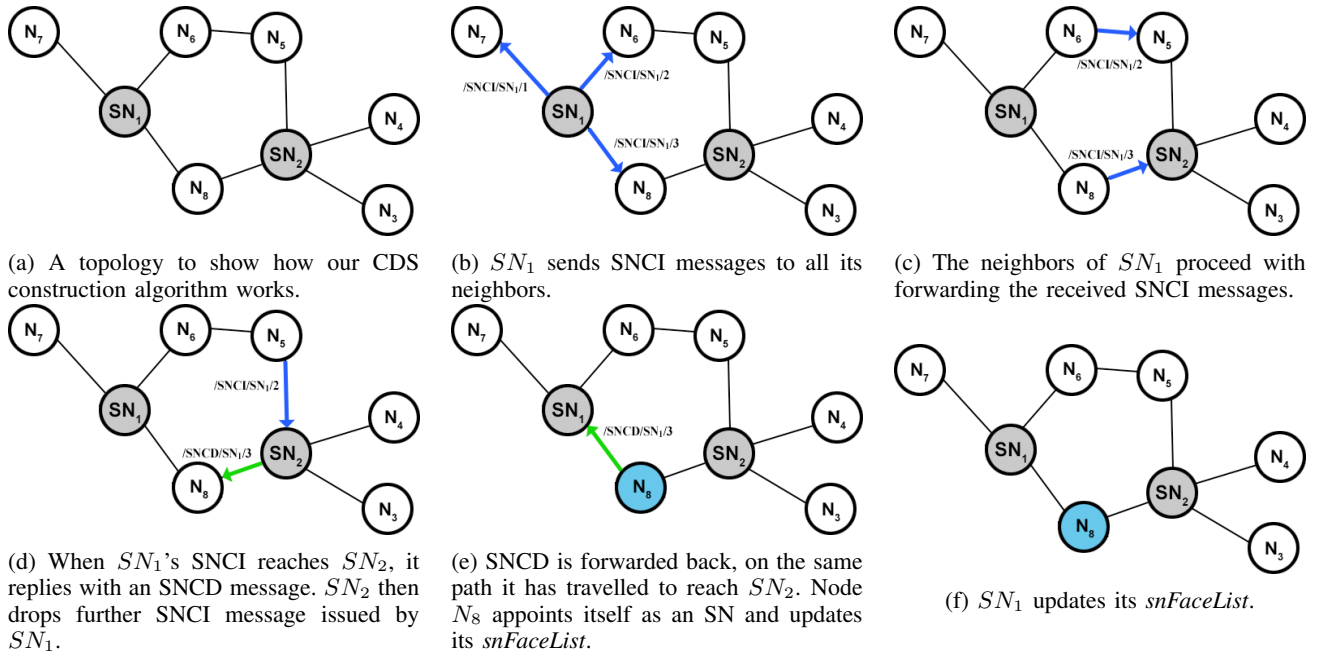


Fig. 3: The CDS construction algorithm

table. Then, the face ID through which the CD message is received is stored in the Content Store entry associated with the received CD message. In such case, the PIT entries of the CII messages are satisfied and are deleted from the PIT.

- 3) In Fig. 2b, node NL waits for a short time interval t_w to receive CD messages from all its neighbors. Then, node NL compares the number of its own neighbors with that reported by its neighbors. If node NL has more neighbors than its neighbors, it appoints itself as an SN. As Fig. 2c shows, node NL sends a Supernode Selection Interest (SSI) message with name $/SSI/NL$ to its neighbor with the highest number of neighbors, which is node DE . If a node has multiple neighbors that have the same highest number of neighbors, the node randomly selects one of them and appoints it as an SN.
- 4) After node DE receives the SSI message from NL (see Fig. 2d), it appoints itself as an SN and responds to NL 's SSI message with a Supernode Selection Data (SSD) message. When node NL receives the SSD message, it sets its SN face ID to the face ID over which the SSD message was received from node DE .

