BOSTON UNIVERSITY

GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation

OVERLAY NETWORK CREATION AND MAINTENANCE WITH SELFISH USERS

by

GEORGIOS SMARAGDAKIS

Diploma, Technical University of Crete, 2002

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2009

Approved by

First Reader

Azer Bestavros, Ph.D.
Professor of Computer Science
Boston University

Second Reader

Nikolaos Laoutaris, Ph.D.
Researcher
Telefónica Research, Barcelona

Third Reader

John W. Byers, Ph.D.
Associate Professor of Computer Science
Boston University

Στην οικογένειά μου,

Θεμιστοκλή, Άννα και Εμμανουήλ.

Acknowledgments

In the last five years I have been privileged to work, study, and reside in a great academic environment that motivated me to challenge my limits. I would like to express my acknowledgments to colleagues and friends who encouraged and inspired my work during the graduate school years.

I am grateful to my advisor, Azer Bestavros, for his support and encouragement throughout my graduate studies. Under his supervision, I was provided full academic freedom to build my research agenda and was encouraged to think outside the box. Despite his extremely heavy schedule as the department chair, Azer was always available to discuss my research and academic development.

I would like to express my gratitude to my co-advisor, Nikolaos Laoutaris. Nikos taught me how to attack challenging networking problems by peeling off non-essential details to reveal the underlying fundamental computer science problem. He also introduced me to game theory, approximation algorithms, and operational research, which are the algorithmic foundations of my research in this dissertation.

My second co-advisor, John Byers, is one of the most effective teachers I have had. Taking coursework with him changed my outlook on computer networks and introduced me to new areas of research.

The faculty, students, and staff of the computer science department at Boston University provided a stimulating and personal environment. My special thanks go to the members of the WING group for their valuable feedback on my research. I would like to express my gratitude to my co-authors who contributed to my research efforts. I am also thankful to my office- and room-mate Vijay Erramilli, to Panagiotis Papapetrou, Niky Riga, Vassilis Athitsos, Michalis Potamias, and Irena Tsvetkova for being always there for me.

My research was financially supported by National Science Foundation, Boston University, Telefónica Research-Barcelona, and the European Commission.

I dedicate this dissertation to my father, my mother, and my brother. Words are not strong enough to express my feelings for them.

OVERLAY NETWORK CREATION AND MAINTENANCE WITH SELFISH USERS

(Order No.)

GEORGIOS SMARAGDAKIS

Boston University, Graduate School of Arts and Sciences, 2009

Major Professor: Azer Bestavros, Professor of Computer Science Department

ABSTRACT

Overlay networks have been used for adding and enhancing functionality to the end-users without requiring modifications in the Internet core mechanisms. Overlay networks have been used for a variety of popular applications including routing, file sharing, content distribution, and server deployment. Previous work has focused on devising practical neighbor selection heuristics under the assumption that users conform to a specific wiring protocol. This is not a valid assumption in highly decentralized systems like overlay networks. Overlay users may act selfishly and deviate from the default wiring protocols by utilizing knowledge they have about the network when selecting neighbors to improve the performance they receive from the overlay.

This thesis goes against the conventional thinking that overlay users conform to a specific protocol. The contributions of this thesis are threefold. It provides a systematic evaluation of the design space of selfish neighbor selection strategies in real overlays, evaluates the performance of overlay networks that consist of users that select their neighbors selfishly, and examines the implications of selfish neighbor and server selection to overlay protocol design and service provisioning respectively.

This thesis develops a game-theoretic framework that provides a unified approach to modeling Selfish Neighbor Selection (SNS) wiring procedures on behalf of selfish users. The model is general, and takes into consideration costs reflecting network latency and user preference profiles, the inherent directionality in overlay maintenance protocols, and

connectivity constraints imposed on the system designer. Within this framework the notion of user's "best response" wiring strategy is formalized as a k-median problem on asymmetric distance and is used to obtain overlay structures in which no node can re-wire to improve the performance it receives from the overlay. Evaluation results presented in this thesis indicate that selfish users can reap substantial performance benefits when connecting to overlay networks composed of non-selfish users. In addition, in overlays that are dominated by selfish users, the resulting stable wirings are optimized to such great extent that even non-selfish newcomers can extract near-optimal performance through naïve wiring strategies.

To capitalize on the performance advantages of optimal neighbor selection strategies and the emergent global wirings that result, this thesis presents EGOIST: an SNS-inspired overlay network creation and maintenance routing system. Through an extensive measurement study on the deployed prototype, results presented in this thesis show that EGOIST's neighbor selection primitives outperform existing heuristics on a variety of performance metrics, including delay, available bandwidth, and node utilization. Moreover, these results demonstrate that EGOIST is competitive with an optimal but unscalable full-mesh approach, remains highly effective under significant churn, is robust to cheating, and incurs minimal overheads.

This thesis also studies selfish neighbor selection strategies for swarming applications. The main focus is on n-way broadcast applications where each of n overlay user wants to push its own distinct file to all other destinations as well as download their respective data files. Results presented in this thesis demonstrate that the performance of our swarming protocol for n-way broadcast on top of overlays of selfish users is far superior than the performance on top of existing overlays.

In the context of service provisioning, this thesis examines the use of distributed approaches that enable a provider to determine the number and location of servers for optimal delivery of content or services to its selfish end-users. To leverage recent advances in virtualization technologies, this thesis develops and evaluates a distributed protocol to migrate servers based on end-users demand and only on local topological knowledge. Results under a range of network topologies and workloads suggest that the performance of the distributed deployment is comparable to that of the optimal but unscalable centralized deployment.

Contents

List of Tables

List of Figures

List of Abbreviations

API	....	Application Programming Interface
AS	....	Autonomous System (in the Internet)
BA	....	Barabási-Albert graph
BR	....	Best Response
CDF	....	Cumulative Distribution Function
CDN	....	Content Distribution Network
CPU	....	Central Processing Unit
DHT	....	Distributed Hashing Table
ER	....	Erdös-Rényi graph
GSH	....	Generic Service Host (in the Internet)
FIFO	....	First In First Out scheduling
ILP	....	Integer Linear Programming
IP	....	Internet Protocol
ISP	....	Internet Service Provider
MF	....	Maximum Flow
MRF	....	Most Replicated First scheduling
MST	....	Minimum Spanning Tree
NP	....	Non-deterministic Polynomial time
ON	....	Ordinary Node
P2P	....	Peer-to-Peer
RTT	....	Round Trip Time
SLA	....	Service Level Agreements (in the Internet)
SN	....	Super Node

SNS	....	Selfish Neighbor Selection
UFL	....	Uncapacitated Facility Location problem
UKM	....	Uncapacitated k-median problem
VM	....	Virtual Machine

Chapter 1

Introduction

Overlay networks are computer networks that are built on top of physical networks of interconnected routers and end-systems. A node in an overlay network is a process running at an end-system and has neighbors, to which it connects through logical links that in reality are multi-hop paths in the underlying physical network. For the rest of the thesis we will use the terms overlay node and user interchangeably. Overlay networks have been developed for adding and enhancing functionality to the end-users without requiring modifications in the Internet core mechanisms. Overlays have been used for a variety of popular applications including routing [101, 2], peer-to-peer file sharing [93], content distribution [27], data-center applications [60], and online multi-player games [10], among others.

A common goal of overlay designers has been the improvement of the average performance that an overlay user receives. Previous work has focused on devising practical neighbor selection heuristics under the assumption that users conform to a specific wiring protocol [101, 2, 70, 69, 119, 73, 45, 121, 102, 113, 30, 102, 108, 96, 43, 109, 24, 55, 19]. This is not a valid assumption in highly decentralized systems like overlay networks. Overlay users are typically governed by agents whose interest is not the optimization of the overlay's performance, but rather the maximization of their own benefit. Therefore, users may act selfishly by choosing to connect to their best neighbors. Overlay users are incented to deviate from the wiring protocol, utilize more information about the network, and re-wire in order to improve the performance they receive from the overlay. In overlay networks both the collection of topological information as well as re-wiring is easy. While much attention has been paid to the harmful downsides of selfish behavior in different settings [94, 100, 86], the impact of adopting Selfish Neighbor Selection (SNS) strategies in real overlay networks has been an open problem up to now [35].

1.1 Conceptual Contributions

In this thesis we go against the conventional thinking that overlay users conform to a specific wiring protocol, whose objective is the optimization of the overlay's performance. Our main focus is on the performance characteristics of overlays, consisting of users that are self-interested and select their neighbors selfishly. In the following paragraphs we comment on why this is a promising approach to study the performance of real overlays.

In highly decentralized overlays, where users belong to different administrative authorities, auditing or enforcing global wiring can be difficult or impossible. In this setting it is very difficult to identify non-compliant wiring behavior. Individual users who conform to the wiring protocol, may experience significant performance degradation. An architectural solution to protect individual users as well as the system from being exploited is to provide the best wiring strategy to all the users in the overlay. A system that provides such a wiring strategy to the users creates additional incentives to users not to deviate from the protocol. Analytical and experimental results obtained under this framework are more general as they relax the assumption that users will sacrifice their performance towards improving the overlay's performance.

Selfish wiring ability increases the awareness of end-users about possible misconfiguration or bugs of the wiring protocol. Wiring protocols that attempt to optimize the average performance in the overlay may lead to substantial performance degradation of individual users. The only way for an individual user to avoid such pathological cases is to react by deviating from the wiring protocol. Moreover, selfish wiring leads to scalable and distributed deployment of networks, where the monitoring cost and wiring decisions are outsourced to the end-users.

An overlay that offers the best wiring strategies to end-users and increases end-users awareness provides additional incentives for new users to join the overlay. We believe that the study of such systems has not received enough attention. In this thesis we first model what is the best wiring strategy for a user and then use this model to study the performance of different types of overlays, which are composed of selfish users. This is a more realistic approach to studying the performance characteristics of complex overlay systems. In many overlay systems, only the individual objective function can be derived, as the social objective function may be too complex [106].

1.2 Technical Contributions

In this Section we highlight the main technical contributions of the thesis. We demonstrate the implications and potential for adopting selfish wiring strategies in overlay network creation and maintenance.

We develop a game-theoretic framework that provides a unified approach to modeling selfish neighbor selection wiring procedures on behalf of selfish users. Our model is richer than previously proposed ones, attempting to derive selfish neighbor selection strategies in the physical layer [34, 23, 99, 29]. Our model is general enough to take into consideration costs, reflecting network latency, user preference profiles, the inherent directionality in overlay maintenance protocols, and connectivity constraints imposed on the system designer. Within this framework the notion of user's "best response" is formalized as a k-median problem on asymmetric distance. We use this formulation to quantify the performance gain of a selfish user when compared to this of a conformant user. Our evaluation shows that selfish users can reap substantial performance benefit, especially when connecting to overlays composed of naïve users.

We use the above-mentioned formulation in order to obtain overlays that are composed of selfish users. In overlays that are composed of selfish users, the resulting wirings are optimized to such an extend that even non-informed newcomers can extract near-optimal performance through naïve neighbor selection strategies. We also show the potential benefits that selfish neighbor selection strategy offers to overlays like real-time and file-searching systems.

To capitalize on the performance advantages of selfish neighbor selection strategies and the emergent global wirings that result, we present the design and evaluation of EGOIST. EGOIST is an SNS-inspired overlay network creation and maintenance routing system. In EGOIST, each user participates in a link-state protocol [92], where it reports its neighbors and the distance to them. Each overlay user constructs the full topology of the overlay and is informed about the wiring changes that take place in the overlay. Periodically, each overlay user monitors the distance to all the other users in the network and then connects to its best neighbors. Through an extensive measurement study on our deployed prototype over PlanetLab, our results show that EGOIST's neighbor selection primitives outperform existing heuristics on a variety of performance metrics, including delay, available bandwidth, and node utilization. Moreover, our results demonstrate that EGOIST is competitive with an optimal but unscalable full-mesh approach, remains highly effective under significant churn and is robust to cheating. EGOIST also incurs minimal measurement, computational, and re-wiring overheads. Furthermore, we present architectural decisions to make EGOIST scalable. We present the potential benefits that EGOIST offers to many applications, including multi-player peer-to-peer games, multi-path file transfers, and real-time traffic over IP. EGOIST has been released to the research community and can be accessed from the EGOIST project web site at `http://csr.bu.edu/sns/`.

In the context of file sharing, selfish neighbor selection has different characteristics. We study the implications of selfish neighbor selection strategies on swarming applications. For many swarming applications, including the popular BitTorrent [24], the evolving random topology is justifiable, given the scale of peer-to-peer file swapping networks. On the other hand, many high-performance file-sharing applications, are realizable only in small to medium scale networks. Therefore, we consider n-way broadcasting – a class of applications in which each one of the n overlay users must push a large file to all other peers, as well as pull the files pushed by these other peers. Examples of n-broadcasting include high-performance applications like distribution of large scientific data-sets, distribution of large-scale traffic log files for network-wide distributed intrusion or anomaly detection schemes [68], synchronization of distributed databases [9], and several other enterprise applications. We show that selfish neighbor selection can leverage the above scale constraint to construct optimized overlays that take into consideration the end-to-end characteristics of the network. Moreover, we show that the deployment of a single overlay to jointly optimize all the file swappings in parallel, in order to protect the uplink capacity of overlay users, is more appropriate. In this setting a selfish user strives to maximize the available bandwidth to the slowest destination. Our experimental results show that our swarming protocol that operates on top of overlays formed by selfish users delivers far superior performance than this on top of that of existing overlays. At the same time, selfish neighbor selection guarantees download synchronization. Finally, we show how to modify our swarming protocol to allow it to accommodate upload-selfish users.

In the context of service provisioning for Content Distribution Networks (CDNs) and service deployment we study the setting, in which end-users selfishly select how to connect to a server. We present the design and evaluation of a scalable and distributed protocol that enables a provider to dynamically determine the number and location of servers for optimal delivery of content or services to end-user. This protocol relies on the fact that users select their servers selfishly. Our distributed protocol migrates servers, based only on local topological knowledge and end-user demand, to leverage recent advances in virtualization technologies. The local topology is easily obtained through standard topology discovery protocols,[1] while end-user demand can be achieved through measuring locally the outgoing traffic at each server. We prove that our protocol converges to a stable deployment within a small number of migrations, while the size of the topology that is utilized regulates the trade-off between scalability and performance. Our experimental results under a range of network topologies and workloads suggest that the performance of our protocol is comparable to that of the optimal but unscalable centralized deployment. These results also show that the degradation of performance, due to imperfect redirection of users to migrated servers, is minimal.

---

[1] Skitter, `http://www.caida.org/tools/measurement/skitter` or DIMES, `http://www.netdimes.org/`

In this thesis we do not dwell on the negatives, but instead focus on the potential benefits from selfish neighbor selection. These include the obvious benefits to selfish users, and, more surprisingly, to the network as a whole. Indeed, we confirm that selfishness is not a problem, as much as inaction, indifference, or naïve reaction.

1.3 Thesis Organization

In Chapter 2 we introduce SNS and experimentally examine the performance characteristics of overlays, composed of selfish users. In Chapter 3 we present the design and evaluation of the EGOIST overlay routing system. Next, in Chapter 4 we present our swarming protocol for assisting n-broadcasting applications, based on selfish neighbor selection primitives. In Chapter 5 we present the design and evaluation of our distributed server migration protocol. Chapter 6 offers concluding remarks and outlines promising future research directions.

Chapter 2

Selfish Neighbor Selection

Neighbor selection is a key problem for a broad class of distributed services and applications that run atop large, amorphous overlay networks of autonomous nodes (we use the terms node, user and agent interchangeably). For example, in an overlay routing or a peer-to-peer (P2P) file sharing network, a new node must first select a relatively small number of direct neighbors before it can connect to the service.

In these systems, and in many others, it is clear that the impact of the neighbor selection strategy is significant, as evidenced by the emerging body of work exploring network creation games and characterizing the equilibria of these games. To date, however, the bulk of the work and main results in this area have centered on games where edges are undirected, access costs are based on hop-counts, and nodes have potentially unbounded degrees [34, 23, 99, 29]. While this existing body of work is extremely helpful for laying a theoretical foundation and for building intuition, it is not clear how or whether the guidance provided by this prior work generalizes to situations of practical interest, in which underlying assumptions in these prior studies are not satisfied. Another aspect not considered in previous work is the consideration of settings in which some or even most players do not play optimally – a setting which we believe to be typical. Interesting questions along these lines include an assessment of the advantage to a player from employing an optimizing strategy, when most other players do not, or more broadly, whether employing an optimizing strategy by a relatively small number of players could be enough to achieve global efficiencies.

In this Chapter, we formulate and answer such questions using a combination of modeling, analysis, and extensive simulations using synthetic and real datasets. Our starting point is the definition of a network creation game that is better suited for settings of P2P and overlay routing applications – settings that necessitate the relaxation and/or modification of some of the central modeling assumptions of prior work. In that regard, the central aspects of our model are:

(1) Bounded Degree: Most protocols used for implementing overlay routing or content sharing impose hard constraints on the maximum number of overlay neighbors. For example, in popular versions of BitTorrent a client may select up to 35 nodes from a neighbors' list provided by the *Tracker* of a particular torrent file [13].[1] In overlay routing systems [73], the number of immediate nodes has to be kept small so as to reduce the monitoring and reporting overhead imposed by the link-state routing protocol implemented at the overlay layer. Hard constraints on the number of first hop neighbors are also imposed in most peer-to-peer systems to address scalability issues, up-link and down-link fragmentation, and CPU consumption due to contention [114]. Motivated by these systems, we explicitly model such hard constraints on node degrees. Notice that in the prior studies cited above, node degrees were *implicitly bounded* (as opposed to *explicitly constrained*) by virtue of the trade-off between the additional cost of setting up more links and the decreased communication distance achieved through the addition of new links. We also note that some of these earlier network creation games were proposed in the context of physical communication networks. In such networks, the cost of acquiring a link is instrumental to the design and operation of a critical infrastructure. Such concerns do not apply in the case of overlay networks such as those we consider in this paper. Thus, we argue that models in which node degrees are outcomes of an underlying optimization process do not faithfully reflect the realities of systems and applications we consider.

[1] KaZaA and Pasturage include neighbor constraints at multiple levels: ordinary nodes (ON) may select up to 5 super nodes (SN) from a larger list for establishing initial negotiation and then maintain connection with only one of these; SNs may connect to at most 50 other SNs (from a typical population of SNs ranging between 25K and 40K [71]) and accept between 55 to 70 (or 100 to 160) children ONs (depending on their provisioning). New versions of Gnutella and Limewire involve a similar two-level architecture [110] with associated constraints. Similarly, DHT routing protocols like Chord [109] impose hard constraints on the number of first hop neighbors.

(2) Directed Edges: Another important consideration in the settings we envision for our work relates to link directionality. Prior models have generally assumed bi-directional (undirected) links. This is an acceptable assumption that fits naturally with the unbounded node degree assumption for models that target physical telecommunication networks because actual wire-line communication links are almost exclusively bidirectional. In overlay settings we consider, this assumption needs to be relaxed since the fact that node v forwards traffic or requests to node u does not mean that node u may also forward traffic or requests to v.

(3) Non-uniform preference vectors: In our model, we supply each node with a vector that captures its local preference for all other destinations. In overlay routing such preference may capture the percentage of locally generated traffic that a node routes to each destination, and then the aggregation of all preference vectors would amount to a origin/destination traffic matrix. In P2P overlays such preference may amount to speculations from the local node about the quality of, or interest in, the content held by other nodes. Other considerations may also include subjective criteria such as the perceived capacity of the node, its geographic location, or its availability profile.

(4) Representative distance functions: Although the initial models presented in this paper use assumptions made in several previous studies regarding equal unitary pair-wise distances for all one-hop overlay links, later in this paper, we relax this assumption by considering more representative distance models. As was done in [23], we consider synthetic distances obtained using topology generators. In addition, we consider more realistic settings in which topologies are obtained from real Internet settings – namely the PlanetLab overlay and actual AS-level maps – and in which associated distances are obtained through real measurements in these settings.

Our first technical contribution within this model is to express a node's "best response" wiring strategy as a k-median problem on asymmetric distance [5], and use this observation to obtain pure Nash equilibria through iterative best response walks via local search. We then experimentally investigate the properties of stable wirings on synthetic topologies as we vary two key properties of interest: (i) the *edge density* of the graph and (ii) the *non-uniformity of popularity* of nodes within the topology.

Our experimental results then consider neighbor selection problems motivated and driven by measurements of PlanetLab and the AS-level topology with a realistic access cost model. Here, we find that selfish nodes can reap substantial performance benefits when connecting to overlay networks composed of non-selfish nodes. On the other hand, in overlays that are dominated by selfish nodes, the resulting stable wirings are already so highly optimized that even non-selfish newcomers can extract near-optimal performance through heuristic wiring strategies. We conclude the Chapter by providing evidence in support of the potential benefits SNS may offer to overlay applications with different specifications and routing policies.

2.1 Background

Selfish neighbor selection for overlay networks was first mentioned by Feigenbaum and Shenker [35]. Fabrikant et al. [34] studied an unconstrained undirected version of the problem in which nodes can buy as many links as they want at a fixed per link price α. Chun et al. [23] studied experimental an extended version of the problem in which links prices need not be the same. Rocha et al. [99] is in the same spirit. In practice, however, important constraints on node degrees, not captured by these models, lead to richer games with substantively and fundamentally different outcomes.

Bindal et al. [13] propose a locality-enhanced version of BitTorrent in which only m out of the total k neighbors of a BitTorrent node are allowed to belong to a different ISP. Although the capacitated selection of neighbors is a central aspect of this work, their treatment is fundamentally different from ours in several regards: (i) there's no contention between selfish peers, (ii) the minimization objective is on inter-AS traffic therefore only two levels of communication distance are modeled, intra and inter-AS (we use finer topological information that includes exact inter-peer distances), and (iii) their "reachability" constraint amounts to asking for a similar level of data availability as the original one under the standard random neighbor selection mechanism of BitTorrent (we have fundamentally different reachability constraints, expressed as general preference functions over the potential overlay neighbors). Another recent work on neighbor selection is from Godfrey et al. [41]. It aims at selecting neighbors in a way that minimizes the effects of node churn (appearance of new nodes, graceful leaves and sudden malfunctions), but unlike our work, it does not focus on the impact of competing selfish nodes.

2.2 Definitions

Let $V = \{v_1, v_2, \ldots, v_n\}$ denote a set of nodes. Associated with node v_i is a preference vector $p_i = \{p_{i1}, p_{i2}, \ldots, p_{ii-1}, p_{ii+1}, \ldots, p_{in}\}$, where $p_{ij} \in [0, 1]$ denotes the preference of v_i for v_j, $i \neq j$: $\sum_{j=1, j\neq i}^{n} p_{ij} = 1$. Node v_i establishes a *wiring* $s_i = \{v_{i_1}, v_{i_2}, \ldots, v_{i_{k_i}}\}$ by creating links to k_i other nodes (we will use the terms link, wire, and edge interchangeably). Edges are *directed* and *weighted*, thus $e = (v_i, v_j)$ can only be crossed in the direction from v_i to v_j, and has cost d_{ij} ($d_{ji} \neq d_{ij}$ in the general case). Let $S = \{s_1, s_2, \ldots, s_n\}$ denote a *global wiring* between the nodes of V and let $d_S(v_i, v_j)$ denote the cost of a shortest directed path between v_i and v_j over this global wiring; $d_S(v_i, v_j) = M \gg n$ if there's no directed path connecting the two nodes.[2] For the overlay networks discussed here, the above definition of cost amounts to the incurred end-to-end delay when performing shortest-path routing along the overlay topology S, whose direct links have weights that capture the delay of crossing the underlying IP layer path that goes from the one end of the overlay link to the other. Let $C_i(S)$ denote the cost of v_i under the global wiring S, defined as the weighted (by preference) summation of its distances to all other nodes, *i.e.,* $C_i(S) = \sum_{j=1, j\neq i}^{n} p_{ij} \cdot d_S(v_i, v_j)$.

Definition 1 *(The SNS Game) The selfish neighbor selection game is defined by the tuple* $\langle V, \{S_i\}, \{C_i\}\rangle$*, where:*

- V *is the set of* n *players, which in this case are the nodes.*

[2] If the links are also annotated, then $M \gg \max_{i,j} d_{ij}$.

- $\{S_i\}$ *is the set of strategies available to the individual players.* S_i *is the set of strategies available to* v_i. *Strategies correspond to wirings and, thus, player* v_i *has* $\binom{n-1}{k_i}$ *possible strategies* $s_i \in S_i$.

- $\{C_i\}$ *is the set of cost functions for the individual players. The cost of player* v_i *under an outcome* S, *which in this case is a global wiring, is* $C_i(S)$.

The above definition amounts to a local connection [86], non-cooperative, non-zero sum, n-player game [89]. Let $S_{-i} = S - \{s_i\}$ denote the *residual wiring* obtained from S by taking away v_i's outgoing links.

Definition 2 *(Best Response) Given a residual wiring* S_{-i}, *a best response for node* v_i *is a wiring* $s_i \in S_i$ *such that* $C_i(S_{-i} + \{s_i\}) \leq C_i(S_{-i} + \{s'_i\})$, $\forall s'_i \neq s_i$.

Definition 3 *(Stable Wiring) A global wiring* S *is stable iff it is composed of individual wirings that are best responses.*

Therefore stable wirings are pure Nash equilibria of the SNS game, *i.e.,* they have the property that no node can re-wire unilaterally and reduce its cost. Fundamentally different is the work on Selfish Routing [94, 100], in which the network topology is part of the input to the game, and selfish source routing is the outcome. In a way, this is the inverse of our work, in which network-based (shortest-path) routing is an input of the game, and topology is the outcome. Selfish Routing is also based on source routing which is either not provided in most system implementations, or it is difficult to perform well in systems with high churn like peer-to-peer systems. Moreover, the proposed game is not a congestion game (thus it is not a potential game [86]), as the cost of the link is not dependent on the number of the nodes that use it.

2.3 Deriving Stable Wirings

In this section we start with a description of a general method for obtaining the best response of a node under general overlay link weights, which we then refine for the case that link weights are uniform. Next, we describe the iterative best response algorithm that we use for obtaining stable wirings. We conclude this section by presenting a simple lower bound for the social cost of a socially optimal solution — we later use this bound to evaluate the social cost of stable wirings.

2.3.1 The Best-Response of a node

A wiring for a node v_i can be defined using $n-1$ binary unknowns Y_l, $1 \leq l \leq n, l \neq i$: $Y_l = 1$ iff v_i wires to v_l, and 0 otherwise. Define also the binary unknowns X_{lj}: $X_{lj} = 1$ iff v_i has v_l as a first-hop neighbor on a shortest path to v_j. A best response for v_i under residual wiring S_{-i} can be obtained by solving the following Integer Linear Program (ILP): Minimize:

$$C_i(S_{-i}, X) = \sum_{j=1, j\neq i}^{n} p_{ij} \sum_{l=1, l\neq i}^{n} X_{lj} \cdot (d_{il} + d_{S_{-i}}(v_l, v_j)) \tag{2.1}$$

Subject to:

$$\sum_{l=1, l\neq i}^{n} X_{lj} = 1,\ \forall j \neq i \text{ and } \sum_{l=1, l\neq i}^{n} Y_l = k_i \text{ and } X_{lj} \leq Y_l, \forall l, j \neq i, \tag{2.2}$$

where d_{il} is the cost of a wire from v_i to v_l, and $d_{S_{-i}}(v_l, v_j)$ is the cost of a shortest path from v_l to v_j over the wiring S_{-i}.

2.3.2 Connections between SNS Game and Facility Location

When all the wires have the same unitary weight, then the distances d_S are essentially "hop counts", in which case there is an interesting relationship between finding a node's best-response wiring and solving a *k-median problem on asymmetric distance* [5, 83][3]. The latter is defined as follows:

Definition 4 *(Asymmetric k-median) Given a set of nodes V', weight's w_j $\forall v_j \in V'$, and an asymmetric distance function $d_{S'}$ (meaning that in general $d_{S'}(v,u) \neq d_{S'}(u,v)$), select up to k nodes to act as medians so as to minimize $C(V', k, w)$, defined as follows:*

$$C(V', k, w) = \sum_{\forall v_j \in V'} w_j \cdot d_{S'}(v_j, m(v_j)),$$

where $m(v_j)$ is the median that is closest to v_j.

[3]For the definition of the k-median problem see Section 5.1.

Proposition 1 *The best response of node v_i to S_{-i} under uniform link weights ($d_{ij} = 1, \forall i, j \in V$) can be obtained by solving an asymmetric k-median problem, in which:*

1. $V' = V - \{v_i\}$
2. $k = k_i$
3. $w_j = p_{ij}$, $v_j \in V'$
4. $d_{S'}(u, w) = d_{S_{-i}}(w, u)$, $u, w \in V'$,

Proof: Let s_i denote v_i's response to S_{-i}. The resulting cost will be:

$$
\begin{aligned}
C_i(S_{-i} + \{s_i\}) &= \sum_{v_j \in V'} p_{ij} d_{S_{-i}+\{s_i\}}(v_i, v_j) \\
&= \sum_{v_j \in V'} p_{ij} (d_{S_{-i}+\{s_i\}}(v_i, m(v_j)) + d_{S_{-i}+\{s_i\}}(m(v_j), v_j)) \\
&= \sum_{v_j \in V'} p_{ij} d_{S_{-i}+\{s_i\}}(v_i, m(v_j)) + \sum_{v_j \in V'} p_{ij} d_{S_{-i}+\{s_i\}}(m(v_j), v_j) \\
&= \sum_{v_j \in V'} p_{ij} + \sum_{v_j \in V'} p_{ij} d_{S_{-i}+\{s_i\}}(m(v_j), v_j) \\
&= \sum_{v_j \in V'} w_j + \sum_{v_j \in V'} w_j d_{S_{-i}}(m(v_j), v_j) \\
&= c + \sum_{v_j \in V'} w_j d_{S'}(v_j, m(v_j))
\end{aligned}
\tag{2.3}
$$

where c is a constant and $m(v_j)$ is v_i's next-hop neighbor on a shortest path to v_j under the global wiring $S_{-i} + \{s_i\}$. The transition from the third to the fourth line of Equation (2.3) relies on the fact that all distances to first hop neighbors are equal to 1 under hop-count distance. Obtaining the best response requires minimizing $C_i(S_{-i} + \{s_i\})$. Equation (2.3) shows that this is equivalent to minimizing $\sum_{v_j \in V'} w_j d_{S'}(v_j, m(v_j))$, which is exactly the objective function of the above mentioned asymmetric k-median problem. ■

Proposition 1 suggests that v_i's best response is to wire to the k_i medians of a distance function obtained by reversing the end-to-end distances of the residual wiring S_{-i}. Since even the metric version of k-median is NP-hard [83], so is its asymmetric version, and through Proposition 1 the best response of the SNS game as well. For the metric version of the k-median there exist several algorithms that provide constant-factor approximations of an exact solution [21, 54, 6, 49]. These guarantees do not hold for the asymmetric case. For the asymmetric k-median, Lin and Vitter [72] have given a bicriterion approximation that blows up the number of used medians by an $O(\log n)$ multiplicative factor to provide a cost that exceeds the optimal one by an additive factor. Archer [5] has shown that this is the best attainable approximation for this problem unless $NP \subseteq DTIME(n^{O(\log\log n)})$. Despite this negative result, simple heuristics like the p-swapping local search of Arya *et al.* [6] perform typically very well on the directed k-median (as also confirmed by our numerical results later in this paper).

2.3.3 Equilibrium Wirings through Iterative Best Response

Definition 5 *(Iterative best response) Given an initial global wiring $S^{(0)}$, start an iterative procedure where at the m-th iteration the nodes line up according to their identifiers (*i.e., *$v_1, v_2, \ldots$), and perform the following steps:*

1. *v_i computes its best response $s_i^{(m)}$ to $S_{-i}^{(m,i-1)}$, after v_{i-1} and before v_{i+1}*
2. *$S^{(m,i)} = S_{-i}^{(m,i-1)} + \{s_i^{(m)}\}$*

$S^{(m,i-1)}$ is the global wiring at iteration m (after v_{i-1}'s best response and prior to v_i's best response); $S_{-i}^{(m,i-1)}$ is the corresponding residual wiring with respect to v_i ($S_{-1}^{(m,0)} = S^{(m-1,n)} - \{s_1^{(m-1)}\}$ and $S_{-1}^{(1,0)} = S^{(0)} - \{s_1^{(0)}\}$). The iterative best response search stops and returns $S = S^{(\mathcal{M})}$ when at iteration $\mathcal{M}$: $s_i^{(\mathcal{M})} = s_i^{(\mathcal{M}-1)}$, $\forall v_i \in V$, *i.e.,* when no node can profit by re-wiring.

We use the iterative best response method to find stable wirings. In Section 2.4 where we present synthetic results based on hop-count distance we take advantage of the connection established through Proposition 2.3, and employ exact (ILP) and approximate (p-swapping local search [6]) solutions for the directed k-median in order to obtain best responses. In Section 2.5 we employ several real topologies in which distances are not hop-count and, therefore, employ the ILP formulation of Section 2.3.1 in order to obtain best responses.

2.3.4 A Lower Bound on the Cost of a Socially Optimal Wiring

Let S^* denote a socially optimal (SO) wiring, *i.e.,* a global wiring that minimizes the *social cost* $C(S) = \sum_{\forall v_i \in V} C_i(S)$. Let $S^{U,i}$ denote the *utopian* wiring for v_i, *i.e.,* the global wiring that minimizes $C_i(S)$ over all possible global wirings S (this should not be confused with a best response s_i that minimizes $C_i(S_{-i} + \{s_i\})$ granted a particular residual wiring S_{-i}). We can obtain a lower bound L on $C(S^*)$ by summing the costs of the individual utopian solutions, *i.e.,* $L = \sum_{\forall v_i \in V} C_i(S^{U,i})$. We describe $S^{U,i}$ for some interesting cases below. Before that, let o^j_{-i} denote the node with the jth largest out-degree, excluding v_i — let this degree be denoted $k(o^j_{-i})$.

Uniform node preference: When $p_i = p = \{1/n, \ldots, 1/n\}$, $\forall v_i \in V$, it is easy to see that $S^{U,i}$ is a directed tree with downward pointing edges, where: (1) v_i is the root; (2) v_i connects to nodes $o^1_{-i}, o^2_{-i}, \ldots, o^{k_i}_{-i}$ at level 1; (3) these nodes connect to the next $l_1 = \sum_{j=1}^{k_i} k(o^j_{-i})$ nodes with highest degrees $(o^{k_i+1}_{-i}, o^{k_i+2}_{-i}, \ldots, o^{k_i+l_1}_{-i})$ at level 2, and so on.

Uniform out-degree: When $k_i = k$, $\forall v_i \in V$, then $S^{U,i}$ is a directed regular k-ary tree with downward pointing edges, where (1) v_i is the root; (2) level l includes k^l nodes whose preference according to p_i ranks from $\left(\sum_{l'=1}^{l-1} k^{l'}\right) + 1$ to $\sum_{l'=1}^{l} k^{l'}$.

Uniform preference and out-degree: Combining the previous two cases results in a regular k-ary tree with l levels such that:

$$\sum_{l'=1}^{l} k^{l'} \geq n - 1 \Rightarrow k\frac{k^l - 1}{k - 1} \geq n - 1 \Rightarrow l \geq \log_k \left[\frac{(n-1)(k-1)}{k} + 1\right]$$

The resulting (common) cost for all $v_i \in V$ is:

$$\begin{aligned} C_i(S^{U,i}) &= \left(\sum_{l'=1}^{l} l k^{l'}\right) - l\left(\sum_{l'=1}^{l} k^{l'} - (n-1)\right) \\ &= \frac{l(k-1)(n(k-1)+1) - k(k^l - 1)}{(k-1)^2} \end{aligned} \tag{2.4}$$

In Section 2.4.1, we use the aforementioned bound to show numerically that the social cost of stable wirings is close to the social cost of socially optimal wirings.

2.4 Characterization of Stable Wirings

In this section we assume that establishing a direct (overlay) link between any two nodes incurs unit cost and, therefore, the cost between any pair of nodes equals the number of hops along any shortest, directed path that connects these nodes at the overlay layer. Our goal will be to characterize the structure of stable wirings with respect to two key scaling parameters of interest. The first parameter, $\alpha \in [0, 1]$, reflects the non-uniformity (skew) in the popularity of different destinations. We create such non-uniformity by adopting a generalized power-law profile for node popularity with skewness α, meaning that the popularity of the ith most popular node is $q_i = \Lambda/i^{\alpha}$, where $\Lambda = (\sum_{k=1}^{n} \frac{1}{k^{\alpha}})^{-1}$. We construct the preference vector p_i of node v_i by setting $p_{ij} = q_j/(1 - q_i), \forall v_j \in V : v_j \neq v_i$. High values of α mean that there are few highly-popular destinations among all the nodes, whereas low values mean that most destinations are equally popular.

The second parameter, $\beta \in [0, 1]$, determines the *link density* of a regular graph, which relates to the fanout (out-degree) of each node as follows: $k = \lceil n^{\beta} \rceil$.

For a given pair (α, β) we obtain the corresponding stable wiring by using the iterative best response method of Section 2.3.3, where the best response amounts to a solution of a directed k-median problem. Here, it is worthwhile to notice that different node orderings in the iterative best response search may lead to different stable wirings.[4] We have found that different stable wirings perform approximately the same and therefore it is of marginal value to look at the structure of different individual ones.

2.4.1 Social Cost of Stable Wirings

We first consider the quality of stable wirings compared to the utopian wirings described in the previous section. As can be seen for the examples depicted in Figure 2·1 (a) and (b), the gap between the stable solution and the Utopian solution is small, and this result holds across a wide range of settings for α and β, and for various values of n for which

[4]See [63] for a related discussion based on a different *object replication game*.

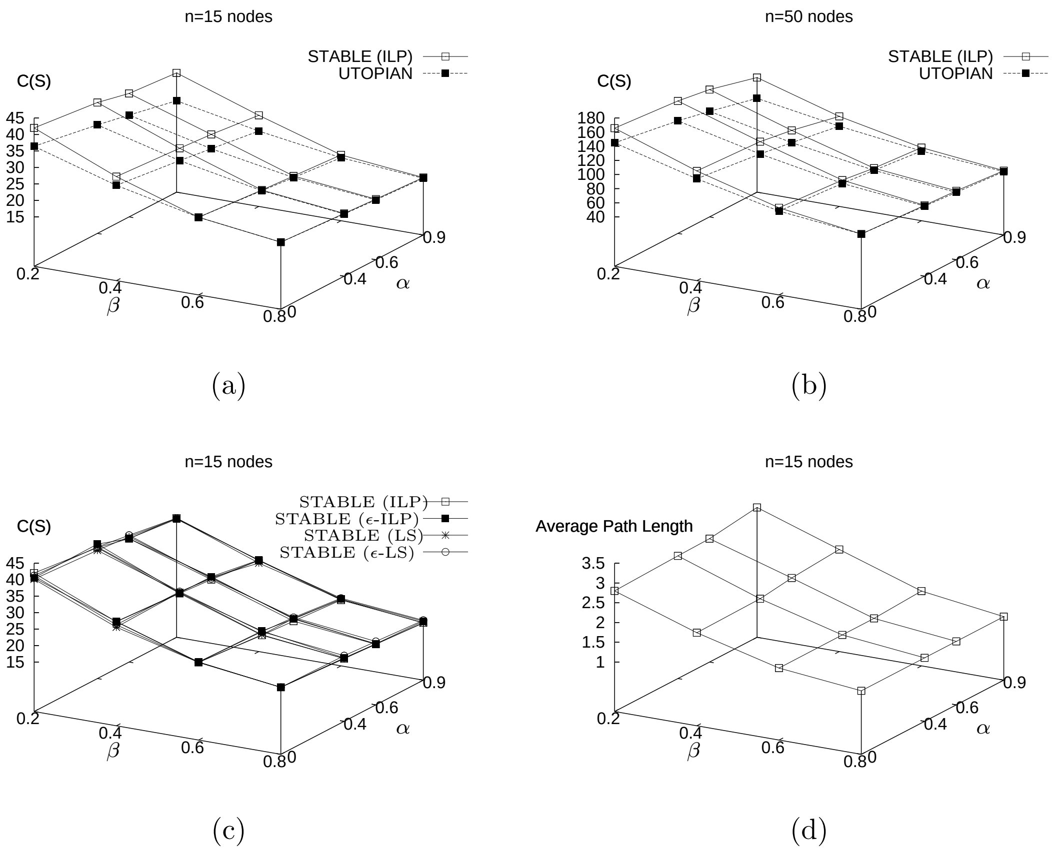


Figure 2·1: (a): Comparison of the social cost $C(S)$ of stable wirings to a lower bound of the cost of a socially optimal solution (the Utopian solution of Section 2.3.4) for $n = 15$. Stable wirings obtained using exact best responses based on an ILP formulation of the directed k-median problem of Section 2.3.2. (b): same as (a) with $n = 50$. (c): Comparison of the social cost $C(S)$ of stable wirings obtained by using exact (ILP) and approximate (LS) best response and corresponding $\epsilon = 5\%$ versions. (d): Average path length for the stable graph obtained by using exact (ILP) best response for $n = 15$.

simulation was tractable. In terms of absolute values, the social cost decreases with both the skew in popularity and link density. In particular, a highly-skewed popularity profile ensures that shorter paths to the most popular destinations are realized, whereas higher link densities reduces the average length of shortest paths, and thus the social cost as well.

Since computing exact best-response wirings is NP-hard, even under hop-count distance, it makes sense to study the performance of *approximate best responses* and corresponding *approximately stable wirings*. For this purpose, we used the Local Search (LS) heuristics described in [6] to solve the k-median problem, which yields the best-response wiring by virtue of Proposition 1. We also considered ϵ-stable versions of the problem in which nodes do not re-wire unless they can reduce their current cost by *at least* a multiplicative factor ϵ (we combined ϵ-stability with both exact (ILP) and approximate (LS) best responses). As evident from Figure 2·1 (c), we found that ϵ-stable wirings have similar social costs.[5]

To summarize, stable wirings have performance close to the socially optimal wirings. Moreover, approximate best-response wirings can be computed fast with LS and ϵ approximations.

