Resilient Scalable Internet Routing and Embedding Algorithms

Foutbestendige schaalbare internetrouting en plaatsingsalgoritmen

Seyedeh Sahel Sahhaf

Promotoren: prof. dr. ir. D. Colle, dr. W. Tavernier
Proefschrift ingediend tot het behalen van de graad van
Doctor in de ingenieurswetenschappen: computerwetenschappen

UNIVERSITEIT
GENT

Vakgroep Informatietechnologie
Voorzitter: prof. dr. ir. D. De Zutter
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2016 - 2017

ISBN 978-90-8578-948-2
NUR 986, 988
Wettelijk depot: D/2016/10.500/80

Ghent University
Faculty of Engineering and Architecture
Department of Information Technology

Promoters: prof. dr. ir. Didier Colle
dr. Wouter Tavernier

Jury Members: prof. dr. ir. Patrick De Baets, Ghent University (Chairman)
prof. dr. ir. Didier Colle, Ghent University (Supervisor)
dr. Wouter Tavernier, Ghent University (Supervisor)
prof. dr. ir. Mario Pickavet, Ghent University (Secretary)
prof. dr. Deep Medhi, University of Missouri-Kansas City
prof. dr. ir. Davide Careglio, Universitat Politécnica de Catalunya
prof. dr. Veerle Fack, Ghent University
dr. ir. Bruno Volckaert, Ghent University

Ghent University
Faculty of Engineering and Architecture

Department of Information technology
Technologiepark-Zwijnaarde 15, 9052 Ghent, Belgium

Tel.: +32-9-331.49.00
Fax.: +32-9-331.48.99

Dissertation to obtain the degree of
Doctor of Computer Science Engineering
Academic year 2016-2017

Acknowledgments

This journey would have not been possible without the support of my family, promoters, mentors, colleagues and friends.

Firstly, I would like to express my sincere gratitude to my promoters Prof. Didier Colle and Dr. Wouter Tavernier for the continuous support of my Ph.D study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of the research and writing of this thesis. I am also thankful to Prof. Mario Pickavet for all the constructive feedback, suggestions and research ideas.

Besides my promoters, I would like to thank the rest of my PhD jury: Prof. Patrick De Baets, Prof. Davide Careglio, Prof. Deep Medhi, Prof. Veerle Fack, Prof. Mario Pickavet, and Dr. Bruno Volckaert, for their insightful comments, questions and encouragement which helped me to improve the quality of my PhD dissertation.

I am grateful to Prof. Piet Demeester for providing me the opportunity to join the IBCN research group and carry out my Ph.D research at Ghent University. I would like to thank all the staff members at IBCN and special thank to Mike Van Puyenbroeck for his guidance during the initial admission phase even before I arrive in Belgium. Also grateful to Martine Buysse and Davinia Stevens for their constant support.

I also take this opportunity to thank all the members of projects: EULER, UNIFY, MECANO and EMD, in which I was involved during my Ph.D. Thank you for the interesting research discussions which were really helpful in achieving my Ph.D thesis.

I am thankful to my colleagues in the office: Sachin, Sander, Sofie, Steven, Thomas, Ratul, Pieter, Maarten, Thijs, Dimitri, Wouter, Ward, Abhishek, Ludwig, Willem and Sofie. Thank you all for your kind and friendly behavior which made these 6 years a pleasant experience for me. Also I thank my other colleagues and friends I met in IBCN during these few years: Maryam, Bahareh, Nasrin, Elnaz, Marlies, Ankita, Mahdi, Bram, Frederic, Domenico, Rodrigo, Wei, Merima, Joke and Andy. A special thank to Thomas who encouraged me to take Dutch courses and who was so kind and patient to plan some 'Dutch lunches' so that we could practice it.

I am grateful to my college/high-school friends: Milad, Sepideh, Andisheh and Shohreh who supported me during the rough times even though they were far away.

I would like to thank my family: my parents and my sister for supporting me

spiritually throughout my Ph.D and my life in general.

Last but not the least, many thanks to Kavoos, for all his love and support. I thank him for being patient and supportive from the beginning of this Ph.D until the final defense. I am grateful to him for always being there for me through all the ups and downs and for being in my life.

Ghent, Nov 2016
Seyedeh Sahel Sahhaf

Table of Contents

List of Figures

List of Tables

List of Acronyms

A

AH	Acceleration Hardware
API	Application Programming Interface
ARP	Address Resolution Protocol
AS	Autonomous System
ASN	Autonomous System Number

B

B-A	Barabasi-Albert
BFD	Bidirectional Forwarding Detection
BFS	Breadth First Search
BGP	Border Gateway Protocol

C

CA	Controller Adapter
CAPEX	Capital Expenditure
CCN	Content-Centric Networking
CF	Cluster Factor
CIDR	Classless Inter-Domain Routing
CNF	Compound Network Function

CPE Customer-Premises Equipment
CPU Central Processing Unit
CS Coordinate Set
CV Coordinate Vector

D

DARPA Defense Advanced Research Projects Agency
DB Database
DC Data Center
DCR Distributed Compact Routing
DHCP Dynamic Host Configuration Protocol
DHT Distributed Hash Table
DNS Domain Name Server
DoV Domain Virtualizer
DPDK Data Plane Development Kit
DPI Deep Packet Inspection
DSBM Decomposition Selection-Backtracking Mapping
DSL Digital Subscriber Line
DTN Delay/Disruption-Tolerant Network

E

EGP Exterior Gateway Protocol
EIGRP Enhanced Interior Gateway Routing Protocol
ENF Elementary Network Function
ESCAPE Extensible Service ChAin Prototyping Environment
ETSI European Telecommunications Standards Institute

F

FE Forwarding Element

F.F Flip Flop

FIB Forwarding Information Base

FPGA Field-Programmable Gate Array

FW Firewall

G

GB Gigabyte

GCLS Geometric Coordinate-Labeling Scheme

GCMR Greedy Compact Multicast Routing

GHz Gigahertz

GLP Generalized Linear Preference

GML Geography Markup Language

GMPLS Generalized Multiprotocol Label Switching

GPS Global Positioning System

GPU Graphics Processing Unit

GTR Greedy Tree-based geometric Routing/Geometric Tree-based greedy Routing

GUI Graphical User Interface

GW Gateway

H

HA High-Availability

HIP Host Identity Protocol

HTTP Hypertext Transfer Protocol

I

ICMP Internet Control Message Protocol
ID Identifier
IDS Intrusion Detection System
IEEE Institute of Electrical and Electronics Engineers
IETF Internet Engineering Task Force
IGMP Internet Group Management Protocol
IGP Interior Gateway Protocol
IGRP Interior Gateway Routing Protocol
IL Infrastructure Layer
ILP Integer Linear Programming
InP Infrastructure Provider
I/O Input/Output
IoT Internet of Things
IP Internet Protocol
IPv4 Internet Protocol version 4
IPv6 Internet Protocol version 6
IRTF Internet Research Task Force
IS-IS Intermediate System to Intermediate System
ISG Industry Specifications Group
IXP Internet Exchange Points

K

KQI Key Quality Indicators

L

LAN Local Area Network

LISP Locator/Identifier Separation Protocol
LPM Longest Prefix Match
LSA Link-State Advertisement
LTE Long-Term Evolution
LXC Linux Container

M

MANO Management and Orchestration
MCN Mobile Cloud Networking
MDT Multicast Distribution Tree
MRAI Minimum Route Advertisement Interval
MTBF Mean Time Between Failures
MTTR Mean Time To Repair

N

NAT Network Address Translation
NF Network Function
NFEP Network Function Embedding Problem
NF-FG Network Function Forwarding Graph
NFG Network Function Graph
NF-IB Network Function Information Base
NFV Network Function Virtualization
NFVRG Network Function Virtualization Research Group
NSC Network Service Chaining
NTP Network Time Protocol

O

OA Optical Amplifier

O/E/O Optical/Electrical/Optical
OL Orchestration Layer
OPEX Operational Expenditure
OSI Open Systems Interconnection
OSPF Open Shortest Path First
OXC Optical Cross Connect

P

P2P Peer-to-Peer
PC Personal Computer
PD Poincare Disk
PIM Protocol Independent Multicast
PoC Proof of Concept
PoP Point of Presence
PPP Point to Point Protocol

Q

QoS Quality of Service

R

RAM Random Access Memory
RD Route Discovery
RG Resource Graph
RIB Routing Information Base
RIP Routing Information Protocol
RO Resource Orchestrator
RSTP Rapid Spanning Tree Protocol

RT Routing Table

S

SAP Service Access Point
SDN Software Defined Networking
SDNRG Software Defined Networking Research Group
SFC Service Function Chain
SG Service Graph
SID Subgraph Isomorphism Detection
SL Service Layer
SLA Service Level Agreement
SMTP Simple Mail Transfer Protocol
SPEC Standard Performance Evaluation Corporation
STP Spanning Tree Protocol

T

TAU Tuning and Analysis Utilities
TCP Transmission Control Protocol

U

UDP User Datagram Protocol

V

VIM Virtualized Infrastructure Manager

VLAN Virtual Local Area Network
VNEP Virtual Network Embedding Problem
VNF Virtual Network Function
VM Virtual Machine
VPN Virtual Private Network

W

WDM Wavelength Division Multiplexing
WMGR Word-Metric-based Greedy Routing
WMS Word-Metric Space
WSN Wireless Sensor Network

Samenvatting
– Summary in Dutch –

Het Internet is het fundamentele communicatiemedium dat mensen, bedrijven en organisaties met elkaar in contact brengt. Gegevens, nieuws en opinies worden momenteel uit meer dan 190 landen via het Internet uitgewisseld. Ieder aspect van onze leven, van sociale tot economische aard, wordt hierdoor beïnvloed. We ontvangen iedere dag nieuws via het Internet en delen informatie en bestanden via 'the cloud'. Daarenboven delen multinationals continue informatie via 'streaming' (bijvoorbeeld radio- en televisieprogramma's van grootschalige sport- of lifeevenementen). Dergelijke toepassingen stellen strikte voorwaarden in termen van kwaliteit en capaciteit van het onderliggende medium.

Nochtans was het Internet oorspronkelijk niet ontwikkeld voor dergelijke hoge eisen. Het initiële idee van het Internet begon met een poging om de topuniversiteiten uit de Verenigde Staten met elkaar te verbinden. Hiermee konden ze al hun onderzoeksgegevens en -infrastructuur delen zonder daarbij tijd of andere middelen te verspillen. Daarom was ARPANET als eerst pakketgeschakeld netwerk gecreëerd. Het netwerk conecteerde vier knooppunten in Universiteit van Californië, Los Angeles, onderzoek center in Stanford, Universiteit van Californië in Santa Barbara en Universiteit van Utah. Na deze succesvolle onderneming in 1969, werd het ware potentieel van dit medium steeds duidelijker voor experts. Sindsdien heeft het zich steeds verder ontwikkeld en uitgebreid, leidend tot het grootschalige en heterogene netwerk van vandaag. In de toekomst wordt er nog verder ingezet op het steeds efficiënter maken van het netwerk, meer automatisering, grotere schaalbaarheid en robuustheid van het netwerk. Optimale (gedistribueerde) routing in het Internet en schaalbare virtualisatie zijn twee belangrijke middelen om dit mogelijk te maken. Deze vormen dan ook de speerpunten van het onderzoek van deze thesis naar het toekomstige Internet.

Schaalbare en dynamische routing worden bemoeilijkt door het steeds toenemende internetverkeer en de exponentiele groei van het netwerk. Er worden dan ook grote onderzoeksinspanningen gedaan om te kunnen voldoen aan deze hoge verwachten. Desalniettemin hebben vele van deze inspanningen geleid tot teleurstellende, incrementele oplossingen en voornamelijk opsmukwerk. Als tegenreactie, werden recentelijk een aantal volledige vernieuwende aanpakken voorgesteld. Geometrische routing is een van deze voorstellen die zich toespitst op het schaalbaar maken van de routing. Hierbij worden coördinaten toegekend aan netwerkknopen in een geometrische ruimte. Bij gulzige routering, gaat een netwerkknoop

dat de cordinaten van zijn buren weet, netwerkverkeer doorsturen naar de dichtstbijzijnde buur. Schaalbare en dynamische routing worden bemoeilijkt door het steeds toenemende internetverkeer en de exponentiele groei van het netwerk. Er worden dan ook grote onderzoeksinspanningen gedaan om te kunnen voldoen aan deze hoge verwachten. Desalniettemin hebben vele van deze inspanningen geleid tot teleurstellende, incrementele oplossingen en voornamelijk opsmukwerk. Als tegenreactie, werden recentelijk een aantal volledige vernieuwende aanpakken voorgesteld. Geometrische routing is een van deze voorstellen die zich toespitst op het schaalbaar maken van de routing. Hierbij worden coördinaten toegekend aan netwerkknopen in een geometrische ruimte. Bij gulzige routering, gaat een netwerkknoop dat de coördinaten van zijn buren weet, netwerkverkeer doorsturen naar de dichtstbijzijnde buur.

In het eerste deel van deze dissertatie wordt de complexe netwerkstructuur van het Internet onderzocht, en wordt geometrische routering geëvalueerd in de context van het inter-domein Internet. Omwille van de grootschalige en dynamische aard van het Internet, zijn schaalbare en dynamische routingmechanismen broodnodig. Daarom heeft dit onderzoek getracht om schaalbare en foutbestendige geometrische routing te onderzoeken voor inter-domein routebepaling.

In dit onderzoek wordt foutbestendigheid tegen enkel- en meervoudige fouten van netwerklinks onderzocht bij geometrische routering. Er worden verschillende herstelmethoden voorgesteld die zowel als protectiemechanisme of als restoratiemechanisme kunnen ingezet worden. Bij protectie worden de herstelpaden berekend en opgezet alvorens de fout optreedt. Bij restoratie worden de herstelpaden pas opgezet nadat de fout is gedetecteerd. Beide technieken hebben voor- en nadelen. Waar protectietechnieken heel snel herstel toelaten, is de kost qua herstelcapaciteit, die op voorhand gereserveerd moet worden, heel hoog. Anderzijds bieden restoratietechnieken meer flexibiliteit in het omgaan met netwerkfouten met als nadeel een trager herstel. Enkelvoudige fouten worden in ons onderzoek afgehandeld door in het netwerk eerst in de breedte door te zoeken, zoekend naar tussenliggende knopen welke een gulzig pad naar de eindbestemming toelaten. De geschikte tussenliggende knoop wordt bevraagd voor z'n coördinaat. Deze knoop wordt vervolgens gebruikt in het verder gulzig gerouteerde pad om de foutieve link te omzeilen. Om met meervoudige fouten van netwerklinks om te gaan, worden disjuncte netwerkpaden gezocht voor elke individuele link. Aan de hand van experimenten, werd de schaalbaarheid en hun beperkte infrastructuurbenodigdheden van de voorgestelde mechanismen gevalideerd. De mechanismen schalen met het aantal links uit de opspannende boom van het netwerk en de geheugenvereisten in elke knoop zijn proportioneel met de graad van de boom. Bovendien is de stretch (i.e., de deviatie t.o.v. het kortste pad) beter dan die van mogelijke alternatieve routingtechnieken.

In de eerste studies steunen we op een bestaande methode om coördinaten toe te kennen aan netwerkknopen in het hyperbolisch vlak. Omdat deze methode behoorlijk complex is, stellen we een eenvoudige maar alternatieve aanpak voor welke steunt op een opspannende boom van het netwerk. In deze aanpak bepaalt de locatie van de knoop in de boom de coördinaat. We stellen vervolgens

een gedistribueerd algoritme voor om deze locatie te berekenen. Steunend op de voorgestelde protectiemechanismen, worden voorberekende herstelpaden gebruikt wanneer een foutieve netwerklink wordt gedetecteerd. Ondanks het snelle foutherstel dat dit toelaat, wordt kwaliteit van de route in de zin van stretch hierdoor negatief beïnvloed. Om hieraan tegemoet te komen en om te kunnen omgaan met verschillende vormen van netwerkdynamiek (bv., nieuwe knoop/link, of verwijdering van knoop/link), wordt het algoritme uitgebreid met een mechanisme om de boom opnieuw te berekenen bij netwerkveranderingen. Op deze manier past de routing zich aan aan de nieuwe toestand van het netwerk. Dit heeft tot gevolg dat de toename in stretch beperkt wordt.

Ondanks de verbeterde schaalbaarheid van geometrische routing, brengt het ook nieuwe uitdagingen qua herstel van netwerkfouten. Dynamiek in het netwerk kan de opspannende boom veranderen die gebruikt wordt voor de coördinaten, met globale veranderingen tot mogelijks gevolg. Hoe dichter de fout zich bij de wortel van de boom bevindt, hoe meer knopen beïnvloed worden en een nieuwe coördinaatberekening vereisen. Daartegenover is de invloed van netwerkfouten bij de bladeren van de boom beperkt tot lokale wijzingen. Zelfs bij netwerkfouten waarbij veel knopen betrokken zijn, blijft de hersteltijd vrij laag. Een prototype van het voorgestelde algoritme werd geïmplementeerd en werd succesvol gedemonstreerd op het IETF 90 Bits-N-Bites evenement.

Naast de studie van het foutherstel van geometrische routing, is er ook onderzoek uitgevoerd naar de schaalbaarheid en efficiëntie van geometrische routing. Dit werd onderzocht door geometrische routing toe te passen op grootschalige netwerken met een gelijkaardige structuur als het Internet. De resultaten werden vergeleken met technieken die steunen op coördinaten in het hyperbolisch vlak. De efficiëntie en schaalbaarheid van de voorgestelde aanpak karakteriseren zich door hun lage stretch, een beperkte nood aan netwerkvoorzieningen, en weinig geheugenvereisten in de voorstelling van de coördinaten.

In het vervolg van de thesis, wordt er op basis van de gedane experimenten en analytische studies, een overzicht van de bestaande routingtechnieken gegeven. Dit overzicht geeft een evenwichtiger beeld over de toepasbaarheid van geometrische boetebepaling in de context van grootschalige inter-domeinnetwerken. Hierin wordt duidelijk dat de geheugen- en schalingskarakteristieken van de voorgestelde geometrische routing moeten afgewogen worden tegen het convergentiegedrag en routekwaliteit bij foutherstel. Een andere waargenomen fenomeen is dat geometrische routebepaling en andere radicaal nieuwe aanpakken vaak de performantie van de routing sterk verbeteren ten koste van minder functionaliteit. In het geval van geometrische routing bijvoorbeeld, wordt de geheugenschaalbaarheid verbeterd maar zijn routeringsvoorkeuren (‘policies’, zoals die mogelijk zijn in de huidige internetrouting) niet toegelaten.

Op basis van de gevonden open uitdagingen, geven we enkele mogelijke richtlijnen voor toekomstig onderzoek. Deze mogelijke pistes omvatten: i) ondersteuning voor routevoorkeuren bij routing in het kader van geometrische routering, ii) het voorzien van een afbeeldingssysteem tussen de identiteiten en coördinaten van knopen, iii) de studie van beveiligingsaspecten en iv) de omschakeling van

bestaande routingsystemen naar geometrische routing.

Als tweede onderdeel van het toekomstige Internet, onderzoekt deze thesis de bevoorrading van infrastructuur voor services in gevirtualizeerde telecomnetwerken. In de gangbare werkwijze worden telecomservices gecreëerd aan de hand van netwerkfuncties geïmplementeerd in hardware die enkel en alleen geschikt en gereserveerd zijn voor deze services. Voorbeelden van netwerkfuncties zijn bijvoorbeeld een firewall, een netwerkadresvertaler (NAT) of een functie om binnendringers in het netwerk te detecteren. Wanneer men gebruik maakt van virtualisatietechnieken, kan benodigde functionaliteiten uitgevoerd worden als software op generieke server-hardware die zich eender waar in het netwerk kan bevinden. Dit concept wordt de virtualisatie van netwerkfuncties (Network Function Virtualization) genoemd. Dit biedt nieuwe opportuniteiten voor telecomoperatoren, omdat nieuwe services snel en efficiënt kunnen opgestart worden aan lage kost. Aan de andere kant brengt dit ook nieuwe uitdagingen met zich mee, omdat nu moet bepaald worden waar bepaalde netwerkfuncties zullen uitgevoerd worden (op welke servers, gebruikmakend van welke netwerkonderdelen). Dit laatste wordt bemoeilijkt doordat enkelvoudige complexe netwerkfuncties op hun beurt weer kunnen opgesplitst worden, op verschillende wijzen, in meerdere eenvoudigere netwerkfuncties waarvan hun plaats ook weer in het netwerk moet bepaald worden. Een 'parental control' netwerkfunctie, die er bijvoorbeeld voor moet zorgen dat jonge kinderen geen 18+ internettoepassingen of websites kunnen bezichtigen, kan bijvoorbeeld n drie eenvoudiger functies worden opgesplitst: i) een netwerkverkeersclassificatie functie, ii) een web proxy, en iii) een firewall. Dit proces wordt servicedecompositie genoemd.

In dit onderzoek brengen we servicedecompositie in rekening tijdens het plaatsingsproces van netwerkfuncties en onderzoeken de gecombineerde optimalisatie van servicedecomposite en plaatsing. Twee nieuwe technieken worden voorgesteld. De eerste spitst zich toe op een Integer Linear Programming (ILP) - optimalisatie. De andere techniek biedt een heuristiek die schaalbaarder is dan de ILP-gebaseerde oplossing. De heuristiek bevat twee fasen: i) selectie van decompositie, ii) backtracking-gebaseerde plaatsing. Deze voorstellen beogen de minimalisatie van de gebruikte netwerkinfrastructuur. Een geoptimaliseerde, fijnmazige decompositie in combinatie met een efficiënte plaatsing wordt door de voorgestelde mechanismen mogelijk gemaakt. De efficiëntie van de technieken wordt gekarakteriseerd door een hoge acceptatiegraad (de verhouding van het aantal services die succesvol geplaatst worden tot het totale aantal services) en een lage plaatsingskost.

In realistische telecomnetwerkscenario's loopt het aantal netwerkcomponenten dat moet behandeld worden al snel op tot tienduizenden. Bij dergelijk hoog cijfer, tot duizenden services per dag en meerdere realisatie-opties per services, is een schaalbare controlearchitectuur nodig om deze services te voorzien. Aan de hand van een 'proof of concept'-prototype worden de belangrijkste tijdscomponenten in de architectuur geïdentificeerd. Deze blokken zijn gerelateerd aan: i) het opvragen van alle decomposities van een service uit een database, ii) selectie van de geschikte decompositie, iii) plaatsing van de geselecteerde decompositie. Op ba-

sis van deze basiscomponenten worden verschillende architecturale verbeteringen voorgesteld om een schaalbaarder raamwerk voor servicevoorziening te bekomen. Het plaatsingsalgoritme kan bijvoorbeeld aangepast worden naar een gedistribueerd of hiërarchisch mechanisme, om het raamwerk schaalbaarder te maken. Deze voorstellen vormen mogelijke toekomstige pistes voor verder onderzoek.

Summary

The Internet is the fundamental communication medium, connecting people, businesses and organizations. More than 190 countries are interconnected via the Internet to exchange data, news and opinions. Every aspect of human lives, from social to economical characteristics, is impacted by this communication network. Every day, we receive news via the Internet and share different information and files through cloud. Additionally, most enterprises want to stream on the Internet (e.g., radio and television broadcast, sport/live events with global audience and multimedia conferencing). As a result, high demands in terms of quality and capacity are imposed. However, the Internet was not initially designed for such high demands.

The idea of the Internet started with the requirement to connect the top universities of the United States. This was to share all the research data without wasting too much time and to share computing resources. Therefore, ARPANET was created as the first wide area packet switching network, connecting four nodes at: University of California, Los Angeles (UCLA), Augmentation Research Center at Stanford Research Institute, University of California, Santa Barbara (UCSB) and University of Utah. After this attempt succeeded in 1969, experts realized how much potential this interconnection medium can have. Therefore researchers started to extend this medium, leading to the extensive heterogeneous interconnected networks of these days. The main evolutions in the Internet involve increased efficiency, automated control, scaling and resiliency. Improved Internet routing and scalable virtualization of Internet services enable such evolutions. These are two aspects of the future Internet which are investigated in this thesis.

With the ever-increasing bandwidth demands and the exponential growth of the network, providing scalable and dynamic routing in the Internet becomes significantly challenging. Prominent research efforts have been made to find solutions to meet the high expectation of today's demands. However, these efforts mainly led to incremental and patch solutions which were proved to be insufficient. As a result, new clean-slate approaches have been emerging in the recent years. Geometric routing is one of these approaches, which intends to solve the scalability limitation of routing in the Internet. In this routing network nodes are assigned coordinates in a metric space. Knowing the coordinates of the neighbors, a node forwards the incoming traffic to the neighbor which decreases the distance towards the intended destination. This is referred to as greedy forwarding.

In this dissertation, first the complex network structure of the Internet is examined and the applicability of geometric routing to the inter-domain setting in the

Internet is investigated. Since the Internet is large and dynamic in nature, scalability and dynamicity/adaptivity of the corresponding routing mechanisms are indispensable. Therefore, in evolving towards a future Internet, this research aims at providing a scalable resilient geometric routing scheme for inter-domain routing.

In this research, resiliency against single and multiple link failures in geometric routing is investigated. Different recovery methods are proposed which can be used as both protection and restoration mechanisms. In protection, the recovery paths are calculated and signaled pro-actively before any failure in the network. In restoration, recovery paths are established after the detection of a failure. Both techniques have advantages and negative points. While protection techniques enable fast failure recovery, the backup capacity, which should be reserved in advance, is quite high. On the other hand, restoration techniques provide more flexibility in dealing with network failures at the cost of slower failure recovery. Single failures are handled by applying a breadth-first search, looking for intermediate nodes which enable greedy forwarding towards the destination. The suitable intermediate node is queried for its coordinate. This is then used by the packets to bypass the failing link. In order to deal with multiple failures, disjoint backup paths are found for each link individually. Through experiments, the scalability of the proposed mechanisms and their limited resource requirements are confirmed. The schemes scale with the number of edges in the spanning tree of the network and the memory requirement in each node is proportional to its degree. Moreover, the performance of the routing in terms of stretch (i.e. deviation from the shortest path length) is improved compared to the existing alternatives.

In the first studies, we rely on an existing method to assign coordinates to network nodes in the hyperbolic plane. Since this method is quite complex, we propose a simple but efficient approach which relies on a spanning tree of the network. In this scheme, the location of the nodes in the tree determines their coordinate. We then propose a distributed algorithm to calculate these coordinates. Relying on the proposed protection mechanisms, pre-calculated backup paths are used upon failure occurrence. Although these mechanisms lead to a very fast failure recovery, the routing quality in terms of stretch is negatively impacted. To address this issue and in order to cope with different types of network dynamics, including node/link addition/removal, the algorithm is extended with a mechanism which triggers tree re-calculation upon network changes. This way the routing adapts to the new state of the network. As a result, the performance loss in routing stretch diminishes.

While geometric routing improves the scalability in terms of memory requirement in the routers, it introduces new challenges in the recovery domain. Network dynamics/failures, impacting the spanning tree used for coordinate calculation, may lead to global changes in the network. This is significantly dependent on the location of the failure. The closer the failure to the root of the tree, the more nodes are impacted and require coordinate re-calculation. However, tree leaves failure result in very limited local changes. In spite of the large number of affected nodes, the recovery time is quite low. A prototype implementation of the proposed algorithm is provided which has been successfully demonstrated in the Internet Engineering Task Force (IETF) 90 Bits-N-Bites event.

In addition to resiliency, the research investigates the scalability and efficiency aspects of geometric routing. This is examined by applying the proposed geometric routing scheme on very large-scale network topologies with characteristics similar to the Internet. The outcomes are compared with another scheme which relies on coordinates in the hyperbolic plane. The efficiency and scalability of the scheme is indicated by the low routing stretch, the very limited required resources for implementing the network component and the low memory requirement for coordinate presentation.

As previous parts relate to the Internet routing, based on the experiments and analytical studies performed so far, we give an overview of the existing routing schemes (both traditional and clean-slate approaches). This overview enables a more careful conclusion regarding applicability of geometric routing to large-scale inter-domain settings. While the documented research illustrates the memory-advantage/scalability of the proposed geometric routing scheme, it indicates new trade-offs between the convergence trend and routing quality in the recovery domain. Another main outcome is that geometric routing and most of the clean-slate approaches improve performance at the cost of decreasing functionality. For example in geometric routing, the memory scalability is improved but routing policy, as in the current Internet routing, is not supported.

Based on the discovered open challenges, we provide a guideline for future research directions. These future directions include: i) support for routing policy in geometric routing, ii) providing a mapping system to bind nodes identifier to coordinates, iii) security concerns and iv) transition to geometric routing.

As a second aspect of future Internet, this thesis investigates the problem of service provisioning in virtualized telecom networks. Traditionally, a service composed of several network functions is implemented by middleboxes and dedicated hardware. Examples of network functions include firewall, network address translator and intrusion detection system. Exploiting network virtualization, these network functions can now be implemented in software and be deployed anywhere in the network on general purpose hardware. This is referred to as Network Function Virtualization (NFV). NFV has provided new opportunities for telecom operators, as novel services can be deployed rapidly with a very low cost. However, new challenges are also introduced on where to place the virtualized network functions of a service. The latter may become quite complex since a large monolithic network function may be decomposed into smaller network function blocks or it can be realized in multiple ways with different implementations. For instance, a 'parental control' network function can be decomposed to 3 less complex network functions: i) traffic classifier, ii) web proxy and iii) firewall. This process is referred to as service decomposition.

In this research, we take service decomposition into account at the time of service placement/embedding and thus, joint optimization of service decomposition and embedding is investigated. Two novel approaches are proposed. One finds the optimal solution relying on Integer Linear Programming (ILP) formulation. The other one provides a heuristic approach to solve scalability limitation of ILP. The latter is composed of two phases: i) decomposition selection and ii) backtracking-

based mapping/embedding. These approaches target minimizing the mapping cost by reducing the consumed resources. Relying on the proposed schemes, an optimized fine-granular decomposition of network functions in a service and related embedding in the network are achieved. The efficiency of the schemes is identified by high acceptance ratio (the ratio between the number of services which are successfully mapped to the network and the total number of services) and low embedding cost.

In realistic telecom network scenarios, the number of network components which should be considered in the embedding problem simply goes beyond ten thousands. Having such a large number of resource elements, order of thousands of services on a daily basis and multiple realization options for each service, a scalable architecture for service provisioning is essential. Relying on a proof of concept prototype which implements the proposed embedding algorithm, we identify the major time consuming blocks. These blocks are related to: i) retrieving all decompositions for a service from a database, ii) selecting a suitable decomposition and iii) embedding the selected decomposition. Based on these blocks, several architectural enhancements are proposed to achieve a more scalable service provisioning framework. Changing the embedding algorithm to a distributed or hierarchical approach are two solutions which can provide a more scalable framework. These are interesting research directions which require further investigation.

1 Introduction

The Internet is a massive dynamic collection of interconnected heterogeneous networks. It is used as the fundamental medium to connect different interested parties enabling them to communicate. Although the current Internet works and successfully meets its responsibilities, it was not initially designed for the high demands of these days. Consequently, it is being more and more challenged by the exponential expansion of the network, the ever-increasing communication demands with high expectation in terms of performance and capacity. Therefore, Internet has been the subject of research for decades to fulfill the ever-increasing future demands. The main evolutions in the Internet are related to resiliency, scaling and automated control. An improved Internet routing and scalable virtualization of Internet services enable such evolutions and increase efficiency for the future Internet. These are two aspects which are investigated in this PhD research.

First, we focus on future network architectures and novel routing algorithms[1]. We perform research on how they can be adapted to satisfy the future communication requirements. Particularly we aim at scalable routing algorithms with recovery techniques to provide resiliency against failures in the network.

Next, we focus on service provisioning in virtualized telecom networks. The rise of network virtualization has introduced new opportunities for telecom operators. It enables rapid and dynamic service provisioning at a low cost. Services are conventionally provided through dedicated hardware and middleboxes. Exploiting different virtualization techniques, these services can be deployed on general

[1] Routing refers to the process of finding a path from one point in the network to another.

purpose hardware in different parts of the network. This reduces the cost and complexity of network design, while increasing flexibility and scalability of service provisioning. In this doctoral dissertation, we explore service orchestration[2] in virtualized telecom networks. Scalable service orchestration frameworks are investigated and optimized placement algorithms for virtualized network functions composing a service are designed.

This chapter presents the required background concepts related to the research topics explored in this thesis (Section 1.1). It indicates the major challenges corresponding to each topic (Section 1.2 and Section 1.3) and provides an overview of the performed research contributions (Section 1.4). The chapter is concluded with the list of publications which resulted from this PhD Research (Section 1.5).

1.1 Background on communication network

This section presents the background to the concepts which are important for designing network architecture and routing algorithms for future Internet. First we present an overview of the current Internet infrastructure. The network layering model is described next and then the addressing schemes and routing protocols used in the Internet are detailed. We then present a formal terminology regarding recovery and introduce reliability performance concepts.

1.1.1 Internet network infrastructure

The Internet network infrastructure can be divided into several network segments as depicted in Fig. 1.1. These segments are as follows:

- Home/enterprise networks
  Home (or small enterprise) networks are networks of a limited number of end-users, personal computers or other devices which are connected via a wired or a wireless Local Area Network (LAN). The speed of these small-scale networks can be up to 100 Gb/s [1] while a higher speed is expected to be achieved in the future.

- Access networks
  These networks connect up to thousand(s) of home/enterprise networks to the Internet. They typically have a tree-like structure or sometimes a ring structure when redundancy is required (e.g. for business users). The scale of these networks is quite larger than home networks, as they usually span a village or city. The typical speed that can be achieved in these networks is between 100 Mb/s and 1 Gb/s per end user.

[2] Service orchestration refers to the process of deploying a set of virtual or physical network functions composing a service.

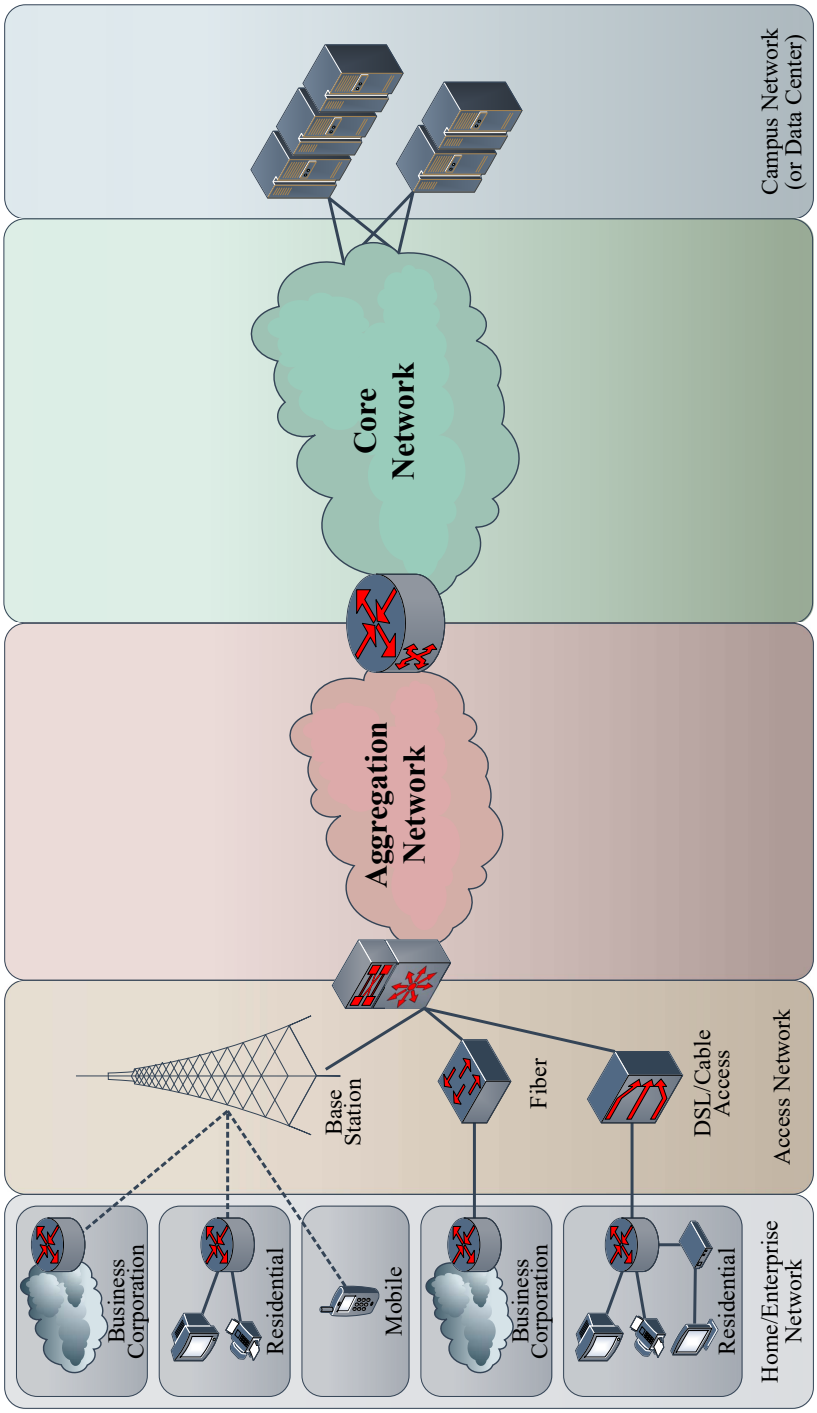


Figure 1.1: Overview of Internet infrastructure segments

- Aggregation or metro networks
  These networks connect cities or larger areas within a city in a slightly meshed network and normally span up to 50 km. They typically rely on a ring or star structure to interconnect access networks. Their role is to aggregate the traffic of access networks towards the core network. The speed in these networks is up to 10 Gb/s.

- Core or backbone network
  The aggregation networks are interconnected to form the core of the Internet. The core network creates a large meshed topology of around 60 K Autonomous Systems (AS), which connects countries and continents at far distance (i.e., hundreds of kilometers). The ASes form the main structure of the Internet. Each AS is controlled by a common administrator (or group of administrators) on behalf of a single party (e.g., a university or a business enterprise). ASes are also known as routing domains. The size of each AS can range between ten to thousands of nodes. The high speeds of 100 Gb/s is achieved in these networks [2]. The focus of this research is mainly on the core network, related challenges and advances.

- Campus networks/ data centers
  Large campus networks are generally managed by some universities or enterprises which are composed of up to thousands of nodes [3]. Both campus networks and data centers [4] are connected directly to the core network or aggregation networks.

1.1.2 Network layering and technology overview

In a communication network generally a layered model is used. This allows to describe how the different software and hardware components, involved in the network, should divide the work and interact with each other. In this model, each layer provides services to the higher layer. There exist two models for describing the Internet structure: i) Open Systems Interconnection (OSI) and ii) Transmission Control Protocol/Internet Protocol (TCP/IP). OSI is a reference model created by the International Organization for Standardization (ISO) to develop networking protocol standards. TCP/IP was developed earlier as the outcome of research and development conducted by the Defense Advanced Research Projects Agency (DARPA). There is not a 1-to-1 mapping between the layers of the two models. However, there is no conflict between them either (see Fig. 1.2). As depicted in Fig. 1.2, the presentation and session layers of the OSI model are absent in the TCP/IP model. The functionality of these two layers are mainly performed by the application layer of the TCP/IP model. Additionally the functionality of the data link and physical layers of OSI model are merged into one layer i.e., host-to-

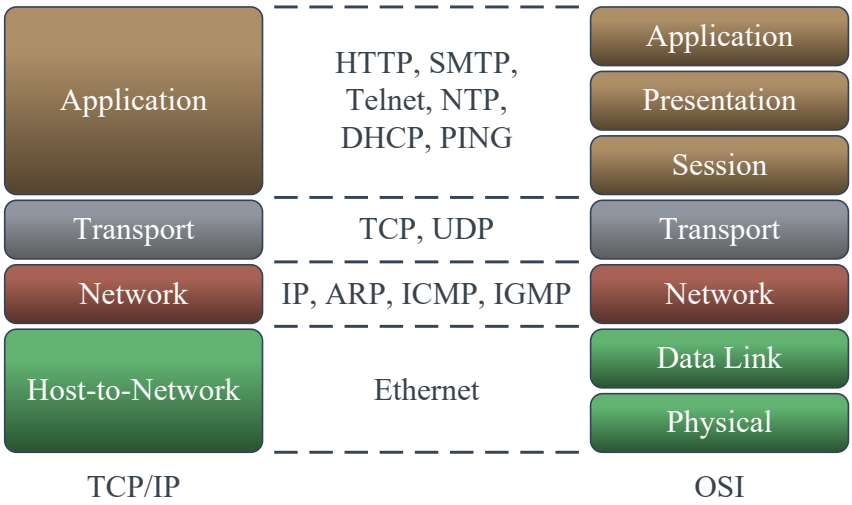


Figure 1.2: Layered models of the Internet and example protocols

network layer in TCP/IP. Below, we explain the main layers of these models with their related technologies.

- The application layer defines different services which are required to run the users applications. Examples include Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP) protocols which allow web-browsing and e-mail.

- The transport layer is responsible for reliable end-to-end communication between the end-hosts. The two known protocols for this layer are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP provides a reliable connection-oriented service with congestion control while UDP services are connection-less without congestion control and guaranteed delivery. In connection-oriented networks, first a connection is established between the source and the destination node. This can be done by manual configuration or through a handshaking process. All the data which goes through this connection is treated the same way. In contrast, connection-less networks do not create any connection in advance. Each intermediate node handles the receiving data based on local decisions. In this case, the source does not monitor the data delivery to the destination.

- The network layer includes the Internet Protocol (IP) [5]. The main responsibility of this layer is to: i) provide connection-less connectivity between end-hosts, ii) provide addressing of the nodes in the network and iii) find a

path between source and destination nodes (routing) and forward messages efficiently.

- The data link layer ensures reliable connectivity (enable error control) between a limited number of network nodes within a same segment. Example protocols for this layer are Ethernet IEEE 802.2 framing and Point to Point Protocol (PPP) framing.

- Finally the physical transmission of the data over a given medium such as twisted pair cable, coaxial cable or optical fiber is managed by the physical layer.

Example protocols of each layer are depicted in Fig. 1.2. The focus of this research is mainly on the network layer and related protocols. Therefore in the following subsection, IP [5] as the fundamental protocol in this layer is detailed.

Data plane vs. control plane. These planes of operation are the building blocks of today's network layered architecture. The data plane is responsible for transporting the end-user data in the network. All the functionality of packet header parsing, packet queuing, scheduling, filtering, encapsulations and forwarding are abstracted into the data or forwarding plane. In addition to the end-user data, control-related data should be exchanged in the network. Functionality such as system configuration, exchange of routing table information and creating forwarding tables for the data plane is handled by the control plane.

1.1.2.1 IP routing

Routing is the process of finding paths between connected networks and is the primary function of the Internet Protocol (IP)[3]. In IP-based networks the end-users data traffic is split into packets which are called IP datagrams. Each IP datagram contains a source and a destination IP address, which indicate the address of sending host and receiving host, respectively. These addresses are used to route the data through the network. In IP-based networks, different network segments are interconnected by IP routers. These routers process incoming datagrams and pass them from one network segment to another based on the destination IP address (see Fig. 1.3). The connectivity achieved by IP-based networks is known to be one of the most scalable network layer technologies available today. It is used to interconnect enterprise networks, and also the Internet backbone.

Addressing and forwarding. IP-based networks rely on IP addresses to route datagrams across the network. The functionality of an IP address is twofold. It is used for both identification and location of network nodes. In the IPv4 [5] addressing scheme, each interface in the network which interconnects end users to

[3] IPv4 [5] or IPv6 [6]

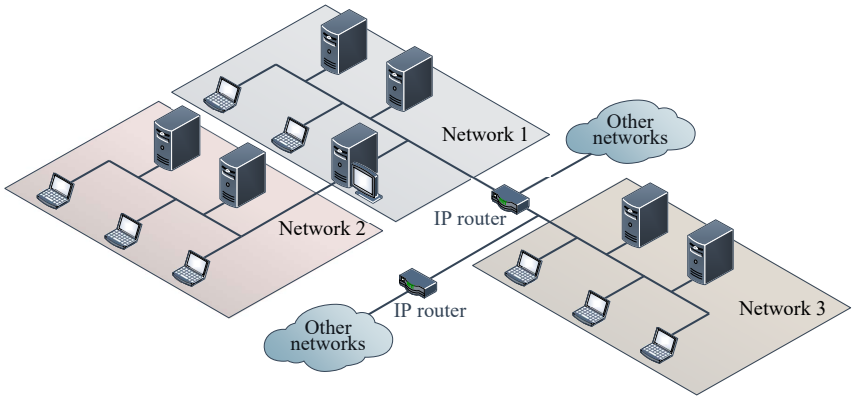


Figure 1.3: Example of an IP network. Different network segments are interconnected by IP routers.

routers is assigned a 32-bit binary identifier. To represent these addresses, typically a dot-notation is used. It splits the address into four numbers between 0 and 255, each separated by a dot (e.g., 192.168.56.2).

The total number of IPv4 addresses ranges between 0.0.0.0 and 255.255.255.255 which leads to $2^{8\times4}$ possible addresses. While this seems a large number, it is no longer enough to address all the devices connected to the Internet. Therefore another type of address format, i.e., IPv6, is proposed to solve this limitation. IPv6 address format is quite different from IPv4. It consists of 128 bits instead of 32. It is represented by 8 sets of 4 hexadecimal digits which are separated by colons (e.g., 2001:0000:9d38:6ab8:2047:1a74:53ed:977b). It is assumed that IPv6 addresses will be sufficient for a long time in the future. In this thesis, we mainly focus on IPv4.

There are two schemes in IPv4 addressing: i) classful and ii) classless addressing. Originally all IP addresses were classful. This means that they belonged to Class A, B, C, D or E. Class D is used for multicast and class E contains the reserved addresses mainly for experimental and future use. IP addresses can be clustered into subnets. This is indicated by the network (prefix) part of the address which is a fixed number of leftmost bits. The rest of the bits in the address identify the host (or node) within that subnet. In classful addressing, the network size and the host size are fixed. These fixed boundaries greatly limit the flexibility and number of addresses that can be assigned. As a result, Classless Inter-Domain Routing (CIDR) [7] was introduced. In CIDR the prefix can have any arbitrary length and it is denoted with a slash (/) followed by the length in bits. For instance, 192.168.56.2/24 means that the leftmost 24 bits are used to indicate the prefix and the remaining 8 bits are used for host identification.

In IP routers, incoming datagrams are forwarded based on their destination IP address. To enable this each router stores a forwarding table,[4] indicating where to send each packet. The concept of prefixes in the IP addresses enables routers to reduce the number of forwarding entries. This is possible because IP addresses are assigned in continuous blocks and a router can typically reach a whole subnet via a single interface. Therefore, instead of storing an individual IP address per destination in the forwarding table, it is enough to store only the corresponding prefix to reach the whole network segment covered by that prefix. In this case, forwarding is based on prefix matching of the destination address's IP prefix in the incoming packet and forwarding table entries. In practice it is possible that in the Internet the prefixes overlap (because CIDR is used everywhere in the network). Therefore for an incoming packet, the destination address's IP prefix may match multiple prefix entries in the Forwarding Information Base (FIB). For example, consider the forwarding table in Fig. 1.4. In this table, IP addresses from 192.24.12.0 to 192.24.15.255 overlap which means that they are matched with both prefixes in the table.

Prefix	Next hop
192.24.0.0/18	D
192.24.12.0/22	B

Figure 1.4: Example of IP forwarding table

In order to forward packets efficiently in the above situation, the routers use Longest Prefix Match (LPM) algorithm. This algorithm finds the entry in table which has the longest prefix, matching with the destination IP address of the incoming packet. The packet is then forwarded to the corresponding next hop. In this example, all the packets in the overlapping range (i.e., 192.24.12.0 to 192.24.15.255) are forwarded to the next hop B, since the entry 192.24.12.0/22 has a longer prefix match (22 bits). Although LPM enables efficient handling of the above situation, the existence of such additional routes increases the number of entries in the router's forwarding table. There are several reasons for the existence of these longer prefixes in the network. An organization may advertise more-specific routes (i.e., longer prefixes) in addition to less-specific ones for the sake of traffic engineering or other specific routing policy. Besides, large organizations may be given large blocks of IP prefixes. The organizations may split the large prefix block into smaller ranges to assign to their different sites around the world. As a result, there is no relation/pattern between the assigned IP prefixes and their location in the network topology. This is referred to as route disaggregation or de-aggregation. The selective advertisement of a few more-specific routes may

[4] Also known as Forwarding Information Base (FIB)

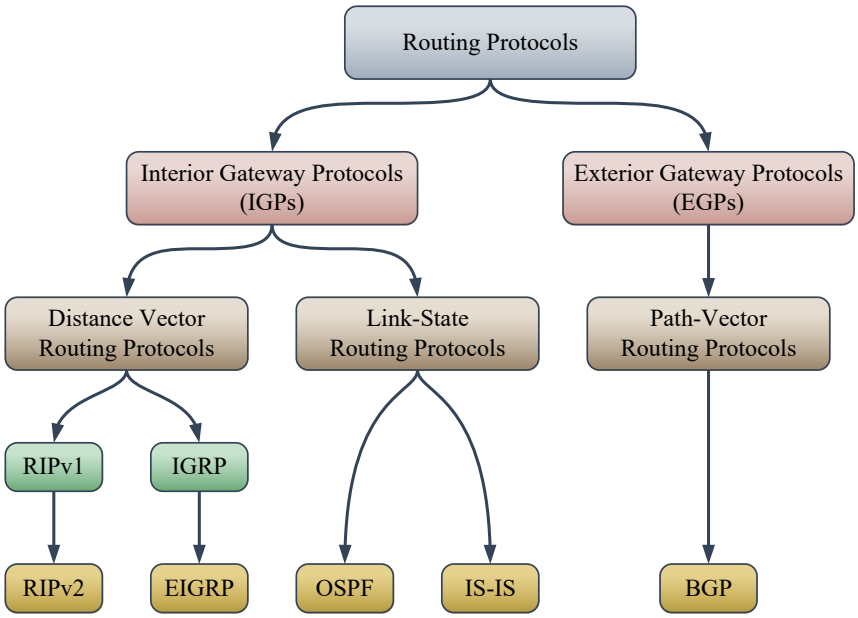


Figure 1.5: Hierarchical classification of routing protocols

be required for some network architectures and can be tolerated. However, excessive disaggregation by organizations is discouraged as it leads to serious scalability issues in the forwarding tables of the backbone routers.

IP routing protocols. IP routing protocols enable routers to build up their forwarding tables that relate final destinations with next hop addresses. These protocols enable exchanging topology or routing information with immediate neighbors and over the network. This is used by the routers to derive paths or next hops towards destinations in different segments of the network. Routing protocols can be classified into several groups depending on their characteristics and scope. Fig. 1.5 illustrates a hierarchical view of routing protocols classification with example protocols.

Due to the large scale of the Internet, routing protocols are only responsible for certain parts of the network. Based on the scope of the protocols they are categorized as: i) Interior Gateway Protocols (IGPs) and ii) Exterior Gateway Protocols (EGPs). IGP is used for routing within a single domain (intra-AS routing) while EGP is used for routing between ASes (inter-AS routing). After this categorization, routing protocols can be classified based on their operation. There are three major classes which are widely used in IP networks.

- Link-state routing protocols

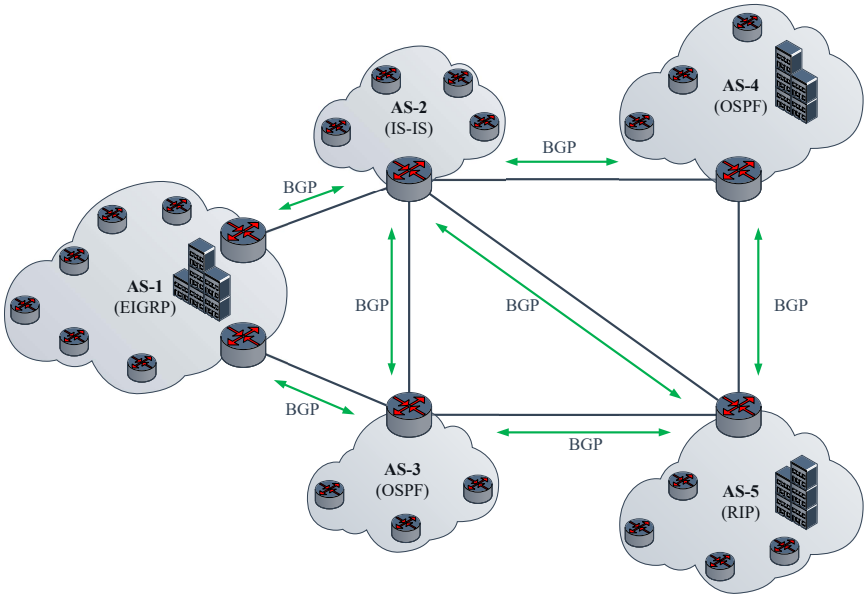


Figure 1.6: Inter-AS and intra-AS routing

In these protocols, each router distributes information about the router, its directly connected links and their state to all the routers in the network. Routers receiving this information can obtain the full picture of network topology. Therefore, each router can calculate the best path to the destination. These protocols rely on a Shortest Path First algorithm such as Dijkstra or Bellman-Ford. In case of a change in a link's state, a routing update referred to as Link-State Advertisement (LSA) is exchanged between routers. Upon arrival of a LSA, the router re-calculates the shortest path to the destinations which are affected. Example protocols include Open-shortest Path First (OSPF [8]) and Intermediate System to Intermediate System (IS-IS [9]).

- Distance-vector routing protocols
  In contrast to link-state routing protocols, a router running a distance-vector routing protocol does not have a full view of the network topology. These protocols base their decisions on two characteristics, distance and vector. Distance indicates how far the destination network is and the metric can be hop count, cost, bandwidth, delay, etc. Vector identifies the direction of the next-hop to reach the destination. In these protocols each router sends its entire routing table to its directly connected neighbors. Based on this received information, the route with the minimum distance to a given destination net-

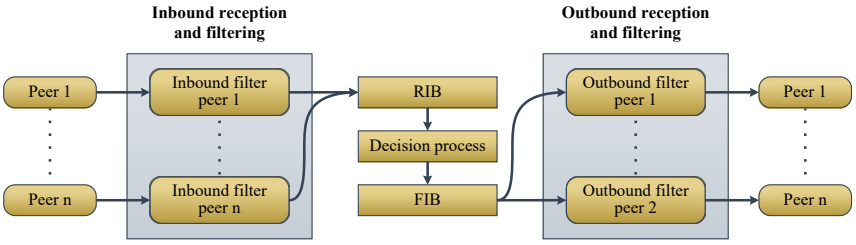


Figure 1.7: Routing information process in BGP routing engine

work is selected. Routing Information Protocol (RIP [10], [11], [12]), Interior Gateway Routing Protocol (IGRP [13]) and Enhanced Interior Gateway Routing Protocol (EIGRP [14]) are protocols in this category.

- Path-vector routing protocols
  These protocols can be considered as the extension of distance-vector routing protocols. Instead of distributing only distance metric per destination network, the entire path to the destination is advertised. This way each router can simply detect a loop and avoid long convergences. Similar to distance-vector protocols, the routers do not have the full view of the network. Additionally these protocols enable routers to apply their local policies in selection of the paths and advertising them. Border Gateway Protocol (BGP) [15] is a routing protocol in this category.

As depicted in Fig. 1.5 both link-state and distance-vector routing protocols are considered as IGPs while path-vector routing is an EGP. Fig. 1.6 illustrates a simple scenario highlighting the deployment of IGPs and EGPs. Note that there is only one protocol available for EGP which is BGP [15]. The focus of this thesis is on inter-domain routing in the core of the Internet. Therefore, we explain BGP in more detail.

Border Gateway Protocol (BGP). Each AS owns one or more border gateways that are connected in a peering relationship with other gateways. BGP is the protocol driving the routing between these routers. In order to discover neighboring ASes, a router running BGP receives routing information (aka BGP updates) from its neighbors. Update messages include Autonomous System Numbers (ASNs) and network prefixes reachable through those ASes. These updates are filtered in the router (e.g., to avoid loops) and stored in the Routing Information Base (RIB). RIB can be considered as the BGP update cache. The latter means that the routing information is stored but not necessarily used. As a result, it is possible that RIB contains multiple paths towards a destination. Next, the router should decide on the best route. This is performed by the BGP decision process. The selected route is added to the routers forwarding table (FIB). The entries in

FIB are used for actual forwarding of the incoming traffic. The BGP decision process enables applying a local policy in each router by giving different priorities to different paths, received for a single destination, and selecting the most preferred one. Similarly a router can decide which routes to advertise to other neighboring routers. Fig. 1.7 provides the explained routing information process flow in a BGP routing engine.

Being the only routing protocol used for inter-AS routing, BGP faces several issues including the scalability of backbone routing tables, path instability and slow convergence [16]. These challenges are further detailed in Section 1.2.

1.1.3 Network recovery

Communication networks can be affected by a wide range of unintentional failures such as natural disasters (e.g., earthquakes and floods), human errors, software/hardware bugs or intentional failures due to maintenance actions or sabotage. Since these networks have an essential role in our social and economical activities, their interruption can cause severe damages. Network recovery refers to any action which brings a network to an operational state after failure.

In this context, reliability of a network element (e.g., a node or link) is defined as the probability of that element to be operational for a certain time frame. Availability refers to the probability that the element is in a functional state at any arbitrary time. Knowing the availability of network elements and assuming that these probabilities are mutually independent, the availability of a network path can simply be calculated by the product of the availability of the elements along the path. The availability of an element can be calculated based on mean time between failures (MTBF) and the mean time to repair (MTTR) of that element. MTBF refers to the average time between two consecutive failures of the element. MTTR is defined as the average time needed to restore the failing element. Using these parameters the availability of an element is defined as:

$$A = 1 - \frac{MTTR}{MTBF}$$

Five nine availability is the gold standard for telecommunication network components. This is typically used as a benchmark which means that only 5 minutes of network interruption is acceptable in a year.

In carrier-grade[5] networks, it is expected that the failure recovery time is approximately 50 ms. This means that the network should recover from a failure within this time.

[5]This term is used to refer to an extremely reliable and well-tested system or component.

