# Traffic Engineering and Quality of Service in the Internet

Cover: an arc map showing the world-wide internet traffic, by Stephen G. Eick and his colleagues, Bell Laboratories, 1996.

# Traffic Engineering and Quality of Service in the Internet

**Proefschrift**

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op maandag 30 maart 2009 om 10.00 uur

door

Bingjie FU

Master of Philosophy University of Bath, Bath, UK
geboren te Daqing, Heilongjiang Province, China.

ISBN 978-90-79746-06-4

This research was funded by the Technology Foundation STW.

Keywords: Traffic Engineering, Quality of Service

to my beloved family

# Contents

# Chapter 1

# Introduction

The need of information transmission over a distance was already realized thousands of years ago. At that time, smoke signals were used to send some predetermined patterns of messages. Later on, the postal system and pigeon post were used to carry messages with more flexible contents. Now, in the information age, remote communication involving the use of electronic transmitters such as the telephone, television, radio or computer, makes the transmission of information faster, and brings ease and fun to users. Internet, the largest decentralized computer system in the world, plays a critical role in nowadays information communication.

Originally, the Internet was a research network built to provide reachability and robustness. Due to the fast development of computer technology, the cost of producing a computer became low enough to allow home computers and personal computers to become widespread. Letting the owners of computers communicate with each other became a natural trend. The only way to ensure global reachability of a large system (e.g., in the number of hosts) such as the Internet is to group hosts into separate networks and to link these networks together. Each separate network needs physical infrastructure to form the backbone and man power to maintain, which results in the appearance of Internet Service Providers (ISPs) [44]. Figure 1.1 from [44] shows the number of unique Autonomous System (AS) numbers advertised in the Border Gateway Protocol (BGP)'s routing table since year $1997$. Till today the Internet evolved to a large distributed system composed of more than $25,000$ ASs operated by different ISPs. These ASs connect to each other to get global reachability. The Internet's AS-level topology follows a hierarchical structure: a few large ASs form the core; regional providers with smaller backbones connect the edge with the core; and small stub networks are attached on the edge.

With the increasing use of Internet for commercial purposes, more stringent applications have been deployed, like Voice over IP (VoIP), Video on Demand (VOD), and Virtual Private Networks (VPN). Companies focusing on providing such applications came on to the stage. VoIP providers like Skype allow users to make telephone calls over the Internet; Content Distribution Networks (CDNs) like Akamai deliver contents such as audio and video media objects to users; many ISPs provide VPN service as one of their main services; *etc*. These ser-

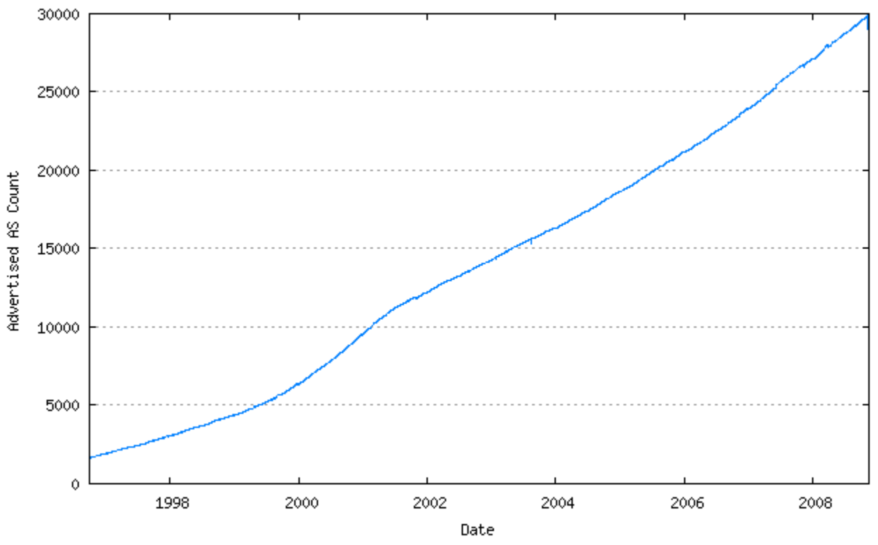


Figure 1.1: Number of advertised ASs along time [44].

vices require specific QoS to be ensured or guaranteed by the network. VoIP applications are sensitive to end-to-end delay; CDNs would use routes with small end-to-end delays and good connections to achieve faster download times and less vulnerability to network congestion or outages; VPNs may require bandwidth to be reserved and connections to be protected. ISPs are driven to rely on traffic engineering [9] to better control the flow of IP packets, to provide a good service and to fulfill their network management goals. Functionalities enabling Quality of Service (QoS), such as Multi Protocol Label Switching (MPLS) [80], Traffic Engineering (TE) extensions to Open Shortest Path First (OSPF) [63] or Intermediate System-Intermediate System (IS-IS) [20] and RSVP-TE, need to be implemented. MPLS is a protocol agnostic data-carrying mechanism. In MPLS networks, labels are attached to data packets, and are used to guide the forwarding of these data packets. The paths followed by these data packets are called Label Switched Paths (LSPs). Resource Reservation Protocol-Traffic Engineering (RSVP-TE) is an extension of the RSVP protocol for traffic engineering. It makes possible to reserve resources to guarantee the QoS to be experienced along time, and can serve as the signaling protocol for establishing MPLS LSPs taking into account network state information like available bandwidth. Some services may require multiple ASs to cooperate to provide consistent guarantees from the source to the destination across ASs.

Traffic engineering and QoS in the two network management categories, namely intradomain and interdomain, face different tasks and challenges. Intradomain traffic engineering and QoS, i.e., traffic engineering and QoS within a single AS/network, concerns only the ISP in charge of this AS/network; while interdomain traffic engineering and QoS, where multiple ASs/networks are involved, require specific regulations, such as business agreements between ASs.

## 1.1 Traffic Engineering and QoS: Intradomain

In pure IP networks, intradomain traffic engineering can be implemented by changing the intradomain routing protocol (called Interior Gateway Protocol or IGP) weights [36]. Many ASs use OSPF/IS-IS as their intradomain routing protocol. These protocols select shortest paths based on static link weights. These IGP link costs reflect the desirability of a link to be selected to carry traffic inside the AS. The link costs can be computed off-line according to the offered traffic, the provisioned network capacity, and the specific objective desired by the network administrator [37], or simply by following Cisco's guideline of IGP costs proportional to the inverse of the link capacity [24].

With the label-based forwarding mechanisms such as MPLS and Generalized Multi Protocol Label Switching (GMPLS) [57], intradomain traffic engineering can be realized more flexibly, e.g., for each pair of source destination routers, a specific route (not forcibly the IGP shortest path) can be assigned to forward the traffic. Moreover, MPLS and/or GMPLS make per-flow path selection with service guarantees possible. Traffic demands with QoS requirements can be accommodated with Label Switched Paths (LSPs) following the routes computed by Traffic Engineering/QoS routing algorithms according to the network status. The label-based forwarding mechanisms provide an opportunity to control traffic in a finer-grained way than by changing IGP weights. The per-flow path selection with resource reservation makes routing with QoS guarantees possible. When the traffic demands require fast restoration in order to survive from failures of network elements (links/nodes), MPLS and/or GMPLS can also be used to establish a pair of link/node disjoint LSPs, one of which acts as the working LSP, and the other acts as the backup LSP. In this part of the thesis on intradomain traffic engineering, we do not dig into this problem, but rather focus on the intradomain traffic engineering done for traffic demands with bandwidth/capacity requirements.

The computation of the exact routes that will be used to set up the LSPs with QoS guarantees may be done off-line or on-line. The computation is done off-line when paths do not need to adapt to traffic dynamics. On-line computation, on the other hand, is required when the routing plans must adapt to the changing network conditions, which is the case when strict QoS guarantee is required and feasible paths are wanted as long as there are some. For intradomain QoS routing, an intradomain QoS routing protocol is expected to determine, update and distribute the dynamically changing network state. The intradomain QoS routing protocol possesses a functionality to give each router a consistent view of the network and its resources by using a Link-State Update Policy (LSUP). The resource information varies rapidly compared to the infrequent changes in network topology (such as the breakdown of a link or router). Based on the resource information, an intradomain QoS algorithm will compute a path which will try to accommodate the QoS requirements of a particular flow. On-line techniques leverage the capability of label-based forwarding mechanisms to choose paths inside the network to better utilize the available resources in the network. On-line algorithms select for each new connection request a path with enough resources/QoS guarantees, based on its knowledge of the network state.

When multiple QoS constraints (e.g., a request requires QoS guarantees in both bandwidth and end-to-end delay) need to be taken into account, the end-to-end paths must guarantee all the QoS requirements and be computed by multi-constraint QoS routing algorithms (cf. [52] for a survey).

## 1.2 Interdomain policy routing and interdomain QoS

To get reachability to the whole Internet, ASs connect to each other through peering relationships. When two ASs are interested to connect to each other, they negotiate and sign a contract that will rule the conditions under which the connectivity between them is to be used, such as how traffic will be exchanged, how the billing will be made, *etc*. ASs use BGP as the interdomain routing protocol. The role of BGP is to propagate reachability information across the whole Internet. The ways in which routing information is exchanged between BGP routers and in which paths are selected by BGP routers, both depend on routing policies [38]. Routing policies [18] implement the business objectives of each ISP by modifying the flow of routing information and the path selection made by BGP.

The amount of traffic that flows on each interdomain connection will lead the money flowing from one AS to another. Thus, besides engineering the packets flowing inside their networks (intradomain traffic engineering), ISPs also need to take the flow of their interdomain traffic into account. ASs engineer their interdomain traffic based on their business interests. For some ASs, reachability is the most important concern and their purpose of interdomain traffic engineering is to get traffic forwarded with the lowest possible cost. For ASs providing specific services such as those content distribution networks, end-to-end performance of the selected routes will influence the quality of the service they deliver, and thus, is crucial for these ASs. They tend to use the forwarding routes with good end-to-end performance via interdomain traffic engineering.

With the configurable parameters involved in the BGP path selection process, ASs can influence the incoming and outgoing traffic by tuning the BGP configurations and route advertisements on their BGP routers. For practical interdomain traffic engineering techniques using BGP, see [78] as a survey. From routing data (i.e., forwarding paths selected by BGP routers), one can observe the consequence of these configurations.

ASs are passive in the sense that they can only use those paths told by their neighbors. The forwarding paths selected by a BGP router are constrained by the routing diversity of this BGP router. As the connections between ASs are coupled with the corresponding business agreements, tasks such as routing and resource reservation across ASs need to obey the limitations defined in the business agreements. In case of dissatisfaction on the routes propagated by a neighbor, one way to obtain more desirable ones is to renegotiate with this neighbor and modify the business agreements in between.

Interdomain QoS, which involves multiple ASs, also needs to follow the business agreements between ASs. When strict QoS is required across multiple ASs, firstly, a route that can provide the QoS guarantees needs to be found. Then, network resources need to be reserved in the involved ASs to guarantee the QoS on the end-to-end path. Both steps need cooperation between ASs. With the extensions to RSVP-TE proposed in [72], long-term Label Switched Paths (LSPs) with resource reservation are possible to be arranged manually across ASs under the condition that the ASs involved allow the resource reservation. While to know the end-to-end path that can guarantee the QoS requirements is not straight forward in the current interdomain environment, the Internet still lacks of a protocol to enable automatic interdomain QoS routing and resource reservation.

Both the research and the engineering communities are working on proposals to make interdomain QoS a reality. Some works [15, 106] have proposed QoS extensions to BGP. The idea is to use BGP to piggyback QoS information since BGP already propagates reachability information across the Internet. The Path Computation Element (PCE) Working Group (WG) in the Internet Engineering Task Force (IETF) focuses on the computation of an interdomain route that can meet the QoS requirements. The PCE architecture requires each domain to be responsible for the computation of path segments in their own networks. Finally, the combination of path segments that is able to guarantee the QoS requirements will be used to set up the interdomain LSP.

## 1.3 Content and contribution of the Thesis

In this thesis, we try to answer the following questions:

- In which situations are traffic engineering/ QoS routing algorithms helpful in efficiently using network resources?
- How ISPs/ASs carry out their interdomain traffic engineering in reality?
- Is it possible to have QoS across the Internet? If yes, what existing functionalities could be used to fulfil this goal? And what functionalities are still missing?

The main body of this thesis is structured into two parts: Part I and Part II that focus on the traffic engineering and QoS in the intradomain and interdomain scenarios respectively. The last chapter of this thesis Chapter 7 high-lights the conclusions drawn from the thesis.

### 1.3.1 Part I

This part of the thesis focuses on the performance of different intradomain traffic engineering mechanisms under different network and traffic scenarios. We aim to identify the scenarios in which on-line traffic engineering and/or QoS routing show advantages in efficiently using the network resources.

**Chapter 2** A Link-State Update Policy (LSUP) has the task to distribute information regarding the network resources, and is therefore considered to be an integral part of future Quality of Service (QoS) routing protocols. Unfortunately, the high dynamics in available resources will result in high deployment costs (e.g., in signalling). In Chapter 2, we re-examine whether the gain in network performance, which is expected under the deployment of LSUPs, will outweigh its investment and complexity costs. The observation in our work, without considering the amount of traffic overhead induced by the link state updates, shows that: 1) under low network load, the performance difference (between routing with and without link state information) stays small; and 2) under high network load, routing without link state updates performs close to or even better than routing with exact link state information. Our results argue that the investment in link state updates may not be cost efficient and that a simple shortest-hop framework is likely to suffice.

**Chapter 3** The evaluation of traffic engineering algorithms is usually carried out using some specific network(s), traffic pattern(s) and traffic engineering objective(s). The behavior of a traffic engineering algorithm is a consequence of the interactions among the network, the traffic demand and the algorithm itself. In Chapter 3, we studied the behavior of on-line traffic engineering algorithms under different scenarios with a systematic approach. We define three distinct load regimes (low, medium, and high) that correspond to different behaviors of the on-line traffic engineering algorithms. We show the impact of network design aspects on the performance of on-line traffic engineering algorithms using synthetic network topologies and traffic demands.

### 1.3.2 Part II

Our contribution on traffic engineering and QoS in the interdomain category includes: 1) investigating, based on publicly available routing data, how ASs do their interdomain traffic engineering; and 2) we show the possibility of having interdomain QoS a reality, identify the existing functional components that can be used to fulfil this goal, and point out the missing functional components that are needed.

**Chapter 4** In Chapter 4, we introduce how interdomain routing works, including the business relationships coupled with the connections between ASs, the routing protocol, and some general rules that constrain route advertisement and selection. Further, we briefly explain how interdomain traffic engineering can be done by tuning parameters involved in route advertisement and selection.

**Chapter 5** ASs do traffic engineering according to their business interests. Thanks to the public routing data available from projects like RIPE NCC [1] and RouteViews [81], we can better understand how interdomain routing is done by ASs via analyzing these data. And hopefully, it will lead to a model which can reproduce the interdomain routing done by each AS as seen from the observations. To have such a model, both the AS-level connectivity and the routing policies between ASs need to be modeled. In Chapter 5, we provide an AS-level topology model that enables the propagation of all paths observed in the routing data, and introduce a new concept: next-hop atoms that capture an AS's preference on using different sets of neighboring ASs for its best routes.

**Chapter 6** As QoS-sensitive applications and services evolve, end-to-end QoS across the ASs remains a challenge for the next generation Internet. Pushing QoS across interdomain boundaries appears to be the bottleneck to provide end-to-end QoS guarantees across domains. In Chapter 6, we review the existing functional components that may serve to establish interdomain LSPs with QoS guarantees, and identify the necessary functional components that are still missing. We show that inter-AS QoS is not out of reach, but that more work needs to be done in specific areas, especially concerning the propagation of QoS information to guide the path computation.

# Part I

# Intradomain Traffic Engineering and QoS

# Chapter 2

# To update network state or not?

Fu, B., F.A. Kuipers, and P. Van Mieghem, 2008, "*To Update Network State or Not?*", in Proc. of QoS-IP 2008, the Fourth International Workshop on QoS in Multiservice IP Networks, Venezia, Italy, February 13-15.

The study of intradomain QoS routing has mainly focused on algorithms that can find paths based on multiple constraints (cf. [52] for a survey). Those algorithms assume that the knowledge on the state of the network is provided by an intradomain QoS routing protocol. Unfortunately, the development of an intradomain QoS routing protocol has received much less attention, most probably due to the complexity of handling network dynamics.

The complexity of QoS routing in comparison to "simple" best-effort routing may result in high costs. Internet Service Providers (ISPs) may not be willing to change their current non-QoS-aware systems if the performance gain of QoS-aware networking is not substantial enough compared to the additional cost. Beside the monetary cost involved in implementing QoS technology, cost also relates to computation and protocol overhead. The computation overhead is bounded by the worst-case complexity of the routing algorithm, while the protocol overhead is caused by the flooding and updating of resource information.

Most of the proposed link-state update policies (LSUPs) consider a trade-off between the protocol overhead and the accuracy of the resource information (see [84]). To reduce the protocol overhead, the number of updates needs to be limited, and consequently, not all resource changes are advertised. This leads to the problem of QoS routing with stale resource information.

Stale resource information may affect routing in the following ways:

- A feasible path cannot be found by the algorithm although one exists.
- A path is found by the routing algorithm but rejected during the set-up process, because not all links along the path can provide the required capacity.

- A non-optimal path is found while there exist other paths, which are more suitable.

It is difficult to indicate how much cost is acceptable in order to gain a certain improvement in network performance, as this may differ per ISP. In this chapter, the profits of using link-state routing systems are estimated indirectly via the comparison of two extreme strategies: (A) routing with exact resource information, and (B) routing without resource information, i.e., without link-state updates (LSUs).

The rest of this chapter is constructed as follows. In Sections 2.1 and 2.2, we discuss the related work on proposed LSUPs and the impact of stale resource information on network performance respectively. In Section 2.3, we motivate our study and explain the two extreme strategies in which we are interested. Examples of performance evaluation are given in Section 2.4. Section 2.5 presents the simulation models. In Section 2.6, we show the performance in selected situations and discuss the simulation results. Finally, we present our conclusions in Section 2.7.

## 2.1 LSUPs

For intradomain QoS routing, many LSUPs have been proposed.

**Periodic LSUP** The periodic LSUP distributes the resources information through the network periodically. It is, for instance, employed by OSPF (Open Shortest Path First) [63] to update the topology information. The periodic LSUP is easy to implement and the update rate is fixed once the period is set. Hence, the periodic LSUP is not coupled to traffic dynamics. Many simulations (e.g. in [83]) suggest that, with relatively small LSU periods, QoS routing performs very well. Naturally, the use of small periods comes at the expense of a large overhead.

**Trigger-based LSUPs** Due to the shortcomings of the periodic LSUP, the trigger-based LSUPs were proposed to better catch the dynamics in link state [6, 54, 84]. In the following we assume that the available bandwidth is our link-state metric. We can distinguish between class-based LSUPs and threshold-based LSUPs. The inaccuracy introduced by the trigger-based LSUPs can be bounded (from above ($c_u$) and from below ($c_l$)). Safety-based routing proposed in [5] employs this property of trigger based LSUPs.

*Class-based LSUPs*

Class-based LSUPs divide the link capacity into several classes. As long as the available bandwidth resides within a class, no updates are triggered. Once the available bandwidth crosses one of the boundaries, the crossed boundary's value is advertised to the whole network. The principles for defining the classes may vary according to different aims. Equal class and exponential class are the most common ones.

Equal class-based LSUPs use only one parameter $N_{Eq}$, which defines the number of classes into which each link's capacity $C_{Full}$ will be divided. If the last updated value currently known by the other network elements is $\frac{2C_{Full}}{N_{Eq}}$, then the actual available bandwidth lies in the range $\left[c_l = \frac{C_{Full}}{N_{Eq}}, c_u = \frac{3C_{Full}}{N_{Eq}}\right]$.

Exponential class-based LSUPs, use 2 parameters to define their classes: $c_{Ex}$ ($c_{Ex} < 1$), the factor used to decide the size of the base class; and $f$ ($f > 1$), the growth factor to decide the relative size of each class. The size of the base class, $C_{BaseClass} = C_{Full} \cdot c_{Ex}$. With $c_{Ex}$ and $f$, we can get a set of classes with boundaries at

$$C_{Full} \cdot \left[0, c_{Ex}, (1+f)c_{Ex}, (1+f+f^2)c_{Ex}, ..., 1\right]$$

The sizes of the classes grow geometrically with the factor $f$. Again, the bounds $c_l$ and $c_u$ refer to the two boundary values next to the last advertised value.

When using the available bandwidth as the metric, the exponential increment principle is usually chosen, as more accurate link-state information is needed when the available bandwidth is low.

*Threshold-based LSUPs*

Compared to the class-based LSUPs, the threshold-based LSUPs do not have fixed values for checking the LSU condition, but use a relative difference between the resources information $C_c$ currently known by the network and the actual link-state information $C_a$ known only by the immediate nodes. A threshold $\xi$ is set by the user/provider. Once the condition $|C_c - C_a| > \xi C_c$ is satisfied, an update is triggered. From the trigger condition, we infer: $c_u = C_c(1+\xi)$ and $c_l = C_c(1-\xi)$.

*Cooperating strategies*

To reduce the protocol overhead caused by LSU traffic, the update rate should be limited. Some complementary strategies can be added to limit the number of updates. Several proposals like hold-down timer [4, 84] and moving average [54] have been provided. The hold-down timer defines a predefined amount of time as the smallest period between 2 consecutive updates on the same link. The moving average strategy does not only look at the current available bandwidth, but considers the average (over a certain window size) to trigger updates. The average value is used to follow the trend of a link-state metric and to avoid updates triggered by a short-lived metric change.

**Combined LSUPs** Based on the LSUPs mentioned above, some works suggest policies combining two or more of them. In [47], the authors suggest a policy, which combines the periodic update and trigger-based updates. The periodic LSUP will update the state information periodically with a fixed period. However, once exceptional events (defined by the router or network administrator) take place, the responsible nodes will advertise this information to the network without considering the periodic update rules.

## 2.2 The impact of stale resource information

Many works study the impact of stale resource information on QoS routing performance. The effects of stale information introduced by the LSUPs mentioned in Section 2.1 have been evaluated in [4, 47, 54, 56], where the LSUP parameters are tuned to achieve different information granularity and thus different network performance. Ma and Steenkiste [55] have compared static routing (routing without LSUs) and other routing algorithms under periodic LSUP. Shaikh *et al.* [84] give an overview of how the combination of LSUP, routing algorithm, traffic pattern and network topology influences the network performance. Additive QoS metrics such as end-to-end delay other than link capacity are also taken into consideration and should be treated differently as presented in [40].

As inaccuracy in resource information is inevitable, routing algorithms were proposed that can tolerate imprecise resource information. Some works [65, 107] suggest using intelligent routing algorithms with local resource information.

## 2.3 Two extreme strategies

A full-fledged QoS architecture requires, apart from topology updates, also resource availability updates. The first kind of updates are slowly varying in time, while the second type – the traffic related – updates are changing much faster. This chapter only focusses on the second kind of updates. The key question boils down to: “Is it worth to implement complicated update strategies for the second kind of changes in a network?” That question is undoubtedly not new, but a clear answer is still lacking. A quite important motivation to *not* update is that the Internet, where dynamic strategies of the second kind are absent, has featured a reasonably good overall functioning.

To simplify the setting, we confine to two extreme strategies $A$ and $B$. Strategy $A$ is the optimal update case that takes *all* information of the past and present into account to allocate a flow in the network at each flow request instant $t$. “Optimal” here refers to any property we would like to maximize or minimize in a network, such as, for example, throughput, rejection rate or blocking probability, or revenue. Each flow is allocated along a path for its entire life-time; we do not consider re-allocations of previous flows. Since the strategy $A$ assumes the knowledge of any past or present information, any network property can be computed optimally in our “Gedanke” experiment. We thus ignore the practical issues that may prevent the strategy $A$ to achieve optimality, such as e.g. the computation time, flooding time and the memory needed. In practice, strategy $A$ cannot be implemented, because the flooding of information always takes a non-zero time such that not all routing topology databases can be guaranteed to be synchronized. Strategy $B$ is completely static and does not update at all. Each flow is just allocated along a fixed, initially computed shortest-hop path from source to destination.

In most real networks, we cannot predict the future. At best, we may possess estimates of the likelihood of a future traffic demand. Hence, strategy $A$ is only optimal in the time interval $[0, t]$, but not necessarily in $[t, T]$, where $T > t$, because future demands are not taken into account.

Our analysis thus belongs to the field of dynamic decision theory, where the key question is "Is my myopic decision rule optimal?". In most cases, update rules are designed to improve the performance in the heavy traffic regime. For, if the traffic is low, the resources are usually plentiful and no need for update rules is perceived. Contrary to common intuition, we show that, in the heavy traffic regime, there are network scenarios where the static strategy $B$ outperforms the "optimal" strategy $A$.

## 2.4 Performance in time

In routing with exact resource information (strategy $A$), the network elements need to update every change, and paths are computed based on this exact information. In routing without LSUs (strategy $B$), only information about the topology is used. Paths are not computed for each flow, but instead, a static routing table is used. If one or more links on the path cannot accommodate the flow, the flow is rejected.

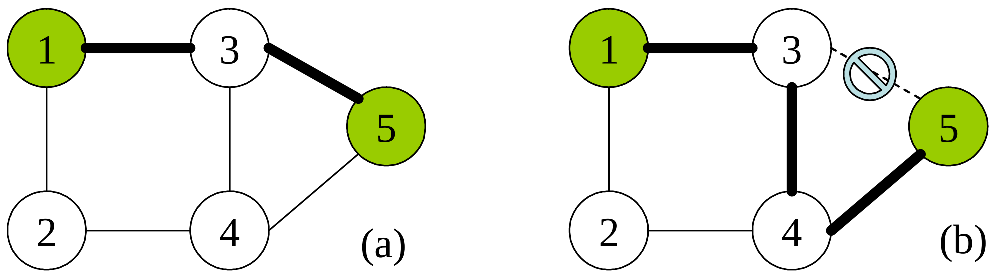


Figure 2.1: (a) the routing without LSUs; (b)the routing with exact resources information

Consider Figure 2.1, where a flow is requested from source node $1$ to destination node $5$. Routing without LSUs uses the shortest path $1 - 3 - 5$ (Figure 2.1 (a)). We assume that link $3-5$ does not have enough capacity to accommodate the new flow. Since the routing protocol without LSUs (strategy $B$) always uses the path $1-3-5$, the new flow is blocked. Strategy $A$, on the other hand, can find another path for the new flow (Figure 2.1 (b)). However, the new path in this example has a larger hopcount than the shortest path, which might be detrimental for future flows.

The effect on future flows is more pronounced under high network loads. Indeed, if an available path with $h$ hops is found for a flow with capacity requirement $c_r$, the total network capacity consumed by this flow is $c_r h$. Ergo, the more hops a path has, the more network capacity will be consumed. The static strategy $B$ chooses for each source-destination pair the path with minimum hopcount. In the first instance where the static routing in strategy $B$

blocks a flow, strategy $A$ has the potential to find an alternative, likely with longer hopcount. Thus, in the first few instances where strategy $B$ blocks flows, strategy $A$ clearly can outperform strategy $B$, but strategy $A$ consumes in those cases, more than the minimum amount of capacity. If the traffic demand remains high, strategy $A$ will find itself suffocated by those expensive flows accepted in the past, which may result in a higher blocking from then on compared to strategy $B$. In Figure 2.1 (b), future flows from $3$ to $4$ and from $4$ to $5$, can be affected by the deviation from the shortest path. Strategy $A$ greedily allocates each subsequent flow, until no network capacity is available anymore. Strategy $B$ allocates always a minimum amount of capacity, but spreads that allocation over time. Over a long time interval $[0, T]$, optimal flow scheduling assumes the knowledge of all future demands. Since the general problem of network flow scheduling in $[0, T]$ is a combinatorial optimization problem that is NP-complete, there do not exist simple greedy algorithms that are optimal. Hence, although strategy $A$ seems optimal, because it uses all possible available information up to time $t$, the impact of the unknown future demands may result in an inferior performance compared to the static strategy $B$.

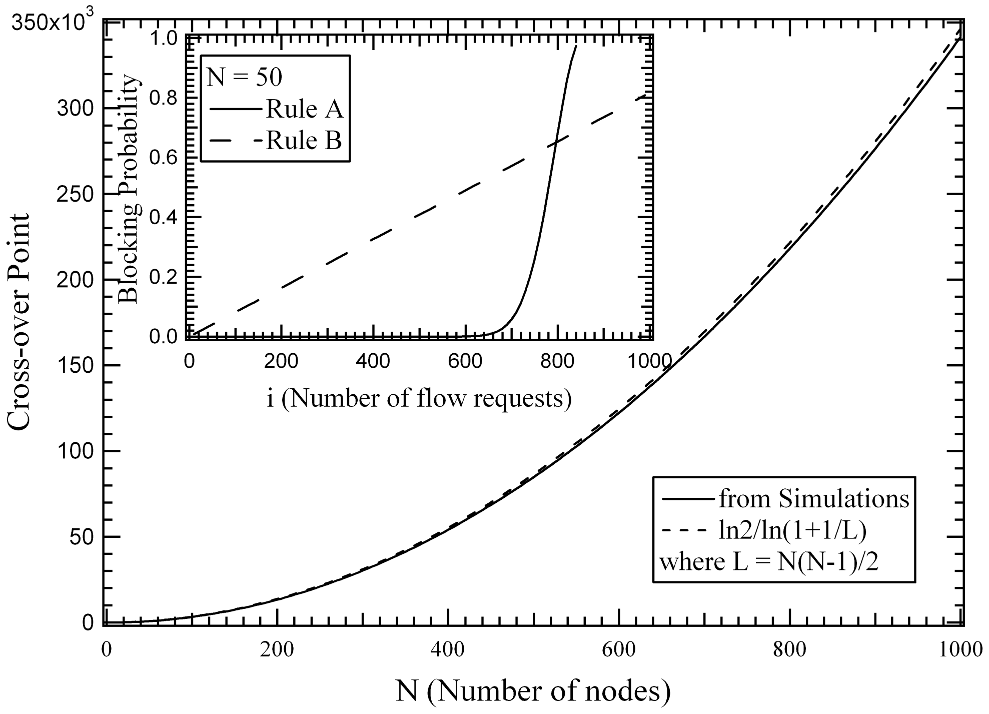


Figure 2.2: The cross-over point as a function of $N$. The blocking probability for rules $A$ and $B$ as function of the number of flow requests $i$ is shown in the insert.

In the following example we will show that indeed static routing can outperform dynamic routing with exact information. Consider the complete graph $K_N$ with $N$ nodes and $L = \binom{N}{2}$

links. We consider the time-frame $[0, T]$ and assume that each allocated flow consumes the entire link capacity for a duration $d \geq T$. Strategy $A$ perfectly routes paths with hopcount restricted to 2. Strategy $B$ only allocates paths on the direct links. We denote the steady state link availability, as $p_A(i)$ and $p_B(i)$, which are a function of the number of flow requests $i$. The blocking probability for strategy $B$ as function of $i$ equals $P_B(i) = 1 - p_B(i)$, which is the probability that the direct link is already allocated. The corresponding blocking probability for strategy $A$ is the probability that the direct link *and* all possible two hop paths between that source-destination pair are occupied. That probability is computed in [62] as

$$P_A(i) = (1 - p_A(i)) \left(1 - p_A^2(i)\right)^{N-2}$$

The expected number of links for strategy $B$ after $i + 1$ requests equals

$$Lp_B(i + 1) = Lp_B(i) - p_B(i)$$

For strategy $A$ we have

$$Lp_A(i + 1) = Lp_A(i) - p_A(i) - 2 (1 - p_A(i)) \left(1 - (1 - p_A^2(i)\right)^{N-2}) \approx Lp_A(i) - 2 + p_A(i)$$

Solving these equations gives:

$$p_A(i) \approx 2 - \left(1 + \frac{1}{L}\right)^i \text{ and } p_B(i) = 1 - \frac{i}{L}$$

For small $i$, $p_A \approx p_B$, which results in

$$P_A(i) < P_B(i)$$

Indeed less flows are blocked by strategy $A$, because more alternative paths exist. However, by using two-hop paths, resources are consumed faster (i.e., $p_A(i)$ decreases faster with $i$ than $p_B(i)$), which increases the chance of blocking for future flows. Hence, after the allocation of a large number of flows, the strategy $B$ is expected to outperform strategy $A$. The cross-over point where $P_A(i) = P_B(i)$ is approximately

$$i_c \approx \frac{\ln 2}{\ln \left(1 + \frac{1}{L}\right)}$$

as illustrated in Figure 2.2.

## 2.5 Simulation scenarios

In this section, we describe our simulation scenarios. We have used a flow-level simulator [49] developed at Delft University of Technology. The network and traffic parameters are introduced as well as the routing algorithm and signaling model. At the end, we explain how we process the data from our simulation results.

## 2.5.1 Network model

We mainly concentrate on the class of random graphs $G_p(N)$ with $N$ the number of nodes, and $p$ the probability that there exists a connection between a pair of nodes. However, we also simulate in a square lattice topology and an MCI topology, which are given in Figure 2.3. Each connection is symmetric, with the two directions treated as two links separately, and all the links have unit capacity.

As the capacity guarantee is the most likely constraint for a customer requiring QoS, we only consider the capacity requirement as our QoS metric when carrying out the simulations. Other interesting QoS metrics like delay and packet loss are often highly correlated to the available capacity (e.g., a lower available capacity is likely to result in higher delays and packet loss). Hence, we believe that the trends observed for available capacity will match to a certain extend with trends in delay and packet loss.

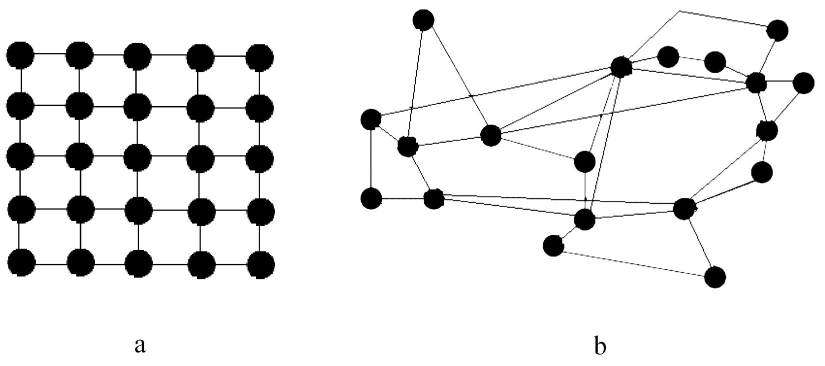


Figure 2.3: a: the square lattice topology. b: the MCI topology

## 2.5.2 Traffic model

The arrival process of the incoming flows is modeled as a Poisson process with rate $\lambda$ flows per unit time. The pairs of source and destination nodes are uniformly selected among the set of nodes. The service time of flows, i.e. the flow duration, is described by a random variable $d$. We denote by $C_r$ ($0 \leq C_r \leq 1$) the capacity requirement of each flow, which is a certain percentage of the unit link capacity.

