

Towards Scalable Management of QoS-based End-to-End Services

R. Bless

Institute of Telematics, University of Karlsruhe

76128 Karlsruhe

Germany

bless@tm.uka.de

Abstract

The Differentiated Services architecture provides basic building blocks for quality of service (QoS) support within the Internet. Definition of scalable QoS mechanisms in the data path was a main objective. However, in order to provide guaranteed end-to-end reservations on demand the management of highly dynamic resource reservations by control entities (e.g., those performing admission control) must be scalable as well.

This paper presents the DARIS architecture that provides a scalable management for QoS-based end-to-end services in the Internet. It defines a novel concept of dynamic and hierarchical aggregation of reservations, which operates on the network level of Autonomous Systems (ASes). The aggregation concept reduces the managed reservation states as well as the number of signaling messages that have to be processed by management entities in intermediate ASes. The aggregation concept was evaluated in detail using simulations including real Internet topologies.

Keywords

Quality of Service Management, Distributed and Scalable Management, Resource Management, Dynamic Aggregation, Signaling, Resource Reservation

1. Introduction

End-to-end communication services with quality of service (QoS) are not yet available globally and on demand in the Internet today. Though scalable solutions exist to support QoS-based communication services in the data plane, a corresponding control plane is still not available. A scalable resource management scheme for setting up and managing reservations between different administrative domains is one of the major missing pieces for global end-to-end QoS deployment.

The Differentiated Services (abbreviated as DiffServ or DS) architecture [2] proposes a scalable solution to the need for QoS support by several applications, and, will allow providers a more competitive differentiation of service offerings. The fundamental concept behind DiffServ is an *aggregation* of packet flows in the data path. Unavoidable complexity such as microflow classification and policing is pushed to the network edges while state in the network core is reduced by aggregation. Consequently, DiffServ can be considered as a basic scalable approach for QoS deployment in the data plane.

An important missing piece for globally available *on-demand services* with an assured QoS from end to end is a scalable management structure (a control plane) with a related signaling protocol to request such services from a domain. For the rest of the paper, it is assumed that end-to-end QoS is provided by a pure DiffServ architecture, although the presented solution may be applied to other QoS environments as well.

Some future services, such as those which offer a guaranteed bandwidth for an end-to-end flow based on the Expedited Forwarding Per-Hop-Behavior [4], require admission control from end to end. Particularly, this also includes the case where service requests have to be exchanged between domains. But per-flow admission control possesses obviously potential scalability problems in the control plane with respect to the numbers of exchanged messages and reservation states. Though many existing management approaches mainly concentrate on scalability within a single administrative domain only, few approaches address global scalability in a multi-domain case. But, as described later in section 4, existing approaches mostly cover special cases for aggregation. The *DARIS (Dynamic Aggregation of Reservations for Internet Services)* concept presented in this paper provides a more general and flexible aggregation scheme that reduces the number of required aggregates.

Section 2 starts with a description of the overall management architecture as well as an illustration of fundamental scalability problems. The DARIS aggregation principle is described in section 2.2. Simulation results are presented and discussed in section 3. Related work is summarized in section 4. The paper closes with a summary and an outlook on future work in section 5.

2. The DARIS QoS Management Architecture

2.1 Motivation and Basic Architecture

The DiffServ architecture provides scalable building blocks for QoS mechanisms in the data plane but does not define a control model for resource management. Firstly, traffic conditioners and DS nodes have to be managed by the administrative control of the domain. One possibility mentioned in [2] is operational control through a control entity and protocols for interaction and configuration (e.g., SNMP or COPS). Secondly, resources have to be managed in order to perform admission control and reservations along data paths within a domain. Requesting services on-demand requires also a signaling protocol from end-systems to control entities as well as between control entities.