The proposed DS construction algorithm requires each node to communicate with its neighbors. If X is the average node degree of a graph G with N nodes, the message complexity of the DS construction algorithm over G will not exceed $O(N)$.

B. Connected Dominating Set Construction

After running our DS construction algorithm (Section III-A), we obtain a DS, where every node is either a dominator, in our case an SN, or a dominated node, which we call hereafter a regular node. This property assures, that the distance between any two SNs does not exceed three hops,

which means that the maximum distance for a message from an SN to reach at least another SN is also three hops. To construct a CDS, we let every SN to know about its nearest SNs. Using this knowledge, each SN decides the nodes to use for connecting the DS so that it will become a CDS (the nodes in between the nearest SNs). Our protocol does not permit the messages sent for CDS construction to travel more than three hops. Therefore, the message complexity of CDS construction over a graph G with N nodes does not exceed $O(N)$. In addition to the parameters *snFlag* and *snFaceId* mentioned in Section III-A, our CDS construction algorithm enables all SNs to also store the following parameters:

- **cFlag** is a flag that indicates whether the associated SN has already run the proposed CDS construction algorithm or not.
- **snFaceList** is a list containing the face ID(s) over which the associated SN can reach other SNs.

We show a small topology, in Fig. 3a, to describe how the proposed CDS construction algorithm connects the DS previously constructed by our DS construction algorithm. For the sake of simplicity of presentation, we only show the messages sent by SN_1 , but similar operations are performed in parallel for each SN.

- 1) The algorithm starts with SN_1 sending over each face i a Supernode Connection Interest (SNCI) message with name $/SNCI/SN1/i$ to all of its neighbors, as shown in Fig. 3b.
- 2) In Fig. 3c, the neighbors of SN_1 proceed with forwarding the received SNCI messages, until another SN is reached.
- 3) In Fig. 3d, when the SNCI message of SN_1 reaches SN_2 , SN_2 replies with a Supernode Connection Data (SNCD) message. Then, SN_2 drops any other received

SNCI messages issued by SN_1 , in this case, the one that is received from node N_5 because SN_2 has already replied to SN_1 's SNCI message before.

- 4) Every node that gets an SNCD message further forwards it, until it reaches the issuing SN. Furthermore, in Fig. 3e, as the SNCD message has reached node N_8 , N_8 appoints itself as an SN and stores the face through which the SNCD message was received in its *snFaceList*.
- 5) In Fig. 3f, when SN_1 receives the SNCD message, it sets its *cFlag* to *true* status and stores the face over which the SNCD message was received in its *snFaceList*. SN_1 and SN_2 are now connected through N_8 .

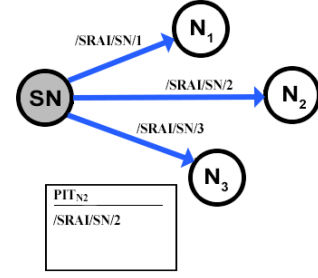
We have to maintain DS and CDS in the following cases: 1) when a new node joins the network, or 2) when a node or a link fails. If a new node joins the network, the maintenance of DS or CDS is quite simple. The new node simply requires to run the DS and CDS construction algorithms described in Sections III-A and III-B to join the DS or the CDS, respectively. To maintain DS and CDS when failures happen, we consider two cases: 1) regular failures (a regular node or a link connecting two regular nodes fails), 2) SN failures (an SN or a link connected to an SN fails). When regular failures happen, if the entire graph is still connected, the DS or CDS will not change. Nevertheless, if an SN or a link connected to an SN fails, the node (or nodes) that were connected to the SN need to be aware of this failure so that they run again the DS and CDS construction algorithms (Sections III-A and III-B).

So far, we have described distributed algorithms for DS and CDS construction. The proposed DS construction algorithm requires fewer message exchanges than our CDS construction algorithm. However, when the network topology is clustered using a DS, SNs are not directly connected. Hence, it requires multi-hop communication through regular nodes so that SNs can reach each other. Therefore, the intuition is that DS-based inter-domain routing requires more bandwidth overhead than CDS-based inter-domain routing.