In support to our results, we note that in a later work [59], it has been established analytically that provably existent stable wirings are guaranteed to perform approximately as well as socially optimal solutions under uniform node popularity. A similar conclusion is reached in the next section (albeit experimentally) for the case of non-uniform popularity.[6]

2.4.2 Topology of Stable Wirings

Next, we take a more in-depth look at the stable wirings that result for given values of α and β, as depicted in the set of graphs in Figure 2·2, where α varies from left to right and β varies from top to bottom.

[5]The results in Figure 2·1 (c) were obtained for $\epsilon = 0.05$, similar results (not shown) were obtained for $\epsilon \in [0.01, 0.1]$.

[6]In [59] was shown that for non-uniform popularity, a Nash equilibrium may not exist, or iterative best response walks may not lead to an equilibrium. Such observations were not made in our simulation study (we were able to find a stable graph starting from any initial graph we tried).

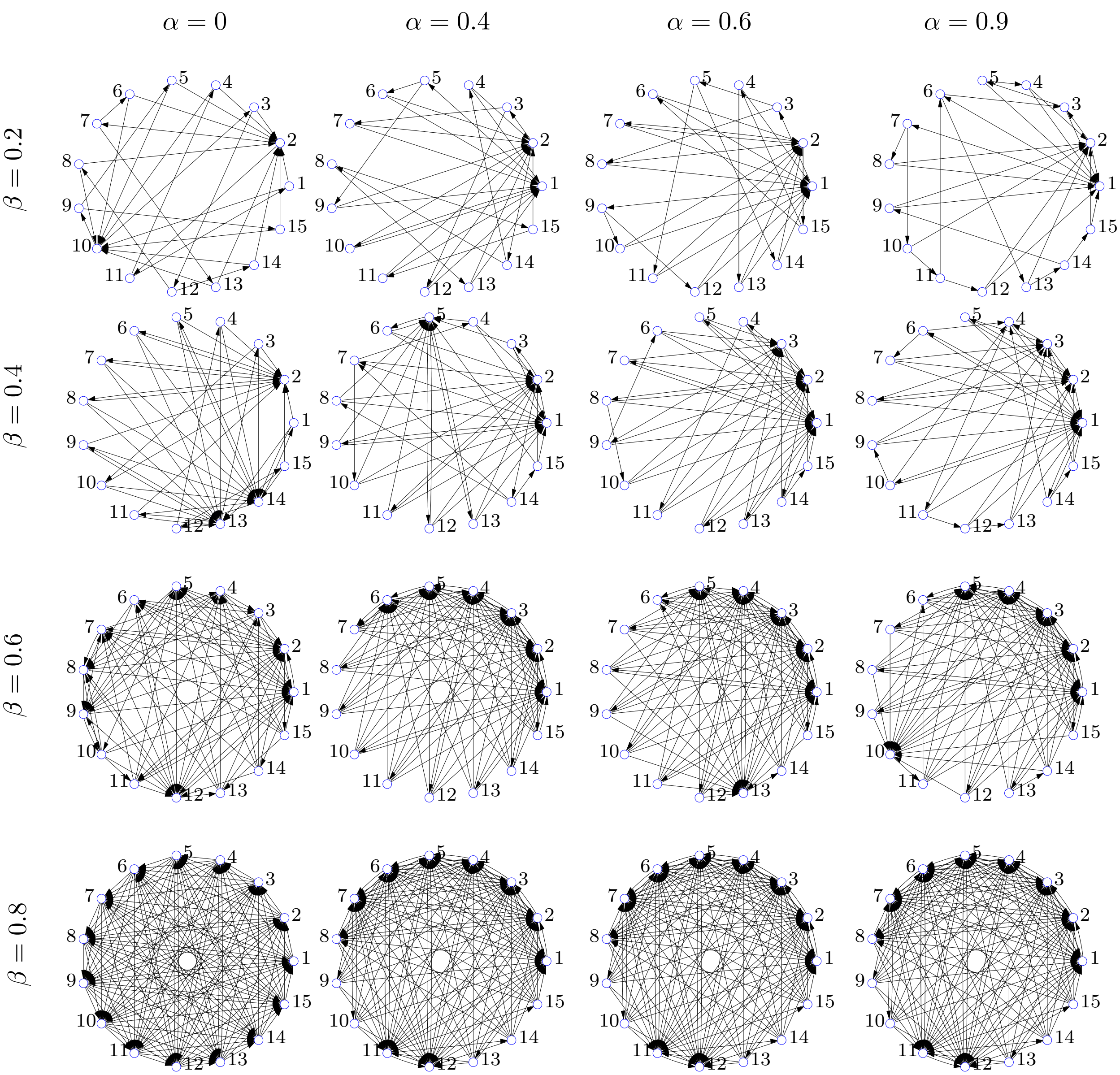


Figure 2·2: Stable wiring motifs for $n = 15$ and different values of α and β.

The first interesting finding is evident from an examination of the structures that emerge when $\alpha = 0$ (*i.e.,* under uniform popularity – the leftmost column in the figure). Despite the equal popularity of nodes, the resulting stable wirings do not exhibit uniform in-degree node distributions. In particular, some nodes tend to be more desirable for other nodes. Had the links been bidirectional, the emergence of such "hubs" could have been easily explained, by noting that they would be serving the purpose of providing short outgoing routes to many destinations. In our case, however, this cannot be the cause since these hubs have many *incoming* links, whereas their outgoing links are just as many as for the other nodes since all nodes have exactly k links, where k is controlled by the link density (β). Having made sure that these hubs did not emerge due to bias in tie-breaking during the computation of best responses, we attribute this "preferential attachment" phenomenon to the *quality rather than the quantity of outgoing links of hub nodes.* In particular, the hubs are nodes that (by coincidence) managed to position their k outgoing links in such a way that is beneficial to others as well (despite the fact that the wiring has been decided solely based on selfish criteria).[7]

Moving on to other larger values of α, where popularity is skewed, the hub creation process becomes a mix of the aforementioned phenomenon and the inherent preference for popular nodes. Nodes that are globally popular are natural candidates for becoming hubs. Even with relatively low skew ($\alpha = 0.4$), the most popular nodes are becoming hubs (node with id=1 is the most popular and that with id=n is the least popular). We see this trend consistently for all values of β, as is to be expected. But interestingly, as β increases further, it is not simply a contiguous sequence of the most popular nodes that end up becoming hubs! For example, in the $\alpha = 0.6, \beta = 0.6$ case, several nodes in the "tail" end of the popularity distribution end up becoming hubs as well, facilitating relay shortcuts as in the uniform popularity case.

We also find that the average path length slowly increases with α for a given β (see

[7]It is also worth mentioning that all the stable graphs (for the same value of k) we found are isomorphic having the structure that was proposed in [59].

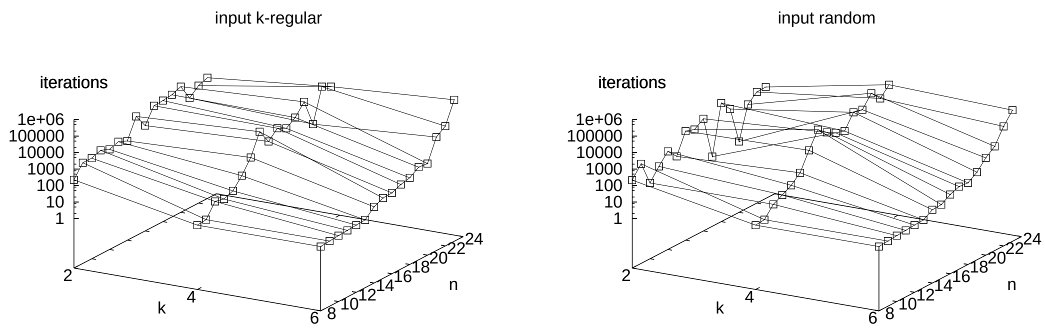


Figure 2·3: Convergence time starting from regular and random initial graphs.

Figure 2·1 (d)). This is to be expected since nodes prefer to be closer to the most popular nodes, and thus place less importance on the distance to much less popular nodes. Although this reduces the newcomers' access costs, it increases some shortest paths, and the diameter.

On a computational note, we observed that the uniform case required an exponential number of iterations for convergence. In the first plot of Figure 2·3, we start round-robin best-response walk from a regular (n, k)-wiring with offsets $[1 : k]$. All our experiments converge to a stable wiring. We plot the lengths of walks for all (n, k)-pairs. In the second plot of Figure 2·3, we repeat the same experiment starting from a wiring constructed as follows: Starting from a simple directed Hamiltonian cycle, we add to every vertex k-1 random out-going links. Both experiments demonstrate lengthy and possible exponential convergence. Moreover, the "random" wiring experiment shows large variance in the length of convergence, especially for sparse wirings. On the other hand, in the presence of non-uniform power-law profile with skewness α, a stable graph was found within a small number of iterations.

2.4.3 Contsraining the In-degree: A Doubly Constrained Overlay

We next examine the effects of constraining the maximum in-degree of nodes so that they never have more than ν incoming links, while maintaining also the constraint on the out-degree. We can enforce this constraint by including in the definition of $C_i(S)$ a large

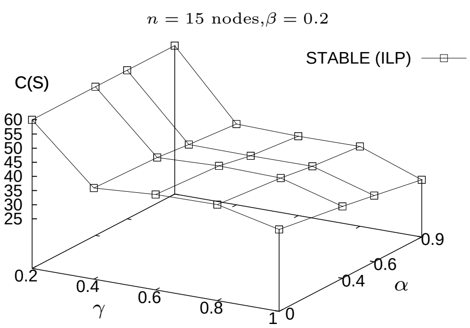


Figure 2·4: The social cost $C(S)$ of doubly capacitated stable wirings for $\beta = 0.2$ and different α, γ.

penalty for connecting to nodes that have more than $\nu - 1$ incoming links. We can define a scaling factor γ for the in-degree as done previously with β for the out-degree.

In Figure 2·4, we fix the out-degree scaling parameter to $\beta = 0.2$, and present the social cost for different values of the in-degree scaling parameter γ. Low values of γ increase the social cost under skewed popularity profiles, as in these cases, the highly-popular nodes quickly reach their maximum in-degree and thus, many nodes have to reach them indirectly through multi-hop paths. Note that without in-degree constraints most nodes would access them in a single hop by establishing a direct overlay link to them. When γ is low, *e.g.*, $\gamma = 0.2$, the resulting graph looks much like a ν-regular graph. With large values of γ, *i.e.*, γ approaching 1, the in-degree constraints become too loose and, thus, the corresponding stable graphs become similar to their unconstrained counterparts.

2.5 Overlay Neighbor Selection: Best Response vs. k-Random, k-Regular, and k-Closest

In this section we take a closer look at the performance benefits from employing best-response wiring instead of simpler wiring strategies. We also depart from the simplistic unit-distance model for the cost of direct links and instead use more realistic cost models on synthetic and measured topologies. Corresponding stable wirings are obtained by using the ILP model for a node's best response under general distances as detailed in Section 2.3.1.

2.5.1 Description and Design Methodology

In Section 2.2, we defined the best response strategy for a node entering a given network. Now, we consider two other natural alternatives. Let d_{ij}^X denote the cost associated with creating a direct overlay link between nodes v_i and v_j under a model X for end-to-end IP layer distances. We say that a "newcomer" node v_i employs a *k-Closest* wiring strategy under the model X when it establishes a wiring s_i such that $d_{ij}^X \leq d_{ij'}^X$ for all $v_j \in s_i, v_{j'} \notin s_i$. We say that a newcomer node v_i employs a *k-Random* wiring strategy when it chooses a wiring s_i uniformly at random from the space of all valid wirings of cardinality k_i. A newcomer node v_i that employs a *k-Regular* wiring strategy if it follows a pre-defined wiring pattern, based on node identifiers, like every other node in the network.

To substantiate the benefits of best response, we consider the initial graph awaiting a "newcomer" upon its arrival. We assume that this initial graph has resulted from having its constituent nodes apply a specific wiring strategy.[8] We refer to an instance of an n node graph for which each of the n nodes employed a k-Closest strategy as a *k-Closest* graph, and attribute similar meanings to a *k-Random* graph, a *k-Regular* graph and a *Best-Response (BR) graph*.

2.5.2 Description of the Datasets

In this section we describe the IP-layer end-to-end distance models X from which we obtain the d_{ij}^X's that are used as weights for direct overlay links between nodes v_i and v_j.[9] The following three datasets are used:

BRITE: The first dataset is synthetically generated from the BRITE topology generator [81] following a Barabási-Albert [8] model with $N = 1000$ nodes and incremental growth parameter $\mu = 2$. The nodes were placed on the plane according to a heavy tail model that creates high density clusters. Based on the observation that the delay between

[8] To guarantee connectivity, nodes that participate in a k-Random or a k-Closest graph, donate one link in order to create a ring. We note that a ring is a feature common to many other overlays, such as the Chord DHT [109] that is abstracted by the k-Regular graph.

[9] Overlay nodes that do not have a direct link communicate through a shortest-path on the overlay topology.

two nodes in high speed networks is highly correlated to their physical distance [122], we assigned weights on the links at the physical layer by calculating the Euclidean distance between their two end nodes.

PlanetLab: PlanetLab is an overlay testbed network of approximately 700 nodes in more than 300 academic, industrial, and government sites around the world. We used a publicly available dataset[10] containing delays obtained using *pings* between all pairs of PlanetLab sites (inter-site delays are more representative than inter-node delays for overlay applications).

AS-level map: As a third dataset, we use the relation-based AS topology map of the Internet from December 2001 measurements[11]. This map was constructed by using the measurement methodology described in [112]. The dataset includes two kinds of relationships between ASes: (1) customer-provider: The customer is typically a smaller AS that pays a larger AS for access to the rest of the Internet. The provider may, in turn, be a customer of an even larger AS. A customer-provider relationship is modeled using a directed link from the provider to the customer. (2) Peer-Peer: Peer ASes are typically of comparable size and have mutual agreements for carrying each other's traffic. Peer-peer relationships are modeled using undirected links. Overall the AS-level map includes 12779 unique ASes, of which 1076 are peers (joined by at least one peer-peer link), and the remaining 11703 are customers. These ASes are connected through 26387 directed and 1336 undirected links. We choose to present results based on the largest connected component of the dataset, which we found to include a substantial part of the total AS topology at the peer level: 497 peer ASes connected with 1012 links (we verified that this component contains all the top-20 larger peer ASes reported in [112]). The ASes that participate in this graph are responsible for routing the majority of the Internet traffic. We measured the hop-count distance between pairs of overlay nodes and used it as weight for a direct link between these two nodes at the overlay layer. To model the characteristics of IP routing

[10] `http://ping.ececs.uc.edu/ping`, accessed on July 10, 2006.
[11] `http://www.cc.gatech.edu/~mihail/ASdata.html`

(unique path), we broke ties by assigning each edge i a weight $1+\epsilon_i$ where ϵ_i is a zero-mean random noise as suggested in [58].

2.5.3 Comparison of Different Graphs

Using as input the weighted graphs from our three datasets, we obtained the social costs resulting from applying the various wiring strategies under consideration, for different values of β. The Best-Response (BR) graph (resulting from having all nodes apply the best-response wiring strategy) was by far the most optimized wiring, thus providing a lower-bound for the simpler k-Random and k-Closest strategies. Table 2.1 summarizes our results by providing the ratios of the social costs of the simple wiring strategies (k-Random, k-Closest, k-Regular) to that of the BR wiring. These results suggest that the premium provided by BR is highest for lower link densities (*i.e.,* when β is small). This is an intuitive result since in denser graphs, there is less of an opportunity for optimization.

On a technical note, none of the k-Random, k-Closest or k-Regular graphs we created was stable. On a computational note, all the stable graphs were found within a small number of iterations. Moreover, we observed that, for all the input datasets, and the given node selection process, the in-degree distribution of stable graphs was quite uniform, thus the load to relay traffic is expected to be quite balanced.

The results in this section give us a baseline for the efficiency of the wirings that result from the adoption by all nodes in the graph of the same strategy (be it k-Random, k-Closest, k-Regular or BR). This sets up the stage for our next set of questions: Given such an initial wiring, what is the marginal utility to a newcomer from executing each one of the three wiring strategies under consideration?

2.5.4 The Value of Best Response

Given an initial wiring created (as described above) by having n overlay nodes follow one of our three wiring strategies, we quantify the benefit to a "newcomer" (*i.e.,* the $n+1$'st node) from choosing its neighbors using one of the three neighbor selection strategies. Twelve

	$\beta = 0.1$			$\beta = 0.2$			$\beta = 0.4$			$\beta = 0.6$			$\beta = 0.8$		
	k-Random/BR	k-Regular/BR	k-Closest/BR	k-Random/BR	k-Regular/BR	k-Closest/BR	k-Random/BR	k-Regular/BR	k-Closest/BR	k-Random/BR	k-Regular/BR	k-Closest/BR	k-Random/BR	k-Regular/BR	k-Closest/BR
BRITE	1.44	3.61	1.53	1.52	2.31	1.84	1.38	1.50	2.07	1.28	1.11	1.46	1.09	1.03	1.16
PlanetLab	2.23	3.84	1.48	1.75	2.74	1.23	1.37	2.10	1.13	1.09	1.41	1.16	1.04	1.18	1.06
AS-level	2.04	4.78	1.90	1.83	2.86	1.61	1.58	2.37	1.39	1.24	1.10	1.23	1.12	1.12	1.16

Table 2.1: Social cost ratios between simple wiring strategies (k-Random, k-Regular, k-Closest) and Best Response.

possibilities exist for applying strategy S1 over a wiring obtained using S2, where S1 and S2 could be k-Random, k-Closest, k-Regular or BR. We use $c(w|G(n))$ to denote the cost of a newcomer using wiring strategy w on a pre-existing graph G of n nodes, or simply $c(w)$ when the graph G is understood. For example, $c(k\text{-Random} — k\text{-Closest})$ denotes the cost of a newcomer using the k-Random wiring strategy to connect to a graph of n nodes, each of which employed the k-Closest wiring strategy to construct the initial graph (to which the newcomer will connect).

In the results presented below, we set $n = 50$ and evaluate the performance for 200 newcomers on the BRITE and AS dataset and 100 newcomers for the PlanetLab dataset (which is smaller). Our main results are shown in Figure 2·5, where each column corresponds to an underlying graph model, and each row corresponds to a strategy employed by the n newcomers. Within each plot, we vary the link density β along the x-axis, and plot the cost ratio of the $n+1$'st arrival for a given strategy versus the cost of the $n+1$'st arrival if it were to use BR.

Connecting to a k-Random Graph: The plots in the top row of Figure 2·5 show the case in which the first n arrivals use k-Random, and thus the underlying graph is poorly optimized.

With such an initial graph, the k-Random wiring is a poor choice for the $(n+1)$st node, as it could lead to significantly higher costs (anywhere from 30% for the BRITE and AS datasets to 60% for the PlanetLab dataset) when compared to using BR. This performance gap closes, as one would expect, when β (and therefore k) becomes large. In fact this trend holds in all cases because finding a closer approximation to BR is easier when each node has more links — and therefore ample opportunity to make good connections, even when using simple strategies. The performance of k-Regular wiring is similar to the k-Random one, as the identifiers are randomly assigned.

Using the k-Closest wiring, on the other hand, turns out to be a very reasonable choice, as it achieves a cost comparable to that achieved by BR (typically within 10% with small exceptions for the BRITE and PlanetLab datasets under low link densities). This finding suggests that in poorly optimized random graphs, simply connecting to your nearby neighbors (at low cost), is a good rule of thumb, especially when edge density is high.

Connecting to a k-Regular Graph: The plots in the second row of Figure 2·5 show the case in which the first n arrivals use k-Regular, and thus the underlying graph is a structured one, where each node follows the same wiring pattern. Here we see again that a BR wirings pays off. The performance of k-Closest and k-Random improve as the graph becomes denser. k-Regular turns out to be a the worst choice[12] because structured graphs seem to eliminate the number of shortcuts.

Connecting to a k-Closest Graph: The plots in the third row of Figure 2·5 show the case in which the first n arrivals use k-Closest, and thus the underlying graph consists mostly of local edges with few shortcuts. Here we see that it is considerably more important for newcomers to behave strategically. For example, on the BRITE topology, naïvely using k-Closest is a poor choice that perpetuates the lack of shortcuts in the underlying graph to the point that even using k-Random or k-Closest turns out to be a better choice! In the other topologies, k-Closest, k-Random, and k-Regular are comparable, and the improvement in quality relative to BR as β increases is much more modest.

One conclusion from the results we obtained above for connection strategies to the k-Random, k-Regular (as we will elaborate next) and k-Closest graphs is of particular importance for P2P applications. In a P2P network, nodes pick neighbors randomly from the list provided to them by a bootstrap server (*i.e.,* the initial graph to which a newcomer would connect is a k-Random or a k-Regular graph). Under such circumstances, *it pays to "cheat"*, by pinging the possible neighbors and connecting to the k-Closest ones. However, if the constituent nodes in the initial graph also cheat, (*i.e.,* the initial graph to which a newcomer would connect is a k-Closest graph), then *it does not pay to cheat; it may even cost!*.

Connecting to a Stable Graph: Finally, the plots in the bottom row of Figure 2·5

[12] Note also that the range on values in the y-axis that represent the newcomer's cost ratio, is higher than before.

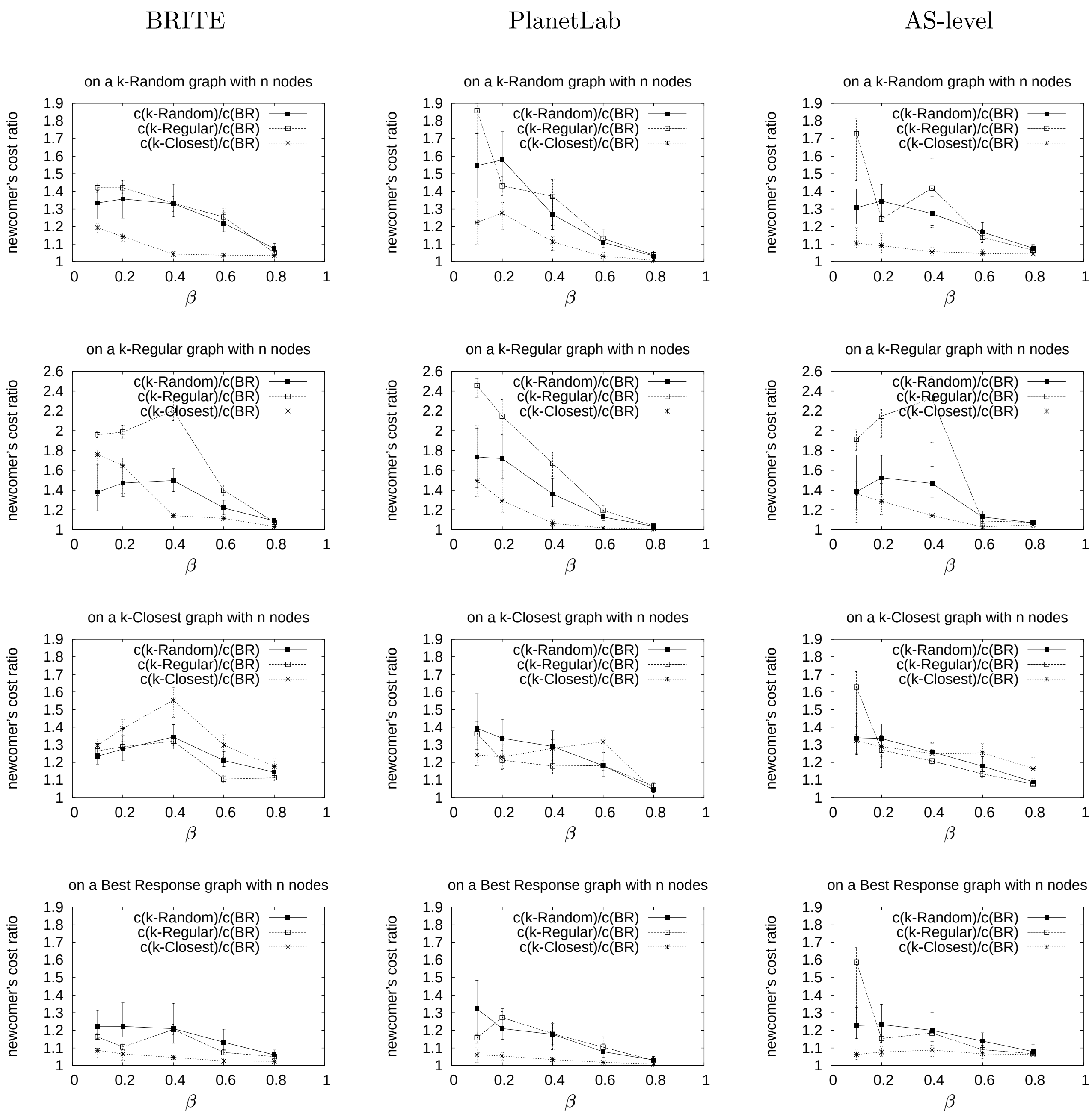


Figure 2·5: The cost ratio between simple wiring (k-Random, k-Regular or k-Closest) and BR wiring for a newcomer node that connects to a pre-existing network of n nodes that was wired using k-Random, k-Regular, k-Closest, or BR. We present the 25-, 50-, 75-quartiles for the aforementioned ratios using three different data sets (BRITE, PlanetLab, AS-level map) for obtaining the costs of establishing direct links.

show the case in which the first n arrivals use BR, and thus the underlying graph ends up being highly optimized, prior to the arrival of newcomers. In this case, the graph is so much optimized for the newcomer that any reasonable strategy might well have good performance. Surprisingly, while the k-Closest strategy does indeed perform well for the newcomer across the three topologies, the alternative strategies of k-Random and k-Regular do not. This seemingly odd result could be explained by noting that given the very low overall costs between nodes in the optimized initial graph, the cost to the newcomer from selecting its own neighbors (as opposed to the cost of reaching all nodes in the graph) could not be ignored. A poor choice of neighbors could backfire.[13]

General Observation: In conclusion, we find common trends across the three topologies with respect to strategic neighbor selection behavior. At the two extremes where the other players are playing completely at random or completely selfishly (top and bottom rows, respectively), the underlying graphs are either too poorly constructed, or too well constructed, for an uninformed newcomer to be at a significant disadvantage. In either of these two situations, the naïve strategy of k-Closest is generally competitive to BR, especially under stable graphs. Picking links at random in these situations however, is unlikely to work well, unless the graph is already dense (large β).

But in the middle regime, in which all the other players adopt k-Closest the newcomer must be much more careful. Here, there is much to be gained by the optimal shortcuts selected in BR, which neither k-Closest nor k-Random typically selects. Our experimental results suggest that k-Closest is one of the worse the possible strategies considered for the newcomer to adopt in this situation. Strikingly, our results advocate that the k-Regular is actually the worst of the possible strategies considered for the newcomer. Constructed overlays seem to reduce to the minimum the number of shortcuts.

[13] Recall that the topologies we considered in this section feature non-unit link costs, and as such, selecting neighbors at random could put the newcomer at a disadvantage, especially if the initial graph was optimized, since the relative penalty from a bad random selection of neighbors would be high.

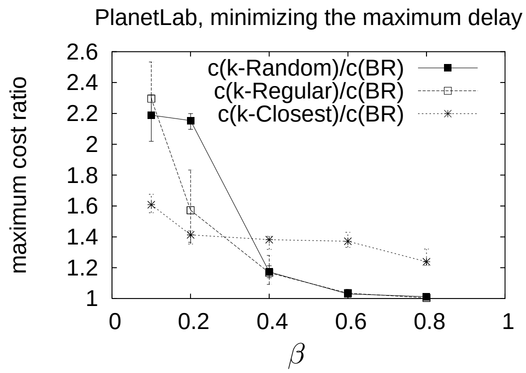


Figure 2·6: The cost ratio between simple wiring (k-Random, k-Regular, or k-Closest) and Min-Max BR wiring. We present the 25-, 50-, 75-quartiles for the aforementioned ratios in the PlanetLab dataset.

2.6 Minimizing the Maximum Delay

In many applications, *e.g.,* real-time streaming, voice conference, a selfish node would strive to minimize the maximum delay to all the other nodes in the overlay in order to receive the best possible quality of the service. The "best response" of the node, henceforth called Min-Max Best Response, is the following

Definition 6 *(Min-Max Best Response) Given a residual wiring S_{-i}, a best response for node v_i is a wiring $s_i \in S_i$ such that $M_i(S_{-i} + \{s_i\}) \leq M_i(S_{-i} + \{s_i'\})$, $\forall s_i' \neq s_i$, where $M_i = \max_j d_S(v_i, v_j)$.*

In Table 2.2 we plot the maximum delay ($\max_i M_i$) ratios between naïve and myopic wiring strategies, namely k-Random, k-Closest, k-Regular, and Min-Max Best Response[14] in the PlanetLab dataset. The performance of Min-Max Best-Response wiring is significantly better than that of naïve or myopic wiring strategies, even in dense graphs (high value of k). Focusing on the quality of service that an individual overlay user receives, we plot the maximum delay ratio between the aforementioned naïve and myopic wirings and Min-Max Best Resposne, along with the 25-, 50-, 75-quartiles in Figure 2·7. The performance of Min-Max Best Response is significantly better than that of naïve or myopic

[14]Stable wirings were derived with local search walks.

	$\beta = 0.1$			$\beta = 0.2$			$\beta = 0.4$			$\beta = 0.6$			$\beta = 0.8$		
	k-Random/BR	k-Regular/BR	k-Closest/BR	k-Random/BR	k-Regular/BR	k-Closest/BR	k-Random/BR	k-Regular/BR	k-Closest/BR	k-Random/BR	k-Regular/BR	k-Closest/BR	k-Random/BR	k-Regular/BR	k-Closest/BR
PlanetLab	2.89	3.09	1.91	2.37	1.79	1.60	1.64	1.82	1.90	1.84	1.86	1.88	1.94	1.94	1.96

Table 2.2: Maximum delay ratios between simple wiring strategies (k-Random, k-Regular, k-Closest) and Min-Max Best Response.

Algorithm 1 s_i=variable degree Best Response(S_{-i},M_T)

If $M_i > M_T$
    Find $s_i \in S_i$ such that $M_i(S_{-i} + \{s_i\}) \leq M_i(S_{-i} + \{s'_i\})$, $\forall s'_i \neq s_i$, $|s_i| = |s'_i| = k$. [1-swap]
    If $M_i \leq M_T$
        Return s_i;
    If $M_i > M_T$
        Find $s_i \in S_i$ such that $M_i(S_{-i} + \{s_i\}) \leq M_i(S_{-i} + \{s'_i\})$, $\forall s'_i \neq s_i$, $|s_i| = |s'_i| = k + 1$. [1-add]
        Return s_i;
If $M_i \leq M_T$
    Find $s_i \in S_i$ such that $M_i(S_{-i} + \{s_i\}) \leq M_i(S_{-i} + \{s'_i\})$, $\forall s'_i \neq s_i$, $|s_i| = |s'_i| = k - 1$. [1-drop]
    If $M_i \leq M_T$
        Return s_i;
    If $M_i > M_T$
        Find $s_i \in S_i$ such that $M_i(S_{-i} + \{s_i\}) \leq M_i(S_{-i} + \{s'_i\})$, $\forall s'_i \neq s_i$, $|s_i| = |s'_i| = k$. [1-swap]
        Return s_i;

wirings for a large range of out-degree values. Furthermore, none of the naïve or myopic wiring strategies is consistently better than another one.

2.7 Overlay Neighbor Selection with variable out-degree

In many applications, like the ones mentioned in the previous section, hard quality-of-service requirements must be satisfied. The estimation of the minimum out-degree that is needed to satisfy the application requirements is hard to be estimated a-priori by the system designer. Moreover, application requirements may change over time, thus an online and distributed estimation of the minimum number of connections per user is more appropriate. In this Section, we examine the performance of best-response wirings, where the out-degree of a node is not constant.

Let us consider a real-time application where the user satisfaction degrades rapidly when the maximum delay of one node to any other in the network is higher than an upper threshold of M_T time units. A selfish node would strive to satisfy the application requirements while keeping its out-degree as small as possible. This can be achieved by a local search heuristic, described in Algorithm 1, where each node can swap or incrementally[15] add or drop out-going connections.

[15] Multiple establishments or drops of connections may lead to faster convergence to a connectivity that satisfies the application requirements, but it might be unfair for a node [63], thus a selfish node would like to adapt its out-degree in an incremental fashion.

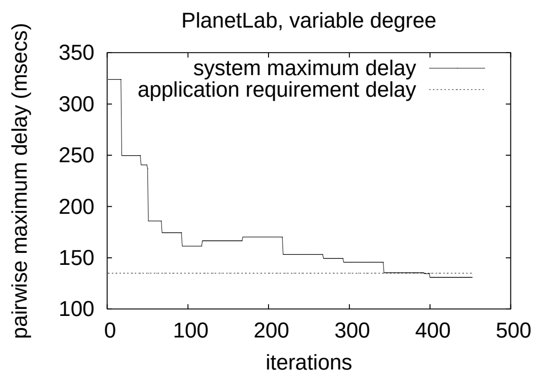


Figure 2·7: The maximum system pairwise delay per re-wire, under variable degree Best Response, in the PlanetLab dataset. The application requirement maximum delay is denoted with the dashed horizontal line.

To evaluate the performance of variable degree Best-Response wiring, we estimated the M_T for the stable graph where each node has out-degree $k = 5$ in the PlanetLab dataset, and use it as the application requirement for the same PlanetLab nodes where the initial out-degree was set $k = 2$. In Figure 2·7, we show that the application requirement is satisfied by all nodes within a small number of individual wirings. The evolving stable graph is still sparse. Out of the $n = 50$ nodes, 35 nodes have out-degree 2, 12 nodes have out-degree 3, one node has out-degree 4, and only two nodes have out-degree 5. This counts up to only 120 links compared to 250 established in the case where the out-degree is uniform and $k = 5$.

2.8 Overlay Neighbor Selection under scoped-flooding

In this Section we shift our attention to potential benefits employing BR neighbor selection strategy in applications other than shortest path routing. To that end we investigate the case of unstructured P2P file sharing. In such networks the outgoing traffic, the search queries,[16] target objects wherever these may lie, instead of specific nodes, as assumed in the original formulation of the BR. Still the initial formulation of BR might be helpful. Consider a P2P file sharing system in which nodes maintain a figure of merit for each other

[16]Once the object is mapped onto a node than a direct connection is established.

node based on direct or third-party experience.[17] The merit value could, for example, indicate quality of content, correlation of interests, or the capacity or reliability of the node. Then, even if queries are for objects and not for nodes, it still is beneficial to have the queries reach meritorious nodes first before propagating further away in the network. One could then incorporate the merit value into the preference weights p_{ij} of the original BR formulation. This results in wirings in which nodes of higher merit are kept closer to the connecting node. Implementing this idea, however, requires addressing some non-trivial technical hurdles.

First, one must augment the current P2P protocols with additional functionality that will permit a node to gather the required information for computing a BR (namely the residual network). In the case of overlay routing, the link-state routing protocol running at the overlay layer provides this information, but in currently deployed P2P file-sharing systems, there is no equivalent capability. Second, even if the required information was to become available, the problem still remains that P2P applications have no way of performing shortest-path routing to a destination, which is a fundamental underlying assumption of the original formulation. This is because a querying node does not know *a priori* the identity of the destination node holding the file of interest. Employing full flooding would of course create an equivalent of shortest-path routing as queries would reach target nodes first through shortest-paths and then through non-shortest paths. Unfortunately, full flooding does not scale, and thus real unstructured P2P file-sharing systems rely on either scoped-flooding.[18] In scoped flooding, a successful search (meaning that the object is located) will indeed go over a shortest path. Objects that exist only outside the scope will simply not be reachable.

[17]Similar in spirit to reputation protocols.

[18]Random walks [75] for forwarding search queries have also been proposed. With random walks located objects are reached through paths that are not generally shortest-paths.

2.8.1 A reformulation of Best Response for scoped-flooding

Consider an unstructured P2P file sharing network employing scoped flooding of search queries with time-to-live r. Granted that in most such networks there's no *a priori* knowledge of other nodes' content, the search performance is optimized by maximizing the number of distinct nodes reachable by scoped flooding. This implies that first hop neighbors should be selected so as to cover as much as possible disjoint parts of the residual overlay topology. We reformulate our notion of BR so as to achieve this goal and compare with the corresponding search performance of k-Random which is the typical choice in many existing systems. We base our discussion on a two-tier unstructured P2P network (like KaZaA and the latest versions of Gnutella) with two types of nodes: Ordinary Nodes (ON) and Super Nodes (SN), which operates as follows:

(1) A newcomer node v_i connects to a bootstrap server and retrieves a set C with $m = |C|$ candidate SN's, from which it has to select k.

(2) The newcomer v_i contacts each one of the candidate SNs $v \in C$ and queries it for its list of first hop neighbors (and the type of each neighbor, ON or SN). Such capability is provided by most widely deployed unstructured P2P systems [111]. Then it queries all SN neighbors recursively up to distance $r - 1$ from the initial candidate v.

(3) After receiving all such information, the newcomer computes for each candidate SN v the set $F(v)$ of unique nodes (both ON and SN) that are reachable from it in $r - 1$ hops.

(4) To compute its BR, v_i has to select a wiring s of cardinality k so as maximize the cardinality of its coverage $F_i^r(s)$ over all possible wirings, with scope r:

$$F_i^r(s) = \bigcup_{v_j \in s} F_j^{r-1}(S_{-i})$$

The "best response" of the node requires the solution of a special case of the Miximum Coverage problem, where the profit of an element and the cost of selecting a subset of elements are uniform (1 and 0 respectively).

Definition 7 *(Maximum Coverage) Given a universe set* $U = \{e_1, ..., e_n\}$*, where each*

Algorithm 2 s_i=Local-k-Greedy-Profit($F_j^{r-1}(S_{-i}),\ \forall v_j \in V_i$)

```
Set s^(0) = ∅ and Φ^(0) = ∅ and C_Φ = 0;
While   C_Φ ≤ k
    v^(t) = arg max_{v_j ∈ V_i} Profit(F_j^{r-1}(S_{-i}) \ Φ^(t-1)) / C_j;
    s^(t) = s^(t-1) ∪ {v^(t)};
    Φ^(t) = Φ^(t-1) ∪ F_j^(r-1)(S_{-i});
    V_i = V_i − {v^(t)};
Return s = s^(k);
```

element e_i has a non-negative profit p_i, a collection $B = \{B_1, ..., B_m\}$ of subsets of U, an associated cost $C_i > 0$, for selecting a subset $B_i \in B$, and an integer k, pick sets of B with total cost at most k, to maximize the total profit of elements covered.

A straight-forward exhaustive search can find such a BR wiring in $O(nkm^k)$, where $m = |V_i|$, since there exist $\binom{m}{k} = O(m^k)$ possible wirings, and computing the cardinality of each one requires performing union operations on k sets $F_j^{r-1}(S_{-i}), \forall\ v_j \in V_i$, each one having size $O(n)$. Unfortunately, this pseudo-polynomial running time is essentially the best that can be achieved, as maximizing the cardinality of $F_i^r(s)$ is NP-hard.

The Algorithm 2 has unbounded approximation ratio but for the special case where $C_i \leq \epsilon k$, for a sufficient small $\epsilon > 0$, $(1 \leq i \leq m)$, then the aforementioned greedy algorithm has a ratio $(1 - \frac{1}{e})$-approximation ratio [47]. If the profit of covering an element is 1 and the cost to cover a set is 0, then the greedy algorithm has $O(nmk)$ complexity, which is capable of producing high-quality solutions, and provably optimal solution for the special case of k=2.

2.8.2 The value of Best Response for scoped-flooding

To demonstrate the benefits of BR as reformulated for scoped-flooding, we use the Gnutella trace presented in [111]. This dataset provides a realistic snapshot of the Gnutella topology with over $305,000$ ONs and SNs. We select an ON from this trace and let it be our "newcomer" node. Then we supply it with an unbiased random sample set C with m candidate nodes (as a bootstrap server for Gnutella would do). We compare the number of unique nodes reachable from this newcomer given its wiring as was reported in the dataset

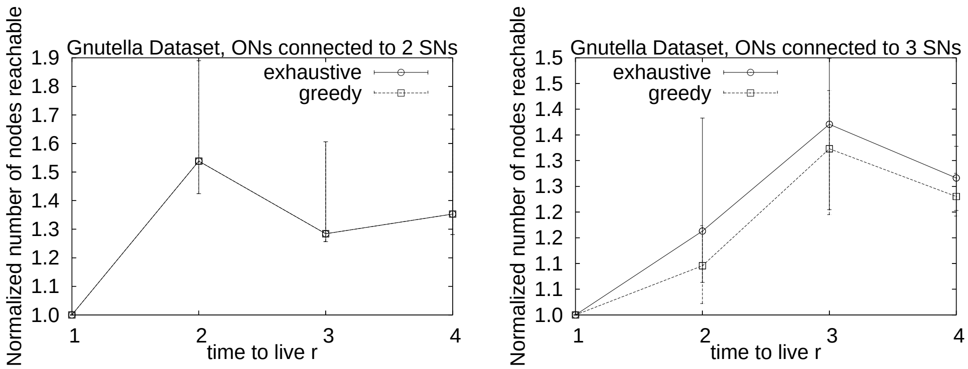


Figure 2·8: Gnutella simulations showing the improvement in node coverage using BR, reformulated for Scoped-flooding, and using exhaustive and greedy search algorithms. Improvement is relative to the coverage achieved using Gnutella's scoped flooding over its default wiring. Results are for ONs connected to two SNs (left) and three SNs (right).

and according to our reformulated BR.

We uniformly at random select two sets of 30 ONs from the aforementioned data set, which are connected to two and three SNs respectively. Each of these ONs connects to the bootstrap server and retrieves candidate SNs ($m = 10$, including those to which it is currently connected).

In Figure 2·8, we plot the ratio of the unique nodes reachable with scoped-flooding for different values of time-to-live (r) using our new BR formulation relative to that reported in the Gnutella dataset. The reformulated BR computed through exhaustive search increases significantly the number of nodes reached, with similar improvements achieved when the greedy heuristic is used for the computation.

2.8.3 Stable wirings under scoped-flooding

The Scoped-flooding game can be defined as in Section 2.2.

