Service level agreement trading for the differentiated services architectures

Service Level Agreement Trading for the Differentiated Services Architecture

George Fankhauser David Schweikert Bernhard Plattner
{gfa,dws,plattner}@tik.ee.ethz.ch
Computer Engineering and Networks Lab
Swiss Federal Institute of Technology, Zürich, Switzerland

Abstract

Bandwidth brokers as proposed in the diffserv framework build on the process of setting up Service Level Agreements (SLA). This is a very static procedure, usually performed manually and based only on a simple description of the SLA. However, diffserv is suited to handle a more dynamic environment and to provide more than connectivity with its service classes. Our proposal enhances bandwidth brokers by including more significant information in SLAs such as the flow's destination network and pricing. In addition, we present a new per-hop behavior that supports flexible allocation of code points which is important to our approach. The system is targeted at large ISPs with good inter-connectivity. We implemented it on top of a flow-based simulation engine. The evaluation is based on parts of the current AS topology and characteristics of aggregated traffic. The results show an improvement of network utilization by up to 40% over a traditional, shortest-path routed inter-domain network for a wide range of network and traffic parameters.

Keywords: Network architecture, differentiated services, service level agreement, pricing, trading, inter-domain QoS routing.

1 Introduction

Internet Service Providers (ISP) face many challenges when they interconnect with each other. In essence, the parties have to come up with a contract, or a Service Level Agreement (SLA) that specifies what service is performed during which period at what cost. Although this principle is simple, the setup of SLAs is not straight forward and in today's Internet it is limited by a range of issues:

- First, SLAs are set up manually; this is time consuming and error-prone. The time-scale of possible SLA changes is therefore limited to several days or even weeks.
- The traffic itself is not characterized other parameters than by terms of bandwidth. Different service classes or levels are usually not part of the contract. This corresponds to the best-effort service provided today but it is not adequate for future network services.
- SLAs are coarse-grained: ISPs set up contracts for transit traffic or local traffic exchange only (peering). The choice here is basically the neighbor's network or the whole world and nothing in between.
- Setting up multiple peers to enhance reliability or to exploit load balancing is difficult using BGP and RPSL only. This is reflected in the many detours found in the Internet. See [25] for examples.
- Finally, prices refer to this static environment. Today, it is difficult for ISPs to react to changes of the market. Many ISPs have found that even simple peering agreements without financial compensation do not work.

Basically, there are two kinds of Bandwidth Brokers (BB):

1. *Centralized agents* that trade raw bandwidth between any defined end points (e.g. services as provided by Ratexchange and other companies)
2. *Decentralized agents* that exchange information about bandwidth allocation between neighboring networks associated with these agents.

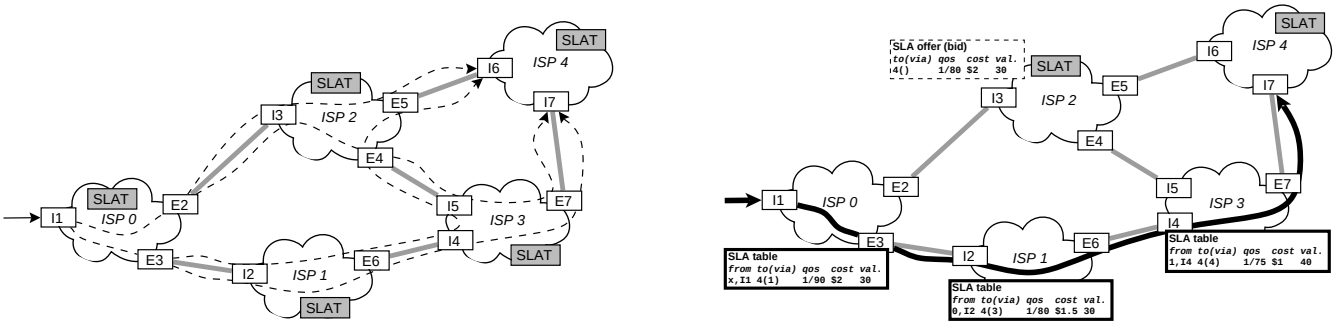
Based on the second kind of BBs we propose SLA traders (SLAT), an approach that addresses above listed issues by the integration of resource allocation, path selection and pricing into the agent's trading process.

The proposal to automate the task of SLA negotiation has gained popularity in the framework of diffserv (DS) [1]. Assuming that both a basic contract and physical connectivity exist between two ISPs, their SLA traders set up new bilateral agreements as demand and load of the networks change. They can either initiate a new SLA or react to a peer's request.

Introducing dynamic SLAs opens the possibility to include more precise traffic descriptions. In a simple system, *per-hop behaviors* (PHB) can be used to select service levels. Besides standardized PHBs, any QoS metrics may be part of an SLA, for instance, burstiness may be specified and used when admitting a neighbor's traffic to a network.

At the same time we should also specify the traffic's destination to address the problem of coarse granularity. According to this, SLA traders may choose to exploit alternate paths at the inter-AS level, balance load between several peers and improve operational reliability[1]. As RFC 2386

[1] A DS domain is "a contiguous set of nodes that operate with a common set of service provisioning policies and PHB definitions." while an autonomous



(a) Alternative paths from *ISP* 0 to *ISP* 4 (dashed lines).

(b) Selected path (bold line) with SLAs.

Figure 1: Example of interconnected ISP networks using dynamic SLAs.

states, QoS routing should not only happen inside but also between ASes. The objective is to "encourage simple, consistent and stable interactions between ASes..." [9].

When we take the SLA trader's view, all the above enhancements provide information on service levels and routing but without knowing the cost of the contract they cannot compare nor trade SLAs. Attaching prices to offered SLAs during the negotiation process gives ISPs a powerful tool to evaluate the potential benefits of new SLAs.

Based on these observations and approaches we propose a new, integrated method for ISP interaction by using SLAs as a central element carrying *service provisioning, routing and pricing information*. To achieve this tight integration we make use of several new architectures and techniques:

- The *DS architecture* is a simple and effective network resource provisioning method usually applied to aggregated traffic with only a few service levels. It uses header marking to classify IP packets based on a user's or provider's profile.
- Between providers, SLAs serve as contracts and specify the profile for aggregated flows that is needed to decide how to mark packets. *SLA traders* are the concept to exchange and negotiate this information. Pricing information is passed along with the SLA to peers which gives providers an immediate feedback how expensive a service is.
- Including information about the domain-path into SLAs leads to a new technique that could best be described as *inter-domain QoS routing*.

In this work we focus on the interaction between DS domains. Resource allocation inside each domain is and has been a fertile field of research and is not treated in this paper.

systems (AS) is commonly defined as "group of IP networks with common routing policy". For readability we will treat the terms "DS domain" and "autonomous system" interchangeably in the following.

To achieve an end-to-end architecture that allows to extend this market-based approach among ISPs to end-users, individual profiles and contracts need to be aggregated to serve as input for SLA traders. Such an approach is proposed in [11]. To keep this step as simple as possible, important results from user behavior surveys have to be considered [6].

In Section 2 we define the SLA trading concept and describe how it fits into the DS architecture. Section 3 and Section 4 explain how traders work and how they are implemented. Using a comprehensive simulation framework we provide quantitative results on the network utilization for AS topologies with self-similar traffic in Section 5. Section 6 sums up related work on BBs and in Section 7 we conclude and discuss future work.

2 The Concept of SLA Trading

First we give a simple example of SLA trading. Assume that an *ISP* 0 has a quantity of aggregated traffic to send to *ISP* 4 (see Fig. 1(a)). Due to the meshed topology of the network multiple paths can be found to reach a destination network. As we can see, some ISPs find themselves in competition to others. In our example, four possible paths lead from *ISP* 0 to *ISP* 4. Consider the service with destination *ISP* 4: *ISP* 3 will receive an offer for the service from *ISP* 2, but since it can go there directly, will probably refuse it (unless it's connection to *ISP* 4 doesn't have enough capacity). Say *ISP* 3 can go to *ISP* 4 with a bandwidth 1, delay 75 and it costs $1. *ISP* 1 could receive an offer from *ISP* 3 with a higher delay, same or lower bandwidth and higher price. *ISP* 1 and *ISP* 2 could then make proposals to *ISP* 0, which will decide which one to buy. In the example Fig. 1(b) *ISP* 0 finally decides to buy the cheaper service from *ISP* 1.

From this small example involving simple QoS metrics, we see that bilateral agreements in form of SLAs build up in a nested manner providing finally an end-to-end service. Cost and delay increase at each ISP (additive metric) along the path while the bandwidth metric is concave and stays at its

minimum. As [22] states, "...[the] observation [is] that multilateral agreements rarely work...". Of course, the advantage of bilateral agreements comes at the expense of a possible service setup delay. However, we can avoid such delays through clever and foresighted contracting (cf. Section 3). Furthermore, we argue that SLA trading happens at a medium time scale (several minutes to hours) and operates on jumbo flows[2].

Restricting the routing, provisioning, measurement and pricing activities to the AS-level solves many scaling problems. The Internet has about 6'000 ASes and there are about ten times as many routed IP networks and 10'000 times as many hosts. Dividing further by applications and traffic source leads to an explosion of micro-flows at the backbone [23].

The *definition of an SLA* provides a common language for heterogeneous trading systems. [1] defines an SLA as "A service contract between a customer and a service provider that specifies the forwarding service a customer should receive. A customer may be a user organization (source domain) or another DS domain (upstream domain). An SLA may include traffic conditioning rules [...]".