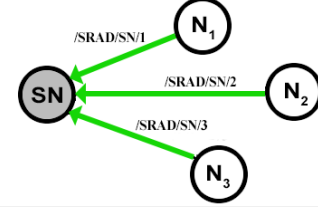
For routing, we consider two cases in Sections IV and V, respectively: 1) L-SCN selects SNs using our DS construction algorithm described in Section III-A, and 2) L-SCN selects SNs using our CDS construction algorithm described in Section III-B.

IV. ROUTING IN A DOMINATING SET

The proposed DS and CDS-based clustering algorithms, presented in Section III, complement L-SCN by efficiently clustering the network topology, which is a requirement of L-SCN. In this Section, we focus on intra-domain and inter-domain routing. At each domain, the associated SN is responsible for both intra-domain and inter-domain routing. Intra-domain routing consists of routing service requests towards service provider(s), which are in the same domain, while inter-domain routing means routing a service request from a domain towards service provider(s) in another domain. Therefore, before routing, each SN needs to know the services and resources provided in its domain. According to this information, an SN decides whether a requested service could be provided



(a) SRAI packet broadcast to pull available services and resources.



CS _{SN}	
/SRAD/SN/1	- BF: 1 1 0 1 0 1 0 1 0 0 1 1 1 0 1 0 1 0 0 0
/SRAD/SN/2	- BF: 1 0 0 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0
/SRAD/SN/3	- BF: 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1 1

(b) SRAD packet forward by regular nodes containing BF and resource information.

Fig. 4: Pulling service and resource availability.

TABLE II: SRAD structure

Timestamp	Issuance time
Available service names	BF bit vector, salt count value.
Available resources	Available CPU, GPU, RAM,

within the domain or it requires inter-domain routing. Note that intra-domain routing protocol is identical for both DS-based and CDS-based clustered network topologies. However, inter-domain routing is different for DS-based and CDS-based clustered networks.

A. Service and Resource Discovery

To acquire knowledge about the provided services and resources (e.g., CPU, RAM), each SN asks the nodes in its domain about their available services and resources using the pull mechanism that we illustrate in Fig. 4. Fig 4a shows that the SN sends over each face i a Service and Resource Availability Interest (SRAI) message with name $/SRAI/SN/i$ to pull information about available services and resources. Each regular node (N_1 , N_2 , or N_3) receives an SRAI message and stores in its PIT this message as well as the information about the reception face for it (see Fig. 4a). Then, each regular node replies with a Service and Resource Availability Data (SRAD) message, with the same name of its corresponding SRAI message that carries a BF containing its available service names as well as a field containing information about its available processing and memory resources. Table II shows the structure of an SRAD message, which includes the issuance time of the SRAD message, a bit vector and a salt count value

