

2-9-2010

Naming and discovery in networks : architecture and economics

Joud Khoury

Follow this and additional works at: https://digitalrepository.unm.edu/ece_etds

Recommended Citation

Khoury, Joud. "Naming and discovery in networks : architecture and economics." (2010). https://digitalrepository.unm.edu/ece_etds/136

This Dissertation is brought to you for free and open access by the Engineering ETDs at UNM Digital Repository. It has been accepted for inclusion in Electrical and Computer Engineering ETDs by an authorized administrator of UNM Digital Repository. For more information, please contact disc@unm.edu.

JOUD SAID KHOURY

Candidate

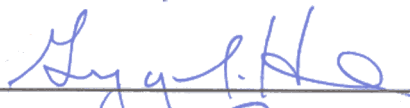
ELECTRICAL & COMPUTER ENGINEERING


Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Chaouki Abdallah  , Chairperson

Gregory Heileman 

Wennie Shu 

Kate Krause 

Naming and Discovery in Networks: Architecture and Economics

by

Joud Said Khoury

B.E., Computer Eng., Lebanese American University, 2003

M.S., Electrical. Eng., University of New Mexico, 2006

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy

Engineering

The University of New Mexico

Albuquerque, New Mexico

December, 2009

©2009, Joud Said Khoury

Dedication

*To the memory of my father, Said
To the hard work of my mother, Layla
To my dear nephews, Said, Serge, Serena, and Thomas
To my lovely twin sister, Jacky
and finally,
To my great brothers, Jack and John
For their endless love and support.*

Without ambition one starts nothing. Without work one finishes nothing. The prize will not be sent to you. You have to win it. The man who knows how will always have a job. The man who also knows why will always be his boss.

- RALPH WALDO EMERSON (Essayist, philosopher and poet)

Acknowledgments

This dissertation would not have been possible without the help of God and many people. First, I would like to thank my advisor and mentor, Chaouki Abdallah, for his continuous support, and his help and patience throughout this work. His guidance makes my working and learning experience a very challenging and enjoyable one. Since I started working with Chaouki, he made me feel that his concern for me goes far beyond the dissertation. He has helped me in every step I have taken. I am very grateful.

I would also like to thank my committee, Gregory Heileman, Wennie Shu, and Kate Krause for their valuable comments and advice. To my colleagues and collaborators, Henry Jerez, Jorge Crichigno, Luca de Cicco, and Venkata Pingali, thank you for your time and thoughts during our research discussions.

I would also like to dedicate this thesis to my mother for her endless love and support and to my brothers and sister. Thank you Layla, Jack, Johnny, and Jacky. Specifically, to my brother Johnny, I am very grateful.

Finally, I want to convey my gratitude to all my friends who helped me and supported me. My journey would have been much harder without their continuous encouragement. To the very special and lovely Vanja Spasic, *hvala* and to my dear friend Tannous Franjeh, thank you.

The work presented in this dissertation was partially funded by the National Science Foundation (NSF) grant CNS-0626380.

Naming and Discovery in Networks: Architecture and Economics

by

Joud Said Khoury

ABSTRACT OF DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy

Engineering

The University of New Mexico

Albuquerque, New Mexico

December, 2009

Naming and Discovery in Networks: Architecture and Economics

by

Joud Said Khoury

B.E., Computer Eng., Lebanese American University, 2003

M.S., Electrical. Eng., University of New Mexico, 2006

Doctor of Philosophy, University of New Mexico, 2009

Abstract

In less than three decades, the Internet was transformed from a research network available to the academic community into an international communication infrastructure. Despite its tremendous success, there is a growing consensus in the research community that the Internet has architectural limitations that need to be addressed in a effort to design a “future Internet”. Among the main technical limitations are the lack of mobility support, and the lack of security and trust. The Internet, and particularly TCP/IP, identifies endpoints using a location/routing identifier, the IP address. Coupling the endpoint identifier to the location identifier hinders mobility and poorly identifies the actual endpoint. On the other hand, the lack of security has been attributed to limitations in both the network and the endpoint. Authentication for example is one of the main concerns in the architecture and is hard to implement partly due to lack of identity support.

The general problem that this dissertation is concerned with is that of designing a future Internet. Towards this end, we focus on two specific sub-problems. The first problem is the lack of a framework for thinking about architectures and their design implications. It was obvious after surveying the literature that the majority of the architectural work remains idiosyncratic and descriptions of network architectures are mostly idiomatic. This has led to the overloading of architectural terms, and to the emergence of a large body of network architecture proposals with no clear understanding of their cross similarities, compatibility points, their unique properties, and architectural performance and soundness. On the other hand, the second problem concerns the limitations of traditional naming and discovery schemes in terms of *service differentiation* and *economic incentives*. One of the recurring themes in the community is the need to separate an entity's identifier from its locator to enhance mobility and security. Separation of identifier and locator is a widely accepted design principle for a future Internet. Separation however requires a process to translate from the identifier to the locator when discovering a network path to some identified entity. We refer to this process as *identifier-based discovery*, or simply discovery, and we recognize two limitations that are inherent in the design of traditional discovery schemes. The first limitation is the homogeneity of the service where all entities are assumed to have the same discovery performance requirements. The second limitation is the inherent incentive mismatch as it relates to sharing the cost of discovery. This dissertation addresses both subproblems, the architectural framework as well as the naming and discovery limitations.

Contents

List of Figures	xv
List of Tables	xvii
1 Introduction	1
1.1 Problem Statement	3
1.2 Contributions	8
1.3 Related Work	12
I Architecture	15
2 A Survey of Novel Network Architectures	16
2.1 Classifying Network Architectures	17
2.1.1 Classification Approach	17
2.1.2 Service model perspective	18
2.2 Technical Reference	22

Contents

2.2.1	Communication-oriented	23
2.2.2	Information-oriented	32
2.2.3	Computation-oriented	40
2.3	Conclusion	41
3	The Transient Network Architecture Instance	42
3.1	Introduction	42
3.2	Transient Netowrk Architecture	45
3.2.1	Area of Influence - AoI	46
3.2.2	Entities and Communication	47
3.2.3	Persistent Identification	48
3.2.4	Distributed control-plane functionality provisioning using the Ghost/Shell model	50
3.3	PINT Framework	51
3.3.1	Components and networking primitives	53
3.3.2	Implementation Details	60
3.3.3	Research Impact	61
3.4	Mesh/Ethernet Deployment	62
3.5	Discussion, Future Work, and Conclusion	64
4	Towards a Taxonomy of Inter-network Architectures	66
4.1	Introduction	66

Contents

4.2	Taxonomy	69
4.2.1	Substrate Structure	69
4.2.2	Information Model	73
4.2.3	Towards a complete taxonomy	83
4.3	Applying the taxonomy	85
4.4	Related Work	88
4.5	Discussion: Value and Limitations	89
4.6	Conclusion	90
5	Towards Formalizing Network Architectural Descriptions	91
5.1	Introduction	91
5.2	Background	93
5.2.1	Architectural Styles: What and Why?	93
5.2.2	Alloy	95
5.3	Case Study	98
5.3.1	FARA Overview	99
5.3.2	FARA model	100
5.4	Related Work	110
5.5	Discussion and Future Work	112
5.6	Conclusion	113

II	Naming and Discovery	114
6	Background on Naming and Discovery	115
6.1	Introduction	115
6.2	Definitions	118
6.3	What is Identifier-based Discovery?	122
6.4	Exploring the Design Space	124
6.4.1	TCP/IP Internet	125
6.4.2	Compact Routing	127
6.5	Conclusion	132
7	Discovery Service Differentiation	134
7.1	Introduction and Motivation	134
7.2	What is Multi-Level Discovery (MLD)?	136
7.3	A Multi-Level Discovery Scheme	137
7.3.1	Background: NICR scheme on trees	137
7.3.2	Extending Laing scheme to support MLD	140
7.4	Discussion and Conclusion	141
8	On the Economics of Identifier-based Discovery	145
8.1	Introduction	145
8.2	Background	147

Contents

8.2.1	Networks and Strategic Behavior	147
8.2.2	Discovery versus Search: Why receiver-based discovery?	157
8.3	A Taxonomy of Discovery Schemes	158
8.4	Incentives and Pricing	161
8.5	Conclusion	164
9	Route Distribution Incentives in BGP	165
9.1	Introduction	165
9.1.1	A Simple Distribution Model	167
9.1.2	Our Results	169
9.1.3	Related work	170
9.2	The General Game	171
9.3	Convergence under HRP	176
9.4	Equilibria	178
9.4.1	The Static Multi-Stage Game with fixed schedule	179
9.4.2	Growth of Incentives	186
9.4.3	A Special Subgame	188
9.4.4	Competition Rewards	191
9.4.5	The Repeated Game	195
9.5	Discussion	199

Contents

III Conclusion	203
10 Conclusion	204
10.1 Open Problems	205
IV Appendices	208
A A Mechanism Design Model	209
A.1 The Discovery Mechanism	210
References	213

List of Figures

2.1	Generalized service model view	19
3.1	Examples of different Areas of Influence that form TNA	46
3.2	PINT components and primitives in sample test-bed showing a multi-hop ad-hoc AoI connected virtually to a traditional ethernet AoI. The PINL layer running on all nodes is able to deliver packets to persistently identified entities.	53
3.3	PINL layer building blocks	55
3.4	Click router asks for agent binding from NELO interface.	58
3.5	PI packet format.	59
3.6	Mesh/Ethernet deployment of 3 AoIs (2 mesh and 1 ethernet network). Red circles represent entities.	63
4.1	Interconnection types; A square represents an abstract element (SE or PE), while an ellipse represents a switching element (SWE). . . .	72
4.2	Abstracting network locations (red circles) and visualizing a locator space.	73

List of Figures

4.3	Identifier space is either integrated (1), or disconnected (2).	74
6.1	Sketch of virtual and physical routing in Abraham name-independent compact routing scheme.	131
8.1	Query Incentive Game: node v has an answer to the query.	150
8.2	Trading Network Game: sellers S to the left (circles) connect to traders T (squares) who in turn connect to buyers B to the right (circles). The buyers' values are indicated inside the circles (1 in this case). Equilibrium bid and ask prices are shown above the links. . .	152
8.3	Representation of some common models for discovery.	159
9.1	Sample network (Not at equilibrium): Solid lines indicate an outcome tree T_d under the advertised rewards.	172
9.2	Line graph: a node's index is the stage at which the node plays; d advertises at stage 0; $K = n$	182
9.3	Ring network with even number of nodes: (i) 2-stage game, (ii) 3-stage game, and general (iii) K -stage game.	184
9.4	The payoff matrix of players 1 and 2 for the 3-stage game on the ring of Figure 9.3(ii) when $r_d = 6$	185
9.5	Plot of $g_K(x) - x$ for $K = 4, 5, 6, 7$	202
A.1	The discovery mechanism design framework: mechanism $M = (O, \xi)$	210

List of Tables

2.1	Matrix view classification of inter-network architectures based on their explicit service model classes.	21
4.1	A BNF syntax for taxonomical specification of network architectures.	85
5.1	Operators in Alloy.	96
8.1	Identifier-based discovery schemes.	159

Chapter 1

Introduction

If you don't make mistakes, you're not working on hard enough problems. And that's a big mistake.

- FRANK WILCZEK (2004 Nobel prize winner, physics)

In less than three decades, the Internet has morphed from a research network available to the academic community into an international communication infrastructure. This unprecedented success and evolution of the Internet has been largely attributed to a set of architectural design goals and principles in the original DARPA Internet architecture [1]. As clearly outlined in Clark's seminal paper [2], the goals include multiplexed utilization of resources, survivability, and openness. Despite its tremendous success, there is a growing consensus in the research community that the Internet has architectural limitations and that those need to be addressed in a effort to design a new generation of the Internet, the “future Internet”. Clean-slate efforts in the United States [3, 4], Europe [5], and Japan (e.g., [6]) are underway to redesign the Internet.

Among the main technical limitations of the Internet are the lack of mobility

Chapter 1. Introduction

support, and the lack of security and trust (check [3, 7, 8, 9]). Endpoint ¹, or in general *entity*, mobility refers to a dynamic change in the endpoint's attachment point possibly while a communication session is in progress. The Internet, and particularly TCP/IP, identifies endpoints with a tuple $\{IP\ address, port\ number\}$. The IP address then serves as the identifier of both the attachment point (a location in the topology) and the entity [11]. So far, the IP address has performed well as a location identifier since it inherently embeds topological information and thus fosters routing scalability under aggressive aggregation (the scalability of hierarchical routing is broken however due to continuous de-aggregation of the address space [12], and recent discoveries on the inefficiency of hierarchical routing over the Internet's topology [13]). When mobility is introduced however, IP loses any meaning of identity reference and degenerates into a pure routing identifier, alternatively *locator*. Coupling the endpoint identifier to the routing identifier hinders mobility and poorly identifies the actual endpoint, which should exist independent of its network location or state. On the other hand, the lack of security has been attributed to limitations in both the network and the endpoint. Authentication for example is one of the main concerns of an architecture and it is hard to implement partly due to lack of identity support. More clearly, in the prevailing end-to-end model of the Internet [10], endpoints of a communication channel have no way of authenticating each other. Rather, the channel itself is generally secured using encryption for example. This leads to spoofing, spam, and many other forms of security breaches. Authentication, accountability, and trust are attractive design goals that could directly benefit from a means to identify entities across all layers of the protocol stack. Finally, other important concerns within the current architecture include the lack of the means to detect, report, and correct errors (or in general to manage the network), the difficulty of incorporating emerging technologies and devices (sensors, vehicles, RFIDs, etc.), and the economic barriers that prevent coordination among the different stakeholders. In addition to

¹Check [10] for more on endpoints in the prevalent end-to-end design of the Internet.

its limitations, the Internet is evolving in ways that were not anticipated during its design. As pointed out by Clark [14], the erosion of trust and the emergence of new stakeholders in the Internet are challenging the “simple end-to-end” principle which is the primary contributor to the Internet’s success.

1.1 Problem Statement

The general problem that this dissertation is concerned with is that of designing a future Internet. This problem however is too broad. As Clark puts it [7], “The reason I stress [clean-slate thinking] is that the Internet is so big, and so successful, that it seems like a fool’s errand to send someone off to invent a different one. Whether the end result is a whole new architecture or just an effective set of changes to the existing one may not matter in the end”. The broadness of the scope follows because both terms “Internet” and “architecture” are too broad. The Internet is a network of networks. It is different things to different parties whether those are operators, businesses, enterprises, government, or users. These parties have different and potentially conflicting goals [14]. The tussle between accountability and privacy/anonymity, and the net neutrality debate [15] are examples of such conflicting goals. However, any architectural design must be based on a well-defined set of goals. Besides the fact that it is hard to converge on these goals, the design space is too wide to explore and too many parameters are involved. While specific dimensions of the design space have been thoroughly examined and understood, such as the TCP/IP protocol [16], we still do not understand the implications of mixing different design parameters - as given by the interactions between different protocols, and systems at different layers of the stack. In fact, after surveying the literature, it became obvious that the majority of the recent architectural work explores a small set of design parameters (in the sense that it is either aimed at exploring novel usage models that adhere to

a class of applications, or at directly addressing a specific limitations of the current Internet). All this is compounded by the fact that both “Internet” and “architecture” are hard to model, to measure, and to simulate, not to mention an “architecture for the future Internet” [17, 18, 19]. The scope of the general problem is thus much wider than that of this dissertation.

In this sense, this dissertation focuses on two specific sub-problems. The first is the lack of a framework for thinking about architectures and their design implications, while the second is the apparent limitations of traditional naming and discovery models in terms of *service differentiation* and *economic incentives*. As it shall become clear throughout the dissertation, the two subproblems are complementary towards designing a future Internet. In order to frame the naming and discovery problem, a better understanding of the architectural design space is needed.

Problem 1: Lack of a framework for thinking about architectures and their design implications

It was obvious after surveying the literature that the majority of the architectural work remains idiosyncratic and descriptions of network architectures are mostly idiomatic. There seems to be a growing consensus in the community about the need for designing a smarter network that is more than just a transparent “bit-plumbing” medium. While such evolution into a smarter and more complex Internet is bringing new potentials and service models, the community generally lacks a unified framework or a taxonomy for thinking about such models and their design implications. In addition, architectural descriptions are idiomatic in nature. This was caused by the evolution of a semantically rich terminology that has been adopted by network architects over time. The terminology, despite being informal, reveals a lot of architectural information and has so far enabled efficient communication between architects.² This state of affairs has however, led to the

²This scenario is very similar to the evolution of software architecture modeling in the context of software engineering [20].

Chapter 1. Introduction

overloading of architectural terms, and to the emergence of a large body of network architecture proposals with no clear understanding of their cross similarities, compatibility points, their unique properties, and architectural performance and soundness. Several models for communication systems have been recently proposed, some of which are focused on particular communication aspects such as binding [21, 22] or routing [23]. Others [24, 25] are more general, and concern themselves with multiple communication aspects such as forwarding, naming, addressing. It is important to note however, that the formal modeling and representation of network architectures is fundamentally different from that of communication systems. In fact, while the communication structure is necessary for defining and representing a network architecture, it is not sufficient. In addition to the communication structure, information and computation structures are building blocks that need to be properly understood within modern network architectures. Communication systems tend to share the same set of elements and are generally concerned with switching properties of networks and their associated communication and control primitives. On the other hand, network architectural descriptions are concerned with high-level architectural abstractions, their interactions, their structural and behavioral properties, and the constraints and invariants that define each architecture.

Problem 2: Limitations of traditional discovery schemes in terms of *service differentiation* and *economic incentives* We have discussed earlier the limitations of the original Internet design in terms of supporting mobility and security. To address these limitations, one of the recurring themes in the community is the need to separate an entity's identifier from its locator to enhance mobility (an entity can move while maintaining the identifier) and security (trust information may be associated with the entity at all levels). For example, several incremental proposals have initially focused on solving the mobility problem by decoupling the host identity from the attachment point [26, 27, 28, 29]. The common approach is to

insert a level of indirection on top of the network layer that manages the abstraction of host identities. Other architectural approaches to separation have been discussed (*e.g.*, [13, 30, 31, 32]). More recently, the identifier-locator separation theme has been adopted by the clean-slate design community (*e.g.*, [6, 33, 34, 35]). Hence, the separation of identifier and locator is a widely accepted design principle for a future Internet. Separation however requires a process to translate from the identifier to the locator when discovering a network path to some identified entity. We refer to this process as *identifier-based discovery*. To eliminate confusion, an *identifier* in this context is a name that identifies an entity rather than a location on the network. Identifier-based discovery, simply referred to as *discovery* hereafter, is a core network service aimed at discovering a network path to an identified entity. Discovery is usually the first step in communication, even before a path to the destination entity is established. Given an identifier of some entity on the network, discovering a path to the entity could either utilize mapping/resolution where the identifier is mapped to some locator (*e.g.*, [28, 36], and the Domain Name System (DNS)), or it could utilize routing-on-identifiers (*e.g.*, [34, 37, 38, 39]). In either case however, an underlying routing scheme that routes on locators typically exists and is utilized after a path has been discovered for efficient communication. We recognize the following limitations that are inherent to the design of traditional discovery schemes:

- **Homogeneity of the service** An identified entity (such as a node or service), wishes to be discoverable by the rest of the network. A discovery mechanism provides such service to the entities. We define the *discovery level* to be a measure of “how discoverable” an entity is by the rest of the network - this is “how easy” it is for the network to discover the entity not the opposite. The performance of discovery, or the discovery level, could significantly affect the entity’s business model especially in time-sensitive application contexts. If discovering an entity takes a significant time relative to the entity’s download

time, the requesting user's experience suffers. As an example, when no caching is involved, the DNS resolution latency comprises a significant part of the total latency to download a webpage (10-30%) [40, 41]. Traditionally, the design of discovery schemes has assumed that all entities have the same discovery performance requirements, thus resulting in homogeneous demand. In such a setting, the discovery schemes deliver a discovery service that is oblivious to the actual, possibly heterogeneous, discovery requirements - and valuations - of the different players. In reality however, the CNN site will likely value a higher discovery level more than a generic residential site. Differentiating the discovery service is thus the first goal.

- **Incentive mismatch** Obviously, there is a cost associated with being discoverable. This could be the cost of distributing and maintaining information (state) about the identifiers to provide a certain discovery level. In the majority of current schemes, the discovery demand is insensitive to cost since no cost structure exists and hence demand flattens out to a homogeneous level. The insensitivity of demand to cost structures makes the problem more important in environments where state is maintained at nodes that are not themselves consumers of the service (hence the cost of state on such nodes needs to be paid for by someone or else there is no reason/incentive to the node to keep the state). Accounting for and sharing the cost of discovery is an interesting problem whose absence in current path discovery schemes has led to critical economic and scalability concerns. As an example, the Internet's Border Gateway Protocol (BGP) [42] control plane functionality is oblivious to cost. More clearly, a node (BGP speaker) that advertises a provider-independent prefix (identifier) does not pay for the cost of being discoverable. Such a cost, which may be large given that the prefix is maintained at every node in the Default Free Zone (DFZ) ³, is paid for by the rest of the network. Such incentive mismatch in the

³The DFZ refers to the set of BGP routers in the Internet that do not have any default

current BGP workings is further exacerbated by provider-independent addressing, multi-homing, and traffic engineering practices [12]. Notice here that BGP with its control and forwarding planes represents a discovery scheme on prefixes which are technically flat identifiers in a largely de-aggregated namespace. Hence, *we conjecture that a discovery scheme should be aware of incentives and cost necessitating that entities pay for the cost of obtaining the service.*

1.2 Contributions

The balance of this dissertation is divided into two parts: part I (chapters 2,3,4,5) addresses problem 1 and aims at framing the architecture space and investigating an architectural instance that is designed around persistent identification of all network entities to foster mobility and security. Building on part I, part II (chapters 6,7,8,9) addresses problem 2 by presenting a general model for discovery in large-scale networks and focusing on two important design goals: service differentiation and economic incentives.

Specifically, we present the following contributions in part I:

- Chapter 2 surveys the inter-network architecture space focusing on radical architectural designs (relative to the original Internet design). We survey the proposals based on the implemented service model whether communication-, information-, or computation-oriented. We show that while the communication structure is necessary for defining and representing a modern network architecture, it is in general insufficient. Information and computation structures are other building blocks that need to be properly understood within modern

route as part of their routing table, i.e., any such router keeps state about every advertised prefix/destination.

Chapter 1. Introduction

networks. The chapter additionally serves as a technical reference for the rest of the dissertation.

- After surveying the literature in chapter 2, we elaborate on one point in the design space. Chapter 3 presents our experiences from the design and implementation of a clean-slate network architecture, the Transient Network Architecture (TNA) [33]. TNA is a novel architecture centered around the theme of persistent identification of all network entities. We introduce the building blocks of TNA and we present the Persistent Identification and NeTworking research framework (PINT) and test-bed deployment. PINT exposes to the research community a modular and extensible set of networking components and primitives, which enables novel research and experimentation atop a persistent identification and networking framework. The framework is designed to support the following key concepts: (1) Intrinsic support for unstructured networks; (2) persistent identification and certification of network entities; (3) distributed control-plane functionality provisioning using mobile agents; and (4) seamless mobility. We present the implementation of the components and primitives within PINT, and we discuss our experiences with the framework based on a first deployment on wireless mesh and traditional ethernet networks.
- Chapter 4 builds on the previous two chapters to present a taxonomy of inter-network architectures. The taxonomy provides a framework for better understanding, organizing, and thinking about the complex architecture design space. Our taxonomy defines a network architecture as a dichotomy between the physical substrate structure, and the information model. On one hand, the substrate structure characterizes the network’s topology, the functional units, and their interconnection structure. On the other hand, the information model, which operates on top of the substrate structure, characterizes the underlying addressing structure, the data entities and the functionality attached to them,

Chapter 1. Introduction

and the relative control structure. To the best of our knowledge, this is the first general, information-centric taxonomy in the literature.

- Chapter 5 investigates the viability of formal architectural modeling. We present a design methodology for formally describing and reasoning about network architectures and architectural styles. The methodology is demonstrated by detailing a formal model for the FARA [30] family of network architectures. The chapter provides a framework for network architects to formally group various architectures into a set of styles based on their common structural and behavioral characteristics, enabling researchers to better represent, analyze, reason about, and infer their important properties.

Building on part I, part II proceeds to address the limitations of traditional discovery schemes as specified in problem 2. We present the following contributions:

- Given the confusion that is generally associated with the terms name, address, identifier, locator, and discovery, chapter 6 re-defines these terms to set the stage for further investigation of naming and discovery problems in later chapters. We revisit the original definitions of name and address, redefine those within a general model elaborate on the confusion that arises among the different terms, and we introduce the discovery problem. Additionally, the chapter discusses the compact routing problem [43, 44, 45], and the concept of stretch, which is relevant to the discussion in chapter 7.
- In terms of service differentiation, chapter 7 introduces the *multi-level discovery* (MLD) framework which is concerned with the design of discovery schemes that can provide different service levels to different sets of entities. We provide a proof-of-concept MLD architecture in the context of Name Independent Compact Routing (NICR) [43, 44]. To the best of our knowledge, this is the

Chapter 1. Introduction

first work to introduce discovery service differentiation and to demonstrate its feasibility.

- Finally, in terms of economic incentives, chapter 8 motivates the problem while chapter 9 presents an incentive model for a general discovery scheme. Specifically, in chapter 8 we present a broad treatment of the main economic issues that arise in the context of identifier-based discovery on large scale networks. Providing a discovery service while accounting for the cost and making sure that the incentives of the players are aligned is the general theme of the chapter. We motivate the subject, present a taxonomy of discovery schemes and proposals based on their business model, and pose several questions that are becoming increasingly important as we proceed to design the inter-network of the future. This sets the stage for chapter 9 which presents an incentive model for route distribution in the context of path vector routing protocols (mainly BGP). We model BGP route distribution and computation using a game in which a BGP speaker advertises its prefix to its direct neighbors promising them a reward for further distributing the route deeper into the network, the neighbors do the same thing with their neighbors, and so on. The result of this cascaded route distribution is an advertised prefix and hence reachability of the BGP speaker. We first study the convergence of BGP protocol dynamics to a unique outcome tree in the defined game. We then proceed to study the existence of equilibria in the *full information* game considering competition dynamics. To the best of our knowledge, this is the first work that presents a taxonomy of discovery models and analytically studies the emerging incentive mismatch problem.

1.3 Related Work

This section reviews broadly related work. We cite more specific related work within each chapter separately.

First, in terms of network architectural proposals, these are surveyed in chapter 2. We survey a wide array of proposals that are either independent contributions to the field or are part of the future Internet initiative [3]. In terms of modeling inter-network architecture, there are two broad areas of related work. The first is concerned with network architecture and communication system modeling, while the second deals with software system modeling. Regarding network architecture modeling, the Internet architecture has been extensively studied over the past decade. Since Clark's seminal paper [2] which highlights the connection between the intended goals of the DARPA Internet and design decisions that govern its current operation, a lot of work has focused on further understanding the Internet's architecture and design principles (*e.g.*, [46, 47, 14, 48, 49, 50]). Other related work in this vein includes communication system modeling or modeling of specific network subsystems [24, 21, 22, 25]. Another class of work relates to software architectural modeling. There has been a lot of focus on formally modeling software architecture [51, 52] and describing architectures using Architecture Description Languages (ADL) [53, 54, 55, 56, 57]. Applying concepts from Object Oriented (OO) programming such as inheritance and composition as well as verification of structural properties and compatibility checking are concepts demonstrated in this vein of work. We build on this work and we apply it to network architecture modeling.

Service differentiation on the Internet, referred to as Quality of Service (QoS) differentiation, is associated with differentiation in levels of performance as it relates to timeliness and bandwidth levels. The idea of service differentiation has been applied to a wide variety of services on the Internet to provide end-to-end guarantees

Chapter 1. Introduction

on performance. This requires QoS provisioning within the network core (as with the IntServ [58] and the DiffServ [59] architectures) as well as in the edges. We refer the reader to [60] for a taxonomy of Internet QoS differentiation. The general idea though in QoS differentiation is to differentiate the performance of data delivery. This is different than the goal we seek in this dissertation. We are concerned with the differentiation of the discovery service where the service level is a measure of how discoverable an entity is by the rest of the network.

Finally, in terms of modeling complex social and economic interactions and incentives of agents, check [61, 62] for an introduction. The work in [61] presents an interesting overview of several tools that are important in bridging computer science and economics to better understanding problems that arise in the context of the Internet. Additionally, [62] presents an interesting overview of several of the problems and applications arising at the interface between information and networks. To study economic incentive issues in networks, two main tools are extensively used: game theory [63] and mechanism design [64, 65]. Game theory is a fundamental mathematical tool for understanding the strategic interactions among selfish agents, particularly on the Internet over which autonomous agents (e.g. ASes) interact. The theory provides several solution concepts to help study games that arise in different situations and that have specific requirements and varying underlying assumptions. For formal definitions of the solution concepts and a comprehensive treatment of the topic, we refer the reader to [63] (and to [65] for an algorithmic treatment and wide range of tools and applications). The most central and widely applicable solution concept is the *pure strategy* Nash equilibrium (PSNE or NE) which could be simply thought of as a set of strategies of the players that forms a stable solution to the game. A more stringent solution concept is the *dominant strategy* solution. Unlike the pure strategy solution, a dominant strategy yields a player the highest payoff independent of the strategies of the rest of the players. Dominant strategies are very attractive solutions when they exist, and when they don't exist game designers

Chapter 1. Introduction

might try to design for them. The mechanism design framework [66] provides this solution allowing the mechanism “designer” to achieve a dominant strategy solution (in addition to other design goals). An important extension to mechanism design framework, Algorithmic Mechanism Design (AMD) [64], deals with the computation complexity of the solution and Distributed AMD [67] further considers the “network complexity” in distributed settings. The AMD framework has been applied to wide variety of networking problems to provide incentives for agents to act truthfully. A small sampling of the work that utilizes game theory and AMD includes inter-domain routing [68, 69, 70, 71], routing in ad-hoc networks [72], multicast cost-sharing mechanisms [73, 74], network formation games [75], peer-to-peer search [76, 77, 78], etc. The work by Bauer et. al [79] assesses the assumptions made by the traditional mechanism design model and its limitations when applied to networking problems, particularly the homogeneous utility functions and the single-shot execution. While it provides tractable solutions, AMD tends on rely on a centralized “designer” and does not model both supply and demand. In this sense, we shall utilize game theoretic tools to study an incentive issue that arises in the context of discovery.

Part I

Architecture

Chapter 2

A Survey of Novel Network Architectures

Internet architectures may be broadly categorized into either incremental or radical efforts. Incremental architectures, such as [80, 81, 82], generally aim at addressing particular limitations of the current Internet architecture through patching, while radical architectures, such as those supported by the FIND [3] initiative, tend to adopt a clean-slate approach to designing a “better” Internet, without being necessarily restricted by the current Internet model.

This chapter serves two main purposes: 1) to survey the literature and highlight commonalities across the spectrum of solutions, and 2) to present a reference for the rest of the dissertation. We start in section 2.1 by classifying architectures based on the service model they are intended to support. In section 2.2, we proceed to overview several clean-slate architectural instances that we shall refer to throughout the rest of the dissertation.

2.1 Classifying Network Architectures

Before discussing our classification approach, we recall some general definitions. A *computer network* is an inter-connection of computers over which information¹ flows. The *network architecture* is the conceptual design and the fundamental operation structure of a computer network. Based on these definitions, one may clearly recognize the obvious defining structures of a computer network: computers and inter-connections, communication, and information structures.

2.1.1 Classification Approach

How do we approach the classification problem given the complexity of the design space? In other words, what should the defining element(s) of our classification model be? We start by recognizing that every design is intended to support a set of goals, which generally encapsulate the pressing needs/requirements of users². Generally speaking, the design process then involves converging on a set of defining structures, and proceeding to optimize those. The outcome is an architectural design that is comprised of the following abstraction levels: 1) the *outer-architecture* represents what the network user can see. This is analogous to the network service interface or *Instruction Set Architecture (ISA)* which defines the addressing modes, the data object semantics, and the available operations; and 2) the *inner-architecture* represents the internal operation structure of the network including the low level substrate structure and the functional aspects to support the outer-architecture.

We believe that both abstraction levels provide useful and complementary insights regarding the architectural landscape. Hence, to help answer our question of what the

¹Information, content, and data are used interchangeably within this chapter, unless otherwise specified, to represent data abstractions recognized by the network.

²Within the discussion, a *user* is the general term used to abstract any entity that utilizes the network services.

defining element(s) of the taxonomy should be, we found it useful to classify some of the existing literature based on their supported service model (or the types of services the network provides to its users). This view has helped us in understanding the underlying goals behind an architectural design, and has additionally highlighted the information model as the main defining element of our taxonomy which we present in chapter 4. The high-level classification, which we refer to as the service-model perspective, is briefly discussed next.

2.1.2 Service model perspective

Classifying architectures from this perspective is motivated by several factors. First, the service model approach implicitly accounts for the needs of the users relative to a network, which is the ultimate goal of any network design. For example, the Internet’s simple “best-effort delivery” service model came about to satisfy a set of goals, as explained in [2], primarily allowing multiplexed utilization of resources (which led to packet switching, domain, gateways), survivability (which led to end-to-end state), etc. Second, most network architectures tend to be naturally categorized and described relative to their service models. For example, we find in the literature the “data-oriented” network architecture [34], the “delay-tolerant” architecture [83], the “differentiated services (diffServ)” architecture [59], and so on. Finally, such a classification could enable future reasoning about - and evaluation of - the degree to which a particular architecture satisfies the service requirements of the users. One such evaluation methodology based on utility was proposed in [46]³.

The generalized service-model perspective is depicted in Figure 2.1. The communication, information, and computation⁴ models represent the building blocks that

³In [46], Shenker defines utility as the degree to which a network service model matches the needs of the network users, i.e., how good an architecture is, is measured by the happiness of its users.

⁴We abuse terminology referring to the terms computation and programmability inter-

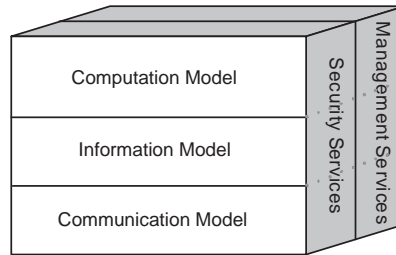


Figure 2.1: Generalized service model view

collectively define, together with the security and management services, the general service model of any network architecture. By building blocks we mean that every architecture must provide these three models, whether explicitly or implicitly⁵. On the other hand, security and management services are not building blocks (since one can easily come up with architectures that do not provide any security or management services), and they operate across the communication, information and computation models.

- *Communication model*: This service model represents the communication and control services offered by the network. For instance, delivery services whether “best-effort”, QoS-aware ([59]), aware of disruption ([83]), and/or geographic location ([84]) all belong to this model. Communication paradigms whether connection-oriented (*e.g.*, ATM) or connectionless (*e.g.*, SMDS, X.25) are classified under this model as well.
- *Information model*: This model deals with the information services that the network provides to its users. The networking community currently recognizes the need for network built-in information services (naming, searching, security, and analysis services) to support a multitude of applications and their

changeably hereafter.

⁵For example, the Internet provides an information model implicitly (the datagram and information transparency) but not explicitly.

requirements (archiving, distribution, etc.) [85].

- *Computation or Programmability model*: This model represents the level of programmability support within the network. The programmability services might potentially span all the other service models, allowing for example the programmability of the communication model and/or the information model etc. Programmable networks [86, 87], for example, provide an explicit computation model.

The security and management services provided by the network are generally, but not necessarily, offered in-band with the rest of the service models. For example, secure communication services include secure end-to-end tunneling and transport (IPSec or SSL), secure identity (HIP [28]) etc. In general, the approach to security and management in traditional networks has been incremental, and not accounted for by design.

In fact, it is possible (and maybe convenient) to fully classify the literature based on the service model view depicted in Figure 2.1 if each of the constituent service models is further divided into its defining elements. However, and as we are mainly interested in the inner-architecture, the major focus of this section is to illustrate some of the prominent architectural work that represents critical points of the aforementioned service spectrum. Additionally, we believe that the independent contributions to the field are converging, and this section aims to highlight such phenomenon by means of a survey.

Chapter 2. A Survey of Novel Network Architectures

		Year	Comm	Info	Comp	Description
Independent Proposals	Internet [2]	1970s	x			providing best-effort delivery of datagrams among globally identified endpoints
	Active Nets [87]	late 90's	x		x	provide a framework for dynamic creation and deployment of network services at runtime
	TRIAD [88]	2000-1	x	x		exposing a "content-layer" that provides transparent access and distribution of named content
	Plutarch [89]	2003	x			provide a communication model that inherently allows inter-operation of semantically disparate domains without mandating uniformity across them
	FARA [30]	2003	x			provide an abstract network model that builds on the Internet's "best effort" service model adding clean separation of endpoint names from network addresses
	TurfNet [90]	2004	x			similar to Plutarch service model, but with global naming
	DONA [34]	2007	x	x		providing data-access (locating and retrieving data) independent of location as well as providing data distribution from multiple locations
FIND [3]	Postcards [91]	2006	x	x		providing reliable delivery (push/pull) of content (large data units or files) to mobile/stationary endpoints using in-network storage/caching
	USwarm [92]	2006	x	x		providing multipoint-to-point bulk data transfer/distribution among hosts (endpoints+intermediaries) with in-network storage/caching
	ITDS [93]	2006	x	x	x	providing information transfer in response to user (endpoint) specified service expressions through in-network processing/data handling
	WiKI [94]	2006	x	x	x	providing a network query interface to users for expressing intent and implementing operations through a declarative framework for managing in-network information and state (router and host state, and data streams)
	TNA [95]	2006	x	x		provides a transient network substrate that enables identification and communication among entities based on global, and persistent (location-independent) identifiers
	PostModern [96]	2006	x		x	providing a tussle-resistant communication service, delivery of functional datagrams, that equips providers with usage control over their networks through policy enforcement, and users with policy-aware control over their traffic forwarding

Table 2.1: Matrix view classification of inter-network architectures based on their explicit service model classes.

Table 2.1 presents, in matrix form, such a survey of the literature, limited to general inter-network architectures. Hence, we do not consider overlays, scoped architectural work (such as naming, or routing architectures) and we do not provide an exhaustive list of inter-network technologies which is not the goal of this sec-

tion ⁶. The work is divided into two parts. The first part overviews some of the independent contributions to the field, while the second part is solely concerned with the FIND [3] work illustrating the community’s view of what the future Internet should look like. More details about each of the architectures of Table 2.1 are provided in section 2.2. Note that Table 2.1 marks the service models only as those are made **explicit** in the architectural description, and consequently it does not contradict our previous claim of the communication, information, and computation models being building blocks. Several insights may be gathered by observing the matrix. First, there seems to be a growing consensus about the need for expanding the network’s service model beyond the communication space, especially as researchers start thinking of designing a future Internet. Additionally, and most importantly, one can clearly notice the emphasis on information services, which is intuitive given the prevailing information-centric usage models with the current Internet.

The next section further elaborates on the service models of each of the architectures of Table 2.1.

2.2 Technical Reference

The section is intended as a technical reference for the rest of the dissertation. We elaborate on the architectures of Table 2.1 as we refer to them frequently throughout the dissertation. Chapter 3 elaborates on a particular architecture, the Transient Network Architecture (TNA).

⁶The majority of inter-network technologies (ATM, X.25, XNS, DECnet etc.) would be classified in our matrix as communication-oriented. We only reference those technologies when they directly serve the goals of our taxonomy. For a comprehensive list of the inter-network technologies, we refer the reader to [97].

2.2.1 Communication-oriented

DARPA Internet

The design principles of the DARPA Internet [1] are clearly outlined in Clark’s seminal paper [2]. The paper highlights the connection between the intended goals of the DARPA Internet and design decisions that govern its current operation. The primary goal of the Internet was to allow multiplexed utilization of its resources, which influenced by the networks (ARPANET) and technologies (packet switching) at that time, led to the adoption of the current Internet structure: domain, packet switching, and gateways as packet switched connecting networks. The other simple goals of the Internet have led to: 1) the survivability requirement resulted in maintaining flow state at end nodes while keeping the core stateless with respect to ongoing flows, 2) the requirement for alternative transport services in terms of latency and reliability has led to the TCP/IP split and the introduction of UDP, 3) the need to support various networks resulted in making a minimum set of assumptions regarding the underlying function provided by the network which is “the network can deliver a packet or datagram”.

The Internet service model can be simply stated as providing *best-effort delivery of datagrams among globally identified endpoints*.

Communication design space

- *Topology*: The topological boundaries within the Internet are referred to as Autonomous Systems (AS) or domains. The domain is an authoritative boundary that maintains local communication policies. Domains are composed hierarchically through customer-provider or peering relationships with a distinguished core set of domains (Tier1).
- *Addressing*: The Internet employs a global hierarchical addressing architecture.

The address space is composed hierarchically to match the topology, rendering the network address a topological forwarding directive. The address space is finite.

- **Naming:** Naming is an out-of-band service that is not part of the core network services, i.e., the network does not recognize a “name”. Hence, the Internet model does not constrain the naming architecture design space. Whether a centralized, global, hierarchical naming architecture (DNS style), or a distributed, flat naming architecture (*e.g.*, OpenDHT) ends up being deployed is irrelevant to the Internet architecture as long as there exists a mechanism to translate a name to an address.
- **Routing and Forwarding:** The Border Gateway Protocol (BGP) [42] is the de-facto standard for inter-domain routing on the Internet. BGP is a policy-based path vector protocol which empowers domains with control over route selection and propagation. Forwarding is thus based on the policies of the domains and the protocol allows for arbitrary preference functions over routes.

Information design space The only information abstraction recognized by the Internet architecture is the datagram. More complex information models must be composed out of the simple datagram. The type of content being delivered over the Internet’s best-effort service whether a static file or an time-sensitive voice stream is irrelevant to the architecture.

Plutarch

Briefly, Plutarch [89] proposes a framework for next generation networks that embraces heterogeneity within and handles it through contexts and interstitial functions. Contexts are like ASes that implement their internal addressing, naming, routing, and transport mechanisms. Interstitial functions map between the set of

functionalities of different contexts. Plutarch is a response to the shortcomings of the current Internet protocol model that unifies all underlying network types through a one-size-fits-all networking and addressing mechanism. This (i.e., IP) has resulted in a semantic bottleneck that is leading to ossification of architecture where it is increasingly difficult to introduce anything but incremental changes. Network Address Translators (RFC 1631), and Resilient Overlay Networks [98] are examples of unclean solutions adopted as a result.

The service model of Plutarch, what users expect from the network, is not addressed in the paper. However, the aim of the work is to provide a communication model that inherently allows inter-operation of heterogeneous networks and mechanisms without mandating uniformity across networks.

Communication design space

- Topology: heterogeneous networks (contexts); boundary: a context is a region of the network that is homogeneous in some way, “a set of bindings with reference to which names may be resolved”; Composition: composition of contexts is not hierarchical, it is either adjacent (border) or containment (nesting); there is no notion of a global or unique “root context”; different namespaces can exist per context; no overlap of contexts;
- Naming and Addressing: local naming and addressing within contexts; inter-context name translation at gateways through “interstitial functions”; Lookup: route-query to chained-context-descriptor mapping (out-of-band); distributed route-query search across contexts (may be flooding);
- Routing: route over the discovered chained-contexts;
- Compatibility with the Internet: Plutarch subsumes the Internet as a context;

TurfNet

The proposal TurfNet [90] is a network architecture for future dynamically composable networks. The architecture is based on the concept of composing autonomous heterogeneous networks, referred to as turfs, dynamically without sharing a global addressing mechanism or network protocol. Composition of networks takes two forms: 1) merging or horizontal composition, and 2) hierarchical independent or vertical composition. Within each AS (turf), independent addressing, routing and resolution mechanisms (control plane functionality) are available and are local to that turf. Across the turfnet, no common network protocol and no shared address space is required. However, a global name space is required to allow communication between the turfnet.