Definition 8 *(Scoped-flooding Game) The (n, k, r)-Scoped-flooding Game is a bounded degree local connection network creation game which is defined by the tuple $\langle V, \{S_i\}, \{F_i^r\}\rangle$, where:*

- *V is the set of n players, which in this case are the nodes.*

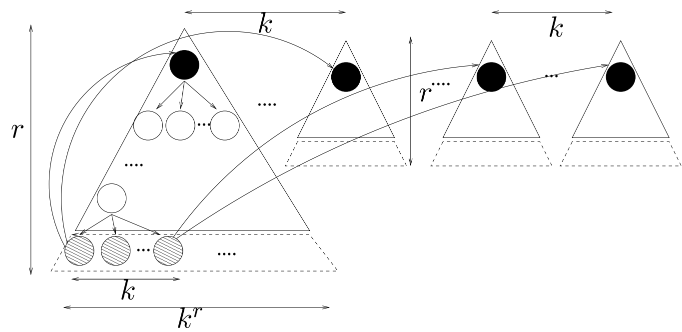


Figure 2·9: Provably stable graph for the uniform (n, k, r)-Scoped-flooding game.

- $\{S_i\}$ *is the set of wiring strategies available to the individual players.* S_i *is the set of wiring strategies available to* v_i*.*

- $\{F_i^r\}$ *is the set of utility functions for the individual players. The utility of player* v_i*, under an outcome (wiring)* S*, is* $F_i^r(S)$*, where* r *is the horizon of the scoped flooding process, and* k *is the maximum out-degree of the node.*

For the uniform (n, k, r)-Scoped-flooding game, pure Nash equilibria with good properties exist.

Proposition 2 *For* $r \leq \frac{1}{2}(\log_k n - 1)$ *there is a configuration of the uniform* (n, k, r)*-Scoped-flooding game, that is pure Nash equilibrium and socially optimal.*

Proof: We describe a pure Nash equilibrium configuration, that is socially optimum. The core of our construction contains a forest of k^r interconnected node-disjoint trees, that we call *utopian* trees (illustrated in Figure 2·9). We call the forest a *utopian forest.* Every utopian tree is a balanced k-ary tree of height r: every node connects by directed links to k immediate children nodes (except for the leaves, which we will accommodate later). Then a utopian tree contains exactly $\frac{k^{r+1}-1}{k-1}$ nodes. One can see that for $r \leq \frac{1}{2}(\log_k n - 1)$ there are enough nodes, so that construction of k^r node-disjoint trees of height r is possible: the total number of distinct nodes contained in the utopian forest is:

$$\frac{k^{r+1}-1}{k-1}k^r \leq k^{2r+1} \leq k^{2\frac{1}{2}(\log_k n - 1)+1} = n$$

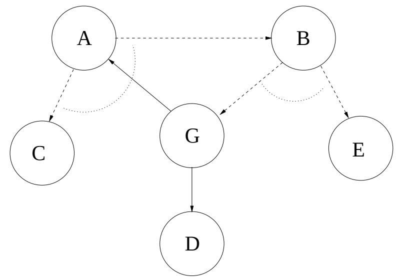


Figure 2·10: A setting where pure Nash Equilibrium for the non uniform (n, k, r)-Scoped-flooding game does not exist. Wirings of nodes A and B are not stable for given profits of nodes C, D, E, F, G. Solid links (and other that are not illustrated) are stable.

We explain how these trees are interconnected and how the remaining nodes connect to them so that the resulting configuration is a pure strategy Nash equilibrium. At first we accommodate the leaf nodes of each tree as follows. For a tree $T_{k,r}$ divide the set of its leaves into k groups of k^{r-1} leaf-nodes each, so that every group corresponds to the leaves of the same subtree of $T_{k,r}$. Let L_p, $p = 1 \dots k$ be these groups. Every leaf node $u \in L_p$ must have k descendants. The total number of descendants of nodes in L_p is k^r. Fix an arbitrary ordering of $v_i \in L_p$, $i = 1 \dots k^{r-1}$. Let u_1 connect to the root of $T_{k,r}$. For the remaining $k^r - 1$ needed descendants we use the roots of the $k^r - 1$ trees of the forest, other than $T_{k,r}$. Each of the nodes (players) other than the ones belonging to the forest simply connects to the roots of k arbitrary distinct trees of the forest. This completes our construction.

It is easy to verify that by construction every node of the utopian forest can reach a maximum possible number of distinct network nodes within the scope horizon r and given k, that is exactly $\frac{k^{r+1}-1}{k-1} - 1 = \frac{k(k^r-1)}{k-1}$. This naturally holds for nodes outside the forest as well. Therefore no node has incentive to change any of its k connections. Furthermore, the social gain is the maximum possible. Thus this construction is socially optimum as well.

■

For non uniform (n, k, r)-Scoped-flooding games, we provide a counter example (with race conditions) to demonstrate that pure Nash equilibria may not exist. Consider the setting where all but the nodes in the network that are illustrated in Figure 2·10 have stable wiring, except nodes A, B. Assume that $k = 2$ and $r = 3$. Both node A and B have stabilized one of their link (that is not illustrated). The profit by connecting to other nodes in the network is the following: for A, profit(B)=1/2, profit(C)=2, profit(D)=1+ϵ, profit(G)=1/2; and for B, profit(A)=1/2, profit(B)=1+ϵ, profit(E)=2, profit(G)=1/2; $\epsilon < 1/2$. Node B re-wires to G and increases its profit from 2 to 2+ϵ, as A is now connected to C. Node A, may have access to D now within r hops, so it re-wires and connects to B, increasing its profit from 2 to 2+ϵ. B now prefers to connects back to E, as this connection gives the highest benefit. A then connects back to C as this is the connection that returns the maximum profit. The aforementioned re-wirings will repeat, thus the total wiring will never stabilize.

2.9 Chapter Summary

Our experimental results on selfish neighbor selection, in a richer model that captures the nuances of overlay applications more faithfully than previous work, reveals numerous subtleties that are not apparent in simpler models. Among our most noteworthy findings is that it is typically in a newcomer's best interest (whether that newcomer is naïve or sophisticated) to have had the prior arrivals behave selfishly, as the underlying "best-response" graph is often highly optimized in favor of the newcomer. A corollary is that suboptimal behavior by a participant is often costly, not only to the individual, but to the population at large, *i.e.,* suboptimal behavior leads to large negative externalities. We also show how SNS variations may lead to optimized graphs to better serve real-time applications and query propagation in unstructured peer-to-peer networks.

Chapter 3

The EGOIST Overlay Routing System

Our evaluation of selfish neighbor selection wiring strategies on static topologies in the previous Chapter demonstrates that selfish users can select neighbors so as to efficiently reach near-equilibria in the Nash sense, while also providing good global performance. One implication of this work is that shortest path overlay routing performs much better over SNS topologies than over random or myopic ones. Left unanswered in this prior work, though, is whether it is practical to build SNS-inspired overlays, how to incorporate additional metrics other than delay, *e.g.,* bandwidth, what is the average performance gain when SNS wiring strategies are used in highly dynamic environments, whether such overlays are robust against churn, and whether they scale. In this Chapter we address the questions mentioned above and describe the design, implementation, and evaluation of EGOIST: an SNS-inspired prototype overlay routing network. EGOIST serves as a building block for the construction of efficient and scalable overlay applications consisting of (potentially) selfish nodes.

Our contributions can be summarized as follows. We first demonstrate through real measurements on PlanetLab that overlay routing atop EGOIST is significantly more efficient than systems utilizing common heuristic neighbor selection strategies under multiple performance metrics, including delay, system load and available bandwidth. Second, we demonstrate that the performance of EGOIST approaches that of a (theoretically-optimal) full-mesh topology, while achieving superior scalability, requiring link announcements proportional to nk compared to n^2 for a full mesh topology. We also demonstrate that the computational, memory and traffic overhead to create and operate EGOIST in minimal.

Third, to accommodate high-churn environments, we introduce a hybrid extension of the "Best-Response" (BR) neighbor selection strategy, in which nodes "donate" a portion of their k links to the system to assure connectivity, leaving the remaining links to be chosen selfishly by the node. Our experiments show that such an extension is warranted, especially when the churn rate is high relative to the size of the network. Fourth, we consider the impact of cheaters – nodes that announce false information in order to benefit themselves, or harm the network. While such behavior can be identified and eliminated through the use of appropriate mechanisms, we show that EGOIST remains robust even without the use of such mechanisms. Finally, we discuss how EGOIST can provide a redirection stepping-stone for the benefit of end-user applications including file transfer and multiplayer peer-to-peer games. The list of EGOIST's artifacts is provided in Section 3.7.

3.1 Background

The work presented in this Chapter is inspired by the SNS game introduced in the previous Chapter. There, the focus was on presented basic theoretic and experimental results, without considering any of the practical systems issues that are covered in this Chapter, such as dealing with churn in realistic network conditions or achieving high global performance without the computational and control message overheads required by theoretical formulations. Network Creation Games that predate SNS [34, 25, 84, 23, 29] have considered settings in which nodes may buy as many links (neighbors) as they like and thus differ fundamentally from our work, in which constraints on the number of neighbors play a central role.[1] Also, as it was highlighted in the previous section, fundamentally different is the work on Selfish Routing [94, 100], in which the network topology is part of the input to the game, and selfish source routing is the outcome. In a way, this is the inverse of our work, in which network-based (shortest-path) routing is an input of the game, and topology is the outcome. Selfish Routing is also based on source routing which is either

[1] We also note that in our target (overlay networks) setting, it is more realistic to impose a limit on the number of neighbors as opposed to a price on the link to a neighbor. This latter assumption is better justified for connectivity at the physical as opposed to overlay layer.

not provided in most system implementations, or it is difficult to perform well in systems with high churn like peer-to-peer systems.

A number of routing overlay systems have been recently proposed [101, 2, 70, 69, 119, 73, 45, 121, 102, 108, 113, 30]. Most of these have been proposed as ways of coping with some of the inefficiencies of native IP routing. The basic design pattern is more or less the same: overlay nodes monitor the characteristics of the overlay links between them (overlay topology may differ among systems) and employ a full-fledged or simpler [45] routing protocol to route at the overlay layer. Some overlay routing systems optimize route hop count [69, 102, 108], others optimize for application delay [101, 2, 94, 70, 119, 73, 45], and others optimize for available bandwidth [121]. These works assume that either all overlay nodes are under central control and thus obediently follow simple empirical neighbor selection strategies as discussed earlier, or bypass the issue altogether by assuming that some fixed overlay design is already in place. With reference to the employed metric, in our work, we provide mechanisms to support optimization of *all* aforementioned metrics and leave it up to the application designers to choose the most suitable one.

Selfish behavior has been studied in the context of providing incentives for nodes to route traffic for others [15].[2] Such works are complementary to ours since we assume that an external mechanism exists for incenting forwarding for other nodes. Chawathe et al. [22] proposed mechanisms for dealing with selfish nodes that lie about their capacities to avoid receiving queries. While we visit some of these issues in this Chapter, we note that this prior work did not focus on neighbor selection nor did it impose any constraints on node degrees.

In structured DHTs, proximity neighbor selection has been proposed to make the overlay topology match the underlying IP topology as much as possible [96, 43] in order to achieve faster lookups: Nodes can choose the physically closest nodes from a set of candidate nodes. While this approach gives to nodes some flexibility in choosing neighbors

[2] The use of incentives has also been studied in other contexts that are fundamentally different from ours, *e.g.,* P2P file sharing [36, 24].

selfishly, the set of nodes from which the choice can be made is constrained by node ID and thus tuning it at will becomes impossible [115]. Undoubtedly, DHTs are able to provide the best possible indexing of objects in a network. On the other hand, routing of traffic on DHTs has been shown to be sub-optimal due to local forwarding [59, 78]. EGOIST can be integrated as a different layer in DHTs; when an object is mapped onto a node, EGOIST is responsible to optimally route the content.

3.2 Preliminaries

Let $V = \{v_1, v_2, \ldots, v_n\}$ denote a set of overlay routing nodes. Node v_i establishes a *wiring* $s_i = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ by creating links to k other nodes (we will use the terms link, wire, and edge interchangeably). Edges are *directed* and *weighted*, thus $e = (v_i, v_j)$ can only be crossed in the direction from v_i to v_j, and has cost d_{ij} (in general, $d_{ji} \neq d_{ij}$). Let $S = \{s_1, s_2, \ldots, s_n\}$ denote a *global wiring* between the nodes of V and let $d_S(v_i, v_j)$ denote the cost of a shortest directed path between v_i and v_j over this global wiring; $d_S(v_i, v_j) = M \gg \max_{i,j} d_{ij}$, if there is no directed path connecting the two nodes. In our implementation, we computed shortest path using Dijkstra's algorithm. Given than the graph is sparse, we used the most efficient implementation of the algorithm using Fibonacci heap that requires $O(|E| + |V| \log |V|)$ amortized time, where $|E|$ is the number of edges in the graph [26].

For the overlay networks discussed here, the above definition of cost amounts to the incurred end-to-end delay when performing shortest-path routing along the overlay topology S, whose direct links have weights that capture the delay of the underlying IP path connecting one end of the overlay link to the other. Let $C_i(S)$ denote the cost of v_i under the global wiring S, defined as a weighted summation of its distances to all other nodes, *i.e.,* $C_i(S) = \sum_{j=1, j\neq i}^{n} p_{ij} \cdot d_S(v_i, v_j)$, where the weight p_{ij} denotes "preference" *e.g.,* the percentage of v_i's traffic that is destined to node v_j.

The best response of a node is defined in Section 2.2. Given a residual wiring $S_{-i} = S - \{s_i\}$, a best response for node v_i is a wiring $s_i \in S_i$ such that $C_i(S_{-i} + \{s_i\}) \leq$

$C_i(S_{-i} + \{s_i'\})$, $\forall s_i' \neq s_i$, where S_i is the set of all possible wirings for v_i. The *Selfish Neighbor Selection* (SNS) game was introduced in Section 2.2 as a strategic game where nodes are the players, wirings are the strategies, and C_i's are the cost functions. It was shown that under hop-count distance, obtaining the BR of v_i requires solving an asymmetric k-median problem on the residual wiring S_{-i} and is, therefore, NP-hard. To overcome the computational obstacle, we applied the local search heuristic [6] that provides a solution in a polynomial number of iterations. Experimental results (in the previous section) showed that the performance of the above heuristic is within 5%. In [59] it was proved that every instance of the SNS game with uniform preference and link weights has pure Nash equilibria whose social cost is within a constant factor of that of a socially-optimal solution. It was also shown that non-uniform instances of the game may have no equilibria at all. In this Chapter however, we are turning our attention to the average performance gain that individual nodes and the system can achieve through best-response walks in a real operational environment.

3.3 Architecture

In this section we overview the architecture of our EGOIST overlay routing system.

3.3.1 Basic Design

EGOIST is a distributed system (evaluated on PlanetLab) that allows the creation and maintenance of an overlay network, in which every node selects and continuously updates its k overlay neighbors in a selfish manner—namely to minimize its (weighted) sum of distances to all destinations under shortest-path routing. For ease of presentation, we will assume that *delay* is used to reflect the cost of a path, noting that other metrics – which we will discuss later in the Chapter and which are incorporated in EGOIST's implementation – could well be used to account for cost, including bandwidth and node utilization.

In EGOIST, a *newcomer* overlay node v_i connects to the system by querying a *bootstrap* node, from which it receives a list of *potential* overlay neighbors. The newcomer connects to at least one of these nodes, enabling it to participate in the link-state routing protocol running at the overlay layer. As a result, after some time, v_i obtains the full residual graph G_{-i} of the overlay. By running all-pairs shortest path algorithm on G_{-i}, the newcomer is able to obtain the pair-wise distance (delay) function $d_{G_{-i}}$. In addition to this information, the newcomer estimates d_{ij}, the weight of a potential direct overlay link from itself to node v_j, for all $v_j \in V_{-i}$. Using the values of d_{ij} and $d_{G_{-i}}$, the newcomer connects to G_{-i} using one of a number of wiring policies (discussed in Section 3.3.2). In our implementation, each node acts as a server that listens to all the messages of the link state protocol and propagates them only to its immediate neighbors. In order to reduce the traffic in the system, each node propagates only unique messages by dropping messages that have been received more than once or have been superseded. There are also two threads, one for estimating d_{ij}, and one responsible for estimating the new wiring and propagating the wiring to the immediate neighbors. In order to minimize the load in the system, a node propagates its wiring to its immediate neighbors only if this changes.

Clearly, obtaining d_{ij} for all n nodes requires $O(n^2)$ measurements.[3] However, we note that these $O(n^2)$ measurements do not have to be announced or be continuously monitored. In particular, each node needs to monitor and send updates only for the k links that it chooses to establish, with $O(n)$ measurements to all nodes in the overlay done much less frequently – namely once per *wiring epoch*, which is defined as the period T between two successive evaluations by a node of its set of candidate links and possible adoption of a new wiring (*i.e.,* re-wiring) based on such evaluation. Since re-wiring is much less frequent than monitoring of the established k links, the load imposed by the link-state protocol is only $O(nk)$ and not $O(n^2)$.

[3] Note that d_{ij} can be obtained through active or passive measurements depending on the metric of interest (see Section 3.4.1 for details).

3.3.2 Neighbor Selection Policies

As its namesake suggests, the default neighbor selection policy in EGOIST is the Best-Response (BR) strategy described in Section 3.2. Using BR, a node selects all its k neighbors so as to minimize a local cost function, which could be expressed in terms of some performance metric (*e.g.,* average delay to all destinations, maximum aggregate throughput to all destinations, or any combination of the above). Since obtaining an exact BR is computational expensive under both delay (see Section 2.3.2) and throughput (see Appendix A), in Section 3.4.1, we employ fast approximate versions based on local search (that was introduced in Section 3.2) to reduce computational costs and enhance scalability. In addition to BR, we have also implemented the following neighbor selection policies in order to perform a comparative evaluation.

***k*-Random**: Each node selects k neighbors randomly. If the resulting graph is not connected, we re-wire some links to enforce a cycle upon it.

***k*-Closest**: Each node selects its k neighbors to be the nodes with the minimum link cost (*e.g.,* , minimum delay from it, maximum bandwidth, *etc.*). Again, if the graph is not connected, we enforce a cycle.

***k*-Regular**: In this case, all nodes follow the same wiring pattern dictated by a common offset vector $o = \{o_1, o_2, \ldots, o_k\}$, used as follows: node i connects to nodes $i + o_j \mod n$, $j = 1, \ldots, k$. In our system, we set $o_j = 1 + (j-1) \cdot \frac{n-1}{k+1}$. One way to visualize this is to consider that all nodes are placed on a ring according to their ids (as with a DHT). Thus, an offset vector makes each node use its k links to connect to other nodes so as to equally divide the periphery of the ring.

3.3.3 Dealing with churn

EGOIST's BR neighbor selection strategy assumes that existing nodes never leave the overlay. Therefore, even in an extreme case in which some nodes are reachable through only a unique path, a node can count on this path always being in place (re-wirings by other nodes will not tear it down as this would also disconnect them [59]). Overlay routing networks (*e.g.,* RON [2]) are not inherently prone to churn to the extent that file-sharing P2P-networks [41, 97] are. Nonetheless, nodes may occasionally go down, or network problems may cause transient disconnections until successive re-wirings establish new paths. One could re-formulate the BR objective function used by a node to take into account the churning behavior of other nodes. This, however, requires modeling of the churn characteristics of various nodes in an overlay, which may not be feasible, particularly for large networks [118].

In EGOIST, we follow a different approach reminiscent of how k-Random and k-Closest policies ensure overlay connectivity. We introduce a hybrid wiring strategy (HybridBR), in which each node uses k_1 of its k links to selfishly optimize its performance using BR, and "donates" the remaining $k_2 = k - k_1$ links to the system to be used for assuring basic connectivity under churn. We call this wiring "hybrid" because, in effect, two wiring strategies are in play – a selfish BR strategy that aims to maximize local performance and a selfless strategy that aims to maintain global connectivity by providing redundant routes.

There are several ways in which a system can use the k_2 donated links of each node to build a connectivity backbone. Young et al. [119] proposed the use of k Minimum Spanning Trees (k-MST). Using k-MST (a centralized construction) to maintain connectivity is problematic, as it must always be updated (due to churn and to changes in edge weights over time), not to mention the overhead and complexities involved in establishing ($k_2/2$)-MSTs. To avoid these complexities, EGOIST uses a simpler solution that forms $k_2/2$ bidirectional cycles. Consider the simplest case $k_2 = 2$, which allows for the creation of a single bidirectional cycle. To accommodate a new node v_{n+1}, node v_n will disconnect from node v_1 and connect to v_{n+1}, whereas the latter will connect to v_1 to close the cycle. For higher $k_2/2$, the system decides $k_2/2$ *offsets* and then each node connects to the nodes taken by adding (modulo n) its id to each offset. If k_2 is small (*e.g.,* 2) then the nodes will need to monitor (*e.g.,* `ping`) the backbone links closely so as to quickly identify and restore disconnections. With higher k_2 the monitoring can be more relaxed due to the existence of alternative routes through other cycles. Computing BR using k_1 links *granted* the existence of the k_2 links can be achieved by restricting the set candidate candidate immediate neighbors for swapping.

We have implemented HybridBR in EGOIST. As hinted above, donated links are monitored aggressively so as to recover promptly from any disconnections in the connectivity backbone through the use of frequent heartbeat signaling. On the other hand, the monitoring and upkeep of the remaining BR links could be done lazily, namely by measuring link costs, and recomputing BR wirings at a pace that is convenient to the node—a pace that reduces probing and computational overheads without risking global connectivity.

To differentiate between these two types of link monitoring strategies (aggressive versus lazy), in EGOIST we allow re-wiring of a dropped link to be performed in one of two different modes: *immediate* and *delayed*. In immediate mode, re-wiring is done as soon as it is determined that the link is dropped, whereas in delayed mode re-wiring is only performed (if necessary) at the preset *wiring epoch* T. Unless otherwise specified, we assume a delayed re-wiring mode is in use.

3.3.4 Dealing with Cheaters

In the aforementioned setting, the selfishness in the selection of neighbors has the game theoretic meaning of local optimization and does not imply any anti-social behavior that needs to be mitigated. In this section, we briefly examine such harmful ways in which a node may "cheat" its way through, as well as possible countermeasures.

The most blatant form of cheating is *free-riding*, *i.e.,* using the system to route one's own traffic but denying routing to any incoming traffic from other nodes. Dealing with such behavior has been the subject of a number of studies, including the works in [15, 18] which propose the adoption of reputation and repudiation or punishment mechanisms that act as incentives for nodes to route, and/or expel misbehaving nodes from the system. These studies are orthogonal to and thus complement our work.

A more elaborate way for a node to cheat is to announce false information via the link-state protocol to discourage others from picking it as an upstream neighbor. For example, a node can cheat by falsely announcing larger-than-actual delays for its potential outgoing links. One could add mechanisms to detect this type of cheating. If the construction of the overlay is based on passive measurements obtained from a virtual coordinate system (as discussed in Section 3.4.1), then nodes could periodically select a random subset of remote nodes and "audit them" by querying the coordinate system for the delays of the outgoing links of the audited nodes and comparing them to the actual values that the audited nodes declare on the link-state routing protocol. Similar audits can be designed using active probing by sending traffic and measuring its delay and comparing it to the expected delay based on the delays that nodes on the end-to-end path declare. In Section 3.4.5, we evaluate the impact of broadcasting false information to cheat the system: we show that even without the use of the aforementioned audit mechanisms, EGOIST is robust to this form of cheating.

3.4 Experimental Evaluation

In this section we present our evaluation for EGOIST based on large scale experiments using extensive measurements of paths between nodes that participated in EGOIST.

3.4.1 Cost Metrics

As alluded earlier, a number of metrics can be used to measure the "cost" of traversing an overlay link. Clearly, the choice of an appropriate one depends largely on the application at hand. In this section, we review the various metrics we have incorporated in EGOIST.

Link and Path Delays: Delays are natural cost metrics for many applications, especially those involving interactive communication. To obtain the delay cost metric, a node needs to obtain estimates for its own delay to potential neighbors, and for the delay between pairs of overlay nodes already in the network. In EGOIST, we estimate directed (one-way) link delays using two different methods: an active method based on `ping`, and a passive method using the `pyxida` virtual coordinate system [65]. Using `ping`, one-way delay is estimated to be one half of the measured `ping` round-trip-times (RTT) averaged over enough samples.

Clearly, a node is able to measure such a value for all of its direct (overlay) neighbors, and is also able to relay such information to any other nodes through the overlay link-state routing protocol. To estimate the distance to nodes that were configured not to reply to `ping`, we used application layer `ping`. Using `pyxida`, delay estimates are available through a simple query to the `pyxida` system. [4]

Node Load: For many overlay applications, it may be the case that the primary determinant of the cost of a path is the performance of the nodes along that path—*e.g.,* if traversal of nodes along the path incur significant overhead due to (say) context switching and frequent crossing of user/kernel spaces. Thus, in EGOIST, we allow the use of a variation of the delay metric in which all outgoing links from a node are assigned the same cost, which is set to be equal to the measured load of the node. When applicable, the estimation of such a metric is straightforward as it requires only local measurements. In EGOIST, we did this by querying the CPU load of the local PlanetLab node, and computing an exponentially-weighted moving average of that load calculated over a given interval (taken to be 1 minute in our experiments querying the `loadavg` reports).

Available Bandwidth: Another important cost metric, especially for content-delivery applications, is the available bandwidth on overlay links. Different available bandwidth estimation tools have been proposed in the literature [103]. In EGOIST , we used `pathChirp` [98], a light-weight, fast and accurate tool, which fits well with PlanetLab-specific constraints, namely: it does not impose a high load on PlanetLab nodes since it does not require the transmission of long sequences of packet trains, and it does not exceed the max-burst limits of Planetlab. `pathChirp` is an end-to-end active probing tool, which requires the installation of sender and receiver `pathChirp` functionality in each EGOIST node. The available bandwidth between a pair of nodes $v, u \in V_{-i}$ is given by:

$$AvailBW(v,u) = \max_{p \in P(v,u)} AvailBW(p),$$

[4] Using `ping` produces more accurate estimates, but subjects the overlay to added load, whereas using `pyxida` produces less accurate estimates, but consumes much less bandwidth.

Algorithm 3 ρ=AvailBW($G(V,E)$, $s \in V$)

Set $W = \{s\}$; and $\rho[s] = \infty$;
for all $y \in V - \{s\}$ do $\rho[y] = d_{sy}$;
while $W \neq V$ do
begin find x = argmax$\{\rho[y] : y \notin W\}$;
    set $W = W \cup \{x\}$;
    for all $y \in V - W$ do
        $\rho[y] = \max\{\rho[y], \rho[x] + d_{xy}\}$
end
Return ρ;

where the available bandwidth for a path p is given by:

$$AvailBW(p) = \min_{e \in p} AvailBW(e),$$

and $P(v,u)$ denotes the set of paths that connects v to u. Thus, finding $P^*(v,u)$ that maximizes the available bandwidth between v and u, and the bottleneck edge, is a "Maximum Bottleneck Bandwidth" problem which can be solved using a simple modification of Dijkstra's algorithm as shown in Algorithm 3.

3.4.2 Baseline Experimental Results

In this section, we present performance results obtained through measurement of EGOIST. These results allow us to make comparisons between the various neighbor selection policies described in Section 3.3.2 for the various cost metrics described above. All the results in this section assume that node churn is not an issue – *i.e.,* once it joins the overlay, a node does not leave. Results showing the impact of node churn on EGOIST performance are presented in Section 3.4.4.

Experimental Setting: We deployed EGOIST on $n = 50$ PlanetLab[5] nodes (30 in North America, 11 in Europe, 7 in Asia, 1 in South America, and 1 in Oceania). Each of these nodes is configured to recompute its wiring every wiring epoch $T = 60$ seconds. EGOIST nodes are not synchronized, thus on average a re-wiring by some EGOIST node occurs

[5]PlanetLab is an overlay testbed network of approximately 700 nodes in more than 300 academic, industrial, and government sites around the world.

every $T/n = 1.2$ seconds. Whether a node ends up re-wiring or not depends on the neighbor selection policy. For k-Random and k-Regular policies, and since our baseline experiments do not feature any node churn, it follows that these policies will not exhibit any re-wiring. For k-Closest, re-wiring would only be the result of dynamic changes in PlanetLab that result in changes to the cost metric in use (and hence what constitutes the closest set of neighbors). For BR, a node may rewire due to changes in PlanetLab conditions, but may also rewire simply as a result of another node's re-wiring. While in theory [59], BR strategies (under certain conditions) converge to some equilibrium in the Nash sense, we note that this is not likely to be the case for real systems such as EGOIST, since dynamic changes of the underlying system (changes in link delays, bandwidth, and node load) are likely to result in perpetual re-wiring by EGOIST nodes. Setting the wiring epoch T in EGOIST has the effect of controlling the timescale of, and consequently the overhead incurred by, BR re-wiring.

Each experiment presented in this section reflects the results obtained by running EGOIST for at least 10 hours on PlanetLab on January 5th, January 15th, September 15th, October 3rd 2007, and April-June 2008.

To be able to compare the impact of neighbor selection on the quality of the resulting overlay, throughout this paper we use the *routing cost* (for an individual node or averaged over all nodes) as the main performance metric. For each experiment, an individual cost metric is calculated for every one of the $n = 50$ nodes in the system. The individual cost metric for a node reflects the cost of routing from that node to all other 49 nodes in the system, assuming a uniform routing preference over all destinations.[6] For each experiment we report the mean of all $n = 50$ individual costs, as well as the 95^{th}-percentile confidence interval.

To facilitate comparisons between various neighbor selection strategies, we often report the *normalized routing cost* (and the 95^{th}-percentile confidence interval), which is the ratio

[6]We note that using a uniform routing preference will tend to deflate the advantage of BR neighbor selection – in other words, the results we present here are conservative, since unlike the other policies we considered, BR is capable of leveraging skew in preference to its advantage.

of the cost achievable using a given strategy to that achievable using BR.

Control Variables: In our first set of experiments, our aim is to identify for the three metrics of interest the payoff (if any) from adopting a selfish neighbor selection strategy, *i.e.,* using a BR policy in EGOIST . This payoff will depend on many variables. While some of these variables are *not* within our control (*e.g.,* the dynamic nature of the Internet as reflected by variability in observed PlanetLab conditions), others are within our control, *e.g.,* n, T, and the various settings for our active measurement techniques.

In order to neutralize the effect of extrinsic variables that are not within our control, experiments reporting on different neighbor selection policies were conducted *concurrently.* To do so, we deploy concurrent EGOIST agents on each of the $n = 50$ PlanetLab nodes we use in our experiments, with each agent using a different selection strategy. In effect, each experiment compares the performance of a *set* of concurrently deployed EGOIST overlay networks, each resulting from the use of a particular neighbor selection policy.

One control variable that is particularly important is the *number of direct neighbors*, k, that an EGOIST node is allowed to have. In many ways, k puts a premium on the significance of making a judicious choice of neighbors. For small values of k, choosing the right set of neighbors has the potential of making a bigger impact on performance, when compared to the impact for larger values of k. Thus, in all the results we present in this section, we show the performance of the various policies over a range of k values.

Overview of Performance Results: Before presenting specific performance results, we make two broad observations: first, in all of our experiments, using a BR policy in EGOIST consistently yields the best performance. While such an outcome was anticipated by virtue of findings reported in the previous Chapter for a static setting, the results we present here are significant because they underscore the payoff in a *real* deployment, where the modeling assumptions made in prior work do not hold. Second, in all of our experiments, with the exception of BR, no single neighbor selection policy was consistently better than all others across all metrics. In other words, while the performance of a given policy may approach that of BR for one metric while dominating all other policies, such policy dominance does

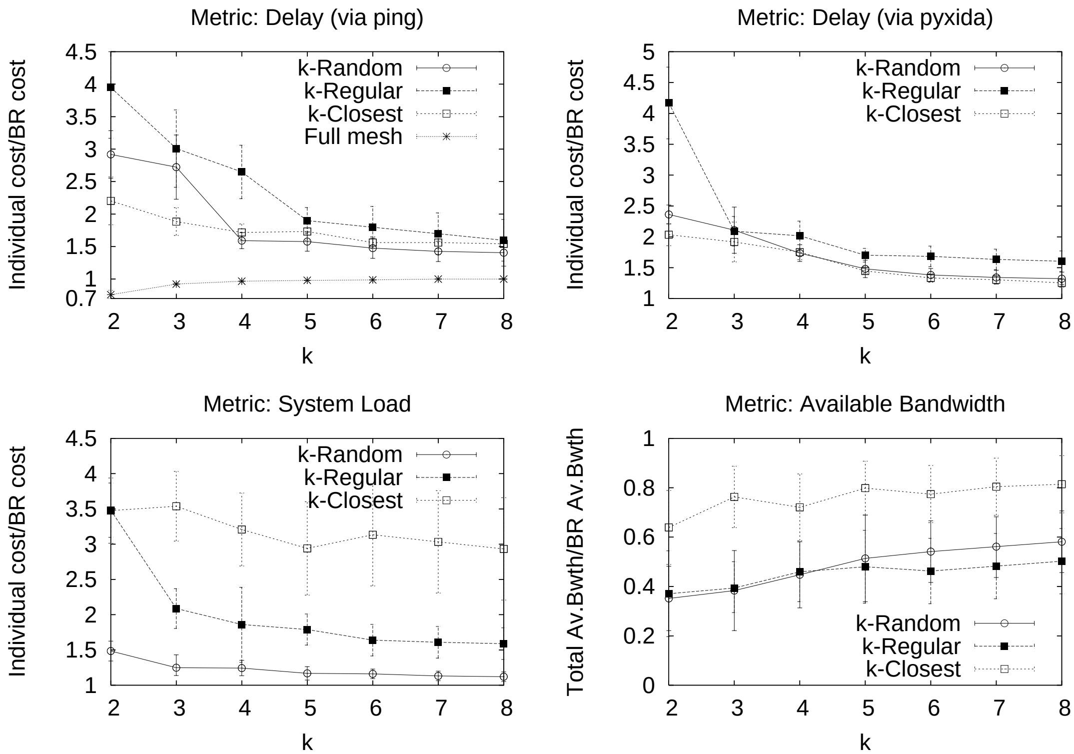


Figure 3·1: PlanetLab Baseline experiments showing the individual costs for various neighbor selection policies (normalized with respect to BR costs) as a function of number of neighbors k for a 50-node EGOIST overlay: Cost metric is `ping` delays (top-left), `pyxida` delays (top-right), node CPU load (bottom-left), and available bandwidth (bottom-right).

not hold across all the metrics we considered.

Results for Delay Metric: Figure 3·1 shows the performance of the various neighbor selection policies in EGOIST normalized with respect to that achievable using BR when the metric of interest is the overlay link/path delay over a range of values for k (with link delays measured using `ping` in the top-left plot, and using `pyxida` in the top-right plot). These results show that BR outperforms all the other wiring policies, especially when k is small, as anticipated in our discussion of the significance of k as a control variable. For example, for $k = 2$, the average delay experienced by an individual node could be anywhere between 200% and 400% *higher* than that achievable using BR. The performance advantage of BR in terms of routing delay stands, even for a moderate number of neighbors. For example, for $k = 5$, BR cuts the routing delay almost by half.

These results confirm the superiority of BR relative to other policies, but do not give us a feel for how close is the performance of EGOIST using BR wiring to the "best possible" performance. To do so, we note that by allowing nodes to connect to all other nodes in the overlay, we would be creating a complete overlay graph with $O(n^2)$ overlay links, obviating the need for a neighbor selection policy. Clearly, the performance of routing over such a rich overlay network gives us an *upper bound* on the achievable performance, and a lower bound on the delay metric. Thus, to provide a point of reference for the performance numbers we presented above, in the top-left plot in Figure 3·1 we also show the performance achieved by deploying EGOIST and setting $k = n - 1$. Here we should note that this lower bound on delay is what a system such as RON [2] would yield, given that routing in RON is done over shortest paths established over a full mesh, and assuming that any of the $O(n^2)$ overlay links could be used for routing. These results show that using BR in EGOIST yields a performance that is quite competitive with RON's lower bound. As expected, the difference is most pronounced for the smallest k we considered—namely, the lowest delay achievable using 49 overlay links per node is only 30% lower than that achievable using BR with 2 overlay links per node. BR is almost indistinguishable from the lower bound for slightly larger values of k (*e.g.,* $k = 4$).

With respect to the other heuristics, the results in the top plots in Figure 3·1 show that k-Closest outperforms k-Random when k is small, but that k-Random ends up outperforming k-Closest for slightly larger values of k. This can be explained by noting that k-Random ends up creating graphs with much smaller diameters than the grid-like graphs resulting from the use of k-Closest, especially as k gets larger. In all experiments, k-Regular performed the worst.

Results for Node Load: The bottom-left plot in Figure 3·1 shows the results we obtained using the node load metric, where the path cost is the sum of the loads of all nodes in the path. These results show clear delineations, with BR delivering the best performance over all values of k, k-Random delivering the second-best performance, and k-Closest delivering the worst performance as it fails to predict anything beyond the immediate neighbor, especially in light of the high variance in node load on PlanetLab.

Results for Available Bandwidth: The bottom-right plot in Figure 3·1 shows the results we obtained using available bandwidth as the cost metric. Recall that, here, the objective is to get the highest possible *aggregate* bandwidth to all destinations (again, assuming a uniform preference for all destinations) – thus, larger is better. These results show trends that are quite similar to those obtained for the delay metric, with BR outperforming all other policies—delivering a two-fold to four-fold improvement over the other policies, over a wide range of values of k.

3.4.3 Measurement and Re-wiring Overheads

In this section we show experimentally that EGOIST introduces a rather small amount of overhead for maintaining the overlay structure.

Active Measurement Load: As mentioned in Section 3.4.2, in the absence of node churn, k-Random and k-Regular do no perform any re-wirings, and thus do not introduce measurement overheads. For k-Closest and BR, the active measurement load is identical.

When the cost metric is delay via `ping`, `ICMP` messages of size 320 bits each (`ECHO` requests/replies) are exchanged once per wiring epoch T. Notice that for established links, there is no need for active measurements since the cost metric for a link would be available by virtue of its use. Thus, the overhead is $\approx (n - k - 1) \cdot 320/T$ bps per node. Using `pyxida`, a single (`http`) request/reply to the `pyxida` server yields the (virtual coordinate space) distances between the node initiating the request and all other nodes in the overlay. This is clearly more efficient than using `ping`, as it injects $\approx (320 + 32n)/T$ bps per node.[7] When the metric is system load, there is no overhead imposed on the network as the system load is measured locally at each node. Finally, when the metric is available bandwidth, our experimental results showed that the bandwidth needed for accurate probing of available

[7]Measurements showed that a rate of one message per (one minute) wiring epoch per node was sufficient to sustain a coordinate system in PlanetLab.

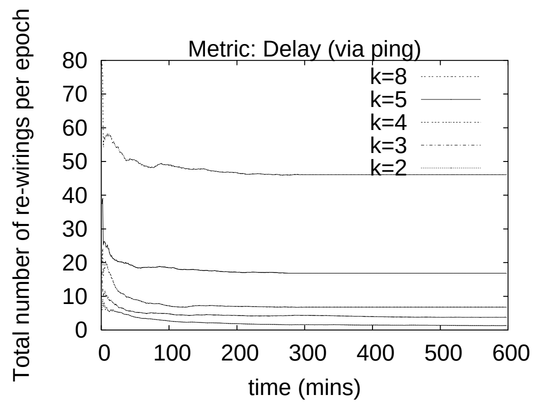


Figure 3·2: PlanetLab experiments showing the total number of re-wirings per epoch in the system as a function of the number of neighbors k in EGOIST overlay.

bandwidth between two nodes in the overlay is less than 2% of the pairwise bandwdith.

Link-State Protocol Load: The overhead (in terms of additional injected traffic) imposed by the link-state protocol is also low. Each node broadcasts a packet with its ID, its neighbors' IDs and the cost of the established links to its k neighbors every $T_{\text{announce}} < T$. The header and padding of the link-state protocol messages require a total of 192 bits, and the payload per neighbor requires 32 bits. Thus, the overhead in terms of injected traffic on the overlay is $\approx (192 + 32 \cdot k)/T_{\text{announce}}$ bps per node. In our experiments we set T_{announce}=20 secs. The above can be seen as an upper limit, as only unique link state messages forwarded in the overlay (as mentioned in Section 3.3.1). In our implementation, no node spent more than 1 Kbps to maintain the network.

Re-wirings Overhead: Figure 3·2 (left) shows the total number of re-wirings per (one minute) epoch for the entire overlay over time. The results suggest that the re-wiring rate decreases fast as EGOIST reaches a "steady state" and that the re-wiring rate is minimal for small values of k. Here we note that as k increases the re-wiring rate increases, but the improvement (in terms of routing cost) is marginal, as a small number of outgoing links is sufficient to significantly decrease the cost. This is evident in Figure 3·3. Finally, we also note that the re-wiring rate can significantly be decreased (with marginal impact on routing cost) by requiring that re-wiring be performed only if connecting to the "new"

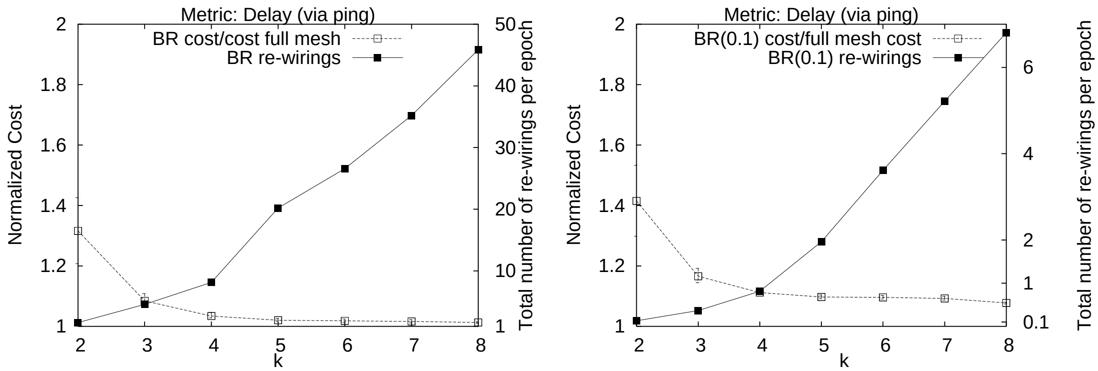


Figure 3·3: PlanetLab experiments showing relationship between individual cost and total number of re-wirings per epoch in the system with exact best response and an approximate best response with $\epsilon = 10\%$ (left and right respectively), as a function of the number of neighbors k in EGOIST overlay.

set of neighbors would improve the local cost to the node by more than a given threshold ϵ. We refer to this modified version of BR as BR(ϵ). Figure 3·3 (right) confirms this by showing the number of re-wirings and resulting performance when $\epsilon = 10\%$.