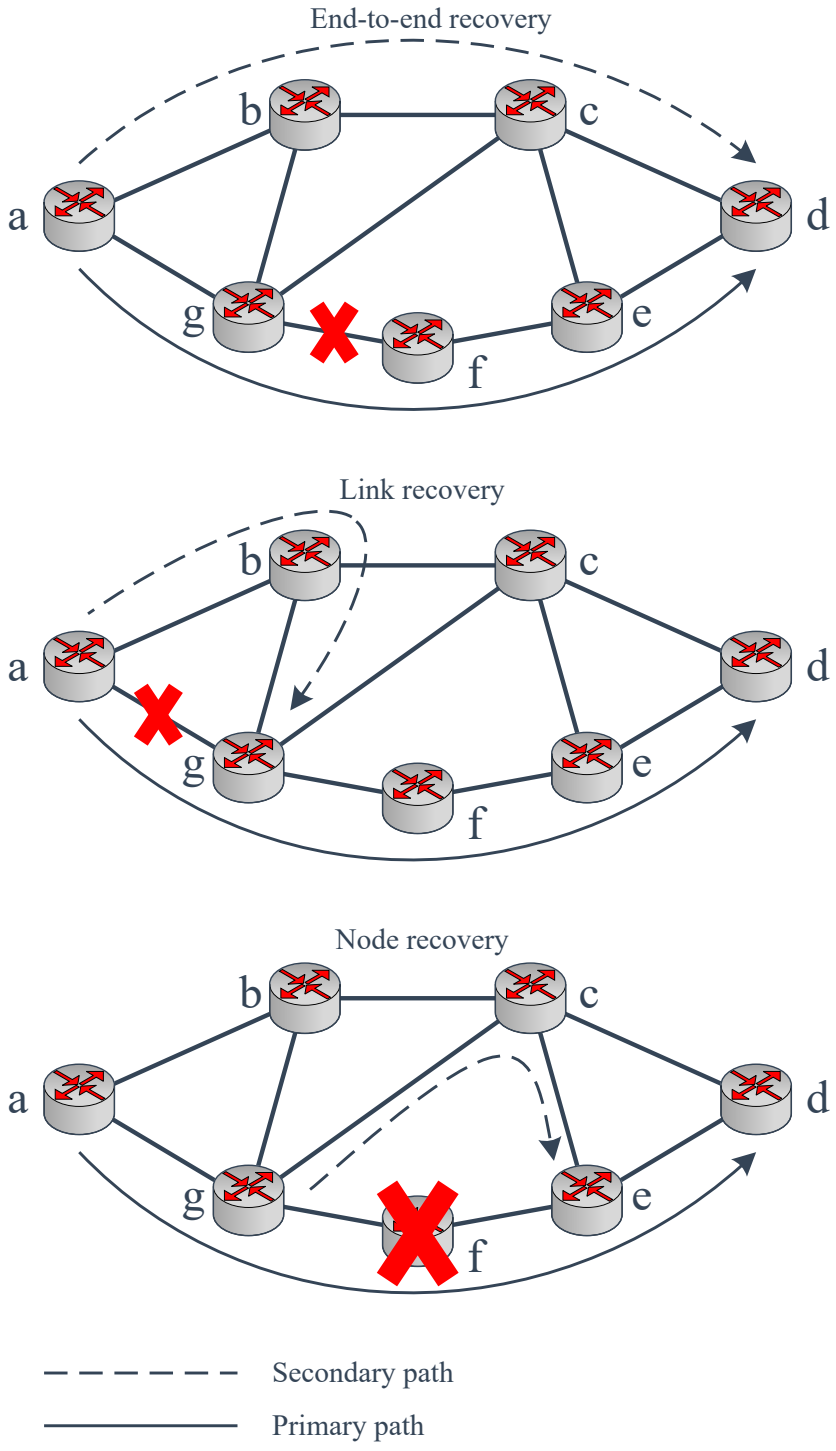


Figure 1.8: Example of end-to-end, link and node failure recovery

1.1.3.1 Recovery scope

The scope of the recovery can be global or local. Global refers to an end-to-end recovery in which a disjoint path is required for recovery provisioning. Local refers to both segment and link/node recovery in which the failing element/segment is bypassed by a local alternative path. The end-to-end and link/node recovery are depicted in Fig. 1.8. In the first figure on top, an end-to-end recovery is provided by an alternate disjoint path from 'a' to 'd'. In the other two figures, the link and node failure are recovered by a local path from 'a' to 'g' and 'g' to 'e' respectively. A recovery scheme provides two types of paths: i) working (primary) and ii) backup (secondary). Working path is used in case of normal operation when there is no failure in the network. Backup path is used upon occurrence of a failure in the working path.

1.1.3.2 Restoration

Restoration refers to a reactive strategy in which a backup path is finalized after the occurrence of a failure. Restoration techniques can be quite flexible with regard to failure scenarios while less backup capacity needs to be reserved in advance. However, the downside of these techniques is that it may take quite some time (i.e., order of seconds) to finalize the backup paths upon failure occurrence.

1.1.3.3 Protection

In contrast to restoration, protection is a proactive strategy in which backup paths are pre-planned and fully signaled before any failure in the network. Therefore upon occurrence of a failure, the backup paths can immediately be used without any additional signaling to establish them. This leads to a quite fast recovery while more resources should be reserved in advance. The recovery time in case of protection is in the order of milliseconds.

There are different variants of protection mechanism depending on the number of backup paths protecting a certain number of working ones.

- 1+1 protection
  This refers to a dedicated protection mechanism in which one backup path protects exactly one working path and traffic is duplicated on both paths. In this case no extra traffic is transmitted over the backup path.

- 1:1 protection
  This is also a dedicated backup path per working path. However, in case there is no failure in the network only the working path is used to send the traffic. Therefore, extra traffic can be sent over the backup path. In case of a failure, the traffic in the affected path is switched to the backup path.

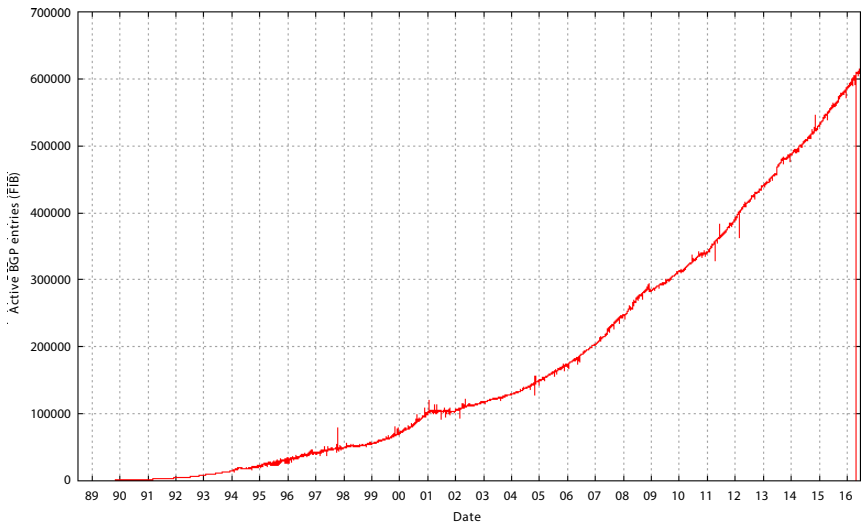


Figure 1.9: Number of active BGP entries in FIB in different years up to 2016, [17]

- 1:N protection
  In this case one backup path is used for protection of N working paths and the backup path can carry extra traffic in case of no failure.

- M : N protection
  This refers to mechanisms in which M backup paths protect N working paths and extra traffic is transmitted in the backup paths in failure-free scenarios.

1.2 The challenge of Internet routing

BGP as the fundamental inter-domain routing protocol in the Internet is significantly challenged by the increasing number of routers, ASes and routes. The linear increase of the number of ASes leads to enormous increase of BGP routing table entries. The reason is that each node should store so many routes in the network (e.g. due to the existence of shorter paths to smaller subnets as explained in Section 1.1). Fig. 1.9 depicts the number of active BGP entries in different years for an AS [17]. This extensive growth in the number of entries raises a severe scalability issue in terms of memory requirements of BGP routers.

The next issue is that BGP is subject to the path exploration phenomenon. This means that BGP routers may announce a route as valid although it is affected by a failure. The root cause of this is the dependency between routes propagated through the network. When a previously announced route is withdrawn, the other dependent routes may still be chosen and announced. These routes are withdrawn

later with the subsequent update messages. However, such a phenomenon leads to a significant increase in the number of update messages received by the routers. Path instability is another issue which is challenging the BGP routing. The issue is that routing tables should be frequently updated to adapt to network dynamics including link failures, new nodes joining or routers rebooting. This causes route flapping which means that the route to a destination in the network changes rapidly. This leads to continuous route addition (removal) to (from) the routing table. Route dampening is a mechanism to suppress flapping routes instead of advertising them. However, such mechanisms negatively impact the BGP convergence behavior. Both path exploration and route dampening lead to very high convergence time in BGP which has been shown to be in the order of minutes [18].

Several improvements to BGP have been proposed over the last twenty years. Enhanced path vector routing protocol (EPIC) annotates the AS paths with additional 'path dependency' information to reduce convergence time [19]. BGP with Root Cause Notification reduces the convergence time by announcing the root cause of a link failure location [20]. Path Exploration Damping augments BGP for selectively damping the propagation of path exploration updates [21]. New route selection schemes are proposed to improve route stability in BGP [22]. Most of these improvements relate to BGP dynamic properties and do not address scalability issue in BGP.

In spite of these prominent research efforts for improving/extending BGP, a suitable alternative is still missing. The reason is that the design of these routing systems tends to follow the exact same approach as the one pursued by BGP. Therefore, it is important to consider clean-slate approaches for designing new routing paradigms. Geometric routing is an alternative paradigm to solve the scalability limitation caused by IP routing. This routing is explained in the following subsection.

1.2.1 Geometric routing

Geometric routing has been proposed as an alternative to conventional LPM-based IP routing. It is used to address the scalability limitation in terms of memory requirement of forwarding tables. In geometric routing, every node in the network is assigned a specific coordinate in a metric space. These coordinates will be used as basis for making routing decisions. They are used by the routers to choose the next hop for sending the incoming packets. To do this, they rely on a distance metric present in the space. Every router, knowing the coordinates of its neighbors, forwards the incoming packets to a neighbor which is closer to the intended destination in terms of distance in the metric space. In this routing, the forwarding decision is only based on local information (i.e., coordinates of neighbors). An example of geometric routing is illustrated in Fig. 1.10.

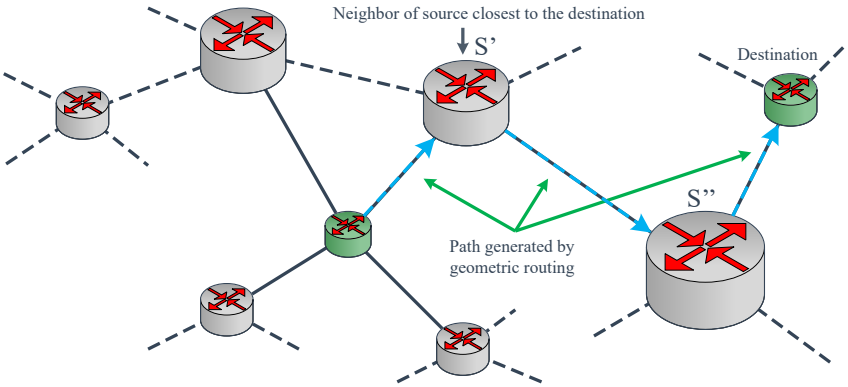


Figure 1.10: Example of geometric routing

Geometric routing was initially proposed for ad-hoc wireless networks and Wireless Sensor Networks (WSNs). In such environments routers have limited battery power and storage. Therefore, the forwarding decision should be inexpensive in terms of computation with low state requirements. Although geometric routing was investigated intensively in wireless networks, its application to wired networks and inter-domain settings, which have very different characteristics compared to wireless networks, remained mainly unexplored. This is the focus of this thesis.

1.2.1.1 Addressing and forwarding

Forwarding in geometric routing relies on network node coordinates. Coordinates are different from IP addresses since they only serve as locators. IP addresses fulfill both roles of identification and location. The impact of separation of these two roles on different aspects of routing (e.g. scalability) is an interesting research topic. This has been investigated vastly in researches related to Locator/Identifier Separation Protocol (LISP) [23] and Host Identity Protocol (HIP) [24].

There are two types of coordinates used in geometric routing:

- Physical coordinates
  Physical location of the nodes can be used as their coordinates. Thus an obvious and straightforward choice is to attach Global Positioning Systems (GPS) coordinates to each node [25].

- Virtual coordinates
  Since physical locations may be unavailable and also it is not guaranteed that every node has a GPS-like system, virtual coordinates have been introduced [26]. This way nodes are assigned virtual coordinates in a metric space (e.g.,

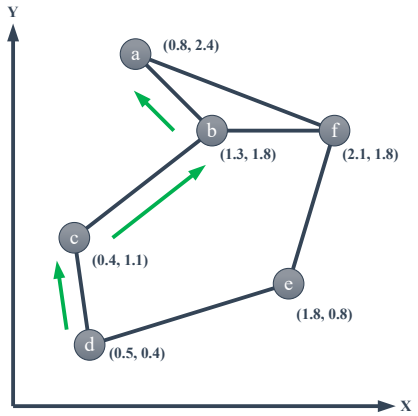


Figure 1.11: Example of greedy forwarding in Euclidean space. (x,y) are the virtual coordinates of the nodes.

Euclidean or Hyperbolic space). This can be considered as a generalization of the physical ones.

Through exchanging messages, nodes inform their neighbors about their coordinates. Upon arrival of such messages, each node can deduce a table referred to as neighbor table which contains the neighbors' coordinates. Since the number of neighbors is limited, the memory requirements of the routers are limited as well.

In geometric routing, forwarding is based on a distance-decreasing policy. In order to forward a packet towards its intended destination, the packet should contain the coordinate of the destination. Knowing the coordinates of the neighbors, upon arrival of a packet, the distance between every neighbor and the packet's destination is calculated. The neighbor which decreases the distance the most is selected as the next hop. This is referred to as greedy routing or forwarding. An example of greedy forwarding in Euclidean space is given in Fig. 1.11. Starting from node 'd' towards 'a', first the Euclidean distance between the neighbors of 'd' and 'a' is calculated. In this example, node 'c' is closer to 'a' compared to 'e'. Therefore, node 'c' is selected as the next hop. The same calculation is performed in nodes 'c' and 'b' until node 'a' is reached.

A major limitation in greedy forwarding is that following the distance-decreasing policy, the packets may get stuck in a local minimum (or void). This means that the current node is the closest node to the destination among its neighbors. This is illustrated in Fig. 1.12 where 'S' is the closest node to 'D' compared to the two neighbors. In this example, forwarding the packets to any neighbors of node 'S' does not decrease the distance towards node 'D'. Therefore, greedy forwarding is not feasible.

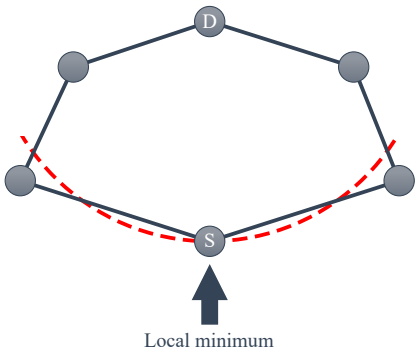


Figure 1.12: Example of local minimum

This has led to the introduction of greedy embeddings [27]. In a formal way a greedy embedding for a given graph $G(V, E)$ into a metric space X, is a function from $V(G)$ to X in such a way that for every graph node $s \neq t$, s has a neighbor u which decreases the distance towards t in metric space X. The embedding is called greedy because greedy routing based on the coordinates, derived from such an embedding, successfully reaches the intended destination. In this embedding, coordinates are assigned in such a way that for every node there is always a distance-decreasing neighbor towards any destination in the network. This way, in every step, a packet gets closer to its intended destination. Therefore, the destination can always be reached, as no local minimum exists. This is a significant concept as successful packet delivery is one of the essentials of the Internet backbone.

1.2.2 Research challenges in geometric routing

Most of the existing research on geometric routing focuses on WSNs, as geometric routing was initially proposed for such networks. The first part of this dissertation investigates the application of this routing on network topologies resembling the Internet backbone. Although introducing this routing in the Internet backbone may solve scalability in terms of memory requirements, it introduces new challenges in coping with dynamics such as network failures and coordinate changes. Therefore, the main contribution of this dissertation is to design a scalable and resilient geometric routing scheme. Different recovery mechanisms (i.e., protection and restoration) for geometric routing are proposed and a scalable greedy embedding with the related greedy forwarder are designed in this work.

1.3 The challenge of telecom virtualization

In recent years, cloud services and cloud networking have been an active research field, gaining massive attention from both providers and customers. From a customer perspective, cloud services reduce the cost while more flexible control of resources is achieved. Such flexibility and better control of resources is enabled by virtualizing the provider's resources. Virtualization is one of the main innovations of these days. It has been used for node processing and storage resources over the last years and it is now extending into the network. Virtualization decouples the services from the limitations of the underlying physical infrastructure. This provides more flexibility for deploying various services while simplifying the network management. This flexibility enables steering traffic where it needs to go (through certain network functions). Therefore, network resources which do not add value to certain traffic are released. These resources are used exclusively for traffics which do need them. From a provider perspective, this enables offering new services and efficient utilization of the hardware. Data centers (DCs) are the enablers of the cloud services. They are mainly composed of generic purpose hardware.

1.3.1 Network function virtualization

In the context of network virtualization, an industrial forum has been established by major service providers within the European Telecommunications Standards Institute (ETSI) referred to as Network Function Virtualization (NFV) [28]. The idea of NFV is to virtualize the network functions such as firewall, network address translation, intrusion detection, deep packet inspection and virtual private network. These are traditionally implemented by dedicated hardware and middleboxes. NFV decouples network functions from the dedicated hardware and enables deploying them in generic purpose hardware in a virtualized environment (e.g., in a virtual machine).

Initially it was assumed that the virtualized capability should only be implemented in DCs. However, the ideal situation is that virtualized network functions can be deployed anywhere in the network. This can be in DCs, network nodes or customer premises. This approach has been the target of many research projects such as UNIFY [29].

A concept closely related to NFV is Software Defined Networking (SDN) [30]. SDN decouples the control plane and data plane of networking equipment. In SDN the control plane is centralized, while data planes (forwarding components) are still distributed. The concept of SDN is depicted in Fig. 1.13. SDN provides a programmability environment in which the control plane programs the behavior of the forwarding components. Although these two concepts are independent, NFV can benefit from the SDN concept in implementation and management. SDN pro-

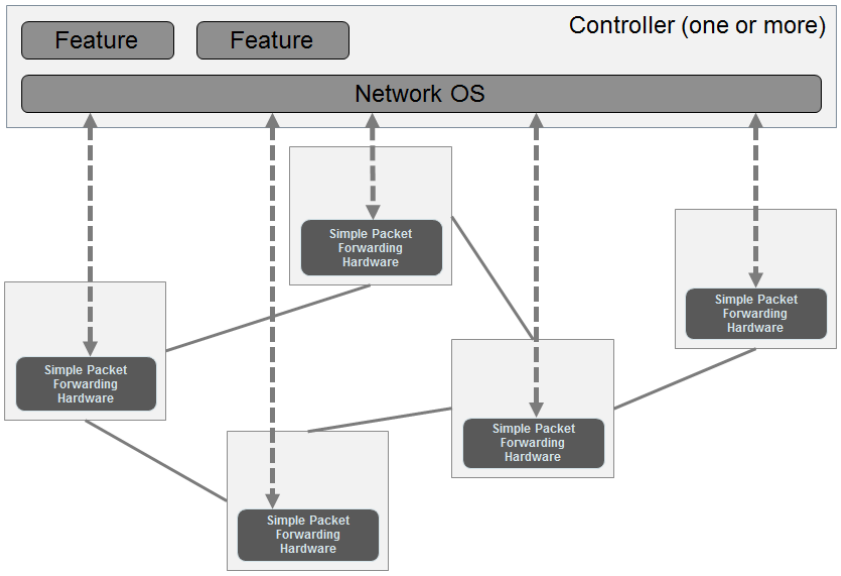


Figure 1.13: Software Defined Networking

vides a new control architecture enabling fine granular control over services. This architecture together with NFV enables service providers to offer various value added functions in addition to connectivity services.

1.3.2 Network service chaining

Network Service Chaining (NSC) is a service concept which has attracted interest and has been the focus of many researchers recently. NSC is enabled by the combination of cloud virtualization techniques, NFV and flexibility provided by SDN. Using these concepts, basic service building blocks (i.e., Network Functions (NFs)) can be chained across the network infrastructure. A service chain actually determines the order of different actions that should be applied to a data stream as it goes through the network. Relying on SDN principles, a controller dynamically creates and re-configures service chains across the network. The NFs within the service may lie in devices in different parts of the network. An example of NSC is illustrated in Fig. 1.14. As we see in this figure different NFs, i.e., gateway (GW), firewall (FW) and deep packet inspection (DPI), are located at different parts of the network (aggregation and core network). Traffic flows through these NFs in a specific order.

In the context of NSC, the control of resources, management and configuration of service chains become quite challenging. The only constraint on where/when to allocate resources to a service is the specification of the service itself. Based

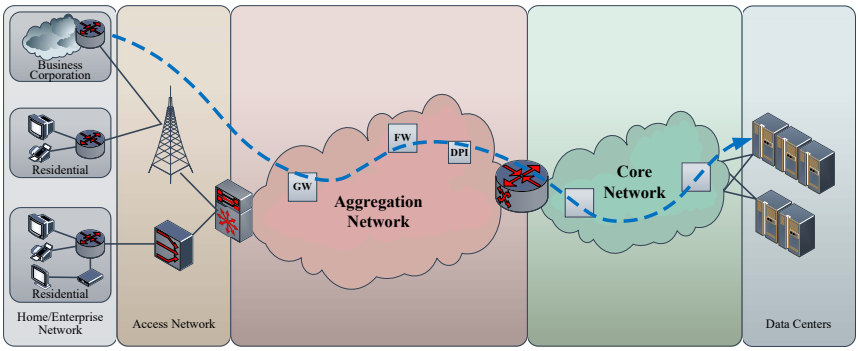


Figure 1.14: Example of Network Service Chaining with Network Functions located at different segments of the network

on the service requirements, the mapping or placement of NFs can be optimized. Sometimes the objective is to reduce the cost of the consumed resources. Other objectives can be to minimize the load, network congestion or energy consumption. Finding an optimal placement of the NFs within the service to the components of the network infrastructure is referred to as the embedding problem. Generally in this problem, the network infrastructure and the service chain is represented by graphs. Given several service graphs composed of multiple NFs and a common infrastructure (physical network), a mapping of the NFs and their logical links to the nodes and links of the physical network needs to be performed. Fig. 1.15 illustrates the general idea of the embedding problem with a simple example. In this figure, the two service graphs on the left should be mapped to the common physical network in the middle. On the right, the final placement of the NFs and their logical links to the physical nodes and links are depicted. The challenges related to this problem are further discussed in the following subsection.

1.3.3 Research challenges in service orchestration

The rise of SDN and NFV introduce opportunities for service providers to quickly deploy novel services with a reduced cost. NFV provides freedom in where to place an NF in a network. This creates new challenges in service orchestration. Existing solutions consider services as large monolithic blocks (e.g., network address translation or intrusion detection). Such monolithic service chains provide limited flexibility for dynamic scaling, demanded by user or caused by infrastructure policies. Relying on service chains, composed of micro functional NFs, enables dynamic orchestration of the NFs. Optimization algorithms can adapt the service chain according to different user demands or business policy changes. In practice, it is possible that a large monolithic service block can be decomposed into

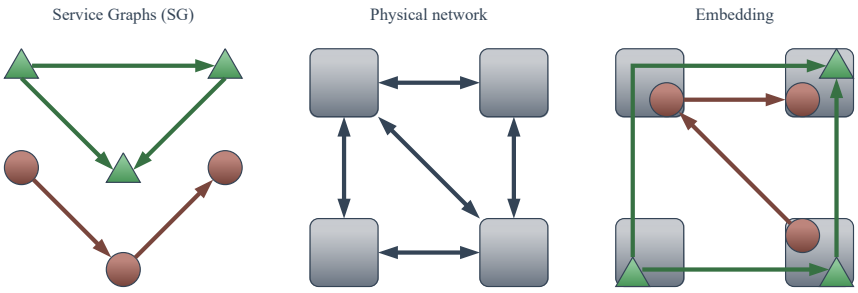


Figure 1.15: General idea of service chain embedding

inter-connected atomic NFs in multiple ways. This concept is referred to as service decomposition and is further detailed in Chapter 6. Having multiple realization options, the placement of NFs (embedding), control of resources, management and configuration of NSCs become quite challenging. The second part of this dissertation mainly focuses on the joint optimization of the service decomposition and embedding of NSCs.

The service orchestration process is significantly impacted by the scale of the telecom operator network and the number of service requests. Realistic service and infrastructure provider scenarios quickly involve ten thousands of resource elements and multiple services to be orchestrated. Such a large number of elements with multiple realization options for a service (service decompositions) challenges the scalability of service orchestration. This is another point which is further investigated in this dissertation.

1.4 Outline and research contributions

In this dissertation we mainly focus on the two research topic of scalable resilient geometric routing and network service chaining with optimized network function embedding. A number of publications were realized within the scope of this PhD some of which are selected to provide a thorough and consistent overview of the conducted work. The different research contributions and related chapters are detailed in this section and illustrated in Fig. 1.16. Each chapter is a journal article (or conference paper) which is already published or is in a submitted state. Section 1.5 provides a complete list of publications resulting from this work.

1.4.1 Geometric routing

In Chapter 2, resiliency against link failure in geometric routing is investigated. In order to forward the packets, network nodes are assigned coordinates in the hy-

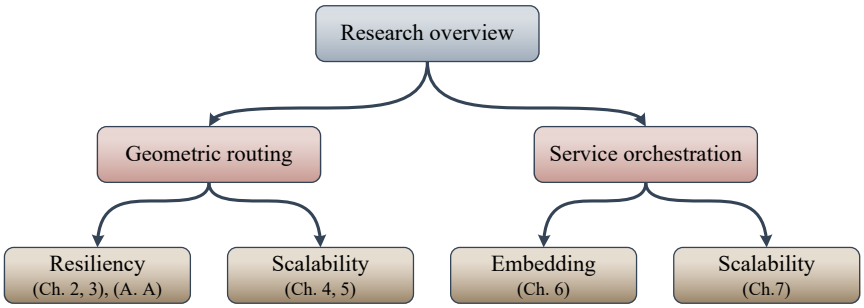


Figure 1.16: Overview of the research contribution. In this diagram, the related Chapters (Ch.) and Appendix (A.) to different research contributions are depicted.

perbolic plane [6], based on the scheme proposed in [31]. Using this scheme, it is guaranteed that a greedy embedding for any connected finite graph can be found. In order to cope with link failures, local recovery techniques for single and multiple failures are proposed. The schemes can be activated proactively or can be executed upon failure detection. They are evaluated on topologies with similar characteristics as the Internet. The proposed schemes require quite limited resources, enable fast switch-over and scale with the number of links in the spanning tree of the network.

In order to avoid complex coordinate calculation in metric spaces such as the hyperbolic plane, we proposed a simple scheme relying on a spanning tree of the network [32]. Network nodes are assigned coordinates based on their location in the tree. In Chapter 3, a distributed algorithm for calculation of this tree-based embedding is proposed and experimentally validated. This algorithm is extended to generate backups according to the scheme explained in Chapter 2. Although this extension provides a fault-tolerant routing scheme, a mechanism for dealing with various types of network dynamics such as node (or link) addition (or removal) is required. Having such a mechanism, the impact of topology changes on routing quality is reduced. To address this, the distributed algorithm is augmented with a mechanism to trigger nodes to re-calculate their coordinates in case of a change in the underlying structure (e.g. failure in the spanning tree). To evaluate the proposed scheme, the relative prototype implementation is provided in Quagga routing software[7] and Click modular router[8]. Quagga is used for the

[6]The hyperbolic plane H is a highly symmetric geometry in which the parallel postulate does not hold any longer. The latter implies that for a given line L in H and a point p not on L, there are at least two lines through p, which do not intersect L. In the Poincare Disk (PD) model of H, all points are represented by the set of points in $\mathbb{R}^2$ within the unit circle, that is $x^2 + y^2 < 1$. Points on the unit circle are ideal points (at infinity) in the resulting hyperbolic plane.

[7]http://www.nongnu.org/quagga/

[8]http://read.cs.ucla.edu/click/click

control plane while Click is used for the data plane implementation (i.e. greedy forwarding). Different metrics such as convergence time, protection time and communication cost are evaluated through emulation experiments. Since no abstraction is involved, realistic results with high accuracy can be achieved. The results indicate interesting convergence behavior in which convergence time/communication cost is quite low. The experiments are performed on large topologies of 1000 nodes which is a factor 10 improvement over the scale of state-of-the-art emulation experiments. The implemented prototype has been successfully demonstrated at IETF 90 Bits-N-Bites event[9].

Appendix A provides an insight on the availability performance of the proposed resilient geometric routing scheme on the actual Internet graph. The resilient scheme performs reasonably well compared to the shortest cycle scheme and improves availability significantly compared to the un-protected scheme.

Targeting large-scale networks, in Chapter 4 the scalability of the tree-based geometric routing is studied in more detail. The efficiency of this routing is identified through routing quality (deviation from the shortest path length, aka stretch[10]) and size of the coordinates. A related low-cost circuit, enabling greedy forwarding, is designed. The simulation results assess the proposal in large synthetic topologies of o(100K) nodes, resembling Internet topology. Comparison with the hyperbolic-based geometric routing indicates the superiority of the tree-based scheme in terms of coordinate memory scaling and complexity of hardware design. Similar performance in terms of stretch is achieved. Successful operation of the proposed greedy forwarder in the optical domain is verified in [33] and [34].

Since Internet is one among many networks including social, technological and communication systems described as complex networks, Chapter 5 provides an overview on advances and remaining challenges of routing in such large-scale networks. Both traditional as well as novel routing schemes are studied among which special attention is given to the variants of geometric routing. Analyzing the experimental results and analytical studies performed so far, the main trends and trade-offs are identified and the main conclusions are drawn. This overview better positions geometric routing among different routing schemes, enabling more careful conclusions regarding (in)suitability of these schemes to large-scale complex settings. It also provides open challenges and guidelines for future research.

1.4.2 Service orchestration

For the remainder of the dissertation, we focus on service orchestration in telecom networks. The problem of service chain embedding and service decomposition are

[9] https://www.ietf.org/meeting/90/ietf-90-bits-n-bites.html

[10] Stretch (S) is the ratio between the length of the path generated by a routing scheme (L_{gp}) and the corresponding shortest path length (L_{sp}): $S = \frac{L_{gp}}{L_{sp}}$

studied in detail. Chapter 6 investigates the joint optimization of network function embedding and service decomposition in the concept of network service chaining. First an algorithm based on Integer Linear Programming (ILP) is proposed to find the exact solution. The target is to minimize the cost of the mapping/embedding, taking into account the service chain requirements and capabilities of the infrastructure. Since the complexity of the exact solution is high, its scalability is quite limited. Therefore, it may not always be feasible to find the optimal solution in large-scale scenarios in a reasonable time. In order to solve this scalability limitation of the ILP, a heuristic solution is provided. The heuristic approach is composed of two phases: i) decomposition selection and ii) mapping. Given a service chain, this approach first selects a suitable service decomposition, taking several parameters (e.g., number of NFs in the decomposition and number of candidate physical nodes which can potentially host the NFs) into account. Then relying on a backtracking mechanism, a mapping of the selected decomposition to the infrastructure is performed. Simulation results indicate the benefits of joint embedding and service decomposition in terms of service acceptance rate and the related mapping cost.

The studies reported in Chapter 7 intend to highlight the role of involved interfaces (for enabling NSC with virtual NFs) in a scalable service orchestrator and identify the major contributors in orchestration time in a proof of concept prototype. This prototype relies on the embedding algorithm proposed in Chapter 6. Based on the identified major contributors, several architectural enhancements are proposed to move towards a more scalable orchestrator. The related challenges are discussed as well. This will provide several guidelines for future work.

Finally, Chapter 8 concludes this dissertation with the main outcomes and presents the future research directions.

1.5 Publications

The research results obtained during this PhD research have been published in scientific journals and presented at a series of international conferences. The following list provides an overview of the publications during my PhD research.

1.5.1 Publications in international journals (listed in the Science Citation Index [11])

1. **Sahel Sahhaf**, Wouter Tavernier, Didier Colle, Mario Pickavet and Piet Demeester, *Link failure recovery technique for greedy routing in the hyperbolic*

[11] The publications listed are recognized as 'A1 publications', according to the following definition used by Ghent University: A1 publications are articles listed in the Science Citation Index Expanded, the Social Science Citation Index or the Arts and Humanities Citation Index of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper.

plane, Computer Communications, Vol. 36(6), 2013, pp. 698-707.

2. **Sahel Sahhaf**, Wouter Tavernier, Didier Colle, Mario Pickavet and Piet Demeester, *Experimental validation of resilient tree-based greedy geometric routing*, Computer Networks, Vol. 82, 2015, pp. 156-171.

3. **Sahel Sahhaf**, Wouter Tavernier, Mattias Rost, Stefan Schmid, Didier Colle, Mario Pickavet and Piet Demeester, *Network service chaining with optimized network function embedding supporting service decompositions*, Computer Networks, Vol. 93, 2015, pp. 492-505.

4. Rein Houthooft, **Sahel Sahhaf**, Wouter Tavernier, Filip De Turck, Didier Colle and Mario Pickavet, *Optimizing robustness in geometric routing via embedding redundancy and regeneration*, Networks, Vol. 66(4), 2015, pp. 320-334.

5. Sachin Sharma, Wouter Tavernier, **Sahel Sahhaf**, Didier Colle, Mario Pickavet and Piet Demeester, *Verification of flow matching functionality in the forwarding plane of openflow networks*, IEICE Transactions on Communications, Vol. 98(11), 2015, pp. 2190-2201.

6. **Sahel Sahhaf**, Wouter Tavernier, Didier Colle, Mario Pickavet and Piet Demeester, *Efficient geometric routing in large-scale complex networks with low-cost node design*, IEICE Transactions on Communications, Vol. 99(3), 2016, pp. 666-674.

7. **Sahel Sahhaf**, Wouter Tavernier, Dimitri Papadimitriou, Davide Careglio, Alok Kumar, Christian Glacet, David Coudert, Nicolas Nisse, Lluís Fabrega, Miguel Camelo, Pere Vila, Pieter Audenaert, Didier Colle and Piet Demeester, *Routing at large scale: advances and challenges for complex networks*, IEEE Network (submitted).

8. **Sahel Sahhaf**, Wouter Tavernier, Didier Colle and Mario Pickavet, *Adaptive availability and bandwidth-aware multipath provisioning for media transfer in SDN-based overlay network*, Computer Communications (submitted).

1.5.2 Publications in international conferences (listed in the Science Citation Index [12])

1. **Sahel Sahhaf**, Wouter Tavernier, Didier Colle, Mario Pickavet and Piet Demeester, *Single failure resiliency in greedy routing*, 9th International Confer-

[12]The publications listed are recognized as 'P1 publications', according to the following definition used by Ghent University: P1 publications are proceedings listed in the Conference Proceedings Citation Index - Science or Conference Proceedings Citation Index - Social Science and Humanities of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper, except for publications that are classified as A1.

ence on Design of Reliable Communication Networks (DRCN), 2013, pp. 306-313, Budapest, Hungary.

2. **Sahel Sahhaf**, Abhishek Dixit, Wouter Tavernier, Didier Colle, Mario Pickavet and Piet Demeester, *Scalable and energy-efficient optical tree-based greedy router*, 15th International Conference on Transparent Optical Networks (ICTON), 2013, pp.1-4, Cartagena, Spain.

3. **Sahel Sahhaf**, Wouter Tavernier, Didier Colle, Mario Pickavet and Piet Demeester, *Availability analysis of resilient geometric routing on Internet topology*, 10th International Conference on Design of Reliable Communication Networks (DRCN), 2014, pp. 1-8, Ghent, Belgium.

4. **Sahel Sahhaf**, Maryam Barshan, Wouter Tavernier, Hendrik Moens, Didier Colle and Mario Pickavet, *Resilient algorithms for advance bandwidth reservation in media production networks*, 12th International Conference on Design of Reliable Communication Networks (DRCN), 2016, pp. 130-137, Paris, France.

1.5.3 Publications in other international conferences

1. **Sahel Sahhaf**, Abhishek Dixit, Wouter Tavernier, Didier Colle, Mario Pickavet and Piet Demeester, *All-optical tree-based greedy router*, Conference on Optical Fiber Communication (OFC), 2014, pp. W2A-7, San Francisco, USA.

2. **Sahel Sahhaf**, Dimitri Papadimitriou, Wouter Tavernier, Didier Colle and Mario Pickavet, *Experimentation of geometric information routing on content locators*, IEEE 22nd International Conference on Network Protocols (ICNP), 2014, pp. 518-524, Raleigh, USA.

3. Rein Houthooft, **Sahel Sahhaf**, Wouter Tavernier, Filip De Turck, Didier Colle and Mario Pickavet, *Fault-tolerant greedy forest routing for complex networks*, 6th International Workshop on Reliable Networks Design and Modeling (RNDM), 2014, pp. 1-8, Barcelona, Spain.

4. Davide Careglio, Dimitri Papadimitriou, Fernando Agraz, **Sahel Sahhaf**, Jordi Perello, Wouter Tavernier, Salvatore Spadaro and Didier Colle, *Development and experimentation towards a multi-cast-enabled Internet*, IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2014, pp. 79-84, Toronto, Canada.

5. Davide Careglio, Dimitri Papadimitriou, Fernando Agraz, **Sahel Sahhaf**, Jordi Perello and Wouter Tavernier, *On the experimentation of the novel*

GCMR multicast routing in a large-scale testbed, IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2014, pp. 105-106, Toronto, Canada.

6. Wouter Tavernier, **Sahel Sahhaf**, Didier Colle and Mario Pickavet, *Towards content-centric geometric routing*, IEEE 21st Symposium on Communications and Vehicular Technology in the Benelux (SCVT), 2014, pp. 133-138, Mons, Belgium.

7. Attila Csoma, Balazs Sonkoly, Levente Csikor, Felician Nemeth, Andras Gulyas, David Jocha, Janos Elek, Wouter Tavernier and **Sahel Sahhaf**, *Multi-layered service orchestration in a multi-domain network environment*, 3rd European Workshop on Software Defined Networks (EWSDN), 2014, pp. 141-142, Budapest, Hungary.

8. Attila Csoma, Balazs Sonkoly, Levente Csikor, Felician Nemeth, Andras Gulyas, Wouter Tavernier and **Sahel Sahhaf**, *ESCAPE: Extensible Service ChAin Prototyping Environment using Mininet, Click, NETCONF and POX*, ACM SIGCOMM Computer Communication Review, 2014, pp. 125-126, Chicago, USA.

9. **Sahel Sahhaf**, Wouter Tavernier, Didier Colle and Mario Pickavet, *Network service chaining with efficient network function mapping based on service decompositions*, 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1-5, London, UK.

10. **Sahel Sahhaf**, Wouter Tavernier, Janos Czentye, Balazs Sonkoly, Pontus Skoldstrom, David Jocha and Jokin Garay, *Scalable architecture for service function chain orchestration*, 4th European Workshop on Software Defined Networks (EWSDN), 2015, pp. 19-24, Bilbao, Spain.

11. Balazs Sonkoly, Janos Czentye, Robert Szabo, David Jocha, Janos Elek, **Sahel Sahhaf**, Wouter Tavernier and Fulvio Risso, *Multi-domain service orchestration over networks and clouds: a unified approach*, ACM SIGCOMM Computer Communication Review, 2015, pp. 377-378, London, UK.

12. Rein Houthooft, **Sahel Sahhaf**, Wouter Tavernier, Filip De Turck, Didier Colle and Mario Pickavet, *Robust geometric forest routing with tunable load balancing*, IEEE Conference on Computer Communications (INFOCOM), 2015, pp. 1382-1390, Hong Kong, China.

13. **Sahel Sahhaf**, Wouter Tavernier, Didier Colle and Mario Pickavet, *Resilient availability and sandwidth-aware multipath provisioning for media transfer*

over the Internet, 8th International Workshop on Resilient Networks Design and Modeling (RNDM), 2016, Halmstad, Sweden.

14. Thomas Soenen, **Sahel Sahhaf**, Wouter Tavernier, Pontus Skoldstrom, Didier Colle and Mario Pickavet, *A model to select the right infrastructure abstraction for service function chaining*, IEEE Conference on Network Function Virtualization and Software Defined Networks, 2016, Palo Alto, USA .

1.5.4 Publications in national conferences

1. **Sahel Sahhaf**, Wouter Tavernier, Didier Colle and Mario Pickavet, *Greedy routing in future internet*, 9th FEA PhD Symposium, 2012, Ghent, Belgium.

2. **Sahel Sahhaf**, Wouter Tavernier, Didier Colle and Mario Pickavet, *Experimentation of geometric information routing on content locators*, 11th FEA PhD Symposium, 2014, Ghent, Belgium.

References

[1] P. J. Winzer. *Beyond 100G ethernet*. IEEE Communications Magazine, 48(7):26–30, 2010.

[2] M. Tahon, M. Van der Wee, S. Verbrugge, D. Colle, and M. Pickavet. *The impact of inter-platform competition on the economic viability of municipal fiber networks*. In Optical Fiber Communication Conference, pages Th1G–3. Optical Society of America, 2014.

[3] D. A. Maltz, G. Xie, J. Zhan, H. Zhang, G. Hjálmtỳsson, and A. Greenberg. *Routing design in operational networks: A look from the inside*. In ACM SIGCOMM Computer Communication Review, volume 34, pages 27–40. ACM, 2004.

[4] M. Al-Fares, A. Loukissas, and A. Vahdat. *A scalable, commodity data center network architecture*. In ACM SIGCOMM Computer Communication Review, volume 38, pages 63–74. ACM, 2008.

[5] J. Postel et al. *IETF RFC 791: Internet protocol*. 1981. Available from: https://tools.ietf.org/html/rfc791.

[6] S. E. Deering. *Internet protocol, version 6 (IPv6) specification*. 1998. Available from: https://www.ietf.org/rfc/rfc2460.txt.

[7] V. Fuller and T. Li. *Classless inter-domain routing (CIDR): The Internet address assignment and aggregation plan*. 2006. Available from: https://tools.ietf.org/html/rfc4632.

[8] J. Moy. *IETF RFC 2328: OSPF Version 2*, 1998. Available from: https://www.ietf.org/rfc/rfc2328.txt.

[9] D. Oran. *IETF RFC 1142: OSI IS-IS intra-domain routing protocol*, 1990. Available from: https://tools.ietf.org/html/rfc1142.

[10] C. Hendrik. *IETF RFC 1058, Routing Information Protocol*. 1988. Available from: https://tools.ietf.org/html/rfc1058.

[11] G. Malkin. *IETF RFC 2453: RIP version 2*, 1998. Available from: https://tools.ietf.org/html/rfc2453.

[12] G. Malkin and R. Minnear. *IETF RFC 2080: Ripng for ipv6*, 1997. Available from: https://tools.ietf.org/html/rfc2080.

[13] C. Hedrick and L. Bosack. *An introduction to IGRP*. Rutgers-The State University of New Jersey Technical Publication, Laboratory for Computer Science, 1991.

[14] D. Farinachi. *Introduction to enhanced IGRP (EIGRP)*. Cisco Systems Inc, 1993.

[15] Y. Rekhter, T. Li, and S. Hares. *IETF RFC 4271: a border gateway protocol 4 (bgp-4)*. 12, 2006. Available from: https://tools.ietf.org/html/rfc4271.

[16] A. Narayanan. *A survey on BGP issues and solutions*. arXiv preprint arXiv:0907.4815, 2009.

[17] G. Huston. *BGP Routing Table Reports*, 2016. Available from: http://bgp.potaroo.net/.

[18] R. Oliveira, B. Zhang, D. Pei, and L. Zhang. *Quantifying path exploration in the internet*. IEEE/ACM Transactions on Networking, 17(2):445–458, 2009.

[19] J. Chandrashekar, Z. Duan, Z.-L. Zhang, and J. Krasky. *Limiting path exploration in BGP*. In Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies., volume 4, pages 2337–2348. IEEE, 2005.

[20] D. Pei, M. Azuma, D. Massey, and L. Zhang. *BGP-RCN: Improving BGP convergence through root cause notification*. Computer Networks, 48(2):175–194, 2005.

[21] G. Huston, M. Rossi, and G. Armitage. *A technique for reducing BGP update announcements through path exploration damping*. IEEE Journal on Selected Areas in Communications, 28(8):1271–1286, 2010.

[22] P. B. Godfrey, M. Caesar, I. Haken, Y. Singer, S. Shenker, and I. Stoica. *Stabilizing route selection in bgp*. IEEE/ACM Transactions on Networking, 23(1):282–299, 2015.

[23] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. *IETF RFC 6830: The locator/id separation protocol (lisp)*, 2013. Available from: https://tools.ietf.org/html/rfc6830.

[24] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. *IETF RFC 5201 Host Identity Protocol*, 2008. Available from: https://tools.ietf.org/html/rfc5201.

[25] B. Karp and H.-T. Kung. *GPSR: Greedy perimeter stateless routing for wireless networks*. In Proceedings of the 6th annual international conference on Mobile computing and networking, pages 243–254. ACM, 2000.

[26] 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 Proceedings of the 2nd conference on Symposium on

Networked Systems Design & Implementation-Volume 2, pages 329–342. USENIX Association, 2005.

[27] C. H. Papadimitriou and D. Ratajczak. *On a conjecture related to geometric routing*. Theoretical Computer Science, 344(1):3–14, 2005.

[28] ETSI. *White Paper: Network Functions Virtualisation (NFV)*, 2013. Available from: http://portal.etsi.org/NFV/NFV_White_Paper2.pdf.

[29] *The UNIFY project*. Available from: http://fp7-unify.eu/.

[30] ONF. *Open Networking Foundation*, 2014. Available from: https://www.opennetworking.org/.

[31] R. Kleinberg. *Geographic routing using hyperbolic space*. In INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, pages 1902–1909, 2007.

[32] S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester. *Single failure resiliency in greedy routing*. In Proceedings of the 9th international conference on Design of Reliable Communication Networks, pages 312–319, 2013.

[33] S. Sahhaf, A. Dixit, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester. *Scalable and energy-efficient optical tree-based greedy router*. In 2013 15th International Conference on Transparent Optical Networks (ICTON), pages 1–4. IEEE, 2013.

[34] S. Sahhaf, A. Dixit, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester. *All-optical tree-based greedy router*. In Optical Fiber Communication Conference, pages W2A–7. Optical Society of America, 2014.

2

Link failure recovery technique for greedy routing in the hyperbolic plane

This chapter proposes recovery mechanisms for single and multiple link failures in geometric routing. In order to perform greedy routing, network nodes are assigned coordinates in the hyperbolic plane. The memory requirement and packet overhead imposed by the schemes are quite limited. Additionally, the proposed schemes scale with the number of edges in the spanning tree of the network. These make them promising solutions for large-scale networks.

⋆ ⋆ ⋆

S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, P. Demeester

Published in Computer Communications, 2013.

Abstract The scalability of current routing protocols is limited by the linearly increasing size of the corresponding routing tables. Greedy routing has been proposed as a solution to this scalability problem. In greedy routing, every node is assigned a coordinate. These coordinates are used in order to forward a packet to a neighbor which is closer to the destination. Current greedy methods cannot efficiently cope with failures in topology. Using methods which require large resources and have significant loss in the quality of the routing (stretch loss) makes greedy routing useless in large-scale networks. In this paper, local techniques for

single and multiple link failure recovery are proposed. The methods require very limited resources and result into limited loss in routing quality. The proposed schemes allow fast switch-over and scale with the number of links in the spanning tree of the network. Scalability, simplicity and low overhead of the methods make them suitable for large networks. The proposed techniques are evaluated in an experimental environment.

2.1 Introduction

In traditional IP routing, forwarding decisions are based on longest-prefix matching in a routing table. Routing protocols scale corresponding to the topology size. The more IP addresses/prefixes assigned, the larger the routing tables have to become ([1] reports more than 367 K FIB entries in current BGP routers). Geographic routing schemes have been proposed as a solution to this problem [2].

Greedy routing stems from the idea of geographic routing where every node in a network receives a GPS coordinate [2]. Network nodes knowing the coordinates of their neighbors could then route greedy, by relaying incoming packets to the neighbors which are closer to the packet's intended destination such that the distance towards the packet's destination decreases. Repeatedly applying this distance-decreasing policy will eventually lead to the destination. The same idea could be reused by assigning virtual coordinates in a given geometric space. A drawback of the mentioned techniques is that packets might get stuck into a local minimum (lakes or voids). Greedy embeddings, as proposed by Papadimitriou and Ratajczak [3], avoid this situation by ensuring that coordinates are mapped to the network nodes in such a way that local minima can never happen. More formally, a *greedy embedding* for a given graph $G(V, E)$ into a metric space X, is a function from V to X such that for all the graph nodes $s \neq t$, s has a neighbor u which decreases the distance toward t in metric space X. This implies that using a greedy routing algorithm, packets will eventually reach the destination in any case.

Kleinberg proved in [4] that it is possible to find such a greedy embedding in the hyperbolic plane for any connected finite graph. We will base our work on this seminal work. As will be described in more detail in the next section, the paper allows network nodes to derive their coordinates in the Poincare Disk in a distributed way, based on a spanning tree of the original network topology. The latter paper however did not investigate the effect of link/node failures in the network. In Section 2.4, it will be explained that even a single link/node failure might affect the greedy embedding in such a way that greedy routing is no longer possible. As will be described in next section, the reason is that greedy embedding is based on the connectivity of the graph's spanning tree. Therefore, a change in this connectivity might result in recalculation of coordinates in order to have a successful packet delivery to a specified destination. As possibly too many nodes

require changing their coordinates, the resulting disruptions would be very long.

Our goal is to avoid coordinate recalculation upon link failures. Therefore, in this paper, we propose recovery methods for single and multiple link failures. Using these methods, we are able to bypass failing links without the requirement of recalculating the coordinates. While greedy routing is proceeding, upon a failure detection an alternative path is selected which recovers the faulty link. Unlike many previous studies, our approach avoids the lakes or voids from the beginning and therefore, there is no requirement to keep track of local minima or the visited nodes along the path. This makes the overhead added to the header of a packet very limited. The proposed schemes can be used in a pro-active mode, enabling fast switch-over. The methods are scaling well which makes them suitable for large-scale networks. As we will see later, the techniques scale with the number of links in the spanning tree of the network. This is equivalent to scaling linearly with the number of nodes in the network.

The rest of the paper is structured as follows. Section 2.2 briefly describes some of the greedy embedding related works and the works focusing on dynamics of the network in greedy routing. Greedy embedding of Kleinberg which we base our work on is explained in Section 2.3. The next section describes the proposed recovery methods for single and multiple link failures in greedy routing. Experimental evaluation of the proposed algorithms are described in Section 2.5. Finally, the last section concludes the paper.

2.2 Existing work

Studies on geographic routing by Karp and Kung [2] are one of the primary steps towards greedy routing. In their work, every node is assigned a coordinate which reflects the physical location of that node. In this routing, every node has knowledge about the coordinates of all of its neighbors. Using this information a node forwards an incoming packet to a neighbor which decreases the distance towards the destination the most. In [5], the same idea was reused but instead of applying the physical location of nodes to the coordinates a virtual coordinate in some other metric space was assigned to each node of the network, leading to the introduction of geometric routing.

The problem of greedy routing (both geographic and geometric routing) is that the packets might get stuck in a local minimum (also called void or lake), which means that there is no other neighbor which decreases the distance towards the destination more than the current node. In order to overcome this problem two groups of techniques were proposed. In the first group, the void or lake is passed by routing around this area and greedy routing is resumed from the point that we reach a node which is closer to the destination than the local minimum. This technique is referred as face routing. The two problems of this group are: (i) local

parts of the graph must be planar (or planarized) and (ii) the latter might not be possible for all kind of graphs. A survey on different techniques to handle voids in geographic routing is available in [6].

The second group tries to find a greedy embedding of the nodes in a metric space in which no voids exist any more. As was mentioned in the introduction, using these embedded coordinates guarantees that greedy routing is always successful in delivering the packets to every destination. Groundbreaking work of Kleinberg [4] proved the existence of a greedy embedding for every connected finite graph, in the hyperbolic plane.

As [4] did not explicitly discuss the worst-case required coordinate precision, the authors of [7] proposed a greedy embedding in the hyperbolic plane using only $O(logn)$ bits for coordinate representation, where n is the number of nodes in the network. However, they did not evaluate the resulting stretch of the proposed scheme.

The two works [8, 9] propose methods to embed graphs in Euclidean space. In [8], the embedding is into a Euclidean $O(log^2 n)$-dimensional space and authors in [9] propose an embedding into Euclidean space of poly-logarithmic dimension.

Boguna et al. [10] showed that statistical structure of Internet graph[1] follows a similar statistical structure of a hyperbolic geometry. They proposed that Internet inherently follows a hidden hyperbolic structure. Papadopoulos et. al. [11] also proposed a mechanism of network growth in a hyperbolic hidden metric space in which greedy forwarding is always successful even in dynamic network topologies.

The issue of network dynamics such as link/node failures or node additions in greedy routing (based on greedy embeddings) has not been studied significantly. However, the authors of [12] proposed an alternative embedding in the hyperbolic plane, which enables incremental embedding of network nodes, without disturbing the global embedding. Nevertheless, the resulting embedding cannot easily be adapted to cope with the removal of a network node/link. In the latter situation, the authors suggest not to change the embedding, but to use an adapted routing algorithm in case a packet gets stuck in a local minimum. The resulting routing is referred as Gravity-Pressure routing. The technique is different from face routing as it keeps forwarding via neighbors which decrease the distance towards the destination as much as possible. However, in order to avoid that the packet gets stuck into a loop, a path trace needs to be maintained in each packet from the moment a local minimum is detected, which puts large overhead to the header of the packets.

In this paper, we base our work on greedy embedding of Kleinberg [4] and try to handle the network dynamics (link failures) by two recovery methods. The proposed schemes do not need a re-embedding, guarantee a given link failure coverage, and can be used as a protection scheme, enabling faster switch-over than

[1] Scale-free graph with power law degree distribution.

existing alternatives. The used recovery paths demonstrate interesting stretch characteristics compared to existing dynamic alternatives (Gravity-Pressure routing). Unlike [12], our approach does not need to maintain the list of the nodes visited by a packet therefore, the overhead added to the header of a packet is not significant. The proposed methods can also be applied to large-scale networks due to their scalability, simplicity and low overhead which is suitable for distributed implementations.

2.3 Greedy embedding in the hyperbolic plane

The hyperbolic plane H is a highly symmetric geometry in which the parallel postulate does not hold any longer. The latter implies that for a given line L in H and a point p not on L, there are at least two lines through p, which do not intersect L. In the Poincare Disk (PD) model of H, all points are represented by the set of points in $\mathbb{R}^2$ within the unit circle, that is $x^2 + y^2 < 1$. Points on the unit circle are ideal points (at infinity) in the resulting hyperbolic plane. The distance between points u and v in the PD is defined as $d(u, v) = arccosh(1 + \delta(u, v))$, where

$$\delta(u, v) = 2\frac{||u - v||^2}{(1 - ||u||^2)(1 - ||v||^2)}$$

$||.||$ denotes the usual Euclidean norm.

A crucial characteristic of the hyperbolic plane is that because of its high symmetry, it allows infinitely many uniform tilings using d-regular polygons. Dual to these tilings is the d-regular tree formed by the midpoints of the polygons producing the tiling. Therefore, any spanning tree of a given network topology can be mapped to a subtree of a sufficiently large d-regular tree dual to a corresponding tiling. This is the core property which is used by Kleinberg [4] to construct an embedding of any graph. A tiling of d-regular polygons of the PD can be deduced using Möbius transformations which are chosen with respect to d. We briefly explain the required subtasks in order to embed a graph into the hyperbolic plane and for theoretical explanations and mathematical proofs, we refer interested readers to [4]. The required subtasks for computing virtual coordinates of network nodes (embedding) are as follows:

1. Construct a spanning tree of the network.
2. Determine the maximum degree of this tree, d.
3. Generate infinite d-regular tree in the hyperbolic plane and name each node of the spanning tree with a node of this infinite d-regular tree.

These steps are clearly depicted in Fig. 2.1. In the first step, a spanning tree of the graph is constructed, the maximum degree of this tree is 3 therefore, the

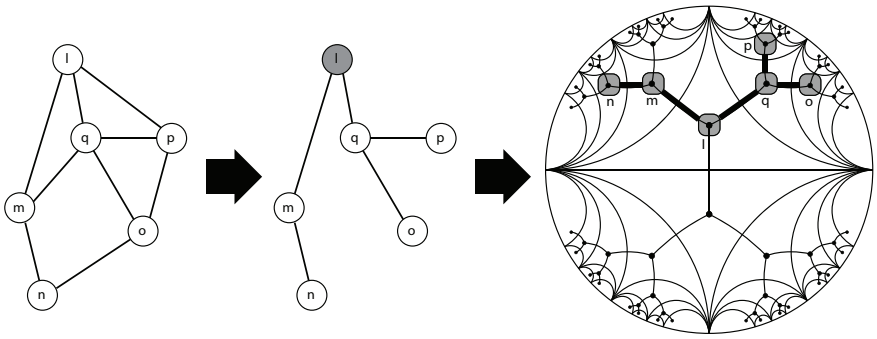


Figure 2.1: Steps for embedding a graph into the hyperbolic plane using Kleinberg's greedy embedding

infinite 3-regular tree in the hyperbolic plane is generated. Kleinberg [4] proposes a distributed algorithm to assign the coordinates of the d-regular tree in the PD to the network nodes. In the final step in the figure, we observe the embedded graph in the hyperbolic plane.

The spanning tree used for determining the greedy embedding (coordinates) ensures that there is always a distance-decreasing path (via the tree). However, greedy routing based on this embedding is not the same as routing on the resulting tree, because shortcuts (the edges which are not in the spanning tree) will be taken as well.

2.4 Recovery method in greedy routing

We apply the greedy embedding as was explained in Section 2.3 to assign coordinates to the network nodes in the hyperbolic plane. We assume that the greedy routing always uses these virtual coordinates in order to proceed.