Following Shaikh *et al.* [84], the network load is defined as:

$$\rho_N = \frac{\lambda E[d] E[C_r] E[h]}{L}$$

where $E[d]$ is the mean flow duration, $E[C_r]$ is the mean capacity requirement, $E[h]$ is the mean hopcount of the shortest paths between all pairs of source and destination nodes, and $L$ is the number of links in the network.

### 2.5.3 Routing algorithm

For each new flow, the source node uses his own view on the resource information to compute the path based on the flow's QoS requirements.

We use the widest-shortest path (WSP) algorithm with pruning. As explained in [55], this algorithm gives high priority to limiting the hopcount, thus limiting the resource consumption. It is said to perform better than the shortest-widest path algorithm, which gives high priority to balancing the network load. The basic steps of the WSP algorithm are given below:

1. Based on the resource information of the source node, all the links $l_{i,j}$ (the link from node $i$ to node $j$) for which the known available capacity $C_{l_{i,j}} < C_r$ are pruned.
2. Compute the minimum hop paths in the pruned graph.
3. If there is only one path found with the smallest hopcount, this path will be used to carry the flow; else if more than one shortest paths exist, we select the path $P$ with the maximum available capacity ($C_P = \min_{l_{k,m} \in P}(C_{l_{k,m}})$).

The available capacity $C_{l_{i,j}}$ of link $l_{i,j}$ known by the source node, is of great importance in the process of selecting the path. Inaccuracy of $C_{l_{i,j}}$ may result in finding a non-optimal path. A flow fails during the routing process if no paths can be found by the routing algorithm.

### 2.5.4 Signaling model

Once a path is found by the routing algorithm for a new flow, from the source node on, the amount of resources needed will be reserved on each link along the path. If all the links along this path can accommodate the new flow, i.e., $C_{l_{i,j}} \geq C_r, \forall\, l_{i,j} \in P$, the path is set up. The resources required by this flow will be reserved during the service time until this flow terminates. As the information of the source node may be inaccurate, the path computed by the routing algorithm might be unfeasible. In this case, the resources that have already been reserved need to be released, and the flow is rejected during the setup process. We call this a setup failure.

Upon a failure during either the routing process or the set up process, we drop the flow without rerouting, because our aim is to see how stale information affects the network performance under the same path selection process.

### 2.5.5 Statistical model

The square lattice topology and the MCI topology are fixed. For each class of random graphs, we generate $1000$ different connected graphs.

For each combination of graph, network load and extreme LSUP, $10$ realizations are simulated. In each simulation, a set of $200,000$ flows are offered to the network in a Poissonian way. The first $100,000$ flows are used as warm-up flows whose routing results will not be taken into account when collecting the data. In this way, we model a steady-state behavior. Once the $10$ iterations are finished, an average will be taken for all the data.

## 2.6 Simulation results

We first present our simulation results after which we collectively discuss them. The blocking ratio given by routing mechanism $x$ is the comparison metric and defined as follows:

$$BR_x = \frac{Number\ of\ failed\,flows\ with\ x}{Total\ number\ of\ flows}$$

Two types of traffic were tested: one with the flow duration $d$ set to $100$ time units, and the capacity requirement of each flow $C_r$ set to $5\%$, a constant percentage of the unit link capacity; the other with the flow duration $d$ following a Weibull distribution with mean $100$ time units to capture the long-tailed nature of connection durations, and $C_r$ set be to a uniform distribution in the range $[2.5\%,\ 7.5\%]$ of the unit link capacity. Smaller capacity requirements may be more realistic, but will cause scalability problems in time consumption for our simulations. Larger capacity requirements are chosen to capture the performance of routing mechanisms with an acceptable time consumption. As a consequence, our simulation results may give an exaggerated view on the performance of the considered routing strategies.

For each graph, we simulate with different load $\rho_N$ ($\rho_N < 1$), and the flows' arrival rate $\lambda$ is calculated as:

$$\lambda = \frac{L\rho_N}{E[d]E[C_r]E[h]}$$

### 2.6.1 MCI topology

Figure 2.4 shows the network performance of the two extreme strategies in the MCI topology, as a function of the network load under two traffic patterns. The data points in this figure have their corresponding 90% confidence intervals smaller than 5% of the shown average value, and are considered as valid to present the performance of the two routing strategies. For both traffic patterns, the blocking ratios for both extreme strategies increase quickly as the network load increases, and the biggest difference happens somewhere near $\rho_N = 0.5$. As expected, routing without LSUs performs worse than with exact resource information, but the difference is small under low and very high network loads.

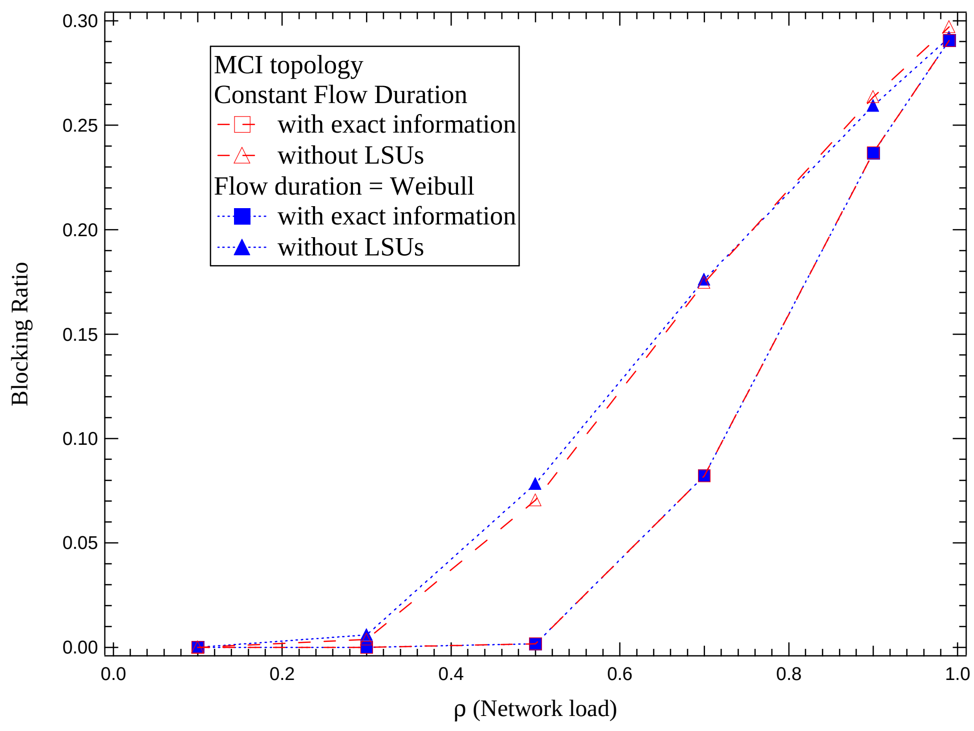


Figure 2.4: The blocking ratio on MCI under different types of flow duration, different capacity requirements, and different network loads.

### 2.6.2 Square lattice topology

For the lattice, the network performance under two traffic patterns is shown in Figure 2.5. In this figure, the data points have their corresponding 90% confidence intervals smaller than 8% of the shown average value. We expect that the curves are able to show the relative performance of these routing strategies. Again, under low network loads, routing without LSUs has a similar performance as routing with exact resource information. However, its performance decreases quickly as the network load increases. The reason is that, in a topology like lattice, between each pair of nodes, there may exist multiple shortest paths, and using only one out of this set could result in earlier blocking. Randomized routing [5] was introduced to balance the load. For each pair of source and destination nodes, a set of shortest paths are computed, once a flow comes, the source selects one path randomly from this set. We applied the randomized routing to the traffic pattern with constant flow duration. Figure 2.5 shows that randomized routing improves the network performance considerably for the no LSUs case, but it still gives higher blocking than routing with exact resource information. The routing with exact resource information starts giving increased blocking ratios at a heavier network load.

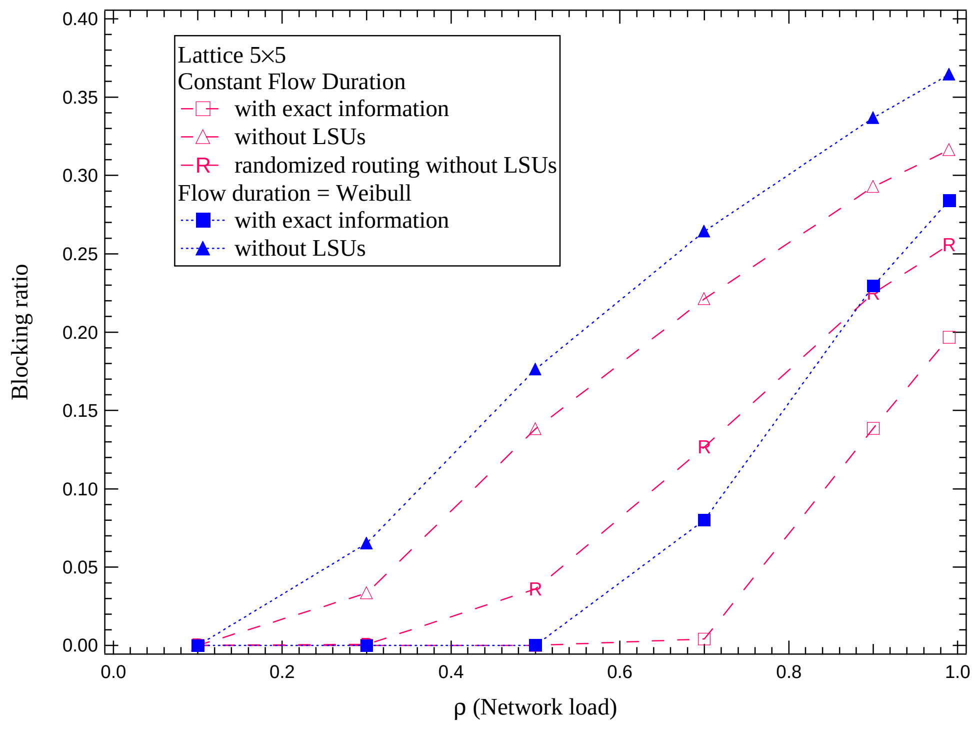


Figure 2.5: The blocking ratio on lattice under different types of flow duration, different capacity requirements, and different network loads.

### 2.6.3 Random graphs

We have performed simulations in the class of random graphs $G_p(N)$, for different $N$ and different $p$. The critical link density $p_c$ in the random graphs equals $p_c \approx \frac{\ln N}{N}$. Choosing $p < p_c$ results, with high probability, in disconnected graphs. We consider only $G_p(N)$ with $p > p_c$.

Figure 2.6 shows the network behavior in $G_{0.1}(50)$ under different network loads for the two traffic patterns. In the $4$ curves representing the blocking ratios for different routing mechanisms under different traffic patterns, the curve for routing without LSUs under constant flow duration gives the biggest relative 95% confidence interval, which is always smaller than $10\%$ of the average blocking ratios shown as the data points on this curve. This indicates that, with the average blocking ratio taken for each point in the curves, we can give a quite accurate view on the performance of different routing mechanisms under different traffic patterns.

Figure 2.7 shows the network behavior in the class of random graphs $G_p(N)$ with different $N$, and different $p$ under network load $\rho = 0.99$ for the type of traffic with constant flow duration.

As can be seen from the Figures, when the network load is low, the difference between

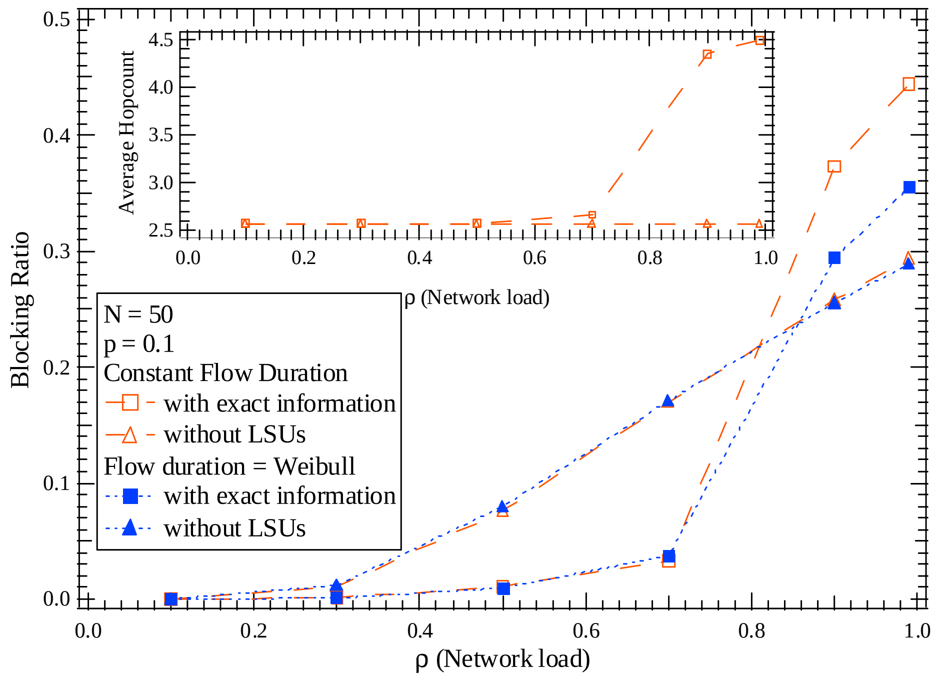


Figure 2.6: The blocking ratios on the class of random graphs under different types of flow duration, different capacity requirements, and different network loads with $N = 50$ and $p = 0.1$. The average hopcount for the constant flow duration is put as insert.

strategy $A$ and strategy $B$ is small, since it is unlikely that the shortest paths will get congested. The difference increases with $\rho_N$, provided the network load is not very high. For $\rho_N \to 1$, shortest paths become rapidly congested and, under strategy $A$, longer-hop paths are used. Allocating long-hop paths may block future requests from being allocated, which explains the steep increase in blocking probability for the class of random graph. This explanation is substantiated by the insert in Figure 2.6, which gives the expected hopcount between all pairs of nodes as a function of network load.

In the case of strategy $B$, no matter how many possible paths there exist for a certain pair of source and destination nodes, only the one with the smallest hopcount is selected without considering the resource information. Thus, the average hopcount does not change as the network load changes.

In the case of strategy $A$, under small loads, the expected hopcount equals that of the shortest paths (used with strategy $B$). However, when $\rho_N$ reaches a certain value, the average hopcount $E[h^{'}]$ for routing with strategy $A$ increases quickly. The actual load $\rho^{'}$ which each link experiences, can be given as:

$$\rho^{'} = \frac{\lambda E[d]E[C_r]E[h^{'}]}{L}$$

$\rho'$ goes to 1 before $\rho_N$ does. This is the reason why the curves for routing with strategy $A$ cross those for routing with strategy $B$ in Figure 2.6.

Figure 2.7 shows that under the traffic model with the constant flow duration, $BR_A$ increases as $N$ increases. As the network grows, the routing with strategy $A$ gives worse performance with more computation and communication overhead. This is neither economical nor scalable.

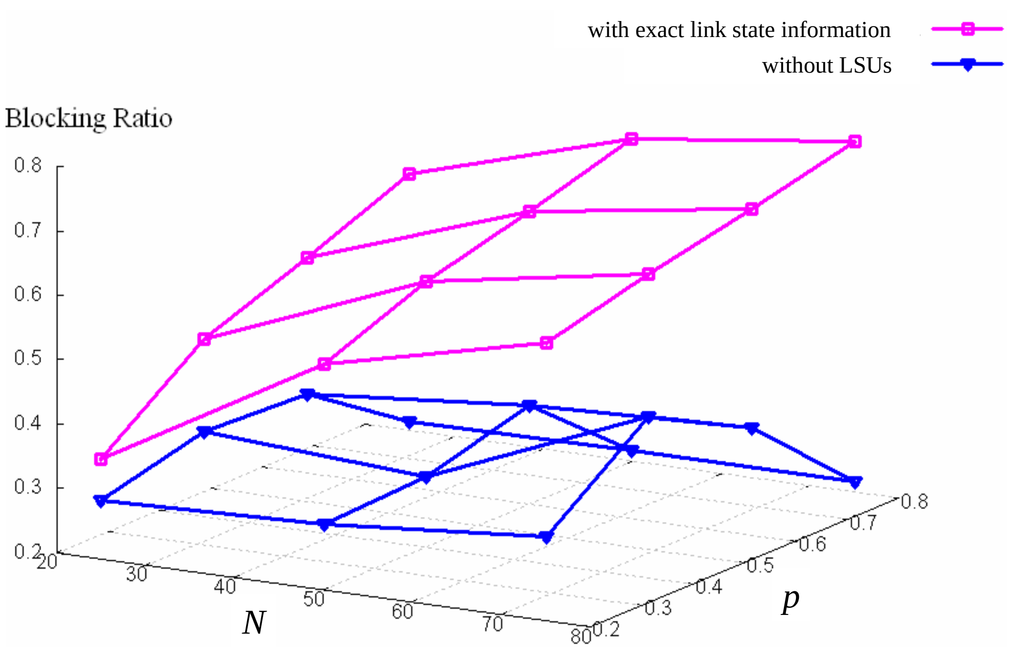


Figure 2.7: The blocking ratios in the class of random graphs $G_p(N)$ with different $N$, and different $p$ under network load $\rho = 0.99$, with the flow arrival process following the Poisson process; flow duration $d$ set to 100 time units; capacity requirement $C_r$ set to be $5\%$ of the unit link capacity.

For $BR_B$ we observe for each $N$, a bell-shaped curve, with a peak depending on the number of nodes $N$. For small $p$, the graph will approximate a tree-like structure and the shortest paths between all pairs of nodes are likely to span nearly all links. In that case the load is well balanced. Similarly for $p \rightarrow 1$, the shortest paths are likely to be the direct links and by using these links the network does not waste many resources, which would affect the allocation of other flows. In between these densities, the shortest paths may only span (i.e., use) a small portion of the network, causing an increased chance of blocking (which might be reduced via randomized routing as shown for the lattice networks).

We carried out similar simulations under the traffic model with Weibull distributed flow duration, and obtained similar results as under the traffic model with constant flow duration.

## 2.7 Conclusion

In this chapter, we have presented analytical and simulation results for two extreme link state update policies (LSUPs), namely *routing with exact resource information* and *routing with static information*. Different traffic and network scenarios have been evaluated. Our results show that the extremes react differently to different network models. In each network model, the difference in performance varies with the network load, and stays small under low network loads.

For the MCI topology, the performance difference remained fairly small.

For the square lattice topology, routing with exact resource information gives small blocking ratio even under heavy network load, while routing with static information degrades very fast as the network load increases. However, by introducing randomized routing without LSUs, the network performance could be improved substantially.

The class of random graphs $G_p(N)$ performs differently as $p$ or $N$ varies. When the amount of traffic running in the network is relatively small compared to the high capacity provided by the core network, routing without LSUs is expected to give satisfactory performance. However, under very high network loads, the intelligence of knowing exactly the resource information backfires, because it leads to longer-hop paths, which results in a more congested network, leading to high blocking ratios.

The most striking observation in our work is that under high network load the "no updates" routing performs close to or even better than the "exact" routing. If we would have counted the amount of traffic overhead induced by the LSUs, this observation would even be more pronounced. Our results argue that in this case an investment in LSUPs will not be cost efficient and that a simple shortest-hop framework is likely to suffice.

# Chapter 3

# Importance of scenarios

Fu, B. and S. Uhlig, 2008, "*On the relevance of on-line traffic engineering*", in Proc. of the 19th ITC Specialist Seminar on Network Usage and Traffic, Berlin, Germany, October.

Many factors affect the outcome of the path finding and resource allocation process by on-line traffic engineering algorithms. The topology structure, the link costs, the residual link capacities, the path selection, *etc*, affect the global performance of the traffic engineering algorithms. Obtaining a priori knowledge of the behavior of a traffic engineering algorithm on a particular network scenario is very hard due to the interactions between the different aspects of the problem (see Figure 3.1).

Traffic engineering algorithms are usually designed to optimize a particular objective function and address a specific traffic engineering problem [13, 37, 50, 104]. Setting up scenarios under which traffic engineering algorithms will be compared is difficult because some traffic engineering algorithms have not been designed to perform well under specific situations. Not all scenarios are practically relevant. If some traffic engineering algorithm performs poorly under a scenario that is not relevant in practice, this should not undermine the suitability of the traffic engineering algorithm.

We categorize the different factors that affect the behavior of traffic engineering algorithms as follows:

- network topology: The location of Points of Presence (POP) and the existing connectivity (e.g. fiber) between POPs largely determine the network structure. Most ISPs design their networks with specific redundancy requirements, leading to rather sparse topologies that trade-off cost and robustness [19].
- traffic demand: The amount of traffic between each pair of routers.
- network provisioning: Network provisioning consists in scaling the link capacities, so as to accommodate the traffic between all pairs of routers.

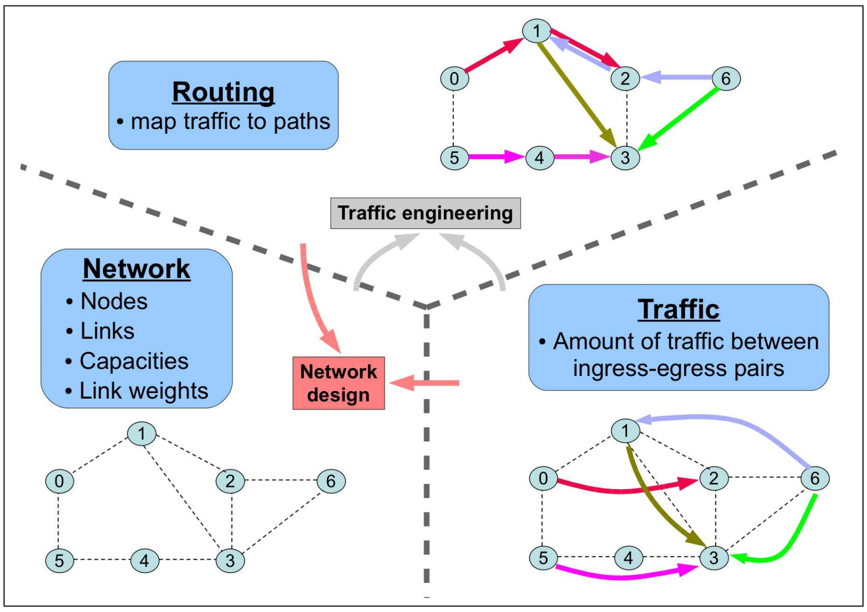


Figure 3.1: Schematic representation of the traffic engineering/network design problem.

As illustrated on Figure 3.1, the network, traffic and routing are intertwined. The design of the network (capacities and link weights) must ensure that the traffic demand can be satisfied. To assign link capacities, network design requires an initial model of routing to determine how much traffic will flow on each link. In today's networks, link weights are set based on the delay, the capacity [68], or a combination of delay and capacity of the links. The goal of setting IGP weights in such a way is to attract traffic towards high-capacity and low-delay links. In practice, many backbone operators use the ad-hoc approach of observing the flow of traffic through the network [34], and iteratively adjusting a weight whenever the load on the corresponding link is higher or lower than desired. The problem has been addressed in [36, 37, 66] using different techniques from operations research. Observing traffic in large backbone networks requires significant resources [34], so that in practice the real traffic demand is inferred rather than observed [60, 111]. Few observed traffic demands are publicly available [2, 96].

In this chapter, we study the behavior of on-line traffic engineering algorithms, using a set of real and synthetic traffic demands and backbone network topologies. To our knowledge, this is the first time that the relationship between network design and the behavior of on-line traffic engineering algorithms is studied in such a way. Our contributions consist in identifying several factors that determine the relevance of on-line traffic engineering algorithms. Due to the small viability of different time scales [95], in this paper we use arbitrary scalings, which are supposed to show the general behaviors.

The remainder of this chapter is organized as follows. We review the literature by presenting some well-known traffic engineering algorithms in Section 3.1. The behavior of traffic engineering algorithms on some real-world scenarios is studied in Section 3.2. In Section 3.3, we build our own scenarios to study the impact of network design aspects on traffic engineering algorithms.

## 3.1 Intradomain routing algorithms

Many Autonomous Systems (ASs) use OSPF/IS-IS as their intradomain routing protocol. These protocols select shortest paths based on static link weights. These IGP link costs reflect the desirability of a link to be selected to carry traffic inside the AS. We call the routing algorithm used by these routing protocols the *Static Shortest Path* algorithm (SSP), as the shortest paths will remain the same no matter the amount of traffic actually carried on those links, as long as the link weights do not change. The link costs can be computed off-line according to the offered traffic, the provisioned network capacity, and the specific objective desired by the network administrator [37], or simply by following Cisco's guideline of IGP costs proportional to the inverse of the link capacity [24]. Recently, splitting traffic among several shortest-path, called equal-cost multi-path (ECMP) [63], has been deployed by an increasing number of network operators. ECMP provides some load-balancing and makes the granularity of the traffic finer compared to non-ECMP.

The algorithms that use the off-line computed link costs, are called off-line routing algorithms. SSP is computationally efficient, but unaware of the link utilization. If the link costs setting is far from the optimal setting with respect to the current traffic demand, then SSP may heavily load some links while avoiding others that have plenty of unused capacity. When capacity is not available on the shortest path, IP packets are simply dropped as the router buffers saturate or connection requests are rejected as feasible paths do not exist.

On-line routing algorithms have been proposed to find paths with bandwidth requirements. On-line algorithms use dynamic information such as the link residual capacities to compute the feasible paths, and are able to find one, provided it exists. There is an extensive literature on on-line traffic engineering algorithms. Due to space limitations, we do not discuss all of them. In this section, we select a set of representative traffic engineering algorithms, from simple to computationally complex ones.

The Widest Shortest Path algorithm (WSP) [41] computes the shortest IGP paths in the network formed by links with sufficient residual capacities to accommodate the incoming request, and selects from all the feasible shortest IGP paths the one with the maximum bottleneck residual capacity to load-balance network traffic. It avoids using heavily loaded links unless there is no other option. Only feasible shortest IGP paths are considered.

Another instance of on-line algorithm is to use shortest-path routing with dynamic link weights. Instead of using SSP with a static inverse of the link capacity as the link weights, one can use the inverse of the residual capacity. As links get loaded, the weights will adapt to reflect the available capacity in the network. We call this technique dynamic shortest path with inverse residual capacity (DSP-Inv).

The family of Minimum Interference Routing Algorithms (MIRA) [50] tries to select, between each source-destination pair, a path that interferes the least with other source-destination pairs. MIRA takes into account the information of source-destination pairs $(S_i, D_i)$ and weights them by their importance $\alpha_i$. The importance of a source-destination pair can be set for example as the fraction of the total traffic it represents. To minimize interference, those algorithms maximize the sum of the residual weighted max-flows[1] between all source-destination pairs. Upon the arrival of a connection request from $S_i$ to $D_i$, the algorithm will select a path that can maximize

$$\sum_{(S_j,D_j),j\neq i} \alpha_j \theta_j \tag{3.1}$$

where $\theta_j$ represents the residual max-flow for $(S_j, D_j)$ after a path is selected for $(S_i, D_i)$. Maximizing this objective function is NP-hard [50]. The MIRA algorithms reach a suboptimal solution by computing the set of *critical* links $L_j$ for all other pairs $(S_j, D_j)(j \neq i)$. A link $l$ is called *critical* for a source-destination pair $(S, D)$, if the reduction in $l$'s capacity leads to the reduction in $(S, D)$'s residual max-flow. Then link costs are set according to the link *criticality*, and shortest path computation is applied to this weighted topology. When only the importance of each source-destination pair is considered in the link *criticality*, the link cost can be set as

$$C(l) = \sum_{(S_j,D_j),l\in L_j} \alpha_j \tag{3.2}$$

or as

$$C(l) = \sum_{(S_j,D_j),l\in L_j} \frac{\alpha_j}{R(l)} \tag{3.3}$$

when the residual capacity of each link is also considered, where $R(l)$ is the residual capacity of $l$. When MIRA is using equation 3.2 (equation 3.3) for its link costs, it is called MIRA-TM (MIRA-TM-Cap). For further details about MIRA we refer to [50].

We use the example in Figure 3.2 to show how different algorithms choose paths. Let us focus on two source-destination pairs: $(S_1, D_1)$ and $(S_2, D_2)$, to which we set the same weight $\alpha$ when using the MIRA algorithms. Each link in Figure 3.2 is labeled with a pair of values $(w, c)$ in black, where $w$ is the link weight and $c$ is the current residual capacity. For $(S_1, D_1)$, SSP and WSP will choose the same path, $0 - 1 - 2$, as it is the only shortest path according to the link weight settings. For $(S_2, D_2)$ on the other hand, different paths can be chosen: SSP will choose $1 - 2 - 6$ or $1 - 3 - 6$; WSP will choose $1 - 3 - 6$ as its bottleneck capacity is larger than that of $1 - 2 - 6$. DSP-Inv will choose the path $0 - 1 - 2$ for $(S_1, D_1)$. For $(S_2, D_2)$, DSP-Inv will choose the path $1 - 3 - 6$. When a connection request comes from $S_1$ to $D_1$ for 1 unit of capacity, the MIRA algorithms first identify the *critical* links for $(S_2, D_2)$: $1 - 2$, $2 - 6$ and $3 - 6$. The weights of these links will be increased by $\alpha$. A shortest

[1]The max-flow of a source-destination pair $(S, D)$ is the maximum amount of traffic that can be pushed between this pair. Multiple link-disjoint paths may be used.

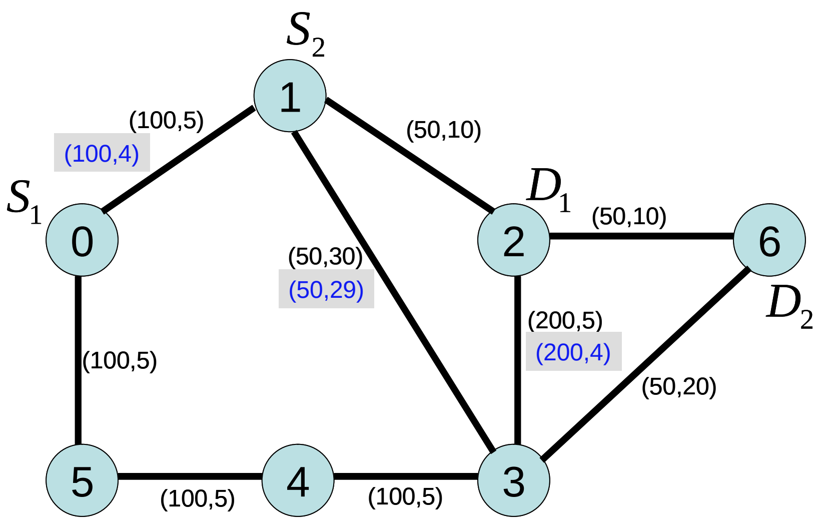


Figure 3.2: Example topology.

path computation will then be carried out based on this weight setting and the resulting path will be $0-1-3-2$ or $0-5-4-3-2$. Assume that path $0-1-3-2$ is chosen by the routing algorithm, then the properties of links along this path can be represented by the pairs with a grey background on Figure 3.2. If now a request from $S_2$ to $D_2$ comes in this network, the MIRA algorithms will identify the *critical* links for $(S_1, D_1)$ as being $0-1$, $0-5$, $5-4$ and $4-3$. $\alpha$ will be set as the weight for these links, and a shortest path will be computed on this weights setting. The resulting path will be one among $1-2-6$, $1-3-6$, $1-2-3-6$ and $1-3-2-6$.

## 3.2 Behavior on real-world networks

In this section, we observe the behavior of on-line traffic engineering algorithms on real-world networks and real traffic demands. In Section 3.3 we rely on generated topologies to show the impact of different aspects of network design on the behavior of traffic engineering algorithms.

### 3.2.1 Scenarios

We use real topologies and traffic matrices from Abilene [2] and GEANT [96]. Abilene and GEANT are the US and European academic backbones. They both provide transit service to universities and research institutions in the US and Europe. The Abilene topology has 12 nodes and 30 directed links while GEANT has 22 nodes and 72 directed links. The link capacities and link costs are kept as in reality, with link costs in Abilene assigned based on the geographical distance, and that in GEANT following a modified version of Cisco's proposal (proportional to the inverse of the link capacity). For Abilene, a traffic matrix spans $5$ minutes, and for GEANT $15$ minutes. When using a traffic matrix, for each source-destination pair, we deduce the average bandwidth requirements from the traffic matrix. The traffic between each source-destination pair is further split into small pieces to create the requests for path establishment inside the network. The traffic demand from the traffic matrix between each source-destination pair is split into $200$ equal pieces. In practice, the order in which the connection requests arrive may have an impact on the behavior of the traffic engineering algorithm. If the order of the connection requests is important for the performance of the traffic engineering algorithm, then an off-line traffic engineering algorithm should be used in order to properly schedule the requests. As we are interested in on-line traffic engineering algorithms in this section, we do not consider the interactions between the connection requests. Therefore, the requests we use are small enough to be close to a fluid for the network[2]. Connection requests are handled in the following way:

- Select uniformly an source-destination pair from all potential ones;
- Inject a connection request between this pair with a corresponding bandwidth requirement;
- Reserve the capacity for this connection request if a feasible path is found by the algorithm; the capacity will be reserved for this connection until the end of each analysis.

No matter which algorithm is used to compute the paths (including SSP), a reservation is made in the network if a feasible path is found by this algorithm.

