A Graph-Theoretical Notation for the Construction of LSP Hierarchies

Michael Menth and Norbert Hauck
Institute of Computer Science, University of Würzburg
Am Hubland, D-97074 Würzburg, Germany
phone: (+49) 931-8886644, fax: (+49) 931-8886632
email:{menth|hauck}@informatik.uni-wuerzburg.de

Abstract

To achieve scalability for reservations in packet switched networks, the concept of tunnel and funnel aggregation is used. They are both supported in the data forwarding plane of Multiprotocol Label Switching (MPLS) by label switched paths (LSPs). We propose the recursive application of tunnels and funnels that leads to an LSP hierarchy which yields more scalable reservation structures. We propose a graph theoretical notation for tunnels, funnels, and the recursive application of these concepts. Valid aggregation scenarios are described by constraints in the hierarchy graph. A cost function assesses the scalability of an LSP hierarchy with respect to the number reservation states in the routers. The notation, the constraints, and the cost function are the base for heuristics that find suited overlay networks. We explain the principles of our aggregation algorithms whose output can be directly used for MPLS network configuration. Simulation results demonstrate the effectiveness of our approach.

Keywords: MPLS, reservation aggregation, scalability, QoS

1 Introduction

The challenge of future IP networks is the provisioning of toll quality data transport for real-time applications, i.e. quality of service (QoS) in terms of loss and delay bounds must be met. For this purpose, the Internet Engineering Task Force (IETF) proposed the Integrated Services (IntServ) approach [1, 2] which is tightly coupled with the Resource Reservation Protocol (RSVP) [3]. Every reservation induces a state in every router on its path, therefore, IntServ works well for small intranets with only a few QoS flows. However, it fails on a large scale like in backbone networks since the number of reservations is large and the routers would be mostly busy with bookkeeping.

The Differentiated Services (DiffServ) approach [4, 5] was a reaction to that problem. Traffic is handled on an aggregate basis and per flow information is avoided. To support QoS, a bandwidth broker (BB) is introduced. Essentially, it is a central entity that stores the distributed resource allocation information from IntServ and relieves the routers from that burden. But this does not solve the actual scalability problem for end-to-end (e2e) reservations [6], it even introduces a single point of failure.

An effective means to reduce the amount of reservation information especially in transit networks is reservation aggregation. Several flows are forwarded using a single reservation that is large enough to provide all of them with the desired QoS. This aggregate reservation may have a single start and end point which corresponds to a tunnel [7, 8]. If the aggregate reservation has several start points and a sink tree structure towards a single end point, the reservation is a funnel [9, 10]. Tunnels and funnels may be applied recursively to obtain an even higher degree of scalability for reservation states.

Multiprotocol Label Switching (MPLS) helps to aggregate traffic in label switched paths (LSPs) in tunnels and funnels. Tunnels can be established with QoS attributes [11, 12] and hierarchical structures of LSP tunnels are currently promoted [13, 14].

In this work, we suggest a graph-theoretical notation for a hierarchical system of tunnel and funnel reservations which corresponds to an LSP hierarchy in the MPLS context. The notation may be used by BBs or network management software for network configuration and resource allocation. Furthermore, we identify constraints in an LSP hierarchy due to aggregation and due to the network topology and formulate them in our notation. A cost function evaluates the quality of a given LSP hierarchy. Based on the notation and the constraints we have developed some heuristics that find suitable LSP hierarchies for given networking scenarios. We compare several alternatives and show the effectiveness of the reservation aggregation approach with respect to scalability.

The problem of optimal virtual path layout in ATM networks [15, 16, 17] is similar to the design of an LSP hierarchy. The fundamental difference is that ATM technology offers only a single aggregation level using the virtual path concept and it does not allow sink trees. This makes the work for ATM networks significantly simpler and more straightforward.

The paper is structured as follows. In Section 2, we give an introduction to the scalability problem for e2e

reservations. We explain the idea of reservation aggregation and explain how tunnels and funnels work. We suggest to use MPLS to support these reservation aggregation principles. The recursive application of reservation aggregation is possible and leads to an LSP hierarchy. Section 3 covers the graph-theoretical notation for reservation aggregation, derives the constraints in the hierarchy graph due to flow aggregation, and proposes a cost function. Section 4 explains the basic approaches of our heuristics for finding LSP hierarchies in given networking scenarios and simulation results evaluate these algorithms. A comparison between simple LSP aggregation and hierarchical LSP aggregation recommends the use of hierarchical LSPs structures to enforce scalability in IP networks. Finally, we conclude this paper with a short summary and an outlook for further research.

2 Reservation Aggregation for Scalable E2E Resource Allocation

In this section we describe the scalability problem for e2e reservations using IntServ as an example. The alternative DiffServ approach can neither solve the basic problem. MPLS technology supports the concepts of tunnels and funnels for flow aggregation in the data path. These concepts are also used for reservation aggregation to achieve scalability in the control path. We explain how tunnel and funnel reservations work and motivate their recursive application in an LSP hierarchy.

2.1 Integrated Services

IntServ is characterized by the separate handling of each individual e2e flow. The Resource Reservation Protocol (RSVP) [3] establishes a state for every e2e flow in all routers along its path. These states comprise traffic and reservation descriptors (T_{spec},R_{spec}) that are used to manage the capacity on every outgoing interface and to enforce policing on a per flow basis. In particular, an admission control (AC) entity uses these data to decide whether an additional flow can be admitted. The packets of every flow are classified and policed, and individual queue scheduling states are maintained for each flow to meet the booked QoS objectives. These actions require many lookups for each forwarded IP packet and, therefore, IntServ works fine for a few ten thousand reservations but it is difficult to support in backbone networks where the number of QoS flows can be very large.