We also measured the memory and CPU consumption using `time` of Unix. In Figure 3·4 we show the average CPU and memory utilization, along with the average bandwidth consumption to maintain the overlay per node. Both the CPU and memory consumption is close to 0%, and the bandwidth consumption per node is negligible. It is worth mentioning that the in-degree was quite uniform in all our experiments, thus no node allocated significantly more CPU power, memory, or bandwidth than any other in the overlay.

3.4.4 Effect of Churn

In the original SNS formulation,the graphs resulting from the SNS-game as well as from the empirical wiring strategies were guaranteed to be connected, so they could be compared in terms of average or maximum distance. Node churn, however, can lead to disconnected graphs, therefore we have to use a different metric. For that purpose, we choose the *Efficiency* metric [64], where the Efficiency ϵ_{ij} between node i and j ($j \neq i$) is inversely proportional to the shortest communication distance d_{ij} when i and j are connected. If

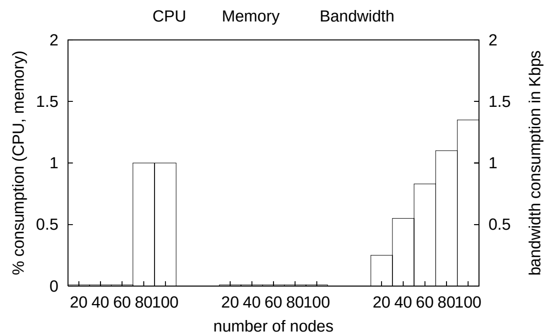


Figure 3·4: Percentage of CPU and memory consumption, and bandwidth consumption in EGOIST. The metric is delay via `ping`.

there is no path in the graph between node i and j then $\epsilon_{ij} = 0$. The Efficiency ϵ_i of a node i defined as:

$$\epsilon_i = \frac{1}{n-1} \sum_{j \neq i} \epsilon_{ij}$$

To evaluate the efficiency of nodes in EGOIST overlays under churn, we allow each of the $n = 50$ nodes in the overlays to exhibit ON and OFF periods. During its ON periods, a node "joins" the overlay, performs re-wiring according to the chosen policy, and fully participates in the link-state routing protocol. During its OFF periods, a node simply drops out from any activity related to the overlay. The ON/OFF periods we use in our experiments are derived from real data sets of the churn observed for PlanetLab nodes [41], with adjustments to the timescale to control the intensity of churn.

In addition to evaluating the efficiency of various neighbor selection policies we have considered so far, we also evaluate the efficiency of HybridBR, which allows a node to donate $k_2 = 2$ of its links to ensure connectivity (*i.e.,* boost the efficiency of the overlay) while using BR for the remaining links.

The top plot in Figure 3·5 shows the achievable efficiency of the various neighbor selection policies when churn is present. As before, the efficiency of the various policies is normalized with respect to that achievable using BR, and is shown as a function of k. As with all the metrics we considered so far, BR outperforms all other policies (including

HybridBR), but as EGOIST nodes are allowed to have more neighbors (*i.e.,* as k increases), the efficiency of the HybridBR approaches that of BR, with the efficiency of k-Closest decisively better than k-Random and k-Regular.

The above results imply that under the level of churn in these experiments, it is not justifiable for BR to donate two of its links simply to ensure connectivity, especially when k is small. Notice that BR overlays that get disconnected due to churn will naturally heal as soon as any of its active nodes decides to rewire. This is so because the (infinite) cost of reaching the disconnected nodes will act as an incentive for nodes to choose disconnected nodes as direct neighbors, thus reconnecting the overlay. As noted earlier, re-wiring occurs every T/n units of time on average (1.2 seconds under our settings), which implies that the vulnerability of BR to disconnections due to churn is highest for smaller overlays and if re-wiring is done infrequently. Our results also showed that adding or removing a node from the overlay does not increase the number of re-wires in the system. Under moderate churn, and random selection of a node to add or delete, the number of re-wirings in the system are similar to those reported in Section 3.4.3.

Our last question then is whether at much higher churn rates, it is the case that the use of HybridBR would be justified. To answer this question, we changed the timescale of the ON/OFF churn processes to emulate more frequent joins and leaves. The bottom plot in Figure 3·5 shows the results by plotting the efficiency metric for the various policies as a function of the churn rate (on the x-axis), which we define (as in [41]) to be the sum of the fraction of the overlay network nodes that changed state (ON/OFF), normalized by time T:

$$Churn = \frac{1}{T} \sum_{events\ i} \frac{|U_{i-1} \ominus U_i|}{max\{|U_{i-1}|, |U_i|\}},$$

where U_i is the new set of nodes in the overlay following an event i that alters the membership in the set of nodes that participate in the overlay, and $\ominus$ is the symmetric set difference. Thus, a churn rate of 0.01 implies that, on average, 1% of the nodes join or leave the overlay per second. For an overlay of size $n = 50$, this translates to a join or leave

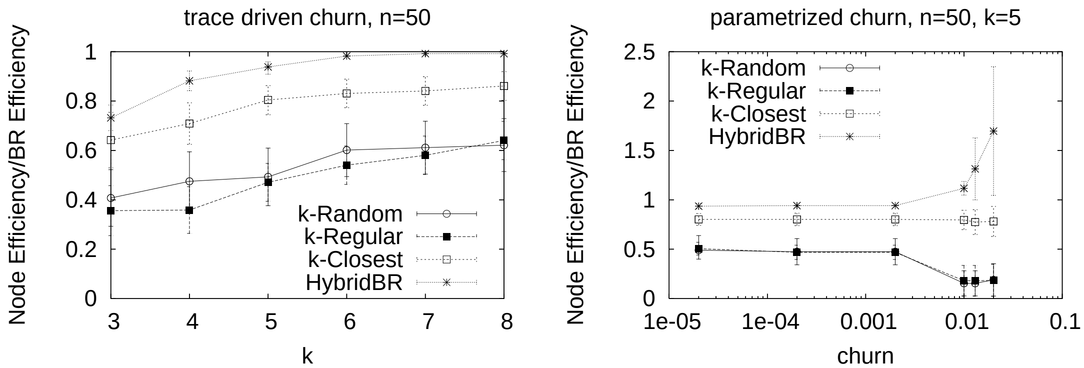


Figure 3·5: PlanetLab experiments with node churn showing the efficiency of neighbor selection policies (normalized with respect to BR) as a function of the number of neighbors k (top) and churn (bottom) for a 50-node EGOIST overlay.

event every two seconds.

As expected, when churn rate increases significantly to the point where the average time between churn events approaches T/n, the efficiency of HybridBR eventually surpasses that of BR. The results also suggest that under such conditions, the efficiency of both k-Random and k-Regular fall dramatically, whereas that of k-Closest remains level with that of BR.

3.4.5 Vulnerability to Abuse

As we discussed in section 3.3.3, cheating nodes may attempt to game the system by declaring false link costs to their neighbors in order to benefit from EGOIST without contributing their own resources to the overlay. Due to the combinatorial nature of the optimization problem underlying BR re-wiring, and the out of order rewirings of individual nodes, it is very hard for individual cheaters to derive the proper costs that will lead to wirings that will be of benefit to them individually, while harming others. Theoretical results [7] advocate that such behavior may even lead to worse equilibria for cheaters in routing games. Thus, in this section, we present results from a series of experiments aimed to assess EGOIST's vulnerability to cheaters that misrepresent the cost of their outgoing links (simply by inflating them), in the hopes of discouraging others from selecting them as neighbors.

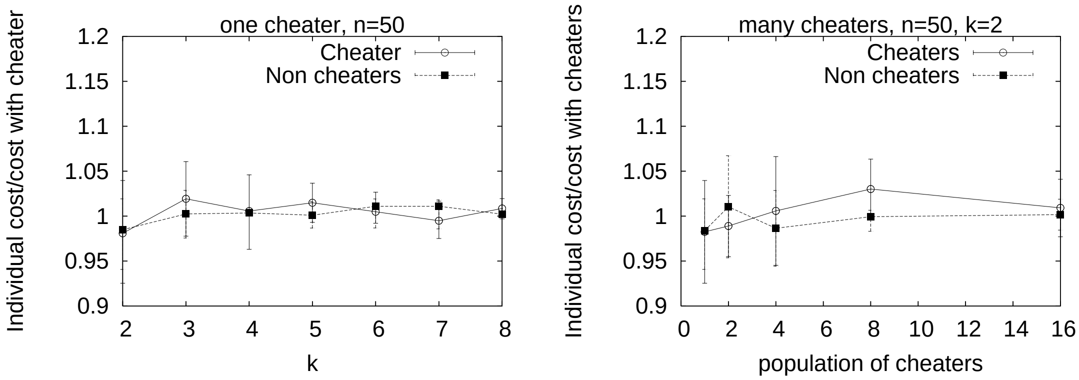


Figure 3·6: PlanetLab experiments with cheater showing the robustness of neighbor selection policies (normalized with respect to BR) as a function of the number of neighbors k in the presence of one cheater (top) and many cheaters for $k = 2$ (bottom) for a 50-node EGOIST overlay.

As described in Section 3.3.3, one could add mechanisms to detect when cheaters make such false representations. These mechanisms would take the form of passive or active measurement audits from other nodes. Determining how often nodes should perform such random audits and what these nodes do when cheating nodes are identified can be complex. Thus, it would be preferable if one can show that the impact from such abuse is minimal. Clearly, an assessment of the impact of the full spectrum of possible false announcements is beyond the scope of this paper. Thus, we only consider the impact from inflated delay announcements by a single node and by a variable fraction of the nodes.

In Figure 3·6 (left), we show the impact from a single cheater announcing link costs that are twice as high as the real ones. The figure shows the individual cost for both the cheater and for all other normal nodes for different values of k. The cost for both types of nodes is very close to the cost without the presence of the cheater. We also evaluate the robustness of EGOIST in the presence of many cheaters (up to one-third of the population). These results, shown in Figure 3·6 (right), yield consistent observations even when the number of outgoing links is very small ($k = 2$), which is the setting in which the impact of bad re-wirings is amplified. These results provide evidence that EGOIST is fairly robust to abuse by cheaters, even without the deployment of auditing mechanisms.[8]

3.5 Scalability Issues

In this section we address potential scaling limitations of EGOIST by describing methods that *sample* the large space of possible neighbors and compute BR wirings based on only these samples. We also propose a layered architecture where some of the nodes (*e.g.*, commodity or super nodes) are participating in EGOIST, while others use them as relays to route traffic.

3.5.1 Scalability via Sampling

A sampling technique might not be necessary for current overlay networks that are of small to moderate sizes, such as PlanetLab, but are likely to become essential in emerging overlays of massive scale. One such example we foresee is that of future "P2P reincarnations" of overlay routing that allow participating nodes to opportunistically choose overlay routes with minimal overhead. Unlike today's systems such as RON, which require central installation and maintenance by an interested party, these large systems would likely be self-organizing and self-regulating.

There are several aspects of an overlay routing network that become potentially problematic at scale: the overhead of the underlying link-state protocol, the cost of performing local search to compute BR, and scaling questions associated with the sampling process itself. We view the scaling issues associated with link-state routing as modest, since in EGOIST we limit the number of monitored and announced links to much less than $O(n^2)$ (*i.e.,* when $k \ll n$), and thus the per-node communication complexity scales as a function of k, not n.

A more significant scaling issue is imposed by the computational complexity of computing best responses. As mentioned before, computing an exact BR is an NP-hard problem.

[8] Similar observations were also obtained when the abuse amounted to advertising lower values than the actual delays.

Approximate solutions based on local search perform well in practice. However, even local search [6], imposes substantial computational burden (polynomial number of iterations, each one requiring $n^{O(p)}$, $p \in [1, k]$ being a parameter of the algorithm). Such high order polynomial complexity becomes difficult to handle for large n, especially when nodes must re-wire frequently to cope with the dynamics of the network. To handle such cases, we propose scaling down the input by computing BR based on a *limited* number of samples from the residual overlay graph. This enables us to run a computationally efficient algorithm (sampling) on the large input, and then run a computationally expensive BR algorithm on the scaled input. Later we will show that with an appropriate sampling technique in place, BR retains its performance edge over the other heuristics.

A natural approach would be to compute v_i's BR based on a sample of m nodes obtained through *unbiased random sampling* of the total n nodes of G_{-i}. This would limit the input to the parts of the distance function $d_{G_{-i}}$ that involve pairs that belong to the chosen sample. Also v_i would need to measure its distance to only those m samples. As we will show experimentally, such an approach has some value, but there is much more to gain by a simple biased sampling.

Topology-Based Biased Random Sampling: The basic idea is to take $m' > m$ random samples and apply topological filters to keep those m that are likely to yield the best results.

The heuristic approach we apply is to bias our samples towards nodes with the largest neighborhoods of radius r (*e.g.,* with the highest number of distinct nodes reachable in r hops). Defining $F(v_j)$ to be the size of the neighborhood of radius r around v_j, we give consideration to $|F(v_j)|$ as well as the distances of nodes within $F(v_j)$ from the perspective of the source v_i. This reflects the intuition that an ideal candidate for v_i has a large neighborhood of nodes, many of which are relatively close to v_i. Our ranking function b_{ij} establishes a priority order on candidates v_j as follows:

$$b_{ij} = \frac{|F(v_j)|}{\sum_{u \in F(v_j)} d(v_i, u)}$$

Using this ranking function, v_i chooses a sample of m nodes with the highest b_{ij} values

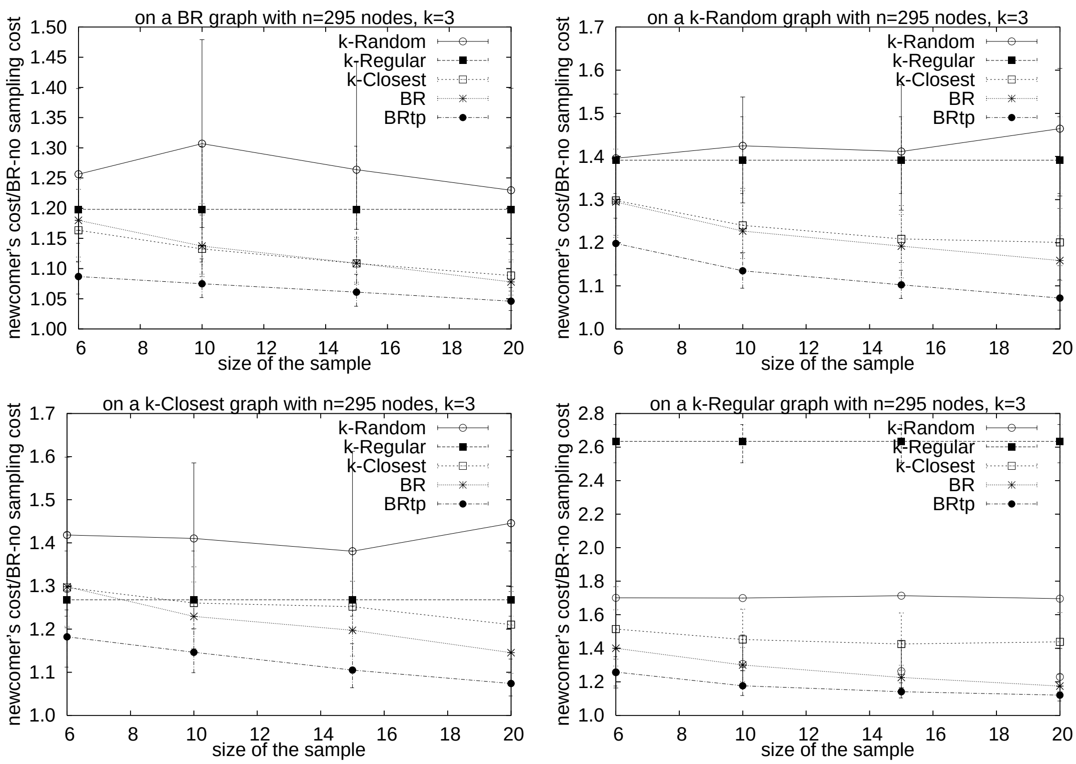


Figure 3·7: PlanetLab Simulation. The cost incurred by simple wiring strategies (k-Random, k-Regular, k-Closest with random sampling), BR with random sampling, and BR with topology-based biased random sampling (normalized against the cost of BR with no sampling) in a BR (top-left), k-Random (top-right), k-Closest (bottom-left), and k-Regular (bottom-right) graph.

and computes its BR based on these nodes only.

Finally, we need to verify that this sampling procedure itself is not prohibitive. Standard random-walk based methods can query a set of m' pseudorandomly-generated nodes in a k-regular graph with suitable expansion properties using $O(m' \log n / \log k)$ messages. Each node must be able to approximately maintain and express the number of nodes within its r-radius neighborhood, which requires $O(k^r)$ space. Nodes also must compute the b_{ij} values, which requires $O(m'k^r)$ distance lookups. All of this amounts to a reasonable overhead for the small fixed values of r and k that we focus on in this work.

Experimental Validation: To assess scalability, instead of our 50 node PlanetLab prototype, we use a publicly available trace [9] containing delays obtained using `pings` between all pairs of existing PlanetLab sites.[10] We use these data sets to conduct simulations. We test the four neighbor selection strategies of Section 3.3.2. In our simulation, an n-node network is constructed incrementally using the BR strategy (without sampling). A newcomer joins the network using one of the following strategies: k-Random, k-Regular, k-Closest, and BR, each with random sampling, and BR with topology-based sampling. In the simulation, $n = 295$, $k = 3$, and the neighborhood size $r = 2$. In Figure 3·7 (top-left), we plot the ratio of the newcomer's cost to the cost of using BR with no sampling for different sample sizes. The line labeled "BR" denotes the ratio when the newcomer uses BR with random sampling; "BRtp" denotes BR with topology-based sampling.

Our general observations across the experiments are that BR with sampling fares better than any of the three empirical rules, and that even for small m/n, the newcomer's cost ratio is not much larger than 1. We also find that topology-awareness in sampling improves the BR wiring significantly in all cases considered. It is also worth mentioning that the performance of simple heuristics with random sampling in a BR graph is good, due to its highly optimized structure. In graphs formed by nodes that follow the previously mentioned random or myopic heuristics, we observed that the performance gain of topology-biased random sampling is substantially better compared to any other wiring policy which is based on random sampling (see Figures 3·7).

3.5.2 Layered Architecture

An architectural way to achieve sampling is to deploy a layered version of EGOIST, where some of the nodes *e.g.,* commodity or supernodes, are responsible to relay traffic of nodes that do not participate in EGOIST, henceforth called non-overlay nodes. The non-overlay sender first initiates a request to the bootstrap server to receive a list of overlay nodes that

[9] `http://ping.ececs.uc.edu/ping/`, accessed on July 10, 2006.

[10] We are interested in sampling one node per AS, in order to achieve a more representative view of the network.

participate in EGOIST. The same process is initiated by the non-overlay receiver. The receiver non-overlay node contacts the non-overlay sender and sends the IP of the overlay node to which he will receive service. The previous "hand-shaking" also ensures that both parties agree to exchange traffic. The sender then encapsulates the IP of the receiver, adds as a destination of each packet the IP of the overlay node attached to the receiver and sends the packets to its closest overlay nodes. The packets are routed using EGOIST (if the attached overlay nodes of the sender and receiver are not identical), and when the packet reaches the overlay node attached to the receiver, the overlay node decapsulates the header and sends the packet to the non-overlay receiver. An application programming interface (API) to support this application in provided as an artifact of EGOIST (see Section 3.7).

3.6 Applications

EGOIST is a general purpose overlay routing network that can be used by applications to supplement traditional IP routing. The main difference between an EGOIST overlay and other routing overlays is that by virtue of its BR-wiring strategy, an application contacting its local EGOIST node can be assured that this node will provide better paths than a node that connects to the overlay non-selfishly, *e.g.,* using previously-mentioned random or myopic heuristics. Stated otherwise, the selfish selection of neighbors in EGOIST is just a manifestation of the desire of local applications to get the best possible service for themselves.

An application can connect to an EGOIST node by using a protocol interface that the latter exposes. This is an example of an application instance using EGOIST as a virtual router to communicate over the overlay with another application instance getting access to the overlay from a different EGOIST node. In the artifacts section we provide information on how to access our publicly available implementation which permits using PlanetLab nodes as such virtual routers.

A second option is to integrate EGOIST directly into an application through an API and a corresponding library which we have implemented and made available. In this case, both the application and its local EGOIST instance run at the same node. We have evaluated the performance benefits that EGOIST offers to different kinds of applications on top of EGOIST, including multiplayer P2P games, multi-path file transfer, and real-time traffic over IP.

3.6.1 Multiplayer P2P Games

Recently there has been intense interest [11, 10] for porting online multi-player games into P2P architectures that scale bett er and do not require dedicated expensive infrastructure. In this section we demonstrate the potential value of EGOIST for such applications.

We obtained from [10] a trace containing the movements of 100 players (artificial intelligence bots) participating in a game of Quake III. In Quake III, players are located in a virtual 3D world and interact frequently as they come into contact to fight each other. Two common events of the game are the creation of a new object (*e.g.,* a missile), and the update of an existing object (*e.g.,* update of its coordinates). Each update is about 230 bytes. All these updates have to be delivered to all the players that are in the vicinity of the affected object in the virtual world. This requires for building a multicast tree rooted at each player that is updating some of its objects.

We distributed the 100 players among our 25 EGOIST nodes on PlanetLab and used the EGOIST overlay to deliver the updates. We set $k = 2$ and mapped the L_3 distance of players i and j in the virtual world into the preference weight p_{ij} that defines the preference that the local EGOIST node of i has for sending messages to the local EGOIST node of j. Since the main requirement in this case is for high interactivity, we employed the delay-based version of BR. With this mapping, nodes pick as neighbors other nodes that host players that are closer in the virtual world which implies interaction, and thus requirement for small end-to-end delay. The value $k = 2$ is justified from the fact that due to human perceptual limitations, players usually pay attention and interact with a small number of other players [10].

In the above setting, we replayed the trace for a period of three minutes involving

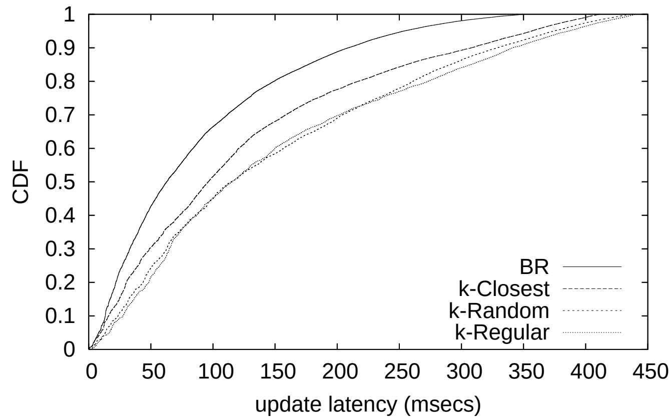


Figure 3·8: Comparison of update latencies for various neighbor selection policies.

more than 108,000 events. We compared the update latencies when sent over EGOIST and over k-Random, k-Closest and k-Regular wiring policies. The cumulative distribution function of update latencies is illustrated in Figure 3·8. Both the median (∼65 msecs) and the 95^{th}-percentile update latency over EGOIST is less than half of the corresponding latencies over k-Random and k-Regular, and less than two-thirds of those over k-Closest. Experimentally, it has been shown that update latency higher than 200 msecs may effect the quality of user's experience [10]. More than 90% of packets sent over EGOIST were delivered earlier than 200 msecs and only 60-70% under the other topologies.

3.6.2 Multipath File Transfer

File transfer applications can take advantage of redirection opportunities offered by (bandwidth-based) EGOIST to increase their effective end-to-end transmission rates by performing multipath transfers through first-hop neighbors. The idea is quite simple: Source node v_i uses EGOIST to establish up to k parallel sessions to a target node v_j, each one redirected through a different first-hop EGOIST neighbor $v_l \in s_i$. Each session requires establishing two virtual channels over EGOIST: $v_i \rightarrow v_l$ (single-hop overlay path) and $v_l \rightarrow v_j$ (multi-hop overlay path). The purpose of redirection through neighbors is to take advantage of

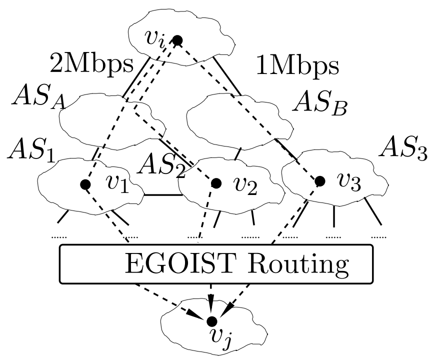


Figure 3·9: Source node v_i sending to target v_j through its $k = 3$ immediate EGOIST neighbors v_1, v_2 and v_3. v_i takes full advantage of its 2-homed AS$_i$: $v_i \rightarrow v_1$ and $v_i \rightarrow v_2$ use the maximum allowed bandwidth at the peering point with AS$_A$ (2 Mbps), whereas $v_i \rightarrow v_3$ uses the maximum allowed bandwidth at the peering point with AS$_B$ (1 Mbps). Assuming no bottlenecks exist further down, this gives an aggregate transmission rate of 3 Mbps, whereas any single-path scheme (even with parallel connections) would have limited to 1 or 2 Mbps.

potentially multihomed source and target ASes (henceforth AS$_i$ and AS$_j$) and thus alleviate bottlenecks caused by session-level[11] traffic shaping and rate-limiting at AS peering points. [12] As long as the number of EGOIST neighbors k is sufficiently larger than $|\text{AS}_i|$, the number of ASes to which AS$_i$ has a peering relationship, there is good chance that at least one overlay neighbor is behind each peering point. Redirecting through this neighbor permits v_i to utilize up to the maximum allowed rate at that peering point (see Figure 3·9 for an illustration). If peering points permit a given maximum rate for each session, the aforementioned multi-path redirection can increases the maximum total rate out of v_i by up to a multiplicative factor $|\text{AS}_i|$ (observe that establishing the same number of parallel connections going over the same path would not yield the same benefit, since they will all be part of a single session, and hence be subject to the same rate limits at peering points). Of course, the real end-to-end benefit can be much smaller due to bottlenecks on the overlay paths from v_l to v_j, especially in the last hops before closing-in on the target v_j (large $|\text{AS}_j|$'s working again in favor of the application). To get a feeling for the potential

[11] A session identified as a (source,target) IP pair.
[12] `http://www.wired.com/software/webservices/news/2007/08/p2p`

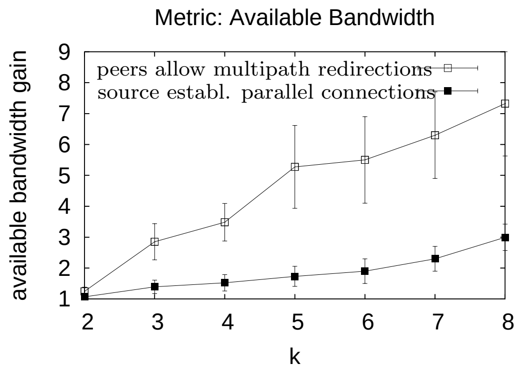


Figure 3·10: PlanetLab experiment showing the available bandwidth gain between source node u_i and target node u_j when the source establishes k parallel connections and when all the peering points between the aforementioned nodes, allow multi-path redirection in our 50-node EGOIST overlay.

benefits on a real topology, we perform the following experiment.

In our 50-node EGOIST overlay, we select a source-target pair and we estimate the available bandwidth that can be realized if the source establishes k parallel connections going through its immediate neighbors. Then we compare this value with the available bandwidth that is realized when the source routes the traffic using the unique path to the destination offered by IP. We repeat the experiment for all source-target pairs and we plot the average along with the 95^{th}-percentile confidence intervals in Figure 3·10. Furthermore, we estimate the theoretically maximum available bandwidth that can be realized when all peers allow multipath redirections for all source-target pairs (*i.e.,* when the total bandwidth becomes equal to a max-flow from v_i to v_j).

3.6.3 Real-time Traffic over IP

Applications that transmit real-time (*i.e.,* delay- and loss-sensitive) traffic can use the redirection infrastructure of (delay-based) EGOIST to send additional copies of the original stream through multiple disjoint paths, thus improving the chance that at least one copy of every packet will reach its destination before the designated playout time [62]. Some P2P voice-over-IP (VoIP) applications, like *Skype*, are already in position to implement such schemes as they have achieved a huge user-base that provides ample opportunities for

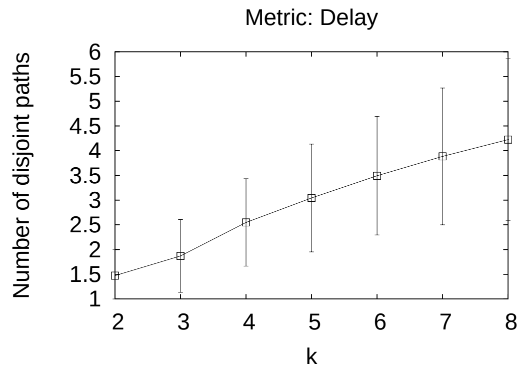


Figure 3·11: PlanetLab experiment showing the available bandwidth the number of disjoint paths between the source node u_i and target node u_j when the source establishes k parallel connections in our 50-node EGOIST overlay.

redirection. EGOIST on the other hand can assist applications that have not yet achieved high penetration (*e.g.,* high quality video-conferencing) and thus would otherwise have to rely on delay jitter-prone single-path delivery. Substantiating this claim requires measuring precise timing information and making sure that OS introduced delays do not interfere with the purpose of the experiment. We leave such elaborate experiments to future work. Our initial results show that the number of disjoint paths increases linearly with the number of parallel connections, as illustrated in Figure 3·11.

3.7 Artifacts

Our EGOIST prototype is currently deployed on PlanetLab. A live demonstration of the overlay routing topolopy maintained by EGOIST can be accessed from the EGOIST project web site at `http://csr.bu.edu/sns/`. Traces from all experiments used in this paper are also available from the project web site. EGOIST source code has also been released to the research community.

3.8 Chapter Summary

In this section we have shown how recent theoretical results on Selfish Neighbor Selection (SNS) could be leveraged for overlay routing applications. Through the development and deployment of our EGOIST prototype routing network on PlanetLab, we have established that Best-Response (BR) neighbor selection strategies can indeed be realized in practice, that they provide a substantial performance boost when compared to simpler empirical strategies, and that they scale much better than full-mesh approaches which require intensive monitoring of $O(n^2)$ links. We have substantiated these benefits under different performance metrics, active and passive link monitoring strategies, in static and churn-prone environments, and in the presence of truthful and untruthful nodes. Furthermore, we proactively equipped EGOIST nodes with the ability to compute their best responses based on samples of the residual network, so as to be in position to handle possible future scale growth in overlay routing networks.

Chapter 4

Swarming on Optimized Graphs

In previous Chapters we provided evidence in support of the potential benefits selfish neighbor selection in the network layer offers to network creation and maintenance. Left unanswered in this previous work, though, is whether the selfish neighbor selection primitives are superior when the neighbor selection takes place in the application layer. To that end, we focus on file-sharing applications.

In modern file-sharing systems, *swarming*, *i.e.*, parallel download of chunks of a file from multiple peers with concurrent upload to other requesting peers, has been one of the most efficient methods for multicasting bulk data. Many file-sharing systems, including the popular BitTorrent [24], are using the swarming principle to deliver voluminous amount of content. A fundamental characteristic of the existing swarming-based systems like BitTorrent is that the overlay graph resulting from its bootstrap and choke/unchoke[1] algorithms is mostly ad-hoc, in the sense that it is the outgrowth of random choices of neighboring peers. This is justified given the scale of P2P file swapping networks. Moreover, to address scalability issues the local scheduling of chunks in the network is based on the well established *Local Rarest First* (LRF) heuristic [66] that looks at the peer-set and issues a request for any missing chunk that is among the least replicated ones in the peer-set. Most of the work to improve swarming applications focused on changing the choke/unchoke algorithm [24] or the local scheduling procedures [12]. We argue that any performance improvement that can be achieved by enhancing the above mechanisms is tied to the topological characteristics of the file-sharing network. The optimization of the topology might not be critical for P2P

[1]The choke/unchoke mechanism is responsible to select the immediate neighbors to exchange traffic.

file swapping applications, like BitTorrent, where a user might be patient to download the file. On the other hand, there are many high-performance file-sharing applications, where the enhancement of the topology is critical.

In this Chapter we consider n-way broadcasting — a class of applications, in which each one of n overlay nodes must push a very large chunk of data (a distinct file) to all other $n-1$ peers, as well as pull the $n-1$ files pushed by these other peers. Once completed, this push-pull cycle may be repeated with new sets of files. Applications using n-way broadcasting would involve small/medium-sized networks, as they are inherently of n^2 nature. Examples include high-performance applications like distribution of large scientific data-sets in grid computing, distribution of large traffic log files for network-wide distributed intrusion/anomaly detection schemes [68], synchronization of distributed databases [9], and several other enterprise applications. Contrary to the prevailing assumption underlying the design of BitTorrent-like systems, the nodes that make up such networks are basically cooperative in routing each other traffic (at an extreme case they belong to the same administrative authority).

Even for relatively small networks, n parallel broadcasts of distinct large files can create data volumes that are impossible to handle via centralized solutions: uploading each file to a centralized server and then copying it back to all destinations in a point-to-point manner means that the same file is transmitted $O(n)$ times over the same link, *i.e.,* imposing an $O(n)$ stress on the physical links.

n-way broadcast via swarming: Swarming is clearly an attractive approach to supporting n-way broadcast applications. The obvious solution is to outsource the push-pull functionality to BitTorrent: set-up n different torrents, each one seeded by a different node.

In this Chapter, we question the effectiveness of BitTorrent for n-way broadcasting (which is not what it is primarily designed to support). In particular, we note that BitTorrent runs on the topologies that result from the composition of its bootstrapping and choke/unchoke algorithms. These topologies are mostly unoptimized. Indeed, the only topological optimization in BitTorrent is a local one: under the choke/unchoke algorithm, fast peers are matched up with other fast peers *from within the same randomly bootstrapped neighborhood*. By virtue of the relatively small size of neighborhoods compared to the entire network, the resulting topology is close to being random. While randomly-bootstrapped graphs may possess desirable theoretical properties (such as small diameters), they are likely to be inefficient when compared to graphs that are systematically constructed to optimize a specific application. Notice that BitTorrent's matching of fast nodes is mostly in the protocol as an efficient tool against free-riding, rather than as a conscious attempt to optimize the overall overlay topology for applications such as n-way broadcast.

Swarming over optimized overlays: For n-way broadcast applications (as well as for other potential classes of applications), the overriding goal is to optimize the efficiency of the entire overlay as opposed to creating a tit-for-tat environment to reign in selfish, free-riding behavior of individual nodes. Also, the scale of the applications we envision makes it possible/practical to optimize the construction of the overlay, especially if distributed optimization is used.

Armed with this realization, our goal will be to construct highly efficient topologies to be used by swarming protocols for n-way broadcast. Specifically, we construct an optimized, common overlay network, upon which swarming is used. In order to control the stress of the physical links supporting the overlay, we impose an upper bound on the degree of the nodes in the constructed overlay network.

Next we present justification for several of the salient features of our solution – features that will be developed and presented fully later in the chapter.

Why swarming on top of an overlay? Because hop-by-hop relay of the entire file over a shortest-path tree embedded on the overlay topology and rooted at the seed node would take too long. We want to harness the power of parallel downloads as exemplified in BitTorrent.

Why use a common overlay? Because a topological optimization requires monitoring the performance of overlay links, and we want to amortize the cost of such monitoring — pay it only once per link and reuse the result for the benefit of all n transmissions (and avoid monitoring the same link up to n times as can happen if one builds n independent overlays).

How could swarming benefit from an end-to-end optimized overlay? Our overlays are optimized for end-to-end performance over multi-hop paths, *e.g.,* by maximizing the minimum available bandwidth to any destination over multiple paths, or by maximizing the total available bandwidth to all destinations over all available paths. From a single node's perspective, swarming involves point-to-point transfers within the neighborhood of that node. Each node, however, has in its neighborhood nodes that also belong to other "adjacent" neighborhoods. Noting this, one can see that, through swarming, data chunks eventually reach their destinations through multi-hop paths formed through single hop transfers between neighborhoods. If these multi-hop paths are end-to-end optimized, then swarming will be more effective in operating upon them as compared to upon unoptimized paths.

Why optimize the overlay based solely on network characteristics, without consideration of data availability? Arguably, one could conceive of more general overlay constructions in which neighbors are selected based on criteria involving both the network characteristics and the availability of chunks at each candidate connection point. In our work, we adopt a bandwidth-centric/data-agnostic approach to the construction of the overlay for two main reasons: (1) for large objects it is high bandwidth that leads to small delivery completion times and high object throughput; (2) the global state in terms of available chunks per node changes too frequently (with each successful chunk exchange between two nodes), resulting in an optimized topology that changes too frequently to be of practical use. The fact that we do not consider data availability in the construction of the overlay does not mean that data availability does not play a role in our approach: it does, but not at the overlay construction time-scale. Specifically, *we advocate a "two-pronged approach" operating at two distinct time scales*: at a coarse time scale, we address issues related to network characteristics through the construction of a dynamic, distributively optimized overlay, and at a finer time scale, we address issues related to data availability through the upload/download scheduling algorithms employed in the swarming protocol that runs on top of the overlay.

4.1 Background

This work is the fusion of two very recent thrusts in networking research: *network creation games* and *swarming protocols*. Network creation games appeared in computer science with the work of Fabrikant et al. [34] in which a set of nodes forms a network in a distributed manner driven by self-interest — each node pays for the creation of a number of links to other "neighbors" so as to minimize a hybrid cost that captures the purchase cost of these links and the delay for routing packets to all other destinations using own and remote links. The model targeted the creation of physical telecommunication networks through peering agreements between ISPs (hence the explicit modeling of the cost of buying a link). In Chapter 2, we studied the "capacitated" version of the above problem, targeting the construction of overlay routing networks — each node is given a bound on the number of immediate peering relationships that it can establish (defined by the protocol that implements such an overlay network) and selects the best neighbors so as to minimize its sum of distances to all destinations through shortest-path routes over the resulting overlay topology. These works differ fundamentally from the one presented in the Chapter in that they target *routing*, *i.e.,* they assume that a packet from v to u is of interest only to u. Intermediate nodes w that lay on the overlay path from v to u are there just to assist in the routing of the packet. In the setting described in this Chapter, each node is *broadcasting* a file to all destinations and thus intermediate nodes are also receivers in addition to being relay points. More fundamentally, in our case the delivery of information from v to u occurs not through a single path but (potentially) through all the available connected paths between the two end-points (because the file is cut into chunks which travel in parallel along different paths on the overlay). For this reason we employ max-flows as building blocks for designing the overlay (as opposed to shortest-paths which are used in point-to-point routing presented in the previous Chapters. Max-flows reflect better the nature of our application (broadcasting) as well as the nature of the employed technique for implementing it (swarming).

The BitTorrent protocol [24] has established swarming as one of the most fresh and promising ideas in contemporary networking research and thus has kicked-started a (tidal) wave of research articles in the area. Our fundamental difference from this body of work, whether analytic, *e.g.,* Qiu and Srikant [93], Massoulie and Vojnovic [80], Kumar and Ross [57], experimental, *e.g.,* Bharambe [12], or measurement based, *e.g.,* Izal et al. [48], Legout et al. [66], is that we have substituted the (close to) random graph resulting from BitTorrent's bootstrap and choke/unchoke algorithms with a highly efficient distributively optimized graph. As we show later on, such a switch boosts the performance of a swarming protocol running on top of it. We are able to obtain such highly efficient graphs because our interest is on smaller networks. We show that at such scales one can do much better than close to random.

Some other relevant works are the following ones. Massoulie et al. [79] recently showed that a simple distributed randomized algorithm can achieve the theoretical optimal broadcast rate given by Edmond's theorem [31] for a source node in a flow network. Compared to this work, we let each node select its neighbors and thus participate in the construction of the flow network, as opposed to taking it for granted. Gkantsidis and Rodriguez [40] have proposed the use of network coding as an alternative to BitTorrent's chunk scheduling algorithm. The performance benefit/added complexity ratio of employing network coding is not yet generally agreed upon [66]. Although we focus on BitTorrent-like swarming here, our optimized topologies should also benefit network-coding based swarming because they are oblivious to whether network coding is used or not.

Guo et al. [44] and Tien et al. [114] look at the design of multi-torrent systems. Their contribution is mostly on the measurement and the design of inter-peer incentive mechanisms for peers that participate in multiple torrents concurrently. They do not look at overlay construction issues. Interestingly, Tien et al. [114] provide justification for one of our design choices, which is to enforce that at any time there should be only one active torrent between any two nodes (more in Section 4.3). They show that deviating from this choice and allowing transferring between two nodes multiple chunks in parallel (one for each torrent), slows down the system by over-partitioning the upload bandwidth of nodes.

Other end-system multicast systems such as SplitStream from Castro et al. [19] and Bullet from Costic et al. [55] could be used to support n-way broadcasting by creating a separate overlay for each source. The problem with this approach is that there is no coordination across different overlays and thus there can be performance inefficiencies as well as significant overheads due to the redundant monitoring of the same physical paths multiple times from different overlays. Our approach is to construct one overlay for all sources and thus jointly optimize as well as share the monitoring cost.

The only work we are aware of on the intersection of overlay creation and BitTorrent is a very recent one from Zhang et al. [120]. It looks at the formation of Nash equilibria topologies in view of download-selfish peers that participate in a single torrent. Our overlay formation, although distributed and based on local utility functions is: (1) primarily targeting the optimization of the social utility of the network, meaning that all nodes are assumed to be under common control, and (2) considering both upload and download performance for multiple torrents, one at each node. We examine selfishness issues and how these could be addressed towards the end of our article, but this is just a supplement of our main contribution.