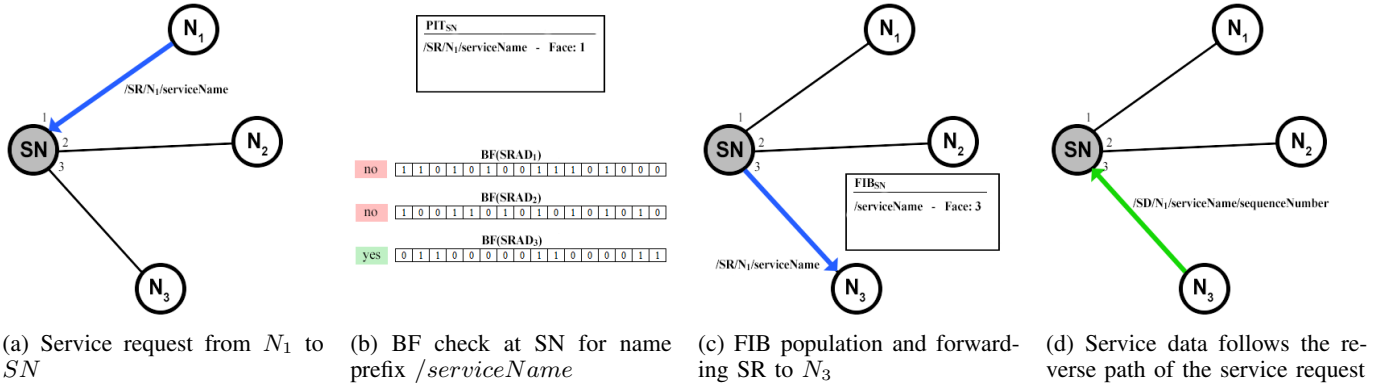


Fig. 5: Intra-domain routing

that are needed to retrieve the BF containing available service names, and available processing and memory resources.

As Fig. 4b shows, the SN stores the received SRAD messages as well as the information about their reception faces in its content store.

B. Intra-Domain Routing

Fig. 5 describes FIB population and intra-domain routing. When node N_1 requires a service, it sends a Service Request (SR) message SR_1 with name $/SR/N_1/serviceName$ to the SN (see Fig. 5a). When SN receives message SR_1 , it stores this message in the PIT and checks the name prefix $/serviceName$ (last name component of message SR_1) against all the BFs of the stored SRAD messages (see Fig. 5b). Since only the BF of $SRAD_3$ claims to contain name prefix $/serviceName$, SN assigns in the FIB face ID 3 as the next hop face ID for $/serviceName$ and forwards message SR_1 over this face towards node N_3 (Fig. 5c). If multiple BFs claim to contain a service name, SN checks the associated SRAD messages for resource availability and forwards SR_1 to the node with highest available resources. When node N_3 receives message SR_1 , it replies with Service Data (SD) message(s) with name(s) $/SD/N_3/serviceName/sequenceNumber$ if it tends to provide the demanded service. Otherwise, node N_3 sends a Service Provision Refusal (SPR) message with name $/refusal/N_3/serviceName$ to the SN. If SN receives such an SPR message from N_3 , it removes the FIB entry for name prefix $/serviceName$ and checks whether another regular node is available in the domain to provide the demanded service. If no node in the domain provides a certain service, the requested service requires inter-domain routing to be retrieved. In Fig. 5b, when SN checks the BFs of its stored SRAD messages to reply to SR_1 , a false positive error might happen. For example, if a false positive error happens at $BF(SRAD_1)$, SN forwards message SR_1 to N_1 . Since N_1 does not provide the service requested by message SR_1 , N_1 returns an SPR message to deny the provision of the requested service. Nevertheless, false negative errors are impossible using BFs. Thus, SN will anyway forward message SR_1 towards the correct service provider, i.e., node N_3 . Therefore, the proposed intra-domain routing mechanism is robust to false positive errors.

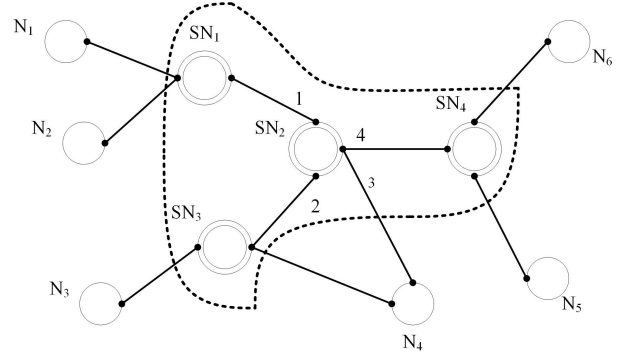


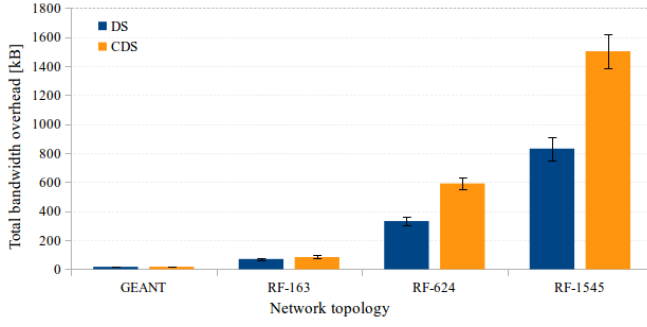
Fig. 6: Inter-domain routing for CDS-based clustering.

