DISSERTATION

# Adaptive Multi-Path Routing for Internet Traffic Engineering

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften

eingereicht an der
Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von

Dipl.-Ing. Ivan Gojmerac

Wien, April 2007


Technische
Universität
Wien

Vienna
University of
Technology

Supervisors

Prof. Dr.-Ing. Dietmar Dietrich
Institute of Computer Technology
Vienna University of Technology

and

Prof. Dr.-Ing. Ralf Lehnert
Chair for Telecommunications
Department of Electrical Engineering and Information Technology
Dresden University of Technology

and

Dr. rer. nat. Peter Reichl
Telecommunications Research Center Vienna (ftw.)

ftw. Dissertation Series

Ivan Gojmerac

# Adaptive Multi-Path Routing for Internet Traffic Engineering


telecommunications research center vienna

This work was carried out with funding from **K***plus* in the ftw. project N0.

This thesis has been prepared using LaTeX.

September 2007
1. Auflage

Herausgeber: Forschungszentrum Telekommunikation Wien
Printed in Austria
ISBN 3-902447-14-8

*To my parents*

*Mirko and the memory of Neda Gojmerac*

# Abstract

The present thesis proposes Adaptive Multi-Path routing (AMP) as a novel algorithm for dynamic traffic engineering in the Internet. The main objective of AMP is to distribute load within a network domain in a continuous manner, trying to offload congested links and paths by responding to the observed traffic conditions in real-time. In contrast to similar multi-path routing proposals, the central innovation of AMP lies in the restriction of the exchange of load information to a local scope, thus achieving a significant reduction of signaling overhead and memory consumption in the routers. Most importantly, in spite of its local nature, AMP's signaling mechanism nevertheless enables global propagation of congestion information, leading to an efficient handling of overload events in the network. Apart from providing an extensive overview of the related theory and approaches, this thesis introduces and discusses AMP in depth, after which an exhaustive performance evaluation of the algorithm is presented. This evaluation is carried out using two distinct simulation environments, which model IP networks either at the level of individual packets or individual flows, and for both of which the full functionality of AMP has been implemented in detail. Furthermore, the investigated simulation scenarios represent the state-of-the-art concerning the choice of topologies and traffic models, thus offering a comprehensive insight into the algorithm's behavior in real systems. The results of the performance evaluations consistently demonstrate AMP's efficiency in terms of load balancing performance, and they also underline the stable behavior of the algorithm throughout the investigated scenarios. Finally, the thesis discusses traffic engineering in the current context of IP networks on a more general level, before it presents important potential areas of application for AMP in novel and emerging networking solutions and architectures.

# Kurzfassung

Die vorliegende Arbeit stellt das *Adaptive Multi-Path* Routing (AMP) als einen neuen Algorithmus für dynamisches Traffic Engineering im Internet vor. Das Hauptziel des AMP besteht darin, die Last innerhalb einer Netzdomäne dynamisch zu verteilen, um die überlasteten Verbindungen und Pfade in Realzeit, entsprechend der momentanen Verkehrssituation und den vorhandenen Ausweichmöglichkeiten, zu entlasten. Im Gegensatz zu verwandten Ansätzen, die den Verkehr auf mehrere Pfade aufteilen, führt AMP als seine wichtigste Innovation die Einschränkung der Lastsignalisierung auf den lokalen Umkreis einer Verbindung ein und ermöglicht somit sowohl eine Verringerung der Signalisierungslast, als auch des Speicherbedarfs in den Routern. Dabei ist hervorzuheben, dass AMP trotz seiner lokalen Signalisierungsarchitektur eine globale Ausbreitung der Lastinformation innerhalb der gesamten Netzdomäne ermöglicht und so zu einer effizienten Behandlung von Überlastsituationen im Netz führt. Darüber hinaus stellt diese Arbeit sowohl einen ausführlichen Überblick über die relevante Theorie und die daraus abgeleiteten Ansätze vor, als auch eine tiefgehende Leistungsbewertung des Algorithmus in zwei unterschiedlichen Simulationsumgebungen, die IP-Netze entweder auf Paket- oder auf Flussebene abbilden und in denen jeweils die gesamte Funktionalität von AMP implementiert wurde. Die untersuchten Simulationsszenarien stellen dabei den letzten Stand der Technik dar, indem ausgeklügelte Topologie- und Verkehrsmodelle verwendet werden, um eine hohe Übertragbarkeit der Erkenntnisse in die Realität zu ermöglichen. Die Ergebnisse der durchgeführten Leistungsbewertung demonstrieren in konsistenter Weise die Effizienz der durch AMP erzielten Lastverteilung, und sie unterstreichen auch das stabile Verhalten des Algorithmus in allen untersuchten Szenarien. Anschließend wird in einer etwas allgemeineren Betrachtung die momentane Bedeutung von Traffic Engineering im aktuellen Kontext der IP-Netze diskutiert, wonach zum Schluss noch wichtige potentielle Anwendungsgebiete für AMP im Bereich von neuen Netzlösungen und -architekturen dargestellt werden.

# Preface

During my time in the MIOC High School in Zagreb, Croatia, in the mid 1990ies, I was fortunate to witness two crucial developments in the world of telecommunications at a relatively early age: the emergence of the Internet as a novel medium for free world-wide data communication and the deployment of second generation digital cellular networks, both of which rapidly gained momentum in the broad consumer market. Being a technological enthusiast by nature, I was immediately taken by the seemingly limitless capabilities offered by these new means of communication, and I very soon decided to make them the center of my professional interest.

As a natural next step, I went on to study Electrical Engineering at the University of Zagreb with the focus set upon telecommunications, and after completing my studies in 2001 I was encouraged by my professors to continue my studies towards a Ph.D. degree. Accordingly, during the same year I enrolled as Ph.D. student at Vienna University of Technology, and started participating in scientific projects at the Telecommunications Research Center Vienna (ftw.).

After exploring different potential research topics in the area of communication networks, several months into my studies I finally settled down with the topic of Internet routing and traffic engineering. Among all the various research directions, this topic exerted the greatest fascination on me, as it dealt most directly with the IP layer which represents the central building block of the future converged communications backbone. In other words, while it was obvious that lower layer technologies, and especially the application layer technologies were about to experience continuous changes during the course of the next years, I considered myself very lucky to work on a layer which in comparison apparently displayed a perspective of perpetual stability. Essentially, the topic of routing and traffic engineering is to a certain extent independent of the underlying technologies, as it is mainly based upon a number of seminal results in the areas of graph theory and optimization algorithms. However, the relative maturity and stability of the results also implies that it is more difficult to come up with substantial innovations and scientific advances, making research in this area a very risky undertaking.

Nevertheless, I immediately liked the idea of such a challenge, which of course greatly contributed to my research enthusiasm. After an extensive survey of the existing literature and approaches in intra-domain IP routing and traffic engineering, I came up with an original proposal for a simple and light-weight approach to improving the allocation of traffic within individual IP domains, and named it Adaptive Multi-Path routing (AMP). Accordingly, the formulation, the extensive performance evaluation of AMP, and the identification of the most important application scenarios represent the central contributions of this Ph.D. thesis.

I owe lots of thanks to many people who supported me during the course of my Ph.D. I want to thank my first supervisor, Professor Dietmar Dietrich of Vienna University of Technology, for strongly encouraging my scientific progress, and for providing me with numerous valuable suggestions during the course of writing my thesis. Beyond his technical contributions, I am also grateful for his excellent advice concerning style, which puts great emphasis on European traditions in scientific writing, and which largely helped increase the overall quality of my thesis.

I would also like to express many thanks to my second supervisor, Professor Ralf Lehnert of Dresden University of Technology, whose exhaustive experience and expertise in the area of communication networks strongly contributed to the quality of my work. I am especially grateful for his detailed and precisely pinpointed comments and suggestions concerning AMP, which greatly helped me in focusing my research upon several selected aspects of the algorithm.

My special gratitude goes to my mentor at ftw., Dr. Peter Reichl, who always believed in me and my work, and who pivotally backed me in several crucial moments of my Ph.D. studies – my continuous and focused work on AMP would not have been possible without his wholehearted support. He never spared time or effort for in-depth technical discussions about my research, thereby generously enabling me to benefit from his vast professional experience.

Furthermore, I would like to cordially thank Professor Markus Rupp, Dean of the Faculty of Electrical Engineering and Information Technology at Vienna University of Technology, for his support concerning many practical aspects of my Ph.D. studies.

I also owe special thanks to the Telecommunications Research Center Vienna (ftw.), and particularly to its CEOs Dr. Markus Kommenda and M.Sc. Horst Rode for providing me with a great scientific and social environment for my Ph.D. research in the last years. I would also like to thank all colleagues of ftw.'s projects A0 and N0 for their kind support, and particularly its project managers Dr. Thomas Ziegler and Dr. Fabio Ricciato. My special gratitude also goes to my colleague M.Sc. Lasse Jansen for our excellent collaboration in the area of flow-level network simulation techniques.

Coming back to the beginnings of my Ph.D. studies, I would like to thank Professor Maja Matijašević of the University of Zagreb, who encouraged me to proceed towards a doctoral degree in electrical engineering, as well as Dr. Igor Brusić and Dr. Sandford Bessler for paving the way for my studies at Vienna University of Technology.

Last, but not least, I would like to thank all of my family and friends for their enormous support and patience during the last years. Without them this Ph.D. would not have been possible.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Since the early days of packet switched networks the routing of traffic has represented one of the core topics in networking research and development [1]. The most important problems which had to be dealt with were initially associated to the basic task of finding routes between pairs of remote nodes in the network. Therefore, different routing architectures have been explored, with different types of addressing and path calculation, ranging e.g. from connection oriented hierarchical approaches similar to circuit switched telephone networks [2], to connectionless architectures which tolerate non-hierarchical and highly irregular network structures [3].

As far as the Internet is concerned, the routing architecture within individual administrative domains has practically not changed since the advent of the ARPAnet [1]: a scalar value called the *link weight* is associated to each link in the network, and the paths between a pair of nodes are determined such that the sum of their link weights is minimized. This basically reduces the routing problem in the network to finding shortest paths in a directed weighted graph, which represents a well known problem from graph theory, with a number of corresponding algorithmic solutions [4, 5, 6].

Based upon these fundamental algorithms from graph theory, routing protocols have been developed which have the basic tasks of setting up the network wide routes automatically (i.e. without any need for manual configuration), and quickly reconfiguring the routes in the case of topological changes. Accordingly, two large groups of protocols have emerged: *distance vector* protocols based upon the theoretical work by Bellman, Ford, and Fulkerson [4, 5], and *link state* protocols based upon the algorithm by Dijkstra [6]. Due to their better properties concerning routing convergence times, link state protocols have eventually prevailed, leading to the development of the currently most widely deployed protocols like Open Shortest Path First (OSPF) [7, 8] and Intermediate System to Intermediate System (IS-IS) [9].

As the problem of ensuring network wide connectivity has largely been solved, the focus of research has gradually shifted towards issues related to the distribution of traffic

throughout the network. The main reason for this shift lies in the traffic agnostic nature of routing with the current protocols, which base their routing decisions exclusively upon the network topology, and thereby completely disregard the actual traffic demands and link utilizations. In many cases this can represent a major drawback, as certain parts of the network may become congested due to excessive traffic load, and at the same time topologically redundant parts of the network may be underloaded, with the total effect of wasting large portions of the installed network capacity.

In order to counteract this problem, the discipline of *traffic engineering* has been introduced, which aims at optimizing the performance of operational networks [10]. In contrast to *network design* and *network dimensioning*, the main working assumption of traffic engineering is that the network topology and the installed link capacities are invariantly determined. Accordingly, traffic engineering can only act upon the routing of traffic in order to improve network performance by trying to prevent the emergence of individual congestion hotspots.

As traffic engineering is based upon routing, the introduced performance improvements can be observed only at a medium time scale of minutes and hours. In other words, traffic engineering does not hold the potential of combating short term congestion which may occur due to a sudden surge of traffic demands. Such phenomena occurring at the time scale of seconds are rather a case for Quality of Service (QoS) mechanisms like e.g. different packet scheduling and queueing strategies. At the other extreme, traffic engineering is also not suited for alleviating long term congestion, which typically results from a steady growth in traffic demands; at the time scale of multiple days, weeks, and months performance degradations should be counteracted by carefully planned network capacity upgrades, and thus they fall into the domain of network planning.

Similarly to the evolution observed in the area of routing algorithms and protocols, many different approaches for traffic engineering in Internet Protocol (IP) based networks have been proposed in recent years and they have undergone careful evaluation in the scientific community. Interestingly, the work performed in the area of IP traffic engineering has resulted in a plethora of approaches which significantly differ in many aspects, sometimes requiring even completely novel types of underlying networking technologies. Whereas some techniques merely aim at performing traffic engineering by optimizing network configuration, other approaches require new routing and packet forwarding techniques, like e.g. Multi-Protocol Label Switching (MPLS) [11].

Link weight optimization may serve as a typical example of approaches which act merely upon routing configuration. Its main principle is to determine the link weights which will result in a network wide traffic distribution which is optimal with respect to a well defined performance objective, like e.g. minimizing the maximum link utilization in the network. Although link weight optimization does not require the introduction of any new technologies, it does necessitate substantial management overhead as the knowledge of the traffic matrix (or even better the traffic demand matrix) is a prerequisite for

performing the optimization procedure. Additionally, the calculated link weights must be introduced into the network, which adds further overhead to the procedure. And as the traffic (demand) matrix typically displays time-of-day effects, the whole process of traffic measurements, optimization, and the introduction of the newly calculated link weights must be performed periodically, which practically makes an automated network management framework mandatory for this approach.

Traffic engineering techniques based upon Multi-Protocol Label Switching (MPLS) are in many ways analogous to link weight optimization, as they mostly also require traffic measurements, optimization procedures, and a framework for periodically introducing the calculated configuration to the network elements. The greatest difference between MPLS and link weight optimization is that MPLS does not use the plain intra-domain IP routing architecture, but rather employs a packet forwarding infrastructure of its own, thus imposing strict technological requirements upon the capabilities of the network elements.

Unlike link weight optimization and MPLS, which act at the level of network configuration, an alternative approach performs traffic engineering within the framework of the routing algorithms and protocols. Using this paradigm, traffic engineering does not any more represent only a configuration add-on. Instead, it is fully integrated into standard network operation, performing the necessary actions completely autonomously within the nodes, thus not requiring any additional network management efforts. The recently proposed Optimized Multi-Path algorithm (OMP) may serve as a good example of this type of approach [12]. The main idea of OMP is to employ the message flooding framework of link state routing algorithms for disseminating link load information throughout the network. Based upon the information received from all other nodes in the network about topology and link load, each node can autonomously determine how to distribute the traffic for individual destinations among the available next hop nodes. Therefore, the main prerequisite of OMP is that multiple next hop neighbors for each destination exist in as many nodes as possible, i.e. the existence of multiple paths is fundamental to its operation. While OMP does have favorable properties like completely autonomous operation, at the same time it imposes substantial requirements upon the memory consumption in routers, and it also produces indeterministic signaling overhead by disseminating load information using the message flooding mechanism.

Aiming at combining the most favorable properties of the mentioned approaches, the main objective of this thesis is to explore, analyze, and evaluate the possibilities for designing traffic engineering techniques which operate autonomously within the nodes of the network, and which produce only a minimal overhead in terms of signaling load, memory consumption in routers, and computational complexity. The envisioned solution should be easily deployable in networks using the current routing architecture, without introducing completely novel switching or signaling techniques, and additionally it is supposed to be easily implementable in state of the art networking hardware.

## 1.2 Main Contributions and Results

Having stated the design and evaluation of an efficient traffic engineering solution as the main objective of this work, this goal is achieved by proposing a novel routing algorithm for dynamic load balancing called Adaptive Multi-Path routing (AMP). Similarly to OMP, AMP operates completely autonomously within the network nodes without requiring any manual interventions, with the main prerequisite for its operation also being the presence of multiple paths between the communication pairs in the network. In response to the current traffic conditions in the network (i.e. to the instantaneous traffic matrix and congestion level) AMP shifts traffic among the multiple paths in order to alleviate congestion, eventually reaching a fix point in the global (i.e. network-wide) traffic distribution. However, while OMP employs global signaling of load information, AMP exclusively relies upon local exchange of load information between neighbor nodes, thus reducing the view of the network in each node to the local scope, and obsoleting global message passing procedures. The central innovation of AMP lies in the quasi-recursive property of the signaling algorithm, which simultaneously enables seemingly contradictory goals of global propagation of load information throughout the network domain, while reducing message exchange only to neighbor nodes.

Focusing upon AMP performance and characteristics, this work yields a number of original contributions, including:

- The introduction and concise algorithmic formulation of Adaptive Multi-Path routing (AMP) [13, 14, 15, 16],
- Performance evaluation of AMP in a flow-level network simulator which has been specifically designed for multi-path routing scenarios,
- Performance evaluation of AMP in the standard *ns-2* packet-level network simulator [www-1],
- Investigation and discussion of stability issues,
- A survey of possible applications of AMP in current IP networks, with a special focus on application level overlays and wireless mesh networks.

In order to achieve these results, first an in-depth review of the related work in the areas of IP routing and traffic engineering has been performed, starting with an overview of the main principles and results from graph theory, followed by a survey of related routing algorithms and protocols. Furthermore, the major traffic engineering techniques already mentioned in this introduction have been thoroughly inspected, especially focusing on the approaches most closely related to the proposed AMP algorithm, i.e. the OMP technique and the minimum delay routing algorithm by Gallager [17].

Based on these insights, the Adaptive Multi-Path algorithm (AMP) including its mechanisms like signaling, multi-path criteria, load balancing, and packet forwarding, has been proposed and evaluated both using flow-level and packet-level simulations.

To this end, a broad variety of simulation scenarios has been provided, with a focus upon creating a large number of synthetic simulation scenarios, based upon which meaningful performance statistics can be derived. In this context, the problems of generating artificial topologies and traffic matrices have been subject to a detailed discussion, as well as the problem of making the appropriate choice of performance metrics. After providing an analysis of the simulation results, the algorithm's performance, stability, and convergence properties have been investigated in detail.

In addition to flow-level simulations, performance evaluations of AMP have also been carried out using the standard *ns-2* packet level simulator, with special focus set upon the Web traffic model used in the simulations. The main results concerning performance and stability have been focused on two real ISP topologies of different size.

In order to round up the present work, several prospective areas of application for the Adaptive Multi-Path algorithm have been evaluated, demonstrating AMP's potential beyond performing traffic engineering for core IP networks with high capacity links. Here, special emphasis has been placed upon IP-based *wireless mesh networks* as the technology with the highest application utility for AMP in the current networking context.

# Chapter 2

# Related Work

Starting with a summary of related work in the area of graph theory, this chapter surveys fundamental approaches relevant for the development of routing protocols and traffic engineering. Additionally, the approaches which are most closely related to the proposed AMP algorithm are examined in a more detailed manner.

## 2.1 Routing Basics

In this section, the origin of the most widely used Internet routing protocols will be examined in detail, beginning with their graph theoretical fundamentals, followed by a description of their important properties, and examples of practical implementations. Due to the vastness of the area, this section does not claim completeness, but rather tries to provide an excerpt of the most important aspects from the perspective of routing and traffic engineering for communication networks.

The Internet is basically a set of individual networks (i.e. individual administrative *domains*), which are today interconnected to the extent that they comprise a global system which appears completely homogeneous and transparent to its users. Figure 2.1.1 provides a high-level overview of current Internet architecture.

There are two fundamental factors which have enabled the huge growth and the enormous global success of the Internet:

- An identical set of networking protocols is used in all networks, converging around the Internet Protocol (IP) which serves as the fundamental vehicle for transporting data between interconnected hosts. IP is based upon the packet switching paradigm, which completely obsoletes the setup of connections as known from the circuit switching world of telephone networks. Instead, IP hosts encapsulate data into packets which are transparently *injected* into the network, knowing that they will arrive at their destinations solely based upon the address declared in the packet header. This simple forwarding paradigm enables enormous simplification

of the core networks, as they do not need to keep state information about individual connections, but rather only need to know the correct routes for reaching the packets' destinations.

- The applied addressing scheme is globally co-ordinated, i.e. unique addresses are assigned to each administrative domain (i.e. IP network) comprising the Internet. Currently, the global address space in the Internet is restricted to 32 bits (IP version 4), enabling a theoretical total of $2^{32}$ = 4,294,967,296 entities to be directly reachable, thus struggling to meet the current global demand, whereas in the proposed successor protocol (IP version 6) this limit should increase to 128 bits, yielding an enormous total number of $2^{128}$ addresses which should easily meet future requirements [18]. The global addressing scheme which has been implemented from the early days of the Internet has enabled gradual and safe interconnection of individual domains, without the risk of addressing conflicts hampering connectivity on the global scale.

With the packet switching and addressing schemes in place, the only missing prerequisite for enabling global connectivity is the presence of correct routing information in the networks' nodes. As configuring routing tables manually is hardly feasible even for networks of very small size, automatic configuration mechanisms called *routing protocols* are needed for disseminating correct routing information throughout a network, yielding two different groups of protocols which differ with respect to their area of application: for disseminating routing information within individual administrative domains so called *intra-domain* routing protocols are used, whereas for the global exchange of routing information so called *inter-domain* routing protocols are required. Although the basic functionality of both groups of protocols is to disseminate reachability information, their practical focus is somewhat different: intra-domain routing protocols primarily focus upon determining optimal routes for individual destinations within a given topology and for a particular routing rule, and they pay special attention to protocol robustness and fast route reconfiguration in the presence of network faults, like e.g. link or router failures. On the other hand, inter-domain protocols focus upon the exchange of reachability information between individual administrative domains (i.e. between networks of different operators), which practically reduces to summarizing the addresses of all hosts within the domain, and advertising this information to neighboring domains and further beyond. The addresses used within individual domains are usually mutually adjacent, meaning that they fall into a particular range of the global addressing space, and thus they can be summarized with a so-called *IP address prefix*, which merely represents a unique combination of a number of initial bits which are the same for all addresses of the domain. This simple scheme of address summarization is crucial for minimizing the amount of information which is exchanged between individual IP domains globally, and thus makes the current inter-domain routing architecture practically feasible.

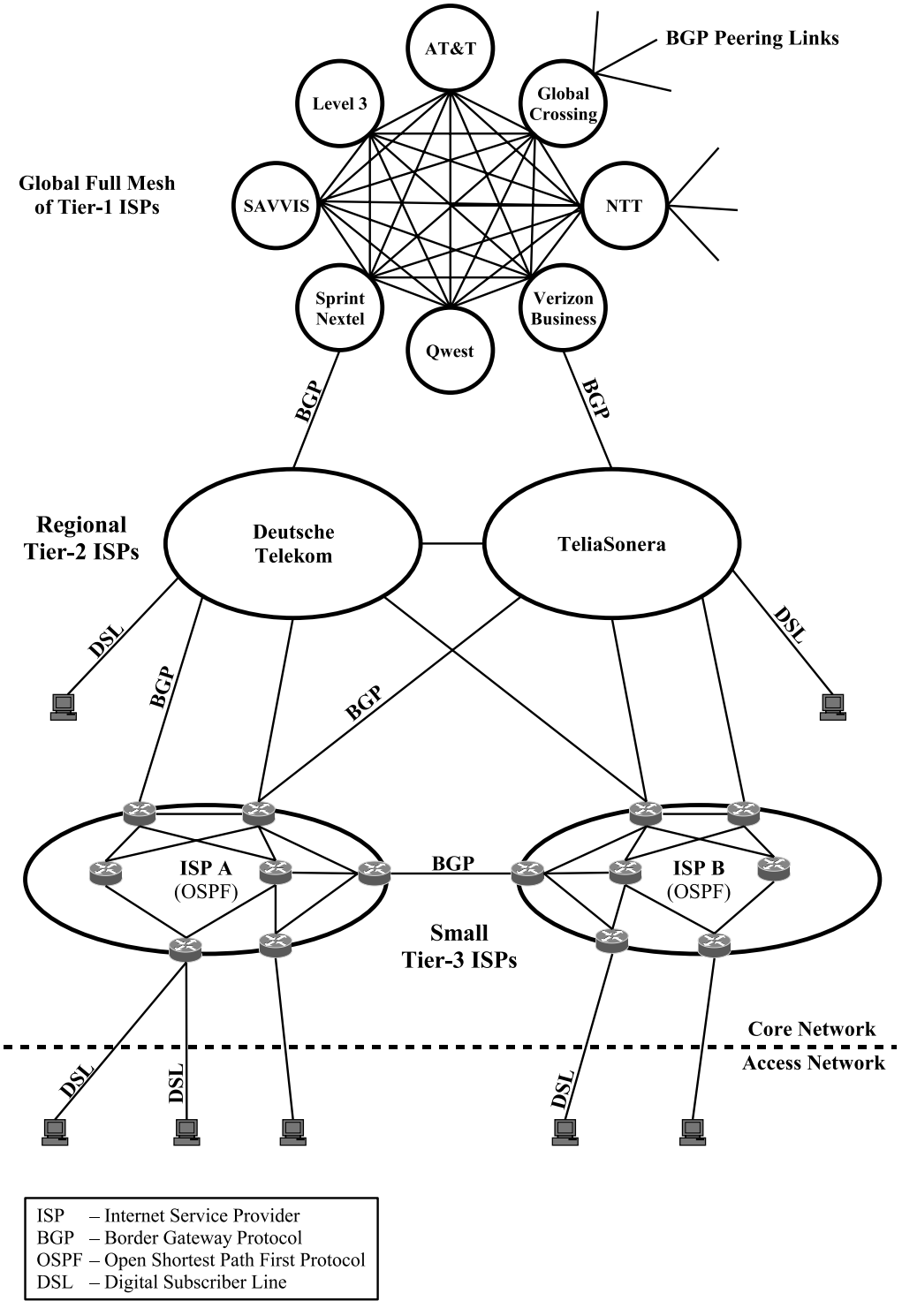



Figure 2.1.1: Current high-level architecture of the Internet.

### 2.1.1 Graph Theoretical Fundamentals

In the modern society, which heavily relies upon communications and transportation, networks of different kinds represent the fundamental backbone for almost all activities, like e.g. communication networks (telephone, Internet, etc.), electrical power grids, water, oil, and gas supply networks, transportation networks (air, rail, water, and road), etc. The principal functionality common to all these types of networks is that they are used for moving a resource (e.g. persons, goods, or information) from one point in the network to another. Therefore, very similar properties are desired for all of these networks: their transport (or transmission) capacity should be used efficiently, they should be reliable, meaning that they should display some degree of redundancy, and last but not least, they should be economically viable.

As much as different networks have been used throughout history, their theoretical investigation has largely lagged behind their practical application. This is rather surprising, as some ancient and most important problems had thus remained unsolved for decades [19]. A prominent example is routing resources along the shortest path from origin to destination – amazingly enough, before the *label setting* and *label correcting* algorithms had been developed in the 1950s and 1960s, which will be described in latter parts of this section, no efficient or much less optimal solutions had been available.

Graphs $G = (N, L)$ are basically simple abstract representations of networks of any kind, which are comprised of a set of nodes $N$ interconnected by a set of links $L$. Depending upon the type of networks and problems investigated, structures can be represented using many different types of graphs, and types of structure definition. In this section only a subset of the vast apparatus from graph theory is elaborated, focusing upon the *shortest path problem* which is fundamental to all routing and traffic engineering problems in communication networks.

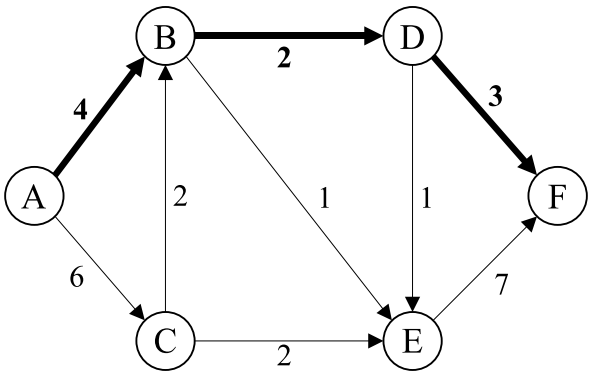



Figure 2.1.2: Shortest path A → F in a directed weighted graph.

In general, graph theory distinguishes between undirected and directed graphs. In *undirected graphs* links can be used in both directions, whereas in *directed graphs* links

can only be traversed in one direction (see Figure 2.1.2 for an example). Directed graphs are called acyclical if they do not contain any directed loops, whose traversal eventually leads back to the originating node. Furthermore, based upon the type of problem elaborated, the links will be associated scalar values with different semantics. In the case of the *shortest path problem* these values are called *weights* (or *costs*), and they serve as a basis for calculating the network wide routes. In Figure 2.1.2, an example for a shortest path between the nodes $A$ and $F$ is sketched, as the sum of link weights comprising the path is minimal.

Shortest path algorithms are usually classified into two major groups: *label setting* and *label correcting algorithms* [19]. Both approaches are iterative, but they differ with respect to the character of labels associated to the individual nodes in each step of the calculation process, which ultimately represent the distance from the source to the labeled node. In the former approach, in each step one node receives a *permanent label*, which represents its optimal (shortest) distance from the source node. In the latter approach, the labels are assigned only temporarily, and they keep changing at each iteration. When the algorithm finishes, the currently assigned labels become permanent, i.e. they represent the optimal (shortest) distance from the source node defined at the beginning of the calculation. Label setting algorithms are less general in that they apply only to directed acyclical graphs with nonnegative link weights, while label correcting algorithms apply to all classes of problems. However, on the other hand, label setting algorithms are much more efficient, i.e. they have much better worst-case complexity bounds [19].

The algorithm by Dijkstra certainly represents the most important contribution among the label setting shortest path algorithms [6]. The purpose of the algorithm is finding shortest paths from the source node $s$ to all other nodes in a graph with non-negative link weights. The main idea of the algorithm is to maintain a distance label $d(i)$ with each node $i$, which represents an upper bound on the shortest path length to node $i$. At the end of each iteration of the algorithm, the nodes are subdivided into two distinct groups: nodes with permanent labels and nodes with temporary labels, where permanently assigned labels represent optimal distances from the source node, and therefore denote nodes which have already been added into the final shortest path tree. The algorithm starts by defining the source node as the root of the shortest path tree, and labeling it permanently with a distance of zero. Before transferring a node to the group of permanently labeled ones during an iteration, the minimal distance of all temporarily labeled nodes is calculated as follows: each permanently labeled node scans all of its temporarily labeled neighbor nodes and calculates the sum of its distance from the source node and the link weight to the neighbor node; if the calculated value is less than the label temporarily assigned to the neighbor node, the distance label of the neighbor node is updated with the newly calculated value. This procedure is performed in all permanently labeled nodes which have temporarily labeled neighbors. In the final

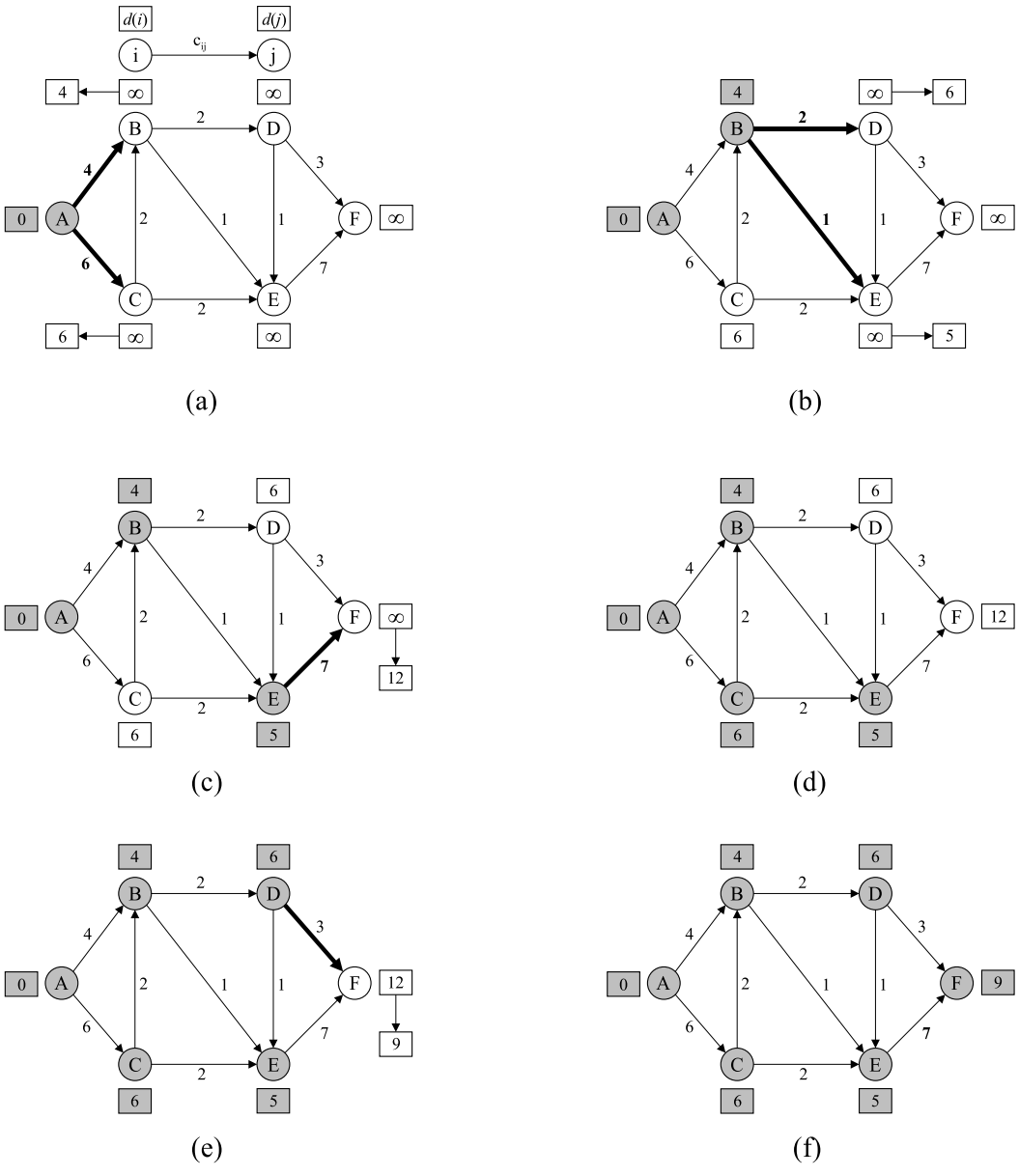



Figure 2.1.3: Illustration of Dijkstra's algorithm (extending Figure 4.7 from [19]).

step of each iteration, the temporarily labeled node with the least value is added to the shortest path tree, and its distance label becomes permanent. The algorithm terminates once all nodes are assigned permanent distance labels, and they are added to the final complete shortest path tree. Figure 2.1.3 illustrates a simple example of Dijkstra's algorithm. The example network consists of 6 nodes, $A$–$F$, which are initialized as unreachable ($\infty$ label in the corresponding boxes), except for the source node $A$ which is permanently labeled 0 (grey box). In each step of the algorithm, temporary labels of the adjacent nodes are adapted according to the weights of the connecting links, after which the node with the smallest label becomes permanent (grey circle). Eventually, the procedure terminates when all nodes have become permanently labeled.

The precise statement of the algorithm is presented below, with $S$ being the set of nodes with permanently assigned labels, $\overline{S}$ the set of nodes with temporarily assigned labels, $n$ the total number of nodes, $s$ the source node, $N$ the set of all nodes, $d(i)$ the label of node $i$, $pred(i)$ the predecessor of the node $i$ in the shortest path tree, $c_{ij}$ the weight for the link from $i$ to $j$, and $A(i)$ the set of node $i$'s neighbor nodes with temporarily assigned labels [19]:

```
DIJKSTRA'S ALGORITHM()
 1  S := {}; S̄ := N;
 2  d(i) := ∞ for each node i ∈ N;
 3  d(s) := 0 and pred(s) := 0;
 4  while | S |< n
 5  do
 6       let i ∈ S̄ be a node for which d(i) = min{ d(j) : j ∈ S̄ }
 7       S := S ∪ {i}; S̄ := S̄ \ {i};
 8       for each j ∈ A(i)
 9       do
10            if d(j) > d(i) + c_ij
11               then
12                    d(j) := d(i) + c_ij;
13                    pred(j) := i;
```

The complexity of Dijkstra's algorithm is allocated to two basic operations [19]:

- *Selection of nodes.* This part of the algorithm is performed $n$ times (where $n$ is the total number of nodes), and each iteration of the algorithm requires the scan of all temporarily labeled neighbor nodes. Therefore, the total node selection time is $n + (n - 1) + (n - 2) + ... + 1 = O(n^2)$
- *Distance updates.* This operation is performed $|A(i)|$ times for node $i$. Overall, this operation is performed $\sum_{i \in N} |A(i)| = m$ times. As each individual distance update has complexity $O(1)$, the overall distance update algorithm has complexity $O(m)$.

Based upon the analyzed complexities of the node selection and the distance update algorithms, the conclusion is reached that the overall complexity of Dijkstra's algorithm in its original statement is $O(n^2)$. In recent years, many versions of Dijkstra's algorithm have been suggested which significantly reduce the calculation complexity for non-fully meshed networks to $O(m+n\log n)$, making that version of Dijkstra's algorithm the prime choice for practical implementation of shortest path calculations in most communication networks [19].

From the group of *label correcting* algorithms, the *generic label correcting algorithm* will be considered next. At the core of its operation is also the concept of distance labels attributed to each node, which are reduced at each iteration of the algorithm by considering only local information, i.e. the distance labels of neighbor nodes and the length of the links to those nodes. The generic label correcting algorithm works as follows: a set of distance labels $d()$ is maintained at each step of the operation. In each step, the label of node $j$, $d(j)$ either has the value of $\infty$, indicating that a path from the source node to node $j$ has not yet been discovered, or it assumes the value of the sum of $c_{ij}$ and $d(i)$, if $d(j) > d(i) + c_{ij}$. The generic label-correcting algorithm is a procedure for successively updating the distance labels until they reflect the precise shortest path values, i.e. until no further changes to the distance labels can be made. A concrete example of the generic label correcting algorithm is given in Figure 2.1.4, where in each step the distance towards source node $A$ is updated for another node of the graph. Furthermore, the generic label correcting algorithm also maintains a predecessor index for each node $j$, $pred(j)$, which eventually enables tracing the shortest path back to the source node.