SLA trading protocols and the traders itself may change from location to location. In our scenario, SLAs include the destination of the jumbo flow to ensure end-to-end service. However, in a relaxed form, SLAs may also describe services that do not have an end-to-end significance (e.g. the specify the destination as a wildcard operator). In detail, we define SLAs at each ISP by the following parameters:

- A *traffic description*. This includes support for defined PHBs as well as a QoS-vector (e.g. bandwidth and delay) for a specific traffic description. Furthermore, information about traffic conditioning may be included. Using a specific parameters instead of a PHB has the advantage of being a universal metric understood by all ISPs. It is their obligation to map service requests and offers existing PHB in their respective domains. In Section 4 we discuss how a jumbo flow can be mapped to an experimental PHBs supporting bandwidth and delay.
- A *geographical scope* from the ISP's network to some other destination network. We require the scope to be explicit which means that the traffic's destination does not exceed the defined scope.
- *Duration* of the agreement. All SLAs expire after an interval specified in the contract.
- *Cost* of the agreement. SLAs are always associated with a price. Local pricing methods and business strategies are used to calculate prices for new offers [3].

2.1 SLA Trading Optimization Objectives

SLA trading is performed by SLA traders situated in the ISP's DS domain. For the time being we assume traders to be centralized for each AS. SLA traders make local decisions about what services are provided to which peers. Such decisions are made spontaneously or they are the reaction to an external event.

Initially, SLA traders at ISP i may offer services to any peer ISP j only (one inter-domain hop). For such SLAs a price $p_i > 0$ is calculated [4].

Once offers from other partners are received and accepted as an SLA, an ISP may build new services upon existing ones. The price for such a service is the sum of the SLA price offered by the peer plus the cost of the ISP's own resource. Or, if all the nesting is uncoiled, the sum of all local prices set by all ISPs involved.

Each time an SLA trader wants to construct a new service it may compare offers made by all of its peers. Usually the best offer, with respect to fitness of the service and price will be taken. However, this is not a necessity: by adding policy mechanisms to peers' offers, our purely market-based approach can be distorted by regulation. In general, only policies about the peer ISPs are expressible[5].

We model the inter-domain network by a capacitated graph $G(N,L,\vec{Q}), L \subset N \times N$ with ISPs $i \in N$, links $l(i,j) \in L$, $i \neq j$, between any ISP i,j, and $\vec{Q}(l)$ having bandwidth and delay properties $< B_l, D_l >$. S is the set of all possible source and destination pairs $s(i,j) \in S$ where $L \subseteq S$. If $s \in L \mapsto s(i,j) = l(i,j)$. However if a *path* $s \notin L$ is selected it is mapped onto

$$\begin{aligned} s(i,j,\vec{k}) \mapsto & < l(i,k_0), l(k_0,k_1) \ldots l(k_{K-1},j) > \\ & \min(\forall B(l(i,j))) \geq B_s, i,j \in k_0, \ldots k_{K-1} \\ & \sum(D(l(i,j))) \leq D_s, i,j \in k_0, \ldots k_{K-1}. \end{aligned} \tag{1}$$

Users define a valuation $u_q(i,j,\vec{q})$ for each traffic type, according to $\vec{Q}(s(i,j,\vec{k}))$ (utility function). For best-effort traffic, where $\vec{Q}(s(i,j,\vec{k}))$ is unspecified, $u_q = const$. For each source/destination pair and selected path $s(i,j,\vec{k})$ traffic flows $f_{i,j,\vec{k}}$ give a utility of $u = u_q(s(i,j,\vec{k}))$. Now we introduce a cost function per link and path which is defined by each ISP locally:

$$\begin{aligned} c_{l(i,j)} &= c(l(i,j),\vec{q}) \\ c_{s(i,j),k} &= \sum_k c(l_k(i,j),\vec{q}) \end{aligned} \tag{2}$$

which yields the total cost for all flows in the network:

$$\sum_k \left(c(s(i,j),\vec{k}) \times f_{i,j,\vec{k}} \right). \tag{3}$$

In a network with *no overhead cost* for the exchange of SLA

[2] Jumbo flows are defined between two ASes [10].

[3] For the sake of simplicity we assume global currency.

[4] It is the ISP's business whether to employ a model that always covers immediately its own cost or to decide to implement a long-term strategy where, e.g. discounting may be used.

[5] One could also think about an extension to global policies by, e.g. excluding providers from the path of nested SLAs. Policy attributes would then become part of the SLA itself and could be propagated to the next provider.

messages, total user benefit would be total utility minus cost of all flows:

$$\sum_k u_q(s(i,j,\vec{k})) - \sum_k c(l(i,j),\vec{q}) \times f_{i,j,\vec{k}}. \tag{4}$$

There exist approximations for the global maximum given by Eq. 3, however, with the assumption that (i) $\vec{q}$ can be folded into a single metric (e.g. effective bandwidth) and (ii) signalling messages are neglected [17].

But in reality the overhead cost $c_{m(i,j)}$ associated with every exchange of every SLA message is significant. We restrict such message exchanges $m(i,j) \in L, i \neq j$ to avoid bootstrap problems (i.e. m is exchanged point-to-point). Since the exchange of SLAs is not directly related to a flow, maximization of network efficiency is only defined for all flows and messages concurrently. Considering this overhead, the objective becomes:

$$\max_{f(\vec{k}),u_q,c_m} \left(\sum_k u_q(s(i,j,\vec{k})) - \sum_k \left(c(l(i,j),\vec{q}) \times f_{i,j,\vec{k}} \right) - \sum_{i,j} c_{m(i,j)} \right). \tag{5}$$

$$f_{i,j,\vec{k}} \rightarrow \sum_k c_{m(i,j)} \cdot p_{\vec{k}\times\vec{k}} \tag{6}$$

indicates the cost of SLA messages triggered by flow $f_{i,j,\vec{k}}$. $p_{\vec{k}\times\vec{k}}$ is the probability for a message exchange at link k. Note that $f_{\vec{k}}$ can trigger a tree of messages with $p_{\vec{k}\times\vec{k}}$ that could potentially flood the network. Let $P_{s,d}$ be the probability matrix $s,d \in N \times N$ that defines for all directed links the probability that an SLA message is forwarded. $P_{s,d}$ is the difference of a "propagation" and a "cache" component $P_{s,d} - C_{s,d}$. The propagation contribution has a center around s and a rim of zeros:

$$P_{s,d} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \vdots & \iddots & 0 \\ 0 & \cdots & p_{s,,} & \cdots & 0 \\ 0 & \iddots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{7}$$

Probabilities drop with increasing AS distance since traders can evaluate the AS hop count of SLA requests (Section 2.3). $C_{s,d}$ defines the probabilities that an SLA with egress node/link j for a destination d exists at an AS. While $P_{s,d}$ is tractable in the sense that we may bound it by hop count (probabilistic bound)[6] the term $C_{s,d}$ depends heavily on spatial distribution of all active flows. Adding flows to paths that are already used increases the chance to find existing SLAs. We will see in Section 5 how SLA trading can benefit from non-uniform distributions of inter-AS flows.

We use the analytical model of the *global* objective to derive local goals to achieve a more precise and configurable model that includes spatial and temporal traffic behavior. Each player's objective has a more specific, local form for an ISP at an AS a with egress node j. The SLA message overhead $c_{m(a,j_n)}$ is now under control of a single AS. The utility $u_q(s(a,j,\vec{k}))$ is modified by applying a price function $p_j(q,s)$ which gives the provider's profit maximum as follows:

$$\max_{f(\vec{k}),p_{q,s},c_m} \left(\sum_j \left(p_j(q,s) - c(l(a,j),\vec{q}) \right) \times f_{a,j,\vec{k}} - \sum_{j_n} c_{m(a,j_n)} \right), \quad j \neq a, j_n \in l(a,j_n), j \in \vec{k}. \tag{8}$$

We suppose there are only ISPs in our network. Individual user demand is collected at each AS and the aggregated traffic per service class $\vec{q}$ and destination j originates as $f_{a,j,\vec{k}}$. The similarity of the global and local objective is due to the nested structure of SLAs: we expect most of the flows $f_{a,j,\vec{k}}$ not to be originated from a but expressed as a "sub-SLA" that was sold to an ISP b for a path $s(i,j,\vec{k}), \vec{k} =< i,\ldots b,a,\ldots j >$. If we set $p_{q,s} \equiv u_q(s(a,j,\vec{k}))$ the sum of local maxima approaches the global objective. But $p \not\equiv u$ since u is not revealed to ISPs in practice. It also depends heavily on the application type and its adaptiveness [2]. What we can require is $p' \geq 0, \forall q,s \wedge u' \geq 0, \forall i,j,\vec{k}$. This direct and non-uniform consideration of local maximization leads to a deviation from the global goal. However, above requirement still ensures convergence.

Implementation of the local objective is still far from being trivial. Practical implications for simple trade-off rules can be stated by (i) The size and duration of offered and bought SLAs involves a trade-off between messaging overhead and undesired capacity that might not be resold. (ii) AS distance is a measure that is used to favor short AS-paths. (iii) $(p-c) \times f < 0$ if long-term strategies are followed (high risk-awareness).

2.2 Direction of Service

In DS, it is the sender that marks the packets according to the desired service. The type of service has a direct relationship to the payment for this service. Therefore, the only possible payment scheme is "sender-pays" and the only sensible SLA offers are proposals of forwarding packets from the peer ISP to a destination. Other payment schemes have to be implemented on a higher layer.

For example, in a video distribution scenario, the video sender could charge the customers for different qualities of delivery and then mark the packets accordingly. But on the SLA level this distributor would still have to pay its peer ISPs [3].

[6] We follow the design that gives the trader the freedom to forward requests or not. A strict bound could be enforced by changing the AS hop count field to a TTL style field that is initialized by the original source.

With SLA trading, the money injected into the network will be distributed among all involved ISPs. This is an incentive for the providers to improve their services.