C. Inter-Domain Routing for DS-based Clustering

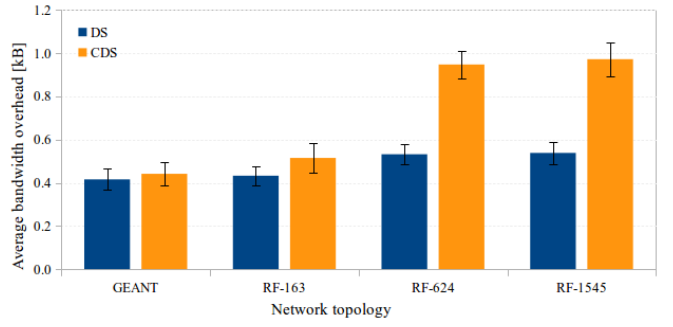
If an SN receives an SR message, it first checks the name of the SR message against all the BFs of the SRAD messages stored in its CS table. If one of the BFs contains the name of the SR message, this message is routed inside the domain (intra-domain routing). Otherwise, the SR message is flooded until it reaches another SN (As explained in Section III, the distance between two SNs in a DS does not exceed three hops). When another SN receives the SR message, it checks whether the service requested by message SR is provided in its domain. If the requested service is not available, the SN again floods the SR message until it reaches an SN that its domain provides the requested service. In Section VI, we will observe that, due to these SR message floodings, DS-based routing requires much more bandwidth overhead for routing SR messages compared to CDS-based routing.

V. ROUTING IN A CONNECTED DOMINATING SET

When CDS is used for clustering, SNs are directly connected. Therefore, inter-domain routing is different than DS-based routing. Fig. 6 shows an example CDS of four SNs SN_1, SN_2, SN_3 and SN_4 inside the dotted area. We assume that each SN records the face(s) over which it is connected to other SNs in a vector called $snFaceList$. For example, $snFaceList(SN_2) = 1, 2, 4$. In Fig. 6, when SN SN_2 receives service request message SR with name SRN from node N_4 , which is asking for a service not provided in the domain of



(a) Total bandwidth overhead for clustering algorithms



(b) Average transmitted bytes per node for clustering algorithms

Fig. 7: Results in terms of bandwidth overhead for DS and CDS construction

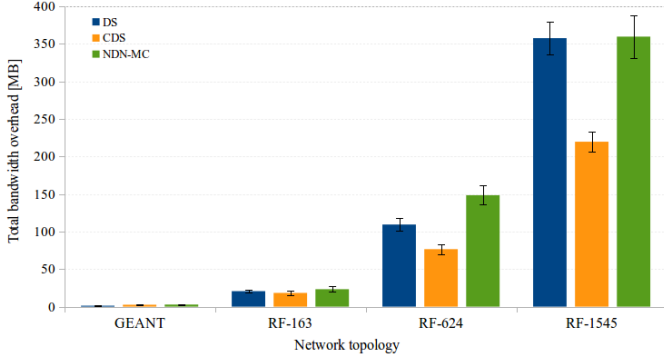


Fig. 8: Total bandwidth overhead for service request routing

SN SN_2 , this SN checks its FIB for name SRN . If no FIB entry is found, SN_2 forwards message SR over all the faces 1, 2 and 4 that exist in its $snFaceList$. Later, if SN_2 receives an SD message with name SRN over face 2, it populates the FIB for name SRN with next hop face 2. This FIB population helps SN_2 not to forward future service requests with name SRN over all the faces of its $snFaceList$.