The formal description of the generic label correcting algorithm is given below:

```
GENERIC LABEL-CORRECTING ALGORITHM()
1  d(s) := 0 and pred(s) := 0;
2  d(j) := ∞ for each j ∈ N \ {s}
3  while  some link(i, j) satisfies d(j) > d(i) + c_ij
4  do
5       d(j) := d(i) + c_ij;
6       pred(j) := i;
```

Among the many different versions of label correcting algorithms [5, 20, 21, 22], for the purpose of routing in communication networks, distributed versions of the algorithm which find all shortest paths in directed graphs with nonnegative link lengths are of most interest, and accordingly they will be discussed in the next section. For the purpose of completeness, in this context it is worth noting that the generic label correcting algorithm has complexity $O(n^2C)$, where $C$ denotes the maximum link weight difference in the graph.

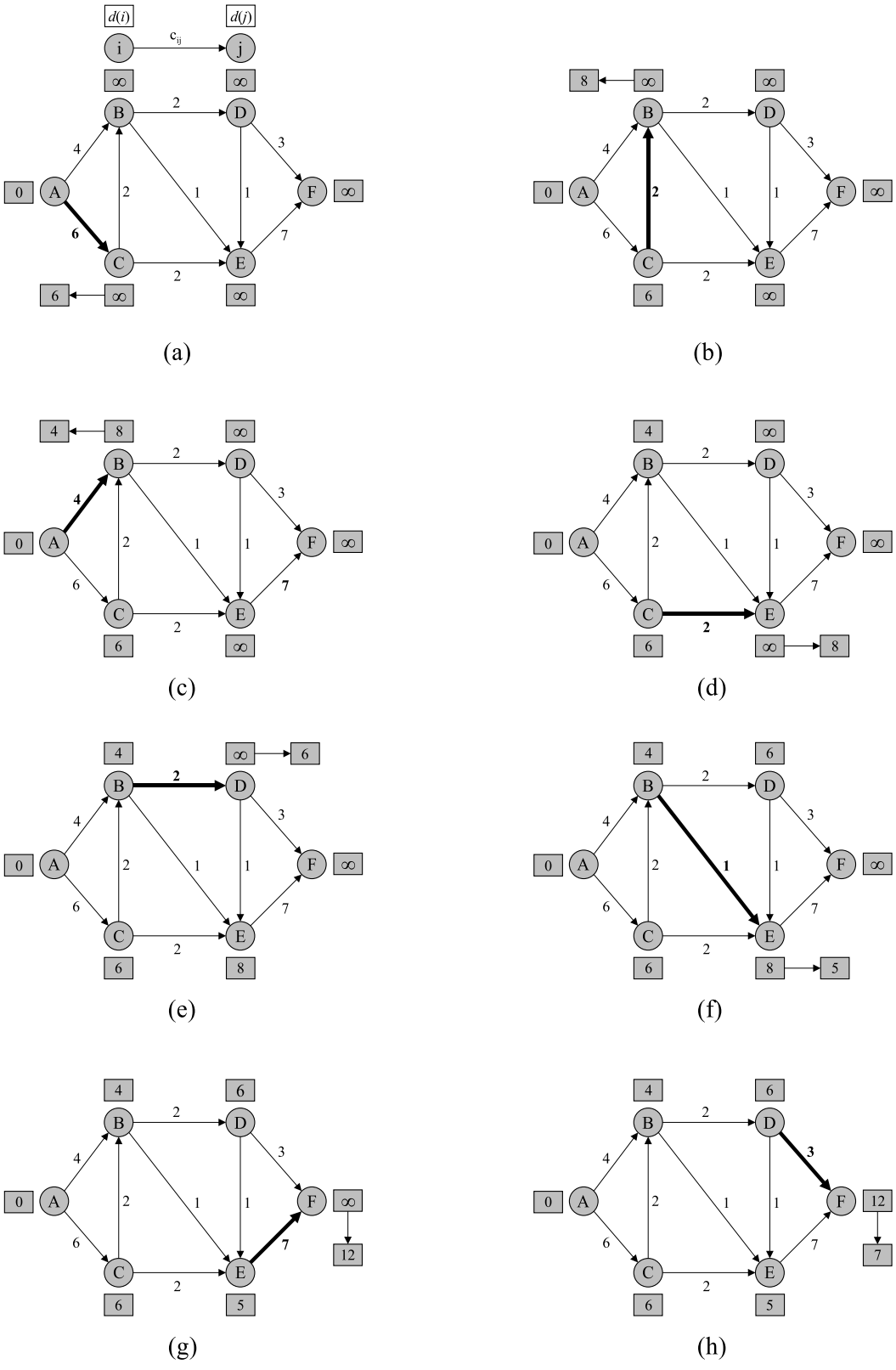



Figure 2.1.4: Illustration of the *generic label correcting algorithm.*

### 2.1.2 Link State and Distance Vector Routing Algorithms and Protocols

This section introduces distance vector and links state routing algorithms, which are fundamental to the current intra-domain routing protocols. Both of these algorithms represent concrete adaptations of the algorithms from graph theory which have been introduced in the previous section.

In the current intra-domain routing architecture, routers are interconnected by links which are assigned non-negative weights. Furthermore, the packetized traffic payload is routed in a connectionless manner using the hop-by-hop routing strategy, meaning that each router individually decides where to forward packets such that they reach their destinations as declared in the packet headers. In order to ensure efficient, loop free routing in such a distributed architecture, the routers are obliged to follow a simple common routing rule: packets should always be routed via the shortest paths, where path length represents the sum of the link weights of all links included.

In order to determine the shortest paths in large network structures, fully automated routing protocols are required for several reasons: firstly, the task of setting up the routing tables is manually often hardly feasible in networks whose size exceeds several nodes, and secondly, due to reconfiguration and outages network topologies may change, requiring fast adaptation of routing to the new situation. Therefore, the development of reliable routing protocols represented the top priority in the early Internet, and resulted in the emergence of two major groups of protocols: *link state* and *distance vector*.

The main idea of *link state* protocols is that all nodes in the network announce the state of their unidirectional links towards their neighbors to the entire network, and that each node in the network calculates a shortest path tree with itself as the source, using Dijkstra's algorithm based upon the knowledge of the entire global topology as announced by the other nodes. After having calculated its shortest path tree, each node is able to match individual destinations in the network to particular next hops, i.e. to establish a routing table. The mechanism used for announcing link state information is flooding: each node forwards received (or generated) link state information unchanged to all of its neighbor nodes. Although this simple mechanism is prone to generating a substantial amount of signaling overhead (as will be discussed in later parts of this thesis), it is very efficient in guaranteeing that all signaling messages eventually reach all nodes in a connected network.

Modern day implementations of link state algorithms, like e.g. Open Shortest Path First (OSPF) [7, 8, 23] or Intermediate System to Intermediate System (IS–IS) [9] fulfill the task of offering robust network connectivity and loop-free routing in a very efficient manner, and thus they are currently the most widespread intra-domain protocols in the Internet. In order to enable efficient operation of the algorithm in very large network structures of several thousands of nodes, link state algorithms additionally offer the

possibility of subdividing the network hierarchically, which opens the potential of reducing the number of IP destination address prefixes which have to be announced by the protocols, and even more importantly the possibility of reducing the signaling overhead generated by the flooding mechanism, because information from lower hierarchy levels of the network is normally not announced throughout the entire network domain [3].

Based upon the work of R. E. Bellman, L. R. Ford and D. R. Fulkerson in the area of label correcting shortest path algorithms, *distance vector* routing protocols follow a completely different paradigm for establishing consistent network wide routing tables. Instead of employing global dissemination of topology information which would subsequently be used for calculating the routes in a distributed manner, distance vector algorithms rely upon the exchange of reachability information between the neighbor nodes. In each iteration of the algorithm, the nodes convey so called *distance vectors* which contain their current shortest distance labels to *all* destinations. In the first iteration, only the neighbor nodes are marked as reachable, at the cost of the weight of the directly connecting link. Subsequently, as nodes keep exchanging their distance vectors (which practically corresponds to exchanging entire routing tables in each iteration) they update their temporary labels for individual destinations if the sum of the label announced by the neighbor and the weight of the connecting link is lower than the current local label. Eventually, the network will converge (i.e. the temporary labels will become permanent) at the moment when all network wide labels remain unchanged during an iteration. This particular version of label correcting shortest path algorithms applied in distance vector protocols is often referred to as the *Distributed Bellman-Ford Algorithm* (cf. [3]).

At first glance distance vector protocols appear much more simple and straightforward than link state protocols as route calculation is reduced to a simple exchange of routing tables, whereas links state protocols necessitate the existence of multiple data structures for topology information exchange, route calculation, and the routing tables. Additionally, link state protocols incorporate two independent complex algorithms for the dissemination of topology information and route calculation. While this initial impression indeed holds as far as implementation and computational complexity of the algorithm is concerned, the simplicity of the distance vector protocols comes at a price when the convergence properties of the algorithm are considered: firstly, after a network running distance vector protocols is initialized, up to $d$ iterations of the algorithm are needed in order for the network wide routing tables to converge, where $d$ is the network diameter, thus introducing a considerable penalty for large networks compared to link state protocols. Moreover, the most significant drawback of distance vector protocols is their very slow convergence in the presence of topological changes, which can lead to the formation of transient routing loops. Although many add-ons and improvements have been introduced to distance vector routing algorithms, due to their unfavorable convergence properties these protocols are today mostly found only in legacy networks,

whereas link state protocols like IS-IS and OSPF represent the mainstream in contemporary intra-domain IP routing.

However, a variant of distance vector protocols, the *path vector protocol*, is used in the Border Gateway Protocol (BGP-4) for the exchange of global inter-domain routing information [24]. The main difference between distance vector and path vector protocols is that instead of announcing only single scalar labels for particular destinations, path vector protocols announce the entire path towards each destination, which helps to alleviate certain drawbacks of distance vector protocols, although routing stability issues in the presence of topological changes still represent a major concern. The main reason why link state routing protocols are not used for inter-domain routing lies simply in their limited scalability due to their resource intensive signaling, advancing path vector protocols to become the only viable choice in this area of application.

## 2.2 State of the Art in IP Traffic Engineering

### 2.2.1 Approaches Based upon Link Weight Optimization

The routing of traffic within IP networks on top of the proven architecture which employs shortest path routing based upon link weights today represents the state of the art as far as concrete implementations in major Internet Service Provider (ISP) networks are concerned. Due to the mature link state protocols deployed, these networks display very high robustness, as the traffic is swiftly and reliably distributed onto alternative routes in the presence of equipment failures, thus alleviating the main concern of any network provider [25, 26, 27, 28, 29].

However, apart from these beneficial properties, link state routing algorithms also hold the problem of appropriate network configuration, which still has not been generally resolved by network equipment vendors. The main issue concerning the configuration of link state protocols is the assignment of link weights, as they have exclusive influence upon the network-wide setup of routes. But unfortunately, deriving a meaningful set of link weights for a given network proves to be very difficult, as the value imposed upon an individual link always has the potential of affecting many different paths throughout the network. In other words, in architectures based upon the concept of weighted directed links it is generally impossible to have explicit influence upon the paths of individual connections, and therefore it is hardly possible to find an intuitive approach to the traffic engineering problem.

Unfortunately, major IP routing equipment manufacturers like Cisco Systems [www-2] and others have also merely come up with several *best practice* heuristics for setting the link weights, like e.g. setting all link weights in the backbone network to the same value, which assures that traffic is always routed along paths with the minimum number of hops, and thus leads to a minimization of the average link load in the network. However,

this *min-hop* routing strategy does not hold the potential of distributing the traffic away from links prone to congestion, but rather treats all links equally, irrespective of their characteristics like e.g. transmission capacity. In order to take the link characteristics into consideration and thus statistically try to avoid excessive usage of low bandwidth links, the most widely applied Cisco heuristic recommends setting link weights inversely proportionally to the link capacities. Another purposeful heuristic method is to set the link weights proportionally to the link propagation delay, which in the absence of congestion leads to a minimization of the delays experienced by the flows throughout the network.

Nevertheless, the mentioned approaches all remain traffic agnostic, i.e. they do not adapt the routing to the current traffic conditions in the networks. In many cases, this can lead to congestion in networks which hold the potential of accommodating even a multiple of the present volume of traffic [30, 31, 32]. Therefore, the current state of the art in link state routing leaves a lot of room for improvement using traffic engineering methods, which aim at improving the performance of the network by optimizing the utilization of its resources.

In this section, traffic engineering methods are presented which do not require the introduction of any new switching or routing technologies, but which instead focus upon the configuration of the current IP networks running link state protocols. The fundamental idea in this context is to optimize the network wide link weights according to a particular objective function. In [30, 31, 32], the authors investigate the AT&T backbone network with traffic demands projected from previously conducted measurements. As the authors demonstrate that optimizing link weight settings for a set of traffic demands is NP-hard, they choose to apply an optimization algorithm based upon a local search heuristic. In order to achieve a meaningful distribution of load, special attention is paid to choosing an appropriate objective function: if minimizing the maximum link utilization is defined as the sole objective, the algorithm may produce results which contain very long paths for particular connections, thus substantially raising the average link utilization in the network. On the other hand, if the focus is set too strictly upon the average link utilization, overloaded links may be disregarded in the optimization, leading to the formation of congestion hotspots in the network. The main idea of the chosen objective function is that it is cheap to send a flow over a link with small utilization, whereas this becomes more expensive if the utilization approaches 100%, and assumes disproportionally high values if the utilization exceeds 110%. In its concrete realization, the function is piecewise linear increasing and convex, and thus maximally adapted to the requirements of optimization procedures (see Figure 2.2.1).

Contrary to the previously widespread beliefs that link state routing algorithms cannot be configured such that they spread load evenly across the network, Fortz and Thorup show that in the case of the AT&T backbone network their heuristic generates weight settings which make link state protocols perform within a few percent from

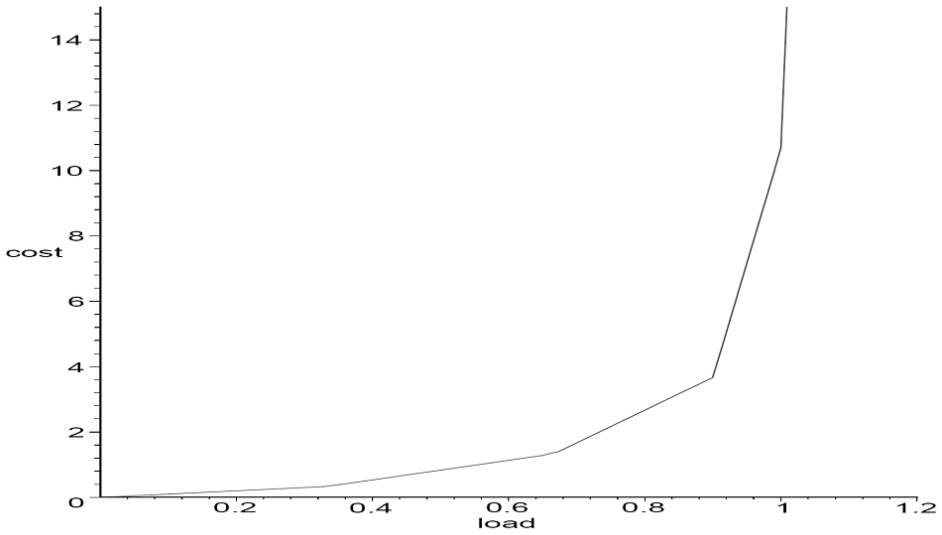



Figure 2.2.1: Link cost $\phi_a(l(a))$ as a function of load $l(a)$ on link $a$ for link capacity $c(a) = 1$. (Source: [30])

*optimal general routing* which does not impose any limitations upon the path(s) flows may take between individual pairs of hosts [30, 31, 32].

Additionally, the traffic engineering potential of optimized OSPF weights is investigated on artificial network topologies based upon the results from [33, 34]. The results of these experiments show that the traffic engineering potential does not quite reach the excellent results of the AT&T network example, but nevertheless does enable the accommodation of 50-110% higher traffic demands than the standard heuristics for setting link weights either inversely proportionally to the corresponding capacities, or proportionally to the link propagation delays.

In spite of its impressive efficiency, link weight optimization has not yet seen large scale deployment due to several limiting factors. Firstly, this approach requires timely and accurate information about the state of the network, i.e. a list of the operational routers and links, the link capacities, and last but not least the traffic demand matrix for the entire network. As far as the configuration of the network is concerned, this can be obtained either from the routers' configuration files, or polled via the Simple Network Management Protocol (SNMP) [35], or it can be obtained from dedicated software for the monitoring of routing protocols, like e.g. the *Python Routeing Toolkit* [www-3].

As deriving traffic matrices for operational networks represents a complex challenge in the current technological environment [36, 37, 38], special attention must be devoted to this particular task. In [32], four approaches as laid out in [39] are proposed for the purpose of link weight optimization: (1) obtaining the necessary traffic statistics directly from the SNMP Management Information Bases (MIBs) [40], (2) computing the traffic matrices based upon measurements at the edge of the network, (3) inferring the traffic matrix from measurements on core network links by correlating them with routing data,

and (4) by online traffic sampling as with *Cisco NetFlow* [www-4]. Additionally, current traffic measurements show that traffic matrices display typical time-of-day behavior [41], meaning that the optimization should certainly take this fact into account, either by calculating multiple sets of link weights for different times of day, or by performing the optimizations such that the introduced link weights are applicable throughout the 24-hour period (see Figure 2.2.2).

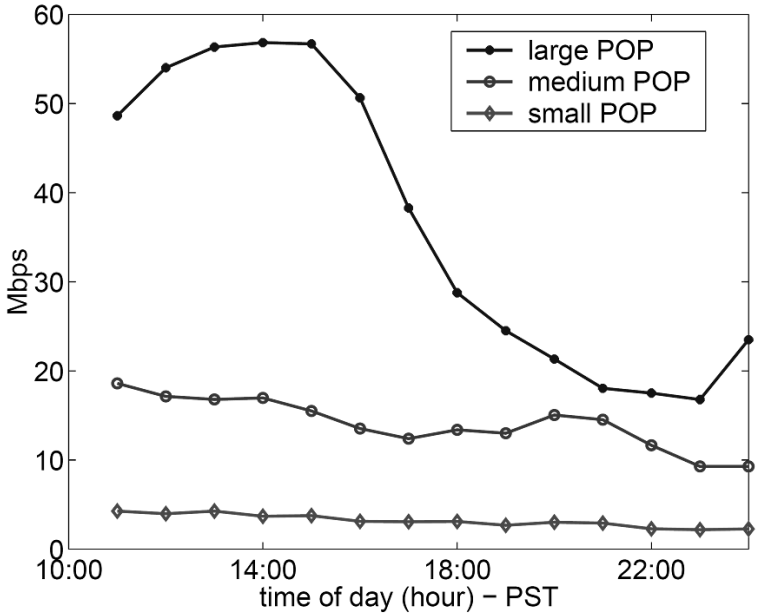



Figure 2.2.2: Time of day effects in Sprint Points-of-Presence (POPs) in three example U.S. cities of different size. (Source: [41])

As a further deployment obstacle, the applicability of link weight optimization is somewhat reduced in networks with frequent equipment outages or reconfiguration and upgrade activities. The main reason is not related to keeping topology information up to date (which is quite easily accomplished with link state routing protocols), but is rather attributed to the fact that a given set of optimized link weights only applies well to a given {topology, traffic matrix} pair, and that it does not have to produce satisfying results in the case of topology changes. This drawback of link weight optimization becomes even more aggravated by recent measurements and analyzes performed by a tier-1 ISP, which demonstrate that equipment outages are regularly and frequently occurring events in large network structures, and that they should rather be regarded as integral parts of network operations than as exceptional events [25, 26, 27].

In order to improve the performance of link weight optimization in topologies with frequent outages, [32] proposes the application of an optimization scheme which would reoptimize the routing in the presence of equipment failures by changing only one or at most a few link weights in the whole network. In contrast, [28] proposes to perform link weight optimization such that the network performance is optimized for all cases of single link failures. The main rationale behind this approach is based upon the findings

in [25], which state that more than 80% of failures last less than 10 minutes, such that performing adjustments to link weights hardly makes sense for shorter periods of time. The chosen optimization approach yields quite promising results, as overload in the case of single failures can be reduced up to 40% compared to the standard optimization approach, with the cost of only a slight 10% performance degradation during regular network operation.

Last, but not least, in order to perform link weight optimization in a large operational network, a comprehensive network management framework similar to the one(s) proposed in [42, 43] is needed, as the tasks of continuous network measurements and topology monitoring, running of the optimization algorithm, and introducing the calculated link weights to the routers is hardly manually feasible, or at least very inefficient. Furthermore, as individual link weight changes may affect an enormous number of paths (especially if link weights are decreased), the effect of the introduced changes should ideally be verifiable in advance, which makes a strong case for the integration of the management framework with a network simulation environment.

### 2.2.2 IP Traffic Engineering in Path/Circuit Switching Networks

Apart from the plain IP routing architecture as described in the previous sections, additional IP-compatible network architectures have been investigated in recent years. The most prominent example among such proposals is Multi-Protocol Label Switching (MPLS) which has been introduced in [11, 44]. The basic idea of MPLS is to completely separate route control and packet forwarding in the network, while still remaining backward compatible with existing IP infrastructures. In order to achieve this, MPLS introduces an additional header to IP packets, which contains all the information necessary for making forwarding decisions. Accordingly, in MPLS-enabled IP routers traffic is not any more necessarily forwarded according to the shortest path rule, based upon the link weight settings and the destination IP address. Instead, IP packets are attributed to a so called Forwarding Equivalency Class (FEC) at the moment they enter the MPLS domain, and based upon this classification they are assigned to a corresponding Label Switched Path (LSP) and forwarded from ingress to egress, such that the network operator has got full control over packet classification and path establishment using the label switching paradigm, as described in [11]. Since normal IP routing, which usually runs in parallel to MPLS, does not have any influence upon MPLS packets, the LSPs are practically hidden from these mechanisms, and due to this stealth property they are often referred to as *tunnels*. Furthermore, as the semantics for establishing the FECs are completely arbitrary, and not necessarily related to traditional routing architectures (although this still represents a possibility), FECs are usually referred to as *switched circuits*. In the early days of MPLS, the forwarding mechanism which makes

routing decisions based upon the short labels instead of the larger IP destination prefix had been hailed as a significant simplification, opening the path to routing for very high link speeds, and having the potential of significantly reducing the cost of switching equipment. However, recent significant advances in routing table search have practically completely alleviated this advantage, and made native IP routing equally applicable for networking scenarios with very high link capacities [45, 46].

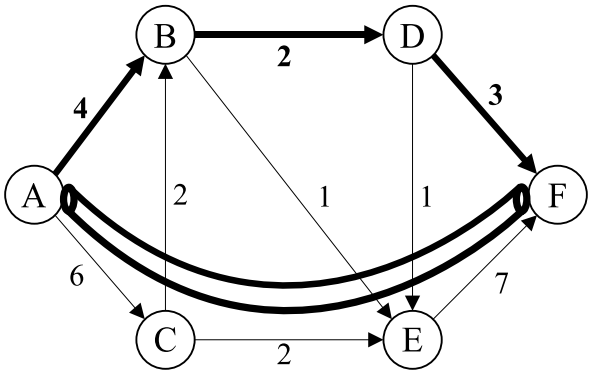



Figure 2.2.3: Arbitrary MPLS tunnel from node $A$ to node $F$, $A \rightarrow C \rightarrow E \rightarrow F$ which disregards the shortest path according to the link weights, $A \rightarrow B \rightarrow D \rightarrow F$.

The flexibility in the choice of paths advances MPLS to become one of the most popular technologies for implementing traffic engineering solutions. Contrary to the traditional shortest path architecture described in the previous section, MPLS does offer very straightforward explicit control over the paths of individual flows (see Figure 2.2.3). Accordingly, in a network of $n$ nodes, a total of $n \cdot (n-1)$ independent single paths may be established, allowing for a dedicated *switched circuit* for each pair of nodes. However, although this may seem very attractive at first, the abundance of possible combinations automatically advances the configuration of such networks to a respectable optimization issue, yielding a wide variety of proposed solutions.

In [47], a set of novel techniques for traffic engineering in Quality of Service (QoS) supported data network is proposed, which is based upon solving techniques for *multi-commodity flow problems* [48, 49]. These techniques address the design of the topology and the size of LSPs by performing network wide optimization subject to constraints on routing imposed by end-to-end QoS requirements and other considerations. Although the proposed MPLS provisioning architecture represents a highly scalable solution capable of managing hundreds or even thousands of nodes, this approach also incorporates several inherent drawbacks, which are practically identical to those of link weight optimization techniques: firstly, the knowledge of the network wide traffic matrix is required as an input to the optimization algorithm, and secondly, due to the time variant nature of the traffic demand matrix, periodic re-optimization of the network configuration is required, holding the potential of introducing significant disturbances to the routing of

individual flows.

In [50], a software system called Routing and Traffic Engineering Server (RATES) is proposed for MPLS traffic engineering. The server implementation consists of a policy and a flow database, an interface for traffic engineering policy definition, and a Common Open Policy Service (COPS) [51] based implementation for communicating path and resource information to the routers. The main idea of RATES is to set up bandwidth guaranteed LSPs between ingress-egress node pairs, employing path selection based upon the *Minimum Interference Routing Algorithm* (MIRA) as proposed in [52, 53]. MIRA aims at reducing the *interference* of newly arrived routing requests to potential unknown future requests, with *interference* defined as the phenomenon of reducing the maximum available bandwidth between other ingress-egress pairs. From a practical perspective, the most important difference to the previously described technique from [47] is that no global traffic matrix is required. Instead, an on-line solution is proposed which handles arriving traffic requests in a one-by-one manner, largely obsoleting heavy traffic measurement requirements.

A further online algorithm for the dynamic routing of bandwidth guaranteed LSPs is proposed in [54]. In the presented model, LSP setup requests denoting the node pair and the required bandwidth arrive dynamically analogously to the model from [52, 53]. Although inspired by the before mentioned MIRA algorithm, the proposed solution claims to circumvent several of its drawbacks by additionally considering the links' residual bandwidth and path hop counts.

In [55], a scalable distributed traffic engineering system is presented which is based upon a regular exchange of link load information between the traffic engineering units. This reactive system is realized with two load re-allocation strategies, which either employ the rerouting of individual LSPs, or perform load balancing within a multi-path structure. The results of the approach show its significant potential compared to shortest path routing, with only minimal overhead added to the network's operational complexity.

The practical applicability of the mentioned traffic engineering solutions based upon MPLS depends primarily upon the robustness to network equipment failures. Although a multitude of mechanisms has been proposed for the resilience of MPLS networks [56, 57, 58], the compatibility of the described traffic engineering solutions to these mechanisms must be carefully checked. Furthermore, even if the chosen combination of approaches is compatible at the level of protocol correctness, it remains to be verified whether the traffic engineering algorithms perform well in the presence of failures, or if additional refinements to the mechanisms or well defined deployment strategies are required, as e.g. in the case of link weight optimization.

### 2.2.3 Traffic Engineering Approaches Based Upon Self-Adaptive Routing

In contrast to traffic engineering approaches based upon link weight optimization or the optimization of MPLS LSPs, self-adaptive routing algorithms try to keep all route adjustments within the framework of the routing protocols themselves. In this way, the network can adapt to the current traffic conditions completely autonomously, i.e. without the need of manual interventions.

The first examples of self-adaptive routing in packet switched networks date back to the deployment of the ARPANET [1]. The first algorithm deployed in the ARPANET was based upon the distributed version of the Bellman-Ford shortest path algorithm, as described in Section 2.1.2 and in [59]. Each node maintained a table of shortest distances and next hops to each destination, and it exchanged its routing tables with its neighbors every 2/3 seconds [60]. The only element which assured the adaptation of the routes was the link weight metric, which was simply set proportionally to the instantaneous queueing delay, and which therefore dynamically reflected the changes to the queue length. The main objectives of this routing scheme were to route traffic along the minimum delay paths in the network, and at the same time to avoid the formation of congestion hotspots, which would automatically be reflected by a surge in the experienced delay. However, the chosen metric proved to be a very poor indicator of the current delay, leading to network wide routing oscillations. Additionally, the oscillating behavior was further aggravated by the formation of routing loops, which were a direct consequence of distance vector routing protocols' inability to converge efficiently in environments with unstable link weight metrics or topologies [60].

In order to improve the stability and the performance of ARPANET routing, several years later a novel routing algorithm was introduced [60, 61]. The most important change compared to the initial algorithm was the switch to link state routing, as well as the introduction of a more stable link weight metric, which was based upon a measured 10-second average of the link delay. While this scheme proved to perform well under the conditions of low and medium load, it was also prone to oscillations during periods of very high load. The main reason for this behavior lies in the too coarse granularity of load shifting, as the change of a single link weight may affect the paths of many different flows. In other words, in the case of high load, this scheme often leads to a sharp increase in the metric of the most affected link, which induces a shift of a large number of flows to alternative paths. However, it is easily possible that this shift will immediately produce an overload on the alternative paths, leading to an increase of the corresponding link metrics. In the next step, this will lead to a shift of traffic away from the alternative paths, very likely back to the original path, as its link load metric will most likely have subsided in the meantime, etc. Figure 2.2.4 presents an illustrative example of this oscillating behavior in the well-known *fish topology*, in which the allocation of the traffic

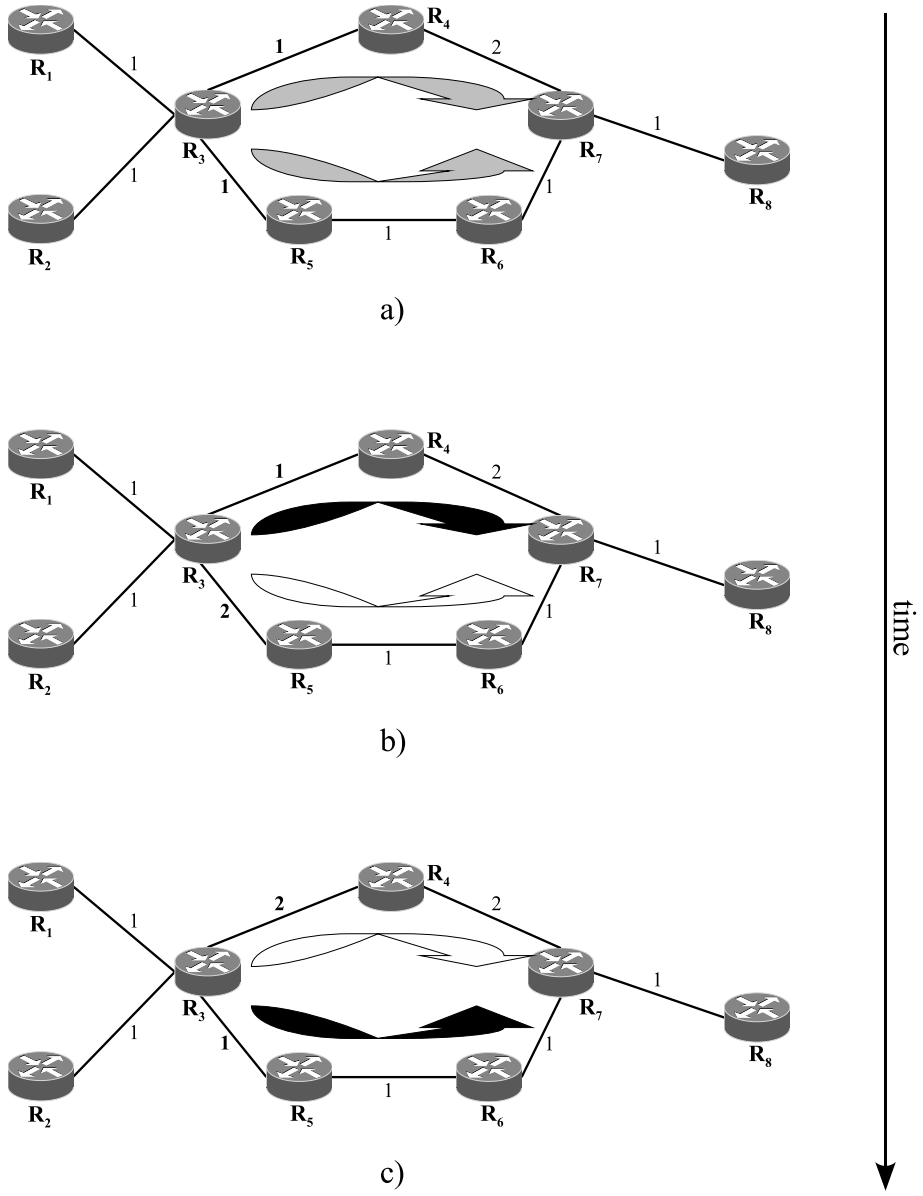



Figure 2.2.4: Load oscillations observed on two paths connecting nodes $R_3$ and $R_7$.

between the nodes $R_3$ and $R_7$ changes due to variations in the link weights $R_3 \rightarrow R_4$ and $R_3 \rightarrow R_5$, always making use of the available shortest path(s).

Unfortunately, with this routing strategy there are no mechanisms for dampening the described routing oscillations, but instead they will persist during the entire period of high traffic load, leading to several highly unfavorable effects throughout the network: firstly, the persistent updates of the link weights will induce significant overhead in terms of signaling load in the network and the consumption of processing power in the routers. Secondly, instead of leading to the desired objective of an even distribution of traffic throughout the network, oscillations will have the opposite effect by causing flip-flop like overload in parts of the network, while other parts are hardly loaded at all. In order to counteract this behavior, exhaustive modifications of the ARPANET routing metric

are proposed in [60], which retain the beneficial properties of the previous algorithm in the case of low or medium load, but enable diverting only a portion of flows from highly congested links by refining the granularity of changes to the link weights metrics, allowing the remaining paths to make efficient use of the links.

Although the approach of dynamic link weight changes can be tailored to fit particular types of network topologies and traffic characteristics, due its very coarse granularity of load shifting it is hardly recommendable as a generic solution for Internet traffic engineering. In terms of practical modern day implementations of this scheme it is important to mention the Interior Gateway Routing Protocol (IGRP) and the Enhanced Interior Gateway Routing Protocol (EIGRP) by Cisco Systems, which do enable the inclusion of link load information into the link weight metrics [3, 62]. However, similarly to the phenomena observed in ARPANET routing, [63] demonstrates that IGRP routing will always oscillate between two worst case traffic distributions, unless the link weight metric is chosen such that its traffic-insensitive component has a high relative weight compared to its traffic-sensitive component. In order to assure maximum network stability, dynamic link weight adaptation is turned off in standard Cisco router configurations, such that it needs to be explicitly enabled by the network operator if it is to be used [3].

Originating from a somewhat different school of thought, the two traffic engineering approaches described next base their routing decisions upon local information about link overload, and upon local reactions to this overload by deflecting traffic upon less loaded links. Interestingly, although they are in many ways similar, the algorithms are rooted in two completely distinct networking technologies, i.e. in circuit switched telephony and in IP networks, and thus they offer a good opportunity for detecting possible analogies in key traffic engineering principles.

The Dynamic Alternative Routing algorithm (DAR) is a control scheme for fully meshed circuit switched telephone networks which has been proposed in [64, 65], and subsequently applied in the British Telecom trunk network [66]. The fundamental idea of DAR is as follows: as long as there is enough residual capacity, a call is always routed via the direct link to the other digital switch. However, if the direct link is congested, the call is routed via a predefined alternative node, i.e. it is routed to the destination via an intermediate hop. If this works well, the alternative node is kept in memory and serves as a default alternative node also in the future. If the alternative path is congested as well, then another alternative path is chosen among the $(n-2)$ remaining nodes in the network, and serves as the new default alternative. In order to ensure optimal usage of the network wide installed capacity, it is important to make sure that the connections are whenever possible routed along the direct path. More specifically, this translates to the requirement that a diverted connection should avoid to force another direct connection to take an alternative path [67, 68]. The concrete implementation of this imperative is performed by means of *trunk reservation*, meaning that a minimum number of residual

channels should always be reserved for the direct connections [69].

In [70], a similar approach is applied in IP networks, i.e. the *deflection routing scheme*. In the case of link overload, which mostly occurs during periods of link failures, the routers individually make decisions to reroute traffic for particular destination prefixes to alternative paths. In order to perform this task within the current architecture of intra-domain IP routing, having at least two viable next hops for each destination prefix is a strong prerequisite for this approach. However, as multiple paths towards a destination are only allowed if they have the same minimal distance metric, many practical topologies will not meet this hard requirement. Therefore, a relaxation of the shortest path criterion called *strictly decreasing cost criterion* is proposed, which states that each neighbor node may be used as a viable next hop in order to reach a particular destination if it is strictly closer to the destination in terms of the distance metric than the current node itself. Although this criterion in many practical cases allows for a much larger number of multiple paths [15], it still reliably prevents the formation of routing loops, as the distance to the destination is strictly decreasing along each hop of the packet's path to the destination.

## 2.3 Traffic Engineering Approaches in a Multi-Path Routing Context

In contrast to the approaches described so far, this section will cover adaptive routing approaches which are more closely related to the proposed Adaptive Multi-Path routing (AMP).

### 2.3.1 The Optimized Multi-Path Algorithm

Starting with the state of the art in intra-domain routing, the main goal of the Optimized Multi-Path (OMP) approach is to extend the current routing protocols with traffic engineering capabilities while remaining completely compatible with the current intra-domain routing architecture [12, 71]. In order to achieve this, OMP builds upon the favorable properties of link state routing protocols like Open Shortest Path First (OSPF), and their capability of employing multiple shortest paths between two nodes if they exist, called Equal Cost Multi-Path routing (ECMP), which enables static and uniform distribution of traffic upon all available paths towards individual destinations [8]. OMP mainly introduces changes with respect to OSPF's traffic agnostic nature of routing: Avoiding the mistakes of the previous proposals, which attempted to perform load balancing by acting upon the link weight settings, a novel approach has been chosen which enables very high granularity of load shifting, thus paving the way towards stable operation in realistic network environments.