2.3 A Simple Protocol for SLA Trading

Signaling *demand and supply* between ISPs needs an appropriate protocol to transport SLA messages. Many alternatives are proposed, under development or already available: new BGP attributes, the Internet Open Trading Protocol (IOTP), RSVP extensions and others[7]. However, they either fit into a specialized environment or are quite general and therefore too complex. Because SLA trading is done only between peers the protocol can be kept very simple. For this reason we defined a minimalist peer-to-peer protocol called *SLA trading protocol (SLATP)*. Each SLATP protocol data unit consists of a type (ask, bid, accept, reject, confirm), an AS hop count field and an SLA as defined in Section 2. Fig. 2 shows the message sequence chart. SLATP may be implemented on top of any datagram service.[8]

By sending *ask messages*, *ISP n* may request the service of *ISP m*. The main reason for this message is to speed up SLA setup and therefore convergence. In a perfect, coordinated world where each ISP would advertise all its available resources by using *bid messages* this might not be needed. Bids are mandatory since they initiate SLAs. SLA bids may be accepted by sending an *accept message*. Upon an accept the bidding party will send a *confirm message* to seal the contract or send a *reject message* to cancel the offer (bid has expired or for error reasons).

We should also mention here, that as long as ISPs adhere to the basic SLA structure they can deploy a trading protocol of their own choice (upon mutual agreement). In Appendix B we give a formal description of SLATP.

Implementing a protocol for SLA exchange raises two *security and fairness issues*. First, an SLA represents a contract of some monetary value. Therefore *non-repudiation must be ensured*. Using a public key infrastructure to double-sign every SLA exchange solves the problem. Second, when crossing domain boundaries we must be prepared that traffic is eavesdropped or intercepted. Encryption of SLA messages is important to *prevent market manipulations* (e.g. one ISP is intercepting traffic of its competitors to "optimize" his own bids). Although we didn't implement security features in SLATP they can be easily added.

2.4 SLA Enforcement

There should be also an incentive for a provider to buy enough SLAs, so that the incoming in-profile traffic will remain in-profile on the next provider. Or in other words, if a sold SLA depends on other bought SLAs the first one must not exceed the second. Unfortunately this is not straightforward to check, because it is impossible for a customer to find out whether his packets were remarked from in-profile to out-of-profile. One approach is to use "quality probes", a ping-like protocol, that returns information on the delivered QoS. This is, however, out of scope for this paper and an issue of further research. Basically, the protocol should tell us at which ISP undesired remarking happens.

[7] Other alternatives are the BB Transfer Protocol, DIAMETER, COPS, SNMP.

[8] For large ASes communication between ingress / egress nodes and the SLA trader may pose a scaling problem. As for IBGP, known techniques such as route-reflectors and confederations (see RFC 1965 / 1966) could be applied also to an "interior SLATP".

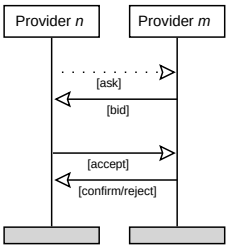


Figure 2: SLA trading protocol sequence chart.

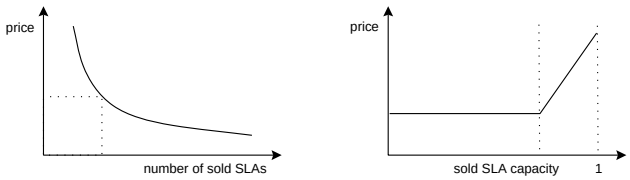


Figure 3: Demand curve and residual bandwidth price function.

3 SLA Traders

Every SLA trader can buy and sell SLAs. Established contracts are stored in a table. Based on that information, it will set up DS classifiers / markers at ingress nodes and remarkers at egress nodes (see section 4 for a more detailed discussion of that subject). Routing and switching traffic between ingress and egress node according to the SLAs is outside the scope of this paper and delegated to an appropriate label switching architecture. SLA traders will also remove expired entries from that table and provision their internal networks accordingly.

3.1 Provisioning of Resources Across Domains

There are two types of resources: owned resources of the ISP and SLAs bought from peer ISPs.

Owned resources are the "raw material" used to construct services. These resources are static or dynamic links from an ISP to another ISP in form of leased lines, ATM channels, optical fibers, etc., attached to routers of the ISP. In this paper we will make two assumptions about owned links: (i) Every link is *unidirectional*; and (ii) every link belongs to the ISP from which the link originates (i.e. the traffic source for

that link). Bidirectional links are modeled as two unidirectional links. The second assumption addresses the sender-based packet marking of the DS architecture (see Section 2.2). An SLA trader has to decide how to provision it's external resources such as to make the most money possible by reselling them to other ISPs. In other words, it will have to decide which SLAs to buy by trying to improve the value of its offer while keeping its cost as low as possible.

3.2 Bid Generation and Pricing

SLA traders make bids, based on available resources, to other ISPs. More fine-grained SLA bids will be more probable of being accepted because of the broader service palette, but will also involve more protocol overhead for communicating them. Since a link is owned by the source ISP, frequent advertising of many bids will eat up part of its own link resources, leaving less space for services to sell. This is an optimization problem, which will drive the development of new and efficient trading protocols and optimal customized offers for each peers matching the supply with their demand[9].

The goal of commercial ISPs is to maximize profit which is reflected in how SLATs will behave. The price of the sold services will be calculated to cover the costs of the ISP *and* to make profit.

Since SLA Trading involves *competition*, the lower the price for a service, the more probable it is that an ISP will purchase it. An ISP will therefore try to *optimize the price* of the services to get the most possible profit. In a friction-free, single-good economic system where the demand curve is known, this would mean taking the price which does maximize the rectangular area shown in Fig. 3(a). The economic system of networking services is, however, a multiple-good market and possibly not fair (two customers can be sold the same service for different prices), which makes this complex optimization problem better solvable with iterative algorithms.

The pricing strategy used in SLA trading is *residual bandwidth pricing*. The price function is shown in Fig. 3(b). Also known from other approaches to QoS routing [31] such a function ensures that prices get higher the more the SLA or link resources are used. The network operator adapts the base price and slope of this function to the point where he maximizes the price volume product of all of its resources (Eq. 8). Residual bandwidth serves as a primary pricing instrument while other metrics, such as delay, are checked as a constraint. *Competition* is of primary importance to the SLA Trading framework. If a provider has a much faster and less expensive link to a destination, it is better for global efficiency that its services are being preferred over the others.

Consider the following example: All ISPs of Fig. 4(c) are assumed to be configured equally (same link speed, same price function). The shortest path from *ISP*0 to *ISP*4 is therefore also the cheapest (2 hops or $2 \cdot p$). When demand increases the price for the shortest path increases also until the initially

[9]This is a kind of *self-regulated signaling*.

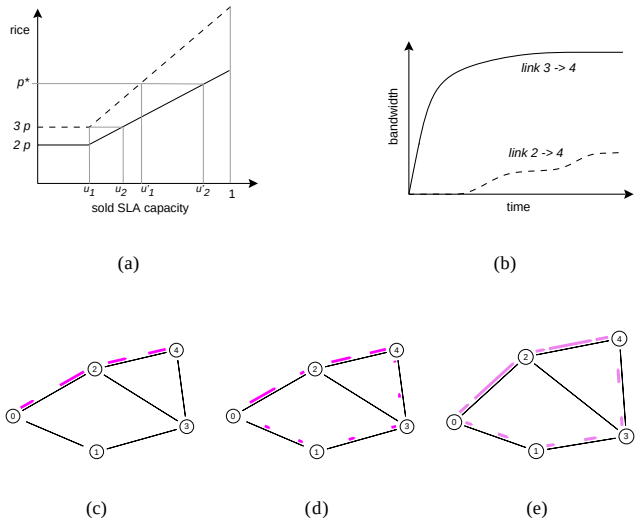


Figure 4: Cumulative price (a) and bandwidth (b) for paths $0 \rightarrow 1 \rightarrow 3 \rightarrow 4$ and $0 \rightarrow 2 \rightarrow 4$; (c) shows initial path selection; (d) and (e) show subsequent flow and path changes.

unattractive longer path (3 hops or $3 \cdot p$) becomes competitive and is finally selected as an alternate route. Fig. 4(c) ff. show the selected paths, price and bandwidth of such a simple setup. Depending on the demand for destination *ISP*4 a price p^* is minimal for the desired flow of bandwidth $u'_1 + u'_2$ (Fig. 4(a)). What is neglected in theory is a certain granularity of SLAs. Implementations exhibit quantization of demand (shown as bandwidth levels in Fig. 4(b)).

3.3 Trading Algorithms

We call the algorithm responsible for the determination of what resources are needed the *provisioning algorithm*. A *passive provisioning* algorithm does wait for requests (in form of asks messages, see Section 2.3) from its customers to select which resources to buy. An *active provisioning* algorithm tries to forecast future needs. It will then buy resources in advance, before they become scarce. Buying in advanced may be based on statistical information (e.g. previous weeks usage by time of day) or on trend analysis.

Once an SLA trader knows it needs to buy some resource from one of its peers, it will have to select one of the bids and buy it. The selection of the bid is made based on the bid's value for the SLA trader and its price. For bids of equal value, if no special policy is applied, the bid with the lower price will be selected.

The SLA trader will also have to evaluate if the selected bid is worth buying using a *profitability analysis* algorithm. This algorithm does evaluate if by buying that bid, money will be made through the selling of derived services. It is this algorithm which will also ensure that SLA traders won't build service loops. The reasons for loop-freeness are:

- QoS guarantees are defined in the SLA. For example, delay would add indefinitely for each loop, or bandwidth would be exhausted at some point.
- "Stupid" traders which build loops will loose money doing so and will therefore be eliminated from the market.

Consider the forwarding service to *ISP* 0 in Fig. 1(b). Suppose that *ISP* 2 did sell that service to *ISP* 4, which in turn sold it to *ISP* 3. Suppose further that *ISP* 2's resources to *ISP* 0 become scarce and that *ISP* 2 wants to buy more because that service is highly demanded. If *ISP* 2 behaves badly, it could buy further bandwidth for example from *ISP* 3. A loop is the result. However, this is different from an *infinite routing-loop*, it's rather a *limited service-loop*. What are the consequences of these service loops?