VI. PERFORMANCE EVALUATION

To evaluate the performance of our protocols, we have implemented them in *ndnSIM* [20]. We compare our protocols with NDN *multicast* forwarding strategy [21]. Our simulation code is available online at [22]

A. Settings

For performance evaluation, we use GEANT [8], and Rocketfuel topologies [9]. GEANT topology consists of 40 routers. From Rocketfuel topology traces [9], we use three topologies with sizes 163, 624, and 1545 routers called RF-163, RF-624, and RF-1545, respectively. We place 10 service providers and 20 clients randomly in all the topologies. We assume that service providers process the received service requests and the processing time of a service request is uniformly distributed between 100 and 2000 milliseconds. If a client sends an SR message at time t_1 and receives the processed SD at time t_2 , the client considers $t_2 - t_1$ the *service retrieval time*.

Similar to [5], we assume that SNs pull the available services and resources every 10 seconds, while clients request services using a random function with exponential distribution with the mean value of 2 seconds. We assume that service universe (i.e., the set of all available services) consists of 1000 unique services. The results are averaged over ten simulations and the reported mean values have 95% confidence intervals.

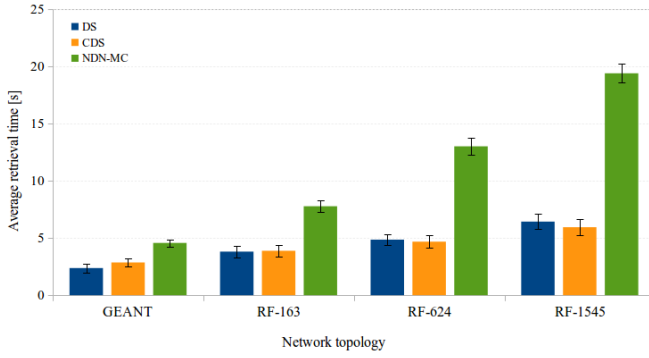
In the following, we analyze the performance of our protocols according to three metrics: 1) the required bandwidth overhead for clustering, 2) the required bandwidth overhead for routing, and 3) average service retrieval time.

B. Bandwidth Overhead of Clustering

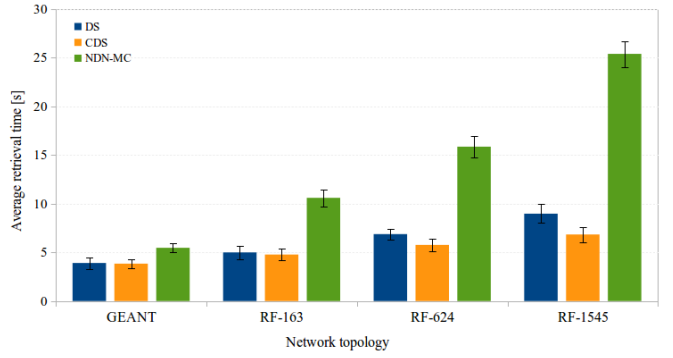
In Fig. 7a, we show the total number of Interest messages sent for DS and CDS construction. From this evaluation, we can see that when topology size (i.e., the number of nodes) increases, the number of messages exchanged for DS and CDS construction also increases. Further, Fig. 7a makes clear that for all of the considered topologies, the proposed CDS construction algorithm requires more bandwidth overhead than the proposed DS construction algorithm because it builds on top of the DS construction algorithm. Nevertheless, from Fig. 7a we observe that for RF-624 and RF-1545, the proposed CDS construction algorithm requires more than twice the bandwidth overhead that the DS construction algorithm requires. This result confirms our intuition that when topology size increases, the required bandwidth overhead for CDS construction compared to that needed for DS construction also increases. In Fig. 7b, we present the average number of Interest messages sent for DS and CDS construction per node. In Figs 7a and 7b, if we focus on each of the considered topologies separately and compare DS-based and CDS-based clustering algorithms, we can note the maximum difference between the orange and blue curves if RF-624 is used. This is an important observation, which confirms the impact of topology structure on the complexities of DS and CDS construction algorithms in terms of the required bandwidth overhead.