Designed as a traffic engineering extension to OSPF, OMP uses global information about the network wide link loads in order to route traffic such that congestion hotspots are alleviated by means of traffic redistribution. Whereas the dissemination of link load information is indeed performed throughout the network using the link state protocols' flooding mechanism, the load balancing actions are performed strictly locally in individual nodes, without resorting to any kind of global coordination. In the rest of this section, OMP's main building blocks in terms of multi-path criteria, link load metrics, signaling, packet forwarding, and load balancing will be explained, followed by two simulation studies evaluating the algorithm's performance.

In order for the ECMP strategy to yield a high number of paths, link weight settings need to be carefully tuned, which is often not consistently possible in large network structures. Therefore, OMP employs the *strictly decreasing cost criterion* which has already been mentioned in the context of the deflection routing scheme in Section 2.2.3. Using this criterion, which is denoted as the *relaxed best path criterion* in the context of OMP, the number of multiple paths between most {source, destination} pairs in the network can be significantly increased compared to ECMP, as the next hops do not necessarily have to be on the shortest path routes [15]. For reacting meaningfully to network wide link load information, OMP must have a mechanism for determining the entire paths to the destination with a list of all links contained therein. Only in this way the algorithm can determine which local load balancing actions to perform in order to reduce (or increase) the load on a distant link. For this purpose, OMP defines the so called *next hop structure*, which is essentially a *per destination* data structure containing the list of links comprising the different paths towards a destination. Based upon this information, OMP's load balancing algorithm decides on a *per destination* basis about the relative traffic share each path from a next hop structure should receive.

OMP packet forwarding is based upon the 16-bit Cyclic Redundancy Check scheme (CRC-16) known from many ECMP implementations, which has the favorable property of spreading realistic IP traffic very evenly across the hash space, making this scheme the prime choice for Internet load balancing for a wide area of applications [72]. The main difference compared to ECMP is that traffic is generally routed asymmetrically, reflecting the computed shares of traffic for the individual paths in the next hop structure. For this purpose, the shares allocated to the individual paths for a specific destination are mapped to the corresponding next hop nodes, thus defining the relative sizes of the hash space attributed to each next hop link. Accordingly, the packet forwarding procedure in OMP nodes works as follows: after a packet is received, its destination address is checked, after which the viable next hop links for reaching that destination are determined. If there is only one next hop, the packet is immediately forwarded via the corresponding interface. Otherwise, if multiple next hops are viable, the CRC-16 hash value is computed based upon the source and destination IP addresses of the packet, after which the next hop node holding the CRC-16 hash space containing that value is

determined. Finally, the packet is forwarded via the corresponding link.

Concerning the link load information, OMP does not only resort to plain link utilization, $\rho$, as a ratio of transmission rate and the link capacity. Instead, information about packet losses is included in the calculation in order to be able to compare multiple links which display very high link utilization, a situation which may easily occur in networks which predominantly carry elastic traffic employing the Transmission Control Protocol (TCP) [73]. The chosen link load metric is based upon the findings about TCP dynamics from [74], which state that TCP connections on congested links slow down roughly in proportion to the square root of loss, and that TCP practically does not display any slowdown at packet loss probabilities below 1%. Accordingly, the metric of *equivalent load*, $\rho'$, is established using the following formula:

$$\rho' = \rho \cdot K \cdot \sqrt{P}\,, \tag{2.3.1}$$

where K is a scaling factor defining the packet loss probability at which the equivalent load metric starts exceeding simple link utilization, and $P$ the packet loss probability observed on the link. The recommended default setting for the factor $K$ is 10, thus defining 1% as the boundary packet loss probability at which TCP starts slowing down.

As far as the dissemination of equivalent load information is concerned, a novel data structure defined in [75] is used, which enables supplementary information to be announced via the Link State Advertisement (LSA) flooding mechanism of OSPF. In order to ensure sufficiently fast updates of link load information, and yet to avoid any unnecessary flooding of link load LSAs, the decision whether to flood is based upon three parameters: the absolute load on the link, the relative load change to the last flooded information for a particular link, and finally the time elapsed since the last load change was considered. Accordingly, the likelyhood of issuing an LSA update increases with the difference in load since the last announcement, as well as with the time elapsed since the last LSA was issued.

Based upon the received link load information, the load adjustment algorithm (which comprises the core of OMP) identifies the highest (i.e. in OMP terms *critically*) loaded link in the next hop structure for each destination, and adjusts the relative load shares of traffic for the paths in the next hop structure such that the shares of those paths which do not contain the critically loaded link are increased, while the shares of those paths containing the critically loaded link are reduced. As already mentioned, these shares are defined as portions of the CRC-16 hash space, thus allowing for very fine-granular load shifts, as the traffic for each destination may be split at the granularity of $2^{16} = 65{,}536$. Once the load shares for each path are adjusted, the individual paths from the next hop structures are mapped onto the next hop links by building a sum over the hash space portions of all paths sharing a link.

In order to bridge the seemingly contrary objectives of fast load adjustment and

the stability of network wide routing, OMP load balancing defines a very sophisticated mechanism for controlling the dynamics of load shifting. The algorithm always starts by shifting load at a predefined minimum rate. If the direction of load shifting persists, i.e. if the same link remains highest loaded in a next hop structure, the rate of load shifting (i.e. the load shares of the receiving paths) will increase exponentially. Once the direction of load shifting changes, those paths which have previously contained the critically loaded link start receiving traffic again, but only at half the rate they had had the last time before becoming critical. Overall, this sort of behavior enables reaching the fix point in traffic distribution very fast with a minimum of overshoots, and it enables stable routing behavior even in the presence of very variable traffic patterns.

In [76], the performance of the Optimized Multi-Path algorithm is evaluated and compared to the performance of two standard routing strategies, i.e. shortest path routing (SPR) and equal-cost multi-path routing (ECMP) throughout the investigated simulation scenarios. All experiments have been performed using a 13-node, 16-link network topology which corresponds to the National Science Foundation's (NSF) Internet backbone in 1989 (Figure 2.3.1) [18], and non-elastic traffic according to the Poisson model [77]. The link weights in the experiment were all set to 1, meaning that OMP's relaxed best path criterion merely reproduced the paths generated by ECMP.

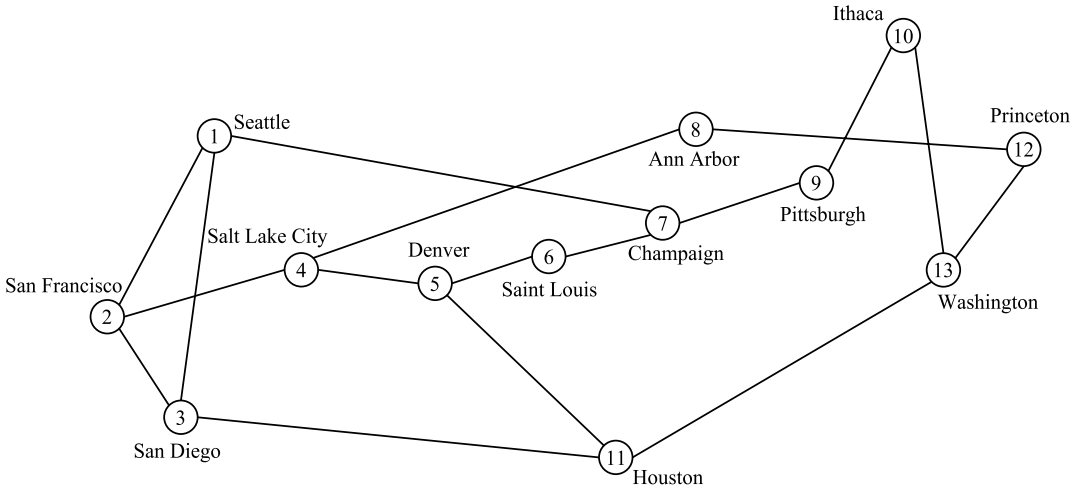



Figure 2.3.1: National Science Foundation (NSF) Internet backbone in 1989. (Source: [18])

In the first experiment, the performance of OSPF-OMP under a wide range of traffic intensities was compared to shortest path routing and ECMP. For this purpose, a simple traffic model was used in which all nodes send an equal number of messages to all other nodes, and different load levels were produced by a linear scaling of the traffic intensities across all nodes. The results of this experiment show that OMP performs better than both shortest path routing and ECMP at light, medium, and heavy network loads. The observed improvements in packet delivery time range from 2% to 7% for shortest path

routing, and 1% to 4% for ECMP. The performance of OMP is even more impressive in terms of packet loss, as with OMP routing no packet losses were observed with light and medium traffic load, while the other two routing strategies display noteworthy packet loss rates. In the presence of high traffic load, OMP did loose packets, however only at a fraction of the rate of shortest path routing and ECMP.

The second experiment examines OMP's response to changes in network wide traffic distribution. As shortest path routing and ECMP route traffic statically with respect to the traffic conditions in the network, OMP's traffic sensitive character should enable better network resource utilization for a much wider set of load allocations. The change in traffic distribution was produced as follows: in the first part of the simulation a uniform traffic distribution was used as in the first experiment. Subsequently, the load distribution was changed by introducing client-server like behavior by turning 5 of the 13 nodes into servers sending traffic to the remaining 8 nodes, which in turn did not send any traffic at all. The packet delivery times for shortest path routing and ECMP displayed an increase of approximately 19%, and remained at that level for the rest of the simulation due to their static nature. On the other hand, whereas OMP initially showed a similar rise in packet delivery times, 3 minutes after the start of the convergence process the packet delivery times started subsiding, and finally reached the level of only a slight 10% increase compared to the first phase of the simulation.

In the third experiment, the behavior of OMP in the presence of a sudden 3 seconds long surge of traffic volume has been investigated. The observed results show that OMP is able to accommodate such a sudden peak in traffic demand much better, as the traffic is much better distributed during stable traffic periods across the multiple paths compared to ECMP, such that the queues in the network have more memory capacity, thus leading to much smaller overall packet loss.

Simulations of OMP performed within the Internet Engineering Task Force's (IETF's) [www-5] process of algorithm standardization are presented in [www-6]. In these simulations OMP is evaluated using a 12-node, 19-links network, representing an IP backbone connecting major U.S. metropolitan areas, and the main focus of the investigation is set upon the allocation of network wide traffic, the convergence properties of the algorithm, and upon OMP's stability in the presence of adverse traffic conditions. In the experiment evaluating network wide traffic distribution by using a fix traffic matrix, OMP has been compared to simple shortest path routing and to ECMP. With ECMP the highest loaded link in the network displayed 109% link utilization, whereas this *worst link metric* could be reduced to only 89% using OMP. Furthermore, in another experiment, the convergence properties of the algorithm have been examined, and it has been shown that under stable traffic conditions OMP achieves link utilizations near the final convergence fix point already after 15 minutes operation, displaying only moderate potential for further improvements in subsequent time intervals.

Even more importantly, [www-6] evaluates the stability of OMP for two different

test cases: firstly, the algorithm's behavior in the presence of a fast rise in traffic is investigated, and secondly, OMP's reaction to high noise in traffic levels is examined. As far as situations are considered in which a fast rise in traffic can be observed followed by a plateau, OMP proves to behave very stably, closely following the changing traffic demands, and displaying only a slight overshoot at the moment when the plateau is reached, which is however expected, as OMP load balancing is only a lightly dampened controller. In the presence of noise, i.e. random traffic superimposed upon the otherwise stable traffic matrix, OMP also displays very stable behavior, without any tendencies towards falling into oscillations.

Additionally, it is important to note that the formulation of the load balancing algorithm explicitly yields the optimization goal of OMP: as each next hop structure (corresponding to an individual destination in the network) is always readjusted such that the maximally loaded link is offloaded, this traffic engineering scheme ultimately aims at minimizing the maximum link utilization in the network. In this context it is important to recall that traffic engineering schemes which exclusively focus upon minimizing the maximum link utilization are often prone to increasing the average link utilization in the network, as longer paths are often chosen in order to alleviate overload on congested links. As the *relaxed best path criterion* used by OMP proves to be a very powerful criterion for maximizing the number of multiple paths [15], thereby potentially increasing the average path length, the potential increase in average link utilization needs to be carefully evaluated when considering the deployment of OMP.

Last, but not least, it is important to mention that the algorithm's complexity might represent a major concern in the context of deployment in operational IP networks. The most important factors contributing to complexity are the large memory requirements needed for storing and managing the next hop structures in the individual nodes, as they contain entire multi-paths towards all destinations in the network. Even more importantly, the process of load adjustment is very complex, as separate computations need to be performed for each next hop structure, and as a different set of numerous variables need to be kept on a per destination basis. As a further aggravating factor, the signaling of load information using the link state flooding algorithm is an indeterministic process, which significantly hampers efforts towards estimating the worst case overhead of the algorithm.

### 2.3.2 Gallager's Minimum Delay Routing Algorithm

An alternative routing algorithm implementing minimum delay routing using distributed computation was presented by R. G. Gallager in [17]. The goal of the algorithm is to specify routing tables in each node of the network which will determine the distribution of traffic for each destination onto the available next hop links, such that the average delay experienced by packets traversing the network is minimized. In the algorithmic

formulation following this paragraph, it is shown that this goal is practically achieved by equalizing the marginal delays (i.e. the derivatives of delay) for all destinations on each viable output link in every node of the network. As a fully distributed mechanism, Gallager's algorithm is independently applied in each node of the network, updating the node's routing table based upon information about the marginal delay towards every destination node in the network which is exchanged with each adjacent node. A stable network topology is assumed with this scheme, as well as a stationary (or at least *near* stationary) traffic matrix. Notably, the routing determined by the algorithm is guaranteed to be loop free in each iteration leading to the state of convergence. The algorithm presented by Gallager is in many ways quite similar to the previously described routing algorithm of the early ARPANET (see Section 2.2.3), with the major difference that the ARPANET attempts to send each packet over a route that minimizes the packet's delay while completely disregarding other packet's delays, whereas here packets are sent via routes that minimize the average delay throughout the network.

In the remainder of this section, a compact formulation of Gallager's criteria is provided (see [78] for further details):

A computer network $G = (N, L)$ is comprised of $N$ nodes and $L$ links connecting them. Each link is defined as bidirectional, with possibly different link weights per direction.

Let $r_j^i \geq 0$ be the expected input traffic, measured in bits per second, entering the network at node $i$ and destined for node $j$. Let $t_j^i$ be the sum of $r_j^i$ and the traffic arriving from neighbors of $i$ for destination $j$. Furthermore, let routing parameter $\phi_{jk}^i$ be the fraction of traffic $t_j^i$ that leaves node $i$ over link $(i, k)$. Assume that the network does not lose any packets. Then, applying the conservation law follows

$$t_j^i = r_j^i + \sum_{k \in N^i} t_j^k \phi_{ji}^k \tag{2.3.2}$$

where $N_i$ is the set of neighbors of router $i$.

Let $f_{ik}$ be the expected traffic, measured in bits per second, on link $(i, k)$. Because $t_j^i \phi_{jk}^i$ is the traffic destined for node $j$ on link $(i, k)$, the following equation is used for finding $f_{ik}$:

$$f_{ik} = \sum_{j \in N} t_j^i \phi_{jk}^i \tag{2.3.3}$$

Note that $0 \leq f_{ik} \leq C_{ik}$, where $C_{ik}$ is the capacity of link $(i, k)$ in bits per second.

Furthermore, for each node $i$ and destination $j$, the routing parameters $\phi_{jk}^i$ satisfy the following conditions:

1. $\phi_{jk}^i = 0$ if $(i, k) \notin L$ or $i = j$. Clearly if the link does not exist, it cannot carry any traffic.

2. $\phi^i_{jk} \geq 0$. Negative amounts of traffic, of course, cannot be allocated.

3. $\sum_{k \in N^i} \phi^i_{jk} = 1$. All traffic must be allocated.

Let $D_{ik}$ be defined as the expected number of packets per second transmitted on link $(i,k)$ times the expected delay per packet, including the queueing delays at the link. It is assumed that the packets are delayed only by the links of the network, and that $D_{ik}$ depends only upon flow $f_{ik}$ through link $(i,k)$ and link characteristics such as propagation delay and link capacity. $D_{ik}(f_{ik})$ is a continuous and convex function. The total expected delay per message times the total expected number of message arrivals per second is given by:

$$D_T = \sum_{(i,k) \in L} D_{ik}(f_{ik}) \tag{2.3.4}$$

Note that the node traffic flow set $t = \{t^i_j\}$ and link flow set $f = \{f_{ik}\}$ can be obtained from $r = \{r^i_j\}$ and $\phi = \{\phi^i_{jk}\}$. $D_T$ can therefore be expressed as a function of $r$ and $\phi$ using (2.3.2) and (2.3.3). The minimum delay routing problem can now be stated as follows: for a given fixed topology with link capacities $C_{ik}$ and input traffic flow set $r$, and delay function $D_{ik}(f_{ik})$ for each link $(i,k)$, the minimization problem consists of computing the routing parameter set $\phi$ such that the total expected delay $D_T$ is minimized.

To solve this minimization problem, in [17] Gallager derived the necessary and sufficient conditions and described an algorithm to compute routing parameter set $\phi$ such that these conditions are satisfied. To find the conditions, first the partial derivative of the total delay $D_T$ of (2.3.4) with respect to $r$ and $\phi$ is obtained, that is,

$$\frac{\partial D_T}{\partial r^i_j} = \sum_{k \in N^i} \phi^i_{jk}[(D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r^k_j}] \tag{2.3.5}$$

$$\frac{\partial D_T}{\partial \phi^i_{jk}} = t^i_j[(D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r^k_j}] \tag{2.3.6}$$

where $D'_{ik}(f_{ik}) = \frac{\partial D_{ik}(f_{ik})}{\partial f_{ik}}$ and is called *marginal* or *incremental* delay. $\frac{\partial D_T}{r^i_j}$ is the marginal distance from node $i$ to node $j$.

Gallager's main theorem states that the necessary condition for a minimum of $D_T$ with respect to $\phi$ for all $i \neq j$ and $(i,k) \in L$ is

$$\frac{\partial D_T}{\partial \phi^i_{jk}} = \begin{cases} = \lambda_{ij} & \phi^i_{jk} > 0 \\ \geq \lambda_{ij} & \phi^i_{jk} = 0 \end{cases} \tag{2.3.7}$$

where $\lambda_{ij}$ is some positive number, and the sufficient condition to minimize $D_T$ with respect to $\phi$ is for all $i \neq j$ and $(i,k) \in L$ is

$$D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_j^k} \geq \frac{\partial D_T}{\partial r_j^i} \tag{2.3.8}$$

Equation (2.3.5) shows the relation between node's marginal distance and the marginal distance of neighbors to a particular destination. Equations (2.3.6), (2.3.7), and (2.3.8) indicate that under *perfect load balancing*, i.e. when routing parameter set $\phi$ yields the minimum delay, the marginal distances through neighbors in the successor set are equal, and the marginal distances through neighbors not in the successor are higher than those in the successor sets. Let $D_j^i$ denote the marginal distance $\frac{\partial D_T}{\partial r_j^i}$ from $i$ to $j$. Let $l_k^i$ denote the marginal delay $D'_{ik}(f_{ik})$ as the cost of the link from $i$ to $j$. Let $S_j^i$ be the set of neighbors through which router $i$ forwards traffic towards $j$. Now the minimum delay routing problem becomes one of determining routing parameters $\{\phi_{jk}^i\}$ at each node $i$ for each destination $j$.

In order to achieve an optimal traffic distribution, each node $i$ must incrementally decrease those routing variables $\phi_{ik}^j$ for which the marginal delay $D'_{ik}(f_{ik}) + \frac{\partial D_T}{\partial r_j^k}$ is large, and increase those for which it is small. Concerning the exchange of routing information, in Gallager's approach each node must send information to all of its neighbor nodes about its marginal delays towards each destination in the network. The amount of information sent using this routing strategy exactly corresponds to that of ARPANET routing, with the essential difference that instead of the delays, Gallager's algorithm exchanges values of marginal delay [17]. Following its initial formulation, Gallager's algorithm has been further refined in [79, 80] to handle topological changes, based upon the techniques proposed in [81], thus gaining more significance in the context of practical real-world implementations.

## 2.4 Concluding Remarks

Over the last years, measurements performed by different Internet Service Providers (ISPs) have shown that traffic demands in the networks change dynamically, and that they exhibit large time-of-day variations [41]. This provides strong motivation for optimizing the performance of operational networks by adapting the allocation of traffic in the network to the current conditions. In order to achieve this, different traffic engineering techniques have been proposed in recent years, yielding several different groups of possible approaches.

Link weight optimization represents a rather straightforward possibility for optimizing the network wide allocation of traffic. The main advantage of this method is that it does not require any changes to the underlying networking technologies, as performance improvements are provided only by means of configuration. However, this approach also goes hand in hand with the mandatory requirement of knowing the traffic matrix (or even more preferably the traffic *demand* matrix) for the entire network, as well as a well-

designed process for conveying the calculated link weights into router configurations. As networks usually need to be reoptimized periodically, this configuration task requires either a large network management overhead paired with detailed understanding of the technique, or the presence of an automated network management infrastructure, which may be associated with significant investments.

The situation is also very similar with traffic engineering based upon multi-protocol label switching (MPLS), which either performs optimization of network wide label switched paths (LSPs), or applies online algorithms for the setup of LSPs in real time. In any case, this approach requires the deployment of a completely novel traffic forwarding technique, which in many cases may be associated with huge investments into the network infrastructure.

Although the Optimized Multi-Path algorithm (OMP) combines many favorable properties, due to its large data structures and load signaling based upon link state protocols' flooding algorithm OMP at the same time produces significant (and unpredictable) overhead both in terms of memory requirements and signaling. In contrast, Gallager's algorithm produces minimal signaling overhead, as only the marginal delay to each destination node needs to be communicated to neighbor nodes, but unfortunately the algorithm can hardly be applied to real networks due to its requirements concerning traffic dynamics (as only quasi-static traffic matrices are allowed) and the setting of comprehensive initial load distribution parameters.

Summarizing the main features and properties of the described approaches, the conclusion is reached that for each of these mechanisms there is still a lot of room for further improvements. As the main contribution of this thesis, the following chapter will introduce Adaptive Multi-Path routing (AMP) as a traffic engineering technique which aims to combine the most favorable properties of the existing approaches in a novel and original algorithmic solution.

# Chapter 3

# Adaptive Multi-Path Routing (AMP)

The main objectives which led to the development of AMP were to design a mechanism which would provide simple and fully automated traffic engineering by performing continuous load balancing within individual routing domains (cf. Section 1.1). At the same time, no management overhead was supposed to be added, and additionally the algorithm was supposed to run completely autonomously within the routing plane of the nodes, producing only a minimal overhead in terms of signaling, memory, and computational complexity. The proposed algorithm was supposed to be deployable in current intra-domain routing environments, and to be implementable in state-of-the-art networking hardware.

## 3.1 Basic Idea

Trying to retain OMP's advantages in terms of the level of automation and the use of multi-path structures, and at the same time aiming at alleviating the mentioned drawbacks of such an approach, primarily regarding the global signaling using message flooding, AMP chooses to reduce its view of the network to the local scope. Therefore, with AMP, an arbitrary network node $X$ does not not know about the state of all links in the network, but in contrast it is only informed about its immediate neighborhood. In this sense, the propagation of congestion information throughout the network is based upon a so-called *backpressure mechanism*. For an intuitive description of this concept, the IP network can be perceived as a meshed system of unidirectional porous rubber tubes transporting viscous fluid. As soon as the fluid hits upon resistance, e.g. due to a very tight section or an obstacle (corresponding to a congestion situation in the network), this leads to an instantaneous build-up of local pressure with two possible consequences: the pressure starts propagating opposite to the flow direction (backpressure), tube after tube with decreasing intensity due to the viscosity of the fluid, eventually leading to the

establishment of a new global pressure/loss equilibrium. Secondly, persistent pressure (congestion) may also cause fluid to leak locally through the rubber, corresponding to packet loss in the network.

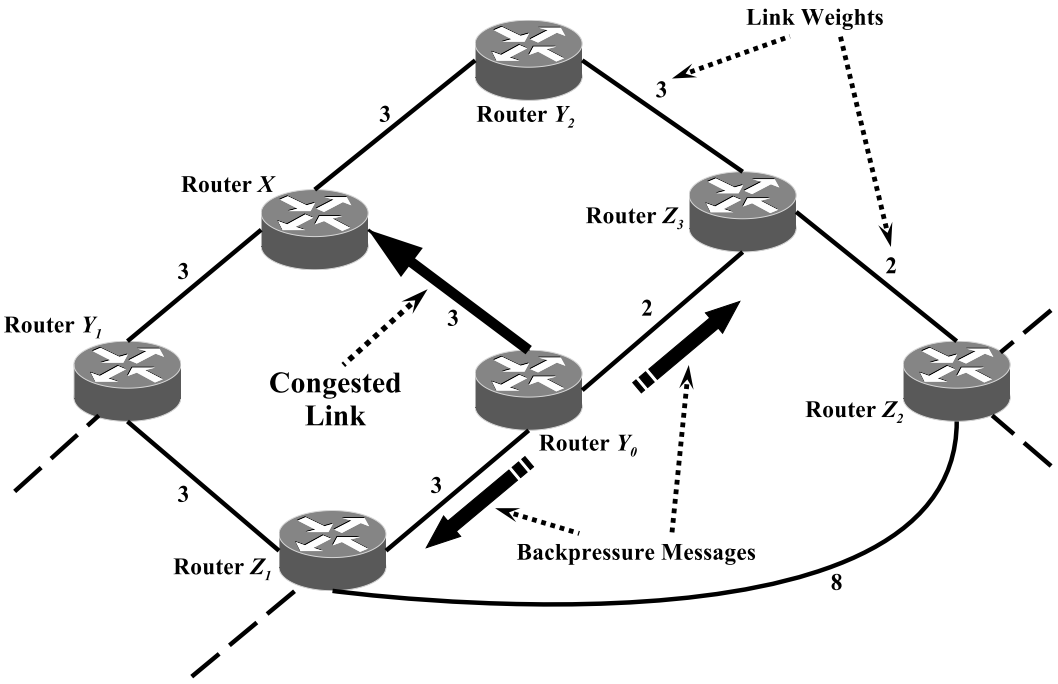



Figure 3.1.1: Basic idea of the backpressure mechanism.

Following this intuitive description, the interplay between local dissemination and global propagation of load information is described more formally. Consider an arbitrary node $X$ having a set $\Omega_X$ of neighbor nodes $Y_i \in \Omega_X$, $i \geq 0$, and let $\overline{XY_i}$ denote the generic directed link from node $X$ to node $Y_i$. The main principle of AMP's operation is depicted in Figure 3.1.1: In contrast to OMP, where an increase in utilization on link $\overline{Y_0X}$ causes nodes all over the network to offload some of their paths containing link $\overline{Y_0X}$, with AMP the only node to immediately react is $Y_0$ as end-node of $\overline{Y_0X}$, trying to shift traffic away from $\overline{Y_0X}$ onto alternative paths. Additionally, node $Y_0$ periodically sends out so-called *backpressure messages* (BMs) to each adjacent node $N_j \in \Omega_{Y_0}$ in order to inform $N_j$ about its contribution to the congestion situation on link $\overline{Y_0X}$. All $N_j \in \Omega_{Y_0}$ in turn pass on this information to their own neighbor nodes, again in proportion to those nodes' respective contributions to the congestion situation, etc.

Due to its well designed signaling mechanism, AMP manages to combine the favorable properties of both OMP and the approach by Gallager. It applies traffic engineering by operating completely autonomously in the nodes of the network, making use of topological redundancies (i.e. the presence of multiple paths), and at the same time it keeps the exchange of load information completely local, thus enabling very low signaling over-

head. Furthermore, the algorithm does not impose any limitations upon the character of input traffic, tolerating traffic matrices with an arbitrary degree of traffic variability.

## 3.2 Algorithmic Details of AMP

The next sections will describe in detail each part of the Adaptive Multi-Path algorithm, including the calculation of multiple paths, link load metrics, the signaling mechanism, packet forwarding, and the load balancing algorithm.

### 3.2.1 Calculation of Multiple Paths

One of the fundamental prerequisites for any traffic engineering scheme is that the network topology exhibits a sufficient amount of redundancy which would enable traffic to be deflected onto alternative paths in the case one or several links in the network become congested. However, topological redundancy is not sufficient in its own, as the applied routing architecture must incorporate principles and mechanisms which allow for its practical use.

As already discussed in the previous chapter, in the current IP intra-domain routing architecture normally only the shortest paths are used, i.e. paths which minimize the sum of weights of all comprising links. In addition to using only a single shortest path, the Equal Cost Multi-Path scheme (ECMP) allows for multiple shortest paths to be used provided they exist, and splits the traffic uniformly upon all paths available.

As one of the most important objectives when developing AMP was to stay within the current intra-domain routing architecture, a careful way of increasing the number of multiple paths compared to ECMP was required. In this context, it is important to notice that unlike with connection-oriented approaches like Multi-Protocol Label Switching (MPLS) the current architecture imposes severe limitations with respect to the number of multi-path options. Whereas with MPLS all possible combinations of paths within a particular network are feasible, the link weight semantics can hardly be tweaked.

Therefore, AMP offers the option of using the *relaxed best path criterion* as proposed in [12, 70] (see Section 2.3.1). This criterion states that in each node of the network each neighbor node which is closer to the destination node (in terms of the sum of link weights) can be used as a viable hop for forwarding traffic towards that destination. Note that this criterion holds great potential of increasing the number of multiple paths in the network, as packets do not need to be forwarded solely along the shortest paths. At the same time routing loops are reliably avoided, as the distance of each packet towards its destination always decreases along each hop of its path.

In order to illustrate this criterion's potential for increasing the network wide number of multiple paths, Figure 3.2.1 provides the same example topology as Figure 3.1.1 with

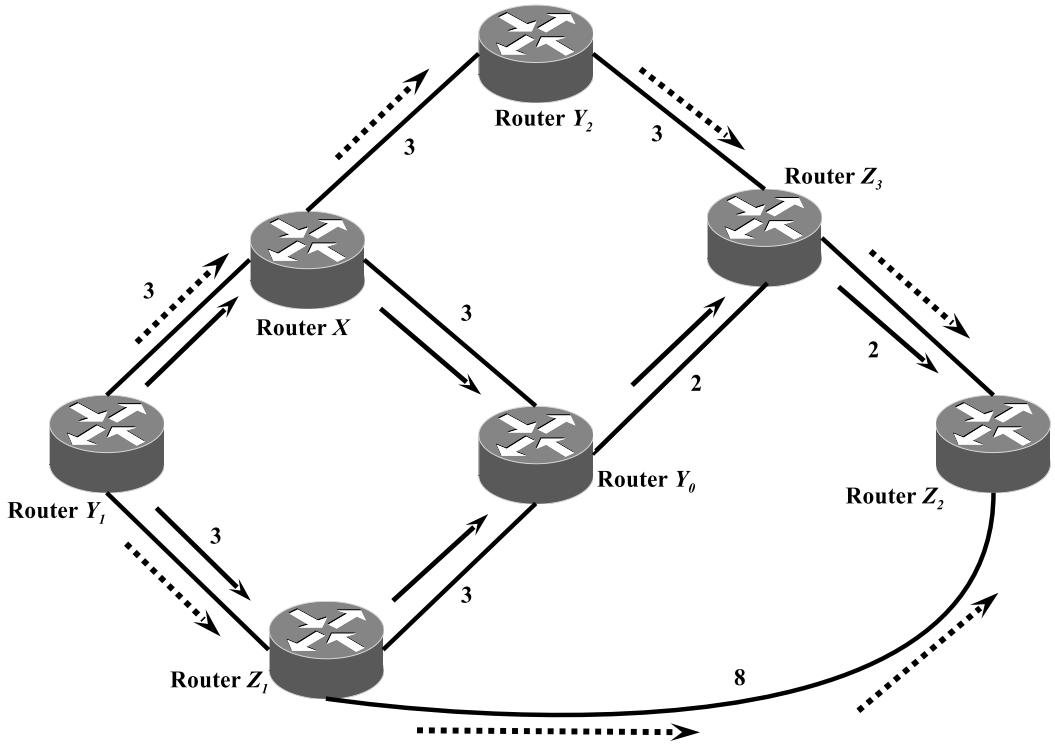



Figure 3.2.1: Additional paths obtained using the *relaxed best path criterion*.

the shortest path from node $Y_1$ to node $Z_2$ denoted with full line arrows, whereas the additionally obtained paths are represented by dotted line arrows. Besides the two shortest paths $\overline{Y_1XY_0Z_3Z_2}$ and $\overline{Y_1Z_1Y_0Z_3Z_2}$ with a total metric (i.e. sum of link weights) of 10, in the nodes $X$ and $Z_1$ two additional paths bifurcate according to the mentioned criterion. In the case of the path $\overline{Y_1XY_2Z_3Z_2}$ note that in node $X$ the next hop node $Y_2$ is closer to the destination $Z_2$ than node $X$ itself (distance of 5 vs. distance of 7), and thus node $Y_2$ becomes a viable next hop for forwarding traffic to node $Z_2$ in node $X$. Similarly, in the case of the path $\overline{Y_1Z_1Z_2}$, node $Z_1$ is allowed to route directly to node $Z_2$ according to the relaxed best path criterion, as node $Z_2$ is per definition a distance of 0 away from itself.

The potential of the relaxed best path criterion in increasing the number of multiple paths was demonstrated in [15], where the number of equal cost multi-paths and relaxed best paths has been investigated for different topologies. For the purpose of that analysis, numerous random Waxman topologies were generated using the Georgia Tech Topology Generator, with the link weights set randomly to integer values in the range from 1 to 5 [82, www-8]. Motivated by the observation that long-distance links are generally more expensive, Waxman's model places nodes at random in a two-dimensional space and adds links probabilistically between individual pairs of nodes, inversely proportional to their distance. For each random network size from 10 to 100 nodes, the average values for 5 different random topologies have been considered.

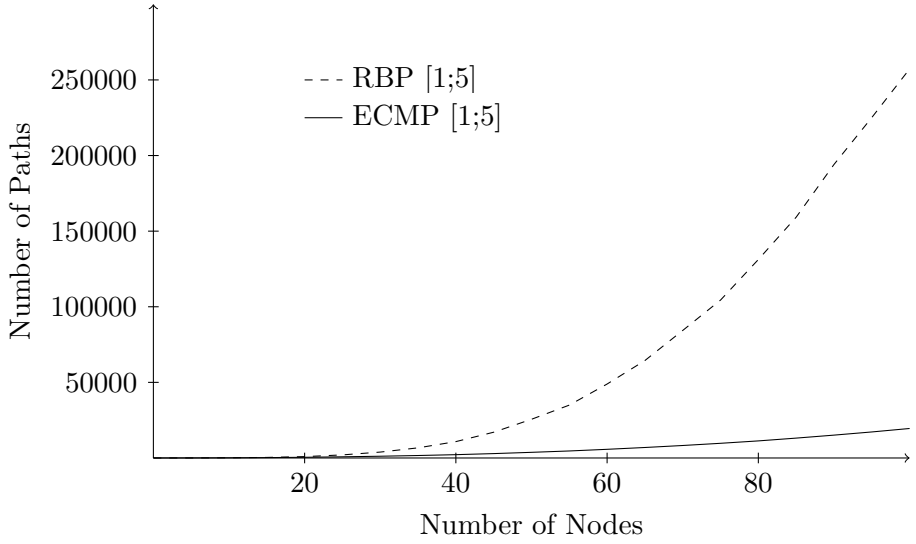



Figure 3.2.2: Comparison of the absolute number of Equal Cost Multi-Paths (ECMP) and Relaxed Best Paths (RBP) in artificial topologies.

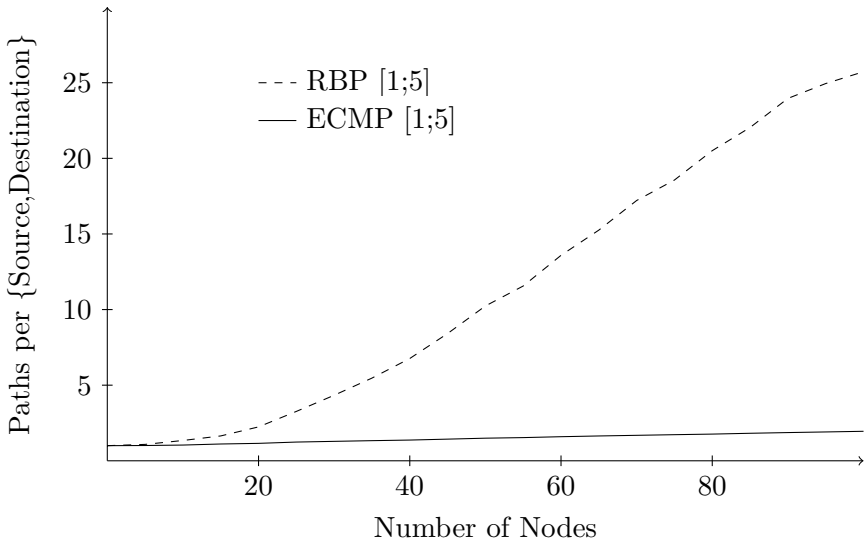



Figure 3.2.3: Number of multiple paths per {Source, Destination} pair in artificial topologies.

The results of the investigations are shown in Figures 3.2.2 and 3.2.3. Figure 3.2.2 compares the number of ECMP paths and relaxed best paths for different sizes of artificial topologies. Interestingly, the number of total ECMP paths in the network grows only moderately with increasing network size, whereas the number of relaxed best paths displays fast growth. Figure 3.2.3 presents the obtained results using a different scale on the y-axis, providing the number of multi-paths for each source-destination pair in the network. In this figure, the difference between ECMP and the relaxed best path criterion is even more impressive, with hardly any growth noticeable with ECMP, while the number of multi-paths shows steady growth, reaching the value of approximately 10 for networks of 50 nodes, and 26 for networks with 100 nodes.

