Scalable Routing for Networks of Dynamic Substrates

by

Boris Drazic

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

© Copyright 2014 by Boris Drazic

Abstract

Scalable Routing for Networks of Dynamic Substrates

Boris Drazic
Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto
2014

The ever-increasing number of devices capable of, not only connecting to existing communication networks, but also, independently creating new ones is defining a new communication network, in which the Internet is only one of the substrate networks providing connectivity between diverse devices. This is a network with many interconnected mobile devices connecting to infrastructure networks and creating their own dynamic substrate networks.

We present a novel routing scheme for diverse collections of substrate networks with a mix of mobile and static nodes. A key element of the routing scheme is to utilize the exiting routing paths in substrate networks, and set up routing paths between substrate networks. We use sets of nodes as landmarks and define locators that describe node position in the network relative to landmarks. This allows our routing scheme to scale to a large number of nodes, as only information about landmarks needs to be propagated throughout the network.

Contents

List of Tables

List of Figures

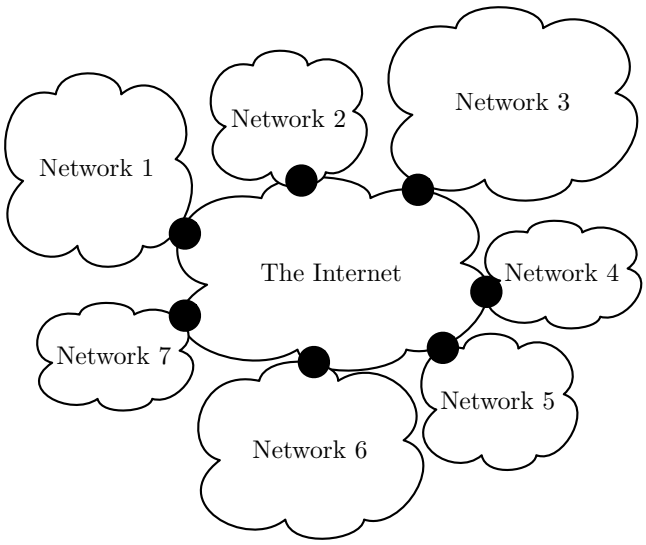
Chapter 1

Introduction

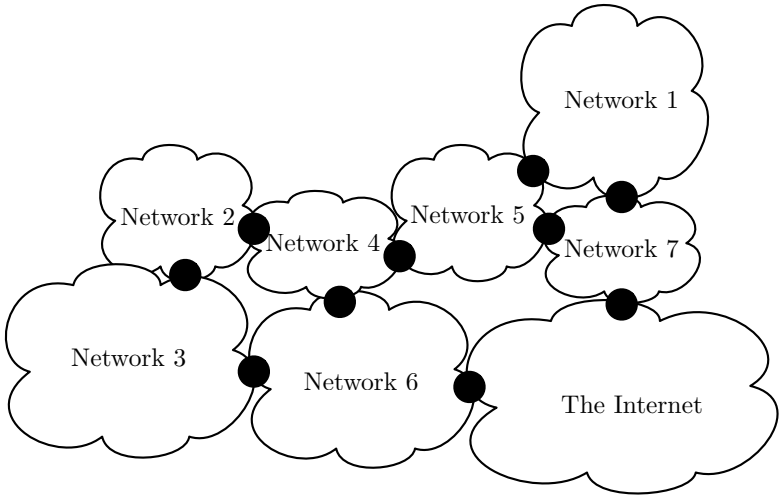
Since its inception the Internet has continually grown and has become a global medium to interconnect people, share ideas, and conduct business. The Internet has evolved to not only support the ever increasing amount of data carried over it, but also to provide new and improved services to users. The growth of the number of users that connect to the Internet, the types of devices they use to connect, and the number of services the Internet offers has raised concerns in the networking community that the original design principles of the Internet are inadequate to support the requirements for the future Internet. Limitations of the current Internet in security, mobility, reliability, and scalability have prompted some Internet researchers to suggest that the evolutionary approach to the development of the Internet should be replaced with a more radical rethinking of the Internet, called "clean-slate design" [1, 2, 3, 4].

The original design of the Internet envisioned a packet switched communication medium that interconnects existing independent networks [5]. The Internet architecture makes minimal assumptions about the functions that member networks provide and delegates the implementation of advanced services to nodes that want to use them. The end-to-end principle [6] argues that the benefits of providing functionality at the network layer, instead of at nodes, are small compared to the cost of providing them and has been a key tenet of the Internet design. However, as Blumenthal and Clark note in [7], a growing number of Internet users wants to access advanced network services without carrying the burden of managing them and expects the network to provide these services. The gap between user demand and the functionality offered by the Internet can be bridged by overlay networks that meet specific user demands, such as distributed lookup [8], resilient routing [9], and peer-to-peer data sharing [10]. Overlay networks are build on top of the Internet as new networks that provide additional functionality to users without the need to change the Internet it self.

As identified in Clark et al. [11], innovation is happening at the edge of the Internet, where networks using novel communication protocols are deployed, e.g., sensor networks, Bluetooth networks, Mobile Ad-hoc Networks (MANETs), and cellular networks. Many of these networks are designed for mobile nodes and characterized by frequent change of connectivity, e.g., Vehicular Ad-hoc Networks (VANETs), or entire networks are mobile, e.g., Personal Area Networks (PANs). Generally these networks are assumed to connect to the Internet, leading to a view of connectivity between networks illustrated in Fig. 1.1(a). The Internet is a central network; all other networks connect to the Internet, but not to each other. Communication between networks is established via communication paths set up in the Internet.



(a) Internet as the central network.



(b) Internet as one of the networks.

Figure 1.1: Interconnected collection of networks.

However, in an environment where many networks are mobile and have only intermittent connectivity the assumption of permanent connectivity to the Internet can be limiting. Instead, we envision an interconnected collection of networks, where the Internet is connected to some but not all networks, as illustrated in Fig. 1.1(b). In such a collection of networks, networks must establish communication paths without support of the Internet.

In the remainder of this introduction, we discuss difficulties for providing communication in a collection of networks and also provide a short overview of the existing approaches. We then briefly describe our proposed routing scheme, which enables communication between interconnected networks, without relying on the existence of the Internet, or any other central network. Finally, we outline the organization

of the rest of the thesis.

1.1 Substrate and Multi-Substrate Networks

A network consists of devices with one or more attachment points (network interfaces) per device. Network attachment points may reside at or above the data link layer and can have different types, e.g., WiFi or Bluetooth interfaces, IP interfaces, TCP server ports, or even application-defined interfaces. We refer to attachment points of the same type, which implement the same protocol and use the same configuration, as *compatible attachment points*. For example, WiFi attachment points implementing IEEE 802.11 are compatible if they use the same service set identifier (SSID), and IP interfaces are compatible if they use the same version of IP and addresses from the same IP address space.

Communication is enabled when compatible attachment points at two different devices are linked by a bidirectional communication channel. The communication channel may be a point-to-point link, a shared broadcast link, a switched network, or an inter-network. We refer in general to communication channels as *links*, and assume that all links are bidirectional. When devices are connected by a link, we say that the devices are *one-hop connected*. Devices that are one-hop connected are referred to as *neighbours*.

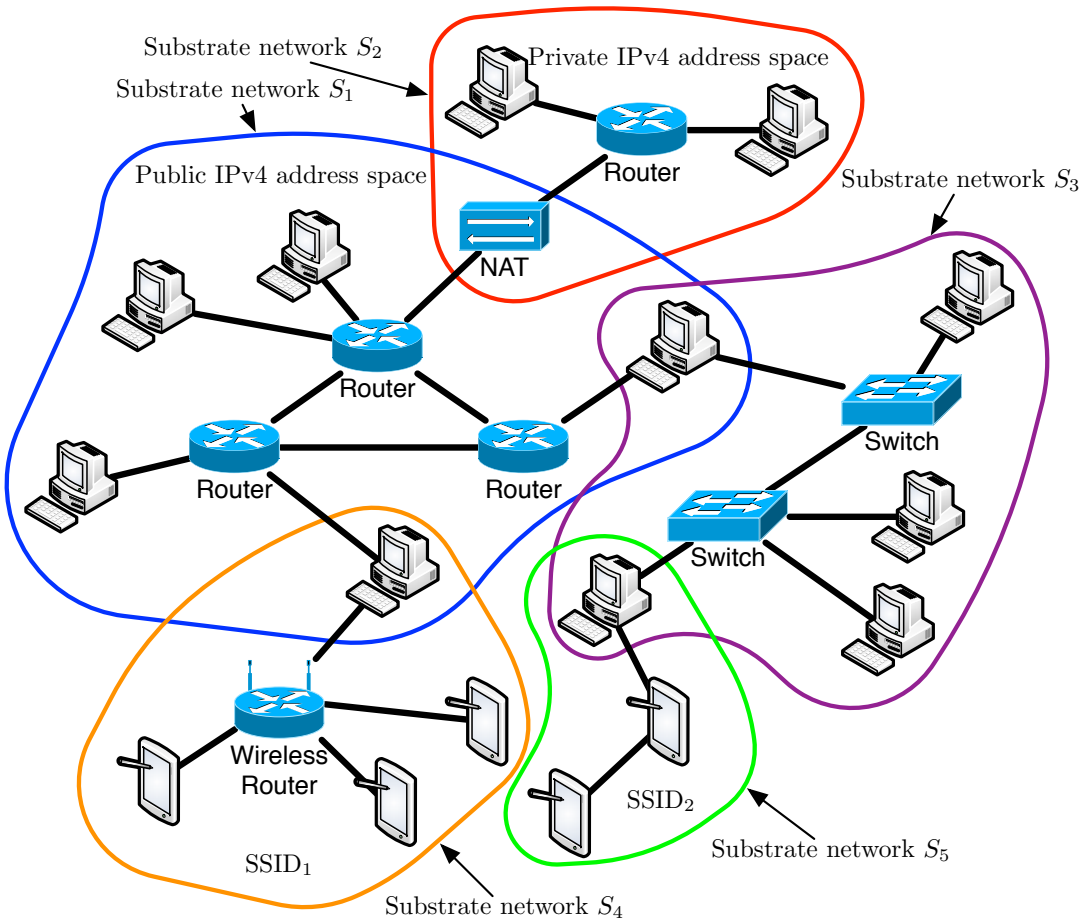
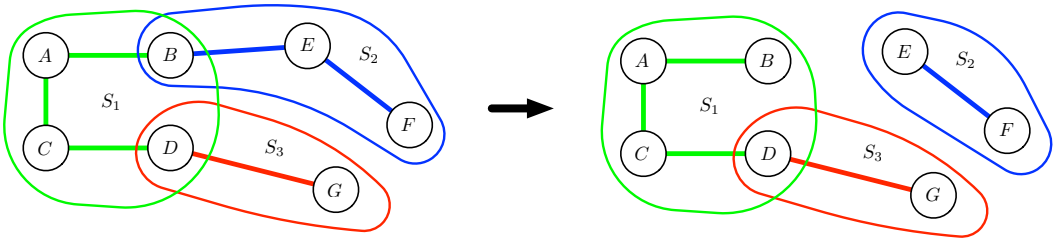


Figure 1.2: Example of substrate networks.

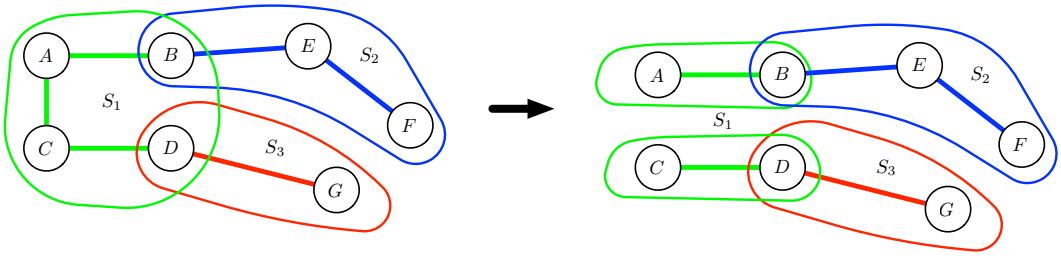
We define a *substrate network* as a collection of devices with compatible attachment points and links between one-hop connected devices. For illustration, Fig. 1.2 shows five substrate networks labelled S_1 through S_5. Substrate networks S_1 and S_2 are defined at the network later and the attachment points implement IPv4. Attachment points of devices in S_1 and S_2 differ, since, in S_1, addresses from the public IP address space are used and in S_2 addresses from a private IP address space are used. S_3 is defined at the data link layer, with attachment points implementing IEEE 802.3. Attachment points in S_4 and S_5 implement IEEE 802.11, but are using different SSIDs.

We refer to *routing* as the ability to set up a multi-hop path of devices, where adjacent devices on the path are one-hop connected. Devices that are not one-hop connected can establish communication by having data forwarded along multi-hop paths set up by routing. A protocol that sets up communication paths is referred to as a *routing protocol*. We refer to *intra-substrate routing* as the routing within a substrate network. For example, intra-substrate routing may be realized with Spanning Tree Protocol (SPT) in a data link layer substrate network, Open Shortest Path First (OSPF) in an IP substrate network, or Ad-hoc On-Demand Distance Vector (AODV) in a wireless ad-hoc substrate network. Some substrate networks may not provide intra-substrate routing, for example, a collection of devices with WiFi attachment points not implementing routing. In this substrate network, a device can forward messages only to its neighbours.

A device that has at least two attachment points connected to different substrate networks is called a *multi-homed* device. Connections between substrate networks are realized by multi-homed devices attached to more than one substrate network. We refer to a *multi-substrate network* as a collection of interconnected substrate networks.



(a) External change of connectivity.



(b) Internal change of connectivity.

Figure 1.3: Dynamic substrates.

Substrate networks with mobile nodes may become dynamic. *Dynamic substrate networks* are characterized by exhibiting two types of changes of connectivity between devices: (1) *external* changes to the connectivity between substrate networks, and *internal* changes to the availability of end-to-end paths

between devices in the same substrate network.

For illustration, Fig. 1.3(a) shows an external change of connectivity for a multi-substrate network composed of three substrate networks (S_1, S_2, and S_3). Initially, node B is attached to substrate networks S_1 and S_2, creating a connection between S_1 and S_2. After node B loses its attachment to substrate network S_2, substrate networks S_1 and S_2 are no longer connected.

Internal changes of connectivity occur due to changes of connectivity between devices within the same substrate network, which may cause partitions of substrate networks. A substrate network is *partitioned* if an end-to-end path of one-hop connected devices is not available between all nodes attached to the substrate network. For illustration, Fig. 1.3(b) shows an internal change of connectivity in substrate network S_1, when one-hop connectivity between nodes A an C is lost. Substrate S_1 is partitioned since there is no end-to-end path from nodes A and B to nodes C and D. Substrate networks S_2 and S_3 are still connected to substrate network S_1. However, since S_1 is partitioned, messages cannot be forwarded form S_2 to S_3 via S_1.

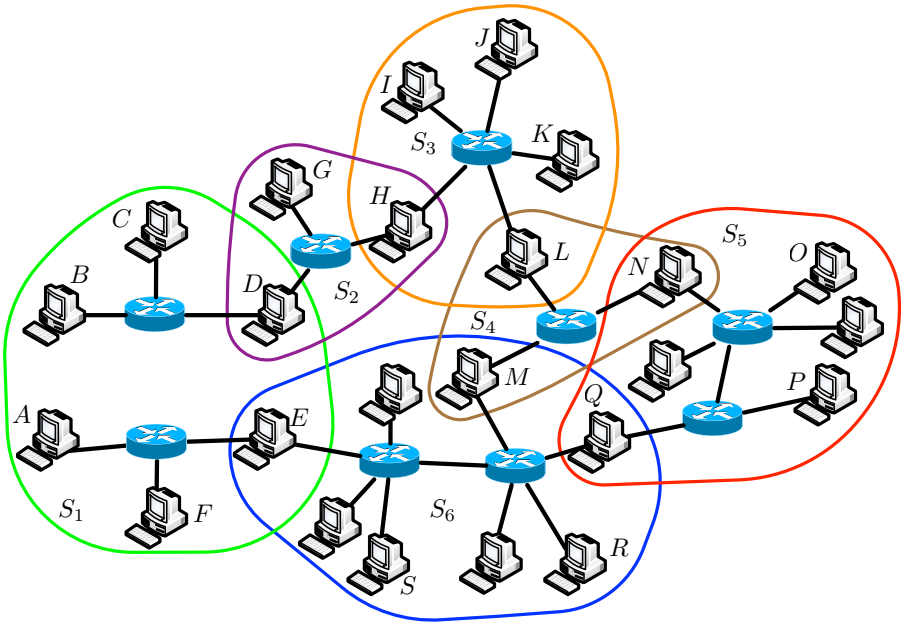
1.2 Routing in a Dynamic Multi-Substrate Network

We view substrate networks as communication channels, which use intra-substrate routing to enable communication between nodes in the substrate networks. Therefore, when routing in a multi-substrate network, we consider nodes that can communicate in a substrate network as one-hop connected. For illustration, Fig. 1.4(a) shows a multi-substrate network with six substrate networks, labelled S_1 through S_6. Fig. 1.4(b) show nodes and one-hop connections between nodes from Fig. 1.4(a). We assume that all substrate networks provide intra-substrate routing. In unpartitioned substrate networks, S_2, S_3, S_4, S_5, and S_6, all nodes can communicate, and thus all nodes are one-hop connected. Substrate network S_1 is partitioned, and nodes B, C, and D cannot communicate with nodes A, E, and F. Nodes B, C, and D are one-hop connected, but none of them is one-hop connected to nodes A, E, or F. One-hop connections do not exist between nodes that are not in a same substrate network.

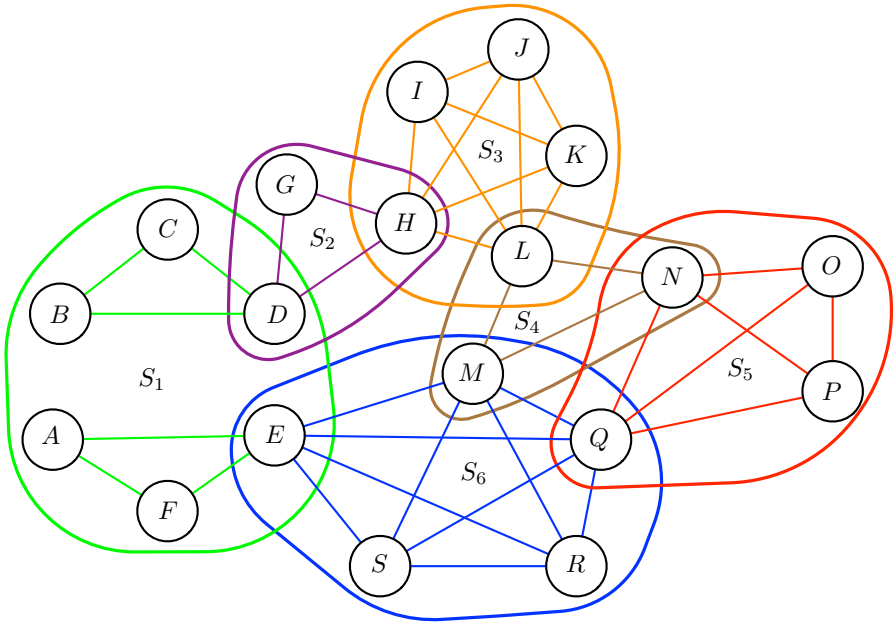
Routing in a multi-substrate network enables communication between nodes by setting up multi-hop paths of nodes, where adjacent nodes are in the same substrate network. Routing in a multi-substrate network can be done as: (1) *flat routing*, or (2) *two-level routing*.

1.2.1 Flat Routing

Flat routing views the multi-substrate network as a flat network of nodes and links. The flat network view captures connectivity between nodes, without information about substrate networks that provide the connectivity. The multi-substrate network from Fig. 1.4 viewed as a flat network is illustrated in Fig. 1.5. Shortest paths routing protocols can be used to realize flat routing for a multi-substrate network. In this case, the routing table of each node stores information about paths to all other nodes. However, for large multi-substrate networks, with many nodes, the size of routing tables may become prohibitively large. The problem of routing table sizes can be mitigated if nodes do not store information about paths to all other nodes. Nodes may be organized into groups of nodes, which are further organized into groups, and implement a hierarchical routing scheme. For illustration, Fig. 1.6 shows the flat network from Fig. 1.5 with two groups, labelled group $\mathbb{A}$ and group $\mathbb{B}$. Group $\mathbb{A}$ contains groups $\mathbb{A}.\mathbb{A}$ and $\mathbb{A}.\mathbb{B}$, and group $\mathbb{B}$ contains groups $\mathbb{B}.\mathbb{A}$, $\mathbb{B}.\mathbb{B}$, and $\mathbb{B}.\mathbb{C}$ Nodes store information about shortest paths to other nodes in the same group, and shortest paths from their group of nodes to other groups, for each level



(a) Multi-substrate network.



(b) One-hop connections between nodes in a multi-substrate network.

Figure 1.4: Multi-substrate network and one-hop connections.

of the hierarchy. For instance, in Fig. 1.6 node L is in Group $\mathbb{B}.\mathbb{C}$ and thus stores information about shortest paths to nodes K, N, and O; to groups $\mathbb{B}.\mathbb{A}$ and $\mathbb{B}.\mathbb{B}$; ant to group $\mathbb{A}$. Messages are forwarded following the hierarchy of groups, that is, a message is first forwarded to the lowest common ancestor group of the sender and destination node. A message is next forwarded to lower level groups, which

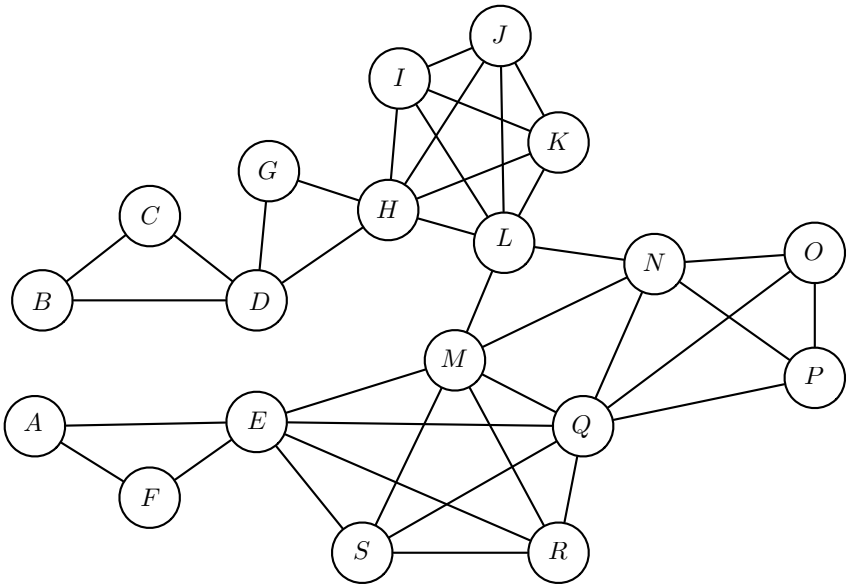


Figure 1.5: Flat network view.

contain the destination node, and finally forwarded to the destination node in the lowest level group. However, a path taken by a hierarchical routing scheme may be longer than the shortest path to the destination node. For illustration, Fig. 1.6 shows the shortest path from source node L to destination node S, with dashed arrows. Path taken by hierarchical routing is shown with solid arrows.

We refer to *path stretch* as the average ratio of lengths of paths set up by a routing protocol and shortest paths. The smallest path stretch of one is achieved by shortest path routing. Routing table sizes and path stretch present a trade-off for routing, i.e., to reduce routing tables size a protocol has to increase path stretch.

1.2.2 Two-level Routing

Two-level routing groups nodes together based on their ability to communicate in substrate networks. A multi-substrate network is viewed as a two-level network. The top level consists of interconnected substrate network. At the bottom level, each substrate network is composed of one-hop connected nodes. For partitioned substrate networks, each group of nodes that can communicate is represented as a separate substrate network at the top level. For substrate networks that do not implement intra-substrate routing, each pair of one-hop connected nodes is represented as a separate substrate network at the top level. Thus, at the bottom level, all pairs of nodes in a substrate network can communicate and are one-hop connected. For illustration, the multi-substrate network from Fig. 1.4 viewed as a two-level network is shown in Fig. 1.7. The top level contains substrate networks, which are connected if they share a multi-homed node. For example, substrate networks S_2 and S_3 are connected by node H. Partitioned substrate network S_1 is represented as two separate substrate networks labelled S_1' and S_1''. At the bottom level, there are no connections between nodes from different substrate networks. Nodes attached to more than one substrate network are represented in each substrate network. For instance, node H is at the bottom level both in substrate networks S_2 and S_3.

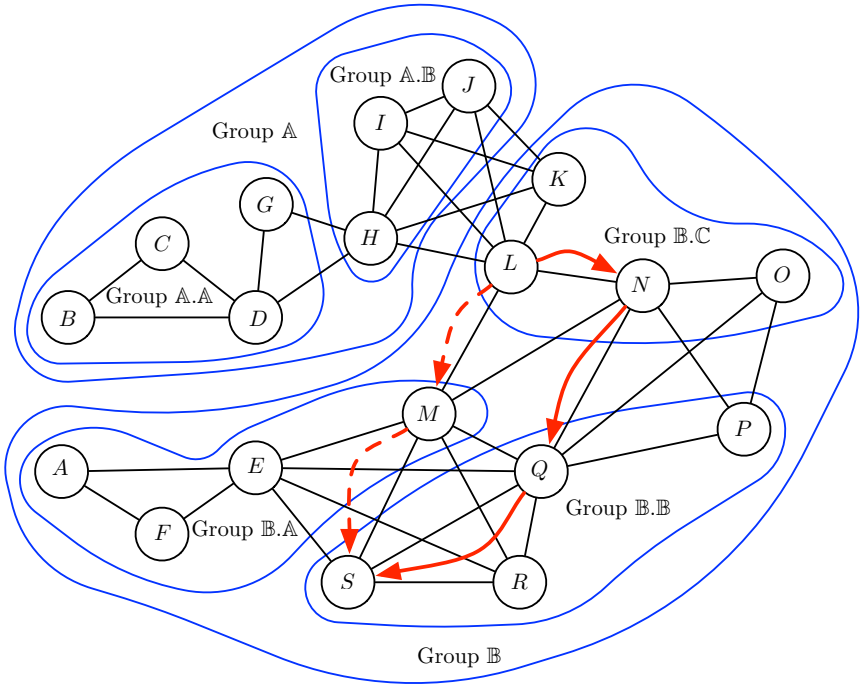


Figure 1.6: Grouping nodes in a flat network.

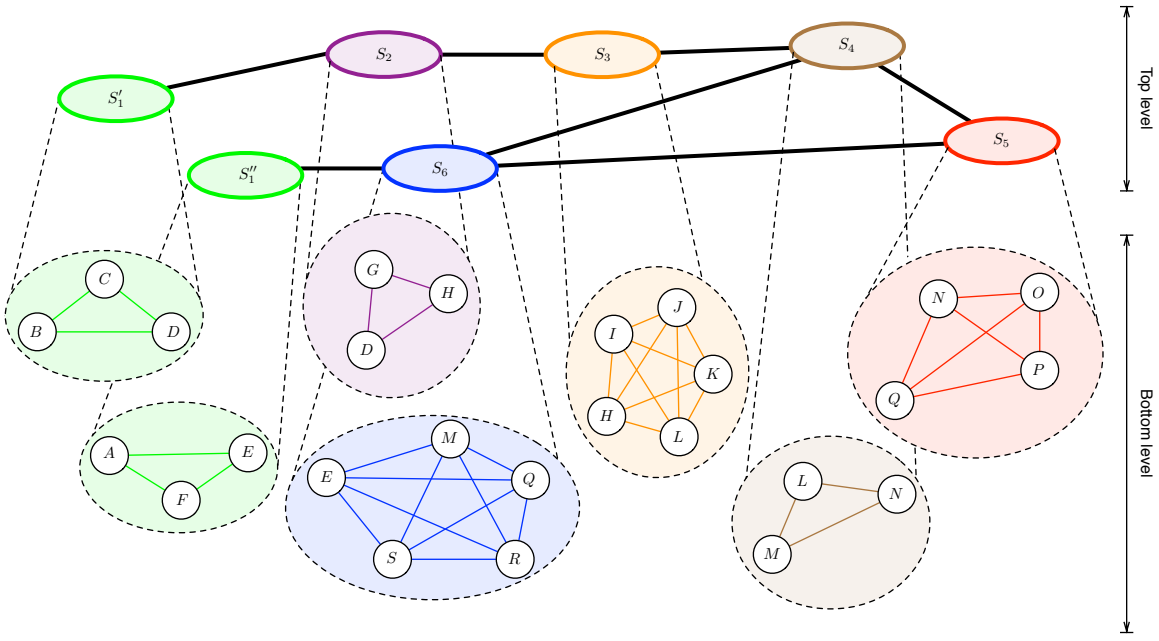


Figure 1.7: Two-level network view.