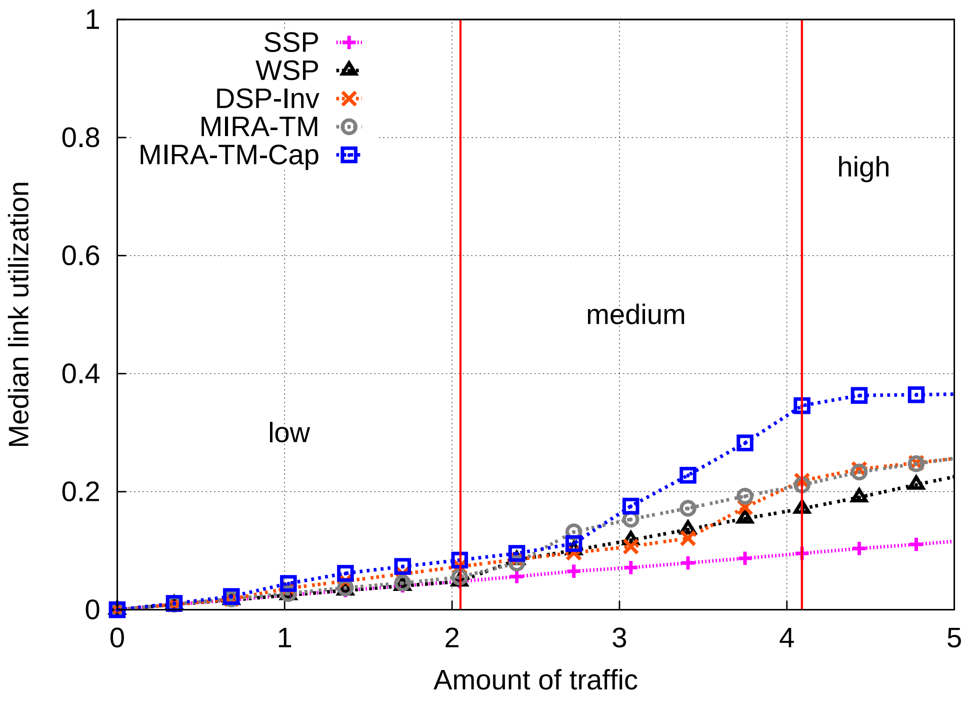
By adjusting the total number of injected connection requests, different *load regimes* can be reached. We show in Section 3.2.2 how different *load regimes* appear when scaling traffic demands.

To check our results with the literature, we also borrowed the scenario from [50]. In this topology, each link is bidirectional (can be treated as $2$ independent unidirectional links with the same capacity). Some links have capacities of $1200$ units and the others have $4800$ units. The connection requests arrive randomly, with the same rate for all given source-destination pairs, and with a bandwidth requirement uniformly distributed between $1$ and $3$ units. The connection requests are injected into the network in the same way as explained for Abilene and GEANT.

### 3.2.2 Impact of traffic load

In this section we provide simulation results of the different traffic engineering algorithms applied to both Abilene and GEANT. For each network, we pick one traffic matrix out of

[2]We considered coarser connection requests and noticed that increasing their size does not significantly change our results, but makes the curves appear less smooth.



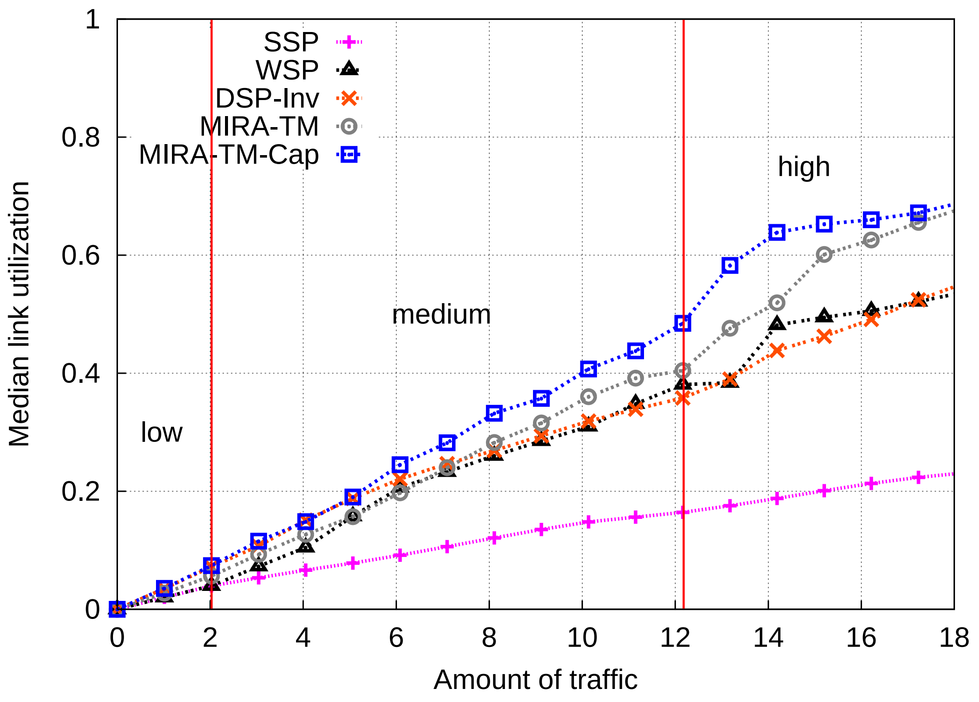


Figure 3.3: Median link utilization under different load regimes: Abilene (top) and GEANT (down).

several measured ones during peak time. Figure 3.3 shows the median link utilization after each connection request is routed by the traffic engineering algorithms. The median link utilization is the value such that half of the links in the network have a smaller utilization. We use the median because we want to have a picture of the global link utilization in the network, and because it is less sensitive to extreme values than the average. The average link utilization provides similar results. The wide variations between the minimum and the maximum link utilization make those statistics not insightful.

To see how traffic engineering algorithms manage to handle different traffic loads, we inject requests proportionately to the real traffic matrices until the connection requests for a significant fraction of the source-destination pairs cannot be accepted. By doing this, we ensure that we sample all *load regimes* that correspond to the traffic distribution given by the traffic matrix.

The results for the following traffic engineering algorithms are shown on Figure 3.3: SSP, WSP, DSP-Inv, and 2 variants of MIRA (MIRA-TM and MIRA-TM-Cap). MIRA-TM considers in its weighting the traffic between source-destination pairs (see equation 3.2). MIRA-TM-Cap takes into account not only the traffic between source-destination pairs, but also the residual capacity of the critical links (see equation 3.3).

The x-axis of both graphs of Figure 3.3 give the sum of the connection requests that have been pushed into the network so far (total traffic that has been accepted). We scaled this sum to quantify the traffic demands in terms of the measured traffic matrices, without revealing the exact amount of traffic that a single traffic matrix represents. The top graph in Figure 3.3 makes three different *load regimes* appear: low load, medium load, and high load. When amount of traffic is low enough, the algorithms can be divided into two groups: 1) SSP and WSP; 2) DSP-Inv and the MIRA family. SSP and WSP share the same link costs settings which do not give multiple shortest IGP paths in the Abilene network, so when the load is low enough SSP and WSP will choose the same paths. DSP-Inv and the MIRA algorithms on the other hand do not follow the static link costs settings of the network, but use longer paths with respect to the static IGP costs to avoid critical links and hence reduce interference with other requests. If network administrators prefer low link utilization, traffic engineering algorithms like DSP-Inv or MIRA perform worse than SSP and WSP as seen from the graph for median link utilization.

In what we call the medium load regime, SSP and WSP choose different paths. WSP chooses longer paths on average, like MIRA, hence the median link utilization increases faster than for SSP. When the network starts to become loaded, WSP detects that some shortest paths are not feasible any more and thus turns to alternative paths. SSP always uses the paths computed based on the static link costs settings, resulting in blocked connection requests when load becomes too high. One of the interests of traffic engineering algorithms lies in their ability to find feasible paths as long as there exists some, even when load is becoming high in the network.

Finally, the high load regime occurs when not enough capacity is left in the network. In this situation, even the MIRA family is unable to satisfy connection requests between a significant fraction of the source-destination pairs. This situation should never happen in practice, as it indicates inadequate network provisioning.

For the GEANT network (bottom graph Figure 3.3), we also observe three distinct load regimes. In the low load region SSP and WSP behave in the same way. Then in the medium load region WSP chooses different paths from those chosen by SSP and has a higher median link utilization. Finally, in the high load regime, the median link utilization hardly increases due to lack of available capacity to satisfy connection requests.

### 3.2.3 Available capacity in the network

Figure 3.3 showed that the median link utilization stays relatively low, even under the high load regime. Although many links still have capacity left, traffic engineering algorithms are unable to satisfy a significant fraction of the connection requests. As paths between source-destinations are typically made of several links, connection requests between most of the source-destination pairs cannot be satisfied, as soon as a significant fraction of the links are highly loaded.

Link utilization-related metrics give a picture that is easy to understand for network operators, as it is closely related to delay [70]. To understand the behavior of traffic engineering algorithms on the other hand, a metric based on the capacity left in the network is more helpful. For this, we rely on the sum of the residual max-flows of all source-destination pairs. The residual max-flow of a particular source-destination pair tells how much usable capacity is left in the network to satisfy connection requests between the considered source-destination pair. The sum of the residual max-flows for all source-destination pairs tells how much capacity is usable by traffic engineering algorithms to satisfy connection requests.

Figure 3.4 shows how the sum of residual max-flows for all source-destination pairs evolves for increasing load in the Abilene network (x-axis is the same as on Figure 3.3). Note that the value of the sum of the residual max-flows is meaningless, only the relative speed at which it decreases for different traffic engineering algorithms is of relevance. We thus normalize it in all figures.

In the low load regime, there is no big difference between the algorithms. SSP and WSP use a bit more of the total max-flow than the other three algorithms. In the medium load regime, SSP cannot satisfy some connection requests, hence does not use the available max-flow. WSP, DSP-Inv and the two MIRA variants manage to explore available paths to route connection requests. In the high load regime, the four dynamic algorithms hardly manage to satisfy connection requests. Hence, they have a very slowly decreasing residual max-flow.

Overall, on-line traffic engineering algorithms behave in a similar way. They use longer paths than SSP to make a better use of the available capacity. Unless those longer paths do not interfere with later requests from other source-destination pairs, traffic engineering algorithms should perform equally well on the scale of the whole network, as measured by the sum of the residual max-flow. As networks contain less links than source-destination

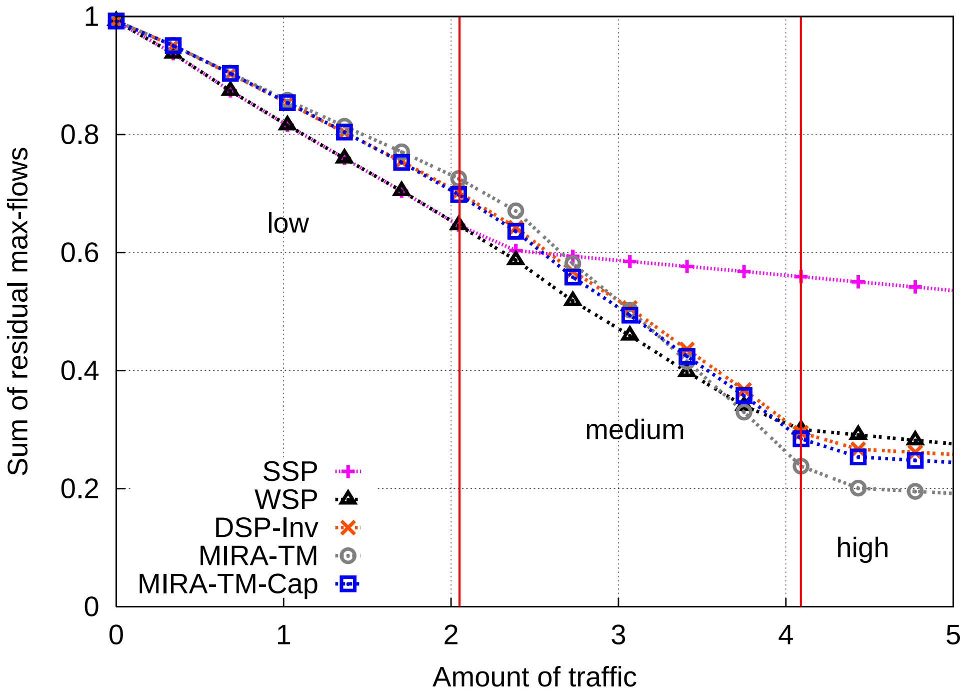


Figure 3.4: Network-wide residual max-flow on Abilene network.

pairs, it is unlikely that alternative paths that do not interfere with other source-destination pairs may be found in the network.

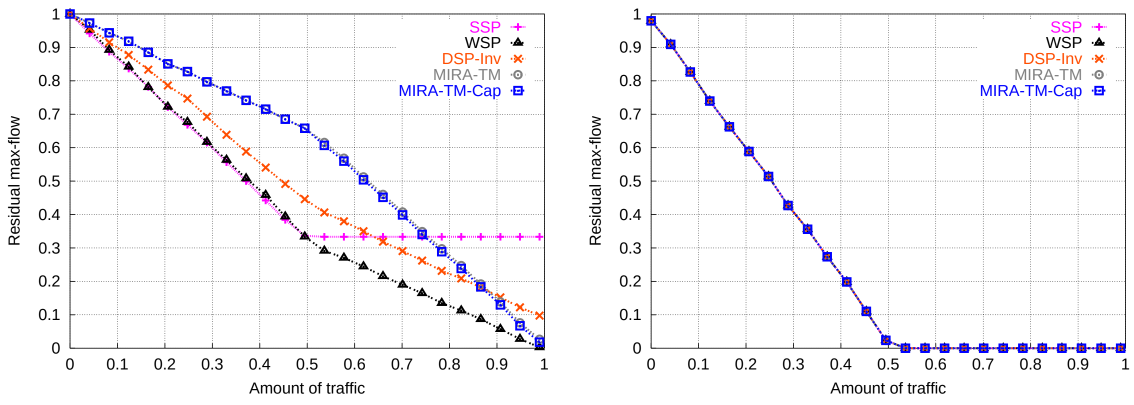


Figure 3.5: Residual max-flow for pair $(S_1, D_1)$ (left) and pair $(S_2, D_2)$ (right)

To better understand why traffic engineering algorithms behave globally in a similar way, we need to focus on specific source-destination pairs. To make the explanation simpler, we take the same topology as used in [50]. The MIRA algorithms were proposed to try to minimize interference between different source-destination pairs. The insight behind MIRA is that in order to better utilize the available capacity in the network, each source-destination pair should try to prevent interfering with the residual max-flow of other source-destination pairs when choosing their path. Another important aspect of interference is that different source-destination pairs have different possibilities to have their paths routed in order to minimize interference. We implemented the scenario of [50] and computed after each request the residual max-flow of each source-destination pair.

Figure 3.5 shows the residual max-flows of $2$ of the $4$ source-destination pairs used in [50], $(S_1, D_1)$ and $(S_2, D_2)$. For $(S_1, D_1)$, we obtain results similar to those shown in [50]. The two variants of the MIRA algorithm leave a larger residual max-flow compared to the other three algorithms for an amount of traffic smaller than $0.75$. Next comes DSP-Inv, and finally WSP and SSP. At an amount of traffic of $0.5$, SSP cannot use the whole max-flow of $(S_1, D_1)$ due to its inability to choose alternative paths. DSP-Inv is also unable to use the whole max-flow of $(S_1, D_1)$, although to a smaller extent than SSP. WSP and the two MIRA variants on the other hand manage to use the whole max-flow of $(S_1, D_1)$.

For $(S_2, D_2)$ (right of Figure 3.5), the residual max-flows found for all the considered algorithms are the same. Although there are two bottleneck links forming $(S_2, D_2)$'s max-flow, both links are also used by $(S_4, D_4)$'s max-flow. No traffic engineering algorithm can prevent interference to happen for $(S_2, D_2)$.

From a network-wide perspective, traffic engineering algorithms alleviate problems on some links by shifting it to other links. How much traffic engineering helps, however, depends much on the considered source-destination pair. When alternative paths can be used without interfering with other source-destination pairs, traffic engineering algorithms may help. When critical links cannot be bypassed, traffic engineering cannot compensate for inadequate provisioning in the network or for links that belong to the residual max-flow of several source-destination pairs.

### 3.2.4 Impact of traffic pattern

Network traffic is dynamic, it exhibits daily and weekly patterns [25, 35, 94]. Different traffic patterns might complexify the picture of traffic engineering algorithms we have shown so far. In the previous section, we showed the impact of the load regime on the behavior of traffic engineering algorithms. Over time, both the total amount of traffic and the distribution of the traffic among source-destination pairs may change. On Figure 3.6, we show the distribution of the traffic among source-destination pairs for four different traffic matrices of Abilene. We selected two days, June $1^{st}$ and $2^{nd}$ $2004$, and within those two days we selected two time intervals, one during peak time ($16:55$) and another during non-peak time ($21:35$).

We observe on Figure 3.6 that the distribution of the traffic differs much between peak and non-peak time for Abilene. During peak time, one particular source-destination pair is responsible for about $60\%$ of the total traffic. During non-peak time on the other hand, the traffic distribution among source-destination pairs is less uneven, but still far from uniform. During non-peak time, $20$ among the $132$ source-destination pairs are responsible for about

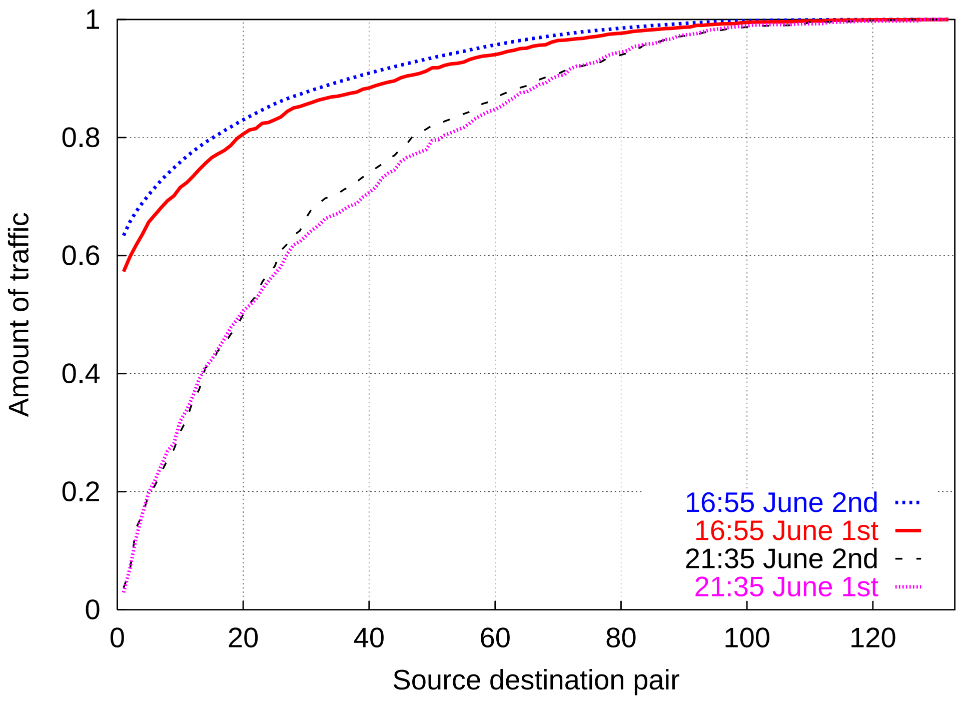


Figure 3.6: Distribution of traffic among source destination pairs (Abilene).

50% of the total traffic. Note that the variations in the traffic demand of GEANT are limited and do not impact our results.

Similarly to Section 3.2.2 and 3.2.3 where the results under traffic matrix of Abilene taken at $16:55$ June 1st were shown, we performed the comparison of traffic engineering algorithms on several other traffic matrices (the other three instances whose distributions are shown in Figure 3.6) of Abilene to see how different traffic matrices affect the comparison. The behaviors of traffic engineering algorithms on these traffic matrices are shown in Figure 3.7. Surprisingly, we do not observe major differences with the results from Section 3.2.2 and 3.2.3. As soon as the network becomes loaded, we observe different behaviors of the traffic engineering algorithms, corresponding to the medium or high load regimes. When the amount of traffic is low, then the traffic engineering algorithms behave as in the low load regime.

While traffic matrices for $16:55$ June 1st and $16:55$ June 2nd push the network to experience the $3$ load regimes with traffic scale $5$, the figures for $21:35$ June 1st and $21:35$ June 2nd show a linear increase in median link utilization and decrease in residual max-flow. The reason is that the traffic matrices taken at $21:35$ correspond to relatively low traffic, compared with the ones taken at $16:55$ which correspond to peak time. With the same scale of traffic load, where traffic matrices for $16:55$ June 1st and $16:55$ June 2nd already push the network to the high load regime, the network under traffic matrices for $21:35$ June 1st and $21:35$ June 2nd stays in the low load regime.

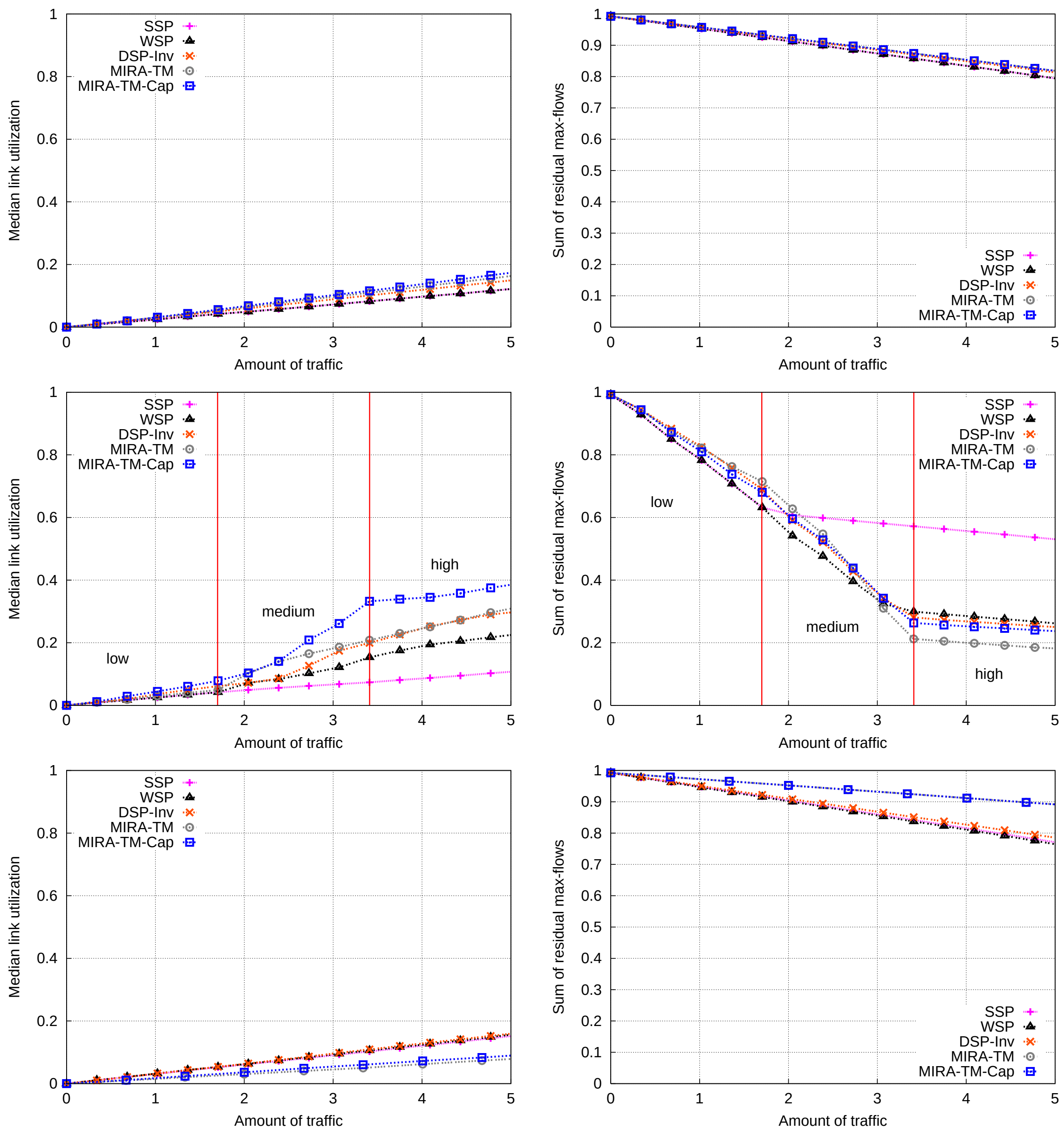


Figure 3.7: Performance of traffic engineering algorithms under different traffic matrices of Abilene. $21:35$ June 1st (top), $16:55$ June 2nd (middle) and $21:35$ June 2nd (down).

## 3.2.5 Traffic engineering algorithms and load regimes

From the results of this section, we identified one main factor that affects the behavior of the traffic engineering algorithms: the total amount of traffic the network has to handle. This total amount of traffic translates, through the choice of the paths made by the routing algorithms, into a load regime. We are now in a position where we can loosely define the three load regimes we observed in this section. The low load regime is defined by a behavior of the traffic engineering algorithms that is similar to the one of SSP. In this situation, it is not necessary to use non-shortest paths to accept the connection requests. As soon as SSP is not able to satisfy some connection requests, we enter the "medium load regime". In this medium load regime, traffic engineering algorithms manage to better use the available capacity than SSP, to accept connection requests that would be blocked otherwise. Finally, when the amount of traffic is so high, that even traffic engineering algorithms are unable to satisfy connection requests, we enter the "high load regime". In this high load regime, the solution is to increase the capacity of the links or to add redundant links.

## 3.3 Importance of network design aspects

Section 3.2 showed the differences between traffic engineering algorithms under different load regimes. As we relied on real-world networks and traffic demands, we could not pinpoint the importance of specific aspects of network design on the behavior traffic engineering algorithms. In this section, we build up scenarios to study the impact of the network topology and network dimensioning. We propose scenarios based on differently connected topologies, with different ways of assigning link capacities and generating traffic demands.

**Topology** Our topologies are generated using the iGen topology generator [74]. iGen allows to generate random points in one or any continent, and then to connect the nodes using network design heuristics [19]. It can also set capacities of the links and IGP link weights (based on physical distance or the inverse of the link capacity). We choose topologies with 25 nodes, and randomly generated points in Northern-America.

**Link weights** The weight for each link $l_{ij}$ is assigned as a piecewise linear function of the geographical distance between node $i$ and $j$.

**Traffic demand and capacity provisioning** For each topology, two combinations of traffic demand and capacity provisioning are considered:

- *Properly provisioned networks:* We generate the traffic demand based on a gravity model [51]. First, we generate for each edge node $i$ (source or destination) a total traffic demand $x_i$ following a uniform distribution. The amount of traffic $X_{ij}$ from node $i$ to node $j$ is set proportional to the product of the traffic demands $x_i$ and $x_j$, i.e., $X_{ij} = \beta x_i x_j$, where $\beta$ is some constant. We assign capacities to links by first assigning paths to source-destination pairs. Paths are obtained by running a shortest path using the already assigned link weights. Once a link is used by the path from node $i$ to $j$, the capacity of this link will be increased by $1.1$ times the corresponding traffic amount

$X_{ij}$. The factor $1.1$ provides a very small over-provisioning. When links are not used by any shortest path, they would end up with $0$ capacity. We assign to these links the average capacity of the non-empty links.

- *Improperly provisioned networks:* The traffic and link capacities are considered separately. All links are assigned the same capacity. All source-destination pairs have the same total amount of traffic, but connection requests have a uniform size between $1$ and $3$ units of traffic.

In the *properly provisioned networks*, the capacities of links are assigned in such a way that the provisioning of the network matches the traffic matrix, under the assumption that shortest paths are used. In *improperly provisioned networks*, the network capacities are not designed to match the traffic demand at all. We thus expect different behaviors of the traffic engineering algorithms under the two types of networks.

### 3.3.1 Minimally-connected topology

We start by relying on the worst possible connectivity a network can have: a tree. We use the MST (minimum spanning tree) heuristic from iGen to generate the topology. There are $24$ links in this topology. All the algorithms perform in the same way (figures not shown), whether or not the network is properly provisioned. For any source-destination pair, only one path is physically available, so all algorithms have no choice but to choose the same path.

### 3.3.2 Well-connected topology

Real-world topologies are typically designed to have minimal cost, while being able to stand any link failure. Several methods exist to generate such graphs [19]. We use the Two-Trees heuristic from iGen to generate a graph made of two link-disjoint MSTs. Such a graph remains connected after any single link failure. At least 2 paths exist between any pair of nodes. The topology has $48$ links.

#### Properly provisioned network

Figure 3.8 shows the median link utilization and the residual max-flow after each connection request on the properly provisioned Two-Trees topology. There is not much difference between the traffic engineering algorithms, both in terms of the median link utilization and the sum of the residual max-flows. As the network provisioning matches the traffic matrix, all algorithms manage to reach a high median link utilization ($0.93$) when the demand equals the network capacity (amount of traffic = $1$). It is interesting to note that both SSP and WSP have a linear increase (resp. decrease) of the median link utilization (resp. sum of residual max-flow) until the network is fully loaded. As SSP and WSP use shortest paths that better match

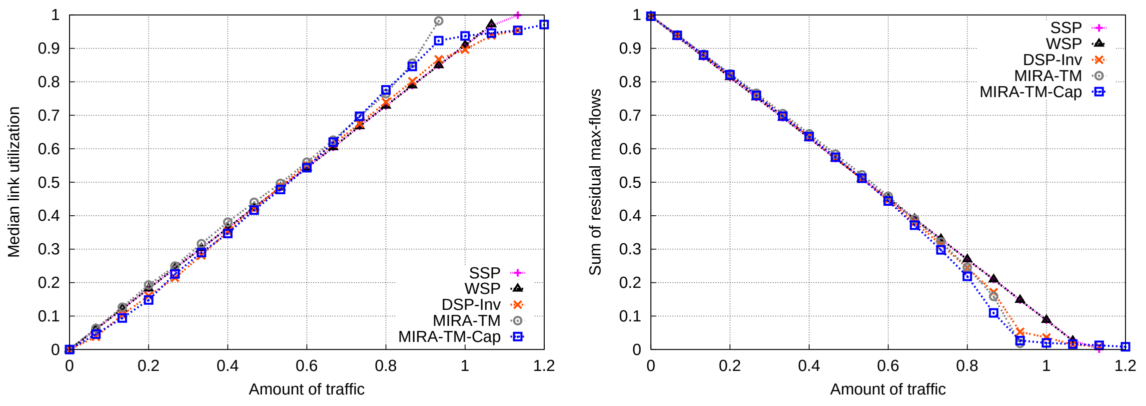


Figure 3.8: Median link utilization (left) and network-wide residual max-flow (right) in properly provisioned Two-Trees topology.

to the way the network was provisioned, their behavior is more appropriate in this network than more complex traffic engineering algorithms.

## Improperly provisioned network

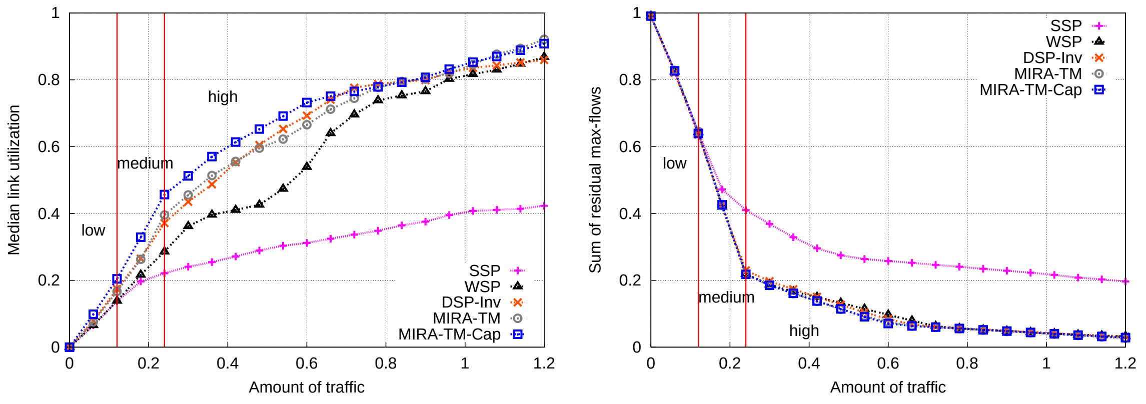


Figure 3.9: Median link utilization (left) and network-wide residual max-flow (right) in improperly provisioned Two-Trees topology.

Figure 3.9 shows the median link utilization and the residual max-flow after each connection request on the improperly provisioned Two-Trees topology. In well-provisioned networks, traffic engineering algorithms are hardly useful. In networks that have not been provisioned in such a way as to match the traffic demand, the typical low-medium-high load regimes appear quite early, when the median utilization is rather low.

In the low load regime, all algorithms give the same value of the sum of residual max-flow (right of Figure 3.9). In the medium load regime, SSP blocks connection requests while all other algorithms manage to find feasible paths in the network.

### 3.3.3 Highly-connected topology

If local redundancy is required in a topology, any three nodes close to each other can be connected by a triangle. The Delaunay triangulation procedure ensures such a connectivity, leading to locally well-connected graphs. Using the Delaunay heuristic in iGen to connect the 25 nodes, we obtain a topology with 65 links.

#### Properly provisioned network

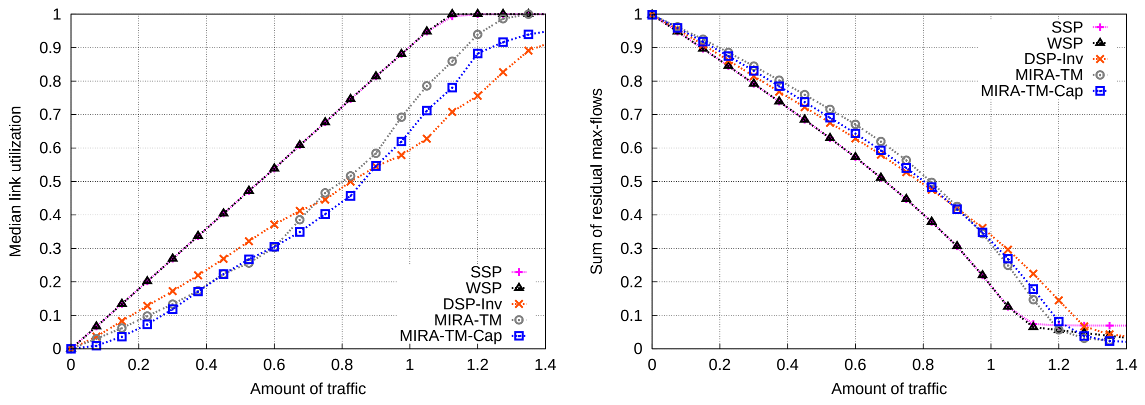


Figure 3.10: Median link utilization (left) and network-wide residual max-flow (right) in properly provisioned Delaunay topology.

As this topology provides much redundancy, when assigning link capacities in the properly provisioned scenario, some links that are never used by the shortest paths are assigned a capacity within the same scale as other links (their average). As shown in Figure 3.10, SSP and WSP have a median link utilization that increases linearly with the amount of traffic until it reaches a value of 1, faster than the other algorithms. DSP-Inv and the MIRA algorithms manage to leave a higher residual max-flow than SSP and WSP. The good provisioning and link redundancy allows all algorithms to use available capacity even when the link utilization becomes high. Contrary to the well-provisioned Two Trees topology, the redundancy of the Delaunay topology allows algorithms to choose very different paths, hence the differences between the algorithms.