Well-known examples for concepts using a dedicated control entity within a domain are “Bandwidth Brokers” [8] or “Reservation Agents” [11]. In these approaches, a logically centralized agent controls resources of a complete domain. It receives requests to provide a certain amount of resources for a particular data flow through its domain. The agent admits or rejects a new request based on its internal bookkeeping of reserved resources. Basically, this requires knowledge of the internal domain topology, resource capacities and currently selected routes. Though the agent is not directly located on the data path, the latter information can be provided by passively participating in link state-based intra-domain routing protocols (e.g., the widely used OSPF or IS-IS protocols) and an inter-domain routing protocol (e.g., BGP). If the agent made a positive intra-domain admission control decision he requests resources from the agent of the next adjacent DS

domain on the path to the flow's destination. Hence, it also performs inter-domain signaling and negotiation in order to set up a path of reserved resources from end to end. Upon successful resource negotiation, the agent installs or adapts already installed traffic profiles in boundary routers to activate the actual service. Usually, interior DS routers of a DS domain only need an initial configuration (so-called "resource partitioning") of their forwarding resources for different *per-hop behaviors (PHBs)*, and, thus, do not need to be involved in signaling or admission control functions for single or aggregated flows at all.

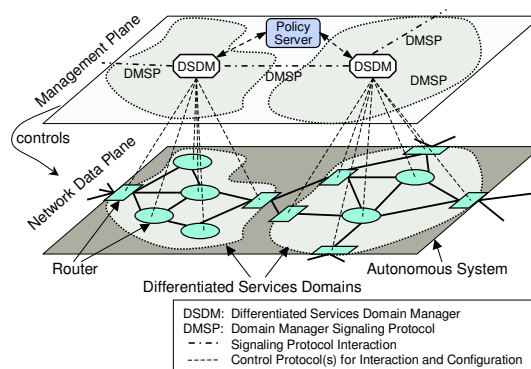


Figure 1: Separation of control and data planes

In the DARIS architecture, such a management entity or agent is called *Differentiated Services Domain Manager (DSDM)*. Figure 1 shows the basic architecture consisting of two planes: a control plane and a data plane. A larger *Autonomous System (AS)* may be sub-divided into several DS domains (usually corresponding to routing areas), each controlled by a single DSDM (essentially an extended Bandwidth Broker). Within a domain routers are controlled by the DSDM using the aforementioned protocols for interaction and configuration such as SNMP or COPS. Additionally, a policy server can be used to control resources of the whole AS. This allows arbitration if DSDMs share control of resources in parts of the network (e.g., a backbone area). DSDMs communicate with each other and with end-systems by the *Domain Manager Signaling Protocol (DMSP)*.

Though DiffServ prevents scalability problems in the data forwarding plane, scalability problems with respect to the number of states and signaling messages have to be expected in an associated control plane if admission control is performed per end-to-end flow. This is illustrated by the following example. If global telephone calls over IP use DiffServ (e.g., on an EF-basis) to ensure a common QoS level, resources for every call have to be admitted and reserved from end to end on demand. Figure 2(a) shows a simplified scenario (without AS interconnections) in which the forwarding of reservation request messages between ASes is sketched for three reservations. This should especially illustrate, that ASes which aggregate traffic (cf., AS A and AS B) have to process the sum of all signaling messages and to keep state for every reservation. Hence, per-microflow reservations are usually not considered to be scalable.

Scalability problems in the control plane can be solved by using aggregation as well. For instance, the DSDM in AS A may decide to combine the three existing reservations

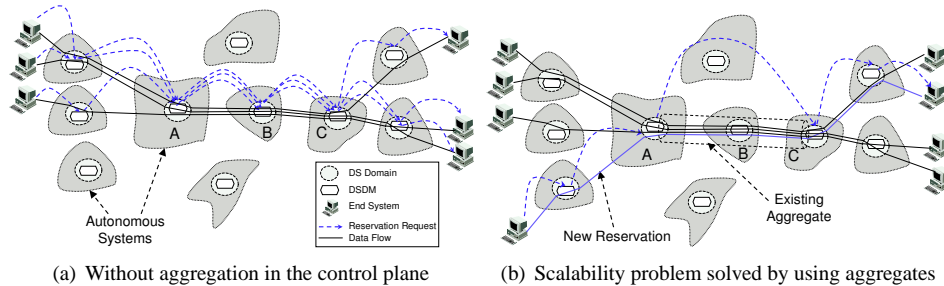


Figure 2: Scalability problems

into a single aggregate. In this case, the aggregate spans ASes A , B and C on the common partial path of all three reservations. Consequently, the DSDM in AS B can delete three established states for the single reservations but has to keep a single state for the new aggregate. If there is enough capacity in the already established aggregate to integrate an additional flow, a new reservation that traverses the same partial path can be included without any additional overhead. Moreover, the DSDM in AS A can directly forward the reservation request to the aggregation endpoint in AS C , thus AS B is also released from processing the signaling messages.