On the bottom level, the nodes in a substrate network are one-hop connected, and routing for a multi-substrate network does not need to set up any additional paths. On the top level, routing sets up paths between substrate networks. The routing table of each node stores information about paths to

substrate networks and not individual nodes. To forward a message to a destination node, the source node thus needs to know a substrate network to which the destination nodes are attached. Therefore, routing needs to identify substrate networks and associate nodes with substrate networks. Since two-level routing sets up shortest paths to substrate networks and not individual nodes, two-level routing may produce path stretch larger than one. On the other hand, the size of routing tables is proportional to the number of substrate networks and not to the number of nodes, as in shortest path routing.

1.3 Problem Statement and Contributions

This thesis addresses the problem of designing two-level routing for multi-substrate networks, in which substrate networks provide intra-substrate routing. A challenge for routing is to leverage the existing intra-substrate routing, by avoiding setting up paths between nodes that can already communicate in substrate networks. Another challenge for routing is to be able to scale to multi-substrate networks of any size.

We propose to identify subsets of nodes, which can communicate via paths set up by intra-substrate routing, and refer to them as *reachability domains*. Further, we present *landmark domains routing*, which sets up paths between reachability domains and uses paths set up by intra-substrate routing within reachability domains. To achieve scalability of the proposed routing scheme, we designate a subset of reachability domains as *landmark domains*. Nodes store information only about paths to landmark domains, and not to all reachability domains. Each node has a *locator* that describes a path from a landmark domain to the node. Messages are forwarded first to a landmark domain and then using a locator to the destination node.

We show that the proposed landmark domains routing scales to large multi-substrate networks and most of the set up paths have length close to the length of shortest paths. Through simulations we verify the correct operation of landmark domains routing and show its performance is comparable to or better than a compact routing scheme Disco [12] and two overlay based routing schemes: UIP [13] and VRR [14].

1.4 Thesis Organization

Chapter 2 presents research related to our routing scheme. Chapter 3 presents the concepts that build up landmark domains routing Here we discuss how are reachability domains detected and managed. We also describe what is a node locator and how it is constructed and used in message forwarding. Chapter 4 gives the implementation of the presented routing, with the evaluation presented in Chapter 5. Chapter 5 also details the generation of test network topologies and the setup of simulations. Finally, Chapter 6 provides the conclusion and gives some directions on future work.

Chapter 2

Related Work

In this chapter we review the related work in routing for multi-substrate networks. We start by discussing the idea of identifier/locator separation, which is common to many proposals for routing in multi-substrate networks. Next, we discuss approaches to flat routing, focusing on compact routing and greedy routing. In the last section we present approaches to two-level routing.

2.1 Identifier/Locator Separation

Differences between node identity (or a name) and location (or an address) have been expressed in 1978 by Shoch [15] and his ideas further expanded by Saltzer in [16]. Shoch states that:

- "the name of a resource indicates what we seek,
- an address indicates where it is, and
- a route tells us how to get there".

Further, addresses must be meaningful through the network and drawn from an uniform address space. Many authors (e.g., [17, 18, 19]) argue that the identity of a node should be separated from its location. Using a single entity to express both identity and location (e.g., the IPv4 address in the Internet) complicates support for network features such as mobility and multi-homing. For example, as a mobile node moves through a multi-substrate network, the mobile node changes its locator. If identifier and locator functionalities are coupled in a single entity, the node is forced to change its identity every time it changes its location. As a result, all references to the node need to be updated throughout the multi-substrate network. Another example is a multi-homed node, which has a separate locator in each substrate network. If locators are used as identifiers, a multi-homed node has multiple identifiers, all of which refer to its single identity, thus loosing the one-to-one mapping from identifiers to nodes.

Pip [20], designed by Francis, is an early adoption of an identifier/locator separation in a datagram based Internet protocol, which was intended to replace IPv4. In Pip, nodes are uniquely identified by 64-bit numbers drawn from a flat name-space, refereed to as *Pip Identifiers* (Pip IDs). Pip assumes a hierarchy of substrate networks created by the provider-customer relation between Internet service providers. A *Pip Address* is encoded as a sequence of separate numbers, one number for each level of the hierarchy, and is used as a locator. The last part of a Pip Address is a Pip ID of the destination node, which is only used to deliver a message once the message reaches the destination substrate network.

When identifier/locator separation is used, nodes are referred to by their identifiers. That is, a node sending a message knows the identifier of the destination node and not the locator. Routing that uses identifier/locator separation needs an additional mechanism to obtaining node locators from node identifiers, referred to as *identifier resolution service*. Identifier resolution service may be implemented by different distributed directory services (e.g., Domain Name System (DNS) or Distributed Hash Table (DHT)). Every node is responsible to register a mapping from its identifier to locator(s) with the identifier resolution service. When a locator of a node changes, e.g., due to a link failure or node movement, the node updates the mapping from its identifier to locator(s).

Andreas et al. [21], argue that the identifier resolution service should be a part of the network layer. The authors divide the network layer into two layers: (1) the *naming layer* and (2) the *forwarding layer*. The naming layer resolves the identifier of the destination into a locator, and then the forwarding layer delivers messages to the destination using a locator. The naming layer is realized as an overlay network, which can deliver messages based on identifiers. Identifier of a node is resolved into a locator by forwarding a locator query message, in the overlay network, to the node. The node replies with a message containing the locator of the node. The authors argue that since identifier resolution is a one-time action, messages in the naming layer can take longer paths than messages in the forwarding layer.

Bless et al. [22] present a routing scheme that implementing the ideas of Andreas et al. [21] for a multi-substrate network. The authors assume that substrate networks are connected by relay devices, i.e., NAT devices and protocol translators. Each node is responsible to create its own locator, called *Endpoint Descriptor*. An Endpoint Descriptor contains addresses of a node and relay devices, which can be used to contact the node. The authors propose using two layers: (1) *Base Overlay* layer as the naming layer and (2) *Base Communication* layer as a forwarding layer. The Base Overlay layer is realized as a distributed hash table, which stores mappings from node identifiers to Endpoint Descriptors. The Base Communications layer forwards messages based on Endpoint Descriptors. The work of Bless et al. is extended by Mies et al. [23] who introduce a protocol for discovery and management of *connectivity domains*, using a gossiping mechanism. A connectivity domain is a set of nodes in a substrate network that can communicate without using relay devices. Connectivity domains are used to detect relay devices, which are used by nodes in their Endpoint Descriptors.

2.2 Flat Routing

Destination	Next-hop	Description
1.1.2	1.1.2	Nodes in Level 1 group
1.1.3	1.1.3	
1.2	1.1.3	Level 1 groups in Level 2 group
2	1.1.2	Level 2 groups in Level 3 group
3	1.1.3	

Table 2.1: Routing table of node A from Fig. 2.1.

Recall from Section 1.2.1 that flat routing may be realized as shortest path routing, with routing table sizes proportional to the number of nodes in the multi-substrate network. The idea of a trade-off between path lengths and routing table sizes is presented by Kleinrock and Kamoun [24] as *hierarchical*

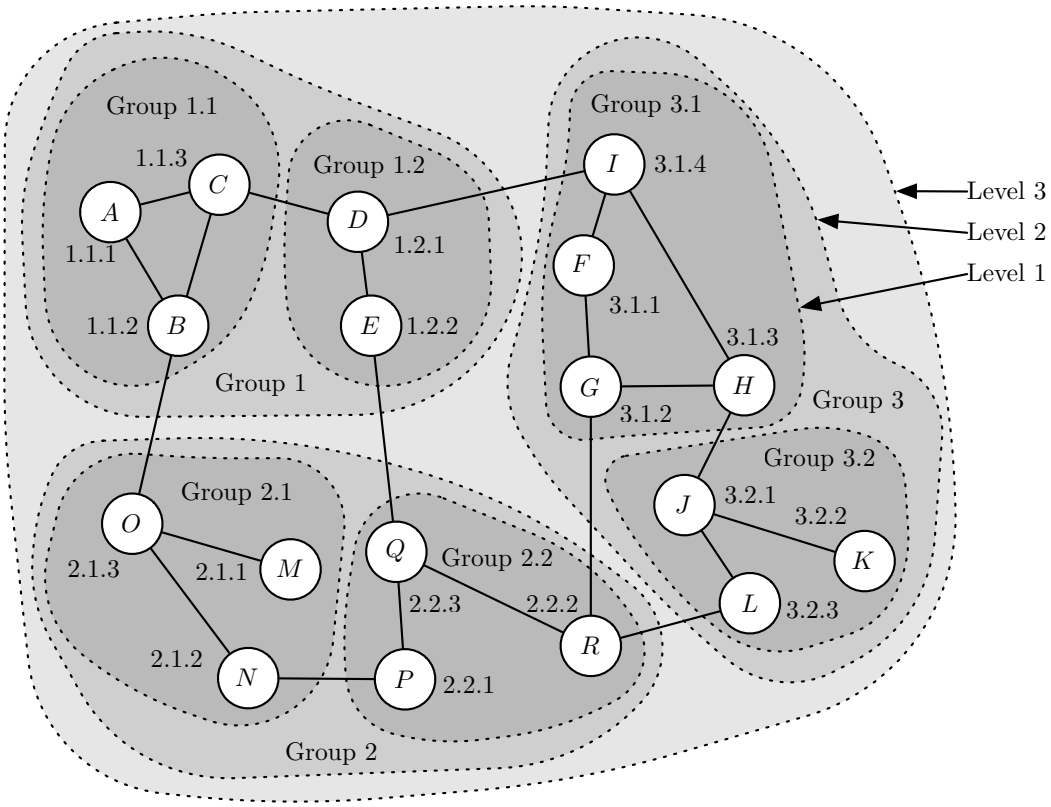


Figure 2.1: A three-level hierarchy for hierarchical routing.

routing. Nodes are organized into groups and groups of nodes are further grouped together to form a hierarchical structure. A node address encodes the groups to which the node belongs at each level of the hierarchy. For illustration, Fig. 2.1 shows a three-level hierarchy of groups, with addresses shown next to the nodes. A node stores the information about the next-hop nodes on the shortest paths to other nodes in the same lowest level (Level 1) group and to one node in each Level i group in the same Level $(i+1)$ group as the node. For example, routing table of node A from Fig. 2.1 is shown in Table 2.1. Node A has routing table entries for nodes in Group 1.1 (at Level 1), Group 1.2 in Group 1 (at Level 2), and Group 2 and Group 3 (at Level 3). A message is delivered to a destination by forwarding the message to groups, in order specified in the address of the destination.

Kleinrock and Kamoun show that in a network with n nodes the minimum number of routing entries a node needs to stores is $e \ln(n)$. However, the paths set up by hierarchical routing are longer than the shortest paths. For example, consider a path from node E to node L in Fig. 2.1. The shortest path is E, Q, R, L and is three hops long. Since node L has address 3.2.3, the path taken by hierarchical routing goes first to Group 3, then to Group 3.2, and finally to node L. This path is E, D, I, H, J, L and is five hops long.

Tsuchiya [25] presents hierarchical routing, called landmark hierarchy, better suited for dynamically changing networks. A landmark is a node whose neighbours within a certain radius (i.e., a number of hops away) store information about shortest paths to the node. A hierarchy of landmarks is created by assigning larger radii to subsets of landmarks, in such a way that each landmark at level i has at least

one landmark of level $i+1$ in its radius. The radius of a landmark at the top level of the hierarchy must be larger than the diameter of the network. This means that all nodes store information about the shortest paths to landmarks at the top level of the hierarchy. A node address is a sequence of addresses of landmarks, one for each level of the hierarchy, starting at the highest level. Messages are delivered to a destination by forwarding them between landmarks in the address of the destination. By construction of the landmark hierarchy, each landmark represented in an address stores information about the shortest path to the next landmark in the address. Tsuchiya shows that path lengths and the size of the stored state are less sensitive to placement of landmarks than to the placement of node groups in the routing hierarchy of Kleinrock and Kamoun.

Hierarchical routing and landmark routing try to reduce the size of routing tables to create a scalable routing scheme. A routing scheme is said to be scalable if the size of routing tables grows slower than linearly with the number of nodes in a network. The trade-off between the routing table sizes and path lengths in designing scalable routing is formalized in the *compact routing* research, described in the next section. Research in the *greedy routing* explores scalable routing that does not set up routing paths, rather, nodes make forwarding decisions using only the local information about their neighbours. We describe the research in greedy routing after the research in compact routing.

2.2.1 Compact Routing

The research in compact routing explores the theoretical limits of routing scalability and also provides concrete routing schemes that satisfy the limits. The increase in the path length produced by a routing algorithm, compared to the shortest path, is referred to as *path stretch*. Path stretch of a routing scheme is defined as the maximum ratio of the length of a path produced by a routing scheme and the shortest path, between any pair of nodes, for any substrate network that the scheme can operate on. A routing scheme operating on a network with n nodes is said to be compact if the scheme:

(1) uses addresses and headers with the size that is logarithmic to n,

(2) produces routing tables with the size sub-linear to n, and

(3) guarantees path stretch bounded by a constant.

Compact routing schemes are classified into: *name-depend* and *name-independent*. A name-dependent routing scheme defines addresses for nodes and encodes some topological information in the addresses. In the name-independent routing scheme nodes can be assigned arbitrary addresses, i.e., addresses drawn from a flat name-space. Thus, name-dependent addresses are comparable to locators and name-independent addresses are comparable to identifiers.

Compact routing schemes follow a basic idea of hierarchical routing: a node knows optimal paths to nodes that are close and sub-optimal paths to nodes that are further away. Using the idea of landmarks from Tuschiya [25], a compact routing scheme designates a subset of nodes as landmarks. Nodes that are close to some node N are referred to as the *vicinity* of node N. Landmarks and vicinities are selected in such a way that the vicinity of each node contains at least one landmark. A node stores information about shortest paths to all landmarks and the nodes in its vicinity. An address of a node, say N, consists of the landmark, say L, closest to node N and information necessary to forward a message from the selected landmark L to node N. The information may be a source route from L to N. Or, if nodes on a path from L to N store routing entries for N, the information can be the identifier of node N. A

node forwards messages to destinations in its vicinity using shortest paths. For a destination outside of vicinity (faraway node) messages are forwarded to the landmark closest to the destination and from the landmark to the destination.

An early work in the compact routing by Peleg and Upfal [26] establishes a lower bound of $\Omega(n^{\frac{1}{2k_{PU}+4}})$ bits of memory per node in a network with n nodes, to guarantee path stretch of $k_{PU} \geq 1$. Further work by Gavoille and Gengler [27] proofs that for any network with n nodes all routing schemes with path stretch strictly below 3 ("stretch<3 routing") require at least $\Omega(n)$ bits of memory per node. Thus, a compact routing scheme cannot exist for a path stretch strictly less than 3, since the minimum routing table size is not sub-linear. Thorup and Zwick [28] show that all "stretch<5 routing" schemes require at least $\Omega(n^{1/2})$ bits of memory per node on any n node network.

Cowen [29] presents the first compact routing scheme with the minimum possible path stretch of 3, the size of the routing tables bounded by $O(n^{\frac{2}{3}} \log^{\frac{4}{3}} n)$ at every node, and node names of size $\log(n)$. For each node, its vicinity consists of the set of n^{α_C} nodes closest to the node. At most $O(n^{1-\alpha_C} \log n)$ landmarks are selected in such a way that each node's vicinity contains at least one landmark. Up to $O(n^{\frac{1+\alpha_C}{2}})$ additional landmarks are selected among the nodes that are in more than $n^{\frac{1+\alpha_C}{2}}$ vicinities. Nodes are assigned topologically significant locators (i.e., the routing scheme is name-dependent) consisting of three parts: (1) a name of a node; (2) a name of a landmark; and (3) a name of the first link on the shortest path from the chosen landmark to the node ("output port"). Nodes select the closest landmark for their address and store next-hop nodes on shortest paths to all the landmarks. Every node N also stores the information about the next-hop nodes on the shortest paths to nodes closer to N than any landmark node. By setting $\alpha_C = \frac{1}{3} + \frac{2 \log \log n}{3 \log n}$ the number of entries in a routing table is bounded by $O(n^{\frac{2}{3}} \log^{\frac{4}{3}} n)$.

Cowen's bound on the routing table size of $\tilde{O}(n^{\frac{2}{3}})$ bits per node for stretch=3 compact routing was improved by Thorup and Zwick [**?**] to $\tilde{O}(n^{\frac{1}{2}})$ bits per node.[1] Since the same authors proved earlier in [28] that any stretch<5 compact routing scheme has the routing table size of at least $\Omega(n^{1/2})$ bits per node, this results is optimal up to logarithmic factors. The proposed compact routing scheme, refereed to as TZ, is similar to Cowen's, however it differs in the landmark selection. TZ defines a node's vicinity as the set of all nodes closer to the node than any landmark and randomly selects s_{TZ} nodes as landmarks. The authors prove that all resulting vicinities have size bounded by $\tilde{O}(\frac{n}{s_{TZ}})$. A routing table of a node N, holds one entry for each landmark in the network, and one entry for each node in the vicinity of node N, resulting in the routing table size bounded by $s_{TZ} + \tilde{O}(\frac{n}{s_{TZ}})$. By setting $s_{TZ} = \left(\frac{n}{\log n}\right)^{\frac{1}{2}}$ the authors prove that the size of the routing tables is bounded by $\tilde{O}(n^{\frac{1}{2}})$.

Compact routing schemes presented by Peleg and Upfal [26], Cowen [29], and Thorup and Zwick [**?**] are name-dependent, i.e., the routing scheme can select the identifiers for nodes. Awerbuch et al. [30] argue that name-independent routing is more difficult to implement than name-dependent routing, since it requires mapping from identifiers to locators. However, the authors argue, name-independent routing is more appropriate for networks where node connectivity changes frequently, since name-independent routing does not require changing node identifiers as their connectivity changes.

A name-independent compact routing scheme proposed by Arias et al. [31] includes name resolution as a part of routing. The authors achieve path stretch of 5, while bounding the stored state at any node to $\tilde{O}(n^{\frac{1}{2}})$. Similarly to name-dependent schemes, vicinities are defined as the $n^{\frac{1}{2}}$ neighbours closes to a node. Up to $O(n^{\frac{1}{2}} \log n)$ nodes are selected as landmarks in such a way that each vicinity contains at

[1] $\tilde{O}(\cdot)$ is used for $O(\cdot)$ notation that ignores logarithmic factors, i.e., $f(n) = \tilde{O}(g(n))$ is shorthand for $f(n) = O(g(n) \log^k g(n))$ for some k.

least one landmark. The namespace, from which node names are drawn, is divided into $n^{\frac{1}{2}}$ blocks. A subset of nodes is selected to store name to address mappings in such a way that: (1) no node stores mappings for more than $O(\log n)$ blocks and (2) each vicinity contains nodes responsible for all blocks. A node stores information about the neighbours, in its vicinity, which store name to address mappings. Message destinations are given as names, and name resolution is performed before forwarding. If the destination is in the vicinity of the sending node or is a landmark, the sending node has a mapping for the name, and sends the message directly to the destination using its routing table. Otherwise, the message is sent to the node in the vicinity, which stores mappings for the block of names that includes the name of the destination. From the node that stores name to address mapping the message is forwarded to a landmark and finally to the destination node.

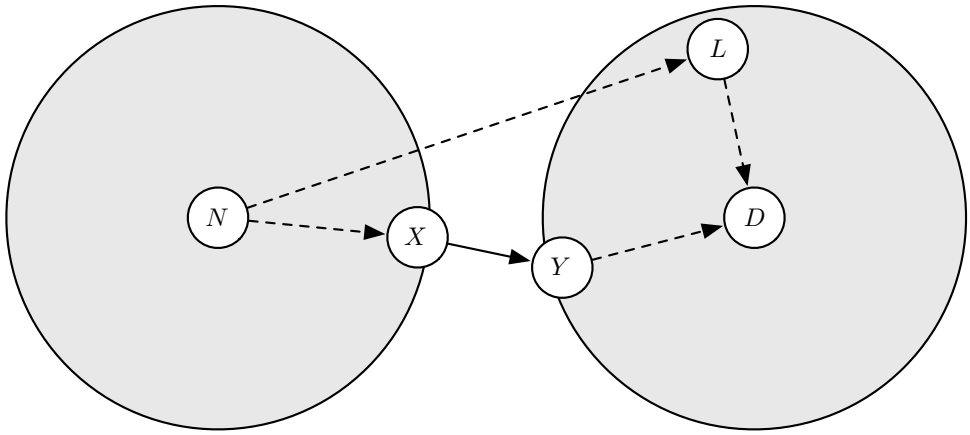


Figure 2.2: Path to a node in a neighbouring vicinity

Abraham et al. [32] present a name-independent compact routing scheme with $\tilde{O}(n^{\frac{1}{2}})$ storage requirement, which reduces path stretch to 3, making this scheme optimal up to logarithmical factors. Their routing scheme is similar to the scheme of Arias et al. [31], with a node storing the shortest paths to the nodes in its vicinity and additionally to the nodes in neighbouring vicinities. For illustration, Fig. 2.2 shows a source node N and a destination node D with their vicinities indicated with grey circles. Node L is a landmark closes to destination D. Vicinities of nodes N and D are neighbouring vicinities if neighbour nodes X and Y exist, such that X is in the vicinity of N and Y is in the vicinity of D. The authors show that the path from N to D, using nodes X and Y is shorter than the path using node L, and can be used to create a routing scheme with path stretch 3.

Westphal and Kempf [33] are the first to propose a compact routing scheme for a network with mobile nodes. They explore a specific model of a dynamic network where a limited number of mobile nodes is attached to a network of static nodes. Mobile nodes cannot communicate directly to other mobile nodes and a mobile node can attach only to one static node (parent static node). The authors apply the TZ compact routing scheme [**?**] to the static part of the network. Forwarding is the same as in TZ, with two rules added for mobile nodes: (1) messages for mobile nodes are forwarded to their parent static nodes, which pass messages to destinations, and (2) all messages originating at mobile nodes are passed to parent static nodes for forwarding. Westphal and Kempf achieve path stretch of 3 and routing table sizes bounded by $O(\sqrt{n \log(n)})$.

The first compact routing scheme for an unrestricted dynamic network is presented by Singla et al.

[12] and is called *Disco*. In Disco, landmarks are selected uniform-randomly, with each node independently deciding if it should become a landmark, to produce $\Theta(\sqrt{n \log n})$ landmarks with high probability. A node's vicinity is defined as the $\Theta(\sqrt{n \log n})$ nodes closest to the node, and each node stores routing table entries for shortest paths to all nodes in its vicinity. A node's address is composed of the identifier of the closest landmark and a source route from this landmark to the node. Forwarding is done first to the landmark in the address of the destination, by using the routing tables, and them to the destination by using the source route in the address. This routing scheme guarantees path stretch of 3, while the state per node is bounded by $\tilde{O}(r_S n^{\frac{1}{2}})$, where r_S is the maximal size of a source route.

2.2.2 Greedy Routing

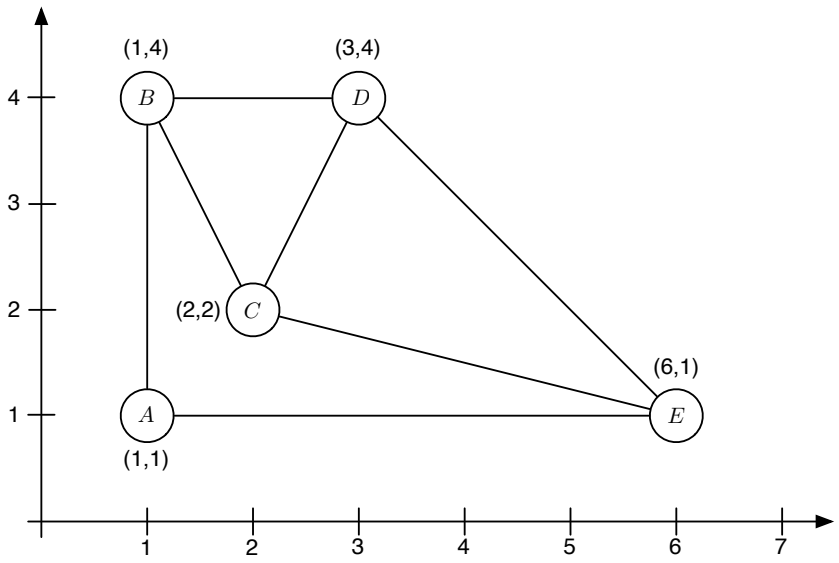


Figure 2.3: Greedy routing.

The research in *greedy routing* shows that the setup of routing paths can be completely avoided. In greedy routing, node addresses are coordinates from a metric space equipped with a distance function. A node forwards a message by sending it to the neighbour node closest to the destination, measured by the distance function. Each time a message is passed from one node to another, the distance to the destination decreases. For illustration, Fig. 2.3 shows five nodes positioned in a Cartesian coordinate system, with distance function δ measuring Euclidean distance. Node A forwards a message for destination D by comparing distances of its neighbour nodes to node D. Since $\delta(B, D) < \delta(E, D)$, node A forwards the message to node B, which has a link to destination D.

Research on greedy routing starts with *geographical routing*, where node's coordinates are its geographical coordinates (e.g., obtained from a GPS device). A node stores routing table entries with the coordinates of each neighbour node and the information how to reach the neighbour (e.g., an output port or an address in a substrate network). To forward a message, a node compares the coordinates of the destination to the coordinates of all its neighbours, and forwards the message to the neighbour geographically closest to the destination. In geographical routing, since the routing table size is equal to the number of node's neighbours, as long as the number of neighbours grows sub-linearly with the size

of the network, so does the routing table size.

Using geographical routing does not guarantee that all messages will be delivered to their destinations due to the existence of local minima. A node that is closer to a destination than any of its neighbours is a local minimum for the destination. For example, in Fig. 2.3 node A is a local minimum for destination C: $\delta(A, C) < \delta(B, C) < \delta(E, C)$. That is, node A cannot forward a message for node C to a neighbour that is closer to C than node A. Greedy routing deals with the problem of local minima by using special routing rules to forward a message away from the local minima, e.g., perform an expanding ring search until a closer node is found [34], perform scoped flooding with a scope derived from coordinates [35], or send the message to a parent node in a global embedded tree [36]. After Rao et al. [34] proposed using coordinates drawn from a virtual metric space, instead of the geographical coordinates, another way of dealing with local minima became possible. By carefully assigning virtual coordinates to nodes, existence of local minima can be completely avoided. Such an assignment is called a *greedy embedding*.

Rao et al. [34] propose assigning virtual coordinates to nodes to avoid the need for GPS devices. Two nodes are preselected to be *beacon* nodes, designated first and second beacon. Beacon nodes flood the network with HELLO messages, which are forwarded by other nodes to their neighbours, only the first time the message is received. HELLO messages have a hop count field, and every time a node forwards a HELLO message, the node increases the hop count field. The distance of a node to a beacon node is defined as the minimum hop count in the HELLO messages the node received from a beacon. Any node further away from the first beacon than any of its two-hop neighbours becomes a *perimeter* node. Next, all perimeter nodes flood the network with HELLO messages to enable all nodes to compute their distances from all perimeter nodes. The perimeter nodes again flood the network, this time with messages carrying their distances from other perimeter nodes, and as a result each perimeter node knows distances between every pair of perimeter nodes. The perimeter nodes chose their virtual coordinates to minimize:

$$\sum_{I,J \in perimeter\_set} \left(measured\_dist(I, J) - dist(I, J)\right)^2,$$

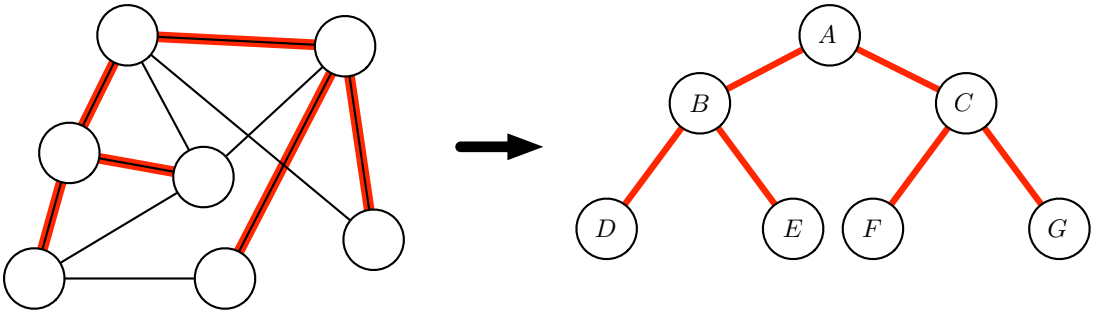
where *perimeter_set* is the set of all perimeter nodes, *measured_dist* is distance between nodes measured in the number of hops, and *dist* is Euclidean distance between the virtual coordinates. Non-perimeter nodes calculate their virtual coordinates through an iterative relaxation procedure. Each non-perimeter node periodically updates its coordinates, such that the x coordinate of the node is the average of the x coordinates of all neighbours of the node, and the same for the y coordinate. The constructed virtual coordinates are used in greedy routing in the same way as geographical coordinates are used.