In order to maximize the practical use of the relaxed best path criterion in real topologies, a simple condition for link weight settings is derived in [70], which states that if all the link weights in the network are set within a ratio of $1 + 1/(d-1)$ of each other, where $d$ denotes the network diameter, all minimum hop multi-paths between any pair of nodes satisfy the relaxed best path criterion. Although this criterion does not increase the total number of paths compared to using minimum hop multi-paths in identical topologies, it does represent a valuable contribution in practical cases where the link weights may have to assume different values for various administrative purposes. Overall, based upon the topology and link weight settings used, with AMP network managers can choose whether they wish to apply the relaxed best path criterion, or simply let AMP operate on the set of existing multiple (strictly-)shortest paths.

### 3.2.2 Link Load Metrics

A meaningful choice of link load metrics is crucial for making appropriate load balancing decisions. This is especially true in the case of elastic traffic like TCP, where all connections tend to make maximum use of the available capacity, meaning that simple link utilization $\rho$, defined as

$$\rho = \frac{Carried\ Traffic\ Volume}{Link\ Capacity \cdot Time\ Period} \tag{3.2.1}$$

may often approach the 100% limit. However, if two or more links display almost equal, near 100% link utilizations, this does not necessarily mean that they are equally loaded, i.e. the levels of load offered by the corresponding traffic sources might differ dramatically, and the load balancing algorithms should capture this fact correctly. Fortunately, in the case of TCP traffic which is predominant in today's Internet, the levels of offered load can be efficiently estimated by evaluating the packet loss probabilities on the links, based upon the fact that in the presence of congestion TCP roughly slows down in proportion to the square root of the packet loss probability observed on the link [74]. For packet loss levels lower than 1%, [74] makes the assumption that TCP does not introduce any slowdown, meaning that below this boundary simple link utilization from

(3.2.1) is sufficient for estimating the offered link load. Summarizing these observations of TCP dynamics, the following metric for estimating link load called *equivalent load*, $\rho'$, is derived in [12], and in the same form it is also applied in the context of AMP, where $P$ describes the packet loss probability on the link:

$$\rho' = \max\{\rho,\ \rho \cdot K \cdot \sqrt{P}\} \tag{3.2.2}$$

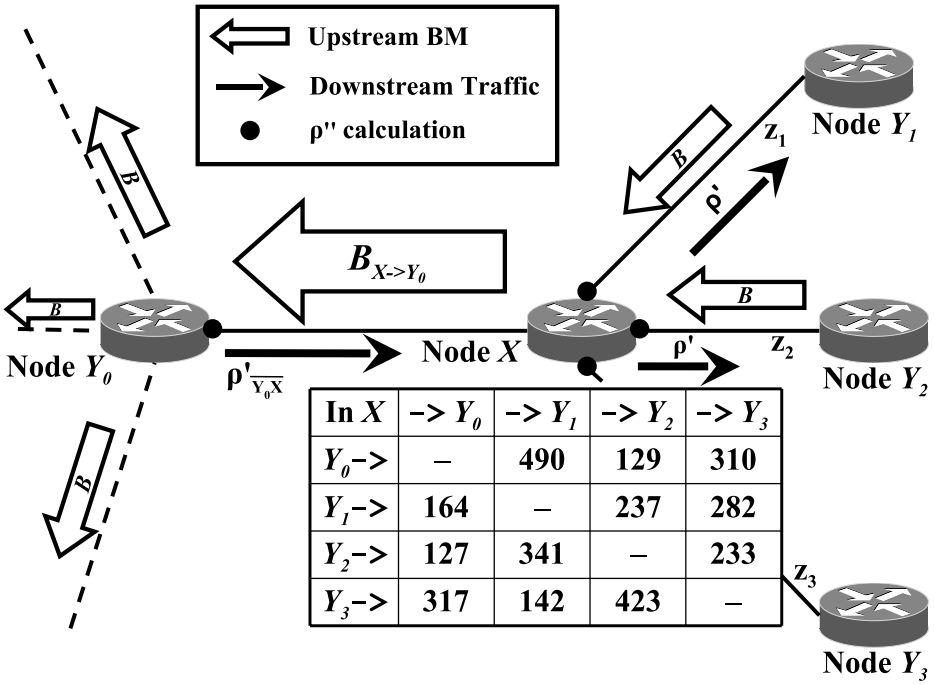
In (3.2.2), the scaling factor $K$ determines the packet loss boundary at which $\rho'$ exceeds the plain link utilization $\rho$ from (3.2.1), meaning that for $K > 1/\sqrt{P}$ the metric will react sensitively to packet losses. The recommended default value for $K$ is 10, as the introduced metric is supposed to exceed plain link utilization only for packet loss values above 1%. Accordingly, the maximum value which the equivalent load metric may assume is 10, reflecting a link with a hypothetical packet loss level of 100%.

The potential impact of this specific choice of link load metric upon the stability of the load adjustment algorithms is further discussed in [12]. The load adjustment should remain stable as long as the first derivative of the equivalent load metric over the offered load stays positive. If within some region this derivate becomes negative, then the load adjustment will oscillate within that range. This effect will be most profound with drop-tail queueing strategies and small buffer settings, as in such cases the plain link utilization in the presence of high offered load may drop significantly below 100%, while the packet loss levels increase only moderately. In order to counteract this effect, the formulation of the metric as given in (3.2.2) introduces a rather aggressive compensation for loss, making such oscillations very unlikely, and bounding them to a very small range in case they do occur.

### 3.2.3 Adaptive Multi-Path Routing (AMP) Signaling

The signaling mechanism represents the essential part of the Adaptive Multi-Path algorithm (AMP), and at the same time one of the central contributions of this work. While Section 3.1 provides an intuitive description of the main principles of AMP operation, a concise statement of the signaling mechanism is provided in this section. Consider link $\overline{Y_0X}$ from Figure 3.2.4 as an example for describing the information required by $Y_0$ for each of its output links. $Y_0$ essentially requires two types of information: firstly, the equivalent load on the link $\overline{Y_0X}$, and secondly, information about the extent to which traffic routed from $Y_0$ via $X$ contributes to congestion on the links $\overline{XY_i}$, $i > 1$, as well as the links in the network further downstream. As $Y_0$ can directly measure and calculate $\rho'$ for link $\overline{Y_0X}$ itself, the remaining information required by $Y_0$ has to be obtained from other nodes by means of signaling. For this purpose, AMP introduces a specific type of signaling messages sent between adjacent nodes called *backpressure messages* (BMs).

$BM(X, Y_0)$, i.e. the backpressure message sent from node $X$ to node $Y_0$, should contain both directly measurable information about the explicit load situation on links




| In X | -> $Y_0$ | -> $Y_1$ | -> $Y_2$ | -> $Y_3$ |
|---|---|---|---|---|
| $Y_0$-> | – | 490 | 129 | 310 |
| $Y_1$-> | 164 | – | 237 | 282 |
| $Y_2$-> | 127 | 341 | – | 233 |
| $Y_3$-> | 317 | 142 | 423 | – |

Figure 3.2.4: Example for a Backpressure Message (BM) from $X$ to $Y_0$ and *in/out matrix* in node $X$.

$\overline{XY_i}$, $i \geq 1$, as well as indirectly obtained information about the situation further downstream as reported to node $X$ by nodes $Y_i$, $i = 1, 2, 3, ..., n$, where $n$ is the number of output links for node $X$. In order to keep the backpressure messages small, these $2n$ parameters are eventually mapped to a scalar $B_{X\to Y_0}$ describing the congestion situation on the downstream link of node $X$. For reasons of simplicity, from now on the generic $BM(X, Y_0)$ is identified with its respective scalar content $B_{X\to Y_0}$. Then, $B_{X\to Y_0}$ may be considered as a function $f$ of the mentioned $2n$ parameters:

$$B_{X\to Y_0} = f(\rho'_{\overline{XY_1}}, ..., \rho'_{\overline{XY_n}}, B_{Y_1\to X}, ..., B_{Y_n\to X}) \tag{3.2.3}$$

To describe $f$, the independence of the output links is used such that the number of parameters is reduced to one per link, summarizing the situation on each link $\overline{XY_i}$ with a function $g$, i.e.,

$$g_i = g(\rho'_{\overline{XY_i}}, B_{Y_i\to X}) \quad \forall i = 1, ..., n \tag{3.2.4}$$

As neither the output link nor the network beyond should be overloaded, the *maximum* function is a good candidate for $g$. This leads to the interpretation of $g_i$ as *effective equivalent load* $\rho''$ for link $\overline{XY_i}$:

$$\rho''_{\overline{XY_i}} = \max\{\rho'_{\overline{XY_i}}, B_{Y_i\to X}\}, \quad i = 1, ..., n \tag{3.2.5}$$

The calculation of $B_{X\to Y_0}$ (3.2.3) is additionally refined by summarizing the $n$ parameters $g_i \equiv \rho''_{\overline{XY_i}}$ according to a function $h$:

$$\begin{aligned} B_{X\to Y_0} &= f(\rho'_{\overline{XY_1}}, B_{Y_1\to X}; ...; \rho'_{\overline{XY_n}}, B_{Y_n\to X}) \\ &= h(g(\rho'_{\overline{XY_1}}, B_{Y_1\to X}), ..., g(\rho'_{\overline{XY_n}}, B_{Y_n\to X})) \\ &= h(\rho''_{\overline{XY_1}}, ..., \rho''_{\overline{XY_n}}) \end{aligned} \tag{3.2.6}$$

Finally, $h$ is defined as a weighted sum of the $g_i$, where the weight for the link $\overline{XY_i}$ corresponds to the ratio between traffic on link $\overline{XY_i}$ that has arrived from node $Y_0$ via $X$ and the total traffic on link $\overline{XY_i}$. This sum combines all information which is available to node $X$ about $Y_0$'s contribution to the congestion situation on the downstream part of the network:

$$B_{X\to Y_0} = \sum_{Y_i\in\Omega_X\setminus Y_0} \frac{\beta_{\overline{XY_i}}(Y_0)}{\beta_{\overline{XY_i}}} \cdot \rho''_{\overline{XY_i}} \tag{3.2.7}$$

Remember from Section 3.1 that $\Omega_X$ is the set of all neighbor nodes of node $X$, the downstream link between nodes $X$ and $Y_i$ is called $\overline{XY_i}$, $\beta_{\overline{XY_i}}(Y_0)$ is the number of bytes sent from node $Y_0$ via $X$ to $Y_i$. Finally, $\beta_{\overline{XY_i}}$ denotes the total number of bytes sent from any node $\in \Omega_X \setminus Y_i$ via $X$ to $Y_i$.

Note that calculating $\beta_{\overline{XY_i}}(Y_0)$ in (3.2.7) requires node $X$ to map precisely traffic on the input link $Y_0X$ to the output links $\overline{XY_i}$, $i = 1, ..., n$. This mapping is described in a so-called *in/out matrix* stored in node $X$ (see Figure 3.2.4). This matrix contains for every node pair $(P, Q)$, $P, Q \in \Omega_X$, $P \neq Q$, the number of bytes carried between these nodes via $X$.

As fast propagation of load information throughout the network domain is a strongly preferred property with AMP due to its local view of the network, it is recommended that the backpressure messages be exchanged relatively frequently between adjacent nodes. Therefore, one second is envisioned as the default interval between two consecutive backpressure messages, i.e. $\tau_{BM}$=1s. At the same time, the generated overhead in terms of link load is very low, as the BMs carry only a single scalar value, resulting in a total BM packet size which is only slightly larger than the minimal message size of the applied intra-domain routing protocol.

### 3.2.4 Packet Forwarding

In the current IP intra-domain routing architecture, the existence of multiple paths in network topologies is hardly used for the purpose of traffic engineering. As the only mechanism currently deployed, Equal Cost Multi-Path routing (ECMP) enables the distribution of traffic among multiple paths only if they all satisfy the shortest path

criterion, i.e. if they all have the same minimal path length [8]. Apart from this very limiting constraint, ECMP additionally distributes traffic only uniformly among multiple next hops, although the possibility of distributing traffic unevenly is generally preferable for the purposes of traffic engineering. From a technical point of view, ECMP offers three principal ways of forwarding packets onto the next hops:

- Per packet round robin,
- Dividing destination prefixes among available next hops in the forwarding entries,
- Dividing traffic according to a hash function applied to the source destination pair.

The first technique of per packet round robin represents by far the most simple mechanism, but it is applicable only if the delay differences on the paths are very small, as otherwise TCP connections will experience packet disordering and thus display very poor performance. The second technique of dividing destination prefixes among multiple next hops is not recommendable in general, as very short IP destination prefixes will lead to a very coarse granularity of load shifting.

Many hashing based schemes offer very desirable properties for traffic distribution. In the case of the most widely applied hash function for load balancing purposes, the 16-bit Cyclic Redundancy Check (CRC-16), realistic IP traffic is very evenly spread across the hash space, making this scheme the prime choice for Internet load balancing [72]. The main principle of CRC-16 based load balancing is to perform a hash of the source and destination IP address for each packet, and for each destination address to determine boundaries within the solution space. The relative sizes of the hash space subsets defined in this way for each destination then represent the relative portions of the volumes of traffic which should be routed via the corresponding viable next hops for that destination. Additionally, this technique's 16-bit structure enables very high granularity of load shifting, as $1/2^{16} = 1/65536$ shares of traffic can be manipulated. Employed in the context of ECMP, this scheme consistently ensures that the traffic will be uniformly distributed among the available next hops, as the relative hash space shares are always equally sized with this routing strategy.

In the case of AMP, the CRC-16 strategy is also applied, but with the fundamental difference that traffic is generally forwarded to the next hops in unequal shares. This is accomplished by allocating shares of hash space to individual next hops in proportion to the amount of traffic which should be sent out on the respective links. An example load distribution in router $X$ from Figure 3.2.1 is displayed in Table 3.2.1: According to the *relaxed best path* routing rule, a subset of the three neighbor routers of $X$ are determined as viable next hops, and a portion of the hash space is allocated to each of them. In the case that packets are destined to the immediate neighbor routers $Y_0$, $Y_1$, and $Y_2$, the only viable next hops in this configuration will be these nodes themselves, resulting in the entire hash space to be allocated to a single next hop (first three lines

| | Next Hop $Y_0$ | Next Hop $Y_1$ | Next Hop $Y_2$ |
|---|---|---|---|
| Destination $Y_0$ | [0, 65535] | – | – |
| Destination $Y_1$ | – | [0, 65535] | – |
| Destination $Y_2$ | – | – | [0, 65535] |
| Destination $Z_1$ | [0, 32767] | [32768, 65535] | – |
| Destination $Z_2$ | [0, 38353] | – | [38354, 65535] |
| Destination $Z_3$ | [0, 23946] | – | [23947, 65535] |

Table 3.2.1: Routing table in Router $X$ from Figure 3.2.1 displaying *per destination* CRC-16 hash space ranges allocated to individual next hops.

in Table 3.2.1). In the case of the destinations $Z_1$, $Z_2$, and $Z_3$, the mentioned routing rule yields two viable next hops, meaning that the hash space must be subdivided in two separate shares for each of them. In this example, the hash space for destination $Z_1$ is divided uniformly among the next hops $Y_0$ and $Y_1$ – accordingly, all packets destined for $Z_1$ whose CRC-16 calculations over their {source, destination} address pairs yield a value less or equal to 32767 will be routed via $Y_0$, whereas packets with hash values in the range [32768, 65535] will be forwarded via $Y_1$. In the case of the destinations $Z_2$ and $Z_3$, unequal distributions of traffic are chosen, with the majority of traffic destined for $Z_2$ routed via $Y_0$, whereas the majority of traffic towards $Z_3$ is forwarded to $Y_2$. The exact way in which AMP determines the mentioned hash space ranges will be described in the next section, which provides the concise formulation of the load balancing algorithm.

### 3.2.5 AMP Load Balancing

Whereas AMP's signaling architecture represents a completely new and original approach, for the purpose of load balancing a modified version of the algorithm from [12] is applied, in which the load balancing decisions are based exclusively upon the local view of the network in the node, as obtained from load measurements and the backpressure messages (BMs) described in Section 3.2.3. The main motivation for resorting to an existing algorithm lies in its beneficial dynamic properties: the proposed algorithm achieves both fast convergence towards a fix point in traffic distribution, and it also operates stably even in the presence of very noisy traffic patterns (see [www-6]).

The load balancing algorithm of AMP performs *per destination* load shifting, meaning that in each iteration the algorithm will adjust the relative hash space shares of the individual next hops for each destination prefix separately, with the objective of equalizing the *effective equivalent load* value, $\rho''$, observed on all of its output links.

Whenever AMP needs to readjust traffic shares for a particular destination, the node performing the adjustments goes beyond considering only the current traffic distribution and signaling information. Moreover, the node keeps track of previous actions in order to ensure a rapid but nevertheless stable convergence towards an equilibrium of load on

each output link: as long as the underloaded output links in each node have got room for accepting traffic from the *critically* (i.e. highest) loaded link, the percentage of overall traffic shifted onto the less loaded links within a single control period keeps growing exponentially. In case one of the receiving links changes status and becomes critical, the load balancing algorithm will start shifting traffic away from the new critical output link onto the other links, again in exponentially increasing steps. Finally, when the next link becomes critical, the direction of load shifting again changes, etc.

As far as the implementation of the algorithm is concerned, it is important to note that it is based upon storing the current load balancing step size for each output link on a *per destination* basis. Beyond enabling the exponential growth of load balancing step sizes (and thus ensuring agility and fast convergence of the algorithm), the manipulation of the step size parameter is also crucial for guaranteeing the algorithm's stability, and is based upon the following principle: each time a node ceases to be critical, it again starts accepting load, but only with half of the rate it had had before the last time it became critical. The main consequence of such design is that the load balancing step sizes will multiplicatively decrease as the *is critical* status is being passed around among the output links in a quasi round robin fashion, which happens whenever pairs of link utilizations are approximately equal. This behavior quickly leads towards the stabilization of routes as the step sizes will converge towards minimal values around the targeted fix point in traffic distribution.

Furthermore, the load balancing interval (i.e. the time period between two consecutive steps of the load balancing algorithm) is chosen such that it is *at least* an order of magnitude larger than the control period of TCP (which corresponds to the connections' *round trip time*). This substantial difference in the time scales of operation should reliably ensure that the control mechanisms of TCP (as the predominant type of Internet traffic) and AMP do not interfere. Accordingly, the default setting of AMP's load balancing interval, $\tau_{LB}$, is set to 5 seconds, i.e. $\tau_{LB} = 5$s.

The concise statement of the algorithm executed separately in each node of the network is given below, where $D$ represents the set of all destinations in the network, $L$ the set of all viable next hop links for a particular destination in the computing node, and $\alpha$, $\beta$ integer values determining the convergence speed of the algorithm, with the default settings of $\alpha = 4$ and $\beta = 4$:

```
AMP Load Balancing (D, L, α, β)
 1  for  each destination d ∈ D
 2     do
 3
 4        % First the status of the viable next hop links is being updated
 5        for  each link l ∈ D
 6           do
```

```
7            update IsCritical(l) status ; % Check if l is highest loaded
8            update WasCritical(l) status ; % Check if l was prev. highest loaded
9
10       % Next the move increments (i.e. load shift sizes) are being determined
11       for  each link l ∈ L(D)
12           do
13               if IsCritical(l) = TRUE
14                 then
15                       continue;
16               if IsCritical(l) = FALSE
17                 then
18                       if MoveCount(d, l) > β
19                         then
20                               Increase(d, l) := MoveIncrement(d, l)/α;
21                         else  Increase(d, l) := MoveIncrement(d, l)/(2 · α);
22                       if Increase(d, l) < 1
23                         then
24                               Increase(d, l) := 1;
25                       MoveIncrement(d, l) := MoveIncrement(d, l) + Increase(d, l);
26                 else  if MoveIncrement(d, l) > 1
27                         then
28                               MoveIncrement(d, l) := 1;
29                       MoveIncrement(d, l) := MoveIncrement(d, l)/2;
30                       MoveCount(d, l) := 0;
31               if MoveIncrement(d, l) < 1
32                 then
33                       MoveIncrement(d, l) := 1;
34               if MoveIncrement(d, l) > 65535
35                 then
36                       MoveIncrement(d, l) := 65535;
37
38       % Finally, traffic is being shifted
39       for  each link l1 ∈ L(D)
40           do
41               if IsCritical(l1) = FALSE
42                 then
43                       continue;
44               for  each link l2 ∈ L(D)
45                  do
46                     if IsCritical(l2) = TRUE
```

```
47                  then
48                      continue;
49              Move := MoveIncrement(d, l2);
50              if Move < 1
51                  then
52                      Move := 1;
53              if Move > 65536 − TrafficShare(d, l2))
54                  then
55                      Move := 65536 − TrafficShare(d, l2);
56                      MoveIncrement(d, l2) := Move;
57              if Move > TrafficShare(d, l1)
58                  then
59                      Move := TrafficShare(d, l1);
60              TrafficShare(d, l2) := TrafficShare(d, l2) + Move;
61              TrafficShare(d, l1) := TrafficShare(d, l1) − Move;
```

## 3.3 Concluding Remarks

Starting with the objective of designing a mechanism which would provide fully automated traffic engineering by performing load balancing continuously within individual autonomous systems, this chapter has presented Adaptive Multi-Path routing (AMP) as a simple, efficient, and light-weight solution in the context of the current intra-domain routing architecture.

The central innovation of AMP consists of the local exchange of load information based upon the *backpressure mechanism*, which manages to fulfill seemingly contrary objectives of efficient signaling in terms of bandwidth consumption and fast propagation of load information throughout the network domain. After presenting AMP's extensions to the current intra-domain routing algorithms in terms of multi-path calculation and the applied load metrics, the signaling architecture of AMP has been presented in detail. Subsequently, packet forwarding mechanisms which split traffic in unequal shares without causing packet disordering have been presented, after which the detailed algorithmic formulation of AMP's load balancing mechanism has been provided in pseudocode.

Following this introduction into AMP, Chapters 4 and 5 will present the extensive performance evaluation of the algorithm which has been carried out using two distinct types of simulation techniques, i.e. *flow-level* and *packet-level* simulation.

# Chapter 4

# AMP Performance Evaluation in a Dedicated Flow-Level Simulator

Starting with a short introduction into the basic principles of flow-level simulation of data networks, this chapter presents the evaluation of Adaptive Multi-Path Routing (AMP) with respect to its statistical performance for a broad variety of artificial topologies and generic traffic patterns, as well as representative investigations into the stability of the algorithm.

## 4.1 Principles of Flow-level Simulation

One of the key phases in the research and development process of all technical systems is the evaluation of their performance and capabilities. Until the advent of fast microcomputers, and especially until their broad availability in the 1970s, performance evaluation in technical sciences was basically limited to two traditional methods:

- *Analytical modeling*: Many systems are based upon well known principles which can be captured analytically, and therefore their performance can be estimated using mathematical methods which have been developed for the corresponding model. The *call model* in circuit switched networks based upon the *Poisson* arrival process, and the *Erlang* formula for call blocking probability may serve as typical successful examples for analytical modeling in the area of telecommunication networks [77].

- *Experimental evaluation*: As the establishment of analytical models may be very difficult or even impossible for many technical systems, experimental evaluation is often the only available method for the estimation of their performance. In the area of telecommunication networks, the main factors which limit the success of analytical modeling are system size, protocol diversity and statefulness, complex input processes based upon different host and user behaviors, etc. Moreover, ex-

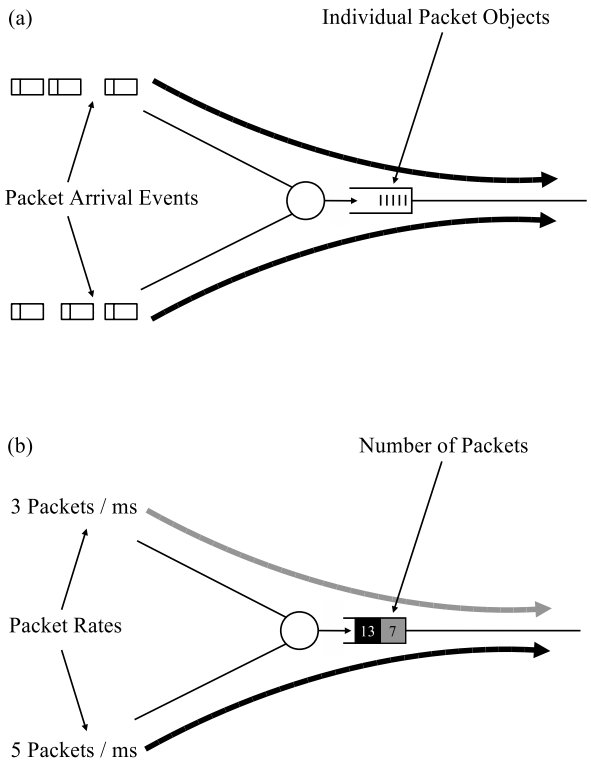



Figure 4.1.1: (a) Packet-level simulation vs. (b) flow-level simulation.

perimental performance evaluation is usually very difficult for most large systems, as many performance aspects cannot be captured in small-scale experiments, and as the setup of large experiments is often associated with substantial efforts and costs.

Fortunately, with the availability of abundant processing power, in recent decades *computer simulation* has emerged as an additional type of scientific investigation methodology. The main idea of computer simulation is to model the investigated system in software, and to perform *virtual experiments* by observing the model's behavior using the simulator. The applicability of results obtained from simulations mostly depends upon the meaningful abstraction of important system properties, as well as upon the correctness of the simulator's implementation, both of which are usually verified by comparing the simulation results to small-scale experiments, or to analytical results which are possibly available for particular components or parts of the system [www-7].

The simulation of communication networks can be performed at different levels of abstraction. As far as packet-switched networks are concerned, the most common (and obvious) choice is to simulate them at the level of packets which are being sent between the communicating nodes (Figure 4.1.1a). This is performed by modeling each packet arrival and packet departure on a link or queue as a separate event in the simulator. The Network Simulator (ns-2) [www-1], Opnet Modeler [www-9], and OMNeT++ [www-10] represent the most widely used simulation environments of this kind.

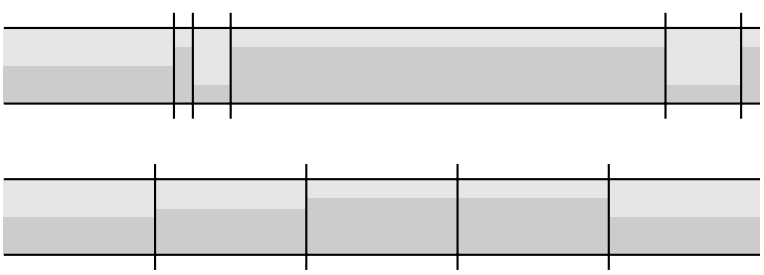



Figure 4.1.2: Difference between continuous-time and discrete-time flow-level simulation.

While packet-level simulation enables highly complex network models yielding very realistic simulation results, its precision comes at the price of great computational complexity. This is especially true for large networks with high link capacities (e.g. networks with more than 30 nodes and Gbit/s-scale links), the performance of which cannot be evaluated in packet-level simulators without making significant abstractions in the simulation scenarios. Although scenarios with lower bandwidths suffice for investigating most issues in contemporary networking, some classes of problems like *routing* and *peer-to-peer protocols* do indeed require a minimum critical network size and/or link capacities in order to display all effects which may be of interest in real systems.

In order to address these issues, the concept of simulating networks at the level of individual flows (i.e. connections) has been developed in a plethora of different variants. Following [83], this section provides a comprehensive survey on some basic principles of flow-based simulation techniques, focusing especially upon Internet routing approaches (see also [84] for further details).

In [85], a *continuous-time* approach to flow-level simulation is presented, called *continuous-time* simulation, in which the events in the simulator are correlated to data rate changes in individual flows. This approach works especially well when simulating systems with stable traffic patterns, as only a small number of events is generated even for long durations of simulated real-system time (i.e. *simulation time*). In other words, as long as the traffic conditions in the network do not change, no new simulation events are generated, such that entire intervals of simulation time can be efficiently *bridged* by the simulator. However, for networks carrying more complex traffic patterns with frequent data rate changes, the efficiency of continuous-time flow-level simulation displays a significant decline, which is not only linearly attributed to the higher number of data rate changes, but rather also to the fact that each data rate change on a congested link potentially also introduces data rate changes to all other flows traversing this link [86, 87]. Therefore, continuous-time flow-level simulation does not represent a feasible solution for many classes of important networking problems.

In order to circumvent the mentioned deficiencies of continuous-time simulation, [88, 89, 90] propose *discrete-time simulation*, which defines a fix *clock-cycle* in which the flows are manipulated (see Figure 4.1.2). More precisely, the flows are not any more

modeled only with the notion of a network wide *rate*, but rather they are represented by the *number of their packets* currently present in different sections of the network (i.e. the links and buffers), such that the flow interactions and networks dynamics are implemented by *shifting* the mentioned *packet quantities* through the network (see Figure 4.1.1b).

A more detailed understanding of the main principles of discrete-time flow-level simulation can be easily achieved by the consecutive introduction of its fundamental techniques: firstly, the modeling of packet streams without the consideration of buffers will be introduced (i.e. for the case of queueless networks), followed by a formal description of the buffering model, as well as a concrete graphical example.

All flows traversing a link obtain capacity shares proportional to their data rates, meaning that if the offered traffic (i.e. the traffic demand) exceeds the link capacity, packets are being discarded proportionally to their rates. Furthermore, the individual links in the simulator are logically subdivided into a number of *time slots* corresponding to the ratio of their propagation delay and the duration of one clock cycle. For example, if some link has a propagation delay of 10ms, and if the clock cycle is set to 1ms, this link will be subdivided into 10 time slots, and the data streams traversing it will shift their packets one slot per clock cycle until they reach their destination. Accordingly, the data rates of the individual flows at their sources are also modeled such that they correspond to the time-slotted nature of the simulation, and thus they are entirely represented by the number of packets generated during each time slot. Depending upon the arrival process of the flows, the number of generated packets can vary in each cycle, or it can remain time invariant, as e.g. in the case of Constant Bit-Rate (CBR) flows.

On a more formal level, the overall offered traffic $\lambda_l[t]$ on link $l$ can be described as the sum of the packet arrival rates $\lambda_l^a[t]$ of all traversing flows $a \in A_l$:

$$\lambda_l[t] = \sum_{a \in A_l} \lambda_l^a[t] \tag{4.1.1}$$

If the offered traffic $\lambda_l[t]$ exceeds the link capacity $c_l$ in the absence of a buffer, the packet loss probability on link $l$, $p_l[t]$, corresponds to

$$p_l[t] = max\{1 - \frac{c_l}{\lambda_l[t]}, 0\} \tag{4.1.2}$$

Packets which do not get discarded arrive at the next hop link after $d_l$ clock-cycles, where $d_l$ denotes the propagation delay of link $l$, and $next_l^a$ denotes the next hop link of $l$ for flow $a$:

$$\lambda^a_{next_l^a}[t + d_l] = \lambda_l^a[t] \cdot (1 - p_l[t]) \tag{4.1.3}$$

Using this system of equations, queueless networks can be simulated very efficiently, with computational complexity directly proportional to the clock cycle frequency. In

this context it is also important to stress that higher clock cycle frequencies automatically result in higher precision of the simulation, thus representing a classical precision/performance tradeoff.

The queueing model in discrete-time flow-level simulation is also very similar to the queueless model described above. For each flow and for each link, the number of packets in the corresponding queue is separately kept track of. In each time step (i.e. clock cycle) of the simulation, the flows send a number of packets on the link proportional to their representation in the queue. If the link capacity is higher than the traffic offered by the queue, the offered traffic of predecessor links is immediately considered for transmission, as this situation basically corresponds to periods of empty queues in the modeled network. Similarly, if the queue becomes full, packets are discarded proportionally to the data rates of the belonging flows.

Starting with the definition of the total offered traffic $\lambda_l[t]$ from Equation (4.1.1), the packet loss probability at queue $q_l$ with capacity $q_l^{max}$ in time-step $t$ corresponds to

$$p_l[t] = max\{1 - \frac{q_l^{max} - q_l[t] + c_l}{\lambda_l[t]}, 0\}, \tag{4.1.4}$$

where $q_l[t]$ denotes the instantaneous queue size on link $l$ in time-step $t$.

The queue evolution, i.e. its fill-state in the next time-step $t_l[t+1]$, is defined by its current fill-state, the offered traffic, the packet loss probability, and the link capacity, yielding,

$$q_l[t+1] = max\{q_l[t] + (1 - p_l[t]) \cdot \lambda_l[t] - c_l, 0\}. \tag{4.1.5}$$

Furthermore, the arrival rate of flow $a$ at the next queue along its path after passing link $l$ is defined by the fill-state of the queue $q_l$ and the packets arriving at link $l$, delayed by the link's propagation delay $d_l$:

$$\lambda^a_{next_l^a}[t + d_l] = \frac{q_l^a[t]}{q_l[t]} \cdot min\{c_l, q_l[t]\} + \frac{\lambda_l^a[t]}{\lambda_l[t]} \cdot min\{max\{c_l - q_l[t], 0\}, \lambda_l[t]\}. \tag{4.1.6}$$

Packets from flow $a$ in queue $q_l$ which are not sent via link $l$ during time-slot $t$, and the newly arrived packets of flow $a$ which fit into queue $q_l$ and cannot be immediately sent via $l$ are added into the queue $q_l$ in time-slot $t+1$:

$$q_l^a[t+1] = q_l^a[t] - \frac{q_l^a[t]}{q_l[t]} \cdot min\{c_l, q_l[t]\} + \lambda_l^a[t] \cdot (1 - p_l[t]) - \frac{\lambda_l^a[t]}{\lambda_l[t]} \cdot min\{max\{c_l - q_l[t], 0\}, \lambda_l[t]\}. \tag{4.1.7}$$

Figure 4.1.3 represents a practical example of flow-level simulation with drop-tail queues, in which the traffic of two different flows coming from a link with higher capacity (10 packets/s) is routed via a link with lower capacity (4 packets/s). During the first

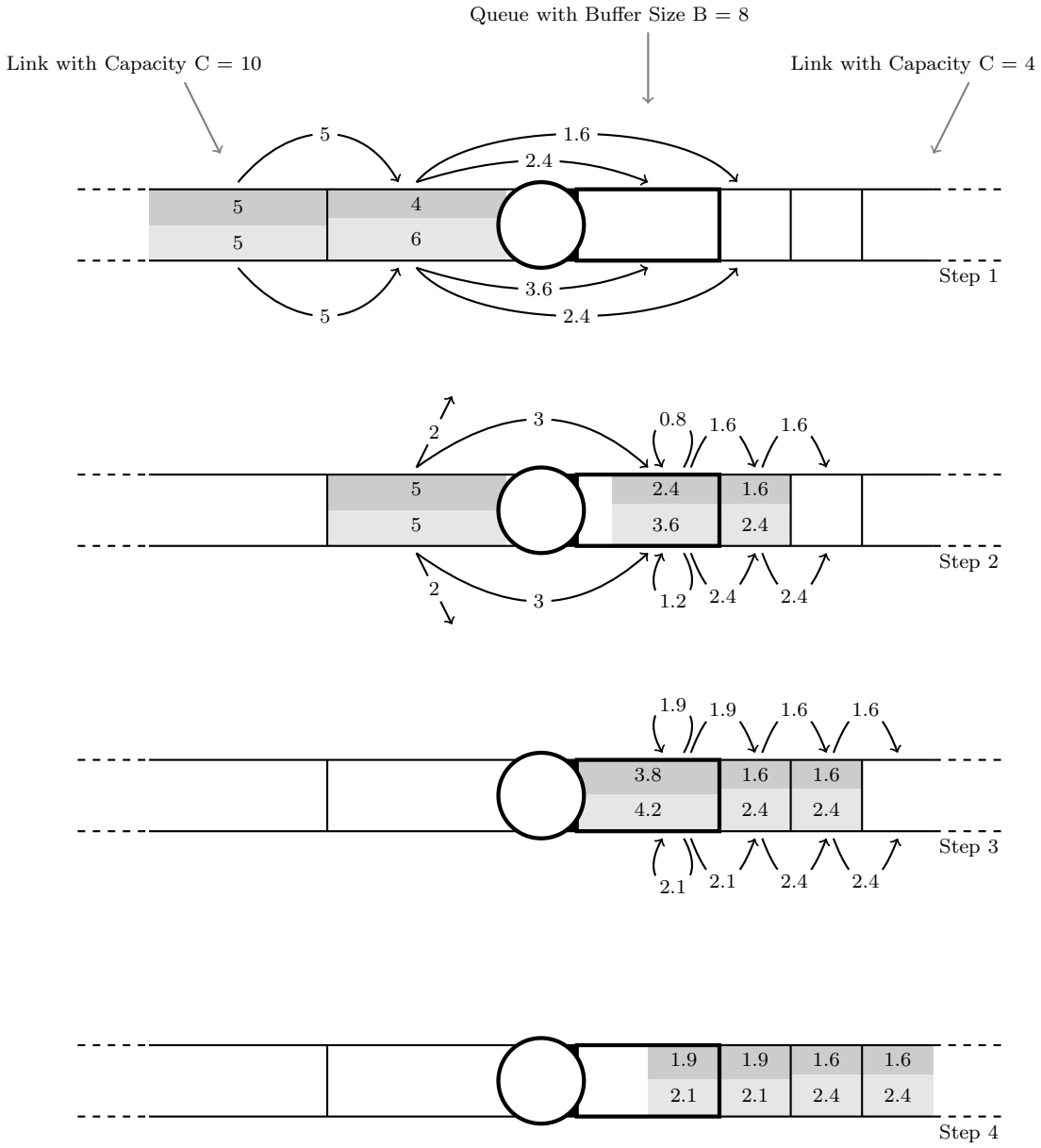



Figure 4.1.3: Example queueing behavior in time-discrete flow-level simulation.

time-step of the simulation 4 packets of one flow, and 6 packets of the flow are routed to the bottleneck link. The arrows denote the packet quantities which will be transferred to individual time-slots during this simulation step – the bottleneck link will be able to accept 4 packets in total, with the capacity shared in proportion to the individual flows' packet quantities in the previous time slot. The rest of the traffic which cannot be accepted by the link is transferred to the queue (again proportionally to individual flows' contributions), resulting in a total unused queue capacity of 2 packets. In the second step of the simulation, the queue is able to transfer 4 packets to the bottleneck link, resulting in a total available space of 6 packets for the newly incoming traffic. However, as the arriving traffic in this time-slot corresponds to a quantity of 10 packets, it is obvious that the queue will have to discard 4 packets, again in proportion to the relative contributions of the individual flows. In steps 3 and 4, the queue size will eventually start to decrease, as there is no newly arriving input traffic at the queue, resulting in a step-wise evacuation of the individual flows' traffic from the system. Note that this example describes only a simplified version of flow-level simulation in which the queues themselves are not subdivided into multiple time-slots in proportion to their size, resulting in deviations from the well-known *first-in first-out* (FIFO) queueing principle. For a more detailed formal description of the flow-level simulation technique which adheres to the FIFO property, please refer to [83].