C. Bandwidth Overhead of Routing

In Fig. 8, we compare our DS-based and CDS-based routing protocols with NDN *multicast* forwarding strategy [21] in



(a) Average service retrieval time with full link capacities



(b) Average service retrieval time with reduced link capacities

Fig. 9: Results in terms of service retrieval time.

terms of bandwidth overhead for service request routing. For all the considered topologies, we expected less overhead for service request routing using a CDS. However, interestingly, we observe from Fig. 8 that for GEANT topology, we actually see that DS requires less overhead for service request routing. This result confirms the higher complexity of CDS-based routing compared to DS-based routing in terms of bandwidth overhead, when small topologies are used. For larger topologies, the results are as expected and the CDS-based routing requires much less bandwidth overhead for routing service requests. The CDS-based routing protocol requires up to 39% less overhead to route service request messages in the larger topologies.

D. Average Service Retrieval Time

In Fig. 9, we show results in terms of average service retrieval time for service provision. From this figure we can see that NDN MultiCast (NDN-MC) forwarding strategy has always significantly higher average service retrieval time than both DS and CDS-based routing. The reason is that NDN-MC floods SR messages, which leads to overloading many service providers. Consequently, when NDN-MC strategy is employed, many SR messages are queued at overloaded service providers until the service providers become idle to process them. This results in a 69% shorter service retrieval time for the proposed CDS-based routing approach compared to NDN-MC. For small topologies, like GEANT, DS-based routing performs around 17% faster than CDS-based routing, while for large topologies, like RF-1545, CDS-based routing performs 8% faster than DS-based routing. From Fig. 9b, we observe the impact of restricted link capacities (i.e., when all the links have only 10% of their original capacities) on average service retrieval time. This figure makes clear that the bandwidth-hungry nature of NDN-MC forwarding strategy has considerable impact on average service retrieval time. Overall, we observe similar performance for DS and CDS-based routing in terms of average service retrieval time, while CDS-based routing performs slightly better than DS-based routing when there are tight bandwidth resources, which is because of higher bandwidth requirement for DS-based routing. Further, with reduced link capacities, we note that

the service retrieval takes up to 73% less time with our CDS-based routing than NDN-MC. CDS-based routing is also up to 24% faster than DS-based routing with reduced link capacities.

VII. CONCLUSIONS

In this paper, we studied intra-domain and inter-domain routing of service requests in SCNs. We proposed fully distributed algorithms for constructing DS and CDS to select SNs in the network topology that are responsible for managing intra and inter-domain communications. Next, we leveraged SNs to implement intra and inter-domain service and resource discovery for SCNs. For service discovery, we used BF-based and resource-aware intra-domain and inter-domain routing protocols to route service requests.

We assessed the performance of our routing protocols for various topologies of different sizes. From the results, we observed that with larger topologies, DS construction requires much less bandwidth than CDS construction. We compared the proposed DS-based and CDS-based routing protocols with NDN-MC. The results showed that DS-based routing protocol requires less bandwidth overhead for routing service requests than NDN-MC forwarding strategy, while CDS-based routing protocol requires much less bandwidth overhead for service request routing compared to DS-based routing protocol. Finally, we observed that both DS-based and CDS-based routing protocols outperform NDN-MC forwarding strategy in terms of service retrieval time. To the best of our knowledge, this paper is the first work that leverages DS and CDS concepts for BF-based and resource-aware intra and inter-domain routing in SCNs. Our future work includes complete complexity analysis of the proposed algorithms, further comparative analysis with other than DS/CDS routing protocols, and measuring CPU/memory resources that network nodes consume to run our algorithms using a real testbed implementation.