Newsome and Song [36] present an assignment of virtual coordinates that does not use flooding. The authors construct a tree with additional cross-links between nodes at the same height forming a ring, refereed to as a *ringed tree*. All nodes are assigned coordinates in a virtual polar coordinate space with the origin at the root node. A coordinate of a node is a pair containing the distance from the root of the tree to the node (i.e., the tree level), and a virtual angle range, which uniquely identifies a node within a level. A node is assigned virtual angle range by the parent of the node, with root node having virtual angle range 2π. A parent assigns each child a part of its virtual angle range, proportional to the number of nodes in the sub-tree rooted at the child node. Angles are assigned to children in such a way that they monotonically increase along the ring of nodes created by cross-links on a particular level of the tree. A node forwarding a message sends the message to the neighbour with the angle range closest to the angle range of the destination node, and if the node is local minimum, the message is sent to

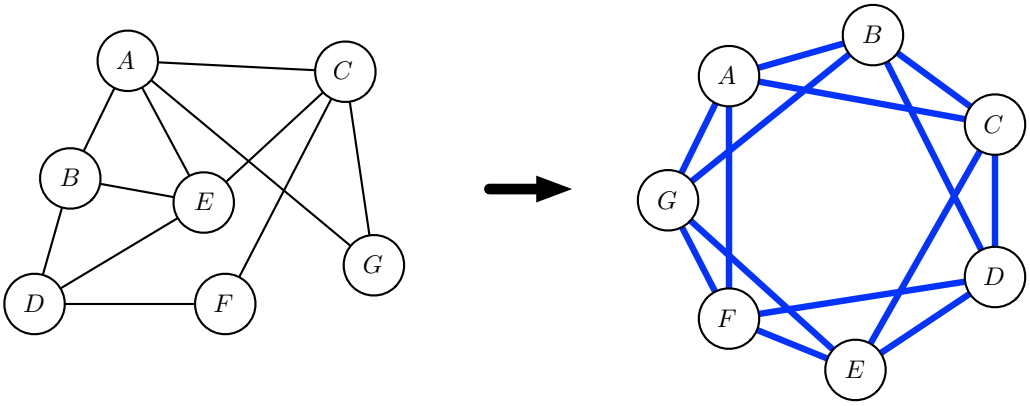
the parent of the node. The ringed tree contains all nodes that are present in the network, but only a subset of the links between nodes. When forwarding a message, a node considers only neighbours in the constructed ringed tree and not all of its neighbours in the substrate network.

Fonseca et al. propose Beacon Vector Routing (BVR) [35], that follows the ideas presented by Newsome and Song [36]. BVR also uses trees to assign virtual coordinates to nodes. However, when forwarding all neighbour nodes are considered and not only neighbours contained in the built trees. A small number of nodes in the network are selected to be *beacons*. Each beacon becomes a root of a spanning tree. Using the spanning trees, every node computes its distance to all beacons, where the distance from a node to a beacon is the level of the tree on which the node resides. A virtual coordinate of a node is a vector of distances from the node to all the beacons.

Herzen et al. [37] use spanning trees to create virtual coordinates for nodes that do not result in local minima, i.e., the authors achieve greedy embedding. The spanning trees are constructed and maintained only to determine virtual coordinates for nodes, and are not used for greedy routing. Distance between two nodes is measured as the smallest length of a shortest path between the nodes in any spanning tree. The authors prove properties of their routing scheme for a class of substrate networks with n nodes, where the number of neighbours a node has follows a power law distribution with parameter $2 < \gamma < 3$. In these networks, the proposed scheme achieves polylogarithmic scalability and the path stretch is bounded by $O(\log(n))$.



(a) Selecting node coordinates.



(b) Selecting links between nodes.

Figure 2.4: Greedy embedding.

The scheme of Herzen et al. achieves greedy embedding by selecting node coordinates. As illustrated in Fig. 2.4(a), the scheme starts by computing a spanning tree (links highlighted red) for a substrate network. Using the computed spanning tree, the scheme assigns coordinates to nodes to achieve greedy embedding. Greedy embedding can also be achieved by selecting links between nodes, instead of node coordinates, as illustrated in Fig. 2.4(b). We start with a substrate network, where nodes are already assigned coordinates. Greedy embedding is achieved by *creating* links between nodes, in such a way that no node is local minimum for any destination. Links between nodes are created as *overlay links*. Every overlay link is realized as a path of links in the substrate network. For a single substrate network, an overlay link can be realized as a path set up by the intra-substrate routing. In a multi-substrate network, a greedy routing scheme needs to set up and maintain paths that realize overlay links.

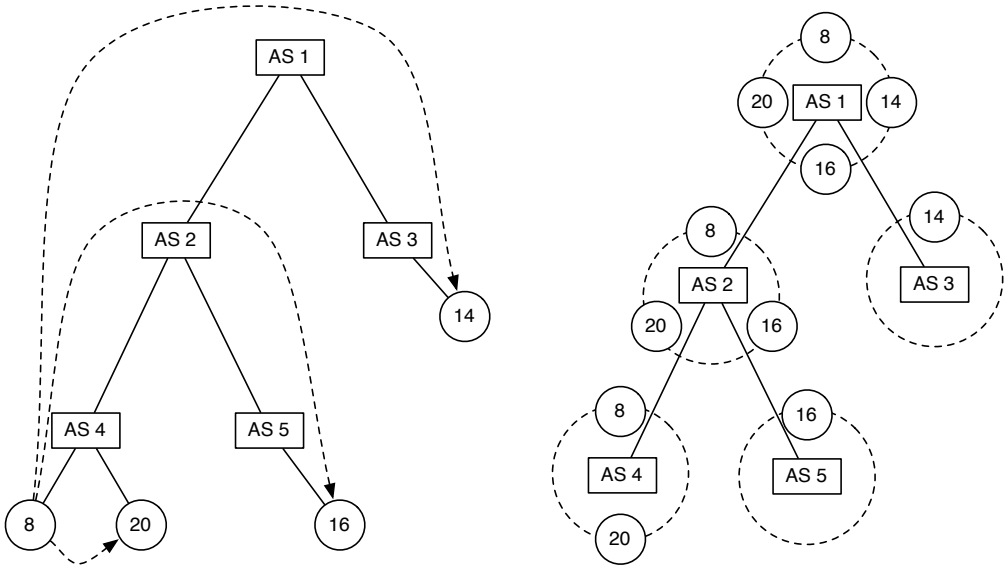
Routing schemes based on DHTs, such as Chord [8], Tapestry [38], and Pastry [10], are examples of greedy routing for a single substrate network, which accomplishes greedy embedding by creating overlay links between nodes.

Stoica et al. [8] propose a routing scheme called *Chord*. All nodes are assigned unique, m-bit long numerical identifiers drawn from a flat name-space, which are used as virtual coordinates. Identifiers are wrapped in a *Chord ring*, with every node storing addresses of its successor and predecessor. A node also has a *finger table*, which stores m entries. The first entry is for a node with identifier larger than the identifier of the owner of the finger table by at least one, i.e., the successor node. Second entry is for a node with identifier larger by at least two, and i^{th} entry is for a node with identifier larger by at least 2^{i-1} than the identifier of the node storing the finger table. For any two nodes A and B, with identifiers $id(A)$ and $id(B)$, the distance function is defined as $\delta(A, B) = |id(A) - id(B)|$. Messages are forwarded greedily, i.e., a node forwarding a message sends the message to the node from its finger table closest to the destination. The finger table provides possibilities to forward a message across the Chord ring, without traversing all the nodes on the ring in between. The authors prove that, for a network with n nodes, the number of nodes traversed when a message is forward is bounded by $O(\log n)$ with high probability, and the stored state per node is bounded by $O(\log n)$.

Chord creates links between nodes based only on node identifiers, without regard for the links in the substrate network. Zhao et al. [38] propose *Tapestry* as a greedy routing scheme, which considers the distance between nodes in the substrate network when creating overlay links between nodes. In Tapestry, every node is assigned a numerical identifier, which is a binary number with m digits. A routing table has m levels, and at i^{th} level stores pointers to nodes with identifiers that match the owner node's identifier in the last i digits (suffix), but have a different digit immediately preceding the suffix. Distance between two nodes is defined as the length of the longest common suffix of node identifiers. A node forwards a message to the node in the routing table closest to the destination. *Pastry* [10] proposed by Rowstron and Druschel operates similarly to Tapestry. Nodes are assigned 128-bit identifiers, viewed as sequences of digits in base 2^b. A node N keeps a *routing table*, with one row for each digit of the identifier and with 2^b entries in a row. Entries in row i are for nodes that have the same first i digits (prefix) as node N, and there is one entry for each of the 2^b possible digits at position $i + 1$. Node N also stores a *leaf set* of L_{Pastry} nodes with numerically closest identifiers to N, and a *neighbourhood set* of M_{Pastry} nodes that are closest to N in the substrate network. Distance between nodes is defined as the length of the longest common prefix of node identifiers.

The size of routing tables in Chord, Tapestry, and Pastry is bounded by $O(\log n)$ per node. The number of intermediate nodes traversed when routing a message is bounded by $O(\log n)$. In Chord, the

rules for filling finger tables define the exact node for which a finger table entry is created. In Tapestry and Pastry, there is more than one node that can be used to fill a routing table entry. Thus, when creating routing table entries, a node picks the node that is closest to it in the substrate network. As a result, although the number of nodes traversed to forward a message is bounded by the same value as in Chord, the routing paths between intermediate nodes are shorter in Tapestry and Pastry.



(a) Successors of node with identifier 8. (Redrawn from [39].)

(b) ROFL rings for network in Fig. 2.5(a).

Figure 2.5: ROFL rings for a hierarchy of ASes.

ROFL: Routing on Flat Labels [39] by Caesar et al., performs greedy routing on a multi-substrate network. ROFL views the multi-substrate network as a hierarchy of Autonomous Systems (ASes), connected by routers. As in Chord [8] nodes have numerical identifiers, which are wrapped in a ROFL ring. Since ROFL does not assume routing between ASes exist, each node maintains a source route to its predecessor and successor, as a hop-by-hop sequence of one-hop connected routers. One ROFL ring is built for each AS, starting from leaf ASes. ROFL ring of a leaf AS contains all nodes connected to the AS. If an AS has children ASes, the ROFL ring is built by merging ROFL rings of child ASes. For illustration, Fig. 2.5 shows a network with five ASes and four nodes with identifiers: 8, 14, 16, and 20. The node with identifier 8 participates in the ROFL ring of AS 4, together with the node with identifier 20. The ROFL ring of AS 2 is created by merging the rings of child ASes: AS 4 and AS 5. For the node with identifier 8, this means that it gets the node with identifier 16 as a successor in the ROFL ring of AS 2. The ROFL ring of AS 1 is created by merging rings of AS 2 and AS 3, and in AS 1, node 8 has as a successor node 14. By using greedy routing a message sent between two ASes will never traverse higher than the least-common ancestor AS in the hierarchy of ASes.

Ford [13, 40] proposes Unmanaged Internet Protocol (UIP) as a greedy routing scheme for a general multi-substrate network. In UIP, nodes are assigned numerical identifiers viewed as binary strings. A node has an overlay link to at least one node for each possible length of the longest common prefix, i.e., to a node with longest common prefix of length zero, one, two, ..., up to the identifier length. If two nodes connected by an overlay link are in different substrate networks, they cannot use a path set up

by the intra-substrate routing to realize the overlay link. Therefore, UIP introduces *virtual links* as a mechanism for two nodes to communicate by using a relay node. Consider nodes A and B attached to substrate networks S_1 and S_2 respectively and node R attached to both substrate networks. Node A establishes a virtual link to node B by storing information that node B is reachable via node R. Messages from node A to node B are forwarded on the virtual link in two parts: (1) from A to R using a path set up by the intra-substrate routing in S_1 and (2) from R to B using a path set up by the intra-substrate routing in S_2. Once a virtual link is established, it can be used to build further virtual links. Virtual links allow UIP to establish overlay links between nodes that are not in the same substrate network.

Caesar et al. [14] propose *Virtual Ring Routing* (VRR) as a greedy routing scheme that is implemented directly on top of the data link layer, i.e., VRR does not assume existence of intra-substrate routing. Nodes in VRR are assigned numerical identifiers and ordered in a circle, with wrapping around zero, referred to as a *virtual ring*. Each node N stores a *virtual neighbour set* (*vset*) of the $\frac{r_{VRR}}{2}$ closest nodes with identifiers smaller and the $\frac{r_{VRR}}{2}$ closest nodes with identifiers larger than the identifier of node N. Each node also stores a *physical neighbour set* (*pset*) of all nodes it can communicate with at the data link layer, i.e., one-hop connected nodes. Since VRR assumes no routing in the substrate networks, VRR introduces *virtual links* as a mechanism to enable communication between a node and nodes in its *vset*. Virtual links in VRR follow the same ideas as virtual links in UIP: by concatenating existing links between nodes, new links are established. If links $(N_0, N_1), (N_1, N_2), \ldots, (N_{i-1}, N_i)$ exist, they can be concatenated to a create virtual link (N_0, N_i), by inserting routing table entries for the virtual link at nodes $N_0, N_1, \ldots, N_i$. Nodes N_0 and N_i are referred to as end-point nodes and nodes $N_1, N_2, \ldots, N_{i-1}$ as intermediate nodes of a virtual link. The inserted routing table entries store information bout the next-hop nodes for both end-points of the virtual link. A routing table stores entries for all nodes in the *vset* and *pset* and also for end-point nodes of all virtual links for which the node is an intermediate node. Routing is done greedily by selecting the node from the routing table that has an identifier that is numerically closest to the destination node identifier. Adding end-points of virtual links to the routing tables of intermediate nodes provides routing shortcuts, similar to fingers in Chord [8]. Using only nodes from *vset* for routing, each message travels along the virtual ring, moving at most $\frac{r_{VRR}}{2}$ identifiers between two nodes. End-point nodes of virtual links passing through an intermediate node have identifiers that depend on the connectivity in the substrate network. As such, it is expected that the identifiers of end-point nodes are randomly distributed along the identifier space. Thus, when considering end-point nodes during forwarding, a message can move more than $\frac{r_{VRR}}{2}$ identifiers in one hop.

2.3 Two-level Routing

Recall from Section 1.2.2 that in two-level routing nodes are organized into groups and routing paths are set up between groups of nodes. Nodes in a group are assumed to be able to communicate to each other. Hinden [41] proposes to apply two-level routing to the Internet, with the goal of reducing the size of routing tables stored at routers. Nodes are grouped into *autonomous domains* (ADs), which are assigned IP addresses from a set of designated addresses for ADs. Routing sets up paths between ADs. Before forwarding a message, DNS is used to obtain the address of the AD in which the destination node resides. Next, the message is encapsulated with an additional header containing addresses of the

source and destination ADs. The encapsulated message is forwarded using the added header and the destination AD address. When a message enters the destination AD the added header is removed and the original message is delivered to the destination node.

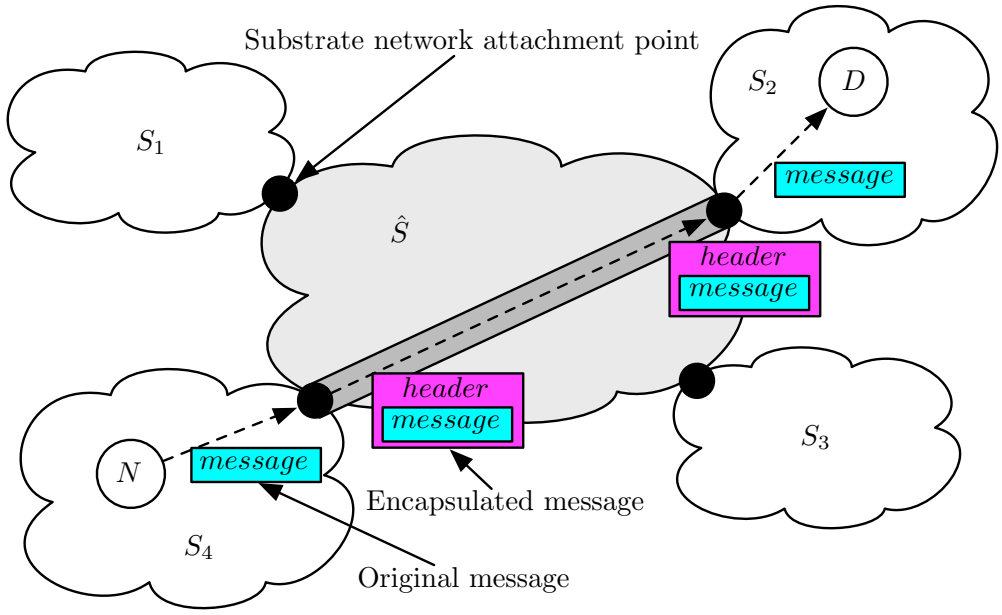


Figure 2.6: Multi-substrate network with $\hat{S}$.

Ideas about encapsulation presented by Hinden may be applied to routing between substrate networks, which are all connected to a single substrate network, referred to as $\hat{S}$. For example, the Internet is $\hat{S}$ and the private networks connected through NAT devices to the Internet are the other substrate networks. Places where substrate networks connect to $\hat{S}$ are referred to as *substrate network attachment points*. Each substrate network attachment point has an address in $\hat{S}$, which is used as the address of the substrate network. Routing between substrate networks is provided by $\hat{S}$. Messages entering $\hat{S}$ are encapsulated with a header appropriate for $\hat{S}$, with the destination set to the address of the destination substrate network. For illustration, Fig. 2.6 shows $\hat{S}$ with four substrate networks S_1 through S_4. A message from node N in substrate network S_4 is forwarded to node D in substrate network S_2. The original message is encapsulated when entering $\hat{S}$ and then forwarded in $\hat{S}$ using the added header. When the message reaches S_2, the message is decapsulated by removing the added header, and the original message is delivered to node D. Using encapsulation to forward messages through $\hat{S}$ is the central idea in several proposals for routing between substrate networks, e.g., LISP [42], ILNP [43], RANGI [44], and IPNL [45].

Farinacci et al. propose Locator/ID Separation Protocol (LISP) [42], which uses encapsulation in $\hat{S}$. The authors assume all substrate networks use IP routing. A node address is composed of: (1) the address of the substrate network attachment point in $\hat{S}$ and (2) the address of the node in the substrate network. Messages are forwarded by the IP routing through $\hat{S}$ using destination substrate network address and in the destination substrate network by using the destination node address.

Randall and Saleemthe also use encapsulation to forward messages in $\hat{S}$ in Identifier-Locator Network Protocol (ILNP) [43]. Each node is assumed to have an identifier, which is independent of the substrate

network in which the node resides. A node address is composed of: (1) the address of the substrate network attachment point in $\hat{S}$ and (2) the node identifier. Each substrate network is assumed to have an identifier resolution service, which is used to resolve node identifiers into addresses. When a forwarded message is decapsulated in the destination substrate network, identifier of the destination node is resolved into an address. The address is used to complete the forwarding via a path set up by the intra-substrate routing.

Routing Architecture for the Next Generation Internet (RANGI) [44] by Xu assumes $\hat{S}$ with IPv6 routing and the other substrate networks with IPv4 routing. Node addresses are IPv4-embedded IPv6 addresses, where the leftmost 96 bits identify a substrate network. The remaining 32 bits are an IPv4 address, which is unique within the substrate network. Messages are forwarded through $\hat{S}$ by using the first 96 bits of the address as an IPv6 address of the destination. In the destination substrate network, messages are forwarded by using the last 32 bits of the address as an IPv4 address.

Francis and Gummadi propose IP Next Layer (IPNL) [45], where each node has a unique identifier, which is a Fully Qualified Domain Names (FQDN). The authors assume $\hat{S}$ exists, however, substrate networks may connect to $\hat{S}$ through other substrate networks. Node addresses are short, fixed-length, numerical fields consisting of three parts:

(1) a globally unique IP address of the substrate network attachment point in $\hat{S}$,

(2) a *Realm Number* (RN) identifying the substrate network where the node resides, and

(3) an IP address of the node that needs to be valid only in its substrate network.

Messages are forwarded first through $\hat{S}$ and then forwarded to the destination substrate network based on the RN. Forwarding is completed by using the paths set up by the intra-substrate routing and the IP address of the destination node.

4+4 [46] and GSE [47] assume $\hat{S}$ with directly attached substrate networks. Instead of using encapsulation to forward messages through $\hat{S}$, the authors modify message headers to contain two source and two destination addresses. 4+4 considers the public Internet as $\hat{S}$, while the substrate networks are behind NAT devices and use private IPv4 addresses. Message headers contain both private addresses of nodes and public addresses of NAT devices. Only one pair of source and destination addresses is used at a time when forwarding. NAT devices swap the addresses as messages travel between the public and the private address spaces. That is, the public address of the NAT device of the destination substrate network is used when forwarding in $\hat{S}$, and the private address of the destination node is used when forwarding in the destination substrate network. GSE assumes that $\hat{S}$ and all other substrate networks use IPv6 addresses. To forward messages between the substrate networks, GSE changes the semantics of the IPv6 address. The first eight bytes are interpreted as the address of the attachment point of a substrate network and the last eight bytes are the address of a node in the substrate network. Therefore, when forwarding in $\hat{S}$ the first eight bytes are used as the destination address and in the other substrate networks the last eight bytes are used as the address of the destination.

The assumption of a single substrate network $\hat{S}$, to which all other substrate networks connect, is relaxed by Ahlgren et al. [48] and Feldmann et al. [49]. They replace $\hat{S}$ with a hierarchy of substrate networks, referred to as $\hat{\mathbb{S}}$. Fig. 2.7 illustrates such a network, with a three-level hierarchy of $\hat{\mathbb{S}}$. For example, this hierarchy can be created by customer-provider relation between Internet Service Providers (ISPs), or created dynamically as Schmid et al. propose in TurfNet [50]. By turning a large substrate

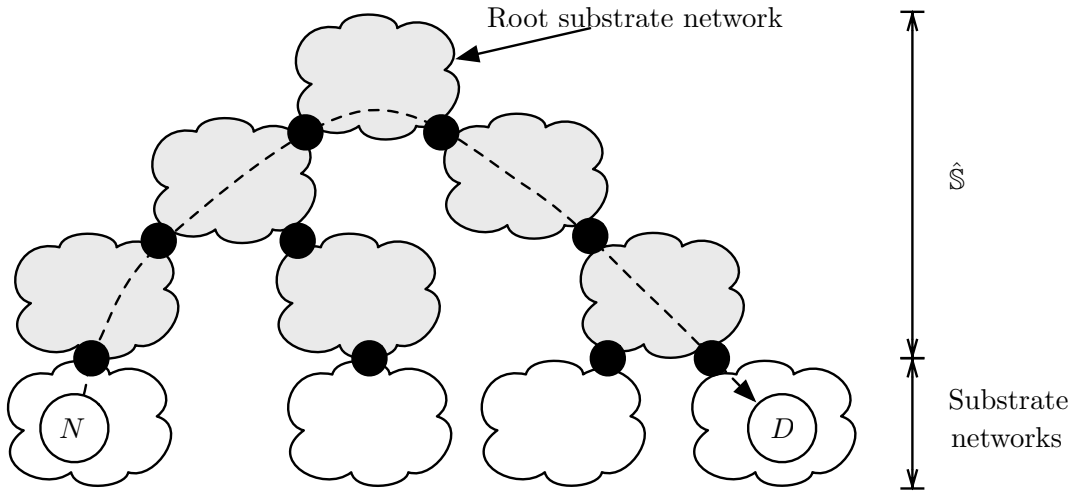


Figure 2.7: Multi-substrate network with $\hat{\mathbb{S}}$.

network into a hierarchy of substrate networks, the size of routing tables at nodes is reduced. A node is only required to store information about paths to other nodes in its substrate network and an additional path to the parent substrate network in $\hat{\mathbb{S}}$. Routing between substrate networks is realized using source routes based on substrate networks. A source route for a node in a substrate network, say S_1, contains substrate networks on a path from the root substrate network to S_1. Since each node stores information about a path to its parent substrate network, a message can be easily forwarded to the root substrate network. A message is forwarded from the root substrate network to the destination substrate network using a source route. Messages are forwarded through substrate networks in $\hat{\mathbb{S}}$ using encapsulation.

Ahlgren et al. [48] assume a small number of stable substrate networks which make up $\hat{\mathbb{S}}$, to which all other substrate networks attach through routers. Every node and router has a globally unique identifier, referred to as *Node Identifier* (NID). Each router stores mappings from FQDNs to NIDs to addresses of nodes in its substrate network. Routers participate in a single DHT, which enables them to discover each other's addresses, and also participate in a network wide DNS. Before sending a message, a node uses DNS to resolve FQDN of the destination node into the NID of the destination node and the NID of the router of the destination substrate network. A node sends a message to its own router, which uses the DHT to resolve the NID of the destination router into an address. A message is forwarded to the destination router through substrate networks in $\hat{\mathbb{S}}$. The destination router uses stored mappings to obtain the address associated with the NID of the destination node, and delivers the message to the destination node.

Feldmann et al. propose Hierarchical Architecture for Internet Routing (HAIR) [49], which uses a hierarchy of substrate networks, with an arbitrary number of levels of hierarchy. HAIR assumes a stable core substrate network (CORE) and several levels of smaller access or enterprise substrate networks (INT), which together make up $\hat{\mathbb{S}}$. Edge substrate networks (EDGE) connect to INT substrate networks. A node's address is a source route from the CORE towards a node, recording substrate networks on the path. All nodes store a path to their parent substrate network, used to forward messages to the CORE substrate network. From the CORE substrate network to the destination node, messages are forwarded

using source routes.

Schmid et al. [50] propose TurfNet, which assumes no particular hierarchy of substrate networks, but rather creates its own hierarchy. The substrate networks are called *TurfNets* and are connected through gateways nodes. Every TurfNet has a logical entity, called *TurfControl*, that is responsible for TurfNets essential services, such as, address allocation and resolution, routing, and turf composition. Turf composition allows independent TurfNets to create new TurfNets. That is, when TurfNets tn_A and tn_B compose to form TurfNet tn_C, the other TurfNets see only tn_C, without knowing which TurfNets are contained within tn_C. The TurfNets form a hierarchy of tiers, where TurfNets can be connected to other TurfNets in their own tier and to the TurfNets in tiers immediately above and bellow. Each node has a globally unique identifier and registers it with TurfControl to establish a mapping from the identifier to its local address in the TurfNet. All registrations are propagated from the child TurfNet to the parent TurfNet, until the top tier of the hierarchy is reached. The TurfControls along the path set state that maps registered identifier to the TurfNet from where the registration request came, thus setting forwarding state to forward message from the top tier down to the registered node. TurfNet uses name resolution to discover routing paths, however unlike [48], the name resolution does not return a path description. Instead, a resolution message is sent from TurfControl to the TurfControl in the parent TurfNet, until a TurfNet, whose TurfControl knows about the identifier of the destination node, is reached. The resolution message is returned to the sender node using state stored when the sender node registered its identifier. As the message travels to the sender node, state is set in TurfControls of the intermediate TurfNets. This state maps the destination nodes identifier to the TurfNet from which the resolution message arrived. Thus, when the resolution is completed, a path from source to destination node is set up in TurfControl of all TurfNets on the path.

Clark et al. in FARA [51] and Crowcroft et al. in Plutarch [52] argue that a multi-substrate network should be viewed as a collection of substrate networks without any inherent hierarchy. In FARA, messages are forwarded by placing *forwarding directives* in message headers. A forwarding directive is defined as the information needed to cause eventual delivery of a message to the destination. Substrate networks are traversed using substrate specific forwarding directives, e.g., an IPv4 address on the public Internet. A forwarding directive from a source to a destination is a concatenation of substrate forwarding directives. The authors provide specification of forwarding directives for a multi-substrate network with a single substrate network $\hat{S}$. Each nodes is assumed to be able to obtain a forwarding directive from it self to $\hat{S}$. Then a forwarding directive from a node A to a node B is concatenation of: (1) a forwarding directive from A to $\hat{S}$, and (2) a forwarding directive from $\hat{S}$ to B, obtained by reversing the forwarding directive from B to $\hat{S}$. Plutarch also describes message forwarding on an abstract level. A message can be forwarded between two nodes if a chain of substrate networks between them is known to the sender node. The authors suggest an epidemic-style gossiping mechanism to discover chains of substrate networks. Neither FARA nor Plutarch proposes a concrete mechanism for routing between substrate networks.

Chapter 3

Landmark Domains Routing

Landmark domains routing provides forwarding of messages between nodes in a multi-substrate network. Nodes are organized into *reachability domains*, which are sets of nodes that can communicate using the paths set up by the intra-substrate routing. Landmark domains routing sets up routing paths between reachability domains. A subset of reachability domains is designated as *landmark domains*. Each node stores information about a next-hop node on a path to every landmark domain. All nodes participate in the next-hop forwarding of messages to landmark domains. From a landmark domain to a destination node, messages are forwarded by using source routes. Each node in landmark domains routing has a locator, which contains a source route starting with a node in a landmark domain. Locators are used to designate message destinations when forwarding. A node discovers its locator by forwarding a special locator discovery message to the closest landmark domain and by recording the path the message takes. Since we assume that all links are bidirectional, a source route from the landmark domain to the node is the revers of the path that the message took.