The following sections will provide an introduction into the problems associated with the performance evaluation of routing algorithms, followed by an overview of AMP simulations carried out using a dedicated simulator based upon the described simulation model.

## 4.2 Setup of Simulation Scenarios for the Performance Evaluation of Adaptive Multi-Path Routing (AMP)

Whereas for many types of networking problems there exist well established schemes for their performance evaluation, the situation is quite different in the area of Internet routing. The main problem in this context is that the behavior of routing algorithms is affected by a variety of different factors, including primarily the topology, the traffic matrix, type of traffic involved, robustness and availability of the network, etc. This variety of parameters naturally yields an inexhaustive space of simulation scenarios, thereby aggravating efforts towards making meaningful statements about the characteristics of the investigated algorithm. In this section, the problems of finding appropriate simulation topologies and traffic matrices for the simulation of IP traffic engineering algorithms are discussed, and the chosen solution for the flow-level simulation of AMP is presented.

When starting to design simulation scenarios for routing algorithms, the issue to

be addressed first is typically the choice of topology. The most natural solution in this context would of course be to simulate the investigated problems on a set of real topologies. However, apart from a few available examples like in [www-11], information about real ISP topologies is hardly available to the general scientific audience.

In order to address this problem, the idea of generating artificial topologies which would closely resemble real networks has emerged relatively early, with the approach by Waxman as one of the most prominent examples (see Section 3.2.1). Apart from this approach, other models which explicitly introduce non-random network structures have been proposed in [34, 91], following an inspection of real networks which showed that instead of being random, they display obvious hierarchical features. Furthermore, this approach argued that topology generators should reflect the applied network design principles, which are usually based on certain connectivity and redundancy requirements that are not reflected in random topologies (cf. [92]).

Another type of topology generators was proposed in [93, 94], based upon the discovery that both the graphs of Autonomous System (AS) level and router level topologies in the Internet display power law relationships in their connectivity. Following these observations, the BRITE topology generator has been chosen for the purpose of flow-level simulations of the Adaptive Multi-Path algorithm (AMP) [95, www-12]. More specifically, the power-law model of artificial topologies presented in [96] is applied, which is based upon the so-called *preferential attachment* principle saying that the growth of the network is realized by the sequential addition of new nodes, where each newly added node connects to some existing node preferentially, such that it is more likely to attach to a node that already has many connections.

As far as traffic generation is concerned, there hardly exist any concrete directions, apart from some pioneering attempts like e.g. in [97], where some initial observations are provided based upon measurements in the Sprint IP backbone network, without introducing a comprehensive and mature methodology for traffic matrix generation. This fact is especially troublesome in the context of IP traffic engineering research, as a carefully designed traffic matrix represents a fundamental prerequisite for the investigated mechanisms to demonstrate their performance.

In order to circumvent this problem, researchers have reverted to some very simple traffic generation heuristics for their performance evaluation tasks. The *gravity model* represents a widely used heuristic of this kind. The main principle of this approach is to associate a number of users to each node in a topology, and to generate traffic such that each node sends traffic at a rate proportional to its user numbers, with the traffic fan-out (i.e. the distribution towards other destinations in the network) achieved such that each node distributes its outgoing traffic in proportion to the other nodes' relative user numbers (cf. [13]).

Although the gravity model is very straightforward, there are several open issues associated to its practical application: firstly, in each given topology the *relative sizes*

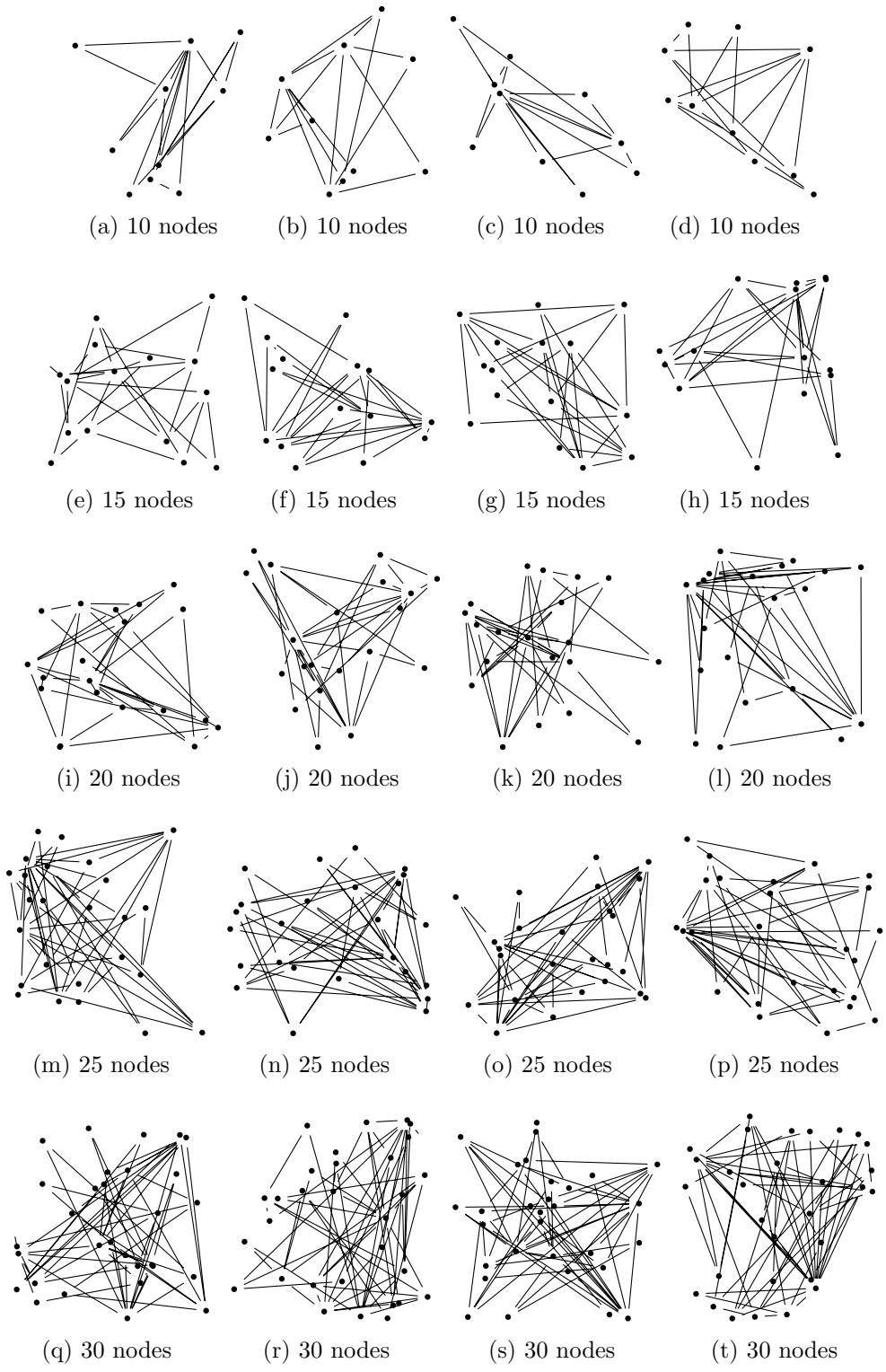
(i.e. the relative user numbers) of the individual nodes must be determined. As far as real topologies are concerned, the most logical and most commonly applied solution is to assign user numbers proportional to the relative sizes of the cities or metropolitan areas corresponding to the individual nodes. Whereas this solution represents an attractive option in the case of real topologies, the situation is much more difficult in the case of artificial topologies, as the user numbers do not necessarily need to be strongly correlated to physical topology features. Secondly, even if a set of plausible node sizes is chosen, it is difficult to scale the absolute traffic intensity in an intuitive manner such that a realistic network wide load pattern is achieved.

For the purpose of AMP performance evaluation in a flow-level simulation environment, the described gravity model has been used, with random population numbers from a set of predefined discrete values assigned to each node in the artificial topologies. As far as traffic generation is concerned, a search algorithm has been developed which scales the absolute traffic intensity according to a set of user defined parameters. In the presented simulation scenarios, the user input is determined by two parameters, $\alpha$ and $\beta$, where $\alpha$ represents a percentage of the overall number of links which display link utilizations of at least $\beta$. In a concrete example, the user may define that at least $\alpha = 5\%$ of links in the network should display link utilizations of at least $\beta = 120\%$, where *link utilization* represents a simple ratio of offered load and link capacity. The described search mechanism is implemented as logarithmic search, in which simulations of short duration are performed consecutively, until the desired level of constant bit-rate (CBR) traffic intensities at the sources are found.

## 4.3 Simulation Scenarios and Results

Whenever novel traffic engineering strategies are to be investigated using simulations, a multitude of individual scenarios featuring different topologies and load levels is required in order to obtain an insight into their statistical performance. In other words, single scenarios always bear the risk of displaying atypical behavior of the evaluated scheme, and thus may lead to wrong conclusions. However, at the same time it is important to mention that the mere statistical mean performance also does not suffice in its own, but rather the individual simulation runs also must be microscopically investigated in order to check the investigated scheme for anomalous behavior, like e.g. oscillations in the case of IP routing.

In this section, the statistical performance of AMP is investigated by means of a large number of different simulation scenarios. Accordingly, 5 different topology sizes are used (10, 15, 20, 25, and 30 nodes), and 20 individual topologies of each size are generated using the BRITE topology generator [95, www-12], according to the model by Barabási and Albert presented in [96]. Figure 4.3.1 depicts representative graphical samples of Barabási and Albert topologies of the different sizes used in the performed



(a) 10 nodes (b) 10 nodes (c) 10 nodes (d) 10 nodes

(e) 15 nodes (f) 15 nodes (g) 15 nodes (h) 15 nodes

(i) 20 nodes (j) 20 nodes (k) 20 nodes (l) 20 nodes

(m) 25 nodes (n) 25 nodes (o) 25 nodes (p) 25 nodes

(q) 30 nodes (r) 30 nodes (s) 30 nodes (t) 30 nodes

Figure 4.3.1: Artificial Barabási-Albert topologies with 10 to 30 nodes.

experiments.

The resulting total of 100 individual topologies has further been simulated with five different load levels for each topology, using factors $\alpha$=5%, and $\beta \in \{105\%, 110\%, 115\%, 120\%, 125\%\}$. The applied Constant Bit-Rate (CBR) traffic has been generated according to the gravity model described in the previous section, using two distinct population sizes with a relative ratio of 2, with the smaller size assigned to 70% of the nodes, and the greater (i.e. double) size assigned to the remaining 30%.

As far as the link capacities and link weights are concerned, two distinct network configurations have been investigated: in the first strategy, all link weights and all link capacities were set to the same value, effectively resulting in the *minimum hop* routing strategy (*min-hop*). In the second strategy, two link capacities with a relative ratio of 4 were used, with 20% of the links assigned the higher capacity, and 80% the lower capacity, with link weights set according to the Cisco heuristic, meaning that they are inversely proportional to the corresponding links' capacities. For the purpose of these simulations, traffic is routed only along the shortest paths without relying upon the *relaxed best path criterion*, in order not to overemphasize the possibly extensive path diversity phenomena in the investigated graphs, and thereby to focus primarily upon the load balancing performance of AMP.

The described combinations of investigated topologies, load levels, and routing strategies result in a total of 1000 simulations, providing a reasonable number for making initial observations concerning the statistical performance of AMP. Following the simulations of OMP involving Constant Bit-Rate (CBR) traffic presented in [76], the relative change of the packet loss rate is chosen as the main performance metric. In the context of statistical performance evaluation this metric is particularly beneficial, as it alleviates the need of presenting absolute performance figures (e.g. in terms of flow bandwidth, etc.), thus making the performance of the algorithm in different topologies easily comparable.

In all simulations, the snapshots of two different periods of simulation time have been taken: firstly, the initial phase, in which the traffic is uniformly distributed among all multiple shortest paths towards each destination in each node, and secondly the final phase in which the algorithm has converged such that the fix point in network wide traffic distribution has been reached, corresponding to the time of 100 AMP control periods after the start of the simulation. Figures 4.3.2, 4.3.3, 4.3.4, 4.3.5, and 4.3.6 show the performance of AMP for $\alpha$=5%, and for factors $\beta$ of 105%, 110%, 115%, 120%, and 125%, respectively. The figures demonstrate significant performance improvements of AMP, and show that the *min-hop strategy* consistently outperforms the strategy applying the *Cisco link weight heuristic*. Furthermore, the performance improvements of AMP are larger for lower levels of overload, which is expected as in this case the network holds greater amounts of residual capacity, reaching impressive packet loss rate reductions of almost 50% with the min-hop strategy. Accordingly, with growing load levels (i.e. $\beta$), the network gradually becomes consistently congested, reducing the traffic

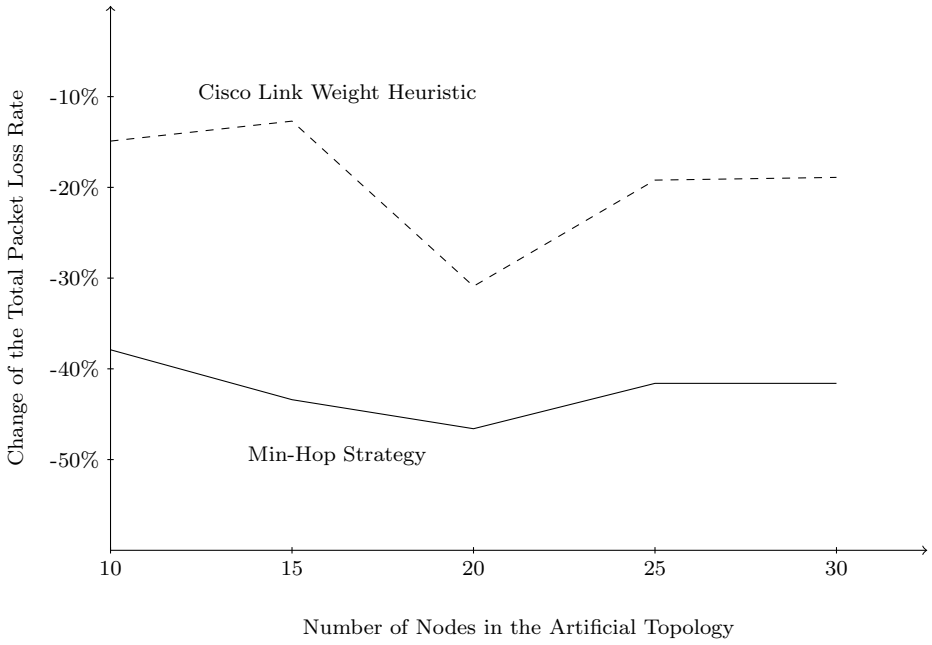



Figure 4.3.2: AMP in artificial topologies running CBR traffic with $\alpha$=5% and $\beta$=105%.

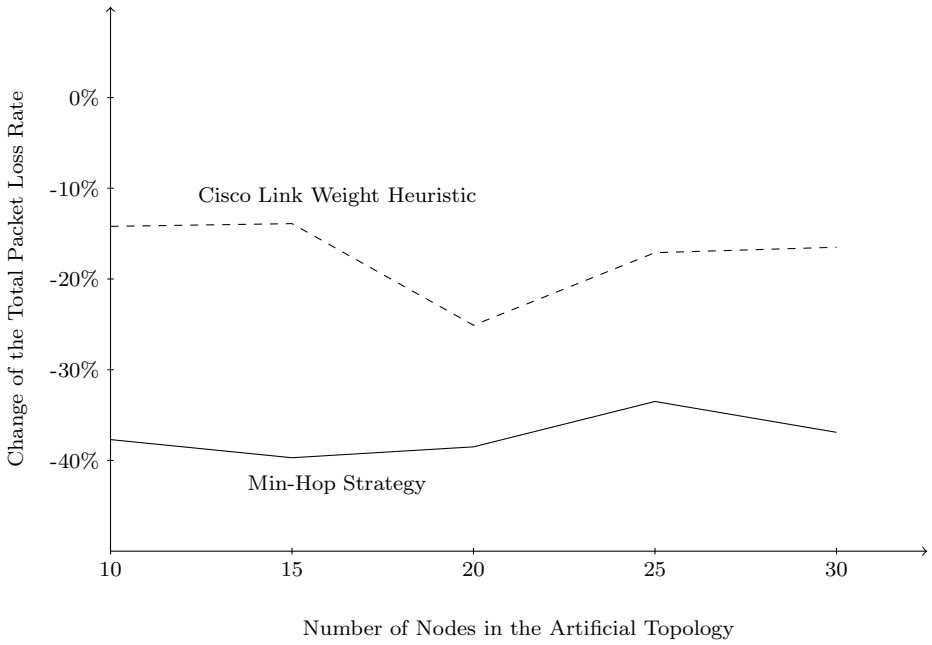



Figure 4.3.3: AMP in artificial topologies running CBR traffic with $\alpha$=5% and $\beta$=110%.

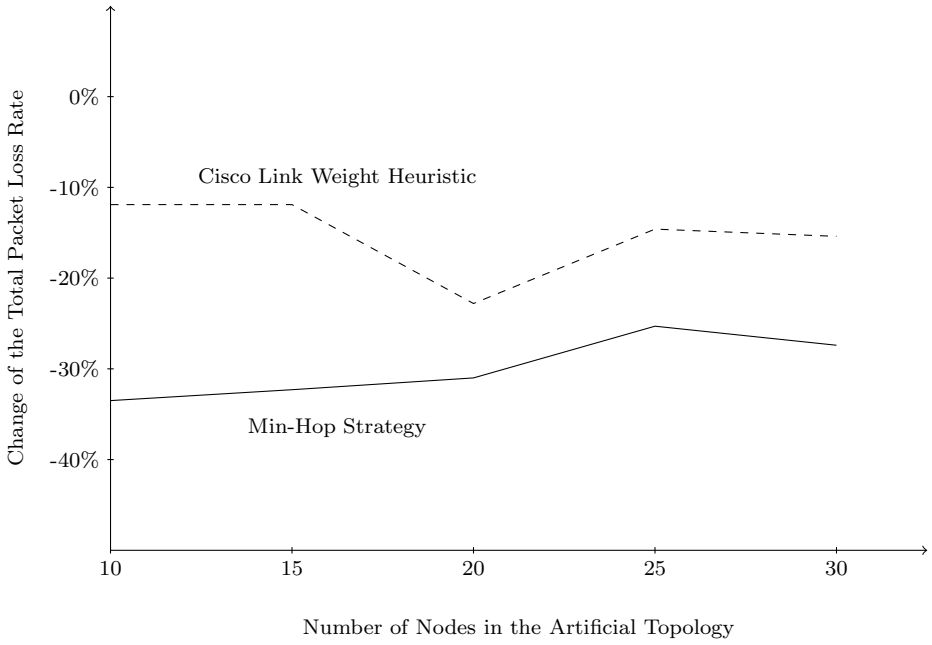



Figure 4.3.4: AMP in artificial topologies running CBR traffic with $\alpha$=5% and $\beta$=115%.

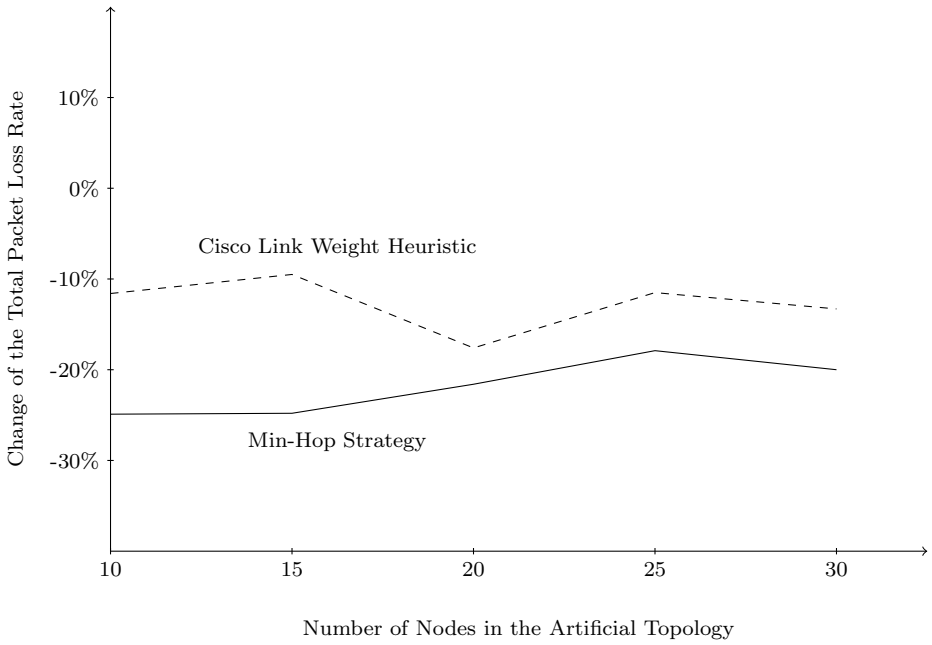



Figure 4.3.5: AMP in artificial topologies running CBR traffic with $\alpha$=5% and $\beta$=120%.

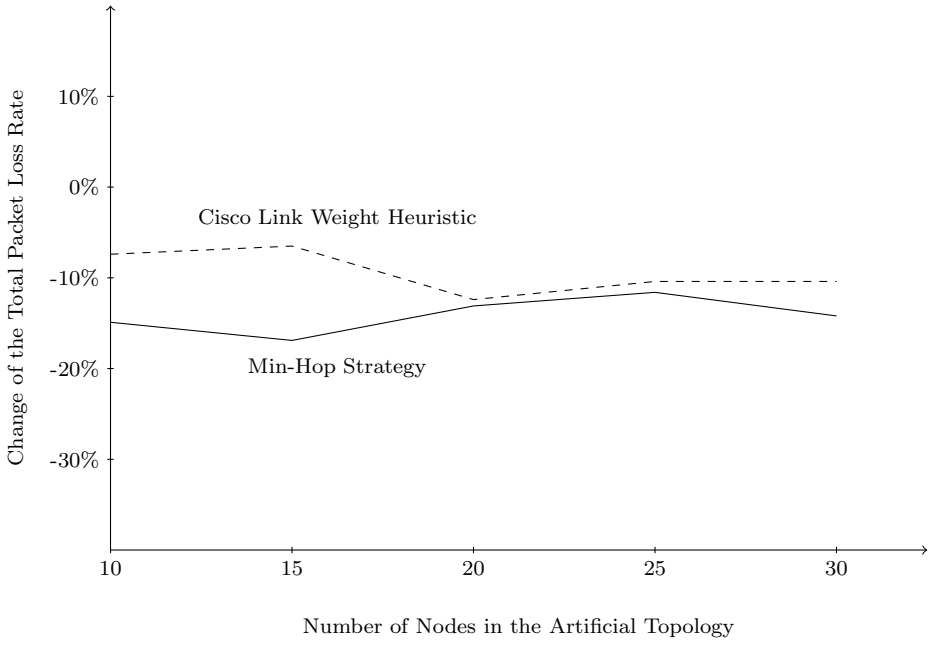



Figure 4.3.6: AMP in artificial topologies running CBR traffic with $\alpha$=5% and $\beta$=125%.

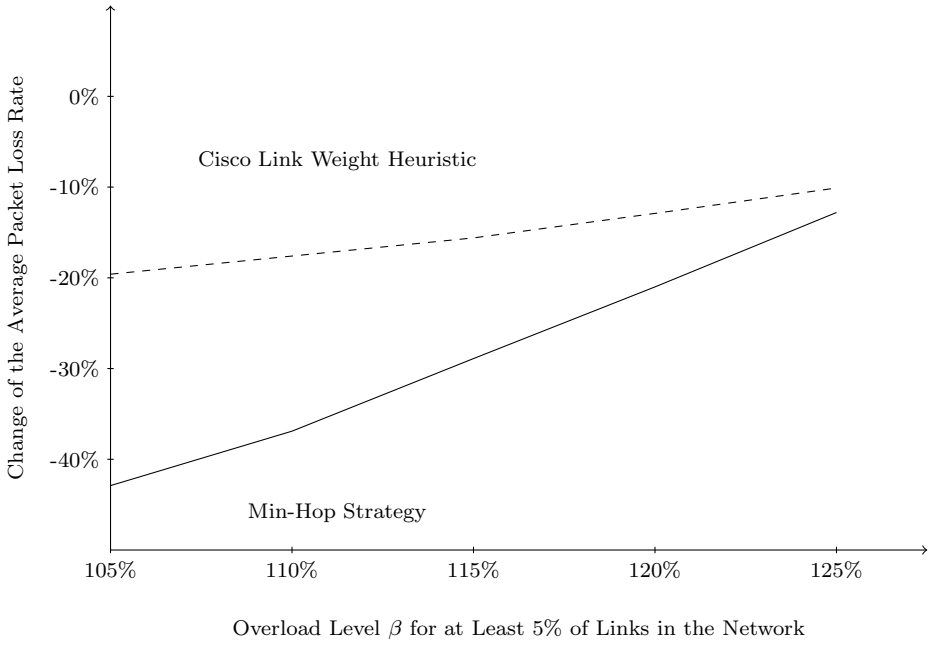



Figure 4.3.7: Average AMP performance improvement for artificial topologies with different load levels.

engineering potential of any routing strategy. This trend is particularly well observable in Figure 4.3.7, in which the average performance improvements for different load levels are summarized in a single graph.

## 4.4 Stability and Convergence Properties

While the macroscopic evaluation presented in the previous section is justified when focusing upon the global performance of an algorithm, it usually fails to deliver hard statements about its effects upon individual link loads and traffic streams in the network. This aspect is particularly important when it comes to stability, as the interesting and possibly most important effects occurring at the level of microflows (i.e. a quadruple of source and destination IP addresses and their respective ports) may easily be masked by overall system statistics (like e.g. link utilization).

Therefore, this section presents microscopic investigations into the operation of AMP, especially focusing upon the load balancing mechanism itself. Two aspects are of particular interest in this context: firstly, the algorithm's convergence towards a fix point in traffic distribution under stable traffic conditions (i.e. in the presence of a time-invariant traffic matrix) and secondly, AMP's response to an unstable traffic pattern which is of utmost importance for overall system stability, which is otherwise notoriously difficult to prove analytically for mechanisms based upon feedback from the network (cf. [98]).

The investigations have been performed in the form of a series of experiments using the described flow-level simulation environment. Besides its efficiency in terms of computation time for networks with high bandwidth links, the main advantage of flow-level simulations in the context of this work lies in the complete absence of synchronization and other timing effects which may occur in packet-level simulations. The well-known *lock-out effect* which may easily occur in packet-level simulations with constant bit-rate traffic and drop-tail queues is a typical example of such normally misleading effects [99].

For all examples presented in this section, non-elastic constant bit-rate and on-off traffic has been used in order to avoid any interferences of multiple control mechanisms, which may theoretically occur in networks simultaneously running adaptive routing algorithms and transporting elastic traffic like TCP. The presented experiments thus isolate the effects of AMP, and demonstrate the load balancing algorithm's response to different traffic patterns.

In the first experiment, AMP's convergence behavior in a minimalistic scenario is examined, in which constant bit-rate traffic of 20 packets/s is sent between the pair of nodes $\{R_1, R_4\}$ connected via two paths of bottleneck capacities 5 packets/s ($\overline{R_1R_2R_4}$) and 15 packets/s ($\overline{R_1R_3R_4}$) (see Figure 4.4.1). During the initialization phase of the algorithm, AMP always distributes the traffic uniformly among all next hops available for a particular destination. Accordingly, 50% of the load (i.e. 10 Packets/s) will initially be sent both on the upper and the lower path. Initially, this will inevitably result in

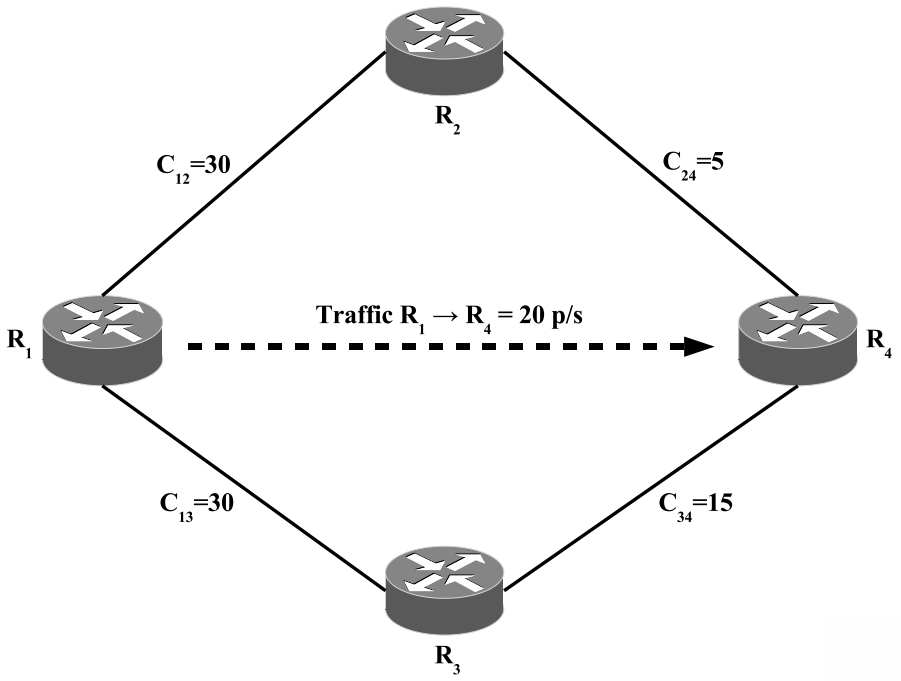



Figure 4.4.1: Setup of Stability Experiment 1 demonstrating AMP's convergence towards a stable traffic distribution.

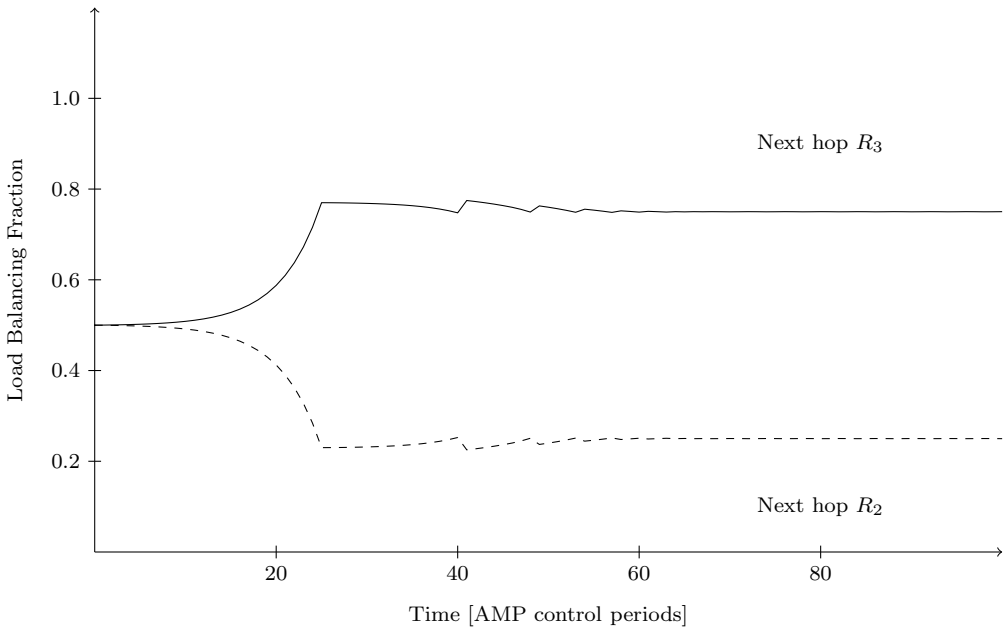



Figure 4.4.2: Load distribution in $R_1$ in Experiment 1.

congestion and packet loss on the link $\overline{R_2R_4}$, as it only has a capacity of 5 packets/s and thus can accommodate only 25% of the load. Therefore, node $R_2$ will signal this overload situation back to node $R_1$, which will then try to achieve equilibrium by redistributing the load between the two paths. Please note that for the purpose of performing these experiments, the choice of packet forwarding strategies (as described in Section 3.2.4) will not have any influence upon the results, as only constant bit-rate traffic is used and as the focus is set exclusively upon the stability aspects of the algorithm. Therefore, the abstraction is legitimately made of regarding the constant bit-rate traffic between the core nodes of the network as an aggregate of a high number of individual microflows, such that the traffic is distributed at an arbitrary fine granularity in these simulations.

Figure 4.4.2 displays the traffic fractions in node $R_1$ with respect to the two available paths towards node $R_4$, on which the load moves towards the optimal distribution of 25% and 75%, respectively, in an efficient way, assuring optimal usage of the available network capacity. Even more importantly, this figure provides a means for the visual inspection of the dynamics of the load balancing algorithm, which achieves convergence rapidly (within roughly 60 load balancing steps in this example) and with a minimal number of overshoots, thus enabling high overall efficiency of load distribution. Each load balancing step corresponds to one control period of the AMP algorithm, which is usually chosen in the order of magnitude of seconds (cf. Section 3.2.5). The concrete value of the control period may be set differently depending upon the predominant type of traffic in the network: if only non-elastic traffic is carried, the control period may assume very small values, whereas this would not be advisable in the case of elastic traffic like TCP, as too small values of the control period might lead to interference between the congestion control and the load balancing mechanism.

In the second experiment, the algorithm's response to different time-variant traffic patterns is tested. A classical approach when testing control systems is to introduce an on-off input pattern in order to try and provoke oscillations. The precise setup of the experiment is shown in Figure 4.4.3: a constant bit-rate traffic stream is set up between $R_1$ and $R_6$, and additionally a periodic on-off traffic stream is established between $R_7$ and $R_8$.

All links in this example have capacities of 15 packets/s, whereas both the CBR and the on-off traffic streams will have peak rates of 15 packets/s. These two streams will interfere on link $\overline{R_2R_4}$, whereby the measured load on the link itself will vary at the frequency of the on-off traffic pattern between the nodes $R_7$ and $R_8$ (see Figure 4.4.4). The sudden variations in traffic load will provoke ongoing changes in the direction of load shifting, and taking the mentioned values into account it is obvious that the link $\overline{R_2R_4}$ becomes a bottleneck, as the peak traffic demand will by far exceed its capacity. In this context it is interesting to monitor the fractions of traffic between $R_1$ and $R_6$ sent via $R_2$ and $R_3$, respectively (see Figure 4.4.5).

During the *on* periods of the on-off traffic stream, AMP will react by sending back-

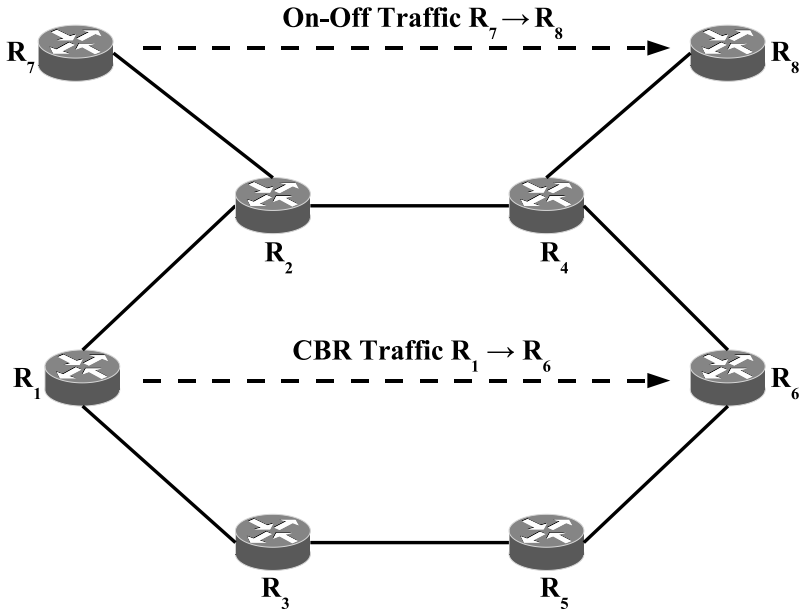



Figure 4.4.3: Setup of Stability Experiment 2 involving CBR and on-off traffic.

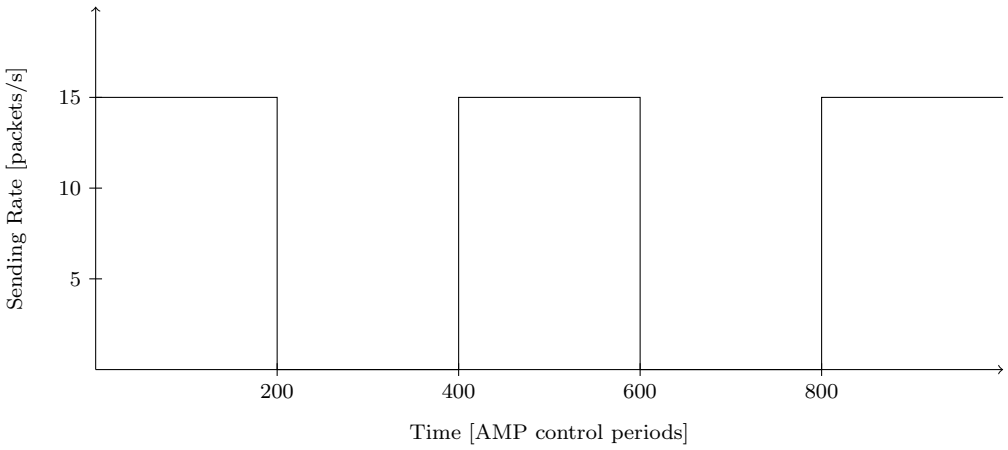



Figure 4.4.4: On-off traffic from $R_7$ to $R_8$ in Experiment 2.