2.2 Hierarchical Dynamic Aggregation of Reservations

The now presented hierarchical dynamic aggregation principle is more flexible than other existing approaches that are summarized in section 4. This is due to the fact that it allows creation of aggregates between arbitrary domains and also permits to nest aggregates in a fully hierarchical manner.

First of all, the unit of aggregation has to be determined. The flexibility of the Diff-Serv architecture allows every provider to use different codepoints and PHBs to realize a certain *per-domain behavior* (PDB). Thus, various codepoints and PHBs can be used in and between different domains along a path from end to end. However, for end-to-end service provisioning multiple domains have to agree on a common service definition and its guaranteed parameters. A simple example is a service that offers a guaranteed bandwidth within a specific time interval. Because codepoints, PHBs and PDBs may differ from domain to domain, a useful unit of aggregation is a concertedly defined unicast service with its specific parameters and related resources (e.g., bandwidth). This service, however, must be mapped to a corresponding PDB.

In this context, it also has to be mentioned that the described aggregation of resource reservations is different from possibly existing DS aggregates between adjacent domains. Usually these aggregates have to be maintained independently of the reservation aggregates, because the actual type of DS aggregates depends on the provisioning policy of the providers. Particularly, the proposed resource aggregation scheme requires no additional mechanisms (e.g., classification) in the data path.

Establishment of Aggregates

As mentioned before, aggregation of reservations is possible between arbitrary DiffServ domains. It is assumed that every DSDM maintains an internal graph representation of the Autonomous System topology. In this graph, a *node* represents an AS and an *edge* a link between ASes. The graph can be constructed with the help of the inter-domain

routing table (e.g., as passive BGP peer). A DSDM records every established reservation in this graph along the reservation’s path, i.e., every graph node also contains a set of reservations that traverse this AS or end in it. Surely, this graph represents only the view of this specific DSDM, therefore, it merely sees directed paths from its own node to other ASes, and, it just records reservations for which it receives requests. Due to the passive coupling to the inter-domain routing a DSDM can detect and handle inter-domain routing changes. For the sake of a clearer description, in the rest of the paper the special case is assumed that only one DSDM exists per AS.

Basic Operation

When a new reservation request is received, the DSDM checks whether other reservations exist already along the path towards the destination. Thus, the number of existing reservations is checked in each AS along the reverse path, i.e., backwards from the destination to the own AS. If there exist at least k reservations, a new aggregate can be created. This aggregate subsumes the capacity of k reservations plus the resources of the newly requested reservation. Furthermore, the aggregate’s capacity can be increased by an additional amount in order to include future reservations without having to increase the capacity first. The additional unused capacity can nevertheless be utilized by low priority services such as the traditional best-effort delivery. By starting from the destination network longer aggregates should be created at first, if possible. This achieves a better reduction of states and signaling messages.

However, only aggregates of at least two hops make sense. Creating a reservation aggregate between two directly adjacent ASes does not provide any benefit. Therefore, an aggregate must comprise at least one more AS besides starting point and end point. Furthermore, there exists at most one aggregate between a pair of ASes that describes starting point and end point of the aggregate.

The aforementioned directed graph $G = (V, E)$ with $V := \{v | v \text{ is a node (AS)}\}$ and $E := \{(u, v) | u, v \in V\} \subseteq V \times V$ represents the Internet topology at AS level. A *path* p between v_0 and v_l of length l is defined as sequence of edges $(v_0, v_1), \dots, (v_{l-1}, v_l)$ with $(v_i, v_{i+1}) \in E$ for $i = 0, \dots, l - 1$. In the following, the notation $A_p(u, w)$ denotes an aggregate along path p with node u as starting point and node w as end point of the aggregate. The value of $A_p(u, w)$ is defined as the capacity of the aggregate (e.g., bandwidth). $\Sigma(A_p(u, w))$ denotes the sum of resources of all flows that are aggregated in $A_p(u, w)$, whereas $\Delta(A_p(u, w)) := A_p(u, w) - \Sigma(A_p(u, w))$ denotes the amount of “unused” resources in the aggregate $A_p(u, w)$, i.e., these resources are currently not reserved by aggregated flows.