The rest of this chapter starts by presenting how reachability domains are detected, identified, and maintained. Next, we explain landmark domains and node locators. Details of message forwarding are presented in the following section. The chapter concludes with a presentation of locator discovery.

3.1 Reachability Domains

A reachability domain (RD) is a set of nodes, attached to the same substrate network, such that a path exists between every pair of nodes in the set. We assume that intra-substrate routing sets up paths between nodes in a reachability domain. Before starting discovery of the routing paths between reachability domains, landmark domains routing needs to detect reachability domains. To detect reachability domains nodes select a unique identifier for each reachability domain in the multi-substrate network. RD identifiers are numbers drawn from a flat name-space.

Two nodes are members of the same RD if: (a) they are attached to the same substrate network, say S_1; and (b) if they can exchange messages in substrate network S_1 (possibly using intra-substrate routing in S_1). Assuming that a substrate network offers message broadcast, a node can detect other members of an RD in the substrate network with a broadcast of an RD discovery message in the substrate network. All nodes that receive the RD discovery message belong to the same reachability domain as the sender. For illustration, Fig. 3.1 shows substrate network S_1 with nodes A through I. Nodes A through

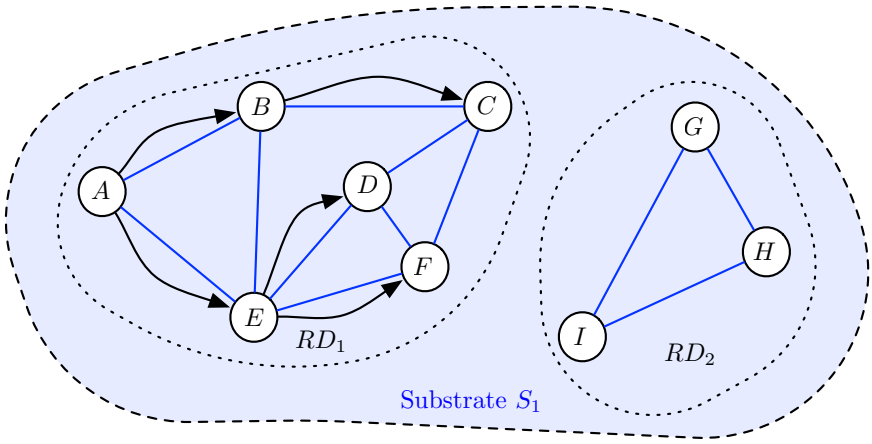


Figure 3.1: Detection of reachability domain members in a broadcast substrate network.

F belong to reachability domain RD_1 and nodes G, H, and I belong to reachability domain RD_2. An RD discovery message broadcast by any node in RD_1 reaches all other nodes in RD_1, as illustrated in Fig. 3.1 for node A broadcasting an RD discovery message. Since there is no path between nodes from RD_1 and RD_2 in substrate network S_1, an RD discovery message broadcast by a node in RD_1 does not reach nodes in RD_2. In substrate networks that do not offer message broadcast, RD discovery messages may be flooded instead. When flooding a message, each node sends a copy of the message to all of its neighbours, apart from the neighbour from which the message arrived. A node may receive several copies of the same message during flooding. To avoid unnecessary duplication of messages and to reduce message overhead, landmark domains routing uses per-substrate overlay networks for broadcasting. In each substrate network nodes try to build an overlay network, using only paths set up by intra-substrate routing. Since only nodes in the same reachability domain can communicate, nodes build a separate overlay network for each reachability domain. By broadcasting an RD discovery messages in an overlay network a node can detect other nodes in the same RD. The overlay network is used only to broadcast control messages in an RD and is not used in message forwarding. Messages between nodes in the same RD are forwarded via paths set up by the intra-substrate routing.

The identifier of a reachability domain, referred to as *RD-ID*, is a random number selected by one of the members of an RD. To bootstrap construction of an RD, every node picks a random number and broadcasts the number by using the overlay network. The largest number is picked as the RD-ID of the reachability domain and the node that broadcasts the selected RD-ID becomes the *leader node* for the RD.

Node membership in reachability domains may change as links between nodes fail or new links become available. To detect changes in RD membership, each leader node periodically broadcasts an RD-ID. All nodes record the last time they received an RD-ID from the leader node. If a node has not recently received an RD-ID from the leader node, the node concludes it is no longer in the same RD as the leader node. A node may not receive an RD-ID for two reasons: (1) the leader node has failed, or (2) an intra-substrate path is no longer available to the leader node. In both cases, a node that detects it is no longer in the same RD as the leader node assumes to be the sole member of a new RD. The node bootstraps construction of a new RD by broadcasting a randomly selected number and assuming the

role of the leader node.

Broadcasting RD-IDs allows nodes to merge RDs that belong to the same substrate network. Two RDs are merged into a single RD when a node from one RD detects that a path in a substrate network is available to a node from a different RD. Nodes from both RDs construct a single overlay network. Leader nodes from each RD continue to broadcast different RD-IDs on the single overlay network. The larger RD-ID is selected as the RD-ID of the merged reachability domain. The leader broadcasting the smaller RD-ID stops broadcasting its own (smaller) RD-ID and joins the RD with the larger RD-ID. Eventually, nodes from both RDs receive only a single RD-ID and are members of the same RD.

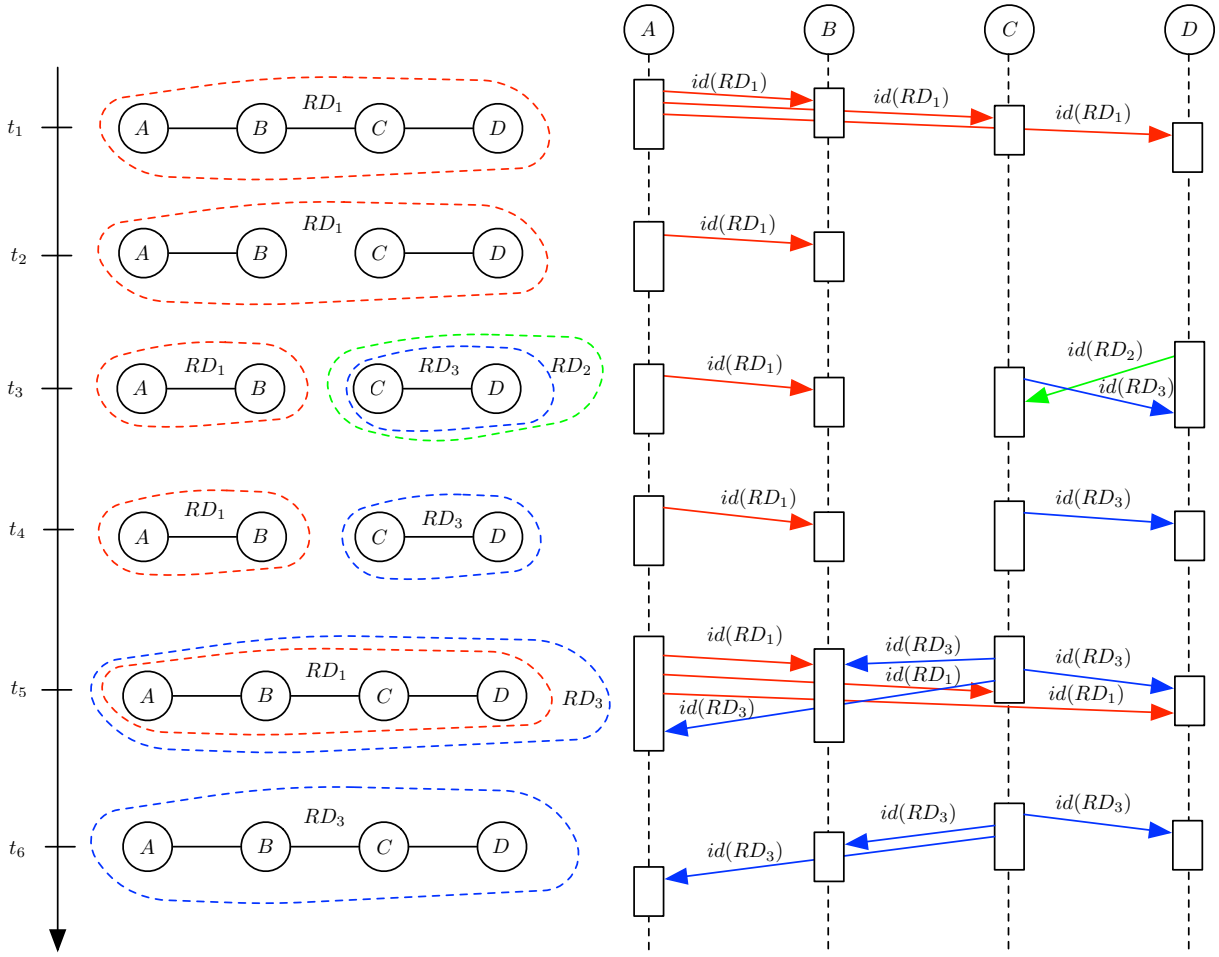


Figure 3.2: Detection of reachability domain split and merge.

To illustrate the detection of reachability domains splitting and merging, Fig. 3.2 shows a substrate network with four nodes, over time. The figure shows broadcast of RD-IDs on the right hand side. Initially, at time t_1, all four nodes are in reachability domain RD_1. Node A is the leader node and broadcasts RD-ID $id(RD_1)$. At time t_2, the link between nodes B and C fails and RD-ID broadcast of node A reaches only node B. At time t_3 nodes C and D detect they stopped receiving RD-ID from node A. Both node C and D assume that they are the sole members of new RDs. The nodes pick new RD-IDs and start broadcasting them. Assuming $id(RD_3)$, picked by node C, is larger than $id(RD_2)$, picked by node D, $id(RD_3)$ becomes the new RD-ID. By time t_4, nodes C and D have agreed on RD-ID

$id(RD_3)$ and node C, as the leader node, broadcasts the RD-ID. A time t_5 the link between nodes B and C becomes available again. Both leader nodes A and C broadcast their RD-IDs to all other nodes in the substrate network. Assuming $id(RD_3) > id(RD_1)$, nodes A and B join RD_3. Node A stops broadcasting $id(RD_1)$ and node C is the only leader node.

3.2 Landmark Domains and Locators

An objective of landmark domains routing is to set up paths between reachability domains. Shortest paths between RDs may be set up by performing shortest path routing. To store the information about the shortest paths to all RDs, each node requires a routing table of size proportional to the number of RDs in the multi-substrate network. Therefore, shortest path routing may not be viable for multi-substrate networks with a large number of RDs. To reduce the size of routing tables, nodes in landmark domains routing set up paths only to a subset of reachability domains, called landmark domains. We select landmark domains uniformly at random from the set of all reachability domains. Every node has a landmark domains routing table, which stores next-hop nodes for paths to all landmark domains and distances to landmark domains. Let $\delta : \mathcal{N} \times \mathcal{N} \to \mathbb{R}_0^+$ be a function measuring distance between two nodes from a set $\mathcal{N}$ of nodes in a multi-substrate network. We express the distance from a node to a landmark domain by a function $\delta' : \mathcal{N} \times \left(2^{\mathcal{N}} \setminus \{\emptyset\}\right) \to \mathbb{R}_0^+$ such that

$$\delta'(N, \mathcal{L}) = \min_{L \in \mathcal{L}} \{\delta(N, L)\}.$$

For any node N and any landmark domain $\mathcal{L}$, the distance from N to $\mathcal{L}$ is the minimal distance from N to any node in landmark domain $\mathcal{L}$.

Recall that each node has a locator, which contains a source route from a node in a landmark domain to the node. A locator also contains the RD-ID of the landmark domain in which the source route starts. Landmark domains routing tables are used to forward a message to the landmark domain given in the locator of the destination node. The source route in the locator is used to forward a message from the landmark domain to the destination. A locator of a node N is represented as $\{id(\mathcal{L}) : sr(\mathcal{L}, N)\}$, where $id(\mathcal{L})$ is the RD-ID of a landmark domain $\mathcal{L}$ and $sr(\mathcal{L}, N)$ is a source route from any node in $\mathcal{L}$ to node N. A source route $sr(\mathcal{L}, N)$, containing m nodes, has a format:

$$\Big(\langle id(RD^{(1)}), sa(N^{(1)})\rangle, \quad \langle id(RD^{(2)}), sa(N^{(2)})\rangle, \quad \ldots, \langle id(RD^{(m)}), sa(N^{(m)})\rangle\Big),$$

where superscripts indicate position of RDs and nodes in the source route. The first RD in the source route is the landmark domain $\mathcal{L}$ and last node in the source route is node N, i.e., $RD^{(1)} = \mathcal{L}$ and $N^{(m)} = N$. Substrate address of node $N^{(i)}$ in substrate network of reachability domain $RD^{(i)}$ is indicated with $sa(N^{(i)})$. Any node in $RD^{(i)}$ can send a message to node $N^{(i)}$ using the substrate address $sa(N^{(i)})$ and the intra-substrate routing. We require, for $1 \leq i < m$, that node $N^{(i)}$ is a member of reachability domain $RD^{(i+1)}$, i.e., node $N^{(i)}$ can send a message to node $N^{(i+1)}$ by using the intra-substrate routing. When forwarding a message on a source route, node $N^{(i)}$ uses $id(RD^{(i+1)})$ to select an outgoing interface connected to the substrate network in which $RD^{(i+1)}$ is defined. In this substrate network, node $N^{(i)}$ can use $sa(N^{(i+1)})$ to forward a message to node $N^{(i+1)}$. We use the property of reachability domains that all members can exchange messages using the intra-substrate routing to reduce the length of the source route. We define a *partial source route* as a source route where each subsequence of nodes with

the same RD-ID is replaced by the last node in the sequence. For example, consider a source route:

$$\Big(\langle id(RD_1), sa_1(N_1)\rangle, \langle id(RD_2), sa_2(N_2)\rangle, \langle id(RD_2), sa_2(N_3)\rangle, \langle id(RD_2), sa_2(N_4)\rangle, \Big),$$

where $sa_i(N_j)$ is the substrate address of node N_j in substrate network of reachability domain RD_i. This source route can be replaced by partial source route:

$$\Big(\langle id(RD_1), sa_1(N_1)\rangle, \langle id(RD_2), sa_2(N_4)\rangle, \Big),$$

since node N_1 is a member of RD_2 and thus can forward messages to node N_4, which is also in RD_2.

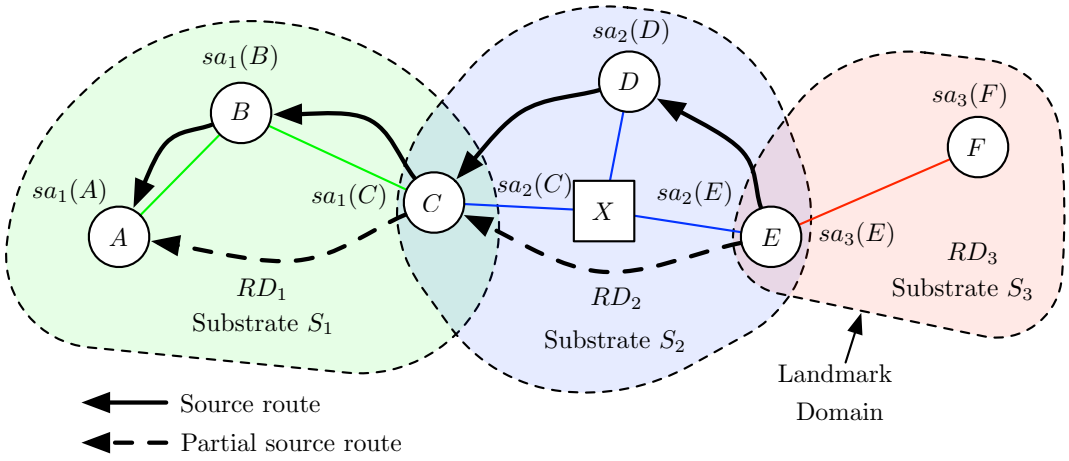


Figure 3.3: Source route and partial source route.

To illustrate the usage of node locators, Fig. 3.3 shows three substrate networks S_1, S_2, and S_3. Each substrate network is also a reachability domain. We assume that reachability domain RD_3 is a landmark domain. Circles represent nodes (A through F) that participate in the landmark domains routing, while squares represents devices that do not participate. The locator of node A is $\{id(RD_3) : sr(RD_3, A)\}$. Solid arrows show $sr(RD_3, A)$ realized as a full source route and dashed arrows show $sr(RD_3, A)$ realized as a partial source route. The full source route is ($\langle id(RD_3), sa_3(E)\rangle$, $\langle id(RD_2), sa_2(D)\rangle$, $\langle id(RD_2), sa_2(C)\rangle$, $\langle id(RD_1), sa_1(B)\rangle$, $\langle id(RD_1), sa_1(A)\rangle$) and the partial source route is ($\langle id(RD_3), sa_3(E)\rangle$, $\langle id(RD_2), sa_2(C)\rangle$, $\langle id(RD_1), sa_1(A)\rangle$).

3.3 Message Forwarding

The forwarding process in landmark domains routing consists of three distinct parts:

(1) Next-hop forwarding from the source node to the landmark domain in the destination node's locator. A node forwarding a message obtains the substrate address of the next-hop node from the landmark domains routing table. A message is forwarded to the next-hop node via a path set up by the intra-substrate routing.

(2) Forwarding inside the landmark domain to the first substrate address in the source route.

(3) Forwarding from the landmark domain to the destination node, using the source route and paths set up by the intra-substrate routing between the nodes included in the source route.

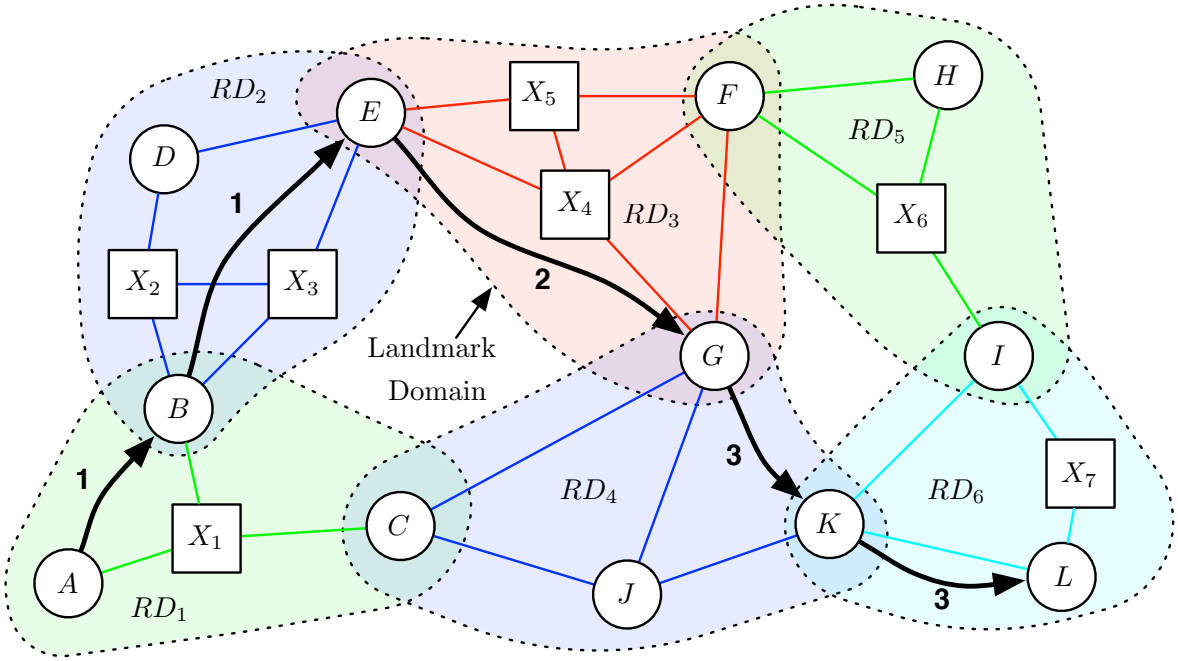


Figure 3.4: Landmark domains routing.

To illustrate message forwarding, Fig. 3.4 shows a multi-substrate network with six reachability domains, where reachability domain RD_3 is selected as the landmark domain. Circles represent nodes participating in landmark domains routing, while squares represent other devices that do not participate. The locator of node L is:

$$\{id(RD_3) : (\langle id(RD_3), sa_3(G)\rangle, \langle id(RD_4), sa_4(K)\rangle, \langle id(RD_6), sa_6(L)\rangle)\}\,.$$

We explain forwarding of a message from node A to node L.

(1) Node A obtains substrate address $sa_1(B)$, as the next-hop node on a path to landmark domain RD_3, from the landmark domains routing table. Using $sa_1(B)$, the message is delivered to node B. Node B obtains $sa_2(E)$ from the landmark domains routing table and forwards the message to next-hop node E.

(2) As ingress node to the landmark domain, node E needs to forward the message to the egress node G, i.e., the first node in the source route. Thus, node E uses intra-substrate routing to forward the message to the first substrate address $sa_3(G)$ in the source route.

(3) From the source route, node G obtains the substrate address $sa_4(K)$ of the next node on the path towards the destination. Node G relies on the intra-substrate routing to perform multi-hop forwarding to node K. Node K forwards the message to the destination node using the intra-substrate routing and the last substrate address $sa_6(L)$ in the source route.

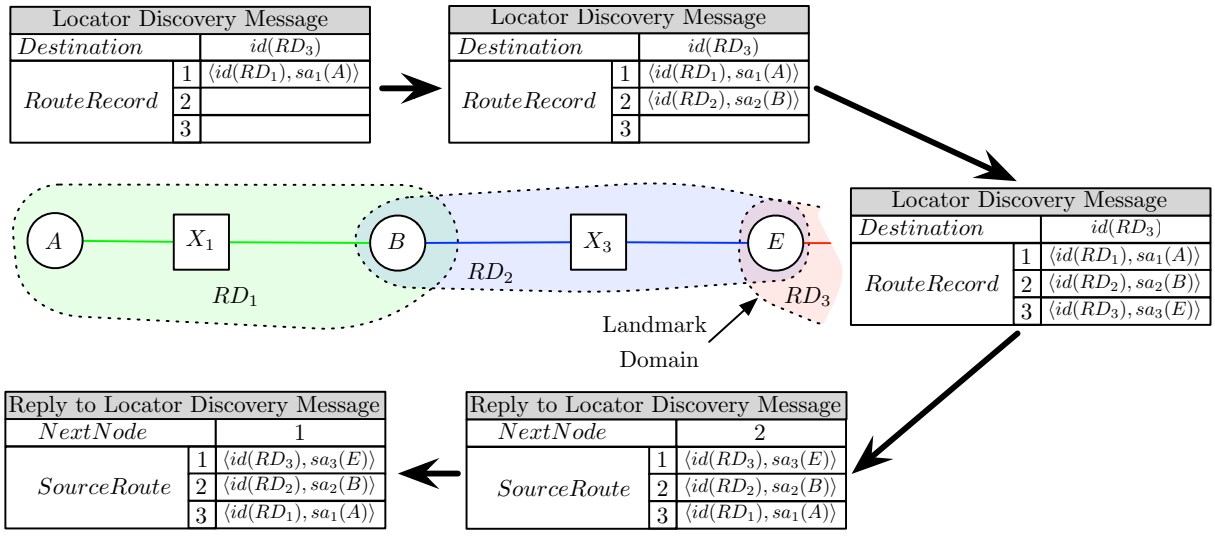


Figure 3.5: Locator discovery.

3.4 Locator Discovery

In the landmark domains routing, a node is responsible to obtain its locator and update the locator as the node moves in the network. A node obtains a locator by routing a *locator discovery* message towards a landmark domain. To keep the size of partial source routes small, a node sending a locator discovery message selects the closest landmark domain from the landmark domain routing table as the destination landmark domain. Locator discovery messages have a *RouteRecord* field, which is filled by nodes forwarding the message. Any node N forwarding a locator discovery message via some reachability domain RD_i appends $\langle id(RD_i), sa_i(N)\rangle$ to the *RouteRecord* field, where $sa_i(N)$ is the substrate address of node N in reachability domain RD_i. When a locator discovery message reaches a node in the landmark domain, the node appends the RD-ID of the landmark domain and the substrate address of the node in the landmark domain. Recall from Section 1.1 that we assume all links are bidirectional. Thus, by reversing the information in the *RouteRecord*, the node in the landmark domain obtains a partial source route for the node that sent the locator discovery message. The node in the landmark domain uses the partial source route to forward a reply to the locator discovery message to the sender node.

For illustration, Fig. 3.5 shows a part of the network from Fig. 3.4. Node A sends a locator discovery message to landmark domain RD_3, shown in top of the figure. Destination of the message is set to the RD-ID of landmark domain RD_3, $id(RD_3)$. From the landmark domains routing table node A obtains substrate address $sa_1(B)$ as the next-hop node in RD_1 for the path to the landmark domain RD_3. Node A adds $\langle id(RD_1), sa_1(A)\rangle$ to the *RouteRecord* field, and forwards the locator discovery message to node B using a path set up by the intra-substrate routing. Node B records its RD-ID and substrate address $\langle id(RD_2), sa_2(B)\rangle$ and forwards the message to the next node on the path to RD_3. The locator discovery message reaches node E in landmark domain RD_3, which adds $\langle id(RD_3), sa_3(E)\rangle$ to the *RouteRecord* field. Node E creates a replay to the locator discovery message, shown in Fig. 3.5 at the bottom. $NextNode$ filed indicates which node in the $SourceRoute$ field is the next node that receives the message. Node E fills the $SourceRoute$ field with the reversed route record from the *RouteRecord* field in the locator discovery message and sets the *next* field to 2, corresponding to node B in source

route. The reply to the locator discovery message is returned to node A, by following the source route. Node A can construct its locator using the source route in the message as:

$$\{id(RD_3) : (\langle id(RD_3), sa_3(E)\rangle, \langle id(RD_2), sa_2(B)\rangle, \langle id(RD_1), sa_1(A)\rangle)\}\,.$$

Chapter 4

Implementation

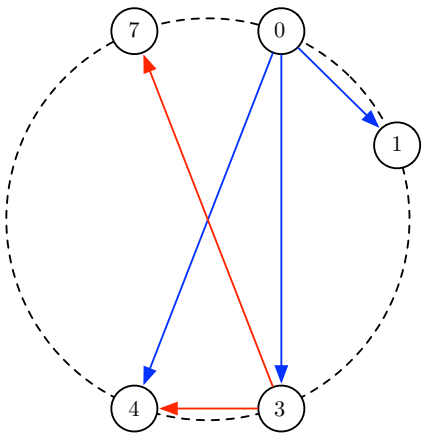
In this chapter we present the implementation of landmark domains routing. We start by explaining building of the overlay networks, which help nodes determine their membership in reachability domains. Next, we explain how nodes broadcast messages in the overlay networks. The following section describes how nodes agree on an RD-ID for a reachability domain. The final section details discovery of paths to landmark domains.

4.1 Overlay Networks

Recall from Section 3.1 that in each substrate network nodes try to build an overlay network. Nodes in each reachability domain organize into a separate overlay network, and use the overlay network to agree on an RD-ID. The leader node periodically broadcasts the RD-ID of a reachability domain to the other members, via the overlay network. We have selected to realize overlay networks as Chord [8] rings, which have mechanisms to deal with changes of connectivity between nodes and node failures. In Chord, all nodes that participate in a Chord ring are assumed to be members of a same substrate network and have a single substrate address. Therefore, all nodes trying to build a Chord ring in a substrate network, say S_1, use only their substrate addresses from substrate network S_1.

In Chord, every node N has an m-bit long numerical identifier, $id(N)$, which determines node's position in the Chord ring. The successor of node N, $succ(N)$, is defied as the first node with the identifier larger than the identifier of node N. The predecessor of node N, $pred(N)$, is defined as the first node with the identifier smaller than the identifier of node N. Every node stores substrate addresses of its successor and predecessor, and a finger table with m entries, called fingers. In the finger table of node N, $finger[i]$ stores the substrate address of node with identifier $id(N) + 2^{i-1}$. If the node with identifier $id(N) + 2^{i-1}$ does not exist, then $finger[i]$ stores the substrate address of the first node with identifier larger than $id(N) + 2^{i-1}$ that exists, i.e., the successor of node with identifier $id(N) + 2^{i-1}$. A node also stores substrate addresses of r potential successors, referred to as a *successor list*, which can be used if a node loses connectivity to its current successor. To join a Chord ring, a node needs to be able to contact a node already in the Chord ring. To achieve this, each node is configured with a *buddy list*, which contains substrate addresses of a small number of nodes participating in the landmark domains routing.