Using this embedding, a single link failure might invalidate the greedy embedding. In Section 2.3, we observed that greedy embedding is based on the spanning tree of the network and this spanning tree ensures that there is always a distance-decreasing path (via the tree). However, if a link/node failure affects the connectivity of the spanning tree, it might also affect the greedy embedding and therefore the greedy routing. A simple example in Fig. 2.2 depicts this problem. In the figure, edges of the spanning tree are represented by solid lines and shortcuts in the network are represented by dashed lines. The integer IDs are used only to represent the network nodes. Note that these IDs are not equal to the nodes' virtual coordinates in the hyperbolic plane which are used for the greedy routing. In this example, the greedy path from node 1 to node 7 is on the tree ($1 \rightarrow 3 \rightarrow 7$). The failure in link (1-3) makes node 1 a local minimum (based on the embedding in

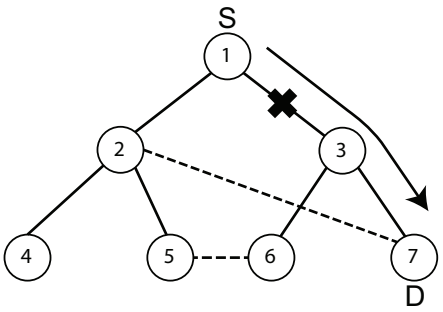


Figure 2.2: Example for a single link failure which affects the greedy embedding.

the hyperbolic plane node 2 is not closer to 7 than node 1). Therefore, the packets get stuck in this node.

A solution to this problem can be to find another embedding which makes greedy routing possible again. However, this solution might cause frequent recalculation of the coordinates, which is not realistic in large-scale networks. Coordinate recalculation of too many nodes results into too long disruptions which is not desired. Our goal is to overcome the issue of link failure in an efficient way without any changes in the coordinates. Therefore, we propose recovery methods for single and multiple link failures in greedy routing. As we know, recovery methods can be protection schemes or restoration schemes or a combination of both. In protection methods, a backup path is determined before any failure in the network. These methods enable very fast recovery. However, in the restoration methods the backup path can be installed on-demand dynamically [13]. Our proposed methods are in pro-active mode (protection) however, the method for single failure is also suitable for reactive mode (restoration). The methods assume that the network is biconnected.[2]

2.4.1 Single link failure

In this section, we propose a recovery method to be used in case of a single link failure. There are some methods and protocols available in order to detect a failure in the network. An example protocol could be Bidirectional Forwarding Detection (BFD) [14]. Our proposed method is applicable only if there is a single link failure in the network and the failed link belongs to the spanning tree of the network. The reason for applying the method only for tree edges is that, using Kleinberg's greedy embedding if the faulty link is a shortcut (a link which is not in the spanning tree) greedy routing can still be continued on the tree edges. The problem of getting

[2] A biconnected graph is a connected graph that cannot be broken into disconnected pieces by deleting any single vertex.

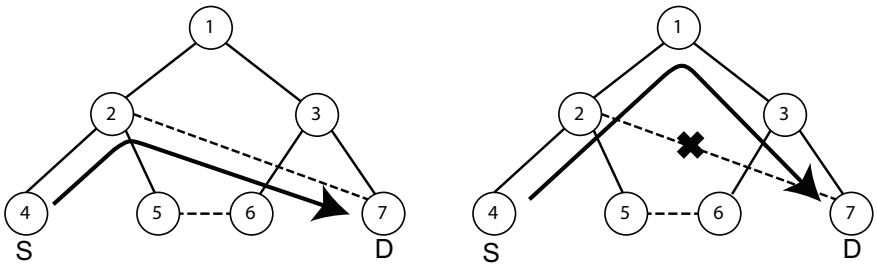


Figure 2.3: Example for failure in shortcut links. The left figure depicts the route before failure. On the right the route after failure is depicted.

stuck in a local minimum will only occur when a tree edge is faulty. Fig. 2.3 depicts a scenario in which a shortcut is faulty but as we see, greedy routing is possible using the tree edges. On the left, we observe that greedy route between node 4 and 7 is $4 \rightarrow 2 \rightarrow 7$. When the shortcut $2 - 7$ is faulty, the greedy route on the tree can be used ($4 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 7$) instead. So the recovery method is used only when greedy routing is no longer possible and that is when the faulty edge is in the spanning tree of the network.

The basic idea of the proposed method is that upon a tree link failure, we will tunnel to a node which greedy routing from that node to the destination will certainly work. In the following sub-sections, we describe the method in detail.

2.4.1.1 Upstream/downstream failure recovery

Upon a tree-edge failure, an alternate path is required. Finding such a path, we need to distinguish between upstream and downstream failures. Upstream failure refers to a situation when the faulty tree edge is needed to be passed upward (getting closer to the root). In downstream failure the faulty edge is required to be passed downward.[3]

The idea is that upon a failure detection, a search is performed in order to find an intermediate node from which greedy routing to the destination is possible. Distinguishing between upstream and downstream failure is required due to this intermediate node.

First, we consider a downstream failure scenario. We need to find a node which has a shortcut to the subtree[4] below the faulty link. This subtree is rooted at the higher level node attached to the faulty link. In an upstream failure scenario, it is enough that the node has a shortcut which can be used to go out of the subtree

[3] As we pass a link upward, we reach to a lower level node in the tree. When we pass a link downward, a higher level node is reached. The level of a node is defined as the distance from the root node in the tree. Root has level zero.

[4] A subset of nodes in the tree along with the corresponding edges. A subtree is a tree itself.

below the faulty link. Knowing these conditions for upstream and downstream intermediate nodes, the required steps in order to recover from the faulty link are as follows:

1. Initiate from the failure detecting node a Breadth First Search (BFS) polling for the closest potential intermediate node (with the aforementioned conditions for upstream/downstream failure).

2. When the potential intermediate node is found, the node sends an acknowledgment back to the failure detecting node indicating its coordinate.

There are a significant number of distributed algorithms in the literature to perform a BFS which directly or with a little change can be applied to our scenario [15, 16].

Once the intermediate node is found, the forwarding procedure is as follows:

1. From the failure detecting node greedy route to the intermediate node.

2. Use the shortcut of the intermediate node to go to the desired subtree.

3. Continue greedy routing to the destination on the spanning tree (disabling shortcuts).

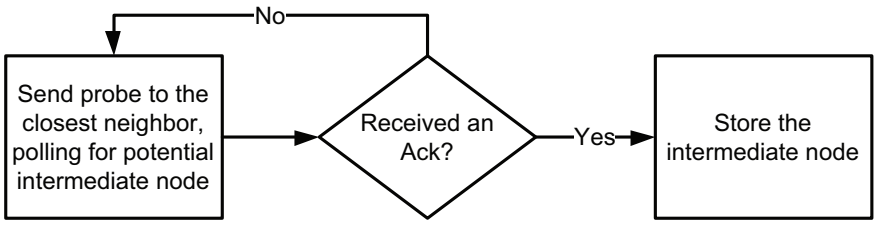
Using this method, there is no need to set any entries in the nodes which are along the alternate path because we first tunnel to the intermediate node and then continue greedy routing to the destination. The only overhead added to the header of the packet is the coordinate of the intermediate node and a flag. This flag indicates that the packet should be first greedy routed to the intermediate node and then continue routing on the tree.

The explained method can be used to find an alternate path on-demand upon detection of a failure or as a protection method (pro-active) in which the intermediate node is pre-computed.

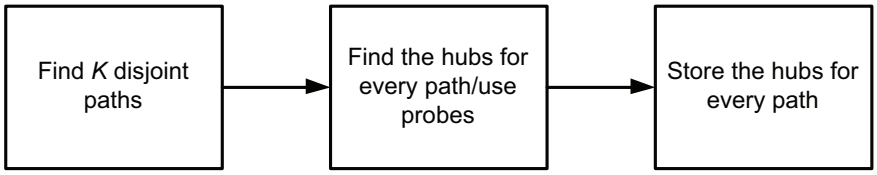
Considering the pro-active mode, the intermediate nodes corresponding to every tree edge are pre-computed in network nodes. Therefore, upon a failure detection packets towards a given destination can locally install the pre-computed backup entry. This enables very fast switch-over. As this recovery method is required only for the edges in the spanning tree of the network, it is scaling with the number of edges in the tree. Therefore, in the pro-active mode, every node in the network only needs to calculate the intermediate node corresponding to every tree edge which is attached to that node. Fig. 2.4a is the diagram for the control procedure, performed by a node for a tree edge attached to that node.

2.4.1.2 Subtree determination

In this section, we explain how to determine that a shortcut goes to a certain subtree.



(a) Diagram of recovery method for single link failure



(b) Diagram of recovery method for multiple link failures

Figure 2.4: Diagram of recovery methods for single and multiple link failures (control procedure)

In order to find the intermediate node to bypass the faulty link, it is required to know: (i) the level of a node in the tree (distance from the root node in the tree) and (ii) the subtree it belongs to. The level of nodes are first used for determination of upstream and downstream failures in order to figure out what kind of intermediate node should be looked for. The second use of node levels is for subtree determination (to which subtree a node belongs). Unfortunately, extracting this information from the virtual coordinates of nodes is not easy. Therefore, another numbering system is required for the network nodes. For this purpose, any numbering which determines the location of a node in the spanning tree can be used. We use a very simple numbering method in which an integer number is assigned to each node. The assignment can be done in a distributed way (parallel to the hyperbolic coordinate assignment). Knowing the degree of the tree, each node is able to determine the integer ID of its parent, its children and its level in the tree. A two-pass algorithm can be used to determine the degree of the tree. Every node reports to its parent the maximum degree of the subtree rooted at itself and then root calculates the maximum degree of the tree and broadcasts it to every node [4]. A very simple formula for calculation of integer IDs and the level of a node are as follows. In a k-ary tree (the maximum degree of each node is k+1):

$$parent(i) = \left\lfloor \frac{(k + i - 2)}{k} \right\rfloor$$

$$j^{th}child(i) = (k \times i) - (k - 2) + j$$

$S(h)$ denotes the number of nodes in a perfect k-ary tree of height h, which can be

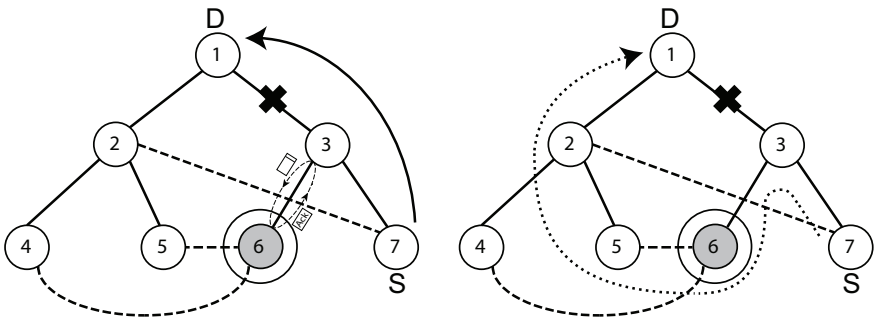


Figure 2.5: Example for upstream failure. On the left the intermediate node is determined. On the right the path after recovery is depicted by the dot line.

formulated as follows:

$$S(h) = \frac{k^{h+1} - 1}{k - 1}$$

The tree-level of a node can be determined using $S(h)$. For a node with integer ID of i, we need to find the h in the following inequality:

$$\frac{k^h - 1}{k - 1} < i \leq \frac{k^{h+1} - 1}{k - 1}$$

The first two formulas help us to find the immediate parent and children IDs of a node. However, in order to find out that a node is below a certain subtree (a subtree which is rooted at a certain node) the level of nodes should be calculated. As was explained earlier, the level of a node in the tree determines the distance of that node to the root of the tree. For example, we want to know if node A is in the subtree rooted at node B. First the levels of both nodes should be calculated (using $S(h)$). Assume that node A is in level 4 and node B is in level 1. In order to find out that node A is in the subtree of node B, we need to check if node B is a parent/ancestor of node A. This is performed by repetitively (level (A)-level (B) times) checking if the ID of the next parent of node A is equal to the ID of B. Therefore, in our example we need to apply the parent formula on the ID of node A for 3 $(4 - 1)$ times.

In Figs. 2.5 and 2.6 two simple examples for upstream and downstream failures are depicted. As we see in Fig. 2.5, node 3 upon failure detection, starts BFS polling for potential intermediate node. Node 6 has a shortcut which takes us out of the subtree below the faulty link (the subtree rooted at node 3, in this example it consists of nodes 3, 6 and 7). Therefore, node 6 sends an acknowledgment to the failure detecting node. In the right part of the figure, the alternative path using intermediate node is depicted. The packets are first greedy routed to the intermediate node (node 6), then the shortcut to the desired subtree is taken (link $6 - 5$), the rest of the routing should be continued on the tree ($5 \rightarrow 2 \rightarrow 1$). The

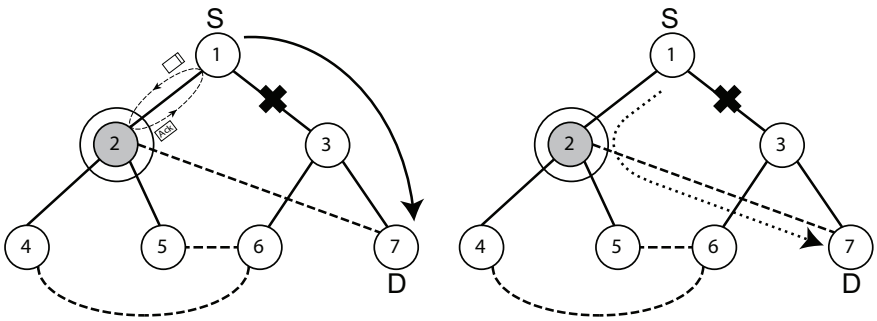


Figure 2.6: Example for downstream failure. On the left the intermediate node is determined. On the right the path after recovery is depicted by the dot line.

assigned integer IDs to the network nodes is based on a 2-ary tree (k=2). These IDs are only used to determine the location of nodes in the tree and the greedy routing is still based on the virtual coordinates in the hyperbolic plane. In Fig. 2.6, in the same way the intermediate node is determined (node 2) and the alternative path is depicted in the right part of the figure ($1 \rightarrow 2 \rightarrow 7$).

2.4.1.3 Exchanged packets

In order to have an overview on the number of packets that are exchanged in order to find the intermediate node and the complexity of these packets and the algorithms used, we explain the method from packet exchange perspective here. The failure detecting/protecting node starts a BFS search by sending a packet (probe) with the integer ID of the node attached to the other side of the faulty tree edge and the direction of the failure (upstream or downstream) to its first neighbor (illustrated in Figs. 2.5 and 2.6). Upon receiving such a packet, a node can check whether it has a suitable shortcut or not. If not, the next node in a BFS manner is checked. When the intermediate node is found, the node sends an acknowledgment to the failure detecting/protecting node. This acknowledgment contains the hyperbolic coordinate of the intermediate node. In the experimental evaluation, we observe the required number of communication messages in order to find the intermediate node corresponding to tree edges in the network.

2.4.1.4 Correctness of the method

The proof presented in this section is applied to the embedded graph in the hyperbolic plane. The integer IDs in the examples/figures are the numbering explained in Section 2.4.1.2 and are not used for greedy routing.

Using this method, the alternative path does not consist of the faulty link and it reaches the destination.

Proof. Let $G(V, E)$ be a biconnected graph which is embedded in the hyperbolic plane H using the greedy embedding $f : V(G) \rightarrow H$ described in Section 2.3. T is the spanning tree of G. s and d are two nodes in G which are joined by a path $s = s_0, s_1, ...s_i, s_{i+1}, ..., s_k = d$ using greedy routing. Let s_i, s_{i+1} be the faulty link which is also a tree edge. Based on the assumptions, the removal of s_i, s_{i+1} link should not cause the network to be disconnected. Therefore, a node with a shortcut to the desired subtree (intermediate node) will definitely be found. In upstream failure scenario, this node is located in the subtree below the faulty link and in downstream failure scenario, it is located out of the subtree below the faulty link. In both cases, the path on the tree (T) connects the failure detecting node to the intermediate node. This means that there exists at least one distance-decreasing path via the tree and therefore, the greedy routing between these two nodes is guaranteed.

In a downstream failure scenario, the packets that need to pass the faulty link are definitely destined to the nodes which are located in the subtree below the faulty link. Otherwise, the greedy routing would not lead the packets to go into that subtree. Knowing this, when the packets arrive at the intermediate node and then take the shortcut, they end up in the subtree below the faulty link. This subtree does not consist of the faulty link therefore, there is at least one path between every two nodes of the subtree via that subtree. As the packets' destinations are located in this subtree, greedy routing on the tree will definitely leads to them. In this part of the routing, the shortcuts are disabled due to the fact that they might lead the packet to a node which is out of the desired subtree, causing the packet to get stuck in a loop.

The same can be explained for an upstream failure scenario. The packets are destined to the nodes which are located out of the subtree below the faulty link. Therefore, upon arrival to the intermediate node and taking the shortcut, they end up out of the subtree below the faulty link. Routing first to the intermediate node and then to the destination is exactly the same as the downstream failure scenario.

The path after recovery will be $s = s_0, s_1, ..., s_i, s'_1, s'_2, ..., t_1, t_2, ..., t_k = s_k = d$. s_is are the nodes on the path before reaching the faulty link. s'_is are the nodes which are passed to reach the intermediate node and t_is are representing the path on the tree. We can observe this in the previous examples. Consider Fig. 2.5. The final path is $7 \rightarrow 3 \rightarrow 6 \rightarrow 5 \rightarrow 2 \rightarrow 1$. As we see, some nodes are s_i (7 and 3), some are s'_i (6) and some are t_i (5, 2 and 1).

2.4.2 Multiple link failures

The recovery method used for a single link failure cannot be applied to the multiple failure scenarios. The reason is that the information about all faulty links in the network might be required in every node. This information is used in order to find

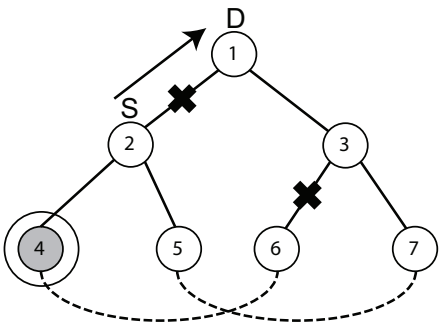


Figure 2.7: Problem with multiple link failures using single link failure recovery method.

a potential intermediate node which does not lead to a subtree with a faulty link. Considering large-scale networks, providing such information in all the network nodes is not feasible and realistic. The problem of not having such information is illustrated in Fig. 2.7. Based on the method for single link failure, node 2 starts searching for the potential intermediate node and node 4 is a suitable choice (has a shortcut out of the subtree below the faulty link). But if the shortcut $4-6$ is taken, routing cannot be continued on the tree. Therefore, the list of faulty links was required in node 4 in order to avoid considering shortcut $4-6$ as a suitable one and node 5 was chosen as the potential intermediate node.

In this section, we propose another method in order to recover multiple link failures in the network.

2.4.2.1 Multiple link failure recovery

Similar to the previous explanations, only the tree-edge failures require recovery.

In scenarios with multiple link failures, the proposed scheme recovers every faulty edge locally. This means that if the next hop of a packet is a faulty tree edge, a backup path to the node attached to the other side of the faulty link is used. In the failure detecting node, packets use the backup path to reach the node on the other side of the faulty link and then from that node greedy routing to the destination is resumed. Upon facing another faulty tree edge along the path to the destination, the same procedure can be repeated. Using this method, the faulty edges are recovered locally and unlike the previous scheme, there is no need to distinguish between upstream and downstream failures.

The proposed scheme is a recovery method in the pro-active mode. There is always a trade-off of speed and resources between the methods in pro-active and re-active mode. Using this scheme, we could have a fast recovery in the cost of more resources.

As the recovery is only required for the tree edges in the network, every node

should find the backup path only for the tree edges which are connected to that node. In the pro-active mode, backup paths are pre-computed before a failure. Therefore, in order to recover multiple failures, multiple disjoint paths are calculated. The maximum available disjoint paths between two nodes determines the maximum number of simultaneous failures that is guaranteed to be recovered.

In the proposed method, every node should find the maximum disjoint paths between itself and its neighbor if the link between them is in the spanning tree of the network. These disjoint paths should be found for every tree edge which is connected to that node. However, based on the topology, the number of possible disjoint paths might be different for every tree edge. Therefore, the minimum number among all possible disjoint paths for all the tree edges (minimum cut of the network) is the maximum number of simultaneous failures that is guaranteed to be recovered using this method. Studies have been performed to find all the disjoint paths between two nodes in a network. [17] has proposed a distributed algorithm to find k node-disjoint paths between two nodes in a network which can be adapted to find all link-disjoint paths as well. Their algorithm runs in $O(kn)$ time using $O(km)$ messages where n is the number of nodes and m is the number of links in the network.

A problem that might appear when considering multiple disjoint paths, is that storing an entry in the intermediate nodes along every disjoint path causes a large memory overhead in those nodes. In order to avoid such memory overhead in the nodes, we take benefit of greedy routing. Instead of setting an entry in every node along the backup paths, we check if we can go through the backup paths using greedy routing. This means that once the disjoint paths are found, we start checking each of them to see up to which node in the path, greedy routing path is equal to the backup path. We consider that node as a hub node. This procedure is repeated from the hub node for the rest of the nodes along the backup path. At the end of this checking, we find some hub nodes which greedy routing between them is equal to the backup path. In Section 2.4.2.2, we explain how these hub nodes can be found using some probe messages. Every protecting node only needs to store these hub nodes for the different disjoint paths it calculated for every tree edge. None of the nodes along the backup paths require storing any entry for the paths that go through them. Fig. 2.4b depicts the steps of the control procedure in a network node considering one tree edge.

If there exists at maximum K disjoint paths in every node towards the next hop and every disjoint path maximally has L hubs and the maximum degree in every node is M then, the resulting memory required in every node is at maximum $M \times K \times L$. Therefore, the required memory in nodes is dependent on the number of failures that is guaranteed to be recovered and also the degree of nodes in the spanning tree of the network. We will see the experimental results for the required memory and the degree distribution of nodes in the spanning tree of a network in

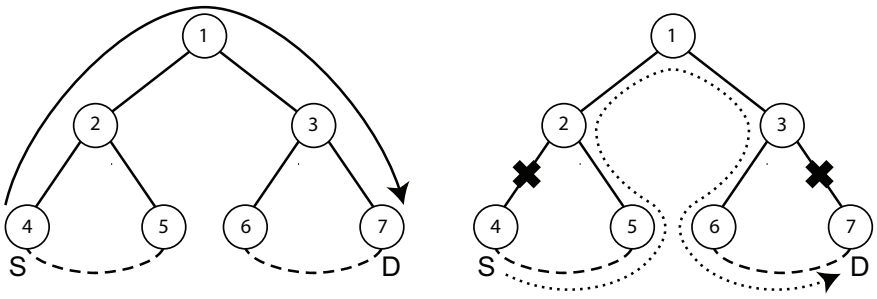


Figure 2.8: Example for multiple link failures. On the right the path after recovery is depicted by the dot line.

Section 2.5.2.2.

Finding the backup paths for every tree edge and the hub nodes corresponding to them and storing in the protecting node are the steps of control procedures. In the next paragraph, the forwarding procedure is explained.

Protocols similar to BFD [14] can be used in order to test the liveness of the backup paths of a tree edge. As this is a $K:1$ protection, there is no need to have all the K backups active. Once the next hop of a packet is a faulty tree edge, the hub nodes corresponding to one of the disjoint paths for that edge is added to the header of the packet. The packet is first greedy routed to the first hub then to the second hub and so on until it reaches to the node on the other side of the faulty edge. From there, greedy routing to the destination can be continued. The results in Section 2.5.2.2 show that this method has a very low overhead to the header of the packets. Fig. 2.8 depicts an example for multiple tree-edge failures and the path after recovery (using backup path corresponding to a faulty edge). The correctness of the scheme can be proved by considering that the scheme actually follows the greedy route to the destination. Although, in case of facing a faulty tree edge a backup path corresponding to that edge is used instead. The scheme is successful as long as there exist at least one backup path for every tree edge in the network.

2.4.2.2 Finding hub nodes in a backup path

In this section, the required steps in order to find the hub nodes in a path are explained in detail.

We refer to a node attached to a link under protection as a protecting node. The hub nodes corresponding to a backup path are stored in this node. These hubs can be found using probe messages. The protecting node starts the probing. Every node along the path has an entry to the next hop. However, once the hub nodes are found there is no need to keep these entries. Two types of probe messages are

used. The first type is used to check if greedy routing to a node along the path gives the same path as the backup path (Grdy-Chk message). The second type is used to inform a node along the path to start the same procedure for the rest of the nodes (Start-Chk message).

The procedure starts by the protecting node which sends a Grdy-Chk message to the first node along the path to see if greedy routing is equal to the backup path. Every time the Grdy-Chk reaches the specified node in the destination field of the message, that node sends back an acknowledgment to the protecting node. This acknowledgment contains the next hop along the path. The protecting node always stores the last node which sent back an acknowledgment and sends another Grdy-Chk with the next node along the path as the destination of the message. The procedure continues up to the point that greedy routing and the backup path are not the same. The first node that detects this issue sends back a NotAck to the protecting node. Upon arrival of such a message, the protecting node keeps the previously stored node as the first hub and sends a Start-Chk message to that hub node. Upon arrival of such message in the hub node, it starts sending Grdy-Chk messages to the rest of the nodes along the path. The hub node continues checking, up to the point that it receives a NotAck. In this case, it sends back the last stored node to the protecting node. Protecting node stores the second hub and continues the procedure up to the point that all the hub nodes are found.

2.4.3 Comparison with tree-oriented routing algorithms

In this section, we explain the major differences between the proposed schemes and the well-known tree-oriented algorithms such as Ethernet Spanning Tree Protocol (STP), IEEE 802.1D standard and its variant Rapid Spanning Tree Protocol (RSTP), IEEE 802.1w. These protocols ensure a loop-free topology in a network. RSTP provides significantly faster spanning tree convergence after a topology change.

STP and RSTP algorithms dynamically calculate a tree in a distributed way. Upon every topology change, protocol messages are exchanged between network nodes and the tree is re-converged. However, the proposed schemes in this paper calculate only one spanning tree and install backups on demand or by protection. As the methods recover failures locally, there is no need to inform all the network nodes about the changes.

The other difference is the routing itself. STP and RSTP route only on the spanning tree of the network (tree routing). However, as explained in this paper, greedy routing using virtual coordinates is not equal to tree routing. The shortcuts are also used which makes the routing much more efficient than routing only on the tree. [4] compares the performance of greedy routing and routing on the spanning tree in terms of stretch.

2.5 Evaluation of the recovery method

Our simulation environment for the evaluation of the proposed algorithms is based on Python/C++. The Networkx-library is used for the implementation of Graph-based algorithms, and Numpy and Scipy packages are used for numerical algorithms. Some parts of the code were optimized in C++ to allow for large scale network simulations.

The proposed schemes were evaluated on representative network topologies similar to the Internet. These graphs were evaluated at different scales up to the networks with 1000 nodes. This class of network topologies is based on Barabasi-Albert (B-A) model [18]. The authors of the latter have defined a method of randomly generating scale-free[5] networks using a preferential attachment mechanism. The latter results into the fact that the more connected a node is, the more likely it is to receive new links, resulting into a power-law degree distribution with γ equal to 3. We repeated every experiment over 20 different generated B-A graphs although, all of the graphs have very similar characteristics (degree distribution, number of edges, ...). Therefore, every value on the result graphs is the average value over these different networks.

Our experimental evaluation consists of two sets of results. One set is related to the single link failure recovery method and the other set is related to the multiple link failure recovery method. The proposed schemes are evaluated in terms of routing quality (stretch) and overhead.

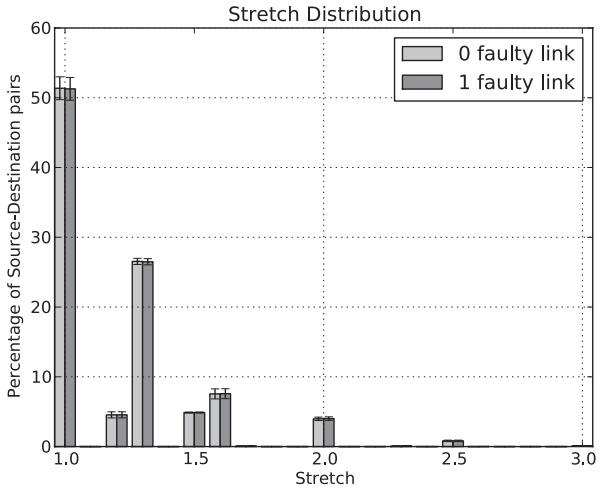
2.5.1 Experimental results for single link failure recovery method

2.5.1.1 Stretch evaluation

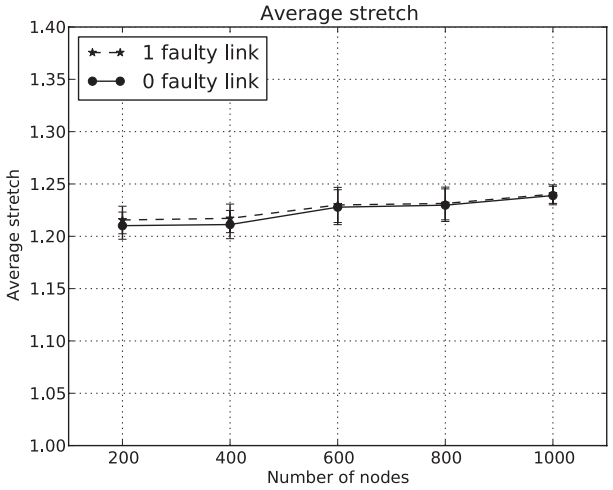
In this section, we evaluate the routing quality (i.e. stretch) of the proposed scheme. The stretch is defined as the ratio of the length of the path as produced by the greedy routing scheme, to the shortest path length for the same source-destination pair. We compare the stretch for two scenarios. In one case, there is no faulty link in the network and the stretch represents the ratio of greedy routing path length to the shortest path length. In the second scenario, there is one faulty tree-edge in the network. Therefore, the stretch represents the ratio between the path length generated by greedy routing with the recovery method to the shortest path length after removing the faulty edge from the graph.

Every graph is embedded in the hyperbolic plane using the method in Section 2.3. Therefore, these virtual coordinates are used in order to perform the greedy routing. For the scenario with one single failure in the network, we evaluate the

[5] A scale-free network is a network whose degree distribution follows a power law, at least asymptotically. That is, the fraction $P(k)$ of nodes in the network having k connections to other nodes goes for large values of k as $P(k) \approx c.k^{-\gamma}$. The value of γ is typically in the range $2 < \gamma < 3$.



(a) Stretch distribution for single link failure scenario. Network size=1000 nodes.



(b) Average stretch for single link failure scenario for different network sizes.

Figure 2.9: Stretch evaluation for single link failure recovery method

stretch considering every tree-edge, one at a time as a failing edge and take the average over all of the results. As was mentioned, this procedure is repeated for different B-A networks and the values in the graphs are averaged over all of the samples.

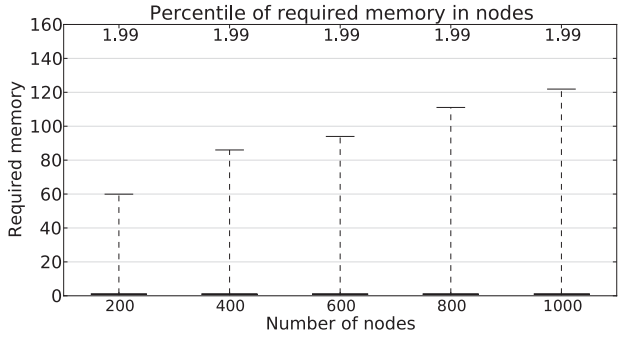
In Fig. 2.9a, the stretch distribution for graphs with 1000 nodes is illustrated and we observe that the stretch distribution upon a single link failure is almost the same as the stretch distribution without any link failure and the recovery method would not change the stretch significantly. In Fig. 2.9b, the average stretch for the two scenarios over networks with different sizes is depicted. For this experiment, the stretch of all possible source-destination pairs in a network is calculated and then averaged. In both Fig. 2.9a and Fig. 2.9b, 95% confidence intervals are depicted. We see that the scheme scales very well with the increase in the number of nodes as the change in the average stretch for B-A graphs is insignificant. Observing this trend enables us to extrapolate the resulting performance up to large-scale topologies.

2.5.1.2 Overhead evaluation

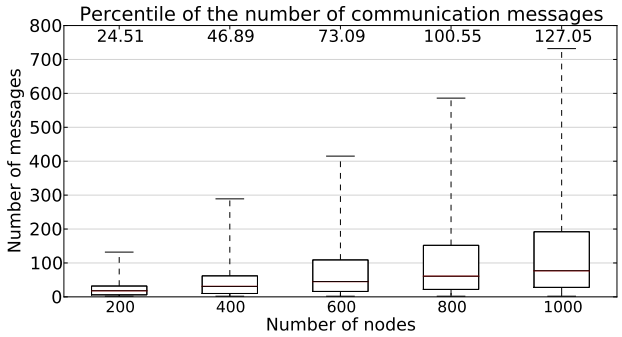
The overhead of the proposed scheme in terms of required memory in the network nodes and the number of communication messages were evaluated.

As explained in previous sections, the proposed methods are only required for the failures in the tree edges. Therefore, they scale with the number of edges in the spanning tree of the network. In the single failure recovery method, every node A needs to store an intermediate node B for every tree link attached to node A. Therefore, the degree of each node in the spanning tree of the network is equal to the required memory in that node. We calculated the degree of each node in the spanning tree and depicted the percentile using box-plot chart to have an overview of the distribution of degree/required memory in network nodes. Fig. 2.10a illustrates the results for networks with different sizes. The boxplot was used in order to depict the minimum value, lower quartile, median, upper quartile and the maximum value. However, we only see two values in the resulting graphs. The reason for this representation is that up to 75% of the nodes have the degree of one in the tree, but there exists some nodes with a very high degree. This can be explained from the properties of B-A graphs. In the generation of these graphs the more connected a node is, the more likely it is to receive new links which results into some nodes having very high degrees. As we used BFS to generate the spanning tree and chose the node with maximum degree as the root of the tree therefore, there exists a few nodes with very high degree and lots of other nodes with degree one. The values on top of the graph are the average required memory in the network nodes.

Our next experiment evaluates the required number of communication messages in order to find the potential intermediate node corresponding to a tree edge.



(a) Percentile of the required memory in network nodes



(b) Percentile of the number of communication messages for all the tree edges in the network

Figure 2.10: Overhead evaluation for single link failure recovery method.

These messages include the probes and the acknowledgment. As explained in Section 2.4.1.3, a node starts a BFS using probe messages and when the intermediate node is found it receives an acknowledgment. We calculated the number of these messages for every tree edge in the network and depicted the percentile of them in Fig. 2.10b. The results are for networks with different sizes up to 1000 nodes. The values on top represent the average number of required messages. In the networks of 1000 nodes, for up to 75% of tree edges, less than 200 messages were exchanged. The maximum number of exchanged messages was 732 and on average 127 messages were required.

In order to have an overview of the distance between the intermediate node and the failure detecting/protecting node, we calculated this distance (in hops) for every tree edge in the network. We only report the results for the networks with 1000 nodes. Considering all the tree edges in the network, the intermediate node for 75% of links located only two hops away from the failure detecting node and

the maximum distance was 4. The distance on average was 2 for all the tree edges.

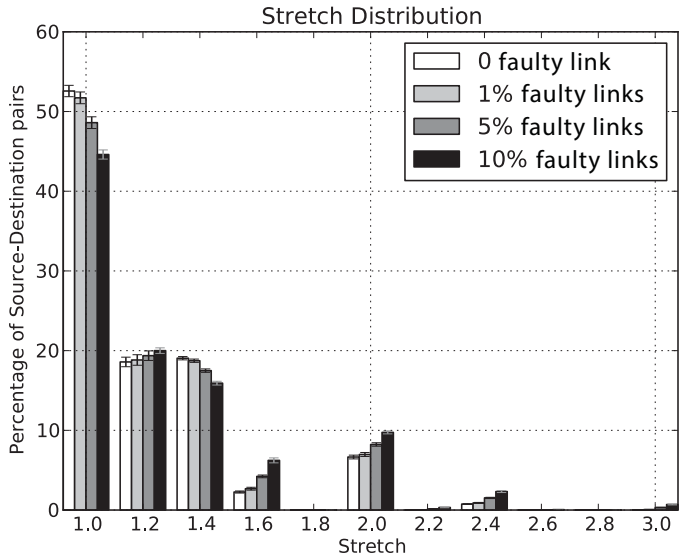
2.5.2 Experimental results for multiple link failure recovery method

2.5.2.1 Stretch evaluation

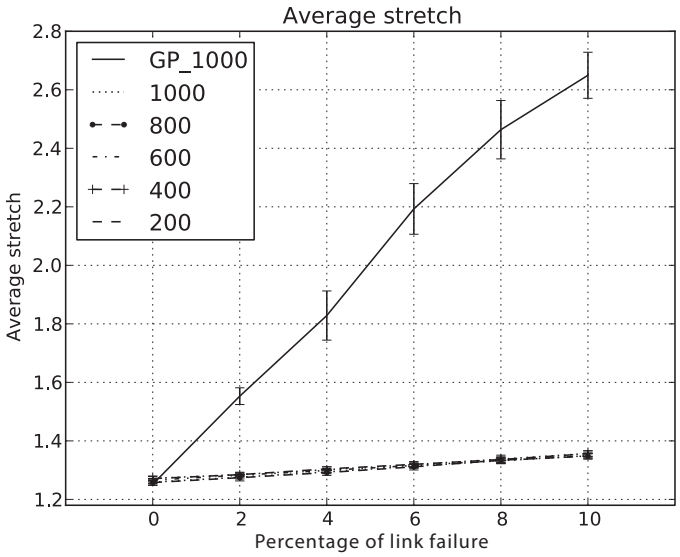
The quality of the proposed scheme for multiple failure scenarios was evaluated. In this section, we present similar graphs as in Section 2.5.1.1 for stretch distribution and average stretch in networks with different number of nodes. The proposed method guarantees the recovery of a given number of failures (at most equal to the minimum cut of the network) however, the method is successful as long as the protection mechanism is not broken. This means that there is at least one backup path for every tree edge in the network.

For this experiment, we generated B-A graphs with minimum degree of 10 and we could consider 9 disjoint paths for every tree edge in the network. Therefore, the maximum number of failures that the method could guarantee to recover was 9. However, we evaluated our method for scenarios with more number of failures than only 9 to have an overview of the performance of the method in occurrence of more failures. In Fig. 2.11a, the stretch distribution for networks of 1000 nodes in 4 different scenarios is depicted. In every scenario, different percentages of failures in tree edges were examined. The faulty tree edges were chosen randomly (uniform distribution). As expected, with more failures in the network the pairs of nodes with higher stretch increased. However, looking at the results in Fig. 2.11b the average stretch does not exceed 1.4. In order to compare the performance of the proposed scheme with the existing methods, we implemented Gravity Pressure algorithm [12] based on the pseudocode available in the paper. We chose this algorithm due to the fact that it can be applied to the greedy embedding in the hyperbolic plane and unlike other methods it can be applied to every topology without any restriction (like planarization for using face routing). The average stretch for networks of 1000 nodes with different percentages of tree-edge failures using this algorithm is depicted in Fig. 2.11b. As the authors explain in [12], using this method, the average stretch increases with more failures in the network up to the point that the network is sparse. Therefore, after certain amount of failures the average stretch decreases. As failure of 10% of tree-edges does not make the network sparse, Gravity Pressure might not always find the most efficient path. Therefore, as we see in the results, with more failures, the average stretch increases up to 2.6. In the graphs of Fig. 2.11, the 95% confidence interval of the values are depicted. In Fig. 2.11b, the width of the confidence intervals of the values for the proposed scheme is not more than 0.02.

As mentioned earlier, our recovery method is successful up to the point that the recovery mechanism breaks. This means that for some of the tree-edges all

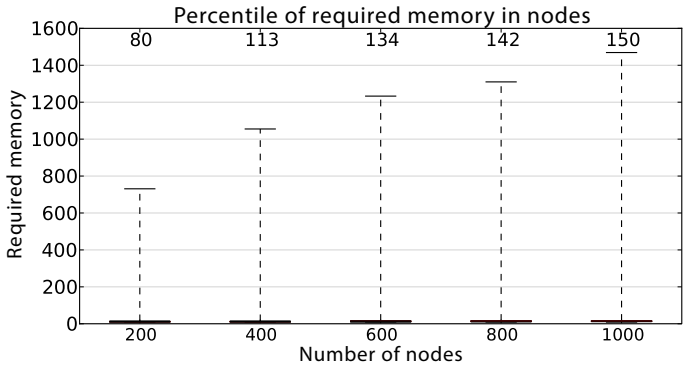


(a) Stretch distribution for multiple link failure scenario. Network size=1000 nodes.

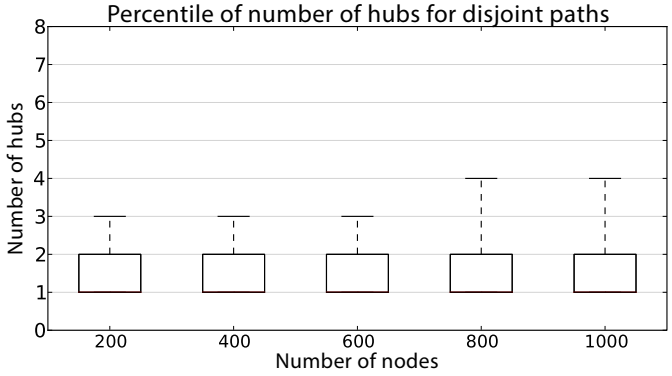


(b) Average stretch for multiple link failure scenario for different network sizes.

Figure 2.11: Stretch evaluation for multiple link failure recovery method.



(a) Percentile of the required memory in network nodes



(b) Percentile of number of hubs in the disjoint paths of all the tree edge

Figure 2.12: Overhead evaluation for multiple link failure recovery method.

the backup paths are broken. We evaluated different scenarios with different percentages of failures in the tree-edges and checked how many pairs of source and destination are disconnected. Different scenarios of random failures up to 50% of faults in tree-edges in networks of 1000 nodes were considered. In all the scenarios up to 30% of failures, the method was successful. However, in some scenarios of 40% and 50% of failures, around 2600 pairs were disconnected on average which is less than one percent of all possible pairs in the network.

2.5.2.2 Overhead evaluation

As this method is applied only for the tree edges in the network therefore, the required memory in every node is proportional to the degree of that node in the tree. As explained in Section 2.4.2.1, every node requires to store the hub nodes

for every disjoint path corresponding to the tree edge that is attached to that node. In our experiment, the number of disjoint paths for every tree edge is 9. Based on the degree distribution in the spanning tree of the B-A networks, there are a few nodes with very high degree and also lots of nodes with degree one. Having this in mind, the required memory in network nodes is depicted in Fig. 2.12a. The minimum value is 9 and in almost all the networks, 75% of the values is 16. The maximum value reaches up to 1470 for networks of 1000 nodes and that is due to the existence of those few nodes with very high degree. The maximum degree of a node in the spanning tree is depicted on top of the graph.

The last experiment evaluates the number of hubs for disjoint paths corresponding to every tree edge. We found 9 disjoint paths for every tree edge in the network. For each of them the number of hubs was calculated. Fig. 2.12b illustrates the percentile of the number of hubs for all the disjoint paths in networks with different sizes. As we observe, 75% of paths have only 2 hubs and the maximum number of hubs is 4 in networks of 1000 nodes. The hubs for a backup path are added to the header of a packet. As Gravity Pressure also uses a path trace in each packet, we calculated the size of the table stored in every packet which is in the Pressure mode of the algorithm in order to have a comparison with the proposed scheme. The percentile of the size of the table in the packets in Pressure mode is as follows, 25% of the packets have the table of size 3, the median is 9 while 75% of packets have the table of size 28 and the maximum value is 290 for networks with 1000 nodes and 10% of faulty tree-edges. On average the size of this table is 24.

2.6 Conclusion

In this paper recovery techniques for single and multiple link failures in greedy routing were proposed. Both techniques avoid the lakes or voids in the first place. It is not required to maintain the list of local minima in the network in order to have a successful delivery to a destination. The proposed schemes do not need coordinate re-calculation, guarantee a given link failure coverage, and can be used as a protection scheme, enabling faster switch-over than existing alternatives. Our methods only require local communications. Both recovery methods scale with the number of edges in the spanning tree of the network. These methods were evaluated in an experimental environment on graphs with properties similar to the Internet. The used recovery paths demonstrated interesting stretch characteristics compared to existing dynamic alternatives. Other experiments gave an overview on the overhead of the methods. Based on the results, the required memory in every node was proportional to the degree of the node and the number of failures that were guaranteed to be recovered. The overhead added to the header of the packet was very low. Scalability, simplicity and limited resources make the schemes suit-

able for large-scale networks. Locality, limited loss of routing quality, and low overhead make the proposed schemes as promising recovery techniques.

Acknowledgment

This work is partly funded by the European Commission through the EULER project (Grant 258307), part of the Future Internet Research and Experimentation (FIRE) objective of the Seventh Framework Programme (FP7).

References

[1] G. Huston. *BGP Routing Table Reports*, 2011. http://bgp.potaroo.net/.

[2] B. Karp and H. Kung. *GPSR: greedy perimeter stateless routing for wireless networks*. In Proceedings of the 6th annual international conference on Mobile computing and networking, pages 243–254. ACM, 2000.

[3] C. Papadimitriou and D. Ratajczak. *On a conjecture related to geometric routing*. Theoretical Computer Science, 344(1):3–14, 2005.

[4] R. Kleinberg. *Geographic routing using hyperbolic space*. In INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, pages 1902–1909, 2007.

[5] R. Fonseca, S. Ratnasamy, J. Zhao, C. Ee, D. Culler, S. Shenker, and I. Stoica. *Beacon vector routing: Scalable point-to-point routing in wireless sensornets*. In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2, pages 329–342. USENIX Association, 2005.

[6] D. Chen and P. Varshney. *A survey of void handling techniques for geographic routing in wireless networks*. IEEE Communications Surveys and Tutorials, 9(1):50–67, 2007.

[7] D. Eppstein and M. Goodrich. *Succinct Greedy Geometric Routing Using Hyperbolic Geometry*. IEEE Transactions on Computers, 60(11):1571–1580, 2011.

[8] R. Flury, S. Pemmaraju, and R. Wattenhofer. *Greedy routing with bounded stretch*. In INFOCOM 2009, IEEE, pages 1737–1745, 2009.

[9] C. Westphal and G. Pei. *Scalable routing via greedy embedding*. In INFOCOM 2009, IEEE, pages 2826–2830, 2009.

[10] M. Boguna, F. Papadopoulos, and D. Krioukov. *Sustaining the Internet with hyperbolic mapping*. Nature Communications, 1(6):1–8, 2010.

[11] F. Papadopoulos, D. Krioukov, M. Bogua, and A. Vahdat. *Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces*. In INFOCOM 2010, IEEE, pages 1–9, 2010.

[12] A. Cvetkovski and M. Crovella. *Hyperbolic embedding and routing for dynamic graphs*. In INFOCOM 2009, IEEE, pages 1647–1655, 2009.

[13] J. Vasseur, M. Pickavet, and P. Demeester. *Network recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Morgan Kaufmann Publishers, 2004.

[14] D. Katz and D. Ward. *Bidirectional forwarding detection (BFD)*. RFC5880, 2010.

[15] S. Makki. *Efficient distributed breadth-first search algorithm*. Computer Communications, 19(8):628–636, 1996.

[16] B. Awerbuch and R. Gallager. *A new distributed algorithm to find breadth first search trees*. IEEE Transactions on Information Theory, 33(3):315–322, 1987.

[17] S. Arora, H. Lee, and R. Thurimella. *Algorithms for finding disjoint paths in mobile networks*, 2003.

[18] A. Barabási and R. Albert. *Emergence of scaling in random networks*. Science, 286(5439):509–511, 1999.

3

Experimental validation of resilient tree-based greedy geometric routing

In the previous chapter, different mechanisms were proposed to provide failure recovery in geometric routing. Hyperbolic coordinates were used to perform greedy routing in the network. Instead of exploiting such complex coordinates, we propose a simple mechanism to assign coordinates based on a spanning tree of the network. This chapter proposes a distributed algorithm to calculate these coordinates. The proposed recovery mechanisms in the previous chapter provide a fault-tolerant geometric routing scheme. However, they rely on fixed pre-defined recovery paths which makes them quite static approaches. As a result, these schemes may lead to performance loss in very dynamic networks. Therefore, an adaptive and a more dynamic mechanism to deal with different types of network changes (e.g., link/node failure/addition) is necessary. To this end, the distributed algorithm is extended with a mechanism to trigger coordinate re-calculation upon network changes. A software prototype of the proposed scheme is developed and its performance is evaluated on a testbed on the iLab.t virtual wall[1] *at iMinds. Different trends and trade-offs in the recovery domain of geometric routing are identified.*

⋆ ⋆ ⋆

S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, P. Demeester

[1] http://ilabt.iminds.be/iminds-virtualwall-overview

Published in Computer Networks, 2015.

Abstract Geometric routing is an alternative for IP routing based on longest prefix matching. Using this routing paradigm, every node in the network is assigned a coordinate and packets are forwarded towards their intended destination following a distance-decreasing policy (greedy forwarding). This approach makes the routers significantly more memory-efficient compared to the current IP routers. In this routing, greedy embeddings are used to guarantee a 100% successful delivery to every destination in the network. Most of the existing proposals lack resiliency mechanisms to react efficiently to network changes. We propose a distributed algorithm to calculate a greedy embedding based on a spanning tree of the network. In this algorithm, nodes are triggered to re-calculate their coordinates upon a change in the topology such as link or node failures. The advantage of this approach is that it recovers from topology failures within a very short period of time. We further extend the algorithm to generate backups to apply protection in distributed setups. Different trade-offs and trends of re-convergence for geometric routing have been evaluated in an emulation environment. Realistic results are achieved through emulation as no model or abstraction is involved. The proposed routing scheme is implemented in Quagga routing software and new elements are developed in Click modular router to enable greedy forwarding. For the first time, the performance of this scheme is evaluated through emulation on a large topology of 1000 nodes and the results are compared with BGP. The experimental results indicate that the proposed scheme has interesting characteristics in terms of convergence time upon a change in the network topology.

3.1 Introduction

Increasing numbers of components are interconnected via communication networks such as the Internet. In such large networks, efficient routing becomes more and more important. Efficient routing schemes require low memory in network nodes and act rapidly upon network failures with minimal global impact and short convergence time.

Geometric routing is a more recent paradigm which has been proposed as an alternative to IP routing [1]. Scalability of routing tables in conventional IP routers based on longest prefix matching is becoming an issue in the near future ([2] reports more than 500 K FIB entries in current BGP routers). In geometric routing, network nodes are assigned (virtual) coordinates in a metric space. Upon arrival of a packet, the distance between every neighboring node and the packet's destination is calculated based on the coordinates. The neighbor with the maximal decrease in the distance is selected as the next hop. Following this distance-decreasing policy can lead the packets to their intended destination. As only local information is

required in every node, it is considered to be significantly more memory-efficient compared to the conventional IP routing. This scheme is referred to as greedy routing/forwarding because in each step, the neighboring node with maximum decrease in the distance is selected.

A known issue in greedy routing is that packets might get stuck in local minima (void) meaning that the current location is the closest to the destination among all the neighboring nodes. A possible solution which is also considered in this work is to use greedy embeddings [3]. Greedy embeddings find mapping between nodes and coordinates in such a way that there is always a distance-decreasing neighbor towards any destination in the network topology. Greedy routing based on these embeddings never gives rise to local minima. In [4], we proposed a simple greedy embedding based on a spanning tree of the network.

3.1.1 Our contribution

In this paper, a distributed algorithm for calculation of the embedding in [4] is proposed and experimentally validated. We extend this algorithm to generate backups according to one of the protection techniques we proposed in [5]. Although, this technique provides a fault-tolerant routing scheme, a mechanism for full convergence from different types of network dynamics such as failure and addition is required to minimize the impact of topology changes on routing quality (e.g. stretch[2]). Therefore, in the proposed distributed algorithm, nodes are triggered to re-calculate their coordinates in case the structure on which the greedy embedding is built (a spanning tree in our case) is broken. Any type of failure (link/node) can be recovered as long as the network remains connected and hence a spanning tree can be re-constructed.

Importantly, this paper is different from the previous works in the sense that: (i) a distributed algorithm and relative prototype implementation are provided and (ii) all the evaluations are through emulation and different metrics such as convergence time, protection time and communication cost are evaluated while in [4, 5] we only evaluated the stretch and memory overhead of the schemes.

Although geometric routing is very memory scalable, up to now, it was never considered as a realistic option for large-scale networks due to the lack of good recovery mechanisms and it was only evaluated in simulation environments. This paper builds further on the recent developments in making geometric routing more robust, extends the state-of-the-art design, and implements a prototype in an emulation environment in order to validate feasibility of the resulting scheme for larger networks. Although it is too early to propose the resulting geometric routing as a BGP alternative, we compared a prototype of geometric routing with BGP (as it is the only routing protocol which has been actually applied in large-scale networks)

[2]The ratio between the length of the greedy path and the shortest path.

to identify routing performance trends and comparisons in terms of convergence time, stretch and adaptability.

We implemented the control plane of geometric routing i.e. the proposed distributed algorithm in Quagga routing software.[3] The forwarding plane was implemented as new elements in Click modular router[4] to enable greedy forwarding of the packets. The experimentation covers the evaluation of both protection scheme and the full convergence (coordinate re-calculation). The iLab.t virtual wall facility is used as the test environment.[5] To the best of our knowledge this is one of the first works to perform emulation experiments on large topologies of 1000 nodes. This scale is a factor 10 improvement over state-of-the-art emulation experiments. As no abstraction is involved in emulation, high accuracy and sensitivity is achieved in the results. Experimental results in this paper show that the convergence time/communication cost of the proposed scheme upon different types of dynamics in the topology is very low.

The rest of the paper is structured as follows: Section 3.2 explains the related work. The tree-based greedy embedding and the protection technique together with the proposed distributed algorithm are described in Section 3.3. Section 3.4 includes the routing platform description and the implementation of the proposed scheme in Quagga and Click software. The experimental scenarios, results and their analysis are reported in Section 3.5. Finally, Section 3.6 concludes the paper.

3.2 Related work

Steps towards greedy routing start with the idea of geographic routing proposed by Karp and Kung [6] in which network nodes receive their GPS coordinates. Fonseca et al. used the same idea but instead of physical location of the nodes, a virtual coordinate in a metric space was assigned to each node in the network leading to the term geometric routing [1].

Two groups of techniques were proposed to solve the issue of the local minimum in greedy routing. In the first group which is referred to as face routing, the void is passed by routing around this area and greedy routing is resumed from the node which is closer to the destination than the local minimum [7]. This type of technique suffers from two limitations: (i) local parts of the graph must be planar (planarized), (ii) the latter might not be possible for all types of graphs. In the second group, greedy embeddings are used to avoid local minima. Using these embeddings, 100% successful delivery to every destination is guaranteed. Kleinberg proved that it is possible to find such an embedding in a two dimensional hyperbolic plane for any connected finite graph using a spanning tree of the net-

[3] http://www.nongnu.org/quagga/
[4] http://read.cs.ucla.edu/click/click
[5] http://www.iminds.be/en/succeed-with-digital-research/technical-testing

work [8]. Authors of [9] proposed another greedy embedding in the hyperbolic plane which requires $O(logn)$ bits to represent the coordinates (n is the number of the network nodes). In [10], graphs are embedded into a $O(log^2 n)$-dimension Euclidean space while in [11], it is a poly-logarithmic dimension Euclidean space.

In [12, 13] and references therein Delaunay triangulations are used to enable greedy forwarding focusing on 3-dimensional wireless networks in Euclidean space. In particular, it has been shown in [14] that Euclidean spaces are not well suited to represent Internet nodes.

Using greedy embeddings based on multiple trees to improve the performance of geometric routing was proposed in [15, 16]. Authors in [16] focus on the Internet topology and assuming a logarithmic depth of the spanning tree, the memory complexity of the embedding is poly-logarithmic. In a recent work [17], enabling load balancing in geometric routing using multiple trees was proposed. Indeed using several trees improves the routing performance but increases the overall overhead of the approach.

In greedy routing, network changes such as link or node failures might affect the greedy embedding and lead the packets to local minima. Besides face routing very few studies investigated the network dynamics (failure/addition) in greedy routing. In [18], authors proposed a dynamic embedding to cope with node addition. However, network failures are handled by an adapted routing referred to as gravity-pressure. In this routing, packets that reach a dead end (void) are forwarded to the neighbors which are closer to the destination and a path trace is maintained in the packets in order to avoid loops. This imposes a large overhead to the packets header especially in large-scale topologies. In [4, 5], we proposed different recovery techniques for link/node failures in greedy routing. As the schemes were for protection, backup paths were used upon facing failures. In a more recent work, the effect of using multiple spanning trees on network resiliency was investigated [19]. However, similar to other approaches based on multiple trees, the good performance in the latter was at the cost of: (i) increased overall overhead due to construction of multiple trees, (ii) relative increased storage requirements of the packet headers and (iii) increased computational complexity of the forwarding plane.

Although, the aforementioned techniques provide fault-tolerant greedy routing, they mostly lack a general mechanism to react efficiently to topological changes. Therefore, in this work we propose a distributed algorithm for greedy embedding which deals with any type of network changes (failure/addition) in a short period of time. Instead of relying on complex geometry (e.g. hyperbolic), the algorithm constructs a spanning tree and calculates the coordinates based on that. Different types of dynamics in the network are handled by re-construction of the tree and re-calculation of the coordinates. In order to identify and quantify trends and trade-offs of re-convergence behavior of greedy routing, we extend the proposed

algorithm to generate backups to apply the protection scheme proposed in [5] in distributed setups.

The proposed scheme is most comparable with the PIE protocol [16] as both target large-scale networks and use tree structure to guarantee packet delivery. However, as PIE uses multiple trees to improve the performance in terms of stretch, it increases the overall overhead of the scheme. Assuming a single spanning tree in PIE, in general we expect similar performance in terms of stretch with higher number of exchanged messages and longer convergence times in PIE because: (i) it uses separate modules for tree generation and coordinate calculation while we combine them in a single module, (ii) in PIE, the two modules act on a periodic basis in order to accommodate possible changes in the tree, while in our scheme the module runs upon tree changes and (iii) PIE uses the STP protocol known for long convergence times.