- Every ISP other than *ISP* 2 will gain money and the already existing services won't be affected.
- *ISP* 2 will, for each service-loop as described before, rebuy it's own service to *ISP* 3 paying also to every other ISP involved in the service loop. In other words, the QoS guarantees will remain satisfied and *ISP* 2 will go soon out of business because it will repeatedly loose money.

Another aspect that has to be considered when implementing traders, is the aspect of *temporal and spatial fragmentation of bids*. Buying services that do not fit the requirements exactly impose a risk on the ISP (e.g. the needed service is indeed offered by a peer, but for too long or only bulk quantities are available). This aspect is included in the *profitability analysis* algorithm by setting limits to the left-over if bids do not fit exactly current demand. To compare bids a gain function $g\left(\Delta(\vec{I}_w, \vec{B}_w)\right)$ is defined where $\vec{I}_w$ references the ideal SLA that is requested by a peer or is derived from flow measurement of local demand. $\vec{B}_w$ is the bid to compare. $\vec{w}$ is a set of weights that gives a local parameterization of bandwidth, delay, volume, price and AS distance. Setting for example an emphasis on volume gives bids with a matching time bandwidth product a higher chance of getting selected.

The algorithm shown in Fig. 5 is a simplified version of a trading algorithm that includes the basic functions to receive and make bids. On receiving, a profitability check based on the gain function *g* and on expected usage is made prior to accepting it. Expected usage is constant here but might be adjusted by service providers.

Table 1: Parameters and constants.

UPDATE_PERIOD	100 ms .. 10 s
CONNECTIVITY_BW	64 .. 128 kbps
MIN_RBWBW	0.3
rbwbw	current residual bandwidth for each SLA
volume()	volume function for an SLA object (*time* × *bandwidth*)
send_bid()	sends an offered SLA to peer
accept_bid()	sends an accept message
known_dests	reachability list

```
struct Bid {
   as_dest,  // AS destination
   bw,       // bandwidth
   price
}

process trading () {
   while (true) {
      for each d in known_as_dests {
         /* buy bids */
         if bw_to_as(d)<CONNECTIVITY_BW or rbwbw_to_as(d)<MIN_RBWBW {
            bid = [find bid with highest volume/price ratio]
            if bid and is_profitable(bid) accept_bid(bid)
         }
         /* make bids */
         for each n in neighbours {
            if !bid_already_sent(n, d, bw) make_bid(n, d, bw)
         }
      }  sleep(UPDATE_PERIOD)
   }
}

process bid_recv() {
   while (true) {
      wait(bid_received(bid)) {
         if !is_as_dest_known(bid.as_dest) known_as_dests.add(bid.as_dest)
      }
   }
}

boolean is_profitable(bid) {
   expected_volume = volume(bid)*EXPECTED_USAGE;
   expected_income = price(bid.dest,bid.bw) * expected_volume
   if bid.price < expected_income return true
   else return false
}

void make_bid(neighbour, as_dest, bw) {
   bid = new Bid(as_dest, bw)
   bid.price = volume(bid)*price(as_dest, bw)
   send_bid(neighbour, bid)
}
```

Figure 5: A Simple SLA trading algorithm.

Also, connectivity is established without demand from customers to mimic a standard routing protocol that exchanges reachability information at each update period. See Table 1 for sample parameters.

Summarizing from the previous and this section we find the following minimum requirements for SLA traders:

1. $c(l(i,j),\vec{q}) > 0, c_{m(i,j)} > 0$ (no zero cost for crossing a domain, signalling is also associated with a non-zero cost).
2. Traffic scheduled according SLA (either trusted or checked, see also Section 2.4)
3. Common SLA format and semantics
4. Common trading protocol (at least for each pair of peers)
5. Global, common addressing

4 Integration and Implementation of SLAT in diffserv

We developed an experimental version of the SLA trading framework using a simulation environment [27]. We made several abstractions to keep complexity and run-time overhead low:

1. Packets belonging to the same source and destination network, and traffic class are modeled as a jumbo flow, and
2. DS domains were abstracted into nodes. As a consequence, ingress and egress nodes of DS domains become links in the abstraction and interior nodes disappear.

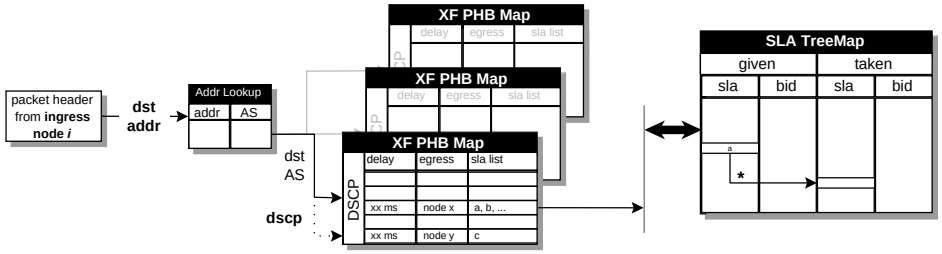


Figure 6: Local per-destination and per-ingress node PHB tables; global SLA table.

Originally, 3 bits of the IPv4 header's type-of-service byte were used as "IP precedence field". Similarly, in IPv6 a "traffic class octet" was defined. In DS this octet was redefined and called *DS field*. It is split into 6 bits forming a code space of DSCPs and 2 bits which are currently unused [21].

Thus, the DS field has 64 entries for DSCPs. From Table 2 we can see that the code space is halved into a standard and experimental part. The standard pool is being filled with standardized PHBs as well as with a default PHB for best effort traffic and a PHB for prioritized routing and network control packets.

As mentioned earlier, SLA trading can be applied to any notion of PHB. However, in our simulation environment we chose to add an experimental PHB (called XF for experimental forwarding) that provides 16 code points with a specified delay, and the egress node. These combined entries are *dynamically allocated* by the SLA trader. A fixed number of DSCPs are reserved and the ones most often asked for are kept in these slots. Thus, the semantics of DSCPs become dynamic and the currently associated QoS features have to be communicated via SLATP to peers. In addition, this scheme is applied to each ingress node locally (Fig. 6 on the left side, local view of ingress node *i*), i.e. incoming packets from different peers having the same DSCP do not necessarily get the same service level.

As a central data structure in each DS domain, a table of "given" and "taken" SLAs and outstanding bids (offers) is maintained. As described in Section 3, this table is the main instrument of the trader process. It contains all the dependencies of sold SLAs on bought SLAs. Owned resources (the links to the peers) are treated like bought SLAs that are always available.

When SLAs are set up, DSCP tables for the particular agreement have to be checked for code space first. It is the traders responsibility to do an economical allocation of this scarce space. For each ingress node / destination pair a possible new combination of delay / egress node may be allocated when a new SLA is agreed upon. Since the DSCPs are a limited resource, the following is done to limit this problem:

- Merging of jumbo flows with same service characteristics and service quantization. Merging can be done, if an SLA uses the same delay / egress node pair for multiple bandwidths, e.g. for 1 Mbps and 4 Mbps ($\rightarrow$ upgrading / extending SLAs). Note that the bandwidth is not defined in the PHB table, it is stored in the larger SLA table.
- Very fine grained delay specs are usually not needed and can be quantized to become "reasonably granular". For the egress node information, a merger is not possible as long as we want to support multiple paths through the DS cloud.
- Finally, the DSCP encodes the next-hop AS (the egress node). For example, a large AS can maintain 8 alternate routes to the same destination AS while supporting 2 delay classes (delay sensitive / don't care). In [12] we give an overview on how AS connectivity is structured. To answer the question how many alternate routes should be supported we look at the subset of class 1 and 2 backbone ASes which appear to be good candidates for re-routing traffic among themselves (about 90 core networks) and sort again by connectivity. Now, the best connected AS has an out-degree of about 50 and the mean connectivity among these networks is 12 (see Fig. 7). Usually only a subset of these alternatives are

Table 2: Overview of DS code point space.

PHB	Code Space	# of points
Std. Pool 1	xxxxx0	32
Best Effort	000000	1
Network control	11x000	2
Assured Forwarding (AF)	001x10,0x1100,0011x0, ...	12
Expedited Forwarding (EF)	101110	1
Exp. Pool 2	xxxx11	16
Exp. Pool 3	xxxx01	16

Table 3: XF PHB code / address space.

Feature	# of bits
DSCP	4
Egress node	x
Delay class	$4-x$
Packet header, SLA table, ingress node	~ 40
Destination AS	16
Bandwidth	~ 10
Ingress nodes	~ 12

used for multi-path selection which justifies the limitation to 8 routes for each AS. [10]

Therefore, XF's dynamic allocation scheme of DSCPs can sustain even large and complex DS domains. In Table 3 the complete code space, including addressing and information stored in the SLA table, is given as a reference. x is the trade-off between alternate paths and delay classes using 16 DSCPs. Our dynamic DSCP allocation approach results in a dynamic "inter-domain label-switching" technique, compared to PHBs with static, global significance.

4.1 Packet Handling at Ingress Nodes

Now that the usage of DSCPs is defined, we look into the basic operations of forwarding packets through DS domains. The two basic data structures used are local PHB maps for the dynamic allocation of code points (per ingress node and destination) and an SLA table (one for each DS domain). As shown in Fig. 6, incoming packets select a PHB table depending on the entry node into the network, then the address is used to find the PHB table for that ingress / destination pair, and finally, the DS field itself is used to select the code point in this PHB table (for the destination AS address selection hashing is used while DSCP tables are short enough for a direct index lookup). Inside the DSCP's table entry, a list of SLA identifiers is kept. The SLAs in this list share all the same ingress, egress, destination address and delay class.

Finding the list of SLAs from the packets destination AS and DSCP information as keys is a Multi-Field classification step (MF). Destination address to AS mapping can be done using traditional routing lookup methods. This conversion step is omitted in our implementation due to the collapsed DS domains.

