UNIVERSITÄT BERN

BACHELOR THESIS

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

Author:
Vincent Hofer

Supervising Assistant:
Ali Marandi

Supervisor:
Prof. Dr. Torsten Braun

Communication and Distributed Systems
Institute of Computer Science

December 11, 2019

Abstract

A service-centric network requires a routing protocol that routes 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. 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. In order to appoint appropriate nodes as supernodes in the network topology, in this thesis, we use Dominating Sets (DS) and Connected Dominating Sets (CDS).

A DS is a subset of a graph, where each element of the graph is either in the subset or directly adjacent to an element of the subset. A CDS is a DS, where all elements of the subset are connected. We propose fully distributed algorithms for constructing DS as well as CDS over the network topology.

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.

Contents

List of Figures

List of Tables

List of Abbreviations

SN	**S**uper**n**ode
DS	**D**ominating **S**et
CDS	**C**onnected **D**ominating **S**et
BF	**B**loom **F**ilter
ICN	**I**nformation-**C**entric **N**etworking
SCN	**S**ervice-**C**entric **N**etworking
NDN	**N**amed **D**ata **N**etwork
PIT	**P**ending **I**nterest **T**able
FIB	**F**orwarding **I**nformation **B**ase
CS	**C**ontent **S**tore
CII	**C**lustering **I**intializing **I**nterest
CD	**C**lustering **D**ata
SSI	**S**upernode **S**election **I**nterest
SSD	**S**upernode **S**election **D**ata
SNCI	**S**upernode **C**onnection **I**nterest
SNCD	**S**upernode **C**onnection **D**ata
SRAI	**S**ervice and **R**esource **A**vailabilty **I**nterest
SRAD	**S**ervice and **R**esource **A**vailabilty **D**ata
SR	**S**ervice **R**equest
SD	**S**ervice **D**ata
SPR	**S**ervice **P**rovision **R**efusal

Chapter 1

Introduction

1.1 Motivation

The Internet was initially designed as a communication network. The nature of TCP/IP is carrying data between two endpoints. However, nowadays users see the Internet as a content distribution network. Thus, users do not care about the location of the content objects. This is where the network paradigm Information-Centric Networking (ICN) [1] comes into play. The goal of ICN is to shift from host-centric to content-centric networking to better accommodate content distribution, meaning, the consumer does not need to know 'where' but rather 'what' he wants to retrieve. The benefits of ICN have been discussed extensively in other publications [1, 2, 3]. Apart from content retrieveal, users might demand processed content objects, i.e., services, such as calculations, video transcoding, Google Maps directions, etc. Therefore, in [4], Service-Centric Networking (SCN) is proposed to extend ICN so that it supports service retrieval.

The goal of this thesis was to implement a Layered SCN Architecture (L-SCN) [5], where the network is divided into domains. Each domain is managed by a Super Node (SN) that is aware of the provided services and the available resources in the domain. Non SNs are called regular nodes. The regular nodes of each domain are managed by the SN of the domain.

1.2 Research Questions

This thesis tries to solve the following questions:

- Can the L-SCN architecture be implemented on a network where SNs have not been elected already?
- What is the impact of using either a DS or a CDS?
- What are the benefits of using the L-SCN architecture compared to other routing strategies?

1.3 Contribution

L-SCN [5] assumes that nodes are clustered and SNs are selected. To implement this architecture and assess its performance, we first needed an algorithm to divide any network topology into domains and elect SNs, as the design of clustering algorithms was mentioned as future work in [5].

In the literature, Dominating Sets (DS) and Connected Dominating Sets (CDS) [6] have been used to create virtual backbone for routing in ad-hoc networks [7, 8, 9, 10, 11]. However, all these works model the network as unit-disk graphs, which already brings some constraints. We wanted the clustering algorithm to be able to work on arbitrary network topologies. Therefore, we first developed an algorithm electing SNs based on a Dominating Set (DS). As the work in [5] uses connected

SNs, we then developed an algorithm, which would connect the created DS into a Connected Dominating Set (CDS).

We then implemented two variations of the L-SCN [5] architecture, using a DS and a CDS as a virtual backbone for routing, respectively, and implemented Bloom Filter (BF)-based routing for both architectures in ndnSIM [12]. We compared the proposed protocols with the NDN multicast strategy [13] with different parameters and different real world topologies, namely GEANT [14] and Rocketfuel [15]. We assessed the performance and observed 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 [16], and 4) the CDS-based routing 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.

This thesis is structured as follows. In chapter 2, the related work and theoretical background are presented. In chapter 3, we discuss the clustering algorithms. Chapter 4 discusses the routing of services in our architectures. In chapter 5, we evaluate the performance of the implemented architectures. The final chapter 6 concludes the work and summarizes the thesis.

Chapter 2

Preliminaries and Related Works

2.1 Related Work

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 Network (CCN) [2] is the first 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, and there are two types of messages namely *Interest* and *Data* messages that 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. CCN routers can store copies of passing 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 cache 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 algorithms to cluster network nodes and to choose the SNs. To cope with this issue, in this thesis, 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 have been 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) [17] to compress the sets of their available service names saving significant storage resources.

The FUSION project [18] introduces another approach to service-oriented networking which consists of three different layers. The lowest layer corresponds to the underlying low level IP routing using traditional end-to-end protocols. The middle layer is the service router layer, which is responsible to forward requests from clients to the service instances. Finally, the top layer is the execution plane, which is responsible for service instances to run. Moreover, the work in [18] divides the service orchestration and resource management into two layers, to achieve better scalability. Also, it introduces session slots to achieve service delivery and adds preemptive reservation mechanism for those slots.

The work in [19] discusses the division of Machine-to-Machine services in two categories: a centralized approach (Orchestration), where a single application collects data and sends orders, and a distributed approach (Choreography), in which multiple applications offer and use services for collaboration. For the evaluation of the architecture it considers wireless sensor and actuator networks. The evaluation shows the improvement offered by the decentralized Choreography compared to centralized approaches.

Further benefits and drawbacks of Orchestration and Choreography are discussed in [20] and a hybrid architecture combining the benefits of the above architectures is proposed. [21] discusses edge computing, a paradigm where resources are not offloaded on the cloud, but rather at the edge of the network, close to end-devices, to reduce communication latency and bandwidth demand to distant data centers. [21] compares their proposed paradigm with ICNs and shows, that ICN is limited to non-interactive content and that, on the other hand, edge computing allows service-oriented networking. Finally, [21] presents an experimental evaluation of an edge computing implementation represented by mobile gaming.