Communication Design Space Separate naming/addressing

- Topology: hierarchically ASs or turfs; composed horizontally (merging) or vertically (customer-provider/peering);
- Addressing: local address space per turf;
- Naming: Global namespace; Lookup: name to address mapping (in-band, i.e., lookup creates forwarding state), recursive lookup creates address and protocol translation state (soft state) within boundary routers up to root;
- Forwarding/Routing: local forwarding within turf using local addresses; intra-turf: hosts external to the turf are mapped to turf address space; mapping soft state maintained by boundary gateways; end-to-end path composed of up-path to root (created during lookup) and down-path to destination turf (created through registration); routing = address + protocol translation at boundary gateways.

FARA

The theme is to separate naming and addressing.

Communication Design Space Even though the FARA proposal does not belong to the radical architectures category and is more of an abstract model for architectures, we include it here for the insight it provides into designing architectures and for its influence on several of the architectures discussed. The basic idea in the FARA [30] architecture is to separate the network address (attachment point) from the entity's address by a "red line". The network layer address is referred to as the Forwarding Directive FD. This is a set of information that if presented to the network can deliver a packet to that location in the network. For example, in the current Internet, IP is a forwarding directive that belongs to a global namespace. Presenting the network with an IP address and a packet is enough for the network to deliver the packet to the destination. In M-FARA instantiation, and unlike the traditional Internet, no global address space is assumed, i.e., no IP addresses. However, there is a set of local address spaces referred to as "addressing realms". Topological information is represented in terms of these private realms that a packet will traverse in transit towards its destination. The FD here is thus a set of sub-FDs specifying the realms on the path. So, in the case that the source entity and the destination entity belong to the same realm, the destination FD has NO topological information. When the source and destination belong to different realms, the destination FD must specify the topology which can be very complex if the private realm addressing is flat. M-FARA's addressing introduces a 2 level hierarchy with a designated globally known "core". Thus the destination FD in this case will consist of (FDup, FDdown) pair of FDs specifying how to reach the core from the source and then how to reach the destination from the core. This design supports mobility across realm boundaries.

NIRA

NIRA [99] is a novel Inter-domain routing architecture that equips users with the ability to choose domain-level routes. The user control over the sequence of providers (ISPs) that packets take introduces competition among the backbone ISPs, thus driving innovation and lowering service cost. NIRA's support for user choice introduces several problems including practical provider compensation, route discovery and representation, and security, which are addressed in the paper.

The service model of NIRA [99] may be stated as *providing an Internet-style communication service that inherently supports user-selected domain level routes*.

Communication design space NIRA reuses many of the Internet's design decisions:

- Topology: strictly hierarchical ASs (domains) with distinguished core (customer-provider and peering relations); concept of domain hierarchy from edge user to core referred to as up-graph;
- Addressing: globally unique, hierarchical address space (IPv6 reused); address encapsulates topological location and provider hierarchy; (scheme: provider rooted hierarchical addressing)
- Routing: valley-free; unicast: 2-segment route (uphill+downhill), one from source to core (uphill) and one from core to destination (downhill); downhill discovered through name service; proactive detection of route failures; BGP-like inter-domain routing within the core, and path vector inter-domain routing over provider hierarchy;
- Naming: naming system required to map endpoint names to downhill route segment; name system design not mandated (may be DHT or DNS style for example).

ROFL

Routing On Flat Labels (ROFL [37]) is a radical architecture that addresses the challenge of how to scalably route on flat labels with no location identifiers. Contrast to the previous proposals, ROFL eliminates location identification (Forwarding directive) and solely depends on persistent identifiers to route inter and intra-domain traffic. The advantages of such approach in addition to mobility and multihoming are fate sharing (no control path since no need to contact resolution infrastructure), simple address allocation, better capture of identities at network layer, and independence from any external resolution systems.

The service model of ROFL [37] may be stated as *providing an Internet-style communication service that utilizes state independent flat labels for network identification and routing.*

Communication design space ROFL eliminates the address dependence on location, hence the network address itself becomes the name.

- Topology: hierarchical ASs; AS up-graph required (domain's provider hierarchy);
- Addressing/Naming: Namespace: flat namespace, circular space similar to Chord [100]; Name semantics: unique persistent and global identifiers, semantic free as in [31], self certifying (HIP [28] public key hash); Naming system: global DHT formation and maintenance as nodes join and leave;
- Routing/Forwarding: compact routing; no name/address resolution since routing on flat DHT; ID translated into source route during forwarding and route follows successor pointers; DHT formed over routers and static hosts.

Postmodern Internet

The postmodern Internet architecture [96] is a reaction to the rigidity of the current Internet’s network layer with respect to different and possibly conflicting policies of stakeholders. The architecture aims to provide a minimalist network layer that anticipates tussle⁷ and accommodates for it through flexibility in introducing policies. Some example policies include inter-domain routing policies, packet filtering policies, policies of who can specify forwarding and who has access to what, service policies, etc..

The service model of the architecture can be stated as *providing a tussle-resistant communication service, delivery of functional datagrams, that equips providers with usage control over their networks through policy enforcement, and users with policy-aware control over their traffic forwarding.*

Communication/Computation design space

- Topology: hierarchical virtual realms (trust boundaries);
- Packets and forwarding: functional datagrams (smart packets containing functional blocks - how, what, where, knobs and dials); user-control over forwarding paths/directives (FDs) when aligned with provider policies; (mechanisms: Resolution from “destination specs” to LinkIDs to reach destination)
- Addressing and Routing: both decoupled from forwarding and not restricted (not part of network layer), i.e., no global addressing mechanism required; (mechanism: use globally unique Link IDs instead to determine paths; inter-realm LinkID routing information dissemination)
- Transparency: in-network packet processing and rewrites of functional blocks.

⁷Users need to control how their traffic is delivered, while providers try to control their network usage.

Information/Security design space functional packets recognized by network; network entities (hosts and realms) have trusted identities generated by decentralized PKI infra; signed packets; Accountability and path signatures;

Geometric Stack

The proposal by Gruteser [84] calls for making geographic/spatial location information an inherent service of the network, for the latter to better address the needs of dense wireless/mobile access networks (geographic routing/addressing/tracking/dissemination). A novel stack is proposed to provide communication through physical space rather than network space (topology) enabling a multitude of applications that utilize geo-routing, geo-casting, and localization.

The service model of [84] may be stated as *providing a distributed location service and a spacial routing primitive for location-centric communication.*

Communication design space: packet-switched; location information available over some coordinate systems with translation among them;

- Topology: hierarchical topology; high-speed wired backbone with wireless edge networks; nodes associated with home areas;
- Addressing: address is geographic identifier (unicast: host ID + last position; geocast: set of coordinates of a zone);
- Forwarding and Routing: linear geometric routing (along 1-D paths called trajectories); source node specified path equation and network decides on forwarding through next-hop local forwarding by intermediate nodes (not source routing but path specification);
- Naming: host identifier (*e.g.*, MAC derivative); Lookup: distributed service to resolve identifier to location (DHT based), location based service.

2.2.2 Information-oriented

The architectures we discuss in this section aim to achieve a similar objective, which is efficient network information support (typically in-band), a service that the current Internet model fails to provide. While sharing the general goal, the design decisions (and mechanisms) employed by each remain different, which is what we try to illuminate next. There is a direct coupling between the information and the communication models, and the latter is generally intended to provide the necessary requirements for efficient data access (time, space, disruption, disaster, etc).

TRIAD

The original Internet architecture provided transport mechanisms that are transparent to the applications or services employing them [2]. As the Internet usage models become more content-oriented (Web traffic, multimedia, or p2p traffic), more intelligence is overlaid on top of the traditional Internet design to provide faster and more reliable content access as is the case with Content Distribution Networks (CDN). TRIAD [88] is a novel architecture that treats content as first-class shifting the communication paradigm from host-centric to content-centric communication. The proposal exposes the limitations of current content distribution models, whether scalability, latency, or architectural openness and consistency. TRIAD addresses the content problem by making explicit a *content layer* that can efficiently *route* towards content. The content layer spans the core of the network by extending traditional IP routers to support *name-based routing*.

TRIAD's service model can be stated as *exposing a "content-layer" that provides transparent access and distribution of named content*.

Information design space

Chapter 2. A Survey of Novel Network Architectures

- Type of information: datagrams/packets, services, and content
- Naming: semantics: URL names compatible with DNS, URL split into domain name of content server and file name, service names are persistent whereas content names (service name + file name) is not; namespace: global hierarchical namespace; naming system: distributed naming infrastructure with no single point of failure;
- Routing: name-based routing: inter-domain content routers (CRs) route based on names towards content servers (caches for closest replicas, transformers); routing is a distributed in-band search operation; single-source multicast support
- Tussle: Content routers (CRs) are provided as ISP infrastructure (similar to BGP routers now) and are thus controlled by the domain's authority, hence ISP control over directory service. This stands in contrast to the current Internet where ISPs have no control over DNS. Additionally, coupling naming and routing at the domain level can potentially lead to more tussle.

Communication design space host-to-content; Delivery of content depends on the communication model (TRIAD reuses HTTP/TCP/IP transport). TRIAD does not mandate the Internet model though.

- Addressing and routing: Addressing is global hierarchical (IPv4 reused). Address is only a forwarding directive used for transient routing/forwarding of information and not for lookup;
- Naming and routing: There is an explicit separation between name and address/forwarding directive. All network entities (hosts) are identified by names (DNS). Name is end-to-end identifier, information model handles routing based on names.

Chapter 2. A Survey of Novel Network Architectures

- **Transport:** in-band lookup and connection setup/transport (using DRP as an alternative to TCP), hence symmetric search and delivery paths (lookup is as available as delivery);
- **Mobility:** Indirection based host mobility, Name based routing abstracts topological location, but since endpoints (hosts) do not advertise their names (for scalability), endpoint acquires new name in visiting domain and inserts a redirection in its home domain.
- **Compatibility with Internet:** highly compatible with Internet infrastructure, requires extensions at directory level.

We can clearly see the significant overlap between TRIAD's information and communication models. The latter is designed to inherently support the former.

DONA

The Data Oriented Network Architecture [34] is a clean-slate redesign of the naming and resolution mechanisms on the Internet. Similar to TRIAD [88], DONA is a reaction to the evolution of the Internet usage models. Initially, the Internet was designed to enable host-to-host communication (FTP, telnet where source explicitly specifies address/locator of destination) over a transparent forwarding engine. This model has significantly changed since into a data-centric model where users access content and services independent of the location of content, services and of the users. DONA proposes replacing DNS names with flat, self-certifying names and replacing the name resolution mechanisms with a name-based anycast primitive that lives over IP. The main design requirements for DONA are persistence, availability and authenticity. The architecture itself is a synthesis of ideas from HIP [28], TRIAD [88], and SFS [101].

Chapter 2. A Survey of Novel Network Architectures

DONA's service model can be stated as *providing data-access (locating and retrieving data) independent of location as well as providing data distribution from multiple locations*. The design decisions of DONA are very similar to those of TRIAD, except for the naming architecture. We simply point the differences.

Information design space

- Naming: Semantics: persistent, self-certifying, flat name (HIP [28]);
- Tussle: ISP control over user's activity (content lookup and registration activity); ISP physical control over Resolution Handlers (which are similar to content routers in TRIAD);
- Security: Authentication and integrity of information (mechanism: PKey cryptography).

Postcards from the Edge

The proposal "Postcards from the Edge" by Yates et. al. is a clean-slate cache-and-forward architecture for a future internet [91]. The architecture is a response to the revolution in access technologies, primarily wireless and mobility, that overwhelm the Internet's basic design assumptions. Given the continuously increasing capacity and decreasing cost of in-network storage, the authors propose an architecture that provides uniform reliable transport of large files across heterogeneous access networks and in the face of intermittent connectivity.

The service model can be stated as *providing reliable delivery (push/pull) of content (large data units or files) to mobile/stationary endpoints using in-network storage/-caching*.

Information design space

- Type of info: large data units (files);

Chapter 2. A Survey of Novel Network Architectures

- Naming of content: Semantics: globally unique (UFID.FQDN) names for files; name service: out-of-band service (i.e., lookup and transport are not coupled), hierarchical name resolution system (DNS-style); Lookup: distributed name-to-cache(s) mapping
- Routing/search: no routing based on names; out-of-band search/lookup to name service (i.e., lookup name then contact host similar to Internet); Rendezvous push/pull through post office nodes (every node knows PO current PO nodes where former can pick its content).

Communication design space: host-to-host; builds on top of Internet best-effort IP service for addressing and routing of control traffic. IP is not essential to the design though.

- Topology: hierarchical topology formed of high-speed wired backbone connected to edge access networks;
- Naming of endpoints: location aware rendezvous service; Lookup: distributed name to post-office mapping; Security: endpoint associated with home autonomous system for authentication when mobile;
- Routing: hop-by-hop routing on location information; Supports Type-of-Service (TOS) to distinguish between transport and caching
- Mobility: artifact of rendezvous
- Compliant with Internet: builds on IP, hence highly compliant.

USwarm

Universal Swarm (USwarm) is a proposal by Venkataramani and Towsley [92] that applies swarming techniques (such as BitTorrent) to design a universal data transfer

architecture that learns from p2p architectures and eliminates selfishness of peers through incentives. The architecture is a response to the data transfer shortcomings on the current Internet. The whole Internet is modeled as a single swarm that employs a distributed metadata resolution system to resolve data-to-peers(s) that can serve the data (analogous to BitTorrent Tracker). An intentional naming system is employed as well to resolve intentions to metadata. Hence 2 level resolution is proposed: intention to metadata to provider peers.

The service model of USwarm can be stated as *providing multipoint-to-point bulk data transfer/distribution among hosts (endpoints+intermediaries) with in-network storage/caching.*

Information design space:

- Type: datagrams, data, metadata
- Naming: Bittorrent model - semantics: self-certifying (publisher, hash) tuple, principal name globally unique and authentic, metadata uniquely specifies data and contains name plus block ids; infrastructure out-of-band intentional resolution system IRS (map intent to metadata); distributed p2p lookup/search;
- Routing/search: locality-aware distributed tracking that involves peers and intermediaries to locate content (i.e., resolve metadata to peers); more control over routing decisions for ISPs (traffic engineering) and users;
- Transport: multipoint to point transport of information; incentive-aware;
- Security: authenticity and integrity of data (mechanism: crypto).

Communication design space: Internet style point-to-point communication employed; no constraints on communication model.

ITDS

The proposal ITDS ⁸ [93] is a response to the simple store-and-forward model of the current Internet. The architecture calls for in-network support of a broader range of services by dealing with information abstractions ⁹ rather than simple bit transfers. Hence the communication model proposed is aware of information rather than simply being a bit-plumbing medium. Additionally, in-network data processing is proposed to implement dynamic user service requirements.

The service model of [93] may be stated as *providing information transfer in response to user (endpoint) specified service expressions through in-network processing/data handling*.

Information/data and Computation design space:

- Type: data, information; semantics: various transfer characteristics (streaming, random access, interactive, canned, . . .)
- Processing: general purpose computation model on routers to support data services; data services already offered by network and not dynamically deployed by users (contrast to active nets for example); service specifications are mapped to computation model rather than dynamically introduced; service composition, control;
- Storage: in-network storage/caching possible on routers.

Computation design space: User specifies intent and network maps computation to resources, hence, limited user control. **Communication design space:** can operate on top of various communication models (*e.g.*, Internet); ITDS focuses on information transfer models “on top of the network”.

⁸Information Transfer and Data Services

⁹The proposal explicitly differentiates between information (*e.g.*, “the requested web page did not change”) and data/content (*e.g.*, the actual web page data).

WiKI

The proposal Wireless Knowledge Infrastructure (WiKI) [94] addresses the limitations of the current Internet in supporting the needs of applications and services given the huge proliferation of wireless, mobile, and ubiquitous computing. WiKI takes a clean-slate approach to designing a future declarative network in which in-network state (router, network, and host state) is treated as distributed data that can be queried by users through declarative languages. Such approach separates logical representation from actual implementation making the network more flexible to change, and more informed about its operations. Hence, WiKI realizes a knowledge plane [102] for the Internet.

The service model of WiKI may be stated as *providing a network query interface to users for expressing intent and implementing operations through a declarative framework for managing in-network information and state (router and host state, and data streams)*.

Information design space:

- Type: internal network information (state), data streams, continuous/static queries; cross layer, cross domain views of data streams; archives
- Processing: integration, aggregation, fusion, joins, etc.
- Security: policy enforcement; distributed monitoring; access-control.

Computation design space: built-in distributed WiKI runtime (proxies); query processing and optimization; user-control: user has control over computation (*e.g.*, route selection/protocol, service composition) through queries (mechanism: Network Datalog language to specify computation); **Communication design space:** wireless/mobile endpoints query WiKI proxy nodes (infrastructure or overlay) that perform query processing; can operate as overlay.

2.2.3 Computation-oriented

A detailed survey on active network is provided by Tennenhouse [87], and a more general one on programmable networks is provided by Campell [86]. The area of active networks has extensively explored the idea of programmable networks with the ability to introduce change into networks dynamically. Some pointers to prominent work in this field can be found at DARPA's site <http://www.sds.lcs.mit.edu/darpa-activenet/>, and at the IEEE issue [103, 104]. This section overviews several example of programmable networks instead of surveying the AN field.

In [105], the authors introduce the concept of programming the network by shifting the computing paradigm on current networks from an end-to-end system to a system in which each fine-grained data element participates in the computation. Within active networks, traditional packets are replaced by “capsules” which contain programs and content simultaneously. Nodes on the network (routers, switches, servers - firewall) can dynamically execute the capsule programs safely and efficiently. The capsules are loaded into a transient environment on the node and are safely executed/interpreted eliminating the ability of the capsule programs to stray beyond the restricted env and thus to compromise the shared resources on the network. The programming abstraction provided by active networks allows user-driven customization of the infrastructure to enable faster deployment new services. It also enables for dynamically adaptive protocols on the network, thus tailoring the infra for user/application needs. Logically, active networks shift the intelligence in the network from the node to the capsule. A capsule for example will compute its path within the network (might need to access routing tables on nodes) instead of the capsule (packet) being dumb. PLAN [106] is a functional programming language for active networks used within packets. It is resource limited and secure enabling a smart means of communicating between nodes. PLANet [107] is an active network implementation that utilizes PLAN to implement network layer on top of an

IP-free link layer. PLANet uses active packets and allows active extensions, used to operate the network, to be downloaded to routers (for ex. to implement services like DNS, address resolution, routing). An active packet within PLANet needs to explicitly specify the destination of evaluation, avoiding the evaluation of the packet program on every node on the packet's path. This is mainly due to the reason that evaluation is computationally expensive. Programs in packets are marshalled at the source node and unmarshalled only at point of evaluation. In general a packet is only forwarded by an intermediary node (by executing an *routeFun* attribute specified by the packet) until it reaches the intended destination on which it is evaluated. Addressing within PLANet is based on 48 bit addresses (implemented with 32 bit IP appended to 16 bit port number) assigning one address per network interface on a node.

2.3 Conclusion

This chapter presented a survey of novel network architectures. We have shown that *while the communication structure is necessary for defining and representing a modern network architecture, it is in general insufficient*. Information and computation structures are other building blocks that need to be properly understood within modern networks. We shall leverage this observation later in the chapter 4 to present a taxonomy that revolves around the architecture's information model. In the next chapter, we elaborate on a specific clean-slate architectural instance, the Transient Network Architecture (TNA). TNA is a novel architecture centered around the concept of persistent identification of all network entities. We introduce the building blocks of TNA and we present the Persistent Identification and NeTworking research framework (PINT) and test-bed deployment.

Chapter 3

The Transient Network Architecture Instance

3.1 Introduction

In chapter 2, we have surveyed a large variety of clean-slate network architectures and proposals. As we have seen, the different designs explore different points in the design space and aim at providing new services as part of the network's functionality. Supporting content location and distribution, programmability and service composition, storage and caching, virtualization and re-configurability, identification and authentication by design are all examples of such services that are of interest for a future Internet (check Table 2.1). This chapter elaborates on one point in the design space by presenting the Transient Network Architecture (TNA) and a framework for experimenting with it. TNA is centered around the theme of persistent identification of all network entities to foster mobility and security.

The emergence of key wireless technologies, the proliferation of mobile devices, and the nomadic user and computing lifestyles on current networks are continu-

ously evolving in synergy. Wireless mesh networks (WMNs), wireless sensor networks (WSNs), mobile ad-hoc networks (MANETs), and vehicular area networks (VANs) are examples of self-organizing unstructured networks that have their local communication paradigms and are optimized to perform under their particular physical constraints. The Internet Protocol (IP) is currently employed to provide inter-networking among heterogeneous access networks. IP unifies the underlying forwarding mechanisms and the routing identifiers providing end-to-end transparency. In other words, the whole intermediate network appears to be homogeneous with a well-defined topology as far as the endpoint is concerned. This abstraction has been very successful and scalable and is based on the assumptions of the original Internet design.

However, with the emergence of heterogeneous access technologies, and with the continuous adoption of wireless communication, maintaining the end-to-end IP abstraction is becoming harder putting more strain on the evolution of the network. Additionally, unifying the addressing scheme (IP address) has led to inefficiencies within emerging networks. Such networks must support IP addressing with the added administration requirements despite the fact that the topological IP address has little physical significance as a routing directive within these networks. Part of our recent work [108, 109] has demonstrated that a persistent identifier might be utilized as a routing identifier (forwarding directive) within a local mesh network that implements a multi-hop routing protocol, hence replacing IP. Another recent work by Kim et al. [110] shows that ethernet bridging can be made scalable and efficient and can route based on MAC addresses within an enterprise network eliminating the need for internal IP subnetting and administration. Again, in this scenario, IP is only useful for external reachability and application interoperability. Add to this the fact that even when IP is implemented within such networks, the majority of communications between the endpoints requires a high level naming system and an indirection mechanism, whether hierarchical (eg. DNS) or flat (eg. DHT [111]),

Chapter 3. The Transient Network Architecture Instance

which endpoints can utilize.

Several research test-beds have been recently proposed to enable experimentation with next generation networks, coexistence of heterogeneous systems, mobile networking, and wireless environments, etc. [112, 113]. This chapter presents the Persistent Identification and NeTworking research framework (PINT) and test-bed deployment that was initiated at the university of New Mexico and the Corporation for National Research Initiatives (CNRI) , as part of the Transient Network Architecture (TNA) [33] project. PINT may either coexist as a deployment on top of the readily available test-beds to provide the identification framework, or it may be deployed into a separate test-bed for scoped research with persistent identification. Briefly, PINT exposes to the research community a modular and extensible set of networking components and primitives, which enables novel research and experimentation atop a persistent identification and networking framework. The framework is designed to support the following key concepts:

- Intrinsic support for unstructured networks;
- persistent identification and certification of network entities;
- distributed control-plane functionality provisioning using mobile agents; and
- seamless mobility.

The framework components include 1) entities that represent the communicating endpoints, 2) areas of influence that abstract sets of entities sharing a common communication protocol, 3) a virtualization model for agent based provisioning of control-plane functionality, and 3) a network substrate virtualization. Novel networking primitives are exposed through the Persistent Identification and Networking Layer (PINL), allowing mobile and stationary entities to communicate securely based

on persistent identifiers that are location independent. The chapter presents a modular, extensible, and portable implementation of the components and primitives within PINT. It then discusses our experiences with the framework so far, based on a first deployment on wireless mesh and traditional ethernet networks.

The remainder of this chapter is structured as follows. Section 3.2 discusses the principal TNA design decisions that guided the development of PINT. Section 3.3 describes the PINT framework and test-bed, and the research opportunities enabled thereof. A deployment over mesh networks is then illustrated in section 3.4. Section 3.5 overviews our current and future work and concludes.

3.2 Transient Network Architecture

We have previously introduced a general architectural vision for a possible future Internet which we call the Transient Network Architecture (TNA) [33]. TNA represents an abstract vision which PINT instantiates. The main goal of TNA is to enable seamless end-to-end communication between mobile and stationary devices across heterogeneous networks and through multiple communication environments. TNA builds on the original logical model of the Internet to form a logical network that allows the effective merging of heterogeneous networks without forcing them to modify their communication protocol but rather their logical coordination mechanism. Mobility, security, and identity persistence are some of the characteristics that TNA tries to support by design.

In this section, we layout the principal design guidelines that pertain to TNA and that guided the development of PINT.

3.2.1 Area of Influence - AoI

Whenever we speak of a local network, we are referring to what we call the Area of Influence (AoI), i.e., the AoI captures the scope of “local” when trying to understand how local is “local”. Briefly, an AoI is a local communication community that defines its own communication protocols and network architecture implementations. Examples of local implementations include, but are not limited to, LANs, Cellular networks, MANETs, sensor nets, and mesh networks. These networks implement their own communication mechanisms and protocols and can survive independently of the global system. A sketch of how currently available networks can fit into the AoI framework, is shown in Figure 3.1. The figure shows how the nodes of a mesh network, for instance, may assemble into an AoI. The AoIs themselves may define their own local communication implementation such as Ethernet, RF or Bluetooth, and even their own local identification mechanisms. The basic constituents of AoIs are network entities which we formally define next.

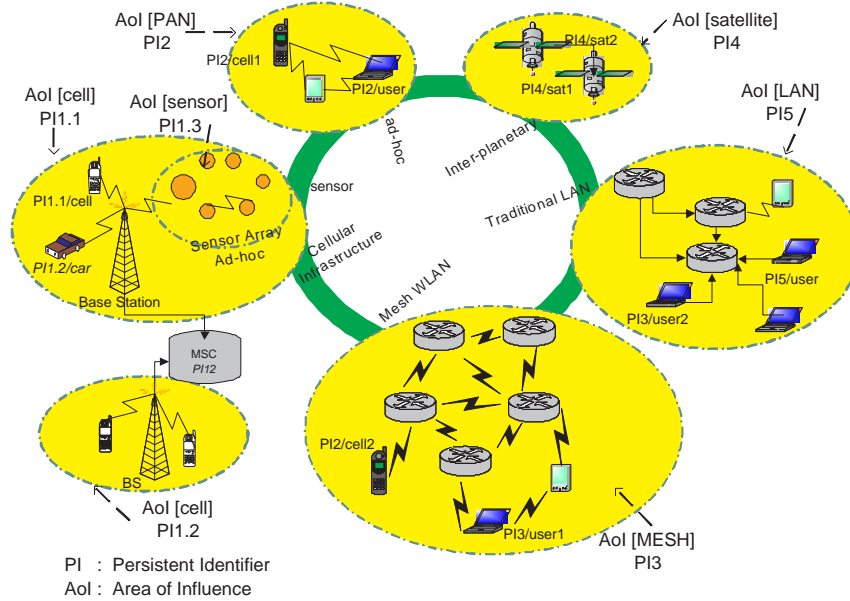


Figure 3.1: Examples of different Areas of Influence that form TNA

3.2.2 Entities and Communication

Based on the definition in [30], an *entity* is the end-point of communication. It is an abstract construct that can represent different network elements including, but not limited to, a process, a thread, a device, a cluster of devices, or a service. The entity is the smallest element on the network that can be mobile. Each entity has its own Persistent Identifier (PI) that is globally unique, and secure by design. Security, as we shall see later, results from the direct association of the PI with a set of credentials that can be challenged by the network at any point in time.

How is this different from the traditional Internet implementation and what advantages does it offer? To answer this question, it is instructive to understand the relation between the entity and the attachment point to which the entity is bound. Traditionally, the Internet and particularly IP has taken a *location-oriented* paradigm to identifying entities, i.e. the most basic entity identifier expressed as a tuple $\{IP\ address, port\ number\}$ is directly dependent on the topological IP address. So far, the IP address has performed well as a location identifier since it inherently embeds topological information and thus fosters routing scalability. However, when mobility is introduced as in the case of wireless networks, IP loses any meaning of identity reference and degenerates into a pure routing identifier. Coupling the entity identifier to the routing identifier hinders mobility and poorly identifies the actual entity, which should exist independent of its network location or state. Several proposals have focused on solving the mobility problem by decoupling the host identity from the attachment point [28, 29, 26, 27]. Most of these efforts propose inserting a level of indirection on top of the network layer that manages the abstraction of host identities. These proposals share the overlay approach on top of IP whereby a high level address is translated to an IP address at some point and routing is an end-to-end, IP-based mechanism. The bottom-line is, *the Internet architecture design makes it inefficient to initiate communication with an arbitrary entity on the*

current Internet, unless that entity has a public IPv4 (or an IPv6) address. As a result, several architectures were proposed to solve the Internet addressing issue as in [82, 81, 80]. However, even when a public address is available, inefficient mobility management schemes prevail requiring centralized infrastructure and continuous end-to-end negotiations between the endpoints over a simple “core”.

Our *entity-oriented* approach to identification and communication elevates the entity to become the first-class network citizen and the center of design. More precisely, and contrasted to the traditional Internet approach, our starting design point is an entity with a globally unique PI that is independent of any topological information. We do so by asking the question: *Starting with persistently identified entities, how should an underlying network be engineered to seamlessly and securely incorporate those entities?*

3.2.3 Persistent Identification

After introducing our *entity-oriented* approach to identification, we describe the characteristics of the PI, and the certification and resolution mechanisms attached to it.

Persistence and global uniqueness are two attractive characteristics of the PI. Persistence of the identifier is essential when the attributes (e.g., state and location information) of the identified entity change, but the identifier itself persists. Global uniqueness is necessary to avoid identifier conflicts especially when the identified entity is highly mobile.

Certification and Resolution

An identifier is used by the entity for interaction with the rest of the system provided the identifier can be challenged and certified within the environment of communication whenever necessary. We isolate three certification realms, as follows:

- *Instance (Red Realm)* is defined relative to the user. It represents the authoritative domain of the user to which a set of entities belong.
- *Local (Yellow Realm)* is defined relative to the local network, AoI. This realm represents the authoritative domain of an AoI which is essential for local interactions among the entities of the AoI.
- *Global (Green Realm)* is perhaps the most challenging to create and maintain, simply because it has to simultaneously guarantee global certification and scalability. The *Green Realm* represents globally trusted authorities. Note that at this level, many globally trusted authorities can co-exist and inter-operate avoiding the pitfalls of a single trust system as is the case with the current Internet.

The colors of the realms indicate the level of trust within the system. For example, certification by the *Green Realm* represents the highest level of trust with respect to the overall system. Certification, or the flow of trust, is “top-down”, from *Green* to *Yellow* to *Red Realms*. Hence, an identifier certified by the *Green Realm* is globally trusted, while an identifier certified by the *Yellow Realm* may only be used for secure interactions within the AoI.

As to resolution, the PI is generally resolved into some information useful for the interaction between the communicating entities. The result of the PI resolution requires certification by the respective realm. Resolution is performed in a “bottom-up” fashion as follows: First, try to resolve against the *Red Realm*. A failure here will

percolate the resolution one level up against the *Yellow Realm* and then against the *Green Realm*. The mechanisms for certification and resolution are closely coupled and their details will depend on the particular architecture implementation.

3.2.4 Distributed control-plane functionality provisioning using the Ghost/Shell model

TNA defines two new abstractions, as follows:

Ghost is the abstraction of a service that provides control-plane functionality; the Ghost is itself an entity that is persistently identified, hence it is mobile.

Shell is the abstraction of the platform/infrastructure over which the Ghosts execute.

TNA utilizes the concept of mobile agents in distributed systems [114] to instantiate Ghosts. The concept of mobile agents provides a novel approach to flexible and scalable distributed network management by better utilizing the network resources and minimizing human intervention [115]. For example, an agent can move the intelligence to the resource instead of moving the resource itself which can save bandwidth. Technologies to support mobile agents (e.g., JINI, JAVA RMI, and CORBA) are becoming more popular and are moving closer to mainstream adoption. The agent makes its own decisions and listens to external requests. It can execute custom business logic, move itself across the network, terminate itself, etc.

3.3 PINT Framework

Before delving into the details of the PINT implementation, we summarize the key features of our framework:

- *Intrinsic support for unstructured networks*: the framework is designed with emerging networks in mind, especially wireless environments. WMNs, WSNs, MANets, VANs, and traditional structured networks should all be able to participate and seamlessly inter-network, while respecting each of the networks' local communication paradigm and protocol implementation.
- *Persistent identification and certification of network entities*: The advantages of using the PI as the network address are several, including:
 - Mobility: The independence of the PI from its attributes is an attractive property for a network layer identifier. The direct advantage of persistence is mobility since an entity that is persistently addressed by the network layer is reachable on that address at all times. Consequently, mobility occurs natively eliminating the network layer indirection introduced by other proposals [28, 29, 26, 27]. In other words, the indirection from a persistent name to a forwarding address (e.g., DNS name [116] to IP address) is eliminated in our framework, since the PI is itself the forwarding address.
 - Security: The PI address is stamped, i.e., it is inherently associated with security information (e.g., public/private keys) which can be used at all times by the communicating parties (and the network if necessary) for accountability, identifier authentication, and confidentiality.

PINT allows the experimentation with different persistent identification technologies. The framework is oblivious of the particular semantics of the PI, or

the mechanisms attached to it including authentication, resolution, and registration. Consequently, several current technologies can be experimented with. For example, the PI implementation might be hierarchical as in the case of the current Handle System [117] and the Domain Name System (DNS), or flat as in the case of hashes whether self-certifying (e.g HIP [28]) or not (e.g., Chord, Pastry).

- *A novel approach to dynamic and extensible network control-plane service provisioning* using mobile agents; Routing as well as identification are essential network services that provide control plane functionality. The abstraction of each such service is what we have previously referred to as a Ghost (section 3.2.4). Within the PINT framework, Ghosts are implemented as mobile agents, of which we isolate the following:
 - Identification Ghost: This agent is particularly disseminated into the network with the goal of implementing the identification service for the AoI. Managing the namespace including creating, removing, and updating persistent identifiers within the AoI are operations of the identification service which this Ghost implements. The entities within the AoI are oblivious of the actual implementation specifics of the identification service. For example, upgrading the identification service model from a centralized system to a P2P system requires simply upgrading the identification Ghosts within the the AoIs and the upgrade is transparent to the AoI entities. The same is true with the routing Ghost.
 - Routing Ghost: It is similar to the identification Ghost except for its functionality. The routing Ghost implements the actual PI routing protocol that delivers packets to their correct destination(s).

Ghosts¹ may register for providing a discovery service that allows for their

¹Note here that the Ghost is a logical entity, and it might be that both the identification

automatic discovery by other entities within the network. Note that the Ghosts do not represent infrastructural components within the AoI, but instead provide dynamic on-the-fly services for the rest of the entities in the AoI. For example, in an emergency (first responder) network, we envision a set of nodes rapidly forming into an AoI with the necessary Ghosts automatically initializing the AoI and relocating to optimize the network utility. The routing Ghost, for instance, locates a node with Internet connectivity bridging the emergency network to the Internet. Optimizing the placement of Ghosts for maximum network utility is a topic we re investigating in parallel [118].

- *Seamless entity mobility*: directly results from the network being PI-aware. Entities, whether devices, services, or processes can relocate and re-bind while still being reachable on their PI. PINT is generic enough to allow the deployment and experimentation with various mobility mangement schemes.

3.3.1 Components and networking primitives

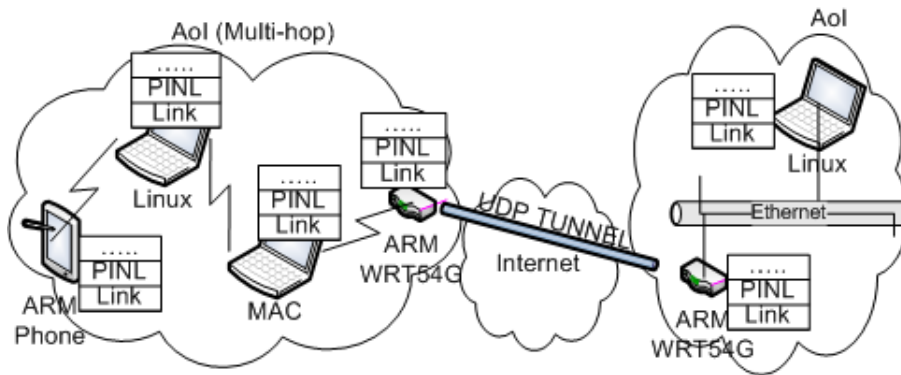


Figure 3.2: PINT components and primitives in sample test-bed showing a multi-hop ad-hoc AoI connected virtually to a traditional ethernet AoI. The PINL layer running on all nodes is able to deliver packets to persistently identified entities.

and the routing Ghosts are implemented as one physical entity.

Figure 3.2 shows the basic components within the PINT framework. First, a set of nodes is abstracted into an AoI. AoIs are allowed to inter-connect either through dedicated links, or through virtual UDP tunnels that abstract the Internet link. Nodes implement the Persistent Identification and Network Layer (PINL) as part of a modified networking stack. Entities attach to the network through PINL, either directly, or through transport layers that can add reliability and/or security to the end-to-end communication. We start by describing the entity identification assumptions and continue to discuss the details of the PINL layer and the primitives and interfaces it exposes to upper layers.

Entity identification

With the proliferation of mobile devices and the anticipated large scale of the network, comes the challenge of how to design a system that is capable of identifying individual entities at a large scale. PINT makes some assumptions in this regard in order to organize entities within the system. First, in order to participate in the system, an entity must acquire a stamped PI, i.e., a PI associated with a stamp. The latter is a credential acquired from a certification authority (CA) to authenticate the owner(s) of the PI. Second, and for scalability reasons, we allow the aggregation of a set of processes into a single entity by assigning a different *type* to each. An aggregated process set appears as a single entity with respect to the rest of the network. Hence, in the case that one of the processes intends to become mobile (for example to migrate), that process must obtain a valid globally unique PI that identifies the process itself. Our *type* analogy is similar to the application port number in current TCP/IP stack and is useful for local demultiplexing.

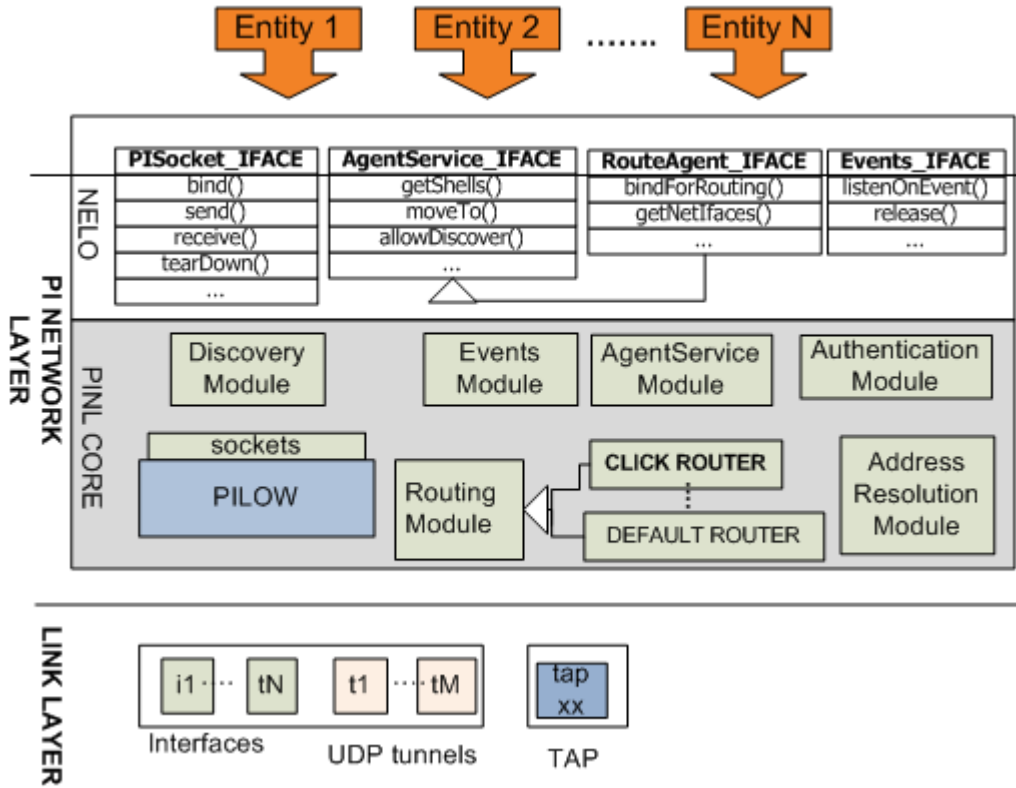


Figure 3.3: PINL layer building blocks

Persistent Identification and Networking Layer

PINL provides the necessary network services to foster the evolution of the network. Services and protocols belonging to this layer mainly handle the initialization of entities within the AoI, and packet delivery between persistently identified entities that may be challenged and authenticated based on their PIs. Presented with a PI, this layer is intelligent enough to deliver a packet to its destination(s). Reliable and/or secure delivery mechanisms are part of a separate upper layer, and motivate an interesting future research effort.

Figure 3.3 shows the componentized architecture of PINL. The PINL layer includes a set of modules that implement the basic layer services, and exposes an

extensible neutralizing interface, which we refer to as the NELO interface², to the upper layers. Entities may directly interface with PINL through the NELO interface. The details of the modules and the interface follow:

- PILOW: The main responsibilities of PILOW is switching incoming requests between modules and maintaining layer state such as PI tables and ARP tables. Additionally, it implements the RouterEngine interface which is employed to route packets to their destination. We ship a default implementation of the RoutingEngine interface in order to provide basic routing functionalities, i.e. routing packets within the AoI. The default routing algorithm may be overridden at runtime when a new entity (a routing Ghost) assumes routing responsibilities through the use of the NELO interface. A more complex router can thus be implemented on-the-fly as we shall see later in the discussion. PILOW additionally implements the PI_Socket_IFACE which provides the traditional socket primitives to entities based on a connectionless transport mechanism that simply demultiplexes incoming packets to resident entities. For example, to use this interface, an entity implements the following code:

```
socket . bind ( pi , type );  
//bind entity with PI 'pi' to a socket  
...  
socket . send ( pi_packet );  
//sends a PIPacket out  
...  
while ( true ) {  
    PIPacket pi_packet = socket . receive ( );  
//listen for incoming packets  
}
```

²NELO stands for Neutral Environment Language for Operation.

Chapter 3. The Transient Network Architecture Instance

- AgentService module: provides service to Ghosts in general. Since Ghosts are abstracted as entities, the Ghost must bind to the layer and authenticate itself before executing. This module implements the AgentService_IFACE, which might be extended to add particular agent functionality. The RouteAgent_IFACE for example extends AgentService_IFACE, introducing functionality specific to routing agents. Through this interface, for example, a routing Ghost can securely bind to PINL overriding the routing service.
- Discovery module: provides a discovery service to Ghosts and entities in general. An agent may invoke *allowDiscover()* on the AgentService_IFACE to enable external entities to discover it. Upon invocation of the *allowDiscover()* function, the discovery module will answer discovery requests destined to the registering agent entity. For example, as we shall see section 3.4, a routing agent within an AoI may assume the role of routing beyond the AoI, hence acting as a default gateway that can be discovered by all the AoI entities.
- Routing module: accepts packets from PILOW for routing, based on PI. Within our framework, the routing service can be easily extended to support various routing implementations. The actual router implementation is determined at runtime for extensibility. A simple device will normally utilize a default router, while a gateway will need a more complex router implementation (e.g., Click router [119]). PILOW is oblivious of the router type and will forward packets to whatever router currently active on the node. The flexibility of this implementation is better explained by introducing a simple example in which a Click router asks PINL to replace the default routing algorithm at run time. Figure 3.4 shows how a Click routing Ghost is able to override the default gateway implementation: 1) the Click entity asks for router binding issuing *RouteAgent_IFACE.bindForRouting* primitive; 2) the PINL AgentService module (see Figure 3.3) will then authenticate the Ghost; 3) when authenticated,

PILOW sets up a `tap` interface through which the PINL daemon and the Click entity communicate.