2.2 Differentiated Services

The DiffServ approach uses the Differentiated Services Code Points (DSCPs) in the IP header to define only a few different Per Hop Behaviors (PHBs) which are essentially transport service classes. The routers treat the IP packets with low or high priority in the forwarding process according to their DSCPs. No per flow information is stored and the forwarding process operates on aggregated traffic and not on single e2e flows. Therefore, this basic architecture scales well for large networks. QoS can be achieved by massive capacity overprovisioning. But for reasonable network utilization, smart resource management is required that also includes AC at least at the network boundaries. Therefore, bandwidth brokers manage the network-wide resources and perform AC for individual reservation requests. They store per flow information for packet classification, policing and shaping to control the ingress of the network. This relieves the routers from the burden of per flow management but it shifts the distributed scalability problem in terms of information states to a central entity and the actual problem is not solved.

2.3 Multiprotocol Label Switching

MPLS is a mechanism to allow packet switching instead of routing over any network layer protocol [18]. A connection in MPLS is called a label switched path (LSP). The first label switching router (LSR) equips the IP packet with a label of 4 bytes and sends it to the next LSR (cf. Figure 1). The LSRs classify a packet according to its incoming interface and label. Based on this information, label swapping is performed and the packet is forwarded to the corresponding outgoing interfaces. The last LSR removes the label from the IP packet header and forwards the packet according to IP routing. LSPs can also use other LSPs as forwarding adjacencies since LSPs create a logical point-to-point (p2p) link between the ingress and the egress LSR. The resulting data packets carry several stacked MPLS labels in their headers.

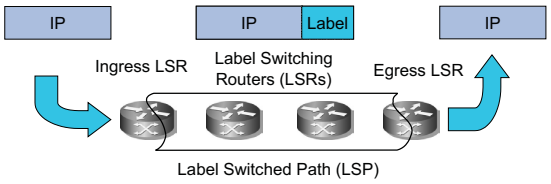

Figure 1: An LSP is a virtual, tag-switched connection between ingress and egress LSR.

The label swapping process requires entries for every LSP in the management information base (MIB) of the LSRs. This causes also information states per LSP like in IntServ for e2e reservations. There are two major protocol alternatives for establishing an LSP.

MPLS is often viewed as a modified version of the Asynchronous Transfer Mode (ATM) with variable cell size. But there is a profound difference: ATM enables with its virtual channel and virtual path concept only simple aggregation of basic flows while MPLS allows for many-fold aggregation using multiple label stacking.

2.4 Aggregation Alternatives

Several flows are aggregated if they are comprised to a new flow. In principle, there are two alternatives to accomplish this: funnels and tunnels. We explain these fundamental concepts in the data path and use MPLS terminology for that objective.

2.4.1 Tunnels

An ingress LSR aggregates several flows into a tunnel LSP by pushing a new common label onto the label stack of the packets. The intermediate LSRs of the tunnel LSP can not distinguish the e2e flows anymore and also their control messages are transported as common MPLS packets with the same label. As a consequence, the control messages are bypassed at these LSRs and the e2e flows do not set up any states. This achieves scalability for e2e flows on that LSP. When the egress LSR of the tunnel LSP removes the uppermost label, the original flows are restored. Hence, LSP tunnels act like logical links between ingress and egress LSR. The resources for that logical link are assigned by a single aggregate reservation for that LSP whose size can be naively computed as the sum of the reservations of the carried flows. This is achieved by an extension of RSVP and LDP for LSP tunnels [11, 12], i.e. by sending RSVP control messages.

Hence, tunnels reduce the state information in the intermediate routers, they allow for reservation aggregation and deaggregation, and this concept is already implemented in MPLS. But if e2e connectivity must be realized in a star network with n access routers, the center node has to handle all possible tunnels which amounts to exactly $n \cdot (n-1)$ LSPs. Thus, the number of required aggregates scales quadraticly with the network size and that can be too much in large networks.

2.4.2 Funnels

An LSR merges LSP flows into one LSP by substituting their uppermost label by a new common label. Thus, funnels reduce the amount of state information in the LSRs towards the destination router. Figure 2 visualizes the resulting sink tree towards a common destination and motivates the name funnel for this kind of aggregation. Finally, the egress LSR of the sink tree removes the uppermost label. Note that the original aggregate at the ingress LSR can not be inferred from the label at the egress. Hence, flow related information can be aggregated but it can not be deaggregated. When flows are merged, their reservations can be (naively spoken) summed up into a combined downstream reservation. At the end of the funnel, only one reservation exists and the information about the demand at the ingress LSRs is lost at the egress LSR.

To achieve full connectivity in a network with n nodes, every node is an ingress LSR for $n-1$ LSPs that part of sink trees to any other node. This means that the number of required LSPs in an LSR scales linearly with the network size and so does the number of reservation states.

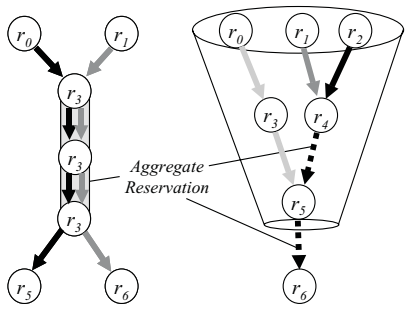

Figure 2: Tunnel aggregation in contrast to funnel aggregation.

In [9] funnels are set up for bandwidth brokers and in [10] the Border Gateway Reservation Protocol (BGRP) suggests soft state funnel reservations. Funnels are implemented in MPLS only in the data path but not yet in the control path, therefore, BGRP can be an example for the enhancement of RSVP-TE or CR-LDP towards sink tree reservation capabilities.