General Virtual Networking (GVN) [22] introduces a new framework enabling simple implementations for SCN and other related architectures. This scheme is based on Software Defined Networking (SDN), thus this framework adds a layer between the Network and Transport layers. Using the GVN architecture, an end-host can either process the additional information provided by the GVN header, or simply use the usual IP destination addresses, especially if the router is not GVN-enabled.

2.2 Bloom Filters

A Bloom Filter (BF) [17] is a data structure that is used to compactly represent a set. BFs have been used in networking [23, 24, 25] 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 [24]. While using BFs, false positive reports are possible, while false negative reports cannot happen as shown in [17]. 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. This trade-off is shown in Equation (2.1)

$$m = -\frac{nln(p)}{(ln(2))^2}, k = \frac{m}{n}ln(2). \tag{2.1}$$

BF-based routing have been studied in several works [24, 26, 25]. Push-based BFR [24] 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 [25]. 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 [25]. This strategy reduces significantly the required bandwidth and storage overhead for BF-based content advertisements.

2.3 Dominating Sets

For a graph $G =< V, E >$, 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. On a given graph there are many different possible DSs. 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, meaning the minimum possible number of elements, while still remaining a DS, 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. 2.1, we illustrate examples of DS, MDS, CDS, and MCDS over a graph. The determination of MDS and MCDS are NP-hard problems [7].

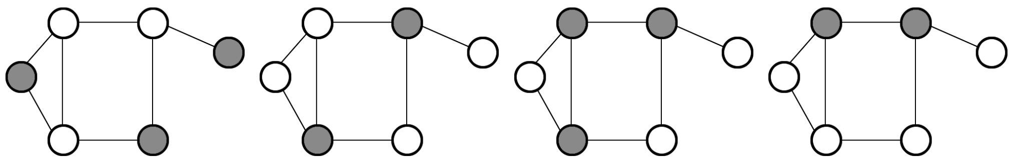

(a) DS (b) MDS (c) CDS (d) MCDS

FIGURE 2.1: Different Dominating Sets; Dominator nodes are grey, Dominated nodes are white

DS and CDS find applications in several networking problems, e.g., creating a virtual backbone for routing in ad-hoc networks [7, 8, 9, 10, 11]. As finding an MCDS is NP-hard, the work in [7] proposed to approximate MCDSs, 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 [7] 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 [8] 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 [9], 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 [10] examines distributed algorithms proposed for MCDS approximations and presents distributed construction of an approximate MCDS, for unit disk graphs.

The work in [11] uses DSs to provide a solution for collaborative caching in ICN. It leverages DSs for efficient collaborative caching to reduce caching redundancy in the network. It combines routing and caching strategies with a CDS. In [11], the most popular content objects are cached at the core routers. The benefit of having a CDS for the core routers is much simpler routing, 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 [11] is the only one 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 problem with the work in [11] 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.

In [7, 8, 9, 10], the presented solutions are for wireless ad-hoc networks, modeled as unit disk graphs. Differently from these works, in this thesis, we focus on using DS and CDS for intra-domain and inter-domain routing and propose a fully distributed algorithms to construct these sets for any arbitrary network topology.

2.4 ndnSIM Overview

ndnSIM [12] is an NDN simulator, based on ns-3, which is an open-source, discrete-event network simulator. Most of the code is written in C++. ndnSIM uses applications, which can be *installed* on nodes. Like NDN [3], ndnSIM uses two different types of messages, *Interest* and *Data* messages, for the communication between applications. The two main categories of application are *Consumer* and *Producer* applications. Consumer applications periodically send Interest messages, while producer applications return Data messages upon receiving Interest messages. The Named Data Networking Forwarder (NFD) implements the forwarding daemon, which processes Interest and Data messages, checks and populates the Pending Interest Tables (PIT), the Forwarding Interest Bases (FIB) and the CS.

Interest messages carry random nonces to prevent loops in the routing. Data messages follow the trail of the corresponding Interest message, therefore also avoiding loops. Incoming interest messages are stored in the PITs. These PIT entries are deleted as soon as the corresponding Data message reaches the node. ndnSIM also differentiate between so-called networking faces, the interface between two nodes, and application or local faces, the interface between a node and the application installed on it.

Chapter 3

Clustering

In this thesis, 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.

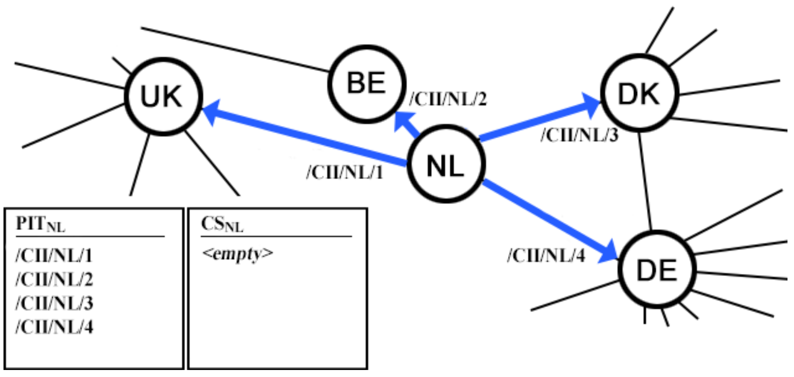
3.1 Dominating Set Construction

To describe our algorithm, we use a part of the GEANT topology [14] illustrated in Fig. 3.1. In this topology, nodes represent core routers located at an European country. In Fig. 3.1, we focus on the node with label NL that represents Netherlands and show the message exchanges needed for constructing a DS. Interest messages are shown with blue arrows and Data messages with green arrows. We only show the messages sent by NL, but the algorithm runs in parallel for each node.

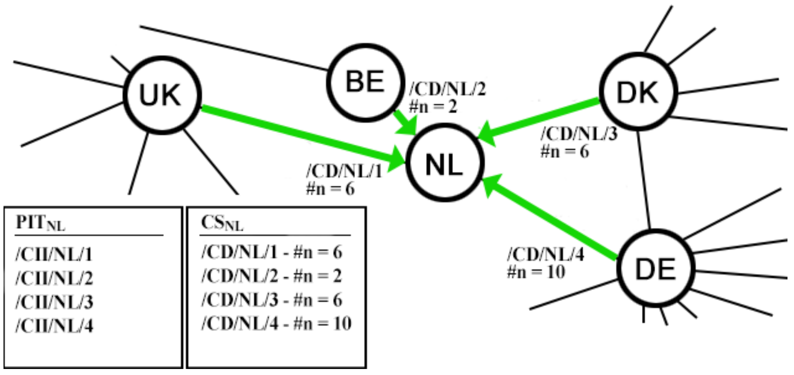
1. The algorithm starts in Fig. 3.1a with node NL sending Clustering Initializing Interest (CII) with name $/CII/NL/faceID$ over each of its faces to ask each neighbour to send back the number of its neighbours.
2. In Fig. 3.1b, we show that when NL's neighbours receive a CII message, they respond with a Clustering Data (CD) message with the same name containing the number of its neighbours 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 table. Node NL waits for a short time interval t_w to receive CD messages from all its neighbours. Then, node NL compares the number of its own neighbours with that reported by its neighbours.
3. Then, as Fig. 3.1c shows, node NL sends a Supernode Selection Interest (SSI) message with name $/SSI/NL$ to its neighbour with the highest number of neighbours, which is node DE. If a node has multiple neighbours that have the same highest number of neighbours, the node randomly selects one of them and appoints it as an SN. (Note that if node NL has more neighbours than all of its neighbours, it appoints itself as an SN.)
4. The final step for DS construction is shown in Fig. 3.1d. After node DE receives the SSI message from NL, 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.