4.2 Peer-set Selection

Let $V = \{v_1, v_2, \ldots, v_n\}$ denote a set of nodes. Node v_i selects k other nodes to be in its *peer-set* $s_i = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ and establishes bidirectional links to them. Let $S = \{s_1, s_2, \ldots, s_n\}$ denote the edge set of the *overlay graph* $G = (V, S)$ resulting from the superposition of the individual peer-sets. Each link of G is annotated with a capacity c_{ij} which captures the available bandwidth [50] (availbw) on the the underlying IP layer path that goes from v_i to v_j. Capacities can be asymmetric, meaning that $c_{ij} \neq c_{ji}$ in the general case. Let $MF(v_i, v_j, S)$ denote the resulting max-flow from v_i to v_j under S. Let also $\Phi(v_i, S)$ and $\Psi(v_i, S)$ denote the minimum max-flow from v_i to any other node under S, and the sum of max-flows from v_i to all other nodes under S, respectively, *i.e.*:

$$\Phi(v_i, S) = \min_{v_j \in V_{-i}} MF(v_i, v_j, S),$$

$$\Psi(v_i, S) = \sum_{v_j \in V_{-i}} MF(v_i, v_j, S)$$

In the above definitions, each max-flow from v_i to an individual destination is computed independently of other max-flows from the same node to different destinations (*i.e.*, each one is computed on an empty flow network G). These definitions should not be confused with multi-commodity flow problems in which multiple distinct flows co-exist. [2]

Definition 9 *(Max-Min and Max-Sum peer-sets) A peer-set s_i is called Max-Min if it maximizes the minimum max-flow of node v_i,* i.e., $\Phi(v_i, \{s_i\} + S_{-i}) \geq \Phi(v_i, \{s_{i'}\} + S_{-i})$, $\forall s_{i'} \neq s_i$*, where S_{-i} denotes the superposition of the peer-sets of all nodes but v_i. Similarly, a peer-set is called Max-Sum if* $\Psi(v_i, \{s_i\} + S_{-i}) \geq \Psi(v_i, \{s_{i'}\} + S_{-i})$, $\forall s_{i'} \neq s_i$.

Lemma 1 *Finding a Max-Min or Max-Sum peer-set for v_i given S_{-i} is an NP-hard problem.*

Proof: See Appendices C and D. ■

These peer-set selection policies optimize the connectivity of a given node to the remaining network. One could say that this constitutes selfish behavior. This is indeed the case if the nodes use this connectivity to *only* disseminate their own file. However, when they also indiscriminately relay the files of others, which is the assumption for the applications we consider, then optimizing one's connectivity boosts the aggregate social performance of the network. Later on, in Section 4.6 we discuss what happens when the swarming protocol (running above the overlay) ceases to be indiscriminate with respect to the upload quality it gives to local and remote files.

Why Max-Min and Max-Sum? Given a flow network G, the *broadcast problem* asks what is the maximum (broadcast) rate at which a source v_i can deliver its stream concurrently to all other nodes. Edmonds showed in [31] that the broadcast rate is equal

[2] Note also that to derive a social objective function for n-way broadcasting is too complex.

to $\min_{v_j \in V_{-i}} \text{mincut}(v_i, v_j)$, which in view of the max-flow/min-cut theorem is equal to $\min_{v_j \in V_{-i}} MF(v_i, v_j)$. Therefore, the Max-Min peer-set is the peer-set that maximizes the broadcast rate of a node, or conversely the delivery rate to the slowest receiving peers. It does so by placing the links so as to boost the max-flow to these slowest peers. Of course for this to be possible there must be available bandwidth to be utilized at the IP level (this is reflected on the c_{ij}'s which steers the peer-set selection, and which are obtained through measurements as explained in Section 4.3). Edmonds gave an exponential time centralized algorithm for achieving the broadcast rate, which was later improved to a small polynomial time by Lovasz, Gabow and others [46]. Recently, Massoulie et al. [79] showed that a simple randomized decentralized algorithm can achieve a delivery rate that is arbitrarily close to the broadcast rate.

A Max-Sum peer-set on the other hand is a peer-set that maximizes the *theoretical maximum aggregate transmission rate* from a node. Contrary to the Max-Min peer-set that maximizes a provably attainable broadcasting rate, the Max-Sum maximizes only an upper bound on the aggregate rate which, in the general case, is not attainable due to contention for link bandwidth when max-flows from the same source to different destinations share common overlay links.[3] We elaborate with an example.

Consider the flow network of Figure 4·1 (top-left) in which all links have unit capacity and node 1 is the source. Computing each max-flow on an empty network we get that the max-flow from the source to nodes 2, 3, and 4 is equal to 1 whereas that to nodes 5 and 6 is equal to 2, thereby $\Psi(1) = 7$. Consider now the maximum real flows that can exist concurrently from the source to nodes 5 and 6 (top-center). Breaking the file into two equal parts A and B the source can transmit A at full rate over the dotted paths ($1 \to 2 \to 5$ and $1 \to 4 \to 6$) and B at full rate over the dashed path (only once over link (1,3)) and achieve concurrent real flows that match the capacity of corresponding

[3]The contention between max-flows "from" different sources does not come explicitly in these objective functions. It is captured in our framework through the measured availbw c_{ij}: the availbw on a direct overlay link from v_i to v_j depends on the capacity of the underlying physical path *and the amount of this capacity already captured by the competing max-flows from other sources. At this level the problem is indeed a multi-commodity flow.*

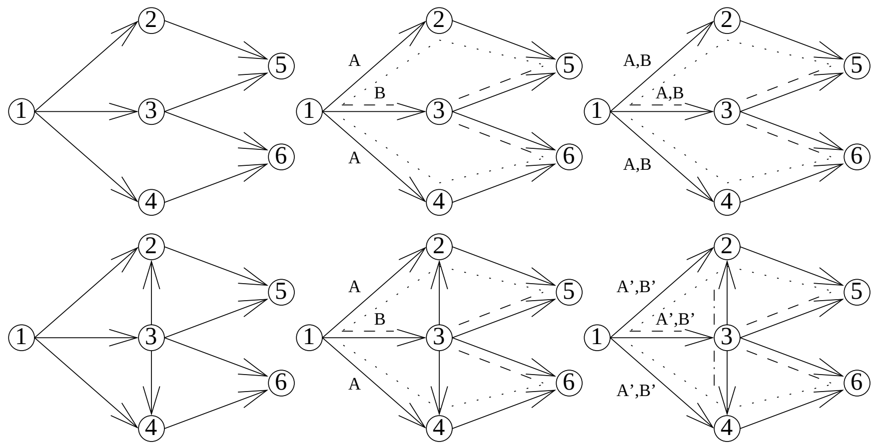


Figure 4·1: Mixing max-flows. Left: empty network. Middle: RF(1,5) and RF(1,6) co-existing. Right: RF(1,2), RF(1,3), RF(1,4), RF(1,5), RF(1,6) co-existing. Top: initial network. Bottom: Initial augmented with edges, (3,2) and (3,4).

max-flows on an empty graph, *i.e.*, RF(1,5)=MF(1,5)=2 and RF(1,6)=MF(1,6)=2. This is possible because a single transmission of B on the edge (1,3) suffices for contributing to both RF(1,5) and RF(1,6). Thus the two flows don't compete for bandwidth on the shared link and can achieve the same capacity as the corresponding max-flows on empty networks. This is not, however, generally possible. On the top-right part of the figure we depict the situation when sending from the source to all destinations (nodes 2-6) concurrently. In this case the entire file (both A and B) has to go over links (1,2), (1,3), and (1,4) and thus RF(1,5)=RF(1,6)=1<MF(1,5)=MF(1,6)=2 leading to a real aggregate rate $\tilde{\Psi}(1) = 5$ smaller than the bound $\Psi(1) = 7$.

Generally, the bound becomes less tight with increasing link density k/n. On the bottom-left part of Figure 4·1 we add to the previous network two new links: (3,2) and (3,4). It is easy to verify that the max-flow from the source to nodes 2, 4, 5, and 6 is now 2 and to node 3 is 1, leading to $\Phi(1) = 9$. As before, if we consider only the flows to 5 and 6, it is easy to see that their max-flow values can co-exist. Considering, however, the flows to all destinations, we see that any partition of the file into parts will inevitably lead again to all real flows being 1, whereas the corresponding max-flows with the exception of MF(1,3) are now 2.[4] In other words, although the new links increased both MF(1,2) and MF(1,4) by 1 compared to the previous network, they cannot increase any of the real flows and thus widen the gap between the bound ($\Phi(1) = 9$) and the maximum attainable aggregate rate ($\tilde{\Phi}(1) = 5$).

To sum up, we propose and study these peer selection policies for the following reasons: (1) Max-flows are used to capture the fact that in a swarming protocol the chunks of a source node v_i travel towards a sink node v_j over (potentially) all the available paths of the overlay graph of point-to-point peer relationships. (2) The gap between the bound on the aggregate rate $\Psi(v_i, S)$ given by a Max-Sum peer-set and the actual maximum attainable aggregate rate $\tilde{\Psi}(v_i, S)$ which factors in the sharing of overlay links from multiple max-flows to different destinations, is reduced by the fact that swarming protocols guarantee that any chunk is transmitted at most once between any two peers; therefore, $\tilde{\Psi}(v_i, S)$ can use an overlay link multiple times (for different max-flows) but would seize bandwidth only once, thereby reducing its gap from the bound $\Psi(v_i, S)$ that assumes that the entire flow network is available to each individual max-flow from v_i. (3) The overlay network has to be rather sparse (small k) so as to limit the stress on the physical links. Thus the bound Max-Sum won't be very much off from the actual achievable aggregate rate and it makes sense optimizing the peer-set based on it. Regarding Max-Min, this is provably attainable, and optimal for broadcast rate as discussed earlier.

Since a node cares to both upload its local file to all other nodes as well as download from them all remote files, we combine the previous definitions in the following objective functions:

$$\dot{\Phi}(v_i, s_i) = \alpha \Phi(v_i, \{s_i\} + S_{-i}) + (1 - \alpha) \min_{v_j \in V_{-i}} MF(v_j, v_i, \{s_i\} + S_{-i}),$$

$$\dot{\Psi}(v_i, s_i) = \alpha \Psi(v_i, \{s_i\} + S_{-i}) + (1 - \alpha) \sum_{v_j \in V_{-i}} MF(v_j, v_i, \{s_i\} + S_{-i})$$

[4]The fact that the entire file has to go over the edge (1,3), eliminates any chance for increasing the real flows to nodes 2, 4, 5, and 6 beyond 1.

In the above functions, the parameter α regulates the relative importance between upload and download quality in selecting a peer-set. If the link capacities are symmetric, then optimizing $\dot{\Phi}$ or $\dot{\Psi}$ reduces to optimizing Φ or Ψ, independently of α.

4.3 Node Architecture

Nodes consist of the following components: a *peer selection module* implementing the peer-set selection algorithms described in Section 4.2; a *downloader module*, responsible for issuing requests to neighboring nodes and downloading missing chunks; and an *uploader module*, responsible for sending back local and in-transit chunks (an in-transit chunk is a chunk that does not belong to the local source file). In this Section we describe these three modules under the assumption that nodes are cooperative in routing each other traffic (therefore we don't need mechanisms like choke/unchoke). Later on, in Section 4.6 we discuss the necessary changes for dealing with selfishly behaving nodes. This should not lead to the conclusion that nodes are not selfishly re-wire in order to minimize the max flow to the slowest destination, or maximizing the sum of the max flows to all the destinations in the overlay. Nodes are cooperative in only routing each other traffic, and may re-wire to satisfy their abovementioned utilities.

4.3.1 Peer Selection Module

Every time period T, a node: (1) measures its available bandwidth to all other nodes using `pathChirp` [98], (2) executes a peer-set selection algorithm from Section 4.2 and connects to the corresponding nodes (incoming links are left untouched). Since both Max-Min and Max-Sum are NP-hard, we use fast local-search heuristics to compute approximately optimal peer-sets (which we verified to be always within 1% of the exact optimal for all problem sizes on which we were able to use integer linear programming to compute the latter). Once links are established, the node keeps monitoring them (including the incoming ones) and relays their capacity to all other nodes through an overlay link-state announcement protocol. Remote nodes need this information to compute their own peer-sets. Although each node measures $O(n)$ overlay links every re-wiring epoch T, the monitoring and announcement overhead is only $O(kn)$ and not $O(n^2)$ since only the $O(k)$ established links are monitored and announced in between the (infrequent) rewiring epochs, where $k \ll n$.

4.3.2 The Downloader Module

The downloader module monitors the available chunks on the peer-set and issues requests for downloading missing ones. The selection is based on the well established *Local Rarest First* (LRF) heuristic [66] that looks at the peer-set and issues a request for any missing chunk that is among the least replicated ones in the peer-set. New requests are triggered either upon the completion of a download, or if an overlay link is inactive, upon the detection on the other side of the link of a missing chunk.

4.4 The Uploader Module

The uploader receives requests and sends back chunks. Our baseline uploader allows for *up to 1 active upload (chunk) per overlay link (neighbor)*. It implements this by maintaining a FIFO queue for each overlay connection. This choice bounds the number of concurrent uploads by the number of neighbors thereby avoiding excessive fragmentation (over partitioning) of the upload bandwidth of the local (physical) access link of a node (this choice is backed-up by results appearing in [114]). We also experimented with an uploader that allows up to 1 active chunk per source file per connection, but this can lead to up $n-1$ parallel uploads per overlay link, which becomes problematic as n increases. Indeed, over-partitioning the upload bandwidth defeats the entire concept of swarming: it takes too much time to upload an entire chunk, and during this time the downloading node is under utilizing its upload bandwidth as it cannot relay the chunk before it completes the reception. We want to note, however, that our baseline design is by no means claimed to be optimal. For an example consider a node that can upload to its first $k-1$ neighbors with rate x and to the last one with rate larger than $k \cdot x$. Then as long as this last neighbor can always find k missing chunks from our node, and can also itself disseminate them further down in the network faster than the $k-1$ slow neighbors, then the system would be better off allowing up to k parallel uploads to the fast one at the expense of the slow ones. Such situations though are rather peculiar and even if they arise, it is difficult to check the necessary conditions for taking advantage of them, so we leave their investigation to future work and stick to the simple one-chunk-per-connection policy.

4.5 Performance Evaluation

In this Section we compare the performance of Max-Min and Max-Sum peer selection policies against three reference selection policies: Random (node v_i selects k peers at random from the set of all nodes in V_{-i}); k-Widest (node v_i selects node v_j if c_{ij} is among the k largest ones across all nodes in V_{-i}); Rand k-Widest (v_i performs k-Widest on a random subset of V_{-i} of size $\beta \cdot k$). Rand k-Widest is included in the evaluation to mimic the effect of combining random bootstrapping with choke/unchoke in BitTorrent. Unless otherwise noted, we used $\beta = 2$.

We compare these policies in terms of (node,remote file) *finish times*. We denote $f(j,i)$ the time that the sink v_j completes downloading the file of source v_i, assuming that all exchanges start at time 0. In all experiments we assume that nodes are fully cooperative in routing each other traffic (they belong to the same authority) and thus follow exactly and truthfully the peer-selection policies of Section 4.2 and the swarming protocol of Section 4.3 (*i.e.,* no choke/unchoke mechanism is employed). We discuss the impact of selfishly behaving nodes in Section 4.6.

Our performance evaluation is done in two settings. In the first, we assume that the n-way broadcast is to be carried over the Internet. We do so by evaluating the performance of a prototype implementation of our architecture on PlanetLab. In the second, we assume that the n-way broadcast is to be carried on a closed (controlled/isolated) network. We do so by evaluating the performance of a prototype implementation of our architecture on a discrete event simulator of the closed network.

4.5.1 Case Study I: A PlanetLab Prototype

In this setting, we compare the performance of different overlay topologies when the underlying physical network is the Internet and the overlay nodes are single-homed, *i.e.*, all overlay links of a node go over the same physical access link. For this purpose we selected $n = 15$ PlanetLab nodes. The distribution of nodes is as follows (we tried to use operationally stable and geographically diverse node set) : ten in North America (planetlab4.csail.mit.edu, planetlab2.millennium.berkeley.edu, planetlab2.utep.edu, planet lab2.acis.ufl.edu, planetlab-8.cs.princeton.edu, planetlab-2.cs.colostate.edu, planetlab5.cs. duke.edu, planetlab1.cs.northwestern.edu, planetlab3.flux.utah.edu, planetlab01.cs.washi ngton.edu), one in South America (planetlab-02.ece.uprm.edu), three in Europe (planet2. zib.de, planet2.colbud.hu, planetlab3.xeno.cl.cam.ac.uk), and one in Asia (planetlab1.net media.gist.ac.kr). Each one of the aforementioned nodes disseminated a unique 100MBytes file and allow it to connect to $k = 2$ neighbors (and accept additional incoming links). Notice that we limited our experiment to only 15 nodes and only 100MBytes per node so as to keep the amount of exchanged traffic on PlanetLab at reasonable levels, while also allowing us to monitor the network throughout the experiment. Notice that if data were to be transfered in a point-to-point manner, then it would amount to over a Terabyte for each execution of the entire experiment: 5 different peer-set selection policies, each one generating $15 \cdot 14 \cdot 100$MBytes of data at each run, and repeated 10 times to get confidence intervals (the experiment was performed between June 4th and June 30th). We let the re-wiring epoch be $T = 10$ minutes and the measurement/announcement epoch for existing links be 2 minutes. Also we set $\alpha = 0.5$ to indicate that nodes care equally for download and upload quality. In all our experiments we used `pathChirp` [98], a light, fast and accurate tool, which fits well with the PlanetLab-specific constraints, namely it does not impose a high load on PlanetLab nodes, since it does not require the transmission of long sequences of packet trains, and does not exceed the max-burst limits of PlanetLab. `pathChirp` is an end-to-end active probing tool, which requires the installation of sender and receiver module of the aforementioned tool in each node. The additional overhead of the tool in

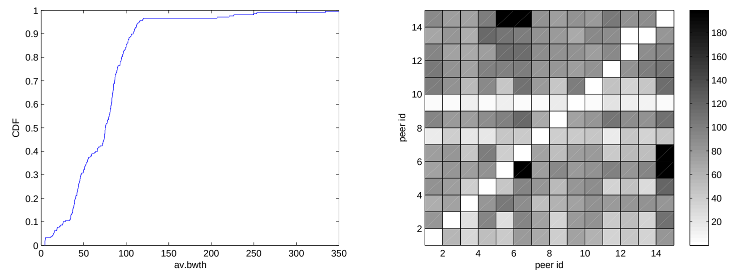


Figure 4·2: PlanetLab experiment, empirical CDF and scatter plots of available bandwidths.

terms of bandwidth consumption is negligible and does not affect the performance of the content distribution. We limited the maximum experiment duration to 10 seconds per peer (thus a full estimation for the available bandwidth from any node to all the other nodes was achieved in less than 2 minutes) and we used as available bandwidth the average available bandwidth (per peer) observed during the experiment. In Figure 4·2 we plot the cumulative distribution function (CDF) of the pairwise available bandwidth as well as the scatter plot illustrating the available bandwidth among nodes of a typical experiment in PlanetLab. The diversity of available bandwidth between peers that observed was moderate. We did not observe huge variability of the available bandwidth while performing our experiments (variability was limited to the available bandwidth among a few nodes only).

In order to perform the experiment, we modified both the client and the tracker part. We used the `mainline 4.0.2` BitTorrent client (written in `python`). We disabled the choke, unchoke and optimistic unchoke functionality and we set no limits for both the upload and download rate as well as the number of active peers. Although we are aware of the intrinsic limitations of PlanetLab as well as the PlanetLab policy of fair sharing of bandwidth among slices that use the same node, we were able to achieve very high upload and download rates (close to the estimated available bandwidth). To minimize the interaction of our experiment with other bandwidth demanding experiments, we performed the experiments after monitoring the activity of competing slices for bandwidth in the selected nodes.

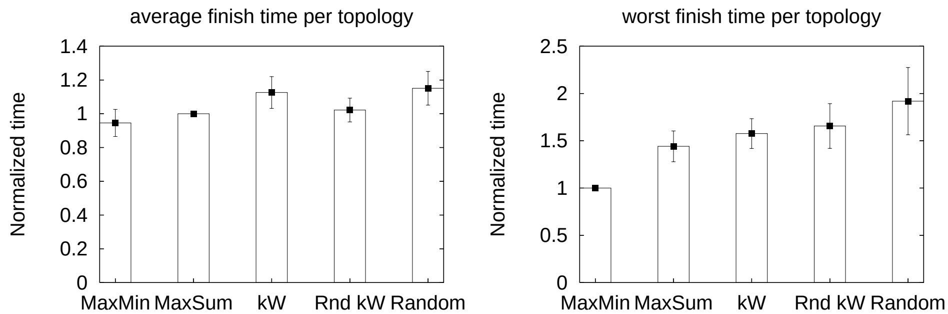


Figure 4·3: PlanetLab experiment, performance evaluation of different wiring strategies.

We used the `phpbttrkplus-2.2` BitTorrent tracker, which is a php-based tracker that maintains records about the activity of nodes (in a `mysql` database). We installed the aforementioned tracker in one of our machines (`egoist.bu.edu`), and we modified it in order to reply to requests initiated by nodes, by providing the summary of the requested peer set (ip, port, status) and not of a random peer set (as was initially designed).

For a node v_j, we compute its maximum finish time $\max(j) = \max_{i \neq j} f(j,i)$, *i.e.*, the time at which it has completed downloading *all* $n-1$ remote files, as well as its average finish time $\text{avg}(j) = 1/(n-1)\sum_{i \neq j} f(j,i)$. For peer-set selection policy $\mathcal{X}$, we let $\max(\mathcal{X}) = \max_j \max(j)$ denote its *maximum finish time* across all nodes, and $\text{avg}(\mathcal{X}) = 1/n \sum_j \text{avg}(j)$ denote its *average finish time* across all nodes.

On the left-hand-side of Figure 4·3 we present the normalized average finish time of each policy with respect to the average finish time of the Max-Sum policy. On the right-hand-side, we present the normalized maximum finish time of each policy with respect to the maximum finish time of the Max-Min policy. These results show that the various policies perform quite similarly with respect to average finish time. When looking at maximum finish times though, the picture is completely different. Max-Min manages to complete all downloads anywhere between 40% and 120% faster than the heuristics and almost 30% faster than Max-Sum. This can be very significant for Bulk Synchronous Parallel (BSP) applications [14], in which the global progress depends on the finish time of the slowest node. It is worth noting that optimizing the worst case finish time is much more difficult than optimizing the average, and thus it should come as no surprise that the heuristics perform well on average but fail to improve the worst case.

4.5.2 Case Study II: A Dedicated Network Prototype

In this setting, we examine overlay networks whose links are dedicated, meaning that they do not compete for bandwidth on the underlying physical network. This model is plausible for (multi-homed) networks set-up in support of an enterprise through the acquisition of dedicated links to connect its various locations. Such link acquisitions could be done through SLA contracts with ISPs, or through virtualization technologies such as those envisioned for GENI.[5] In either cases, a dedicated link could be set up between two enterprise nodes i and j for a given price. Any such dedicated link will have a nominal capacity c_{ij}, which may depend on any number of factors (*e.g.*, physical constraints of the underlying technology, the demand at the ISP for carrying traffic between these two locations, or the price paid for various links. Since setting up a complete network to connect all n nodes directly to each other may not be feasible (especially for systems of moderate sizes), designers of such enterprise networks are likely to construct the network so as to maximize its utility with respect to some objective function. Independent of which process/strategy is used to construct the optimized overlay, the resulting network would allow all enterprise nodes to communicate either directly or through overlay paths.

The construction we propose for optimizing the overlay for n-way broadcast proceeds as follows. First, we order the nodes according to their ids. Next, we proceed in rounds in which nodes take turns in selecting their peer-sets (as discussed in Section 4.2). This process is repeated until we converge by reaching a round that does not introduce changes in the constructed topology.[6]

---

[5] `http://geni.net`

[6] It is worth noting that the convergence of the above procedure relates to a question regarding the existence of pure Nash equilibria, and their reachability through local improvement paths, in a strategic game with Max-Min or Max-Sum as its payoff function. Although interesting from a theoretical standpoint, the question is not directly relevant here as we have assumed that nodes forward indiscriminately local and

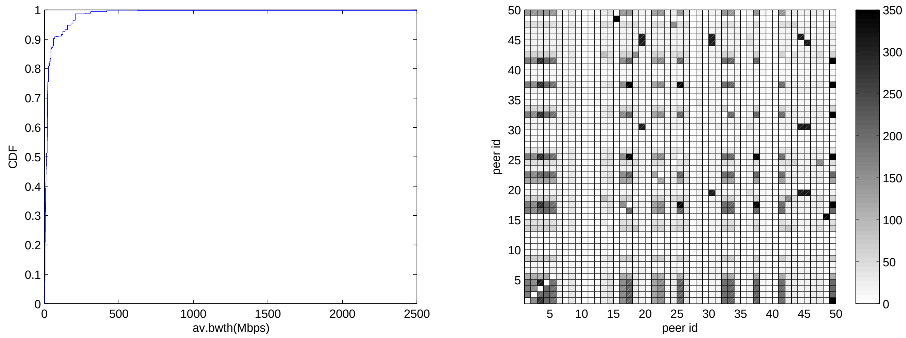


Figure 4·4: Sprint topology, empirical CDF and scatter plots of available bandwidths.

Towards our goal of evaluating the impact of various peer selection policies on the performance of n-way broadcast in this setting, we developed a discrete-event simulator that is able to run over dedicated overlay networks. We constructed the dedicated overlay (enterprise) network using the procedure described above, using the publicly available trace of Sprint's physical topology taken from Rocketfuel [107].[7] In particular, we assumed that the dedicated capacity that could be acquired from the ISP (Sprint) would reflect an "equal-share" partitioning, which we approximated as follows. We counted the number of shortest-paths (for all physical node pairs) that go over a physical link and set the available bandwidth of that link to be its real capacity divided by this number. [8] Then, for an overlay link (i, j) we set $c_{i,j}$ to be equal to the available bandwidth of the tightest physical link on the induced shortest-path over the physical topology. This produces the amount of available bandwidth that the ISP can guarantee for the new application if it admits it into its network and treats it equally with pre-existing ones. In Figure 4·4 we plot the CDF of the pairwise available bandwidth as well as the scatter plot illustrating the available

in-transit chunks. In all our experiments we got fast convergence but could also stop prematurely after a maximum number of iterations so as to deal with inexistence, unreachability, or slow convergence to stable topologies.

[7]The topology was inferred using the methodology described in [107]. The link weights we used for the shortest path algorithm are those inferred in [76]. The capacities of the links were publicly available by Sprint.

[8]The idea is that each pair of physical nodes represents a different application that is assigned an equal share of the physical capacity of all links on which it competes with other applications.

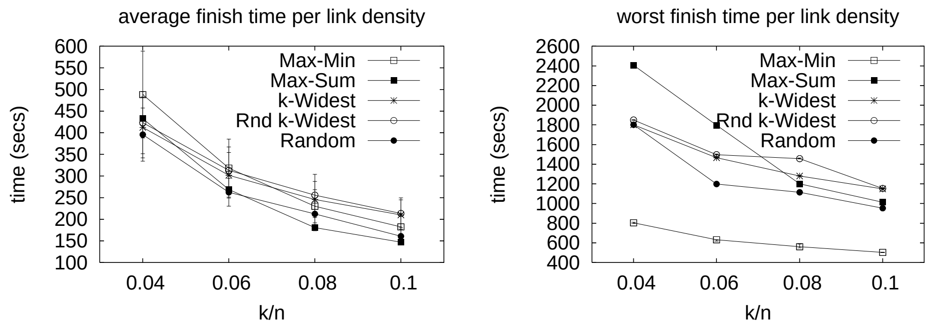


Figure 4·5: Simulation of a closed network based on Sprint's topology.

bandwidth among nodes of a typical experiment. The diversity of available bandwidth between peers is more intense (compared to the PlanetLab experiment), as there are nodes which are connected to other nodes achieving very high available bandwidth and others that are connecting achieving very low available bandwidth.

One advantage of simulations (compared to PlanetLab prototyping) is that it allows us to consider a bigger network. In particular, in the experiments that follow, we study overlays of size $n = 50$ nodes, which are randomly selected[9] from the physical Sprint network — each node holding a 500Mbytes file. As in the PlanetLab prototype, there is no notion of choke, unchoke and optimistic unchoke. The local piece selection follows a rarest first policy, there is no limit in the upload and download rate and the files are cut into 256Kbytes long chunks (that maintains blocks of 16Kbytes which is the actual transmission unit).

In Figure 4·5 we compare the average and maximum finish times of different policies for different link densities (k/n). Compared to the previous results from PlanetLab, we observe a qualitatively similar behavior. The gap, however, between Max-Min and the rest in terms of maximum finish time widens substantially: Max-Min is able to finish 2-3 times faster in this setting, even for relatively large k/n ($\sim$10%). The reason is that Max-Min has more real bandwidth to work with in this case: When it places a link (i, j),

[9]the CDF of available bandwidths for the sampled set is similar with the one when consider all the nodes of the Sprint dataset.

the capacity (both upload and download) of the two end-points increases by the capacity of the newly-added dedicated overlay link, whereas in PlanetLab the physical bandwidth is fixed, so when Max-Min places an overlay link it can only benefit by whatever unused bandwidth exists on the underlying physical network.

It is worth noticing that Max-Sum may lead to poor performance when the ratio k/n is low.[10] This is expected as the rational behind the Max-Sum wiring strategy is to maximize the average maximum flow from one node to all the other nodes. Nodes that do not contribute significantly in increasing the maximum flow are not popular, thus not a lot of connections are established (by other overlay nodes) to these nodes. As more network resources (links) are allowed to available to overlay nodes, they establish connections that do not contribute a lot in the maximum flow, improving the worst finish time. This is observed in Figure 4·5(right); the worst finish time of Max-Sum decreases significantly as the link density increases. Similar observations are made for the average finish time (see Figure 4·5(left)), although there are no significant differences among the performance of different wirings.

Another important observation is that under any wiring strategy the worst finish time of the nodes is almost identical (see Figure 4·5(right)) . This is another indication that the finish time is dominating by the slowest pieces (see also Figure 4·6). It is worth mentioning that the performance of k-Widest may be worst than the performance of Rand k-Widest, as a globally greedy selection of peers may penalize more the slowest peers than a local greedy one.

In order to characterize the graphs obtained by Max-Min and Max-Sum, we compare them with the construction where each node can guarantee the best output rate (to all the other destinations) for itself. Such a construction is rather utopian, as paths between nodes are not disjoint. Let $maxout(v)$ be the sum of the bandwidths of the node v's outgoing links (assuming that node v established k links and $n-k-1$ links are established by other

[10]This should not be confused with the discussion in Section 4.2 on the tight bound of Max-Sum under low link density.

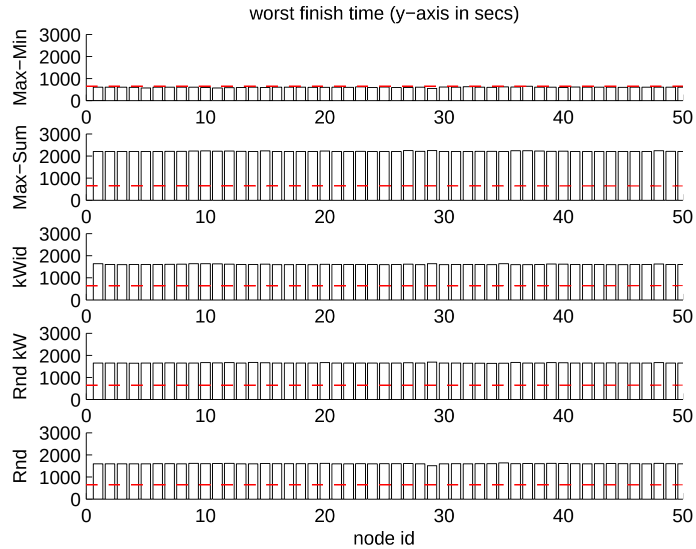


Figure 4·6: Worst finish time per node based on Sprint's topology with link density $k/n = 0.04$. The dashed line indicates the worst finish time on the Max-Min topology.

nodes). In the aforementioned graph, node v's total output rate cannot exceed:

$$s(v) \ = \ \sum_{\forall u \neq v} min(maxout(v), maxout(u))$$

Let us define the $\sum_{\forall v} s(v)$ as the *Utopian* Max-Sum social rate. Define as the *Utopian* Max-Min rate the value of

$$s^{min} \ = \ \min_{\forall u,v} (min(maxout(v), maxout(u))$$

The Max-Sum and Max-Min social rates are defined accordingly for any wiring where m_i $(n-1 \geq m_i \geq k)$ links are used by any node v_i, on a given topology.

In Figure 4·7, we illustrate the Max-Sum and Max-Min social rate obtained by the Max-Sum and the Max-Min wiring normalized by the Utopian Max-Sum and Utopian Max-Min social rate (left and right figure) respectively, for different values of link density. Both the Utopian social rates increase with link density and Max-Min social rate of the Max-Min wiring is close to the Utopian once even for low link density.

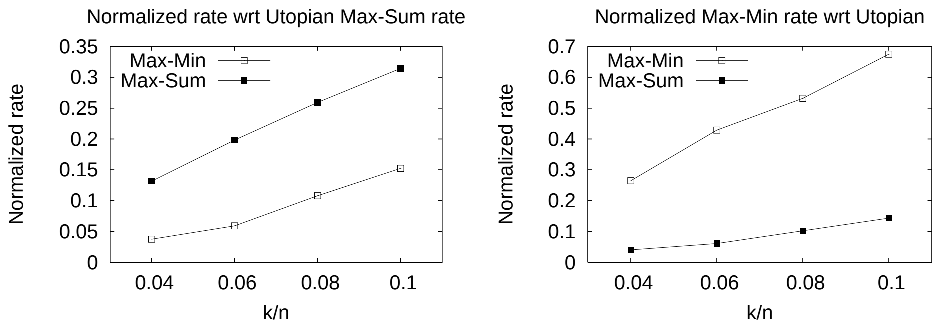


Figure 4·7: Normalized Max-Sum and Max-Min rate with respect to the Utopian one, on Sprint topology.

In Figure 4·8, we illustrate the node degree; in the x-axis nodes are ranked according to their *maxout*, *i.e.,* the node with the lowest *maxout* is ranked last for low link density (qualitatively similar observations are obtained for higher link density). As was expected, on the Max-Min topology, nodes with low *maxout* have high degree. It is worth noting, that simple heuristics like link establishment between any node with the node with the lowest *maxout* may be useful only in the extreme scenario where there is only one node with low *maxout* (as we will comment in the next section). In general the distribution of the degree of the nodes depends on the distribution of the *maxout*, thus it is difficult to construct heuristics that can work well in practice.

Turning our attention to the average and worst time needed for a document to be disseminated, we observed that the Max-Min wiring strategy has the tendency to (slightly) increase the average download time, but the decrease of the worst time of any file to be disseminated is significant. The delay is mainly due to the injection of rare pieces by the slowest node or nodes. To get a feeling of this we plot the CDF of the average and worst time needed for a document to be delivered for low link density (see Figure 4·9). Qualitatively similar observations are obtained for higher link density. Finally, an important observation, is that it seems that there are always pieces to be requested (thus the assumption of utilized parallel downloads with TCP is valid). This is consistent with observations obtained in the PlanetLab prototype. Contrary to the case of a single torrent, in multi-torrent applications

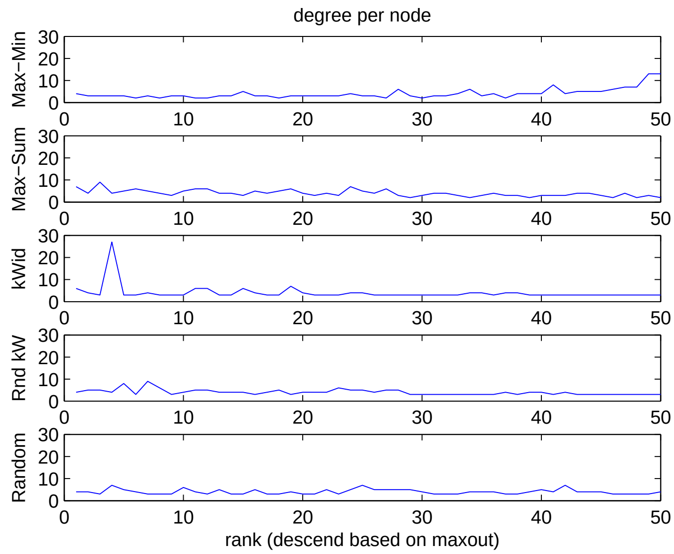


Figure 4·8: Node degree on different topologies for $k/n = 0.04$.

pieces from different files are distributed among the nodes, thus the chances that there is always a piece to forward is high.

4.5.3 Case Study III: The Effect of an Outlier

In this Section we study the case where there is a very slow node (the *maxout* of this node significantly deviates from the value of *maxout* of the other nodes) using again the Spint dataset. In Figure 4·10, we illustrate the average and worst finish time under different wiring strategies. In the presence of a very slow node, the performance of Max-Min topology is superior compared with the performance of the other wiring strategies, for worst finish time and for the average finish time for high link density. Max-Min is able to finish 3-6 times faster even for relatively large k/n.

Moreover, as it is illustrated in Figure 4·11, the average delay that is introduced for the dissemination of the documents, except the one that is uploaded by the slowest node, is negligible. On the other hand, the improvement of the worst finish time in the Max-Min topology is significant. It is worth mentioning that in the presence of a very slow node the performance of the Max-Sum can be very bad. In this setting, the performance of a simple

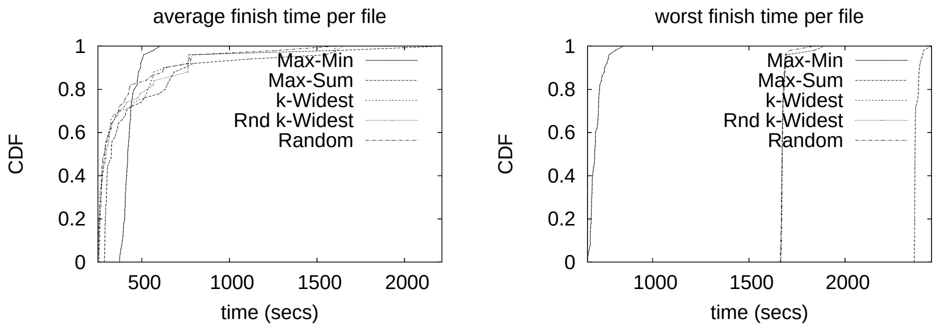


Figure 4·9: CDF of the average and worst delivery time of a file to all the nodes for link density $k/n = 0.04$.

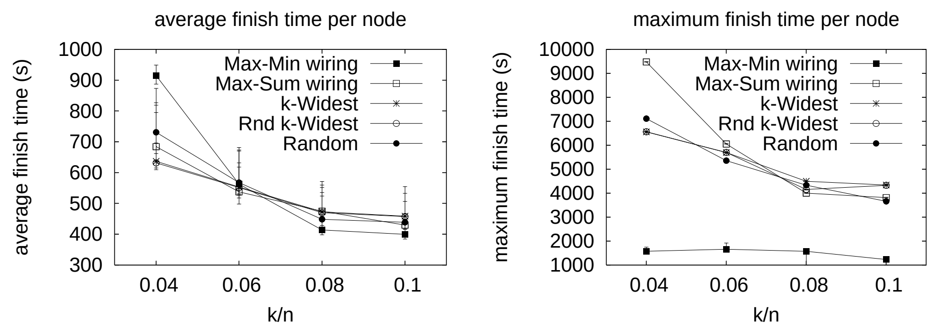


Figure 4·10: Simulation of a closed network, based on Sprint's topology, in the presence of a very slow node.

heuristic where each node establishes connections with the slowest node may improve the worst finish time (although still the Max-Min will provide the lower worst finish time as it takes into consideration the capability of each node).

4.6 Dealing with Selfish Behavior

Up to now we have assumed that nodes are fully cooperative in routing each other traffic, which is a realistic assumption for the applications enumerated in the introduction. In this Section we will try to explore ways to accommodate applications that involve selfish nodes. We will focus on the following definition of selfishness:

Definition 10 *(Upload-selfishness) An* upload-selfish *node is a node that wants to use as much of its upload capacity as possible for forwarding its local chunks and avoid "wasting" it in relaying the in-transit chunks that it holds.*

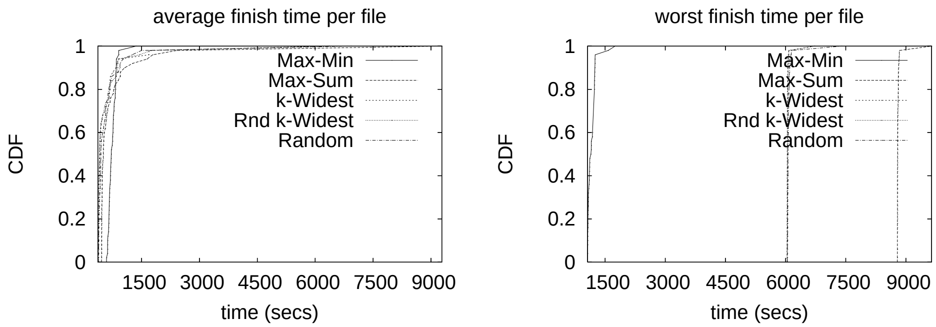


Figure 4·11: CDF for the average and worst delivery time of a file to all the nodes for link density $k/n = 0.04$, in the presence of a very slow node.

4.6.1 A Brief Taxonomy of Deterrence Mechanisms

The amount of extra benefit for an upload-selfish node (and potential harm to others) depends on the mechanisms that the network deploys for discouraging such behavior. We examine the following cases.

Case 0 (neutral): Here the network stays neutral and does not deploy any deterrence mechanism. In such a setting, the upload-selfish node could simply upload its own chunks and ignore all other requests. The harm to cooperative nodes can easily be quantified for this case, so we don't discuss it further; it will be proportional to the number of upload-selfish nodes, and cooperative nodes will be slowed down and at an extreme case will be unable to receive some files (*e.g.*, when all their neighbors are upload-selfish, which is similar to the case of an eclipse attack [105]).

Case 1 (oblivious retribution): A network can employ several retribution mechanisms to punish a node that fails to deliver a chunk after a request. The choke/unchoke [24] mechanism of BitTorrent, or modified versions based on bit-level tit-for-tat [12, 44] are two established existing proposals. Contrary to the original BitTorrent, such mechanisms are marginally useful here because they are oblivious to whether a node uploads local or in-transit chunks. An upload-selfish node will appear to be contributing by the mere fact that it is certainly uploading its own chunks. Thus oblivious strategies fail to punish nodes that "free-ride" by not uploading in-transit chunks.