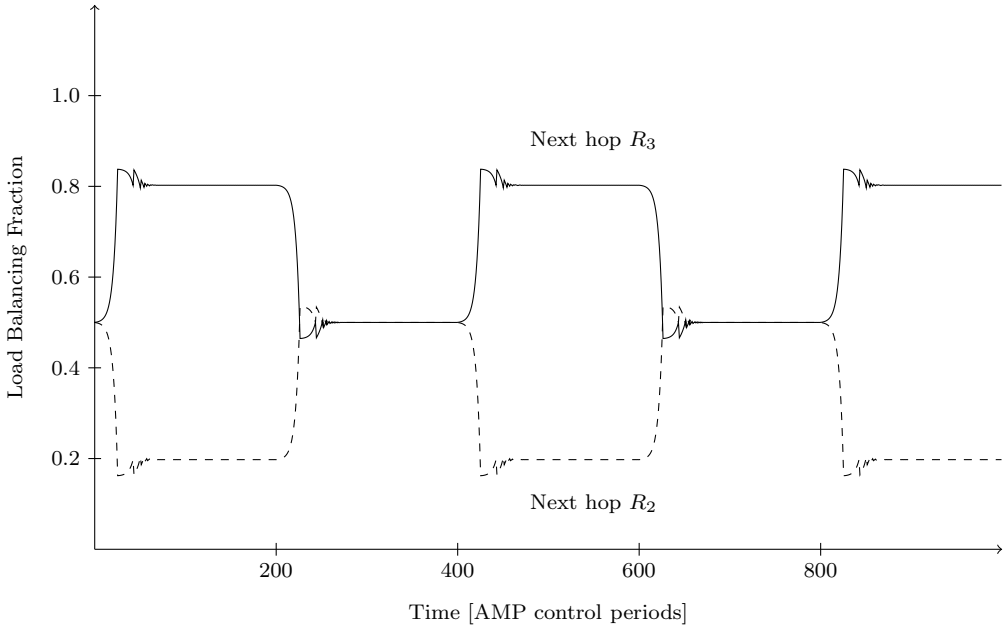



Figure 4.4.5: Load distribution in $R_1$ in Experiment 2.

pressure messages which will inform the node $R_1$ about the overload on the link $\overline{R_2R_4}$. Node $R_1$ will then react by increasing the portion of traffic sent via link $\overline{R_1R_3}$, until the *off* period of the on-off stream starts. At this moment in time, AMP will reverse the direction of load shifting, and again try to route more traffic via node $R_2$. Figure 4.4.5 demonstrates that such an instable traffic pattern will have no influence upon the stability of AMP operation, as the load fractions will remain stable at the time scale of the AMP control period. Furthermore, it is important to stress in this context that AMP has proved to operate oscillation free even for very small of very large ratios of those values.

In the third experiment, AMP's stability is tested in the presence of two on-off traffic streams belonging to different multi-path structures which share a common bottleneck link ($\overline{R_3R_5}$ in Figure 4.4.6). The two streams $\overline{R_1R_6}$ and $\overline{R_7R_8}$ have peak rates of 15 packets/s, summing up to a total peak rate of 30 packets/s, which precisely corresponds to the *min cut – max flow* theoretical network capacity as the links $\overline{R_2R_4}$, $\overline{R_3R_5}$, and $\overline{R_9R_{10}}$ have capacities of 10 packets/s each. In order to impose additional strain upon AMP in terms of unstable traffic patterns, different on-off frequencies are chosen for the two streams, which should generate a periodic, but still relatively uneven pattern of load shifting direction changes (see Figure 4.4.7).

As demonstrated in Figures 4.4.8 and 4.4.9, which display the dynamics of the load balancing fractions in the routers $R_1$ and $R_7$, AMP responds quickly to the variable input traffic, and shifts load efficiently onto the alternative paths. Most importantly, the frequent changes of load shifting directions do not cause any oscillations of the load balancing algorithm, which reliably ensures stable operation of the entire network.

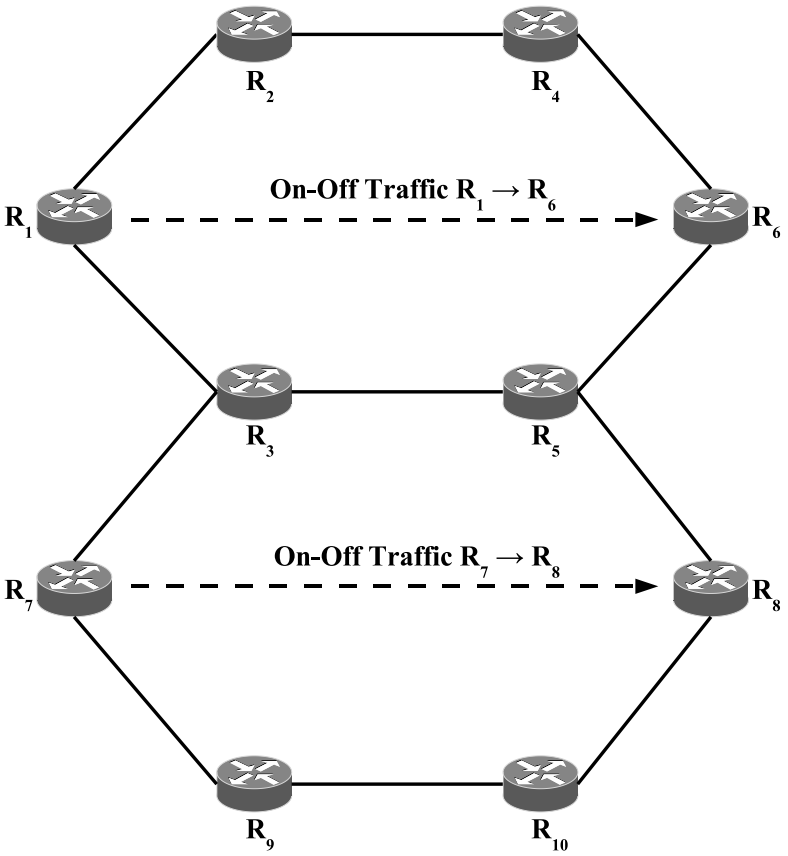



Figure 4.4.6: Setup of Stability Experiment 3 featuring two interferring on-off traffic streams.

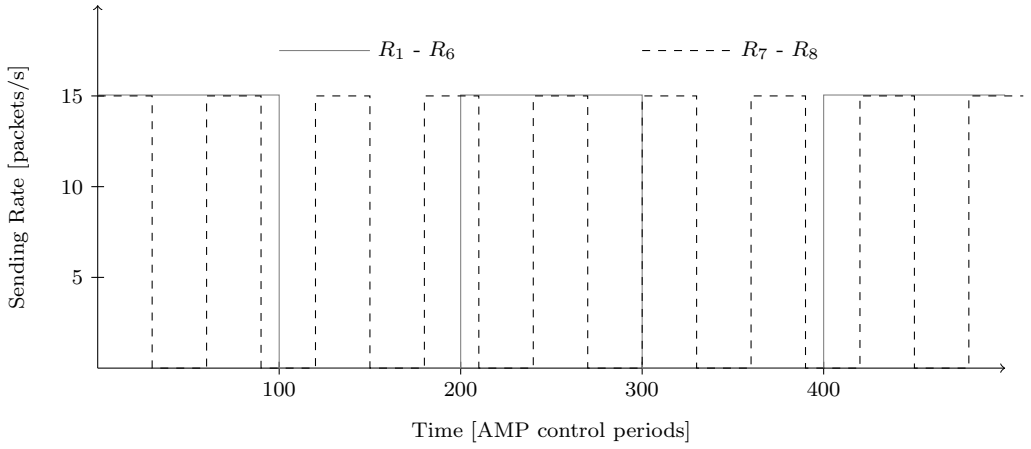



Figure 4.4.7: Two on-off traffic sources in Experiment 3.

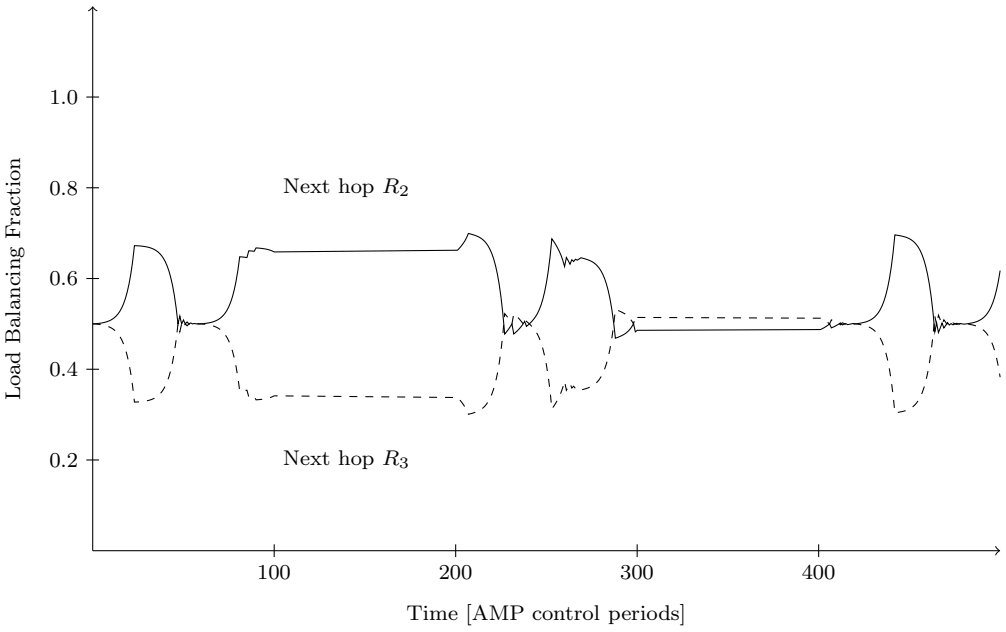



Figure 4.4.8: Load distribution in $R_1$ in Experiment 3.

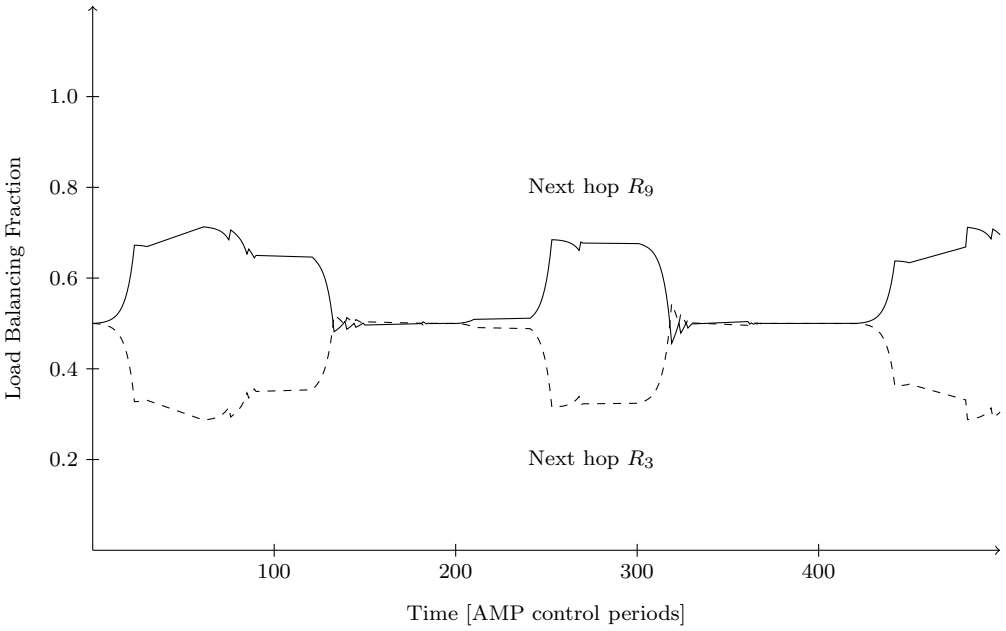



Figure 4.4.9: Load distribution in $R_7$ in Experiment 3.

## 4.5 Concluding Remarks

Following Chapter 3, in which the Adaptive Multi-Path algorithm (AMP) has been presented in detail, this chapter has introduced the two main methodologies for the simulation of communication networks, after which it has dealt with the flow-level case in detail, including a comprehensive description of the corresponding AMP simulations and the obtained results. The main motivation for performaing those simulations is to get an insight into the statistical performance of AMP, which represents a significant challenge in the performance evaluation of all traffic engineering algorithms due to the inexhaustive number of parameters which can be varied in each iteration of the simulation process.

Accordingly, in this chapter AMP has been compared to standard IP routing for a large number of combinations of different artificial topologies, link weight settings, traffic distributions, and absolute traffic intensities, demonstrating the statistical performance advantage of AMP in situations of network overload, as well as its stability in the context of instable traffic patterns. The simulations have been performed using a dedicated flow-level simulation environment, which is particularly well suited for exploring such large parameter spaces due to its low computational complexity. Furthermore, Constant Bit-Rate (CBR) traffic has been chosen for the purpose of these simulations in order to avoid possible interferences between congestion control mechanisms like TCP and AMP load balancing, enabling exclusive attribution of the observed changes in traffic distribution and network utilization to the traffic engineering mechanism itself.

However, as the majority of traffic in today's Internet employs TCP at the transport layer (like e.g. Web traffic and the majority of peer-to-peer traffic), it is of utmost importance to investigate AMP behavior and performance also in this context. This issue will be addressed in the next chapter, in which the more traditional packet-level simulation approach will be employed as the tool of choice for the performance evaluation of the investigated scenarios.

# Chapter 5

# AMP Performance Evaluation in the Packet-Level *ns-2* Simulator

## 5.1 From Flow-Level to Packet-Level Simulation of AMP

Whereas the flow-level simulations presented in Chapter 4 have represented a meaningful choice for providing fundamental insights into AMP performance with non-elastic traffic patterns, they of course cannot safely predict the algorithm's behavior in the presence of more complex traffic patterns like e.g. TCP, thus necessitating further performance evaluation efforts.

Currently, the theoretical fundamentals for modeling elastic traffic like TCP, and especially its main applications like e.g. Web traffic, are not completely mature for flow-level simulations environments. The main problem lies in the fact that TCP traffic is based upon a *per host* state machine [73] which adapts the sending rate to the observed network conditions, and as such it is very difficult to model TCP at the level of traffic aggregates between individual {source, destination} pairs, which on the other hand is a major prerequisite for the speed advantage of the flow-level approach. However, different TCP microflows between a single {source, destination} pair can nevertheless be modeled as a single aggregate, under the assumption that all microflows always experience the same network conditions, i.e. the same round trip delay and packet loss probability on the path.

Following this idea, [100] presents a flow-level simulation model which assumes that all TCP connections are in *congestion avoidance* mode, and that their duration is infinite, corresponding to a bulk File Transfer Protocol (FTP) connection. The evolution of the average TCP congestion window (which divided by the round trip delay yields the sending rate) is in this model defined by a simple differential equation based upon the instantaneous round trip delay and the packet loss probability on the path:

$$\frac{dW(t)}{dt} = \frac{1}{RTT(t)} - \frac{W(t)}{2} \cdot P(t), \tag{5.1.1}$$

where $W$ is the congestion window, $RTT$ the round trip delay, and $P$ the packet loss probability on the path. The first part of the equation models the increase of the congestion window after a single round trip delay, whereas the second part of the equation models the window's multiplicative decrease in the presence of packet losses. The $RTT$ value and the total number of lost packets is being calculated by the simulator in each time step $t$, whereas the path loss probability $P$ corresponds to the number of lost packets of an aggregate divided by the number of active connections within that aggregate.

A more sophisticated TCP model is introduced in [101], which additionally enables the simulation of TCP connections in the *slow start* mode, and TCP connections of finite length. Both new features of this approach represent a prerequisite for modeling more complex traffic types than just simple bulk FTP transfers, like e.g. Web traffic, where most connections have got very short durations, and therefore largely remain in the *slow start* mode during their entire lifecycle. However, the computational complexity of this approach is also by far greater than that of alternative models, as each aggregate is associated a set of variables which exactly track the distribution of the *slow start* and *congestion avoidance* windows in every step of the simulation. Unfortunately, this added overhead largely diminishes the speed advantage of the flow-level approach, making it less attractive for the simulation of large-scale networking scenarios.

Therefore, for the purpose of performance evaluation of AMP in the presence of realistic traffic models, like e.g. Web traffic, packet-level simulation still represents the primary choice, especially as simulation results prove to be very close to real values for a broad spectrum of scenarios, mainly due to the very mature implementation of the TCP stack in widely used packet-level simulators like e.g. ns-2 (cf. [102]). Accordingly, the *ns-2* framework has been chosen for the simulation of AMP in combination with Web traffic, and the next sections provide an overview of important implementation highlights, simulation scenarios, and results.

## 5.2 AMP Implementation Highlights in the *ns-2* Simulator

The IP routing modules available in the *ns-2* simulator in general do not support traffic sensitive routing, meaning that the implementation of a dynamic routing scheme like AMP requires extensive software design and programming efforts. The implementation of AMP is based upon the *link state routing* module, which provides the dissemination of routing information using the flooding mechanism, and the construction of routing tables

in each node using Dijkstra's shortest path algorithm, at the same time considering the weights associated to individual links. For the purpose of AMP performance evaluation, the full functionality of the algorithm has been implemented in the *ns-2* simulator, including the calculation of paths according to the relaxed best path criterion, the calculation of AMP-specific traffic-sensitive link metrics, AMP signaling based upon the backpressure mechanism, packet forwarding, and load balancing (cf. Chapter 3).

As far as the 16-bit Cyclic Redundancy Check (CRC-16) scheme for packet forwarding is concerned, there is no fast software solution for *ns-2*. Moreover, *ns-2* routing is not at all based upon IP addresses, but instead simple integer node identifiers are used, and therefore a different solution had to be used in order to correctly capture AMP's behavior within the simulator. Based upon the realistic assumption that CRC-16 is very effective in spreading microflows evenly in the 16-bit solution space [72], each microflow in the simulation is assigned a randomly generated integer ID upon its start. Consequently, all packet forwarding decisions in the *ns-2* nodes are based upon comparing this random ID with appropriately set tresholds, thus significantly reducing the computational complexity, and making the simulation of large backbone networks with realistic traffic feasible.

## 5.3 Performance Evaluation of AMP in the AT&T-US Backbone Network

Using *ns-2* packet-level implementation of AMP described in the previous section, the performance of AMP is compared to standard IP shortest path routing (SPR) and equal-cost multi-path routing (ECMP) in the AT&T-US backbone topology of 27 nodes and 47 links [103], as sketched in Figure 5.3.1. In this topology, the majority of network connections are OC-48 links, corresponding to a capacity of 2.4 Gbit/s, whereas the five bold lines in Figure 5.3.1 represent OC-192 links, corresponding to a capacity of 9.6 Gbit/s. As far as the link costs are concerned, they were set inversely proportional to the link capacities, resulting in two integer values with a relative ratio of 4.

For traffic generation, the SURGE Web traffic model is used, as described in [104]. In this model, each Web user transmits a flow, stays idle for a *user think time*, and then transmits another flow, where both the flow sizes and the think times are Pareto distributed. In the performed simulations, each node hosts a virtual number of Web users proportional to the approximate population size of the corresponding US city where the node is located in reality, and different load levels were produced by linearly scaling the percentage of *active* users at the individual nodes. Similarly to [41], the spatial distribution (i.e. the fan out) of HTTP requests in each node is set proportionally to the relative population sizes of the other nodes (i.e. cities) in the topology, resulting in a traffic distribution according to the *gravity model*, analogous to the one used in

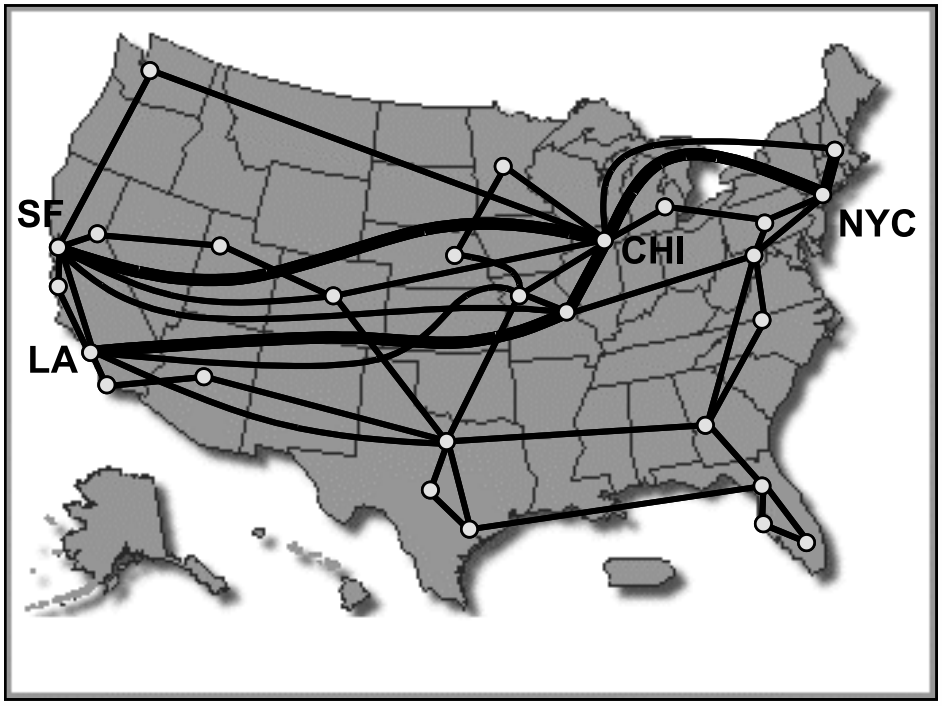



Figure 5.3.1: AT&T-US network topology.

flow-level simulations from Chapter 4.

As far as the simulations' level of realism is concerned, the only relevant deviation from the real network had to be made concerning the link capacities, as too large link capacities in the simulation would result in a too high number of events per duration of simulated time, making the simulation of large networks impossible within a realistic time frame. Therefore, in the presented simulations the link capacities have been reduced to 15 Mbit/s and 60 Mbit/s. However, even with such link capacity reductions, the simulation of large networks like AT&T-US still requires enormous computational resources. E.g., even the simulation results presented in Figures 5.3.2 to 5.3.4 have required several weeks of processor time on ten state of the art PCs, which clearly demonstrates the limitations of packet-level simulation for large-scale networking problems featuring realistic traffic patterns.

Figures 5.3.2 to 5.3.4 show the main simulation results, comparing total TCP goodput, average Web page response time, and the average coefficient of variation (CoV) of the observed link utilizations for the three investigated routing strategies, i.e. shortest path routing (SPR), equal-cost multi-path routing (ECMP), and adaptive multi-path routing (AMP). Note that in each case the initial simulation phase (before reaching equilibrium) has been dropped, and the remaining results correspond to a simulation time of 16000 seconds each. The various load situations are characterized by the total number of users in the system which is displayed at the x-axes.

Figures 5.3.2 and 5.3.3 represent the user perspective. As far as TCP goodput is concerned, AMP clearly outperforms the other two routing strategies (Figure 5.3.2) by

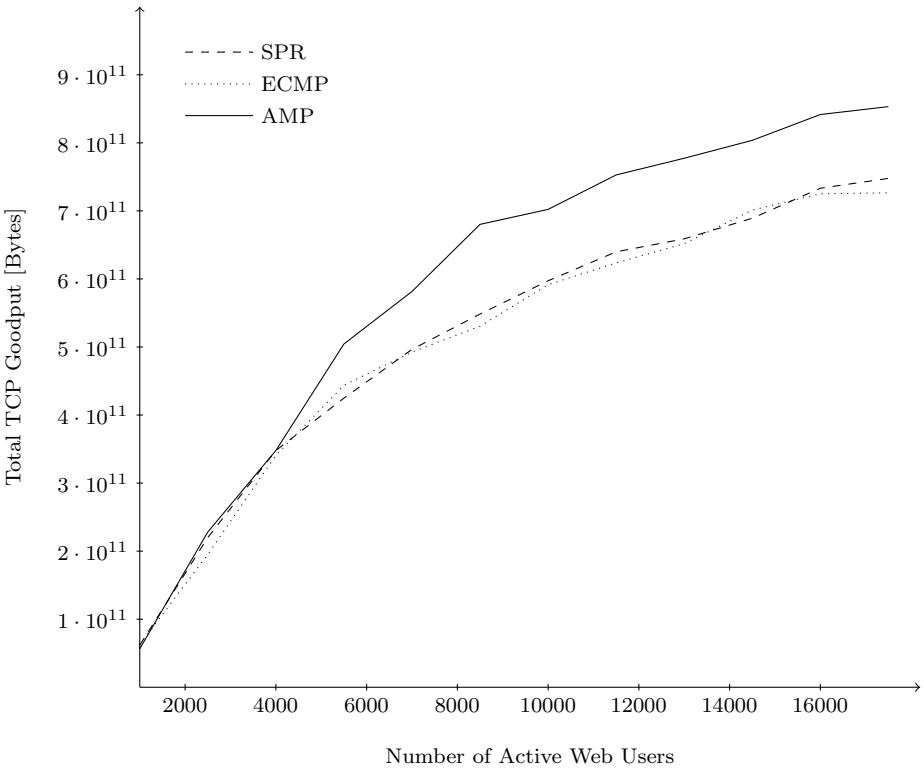



Figure 5.3.2: Total TCP goodput in the AT&T-US network.

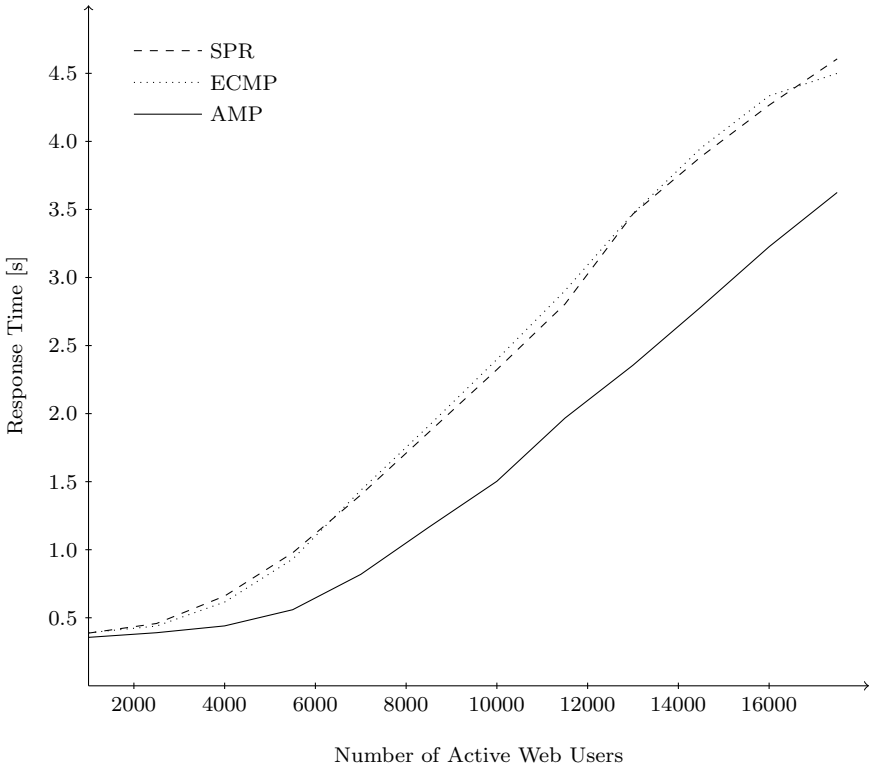



Figure 5.3.3: Average Web page response time in the AT&T-US network.

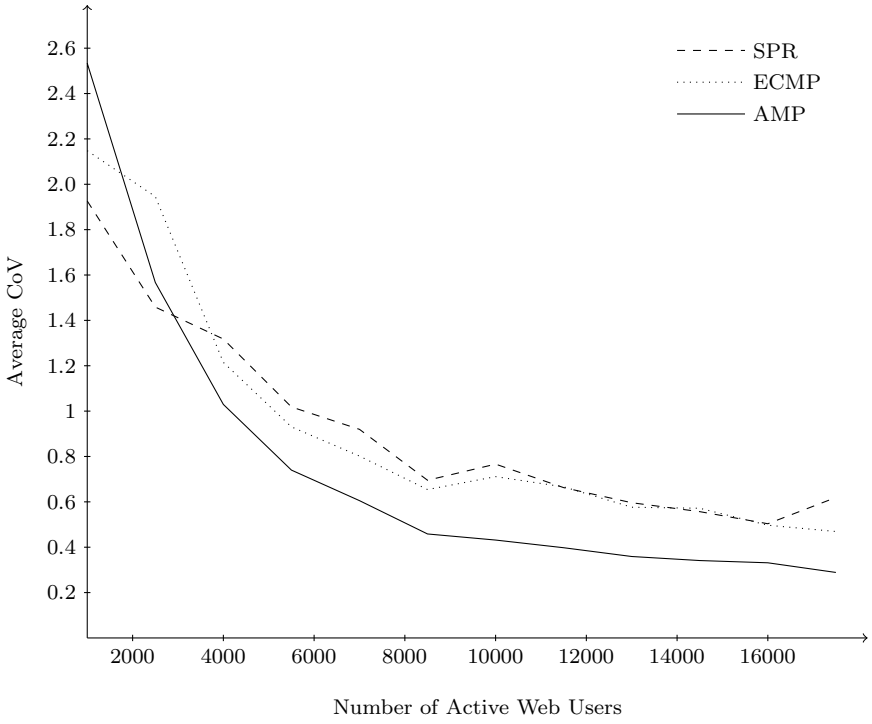



Figure 5.3.4: Average CoVs of link load in the AT&T-US network.

providing improvements of up to 28%. Note that the curve enters a state of saturation when the number of users approaches 17000, indicating that the system has reached the state of very high load.

The added benefit of AMP can also be observed in terms of the most important metric for the Web user, i.e. the average Web page response time, for which AMP achieves reductions of up to 43% compared to SPR and ECMP, whereas both static schemes perform more or less identically (Figure 5.3.3).

Finally, Figure 5.3.4 compares the average CoVs of link loads for the three routing schemes, which turn out to be quite close to each other. As SPR and ECMP by definition cannot display oscillations caused by adaptive routing, this fact demonstrates the absence of oscillating behavior for AMP throughout the investigated scenarios.

## 5.4 Performance Evaluation of AMP in the German B-WiN Research Network

While the simulations from the previous subsection demonstrate AMP's significant performance improvements in terms of Web page response times and total TCP goodput for a large IP backbone network, they do not provide an insight into AMP's performance in smaller network topologies with less multi-path diversity. Therefore, in this section the performance of AMP is investigated for the German B-WiN research network, which

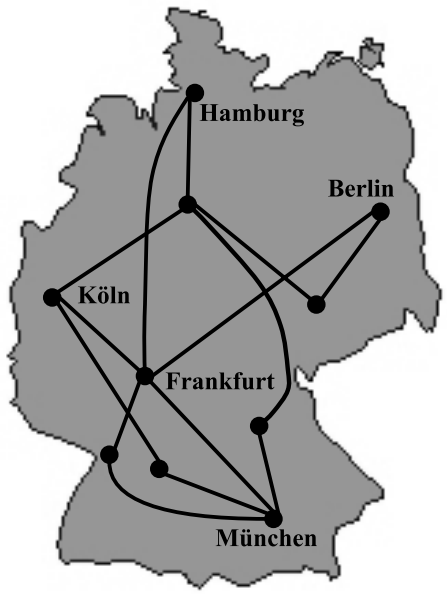



Figure 5.4.1: Topology of the German B-WiN research network.

may serve as a representative example of medium-size European ISP networks (Figure 5.4.1). The B-WiN network is comprised of 10 nodes and 14 links with link capacities ranging from 53.0 to 133.6 Mbit/s [105].

Using the B-WiN network topology, the same type of simulations has been performed as with the AT&T-US network, meaning that AMP has been compared with shortest path routing (SPR) and equal-cost multi-path routing (ECMP) for different load levels of Web traffic according to the SURGE model. The distribution of traffic again corresponds to the described gravity model, and as far as the link weights are concerned, they have been set arbitrarily either to 1 or 2 in order to ensure some degree of diversity in the environment with inconsistent link capacities.

Figures 5.4.2 and 5.4.3 present the simulation results in terms of the total TCP goodput and the Web page response time for various load conditions, ranging from very low load to very high load, close to network saturation. In terms of total TCP goodput, AMP clearly outperforms the other two routing strategies, achieving an increase of up to 18% (Figure 5.4.2). In terms of average Web page response times the performance improvements correspond to reductions of up to 35%. Furthermore, it is important to note that AMP performs consistently better than the other routing strategies for practically all investigated traffic loads in this topology, and additionally, it can be observed that for very low loads, all three routing strategies perform similarly well, as the traffic flows do not experience significant congestion in the network.

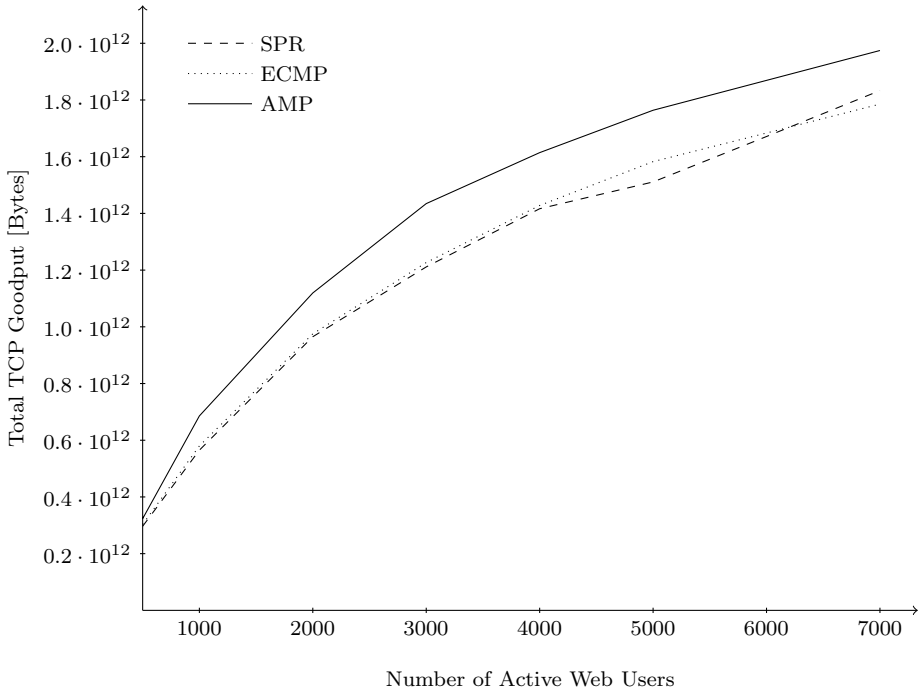



Figure 5.4.2: Total TCP goodput in the German B-WiN research network.

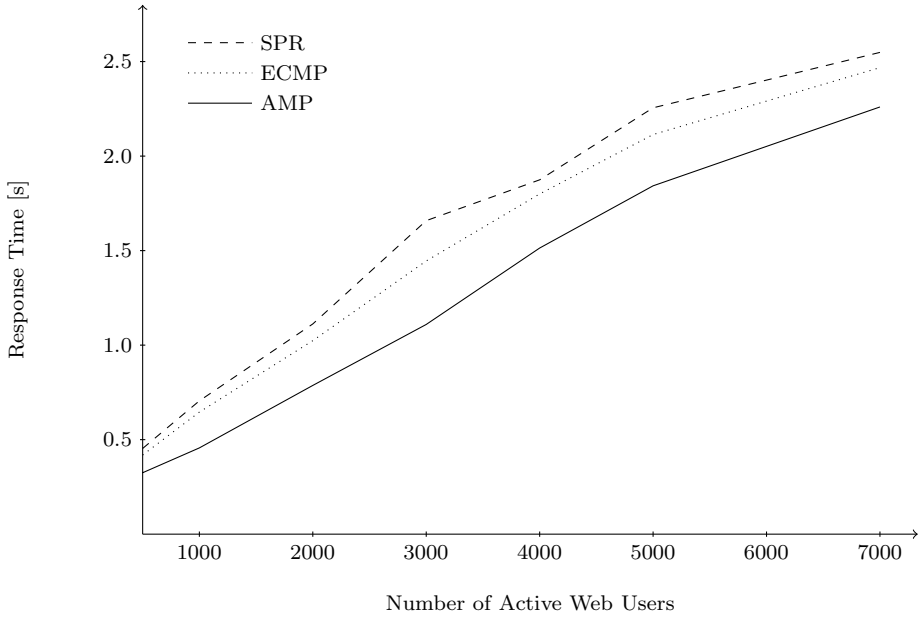



Figure 5.4.3: Average Web page response time in the German B-WiN research network.

## 5.5 Concluding Remarks

Chapters 4 and 5 present the methodologies and results of the performed AMP simulations, focusing upon AMP's traffic engineering potential in terms of load balancing efficiency and its stability as the two main indicators of algorithmic performance. For this purpose, two completely independent implementations of AMP have been realized for two distinct platforms, i.e. for the *ns-2* Network Simulator, and for a dedicated flow-level simulation framework, which has been developed with special consideration of applicability to multi-path routing problems.