An example of a Chord ring is given in Fig. 4.1, with node identifiers shown. Nodes have three bit



Data stored at node 0		
Entry	Identifier	Substrate address
finger[1]	1	*sa*(1)
finger[2]	3	*sa*(3)
finger[3]	4	*sa*(4)
successor list	1, 3	*sa*(1), *sa*(3)
predecessor	7	*sa*(7)

Data stored at node 3		
Entry	Identifier	Substrate address
finger[1]	4	*sa*(4)
finger[2]	7	*sa*(7)
finger[3]	7	*sa*(7)
successor list	4, 7	*sa*(4), *sa*(7)
predecessor	3	*sa*(3)

Figure 4.1: Chord ring with fingers for nodes 0 and 3 shown.

long identifiers ($m = 3$) and each node stores two potential successors ($r = 2$). The successor of node 0 is the next node in the ring, in clockwise direction, and is equal to $finger[1]$. $finger[2]$ should point to node with identifier 2, since $0 + 2^{2-1} = 2$. Because node with identifier 2 does not exist, $finger[2]$ of node 0 points to the successor of node 2, i.e., to node 3.

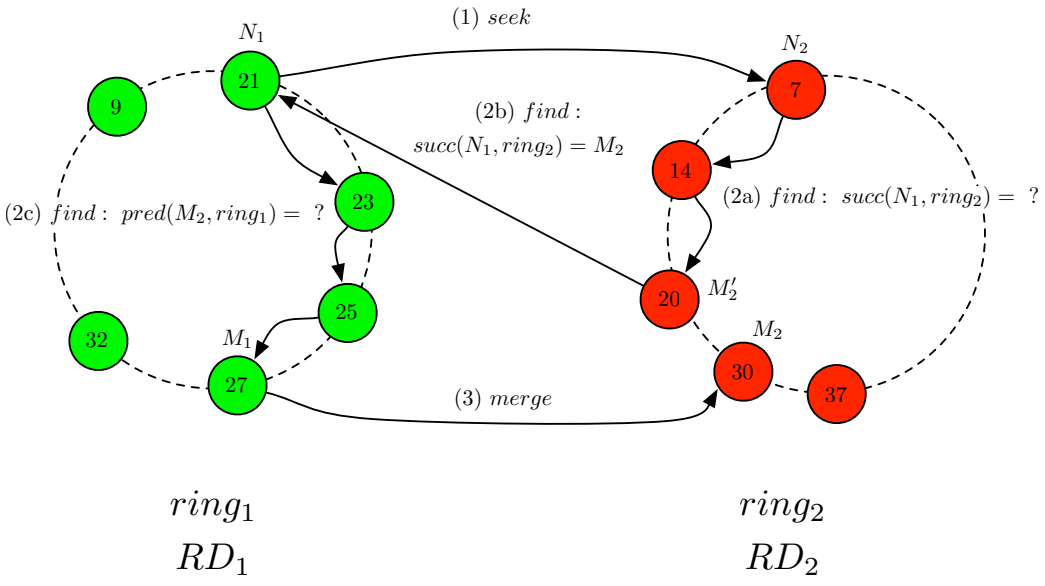
In a substrate network with more than one reachability domain, a separate Chord ring is created for each RD. When RDs are merged, so are their Chord rings. To merge nodes into a single Chord ring, we modify the *simple ring unification algorithm* presented by Shafaat et al. [53]. Nodes merge into a single Chord ring in three parts:

(1) A node detects that it can exchange messages with a node from a different Chord ring in the same substrate network.

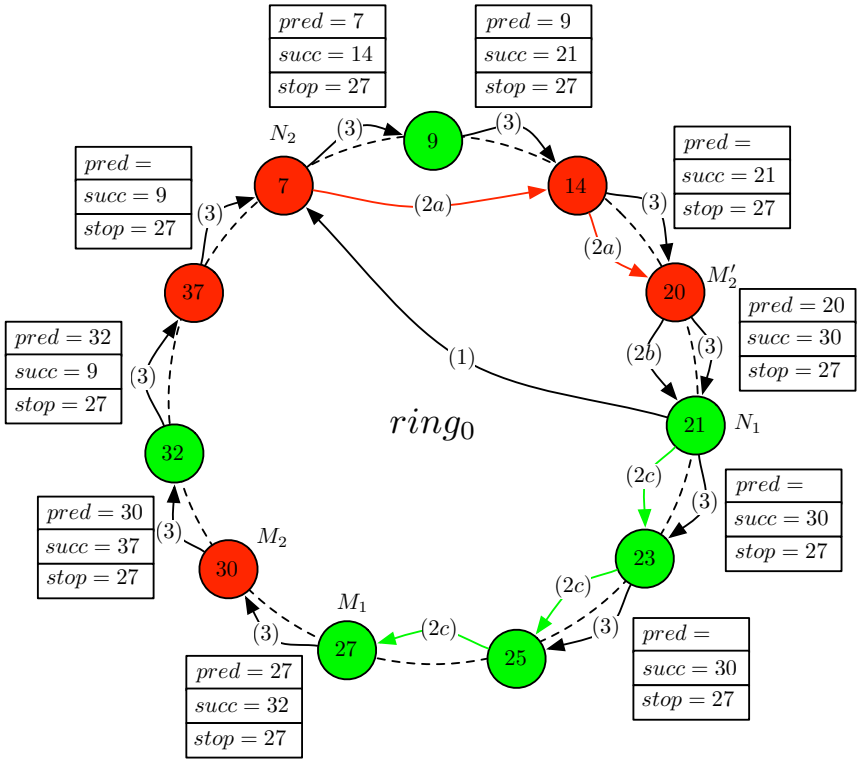
(2) Nodes from both rings find a position where rings can be merged. That is, some node M_1 from one ring needs to find a node M_2 from the other ring, such that $succ(M_1) = M_2$ in the merged ring.

(3) Nodes merge into a single Chord ring by updating their pointers to successor and predecessor nodes.

To test the need to merge RDs and Chord rings, every node periodically tries to send *seek* messages to nodes from its buddy list, using paths set up by the intra-substrate routing. If a substrate network supports broadcast, *seek* messages are instead broadcast. A node sends *seek* messages to try to reach nodes that are members of the same substrate network as the node, but are members of a different RD. A successful delivery of a *seek* message from a node in one RD to a node in another RD indicates that nodes from both RDs can communicate to each other, and thus should belong to the same RD. A *seek* message includes the RD-ID of the sender's node reachability domain. The receiver of a *seek* message



(a) Two Chord rings before the merging.



(b) Single Chord ring after the merging.

Figure 4.2: Merging of two Chord rings.

compares the RD-ID in the *seek* message to the RD-ID of its own RD. If RD-IDs are equal, both nodes are in the same RD and no further actions are taken. If RD-IDs are different, the receiving node is in a different RD than the sending node and the two RDs should be merged.

To explain how nodes from two Chord rings build a single Chord ring, we consider a *seek* message sent by a node N_1 to a node N_2. Node N_1 belongs to a Chord ring named $ring_1$ (and reachability domain RD_1) and node N_2 belongs to a Chord ring named $ring_2$ (and reachability domain RD_2), as illustrated in Fig. 4.2(a). The Chord ring, named $ring_0$, that nodes form $ring_1$ and $ring_2$ are building is shown in Fig. 4.2(b). Let $succ(N_i, ring_j)$ be the successor of node N_i in the Chord ring named $ring_j$ and let $pred(N_i, ring_j)$ be the predecessor of node N_i in the Chord ring named $ring_j$. Nodes from $ring_1$ and $ring_2$ build $ring_0$ in three parts:

(1) Node N_1 sends a *seek* message to a node N_2 from its buddy list, illustrated with an arrow from node N_1 to node N_2 in Fig. 4.2(a). The *seek* message contains the RD-ID of RD_1 ($id(RD_1)$), and the identifier and the substrate address of node N_1. Node N_2 is a member of RD_2, and since $id(RD_1) \neq id(RD_2)$ node N_2 concludes that it can communicate with a node from RD_1.

Algorithm 1 N : PROCESSFIND(msg)

```
1: if msg.M2 = null then // Searching for M2 = succ(N1, ring2)
2:     if id(msg.N1) ∈ (id(N), id(succ(N)))  then
3:         // Successor of this node is M2, send msg to ring1
4:         msg.M2 ← succ(N)
5:         sendTo(msg, msg.N1)
6:     else
7:         // Continue searching for M2
8:         nextNode ← getClosestPreceedingNode(msg.N1)
9:         sendTo(msg, nextNode)
10:    end if
11: else // Searching for M1 = pred(M2, ring1)
12:    if id(msg.M2) ∈ (id(N), id(succ(N)) then
13:        // This node is M1, start merging
14:        mm ← MergeMsg(N, succ(N), id(N)
15:        succ(N) ← msg.M2
16:        sendTo(mm, msg.M2)
17:    else
18:        // Continue searching for M1
19:        nextNode ← getClosestPreceedingNode(msg.M2)
20:        sendTo(msg, nextNode)
21:    end if
22: end if
```

(2) Nodes from $ring_1$ and $ring_2$ start building $ring_0$ by finding a node M_1 from $ring_1$ and a node M_2 from $ring_2$ such that:

$$succ(M_1, ring_0) = M_2.$$

Nodes M_1 and M_2 build $ring_0$ and will inform other nodes how to join $ring_0$ in part (3) of the merging. Nodes M_1 and M_2 are discovered using a *find* message, which has fields for the identifiers and substrate addresses of nodes N_1 and M_1. We present discovery of nodes M_1 and M_2 (see Algorithm 1) in three steps:

(2a) Node N_2, which has received the *seek* message from node N_1, starts searching for node M_2 in $ring_2$ such that:

$$M_2 = succ(N_1, ring_2).$$

Node N_2 creates a *find* message and records the identifier and the substrate address of node N_1 (from the *seek* message) in the *find* message. Node N_2 forwards the *find* message in $ring_2$ towards node N_1, illustrated by arrows inside $ring_2$ in Fig. 4.2(a). Since node N_1 is not a member of $ring_2$, forwarding of the *find* message terminates at a node M_2', which is the node in $ring_2$ with identifier closest to $id(N_1)$, but not larger than $id(N_1)$:

$$id\big(M_2'\big) < id\big(N_1\big) < id\big(succ\,(M_2', ring_2)\,\big).$$

It follows that $succ(N_1, ring_2) = succ(M_2', ring_2) = M_2$. Node M_2' stores the identifier and the substrate address of node M_2 in the *find* message.

(2b) Node M_2' sends the *find* message to node N_1, illustrated by an arrow from node M_2' to node N_1 in Fig. 4.2(a).

(2c) Node N_1 starts searching for node M_1 in $ring_1$ such that:

$$M_1 = pred(M_2, ring_1).$$

Node N_1 forwards the *find* message in $ring_1$ towards node M_2, illustrated by arrows inside $ring_1$ in Fig. 4.2(a). Forwarding of the *find* message terminates at a node M_1, which is the node in $ring_1$ with the identifier closest to $id(M_2)$, but not larger than $id(M_2)$:

$$id\big(M_1\big) < id\big(M_2\big) < id\big(succ\,(M_1, ring_1)\,\big).$$

That is, node M_1 is the predecessor of node M_2 in $ring_1$. Since $M_2 = succ(N_1, ring_2)$ there is no node in $ring_2$ with the identifier in the interval $\big(id(N_1), id(M_2)\big)$. Thus $pred(M_2, ring_0) = pred(M_2, ring_1) = M_1$.

Algorithm 2 N : processMerge(msg)

```
1: if msg.pred ≠ null then
2:     // Update predecessor node
3:     pred(N) ← msg.pred
4: end if
5: // If msg has not traversed the entire merged ring
6: if msg.stop ≠ id(succ(N)) then
7:     if id(msg.succ) ∈ (id(N), id(succ(N))) then
8:         // Update successor and inform new successor of old successor
9:         oldSuccessor ← succ(N)
10:        succ(N) ← msg.succ
11:        msg.succ ← oldSuccessor
12:        msg.pred ← N
13:    else
14:        // If this node did not change its successor node, the successor node
15:        // does not change its predecessor node
16:        msg.pred ← null
17:    end if
18:    sendTo(msg, succ(N));
19: end if
```

(3) If $M_1 = pred(M_2, ring_0)$ then $M_2 = succ(M_1, ring_0)$. Node M_1 builds $ring_0$ by setting node M_2 as

its successor. A *merge* message is used to inform other nodes to join $ring_0$. A *merge* message has three fields:

pred – the identifier and the substrate address of the new predecessor node,

succ – the identifier and the substrate address of a potential successor node, and

stop – the identifier of the node that created the *merge* message, i.e. $id(M_1)$.

Node M_1 creates a *merge* message and sets $pred = M_1$, $succ = succ(M_1, ring_1)$, and $stop = id(M_1)$. Node M_1 sends the *merge* message to node M_2, as illustrated by an arrow from node M_1 to node M_2 in Fig. 4.2(a). Passing of the *merge* message (see Algorithm 2) is illustrated in Fig. 4.2(b) with arrows outside $ring_0$ and the values of fields *pred*, *succ*, and *stop* shown.

Any node receiving the *merge* message checks if the *pred* field is set. If the *pred* field is set, the node updates its predecessor to the node stored in the *pred* field and clears the field. A node receiving the *merge* message also checks if *succ* is a better successor than nodes current successor.

If *succ* **is a better successor** than the current successor, then the node updates its successor to the node stored in the *succ* field. The node records its old successor in the *succ* field of the *merge* message. The node also records its own identifier and substrate address in the *pred* field, so that the new successor can update its predecessor to this node. The node sends the *merge* message to its new successor.

If *succ* **is not a better successor** than the current successor, then the node forwards the *merge* message to its current successor, without modifying the *pred* or *succ* fields.

When a node with the identifier equal to *stop* is reached, forwarding of the *merge* message stops and all nodes from $ring_1$ and $ring_2$ have joined $ring_0$.

4.2 Broadcasting in a Chord Ring

Algorithm 3 N: ForwardBroadcastMessage(msg)

```
1: // Initialize broadcast
2: if msg.topId = null then
3:     msg.topId ← id(N)
4:     sendTo(msg, finger[m])
5:     msg.topId ← id(finger[m])
6: end if
7: // Send to finger nodes in interval this node is responsible for
8: for i = (m − 1) down to 1 do
9:     // Skip fingers not in interval
10:    if id(finger[i]) ∈ (id(N), msg.topId) then
11:        msg.topId ← id(finger[i + 1]);
12:        sendTo(msg, finger[i]);
13:    end if
14: end for
```

A simple way to broadcast a message in a Chord ring is to forward a message along the Chord ring. If there are n nodes, the broadcast is completed after $(n - 1)$ rounds of transmission. To reduce

the number of rounds of transmission, we use the broadcast mechanism proposed by El-Ansary et al. [54], where the number of rounds of transmission to complete a broadcast is bounded by $O(\log_2(n))$. Some node N, which wants to broadcast a message, sends a copy of the message to each of its finger nodes: $finger[1], finger[2], \ldots, finger[m]$. Finger node $finger[i]$ is responsible to forward the broadcast message to nodes with identifiers in the interval $\Big(id(finger[i]), id(finger[i+1])\Big)$, for $1 \leq i < m$. Node $finger[m]$ is responsible for the interval $\Big(id(finger[m]), id(N)\Big)$. Finger nodes continue the broadcast by forwarding the message to a subset of their finger nodes, which fall in the interval they are responsible for. Message broadcasting is presented in Algorithm 3, where node N receives a message msg to broadcast. Message has a field $topId$, which stores the end of interval node N should broadcast to. Node N is thus assigned the interval $\big(id(N), topId\big)$. If node N initializes the broadcast (lines 1 – 6), N forwards a copy of msg to its $finger[m]$, setting $topId$ to its own identifier $id(N)$. Other nodes, which forward the broadcast message, never forward to their $finger[m]$ node, since $finger[m]$ node is outside the interval that they are responsible for. A node forwarding a broadcast message sends the message to a subset of its finger nodes $finger[1], finger[2], \ldots, finger[m-1]$, which are in the interval the node is responsible for (lines 7 – 14).

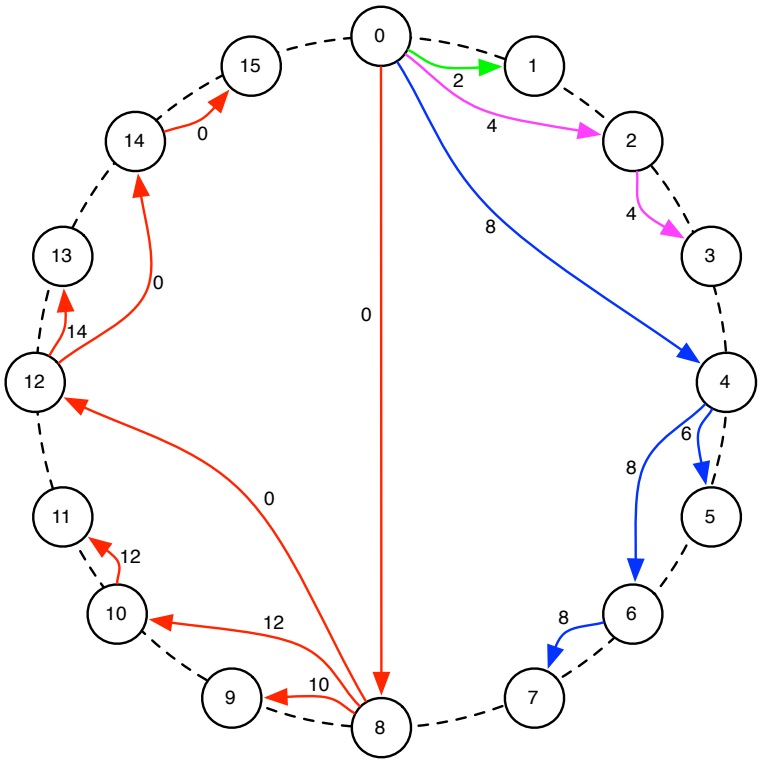


Figure 4.3: Broadcasting a message originating at node 0.

Fig. 4.4 shows an example where node 0 broadcasts a message in a Chord ring with $m = 4$. Arrows indicate forwarding of messages and the numbers next to the arrows indicate the value of $topId$ field in messages. To initiate the broadcast, node 0 sends the message to its finger nodes 1, 2, 4, and 8. By setting the $topId$ value, node 0 creates four intervals: $(1, 1), (2, 4), (4, 8)$, and $(8, 0)$. Forwarding of the message in each interval created by node 0 is shown by different colour arrows. Paths that the broadcast

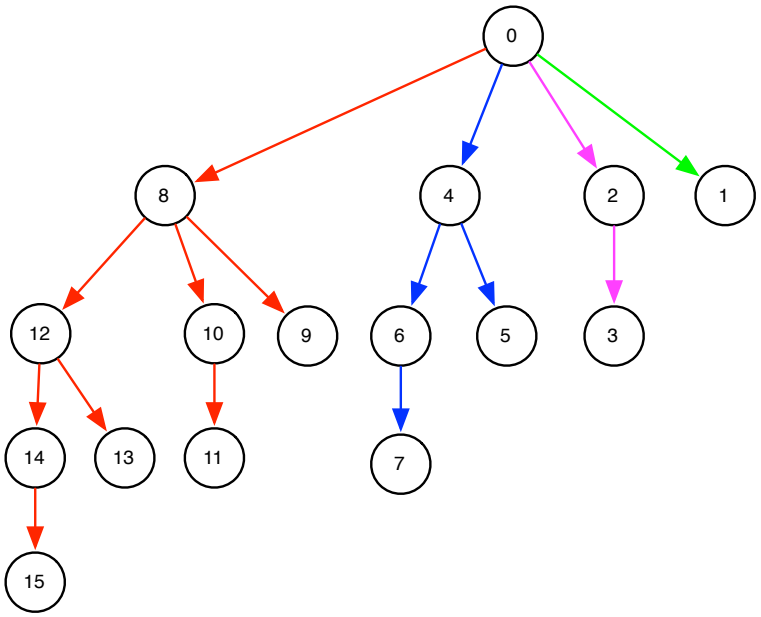


Figure 4.4: Broadcasting a message originating at node 0, shown as a tree.

message takes are redrawn as a tree in Fig. 4.4. Each of the nodes 1, 2, 4, and 8 is responsible to broadcast the message to the nodes in the sub-tree rooted at the node.

4.3 Determination of RD-IDs

Name	Description
rdId	identifier of the reachability domain
leader	*true* if the node is leader node, *false* otherwise
seqNo	last received sequence number from the leader node
lastTime	time when *seqNo* was received

Table 4.1: Information stored by a node for a reachability domain.

Recall from Section 3.1 that nodes form reachability domains by agreeing on an RD-ID for the reachability domain. The leader node broadcast an *RdInfo* message every $T_{checkRd}$ seconds to all members of the RD. The *RdInfo* message carries the RD-ID of the reachability domain and a sequence number. On each broadcast, the leader node increases the sequence number by one. Information that a member node stores for an RD is shown in Table 4.1.

When a node initially connects to a substrate network, the node tries to join an existing RD. A node waits for $T_{joinRdWait}$ time to receive an *RdInfo* message with an RD–ID. If a node does not receive an *RdInfo* message, the node creates a new RD by selecting a random RD-ID ,and becomes the leader node of the RD. The node broadcasts *RdInfo* messages, starting with sequence number zero.

Actions taken by any node N when receiving an *RdInfo* message are shown in Algorithm 4. Node N adopts the received RD–ID in three cases:

Algorithm 4 N : processRdInfo(msg)

```
1: if (rdId = null) || (msg.rdId > rdId)
2:  || (rdId = msg.rdId && seqNo < msg.seqNo) then
3:     rdId ← msg.rdId
4:     seqNo ← msg.seqNo
5:     lastTime ← currentTime
6:     // Continue broadcast of msg
7:     ForwardBroadcastMessage(msg)
8: end if
```

1. Node N has no RD–ID. (In this case, node N adopts the received RD–ID, and joins the existing reachability domain.)

2. The RD-ID of node N is numerically smaller than the one received.

3. The RD-ID of node N is the same as the one received and the received sequence number is larger than the sequence number of node N. (Node N checks sequence numbers to prevent forwarding old *rdInfo* messages that might be present in the network during changes of the RD membership.)

In all cases, node N stores the sequence number from the *rdInfo* message and updates the last time it heard from the leader node. Node N also continues the broadcast of the *rdInfo* message.

Algorithm 5 N : CheckRd

```
1: if leader then
2:     // Leader node updates sequence number and broadcasts
3:     seqNo++
4:     msg ← rdInfo(rdId, seqNo)
5:     msg.topId ← null
6:     ForwardBroadcastMessage(msg)
7: else if lastTime < currentTime − 2 ∗ T_checkRd then
8:     // Node N has not heard from leader in T_checkRd, concludes a split
9:     // has occurred, and picks a new RD-ID
10:    rdId ← random
11:    seqNo ← 0
12:    lastTime ← currentTime
13:    leader ← true
14:    // Broadcast new RD-ID
15:    msg ← rdInfo(rdId, seqNo)
16:    msg.topId ← null
17:    ForwardeBroadcastMessage(msg)
18: end if
```

Each node N checks for partitions of the reachability domain every $T_{checkRd}$ seconds (see Algorithm 5). If node N is the leader node, node N increases the sequence number and broadcasts an *rdInfo* message. Otherwise, if node N has not received an *rdInfo* message in the last $T_{checkRd}$ seconds, node N concludes that a path to the leader node is no longer available and that a partition of the reachability domain has occurred. In this case, node N picks a new RD-ID, sets it self as the leader of the new reachability domain, and broadcasts an *rdInfo* message with the new RD-ID and the sequence number set to zero.

4.4 Landmark Domains Routing

LM	RD	Dist	NextHop
$id(LD_1)$	$id(RD_1)$	$distance_{11}$	$sa_1(N_{11})$
	$id(RD_2)$	$distance_{12}$	$sa_2(N_{12})$
	...	...	...
	$id(RD_a)$	$distance_{1a}$	$sa_a(N_{1a})$
$id(LD_2)$	$id(RD_1)$	$distance_{21}$	$sa_1(N_{21})$
	$id(RD_2)$	$distance_{22}$	$sa_2(N_{22})$
	...	...	...
	$id(RD_a)$	$distance_{2a}$	$sa_a(N_{2a})$
...	...	...	...
$id(LD_b)$	$id(RD_1)$	$distance_{b1}$	$sa_1(N_{b1})$
	$id(RD_2)$	$distance_{b2}$	$sa_2(N_{b2})$
	...	...	...
	$id(RD_a)$	$distance_{ba}$	$sa_a(N_{ba})$

Table 4.2: Landmark domains routing table.

This section describes the discovery of paths to landmark domains. We use distance vector routing, which operates at the granularity of reachability domains. That is, we consider connectivity between RDs and measure path lengths in the number of traversed RDs. Member nodes of an RD store information about next-hop RDs on paths to every landmark domain. A node that is a member of more than one RD, stores information about a path to every landmark domain from each RD. Nodes store information about paths to landmark domains in the landmark domains routing table. Table 4.2 shows the structure of the landmark domains routing table for a node connected to a RDs in a network with b landmark domains. For each landmark domain, the node stores information about paths obtained from all reachability domains. A single entry in the landmark domains routing table has the following information:

LD – RD-ID of a landmark domain,

RD – RD-ID of the reachability domain from which the routing information is obtained,

Dist – distance to the landmark domain measured in the number of traversed RDs,

NextHop – substrate address of a next-hop node on a path to the landmark domain. (The next-hop node is a member of the next-hop RD.)

To send a message to a particular landmark domain, a node selects the entry with the shortest distance. The node uses the substrate address of the next-hop node to forward a message to the next-hop RD, using intra-substrate routing.

The discovery of paths to a landmark domain, say RD_L, starts with some node N, which is multi-homed in RD_L and some other reachability domain RD_O. Since node N is a member of RD_L, node N has a path of zero hops to the landmark domain RD_L. Node N informs other members of RD_O about a path to RD_L, one hop long, with node N as the next-hop node. Nodes multi-homed in RD_O now know a one hop long path to RD_L, via node N, and can advertise two hop long path to RD_L via themselves.

Nodes in any reachability domain RD_O periodically exchange routing information. Ideally, every member node would inform all other member nodes about shortest paths to all landmark domains the node has stored. Then, member nodes can locally compute shortest paths from RD_O to all landmark

domains. However, all nodes broadcasting routing information might overwhelm the substrate network. Instead, a single member node collects routing information from a subset of member nodes, compute shortest paths, and broadcast the shortest paths to all member nodes.

All member nodes of RD_O check every $T_{checkRt}$ seconds if they have received a routing table update in the last $T_{checkRt}$ seconds. Assume node N in RD_O has detected it has not received a routing table update in last $T_{checkRt}$ seconds. Node N sends a *pull* message to all next-hop nodes from its landmark domains routing table where RD is equal to $id(RD_O)$. Recipients of the *pull* message reply to node N with a *pullReply* message containing landmark domains routing table entries with shortest distances to all landmark domains. Node N computes shortest paths to all landmark domains for RD_O and puts them in a *push* message. The *push* message is broadcast to all member nodes of RD_O, which update their landmark domains routing table. The broadcast is performed in the same way as described in Section 4.2.

Nodes pull the routing information only from next-hop nodes, to limit the number of messages exchanged. This allows nodes in an RD to detect if an existing path to a landmark domain has failed. However, if some node N in RD_O learns of a better path to a landmark domain, from some reachability domain other than RD_O, N starts the update process by sending a *pull* message. Since N computes shortest paths and sends out a *push* message, N can inform all other member nodes about the new path.

Chapter 5

Evaluation

In this chapter, we present the evaluation of landmark domains routing. We evaluate three properties of the proposed routing scheme:

Stored state – the amount of information stored at each node by the routing protocol, i.e., the size of routing tables,

Path stretch – the ratio between the length of a path set up by the routing protocol and the shortest path from source to destination, and

Message overhead – the communication cost of setting up routing paths.

When designing a routing protocol, it is desirable to be able to set up routing paths with a small message overhead. Paths set up by the routing protocol should have a small path stretch, so that messages are forwarded on paths with lengths close to the shortest paths. The stored state should also be small, so that it can be stored at nodes with limited memory. Stored state, message overhead, and path stretch present a trade-off: stored state and message overhead can be reduced at the cost of increased path stretch. To be able to operate in networks of any size, a routing protocol should use stored state that does not grow fast with the size of the network. For instance, shortest path routing protocols produce the smallest path stretch of one. In these protocols the stored state at each node is proportional to the number of nodes in the network. Using shortest path routing protocols in large networks may result in amount of stored state that exceeds available memory at nodes. Routing protocols suitable for networks of any size are called *scalable* routing protocols. In scalable routing protocols, the trade-off between stored state, message overhead, and path stretch has an additional constraint that stored state at a node grows slower than the number of nodes in the network.

We will evaluate the scalability and performance of landmark domains routing. First, we evaluate the scalability of landmark domains routing through numerical analysis. We explore how the stored state and path stretch grows as the size of a multi-substrate network increases. In the second part of this chapter we evaluate the implementation of the proposed routing scheme using simulations. We compare landmark domains routing to existing compact routing scheme (Disco [12]), and overlay network based greedy routing schemes (UIP [13] and VRR [14]).

5.1 Numerical Analysis

In this section we present a numerical analysis of the scaling properties of landmark domains routing. First, we explore the size of stored state at a node as a function of the number of nodes and substrate networks in a multi-substrate network. In the second part of this section we present an analysis of path stretch for large multi-substrate networks.