Because resources for the k reservations were already allocated, the aggregate is installed along path p by allocating the capacity that is required for the new reservation and the additional capacity $\Delta(A_p(u, w))$. If this allocation is successful, the states of aggregated flows are deleted between, but not including, starting point u and end point w .

Figure 3 illustrates the aggregation process by a simple example. Figure 3(a) shows an AS graph in which each bullet represents an AS (bullets between u and v as well as between v and w have no assigned letter). A new reservation is requested for a flow between u and w . Upon receiving the new reservation request the DSDM in AS v decides to create an aggregate along path p' . Figure 3(b) shows the new requested reservation between u and w as well as the three already existing reservations (between AS pairs

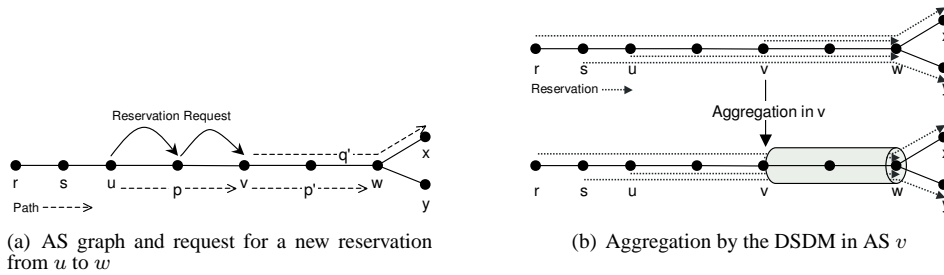


Figure 3: Creation of an aggregate

(r, x) , (v, w) and (s, y)). Thus, if the aggregation threshold is $k = 3$ the DSDM in v requests from its downstream adjacent DSDM to create an aggregate $A_{p'}(v, w)$ along path p' that includes the four reservations.

Nested Aggregates

As mentioned before, in the DARIS approach, aggregates can be hierarchically nested. Two types of nested aggregates are depicted in figure 4. If two aggregates have the same starting point but different end points (cf., fig. 4(a)) the nested aggregates are both visible within a DSDM at the starting point. In the following, the larger (with respect to capacity) but shorter aggregate is also denoted as “encompassing aggregate” of the embedded longer aggregate. Because there exists at most one aggregate between a pair of two ASes, every aggregate has at most one encompassing aggregate and at most one directly embedded (longer) aggregate (that may include another aggregate in turn, and so on). Consequently, all nested aggregates that start at the same AS have a different length, whereby every embedded aggregate is at least one AS hop longer than its encompassing aggregate. In other cases, aggregate nesting is not visible for a starting point of an aggregate, if other DSDMs downstream embedded this aggregate into a larger one (cf., fig. 4(b)). Thus, an aggregate may be contained theoretically in several other aggregates at different ASes.

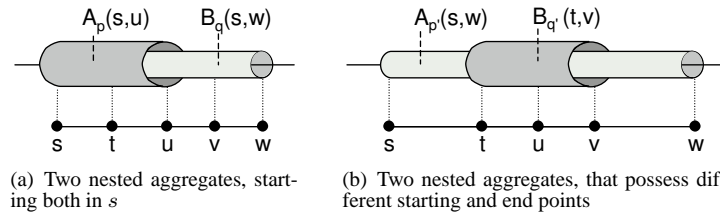


Figure 4: Different kinds of nested aggregates

Hierarchical aggregates are important in order to let longer aggregates (which are more efficient) evolve over time and to preserve the independence of the aggregation decision within ASes. Shorter aggregates allow an earlier aggregation to reduce the number of managed states within core transit networks whereas longer aggregates will save more states and messages. The latter allow to shift reservations from shorter aggregates to longer ones in most cases. However, it is necessary to have a closer look at different situations at aggregate creation.