We explain briefly how signaling works in BGRP to set up and modify a funnel reservation. A PROBE message is sent from a source router to a destination router and collects the visited border routers. Upon receipt of a PROBE message, the border routers check for available resources, and forward the PROBE packet towards the destination. The destination border router terminates this process. It converts the PROBE message into a GRAFT message and inserts an ID that identifies its sink tree. The GRAFT message travels back on the collected path, establishes the required reservation states, marks them with the ID, or updates them if they are already existent. The PROBE and GRAFT messages contain only a relative reservation offset, therefore, the communication for GRAFT messages must be reliable (e.g. over TCP). BGRP is a soft state protocol, therefore, neighboring routers exchange explicit REFRESH messages to keep the reservation alive. In principle, this concept can be adopted for the signaling of funnel reservations in MPLS networks to make its control plane as scalable as its data plane.

2.5 Scalable QoS Support Using an LSP Hierarchy

We imagine v virtual private networks (VPNs). They all consist of two sites with w subnetworks each and these sites are linked by a hierarchically structured transit network (MAN, WAN). All hosts of the subnetworks of a single VPN communicate with each other using e2e reservation based on the IntServ model.

Without any aggregation the core routers in the WAN are overloaded with e2e reservations. Therefore, we inter-

connect the subnetworks of a single VPN by tunnel reservations. This results into $2 \cdot v \cdot w^2$ tunnel reservations that are visible in the transit network. The simple funnel aggregation requires only $2 \cdot v \cdot w$ reservations in the MAN and WAN. This denotes a significantly smaller number of states than with the tunnel solution.

Now, we suggest a recursive application of the tunnel and funnel aggregation concept that takes advantage of the hierarchical network structure. The subnetworks of a single VPN at the two main sites are interconnected by $2 \cdot w$ funnels. These reservations can be aggregated by two LSPs that link two main sites of a VPN in both directions. This reduces the complexity to $2 \cdot v$ reservations in the transit network.

The number of reservation and MPLS forwarding states in the WAN can be further reduced. The first core router in the WAN eventually aggregates the $2 \cdot v$ LSPs into 2 LSPs in the core which is independent of v and w. The tunnels are set up for the transport through WANs and MANs because they can take advantage of the hierarchical network structure (Figure 3) due to their deaggregation capability. The resulting hierarchical reservation aggregation can achieve very high scalability in large networks.

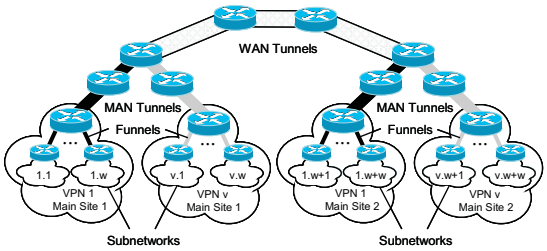

Figure 3: Hierarchical traffic aggregation using tunnels and funnels reduces the number of flows in the transit network.

In addition to forwarding and reservation scalability, MPLS technology also simplifies the packet classification process because aggregated LSP flows can be simply identified by the packet labels and the individual packet headers do not have to be investigated anymore.

This example motivates the introduction of a hierarchical tunnel and funnel structure as an overlay topology to reduce the number of reservations in IP/MPLS networks. In our example, the structure of the LSP hierarchy is obvious. However, reservation aggregation should also be done in networks without an obvious hierarchy to reduce the number of state. As a first step towards that goal, we suggest a graph-theoretical notation for LSP hierarchies and derive constraints in such a graph that reflect the conditions introduced by flow tunneling and merging.

3 Graph-Theoretical Notations for an LSP Hierarchy

In this section, we present a graph-theoretical notation for aggregation concepts. We first explain the basic terms for network topology, LSPs, paths, and flows, and develop a concept to write down the LSP hierarchy which translates directly into a configuration for an MPLS network. If an LSP hierarchy is constructed for a given networking scenario, some constraints that arise from flow tunneling and merging must be respected. We formulate these constraints as rules within a hierarchy graph. A simple cost function evaluates the corresponding amount of RSVP and MPLS states in the router MIBs. This yields eventually an optimization problem that helps to find the best overlay model.

3.1 Network Topology, Links and Flows

A *network topology* consists of a set of *routers* $\mathcal{R}$ which are connected by a set of *network segments* $\mathcal{S}$. A network segment s is identified in router r_0 by an outgoing interface and in router r_1 by an incoming interface. A network segment s is a link and the operator $c(s) = (r_0, r_1)$, $r_0, r_1 \in \mathcal{R}$, returns the routers that are soruce and sink of link. Hence, the links are directed.

An *LSP* m is a set of links that are connected as a sink tree with sink router r_s. The *set of LSPs* is denoted by $\mathcal{M}$.

The introduction of LSPs defines new forwarding adjacencies (LSP threads). An *LSP thread* t is a concatenation of contiguous links $t = [l_i]_{0 \le i < n}$ such that $c(l_i) = (r_i, r_{i+1})$, $0 \le i < n$ holds and the last router is the sink of the LSP ($r_n = r_s$). LSP threads and network segments are both links and may be used to construct other LSPs.

An LSP may have specific LSRs (*tunnel points*) that tunnel other flows, i.e. a label is pushed onto the label stack. Hence, a tunnel point is the start LSR of an LSP thread. There are also *merge points* which are tunnels that swap two different label values from two different flows to a common value. An *LSP stub* is a maximum contiguous concatenation of links in an LSP m without intermediate tunnel and merge points. The *set of all LSP stubs* q is denoted by $\mathcal{Q}$. As a consequence, LSP m is partitioned into its LSP stubs and these do not overlap:

$$\forall q, q' \in \mathcal{Q} : (q \neq q' \Rightarrow q \cap q' = \emptyset) .$$

If an LSP has only one LSP thread t, this is also the only LSP stub q. Moreover, this LSP is a *tunnel* since there is exact one ingress LSR and one egress LSR (cf. Figure 4). Otherwise, the LSP has several ingress LSRs and represents a *funnel* (cf. Figure 5). We distinguish the kind of LSPs because a tunnel LSP can be set up with already existing signaling protocols while a funnel LSP requires more complex BGRP like mechanisms for its setup.