#### Improperly provisioned network

Figures 3.11 shows the median link utilization and network-wide residual max-flow after each connection request on the improperly provisioned Delaunay topology. In the low load

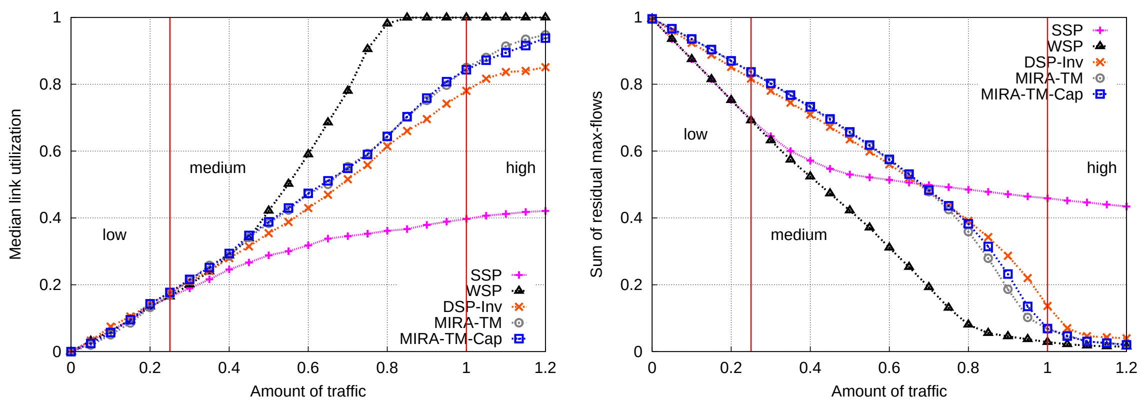


Figure 3.11: Median link utilization (left) and network-wide residual max-flow (right) in the improperly provisioned Delaunay topology.

regime, DSP-Inv and the two MIRA variants manage to route the requests while leaving a larger residual max-flow than SSP and WSP. SSP keeps a low link utilization at the cost of blocking connection requests, while WSP has the highest median link utilization not to block requests. DSP-Inv and the two MIRA variants manage to both keep a relatively low median link utilization while leaving a residual max-flow larger than WSP.

### 3.3.4 Discussion

In Section 3.2 we studied the behavior of traffic engineering algorithms on two real-world networks: Abilene and GEANT. Those two networks have different topological structure, as well as different strategies to set up their IGP weights. The Abilene network is very close to a Minimum Spanning Tree, with 15 undirected links for 12 nodes. The Abilene network has been designed to have minimal number of links, while still provides necessary redundancy. It is 2-connected, i.e. any single link failure still leaves the network connected. Abilene nodes have node degree 2 or 3. The GEANT network on the other hand is closer to a Two Trees topology than an MST, with 36 undirected links for 22 nodes. GEANT nodes have node degree from 2 to 8, so this network has been designed with far more redundancy than Abilene. Both Abilene and GEANT are over-provisioned networks, the link utilization is pretty low. Over-provisioned is not equivalent to what we call "properly provisioned" in this section. Proper provisioning means that the capacity of the links in the network match the traffic between all source-destination pairs. In Section 3.2, we observe that when pushing these networks to their limit[3], they behave like improperly provisioned networks. It is unlikely that these networks relied on capacity dimensioning to match the traffic demand. Instead, they probably use a simple over-provisioning strategy.

[3]We expect that both Abilene and GEANT assume that their traffic should not grow in that way in practice.

## 3.4 Conclusion

In this chapter, we studied the behavior of on-line traffic engineering algorithms under different scenarios. We first relied on two real-world networks, Abilene and GEANT, and their traffic. By scaling their traffic demand, we observed the behavior of traffic engineering algorithms when networks are pushed to their limits. We identified three distinct load regimes (low, medium, and high) that correspond to different behaviors of the traffic engineering algorithms. In the low load regime, traffic engineering algorithms do not provide much benefit compared to shortest-path routing. In the medium load regime, where shortest-path routing blocks some connection requests, traffic engineering algorithms manage to better use the available capacity in the network. Finally, the high load regime corresponds to a situation where not enough capacity is left in the network, so traffic engineering algorithms cannot route connection requests. In high load regime, network capacity or redundant links should be added.

In the second part of the chapter, we studied the behavior of traffic engineering algorithms using synthetic network topologies and traffic demands. We compared the behavior of each type of topology under two network provisioning scenarios. In the first scenario, we provisioned the link capacities as to match the traffic demand. In the second scenario, we provisioned the network without taking into account the traffic demand. As long as the network is properly provisioned, traffic engineering algorithms do not provide a significant improvement in using the available capacity, even in highly-connected topologies where many paths are available. When network provisioning does not match the traffic demand, traffic engineering algorithms are able to use the redundant links to compensate for the poor provisioning.

We expect that most real-world networks are in a low load regime, where shortest-path routing is good enough. However, as the traffic demand evolves, networks need be updated. One potential further work is to develop a metric that quantifies how badly the current network does not match the current traffic demand. This metric would take into account the network growth and the routing algorithm used, and give insight into when the situation is becoming critical enough so that either traffic engineering algorithms should be used, or when the network must be upgraded. We also expect that the topological structure has a non-trivial impact on the behavior of the traffic engineering algorithms.

# Part II

# Interdomain Traffic Engineering and QoS

# Chapter 4

# Interdomain routing – the business involved reality

In this chapter, we provide background information that will be revisited in Chapter 5 and 6. We introduce how interdomain routing works, and explain briefly how interdomain traffic engineering can be done by tuning parameters involved in route advertisement and selection.

## 4.1 Business relationships between ASs

To get reachability to the whole Internet, ASs connect to each other through peering relationships. When two ASs are interested to connect to each other, they negotiate and sign a contract that will rule the conditions under which the connectivity between them is to be used, such as how traffic will be exchanged, how the billing will be made, *etc*. A peering relationship between two ASs consists in a direct connection between them, either at a private location or a public exchange point. One or several physical links connect their border routers, and the border routers need to be configured consistently to the agreements of the peering contract.

Each interdomain link, which connects two ASs, can be classified as one out of three types of business relationships, namely customer-to-provider, provider-to-customer and peer-to-peer. Large ASs provide transit service to their customer ASs. In addition, they may peer with each other to transit traffic between each other's customers with shared costs and mutual benefits. Small ASs need to pay for the transit service provided by larger ASs to have Internet-wide reachability. Different ways to infer those business relationships have been proposed in the literature [11, 28, 38, 105]. These techniques infer simplified business relationships [18], but modeling the reality of business relationships and routing policies based on the available data is very complex [61]. Note that inferred business relationships are typically blind to multiple physical links between two ASs.

## 4.2 The Border Gateway Protocol (BGP)

The Internet is using the Border Gateway Protocol (BGP) as the interdomain routing protocol. The role of BGP is to propagate reachability information across the whole Internet. The way routing information is exchanged between BGP routers and paths are selected by BGP both depend on routing policies [38]. Routing policies [18] implement the business objectives (see Section 4.1) of each ISP by modifying the flow of routing information and the path selection made by BGP. As will be explained in this section, two factors influence the distinct routes known by a BGP router: business relationships and the best path selection of BGP.

Figure 4.1 illustrates how BGP selects the best paths and propagates reachability information. For each destination prefix, i.e. a block of reachable IP addresses, a BGP router receives from its neighboring BGP routers their best AS paths. An AS path is a sequence of intermediate ASs which form a direct route for traffic from the source to reach the destination. For each prefix, a BGP router selects its best path in the following manner:

- it discards the AS paths having its own AS number to avoid loops,
- it filters the AS paths according to the import policies. That is, a router keeps the AS paths which pass the import filters in the RIB-Ins, (see Figure 4.1, $R1$ only keeps the routes learned from $R10$, $R12$ and $R13$)
- it selects from the AS paths in the RIB-Ins its best AS path (the route learned from $R12$ in Figure 4.1) according to the BGP decision process.

RIB-In stands for Routing Information Base-Inbound, and contains all the valid routes that passed the import filters. The routes that are stored in the RIB-Ins then undergo the BGP decision process, that selects a single route towards any destination prefix. As shown on Figure 4.1, the BGP decision process is composed of a list of criteria (called "rules"). It aims at choosing among a given set of routes towards a given destination, a single route, called "best" route. Each criterion of the decision process will reduce the set of candidate routes, by selecting only those that have the best value of the considered attribute. We present here a simplified version of the BGP decision process, with 7 rules. For details about the BGP decision process, we refer to [108].

After selecting the best route, the BGP router attaches the AS number of its AS to the AS path and propagates this AS path to its neighboring BGP routers according to the export policies (outbound filters). In Figure 4.1, the route is not advertised to $R31$. Although the BGP router may learn many AS paths from its peers, **only one** is selected as the best AS path and used to forward traffic towards each destination prefix. Moreover, **only one** is advertised to the neighbors.

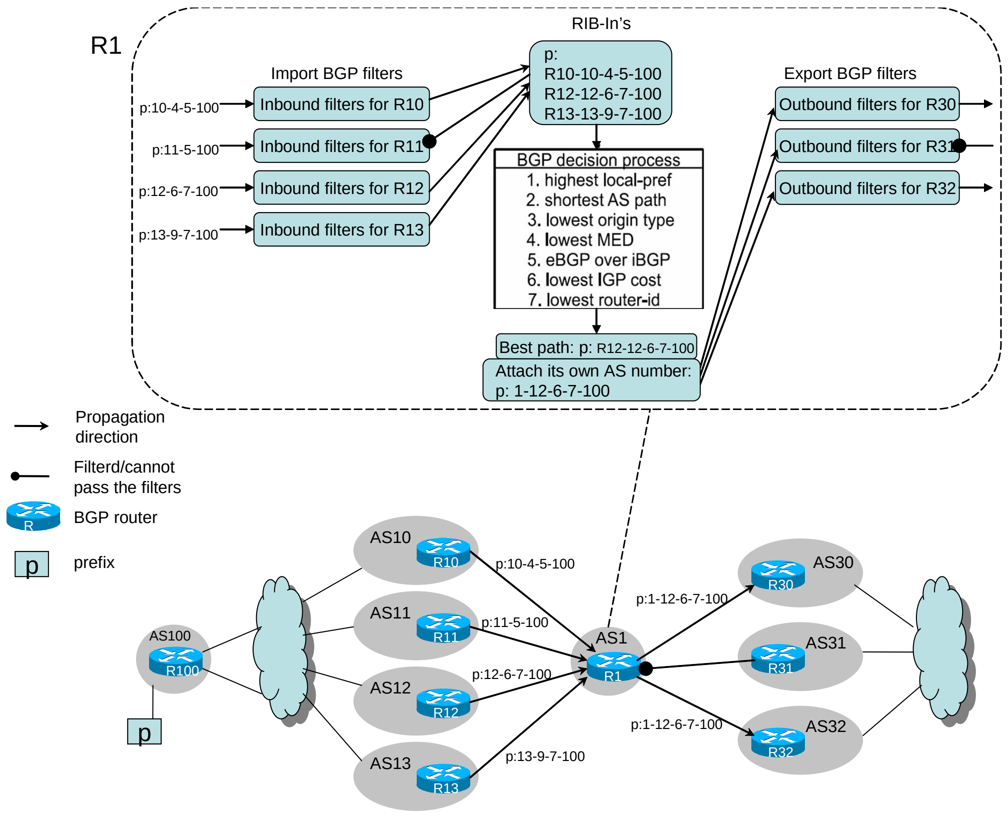


Figure 4.1: BGP path selection and propagation.

## 4.3 Routing policies

Routes learned from different types of peers are typically considered differently. For example, a route learned from a customer peer will be preferred to a route received from a peer, and finally a route learned from a provider is least preferred. The reason for doing this is the cost associated with the use of a neighbor to forward traffic. That is, an AS receives money to provide connectivity to its customers, shares the cost of the traffic exchanged with its peers, and pays its providers to send traffic through them. This preference among differently learned routes can be transferred into a value of the `local-pref` attribute of the BGP routes. As shown in Figure 4.1, the `local-pref` attribute is used in the first step of the BGP decision process. The value of the `local-pref` attribute will typically be assigned to a route upon reception of a route from a peer.

Business relationships also introduce constraints on BGP route propagation, e.g., the so called valley-free route propagation property [38]. This property is verified [61] because ASs typically want to avoid propagating routes which result in using themselves to transit traffic between two larger ASs [32]. An AS propagates to its providers its routes, the routes learned from its customers, but not the routes learned from its other providers or peers. An

AS propagates to its peers its routes, the routes learned from its customers, but not the routes learned from its providers and other peers. An AS propagates to its customers its routes, the routes learned from its customers, providers and peers.

## 4.4 Interdomain traffic engineering with BGP

As the connections between ASs are built up based on their business agreements. The amount of traffic that flows on each interdomain connection will leads the money flowing from one AS to another. Thus, besides engineering the packets flowing inside their networks, ISPs also need to take the flow of their interdomain traffic into account. ASs engineer their interdomain traffic based on their business interests. For some ASs, reachability is the most important concern and their purpose of interdomain traffic engineering is to get traffic forwarded with the lowest possible cost. For ASs providing specific services such as those content distribution networks, end-to-end performance of the selected routes will influence the quality of the service they deliver, and thus, is crucial for these ASs. They have the aim of interdomain traffic engineering as to use the forwarding routes with good end-to-end performance.

With the configurable parameters involved in the BGP path selection process, ASs can influence the incoming and outgoing traffic by tuning the BGP configurations and route advertisements on their BGP routers.

### 4.4.1 Controlling outbound traffic

To control how traffic leaves the AS's network, is actually for the AS to control the selection of routes that will be used to forward traffic towards each destination prefix. As shown in Figure 4.1, for each destination prefix, the BGP router will select a best route out of the many learned ones according to the BGP decision process. The parameters (such as the local-pref value) used in the BGP decision process can be configured as fine as on a per-prefix basis, which makes the per-prefix parameter configuration (i.e., route selection) possible. Outbound traffic engineering can be realized by tuning the parameters used in the BGP decision process. E.g., for a destination prefix, if a certain route is desired, the network administrator can assign a high enough local-pref value to this route to ensure that it is preferred to all other learned routes and thus will be selected by the BGP decision process.

Different ASs may have different aims of engineering their outbound traffic. Some multi-homing stubs have a primary provider as well as a backup provider. They prefer to use the primary provider to forward traffic due to the cheap billing this provider asks, and only use the backup provider when necessary (e.g., when the primary provider is down). Some other ASs would load-balance their traffic among several peering ASs with different load-balancing purposes. E.g., ASs may want to keep their external links' utilization low due to the limited link capacity each external link provides; ASs may select routes according to their end-to-end performance; they may want to minimize the money they need to pay to get the traffic forwarded; *etc*. For this aim of load-balancing, preference of routes on a per prefix-group basis needs to be configured.

### 4.4.2 Controlling inbound traffic

With BGP as the interdomain routing protocol, the router sending packets has the rights to decide the forwarding route, although this router can only select the route from those learned from its neighbors. The control of inbound interdomain traffic is not so easy as that of outbound interdomain traffic, due to the ownership of the route selection rights. But ASs can still influence to some extent their inbound interdomain traffic.

A simple example is selective advertisement as shown in Figure 4.2. The origin AS originates two prefixes $p_1$ and $p_2$. It wants to receive traffic towards $p_1$ via border router $R11$, and traffic towards $p_2$ via border router $R12$. This goal can be reached by advertising the two prefixes on different external links.

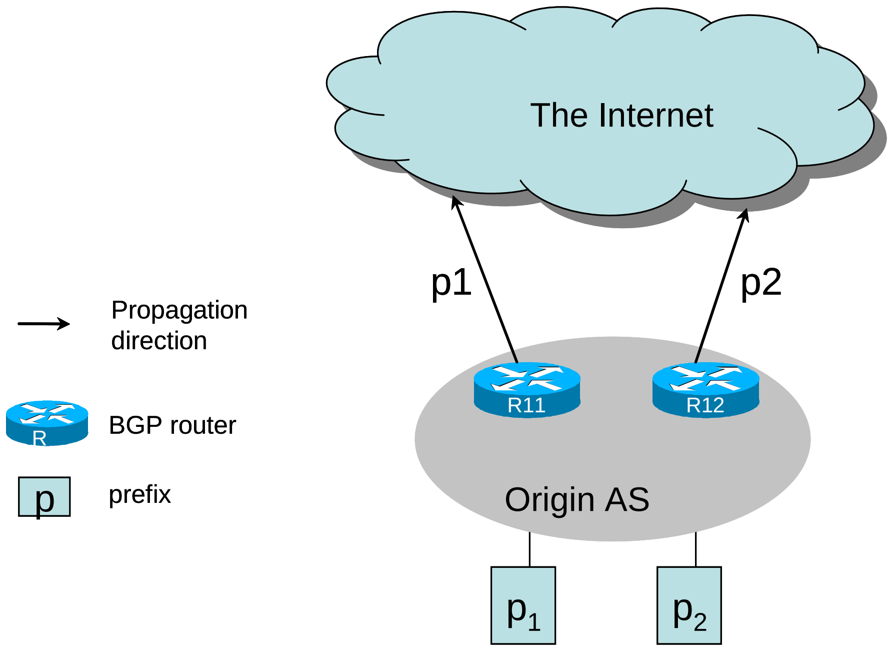


Figure 4.2: Selective advertisement.

ASs may use the second step (AS path length) in the BGP decision process to try to influence the route choices of remote ASs. They increase the AS path length by prepending their own AS numbers several times. This action will affect the path choice if the second step (AS path length comparison) in the BGP decision process determines the best route. On the other hand, if the local-pref value were deterministic, the prepending would be helpless.

In some business agreements, the customers require finer control on the redistribution of their routes by the providers, and the providers offer the customers this authority of control as a service. The control of route advertisement carried by upstream providers can be realized by using the BGP community attribute. Communities can be used as flags in order to mark a set of routes. Routers in the providers can then use these flags to apply specific routing polices/actions within their network. E.g., the customers may require the providers to prepend the AS path when advertising the route to a specified set of peers.

# Chapter 5

# AS-level connectivity and policy granularity

Mühlbauer, W., S. Uhlig, B. Fu, M. Meulle, and O. Maennel, 2007, "*In search for an appropriate granularity to model routing policies*", in Proc. of the ACM SIGCOMM conference, Kyoto, Japan, August.

So far, models of the network structure have been mostly interdomain level models that do not care about details of the ASs [38, 58, 59]. However ASs are not simple nodes in a graph. Rather they consist of routers spanning often large geographic regions. The internal structure of an AS *does* matter. It influences interdomain routing, for instance via hot-potato routing [91, 93]. Further, there are multiple connections between ASs, typically from different routers in different locations, which adds to the diversity of known routes [33, 92].

Besides the impact of the interdomain topology, interdomain routing is controlled by diverse policies, decided locally by each AS, and is not directly observable from available BGP data. Those policies act globally across the entire system [39]. Hence the topology of the interdomain graph is not, in itself, enough to model the reality of interdomain routing. Policies also need to be considered to capture the reality of the path choices made by each AS.

In this chapter, we provide an AS-level topology model with a similar approach to that used in [64], as we also build an AS connectivity graph that enables the propagation of all routes present in the observed BGP paths. Our model differs from the one used in [64] in that we try to minimize the total number of interdomain links.

Further in this chapter, we study the granularity of routing policies in the Internet by looking into the choice of paths each AS makes as observed from BGP data from multiple vantage points. Policies are not easily defined [18] as they encompass the business and engineering decisions made by each AS, both on commercial agreements (business relationships) and on technical aspects (router configuration, interdomain routing behavior, *etc.*). We introduce a new concept: *next-hop atoms*. Next-hop atoms capture the different sets of neighboring ASs that each AS chooses for its best routes. We show that a large fraction of next-hop atoms correspond to per-neighbor path choices. A non-negligible fraction of path choices, however, correspond to hot-potato routing[1] and tie-breaking[2] within the BGP decision process, very detailed aspects of Internet routing.

The remainder of this chapter is structured as follows. Section 5.1 introduces the BGP data used for building up the AS-topology model and investigating the granularity of routing policies. Section 5.2 presents our AS-topology model. Section 5.3 analyzes the known bounds for policies studied in the literature. Section 5.4 discusses the difference between routing policies and path choices. In Section 5.4 we come up with a new abstraction that captures the selection of paths by ASs: next-hop atoms. Section 5.5 shows how ASs select their next-hops in a dynamic scenario based on a set of BGP dumps and updates. The related work is described in Section 5.6. Finally, Section 5.7 concludes this chapter and discusses further work.

## 5.1 Data

Different techniques exist to collect BGP feeds from an AS. One of the most common techniques is to rely on a dedicated workstation running a software router that peers with a BGP router inside the AS. We refer to each peering session from which we can gather BGP data as an *observation point*, and the AS to which we peer as the *observation AS*.

We use BGP data from more than $1,300$ BGP observation points, including those provided by RIPE NCC [1], RouteViews [81], GEANT [45], and Abilene [2]. The observation points are connected to more than $700$ ASs, and in $30\%$ of these ASs we have feeds from multiple locations.

We use a static view of the routes observed at a particular point in time to build up our AS-topology model and to investigate the granularity of routing policies. The table dumps provided by the route monitors are taken at slightly different times. We use the information provided in these dumps regarding when a route was learned to extract those routes that were valid table entries on Sun, Nov. 13th 2005, at $7:30$am UTC and have not changed for at least one hour.

Our dataset contains routes with $4,730,222$ different AS paths between $3,271,351$ different AS-pairs. An AS-level topology is derived from the AS paths. If two ASs are next to each other on a path we assume they have an agreement to exchange data and are therefore neighbors in the AS-topology graph. We are able to identify $21,159$ ASs and $58,903$ AS-level edges.

[1]When multiple equally-good BGP routes exist, the router tends to select the one with the closest egress point, based on the intradomain path cost (see Figure 4.1, step 6 in the BGP decision process).

[2]If multiple routes are equivalent when comparing BGP attributes, the router decides which route to pick based on a tie break such as the router ID (see Figure 4.1, step 7 in the BGP decision process).

## 5.2 AS-topology model

To study the granularity of policies, we need a topology model of the Internet. Section 5.2.1 describes some properties of the AS connectivity observed in the data described in Section 5.1, and precedes our explanation in Section 5.2.2 of how the AS graph of our model is built from the observed paths.

### 5.2.1 AS-level connectivity

As already shown in [58, 64], for an AS topology model to capture route diversity, ASs cannot be considered atomic entities. In order to represent the intradomain routing diversity, we allow each AS to consist of multiple quasi-routers. A *quasi-router* represents a group of routers within an AS, all making the same choice about their best routes. Thus the "quasi-router topology" does not represent the physical router topology of a network, rather the logical partitioning of its observed path choices. An AS has to be modeled with multiple quasi-routers if it receives and chooses as best multiple paths towards at least one prefix.

Figure 5.1 provides the number of quasi-routers per AS that are required to capture BGP path diversity. In the data analysis results done with the snapshot of Internet routing in this chapter, we do not consider stub ASs, i.e., ASs that appear as the last AS hop on any AS path in our data (pure originating AS)[3]. Among the $3,535$ remaining ASs, $267$ require more than a single quasi-router. Only $2$ ASs need as many as $8$ and $9$ quasi-routers to account for their observed routing diversity. Typically, well-known tier-1 ASs require several quasi-routers. This is consistent with [92] which showed, based on active measurements, that tier-1 ASs have high path diversity. On the other hand, a low number of quasi-routers per AS is due both to the sampling of the available paths of the observed BGP paths, as well as the loss of BGP routing diversity inside ASs [97].

Diversity of the AS paths is strongly related to the AS-level connectivity. Figure 5.2, in which we consider the same $3,535$ ASs as in Figure 5.1, shows a scatterplot of the relationship between the number of required quasi-routers and the number of neighboring ASs. We observe that ASs that do not have many neighbors also tend to have a small number of quasi-routers. Highly connected ASs on the other hand may have many quasi-routers, although this is not necessarily always true. Some ASs have hundreds of neighbors, yet a single quasi-router is enough to account for their routing diversity. Note that some of the considered ASs have only one neighbor. These ASs should in practice provide at least partial transit since they are not pure origin ASs. Recall that we do not consider stub ASs for our data analysis and therefore removed these ASs and the incident links from the BGP data. It is likely that those transit ASs for which we see a single neighboring AS (after having removed the pure origin ASs) have more neighbors but which are not visible due to missing AS-level links from the BGP data.

[3]Although being transit domains, some ASs may only have one AS neighbor after removing stub ASs.

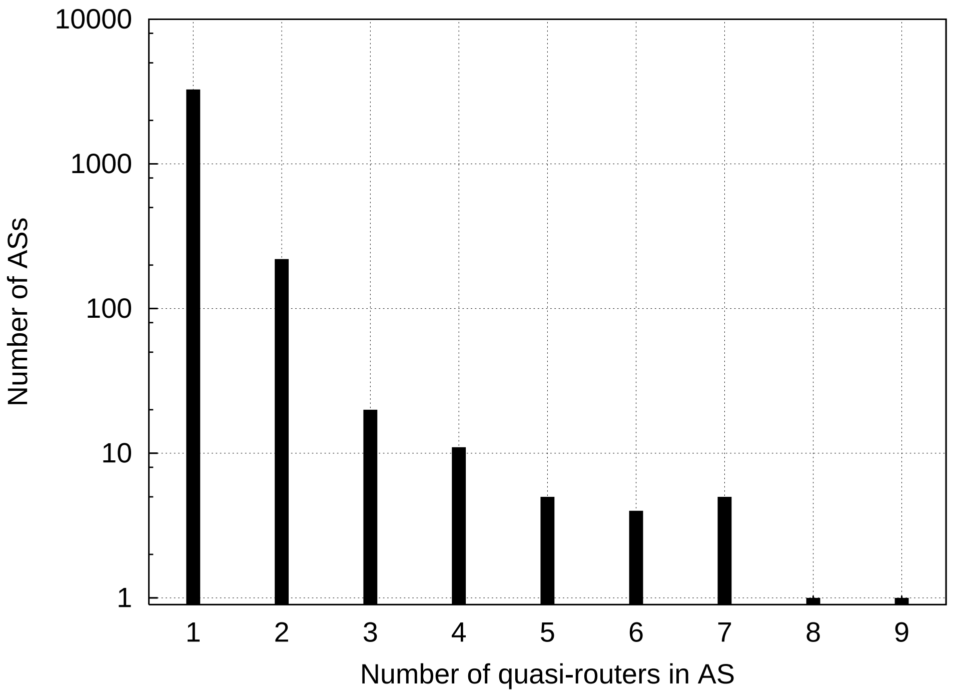


Figure 5.1: Number of quasi-routers per AS.

As previously stated, there are two reasons why an AS requires several quasi-routers: (i) the AS receives and selects as best multiple paths towards a given prefix from a given neighbor; and (ii) the AS receives and selects as best different paths towards a prefix but from different neighbors. From Figure 5.2, we can observe that highly connected ASs have a far larger number of neighbors than quasi-routers. ASs thus select a very small subset of best paths compared to the number of paths they may receive from their neighbors, for any prefix. Note that the first reason why an AS might need several quasi-routers does not seem to be common. For only 623 pairs of neighboring ASs do we observe in the data that an AS chooses from one of its neighbors more than one path towards at least one prefix. Further, in only 19 cases do we observe an AS receiving more than 2 distinct paths from a given neighbor towards at least one prefix.

### 5.2.2 Building a quasi-router-level graph

For our study of the granularity of routing policies we need a topology model of the Internet. We capture the interdomain connectivity via an AS-topology graph as extracted from the BGP data (see Section 5.1). In order to represent the intradomain routing diversity, we allow each AS to consist of multiple quasi-routers.

To ensure the connectivity of our model is minimal, the topology is built when assigning

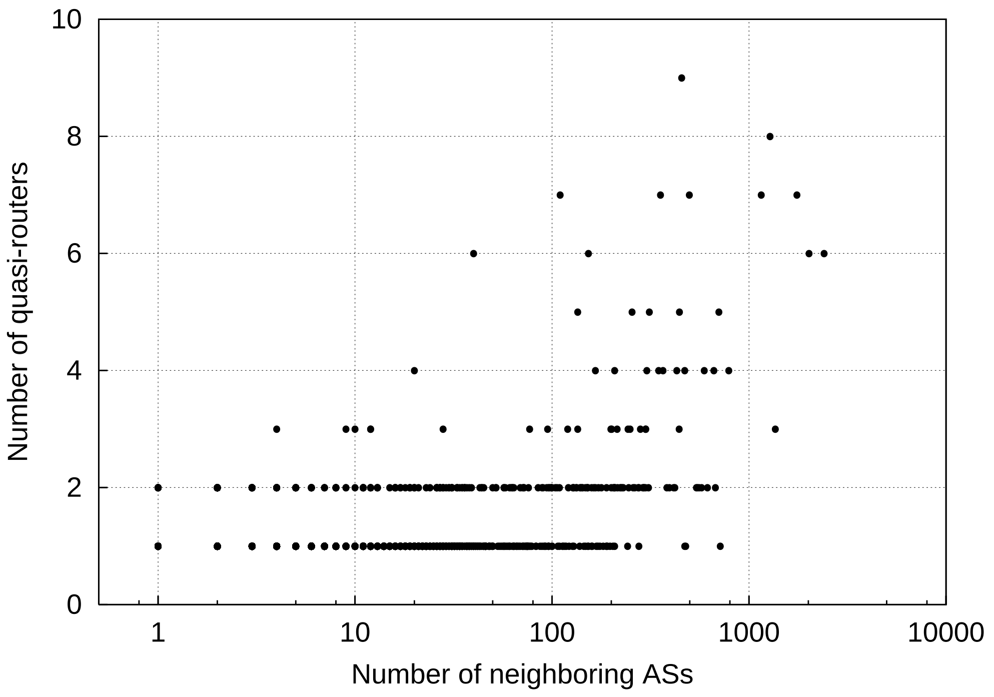


Figure 5.2: Relationship between number of neighboring ASs and number of quasi-routers.

to quasi-routers AS path suffixes observed in the data. A suffix $s$ of an AS path $P$ is any substring $Q$ such that $P = Qs$. The AS topology we create has as few quasi-routers per AS as possible, and an edge exists between two quasi-routers if some suffix has to be propagated between the two quasi-routers.

Before explaining how our topology is built, let us first introduce some notations:

- $AS^p$: ASs that can reach prefix $p$.
- $TBA\_SUFF^p_{as}$: suffixes to be assigned within AS $as$ towards prefix $p$.
- $A\_SUFF^p_{as}$: suffixes already assigned within AS $as$ towards prefix $p$.
- $SUFF^p_{as}$: suffixes AS $as$ has towards prefix $p$. $SUFF^p_{as} = TBA\_SUFF^p_{as} \cup A\_SUFF^p_{as}$.
- $TBA\_AS^p$: $as \in AS^p \mid TBA\_SUFF^p_{as} \neq \emptyset$.
- $A\_AS^p$: $as \in AS^p \mid TBA\_SUFF^p_{as} = \emptyset$.
- $QR_{as}$: quasi-routers of AS $as$.
- $FREE\_QR^p_{as}$: quasi-routers of AS $as$ that are free for prefix $p$.

- $DA\_SUFF^p_{as}$: suffixes such that $s \in TBA\_SUFF^p_{as} \wedge (reusableRouter(s,p) \neq \emptyset) \wedge (nextHop(s,p) \neq \emptyset)$.

- $NDA\_SUFF^p_{as}$: suffixes such that $s \in TBA\_SUFF^p_{as} \wedge (reusableRouter(s,p) = \emptyset) \wedge (nextHop(s,p) \neq \emptyset))$.

- $ASSIGNED^p_s$: quasi-router to which suffix $s$ towards prefix $p$ is assigned.

Figure 5.3 presents the pseudo-code explaining how our topology is built with the sub-functions presented in Figures 5.4, 5.5, 5.6, 5.7, 5.8 and 5.9. Our assignment works on a per-prefix basis. First we set all quasi-routers as free to be assigned paths towards the considered prefix, and set all suffixes towards this prefix as to be assigned. Then, as long as there are suffixes that are not assigned, we try to assign them by starting with those suffixes closest to the originating AS(s) of the prefix. When trying to assign suffixes, we first reuse existing connectivity between quasi-routers. If no link between the first two ASs on the suffix can be reused for this prefix, then we create a new link between a free quasi-router in the first AS on the suffix and the next AS. Note that the creation of the topology (links between quasi-routers) follows directly from the path assignment.

Let us come back for a while to the analysis of our data. The number of necessary quasi-routers in an AS is not the only parameter that matters for allowing an AS topology model to reproduce the paths observed in BGP data. Even though only few quasi-routers might be necessary to account for the routing diversity of an AS [64], the way in which quasi-routers between two ASs are connected also matters. If in general an AS requires the same number of interdomain links as it has neighboring ASs, it means that even though this AS might have many neighbors, only a single neighbor at a time is used as next hop AS for any prefix. If an AS in our model has much more interdomain links than neighboring ASs on the other hand, it means that the considered AS uses several neighboring ASs for its best routes towards some prefixes.

$3,150$ among the $3,535$ transit ASs of our data require a single interdomain link with any of their neighboring ASs. Only 386 ASs require more than one interdomain link per neighbor, and 41 ASs more than 2 interdomain links. As seen from BGP data, only a very small fraction of the ASs choose their best paths from several neighbors at the same time towards any of their prefixes.

Figure 5.10 gives the percentage of ASs (among the $3,535$ ASs considered previously) that have their *links to neighbors ratio* smaller than some value. This ratio is the number of links from all quasi-routers an AS has, divided by the number of neighboring ASs this AS has. We observe that most ASs have a ratio of one, indicating that they choose as best route no more than one path towards any prefix over their set of neighbors. Only $386$ ASs have a ratio larger than one, and $41$ larger than $2$. This means that, as seen by BGP data, only a very small fraction of the ASs choose their best paths from several neighbors at the same time towards some of their prefixes.

```
ALGORITHM TO ASSIGN PATHS TO Quasi-routers
foreach prefix $p$ {
    // Initialize set of ASs to be processed
    $TBA\_AS^p = AS^p$
    foreach $as \in TBA\_AS^p${
        // Initialize quasi-routers and unassigned suffixes
        $FREE\_QR^p_{as} = QR_{as}$
        $TBA\_SUFF^p_{as} = SUFF^p_{as}$
    } end foreach $as$
    while $TBA\_AS^p \neq \emptyset$ {
        $SORTED\_AS$ = sortAS($p$) {
        foreach $as \in SORTED\_AS$ {
            // Assign suffixes that can reuse existing connectivity
            foreach $s \in DA\_SUFF^p_{as}$ {
                assignSuffix($s$,$p$,reusableRouter($s$,$p$))
            } // end foreach suffix $s$
            // Assign suffixes that cannot reuse existing connectivity
            foreach $s \in NDA\_SUFF^p_{as}$ {
                assignSuffix($s$,$p$,findFreeRouter($s$,$p$))
                // Create connectivity between selected free router and next hop
                createLink(findFreeRouter($s$,$p$),nextHop($s$,$p$))
            } // end foreach suffix $s$
        } // end foreach sorted $as$
    } // end while $as$ to be assigned for $p$
} // end foreach prefix $p$
```

Figure 5.3: Building up the AS connectivity with minimal number of quasi-routers.

```
sub sortAS(prefix $p$)
    Sort the ASs $\in TBA\_AS^p$ by increasing distance $dist(as, p)$ towards the originating
        AS(s) of prefix $p$.
    $dist(as, p)$: number of AS hops of the shortest suffix $as$ has towards prefix $p$.
    returns list of sorted ASs.
```

Figure 5.4: Subfunction: sortAS.