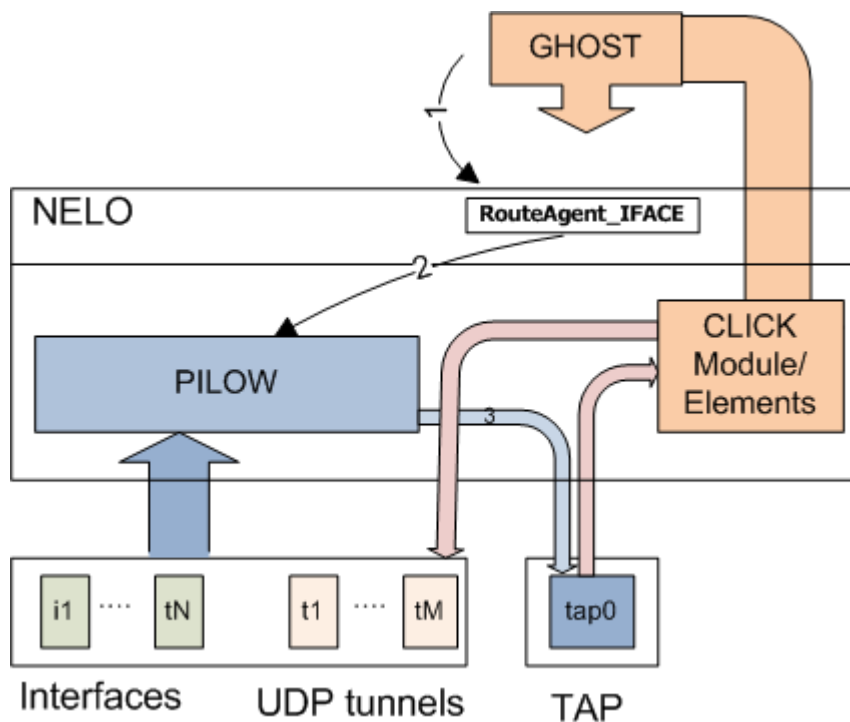


Figure 3.4: Click router asks for agent binding from NELO interface.

- **Authentication module:** The authentication module defines the primitives that allow identifier authentication mechanisms, validates certificates, signatures, etc. The complexity of this module will eventually depend on the actual PI implementation technology.
- **Events module:** enables entities or upper layers to listen on network events through an extensible `Events_IFACE`. The module takes care of propagating registered event callbacks to upper layers (transport protocols or entities),

Protocols

1. Simple Persistent Identification Protocol (SPIP): is the basic networking protocol used for communication. The format of the SPIP packet is illustrated in Figure 3.5. This is the most basic unit of communication that all entities within the PINT framework currently use to communicate. The source and

Bits	0-7	8-15	16-23	24-31
0	Dst. PI Address Length	Src. PI Address Length	Payload Length	
32	Src. Type		Dst. Type	
64	Header Checksum			
⋮	Src. PI Address			
	Dst. PI Address			
	Payload			

Figure 3.5: PI packet format.

destination PI addresses are variable length with a max size of 32 bytes. We allow a variable size identifier to support different implementations of the PI, such as a string (a *handle* in the Handle System implementation [117]) or a hash (HIP [28])³. In order to send a packet, the sender entity addresses the packets to the 2-tuple $\{PI, type\}$ identifier of the destination entity.

Regarding the ARP and the Discovery protocols, we have simply extended current implementations of those and introduced a PI ethernet frame type specific to our implementation.

³Despite the expensive header size, we have deliberately chosen a 32 byte max PI size to allow experimentation with various PI technologies. WSNs, for example, are expected to utilize a significantly smaller PI size.

3.3.2 Implementation Details

The implementation of the Persistent Identification and Networking Layer (PINL) consists of a daemon running at the user-space, and a client library which is used by the entities in order to utilize the functionality provided by the daemon. The layer code is written in object oriented C by exploiting the facilities provided by the portable GLib library [120]. Additionally, a java JNI interface is provided for the NELO to allow Java entities to communicate with PINL.

The C code has been designed with portability in mind, and at the same time targetting embedded devices with very limited RAM and CPU resources like mobile phones, PDAs and routers. In this sense, we have designed the code base to depend only on highly portable libraries such as libpcap, libglib and libgnet which are known to run on Unixes, Windows, Mac OS X and on different architectures like x86, ARM, Mipsel, and SPARC. The PINL has been compiled and tested on the following platforms and devices without the need for patching the code base: Linux (Debian/Ubuntu), Mac OS X, neo1973 using openmoko, n770/n800 using maemo platform, and on the router WRT54G using the openwrt distribution.

The daemon is the fundamental building block of PINL, implementing all the functionality that we have described so far in the previous sections and exposing the NELO interface to the entities. In particular, we have used libpcap to bypass the IP layer both when sending and receiving PI packets; libpcap listens on all the interfaces that are configured and captures all inbound frames that are either PI packets, Discovery packets or ARP packets by inspecting the MAC type field within a frame. The captured frame is then received by the PILOW module by issuing a callback function (see Figure 3.3) that, based on the MAC type, forwards the frame payload to the right module. When a module needs to send a packet on the network it will use the inject feature which is available using the pcap library.

Using pcap as an interface to the link layer is very convenient as we are able to send and receive frames bypassing the IP layer and providing a service that is oblivious of the underlying link layer.

In order to be able to communicate with the PINL daemon the entities must link their code to the *pientity* library. The latter internally utilizes sockets to implement inter process communication. The library exports a very simple API through which the entities can communicate in a transparent way with the NELO interface.

3.3.3 Research Impact

PINT provides a research framework and test-bed for emerging networks such as wireless mesh networks, wireless sensor networks, MANets, as well as traditional networks to inter-connect and communicate beyond the limitations of the traditional Internet Protocol (IP), which was not designed for wireless and mobile environments. Those networks can experiment with a novel persistent identification framework locally and globally, exploiting the novel identification and networking primitives. PINT may either coexist as a deployment on top of the readily available test-beds such as ORBIT [112] to provide the identification framework, or it may be deployed into a separate test-bed for scoped research with persistent identification.

We are currently pursuing several interesting research topics that are based on the TNA architecture. Some of the prominent topics involve:

- Investigating inter-AoI routing implementaions based on PIs; routing based on PIs is a critical research challenge and is essential for our framework to function properly;
- Investigating transport protocols that provide reliability and/or security; currently, as part of the PINL implementation, we provide a simple connectionless

transport protocol that demultiplexes incoming packets based on PI and PI-type combinations. We envision different transport protocols emerging on top of PINL, which can add reliability, congestion control, and security to communication while respecting the wireless and mobile nature of the communication;

- Investigating efficient mobility management schemes.

PINT provides the framework to experiment with the feasibility, efficiency, and scalability of possible solutions to the above topics. A preliminary deployment of mesh/ethernet AoIs is discussed next.

3.4 Mesh/Ethernet Deployment

In this section, we describe a mesh/ethernet deployment over PINT at the ECE department building at the university of new mexico. The goal of the deployment is to validate the operation of the components and primitives rather than to measure their performance. Performance measurements will directly depend on the inter-AoI routing mechanisms, PI technology, and mobility management schemes that we will end up adopting and this is part of our future work.

We setup two distinct multi-hop mesh networks with SSIDs *mesh1* and *mesh2*, respectively, and an ethernet network as shown in Figure 3.6. *mesh1* is comprised of 4 nodes dispersed across the first floor of the building, while *mesh2* is comprised of 4 nodes dispersed across the third floor, and the ethernet network is comprised of 3 nodes deployed in the second floor. As part of each network is a special WRT54G router node that runs the PINL layer. The three networks are connected through the routers with UDP tunnels that traverse the local IP network internal to the building. All the nodes run the PINL layer at the user level on top of an Ubuntu7.04 OS. Within the mesh networks, PINL attaches to the AWDS mesh link state routing

protocol [121] which exports a virtual layer 2 interface (“awds0”). As to the nodes within the ethernet network, PINL attaches to layer 2 through interface “eth0”. A sketch of the complete deployment is shown in Figure 3.6.

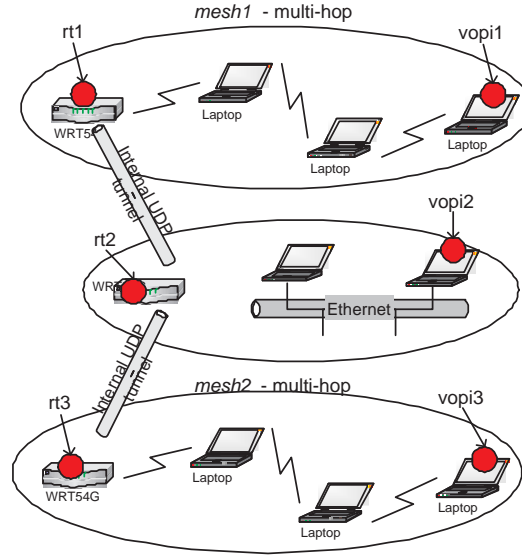


Figure 3.6: Mesh/Ethernet deployment of 3 AoIs (2 mesh and 1 ethernet network). Red circles represent entities.

All nodes employ the default routing engine that ships with PINL, except for the WRT54G nodes that are running a virtualized Click router to handle the inter-AoI PI based routing, i.e., acting as a gateway. The Click entities are represented by *rt1*, *rt2*, and *rt3* in Figure 3.6 Internal AoI nodes use the discovery protocol to discover the gateway, and forward all traffic that is not local to the latter. To know whether a destination PI is local, the default routing engine employs an extended ARP mechanism for local resolution of PIs. Additionally, entities utilize the discovery protocol to proactively announce their presence to the gateway, which in turn maintains soft state about the local network. Finally, and most importantly, the gateways implement a simple PI propagation protocol periodically exchanging their local state. This inter-AoI PI propagation mechanism is not scalable; however, it is just a proof of concept implementation that enables the gateways to locate external

entities and hence, to route inter-AoI traffic correctly. As mentioned previously, part of our current research is targeted at examining efficient and scalable PI propagation, caching, and replication mechanisms.

Aside from the deployment, our experience with the PINL layer particularly shows that PINL is performing as good as UDP/IP over a local mesh network, and that both PINL and IP are constrained by the underlying physical and link layer characteristics.

3.5 Discussion, Future Work, and Conclusion

This chapter introduced the PINT framework developed by UNM and CNRI. PINT exposes to the research community a modular and extensible set of networking components and primitives, which enables novel research and experimentation atop a persistent identification and networking platform. The framework may either coexist as a deployment on top of readily available test-beds to provide a novel identification framework, or it may be deployed into a separate test-bed for scoped research with persistent identification and networking. Aside from investigating the identification, routing, and mobility mechanisms as discussed in section 3.3.3, in the future we hope to enhance our implementation to allow easy deployment of the framework within the ORBIT test-bed [112] and the bridging of external networks to the ORBIT deployment. Briefly, ORBIT is the Open Access Research test-bed for Next Generation Wireless Networks. It is a radio grid (20x20 APs) developed for scalable evaluation of next generation wireless network protocols. The grid allows multiple simultaneous experiments specified using scripts and uses virtualization of APs for that purpose. This is essential for broad participation of the research community especially with the recent bridging of ORBIT and PlanetLab [113]. Finally, for what concerns the implementation of the Persistent Identification Network Layer, we hope to port the code to kernel space by targeting first the Linux operating system once the API be-

Chapter 3. The Transient Network Architecture Instance

comes stable. We would like to note that an initial version of this chapter appeared in [122].

The common practice in the community for evaluating architectural work is through experimentation. Multiple test-beds are readily available for experimentation (such as VINI [113], ORBIT [112], and PlanetLab [123]). In addition, the Global Environment for Network Innovations (GENI) [4] initiative is tasked with creating a global test-bed for experimenting with clean-slate network architectures. As the number of architectural proposals grow, however, the architectural work remains idiosyncratic and descriptions of network architectures are mostly idiomatic. The community generally lacks a unified framework or a taxonomy for thinking about new models and their design implications. Chapters 4 and 5 attempt to add a formalism dimension to the evaluation process, evaluation through formal modeling. Chapter 4 presents a taxonomy of network architectures, while chapter 5 provides a framework for their formal modeling.

Chapter 4

Towards a Taxonomy of Inter-network Architectures

4.1 Introduction

Chapter 2 presented a survey of the diverse and novel Internet architectures by examining their supported service models whether communication-, information-, or computation-centric. Classifying the literature based on the supported service model has helped us understand the underlying goals behind an architectural design, and has additionally highlighted the information model as the main defining element of our taxonomy. Besides, surveying a large set of architectural proposals, we have presented the TNA architecture along with an implementation in chapter 3.

Unfortunately however, the majority of the architectural work remains idiosyncratic and descriptions of network architectures are mostly idiomatic. This current state of affairs is expected to worsen as we start designing and deploying a future Internet, an effort already initiated by NSF's FIND [3], and GENI [4] initiatives. In fact, after surveying the literature, it became obvious that the majority of the recent

architectural work is either aimed at exploring novel usage models that adhere to a class of applications, or at directly addressing a set of limitations of the current Internet ¹. Ostensibly, there seems to be a growing consensus in the community about the need for designing a smarter network that is more than just a transparent “bit-plumbing” medium. While such evolution into a smarter and more complex Internet is bringing new potentials and service models, the community generally lacks a unified framework or a taxonomy for thinking about such models and their design implications.

This chapter presents an attempt towards a taxonomy of inter-networking architectures. We believe that a network architecture taxonomy is a timely contribution that can potentially frame the architectural work, clarifying the problem and the solution spaces. Additionally, such a taxonomy provides a unified framework for networking researchers: (1) to better reason about their work at the architectural level, (2) to clearly compare the different proposals and better understand their similarities and differences, and (3) to explore new dimensions for contributing to the field. Our taxonomy defines a network architecture based on the information model. The latter operates on top of the substrate structure and characterizes the underlying addressing structure, the data objects and the functionality attached to them, and the relative control structure. It is worthwhile mentioning that several classes of our taxonomy may be further elaborated. Additionally, we fully expect that several new classes and properties will be added by other researchers. We would like to note that the current taxonomy is not intended for evaluating the performance of architectures and for determining whether one architecture is better than another. Any such effort would require a thorough understanding of the design space (design parameters, relationships, cost structures, etc.), an effort that we believe is more likely to succeed at

¹Those limitations are mainly the lack of information, security, management, troubleshooting, mobility and QoS support, and the economic conflicts as acknowledged by the community [7, 8, 9].

Chapter 4. Towards a Taxonomy of Inter-network Architectures

narrower scopes than the one at hand. In addition, we would like to mention that the literature is replete with network proposals that correspond to the different taxons discussed throughout the chapter. The examples we provide throughout are solely meant to help the reader assimilate our ideas rather than provide an exhaustive list of the related work.

As we started studying the taxonomy problem, it seemed that the body of network architecture work is difficult to classify due to the independent nature of the many contributions to the field. However, we have noticed that modern networks are becoming increasingly intelligent, and the intelligence is being manifested by introducing more processing [124] and storage elements [125], and by providing the users with richer instruction sets instead of the simple static IP packet. Interestingly, such evolving network architectures resemble the computer architecture field, in the sense that a network architecture is currently being designed to provide a general purpose computing platform to its diverse users. Consequently, it is our belief that the modern network architecture and the computer architecture converge conceptually at the architectural level, despite the fact that they significantly diverge otherwise, primarily due to the distributed and large-scale nature of network architectures. We shall leverage this idea to directly apply some useful taxonomical notions from the computer architecture field to our work, particularly from [126, 127].

The remainder of the chapter is structured as follows: building on the service-model perspective of chapter 2, a taxonomy based on the information model is then discussed in section 4.2. We demonstrate the descriptive power of the taxonomy by applying it to the Data-Oriented Networks Architecture (DONA) [34] in section 4.3. Related work is then presented in section 4.4 before concluding with a discussion of the value and limitations of our work in section 4.5.

4.2 Taxonomy

Our taxonomy is based on the network's information model, as it aims to clarify the following questions:

- What are the types of data objects recognized by the network?; and
- How does the network operate on those objects? In other words, how is the network capable of manipulating the objects?

Towards this end, the taxonomy defines architectures starting with the underlying substrate structure (the topology, the functional units, and their interconnection structure) over which the information model operates, and ending with the information model itself (addressing structure, types of data objects, and control structure).

4.2.1 Substrate Structure

The network substrate is comprised of the underlying physical network elements over which the information model is defined. The substrate structure, hence, describes the network topology, the functional units, and their interconnection structure.

Topology

Assumptions regarding the inter-network topology are crucial to our analysis. We assume the inter-network is composed of *zones*. A zone forms an autonomous part of the inter-network and represents a logical region with explicit boundaries. We intentionally define the notion of a network zone to be abstract enough to encapsulate the various definitions proposed in the literature, including the Internet Autonomous Systems (AS), Contexts in Plutarch [89], Turfs in Turfnet [90], and so on. The zone

has an explicit “boundary”, a logical construct that can take various forms, such as administrative, physical, protocol, or even social boundaries. Within the remainder of the discussion, the notions of “global” and “local” are to be interpreted relative to the zone. For example, a global function (examples of functions are addressing, naming, and forwarding) is one that operates across zones, whereas a local function is to be interpreted as zone-local.

Of particular interest to our taxonomy are the following topological properties:

- *Structure*: is an important property that can take any of the values: *hierarchical*, *flat*, or *special* (e.g., ring) topology. An inter-network that is composed of hierarchical zones will topologically include a root set of zones, generally referred to as “Tier-1”. A flat topology on the other hand does not necessitate a topological root.
- *Composition*: The topological structure depends on how the zones are composed. Composition can take three forms as follows: 1) *controlled-overlap* means that part of the topology is shared by multiple zones, 2) *integration* is when one zone subsumes another resulting in an integrated data/control plane for the composed zone (sometimes referred to as horizontal composition), and 3) *direct peering* is when zones, generally heterogeneous, directly connect through dedicated elements (sometimes referred to as vertical composition). Note that from a physical viewpoint, direct peering encapsulates the Internet AS relationships, whether customer-provider, or peering.

Components and Interconnections

We isolate the following components types or functional units:

- Storage Elements (SEs) which may be of two types:

Chapter 4. Towards a Taxonomy of Inter-network Architectures

- *Memory Elements* (ME) are abstract elements that store information within the network, such as content servers/providers; and
- *Cache Elements* (CE) are memory elements that provide faster access to their information, either by being physically closer to the user and/or because they are connected to the user by a higher bandwidth link than the ME. Examples include proxies or caches used in content distribution networks (CDNs).
- Processing Elements (PEs) which perform information processing and may be further divided into:
 - *Data Processing Element* (DPE); and
 - *Instruction Processing Element* (IPE).

More details on instructions and data are presented later in the information model. However, for now, one may envision an IPE instance to be a router, or a proxy that operates on packets ², while a DPE instance might be a content transcoding element inside the network.

- Switching Elements (SWEs) are abstract elements that switch information between SEs and PEs.

Having introduced the abstract component types, we proceed to describe their properties that are of interest to this taxonomy, as follows:

1. The *Dispersal* property/factor is specified for each of the above component types. It describes the required physical distribution/placement of an element type relative to the topology, with a number n to mean one element (or a constant set of elements) per n zones. Values for n may be: 1 (to mean an element exists for each zone), k (to mean an element exists for a group of k

²A packet is a form of a static instruction.

zones, such as a Tier-1 ISP provider hierarchy or the set of edge domains on the Internet), and Z (to mean an element exists for all the zones, such as in the case of a centralized global service), where Z is assumed to be the total number of zones in the topology.

2. The *Interconnection* property describes the logical interconnection structure among the component types. Two combinations of element interconnections

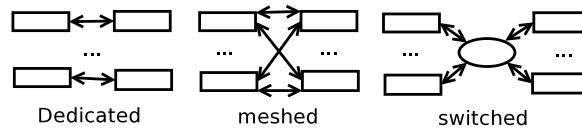


Figure 4.1: Interconnection types; A square represents an abstract element (SE or PE), while an ellipse represents a switching element (SWE).

are of interest to us, mainly those specifying the $PE - PE$, and the $PE - SE$ element interconnections. The different types of interconnections are depicted in Figure 4.1, and those may be: 1) *dedicated* to mean that the i^{th} component of the first type is connected to the i^{th} component of the second type; 2) *meshed* to mean that every component of the first type is directly connected to every component of the second type; and 3) *switched* to mean that a switching element connects components of the first type to those of the second type.

Consequently, the tuple $(Dispersal, Interconnection)$ fully describes the component interconnection structure. Additionally, it directly relates to scalability by exposing the bottleneck infrastructure elements. We briefly present some examples related to the current Internet substrate structure to better illustrate the aforementioned properties. Internet routers are IPEs (that process implicit forward instruction) with dispersal factor $n = 1$, and for which their IPE-IPE interconnections is meshed. DNS infrastructure elements, and particularly domain DNS servers are IPEs (that process resolve instructions) with $n = 1$ and may simultaneously be CEs (that cache

query results) and MEs (that serve the domain's zone files) with $n = 1$. The DNS root servers, however, are MEs (root database) with $n = Z$ ³. Additionally, the IPE-ME interconnection is generally switched since resolutions have to pass by the root servers that act as the switch between the IPEs and the MEs.

4.2.2 Information Model

The information model is defined based on three classes of data objects that encapsulate the information abstractions recognized by the network. At the core of the information model is the notion of data objects (alternatively entities) that are bound to and accessed from network 'locations' relative to some addressing structure. Consequently, before delving into the details of the data objects, we discuss the first defining element of the network information model, namely *the addressing structure*.

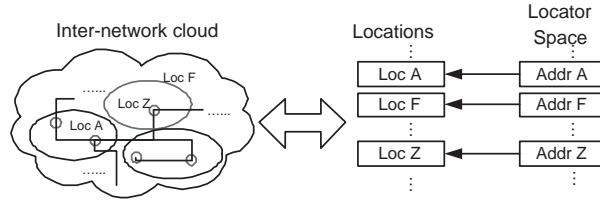


Figure 4.2: Abstracting network locations (red circles) and visualizing a locator space.

Addressing Structure

We briefly introduce the concepts of locator and identifier in this chapter, and we shall elaborate on these concepts later in chapter 6 to eliminate the confusion that

³Assuming not replicated.

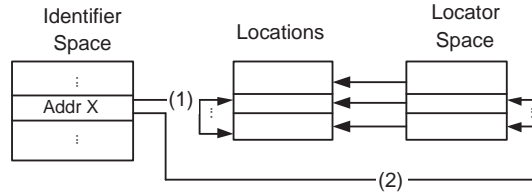


Figure 4.3: Identifier space is either integrated (1), or disconnected (2).

generally arises when discussing them. The information model starts by assuming the existence of *locations* within the network, where the term ‘location’ may have topological or geographical connotation. When the locations are addressable, we obtain the notion of the network *locator* belonging to the *locator space*. The latter consists of all the possible addresses of all the addressable locations on the network and is denoted by \mathcal{L} . Consequently, the locator is defined as a location identifier. Figure 4.2 depicts our visualization of the locations and of the address space constructs. Examples of Addr A in Figure 4.2 may be an IP address, a path, a set of coordinates, etc. (as long as there exists an underlying control that can link the locator to the network location it points to).

On the other hand, when the objects on the network are being addressed, we obtain the notion of the *identifier* belonging to a *identifier space*, \mathcal{I} . Some examples of systems that instantiate identifier spaces are naming/directory systems, metadata registries, and trackers. According to our definitions, there is a conceptual difference between the locator and the identifier in terms of what is being identified. The latter generally identifies some high level information abstraction (such as a host, or a content object) in contrast to identifying a location with the former. Data objects, which we shall characterize shortly, are always bound to the locations, and hence every access to an object on the network will require an address (locator or identifier) to succeed. Throughout the rest of the discussion, an *address* is to be interpreted as either a locator or an identifier unless otherwise specified.

As far as the taxonomy is concerned, the following set of properties characterizes the addressing structure. The first property, *address spaces*, describes whether the addressing structure explicitly defines and makes available as part of the *ISA* a locator space (\mathcal{L}), an identifier space (\mathcal{I}), or both.

- **Locator:** Solely providing a \mathcal{L} implies that objects are only addressable (and generally accessible) by location. In addition to necessitating prior knowledge of location, this model falls short of supporting object binding volatility (such as in the case of mobility, re-homing, and disconnections). The IP addressing architecture [2] is one example in which only the IP locator space is made available as part of the *ISA* addressing structure.
- **Virtual:** On the other hand, solely providing an identifier space implies that only objects are addressable and not locations. Since no locator space is provided, there is no embedded notion of location on the network from the user's perspective. In this sense, the identifier space is directly *integrated* with locations, i.e., an access to an identifier will automatically result in accessing the location(s) to which the identifier points. Figure 4.3 (1) illustrates this addressing style. Some example architectures that support this style are [37] and [95].
- **Both:** When both spaces are provided, it is necessary to characterize their relationship defined with *space-correlation*, as follows:

- **Independent:** $\mathcal{I} \cap \mathcal{L} = \emptyset$

This is the general case of current addressing architectures in which the spaces are semantically and syntactically independent, and are only related through the mapping/search function. Examples of \mathcal{I} could be a space of flat hashes (e.g., DHT approaches) or human-readable strings (e.g., DNS) which is independent of an underlying locator space (e.g., IP,

or topology labels in labeled compact routing [44]).

– **Correlated-partitioned:** $\mathcal{I} \subset \mathcal{L}$

In this model (and the following one), the spaces are consolidated (generally syntactically) and the semantic distinction between the spaces is made statically (i.e. known a priori) or dynamically (i.e. at runtime). The model has the feature that the mapping function from locators to locations (otherwise referred to as routing) is inherently aware of \mathcal{I} and could be reused for identifier translation. An example here is Mobile IP [27].

– **Correlated-embedded:** $\mathcal{L} \subset \mathcal{I}$

If we are allowed to think of the IPv6 space as an identifier space, then an example of this model would be the embedding of the IPv4 space in the IPv6 space.

– **Partial-overlap:** $\mathcal{L} \cap \mathcal{I} \neq \{\emptyset, \mathcal{L}, \mathcal{I}\}$

We are not aware of any addressing architecture that supports this model.

Second, for each of the locator and the identifier spaces, two structural properties are defined:

- *Space structure:* may be **hierarchical**, **flat**, or **special**; and
- *Addressing scope:* defines the scope over which the address is valid. Values for scope are based on the topology structure defined in the previous section, and those include: **local** (per zone), **global** (to all zones), and **partial** (to a set of zones).

To gain a better insight into the taxonomy's descriptive powers, consider the following examples of addressing structures described with the properties just introduced:

1) *locator space* is (*hierarchical, global*): The Internet and NIRA [99] are two architectures that explicitly rely on a global, strictly hierarchical addressing scheme. Some of the advantages of this scheme are scalable routing and small routing table sizes ⁴; 2) *identifier space* is (*flat, global*): ROFL [37] and SFS [31] are architectures that utilize this flat, DHT-style addressing scheme. Some of its advantages include semantic-free, flat, and location independent-addressing. On the other hand, some of the disadvantages include global maintenance overhead, consistency issues, and scalability concerns; and 3) *Either space* is (**, local*) ⁵: This is the case for example when each network has its own private address space. Plutarch [89], Turfnet [90] are some examples. Some of the advantages of such mode are provider-independent addressing and easier re/multi-homing. However, some disadvantages include extensive translation, complex routing, and larger global routing tables.

Having discussed the addressing structure, we proceed to identify the different classes of network data objects that comprise the second defining element of the information model.

Data Objects

The data objects are characterized by the ordered tuple (C, S, F) , where C denotes the object class, S denotes the scope or context within which the object is meaningful, and F denotes the set of functions applicable to the object. We isolate three classes C of data objects ⁶: 1) *primitive objects*, 2) *group objects*, and 3) *complex objects*, and their respective functions as follows:

⁴Scalability depends on efficient address aggregation, however. The de-aggregation practice on the current Internet's routing system has driven the latter to be unscalable [12].

⁵The symbol '*' is the wildcard character.

⁶Recall that we are solely concerned with the internal network information model, and hence the end-to-end data abstractions which are transparent to the network are irrelevant to this taxonomy.

Primitive Data Objects - are further categorized as either *carrier objects* or *consumable objects*. The former set represents the information carriers that are stored or processed within the network but are generally not consumed, i.e., neither bound to nor accessed from locations. We have identified the following set of carriers: 1) locator; 2) identifier; 3) instruction, represents a functional expression to be executed by the network. Instructions may range in their functional expressiveness. For example, the IP packet is an instance of an instruction data object that is inherently static, i.e., the packet implicitly instructs the network to deliver a payload from source to destination. On the other hand, [93] recognizes a more expressive instruction set for dynamic service composition; 4) Data Unit, is the unit of communication and could include control and data; 5) stream, is an aggregation of Data Units; and 6) status block, may be of different types. A status block encapsulates the internal network state as well the status of operations performed within the network.

As to the consumable objects, those represent data objects that are explicitly bound to the network locations and consumed from their locations. The following two styles of consumable objects are identified:

- Raw Information Bit Stream (RIBS): represents an untyped consumable data object that appears to the network as a bit stream. All RIBS objects must be self-descriptive, i.e., knowledge about the RIBS data (e.g., typing, and interpretation) is encapsulated within the data itself. The Internet, for example, solely supports RIBS objects and is hence transparent of information types (i.e. all typing intelligence is end-to-end);
- Typed Abstract Content Object (TACO): represents any typed consumable data. Interpretation of TACOs is generally part of the network's control structure. Several examples of possible typed content objects may be recognized from today's applications, including: static and dynamic content (such as files and web pages), continuous content (such as multi-media or live sensor feeds),

interactive content (such as the case with online gaming), and metadata objects.

The object scope S defines the scope within which the object is valid, taking values: *local*, *global*, or *partial* relative to the substrate structure.

Several classes of functions F are applicable to primitive data objects. While we overview these classes, we simply focus on characterizing the binding, access, and transfer functions that the network makes available for manipulating the consumable data objects.

- **Binding Functions:** In its simplest form, binding is the process of assigning a data object to some location on the network. Assume X and Y denote network addresses, we isolate the following forms of binding:
 - **Direct-value binding** has the form $X = value$. Binding a data object to an IP address, or a host join in [37] are some examples;
 - **Processed value binding** has the form $X = f(.)$. Some processing is performed before assigning the object. For example, $f(.)$ might be a query whose result is assigned to an address;
 - **Multiple-value binding** ⁷ has the form $X = Y$. The data object in Y is replicated to X . In general, this form of binding requires some form of COPY/MOVE instruction as part of the architecture's *ISA*. Data replication as in the case of CDNs is an instance of such binding style;
 - **Shared-value binding** ⁸ has the form $X = \&Y$ to mean that X points to the same data object as Y . Mobile IP [27] is an example here.

⁷This is similar to assignment by value.

⁸This is similar to assignment by reference.

While instances of such binding styles are present in the literature and were noted above, the majority of the styles are still not explicitly supported by the current architectures' instruction sets.

- **Access Functions:** Accessing information on the network may be characterized by the following properties: 1) *access type* specifies whether primitive, group and/or complex object access is supported; 2) *access paradigm* specifies **synchronous** and/or **asynchronous** access. For example, publish-subscribe architectures provide an asynchronous access paradigm; 3) *access mode* dictates whether **read**, **write**, and/or **read-modify-write** are supported; ; and finally 4) *addressing mode* which is characterized with:

- **Direct/absolute addressing** (locator and/or identifier): The absolute address of the object to be accessed is known, whether locator (e.g., IP addressing) or identifier (e.g., [90]).
- **Indirect addressing (locator and/or identifier)** represents the indirection style addressing, where the absolute address of the object is unknown, but an alternative address (pointer) is used for indirect access. Locator-to-locator indirection (e.g., [27]), and identifier-to-locator indirection (e.g., [29]) are some flavors of indirect addressing.
- **Associative addressing** (locator and/or identifier) is analogous to intentional addressing, in which the sought object's address is unknown, but some of its attributes are known and are employed for addressing. Distributed searching, whether in the locator space (e.g., [89]) or the identifier space (e.g., [92]), is generally utilized to locate the objects of interest.
- **Group addressing** (locator and/or identifier) involves addressing a group of locations (e.g., geocast and multicast addressing), or a group of objects. Addressing a group of objects is equivalent to addressing a group object type (to be discussed shortly).

- ***K*-preference addressing** (locator and/or identifier): is similar to group addressing except that k elements of the group are addressed instead of the whole group. Anycast addressing, for example, is a special case of this mode in which $k = 1$ and the preference is ‘any’ (e.g., IP anycast, [34]).
- **Transfer/Delivery Functions:** Two properties of the delivery function are essential to this taxonomy, while several others may be deduced from other properties of the information model (such as the addressing modes). First, the *information recognition* describes whether the delivery is cognizant of information types. For example, delivery of continuous content objects (such as multi-media stream) requires time-sensitive transport mechanisms to preserve the real-time nature of the data. Second, the *transfer multiplicity* denotes the multiplicities at both ends of the transfer pipe and can take the following forms starting with the multiplicity of the information source:
 - **1-1** is single source, single destination transfer similar to unicast delivery;
 - **1-N** is single source, multiple (or group) destination transfer similar to multicast delivery; and
 - **N-1** represents multiple source delivery as is the case with swarming architectures (e.g., Bittorrent and USwarm [92]);
 - **N-N** is a multiple source, multiple destination delivery model. This model is probably the most intriguing. An example of such model would be a swarm-like information distribution to a multicast group.

The rest of the functions apply to primitive data objects in general. We simply distinguish those and we leave their characterization for a future work. To start with, **Transformation Functions** convert the data objects from one representation to another. Some examples include interstitial functions [89], and NAT boxes that perform address/instruction/protocol translation, and transcoding. **Decoding**

Functions interpret the data objects and generate control vectors as a result. Interpretation of the data objects follows from their representation. On the current Internet, for example, every router interprets the instruction in the same way and generates a forwarding control vector that determines the next hop. **Construction Functions** produce new objects and their values. Information fusion/integration, aggregation, joining, splitting etc. are some examples of manufacturing functions. Finally, **Status Functions** get/set the various status blocks within the network, whether those involve internal network state or operation status.

Group Data Objects - A group data object is a collection of primitive objects that generally share some properties such as their type or their access control/policy. The constituent elements of the group belong to the same address space. While the group as a whole is an addressable entity, its constituent elements might not be individually addressable. Elements are identified by a combination of the group object identifier and the element identifier within the group. Group addressing is thus required for group object access. One example of a group object on the current Internet is the multicast group. Additional functions that apply to group data objects include: creating the group object, adding elements (group joining), removing elements (group leaving), and removing the group object.

Complex Data Objects - Complex data objects are simply data structures that the architecture makes explicit. It is intuitive that such complex data structures will emerge in the future, but it is hard at this stage to anticipate their properties. One may envision an explicit distributed stack data structure for example that is tailored to some architecture with an explicit push/pop usage model.

The third and final defining element of the information model is *the control structure* which defines the underlying control to support the information model. Almost every aspect of the information model discussed so far requires its dedicated control protocols and algorithms. For example, control for mapping characterizes the control

required for mapping from identifier/locator spaces to location, and for maintaining the pointers. Control for data access defines the control to support the addressing model and modes, etc. Clearly, such control structures (and others) represent a significant body of the networking research, where each aspect stands alone as a research topic by itself. Consequently, characterizing the control structure is beyond the scope of this dissertation and is left for a future work.

4.2.3 Towards a complete taxonomy

Our approach towards a complete network architecture taxonomy is syntactically defined (using a BNF metasyntax) in Table 4.1. We have decided to represent the taxonomy textually rather than graphically since the textual representation is clear and compact. We clarify the following notation: ‘,’ means concatenation; (x, y) means grouping in which terms x , and y are separated by any whitespace character; $\{x\}$ means a set of elements of x ; $< x >$ means term x is left unspecified; and $[x]$ means optional term.

<i>network_arch</i> :=	(‘ARCH’, id, ‘begin’, <i>substrate_struct</i> , <i>info_model</i> , ‘end’)
<i>substrate_struct</i> :=	(‘SUB_STRUCTURE’, ‘begin’, <i>topology</i> , { <i>component</i> }, { <i>interconnection</i> }, ‘end’)
<i>topology</i> :=	(<i>top_struct</i> , <i>top_composition</i>), “,”
<i>component</i> :=	([id], <i>component_type</i> , <i>dispersal_factor</i>), “,”
<i>interconnection</i> :=	([id], <i>ic_type</i> , <i>ic_link</i>), “,”
<i>top_struct</i> :=	“hierarchical” “flat” “special”

Chapter 4. Towards a Taxonomy of Inter-network Architectures

$ \begin{aligned} top_composition &:= \text{"controlled_overlap" "integration" "direct_peering"} \\ component_type &:= \text{"SE" "ME" "CE" "PE" "IPE" "DPE" "SWE"} \\ dispersal_factor &:= \text{"1" "k" "Z"} \\ ic_type &:= \text{"PE-PE" "PE-SE"} \\ ic_link &:= \text{"dedicated" "meshed" "switched"} \\ id &:= \text{character, \{character digit "_" \}} \end{aligned} $
$ \begin{aligned} info_model &:= (\text{'INFO'}, \text{'begin'}, addr_struct, \{data_type\}, \\ &\quad <control_struct>, \text{'end'}) \\ \\ addr_struct &:= (\text{'ADDR_STRUCT'}, \text{'begin'}, loc_space, id_space, \\ &\quad both_spaces, \text{'end'}) \\ data_type &:= (\text{'DATA'}, \text{'begin'}, data_class, \{function\}, \text{'end'}) \\ \\ loc_space &:= (\text{'LOC_SPACE'}, space_structure, addr_scope), \text{";" } \\ id_space &:= (\text{'ID_SPACE'}, space_structure, addr_scope), \text{";" } \\ both_spaces &:= (\text{'BOTH_SPACES'}, space_correlation), \text{";" } \\ data_class &:= (class_type, \{data_object\}) \\ function &:= (binding_fcn access_fcn transfer_fcn) \\ &\quad search_fcn), \text{";" } \\ \\ space_structure &:= \text{"hierarchical_addr" "flat_addr" "special"} \\ addr_scope &:= \text{"local_scope" "global_scope" "partial_scope"} \\ space_correlation &:= \text{"independent" "partitioned" "embedded" "overlap"} \\ class_type &:= (\text{"primitive" "group" "complex"}, \text{";" } \\ data_object &:= (\text{"locator" "identifier" "instruction" "data_unit" } \\ &\quad \text{"stream" "status_block" "RIBS" "TACO" } \\ &\quad group_object complex_object, object_scope), \text{";" } \end{aligned} $

<i>binding_fcn</i>	$:=$	(“FN_BINDING”, { <i>assign_mode</i> })
<i>access_fcn</i>	$:=$	(“FN_ACCESS”, <i>access_type</i> , <i>access_paradigm</i> , { <i>access_mode</i> }, { <i>addressing_mode</i> })
<i>transfer_fcn</i>	$:=$	(“FN_TRANSFER”, <i>info_cognizant</i> , { <i>s2d_multiplicity</i> })
<i>group_object</i>	$:=$	‘group_’, id
<i>complex_object</i>	$:=$	‘complex_’, id
<i>object_scope</i>	$:=$	<i>addr_scope</i>
<i>assign_mode</i>	$:=$	“direct-value” “processed-value” “multiple-value” “shared-value”
<i>access_type</i>	$:=$	{ <i>class_type</i> }
<i>access_paradigm</i>	$:=$	“synchronous” “asynchronous”
<i>access_mode</i>	$:=$	“read” “write” “read-modify-write”
<i>addressing_mode</i>	$:=$	“direct_locator” “direct_identifier” “indirect_locator” “indirect_identifier” “associative_locator” ‘associative_identifier’ “group_locator” “group_identifier” “k-preference”
<i>info_cognizant</i>	$:=$	{ <i>class_type</i> }
<i>s2d_multiplicity</i>	$:=$	“1-1” “1-N” “N-1” “N-N”

Table 4.1: A BNF syntax for taxonomical specification of network architectures.

4.3 Applying the taxonomy

To illustrate the applicability of our taxonomy in terms of its classification powers, we have applied it to a rather special network architecture, the Data Oriented Network

Chapter 4. Towards a Taxonomy of Inter-network Architectures

Architecture (DONA [34]) ⁹. The textual description is listed below. In the listing below, ‘%’ stands for comment, “NA” means the term is irrelevant.

```
ARCH DONA
begin
  SUB_STRUCT begin
    % -topology structure
    hierarchical NA; %Internet ASes
    % -components
    RH IPE 1; %Resolution Handlers
    ROUTER IPE 1; %traditional BGP routers
    PROVIDER ME k; %content providers
    CACHE CE k; %content caches, extended RH
    % -interconnection structure
    PROVIDERS ME-ME meshed;
    %exploits hierarchical topology
    RH_RH PE-PE meshed;
  end
  INFO_MODEL begin
    ADDR_STRUCT begin
      % -addressing structure
      LOC_SPACE hierarchical_addr global_scope;
      %IP addressable locations, but global
      % addressing is not necessary
      ID_SPACE flat_addr global_scope;
      %HIP style identification
      BOTH_SPACES independent;
```

⁹DONA’s description is based on our understanding of the architecture, which may well be incomplete.

Chapter 4. Towards a Taxonomy of Inter-network Architectures

```
end %ADDR_STRUCT end
DATA begin
% -data type(s)
primitive:
  locator global; %IP, or maybe src route
  identifier global;% name is (P:L) tuple
  instruction global; %find,lookup packets
  data_unit global; stream global;
  RIBS global; % datum-metadata, service
    %e2e type intelligence
% -functions
FN_BINDING direct-value;
    %REGISTER(.) func
FN_ACCESS primitive:group:
  synchronous
    direct_locator direct_identifier
    k-preference; %anycast FIND(.) func
FN_TRANSFER NA 1-1 1-N; % for multicast
    % FIND required
end %DATA
end %INFO end
end
```

A significant amount of knowledge about the architecture is conveyed by simply observing such a compact taxonomical representation. Additionally, architectures are easily compared along their convergence and divergence points by observing their respective representations side by side. For example, it is easy to notice the significant similarity, from our taxonomy point of view, between DONA and TRIAD [88] by

representing the latter. Aside from the differences in terms of the control structure and name semantics which we do not consider in the taxonomy, their main other difference is the identifier space structure.

4.4 Related Work

The major differentiator of our work is its generality in understanding networks at the architectural level rather than being confined to the communication/switching properties, or to the computational properties, or to particular scoped network architectures that focus on naming, or routing, or content delivery. This section overviews the related work. To start with, some recent work has focused on creating a taxonomy for overlay (or virtualized) networks, relative to the current Internet. Clark et. al [128], presents a taxonomy of overlays that helps thinking about their motivations and their implications. Augusto [129] classifies networks based on their application-specific or purpose-specific nature. Moreover, [130] presents a simple taxonomy of Network Computing (NC) systems (or overlays) based on their applications, platforms, and management. Additionally, classifying a particular type of overlay, the Content Delivery Networks (CDN), has been the subject of some recent work [131, 132]. Again, while all this work is related (and complementary) to ours, our work addresses the general architecture classification problem.

Other recent work has focused on modeling and reasoning about the communication aspects (mainly switching and binding properties) of networks [24, 22]. Such network modeling work is complementary to ours in trying to better understand and formally reason about the network architecture space.

Classifying programmable networks has also been addressed in the literature [86, 87]. Reference [86] provides a generalized model for programmable networks that explicitly includes a computational model relevant to such networking environments.

In the same way, our general service model perspective acknowledges a computational model as a building block for modern network architectures.