Case 2 (non-oblivious retribution): Now, let's assume that there exists a non-oblivious retribution mechanism that punishes a node that fails to service requests[11] for in-transit chunks that it holds. What can a selfish node do against such mechanism? The simplest strategy is to hide (by not announcing) the availability of in-transit chunks it holds, and thus get rid of the burden of having to service requests for these chunks. This can be addressed with a simple *two-hop announcement strategy* in which a node that uploads to another node announces on its behalf the availability of the chunk (using HAVE messages [24]) to downloaders belonging to the peer-set of the receiving node. This requires obtaining upon bootstrap (and re-wiring) second hop neighbors. Assuming that the retribution is severe enough, the upload-selfish node will have to honor all requests. Despite that, the upload-selfish node still has some room to game the system by changing the uploader and the downloader as follows.

– The upload-selfish node can substitute each FIFO queue at its uploader with a *selfish FIFO* (S-FIFO) that gives priority (preemptive or non preemptive) to requests for local chunks.

– The upload-selfish node can switch from Least Replicated First to Most Replicated First (MRF) downloads. Highly replicated chunks receive fewer requests and thus reduce the "waste" of upload bandwidth for sending in-transit chunks, is smaller (most nodes already have these chunks, and any requests for these chunks will be divided over many peers).

Since it is difficult to detect such deviations from the protocol, we instead quantify their impact.

4.6.2 Quantifying the Impact of Selfish FIFO/MRF

We quantify the advantage for a single upload-selfish node by looking at the ratio between the time it takes to upload its file to all other nodes when it is selfish and when it is cooperative, granted that all other nodes are cooperative. We examined this ratio for

[11] We do not want to punish nodes that don't have enough in-transit content for whatever reason (slow local link or peer-set) but would relay if they had, so we only punish when a request exists and is not honored.

different overlays built on the Sprint trace and for different choices with respect to the choice of selfish node. We consider three cases, where the selfish node is : (1) the *slowest* node, *i.e.*, the one whose adjacent links have the minimum aggregate upload capacity; (2) the *fastest* node; or (3) a *typical* node (median upload capacity).

On the Max-Min overlay the selfish node reduced its maximum upload finish time by 30% when it was the slowest one. There is also, on average, a 15% reduction on the worst finish time of all the other nodes. When it was a typical (or the fastest one), then it got almost no benefit, since in these cases the bottleneck is at the downloading nodes (so a local selfishness behavior cannot help). In all other overlays, the selfish node got almost no benefit, even when it was the slowest node. Unlike the Max-Min, the other overlays are not optimized for the slowest node, so even if this bottleneck node tries to selfishly upload its file, it cannot really benefit because it has very limited bandwidth.

From the above, it is clear that there exist cases in which upload-selfishness pays substantially. Granted that upload-selfishness is hard to detect, we also look at its impact on the cooperative nodes. We consider again a single selfish node (one can easily extrapolate for multiple selfish nodes). The impact depends on the considered metric and on the identity of the selfish node. If we care about the worst-case download time of cooperative nodes and let the selfish node be the slowest node, then counter-intuitively, the impact on the cooperative nodes is positive. This is simply because by being selfish, the slowest node helps all other nodes improve their (bottleneck) downloads from it. To get a feeling of this we show a scatter-plot on the first row (left plot) of Figure 4·12 with the download time for each pair (node,remote file) when the topology is random and all the nodes are cooperative. The solid black line that stands out corresponds to the slowest node (node 29), whose file is the last one to be downloaded by all others. Qualitatively similar observations[12] are obtained for any other wiring strategy except the Max-Min one (see the first two rows of Figure 4·12). To contrast this, we plot in the last row the corresponding times when the topology is Max-Min (left plot) and when the topology is Max-Min with the slowest

[12]Note that the slowest node may not be the same among different topologies.

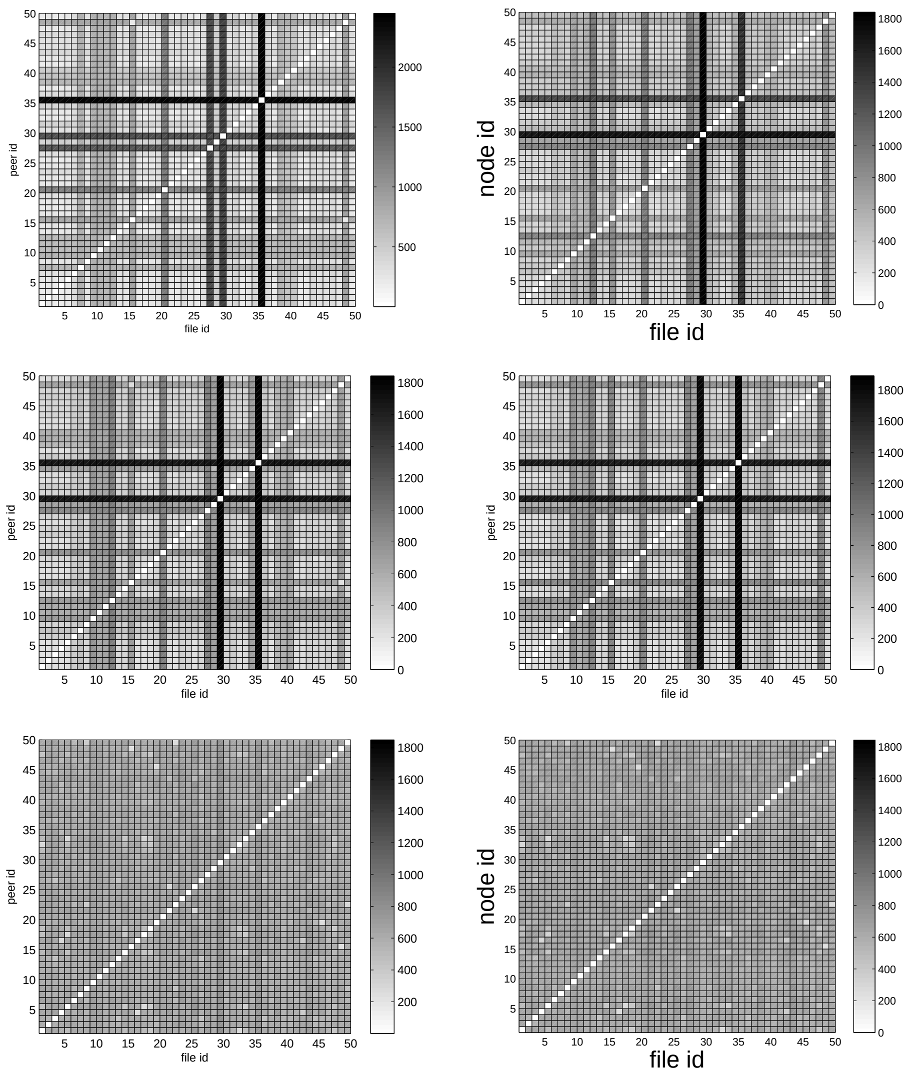


Figure 4·12: Maximum finish time for all nodes and all files under different wirings (from left to right, top to bottom): MaxSum, Random, k-Widest, Rnd k-Widest, MaxMin strategies with non upload-selfish nodes and MaxMin with upload-selfish slowest node (k/n=0.04).

node is upload-selfish (right plot). A first observation is that the Max-Min topology does a pretty good job at smoothing out the differences in the maximum finish times with slight increase on the average finish time (note that some cells may be darker on the Max-Min topology compared to the corresponding cells on other topologies). As it can be seen, the combination of Max-Min topology and upload-selfish scheduling on the slowest node (see last row, right plot) does even a better job at smoothing out the differences in maximum finish times. If, on the other hand, the selfish node is a typical node, or the fastest node, then its effect on the download quality of others is rather marginal. First, its own file is not a bottleneck. Second, the relay of in-transit chunks is largely carried by the other $n-1$ nodes. Third, S-FIFO and MRF impact primarily first-hop neighbors and have small impact on nodes further away. Qualitatively similar observations are obtained even in the presence of a very slow node (node 44), as it is illustrated in Figure 4·13.

Overall, upload-selfishness, unlike its name suggest, is not necessarily bad. A socially inclining global scheduling policy, for example, would certainly make slow nodes upload only their own chunks so as to reduce the severity of the bottlenecks that they cause. More generally, for social optimality, one should split the upload bandwidth of a node between local and in-transit chunks according to the relative speed of the node. Nodes who are fast should contribute heavily in relaying in-transit chunks. Nodes who are slow, should focus only on uploading their own chunks so as to avoid becoming bottleneck points. Stated differently, *a single uploading policy across all nodes cannot be socially optimal*. We postpone the investigation of node-dependent upload scheduling for future work (see Section VI of [61] for a similar discussion based on our previous work on selfish caching).

4.6.3 Download-Selfishness

It is tempting to ask whether a notion of *download-selfishness* would make sense. Our answer leans towards the negative. First, there is no contention between local and in-transit chunks in the incoming direction towards a node — only in-transit chunks flow there. Second, as long as the downloader keeps all its overlay connections busy by immediately identifying and requesting missing chunks, its download-finish time will be the same, so it gets no foreseeable benefit by deviating from LRF. Finally, trying to manipulate the system by advertising false c_{ij}'s for the established links can be disclosed by having nodes periodically "audit" others by measuring some remote c_{ij}'s and comparing with the advertised values on the link-state protocol. Such methods are quite elaborate and fall outside the scope of the current work.

4.7 Chapter Summary

In this Chapter we showed that swarming protocols for bulk data transfers perform much better when operating over optimized overlay topologies that take into consideration the end-to-end performance characteristics of the underlying network. Such topologies improve the aggregate transmission capacity of nodes, but where they make a huge difference compared to existing heuristic approaches, is on relieving bottleneck points. Random and myopic heuristics used in practice lack the required sophistication for overcoming such bottlenecks.

Our optimized topologies are oblivious to the details of the swarming protocol that runs on top. They leverage the available bandwidth of the underlying network and abstract the swarming protocol by viewing it as a series of max-flows. Thus they can benefit a variety of swarming protocols with different upload/download scheduling characteristics. Since our topologies are data-blind, it is the job of the swarming protocol to make the best use of the end-to-end bandwidth that they offer. To that end, we have shown that a commonly parametrized swarming protocol is far from being optimal.

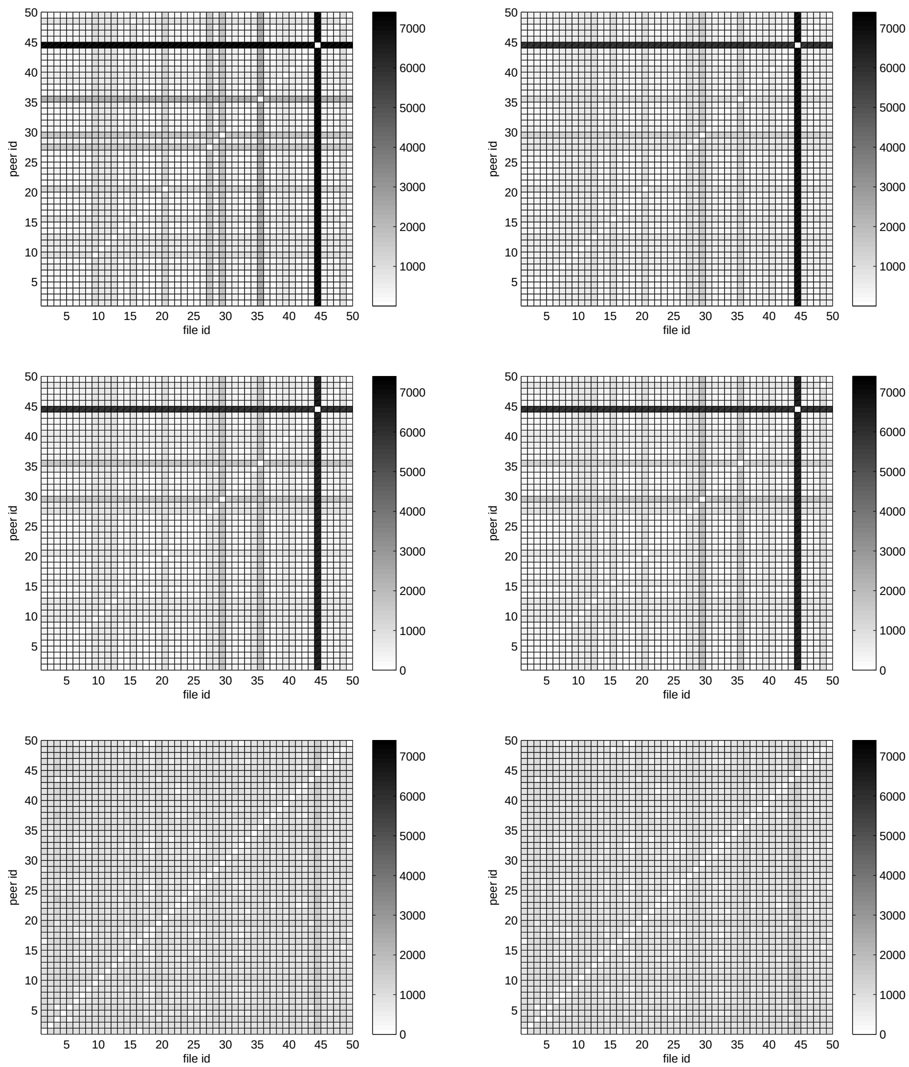


Figure 4·13: Maximum finish time for all nodes and all files under different wirings (from left to right): MaxSum, Random, k-Widest, Rnd k-Widest, MaxMin strategies with non upload-selfish nodes and Max-Min with upload-selfish slowest node, in the presence of a very slow node (k/4=0.04).

Chapter 5

Distributed Server Migration

Selfish wiring strategies have significant implications to service provisioning. A selfish overlay node would connect to the server that provides the best service, while at the same time would use its demand to bias the deployment of the service toward locating more servers in its network vicinity. On the other hand, a service provider, would like to deploy a network of servers to satisfy its clients demand, while keeping deployment cost minimal by following a pay-as-you-go model.

In this Chapter, we examine the use of distributed and scalable approaches that enable a provider to determine the number and location of servers for optimal delivery of content or service to its selfish users. The classical, centralized service provisioning approach requires knowledge of a-priori demand and topological information, both of which are intractable to obtain in large-scale dynamic networks. To leverage recent advances in virtualization technologies [90] we develop and evaluate a distributed protocol to migrate servers based on end-users demand and only local topological knowledge. Results under a range of workloads and network topologies suggest that the performance of the proposed approach is comparable to that of the optimal but unscalable centralized one. Surprising, our results indicate that selfish wiring on behalf of end-users can be a powerful tool towards fully decentralized overlay network deployment.

Imagine a large-scale bandwidth/processing-intensive service such as the real-time distribution of software updates and patches [39], a distributed data-center [60], a cloud computing platform [38, 28], *etc.* Such services must cope with the typically *voluminous* and *bursty* demand — both in terms of overall load and geographical distribution of the sources of demand — due to recently observed flash-crowd phenomena. To deploy such services, decisions must be made on: (1) the location, and optionally, (2) the number of nodes (or hosting infrastructures) used to deliver the service. Two well-known formulations of classic *Facility Location Theory* [83] can be used as starting points for addressing decisions (1) and (2), respectively: The *uncapacitated k-median (UKM)* problem prescribes the locations for instantiating a fixed number of service facilities so as to minimize the distance between users and the closest facility capable of delivering the service. In the *uncapacitated facility location (UFL)* problem, the number of facilities is not fixed, but *jointly* derived along with the locations as part of a solution that minimizes the combined service hosting and access costs. For the rest of the Section we will use the terms facility and server interchangeably.

Limitations of existing approaches: Even though it provides a solid basis for analyzing the fundamental issues involved in the deployment of network services, facility location theory is not without its limitations. First and foremost, proposed solutions for UKM and UFL are centralized, so they require the gathering and the transmission of the entire topological and demand information to a central point, which is not possible (not to mention practical) for large networks. Second, such solutions are not adaptive in the sense that they do not allow for easy reconfiguration in response to changes in the topology and the intensity of the demand for service. To address these limitations we propose distributed versions of UKM and UFL, which we use as means of constructing an automatic service deployment scheme.

A scalable approach to automatic service deployment: We develop a scheme in which an initial set of service facilities are allowed to migrate adaptively to the best network locations, and optionally to increase/decrease in number so as to best service the current demand. Our scheme is based on developing distributed versions of the UKM problem (for the case in which the total number of facilities must remain fixed) and the UFL problem (when additional facilities can be acquired at a price or some of them be closed down). Both problems are combined under a common framework with the following characteristics: An existing facility gathers the topology of its immediate surrounding area, which is defined by an r-*ball* of neighbors – nodes that are within a *radius* of r hops from the facility. The facility also monitors the demand that it receives from the nodes that have it as closest facility. It keeps an exact representation of demand from within its r-ball, and an approximate representation for all the nodes on the *ring* of its r-ball (nodes outside the r-ball that receive service from it). In the latter case, the demand of nodes on the *"skin"* of the r-ball is increased proportionally to account for the aggregate demand that flows in from outside the r-ball through that node. When multiple r-balls intersect, they join to form more complex r-*shapes*. The observed topology and demand information is then used to re-optimize the current location (and optionally the number of) facilities by solving the UKM (or the UFL) problem in the vicinity of the r-shape.

The trade-off between scalability and performance: Reducing the radius r decreases the amount of topological information that needs to be gathered and processed centrally at any point (*i.e.,* at facilities that re-optimize their positions). This is a plus for scalability. On the other hand, reducing r harms the overall performance as compared to centralized solutions that consider the entire topological information. This is a minus for performance. We examine this trade-off experimentally using synthetic (Erdös-Rényi [32] and Barabási-Albert [8]) and real (AS-level [112]) topologies. We show that even for very small radii, *e.g.,* $r = 1$ (*i.e.,* facility migration is allowed only to first-hop neighbors), or $r = 2$ (*i.e.,* facility migration is allowed only up to second-hop neighbors), the performance of the distributed approach tracks closely that of the centralized one. Thus, increasing r much more is not necessary for performance, and might also be infeasible since even for relatively small r, the number of nodes contained in an r-shape increases very fast (owing to the small, typically $O(\log n)$, diameter of most networks, including the aforementioned ones).

A case study — large-scale timely distribution of customized software: Consider a large scale software update system, similar to that used for *Microsoft Windows Update.*[1] Such a system not only delivers terabytes of data to millions of users, but also it has to incorporate complex decision processes for customizing the delivered updates to

[1] `http://update.microsoft.com`

the peculiarities of different clients [39] with respect to localization, previously-installed updates, compatibilities, and optional components, among others. This complex process goes beyond the dissemination of a single large file, where a peer-to-peer approach is an obvious solution [55]. Moreover, it is unlikely that software providers will be willing to trust intermediaries with such processes. Rather, we believe that such applications are likely to rely on dedicated or virtual hosts, *e.g.,* servers offered for lease through third-party overlay networks – *a la* Akamai or Planet Lab, or the newest breed of Cloud Computing platforms (*e.g.,* Amazon EC2[2]). To that end, we believe that the use of our distributed facility location approach presents significant advantages in terms of optimizing the cost and efficiency of deploying such applications.[3] In the remainder of this section, we provide a mapping from the aforementioned software distribution service to our abstract UKM and UFL problems.

Service providers, hosts, and clients: We envision the availability of a set of network hosts upon which specific functionalities may be installed and instantiated on demand. We use the term "Generic Service Host" (GSH) to refer to the software and hardware infrastructure necessary to host a service. For instance, a GSH could be a well-provisioned Linux server, a virtual machine (VM) slice similar to that used in Planet Lab[4] or that envisioned in GENI[5], or a set of resources in a Cloud Computing platform (*e.g.,* an Amazon Machine Image (AMI) in the context of EC2).

A GSH may be in Working (W) or Stand-By (SB) mode. In W mode, the GSH constitutes a service facility that is able to respond to client requests for service, whereas in SB mode, the GSH does not offer the actual service, but is ready to switch to W if it is so directed.[6] Thus the set of facilities used to deliver a service is precisely the set of GSHs in

[2] `http://aws.amazon.com/ec2`

[3] It is important to note that the large-scale timely distribution of customized content is hardly unique to the dissemination of software updates, as it could be equally instrumental for "Virtual Product Placement" in live content as well as in video-on-demand services, to mention two examples.

[4] `http://www.planet-lab.org`

[5] `http://www.geni.net/GDD/GDD-06-08.pdf`

[6] Switching to W might involve the transfer of executable and configuration files for the service from other GSHs or from the service provider.

W mode. By switching back and forth between W mode and SB mode, the *number* as well as the *location* of facilities used to deliver the service could be controlled in a distributed fashion. In particular, a GSH in W mode (*i.e.*, a facility) monitors the topology and the corresponding demand in its vicinity and is thus capable of re-optimizing the location of the facility.

Third-party Autonomous Systems (AS) may host the GSHs of service providers, possibly for a fee.[7] In particular, the hosting AS may charge the service provider for the assets it dedicates to the GSHs, including the software/hardware infrastructure supporting the GSHs as well as the bandwidth used to carry the traffic to/from GSHs in W mode.

The implementation of the above-sketched scenarios requires each GSH to be able to construct its surrounding AS-level topology up to a radius r. This can be achieved through standard topology discovery protocols.[8] We also assume that a client is selfish thus is able to locate the facility closest to it and can be informed by a GSH of the service regarding GSH's W or SB status. Both of these could be easily achieved with simple standard resource discovery mechanisms like server selection [17], DNS re-direction [91, 33] (appropriate for application-level realizations of our distributed facility location approach), proximity-based or density-based anycast routing [67] (appropriate for network layer realizations). Furthermore, we show in Section 5.6 that the performance of our scheme degrades gracefully as re-direction becomes more imprecise.

5.1 Background

Let $G = (V, E)$ represent a network defined by a node set $V = \{v_1, v_2, \ldots, v_n\}$ and an undirected edge set E. Let $d(v_i, v_j)$ denote the length of a shortest path between v_i and v_j, and $s(v_j)$ the (user) service demand originating from node v_j. Let $F \subseteq V$ denote a set of facility nodes – *i.e.,* nodes on which the service is instantiated. If the number of available

[7] Notice that each AS (or a smaller organizational unit therein) is also a client of the service, with demand proportional to the aggregate number of requests originating from its end-users (*e.g.,* number of downloads of a service pack).

[8] `http://www.caida.org/tools/measurement/skitter`

facilities $k = |F|$ is given, then the specification of their exact locations amounts to solving the following uncapacitated k-median problem:

Definition 11 *(UKM) Given a node set V with pair-wise distance function d and service demands $s(v_j)$, $\forall v_j \in V$, select up to k nodes to act as medians (facilities) so as to minimize the service cost $C(V, s, k)$:*

$$C(V, s, k) = \sum_{\forall v_j \in V} s(v_j) d(v_j, m(v_j)), \tag{5.1}$$

where $m(v_j) \in F$ is the median that is closer to v_j.

On the other hand, if instead of k, one is given the costs $f(v_j)$ for setting up a facility at node v_j, then the specification of the facility set F amounts to solving the following uncapacitated facility location problem:

Definition 12 *(UFL) Given a node set V with pair-wise distance function d and service demands $s(v_j)$ and facility costs $f(v_j)$, $\forall v_j \in V$, select a set of nodes to act as facilities so as to minimize the joint cost $C(V, s, f)$ of acquiring the facilities and servicing the demand:*

$$C(V, s, f) = \sum_{\forall v_j \in F} f(v_j) + \sum_{\forall v_j \in V} s(v_j) d(v_j, m(v_j)), \tag{5.2}$$

where $m(v_j) \in F$ is the facility that is closer to v_j.

For general graphs, both UKM and UFL are NP-hard problems [53]. A variety of approximation algorithms have been developed under metric distance using a plethora of techniques, including rounding of linear programs [21], local search [54, 6], and primal-dual methods [49].

There is a huge literature on facility location theory. Initial results are surveyed in the book by Mirchandani and Francis [83]. A large number of subsequent works focused on developing centralized approximation algorithms [21, 54, 6, 49]. The authors of [16] have proposed an alternative approach for approximating facility location problems based on a continuous "high-density" model. Recently, generalizations of the classical centralized facility location problem have appeared in [77, 37]. The first mention of distributed facility location seems to have been from Jain and Vazirani [49] while commenting on their primal-dual approximation method, but they do not pursue the matter further. To the best of our knowledge, the only work in which distributed facility location has been the focal point seems to be the recent work of Moscibroda and Wattenhofer [85]. This work, however, is mostly focused on deriving worst-case performance bounds for distributed facility location. It is based on primal-dual techniques that are amenable to such analysis, but which are too complicated for practical implementation purposes, as compared to our work. Furthermore, [85] does not include any experimental results or implementation guidelines of practical purposes. The online version of facility location, in which request arrive one at a time according to an arbitrary pattern, has been studied by Meyerson [82] that gave a randomized online $O(1)$-competitive algorithm for the case that requests arrive randomly and a $O(\log n)$-competitive algorithm for the case that arrival order is selected by an adversary. Oikonomou and Stavrakakis [87] have proposed a fully distributed approach for service migration — their results, however, are limited to a single facility (representing a unique service point) and assume tree topologies.

Several application-oriented approaches to distributed service deployment have appeared in the literature, *e.g.,* Yamamoto and Leduc [117] (deployment of multicast reflectors), Rabinovich and Aggarwal [95] (deployment of mirrored web-content), Chambers et al. [20] (on-line multi-player network games), Cronin et al. [27] (constrained mirror placement), and Krishnan et al. [56] (cache placement). The aforementioned works are strongly tied to their specific applications and do not have the underlying generality offered by the distributed facility location approach adopted in our work. Relevant to our work are also the works of Oppenheimer et al. [88] on systems aspects of a distributed shared platform for service deployment, and Loukopoulos et al. [74] on the overheads of updating replica placements under non-stationary demand.

5.2 A Limited Horizon Approach to Distributed Facility Location

In this section we develop distributed versions of UKM and UFL by utilizing a natural limited horizon approach in which facilities have exact knowledge of the topology of their r-ball (surrounding topology up to r-hop neighbors), exact knowledge of the demand of each node in their r-ball and approximate knowledge of the aggregate demand from nodes on the ring surrounding their r-ball. Our distributed approach will be based on an iterative method in which the location and the number of facilities (in the case of UFL only) may change between iterations.

5.2.1 Definitions

We make use of the following definitions, most of which are superscripted by m, the ordinal number of the current iteration. Let $F^{(m)} \subseteq V$ denote the set of facility nodes at the mth iteration. Let $V_i^{(m)}$ denote the *r-ball* of facility node v_i, *i.e.,* the set of nodes within radius r from v_i. Let $U_i^{(m)}$ denote the *ring* of facility node v_i, *i.e.,* the set of nodes not contained in $V_i^{(m)}$, but are being served by facility v_i, or equivalently, the nodes that have v_i as their closest facility. The *domain* $W_i^{(m)} = V_i^{(m)} \bigcup U_i^{(m)}$ of a facility node consists of its r-ball and the surrounding ring.

From the previous definitions it is easy to see that $V = V^{(m)} \bigcup U^{(m)}$, where $V^{(m)} = \bigcup_{v_i \in F^{(m)}} V_i^{(m)}$, $U^{(m)} = \bigcup_{v_i \in F^{(m)}} U_i^{(m)}$.

5.2.2 The Distributed Algorithm

Our distributed algorithm starts with an arbitrary initial batch of facilities, which are then refined iteratively through relocation and duplication until a (locally) optimal solution is reached. It includes the following steps:

Initialization: Pick randomly an initial set $F^{(0)} \subseteq V$ of $k_0 = |F^{(0)}|$ nodes to act as facilities. Let $\mathcal{F} = F^{(0)}$ denote a temporary variable containing the "unprocessed" facilities from the current batch. Also, let $\mathcal{F}^- = F^{(0)}$ denote a variable containing this current batch of facilities.

Iteration m: Pick a facility $v_i \in \mathcal{F}$ and process it by executing the following steps:

1. Construct the topology of its surrounding r-ball by using an appropriate neighborhood discovery protocol (see [78] for such an example).

2. Test whether its r-ball can be merged with the r-balls of other nearby facilities. We say that two or more facilities can be merged (to actually mean that their r-balls can be merged), when their r-balls intersect, *i.e.,* when there exists at least one node that is within distance r from all the facilities . Let $J \subseteq F^{(m)}$ denote a set composed of v_i and the facilities that can be merged with it.[9] J induces an r-*shape* $G_J = (V_J, E_J)$, defined as the sub-graph of G composed of the facilities of J, their neighbors up to distance r, and the edges between them. We can place constraints on the maximal size of r-shapes to guarantee that it is always much smaller than $O(n)$.

3. Re-optimize the r-shape G_J. If the original problem is UKM, solve the $|J|$-median within the r-shape — this can produce new locations for the $|J|$ facilities. If the original problem is UFL, solve the UFL within the r-shape — this can produce new locations as well as change the number of facilities (make it smaller or larger than $|J|$). In both cases the re-optimization is conducted by using a centralized algorithm.[10] The details regarding the optimization of r-shapes are given in Section 5.2.3.

4. Remove processed facilities, both the original v_i and the ones merged with it, from the set of unprocessed facilities of the latest batch, *i.e.,* set $\mathcal{F} = \mathcal{F} \setminus (J \bigcap \mathcal{F}^-)$. Also update $F^{(m)}$ with the new locations of the facilities after the re-optimization.

5. Test for convergence. If $\mathcal{F} \neq \emptyset$ then some facilities from the latest batch have not yet been processed, so perform another iteration. Otherwise, if the configuration of facilities changed with respect to the initial one for the latest batch, *i.e.,* $F^{(m)} \neq \mathcal{F}^-$,

[9] The merging operation is recursive. When an initial r-ball merges with a second one, then additional facilities that can merge with the second one merge as well, and so on.

[10] The numerical results presented in Sections 5.4 and 5.5 are obtained by using Integer Linear Programming (ILP) formulations [83] and local-search heuristics [6] for solving UKM and UFL within r-shapes. Since both perform very closely in all our experiments, we don't discriminate between the two.

then form a new batch by setting $\mathcal{F} = F^{(m)}$ and $\mathcal{F}^{-} = F^{(m)}$, and perform another iteration. Else (if $F^{(m)} = \mathcal{F}^{-}$), then no beneficial relocation or elimination is possible, so terminate by returning the (locally) optimal solution $F^{(m)}$.

5.2.3 Optimizing r-shapes

As discussed in Section 5.1, the input of a UKM problem is defined completely by a tuple $\langle V, s, k\rangle$, containing the topology, the demand, and the number of allowed medians. A UFL problem is defined by a tuple $\langle V, s, f\rangle$, similar to the previous one, but with facility creation costs instead of a fixed constraint on the number of allowed facilities. For the optimization of an r-shape, we set:

- $V = V_J$, and
- $k = |J|$, for the case of UKM, or $f = \{f(v_j) : \forall v_j \in V_J\}$, for the case of UFL.

Regarding service demand, a straightforward approach would be to set $s = \{s(v_j) : \forall v_j \in V_J\}$, *i.e.,* retain in the re-optimization of the r-shape the original demand of the nodes of the r-shape. Such an approach would, nonetheless, be inaccurate since the facilities within an r-shape service the demand of the nodes of the r-shape, *as well as those in the corresponding ring of the r-shape.* Since there are typically a few facilities, each one has to service a potentially large number of nodes (*e.g.,* of order $O(n)$), and thus the rings are typically much larger than the corresponding r-shapes.[11] *Re-optimizing the arrangement of facilities within an r-shape without considering the demand that flows-in from the ring would, therefore, amount to disregarding too much information (as compared to the information considered by a centralized solution).* Including the nodes of the ring into the optimization is, of course, not an option, as the ring can be arbitrarily large ($O(n)$) and, therefore, considering its topology would contradict our prime objective — to perform facility location in a scalable, distributed manner.

[11] Notice that r is intentionally kept small to limit the size of the individual re-optimizations.

Our solution for this issue is to *consider the demand of the ring implicitly by mapping it into the local demand of the nodes that constitute the* skin *of the r-shape*. The skin consists of nodes on the border (or edge) of the r-shape, *i.e.,* nodes of the r-shape that have direct links to nodes of the ring. This intermediate approach bridges the gap between absolute disregard for the ring, and full consideration of its exact topology. The details of the mapping are as follows. Let v_i denote a facility inside an r-shape G_J. Let $v_j \in U$ denote a node in the corresponding ring, having the property that v_i is v_j's closest facility. Let v_k denote a node on the skin of G_J, having the property that v_k is included in a shortest path from v_j to v_i. To take into consideration the demand from v_j while optimizing the r-shape G_J, we map that demand onto the demand of v_k, *i.e.,* we set: $s(v_k) = s(v_k) + s(v_j)$.

5.3 A More Detailed Examination of Distributed Facility Location

The previous section has provided an overview of the basic characteristics of the proposed distributed facility location approach. The section goes beyond that to look closer at some important albeit more complex properties of the proposed solution.

5.3.1 Convergence of the Iterative Method

We start with the issue of convergence. First we show that the iterative algorithm of Section 5.2.2 converges in a finite number of iterations. Then we show how to control the convergence speed so as to adapt it to the requirements of practical systems.

Proposition 3 *The iterative local search approach for distributed facility location converges in a finite number of iterations.*

Proof: Since the solution space is finite, it suffices to show that there cannot be loops, *i.e.,* repeated visits to the same configuration of facilities. A sufficient condition for this is that the cost (either Equation (5.1) or (5.2) depending on whether we are considering distributed UKM or UFL) be monotonically decreasing between successive iterations, *i.e.,* $c^{(m)} \geq c^{(m+1)}$. Below, we show that this is the case for the UKM applied to r-shapes with

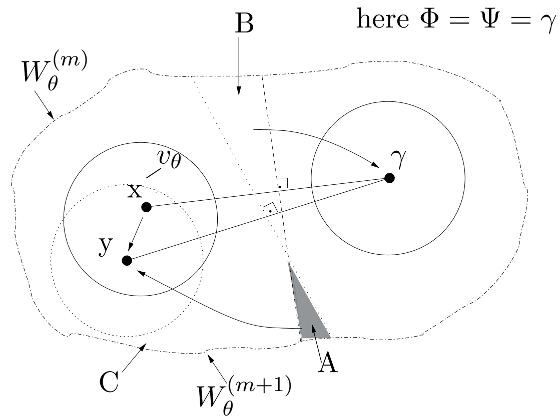


Figure 5·1: Depiction of the move of a facility from X to Y and of the sets A, B, and C.

a single facility. The cases of UKM applied to r-shapes with multiple facilities, and of UFL follow from straightforward generalizations of the same proof.

Suppose that during iteration $m+1$ facility v_θ is processed and that between iteration m and $m+1$, v_θ is located at node x, whereas after iteration $m+1$, v_θ is located at node y (see also Figure 5·1). If $x \equiv y$, then $c^{(m)} = c^{(m+1)}$. For the case that $x \neq y$, we need to prove that $c^{(m)} > c^{(m+1)}$.

For the case in which $W_\theta^{(m)} \equiv W_\theta^{(m+1)}$, it is easy to show that $c^{(m)} > c^{(m+1)}$. Indeed, since the facility moves from x to y it must have been that this reduces the cost of the domain of v_θ, *i.e.,* $c(W_\theta^{(m)}) > c(W_\theta^{(m+1)})$, which implies $c^{(m)} > c^{(m+1)}$, since no other domain is affected.

The case in which $W_\theta^{(m)} \neq W_\theta^{(m+1)}$ is somewhat more involved. It implies that there exist sets of nodes A, B: $A \cup B \neq \emptyset$, $A = \{z \in V : z \notin W_\theta^{(m)}, z \in W_\theta^{(m+1)}\}$ and $B = \{z \in V : z \in W_\theta^{(m)}, z \notin W_\theta^{(m+1)}\}$. A is actually the set of nodes that were not served by facility v_θ before the $m+1$ iteration and are served after the $m+1$ iteration. Similarly, B is the set of nodes that were served by facility v_θ before the $m+1$ iteration and are not served after the $m+1$ iteration. Let $C = \{z \in V : z \in W_\theta^{(m)}, z \in W_\theta^{(m+1)}\}$ be the set of nodes that remained in the domain of v_θ after its move from x to y (Figure 5·1 depicts the aforementioned sets). Since $W_\theta^{(m)} = B \cup C$ (B, C disjoint) and the re-optimization of $W_\theta^{(m)}$ moved the facility v_θ from x to y, it must be that:

$$c(B, x) + c(C, x) > c(B, y) + c(C, y) \tag{5.3}$$

where $c(B, x)$ denotes the cost of servicing the nodes of B from x (similar definitions for $c(C, x)$, $c(C, y)$).

Let Φ denote the set of facilities that used to service the nodes of A before they entered the domain of v_θ at $m + 1$. Similarly, let Ψ denote the set of facilities that get to service the nodes of B after they leave the domain of v_θ at $m + 1$. From the previous definitions it follows that:

$$c(A, y) < c(A, \Phi) \tag{5.4}$$

$$c(B, y) > c(B, \Psi) \tag{5.5}$$

Using Equation (5.5) in Equation (5.3) we obtain:

$$c(B, x) + c(C, x) > c(B, \Psi) + c(C, y) \tag{5.6}$$

Applying Equations (5.6) and (5.4) to the difference $c^{(m)} - c^{(m+1)}$, we can now show the following:

$$\begin{aligned}
&c^{(m)} - c^{(m+1)} = \\
&\Big(c(B, x) + c(C, x) + c(A, \Phi)\Big) - \Big(c(A, y) + c(C, y) + c(B, \Psi)\Big) = \\
&\Big(c(B, x) + c(C, x) - c(B, \Psi) - c(C, y)\Big) + \Big(c(A, \Phi) - c(A, y)\Big) > 0
\end{aligned}$$

which proves the claim also for the $W_\theta^{(m)} \neq W_\theta^{(m+1)}$ case, thus completing the proof. ■

We can control the convergence speed by requiring each turn to reduce the cost by a factor of α, in order for the turn to be accepted and continue the optimizing process; *i.e.,* accept the outcome from the re-optimization of an r-shape at the mth iteration, only if $c^{(m)} \geq (1 + \alpha)c^{(m+1)}$. In this case, the following proposition describes the convergence speed.

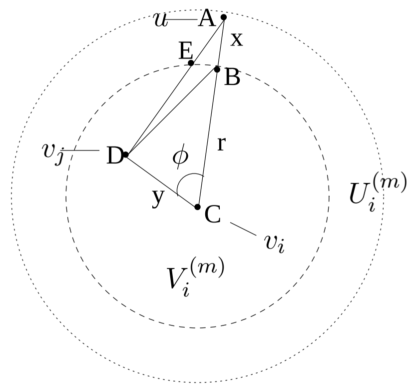


Figure 5·2: Example of a possible facility movement from node v_i to node v_j with respect to a particular node $u \in U_i$.

Proposition 4 *The iterative local search approach for distributed facility location converges in $O(\log_{1+\alpha} n)$ steps.*

Proof: Let $c^{(0)}$, $c^{(M)}$, c^* denote the initial cost, a locally minimum cost obtained at the last (Mth) iteration, and the minimum cost of a (globally) optimal solution, respectively. Here we consider M to be the number of "effective" iterations, *i.e.,* ones that reduce the cost by the required factor. The total number of iterations can be a multiple of M up to a constant given by the number of facilities. Since we are interested in asymptotic complexity we can disregard this and focus on M.

For $m < M$ we have required that $c^{(m)} \geq (1+\alpha)c^{(m+1)}$, or equivalently, $c^{(0)} \geq (1+\alpha)^m c^{(m)}$. Thus when the iteration converges we have:

$$c^{(0)} \geq (1+\alpha)^M c^{(M)} \Rightarrow M \leq \log_{1+\alpha} \frac{c^{(0)}}{c^{(M)}} \leq \log_{1+\alpha} \frac{c^{(0)}}{c^*} \tag{5.7}$$

Given the definition of the cost and the fact that node service demands ($s(v)$'s) are constants with respect to the size of the input (n), it is easy to see that $c^{(0)}$ can be upper bounded by $O(n^2)$ and c^* be lower bounded by $\Omega(n)$. This leads to an $O(n)$ upper bound for $\frac{c^{(0)}}{c^*}$. Substituting in Equation (5.7) gives the claimed upper bound for the number of iterations. ■

5.3.2 The Mapping Error and its Effect on Local Optimizations

In this section we discuss an important difference between solving a centralized version of UKM or UFL (Definitions 11, 12) applied to the entire network and our case where these problems are solved within an r-shape based on the demand that results from a fixed mapping of the ring demand onto the skin. In the centralized case, the amount of demand generated by a node is not affected by the particular configuration of the facilities within the graph, since all nodes in the network are included and considered with their original service demand. In our case, however, the amount of demand generated by a skin node can be affected by the particular configuration of facilities within the r-shape. In Figure 5·2 we illustrate why this is the case. Node u on the ring has a shortest path to facility node v_i that intersects the skin of v_i's r-ball at point B, thereby increasing the demand of a local node at B by $s(u)$. As the locations of the facilities may change during the various steps of the local optimizing process (*e.g.,* the facility moves from C to D, Figure 5·2), the skin node along the shortest path between u and the new location of the facility may change (node/point E in Figure 5·2). Consequently, a demand *mapping error* is introduced by keeping the mapping fixed (as initially determined) throughout the location optimization process. Let $\Delta_i(r, j, u)$ denote the amount of mapping error attributed to ring node u with respect to a move of the facility from v_i to v_j under the aforementioned fixed mapping and radius r. Then the *total mapping error* introduced in domain W_i under radius r is given by:

$$\Delta_i(r) = \sum_{\substack{v_j \in V_i \\ v_j \neq v_i}} \sum_{u \in U_i v_j \neq v_i} \Delta_i(r, j, u). \tag{5.8}$$

The mapping error in Equation (5.8) could be eliminated by re-computing the skin mapping at each stage of the optimizing process (*i.e.,* for each new intermediate facility configuration). Such an approach not only would add to the computational cost but – most important – would be practically extremely difficult to implement as it would require the collection of demand statistics under each new facility placement, delaying the optimization process and inducing substantial overhead. Instead of trying to eliminate the mapping error one could try to assess its magnitude (and potential impact) on the effectiveness of the distributed UKM/UFL. This is explored next.