## 5.3 Bounds on policy granularity

To find an appropriate way to model policies in the Internet, it is important to start with realistic bounds that define the finest and coarsest granularities at which policies are applied in the Internet. There are two ends to this spectrum. The finest granularity is the one of BGP

```
**sub** assignSuffix(suffix $s$, prefix $p$, quasi-router $r$)
    $TBA\_SUFF^p_{as} = TBA\_SUFF^p_{as}/s$
    $A\_SUFF^p_{as} = A\_SUFF^p_{as} \cup s$
    $FREE\_QR^p_{as} = FREE\_QR^p_{as}/r$
    $ASSIGNED^p_s = r$
```

Figure 5.5: Subfunction: assignSuffix.

```
**sub** reusableRouter(suffix $s$, prefix $p$)
    Returns quasi-router $r$ (in first AS on suffix) s.t. there is a link between $r$ and nextHop($s$,$p$).
    If no such quasi-router exists return $\emptyset$.
```

Figure 5.6: Subfunction: reusableRouter.

```
**sub** nextHop(suffix $s$, prefix $p$)
    **Returns** the quasi-router that stores suffix $s'$ towards $p$, with $s = as.s'$
        and $as$ is the first AS on suffix $s$.
    If no such quasi-router exists, **return** $\emptyset$.
```

Figure 5.7: Subfunction: nextHop.

```
**sub** findFreeRouter(suffix $s$, prefix $p$)
    **Returns** any quasi-router $r \in FREE\_QR^p_{as}$.
```

Figure 5.8: Subfunction: findFreeRouter.

```
**sub** createLink(quasi-router $r$, quasi-router $q$)
    Creates a interdomain link between $r$ and $q$.
```

Figure 5.9: Subfunction: creatLink.

atoms [3, 17], which are sets of prefixes originated by a given AS that receive equivalent treatment by routers in the Internet. BGP atoms are as fine as the set of policies that the observed BGP paths encounter, which can be as fine as on a per-prefix basis. The coarsest granularity does not depend on the originated prefixes, but only on the neighbors from which routes are received. It is the granularity of business relationships. ASs may configure policies as coarse as per-neighboring AS, hence treating all prefixes, received from a given neighbor, in the same way.

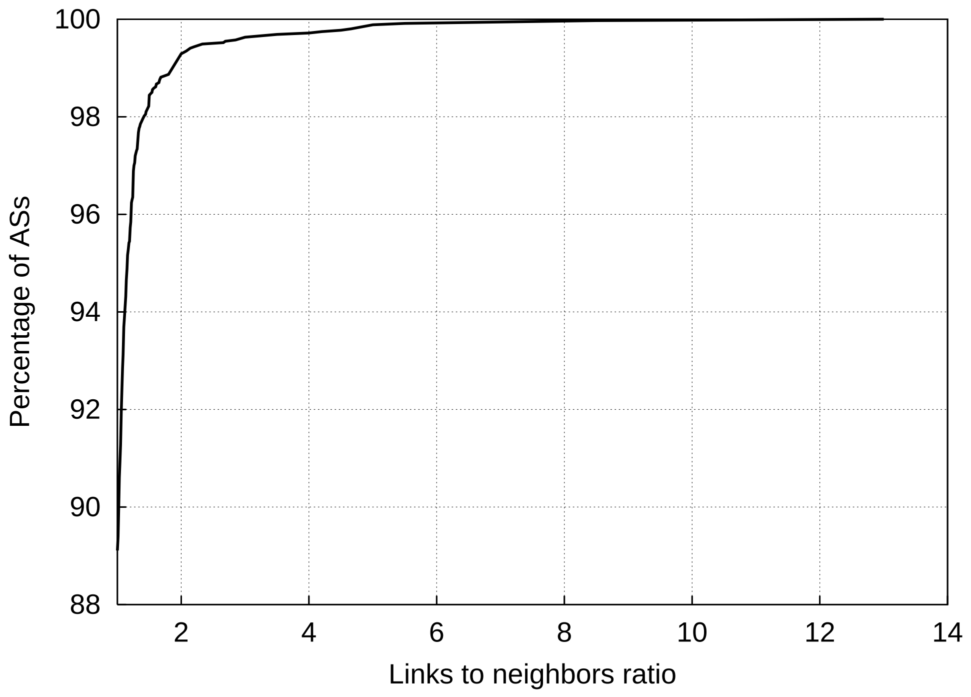


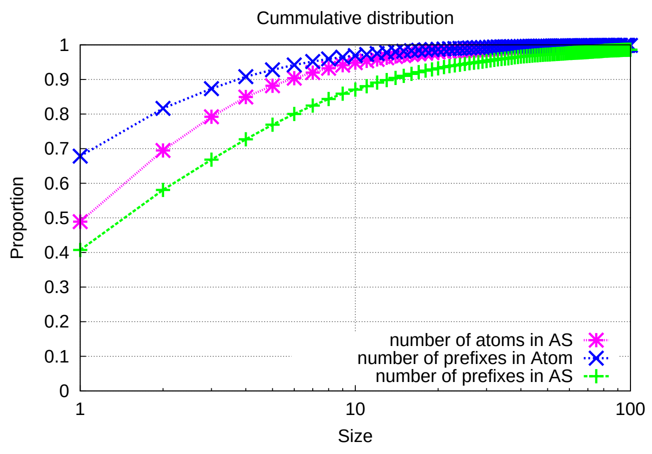
Figure 5.10: Links to neighbors ratio.

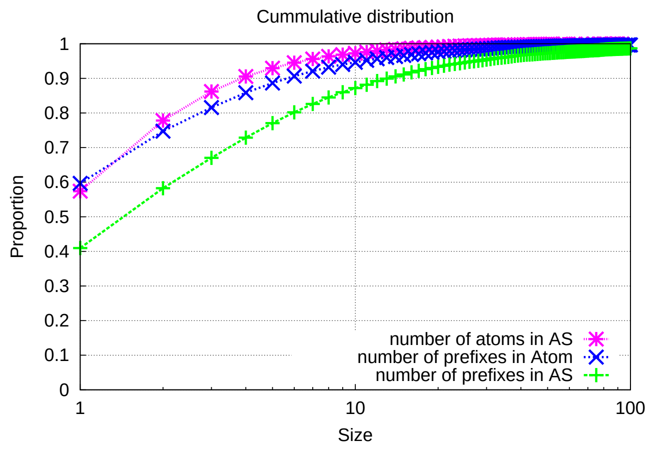
### 5.3.1 BGP atoms

For interdomain routing, each prefix is handled independently from other prefixes. However, groups of prefixes may receive equal treatment by a given set of BGP routers, due to the granularity of routing policies. The analysis of BGP routing tables has shown that clusters of prefixes originated by given ASs undergo the same routing policies [3, 17]. Groups of prefixes (originated by a given AS) that receive *equivalent treatment* by a set of BGP routers are called *BGP atoms* [3, 17]. As BGP monitors see only a sample of the outcome of routing policies through observed AS paths, not the policies themselves, a *BGP atom* is defined as a set of prefixes that share the same set of AS paths as seen from a set of BGP routers [17]. Two prefixes are put in the same BGP atom if their `AS-PATH` is the same, as seen by all observation points. The finest granularity of a BGP atom is a single prefix, whereas the coarsest is all the prefixes originated by an AS (or a set of origin ASs in the case of MOAS prefixes [112]).

Atoms' sizes vary across and within origin ASs. A large fraction of atoms consist of a single prefix, while some atoms consist of tens of prefixes. As BGP atoms are defined with respect to a given set of vantage points, policies applied by single ASs might be coarser than per BGP atom.

We use a similar approach to [17] to compute atoms. As most of the observation points do





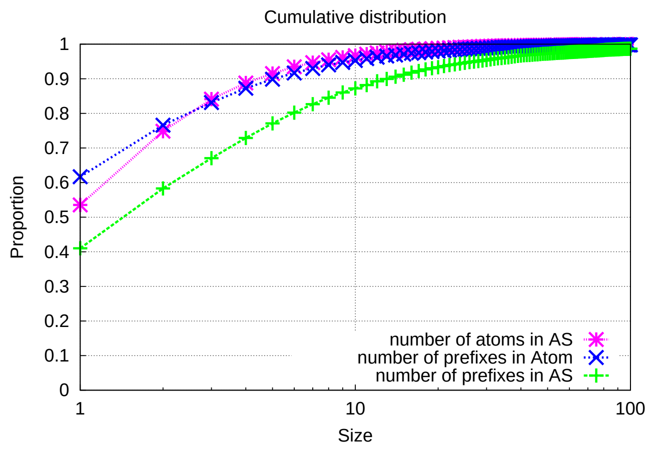


Figure 5.11: Atoms structure for dataset: full dataset (top), RouteViews subset (middle) and RIPE NCC subset (bottom).

not report paths towards all prefixes, we select the data given by those observation points that can see most of the prefixes (more than $160,000$). We consider these observation points as global enough. Before building the atoms, we skip the AS-PATHs with loops while keeping prepending as it is a policy. After this preprocessing of the dataset, we compute the atom structure by grouping the prefixes who have the same set of observation points and AS-PATH combinations as in [17].

Figure 5.11 compares the atoms structure of the full dataset (top) with the subset coming from RouteViews (middle) and RIPE NCC (bottom). Each graph displays three cumulative curves: the number of prefixes per origin AS, the number of prefixes in each atom, and the number of atoms in each AS. [3] that relied on RIPE NCC data shows similar curves as our bottom graph in Figure 5.11. From the three graphs shown in Figure 5.11, first we observe that the number of prefixes per origin AS is virtually the same for the three datasets. More than $40\%$ of the origin ASs advertise a single prefix. This means that more than $40\%$ of the atoms will be made of a single prefix. We observe in the curves giving the number of prefixes per atom that indeed $70\%$ of the atoms consist of a single prefix for the full dataset while about $60\%$ for the RouteViews and RIPE NCC subsets.

We believe that we observe finer atoms in the full dataset compared to [3] because our full dataset provides a more extensive coverage of the actual BGP paths. With an increasing number of paths, we also observe the effect of more policies, leading to smaller atoms due to more diverse path choices. The curves in Figure 5.11, showing the number of atoms in each AS, confirm that the full dataset used in this study sees ASs that have more atoms than that observed using only RouteViews or RIPE NCC data. In our full dataset, about $30\%$ of the ASs contain two or more atoms, whereas [3] observed only slightly more than $25\%$ of ASs with two atoms or more.

## 5.3.2 Business relationships

Business relationships are the most popular model for policies in the literature, and they rely on the coarsest granularity possible for policies: filtering rules defined on a per-neighbor basis. [61] simulates the path choices with our model, when the only policies configured are inferred business relationships. Customer-provider and peering relationships are inferred by applying the CSP algorithm [42] to the data (see the appendix of [61] for more details). Then, the paths chosen in the model with those observed in BGP data are compared.

They [61] rely on C-BGP [75, 76] to compute the outcome of the BGP decision process and the set of learned routes at every router of our AS-level topology model. C-BGP computes the steady-state choice of the BGP routers after the exchange of the BGP messages has converged. Their results [61] show that only $10.1\%$ of the paths agree between the simulation and the observations; for $60.9\%$ of the paths, the corresponding path is not even propagated to the AS that should observe that path in the simulations; and only $24.6\%$ of the paths are learned by the right AS but not selected as best path by any quasi-router of that AS.

The results are found to be disappointing. Introducing business relationships does not seem to solve any inconsistencies between the paths propagated in our model and the routes actually observed in the Internet. This does not mean that business relationships are "wrong". Quite the contrary, it is easy to justify them. Certainly, contracts between two ASs will reflect the economic status of peering partners. However, in the context of this section, the question arises if business relationships are too coarse-grained and to what extent having per-neighbor policies is responsible for high inconsistencies between paths propagated in our model and those we observe in the data.

### 5.3.3 Atoms vs. relationships

We believe that neither BGP atoms nor business relationships give an ultimate answer to the problem of which granularity should be used for modeling routing policies.

On one hand, business relationships appear too coarse, as they result in high inconsistencies between the paths propagated in our model and the routes actually observed in the Internet. We want to point out that this does not necessarily mean that business relationships are "wrong". However, it is unclear to what extent having per-neighbor policies is responsible for this high inconsistency.

BGP atoms, on the other hand, also have shortcomings. Two prefixes are put in the same atom if their `AS-PATH` is the same, as seen by *all* our observation points. According to this definition BGP atoms describe policies *across many* ASs, i.e., observation points. We believe that relying on BGP atoms is therefore dangerous for our study, as atoms do not discriminate different interdomain links and parts of the topology. For example, BGP atoms do not capture situations where a large fraction of the policies in the Internet are defined as coarse as per-neighboring AS, while only a small subset of ASs configure policies on a per-prefix level. In this case, BGP atoms are prone to generalize and would suggest that probably all ASs have their policies defined on a per-prefix level.

## 5.4 From routing policies to path choices

In [61], the authors show that modeling policies both as per-prefix filters and as business relationships has severe drawbacks. Relying on business relationships is more scalable as less configuration is required in the model. Unfortunately, inferred relationships are not enough to lead to correct path choices. Per-prefix filtering, on the other hand, allows for models highly consistent with observed path choices, but it is not scalable as its granularity is the finest possible. If we now want to answer the question of what is the right granularity to implement routing policies in an Internet-wide model, we realize that it does not have a definitive answer. Our conclusion so far is that business relationships are, in general, the right way to set routing policies in a model (see [61]). However, predicting path choices requires more details about routing policies: one also has to guess which path to select as best from a set of equivalent paths, all permitted by policies.

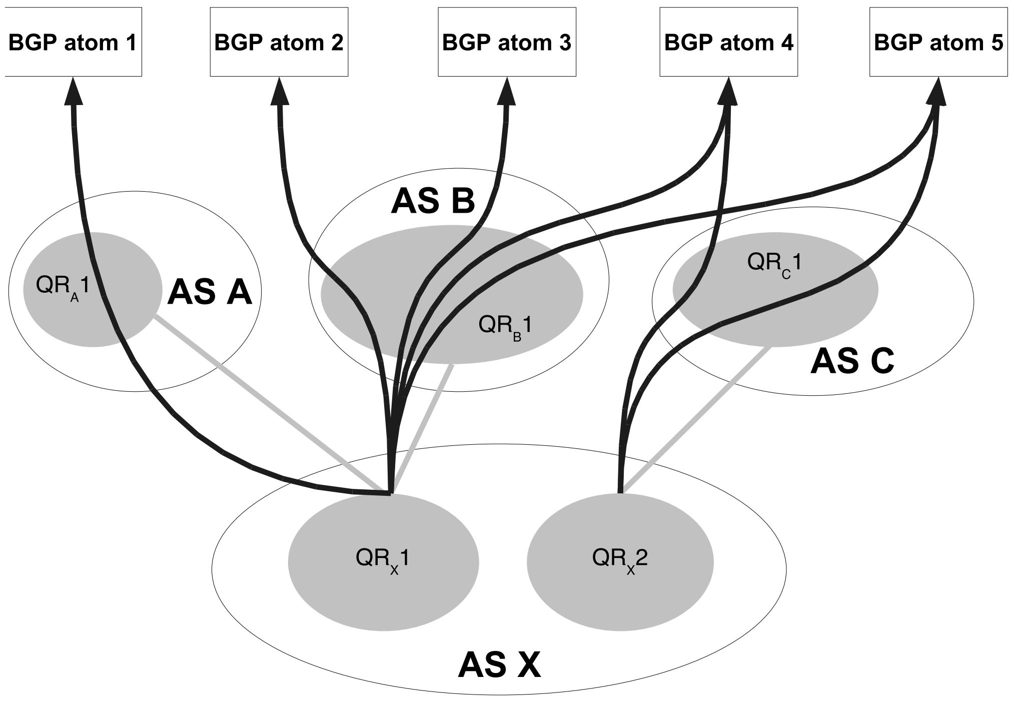


Figure 5.12: Example of path choices and next-hop atoms.

To make the discussion more concrete, we need to introduce some concept that will crystallize this choice of the paths some AS performs. We call it the *next-hop atom*. A *next-hop atom* $NH$ of an AS $X$ is a subset of $X$'s neighbors that $X$ chooses as next-hops for its best routes towards a given set of BGP atoms[4]. All BGP atoms for which we see that an AS uses the same set of neighbors for its best routes belong to the same next-hop atom. The aim of *next-hop atoms* is to capture the distinct sets of neighboring ASs an AS requires to describe its path choices towards groups of prefixes. Note that next-hop atoms do not reveal why some AS prefers some paths to others. Next-hop atoms only describe the choice ASs make, not the reasons for their choice.

Figure 5.12 illustrates an example of the choice of paths made by AS $X$ towards five different BGP atoms. AS $X$ is composed of two quasi-routers, $QR_X1$ and $QR_X2$. It has three neighboring ASs: $A$, $B$ and $C$, each composed of a single quasi-router. The best path, AS $X$ chooses towards BGP atom 1, has as next-hop AS $A$. To reach atoms $2$ and $3$, $X$ uses as its next hop AS $B$, whereas the best paths towards both atom $4$ and $5$ go through AS $B$ and $C$. In this example, AS $X$ requires two quasi-routers because it has to choose two different best paths towards atoms $4$ and $5$.

In the case of the example in Figure 5.12, AS $X$ has three different next-hop atoms: $NH_1$

[4]The definition of next-hop atoms can be trivially extended to next-hop routers if more detailed information about ASs is available.

contains next-hop $A$ towards BGP atom $1$, $NH_2$ contains next-hop $B$ towards BGP atoms $2$ and $3$ (since AS $X$ chooses its best routes towards BGP atoms $2$ and $3$ via AS $B$), and $NH_3$ contains next-hops $B$ and $C$ towards BGP atoms $4$ and $5$ (because AS $X$ chooses its best routes towards BGP atoms $4$ and $5$ via AS $B$ and AS $C$). Among all possible combinations of next-hop ASs, only a subset will actually be used to send traffic towards BGP atoms. In our example, we only need three distinct combinations of neighboring ASs towards the five considered BGP atoms. A next-hop atom captures the coarsest granularity (across prefixes) at which an AS chooses its best paths in distinct ways (among its neighbors).

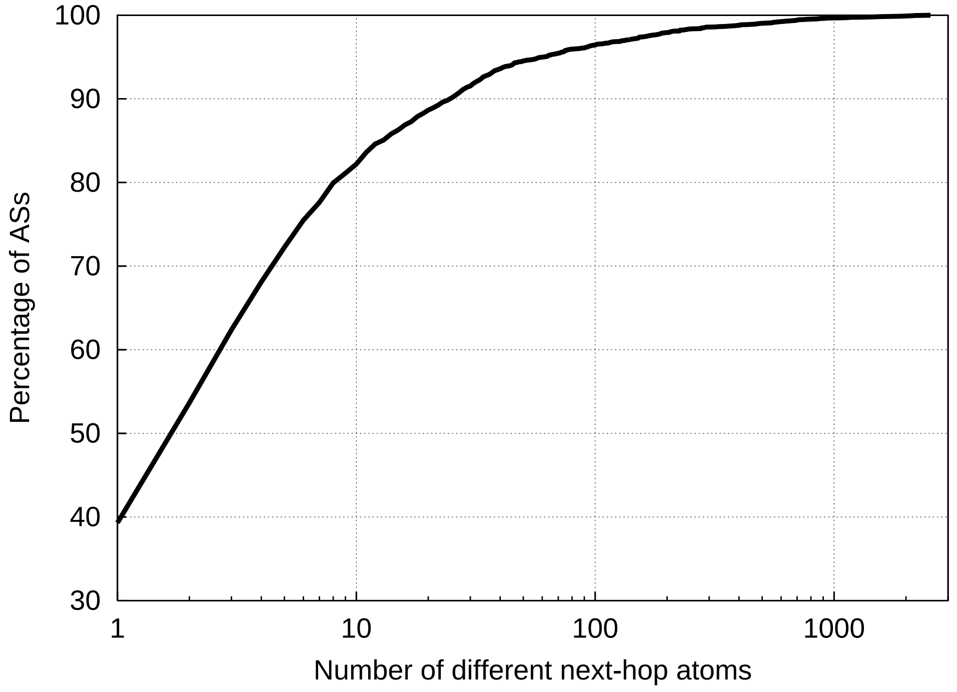


Figure 5.13: Number of next-hop atoms per AS.

The reason to define next-hop atoms in terms of BGP atoms is that BGP atoms define the finest granularity at which sets of prefixes share the same path choices. One might choose to use prefixes instead of BGP atoms.

Now that we have the concept of next-hop atoms to capture the granularity at which ASs select their paths, we can study the observed granularity at which ASs choose their paths. The simplest way an AS can select its best paths is by always using the same set of neighbors for all prefixes. Such an AS would have the same next-hop atom towards all prefixes. Single-homed ASs are in this situation as they have a single neighbor from which to choose their paths. Large transit providers on the other hand are expected to have a large number of different next-hop atoms due to their larger number of neighbors.

Figure 5.13 shows the distribution of the number of next-hop atoms per AS, over the

$3,535$ transit ASs considered in Section 5.1. We observe that about $40\%$ of the $3,535$ ASs have a single next-hop atom. Modeling routing policies for those ASs is trivial: they select, for all prefixes, the same set of neighbors. For the remaining $60\%$ of the transit ASs, there can be between a few next-hop atoms up to hundreds. As already mentioned, one expects that the larger the AS, the more diverse its set of path choices, hence the larger its set of next-hop atoms. Figure 5.14 confirms this belief by giving, for each of the $3,535$ transit ASs, the relationship between the number of neighboring ASs and the number of next-hop atoms. A vast majority of the ASs ($94\%$) fall on the $x = y$ line, i.e., have exactly as many next-hop atoms as they have neighbors. Only some highly connected ASs have far more next-hops atoms than neighbors (up to $13$ times).

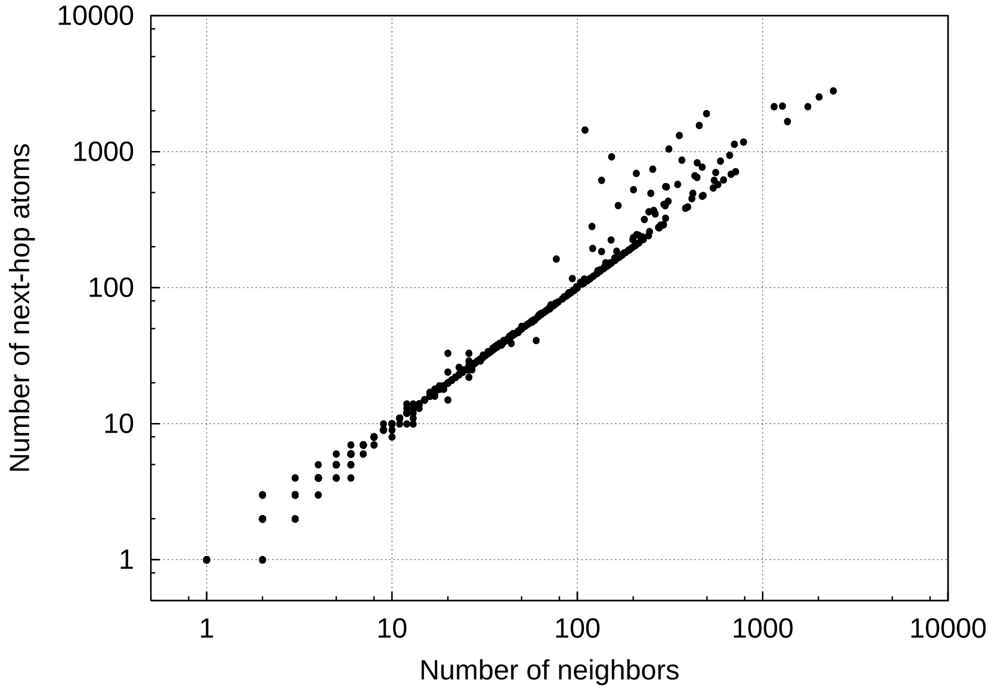


Figure 5.14: Relationship between number of next-hop atoms and neighbors.

One might conclude from Figure 5.14 that since the vast majority of ASs have as many next-hop atoms as neighbors, per-neighbor path choices are the rule. This is only true to some extent. Figure 5.14 does not give any information about how many neighboring ASs any next-hop atom contains. Among all next-hop atoms from our $3,535$ transit ASs, more than $75\%$ contain a single neighboring AS (see Figure 5.15). Only for those next-hop atoms can we configure per-neighbor policies. For the remaining next-hop atoms, preferring a single over all others does not work. In that case, it cannot be only `local-pref` that decides about the choice of the best path, but other rules like MED, IGP cost or other tie-breaking steps of the BGP decision process. One cannot hope to model such detailed information about path choices by routers, especially by relying only on BGP data from a limited set of vantage points.

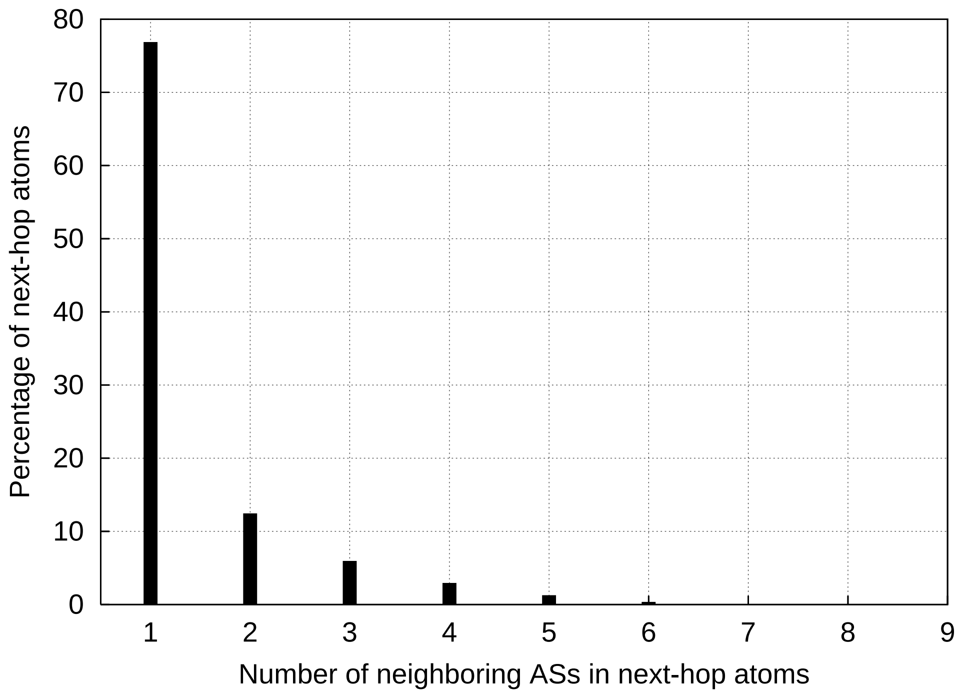


Figure 5.15: Neighboring ASs in next-hop atoms.

Even though per-neighbor path preferences appear quite common in the Internet, a non-negligible fraction of the path choices are made not by routing policies, but by tie-breaking within the BGP decision process.

To further illustrate the complexity of path choices made by ASs, we study $5$ large tier-1 providers in our data. As tier-1 providers have large networks and many neighbors, we would expect them to have complex path choices. Figure 5.16 provides the number of neighboring ASs in the next-hop atoms of $5$ tier-1 providers we selected: UUNET (AS$701$), AT&T (AS$7018$), LEVEL3 (AS$3356$), AOL (AS$1668$), and OPENTRANSIT (AS$5511$). We observe huge differences in the fraction of next-hop atoms that are made of a single neighbor (per-neighbor path choices). UUNET has more than $85\%$ of its next-hop atoms consisting of a single neighbor: its path choices are hence very coarse. AOL on the other hand, has less than $5\%$ of its next-hop atoms consisting of a single neighbor. AOL's next-hop atom granularity reflects its business as content provider. AOL is more likely to choose to leverage its path diversity so as to optimize the performance of the paths. OPENTRANSIT is closer to AOL than the other $3$ tier-1 providers. UUNET and AT&T have a small fraction of next-hop atoms made of several neighboring ASs. LEVEL3 stands in the middle of those $5$ tier-1 providers in the granularity of its path choices.

Modeling how ASs select their path hence depends on the kind of AS being considered. Capturing the full diversity of paths propagated in the Internet, therefore, is not sufficient. We also have to find out what rule of the BGP decision process was used to decide about the path to reach a given prefix. We do not expect this to be an easy task, as it implies inferring very detailed information about AS network engineering.

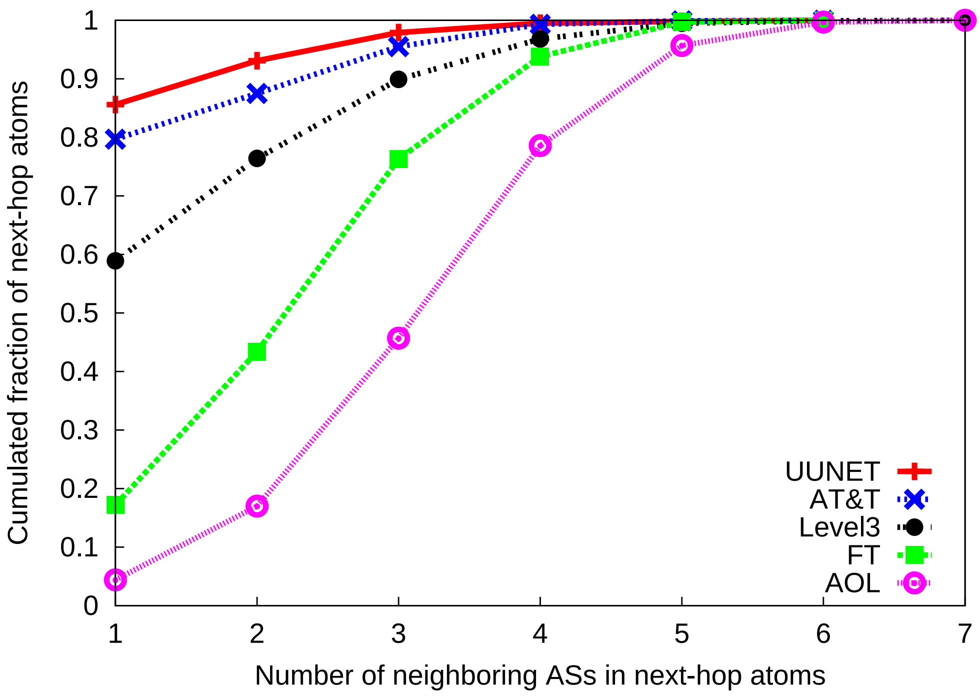


Figure 5.16: Number of neighboring ASs in next-hop atoms for tier-1 providers.

## 5.5 Preference to next-hop AS(s) over time

To compensate for the static property of the dataset used in the previous sections of this chapter, we take BGP dumps and updates from RouteViews for a certain time duration to see how ASs select their next-hops in a dynamic scenario. This section is structured as follows. Section 5.5.1 introduces the BGP dumps and updates dataset we use in this section. Section 5.5.2 presents analysis results on the preference of next-hop AS(s) in the term of the fraction of time that a certain next-hop AS is used to forward traffic.

### 5.5.1 Data over time

For the analysis in this section, we rely on the BGP dumps and updates from RouteViews for the time duration between Feb. 1st 2008 and July 31st 2008. This dataset provides dumps and updates collected from observation points of totally $39$ ASs. Some of these ASs have only a single observation point peering with RouteViews, and others have multiple observation points. With these dumps and updates, we obtain for each observation point, each of its destination prefixes, the routes that are ever used and for which time periods.

Note, some observation points do not show up in the data for some time periods in our considered time duration.

## 5.5.2 Time fraction analysis

We analyze the time fraction that each next-hop AS is used by each direct peer to reach each destination. The selection of a next-hop AS reflects the “preference” [5] to this next-hop AS in the corresponding step in the BGP decision process. And thus, the time fraction that each next-hop AS is used reflects to what extent this next-hop AS is “preferred”.

The *time fraction* is defined as:

$$\frac{T^{DP_i}_{p_j,NH_k}(M)}{T^{DP_i}_{p_j}(M)} \tag{5.1}$$

where $T^{DP_i}_{p_j,NH_k}(M)$ is the amount of time that next-hop AS $NH_k$ is used by direct peer $DP_i$ to reach destination prefix $p_j$ in month $M$, and $T^{DP_i}_{p_j}(M)$ is the total amount of time that prefix $p_j$ shows up in the routing table of direct peer $DP_i$ in month $M$.

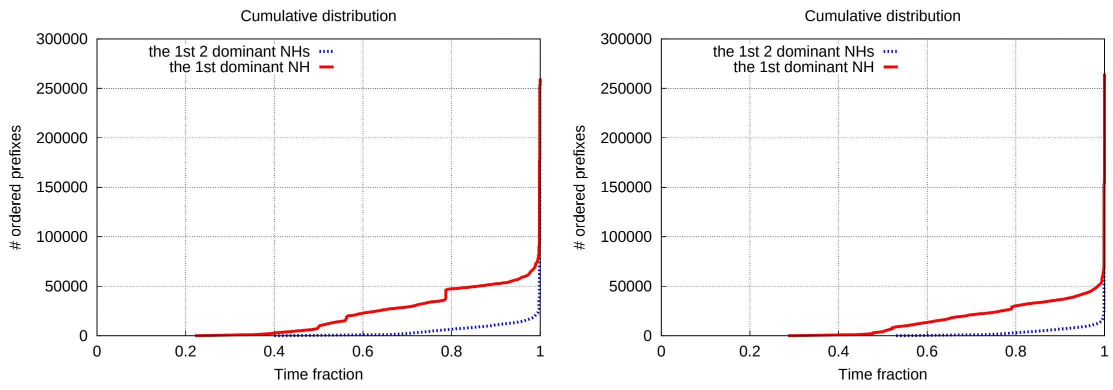


Figure 5.17: Time fractions of direct peer AS286’s next-hop AS(s) for months $2008.02$ (left) and $2008.05$ (right).