4.5 Discussion: Value and Limitations

The main contribution of our work is a taxonomy that helps organizing and thinking about the architectural space. The taxonomy is based on a bottom up characterization of a network starting with the underlying physical substrate (the topology, components, and their interconnections) and ending with the information model (the addressing structure, the data objects and the operations allowed on them, and the control structure). In terms of value, in addition to offering a comprehensive overview of the set of architectural possibilities as well as a tutorial for introduction to the field, our taxonomy helps organizing and thinking about the architecture space beyond the communication model (switching/delivery characteristics of networks) which has been the major approach adopted in the literature for taxonomizing architecture (as we have seen with connection-oriented vs connectionless models). The latter approach has weak discriminatory power and hence fails to distinguish the different (and particularly modern) architectures as to their information structures. Our taxonomy additionally helps in highlighting gaps in the design space for exploring new contributions to the field by identifying unexplored research areas. Examples of some gaps that were highlighted include the *ISA* support for binding models and addressing modes, the N-N delivery, and address *space correlation*. Moreover, the descriptive nature of our taxonomy helps in comparing modern network architectures along their convergence and divergence points. Finally, the taxonomy helps set the stage to for attempting to answer the question of whether intelligence in the network is useful, and what is the minimal set of functionality that could be part of an architecture while maintaining the elegance of end-to-end design. In this regard, we lack

the expertise necessary to take any position in trying to answer those questions.

In terms of its limitations, the chapter in its current form falls short of providing tangible outcomes beyond the descriptive one. Additionally, by no means do we claim that our taxonomy is complete. The taxonomy does not provide characterizations for the control structure, for security in general, and for the timeliness of information. Each of these missing structures spans multiple aspects of the information model, and their treatment is left for a separate work due to lack of space. Despite those limitations at this point, which we plan to address in a future work, we believe that this chapter is helpful to the community.

4.6 Conclusion

In chapters 2 and 4, we have seen that while the communication structure is necessary for defining and representing a network architecture, it is not sufficient. Information and computation structures are building blocks that need to be properly understood within modern network architectures. This chapter presented a classification model that is descriptive in nature and that helps in framing the solution space, and in finding similarities and differences among architectural designs. We would like to note that an initial version of this chapter appeared in [133]. Chapter 5 presents a design methodology for formally describing and reasoning about network architectures and architectural styles. The framework enables researchers to better represent, analyze, reason about, and infer important properties about their architectures.

Chapter 5

Towards Formalizing Network Architectural Descriptions

5.1 Introduction

Despite the rich literature on network architecture and communication system design, the current practice of describing architectures remains informal and idiosyncratic. This was caused by the evolution of a semantically rich terminology that has been adopted by network architects over time. The terminology, despite being informal, reveals a lot of architectural information and has so far enabled efficient communication between architects. This scenario is very similar to the evolution of software architecture modeling in the context of software engineering [20]. This state of affairs has however, led to the overloading of architectural terms, and to the emergence of a large body of network architecture proposals with no clear understanding of their cross similarities, compatibility points, their unique properties, and architectural performance and soundness.

Several models for communication systems have been recently proposed, some of

which are focused on particular communication aspects such as binding [21, 22] or routing [23]. Others [24, 25] are more general, and concern themselves with multiple communication aspects such as forwarding, naming, addressing. It is important to notice however, that the formal modeling and representation of network architectures is fundamentally different from that of communication systems. In fact, while the communication structure is necessary for defining and representing a network architecture, it is not sufficient. In addition to the communication structure, information and computation structures are building blocks that need to be properly understood within modern network architectures. Communication systems tend to share the same set of elements and are generally concerned with switching properties of networks and their associated communication and control primitives. On the other hand, network architectural descriptions are concerned with high-level architectural abstractions, their interactions, their structural and behavioral properties, and the constraints and invariants that define each architecture.

Towards formalizing network architectural descriptions, we utilize concepts relevant to architectural *style* modeling. An architectural style¹ is a family of network architectures that share a common representation vocabulary. Hence, while architectural *instances* specializing a particular style may vary in their particulars, their overall structure remains the same and obey the general style constraints. There are significant advantages associated with architectural style design. Those include a better overall system understandability by defining a precise common design vocabulary, the availability of design re-use among all instances of a class, architectural interoperability, and specialized analysis of a class of architectures by constraining the design space [55]. This chapter presents a design methodology for formally describing and reasoning about network architectures and architectural styles. The methodology is demonstrated by detailing a formal model for the FARA [30] fam-

¹Architecture *style* (or *pattern*) is a term commonly used in the software engineering field [20].

ily of network architectures. Our work provides a framework for network architects to formally group various architectures into a set of styles based on their common structural and behavioral characteristics, enabling researchers to better represent, analyze, reason about, and infer their important properties.

The rest of the chapter is organized as follows: Section 5.2 presents the necessary background related to architectural styles and to the language *Alloy*. Alloy is a simple declarative language based on relations and first-order predicate logic and is the language that we shall use throughout the discussion for formal modeling and verification. Section 5.3 details our approach through a case study of the FARA [30] class of network architectures. Section 5.4 then discusses the related work. Finally, we present a discussion of our approach and our current and future work towards formalizing network architectural descriptions in section 5.5, before concluding in sections 5.6.

5.2 Background

5.2.1 Architectural Styles: What and Why?

Software architectures are usually viewed as a set of interconnected elements that define the structure of a system. The elements are mainly components (computational and storage elements) and connectors (interactions among the components). For example, in a client-server architecture description, one might model the client and server elements as components and an RPC communication protocol between them as a connector. An architectural *style* represents a family of architectures that share a common structural organization. Despite the different representations of a style [51, 54, 56, 20], it is typically composed of component/connector types, and a collection of constraints on how the types are combined. Associated with a style are

a design vocabulary, an underlying computational model, and invariants [20].

Styles may be treated as stand-alone structures and may be related through inheritance, or composition. *Inheritance*, an extremely attractive property for describing architectural styles is the ability of a sub-style to extend one or more super-styles inheriting their structural properties, vocabulary, and constraints/invariants. *Composition* is another form relating multiple styles. The composed style is an aggregation of the vocabulary, structure, and constraints of the its constituent styles. Generally, the composed style introduces a new structure to relate the constituent styles together.

The advantages of *modeling* architectural styles are several (check [55]). First, given the abstraction level of an architectural style, it is generally hard to verify properties pertaining to the style or even to implement the style itself. A compact model then allows the verification of a style's structural and behavioral properties over constrained instance sets without having to actually implement the style. This is an important step when applied prior to the actual instantiation of a complete architecture from the style. In other words, a formal model helps the transition from abstract style design to actual instantiations. Additionally, claims of compatible network architectures, whether those pertaining to general architectures, or to scoped architectures (such as naming, addressing, or routing) may then be logically verified. Finally, a formal model helps to classify the literature into related styles and architectures, and to succinctly illuminate the relations between them, whatever forms those may end up taking.

Modeling the structural properties of software architectural styles has generally been associated with the component/connector abstractions, and has utilized architectural description languages (ADLs) [51, 54, 55, 56] for formal description. We believe that traditional component/connector abstractions associated with style mod-

eling do not provide sufficient abstractions for network architects to work with ². Therefore, we simply borrow the notion of “architectural style” without constraining ourselves to the component, connector, port, and role abstractions. Additionally, we choose to use the Alloy modeling language [134] rather than ADLs based on Alloy’s simplicity, its expressive power and ability to describe structural and behavioral aspects of an architectural style, and its ability to model desired specification properties that fit our needs (invariants, inheritance, and composition). Despite Alloy’s scalability concerns, we have found it useful to formally describe network architectures/styles because of the presumably small scope of abstractions involved in describing network architectural styles.

5.2.2 Alloy

Architectural design revolves around exploring the right abstractions, which are simple ideas expressed in some primitive form. Designing those abstractions requires a formal specification language that is intuitive, expressive, and at the same time avoids the intricacies of coding. Alloy [134] is one such language that we use to write our formalization of the FARA style [30] (to be detailed shortly). Alloy is a declarative language based on relations and first-order predicate logic. A brief overview of Alloy’s logic, language, and analysis follows. A complete reference is located elsewhere [134].

The Logic - At the core of Alloy is a relational logic that combines relational algebra with first-order predicate logic. Structures are composed of *atoms* and *relations*. Atoms represent typed, immutable structures that are uninterpreted and can be related through relations. A relation is a set of tuples each being an atom and can have arbitrary arity. Relations are combined with *operators* to form *expressions*.

²The component/connector abstractions might be sufficient when modeling communication systems, as may be deduced from the axiomatic model in [24].

Set operators	Relational operators	Logical operators
+ for union	\rightarrow for product	! for negation
− for difference	. for join	&& for conjunction
& for intersection	\sim for transpose	for disjunction
<i>in</i> for subset	\wedge for transitive closure	\Rightarrow for implication
= for equality	* for reflexive-transitive closure	, for alternative
		\Leftrightarrow for bi-implication

Table 5.1: Operators in Alloy.

Some of the most common operators in Alloy are tabulated in Table 5.1.

Constraints are formed of expressions and logical operators. Quantified constraints take the form $Q\ x : e|F$, where F is a constraint over x , e is an expression bounding x , and Q is a quantifier that can take values **all** (universal), **some** (existential), **no** (no values), and **lone** (at most one value). For example, **no** $x : e|F$ is true when no x in e satisfies F . When *Let* is used as in $Let\ a = b|F$, every occurrence of a in F is replaced by b .

Declarations in Alloy take the form *relation-name* : *expression*, where expression is the bounding expression for the declared relation (as if : is replaced by **in**). Bounding expressions can specify multiplicity markings, which can take values **one** (exactly one), **some** (one or more), **set** (zero or more), or **lone** (at most one). For example, $r : Am \rightarrow nB$, where m and n are multiplicities, is a declaration saying that relation r is constrained to map each element of set A to n elements of set B , and each element of set B to m elements of set A .

The Language - In addition to the logic, Alloy provides some language constructs to help organize a *model*. A model in Alloy may consist of signatures (**sig**), facts (**fact**), functions (**fun**), predicates (**pred**), and assertions (**assert**).

Signature: A signature, declared with **sig**, introduces a basic type along with a collection of fields, their types and restrictions over their values. A signature can **extend** another signature inheriting its fields and constraints. An **abstract** signature has no elements except those belonging to its extensions. For example, if

Chapter 5. Towards Formalizing Network Architectural Descriptions

we write:

```
abstract sig A {                                abstract sig B {}
  f: set B                                       sig A1 extends A {}
}{--constraints go here}                      sig A2 extends A {}
```

one sig C{} --‘one’ means sig constrained to one element

we have declared three elements A , $A1$, and $A2$. Since $A1$ and $A2$ extend A , it follows that A in $A1 + A2$. Additionally, because A is abstract, it follows that $A = A1 + A2$ and $A1$ and $A2$ are disjoint sets that partition A . A declares a field f of type B . This is saying that for each element A , $A.f$ is a set of type B , i.e., the relation f is mapping from elements in A to elements in B .

Facts, Predicates, Functions, and Assertions: A **fact** is simply a constraint that is assumed always to hold, and hence needs not be explicitly invoked. Facts usually describe global model constraints. The facts and the signature constraints thus constitute a complete set of structural constraints over the model.

A function, declared with **fun**, is a named reusable expression that can be invoked within the model. A function takes zero or more arguments and returns either a true/false or a relational value.

A predicate, declared with **pred**, is a named reusable constraint that can be invoked. A predicate takes zero or more arguments.

An assertion, declared with **assert**, is a named constraint that is intended to follow from the model’s facts. Assertions take no arguments and are usually checked by the Alloy Analyzer as discussed next.

The Analysis - The Alloy Analyzer (AA) [135] is an automated tool for analyzing models written in Alloy. Two kinds of analysis are enabled by AA, based on *commands*. The first is *simulation* (using **run** command) whereby the validity of a predicate or function is verified by showing a snapshot of the system for which

the predicate is valid. The second analysis technique is *checking* (using **check** command), whereby an assertion is tested and AA tries to find a counterexample. This requires a finite *scope*, bounding the number of atom instances within the universe, within which AA looks for solutions. Given the undecidability of predicate logic, a finite scope is necessary to bound the space within which AA searches. Finding an instance to a predicate or a counterexample to an assertion guarantees the consistency of the constraint. However, failure to find such instance simply makes it inconsistent *within the scope*. The intuition is that subtle design bugs are likely to be detected even in small scopes.

5.3 Case Study

To motivate the usefulness of formal architectural modeling, and the expressiveness of the Alloy language, we represent the FARA[30] family of network architectures (or the FARA architectural style) using a formal model. Briefly, FARA [30] is an abstract network model in which the current Internet architecture is generalized and remodeled to enable clean separation of endpoint names from network addresses. Modeling FARA is an illustrative exercise in *architectural abstraction*, whereby a basis set of structural and behavioral components, assumptions, and constraints (invariants) that pertain to a desired class of architectures are extracted at the first stage of design to describe the general architectural model. Instantiations of the general model may then specialize it, obeying the general design assumptions and invariants. The authors of FARA had to implement a prototype of a FARA instantiation, M-FARA [30], in order to validate FARA’s usefulness, and self-consistency. One of the goals of this section is to show how a formal model can be expressive and efficient in validating architectural design decisions, hoping to replace “validation through implementation” by “validation through formal modeling”. Aside from pro-

viding a conceptual framework for reasoning about a class of architectures, a formal model of an architectural style (such as FARA) transcends into a formal framework over which essential architectural design decisions can be modeled and verified.

5.3.1 FARA Overview

The sought goals of FARA include cleanly decoupling the application identity from network address, avoiding a new global namespace, and providing security with a range of assurance levels. An overview of the basic components, assumptions, and functionality of the FARA style follows. More details are provided within the formal model in the next section. The FARA abstraction recognizes communication among pairs of *entities* via logical links referred to as *associations* on top of a *communication substrate*. An entity in FARA is the end-point of communication and smallest unit that can be mobile, such as a process, a thread, or a cluster of devices. An association is a logical communication link between a pair of entities representing persistent communication state. Entities maintain local association state and may have multiple concurrent associations. A packet belongs to one association and carries an association ID (AID) that enables the receiving entity to correctly demultiplex the packet to its association. As to the communication substrate, it represents underlying infrastructure that is able to deliver packets on behalf of associations. Addressing, routing, forwarding are mechanisms employed by the substrate and are left unspecified by FARA. However, FARA assumes connectionless point-to-point communication between entities. An entity supplies the substrate with a packet and header that contains a destination Forwarding Directive (FD). The latter contains enough information that the substrate can use to deliver the packet all the way to the destination entity that contains the association. The clear separation between entities and associations on one hand, and the communication substrate on the other hand is visualized in FARA by a “red line” horizontally separating the two

Listing 1

```
abstract sig AID{}

abstract sig Entity{
  associations: Entity->Time,
  state: associations->one AID,
}{
  no (this & associations.univ)
  all t:Time, aid:AID |
    lone (state.aid).t
  #state = #associations
}
abstract sig RString {}
```

Listing 2

```
abstract sig FD{}
abstract sig Packet{
  dstFD: FD,
  replyFD: FD
}
abstract sig DPacket extends Packet{
  srcAID: AID,
  dstAID: AID
}
abstract sig SPacket extends Packet{
  ri: RString
}
```

(entities/associations above the line).

5.3.2 FARA model

We hereby lay out a formal description of FARA’s basic structural and behavioral components (static and dynamic properties) along with the constraints attached to the components and to the overall architectural style. The description accounts for dynamic behavior by explicitly including logical time steps to model evolution over time. Note however that analyzing the static properties of the architecture, simply requires dealing with a snapshot of the system at some timestep t , i.e., constraining the analysis scope of the *Time* signature to 1 instance.

Structural aspects

A formal definition of the *entity* and the *association* is given in Listing 1. An **Entity** is an abstract element that can have multiple concurrent **associations**. An association is a relation between two entities over time. Each entity maintains local immutable **state** per association, the association ID (**AID**). A particular association

has exactly one AID, and AIDs are reusable over time. Several constraints are attached to the entity definition: the first constraint eliminates associations that connect an entity to itself for simplicity. The second constraint is one of FARA's key assumptions, and it states that no two associations of an entity can have the same AID at any given time. The third structural consistency constraint forces each association to have state. An entity does not define a universal name since FARA does not require a global namespace. Our approach to modeling an association as part of the entity's signature versus modeling it as a separate semantic element renders the dynamic constraints simpler and clearer.

Listing 2 defines the Forwarding Directive (FD) and the packet abstractions. The **FD** encapsulates enough topological information to allow the substrate to deliver a packet to its intended destination. A generic packet, **Packet**, says nothing about the identity of the entities, and must indicate a destination forwarding directive (**dstFD**) that will be used by the communication substrate (to be defined shortly) to deliver the packet to a destination entity. A packet might also include a reply FD (**replyFD**) which the destination entity utilizes on the reverse path. FARA distinguishes between a packet that belongs to an association, a **DPacket**, and a setup packet, **SPacket**, that bootstraps an association. DPacket must specify the association state at both ends of an association, **srcAID** and **dstAID**, allowing the destination entity to correctly demultiplex the packet to its association. SPacket includes a rendezvous information string, **ri** of type **RIStrng**, and does not include association state since the association is being bootstrapped.

Listing 3 defines the communication substrate component, **CommSubstrate**, representing a single global medium (the underlying operating systems and network) that is able to deliver packets on behalf of associations. The substrate assumes a basic connectionless delivery, **delivery**, without making any assumptions about the delivery function itself. A particular FARA instance, as we shall see later, will

provide the respective addressing, routing and forwarding mechanisms required for successful packet delivery. Supplied with an FD, the substrate delivers a packet all the way to its destination entity. The point-to-point assumption in FARA is modeled as part of the CommSubstrate constraints specifying that an FD can lead to a single entity at any time. So far, the model defines entities and associations independently of the mechanisms employed by the substrate for packet delivery. This acknowledges FARA’s “red line” logical separation, whereby entities and associations operate above the line while the communication substrate operates below the line. Additionally, as a key assumption of FARA, no global address space is defined, with the intent of supporting a multitude of forwarding mechanisms.

Global style constraints, or simply invariants, are specified in Listing 4. The first consistency invariant constrains the association to be symmetric. Hence, entity A has an association with entity B if and only if the latter has an association with entity A. The second constraint eliminates dangling association states.

Having formally described the style, we may now proceed to validate some of its properties, specified as predicates and checked through the AA. For example, to check whether an entity might have overlapping state for distinct associations at some time, we define and run the predicate in Listing 5. AA does not find any instance of overlapping state within the simulated scope (7 Entity, Packet, FD, etc.; 15 AID; and 20 Time instances). This guarantees the correctness of the above claim only within the specified finite scope, and not in general. However, if inconsistent

Listing 3

```
abstract one sig CommSubstrate{
  delivery: FD-> Entity -> Time
}{
  all t:Time | delivery.t in
    FD -> one Entity
}
```

Listing 4

```
fact Invariants{
  all t:Time | associations.t
    = ~(associations.t)
    Time.(Entity.(Entity.state))
    = AID
}
```


Listing 5

<pre> pred showOverlapState { all t:Time some disj e1,e2,e3:Entity let w12=getAssociation[e1,e2,t], w13=getAssociation[e1,e3,t] e1.w12=e1.w13 and some w12 } run showOverlapState for 7 but 15 AID, 20 Time </pre>	<pre> --Returns the entity AIDs on both --sides of the association fun getAssociation [fst,snd:Entity,t:Time]:Entity->AID { fst -> t.(snd.(fst.state)) + snd -> t.(fst.(snd.state)) } </pre>
--	---

models can indeed be found, it is likely to find those within the specified scope.

Functional aspects

This section shows how functional aspects are formally specified at a high level of abstraction, leaving the details for architectural instances to specify.

The first function specified in FARA deals with the creation of associations. To model the system's dynamic behavior as a response to establishing and tearing down associations, we use Alloy traces to capture state transitions over time. Initially, at time t_0 , there are no associations. As presented in Listing 6, we consider two events that may change the system's state, the establishment or the tearing down of an association. The time instants **t1** and **t2** describe the state of the system before and after an operation is performed, respectively.

Given the possible state transitions of the system, we can form those into an execution trace by modeling the latter as a fact (Listing 7). Assertions may then be checked against the trace. An invalid assertion will demonstrate a trace showing how the assertion was violated. The Alloy analyzer may be used to show some execution trace of the system. For example, running the **showSomeState** assertion using AA, we obtain a counterexample showing a sample trace which, when projected over time, clearly demonstrates the state change resulting from creating or tearing down

Listing 6

```

pred init[t:Time]{
  no associations.t
}
pred establishAssociation
[t1,t2:Time, fst,snd: Entity]{
  --Preconditions
  ---association does not exist
  let aset = {fst->snd+snd->fst}
    | no (aset & associations.t1)
  --Postconditions
  --no association change
  let aset = {fst->snd+snd->fst} |
  {
    noAssociationStateChange[t1,t2]
    associations.t2 =
      associations.t1 + aset
  }
}

pred teardownAssociation
[t1,t2:Time, fst,snd: Entity]{
  --association exists
  let aset={fst->snd+snd->fst}|
    some (aset & associations.t1)
  --remove it
  let aset={fst->snd+snd->fst}|
    associations.t2 =
      associations.t1 - aset
}
--associations @t1 valid @t2
pred noAssociationStateChange
[t1,t2: Time] {
  all e1,e2:Entity |
    getAssociation[e1,e2,t1]
      in getAssociation[e1,e2,t2]
}

```

associations.

M-FARA: an Instantiation

M-FARA [30] is an instantiation of FARA that specifies its own addressing, forwarding, and FD management mechanisms. M-FARA is not a complete architecture, but it is specific enough to explore two points in the FARA design space: 1) location/i-

Listing 7

```

fact Traces {
  init [T0/first[]]
  all t:Time-T0/last[] |
    let t' = T0/next[t] |
      some disj e1,e2:Entity|
        establishAssociation[t,t',e1,e2]
        or teardownAssociation[t,t',e1,e2]
}

assert showSomeState{
  no e:Entity |
    #e.associations >=1
}

check showSomeState for 4
but 7 AID, 7 Time,
0 RString, 0 Packet

```

identity separation, and 2) mobility. This section models M-FARA, particularly its addressing and forwarding mechanisms, using Alloy to demonstrate style specialization.

First, a new module for M-FARA is created importing the FARA module just defined. Several new addressing and topological abstractions are introduced by the M-FARA module, as shown in Listing 8. M-FARA assumes multiple addressing realms, **Domains**, each having a **space** of unique addresses. A **subFD** represents a set of addresses that determine a local path within a domain. A domain has a static address space, **space**, and a dynamic forwarding mechanism, **forwarding**. The latter delivers a packet that is destined to some subFD to the entity that is bound to the respective subFD. Moreover, the topology assumed in M-FARA consists of

Listing 8

<pre> sig subFD{} abstract sig Domain { space: set subFD, forwarding: space->Entity->Time }{ --point2point forwarding all t:Time forwarding.t in subFD -> lone Entity } --*No global address space*-- one sig MF_CommSubstrate extends CommSubstrate{ domains: set Domain, } </pre>	<pre> one sig Core extends Domain {} sig PrivDomain extends Domain{ upspace: some subFD, downspace: set subFD }{ upspace in space downspace in space no (upspace & downspace) -- up forwarding is implicit no ((forwarding.Time).Entity) & upspace } </pre>
--	--

a two-level domain hierarchy with a single distinguished central “**Core**” domain to which the private domains, **PrivDomains**, connect (Listing 8). The extended communication substrate, **MF_CommSubstrate**, may thus be viewed as the set of all domains including the core. Part of a private domain’s space, **upspace**, is used to reach the “core” domain. Similarly, part of the “core” domain’s space, **downspace**, is used by the core to reach the private domains. In this model, it is implicitly assumed

Listing 9

<pre>sig MF_FD extends FD { up: lone subFD, down: one subFD }</pre>	<pre>sig MF_Entity extends Entity{ --canonical route fddown: subFD -> Time, }</pre>
---	--

that the forwarding function of every domain delivers subFDs belonging to upspace to the core. On the other hand, forwarding from the core down to the domain is explicitly specified in the domain’s forwarding function (hence subFDs belonging to downspace originate at the “core”).

Listing 9 defines the complete end-to-end FD in M-FARA, **MF_FD**. It consists of a tuple $(FDup, FDdown)$ which the substrate can use to forward a packet from the source up to the “core” (**up**), and then from the “core” down to the destination entity (**down**). Regarding the entity abstraction, **MF_Entity**, M-FARA extends the entity definition with the local subFD to which the entity is bound, **fddown** and on which it is reachable. M-FARA does not specify whether an entity may be multi-homed (simultaneously bound to multiple domains) or not and our model does not restrict that either.

Some general structural constraints apply to the model and are expressed in Listing 10. No dangling subFDs or domains are allowed. Additionally, a subFD can belong to a single domain’s address space. Finally, the forwarding operation is local to the domain, i.e., an entry in the domain’s forwarding table means that the entity

Listing 10

<pre>fact Invariants{ --no dangling subFDs Domain.space = subFD --no Dangling Domains MF_CommSubstrate.domains = Domain --space is private all sf: subFD lone space.sf</pre>	<pre>--Forwarding is local to a domain all t:Time, d:Domain let fwd = d.forwarding.t all sfd:subFD, e:MF_Entity {sfd ->e in fwd => sfd->t in e.fddown} }</pre>
--	---

is bound to the domain.

Modeling mobility in M-FARA is another interesting exercise, which we do not address in this chapter. This task requires extending the FARA dynamical behavior, which so far includes establishing and tearing down associations, with a new mobility operation.

Abstract style properties

We have so far modeled an architectural style, FARA, and a particular instantiation of the style, M-FARA. The FARA style advertises a global theme of separating the entity from the communication substrate, and a set of style goals and properties. Despite the fact that the style leaves much of the functional details unspecified (such as addressing and forwarding mechanisms in our example), it is still essential for the style architect to model *super-properties*. *A super-property is a property of the style that is expressed in terms of abstract unspecified functionality.* In other words, the architect needs to confirm that any instantiation of the style that specifies the missing functionality will do that in such a way that the super-properties are respected. In object-oriented programming, such design methodology is known as polymorphism. This section demonstrates a process for modeling style super-properties and checking those against the instantiation, by referring back to the FARA style and the M-FARA instantiation models.

As a first step, the style model includes the super-properties as facts, predicates, or assertions expressed in terms of unspecified functionality. The snippet in Listing 11 augments the previous FARA model with two new invariants (super-properties), expressed in Alloy as facts. The first fact is a “below the line” property. It states that delivery, which we have previously defined as part of the **CommSubstrate** in FARA, must be supported by the substrate’s addressing and forwarding mechanisms.

Listing 11

```
--Step1: super-property 1
fact {
  all t:Time |
    let delv=CommSubstrate.delivery.t
    | all fd:FD, e:Entity
    | {fd->e in delv =>
        this/isDeliverable[fd,e,t]}
}
--super-property 2
fact {
  all t:Time, e:Entity |
    let ea = e.associations.t
    |some fd:FD |
        this/ise2eDeliverable[e,ea,fd,t]
}
--*To be specified by Instance
pred isDeliverable
[fd:FD, e:Entity, t:Time]{}

pred ise2eDeliverable
[src,dst:Entity,dstfd:FD,t:Time] {}
```

Listing 12

```
--Step 2
--Replicate facts from FARA
...
--*overridden function
pred isDeliverable
[dst:FD,e:Entity,t:Time]{
  let d_sfd=dst.down,
  d_dom = (getDomain[d_sfd])
  |d_sfd in d_dom.downspace and
  (d_sfd->e in
    d_dom.forwarding.t) }

--*overridden function
pred ise2eDeliverable
[src,dst:Entity,dstfd:FD,t:Time]{
  some dstfd.up and
  dstfd.up in
  (getEntityAttachments[src,t].univ)
  .upspace
  this/isDeliverable [dstfd,dst,t]
}
```

In other words, if the substrate is able to deliver a message to an entity based on some destination FD, then the substrate’s forwarding mechanism must be able to deliver to that entity, hence satisfying **isDeliverable**. Again, note that **isDeliverable** is left unspecified by the style (in step 1), and is to be implemented by an instantiating architecture based on the forwarding mechanisms employed. The second fact is an end-to-end property (“above the line”) stating that an association exists and is valid only if packets are able to flow over the association from source to destination. In other words, there must exist some FD that satisfies **ise2eDeliverable**.

As a second step, the style instantiation extends the style model implementing the unspecified functionality. Super-properties are then enforced and checked against the instantiation to verify that the desired style goals are satisfied by all instantiations. To illustrate this step, the M-FARA model is augmented with the

Alloy snippet in Listing 12, overriding the abstract functionality, `isDeliverable` and `ise2eDeliverable`³. In M-FARA, `isDeliverable` or deliverability implies that: 1) some packet may be forwarded from the “core” down to destination’s domain, i.e., the *FDdown* part of the destination FD should belong to the **downspace** of the entity’s current domain, and 2) the domain’s forwarding function delivers to the entity given *FDdown*. End-to-end deliverability, in turn, requires two valid paths: one from the source entity’s domain up to the core, and another from the “core” down to the destination entity.

In the same fashion that facts about the style were replicated in the instantiation above, assertions and predicates may also be replicated. It is straightforward to add assertions that verify the facts introduced above. For example, assertions dealing with mobility may easily be implemented.

Composition

Having already demonstrated inheritance and polymorphism in style modeling, we proceed to define and briefly overview *composition* as a means for composing separately defined modules or styles and checking for their compatibility. Let $S_i|_{i=1}^n$, $n > 1$ be two or more styles, and let P_i , $i = 1..n$, be the global consistency constraints defined by S_i . The new composed style is denoted by $S = C(S_1, \dots, S_n)$ and contains the merged constraint set $\bigcup_{i=1}^n P_i$. S_i s are compatible styles *iff* the new consistency constraint $P = \&\&_{i=1}^n P_i$ is satisfied by S .

As an example of composition, assume that a global-hierarchical addressing style, GHAR, is defined in which address spaces or domains are composed hierarchically (for example through customer-provider or peering relationships) with a distinguished core. The FARA style may then be composed with GHAR into a new style, say

³In Alloy, the super-properties have to be replicated to the M-FARA model since Alloy does not directly support inheritance of a style or “module”.

FARA-GH. An entity in FARA-GH extends the FARA entity and defines a global address field that is inherently hierarchical. Interestingly, the new FARA-GH architecture resembles the NIRA [99] routing architecture with the added conceptual clarity and design space partitioning.

5.4 Related Work

There are two broad areas of related work. The first is concerned with network architecture and communication system modeling, while the second deals with software system modeling.

Regarding network architecture modeling, the Internet architecture has been thoroughly studied over the past decade. The design principles of the DARPA Internet are clearly outlined in Clark’s seminal paper [2]. The paper highlights the connection between the intended goals of the DARPA Internet and design decisions that govern its current operation. The paper was intended to illuminate the Internet’s design principles rather than to formally model the Internet architecture. The same applies for other architectural design papers [47, 50].

A methodology for designing and assessing evolvable network architectures based on *invariants* (or *fixed points*) is proposed in [48]. The authors highlight a useful point which calls for considering invariants at an early design phase. However, they do not provide a complete design methodology or formal framework for reasoning about network architectures. Our formalization model inherently accounts for invariants as a part of the complete architectural description, and hence provides the architect with a clearer formal framework to work with invariants.

As to communication system modeling, we identify several relevant proposals that we believe are complementary to our work. However, our work is again concerned with modeling general architectural descriptions rather than switching properties of

networks. Karsten et al. [24] have proposed a general axiomatic basis to consistently model communication primitives such as forwarding, naming, and addressing for better expressing architectural invariants and formally proving properties about node reachability within any communication system.

Another relevant work is that proposed by Zave [21, 22]. In [22], the author utilizes the Alloy modeling language to formally model identifier binding schemes which enables informed architectural design decisions for better supporting networking services. A less general abstraction of the domain and the requirements on binding composition to satisfy inter-operation was modeled by in [21].

The proposal MDCM [25] attempts to describe a wide variety of multi-domain, multi-layer communication systems through a unified model.

Regarding the modeling of software architectures, a lot of work has focused on formally describing those using Architecture Description Languages (ADL) [51, 54, 55, 56]. Some of the common ADLs are the Acme ADL with the underlying first-order logic [55], extended WRIGHT [54], process ADL with the underlying process algebra [56], and π -ADL with the underlying π -calculus [57]. The Acme model in [55] utilizes Alloy and is a very relevant work to ours. Style inheritance and composition as well as verification of structural properties and compatibility checking are concepts demonstrated by the authors; however, their current model falls short of capturing the behavioral aspects of the architectural style. Alternatively, the model in [56] explicitly involves topology specification (i.e. component/connector instances and their interconnections) as part of the architectural style description, which we believe is not an efficient approach considering the level of abstraction at hand.

Finally, Alloy has been utilized within several modeling case studies that as described on the Alloy website [135]. We mention here some of those that pertain to networking and that were useful for this work. Khurshid [136] has used Alloy for modeling and correcting the architecture of the Intentional Naming System (INS). Jackson [135] has used it to model the Chord peer-to-peer lookup protocol. Some

recent work by Narain [137] utilizes Alloy’s model finding techniques to find network configurations that satisfy a set of input requirements expressed with predicate logic.

5.5 Discussion and Future Work

As previously stated, we have refrained from using the component/connector/interface abstractions for modeling network architectural styles. By surveying the network architecture literature, we have noticed that architects have different approaches to modeling abstractions. It is our belief that constraining them to component/connector/interface abstractions limits the expressiveness of the model and hence the innovation. It is additionally hard to anticipate whether and what modeling abstractions for networks will emerge in the future. The language we have utilized, Alloy, is generic and flexible enough to allow the architect to represent whatever abstractions she finds suitable. Despite the scalability concerns associated with constrained instances in Alloy, which does not represent a major limitation to us considering the high level of abstraction being modeled (and hence the presumable small instance sets required), the problem is currently being addressed in the literature (such as in [138]).

While this work has presented a first step towards formalizing network architectures and architectural styles, several research challenges remain to be solved and we address those as part of our current and future research. First, there needs to be a consensus regarding the most imminent styles that span the network architecture design space. Modern and future network architectures, as has been recently acknowledged [3], are being equipped with more intelligence, generally introducing information and computation structures that are manifested through increased in-network processing and storage. Extracting a complete, and disjoint set of network architectural styles may potentially frame the architectural problem and provide a

formal framework for classifying, relating, and reasoning about architectures. Towards this end, we believe that the architecture taxonomy presented in chapter 4 is a timely and essential contribution.

5.6 Conclusion

To conclude, this chapter has presented a methodology towards formally describing and modeling network architectures and architectural styles. Style inheritance, polymorphism, and composition were demonstrated on the FARA class of network architectures using the Alloy modeling language. Our work helps network architects and researchers, whereby architects are able to formally represent and group various architectural patterns into styles, while researchers are provided with a means to better understand, analyze, and reason about network architectures. We would like to note that an initial version of this chapter appeared in [139]. This chapter concludes part I of the dissertation which has focused on general architectural design, classification, and modeling. Building on part I and specifically on our experiences with the TNA architecture, part II proceeds to introduce naming and discovery in networks, and to present two contributions in that vein.

Part II

Naming and Discovery

Chapter 6

Background on Naming and Discovery

6.1 Introduction

Almost every networking application relies on discovery and naming/identification services. As we have seen in section 2.2, naming and discovery is an integral part of a network architecture. This chapter defines and elaborates on the terms *identifier* and *locator* and defines *identifier-based discovery* to set the stage for further investigation of naming and discovery problems in later chapters. Confusion about naming and addressing in communication networks (with the terms name, address, route, identifier etc.) dates almost three decades back. Shoch [140] constructed the general definitions of the terms name, address, and route. Building on that, Saltzer in his RFC [141] explains the confusion by shedding a different and very helpful perspective on the constructs of naming and addressing in data communication networks. He argues that in order to distinguish names and addresses, it is helpful to distinguish four different kinds of objects: a service, a node, an attachment point, and a path.

Chapter 6. Background on Naming and Discovery

Any of the four kinds of objects may have a *name* where a name identifies “what you want”. On the other hand, the *address* of an object is merely a name of the object it is bound to, hence the importance of the *binding* concept. For example, the address of a node is the name of an attachment point to which the node is bound, and the address of the attachment point is the name of a path to which the attachment point is bound. The address then identifies “where the object is”. The route identifies “how to get there”. Saltzer presents a clarifying example which we think is instructive to review here: when trying to understand whether the IP address is a name of the attachment point or a name of the node, confusion may arise. Consider a node x having IP address $x.y$. The node changes its attachment point and keeps the address $x.y$ unchanged in the new attachment point by changing the forwarding tables within the network. One may be tempted to think that the IP address is then a name of the node since it remained unchanged across changes of the attachment point. Notice however that by changing the routing tables within the network, what has really happened is that the permanent name of the new attachment point has changed. Hence the IP address is the name of the attachment point instead (which in this case is the name of the node as well). There is a subtle difference when it comes to changing the name versus changing the binding which generally confuses the discussion. One needs to distinguish two types of bindings: the binding between a name and the named object which is generally a long-term binding; and the binding of an object to another object to which it is bound. Saltzer leaves room for interpretation when he seems to explain a long-term binding as one requiring change in “more than one table”:

... The association of the name with the service is quite permanent, and because of that permanence is not usually expressed in a single, easily changed table.

... Changing tables superficially appears to be what rebinding is all about,

the need to change more than one table is the tip-off that something deeper is going on.

Chiappa [11] utilizes Saltzer's model to elaborate on the problems that arise from coupling the attachment point name and the node name in the Internet TCP/IP implementation (the IP address provides both names). Chiappa deviates from the generic definition of "address" (as the name of some object to which something is bound) to link it to a specific object which is the attachment point:

..... the exact definition of "address", at least in an internetwork with routers, ought to be ... "the name of a network connection entity to which the system of routers will deliver a packet".

Chiappa introduces the "endpoint" object which is the endpoint of communication in the end-to-end TCP/IP architecture. He differentiates it from a node or host object by recognizing that a mobile process on a host is a different object than the host.

Building on Saltzer's definitions, we revisit the generic concepts of name and address in information networks and we attempt at redefining those by fixing a boundary between different types of objects. Our goal is to further clarify the concepts of naming and addressing in light of recent work on the topic, and to set up a framework for thinking about naming and discovery that will help us investigate the design space, and will set the stage for later chapters.

We use the graph abstraction to model a network where a node is an abstract construct. We assume the existence of a logical plane that divides each node ¹ (and hence the whole network) into two spaces: an upper space in which entities (*e.g.*, endpoints [11]) reside and a lower space in which locations reside (check [140, 11, 30]

¹When we refer to a node, we are referring to the an abstract node in the graph abstraction. This is not to be confused with Saltzer's node object [141] which is analogous to the host. The distinction between the two should be clear from the context.

for similar models). Only after fixing the boundary, we allow ourselves to talk of names (identifiers) and addresses (locators) which are again names of different objects at different layers. We present definitions for the terms identifiers and locators that conform to Saltzer's RFC [141]. However, our definitions are less restrictive in the sense that we are solely interested in highlighting the boundary relationship between the location and the entity abstractions or objects. We use the term entity as a generic term to refer to endpoints in the upper space (subsumes node and service objects in [141], processes, etc.) and location to refer to anything in the lower space (subsumes attachment point and path objects in [141]). Multiple objects and levels of naming could exist in the upper space such as intentional names to service names, and service names to node names. We only model the entity abstraction as we are mainly interested in examining that binding between the entity and the underlying location.

The rest of the chapter is organized as follows: section 6.2 redefines the terms identifier and locator and elaborates on the confusion that arises between them. Section 6.3 defines discovery and relates it to routing. Finally, we illustrate the concepts in section 6.4 by examining two special architectural designs: the TCP/IP Internet [1], and compact routing design [44] before concluding in section 6.5.

6.2 Definitions

We now define a simple and generic model for addressing in information networks whose currency is *locations*, and *entities*. All the definitions presented hereafter assume the existence of some undirected graph $G = (V, E)$. Topology in this context refers to the graph topology. Denote the neighbor set of node v by $N_v = \{u \in V : (v, u) \in E\}$.

Locations Location is a relative concept. We denote the *location* of a node v relative to node u by l_v^u and we keep it abstract. We assume that every node v knows about the location of its neighbors, and hence $l_u^v = (v, u) \in E, \forall u \in N_v$. The location of node v , l_v , is then defined to be $l_v = \bigcup_{u \in V} l_v^u$ which is the node's relative location to the rest of the network.

Locators and Locator Space We then define a *locator* of node v kept by node u , denoted by $loc_u[v]$, as a name of the location object or its address “how to get to it”². Some examples of $loc_u[v]$ include a route $u \rightsquigarrow v$ in G (*e.g.*, source routing IETF RFC 4728), or the distance $d(u, v)$ and a direction in some metric space (*e.g.*, greedy routing [142]), or a set of coordinates in a cartesian metric space (assuming the graph is embedded in the metric space), or an IP address (as in IP addressing and BGP [42]). The multiple paths to some location l_v potentially implies multiple locators, and the relative concept potentially allows different locators to be used by different groups of nodes (this concept shall become clearer when we discuss compact routing architectures in section 6.4.2). We say that a locator is *global* when it is the same for all $u \in V$ and we refer to it as simply $loc[v]$. Again the same location might have multiple global locators. The set of all locators is referred to as the *locator space* and denoted by \mathcal{L} , i.e., $loc_u[v] \in \mathcal{L}, \forall u, v \in V$.

Forwarding Functions and State Since the ultimate goal of locating something on an information network is to access it, locators and forwarding work in tandem to provide location access. The forwarding function provides the means to get to the location using a locator. The forwarding function is local to a node. It is defined as a set-valued function $f_u : \mathcal{L} \rightarrow V$ which maps a locator to a set of neighbor nodes $u \in N_v$. The function generally utilizes forwarding information

²We do not distinguish between the name of a location object and the name of the path object (how to get to the location) and we use locator to refer to either name. The reason is that below the imaginary plane, we are only interested in how to get to the location object.

(hard state) that is kept locally at the node - the bindings between location name and path(s). When the locator is the name of a location that does not carry information about how to get to the location, then the binding between the location and the path(s) to the location is in the forwarding state. There is generally a tradeoff between the path information that the locator carries (how to get to the location) and the amount of network forwarding state; the more information in the locator, the less forwarding state is required and vice versa. For example when the locator is a source route then no forwarding state is needed.

Entities and Default Entities : In contrast to locations, entities reside above the imaginary plane. Each node $v \in V$ can host a set of entities $O_v \subseteq \mathcal{O}$ where \mathcal{O} is the set of all entities. Among the set of entities hosted on a node v , let o_v , be the node's default entity $\forall v \in V$ where a default entity is intentionally defined to be attached to a node. The entity subsumes Saltzer's node and service objects [141] and is intended to be more generic than the endpoint [11] in the sense that it is not restricted to end-to-end architectures or TCP/IP. The default entity is identical to Saltzer's node object except that here we treat Saltzer's node and service objects the same way - entities that reside above the imaginary plane. The best way to think of entities is "things" that attach to locations. Examples of entities include endpoints of communication (such as services or processes), users, or even higher level abstractions. There are instances in which a node object and location object are confused. For example, in the Internet a node is an object that is distinct from the location and that may be named separately. In compact routing on the other hand (as discussed in section 6.4.2), the notion of a node in the graph abstraction might encapsulate both an entity and a location. The default object is intended to alleviate the confusion in the following sense: a location object by definition can not move while a default entity can move. When the entity and the location

have the same name, then one may think of this name as simultaneously naming the location and the default object bound to the location. By definition, when we say that the default object moved and maintained the same name across the location change, then the new location has been assigned a new name. In a Distributed Hash Table (DHT) when the node and the data files hosted on it take names from the same namespace(*e.g.*, [143, 100], one may think of the files as entities and the node as a default entity.