Each simulation framework is especially well suited for a particular type of performance evaluation, with the flow-level and packet-level simulators proving to represent a combination of quite complementary approaches. Due to its high computational efficiency, the flow-level approach is very well suited for performing a large number of simulations with different topologies, traffic distributions, and load levels, thus paving the way towards investigating statistical performance of the algorithm. However, flow-level simulation also imposes important limitations with respect to the types of traffic which can be investigated, as there are no efficient flow-level models for elastic traffic like TCP which would capture the whole spectrum of associated effects correctly.

Vice versa, packet-level simulation can almost perfectly reproduce the effects observed with all types of traffic, as very mature protocol stacks of different stateful mechanisms (especially TCP) are available in standard network simulators. However, the packet-level approach is also inherently associated with natural limitations concerning its scalability – large network topologies with realistic traffic models can hardly be simulated within a reasonable time frame, as the number of simulated events per duration of simulated time grows linearly with the total transmission capacity of the investigated network. This is particularly troublesome in the context of traffic engineering and routing in general, as many desirable and undesirable effects of the different algorithms appear only in large network structures.

In Section 4.3, the statistical performance of AMP is investigated by means of a large number of simulations with different experiment setups. Specifically, 5 different topology sizes ranging from 10 to 30 nodes have been used in order to cover a representative spectrum of topology sizes, and 20 distinct artificial topologies have been created for each size. The resulting number of 100 different topologies has further been simulated with five different load levels of CBR traffic for each topology, always in combination with two distinct link weight setting strategies. This amounts to a total of 1000 performed simulations, which represents a reasonable number for making meaningful staristical performance estimates of the algorithm. The results demonstrate significant performance improvements of AMP, which are larger for lower levels of overload (as may be expected due to greater amounts of residual capacity), reaching packet loss rate reductions of almost 50%.

In addition to the macroscopic performance evaluation focused upon the global performance of AMP, the flow-level simulation framework has also been used for microscopic experiments aimed at investigating the stability and convergence properties of the AMP load balancing mechanism by studying the algorithm's effects upon individual traffic flows. For this purpose, Section 4.4 presents several types of experiments which investigate both AMP's convergence towards a stable traffic distribution for a time-invariant traffic matrix, and AMP's response to unstable traffic patterns which is of crucial importance for overall system stability. Apart from reliable and rapid convergence towards a fix point in traffic distribution, the simulations have also demonstrated oscillation free performance of AMP in the presence of disruptive input traffic, thereby raising confidence in stable AMP operation in larger network structures.

In Sections 5.3 and 5.4, AMP is investigated using *ns-2* packet-level simulation for two real ISP topologies of different size, using the SURGE Web traffic model in combination with realistic traffic distribution according to the gravity model. The algorithm is compared to shortest path routing and equal-cost multi-path routing as the standard traffic agnostic routing strategies which are applied in real ISP networks. In terms of algorithm performance, the focus is set upon two metrics of major practical relevance: Web page response times are investigated in order to gain insight into the potential impact of different routing strategies upon individual users, whereas total TCP goodput in the network reflects the overall efficiency of network resource usage. In both the large-scale AT&T-US network topology, and the medium-size topology of the German B-WiN Research Network, AMP has proved to achieve significant performance improvements over the two traditional routing strategies. In the case of the AT&T network AMP achieves Web page response time reductions of up to 43%, and similarly, in the case of the B-WiN network response time reductions of up to 35% are achieved. As far as algorithm stability is concerned, in Section 5.3 the averages of the coefficient of variation of link load for each of the routing strategies have been compared, with the result that they are quite close to each other. As the traffic-insensitive routing strategies cannot display routing induced oscillations by definition, this fact further underlines the findings about stable AMP behavior from Section 4.4 on a global scale.

Finally, both evaluation approaches used, i.e. flow-level simulation and packet level simulation, demonstrate the consistent performance improvements achieved by AMP in situations of network overload. These improvements are most profound in situations of light to moderate congestion, as then AMP encounters enough inherent traffic engineering potential for load reallocation, whereas any traffic engineering scheme can at best achieve only slight improvements in conditions of major network-wide overload. As far as AMP stability is concerned, both approaches again produce consistent results, while addressing the issue from completely different perspectives – in case of flow-level simulation the overall network stability in the presence of AMP routing has been investigated, whereas using flow-level simulation specific microscopic scenarios have been

developed in order to examine the behavior of AMP's load balancing mechanism. Most importantly, the results of both evaluation approaches yield mutually consistent results, demonstrating the stability of AMP routing throughout the investigated scenarios.

# Chapter 6

# Applications of the Adaptive Multi-Path Routing Algorithm

## 6.1 Traffic Engineering in the Context of Today's Internet – A Critical Discussion

Before concluding the thesis with a couple of promising examples for the application of AMP in current and upcoming networking scenarios, this section aims at providing a somewhat critical but still realistic reflection on the relevance of traffic engineering in general, especially in the context of current Internet Service Provider (ISP) networks and networking technologies. For this purpose, a review of contemporary ISP practices with respect to resilience and quality of service (QoS) issues at the IP layer is provided, which eventually leads to a more focused view on potential future application scenarios for AMP. Of course, the remark can be made that resilience engineering could also be performed at lower layers, however this would assume complete dependence e.g. upon link-layer mechanisms which themselves are also prone to failures. Moreover, these lower layer mechanisms cannot address issues related to layer-3 equipment failures, meaning that e.g. IP router failures still necessitate the enforcement of adequate resilience responses also on layer-3. Therefore, the subsequent discussion is restricted to the IP layer only.

The main objective of commercial Internet Service Providers is to meet their customers' expectations concerning the provisioned data transport, which are often formalized as Service Level Agreements (SLAs), specifying the required technical parameters of the offered service [106]. On the one hand, these parameters describe the required data transport standards, like e.g. minimum (i.e. guaranteed) bandwidth, packet loss characteristics, packet delay and delay variation, and on the other hand they also specify the reliability and availability of the provisioned Internet connectivity. The former may be summarized under the term *QoS in the strict sense*, whereas the term *QoS in the wide sense* may be used if service availability aspects are additionally considered [107].

As service guarantees are very difficult to deliver for connections which traverse multiple domains, SLAs normally apply only to IP traffic which originates and terminates within the network of a single ISP. There are many different ways towards achieving the desired strict-sense QoS, ranging from approaches like DiffServ, IntServ, and MPLS, to the relatively simple bandwidth overprovisioning paradigm which does not necessitate the introduction of novel routing or switching technologies into the core network. Furthermore, as the customers are typically not interested in the data transport technology and infrastructure beyond their own premises, in most cases ISPs can freely choose their QoS approach.

Apart from their technical objectives and SLA obligations, commercial ISPs also need to generate revenue which will ensure that their operation is economically viable. Therefore, a major task of all ISPs is to ensure that they operate their networks efficiently, making maximum use of the installed network capacity while at the same time fulfilling the sometimes stringent SLA requirements, and in this way enabling a competitive Internet service offering. For this purpose, ISPs will typically also consider traffic engineering activities and techniques which would enable the optimization of their efficiency. In this context it is important to mention that strict-sense QoS and traffic engineering should be regarded as two independent disciplines: whereas strict-sense QoS focuses upon traffic characteristics at the time-scale of connection round trip times, traffic engineering activities aim at optimizing load distribution at the time scale of multiple minutes and hours. However, as networks with optimized load allocation may also display more favorable QoS properties due to reduced occurrences of congestion, it is legitimate to consider strict-sense QoS and traffic engineering as *close friends*.

As far as service availability is concerned, stringent SLAs require that there are no significant service interruptions even in the case of equipment outages in the core network. This directly necessitates hard requirements with respect to residual network capacity for the anticipated cases of failures (cf. [108]). In other words, as each link or router failure in the network has the potential of reducing the *min cut – max flow* capacity between a pair of nodes in the network, the topology, link capacities, and routing mechanisms should be designed such that they enable a loss-free diversion of traffic from the failed path(s) onto the alternative path(s). As a positive side-effect, during regular network operation (i.e. in the absence of failures) this translates to a certain percentage of network capacity which cannot be actively used, but which at the same time improves strict-sense QoS as the average link utilization is reduced, leading to overall reductions in packet loss rate and delay variation.

In the case of plain IP routing, which is based upon the standard intra-domain routing architecture using link weights (cf. Chapter 2), residual bandwidth requirements often correspond to the restriction that link utilization should never exceed 50%, as in the case of failure another link should be able to take over the entire traffic of the failed link, provided that single path routing is used and that all links have equal capacities.

And indeed, due to a lack of sophisticated planning for resilience, the 50%-mark often serves a a rule of thumb in many ISP networks. Accordingly, even major U.S. tier-1 ISPs like *Sprint* recognize bandwidth overprovisioning as a valid engineering practice for robust provisioning of IP core networks in the absence of fine-grained information about the traffic dynamics inside the network [41]. Maybe even more astonishingly, opting for a radical native IP design Sprint relies solely upon link state routing mechanisms for rerouting traffic. In order to achieve fast reactions to failures, they merely reduce link state-specific timers for link state advertisement distribution and route recalculation, which reliably enables sub-second recovery with current IP routing hardware [26].

The most interesting question still left open now is which provisioning scheme makes most sense both from a technological and an economical point of view. If link and switching capacities were available in abundance, then the native IP approach by Sprint would represent the most promising candidate, as all performance issues could reliably be solved by the simple overprovisioning scheme. In such a scenario, the attractiveness of traffic engineering techniques would certainly decline, as in practice there would not be any need to counteract congestion events on a medium time-scale of minutes or hours. At most, offline traffic engineering approaches like link weight optimization [30, 31, 32] might still be useful for optimal adaptation of the routes to the peak traffic demand, however at the cost of dropping any of the heuristic link weight setting paradigms which are often favored by network administrators for reasons of intuitive routing setup and transparency. On the other hand, if network capacity represented a scarce resource, approaches which optimize network utilization with respect to resilience requirements, like e.g. in [28] or [108], would certainly gain in attractiveness. The same is also true for strict-sense QoS paradigms like e.g. DiffServ, as presumably a low percentage of high-priority traffic (like e.g. emergency services) would require guarantees for lossless and timely delivery even in the presence of severe component outages.

Currently, the majority of ISPs employs the bandwidth overprovisioning scheme, as the transport capacity is readily available and affordable [109]. Furthermore, as both link capacity and router switching capacity grow roughly according to Moore's Law, doubling every 12 months [110] and 16 months (cf. [111]), respectively, and as only a fraction of the globally available optical transport capacity has yet been put into use, there do not seem to be any strong technological or market forces towards abandoning this scheme. Nevertheless, the overprovisioning approach is still very much disputed, especially in the Internet Economics research community, where it is either considered to be a valid engineering approach in the presence of abundant resources, or at most a suboptimal *best practice* solution which bears the risk of economic inefficiency due to the well-known *tragedy of the commons* [112].

In any case, as long as bandwidth in core networks is not among the cricital network resources, traffic engineering might remain a discipline of lesser importance. This means that currently there are no strong incentives for standardization bodies, equipment ven-

dors, or network operators to adopt novel routing schemes, even if they provably perform well for a large variety of realistic scenarios. However, it would be very unwise to think of traffic engineering mechanisms exclusively in the context of ISP backbone networks, because many of them are generically suited for all types of fixed IP infrastructures. A typical example for a field of application in which the available capacity in the IP backbone network cannot be abundant by definition will be presented towards the end of this chapter, advancing AMP to become a very attractive potential solution.

## 6.2 Applicability of Adaptive Multi-Path Routing to IP Overlay Networking

As a first example, this section discusses whether AMP can provide an attractive routing solution for IP overlay networks. In recent years, this architecture has become one of the most prominent topics in Internet research [113, 114, 115, 116, 117], as routing traffic at the application level holds the potential of increasing network robustness by circumventing slow inter-domain routing convergence, as well as increasing quality-of-service (QoS) by choosing the communication paths more freely without having to rely on standard IP routing mechanisms.

Therefore, in the context of AMP the question naturally arises whether there are networking scenarios in which the proposed mechanisms could be reused in an overlay routing architecture. Intuitively, an application of AMP similar to the well-known Resilient Overlay Network (RON) concept as proposed in [113] could be envisioned, where a full mesh overlay of application level routers are used for choosing the best paths between the overlay nodes. In the RON scheme, the overlay nodes aggressively probe the paths for their instantaneous performance, and they exchange this information frequently with all other nodes.

However, there is a fundamental difference between RON and AMP with respect to their routing actions: RON chooses and uses the best available path in a fashion which is comparable to Dynamic Alternative Routing (DAR), i.e. either the direct Internet path towards the other overlay node is chosen, or the indirect path via a single intermediate RON-node is taken. Alternatively, in order to reduce the perceived packet losses of individual connections at the receiver, RON also holds the possibility of duplicating traffic between two overlay nodes by sending it simultaneously via two different paths. In contrast, AMP does not make routing decisions in a binary fashion. Instead, it is fundamental to AMP's principle of operation that traffic is gradually shifted between the available paths (i.e. next hops) until an equilibrium of load and backpressure is reached. In other words, the essential property of AMP is that it is a *feedback mechanism*, meaning that AMP's fundamental assumption is that the routing actions it takes have the potential of significantly influencing network conditions. Therefore, the application

of AMP mechanisms in overlay networks is not purposeful in scenarios where the core network bandwidth greatly exceeds the available access link bandwidth of the overlay nodes, as the traffic generated by the overlay will not have the potential of significantly impacting core network conditions. In this scenario, an overlay network like RON with binary routing decisions could make sense for choosing the best path for metrics like transmission delay, as e.g. satellite links could reliably be circumvented, however, a fine-grained feedback mechanism like AMP would not be able to add any comparative benefit.

## 6.3 Applicability of Adaptive Multi-Path Routing in Wireless Mesh Networks

In many fields of application, wireless networks constitute an important element of the communication infrastructure. In certain areas, like e.g. telephony, the wireless mobile alternative has experienced rapid growth in recent years, and it has become an indispensible tool for personal real-time communication. However, in numerous other areas of application wireless networks either have not yet seen broad deployment among the general population, or their nature is such that their operation is largely concealed to the end users.

The most interesting type of such networks are *wireless mesh networks*, which enable multi-hop communication across a wireless infrastructure. In general, wireless mesh networks can be subdivided into three categories according to their architecture [118]:

- *Infrastructure/backbone wireless mesh networks.* In this type of networks, a mesh of wireless nodes forms the communication backbone for the clients (Figure 6.3.1). With gateway functionality, the backbone nodes can also be connected to other networks, like e.g. the Internet, and in this way provide an access infrastructure for their clients. This solution of *infrastructure meshing* is particularly interesting in case no wired connectivity is at hand, or if an alternative/complementary network has to be built.

- *Client wireless mesh networks.* Apart from infrastructure networks, wireless meshes can also be comprised exclusively of end devices, i.e. *clients* (Figure 6.3.2). In this type of networks, the nodes provide both routing functionalities, and serve as an application platform to their users. As the traffic between a pair of nodes can be carried via an intermediate relay node, client wireless mesh networks have the potential of increasing the efficiency of bandwidth utilization and the network size, as well as reducing the transmission power of individual nodes, thus leading to improved conservation of battery resources.

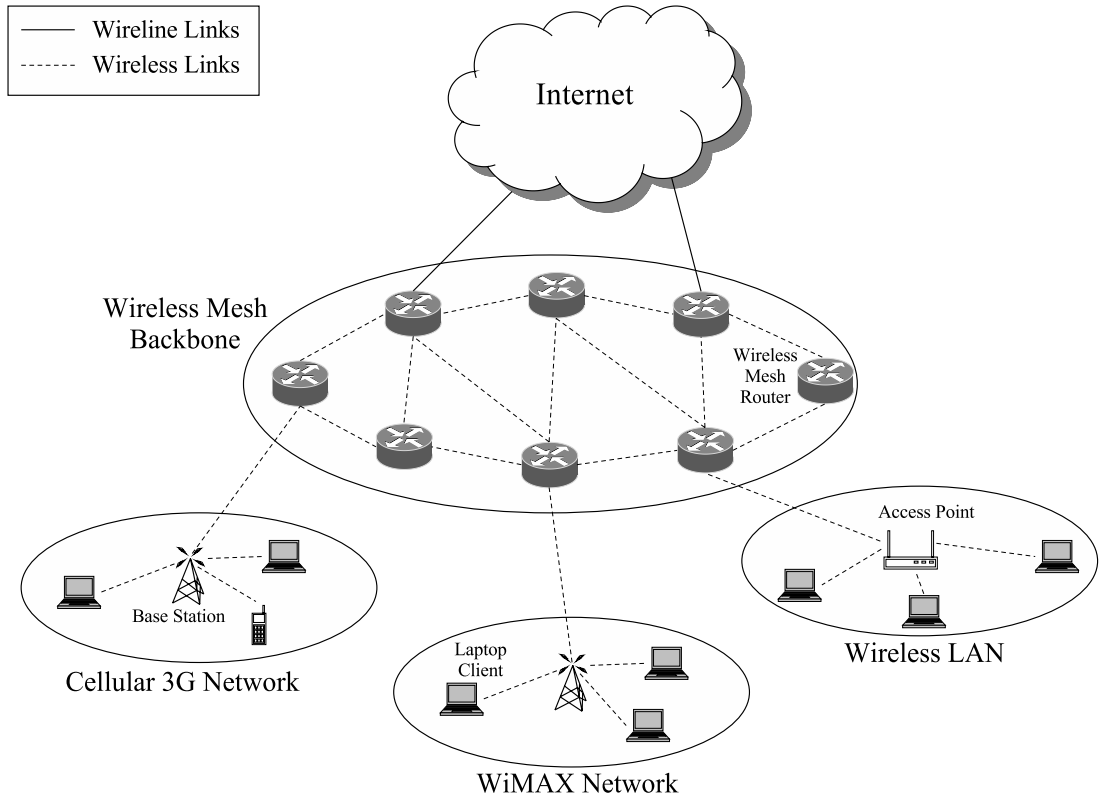



Figure 6.3.1: Example of an infrastructure/backbone wireless mesh network.

- *Hybrid wireless mesh networks.* If mesh networking capabilities are present both at infrastructure level (i.e. between the backbone nodes) and the client level, this type of network is denoted as a *hybrid* wireless mesh (Figure 6.3.3). Hybrid wireless mesh networks have the potential of further increasing communication efficiency, as individual clients can communicate directly in some cases, thereby obsoleting relay services of backbone nodes.

In recent years, wireless mesh networks have mainly been studied in the context of *ad hoc* networking, where it is assumed that a group of nodes distributed over some geographic area sets up a connected multihop network in a wireless manner without resorting to external configuration efforts. For this purpose, special routing protocols have been developed which partially reuse the ideas of the existing link-state and distance vector routing algorithms, but modify their functionalities towards supporting environments with significant node mobility, and frequently changing topologies [119, 120]. The main requirements set upon these protocols and system design in general were to ensure network scalability, efficient bandwidth utilization, low power consumption (which is of special importance in the context of finite lifetime nodes, i.e. devices with limited battery capacity), QoS, etc.

In addition to ad hoc networking, wireless mesh networks have lately also gained im-

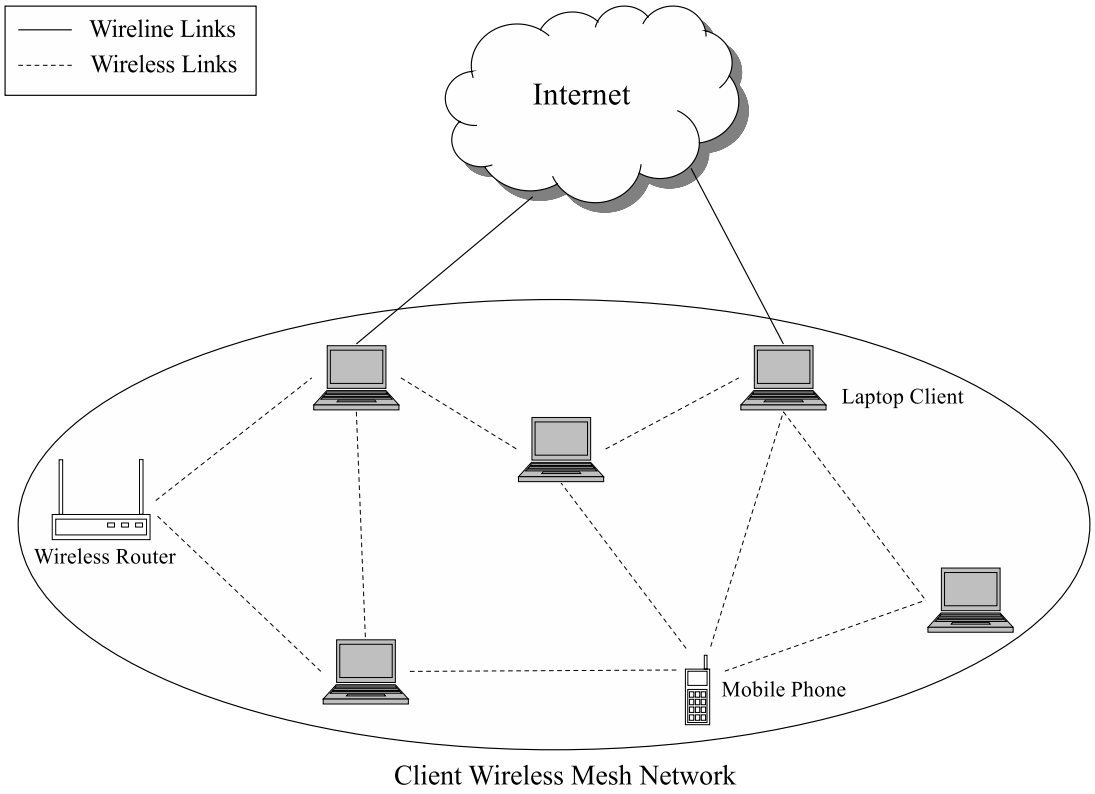



Figure 6.3.2: Example of a client wireless mesh network.

portance in the area of planned fixed network infrastructures. The main driving factor in this development was the advent of two novel types of wireless technologies, i.e. IEEE 802.11 Wireless Local Area Networks (WLANs) [www-13] and IEEE 802.16 Worldwide Interoperability for Microwave Access (WiMAX) networks [www-14], in combination with efforts towards provisioning Internet connectivity without resorting to existing wireline installations. In other words, the combination of a wireless last mile and a completely wireless backbone infrastructure holds the potential of enormous cost savings, as no dedicated lines need to be leased from other providers and no new wireline infrastructures need to be deployed. Furthermore, if the envisioned wireless backbone does not resort to licensed spectrum technologies, which is e.g. the case with IEEE 802.11 WLANs, entirely new network operators can start commercial operation while almost completely circumventing the relevant regulatory procedures.

From a technical point of view, fixed infrastructure wireless mesh backbone networks also hold the advantage of dramatically reduced complexity in comparison with their mobile, ad hoc counterparts. As backbone connections can be built using *directional* wireless links, many considerations which are necessary in the specific context of ad hoc networks are obsolete, as it may be assumed that the directed links are far less prone to signal interferences [121]. Therefore, with this type of networks the whole suite of traditional IP protocols can be reused, including standard intra-domain routing protocols

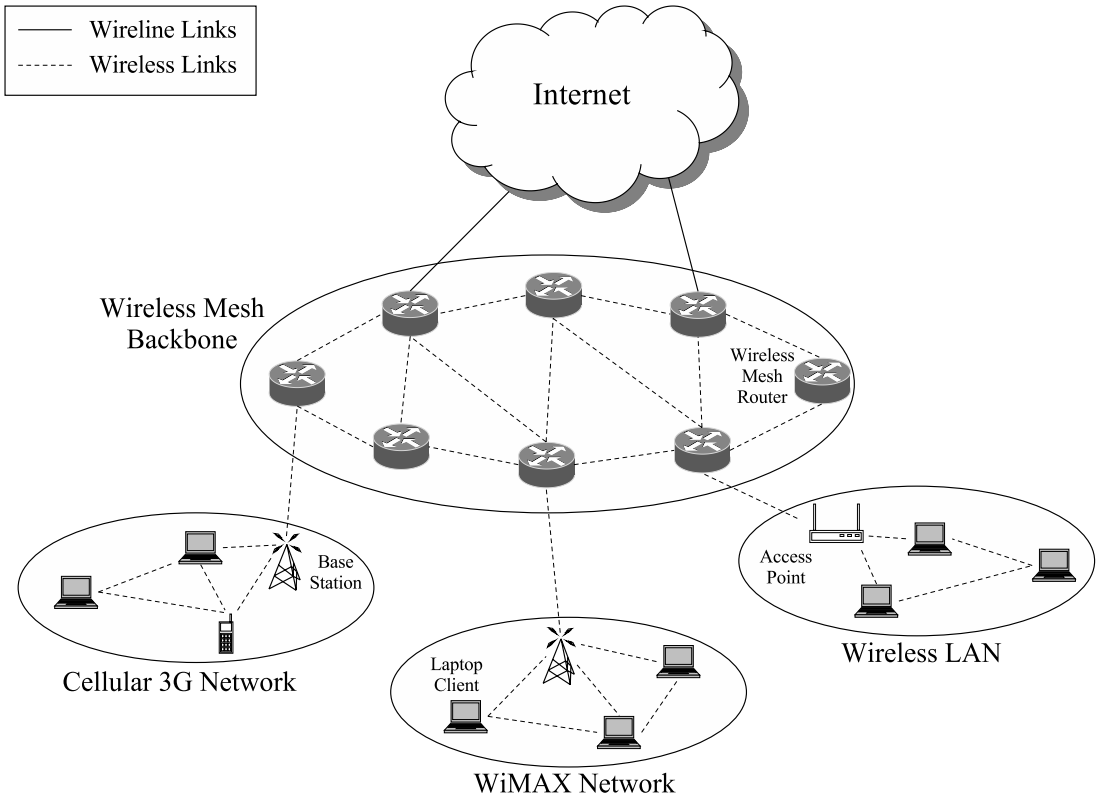



Figure 6.3.3: Example of a hybrid wireless mesh network.

like OSPF or IS-IS [122]. A typical example of infrastructure wireless mesh networks, the *Magnets* network in Berlin, can be found in [www-15], where a hybrid system of access point connected into a wireless mesh backbone network is built solely using IEEE 802.11 WLAN technology in combination with directional antennas. Contrary to the widespread belief that interferences with local WLAN hot-spots will render such a design inefficient, [121] demonstrates that with careful planning and deployment, the standard IEEE 802.11a and IEEE 802.11g technologies enable the setup of reliable backbone networks with very stable throughput characteristics even in dense metropolitan areas.

In [123], AMP has been investigated *in concreto* for a commercial metropolitan ISP network which is comprised solely of an IEEE 802.11a IP backbone connecting private and public Internet hotspots into a wireless mesh backbone. Figure 6.3.4 depicts the first deployment phase of the corresponding test network which has been set up in a small-size German city [123]. Note that the grey circles correspond to the coverage area of local IEEE 802.11 access points, which are interconnected with directional IEEE 802.11 radio links (black lines). During the setup it has turned out that the network topology mainly depends upon the geographic topology and various legal requirements (e.g. the consent of real-estate owners, or regulations concerning the protection of historic buildings and monuments). Accordingly, directional links can only be planned using available locations, which additionally require lines-of-sight (LOS) towards other

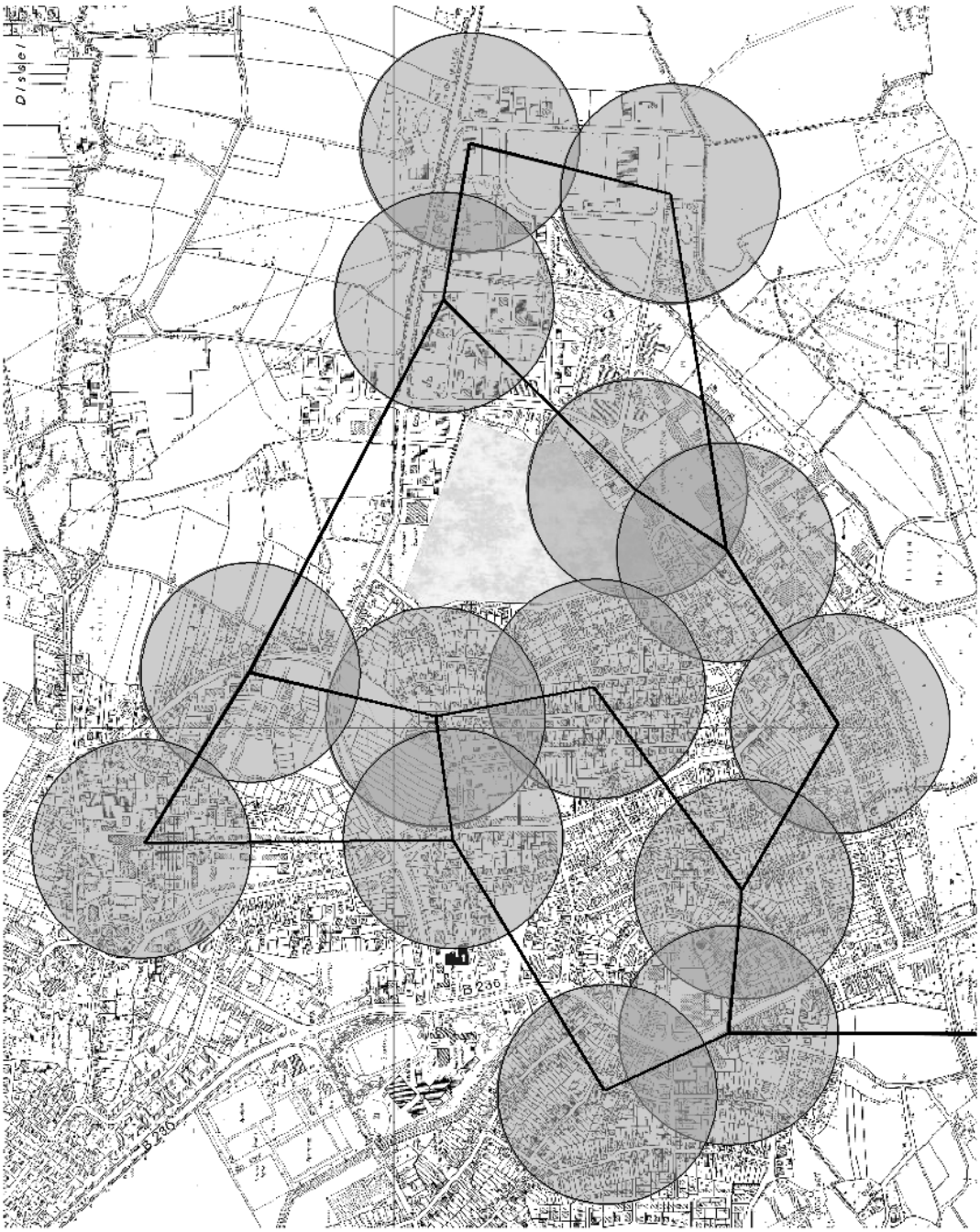


Figure 6.3.4: Local-Web wireless backbone network. (Source: [123])

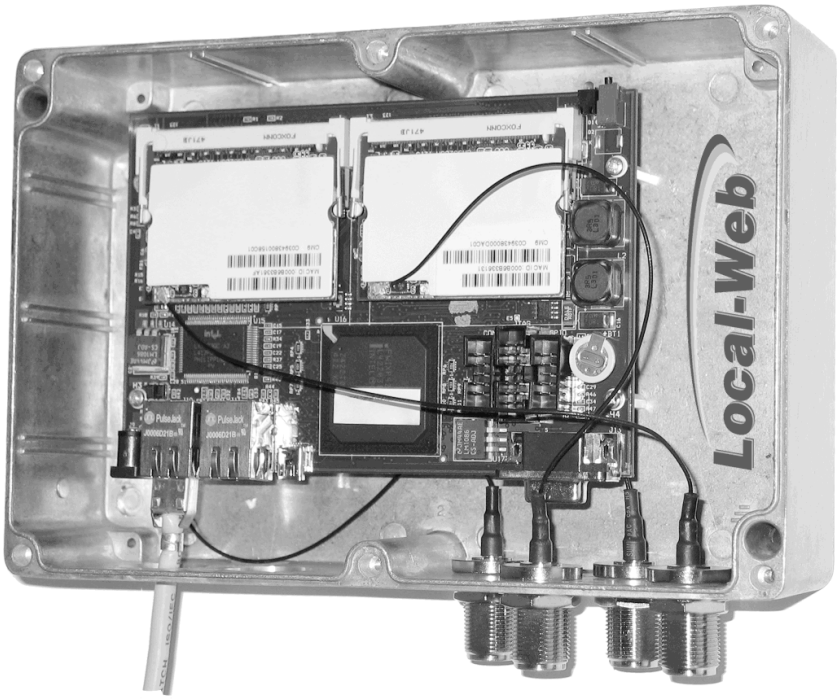



Figure 6.3.5: Local-Web wireless backbone router. (Source: [123])

available locations. Taking these strict boundary conditions into account, the remaining degrees of freedom for planning the network topology turn out to be very limited, such that it is nearly impossible to end up with some sort of a regular network structure which would allow for a high level of multi-path diversity.

In addition to the description of the network structure, [123] also clearly identifies the necessity of traffic engineering for wireless mesh backbones, as bandwidth represents an inherently scarce resource in this type of networking solutions. Accordingly, a number of traditional traffic engineering schemes is investigated, with the objective that the results should be well applicable to wireless mesh networks with directional links.

For the investigated scenario, [123] finds that AMP should be similarly well applicable to infrastructure-type wireless mesh networks with directional links, as in the case of wireline backbone IP networks, which has been demonstrated in [13] and [14]. The main reason for the validity of this analogy is that the wireless mesh topologies envisioned for the final project phases, as well as the corresponding traffic types match the investigated core network scenarios quite well.

As far as the implementation of AMP in the production network is concerned, the custom tailored router depicted in Figure 6.3.5 has been foreseen. The router is equipped with two ethernet interfaces and four wireless network interfaces, with one of them dedicated for usage as a local access point for the coverage area around the router, such that normally three directional links are available for interconnection.

However, several open questions still remain in this context, as wireless mesh backbones may be assumed to have lower levels of overall network availability, and as wireless

backbone links may display a non-negligible level of capacity oscillations. Therefore, AMPs behavior in the presence of unstable network conditions should be further investigated in order to optimize the algorithm's parameters for the envisioned scenario.

3G wireless networks like e.g. UMTS have recently also emerged as an important area of application for wireless mesh infrastructure networks which connect the 3G base stations to the 3G core network [124, 125]. The main reason for the growing importance of wireless mesh solutions in the context of 3G networks lies in the partial inability of the current wireline infrastructure to carry high capacity data traffic. In other words, the introduction of novel technologies like the High Speed Downlink Packet Access (HSDPA), which offers downlink data speeds to end users of up to 14 Mbit/s [126], and the advent of other technologies which are likely to increase transport capacities by an order of magnitude as part of the Long Term Evolution (LTE) of 3G networks [127], necessitate novel solutions for the communication links between the 3G base stations and the 3G core. While the required speeds can be achieved by the introduction of more efficient wireline technologies, the IEEE 802.16 WiMAX technology represents an attractive alternative, especially if there is no legacy infrastructure at hand, which is often the case in rural areas with suboptimal wireless coverage [124]. In this case, a combination of high-speed 3G base stations and a multihop wireless mesh IEEE 802.16 backhaul network may prove to be a very efficient option for introducing broadband Internet access. Depending upon the design and the topology of the wireless mesh infrastructure in such a scenario, AMP may again play a constructive role in distributing traffic evenly throughout the deployed access network.

## 6.4 Concluding Remarks

In this chapter possible applications of the Adaptive Multi-Path algorithm (AMP) have been investigated in the contemporary networking context. Starting with a critical discussion about the necessity of traffic engineering activities, a holistic perspective of issues in current ISP networks is taken, leading to several interesting conclusions. The issue which is most often disregarded in architectural considerations is that for most cases of deployed ISP core networks it is highly unrealistic to attempt to effectively *squeeze out* the last bit of the available transmission capacity. This is primarily due to stringent SLAs which impose high resilience requirements, which in turn necessitate a certain degree of residual capacity being present in the network during regular operation. In the case of network faults, this spare capacity is then used for accommodating traffic from failed paths. In other words, network overprovisioning can be regarded as a natural by-product of practical engineering for resilience, and furthermore, as the overprovisioning approach inherently minimizes queue build-up and packet losses on the links, the quality-of-service (QoS) observed by the traffic flows is automatically increased throughout the network domain. Therefore, overprovisioning is today widely regarded and applied as a valid

approach for operating ISP networks, representing a simple and efficient mechanism for assuring compliance with the growingly demanding SLAs.

In contrast to the current technological context of IP core networks, which display exponential growth of transmission and switching capacity, there is a variety of emerging problems and applications in which capacity inherently represents a scarce resource, advancing traffic engineering to become an important practical discipline in those areas. A typical example in the context of IP networking are the increasingly relevant *wireless mesh networks*. Due to many analogies between the structures of IP core networks and *fixed infrastructure* wireless mesh networks with directional links, the existing IP traffic engineering techniques, including AMP, are particularly attractive for such novel use cases. Beyond AMP's theoretical applicability in wireless mesh networks, a concrete example of a commercial ISP operating a wireless mesh backhaul network has been presented, where the network operator itself has envisioned AMP as the best available traffic engineering solution for practical implementation throughout the deployed network.

# Chapter 7

# Conclusions and Future Research Directions

Throughout the history of telecommunications, the efficiency of network design, and the optimization of the deployed resources have been in the center of scientific and practical efforts. Initially, the structures which had to be built were telegraph and telephony networks, which were planned and deployed in a highly hierarchical manner, dimensioned according to some practical experiences. Fortunately, theoretical models which allowed for proper network dimensioning, particularly the model by Erlang, became available very early [77], subsequently enabling the deployment of highly optimized networks with respect to the expected traffic demands. Due to the mentioned hierarchical nature of those networks, and the relatively simple circuit switched call model, their planning did not necessitate in depth graph theoretical foundations, which at that time largely were not available anyhow.