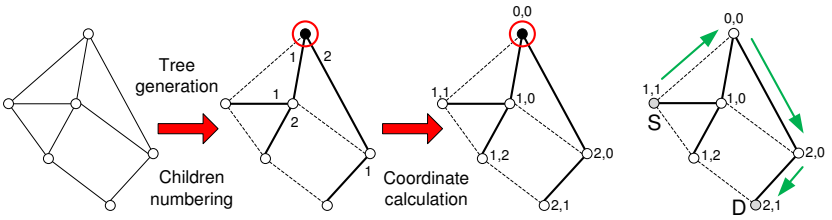
In comparison to link state routing protocols such as OSPF (Open Shortest Path First), the proposed scheme is only based on local information[6] in the network nodes while in OSPF, every node constructs a graph of network topology and calculates the shortest path to every destination in the network. Indeed having a global view is not problematic in small/medium size topologies. But in large-scale topologies where memory scalability is an issue, the added value of the proposed scheme is apparent. However, better memory scalability in this scheme is at the cost of a possible deviation from shortest paths.

3.3 Greedy Tree-based geometric Routing (GTR)

GTR primarily comprises two components: (i) tree-based embedding and (ii) greedy forwarding. In the embedding scheme, network nodes are assigned virtual coordinates based on a spanning tree which are then used by the greedy forwarding component to forward the packets towards the intended destinations.

In the concept of geometric routing (or any locator based routing scheme), to send a packet to a destination t, a node v needs to know the locator/coordinates of the node t. Therefore, a scalable mapping system to bind node identifiers (e.g. existing IP addresses in the context of a migration scenario) to node locators (e.g., coordinates) is required. A possible option is to use a DNS-like mechanism for the aforementioned name-to-locator resolutions. There exist several proposals in the literature considering the Locator/Identifier split principle in the routing architecture for a scalable future Internet [20]. These proposed mapping systems focus on different aspects such as scalability, resiliency, security and end-host mobility. However, the focus of this work is not on such a mapping system and its detail is out of the scope of the paper. We therefore assume that this mapping system exists in the network.

[6]The forwarding decisions are only based on neighbors' coordinates.



(a) Tree-based greedy embedding (b) Greedy forwarding

Figure 3.1: The steps of: i) tree generation ii) children numbering and iii) coordinates calculation are depicted in (a). The dashed lines represent the links which are not in the tree. An example of greedy forwarding (from S to D) based on the calculated CVs is depicted in (b).

In the rest of this section we describe (i) the tree-based greedy embedding and corresponding greedy forwarding, (ii) the protection scheme for single link failure in geometric routing and (iii) the distributed algorithm for these schemes.

3.3.1 Tree-based greedy embedding

In [4], instead of relying on a complex geometry (e.g. hyperbolic) we used a simple embedding based on a spanning tree of the network which decreases the complexity and overhead of the scheme. The steps for calculating this embedding are: (i) a rooted spanning tree is generated, (ii) the root node sets its coordinate vector (CV) to zero, (iii) each node numbers its children from 1 to d (d is the number of children of the node in the tree), (iv) a node calculates the CV of its children by putting the number assigned to each child in place of the first zero coordinate in its own CV. Each coordinate in a CV should have enough bits to be able to show a value equal to the maximum degree of the tree and the number of coordinates in a CV is equal to the depth of the tree. Therefore, the required memory to represent the CVs is determined by the depth and the maximum degree of the tree. Fig. 3.1a depicts an example for this embedding. In this context, tree-distance which is the hop count on the tree between two nodes is considered as metric and is calculated easily using the assigned CVs.

Given the CVs of two nodes e.g. nodes (1,1) and (1,2) in Fig. 3.1a, in order to calculate the tree-distance between them the largest common prefix of the two CVs is determined which is (1). In each CV, the number of non-zero coordinates after the common prefix is counted (it is one in both CVs). In case of no common prefix, all the non-zero coordinates of each CV are counted. The sum of these two numbers indicates the tree-distance between the two nodes. In the rest of the paper the term ‘CV’ and ‘coordinates’ are used interchangeably.

Note that a distance-decreasing path is guaranteed via the spanning tree of the network. However, greedy forwarding based on the explained embedding is different from tree routing[7] because the shortcut links[8] can also be used. An example of greedy forwarding (from S=(1,1) to D=(2,1)) is given in Fig. 3.1b. Each node is aware of its neighbors CVs. In every node the tree-distance between each neighbor and node D is calculated and the one with the minimum distance is selected as the next hop. In node S, the tree-distances (td) are as follows: td((0,0),(2,1))=2, td((1,0),(2,1))=3, td((1,2),(2,1))=4. Therefore, the neighbor (0,0) is selected as the next hop. Repeating the same procedure in every node results in the path depicted by arrows ((1,1)→(0,0)→(2,0)→(2,1)). This is not equal to the shortest path ((1,1)→(1,2)→(2,1)), but it is more efficient than tree routing ((1,1)→(1,0)→(0,0)→(2,0)→(2,1)).

Below, we provide an overview on the process sequence of GTR:

1. Routers start exchanging messages to create a spanning tree.
2. They calculate the coordinates based on the constructed spanning tree.
3. They register their CV to the mapping system.
4. Each router stores the CVs of its neighbors to enable greedy forwarding.

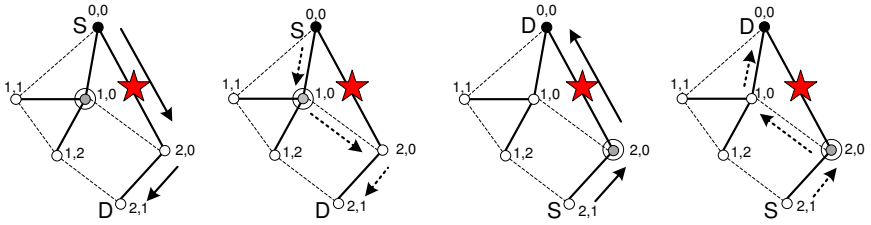
In GTR in order to send a packet from node v to node t, the mapping system is queried by v to get the CV of t. The packet is then forwarded towards t following distance-decreasing policy. The above processes are detailed in Section 3.3.3.

3.3.2 Single link failure protection technique

In [5], we proposed recovery techniques for link failures in geometric routing. The protection is used only for tree link failures, as shortcut failures do not affect the greedy embedding. For every tree edge e attached to a node, we look for an intermediate node from which greedy routing to the destination is possible (assuming e is failing). We need to distinguish between upward and downward failures. An upward failure refers to a scenario where the failing link was supposed to be passed in the direction towards the root of the tree. In case of a downward failure, the link was supposed to be passed towards leaves of the tree. In a downward failure scenario, we search to find a node which has a shortcut to the subtree below the failing link. In an upward failure scenario a node with a shortcut to another node out of the subtree below the failing link is found. As the tree-based embedding determines the location of nodes in the tree, it can be used to find the suitable intermediate nodes. The forwarding upon failure detection is as follows: (i) greedy

[7] In tree routing, a spanning tree of the network is constructed and packets are forwarded towards their destination using only tree links.

[8] Links which are not in the spanning tree of the network.



(a) Downward failure (b) Upward failure

Figure 3.2: (a) and (b) depict a downward and an upward failure scenario respectively. The primary path is depicted in the graph at left and the recovery path is depicted on the right graph with dashed arrows. The selected intermediate nodes are colored gray.

forward to the intermediate node, (ii) take the shortcut to the corresponding subtree, (iii) resume greedy forwarding on the tree. Fig. 3.2 depicts examples of upward and downward failures. The primary and recovery paths are depicted by solid and dashed arrows respectively. For detailed explanation of the scheme, we refer the interested readers to [5].

3.3.3 Proposed distributed algorithm

As evaluated in [21], the choice of the spanning tree may impact the resulting performance of the system. Finding the optimal spanning tree which minimizes the path stretch is a hard problem, as the number of spanning trees for a given graph grows rapidly by increasing the number of nodes and edges in the graph [22]. Based on our experiments, trees with low depth and high degree result in low average stretch and reduce coordinate memory. For this reason, our goal is to formulate a spanning tree construction method which targets the minimization of the resulting depth, while maximizing the used degree. Constructing such a tree can be done by generating a Breadth-First-Search (BFS) spanning tree which is rooted at a node with maximum degree in the network. In this way, the generated tree has the maximum possible degree and a very low depth.

The distributed implementation of the BFS tree construction has been investigated abundantly in the literature. However, most of these studies assume a predefined root node. Therefore, they cannot be used directly in our case.

In our proposed algorithm, each node of the network should have a unique identifier (ID). This ID is composed of two parts: (i) the degree of the node and (ii) a unique number. These two parts are concatenated in such a way that a node with higher degree has a higher ID as well (ID = degree:number). An example of these IDs is given in Fig. 3.3. The algorithm generates a BFS spanning tree which is rooted at the node with the highest ID. Once the tree is generated and the coordinates are assigned, the search for finding the suitable intermediate nodes

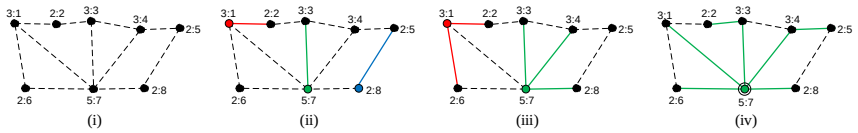


Figure 3.3: Example of tree construction process. (i) and (iv) depict the initial and the final states of the tree construction process respectively. (ii) and (iii) depict two possible intermediate stages with partial trees.

required for the link failure protection is started.

3.3.3.1 Tree construction

Each node in the network initiates spanning tree generation considering itself as the root of the tree. As the objective is to generate a single spanning tree, all except one tree should be suppressed based on a rule. We use the nodes IDs as tie-breaker allowing only the tree initiated by the root with the higher ID to continue. Additionally, the level[9] of the nodes in the selected tree should be taken into account to enable minimal depth tree construction. This is done by sending ANNOUNCEMENT messages which include two parameters to indicate: (i) the ID of the root node initiating the tree and (ii) the level of the node receiving the message upon joining the tree.

When an ANNOUNCEMENT(*newroot,newlevel*) from j arrives at i:

- if $(newroot > myroot)$ or $(newroot = myroot$ and $newlevel < mylevel)$: node i should suppress its current tree/current location in the tree due to its lower priority. It updates its data structures and joins j's subtree with *newroot* as the root and *newlevel* as the location in the tree. It then announces the new states to its neighbors except j.

Eventually all nodes receive the ANNOUNCEMENT message indicating the root node with the highest ID and abandon the lower priority trees.

Fig. 3.3 illustrates an example for this tree construction process. Fig. 3.3(i) depicts the initial state when no link belongs to any tree yet (represented by dashed lines). Before convergence to a single tree, it is possible to have partial trees in the intermediate stages. Fig. 3.3(ii) and (iii) depict two possible intermediate stages. Finally a single spanning tree rooted at the highest ID node with minimum depth is constructed which is illustrated in 3.3(iv).

[9]The hop count of a node to the root in the tree.

3.3.3.2 Coordinate calculation

CV calculation and spanning tree generation can be performed simultaneously. The exchanged messages between network nodes to construct the spanning tree can include the calculated CVs and thus there is no overhead in terms of number of messages imposed by the coordinate calculation process. This means that once node i sends an ANNOUNCEMENT message to its neighbor j, it can calculate the corresponding CV (see Section 3.3.1) for j and add it to the same message. Therefore, the ANNOUNCEMENT message of i includes: (i) the ID of the root node initiating the tree, (ii) the level of node j if it joins the tree and (iii) the CV of node j if it joins the tree. Upon arrival of this message, if node j abandons its current tree, it should update its CV to the one announced in the message.

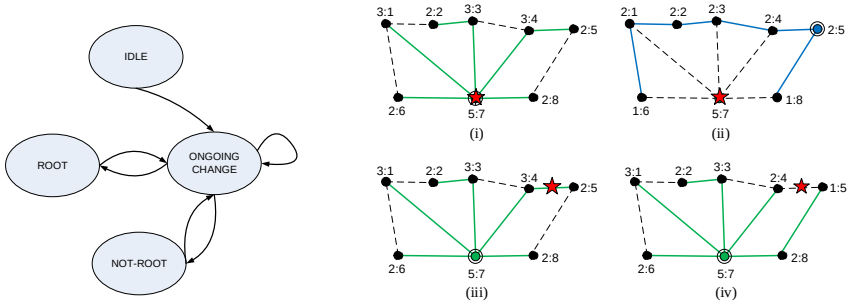
Each node announces its CV and ID to its neighbors (using ADVERTISEMENT messages) and nodes store the neighbors information in a table to enable greedy forwarding. Additionally, each node registers its CV in the mapping system. Upon a change in a node's CV, the new CV is announced to all the neighbors and the mapping system is updated.

3.3.3.3 Algorithm complexity

The algorithm follows a mechanism similar to a distributed variant of the Bellman-Ford algorithm. Assuming a single initiator, the complexity of number of exchanged messages is $O(nm)$, with n being the number of network nodes and m, the number of network edges. The reason is that a node can maximally change its level n times and each time it sends d (degree of the node) notifications. In our case as there are n initiators, a node can change its level $n \times n$ times in the worst case. If we take the sum over all the network nodes, the number of messages is in the order of $O(n^2m)$. Note that this scheme is more scalable compared to path vector routing protocols such as BGP because: (i) the worst case of $O(n^2m)$ can only happen when a tree is constructed for the first time and unlike BGP sending messages is not a continuous process. The message complexity of BGP is dependent on the value of a timer and might vary between $O(n!m)$ and $O(nm)$ [23] and (ii) the scheme provides better scalability in terms of memory in the forwarding as the next hop of a packet can be determined relying on local information (neighbors CVs).

3.3.3.4 Algorithm states

Fig. 3.4a illustrates the different states of the algorithm. All nodes start from the *IDLE* state. They move to *ONGOING-CHANGE* state when they start sending ANNOUNCEMENT messages. Eventually when all nodes receive the ANNOUNCEMENT message indicating the highest ID root node and when the lowest



(a) Algorithm states

(b) Examples for two types of failures in the tree

Figure 3.4: State diagram of the tree generation algorithm is depicted in (a). The root failure and corresponding converged tree is depicted in (b).i and (b).ii, and edge failure is depicted in (b).iii and (b).iv.

level in that tree is found, the nodes reach stability and no more messages are exchanged. The highest ID node is selected as the root and moves to the *ROOT* state and all the others have found their location in the tree and move to *NOT-ROOT* state. Thus the last two states are the stable states for the nodes. Any change in the tree (failure/addition) might bring back the nodes to the *ONGOING-CHANGE* state.

3.3.3.5 Network failures

As the proposed greedy embedding is based on a spanning tree, a change in this structure such as link/node failure might impact the embedding. Therefore, the tree and the CV of the affected nodes should be re-calculated. We categorize the failures in three groups:

- *Failures in the root node or attached to it.* If the root node is the failing component, a new spanning tree rooted at another node in the network is constructed. Additionally, a failure attached to the root node causes the tree to be re-constructed. If there is a node with higher degree compared to the previous root node in the network, the converged tree will be rooted at that. The reason for adding such a mechanism is to keep the tree structure close to the proposed maximal degree-minimal depth tree which enables a good performance in geometric routing. If no proper reaction occurs in case of such failures (because the root node is up and running), the resulting tree could have very high depth and the routing performance could be degraded drastically over time. However, as the overhead of a new tree construction can be high (see message complexity), it should be avoided as much as possible. We make sure that a mechanism to deal with such failures exits but different policies on when to use it can be considered. For example the

mechanism could be run only if a certain threshold is reached. Another possible policy is to have a clever combination of protection (see [4, 5]) and tree re-construction mechanisms. The experiments in Section 3.5.3.3 identify the trade-off between the communication overhead and routing performance if the protection mechanism is used.

- *Failures unrelated to the root in the tree.* In case of these failures, the tree is partially re-constructed. This means that the root node does not change and only the nodes in the disconnected subtree should rewire to the rest of the tree and re-calculate their CVs accordingly. Considering the simple scenario of a leaf node upon a link failure, the leaf is disconnected from the tree and it should reconnect to the rest of the tree. However, in this case no other node in the tree is affected. Fig. 3.4b depicts examples for the two types of failures in the tree and the corresponding converged trees.

- *Shortcut failures.* These failures do not affect the tree structure and the greedy embedding therefore, no action is required.

In order to enable the re-convergence mechanism in the algorithm, the ANNOUNCEMENT message is extended with a new parameter called *failed node*. In the proposed algorithm, link and node failures are treated as node failures. This means that the failure detecting node assumes that the node attached to the other side of the link is failing. Assuming that nodes were in a stable state before the failure, the failure detecting node should trigger (neighboring) nodes to start exchanging messages again if: (i) it is in the disconnected subtree and should rewire to the rest of the tree, or (ii) the tree should be globally re-constructed depending on the type of the failure. This triggering is simply done by restarting the algorithm in the node which means initializing the states and sending ANNOUNCEMENT messages to the neighbors. This initialization is required as it provides a mechanism to avoid loops. This is explained later in this section.

Additionally, the ID of the failing node is set in the *failed node* field of the message. Because, in case of failures in the first group, all nodes should be triggered to restart the algorithm to construct a new tree. Using the *failed node* field, all nodes are informed about this failure and act accordingly. In case of failures in the second group, only the nodes in the disconnected subtree are triggered to restart the algorithm to rewire to the rest of the tree. A simple improvement in case of a link failure attached to the root is that both nodes announce the ID of the root in the *failed node* field. This enables faster informing of all nodes about the change in the root.

The pseudo code of this algorithm is depicted in Fig. 3.5. Lines (2a) and (2d) are handling the two types of failures in the tree. (2a) restarts the algorithm in case of the root failure and (2d) restarts the algorithm if the node is in the disconnected

subtree. This is indicated by receiving a lower ID root/higher level than the current root/level from the parent node.

```
int parent, myroot, mylevel, myCV ← Ø, set of int children ← Ø, unrelated ← Ø
(1) When the node initiates the algorithm as a root:
(1a) parent ← Ø; myroot ← i; mylevel ← 0; myCV← 0
(1b) for all the neighbors:
(1c)        send ANNOUNCEMENT(root = i, level = 1, CV = calculated CV for the neighbor)
(2) When ANNOUNCEMENT(newroot, newlevel, newCV, failed node) arrives from j:
(2a) if (failed node = myroot) then //root node is failing – restart algorithm
(2b)        for all the neighbors:
(2c)                send ANNOUNCEMENT(root = i, level = 1, CV = calculated CV for the neighbor, failed node)
(2d) if (newroot < myroot ) or (newroot = myroot and newlevel > mylevel) and (j=parent) then //failure in upper level- restart algorithm
(2e)        for all the neighbors:
(2f)                send ANNOUNCEMENT(root = i, level = 1, CV = calculated CV for the neighbor, failed node)
(2g) if (newroot > myroot) or (newroot = myroot and newlevel < mylevel) then
(2h)        parent ← j; myroot ← newroot; mylevel ← newlevel; myCV ← newCV
(2i)        for all the neighbors except j:
(2j)                send ANNOUNCEMENT(root = myroot, level = mylevel+1, CV = calculated CV for the neighbor)
(2k) else if (newroot < myroot) or (newroot = myroot and newlevel−1> mylevel+1) then
(2l)        send ANNOUNCEMENT(root = myroot, level = mylevel+1, CV = calculated CV for j) and send REJECT(root=newroot) to j
(2m) else send REJECT(root=newroot) to j
(3) When ACCEPT(newroot) arrives from j:
(3a) if (newroot=myroot) then children ← children U {j}
(3b)        if (children U unrelated) = (neighbors/{parent}) then
(3c)                if (i = myroot ) then
(3d)                        send START to all children and send PROBE (direction=down, CV=CV of each child) for every tree edge
(3e)                else send ACCEPT(root=myroot) to parent.
(4) When REJECT(newroot) arrives from j:
(4a) if (newroot=myroot) then unrelated ← unrelated U {j}
(4b)        if (children U unrelated )= (neighbors/{parent}) then
(4c)                if (i!=myroot) then send ACCEPT(root=myroot) to parent
(5) When START arrives from j:
(5a) send START to all the children and send PROBE (direction=up/down, CV of the neighbor) for every tree edge
(6) When PROBE (newdirection, newCV) arrives from j:
(6a) Look for the desired shortcut based on the newdirection and newCV
(6b) if the shortcut is found then send ACKP (myCV, newCV) to j
(6c) else send PROBE(newdirection, newCV) to neighbors except j
(7) When ACKP (newCV, CV) arrives from j:
(7a) if destined for this node then store the newCV
(7b) else forward ACKP in the correct direction (greedy forwarding)
```

Figure 3.5: Pseudo code of the distributed algorithm for CV calculation/protection in node i

This algorithm is similar to distance-vector algorithms in the sense that: (i) every node calculates the distance (in terms of hop counts) to the root node, (ii) it announces this hop count to its neighbors and (iii) nodes update their distance to the root based on the announcements received from the neighbors. Therefore,

strategies such as split horizon (poison reverse)[10] are required to avoid loops. As explained above, once a node detects a failure it restarts the algorithm and announces itself as the new root to its neighbors. Note that its ID is lower than the current root node ID and this is used to provide a loop avoidance mechanism in the algorithm. In node i, receiving a lower ID root/higher level in an ANNOUNCEMENT message from the parent is the indication of a failure in an upper level in the tree. Knowing this, node i avoids sending back its current root ID/level to its parent and instead it restarts the algorithm to find a new location in the tree (see line (2d) in the pseudo code). This way all the nodes in the disconnected subtree are informed about the failure (by receiving lower root ID/higher level from the parent) and loops are avoided as well.

Another mechanism which avoids invalid tree constructions/loops caused by the root node failure is to put the ID of the failing node in the ANNOUNCEMENT messages. If the root node crashes, all other nodes are informed about it and before announcing stale information about the crashed root they are forced to restart the algorithm to find a new root (see line (2a) in the pseudo code).

3.3.3.6 Network additions

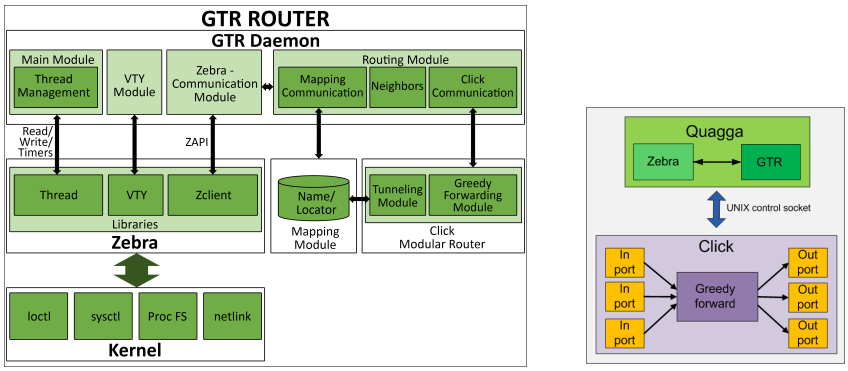
Node and link addition scenarios are also treated the same. The new node (or nodes with new edges) restarts the algorithm by announcing itself as the root node to its neighbors. Messages are exchanged until the new node finds its location in the tree. If the ID of the new node is higher than the current root node, all the nodes in the network change their states and select the new node as their root.

3.3.3.7 Protection

In order to add the functionality for finding the intermediate nodes with suitable shortcuts corresponding to the tree edges (required for the protection scheme), other types of messages are defined in the algorithm:

- ACCEPT: sent to the parent if the node has received a reply from all of its neighbors.
- REJECT: sent to the neighbor if it does not offer a better state (higher root ID, lower level in the tree) in the ANNOUNCEMENT.
- START: used to inform nodes to start looking for the intermediate nodes.
- PROBE: sent to search for the desired intermediate nodes.
- ACKP: sent to the protecting node once an intermediate node with the desired shortcut is found.

[10] A method of preventing routing loops in distance-vector routing protocols by prohibiting a router from advertising a route back onto the interface from which it was learned.



(a) Architecture of GTR implementation

(b) Node architecture

Figure 3.6: Architecture of GTR implementation and a GTR-enabled node

The first two types are used to determine the relation of nodes and their neighbors in the tree (child, parent). The root node uses these messages to know when the tree is generated. Once it receives ACCEPT from all of its neighbors, it informs network nodes by START messages to start looking for intermediate nodes by sending PROBE messages. In Fig. 3.5, the sets 'children' and 'unrelated' are used to store the children of a node and the node's neighbors in the network which are not connected via a link in the tree respectively.

3.4 Routing platform

In order to experimentally evaluate GTR, it has been implemented as a prototype. This prototype runs on top of an existing Quagga routing platform in the iLab.t virtual wall emulation testbed platform.

3.4.1 GTR implementation in Quagga routing suite

Quagga is an open source software routing suite which provides different routing protocols for UNIX based platforms and is written in C. Quagga is composed of several daemons, each for a routing protocol. Some of the available implementations are RIP, OSPF and BGP. All of these daemons interact with the kernel routing table through zebra daemon. Daemons can be controlled by a virtual terminal access provided by a telnet-based interface. Quagga architecture has a very rich library which facilitates the development of new daemons.

Fig. 3.6a depicts the architecture of the GTR implementation. The main modules in this architecture are as follows:

- GTR daemon in Quagga (control plane).

- Forwarding module in Click (forwarding plane).
- Mapping module.

The implemented GTR daemon consists of 3 major modules:

- greedyd-main.c: This is the main function and the main entry of the GTR daemon. It also includes the thread management.
- greedy-zebra.c: This module consists of the communication functions with zebra daemon.
- greedyd.c: This module is composed of GTR daemon initialization and all of the logic for tree generation, coordinate calculation and intermediate node determination for link failure protection.

In the GTR daemon, it is possible to select between protection and coordinate re-calculation mechanisms. This can be set in the daemon configuration file in Quagga.

As the GTR daemon is only the routing engine, the forwarding plane is implemented in Click modular router to enable greedy forwarding of packets. As mentioned before, a mapping module is required to bind names to locators. This module interacts with GTR daemon to get the mapping between a node's name and its CV which is then queried by the Click module to forward packets. For simplicity, we considered a centralized mapping base reachable by all nodes, because this mapping system is out of the scope of this paper.

3.4.2 Greedy forwarder implementation in Click modular router

The Click modular router is an open source platform made in C++ that allows a Linux PC to act as a router/switch. Click has a modular architecture which means that the configuration of a router/switch boils down to the interconnection of a chain of modules/elements which process packets or frames.

In order to greedy forward the packets, we developed Greedy-forward element in Click. The packets consist of: (i) header which is composed of source and destination CVs of the packet and (ii) data. The major functionality of the Greedy-forward element is to extract the CV of the destination from the incoming packets and calculate the distance between every neighbor and the destination. In case of protection, the header of the packet is augmented with the CV of the intermediate node and a flag which indicates the routing mode: greedy/protection (see [5]). The neighbors' information (such as CV, ID and the port they are connected to) is stored in the neighbor table of this element. All the routing information such as node's CV and the neighbors' information are calculated by the GTR daemon in Quagga and set through Unix control socket to the Click element. The architecture

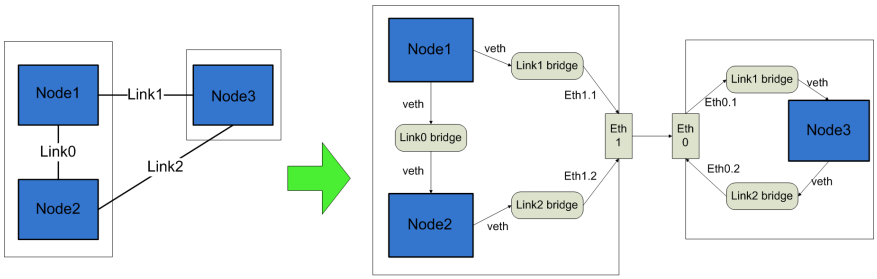


Figure 3.7: Connecting LXCs using bridges

of a GTR-enabled node is depicted in Fig. 3.6b. The main methods in this element are as follows:

- Tree-distance: calculates tree-distance between two nodes.
- Push: greedy forwards the packet or calls the protection-forward method.
- Protection-forward: forwards the packet based on the protection scheme.

3.5 Experimentation and discussion of the results

3.5.1 iLab.t virtual wall platform

We perform our experimentation on iLab.t testbed of iMinds. The iLab.t virtual wall is a generic test environment which provides computing hardware and different software tools to evaluate the performance of the novel solutions. There exist two virtual wall setups: wall1 consists of 200 servers and wall2 has 100 servers. The servers in each wall are interconnected by a nonblocking 1.5 Tb/s VLAN Ethernet switch. The servers are equipped with 4-6 Ethernet interfaces. The Emulab software of the University of Utah is run on this testbed. As the goal of our experimentation is to evaluate the proposed scheme in large-scale topologies, virtualization techniques were considered to allow for physical nodes to host multiple virtual nodes. We re-compiled a recent Debian 3.* kernel to support LXC container-based virtualization, and transparent VLAN bridging between LXCs to support more number of interfaces. Fig. 3.7 depicts how bridges are used to connect virtual nodes (LXCs) in different physical nodes.

3.5.2 Emulation vs. simulation

Simulation experiments focus on the evaluation of spatial metrics while emulation experiments concentrate on time-sensitive and resource-sensitive metrics. Simulation and emulation are complementary and some experiments are suited for simulation while others are better placed for emulation. In simulation, abstractions

are made to model node and network while in emulation no model or abstraction is involved but a prototype implementation is made. Emulation entities can interact with real hardware implementations. Therefore, more realistic results can be achieved through the emulation experiments.

3.5.3 Experimentation setup and objectives

We perform different experiments to evaluate the convergence behavior of GTR and its recovery capabilities upon occurrence of network failures and compare the experimental results with BGP. We selected BGP for comparison because it is the only routing protocol which has been implemented and actually applied in large-scale (inter-domain) networks. It is well-understood and its pros and cons are known.

This experimental evaluation should enable more careful conclusions regarding GTR applicability to inter-domain settings. For these experiments, a topology based on the GLP model [24] with 1 K nodes, and parameters: $m = 1.13$, $m0 = 10$, $p = .5972$, $beta = .1004$ is considered. These parameters have been corroborated by the extensive study of [25].

3.5.3.1 Stretch evaluation of GTR

Before starting the GTR evaluation in scenarios with different network dynamics, we evaluated its performance in terms of stretch to determine how the path length is affected using this scheme. Routing stretch is defined as the ratio between the length of the path produced by greedy forwarding and the length of the shortest path for the same source-destination pair. We report the average stretch for topologies based on the GLP model ranging from 200 to 1000 nodes with parameters mentioned earlier. Note that all the other experiments in the following subsections are on a GLP topology with 1000 nodes and the aforementioned parameters unless stated otherwise. We compared the stretch results of GTR with tree routing.

For this experiment, we calculated the stretch for every pair of nodes in the network. The average stretch for each topology is calculated and this is repeated for 10 different GLP topologies and the average over all of them is reported in Table 3.1. As we see, the average stretch of GTR scales very well with the increase in the number of nodes and it outperforms tree routing significantly.

Table 3.1: The average stretch of GTR and tree routing on different size GLP networks

Network size	200	400	600	800	1000
GTR avg stretch	1.095	1.094	1.096	1.099	1.101
Tree-routing avg stretch	1.347	1.346	1.354	1.366	1.372

In the following subsections, the GTR stretch is evaluated in different dynamic scenarios. This gives an overview on how the routing quality is affected by network changes.

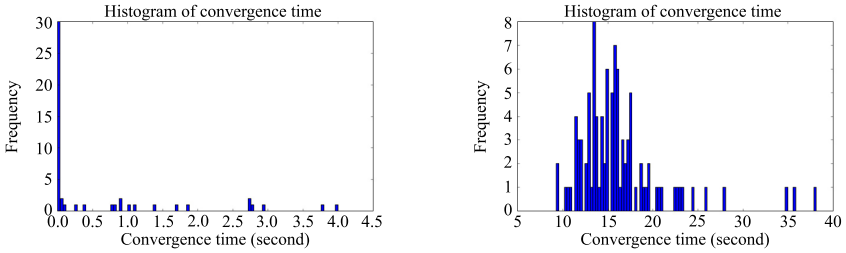
3.5.3.2 Recovery capabilities of GTR

In routing schemes such as GTR with embedding based on a tree structure, a single failure in the topology might affect the tree connectivity and break the embedding. In order to re-calculate the embedding the tree structure should be re-constructed. Based on the location of the failure, local or global changes in the coordinates of the nodes are possible. The closer the failure to the root of the tree, the higher the number of affected nodes. While in case of a failure attached to a leaf node, only the leaf node updates its location in the tree.

In the experiments of this section, we evaluate the performance of GTR in terms of convergence time, communication cost and routing quality (stretch) considering different failure scenarios (single and multiple failures). We emulate the failures by disabling the interfaces in a node. We select random links in the network to be failing. As explained in Section 3.3.3, once a node detects the failure (this is done through a zebra daemon), it re-starts the algorithm by announcing itself as the root to its neighbors. The message exchanges continue until new locations in the tree and thus new coordinates are found for the node and all of its descendants. The network reaches stability once no more messages are exchanged between nodes. We capture all the exchanged messages (control traffic and updates in the mapping component) and the time between occurrence of a failure and the last sent message is considered as the convergence time. Note that in this set of experiments no protection scheme is used and coordinates of the affected nodes are re-calculated.

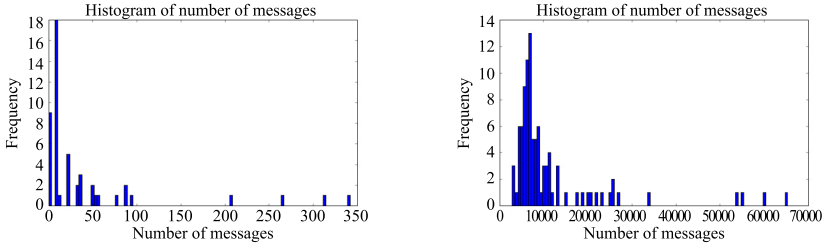
The same experiment is repeated on BGP components and the control traffic is captured in the network. The convergence time is calculated from the moment a failure occurs until the last UPDATE message sent in the network. BGP keeps on sending KEEPALIVE messages after convergence (this is different from GTR in which no more messages are exchanged after convergence). Indeed, there are several ways to reduce BGP convergence time such as tuning the MRAI (Minimum Route Advertisement Interval) timer or using BGP Fast-reroute or add-path technique. We use an optimal value for MRAI for baseline comparison.

In order to evaluate the re-convergence trend for single link failure scenario, we started GTR components on the network nodes and waited for some time that the routing components converge globally (this time is obtained experimentally and is in the order of few minutes). Then 100 links were selected randomly and disabled one at a time, convergence time was measured and links were enabled again. The first, 25th, 50th, 75th and 99th percentile values for convergence time of GTR are: 0, 0, 0.16, 0.29 and 4.8 s respectively. As not all link failures leads to changes of



(a) Single tree link failure - GTR (b) Single link failure - BGP

Figure 3.8: Distribution of convergence time in single failure scenario



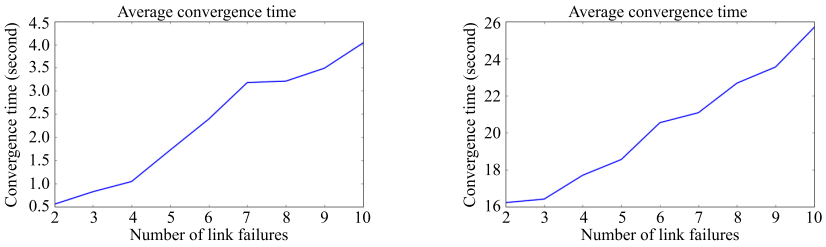
(a) Single tree link failure - GTR (b) Single link failure - BGP

Figure 3.9: Distribution of number of exchanged messages in single failure scenario

coordinates, the convergence time for some of the failures is zero. Therefore, we repeated the experiment considering only single tree link failures (50 random tree links). Fig. 3.8a depicts the distribution of the convergence time for the single tree link failures. The first, 25th, 50th, 75th and 99th percentile values of the measured convergence times are: 0.0169, 0.0226, 0.0315, 1.0398 and 4.003 s. Some of the tree link failures affect very few nodes (e.g. only a leaf node) and therefore, the corresponding convergence times are low (tens of milliseconds). The closer the failure to the root of the spanning tree, the higher the number of affected nodes and therefore, the higher the convergence time and the communication cost.

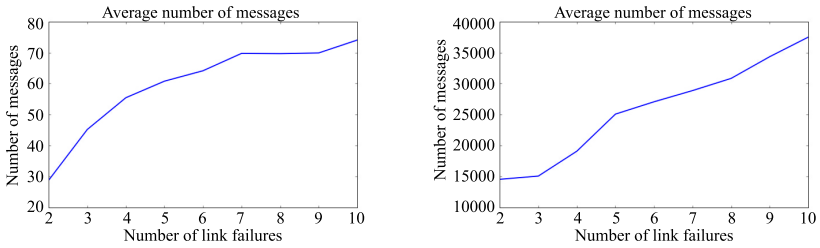
We repeated the same experiment on BGP routing components. For a fair comparison with GTR (as it uses no timer), we tuned the MRAI timer to 5 s which seems to be close to optimal value for eBGP [26]. Fig. 3.8b depicts the distribution of the convergence time for 100 link failures and Fig. 3.9 illustrates the distribution of the number of exchanged messages during the convergence of both GTR and BGP. Indeed, BGP with default MRAI value equal to 30 s, showed higher convergence time of 57.46 s on average.

In the next experiments, we evaluated the behavior of GTR in case of multiple simultaneous failures (up to 10 link failures) and compared it with BGP. Links were selected randomly. Figs. 3.10 and 3.11 report the average convergence time



(a) Multiple link failures - GTR (b) Multiple link failures - BGP

Figure 3.10: Average convergence time in multiple failures scenario



(a) Multiple link failures - GTR (b) Multiple link failures - BGP

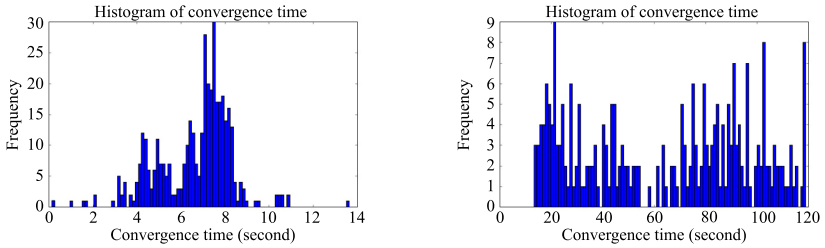
Figure 3.11: Average number of exchanged messages in multiple failures scenario

and number of exchanged messages over 100 repetitions of the experiments. In both scenarios we observe an increase trend and in GTR, the average convergence time in 10 simultaneous failures does not exceed 4.5 s while in BGP, the average convergence time for 10 link failures is almost 26 s.

In order to have a view on the routing quality of GTR, we evaluated the average stretch in the scenario of multiple link failures. The average stretch remained almost the same for different number of failures and was equal to 1.0176. Although 10 link failures are less than 1 percent of the links in the topology, this result shows that re-calculation of the coordinates results in good performance in terms of stretch with low convergence time.

3.5.3.3 Impact of topology dynamics on GTR

Topology changes such as link/node failures/additions may change the structure on which the coordinates in the geometric routing system is built (spanning tree in case of GTR). The resulting changes may affect even non-local parts (with respect to failure/addition) of the network for a certain period of time. A clever combination of protection and coordinate re-calculation techniques may render geometric routing schemes more or less prone to these events. The goal of this experiment is to evaluate how much a topology may change before it has a strong impact on the



(a) Topology change - GTR

(b) Topology change - BGP

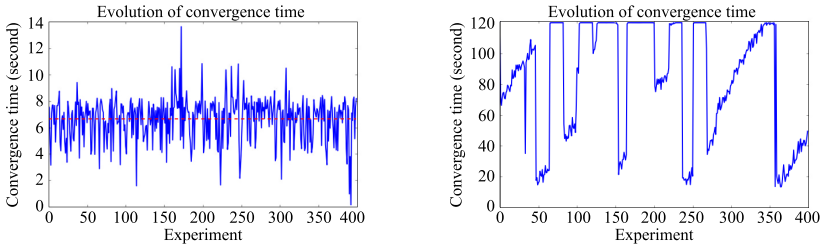
Figure 3.12: Distribution of the convergence time in 10% topology change scenario

performance of geometric routing systems in terms of stretch, convergence time and communication cost.

In this experiment, we assume that 10% of the topology changes. To this end, 400 random links ($\approx 10\%$ of links) were selected to be failing and they were disabled one at a time and were added to different nodes (selected randomly) one by one. Thus in every step of the experiment, a single link is removed and a new link is added between two other nodes in the topology and we wait long enough (order of few minutes) that the GTR component converges. The convergence behavior is measured similar to previous experiment based on the control messages. Fig. 3.12 illustrates the distribution of the convergence time for both GTR and BGP and the evolution trend of the convergence time is depicted in Fig. 3.13. The distribution of the number of exchanged messages is reported in Fig. 3.14. As we observe in both schemes no special trend is visible which can be explained by the randomness of the changes. In GTR, based on the location of the changes the convergence time differs. However, it does not exceed 14 s for a link failure and addition. Note that in general a link addition leads to higher convergence time than a failure in GTR. The reason is that (i) a shortcut failure does not impact the structure of the greedy embedding and thus no updates are exchanged and (ii) in case of a tree link failure, only the child node and the nodes in the disconnected subtree re-calculate their CVs while the parent node does not look for new CV. In case of link additions, both nodes attached to the link re-calculate their CVs and might affect more nodes.

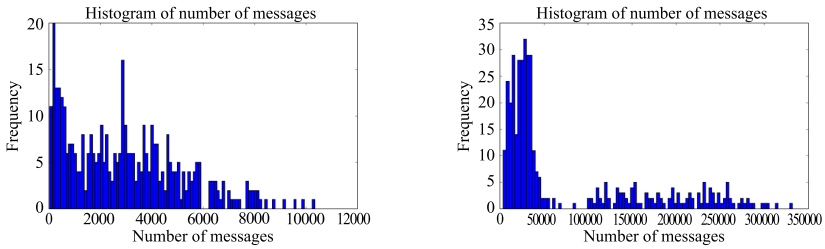
As evaluating stretch for every single link addition/failure was time consuming, we measured the average stretch 20 times during the 10% topology change scenario. This means that the average stretch was evaluated after every 20 link additions/failures. Similar to previous experiment, the re-calculation of the coordinates resulted in good performance in terms of stretch and the average stretch remained almost 1.11 in all steps.

Now that we have evaluated the topology change scenario with coordinate re-calculation scheme in GTR, we consider the protection scheme to find out how



(a) Topology change - GTR (b) Topology change - BGP

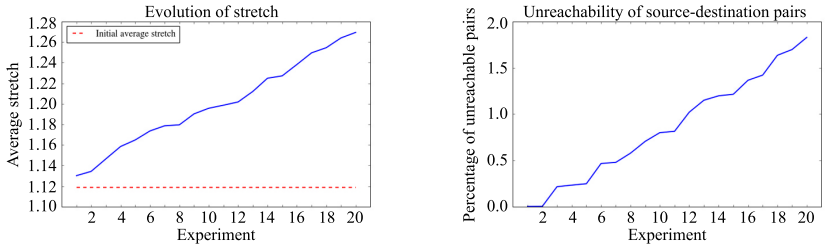
Figure 3.13: Evolution of the convergence time in 10% topology change scenario



(a) Topology change - GTR (b) Topology change - BGP

Figure 3.14: Distribution of number of exchanged messages in 10% topology change

much a topology can change without having a strong impact on the performance of the routing in terms of stretch and nodes reachability. In the proposed protection scheme, the alternative paths are pre-calculated and thus, no updates are exchanged between nodes upon any change in the topology. As there is no communication cost/convergence time, only the switch-over time is important and also the overhead on memory consumption of network nodes should be considered which was evaluated in [5]. However, as the topology changes, the pre-calculated paths might be far from optimal resulting in bad performance of the routing in terms of stretch. In Fig. 3.15a, we report the average stretch upon 10% topology change considering the protection technique. Similar to the previous experiment, the average stretch was measured 20 times during the 10% topology change scenario. This means that we evaluated the average stretch after every 20 link additions/failures in the network. Note that the protection scheme for single link failure was implemented (due to its low overhead/memory consumption). Therefore, depending on the changes, some of the nodes might not be reachable anymore. The reported values in Fig. 3.15a were measured only for the reachable source-destination pairs. The number of disconnected pairs in every 20 steps is depicted in Fig. 3.15b. Although, the average stretch increases, it does not exceed 1.28 and only less than 2% of the pairs are unreachable after 10% topology change. This study identifies



(a) Average stretch - GTR

(b) Disconnected pairs - GTR

Figure 3.15: Average stretch and reachability using protection scheme

the trade-off between communication cost/convergence time and routing quality in terms of stretch and nodes reachability.

In the last experiment, we evaluated the protection time of the implemented scheme in GTR which was measured at the forwarding plane. As it is not affected significantly by the size of the topology, we evaluated it in a GLP topology with 100 nodes. We selected a pair of nodes randomly and sent traffic (generated in Click) from source to the destination every 1 ms. It was ensured that the greedy path between source and destination went through at least one tree link. A failure was emulated by disabling a tree link along the path between the two nodes. We monitored the traffic in the destination, and the protection time was calculated by measuring the gap in the received traffic. We repeated this experiment 20 times and the average protection time is 20.4 ms. The first, 25th, 50th, 75th and 99th percentile values are 5, 12, 19.5, 28.5 and 38 ms respectively. Although the protection time is not evaluated for all possible source-destination pairs, the reported values give an idea on how the convergence time can be improved using the protection scheme compared to the tree re-construction approach.

3.5.4 Lessons and discussions

Greedy geometric routing based on a spanning tree has desirable properties in terms of memory and computational resources resulting from its relative simplicity. However, network failures which affect the spanning tree connectivity might cause local or global changes to nodes depending on their location. The closer the failure to the root of the tree, the larger the number of affected nodes and the higher the convergence time/communication cost. However, the experimental results showed that these values are a lot less compared to the results of BGP with MRAI timer set to 5 s. As expected, the re-calculation of coordinates resulted in good performance of the routing in terms of stretch and the average stretch remained almost unchanged for different failure scenarios in the network. Generally in GTR, the network failures result in potentially high number of affected nodes/-

paths, while the convergence time for their recovery is relatively low.

With regard to BGP, MRAI timer tuning, using Fast-reroute or add-path technique affect the convergence behavior and can reduce the convergence time significantly. For a baseline comparison we only considered MRAI tuning and used a close to optimal value (5 s) for MRAI timer.

The experimental evaluation identified another trade-off between convergence time/communication cost and stretch/reachability in GTR using protection and coordinate re-calculation mechanisms. In case of using the protection scheme, backup paths are used and coordinate re-calculation is avoided (no communication cost in case of failures) but the stretch and node reachability are negatively affected. However, the low performance loss in the routing quality in case of using the protection scheme suggests that with clever combination of protection (especially in the levels of the tree close to the root node) and coordinate re-calculation schemes, good performance in terms of stretch and convergence time/communication cost can be achieved and global re-embeddings can be avoided.

3.6 Conclusions

Due to lack of resiliency mechanisms, geometric routing has not been considered as a valid option for routing in large-scale networks and never been thoroughly evaluated in realistic emulation settings. In this paper we showed that geometric routing can be made robust to network failures relying on distributed algorithms. A prototype was made and performance trends were compared to BGP which is the only routing scheme currently active in large-scale network settings. The experiments were performed on a large topology of 1000 nodes on a emulation testbed. A geometric routing such as the one proposed will not replace BGP in coming years, however, the paper showed that in this routing scheme, network failures can be recovered rapidly and the convergence time and communication cost are much lower than BGP even with the MRAI timer set to 5 s. In addition, using a protection scheme in combination with the proposed distributed algorithm, good performance in terms of stretch can be achieved while the convergence time/communication cost remains low and global re-embedding is avoided. Open points and further work related to applicability of geometric routing in large networks are designing a scalable mapping system which binds the names (e.g. IP addresses) to locators (coordinates) and enabling routing policy.

Acknowledgment

This work is partly funded by the European Commission through the EULER project (Grant 258307), part of the Future Internet Research and Experimentation

(FIRE) objective of the Seventh Framework Programme (FP7).

References

[1] R. Fonseca, S. Ratnasamy, J. Zhao, C. Ee, D. Culler, S. Shenker, and I. Stoica. *Beacon vector routing: Scalable point-to-point routing in wireless sensornets*. In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2, pages 329–342. USENIX Association, 2005.

[2] G. Huston. *BGP Routing Table Reports*, 2014. http://bgp.potaroo.net/.

[3] C. Papadimitriou and D. Ratajczak. *On a conjecture related to geometric routing*. Theoretical Computer Science, 344(1):3–14, 2005.

[4] S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester. *Single failure resiliency in greedy routing*. In Proceedings of the 9th international conference on Design of Reliable Communication Networks, pages 312–319, 2013.

[5] S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester. *Link failure recovery technique for greedy routing in the hyperbolic plane*. Computer Communications, 36(6):698–707, 2013.

[6] B. Karp and H. Kung. *GPSR: greedy perimeter stateless routing for wireless networks*. In Proceedings of the 6th annual international conference on Mobile computing and networking, pages 243–254. ACM, 2000.

[7] A. Maghsoudlou, M. St-Hilaire, and T. Kunz. *A Survey on Geographic Routing Protocols for Mobile Ad hoc Networks*. Systems and Computer Engineering, Technical Report SCE-11-03.–Carleton University.–2011.–49 p, 2011.

[8] R. Kleinberg. *Geographic routing using hyperbolic space*. In INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, pages 1902–1909, 2007.

[9] D. Eppstein and M. Goodrich. *Succinct Greedy Geometric Routing Using Hyperbolic Geometry*. IEEE Transactions on Computers, 60(11):1571–1580, 2011.

[10] R. Flury, S. Pemmaraju, and R. Wattenhofer. *Greedy routing with bounded stretch*. In INFOCOM 2009, IEEE, pages 1737–1745, 2009.

[11] C. Westphal and G. Pei. *Scalable routing via greedy embedding*. In INFOCOM 2009, IEEE, pages 2826–2830, 2009.

[12] Y. Wang, C.-W. Yi, M. Huang, and F. Li. *Three-dimensional greedy routing in large-scale random wireless sensor networks*. Ad Hoc Networks, 11(4):1331–1344, 2013.

[13] S. S. Lam and C. Qian. *Geographic routing in-dimensional spaces with guaranteed delivery and low stretch*. Networking, IEEE/ACM Transactions on, 21(2):663–677, 2013.

[14] S. Lee, Z.-L. Zhang, S. Sahu, and D. Saha. *On suitability of Euclidean embedding for host-based network coordinate systems*. Networking, IEEE/ACM Transactions on, 18(1):27–40, 2010.

[15] M. Tang, H. Chen, G. Zhang, and J. Yang. *Tree Cover Based Geographic Routing with Guaranteed Delivery*. In Communications (ICC), 2010 IEEE International Conference on, pages 1–5. IEEE, 2010.

[16] J. Herzen, C. Westphal, and P. Thiran. *Scalable routing easy as PIE: A practical isometric embedding protocol*. In Network Protocols (ICNP), 2011 19th IEEE International Conference on, pages 49–58. IEEE, 2011.

[17] R. Houthooft, S. Sahhaf, W. Tavernier, F. De Turck, D. Colle, and M. Pickavet. *Robust Geometric Forest Routing with Tunable Load Balancing*. In Proceedings of INFOCOM 2015, 2015.

[18] A. Cvetkovski and M. Crovella. *Hyperbolic embedding and routing for dynamic graphs*. In INFOCOM 2009, IEEE, pages 1647–1655, 2009.

[19] R. Houthooft, S. Sahhaf, W. Tavernier, F. De Turck, D. Colle, and M. Pickavet. *Fault-tolerant Greedy Forest Routing for Complex Networks*. In Proceedings of 6th International Workshop on Reliable Networks Design and Modeling (RNDM), 2014.

[20] M. Hoefling, M. Menth, and M. Hartmann. *A Survey of Mapping Systems for Locator/Identifier Split Internet Routing*. IEEE Communications Surveys and Tutorials, 15:1842–1858, 2013.

[21] A. Cvetkovski and M. Crovella. *On the choice of a spanning tree for greedy embedding of network graphs*. Networking Science, 3(1-4):2–12, 2013.

[22] G. Kirchhoff. *Ueber die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird*. Annalen der Physik, 148(12):497–508, 1847.

[23] A. Bremler-Barr, Y. Afek, and S. Schwarz. *Improved BGP convergence via ghost flushing*. In INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, volume 2, pages 927–937. IEEE, 2003.

[24] T. Bu and D. Towsley. *On distinguishing between Internet power law topology generators*. In INFOCOM 2002. Twenty-First Annual Joint Conference

of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 2, pages 638–647. IEEE, 2002.

[25] H. Haddadi, D. Fay, S. Uhlig, A. Moore, R. Mortier, A. Jamakovic, and M. Rio. *Tuning topology generators using spectral distributions*. In Performance Evaluation: Metrics, Models and Benchmarks, pages 154–173. Springer, 2008.

[26] Q. Jian, H. Ruibing, and L. Xing. *The optimal rate-limiting timer of BGP for routing convergence*. IEICE transactions on communications, 88(4):1338–1346, 2005.

4

Efficient geometric routing in large-scale complex networks with low-cost node design

While the previous two chapters focused on resiliency in geometric routing, this chapter investigates the scalability and efficiency of the proposed tree-based geometric routing scheme. The objective is to indicate that the simplicity of the scheme does not negatively impact the routing performance. Furthermore, the proposed scheme enables implementing a greedy router at a very low cost. This cost is measured in terms of required silicon area for implementing the hardware. Simulation experiments on large-scale networks resembling the Internet topology are conducted. The proposed scheme is compared to geometric routing based on hyperbolic coordinates. The superiority of our scheme is confirmed with the lower memory requirements for coordinate representation and limited resources for the greedy router implementation, while similar routing stretch is achieved.

⋆ ⋆ ⋆

S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, P. Demeester

Published in IEICE Transactions on Communications, 2016.

Abstract The growth of the size of the routing tables limits the scalability of the conventional IP routing. As scalable routing schemes for large-scale networks are

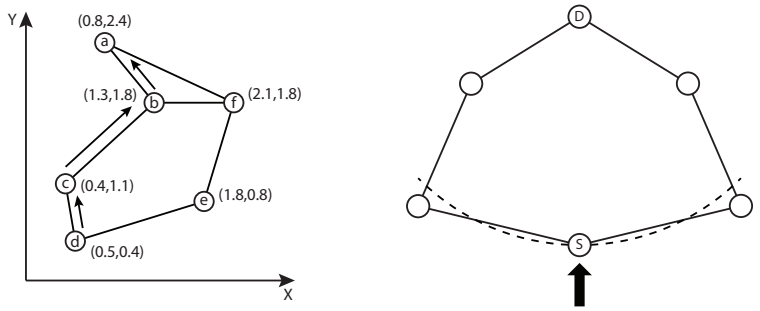
highly demanded, this paper proposes and evaluates an efficient geometric routing scheme and related low-cost node design applicable to large-scale networks. The approach guarantees that greedy forwarding on derived coordinates will result in successful packet delivery to every destination in the network by relying on coordinates deduced from a spanning tree of the network. The efficiency of the proposed scheme is measured in terms of routing quality (stretch) and size of the coordinates. The cost of the proposed router is quantified in terms of area complexity of the hardware design and all the evaluations involve comparison with a state-of-the-art approach with virtual coordinates in the hyperbolic plane. Extensive simulations assess the proposal in large topologies consisting of up to 100K nodes. Experiments show that the scheme has stretch properties comparable to geometric routing in the hyperbolic plane, while enabling a more efficient hardware design, and scaling considerably better in terms of storage requirements for coordinate representation. These attractive properties make the scheme promising for routing in large networks.

4.1 Introduction

Geometric routing has been proposed in the literature as an alternative to traditional lookup-based routing schemes to solve their scalability issue in terms of memory consumption. In this routing, only local information is required to find the next hop of a packet. Although geometric routing was initially designed for ad hoc and wireless sensor networks (WSNs) [1], the applicability of this routing paradigm to wired networks has gained increasing interest and has been investigated as well [2]. However, the latter require different design approaches because unlike WSNs, many of the inhomogeneously wired networks are modeled as scale-free networks[1]. Geometric routing is being applied in different areas. Concretely, Content-Centric Networking (CCN) is one example which can benefit significantly from geometric routing. CCN is known to work in many environments from high-speed data centers to resource constrained sensors. It is also used in Internet of Things (IoT) scenarios as was investigated in [3]. Using this routing, efficient and scalable content-based forwarding is possible as was successfully demonstrated in our previous works [4, 5].

The idea of geometric routing is to attach coordinates to nodes in a network. An obvious choice is to attach GPS coordinates to the routers of a communication network [1]. Every node in the network is aware of its own coordinate and the coordinates of its neighbors. Upon arrival of a packet, the distance between every neighbor and the packet's destination is calculated and the neighbor

[1] A scale-free network is a network whose degree distribution follows a power law, at least asymptotically. That is, the fraction $P(k)$ of nodes in the network having k connections to other nodes goes for large values of k as $P(k) \approx c.k^{-\gamma}$. Wikipedia, https://en.wikipedia.org/wiki/Scale-free_network



(a) Example of greedy forwarding (b) Example of local minimum

Figure 4.1: Example of greedy forwarding in Euclidean space is given in (a) and (b) depicts an example of local minimum in greedy forwarding

with maximal decrease in the distance is selected as the next hop. Following this distance-decreasing policy, the packet can eventually reach the intended destination. This scheme is referred to as greedy forwarding/routing because in every step, the neighbor which maximally decreases the distance towards the destination is selected. In [6] authors used the same idea but instead of physical location of the nodes, virtual coordinates are assigned to them. In this routing, each node stores only the coordinates of its neighbors which is more memory-efficient than lookup-based routing. The cost of this efficiency is more computation complexity in determining the next hop (distance calculation instead of a lookup). Therefore, this routing replaces the lookups with more computation in forwarding. Figure 4.1(a) depicts an example of greedy forwarding (from 'd' to 'a') in a Euclidean space.