The list of SLA identifiers contains also entries for the next DSCP used for remarking (the ones that are valid in the next-hop network) and some domain-specific information how to get the packet to the egress node. Since we focus on inter-domain issues, we intentionally do not describe how jumbo flows are switched from ingress to egress node. Many existing solutions exist for this problem, e.g. PASTE [16] could be employed, which in turn is based on MPLS and RSVP. Such solutions have to deal with resource allocation inside the DS domain, explicit routing through the DS cloud, and possible state setup in the interior routers / switches.

4.2 Packet Handling at Egress Nodes

As already mentioned, packets may need to be remarked at the egress node to conform to the SLA definition of the next hop on the path. If the DSCP is neither used nor changed while in transit through the ISP's DS cloud, this step may be performed already at the ingress point. This has the advantage that the MF classification described above does not need to be repeated at egress nodes.

[10] As we immediately see, this is the local choice per AS, whereas the paths crossing n domains may take 8^n ways.

If it must be done at the egress node, the relationship between ingress node i and the flow used to transport these packets to the egress node must be forwarded via DS domain internal signaling.

Basically, we end up with one or possibly many DSCPs according to the SLAs for outgoing traffic. With a single DSCP, all the classified packets are remarked and sent out to the peer. If many contracts were used for same service characteristics and paths but differing in temporal scope and bandwidth, the outgoing packets have to be remarked according to their *share*. Packet scheduling algorithms, for example deficit round robin [29], are a good choice to implement this "fair-share remarking" process.

5 Results

This section shows how SLA trading performs in inter-AS routing and resource allocation. For the evaluation of SLA trading we use an event driven simulator called *flowsim* [27]. DS domains were collapsed to nodes at the inter-domain level (i.e. all nodes inside a DS domain are no longer visible, and ingress and egress routers of the DS domain become links of the collapsed node).

First, we use simple setups to show basic allocation and convergence with different routes and traders. Then we move on to core network AS topologies using traffic models that are typical for this environment.

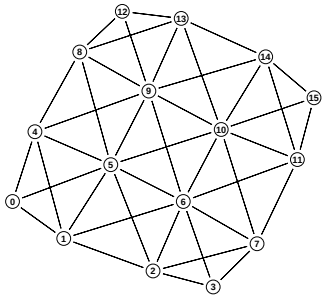
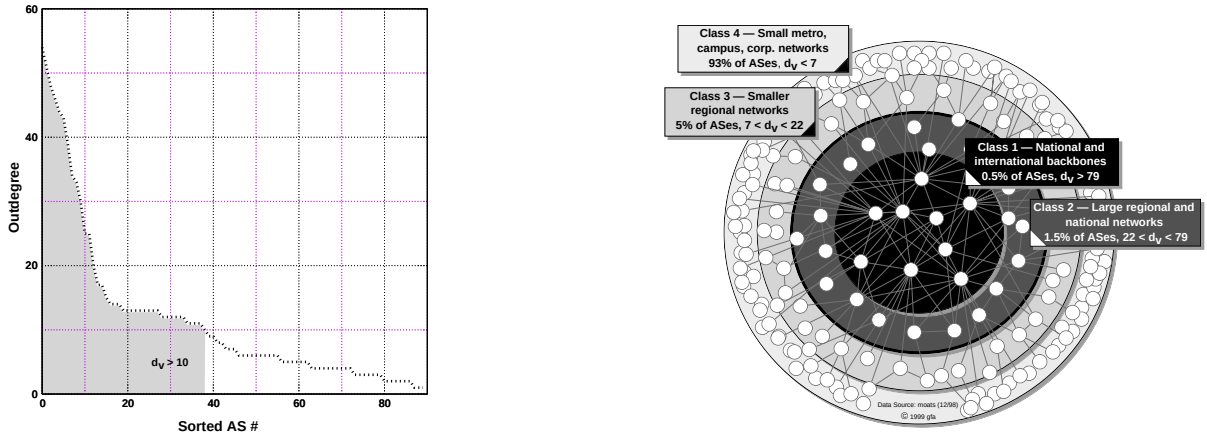


Figure 8: Grid topology.

5.1 Basic Properties of SLA Trading

The first experiment shows the *load balancing* capabilities of SLA traders. It is run on the topology as shown in Fig. 8(a) with all links having 10 Mbps capacity. Non-bursty traffic (aggregated from 500 Poisson distributed calls) at a rate of 30 Mbps of offered load is generated at *ISP* 0 with destination *ISP* 15. In Fig. 9(a) the throughput with SLA trading and DV routing (distance vector) are plotted. While DV is clearly limiting the throughput to the bottleneck link speed of the shortest path (via *ISP* 5 and *ISP* 10), SLA trading needs more time to setup the alternative paths but achieves a much higher throughput by using three paths via ISPs 1 and 4. The 3 paths



(a) (b)

Figure 7: AS distribution and classification.

and the sum of all flows is shown. The profitable traders (Section 3) that were used for SLAT buy bandwidth incrementally and converge after about 7 s in this configuration. The theoretical maximum of 30 Mbps is not achieved due to configurable margins for pre-arranged SLAs to other destinations and for best-effort traffic (about 10% of each link).

Competition among ISPs ensures the selection of the best bids from peers. Observing a single bottle-neck link (10 Mbps) a profitable trader sells SLAs to these three peer ISPs. They are configured with an increasing willingness to pay (WTP). From Fig. 9(b) we can see the throughput each ISP gets. The link is beginning to fill and due to differentiated selection of bids the three ISPs end up with about 5, 3, and 2 Mbps link-share, proportional to their spending.

The next experiment is run on the topology as shown in Fig. 8(a) with all links having 200 Mbps capacity. Non-bursty traffic with 100 Mbps of offered load is generated at *ISP* 0 with destination *ISP* 15. We investigate the effect of *heterogeneity caused by local decisions and strategies* applied by different ISPs. We measure the average throughput that an *ISP* 0 achieves in function of the strategies used for the implementation of *ISP* 5 (the "Black Sheep"). All other traders always employ the *Profitable* strategy. The experiment is repeated with the same setup for each trading strategy. The result (Fig. 9(c)) shows the time each trader needs to adapt to the generated load. The *Profitable* trader (this is also the homogeneous case) is the best implementation (convergence after about 3 s). The *Trendy* trader follows because it makes sensible decisions based on service usage (4 s). A *Greedy* trader makes sub-optimal decisions. It is trying to setup lots of SLAs in advance for destinations that are not used in this setup (5 s). Finally, the *Null* Trader which doesn't buy or sell anything at all requires the setup of alternate routes through ISPs 4 and 1 (convergence without using the path via 5 after about 6 s). See Appendix A.2 for more information on trader types.

5.2 SLA Trading in the Core Network

As we have seen in Section 2, analysis of local objectives of ISPs is hard. Most important, dependencies exist between flows and their handling at each domain. In addition, our goal to give each ISP the choice of local decisions and strategies would complicate an analysis further. Therefore, the simulation approach was extended with statistical results from topology and traffic analysis:

- Subgraphs of recent AS topologies [19] were used for simulation. From class 1 and 2 ASes those with $d_v \geq 10$ were selected (gray shaded area in Fig. 7(a)). A list of these ASes is given in Appendix A.
- Traffic was generated using synthesized self-similar patterns as described in [24]. The Fractional Gaussian Noise (FGN) distribution is used as self-similar processes to model inter-AS flows with high burstiness across time and aggregation scales. The spatial inter-AS traffic distribution is suspected to be *highly non-uniform (hot spots)* [12]. This is supported by a recent study on the distribution of jumbo flows (source AS to destination AS aggregate flow) [10]. Such a jumbo flow model has been used for simulations network that shows such a highly non-uniform locality of flows.

The model is defined by the parameters given in Table 4. The main output parameter is utilization of the network. We load a network configuration until all SLA traders in the system converge to a steady state or, in the case of dynamic setups, an observation period with a steady average is reached. As a comparison, a shortest path, best-effort setup is used. Note that we do not include the added value of bandwidth and delay bounds that is provided by the SLA approach in this comparison.

First, we give a visual representation of a small inter-domain network and show the effectiveness of SLA Trading

using a subset of 8 ASes out of the 30 best-connected networks. 90% of the total of 100 jumbo-flows flows are setup between 10% of the possible source / destination pairs (i.e. $\nu = 0.1$). The other parameters are $V = 0$, $M_{total} = 480$, $F = 50$, $L = 100$. Fig. 10(a) shows the non-uniform distribution of flows when a DV-based shortest path algorithm is used for routing. In Fig. 10(b) the same situation is shown when SLA trading is used. We observe a smoothed out distribution of flows and generally less loaded links. Choosing this setup an increase of total network utilization by 38.9% is achieved. Utilization here is the sum of the net throughput for all flows excluding overhead for DV or SLAT messages (Fig. 10(c))[11].

Usage patters and change in demand cause *spatial traffic variations* on the medium and long time scale [10]. For example, promotions of e-commerce goods, web-casts of popular events etc. control such demand patterns. Fortunately, there is no significant spatial variation on the small time scale (we will see later that this stands in contrast to temporal traffic variations).

Fig. 11 shows the throughput obtained in the 8 AS network (Fig. 10(a)) for different values of ν where 0.0 and 1.0 indicate a total asymmetric load (all flows on one spot) and 0.5 stands for the evenly loaded network. The network was configured with $F = 50$; $M_{total} = 2400$; $L = 10$; $V = 0.0$. This implies that the traffic exceeds the networks capacity. As expected, an asymmetric load is an opportunity for SLA traders to reroute flows. Low values of ν that have been observed in real networks give an improvement of about 45%. At $\nu = 0.3$ the gain over DV is still about 20% (Fig. 11(c)). The uneven steps observed between different spatial distributions reflect the random set selection of flows' start and end points.