However, with the advent of packet switched data networks in the 1960ies [128], things have radically changed for several reasons: firstly, the old circuit switched call model with its Poisson process for call arrivals and exponentially distributed call holding times was very inefficient for novel interactive data applications like e.g. *remote login* [129], and secondly, the nature of the first networks of this kind was inherently non-hierarchical, such that it was not possible to design them in complete analogy to circuit switched telephony networks. But fortunately, at the same time as the problems in building packet switched data networks started to become clear, the mathematical discipline of graph theory has made several seminal advances, the most prominent of which are certainly the works by Dijkstra, Bellman, Ford, and Fulkerson [4, 5, 6]. All of them have investigated the *shortest path problem*, which is fundamental to establishing any form of communication in network structures which are not completely hierarchical.

As graphs are merely mathematical abstractions of networks, the results could very well be translated into practical solutions for routing traffic along shortest paths in connectionless data networks. Accordingly, the graph theoretical findings have been trans-

lated into concrete network protocols for disseminating topology information throughout a network domain in a distributed manner, yielding two distinct groups of routing mechanisms, i.e. *distance vector* and *link state* protocols [3].

Although the developed protocols have proved to work more or less reliably (with link state protocols in the course of time having turned out to be the more reasonable choice for most types of problems), they mostly had the inherent drawback of considering solely the network topology as an input for their route calculation. More specifically, each link in the topology is statically assigned a scalar value called the *link weight*, and traffic is routed along those paths for which the sum of their link weights is minimal. In other words, these protocols are *traffic insensitive*, meaning that in such networks traffic may be routed along an overloaded (i.e. congested) path, even if parallel uncongested paths exist. This fact has motivated the emergence of the discipline of *traffic engineering*, which aims at optimizing the usage of existing resources in operational networks [10]. Unfortunately, initially there were only a few and, from today's perspective, slightly naive approaches, which had only limited success in optimizing the load allocation. The main reason for this lies in the fact that in weighted directed graphs the routes can only be influenced implicitly by introducing changes to individual link weights. But as each change to a single link weight can impact a multitude of routes (especially if the link weight is decreased), the possibilities of this type of intervention have soon proved to be too coarse for achieving the desired load allocation in an intuitive and purposeful manner [60, 63].

Accordingly, many different solutions have been proposed, like e.g. optimizing the network wide link weights using heuristic algorithms which take the network wide traffic matrix as an input [31]. Similarly, some old ideas from the circuit switching world have been re-introduced in the form of Multi-Protocol Label Switching (MPLS), which basically enables the switching of virtual circuits in IP networks [44]. However, both of these approaches are associated with enormous network management overhead, as they require the knowledge of the entire network-wide traffic matrix for efficient operation. In other words, switching circuits in a non-hierarchical network is far away from being an intuitively solvable task, but instead it rather represents a complex optimization problem of its own.

Consequently, the idea of dynamic routing protocols has emerged, with the goal of autonomous load distribution mechanisms being deployed inside the network nodes, which take their actions based upon information exchanged throughout the network domain. One of the most prominent early theoretical examples of such an approach is certainly the algorithm by Gallager, which achieves the minimization of the average packet delay using only minimal signaling resources by restricting the exchange of information to neighbor nodes [17]. However, Gallager's approach can only serve for the optimization of networks with quasi-static traffic matrices, and therefore its applicability in realistic scenarios is limited. More recently, the Optimized Multi-Path (OMP)

approach has been introduced for optimizing the network-wide load allocation for individual IP network domains [12]. The essential idea of OMP is to reuse the link state routing's flooding mechanism for the network-wide dissemination of load information, and to use the loading values of distant links in each node autonomously for making fine-grained changes to the distribution of traffic among multiple paths from source to destination. Unfortunately, at the same time OMP also introduces non-deterministic signaling overhead, which can become quite excessive whenever link state flooding is used, and furthermore it requires extensive memory capacities inside the routers.

Aiming at combining the most favorable properties of the existing approaches, this thesis proposes and evaluates Adaptive Multi-Path routing (AMP) as its central contribution. The essential idea of AMP is to reduce the exchange of signaling messages to the local scope by restricting the exchange of load information to neighbor nodes [13, 14]. Although being exchanged only locally, the so-called *backpressure* signaling messages include information about the state of the network beyond the next hop in a quasi-recursive manner, such that they simultaneously enable the seemingly contrary objectives of local signaling and global propagation of load information throughout the entire network domain. More precisely, each upstream node which sends traffic, and thus potentially causes disturbances somewhere downstream in the network, will be informed about its approximate relative contribution to congestion. In response to the received backpressure messages, as well as to the load directly measured on its output links, each node will subsequently try to offload the congested next hop paths (i.e. links) by reallocating the traffic to less congested areas of the network. For reasons of simplicity and compatibility with the current IP routing infrastructure, the multi-path structures required for load distribution are determined solely based upon the statically assigned link weights. Furthermore, in order to avoid any oscillations of the routing as a reaction to changing traffic patterns, a fine-grained load balancing mechanism with sophisticated load shifting dynamics has been applied.

The performance of AMP has been subject to a detailed evaluation, based on using two distinct simulation techniques, i.e. flow-level simulation and packet-level simulation. In the flow-level case, an entirely novel simulation framework has been developed which is especially well suited for the investigation of multi-path routing problems, and the entire functionality of AMP has been implemented therein. Using this simulation environment the statistical performance of AMP has been evaluated in over 1000 scenarios, which are based on state-of-the-art know-how concerning the construction of artificial communication network topologies and models for non-elastic traffic. The obtained simulation results demonstrate significant performance improvements obtained by AMP compared to the static routing strategies for a wide spectrum of traffic overload intensities. Furthermore, using the flow-level simulation framework, AMP's stability has been studied by applying disruptive traffic patterns in example networks, trying to provokatively drive the algorithm into oscillations. However, in all investigated scenarios

AMP has proved to operate oscillation-free, which of course represents the fundamental prerequisite for the algorithm's deployment in realistic network scenarios.

As far as packet-level simulation is concerned, the complete functionality of AMP has also been implemented in the widely used *ns-2* Network Simulator. Using this simulation tool, performance evaluations which are complementary to the ones carried out in the flow-level environment were made, focusing primarily on AMP performance in the presence of highly realistic elastic traffic, like e.g. Web traffic which utilizes TCP on the transport level. In order to achieve an optimal level of realism, two differently sized real ISP topologies have been investigated, i.e. the AT&T-US network and the German B-WiN Research Network, with a state-of-the-art Web traffic model and for a broad spectrum of load situations. In all simulations carried out, AMP has shown to significantly outperform the traditional static routing strategies in terms of important metrics like average Web page response time and total TCP goodput in the network. Furthermore, AMP has proved to operate oscillation free in all investigated scenarios. Summarizing the performance evaluation efforts of AMP in the two different simulation environments, it is important to observe that the obtained simulation results are mutually consistent, demonstrating the high overall traffic engineering potential of AMP, as well as the algorithm's stability in the presence of a wide variety of traffic patterns.

Finally, a somewhat critical discussion of the status of traffic engineering in the current context of IP core networks has shed some light onto the close interplay between engineering for network resilience, bandwidth overprovisioning, and traffic engineering. The bandwidth overprovisioning approach today represents the most widely used paradigm in operating IP networks, as it simultaneously provides both loss-free packet transport (i.e. QoS), and guarantees a large amount of spare capacity for the case of equipment outages. Therefore, in networks with overprovisioned bandwidth, traffic engineering efforts make only limited sense, as normally the network should not experience any congestion by definition. However, as the entire world of networking will converge to IP as the common protocol in the coming years, other areas of application apart from core IP networks will become increasingly interesting for the optimization of load allocation, as bandwidth overprovsioning is not always a feasible approach. This is especially true for infrastructure-type wireless mesh networks which serve as backhaul for connecting remote base stations and clients to core IP networks, as their application will become important in the context of efforts towards provisioning high capacity wireless broadband access. Based upon the findings of this thesis, AMP should be very well applicable in those types of networks, which is further underlined by the real case of a commercial ISP which has identified AMP as the traffic engineering solution to be deployed in their operational network.

Of course, research in dynamic traffic engineering is far from coming to an end. As far as future research directions in the context of AMP are concerned, there are several issues which offer potential for theoretical modeling and algorithm refinement. An important

topic for further work is certainly the investigation of AMP stability. As developing a theoretical stability model of AMP based on general control engineering principles (e.g. through a Lyapunov-type approach) has been out of scope for this thesis, the demonstration of the algorithm's stability has been restricted to extensive simulations for a vast plethora of scenarios. However, of course simulations cannot replace the provision of a firm proof of AMP's deterministic and oscillation-free convergence towards a global fix point in traffic distribution in the presence of stable traffic patterns, and even more importantly, the absence of routing instabilities in the presence of adverse traffic patterns. Furthermore, as AMP is a heuristic algorithm, the mediation of the available information which is currently solved by a subtle interplay between averaging and maximizing relevant congestion information in each node and which determines the exact semantics of the backpressure mechanisms might bear some more potential for further fine-tuning for different types of network topologies and traffic patterns. And last, but not least, the Adaptive Multi-Path algorithm's performance evaluation, adaptation, and deployment scenarios for different novel areas of application, like e.g. wireless mesh networks, represent valuable opportunities for further investigations.

In the larger context of IP networking in the next years, there are several topics which will be of high importance for the entire research area of traffic engineering. Building a standardized and comprehensive measurement infrastructure certainly represents one of them, bearing the potential of enabling the development of traffic sensitive routing protocols for IP networks. Furthermore, such a standardized network monitoring solution would also facilitate other traffic engineering activities, like e.g. link weight optimization, as one of the most limiting practical obstacles for such offline schemes lies in the requirement of having a precise network-wide traffic matrix at hand in real-time. As far as inter-domain traffic engineering activities are concerned, to this end the degrees of freedom offered by standard BGP-4 routing protocol have been quite limited, so far resulting only in incremental fine-tuning of the existing mechanisms for a number of concrete scenarios. But with the enormous growth of BGP routing tables projected for the next years, which is mostly attributed to the increasingly popular multi-homing of smaller networks with large address prefixes, the design of a successor protocol is advancing to become an important research topic of its own, opening the potential of developing a new global traffic engineering solution. Finally, the requirements imposed upon future traffic engineering schemes will equally be determined by the emerging novel types of networking applications, their acceptance by the users, and by the development of more efficient switching and transport technologies. Together, these factors will decide upon the upcoming network provisioning paradigms, and thus crucially influence the status of traffic engineering in the mid-term future.

# Bibliography


[1] J. M. McQuillan and D. C. Walden, *The ARPANET Design Decisions*, J. Computer Networks, Vol. 1, September 1977.

[2] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*, Addison-Wesley, Reading, Massachusetts, USA, 1997.

[3] J. Doyle, *Routing TCP/IP – Volume 1*, Macmillan Technical Publishing, Indianapolis, Indiana, USA, 1998.

[4] R. E. Bellmann, *Dynamic Programming*, Princeton, New Jersey, USA, 1957.

[5] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton, New Jersey, USA, 1962.

[6] E. W. Dijkstra, *A Note on Two Problems in Connexion With Graphs*, in Numerische Mathematik, Vol. 1, Mathematisch Centrum, Amsterdam, The Netherlands, 1959, pp. 269–271.

[7] J. Moy, *RFC 2328 – OSPF Version 2*, IETF, September 1998.

[8] J. Moy, *OSPF: Anatomy of an Internet Routing Protocol*, Addison-Wesley, 1998.

[9] D. Oran, *RFC 1142 – OSI IS-IS Intra-Domain Routing Protocol*, IETF, February 1990.

[10] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, *RFC 3272 – Overview and Principles of Internet Traffic Engineering*, IETF, May 2002.

[11] E. Rosen, A. Viswanathan, and R. Callon, *RFC 3031 – Multiprotocol Label Switching Architecture*, IETF, September 2001.

[12] C. Villamizar, *OSPF Optimized Multipath (OSPF–OMP)*, IETF Internet Draft, 2002.

[13] I. Gojmerac, T. Ziegler, F. Ricciato, and P. Reichl, *Adaptive Multipath Routing for Dynamic Traffic Engineering*, in Proc. IEEE GLOBECOM'03, San Francisco, California, USA, December 2003, pp. 3058–3062.

[14] I. Gojmerac, T. Ziegler, and P. Reichl, *Adaptive Multipath Routing Based on Local Distribution of Link Load Information*, in Proc. 4th COST 263 International Workshop on Quality of Future Internet Services (QoFIS'03), Stockholm, Sweden, October 2003, pp. 122–131.

[15] I. Gojmerac, L. Jansen, T. Ziegler, and P. Reichl, *Feasibility Aspects of AMP Performance Evaluation in a Fluid Simulation Environment*, in Proc. 3rd MMBnet Workshop, Hamburg, Germany, September 2005.

[16] I. Gojmerac, L. Jansen, P. Reichl, and T. Ziegler, *A Simulation Study of Microscopic AMP Behavior*, in Proc. 4th Polish-German Teletraffic Symposium (PGTS'06), Wroclaw, Poland, September 2006, pp. 95–104.

[17] R. G. Gallager, *A Minimum Delay Routing Algorithm Using Distributed Computation*, IEEE Transactions on Communications, Vol. 25, January 1977, pp. 73–85.

[18] A. Tanenbaum, *Computer Networks, 3rd ed.*, Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1996.

[19] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1993.

[20] L. R. Ford, *Network Flow Theory, Report P-923*, Rand Corp., Santa Monica, California, USA, 1956.

[21] R. Bellman, *On a Routing Problem*, Quarterly of Applied Mathematics, 1958, pp. 87–90.

[22] R. W. Floyd, *Algorithm 97: Shortest Path*, Communications of ACM, 1962, p. 345.

[23] J. Moy, *OSPF – Complete Implementation*, Addison-Wesley, 1999.

[24] Y. Rekhter, T. Li, and S. Hares, *RFC 4271 – A Border Gateway Protocol 4 (BGP-4)*, IETF, January 2006.

[25] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, *Analysis of Link Failures in an IP Backbone*, in 2nd ACM SIGCOMM Workshop on Internet Measurment, Marseilles, France, November 2002, pp. 237–242.

[26] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot, *Feasibility of IP Restoration in a Tier 1 Backbone*, IEEE Network – Special Issue on Protection, Restoration and Disaster Recovery, Vol. 18, March/April 2004, pp. 13–19.

[27] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, *Characterization of Failures in an IP Backbone*, in IEEE INFOCOM'04, Hong Kong, China, March 2004, pp. 2307–2317.

[28] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, and C. Diot, *IGP Link Weight Assignment for Transient Link Failures*, in 18th International Teletraffic Congress, Berlin, Germany, August 2003.

[29] F. Giroire, A. Nucci, N. Taft, and C. Diot, *Increasing the Robustness of IP Backbones in the Absence of Optical Level Protection*, in IEEE INFOCOM'03, San Francisco, California, USA, March 2003, pp. 1–11.

[30] B. Fortz and M. Thorup, *Internet Traffic Engineering by Optimizing OSPF Weights*, in IEEE INFOCOM'00, Tel Aviv, Israel, March 2000, pp. 519–528.

[31] B. Fortz and M. Thorup, *Optimizing OSPF/IS-IS Weights in a Changing World*, IEEE Journal on Selected Areas in Communications, Vol. 20, May 2002, pp. 756–767.

[32] B. Fortz, J. Rexford, and M. Thorup, *Traffic Engineering with Traditional IP Routing Protocols*, IEEE Communications Magazine, Vol. 40, October 2002, pp. 118–124.

[33] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, *How to Model an Internetwork*, in IEEE INFOCOM'96, San Francisco, California, USA, March 1996, pp. 594–602.

[34] K. L. Calvert, M. Doar, and E. W. Zegura, *Modeling Internet Topology*, IEEE Communications Magazine, Vol. 35, June 1997, pp. 160–163.

[35] D. Harrington, R. Presuhn, and B. Wijnen, *RFC 3411 – An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*, IETF, December 2002.

[36] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, *Traffic Matrix Estimation: Existing Techniques and New Directions*, in ACM SIGCOMM'02, Pittsburgh, Pennsylvania, USA, August 2002, pp. 161–174.

[37] A. Nucci, R. Cruz, N. Taft, and C. Diot, *Design of IGP Link Weight Changes for Estimation of Traffic Matrices*, in IEEE INFOCOM'04, Hong Kong, China, March 2004, pp. 2341–2351.

[38] A. Feldmann, A. Greenberg, C. Lund, and N. Reingold, *Deriving Traffic Demands for Operational Networks: Methodology and Experience*, IEEE/ACM Transactions on Networking, Vol. 9, June 2001, pp. 265–279.

[39] M. Grossglauser and J. Rexford, *Passive Traffic Measurement for IP Operations, in The Internet as a Large-Scale Complex System*, Oxford University Press, 2005, pp. 91–120.

[40] R. Presuhn, J. Case, M. Rose, and S. Waldbusser, *RFC 3418 – Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*, IETF, December 2002.

[41] S. Bhattacharyya, C. Diot, J. Jetcheva, and N. Taft, *POP-Level and Access-Link-Level Traffic Dynamics in a Tier-1 POP*, in ACM SIGCOMM Internet Measurement Workshop, San Francisco, California, USA, November 2001, pp. 39–53.

[42] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, *NetScope: Traffic engineering for IP Networks*, IEEE Network Magazine, Special Issue on Internet Traffic Engineering, Vol. 14, March/April 2000, pp. 11–19.

[43] W. Haidegger, P. Reichl, S. Bessler, N. Jozefiak, and M. Teufel, *Ende-zu-Ende Performance Management in Heterogenen Hochgeschwindigkeitsnetzen*, PIK – Praxis der Informationsverarbeitung und Kommunikation, 2002.

[44] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, *RFC 2702 – Requirements for Traffic Engineering over MPLS*, IETF, September 1999.

[45] J. van Lunteren and T. Engbersen, *Fast and Scalable Packet Classification*, IEEE Journal on Selected Areas in Communications, Vol. 21, May 2003, pp. 560–571.

[46] J. van Lunteren, *Searching Very Large Routing Tables in Wide Embedded Memory*, in Proc. IEEE GLOBECOM'01, San Antonio, Texas, USA, November 2001, pp. 1615–1619.

[47] D. Mitra and K. G. Ramakrishnan, *A Case Study of Multiservice, Multipriority Traffic Engineering Design for Data Networks*, in IEEE GLOBECOM'99, Vol. 3, Rio de Janeiro, Brasil, December 1999, pp. 1615–1619.

[48] D. G. Luenberger, *Linear and Nonlinear Programming, 2nd ed.*, Addison-Wesley, Reading, Massachusetts, USA, 1984.

[49] D. Bertsekas and R. Gallager, *Data Networks, 2nd ed.*, Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1992.

[50] P. Aukia, M. Kodialam, P. V. Koppol, T. V. Lakshman, H. Sarin, and B. Suter, *RATES: A Server for MPLS Traffic Engineering*, IEEE Network Magazine, Vol. 14, March/April 2000, pp. 34–41.

[51] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry, *RFC 2748 – The COPS (Common Open Policy Service) Protocol*, IETF, January 2000.

[52] M. Kodialam and T. V. Lakshman, *Minimum Interference Routing with Applications to MPLS Traffic Engineering*, in IEEE INFOCOM'00, Tel Aviv, Israel, March 2000, pp. 884–893.

[53] K. Kar, M. Kodialam, and T. V. Lakshman, *Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications*, IEEE Journal on Selected Areas in Communications, Vol. 18, December 2000, pp. 2566–2579.

[54] B. Wang, X. Su, and C. L. P. Chen, *A New Bandwidth Guaranteed Routing Algorithm for MPLS Traffic Engineering*, in IEEE International Conference on Communications (ICC'02), New York City, New York, USA, April 2002, pp. 1001–1005.

[55] S. Butenweg, *Two Distributed Reactive MPLS Traffic Engineering Mechanisms for Throughput Optimization in Best Effort MPLS Networks*, in IEEE International Symposium on Computers and Communication (ISCC'03), Kemer, Turkey, June 2003, pp. 379–384.

[56] V. Sharma and F. Hellstrand, *RFC 3469 – Framework for Multi-Protocol Label Switching (MPLS)-based Recovery*, IETF, February 2003.

[57] C. Huang, V. Sharma, K. Owens, and S. Makam, *Building Reliable MPLS Networks Using a Path Protection Mechanism*, IEEE Communications Magazine, Vol. 40, March 2002, pp. 156–162.

[58] M. Menth, R. Martin, and U. Spoerlein, *Network Dimensioning for the Self-Protecting Multipath: A Performance Study*, in IEEE International Conference on Communications (ICC'06), Istanbul, Turkey, June 2006.

[59] J. M. McQuillan, G. Falk, and I. Richer, *A Review of the Development and Performance of the ARPANET Routing Algorithm*, IEEE Transactions on Communications, Vol. 26, December 1978, pp. 1802–1811.

[60] A. Khanna and J. Zinky, *The Revised ARPANET Routing Metric*, ACM SIGCOMM Computer Communication Review, Vol. 19, September 1989, pp. 45–56.

[61] J. M. McQuillan, G. Falk, and I. Richer, *The New Routing Algorithm for the ARPANET*, IEEE Transactions on Communications, Vol. 28, May 1980, pp. 711–719.

[62] C. L. Hedrick, *An Introduction to IGRP*, Technical Report, Rutgers University, New Jersey, USA, August 1991.

[63] S. Low and P. Varaiya, *Stability of a Class of Dynamic Routing Protocols (IGRP)*, in IEEE INFOCOM'93, San Francisco, California, USA, March 1993, pp. 610–616.

[64] R. J. Gibbens, *Dynamic Routing in Circuit-Switched Networks: The Dynamic Alternative Routing Strategy*, Ph.D. Thesis, University of Cambridge, United Kingdom, 1988.

[65] R. J. Gibbens, F. P. Kelly, and P. B. Key, *Dynamic Alternative Routing – Modeling and Behavior*, in 12th International Teletraffic Congress, Torino, Italy, June 1988.

[66] R. R. Stacey and D. J. Songhurst, *Dynamic Alternative Routing in the British Telecom Trunk Network*, in International Switching Symposium (ISS'87), Phoenix, Arizona, USA, March 1987.

[67] F. P. Kelly, *Bounds on the Performance of Dynamic Routing Strategies for Highly Connected Networks*, Mathematics of Operations Research, Vol. 19, 1994, pp. 1–20.

[68] R. J. Gibbens and P. Reichl, *Performance Bounds Applied to Loss Networks*, Complex Stochastic Systems and Engineering (ed. D. M. Titterington), 1995, pp. 267–279.

[69] P. Key, *Optimal Control and Trunk Reservation in Loss Networks*, Probability in the Engineering and Informational Sciences, Vol. 4, 1990, pp. 203–242.

[70] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot, *An Approach to Alleviate Link Overload as Observed on an IP Backbone*, in IEEE INFOCOM'03, San Francisco, California, USA, March 2003, pp. 406–416.

[71] C. Villamizar and T. Li, *IS-IS Optimized Multipath (ISIS–OMP)*, IETF Internet Draft, 2002.

[72] Z. Cao, Z. Wang, and E. Zegura, *Performance of Hashing-based Schemes for Internet Load Balancing*, in IEEE INFOCOM'00, Tel Aviv, Israel, March 2000, pp. 332–341.

[73] M. Allman, V. Paxson, and W. Stevens, *RFC 2581 – TCP Congestion Control*, IETF, January 1997.

[74] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, *The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm*, ACM Computer Communications Review, Vol. 27, July 1997, pp. 67–82.

[75] R. Coltun, *RFC 2370 – The OSPF Opaque LSA Option*, IETF, July 1998.

[76] G. M. Schneider and T. Nemeth, *A Simulation Study of the OSPF-OMP Routing Algorithm*, J. Computer Networks, Vol. 39, July 2002, pp. 457–468.

[77] L. Kleinrock, *Queueing Systems, Vol. 1 – Theory*, Wiley Interscience, New York, USA, 1975.

[78] J. J. Garcia-Luna-Aceves and S. Vutukury, *A Practical Approach to Minimizing Delays in Internet Routing*, in IEEE International Conference on Communications (ICC'99), Vancouver, Canada, June 1999, pp. 479–483.

[79] A. Segall, *The Modeling of Adaptive Routing in Data-Communication Networks*, IEEE Transactions on Communications, Vol. 25, January 1977, pp. 85–95.

[80] A. Segall and M. Sidi, *A Failsafe Distributed Protocol for Minimum Delay Routing*, IEEE Transactions on Communications, Vol. 29, May 1981, pp. 689–695.

[81] P. M. A. Segall, *A Failsafe Distributed Routing Protocol*, IEEE Transactions on Communications, Vol. 27, September 1979, pp. 1280–1287.

[82] B. M. Waxman, *Routing of Multipoint Connections*, IEEE Journal on Selected Areas in Communications, Vol. 6, December 1988, pp. 1617–1622.

[83] L. Jansen, I. Gojmerac, M. Menth, P. Reichl, and P. Tran-Gia, *An Algorithmic Framework for Discrete-Time Flow-Level Simulation of Data Networks*, in 20th International Teletraffic Congress, Ottawa, Canada, June 2007, pp. 865–877.

[84] L. Jansen, *Fluid-Simulatoren zur Beschleunigung von Leistungsuntersuchungen in Paketnetzen*, Master Thesis, University of Würzburg, Germany, 2006.

[85] D. Mitra, D. Anick, and M. M. Sondhi, *Stochastic Theory of a Data Handling System with Multiple Sources*, Bell Systems Technical Journal, Vol. 61, 1982, pp. 1871–1894.

[86] B. Liu, D. R. Figueiredo, Y. Guo, J. F. Kursoe, and D. F. Towsley, *A Study of Networks Simulation Efficiency: Fluid Simulation vs. Packet-level Simulation*, in IEEE INFOCOM'01, Anchorage, Alaska, USA, April 2001, pp. 1244–1253.

[87] B. Liu, Y. Guo, J. Kursoe, D. Towsley, and W. Gong, *Fluid Simulation of Large Scale Networks: Issues and Tradeoffs*, Technical Report UM-CS-1999-038, University of Massachusetts, MA, USA, 1999.

[88] A. Yan and W.-B. Gong, *Time-Driven Fluid Simulation for High-Speed Networks*, IEEE Transactions on Information Theory, Vol. 45, July 1999, pp. 1588–1599.

[89] Y. Guo, W. Gong, and D. Towsley, *Time-Stepped Hybrid Simulation (TSHS) for Large Scale Networks*, in IEEE INFOCOM'00, Tel Aviv, Israel, March 2000, pp. 441–450.

[90] Y. Liu, F. L. Presti, V. Misra, D. Towsley, and Y. Gu, *Fluid Models and Solutions for Large-Scale IP Networks*, in ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, San Diego, California, USA, June 2003, pp. 91–101.

[91] M. B. Doar, *A Better Model for Generating Test Networks*, in IEEE GLOBECOM'96, London, United Kingdom, November 1996, pp. 86–93.

[92] L. Li, D. Alderson, W. Willinger, and J. Doyle, *A First-Principles Approach to Understanding the Internet's Router-level Topology*, ACM SIGCOMM Computer Communication Review, Vol. 34, October 2004, pp. 3–14.

[93] M. Faloutsos, P. Faloutsos, and C. Faloutsos, *On Power-law Relationships of the Internet Topology*, ACM SIGCOMM Computer Communication Review, Vol. 29, October 1999, pp. 251–262.

[94] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos, *Power Laws and the AS-level Internet Topology*, IEEE/ACM Transactions on Networking, Vol. 11, August 2003, pp. 514–524.

[95] A. Medina, A. Lakhina, I. Matta, and J. Byers, *BRITE: An Approach to Universal Topology Generation*, in Proc. IEEE MASCOT'01, Cincinnati, Ohio, USA, August 2001, pp. 346–353.

[96] A.-L. Barabási and R. Albert, *Emergence of Scaling in Random Networks*, Science, Vol. 286, October 1999, pp. 509–512.

[97] A. Nucci, A. Sridharan, and N. Taft, *The Problem of Synthetically Generating IP Traffic Matrices: Initial Recommendations*, ACM SIGCOMM Computer Communication Review, Vol. 35, July 2005, pp. 19–31.

[98] S. Zimmermann, *Congestion Pricing as Scalable, Efficient and Stable Congestion Control for Future IP Networks*, VDE Verlag, Berlin, Germany, 2005.

[99] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, *RFC 2309 – Recommendations on Queue Management and Congestion Avoidance in the Internet*, IETF, April 1998.

[100] V. Misra, W.-B. Gong, and D. F. Towsley, *Fluid-Based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED*, ACM SIGCOMM Computer Communication Review, Vol. 30, October 2000, pp. 151–160.

[101] M. Ajmone Marsan, M. Garetto, P. Giaconne, E. Leonardi, E. Schiattarella, and A. Tarello, *Using Partial Differential Equations to Model TCP Mice and Elephants in Large IP Networks*, IEEE/ACM Transactions on Networking, Vol. 13, December 2005, pp. 1289–1301.

[102] E. Hasenleithner and T. Ziegler, *Comparison of Simulation and Measurement using State of the Art Web Traffic Models*, in IEEE Symposium on Computers and Communications (ISCC'03), Kemer, Antalya, Turkey, June 2003, pp. 1172–1177.

[103] E. J. Anderson, T. E. Anderson, S. D. Gribble, A. R. Karlin, and D. J. Wetherall, *Towards Efficient and Robust Adaptive Routing*, in ACM SIGCOMM'02 (Submission), Pittsburgh, Pennsylvania, USA, August 2002.

[104] P. Barford and M. E. Crovella, *Generating Representative Web Workloads for Network and Server Performance Evaluation*, in ACM SIGMETRICS'98, Madison, Wisconsin, USA, June 1998, pp. 151–160.

[105] K. Below, C. Schwill, and U. Killat, *Erhöhung des Nutzungsgrades eines ATM Netzes für den Wissenschaftsbereich (ERNANI)*, Technical Report, Dept. Communication Networks, Technical University Hamburg-Harburg, September 2001.

[106] S. D'Antonio, M. D'Arienzo, M. Esposito, S. P. Romano, and G. Ventre, *Managing Service Level Agreements in Premium IP Networks: A Business-Oriented Approach*, J. Computer Networks, Vol. 46, December 2004, pp. 853–866.

[107] I. Gojmerac, F. Hammer, F. Ricciato, H. T. Tran, and T. Ziegler, *Scalable QoS: State-of-the-Art Architectural Solutions and Developments*, Technical Report FTW-TR-2004-003, FTW, April 2004.

[108] M. Menth, *Efficient Admission Control and Routing for Resilient Communication Networks*, Ph.D. Thesis, University of Würzburg, Germany, 2004.

[109] C. Diot, *A Tier-1 IP Backbone Network, Architecture, Performance*, Tutorial at ICNP'02, Paris, France, 2002.

[110] K. G. Coffman and A. M. Odlyzko, *Internet Growth: Is There a "Moore's Law" for Data Traffic?*, Technical Report, AT&T Labs, 11 April 2000.

[111] S. Iyer, R. Zhang, and N. McKeown, *Routers with a Single Stage of Buffering*, in ACM SIGCOMM'02, Pittsburgh, Pennsylvania, USA, August 2002, pp. 251–264.

[112] C. Courcoubetis and R. Weber, *Pricing Communication Networks. Economics, Technology and Modelling*, Wiley Interscience, New York, USA, 2003.

[113] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, *Resilient Overlay Networks*, in ACM SOSP'01, Banff, Canada, October 2001, pp. 131–145.

[114] Z. Li and P. Mohapatra, *QRON: QoS-Aware Routing in Overlay Networks*, IEEE Journal on Selected Areas in Communications, Vol. 22, January 2004, pp. 29–40.

[115] A. Nakao, L. Peterson, and A. Bavier, *Scalable Routing Overlay Networks*, ACM SIGOPS Operating Systems Review, Vol. 40, January 2006, pp. 49–61.

[116] D. Doval and D. O'Mahony, *Overlay Networks – A Scalable Alternative for P2P*, IEEE Internet Computing, Vol. 7, July/August 2003, pp. 79–82.

[117] B. Gusmão Rocha, V. Almeida, and D. Guedes, *Increasing QoS in Selfish Overlay Networks*, IEEE Internet Computing, Vol. 10, May/June 2006, pp. 24–31.

[118] I. F. Akyldiz and X. Wang, *A Survey on Wireless Mesh Networks*, IEEE Radio Communications, Vol. 43, September 2005, pp. 23–30.

[119] C. Perkins, E. Belding-Royer, and S. Das, *RFC 3561 – Ad hoc On-Demand Distance Vector (AODV) Routing*, IETF, July 2003.

[120] D. R. Johnson, D. A. Maltz, and Y.-C. Hu, *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, IETF Internet Draft, 19 July 2004.

[121] R. P. Karrer, I. Matyasovszki, A. Botta, and A. Pescapé, *Experimental Evaluation and Characterization of the Magnets Wireless Backbone*, in ACM WiNTECH'06, Los Angeles, California, USA, September 2006, pp. 26–33.

[122] M. Jaseemuddin, A. Esmailpour, A. Alwan, and O. Bazan, *Integrated Routing System for Wireless Mesh Networks*, in Canadian Conference on Electrical and Computer Engineering, Ottawa, Canada, May 2006, pp. 1003 – 1007.

[123] B. Schmidt, *Dynamisches Traffic Engineering in einem drahtlosen IP-Backbone*, Master Thesis, Humboldt University of Berlin, Germany, 2005.

[124] S. Bhatnagar, S. Ganguly, and R. Izmailov, *Design of IEEE 802.16-based Multi-hop Wireless Backhaul Networks – Invited Paper*, in ACM AccessNets'06, Athens, Greece, September 2006.

[125] H. Viswanathan and S. Mukherjee, *Throughput-Range Tradeoff of Wireless Mesh Backhaul Networks*, IEEE Journal on Selected Areas in Communications, Vol. 24, March 2006, pp. 593–602.

[126] M. Harteneck, M. Boloorian, S. Georgoulis, and R. Tanner, *Throughput Measurements of HSDPA 14 Mbit/s Terminal*, IEE Electronics Letters, Vol. 41, March 2005, pp. 425–427.

[127] E. Dahlman, H. Ekström, A. Furuskär, Y. Jading, J. Karlsson, M. Lundevall, and S. Parkvall, *The 3G Long-Term Evolution – Radio Interface Concepts and Performance Evaluation*, in IEEE 63rd Vehicular Technology Conference (VTC 2006), Melbourne, Australia, May 2006, pp. 137 – 141.

[128] J. E. O'Neill, *The Role of ARPA in the Development of the ARPANET, 1961-1972*, IEEE Annals of the History of Computing, Vol. 17, 1995, pp. 76–81.

[129] D. D. Clark, *The Design Philosophy of the DARPA Internet Protocols*, ACM SIGCOMM Computer Communication Review, Vol. 18, 1988, pp. 106–114.

# List of Internet Links


[www-1] NS-2, *The Network Simulator*, http://www.isi.edu/nsnam/ns/, April 2007.

[www-2] Cisco Systems, *Configuring OSPF – Cisco IOS IP and IP Routing Configuration Guide*, http://www.cisco.com/, April 2007.

[www-3] R. Mortier, *Python Routeing Toolkit (PyRT)*, http://ipmon.sprint.com/pyrt/, April 2007.

[www-4] Cisco Systems, *Cisco IOS NetFlow*, http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html, April 2007.

[www-5] Internet Engineering Task Force, *IETF*, http://www.ietf.org/, April 2007.

[www-6] IETF, *OMP Simulations*, http://www.faster-light.net/omp/simulations.html, April 2007.

[www-7] H. Perros, *Computer Simulation Techniques*, Computer Science Department, North Carolina State University, Raleigh, North Carolina, USA, http://www.csc.ncsu.edu/faculty/perros/simulation.pdf, April 2007.

[www-8] Georgia Institute of Technology, *Georgia Tech Internetwork Topology Models GT-ITM*, http://www.cc.gatech.edu/projects/gtitm/, April 2007.

[www-9] OPNET Technologies, *OPNET Modeler*, http://www.opnet.com/, April 2007.

[www-10] OMNeT++ Project, *OMNeT++ Simulation Environment*, http://www.omnetpp.org/, April 2007.

[www-11] Martin Dodge, *An Atlas of Cyberspaces*, http://www.cybergeography.org/atlas/topology.html, April 2007.

[www-12] Boston University, Computer Science Department, *BRITE Topology Generator*, http://www.cs.bu.edu/brite/, April 2007.

[www-13] IEEE, *IEEE 802.11 Standard*, http://www.ieee802.org/11/, April 2007.

[www-14] IEEE, *IEEE 802.16 Standard*, http://www.ieee802.org/16/, April 2007.

[www-15] Deutsche Telekom Laboratories, *Magnets Backbone Website*, http://www.deutsche-telekom-laboratories.de/∼karrer/magnets.html, April 2007.

# Abbreviations

| | |
|---|---|
| **AMP** | Adaptive Multi-Path routing |
| **BGP** | Border Gateway Protocol |
| **COPS** | Common Open Policy Service |
| **CRC** | Cyclic Redundancy Check |
| **DAR** | Dynamic Alternative Routing |
| **ECMP** | Equal Cost Multi-Path routing |
| **EIGRP** | Enhanced Interior Gateway Routing Protocol |
| **FEC** | Forwarding Equivalency Class |
| **IGRP** | Interior Gateway Routing Protocol |
| **IP** | Internet Protocol |
| **IS-IS** | Intermediate System to Intermediate System |
| **ISP** | Internet Service Provider |
| **LSP** | Label Switched Path |
| **LTE** | Long Term Evolution |
| **MIB** | Management Information Base |
| **MIRA** | Minimum Interference Routing Algorithm |
| **MPLS** | Multi-Protocol Label Switching |
| **OMP** | Optimized Multi-Path routing |
| **OSPF** | Open Shortest Path First |
| **QoS** | Quality of Service |
| **RATES** | Routing and Traffic Engineering Server |
| **RON** | Resilient Overlay Network |
| **SNMP** | Simple Network Management Protocol |
| **WiMAX** | Worldwide Interoperability for Microwave Access |
| **WLAN** | Wireless Local Area Network |