Additionally to the standard ndnSIM implementation, our DS construction algorithm enables the nodes to store the following parameters:

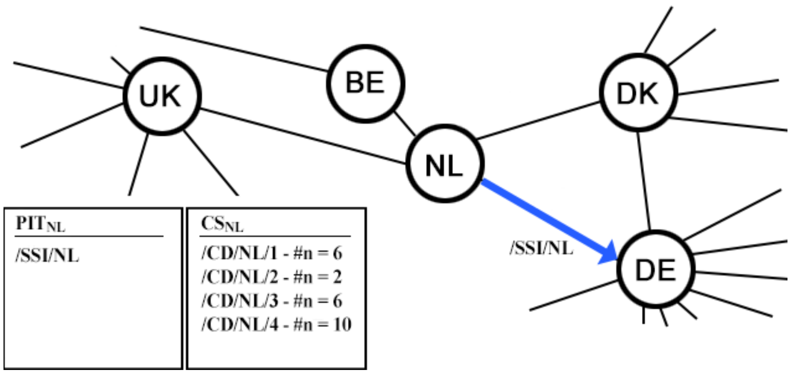
- **snFlag:** a flag that indicates whether a node is an SN or not.
- **snFaceId:** the face ID of the face over which the SN of the associated domain can be reached.



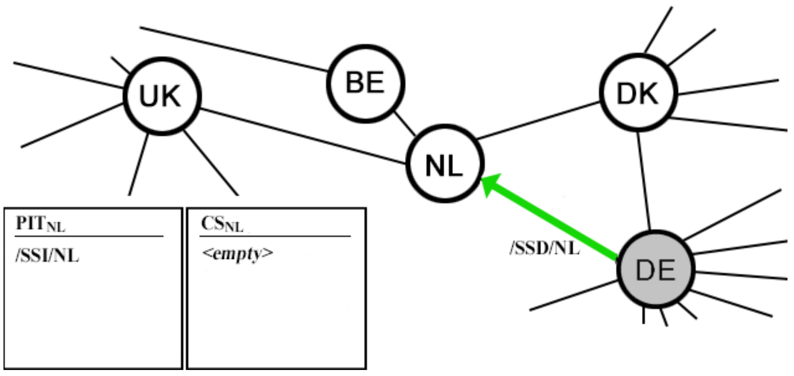
(a) Node NL sends a CII message to its neighbours.



(b) Each neighbour replies with a CD message.



(c) Node NL sends an SSI message to node DE.



(d) Node DE acknowledges node NL with an SSD message.

FIGURE 3.1: The clustering algorithm described with a part of the GEANT network

For clustering the nodes in the network topology we needed to be able to process some data at the nodes. We decided to use consumer and producer applications installed on every node to perform the clustering. As the consumer applications are initialized, they start to send CII messages to all their neighbours as shown earlier. The structure of these Interest messages are shown in Table 3.1. The CII message is then forwarded to the clustering producer apps of its neighbouring nodes.

TABLE 3.1: CII structure

Name	Interest Name
Nonce	Random number to assure uniqueness of the Interest message

In the producer app, the number of neighbouring nodes of the node, on which the producer is installed, is written in the Clustering Data (CD) message, shown in Table 3.2. If a CD message arrives at a node, through a networking face, this face through which it was received is written in the CS along with the CD message, to know, over which face the Supernode Selection Interest (SSI) message will have to be forwarded. After the CDs for all the CIIs have been received by the consumer application, or after a short time interval t_w, it searches through the CS to find the entry with the highest number of neighbours. There are two possibilities: 1) The node on which the producer app is installed has the highest number of neighbours amongst its neighbouring nodes. In this case, it sets its *snFlag* to *true* and installs a new Supernode (SN) application on the node. 2) Another node has the highest number of neighbours. Then, the producer app sends an SSI message to this candidate.

TABLE 3.2: CD structure

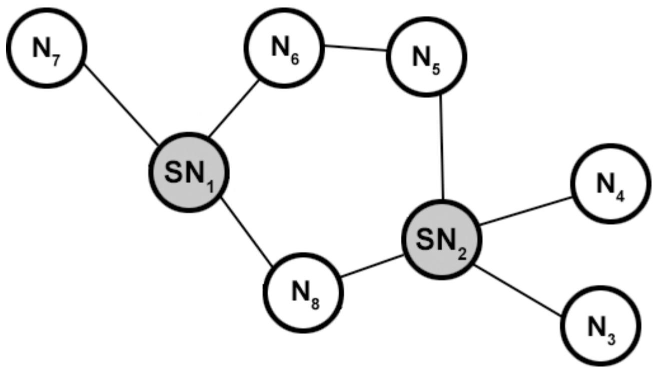
Name	Content Name
Signature	Signature of the issuing Producer application
Neighbours	Number of neighbours of the node
Face	Networking face over which the node is reachable

If a consumer application receives an SSI message, it sets its *snFlag* in the node to *true*, installs a new SN application and then sends a Supernode Selection Data (SSD) message to the node who issued the SSI message. Then, when the consumer application gets the SSD message, it sets the *snFaceId* to the face over which the SSD message was received. We now have a DS.