In greedy forwarding, packets might get stuck in a local minimum (void). This means that the current node is the closest node to the destination among all of its neighbors. Figure 4.1(b) depicts an example of a local minimum. Greedy embeddings have been proposed to solve this issue [7]. A greedy embedding for a given graph $G(V, E)$ into a metric space X, is a function from $V(G)$ to X such that for all graph nodes $s \neq t$, s has a neighbor u which decreases the distance toward t in metric space X. Several works proposed greedy embeddings in different metric spaces such as hyperbolic plane [8, 9] and multidimensional Euclidean spaces [10, 11]. Greedy forwarding based on these embeddings never gives rise to local minima and thus 100% successful delivery of the packets is guaranteed. Therefore, in geometric routing, a structured address space is required to guarantee successful packet delivery. However, such dependency on a structured space results in: i) possible deviation from shortest paths and ii) non-local changes with regard to the location of topological changes.

An efficient geometric routing is required especially in large networks where

scalability is the main concern. Such an efficient routing scheme should facilitate computation in forwarding plane to enable low-cost design while minimizing required memory in network nodes and act rapidly upon network changes with short convergence time.

Our contribution. In this paper, we investigate a simple and powerful greedy embedding. In this embedding, instead of relying on complex geometry, network nodes coordinates are derived from a spanning tree of the network. This decreases the overall overhead/complexity of the scheme compared to approaches based on hyperbolic [8, 9] or high-dimensional Euclidean spaces [10, 11]. The less complex the approach, the more likely it would be used in practice. The main goal of this paper is to illustrate that a geometric router can be implemented in a very low-cost way without sacrificing efficiency of the routing. We propose an efficient low-cost circuit to greedy forward the packets based on the calculated tree-coordinates. The cost of the proposed forwarder is measured in terms of area complexity of the hardware design. Routing efficiency is measured in terms of routing quality (deviation from the shortest path length) and the required storage for coordinate representation. The efficiency of the routing is compared with a state-of-the-art embedding in the hyperbolic plane proposed in [8]. Evaluation results indicate that the proposed scheme can be implemented through a very efficient low-cost design without degrading the routing performance in terms of stretch[2] while scaling significantly better in terms of coordinate precision compared to the hyperbolic embedding. Additionally, in [12], we have evaluated the convergence behavior of this scheme which shows good performance in terms of convergence time upon changes in the network topology. These make the scheme very suitable candidate for routing in large-scale networks.

Importantly, this paper is different from the existing work in the sense that it explores: i) the experimental execution of the mentioned embeddings on large-scale networks with respect to the resulting stretch and required coordinate precision, and ii) a novel low-cost hardware design of a greedy forwarder. To the best of our knowledge, no research has been conducted so far focusing on such design. We evaluate these topics and show that our scheme outperforms the hyperbolic-embedding in several aspects. The advantage of using the proposed embedding is apparent in the hardware implementation. The proposed scheme in this paper is applicable to a wide range of topologies, however this paper targets scale-free complex networks in general. A wide range of social, biological, technological and communication systems can be described as complex networks. Scale-free networks are one of the well-known and much studied classes of complex networks in which the nodes connectivities (nodes degrees) follow a power-law distribution. Such networks expand continuously by the addition of new nodes and new nodes attach preferentially to already well-connected nodes. Technological

[2]The ratio between the length of the path produced by greedy forwarding and shortest path length.

networks such as the router-level graphs and domain-level graphs in the Internet, peer-to-peer networks and the World Wide Web belong to this category of the networks. Graphs that represent such networks typically have thousands to millions of nodes and links. Therefore, all our evaluations are performed on large-scale scale-free topologies. This should enable better conclusions regarding the applicability of the proposed scheme on a wide range of real systems.

The rest of the paper is organized as follows. Section 4.2 explains the related work. In Sect. 4.3, the tree-based embedding is explained in detail. Section 4.4 describes the hardware design of a greedy forwarder based on the embedding. Section 4.5 includes the experimental results and Sect. 4.6 discusses existing challenges and future plans. Finally Sect. 4.7 concludes the paper.

4.2 Related work

Although many proposed approaches for geometric routing target different types of networks (e.g. unit-disk graph[3]) rather than scale-free topologies, it is beneficial to investigate the existing approaches used to avoid local minima to guarantee packet delivery. In the following we explain the existing techniques however, not all of them are applicable to scale-free topologies as they do not posses the desired graph properties.

Two groups of techniques have been proposed to solve the local minimum issue in geometric routing: i) face routing techniques and ii) greedy embeddings. In face routing techniques, the void is bypassed by routing around this area and greedy forwarding is resumed from the moment a closer node than the local minimum is reached [13]. However, such techniques suffer from several issues: i) the graph should be planar/planarized and ii) planarizing might not be feasible in every graph.

In greedy embeddings, nodes are embedded in a metric space in such a way that for every node there is always a distance decreasing neighbor toward every destination in the network. There are several works which propose greedy embeddings in different metric spaces. Works such as [8] and [9] proposed embeddings in hyperbolic space. In [14] authors proposed another embedding in the hyperbolic plane which requires $O(log\, n)$ bits (with n being the network size). [10] proposed an embedding in a Euclidean $O(log^2 n)$-dimensional space and a poly-logarithmic dimension embedding into Euclidean space was proposed in [11].

There are several proposed embeddings based on one or multiple spanning trees of the network [8, 15–17]. These works are interesting in the sense that they guarantee the packet delivery using a simple structure such as a tree. Some of them consider simple labeling/numbering of nodes instead of coordinates in a complex

[3]A graph $G = (V, E)$ is a unit-disk graph when $\forall u, v \in V : v \in N(u) \Leftrightarrow \delta(u, v) \leq 1$ in case G is embedded into a Euclidean space, with δ the Euclidean distance.

geometry. While [8] embeds a network spanning tree in the hyperbolic plane, [15] and [16] use a labeling of the nodes with the path from the tree root to each node. In [17] a different embedding compared to [15, 16] with poly-logarithmic memory complexity was proposed. In their work, the good performance in terms of stretch is achieved at the cost of higher complexity in the forwarding process and relying on multiple trees. In a recent work [18], enabling load balancing in geometric routing using multiple trees was proposed. The effect of using multiple spanning trees on network resiliency was investigated in [19]. In general, using multiple trees improves the geometric routing performance. This good performance is at the cost of: i) construction of multiple trees ii) increased storage requirements of the packet headers (as a node location is represented by multiple coordinates) and iii) increased computational complexity of the forwarding plane (as the next hop is determined considering multiple coordinates).

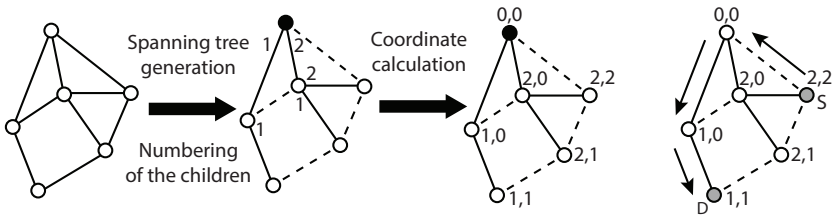
Through experimental results we show that despite the simplicity of the embedding based on a single tree, a good performance in terms of stretch and coordinates bit precision can be achieved. We then propose a circuit to implement a greedy forwarder based on this embedding. The simplicity of the embedding and forwarding result in a low-cost hardware design. Such an embedding with the low-cost hardware and stretch close to 1 is a very promising scheme for large-scale networks.

4.3 Greedy Tree-based geometric Routing (GTR)

GTR is composed of two components: i) tree-based greedy embedding and ii) greedy forwarding. In the embedding scheme, a spanning tree of the network is constructed and nodes coordinates are derived from this tree. These coordinates are then used by the forwarding component to forward the packets to the intended destinations. In the concept of geometric routing, to send a packet from node s to d, s should know the coordinate of d. Therefore, a mapping system to bind node identifiers to node coordinates is required. Such a mapping system can be based on a DNS-like mechanism. There are several proposals in the literature for the mapping systems [20] focusing on different aspects such as scalability, resiliency, security and end-host mobility. However, the detail of such mapping system is out of the scope of the paper and we assume there exists a mapping system in the network.

4.3.1 Tree-based greedy embedding

Instead of embedding network nodes to a complex metric space (e.g. hyperbolic), this paper evaluates the use of tree-coordinates. This will reduce the computational overhead and complexity of the overall scheme. As we will see in Sect. 4.5, this



(a) Tree-based greedy embedding (b) Greedy forwarding

Figure 4.2: The tree-based embedding is depicted in (a) and (b) illustrates an example of greedy forwarding

will not penalize the resulting performance. The approach relies on construction of a spanning tree of the network.

Given a spanning tree (see Fig. 4.2(a)), every node can be assigned a Coordinate Set (CS). The number of coordinates in the sets is determined by the depth of the tree. These sets can be derived using the following steps:

1. The root node sets all of its coordinates to zero: (0,...0), and each node numbers its children from 1 to d (d is the number of the node's children in the tree).

2. A node calculates the CSs of its children by putting the number assigned to each child in place of the first zero coordinate in its own CS, e.g. (1,0,...0) for the first child of the root node, and (d,0,...0) for the last.

In [15] and [16] similar coordinates in ad hoc and sensor networks were used. However, the routing in [15] only used the tree edges and is referred to as tree routing. They discussed several possibilities such as using multiple trees and shortcuts[4] to further improve the efficiency of their routing.

These coordinates reflect the location of the nodes in the tree. Therefore, the depth and the maximum degree of the tree determine the size of the CSs. Each coordinate in a CS should have enough bits to be able to show a value equal to the maximum degree of the tree. Experimental results in Sect. 4.5 confirm that given a suitable tree, the coordinate size differs significantly from the characterization of $O(n)$ bit required, made in [16] (n is the network size).

As investigated in [21], the choice of the spanning tree may impact the performance of the geometric routing. For this reason, we focus on a tree which has minimal depth with a root node which has large(st) degree in order to minimize memory requirements for coordinate representation and enable shorter paths. This can be done by selecting the node with maximum degree as the root of the tree

[4]Links which are not in the spanning tree.

and constructing a Breadth-First-Search (BFS) tree from that node. As our focus is on scale-free networks, such a tree construction method does not generate very deep branches because the average distance between nodes in scale-free networks is very short. In [12], we proposed a distributed algorithm to implement such a tree. In this algorithm, each node of the network has a unique identifier (ID) which is composed of two parts: i) the degree of the node and ii) a unique number to be used as tie breaker. These two parts are concatenated in such a way that a node with higher degree has a higher ID as well. Each node in the network initiates spanning tree generation considering itself as the root of the tree. As we need to generate a single spanning tree, all except one tree is suppressed based on a rule. Using the nodes IDs, the tree initiated by the root with the highest ID remains. Additionally, the level of the nodes in the selected tree is taken into account to enable BFS tree construction. This is done by sending ANNOUNCEMENT messages which include two parameters to indicate: (i) the ID of the root node initiating the tree and (ii) the level of the node receiving the message upon joining the tree. When an ANNOUNCEMENT(newroot,newlevel) from j arrives at i:

- if (newroot $>$ myroot) or (newroot $=$ myroot and newlevel $<$ mylevel): node i should suppress its current tree/location in the tree due to its lower priority. It updates its data structures and joins js subtree with newroot as the root and newlevel as the location in the tree. It then announces the new states to its neighbors except j.

Eventually all nodes receive the ANNOUNCEMENT message indicating the root node with the highest ID and abandon the lower priority trees. Note that the tree-based embedding and the spanning tree generation can be performed simultaneously [12]. The messages that are exchanged between network nodes to generate the spanning tree can include the calculated CSs. This means that once node i sends an ANNOUNCEMENT message to its neighbor j, it can calculate the corresponding CS (see Sect. 4.3.1) for j and add it to the same message. Upon arrival of this message, if j decides to change its parent to i, it should also update its CS to the one announced in the message. This way, once the tree is generated the coordinates are also calculated.

Each node announces its CS and ID to its neighbors (using ADVERTISEMENT messages) and nodes store the neighbors CS in a table to enable greedy forwarding.

4.3.2 Greedy forwarding based on coordinate sets (CS)

Once every node has its deduced CS using the above procedure, packets can be forwarded towards neighbors which (maximally) reduce the distance towards the CS of the destination node mentioned in a received packet. This process is different

from tree routing because the shortcut links can also be used. In this context, we propose to use tree-distance as metric. This refers to the hop count on the tree between two nodes and can be calculated as follows:

1. The closest common ancestor to both nodes is found.

2. The hop count of each node to the ancestor is counted.

3. The sum of these two hop counts determines the tree-distance between them.

This distance can be calculated easily using the assigned CSs to the network nodes. Consider CSs (2,1) and (2,2) in Fig. 4.2(a). Their longest common prefix (2) determines the CS of the closest common ancestor which is (2,0). The number of non-zero coordinates after the common prefix in each CS determines the hop count of each node to the common ancestor. Both CSs have only one non-zero coordinate after the common (2). Thus, the tree-distance between them is 2. Note that in case of no common prefix, the closest common ancestor is the root (0,0) and all non-zero coordinates of each CS should be counted.

Figure 4.2(b) depicts an example of greedy forwarding (from S = (2,2) to D = (1,1)) based on the derived CSs. Each node is aware of the CSs of its neighbors. In each node the tree-distance between every neighbor and D is calculated and the one with maximum decrease in the distance is selected as the next hop. In node S, the tree-distances (td) are as follows: td((0,0),(1,1))=2, td((2,0),(1,1))=3 and td((2,1),(1,1))=4. Therefore, (0,0) is selected as next hop. Following this greedy forwarding leads to the path depicted by arrows. This path is 1 hop longer than the shortest path ((2,2)→(2,1)→(1,1)) and one hop shorter than the path produced by tree routing ((2,2)→(2,0)→(0,0)→(1,0)→(1,1)).

4.3.3 Delivery guarantee

In this section, we explain that the tree-based embedding is a greedy embedding. Based on [8], if $H \subseteq G$ is a subgraph containing all vertices of G, then every greedy embedding of H is also a greedy embedding of G. In order to verify that the tree-based embedding is a greedy embedding, it suffices to show that for every path $s = s_0, s_1, ..., s_k = t$ in the tree, the inequality $td(s_0, t) > td(s_1, t)$ is satisfied (with $td(i, j)$ being the tree-distance between i and j). This is always satisfied, because there is exactly one path between every two nodes in a spanning tree of a network. Therefore, the hop count of the second node along the path (s_1) to the destination (t) is exactly one hop less than the hop count between the first node (s_0) and the destination (t). Thus distance can always be decreased by 1 by following the tree. Whether shortcuts exist or not does not change this fact, they only help getting closer to the destination "faster".

4.4 Hardware design of a greedy forwarder

As geometric routing is a recent paradigm, emerging in the last decade, no research has focused on the design of a circuit in hardware supporting greedy forwarding of the packets yet. In order to close the gap between geometric routing in theory and its applicability in practice, we propose a hardware design of a greedy forwarder.

The simple structure of the proposed coordinates and the distance function in the forwarding process (explained in Sect. 4.3) enables designing of a low-cost greedy forwarder. The cost of this design was quantified in terms of area complexity of the hardware and was compared to a state-of-the-art approach with virtual coordinates in the hyperbolic plane which is reported in Sect. 4.5. The main objective of these evaluations is to illustrate that a geometric router can be implemented in a very low-cost way without sacrificing efficiency of the routing.

A greedy forwarder, independent of the type of coordinates and the distance function, is composed of two major components: i) a distance calculator and ii) a comparator. The distance calculator calculates the distance between every neighbor and the destination of a packet and the comparator selects the minimum value among them. We propose a circuit for the tree-distance calculator. For the sake of clarity, the circuit for only one neighbor is explained which can be easily extended for more neighbors. In this circuit, given two CSs (the CS of the destination 'D' and a neighbor 'N') the tree-distance between them is calculated.

Figure 4.3 depicts the proposed design for a tree-distance calculator. The idea in this circuit is to find the first uncommon coordinate in the two CSs and start counting the non-zero coordinates from that location in each CS. As illustrated in this figure, the input CSs are serial. The reason is twofold: i) the packets arrive sequentially and thus the destination CS included in the packet arrives sequentially as well (one bit after the other) and ii) compared to parallel designs, bit-serial designs result in a huge reduction in the required hardware. To serialize the neighbor CS, it is stored in a shift register. In Fig. 4.3 the values at different stages of the architecture are marked for sample CSs including 3 coordinates each composed of 3 bits.

We first explain the upper part of the circuit. The inputs are fed into a XOR gate which detects the different bits in the two CSs. By feeding the output of the XOR to an OR gate together with a feedback from a flip flop (F.F), we get a signal in the output of the F.F.1 which is set to '1' when the first uncommon bit in the two CSs is detected and it remains '1' until the last bit of the CS. As the coordinates should be counted and not the bits in each CS, the output of the F.F.1 is fed to an AND gate together with a pattern which determines the end of each coordinate by a '1' bit. This pattern can simply be generated by a conventional modulo-K up counter (K is the number of bits in a coordinate). Such a counter starts counting from 0 and is reset when it reaches K. The outputs of F.Fs producing a logic '1'

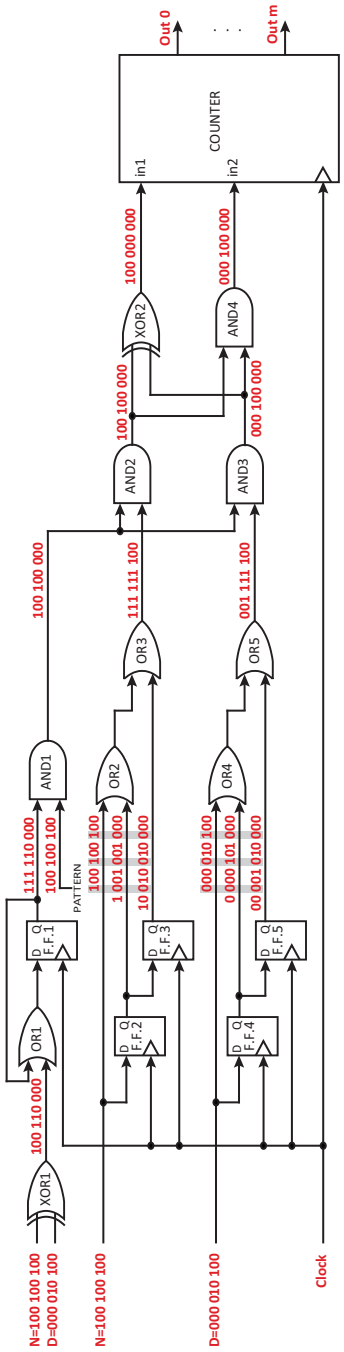


Figure 4.3: The tree-distance calculator circuit

when the number in the counter is K are fed to an AND gate whose output is tied to the reset pins of the F.Fs in the counter. Using another AND gate with its inputs tied to the outputs of F.Fs producing a logic '1' when the number in the counter is $K-1$, the desired pattern is generated as this AND's output.

The lower part of the circuit guarantees that only non-zero coordinates in each CS are counted. Note that in the tree-based embedding, as each node numbers its children starting from 1, it is impossible to have a zero coordinate between non-zero ones in a CS. Therefore, including such a logic is sufficient to count the hop counts correctly. The design of this part is dependent on the number of bits in a coordinate. Consider CS of the neighbor 'N' in Fig. 4.3. This CS should be delayed $K-1$ times using $K-1$ F.Fs (K is the number of bits in a coordinate). The output of each F.F is the input of the next one (see F.F.2 and F.F.3). The outputs of F.Fs and the original CS 'N' are combined. This can be done by a K-way OR gate or $(K-1)$ 2-input OR gates (see OR2 and OR3). If there is at least one bit equal to '1' in each coordinate of 'N', the output of OR3 is '1' at the location of the last bit of that coordinate. These locations are highlighted in the inputs of the OR gates (OR2 and OR3). The same logic is repeated for CS of the destination 'D'. Applying the output of OR3 (OR5) and the output of AND1 to AND2 (AND3) results in a signal which is '1' only at the end of each non-zero coordinate and only if the first uncommon coordinate is detected. We refer to the outputs of AND2 and AND3 as count-pulse1 and count-pulse2 respectively. Counting the number of '1's in each count-pulse and calculating the sum of these two numbers determine the tree-distance between the two CSs. Instead of using two counters and an adder, we propose a counter which is capable of parallel counting. Given the two count-pulses if both are '0' no counting should take place, if only one of them is '1' the counter should count 1 and if both are '1', the counter should count 2. Using AND4, we determine where both count-pulses are '1' and using XOR2, we determine where only one of them is '1'. Feeding the outputs of AND4 and XOR2 to the inputs of the proposed counter, the hop counts of both CSs are counted.

Figure 4.4 depicts the circuit of a 3-bit counter which is a modification of a conventional counter based on D-flip flops. Although asynchronous counters are less complex and require fewer components, we selected the synchronous design

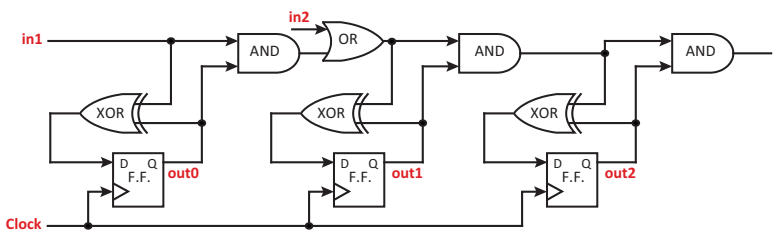


Figure 4.4: The 3-bit counter circuit

as it is faster and more reliable. In this counter an OR gate is added to the second stage of the counter. In case 'in1' is '1' the counter counts 1 and if 'in2' is '1', it counts 2. In our design 'in1' and 'in2' of the counter are not '1' at the same time.

4.5 Performance evaluation and analysis

GTR was evaluated in a custom simulation environment developed in Python/C++ on a set of network topologies defined by the Barabasi-Albert (B-A) model [22] which generates random scale-free networks. These topologies were evaluated in different scales.

We evaluated GTR in terms of stretch, coordinate size and area complexity of the proposed hardware design. In the experiments, the selected spanning tree for embedding is a BFS tree rooted at the maximum degree node.

In order to avoid that we compare GTR only to 'legacy' routing mechanisms (i.e. shortest path routing), we compare it to a state-of-the-art geometric routing scheme in order to make fair comparisons and be able to estimate performance difference of routing schemes of the same class. We selected hyperbolic-embedding of [8] for comparison for several reasons: i) it is based on a single spanning tree (other works such as [16, 17] benefit from multiple trees in the network) ii) it is applicable to any connected topology and provides 100% successful delivery (no need for planarization and face routing techniques) and iii) it results in good performance in terms of stretch (given a suitable tree). We want to explore if the performance is the result of complicated hyperbolic structure/computations or a simple tree labeling with simple distance calculation/forwarding can be equally(more) efficient.

4.5.1 Routing stretch evaluation

Routing stretch is defined as the ratio between the length of the path produced by greedy forwarding and the length of the shortest path for the same source-destination pair. We evaluated GTR performance in terms of stretch to determine how the path length is affected using this scheme. We compared the stretch of GTR with: i) the stretch of greedy forwarding based on the hyperbolic-coordinates [8] and ii) the stretch of tree routing [15] when shortcuts were not used.

In this experiment, we calculated the stretch for every pair of nodes in the network. The average stretch for each topology is calculated and this is repeated for 10 B-A topologies and the average over all of them is reported. As we see in Fig. 4.5, the stretch of GTR is very similar to the routing based on hyperbolic-embedding and also outperforms tree routing. In Fig. 4.6, we report the stretch percentile of GTR on large-scale B-A topologies using Boxplot. Boxplots are used to represent the stretch values through their quartiles. The values on top of the

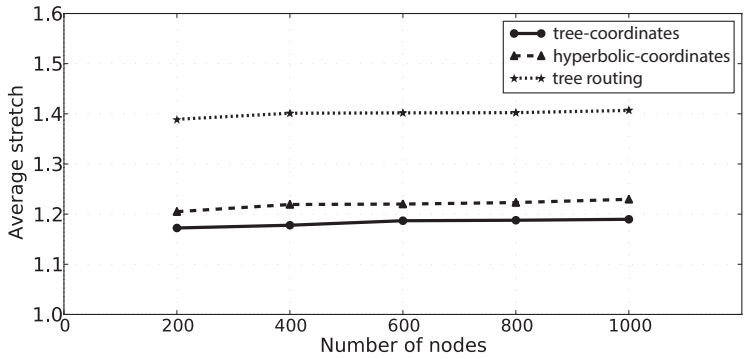


Figure 4.5: Average stretch in B-A networks

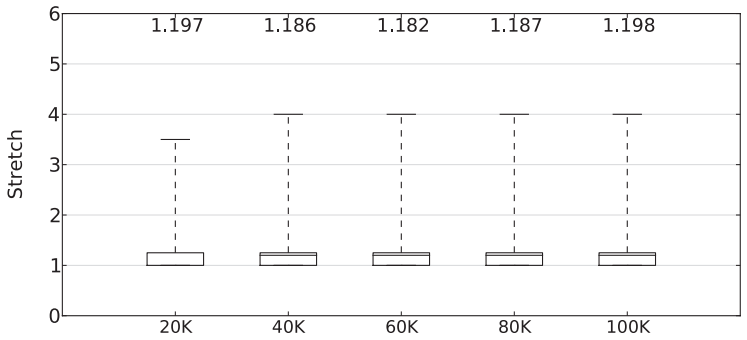


Figure 4.6: Stretch percentile of GTR in B-A networks

graph are average stretch. The results show that the tree-based embedding scales very well with the increase in the number of nodes and the average stretch remains close to 1. In almost all topologies, up to 75 percentile of the stretch is 1.25 and the maximum stretch is 4.

4.5.2 Coordinate size evaluation

We calculated the required number of bits for tree-based coordinate representation and compared it with the minimal floating point bit precision for hyperbolic-coordinate which results in a valid greedy embedding. As a hyperbolic-coordinate consists of 2 floating points, the total size of a coordinate is twice the minimal bit precision.

Figure 4.7 depicts the obtained results for B-A networks of varying scale between 50 nodes and $20K$ nodes, and the required number of bits for representing the coordinates, for both tree and hyperbolic coordinates. The difference in the bit

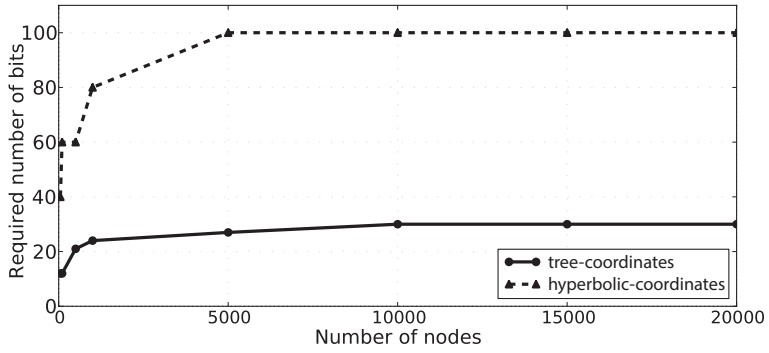


Figure 4.7: Required number of bits in coordinates in B-A networks

requirement in the two embeddings is significant and the tree-based coordinates use the address space more efficiently. The results show that given a suitable spanning tree as basis for the embedding, that in scale-free topologies, the required space-complexity for storing tree-based coordinates is significantly more scalable than the hyperbolic-coordinates.

4.5.3 Area complexity evaluation

We evaluated the performance of the proposed circuit in terms of required silicon area and compared it with a hyperbolic-distance calculator. With this comparison, we determine how much chip area is saved in our design.

The hyperbolic distance function in the Poincare Disk (a model in the hyperbolic geometry used in [8]) is defined by $d(u,v) = arccosh(1 + \delta(u,v))$ with $\delta(u,v) = 2\frac{\|u-v\|^2}{(1-\|u\|^2)(1-\|v\|^2)}$, in which $\|.\|$ refers to the Euclidean norm. This function is composed of arccosh, multiplications/divisions and additions/subtractions. As in greedy forwarding, we do not need to find the exact distance between points, we can simplify the distance function. Also implementing complex mathematical functions such as arccosh and division consumes a large silicon area. Because $arccosh$ is a monotonic function, we can avoid this calculation as it would not change the resulting comparison. The divisions can be removed for the same reason. As the coordinate of the destination is the common factor in every distance calculation, the division to this coordinate can be removed. The division to the second factor can be replaced by a multiplication to the inverse of that factor, but this value should be stored for every neighbor. With these simplifications, there are still 3 multiplications and 3 additions required for distance calculation. We consider these modules in the calculation of the silicon area of the hyperbolic-distance calculator. Note that for comparison, we considered a sequential design for this calculator.

Table 4.1: Logic modules normalized area

Element	AND	OR	XOR	D-FF	Full adder
Area	1	1	1.99	4.24	5.24

Table 4.2: Normalized area(μm^2) of the tree (T) and hyperbolic (H) distance calculator for different scale B-A networks

	1K	5K	10K	15K	20K
T	128.97	144.35	155.83	155.83	155.83
H	18773.6	21517.6	21517.6	21517.6	21517.6

The required silicon areas of the designs are derived from their basic logic modules. To evaluate the area complexity of our design against the hyperbolic-distance calculator, the area of the logic cells are normalized with an AND gate from the same standard cell library. This benchmarking method has been widely adopted in the literature. The normalized areas of different logic cells are reported in Table 4.1 based on [23]. The required area of both designs in different scales from $1K$ to $20K$ nodes are reported in Table 4.2. For the adders in the hyperbolic-based design, we used the area of a full adder and a flip flop (i.e., a sequential full adder), and the area of the multipliers are calculated based on [24]. The forwarder based on the tree-based embedding is significantly more area-efficient than hyperbolic-based design. The evaluations in this section indicate that a geometric router can be implemented in a very low-cost way without sacrificing efficiency of the routing.

4.6 Challenges and future work

One of the challenging tasks in geometric routing is dealing with dynamics in the topology. In [12] we observed the convergence behavior of the geometric routing scheme in case of topology changes which was dependent on the location of the changes in the topology. As the embeddings are based on the connectivity of a spanning tree, a single failure in the tree might change the coordinates of many nodes. However, the results in [12] indicated that although the number of affected nodes might be high, the corresponding convergence is quite fast in networks with 1000 nodes. The number of affected nodes in very large networks (o(100K) nodes) could be quite high and thus, the convergence time might be very large as well. As a solution, protection techniques can be used to avoid changes in the coordinates as we successfully demonstrated in [25] and [26]. A clever combination of the protection schemes and coordinate re-calculation technique can lead to good performance in terms of stretch and convergence time [12]. Another solution to improve the convergence behavior of geometric routing in large-scale networks

involves using multiple trees as we proposed in [19]. Minimizing the overlap of the constructed trees, there is a high chance that at least one tree remains unaffected upon failures in the network and greedy forwarding remains successful without any coordinate re-calculation. However, the complexity of the scheme may slightly increase.

Another challenge is to provide a scalable mapping system to bind node identifiers (e.g. existing IP addresses) to node locators (e.g., coordinates). Although there exist several proposals in the literature, there are still some open issues towards a scalable mapping system with fast convergence [20].

In the proposed scheme, although shortcut links are used to forward the packets to their intended destination, it is still possible that some traffic is routed towards the root of the tree in order to reach the destination. This might lead to congestion in the root node or the links/nodes close to that. In order to avoid such an issue, we investigated the usage of multiple trees to enable load balancing in geometric routing [18]. Using multiple trees, in every step, the neighbor with less loaded link is selected as the next hop. This way the congestion in different parts of the network is avoided.

Finally, the scheme should be investigated thoroughly in terms of security vulnerability and possible solutions. One of the main vulnerability in the proposed scheme is that nodes might claim false degrees and the frequent appearance/disappearance of such nodes leads to frequent global coordinate re-calculation if the announced degree is higher than the degree of the current root node. Using protection mechanisms [25, 26] is one way to avoid global coordinate re-calculations. However, as explained in [12], there is a trade-off between communication cost and stretch/node reachability when protection is used. This means that when there is a change in the network, the pre-determined backup paths are used which causes no communication cost but the stretch and node reachability are negatively affected. A possible solution to improve this performance is to have (partial/global) coordinate re-calculation periodically, upon reaching a certain threshold or based on some condition in the network. The second approach involves using authentication mechanisms. This way we avoid that any malicious node becomes the root of the tree and triggers the global coordinate re-calculation and only authenticated nodes can be the root node. As future work, we will investigate possible approaches to solve the security vulnerability of the scheme.

4.7 Conclusion

We investigated a simple but powerful approach to calculate virtual coordinates for network nodes in geometric routing. Greedy forwarding based on these coordinates guarantees 100% successful delivery of the packets to their destination. The coordinates are based on a spanning tree of the network. We proposed a novel low-

cost circuit to greedy forward the packets based on the deduced tree-coordinates. We evaluated the tree-based embedding thoroughly in terms of stretch, coordinates bit requirement, and silicon area complexity of the proposed hardware design and compared it with an existing hyperbolic-embedding. The results showed that good routing performance can be achieved without complex coordinate and distance calculations, with an efficient hardware node design and more scalable memory requirements for coordinate representation. To validate the feasibility of the scheme on large networks, some of the simulation experiments were performed on topologies of up to o(100K) nodes. These results bring geometric routing based on tree-coordinates to the pole position as a memory-scalable scheme for routing in large-scale complex networks with low stretch and low (hardware) complexity.

Acknowledgment

This work is partly funded by the European Commission through the EULER project (grant 258307), part of the Future Internet Research and Experimentation (FIRE) objective of the Seventh Framework Programme (FP7).

References

[1] B. Karp and H. Kung. *GPSR: greedy perimeter stateless routing for wireless networks*. In Proceedings of the 6th annual international conference on Mobile computing and networking, pages 243–254. ACM, 2000.

[2] M. Boguna, F. Papadopoulos, and D. Krioukov. *Sustaining the Internet with hyperbolic mapping*. Nature Communications, 1(6):1–8, 2010.

[3] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch. *Information centric networking in the IoT: experiments with NDN in the wild*. arXiv preprint arXiv:1406.6608, 2014.

[4] W. Tavernier, S. Sahhaf, D. Colle, M. Pickavet, and P. Demeester. *Towards Content-Centric Geometric Routing*. In Communications and Vehicular Technology in the Benelux (SCVT), 2014 IEEE 21st Symposium on, pages 133–138. IEEE, 2014.

[5] S. Sahhaf, D. Papadimitriou, W. Tavernier, D. Colle, and M. Pickavet. *Experimentation of geometric information routing on content locators*. In Network Protocols (ICNP), 2014 IEEE 22nd International Conference on, pages 518–524. IEEE, 2014.

[6] R. Fonseca, S. Ratnasamy, J. Zhao, C. Ee, D. Culler, S. Shenker, and I. Stoica. *Beacon vector routing: Scalable point-to-point routing in wireless sensornets*. In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2, pages 329–342. USENIX Association, 2005.

[7] C. Papadimitriou and D. Ratajczak. *On a conjecture related to geometric routing*. Theoretical Computer Science, 344(1):3–14, 2005.

[8] R. Kleinberg. *Geographic routing using hyperbolic space*. In INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, pages 1902–1909, 2007.

[9] A. Cvetkovski and M. Crovella. *Hyperbolic embedding and routing for dynamic graphs*. In INFOCOM 2009, IEEE, pages 1647–1655, 2009.

[10] R. Flury, S. Pemmaraju, and R. Wattenhofer. *Greedy routing with bounded stretch*. In INFOCOM 2009, IEEE, pages 1737–1745, 2009.

[11] C. Westphal and G. Pei. *Scalable routing via greedy embedding*. In INFOCOM 2009, IEEE, pages 2826–2830, 2009.

[12] S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester. *Experimental validation of resilient tree-based greedy geometric routing*. Computer Networks, 2015.

[13] A. Maghsoudlou, M. St-Hilaire, and T. Kunz. *A Survey on Geographic Routing Protocols for Mobile Ad hoc Networks*. Systems and Computer Engineering, Technical Report SCE-11-03.–Carleton University.–2011.–49 p, 2011.

[14] D. Eppstein and M. Goodrich. *Succinct Greedy Geometric Routing Using Hyperbolic Geometry*. IEEE Transactions on Computers, 60(11):1571–1580, 2011.

[15] E. Chávez, N. Mitton, and H. Tejeda. *Routing in wireless networks with position trees*. In Ad-Hoc, Mobile, and Wireless Networks, pages 32–45. Springer, 2007.

[16] M. Tang, H. Chen, G. Zhang, and J. Yang. *Tree Cover Based Geographic Routing with Guaranteed Delivery*. In Communications (ICC), 2010 IEEE International Conference on, pages 1–5. IEEE, 2010.

[17] J. Herzen, C. Westphal, and P. Thiran. *Scalable routing easy as PIE: A practical isometric embedding protocol*. In Network Protocols (ICNP), 2011 19th IEEE International Conference on, pages 49–58. IEEE, 2011.

[18] R. Houthooft, S. Sahhaf, W. Tavernier, F. De Turck, D. Colle, and M. Pickavet. *Robust Geometric Forest Routing with Tunable Load Balancing*. In Proceedings of INFOCOM 2015, 2015.

[19] R. Houthooft, S. Sahhaf, W. Tavernier, F. De Turck, D. Colle, and M. Pickavet. *Fault-tolerant Greedy Forest Routing for Complex Networks*. In Proceedings of 6th International Workshop on Reliable Networks Design and Modeling (RNDM), 2014.

[20] M. Hoefling, M. Menth, and M. Hartmann. *A Survey of Mapping Systems for Locator/Identifier Split Internet Routing*. IEEE Communications Surveys and Tutorials, 15:1842–1858, 2013.

[21] A. Cvetkovski and M. Crovella. *On the choice of a spanning tree for greedy embedding of network graphs*. Networking Science, 3(1-4):2–12, 2013.

[22] A. Barabási and R. Albert. *Emergence of scaling in random networks*. Science, 286(5439):509–511, 1999.

[23] *Standard cell library*. www.zhuhai.gov.cn/image20010518/6428.pdf.

[24] M. R. Meher, C. C. Jong, and C.-H. Chang. *A High Bit Rate Serial-Serial Multiplier With On-the-Fly Accumulation by Asynchronous Counters*. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 19(10):1733–1745, 2011.

[25] S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester. *Link failure recovery technique for greedy routing in the hyperbolic plane*. Computer Communications, 36(6):698–707, 2013.

[26] S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester. *Single failure resiliency in greedy routing*. In Design of Reliable Communication Networks (DRCN), 2013 9th International Conference on the, pages 306–313. IEEE, 2013.

5

Routing at large scale: advances and challenges for complex networks

This chapter presents an overview of the existing routing schemes. Different experimental results and analytical studies, performed so far, are analyzed to enable better and more careful conclusions regarding the applicability of these schemes to large-scale settings. The identified trends and trade-offs better positions the proposed geometric routing scheme among other alternatives. In this study, several open problems are identified which provide guidelines for future research directions.

⋆ ⋆ ⋆

S. Sahhaf, W. Tavernier, D. Papadimitriou, D. Careglio, A. Kumar, C. Glacet, D. Coudert, N. Nisse, L. Fábrega, M. Camelo, P. Vilá, P. Audenaert, D. Colle, P. Demeester

Submitted to IEEE Network, Aug. 2016.

Abstract A wide range of social, technological and communication systems can be described as complex networks. Scale-free networks are one of the well-known classes of complex networks in which nodes degree follow a power-law distribution. The design of scalable, adaptive and resilient routing schemes in such networks is very challenging. In this article we present an overview of required

routing functionality, categorize the potential design dimensions of routing protocols among existing routing schemes and analyze experimental results and analytical studies performed so far to identify the main trends/trade-offs and draw main conclusions. Besides traditional schemes such as path-vector routing, the article pays attention to advances in compact routing and geometric routing since they are known to significantly improve the scalability in terms of memory space. The identified trade-offs and the outcomes of this overview enable more careful conclusions regarding the (in-)suitability of different routing schemes to large-scale complex networks and provide a guideline for future routing research.

5.1 Introduction

Complex networks refer to large, dynamic networks consisting of potentially billions of nodes and links which are used to describe a wide range of social, biological, technological and communication systems. Scale-free networks as one well-known/much studied class of complex networks have degree distribution[1] that follows a power-law function. In such networks, new nodes attach preferentially to already well-connected nodes. The network of Autonomous Systems[2] (ASes) forming the core of the Internet graph, is an example of such networks. Routing[3] in these networks is challenging because of (i) the size of the network, and (ii) the properties and performance expected from these networks, particularly, any-to-any connectivity, availability, and reliability.

Routing research has evolved very pragmatically in communication networks from small scale to larger scale in technologies including wireless, ad-hoc/sensor networks, the Internet, etc. Since new networks of increasing scale are popping up every day (e.g., IoT), it is important to consider clean-slate approaches considering the entire design space of routing paradigms to avoid getting 'trapped' again in legacy protocols/paradigms.

In this paper we try to open this design question by clearly and cautiously categorizing/grouping the potential design dimensions of routing protocols among existing routing schemes (traditional ones as well as novel ones), analyzing experimental results performed so far, and drawing some main conclusions, guidelines and open challenges for routing schemes in future settings.

This article synthesizes the fundamental aspects of routing schemes for complex networks, as well as lessons learned from experimental routing research stemming from the EULER project[4]. Particular attention will be given to (i) new classes

[1] The probability that a node selected uniformly at random has a certain number of links.

[2] In the Internet, an autonomous system is a single network or a group of networks which is managed and supervised by a single administrative entity or organization.

[3] The process of finding/selecting paths between given nodes of a communication network.

[4] http://www.euler-fire-project.eu/

of path-based routing schemes,[5] (ii) new routing paradigms subdivided into locator space dependent[6] and locator space independent and (iii) a new route discovery scheme in which networks' structural properties are used. Additionally, a brief overview of path-vector schemes improvements and routing advances in delay/disruption-tolerant networks (DTN) and peer-to-peer overlays is provided.

This work presents an overview of the design dimensions of routing protocols, challenges, and a perspective/guideline for future routing research in complex networks.

5.2 Routing design problem

Routing is the process of finding/selecting paths between given nodes of a communication network. A path is a finite sequence of nodes from a source towards a destination node. The distance between two nodes is the sum of the cost of the links used along its shortest interconnecting path. The diameter of a network ($D(G)$) is the maximum distance of any two nodes.

5.2.1 The routing function

Routing is decomposed into the following functionalities:

- **Identification and location.** In order to derive paths between nodes of a topology, nodes should be identified. A node identifier might be a number/label. Identification functionality does not necessarily imply a location within the topology[7]. Thus the routing should focus on (i) structuring the topological space into addresses/locators, (ii) mapping the identifier to the network nodes locator when needed.

- **Discovery/distribution.** This is required to discover/distribute information related to (i) routes or (ii) topology characteristics. It can be push-based (local changes are distributed towards remote nodes) or pull-based (on demand search) or a combination.

- **Route determination/calculation.** This functionality determines routes towards a given destination. This operation can involve routing path calculations superimposed by policies and/or route selection/filtering functionality, or can be guided by a substructure of the discovered topology (e.g., network spanning tree).

[5] Schemes which maintain the path information to reach a destination.

[6] Routing paradigms which derive paths based on locators/coordinates assigned to network nodes in a metric space.

[7] Note that traditional IP addresses fulfill both roles, leading to significant issues regarding node mobility, multi-homing, etc., as confirmed by the invention of, e.g., LISP and HIP protocols. The impact of Locator/Id separation is detailed in [1].

- **Routing entry determination.** This determines routing entries based on the outcome of route determination functionality. The outcome can be a selected set of potential next hops for given network locations, or a procedure to decide how such a routing entry can be determined on the fly.
- **Multicast.** Multicast routing is a distributed algorithm that allows any node to route multicast traffic to a group of destination nodes, called multicast group. To enable point-to-multipoint traffic distribution, the multicast routing protocol builds a tree between the source and the multicast group called Multicast Distribution Tree (MDT). Multicast routing is (re-)gaining interest given the increasing popularity of multimedia streaming/content traffic, since it yields bandwidth savings competing with/complementing cached content distribution techniques. Multicast tree membership management handles the multicast membership, which involves the join/subscription and leave/un-subscription actions.
- **Policy.** Policing routing including routing-engineering, traffic-engineering and administrative policies affects both local processing performance and the overall performance resulting from local decisions. Limiting policing capabilities leads to local performance increase but may decrease global performance, while increasing flexibility may increase global performance. From the routing design perspective, this leads to a major consequence: starting from a relatively simple routing procedure and requiring homogeneous policy strategy (which is unlikely in organically controlled organizations such as the Internet) may lead to detrimental effects in terms of performance.

A routing function determines the next-hop along a path from a source towards a destination. This path is determined by the routing schemes which are described according to the following key-properties:

- **uncoordinated vs. coordinated routing decision**: In an uncoordinated routing each node takes its routing decision independently of others though each participating entity collaborate to global shared objectives, such as connectivity and availability.
- **distributed vs. centralized**: Unlike a centralized algorithm, a distributed algorithm is executed locally at each node and independently of other nodes. They are different from uncoordinated algorithms as distribution is about computation while the latter refers to routing decision.
- **control vs. data-driven**: Control-driven algorithms are triggered by independent processes exchanging control information, while data-driven algorithms only trigger routing algorithms when data packets travel through the network.

- **deterministic vs. statistic**: In deterministic routing the path determination between a set of nodes are fixed and independent of time or particular data within control/data traffic between nodes. Statistical algorithms introduce a degree of randomness within the generated routes.

- **stateful vs. stateless**: Unlike stateless algorithms, stateful algorithms require the maintenance of states to operate, e.g., for storing information related to the interaction with other nodes.

We mainly focus on the advances to control-driven, stateful distributed routing (meaning routing information is exchanged via dedicated messages, nodes store routing table (RT) entries and their computation are distributed), the other dimensions being dependent on the schemes.

5.2.2 Trade-offs in routing

When routing at large-scale (above 10000 nodes), three cost dimensions can be identified:

1. **Memory cost.** The memory space in a node required to store the routing information used by the routing algorithm (input) and to store the RTs (output).

2. **Stretch cost.** Stretch is the ratio between the length of a path generated by the scheme and the corresponding shortest paths. The stretch of a routing scheme is the highest stretch among all source-destination pairs.

3. **Adaptation cost.** Communication complexity which refers to (i) the number of exchanged messages between nodes for the computation of the RT entries and (ii) convergence time as the difference in time between the sending of the first message and the reception of the last message during the execution of the routing algorithm.

Upon designing a routing scheme, a trade-off should be taken into account between different criteria depicted in Fig. 5.1. When designing a 'static' scheme, there is a trade-off between memory space and stretch. When distributing such a scheme, communication cost becomes an additional criteria impacting the previous trade-off; and moving to an adaptive scheme, computational complexity adds to it. Fig. 5.1 also shows that when designing a 'dynamic' scheme, both distribution and adaptation should be considered. Computational complexity is not the main criteria when moving to a distributed scheme (indicated with gray color).

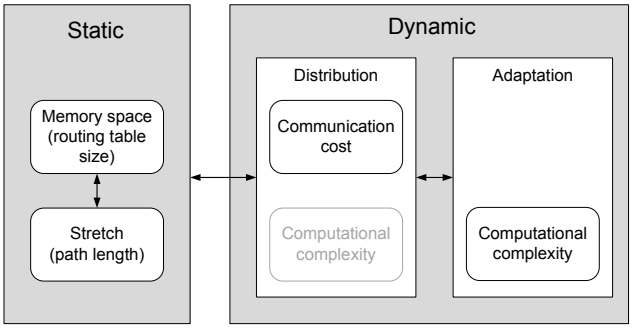


Figure 5.1: Fundamental trade-offs in routing schemes

5.2.3 Challenges in the Internet routing system

Since we target large-scale complex networks, in this section we explain the main open challenges in the Internet routing as it is one known large-scale scale-free network in nature/technical domain.

The current Internet routing follows a 2-level hierarchy: routing between almost 60 K ASes in the core (forming a scale-free network), and routing within the ASes. The true challenge is in the inter-AS routing, driven by the Border Gateway Protocol (BGP) which is a path-vector routing protocol[8], exchanging network reachability information with peering BGP routers. Reachability information includes an AS path listing the sequence of AS numbers traversed by the BGP route advertisement comprising reachability information from the originating AS. Discovered path information is used by BGP routers for constructing the AS connectivity graph for this reachability, and to detect/avoid routing loops by performing a route selection process combined with shortest path routing. Policies are determined to maintain business relationships between peering ASes or by load balancing strategies during high-traffic periods. BGP is subject to the Path Exploration phenomenon: BGP routers may announce as valid, routes that are affected by a failure which are withdrawn later during subsequent routing updates. This is (one of) the main reasons for the large number of update messages received by inter-domain routers.

Internet routing is significantly challenged by the increasing number of routers, ASes, and routes. This situation is exacerbated due to site multi-homing, AS multi-homing, traffic engineering and the increasing need for connectivity availability from the increasing number of connected hosts.

[8] A routing protocol which maintains path information to reach the destinations. This information is updated dynamically. Using this scheme, the routing tables include the destination network, the next router and the path to the destination. It is easy to detect routing loops and discard the update messages which are looping in the network.

The main issues in the Internet architecture are the scalability, convergence, and stability properties of its inter-domain routing. Solving them requires addressing multiple dimensions altogether, e.g. the RT size growth resulting from a large number of message exchanges induced by topological/policy changes. Both dimensions increase memory and processing requirements of routing engines. Solving the scalability of the Internet routing, considering its dynamics, is challenging. Convergence time should not be delayed whereas scalability improvement minimizes the number of exchanged messages preventing routers overload. Also, addressing routing stability consistently with planned BGP routing policy implies eliminating 'wedgies', i.e., non-deterministic/unintended but stable routing states, and 'dispute wheels', i.e., non-deterministic/unintended but unstable states. However, when considering the existing Internet routing and the considerable research efforts made to improve it, one might wonder when considering a complex network with similar conditions/constraints as the Internet, whether the only feasible solution is a local policy-based path-vector routing system or would there be a more promising model beyond routing IP packets?

5.3 Routing schemes

We consider the following classification: (i) path-vector schemes improvements, (ii) routing schemes (clean-slate approaches) in complex networks, and (iii) routing schemes in DTN and P2P networks.

5.3.1 Path-vector schemes improvements

Numerous enhancements to path-vector schemes have been proposed over the last twenty years. BGP is an example of a path-vector protocol driving the inter-AS routing in the Internet. Many of the enhancements relate to BGP dynamic properties. Examples include (i) enhanced path vector routing protocol (EPIC) which annotates the AS paths with additional 'path dependency' information to reduce convergence time, (ii) BGP with Root Cause Notification reduces the convergence time by announcing the root cause of a link failure location, (iii) Path Exploration Damping augments BGP for selectively damping the propagation of path exploration updates. Recently, new route selection schemes are proposed to improve route stability in BGP [2].

5.3.2 Routing schemes in complex networks

Table 5.1 positions our proposed routing schemes[9] (clean-slate approaches) with respect to their adaptation capability to topology/policy dynamics and the dis-

[9]Routing schemes proposed within the FP7 EULER project.

Table 5.1: Position of different routing schemes with respect to the capability to adapt to dynamics and distribution. Static schemes have fixed routes which do not adjust in case of a change in the network. Fault-tolerant/adaptive refers to the capability to react to changes in the network and adjust the routes.

	Centralized	**Distributed**	**Static**	**Fault-tolerant/ adaptive**
Centralized Compact routing (AGMaNT)	✓	-	✓	-
Distributed Compact Routing (DCR) [3]	-	✓	✓	-
Greedy Compact Multicast Routing (GCMR) [4]	-	✓	-	✓
Geometric Tree-based greedy Routing (GTR) [5]	-	✓	-	✓
Word-Metric-based Greedy Routing (WMGR) [6]	-	✓	-	✓
Geometric Coordinate-Labeling Scheme (GCLS) [7]	-	✓	-	✓
Route Discovery (RD)	-	✓	-	✓
Border Gateway Protocol (BGP)	-	✓	-	✓

tribution of the computation/decision process. BGP is reported for comparison. As mentioned earlier, these are control-driven, stateful and distributed routing schemes.

5.3.2.1 Compact routing

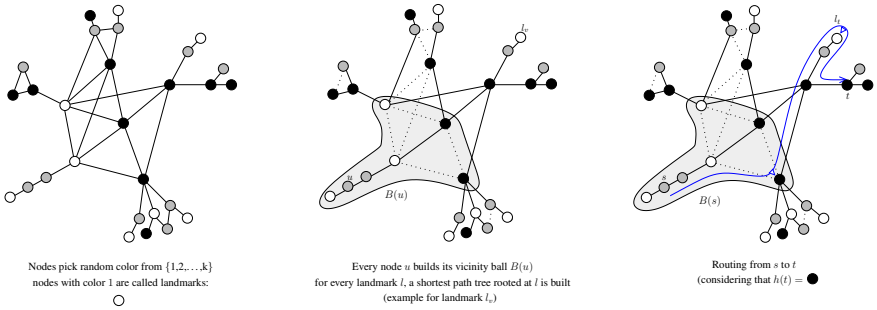
The goal of compact routing is to reduce the amount of storage space in each node. It is challenging to design algorithms with a good trade-off between the memory space and the resulting stretch. The theoretical bounds concern worst-case analysis, and one of the contribution of EULER is to show that much better trade-offs are achieved in actual networks. We have investigated two research directions: unicast and multicast compact routing.

Distributed Compact Routing (DCR). We proposed an asynchronous distributed compact routing scheme, DCR [3], based on the centralized scheme, AGMaNT.

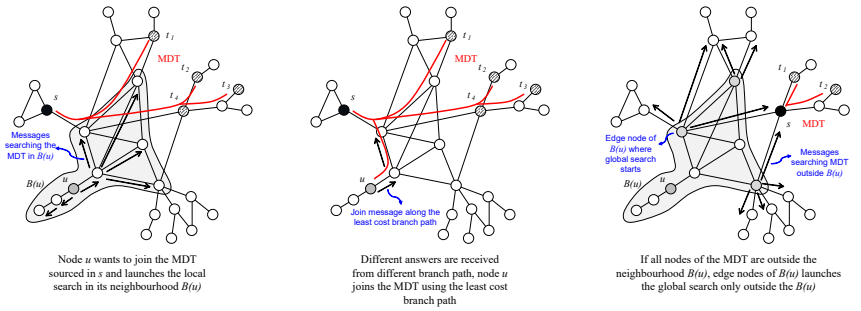
In this scheme, each node in the network picks a color from a small set of colors at the same time. All nodes share a hash function that maps identity/address to an element of the color set.

Vicinity of a node is the minimal set of close nodes that contains at least one node of each color. The size of this set can be proven to be proportional to the number of colors. Each node stores a direct route to the nodes in its vicinity. Moreover, for all other nodes having a hash value equal to its color it stores the address of/route to a landmark that has that node in its vicinity. When a node has to forward a packet, it first checks whether it has a route based on its identifier. If not, the hash is determined and the packet is forwarded to a node in the vicinity with the same color. The routing path via a landmark has to be encoded in the header to allow routing which imposes storing a compressed path. To this end, we use a compact routing scheme dedicated to trees. Fig. 5.2a visualizes the steps of DCR.

Greedy Compact Multicast Routing (GCMR). GCMR [4] is a multicast scheme which minimizes the RT size of each node at the expense of (i) paths with relative small deviation from the optimal stretch and (ii) higher communication cost compared to shortest path tree. GCMR minimizes the storage of routing information by requiring only neighbor-related information. Thus, it does not rely on the construction of global structures such as sparse covers or trees. To limit the communication cost, the routing information needed to reach a given multicast source is acquired by means of an incremental two-stage search process: firstly the joining node searches nodes belonging to the multicast tree in its neighborhood (local search), in case of unsuccessfulness, the search is then continued over the remaining unexplored topology (global search). The request message comprises a path budget which is used to limit the distance traveled by requests in the local search. Starting from the joining node, this value is decremented in every intermediate



(a) Node s does not know about node t. It forwards the packet to w, the closest node of color $h(t)$ it knows. From w routing is done via a shortest path in the tree rooted in l_t using compact routing techniques for trees.



(b) Two scenarios of local search and global search for joining the MDT is depicted.

Figure 5.2: Variants of compact routing. (a) depicts different steps in DCR, (b) illustrates an example of GCMR.

node.

The joining node sends a request to its neighbors and starts a timer. The neighbors propagate the message following a split horizon[10] until it reaches a node which is in the MDT or it is an edge node of the neighborhood (i.e., path budget reaches 0). The receiving node sends back a reply indicating whether it belongs to MDT or not. Based on this information all the nodes along the path to the joining node compute their path cost. At the joining node, if the timer expires and no reply message is received, it triggers the global search. The joining node sends a request message directly to each edge node. This is possible because during the local search, the received reply messages include the identifier of the edge nodes which initiated them. Additionally each intermediate node keeps an active interface towards each edge node. In the global search the path budget is set to the graph diameter and the waiting timer to a value that prevents waiting indefinitely.

[10]Split horizon is a method to prevent a routing loop in a network. The principle of this method is to never send back the routing information of a packet in the direction from which it was received.

Fig. 5.2b illustrates an example of GCMR.

GCMR is adaptive and the adaptability mechanism, which is based on a modified two-stage search process, is initiated by the upstream node with respect to the point of change.

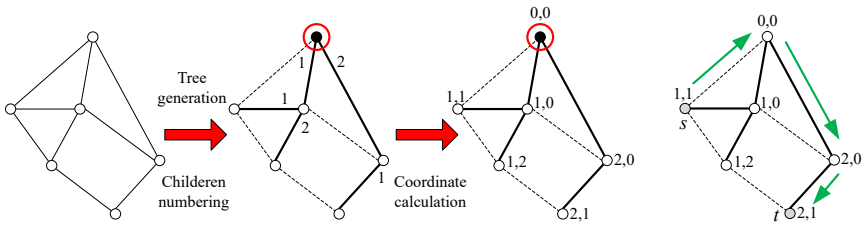
5.3.2.2 Geometric routing

Geometric routing provides an alternative mechanism trading-off dynamics with increased memory efficiency. We have investigated three classes: (i) Geometric Tree-based greedy Routing, (ii) Word-Metric-based Greedy Routing and (iii) Geometric Coordinate-Labeling Scheme. The first two schemes are based on a tree structure and follow similar procedures. All 3 schemes rely on embeddings into metric spaces to assign coordinates to nodes which are used as locators to perform point-to-point routing decisions in this space.