Overlapping Aggregates

Several cases (which are illustrated in figure 5) have to be carefully considered when new aggregates are created. It is assumed that aggregate $A_p(u, w)$ is already established when a new aggregate $B_q(r, x)$ is going to be created. The latter will aggregate some reservations that were already aggregated in $A_p(u, w)$.

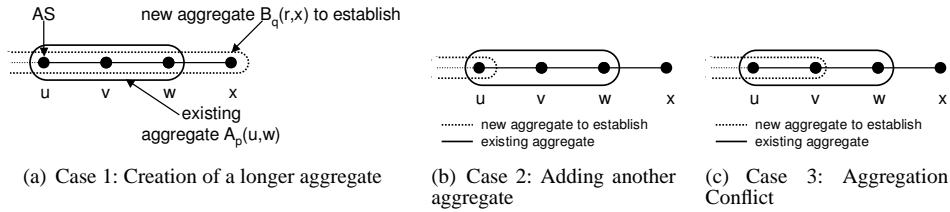


Figure 5: Different cases of creating a new aggregate

In case 1 (cf., fig. 5(a)) a longer aggregate is going to be created (starting point r lies before u and end point x behind w , as a special case x may even fall together with w). In this case all reservations of $B_q(r, x)$ that are already aggregated in $A_p(u, w)$ should be shifted to the new and longer aggregate. The new aggregate B_q will be integrated into the existing aggregate A_p . In case 2 (cf., fig. 5(b)) the new aggregate ends at the starting point u of the already existing aggregate $A_p(u, w)$. Consequently, at AS u some reservations are aggregated simultaneously in both aggregates. But in case 3 (cf., fig. 5(c)), where the new end point x falls into aggregate A_p (excluding end points u and w), the creation of the new aggregate will not be possible: if $B_q(r, v)$ contains already aggregated reservations, they are not known at the DSDM in v any longer, because all state information was deleted by the former aggregation process. This situation is called *aggregation conflict*. In this case, the DSDM in u must reject a request to form the new aggregate B_q . However, a new aggregation attempt can be made without the reservations that are affected by the conflict. Thus, the DSDM that initiated the aggregation excludes them from its internal list of potential aggregation candidates at AS v . Please note that reservations starting at v or w (or traversing them, but not u) do not cause conflicts with respect to $A_p(u, w)$, because they cannot be included in it.

Furthermore, there is a small chance that aggregation conflicts occur between nested longer aggregates and some reservations due to an earlier conflict during establishment of the enclosing shorter aggregate. Figure 6 shows an example for this kind of “indirect conflicts”. The situation of an earlier conflict is shown in figure 6(a), where the reservation for flow $f_o(r, x)$ was included into aggregate $A_p(t, v)$. It is assumed that afterwards, during creation of aggregate $B_q(s, u)$ a conflict with respect to f in u occurred, i.e., f could not be included into $B_q(s, u)$. This situation may lead later on to an indirect conflict for a new aggregate $C_{q'}(s, w)$ that wants to include $f_o(r, x)$ (cf., fig. 6(b)): f cannot be included into $C_{q'}(s, w)$ due to the earlier conflict with $B_q(s, u)$. As simulations showed, these “indirect conflicts” are rare events, but nevertheless have to be handled by DSDMs. When checking aggregation possibilities a DSDM simply verifies for every reservation that is going to be aggregated if there was already conflict at the endpoint of the encompassing aggregate. If this is true the reservation cannot be part of the potential aggregate.

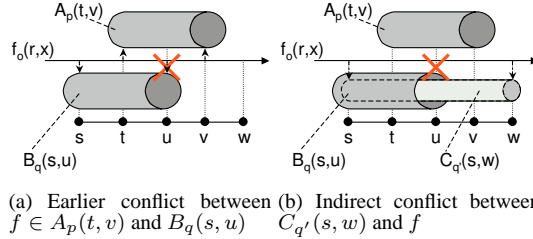


Figure 6: Indirect conflicts during aggregate nesting

In summary, every DSDM makes its own autonomous decision when and where to aggregate. Therefore, the autonomy of the service providers is kept. It is possible that some reservations encounter conflicts during the aggregation process. In this case, reservations that are affected by a conflict cannot be aggregated along a certain path. But it is often possible to establish another aggregate including a different set of reservations.