Identifiers An *identifier* is a name of some entity $o \in \mathcal{O}$ and is denoted by $ID[o]$.

The identifier exists in some arbitrary identifier space \mathcal{I} . Note that one may refer to $ID[o_v]$ as the node identifier or the default entity identifier to mean the same thing. In this sense, the set of node identifiers is defined as $\hat{I} = \{ID[o_v] : \forall v \in V\}$, satisfying $\hat{I} \subseteq \mathcal{I}$. Multiple identifier spaces may exist. Depending on the design, multiple entities may have the same identifier (as in the case of entity replicas in CDNs) or an entity may have multiple identifiers.

Before concluding this section, it is instructive to revisit some of the causes of confusion in terminology. First, notice that the definition of a locator does not require that it “follows topology” (Rekhter’s law ³). As we just mentioned a locator does not need to carry any topological information. For example, one may identify locations with flat names and in the extreme case maintain $O(n)$ state per node about the locator space (as in Provider-Independent IP addressing). In this case, the flat name is still a locator. The binding between the name of the location and location itself is maintained in the forwarding state. Translating the locator to an address is performed by the forwarding function. If some location with a flat name is to relocate and keep its name, then it must update the forwarding state in the whole network. In this case, the name of the new location has changed versus the binding

³For scalability, Yakov Rekhter (one of the authors of BGP4 [42]) states that “Addressing can follow topology or topology can follow addressing. Choose one..” [144].

between the name and the location. This concept is important as it eliminates a lot of the confusion in terminology. If a location changes then by definition it is a new location. Keeping the name after the change means that the new location has been renamed. On the other hand, if an entity moves to a new location, the entity is still the same object conceptually. It is this distinction that gives the locator its definition (as the name of a location) and distinguishes it from the identifier. Finally, one may not directly assume that if the forwarding system is aware of the name then the name automatically becomes a locator (check [143] for a system in which the forwarding system is aware of entity names as well). According to our definition, the forwarding function is tied to a context: the context of location objects. Whenever the forwarding function is aware of a name then the name becomes a locator **in that context**. The name could be an identifier in a different context: the context of entities. To eliminate the confusion, it helps to think of contexts in which objects exist before making a distinction. On the TCP/IP internet, the layering design cleanly and explicitly separates the contexts where each layer is a different context that recognizes a different type/set of objects. In the compact routing design (as we shall see later in section 6.4), there is no explicit separation of contexts of objects.

6.3 What is Identifier-based Discovery?

To recap, it should be clear that the name of an entity is an identifier while the address of an entity is a locator which is the name or address of the location to which the entity is bound. Discovery is a general term that could mean different things in different contexts depending on what is being discovered. In a broad sense, discovery is the process of finding resources of interest to the seeker starting with some expression of what is needed. Generally, the seeker does not know at time of discovery whether a resource exists, who provides it, or where it is located. There are

Chapter 6. Background on Naming and Discovery

two main activities involved in discovery: the first step is announcing or advertising the resource availability (along with state about the resource) in some context. This step creates the bindings between the resource and some other object or attribute of interest in the particular context. The second step is discovering the resource starting with some expression of what is needed. In our network model, the resource to be discovered is a named entity and discovering a route/path to the entity is the end goal using the entity name. Discovering a route to some object (whether an entity or a location) is termed *routing*. Routing utilizes the underlying forwarding function when discovering the route(s). Identifier-based discovery, or simply discovery hereafter, is a set-valued function defined as $df_u : \mathcal{I} \rightarrow V$ which maps an identifier to a set of locations to which the identified object is bound. Discovery subsumes routing. To see how, we distinguish two scenarios as follows:

- Scenario 1: the forwarding function ff is aware of identifiers, then identifier-based discovery is essentially routing on the identifiers (*e.g.*, [37, 39]).
- Scenario 2: the forwarding function ff is not aware of identifiers, then discovery is a two-step process: first discover the mapping from the identifier to some locator(s), and then discover the route(s) based on the locator(s). Only the second step is routing. In this case, discovery subsumes routing.

Note that the forwarding function may be aware of identifiers only (*e.g.*, [37]), of locators only (*e.g.*, [42] and IP), or of both (*e.g.*, [39]).

Generally, the process of identifier-based path discovery involves a search or discovery query that is forwarded based on a series of calls “forward to *next node* that should have more (\geq) information about the named destination” starting at a source node. Discovery schemes in large-scale networks require maintaining distributed state about the identifier space. Note here that by considering path discovery that involves distributed in-network state, we are clearly restricting the discussion to

stateful routing (proactive) schemes which seem to be more common in large-scale networks. Reactive or on-demand discovery schemes generally involve flooding which renders them less efficient to implement at large scales. From an algorithmic standpoint, a generalized discovery scheme provides the following operations:

- Discovery operations: encapsulate the interface that the entities use to communicate with the mechanism and include two operations:
 - *join(i , $level$)*: allows an entity i to request a discovery service possibly expressing a desired service level (and potentially her valuation of some service level as shall become clearer in the next chapter).
 - *discover(i , j)*: allows an entity i to discover entity j .
- Service operations: are implemented on the service nodes and dictate a set of rules for maintaining state about the namespace and for handling the above queries.

6.4 Exploring the Design Space

The design of discovery schemes aims at satisfying a set of requirements and is based on a set of assumptions. Some of the common requirements we observe in the literature include *efficiency*, *scalability*, *dynamism support*, *user-control*, *robustness*, *resilience*, *manageability*, *trust*, *security*, *privacy and anonymity*, *accountability*, *economic requirements*, etc. In terms of assumptions, the most common ones address the underlying graph structure, and model. For example, assumptions about the graph model include general ones such as hierarchical, *scale-free*, or *small-world* assumptions, or more specific structural assumptions of underlying metric embeddings. Other assumptions specify whether graph is static or dynamic. In order to illustrate the definitions of locator and identifier and to highlight some of the inherent tradeoffs

in the design of discovery schemes, we explore two architectural designs: the TCP/IP Internet, and compact routing [44, 45] and we examine the tradeoffs between *scalability*, *efficiency*, and *dynamism support*.

6.4.1 TCP/IP Internet

The IP address on the Internet names two objects simultaneously: the node object and the attachment point [141]. To be more precise, the IP address names the end-point of a TCP connection using {IP address, port} [11]. Hence, the IP address is simultaneously an identifier and a locator. This design has advantages and disadvantages. Our imaginary plane abstraction may be directly mapped to the TCP/IP Internet by drawing this plane between the network layer and the transport layer. The inter-network routing system resides below the plane while the entities (end-points) reside above the plane. The original design of the routing system assumes aggressive aggregation of the addressing space where locators follow the hierarchical topology. This allows the routing system to scale as long as the topology and the addressing structure closely follow each other. However, recently provider-independent addressing, multi-homing, and traffic engineering practices have put strains on the routing system [12]. Multi-homing for example (a customer connecting to multiple providers), requires that a customer AS advertise a provider supplied prefix through its multiple providers. For example, assume the provider's prefix is 192.0.0.0/8 and part of it is delegated to the customer, say 192.10.0.0/16. This addressing structure follows the hierarchical topology (customer-provider). In the event that the customer connects to another provider for reasons of redundancy, the customer now advertises the prefix 192.10.0.0/16 through the other provider. This requires the first provider, who originally aggregated the customer's prefix, to de-aggregate the general prefix 192.0.0.0/8 and advertise the sub-prefix 192.10.0.0/16 as well (or otherwise the more specific route through the second provider will always be used to reach the cus-

tomers!). The outcome is that the global routing table will now contain two prefixes due to de-aggregation. Provider-Independent (PI) prefixes as well may not be aggregated and each prefix requires $O(n)$ ⁴ state in the global routing table (the BGP Routing Information Base (RIB) [42]) since every router in the Default Free Zone (DFZ) must keep state about the prefix. This deviation from the original design causes serious scalability issues with the routing system which again may only scale with aggressive aggregation. This reality is exacerbated by the fact that the number of BGP prefixes in the global routing table/RIB is increasing exponentially at a rate of roughly 100,000 entries every 2 years and is expected to reach a total of 388,000 entries in 2011 [145]. Remedies to the scalability problem, such as [146, 147], at best scale linearly. Hence, while the original design of the routing system is scalable, the current reality is that it is not scalable.

In terms of efficiency, since the entity name is the same as the location name, we focus on the efficiency of the routing system. The Border Gateway Protocol (BGP) is a policy-based path-vector protocol and is the de-facto protocol for Internet inter-domain routing. The protocol's specification [42] was initially intended to empower domains with control over *route selection* (which path or route to pick among multiple advertised routes to a destination), and *route propagation* (who to export the route to among an AS's direct neighbors) [148]. Route reachability information is broadcasted in BGP and nodes pick the routes that they value most which are not necessarily the shortest routes. Routing is intended to allow for a rich set of AS policies to be implemented [149, 70]. For example, if all ASes agree to implement shortest path, then BGP allows for it (but that is not the goal neither the reality as we shall explain in chapter 9). In [13], it has been shown that hierarchical aggregation schemes (such as BGP [42]) are not optimal⁵ when it comes to the Internet

⁴ n being the number of nodes.

⁵Hierarchical addressing is not efficient since it requires large distances between nodes for efficiency.

Chapter 6. Background on Naming and Discovery

topology which is *scale-free* (i.e. power law [150] degree distribution) and obeys *small world* (with more than 80% of AS pairs 2-4 hops apart).

Finally, in terms of dynamism we distinguish topology dynamics versus entity dynamics. BGP is designed to handle topology dynamics gracefully and to route around link or node failures. However, any such failure requires $O(n)$ communication which does not scale. For example, a link failure results in a BGP route withdraw message(s) that is propagated to all the DFZ. BGP dynamics are a major concern to the scalability of the Internet routing system [12]. In terms of entity dynamics, overloading of the entity name and the location name with the same IP address hinders mobility and portability [11]. If an entity wishes to move to a new location while keeping its name, then the permanent name of the new location must be changed which is very costly. This is perhaps the main drawback of using the same name for both location and entity objects. To remedy the problem, one of the recurring themes in the community is the need to separate the entity's identifier from its locator to enhance mobility (entity can move while maintaining the identifier) and security (trust information may be associated with the object at all levels). Check for example [11, 26, 27, 28, 29, 151, 144] for incremental proposals, and [30, 31, 32] for architectural approaches and considerations.

6.4.2 Compact Routing

We introduce the compact routing problem here which we shall refer to throughout the rest of the dissertation, and we review its two variants: *labeled* versus *name-independent* compact routing. We refer the reader to [44, 45] for surveys on the topic. Given a weighted, undirected graph $G = (V, E, w)$, a compact routing scheme routes messages between nodes with the goal of minimizing *stretch*, and *space*. Stretch is defined as the ratio between the cost of the path taken by the routing scheme, to the

minimum cost path. The maximum of the ratio for all source-destination pairs in G is the stretch. Space is the routing state (in bits) stored per node. Clearly, there is a tradeoff between space and stretch: the more information nodes keep about the graph (hence the more space), the better path they can choose for routing (hence smaller stretch). For example, a trivial stretch 1 scheme may be devised when each node keeps $O(n)$ state about the rest of the network. For universal graphs, Gavaille et al. [44] gave a lower bound dictating that the stretch is at least 3 when each node keeps $o(n)$ bits, which is the best that any routing scheme may achieve. The first variant of compact routing is labeled (or name-dependent) compact routing (LCR) which allows the designer of the routing scheme to pick node identifiers to better suit the routing scheme, giving her more control by potentially embedding topological information into the identifiers. On the other hand, name-independent compact routing (NICR), first distinguished in [38], allows nodes to be named arbitrarily, making the stretch at best larger (or equal) than that with labeled routing. This last observation is intuitive, since the name-independent routing scheme has to discover the additional binding between the name and the route before routing - a step that increases stretch [39]. The compact routing framework models a single abstract object, the node, and does not distinguish between locations and entities. In this sense, there is room for different interpretations of what is being named when we say the “node name”. Is it the default entity or the location object? In LCR, the name which is referred to as the label names a location (a route) and discovery is simply discovering a path to the named location starting with the label (for example a route on a spanning tree). In NICR, on the other hand, we view the node name as an identifier (of the default entity) and a locator at the same time (similar to the IP address). When the NICR scheme builds on an underlying LCR scheme, then discovery involves two steps: first discover the mapping from the name to a routing label and then from the label to the route.

Compact routing distinguishes either (1) *single-source* routing (simply routing

Chapter 6. Background on Naming and Discovery

from a designated source node to all destinations - generally performed using a tree where source = root), or (2) *any-pair* routing (any node should find any other node). The general approach in the present compact routing schemes is to split the namespace into compact sets (of sub-linear size example \sqrt{n}). Each one of those sets forms a group (or a color) and the state about each group is distributed throughout the network. Looking up a name requires identifying the group to which the name belongs, and contacting a group representative who generally knows about the locators of all names in the group. For example, in the single-source routing case on trees, any graph degenerates into a spanning tree rooted at the source. Each of the neighboring nodes of the root node will maintain state about a distinct group allowing the root to lookup any destination node by referring to the respective group representative node which is one of its neighbors. Check Arias et al. [152] for such a stretch-3 scheme. In the same sense, in any-pair routing on general graphs, where any node needs to be able to lookup any other node, each node will know about the group representatives that generally belong to its neighborhood. For example in the optimal stretch-3 routing scheme by Abraham et al. [39] on general graphs which we shall elaborate on shortly (and the previous stretch-5 by Arias et al. [152]), this is exactly the case. Each neighborhood is *fully colored* and a node that represents the group/color knows about all nodes in the group. The challenge with such schemes lies in the means to distribute the group responsibilities to nodes (i.e. which node represents what groups?) such that (1) each neighborhood has at least a representative for each group, and (2) no node represents more than a logarithmic number of groups. It has been shown in [152] [Lemma 3.1] that such assignment exists and is computable in polynomial time.

The space vs. stretch tradeoff in the routing system may be directly associated with scalability vs. efficiency, respectively. While the compact routing framework quantifies the tradeoff between scalability and efficiency and presents several schemes along the spectrum, its major disadvantage lies in the fact that it is not concerned

with the dynamics of the network (for example node churn). The compact routing schemes consider a static graph on which a data structure is constructed, ignoring the construction algorithm and cost of constructing the data structure. The whole state might need to be re-constructed for a single node join which makes compact routing schemes in general unattractive for dynamic networks. As we shall see in the next chapter, our interest in compact routing is primarily due to the mathematical bounds imposed on space and stretch which supports our service differentiation concept. Before concluding this section, we present a concrete example that illustrates the NICR design problem. For optimal LCR schemes, check the Thorup-Zwick (TZ) scheme [153] which scales infinitely on universal graphs, and the Brady-Cowen LCR scheme which is specialized for power law graphs and scales infinitely [154]. Abraham et. al [39] developed a universal name-independent compact routing scheme that is optimal in the strict sense, i.e., requires $O(\sqrt{n})$ space with stretch 3. Recall that this is proven to be the lower bound achievable by **any** compact routing scheme. This result is interesting as it proves that choosing arbitrary (flat) locators does not necessarily degrade the performance (scalability and efficiency) of the routing scheme. A sketch of Abraham's routing scheme is provided next:

The Abraham Scheme

Briefly, the scheme operates as follows on a graph $G = (V, E, w)$: each node u keeps track of its vicinity ball $B_k(u)$ that includes the set of k closest ⁶ nodes to u in G . The value of k is picked to be $8\sqrt{n}\log n$. Each node u has a color, $c(u)$, assigned from a set of \sqrt{n} colors. The node's color is determined by hashing its name (which may be arbitrary), and picking the first $\log \sqrt{n}$ bits from the hash $h(u)$ as the color. The number of nodes belonging to a particular color set is shown to be $\leq 2\sqrt{n}$. Any

⁶Closeness is based on distance measure $d(u, v)$ which is the cost of a path from u to v , i.e., sum of weights on edges.

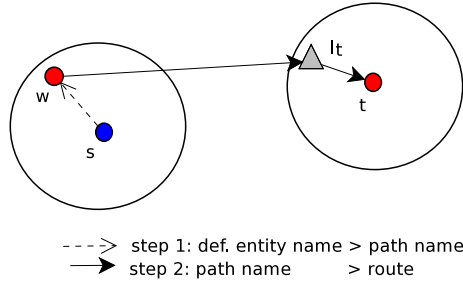


Figure 6.1: Sketch of virtual and physical routing in Abraham name-independent compact routing scheme.

one of the color sets is chosen to be the landmark set L . Based on the balancing assumption which results from hashing, each node u will have a landmark node l_u in its vicinity. In Figure 6.1, we show the vicinities of two nodes: a source node s with blue color (i.e. $c(u) = \text{blue}$) and a red destination node t as well as t 's landmark node l_t . The routing scheme builds on optimal labeled compact routing in trees, which is shown to be performed optimally using $O(\log^2 n / \log \log n)$ space [155] per node. Hence, in terms of routing state (space), each node u maintains the following:

- 1) for each landmark node l , u 's label for the minimum spanning tree (MST) rooted at l requiring $O(\sqrt{n})$ space (note that routing on those trees is optimal - stretch 1);
- 2) for each node $v \in B(u)$, node u 's label in the MST rooted at v requiring $O(\sqrt{n})$ space; and
- 3) for each node v having the same color as u , i.e., $h(v) = c(u)$, l_v 's and v 's labels in the MST rooted at l_v requiring $O(\sqrt{n})$ space but no additional MSTs ⁷.

Given this $O(\sqrt{n})$ space at each node, it can be shown that routing is optimal with stretch 3 requiring message header re-writes [39]. Figure 6.1 shows how node s routes optimally towards node t , which has a different color just by knowing t 's flat name denoted as $\langle t \rangle$. Note that names for the nodes are globally unique and are

⁷Note that the size of a neighborhood $B(u)$ is $4\alpha\sqrt{n}\log(n)$ which is very large, i.e., for $n \leq 65,000$, $|B(u)| > n$ and hence n has to be very large or otherwise each node will have to know about every other node. Hence, Abraham scheme is not very useful for AS level graph where $n \approx 10,000$.

picked arbitrarily from the integer set $\{1, 2, \dots, n\}$. The name of a node is independent of topology, whereas its label in some MST is topology dependent. Starting with the identifier $\langle t \rangle$, discovery involves two steps as follows: find a node w in my vicinity that has same color as t (dotted arrow in Figure 6.1). Node w is guaranteed to have a binding between the t 's identifier and a locator (the locator is a label in the MST rooted at l_t). Step 2 (solid arrows in Figure 6.1) involves optimally routing on the MST.

This scheme focuses at naming default objects (or nodes) with unique names. Extending Abraham's algorithm to support entities as Distributed Hash Tables (DHTs) is straightforward as described in [143]. Two variants of the DHT problem are distinguished depending on whether the designer can pick the nodes on which entities may be hosted or can not pick the location. In the first variant ⁸, an object o is hosted on a node u such that $c(u) = h(o)$ (u has the same color as o), and $\langle u \rangle$ is closest to $\langle o \rangle$. It is easy to show that routing towards any object o is optimal in the constructed DHT. In the other variant of the DHT problem, the designer does not have control over the placement of o . Generally, such model is employed for *locality-aware* closest copy routing. The concept of *locality-aware* means that the cost of locating an object o is proportional to the distance to the closest copy of o . Again, it is shown in [143] that the Abraham routing scheme may be extended incurring $O(\sqrt{n})$ pointers in the DHT per object name and maintaining scalability.

6.5 Conclusion

In this chapter, we have formalized the definitions of identifier, and locator. According to our definitions, neither the association between a name and its form, nor between the name and some system are relevant in making the distinction between

⁸This is similar to structured P2P networks - DHTs - of single copy objects (*e.g.*, [100, 156]).

Chapter 6. Background on Naming and Discovery

an identifier and a locator. It is only the association between the name and the object that is being named that qualifies the distinction. We defined the discovery function to be the process of discovering a path to a named entity. To illustrate our definitions, we examined two architectural designs that generally lead to confusion: the TCP/IP Internet, and the compact routing architectures. We focused on the inherent tradeoffs between scalability, efficiency and dynamics when it comes to the design of discovery schemes. Finally, we have introduced the compact routing problem, and the concept of stretch, which will be relevant to the discussion in chapter 7 when we introduce discovery service differentiation. This chapter is necessary to eliminate the terminology confusion and to set the stage for further investigation of the topic throughout the rest of the dissertation.

Chapter 7

Discovery Service Differentiation

7.1 Introduction and Motivation

In the previous chapter we have elaborated on the definitions of the terms identifier, and locator and we have introduced discovery. Briefly, identifiers and locators are names of different objects. We isolated two different types of objects on the network, the entities and the locations to which they are bound. Identifiers name entities whereas locators name locations. Identifier-based discovery (simply referred to as discovery hereafter) is a core network service aimed at discovering a network path to an identified entity. Discovery is usually the first step in communication, even before a path to the destination entity is established. Given an identifier of some entity on the network, discovering a path to the entity could either utilize mapping/resolution where the identifier is mapped to some locator¹ (see for example [36, 28, 144], and the Domain Name System (DNS)), or it could utilize routing-on-identifiers (see [34, 37, 38, 39] etc.). In either case however, an underlying routing scheme that routes on locators typically exists and is utilized after a path has been

¹The terms locator and label are used interchangeably in this context.

Chapter 7. Discovery Service Differentiation

discovered for efficient communication. Note that the terms identifier and locator are both names at different layers of abstraction. We differentiate the two terms only after we fix an upper layer: an identifier at the upper layer maps into a locator which is an address relative to the upper layer.

This chapter is concerned specifically with the differentiation of the discovery service. A named entity (such as a node or service), referred to as a player, demands to be discoverable by the rest of the network. A discovery scheme provides such service to the players. We define the *discovery level* to be a measure of “how discoverable” a player is by the rest of the network. This is “how easy” it is for the network to discover the player not the opposite. The performance of discovery, or the discovery level, could significantly affect the player’s business model especially in time-sensitive application contexts. If discovering an entity takes a significant time relative to the entity’s delivery/download time, the experience of the requesting user suffers. As an example, when no caching is involved, the DNS resolution latency comprises a significant part of the total latency to download a webpage (10-30 %) [40, 41]. This overhead becomes more noticeable in Content Distribution Networks (CDNs), where content objects are extensively replicated throughout the network closer to the user and the discovery (or resolution) could potentially become the bottleneck. Traditionally, the design of discovery schemes has assumed that all players have the same discovery performance requirements, thus resulting in homogeneous demand. In such a setting, the discovery schemes deliver a discovery service that is oblivious to the actual, possibly heterogeneous, discovery requirements - and valuations - of the different players. In reality however, the CNN site will likely value a higher discovery level more than a generic residential site. The main question posed in this chapter is therefore the following: should the design of discovery schemes account for service differentiation? We answer this question by introducing the *Multi-Level Discovery* (MLD) framework which is concerned with the design of discovery schemes that can provide different service levels to different sets of players. To further motivate the

problem, we note that on the current Internet, Akamai provides such an expedited resolution service [125]. However, the service which is based on DNS suffers from the same pitfalls of the latter (expensive first lookup and critical dependence on caching) and tightly couples the content distribution provider with the resolution service provider.

The first question we ask is whether differentiated discovery is algorithmically feasible i.e. is it possible to devise a scheme that is scalable and that provides different levels of service to different players. Along this dimension, we define the algorithmic problem in section 7.2, and we present a proof-of-concept MLD scheme in section 7.3 along with an analysis of its scalability properties.

7.2 What is Multi-Level Discovery (MLD)?

We start by providing a generic definition of the MLD problem. The problem specifics will depend on the context, mainly the design assumptions and requirements.

Definition 1. *Multi-level discovery (MLD) problem statement: Given a graph $G = (V, E)$, a set of nodes with unique identifiers (identifier of node i is simply $\langle i \rangle$), set of m discovery levels where each node is associated with some level $l \in \Lambda$ ², and possibly some underlying routing function f_p that routes on locators, devise a discovery scheme that routes on identifiers. The set Λ of possible discovery levels is known to all nodes. The scheme is expected to deliver to each node i in G its requested discovery level $l \in \Lambda$.*

The main challenges inherent to the MLD problem arise from the following requirements:

²When the set of discovery levels is discrete, a level becomes a “class” of service.

- different levels of service must be supported by the same scheme, and
- the discovery level of a destination $\langle t \rangle$ is unknown at the time of discovery. The challenge here is that information about the discovery level of the destination is to be discovered as well by the scheme and is not known *a priori*. The only attribute that is known *a priori* is the identifier.

7.3 A Multi-Level Discovery Scheme

A traditional class of discovery schemes that satisfies the single default entity per node assumption and that has been extensively investigated in the research community is the general Name Independent Compact Routing (NICR) problem first introduced in [38]. We have introduced the NICR problem previously in chapter 6 and we have reviewed Abraham’s optimal NICR scheme on universal graphs [39]. NICR is of particular interest to this section and we shall extend the framework for implementing a MLD scheme. We restrict our attention to trees rather than universal graphs. More specifically, we extend Laing’s NICR scheme [157] which operates on top of the optimal Thorup-Zwick labeled routing scheme on trees [153]. The latter represents a locator-based routing function over which the identifier-based discovery scheme is implemented.

7.3.1 Background: NICR scheme on trees

A name-independent compact routing scheme on trees (NICRT) is developed by Laing [157] with a space/stretch tradeoff based on a parameter k . The scheme achieves stretch $2^k - 1$ for a space requirement of $\tilde{O}(k^2 n^{1/k})$, where n is the number of nodes. From a high level perspective, the tradeoff is achieved by asking each node to know about a set Σ^i of nodes ($|\Sigma^i| = n^{i/k}$) at concentric circles or neighborhoods

Chapter 7. Discovery Service Differentiation

$N^i, 0 \leq i \leq k - 1$ from itself. Routing towards a destination d proceeds through prefix matching of d 's identifier $\langle d \rangle$ represented in base $n^{1/k}$ (denoted by $\langle d \rangle_{n^{1/k}}$). Delivery is guaranteed in at most k hops i.e. by matching the k letters of d 's identifier base $n^{1/k}$. The main idea is that as the value k increases (i.e. as the number of concentric circles or layers increases), a node will keep less information about the rest of the network but the stretch which is directly proportional to the number of layers will increase. On the other hand, as k decreases (i.e. fewer layers), a node will keep more information about the rest of the network and the stretch decreases accordingly.

Laing's scheme is based on a coloring theorem for trees. The coloring theorem states that any tree with n nodes can be colored with q colors such that every neighborhood $N_q(v)$ of size q (for every node $v \in V$) is distinctly colored i.e. each node in $N_q(v)$ has a unique distinct color from the set of colors $[q]$ (check [157]). For reference, we include the theorem here:

Theorem 1. [157] *Let $T = (V, E)$ be a tree with $n \geq q$ nodes ($q \geq 1$), and positive edge weights. There exists a function $c : V \rightarrow [q]$ such that $\forall v \in V, N_q(v)$ is fully-colored.*

The theorem is used in the NICRT scheme to uniquely color neighborhoods $N^i(v)$ of size $n^{i/k}$ at each layer $i, 0 \dots k - 1$.

Laing's scheme works as follows: Given a tree $T = (V, E, w)$, and a $k \geq 1$, multiple layers of coloring are assigned to nodes as follows: at layer $1 \leq i \leq k - 1$, T is fully colored with Σ^i colors where $|\Sigma^i| = n^{i/k}$ and $\Sigma = \{0, 1, \dots, n^{1/k} - 1\}$ is the alphabet. Note that the neighborhood of a node v is denoted by $N^i(v)$ and is the set of $n^{i/k}$ closest nodes to v including the latter. Hence $|\Sigma^i| = |N^i|$ and the coloring theorem achieves a full coloring. Each node $u \in V$ is hence assigned a unique color $c_i(u)$ at layer i , where $c_i(u) \in \Sigma^i$. In addition to the $k - 1$ colors node u obtains, it

has its unique identifier $\langle u \rangle$ picked from the set $\{0, \dots, n-1\}$ and represented in base $n^{1/k}$ and padded to the left with zeroes. Thus $|\langle u \rangle| = k$.

Storage: Each node u has an identifier $\langle u \rangle$ and $k-1$ colors $c_i(u)$. Denote by $\sigma_i(u)$ the length i prefix of $\langle u \rangle$. In addition to the labeled compact routing table information of [153]³, node u creates its routing table according to Algorithm 1.

Routing: In terms of routing to some destination t with identifier $\langle t \rangle$ starting at some source s , routing proceeds as indicated in Algorithm 2.

Note in Algorithm 2 that each next hop (i.e. v_{i+1}) is guaranteed to belong to $N^{i+1}(v_i)$. Note as well that the only node that matches $\sigma_k(t)$ is the node whose identifier is $\langle t \rangle$ which guarantees delivery [157].

Algorithm 1 Routing table construction for node u

```

1: for each layer  $i, 0 \dots k-1$  do
2:   Let  $\lambda = \{c_i(u), \sigma_i(u)\}$ , where  $c_0(u)$  and  $\sigma_0(u)$  are the empty string  $\epsilon$ 
3:   for each  $\tau \in \Sigma$  do
4:     store label of closest node  $v$  to  $u$  that satisfies  $c_{i+1}(v) = \lambda\tau$  or  $\sigma_{i+1}(v) = \lambda\tau$ 
5:   end for
6: end for

```

Algorithm 2 Routing to $\langle t \rangle$

```

1: let  $v_0 = s$ 
2: for each layer  $i, 0 \dots k-1$  do
3:   route to node  $v_{i+1}$  which is the closest node to  $v_i$  that matches  $\sigma_{i+1}(t)$  i.e. node  $v_{i+1}$ 
      satisfies  $c_{i+1}(v_{i+1}) = \sigma_{i+1}(t)$  or  $\sigma_{i+1}(v_{i+1}) = \sigma_{i+1}(t)$ 
4: end for

```

³This information is used for optimal stretch-1 routing based on locators (topological labels).

7.3.2 Extending Laing scheme to support MLD

In the preceding scheme, the effect of the parameter k was to control the space/stretch tradeoff achieving stretch $2^k - 1$ for a space requirement of $\tilde{O}(k^2 n^{1/k})$. In this section, we extend Laing's scheme by allowing multiple stretch levels (or multiple values of k) on the same tree T for different sets of nodes. Discovery levels will correspond to values of k in Laing's scheme which directly determines the stretch.

More clearly, we assume the existence of a set $K = \{k_1, \dots, k_m\}$ ($m = |\Lambda|$) of stretch levels ordered in ascending order with $\Lambda \subseteq \mathbb{Z}^+$. Assume also without loss of generality that n is a k_m th power and that $k_1 \geq 2$. Each k_i corresponds to a discovery level $l = \frac{1}{2^{k_i}-1}, l \in [0, 1]$ and we assume that $m = |\Lambda| = O(n^{\frac{1}{k_m}})$. The main idea that we shall use for extending Laing's algorithm to support multiple discovery levels on the same tree T introduces ACCELERATE tables that expedite discovery/routing for nodes that demand higher discovery levels. The extended scheme starts by providing the lowest discovery level ($\frac{1}{2^{k_m}-1}$) to all nodes by constructing Laing scheme for $k = k_m$. The pseudocode for construction of the routing tables is listed in Algorithm 3. Lines 6, 17 in Algorithm 3 and lines 5, 6 in Algorithm 4 encapsulate the main logic for expedited discovery.

In terms of routing to destination $\langle t \rangle$ using the extended scheme, we extend routing Algorithm 2 as depicted in Algorithm 4 given that each node knows the set of stretch levels $k_j, j = 1 \dots m$.

Analysis: It can be easily verified that delivery is guaranteed as well as $d(v_i, v_{i+1}) \leq 2^i d(s, t)$ in the extended algorithms (check [158]). In order to maintain the sub-linear space requirements at each node, the extra state maintained at each node for discovering higher level nodes must be less than a constant factor of $k^2 n^{\frac{1}{k}}$. First, at line 15 of Algorithm 3, in the worst case there are at most $n^{1-\frac{s}{k_m}}$ nodes in D_{k_j} that have the same length s prefix (when $|D_{k_j}| = n$) i.e. that can potentially introduce state on the

same set of nodes B_s . Thus the maximum increase in any node's routing table size is $m \cdot n^{1-\frac{s}{k_m}}$. We have already assumed that the total number of levels $m = O(n^{\frac{1}{k_m}})$. Formally, in order to maintain sub-linear space at each node, the following condition must hold: $n^{1-\frac{s}{k_m}} \leq \alpha k_m^2 n^{\frac{1}{k_m}}$ for some large constant α , or $s \geq k_m(1 - \frac{\log \alpha k_m^2}{\log n}) - 1$. This constraint must hold when choosing the set of possible discovery levels Λ (and hence the respective set K) in order for the extended routing scheme to satisfy the sublinear space requirement inherent to compact routing design.

7.4 Discussion and Conclusion

The MLD framework allows for discovery service differentiation. We have defined the problem, motivated it, and demonstrated its algorithmic feasibility in the context of NICR. As mentioned earlier in chapter 6, the major disadvantage inherent to most compact routing schemes is the fact that they are not concerned with the dynamics of the network and particularly with node churn. The schemes consider a static graph on which a data structure is constructed and do not worry about the construction algorithm and cost. However, our interest in compact routing in this chapter is primarily due to the mathematical bounds imposed on space and stretch which support our discovery level concept by providing guarantees on levels of performance. While we have studied the problem for NICR, differentiation of discovery is important as well in the context of the TCP/IP Internet. All current discovery schemes (such as DNS) suffer from the same problem: performance of discovery. If we ever think of using domain names as endpoint identifiers in the TCP/IP Internet, the impact of discovery (i.e. the first mapping from domain name to IP address) becomes of great importance. This is due to the fact that TCP treats the first packet as representative of congestion. We would like to note that an initial version of this chapter appeared in [159].

Chapter 7. Discovery Service Differentiation

Finally, in this chapter we have focused on the algorithmic feasibility questions. Notice however that there is a non-trivial cost associated with being discoverable. This could be the cost of distributing and maintaining information (state) about the identifiers to provide a certain discovery level. Hence, the second challenge is that of providing an economic model that accounts for cost and valuation in the design of discovery mechanisms. The next two chapters (chapters 8 and 9) are dedicated to studying the economic dimension. By adding an economic dimension to the discovery design space, we hope to gain more knowledge about the complex design decisions pertaining to naming and discovery in networks, and to be able to design discovery mechanisms that are suitable for a future Internet.

Algorithm 3 Extended table construction for node u

```

1: Let  $K' = \{k_1 - 2, \dots, k_{m-1} - 2\}$ 
2: for each layer  $i, 0 \dots k_m - 1$  do
3:   Let  $\lambda = \{c_i(u), \sigma_i(u)\}$ , where  $c_0(u)$  and  $\sigma_0(u)$  are the empty string  $\epsilon$ 
4:   for each  $\tau \in \Sigma$  do
5:     if  $i \in K'$  then
6:       store label of closest node  $v$  to  $u$  that satisfies  $c_{i+1}(v) = \lambda\tau$ 
7:       store label of closest node  $w$  to  $u$  that satisfies  $\sigma_{i+1}(w) = \lambda\tau$  only if  $w \in N^{i+1}(u)$ 
8:     else
9:       store label of closest node  $v$  to  $u$  that satisfies  $c_{i+1}(v) = \lambda\tau$  or  $\sigma_{i+1}(v) = \lambda\tau$ 
10:    end if
11:  end for
12: end for
    {Construct the ACCELERATE table}
13: for each level  $k_j, j : m - 1$  downto 1 do
14:   Let  $s = k_j - 1$ 
15:   Let  $D_{k_j}$  be set of nodes requiring level  $k_j$ 
16:   for each node  $u \in D_{k_j}$  do
17:     Let  $B_s$  be set of nodes whose color at layer  $s$  is  $\sigma_s(u)$ 
18:     Add extra pointer  $\{ \langle u \rangle_{n^{1/k_m}} \rightarrow label(u) \}$  at each node  $v \in B_s$ 
19:   end for
20: end for

```

Algorithm 4 Routing to $\langle t \rangle$ using extended scheme

```

1: let  $v_0 = s$ 
2: for each layer  $i, 0 \dots k_m - 1$  do
3:   if  $\langle t \rangle \in \text{ACCELERATE table}$  then
4:     route directly to  $t$  using  $\text{label}(t)$ 
5:   else if  $i = k_j - 2$  for any  $j = 1 \dots m$  then
6:     route to node  $v_{i+1}$  which is the closest node to  $v_i$  such that  $c_{i+1}(v_{i+1}) = \sigma_{i+1}(t)$ 
       or  $\langle v_{i+1} \rangle = \langle t \rangle$ 
7:   else
8:     route to node  $v_{i+1}$  which is the closest node to  $v_i$  that matches  $\sigma_{i+1}(t)$  i.e. node
        $v_{i+1}$  satisfies  $c_{i+1}(v_{i+1}) = \sigma_{i+1}(t)$  or  $\sigma_{i+1}(v_{i+1}) = \sigma_{i+1}(t)$ 
9:   end if
10: end for

```

Chapter 8

On the Economics of Identifier-based Discovery

8.1 Introduction

Traditionally, the design process in the context of the Internet has focused on sources of value as they relate to performance, robustness, resilience, reliability, etc. with less emphasis on the socio-economical dynamics that underly the latter. The value of any new design in the new era does not solely depend on performance and must take into account the complex social and economic interactions and incentives of the agents using the design if success is to be reached [61, 62]. Check [61] for an interesting overview of several tools that are important in bridging computer science and economics to better understand the complex socio-economic interactions in the context of the Internet, and [62] for an interesting overview of several of the problems and applications arising at the interface between information and networks.

The previous chapters have motivated the importance of naming and discovery in computer networks. Hereafter we assume that a naming or identification system

Chapter 8. On the Economics of Identifier-based Discovery

for a large scale network, the Internet mainly, is required given the network's mobile and ubiquitous usage models. For example, on the Internet, this translates into either designing a new system or enhancing the current ones (for example DNS). While there is a rich literature on applying game theory and economics models to Internet games, we find in the networking literature a number of proposals for Internet discovery schemes (and id routing) requiring significant coordination among selfish users while ignoring the economic aspects that may possibly render them infeasible or inefficient (and we shall give several examples of such system or proposals later in section 8.3). In a future Internet in which domains or Autonomous Systems (ASes) are self-interested, welfare-maximizing agents, the design of any identifier-based discovery scheme could benefit from establishing the right economic models. The problem on the Internet specifically is exacerbated as there are multiple layers of identification managed by different systems, mainly DNS [116] at the application and the Border Gateway Protocol (BGP) [42] at the network layer.

In chapter 6, we have introduced discovery in large-scale networks. We have additionally defined the *multi-level discovery* framework in chapter 7 which is concerned with the design of discovery schemes that can provide different service levels to different sets of nodes. Obviously, there is a cost associated with being discoverable. This could be the cost of distributing and maintaining information (state) about the identifiers. In current schemes, the discovery demand is actually insensitive to cost since no cost structure exists and hence demand flattens out to a homogeneous level. Accounting for and sharing the cost of discovery is an interesting problem whose absence in current path discovery schemes has led to critical economic and scalability concerns. As an example, the Internet's BGP [42] control plane functionality is oblivious to cost. A BGP speaker that advertises a provider-independent prefix (identifier) does not pay for the cost of being discoverable. Notice here that BGP with its control and forwarding planes represents a discovery scheme on prefixes which are technically flat identifiers in a largely de-aggregated namespace. Hereafter, we refer

to this form of BGP as BGP-DA for De-Aggregation. This problem becomes more important in settings where the state (and the cost) is incurred by service nodes that are not themselves players.¹ In this case, the cost must be paid for or else the service nodes will have no incentive to implement the discovery service. Hence, *we conjecture that a discovery scheme should be aware of incentives and cost necessitating that players/nodes pay for the cost of getting the service.* Providing such a service while accounting for the cost and making sure that the incentives of the players are aligned is the general economic problem that we frame in this chapter.

The rest of the chapter is organized as follows: first we review background material in section 8.2. Specifically, we motivate the notion of strategic interactions on networks by presenting three games in section 8.2.1 that we shall refer to throughout the discussion. We also distinguish between the search function and receiver-based discovery function in section 8.2.2. Distinguishing the two functions is important to frame our work. Section 8.3 presents a taxonomy of discovery schemes based on their business models. Finally, section 8.4 presents our thoughts on suitable economic models for the different discovery models highlighted in the taxonomy before concluding in section 8.5.

8.2 Background

8.2.1 Networks and Strategic Behavior

Game theory is a fundamental mathematical tool for understanding the strategic interactions among selfish network agents, particularly on the Internet over which self-interested agents (e.g., ASes) interact. The theory provides several solution con-

¹Service nodes implement the discovery service. Players are customers of the discovery service or agents that wish to be discoverable.

cepts to help study games that arise in different situations and that have specific requirements and varying underlying assumptions [65]. We overview some basic ones here and we provide examples to illustrate each. The most central and widely applicable solution concept is the *pure strategy* Nash equilibrium (PSNE or NE) which could be simply thought of as a set of strategies that forms a stable solution of the game. A set of strategies for the players is termed a *strategy profile*. Under NE strategy profile, no player can move profitably (i.e., increase her payoff) by deviating from her strategy given every other player's strategy. Despite its wide applicability, the NE solution has several shortcomings in that it may not exist (and hence might require mixing), there could be multiple equilibria, and it might be computationally intractable to get to it. In this sense, the *mixed strategy* solution concept was developed by Nash to guarantee that an equilibrium will always exist in the game by mixing the player's strategies (introducing probability distributions over the pure strategies and hence rendering the strategy space a convex set). A more stringent solution concept is the *dominant strategy* solution. Unlike the pure strategy solution, a dominant strategy yields a player the highest payoff independent of the strategies of the rest of the players. Dominant strategies are very attractive solutions when they exist, and when they do not exist, game designers might try to design for them. For example, when a player's strategy is to declare some private information that is necessary to the social welfare of the game, an attractive solution would be to make the truthful revelation a dominant strategy hence making sure that the player will never have an incentive to lie. The mechanism design framework [66] provides exactly this solution allowing the mechanism "designer" to achieve a dominant strategy solution (in addition to other design goals). A extension to mechanism design, Algorithmic Mechanism Design (AMD) [64], deals with the computational complexity of the solution and Distributed AMD [67] further considers the "network complexity" in distributed settings. Several other solution concepts exist; however, we will only overview one more which is the *subgame perfect* Nash equilibrium (SPNE) which ex-

tends the one-shot NE concept to settings in which players take turns playing (e.g., player 1 plays first, then player 2 plays). In such setting, the subgame perfect NE (SPNE) becomes more “natural” as it captures the order of decision taking. Briefly, a SPNE is a NE in every subgame of the original game where a subgame could be informally defined as a portion of the game that can be independently analyzed. Note that by the formal definition of a subgame, every game is a subgame of itself and hence every SPNE is necessarily a NE. For formal definitions of the solution concepts and a comprehensive treatment of the topic, we refer the reader to [63].

How does strategy factor into networking problems? To motivate the importance of strategic behavior, we hereby present three networking applications that employ different solution concepts and that we shall refer to throughout the discussion. Our hope is that the games highlight some of the basic economic issues that are of interest to network settings and the tools that are useful in studying these settings. Note that the games we present here might not be straightforward for the unexperienced reader who we refer to [63, 65] for introductory material on the subject. The first application we present is that of “query incentive networks” and is due to Kleinberg and Raghavan [78]. The second application is that of “trading networks with price setting agents” due to Blume et al. [160]. The common aspect of the first two games is that price setting is a strategic behavior of the players which is not the case with the third application we present, “Incentive-compatible interdomain routing” due to Feigenbaum et al. [161]. Additionally, while the first two games are solely interested in studying the equilibria, the third presents a distributed mechanism that achieves the solution.

Nash Equilibria and Query Incentive Networks Game [78]

Query incentives are motivated in peer-to-peer and in social networks where some root node issues a query seeking a piece of information or a service on the network.