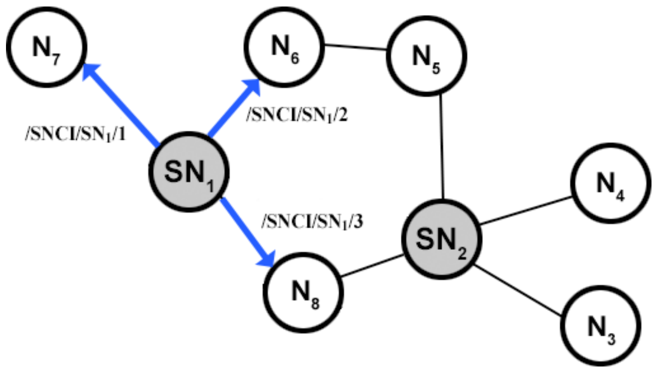
3.2 Connected Dominating Set Construction

After running our DS construction algorithm (Section 3.1), we have a DS, thus, 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. Therefore, the maximum distance for a message from an SN to reach at least another SN is 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. Hence, our CDS construction algorithm provides connectivity of SNs in a localized and distributed manner.

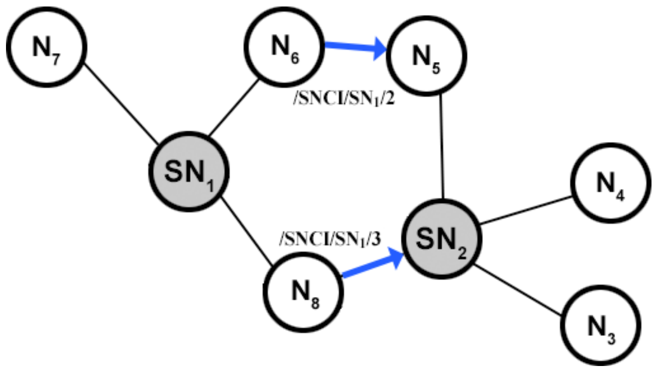
We show a small topology, illustrated in Fig. 3.2 and 3.3, to describe how the proposed CDS construction algorithm connects the DS previously constructed by our DS construction algorithm. For the sake of simplicity of presentation, we only show the messages sent by SN_1, but similar operations are applied in parallel for each SN.



(a) Previously clustered topology



(b) SN_1 sends SNCI messages to all its neighbours.



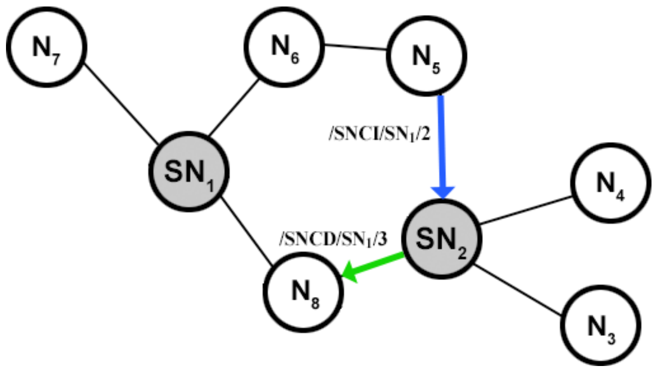
(c) Forwarding of the received SNCI messages.

FIGURE 3.2: The CDS construction algorithm

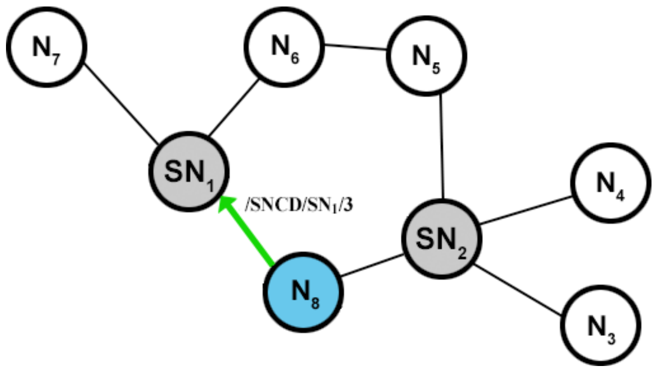
1. The algorithm starts on a previously constructed DS as shown in Fig. 3.2a.
2. In Fig. 3.2b SN_1 sends a so-called Supernode Connection Interest (SNCI) message with name $/SNCI/SN1/i$ over each face i to all of its neighbours.
3. In Fig. 3.2c, the neighbours of SN_1 proceed with forwarding the received SNCI messages, until another SN is reached.
4. In Fig. 3.3a, 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.

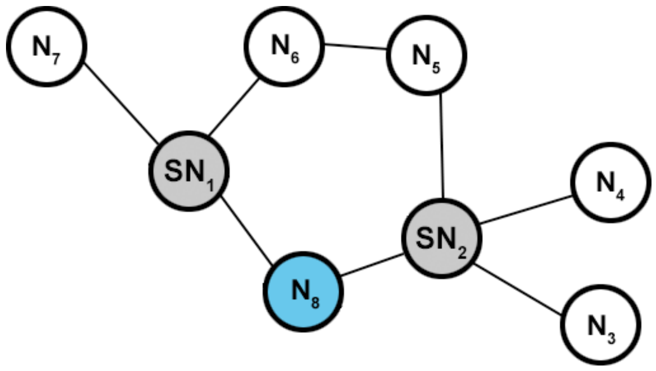
5. Every node that gets an SNCD message further forwards it, until it reaches the issuing SN. Furthermore, in Fig. 3.3b, as the SNCD message has reached node N_8, N_8 appoints itself as an SN and stores the face through which the SNCD message was received in its *snFaceList*.
6. In Fig. 3.3c, when SN_1 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.



(a) SN_2 replies with an SNCD message.



(b) SNCD is forwarded back, on the same path it has travelled to reach SN_2.



(c) SN_1 updates its *snFaceList*.

FIGURE 3.3: The CDS construction algorithm

In addition to the parameters *snFlag* and *snFaceId* mentioned in Section 3.1, 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.

The CDS construction algorithm starts, when the SN applications previously installed by the DS construction algorithm are initialized. All the SNs now broadcast an Supernode Connection Interest (SNCI) message as shown in Table 3.3. As discussed earlier, the nature of a DS assures that the distance for a message of a dominator, in our case an SN, to reach at least another dominator is at most 3 hops. The SNCI messages are broadcasted up to a maximum of three hops or until they reach another SN, we modified the NFD code to check this maximal hop count.

TABLE 3.3: SNCI structure

Name	Interest Name
Nonce	Random number to assure uniqueness of the Interest message

TABLE 3.4: SNCD structure

Name	Content Name
Signature	Signature of the issuing SN

If an SNCI message reaches another SN, this SN responds with a Supernode Connection Data (SNCD) message as shown in Table 3.4. The SNCD message then travels back to the requesting SN. On its way back, if the message passes by any regular nodes, the NFD adds the networking face over which the SNCD message was received to the node's *snFaceList*, a list containing the known neighbouring SN, later used for routing. The NFD also sets the node's *snFlag* to true. The SNCD then still gets forwarded until it reaches the requesting SN. Since the topology is usually well-connected, there are multiple paths from an SN to another, so there is a possiblity, that the same SNCI messages reaches the same SN multiple times, but it only gets processed the first time, if an SNCI with the same nonce reaches an SN the second time, it gets dropped. After all the SNs have done this step, we now have a CDS.