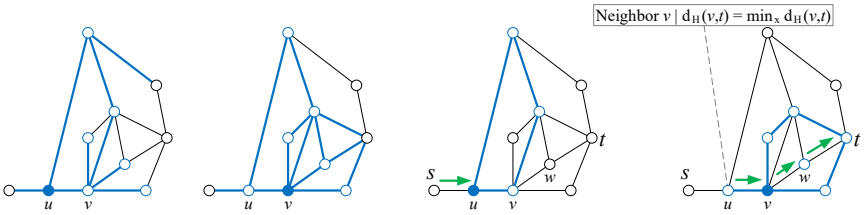
Geometric Tree-based greedy Routing (GTR) and Word-Metric-based Greedy Routing (WMGR). GTR [5] and WMGR [6] comprise two components: (i) greedy embedding and (ii) greedy routing/forwarding. The greedy embedding scheme finds mapping between nodes and coordinates in a metric space in such a way that there is always a distance-decreasing neighbor towards any destination in the network. These coordinates are then used by the forwarding component to forward the packets towards the intended destinations. Knowing the coordinates of the neighbors, in order to forward an incoming packet, the distance between every neighbor and the packet's destination is calculated. The neighbor with the maximal decrease in the distance is selected as the next hop. This scheme is referred to as greedy routing/forwarding because in each step, the node with maximum decrease in the distance is selected.

In GTR and WMGR, coordinates are determined based on a network spanning tree. While GTR calculates coordinates based on the path from the tree root to each node, WMGR relies on a Word-Metric Space (WMS) which is generated by an algebraic group, where the distance function between two elements is the shortest path length of the corresponding vertices in the Cayley Graph of the group. Considering the free group[11] with a generating set S, the embedding in WMGR involves mapping the network spanning tree into the Cayley Graph of the free group. The required steps for calculating the embedding in both schemes are: (i) a rooted spanning tree of the network is generated, (ii) in GTR, the root node sets its coordinates to zero while in WMGR, knowing that S is an alphabet of symbols s_i and a word is a sequence of these symbols, the root is assigned a label that is empty word of the group (identity), (iii) in GTR, each node numbers its children from 1 to d and calculates their Coordinate Sets (CS) by putting the child's assigned number

[11] In mathematics, the free group over a given set S comprises all expressions (words or terms) that can be generated from the members of S.



(a) A spanning tree of the network is generated and children of each node are numbered from 1 to d. The root node coordinate is set to zero. The coordinate set of each node is calculated by its parent. In greedy forwarding, the neighbor with minimum tree-distance towards t is selected as the next hop. The greedy forwarding is not necessarily on the tree.



(b) Learning the k-hop vicinity for $k = 2$ is depicted in nodes u and v. Both routing in a vicinity and between vicinities are illustrated.

Figure 5.3: Variants of geometric routing. (a) depicts an example of GTR embedding and forwarding, and (b) indicates the principles of GCLS.

in place of the first zero coordinate in its own CS. Similarly, in WMGR, every node v assigns to its i-th child, a label that is the concatenation of its label and s_i.

For GTR, we propose to use tree-distance as metric which is the hop-count on the tree between two nodes and is calculated as follows:

1. The closest common ancestor to both nodes is found.
2. The hop-count of each node to the ancestor is counted.
3. The sum of these hop-counts determines the tree-distance between them.

Fig. 5.3a depicts the greedy embedding and greedy forwarding in GTR.

In WMGR, given the labels of two vertices u and v, we distinguish between a common prefix (the set of first symbols that are equal) and the suffixes (the rest of symbols). The distance between u and v is the length (number of symbols) of the word composed by the concatenation of these two suffixes.

Simulation experiments proved that both schemes perform equally well as other greedy embedding-based schemes in terms of stretch but better in terms of coordinate memory scaling.

In these schemes, adaptivity with respect to changing topology is achieved via an on-demand discovery component to bypass failing elements. The latter can be proactively activated, or can be executed upon failure detection. If these techniques are not applied, re-convergence of the supporting spanning tree is needed, resulting in coordinates re-calculation for a (sub-)tree of the topology.

Geometric Coordinate-Labeling Scheme (GCLS). GCLS [7] is the extension of the previously explained geometric routing schemes. It uses k-hop neighborhood information instead of the default 1-hop neighborhood. GCLS relies on hyperbolic geometry in which coordinate calculation is based on a distributed process where all nodes send information to their neighbors. Coordinates are then derived from round-trip times [8] transformed into hyperbolic distances.

In order to dynamically populate the routing tables with entries pointing to the calculated coordinates, this scheme follows a modified distance-vector algorithm. Each node maintains a vector of distance from itself to all nodes within k hops. Note that the calculated distances in this modified version are based on the hyperbolic distance.

The nodes within maximally k-hop distance form a k-hop vicinity. In the RT of each node in a vicinity, there is an exact match for each destination node which belongs to the same vicinity.

The scheme combines exact match lookup (locally reachable vicinities) and greedy forwarding (remotely accessible vicinities). Upon receiving a packet, first it is checked if an exact match is found in the local RT. In case of a miss, the hyperbolic distance between every neighbor and the destination is calculated and the neighbor with minimum distance is selected as the next hop. Fig. 5.3b depicts the principles of GCLS.

5.3.2.3 Route Discovery with network's structural properties (RD)

We designed a route discovery scheme for an inter-AS network where each network is a member of a specific group. The country code (ISO 3166) is used for defining groups in the Internet and assumed that at least one path exists between each pair of nodes. This scheme is based on limited network information, i.e., 2-hop neighborhood information, and membership of nodes to groups, whose efficiency is based on the existence of highly popular nodes and the similarity of adjacent nodes.

The route discovery scheme is initiated by the source node that issues a discovery packet, which is forwarded to a neighbor with the optimal decision rule exploiting the local information. Similarly, the discovery packet is forwarded to the subsequent nodes, until it reaches the destination, hopefully with the smallest

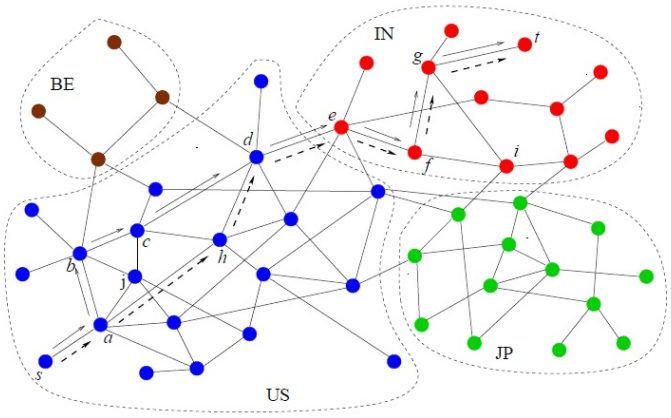


Figure 5.4: Example of route discovery mechanism. This mechanism first finds a path $s-a-b-c-d-e-f-g-t$. The path optimization mechanism attempts to reduce the length of this path at each node. This mechanism produces a shorter path, $s-a-h-d-e-f-g-t$. The 2-hop neighbor information of d contains a. As a consequence, b and c are replaced by h. The loop avoidance mechanism prevents retracing the already visited node e, once f is reached. This enables the selection of g as the next node, which has the same preference as e. Since b has an option to choose the next neighbor among c and j, a random selection is applied to pick c.

number of hops. In this scheme every condition, which is used for finding the next hop, is first checked for the immediate neighbors and if no neighbor satisfies it the 2-hop neighbors are checked. The next hop is selected based on the similarity of the immediate/2-hop neighbor to the destination. The similarity means that either the node has the same AS Number (ASN) as the destination or it shares the country code with the destination. Otherwise, the more connected immediate/2-hop neighbor determines the next hop. The connectivity is expressed by the node degree. Once the discovery packet reaches a node sharing the country code with the destination, the destination's ASN is sought within the particular country. During the discovery process, an online path optimization mechanism is employed to reduce the path length of the searched path by utilizing 2-hop neighborhood information. The discovery mechanism does not consider an already visited neighbor as a next node to avoid loops in the final path. An example of this mechanism is provided in Fig. 5.4.

5.3.3 Routing schemes for DTN and P2P networks

5.3.3.1 Delay Tolerant Network (DTN)

The concept of DTN was introduced initially in the research efforts made for Interplanetary Internet. However, today it is known that similar concepts can be

applied to many other networks called 'challenged networks'. The main characteristics of such networks are frequent disruption, sparse network density, high error rate, delay and mobility. Routing in such networks is quite challenging. [9] surveys many of the recent routing schemes for DTNs. A known scheme in such networks is epidemic routing in which a message is replicated to all neighbors except the one on which the message arrived. Different improvements to this routing and hybrid schemes such as epidemic routing combined with network coding are described in [9]. Similar to DTNs, wireless mobile ad hoc networks are considered infrastructureless and dynamic in nature. Stochastic routing is considered to be a promising paradigm in such networks. In this routing the next hop in a path is selected according to a probability distribution. Many factors such as load, residual energy and forwarding cost can be used to influence this distribution.

5.3.3.2 Peer-to-Peer (P2P) overlay network

P2P networks, initially introduced as a simple music sharing application, are today responsible for a significant share of the Internet traffic. P2P overlays are logical topologies on top of the physical networks which can be built dynamically. These networks are highly scalable, resilient and self configurable which motivate their widespread use. [10] surveys several algorithms and mechanisms considered in P2P overlay networks. One interesting concept considered in P2P is applying a DHT structure on top of the overlay. Using this structure (key, value) pairs are stored in a DHT and all participating nodes can retrieve the value associated with a key efficiently.

5.4 Comparative analysis and identified trade-offs

Within EULER we performed in-depth evaluations of the schemes explained in Section 5.3.2 on large-scale scale-free networks and compared the results/identified trade-offs with BGP since it is the only routing protocol which has been really applied in a large-scale scale-free network.

Table 5.2 compares the upper bounds of the performance metrics characterizing routing algorithms. These complexities correspond to the case of scale-free networks. Then, we detail the trends/trade-offs in different routing components, identified through excessive simulation/emulation experiments, which should enable more careful conclusions regarding the applicability of the schemes to large-scale/complex networks.

DCR. Simulation results indicate that the actual stretch and the RT size of DCR are far better than the theoretical ones [3]. Comparative evaluations of different algorithms indicate that exploiting topological properties helps improving the performance in some approaches [11]. For instance, CLUSTER using power-

Table 5.2: Comparison stretch - memory - communication cost. In this table, m stands for the number of links, n is the number of nodes and $D(G)$ is the diameter of graph G. $\Delta(G)$ represents the maximum nodes degree and h is the size of MDT in multicast routing. δ is the Gromov delta which measures the deviation of the graph from tree-likeness.

	Stretch	**Memory complexity - input**	**Memory complexity - output**	**Communication complexity**
DCR	5	$\tilde{O}(\sqrt{n})$	$\tilde{O}(\sqrt{n})$	$O(n^{3/2})$
GCMR	$O(\frac{D(G)+1}{2})$	$O(\Delta(G) \cdot \log(n))$	$O(h \cdot \log(n))$	$2m$
GTR	$O(\log(n))$	$O(\Delta(G))$	$O(\Delta(G) \cdot \log^2(n))$	$O(n + \log(n) \cdot m)$
WMGR	$O(\log(n))$	$O(\Delta(G))$	$O(\Delta(G) \cdot \log^2(n))$	$O(n + \log(n) \cdot m)$
GCLS	2δ	$O((n(n-1))^{1/2} \cdot \log(n))$	$O(n^{1/2} \cdot \log(n))$	$O(m.n^{1/2})$
BGP	1	$O(D(G) \cdot n \cdot (n-1) \cdot \log(n))$	$O(D(G) \cdot n \cdot \log(n))$	$O(n^2 \cdot (n-1) \cdot polylog(n))$

law graphs properties, is efficient on every criteria if the network has small-world properties[12]. However, the performance of this algorithm degrades drastically in other networks (e.g. Unit Disk Graph[13]) [11]. On the contrary DCR has a trade-off between communication cost/stretch independent of the considered graph. In different topologies, DCR achieves a communication cost almost 10 times smaller than BGP with an average stretch of less than 2 and a maximum number of entries 10 times smaller than BGP.

GCMR. Simulation/emulation results confirm that GCMR, compared to state-of-the-art such as PIM, SPT and ACMR [12], achieves the lowest memory space for storing the routing information, a minimum stretch factor increase (w.r.t. the optimal one), and the lowest recovery/convergence time in case of failure while further improvements in terms of communication cost are required [4]. Regarding identified trade-offs, additional information in the RTs (i) allows large reduction in the communication cost, (ii) decreases the stretch and (iii) allows a low reduction in the convergence time.

GTR-WMGR. Simulation/emulation outcomes support the memory-advantage of both GTR and WMGR, but clearly indicate the resulting cost in the recovery domain [5], [6]. Similar to other greedy routing schemes the RT size is bounded by the maximum vertex degree. On scale-free graphs, these schemes achieve good trade-offs among different metrics: they are scalable in storage space, they are succinct (labels are of size $O(polylog(n))$ bits), and they have a bounded low-stretch. The identified trade-offs in case of failures are:

- potentially high number of affected paths, with a generally low convergence time
- protection: fast recovery with no communication overhead, but high stretch
- restoration: high convergence-time/communication-cost with potentially low stretch

Route Discovery (RD). This scheme could discover near-optimal paths in most cases, even when a significant number of links/nodes are suppressed. Incorporating a moderate global knowledge about the network structure group membership induces a steep improvement in performances. The identified trade-offs are:

- group information in the packet and at each node decreases the search area

[12] In a network with small-world properties, the typical hop-count between two randomly chosen nodes grows proportionally to the logarithm of the number of nodes in the network.

[13] A Unit Disk Graph is the intersection graph of circles of unit radius in the Euclidean plane. In this graph, each vertex corresponds to a circle and two vertices are connected by an edge if and only if the corresponding circles intersect.

- online path optimization mechanism significantly reduces the discovered path length
- topological information needed at each node depends on the node degree of its neighbors

Schemes such as GCLS, GCMR and RD show adaptability to failures by only requiring re-computation of the routes affected by the failure. The number of affected routes is proportional to the centrality of the failing entity. GTR and WMGR provide protection capability to overcome pre-determined failure patterns and if no protection exists, they re-calculate the coordinates of the affected nodes. DCR on the other hand does not provide dedicated processing for information state changes and require the full re-computation of the routing tables.

Exploiting the topological properties of scale-free networks can improve the performance of several compact routing schemes [11]. This was also confirmed in GTR and WMGR schemes as the tree construction method (i.e., selection of maximum degree node as root and construction of Breadth-First-Search tree) does not generate deep branches due to the short average distance between nodes in scale-free networks. This minimizes the memory requirements for coordinate representation and enables shorter paths [13].

In all schemes, a scalable mapping system to bind node identifiers to node locators is required. An option is to use DNS-like servers for these name-to-address/address-to-address resolutions. The main identified trade-off is between communication-cost and convergence-time. The slower the polling scheme relative to the mapping service, the smaller the communication-cost but the longer the convergence time.

The proposed schemes have different packet forwarding process. While GTR, WMGR and RD replace the look up with more computation in the forwarding plane, GCLS and DCR look up the next hop of a packet from the RTs.

5.5 Conclusion and future directions

We presented an overview of potential design dimensions of routing protocols, the routing functionality and existing routing schemes. The focus of the article was mainly on advances in compact routing, algebraic routing and geometric routing. Through analyzing experimental/analytical results performed so far, we identified the main open challenges:

- One cause of absence of an alternative to BGP is that the design of many routing systems (mainly path-vector schemes enhancements) tends to follow the same approach as the one pursued by BGP. To overcome this, clean-slate approaches should be investigated.

- Most investigated schemes increase performance by decreasing functionality e.g., all the schemes in Section 5.3.2 improve the scalability in terms of memory however the same level of policy as in BGP is not supported in any of them.

- The main difference in the schemes discovery process results from the exchange of routing information: pull vs. push. All alternatives use a distance metric/spatial routing metric which subdivide between local and global metrics and between metrics derived from the topology properties (e.g. node degree) and universal metrics. These dimensions are tightly related and our results corroborate that schemes such as BGP, which is independent of global or link metrics and is driven by local policy decisions, will be challenging to replace as long as the Internet domains are operated organically.

- From the experimental perspective, due to the increasing level of path processing granularity combined with a larger parameter space, deriving common path characteristics to obtain a representative policy model together with the AS relationships remains challenging.

The experiments performed so far indicated that the proposed schemes have interesting characteristics in terms of memory usage, stretch and convergence behavior which make them promising schemes for large-scale complex networks. Indeed, there are some open problems which require further research:

- Many of the schemes rely on a tree construction. It is thus appropriate to further investigate multi-path routing via independent trees in order to extend these schemes with fault-tolerance and load balancing [14]. Additionally, using multiple trees is a starting point for enabling routing policy in these schemes.

- Many of the schemes require a mapping system. Despite the research efforts (mainly in LISP [15]) a scalable, secure and highly reliable mapping system with fast convergence is still missing.

- The proposed schemes find applicability in upper layers (IT/computing systems, information/file systems) when the number of entities reaches at least 10^{10}. Concretely, Content-Centric Networking (CCN) is one paradigm which can benefit from the proposed geometric routing schemes. Using these schemes, an efficient and scalable content-based forwarding is possible which was demonstrated in [7]. If capacity saving remains a key objective, integrating multicast benefits with CCN should be further investigated.

- Although, the routing schemes in Section 5.3.3 are proposed for networks far from complex networks, it is an interesting research direction to investi-

gate the applicability of such routing schemes in large-scale scale-free networks. Particularly, schemes such as stochastic routing may be a promising alternative if parameters such as network load is used in calculation of probability distribution. This way an adaptive load balancing mechanism can be achieved. P2P networks, as potential data distribution paradigm of future Internet, require further investigation to improve aspects such as security, dynamicity, redundancy and load balancing [10].

- Finally, it is challenging to translate schemes/algorithms into protocols and it is a research topic on its own.

Acknowledgment

This work is partly funded by the European Commission through the EULER project (Grant 258307), part of the Future Internet Research and Experimentation (FIRE) objective of the Seventh Framework Programme (FP7) and the UGent BOF/GOA project 'Autonomic Networked Multimedia Systems'.

References

[1] F. Coras, D. Saucez, L. Iannone, and A. Cabellos-Aparicio. *Locator/ID Separation Protocol (LISP) Impact*. 2016.

[2] P. Godfrey, M. Caesar, I. Haken, Y. Singer, S. Shenker, and I. Stoica. *Stabilizing route selection in BGP*. Networking, IEEE/ACM Transactions on, 23(1):282–299, 2015.

[3] C. Gavoille, C. Glacet, N. Hanusse, and D. Ilcinkas. *On the communication complexity of distributed name-independent routing schemes*. In International Symposium on Distributed Computing, pages 418–432. Springer, 2013.

[4] D. Careglio, D. Papadimitriou, F. Agraz Bujan, S. Sahhaf, J. Perelló Muntan, W. Tavernier, S. Spadaro, and D. Colle. *Development and experimentation towards a multicast-enabled Internet*. In 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pages 79–84. Institute of Electrical and Electronics Engineers (IEEE), 2014.

[5] S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester. *Experimental validation of resilient tree-based greedy geometric routing*. Computer Networks, 82:156–171, 2015.

[6] M. Camelo, D. Papadimitriou, L. Fàbrega, and P. Vilà. *Geometric Routing With Word-Metric Spaces*. IEEE Communications Letters, 18(12):2125–2128, 2014.

[7] S. Sahhaf, D. Papadimitriou, W. Tavernier, D. Colle, and M. Pickavet. *Experimentation of geometric information routing on content locators*. In Network Protocols (ICNP), 2014 IEEE 22nd International Conference on, pages 518–524. IEEE, 2014.

[8] T. E. Ng and H. Zhang. *Predicting Internet network distance with coordinates-based approaches*. In INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, volume 1, pages 170–179. IEEE, 2002.

[9] Y. Cao and Z. Sun. *Routing in delay/disruption tolerant networks: A taxonomy, survey and challenges*. Communications Surveys & Tutorials, IEEE, 15(2):654–677, 2013.

[10] A. Malatras. *State-of-the-art survey on P2P overlay networks in pervasive computing environments*. Journal of Network and Computer Applications, 55:1–23, 2015.

[11] C. Gavoille, C. Glacet, N. Hanusse, and D. Ilcinkas. *Brief Announcement: Routing the Internet with Very Few Entries*. In Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, pages 33–35. ACM, 2015.

[12] I. Abraham, D. Malkhi, and D. Ratajczak. *Compact multicast routing*. In Distributed Computing, pages 364–378. Springer, 2009.

[13] S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester. *Efficient Geometric Routing in Large-Scale Complex Networks with Low-Cost Node Design*. IEICE Transactions on Communications, 99(3):666–674, 2016.

[14] R. Houthooft, S. Sahhaf, W. Tavernier, F. De Turck, D. Colle, and M. Pickavet. *Robust geometric forest routing with tunable load balancing*. In Computer Communications (INFOCOM), 2015 IEEE Conference on, pages 1382–1390. IEEE, 2015.

[15] A. Anisul and H. Flinck. *A Compact Routing based Mapping System for the Locator/ID Separation Protocol (LISP)*. International Journal of Computer Applications, 127(5):1–8, 2015.

6

Network service chaining with optimized network function embedding supporting service decompositions

The rise of NFV and SDN introduce opportunities for service providers to quickly deploy novel services with a reduced cost. In this context, Network Service Chaining (NSC) indicates how basic service building blocks (i.e. Network Functions (NFs) such as firewall, network address translation and deep packet inspection) are chained across the network infrastructure. Since NFV provides freedom in where to place the NFs of NSCs in a network, new challenges are introduced in service orchestration. This chapter and the next one focus on the service orchestration research topic and its related challenges. In the context of NSC, a large monolithic service block can be decomposed into inter-connected atomic NFs in several ways. This is referred to as service decomposition. Having multiple realization options, the placement of NFs (embedding) becomes quite challenging. This chapter proposes solutions for joint optimization of NFs embedding and service decompositions to be used in virtualized telecom networks. An ILP-based algorithm and a heuristic approach are proposed to minimize the mapping cost while taking service requirements and network capabilities into account. The advantages of the proposed schemes are identified through high service acceptance rate and the related low mapping cost.

⋆ ⋆ ⋆

S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, P. Demeester

Published in Computer Networks, 2015.

Abstract The rise of Software-Defined Networking (SDN) and Network Function Virtualization (NFV) introduce opportunities for service providers to reduce CAPEX/OPEX and to offer and quickly deploy novel network services. In particular, SDN and NFV enable the flexible composition of network functions, a generic service concept known as network service chaining (NSC).

However, the control of resources, management and configuration of network service chains are challenging. In particular, there typically exist multiple options on how an abstract network service can be decomposed into more refined, interconnected network functions. Moreover, efficient algorithms have to be devised to allocate the network functions. The underlying algorithmic problem can be seen as a novel generalization of the Virtual Network Embedding Problem (VNEP), where there exist multiple realization options. The joint optimization of decomposition and embedding has not been studied in the literature before.

This paper studies the problem of how to optimally decompose and embed network services. In particular, we propose two novel algorithms to map NSCs to the network infrastructure while allowing possible decompositions of network functions. The first algorithm is based on Integer Linear Programming (ILP) which minimizes the cost of the mapping based on the NSCs requirements and infrastructure capabilities. The second one is a heuristic algorithm to solve the scalability issue of the ILP formulation. It targets to minimize the mapping cost by making a reasonable selection of the network function decompositions. The experimental results indicate that considering network function decompositions at the time of the embedding significantly improves the embedding performance in terms of acceptance ratio while decreasing the mapping cost in the long run in both optimal and heuristic solutions.

6.1 Introduction

Network service chaining (NSC) is a service concept which has gained much interest from both practitioners and researchers. NSC promises increased flexibility and cost-efficiency for future carrier networks. NSC is enabled by Software-Defined Networking (SDN) and Network Function Virtualization (NFV). Employing SDN and NFV developments simplifies the service chain provisioning significantly and enables the introduction of new services. Traditionally, a service composed of several functions is implemented by middleboxes and traffic should flow through these middleboxes in a given order. A service chain is an abstraction to define

high-level services in a more generic way. The service is composed of a chain of high-level Network Functions (NFs) with pre-defined parameters referred to as Service Graph (SG). Different aspects of service chaining, its limitations and existing challenges are investigated by different activities and research projects such as: (i) a dedicated working group (Service Function Chaining Working Group) in IETF which focuses on the service chaining architecture, (ii) the Network Functions Virtualization (NFV) group within ETSI which investigates software-based telecommunications services to be run in virtualized environment instead of special purpose appliances, (iii) UNIFY, a EU-funded FP7 project, which focuses on developing an automated, dynamic service creation architecture based on a dynamic fine-granular service chaining model leveraging Cloud virtualization techniques and SDN.

An NF can be decomposed in multiple ways to several less abstract, more refined NFs, and thus an SG composed of several high-level NFs can be realized through multiple options referred to as service decompositions. To be more precise, NF decomposition refers to (i) translation of a high-level/abstract NF (e.g., Firewall) to more refined NFs (e.g. an OpenFlow- or an iptables-based Firewall) or (ii) decomposing a compound NF into multiple NFs which can potentially be abstract and are interconnected in a graph (e.g. a load-balanced Firewall can be decomposed into any number of Firewalls preceded by a number of load balancers).

To give an example, consider that a user requests a service including a parental control NF. The functionality of this NF can be decomposed to (i) Traffic Classifier, (ii) Web Proxy and (iii) Firewall NFs. Each of these NFs can be realized through more refined NFs, e.g. a Firewall can be implemented as (i) iptables-based Firewall or (ii) OpenFlow-based Firewall. These NFs should be traversed in a given order and the logical connectivity between them is as follows: Traffic Classifier $\rightarrow$ Web Proxy $\rightarrow$ Firewall. This connectivity can be represented by a graph which is referred to as a Network Function Graph (NFG). Service decomposition is defined as a mapping of each NF into a set of NFGs: $NF_i \rightarrow \{NFG_1^i, NFG_2^i, ...\}$.

Having multiple decomposition options for service realization, a challenging task is to find an optimal placement of the NFs within the service to the components of an infrastructure. We refer to this problem as the Network Function Embedding Problem (NFEP) and Fig. 6.1 illustrates its general idea. As we see in this figure, given multiple SGs composed of NFs and a common physical network, we look for a placement of NFs/logical links to the nodes/links of the physical network. In this figure, only one decomposition is depicted for each SG. However, as explained earlier, an SG can be realized through multiple decompositions. This problem can be seen as a generalization of the Virtual Network Embedding Problem (VNEP) in which virtual networks are mapped to a common infrastructure without having multiple realization options.

The literature is rich on algorithmic proposals to solve the VNEP [1]. However,

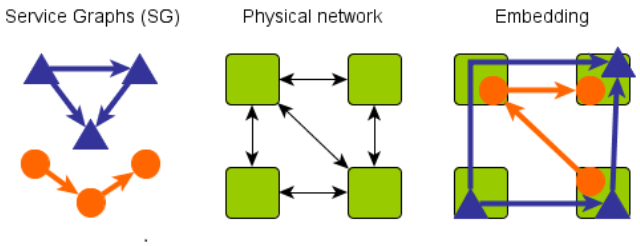


Figure 6.1: Network Function Embedding concept

no work considered to decompose and embed an SG at the same time.

Our contribution. The decompositions of an SG need to reflect required hardware resources and capabilities (e.g. requirement for iptables-based Firewall). Selecting a decomposition independently of available resources in the infrastructure may yield mapping solutions far from optimal. We accordingly present algorithms (optimal and heuristic) for the NFEP which take the SG decomposition opportunities into account. This would certainly improve the performance of the embedding as a reasonable decomposition is selected which corresponds to the existing resources and thus leads to a better placement of the NFs.

To the best of our knowledge, the joint optimization of SG decomposing and its embedding has not been investigated in prior work.

We first propose an Integer Linear Programming (ILP) model to solve the NFEP. This model considers SG decomposition options as the input of the embedding problem. The objective is to minimize the cost of the mapping based on the SG requirements and infrastructure capabilities. We define the cost of mapping an SG as the cost of the total substrate resources allocated to that SG which is calculated based on: (i) the cost of each unit of CPU, memory and storage in a physical node (ii) the cost of each unit of bandwidth in a physical link and (iii) the resource usage of the given SG. The cost per unit of capacity (i,ii) is determined by the infrastructure provider (InP). The algorithm maps the NFs within an SG to the components of the physical network in such a way that the resource consumption is minimized and the QoS requirements of the SG are satisfied. This is equivalent to maximizing the number of service requests which are accepted and thus the acceptance ratio in the long run is increased. One of the main constraints in this mapping is that NFs in an SG can be of different types (e.g. a VM image, a process in a container or a hardware appliance). However, not all types, e.g., iptables-based and OpenFlow-based Firewalls, are supported by all infrastructure nodes.

Solving the VNEP is NP-hard in most of the cases [2] and allowing for all possible SG decompositions generally increases the complexity drastically. As a

result, finding the optimal solution might not be feasible in large-scale scenarios. We therefore propose a heuristic algorithm to overcome the scalability limitation of the ILP solution. In this scheme, first a reasonable decomposition is selected for the SG and then NFs of the selected decomposition are placed on physical network components based on a backtracking mechanism. This algorithm was briefly presented in [3] as a short paper, without any thorough evaluation. We use the proposed ILP-based approach to benchmark the heuristic algorithm. Therefore, this paper extends the work in [3] by providing an ILP model and a thorough evaluation of the proposed scheme. These evaluations should enable more accurate conclusions regarding the performance of the heuristic-based approach compared to the optimal solution.

The proposed approaches are evaluated thoroughly and compared in an extensive computational evaluation. The experimental results indicate that employing SG decomposition options at the time of the embedding improves the performance of the embedding significantly in terms of acceptance ratio while decreasing the cost compared to scenarios in which a decomposition is selected independent of the available resources and NFs requirements.

The rest of the paper is organized as follows. Section 6.2 explains the related work. Service decomposition is described in detail in Section 6.3. Section 6.4 describes the problem, and Section 6.5 details the proposed ILP model together with the ILP-based algorithm. The heuristic-based algorithm is explained in Section 6.6. Section 6.7 reports the performance evaluation results. Finally Section 6.8 concludes the paper.

6.2 Related work

The VNEP is known to be NP-hard [2] and therefore finding the optimal solution might not be affordable in large-scale scenarios (within reasonable time). As a result two different types of approaches are considered: (i) Exact solutions which provide optimal solutions but are generally only applicable on small-scale problems, (ii) Heuristic-based approaches which trade off optimality with runtime. Many of the algorithmic approaches to solve the VNEP have been detailed in the survey [1]. There are several proposals in the literature which formulate the VNEP as Integer Linear Programming (ILP) and find the optimal solution. In [4], the authors used ILP formulation to find a solution which minimizes the embedding cost and maximizes the acceptance ratio. Another ILP-based approach was introduced in [5] which focuses on minimizing the consumed resources in physical network to switch off the remaining resources of the network and save energy. Zhang et al. proposed an ILP model to achieve optimal resilient solution while satisfying the requested QoS requirements [6]. Dynamic reconfiguration of mappings and migrations were studied in [7] and Mixed Integer Programming was used to solve

the problem.

The literature on heuristic algorithms is much more diverse and ideas from very different fields have been considered. In [8] two heuristic-based approaches were presented. One focuses on minimization of resource consumption and the second one aims at load balancing. The same work proposed an ILP approach to benchmark the heuristic-based algorithms. In [9] VNEP is solved by a heuristic approach based on Subgraph Isomorphism Detection (SID) which maps nodes and links during same stage. Several heuristics focus on resiliency in the embedding which try to recover physical network failures [10–12]. Meta-heuristics such as ant colony optimization, simulated annealing, genetic algorithms or tabu search are used to find a close to optimal solutions. An example is the Max-Min Ant Colony meta-heuristic proposed in [13] to solve the VNEP.

While the literature on the VNEP is very rich, the specific problem we attend to – namely of how to decompose *and* map an SG at the same time – was not considered before. Hence, we cannot use any previous work as a baseline, but use our ILP to obtain one.

Apart from the VNEP, the most closely related works are the following. In the work of Basta et al. [14] the function placement problem for LTE mobile core gateways is considered under different decompositions based on NFV and SDN. The function placement problem, which maps each of the (potentially decomposed) functions onto datacenters and establishes paths between the components is then solved according to the different decompositions. As shown in [14], depending on the chosen decomposition, the bandwidth usage and path latencies may vary to a great extent. This not only shows the benefits of combining NFV and SDN, but particularly, that it may be beneficial to decompose common *complex functions*.

Mehraghdam et al. [15] consider the problem of embedding service chains under the relaxation that the order of NFs may be underspecified and NF A may be used before B or vice versa. Assuming that no restriction on the order of the NFs is given, this leads to $n!$ many possible orderings for n NFs. The authors shortly discuss the computational complexity of finding the *optimal* ordering and argue for using a heuristic for choosing a *good* ordering.

In [16], authors focus on NF placement for NFV chaining in packet/optical datacenters with the objective of minimizing the expensive optical/electrical/optical (O/E/O) conversions. These conversions are needed because chaining within a 'performance optimized datacenter' (pod) is based on packet switching while between pods optical technologies are used. They try to minimize the O/E/O conversions by placing the NFs of the same chain in the same pod. They propose both a Binary Integer Programming formulation of the NF placement problem and an alternative heuristic algorithm. However, no notion of NF decomposition is considered in this paper.

Ahmed and Papadimitriou [17] proposed solutions for the discovery and se-

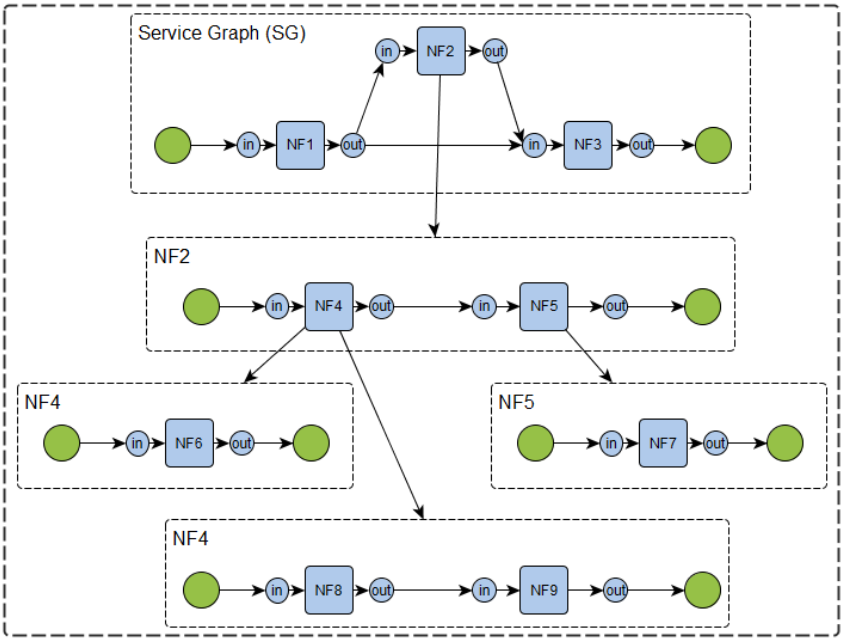


Figure 6.2: Example of service decomposition

lection of middleboxes along the traffic path. NFs composing a service chain are assigned to middleboxes, while preserving the order of the NFs as specified in the service chain. In their work, they address the challenge of flow processing establishment across multiple NF providers.

6.3 Service decomposition

The SG initially requested by a user is described by NFs and their logical connectivity. At this level the NFs might be either Elementary Network Functions (ENF) which means that their low-level implementation and resource requirements are available or abstract/Compound Network Functions (CNF) which means that they can be implemented through more refined NFs or they are composed of several ENFs.

Service decomposition is the process of transforming an SG containing abstract NFs to SGs containing less abstract, more implementation-close NFs. Additionally, it refers to decomposing the functionality of a CNF into multiple less complex (potentially abstract) NFs interconnected in form of a graph with the same external interfaces as the abstract NF (see Fig. 6.2). The main advantages of ser-

vice decomposition are: (i) re-usability of elementary blocks, (ii) the possibility to generate new and more complex services and iii) the possibility to request services without any concern about the detailed implementation. These advantages simplify the network service chaining and provide opportunities for service providers to reduce the cost in CAPEX and OPEX.

An example service decomposition is illustrated in Fig. 6.2. The high-level compound NF2 is decomposed to NF4 and NF5. Each of them can be decomposed to more elementary NFs (e.g., NF6, NF7, etc.). These decompositions can be stored in a tree-like data structure in a database which is used by the embedding algorithms. Note that the leaves of this tree are elementary NFs for which all the resource demands and required low-level implementation are available.

6.4 Problem description

Service requests arrive over time and the embedding algorithm should decide whether the NFs within the requested SG and their corresponding connections can be mapped to the components of the physical network or not. Once requests are accepted, the required resources (physical links and nodes) are assigned and they are released once the requests expire.

Elementary NFs within an SG can be of different types which means that they can be implemented in different ways using different techniques such as: (i) Virtual Machine (VM) images using different virtualization techniques: VMware, VirtualBox, Xen, (ii) hardware appliances, (iii) process in a container or iv) packet I/O drivers, as e.g. Intel's Data Plane Development Kit (DPDK)[1] which is a set of libraries and drivers for fast packet processing.

Having several types of NFs imposes additional constraints on the embedding problem because not all physical components of the network support all types. Additionally, interconnecting NFs of different types is more complex compared to interconnection of the NFs with the same type. For example it is more complicated to connect an NF which is a process in a container to a DPDK-based NF compared to the case in which both NFs are DPDK-based or both are the same-type processes in containers. To reduce this complexity, we prioritize an SG decomposition in which the number of same-type NFs which are directly interconnected is higher. In addition, such a prioritization (i) reduces the amount of required resources, and (ii) improves NF performance by reducing the communication overhead and latency. Prioritizing NFs of the same type enables mapping of more NFs to the same physical node; this leads to less network resource consumption as no physical links are used for the mapping. Interconnecting NFs over physical links implies additional communication and/or computational overhead due to e.g. addi-

[1] http://dpdk.org/

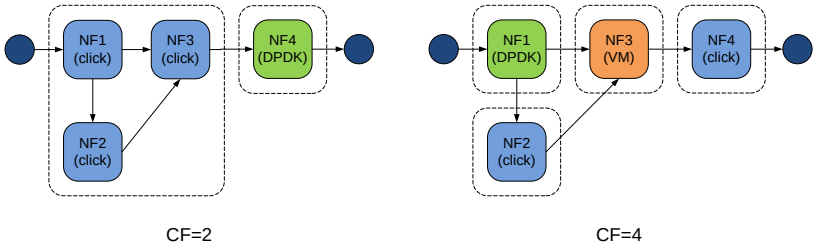


Figure 6.3: Example of Clustering and CF in service decompositions

tional tunneling requirements, which is not as high if NFs are located in the same physical node.

In order to enable this prioritization, we define a parameter referred to as Cluster-Factor (CF) which is calculated as follows for each decomposition: given a service decomposition, the NFs with similar types which are connected directly (i.e., without intermediate nodes with other types) are grouped in the same cluster. The number of clusters in the decomposition determines the CF of that decomposition. Fig. 6.3 illustrates two service decompositions with different CFs. As we see, in the decomposition at left, the Click-based NFs (NF1, NF2 and NF3) are connected directly and thus are grouped in one cluster while NF4 which is a DPDK-based NF is placed in another cluster. Therefore, the CF of this decomposition is equal to 2. In the decomposition at right, although there are NFs with the same type (2 Click-based NFs: NF2 and NF4), they cannot be grouped in the same cluster because of the intermediate NFs (NF1 and NF3) which have different types. As a result each of the NFs is grouped in a different cluster and the CF of this decomposition is 4.

Our objective is to minimize the mapping cost of a given SG. This cost refers to the cost of the total substrate resources allocated to the SG which is calculated based on: (i) the cost of each unit of CPU, memory and storage in a physical node (ii) the cost of each unit of bandwidth in a physical link and (iii) the resource usage of the given SG. The cost per unit of capacity (i,ii) is determined by the InP. Minimizing the mapping cost allows accepting more requests over time and increases the acceptance ratio. Acceptance ratio is a metric that measures the ratio of the accepted service requests which refer to services that are successfully mapped to the physical network. As service decompositions are known from the design time, we can make a resource-aware decomposition selection taking decompositions CF into account at the time of the embedding.

6.5 Integer Linear Programming formulation

In this section, we introduce notations, variables, objective function and constraints which are used in the Integer Linear Programming formulation of the problem.

6.5.1 Physical network

In the model, the physical infrastructure is represented as an undirected graph. The infrastructure consists of nodes (N) connected via links (L). Each node has a certain capacity in terms of computation, memory and disk/storage, and links have delay and capacity in terms of bandwidth. These resources are actually the residual capacities based on previous mappings. Below we describe the parameters of the physical network infrastructure.

$$G_p = (N_p, L_p)$$

Computation capacity (C), storage capacity (S) and memory capacity (M) with their corresponding unit cost are defined as follows:

$$\forall u \in N_p : \; C_u, S_u, M_u \in N^+$$

$$\forall u \in N_p : \; C_{cost_u}, S_{cost_u}, M_{cost_u} \in N^+$$

Each physical node can support different implementation types of NFs including: (i) Virtual Machine (VM) images (ii) process in a container (iii) packet I/O drivers and (iv) hardware appliances. Note that the model can simply be extended if other types of NF implementation are supported by the infrastructure nodes:

$$\forall u \in N_p : \; u \in \text{VM}, \; \text{iff.} \; u \; \text{may host a VM}$$

$$\forall u \in N_p : \; u \in \text{process}, \; \text{iff.} \; u \; \text{may host a process}$$

$$\forall u \in N_p : \; u \in \text{I/O}, \; \text{iff.} \; u \; \text{may host packet I/O drivers}$$

$$\forall u \in N_p : \; u \in \text{hardware}, \; \text{iff.} \; u \; \text{has hardware appliances}$$

Propagation delay (D) and bandwidth capacity (BW) for physical links are described as follows:

$$\forall e_{uv} \in L_p : \; D_{e_{uv}}, BW_{e_{uv}}, BW_{cost_{e_{uv}}} \in N^+$$

6.5.2 Service request

As explained in Section 6.3, an SG can be realized through multiple decompositions. Therefore, for each SG there exists a decomposition set $Decomp_{SG}$. Note that this set contains *all* possible decompositions such that hierarchical decompositions are already 'fully' resolved.

$$\forall SG: \ Decomp_{SG} = \{dc_1, dc_2, ..., dc_n\}$$

Each decomposition is represented as a directed graph to support the dependency between different NFs. Therefore, the NFs in the decomposition are represented as nodes connected via the directed links in the graph. Each NF, has some requirements in terms of computation, memory and storage and links connecting different NFs have requirements in terms of delay and bandwidth.

$$G_{dc} = (N_{dc}, L_{dc})$$

The computation (c), memory (m) and storage (s) requirements of each NF (node) in the decomposition is defined as:

$$\forall i \in N_{dc}: \ c_i, s_i, m_i \in N^+$$

Each NF can be implemented differently and thus can be of different type:

$$\forall i \in N_{dc}: \ i \in \{\mathrm{VM}, \mathrm{process}, \mathrm{I/O}, \mathrm{hardware}\}$$

The maximum allowed delay (d) and bandwidth (bw) requirements of links in the decomposition are defined as:

$$\forall e_{ij} \in L_{dc}: \ d_{e_{ij}}, bw_{e_{ij}} \in N^+$$

Finally each decomposition is assigned a Cluster Factor (CF) which is the number of clusters in the decomposition. The clusters in a decomposition include same-type NFs which are directly interconnected and can probably be mapped on the same physical node.

$$\forall dc \in Decomp_{SG}: \ CF_{dc} \in N^+$$

6.5.3 Decision variables

In our ILP model, different decision variables are required. The x_u^i is used to indicate if NF i independent of its type is mapped on physical node u.

$$x_u^i \in \{0, 1\} \ \forall dc \in Decomp_{SG}, \forall u \in N_p, \forall i \in N_{dc}$$

The next variable is 1 if virtual link e_{ij} is mapped to physical link e_{uv}, and 0 if not.

$$f_{e_{uv}}^{e_{ij}} \in \{0,1\}\, \forall dc \in Decomp_{SG}, \forall e_{ij} \in L_{dc}, \forall e_{uv} \in L_p$$

The z^{dc} variable is used to indicate if decomposition dc is selected for the mapping or not.

$$z^{dc} \in \{0,1\}\ \forall dc \in Decomp_{SG}$$

6.5.4 Objective function

As explained in Section 6.4, the objective is to minimize the total cost of the mapping (i.e., the cost of the total substrate resources allocated to the given SG) while prioritizing the decompositions with lower CF and the advantages of this prioritization are: (i) lower complexity in interconnection of NFs, (ii) improving NF performance by reducing communication overhead and latency, and (iii) reducing the compute and network resources.

Minimize:

$$\sum_{dc \in Decomp_{SG}} \sum_{u \in N_p} \sum_{i \in N_{dc}} cost(i,u) + \sum_{dc \in Decomp_{SG}} \sum_{e_{uv} \in L_p} \sum_{e_{ij} \in L_{dc}} cost(e_{ij}, e_{uv}) \tag{6.1}$$

Below, we detail this objective function. In the rest of this paper, the notations 'VM', 'PRC', 'I/O' and 'HW' refer to different types of NFs (VM, process, I/O, hardware) explained earlier.

$$\begin{aligned}
&\sum_{dc \in Decomp_{SG}} \sum_{u \in N_{p(VM)}} \sum_{i \in N_{dc(VM)}} (c_i \times C_{cost_u} + s_i \times S_{cost_u} + m_i \times M_{cost_u}) \times \\
&\qquad CF_{dc} \times x_u^i + \\
&\sum_{dc \in Decomp_{SG}} \sum_{u \in N_{p(PRC)}} \sum_{i \in N_{dc(PRC)}} (c_i \times C_{cost_u} + s_i \times S_{cost_u} + m_i \times M_{cost_u}) \times \\
&\qquad CF_{dc} \times x_u^i + \\
&\sum_{dc \in Decomp_{SG}} \sum_{u \in N_{p(I/O)}} \sum_{i \in N_{dc(I/O)}} (c_i \times C_{cost_u} + s_i \times S_{cost_u} + m_i \times M_{cost_u}) \times \\
&\qquad CF_{dc} \times x_u^i + \\
&\sum_{dc \in Decomp_{SG}} \sum_{u \in N_{p(HW)}} \sum_{i \in N_{d(HW)}} (c_i \times C_{cost_u} + s_i \times S_{cost_u} + m_i \times M_{cost_u}) \times \\
&\qquad CF_{dc} \times x_u^i + \\
&\qquad \sum_{dc \in Decomp_{SG}} \sum_{e_{uv} \in L_p} \sum_{e_{ij} \in L_{dc}} (bw_{e_{ij}} \times BW_{cost_{e_{uv}}} \times f_{e_{uv}}^{e_{ij}})
\end{aligned} \tag{6.2}$$

Note that in this objective function, we multiplied the cost of the mapping by the CF. Through this multiplication, it is ensured that the decompositions with lower CF are preferred over the decompositions with higher CF.

As we see in the objective function, the sum over all the decompositions of the given SG is minimized. The reason is that all of these decompositions should be checked and the one which leads to minimum cost should be selected and mapped. In order to guarantee that only one decomposition is selected, we add several constraints which are detailed in Section 6.5.5.1.

6.5.5 Constraints

There are different types of constraints which should be considered in the ILP formulation. We categorize them in 4 groups: (i) decomposition constraints, (ii) physical nodes, (iii) link to path mapping and (iv) QoS requirements.

6.5.5.1 Decomposition mapping constraints

These constraints guarantee that only one of the decompositions of an SG is selected and all the NFs of the selected decomposition are mapped only once.

$$\sum_{dc \in Decomp_{SG}} z^{dc} = 1 \tag{6.3}$$

$$\sum_{u \in N_{p(type)}} x_u^i = z^{dc} \tag{6.4}$$

$$\forall dc \in Decomp_{SG}, \forall i \in N_{dc(type)}, \forall type \in \{\mathrm{VM}, \mathrm{PRC}, \mathrm{I/O}, \mathrm{HW}\}$$

6.5.5.2 Physical node constraints

For each physical node, the sum of the requirements of all the mapped NFs should not exceed the capacity of that node. Therefore, we should add a constraint for each of the compute, memory and storage capacities while considering different types (constraints 6.5-6.7).

$$\sum_{i \in N_{dc(type)}} c_i \, x_u^i \leq C_u \tag{6.5}$$

$$\forall dc \in Decomp_{SG}, \ \forall u \in N_{p(type)}, \forall type \in \{\mathrm{VM}, \mathrm{PRC}, \mathrm{I/O}, \mathrm{HW}\}$$

$$\sum_{i \in N_{dc(type)}} m_i \, x_u^i \leq M_u \tag{6.6}$$

$$\forall dc \in Decomp_{SG}, \ \forall u \in N_{p(type)}, \forall type \in \{\mathrm{VM}, \mathrm{PRC}, \mathrm{I/O}, \mathrm{HW}\}$$

$$\sum_{i \in N_{dc(type)}} s_i \, x_u^i \leq S_u \tag{6.7}$$
$$\forall dc \in Decomp_{SG},\ \forall u \in N_{p(type)}, \forall type \in \{\mathrm{VM}, \mathrm{PRC}, \mathrm{I/O}, \mathrm{HW}\}$$

6.5.5.3 Link to path mapping

When a link in the SG cannot be mapped to a single physical link, it should be mapped to a single path. In case of the latter, the link (or corresponding flow) in the SG cannot be split into several paths and thus no multipath is considered in the model. The next constraint makes sure that a simple unsplittable/single path for such a mapping is used.

$$\sum_{e_{uv} \in L_p, u=src} f_{e_{uv}}^{e_{ij}} - \sum_{e_{uv} \in L_p, u=dst} f_{e_{uv}}^{e_{ij}} = x_u^j - x_u^i \tag{6.8}$$
$$\forall dc \in Decomp_{SG}, \forall e_{ij} \in L_{dc}, \forall u \in N_p$$

6.5.5.4 Quality of service requirements

For each physical link, the sum of the required bandwidth of all the links in each decomposition should not exceed the bandwidth capacity of the link.

$$\sum_{e_{ij} \in L_{dc}} bw_{e_{ij}} \cdot f_{e_{uv}}^{e_{ij}} \leq BW_{e_{uv}} \ \forall dc \in Decomp_{SG}, \forall e_{uv} \in L_p \tag{6.9}$$

The next constraint guarantees that the sum of all physical link delays used for mapping a single virtual link in the service request does not exceed the maximum allowed delay in the request.

$$\sum_{e_{uv} \in L_p} D_{e_{uv}} \cdot f_{e_{uv}}^{e_{ij}} \leq d_{e_{ij}} \ \forall dc \in Decomp_{SG}, \forall e_{ij} \in L_{dc} \tag{6.10}$$

As we used an Integer Linear Programming formulation to solve the embedding problem, the model is limited to linear parameters. It can be extended with other parameters as long as they are linear. Note that parameters such as delay jitter and loss cannot be modeled through linear constraints and thus, they are not considered in this model. It is worth mentioning that modeling such optimization problems as nonlinear programs to account for jitter and loss is hard but leads to high accuracy. However, such an accurate model requires very large computation which makes it impossible to solve the optimization problem in a reasonable time.

6.5.6 ILP-based algorithm

This algorithm implements the optimal ILP-based model which was detailed extensively in this section. The requests arrive over time and given all possible decompositions of an SG, the algorithm selects the decomposition which leads to minimum cost and embeds the corresponding NFs and logical links to physical network in such a way that the resource consumption is minimized. As a result, more requests can be embedded into the physical network and therefore, the acceptance ratio will increase in the long run [1]. The achieved embedding is based on the service requests constraints and the physical network limitations. The impact of considering service decompositions on different metrics such as acceptance ratio and mapping cost is evaluated in Section 6.7.

6.6 Decomposition selection-backtracking mapping algorithm: DSBM

As the ILP-based algorithm has scalability limitations, we propose a heuristic-based approach which is referred to as DSBM [3]. Similar to any heuristic approach, the proposed scheme compromises optimality for short execution time. This algorithm comprises two phases: (i) decomposition selection and (ii) mapping. Given an SG and all possible decompositions in the first phase, we measure a cost corresponding to each decomposition and the one with minimum cost is selected. This decomposition is given as the input of the mapping phase and based on a backtracking mechanism a placement of the corresponding NFs and logical interconnections on the physical network are determined. These two phases are further detailed in this section.

Decomposition selection. Given the physical network G_p, the service SG and all of its decompositions $Decomp$, the CF of each decomposition dc is calculated. As mentioned in Section 6.4, the objective is to prioritize a decomposition with lower CF and the advantage of this prioritization is twofold: (i) the number of same-type NFs which are directly connected is more and thus their interconnection is less complex and (ii) the more the number of NFs with the same type, the more NFs might be mapped to a same physical node and if they are directly connected, this leads to less network resource consumption as no physical links are used for the mapping.

In addition, for each NF in a decomposition dc, the candidate physical nodes which can potentially host that NF are determined. A candidate physical node should support the NF type (VM, process, I/O or hardware appliance) and should have enough capacity to meet the requirements of the NF. We define parameter p to be the minimum number of candidate physical nodes for NFs of a dc. To have a concrete example, consider a service decomposition composed of NF1 and

NF2 which can be hosted potentially by 2 and 4 physical nodes respectively. Then $p = min\{2, 4\} = 2$. This parameter is defined to enable selection of a decomposition which is less restricting. It enables a resource-aware decomposition selection which means that a decomposition with NFs which can potentially be mapped to more physical nodes is selected. Such a decomposition enables embedding more service requests over time. We define a cost function which combines CF, p and n (with n being the number of NFs in a decomposition).

$$C(dc) = a \cdot 1/p_{dc} + b \cdot CF_{dc} + g \cdot n_{dc}$$

In the decomposition selection phase of the algorithm, a decomposition with minimum cost $C(dc)$ is selected. The parameters a, b and g are defined to tune the impact of different factors in the cost function. The pseudo code of this phase is illustrated in Algorithm 1.

Mapping. The selected decomposition dc in the first phase is given as the input of the mapping phase. In this phase, first the NFs of dc are clustered based on their types and their interconnection (this was explained for CF calculation). Then the clusters and the corresponding NFs are sorted based on their requirements in descending order. We start mapping the NFs of the cluster with maximum requirement.

For each unmapped NF in the sorted list, all of its candidate physical nodes are sorted based on their distance (in terms of hop count) to the already used physical nodes in ascending order. This way we first check the physical nodes which are closer to the rest of the used nodes and thus, a lower number of physical links might be used for mapping of the logical interconnections. We select a physical node from the sorted list by which the logical links connected to the NF can also be mapped in the physical network. If none of the candidate physical nodes can provide such a mapping, the algorithm backtracks to the previous mapped NF and checks the next candidate. The pseudo codes for this phase are reported in

algorithm 1: DecompositionSelection pseudo code

DecompositionSelection($Decomp$**)**
Data: service decompositions $Decomp$
Result: minimum cost decomposition
$Cost$=[];
for $dc \in Decomp$ **do**
  CF_{dc}= number of clusters;
  p_{dc}= minimum (number of candidate physical nodes for NFs in dc);
  n_{dc}= number of NFs;
  $Cost(dc) = a.1/p_{dc} + b.CF_{dc} + g.n_{dc}$
end
select dc with minimum $Cost(dc)$;

algorithm 2: ServiceMapping pseudo code

ServiceMapping (dc)
Data: selected decomposition dc
cluster dc;
$sortedNF$ =sort clusters and their NFs based on their requirements in descending order;
for $NF \in sortedNF$ **do**
  $candidate$= sort the candidate nodes for NF based on their distance to the used physical nodes;
  $i = 0$;
  while *MapNF(NF,candidate[i])==false* **do**
    $i+ = 1$;
    if $i \geq length(candidate)$ **then**
      backtrack to previous mapped NF and select the next candidate in its sorted list;
    end
  end
end

algorithm 3: MapNF pseudo code

MapNF(NF**,** $pnode$**)**
if *not enough capacity in pnode to map NF* **then**
  return false;
end
$success = CheckLinks(NF, pnode, mappedNodes)$;
if $success == true$ **then**
  $mappedNodes+ = (NF, pnode)$;
  $CPU_{pnode} - = CPU_{NF}$;
  $Memory_{pnode} - = Memory_{NF}$;
  $Storage_{pnode} - = Storage_{NF}$;
  return true;
end

Algorithms 2-5 which are detailed in the rest of this section.

Given the selected dc, the **ServiceMapping** function clusters and sorts the NFs and the candidate nodes as explained above. It then tries to map the NFs of the dc one by one by invoking **MapNF** function. If all the candidate physical nodes are checked and the mapping of an NF is unsuccessful, it backtracks to previous mapped NF.

The **MapNF** function, shown in Algorithm 3, checks that available resources such as CPU, memory and disk in the physical node are sufficient for hosting the NF. It then checks that all the connected links to the NF can be mapped to the

algorithm 4: CheckLinks pseudo code

```
CheckLinks(NF, pnode, mappedNodes)
for l ∈ links attached to NF do
    if neighbor attached to l ∈ mappedNodes then
        n= the physical node that the neighbor is mapped to;
        while true do
            path=shortest path between n and pnode;
            if path == Null then
                return false;
            end
            success = CheckQoS(path);
            if success == true then
                for link ∈ path do
                    BW_link− = BW_l
                end
            end
        end
    end
end
return true;
```

physical network by invoking **CheckLinks** function. If it is successful in finding a mapping for all the links, the NF demands are reduced from the resources.

In the **CheckLinks** function, represented in Algorithm 4, it is checked whether all the links connected to an NF can be mapped in the physical network. This function iterates over all the links adjacent to the NF and if the NF attached to the other side of the link was mapped, it checks if there is a path between the physical nodes used for the mapping of the NFs. The shortest path between the two physical node is considered. If such a path exists, it is checked for the QoS requirements.

The **CheckQoS** function, shown in Algorithm 5, checks whether the QoS requirement of the logical link is satisfied by the given path. This function determines whether the bandwidth and delay requirements of the path are fulfilled. If all the above functions are successful a mapping for the requested service is found.

6.7 Performance evaluation

In this section we first describe the simulation environment and then the evaluation results are presented. The goal of the evaluations is to show the impact of the service decomposition choices on the resource footprint.