We take each month as the time unit of analysis. For data from each month, we do the analysis for each direct peer. For each destination prefix that is ever reachable by the considered direct peer in the considered month, for each next-hop AS that is ever used by

[5]We use “” here over preference, because it might not be the local-pref value which decides the path selection. We just want to differentiate it from local-preference.

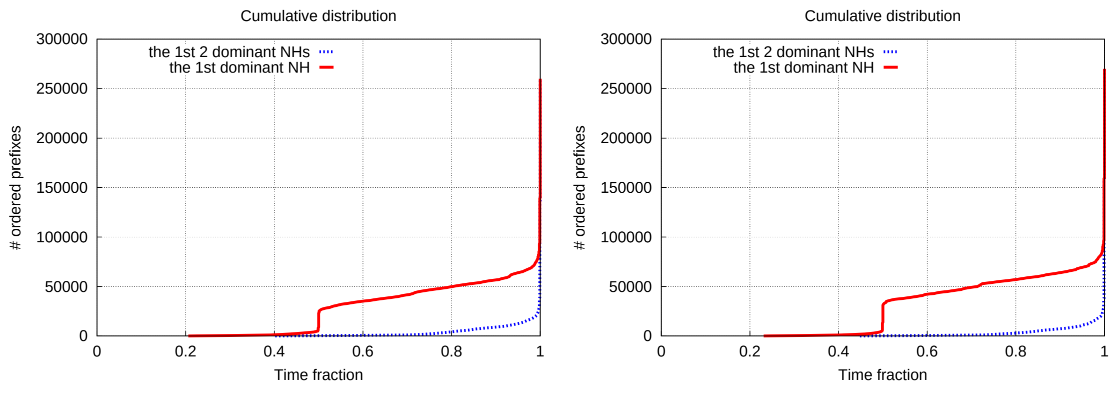


Figure 5.18: Time fractions of direct peer AS2914's next-hop AS(s) for months $2008.02$ (left) and $2008.05$ (right).

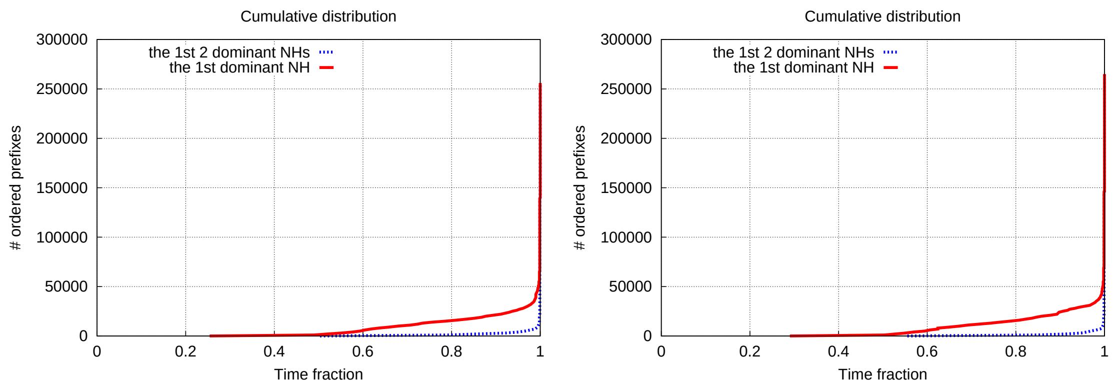


Figure 5.19: Time fractions of direct peer AS5511's next-hop AS(s) for months $2008.02$ (left) and $2008.05$ (right).

this direct peer to reach this destination prefix in this month, we compute the time fraction defined above.

Figures 5.17, 5.18, 5.19 and 5.20 show the analysis results for several ASs that peer with RouteViews. For each direct peer, for each destination prefix, we call the next-hop AS which has the largest time fraction *the* $1$*st dominant next-hop AS*; and the next-hop AS which has the 2nd largest time fraction *the* $2$*nd dominant next-hop AS*. With curve "the $1$st dominant NH", we show the number of prefixes that have their $1$st dominant next-hop ASs' time fractions smaller than $x$. And curve "the $1$st $2$ dominant NHs" shows the number of prefixes that have the sum of their $1$st and $2$nd dominant next-hop ASs' time fractions smaller than $x$.

Note: for direct peers with multiple observation points peering with RouteViews, for each destination prefix, for each next-hop AS, we combine the views of all the observation points in each direct peer.

We can see that all these direct peers give similar curves for next-hop AS(s)' time frac-

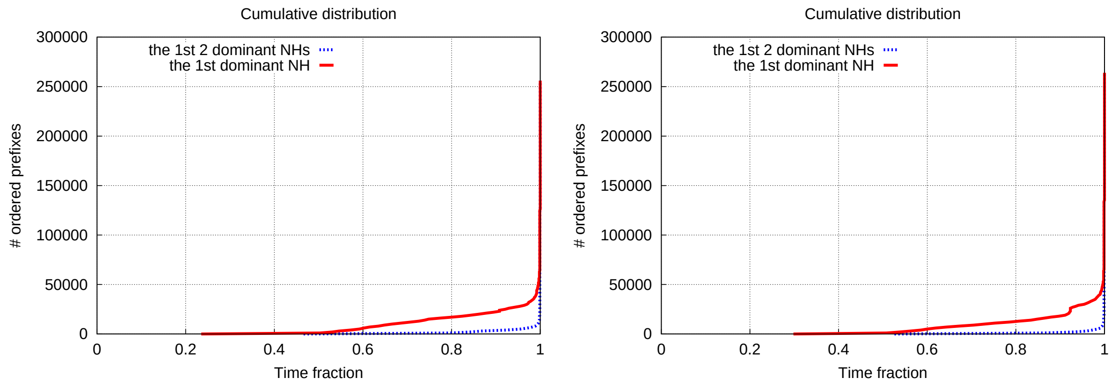


Figure 5.20: Time fractions of direct peer AS7018's next-hop AS(s) for months 2008.02 (left) and 2008.05 (right).

tions:

- for each direct peer, most of the prefixes are reached using the corresponding 1st dominant next-hop AS for most of the time (about 99.9% of the time).
- for the rest of the prefixes, most of them are reached using the corresponding 1st and 2nd dominant next-hop ASs for most of the time (about 99.9% of the time).

We can conclude the following for the direct peers that provide BGP dumps and updates to the dataset.

- The selection of next-hop ASs to reach a certain destination prefix is quite stable. There are a large fraction of prefixes that have their 1st dominant next-hop ASs being used for almost the whole duration (1 month in each figure).
- For most of the prefixes, the 1st 2 dominant next-hop ASs already contribute to nearly 100% of the time. It indicates that for most of the prefixes, when the 1st dominant next-hop AS is not usable (reasons why it is not usable are unknown), the 2nd dominant next-hop AS will be used and the 3rd dominant next-hop AS does not contribute to a big fraction of time.

## 5.6 Related work

Inference of business relationships between ASs [10, 38, 89] has been the most widely studied dimension of routing policies. Routing policies are typically partitioned into a few classes that capture the most common practices in use today [18]. Unfortunately, it is also known that the reality of routing policies [103] and peering relationships is far more complex than those few typical classes [18, 21]. The current approaches for business relationships inference rely on a top-down approach. They first define a set of policies and then try to match those policies with their observations of the system. Yet, policies as used by ISPs have to realize high-level goals [18]. Assuming any kind of consistency of such policies across ASs is questionable, especially as in practice, policies are often configured on a per-router, per-peering, or per-prefix basis [18]. Observed BGP routes do not have to make those high-level policies visible.

Our work is similar to [58, 64] in allowing the propagation of multiple paths across ASs. The authors in [58] aimed at predicting AS paths between any pair of ASs without direct access to the concerned end-points and relied on a new inference of business relationships, as well as other information to predict the AS paths used between any pair of ASs. [64] showed that to reproduce the diversity of the BGP paths observed from multiple vantage points, it is necessary to allow different routing entities inside each AS to store and propagate the routing diversity known to ASs. Another insight of this paper is that agnosticism about policies in the Internet helps to build a model which is completely consistent with observed BGP data and which has good predictive capabilities. The authors used per-prefix filtering policies to force their model to select the paths observed by BGP.

## 5.7 Conclusion

In this chapter, we built an AS-level topology model with a minimal connectivity while still enabling the propagation of all routes observed in BGP data. This AS-level topology model takes into account the internal structure of ASs, and thus serves better as a topology model for interdomain routing than those using single router per AS.

We searched for an appropriate granularity for modeling policies in the Internet. We analyzed the known bounds for policies studied in the literature: BGP atoms and business relationships. BGP atoms do not discriminate different interdomain links and parts of the topology. For example, BGP atoms do not capture situations where a large fraction of the policies in the Internet are defined as coarse as per-neighboring AS, while only a small subset of ASs configure policies on a per-prefix level. In this case, BGP atoms are prone to generalize and would suggest that probably all ASs have their policies defined on a per-prefix level. The *valley-free* properties used for business relationships inference were validated in [61]. However, business relationships do not help to decide which paths among the candidates should be chosen by each AS: after enforcing policies in the model in the form of business relationships, much choice is left as to which route to choose as the best among the candidates. Business relationships do not contain enough information about the path choices made by ASs.

To capture the way individual ASs choose their best paths, we introduced a new abstraction: next-hop atoms. Next-hop atoms capture the different sets of neighboring ASs an AS uses for its best routes. We showed that a large fraction of next-hop atoms correspond to per-neighbor path choices. A non-negligible fraction of path choices however do not correspond to simple per-neighbor preferences, but hot-potato routing and tie-breaking within the BGP decision process, which are very detailed aspects of Internet routing.

The analysis done in the dynamic scenario, although limited by the observation points existing in RouteViews, shows that the selection of next-hop ASs to reach a certain destination prefix is quite stable. There are a large fraction of prefixes that have their 1st dominant next-hop ASs being used for more than $99.9\%$ of the time. For most of the prefixes, the 1st 2 dominant next-hop ASs already contribute to nearly $100\%$ of the time. It indicates that for most of the prefixes, when the 1st dominant next-hop AS is not usable, the 2nd dominant next-hop AS will be used. While for most of the prefixes, the 3rd dominant next-hop AS does not contribute or only contributes to a small fraction of time. It suggests that modeling the dynamic aspects of Internet routing is not impossible.

The work carried out in this chapter provides another step towards a model that may allow prediction of AS paths under "what-if" scenarios. In future work we will validate the policies we derived by testing their predictive capabilities and also by comparing them to actual policies configured by ASs as in [28].

# Chapter 6

# Pushing QoS across interdomain boundaries

Fu, B., C. Pelsser, and S. Uhlig, 2008, "*Pushing QoS across interdomain boundaries*", Chapter 6 of ≪End-to-End Quality of Service Engineering in Next Generation Heterogenous Networks≫, jointly published by ISTE and Wiley & Sons, November.

Today's Internet essentially provides best-effort service. More stringent services like VPN have recently been deployed [79], but crossing AS boundaries proves to be difficult. An Autonomous System is a network under a single administrative authority. Typically, an autonomous system will correspond to the network of an ISP, an enterprise or an academic network. The ability to provide QoS guarantees enables Internet Service Providers to propose new services and consequently leads to new sources of revenue for them. MPLS makes it possible to set up Label Switched Paths LSPs with QoS guarantees in a single AS. The Traffic Engineering (TE) extensions to the ISIS/OSPF routing protocol(s) [48, 86] enable the intradomain routing protocols to distribute the traffic engineering properties of the links. Once the topology and traffic engineering information is known, paths which meet the QoS guarantees can be easily computed with the Constrained Shortest Paths First (CSPF) algorithm or more elaborate algorithms like SAMCRA [98] and DAMOTE [13]. QoS may be provided to flows by establishing MPLS LSPs with RSVP-TE [29] along paths with certain properties. The QoS routing algorithm ensures that the selected path meets the QoS requirements. RSVP-TE makes possible to reserve resources to guarantee the QoS to be experienced along time. When the LSPs need to traverse multiple ASs administered by different ISPs, new path computation techniques together with protocol extensions for establishing LSPs need to be introduced.

If several solutions to the QoS problem exist, ISPs would prefer to choose the one that is as close as possible to their current network design. New functionalities, which can be realized incrementally based on the current system, are more likely to be deployed, especially in the short term. In this chapter, we build up on the existing features to show how we can evolve towards a QoS-capable Internet.

In Section 6.1 of this chapter, we discuss the implications of the current Internet routing system on the establishment of LSPs with QoS constraints across multiple domains, and the requirements for establishing interdomain LSPs. The solutions existing in the literature for signaling the establishment of interdomain LSPs are introduced in Section 6.2. We review the work done in the Path Computation Element (PCE) Working Group (WG) of the Internet Engineering Task Force (IETF) and discuss the applicability of the proposed path computation architecture and techniques in Section 6.3. In Section 6.4, we discuss, based on the existing functional components, the possible solutions we envision for the problem of interdomain LSP establishment and the open issues left to be solved to make interdomain QoS a reality. Finally, we conclude and give some perspectives in Section 6.5.

## 6.1 Background

In this section, we describe the working of the interdomain routing system. We discuss the consequences of the path selection made by the current interdomain routing system on the visibility of the paths, and its implications for establishing inter-AS LSPs with end-to-end QoS guarantees. The requirements for establishing inter-AS LSPs are also detailed.

### 6.1.1 The Internet as a distributed system

An utopian situation would be for a certain router to know about the topology information of all ASs, to rely on CSPF or another QoS routing algorithm to compute the path based on the complete topological information, and to establish the LSP according to the path found by the QoS routing algorithm. It is not applicable in the general interdomain framework, as ASs hide to competitor ASs their internal structure for security and business reasons [87, 109]; and at the same time, routing using a QoS routing algorithm on the complete topology and QoS information would not scale. As a consequence, a single node cannot compute an end-to-end path composed of individual routers for an LSP crossing multiple ASs. Instead, the computation of a path has to be distributed among multiple path computation elements nodes[1], where each node computes a segment of the path based on its knowledge of the local AS topology, the interdomain reachability information provided by BGP and other information needed specially by the segment computation (like constraints on the IP-level path). The establishment of LSPs also needs to be distributed among the crossed ASs. Each AS is in charge of the establishment of LSP segments within its own network.

[1]An abstract node identifies a set of Label Switching Routers (LSR).

### 6.1.2 Impact of interdomain routing on path diversity

As explained in Chapter 4, each BGP router selects **only one** best route from all learned ones, and advertises **only this** best route to its neighbors. Meanwhile, BGP route propagation needs to follow the valley-free properties. Both aspects influence the AS paths learned by a BGP router.

Other routing policies also introduce constraints on the paths known by a router. For instance, ASs may announce only subsets of their prefixes to their neighbors in order to control the amount of incoming traffic they receive from these neighbors [103].

The AS paths received by a BGP router from its neighbors may be only a small subset of all the AS paths its neighbors know, because each BGP router advertises only its best paths. Moreover, because of the redundant connectivity in the intradomain and/or the interdomain topology, the IP-level path to which an AS path corresponds, is one out of possibly many IP-level paths consistent with this AS-level path.

Thus, the AS paths seen by a BGP router (in the RIB-Ins) towards a prefix, represent only a fraction of all the usable connectivity. By usable connectivity, we mean the IP-level paths that are consistent with routing policies. In the rest of this Section 6.1.2, we show the routing diversity obtained from simulations on a topology with routing policies extracted from observed data.

#### Observed Internet routing diversity

In this section, we present a lower bound on the available routing diversity in the Internet. We use the interdomain connectivity provided in Chapter 5. It is extracted using BGP data from more than $1,300$ BGP observation points, including those provided by RIPE NCC [1], RouteViews [81], GEANT [45], and Abilene [2]. This AS-level model allows each AS to be composed of multiple entities, in order to capture the intradomain routing diversity. The AS-level connectivity is minimal while enabling the propagation of all AS paths observed in the BGP data. This minimal connectivity might lead to an underestimation of the actual number of AS paths learned by an AS. The results shown in this section thus only provide a lower bound on the path diversity in the Internet.

The BGP path selections at the routers are simulated on the above connectivity. The only policies configured are those preventing valley-free paths to propagate. Business relationships are inferred using the CSP algorithm [42] on the data described in Section 5.1. Because it has been observed in [61] that trying to implement route preferences consistent with the inferred business relationships leads to path choices in the simulations that are often inconsistent with the paths observed in the BGP data, we only prevent non-valley-free paths from propagating and let the shortest AS paths propagate.

We rely on C-BGP [77] to compute the outcome of the BGP decision process and the set of learned routes at every router of the AS-level connectivity. We show the results for a single randomly chosen destination AS, even though we tried different destination ASs and

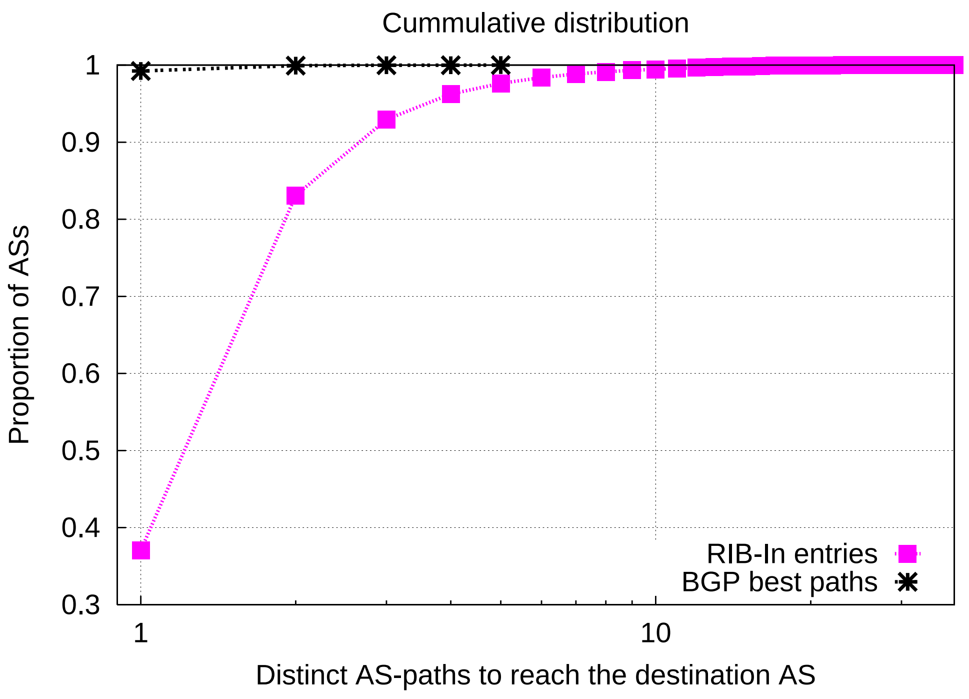


Figure 6.1: Number of AS paths learned/selected to reach a certain destination AS.

got similar distributions for all of them. Simulation results for the selected destination AS are shown in Figure 6.1. On the x-axis of Figure 6.1 we show the number of distinct AS paths that are observed in all routers of an AS towards a given destination AS. On the y-axis, we show the cumulative fraction of ASs that see x or less distinct AS paths towards the destination AS. There are two curves on Figure 6.1: one for the best paths selected by the BGP routers in the simulation, and another for the routes in RIB-Ins of the BGP routers of the simulation. Observations from BGP data only provide information about best routes selected by BGP routers, not RIB-Ins. The upper curve of Figure 6.1 shows that more than $99\%$ of the ASs only have a single AS path to reach the given AS. If the RIB-Ins of the routers are considered, on the other hand, more path diversity is available in large ASs. For the considered destination AS, about $36\%$ of all the ASs learn only $1$ AS path. These ASs are probably stub ASs, that have a single upstream provider, hence they cannot learn more than a single AS path towards any destination. $64\%$ of all the ASs learn multiple AS paths towards the destination AS, with some of them knowing up to $38$ distinct AS paths. This is a lower bound on the actual number of AS paths that are known by ASs compared to reality. Thus, we expect that even more diversity is hidden in the BGP routers in practice.

The previous results indicate that best path selection, and the fact that the BGP routers only advertise their best routes, reduces a lot the path diversity visible in the routers. The actual path diversity that might be usable in practice (RIB-Ins) is highly underestimated because it is not completely visible from BGP data. The BGP data only gives a limited view of the Internet. Using the BGP data to determine the RIB-Ins is bound to give only a lower bound indication. The poor visibility of path diversity does not mean that good QoS paths cannot be found if the actual path diversity present in current BGP routers can be used.

### 6.1.3 Inter-AS LSP requirements

As shown in Chapter 4, the selection of the best BGP route does not depend on the quality of the path in terms of delay, bandwidth, *etc*. This is also verified by the literature [14, 43, 82, 110]. Thus, the best BGP route may not be suitable for a particular type of traffic with given QoS requirements. To provide specific QoS guarantees, it might be necessary to use different paths than those chosen by BGP. One solution for this is to use MPLS LSPs with RSVP-TE. In this section, we discuss the requirements for inter-AS LSPs expressed by ISPs in [109].

- Among these requirements is the desire of ISPs to keep internal AS resources and the set of hops followed by the TE-LSP confidential. This confidentiality requirement implies the constraint of only partly specifying the hops that the TE-LSP must traverse since global topology information is not available. Moreover, it must be possible to perform path optimization inside each transited AS, where the required information is available. In addition, end-to-end optimization of inter-AS LSPs is also required by ISPs.

- A second requirement, the protection requirement, concerns the restoration capabilities of inter-AS LSPs. The proposed solution has to be able to provide rapid local protection against link, Shared Risk Link Group (SRLG) and node failures. An SRLG is a group of links that may fail at the same time. It is a set of links that share a common physical resource such as an Ethernet switch, a fiber, an optical cross-connect, *etc*. Additionally, the proposed solution should support the establishment of multiple link/SRLG/node diversely routed inter-AS traffic engineering LSPs between a pair of LSRs.

- A last requirement, the scalability requirement, is that the proposed solution should be scalable in terms of the amount of IGP flooding, the additional information carried by BGP, the amount of RSVP-TE signaling messages exchanged and state to retain.

## 6.2 RSVP-TE extensions to support interdomain LSPs

RSVP-TE is detailed in [8]. In this section, we discuss extensions to enable the establishment of traffic engineered interdomain LSPs towards a prefix destination. The local or global protection of these interdomain LSPs is discussed afterwards, and are followed by the discussion on the scalability of this technique.

We present the extensions to RSVP-TE proposed in [72] that fulfill both the confidentiality and the protection requirements concurrently while trying to keep the solution scalable. This solution also tries to only impact the head-end LSR, the intermediate AS Border Routers (ASBRs) on the path of the inter-AS LSP and the tail-end LSR of the LSP therefore allowing a smooth migration towards the support of inter-AS LSPs. It does not impact the current BGP and MPLS traffic engineering techniques. Moreover, it does not require additional IGP flooding.

### 6.2.1 Explicit routing of an LSP

The Explicit Route Object (ERO) is well suited for the establishment of inter-AS LSPs in that it enables the head-end of the LSP to partially specify the path to be followed by the LSP. Following nodes crossed by the `Path` message are able to complete this object as the `Path` message goes along. More precisely, the head-end LSR of an inter-AS LSP is only able to fill the ERO with nodes that belong to the same AS and eventually with the list of ASs that will be crossed by the `Path` message. At the entrance of each AS, the ASBR computes the path of the LSP towards the downstream AS and completes the ERO accordingly. This process is illustrated in Figure 6.2. We see that $R0$ computes the path towards $AS1$ and sets the ERO accordingly. Inside $AS1$, $R3$ completes the ERO towards the next AS, $AS2$ and so on. These paths are computed based on the LSP's destination address. The ERO specifies only a set of hops on the path of the inter-AS LSP and it leaves to each crossed AS the responsibility of the local path optimization according to a set of constraints also carried inside the `Path` message of the LSP. This fulfills the local path optimization requirement from the second paragraph of Section 6.1.3. A mechanism for the reoptimization of loosely routed LSPs signaled with RSVP-TE is defined in [100].

The ERO object may be constructed at the head-end LSR either based on a manual configuration that specifies the ASs and/or the ASBRs to be crossed by the LSP, based on the BGP routing table, or based on the path computation result of a path computation entity such as the Path Computation Element (PCE) introduced in Section 6.3.

The interdomain path selection could be performed by relying on QoS information distributed by extensions to BGP. Such QoS enabled advertisements were proposed in [106] and are still under research. Later in this chapter, we look more deeply into interdomain path selection techniques.

### 6.2.2 RRO aggregation and the Path Key

The Record Route Object (RRO) enables to obtain the path followed by an LSP. Thus, it is useful to detect loops inside the LSP's path, to pin the LSP onto its path and to compute LSPs disjoint from this LSP for global or local protection.

Note that recording the path of an inter-AS LSP may be in contradiction to the ISPs' desire to hide the internal topology of ASs. Therefore, it is proposed in [72] to modify the

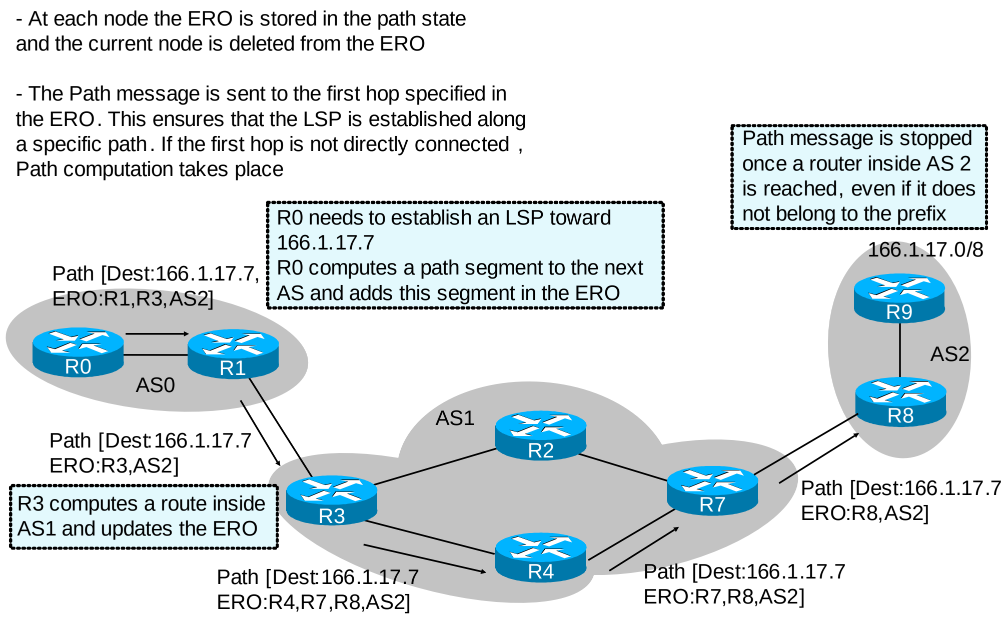

- At each node the ERO is stored in the path state and the current node is deleted from the ERO
- The Path message is sent to the first hop specified in the ERO. This ensures that the LSP is established along a specific path. If the first hop is not directly connected , Path computation takes place
Path message is stopped once a router inside AS 2 is reached, even if it does not belong to the prefix
R0 needs to establish an LSP toward 166.1.17.7
R0 computes a path segment to the next AS and adds this segment in the ERO
Path [Dest:166.1.17.7, ERO:R1,R3,AS2]
166.1.17.0/8
R9
AS2
R0
R1
AS0
R8
AS1
R2
Path [Dest:166.1.17.7 ERO:R3,AS2]
R7
Path [Dest:166.1.17.7 ERO:R8,AS2]
R3 computes a route inside AS1 and updates the ERO
R3
R4
Path [Dest:166.1.17.7 ERO:R4,R7,R8,AS2]
Path [Dest:166.1.17.7 ERO:R7,R8,AS2]

Figure 6.2: Establishment of an inter-AS LSP.

processing of this object at the ASBRs so as to withhold from neighboring ASs the complete path followed by the LSP inside the current AS. This process is called “RRO aggregation”.

The aggregation of the RRO consists in marking the sub-object added by the ingress ASBR inside the AS. Thus, at the last router of the AS, i.e. the egress ASBR, the list of nodes in the AS are removed from the RRO. These sub-objects are replaced by the address of the ingress ASBR, the AS number and the address of the egress ASBR in order to keep enough information to perform loop detection, disjoint path computation and route pinning of the inter-AS LSP. We use the example topology in Figure 6.2 to illustrate the aggregation of the RRO. In $AS1$, the ingress ASBR $R3$ adds its address inside the RRO and marks it. The following LSRs ($R4$ and $R7$) add their addresses inside the RRO. The egress ASBR, $R7$ in Figure 6.2, removes all addresses starting from the marked sub-object, representing the address of $R3$. It replaces these sub-objects by the address of the ingress ASBR ($R3$), its AS number ($AS1$) and its own address ($R7$).

[16] proposes using path keys to fulfill the confidentiality requirement of ISPs. The Path Key contains the identifier of the node that knows the list of nodes composing the confidential path segment. Path Key Sub-objects (PKS) can be stored inside the ERO of the RSVP Path messages. Such sub-object must follow the node responsible for expanding the path key, that is the first node of the confidential path segment. This node sends the path key to the node with identifier contained in the PKS for expansion of the path key into a sequence of nodes.

### 6.2.3 Protection of inter-AS LSPs

Restoration capabilities need to be provided to inter-AS LSPs against link, node and SRLG failures. In this section, we introduce the local protection of inter-AS LSPs [72]. The possibility to establish completely link or node disjoint LSPs can be useful to balance traffic on these disjoint LSPs, or provide reliability to failures. We give ways to address this problem in Section 6.2.4.

#### Local protection of inter-AS LSPs

Techniques to protect AS core nodes and links joining these nodes are described in [69]. The protection of links connecting distinct ASs, called "interdomain" links, is discussed in [71]. These techniques can be combined with the ones described in [69] to protect inter-AS LSPs all the way along their path.

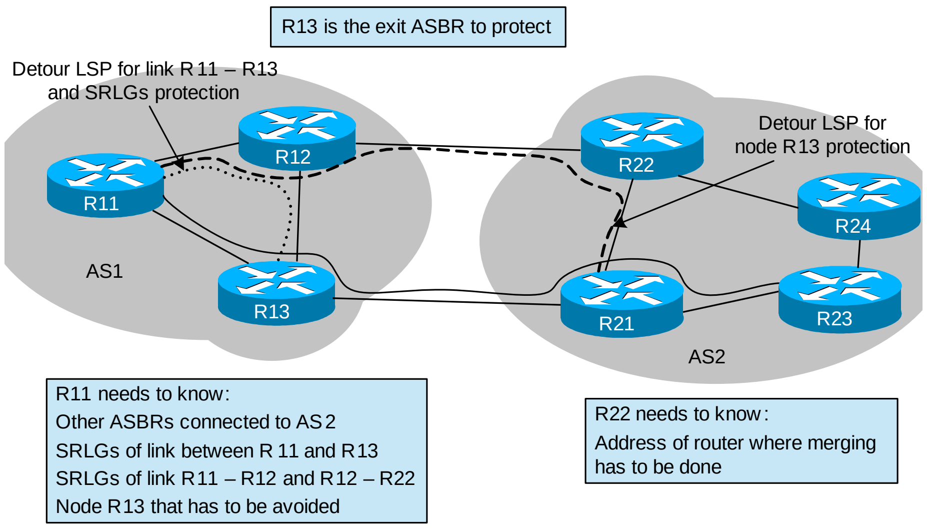

R13 is the exit ASBR to protect
Detour LSP for link R11 – R13 and SRLGs protection
Detour LSP for node R13 protection
R11
R12
R13
R21
R22
R23
R24
AS1
AS2
R11 needs to know:
Other ASBRs connected to AS2
SRLGs of link between R11 and R13
SRLGs of link R11 – R12 and R12 – R22
Node R13 that has to be avoided
R22 needs to know:
Address of router where merging has to be done

Figure 6.3: Local protection against egress ASBR and against the SRLGs of the link preceding this ASBR.

Here, we use an example (in Figure 6.3) to illustrate how to locally protect inter-AS LSPs against the failure of the egress ASBR ($R13$) and of the SRLGs of the link preceding this ASBR ($R11$ - $R13$). This problem is best solved by using two detour LSPs at the node $R11$ on the path of the working LSP, preceding the egress ASBR. A detour LSP protects against the SRLGs of the intra-AS link $R11$ - $R13$. A second detour LSP protects against the egress ASBR failure. The detour protecting against the SRLGs has to merge in the same AS as the link to protect, i.e. it has to merge with the working LSP at the egress ASBR $R13$. This is because other ASs do not know this intra-AS link, nor its SRLGs. The detour protecting against the egress ASBR needs to exclude $R13$ and merge with the working LSP in the next AS $AS2$.

For local protection against other resources such as the ingress ASBR, we refer to [27].

An LSP that is used to protect a set of LSPs crossing common resources, is called a Bypass Tunnel. The establishment of Bypass Tunnels for the protection of inter-AS LSPs is analogous to the establishment of Detour LSPs, exposed previously.

### 6.2.4 End-to-end disjoint LSPs

Sprintson et al. [88] propose a distributed routing algorithm for finding two disjoint inter-AS QoS paths. They rely on a link-state aware topology abstracted from the multi-domain network. For an analysis of different schemes to establish end-to-end disjoint LSPs, we refer the reader to [90].

### 6.2.5 Scalability

As can be seen from Section 6.2.1 and 6.2.2, establishing an interdomain LSP following a loosely specified path goes straight forward as the `Path` message goes across the involved ASs. The route projects contain the necessary information to guide the LSP establishment, and their contents can not be cut. The time duration needed for setting up such an interdomain LSP sums up the time durations used by each AS to process the route objects, to compute the path inside its own network and to set up the LSP segment following this path. Since the path computation and LSP segment establishment inside each AS should scale, the establishment of an interdomain LSP with a given loosely specified path also scales.

## 6.3 State-of-the-art in interdomain PCE

As shown in Section 6.2, with the RSVP-TE extensions, interdomain LSPs can be established if some guide information (e.g., intermediate ASBRs on the path) is provided. In this section, we summarize the work done in the PCE WG of the IETF on the problem of finding an interdomain path that respects certain QoS constraints. We focus on the proposed path computation techniques which enable to determine the guiding information. And then, we discuss the limitations and/or applicability of these path computation techniques.

### 6.3.1 PCE-based architecture

The PCE WG of the IETF is working on an architecture [30] for the computation of paths to support MPLS traffic engineering LSPs. This architecture aims to be applied within a single domain or within a small group of domains (where a domain is a layer, an IGP area or an Autonomous System with limited visibility).