So far, we have described distributed and localized 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. Our algorithm only uses local communication with one-hop neighbours to build a DS.

3.3 Dominating Set Maintenance

We have to maintain DS and CDS when a new node joins the network or 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 3.1 and 3.2 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, no problems will be created and 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 nodes that were connected to the SN need to be aware of this failure so that they again run DS and CDS construction algorithms (Sections 3.1 and 3.2).

Chapter 4
Routing

L-SCN can only operate when the network is clustered, meaning it has been divided into domains and SNs have been elected. The proposed DS and CDS-based construction algorithms, presented in Chapter 3, complement L-SCN by efficiently clustering the network topology. After clustering the network topology, 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 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, because in the CDS-based approach, SNs can communicate directly, whereas in the DS-based approach, SNs need multiple-hop communication.

In this thesis, we propose two variations on the L-SCN architecture [5]. Service and resource discovery (Section 4.1) and intra-domain routing is the same for both variations. But for inter-domain routing, we consider two cases in Sections 4.3 and 4.4, respectively:

1) L-SCN using our DS construction algorithm described in Section 3.1, and

2) L-SCN using our CDS construction algorithm described in Section 3.2.

4.1 Service and Resource Discovery

For routing, SNs need to know the services available in their domain. This is independant, whether a DS or a CDS is used as a backbone. To acquire knowledge about the provided services and available server resources (e.g., CPU, RAM), each SN asks the nodes in its domain about their available services and resources using the pull mechanism illustrated in Fig. 4.1. Fig 4.1a shows that the SN peridodically 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. 4.1a). These SRAI messages contain an empty BF, to assure that the same hash functions and size is used for all the BFs in the domain and facilitate unifying of the BFs. The structure of the SRAI messages is shown in Table 4.1. For using the BFs, a library called Open Bloom Filter [27] was used.

When a service provider application gets an SRAI message, it populates the BF with the service names it provides and sends back an Service and Resource Availabilty Data (SRAD) message, also providing the available resources on this node, as shown in the paper in Table 4.2. When an SN receives an SRAD, it stores it in its CS, along

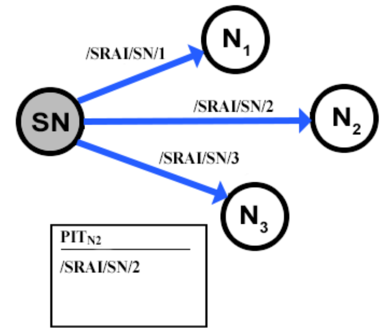
TABLE 4.1: SRAI structure

Name	Interest Name
BF	Empty BF bit vector
Nonce	Random number to assure uniqueness of the Interest message

with the face over which it was received. Furthermore, it unifies the received BF in its own domain BF. As searching a BF can be done in constant time, this domain BF provides faster decision whether the SN needs to perform intra-domain or inter-domain routing.

TABLE 4.2: SRAD structure

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



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

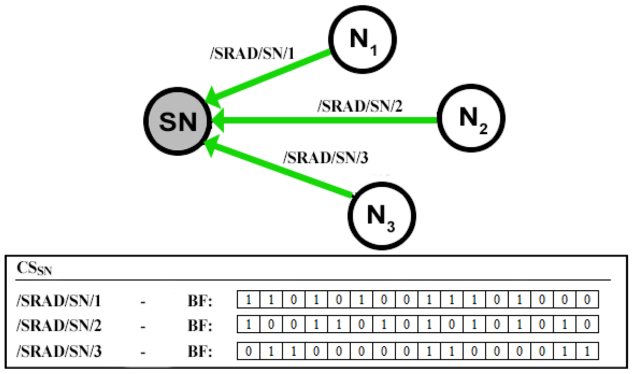

/SRAD/SN/1
/SRAD/SN/2
/SRAD/SN/3
SN
N_1
N_2
N_3
CS_{SN}
/SRAD/SN/1 - BF: 1 1 0 1 0 1 0 0 1 1 1 0 1 0 0 0
/SRAD/SN/2 - BF: 1 0 0 1 1 0 1 0 1 0 1 0 1 0 1 0
/SRAD/SN/3 - BF: 0 1 1 0 0 0 0 0 1 1 0 0 0 0 1 1
(b) SRAD packet forward by regular nodes containing BFs and resource information

FIGURE 4.1: Pulling service and resource availability

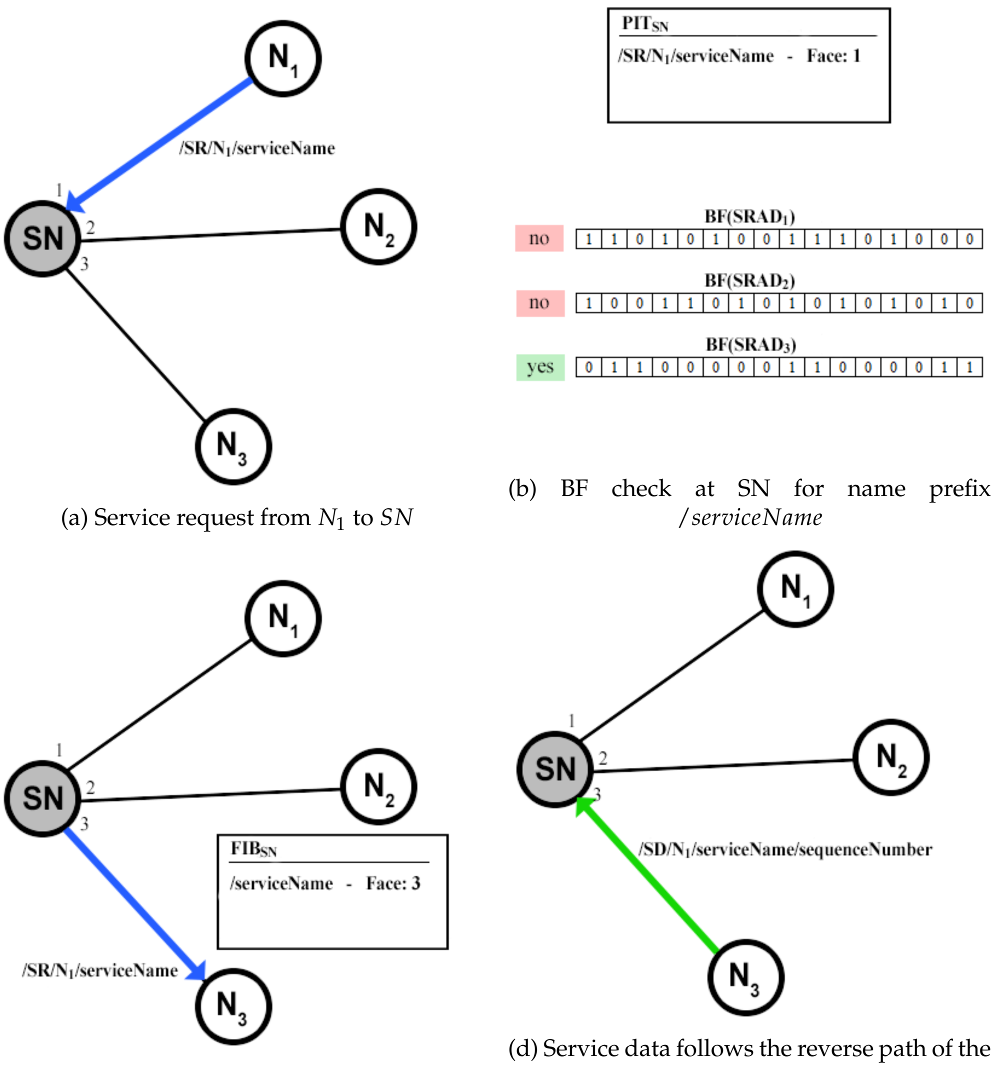
As Fig. 4.1b shows, the SN stores the received SRAD messages as well as the information about their reception faces in its CS.