5.1.1 Stored State

n	Number of nodes
n_S	Number of substrate networks
n_L	Number of landmark domains
n_I	Number of interfaces per node
n_m	Number of entries in finger tables (in Chord rings)
r	Number of entries in successor lists (in Chord rings)

Table 5.1: Parameters for computation of stored state.

In landmark domains routing, the stored state at a node consists of:

- the landmark domains routing table (see Table 4.2),
- information about RDs (see Table 4.1), and
- state stored for Chord rings (finger tables and successor lists).

Table 5.1 shows parameters that determine the amount of stored state at a node in landmark domains routing. A node connects with each of its n_I interfaces to a substrate network, and is a member of a reachability domain in every substrate network. A node stores:

- $n_I n_L$ entries in the landmark domains routing table,
- n_I entries with information about RDs, and
- n_I finger tables, with n_m entries each, and n_I successor lists, with r entries each.

Thus, the stored state at a node in landmark domains routing, denoted by n_e^{LD} and measured in the number of entries, is given by:

$$n_e^{LD} = n_I n_L + n_I + n_I(n_m + r). \tag{5.1}$$

We compare the stored state of landmark domains routing to the stored state of a compact routing scheme Disco [12], as baseline. In Disco, the number of landmark nodes in the multi-substrate network is computed as $\sqrt{n \log(n)}$. Each node stores one routing table entry for each landmark node and one routing table entry for the $\sqrt{n \log(n)}$ closest neighbours. The stored state at a Disco node (n_e^D), measured in the number of entries, is given by:

$$n_e^D = 2\sqrt{n \log(n)} \tag{5.2}$$

For the number of landmark domains we pick $n_L = \sqrt{n_s \log(n_s)}$. Then Equation 5.1 becomes

$$n_e^{LD} = n_I \left(\sqrt{n_s \log(n_s)}\right) + n_I + n_I(n_m + r). \tag{5.3}$$

By selecting $\sqrt{n_s \log(n_s)}$ reachability domains as landmark domains, we ensure that stored state at nodes grows slowly with the increase in the number of reachability domains. Also, this allows us to make a direct comparison between stored state of landmark domains routing and Disco. In a multi-substrate network with one node per substrate network, the number of nodes and substrate networks is equal. With $n_S = n$, the number of landmark domains and landmark nodes is also equal.

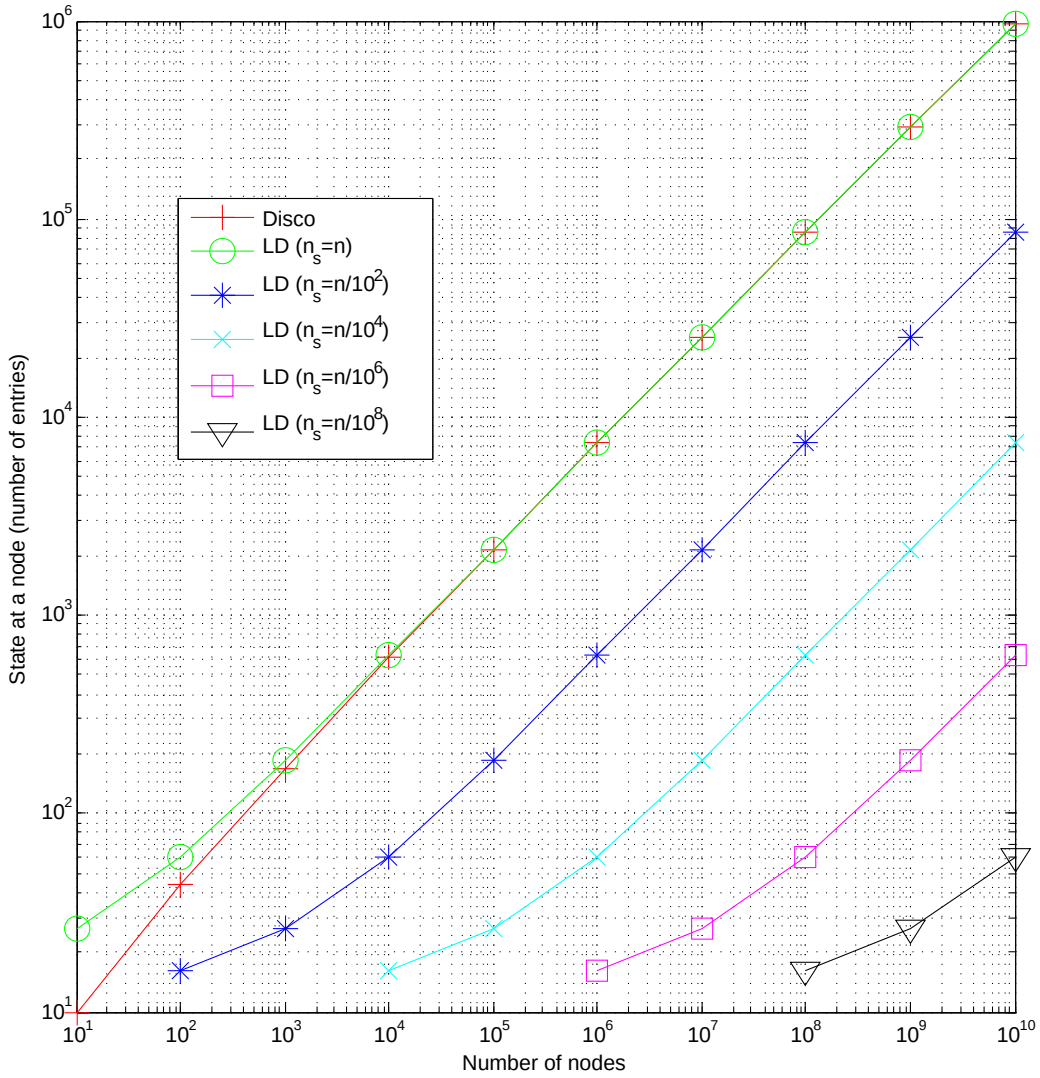


Figure 5.1: Stored state at a node.

To compute stored state we assume that in a multi-substrate network with n nodes and n_S substrate networks, each substrate network has the same number of nodes, given by $\frac{n}{n_S}$. Fig. 5.1 shows the stored state for Disco and landmark domains routing as a function of the number of nodes, parametrized by the total number of substrate networks. The stored state of landmark domains routing grows slower

than linear as the number of nodes increases, showing that landmark domains routing is scalable. When we set $n_S = n$, each substrate network contains only a single node, and the stored state for landmark domains routing and Disco are roughly the same. When we increase the size of substrate networks, the stored state in landmark domains routing decreases significantly, while the stored state in Disco dose not change. This shows that landmark domains routing leverages existing substrate networks to reduce the amount of stored state.

5.1.2 Path Stretch

Recall from Section 3.3 that a message forwarded by landmark domains routing is first forwarded to a landmark domain, and then from the landmark domain to the destination node. The landmark domain is selected by the destination node and included in the locator of the destination node. A node selects the closest landmark domain when creating a locator. In the worst case, the path stretch produced by landmark domains routing is twice the diameter of the network. This path stretch is determined by several factors:

- source and destination nodes are one-hop connected, i.e., the shortest path between source and destination is one hop long, and
- the path from the source node to the landmark domain has the maximal length, i.e., given by the diameter of the network, and
- the path from the landmark domain to the destination node has maximal length, i.e., diameter of the network.

With a sufficiently large number of landmark domains, it is unlikely that nodes will experience the worst case path stretch. As the number of landmark domains increases, the probability that a landmark domain exists close to a node also increases. Since the path stretch depends on the distance from node to its closes landmark domain, the probability of nodes experiencing worst case path stretch decreases. To show this we proceed to compute the path stretch depending on the distance between the destination RD and the closest landmark domain. Consider Fig. 5.2, which depicts a source reachability domain RD_S and a destination reachability domain RD_D. The shortest path from RD_S to RD_D is independent of the location of landmark domains. The path taken by landmark domains routing deviates from the shortest path to visit a landmark domain. The deviation from the shortest path depends on the distance from the landmark domain to the destination RD. For landmark domain LM_1, which is closer to RD_D than LM_2, the deviation is smaller. As a result, the path stretch from RD_S to RD_D is smaller if RD_D uses LM_1 as landmark domain instead of LM_2.

We explore the path stretch for a multi-substrate network modelled as a grid of substrate networks. Each substrate network connects to four other substrate networks, i.e., to substrate networks directly above, below, left, and right. We assume that all nodes that are members of a substrate network are members of a same RD and that there is one RD per substrate network. Further, we assume that all nodes in an RD are one-hop connected, i.e., a path through any RD is one hop long. Thus, the number of hops in a path is equal to the number of RDs the path traverses. We pick one RD as the destination RD and compute the path stretch for messages sent from other RDs to the destination RD. For illustration, Fig. 5.3 shows a scenario with 81 substrate networks. The destination reachability domain is labelled RD_D. The landmark domain closest to RD_D is labelled LM. In the figure, the shortest path from LM

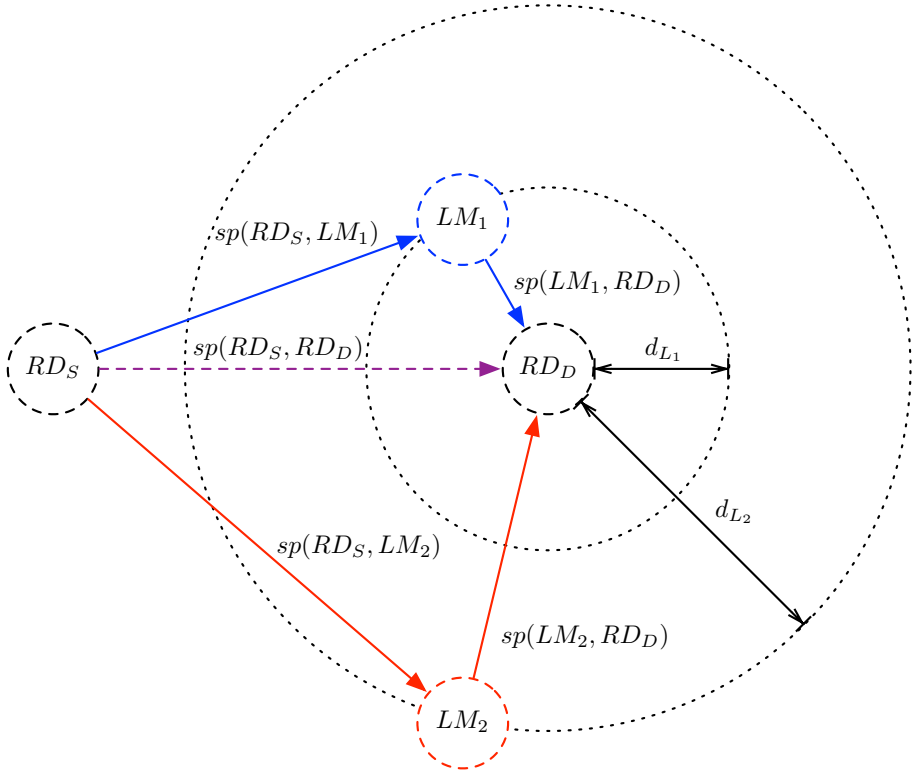


Figure 5.2: Distance of a landmark domain to the destination node.

to RD_D is four hops long. Fig. 5.3 also shows a source reachability domain labelled RD_S. The shortest path from RD_S to RD_D is showed with the dashed arrow and is eight hops long. The path taken by landmark domains routing is shown with solid arrows and is 10 hops long. This results in a path stretch of 1.25.

We assume that each LD is selected uniformly at random from the set of reachability domains, independently from other landmark domains. Let d_L be the number of hops from the destination reachability domain to the closest landmark domain. We are interested in the probability of a landmark domain being close to the destination reachability domain RD_D. Thus, we compute the probability $\Pr[d_L \leq d]$ that the closest landmark domain to RD_D is reachable via a path of d or less hops. We compute $\Pr[d_L \leq d]$ as:

$$\Pr[d_L \leq d] \quad = \quad 1 - \Pr[d_L > d], \tag{5.4}$$

where $\Pr[d_L > d]$ is the probability that the closest landmark domain to RD_D is more than d hops away. We compute this as the probability that all landmark domains are more than d hops away from RD_D. For a single landmark domain, the probability that it is more than d hops away from RD_D is given by:

$$1 - \frac{4\sum_{i=1}^{d} i + 1}{n_s},$$

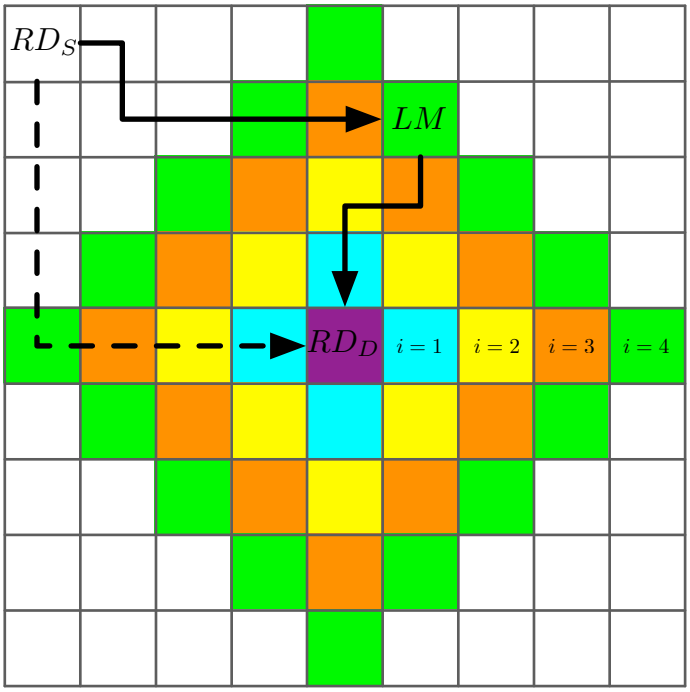


Figure 5.3: Grid model of a multi-substrate network.

where $4\sum_{i=1}^{d} i + 1$ is the number of RDs that are d or less hops away from RD_D and n_S is the total number of RDs. The number of RDs that are $i = 1, i = 2, \ldots, i = d$ hops away form RD_D is counted by $4\sum_{i=1}^{d} i$. We add one to the sum for RD_D it self. Then, for n_L landmark domains, $\Pr[d_L > d]$ is given by:

$$\Pr[d_L > d] = \left(1 - \frac{4\sum_{i=1}^{d} i + 1}{n_s}\right)^{n_L}, \tag{5.5}$$

Using (5.5) in (5.4) we get:

$$\begin{aligned} \Pr[d_L \leq d] &= 1 - \left(1 - \frac{4\sum_{i=1}^{d} i + 1}{n_s}\right)^{n_L} \\ &= 1 - \left(1 - \frac{2d(d+1) + 1}{n_s}\right)^{n_L} \end{aligned} \tag{5.6}$$

For illustration, in Fig. 5.3 we indicate RDs that are one, two, three, and four hops away from RD_D with cyan, yellow, orange, and green colour, respectively. Since landmark domains are selected independently, to get the probability that all n_L landmark domains are more than d hops away we multiply the probability for a single landmark domain n_L times.

Fig. 5.4 shows the probability that at least one landmark domain is d hops away from RD_D, for a multi-substrate network with 10^6 substrate networks. As we increase the number of landmark domains, the probability that a landmark domain exists close to RD_D increases.

We plot the path stretch between all RDs and the destination RD in the centre of the network,

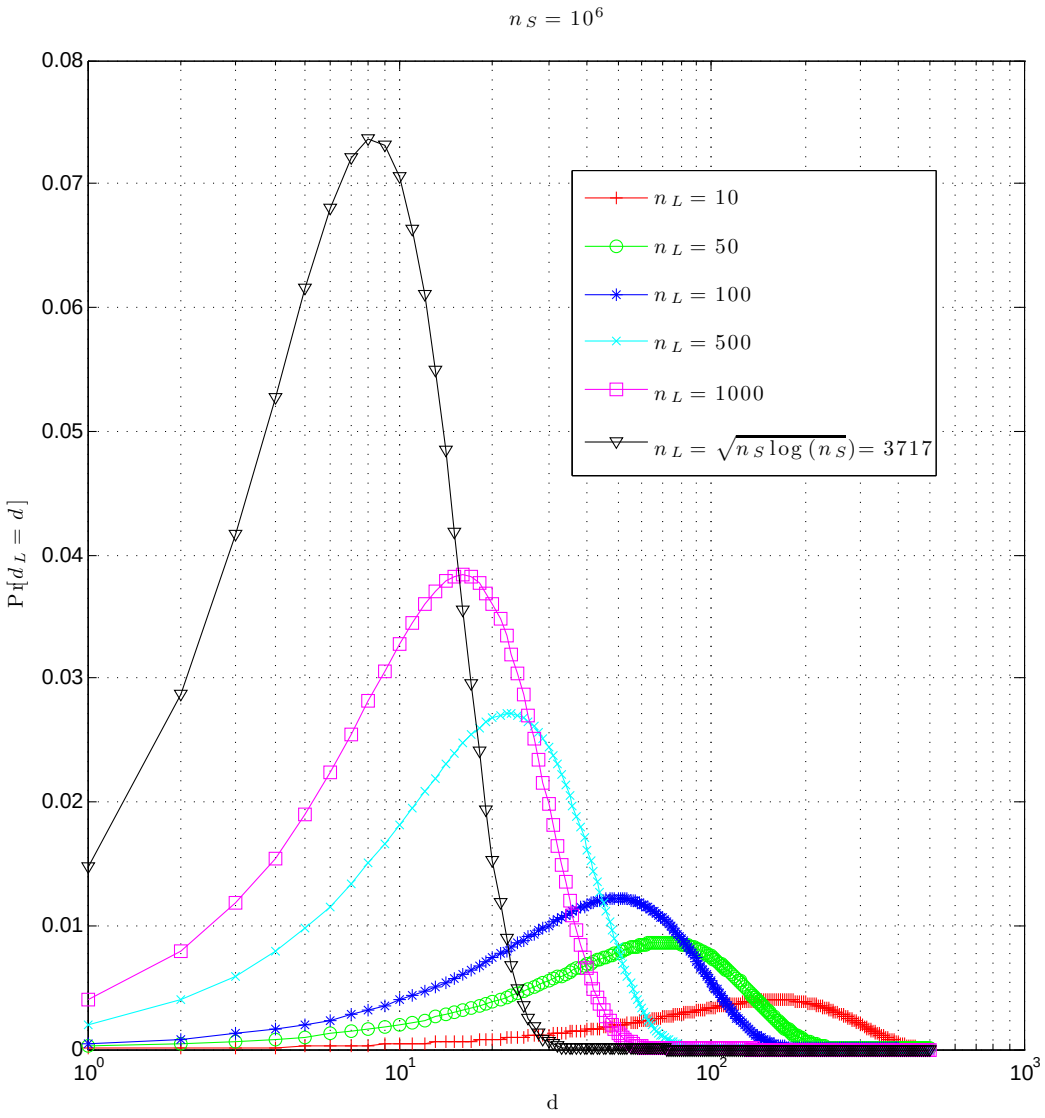


Figure 5.4: Probability of a landmark domain existing d hops from RD_D ($\Pr[d_L = d]$) for a multi-substrate network with 10^6 substrate networks.

depending on the distance of the closest landmark domain to the destination RD, in Fig. 5.5. As the distance from the destination RD to the closest landmark domain approaches one, the path stretch also approaches one. We also see from Fig. 5.5 that the majority of paths have a small path stretch, even when the distance between RD_D and the closest landmark domain is large.

In Fig. 5.6 we plot the path stretch for multi-substrate networks by varying the number of substrate networks. The number of landmark domains is fixed to 1000. For each number of substrate networks we compute the probability of a landmark domain falling d hops away from RD_D and use the expected value to compute the path stretch. We plot path stretch only from 1 to 1.5 since almost all paths have a path stretch in this range. Fig. 5.6 shows that there is little difference in path stretch when the size

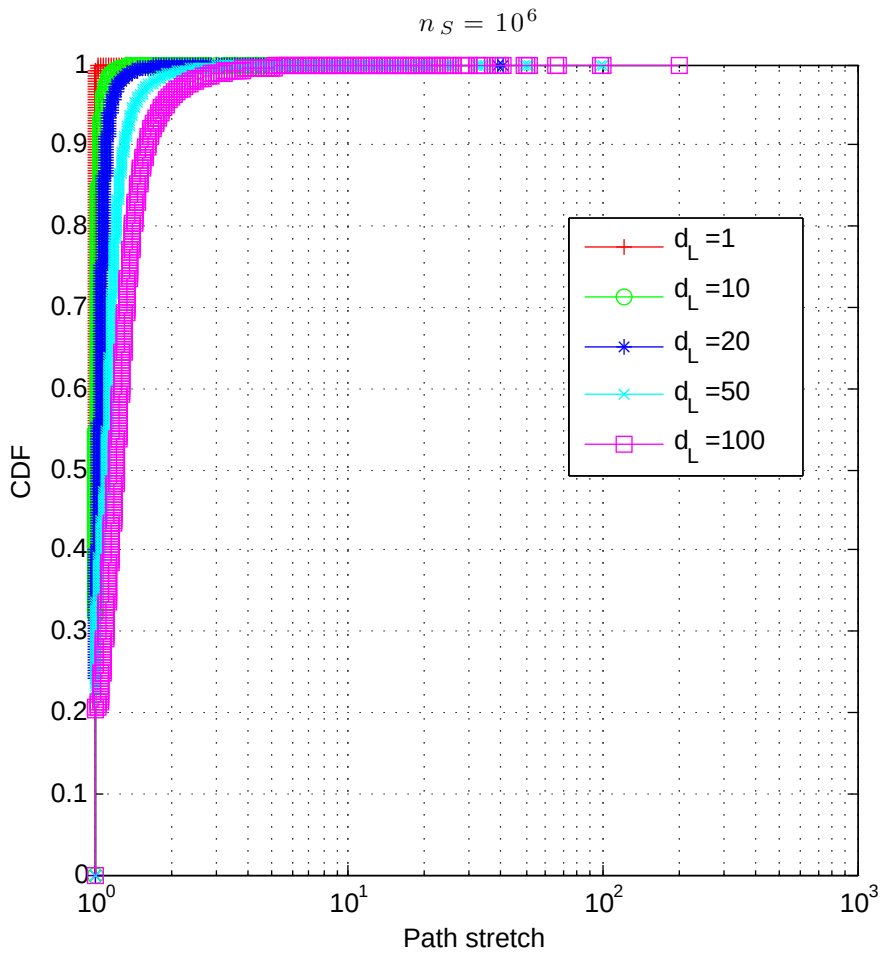


Figure 5.5: Path stretch for a multi-substrate network with 10^6 substrate networks.

of a multi-substrate network is increased, although the expected distance from the landmark domain to RD_D increases significantly. To explain this, lets consider source reachability domain RD_S, destination reachability domain RD_D, and a landmark domain LM. The length of the shortest path from RD_S to RD_D is denoted by $|sp(RD_S, RD_D)|$. The path stretch $ps(RD_S, RD_D)$ can be computed as:

$$\begin{aligned} ps(RD_S, RD_D) &= \frac{|sp(RD_S, LM)| + |sp(LM, RD_D)|}{|sp(RD_S, RD_D)|} \\ &= \frac{|sp(RD_S, RD_D)| + \big(|sp(RD_S, LM)| + |sp(LM, RD_D)| - |sp(RD_S, RD_D)|\big)}{|sp(RD_S, RD_D)|} \\ &= \frac{|sp(RD_S, RD_D)| + dev}{|sp(RD_S, RD_D)|} \\ &= 1 + \frac{dev}{|sp(RD_S, RD_D)|}, \end{aligned} \tag{5.7}$$

where dev expresses the additional path length caused by visiting landmark domain LM, compared to

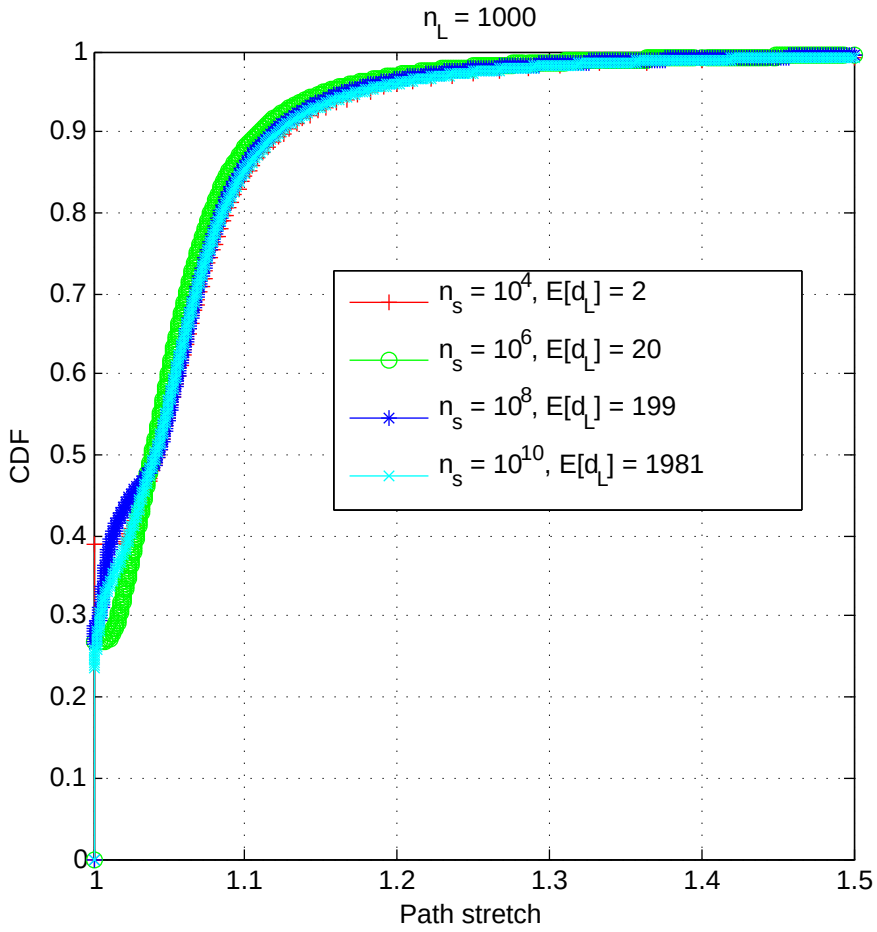


Figure 5.6: Path stretch for a multi-substrate network with 1000 landmark domains.

the shortest path. As the size of a multi-substrate network is increased, the average length of shortest paths increases proportionally. However, even with a fixed number of landmark domains, the expected distance of the closest landmark domain to the destination RD and thus the value of *dev* increase much slower than the length of the average shortest path. Therefore, as the size of multi-substrate network grows, the path stretch is dominated by the length of shortest paths, and therefore approaches one.

5.2 Simulations

In this section we present results obtained from simulations of landmark domains routing. For simulations we use the discrete event simulator OMNeT++ [55, 56]. To evaluate paths set up by simulated routing scheme, we evaluate the transmission of messages between 250^2 randomly selected node pairs. We compute the path stretch between the selected pairs of nodes. Delivery ratio for a node is computed as the number of messages received by the node divided by the number of messages sent to the node. We measure message overhead at a node as the bitrate of received control messages, averaged over the duration of 1 second.

The rest of this section presents how multi-substrate networks are generated and explain configuration parameters for the simulated routing schemes. We present simulation results for static and dynamic multi-substrate networks. In static multi-substrate networks the connectivity between nodes remains the same during the simulation. Dynamic multi-substrate networks contain mobile nodes, whose movement causes changes of connectivity between nodes during the simulation.