The seeker does not know which nodes on the network have the answer (neither does any other node) and hence the only way to find the answer is to propagate the query deeper into the network until a node with an answer is reached. In order to do so, every node needs to incentivize its direct children to propagate the query deeper where hopefully a destination node with an answer will be reached. Propagation is assumed to occur on a tree and incentives are provided by each parent in the tree to its children in the form of rewards. A node that gets offered a reward will itself offer a smaller reward to its children if it does not possess the answer hence pocketing some reward if an answer to the query is found under the node's subtree. We shall refer to this game hereafter as the QUERY-GAME and we note that this game is based on a similar game initially introduced by Li et al. [77].

Formally, each node (player) u receives a reward r from its parent and offers the same reward $f_u(r) < r$ to its children if it does not have the answer. Otherwise, if u has the answer to the query it responds to its parent with the answer. Each node holds the answer with probability $1 - p$ and on average one in every n nodes holds the answer (n is referred to as the rarity of the answer). The node's strategy is hence $f_u(r)$ which is assumed to be integer-valued and the payoff is simply $(r - f_u(r))\alpha_u(\mathbf{f})$ where $\alpha_u(\mathbf{f})$ is the probability that an answer is found in the subtree rooted at u given that node u has played f_u and every other node's strategy is given by $\mathbf{f} = \{f_v, \forall v\}$ (\mathbf{f} is a *strategy profile*). Fig. 8.1 depicts a sample game on a tree.

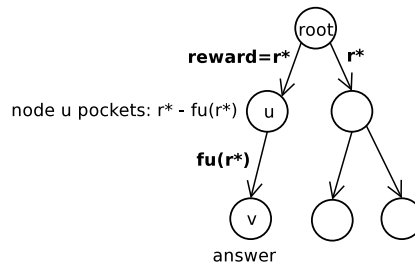


Figure 8.1: Query Incentive Game: node v has an answer to the query.

Chapter 8. *On the Economics of Identifier-based Discovery*

There are several questions that arise in such a game: How will a node act strategically to tradeoff its payoff and the probability that an answer is found in its subtree knowing that a higher promised reward potentially means higher probability of finding an answer but less payoff? How much initial investment r^* is required (as a function of the tree structure and the rarity of the answer n) in order to find an answer with high probability? The authors answer these questions in [78] by modeling a general class of branching processes parametrized on the branching factor b , where the latter is the mean number of active offsprings (or children) per node in the tree constructed using a random branching process [78] (when $b < 1$, the tree is almost surely finite while it is infinite when $b > 1$ with positive probability). When looking for the equilibria, one important point to notice in this game is the interdependency of the players' strategies as given by the tree structure - the strategy of a player will depend on the strategies of its children and so on. The authors show that the Nash equilibrium exists (and is unique with some caveats) by constructing a set of functions \mathbf{g} (a strategy profile) inductively and showing that the resulting strategy profile is indeed an equilibrium. This result simply says that there exists a stable solution to the game such that if the nodes play the strategies \mathbf{g} then no node will be able to move profitably given the strategy profile of the rest of the nodes. However, the model does not provide a recipe to get to the solution. Knowing that a solution exists, the next step is to study the breakpoint structure of rewards to be able to say something about the initial investment required (check [78] for results there). In summary, the goal of this game (and the one in [77]) is to provide incentives for query propagation in decentralized networks with uncertainty about the destination of the answer knowing that such a process could incur cost that must be paid for by someone to keep the incentives aligned. In the next game, we shall discuss a game that uses the SPNE solution.

Subgame Perfect Nash Equilibria and Trading Networks Game [160]

The next game we present is that of trading networks which despite being more motivated from a markets angle will provide several insights into networking games that involve competition. A set of sellers S wish to sell their goods to a set of buyers B indirectly through a set of traders T . While [160] studies both cases where the goods are distinguishable or not, in this brief overview we shall only focus on indistinguishable goods, i.e., a single type of good where all copies are identical. Each seller holds exactly one copy of the good initially and each seller is only interested in buying one copy of the good as well. Trade between the buyers and the sellers can only happen through a set of traders T as specified by a graph G . G specifies how sellers and buyers are connected to the traders where each edge in G connects a node in $B \cup S$ to a node in T . Sellers are assumed to have zero value for the good while each buyer j has a value θ_j for the good. Fig. 8.2 depicts such a setting where the indices i, j, t are used to refer to the sellers S , the buyers B , and the traders T , respectively.

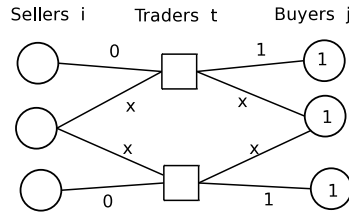


Figure 8.2: Trading Network Game: sellers S to the left (circles) connect to traders T (squares) who in turn connect to buyers B to the right (circles). The buyers' values are indicated inside the circles (1 in this case). Equilibrium bid and ask prices are shown above the links.

We shall refer to this game as the TRADE-GAME. The game aims at studying the process of strategic price setting in markets with intermediaries, and proceeds as follows: first each trader offers a bid price β_{ti} to each seller i to which it is connected,

Chapter 8. On the Economics of Identifier-based Discovery

and an ask price α_{tj} to each buyer j to which it is connected. The vector of bid/ask prices is the strategy profile of the traders. Then buyers and sellers choose among the offers they got, the traders pay the sellers the bid price and get the ask price from the buyers. If a trader gets more buyer offers than the seller offers it has, the trader will have to pay a very large penalty - so that this is not the case at equilibrium. This is so that such a scenario will never happen at equilibrium. The payoffs of the different players are as follows: a player that does not participate in a trade gets no payoff. A buyer that participates in a trade through some trader t gets a payoff of $\theta_j - \alpha_{tj}$, while a seller i that participates in a trade with trader t gets a payoff of β_{ti} (again here assuming the seller has no value for keeping the good). Finally, a trader that participates in trade with a set of buyers and sellers gets a payoff of $\sum_r (\alpha_{tj_r} - \beta_{ti_r})$ minus a penalty if more buyers than sellers accept its offer (where the index r runs for each distinct buyer, seller combination that have accepted t 's offer). It is important to notice that price setting in this game is strategic. Hence, as in the previous game, the first question to ask is how will the traders act strategically to set the market prices knowing that multiple traders could be competing for the same business, and what solution concept is most suitable to studying this game? The solution concept used in this game is the subgame perfect NE which is suitable in such a two stage game where traders play first and then buyers and sellers react. With this in mind, the next step to understanding the strategic behavior of the players (or equivalently the price setting dynamics) is to ask whether a solution (equilibrium) exists and to understand the structure of any such solution. In Fig. 8.2, the equilibrium strategies are shown above the links. Two interesting equilibrium phenomena in this game are the effects of monopoly and perfect competition. Both traders in this example make a maximum profit (of 1) from the single monopolized buyer/seller pairs that have access to one trader, while the traders make zero profit when competing for the business of the middle seller and buyer. This must be the case at equilibrium. It turns out as shown by the authors that the equilibrium always exists and that every

equilibrium is welfare maximizing (where the welfare of an outcome is simply the difference between the values of the buyers and those of the sellers). These results are shown by resorting to the primal/dual solutions of a welfare maximization linear program. In any solution, no trader will be able to make any profit unless the latter is essential for the social welfare of the game (this result captures the case where traders could have different costs and hence only the cheaper ones will be part of the equilibrium). The game (with distinguishable goods) could be directly extended to account for trading costs, i.e., where traders incur costs to perform the trade and the same results hold, i.e., a trader will be able to make profit only when the trader is crucial to the social welfare.

Mechanism Design and Interdomain Routing Game [161]

The third game we present in this section is that of interdomain routing incentives, particularly for BGP. First, we briefly overview how BGP operates after which we proceed to describe the incentive mechanism. The Internet is mainly composed of independent Autonomous Systems (ASes), or administrative domains, that must coordinate to implement a distributed routing algorithm that allows packets to be routed between the domains to reach their intended destinations. BGP is a policy-based path vector protocol and is the de-facto protocol for Internet interdomain routing. The protocol's specification [42] was initially intended to empower domains with control over *route selection* (which path or route to pick among multiple advertised routes to a destination), and *route propagation* (who to export the route to among the direct neighbors of an AS). The commercialization of the Internet quickly transformed ASes into economic entities that act selfishly when implementing their internal policies and particularly the decisions that relate to route selection and propagation [148]. Intuitively, selfishness and the lack of coordination could potentially lead to instabilities in the outcome of the protocol, as is actually the case with BGP.

Griffin et al. have studied this problem and the authors provided the most widely accepted formulation, the stable paths problem, with sufficient conditions under which the protocol converges to a stable solution, the *no dispute wheel* condition [162]. In addition to the algorithmic side of BGP which deals with convergence and stability, recent work has focused on the economic side, particularly studying the equilibria of a BGP game and trying to align the incentives of the players (check [71, 161] and references therein).

The interdomain routing incentive game of [161], hereby referred to as ROUTING-GAME, aims to study the policies (strategies) under which BGP is welfare maximizing (i.e., it maximizes the social welfare), and incentive-compatible (i.e., no player has an incentive to deviate from telling the truth where the player's action is to declare private information), and to design a distributed mechanism to provide these attractive properties. Formally, in this game we are given a graph $G = (N, L)$ that represents the AS level topology (nodes N are the ASes and L the set of links between them). The route computation problem is studied for a single destination d and may be directly extended to all destinations assuming route computation is performed independently per destination. Hence, there exists a set of n players indexed by i , and the destination d . Each player has a valuation function $v_i : P^i \rightarrow \mathbb{R}$ which assigns a real number to every permitted route to d , P^i being the set of all permitted routes from i to d . Note that a route is permitted if it is not dropped by i and its neighbors. No two paths are assumed to have the same valuation. Social welfare of a particular outcome, an allocation of routes $R_i, \forall i$ that forms a tree T_d , is defined to be $W_{T_d} = \sum_{i=1}^n v_i(R_i)$. Clearly, the concept of internal policy is captured with the strict valuation or preference function v_i over the different routes to d which is private information given that the nodes are autonomous. In this sense, and as mentioned earlier, the goal of this problem is to design a mechanism that can maximize the social welfare despite the fact that its components, the v_i functions, are unknown or private. The mechanism design framework and particularly

the Vickery-Clark-Groves (VCG) mechanism provides the solution [65]. To do so, a central bank is assumed to exist whose sole task is to allocate a payment $p_i(T_d)$ to each node i based on the outcome. More clearly, a player may either truthfully reveal her valuation to the mechanism (by always picking the best valued routes to d) or not, hoping to manipulate the outcome to her advantage. Based on the players' actions and hence on the outcome tree T_d , a payment $p_i(T_d)$ will be made by the central bank to each player. The utility of each player from an outcome will then be $u_i(T_d) = v_i(R_i) + p_i(T_d)$. The VCG payment scheme is intentionally designed to make the truthful action a *dominant strategy* for all players, hence no player has an incentive to lie about her valuation. To achieve this, AS i will be compensated an amount p_i proportional to the decrease in the value of all upstream ASes that have picked their best route to d through i when the latter does not participate. This is exactly the impact on the social welfare when i is not playing [65]. From a game standpoint, the solution concept that was targeted is the dominant strategy solution - playing truthfully is a dominant strategy and achieving such an attractive solution comes at the expense of assuming a central bank that regulates payments. The authors show that BGP augmented with a VCG payment scheme is incentive-compatible and welfare maximizing in several well studied settings (assumptions on policies or valuation functions).

In the above problem, and generally in problems involving mechanism design, the common scenario is an allocation mechanism that distributes some resource to a set of participating players. In order for a mechanism to implement the Social Choice Function (SCF), for example maximizing the social welfare of all players, the mechanism needs to know the real private information (such as true valuations for example) of the players. This is the case because players might be able to strategically manipulate the output of the mechanism by lying about their private information or strategies. Hence, the mechanism tries to make "truthfully declaring the private information" a *dominant strategy* for the players.

8.2.2 Discovery versus Search: Why receiver-based discovery?

Before discussing the economic issues that arise in discovery mechanisms, we review two main operations in discovery (discussed earlier in section 6.3) and we discuss the source of value of each operation. This section is important to frame our work.

We introduce the notions of *advertisers* and *seekers*. In identifier-based discovery, advertisers are the entities that wish to be discoverable by the rest of the network using their identifiers. They utilize the *join(i , $level$)* interface to express their wish to the mechanism. Seekers, who could be advertisers as well, wish to locate the advertisers and they utilize the *discover(i , j)* operation to do so. In our model players are advertisers who may simultaneously be seekers (think of a node in a Distributed Hash Table (DHT) for example as in [100]).

It is important to distinguish two different classes of problems that relate to discovery and that have been considered in the literature. The first, distributed information retrieval, is that of locating information without prior knowledge of the providers or the location of the information (information could be located anywhere in the network). This problem is generally referred to as unstructured search (as in Gnutella, Freenet P2P networks, social networks, etc.). One key idea here is that in order for the requester to find the requested information, she must search for it and be willing to invest in the search. The provider either can not or is not willing to do so. Prominent work in this vein that addresses cost and incentive structures includes the work by Kleinberg [78].

The second class of problems, which we are more interested in and which we refer to as identifier-based discovery, aim at discovering a path to a uniquely identified entity assuming the seeker is given the identifier(s) of the destination beforehand. This problem is common in service-centric networks where there generally exists

many competing providers for the same service. Within this class of problems, we distinguish two subclasses based on the cost model employed. The first subclass deals with routing problems and focuses on the transit or forwarding cost which is to be bore by the seeker. Several proposals fall under this subclass and many utilize economic tools based in mechanism design [64, 68, 69]. We distinguish another flavor of the problem by noticing that in service-centric network environments, the seeker gets no utility from the discovery part but rather gets the utility from consuming the service itself. In this sense, the utility of discovery is mainly to the provider or the advertiser: the provider wishes to sell the service and can efficiently do so only when the service is “discoverable”. This is the main point that distinguishes our work from the literature on routing and forwarding incentives. The players may be thought of as providers that receive a utility from being discoverable by the rest of the network, the utility of being famous, the latter being inevitably related to the player’s business. Hence, in the receiver-based business model, the player does not care about whether other players are discoverable or not, whereas with general P2P resource sharing applications the player’s utility is to share the resources of other players and hence to be able to discover the rest of the network (originator-based).

8.3 A Taxonomy of Discovery Schemes

Fig. 8.3 shows some classic models used by current discovery schemes (and proposals). Big circles (light and dark) represent nodes used by the routing function (nodes V). Big dark circles represent a subset of those nodes that maintains state about the identifier space. We refer to these nodes as the service nodes denoted with V_D where $V_D \subseteq V$. Small dark circles (colored red) are the entities that wish to be discovered. We refer to those entities as the players denoted with P . Fig. 8.3 tries to illuminate the relationship between the players P (who receive the discovery service),

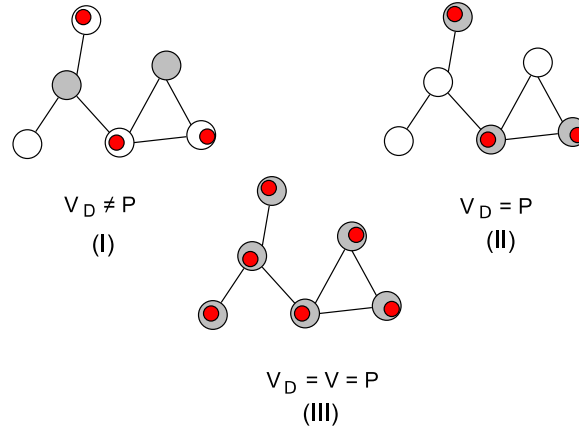


Figure 8.3: Representation of some common models for discovery.

Model	Representative Schemes
model I	DNS, DONA [34], eFIT [144], LIS ([36], etc.)
model II	DHTs (Chord [100], etc.)
model III	NICR ([38, 39], etc.), BGP-DA, ROFL [37]

Table 8.1: Identifier-based discovery schemes.

and the nodes V_D (who provide the service and incur the cost). This relationship is important in an economic setting, such as when studying pricing schemes and when devising a strategic model (and solution concept) for the problem at hand. For example, service nodes in model I (described shortly) may be generally considered to be obedient (i.e., to follow the protocol) as they belong to the same administrative entity (or to multiple competing entities each providing the same service). In models II and III however one needs to consider strategic service nodes in addition to the strategic agents where the two sets could be the same. Some of the representative schemes in the literature that follow these service models are listed in Table 8.1.

In model (I) [$V_D \neq P$], there is a dedicated set of nodes V_D (possibly infrastructure) that keep the state information about the identifier space while the players P

reside on different nodes. DNS is one example of a centralized scheme that follows this model. In DNS, V_D is the set of root/gTLD (for global Top Level Domain) servers and the players are domain servers that keep zone files. Another scheme that uses this model and that is distributed is the recent DONA proposal [34] where V_D is the set of resolution handlers, and the players are generally objects on edge nodes. Another set of proposals that fits under this model is embodied by the Locator-ID-Split (LIS) work which aims at providing discoverability to edge sites (e.g., [36]) or nodes (e.g., [28]) in the Internet.

In model (II) [$V_D = P$], the state is kept on the same set of nodes that the players reside on. In such a model, the players themselves have a common interest in implementing the discovery scheme. The typical example here is Distributed Hash Tables (DHT).

In model (III) [$V_D = V = P$], the state is maintained on all the nodes V and the players are all the nodes. This model is common to proposals that perform routing on identifiers. One class of schemes that fits under this model is represented by the Name Independent Compact Routing (NICR) [38]. In NICR, the forwarding function is aware of the the node identifiers. BGP-DA is the another representative scheme here where the players or nodes are the ASes advertising the prefixes and where it is necessary for all nodes V to keep the state in order for prefix path discovery (i.e., routing in this case) to succeed. Another recent scheme is the DHT-based ROFL [37], in which the routers are the nodes (if we ignore data objects here) that are themselves the players identified by flat identifiers (hashes). Note that models (II) and (III) are the same for our purposes and we shall not make the distinction between the two hereafter.

8.4 Incentives and Pricing

Having introduced the discovery problem and overviewed different discovery models used in the literature, we now proceed to motivate the need for incentives in discovery. Recall that in order for a node to be discoverable, a cost must be incurred by the set of service nodes V_D generally for maintaining *state* about the node's identifier. The term *state* in this context refers to the information stored on the service nodes to allow the players to be discoverable. The per-node state may be thought of as simply the node's routing table which is generally comprised of mappings from identifier to location information. The question that arises then is who pays for maintaining the state, and what incentive models are suitable for the different discovery models. In this section, we present our initial thoughts on incentive models that are applicable to each of the discovery models, and we set the stage for the BGP incentive model which will be discussed in the next chapter.

Model I: $V_D \neq P$

Recall that in this setting, the players P are requesting a discovery service from a set of infrastructure service nodes V_D . When $V_D \neq P$, mechanism design and particularly Distributed Algorithmic Mechanism Design (DAMD) [67] in addition to general cost-sharing models [65] seem to be intuitive frameworks for modeling incentives and pricing. Different situations may arise based on whether the service nodes are obedient or not (obedient service nodes will not try to manipulate the protocol), belong to multiple competing economic entities or not, and on whether the mechanism is subsidized or not. Note that when the mechanism is subsidized, the designer of the mechanism does not have to worry about budget-balance where the latter means that the total payments made by the players must offset the total cost of providing the service.

Assume the service nodes to be obedient and no competition dynamics present, and consider the following DAMD model: each player has a valuation of being discoverable, which she presents to the mechanism. The mechanism logically controls (and operates on) the service nodes collecting all the players' valuations, the demand, and allocating payments back to the players to achieve a mix of goals. These goals could potentially include incentive-compatibility (or strategy-proofness), welfare maximization (or efficiency), and/or budget-balance. When the mechanism is subsidized, the goal of the mechanism is to maximize the social welfare (instead of budget-balance) under the constraint that a cost is associated with providing the service. In this sense, valuations of the service need to be declared truthfully by the players, and hence the goal of incentive-compatibility (especially when the mechanism is able to provide different levels of the service). We sketch such a DAMD model that accounts for service differentiation in Appendix A. A one-shot VCG variant [65] is a natural solution here that could achieve efficiency and incentive-compatibility again assuming that the mechanism could be subsidized in other ways. The VCG pricing scheme is a cost-sharing scheme, i.e., it shares the total cost of providing the service among the participating players. The mechanism will always maximize the social welfare of all the players and will pick prices (cost shares) such that a player i pays an amount equal to the difference in the total welfare of the other players with and without player i 's participation - the damage caused by player i 's participation ². Note that the budget-balance requirement becomes essential when the subsidization assumption does not hold since the total cost must be collected so that service nodes are paid for participating. For example, if a node j is not compensated for the cost of keeping state about the rest of the network, the node will have no incentive to participate. It has been proven by Laffont and Green and later by Satterwaite impossibility theorems [65] that cost-sharing mechanisms can be either strategy-proof and efficient, or strategy-proof and budget-balanced but not both.

²This VCG pricing scheme is referred to as the Clark Pivot rule [65].

When competition among the service providers is present, then the one-shot mechanism design framework seems less practical. This case is more representative of model (I) than the no-competition case. The main idea here is that multiple competing Discovery Service Providers (DSPs) offer the service to the players. Each DSP is assumed to be owned and operated by an autonomous economic entity and DSPs compete for service or market share. Dynamic pricing is more suitable in such a model and a realistic strategic model for this setting based on repeated games was introduced by Afergan [163]. The model discusses price strategies at Internet interchange locations, such as multiple ISPs providing service to a customer (e.g., a CDN). The same model may potentially apply to the discovery mechanism pricing where multiple competing DSPs compete for market share.

Models II, III: $V_D = P$

When the set of service nodes $V_D = P$, players incur a cost due to participation of other players and the issue of incentive and pricing becomes even more challenging. In this distributed setting, the traditional game theoretic and economic tools seem to be more applicable, since the centralized designer and the obedient service nodes assumptions inherent to the mechanism design framework no longer hold. Consider BGP for example where every node that wishes to be discoverable introduces state about its identifier on every other node in the DFZ. NICR [38, 39] schemes on the other hand are less costly as they try to optimize the tradeoff between state and *stretch* (check section 6.4.2 for more on stretch and space/state tradeoffs in the context of compact routing). In this sense, a node that wishes to be discoverable must introduce state on a subset of other nodes in the network. In both examples above, one can directly recognize the incentive mismatch issue and the challenges inherent to the design of incentive and pricing models that are suitable for this setting. In the next chapter, we present one such incentive model for BGP-DA.

8.5 Conclusion

This chapter has motivated the need for considering strategic interactions in network design. We presented three games that highlight the most common solution concepts employed when analyzing strategic interactions among self-interested, welfare-maximizing agents. The solution concepts reviewed are pure-strategy Nash equilibrium (QUERY-GAME), dominant strategy equilibrium (ROUTING-GAME), and subgame-perfect equilibrium (TRADE-GAME). The games are very relevant to the discussion of the next chapter (chapter 9) where we present an incentive model for route distribution in the context of BGP. To illustrate the incentive issues that arise in discovery mechanism design, we presented a taxonomy of discovery schemes based on their business models. We highlighted two main models: one in which the set of service nodes is different than the set of players (model I) and another in which the two sets of nodes are the same (models II and III). We discussed our thoughts on suitable economic models for each of the two discovery models. We would like to note that an initial version of this chapter appeared in [164]. The next chapter elaborates on the incentive issues that arise in the BGP scheme (BGP follows models II,III). We present an incentive model that allows for route distribution (and hence discoverability) while aligning the incentives of all participating agents.

Chapter 9

Route Distribution Incentives in BGP

9.1 Introduction

The Border Gateway Protocol (BGP) is a policy-based path vector protocol and is the de-facto protocol for Internet interdomain routing. The protocol's specification [42] was initially intended to empower domains with control over *route selection*, and *route propagation*. The commercialization of the Internet transformed Autonomous Systems (AS) into economic entities that act selfishly when implementing their internal policies and particularly the decisions that relate to route selection and propagation [148]. BGP is intrinsically about distributing route information to destinations (which are IP prefixes) to establish paths in the network. Path discovery, or simply discovery hereafter, starting with some destination prefix is the outcome of route distribution and route computation.

As discussed earlier in chapter 8, accounting for and sharing the cost of discovery is an interesting problem whose absence from current path discovery schemes has

led to critical economic and scalability concerns. As an example, the BGP control plane functionality is oblivious to cost. More clearly, a node (BGP speaker) that advertises a provider-independent prefix (identifier) does not pay for the cost of being discoverable. Such a cost, which may be large given that the prefix is maintained at every node in the Default Free Zone (DFZ), is paid by the rest of the network. For example, Herrin [165] has preliminarily analyzed the non-trivial cost of maintaining a BGP route. Such incentive mismatch in the current BGP workings is further exacerbated by provider-independent addressing, multi-homing, and traffic engineering practices [12]. Given the fact that the number of BGP prefixes in the global routing table (or RIB) is constantly increasing at a rate of roughly 100,000 entries every 2 years and is expected to reach a total of 388,000 entries in 2011 [145], has motivated us to devise a model that accounts for distribution incentives in BGP.

A large body of work has focused on choosing the right incentives given that ASes are self-interested, utility-maximizing agents. While exploring incentives, most previous work has ignored the control plane incentives (route advertisement/distribution) and has instead focused on the forwarding plane incentives (e.g., transit costs).¹ One possible explanation for this situation is based on the following assumption: a node has an incentive to distribute routes to destinations since the node will get paid for transiting traffic to these destinations, and hence route distribution is ignored as it becomes an artifact of the transit process. We argue that this assumption is not economically viable by considering the arrival of a new customer (BGP speaker). While the servicing edge provider makes money from transiting the new customer's traffic to the customer, the middle providers do not necessarily make money while still incurring the cost to maintain and distribute the customer's route information. In this work, we separate the control plane incentives (incentives to distribute route information) from the forwarding plane incentives (incentives to forward packets)

¹In this chapter, we use the term "control plan" to refer only to route prefix advertisements (not route updates) as we assume that the network structure is static.

and use game theory to model a BGP distribution game. The main problem we are interested in is how to allow BGP prefix information to be distributed globally while aligning the incentives of all the participating agents?

9.1.1 A Simple Distribution Model

We synthesize many of the ideas and results from [70, 78, 161, 71] into a coherent model for studying BGP route distribution incentives. Influenced by the social network query propagation model of Kleinberg and Raghavan [78], we use a completely distributed model in the sense that it does not assume a central bank (in contrast to previous work on truthful mechanisms [65]).

A destination d advertises its prefix and wishes to invest some initial amount of money r_d in order to be globally discoverable (or so that the information about d becomes globally distributed). Since d may distribute its information to its direct neighbors only, d needs to provide incentives to get the information to propagate deeper into the network. Therefore, d must incentivize its neighbors to be distributors of its route, who then incentivize their neighbors to be distributors, and so on. A distributor node will be rewarded based on the role it plays in the outcome routing tree to d , T_d . The utility of the transit node i from distributing d 's route, as we shall describe shortly, increases with the number of nodes that route to d through i - hence the incentive to distribute. While we take BGP as the motivating application, we are interested in the general setting of distributing a good to a set of agents. Agents are located on a network and trade may only occur between directly connected agents. Prices are chosen strategically and the agents are rewarded by volume of sales.

The model seems to correctly capture many of the details behind how policy-based BGP (and in general path-vector protocols) works and the inherent incentives required. Additionally, the model is consistent with the simple path vector formu-

lation introduced by Griffin in [70]. More clearly, it is widely accepted that each AS participating in BGP has as part of its decision space, the following decisions to make:

- import policy: a decision on which routes to d to consider,
- route selection: a decision on what route to d to pick among the multiple possible routes,
- export policy: a decision on who to forward the advertisement to among its direct neighbors.

All three policies are captured in the game model we describe next.

There are two main properties of interest in when it comes to the BGP game model: *convergence*, and *incentives*. The BGP inter-domain routing protocol handles complex interactions between autonomous, competing economic entities that can express local preferences over the different routes. Given the asynchronous interactions among the ASs and the partial information, convergence of BGP to a stable solution becomes an essential property to aim for when studying policies. Griffin et al. [70] defined the stable paths problem which is widely accepted as the general problem that BGP is solving. The authors formulated a general sufficient condition under which the protocol converges to an equilibrium state, mainly the “no dispute wheels” condition. A game-theoretic model was recently developed by Levin et al. [71] builds on the stable paths formalization and studies the incentive-compatibility question. In addition to convergence, incentive issues are crucial to the success and stability of BGP mainly since nodes are assumed to be selfish entities that will act strategically to optimize their utility. In this sense, any distribution and route computation mechanism or policy may only benefit from aligning the incentives of the players to achieve the mechanism’s goals [65, 161, 166, 71].

9.1.2 Our Results

In section 9.2, we present the general distribution game. In this game, a player's pure strategy involves deciding on a "best" route to d as well as determining the reward to offer to her direct neighbors. We define the player's utility as a function of the volume of downstream players that she can recruit based on the assumption that the advertiser or destination, d , receives a fixed marginal utility from each player that maintains a route to d . In this chapter, we are mainly interested in studying the existence of equilibria in the general game. Our main results include:

- First, in section 9.3 we prove that in order to maximize her utility, a player will always choose the route with the highest promised reward. We refer to the resulting policy as the Highest Rewards Path (HRP). We show that under HRP, the BGP protocol always converges to a stable routing tree for any strategy profile by employing the sufficient condition for convergence defined by Griffin et. al [70]. The convergence result allows us to focus on the existence of equilibria as it directly means that any equilibrium strategy profile converges under the BGP protocol to a stable tree.
- Due to the complexity of the strategic dependencies and the competition dynamics, section 9.4 presents the initial equilibrium results on the simplest possible class of graphs with and without competition. Particularly, we present existence results for: 1) the line (and the tree) graphs which involve no competition, and 2) the ring graph which involves competition. We show that a subgame perfect equilibrium always exists for the game induced on the line graph and on the tree, while no such equilibrium exists for the game induced on the ring graph (with an even number of nodes) due to oscillation of *best-response* dynamics under competition when the incentive r_d is "large". To the best of our knowledge, this is the first result to consider competition which

has not been studied in similar previous work [78, 77]. While the full game does not have a subgame perfect equilibrium, we show that there always exists a Nash equilibrium for a special class of subgames. This requires us to first quantify the growth of rewards in the game, or in other words the minimum incentive r_d such that there exists an equilibrium outcome which is a spanning tree (i.e., d is globally discoverable).

- Finally, section 9.4.5 extends the static version of the game to a repeated version. We show how a Nash equilibrium may be constructed in a finitely repeated version of the game by adding a *convergence* constraint on utility which essentially dis-incentivizes oscillation.

9.1.3 Related work

The Simple Path Vector Protocol (SPVP) formalism [70] develops sufficient conditions for the outcome of a path vector protocol to be stable. The two main components of the formalism are *permitted paths* and local strict *preference* relations over alternate paths to some destination. A respective game-theoretic model was developed by Levin [71] that captures these conditions in addition to incentives in a game theoretic setting. Feigenbaum et. al study incentive issues in BGP by considering least cost path (LCP) policies [166] and more general policies [161]. We have elaborated on the ROUTING-GAME of [161] earlier in section 8.2.1. Our model is fundamentally different from [166] (and other works based in mechanism design [69]) in that the prices are strategic, the incentive structure is different, and we do not assume the existence of a central “designer” (or bank) that allocates payments to the players but is rather completely distributed as in real markets. The bank assumption is limiting, and an important question posed in [161] is whether the bank can be eliminated and replaced by direct payments by the nodes. A desirable property of our model is that payments are bilateral and may only flow between neighbors

where a player i should not be able to send a payment to another player j unless the latter is a direct neighbor. This renders the model more robust to manipulation.

Li et. al [77] study an incentive model for query relaying in peer-to-peer (p2p) networks based on rewards, upon which Kleinberg et. al [78] build to model a more general class of trees. As discussed in the QUERY-GAME earlier in section 8.2.1, Kleinberg and Raghavan [78] allude to a similar version of our distribution game in the context of query incentive networks. They pose the general question of whether an equilibrium exists for general Directed Acyclic Graphs (DAGs) in the query propagation game. Both of these probabilistic models do not account for competition. While we borrow the basic idea, we address a different problem which is that of route distribution versus information seeking.

Finally, our work relates to price determination in network markets with intermediaries (refer to the work by Blume et al. [160] and the references therein). We have discussed the TRADE-GAME of [160] earlier in section 8.2.1. A main differentiator of this class of work from other work on market pricing is its consideration of intermediaries and the emergence of prices as a result of strategic behavior rather than competitive analysis or truthful mechanisms. Our work specifically involves the cascading of traders (or distributors) on complex network structures.

9.2 The General Game

Reusing notation from [71, 161], we consider a graph $G = (V, E)$ where V is a set of n nodes (alternatively termed players, or agents) each identified by a unique index $i = \{1, \dots, n\}$, and a destination d , and E is the set of edges or links. Without loss of generality, we study the BGP discovery/route distribution problem for some fixed destination AS with prefix d (as in [70, 71, 161]). The model is extendable to all possible destinations (BGP speakers) by noticing that route distribution and

Chapter 9. Route Distribution Incentives in BGP

computation are performed independently per prefix. The destination d is referred to as the *advertiser* and the set of players in the network are termed *seekers*. Seekers may be distributors who participate in distributing d 's route information to other seeker nodes or consumers who simply consume the route (leaf nodes in the outcome distribution tree). For each seeker node j , Let $P(j)$ be the set of all routes to d that are known to j through advertisements, $P(j) \subseteq \mathcal{P}(j)$, the latter being the set of all simple routes from j . The empty route $\phi \in \mathcal{P}(j)$. Denote by $R_j \in P(j)$ a simple route from j to the destination d with $R_j = \phi$ when no route exists at j , and let $(k, j)R_j$ be the route formed by concatenating link (k, j) with R_j , where $(k, j) \in E$. Denote by $B(i)$ the set of direct neighbors of node i and let $next(R_i)$ be the next hop node on the route R_i from i to d . Define node j to be an *upstream* node relative to node i when $j \in R_i$. The opposite holds for a *downstream* node. Finally, we use $r_{next(R_i)}$ to refer to the reward that the upstream parent from i on R_i offers to i . For example in Fig. 9.1, $next(R_5) = 3$ and 3 is an upstream node relative to 5.

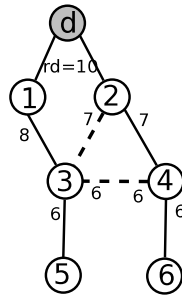


Figure 9.1: Sample network (Not at equilibrium): Solid lines indicate an outcome tree T_d under the advertised rewards.

The general distribution game is as follows: destination d first exports its prefix (identifier) information to its neighbors promising them a reward $r_d \in \mathbb{Z}^+$ ($r_d = 10$ in Figure 9.1) which directly depends on d 's utility of being discoverable. A node j (a player) in turn receives offers from its neighbors where each neighbor i 's offer takes

the form of a reward r_{ij} . A reward r_{ij} that a node i offers to some direct neighbor $j \in B(i)$ is a contract stating that i will pay j an amount that is a function of r_{ij} and of the set of downstream nodes k that decide to route to d through j (i.e., $j \in R_k$ and $R_j = (j, i)R_i$). Note that such a reward model requires that the downstream nodes k notify j of their best route so that the latter can claim its reward from its upstream parent. After receiving the offers, player j strategizes by selecting a route among the possibly multiple advertised routes to d , say $(j, i)R_i$, and deciding on a reward $r_{jl} < r_{ij}$ to send to each *candidate* neighbor $l \in B(j)$ that it has not received a competing offer from. Note then that $r_{lj} < r_{jl}$ where $r_{lj} = 0$ means that j did not receive an offer from neighbor l . Node j then pockets the difference $r_{ij} - r_{jl}$. The process repeats up to some depth that is directly dependent on the initial investment r_d as well as on the strategies of the players. We intentionally keep this reward model abstract at this point, but will revisit it later in the discussion when we define more specific utility functions. For example, in Fig. 9.1, node d promises its neighbor set $\{1, 2\}$ a reward $r_d = 10$. Node 1 exports route $(1, d)$ to its neighbor promising a reward $r_{13} = 8$. Similarly node 2 exports the route $(2, d)$ to its neighbor set $\{3, 4\}$ with $r_{23} = r_{24} = 7$ and so on. Clearly in this model, we assume that a player can strategize per neighbor, presenting different rewards to different neighbors. This assumption is based on the autonomous nature of the nodes and the current practice in BGP where policies may differ significantly across neighbors (as with the widely accepted Gao-Rexford policies [149] for example).

Assumptions To keep our model tractable, we take several simplifying assumptions. In particular, we assume that:

1. the graph is at steady state for the duration of the game, i.e., we do not consider topology dynamics;
2. the advertiser d does not differentiate among the different players (ASes) in

the network, i.e., the ASes are indistinguishable to d .

3. the advertised rewards are integers and are strictly decreasing with depth, i.e., $r_{ij} \in \mathbb{Z}^+$ and $r_{ij} < r_{next(R_i)}, \forall i, j$. We let 1 unit be the cost of distribution (a similar assumption was taken in [78] to avoid the degenerate case of never running out of rewards, referred to as “Zeno’s Paradox”);
4. a node that does not participate will have a utility of zero;
5. finally, our choice of the utility function isolates a class of policies which we refer to as the Highest Reward Path (HRP). As the name suggests, HRP policies incentivize players to choose the path that promises the highest reward. Such class of policies may be defined more generally to account for more complex cost structures as part of the decision space ². We assume for the scope of this work that transit costs are extraneous to the model. This is a restrictive assumption given that BGP allows for arbitrary and complex policies that are generally modeled with a valuation or preference function over the different routes to d (see [70, 161]).

Strategy Space: Given a set of advertised routes $P(i)$ where each route $R_i \in P(i)$ is associated with a promised reward $r_{next(R_i)} \in \mathbb{Z}^+$, a *pure strategy* $s_i \in S_i$ of an autonomous node i comprises two decisions:

- After receiving offers from neighboring nodes, pick a single “best” route $R_i \in P(i)$ (where “best” is defined shortly in Theorem 2);
- Pick a reward vector $r_i = [r_{ij}]_j$ promising a reward r_{ij} to each candidate neighbor j (and export route and reward to respective candidate neighbors).

²Metric based policies could be modeled with HRP by fixing one of the players’ decisions. For example, fixing $r_{ij} = r_{next(R_i)} - 1, \forall i, j$ results in hop count metric; or alternatively setting $r_{ij} = r_{next(R_i)} - c_i$, where c_i is some local cost to the node results in Least Cost Path (LCP) policy [161], etc.

Chapter 9. Route Distribution Incentives in BGP

A strategy profile $\mathbf{s} = (s_1, \dots, s_n)$ and a reward r_d define an outcome of the game³. Every outcome determines a set of paths to destination d given by $O_d = (R_1, \dots, R_n)$. A utility function $u_i(\mathbf{s})$ for player i associates every outcome with a real value in \mathbb{R} . We use the notation s_{-i} to refer to the strategy profile of all players excluding i . The Nash equilibrium is defined as follows:

Definition 2. *A Nash Equilibrium (NE) is a strategy profile $\mathbf{s}^* = (s_1^*, \dots, s_n^*)$ such that no player can move profitably by changing her strategy, i.e., for each player i , $u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*), \forall s_i \in S_i$.*

Cost: The cost of participation is local to the node and includes for example the cost associated with the effort that a node spends in maintaining the route information⁴. Other cost factors that depend on the volume of traffic (proportional to the number of downstream nodes in the outcome O_d) are more relevant to the forwarding plane and as mentioned earlier in the assumptions, we ignore this cost in the current model. Hence, we simply assume that every player i incurs a cost c_i which is the cost of participating. We assume for the scope of this chapter that the local cost is constant with $c_i = c = 1$.

Utility: We experiment with a simple class of utility functions which rewards a node linearly based on the number of sales that the node makes. This model incentivizes distribution and potentially requires a large initial investment from d . More clearly, define $N_i(\mathbf{s}) = \{j \in V \setminus \{i\} | i \in R_j\}$ to be the set of nodes that pick their best

³We abuse notation hereafter and we refer to the outcome with simply the strategy profile \mathbf{s} where it should be clear from context that an outcome is defined by the tuple $\langle \mathbf{s}, r_d \rangle$. Notice that a strategy profile may be associated with an outcome if we model r_d as an action. We refrain from doing so to make it explicit that r_d is not strategic.

⁴A preliminary estimate of this cost is shown by Herrin [165] to be \$0.04 per route/router/year for a total cost of at least \$6,200 per year for each advertised route assuming there are around 150,000 DFZ routers that need to be updated.

Chapter 9. Route Distribution Incentives in BGP

route to d going through i (nodes downstream of i) and let $\delta_i(\mathbf{s}) = |N_i(\mathbf{s})|$. Let the utility of a node i from an outcome or strategy profile \mathbf{s} be:

$$u_i(\mathbf{s}) = (r_{next(R_i)} - c_i) + \sum_{\{j|i=next(R_j)\}} (r_{next(R_i)} - r_{ij})(\delta_j(\mathbf{s}) + 1) \quad (9.1)$$

The first term $(r_{next(R_i)} - c_i)$ of (9.1) is incurred by every participating node and is the one unit of reward from the upstream parent on the chosen best path minus the local cost. Based on the fixed cost assumption, we often drop this first term when comparing player payoffs from different strategies since the term is always positive when $c = 1$. The second term of (9.1) (the summation) is incurred only by distributors and is the total profit made by i where $(r_{next(R_i)} - r_{ij})(\delta_j(\mathbf{s}) + 1)$ is i 's profit from the sale to neighbor j (which depends on δ_j). We assume here that node i gets no utility from an oscillating route and gets positive utility when R_i is stable. A rational selfish node will always try to maximize its utility by picking $s_i = (R_i, [r_{ij}]_j)$. There is an inherent tradeoff between $(r_{next(R_i)} - r_{ij})$ and $(\delta_j(\mathbf{s}))$ s.t. $i = next(R_j)$ when trying to maximize the utility in Equation (9.1) in the face of competition as shall become clear later. A higher promised reward r_{ij} allows the node to compete (and possibly increase δ_j) but will cut the profit margin. Finally, we implicitly assume that the destination node d gets a constant marginal utility of r_d for each distinct player that maintains a route to d - the marginal utility of being discoverable by any seeker - and declares r_d truthfully to its direct neighbors (i.e., r_d is not strategic).

9.3 Convergence under HRP

Before proceeding with the game model, we first prove the following theorem which results in the Highest Reward Path (HRP) policy.

Chapter 9. Route Distribution Incentives in BGP

Theorem 2. *In order to maximize its utility, node i must always pick the route R_i with the highest promised reward, i.e., such that $r_{next(R_i)} \geq r_{next(R_l)}, \forall R_l \in P(i)$.*

Proof. The case for $|B(i)| = 1$ is trivial. The case for $|B(i)| = 2$ is trivial as well since i will not be able to make a sale to the higher reward neighbor by picking the lower reward offer. Assume that node i has more than 2 neighbors and that any two neighbors, say k, l advertise routes $R_k, R_l \in P(i)$ s.t. $k = next(R_k), l = next(R_l)$ and $r_{ki} < r_{li}$, and assume that i 's utility for choosing route R_k over R_l either increases or remains the same, i.e., $u_i^{R_k} \geq u_i^{R_l}$. We will show by contradiction that neither of these two scenarios could happen.

scenario 1: $u_i^{R_k} > u_i^{R_l}$ From Equation (9.1), it must be the case that either (case 1) node i was able to make at least one more sale to some neighbor j who would otherwise not buy, or (case 2) some neighbor j who picks $(j, i)R_i$ can strictly increase her $\delta_j(\mathbf{s})$ when i chooses the lower reward path R_k . For case 1, and assuming that r_{ij} is the same when i chooses either route, it is simple to show that we arrive at a contradiction in the case when $j \in \{k, l\}$ (mainly due to the strictly decreasing reward assumption, i.e., $r_i < r_{next(.)}$); and in the case when $j \notin \{k, l\}$, it must be the case that j 's utility increases with i 's route choice, i.e., $u_j^{(j,i)R_k} > u_j^{(j,i)R_l}$. This contradicts with Equation (9.1) since w.r.t. j , both routes have the same next hop node i . The same analogy holds for case 2.

scenario 2: $u_i^{R_k} = u_i^{R_l}$ Using the same analogy of scenario 1, there must exist at least one neighbor j of i that would buy i 's offer only when the latter picks R_k , or otherwise node i will be able to strictly increase its utility by picking R_l pocketing more profit. \square

The theorem implies that a player could perform her two actions sequentially, by first choosing the highest reward route R_i , then deciding on the reward vector r_{ij} to export to its neighbors. Thus, we shall represent player i 's strategy hereafter simply with the rewards vector $[r_{ij}]$ and it should be clear that player i will always pick the

“best” route to be the route with the highest promised reward. When the rewards are equal however, we assume that a node breaks ties consistently.