4.2 Intra-Domain Routing

Fig. 4.2 describes FIB population and intra-domain routing. Like service and resource discovery the principle is independant, whether a DS or a CDS is used as a backbone.

1. As shown in Fig. 4.2a, 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 responsible for the domain. The structure of the SR message is shown in Table 4.3.

2. 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. 4.2b. 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*. When no BF would contain the name prefix, inter-domain routing would be needed to retrieve the requested service.



(a) Service request from N_1 to SN

(b) BF check at SN for name prefix $/serviceName$

(c) FIB population and forwarding SR to N_3

(d) Service data follows the reverse path of the service request

FIGURE 4.2: Intra-domain routing

3. The SR_1 message is sent over next hop face towards node N_3 (Fig. 4.2c). 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.

4. 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, as shown in Fig. 4.2d. 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. 4.2b, 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

TABLE 4.3: SR structure

Name	Interest and Service Name
Hop Count	Hop counter for maximum hop count
Inter-Domain Flag	Boolean variable for intra-domain or inter-domain routing
Nonce	Random number to assure uniqueness of the Interest message

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.

4.3 Inter-Domain Routing in a Dominating Set

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, the SR message is routed inside the domain (see Section 4.2). Otherwise, the SN first checks the *Inter-Domain Flag* to see whether the SR came from its own domain or another domain. If the flag was already set (meaning it came from another domain), it broadcasts the SR to all its neighbours, except through the face over which the SR was received, to avoid loops. If the flag was not set, it changes the value of the *Inter-Domain Flag* to true and then broadcasts the SR to all its neighbours, even through the face over which the SR was received. Fig. 4.3 illustrates why this sort of loop is needed in inter-domain routing for some topologies. The SR message is then flooded until it reaches another SN up to three hops.[1]. 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 resets the hop counter in the SR and floods the SR message until the SR message reaches an SN that its domain provides the requested service. In Chapter 5, 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.

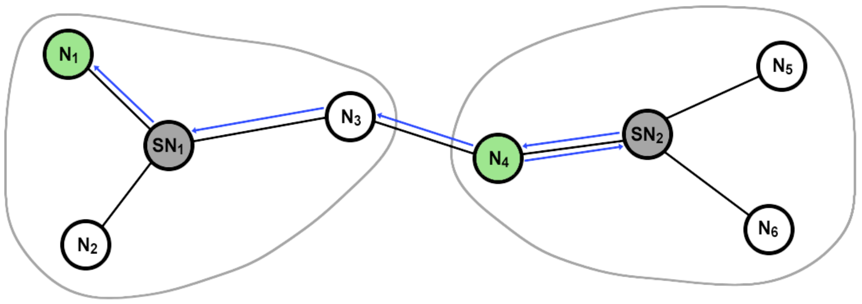


FIGURE 4.3: Example for loop routing

[1]As explained in Chapter 3, the distance between two SNs in a DS does not exceed three hops.

4.4 Inter-Domain 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. 4.4 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. 4.4, when SN SN_2 receives a SR message with name $/SR/N_4/service$ from node N_4, which is asking for a service not provided in the domain of SN_2, this SN checks its FIB for name *service*. If no FIB entry is found and no BF in the CS provides *service*, 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 $/SD/N_4/service$ over face 2, it populates the FIB for name *service* with next hop face 2. As the SNs are connected, only a single hop is needed to reach another SN. This selective multicast reduces overhead compared to the flooding used in the DS approach.

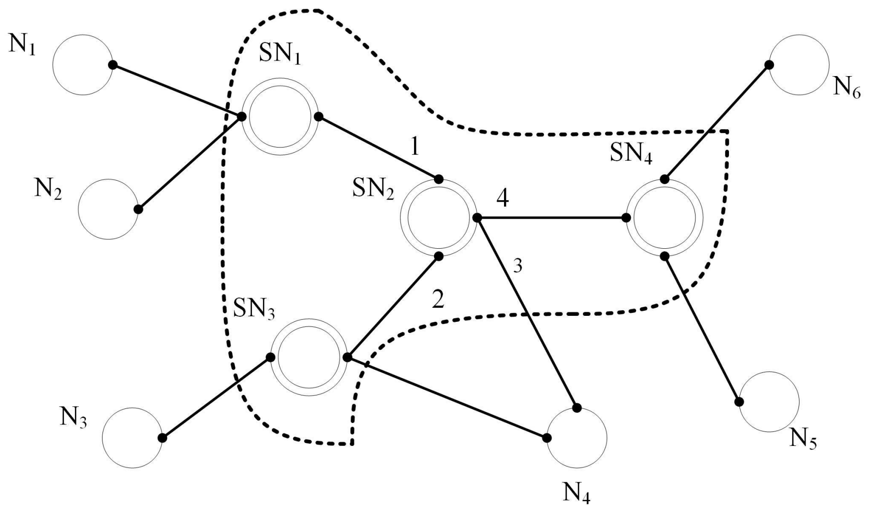


FIGURE 4.4: Inter-domain routing for CDS-based clustering.