The *path* of a network segment s is the segment itself ($path(s) = [s]$) and the path of any other link is the concatenation of the paths of the concatenated links. The length of a path is the number of its contained network segments and the length of a link is the length of its path. A path p *contains* another path p' if all network segments of p' are in the same order in p as in p' ($p \supseteq p'$). The *intersection* ($p \cap p'$) of two paths p and p' is the set of maximum paths that are contained both in p and in p'.

The path of a link l must not contain any loops, i.e.

$$path(l) = [s_i]_{0 \le i < n} \quad \wedge \quad c(s_i) = (r_i, r_{i+1}) \wedge ((i \neq j) \quad \Rightarrow \quad (r_i \neq r_j))$$

We introduce an *order* among the network segments within the path, namely "$s_i <_l s_{i+1}$". The order "$<_l$" is absolute and transitive within a path of a link l. An LSP stub is a link, therefore, each stub q induces an absolute order "$<_q$" on the links and the segments of its path. When this order is extended to an LSP m ("$<_m$"), it is not anymore absolute since network segments can only be compared if they belong to the same LSP thread.

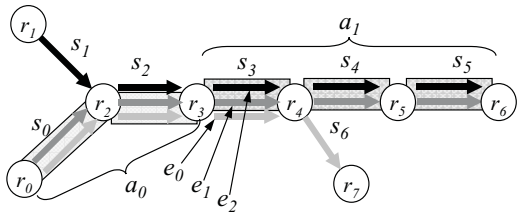

Figure 4: An aggregation scenario: network segments, e2e flows, and two tunnel LSPs.

A *flow* is a subset of packets that are transported on the same path. The operators $path(f)$ and $length(f)$ return the corresponding path and length.

The packets of an e2e flow can be described by a common source and destination IP address and UDP or TCP port number. They can be tunneled from their source to their destination without any constraints. We denote the *set of e2e flows* by $\mathcal{E}$.

Tunnel and merge points aggregate packets from different flows into one common LSP thread. This happens either by label swapping or by pushing a label on the stack. To be able to do that, these flows must not be inside another tunnel at that router. This implies that no other LSP can tunnel LSP threads along their tunnel and merge points. Hence, only the LSP stubs are flows that can be further aggregated.

3.2 LSP Hierarchy

As motivated before, the recursive application of tunnel and funnel LSPs may contribute to a scalable information exchange about resource demands between routers.

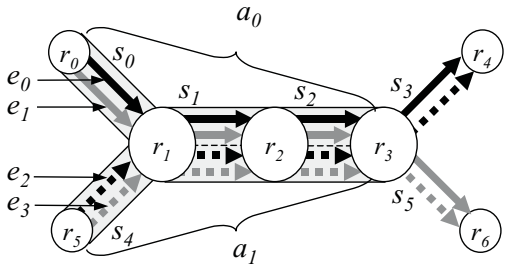

Figure 5: An aggregation scenario: network segments, e2e flows, and a funnel LSP.

We specify the resulting LSP hierarchy with a graph-theoretical notation.

We specify the LSP hierarchy generally by a graph $\mathcal{H} = (\mathcal{F}, \mathcal{T})$ and $\mathcal{M}$. The set of links and flows $\mathcal{F} = \mathcal{S} \cup \mathcal{Q} \cup \mathcal{E}$ corresponds to the nodes in the graph and the set of directed edges $\mathcal{T}$ represents the transport relationship among links and flows. The set $\mathcal{M}$ of LSPs is required to overlay the hierarchy graph with information that mark the LSP structure.

If flow f is transported over the link l, we call l a parent of flow f along $path(l)$ and f a child of l. The parent-child relationship is marked in the graph by a directed edge from the parent to its child. Figure 6 shows an LSP hierarchy that corresponds to the network structure with two tunnel LSPs depicted in Figure 4. The network segments are characterized by rhombuses, e2e flows by rectangles, LSP stubs by thin drawn through ovals and the information from $\mathcal{M}$ is given by the thick dotted ovals. An LSP Hierarchy with one funnel LSP is also illustrated in Figure 5 according to the corresponding networking scenario in Figure 7. For an easier identification of the funnel LSP structure, we connect the LSP stubs together in downstream direction of the sink tree.

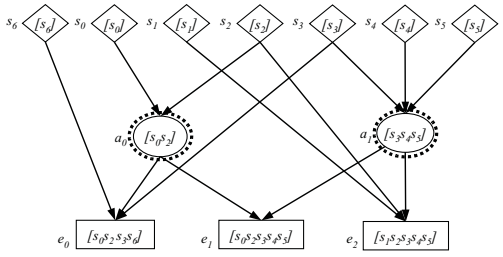

Figure 6: An LSP hierarchy with two tunnel LSPs according to the networking scenario in Figure 4.

The operator $children(l)$ returns the set of flows that are children of link l and, analogously, $parents(f)$ returns all parents of flow f. It is also possible to apply the $parents$ and the $children$ operator on a set of links and flows which yields the union of

their parents or children, respectively. Given this, we can define the offspring of a link by *offspring(l)*= $\cup_{1\leq i<\infty} children^i(l)$ and the ancestors of a flow by $ancestors(f) = \cup_{1\leq i<\infty} parents^i(f)$ by applying the $children$ ($parents$) operator several times. Recall that we unified links and flows, a motivation for this are LSP stubs.