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] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," in *Proc. of the 5th ACM Int. Conf. on Emerging Networking Experiments and Technologies*, ser. CoNEXT '09, New York, NY, USA, 2009.
- [3] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, Jul. 2014.
- [4] T. Braun, V. Hilt, M. Hofmann, I. Rimac, M. Steiner, and M. Varvello, "Service-Centric Networking," in *Proc. of IEEE Int. Conf. on Communications Workshops (ICC), Kyoto, Japan, Jun. 5-9, 2011*.
- [5] M. Gasparyan, T. Braun, and E. Schiller, "L-SCN: Layered SCN Architecture with Supernodes and Bloom Filters," in *Proc. of 14th IEEE Consumer Communications and Networking Conf. (CCNC), Las Vegas, NV, USA, Jan. 8-11, 2017*.
- [6] R. Laskar and H. Walikar, "On domination related concepts in graph theory," in *Combinatorics and graph theory*. Springer, 1981, pp. 308–320.
- [7] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.
- [8] "The GEANT network, 2012," <http://www.topology-zoo.org/dataset.html>, accessed: 2016-07-25.
- [9] "Rocketfuel: An ISP topology mapping engine," <https://research.cs.washington.edu/networking/rocketfuel/>, accessed: 2019-07-20.
- [10] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, "Named Data Networking (NDN) project," *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, vol. 157, p. 158, 2010.
- [11] 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 Int. IFIP Networking Conf., Stockholm, Sweden, Jun. 12-16, 2017*.
- [12] A. Marandi, M. F. Imani, and K. Salamatian, "Practical Bloom Filter-based Epidemic Forwarding and Congestion Control in DTNs: A Comparative Analysis," *Computer Communications*, vol. 48, pp. 98–110, 2014.
- [13] A. Marandi, T. Braun, K. Salamatian, and N. Thomos, "Pull-based Bloom Filter-based Routing for Information-Centric Networks," in *Proc. of the IEEE Consumer Communications and Networking Conf. (CCNC), Las Vegas, NV, USA, Jan. 11-14, 2019*.
- [14] —, "A Comparative Analysis of Bloom Filter-based Routing Protocols for Information-Centric Networks," in *Proc. of IEEE Symp. on Computers and Communications, ISCC, Natal, Brazil, Jun. 25-28, 2018*.
- [15] M. Cardei, M. X. Cheng, X. Cheng, and D.-Z. Du, "Connected Domination in Multihop Ad hoc Wireless Networks," in *Proc. of the Joint Conference on Information Sciences 6*, 2002.
- [16] K. M. Alzoubi, P.-J. Wan, and O. Frieder, "Message-Optimal Connected Dominating Sets in Mobile Ad-hoc Networks," in *Proc. of the 3rd ACM int. symp. on Mobile ad hoc networking & computing, Lausanne, Switzerland, Jun. 9-11, 2002*, pp. 157–164.
- [17] Y. L. S. Zhu and M. T. D.-Z. Du, "Localized Construction of Connected Dominating Set in Wireless Networks," in *Proc. of US National Science Foundation Int. Workshop Theoretical Aspects of Wireless Ad Hoc, Sensor and Peer-to-Peer Networks*, 2004.
- [18] C. Lenzen, Y.-A. Pignolet, and R. Wattenhofer, "Distributed Minimum Dominating Set Approximations in Restricted Families of Graphs," *Distributed Computing*, vol. 26, no. 2, pp. 119–137, Apr 2013.
- [19] Y. Xu, Y. Li, T. Lin, G. Zhang, Z. Wang, and S. Ci, "A Dominating-Set-based Collaborative Caching with Request Routing in Content-Centric Networking," *Proc. of IEEE Int. Conf. on Communications, Hungary, Budapest, Jun. 9-13, 2013*.
- [20] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2.0: A new version of the NDN simulator for NS-3," NDN, Technical Report NDN-0028, January 2015.
- [21] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moiseenko, Y. Yu, W. Shang, Y. Huang, J. P. Abraham, S. DiBenedetto *et al.*, "NFD developer's guide," *Dept. Comput. Sci., Univ. California, Los Angeles, Los Angeles, CA, USA, Tech. Rep. NDN-0021*, 2014.
- [22] "Simulation code," <https://github.com/javinator/DS-SCN>.