4.5 Service Provisioning

When an SR finally reaches a service provider application, the provider will send out a Service Data (SD) message, if it provides the requested service. Because of the small chance of false positives in BFs, there is a possibility, that the SR was routed to a service provider, which unfortunately does not provide the requested service. In this case, the service providers issues a Service Provision Refusal (SPR) message to the SN, as shown in Table 4.4. Upon receiving an SPR, the SN knows, that the chosen route is not valid and therefore needs to reroute the previously received SR, to another face than through which the SPR has been received.

TABLE 4.4: SPR structure

Name	Interest and Service Name
Signature	Signature of the issuing Service Provider

Compared to the ndnSIM implementation which handles only content, and, therefore has no processing time, we needed to add a processing time to simulate services. We decided that each service request had a uniformly distributed processing time between 0.1 and 2s.

Chapter 5

Performance Evaluation

To evaluate the performance of our protocols, we have implemented them in ndnSIM [12]. We compare our protocols with NDN *multicast* forwarding strategy [13].

5.1 Simulation Parameters

For performance evaluation, we use GEANT [14], and Rocketfuel topologies [15]. GEANT topology consists of 40 routers. From Rocketfuel topology traces [15], 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 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, resulting in 1000 unique service requests. 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, meaning the total of transmitted data in the network for the clustering algorithm to perform 2) the required bandwidth overhead for routing, meaning the total of transmitted data in the network for 1000 unique service requests and 3) the average service retrieval time.

5.2 Bandwidth Overhead of Clustering

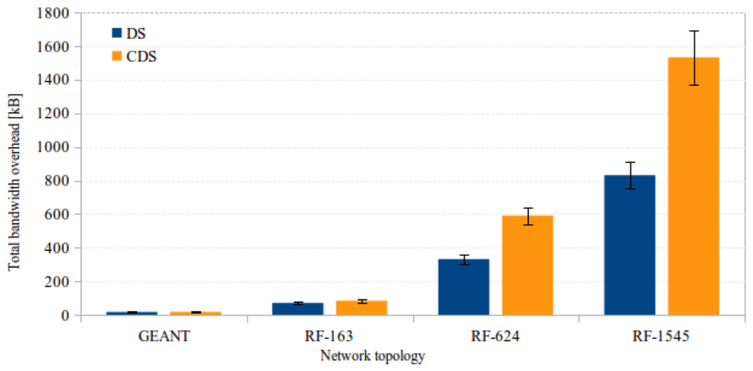


FIGURE 5.1: Total bandwidth overhead for clustering algorithms

In Fig. 5.1, we show the bandwidth overhead, in terms of total transmitted signaling information (in KB) over the network for DS and CDS construction, respectively. Fig. 5.1 shows that when topology size (i.e., the number of nodes) increases, the number of messages exchanged for DS and CDS construction also increases. Further, it shows 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. 5.1 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 DS construction also increases.

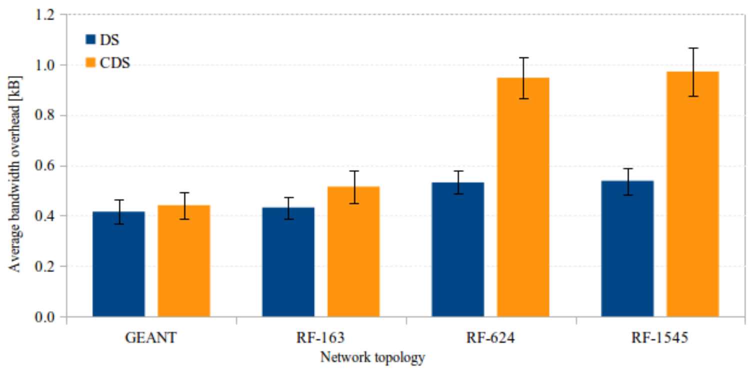


FIGURE 5.2: Average transmitted bytes per node for clustering algorithms

In Fig. 5.2, we show the bandwidth overhead, in terms of average transmitted kilobytes over the network sent for DS and CDS construction per node. We observe similar performance from both Figs. 5.1 and 5.2. In Figs 5.1 and 5.2, if we focus on each of the considered topologies separately and compare DS-based and CDS-based clustering algorithms, we see 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.

5.3 Bandwidth Overhead of Routing

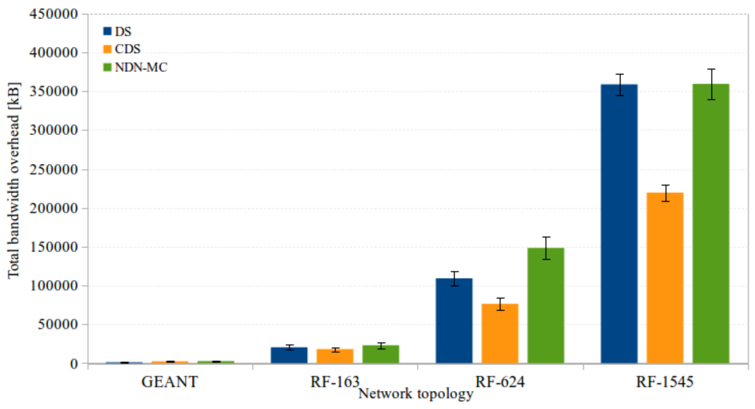


FIGURE 5.3: Total bandwidth overhead for service request routing

In Fig. 5.3, we compare our DS-based and CDS-based routing protocols with NDN *multicast* forwarding strategy [13] comparing bandwidth overhead, in terms of total transmitted kilobytes in the network for service request routing. For all the used topologies, our intuition was to see less overhead for service request routing using a CDS. However, interestingly, we observe from Fig. 5.3 that for GEANT topology, we actually see less overhead for service request routing when a DS is used. 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 we can see that the CDS-based routing requires much less bandwidth overhead for routing service requests. The CDS-based routing protocol uses up to 39% less overhead to route service request messages in the larger topologies.

5.4 Average Service Retrieval Time

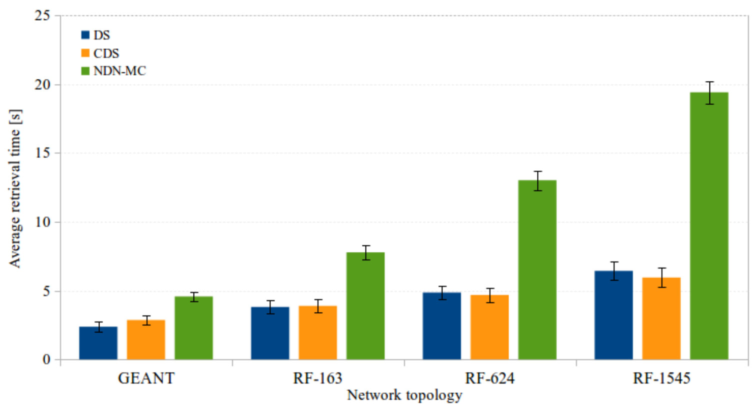


FIGURE 5.4: Average service retrieval time with full link capacities

In Fig. 5.4 we show results in terms of average service retrieval time for service provision with full link capacities. From this figure we can see that NDN Multi-Cast (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 our CDS approach compared to NDN-MC. For small topologies, like GEANT, the DS approach performs around 17% faster, while for large topologies, like RF-1545, the CDS approach performs 8% faster.