The question we attempt to answer here is whether the BGP protocol dynamics converge to a unique outcome tree T_d under some strategy profile \mathbf{s} . A standard model for studying the convergence of BGP protocol dynamics was introduced by Griffin et al. [70], and assumes BGP is an infinite round game in which a *scheduler* entity decides on the *schedule*, i.e., which players participate at each round (models the asynchronous operation of BGP). The authors devised the “no dispute wheels” condition [70], which is the most general condition known to guarantee convergence of possibly “conflicting” BGP policies to a unique stable solution (tree). From Theorem 2, it may be easily shown that “no dispute wheels” exist under HRP policy, i.e., when the nodes choose highest reward path breaking ties consistently. This holds since any dispute wheel violates the assumption of strictly decreasing rewards on the reward structure induced by the wheel. Hence, the BGP outcome converges to a unique tree T_d [70] under any strategy profile \mathbf{s} . This result allows us to focus on the existence of equilibria as it directly means that the BGP protocol dynamics converges to a tree under any equilibrium strategy profile.

9.4 Equilibria

In the general game model defined thus far, the tie-breaking preferences of the players is a defining property of the game, and every outcome (including the equilibrium) depends on the initial reward/utility r_d of the advertiser. Studying the equilibria of the general game for different classes of utility functions and for different underlying graph structures is not an easy problem due to the complexity of the strategic dependencies and the competition dynamics. Hence, we start by studying the game on the simplest possible class of graphs with and without competition. We assume *full*

information as we are interested in studying the existence question initially rather than how the players would arrive at the equilibrium⁵. Particularly, we present existence results for the simplest two classes of graphs: 1) the line (and the tree) graphs which involve no competition, and 2) the ring graph which involves competition.

To study the existence of equilibria on the simple line and ring graphs, we fix the *schedule* of play (i.e., who plays when?) as we formalize shortly. We start by examining a static version of the full-information game in which each player plays once at a single stage based on proximity to d , and we then proceed to examine the repeated version of the static game.

9.4.1 The Static Multi-Stage Game with fixed schedule

In order to apply the correct solution concept, we fix the *schedule* of play. The schedule is based on the inherent order of play in the model: recall that the advertiser d starts by advertising itself and promising a reward r_d ; the game starts at stage 1 where the direct neighbors of d , i.e., the nodes at distance 1 from d , observe r_d and play simultaneously by picking their rewards while the rest of the nodes “do-nothing”. At stage 2, nodes at distance 2 from d observe the stage 1 strategies and then play simultaneously and so on. Stages in this *multi-stage game with observed actions* [63] have no temporal semantics. Rather, they identify the network positions which have strategic significance. The closer a node is to the advertiser, the more power such a node has due to the strictly decreasing rewards assumption. The key concept here is that it is the *information sets* [63] that matter rather than the time of play, i.e., since all the nodes at distance 1 from d observe r_d before playing, all these nodes belong to the same information set whether they play at the same time or at different time instants. We refer to a single play of the multi-stage game as the *static* game. We

⁵This of course is an interesting question in its own right.

resort to the multi-stage model (the fixed schedule) on our simple graphs to eliminate the synchronization problems inherent in the BGP protocol and to focus instead on the existence of equilibria. By restricting the analysis to the fixed schedule, we do not miss any equilibria. This is due to the fact that the fixed schedule is only meant to replace the notion of “fair and infinite schedule” [70] with a more concrete order of play. The resulting game always converges in a single play for any strategy profile, and the outcome tree is necessarily one of shortest-paths (in terms of number of hops) ⁶. The main limitation of this model however is that it can not deal with variable costs c_i for which the outcome (HRP tree) might not be a shortest-path tree.

Formally, and using notation from [63], each player i plays only once at stage $k > 0$ where k is the distance from i to d in number of hops. At every other stage, the player plays the “do nothing” action. The set of player actions at stage k is the stage- k action profile, denoted by $a^k = (a_1^k, \dots, a_n^k)$. Further, denote by $h^{k+1} = (r_d, a^1, \dots, a^k)$, the *history* at the end of stage k which is simply the initial reward r_d concatenated with the sequence of actions at all previous stages. We let $h^1 = (r_d)$. Finally, $h^{k+1} \in H^{k+1}$ the latter being the set of all possible stage- k histories. When the game has a finite number of stages, say $K + 1$, then a terminal history h^{K+1} is equivalent to an outcome of the game (which is a tree T_d) and the set of all outcomes is H^{K+1} .

The pure-strategy of player i who plays at stage $k > 0$ is a function of the history and is given by $s_i : H^k \rightarrow \mathbb{R}^{m_i}$ where m_i is the number of direct neighbors of player i that are at stage $k + 1$ (implicit here is that a player always picks the highest reward route). Starting with r_d (which is h^1), it is clear how the game produces actions at every later stage based on the player strategies resulting in a terminal

⁶This follows in the multi-stage game since a player at stage k will not offer rewards to its neighbors at stage $l < k$, i.e., rewards flow in one direction away from d . The outcome is necessarily a shortest path tree since every player at stage k must pick its best route from the offers its received from neighbors at stage $k - 1$.

action profile or outcome. Hence, given r_d , an outcome in H^{K+1} may be associated with every strategy profile \mathbf{s} , and so the definition of Nash equilibrium (Definition (2)) remains unchanged. Finally, it is worthwhile noting that the “observed actions” requirement (where a player observes the full history before playing) is not necessary for our results in the static game as we shall see in the construction of the equilibrium strategies. Keeping this requirement in the model allows us to classify the play from some stage onward, contingent on a history being reached as a subgame in its own right as we describe next.

Definition 3. [63] *A proper subgame of a full game is a restriction of the full game to a particular history. The subgame inherits the properties of the full game such as payoffs and strategies while simply restricting those to the history.*

In our game, each stage begins a new subgame which restricts the full game to a particular history. For example, a history h^k begins a subgame $G(h^k)$ such that the histories in the subgame are restricted to $h^{k+1} = (h^k, a^k)$, $h^{k+2} = (h^k, a^k, a^{k+1})$, and so on.

Definition 4. [63] *A strategy profile $\mathbf{s}^* = (s_1^*, \dots, s_n^*)$ is a subgame-perfect equilibrium if it is a Nash equilibrium for every proper subgame of the full game.*

Hereafter, the general notion of equilibrium we use is the Nash equilibrium and we shall make it clear when we generalize to subgame perfect equilibria. We are only interested in pure-strategy equilibria [63] and in studying the existence question as the incentive r_d varies. A Nash equilibrium hereafter is a pure-strategy Nash equilibrium. We now proceed to study the equilibria on special networks.

No Competition: the line graph and the tree

In the same spirit as [78] we inductively construct the equilibrium for the line graph (simply referred to as the line hereafter) of Figure 9.2 given the utility function of

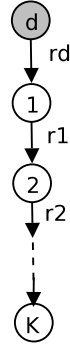


Figure 9.2: Line graph: a node's index is the stage at which the node plays; d advertises at stage 0; $K = n$.

Equation (9.1). We present the result for the line which may be directly extended to trees. Before proceeding with the construction, notice that for the line, $m_i = 1$ for all players except the leaf player since each of those players has a single downstream neighbor. In addition, $\delta_i(\mathbf{s}) = \delta_j(\mathbf{s}) + 1, \forall i, j$ where j is i 's child ($\delta_i = 0$ when i is a leaf). We shall refer to both the player and the stage using the same index since our intention should be clear from the context. For example, the child of player i is $i + 1$ and its parent is $i - 1$ where player i is the player at stage i . Additionally, we simply represent the history $h^{k+1} = (r_k)$ for $k > 0$ where r_k is the reward promised by player k (player k 's action). The strategy of player k is therefore $s_k(h^k) = s_k(r_{k-1})$ which is a singleton (instead of a vector) since $m_i = 1$ (for completeness, let $r_0 = r_d$). This is a *perfect information* game [63] since a single player moves at each stage and has complete information about the actions of all players at previous stages. Hence, backward induction may be used to construct the subgame-perfect equilibrium.

We construct the equilibrium strategy s^* inductively as follows: first, for all players i , let $s_i^*(x) = 0$ when $x \leq c$ (where c is assumed to be 1). Then assume that $s_i^*(x)$ is defined for all $x < r$ and for all i . Obviously, with this information, every player i may compute $\delta_i(x, s_{-i}^*)$ for all $x < r$. This is simply due to the fact that δ_i depends on the downstream players from i who must play an action or reward

Chapter 9. Route Distribution Incentives in BGP

strictly less than r . Finally, for all players i we let $s_i^*(r) = \arg \max_x (r - x) \delta_i(x, s_{-i}^*)$ where $x < r$.

Theorem 3. *The strategy profile s^* is a subgame-perfect equilibrium.*

Sketch of Proof The proof for the line is straightforward and follows from backward induction by constructing the optimal strategies starting with the last player (player K) first, then the next-to-last, and so on up to player 1. The strategies are optimal for every history (by construction) and given the utility function defined in Equation (9.1), no player can move profitably. Notice that in general when $r_{next(R_i)} \leq c$, propagation of the reward will stop simply because at equilibrium no player will want a negative utility and will prefer to not participate instead (the case with the leaf player). \square

The proof may be directly extended to the tree since each player in the tree has a single upstream parent as well and backward induction follows in the same way. On the tree, the strategies of the players that play simultaneously at each stage are also independent.

Competition: The ring

As opposed to the line, we present next a negative result for the ring graph (simply referred to as the ring hereafter). In a ring, each player has a degree of 2 and $m_i = 1$ again for all players except the leaf player. We consider rings with an even number of nodes due to the direct competition dynamics. Figure 9.3 shows the 2-stage, the 3-stage, and general K -stage versions of the game. In the multi-stage game, after observing r_d , players 1 and 2 play simultaneously at stage 1 promising rewards r_1 and r_2 respectively to their downstream children, and so on. We shall refer to the players at stage j using ids $2j - 1$ and $2j$ where the stage of a player i , denoted as $l(i)$, may be computed from the id as $l(i) = \lceil \frac{i}{2} \rceil$. For the rest of the discussion, we

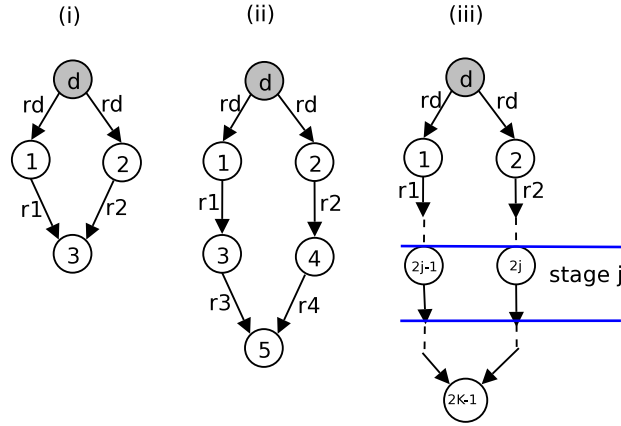


Figure 9.3: Ring network with even number of nodes: (i) 2-stage game, (ii) 3-stage game, and general (iii) K -stage game.

assume WLOG that the player at stage K (with id $2K - 1$) breaks ties by picking the route through the left parent $2K - 3$.

For the 2-stage game in Figure 9.3, it is easy to show that an equilibrium always exists in which $s_1^*(r_d) = s_2^*(r_d) = (r_d - 1)$ when $r_d > 1$ and 0 otherwise. This means that player 3 enjoys the benefits of *perfect competition* due to the Bertrand-style competition [63] between players 1 and 2. The equilibrium in this game is independent of player 3's preference for breaking ties. We now present the following negative result,

Claim 4. *The 3-stage game induced on the ring (of Figure 9.3(ii)) does not have a subgame-perfect equilibrium. Particularly, there exists a class of subgames for $h^1 = r_d > 5$ for which there is no Nash equilibrium.*

Sketch of Proof The proof makes use of a counterexample. Using the backward induction argument, notice first that the best strategy of players 3 and 4 is to play a *Bertrand-style competition* as follows: after observing $a^1 = (r_1, r_2)$, player 3 plays $r_3 = 0$ when $r_1 = 1$, $r_3 = \min(r_1 - 1, r_2 - 1)$ when both $r_1 > 1$ and $r_2 > 1$, and $r_3 = 1$ when $r_1 > 1$ and $r_2 = 1$. Player 4 plays symmetrically. Knowing that, players

Chapter 9. Route Distribution Incentives in BGP

1 and 2 will choose their strategies simultaneously and no equilibria exist for $r_d > 5$ due to oscillation of the best-response dynamics. This may be shown by examining the strategic form game, in normal/matrix form, between players 1 and 2 (in which the utilities are expressed in terms of r_d). We briefly show the subgame for $r_d = 6$ and we leave the elaborate proof as an exercise for the interested reader. Figure 9.4 shows the payoff matrix of players 1 and 2 for playing actions $r_1 \in \{2, 3\}$ (rows) and $r_2 \in \{1, 3\}$ (columns), respectively. The payoff shown is taken to be $u_i = (r_d - r_{ij})\delta_i$ ignoring the first term of Equation (9.1). The actions shown are the only remaining actions after applying iterated strict dominance, i.e., all other possible actions for the players are strictly dominated. Clearly, no pure strategy Nash equilibria exist⁷.

		r2	
		1	3
r1	2	8 , 5	4 , 6
	3	6 , 5	6 , 3

Figure 9.4: The payoff matrix of players 1 and 2 for the 3-stage game on the ring of Figure 9.3(ii) when $r_d = 6$.

The argument could be directly extended to any $r_d > 5$ since player 2 will still have the incentive to oscillate. □

The value $r_d > 5$ signifies the breaking point of equilibrium or the reward at which player 2, when maximizing her utility $(r_d - r_2)\delta_2$, will always oscillate between competing for 5 (by playing large r_2) or not (by playing small r_2). We elaborate on this value later in section 9.4.4. Hence, under the linear utility given in Equation (9.1), an equilibrium does not exist on the simple ring. This negative result for the game induced on the 3-stage ring may be directly extended to the general game

⁷There is however a single mixed strategy equilibrium in which player 1 plays $r_1 = 2$ with probability $\frac{2}{3}$ while player 2 plays $r_2 = 1$ with probability $\frac{1}{2}$, yielding expected payoffs 6 and 5 for players 1 and 2 respectively.

for the K -stage ring by observing that a class of subgames $G(h^{K-2})$ of the general K -stage game are identical to the 3-stage game. While the full game does not have an equilibrium for $K > 2$ stages, we shall show next that there always exists an equilibrium for the special subgame $G(r_d^*)$ (for $h^1 = r_d^*$), where the reward r_d^* is the minimum incentive to guarantee that d 's route is globally distributed at equilibrium. We define and compute r_d^* next before constructing the equilibrium.

9.4.2 Growth of Incentives

We next answer the following question: Find the minimum incentive r_d^* , as a function of the depth of the network K (equivalently the number of stages in the multi-stage game), such that there exists an equilibrium outcome for the subgame $G(r_d^*)$ that is a spanning tree. We seek to compute the function f such that $r_d^* = f(K)$. First, we present a result for the line, before extending it to the ring. On the line, K is simply the number of players, i.e., $K = n$.

Lemma 5. *On the line graph, we have $f(0) = 0$, $f(1) = 1$, $f(2) = 2$, and $\forall k > 2$*

$$f(k) = (k-1)f(k-1) - (k-2)f(k-2) \quad (9.2)$$

Proof. First, $f(0) = 0$, $f(1) = 1$ and $f(2) = 2$ are trivially true given the utility function of Equation (9.1). The proof uses induction on the depth of the network. First, for the base case $k = 3$, in the 3-stage line the Nash equilibrium is for player 1, the player at distance 1 from d , to play $r_1 = 2$ and for player 2 to play $r_2 = 1$ (in every NE, $s_i(1) = 0$, $\forall i$). Given $r_d^* = f(3) = 3$, the utility of player 1 is $u_1 = (3-2)2 \geq (3-r'_2)\delta'_2$, $\forall r'_2 < 3$. Similarly player 2 may not move profitably from playing $r_2 = 1$.

Assume $f(x) = (x-1)f(x-1) - (x-2)f(x-2)$ holds $\forall x < k$. We construct the k -stage game from the $(k-1)$ -stage game by adding a node/player between node d

and node 1 in the $(k - 1)$ -stage game. Notice the player 2 in the k -stage game used to be player 1 in the $(k - 1)$ -stage game. By definition of f , in the k -stage game, when player 1 plays $r_1 = f(k - 1)$ then $\delta_1 = (k - 1)$ and no player $i, 2 \leq i \leq k$ may deviate profitably from playing $r_i = f(k - i)$. Here $r_1 = f(k - 1)$ is the minimum reward to get a $\delta_1 = (k - 1)$. In general, it holds by construction of f that there are k possible outcomes for player 1, corresponding to the values $\delta_1 = 0, 1, \dots, k - 1$. For each of these outcomes, we have an action for player 1, $r_1 = f(x)$, which results in the outcome tree corresponding to $\delta_1 = x, \forall x < k$ and such that no player besides player 1 may deviate profitably contingent on player 1 playing $r_1 = f(x)$ (In this outcome player i plays $f(x - i + 1) \forall 2 \leq i \leq n$). In order for $\delta_1 = k - 1$ to be the equilibrium outcome, it must be the case that $r_1 = f(k - 1)$ maximizes player 1's utility given r_d (and hence no player including player 1 may deviate profitably), i.e., it must be that $\forall 2 \leq j \leq k$

$$(r_d - f(k - 1))(k - 1) \geq (r_d - f(k - j))(k - j)$$

This condition is equivalent to:

$$(r_d - f(k - 1))(k - 1) \geq (r_d - f(k - 2))(k - 2) \quad (9.3)$$

since $(r_d - f(k - 2))(k - 2) \geq (r_d - f(k - j))(k - j), \forall 3 \leq j \leq k$ and for $r_d \geq f(k - 1)$. Equation (9.3) implies that $r_d \geq (k - 1)f(k - 1) - (k - 2)f(k - 2)$. The minimum such incentive is:

$$r_d^* = f(k) = (k - 1)f(k - 1) - (k - 2)f(k - 2) \quad (9.4)$$

which is greater than $f(k - 1)$ concluding the proof. \square

Notice that $f(K)$ grows exponentially with the depth K of the line network ⁸. By subtracting $f(k - 1)$ from both sides of the recurrence relation, it may be shown

⁸On the other hand, on complete d -ary trees, it may be shown that the function $f(k) = \Theta(k) = \Theta(\log_d n)$ for $d \geq 2$ since the number of players, and hence δ_i , grows exponentially

that

$$f(k) - f(k - 1) = (k - 2)! \quad (9.5)$$

9.4.3 A Special Subgame

We now revisit the the K -stage game of Figure 9.3(iii) on the ring and we focus on a specific subgame which is the restriction of the full game to $h_1 = r_d^* = f(K)$, and we denote this subgame by $G(r_d^*)$. Consider the following strategy profile \mathbf{s}^* for the subgame: players at stage j play $s_{2j-1}^*(h^j) = f(K - j)$, and $s_{2j}^*(h^j) = f(K - j - 1)$, $\forall 1 \leq j \leq K - 1$, and let $s_{2K-1}^*(h^K) = 0$.

Theorem 6. *The profile \mathbf{s}^* is a Nash equilibrium for the subgame $G(r_d^*)$ on the K -stage ring, $\forall K > 2$.*

Proof. Notice first that the complete history h^{K+1} which corresponds to r_d^* and \mathbf{s}^* is an outcome that is a spanning tree (each player picks the best route through the upstream parent while the last player $2K - 1$ prefers the left parent who is promising a higher reward). We will show that no player i can deviate from playing s_i^* given s_{-i}^* by considering the players at each stage j , $\forall 2 \leq j \leq K - 1$ first and then we extend the reasoning to the players at stage 1. For the players at stage j we show that player $2j - 1$ may not deviate profitably from playing $s_{2j-1}^*(h^j) = r_{2j-1} = f(K - j)$ given the strategies of the rest of the players (particularly given $s_{2j}^*(h^j) = r_{2j} = f(K - j - 1)$), and the same for player $2j$. Given that $r_{2j} < r_{2j-1}$ (i.e., player $2j$ not competing with player $2j - 1$), then by definition of the function f , there exists an outcome on

with depth K . These growth results on the line graph and the tree seem parallel to the result of Kleinberg and Raghavan [78] (and the elaboration in [167]) which states that the reward required by the root player in order to find an answer to a query with constant probability grows exponentially with the depth of the tree when the branching factor of the tree is $1 < b < 2$, i.e., when each player has an expected number of offsprings $1 < b < 2$, while it grows logarithmically for $b > 2$.

Chapter 9. Route Distribution Incentives in BGP

the ring such that $\delta_{2j-1} = K - j$ when $r_{2j-1} = f(K - j)$ and $r_{2j} < r_{2j-1}$ (this holds at each stage $2 \leq j \leq K - 1$ given the tie-breaking preference of player $2K - 1$). The utility then to player $2j - 1$ of playing $r_{2j-1} = f(K - j)$ is:

$$u_{2j-1} = (f(K - j + 1) - f(K - j))(K - j) \quad (9.6)$$

$$= (f(K - j + 1) - f(K - j - 1))(K - j - 1) \quad (9.7)$$

$$= (K - j)! \quad (9.8)$$

where the second equality holds by definition of function f (Equation (9.4)) and the third equality holds because $(f(K) - f(K - 2))(K - 2) = (f(K) - f(K - 1) + f(K - 1) - f(K - 2))(K - 2) = ((K - 2)! + (K - 3)!)(K - 2) = (K - 1)!$. Given the strategies of the rest of the players, player $2j - 1$ may not deviate profitably, i.e., $u_{2j-1}(f(K - j), s_{-(2j-1)}^*) \geq u_{2j-1}(r', s_{-(2j-1)}^*), \forall r' \neq f(K - j)$. This is simply because playing an $r' > f(K - j)$ will strictly decrease u_{2j-1} since δ_{2j-1} is already maximized ($\delta_{2j-1} = K - j$ in this case), while playing $r' < f(K - j)$ can at best yield player $2j - 1$ the same utility when $r' = f(K - j - 1)$ (Equation (9.7)). The same reasoning holds for player $2j$ who may not deviate profitably by playing $r'' \neq f(K - j - 1)$. Specifically, any $r'' < f(K - j - 1)$ can at best yield player $2j$ the same utility when $r'' = f(K - j - 2)$, and in order to compete with player $2j - 1$ (and possibly increase δ_{2j}) player $2j$ must play $r'' > r_{2j-1} = f(K - j)$ which violates the decreasing rewards assumption. Hence neither player at stage j may deviate profitably for all $2 \leq j \leq K - 1$. It remains to show that players at stage 1 may not deviate profitably. First, player 1 may not deviate profitably using the same argument we used for player $2j - 1$ where $j = 1$. The utility to player 1 is $u_1(f(K - 1), s_{-1}^*) = (K - 1)!$. On the other hand, player 2 gets the same utility as player 1 where $u_2(f(K - 2), s_{-2}^*) = (f(K) - f(K - 2))(K - 2) = (K - 1)!$. In the same way, player 2 may not deviate profitably since playing any $r'_2 \neq f(K - 2)$ may

Chapter 9. Route Distribution Incentives in BGP

not increase u_2 given s_{-2}^* . More clearly, in order for player 2 to compete with player 1 and possibly increase δ_2 from $K - 2$ to $K - 1$, player 2 must play an $r'_2 > f(K - 1)$ which in the best case yields a utility $u_2(r'_2, s_{-2}^*) = (f(K) - r'_2)(K - 1) < (K - 1)!$. Hence, neither player 1 nor player 2 may deviate profitably given the strategies of the other players. Finally, the case for player $2K - 1$ is trivial. This concludes the proof. \square

This result may be interpreted as follows: if the advertiser were to play strategically assuming she has a marginal utility of at least r_d^* and is aiming for a spanning tree (global discoverability), then $r_d^* = f(K)$ will be her Nash strategy in the game induced on the K -stage ring, $\forall K > 2$ (given \mathbf{s}^*).

We have shown in Lemma (5) that the minimum incentive r_d^* on the line (such that there exists an equilibrium spanning tree for the subgame $G(r_d^*)$) as a function of depth K is $r_d^* = f(K)$. We now extend the result to the ring denoting by $f_r(K)$ the growth function for the ring in order to distinguish it from that of the line, $f(K)$.

Corollary 7. *On the ring graph, we have $f_r(k) = f(k)$ as given by Lemma (5).*

Sketch of Proof We have shown in Theorem (6) that \mathbf{s}^* is an equilibrium for the subgame $G(r_d^*)$ for $r_d^* = f(K)$ and that the equilibrium is a spanning tree. What remains to show is that $f(K)$ is the minimum incentive required. This follows by isolating the left branch of the ring, which is a line graph that constitutes of player d and all the players with odd identifiers, and using the same argument of Lemma (5) on this branch: an $r_d < f(K)$ allows player 1 to move profitably by playing an $r_1 < f(K - 1)$ which violates the spanning tree requirement (by definition of f). \square

Next, we present an important result which we utilize to extend the subgame equilibrium of Theorem (6) and later as well for the general equilibrium in the repeated version of the game.

9.4.4 Competition Rewards

Competition on the general K -stage ring starts between players 1 and 2 who compete for the last player $2K - 1$, knowing that the latter picks the highest reward route breaking ties by going through the left parent. This section aims at answering the following question: What does it take for player 2 (and the players in the right branch of the ring) to be able to compete with player 1 (and the players in the left branch of the ring) given that the left branch of the ring is preferred to the right branch under the tie-breaking preference of player $2K - 1$? Formally, for the players $i, 1 \leq i \leq n$, let $V_{odd} = \{i | i \text{ is odd, and } i \notin \{2K - 1\}\}$ be the subset of players i with odd identifiers (players in the left branch of the ring), while $V_{even} = \{i | i \text{ is even}\}$. In an abuse of notation, denote by $\mathbf{s}_{V_{odd}}$ the strategy profile of all the players in V_{odd} and similarly for $\mathbf{s}_{V_{even}}$. We are interested in computing a strategy profile $\hat{\mathbf{s}}_{V_{even}}$ such that $s_i = \min_{r'_i \in \mathbb{Z}^+}(r'_i), \forall s_i \in \hat{\mathbf{s}}_{V_{even}}$ and such that the players in V_{odd} will have no incentive to compete given $\hat{\mathbf{s}}_{V_{even}}$, given that player 1 is playing $r_1 = x$ (r_d is assumed to be arbitrarily large here). Specifically, we are interested in the strategy $s_2 \in \hat{\mathbf{s}}_{V_{even}}$ of player 2 which we refer to as $s_2 = g_K(x)$. In other words, in the subgame $G(h^2)$ for $h^2 = (r_d, r_1 = x, r_2 = g_K(x))$ (for $x \geq f(K - 1)$), the outcome tree is guaranteed to have a $\delta_2 = K - 1$. For example, in the 3-stage ring of Figure 9.3(ii), we have $g_3(x) = x + 1$ or in other words, player 2 must play at least $r_2 = r_1 + 1$ if she is to win over player 5's business and hence compete with player 1. Finding a closed form for $g_K(x)$ is not necessary for the existence results we seek in this chapter. Our goal here is to show that $g_K(x)$ always exists. We show the existence of $g_k(x)$ in the Appendix at the end of this chapter and we show a plot of the function $g_K(x) - x$ in Figure 9.5 which increases exponentially with K . It may be shown that $g_K(x) - x \leq (K - 2)!$ peaking at $x = f(K - 1) + j(K - 2)!, \forall j \geq 0$. Specifically, for $x = f(K - 1)$, then $g_K(x) = f(K)$ according to Equation (9.5).

Having defined $\hat{\mathbf{s}}_{V_{even}}$ and $g_K(x)$, we now proceed to generalize the result of The-

orem (6) as follows:

Theorem 8. *The profile \mathbf{s}^* constructed in Theorem (6) is a Nash equilibrium for the class of subgames $G(h^1)$ for $f(K) \leq h^1 = r_d \leq f(K+1)$ on the K -stage ring, $\forall K > 2$.*

Proof. We need to show that \mathbf{s}^* is a Nash equilibrium for all the histories $f(K) \leq h^1 = r_d \leq f(K+1)$. Theorem (6) presents the proof for the history $h^1 = r_d = f(K)$. To show that \mathbf{s}^* remains an equilibrium for $f(K) < h^1 \leq f(K+1)$, we prove that no player may deviate profitably for all these histories. First, we consider the players in V_{odd} and we note that every player $i \in V_{odd}$ at stage j may not deviate profitably from playing $f(K-j)$ given s_{-i}^* by definition of f and since δ_i is already maximized ($\delta_i = K-j$). As for the players in V_{even} , we start with player 2. We have shown in section 9.4.4 that in order for player 2 to compete with player 1 (who is playing $r_1 = f(K-1)$) and possibly increase δ_2 (from $K-2$ to $K-1$), player 2 must play $r_2 = g_K(f(K-1)) = f(K)$. Any action $r'_2 < r_2$ will not provide enough incentive for the players downstream of 2 to compete and possibly increase δ_2 . Hence, the question is to find the value of r_d such that player 2 may not deviate profitably from playing $f(K-2)$ (i.e., not competing) to playing $f(K)$ (i.e., competing). This requirement follows due to the fact that if player 2 has an incentive to compete, then the outcome of the game will oscillate between competing or not (i.e., no equilibrium) as we have demonstrated on the simple 3-stage game in Claim (4). The requirement may be

Chapter 9. Route Distribution Incentives in BGP

stated as:

$$\begin{aligned}
u_2(f(K), s_{-2}) &\leq u_2(f(K-2), s_{-2}) \\
\Rightarrow (r_d - f(K))(K-1) &\leq (r_d - f(K-2))(K-2) \\
\Rightarrow r_d &\leq (K-1)f(K) - (K-2)f(K-2) \\
&= (K-1)(f(K-1) + (K-2)!) - (K-2)f(K-2) \\
&= f(K) + (K-1)! \\
&= f(K+1)
\end{aligned}$$

where the second inequality follows since as we have already mentioned, player 2 wins over the competition by playing $f(K)$; the fourth and the last inequalities follow from Equation (9.5). Hence, player 2 may not deviate profitably from playing $f(K-2)$ while $r_d \leq f(K+1)$. The same holds for the rest of the players $\in V_{even}$ since their strategies are contingent on player 2's action r_2 . Finally, the case for the last player $2K-1$ is trivial which concludes the proof. \square

We have shown earlier in Claim (4) that no SPE exists in the 3-stage version of the game ($K=3$) by showing that the class of subgames for $r_d > 5 = f(3+1)$ do not have a Nash equilibrium. Theorem (8) explains the significance of the reward $r_d = 5$ where $5 = f(K+1)$ when $K=3$. Hence, the result for $K=3$ conforms to the general result of Theorem (8).

Before concluding this section, we construct the Nash equilibrium for the class of subgames $G(h^1)$ for $h^1 = r_d < f(K)$ on the K -stage ring as follows: Recall first that each player i at stage j observes the history h^j before playing and that $r_{next(R_i)}$ is a component of the history h_j and particularly of the action profile a^{j-1} (for example for any player i at stage 1, $r_{next(R_i)} = r_d$). For each player i at stage j , let $s_i^*(1) = 0$,

Chapter 9. Route Distribution Incentives in BGP

and let $s_i^*(h^j) = f(\kappa^* - 1)$ where $\kappa^* = \max_{\kappa < K} \kappa$ satisfying $f(\kappa) \leq r_{next(R_i)}$. Finally, let $s_{2K-1}^* = 0$.

Theorem 9. *The profile \mathbf{s}^* thus constructed is a Nash equilibrium for the class of subgames $G(h^1)$ for $h^1 = r_d < f(K)$ on the K -stage ring, $\forall K > 2$.*

Proof. For $r_d < f(K)$ there is no competition under s^* and for every player i at stage j , $r_{next(R_i)} < f(K - j + 1)$ which could be shown recursively starting with players at stage 1 and moving downwards. We show that no player can move profitably. By construction of the strategy, each player i will first observe the promised reward $r_{next(R_i)}$, then compute κ^* , then play $f(\kappa^* - 1)$. By definition of κ^* and by definition of f , it must be that for every player i , playing $f(\kappa^* - 1)$ will yield an outcome in which $\delta_i = (\kappa^* - 1)$. Additionally, $(r_{next(R_i)} - f(\kappa^* - 1))(\kappa^* - 1) \geq (r_{next(R_i)} - r'_i)\delta'_i$, $\forall r'_i \neq f(\kappa^* - 1)$. This is because playing $r'_i > f(\kappa^* - 1)$ will still yield a $\delta_i = (\kappa^* - 1)$ while playing $r'_i < f(\kappa^* - 1)$ will yield a $\delta'_i < \kappa^* - 1$ and a weakly lower utility by construction of f i.e., no player may deviate profitably. The case for the last player $2K - 1$ is trivial which concludes the proof. \square

With Theorems (9) and (8), we have constructed the Nash equilibria for the class of subgames $G(h^1)$ for $h^1 \leq f(K + 1)$ on the general K -stage ring. On the hand, we have also shown in Claim (4) that the subgame $G(h^1)$ for $h^1 > f(K + 1)$ does not have a Nash equilibrium. While the static multi-stage game does not exhibit an equilibrium for $r_d > f(K + 1)$, an equilibrium could be constructed in a finitely repeated version of the multi-stage game if we add a *convergence* constraint on utility which essentially dis-incentivizes oscillation as discussed next.

9.4.5 The Repeated Game

First, we revisit the concrete example of the 3-stage game for $r_d = 6$ in Figure 9.4. An important observation in this simple game is that player 1 may guarantee a payoff of 6 by committing to playing the pure strategy $r_1 = 3$ (assuming player 2 knows that) and there is no equilibrium that can yield player 1 a higher payoff. In the general setting when $r_d > 5$, it may be shown that player 1 may threaten player 2 by playing $r_1 = \lceil \frac{r_d}{2} \rceil$ since player 2 will have no incentive to deviate from playing $r_2 = 1$ given r_1 . In this case, the payoff to player 1 is r_d when r_d is even and $r_d - 1$ when r_d is odd, while the payoff of player 2 is $r_d - 1$.

While the static game is instructive, it fails to capture the repeated dynamics of BGP and the convergence concept introduced earlier. Many recent efforts have focused on modeling the repeated dynamics inherent to the BGP game [70, 71, 163, 168]. The repeated dynamics are critical to determining the outcome of the game. Afegan [163] shows that BGP is not incentive compatible in the repeated version of the game (at a specific Internet interchange) which violates the incentive-compatibility result of Feigenbaum et al. [166] obtained in the one-shot version of the game. In addition, the game-theoretic BGP model of Levin et al. [71] models the BGP convergence game as an infinitely repeated game and is based on the widely celebrated model of Griffin et al. [70]. We extend the basic game described thus far to model the repeated play in addition to strategic price setting.

In the repeated version of the game, after d advertises itself and declares r_d to its direct neighbors, a finite horizon, repeated, multi-stage game begins. Each round of the game is exactly the multi-stage game described earlier in section 9.4.1 and r_d remains unchanged throughout the game. Recall that the multi-stage model is intended to capture the order of play which is based on proximity to d . The multi-stage game is played a finite number of rounds whereby the rounds are intended to

Chapter 9. Route Distribution Incentives in BGP

capture repeated strategic price setting among the players over time. Before playing at stage k in round t , player i observes the complete history of all the previous rounds she participated in (rounds $0, 1, \dots, t-1$) as well as the history h^k in round t . We denote this complete history up to stage k in round t by $h^{k,t}$, which is an outcome of the game. Implicitly here, d observes the outcome of every round. For example, player 1 plays at stage 1 in round 0 after observing h^1 , then plays again at stage 1 in round 1 after observing $h^{1,1}$ which is the complete history/outcome of stage 0 in addition to h^1 in round 1, and so on.

Let M be the number of rounds played in the finite horizon game where M is sufficiently large and is common knowledge. The pure strategy of a player is now a map $\{s_i^t\}_{t=0}^{M-1}$ where s_i^t is player i 's strategy at round t which maps every possible complete history $h^{k,t}$ to a feasible action. The key property that may be defined in the repeated model is *convergence*. More clearly, we have specified earlier that a player receives no utility from an oscillating route. The static game fails to capture this concept of route stability since the game is played only once. The repeated play extension on the other hand allows us to more realistically model route convergence and the respective utility. We have defined convergence earlier to be the convergence of the outcome to a stable tree T_d given some strategy profile \mathbf{s} . We have shown that given any profile \mathbf{s} the protocol will converge, and indeed in the multi-stage model, it will do so in a single round, i.e., the outcome of each round t is a tree T_d^t . Convergence in the repeated game is the convergence of player strategies to an equilibrium. Players will only be rewarded for their *stable volume* and this is common knowledge, i.e., a player i may claim rewards from her upstream parent j only when i 's subtree is stable. As we shall see, it is this convergence requirement which results in an equilibrium emerging. Intuitively, we are saying that given that best response dynamics lead to oscillation in the static game, adding the requirement that players will only be paid if they converge will result in convergence. Formally, let T_i^t be i 's subtree in the outcome tree T_d^t of round t . In an M round game, T_i is stable only

Chapter 9. Route Distribution Incentives in BGP

when it remains unchanged from some round t_s onwards. The parameter M , number of rounds, represents the patience of the advertiser d . This is how much d is willing to wait for convergence before distributing any rewards to the players. Suppose that the players discount future payoffs with a common discount factor δ . In order to be able to compare the payoffs of the static game to those of the repeated game, the utility of a player i from repeated play of the multi-stage game, known as the *average discounted payoff* [63], is normalized and is given by,

$$\hat{u}_i(\mathbf{s}) = \frac{1 - \delta}{1 - \delta^M} \sum_{t=t_s}^{M-1} \delta^t u_i(s_i^t, s_{-i}^t) = \frac{\delta^{t_s} - \delta^M}{1 - \delta^M} u_i(s_i^t, s_{-i}^t) \quad (9.9)$$

where $u_i(s_i^t, s_{-i}^t)$ is the per-round payoff which depends solely on the players' actions and the outcome at round t and is given by Equation (9.1); and $t_s = \min(t)$ s.t. $T_i^t = T_i^{t'}, \forall t < t' \leq M$; and $\frac{1-\delta}{1-\delta^M}$ is simply a normalization factor which equates to $\frac{1}{\sum_{t=0}^{M-1} \delta^t}$. First, notice how the utility of a player decreases with t_s according to $\frac{\delta^{t_s} - \delta^M}{1 - \delta^M}$ (as t_s varies between 0 and $M - 1$). Notice also that when T_i^t is stable, it does not necessarily mean that T_j^t is stable $\forall j \in R_i$. We restrict u_i to consider T_i^t only since this is the subtree that i has control over. The reality is that every T_j^t for $j \in R_i$ must be stable before rewards may flow from d towards i .

Consider the following grim trigger strategy s_1^* for player 1 in the 3-stage game of Figure 9.4: let $s_1^*(1) = 0$, $s_1^*(2) = 1$ (Theorem (9)), $s_1^*(3) = s_1^*(4) = 2$ (Theorem (8)), and $s_1^*(r_d)$ for $r_d > 4$ be to play $r_1 = 2$ in round 0 and continue playing 2 until player 2 plays an $r_2 > 2$ after which switch to playing $r_1 = \lceil \frac{r_d}{2} \rceil$ for the rest of the game.⁹ Player 2's strategy s_2^* is to always play $s_2^*(r_d) = 1$ for $r_d \geq 2$ and 0 otherwise. Players 3 and 4 will repeatedly compete in every round playing the strategies of the static game. Finally, $s_5^* = 0$.

Theorem 10. *The strategy profile \mathbf{s}^* is a Nash equilibrium in the 3-stage ($K = 3$) game (for every history h^1).*

⁹Here player 1 has an advantage over player 2 and is threatening the latter to force a desirable outcome.

Chapter 9. Route Distribution Incentives in BGP

Proof. We show that \mathbf{s}^* is a Nash equilibrium for every history h^1 . The case for $h^1 = r_d < 5$ follows from Theorems (9) and (8) since by repeatedly playing the Nash strategies of the static game, then $t_s = 0$ and no player may deviate profitably as can be seen from Equation (9.9) since $\hat{u}_i = u_i, \forall i$.

For $r_d \geq 5$, player 1 is maximizing her average utility $\hat{u}_1(\mathbf{s}^*)$ by playing $r_1 = f(K-1) = 2$ (by definition of f) given that $r_2 = 1$ in every history and hence player 1 may not gain by any deviation. Notice that this is the only history we consider based on player 2's equilibrium strategy as we are constraining our attention to Nash equilibria. What remains is to show that player 2 may not gain by deviating from playing $r_2 = 1$ (i.e., not competing) in any single round while conforming to $r_2 = 1$ in every other round (this is true since player 2 has one of two options when playing in any round: compete, i.e., $r_2 \geq g_K(r_1)$, or don't compete, i.e., $r_2 = 1$; a strategy s_2 is a combination of these options across the rounds; by showing that competing in any single round, say t , may not benefit player 2, it follows directly as we shall show that competing in any future round $t' > t$ as well may not benefit player 2 given s_1^*). In order for player 2 to possibly increase her utility (by increasing δ_2), she must deviate by playing an $r_2 \geq g_K(r_1) = r_1 + 1$ as defined in section 9.4.4. If player 2 deviates in round 0, then given the threat strategy of player 1, player 2 will strictly decrease her average payoff since the deviation will cause $t_s > 0$ without any possibility of increasing u_2 for any $t > 0$ when player 1 switches to playing $r_1 = \lceil \frac{r_d}{2} \rceil$. This holds since $u_2(r_2, r_1)$ does not increase, i.e., $u_2(r'_2, \lceil \frac{r_d}{2} \rceil) < u_2(1, \lceil \frac{r_d}{2} \rceil), \forall r'_2 \geq r_1 = \lceil \frac{r_d}{2} \rceil$ (by construction of $r_1 = \lceil \frac{r_d}{2} \rceil$ as the minimum reward such that player 2 may not benefit by competing given r_d). This argument may be extended to every round in which player 2 participates. Finally, notice that players 3 and 4 may not deviate profitably from repeatedly playing the static strategies (the Bertrand competition) given the strategies of the rest of the players which concludes the proof. \square

For example, in the $r_d = 6$ subgame of Figure 9.4, at equilibrium players 1 and

2 are expected to play $(r_1, r_2) = (2, 1)$ in every round yielding $(\hat{u}_1, \hat{u}_2) = (8, 5)$ (again here ignoring the first term of Equation (9.1)). In the general setting when $r_d \geq 3$, the equilibrium action profile remains $(r_1, r_2) = (2, 1)$ in every round yielding $(\hat{u}_1, \hat{u}_2) = (2r_d - 4, r_d - 1)$. In summary, while no Nash equilibrium exists in the static game, an equilibrium emerges in the repeated model mainly due to the convergence restriction on the players' payoffs which essentially restricts player 2 to concede and avoid oscillation. However, in order for this equilibrium to emerge, player 2 must be aware of the threat strategy of player 1. The result for the 3-stage repeated game on the ring may be extended to the K -stage repeated game.