5.2.1 Multi-substrate Network Generation

n_S	Number of substrate networks
$\frac{n}{n_S}$	Number of nodes in each substrate network
α^p, α^e	Parameter of substrate degree probability distribution (α^p for power-law and α^e for exponential distribution)
β^p, β^e	Parameter of node degree probability distribution (β^p for power-law and β^e for exponential distribution)

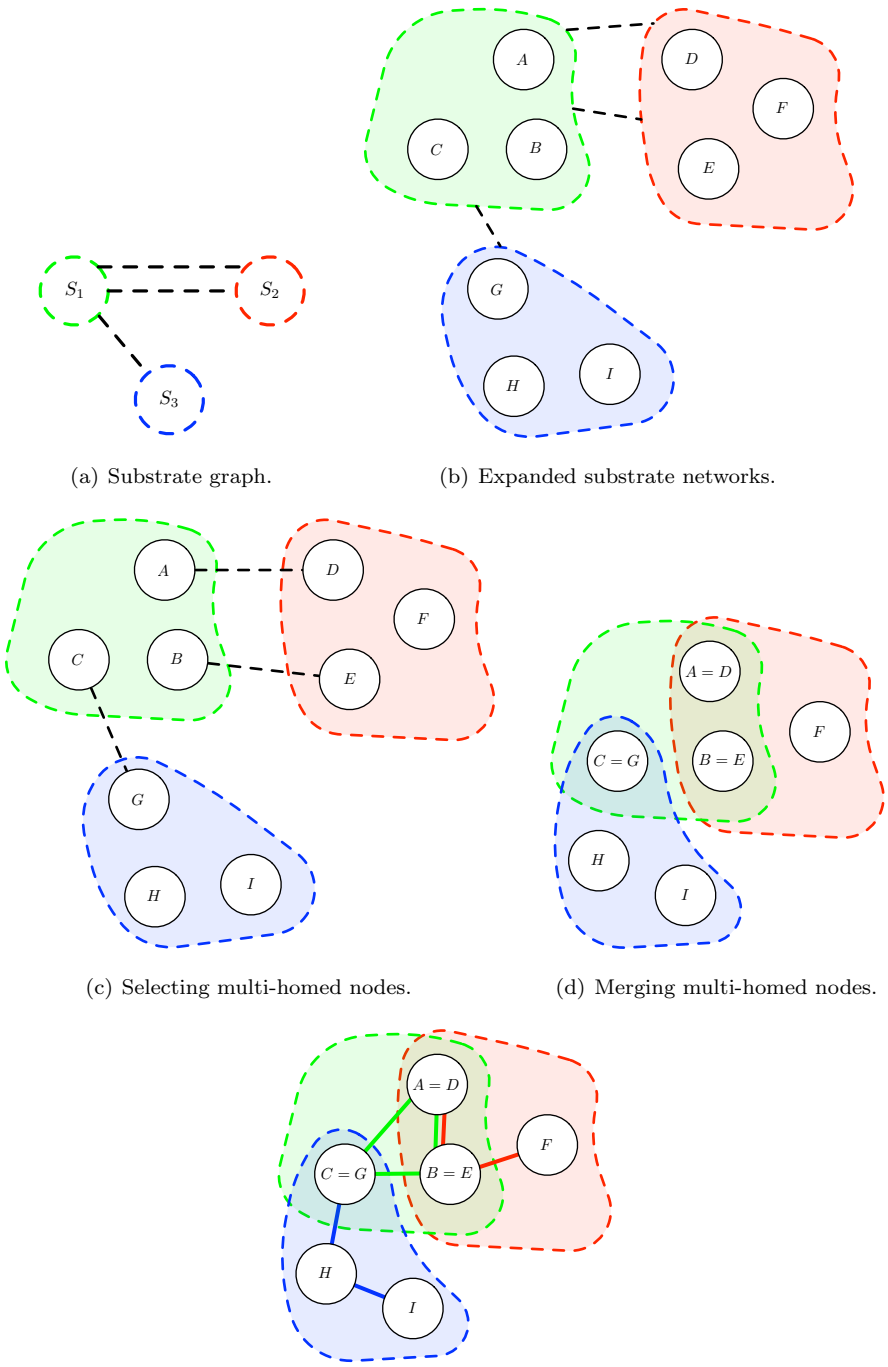
Table 5.2: Parameters for static multi-substrate networks.

To evaluate landmark domains routing we need a topology generator for multi-substrate networks. The generator needs to create connections between substrate networks and links between nodes in substrate networks. We define *substrate network degree* of a substrate network, say S_1, as the number of substrate networks connected to S_1. The node degree of a node N, in a substrate network S_1, is the number of neighbours node N has in S_1. We have developed a custom generator for multi-substrate networks, which produces multi-substrate networks with substrate and node degree governed by a probability distribution. Input parameters for our multi-substrate network generator are shown in Table 5.2.

The topology generator starts by drawing n_S random positive integers from a power-law probability distribution with parameter α^p or an exponential probability distribution with parameter α^e. The numbers are re-drawn until they are a graphic sequence [1], to ensure they can be used as substrate network degrees. The drawn numbers are used as substrate network degrees to create a random substrate network graph with n_S substrate networks, by using the modified *pseudo-graph* (also known as *configuration graph*) approach (see [58]). To each substrate network we attach a number of stubs equal to the substrate network degree. We randomly select pairs of stubs and connect them to create edges, avoiding creating self-loops. If the generated substrate network graph is not connected, the process is repeated (including drawing new random numbers). Next, we assigned $\frac{n}{n_S}$ nodes to each substrate network. We replace edges between substrate networks with multi-homed nodes, by randomly uniformly picking one node from each substrate network connected by an edge and merging the two nodes into a single node. Finally, for each substrate network we create links between nodes, in the same way as edges between substrate networks were created, using parameter β^p or β^e for node degree distribution.

The process of generating a multi-substrate network is illustrated in Fig. 5.7, where $n_S = 3$ and $\frac{n}{n_S} = 3$. First, a random graph with three substrate networks and edges between them is generated (Fig. 5.7(a)). Substrate networks are expanded and three nodes are assigned to each substrate network, as shown in Fig. 5.7(b). For every edge between two substrate networks, one node from each substrate network is selected (Fig. 5.7(c)). For example, for the edge between substrate networks S_1 and S_3 we

[1] The list of degrees of all nodes of a graph, arranged monotonically and beginning with the maximum degree, is called a degree sequence. A finite sequence of non-negative integers is a graphic sequence if it is a permutation of the degree sequence of some graph (see [57]).



(a) Substrate graph.

(b) Expanded substrate networks.

(c) Selecting multi-homed nodes.

(d) Merging multi-homed nodes.

(e) Generating links between nodes.

Figure 5.7: Steps of generating a multi-substrate network.

pick nodes C and G. Next, nodes connected by edges between substrate networks are merged into multi-homed nodes (Fig. 5.7(d)). For example, nodes C and G become a single node, labelled $C = G$, multi-homed in substrate networks S_1 and S_3. Finally, for each substrate network links between nodes are generated (Fig. 5.7(e)).

n_{WS}	Number of wireless substrate networks
n^s_{WI}	Number of wireless interfaces per static node
n^m	Number of mobile nodes
n^m_{WI}	Number of wireless interfaces per mobile node
r_{WI}	Radio range of wireless interfaces
n_G	Number of groups
$\frac{n^m}{n_G}$	Number of mobile nodes in a group
r_G	Group radius
v_{min}	Minimum node (group) speed
v_{max}	Maximum node (group) speed
p	Maximum node (group) pause time
f_x	Length of playing field
f_y	Hight of playing field

Table 5.3: Additional parameters for dynamic multi-substrate networks.

We also generate dynamic multi-substrate networks, which are composed of wired and wireless substrate networks. First, we generate a multi-substrate network with wired substrate networks as described for a static multi-substrate network. We replace some of the nodes in the generated multi-substrate network with routers. Routers are devices, which participate in the intra-substrate routing, but not in the simulated routing. A router connects only to one substrate network. To generate wireless substrate networks and mobile nodes we use parameters shown in Table 5.3. We crate n_{WS} wireless substrate networks and equip each static node with n^s_{WI} wireless interfaces. A static node connects with each of its wireless interfaces to a randomly selected wireless substrate network. Further, we create n^m mobile nodes, each of them equipped with n^m_{WI} wireless interfaces. All wireless interfaces have radio range of r_{WI} metres. To simulate dynamic substrates, we organize mobile nodes into n_G groups, with $\frac{n^m}{n_G}$ nodes in each group. All mobile nodes in the same group connect to the same wireless substrate network with one of their wireless interfaces. With the remaining $n^m_{WI} - 1$ wireless interfaces a mobile node connects to randomly selected substrate networks.

Node mobility is modelled using Reference Point Group Mobility (RPGM) model [59] implemented in BonnMotion [60]. In this model, nodes move together in groups. Each group has radius of r_G metres and nodes in a group move independently within the group radius. Nodes move on a playing field of size $f_x \times f_y$ m^2. Nodes movement on the playing field is superposition of group movement and node's movement in a group. Movement of nodes and groups is modelled using waypoint mobility: nodes (and groups) move from one randomly selected waypoint to another, randomly selecting a new speed (between v_{min} and v_{max}) at each waypoint. Also, nodes (and groups) can remain still (for a random time between 0 and p seconds) when they reach a waypoint.

5.2.2 Simulated Routing Schemes

To evaluate the performance of landmark domains routing we compare it with three existing routing schemes: Disco [12], UIP [13], and VRR [14]. Table 5.4 shows the configuration parameters used. In all routing schemes we use m-bit long binary identifiers.

We use Disco to make a comparison to a compact routing scheme and UIP and VRR to make a comparison to greedy routing schemes. Two-level routing schemes described in Section 2.3 are not suitable for a direct comparison with landmark domains routing. LISP [42], ILNP [43], RANGI [44],

Name	Description	Value
m	Identifier length	14
T_{Rt}^{Disco}	Time between sending routing information in Disco	1 s
n_b^{UIP}	Number of nodes in a bucket in UIP	3
T_{Rt}^{UIP}	Time between sending routing information in UIP	1 s
r^{VRR}	Size of *vset* in VRR	2
$T_{checkRd}$	Time between broadcasting RD-IDs	2 s
$T_{checkRt}$	Time between sending routing information	2 s
T_{waitRt}	Time to wait for replies to a *pull* message	1 s
r	Number of alternate successors in Chord ring	2
$T_{stabilize}$	Time between checking reachability of successor node	1 s
T_{fix}	Time between checking reachability of finger nodes	2 s

Table 5.4: Configuration parameters for simulated routing schemes.

IPNL [45], 4+4 [46], and GSE [47] assume existence of a central substrate network $\hat{S}$, and routing schemes of Ahlgren et al. [48] and Feldman et al. [49] assume existence of a central hierarchy of substrate networks $\hat{\mathbb{S}}$. This routing schemes are not intended to operate on a multi-substrate network without a central substrate network or networks. FARA [51] and Plutarch [52] are designed to operate without the assumption of a central substrate network, however, the authors describe these two routing schemes at an abstract level and do not provide sufficient details to allow implementation. Additionally, none of the mentioned two-level routing schemes supports dynamic substrates, that is, the schemes assume that substrate networks are never partitioned. TurfNet [50] is designed to support dynamic substrate networks and does not assume a central substrate network, but the authors do not provide any details about the routing between substrate networks, thus making this scheme impossible to implement.

Disco is a compact routing scheme, which provides a bound of $\tilde{O}(n^{1/2})$ per node on stored state and guarantees paths stretch below 3. In Disco, every node stores routing entries for landmark nodes and nodes in its vicinity. Each node sends all stored entries to its neighbours every T_{Rt}^{Disco} seconds.

UIP is a routing scheme, for multi-substrate networks, which leverages the existing intra-substrate routing. In UIP, each node stores information about the virtual links to overlay neighbours. A node has m buckets (one for the each possible length of the longest common prefix between two nodes). Bucket i stores information about virtual links to N_{bucket}^{UIP} neighbours whose identifiers match the identifier of the owner node in the first i bits, for $0 \leq i < m$. Each node sends the information stored in its buckets to all overlay neighbours every T_{Rt}^{UIP} seconds. Each time a node learns about a new neighbour, the node sends the information stored in all of its buckets to all overlay neighbours.

VRR is a routing scheme, which can operate on multi-substrate networks, but does not leverage the existing intra-substrate routing. The authors of VRR in [14] show that VRR performs comparably or better than popular MANET routing protocols. [2] Therefore, by comparing landmark domains routing to VRR we can indirectly get a sense of how our routing scheme compares to MANET routing protocols.

In landmark domains routing, leader nodes broadcast RD-IDs every $T_{checkRd}$ seconds. Nodes exchange information from landmark domains routing tables every $T_{checkRt}$ seconds. A node that initiates exchange of information by sending a *pull* message waits for T_{waitRt} seconds to receive *pullReply* messages before broadcasting new routing information. Chord rings are implemented as described in [8]. Each node checks if its successor is still reachable every $T_{stabilize}$ seconds and check if finger nodes

[2] Destination-Sequenced Distance Vector (DSDV) [61], Dynamic Source Routing (DSR) [62], and Ad-hoc On-Demand Distance Vector Routing (AODV) [63].

are reachable every T_{fix} seconds. Nodes also keep r alternative successors in case connectivity to the successor node is lost.

5.2.3 Static Multi-substrate Network

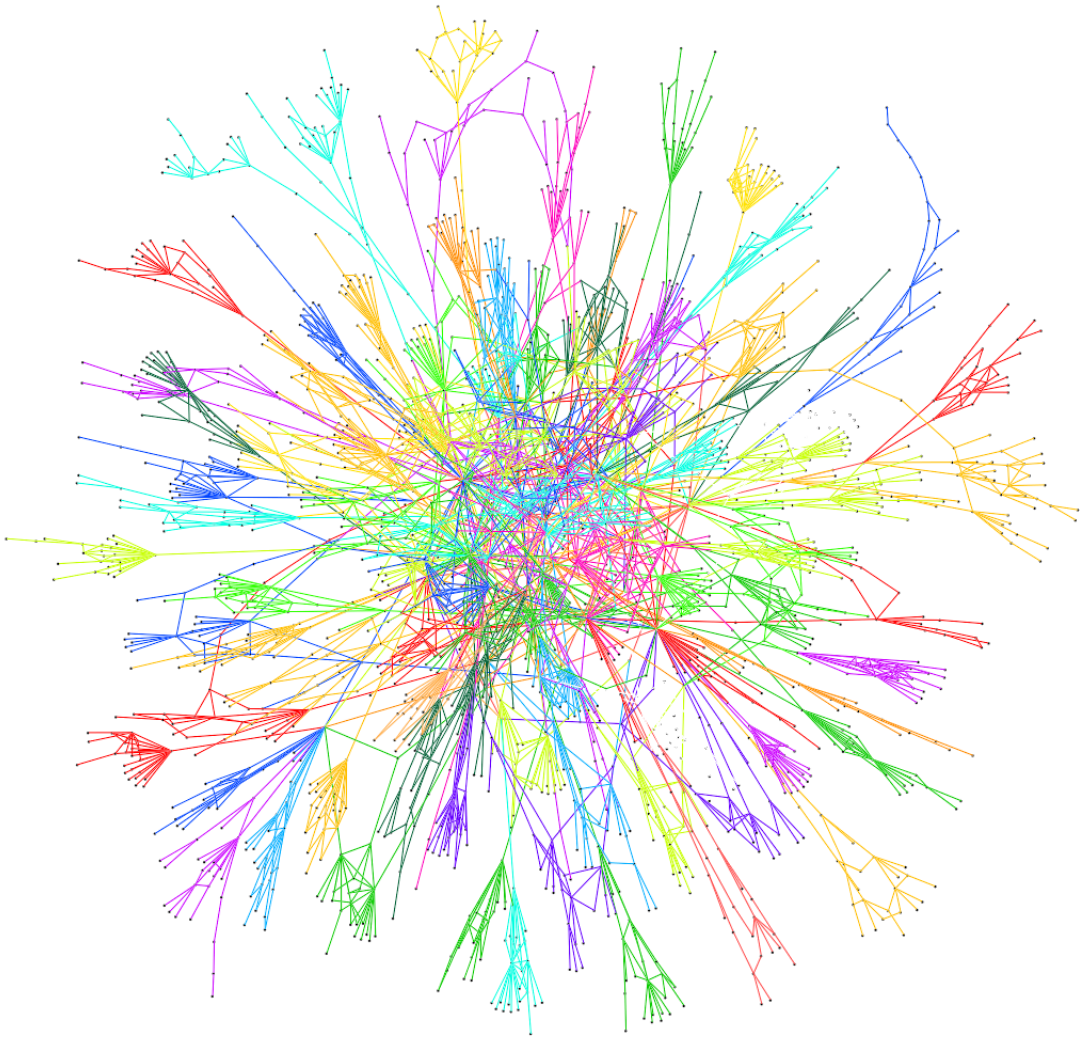


Figure 5.8: Visualization of generated static multi-substrate network ($\alpha^p = 2.2$, $\beta^p = 2.2$, $n_S = 100$, and $\frac{n}{n_S} = 25$). All links in a single substrate network are of the same colour.

To evaluate the performance of landmark domains routing for a static multi-substrate network, we generate a multi-substrate network, with parameters $\alpha^p = 2.2$, and $\beta^p = 2.2$. We vary the number of substrate networks and the number of nodes per substrate network. For each combination of parameters we generate three multi-substrate networks and present average results. For illustration of generated multi-substrate networks, in Fig. 5.8 we show a multi-substrate network generated with $n_S = 100$ and $\frac{n}{n_S} = 25$ as a force-directed graph. All links that belong to a particular substrate network are shown in the same colour (although different substrate network can have the same colour).

To explore the sensitivity of landmark domains routing to the size of substrate networks, we fix the

number of substrate networks to $n_S = 25$ and vary the number of nodes per substrate network, $\frac{n}{n_S}$, from 10 to 200. For landmark domains routing, three RDs are randomly selected as landmark domains. We present the average values of delivery ratio, path stretch, and stored state in Fig. 5.9, Fig. 5.10, and Fig. 5.11, respectively. Error bars show one standard deviation. Values on top of each figure indicate the number of nodes in each substrate network, $\frac{n}{n_S}$. Due to the large number of messages exchanged between nodes, we were unable to simulate UIP for $\frac{n}{n_S} = 200$.

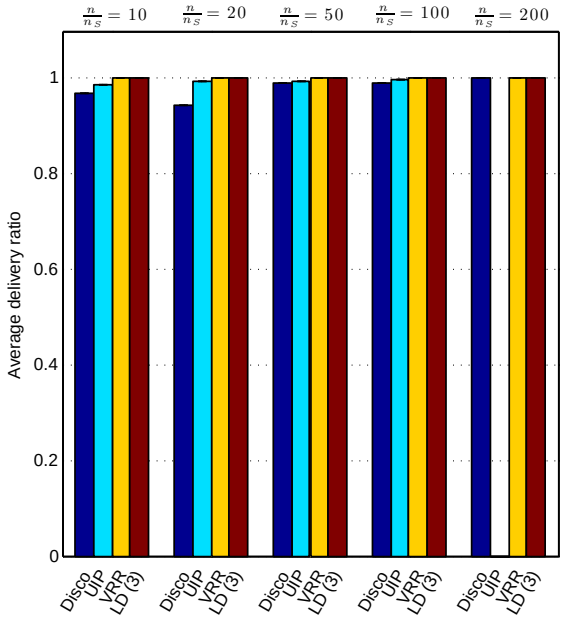


Figure 5.9: Average delivery ratio for a static multi-substrate network as a function of the number of nodes in a substrate network.

Landmark domains routing and VRR deliver all messages (Fig. 5.9). Disco and UIP do not deliver all messages because they impose upper limits on the size of stored routing information. In Disco, node's vicinity can contain more than $\sqrt{n \log n}$ nodes, however a node stores routing entries only for $\sqrt{n \log n}$ randomly selected nodes from its vicinity. A node forwarding a message for a destination in its vicinity, for which the node does not have a routing entry, drops the message. Similarly, in UIP messages are dropped because of the upper limit on the size of buckets. When a new node joins the network, it can replace an existing node X already present in a bucket of a node N. If node N has used the link to node X to form a virtual link to a node Y ($Y \neq X$), by removing X from the bucket the link to node X is also removed and the virtual link to node Y fails. Thus, node N drops messages for node Y.

Average path stretch (Fig. 5.10) of landmark domains routing is comparable to Disco and VRR, and does not grow with the increase of the number of nodes per substrate network. The stored state per node is shown in Fig. 5.11. Landmark domains routing has the smallest amount of stored state for all simulated multi-substrate networks. The stored state of VRR is similar to the stored state of landmark domains routing, with VRR showing much higher variance. All four routing schemes use a small amount of stored state per node, i.e., in order of kilobits. However, the amount of stored state for Disco and UIP grows with the increase of the number of nodes per substrate network, while for landmark domains routing the stored state remains constant.

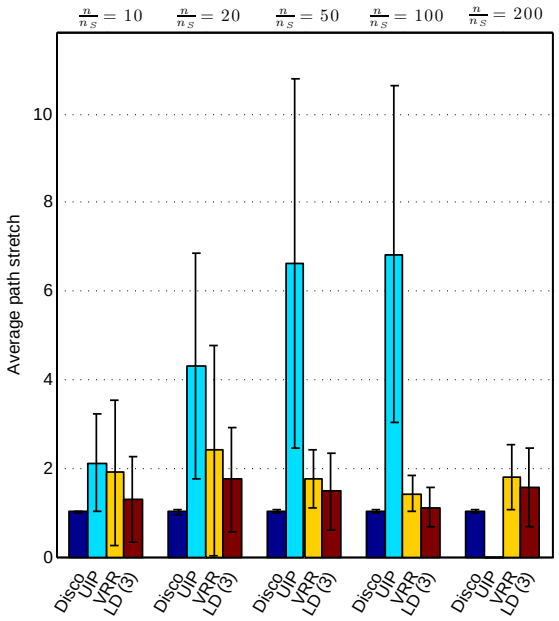


Figure 5.10: Average path stretch for a static multi-substrate network as a function of the number of nodes in a substrate network.

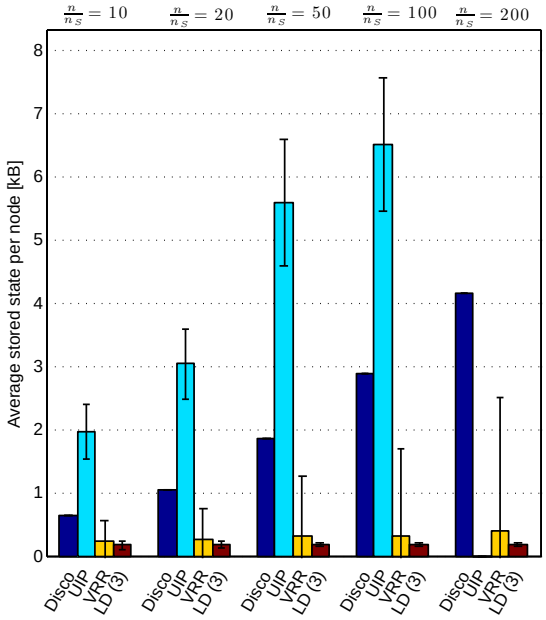


Figure 5.11: Average stored state for a static multi-substrate network as a function of the number of nodes in a substrate network.

Fig. 5.12 and Fig. 5.13 show empirical cumulative distribution functions (CDF) of the path stretch and the stored state, respectively, for a multi-substrate network with $\frac{n}{n_S} = 100$, i.e., 100 nodes per substrate network. We can see that path stretch for Disco remains below its upper bound of 3 for all measured paths. Path stretch for landmark domains routing goes up to 17, however only for a small

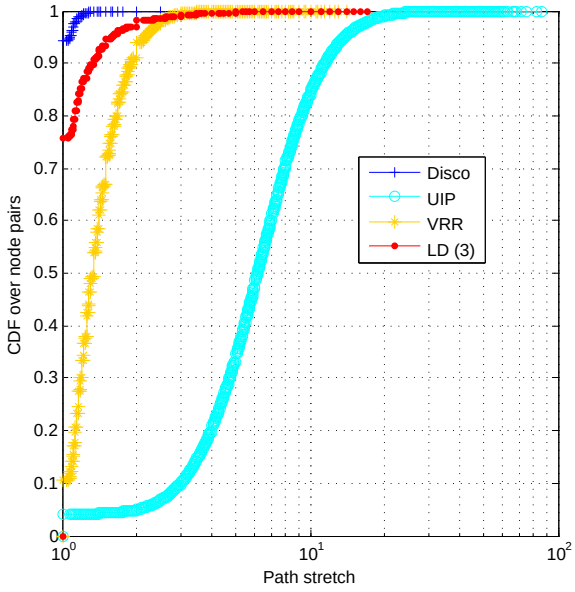


Figure 5.12: Empirical CDF of path stretch for a static multi-substrate network ($\alpha^p = 2.2$, $\beta^p = 2.2$, $n_S = 25$, $\frac{n}{n_S} = 100$).

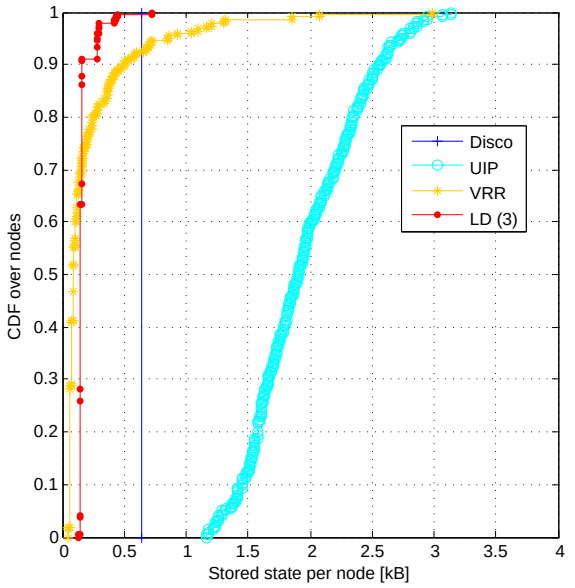


Figure 5.13: Empirical CDF of stored state for a static multi-substrate network ($\alpha^p = 2.2$, $\beta^p = 2.2$, $n_S = 25$, $\frac{n}{n_S} = 100$).

number of paths. In particular, 99.24% of paths have path stretch below 3. Path stretch for UIP is significantly higher, with some paths reaching path stretch of 85.5 and 89.35% of paths having path stretch above 3. Landmark domains routing has on average the smallest amount of stored state of all four compared routing schemes, without major differences in amount of state at individual nodes.

Amount of state per node in VRR is characterized by a majority of nodes storing a small amount of state and a small fraction of nodes storing a significant amount of state. The nodes with the high amount of stored state are the nodes through which a large number of virtual links passes.

To test the sensitivity of landmark domains routing to the number of substrate networks in the multi-substrate network, we fix the number of nodes per substrate network to $\frac{n}{n_S} = 50$. We vary the number of substrate networks n_S between 5 and 50, with $\alpha^p = 2.2$ and $\beta^p = 2.2$ as before. We simulate landmark domains routing with four randomly selected landmark domains (denoted LD(4)) and $0.2 \cdot n_S$ randomly selected landmark domains (denoted LD(*)).

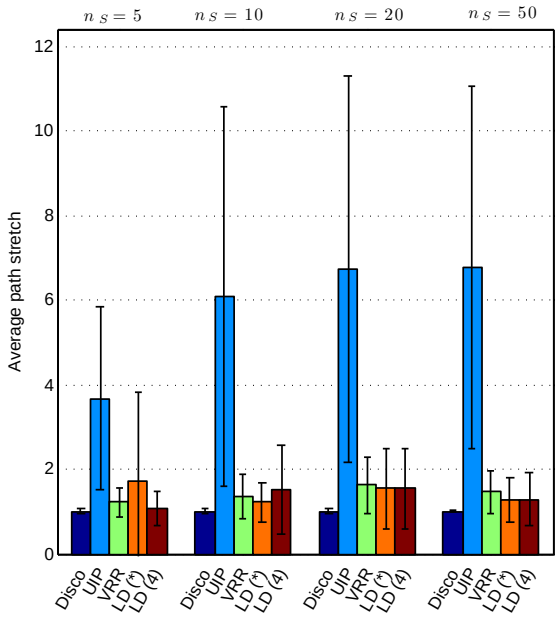


Figure 5.14: Path stretch for a static multi-substrate network as a function of the number of substrate networks.

The delivery ratio is equal to 1 for all protocols. The average paths stretch (Fig. 5.14) and the average stored state (Fig. 5.15) are similar to results when we increase the number of nodes per substrate network. Stored state for LD(*) slightly increases with the increase in the number of substrate networks, since the number of landmark domains also increases with the number of substrate networks.

Next we evaluate the influence of the number of landmark domains used in landmark domains routing. We generate three multi-substrate networks with $n_S = 25$, $\frac{n}{n_S} = 50$, $\alpha^e = 0.08$, and $\beta^p = 2.2$; and present averaged results.

Fig. 5.16 shows the average path stretch and Fig. 5.17 shows the average stored state. We can see that the path stretch decreases as the number of landmark domains increases. Recall from Section 5.1.2 that path stretch depends on how close a landmark domain is to the destination. With the increases in the number of landmark domains, the probability that a landmark domain exists close to the destination node increases and thus path stretch decreases. The simulated multi-substrate network is to small for the length of the shortest path from the source to the destination to dominate the path stretch. The stored state increases proportionally to the number of landmark domains, since every node needs to store routing entries for each landmark domain.

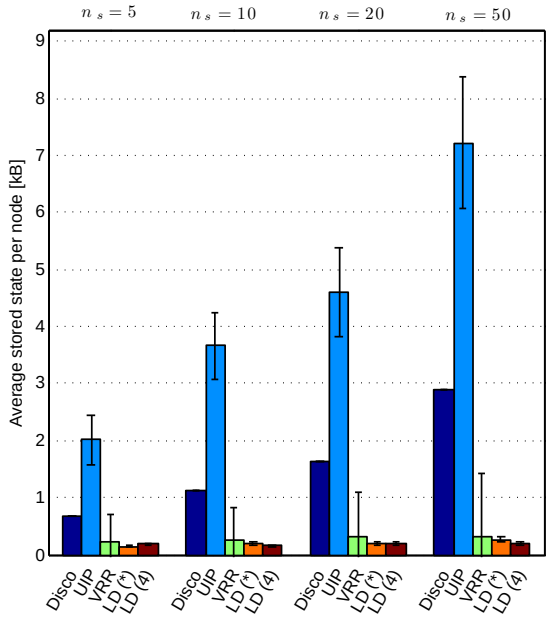


Figure 5.15: Stored state for a static multi-substrate network as a function of the number of substrate networks.