In the next setup we investigate the effectiveness of SLA trading for different *network sizes*. Our initial claim was to support the core of providers at higher values of d_v. We simulated this by starting at the center of the core and look at the 6 to 36 best-connected ASes. Note that up to $d_v = 9$ these subnetworks are fully connected. We chose two simulation runs $S_{1,2}$ with the following parameters: $\nu = 0.18$, $L = 10$, $F = \{80, 100\}$, $M_{total} = \{768, 2400\}$, $V = 0.0$.

[11] DV routing is statically configured, i.e. the routing tables are setup until convergence. Later on the full bandwidth can be used by best-effort traffic. The dynamic SLA trading approach produces a considerable signalling overhead that is using bandwidth not available to SLAs.

Table 4: Parameters of the simulation model.

N	# of ASes
F	# of flows
L	Link speed [Mbps]
d_v	Out degree (# connections to other ASes)
ν	Spatial traffic asymmetry: $((1-\nu)\cdot F)$ flows occupy a fixed set of $N \times N \cdot \nu$ paths. Inside a set flows are uniformly distributed.
V	Normalized variance of FGN traffic
H	Hurst parameter of FGN traffic (long range dependency)
M	Mean traffic rate [Mbps]
TS	Time scale

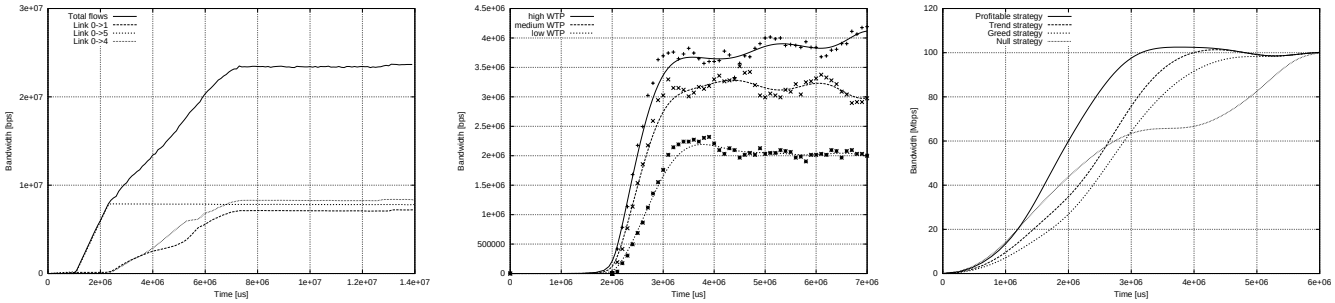
Due to the high traffic load applied the very small network of size 6 is not able to the improve utilization much due to its capacity restriction. Increasing the size shows the desired effect of load balancing very well (Fig. 12). At larger configurations the gain between SLA-based and DV-based routing stays at a high level. For a network size beyond the measured range we expect the gain to drop off eventually. As the network size increases further, connectivity decreases and the average hop-count of each flow increases which makes rerouting and adaptation more difficult. Results are limited to networks consisting of about 40 domains due to the CPU and memory constraints of the simulation environment[12]. Again we notice an uneven stepping between the different network sizes. This is due to the nature of the AS topology itself *and* the flow placement.

To explore the effect of *temporal traffic variance* we observed the trading behavior on a simple triangle topology with single jumbo flow on a direct and a 2-hop alternate path. Link speed is 1 Mbps. The traffic's time scale is 1 s and H is 0.7. While the closed-loop control mechanism of TCP causes variations on the small time scale (order of seconds) usage patters and change in demand cause variations on the medium (minutes) and long time scale (e.g. caused by time of day patterns, order of hours).

Although SLA traders are targeted to the medium and long time scales we chose $TS = 1$ in this experiment to address a lower limit. In Fig. 13(a) bursts are low enough for the trader processes to follow the demand. In Fig. 13(b) gives an example for very high burstiness. Traders are not quick enough to follow the traffic pattern. In addition exceed the physical connection speed of 2 Mbps (2 paths with 1 Mbps each). Fig. 13(c) gives an overview of the average throughput achieved for $V = 0 \ldots 0.5$ and two traffic loads (1.2 and 1.5 Mbps average).

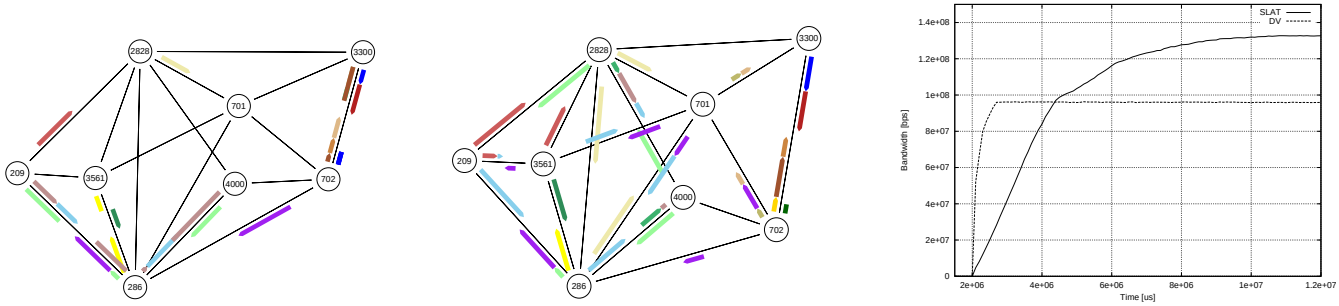
What we see is that bursty traffic makes the provider's life more difficult to find the "right" allocation. Peak rate allocation leads to lower utilization and a mean rate allocation has the drawback of buffering and potential packet loss. With SLA trading as measured here routing and provisioning can follow traffic patterns.

[12] The hardware / software setup used were several AMD K7 / 500 machines running Linux 2.2 / 2.3 and the IBM JDK 1.1.8 with its JIT. Memory usage was limited to 128 MB. A run on a 36 domain network with 100 jumbo flows took about 4 days to complete.



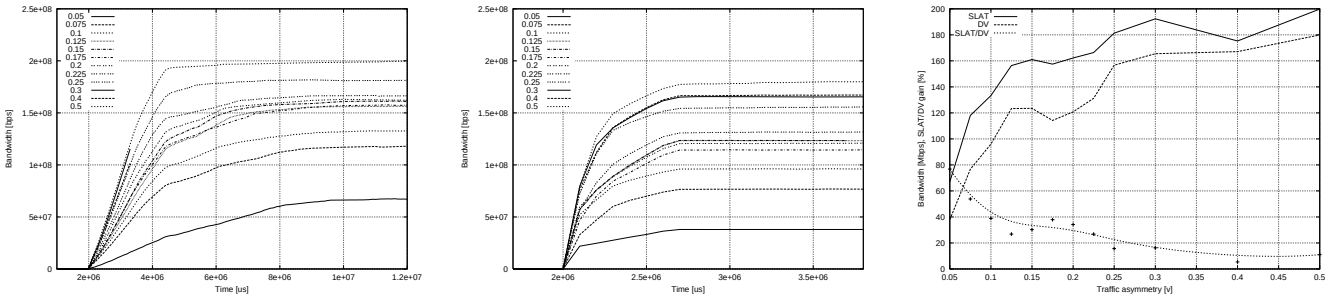
(a) (b) (c)

Figure 9: (a) Load balancing over 3 paths, (b) Competition between 3 ISPs, (c) Different trading strategies



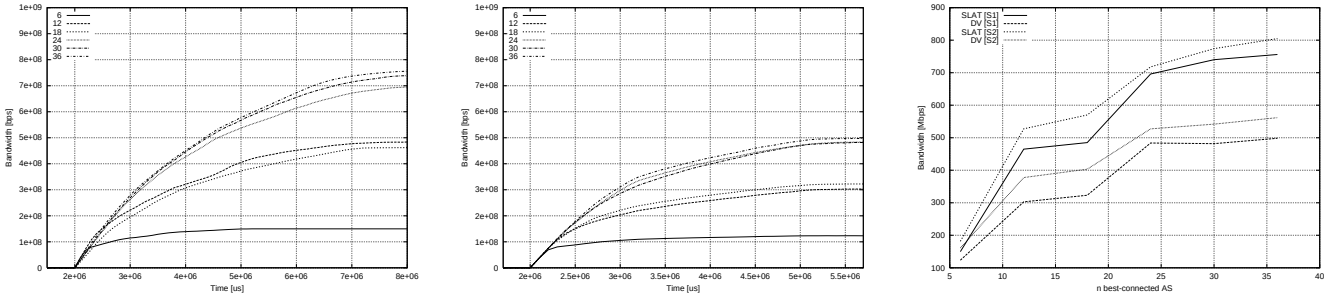
(a) Shortest path routing (DV). (b) Routing with SLA traders. (c) Throughput for all flows.

Figure 10: AS subgraph with 8 ASes.