A PCE is an entity that can collect QoS, topology and reachability information. It can carry out the path computation on behalf of a set of routers (Path Computation Clients(PCCs)). PCEs, in the same AS or in different ASs, can be configured to communicate and cooperate [7] with each other in order to find feasible end-to-end paths. A Traffic Engineering Database (TED) is used to store the information the PCE is interested in. It is used by the PCE to carry out path computations.

The generic requirements to allow the communications within and between PCEs were covered in [7].

An additional mechanism, called PCE discovery, is required in order for the PCCs to learn the list of PCEs that are available in their domain and in neighboring domains. The requirements for such protocol are expressed in [53, 67].

### 6.3.2 Path computation methods

As can be seen from the name, the PCE-based architecture focuses on how to compute paths. In this section, we introduce the proposed path computation methods. First we explain the function of the TED which is used by PCEs to store traffic engineering related information. Then, we present the path computation techniques. These techniques work under the assumption that the AS path is known a priori.

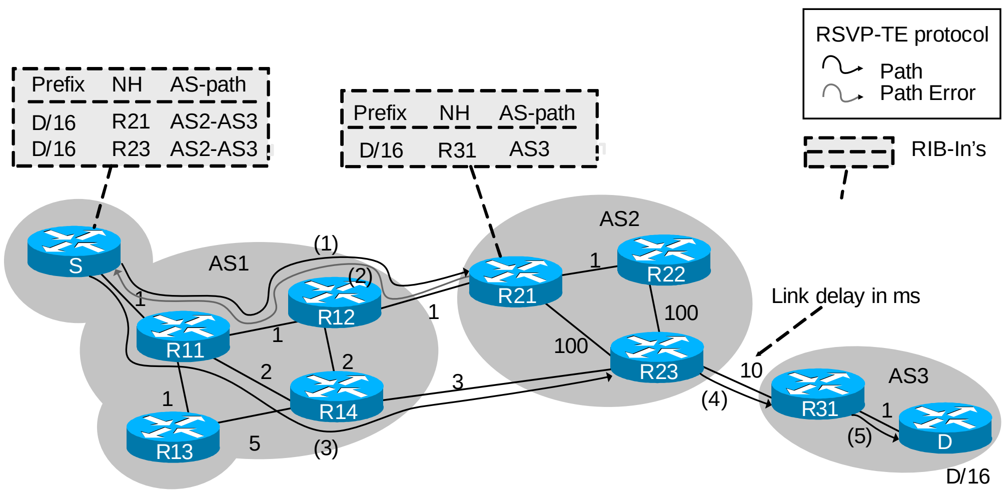
#### TED

The PCE computes path segments respecting given QoS and diversity constraints based on a TED. The content of the TED for interdomain traffic engineering has been discussed at the IETF [31]. It depends on the domain of the PCE. The TED contains at least the topology of the domain and the traffic engineering attributes of the links belonging to the domain. In addition, it may contain the traffic engineering attributes of the links at the border of the domain, for example the inter-AS links. This information is distributed by the traffic engineering extensions to the Interior Gateway Protocols (IGP) [22, 48, 86]. Moreover, the TED must also contain reachability information for destinations outside the domain of the PCE. This information is currently distributed by BGP for destinations outside the AS.

#### Per-domain path computation

This technique relies on the simultaneous computation and establishment of interdomain MPLS LSPs. It makes use of RSVP-TE to establish interdomain MPLS LSPs and of its ability to crankback [29]. That is the capability (1) to stop the establishment of an LSP at a node when it cannot compute a path that respects the constraints of the LSP and (2) to establish the LSP along a different path.

As we mentioned in Section 6.2.1, inside RSVP-TE, it is possible to indicate inside the ERO the path or a portion of the path to be followed by an LSP . The per-domain path



| Prefix | NH | AS-path |
|---|---|---|
| D/16 | R21 | AS2-AS3 |
| D/16 | R23 | AS2-AS3 |

| Prefix | NH | AS-path |
|---|---|---|
| D/16 | R31 | AS3 |

(1) S tries the NH R21 and sends Path to R 21.

(2) R21 can not find a segment that respects the QoS constraints. It sends an Error message to R12. The Error message is propagated upstream to S . S puts NH R21 in the avoiding list .

(3) S tries NH R23 and sends Path to R 23.

(4) R23 tries NH R31 and sends request to R31.

(5) R31 computes path toward D .

Figure 6.4: Per-domain path computation.

computation technique, described in [99], relies on this object. It consists in completing at the ingress router of a domain, the ingress ASBR, the path computation up to the BGP Next-Hop (NH), i.e. last reachable hop towards the destination. This node is either the first hop inside the downstream domain or the last hop inside the current domain. The computed path segment is then stored inside the ERO of the RSVP-TE `Path` message. This message is forwarded along the path specified inside the ERO and requests the establishment of the LSP along the path.

Either a dedicated PCE or the ingress routers in an AS should be responsible for the computation of the path segments. In Figure 6.4, the ingress ASBRs compute the paths. Upon reception of an RSVP `Path` message requesting the establishment of an LSP, an ASBR computes the paths. The ASBR tries usable BGP next-hops to reach the destination and consistent with the given AS path. To determine the usable BGP next-hops is still an open issue. Here, we consider as the usable BGP next-hops those known from the BGP propagation. The ASBR stores the list of BGP next-hops that have already been tried for an LSP and that lead to an infeasible path with regard to the constraints. When the ASBR is not able to complete the path with a segment respecting the QoS properties of the LSP, "crankback" is performed [29]. That is, the ASBR generates an RSVP `Path Error` message and sends it upstream. The upstream ASBR computes a new segment avoiding the BGP next-hops that have already been tried.

Figure 6.4 illustrates the per-domain path computation technique with path computation that takes place at the ingress ASBRs. In this example, an LSP with delay constraint of $100$ ms has to be established from $S$ to $D$ using a given AS path $AS2$ - $AS3$. The source of the LSP $S$ first tries to use BGP next-hop $R21$. It computes a path segment towards $R21$ respecting the constraints based on its knowledge of the internal topology of $AS1$. $S$ generates an RSVP `Path` message with ERO object that contains the computed path segment. Then the `Path` message is sent along the segment. This leads to the establishment of the LSP along the path segment.

At the ingress ASBR inside $AS2$, $R21$, the process described in the previous paragraph is repeated. That is, $R21$ computes the path segment towards BGP next-hop $R31$ in order to reach the tail-end of the LSP. However, $R21$ is not able to provide a path segment that respects the constraints. Consequently, crankback occurs at $R21$.

$R21$ sends a `Path Error` message upstream. When the `Path Error` message arrives at $S$, $S$ tries to compute the path segment avoiding using BGP next-hop $R21$ which leads to an infeasible path. $S$ computes a path segment that ends at BGP next-hop $R23$. It sends a `Path` message along the path segment towards $R23$. $R23$, in the downstream AS, $AS2$, carries out the computation of a path segment starting at $R23$ and ending at the entrance $R31$ inside the downstream AS $AS3$. This path segment is inserted inside the ERO of the `Path` message and the establishment of the LSP continues until the LSP's tail-end is reached.

This technique may or may not rely on PCEs. Moreover, the path computation and the LSP establishment take place at the same time. The computation ends once a path respecting the constraints is found even if it is not the shortest path. With the per-domain path computation technique, if PCEs are used in the computation, they do not communicate among themselves. Thus, discovery of PCEs in neighboring domains is not required.

### Backward recursive path computation

The Backward Recursive PCE-based Computation (BRPC) technique, is described in [102]. It has been designed to find the shortest path for a constrained inter-AS LSP request. It makes the assumption that the list of domains to be crossed by the LSP is known prior to the computation. Thus, the computed path is the best path that can be obtained along this interdomain path.

A PCE, which uses the BRPC technique to compute the path, communicates with other PCEs in order to request the computation of path segments contained in regions for which it does not possess enough topological information. Cooperative PCEs can communicate with each other using the protocols specified in [101].

In this technique, the LSP's head-end sends a PCReq message specifying the constraints for the LSP to the PCE of its domain. Then, a PCReq message is sent from the PCE of one domain to the PCE of the downstream domain. Upon reception from the PCEs in the downstream domain, of multiple path segments starting at the entrances of the downstream domain and ending at the LSP's tail-end together with their QoS properties, a PCE is capable of computing the best segments starting at the entrances of its domain and ending at the tail-end of the LSP, with regard to the constraints. These segments are sent to the upstream PCE inside a PCRep message.

Figure 6.5 illustrates the computation of inter-AS constrained paths by means of BRPC. The LSP to establish is subject to a maximum delay constraint of $100$ ms. The head-end of this LSP is router $S$ in $AS1$. The tail-end of the LSP, node $D$, belongs to $AS3$. The longest matching prefix advertised for $D$ is $D/16$. First, the AS path is determined. The AS path which needs to be followed by the LSP is $AS2$ - $AS3$. The central part of the figure shows the physical topology of the ASs and their interconnections. In the top part of the figure, we see the PCEs of each ASs, labels for the messages exchanged between PCEs and the BGP routes known by the PCEs. The content of the messages exchanged between PCEs is shown at the bottom of the figure.

The head-end of the LSP, $S$ sends a PCReq message to the PCE of its AS, $PCE1$. $PCE1$ sends a PCReq message to $PCE2$, the PCE inside $AS2$. The PCReq message contains the address of the tail-end of the LSP and the constraints for the LSP. The delay constraint is not necessary because the output of the computation technique is the shortest delay path following the given AS path. If the delay of the path returned to the LSP's head-end is above the delay constraint, there is no suitable path for the LSP respecting the given AS path. The LSP establishment fails.

$PCE2$ sends a PCReq to $PCE3$ because $AS3$ is the downstream AS to $AS2$ in the given AS path. $PCE3$ computes a path segment from $R31$ to $D$, the tail-end of the LSP. Then, it sends the segment with its delay in PCRep message (3) upstream to $PCE2$.

When $PCE2$ receives PCRep (3), it computes path segments from the entrances inside its domain to the destination of the LSP. For this purpose, the PCE performs a Shortest Path First (SPF) computation on the graph composed of the local topology, the inter-AS links and the segments received from the downstream PCE. This results in two path segments starting at node $R21$ and node $R23$ respectively and ending at $D$.

Next, $PCE2$ sends the resulting segments and their delays inside PCRep $(4)$ to $PCE1$. After receiving the reply from $PCE2$, $PCE1$ computes the end-to-end path based on the local topology, the inter-AS links connected to $AS1$ and the received segments. The resulting path is $S$ - $R11$ - $R14$ - $R23$ - $R31$ - $D$ with delay of 17 ms. This path is sent in PCRep (5) to the head-end of the LSP, $S$. Finally, $S$ initiates the establishment of the LSP along this path. For this purpose it stores the path returned by $PCE1$ inside the ERO. Thus, the RSVP `Path` message follows the computed path and the LSP is established along this path.

In order to respect the confidentiality requirement of ISPs (see Section 6.1.1), PCEs may return an aggregated RRO or path keys [16] inside PCRep messages, instead of returning path segments that reveal sequences of hops inside their domains.

BRPC relies on PCEs that communicate and cooperate in order to find the shortest path respecting given QoS constraints along a given AS path. Here, PCEs have to discover the PCEs in neighboring domains [53]. With BRPC, contrary to the per-domain path computation technique, it is possible to simultaneously compute a pair of disjoint LSPs, as described in

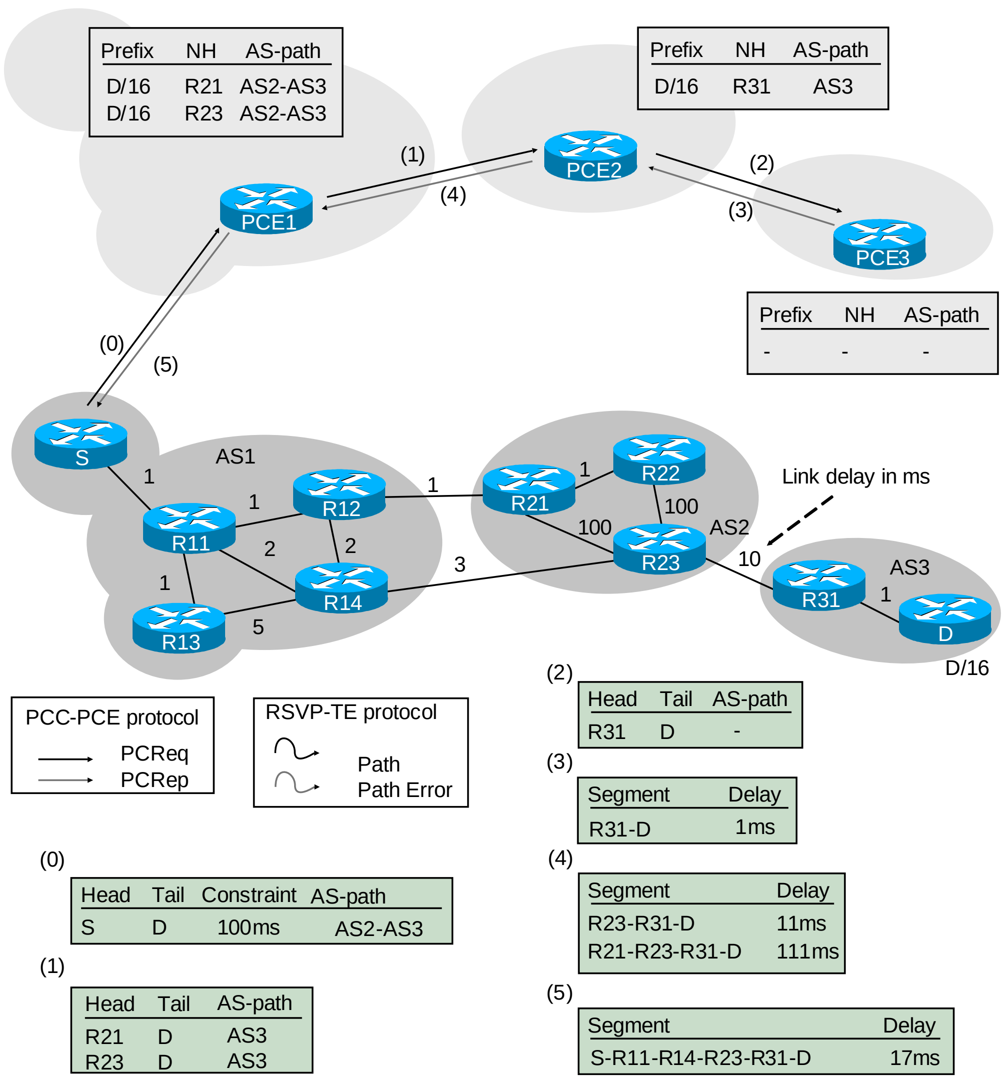

Prefix NH AS-path
D/16 R21 AS2-AS3
D/16 R23 AS2-AS3
Prefix NH AS-path
D/16 R31 AS3
Prefix NH AS-path
- - -
PCE1
PCE2
PCE3
(0)
(1)
(2)
(3)
(4)
(5)
S
AS1
R11
R12
R13
R14
R21
R22
R23
AS2
R31
AS3
D
D/16
Link delay in ms
1
2
5
3
100
10
PCC-PCE protocol
PCReq
PCRep
RSVP-TE protocol
Path
Path Error
(2) Head Tail AS-path / R31 D -
(3) Segment Delay / R31-D 1ms
(4) Segment Delay / R23-R31-D 11ms / R21-R23-R31-D 111ms
(5) Segment Delay / S-R11-R14-R23-R31-D 17ms
(0) Head Tail Constraint AS-path / S D 100ms AS2-AS3
(1) Head Tail AS-path / R21 D AS3 / R23 D AS3

Figure 6.5: Backward recursive PCE-based path computation.

[102], when the AS path for the pair of LSPs is given.

## 6.3.3 Applicability of the path computation techniques

The PCE-based architecture makes it possible to set up inter-AS LSPs with end-to-end QoS constraints. However the following aspects still need to be addressed before having a practically usable solution:

- All the proposed path computation methods assume the awareness of the AS path. If such an AS path is not specified, a solution has to be found to compute this AS path.

- As the QoS properties of the AS paths are not known by the source PCE, using a randomly picked AS path gives little confidence in finding an end-to-end path respecting the QoS constraints. Hence, the QoS properties of AS paths must be obtained or determined in one way or another.

Recall the lower bound on routing diversity provided in Section 6.1.2. Using the BGP best paths, we may obtain a poor sample of the available end-to-end QoS and might fail in finding a feasible path. If the AS path is not given to the PCE for path computation as a priori, the PCE may need to try all learned AS paths. The number of AS paths learned by the PCE can be several times that of the BGP best AS paths selected by the AS, and lead to a computation with much complexity.

## 6.4 Towards inter-AS QoS

Section 6.2 discussed the existing techniques to support interdomain LSPs. Section 6.3 introduced the path computation architecture and techniques. These two components are able to work together and establish inter-AS LSPs with end-to-end QoS constraints. However, before inter-AS LSPs with QoS guarantees are a reality, enough information needs to be known by the entities that will compute the end-to-end path in each AS. The techniques presented in Section 6.3 assume that the AS sequence of a feasible QoS path is known. Given the lack of QoS information available today in the Internet, this seems to be a strong assumption.

Computing an end-to-end QoS path requires finding a trade-off between the amount of QoS information to be distributed across the Internet and the complexity of finding QoS paths. We expect that a solution to the inter-AS QoS problem will address the following two aspects:

- What information should be distributed across the Internet to enable the QoS path computation and how should this information be distributed?

- How should the propagated QoS information be used by an AS willing to establish a QoS path?

In this section, we discuss separately each of the two above questions. The first question, which consists of two inter-related aspects, the content and the distribution of QoS information, is discussed in Section 6.4.1. Section 6.4.2 then presents a possible scenario of end-to-end LSP computation with QoS constraints.

### 6.4.1 Distributing QoS information for inter-AS LSPs

In this section, we first figure out the information that is necessary to make the end-to-end LSP computation and establishment possible, and give a conceptual solution on QoS information aggregation and propagation. We show that PCE proposed in the PCE based architecture is a possible candidate to carry out QoS status computation, path preference computation, etc; and TED may serve to store the resulted “good” paths. At the end, we discuss the possible ways of distributing QoS information.

**Necessary information** Several types of information have to be available to make possible the end-to-end LSP computation and establishment. First, loose or strict path information has to be disseminated. By loose topological information, we mean AS sequences that provide a given QoS. Less loose topological information would be IP-level paths. In practice, we do not expect that ISPs will be willing to reveal IP-level information, both for confidentiality reasons, and also because it would make the computation framework not scalable.

Second, QoS information about the link state has to be available at least within ISP networks. For this, traffic engineering extensions to IGP protocols have to be used, and measurements have also to be carried out by the ISPs inside their AS. Traffic engineering extensions to IGP protocols [48, 86] can provide information about the maximum bandwidth and maximum reservable bandwidth on a link. For delay-related information, measurements have to be used [23]. Depending on how the path computation will be carried out, this information may have to be propagated to special nodes (e.g., PCEs) and kept up to date.

Finally, policies are likely to apply on the use of the resources, as ISPs want to retain control of their own resources. Today, routing policies are not explicitly revealed beyond AS boundaries, because they are bound to contractual agreements. We see no reason why the same rules would not apply to QoS paths. ISPs will decide how much resources can be allocated to QoS paths, and under which conditions. This will probably lead to distinct business relationships for QoS traffic. If ISPs are bound by contractual agreements for QoS traffic, then it is unlikely that a centralized framework for distributing and computing QoS paths will be used. As the current routing architecture is highly distributed, it is unlikely that a centralized solution would be adopted on the short-term, i.e. within the next few years.

**A conceptual solution** A conceptual solution for providing QoS information for BGP next-hops and routes is for ASs to summarize their QoS states towards a specific destination and propagate this information to neighboring ASs according to their QoS-related business agreements. For the summary of an AS’s QoS states towards the destination, work needs to be done in order to allow the QoS summaries from each AS to be combined and propagated along the AS sequence. Moreover, the QoS state should be summarized in such a way that the summary is not very sensitive to the variations of the real QoS state, such that the QoS summary does not need to be updated very frequently, as proposed in [106].

**Path selection and content of TED** The PCE of an AS may be responsible for computing the QoS status for border routers inside the AS. It can be configured to compute and assign

QoS states for border routers in the AS's desired way. E.g., billing as a big drive for the routing policies might be taken into account in the form of preferences to next-hops/routes. ASs would configure their PCEs to assign higher preferences to cheap next-hops/routes which can still provide certain QoS guarantees. The PCE can also selectively store the QoS states for next-hops/routes in the TED. Only a few next-hops/routes with "good" QoS states will contribute to the establishment of inter-AS LSPs with QoS constraints, and whether the QoS states of the rest of next-hops/routes are known in the TED will not affect the path computation outcome. Scalability in the size of the TED can be obtained by caching only the most promising entries (not the full RIB-Ins) from a QoS perspective.

**Centralised vs. distributed information dissemination** Distributing the QoS information across the Internet can be achieved in two ways. First, each AS can push their QoS information to a centralised system that will handle the requests for inter-AS LSP establishment. Requests for Inter-AS LSPs with QoS guarantees will be sent to the centralised system, that will try to find an end-to-end AS path that meets the QoS guarantees. Such a solution is currently investigated by the IPsphere FORUM [46]. The IPsphere FORUM is developing a solution to determine the ASs to cross for an LSP with given QoS requirements and taking into account the business relationships of the ASs. The participation to this IPsphere FORUM is based on membership, and their work is not publicly available. We will not discuss further the centralised solution to the QoS problem, but rather focus on distributed solutions based on the current routing architecture.

The second method to distribute QoS information is to rely on the current distributed model of routing in the Internet. In this model, each AS is responsible to propagate to its neighbors reachability information about every destination. The same principle could apply to propagate QoS-related information. Some works [15, 106] have proposed QoS extensions to BGP. The idea is to use BGP to piggyback QoS information since BGP already propagates reachability information across the Internet. [15] proposed to add a new type of attribute to BGP messages. This new type of BGP attribute gives freedom to ASs to specify whether the QoS information is transitive or if only their neighbors should know about it. Several attributes are proposed in [15], like maximum bandwidth, available capacity, minimum and maximum transit delay. The QoS attributes are associated to the forwarding path towards the destination prefix of the BGP route. This QoS information thus only concerns the best paths chosen by BGP, which we know represent only a small fraction of the whole path diversity known to ASs in the Internet. [106] proposed four statistical QoS metrics, to make the QoS extensions to interdomain routing scalable while ensuring the optimality of QoS routing. [106] proposes to represent QoS information by intervals of the metric values, $[l, u]$, together with a probability $p$ that the instantaneous value of the metric belongs to this interval. As in [15], the QoS information in [106] is related to the best routes selected by BGP.

### 6.4.2 Computing inter-AS LSPs with end-to-end QoS constraints

We expect that providing to ASs appropriate AS paths will be one of the challenges to solve the inter-AS QoS problem. As obtaining a complete view of the topology and traffic engineering information is hardly possible, a PCE that computes a portion of a constrained interdomain LSP must rely on heuristics to choose an appropriate AS path and BGP next-hop among the ones announced for the destination. If a bad choice is made by the heuristic at some PCE, a downstream PCE may not be able to complete the computation of the path. In that case, the solution is to rely on crankback to try alternative AS paths and next-hops. Even though we do not expect that this will be necessary, in the worst case the PCE still needs to search through the whole space of possible AS paths and next-hops.

In [73], Pelsser et al. propose two heuristics, namely "nearest next-hop" and "Vivaldi" [26], for the selection of the ingress node (i.e. the next-hop) in the downstream domains and combine them with the per-domain path computation technique. The "nearest next-hop" selects the next-hop based on the delay information locally available to the domain, and the "Vivaldi" next-hop selection is based on an estimation of the delay of the path that transits through the candidate next-hop. These heuristics select next-hops for inter-AS LSPs with the end-to-end delay as the QoS constraints. It is shown from simulations that these heuristics can limit the computational requirements for finding feasible end-to-end QoS paths. These heuristics work in the absence of QoS information propagation. They serve for the path computation for inter-AS LSPs with the end-to-end delay as the QoS constraints.

**Path computation with QoS information** If, as discussed in Section 6.4.1, the QoS information is available, then it can be directly used as a heuristic for BGP next-hop/AS path selection. It should be designed specifically for path computation for inter-AS LSPs with QoS constraints. It should thus maximize the likelihood that a feasible path will be found. In this section, we show how the path computation can be carried out using the PCE architecture proposed in the IETF with QoS information available.

Once the AS-level path on which to establish the LSP is chosen, it is up to the path computation techniques to compute intra-AS segments thanks to the TED (see Section 6.3.2). The content of the TED is used to find out the best next-hop that satisfies the QoS constraints on the AS path. Once the next-hop is found, the segment is found by the path computation techniques. We show in Figure 6.6 how QoS descriptions can be computed and recorded in TEDs, and propagated with BGP. In Figure 6.7, we show how the per-domain path computation technique uses this information on QoS descriptions to compute the path.

In Figure 6.6 and 6.7, we use delay as the QoS metric. BGP routers propagate their QoS states to all peering BGP routers, while in reality, they may propagate to a selected subset of peering BGP routers according to policies. Each BGP router (BR in the figures), upon receiving BGP QoS updates from border routers towards a destination, will inform the PCE in this AS about this updated QoS state information towards the destination as well as wherefrom the updates were received (i.e., the next-hop). The PCE will compute and record

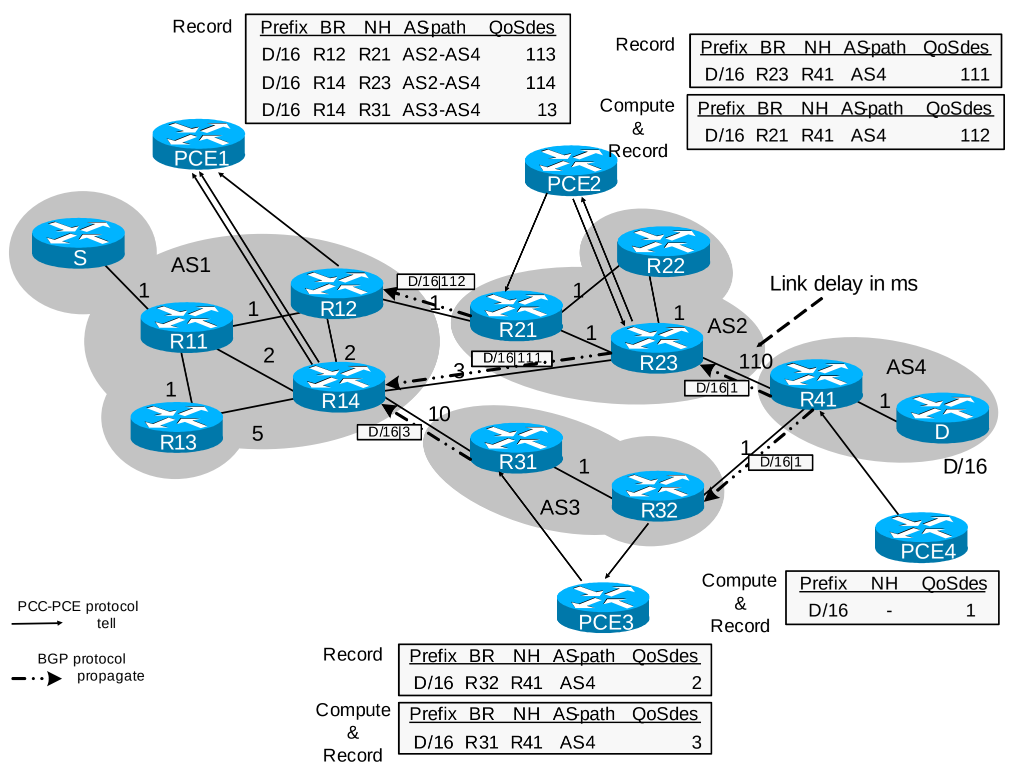
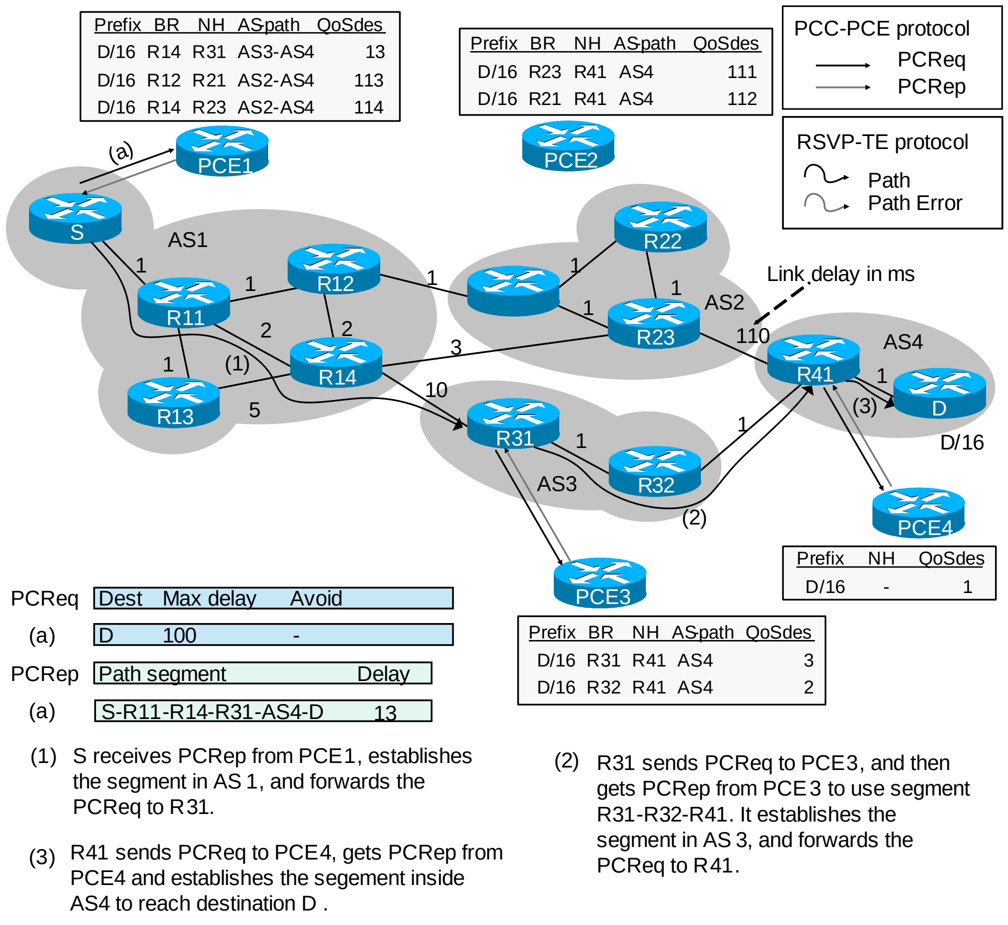


Figure 6.6: QoS summary propagation.

the QoS state of all the BGP routers in this AS towards the destination. Thus the PCE knows the AS-wide QoS state towards all reachable destinations and via which next-hop border router this QoS state might be achieved.

Upon receiving a Path Computation Request (PCReq), the PCE will compute the path based on the information recorded in the TED including the intradomain topology information. Different path computation techniques (the per-domain path computation technique or the BRPC path computation technique) can be used, as our framework tells the PCE where to forward the PCReq but does not stick to a single path computation technique. In the example in Figure 6.7, an LSP with delay constraint of 100 ms needs to be established from $S$ to $D$. The head-end LSR $S$ sends PCReq $(a)$ to the PCE in charge of the path computation in $AS1$. $PCE1$ knows from its TED that next-hop $R31$ has the smallest delay towards $D/16$, and computes the path segment in $AS1$ towards $R14/R31$. PCReq is forwarded to $R31$ which sends PCReq to $PCE3$. $PCE3$ selects next-hop $R41$ and computes the path segment inside $AS3$ towards $R32/R41$. $R41$ receives the PCReq and asks $PCE4$ to compute the path segment towards $D/16$. The LSP can then be established with the segments computed in each AS.



(1) S receives PCRep from PCE1, establishes the segment in AS 1, and forwards the PCReq to R31.

(2) R31 sends PCReq to PCE3, and then gets PCRep from PCE3 to use segment R31-R32-R41. It establishes the segment in AS 3, and forwards the PCReq to R41.

(3) R41 sends PCReq to PCE4, gets PCRep from PCE4 and establishes the segement inside AS4 to reach destination D .

Figure 6.7: Per-domain path computation using QoS information.

### 6.4.3 Qualitative evaluation with regard to the inter-AS LSP requirements

In this section, we qualitatively evaluate our solution with regard to the inter-AS LSP requirements specified in Section 6.1.3.

- The two processes that exchange information between ASs are: 1) the QoS information propagation process; and 2) the process of setting up inter-AS LSPs along the routes computed based on the received QoS information. The QoS information propagated to neighboring ASs by a BGP router only provides the QoS state of this BGP router and the abstract route (the AS-path). Thus, no internal AS resources information is revealed. The RSVP-TE extensions to support interdomain LSPs already fulfil this requirement, as explained in Section 6.2. The confidentiality of AS resources is achieved during both processes. The task of path optimization inside each transited AS, can be carried out by the PCEs that are responsible for LSP segment computation in those ASs.

- The protection of inter-AS LSPs were already addressed in Sections 6.2.3 and 6.2.4.

- In section 6.2.5, we showed the scalability property of using RSVP-TE extensions to establish inter-AS LSPs. The other functional components that may have scalability issues are 1) the propagation of QoS information using BGP; 2) the storage of received QoS information in TED; and 3) path computation based on the information in TED. Using BGP to propagate QoS information only requires adding a new contribute in the BGP message. The size of the BGP message with QoS information will be at the same order as the normal BGP message. As explained in Section 6.4.1, only a few "good" enough next-hops/routes will contribute to the establishment of inter-AS QoS guaranteed LSPs, and thus, only these next-hops/routes need to be stored in the TED. The number of next-hops/routes stored in TED scales with the number of destinations. The path computation based on the information in TED is straightforward as long as the AS specifies its principles for route selection.

## 6.5 Conclusion and perspectives

In this chapter, we explained the complexity of establishing inter-AS LSPs with QoS guarantees. We described the working of the interdomain routing system, and the assumptions on which it relies. We discussed the consequences of the path selection by the current interdomain routing system on the visibility of the interdomain paths. The limited visibility of the available paths does not actually prevent to establish inter-AS LSPs with QoS guarantees. Rather, this limited visibility requires that clever heuristics be designed in order to find the feasible QoS paths. We covered the existing signaling extensions to RSVP-TE that support the establishment of inter-AS LSPs, as well as the protection of those LSPs. The path computation techniques that have been proposed at the IETF were also detailed. Those computation techniques make it possible to find the LSP segments within each AS, so as to compose an end-to-end LSP with QoS guarantees. Finally, we put together those three components, i.e. interdomain routing, signaling, and path computation techniques, and show that inter-AS QoS is not out of reach, but that more work needs to be done in specific areas such as the propagation of QoS related information across ASs.