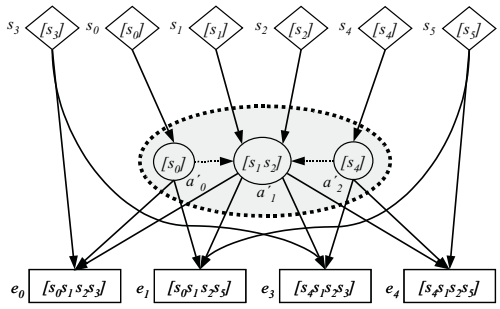

Figure 7: An LSP hierarchy with one funnel LSP according to the networking scenario in Figure 5.

This graph theoretical notation can by easily applied to large structures and processed by algorithms. In principle, there are many choices for aggregation, therefore, there are many possible LSP hierarchies for a single networking scenario. For example, Figure 8 offers an alternative structure to the one given in Figure 6. Our objective is to find an overlay topology that is best suited for the support of real-time services. Therefore, we derive constraints for the graph in order to construct algorithms that can build valid LSP hierarchies automatically.

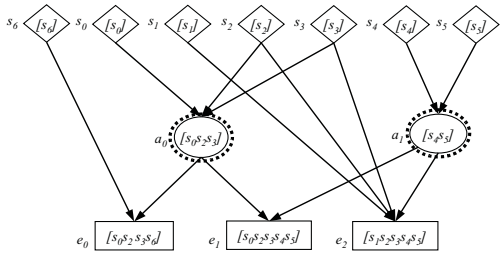

Figure 8: An alternative LSP hierarchy to the one given in Figure 6.

3.3 Properties of a Valid LSP Hierarchy

In this section we derive properties of a valid LSP hierarchy. These constraints arise from general network and aggregation aspects, properties that we can infer from others, and properties that we postulate in order to build an efficient hierarchy. We formulate them as graph-theoretical rules so that they can be easily verified by algorithms.

3.3.1 Constraints Due to Network and Aggregation Aspects

- The network segments $\mathcal{S}$ are physical links. Links can carry traffic, so they can support other flows, i.e. they can be parents. But they are not flows, therefore, they can not be children. Hence, network segments constitute the roots in the LSP hierarchy.
- The LSP stubs $\mathcal{Q}$ are flows and have to be carried over other links. Therefore, they have parents and can not be roots. But on the other side, they compose the LSP threads which are virtual links that have children, otherwise they are useless. Therefore, they can not be leaves in the graph.
- The e2e flows $\mathcal{E}$ are the actual flows that have to be transported and can not serve as links for other flows, so they can not be parents. As a consequence, they are the leaf nodes in the LSP hierarchy.
- Only LSP threads are links that can carry other traffic. In case of an tunnel LSP, there is only one LSP stub which is also an LSP thread. With funnel LSPs this is different. If an LSP stub is parent of another flow f, all the LSP stubs of the same LSP m on the unique path down the sink tree must also be parents of that flow f as well. This can be expressed as

$$g \in m \wedge g \in parents(f) \Rightarrow$$
$$(\forall h \in m : g <_m h \Rightarrow h \in parents(f))$$

  Hence, the funnel stubs do not have the same general tunneling ability as tunnel LSPs.
- A circle in $\mathcal{H}$ would denote that a flow is an ancestor of itself. This is not possible because a flow can not be transported over itself. Hence, an LSP hierarchy does not contain any circles.
- A flow can have several parents if it is transported over a path with several links. However, the paths of the parents can not have any network segments in common. This denotes that the intersection by pairs of paths of the parents is empty:

$$a, b \in parents(f) \Rightarrow path(a) \cap path(b) = \emptyset.$$

- If a is parent of f, then the path of a is a subpath of f ($path(a) \subseteq path(f)$). The flow f contains a in its path and, therefore, $path(a)$ is contained in $path(f)$. Since the $path$ operator works recursively, $path(b)$ is also a subpath of $path(f)$ if it is an ancestor of f although b is not a direct child of f.

$$b \in ancestors(f) \Rightarrow path(b) \subseteq path(f)$$

  Because of this property, we can denote the LSP hierarchy of an LSP stub in bracket notation. For example, e_0 from Figure 6 can be denoted by $e_0 = ((s_0, s_2), s_3, s_6)$. This can serve to embed an e2e flow into an existing hierarchy.

- Note that the inversion of the last sentence is not true. Figure 9 shows that $path(e_0)$ contains $path(a_1)$ but a_1 is not an ancestor of e_0 since a_1 would conflict with a_0 as a parent of e_0.

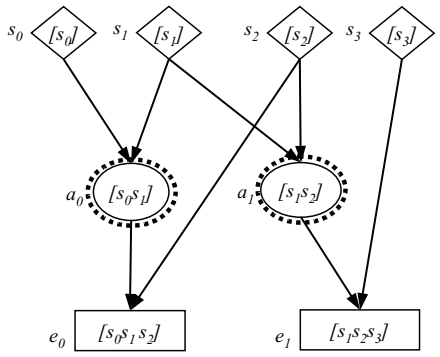

Figure 9: $path(a_1) \subseteq path(e_0) \nRightarrow a_1 \in ancestors(e_0)$

- Only network segments are hardware links that really transport traffic. Hence, the traffic of LSP stubs and e2e flows is finally transported over network segments. Therefore, all network segments in the path of a LSP stub or e2e flow f must be in $ancestors(f)$.

3.3.2 Flow Properties in the LSP Hierarchy

Furthermore, we derive some properties for flows within the LSP hierarchy. They are useful for algorithms that construct LSP hierarchies.