The example depicted in Figure 5·2 helps derive an expression for the mapping error $\Delta_i(r, j, u)$, assuming a two-dimensional plane where nodes are scattered in a uniform and continuous manner over the depicted domain. $\Delta_i(r, j, u)$ corresponds to the length difference of the two different routes between node u (point A) and node v_j (point D). Therefore,

$$\Delta_i(r, j, u) = |AB| + |BD| - |AD|. \tag{5.9}$$

Note that for those cases in which the angle $\hat{\phi}$ between AC and CD, is 0 or π, $|AB|+|BD| = |AD|$, and therefore, $\Delta_i(r, j, u) = 0$. For any other value of $\hat{\phi}$, AB, BD and AD correspond to the edges of the same triangle and therefore, $|AB| + |BD| - |AD| > 0$ or $\Delta_i(r, j, u) > 0$.

Based on Equation (5.9), it is possible to derive an upper bound regarding the total mapping error $\Delta_i(r)$ for this particular environment. In Appendix E, we prove that:

$$\Delta_i(r) \leq 2\pi^2 r^3 (R^2 - r^2), \tag{5.10}$$

where R is the radius of the particular domain W_i (for simplicity we assume that the domain is also a circle).

According to Equation (5.10), the upper bound for $\Delta_i(r)$ is close to 0, when $r \to 0$ or $r \to R$. We are interested in those cases where the r-ball is small. This corresponds to small values of r for the particular (two-dimensional continuous) environment. Therefore, a small radius r in addition to being preferable for scalability reasons has the added advantage of facilitating the use of a simple and practical mapping with small error and expected performance penalty.

5.4 Synthetic Results on ER and BA Graphs

In this section we evaluate our distributed facility location approach on synthetic Erdös-Rényi (ER) [32] and Barabási-Albert (BA) [8] graphs generated using the BRITE generator [81]. For ER graphs, BRITE uses the Waxman model [116] in which the probability that two nodes have a direct link is $P(u, v) = \alpha \cdot e^{-d/(\beta L)}$, where d is the Euclidean distance between u and v, and L is the maximum distance between any two nodes. We maintain the default values of BRITE $\alpha = 0.15,\ \beta = 0.2$ combined with an incremental model in which each node connects to $m = 2$ other nodes. For BA graphs we also use incremental growth with $m = 2$. This parametrization creates graphs in which the number of (undirected) links is almost double the number of vertices (as also observed in real AS traces that we use later in the paper).

5.4.1 Node Coverage with Radius r

Figure 5·3 depicts the fraction of the total node population that can be reached in r hops starting from a certain node in ER and BA graphs, respectively. We plot the mean and the 95% confidence interval of each node under different network sizes $n =$ 400, 600, 800, 1000, representing typical populations of core ASes on the Internet as argued later on. The figures show that a node can reach a substantial fraction of the total node population by using a relatively small r. In ER graphs, $r = 2$ covers $2\% - 10\%$ of the nodes, whereas $r = 3$ increases the coverage to $10\% - 32\%$, depending on network size. The coverage is even higher in BA graphs, where $r = 2$ covers $4\% - 15\%$, whereas $r = 3$ covers $20\% - 50\%$, depending again on network size. These observations are explained by the fact that larger networks exhibit longer shortest paths and diameters and also because BA graphs, owing to their highly skewed (power-law) degree distribution, possess shorter shortest paths and diameters than corresponding ER graphs of the same link density.

5.4.2 Performance of distributed UKM

In this section we examine the performance of our distributed UKM of radius r, hereafter referred to as dUKM(r), when compared to the centralized UKM utilizing full knowledge. We fix the network size to $n = 400$ (matching measurement data on core Internet ASes that we use later on) and assume that all nodes generate the same amount of service demand

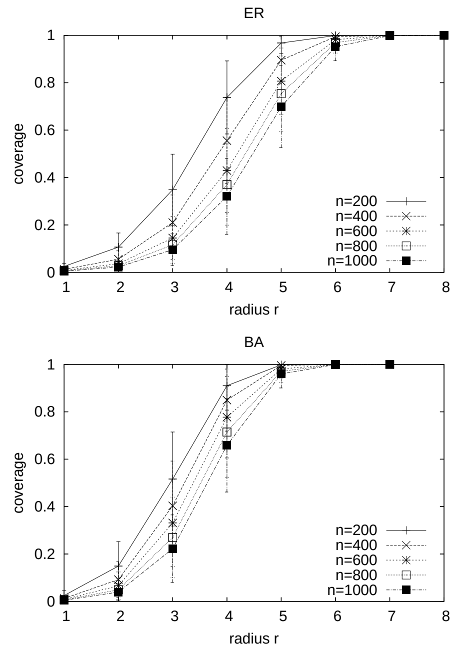


Figure 5·3: Average coverage of a node for different size of ER and BA graphs.

$s(v) = 1, \forall v \in V$. To ensure scalability, we don't want our distributed solution to encounter r-shapes that involve more that 10% of the total nodes, and for this we limit the radius to $r = 1$ and $r = 2$, as suggested by the node coverage results of the previous section. We let the fraction of nodes that are able to act as facilities (*i.e.,* service hosts) take values $k/n = 0.1\%, 0.5\%, 1\%, 2\%$, and 5%. We perform each experiment 10 times to reduce the uncertainty due to the initial random placement of the k facilities.

The plots on the left-hand-side of Figure 5·4 depict the cost of our dUKM(r) approach normalized over that of the centralized UKM, with the plot on top for ER graphs and the plot on the bottom for BA graphs. For both ER and BA graphs, the performance of our distributed solution tracks closely that of the centralized one, with the difference diminishing fast as r and k are increased. The normalized performance for BA graphs converges faster (*i.e.,* at smaller k for a given r) to ratios that approach 1. This owes to the existence of highly-connected nodes (the so call "hubs") in BA graphs — building facilities in few of the hubs is sufficient for approximating closely the performance of the

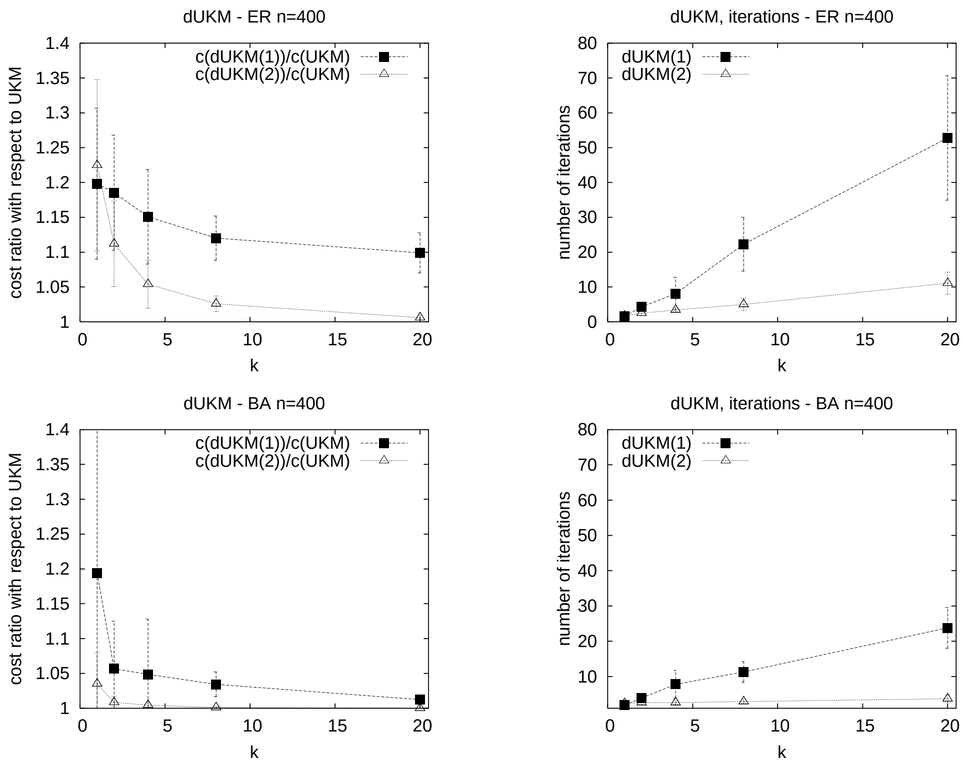


Figure 5·4: The relative performance between dUKM(r) and UKM, and the number of iterations for the convergence of the former, for $r = 1$ and $r = 2$, and different facility densities $k/n = 0.1\%, 0.5\%, 1\%, 2\%,$ and 5% under ER and BA graphs.

centralized UKM. The two plots on the right-hand-side of Figure 5·4 depict the number of iterations needed for dUKM(r) to converge. A smaller value of r requires more iterations as it leads to the creation of a large number of small sub-problems (re-optimizations of many small r-shapes). BA graphs converge in fewer iterations, since for the same value of r BA graphs induce larger r-shapes[12] and, thus, fewer re-optimizations.

5.4.3 Performance of distributed UFL

In order to evaluate the performance of dUFL(r), we need to decide how to set the facility acquisition costs $f(v_j)$, which constitute part of the input of a UFL problem (see Defi-

[12] Again it is the hubs that create large r-shapes. Even under a small r, a hub will be close to the facility that re-optimizes its location, and this will bring many of the hub's immediate neighbors into the r-shape.

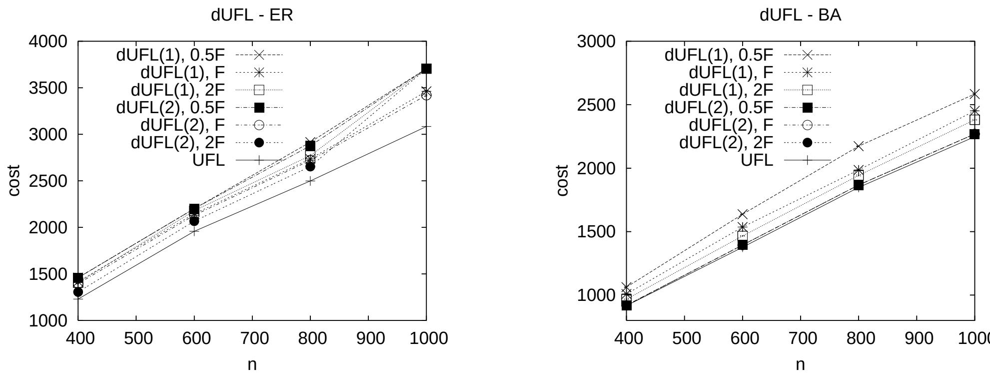
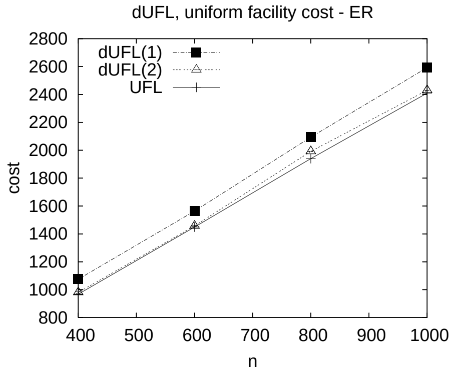


Figure 5·5: Cost comparison between dUFL(r) and UFL, for $r = 1$ and $r = 2$, and different network sizes under ER and BA graphs and degree-based facility cost $f(v_j) = d(v_j)^{1+\alpha_G}$.

nition 12). This is a non-trivial task, essentially a pricing problem for network services. Although pricing is clearly out of scope for this paper, we need to use some form of $f(v_j)$'s to demonstrate our point that, as with UKM, the performance of the distributed version of UFL tracks closely that of the centralized one. To that end, we use two types of facility costs: *uniform*, where all facilities cost the same independently of location (*i.e.,* $f(v_j) = f$, $\forall v_j \in V$) and, *non-uniform*, where the cost of a facility at a given node depends on the location of that node. The uniform cost model is more relevant when the dominant cost is that of setting up the service on the host, whereas the non-uniform cost model is more relevant when the dominant cost is that of operating the facility (implying that this operating cost is proportional to the desirability of the host, which depends on topological location).

For the non-uniform case we will use the following rule: we will make the cost of acquiring a facility proportional to its degree, *i.e.,* proportional to the number of direct links it has to other nodes. The intuition behind this is that a highly connected node will most likely attract more demand from clients, as more shortest-paths will go through it and, thus, building a facility there will create a bigger hot-spot, and therefore the node should charge more for hosting a service.[13] In [51],[52] the authors showed that the "coverage"

[13]As sketched in the introduction, a node may correspond to an AS that charges for allowing network



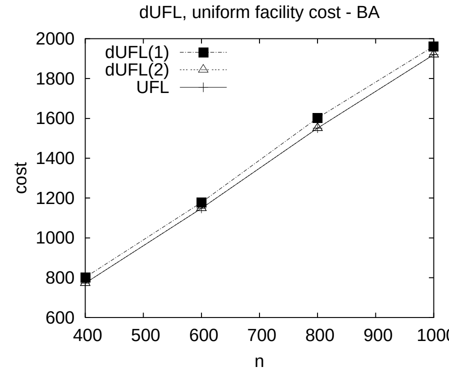


Figure 5·6: Cost comparison between dUFL(r) and UFL, for $r = 1$ and $r = 2$, and different network sizes under ER and BA graphs and uniform facility cost.

of a node increases super-linearly with its degree (or alternatively, the number of shortest paths that go through it). We, therefore, use as facility cost $f(v_j) = d(v_j)^{1+\alpha_G}$, where $d(v_j)$ is the degree of node $v_j \in V$ and α_G is the skewness of the degree distribution of the graph G. In order to estimate the value of α_G, we use the Hill estimator: $\hat{\alpha}_{k,m}^{(Hill)} = 1/\hat{\gamma}_{k,m}$, where: $\hat{\gamma}_{k,m} = \frac{1}{k}\sum_{i=1}^{k} \log \frac{X_{(i)}}{X_{(k+1)}}$, $X_{(i)}$ denotes the i-th largest value in the sample $X_1, ..., X_n$. We prefer the Hill estimator since it is less biased than linear regression.

In Figure 5·5 we plot the cost of dUFL(1), dUFL(2), and centralized UFL, in ER and BA graphs under the aforementioned degree-based facility cost. For dUFL, we present three lines for each radius r, corresponding to different initial number of facilities used in the iterative algorithm of Section 5.2.2. We use $k_0 = 0.5 \cdot F$, F, and $2 \cdot F$, where F denotes the number of facilities opened by the corresponding centralized UFL. As evident from the results, the cost of dUFL is close to that of UFL (around 5-15% for both types of graphs). As with dUKM, the performance improves with r and is slightly better for BA graphs (see the explanation in Section 5.4.2). Also we observe a tendency for lower costs when starting the distributed algorithm with a higher number of initial facilities. Under the non-uniform (degree-based) cost model, both dUFL and UFL open facilities in 2-8% of the total nodes, depending on the example.

services to be installed on its local GSH.

We also evaluate the performance of dUFL under uniform facility cost f; the cost is set at a value that leads to building the same number of facilities as the corresponding degree-based example. Both the distributed and centralized UFL build the same number of facilities, and the performance of dUFL is very close to the centralized one, as is illustrated in Figure 5·6.

Again, we emphasize that our goal here is not to evaluate performance under different pricing scheme, but rather to show that the performance of distributed UFL tracks well that of the centralized, optimal approach.

5.5 Results for Real AS-level Topologies

To further investigate the performance of our distributed approach, as well as better support our sketched application scenario described in the introduction, we include in this section performance results on real AS-level maps under non-uniform service demand from different clients.

5.5.1 Description of the AS-level Dataset

We use the relation-based AS map of the Internet from December 2001[14] obtained using the measurement methodology described in [112]. The dataset includes two kinds of relationships between ASes.

- Costumer-Provider: The costumer is typically a smaller AS that pays a larger AS for providing it with access to the rest of the Internet. The provider may, in turn, be a costumer of an even larger AS. A costumer-provider relationship is modeled using a directed link from the provider to the costumer.

- Peer-Peer: Peer ASes are typically of comparable sizes and have mutual agreements for carrying each other's traffic. Peer-peer relationships are modeled using undirected links.

[14] `http://www.cc.gatech.edu/~mihail/ASdata.html`

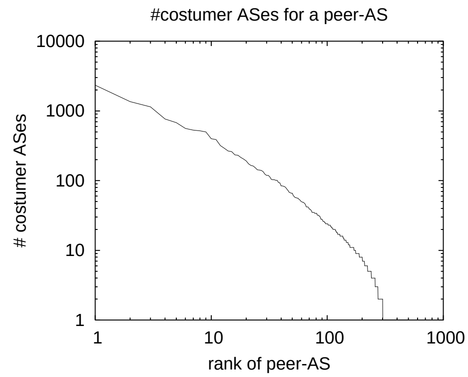


Figure 5·7: Number of costumer ASes for each peer-AS in decreasing order according to rank.

Overall the dataset includes 12,779 unique ASes, 1,076 peers and 11,703 costumers, connected through 26,387 directed and 1,336 undirected links. Since this AS graph is not connected, we chose to present results based on its largest connected component[15], which we found to include a substantial part of the total AS topology at the peer level: 497 peer ASes connected with 1,012 undirected links; we verified that this component contains all the 20 largest peer ASes reported in [112]. Since it would be very difficult to obtain the real complex routing policies of all these networks, we did not consider policy-based routing, but rather asssumed shortest-path routing based on the aforementioned connected component.

We exploit the relationships between ASes in order to derive a more realistic (non-uniform) service demand for the peer ASes that we consider. Our approach is to count for each peer AS the number of costumer ASes that have it as provider, either directly or through other intermediary ASes. We then set the service demand of a peer AS to be proportional to this number. In Figure 5·7 we plot the demand profile of peer ASes (in decreasing order using Log-Log scale). As evident from this plot, the profile is power-law like (with slight deviation towards the tail), meaning that few core ASes carry the majority of the demand that flows from client ASes. In the sequel we present performance results in

[15]There are smaller connected components (2-8 ASes) that are formed by small regional ISPs with peering relationships.

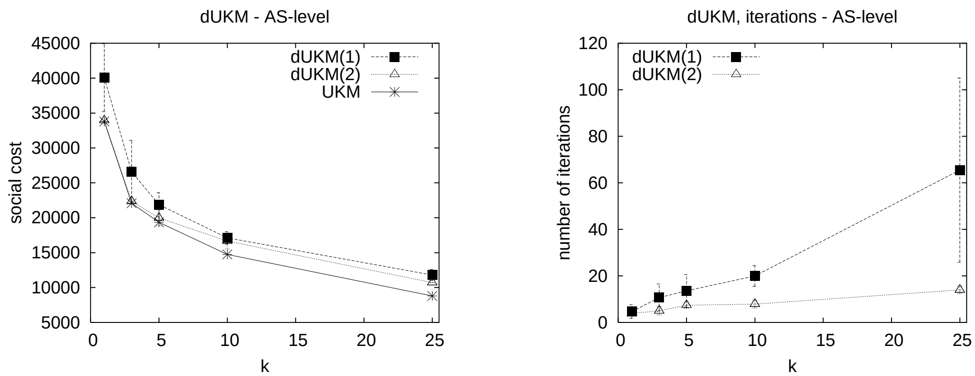


Figure 5·8: The cost of dUKM(r) and UKM, and the number of iterations for the convergence of the former, for $r = 1$ and $r = 2$, and different facility densities $k/n = 0.1\%, 0.5\%, 1\%, 2\%$, and 5% under the AS graph.

which nodes correspond to peer ASs that generate demand that follows the aforementioned power-law like profile. We seek to identify the peer ASes for building service facilities.

5.5.2 Distributed UKM on the AS-level Dataset

The plots on the left-hand-side of Figure 5·8 show the cost of dUKM(1), dUKM(2), and the centralized UKM, under the AS-level graph. Clearly, even for small values of r, the performance of our distributed approaches track closely that of the centralized approach. Regarding the number of iterations needed for convergence, the same observations apply as with the synthetic topologies, *i.e.,* they increase with smaller radii. The substantial benefit from knowledge of only local neighborhood topologies ("neighbors of neighbor") has been observed for a number of applications, including [78] which has also investigated and quantified implementation overhead in an Internet setting.

5.5.3 Distributed UFL on the AS-level Dataset

Table 5.1 presents the performance of dUFL on the AS-level dataset. Again, it is verified that dUFL is very close in performance to UFL, even for small values of r (within 4% for $r = 2$, under both examined facility cost models).

	cost ratio dUFL(1)/UFL		cost ratio dUFL(2)/UFL	
	mean	median	mean	median
degree-based	1.22	1.20	1.04	1.03
uniform	1.01	1.01	1.01	1.01

Table 5.1: Cost ratio between dUFL(r) and UFL in the AS-level topology.

5.6 Non-Stationary Demand and Imperfect Redirection

Up to now, our performance study has been based on assuming (1) stationary demand, and (2) perfect redirection of each client to its closest facility node. The stationary demand assumption is not justified for relatively large time-scales (hours or days), and perfect redirection can be either too costly to implement or too difficult to enforce due to faults or excessive load. In this section we look at the performance of distributed facility location when dropping the aforementioned assumptions. First, we present a measurement study for obtaining the non-stationary demand corresponding to a multi-player on-line game and then use this workload to derive a performance comparison between dUFL and UFL. Then, we assume that mapping a client to its closest facility node has to incur some time lag and study the performance implications of such an imperfect redirection scheme.

5.6.1 Measuring the demand of a popular multi-player game

We used the Mininova web-site[16] to track all requests for joining a torrent corresponding to a popular on-line multi-player game. By tracking the downloads of the game client, which is possible to do due to the use of BitTorrent, we can obtain a rough idea about the demographics of the load put on the game servers, to which we do not have direct access. We then use this workload to quantify the benefits of instantiating game servers dynamically according to dUFL.

More specifically, we connected periodically at 30-minute intervals to the tracker serving this torrent, over a total duration of 42 hours. At each 30-minute interval, we got all the

[16] `http://www.mininova.org`

IPs of participating downloaders by issuing to the tracker multiple requests for neighbors until we got all distinct downloaders at this point in time.[17] In Figure 5·9 (left) we plot the number of concurrent downloads at each measurement point. Overall, we were able to capture a sufficient view of the activity of the torrent and detect expected profiles, *e.g.,* diurnal variation over the course of a day. In total, we saw 34,669 unique users and the population varied from 6,000 to 8,000 concurrent users, *i.e.,* the population variance was close to 25%.

Moving on, we used Routeviews[18] to map each logged IP address to an AS. The variance in the number of concurrent users from a particular AS was even higher. Focusing on the most popular AS, we found out that the variance in the number of concurrent users was as high as 50%, as it is shown in Figure 5·9 (right). Last, we looked at churn at the AS level by counting the number of new ASes joining and existing ASes leaving the torrent over time [41]. Formally, we defined $churn(t) = \frac{U_{t-1} \ominus U_t}{\max\{|U_{t-1}|,|U_t|\}}$, where U_t is the set of ASes at time t, and $\ominus$ is the set difference operator. In Figure 5·10 we plot the evolution of churn. One can observe that AS-level churn is quite high, ranging from 6% to 11%, with no specific pattern. This serves our purpose which is to study the performance of dUFL under non-stationary demand.

5.6.2 Distributed UFL under non-stationarity demand

We consider a distributed server migration scheme given by dUFL with radius $r = 1$. The pricing model for starting a server at an AS is the aforementioned degree-based one of Section 5.4.3. The evaluation assumes an AS-level topology obtained from Routeviews. The demand originating from each AS at each particular point in time is set equal to the value we obtained from measuring the downloads going to the torrent of the game client. We compare the cost of UFL, dUFL(1), static-min, and static-max. Static-min is a

[17] Tracker is a server that maintains the set of distinct downloaders of a torrent. Upon a neighbor set request, the tracker replies with a random subset of the distinct downloaders set. We requested the size of the distinct downloaders set, and then we repeatedly requested for a new neighbor set until we reach the same number of distinct IPs.

[18] `http://www.routeviews.org`

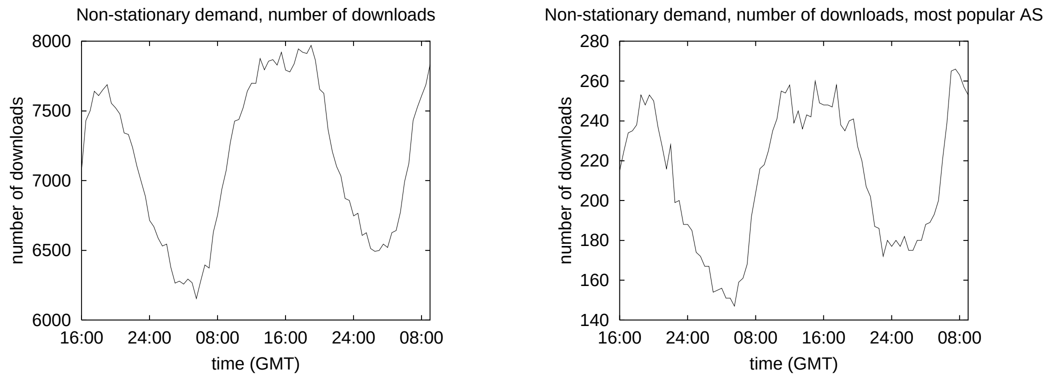


Figure 5·9: The number of concurrent downloads from all ASes and from the most popular AS in the torrent of an on-line multi-player game at each measurement point.

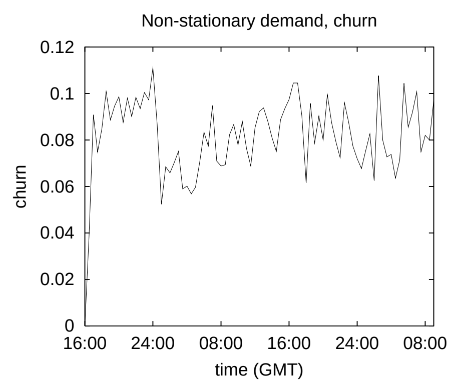


Figure 5·10: Churn evolution in the AS-level in the torrent of a popular on-line multi-player game at each measurement point.

simple heuristic that maintains the same placement across time. The number of maintained facilities is equal to the minimum number of facilities that UFL opened in the duration of the experiment. This is used as a baseline for the performance of an under-provisioned static placement of servers according to minimum load. Static-max captures the cost of an over-provisioned placement according to peek load. Obviously, static-max suffers from a high purchase cost of buying a maximum number of servers (in this case 100), whereas static-min suffers from high communication cost to reach the few bought servers (in this case 70).

We report the average cost in the duration of the experiment (42 hours) for each one

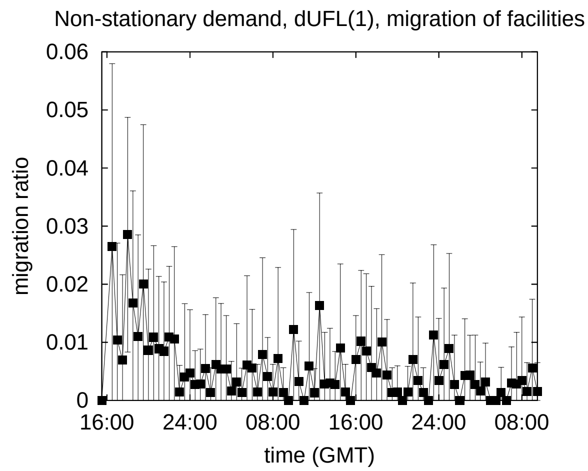


Figure 5·11: Migration ratio of dUFL(1) in the torrent of a popular on-line multi-player game at each measurement point.

of the aforementioned policies. For each policy we repeated the experiment 100 times to remove the effect of the initial random opening of facilities. In Figure 5·12 we plot the resulting average costs along with 95^{th} percentile confidence intervals. One can see that dUFL(1) achieves 4 to 7 times lower cost compared to static-min and static-max. Looking at the close-up, it can also be seen that dUFL(1) is actually pretty close, within 10-20%, of the performance of the centralized UFL computed at each point in time. Taken together, these results indicate that dUFL(1) yields a high performance also under non-stationary demand.

Next, we quantify the number of server migrations required by dUFL(1) to track the offered non-stationary demand. In Figure 5·11 we plot the percentage of servers that are migrated, henceforth referred as migration ratio, along with 95^{th} percentile confidence intervals based on 100 runs. Evidently, migrations are rather rare, typically 0%-3%, after the servers stabilize from their initial random positions, to where dUFL(1) will have them at each point in time. These results suggest that dUFL(1) is relatively robust to demand changes and can typically address them without massive numbers of migrations that are of course costly in terms of bandwidth, etc. Of course, the number of migrations can be reduced further by trading performance with laziness in triggering a migration.

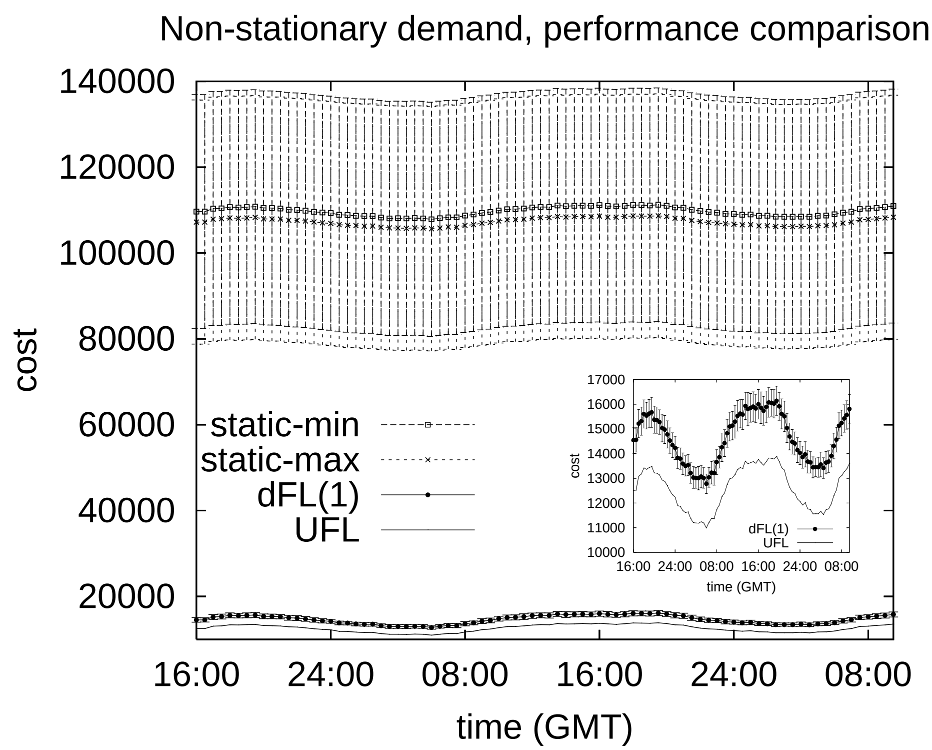


Figure 5·12: Average cost of static-min, static-max, dUFL(1) and UFL in the torrent of a popular on-line multi-player game at each measurement point.

5.6.3 The Effect of Imperfect Redirection

We now move on to dropping the assumption that clients are always redirected to their closest facility, which pretty much implies that there are no performance penalties for them due to server migrations. In many cases it has been shown that perfect redirection is indeed feasible using route triangulation and DNS [33]. In this section, however, we relax this assumption, and study the effects of imperfect redirection. We do so to cover cases in which perfect redirection is either too costly to implement, or exists, but performs sub-optimally due to faults or excessive load.

To this end, we assume that there exists a certain amount of *lag* between the time a server migrates to a new node and the time that the migration is communicated to the affected clients. During this time interval, a client might be receiving service from

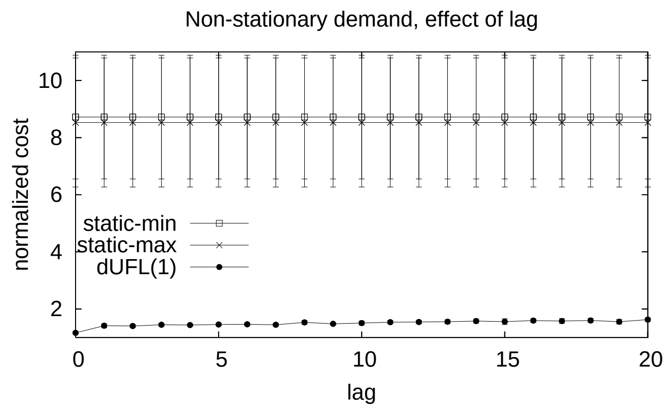


Figure 5·13: Normalized cost of static-min, static-max and dUFL(1) with respect to the cost of UFL in the torrent of a popular on-line multi-player game under various levels of lag.

its previously closest facility which, however, may have ceased to be optimal due to one or several migrations. Since we assume that migrations occur at fixed time intervals, we measure the lag in terms of number of such intervals (1 facility migration at each interval). Notice that under the existence of lag, even with stationary demand, the optimization is no longer guaranteed to be loop-free (as in Section 5.3.1). We solve this by stopping the iterative re-optimization if it reaches a certain high number of iterations.

In Figure 5·13 we plot the cost ratio between dUFL(1) and dUFL and the 95^{th} percentile confidence interval under various levels of lag that range from 0 up to 20 (which means that clients of facility i hear about i's migration after i+lag has completed its migration). As expected, lag puts a performance penalty on dUFL. The degradation, however, is quite smooth, while the performance always remains superior to static-min and static-max.

5.7 Chapter Summary

Selfish wiring by end-users has many implications to service provisioning both for the users and the service provider. The design of an autonomic deployment to address the selfish behavior of users has received very little attention. Leveraging recent advances in virtualization technology we described a distributed approach for the problem of placing servers in large-scale networks. We overcome the scalability limitations of classic centralized approaches by re-optimizing the locations and the number of facilities through local optimizations which are refined in several iterations. Re-optimizations are based on exact topological and demand information from nodes in the immediate vicinity of a facility, assisted by concise approximate representation of demand information from neighboring nodes in the wider domain of the facility. Using extensive synthetic and trace-driven simulations we demonstrate that our distributed approach is able to scale by utilization limited local information without making serious performance sacrifices as compared to centralized optimal solutions. We also demonstrate that our distributed approach yields a high performance under non-stationary demand and imperfect redirection. Our experimental results provide evidence in support of the potential benefits end-user selfish server selection offers towards efficient autonomic service deployment.

Chapter 6

Conclusion

While much attention has been paid to the harmful downsides of selfish behavior, the impact of adopting selfish connectivity in real overlay networks has received little attention.

6.1 Summary

In this thesis, we do not dwell on the negatives, but instead we focus on the potential benefits of selfish neighbor selection in real overlay networks. Our results indicate that selfish neighbor selection primitives, apart from the obvious benefits to selfish users, can be a powerful tool towards distributed overlay network creation, maintenance, monitoring, and troubleshooting. Indeed, we confirm that selfish neighbor selection is not a problem, so much as inaction, indifference, or a naïve reaction, all of which incur high individual and social costs.

We provided a systematic evaluation of the design space for connectivity management in overlays. This evaluation includes the demonstration of implications and promise, resulting from adopting a selfish approach to neighbor selection and distributed service provisioning in real network overlays. We also confirm that local optimization procedures, based on local search heuristics, are capable of creating optimized topologies for different overlay application and addressing scalability issues.

Important implications of selfish neighbor selection to system design were derived. In the context of overlay routing and file sharing, we showed that selfish wiring strategies are easily realizable and can achieve performance that is substantially better than the one achieved by heuristics currently used in overlay systems. Such selfish wirings must be a component of overlay systems to protect them from being exploited. In the context of service provisioning, we showed that distributed service provisioning that relies on end-user selfish wiring is easily realizable and has deployment cost which is close to the optimal.

In overlays that the optimal neighbor or server selection is provided as a service users are incented to follow the protocol. Our surprisingly good results obtained under our framework are robust as the assumption that users will sacrifice their performance towards improving the overlay's performance has been relaxed. We also showed that end-user awareness through selfish neighbor or server selection leads to better overlays.

6.2 Directions for Future Research

Promising future research directions, some of which are part of our current research agenda, include credit-based selfish neighbor selection, the study of the performance characteristics of pair-wise stable graphs, and applications of SNS in cloud computing and network neutrality.

Credit-based Selfish Neighbor Selection. Selfish neighbor selection can be used by a user to increase its revenue. In many contexts like peering agreements between ISPs or wireless network deployments, users are paid, based on the traffic they receive or relay. A selfish user would strive to maximize the difference between payments received and given out, by choosing its first hop neighbors. In [4], the authors studied the case where the pair-wise pricing function is part of the solution, and the traffic matrix is uniform. We would like to study the more realistic scenario where the pricing functions are given and the traffic matrix is not uniform. We would rely on hot-potato routing or optimal scheduling to one hop neighbors [42], rather than on source routing [3], which is impossible to enforce in real implementations.

Towards pair-wise stable graphs. Central to this thesis is the study of the existence and performance of pure Nash equilibria of the SNS game. In pure Nash equilibria, no user can re-wire unilaterally and reduce its cost. The aforementioned equilibria are not the only ones that can appear. A more restrictive set of equilibria is that of the pairwise stable equilibria, where no two users can rewire and reduce their costs [25]. This type of stability is more appropriate in settings where the establishment of connectivity is bidirectional and requires some level of reciprocation on the pair-exchanged traffic.

Cloud Computing Applications. While nowadays the price of storage is decreasing rapidly, the administrative cost of data centers increases mainly because manual configurations have to take place [104]. In cloud computing applications a request is forwarded to many processing and storage units. When a particular request is satisfied, the results are replicated to different storage units. Moreover, the replicas of data have to be in network proximity to enhance retrieval and update time. In principle, the redundancy of replicas makes the data more resilient to loss, but on the other hand it hardens their management, which is why their number has to be bounded. To address all the aforementioned challenges, an autonomic deployment, based on selfish neighbor selection primitives can be used. The location of replicas, their pairwise distance, along with the available storage unit capacity and reliability can be part of the objective function that every request strives to optimize.

Network Neutrality, Anti-censorship and User Satisfaction. The violation of network neutrality has become the subject of recent debate [1] and includes the following instances. First, ISPs throttle P2P traffic in order to reduce operational cost because inter-ISP peering agreements are affected. Second, ISPs and content providers have to comply with government or intellectual property regulations, regarding universal access to content. Another problem is that it is impossible to monitor end-user satisfaction in a huge population of users. All the aforementioned issues can be addressed by selfish neighbor selection principles. In the context of throttling or censorship, a user can rewire in order to gain access through other neighboring users. The metrics that bias this selection can be network proximity or meta information about the content that each user has access to. Moreover, selfish neighbor selection can be used as a distributed monitoring and re-directional mechanism within an ISP to improve end-user satisfaction.

We also believe that results presented in this thesis can be used in contexts other than overlay networks. Our work goes against the conventional thinking that overlay users conform to a specific protocol, which is a novel way to design communication networks starting with a clean slate.

Appendix A

NP-hardness of maximizing the sum of bottleneck bandwidths

Consider a node s that wants to connect to a network composed of m nodes $v_i \in C$, $1 \leq i \leq m$ and n nodes $u_j \in S$, $1 \leq j \leq n$, so as to maximize the sum of bottleneck bandwidths to all destinations as described in Section 3.4.1. Each node v_i has directed links of bandwidth b_2 to a subset S_i of the nodes of S. Node s has k links of bandwidth b_1 which it wants to use for connecting to k distinct nodes of $C \cup S$ (see Figure A·1 for an illustration). Establishing a link to a node of S increases the utility of s by at most b_1 independently of how it uses its remaining $k-1$ links. Establishing a link to a node of C increases its utility by b_1 plus b_2 times the number of nodes of S that s reaches by following the new link. When s can reach a node u_j through direct links to more than one nodes v_i we pick exactly one of the paths $s \to v_i \to u_j$ to be the bottleneck path for destination u_j (all have bandwidth $\min(b_1, b_2)$ so it doesn't matter which one we choose). Granted this construction, it is clear that s will establish direct links only to nodes of C. More over, it will have to choose those nodes of C that maximize the number of distinct reachable nodes of S. Therefore, a solution that maximizes the sum of bottleneck bandwidths to nodes of $C \cup S$ implies an optimal solution to the MAX-UNIQUES(k) problem which is shown in Appendix B to be NP-hard. Therefore, maximizing the sum of bottleneck bandwidths is also an NP-hard problem.

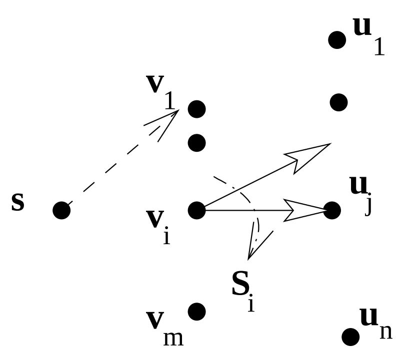


Figure A·1: Reduction from MAX-UNIQUES(k) to max sum of bottleneck bandwidths.

Appendix B

NP-hardness of maximizing the number of uniques (MAX-UNIQUES)

Let MAX-UNIQUES(k) be an optimization problem in which one has to select k subsets U_i, $1 \leq i \leq m$ of a set U with n elements so as to maximize the cardinality of the union $U(k) = \bigcup_{i \in choice} U_i$. Let UNIQUES($k$) be the corresponding decision problem in which one asks whether there is a *choice* leading to $|U(k)| = l$. UNIQUES(k) is clearly NP-complete because for $l = n$ a solution to UNIQUES(k) would imply a solution to SET-COVER. Therefore, MAX-UNIQUES(k) is NP-hard.