Even though the content of this chapter is intended to be as factual as possible, we believe that it argues in favor of the feasibility of providing end-to-end QoS in the Internet. The challenges that have to be faced in order to push QoS beyond AS boundaries are by no means out of reach. Dissemination of QoS information using the current interdomain routing system has already been proposed several times. Due to concerns about the scalability of interdomain routing, those proposals have however not received much interest from the networking community. We believe that the current evolution of the Internet towards more stringent services asks for end-to-end QoS capabilities. Before the deployment of MPLS by ISPs, pushing QoS in the Internet as proposed through Differentiated Services (DiffServ) [12] or Integrated Services (IntServ) [85] was very difficult as it involved significant changes to the core of the Internet. With the wider deployment of BGP/MPLS VPNs in large ISP networks, the situation of today's Internet core makes QoS more and more relevant. We have shown in this chapter that most of the building blocks to make QoS path computation possible have already been defined, even though not completely standardized nor deployed yet. The work carried out at the IETF within the Path Computation Element (PCE) working group is a proof of the interest given by the community to make possible the computation of end-to-end QoS guaranteed paths. This working group has however decided so far to limit its work to path computation within a single domain, given the difficulties of crossing AS boundaries when it involves different companies. We believe that the community should investigate more into inter-AS QoS to make the Internet a better place for QoS-demanding applications.

# Chapter 7

# Conclusions

Applications and services that require QoS guarantees emerge as the Internet expands to contain more users and to deploy more advanced technologies. ISPs are driven to rely on traffic engineering [9] to better control the flow of IP packets, to provide a good service and to fulfill their network management goals. This thesis focuses on the following aspects of the broad problem of Traffic Engineering and QoS in the Internet:

- Intradomain traffic engineering with an emphasis on resources usage.
- How ISPs/ASs carry out their interdomain traffic engineering in reality.
- How to enable end-to-end QoS across the Internet.

This concluding chapter summarizes the findings of our research work done on these topics.

## 7.1 Intradomain traffic engineering

In this thesis, we studied the impact of different factors on the performance of traffic engineering/QoS routing algorithms and thus the scenarios in which traffic engineering/QoS routing can help efficiently use the network resources. We covered the factors such as the accuracy of link state information known by the path computation entity, the network load, the underlying network topology, the provisioning of network resources, *etc*.

Specifically in chapter 2, we investigated the impact of stale link state information on the performance of traffic engineering/QoS routing algorithms by comparing two extreme link state update policies (LSUPs), namely *routing with exact resource information* and *routing with static information*. The extremes react differently to different network models. With the same network model, the difference in performance varies with the network load, and stays small under low network loads. When the amount of traffic running in the network is relatively small compared to the high capacity provided by the core network, routing without LSUs is expected to give satisfactory performance. However, under very high network loads, the intelligence of knowing exactly the resource information backfires, because it leads to longer-hop paths, which results in a more congested network, leading to high blocking ratios. If we would have counted the amount of traffic overhead induced by the LSUs, the performance of *routing with exact resource information* would be even worse.

Further in chapter 3, we studied the behavior of on-line TE algorithms under different scenarios with a more systematic approach. The first contribution of this chapter is defining three distinct load regimes (low, medium, and high) that correspond to different behaviors of the on-line TE algorithms. In the second part of this chapter, we studied the behavior of on-line TE algorithms using synthetic network topologies and traffic demands that aim to show the impact of network design aspects on the performance of on-line TE algorithms. As long as the network is properly provisioned, on-line TE algorithms do not provide a significant improvement in using the available capacity, even in networks with much topological redundancy. When network provisioning does not match the traffic demand, on-line TE algorithms are able to use the redundant links to compensate for the poor provisioning. We expect that most real-world networks are in a low load regime, where shortest-path routing is good enough. However, as the traffic demand evolves, strict QoS might be required, and on-line TE/QoS routing is necessary.

Potential further work on identifying the scenarios in which on-line traffic engineering helps, is to develop a metric that quantifies how badly the current network does not match the current traffic demand. This metric would take into account the network growth and the routing algorithm used, and give insight into when the situation is becoming critical enough so that either on-line TE algorithms should be used, or when the network must be upgraded. We also expect that the topological structure has a non-trivial impact on the behavior of on-line TE algorithms, and further investigation is needed.

## 7.2 Interdomain traffic engineering – the reality

A better understanding on how ISPs/ASs carry out their interdomain traffic engineering in reality was given in Chapter 5. Our contribution consists of an AS-level topology model and a concept for capturing how individual ASs choose their best paths.

We built an AS-level topology model with a minimal connectivity while still enabling the propagation of all routes observed in BGP data. This AS-level topology model takes into account the internal structure of ASs, and thus serves better as a topology model for interdomain routing than those using single router per AS.

We analyzed the known bounds for policies studied in the literature: BGP atoms and business relationships. BGP atoms do not discriminate different interdomain links and parts of the topology, and are prone to generalize and would suggest that probably all ASs have their policies defined on a per-prefix level. The *valley-free* properties used for business relationships inference were validated in [61]. However, business relationships do not help to decide which paths among the candidates should be chosen by each AS: after enforcing policies in the model in the form of business relationships, much choice is left as to which route to choose as the best among the candidates. Business relationships do not contain enough information about the path choices made by ASs.

To capture the way individual ASs choose their best paths, we introduced a new abstraction: next-hop atoms. Next-hop atoms capture the different sets of neighboring ASs an AS uses for its best routes. We showed that a large fraction of next-hop atoms correspond to per-neighbor path choices. A non-negligible fraction of path choices however do not correspond to simple per-neighbor preferences, but hot-potato routing and tie-breaking within the BGP decision process, which are very detailed aspects of Internet routing.

Further analysis done in a dynamic scenario shows that the selection of next-hop ASs to reach a certain destination prefix is quite stable over time. There are a large fraction of prefixes that have their $1$st dominant next-hop ASs being used for more than $99.9\%$ of the time. For almost all the prefixes, the $1$st $2$ dominant next-hop ASs already contribute to nearly $100\%$ of the time. While for almost all the prefixes, the $3$rd dominant next-hop AS does not contribute or only contributes to a small fraction of time. It suggests that modeling the dynamic aspects of Internet routing is not impossible.

In future work we will validate the policies we derived by testing their predictive capabilities and also by comparing them to actual policies configured by ASs as in [28].

## 7.3 Interdomain QoS

In this thesis, we discussed the feasibility of having interdomain QoS a reality and the remaining challenges on which more efforts need to be spent in chapter 6. We covered the existing or already proposed components that are relevant to interdomain QoS: the current interdomain routing system; the signaling extensions to RSVP-TE that support the establishment of inter-AS LSPs, as well as the protection of those LSPs; and the path computation techniques proposed at the IETF.

We discussed the consequences of the path selection and distribution made by the current interdomain routing system on the visibility of the paths, and showed that diverse paths exist today but their QoS is unknown. The current lack of knowledge about the QoS available from BGP routes makes the evaluation of different solutions to the interdomain QoS problem challenging.

With the signaling extensions to RSVP-TE that support the establishment and protection of inter-AS LSPs, interdomain LSPs can be established with the confidentiality requirement and the protection requirement satisfied once the specific routes and the local backup LSP segments for protection are defined.

The path computation techniques proposed at the IETF make it possible to compute the LSP segments within each AS, so as to compose an end-to-end LSP with QoS guarantees when the sequence of ASs to be crossed is known.

Finally, we put together those three components, i.e., interdomain routing, LSP signaling, and path computation techniques. We show that inter-AS QoS is not out of reach, but that more work needs to be done in specific areas, especially concerning how to compute feasible QoS paths. Blind computation techniques without QoS hints or techniques coupled with local heuristics have significant limitations. A better approach would be for ASs to propagate QoS information to guide the search torwards AS sequences across which feasible QoS paths can be found. Computing an end-to-end QoS path requires finding a trade-off between the amount of QoS information to be distributed across the Internet and the complexity of finding QoS paths. Two subproblems need to be addressed by the community, namely *the propagation of the QoS information* and *the selection of an AS sequence based on the propagated QoS information*.

# Abbreviations

| | |
|---|---|
| AS | Autonomous System |
| BGP | Border Gateway Protocol |
| BRPC | Backward Recursive PCE-based Computation |
| CDN | Content Distribution Network |
| CSPF | Constrained Shortest Path First |
| ECMP | Equal-Cost Multi-Path |
| ERO | Explicit Route Object |
| GMPLS | Generalized Multi Protocol Label Switching |
| IETF | Internet Engineering Task Force |
| IGP | Interior Gateway Protocol |
| IP | Internet Protocol |
| IS-IS | Intermediate System-Intermediate System |
| ISP | Internet Service Provider |
| LSP | Label Switched Path |
| LSUP | Link State Update Policy |
| MIRA | Minimum Interference Routing Algorithm |
| MED | Multi-Exit-Discriminator |
| MOAS | Multiple Origin Autonomous System |
| MPLS | Multi Protocol Label Switching |
| MST | Minimum Spanning Tree |
| OSPF | Open Shortest Path First |
| PCE | Path Computation Element |
| POP | Point of Presence |
| QoS | Quality of Service |
| RIB-In | Routing Information Base-Inbound |
| RRO | Record Route Object |
| RSVP | Resource Reservation Protocol |
| SRLG | Shared Risk Link Group |
| SSP | Static Shortest Path |
| TE | Traffic Engineering |
| TED | Traffic Engineering Database |

| | |
|---|---|
| VOD | Video on Demand |
| VoIP | Voice over IP |
| VPN | Virtual Private Network |
| WSP | Widest-Shortest Path |

# Bibliography

[1] RIPE's Routing Information Service. `http://www.ripe.net/ris/`.

[2] Abilene Observatory. `http://abilene.internet2.edu/observatory/`.

[3] Y. Afek, O. Ben-Shalom, and A. Bremler-Barr. On the Structure and Application of BGP Policy Atoms. In *Proc. of ACM SIGCOMM IMW '02*, 2002.

[4] G. Apostolopoulos, R. Guerin, and S. Kamat. Quality of service based routing: A performance perspective. In *Proc. of ACM SIGCOMM*, 1998.

[5] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi. Improving Qos routing performance under inaccurate link state information. In *Proc. of ITC-16*, 1999.

[6] A. Ariza, E. Casilari, and F. Sandoval. Strategies for updating link states in QoS routers. *Electronic Letters*, vol. 36(No. 20), 2000.

[7] J. Ash and J-L. Le Roux. A path computation element (PCE) communication protocol generic requirements. *Request for Comments 4657, IETF*, Sept 2006.

[8] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. RSVP-TE: Extensions to RSVP for LSP tunnels. *Request for Comments 3209, IETF*, December 2001.

[9] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of internet traffic engineering. *Request for Comments 3272, IETF*, May 2002.

[10] G. Battista, M. Patrignani, and M. Pizzonia. Computing the Types of the Relationships between Autonomous Systems. In *Proc. of IEEE INFOCOM*, 2003.

[11] G. Di Battista, T. Erlebach, A. Hall, M. Patrignani, M. Pizzonia, and T. Schank. Computing the Types of the Relationships between Autonomous Systems. *IEEE/ACM Transactions on Networking*, 15(2):267–280, April 2007.

[12] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. *Request for Comments 2475, IETF*, December 1998.

[13] F. Blanchy, L. Mélon, and G. Leduc. An efficient decentralized on-line traffic engineering algorithm for MPLS networks. In *Proc. of ITC-18, Berlin, Germany*, 2003.

[14] Jean-Chrysotome Bolot. End-to-end packet delay and loss behavior in the Internet. In *Proc. of ACM SIGCOMM*, pages 289–298, 1993.

[15] O. Bonaventure. Using BGP to distribute flexible QoS information. *Internet draft, draft-bonaventure-bgp-qos-00, work in progress*, February 2001.

[16] R. Bradford, J-P. Vasseur, and A. Farrel. Preserving topology confidentiality in inter-domain path computation using a key-based mechanism. *Internet draft, draft-ietf-pce-path-key-01.txt, work in progress*, Sept 2007.

[17] A. Broido and KC. Claffy. Analysis of RouteViews BGP Data: Policy Atoms. In *Proceedings of the Network-Related Data Management workshop, Santa Barbara*, May 2001.

[18] M. Caesar and J. Rexford. BGP Routing Policies in ISP Networks. *IEEE Network Magazine*, 2005.

[19] R. Cahn. *Wide area network design: concepts and tools for optimization.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[20] R. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. *Request for Comments 1195, IETF*, December 1990.

[21] H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Towards Capturing Representative AS-Level Internet Topologies. *Computer Networks*, 44(6), April 2004.

[22] M. Chen, R. Zhang, and X. Duan. OSPF extensions in support of Inter-AS multiprotocol label switching (MPLS) and generalized MPLS (GMPLS) traffic engineering. *Internet draft, draft-ietf-ccamp-ospf-interas-te-extension-02.txt, work in progress*, Nov 2007.

[23] B. Choi, S. Moon, Z. Zhang, K. Papagiannaki, and C. Diot. Analysis of point-to-point packet delay in an operational network. *Computer Networks*, 51(13):3812–3827, 2007.

[24] Cisco. OSPF design guide. `http://www.cisco.com/warp/public/104/1.html`.

[25] K. Claffy, H. Braun, and G. Polyzos. Traffic characteristics of the T1 NSFNET backbone. In *INFOCOM*, 1993.

[26] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proc. of ACM SIGCOMM*, 2004.

[27] S. De Cnodder and C. Pelsser. Protection for inter-AS MPLS tunnels. *Internet draft, draft-decnodder-ccamp-interas-protection-00.txt, work in progress*, July 2004.

[28] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, kc claffy, and G. Riley. AS Relationships: Inference and Validation. *ACM Comput. Commun. Rev.*, 37(1), 2007.

[29] A. Farrel, A. Satyanarayana, A. Iwata, N. Fujita, and G. Ash. Crankback signaling extensions for MPLS and GMPLS RSVP-TE. *Request for Comments 4920, IETF*, July 2007.

[30] A. Farrel, J-P. Vasseur, and J. Ash. A path computation element (PCE) based architecture. *Request for Comments 4655, IETF*, Aug 2006.

[31] A. Farrel, J-P. Vasseur, and A. Ayyangar. A framework for inter-domain multiprotocol label switching traffic engineering. *Request for Comments 4726, IETF*, Nov 2006.

[32] N. Feamster, H. Balakrishnan, and J. Rexford. Some foundational problems in interdomain routing. In *In ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, 2004.

[33] N. Feamster, Z. Mao, and J. Rexford. BorderGuard: Detecting Cold Potatoes from Peers. In *Proc. ACM IMC*, 2004.

[34] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: methodology and experience. In *Proc. of ACM SIGCOMM*, 2000.

[35] M. Fomenkov, K. Keys, D. Moore, and K. Claffy. Longitudinal study of internet traffic in 1998-2003. In *Proc. of WISICT'04*, 2004.

[36] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, October 2002.

[37] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proc. of IEEE INFOCOM*, Tel-Aviv, Israel, March 2000.

[38] L. Gao. On inferring autonomous system relationships in the internet. In *Proc. IEEE Global Internet*, 2000.

[39] T. Griffin, F. Bruce Shepherd, and G. Wilfong. The Stable Paths Problem and Interdomain Routing. *IEEE/ACM Transactions on Networking*, 2002.

[40] R. Guerin and A. Orda. Qos-based routing in networks with inaccurate information: Theory and algorithms. *IEEE/ACM Transactions on Networking*, June 1999.

[41] R. Guerin, D. Williams, and A. Orda. QoS routing mechanisms and OSPF extensions. In *Proc. of IEEE Globecom*, 1997.

[42] J. Hao, M. Meulle, and Q. Nguyen. Formulation CSP et approches heuristiques pour l'inférence des accords d'interconnexion dans l'internet. In *In ROADEF'06*, 2006.

[43] B. Huffaker, M. Fomenkov, D.J. Plummer, D. Moore, and k claffy. Distance metrics in the internet. In *IEEE International Telecommunications Symposium*, 2002.

[44] G. Huston. `http://www.potaroo.net/`.

[45] Intel-DANTE. Intel-DANTE Monitoring Project. `http://rtmon.gen.ch.geant2.net/`.

[46] IPsphere FORUM - the business of IP. `http://www.ipsphereforum.org/home`.

[47] M. Kabatepe and M. G. Hluchyj. On the effectiveness of topology update mechanisms for ATM networks. In *Proc. of ICC'98*, 1998.

[48] D. Katz, K. Kompella, and D. Yeung. Traffic engineering (TE) extensions to OSPF version 2. *Request for Comments 3630, IETF*, Sept 2003.

[49] T. Kleiberg, B. Fu, F.A. Kuipers, S. Avallone, B. Quoitin, and P. Van Mieghem. DeSiNe: a flow-level QoS simulator of Networks. In *Proc. of the first International Workshop on the Evaluation of Quality of Service through Simulation in the Future Internet (QoSim)*, 2008.

[50] M. S. Kodialam and T. V. Lakshman. Minimum interference routing with applications to MPLS traffic engineering. In *Proc. of IEEE INFOCOM*, Tel-Aviv, Israel, March 2000.

[51] J. Kowalski and B. Warfield. Modeling traffic demand between nodes in a telecommunications network. In *Proc. of ATNAC'95*, 1995.

[52] F.A. Kuipers, T. Korkmaz, M. Krunz, and P. Van Mieghem. An overview of constraint-based path selection algorithms for Qos routing. *IEEE Communication Magazine*, 40(12):50–55, December 2002.

[53] J-L. Le Roux. Requirements for path computation element (PCE) discovery. *Request for Comments 4674, IETF*, Oct 2006.

[54] B. Lekovic and P. Van Mieghem. Link state update policies for Quality of service routing. *Eighth IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT2001)*, 2001.

[55] Q. Ma and P. Steenkiste. On path selection for traffic with bandwidth guarantees. In *Proc. of ICNP'97*, 1997.

[56] Q. Ma and P. Steenkiste. Qos routing for traffic with performance guarantees. In *Proc. of IFIP Int. Workshop Quality of Service*, 1997.

[57] E. Mannie. Generalized multi-protocol label switching (GMPLS) architecture. *Request for Comments 3945, IETF*, October 2004.

[58] Z.M. Mao, L. Qiu, J. Wang, and Y. Zhang. On AS-level Path Inference. In *Proc. ACM SIGMETRICS*, 2005.

[59] Z.M. Mao, J. Rexford, J. Wang, and R.H. Katz. Towards an Accurate AS-level Traceroute Tool. In *Proc. of ACM SIGCOMM*, 2003.

[60] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: existing techniques and new directions. In *Proc. of ACM SIGCOMM*, pages 161–174, 2002.

[61] W. Mühlbauer, S. Uhlig, B. Fu, M. Meulle, and O. Maennel. In search for an appropriate granularity to model routing policies. In *Proc. of ACM SIGCOMM*, 2007.

[62] P. Van Mieghem. *Performance Analysis of Communications Networks and Systems*. Cambridge University Press, 2006.

[63] J. Moy. OSPF version 2. *Request for Comments 2328, IETF*, April 1998.

[64] W. Mühlbauer, A. Feldmann, O. Maennel, M. Roughan, and S. Uhlig. Building an AS-Topology Model that Captures Route Diversity. In *Proc. of ACM SIGCOMM*, 2006.

[65] S. Nelakuditi, Z. Zhang, R. P. Tsang, and D. H.C. Du. Adaptive proportional routing: a localized QoS routing approach. *IEEE/ACM Transactions on Networking*, 2002.

[66] A. Nucci, S. Bhattacharyya, N. Taft, and C. Diot. IGP link weight assignment for operational tier-1 backbones. *IEEE/ACM Transactions on Networking*, 15(4):789–802, 2007.

[67] Eiji Oki. PCC-PCE communication and PCE discovery requirements for inter-layer traffic engineering. *Internet draft, draft-ietf-pce-inter-layer-req-06.txt, work in progress*, Nov 2007.

[68] D. Oran. OSI IS-IS intradomain routing protocol. *Request for Comments 1142, IETF*, February 1990.

[69] P. Pan, G. Swallow, and A. Atlas. Fast reroute extensions to RSVP-TE for LSP tunnels. *Request for Comments 4090, IETF*, May 2005.

[70] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, and C. Diot. Analysis of measured single-hop delay from an operational backbone network. In *Proc. of IEEE INFOCOM*, June 2002.

[71] C. Pelsser and O. Bonaventure. RSVP-TE extensions for interdomain LSPs. Technical Report 2002-09, University of Namur, Oct 2002.

[72] C. Pelsser and O. Bonaventure. Extending RSVP-TE to support inter-AS LSPs. In *2003 Workshop on High Performance Switching and Routing (HPSR 2003)*, 2003.

[73] C. Pelsser and O. Bonaventure. Path selection techniques to establish constrained interdomain MPLS LSPs. In *In Proc. of IFIP International Networking Conference*, 2006.

[74] B. Quoitin. iGen topology generator. `http://www.info.ucl.ac.be/~bqu/igen/`.

[75] B. Quoitin. C-BGP, an Efficient BGP Simulator. `http://cbgp.info.ucl.ac.be/`, 2003.

[76] B. Quoitin and S. Uhlig. Modeling the Routing of an Autonomous System with C-BGP. *IEEE Network Magazine*, 2005.

[77] B. Quoitin and S. Uhlig. Modeling the routing of an autonomous system with C-BGP. *IEEE Network Magazine*, 2005.

[78] B. Quoitin, S. Uhlig, C. Pelsser, L. Swinnen, and O. Bonaventure. Interdomain traffic engineering with BGP. *IEEE Communications Magazine*, May 2003.

[79] E. Rosen and Y. Rekhter. BGP/MPLS IP virtual private networks (VPN)s. *Request for Comments 4346, IETF*, Feb 2006.

[80] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. *Request for Comments 3031, IETF*, January 2001.

[81] RouteViews Project. `http://www.routeviews.org/`.

[82] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *Proc. of ACM SIGCOMM*, pages 289–299, 1999.

[83] A. Shaikh, J. Rexford, and K. Shin. Evaluating the overheads of source-directed Qos routing. In *Proc. of ICNP*, 1998.

[84] A. Shaikh, J. Rexford, and K. Shin. Evaluating the impact of stale link state on Quality-of-service routing. *IEEE/ACM Transactions on Networking*, vol. 9, April 2001.

[85] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. *Request for Comments 2212, IETF*, September 1997.

[86] H. Smit and T. Li. Intermediate system to intermediate system (IS-IS) extensions for traffic engineering (TE). *Request for Comments 3748, IETF*, June 2004.

[87] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with Rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, Feb 2004.

[88] A. Sprintson, M. Yannuzzi, A. Orda, and X. Masip-Bruin. Reliable routing with QoS guarantees for Multi-Domain IP/MPLS networks. In *Proc. of IEEE INFOCOM*, 2007.

[89] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the Internet Hierarchy from Multiple Vantage Points. In *Proc. of IEEE INFOCOM*, 2002.

[90] T. Takeda, Y. Ikejiri, and J-P. Vasseur. Analysis of inter-domain label switched path (LSP) recovery. *Internet draft, draft-ietf-ccamp-inter-domain-recovery-analysis-02.txt, work in progress*, Sept 2007.

[91] R. Teixeira, N. Duffield, J. Rexford, and M. Roughan. Traffic Matrix Reloaded: Impact of Routing Changes. In *Proc. PAM*, 2005.

[92] R. Teixeira, K. Marzullo, S. Savage, and G. Voelker. In Search of Path Diversity in ISP Networks. In *Proc. ACM IMC*, 2003.

[93] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of Hot-Potato Routing in IP Networks. In *Proc. ACM SIGMETRICS*, 2004.

[94] K. Thompson, G. Miller, and R. Wilder. Wide-Area Internet traffic patterns and characteristics. *IEEE Network magazine*, 11(6), November/December 1997.

[95] S. Uhlig. Nonstationarity and high-order scaling in TCP flow arrivals: a methodological analysis. *ACM SIGCOMM Computer Communication Review*, 34(2), April 2004.

[96] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon. Providing public intradomain traffic matrices to the research community. *ACM SIGCOMM Comput. Commun. Rev.*, 36(1):83–86, 2006.

[97] S. Uhlig and S. Tandel. Quantifying the Impact of Route-Reflection on BGP Routes Diversity inside a Tier-1 Network. In *Proc. of IFIP Networking*, Coimbra, Portugal, May 2006.

[98] P. Van Mieghem, H. De Neve, and F.A. Kuipers. Hop-by-hop quality of service routing. *Computer Networks*, 37(3-4):407–423, 2001.

[99] J-P. Vasseur, A. Ayyangar, and R. Zhang. A per-domain path computation method for establishing inter-domain traffic engineering (TE) label switched paths (LSPs). *Internet draft, draft-ietf-ccamp-inter-domain-pd-path-comp-05, work in progress*, April 2007.

[100] J-P. Vasseur, Y. Ikejiri, and R. Zhang. Reoptimization of multiprotocol label switching (MPLS) traffic engineering (TE) loosely routed label switched path (LSP). *Request for Comments 4736, IETF*, Nov 2006.

[101] J-P. Vasseur and J-L. Le Roux. Path computation element (PCE) communication protocol (PCEP). *Internet draft, draft-ietf-pce-pcep-08.txt, work in progress*, July 2007.

[102] J-P. Vasseur, R. Zhang, N. Bitar, and J-L. Le Roux. A backward recursive pce-based computation (BRPC) procedure to compute shortest interdomain traffic engineering label switched paths. *Internet draft, draft-ietf-pce-brpc-06.txt, work in progress*, Sept 2007.

[103] F. Wang and L. Gao. Inferring and characterizing internet routing policies. In *Proc. of ACM SIGCOMM IMW*, 2003.

[104] H. Wang, H. Xie, L. Qiu, Y. Yang, Y. Zhang, and A. Greenberg. COPE: traffic engineering in dynamic networks. In *Proc. of ACM SIGCOMM'06*, 2006.

[105] J. Xia and L. Gao. On the evaluation of as relationship inferences. In *Proc. of IEEE GLOBECOM*, November 2004.

[106] L. Xiao, K-S. Lui, J. Wang, and K. Nahrstedt. QoS extension to BGP. In *Proc. of ICNP*, 2002.

[107] X. Yuan and A. Saifee. Path selection methods for localized Quality of service routing. In *Proc. of IC3N'01*, 2001.

[108] R. Zhang and M. Bartell. *BGP Design and Implementation.* CISCO Press, 2003.

[109] R. Zhang and J-P. Vasseur. MPLS inter-Autonomous System (AS) Traffic Engineering (TE) requirements. *Request for Comments 4216, IETF*, 2005.

[110] Y. Zhang and N. Duffield. On the constancy of Internet path properties. In *Proc. of ACM SIGCOMM IMW*, pages 197–211, 2001.

[111] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *Proc. of ACM SIGCOMM*, pages 301–312, 2003.

[112] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. Felix Wu, and L. Zhang. An Analysis of BGP Multiple Origin AS (MOAS) Conflicts. In *Proc. ACM SIGCOMM IMW*, 2001.

# Summary

Title: Traffic Engineering and Quality of Service in the Internet

The Internet, born in 1969, has become a “giant” that is crucial for the whole world. It has been and is being developed to make life easier and more enjoyable. As the applications relying on the Internet evolve, Internet Service Providers (ISPs) are driven to rely on traffic engineering to better control the flow of IP packets, to provide a good service and to fulfill their network management goals. Traffic engineering and Quality of Service (QoS) has become a popular topic in both the industry and the research community.

This thesis focuses on the following aspects of the broad problem of traffic engineering and QoS in the Internet:

- Intradomain traffic engineering with an emphasis on resources usage.
- How of ISPs/Autonomous Systems (ASs) carry out their interdomain traffic engineering in reality.
- How to enable end-to-end QoS across the Internet.

The main body of this thesis is structured into two parts.

In Part I., we study the impact of different factors on the performance of intradomain traffic engineering/ QoS routing algorithms and thus the scenarios in which traffic engineering/ QoS routing can help efficiently use the network resources. We cover the factors such as the accuracy of link state information known by the path computation entity, the network load, the underlying network topology, the provisioning of network resources, etc.

Chapter 2. investigates the impact of different link-state update strategies on the performance of traffic engineering/ QoS routing algorithms. We re-examine whether the gain in network performance, which is expected under the deployment of frequent link-state updates, will outweigh its investment and complexity costs. Chapter 3. first categorizes the different factors that may affect the behavior of traffic engineering algorithms. And it further identifies the factors that determine the relevance of on-line traffic engineering algorithms by studying the behavior of on-line traffic engineering algorithms, using a set of real and synthetic traffic demands and backbone network topologies.

Part II. begins with some background information on interdomain routing. We then study how ISPs carry out their interdomain traffic engineering in reality, and also investigate the possibility of having interdomain QoS.

Chapter 4. introduces how interdomain routing works, including the business relationships coupled with the connections between Autonomous Systems (ASs), the routing protocol, and some general rules that constrain route advertisement and selection. Further, we briefly explain how interdomain traffic engineering can be done by tuning parameters involved in route advertisement and selection. Chapter 5. aims to gain a better understanding about how interdomain routing is done by different ASs via analyzing the publicly available routing data. We provide an AS-level topology model that enables the propagation of all paths observed in the routing data, and take the next step of looking at the granularity at which routing policies are used and thus should be modeled. Our work provides necessary components for a model, which can reproduce the interdomain routing done by each AS as seen from the observations. Chapter 6. explains the need of having interdomain QoS. Pushing QoS across interdomain boundaries appears to be the bottleneck to provide end-to-end QoS guarantees across domains. We discuss the possibility of having interdomain QoS by 1) introducing the components that are evolved in the establishment of interdomain Label Switched Paths (LSPs) with QoS guarantees, i.e. interdomain routing, LSP signaling, and path computation techniques; and 2) putting these components together and identifying the missing functionalities.

Author: Bingjie FU

# Acknowledgements

It is a special experience to work towards a Ph.D. degree. This thesis presents partially the work I have done in four years as a Ph.D. candidate in the Network Architectures and Services (NAS) group at Delft University of Technology. I am grateful to all the people who ever interacted with me in this period of my life, and all the organizations that were involved in making my stay in the Netherlands a reality. Here, I would like to express my special gratitude to the following people.

I would like to thank my promotor Prof. Piet Van Mieghem, who gave me the opportunity to come to the Netherlands and work on the STW project "Network dynamics and QoS" in NAS group. His enthusiasm in research has greatly influenced me. I benefitted a lot from his guidance.

I would like to thank all members in my dissertation committee for their time and effort spent on my thesis and all the interesting conversations we had. It is a great honor for me to have them involved in this thesis. I am indebted to Dr. Heijenk, whose comments and suggestions helped a lot in improving the quality of this thesis. My special thanks go to Professor Bonaventure and Professor Domingo-Pascual for their willingness to travel to Delft from Belgium and Spain respectively.

I would like to acknowledge Dr. Steve Uhlig for his guidance and patience with me. I benefitted a lot from the many insightful discussions with him, and enjoyed working on topics stimulated by these discussions.

I feel fortunate to know and collaborate with the following researchers. My thanks go to Wolfgang Mühlbauer and Olaf Maennel from T-labs TUBerlin, Cristel Pelsser from NTT Japan, and Mickael Meulle from France Telecom, for their comments, for the discussions we had and for the work we have done together.

I am grateful to Dr. Fernando Kuipers for his participation in the initiation of the project that funded me, for his guidance and patience. My gratitude goes to Prof. Ariel Orda from Technion Israel for his comments and patience during our discussions when he was visiting NAS group. My gratitude also goes to Prof. Anja Feldmann from T-labs TUBerlin for her comments on some of my work.

I enjoyed the life as a Ph.D. candidate in the 19th floor of the EWI building at TUDelft, and I would like to thank the NASers and the WMCers for their presence. My thanks go to Laura Bauman, Marjon Verkaik-Vonk and Wendy Murtinu-van Schagen for their help in all

kinds of documents and procedures.

Finally, I would like to thank my parents for their endless love and support; my younger sister for always having interesting stories from her school; the family of my god parents for their love; and my boyfriend for his care and support.

# Curriculum Vitae

Bingjie Fu was born on December 20th 1981, in Daqing, Heilongjiang Province, China. She graduated with a Bachelor of Science (B.Sc.) degree in July 2003 from the Electronic Information Engineering Department of Tsinghua University, Beijing, China. In October 2004, she finished her work as a Master of Philosophy (M.Phil.) candidate, and in 2005, she received her M.Phil. degree from the Electrical Engineering Department of University of Bath, Bath, UK. In December 2004, she came to the Netherlands and started her Ph.D. work at Delft University of Technology in the Network Architectures and Services (NAS) group headed by Prof.dr.ir. P.F.A. Van Mieghem.

List of publications:

- Mühlbauer, W., S. Uhlig, B. Fu, M. Meulle, and O. Maennel, 2007, "*In search for an appropriate granularity to model routing policies*", in Proc. of the ACM SIGCOMM conference, Kyoto, Japan, August.

- Fu, B., F.A. Kuipers, and P. Van Mieghem, 2008, "*To Update Network State or Not?*", in Proc. of QoS-IP 2008, the Fourth International Workshop on QoS in Multiservice IP Networks, Venezia, Italy, February 13-15.

- Kleiberg, T., B. Fu, F.A. Kuipers, S. Avallone, B. Quoitin, and P. Van Mieghem, 2008, "*DeSiNe: a flow-level QoS simulator of Networks*", in Proc. of the first International Workshop on the Evaluation of Quality of Service through Simulation in the Future Internet (QoSim), Marseille, France, March.

- Fu, B. and S. Uhlig, 2008, "*On the relevance of on-line traffic engineering*", in Proc. of the 19th ITC Specialist Seminar on Network Usage and Traffic, Berlin, Germany, October.

- Fu, B., C. Pelsser, and S. Uhlig, 2008, "*Pushing QoS across interdomain boundaries*", Chapter 6 of ≪End-to-End Quality of Service Engineering in Next Generation Heterogenous Networks≫, jointly published by ISTE and Wiley & Sons, November.

- Uhlig, S. and B. Fu, 2009, "*Capturing Internet traffic dynamics through graph distances*", in Proc. of the first International Conference on Complex Sciences: Theory and Applications (Complex'2009), Shangai, China, February.

Other publications:

- Fu, B. and S. Uhlig, 2007, "*Establishment of Inter-domain LSPs with end-to-end QoS constraints*", SIREN 2007 poster session, Delft, the Netherlands.
- Fu, B., S. Uhlig and A. Orda, 2008, "*Internet evolution: an optimization problem or a network game?*", SIREN 2008 poster session, Amsterdam, the Netherlands.