- We have argued that $\mathcal{H}$ does not contain any circles. Therefore, we can define the depth of a flow f within $\mathcal{H}$ ($depth(f)$). If f is a leaf node, i.e. an e2e flow, its depth is $depth(f) = 0$. If f is an aggregate, its depth is given by $depth(f) = max_{g \in children(f)}\big(depth(g)\big) + 1$.
- The parents of a flow f correspond to the links in its path. Since the parents of a flow have an empty intersection, the order "$<_f$" can be further extended to the subpaths of $parents(f)$. This entails an absolute order among the parents of a flow.

3.3.3 Postulates for an Effective LSP Hierarchy

LSP hierarchies are expected to reduce the number of states in the routers as efficiently as possible. To support this, we formulate further conditions for the LSP hierarchy.

- An LSP thread is only useful if it aggregates at least n flows. This translates into the condition that every aggregate flow in the graph must have at least n children.
- An LSP is only useful if it is longer than l links. This length l should be larger than 1.

3.4 Cost Function

As mentioned before, the flow parameters are tracked for RSVP connections or LSPs in the MIBs of the routers. Therefore, a vast amount of such states is prohibitive. Our goal is to take advantage of an LSP hierarchy for QoS support in IP networks to reduce the number of states in the involved forwarding machines. Therefore, we suggest a cost function that counts the RSVP states for e2e connections, the states for LSP tunnels, and the states for LSP funnels in the hierarchy graph.

3.4.1 Counting RSVP States for E2E Connections

Routers need the traffic descriptors and QoS requirements for e2e connections whenever these flows are sent over a link. Therefore, we assign n cost points to a router that sends n e2e flows to an outgoing interface. In the context of this work, the link-flow relation is represented by the parent-child relationship in the graph. Therefore, the first router of each concerned parent is charged with n cost points. If the parent a is an LSP stub, it receives only the cost points if it is the ingress router for the e2e flow into this LSP thread. It does not get a cost point if there is another parent b of f in the same LSP and b is farer away from the the sink router ($b <_f a, b <_m a$).

3.4.2 Counting LSP Stub States

With LSP, this is slightly different. We assume that every label induces one cost point and assigns a splitted cost point to the both routers that push, swap or pop it. Therefore, we decide that every LSP tunnel imposes half a cost point on the first and last router in each of its parents (like above). If the parent is an LSP stub, half a cost point is assigned as above and the other half cost point is assigned to the corresponding sink router.

Note that this cost function can also be differently designed, e.g. by differentiating between reservation states and label states.

3.4.3 Evaluation of the Complete LSP Hierarchy

The overall number of states in a router is the sum of states induced by e2e RSVP connections and LSP stubs. In a real world scenario, we prefer a network with equally loaded machines (in terms of reservation states) to a network with many little loaded machines and a few heavily loaded machines. Therefore, a meaningful measure for scalability of the entire hierarchy is the maximum number of states in all routers of the network.

3.5 Application of the Concept

Assume that we have a network topology and a flow matrix. Then, we need algorithms that construct LSP hierarchies that lead to a maximum state reduction in the routers. The outcome is an LSP hierarchy graph that can be easily translated into a network configuration. The graph does not need to be computed in real-time and it suffices to recompute it if the flow matrix has changed significantly. This configuration can be useful in MPLS supported DiffServ networks. The reservations can be aggregated and deaggregated together with the flows to track the amount of admitted premium traffic in a scalable way.

4 Basic Heuristics and Simulation Results

In this section, we first describe the basic structure of a class of algorithms that construct valid LSP hierarchies. They use the graph-theoretical notation and the derived properties of Section 3, and respect the constraints within a LSP hierarchy. We use only tunnel aggregation since MPLS is at the moment not capable to handle funnel reservations. Hence, the output are aggregation scenarios that can be directly transformed into MPLS network configurations. We compare the proposed algorithms and show the state reduction for MPLS networks with up to 128 nodes.

4.1 Heuristics for Least Cost Overlay Networks

Based on the above graph-theoretical notation and the constraints, we construct various algorithms to find LSP hierarchies from a given set of e2e flows within a network topology. We do not explain the algorithms in detail but sketch their basic structure. The algorithm starts with a network hierarchy containing only physical links. New e2e flows are inserted successively by connecting them as children of appropriate existing nodes in the hierarchy and establishing suited tunnel LSPs. The insertion of a flow modifies the hierarchy graph and the aggregation algorithm is sensitive to the order in which the e2e flows are inserted. We implement five different strategies.

- Random: The flows are inserted into the graph in a random order. This approach resembles the situation when a hierarchy is not constructed from scratch but if new flows are added successively to an already existing hierarchy.

- Shortest Path First (SPF): The flows are inserted into the hierarchy according to their path length in ascending order.

- Longest Path First (LPF): The flows are inserted into the hierarchy according to their path length in descending order.

- Most Traffic Path First (MTPF): The overall amount of traffic traversing a segment is evaluated and the traffic on a path is the minimum traffic on its segments. The order of the paths is according to the traffic volume in descending order. This approach correlates to SPF.

- Least Traffic Path First (LTPF): The order of the paths is according to traffic volume in ascending order. This approach correlates to LPF.

The aggregation algorithm files the unconnected subpaths of a new flow f in the set $\mathcal{U}$. To find a candidate node g as a suitable parent of an unconnected subpath $u \in U$, a Depth-First-Search (DFS) algorithm is used for the exploration of the already existing hierarchy. The path of g should share as many segments as possible with u and obey some requirements that diversify the algorithm. If g is a tunnel LSP, g can be directly connected as a parent to u, if g is a leaf node, a copy of g is made and taken as a common parent for g and u.

By the choice of an appropriate parent g, conflicts with constraints in the hierarchy can be avoided. We classify our aggregation algorithms using this aspect.