The focus of the experiments is on quantifying the added value of considering service decompositions at the time of the embedding in terms of cost, acceptance ratio and cost/revenue ratio. In the simulations, we compare the heuristic-based

algorithm 5: CheckQoS pseudo code

CheckQoS($path$)
for $link \in path$ **do**
  $EndtoEndDelay+ = delay_{link}$;
  if *EndtoEndDelay exceeds the constraint* **then**
    remove the link from the network graph;
    return false;
  end
  if $BW_{virtuallink} > BW_{link}$ **then**
    remove the link from the network graph;
    return false;
  end
end
return true;

Table 6.1: List of compared algorithms

Notation	Algorithm description
ILP	Proposed ILP-based algorithm
DSBM	Proposed heuristic-based algorithm
ILP-5	ILP on SGs with 5 NFs
ILP-10	ILP on SGs with 10 NFs
DSBM-5	DSBM on SGs with 5 NFs
DSBM-10	DSBM on SGs with 10 NFs
ILP-random	ILP with random decomposition
DSBM-random	DSBM with random decomposition

approach with the ILP-based algorithm in network scenarios where ILP can be executed in reasonable time scale. We evaluate the effect of employing service decompositions in both schemes. The following approaches are compared in the evaluations: (i) ILP-based algorithm, (ii) DSBM algorithm, (iii) ILP-based algorithm given one decomposition (random selection) and (iv) DSBM given one decomposition (random selection). As there is no other approach which considers service decompositions in the embedding problem, we compared both DSBM and ILP-based algorithms with approaches in which decomposition is selected randomly. The latter refer to approaches in which one decomposition is selected randomly and it is mapped using DSBM or ILP respectively. With such a comparison, we can clearly see the effect of a wise decomposition selection on the performance of the embedding. Table 6.1 presents the notations used for the compared approaches. In DSBM, we also evaluate the effect of different factors in $C(dc)$ on the performance of the embedding by tuning a,b and g parameters. Furthermore,

we report the execution time of both ILP-based and DSBM algorithms in different size physical networks to have an overview on the scalability of both approaches.

6.7.1 Simulation environment

The simulation environment is based on Python code. Libraries such as Networkx and Numpy are used for graph-based and numerical implementations. PuLP is an LP modeler in Python which is used to generate the LP files and the ILP model is solved using the included Cbc solver (from COIN-OR[2]).

For the physical network we considered two scenarios: (i) small network scenario and (ii) large network scenario. We used topologies from the Internet Topology Zoo, [3] which model real world ISP and backbone networks. These topologies are available in GML format. We have parsed these files and converted them to graphs in Networkx library in Python. For the small network, we considered the 'BT Europe' topology with 24 nodes and 37 edges. In the large network scenario, the 'Interoute' topology is used which is an international telecommunications service provider with the Europe's largest cloud services platform. This network is composed of 110 nodes and 148 edges. For both scenarios, it is assumed that some of the nodes have general purpose servers supporting different virtualization technologies, and some of them have specific hardware appliances such as a Firewall. This is selected randomly for the nodes. The CPU, memory and storage capacity of the nodes and bandwidth of the links are numbers uniformly distributed between 100 and 150 in both network scenarios. The cost of each unit of capacity is 1. As only propagation delay is considered in this paper, the delay of each physical link is selected proportional to the distance between the two attached nodes. In the selected topologies, nodes are annotated with their geographic coordinates (latitude and longitude). We used these coordinates to calculate the geographical distance between every two adjacent nodes and set the delay of the corresponding link accordingly. The resulting delays range between 1 and 30 time units.

The parameters reflect the commonly chosen simulation setup in the VNEP literature (see e.g. [18]).

The service requests arrive over time in a Poisson process with an average rate of four requests per 100 time units, each of which has a lifetime, exponentially distributed with an average of $\mu = 1000$ time units. Each request can be realized with a few decompositions which is a number between 2 and 5 with uniform distribution. The number of NFs within each of the decompositions is a number uniformly distributed between 2 and 10. The CPU, memory and storage demands of each NF is a number with uniform distribution between 1 and 20. The NF types are assigned randomly. The bandwidth requirement of each link is a number be-

[2] https://projects.coin-or.org/Cbc
[3] http://www.topology-zoo.org/

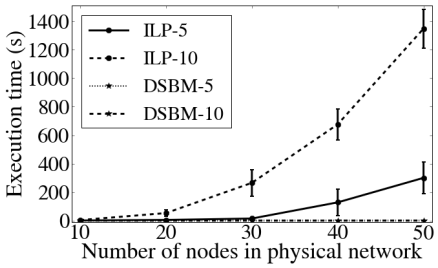


Figure 6.4: Execution time of ILP and DSBM for SGs with 5 and 10 NFs. The 95% confidence interval of the reported average values is depicted.

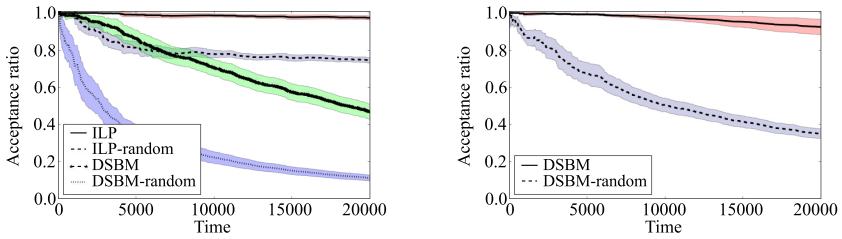
tween 1 and 50, uniformly distributed. The maximum allowed delay of each link is set to 1000 time units. Each pair of NFs within a decomposition is connected with probability 0.5. As a service graph is typically formed as a Directed Acyclic Graph with topologies such as a simple path or a forking path (see IETF drafts [19] on use cases for Service Function Chain), it is checked that: (i) the generated decomposition graph is connected and (ii) there is no cycle in the graph.

The hardware used to run the simulation is Intel Xeon quad-core CPU at 2.40 GHz with 12 GB RAM. Each simulation scenario is iterated 10 times and the average over all the iterations is reported.

6.7.2 Performance metrics

We measure the following performance metrics to evaluate and compare the proposed schemes:

- Execution time: this metric measures the time used by an scheme to find an embedding for a service request.
- Acceptance ratio: it measures the ratio of the accepted service requests which refer to services that are successfully mapped to the physical network.
- Embedding cost: the embedding cost or mapping cost is equivalent to the cost of the total substrate resources used for mapping a service request. In our evaluations, as the cost of each unit of capacity is set to 1, the embedding cost is equal to the total CPU, memory, storage capacity of nodes and bandwidth capacity of the links which are reserved for a service request.
- Embedding cost/revenue: this is the ratio between the embedding cost and the revenue of a service request. We define the revenue of a service request as the sum of the total resource demands of that request. In our evaluations, these demands are in terms of CPU, memory, storage requirements of the NFs and the required bandwidth in links.



(a) BT Europe network (24 nodes)

(b) Interoute network (110 nodes)

Figure 6.5: Service request acceptance ratio over time. The shaded background behind each curve represents the 95% confidence interval on the reported average values.

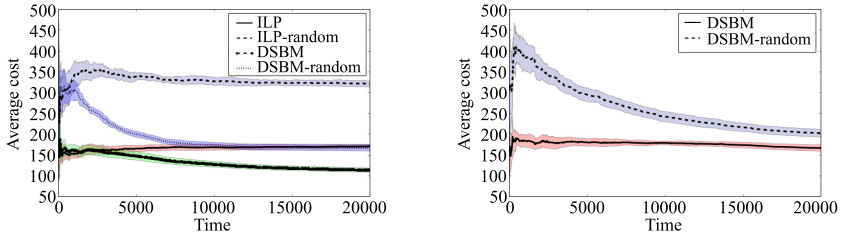
6.7.3 Evaluation results

Before detailing the evaluation results on the two network scenarios explained earlier, we report the results related to the execution time of the proposed algorithms on different size physical networks ranging from 10 to 50 nodes to observe the scalability behavior of the schemes. The topologies were randomly generated and the service requests of two sizes, 5 and 10 were considered. Fig. 6.4 depicts the execution time of different schemes. As expected, execution time of the ILP-based algorithm increases almost exponentially with the increase in the network size. This increase is more if the number of NFs in the service requests increases as well (see ILP-5 compared to ILP-10 in Fig. 6.4). The heuristic-based approach scales significantly better and the execution time in DSBM-10 does not exceed a few 100 ms.

In the rest of this section, the simulation results for small network and large network scenarios are presented. The heuristic-based algorithm was scalable to large network scenario and lead to similar conclusions in both scenarios. Due to scalability issues, the ILP-based approach was only evaluated in the small network scenario.

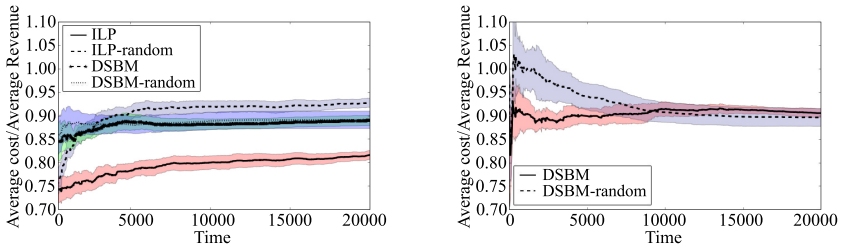
We measured the average acceptance ratio, the corresponding embedding cost and the average cost/average revenue ratio for service requests over time. We report these performance metrics against time to indicate how different schemes perform in the long run.

Fig. 6.5 depicts the service acceptance ratio for 4 different approaches in the two network scenarios: (i) ILP, (ii) DSBM, (iii) ILP-random and (iv) DSBM-random. In both network scenarios, the results indicate significant improvements in terms of acceptance ratio in the proposed ILP-based and heuristic-based algorithms compared to approaches in which a random decomposition is selected. The acceptance ratio of DSBM is higher in the Interoute network compared to the results in BT Europe which is the result of having more resources in the network. In



(a) BT Europe network (24 nodes) (b) Interoute network (110 nodes)

Figure 6.6: Average cost of accepting requests over time. The shaded background behind each curve represents the 95% confidence interval on the reported average values.

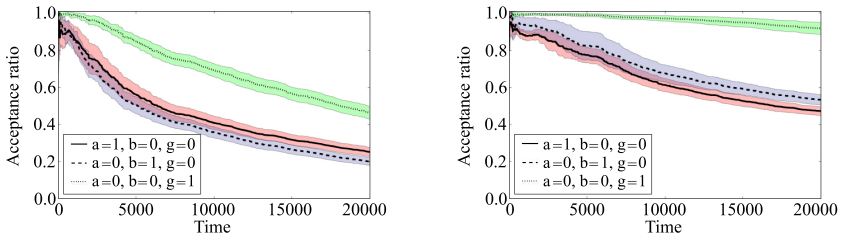


(a) BT Europe network (24 nodes) (b) Interoute network (110 nodes)

Figure 6.7: The ratio between average cost and average revenue over time. The shaded background behind each curve represents the 95% confidence interval on the reported average values.

DSBM the parameters in the $C(dc)$ function are set as follows: $a = 0.25$, $b = 0.25$ and $g = 0.5$. These parameters are tuned experimentally to achieve a reasonable performance. The effect of each factor in $C(dc)$ function on the embedding performance is evaluated and reported later in this section.

The average embedding cost in the four explained approaches are presented in Fig. 6.6. Comparing the proposed schemes and the ones with random decomposition selection, we observe a significant difference in the average cost of the embedding. Both ILP-random and DSBM-random consume more resources compared to ILP and DSBM respectively. Additionally, the results indicate that the ILP-based solutions lead to almost constant average costs while the heuristic solutions result in decrease of the embedding cost in the long run. The reason for such a behavior is explained by the optimality of the embedding solution. In DSBM, due to sub-optimal placement of the service requests, less requests can be accepted in the long run (presented in Fig. 6.5). Furthermore, as there are less available resources in the network compared to when an optimal placement is found, the



(a) BT Europe network (24 nodes)

(b) Interoute network (110 nodes)

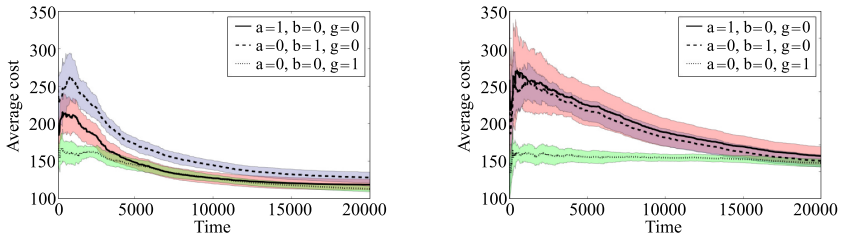
Figure 6.8: Service request acceptance ratio over time in DSBM. The shaded background behind each curve represents the 95% confidence interval on the reported average values.

service requests with less NFs can be accepted. This leads to a decrease in the average embedding cost over time. This behavior is also visible in the Interoute network for DSBM-random approach which is less efficient than DSBM.

Revenue gives an insight into how much an infrastructure provider (InP) will gain by accepting a service request however, it is not useful without considering the cost the InP will incur for mapping that request. Therefore, in the next figure, we report the ratio between average cost and average revenue in different schemes. Fig. 6.7 depicts the results for different approaches in the two network scenarios. Note that the smaller this ratio, the more efficient the scheme is. This means that for mapping a request, fewer resources are used compared to the gained revenue. However, this metric should be considered together with the acceptance ratio to enable a better conclusion regarding the efficiency of the schemes. An efficient scheme should have high acceptance ratio with low cost/revenue ratio. In Fig. 6.7, ILP leads to the lowest ratio while ILP-random has the highest one. Both DSBM and DSBM-random lead to very similar ratios. DSBM performs slightly better compared to DSBM-random which is more noticeable in the Interoute network. As we see, this ratio is even higher than 1 for DSBM-random at the beginning which is the indication of the poor performance considering the very low acceptance ratio.

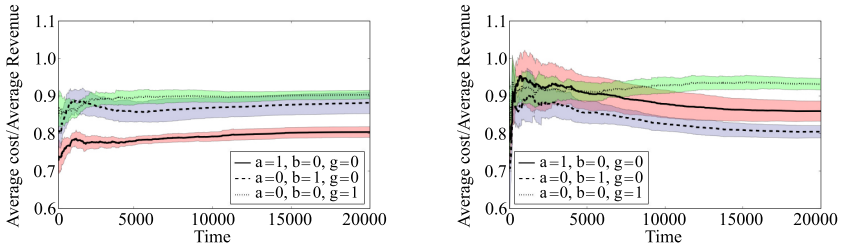
Next, we report the effect of different factors in the $C(dc)$ function of DSBM on the embedding performance. Fig. 6.8 illustrates the service request acceptance ratio when only one of the three factors in $C(dc)$ is considered for decomposition selection. Such an evaluation gives an idea on how to tune the parameters (a, b and g) in $C(dc)$ to have a more efficient embedding. Based on the results, considering the number of NFs in a request leads to higher acceptance ratio compared to cases when only CF ($b = 1$) or p ($a = 1$) are considered. The last two cases result in very similar performance. The related average embedding cost is presented in Fig. 6.9 and as expected, selection of decompositions with less number of NFs leads to lower cost compared to the other two cases.

Fig. 6.10 illustrates the ratio between average cost and average revenue in



(a) BT Europe network (24 nodes) (b) Interoute network (110 nodes)

Figure 6.9: Average cost of accepting requests over time in DSBM. The shaded background behind each curve represents the 95% confidence interval on the reported average values.



(a) BT Europe network (24 nodes) (b) Interoute network (110 nodes)

Figure 6.10: The ratio between average cost and average revenue over time in DSBM. The shaded background behind each curve represents the 95% confidence interval on the reported average values.

DSBM when only one of the factors in $C(dc)$ function is considered. In both networks, when only number of NFs in a request is considered in the $C(dc)$ calculation (i.e., $g = 1$), a higher ratio is achieved. This was expected as this factor is used to select the smaller service decompositions which lead to decreased revenue. Looking at both acceptance ratio and the ratio between average cost and average revenue, considering one single factor is not efficient enough and thus a combination of these factors is required in the decomposition selection phase to have a low-cost embedding with high revenue and high acceptance ratio.

6.8 Conclusion

A network service chain consisting of several network functions can often be realized in multiple ways, as there are multiple options on how to decompose a high-level network function into several less abstract network functions. The process of converting a service chain with abstract network functions to service chains with

more refined network functions is referred to as service decomposition. In this paper, we proposed two novel approaches to use service decompositions at the time of service embedding: (i) an ILP-based algorithm and (ii) a heuristic-based algorithm composed of two phases: decomposition selection and backtracking-based mapping. The heuristic-based approach solves the scalability limitation of the ILP-based algorithm at the cost of less efficient embeddings. The algorithms minimize the resources consumed to map a service request on a physical network while fulfilling service QoS requirements. As a result, more services can be mapped over time and the service acceptance ratio increases in the long run. Importantly this work is different from other existing works in the sense that both service decomposition selection and the embedding are solved at the same time. The proposed approaches were evaluated thoroughly in a simulation environment and were compared with approaches in which decompositions were selected independent of their demands and available resources in the network. The experimental results indicate that significant improvements in terms of acceptance ratio and mapping cost can be achieved if service decompositions are employed at the time of the embedding and a reasonable selection is made. Additionally, simulation results have shown that unlike the ILP-based algorithm, the heuristic-based approach DSBM is scalable and can be used in large-scale networks, where waiting for the optimal solution is prohibitive.

Acknowledgment

This work was conducted within the framework of the FP7 UNIFY project: Grant Agreement No. 619609 and is partially funded by the Commission of the European Union and partly funded by the UGent BOF/GOA project: B/13343/01 'Autonomic Networked Multimedia Systems'.

References

[1] A. Fischer, J. F. Botero, M. Till Beck, H. De Meer, and X. Hesselbach. *Virtual network embedding: A survey*. Communications Surveys & Tutorials, IEEE, 15(4):1888–1906, 2013.

[2] R. McGeer, D. G. Andersen, and S. Schwab. *The network testbed mapping problem*. In Testbeds and Research Infrastructures. Development of Networks and Communities, pages 383–398. Springer, 2011.

[3] S. Sahhaf, W. Tavernier, D. Colle, and M. Pickavet. *Network service chaining with efficient network function mapping based on service decompositions*. In 1st IEEE Conference on Network Softwarization, NetSoft 2015, 2015.

[4] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache. *Virtual network provisioning across multiple substrate networks*. Computer Networks, 55(4):1011–1023, 2011.

[5] J. F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. De Meer. *Energy efficient virtual network embedding*. Communications Letters, IEEE, 16(5):756–759, 2012.

[6] X. Zhang, C. Phillips, and X. Chen. *An overlay mapping model for achieving enhanced qos and resilience performance*. In Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2011 3rd International Congress on, pages 1–7. IEEE, 2011.

[7] G. Schaffrath, S. Schmid, and A. Feldmann. *Optimizing long-lived cloudnets with migrations*. In Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing, pages 99–106. IEEE Computer Society, 2012.

[8] A. Hammad, R. Nejabati, and D. Simeonidou. *Novel methods for virtual network composition*. Computer Networks, 67:14–25, 2014.

[9] J. Lischka and H. Karl. *A virtual network mapping algorithm based on subgraph isomorphism detection*. In Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures, pages 81–88. ACM, 2009.

[10] H. Yu, C. Qiao, V. Anand, X. Liu, H. Di, and G. Sun. *Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures*. In Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE, pages 1–6. IEEE, 2010.

[11] W.-L. Yeow, C. Westphal, and U. C. Kozat. *Designing and embedding reliable virtual infrastructures*. ACM SIGCOMM Computer Communication Review, 41(2):57–64, 2011.

[12] G. Sun, H. Yu, L. Li, V. Anand, and H. Di. *The framework and algorithms for the survivable mapping of virtual network onto a substrate network*. IETE Technical Review, 28(5):381–391, 2011.

[13] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann. *VNE-AC: Virtual network embedding algorithm based on ant colony metaheuristic*. In Communications (ICC), 2011 IEEE International Conference on, pages 1–6. IEEE, 2011.

[14] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann. *Applying NFV and SDN to LTE mobile core gateways, the functions placement problem*. In Proceedings of the 4th workshop on All things cellular: operations, applications, & challenges - AllThingsCellular '14, pages 33–38, New York, New York, USA, 2014. ACM Press. Available from: http://dl.acm.org/citation.cfm?doid=2627585.2627592, doi:10.1145/2627585.2627592.

[15] S. Mehraghdam, M. Keller, and H. Karl. *Specifying and placing chains of virtual network functions*. In Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on, pages 7–13. IEEE, 2014.

[16] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs. *Network Function Placement for NFV Chaining in Packet/Optical Datacenters*. Journal of Lightwave Technology, 33(8):1565–1570, 2014.

[17] A. A. P. Papadimitriou. *MIDAS: Middlebox Discovery and Selection for On-Path Flow Processing*. In 7th International Conference on COMmunication Systems and NETworkS, COMSNETS 2015, 2015.

[18] N. M. K. Chowdhury, M. R. Rahman, and R. Boutaba. *Virtual network embedding with coordinated node and link mapping*. In INFOCOM 2009, IEEE, pages 783–791. IEEE, 2009.

[19] W. Liu, H. Li, O. Huang, M. Boucadair, N. Leymann, Z. Cao, Q. Sun, and C. Pham. *Service Function Chaining (SFC) General Use Cases*. Technical report, Internet-Draft draft-liu-sfc-use-cases-08, 2014.

7

Scalable architecture for service function chain orchestration

In this chapter, we identify the major time consuming blocks in the service provisioning process. To this end, we implement the proposed embedding algorithm of the previous chapter in a proof of concept prototype. Based on the discovered blocks, different architectural improvements are proposed to enable a more scalable service provisioning framework.

⋆ ⋆ ⋆

S. Sahhaf, W. Tavernier, J. Czentye, B. Sonkoly, P. Sköldström, D. Jocha, J. Garay

Published and presented at Fourth European Workshop on Software Defined Networks, 2015.

Abstract Network Function Virtualization (NFV) enables to implement network functions in software-based, high-speed packet processing functions which traditionally are dominated by hardware implementations. Virtualized Network Functions (NFs) may be deployed on generic-purpose servers, e.g., in datacenters. The latter enables flexibility and scalability which previously were only possible for web services deployed on cloud platforms. The merit of NFV is challenged by control challenges related to the selection of NF implementations, discovery and

reservation of sufficient network and server resources, and interconnecting both in a way which fulfills SLAs related to reliability and scalability. This paper details the role of a scalable orchestrator in charge of finding and reserving adequate resources. The latter will steer network and cloud control and management platforms to actually reserve and deploy requested services. We highlight the role of involved interfaces, propose elements of algorithmic components, and will identify major blocks in orchestration time in a proof of concept prototype which accounts for most functional parts in the considered architecture. Based on these evaluations, we propose several architectural enhancements in order to implement a highly scalable network orchestrator for carrier and cloud networks.

7.1 Introduction

Network Function Virtualization (NFV) [1] enables to implement Network Functions (NFs) such as firewalls or NATs, high-speed packet-processing functions in software which traditionally are dominated by hardware implementations or dedicated middleboxes. NF software may be deployed on generic-purpose servers, e.g., in datacenters. Telecom services (e.g., an Intrusion Detection) which can be decomposed into a Service Graph (SG) of NFs, might now benefit from flexibility and scalability-levels which previously were only possible for web services deployed on cloud platforms such as Amazon EC or Google Compute. Modern control paradigms such as Software Defined Networking (SDN) [2] have the merit of simplifying the service chain provisioning process and reducing the cost in CAPEX and OPEX.

The mapping of NFs of services to infrastructure is one of the core tasks of the orchestrator. This requires that the orchestrator has a view of what are the individual resource requirements of NFs, or even if given NFs might be decomposed to smaller NFs (Service Decomposition). However, as indicated in Section 7.4, orchestration in realistic service and infrastructure provider contexts quickly involves ten thousands of resource elements, and a multiple of services to be orchestrated.

Existing research related to service orchestration has largely focused on: i) embedding algorithms in small-scale, idealized settings, i.e., limited number of infrastructure nodes, pre-determined SG decomposition, or on ii) NFV architectures, e.g., [1], providing high-level functionality and interfaces for enabling Service Function Chaining with virtual NFs.

Our contribution. In this paper we identify the context and requirements of NFV orchestration over Telecom and datacenter networks. We intend to bridge the gap between abstract embedding algorithms and NFV architectures, by proposing elements of a realistic and scalable resource orchestrator which is able to optimize placement of networking and computing components across infrastructure. Within

this state-of-the-art emulation NFV framework [3] we implement an orchestrator supporting decomposition, and identify most significant factors contributing to orchestration time. The latter serves the identification of elements for an improved resource orchestrator framework which is truly scalable, as well as the identification of a set of technologies which are able to implement such design.

The rest of the paper is as follows. Section 7.2 describes related work. In Section 7.3 we detail the concepts of Service Function Chaining (SFC). Section 7.4 details the problem and context of orchestration of services in a realistic setting and the relation of the orchestration functionality within NFV architectures. An overview of most important orchestration algorithm component is given in Section 7.5, while Section 7.6 identifies experimental performance results and Section 7.7 proposes architecture improvements. Section 7.8 concludes the paper.

7.2 Related work

Existing research on NFV orchestration focuses on two aspects: i) design and evaluation of embedding algorithms, and ii) design and implementation of control architectures for production environments. The first relates to the problem of mapping a set of service-related (virtualized) resources to physical infrastructure (e.g., servers and switches). Requested resources might involve networking (bandwidth), as well as node-level resources (computing or memory). The latter is referred to as the Virtual Network Embedding Problem (VNEP), for which a survey can be found in [4]. Recent approaches build on this state of the art, for example by integrating the concept of service decomposition [5].

The second category of research is driven by Telecom research and industry. The NFV Industry Specifications Group (NFV ISG) of European Telecommunications Standards Institute (ETSI), is driven by Telecom operators to strategically steer NFV-related activities. The ISG produced and publicly released documents on NFV terminology, use cases, requirements and architectures. NFV Management and Orchestration (MANO) is the ETSI-defined framework for the management and orchestration of computing, networking, storage and virtual machine resources in cloud and carrier networks. Multiple community-driven and private software initiatives, such as OpenMANO (https://github.com/nfvlabs/ openmano) or vConductor [6], are developing proof of concepts of proposed ETSI specs. These recent efforts involve single domain designs, architectures and prototypes where scalable orchestration is not the first focus. OpenStack is a cloud infrastructure management framework in which Heat is the component responsible for orchestration. Network connectivity orchestration here relies on Neutron and is heavily focusing on L3 and above, lacking fine-grained forwarding (L2) flexibility required for NFV orchestration. Larger frameworks such as OPNFV (http://www.opnfv.org/) intend to combine existing cloud and network con-

trol frameworks (e.g., OpenStack, OpenDayLight) to build a reference implementation for NFV management/control.

EU-funded projects such as FP7 T-NOVA or Mobile Cloud Networking (MCN) also investigate the merits of NFV in Telecom. T-NOVA focuses on an 'NFV Marketplace' composed of: i) a 'Network Function Store' including NFs by several 3rd-party developers and ii) a Brokerage platform enabling customers to trade with the T-NOVA service provider and 3rd-party function developers. MCN investigates NFV as an enabler for increased flexibility in the backhaul of mobile networks. Both projects define some kind of orchestrator for mapping virtual resources to physical infrastructure and managing the life-cycle of virtualized resources. These are ongoing, and do not directly focus on the design and implementation of a scalable, potentially multi-domain orchestrator, involving recursivity in the control and orchestration layers as the proposed approach.

7.3 Service function chaining

In order to easily introduce joint programmatic interfaces for controlling different types of resources, such as compute, storage and networking ones, we have defined a common model to be used at different reference points called Network Function Forwarding Graph (NF-FG) [7]. It provides support for functionalities such as resource orchestration or service decomposition, on the one hand, and features such as scalability, dynamicity or support for DevOps in Service Provider environments, on the other. The model is capable of storing service description as SG, resource information as Resource Graph (RG) and mapping of requests to resources as NF-FG (see Fig. 7.1, at left the SG is shown on top and NF-FG with mapping is shown at bottom).

The SG defines the service functions and their logical connectivity, the Service Access Points (SAPs) to the service and the Service Level Specification to meet the Service Level Agreement. It is only used as a standalone element when there are no resources involved yet.

The RG describes the (virtual) resources that will be used to deploy the requested services. It provides a homogeneous representation of the (virtualized) infrastructure, in terms of both capacities and capabilities, at the defined abstraction level. For example, in domains with hierarchical orchestration processes, the RG in the higher level orchestrators has a wider scope and abstracts away the finer grain details of the underlying resources, whereas the RG in the lower level orchestrators has a fine grain detail of the resources.

The NF-FG contains the assignment of NFs to the virtualized software resources; the definition of the forwarding behavior in the virtualized network resources and the service requirements which can be evaluated at the network and software abstraction layers.

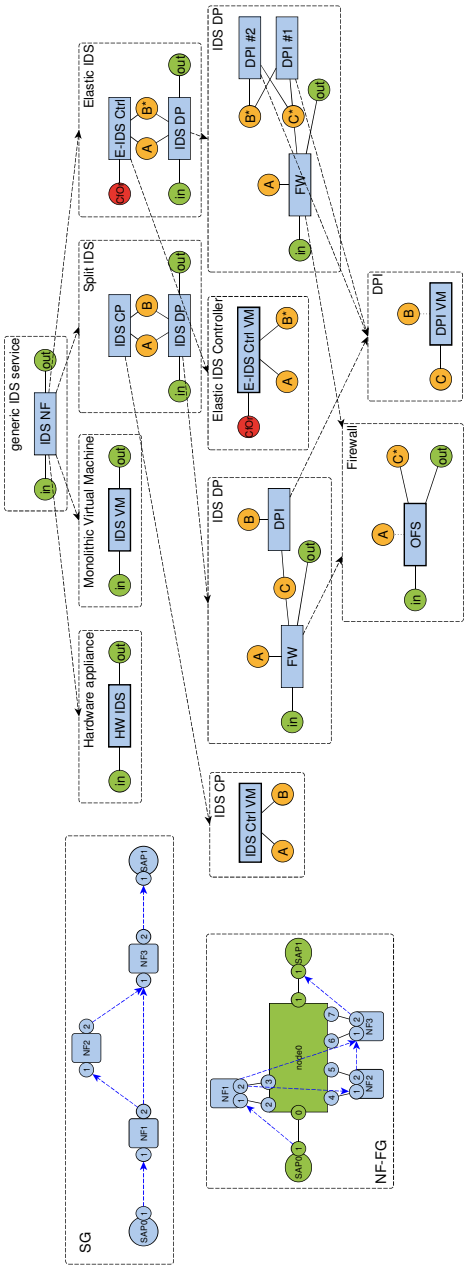


Figure 7.1: Example of SG, NF-FG and Network Function decomposition

The NF-FG evolves from its original definition as SG to RG mapping. On the one hand, while the NF-FG progresses down through the architecture it will be further characterized, the service decomposition process will also decompose the components (e.g. NFs) and it will be split into smaller subgraphs if deployed in different infrastructure domains. On the other hand, during the service lifecycle the NF-FG will also evolve from the initial deployment as a consequence of internal re-optimization processes, modifications to the service requested to the orchestrator or external changes (e.g. infrastructure updates, auto-scaling).

Service decomposition is the process of transforming an NF-FG containing abstract NF(s) to NF-FG(s) containing less abstract, more implementation-close NF(s). This can also include dividing the functionality of a complex NF to several, less complex NFs. This allows for a step-wise translation of high-level (compound) NFs into more elementary NFs, which can eventually be mapped onto the infrastructure. During the decomposition of an NF, the external interfaces remain unchanged. Formally, a decomposition rule can be seen as a NF→NF-FG mapping. There can be multiple decompositions for an NF.

A sample NF decomposition for an Intrusion Detection System (IDS) service is shown at the right of Fig.7.1. It can be implemented with a hardware appliance or a monolithic Virtual Machine (VM). The IDS control logic is decomposed into an IDS Control VM, a Firewall (FW) component to block the identified malicious traffic and a traffic analyzer. The FW may be mapped to a Forwarding Element (FE) and the traffic analysis is realized by a generic Deep Packet Inspection (DPI) VM component.

7.4 Service chain orchestration

An orchestrator is responsible for the service management and orchestration. The main functionalities of an orchestrator are: i) optimal mapping of Virtualized Network Functions (VNFs) across infrastructure, ii) instantiating VNFs at reasonable locations, iii) keeping track of VNFs location, iv) assigning and scaling resources to the VNFs and v) service VNFs monitoring.

Requirements. Orchestration process is impacted by scale of Telecom operator network and the number of service requests. Based on the discussions with network operators and information available on Telecom operator networks such as BT[1] and datacenters info in UK[2], some parameters and requirements for network topology, customers and their requests were identified. A typical Telecom operator network has a hierarchical structure with a dense core router meshed network consisting of inner and outer core Points of Presence (PoPs). The end customers are interconnected to this core network via a hierarchy of tree-structured access-

[1] http://www.kitz.co.uk/adsl/21cn_network.htm
[2] http://www.datacentermap.com/united-kingdom/

and metro aggregation networks. Considering BT, such an infrastructure consists of almost $50K$ devices at different parts in the network and datacenters excluding CPEs. In case of including CPEs, almost $10M$ devices are needed to be orchestrated. The number of new customers per day is equal to almost $2.5K$ and the number of service requests is estimated around $5-10K$ per day. There are several IETF drafts on use cases for Service Function Chain (SFC). Based on these drafts, a service chain request is typically a Directed Acyclic Graph with topologies such as a simple path or a forking path.

Capability. In order for the orchestrator to fully exploit the capabilities of the servers, all the features/capabilities available in the infrastructure should be identified. Devices such as Acceleration Hardware (AH) (e.g., FPGAs, GPUs and MICs) and advanced network interfaces cards can improve the performance of many NFs by offloading several performance-critical tasks in an NF to these devices. The challenges occur when the complexity of NFs increases which makes the implementation of NFs infeasible due to resource limitation in AH. A solution to this issue is the support of service decomposition in the orchestration process (see Section 7.3). Complex NFs can be decomposed to more elementary NFs and there should exist the hardware description of performance-critical NFs to be installed on AH. An important task is to design the NFs in such a way that efficient usage of specific capabilities is ensured. Note that for the mapping of NFs to the infrastructure, both hardware static metrics (e.g., location and supported features) and dynamic metrics (e.g., CPU utilization and current available memory) should be known to the orchestrator. Besides, values of resource demands should be coherent with node specification constraints. Depending on the form of the exposed resources, comparison of demands and available resources can be a challenging task. To address this issue, different profiling tools (e.g., GNU gprof or Tuning and Analysis Utilities (TAU)) can be used to measure the application's performance. Benchmarks such as the one provided by Standard Performance Evaluation Corporation (SPEC) enables direct comparison of processors' performance. Additionally, analytical techniques (e.g., Amdahl's law and Gustafson's law) can be used to model the performance of multi-core CPUs. In spite of existence of several profiling tools, it is challenging to have a benchmarking with high-accuracy.

Interface. A generic API for an orchestrator has to support the following operations: i) instantiate/tear down/change NF-FG: once an NF-FG arrives at the orchestrator, it tries to execute it based on its global resource view. Changing a request includes operations such as modifying the NF demands and inserting/removing NFs in the NF-FG ii) get/send virtual resource info: the orchestrator provides resources, capabilities and topology information iii) notification/alarm: any failure or unexpected event can be reported by the orchestrator iv) get/send observability info: measurement reports on Key Quality Indicators (KQIs) related to NFs can be provided by the orchestrator v) start/stop/restart NFs and switches vi)

connect/disconnect NFs to switches and vii) configure switches. Possible option for addressing the last operations is using different protocols at the southbound interface of the controllers such as OpenFlow, NETCONF and OFconfig.

7.4.1 ESCAPE framework

We have established a prototyping framework called ESCAPE[3] including 3 layers of Infrastructure Layer (IL), Orchestration Layer (OL), Service Layer (SL) and demonstrated the first version in [3]. The main goal of ESCAPE is to support the development of several parts of the service chaining architecture including VNF implementation, traffic steering, virtual network embedding, etc. However, here we focus on the orchestration part. ESCAPE is (mainly) implemented in Python on top of POX (OpenFlow controller) platform and Mininet. The modular approach and loosely coupled components make it easy to change several parts and evaluate own algorithms. The system architecture of the next version of ESCAPE (without the Mininet based IL) is shown in Fig. 7.2.

SL contains an API and a GUI at the top level where users can request and manage services and NFs. The API is capable of formulating SG from the request and passes that to a dedicated service orchestrator which is responsible for gathering resource information (RG) from Virtual resource manager. This is the virtual view provided by the Virtualizer of the lower layer. Mapping of SG to RG is delegated to the SG mapper module which constructs an NF-FG storing the request, the virtual resources and the mapping between NFs and infrastructure nodes.

OL encompasses the most important components of the resource orchestration process which replaces the ETSI's Virtualized Infrastructure Managers (VIMs). An API is set up on the top centralizing the interaction with the upper layer. On the one hand, the request coming as an NF-FG is forwarded to the Resource Orchestrator (RO) via the corresponding Virtualizer (which is responsible for policy enforcement as well). On the other hand, the virtual view created and managed by the Virtualizer is provided as an RG to the upper layer. RO is the key entity managing the components involved in the orchestration. The input is an NF-FG which should be mapped to the abstract domain view provided by the Domain Virtualizer. RO collects and forwards all required data to RO mapper. More specifically, the NF-FG, the domain view (as an RG) and the Network Function Information Base (NF-IB) are passed to the RO mapper which invokes the configured mapping strategy and interacts with the Neo4j graph database containing information on NFs and decomposition rules[4] (see Section 7.4.2). The outcome is a new NF-FG which is sent to the Controller Adaptation part. The role of Controller Adapter

[3]Extensible Service ChAin Prototyping Environment using Mininet, Click, NETCONF and POX (ESCAPE)

[4]NF-IB corresponds to "VNF Catalogue" in NFV MANO with the difference of supporting service decomposition.

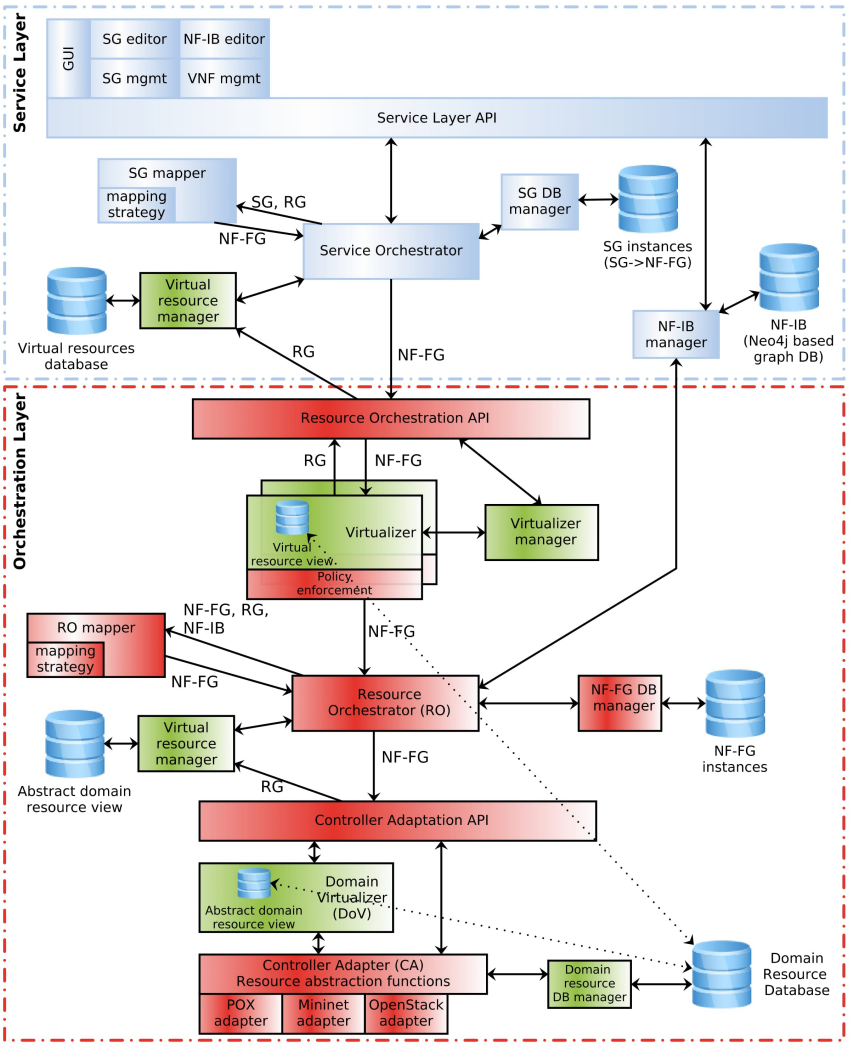


Figure 7.2: System architecture of ESCAPE

(CA) is twofold. First, it gathers technology specific information on resources of different domains then builds an abstract domain view. The interaction with different types of technology domains are handled by adapters (e.g., OpenStack adapter for clouds managed by OpenStack). Second, the incoming NF-FG request is decomposed according to the low-level domains and delegated to the corresponding adapters.

7.4.2 Network Function-Information Base (NF-IB)

The NF-IB is the entity responsible for storing the NF models/abstractions, NF relationships, NF implementation image(s) and NF resource requirements (see Fig. 7.2). The NF-IB supports the definition of abstract NFs such as a FireWall, referring to a type, a potential number of ports/interfaces, as well as dependencies to other NFs. As explained in Section 7.3, abstract NFs might be implemented through more refined NFs or might be decomposed themselves into multiple NFs interconnected into an NF-FG with the same external interfaces as the higher-level NF. The NF-IB is capable of storing these relationships into a tree-like data structure in support of the decomposition process (cfr. Fig. 7.1). The leaves of the decomposition tree are NFs for which low-level implementation and deployment information is available such as images, provisioning scripts, resource requirements in terms of CPU, memory and storage.

We have implemented the NF-IB in Neo4j database. As this database is capable of storing key-value pairs for nodes and edges, for each NF we have stored the explained tree-like structure with all the corresponding information of nodes and links. Several modules have been implemented to enable $i)$ updating of the database and $ii)$ retrieval of all possible decompositions of a given SG.

7.5 Embedding algorithm

We have implemented a proof of concept embedding algorithm which supports service decomposition using the Neo4j-based NF-IB explained in Section 7.4.2. It is implemented in Python in compliance with the ESCAPE framework. Given an NF-FG to this module, it retrieves all possible decompositions from the NF-IB and selects a suitable decomposition which is mapped to the network infrastructure. In order to connect to the Neo4j-based NF-IB from Python, we have used Py2neo library. The embedding algorithm was proposed in [5] but was only evaluated in terms of service acceptance ratio. As detailed in [4], there exist several algorithmic approaches in the literature to solve the embedding problem. Importantly, the implemented embedding algorithm is different from the existing approaches in the sense that service decompositions are taken into account at the time of embedding and a resource-aware selection is made. We briefly explain this algorithm in this section.

The objective of the algorithm is to minimize the embedding cost which is achieved by minimizing the resources consumed in the infrastructure to map a request. This allows accepting more requests over time and increases the acceptance ratio. As service decompositions are known from the design time, we can make a resource-aware decomposition selection which would certainly improve the performance of the embedding as a reasonable decomposition is selected which cor-

responds to the existing resources and thus leads to better placement of the NFs.

The algorithm is based on a backtracking mechanism and is composed of two phases: i) Decomposition selection and ii) Mapping.

In the first phase, given an NF-FG all of its possible decompositions are retrieved from the Neo4j-based NF-IB. For each decomposition a cost is calculated based on: i) number of NFs in the decomposition, ii) number of candidate physical nodes with sufficient capacities which can potentially host the NFs in the decomposition and iii) Cluster Factor (CF) which is calculated as follows: the NFs with similar types which are directly connected (without intermediate NFs with other types) are grouped in a same cluster. The number of clusters in the decomposition is the CF of that decomposition.

NFs types refer to the implementation of the NFs as they can be implemented through different techniques such as: Virtual Machine (VM) images in different virtualization techniques (e.g. Xen, Vmware), process in a container, packet I/O drivers (e.g. DPDK), or hardware appliances. It is of great importance to take NFs types into account at the time of the embedding because not all physical nodes of the infrastructure support all types.

CF is taken into account in the cost function to enable more efficient resource consumption. The more the number of NFs with the same type, the more NFs might be mapped into a same physical node. This leads to less resource consumption, if the similar-type NFs are interconnected directly.

The minimum cost decomposition is selected in the first phase of the algorithm. The mapping phase is based on a backtracking mechanism which tries to minimize the resource consumption of the mapping. The NFs of each cluster (see explanation for CF calculation) are sorted based on their requirements in descending order and the mapping of the NFs of the cluster with maximum requirement starts first. For each of the unmapped NFs, we sort its corresponding candidate physical nodes based on their distance (hop count) to the used physical nodes in ascending order. Every time a physical node is selected to host an NF it is checked if all connected links to the NF can be mapped as well. If not, another candidate physical node is investigated. If none of the nodes can host the NF the algorithm backtracks to the previous mapped NF and selects another candidate node. For more detailed explanation of the algorithm, we refer the interested readers to [5].

7.6 Performance evaluation

The goal of experiments in this paper is to identify the major blocks in orchestration time in the implemented proof of concept prototype. Additionally, we see the effect of increase in the topology size, SG size and number of service decompositions on the performance of the embedding.

In our recent work [5], we have evaluated the proposed embedding algorithm in

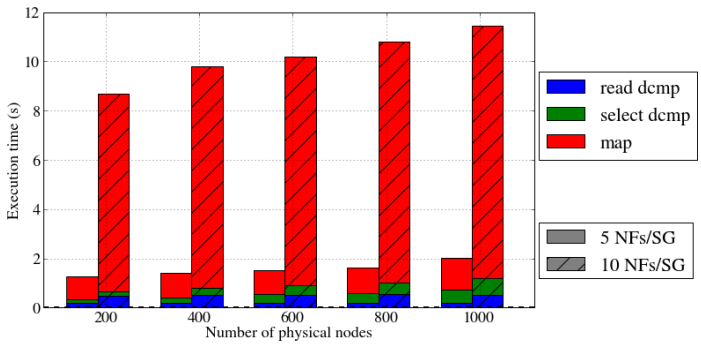


Figure 7.3: Embedding execution time for SGs with one decomposition

terms of cost and acceptance ratio. We refer the interested readers to [5] to see the added value of considering service decompositions at the time of the embedding and the impact of the service decomposition choices on the resource footprint.

As our intention is to evaluate the embedding execution time on physical topologies with different sizes, we have generated random regular networks with 100-1000 nodes with degree 3. For each of the generated topologies, the resources of nodes such as memory, storage and CPU and the links bandwidth and delay are numbers uniformly distributed between 100-300. The SGs are also generated randomly and each pair of nodes is connected with probability 0.5. The resource demands of NFs and links within an SG are numbers uniformly distributed between 1-20. Each scenario is iterated 50 times and the average value is reported.

In the first experiment, we have evaluated the execution time of the embedding of an SG into physical networks of different sizes for two scenarios: i) SGs with 5 NFs and ii) SGs with 10 NFs. Each SG has only one decomposition. Fig. 7.3 reports the execution time of different blocks in the embedding algorithm. These blocks include: i) retrieving/reading of all decompositions from the NF-IB (read dcmp), ii) decomposition selection (select dcmp) and iii) mapping of the selected decomposition (map). Based on the results, the mapping is the dominant block and it increases significantly with the increase in the number of NFs, when only one decomposition exists for an SG.

The next experiment evaluates the effect of increase in the number of decompositions for an SG. Fig. 7.4 reports the execution times for 3 scenarios in which the number of decompositions per NF in an SG changes from 2 to 4. As there are 5 NFs in each SG, the number of decompositions in each scenario is: 5^2, 5^3 and 5^4. As we see 'map' and 'read dcmp' blocks seem to scale quite well, whereas 'select dcmp' is the block which scales poorly with increasing number of nodes in the network. For small topologies with few nodes, 'read dcmp' is the dominant

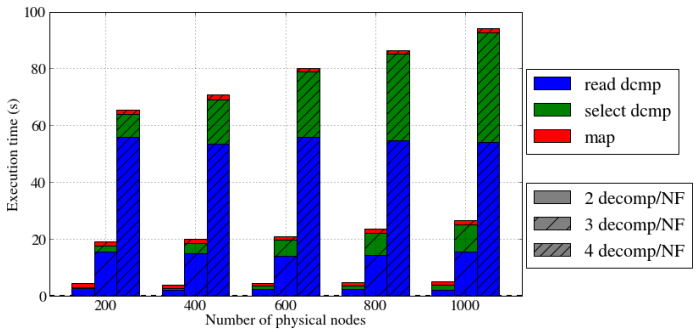


Figure 7.4: Embedding execution time for SGs with 5 NFs

block while for larger topologies (1000 nodes and more) 'select dcmp' seems to be the main concern. It is worth mentioning that the 'read dcmp' block includes the time for reading NFs decomposition from the NF-IB and the time needed for calculating the service decompositions. This is because only NFs decompositions are stored in the NF-IB and possible service decompositions should be calculated upon request.

The experiments in this section clearly identified the major blocks in the orchestration time in the proposed embedding approach. Knowing these blocks, in the next section, we propose several architectural enhancement to improve the scalability of the orchestrator.

7.7 Discussion: Towards a scalable orchestration framework

In this section, we propose several architectural enhancements and explain existing challenges in order to implement a highly scalable orchestrator and meet the requirements mentioned in Section 7.4.

Parallel/distributed embedding. Based on the results, we identified the 'read dcmp' block to be the most time consuming block in the embedding in smaller topologies while 'select dcmp' block seems to be a major issue in larger topologies. Changing the embedding algorithm to a distributed approach in which costly calculations are done in parallel can improve the performance of the embedding significantly. A possibility to parallelize the 'read dcmp' block of the algorithm is to use Neo4j which supports High-Availability (HA) by distributing the full database onto multiple nodes, resulting in database read performance that scales near linearly with the number of nodes in the cluster. Using the Neo4j HA the 'select dcmp' block can simply be computed in parallel as the cost calculation of each decomposition is independent of others. However, parallelizing the 'map' block is

a challenging task. This phase is equivalent to the typical VNEP as SGs composed of atomic NFs are similar to virtual networks which should be mapped to a physical infrastructure and thus similar solutions to VNEP can be considered for the mapping phase (e.g. [8]). The options for mapping parallelization are: i) considering all possible combination of NFs mapping to the physical nodes and selecting the minimum cost mapping. The feasibility/cost of each mapping can be checked in parallel. This approach is feasible only in small topologies (o(100) nodes) as the number of combination increases drastically with a small increase in the topology size, ii) selecting the first-fit physical node for mapping of NFs and finding the shortest path between nodes. NFs mapping/path calculation can be done in parallel. If the first-fit mapping is not successful, the next one is selected. A challenge is to avoid different threads reserving the same resource. Batch scheduling is a solution in which each job gets dedicated access to the resources.

Hierarchical embedding. The other alternative to achieve a scalable orchestration process is to have a hierarchical embedding process. In this process, SG can be divided into different subgraphs using service decompositions available in the NF-IB and each subgraph can be given to a different domain to be orchestrated locally. The main challenge in such distributed embedding relates to the amount of resource and infrastructure information that needs to be advertised to the upper layer orchestrators to facilitate an efficient embedding process. Each domain may expose to upper layers only high-level and aggregated information such as total available capacities and capabilities or aggregated PoP-level information instead of detailed router-level topologies. Such incomplete information in the higher layer orchestrator might lead to inefficient embeddings with performance far from the optimal solution. It is a challenging task to identify the trade-off between the efficiency of the embedding and the amount of infrastructure information exposed by each domain.

Pre-defined service chains. Another enhancement option, independent of the embedding approach, is to have pre-defined service chains with pre-defined decomposition templates. With such templates different parts of the embedding can be done proactively.

7.8 Conclusion

In Service Function Chaining (SFC), virtualized Network Functions (NFs) are chained to compose a network service. This paper has focused on the design of an adequate resource orchestrator to steer the control of SFCs. The main goal of an orchestrator is to map network functions of a requested service (i.e., service function chain) to infrastructure network and compute resources. Orchestration might involve thousands of requests in the period of one business day to be mapped on one or more infrastructure provider networks involving ten thousands of network

elements. Scalability is therefore an important characteristic of an orchestrator component. The system architecture, related components and a new service representation model were explained in detail. Important elements of mapping algorithms were characterized and an algorithm supporting service decomposition was implemented as a proof of concept. The key time consumers within the implemented PoC were identified, and a scalable distributed orchestrator architecture, as well as related technologies were proposed based on these findings.

Acknowledgment

This work was conducted within the framework of the FP7 UNIFY project, which is partially funded by the Commission of the European Union.

References

[1] ETSI. *White Paper: Network Functions Virtualisation (NFV)*, 2013. Available from: http://portal.etsi.org/NFV/NFV_White_Paper2.pdf.

[2] ONF. *Open Networking Foundation*, 2014. Available from: https://www.opennetworking.org/.

[3] A. Csoma, B. Sonkoly, L. Csikor, F. Nemeth, A. Gulyas, W. Tavernier, and S. Sahhaf. *ESCAPE: Extensible Service ChAin Prototyping Environment using Mininet, Click, NETCONF and POX. Demonstation.* In ACM SIGCOMM 2014, 2014.

[4] A. Fischer, J. F. Botero, M. Till Beck, H. De Meer, and X. Hesselbach. *Virtual network embedding: A survey*. Communications Surveys & Tutorials, IEEE, 15(4):1888–1906, 2013.

[5] S. Sahhaf, W. Tavernier, D. Colle, and M. Pickavet. *Network service chaining with efficient network function mapping based on service decompositions*. In 1st IEEE Conference on Network Softwarization, NetSoft 2015, 2015.

[6] W. Shen, M. Yoshida, K. Minato, and W. Imajuku. *vConductor: An enabler for achieving virtual network integration as a service*. Communications Magazine, IEEE, 53(2):116–124, 2015.

[7] W. Tavernier, S. Sharmaa, S. Sahhaf, R. Szabó, D. Jocha, P. Sköldström, J. Matias, J. Garay, G. Agapiou, B. Sonkoly, M. Rost, T. Jungel, A. Rostami, and X. Cai. *D3.1 Programmability framework*. Deliverable 3.1, UNIFY Project, October 2014. Available from: https://www.fp7-unify.eu/files/fp7-unify-eu-docs/Results/Deliverables/UNIFY_D3.1%20Programmability%20framework.pdf.

[8] Q. Yin and T. Roscoe. *VF2x: fast, efficient virtual network mapping for real testbed workloads*. In Testbeds and Research Infrastructure. Development of Networks and Communities, pages 271–286. Springer, 2012.

8

Conclusions and future work

The Internet is the most crucial communication medium for people, businesses and organizations. It is significantly challenged by ever-increasing throughput demands, highly expected performance, capacity and forwarding rates. The exponential growth of routing table size limits the scalability of current routing protocols in the Internet. The other major issue is the slow convergence of these protocols.

The main evolutions in the Internet infrastructure include automated control, scaling, resiliency and increased efficiency which are achieved through an improved routing in the Internet and by providing scalable virtualization of Internet services. These are two aspects of future Internet which were explored in this thesis.

In this dissertation, first the topic of resilient scalable routing for the Internet was investigated. Novel routing algorithms and recovery techniques were designed, implemented and evaluated on large-scale networks resembling Internet topology.

As a second aspect of future Internet, this dissertation targeted service orchestration in virtualized telecom networks. Telecom virtualization is a recent topic, reducing the complexity in network design and requirement of proprietary hardware. Relying on virtualization techniques, the goal is to achieve rapid deployment of novel services and enable scalable and flexible service orchestration. In this context, optimized embedding algorithms for virtualized network functions were proposed. Potential architectural enhancements were suggested to enable a scalable orchestration. In the following subsections, we summarize the performed contributions and present the main conclusions and directions for future work.

8.1 Resilient scalable routing in Internet

The main challenges in the Internet inter-domain routing (i.e. BGP) are limited scalability of the routing tables, slow convergence and instability of the routes. BGP extensions and patch solutions have been the subject of intensive research during the last 40 years. In spite of the numerous research efforts, a realistic alternative is still missing. The main reason is that these alternatives follow a similar approach as the one pursued by BGP. Therefore, in order to overcome BGP limitations, clean-slate approaches should be considered.

Geometric routing is an alternative which relies on local information for making forwarding decisions. It is considered as a promising solution to solve the scalability limitations of routing tables. Therefore in this dissertation, the application of geometric routing to inter-domain settings was studied.

Most of the existing geometric routing schemes lack mechanisms to cope with network dynamics efficiently. Since the Internet is dynamic in nature, local recovery techniques for single and multiple link failures were proposed and evaluated in Chapter 2. The proposed techniques can be executed upon failure detection (restoration) or can be activated pro-actively (protection) to enable fast switchover. We relied on a graph embedding into the hyperbolic plane proposed in [1]. These recovery techniques rely on local communication, avoid local minima in the process of greedy forwarding, and scale with the number of edges in the spanning tree of the network. The packet overhead caused by these techniques is very limited. The latter refers to the maximum 4 extra fields in the packet header, in networks with 1000 nodes, compared to the average 24 extra fields in case of using another existing scheme, i.e., [2]. In the proposed schemes, the memory requirement in the nodes is proportional to the nodes degree. This makes the schemes quite scalable. Additionally, the quality of the routing in terms of stretch[1] is significantly improved compared to other alternatives. In the proposed schemes, the average stretch upon 10% tree link failures in large networks with 1000 nodes, did not exceed 1.4. However it increased up to 2.6 in case of using the scheme in [2].

To simplify the process of graph embedding in geometric routing, we proposed a scheme deducing the nodes coordinates from a spanning tree of the network. In order to calculate these coordinates, a distributed algorithm was presented in Chapter 3. This algorithm provides a mechanism to deal with different types of network dynamics such as nodes/links addition/removal. It triggers nodes to re-calculate their coordinate upon a change in the network. The work in Chapter 3 implemented and evaluated this algorithm in a realistic emulation environment. The experimental results indicated that the convergence behavior of geometric routing varied with the location of the changes in the network. The closer these changes

[1] Stretch indicates the ratio between the length of the path generated by geometric routing and shortest path length.

to the root of the tree, the more nodes in the network were impacted. Despite this dependence on the location, the convergence behavior of geometric routing was more desirable than BGP behavior. In geometric routing, the average convergence time upon 10 link failures in networks with 1000 nodes did not exceed 4.5 s, while in BGP it reached up to 26 s. Although the number of affected nodes/paths are potentially high in geometric routing, the convergence is relatively fast. Additionally, the results indicated that the average stretch remained almost unchanged, upon different number of failures in the network. One main outcome of this study is that coordinate re-calculation, upon network failures, can lead to a good performance in terms of stretch with very low convergence time. Next, we extended the algorithm with the recovery mechanism presented in Chapter 2. The experimental results identified the trade-off between convergence time and stretch in geometric routing. The protection mechanism enabled quite fast failure recovery (i.e., 20.4 ms on average, upon a tree link failure) while avoiding coordinate re-calculation. However, it impacted the routing quality in terms of stretch negatively. The average stretch increased from 1.12 to 1.28 after 10% of network changes. This limited loss in performance suggests that a clever combination of protection (particularly in levels close to the root of the tree) and coordinate re-calculation mechanism can lead to a promising performance in terms of convergence and routing quality.