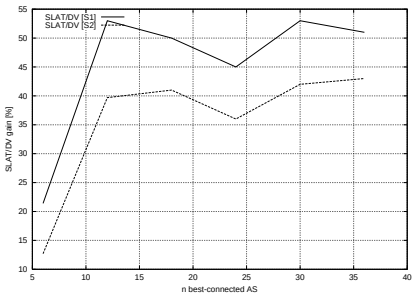


(a) (b) (c)

Figure 11: Hot spot distribution (highly non-uniform to uniform), (a) SLAT, (b) DV, (c) BW / ν, SLAT gain in %

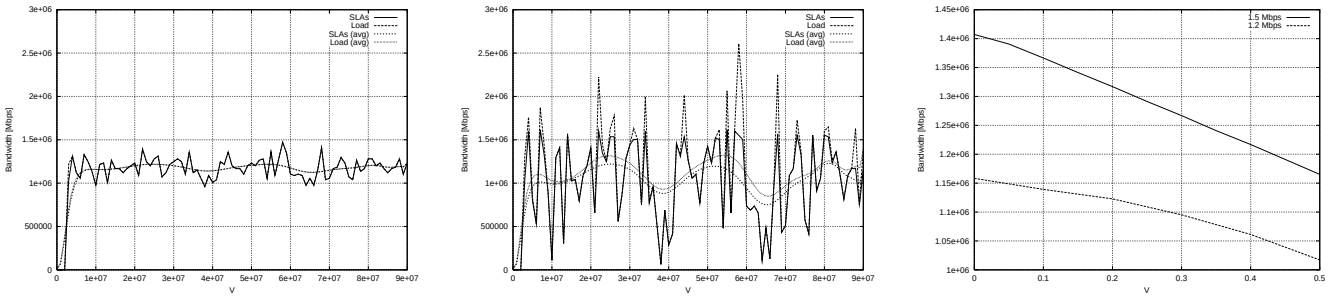


(a) (b) (c)



(d)

Figure 12: Performance of SLAT for AS networks of size 6..36; (a) SLAT, (b) DV, (c) BW / size, (d) SLAT / DV gain in %



(a) (b) (c)

Figure 13: Inter-AS traffic pattern with (a) $V = 0.1$ and (b) $V = 0.5$; (c) Throughput for $V = 0..0.5$ and two different avg. traffic loads.

6 Previous Work on Bandwidth Brokers

This section sums up several proposals and early implementations of bandwidth brokers.

Nichols et al. proposed "bandwidth brokers" for DS in [22]. While their focus is on a global architecture including router mechanisms, packet marking, and integration of other architectures like intserv, one part of their work deals with BBs that are responsible for setting up bilateral SLAs. They observe that the information exchange between BBs can range from static agreements (i.e., classical peering) without any signaling to a dynamic setup even for a single changing flow.

Clark and Fang describe the "allocated-capacity" framework which is based on a single bit to differentiate services [7]. Similar to [14] and our SLAT approach, they base service allocation profiles on traffic specifications, geographic scope, and probability of assurance.

The Internet 2 BB is currently under development and focuses initially on the intra-domain BB. The implementation will be fielded in the CA*Net II [20]. In a first phase basic features will be implemented. Later, SLAs will be treated depending on their service destination (network prefix).

The Telia Bandwidth Broker is intra-domain centered, cooperates with OSPF and uses SNMP for network setup. Emphasis is on protocol processing performance between agents [26].

Semret et al. review the DS framework in the context of a game theoretic approach [28]. Focus in this work is put stability and consistency of bandwidth allocation across several networks. First results indicate that stability can be achieved even for different service classes (AF and EF) and service levels affecting each other. However, they observed instabilities in situations of small networks (3 nodes were used in simulations) and tight provisioning.

In [8], Courcoubetis and Siris investigate how SLAs are priced when demand is measured as effective bandwidth. Furthermore they show how customers can optimally select traffic parameters of SLAs and how their model performs for real-time and non real-time service classes.

The combination of incentive compatible pricing and QoS routing is described in [15]. A theoretical approach to finding least-cost paths by using congestion pricing at each link is presented.

While Bandwidth Brokers are a relatively new concept to DS there is already some experience with carrier-level trading of raw bandwidth. Companies like Ratexchange or Arbinet become global exchanges for bandwidth. However, granularity is still pretty coarse (terms of months or longer, Mbps or faster) and the approach is targeted at "global commodity trading". Some are beginning to specialize in Voice over IP services trading. An overview is given in [18]. A novel approach to bandwidth exchanges is proposed in [5]. It suggests an extension to bandwidth exchanges that considers the routing problem even at a higher layer than SLAT does: any connections offered e.g. between two cities can be combined again and resold. The approach deals with computational complexity first but such a global view will of course also pose a scaling problem with respect to message overhead.

In the Nimrod routing architecture [4] the integration of resource provisioning information is proposed by introducing a "flow mode" with additional state. SLAs are not exchanged explicitly but are part of the routing maps. [13] proposes NetScope, a centralized architecture that integrates service provisioning, routing at the intra- and inter-domain level. SLA trading and NetScope share many similarities but are also complementary with a focus on inter- and intra-domain issues respectively.

Finally, a new working group of the IETF on traffic engineering has been formed and is planning to address the inter-domain QoS routing problem as part of its agenda [30].

7 Conclusion and Future Work

Diffserv is simple yet powerful network resource provisioning framework but without efficient and effective signaling through bandwidth brokers ISPs cannot exploit its full advantages. SLA trading provides such an innovative signaling framework for bilateral agreement negotiation between bandwidth brokers. It supports local optimization, incremental deployment, and evolving definitions of services and PHBs. This is good news for providers since they can pick the mechanisms and policies they like best. And it is also good news for customers. The competition among providers will be perceptible at the edge of the network in form of lower prices and better service.

We have shown that a *market-managed Internet is technically feasible*, if applied to the network's core. At the edge, access providers still need to collect money from end users but they are free to choose their favorite method, including the popular flat-rate.

The main technical contributions of this paper are:

- An innovative SLA trading system that fits very well into the DS framework. While it emphasizes inter-domain QoS routing it is complementary to BGP and does not replace routing of best-effort traffic. As a proof of concept, we demonstrated the system's feasibility in a prototype implementation on top of a flow-based simulation core.
- A working trading system based on market principles and purely local decisions. It encourages competition among large, well-connected ISPs. In addition, it supports local and partial deployment (as it is a compatible add-on) in the current network or in test beds. Each provider is free to choose its own strategies to pursue its local objective.
- A new, dynamic approach to DSCP allocation proposed to simplify and enhance DS PHBs ("inter-AS label switching"). Combined with an intra-domain label-switching method, e.g. MPLS, a hierarchical solution for label-switching is achieved.

- An evaluation of SLA trading using recent, Internet-typical traffic and topology models. Its main result are:
  - Statistical information of inter-domain network topology suggested to apply SLA trading to the core of the network. Simulation results with up to 36 ASes support this claim.
  - Non-uniform jumbo flow distributions (hot spots) are leveled out by SLA trading. For the Internet-typical range of ν-values we show a significant potential for load balancing (20% to 50% higher utilization than a statically configured network).
  - Bursty traffic at different time scales hurts network utilization. Using our dynamic approach, SLAs are reduced in times of low demand and reallocated when needed. Although we showed small improvements at a short time scale it works best on the medium / long time scale.

These results have their significance on the inter-domain level and affect aggregated traffic. They provide a macroscopic view of a very complex system and, due to the nature of observation, they not only abstract but also change some of the details [**?**].

While our first results are very encouraging, we need to address larger systems and more detailed network and traffic models in future work. For example, the AS data set does not reflect multiple connections between ASes which is significant for networks spanning large geographical regions. For the traffic model, more work is needed on the dynamics of hot spots.

Besides more simulations, an implementation of SLA trading on one of the upcoming DS implementations will provide further insights. At the same time, improved traders and more precise traffic descriptions as part of SLAs should be tested.

In our trader implementation we exploited some of the basic strategies (trends, cost / profit analysis, etc.) and evaluated networks with such trader instances located in all networks. In practice, however, each network provider is able to tune its traders with available local information. This should eventually lead to even better results.

References

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. RFC 2475, Dec. 1998.

[2] L. Breslau and S. Shenker. Best-Effort versus Reservations: A Simple Comparative Analysis. In *ACM SIGCOMM 98*, September 1998.

[3] B. Briscoe. The Direction of Value Flow in Multi-Service Connectionless Networks. BT Labs Technical Report, September 1999.

[4] I. Castineyra, N. Chiappa, and M. Steenstrup. The Nimrod Routing Architecture. RFC 1992, August 1996.

[5] G. Cheliotis. A Market-based Model of Bandwidth and a New Approach to End-to-End Path Computation with Quality Guarantees. In *MIT Workshop on Internet Service Quality Economics*, Dec. 1999.

[6] K. Chu. User Reactions to Flat Rate Options under Time Charges with Differentiated Quality of Access: Preliminary Results from INDEX. In *MIT Workshop on Internet Service Quality Economics*, Dec. 1999.

[7] D. D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4):362–373, August 1998.

[8] C. Courcoubetis and V.A. Siris. Managing and Pricing Service Level Agreements for Differentiated Services. In *IEEE/IFIP IWQoS 99*, May 1999.

[9] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in the Internet. RFC 2386, August 1998.

[10] W. Fang and L. Peterson. Inter-AS Traffic Patterns and Their Implications. In *Global Internet 99, Rio de Janeiro, Brazil*, December 1999.

[11] G. Fankhauser. A Network Architecture Based on Market Principles. PhD thesis, ETH Zürich, March 2000.

[12] G. Fankhauser and B. Plattner. Bandwidth Brokers as Mini-Markets. In *MIT Workshop on Internet Service Quality Economics*, Dec. 1999.

[13] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford. NetScope: Traffic Engineering for IP Networks. *IEEE Network Magazine*, March / April 2000.

[14] V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding PHB. RFC 2598, June 1999.

[15] Y. Korilis and A. Orda. Incentive Compatible Pricing Strategies for QoS Routing. In *INFOCOM 99*, March 1999.

[16] T. Li and Y. Rekhter. A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE). RFC 2430, October 1998.

[17] J. K. MacKie-Mason. A Smart Market for Resource Reservation in a Multiple Quality of Service Information Network. Technical Report, University of Michigan, September 1997.

[18] J. Makris. Bandwidth Brokers, Not Exactly Nasdaq. Data Communications Magazine, `http://www.data.com/issue/990507/brokers.html`, May 1999.

[19] Global ISP Interconnectivity by AS number. `http://moat.nlanr.net/AS/background.html`, June 1999.

[20] R. Neilson, J. Wheeler, F. Reichmeyer, and S. Hares, editors. *A Discussion of Bandwidth Broker Requirements for Internet2 Qbone Deployment*. Internet2 Qbone Bandwidth Broker Advisory Council, `http://www.merit.edu/working.groups/i2-qbone-bb`, August 1999. Work in progress.