- The option "find a contained path" (FCP) denotes that the path of g is fully contained in $u \in \mathcal{U}$. In that case, g can be a parent of u without violating any constraints.

- The option "find a maximum path" (FMP) means that the path of g may have also segments that are not contained in u. This relaxed condition allows to find another node g^* that has a larger common subpath with u than g. In that case, g^* is not a direct parent and an auxiliary node $h = g^* \cap u$ is inserted into graph as a common parent of g^* and u. In addition, several other complex adjustments have to be made to the graph to keep the hierarchy in accordance with the constraints.

The second option regards the processing order of the path segments of f.

- The policy "take the first subpath" (TFS) is that the path of g must start with the first segment s_0 of any $u \in \mathcal{U}$. This can be expressed as

$$\forall s \in path(u) : u \in \mathcal{U} \wedge s_0 \neq s \Rightarrow s_0 <_f s.$$

  At the beginning, we have $\mathcal{U} = \{path(f)\}$. If we apply this policy iteratively, $\mathcal{U}$ contains always at most one subpath u, whose length is steadily reduced.

- With the policy "take the longest subpath" (TLS), the path of g is chosen such that it matches the largest possible common subpath within any $u \in \mathcal{U}$ without any constraint for start segments. Thus, we may get a g that shares a a longer path with u than in the TFS case. When this option is applied, $\mathcal{U}$ is potentially broken down into a set of many small subpaths.

Finally, all aggregating nodes with less than n children are eliminated since they do not substantially contribute to state savings in the routers. This is done by connecting their children directly to their parents. The insert order of the e2e flows and the search options (FCP, FMP) and (TFS, TLS) for a suited parents in the hierarchy are orthogonal to each other. Their combination results into 20 different algorithms that we have tested in spanning tree networks where every node is an access router.

4.2 Comparison of the Proposed Algorithms

The aggregation of n e2e flows generates a single new flow and only one aggregate reservation is maintained in transit nodes as opposed to n e2e reservations. This is a trivial result that shows that aggregation makes sense but it does not need to be confirmed by numerical results. In addition to that, hierarchical LSPs can reduce the number of aggregate flows within a transit domain. This is easy to see and they are easy to design for a hierarchically structured network as shown in Figure 3. Regularly structured tree topologies, where only the leaf nodes are able to be source and destination routers for e2e connections, are best for the effectiveness of hierarchical traffic aggregation. We weaken this assumption by admitting randomly constructed spanning tree networks where every node sends real-time streams to any other node. As pointed out before, we first bundle all e2e flows into an e2e tunnel LSP. Then, we reduce the number of MPLS states in the network by the use of further tunnel LSPs.

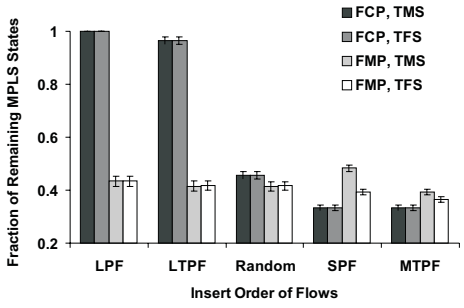

Figure 10: MPLS state reduction by different hierarchical aggregation strategies.

Figure 10 shows the reduction of the maximum number of MPLS states in the router MIBs as an average over multiple simulation runs. The networks contain 32 routers and the topology is randomly constructed for every simulation run. The error bars represent a confidence interval of 95%. The modified aggregation strategies exhibit different performance.

The combination of LPF and FCP does not yield any state savings. When a new flow is inserted into the hierarchy, only longer flows are already connected there and can not be parents of the new and shorter flow. No aggregation is possible. LTPF yields an order similar to LPF and, thus, the performance in combination with FCP is also very bad. The evaluation of the random order is important for establishing a LSP hierarchy incrementally without knowing all flows at the beginning. In that case, FMP is also better than FCP. But FCP yields the best overall performance in combination with SPF and LTPF since all possible parents are already connected in the hierarchy. However, for hierarchical networks like the one in Figure 3, the FMP strategy is important since the path of not a single flow is contained in the path of any other flow in that example. We see that both FCP and FMP are useful. TFS seems to have little advantage compared to TMS, however, TFS also fails wherever FCP is not applicable. Therefore, TFS and TMS are both important strategies. We have shown that only some combination of the above presented options make sense and have argued that all options (except for LPF and LTPF) are useful in special networking scenarios.

4.3 State Reduction in MPLS Networks

Finally, we motivate that the state reduction by hierarchical flow aggregation can downgrade the scalability problem to a minor complexity class. If all possible e2e LSPs are established in a network, their number grows quadraticly with the network size. In case of a star network, all of them induce a state in the center router. However, a star network is not an average topology. Like above, we tested randomly constructed networks with different size to investigate the state reduction potential by tunnel aggregation.

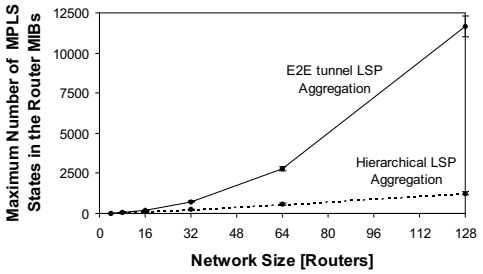

Figure 11: The number of MPLS states grows with the network size.

Figure 11 illustrates that without simple LSP aggrega-

tion, the number of states in the router MIBs still grows quadraticly. In contrast to that, hierarchically aggregated LSPs reduce the growth of that number to a linear one for the average of randomly constructed networks. For 128 routers, the maximum number of MPLS states is reduced by 90%.