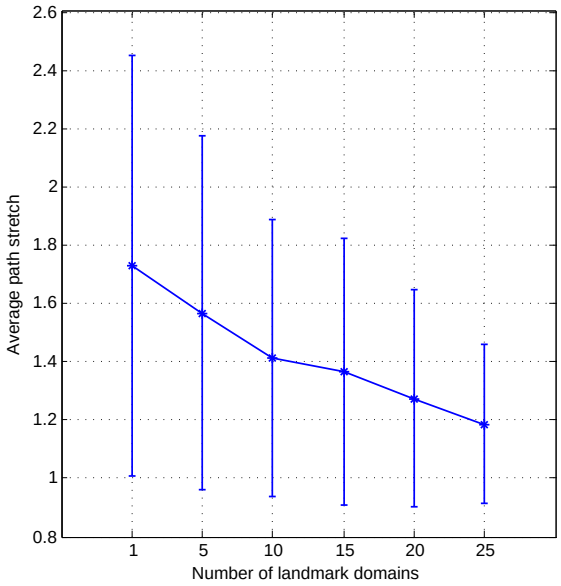


Figure 5.16: Path stretch of landmark domains routing for a static multi-substrate network as a function of the the number of landmark domains.

5.2.4 Dynamic Multi-substrate Network

In this section we explore operation of landmark domains routing in dynamic multi-substrate networks, created using the parameters presented in Table 5.5. We create multi-substrate networks with routers

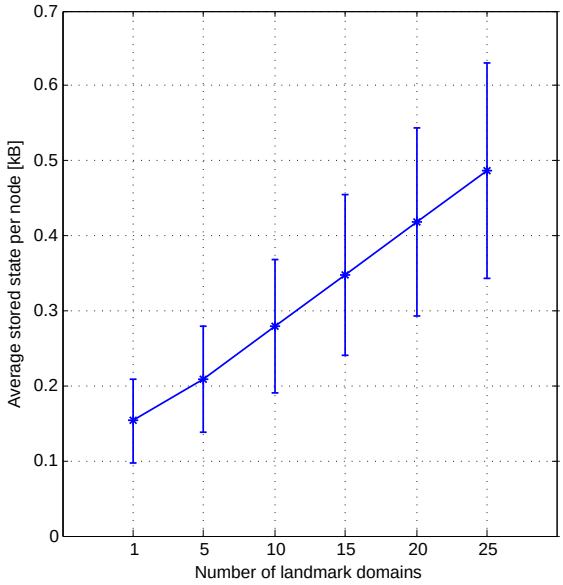


Figure 5.17: Stored state of landmark domains routing for a static multi-substrate network as a function of the number of landmark domains.

Parameter	Value
n_S	10
$\frac{n}{n_S}$	50
α^p	2.2
β^p	2.2
n_{WS}	10
n^s_{WI}	1
n^m	200
r_{WI}	20 m

Parameter	Value
n_G	20
$\frac{n^m}{n_G}$	10
r_G	75 m
v_{min}	0 m/s
v_{max}	1 m/s
p	5 s
f_x	400 m
f_y	400 m

Table 5.5: Values of parameters for dynamic multi-substrate networks.

that participate only in the intra-substrate routing, and not in the multi-substrate routing. Since Disco and VRR assume all devices in the multi-substrate network participate in the multi-substrate routing, and rely on the exchange of messages between neighbour nodes in substrate networks, we do not simulate them for dynamic multi-substrate networks. For each set of parameters we generate three multi-substrate networks and present average results.

To test the sensitivity of landmark domains routing to node movement we vary the maximum speed v_{max} of mobile nodes. We pick 5 landmark domains either in static substrate networks or mobile substrate networks and designate this by LD(5, static) and LD(5, mobile) respectively.

Fig. 5.18 shows the average delivery ratio for landmark domains routing and UIP as the maximum speed of mobile nodes is increased. Landmark domains routing clearly outperforms UIP at all node speeds. As the maximum node speed increases, the average delivery ratio of landmark domains routing decreases, since it cannot discover new routing paths as fast as the connections between nodes change. However, the average path stretch (Fig. 5.19) of our routing scheme remains low for the delivered

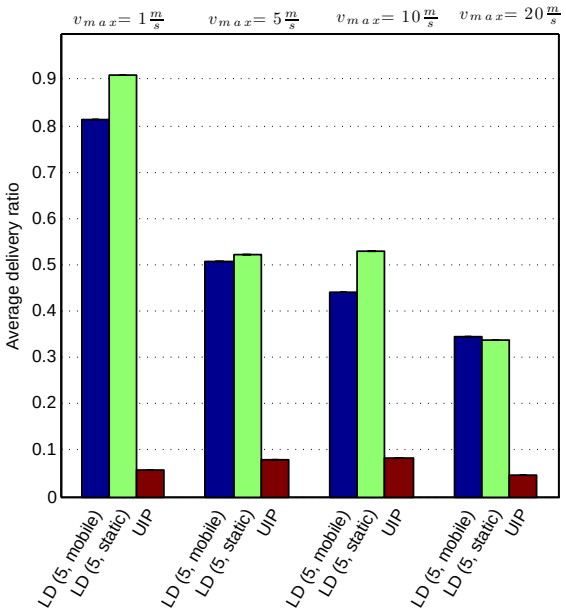


Figure 5.18: Average delivery ratio for a dynamic multi-substrate network as a function of the maximum node speed.

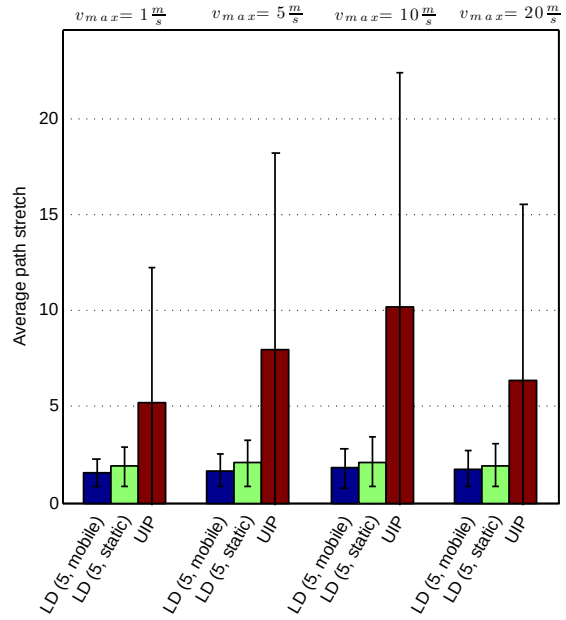


Figure 5.19: Average path stretch for a dynamic multi-substrate network as a function of the maximum node speed.

messages. The average stored state and the message overhead (Fig. 5.20 and Fig. 5.21) show a significant difference between the selection of static and mobile substrate networks for landmark domains. Landmark domains in dynamic substrate networks produce a higher amount of message overhead since paths to landmark domains change more frequently and landmark domains routing tables need to be updated

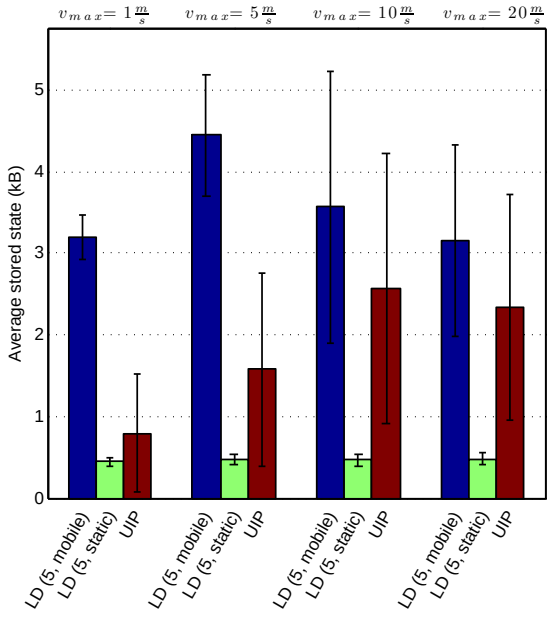


Figure 5.20: Average stored state for a dynamic multi-substrate network as a function of the maximum node speed.

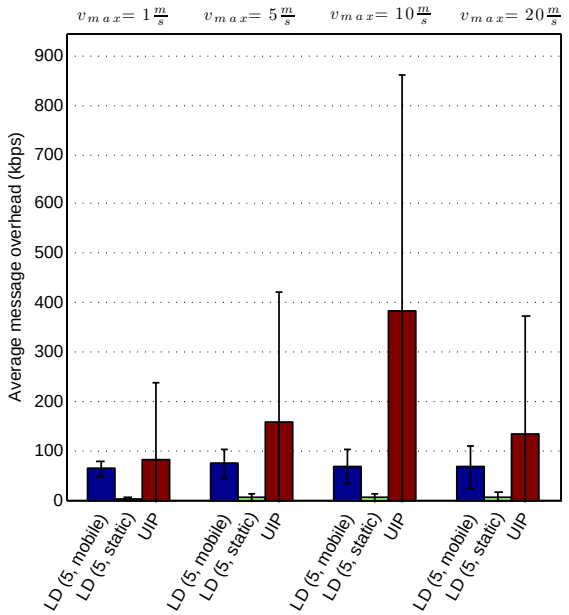


Figure 5.21: Average message overhead for a dynamic multi-substrate network as a function of the maximum node speed.

more often. The average stored state is higher for landmark domains in dynamic substrate networks, since they change their RD-IDs more often than landmark domains in static substrate networks. That is, landmark domains in dynamic substrate networks experience more splits and merges. A change of RD-ID of a landmark domain means that a new entry in landmark domains routing tables needs to be

crated and propagated throughout the network.

Next, we explore the impact of the ratio of static to dynamic substrate networks on landmark domains routing. We use the parameters from Table 5.5, but vary the number of static substrate networks, n_S, from $n_S = 5$ to $n_S = 50$. Landmark domains are selected in static substrate networks. We simulate landmark domains routing with five landmark domains and a number of landmark domains proportional to the number of static substrate networks ($0.2 \cdot n_S$), indicated as LD(5, static) and LD (*, static), respectively.

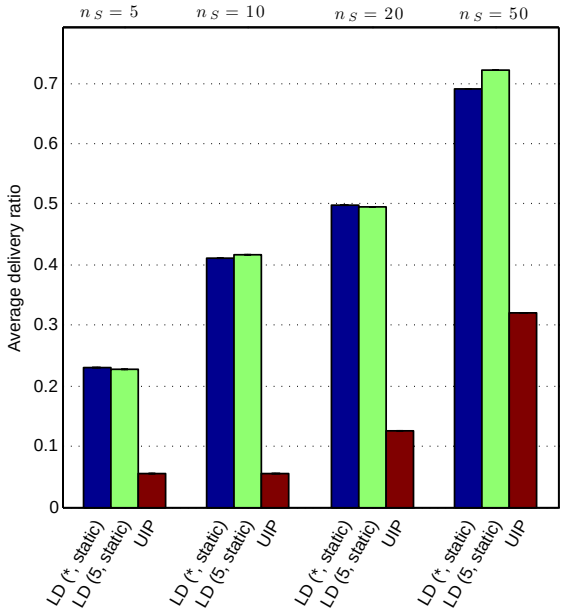


Figure 5.22: Average delivery ratio for a dynamic multi-substrate network as a function of the number of static substrate networks.

The average delivery ratio (Fig. 5.22) increases with the increase in the number of static substrate networks for both landmark domains routing and UIP. With the increase in the number of static substrate networks, both routing schemes benefit from the reduced frequency of change of routing paths. Values of the average path stretch (Fig. 5.23), stored state (Fig. 5.24), and message overhead (Fig. 5.25) remain low for landmark domains routing, for all number of static substrate networks.

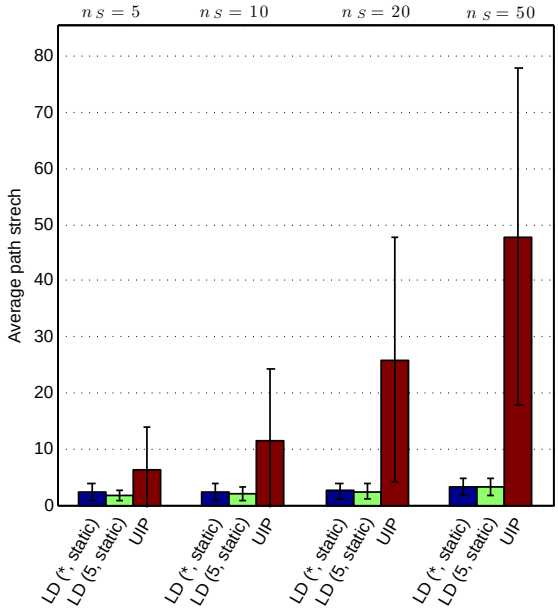


Figure 5.23: Average path stretch for a dynamic multi-substrate network as a function of the number of static substrate networks.

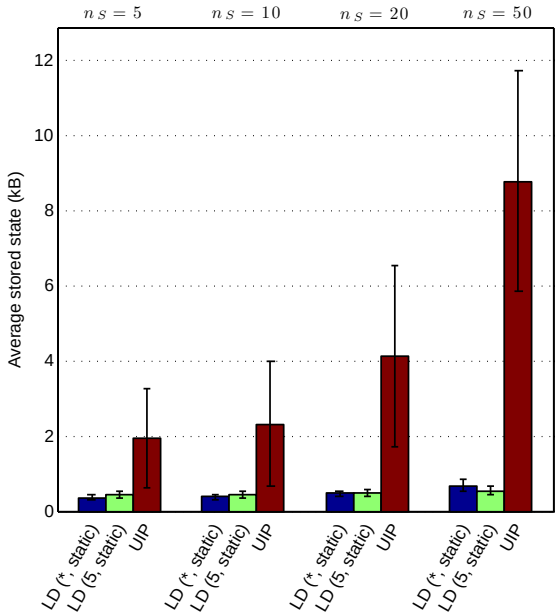


Figure 5.24: Average stored state for a dynamic multi-substrate network as a function of the number of static substrate networks.

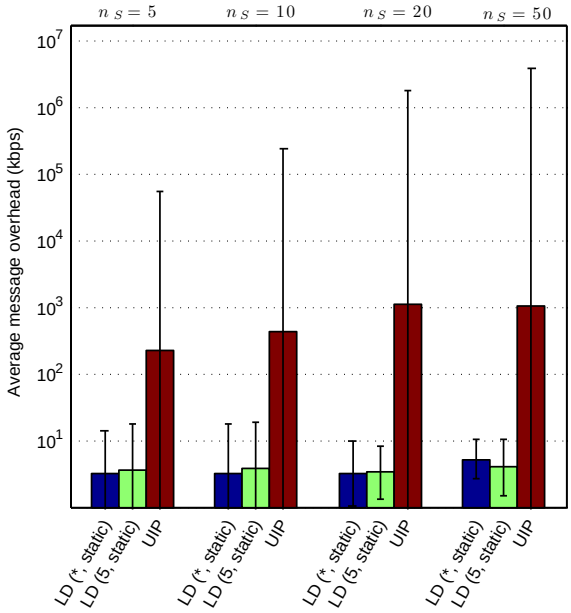


Figure 5.25: Average message overhead for a dynamic multi-substrate network as a function of the number of static substrate networks.

Chapter 6

Conclusion and Future Work

In this chapter we provide the conclusion and present some ways in which the work presented in this thesis can be extended in the future.

6.1 Conclusion

In this thesis, we present landmark domains routing, a scalable routing scheme, which operates on a multi-substrate network composed of static and dynamic substrate networks. We have defined a new addressing scheme, in which node locations are expressed relative to stable positions in the multi-substrate network, called landmark domains. The routing abandons the goal of creating globally consistent routing tables, and instead is based on finding routes to and from landmark domains.

We defined the concept of 'reachability domain', as a region of a substrate network in which all nodes can exchange messages. We use reachability domains as basic building blocks for our routing scheme and select a subset of them as landmark domains. We let nodes set up routing paths to landmark domains. Further, we let each node independently discover a path from a landmark domain to itself and use this path to create a locator, which describes how to reach a node from anywhere in the network. We show that landmark domains routing performs comparably to other routing schemes for multi-substrate networks in a static setting. In a dynamic network, landmark domains routing clearly outperformed UIP in path stretch, delivery ratio, and stored state, while producing orders of magnitude smaller overhead. We also show that selection of stable landmark domains greatly reduces overhead when compared to less stable landmark domains. Through numerical analysis we show that the proposed routing is scalable and requires at most the same amount of stored state as a compact routing scheme. We demonstrate that in large multi-substrate networks, landmark domains routing produces paths with low average path stretch and almost all paths have path stretch below three.

6.2 Future Work

The work presented in this thesis can be extended in several ways.

1. **Selection of Landmark Domains:** We have shown the advantage of selecting landmark domains defined on stable substrate networks, but we did not investigate methods to determine the stability of a substrate network. It is desirable to define a metric that would express the stability of

reachability domains, measure this stability in the multi-substrate network, and conduct selection of landmark domains based on the metric.

2. **Identifier Resolution Service:** In landmark domains routing all nodes have persistent identifiers and ephemeral locators. Since sender of a message knows the destination identifier but not the locator, a service that resolves identifiers into locators is necessary. There are several options for developing this service, e.g., a centralized server or a distributed resolution system modelled after the domain name system. However, as node locators are created and updated by nodes themselves, a system in which node locators are stored at the nodes is also a viable option. A DHT structure can be used to realize a distributed identifier resolution service. This DHT can be built over a substrate network with universal connectivity (provided by landmark domains routing), or directly on top of the multi-substrate network.

3. **Broadcasting in Reachability Domains:** We use a Chord overlay network to establish a routing structure that allows us to broadcast protocol messages to entire reachability domains and also monitor for merges and splits. Future work can investigate how do other overlay networks perform in this context. Further improvements in this area can be made by leveraging multicasting capabilities of some substrate networks to reduce overhead.

Bibliography

[1] A. Feldmann, "Internet clean-slate design: what and why?" *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 59–64, Jul. 2007.

[2] J. Rexford and C. Dovrolis, "Future Internet architecture: clean-slate versus evolutionary research," *Communications of the ACM*, vol. 53, no. 9, pp. 36–40, Sep. 2010.

[3] L. Peterson, T. Anderson, D. Blumenthal, D. Casey, D. Clark, D. Estrin, J. Evans, D. Raychaudhuri, M. Reiter, J. Rexford, S. Shenker, and J. Wroclawski, "GENI design principles," *IEEE Computer Magazine*, vol. 39, no. 9, pp. 102–105, Sep. 2006.

[4] N. Niebert, S. Baucke, I. El-Khayat, M. Johnsson, B. Ohlman, H. Abramowicz, K. Wuenstel, H. Woesner, J. Quittek, and L. M. Correia, "The way 4WARD to the creation of a future Internet," in *Proc. PIMRC*, pp. 1–5, Sep. 2008.

[5] D. D. Clark, "The design philosophy of the DARPA Internet protocols," *SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 106–114, Aug. 1988.

[6] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, Nov. 1984.

[7] M. S. Blumenthal and D. D. Clark, "Rethinking the design of the Internet: the end-to-end arguments vs. the brave new world," *ACM TOIT*, vol. 1, no. 1, pp. 70–109, Aug. 2001.

[8] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. ACM SIGCOMM*, pp. 149–160, Aug. 2001.

[9] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. SOSP*, pp. 131–145, Oct. 2001.

[10] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proc. 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp. 329–350, Nov. 2001.

[11] D. D. Clark, C. Partridge, R. T. Braden, B. Davie, S. Floyd, V. Jacobson, D. Katabi, G. Minshall, K. K. Ramakrishnan, T. Roscoe, I. Stoica, J. Wroclawski, and L. Zhang, "Making the world (of communications) a different place," in *Proc. ACM SIGCOMM*, pp. 91–96, Jul. 2005.

[12] A. Singla, P. B. Godfrey, K. Fall, G. Iannaccone, and S. Ratnasamy, "Scalable routing on flat names," in *Proc. ACM CoNEXT*, pp. 20:1–20:12, Dec. 2010.

[13] B. Ford, "Unmanaged Internet protocol: taming the edge network management crisis," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 93–98, Jan. 2004.

[14] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron, "Virtual ring routing: network routing inspired by DHTs," in *Proc. ACM SIGCOMM*, pp. 351–362, Aug. 2006.

[15] J. F. Shoch, "A note on inter-network naming, addressing, and routing," IEN (Internet Experiment Note) 019, Jan. 1978.

[16] J. Saltzer, "On the naming and binding of network destinations," RFC 1498, Internet Engineering Task Force, Aug. 1993.

[17] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden, "Tussle in cyberspace: defining tomorrow's Internet," in *Proc. SIGCOMM*, pp. 347–356, Aug. 2002.

[18] B. Ahlgren, M. Brunner, L. Eggert, R. Hancock, and S. Schmid, "Invariants: a new design methodology for network architectures," in *Proc. ACM SIGCOMM workshop on Future Directions in Network Architecture (FDNA)*, pp. 65–70, Aug. 2004.

[19] D. D. Clark, K. Sollins, J. Wroclawski, and T. Faber, "Addressing reality: an architectural response to real-world demands on the evolving internet," in *Proc. ACM SIGCOMM workshop on Future Directions in Network Architecture (FDNA)*, pp. 247–257, Aug. 2003.

[20] P. Francis, "Pip near-term architecture," RFC 1621, Internet Engineering Task Force, May 1994.

[21] A. Jonsson, M. Folke, and B. Ahlgren, "The split naming/forwarding network architecture," in *Proc. First Swedish National Computer Networking Workshop (SNCNW 2003)*, Sep. 2003.

[22] R. Bless, C. Hiibsch, S. Mies, and O. P. Waldhorst, "The underlay abstraction in the spontaneous virtual networks (SpoVNet) architecture," in *Proc. Euro-NGI Conference on Next Generation Internet Networks (NGI)*, pp. 115 –122, Apr. 2008.

[23] S. Mies, O. P. Waldhorst, and H. Wippel, "Towards end-to-end connectivity for overlays across heterogeneous networks," in *Proc. IEEE ICC Workshops 2009*, pp. 1–6, Jun. 2009.

[24] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks; performance evaluation and optimization," *Computer Networks*, vol. 1, no. 3, pp. 155 – 174, Jan. 1977.

[25] P. F. Tsuchiya, "The landmark hierarchy: a new hierarchy for routing in very large networks," in *Proc. ACM SIGCOMM*, pp. 35–42, Aug. 1988.

[26] D. Peleg and E. Upfal, "A tradeoff between space and efficiency for routing tables," in *Proc. ACM Symposium on Theory of Computing (STOC)*, pp. 43–52, May 1988.

[27] C. Gavoille and M. Gengler, "Space-efficiency for routing schemes of stretch factor three," *J. Parallel Distrib. Comput.*, vol. 61, no. 5, pp. 679–687, May 2001.

[28] M. Thorup and U. Zwick, "Approximate distance oracles," in *Proc. ACM Symposium on Theory of Computing (STOC)*, pp. 183–192, Jul. 2001.

[29] L. J. Cowen, "Compact routing with minimum stretch," in *Proc. ACM/SIAM symposium on Discrete algorithms (SODA)*, pp. 255–260, Jan. 1999.

[30] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg, "Compact distributed data structures for adaptive routing," in *Proc. ACM Symposium on Theory of Computing (STOC)*, pp. 479–489, May 1989.

[31] M. Arias, L. J. Cowen, K. A. Laing, R. Rajaraman, and O. Taka, "Compact routing with name independence," in *Proc. ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pp. 184–192, Jun. 2003.

[32] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup, "Compact name-independent routing with minimum stretch," in *Proc. ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pp. 20–24, Jun. 2004.

[33] C. Westphal and J. Kempf, "A compact routing architecture for mobility," in *Proc. ACM MobiArch*, pp. 1–6, Aug. 2008.

[34] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *Proc. ACM MobiCom*, pp. 96–108, Sep. 2003.

[35] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica, "Beacon vector routing: scalable point-to-point routing in wireless sensornets," in *Proc. symposium on Networked Systems Design & Implementation (NSDI)*, pp. 329–342, May 2005.

[36] J. Newsome and D. Song, "GEM: Graph EMbedding for routing and data-centric storage in sensor networks without geographic information," in *Proc. SenSys*, pp. 76–88, Nov. 2003.

[37] J. Herzen, C. Westphal, and P. Thiran, "Scalable routing easy as PIE: a practical isometric embedding protocol," in *Proc. IEEE ICNP*, pp. 49–58, Oct. 2011.

[38] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," Computer Science Division, University of California, Berkley, Tech. Rep. UCB/CSD-01-1141, Apr. 2001.

[39] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, "ROFL: routing on flat labels," in *Proc. ACM SIGCOMM*, pp. 363–374, Aug. 2006.

[40] B. Ford, "Scalable Internet routing on topology-independent node identities," Massachusetts Institute of Technology, Tech. Rep., Oct. 2003.

[41] R. M. Hinden, "New scheme for Internet routing and addressing (ENCAPS) for IPNG," RFC 1955, Internet Engineering Task Force, Jun. 1996.

[42] D. Farinacci, V. Fuller, M. David, and L. Darrel, "Locator/ID separation protocol (LISP)," IETF - draft-ietf-lisp-22. Feb. 2012.

[43] R. Atkinson and S. Bhatti, "ILNP architectural description," iETF - draft-irtf-rrg-ilnp-arch-06.txt. Jul. 2012.

[44] X. Xu, "Routing architecture for the next generation Internet (RANGI)," IETF - draft-xu-rangi-04.txt. Aug. 2010.

[45] P. Francis and R. Gummadi, "IPNL: A NAT-extended Internet architecture," in *Proc. ACM SIGCOMM*, pp. 69–80, Aug. 2001.

[46] Z. Turányi, A. Valkó, and A. T. Campbell, "4+4: an architecture for evolving the Internet address space back toward transparency," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 5, pp. 43–54, Oct. 2003.

[47] M. O'Dell, "GSE - An alternate addressing architecture for IPv6," IETF - draft-ietf-ipngwg-gseaddr-00.txt. Feb. 1997.

[48] B. Ahlgren, J. Arkko, L. Eggert, and J. Rajahalme, "A node identity internetworking architecture," in *Proc. IEEE INFOCOM*, pp. 1–6, Apr. 2006.

[49] A. Feldmann, L. Cittadini, W. Mühlbauer, R. Bush, and O. Maennel, "HAIR: hierarchical architecture for Internet routing," in *Proc. 2009 workshop on Re-architecting the Internet (ReArch)*, pp. 43–48, Dec. 2009.

[50] S. Schmid, L. Eggert, M. Brunner, and J. Quittek, "Towards autonomous network domains," in *Proc. IEEE INFOCOM*, pp. 2847–2852, Mar. 2005.

[51] D. Clark, R. Braden, A. Falk, and V. Pingali, "FARA: reorganizing the addressing architecture," in *Proc. ACM SIGCOMM workshop on Future directions in network architecture (FDNA)*, pp. 313–321, Aug. 2003.

[52] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield, "Plutarch: an argument for network pluralism," in *Proc. ACM SIGCOMM workshop on Future directions in network architecture (FDNA)*, pp. 258–266, Aug. 2003.

[53] T. Shafaat, A. Ghodsi, and S. Haridi, "Handling network partitions and mergers in structured overlay networks," in *Proc. IEEE P2P 2007*, pp. 132–139, Sep. 2007.

[54] S. El-Ansary, L. O. Alima, P. Brand, and S. Haridi, "Efficient broadcast in structured P2P networks," in *Proc. International workshop on Peer-To-Peer Systems (IPTPS)*, Feb. 2003.

[55] A. Varga, "The OMNET++ discrete event simulation system," in *Proc. European Simulation Multiconference (ESM)*, pp. 319–324, Jun. 2001.

[56] (2013, May) OMNeT++ Home Page. [Online]. Available: http://www.omnetpp.org/

[57] R. Eggleton and D. Holton, "Graphic sequences," in *Combinatorial Mathematics VI*, ser. Lecture Notes in Mathematics, A. Horadam and W. Wallis, Eds. Springer Verlag, 1979, vol. 748, pp. 1–10.

[58] P. Mahadevan, D. Krioukov, K. Fall, and A. Vahdat, "Systematic topology analysis and generation using degree correlations," in *Proc. ACM SIGCOMM*, pp. 135–146, Aug. 2006.

[59] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang, "A group mobility model for ad hoc wireless networks," in *Proc. ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pp. 53–60, Aug. 1999.

[60] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn, "Bonnmotion: a mobility scenario generation and analysis tool," in *Proc. ICST Conference on Simulation Tools and Techniques (SIMUTools)*, pp. 51:1–51:10, Apr. 2010.

[61] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proc. ACM SIGCOMM*, pp. 234–244, Aug. 1994.

[62] D. B. Johnson and D. A. Maltz, *Dynamic source routing in ad hoc wireless networks*, in *Mobile Computing*, T. Imielinski and H. Korth, Eds., pp. 153–181. Kluwer Academic Publishers, Jan. 1996.

[63] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pp. 90–100, Feb. 1999.