The scalability of the proposed tree-based geometric routing was thoroughly studied in Chapter 4 and compared with the hyperbolic-based approach in [1]. Additionally, an efficient circuit to greedy forward the packets was proposed. The evaluations confirmed that a geometric router can be implemented in a very low-cost way without sacrificing efficiency of the routing. In comparison with the hyperbolic geometric routing, the tree-based approach achieved similar performance in terms of stretch while improving the scalability of the coordinate memory requirement significantly. In large-scale networks of 20000 nodes, the proposed approach required 30 bits for coordinate representation while the hyperbolic-based approach required 100 bits. This study indicates how coordinate complexity is reflected in the complexity of geometric routing and the related forwarding. The more complex the coordinates, the more sophisticated their calculation process and thus the more computation resources are required. Moreover, such coordinates may be quite large in size which in turn leads to large space-complexity for storing, many resources in hardware and also high complexity in the forwarding process.

Finally, we concluded this work with an overview of the existing routing schemes (traditional and clean-slate approaches) and their (in-)suitability to large-scale complex networks. Relying on the experimental results and analytical studies performed so far, the main trade-offs and trends in different schemes were identified. With regard to the proposed geometric routing, both simulation and emulation outcomes confirmed the memory-advantage of this scheme compared to IP-based

routing. In this scheme, the size of the routing table is bounded by the maximum degree of the nodes in the network. The cost of this scalability is in the recovery domain. The latter refers to the potential high number of affected nodes/-paths in case of a failure. One main outcome of this overview is that most of the existing schemes, including geometric routing, increase performance by decreasing functionality. For instance, different alternatives improve the memory scalability but can not support the same level of policy as in BGP. This study better placed geometric routing scheme among other alternatives as potential candidates for large-scale routing. Additionally, the identified remaining challenges provided a guideline for future research which are detailed in the following subsection.

8.1.1 Future directions and trends

There are several open problems which require further investigation. We explain them in 4 different groups.

The first group is related to performance concerns. Since the proposed scheme relies on a spanning tree structure, supporting multipath routing via independent trees is an interesting research direction. Using multiple trees, better fault tolerance and balanced load can be achieved, as was shown in [3, 4]. If the overlap of the constructed trees is minimized, there is a good chance that at least one tree is not impacted upon a failure in the network. As a result greedy forwarding is unaffected and no recovery mechanism needs to be triggered. Regarding load balancing, since there exist multiple trees and thus multiple options as next hop, the neighbor with less load can be selected. This way we can avoid congestion in the network. However as these schemes rely on multiple trees instead of a single one, the challenge is to provide scalable solutions in terms of coordinate memory requirement and also to simplify the forwarding process. The complexity of coordinates, their size and the impact on geometric routing and greedy forwarding were discussed earlier. To reduce this complexity, novel coding schemes should be explored to assign succinct coordinates to network nodes. Another approach is to use compression mechanisms on coordinates. These mechanisms accept plain information (i.e., coordinates) and encode it to use fewer bits. There also exist other mechanisms to accept the encoded form and extract the information. The compressed coordinates can save time (in transmission) and space. The impact of exploiting such mechanisms on computation and memory/storage complexity needs further investigation.

The second group includes business-motivated aspects such as policy, security concerns and migration which are detailed in the following.

One of the main limitations in geometric routing is the lack of support for routing policy including routing engineering, traffic engineering and administrative policies. Such policies affect the performance of the routing. For instance,

traffic engineering enables different QoS guarantees. Exploiting multiple trees in geometric routing provides the freedom to choose between different alternatives as next hop. Therefore, it can be the starting point for supporting policy in this routing.

Another aspect which requires further investigation is security vulnerabilities of geometric routing. Malicious nodes in the network may announce false degrees and may frequently (dis)appear in the network. This leads to frequent coordinate calculation in the network. This becomes problematic if the claimed degree of the malicious node is larger than the degree of the current root. As a result, the node with false degree becomes the new root. Changing the root node leads to coordinate re-calculation of all nodes in the network. Protection mechanisms, used in the levels close to the root node (as suggested in Chapter 3), may provide a temporary solution to avoid global coordinate re-calculations. However, authentication mechanisms should be investigated to avoid such malicious nodes become the root of the tree and trigger such global changes.

Migration from BGP to the proposed geometric routing is another research direction which can be very useful for telecom providers, considering new alternatives to improve the routing performance. In order to avoid the sudden switch to a novel routing system, solutions to enable coexistence of BGP and geometric routing on the same router should be investigated. In case there are network segments capable of geometric routing without any support for conventional IP routing, ingress/egress nodes of these segments should provide packet encapsulation/decapsulation to enable greedy forwarding across them. This means that upon arrival of an IP packet in an ingress node, the coordinate corresponding to the packet destination is added to the header of the packet (encapsulation). Using this coordinate, the packet is then greedy forwarded towards the destination. In the egress node, this coordinate is removed from the packet header (decapsulation) and IP routing can be resumed.

The third group relates to handling negative side-effects of the introduced geometric routing. In this routing since the role of locator and identifier is separated (as opposed to IP addresses), a mapping system to bind node identifiers to node coordinates is required. Such a mapping system should be scalable and enable fast convergence upon changes in the network nodes coordinates. In spite of the existence of several proposals in the literature, a scalable secure and highly reliable mapping which provides fast convergence is still missing [5].

Finally, if the business-motivated aspects and negative side-effects of geometric routing cannot be solved, it is an interesting research to investigate application of other routing schemes, proposed for networks with characteristics far from the Internet, in large-scale inter-domain settings. Examples include routing schemes proposed for Delay Tolerant Networks (DTN) or opportunistic networks. These networks are known for their frequent disruption, sparse network density and high

bit error rate which are very different from the characteristics of inter-domain network in the Internet. Similar to these networks wireless mobile ad-hoc networks are also considered infrastructureless and dynamic in nature. Accordingly, the routing schemes proposed for these networks target adaptivity to such environments. However they may show interesting/promising behavior once used in other settings. This was the case with geometric routing, as it was initially proposed for ad-hoc and Wireless Sensor Networks (WSNs). Stochastic routing is one example which can be a potential alternative, initially proposed for wireless mobile ad-hoc networks. This routing relies on a probability distribution to select the next hop of an incoming packet. This distribution can be affected by different metrics. Therefore, it is possible to influence this routing by metrics such as load, residual energy, forwarding cost, etc. Another example is routing schemes proposed for opportunistic mobile networks, exploiting human social characteristics such as social interaction patterns, mobility patterns and routines. Based on these patterns/characteristics, efficient routing models are proposed. Applying similar approaches in the Internet could be quite interesting. The studies performed so far indicate that over a period of time, a significant part of the nodes are never used as destinations. Traffic mainly flows to the most popular networks and only 15% of the nodes are responsible for 95% of the entire traffic [6][2]. These numbers suggest that recognizing traffic patterns, influenced by human behavior and daily routines, and exploiting them may lead to efficient routing models.

With the discussed remaining challenges, it is still early to introduce geometric routing as a BGP replacement. However, the documented research illustrated the applicability of this routing in the large-scale inter-domain setting of the Internet. The memory-advantage, good performance in terms of stretch and interesting convergence behavior made this routing a potential alternative for future Internet routing. Moreover, the design of the circuit for greedy forwarding was one step closer to exploit this routing in practice. However, assuming that solutions to the open problems such as support of routing policy, scalable mapping system and security concerns are proposed in the next 5 years, the process of migration from BGP to geometric routing is extremely difficult to estimate. This was confirmed in transition from IPv4 to IPv6. Although it is approximately 20 years since the introduction of IPv6, a complete transition to this addressing format is still not feasible. In this context, the Locator/Identifier Separation Protocol (LISP)[3] incremental deployment scenarios can be exploited [7]. LISP routing architecture separates the device identity from its location into two different numbering space. It is a Cisco innovation and different standardization activities are ongoing at the IETF LISP Working Group to develop this architecture. There are several LISP sites deployed

[2]The study has been on GEANT network which is a European academic network and may not be representative for the entire Internet backbone. However it provides useful perception.

[3]http://www.cisco.com/c/en/us/products/ios-nx-os-software/locator-id-separation-protocol-lisp/index.html

around the world, enabling gradual introduction of LISP into the existing IP network. It also provides efficient strategies for IPv6 transition[4]. Similarly LISP can be exploited to gradually deploy geometric routing. Since LISP allows using different address families for the identities and the locators, it is possible to use the coordinates in geometric routing as locators and perform greedy forwarding within LISP sites. This is a promising strategy for transition from BGP in a short time frame.

8.2 Service orchestration in virtualized telecom networks

The rise of network virtualization and software defined networking have introduced new challenges in service orchestration in telecom networks. In spite of the flexibility in service provisioning, enabled by network function virtualization, the freedom where to provision each network function causes new challenges and complexities. This complexity is exacerbated with the fact that a large monolithic network function block may be decomposed into smaller functional blocks or can be realized in multiple ways through different implementations.

Existing works consider that services are large monolithic blocks and thus provide solutions with very limited flexibility, not being able to cope with dynamic scaling demands efficiently. Therefore in Chapter 6, we investigated the joint optimization of service decomposition[5] and service chain embedding. First we proposed an approach to find the exact solution. However since the complexity of this scheme was quite high, a heuristic approach was proposed to solve the scalability limitation of the optimal solution. The superiority of the joint decomposition and embedding was confirmed by higher acceptance ratio and lower embedding cost compared to the approach in which decompositions were selected independent of their resource demands and available resources in the infrastructure network. Taking service decompositions into account at the time of embedding enables better adaptivity to network conditions. This means that a decomposition which better matches the network state (e.g. with regard to available resources) can be selected. This results in an efficient utilization of network resources leading to higher service acceptance rate. In a network with 110 nodes, the proposed approach resulted in acceptance of approximately 90% of service requests while the scenario in which decompositions were selected randomly led to 40% acceptance over time.

Scalability of service orchestration is a significant characteristic, since the number of components in a realistic telecom network infrastructure quickly ex-

[4]http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/locator-id-separation-protocol-lisp/white_paper_c11-629044.pdf

[5]The process of decomposing the high-level complex network functions in a service chain into more refined atomic function blocks.

ceeds ten thousands. Having such a large number of components, thousands of service requests in one day and the possibility of realizing each service in multiple ways intensify the need for a scalable service orchestration architecture. This was studied in Chapter 7 and we identified the major blocks in orchestration time in a proof of concept prototype. This prototype relied on the embedding scheme explained in Chapter 6. The three identified blocks were: i) retrieval of all decompositions for a given service from a database (read dcmp), ii) selection of a desirable decomposition (select dcmp) and iii) embedding/mapping of the selected decomposition. The experimental results indicated that in small networks 'read dcmp' is the most time consuming block while in large networks (more than 1000 nodes) 'select dcmp' is the dominant block. Based on this outcome, we proposed several architectural enhancements to move towards a more scalable service orchestrator. These are interesting future research directions and are detailed in the following subsection.

8.2.1 Future directions and trends

While the proposed approach provides joint optimization of the service decomposition and embedding, it mainly focuses on mapping cost of the network functions. Therefore, there are several ways to extend this approach which are discussed below.

Depending on the user's or provider's requirements different objective functions for the embedding problem can be defined. For instance balancing the load in the network can be considered instead of mapping cost.

In the proposed approach in order to find a suitable service decomposition, a cost function based on several metrics was defined. These metrics were selected according to the objective function considered in the embedding problem. If other objective functions are used, different metrics which are more useful for achieving the goal should be looked for.

Next, in addition to the QoS parameters such as bandwidth and delay, other constraints can be taken into account. Location of a network function may be critical depending on its role. For instance network functions related to security (encryption/decryption, etc) should be placed very close (if not at the same place) to the end users. Taking these additional constraints into account increases the complexity of the embedding problem. Therefore, scalable solutions should be investigated.

The next set of future directions focuses on the scalability of the service orchestration framework.

Our studies indicated that certain blocks of the embedding, as the main function of the service orchestration, do not scale as desired. A solution is to change the embedding algorithm to a distributed approach in which costly calculations are

given to different computation resources. This way, these tasks can be executed in parallel. Although certain blocks in the embedding process are quite straight forward to parallelize, parallelizing the mapping functionality is quite challenging. Solutions should be found to avoid that the same resource is reserved for different network functions at the same time.

Another way to make an embedding approach more scalable is to have a hierarchical embedding process. Assuming that the infrastructure is divided into several domains, the given service chain can be split into sub-chains accordingly. Based on a hierarchical embedding approach, first the service sub-chains are mapped to different domains. Then the local orchestrator at each domain performs the next level of embedding. The main challenge in this case is related to the amount of resources and infrastructure information which is announced by each domain to the upper layer orchestrator. The (in)completeness of this information can significantly impact the performance and efficiency of the embedding process. An interesting research direction is to identify the trade-offs between the efficiency of the embedding and the amount of exposed information about the infrastructure.

The scalability limitation of an orchestration process is partially caused by the multiple realization options enabled by service decomposition. This limitation is exacerbated by the fact that decompositions for each network function are stored individually in the database. Upon arrival of a service request composed of several high-level network functions, different combinations of the network function decompositions should be constructed. The process of retrieving/constructing all possible combination of the decomposed network functions in the requested service is time consuming (as discussed in Chapter 7). To diminish this scalability issue, an option is to pre-define service decompositions. This avoids constructing all possible combination of network function decompositions. Also some parts of the embedding (mainly the parts related to the decomposition selection) can be performed proactively, before the arrival of the service request. This way the required time for embedding a service request upon its arrival is reduced. However such an approach may negatively impact the flexibility and dynamicity of service chaining. Moreover, with pre-defined service decompositions, less adaptivity to network conditions is possible. Therefore, a future research direction could be to identify the potential trade-offs.

Telecom service virtualization and -chaining relying on NFV are rapidly getting traction. The large number of ongoing research projects focusing on these topics, and the involvement of different telecom operators in these projects underline the immediate value of NFV-based solutions. The close and active collaboration of the telecom industry in the context of standardization bodies such as the NFV industry group of ETSI or the Software-Defined Networking Research Group (SDNRG) and Network Function Virtualization Research Group (NFVRG) at IETF/IRTF is progressing quickly in formulating solution architectures, refining

research challenges and potential solutions for NF orchestration and service chaining. Therefore, scalable solutions, as the ones documented in this research, have the potential to be instantly exploited in the process of service provisioning.

References

[1] R. Kleinberg. *Geographic routing using hyperbolic space*. In INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, pages 1902–1909, 2007.

[2] A. Cvetkovski and M. Crovella. *Hyperbolic embedding and routing for dynamic graphs*. In INFOCOM 2009, IEEE, pages 1647–1655, 2009.

[3] R. Houthooft, S. Sahhaf, W. Tavernier, F. De Turck, D. Colle, and M. Pickavet. *Fault-tolerant Greedy Forest Routing for Complex Networks*. In Proceedings of 6th International Workshop on Reliable Networks Design and Modeling (RNDM), 2014.

[4] R. Houthooft, S. Sahhaf, W. Tavernier, F. De Turck, D. Colle, and M. Pickavet. *Robust Geometric Forest Routing with Tunable Load Balancing*. In Proceedings of INFOCOM 2015, 2015.

[5] M. Hoefling, M. Menth, and M. Hartmann. *A survey of mapping systems for locator/identifier split Internet routing*. IEEE Communications Surveys & Tutorials, 15(4):1842–1858, 2013.

[6] J. Mikians, A. Dhamdhere, C. Dovrolis, P. Barlet-Ros, and J. Solé-Pareta. *Towards a statistical characterization of the interdomain traffic matrix*. In International Conference on Research in Networking, pages 111–123. Springer, 2012.

[7] F. Coras, L. Jakab, D. Lewis, A. Cabellos-Aparicio, and J. Domingo-Pascual. *Locator/Identifier Separation Protocol (LISP) Network Element Deployment Considerations*. 2014.

A

Availability analysis of resilient geometric routing on Internet topology

In this appendix, we evaluate the availability of the proposed tree-based geometric routing scheme, relying on the protection mechanism of Chapter 2 for link failure recovery.

⋆ ⋆ ⋆

S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, P. Demeester

Presented at 10th International Conference on the Design of Reliable Communication Networks (DRCN), 2014.

Abstract Scalable routing schemes for large-scale networks, especially future Internet, are required. Geometric routing scheme is a promising candidate to solve the scalability issue of routing tables in conventional IP routing based on longest prefix matching. In this scheme, network nodes are assigned virtual coordinates and packets are forwarded towards their intended destination following a distance-decreasing policy. Dynamics in the network such as node/link failures might affect this forwarding and lead packets to a dead end. We proposed recovery techniques in geometric routing to deliver packets to the destination in case of failures. In this paper, we perform an analysis on the availability of the proposed protection techniques on the Internet graph. The routing scheme over optical transport network is

considered and the reliability data of physical components and a known network availability model are used. This evaluation is compared with the shortest cycle scheme which finds two node disjoint paths between every source and destination in the topology and also with geometric routing with no protection. The results show that the proposed scheme performs reasonably well compared to the shortest cycle scheme and significantly enhances the availability compared to geometric routing without any protection.

A.1 Introduction

Due to ever increasing throughput demands, IP prefix lookup is becoming a bottleneck in future Internet. Higher performance, capacity and forwarding rates are required in order to meet the future demands. The growth of the size of the routing tables is another issue which limits the scalability of the current IP-based routing protocols. BGP is the protocol used to exchange routing and reachability information between Autonomous Systems (AS) in the Internet ([1] reports more than 400K FIB entries in current BGP routers). Geometric routing has been proposed to solve the issue of scalability in routing tables and can be considered as an alternative to BGP to route between ASes (Inter-AS).

In geometric routing, nodes are assigned virtual coordinates and the forwarding is based on a distance-decreasing policy [2]. This means that upon arrival of a packet in a node, the distance of every neighboring node to the destination of the packet is calculated and the one which decreases this distance the most is selected as the next hop. Repeatedly applying this policy will lead the packets to their intended destination. As in every step, the neighbor with the most decreasing distance is selected, the routing/forwarding is referred as greedy.

A problem with greedy forwarding is that packets might reach a local minimum (dead end, void). This means that the current node is closer to the destination than any of its neighbors. Greedy embeddings are proposed to solve this issue [3]. Greedy embeddings map network nodes to coordinates in such a way that for every node there is always a distance decreasing neighbor towards any other node in the network. Greedy routing based on these embeddings guarantees the delivery of the packets to every destination. In [4], we proposed a simple but promising greedy embedding based on a spanning tree of the network. In this scheme, every node is assigned a label (coordinate) indicating the path from the root of the tree to the node.

As the embedding is derived from a spanning tree of the network, a change in the connectivity of the tree (component failures) might affect the embedding and might lead the packets to local minima. Therefore, recovery techniques are required to provide resiliency against failures in the network. In [4] and [5], we proposed protection techniques in geometric routing for link/node failures in the

network. These works showed the scalability of the proposed schemes in terms of resources. As availability is an important performance assessment factor for recovery schemes, the goal of this paper is to evaluate the connection availability of the proposed single-failure resilient geometric routing on a large graph such as the AS-level topology of the existing Internet (CAIDA [6]). In this evaluation, geometric routing over optical transport network is considered and the reliability characteristics of physical components and a well-known availability model are used. In order to have a realistic evaluation, a multi-layer model is used in which recovery in different layers is considered. Therefore, we evaluate the connection availability of geometric routing as a network layer scheme with the assumption that links are protected in lower layers. Having a fair comparison with BGP is challenging because different factors such as routing policies and AS business relationships affect the selection of the paths. Therefore, we compare the availability of the proposed scheme with shortest cycle scheme in which two node disjoint paths between every source and destination in the network are constructed. In this scheme, in case the working path between two nodes is not available, the second node disjoint shortest path is used as the protection path. To the best of our knowledge, this is one of the first studies related to connection availability of geometric routing on the Internet topology.

This evaluation gives an overview on: i) how well the proposed recovery scheme performs compared to the node disjoint shortest path alternative and ii) what is the maximum and minimum connection availability that can be obtained using the model proposed on the Internet topology. These studies are essential especially for services with high availability demands.

The rest of the paper is organized as follows. In the next section, work related to availability analysis in network, geometric routing and recovery techniques are described. A short description of the used greedy embedding and the protection techniques is provided in Section A.3. Section A.4 explains basic concepts related to reliability assessment. Section A.5 presents the evaluation of connection availability for different schemes of single-failure resilient geometric routing, shortest cycle and geometric routing with no protection. Future work is discussed in Section A.6 and finally Section A.7 concludes the paper.

A.2 Related work

Availability analysis especially in wavelength division multiplexing (WDM) optical networks has received great attention in recent years. Many works have proposed analytical models for evaluating the connection availability in these networks for single link failure scenarios [7], [8]. The main observation in [7] was that a mesh network with single link failure restorability is robust under dual-failure events. The authors defined *equivalent unavailability* for a link which means that

the link is considered available if it is physically working or it is physically down but transparently replaced by a restoration path between its end nodes. This is similar to the multi-layer model we consider in the connection availability evaluation in this paper. While evaluating the availability of routing scheme in network layer, we assume that links are protected in lower layers. In [8], different protection schemes, 1:1, 1:N and M:N are considered. The focus of the paper is on path protection strategy in which a backup path for each working path on an end-to-end basis is provided. This is different from our work as nodes are recovered locally and failing links are bypassed by routing towards a hub node. In [9], the reliability analysis on two link-disjoint paths in mesh network is studied. Using two disjoint paths, 100% protection against a single failure is provided and authors try to maximize the reliability regardless of the number of link failures occurring on the network. It is proved that this problem is NP-complete and heuristic algorithms are proposed to find two link-disjoint paths with maximum reliability. In this paper, we compare the connection availability of the proposed schemes for geometric routing with such an scheme in which two disjoint paths for every source and destination in the topology are calculated.

In other works such as [10] analytical model for dual link failure is proposed. In [11], authors proposed a hybrid scheme to achieve high connection availability with low backup resources in double link failure scenario. The hybrid scheme is based on backup re-provisioning, path restoration and 1:1 dedicated path protection. They use ILP models in their proposed scheme. In [12], a model for availability evaluation of protected optical connections in WDM networks employing M:N dedicated protection is proposed. The same authors proposed an analytical model for availability evaluation of WDM network with shared-link connections under multiple link failures in [13]. The effect of topology properties on connection availability in Generalized Multiprotocol Label Switching (GMPLS) over optical transport networks is studied in [14].

As geometric routing scheme suffers from packets reaching local minima, many works have proposed greedy embeddings in different metric spaces to avoid this issue. In [15] and [16], authors proposed greedy embeddings in the hyperbolic plane. Flury et al. proposed greedy embedding in Euclidean space [17]. We proposed a greedy embedding based on a spanning tree of the network in [4]. Dynamics in the topology such as link/node failure might affect the greedy embeddings causing the packets reaching local minima. Face routing techniques pass the void by routing around this area and greedy routing is resumed from the moment that a closer node to the destination is reached [18]. The issue with these type of techniques is that the graph should be planar/planarized which might not be feasible for every graph. Authors in [19] proposed a dynamic greedy embedding to deal with node additions to the network topology. However, in order to deal with network failures a path trace is maintained in packets in order to avoid loops which

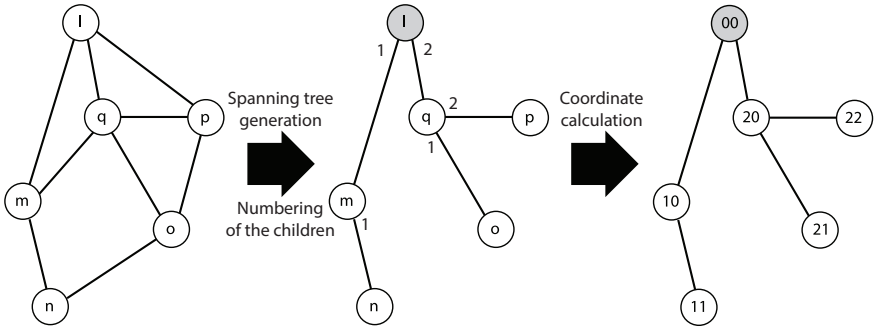


Figure A.1: Steps for tree-based greedy embedding.

imposes a large overhead to packets headers [4]. We proposed different protection techniques for link/node failures which avoid coordinate re-calculations and are based on greedy forwarding [4], [5]. The proposed schemes are scalable with low overhead to the network nodes and packets headers.

In this paper, we consider the tree-based greedy embedding as proposed in [4] and evaluate connection availability of geometric routing with the proposed protection schemes in [4] and [5] over optical transport network in the Internet topology.

A.3 Greedy geometric routing

In this section, we briefly explain the tree-based greedy embedding and the protection techniques on which the availability analysis is performed. For more details on the schemes, we refer the interested readers to [4] and [5].

A.3.1 Tree-based greedy embedding

Rather than relying on complex (e.g. hyperbolic) geometry, in [4] we used tree-based coordinates for greedy forwarding. This will reduce the computational overhead and complexity of the overall scheme. Steps for calculating the labeling based on a spanning tree of the network are as follows:

1. First a rooted spanning tree of the network is generated.
2. The root node sets its coordinate vector (CV) to zero.
3. Each node numbers its children from 1 to d (d is the number of children).
4. A node can calculate the coordinates of its children by adding the number assigned to each child after the last non-zero coordinate in its own CV.

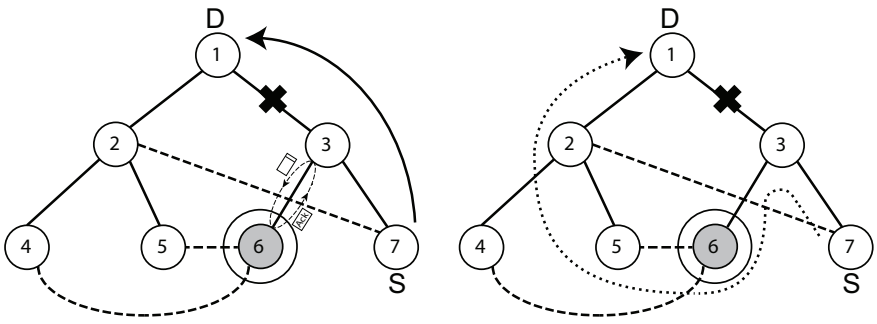


Figure A.2: Example for upward failure scenario.

Figure A.1 depicts an example for this embedding. Once every node has its deduced CV using the above procedure, packets can be forwarded towards neighboring nodes which (maximally) reduce the distance towards the CV of the destination node mentioned in the received packet. In this context, tree-distance is used as the metric which is the hop count on the tree between two nodes.

A.3.2 Single failure protection techniques

In [5], we proposed protection techniques for geometric routing which are only used for tree edge failures. The reason is that in greedy routing based on a spanning tree, in case of a shortcut[1] failure, there is still a distance-decreasing path available via the tree. However, a tree edge failure might affect the greedy embedding causing the packets to reach a local minimum.

In this scheme, we distinguish between upward and downward failures. In upward failures, packets should have passed the failing link to get closer to the root of the tree while in downward failures, packets go deeper in the tree.

For every tree edge attached to a node, a search is performed in order to find an intermediate node (hub node) from which greedy routing to the destination is possible (considering that edge to be failing). In a downward failure scenario, we look for a hub node which has a shortcut to the subtree below the failing link. In an upward failure scenario, it is enough that the hub node has a shortcut to a node out of the subtree below the failing link. The forwarding process in case of a failure is as follows:

1. Greedy route from failure detecting node to hub node.
2. Take the shortcut to go to the desired subtree.
3. Continue greedy routing to the destination on the tree.

[1] A link which is not in the spanning tree of the network.

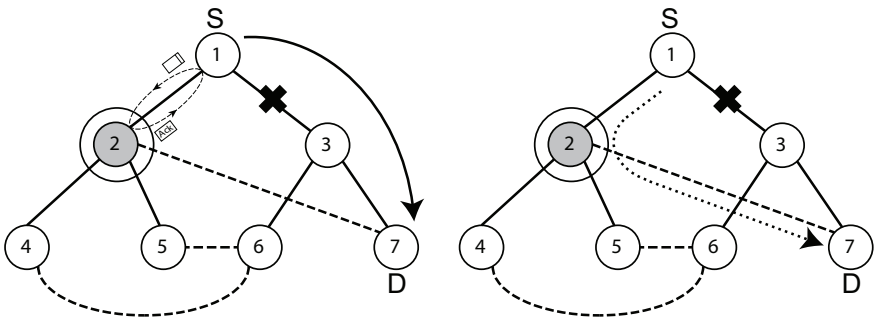


Figure A.3: Example for downward failure scenario.

Figures A.2 and A.3 depict examples for the two explained failure scenarios. The primary greedy path is depicted with solid line on the left and the secondary path after recovery is depicted with dot line on the right. The corresponding hub node for the depicted failing tree edge is colored grey.

We proposed a protection technique for node failure in [4]. In this scheme, every node finds disjoint backup paths to the nodes two hops away. In case of a node failure, a disjoint backup path is used to bypass the failing node and greedy routing is resumed once the two-hops away node is reached.

We refer the interested readers to [4] and [5] for more details on the exchanged messages, augmentation to the packet headers and the overhead added to the network nodes.

A.4 Performance parameters

In this section, we explain the reliability performance parameters which are used for the evaluation of the protection techniques for single failure in geometric routing.

A.4.1 Component availability

Availability of a component is defined as the probability that the component is functional at any arbitrary moment. It can be expressed by the components mean time to repair (MTTR) and the mean time between failures (MTBF). MTTR is defined as the required time for the restoration of the component and MTBF is the time between two consecutive failures of the component [20]. The availability A is:

$$A = 1 - \frac{MTTR}{MTBF} \tag{A.1}$$

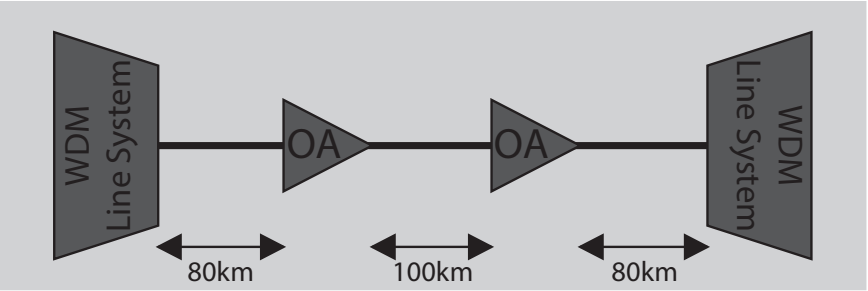


Figure A.4: Example of a bidirectional line [20].

A.4.2 Protected/Unprotected connection availability

The availability of a connection in a network is calculated using the availability of the components (nodes/links) in the network. In an unprotected connection, all the nodes and links/lines along the path of the connection should be available in order to have availability for the connection. Therefore, the availability is defined as:

$$A_{unprotected} = \prod A_i \tag{A.2}$$

A_i is the availability of the ith component (node or link) along the path.

As mentioned earlier, recovery schemes enhance the availability of the connection. A protected connection is available if the working path or the protection path of the connection is available.

$$A_{protected} = A_w + A_w^{'} A_p \tag{A.3}$$

A_w and $A_w^{'}$ are the availability and unavailability of the working path respectively. A_p is the availability of the protection path. This formula can be extended for the availability of a system, A_{system}, with more than one connection and considering single failure:

$$A_{system} = \prod_{i=1}^{M} A_{w_i} + \sum_{j=1}^{M} \left(\prod_{k=1;k\neq j}^{M} A_{w_k} \right) A_{w_j}^{'} A_{p_j} \tag{A.4}$$

M is the number of working connections in the system.

A.4.3 Network availability model

In the network availability model considered in this paper, a node is composed of one Optical cross connect (OXC). A bidirectional line which connects two optical nodes is composed of a series of components such as pieces of physical cable, a number of optical amplifiers (OA) and a line system at each side of the line (Figure

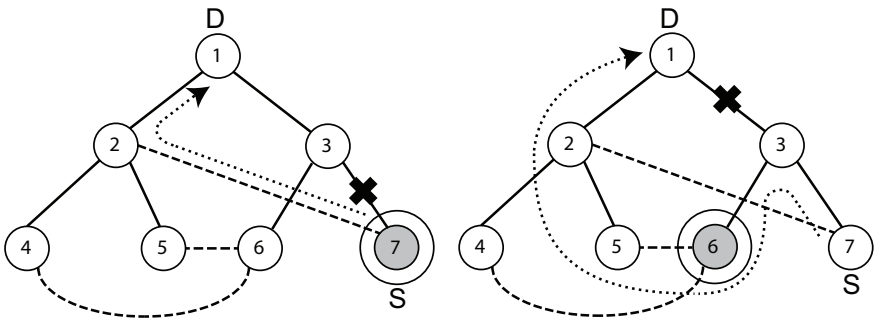


Figure A.5: Example for evaluation of connection availability.

A.4). This line is available only if all the components are available. Assuming statistically independent failures, the availability of a bidirectional line is defined as:

$$A_{line} = A_{cable} \times A_{OA}^{N} \times A_{line-system}^{2} \tag{A.5}$$

N is the number of OAs which is dependent on the length of the line (assuming an OA every 100 Km) [20].

The MTTR and MTBF of physical components reported in [20] are used in formula (A.1) to obtain the availability of single components in the network.

We evaluate the connection availability of geometric routing over optical transport network for every possible source-destination pair in the topology based on the reliability characteristics of the physical components and the connection model in formula (A.3). In order to obtain a more realistic evaluation, we consider a multi-layer model. This means that although the proposed protection scheme is provided for network layer (L3), in the availability evaluation of the connections, links in the topology are considered to be protected in lower layers (L2/L1).

Note that in the proposed protection scheme, in a connection, depending on the link/node which is failing, different backup paths might be used. Therefore, in order to evaluate the availability of a connection, every possible single failure along the connection path is considered and the availability of the corresponding backup path is evaluated and the sum of all represents the protected connection availability. Consider Figure A.5 as a case study. In this example, only link failure is considered. Each of the graphs in this figure depicts a single link failure scenario along the connection 7-3-1 and the corresponding protection path is illustrated by a dot arrow. The availability of the connection 7-1 is calculated as follows:

$$\begin{aligned} A_{connection7-1} = {} & A_{7-3}A_{3-1} + A_{7-3}^{'}A_{7-2}A_{2-1} + \\ & A_{3-1}^{'}A_{7-3}A_{3-6}A_{6-5}A_{5-2}A_{2-1} \end{aligned} \tag{A.6}$$

A.5 Reliability performance evaluation of connections

In this section, we evaluate the connection availability of the Internet topology for different schemes: i) tree-based geometric routing with no protection and ii) tree-based geometric routing with single link/node failure protection technique. In all schemes, we assume that links are protected in lower layers (L2/L1). Having a fair comparison with BGP is a challenging task because in BGP different factors such as AS relationships and routing policies affect the path selection. Therefore, we compare the schemes with another alternative in which the shortest cycle between every source and destination is calculated. In this scheme, in case of the unavailability of the working path, the second node disjoint shortest path is used as the protection path.

First, we explain our simulation methodology and how we modeled the Internet topology and then we report the numerical results.

A.5.1 Methodology for simulation approach

The general steps in our simulation are as follows:

1. Given a topology, the tree-based coordinates as explained in Section A.3 are assigned to the network nodes.

2. The hub node corresponding to every tree edge and the disjoint backup paths for nodes are calculated as explained in Section A.3.

For every possible connection in the network (every possible source-destination pair):

1. The path produced by greedy geometric routing is determined.

2. The availability of the unprotected connection is calculated based on formula (A.2).

3. Components (link/node) along the path are considered failing, one at a time, the protecting path corresponding to that failure is calculated based on the proposed protection technique and the availability of the connection is calculated as explained in Section A.4.

In the availability evaluation of the scheme based on shortest cycle, as for any failure in the working path, the same backup path is used, we directly apply formula (A.3).

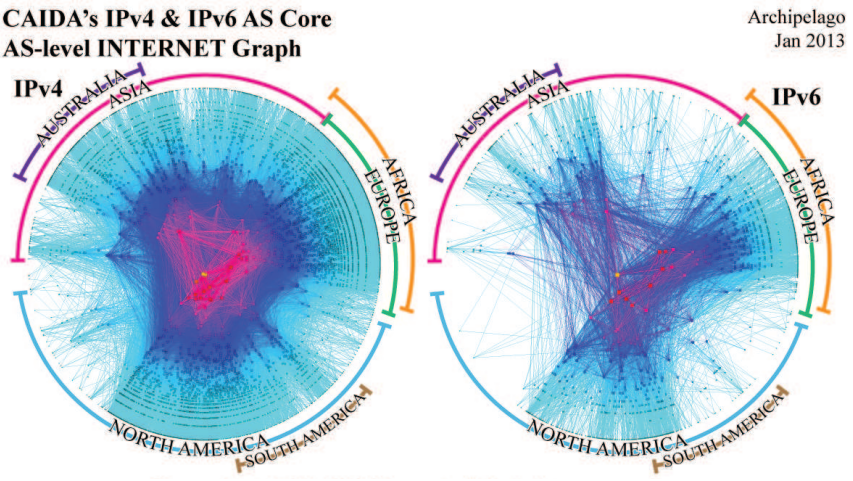


Figure A.6: CAIDA's AS-level Internet graph. http://www.caida.org

A.5.2 Internet topology

As mentioned earlier, geometric routing is considered as an alternative to BGP to route between ASes in the Internet (inter-AS routing). Therefore, we perform our availability analysis on an AS-level graph of CAIDA which represents the topology of the Internet at the level of ASes. Figure A.6 visualizes CAIDA's AS-level Internet graph in January 2013. Due to resource limitation in the simulation environment, we select the CAIDA network which consists out of 16301 ASes and 27646 links [6]. For this network, the average shortest path is 3.77, the minimum node degree is 1 and the maximum node degree is 2331.

One of the major challenges in this analysis is to achieve a network model based on the existing data for the Internet topology. In the following, we explain how these challenges are tackled and what assumptions are made.

In CAIDA AS-level graph, nodes represent ASes and links represent peering relationships. No more data regarding the geolocations and the distances between different ASes is provided. In order to evaluate the connection availability on the CAIDA topology, we require some estimation on the physical distance between adjacent nodes in the network (required for line availability evaluation in formula (A.5)). To this end, we used two different data bases[2]: i) GeoLite ASN which provides the IP addresses to AS numbers mapping and ii) GeoLite City which provides IP addresses to geolocations mapping. Using these databases, we extract the approximate location of each node in CAIDA topology. Assuming that each AS consists of one border router (due to lack of more information), the extracted

[2]http://dev.maxmind.com/geoip/legacy/geolite/

locations represent the geolocation of such routers. The border router of an AS is assumed to be directly connected to the border routers of adjacent ASes. Therefore, the physical distance between connected routers is calculated based on their geolocations. These border routers perform geometric routing to route between ASes.

The first challenge was that the databases could not provide geographic location for almost 10% of the nodes in the network. Therefore, in order to estimate the location of them we used the locations of the adjacent nodes. Based on the locations of the neighboring nodes, the geolocation of the center point is calculated and is assigned to the node without any location. This way, we could estimate the locations of the majority of the nodes with missing locations.

The next challenge was that this process could not be used for the leaf nodes (nodes with only one neighbor). Therefore, in such scenarios we used another process to complete our model.

Consider a leaf node with missing location which has a parent with known location. First we calculate the surface that can be covered by the parent node. This can be calculated based on the locations of other adjacent nodes to the parent. We only consider adjacent nodes with the distance between 50 and 1000 Km. We assume that nodes within the range of 50 Km cover the same area and above 1000 Km are probably not connected directly. Having this in mind, the radius of the surface that the parent node can cover is calculated as half of the average distance of the considered neighbors. If we assume that the leaf nodes are distributed uniformly in the covered area by the parent, half of the calculated radius can be considered as the upper bound for the distance of a leaf node to the parent.

Note that in practice, ASes might be connected to each other through Internet Exchange Points (IXP), however, in an AS-level topology such as CAIDA, these ASes are considered to be adjacent (an edge in the graph). This is a limitation of the model, as it considers a direct physical link between adjacent nodes in the AS-level topology.

Figure A.7 depicts the distribution of the distances calculated for the adjacent nodes in the CAIDA network. As we see, a large percentage of distances are in the range of 10 to 5000 Km. Due to the incompleteness of the databases and the limitation of the model, a few distances reach $20K$ Km. The average of the calculated distances is 2461.23 Km.

Using the calculated distances and formula (A.5) in Section A.4, we evaluated the availability of the links/lines in the CAIDA network and Figure A.8 depicts the distribution of the calculated values. The percentile and the average value of these availabilities are reported in Table A.3. The MTTR and MTBF of different optical components are based on the values proposed in [20] and are reported in Tables A.1 and A.2.

Considering the availability model of a bidirectional line (formula (A.5)), fiber

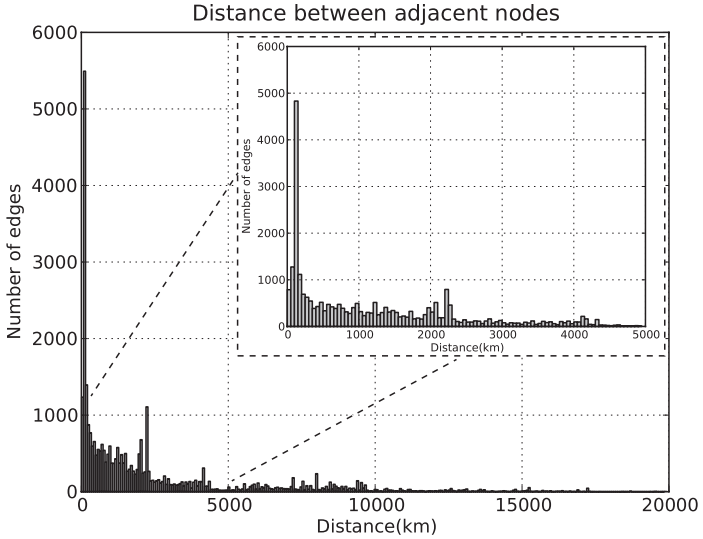


Figure A.7: Histogram of the distances of the adjacent nodes in the CAIDA topology. For better visualization, the inset illustrates the histogram of the range 0-5000 Km.

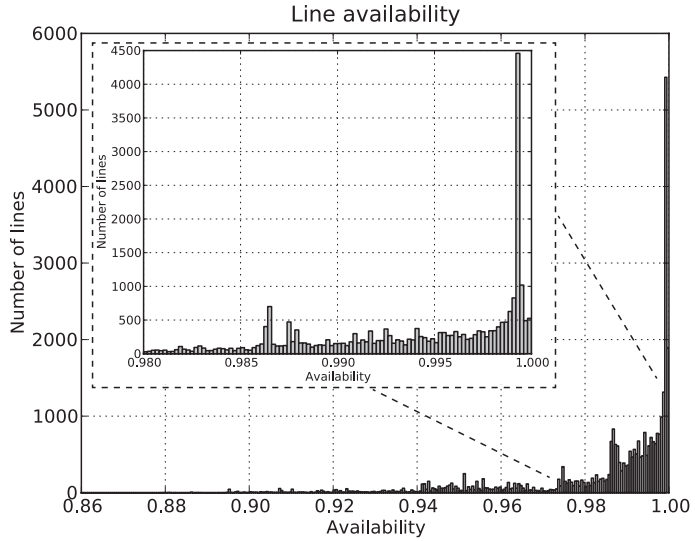


Figure A.8: Histogram of the line/link availability in the CAIDA topology. The inset illustrates the histogram of the range 0.98-1.

Table A.1: MTTR and MTBF values for OAs and WDM Line systems.

	MTBF(hours)	MTTR(hours)
Bidirectional OA	5×10^5	24
Bidirectional WDM Line system	5×10^5	6

Table A.2: Cable Cut and MTBF values for Fiberoptic Cable.

	Cable Cut-CC (km)	MTTR(hours)
Terrestrial Fiberoptic Cable	450	24

optic cable is the dominant component in this model because the cable cuts are frequent and repair times are very long. Therefore, the availability of a line is dependent on its length. Observing the calculated line availabilities, the low minimum availability of 0.8787 can be explained because of the large distances (20K Km) between some nodes. Based on these results, up to 75 percentile of the line availabilities are below 0.999.

Due to these large distances (low line availability) in order to have a realistic evaluation, the multi-layer model is considered in which links are assumed to be protected in lower layers (L2/L1). The percentile and the average value for availability of the protected lines are reported in Table A.3. We observe a significant enhancement in the availability of the lines compared to the values with no protection. The minimum line availability is increased to 0.9852 and even the 25 percentile of the values is above 0.999.

A.5.3 Numerical results

Using the calculated line availabilities (with protection in lower layers L2/L1), we evaluate the connections availability using geometric routing in the CAIDA network based on formula (A.3) and the explanations in Section A.4. The node availability is assumed to be 0.99994 which was proposed in [20].

Note that in CAIDA topology, up to 30% of the nodes are leaf nodes. As these nodes/attached links can not be protected by the proposed protection scheme, their

Table A.3: Percentile and average values for line availability (A) in the CAIDA topology and the availability of the protected links (A-P). (A-L) represents the availability for links attached to the leaf nodes.

	Min.	25	50	75	Max.	Average
A	0.8787	0.9820	0.9925	0.9986	0.99999	0.985
A-P	0.9852	0.99967	0.99994	0.99999	0.99999	0.9993
A-L	0.9852	0.99986	0.99999	0.99999	0.99999	0.99965

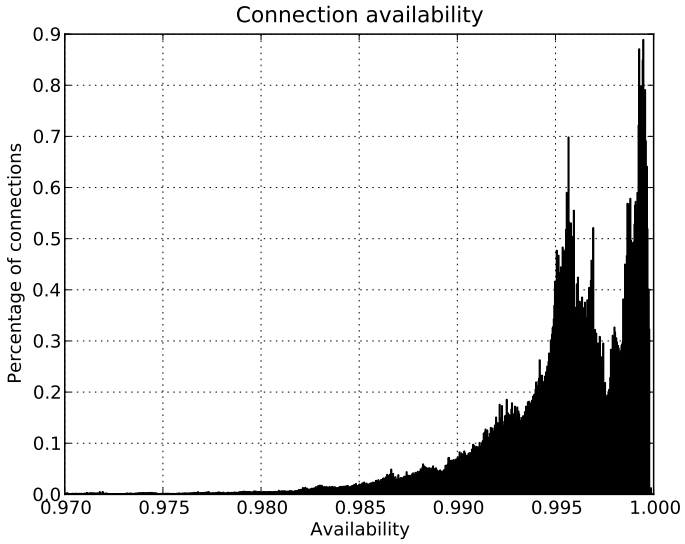


Figure A.9: Histogram of the connection availability of geometric routing on CAIDA topology without protection.

availability is limited to the obtained line availability with protection in lower layers (L1/L2). Table A.3 reports the percentile of the availability of the edges attached to the leaf nodes. As the protection schemes do not enhance the availability of these edges, we report the availability of the connections excluding the leaves.

We first evaluate the connection availability without any recovery scheme and in the second experiment the protection scheme is considered and the explained simulation methodology is followed.

Figure A.9 illustrates the distribution of the connections availability based on geometric routing in CAIDA network when no recovery is considered (for all possible source-destination pairs). For better visualization, the availability in the range 0.97-0.99985 is depicted however, the percentile and average values are reported in Table A.4. As mentioned earlier, length of a link is the dominant factor in the availability of the link and the availability of a connection is also dependent on the length of the links. Therefore, availability of the connections including the very long links can be negatively affected. The low minimum value of 0.94202 can be explained because of the estimation of very large distances for some adjacent nodes in the network.

Figure A.10 depicts the distribution of the connections availability with protection technique in geometric routing. We observe how the availability is enhanced compared to the previous scheme, when no recovery was used. The availability in the range 0.99980-0.99988 is depicted and the percentile and the average value are reported in Table A.4. The minimum availability is increased to 0.98 and the aver-

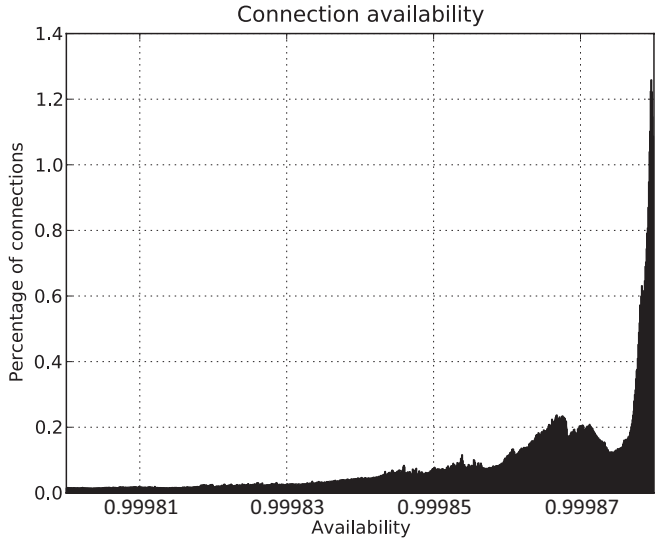


Figure A.10: Histogram of the connection availability of geometric routing on CAIDA topology with protection.

age value is 0.9998 which is a significant enhancement compared to no protection scheme.

In the final scheme, as a comparison, we evaluate the connection availability in an alternative scheme in which the shortest cycle between every two nodes is calculated. Finding the shortest cycle between two nodes in the topology boils down to finding minimum-cost flow of 2 between the nodes. Once the working path is unavailable, the second node-disjoint shortest path is used as the protection path. Availability of a connection is also dependent on the number of hop counts in that connection. The comparison with the shortest cycle scheme gives an overview on how the connection availability is affected by the increase in the number of hop counts. Figure A.11 illustrates the distribution of the connections availability for the shortest cycle scheme. For better visualization, the values in the range of 0.9998 and 0.99999 are depicted. The percentile and the average values are reported in Table A.4. An interesting observation is that, the minimum availability obtained by this scheme is much lower than the one achieved by the proposed protection scheme in geometric routing. This can be explained by the fact that for some connections in the topology, it might not be feasible to find two node disjoint paths while the proposed protection scheme is successful in finding an alternate path. The average connection availability achieved by both schemes are almost the same (0.9998). However, the maximum availability of the shortest cycle reaches 0.99999 while in the geometric scheme it reaches 0.99988.

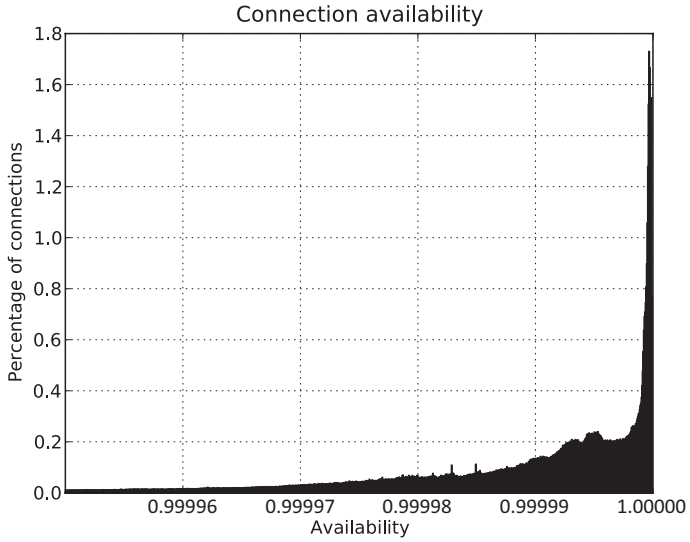


Figure A.11: Histogram of the connection availability of shortest cycle scheme on CAIDA topology.

Table A.4: Percentile and average values for connection availability in 3 schemes :i) geometric routing without protection (G) ii) geometric routing with protection (G-P) and iii) shortest cycle (SH)

	Min	25	50	75	Max.	Average
G	0.94202	0.99388	0.99600	0.99845	0.99985	0.99531
G-P	0.98516	0.99984	0.99986	0.99987	0.99988	0.99983
SH	0.94841	0.99997	0.99999	0.99999	0.99999	0.99987

A.6 Summary and future work

We calculated connection availability of geometric routing over optical transport network on the Internet topology. Finding a reasonable model for the Internet topology was a challenge due to the incompleteness of the available data. This was tackled by using different databases and some estimations on the geolocation of the nodes in the Internet graph.

As fiber optic cable is the dominant component in the availability model of network (frequency of the cuts and long repair times), the availability of connections is dependent on the distances of the adjacent nodes in the topology. Using the explained Internet model, the length of the links in the Internet topology ranges from 10 km to 20K km. Therefore, the availability of connections including the links with large length as 20K km are considerably negatively affected. The other dominant factor in the connection availability is the hop counts between source

and destination of the connections. However, the results in our prior work showed that the proposed protection schemes result into paths with length close to the shortest path length. The experimental results showed that the proposed scheme performs reasonably well compared to the shortest cycle alternative. Although the maximum achievable connection availability is higher in the shortest cycles, the minimum availability achieved by the proposed protection is much higher than the other scheme.

In large topologies such as Internet, the probability of multiple failures is not negligible. In order to obtain higher connection availability, protection against more failures should be considered. An interesting direction for future work is to evaluate the connection availability in geometric routing for multiple failure scenarios. Using recovery schemes against multiple failures should allow for high availability of up to 0.99999 as the ultimate goal.

A.7 Conclusion

In this paper, we considered geometric routing as an alternative to BGP to route between Autonomous Systems (ASes) in the Internet. We used a simple and promising greedy embedding based on a spanning tree of the network. As failures in the network might affect the greedy embedding, recovery techniques were considered to avoid packets reaching a dead end. We performed an availability analysis of the protection scheme for single failure in geometric routing on the Internet topology. For this analysis, we used a network model for the Internet topology which was challenging to achieve due to the incompleteness of the existing data. The availability was evaluated on a AS-level graph of CAIDA. In order to achieve a realistic evaluation, we used a multi-layer model in which links are protected in lower layers than network layer. The results showed significant enhancement in the availability compared to the scheme when no protection for the geometric routing was considered. We compared our results with an existing alternative in which the shortest cycle between the source and the destination is calculated. An interesting observation was that the minimum connection availability achieved by the proposed scheme was much higher than the shortest cycle scheme and the average connection availability obtained by both schemes were almost the same.

Acknowledgment

This work is partly funded by the European Commission through the EULER project (Grant 258307), part of the Future Internet Research and Experimentation (FIRE) objective of the Seventh Framework Programme (FP7).

References

[1] G. Huston. *BGP Routing Table Reports*, 2013. http://bgp.potaroo.net/.

[2] R. Fonseca, S. Ratnasamy, J. Zhao, C. Ee, D. Culler, S. Shenker, and I. Stoica. *Beacon vector routing: Scalable point-to-point routing in wireless sensornets*. In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2, pages 329–342. USENIX Association, 2005.

[3] C. Papadimitriou and D. Ratajczak. *On a conjecture related to geometric routing*. Theoretical Computer Science, 344(1):3–14, 2005.

[4] S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester. *Single failure resiliency in greedy routing*. In Proceedings of the 9th international conference on Design of Reliable Communication Networks, pages 312–319, 2013.

[5] S. Sahhaf, W. Tavernier, D. Colle, M. Pickavet, and P. Demeester. *Link failure recovery technique for greedy routing in the hyperbolic plane*. Computer Communications, 2012.

[6] *The CAIDA AS Relationships Dataset*, January 2004. http://www.caida.org/data/active/as-relationships/.

[7] M. Clouqueur and W. D. Grover. *Availability analysis of span-restorable mesh networks*. Selected Areas in Communications, IEEE Journal on, 20(4):810–821, 2002.

[8] D. Arci, G. Maier, A. Pattavina, D. Petecchi, and M. Tornatore. *Availability models for protection techniques in WDM networks*. In Design of Reliable Communication Networks, 2003.(DRCN 2003). Proceedings. Fourth International Workshop on, pages 158–166. IEEE, 2003.

[9] Q. She, X. Huang, and J. P. Jue. *How reliable can two-path protection be?* Networking, IEEE/ACM Transactions on, 18(3):922–933, 2010.

[10] D. A. Mello, D. A. Schupke, and H. Waldman. *A matrix-based analytical approach to connection unavailability estimation in shared backup path protection*. Communications Letters, IEEE, 9(9):844–846, 2005.

[11] J. Ahmed, C. Cavdar, P. Monti, and L. Wosinska. *Hybrid Survivability Schemes Achieving High Connection Availability With a Reduced Amount of Backup Resources [Invited]*. Journal of Optical Communications and Networking, 5(10):A152–A161, 2013.

[12] M. Azim and M. Kabir. *Availability study of M:N automatic protection switching scheme in WDM networks*. Journal of High Speed Networks, 18(1):1–13, 2011.

[13] M. Azim and M. Kabir. *Availability analysis under multiple link failures in WDM networks with shared-link connections*. Photonic Network Communications, 23(1):83–91, 2012.

[14] E. Calle, J. Segovia, and P. Vila. *Availability Analysis of GMPLS Connections based on Physical Network Topology*. Other IFIP Publications, (1), 2011.

[15] R. Kleinberg. *Geographic routing using hyperbolic space*. In INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, pages 1902–1909, 2007.

[16] D. Eppstein and M. Goodrich. *Succinct Greedy Geometric Routing Using Hyperbolic Geometry*. IEEE Transactions on Computers, 60(11):1571–1580, 2011.

[17] R. Flury, S. Pemmaraju, and R. Wattenhofer. *Greedy routing with bounded stretch*. In INFOCOM 2009, IEEE, pages 1737–1745, 2009.

[18] D. Chen and P. Varshney. *A survey of void handling techniques for geographic routing in wireless networks*. IEEE Communications Surveys and Tutorials, 9(1):50–67, 2007.

[19] A. Cvetkovski and M. Crovella. *Hyperbolic embedding and routing for dynamic graphs*. In INFOCOM 2009, IEEE, pages 1647–1655, 2009.

[20] J. Vasseur, M. Pickavet, and P. Demeester. *Network recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Morgan Kaufmann Publishers, 2004.