LSP hierarchies are able to provide full e2e connectivity. These results show for that case that the number of reservation states in the router MIBs scales on average only rather linearly with the network size. This effect is enforced by suitable network topologies where transit routers offer no access capabilities. When reservations are made according to these LSPs, scalable resource management can be achieved by hierarchically structured overlay networks using tunnel LSPs.

5 Conclusion and Outlook

We gave a short introduction to IntServ and DiffServ and pointed out their shortcomings. We explained the scalability problem for e2e reservations due to per flow signaling and tackled this problem using reservation aggregation by tunnels and funnels. They are both supported in the data plane of MPLS by LSPs. We illustrated how such reservation can be signaled. MPLS-TE, however, supports only tunnel reservations in the control plane of MPLS and should be extended for funnels.

The main contribution of this paper is a graph-theoretical notation of hierarchically organized virtual links and flows and the formulation of technical constraints within the graph due to flow aggregation in the network. This concept is independent of any technology but we use it in the context of MPLS. We suggest a cost function to assess the quality of an LSP hierarchy with respect to scalability which leads to an optimization problem.

We propose several heuristics to find hierarchical LSP overlay networks to support e2e reservations. These algorithms use only tunnels for reservation aggregation and not funnels and their output can be immediately used for the configuration of an MPLS network. A comparison of simulation results shows considerable differences in aggregation performance among the different aggregation algorithms, however all the discussed options have their power in certain network topologies. The simulations also showed that due to the hierarchically structured overlay network, the maximum number of states in a router scales rather linearly than quadraticly with the network size in a fully connected average spanning tree network. Almost 90% of the reservation states can be saved in a network with 128 routers.

Currently, we investigate approaches that reduce the signaling amount for reservation aggregation which leads to a truly scalable resource management. To move from a topology driven LSP setup approach to a traffic driven LSP setup approach, we work on new signaling concepts using tunnel and funnels and combine them with traffic engineering methods.

References

[1] Bob Braden, David Clark, and Scott Shenker, "RFC1633: Integrated Services in the Internet Architecture: an Overview," http://www.ietf.org/rfc/rfc1633.txt, June 1994.

[2] John Wroclawski, "RFC2210: The Use of RSVP with IETF Integrated Services," ftp://ftp.isi.edu/in-notes/rfc2210.txt, Sep. 1997.

[3] Bob Braden et al., "RFC2205: Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification," ftp://ftp.isi.edu/in-notes/rfc2205.txt, Sep. 1997.

[4] Steven Blake et al., "RFC2475: An Architecture for Differentiated Services," ftp://ftp.isi.edu/in-notes/rfc2475.txt, Dec. 1998.

[5] Kathleen Nichols, Van Jacobson, and Lixia Zhang, "RFC2638: A Two-bit Differentiated Services Architecture for the Internet," ftp://ftp.isi.edu/in-notes/rfc2638.txt, July 1999.

[6] Manuel Günther and Torsten Braun, "Evaluation of bandwidth broker signaling," in *International Conference on Network Protocols ICNP'99*, Nov. 1999, pp. 145–152.

[7] Fred Baker, Carol Iturralde, Francois Le Faucheur, and Bruce Davie, "RFC3175: aggregation of RSVP for IPv4 and IPv6 Reservations," http://www.ietf.org/rfc/rfc3175.txt, September 2001.

[8] Ben Teitelbaum, S. Hares, L. Dunn, V. Narayan, R. Neilson, and F. Reichmeyer, "Internet2 QBone: Building a Testbed for Differentiated Services," *IEEE Network Magazine*, Sep. 1999.

[9] Olov Schelén and Stephen Pink, "Aggregating Resource Reservations over Multiple Routing Domains," in *IFIP* 6^{th} *International Workshop on Quality of Service (IWQoS'98)*, May 1998.

[10] Ping Pan and Henning Schulzrinne, "BGRP: A Tree-Based Aggregation Protocol for Inter-Domain Reservations," *Journal of Communications and Networks*, vol. 2, no. 2, pp. 157–167, June 2000.

[11] Daniel O. Awduche, Lou Berger, Der-Hwa Gan, Tony Li, Vijay Srinivasan, and George Swallow, "RFC3209: RSVP-TE: Extensions to RSVP for LSP Tunnels," http://www.ietf.org/rfc/rfc3209.txt, Dec. 2001.

[12] Bilel Jamoussi et al., "RFC3212: Constraint-Based LSP Setup using LDP," http://www.ietf.org/rfc/rfc3212.txt, Jan. 2002.

[13] Kireeti Kompella and Yakov Rekhter, "LSP Hierarchy with Generalized MPLS TE," http://www.ietf.org/internet-drafts/draft-ietf-mpls-lsp-hierarchy-06.txt, May 2002.

[14] Heinrich Hummel and Jochen Grimminger, "Hierarchical LSP," http://www.ietf.org/internet-drafts/draft-hummel-mpls-hierarchical-lsp-01.txt, May 2002.

[15] Sanghyun Ahn, Rose P. Tsang, Sheau-Ru Tong, and David H. C. Du, "Virtual Path Layout Design on ATM Networks," in *IEEE Infocom*, 1994, pp. 192–200.

[16] Ornan Gerstel, Israel Cidon, and Zaks Shmuel, "The Layout of Virtual Paths in ATM Networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 873–884, 1996.

[17] Shai Benjamin and Izhak Rubin, "Path Merging and Destination Removal: New Forwarding Paradigms in Packet Switching Networks," in *IEEE Globecom*, 2000, pp. 192–200.

[18] Eric C. Rosen, Arun Viswanathan, and Ross Callon, "RFC3031: Multiprotocol Label Switching Architecture," http://www.ietf.org/rfc/rfc3031.txt, Jan. 2001.