9.5 Discussion

The Nash equilibria constructed in this chapter are not unique. It is additionally well known that in a multi-stage game setting, the Nash equilibrium notion might not be “credible” as it could present suboptimal responses to histories that would not occur under the equilibrium profile [63], rendering subgame perfect equilibria more suitable in such circumstances. All the Nash equilibria that we have constructed are credible and are consistent with backward induction for the respective histories of the subgames studied. A distinct aspect of our game is that a player i at stage k may not carry an empty threat to an upstream parent at stage $k - 1$, since player i 's actions are constrained by the parent's action as dictated by the network structure and the decreasing rewards assumption. In this chapter, we have studied the equilibria existence question only. Other important questions include quantifying how hard is it to find the equilibria, and devising mechanisms to get to them. These questions, in addition to extending the results to general network structures and relaxing the fixed cost assumption, are part of our ongoing work.

While the distributed incentive model has advantages over centralized mecha-

nisms that rely on a “designer”, the model might suffer from exponential growth of rewards which could potentially make it infeasible for sparse and large diameter networks. Quantifying the suitability of this model to general network structures and to the Internet connectivity graph specifically requires further investigation. Interestingly, while it is a complex network, the Internet’s connectivity graph is a *small-world* network, i.e., the average distance between any two nodes on the Internet is small [169].

Finally, we have only considered the setting in which d ’s marginal utility is constant which seems intuitive in a BGP setting where global reachability is the goal, since every node in the DFZ must keep state information about d or else the latter will be unreachable from some parts of the network. Other economic models that assume the network is a market with elastic demand (based on d ’s utility) and that determine prices based on demand and supply, are interesting to investigate. They may even be more intuitive in settings where it makes sense to advertise (or sell) a piece of information to a local neighborhood.

Appendix: Existence of $g_K(x)$

It is straightforward to show that $g_3(x) = x + 1$ given the Bertrand competition of players 3 and 4 on the 3-stage ring. For $K \geq 4$ and for any $r_1 = x \geq f(K - 1)$ ¹⁰,

¹⁰When $x < f(K - 1)$, then $g_K(x) = f(K - 1)$ by definition of f .

Chapter 9. Route Distribution Incentives in BGP

$\hat{\mathbf{s}}_{V_{even}}$ is part of the solution to the following Integer Linear Program (ILP) ¹¹:

$$\begin{aligned}
& \text{minimize} && \sum_{\substack{i=2 \\ i \text{ odd}}}^{2K-5} r_i - \beta \sum_{\substack{i=2 \\ i \text{ even}}}^{2K-5} r_i \\
& \text{s.t.} && \\
& && -r_{2K-5} + r_{2K-4} = 1 \\
& \forall 3 \leq i \leq 2k-4, && (r_{i-2} - r_i)(k - l(i)) \geq \\
& && (r_{i-2} - f(k - l(i) - 1))(k - l(i) - 1) \\
& \forall 1 \leq j \leq k-3, && r_{2j} \geq r_{2j-1} + 1
\end{aligned}$$

where β is a sufficiently large constant. The variables in the ILP above signify the actions of the players in the subgame $G(h^2)$ while the constraints guarantee that all players compete while they have an incentive to do so knowing that each player may choose between competing or not. The constraints are constructed based on the definition of $\hat{\mathbf{s}}_{V_{even}}$ to make sure that players in V_{odd} have no incentive to compete. Figure 9.5 shows a plot of $g_K(x) - x$ for different values of K and for $x \geq f(K-1)$ where the solution to the above ILP (including $r_2 = g_K(x)$) is computed using the *lp_solve* [170] ILP solver (we set $\beta = 10^5$).

¹¹Where $g_K(x)$ is the r_2 element in the solution.

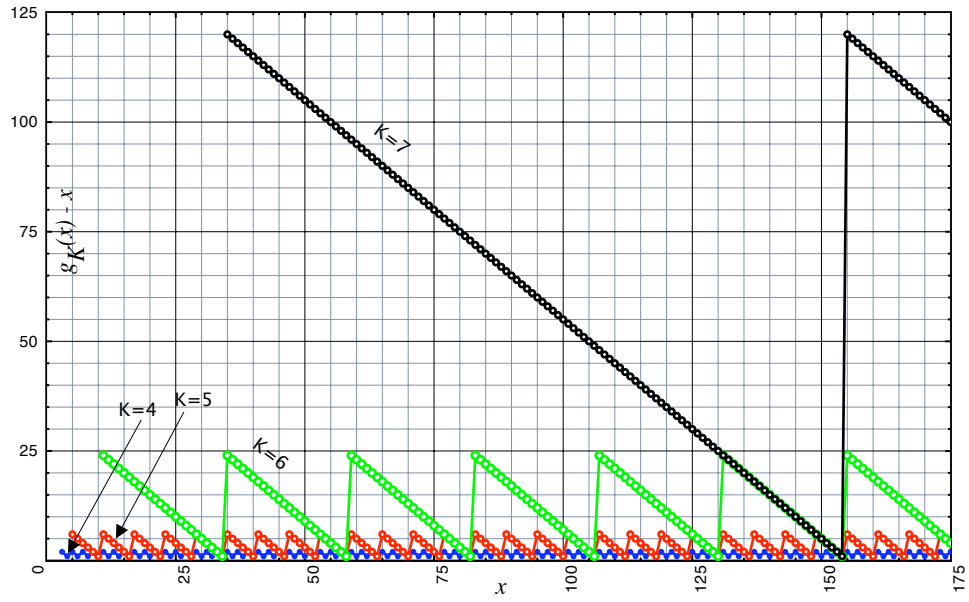


Figure 9.5: Plot of $g_K(x) - x$ for $K = 4, 5, 6, 7$.

Part III

Conclusion

Chapter 10

Conclusion

The reasonable man adapts himself to the world; the unreasonable one persists in trying to adapt the world to himself. Therefore all progress depends on the unreasonable man.

- GEORGE BERNARD SHAW (Playwright, winner of a Nobel prize and an Oscar, and author of *Maxims for Revolutionists*)

In this dissertation, we have explored two problems towards designing a future Internet: the lack of a framework for thinking about network architecture, and the limitations of traditional discovery schemes. The thesis of the dissertation is thus two fold:

- *Architectural contributions will remain idiosyncratic and hard to assess and evaluate unless we devise the correct frameworks to think about the design space, compare architectures, and analyze their properties and their design implications.*
- *The separation of identifier and locator is a widely accepted design principle for a future Internet. The design of discovery models to support this separation*

benefits from accounting for the valuations of (and the cost incurred by) the parties involved.

Our work is the first to present a general information-centric taxonomy for thinking about the network architecture space. Additionally, this work is the first to frame and formalize the discovery problem along with two important aspects of its design process: *service differentiation* and *economic incentives*. By adding an economic dimension to the discovery design space, we gained more knowledge about the complex design decisions pertaining to naming and discovery in networks.

10.1 Open Problems

Designing a future Internet architecture is a very broad problem statement. The community is still in the process of converging on a set of goals and requirements for the future Internet. Much work remains to be done in the requirements analysis phase before we can start seeing concrete outcomes towards the final goal. Our work addresses several aspects of the design process to help guide the design of a future Internet. We reflect on some of the open problems that were suggested by our research and the respective research opportunities:

First, in terms of architectural modeling, while chapters 4 and 5 present an important step towards formalizing network architectures and architectural styles, several research challenges remain to be solved. Extracting a complete and disjoint set of network architectural styles may potentially frame the architectural problem and provide a formal framework for classifying, relating, and reasoning about architectures. The taxonomy is a timely and essential contribution in that direction. However, we believe that several technological and cultural limitations remain to be solved before assessing whether formal modeling will be of significant value to the commu-

Chapter 10. Conclusion

nity. Some of the main limitations we encountered include the difficulty of modeling the behavioral aspects of the architecture (mainly dynamics and evolution), the difficulty of composing large complex systems in formal models, and the difficulty of large scale verification. While recent work has focused on addressing some of these limitations (*e.g.* [138, 171]), significant work remains to be done (engineering tools and techniques, formal languages, and cultural changes are necessary) before formal architectural modeling becomes a reality.

In chapter 7 we presented a framework for discovery service differentiation. The cost associated with “being discoverable” and the resulting incentive mismatch problem that arises were elaborated on in chapter 8. In light of these contributions, an important question is whether one can design a mechanism that allows differentiation and that accounts for the cost and the incentives. A player that demands and receives a higher discovery level is technically introducing more cost into the system i.e. the distributed state is generally proportional to the discovery performance attained. A cost-sharing discovery mechanism determines “how discoverable” players are, and the payments or cost shares they have to make. We sketch one such mechanism based on the mechanism design framework in Appendix A. Some of the main limitations of this model (and of the AMD framework in general) include: (1) formulating a cost function that is tractable given its dependence on globally distributed state; (2) devising distributed implementations of the mechanism and studying their algorithmic complexity [67] for the different discovery models; and (3) investigating the feasibility of implementing the distributed schemes as scalable extensions to legacy discovery schemes such as BGP, DNS, and DHT as identified in [67]. While it provides tractable results, algorithmic mechanism design [64] suffers from several limitations including (i) the simple one-shot model versus the more realistic repeated dynamics that are prevalent in distributed settings, (ii) the obliviousness to malicious behavior, (iii) the inability to concurrently account for demand and supply [79], and the computational hardness of implementing the mechanism in

Chapter 10. Conclusion

a distributed setting [74]. These limitations have guided our decision to investigate a rather distributed incentive model in chapter 9. However, introducing differentiation as part of the rewarding model, devising algorithms that lead to the constructed equilibria, and extending the model to general network structures remain part of our future work.

In chapter 6 we have explained two main aspects of the identifier-based discovery process: (i) advertisers wishing to be discoverable by advertising themselves, and (ii) seekers seeking advertisers. These two aspects have different value arguments and cost structures as we explained in section 8.2.2. For example, sometimes the seeker who seeks a resource gets value from the path discovery process. However, the game model examined in chapter 9 in the context of BGP considers value as it relates to the advertiser only. The other game that considers the forwarding part of discovery is concerned with transit costs versus distribution costs. In summary, the two aspects of the discovery process are distribution and forwarding in the context of BGP. Whether the two games may be combined and analyzed simultaneously is an open question.

Part IV

Appendices

Appendix A

A Mechanism Design Model

The goal in this section is to design identifier-based discovery mechanisms that are: (1) distributed (i.e. inputs and outputs of the mechanism are distributed throughout the network as defined in Distributed AMD [67]), (2) efficient (i.e. the mechanism will try to maximize some concept of *social welfare*), (3) incentive-compatible (i.e. the players will not try to manipulate the outcome of the mechanism to their benefit), and most importantly (4) cost-sharing and possibly budget-balanced. Recall that budget-balance occurs when the global cost of the mechanism is offset by the payments made by the players. Note that the problem we are currently addressing assumes that all participating nodes cooperate to implement routing and forwarding i.e. nodes do not try to computationally manipulate the routing/forwarding functions. The only strategic aspect of our current model is the players' valuations of discovery levels that are declared to the mechanism designer. Such assumption is directly applicable in schemes that follow model I in Figure 8.3 since the players can not manipulate routing when $V_D \neq P$.

A.1 The Discovery Mechanism

The ingredients of the discovery mechanism are: (1) an input valuation function $v_i(\cdot)$ for each player i where $v_i \in V_i$, the latter being the valuation function space $V_i \subseteq \mathbb{R}^\Lambda$, (2) an output function $O : V_1 \times V_2 \dots \times V_N \rightarrow \Lambda^N$ which utilizes some discovery scheme to deliver a discovery level profile \mathbf{L} to the players, $\mathbf{L} \in \Lambda^N$, and (3) a cost-sharing function $\xi : V_1 \times V_2 \dots \times V_N \rightarrow \mathbb{R}^N$ that distributes the payments \mathbf{p} to the players, $\mathbf{p} \in \mathbb{R}^N$. Hence, the discovery mechanism $M : V_1 \times V_2 \dots \times V_N \rightarrow \Lambda^N \times \mathbb{R}^N$ maps valuations to a discovery level profile and a payment profile. We shall briefly describe each of the ingredients next.

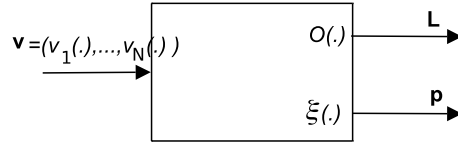


Figure A.1: The discovery mechanism design framework: mechanism $M = (O, \xi)$

The valuation function: Each player i has a private valuation function $v_i : \Lambda \rightarrow \mathbb{R}$, that assigns a real value to each possible discovery level $l \in \Lambda$. Intuitively, a player will have a valuation that matches its true internal business requirements - user demand internal to the player/node will require a certain performance level to satisfy the demand.

The output function: An output of the mechanism is a decision that aggregates the players' valuations. More clearly, the output function $O : V_1 \times V_2 \times \dots \times V_N \rightarrow \Lambda^N$ maps the players' valuations to a discovery level profile $\mathbf{L} \in \Lambda^N$, delivered to the players. Denote by $(\mathbf{L})_i$ the element of vector \mathbf{L} (i.e. the discovery level l) delivered to player i , and by $(\mathbf{L})_{-i}$ the profile delivered to the rest of the players. To deliver a discovery level profile, the mechanism relies on a discovery scheme, denoted as (D, Al) . The discovery scheme (D, Al) dictates 1) how the namespace registrations

Appendix A. A Mechanism Design Model

(or state) are distributed on the nodes $V_D \subseteq V$ (denoted by D), and 2) how the search queries are forwarded such that players will be discoverable (denoted by algorithm Al). Let S_u be the registration state maintained at node $u \in V_D$ and let S be the global state under D , i.e. $S = \bigcup_{u \in V_D} S_u$.

The cost-sharing scheme ξ : In addition to delivering a discovery profile, the mechanism implements a function ξ that distributes payments p_i to the players (entities) where p_i is the amount player i pays to the mechanism.

The cost function C : The cost function is defined by $C : \Lambda^N \rightarrow \mathbb{R}^+ \cup \{0\}$ that assigns to each output profile a real cost for delivering the profile. Given that a scheme (D, Al) will assign a set of registrations S_u to each node u that delivers an output profile \mathbf{L} , the total cost associated with \mathbf{L} under some fixed scheme (D, Al) is $C_{(D, Al)}(\mathbf{L}) = \sum_{u \in V_D} cost(S_u)$ where $cost$ is the cost function of maintaining the S_u registrations at node u . In this sense, the cost we try to formulate is the control plane cost of the discovery scheme (D, Al) ¹. The mechanism assumes the existence of some stable scheme (D, Al) and the cost is minimized over the argument S where the former could be suboptimal. By fixing the discovery scheme, the stability of the mechanism increases and the network complexity that might arise due to changes in v_i decreases.

Utility and welfare: The value that a player i obtains as a result of an output profile (\mathbf{L}) is simply her valuation of the discovery level she receives $v_i((\mathbf{L})_i)$ ². The utility of player i is $u_i = v_i - p_i$. It is implicitly assumed here that the player's preferences are quasi-linear and that no externalities exist i.e. player only cares about the discovery level she receives and not about other player levels. The global welfare of all the players under a scheme (D, Al) is, $NW(\mathbf{L}) = \sum_i v_i(\mathbf{L}) - C_{(D, Al)}(\mathbf{L})$ which

¹Note that we are not accounting for the forwarding plane costs which could be handled through per query rewards. We are solely concerned with the initial cost of constructing and maintaining the state.

²This is the value of being globally discoverable or known at the expected level i.e. the value of being famous!.

Appendix A. A Mechanism Design Model

measures the total benefit obtained by all players independent of payments. A mechanism is said to be efficient if it maximizes the global welfare $NW(L)$ implementing a social choice function.

References

- [1] V. Cerf and R. Kahn, *Communications, IEEE Transactions on*, vol. 22, no. 5, pp. 637–648, 1974.
- [2] D. Clark, “The design philosophy of the darpa internet protocols,” in *SIGCOMM ’88: Symposium proceedings on Communications architectures and protocols*. New York, NY, USA: ACM Press, 1988, pp. 106–114.
- [3] “NSF Nets FIND initiative,” <http://www.nets-find.net/>.
- [4] “GENI: Global environment for network innovations,” <http://www.geni.org/>.
- [5] “European future internet portal,” <http://www.future-internet.eu/>.
- [6] “The AKARI architecture design project,” <http://akari-project.nict.go.jp>.
- [7] “The internet is broken,” MIT Technology Review. <http://www.technologyreview.com/article/16356/>, December 2005.
- [8] S. M. Bellovin, D. D. Clark, A. Perrig, and D. Song, “A clean-slate design for the next-generation secure internet,” March 2005, this is the report of an NSF workshop held in July, 2005.
- [9] A. Feldmann, “Internet clean-slate design: what and why?” *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 59–64, 2007.
- [10] J. H. Saltzer, D. P. Reed, and D. D. Clark, “End-to-end arguments in system design,” *ACM TOCS*, vol. 2, no. 4, pp. 277–288, November 1984.
- [11] J. N. Chiappa, “Endpoints and endpoint names: A proposed enhancement to the internet architecture,” <http://www.chiappa.net/tech/endpoints.txt>, 1999.
- [12] D. Meyer, L. Zhang, and K. Fall, “Report from the iab workshop on routing and addressing,” Internet RFC 4984, Sep 2007.

References

- [13] D. Krioukov, kc claffy, K. Fall, and A. Brady, “On compact routing for the internet,” *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 41–52, 2007.
- [14] M. S. Blumenthal and D. D. Clark, “Rethinking the design of the internet: the end-to-end arguments vs. the brave new world,” *Communications Policy in Transition: The Internet and Beyond*, pp. 91–139, 2001.
- [15] A. Weiss, “Net neutrality?: there’s nothing neutral about it,” *netWorker*, vol. 10, no. 2, pp. 18–25, 2006.
- [16] W. R. Stevens and K. Fall, *TCP/IP Illustrated: The Protocols v. 1*. USA: Addison-Wesley Publishing Company, 2009.
- [17] S. Floyd and V. Paxson, “Difficulties in simulating the internet,” *IEEE/ACM Trans. Netw.*, vol. 9, no. 4, pp. 392–403, 2001.
- [18] M. Coates, A. Hero, R. Nowak, and B. Yu, “Internet tomography,” *IEEE Signal Processing Magazine*, vol. 19, pp. 47–65, 2002.
- [19] D. Garlan, R. Allen, and J. Ockerbloom, “Architectural mismatch, or, why it’s hard to build systems out of existing parts,” in *Proceedings of the 17th International Conference on Software Engineering*, Seattle, Washington, April 1995, pp. 179–185.
- [20] M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, 1996.
- [21] P. Zave, “A formal model of addressing for interoperating networks,” in *FM*, 2005, pp. 318–333.
- [22] —, “Compositional binding in network domains.” in *FM*, ser. Lecture Notes in Computer Science, vol. 4085. Springer, 2006, pp. 332–347.
- [23] T. G. Griffin and J. L. Sobrinho, “Metarouting,” in *SIGCOMM ’05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2005, pp. 1–12.
- [24] M. Karsten, S. Keshav, S. Prasad, and M. Beg, “An axiomatic basis for communication,” in *Proceedings of SIGCOMM 2007*. New York, NY, USA: ACM Press, 2007, pp. 217–228.
- [25] Y. Wang, J. Touch, and J. Silvester, “A unified model for end point resolution and domain conversion for multi-hop, multi-layer communication,” USC/ISI, Technical Report ISI-TR-590, June 2004.

References

- [26] A. C. Snoeren and H. Balakrishnan, “An end-to-end approach to host mobility,” in *Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, August 2000.
- [27] C. E. Perkins, “RFC 3220: Ip mobility support for ipv4,” January 2002.
- [28] R. Moskowitz, P. Nikander, and P. Jokela, “Host identity protocol architecture,” RFC 4423, May 2006.
- [29] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, “Internet indirection infrastructure,” *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 205–218, Apr. 2004.
- [30] D. Clark, R. Braden, A. Falk, and V. Pingali, “Fara: reorganizing the addressing architecture,” in *FDNA '03: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*. New York, NY, USA: ACM Press, 2003, pp. 313–321.
- [31] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish, “A layered naming architecture for the internet,” in *Proceedings of ACM SIGCOMM 2004*. Portland, Oregon, USA: ACM, August 2004, pp. 343–352.
- [32] P. Nikander, “Identifier / locator separation: Exploration of the design space (ilse),” IETF Internet Draft draft-nikander-ram-ilse-00, august 2007.
- [33] “The transient network architecture,” <http://hdl.handle.net/2118/tna>.
- [34] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, “A data-oriented network architecture,” in *Proceedings of SIGCOMM'07*. Kyoto, Japan: ACM, August 27-31 2007.
- [35] “The FP7 4WARD project,” <http://www.4ward-project.eu/>.
- [36] V. Fuller, D. Meyer, and D. Farinacci, “Lisp alternative topology (lisp+alt),” <http://tools.ietf.org/html/draft-fuller-lisp-alt-03.txt>.
- [37] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, “Rofl: routing on flat labels,” in *Proceedings of SIGCOMM 2006*. New York, NY, USA: ACM Press, 2006, pp. 363–374.
- [38] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg, “Compact distributed data structures for adaptive routing,” in *ACM STOC '89*. New York, NY, USA: ACM, 1989, pp. 479–489.

References

- [39] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup, “Compact name-independent routing with minimum stretch,” in *ACM SPAA '04*. New York, NY, USA: ACM Press, 2004, pp. 20–24.
- [40] L. Bent and G. M. Voelker, “Whole page performance,” Workshop on Web Content Caching and Distribution, Boulder CO, August 2002.
- [41] C. Huitema and S. Weerahandi, “Internet measurements: the rising tide and the dns snag,” in *Proceedings of the 13th ITC Specialist Seminar on IP Traffic Measurement Modeling and Management*, ser. IPseminar. Monterrey, CA, USA: ITC, September 18-20 2000.
- [42] Y. Rekhter, T. Li, and S. Hares, “RFC 4271: A border gateway protocol 4 (bgp-4),” 2006.
- [43] B. Awerbuch and D. Peleg, “Routing with polynomial communication-space trade-off,” *SIAM J. Discret. Math.*, vol. 5, no. 2, pp. 151–162, 1992.
- [44] C. Gavoille, “Routing in distributed networks: overview and open problems,” *SIGACT News*, vol. 32, no. 1, pp. 36–52, 2001.
- [45] C. Gavoille and D. Peleg, “Compact and localized distributed data structures,” *Distrib. Comput.*, vol. 16, no. 2-3, pp. 111–120, 2003.
- [46] S. Shenker, “Fundamental design issues for the future internet,” *IEEE Journal on Selected Areas in Communication*, vol. 13, no. 7, September 1995.
- [47] D. D. Clark, K. Sollins, J. Wroclawski, and T. Faber, “Addressing reality: an architectural response to real-world demands on the evolving internet,” in *FDNA '03: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*. New York, NY, USA: ACM Press, 2003, pp. 247–257.
- [48] B. Ahlgren, M. Brunner, L. Eggert, R. Hancock, and S. Schmid, “Invariants: a new design methodology for network architectures,” in *FDNA '04: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*. New York, NY, USA: ACM Press, 2004, pp. 65–70.
- [49] S. Ratnasamy, S. Shenker, and S. McCanne, “Towards an evolvable internet architecture,” in *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2005, pp. 313–324.
- [50] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden, “Tussle in cyberspace: defining tomorrow’s internet,” *IEEE/ACM Trans. Netw.*, vol. 13, no. 3, pp. 462–475, 2005.

References

- [51] G. D. Abowd, R. Allen, and D. Garlan, “Formalizing style to understand descriptions of software architecture,” *ACM Trans. Softw. Eng. Methodol.*, vol. 4, no. 4, pp. 319–364, 1995.
- [52] “The ABLE Project at carnegie mellon university,” <http://www.cs.cmu.edu/~able>.
- [53] N. Medvidovic and R. N. Taylor, “A framework for classifying and comparing architecture description languages,” in *ESEC '97/FSE-5: Proceedings of the 6th European conference held jointly with the 5th ACM SIGSOFT international symposium on Foundations of software engineering*. New York, NY, USA: Springer-Verlag New York, Inc., 1997, pp. 60–76.
- [54] R. Allen and D. Garlan, “A case study in architectural modelling: The aegis system,” in *IWSSD '96: Proceedings of the 8th International Workshop on Software Specification and Design*. Washington, DC, USA: IEEE Computer Society, 1996, p. 6.
- [55] J. S. Kim and D. Garlan, “Analyzing architectural styles with alloy,” in *ROSATEA '06: Proceedings of the ISSA 2006 workshop on Role of software architecture for testing and analysis*. New York, NY, USA: ACM Press, 2006, pp. 70–80.
- [56] M. Bernardo, P. Ciancarini, and L. Donatiello, “Architecting families of software systems with process algebras,” *ACM Trans. Softw. Eng. Methodol.*, vol. 11, no. 4, pp. 386–426, 2002.
- [57] F. Oquendo, “π-method: a model-driven formal method for architecture-centric software engineering,” *SIGSOFT Softw. Eng. Notes*, vol. 31, no. 3, pp. 1–13, 2006.
- [58] R. Braden, D. Clark, and S. Shenker, “Integrated services in the internet architecture: an overview,” RFC 1633, 1994.
- [59] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, “An architecture for differentiated service,” RFC 2475, 1998.
- [60] X. Zhou, J. Wei, and C.-Z. Xu, “Quality-of-service differentiation on the internet: a taxonomy,” *J. Netw. Comput. Appl.*, vol. 30, no. 1, pp. 354–383, 2007.
- [61] C. Papadimitriou, “Algorithms, games, and the internet,” in *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 2001, pp. 749–753.

References

- [62] J. Kleinberg, “The convergence of social and technological networks,” *Commun. ACM*, vol. 51, no. 11, pp. 66–72, 2008.
- [63] D. Fudenberg and J. Tirole, *Game Theory*. MIT Press, October 1991.
- [64] N. Nisan and A. Ronen, “Algorithmic mechanism design,” in *Proceedings of the 31st ACM Symposium on Theory of Computing*, Atlanta, GA, May 1999.
- [65] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007.
- [66] A. Mas-Colell, M. D. Whinston, and J. R. Green, *Microeconomic Theory*. Oxford University Press, June 1995.
- [67] J. Feigenbaum and S. Shenker, “Distributed algorithmic mechanism design: Recent results and future directions,” in *In Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*. ACM Press, 2002, pp. 1–13.
- [68] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker, “A BGP-based mechanism for lowest-cost routing,” *Distrib. Comput.*, vol. 18, no. 1, pp. 61–72, 2005.
- [69] S. Yuen and B. Li, “Strategyproof mechanisms towards dynamic topology formation in autonomous networks,” *Mob. Netw. Appl.*, vol. 10, no. 6, pp. 961–970, 2005.
- [70] T. G. Griffin, F. B. Shepherd, and G. Wilfong, “Policy disputes in path-vector protocols,” in *ICNP ’99: Proceedings of the Seventh Annual International Conference on Network Protocols*. Washington, DC, USA: IEEE Computer Society, 1999, p. 21.
- [71] H. Levin, M. Schapira, and A. Zohar, “Interdomain routing and games,” in *STOC ’08: Proceedings of the 40th annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 2008, pp. 57–66.
- [72] L. Anderegg and S. Eidenbenz, “Ad hoc-VCG: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents,” in *MobiCom ’03: Proceedings of the 9th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2003, pp. 245–259.
- [73] J. Feigenbaum, C. H. Papadimitriou, and S. Shenker, “Sharing the cost of multicast transmissions,” *Journal of Computer and System Sciences*, vol. 63, no. 1, pp. 21–41, 2001.

References

- [74] R. Sami, “Distributed algorithmic mechanism design,” Ph.D. dissertation, New Haven, CT, USA, 2003, director-Joan Feigenbaum.
- [75] A. Fabrikant, A. Luthra, and C. H. Papadimitriou, “On a network creation game,” in *ACM PODC*. ACM Press, 2003, pp. 347–351.
- [76] N. Christin and J. Chuang, “On the cost of participating in a peer-to-peer network,” In IPTPS’04, 2004.
- [77] C. Li, B. Yu, and K. Sycara, “An incentive mechanism for message relaying in unstructured peer-to-peer systems,” in *AAMAS ’07: Proceedings of the 6th international conference on Autonomous agents and multiagent systems*. ACM, 2007, pp. 1–8.
- [78] J. Kleinberg and P. Raghavan, “Query incentive networks,” in *FOCS ’05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 132–141.
- [79] S. J. Bauer and P. Faratin, “Assessing the assumptions underlying mechanism design for the internet,” In *Proceedings of Economics of Networked Systems (NetEcon’06)*, Michigan University, Ann Arbor, US, 2006.
- [80] P. F. Ramakrishna, “Ipn1: A nat-extended internet architecture,” in *Proceedings of SIGCOMM 2001*. New York, NY, USA: ACM Press, 2001, pp. 69–80.
- [81] Z. Turányi, A. Valkó, and A. T. Campbell, “4+4: an architecture for evolving the internet address space back toward transparency,” *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 5, pp. 43–54, 2003.
- [82] T. S. E. Ng, I. Stoica, and H. Zhang, “A waypoint service approach to connect heterogeneous internet address spaces,” in *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2001, pp. 319–332.
- [83] “Delay tolerant networking research group,” <http://www.dtnrg.org/>.
- [84] M. Gruteser, “A geometric stack for location-aware networking,” NSF Nets FIND Initiative.
- [85] “NSF nets FIND: Breakout on networking at the information layer, 4th pi meeting,” [online]: [www.nets-find.net/Meetings/FourthPIMeeting/ BreakOut/InformationLayer.pdf](http://www.nets-find.net/Meetings/FourthPIMeeting/BreakOut/InformationLayer.pdf), November 2007.

References

- [86] A. T. Campbell, H. G. D. Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A survey of programmable networks," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 2, pp. 7–23, 1999.
- [87] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research," *IEEE Communications Magazine*, vol. 35, no. 1, pp. 80–86, 1997.
- [88] M. Gritter and D. R. Cheriton, "An architecture for content routing support in the internet," in *USITS'01: Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems*. Berkeley, CA, USA: USENIX Association, 2001, pp. 4–4.
- [89] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield, "Plutarch: an argument for network pluralism," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 4, pp. 258–266, 2003.
- [90] S. Schmid, L. Eggert, M. Brunner, and J. Quittek, "TurfNet: an architecture for dynamically composable networks." in *WAC*, 2004, pp. 94–114.
- [91] R. Yates, D. Raychaudhuri, S. Paul, and J. Kurose, "Postcards from the edge: A cache-and-forward architecture for the future internet," NSF Nets FIND Initiative.
- [92] A. Venkataramani and D. Towsley, "A swarming architecture for internet data transfer," NSF Nets FIND Initiative.
- [93] T. Wolf, "Service-centric end-to-end abstractions for network architecture," NSF Nets FIND Initiative.
- [94] Z. I. Boon Thau Loo, Jonathan Smith, "Wireless knowledge infrastructure (wiki)," NSF Nets FIND Initiative.
- [95] R. Kahn, C. Abdallah, H. Jerez, G. Heileman, and W. Shu, "The Transient Network Architecture (TNA)," NSF Nets FIND Initiative.
- [96] B. Bhattacharjee, K. Calvert, J. Griffioen, N. Spring, and J. Sterbenz, "Post-modern internetwork architecture," NSF Nets FIND Initiative.
- [97] Cisco Systems Inc., "Internetworking technologies handbook," Indianapolis, IN: Cisco Press, 2004.
- [98] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Symposium on Operating Systems Principles*, 2001, pp. 131–145.

References

- [99] X. Yang, “Nira: a new internet routing architecture,” in *FDNA '03: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*. New York, NY, USA: ACM Press, 2003, pp. 301–312.
- [100] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, “Chord: a scalable peer-to-peer lookup protocol for internet applications,” *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, 2003.
- [101] D. Mazières, M. Kaminsky, M. F. Kaashoek, and E. Witchel, “Separating key management from file system security,” in *SOSP '99: Proceedings of the seventeenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM Press, 1999, pp. 124–139.
- [102] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, “A knowledge plane for the internet,” in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2003, pp. 3–10.
- [103] “DARPA active nets,” <http://www.sds.lcs.mit.edu/darpa-activenet/>.
- [104] *2002 DARPA Active Networks Conference and Exposition (DANCE 2002), 29-31 May 2002, San Francisco, CA, USA*. IEEE Computer Society, 2002.
- [105] D. L. Tennenhouse and D. J. Wetherall, “Towards an active network architecture,” *SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 2, pp. 5–17, 1996.
- [106] M. W. Hicks, A. D. Keromytis, and J. M. Smith, “A secure plan.” in *DANCE*, 2002, pp. 224–237.
- [107] M. Hicks, J. Moore, D. Alexander, C. Gunter, and S. Nettles, “Planet: An active internetwork,” 1999.
- [108] J. Khoury, J. Crichigno, H. Jerez, C. Abdallah, W. Shu, and G. Heileman, “The intermesh architecture (student demo),” in *MobiCom 07*, Montreal, Canada, September 2007.
- [109] —, “The intermesh network architecture,” University of New Mexico, Technical Report ECE-TR-07-007, April 2007, [online]: <http://hdl.handle.net/1928/3052>.
- [110] C. Kim and J. Rexford, “Revisiting ethernet: Plug-and-play made scalable and efficient,” to appear in Proc. of IEEE LANMAN Workshop 2007.

References

- [111] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, "Opendht: a public dht service and its uses," in *Proceedings of SIGCOMM '05*. New York, NY, USA: ACM Press, 2005, pp. 73–84.
- [112] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols," in *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 3, 2005, pp. 1664–1669 Vol. 3.
- [113] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In vini veritas: realistic and controlled network experimentation," in *Proceedings of SIGCOMM '06*. New York, NY, USA: ACM Press, 2006, pp. 3–14.
- [114] G. F. Coulouris and J. Dollimore, *Distributed systems: concepts and design*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1988.
- [115] T. C. Du, E. Y. Li, and A.-P. Chang, "Mobile agents in distributed network management," *Commun. ACM*, vol. 46, no. 7, pp. 127–132, 2003.
- [116] P. Mockapetris, "RFC 1035: Domain names implementation and specification," November 1987.
- [117] "The handle system," <http://www.handle.net>.
- [118] J. Piovesan, C. Abdallah, H. Tanner, H. Jerez, and J. Khoury, "Resource allocation for multi-agent problems in the design of future communication networks," University of New Mexico, Technical Report EECE-TR-07-001, April 2007, [online]: <http://hdl.handle.net/1928/2973>.
- [119] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, 2000.
- [120] "Glib library," <http://www.gtk.org/>.
- [121] "Ad-hoc wireless distribution service - awds," <http://awds.berlios.de/>.
- [122] J. S. Khoury, H. Jerez, and L. De Cicco, "Design and implementation of a framework for persistent identification and communication in emerging networks," in *TridentCom '08: Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1–8.
- [123] "PlanetLab," <http://www.planet-lab.org>.

References

- [124] A. Greenhalgh, F. Huici, M. Hoerdt, P. Papadimitriou, M. Handley, and L. Mathy, “Flow processing and the rise of commodity network hardware,” *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 2, pp. 20–26, 2009.
- [125] “Akamai Technologies,” <http://www.akamai.com>, 2008.
- [126] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.
- [127] W. K. Giloi, “Towards a taxonomy of computer architecture based on the machine data type view,” in *ISCA '83: Proceedings of the 10th annual international symposium on Computer architecture*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1983, pp. 6–15.
- [128] D. Clark, B. Lehr, P. Faratin, R. Sami, and J. Wroclawski, “Overlay networks and future of the internet,” *Communications and Strategies*, no. 63, pp. 1–21, 2006.
- [129] M. Augusto, L. Silva, J. Quaini-Sousa, F. Redigolo, T. C. Carvalho, H. C. Guardia, W. Ruggiero, and B. Ohlman, “A proposal for a taxonomy for virtual networks,” Real Overlays And Distributed Systems (ROADS) workshop, Poland, 2007.
- [130] M. Maheswaran and S. Ali, “A taxonomy of network computing systems,” *Computer*, vol. 37, no. 10, pp. 115–117, 2004.
- [131] R. Buyya, M. Pathan, and A. Vakali, *Content Delivery Networks*, ser. Lecture Notes in Electrical Engineering. Germany: Springer-Verlag, 2008, vol. 9.
- [132] H. Kung and C. Wu, “Content networks: Taxonomy and new approaches,” in *The Internet as a Largescale Complex System*, ser. Santa Fe Institute Series. Oxford Press, 2002.
- [133] J. Khoury and C. Abdallah, “Towards a taxonomy of inter-network architectures,” University of New Mexico, Technical Report ECE-TR-08-008, June 2008, [online]: <http://hdl.handle.net/1928/6626>.
- [134] D. Jackson, *Software Abstractions: Logic, Language, and Analysis*. The MIT Press, 2006.
- [135] “The alloy analyzer,” <http://alloy.mit.edu/>.
- [136] S. Khurshid and D. Jackson, “Exploring the design of an intentional naming scheme with an automatic constraint analyzer,” in *ASE*, 2000, pp. 13–22.

References

- [137] S. Narain, “Network configuration management via model finding,” in *LISA ’05: Proceedings of the 19th conference on Large Installation System Administration Conference*. Berkeley, CA, USA: USENIX Association, 2005, pp. 15–15.
- [138] E. Torlak and D. Jackson, “Kodkod: A relational model finder,” 2007, pp. 632–647.
- [139] J. Khoury, C. Abdallah, and G. Heileman, “Towards formalizing network architectural descriptions,” University of New Mexico, Technical Report ECE-TR-08-03, February 2008, [online]: <http://hdl.handle.net/1928/3674>.
- [140] J. Shoch, “Inter-network naming, addressing, and routing,” in *IEEE COMPCON*. New York: IEEE, Fall 1978, pp. 72–79.
- [141] J. Saltzer, “On the naming and binding of network destinations,” RFC 1498.
- [142] M. Boguna and D. Krioukov, “Navigating ultrasmall worlds in ultrashort time,” *Physical Review Letters*, vol. 102, no. 5, p. 058701, 2009.
- [143] I. Abraham, A. Badola, D. Bickson, D. Malkhi, S. Maloo, and S. Ron, “Practical locality-awareness for large scale information sharing,” in *IPTPS ’05: In Proceeding of the 4th International Workshop on Peer-to-Peer Systems*, ser. Lecture Notes in Computer Science, vol. 3640. Springer, 2005, pp. 173–181.
- [144] D. Massey, L. Wang, B. Zhang, and L. Zhang, “A scalable routing system design for future internet,” in *ACM SIGCOMM workshop on IPv6 and the Future of the Internet*. New York, NY, USA: ACM Press, 2007.
- [145] G. Huston, “Bgp in 2008,” <http://www.potaroo.net/ispcol/2009-03/bgp2008.html>, March 2008.
- [146] R. Gummadi, R. Govindan, N. Kothari, B. Karp, Y.-J. Kim, and S. Shenker, “Reduced state routing in the internet,” *HotNets 04*, 2004.
- [147] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica, “Hlp: a next generation inter-domain routing protocol,” in *SIGCOMM ’05*. New York, NY, USA: ACM, 2005, pp. 13–24.
- [148] M. Caesar and J. Rexford, “Bgp routing policies in isp networks,” *IEEE Network*, vol. 19, no. 6, pp. 5–11, 2005.
- [149] L. Gao, “On inferring autonomous system relationships in the internet,” *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 733–745, 2001.

References

- [150] P. Mahadevan, D. V. Krioukov, M. Fomenkov, X. A. Dimitropoulos, K. C. Claffy, and A. Vahdat, “The internet as-level topology: three data sources and one definitive metric,” *Computer Communication Review*, vol. 36, no. 1, pp. 17–26, 2006.
- [151] M. OÖDell, “An alternate addressing architecture for ipv6,” IETF Draft. February 1997.
- [152] M. Arias, L. J. Cowen, K. A. Laing, R. Rajaraman, and O. Taka, “Compact routing with name independence,” in *ACM SPAA ’03*. New York, NY, USA: ACM, 2003, pp. 184–192.
- [153] M. Thorup and U. Zwick, “Compact routing schemes,” in *ACM SPAA ’01*. New York, NY, USA: ACM, 2001, pp. 1–10.
- [154] A. Brady and L. Cowen, “Compact routing on power-law graphs with additive stretch,” *ALENEX*, 2006.
- [155] P. Fraigniaud and C. Gavoille, “Routing in trees,” in *ICALP ’01: Proceedings of the 28th International Colloquium on Automata, Languages and Programming*. London, UK: Springer-Verlag, 2001, pp. 757–772.
- [156] A. Rowstron and P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems,” in *Proceedings of IFIP/ACM Middleware 2001*. Heidelberg, Germany: ACM, November 2001.
- [157] K. A. Laing, “Name-independent compact routing in trees,” *Inf. Process. Lett.*, vol. 103, no. 2, pp. 57–60, 2007.
- [158] ———, “Name-independent compact routing in trees,” Technical Report, Tufts University [online] http://www.cs.tufts.edu/tech_reports/reports/2003-2/report.pdf.
- [159] J. Khoury and C. Abdallah, “Identifier-based discovery mechanism design in Large-Scale networks,” in *International Workshop on the Network of the Future (with IEEE ICC’09)*, Dresden, Germany, 6 2009.
- [160] L. Blume, D. Easley, J. Kleinberg, and E. Tardos, “Trading networks with price-setting agents,” in *EC ’07: Proceedings of the 8th ACM conference on Electronic commerce*. New York, NY, USA: ACM, 2007, pp. 143–151.
- [161] J. Feigenbaum, V. Ramachandran, and M. Schapira, “Incentive-compatible interdomain routing,” in *EC ’06: Proceedings of the 7th ACM conference on Electronic commerce*. New York, NY, USA: ACM, 2006, pp. 130–139.

References

- [162] T. G. Griffin, F. B. Shepherd, and G. Wilfong, “The stable paths problem and interdomain routing,” *IEEE/ACM Trans. Netw.*, vol. 10, no. 2, pp. 232–243, 2002.
- [163] M. Afergan, “Using repeated games to design incentive-based routing systems,” *INFOCOM 2006*, pp. 1–13, April 2006.
- [164] J. Khoury and C. Abdallah, “Identifier-based discovery in large-scale networks an economic perspective,” in *Applications of Intelligent Control to Engineering Systems*. Springer, 2009, pp. 395–425.
- [165] W. Herrin, “What does a bgp route cost?” <http://bill.herrin.us/network/bgpctest.html>, 2008.
- [166] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker, “A bgp-based mechanism for lowest-cost routing,” *Distrib. Comput.*, vol. 18, no. 1, pp. 61–72, 2005.
- [167] E. Arcaute, A. Kirsch, R. Kumar, D. Liben-Nowell, and S. Vassilvitskii, “On threshold behavior in query incentive networks,” in *EC '07: Proceedings of the 8th ACM conference on Electronic commerce*. New York, NY, USA: ACM, 2007, pp. 66–74.
- [168] S. Shakkottai and R. Srikant, “Economics of network pricing with multiple ISPs,” *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 1233–1245, 2006.
- [169] A.-L. Barabasi, *Linked*. Perseus Publishing, 2002.
- [170] “lp_solve 5.5,” <http://lpsolve.sourceforge.net/5.5/>.
- [171] D. Garlan and B. Schmerl, “@vol: A tool for defining and planning architecture evolution,” in *2009 International Conference on Software Engineering*, 20–22 May 2009.