[21] K. Nichols, S. Blake, F. Baker, and D. Black. Defintion of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474, December 1998.

[22] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the internet. `http://www-nrg.ee.lbl.gov/papers/bitarch.pdf`, November 1997.

[23] P. Pan, E. Hahne, and H. Schulzrinne. BGRP: A Tree-based Aggregation Protocol for Inter-Domain Reservations. Columbia University CS TR No. CUCS-029-99, December 1999.

[24] V. Paxson. Fast, Approximate Synthesis of Fractioanl Gaussian Noise for Generating Self-Similar Network Traffic. *ACM Computer Communication Review*, 27(5):5–18, October 1997.

[25] S. Savage et al. Detour: a Case for Informed Internet Routing and Transport. University of Washington, Seattle, Technical Report UW-CSE-98-10-05, October 1998.

[26] O. Schelen, A. Nilsson, J. Norrgard, and S. Pink. Performance of QoS Agents for Provisioning Network Resources. In *IEEE/IFIP IWQoS 99*, May 1999.

[27] D. Schweikert and G. Fankhauser. *flowsim*, A Flow-based Network Simulator. `http://www.tik.ee.ethz.ch/~fs2`, 1999.

[28] N. Semret, R. Liao, A.T. Campbell, and A.A. Lazar. Peering and Provisioning of Differentiated Internet Services. In *Proceedings of INFOCOM 2000*, March 2000.

[29] M. Shreedar and G. Varghese. Efficient Fair Queueing Using Deficit Round Robin. In *ACM SIGCOMM 95*, August 1995.

[30] Traffic Engineering Working Group Charter. `http://www.ietf.org/html.charters/tewg-charter.html`, Nov. 1999.

[31] Z. Wang and J. Crowcroft. Routing algorithms for supporting resource reservation. *IEEE JSAC*, 1996.

A Simulation Environment

A.1 Topologies

The AS topology used was taken from [19]. The *n* best-connected ASes from the data set were used. Some of the ASes are at the edge of this set (little relative connectivity) because they have the most connections somewhere else (e.g. large Asian, European providers). According to the classification of [10] this selection contains 21 class 1, and 17 class 2 ASes. A table of the 38 best-connected is given below:

Table 5: AS networks sorted by connectivity (data sources: `http://moat.nlanr.net/AS` and `whois.radb.net`).

AS	#conn.	name	AS	#conn.	name
701	1186	Alternet	3549	87	Frontier GlobalCenter
3561	692	Cable & Wireless (CW)	145	83	vBNS
1239	573	SprintLink Backbone	286	83	EUnet Backbone
1	301	GTE Internetworking	6461	76	Abovenet
7018	271	AT&T WorldNet Backbone	1673	71	---
2548	248	DIGEX-AS	3300	69	AUCS Communications
2914	244	Verio Inc.	4200	68	AGIS (Apex Global)
6453	149	Teleglobe Canada Inc.	1221	67	TELSTRA-AS
293	138	ESnet	5459	66	LINX-AS
6347	133	Savvis - St.Louis	5646	66	NAP.NET, LLC
2828	131	Concentric Network	3257	61	Nacamar Global ASN
1740	127	CERFnet	4000	60	Global IP
702	115	UUNET	174	60	Performance Systems
2497	109	IIJNET	7474	56	Optus Communications
1755	101	EBONE AS	5650	54	Electric Lightwave
721	97	---	1849	51	UUNET UK
5696	92	GoodNet AS	2516	50	KDD Japan
209	88	Qwest Communications	11042	48	---
			3333	46	RIPE NCC
			4637	45	Hong Kong Telecom

Link capacity and delay distribution is usually not available. For some networks, however, it can be guessed from whois database entries. For the simulations links speeds were set equally and a uniformly distributed delay (10 .. 20 ms) has been applied.

A.2 Trading Strategies

The following SLA trader strategies were used for separate evaluation in Section 5.1.

A *Null Trader* provides no service at all.

A *Lazy Trader* buys services only on explicit demand of customers (through *Ask Messages*). If every trader were lazy it would mean to flood (bound to AS count) the network with Ask Messages.

A *Greedy Trader* is the opposite and simply buys all of the bids it receives and is only a reference point for a badly behaving trader.

A *Trendy Trader* does an analysis if current usage of resources and buys bids by predicting future demand. The Trendy Trader is an important trader because it has the ability to book ahead for services with high demand.

Profitable Traders are an extension of Trendy Traders. See also Section 3. They conduct in addition a profitability analysis based on past trades. In general, a window of past trades of the same type is maintained and the average price p_a is computed (this window may vary and even stretch beyond expiration of SLAs backwards in time but we currently use a window size of 1). p_a multiplied with an estimate of what amount will be sold (i.e. the bit volume) [13] is compared to each available bid (bid price times volume that has to be purchased). If this balance is positive, the trade is considered being profitable and will be accepted.

B SLA Trading Protocol

B.1 Message Exchange and Handling

Message handling for the SLATP may be initiated by both buyers and sellers. This is shown in the MSC in Fig. 14(a) by an optional *ask* message. Basically, a system that does not use *ask* messages has to rely on a periodic, self-initiated distribution of *bid* messages (very similar to link state advertisements). If *ask* messages are used a peer may respond immediately with a *bid*. The accept message, followed by either a *confirm*, *a reject* or a timeout.

The protocol handler logic is shown in the state transition diagram Fig. 14(b) [14]. While this protocol engine relies mostly on its own messages, two important, external events, signalling demand and supply *may* trigger new messages.

B.2 Message definitions

In this section we give the definition of SLATP messages as the actual Java class interfaces that were used [15].

Sending a message to a SLAT-peer can be local or to a neighbor. If the destination is another node, a *PacketImpl* best-effort flow is created and started to reflect the bandwidth usage of the signalling traffic.

B.2.1 Abstract Message Type and Supporting Types

This is the base message, inherited by all specialized messages.

```
abstract class SLATP_Message implements Message {
    final Service from;
    final Service to;
}
```

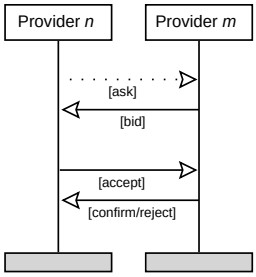
Service objects may be part of SLA Messages.

```
public interface Service
{
        Node get_node();
        int get_port();
}
```

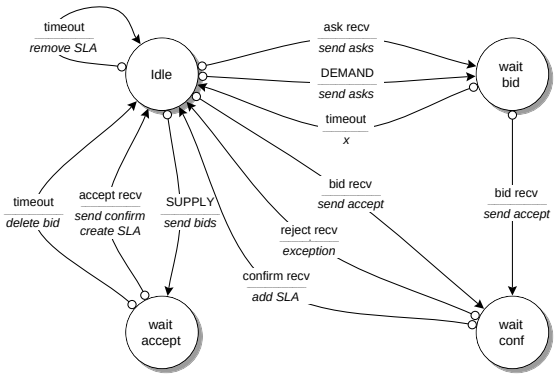
[13] This estimate is based on the history of sold SLAs, outstanding asks messages, and a target provisioning factor.

[14] A similar notation with events and actions as for TCP is used.

[15] No "real" protocol has been used. Instead message objects are passed between SLA Traders



(a) MSC



(b) STD

Figure 14: The SLAT protocol enables ISPs to communicate offers and requests (bids and asks). Upon mutual agreement SLAs may be accepted or rejected. The message sequence chart (MSC) and state transition diagram (STD) of the protocol engine are shown.

Nodes are represented by id and name. They contain a collection of links associated with link managers (e.g. schedulers).[16]

```
public interface Node
{
        int  get_id();
        String get_name();
        int  get_max_ifaces();
        int  get_ifaces_count();
        void set_iface(int n, LinkManager f);
        int  add_iface(LinkManager f);
        LinkManager get_iface(int i);
        int  get_iface(Node n);
        Link get_link(int n);
        Node get_peer(int n);
        Service get_service(int port);
        void set_service(Service service, int port);
        void attach_monitor(NodeMonitor nm);
        void remove_monitor(NodeMonitor nm);
}
```

B.2.2 Ask Message

```
public class SLATP_Ask extends SLATP_Message
{
    public final Node dest;
    public final int bw;
    public final int max_delay;
    public final long expiration;
    public final int as_distance;
}
```

B.2.3 Bid Message

```
public final class SLATP_Bid extends SLATP_Message
{
        SLA sla;
        public SLA get_sla();
        public final int size();
}
```

Bid messages represent offered SLAs and contain therefore such an object which is defined as follows:

```
final public class SLA
{
        public final Node dest;
        public int ds;
        public final int bw;
        public final int max_delay;
        public long time;
        public long expiration;
        public long cost;

        /* SLAs are used for both local computation  *
         * and messages. The following is for local  *
         * use only:                                 */

        int     interface;
        int     peer;
        int     residual_bw;
        long    bid_expiration;
        long    gain;
        boolean accepted;
}
```

B.2.4 Accept Message

```
public final class SLATP_Accept extends SLATP_Message
{
        final int  sla_ds;
        final Node sla_dest;
        public int get_sla_ds();
        public Node get_sla_dest();
        public int size();
}
```

Note that sla_ds and sla_dest identify an SLA.

B.2.5 Reject Message

```
public class SLATP_Reject extends SLATP_Message
{
        final int  sla_ds;
        final Node sla_dest;
        public int get_sla_ds();
        public Node get_sla_dest();
        public final int size();
}
```

B.2.6 Confirm Message

```
public class SLATP_Confirm extends SLATP_Message
{
        final int  sla_ds;
        final Node sla_dest;
        public int get_sla_ds();
        public Node get_sla_dest();
        public final int size();
}
```

[16] For documentation, visit [27].