Appendix C

NP-hardness of maximizing the minimum MAX-FLOW

Consider a node s that wants to select a set of neighbors σ from a network composed of m nodes $v_i \in V$, n nodes $u_j \in U$, and a single node t, so as to maximize its broadcast bandwidth defined to be its minimum max-flow to any destination, *i.e.,* $\Phi(s, \sigma) = \min_{x \in (V \cup U \cup \{t\})} MF(s, x, \sigma)$, where $MF(s, x, \sigma)$ denotes the max-flow from s to x under strategy σ. Node s can use $k < m$ links whose bandwidth is b_1 if the other end-point belongs to V, and $\epsilon \approx 0$ in any other case, implying that an optimal strategy σ for s must satisfy $\sigma \subset V, |\sigma| = k$. Each node v_i has directed links of bandwidth b_2 to a subset U_i of the nodes of U. Each node u_j has a link of bandwidth b_3 to t. Node t has links of bandwidth b_1 to all nodes of V and U (see Figure C·2 for an illustration). Link bandwidths obey:

$$b_1 \gg b_2 \gg b_3 \tag{C.1}$$

Let $\phi(s, X, \sigma) = \min_{x \in X} MF(s, x, \sigma)$ denote s's minimum max-flow to any node in the set X. Combining $k < m$ and (C.1), we get that under any σ, at least one node of V will get s's flow only indirectly through t, *i.e.,* :

$$\phi(s, V, \sigma) = MF(s, t, \sigma) \tag{C.2}$$

The max-flow from s to u_j is equal to the max-flow from s to t, plus b_2 for each connected path $s \to v_i \to u_j$ under σ, minus the amount of flow that crosses the link from u_j to t in a max-flow from s to t under σ. Since this flow on the (u_j, t) link is either 0, or $b_3 < b_2$ when there's at least one connected path $s \to v_i \to u_j$ in σ, we get $MF(s, u_j, \sigma) \geq MF(s, t, \sigma)$,

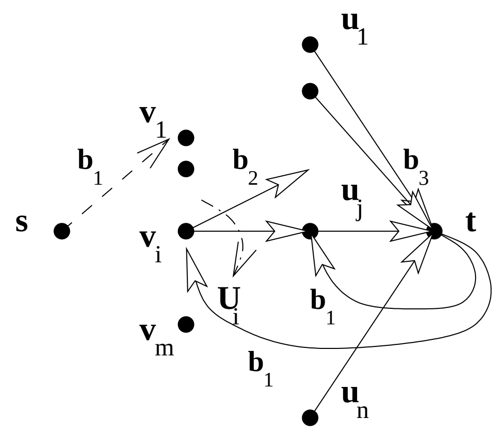


Figure C·2: Reduction from MAX-UNIQUES(k) to Max-Min.

$\forall u_j \in U$, or equivalently:

$$\phi(s, U, \sigma) \geq MF(s, t, \sigma) \tag{C.3}$$

The max-flow to node t is:

$$MF(s, t) = b_3 \cdot paths(s, V, U) \tag{C.4}$$

where $paths(s, V, U)$ is the number of connected paths $s \to v_i \to u_j$ that do not share (v_i, u_j) edges, or equivalently the number of nodes u_j that carry a non-zero flow in a max-flow from s to t. Equations (C.2), (C.3) suggest that the maximization of the broadcast bandwidth calls for the maximization of $MF(s, t)$, which in view of (C.4), is achieved through the maximization of $paths(s, V, U)$. Maximizing $paths(s, V, U)$ requires choosing k subsets V_i so as to maximize the cardinality of their union. A straight-forward reduction from set-cover can be used to show that max $paths(s, V, U)$ is an NP-hard problem (see Appendix B). Therefore, maximizing the broadcast bandwidth is NP-hard as it implies a solution to max $paths(s, V, U)$.

Appendix D

NP-hardness of maximizing the sum of MAX-FLOWs

Consider a node s that wants to connect to a network composed of m nodes $v_i \in V$, n nodes $u_j \in U$, and h nodes $w_l \in W$, where h is a function of the out-degrees of the v_i's as will be explained shortly, so as to maximize the sum of its max-flows to all nodes in the union of V, U, W. Node s can use $k < m$ links whose bandwidth is 1 if the other end-point belongs to V, and $\epsilon \approx 0$ in any other case, implying than an optimal strategy σ for s must satisfy $\sigma \subset V, |\sigma| = k$. Each node v_i has directed links of unit bandwidth to a subset U_i of the nodes of U. Each node u_j has a link of unit bandwidth to each one of the nodes of W (see Figure D·3 for an illustration). The cardinality of W is equal to the highest out-degree of any node in V, *i.e.,* $h = \max_{1 \le i \le m} |U_i|$.

Define $\psi(s, X, \sigma) = \sum_{x \in X} MF(s, x, \sigma)$ where $MF(s, x, \sigma)$ denotes the max-flow from s to x under strategy σ. Node s wants to select a strategy σ that maximizes $\Psi(s, \sigma) = \psi(s, V, \sigma) + \psi(s, U, \sigma) + \psi(s, W, \sigma)$ across all possible strategies. We will show that such an optimal strategy has to maximize the number of nodes in U to which there exists a connected path $s \to v_i \to u_j$.

Notice that $MF(s, v_i, \sigma) = 1$ iff $v_i \in \sigma$ and 0 otherwise, and thus $\psi(s, V, \sigma) = k$ independently of the particular strategy σ chosen. Therefore, we only need to care to maximize $\psi(s, U, \sigma) + \psi(s, W, \sigma)$. If s chooses to connect to v_i, meaning $v_i \in \sigma$, the contribution to $\psi(s, U, \sigma)$ will be $|U_i|$, because each outgoing link of v_i increases by 1 every max-flow from s to a node $u \in U_i$. The contribution to $\psi(s, W, \sigma)$ will be h for each node $u \in U$ that is reachable from s if v_i is included in σ but becomes unreachable if it is taken out ("connecting" u increases all max-flows from s to nodes $w \in W$ by 1). Therefore if by switching $v_i \in \sigma$ with $v_{i'} \notin \sigma$ we get a strategy σ' which has a higher number of nodes

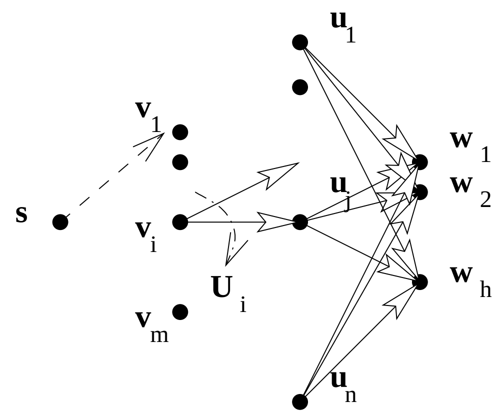


Figure D·3: Reduction from MAX-UNIQUES(k) to Max-Sum.

of U reachable from s, then we should perform the switch because $\Psi(s, \sigma') > \Psi(s, \sigma)$. To see that, notice that the switch can hurt $\psi(s, U, \sigma)$ by at most $h - 1$, if v_i has the highest degree and $v_{i'}$ has degree 1 (it must have at least 1 to be increasing the number of unique u's reached), whereas it benefits $\psi(s, W, \sigma)$ by at least h as it increases the number of nodes of U reachable from s. The above argument implies that an optimal σ must maximize the number of unique nodes of U reachable from s. Therefore, an optimal solution to maximizing the sum of max-flows for s implies an optimal solution to the NP-hard problem MAX-UNIQUES(k) of Appendix B. Therefore, max sum max-flows is an NP-hard problem.

Appendix E

Derivation of an Upper Bound for $\Delta_i(r)$

For the rest, a two-dimensional space is considered over which nodes are scattered in a uniform and continuous manner. The r-ball is considered as a circle with radius r and the entire domain also as a circle with radius R (see Figure 5·2).

Suppose that a node $u \in U_i$ is served by its closest facility node v_i. This case is depicted in Figure 5·2 where u is located at point A and the corresponding facility node v_i is located at point C. Note that line AC intersects with the periphery (skin) of the r-ball at a particular point denoted by B. Clearly, line AC corresponds to the shortest distance between points A and C (nodes u and v_i, respectively). Denoting as x the length of AB, $|AB|$ (the distance of node u from the skin of the r-ball) we can write $AC = x+r$. Line AC may be regarded as the path over which node u uses the resources of the facility located at node v_i.

Suppose that a node $u_j \in V_i$ is considered as a possible alternative facility location. Let D be the point denoting the location of v_j and let y denote the distance between node v_i and node v_j (*i.e.*, the length of CD, $|CD|$). The mapping error, $\Delta_i(r, j, u) = |AB| + |BD| - |AD|$, is always positive since $|AB| + |BD| > |AD|$ (AB, BD and AD correspond to edges of the same triangle) when $A\hat{B}D \neq 0$ and $A\hat{B}D \neq \pi$. The mapping error becomes zero only in the exceptional cases where $A\hat{B}D = 0$ and $A\hat{B}D = \pi$ (corresponding to $\hat{\phi} = \pi$ and $\hat{\phi} = 0$, respectively, as concluded from Figure 5·2).

Let $\Delta_i(r, j)$ be the summation of $\Delta_i(r, j, u)$, $\forall u \in U_i$. Since we have assumed the network area as a two-dimension continuous space, all nodes $u \in U_i$ correspond to the ring area U_i, depicted in Figure 5·2. Consequently, $\Delta_i(r,j)$ is given by the following integral,

$$\Delta_i(r,j) = \int\limits_{U_i} \Delta_i(r,j,u)du. \tag{E.5}$$

Let $\Delta_i(r)$ denote the total mapping error, or the summation of $\Delta_i(r,j)$ for all nodes $j \in V_i$. Therefore,

$$\Delta_i(r) = \int\limits_{V_i} \Delta_i(r,j)dj. \tag{E.6}$$

In Appendix F we derive the following analytical expression for $\Delta_i(r,j,u)$ as a function of parameters x, y, r and $\hat{\phi}$:

$$\begin{aligned} \Delta_i(r,j,u) = \quad & x + \sqrt{r^2 + y^2 - 2yr\cos\hat{\phi}} \\ & - \sqrt{(x+r)^2 + y^2 - 2y(x+r)\cos\hat{\phi}}. \end{aligned} \tag{E.7}$$

$\Delta_i(r,j,u)$ as it is given by Equation (E.7) is difficult to be analyzed. In addition, an analytical expression regarding $\Delta_i(r)$ is not easy to be derived since it is hard to obtain the corresponding integrals. Therefore, in the sequel we obtain an upper bound $\Delta_i(r)$ by using a simple upper bound for $\Delta_i(r,j,u)$ as explained below.

It is easy to see that $r^2+y^2-2yr\cos\hat{\phi} \leq r^2+y^2+2yr = (r+y)^2$, since $-1 \leq \cos\hat{\phi} \leq 1$. Also, $(x+r)^2 + y^2 - 2y(x+r)\cos\hat{\phi} \geq (x+r)^2 + y^2 - 2y(x+r) = (x+r-y)^2$ (note that $y \leq r$).

Based on Equation (E.7), it is concluded that $\Delta_i(r,j,u) \leq x+\sqrt{(r+y)^2}-\sqrt{(x+r-y)^2} = x+r+y-x-r+y$. Therefore, $\Delta_i(r,j,u) \leq 2y$. Given that $y \leq r$,

$$\Delta_i(r,j,u) \leq 2r. \tag{E.8}$$

In order to derive $\Delta_i(r,j)$, according to Equation (E.5), an analytical expression has to be derived for the integral $\int_{U_i} \Delta_i(r,j,u)du$. Note that $0 \leq \Delta_i(r,j,u) \leq 2r$, $\int_{U_i} \Delta_i(r,j,u)du \leq \int_{U_i} 2rdu$ and R corresponds to the radius of the $U_i \cup V_i$ area (note that $R \geq r$). Eventually,

$$\Delta_i(r,j) \leq 2\pi r(R^2 - r^2), \tag{E.9}$$

since the area of the ring U_i is $\pi(R^2 - r^2)$.

In order to derive $\Delta_i(r)$, according to Equation (E.6), an analytical expression has to be derived for the integral $\int_{V_i} \Delta_i(r, j)dj$. Note that $0 \leq \Delta_i(r, j) \leq 2\pi r(R^2 - r^2)$ and $\int_{V_i} \Delta_i(r, j)dj \leq \int_{V_i} 2\pi r(R^2 - r^2)dj$. Eventually,

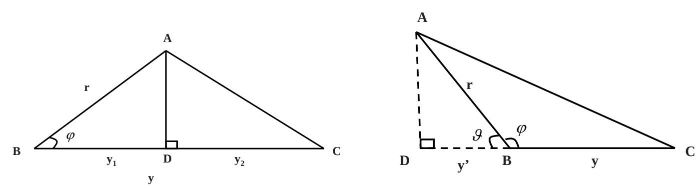
$$\Delta_i(r) \leq 2\pi^2 r^3 (R^2 - r^2), \tag{E.10}$$

since the r-ball area is πr^2.

Appendix F

Derivation of an Analytical Expression for $\Delta_i(r, j, u)$

When one of the angles of a triangle ($\hat{\phi}$) is known as well as the length of both adjacent edges (r and y), then the length of the third edge is possible to be derived as a function of $\hat{\phi}, r, y$. Two different cases may be distinguished with respect to the triangle's particular form, as depicted in Figure F·4.



a. b.

Figure F·4: The two distinguished cases studied to derive the analytical expression for $\Delta_i(r, j, u)$.

For the case depicted in Figure F·4(a), $\cos\hat{\phi} = \frac{y_1}{r}$. Since $y = y_1 + y_2$, $y_2 = y - y_1 = y - r\cos\hat{\phi}$. Furthermore, $\sin\hat{\phi} = \frac{|AD|}{r}$ and $|AD| = r\sin\hat{\phi}$. It holds that $|AC|^2 = |AD|^2 + y_2^2$, or $|AC| = \sqrt{|AD|^2 + y_2^2}$, or $|AC| = \sqrt{r^2\sin^2\hat{\phi} + y^2 + r^2\cos^2\hat{\phi} - 2yr\cos\hat{\phi}}$. Eventually,

$$|AC| = \sqrt{r^2 + y^2 - 2yr\cos\hat{\phi}}. \tag{F.11}$$

The same result is also derived for the case depicted in Figure F·4(b), where $\hat{\theta} = \pi - \hat{\phi}$. For this case, $|AC| = \sqrt{|AD|^2 + (y + y')^2}$. However, $|AD| = r\sin\hat{\theta}$ and $y' = r\cos\hat{\theta}$. Since, $\sin\hat{\theta} = \sin\hat{\phi}$ and $\cos\hat{\theta} = -\cos\hat{\phi}$, $|AD| = r\sin\hat{\phi}$ and $y' = -r\cos\hat{\phi}$. Eventually, Equation (F.11) holds for this case as well.

Bibliography

[1] Vinay Aggarwal, Anja Feldmann, and Christian Scheideler. Can ISPS and P2P users cooperate for improved performance? *SIGCOMM Computuer Communication Review*, 37(3):29–40, 2007.

[2] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. In *SOSP '01: Proceedings of the 18th ACM symposium on Operating systems principles.*

[3] Elliot Anshelevich, Bruce Shepherd, and Gordon Wilfong. Strategic Network Formation through Peering and Service Agreements. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science.*

[4] Esteban Arcaute, Ramesh Johari, and Shie Mannor. Network Formation: Bilateral Contracting and Myopic Dynamics. *Transactions on Automatic Control*, [to appear], 2008.

[5] Aaron Archer. Inapproximability of the asymmetric facility location and k-median problems, 2000. unpublished manuscript, available from the author's web-page.

[6] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.

[7] Moshe Babaioff, Robert Kleinberg, and Christos H. Papadimitriou. Congestion games with malicious players. In *EC '07: Proceedings of the 8th ACM conference on Electronic commerce.*

[8] Albert-Laszlo Barabási and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[9] Philip A. Bernstein and Nathan Goodman. Concurrency control in distributed database systems. *ACM Computer Surveys*, 13(2), 1981.

[10] Ashwin Bharambe, John R. Douceur, Jacob R. Lorch, Thomas Moscibroda, Jeffrey Pang, Srinivasan Seshan, and Xinyu Zhuang. Donnybrook: Enabling Large-Scale, High-Speed, Peer-to-Peer Games. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication.*

[11] Ashwin Bharambe, Jeffrey Pang, and Srinivasan Seshan. Colyseus: a distributed architecture for online multiplayer games. In *NSDI '06: Proceedings of the 3rd conference on Networked Systems Design & Implementation.*

[12] Ashwin R. Bharambe, Cormac Herley, and Venkata N. Padmanabhan. Analyzing and Improving a BitTorrent Network's Performance Mechanisms. In *INFOCOM '06: Proceedings of the 25th IEEE International Conference on Computer Communications.*

[13] Ruchir Bindal, Pei Cao, William Chan, Jan Medval, George Suwala, Tony Bates, and Amy Zhang. Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. In *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems.*

[14] Rob H. Bisseling. *Parallel Scientific Computation: A Structured Approach using BSP and MPI.* Oxford University Press, 2004.

[15] Alberto Blanc, Yi-Kai Liu, Amin Vahdat, and Scott Shenker. Designing Incentives for Peer-to-Peer Routing. In *P2PEcon '04: Proceedings of the 2nd Workshop on the Economics of Peer-to-Peer Systems.*

[16] Craig W. Cameron, Steven H. Low, and David X. Wei. High-density model for server allocation and placement. In *SIGMETRICS '02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems.*

[17] Robert L. Carter and Mark E. Crovella. On the Network Impact of Dynamic Server Selection. *Computer Networks*, 31((23-24)):2529–2558, 1999.

[18] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *OSDI '02: Proceedings of the 8th ACM symposium on Operating systems principles.*

[19] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. SplitStream: high-bandwidth multicast in cooperative environments. In *SOSP '03: Proceedings of the 10th ACM symposium on Operating systems principles.*

[20] Chris Chambers, Wu chi Feng, Wu chang Feng, and Debanjan Saha. A geographic redirection service for on-line games. In *ACM MULTIMEDIA '03: Proceedings of the 11th ACM International Conference on Multimedia.*

[21] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

[22] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making Gnutella-like P2P systems scalable. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications.*

[23] Byung-Gon Chun, Rodrigo Fonseca, Ion Stoica, and John Kubiatowicz. Characterizing selfishly constructed overlay routing networks. In *INFOCOM '04: Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*.

[24] Bram Cohen. Incentives build robustness in bit torrent. In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[25] Jacomo Corbo and David C. Parkes. The Price of Selfish Behavior in Bilateral Network Formation. In *PODC '05: Proceedings of the 24th annual ACM symposium on Principles of distributed computing*.

[26] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.

[27] Eric Cronin, Sugih Jamin, Cheng Jin, Anthony R. Kurc, Danny Raz, and Yuval Shavitt. Constraint mirror placement on the internet. *IEEE Journal on Selected Areas in Communications*, 20(7), 2002.

[28] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminath an Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: amazon's highly available key-value store. In *SOSP '07: Proceedings of the 21st ACM SIGOPS symposium on Operating systems principles*.

[29] Erik D. Demaine, MohammadTaghi Hajiaghayi, Hamid Mahini, and Morteza Zadimoghaddam. The price of anarchy in network creation games. In *PODC '07: Proceedings of the 26th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*.

[30] Zhenhai Duan, Zhi-Li Zhang, and Yiwei Thomas Hou. Service overlay networks: SLAs, QoS, and bandwidth provisioning. *IEEE/ACM Transactions on Networking*, 11(6):870–883, 2003.

[31] Jack Edmonds. Edge-disjoint branchings. In *Proceedings of the 9th Courant Computer Science Symposium on Combinatorial Algorithms, Algorithmics Press*, pages 91–96, 1972.

[32] Paul Erdös and Alfred Rényi. On random graphs I. *Publicationes Mathemaricae Debrecen*, 6:290–297, 1959.

[33] Nathan Faber and Ravi Sundaram. MOVARTO:Server Migration across Networks using Route Triangulation and DNS. In *Proceedings of the VMworld '07*, San Francisco, CA.

[34] Alex Fabrikant, Ankur Luthra, Elitza Maneva, Christos H. Papadimitriou, and Scott Shenker. On a network creation game. In *PODC '03: Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing*.

[35] Joan Feigenbaum and Scott Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 2002.

[36] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust incentive techniques for peer-to-peer networks. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*.

[37] Naveen Garg, Rohit Khandekar, and Vinayaka Pandit. Improved approximation for universal facility location. In *SODA '05: Proceedings of the 16th annual ACM-SIAM symposium on Discrete algorithms*.

[38] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google file system. *SIGOPS Operating Systems Review*, 37(5):29–43, 2003.

[39] Christos Gkantsidis, Thomas Karagiannis, Pablo Rodriguez, and Milan Vojnovic. Planet Scale Software Updates. In *SIGCOMM '06: Proceedings of the ACM SIGCOMM 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*.

[40] Christos Gkantsidis and Pablo Rodriguez. Network Coding for Large Scale Content Distribution. In *INFOCOM '05: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*.

[41] P. Brighten Godfrey, Scott Shenker, and Ion Stoica. Minimizing churn in distributed systems. In *SIGCOMM '06: Proceedings of the ACM SIGCOMM 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*.

[42] David K. Goldenberg, Lili Qiuy, Haiyong Xie, Yang Richard Yang, and Yin Zhang. Optimizing cost and performance for multihoming. In *SIGCOMM '04: Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*.

[43] Krishna Gummadi, Ramakrishna Gummadi, Steve Gribble, Sylvia Ratnasamy, and Ion Stoica. The Impact of DHT Routing Geometry on Resilience and Proximity. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*.

[44] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang. Measurements, analysis, and modeling of bittorrent-like systems. In *IMC '05: Proceedings of the 5th Conference on Internet Measurement*.

[45] Junghee Han, David Watson, and Farnam Jahanian. Topology aware overlay networks. In *INFOCOM '05: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*.

[46] Sandra M. Hedetniemi, Stephen T. Hedetniemi, and Arthur L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18:319–349, 1988.

[47] Dorit S. Hochbaum. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1996.

[48] Mikel Izal, Guillaume Urvoy-Keller, Ernst W Biersack, Pascal A Felber, Anwar Al Hamra, and Luis Garces-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *PAM '04: Proceedings of the 5th International Workshop of Passive and Active Network Measurement*.

[49] Kamal Jain and Vijay V. Vazirani. Primal-Dual Approximation Algorithms for Metric Facility Location and k-Median Problems. In *FOCS '99: The 40th Annual Symposium on Foundations of Computer Science*.

[50] Manish Jain and Constantinos Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput. *IEEE/ACM Transactions on Networking*, 11(4):537–549, 2003.

[51] Shudong Jin and Azer Bestavros. Small-World Internet Topologies: Possible Causes and Implications on Scalability of End-System Multicast. Technical Report BUCS-TR-2002-004, CS Department, Boston University, January 30 2002.

[52] Shudong Jin and Azer Bestavros. Small-world characteristics of internet topologies and implications on multicast scaling. *Computer Networks*, 50(5):648–666, 2006.

[53] O. Kariv and S.L. Hakimi. An Algorithmic Approach to Network Location Problems, Part II: p-medians. *SIAM Journal on Applied Mathematics*, 37:539–560, 1979.

[54] Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. In *SODA '98: Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*.

[55] Dejan Kostic, Adolfo Rodriguez, Jeannie Albrecht, and Amin Vahdat. Bullet: high bandwidth data dissemination using an overlay mesh. In *SOSP '03: Proceedings of the 19th ACM Symposium on Operating Systems Principles*.

[56] P. Krishnan, Danny Raz, and Yuval Shavit. The cache location problem. *IEEE/ACM Transactions on Networking*, 8(5):568–581, 2000.

[57] Rakesh Kumar and Keith W. Ross. Optimal Peer-Assisted File Distribution: Single and Multi-Class Problems. In *HotWeb '06: Proceedings of the 1st IEEE Workshop on Hot Topics in Web Systems and Technologies*.

[58] Anukool Lakhina, John Byers, Mark Crovella, and Peng Xie. Sampling biases in IP topology measurements. In *Proc. of IEEE INFOCOM '03*, April 2003.

[59] Nikolaos Laoutaris, Laura Poplawski, Rajmohan Rajaraman, Ravi Sundaram, and Shang-Hua Teng. A Bounded-Degree Network Formation Game. In *PODC '08: Proceedings of the 27th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing.*

[60] Nikolaos Laoutaris, Pablo Rodriguez, and Laurent Massoulie. ECHOS: edge capacity hosting overlays of nano data centers. *ACM SIGCOMM Computer Communication Review*, 38(1):51–54, 2008.

[61] Nikolaos Laoutaris, Georgios Smaragdakis, Azer Bestavros, Ibrahim Matta, and Ioannis Stavrakakis. Distributed Selfish Caching. *IEEE Transactions on Parallel and Distributed Systems*, 18(10):1361–1376, October 2007.

[62] Nikolaos Laoutaris and Ioannis Stavrakakis. Intrastream synchronization for continuous media streams: A survey of playout schedulers. *IEEE Network Magazine*, 16(3), May 2002.

[63] Nikolaos Laoutaris, Orestis Telelis, Vassilios Zissimopoulos, and Ioannis Stavrakakis. Distributed Selfish Replication. *IEEE Transactions on Parallel and Distributed Systems*, 17(12):1401–1413, 2006.

[64] Vito Latora and Massimo Marchiori. Economic small-world behavior in weighted networks. *The European Physical Journal B*, 32:249–263, 2003.

[65] Jonathan Ledlie, Peter Pietzuch, and Margo Seltzer. Network Coordinates in the Wild. In *NSDI '07: Proceedings of the 4th USENIX Symposium on Networked Systems Design & Implementation.*

[66] Arnaud Legout, G. Urvoy-Keller, and P. Michiardi. Rarest first and choke algorithms are enough. In *IMC '06: Proceedings of the 2006 ACM SIGCOMM Internet Measurement Conference.*

[67] Vincent Lenders, Martin May, and Bernhard Plattner. Density-based vs. Proximity-based Anycast Routing for Mobile Networks. In *INFOCOM '06: Proceedings of the 25th IEEE International Conference on Computer Communications.*

[68] Xin Li, Fang Bian, Mark Crovella, Christophe Diot, Ramesh Govindan, and Gianluca Iannaccone. Detection and identification of network anomalies. In *IMC '06: Proceedings of the 2006 ACM SIGCOMM Internet Measurement Conference.*

[69] Zhi Li and Prasant Mohapatra. Impact of Topology On Overlay Routing Service. In *INFOCOM '04: Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies.*

[70] Zhi Li and Prasant Mohapatra. QRON: QoS-aware routing in overlay networks. *IEEE JSAC*, 22(1):29–40, Jan 2004.

[71] Jian Liang, Rakesh Kumar, and Keith W. Ross. The FastTrack overlay: A measurement study. *Computer Networks*, 50(6):842–858, 2006.

[72] Jyh-Han Lin and Jeffrey Scott Vitter. ϵ-Approximations with Minimum Packing Constraint Violation. In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of Computing.*

[73] Yong Liu, Honggang Zhang, Weibo Gong, and Donald F. Towsley. On the interaction between overlay routing and underlay routing. In *INFOCOM '05: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*

[74] Thanasis Loukopoulos, Petros Lampsas, and Ishfaq Ahmad. Continuous Replica Placement Schemes in Distributed Systems. In *ICS '05: Proceedings of the 19th annual international conference on Supercomputing.*

[75] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proceedings of the 15th annual international conference on Supercomputing.*

[76] Ratul Mahajan, Neil Spring, David Wetherall, and Tom Anderson. Inferring link weights using end-to-end measurements. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment.*

[77] Mohammad Mahdian and Martin Pal. Universal facility location. In *ESA '03: Proceedings of the 11th Annual European Symposium.*

[78] Gurmeet Singh Manku, Moni Naor, and Udi Wieder. Know thy neighbor's neighbor: the power of lookahead in randomized P2P networks. In *STOC '04: Proceedings of the 26th annual ACM symposium on Theory of computing.*

[79] Laurent Massoulie, Andy Twigg, Christos Gkantsidis, and Pablo Rodriguez. Randomized decentralized broadcasting algorithms. In *INFOCOM '07: Proceedings of the 26th Annual IEEE Conference on Computer Communications.*

[80] Laurent Massoulie and Milan Vojnovic. Coupon replication systems. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurements and Modeling of Computer Systems.*

[81] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. BRITE: An Approach to Universal Topology Generation. In *MASCOTS '01: Proceedings of the 9th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems.*

[82] Adam Meyerson. Online Facility Location. In *FOCS '01: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science.*

[83] Pitu B. Mirchandani and Richard L. Francis. *Discrete Location Theory.* John Wiley and Sons, 1990.

[84] Thomas Moscibroda, Stefan Schmid, and Roger Wattenhofer. On the topologies formed by selfish peers. In *PODC '06: Proceedings of the 25th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing.*

[85] Thomas Moscibroda and Roger Wattenhofer. Facility location: distributed approximation. In *PODC '05: Proceedings of the 24th annual ACM symposium on Principles of distributed computing*.

[86] Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[87] Konstantinos Oikonomou and Ioannis Stavrakakis. Service migration: The tree topology case. In *Med-Hoc-Net '06: Proceedings of the 5th IFIP Annual Mediterranean Ad Hoc Networking Workshop*.

[88] David Oppenheimer, Brent Chun, David Patterson, Alex C. Snoeren, and Amin Vahdat. Service placement in a shared wide-area platform. In *USENIX'06: Proceedings of the 2006 USENIX Annual Technical Conference*.

[89] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[90] Steven Osman, Dinesh Subhraveti, Gong Su, and Jason Nieh. The design and implementation of Zap: a system for migrating computing environments. In *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*.

[91] Jianping Pan, Y. Thomas Hou, and Bo Li. An overview dns-based server selection in content distribution networks. *Computer Networks*, 43(6), 2003.

[92] Larry L. Peterson and Bruce S. Davie. *Computer Networks: A Systems Approach, 3rd Edition*. Morgan Kaufmann Publishers Inc., 2003.

[93] Dongyu Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *SIGCOMM '04: Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*.

[94] Lili Qiu, Yang Richard Yang, Yin Zhang, and Scott Shenker. On Selfish Routing in Internet-like Environments. In *SIGCOMM '03: Proceedings of the ACM SIGCOMM 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*.

[95] Michael Rabinovich and Amit Aggarwal. RaDaR: A scalable architecture for a global web hosting service. In *WWW '99: Proceedings of the 8th International World Wide Web Conference*.

[96] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Topologically aware overlay construction and server selection. In *INFOCOM '02: Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*.

[97] Sean Rhea, Daniel Geels, Timothy Roscoe, and John Kubiatowicz. Handling Churn in a DHT. In *USENIX '04: Proceedings of the 2004 USENIX Annual Technical Conference*.

[98] Vinay Ribeiro, Rudolf Riedi, Richard Baraniuk, Jiri Navratil, and Les Cottrell. pathChirp: Efficient Available Bandwidth Estimation for Network Paths. In *PAM '03: Proceedings of the 4th International Workshop of Passive and Active Network Measurement*.

[99] Bruno Gusmao Rocha, Virgilio Almeida, and Dorgival Guedes. Improving reliability of selfish overlay networks. In *WWW '06: Proceedings of the 15th International World Wide Web Conference*, Edinbuurgh, UK, 2006.

[100] Tim Roughgarden and Éva Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2):236–259, 2002.

[101] Stefan Savage, Thomas Anderson, Amit Aggarwal, David Becker, Neal Cardwell, Andy Collins, Eric Hoffman, John Snell, Amin Vahdat, Geoff Voelker, and John Zahorjan. Detour: Informed Internet routing and transport. *IEEE Micro*, 19(1):50–59, 1999.

[102] Srinivasan Seetharaman and Mostafa Ammar. On the Interaction between Dynamic Routing in the Overlay and Native Layers. In *INFOCOM '06: Proceedings of the 25th IEEE International Conference on Computer Communications*.

[103] Alok Shriram, Margaret Murray, Young Hyun, Nevil Brownlee, Andre Broido, Marina Fomenkov, and Kimberly C. Claffy. Comparison of Public End-to-End Bandwidth Estimation Tools on High-Speed Links. In *PAM '05: Proceedings of the 6th International Workshop of Passive and Active Network Measurement*.

[104] Aameek Singh, Madhukar Korupolu, and Bhuvan Bamba. SPARK: Integrated Resource Allocation in Virtualization-Enabled SAN Data Centers. IBM Research Report RJ10407.

[105] Atul Singh, Miguel Castro, Peter Druschel, and Antony Rowstron. Defending against eclipse attacks on overlay networks. In *Proc. of ACM SIGOPS European Workshop*, page 21, 2004.

[106] Georgios Smaragdakis, Nikolaos Laoutaris, Pietro Michiardi, Azer Bestavros, John W. Byers, and Mema Roussopoulos. Swarming on Optimized Graphs for n-way Broadcast. In *INFOCOM '08: Proceedings of the 27th Annual IEEE Conference on Computer Communications*.

[107] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson. Measuring ISP topologies with rocketfuel. *IEEE/ACM Transanctions on Networking*, 12(1):2–16, 2004.

[108] Srinivasan Seetharaman and Voler Hilt and Markus Hofmann and Mostafa Ammar. Preemptive strategies to improve routing performance of native and overlay layers. In *INFOCOM '07: Proceedings of the 26th Annual IEEE Conference on Computer Communications*.

[109] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, , M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, 2003.

[110] Daniel Stutzbach and Reza Rejaie. Characterizing the two-tier Gnutella topology. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurements and Modeling of Computer Systems.*

[111] Daniel Stutzbach, Reza Rejaie, Nick Duffield, Subhabrata Sen, and Walter Willinger. On unbiased sampling for unstructured peer-to-peer networks. In *IMC '06: Proceedings of the 2006 ACM SIGCOMM Internet Measurement Conference.*

[112] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H. Katz. Characterizing the internet hierarchy from multiple vantage points. In *INFOCOM '02: Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies.*

[113] Lakshminarayanan Subramanian, Ion Stoica, Hari Balakrishnan, and Randy H. Katz. OverQoS: offering internet QoS using overlays. *SIGCOMM Computer Communication Review*, 33(1):11–16, 2003.

[114] Ye Tian, Di Wu, and Kam-Wing Ng. Analyzing multiple file downloading in bittorrent. In *ICPP '06: Proceedings of the 2006 International Conference on Parallel Processing.*

[115] Vivek Vishnumurthy and Paul Francis. A comparison of structured and unstructured p2p approaches to heterogeneous random peer selection. In *USENIX '07: Proceedings of the 2007 USENIX Annual Technical Conference.*

[116] Bernard M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, 1988.

[117] Lidia Yamamoto and Guy Leduc. Autonomous reflectors over active networks: towards seamless group communication. *The Interdisciplinary Journal of Artificial Intelligence & the Simulation of Behaviour*, 1(1):125–146, 2001.

[118] Zhongmei Yao, Derek Leonard, Xiaoming Wang, and Dmitri Loguinov. Modeling Heterogeneous User Churn and Local Resilience of Unstructured P2P Networks. In *ICNP '06: Proceedings of the 14th IEEE International Conference on Network Protocols.*

[119] Anthony Young, Jiang Chen, Zheng Ma, Arvind Krishnamurthy, Larry L. Peterson, and Randy Wang. Overlay Mesh Construction Using Interleaved Spanning Trees. In *INFOCOM '04: Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies.*

[120] Honggang Zhang, Giovanni Neglia, Don Towsley, and Giuseppe Lo Presti. On unstructured file sharing networks. In *INFOCOM '07: Proceedings of the 26th Annual IEEE Conference on Computer Communications*.

[121] Yong Zhu, Constantinos Dovrolis, and Mostafa H. Ammar. Dynamic overlay routing based on available bandwidth estimation: A simulation study. *Computer Networks*, 50(6):742–762, 2006.

[122] Artur Ziviani, Serge Fdida, José Ferreira de Rezende, and Otto Carlos Muniz Bandeira Duarte. Toward a measurement-based geographic location service. In *PAM '04: Proceedings of the 5th International Workshop of Passive and Active Network Measurement*.

Curriculum Vitae

EDUCATION

Ph.D. Candidate in Computer Science, Boston University, 2003-present.
Dissertation Title: *"Overlay Network Creation and Maintenance with Selfish Users"*.
Dissertation Committee: Azer Bestavros, Nikolaos Laoutaris, John W. Byers.

Diploma in Electronic and Computer Engineering,
Technical University of Crete, 2002.

HONORS and AWARDS

Boston University Graduate Fellow, 2003 to present

Honorable Mention Award from the Center for Information and Systems Engineering - Best Posters presented at the Boston University Science and Engineering Research Symposium, 2007.

2^{nd} Place Award - Best Posters for Boston University/Computer Science Research Day, 2007.

Honorable Mention Award from the Center for Information and Systems Engineering - Best Posters presented at the Boston University Science and Engineering Research Symposium, 2006.

2^{nd} Place Award - Best Posters for Boston University/Computer Science Research Day, 2006.

1^{st} ERICSSON Award of Excellence in Telecommunications, 2003.

Graduating ranking second, Class of 2002, Technical University of Crete, 2002.

Annual Scholarship of the Technical Chamber of Greece , 1999-2000.

Annual Scholarship of the Greek National Fellowship Foundation, 1998-2000.

Academic Excellence Award, Greek Ministry of Labor and Social Affairs, 1998-2000.

Annual Scholarship of the Paidia Foundation, 1999-2000.

Honorary Diploma of the municipality of Piraeus for the Social, Environmental and Economic analysis of the City of Piraeus, 1996. 1^{st} Awards of Excellent Studies from Greek Ministry of Education, 1990-1996.

PUBLICATIONS

REFEREED JOURNALS

I. C. Paschalidis and G. Smaragdakis, Spatio-Temporal Network Anomaly Detection by Assessing Deviations of Empirical Measures, IEEE/ACM Transactions on Networking, [to appear].

N. Laoutaris, G. Smaragdakis, A. Bestavros, I. Matta, and I. Stavrakakis, Distributed Selfish Caching, IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 10, pp. 1361–1376, October 2007.

G. Smaragdakis, N. Laoutaris, A. Bestavros, I. Matta, and I. Stavrakakis, Mistreatment-Resilient Distributed Caching, Computer Networks, vol. 51, no. 11, pp. 2917–2937, August 2007.

REFEREED CONFERENCES

G. Smaragdakis, N. Laoutaris, P. Michiardi, A. Bestavros, J. W. Byers, and M. Roussopoulos, Swarming on optimized graphs for n-way broadcast, in Proceedings of IEEE INFOCOM 2008, Phoenix, AZ, April 2008.

N. Laoutaris, G. Smaragdakis, A. Bestavros, and J. W. Byers, Implications of Selfish Neighbor Selection in Overlay Networks, in Proceedings of IEEE INFOCOM 2007, Anchorage, AK, May 2007.

N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros, Distributed Placement of Service Facilities in Large-Scale Networks, in Proceedings of IEEE INFOCOM 2007, Anchorage, AK, May 2007.

N. Laoutaris, G. Smaragdakis, A. Bestavros, and I. Stavrakakis, Mistreatment in Distributed Caching Groups: Causes and Implications, in Proceedings of IEEE INFOCOM 2006, Barcelona, Spain, April 2006.

G. Smaragdakis, N. Laoutaris, I. Matta, A. Bestavros, and I. Stavrakakis, A Feedback Control Approach to Mitigating Mistreatment in Distributed Caching Groups, in Proceedings of IFIP Networking 2006, Coimbra, Portugal, May 2006.

I. C. Paschalidis and G. Smaragdakis, A Large Deviations Approach to Statistical Traffic Anomaly Detection, in Proceedings of IEEE CDC 2006, San Diego, CA, December 2006.

D. Barman, G. Smaragdakis, and I. Matta, The Effect of Router Buffer Size on HighSpeed TCP Performance, in Proceedings of IEEE Globecom 2004 - Global Internet and Next Generation Networks, Dallas, TX, December 2004.

G. Smaragdakis, I. Matta, and A. Bestavros, SEP: A Stable Election Protocol for clustered heterogeneous wireless sensor networks, in Proceedings of Second International Workshop on Sensor and Actor Network Protocols and Applications (SANPA 2004), Boston, MA, August 2004.

UNDER SUBMISSION

G. Smaragdakis, N. Laoutaris, A. Bestavros, J. W. Byers, Selfish Overlay Network Formation, Under Submission, 2008.

G. Smaragdakis, V. Lekakis, N. Laoutaris, A. Bestavros, J. W. Byers, and M. Roussopoulos, EGOIST: Overlay Routing using Selfish Neighbor Selection, Under Submission, 2008.

G. Smaragdakis, N. Laoutaris, K. Oikonomou, I. Stavrakakis, and A. Bestavros, Distributed Server Migration for Scalable Internet Service Deployment, Under Submission, 2008.

N. Laoutaris, G. Smaragdakis, R. Sundaram, and P. Rodriguez, Delay-Tolerant Bulk Data Transfer on the Internet, Under Submission, 2008.

THESES

G. Smaragdakis, TCP Performance over UMTS Network, Diploma Thesis, Electronic and Computer Engineering Department, Technical University of Crete, October 2002.

PROFESSIONAL SERVICE

IEEE HotWeb 2006 (Publication Chair), IEEE ICNP 2005 (Web Administrator), PAM 2005 (Local Arrangements), IEEE ASWN 2004 (Publication Chair).

Technical Program Committee Member for: ACM CoNEXT 2008 (Shadow).

Reviewer for ACM SIGCOMM Computer Communication Review, Journal of Computer Networks, IEEE INFOCOM (2007, 2006, 2005), ACM SIGMETRCS (2008, 2007), ACM PODC 2008, IEEE ICNP (2005, 2004), IEEE Global Internet Symposium 2007, IEEE ICDCS (2003), IEEE ICC (2005, 2004), IEEE RTSS 2004, ACM Multimedia 2004, IEEE PIMRC 2005.

INVITED TALKS

"Selfish Overlay Network Formation: Resource Allocation Strategies and Implications to Protocol Design", Centre Tecnològic de Telecomunicacions de Catalunya (04/2008); Deutsche Telekom Laboratories Berlin (03/2008); Telefónica Research Barcelona (03/2008); Boston University (12/2007).

"Resource Allocation Strategies for Scalable Content Delivery on the Internet", Boston University (10/2007).

"The Selfish Neighbor Selection Problem In Overlay Networks", HotWeb (11/2006), University of Athens (07/2007).

"A Large Deviations Approach to statistical Traffic Anomaly Detection", Boston University (02/2006).

RESEARCH EXPERIENCE

Research Intern, Telefónica Research, Barcelona, January-May 2008.

Research and Teaching Fellow, Boston University, Computer Science Department, Web and Internetworking Group, September 2003 to present.

Affiliated Researcher, Greek National Center for Scientific Research, Institute of Informatics and Telecommunications, April - August 2003.

Undergraduate Affiliated Student, Technical University of Crete, Electronic and Computer Engineering Department, Information and Computer Networks Laboratory, September 2001 - August 2002.

TEACHING EXPERIENCE

Teaching Fellow for: Fundamentals of Computing Systems (CS 350; Spring 2007), Introduction to Data Structures (CS 112; Spring 2005), Quantitative Methods for Information Systems (MET CS 546; Summer 2005), Introduction to Computers (CS 101; Spring 2006, Fall 2005, Spring 2004).

PROFESSIONAL EXPERIENCE

Telecommunications Engineer, Value Added Services, NOKIA Networks, NOKIA Hellas, August - September 2002.

Software Developer, Technical University of Crete, Electronic and Computer Engineering Department, Laboratory of Distributed Multimedia, Information Systems and Applications, September 2001 - February 2002.

Programmer, Social Security Institute, Greece, August - September 1999, August 2001.