From Fig. 5.5, 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. Also, we note an even higher gain, compared to NDN-MC with reduced link capacity. The service retrieval takes up to 73% less time with our CDS-based routing. The CDS approach is also up to 24% faster than the DS approach with reduced link capacities.

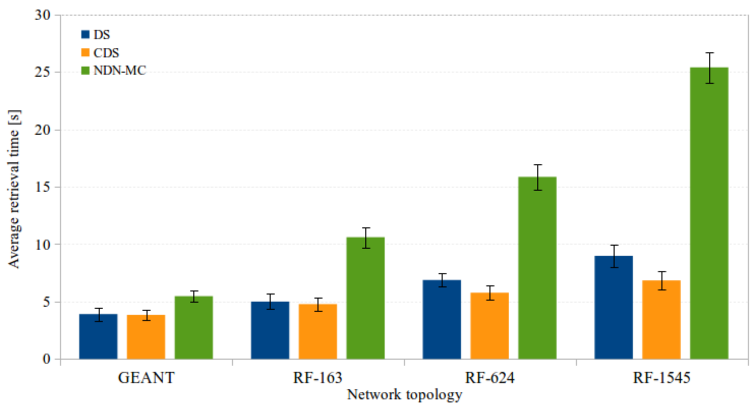


FIGURE 5.5: Average service retrieval time with reduced link capacities

Chapter 6
Conclusions

In this thesis, we studied intra-domain and inter-domain routing of service requests in SCNs and implemented a Layered Service-Centric Network architecture in ndnSIM. We first discussed related work and background informations in Chapter 2. In Chapter 3, we then proposed fully distributed and localized 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 in Chapter 4. For service discovery, we used BFs to reduce the required bandwidth overhead for service availability advertisements. Using the information collected during the proposed service and resource discovery method, we implemented BF-based and resource-aware intra-domain and inter-domain routing protocols to route service requests.

In Chapter 5, we assessed the performance of our DS-based and CDS-based routing protocols using various topologies with 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 from the results 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.

Bibliography

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A Survey of Information-Centric Networking," *IEEE Communications Magazine*, vol. 50, 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 *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, (New York, NY, USA), pp. 1–12, ACM, 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, pp. 66–73, July 2014.

[4] T. Braun, V. Hilt, M. Hofmann, I. Rimac, M. Steiner, and M. Varvello, "Service-Centric Networking," in *2011 IEEE International Conference on Communications Workshops (ICC)*, pp. 1–6, June 2011.

[5] M. Gasparyan, T. Braun, and E. Schiller, "L-SCN: Layered SCN Architecture with Supernodes and Bloom Filters," in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pp. 899–904, Jan 2017.

[6] R. Laskar and H. Walikar, "On domination related concepts in graph theory," in *Combinatorics and graph theory*, pp. 308–320, Springer, 1981.

[7] M. Cardei, M. X. Cheng, X. Cheng, and D.-Z. Du, "Connected Domination in Multihop Ad hoc Wireless Networks.," in *JCIS*, pp. 251–255, Citeseer, 2002.

[8] K. M. Alzoubi, P.-J. Wan, and O. Frieder, "Message-Optimal Connected Dominating Sets in Mobile Ad-hoc Networks," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking and computing*, pp. 157–164, ACM, 2002.

[9] Y. L. S. Zhu and M. T. D.-Z. Du, "Localized Construction of Connected Dominating Set in Wireless Networks," in *Proceedings of US National Science Foundation International Workshop Theoritical Aspects of Wireless Ad Hoc, Sensor and Peer-to-Peer Networks*, 2004.

[10] C. Lenzen, Y.-A. Pignolet, and R. Wattenhofer, "Distributed Minimum Dominating Set Approximations in Restricted Families of Graphs," *Distributed Computing*, 2013.

[11] 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," *IEEE International Conference on Communications*, 2013.

[12] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2.0: A new version of the NDN simulator for NS-3," Technical Report NDN-0028, NDN, January 2015.

[13] 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.

[14] "The GEANT network, 2012." `http://www.topology-zoo.org/dataset.html`. Accessed: 2016-07-25.

[15] "Rocketfuel: An ISP topology mapping engine." `https://research.cs.washington.edu/networking/rocketfuel/`. Accessed: 2019-07-20.

[16] 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.

[17] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Commun. ACM*, vol. 13, pp. 422–426, July 1970.

[18] D. Griffin, P. Simoens, F. Vandeputte, D. Bursztynowski, and F. Schamel, "Service oriented networking," *European Conference on Networks and Communications*, 2014.

[19] S. Cherrier, Y. Ghamri-Doudane, S. Lohier, and G. Roussel, "Services collaboration in wireless sensor and actuator networks: Orchestration versus choreography," *IEEE Symposium on Computers and Communications*, 2012.

[20] S. Cherrier and R. Langar, "Services organisation in iot : mixing orchestration and choreography," 2018.

[21] G. Premsankar, M. D. Francesco, and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet of Things Journal*, 2018.

[22] S. Salsano, N. Blefari-Melazzi, F. L. Presti, G. Siracusano, and P. Ventre, "Generalized virtual networking: an enabler for service centric networking and network function virtualization," *Networks 2014*, 2014.

[23] 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.

[24] 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, Stockholm, Sweden, Jun. 12-16, 2017*, pp. 1–9, Jun. 2017.

[25] 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 Conference (CCNC), Las Vegas, USA, Jan. 11-14, 2019*, pp. 1–7, Jan. 2019.

[26] A. Marandi, T. Braun, K. Salamatian, and N. Thomos, "A Comparative Analysis of Bloom Filter-based Routing Protocols for Information-Centric Networks," in *2018 IEEE Symposium on Computers and Communications, ISCC 2018, Natal, Brazil, June 25-28, 2018*, pp. 255–261, 2018.

[27] "C++ Bloom Filter library - by Arash Partow." `http://www.partow.net/programming/bloomfilter/index.html`. Accessed: 2019-05-13.