Change and Termination of Aggregates

The capacity of an aggregate may need to be changed over time. Especially if a new reservation should be integrated into an already existing aggregate, the aggregate's capacity has to be increased first. In order to minimize aggregate changes and to save signaling messages, the requested additional amount should be larger than required, i.e., $\Delta(A_p(u,w)) > 0$. This allows to integrate future reservations without having to increase the encompassing aggregate's capacity first.

Similarly, if an aggregated reservation is released, the aggregate's capacity should not be decreased immediately. In this case a hysteresis function can be applied, to adapt the capacity only in larger intervals and bigger steps. Thus, the capacity of the aggregate should be adapted on a coarser time scale, e.g., in some few minutes. An example for such an algorithm is given in the next section (cf. figure 9(a)).

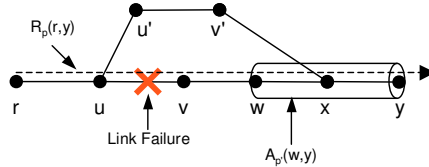


Figure 7: Inter-domain routing changes may cause new conflicts

Inter-domain route changes are not straightforward to handle, because they might cause new conflicts. Figure 7 illustrates this situation. Reservation $R_p(r,y)$ is re-routed via u' and v' due to a link failure between u and v . Unfortunately, in this case the new route ends within an existing aggregate $A_p'(w,y)$ that aggregated R_p , too. Because the DSDM in x already deleted all state information about R_p it cannot address R_p and perform admission control again (necessary due to new ingress). Instead of applying more complex re-reservation schemes, the DARIS approach reduces this situation to a simple case: the reservation is aborted and has to be established again from end to end along the complete new path. However, it must be noted that stable routes are a precondition for providing stable QoS, because a route change requires re-reservation of resources, that may even fail.

Aggregates are terminated automatically some period after the last reservation was released. This period is usually in the same time domain as the previously mentioned changing intervals.

3. Evaluation

First of all, the applicability of the hierarchical aggregation principle was examined. It had to be investigated whether current AS paths in the Internet are long enough for aggregation. For this purpose, several BGP table dumps from the routeviews project [9] were used and converted into a graph representation. Subsequently, for every not pure transit AS shortest path lengths to every other AS were calculated and counted. The result showed that more than 92,16% of all paths are *longer* than two hops, i.e., in 92,16% of all cases an aggregate can span four or more ASes (saving states and message processing in at least two ASes). Though the used BGP tables may not reflect the complete real topology [3], it can be inferred that enough aggregation possibilities exist in the real Internet.

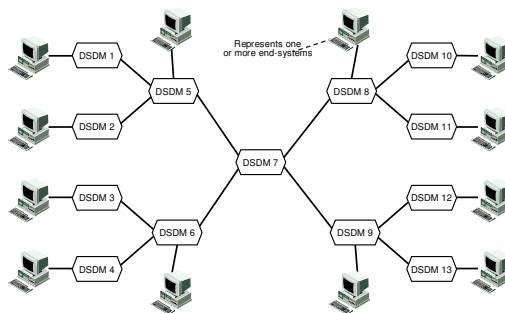


Figure 8: A simple simulation topology

The DARIS principle was simulated by using the C++-based OMNeT++ simulation environment that is freely available for academic research. In order to investigate and evaluate the previously described aggregation concept a simple topology as depicted in figure 8 was used at first (an evaluation within a realistic Internet topology is described later). This topology consists of 13 DSDMs, where DSDM 5 has to handle messages from DSDMs 1 and 2, and, DSDM 6 has to handle messages from DSDMs 3 and 4 respectively. Similarly, DSDM 7 has to manage reservations from DSDM 5 and 6. The right side is symmetric to the left side, so analogous statements are valid for DSDMs 8–13. Every DSDM, except DSDM 7, has an end-system that creates randomly reservation requests to every other end-system with equal probability. This represents the worst case for aggregation because every other distribution would yield even better aggregation possibilities. Furthermore, DSDM 7 will be stressed by handling the most reservations. Link failures or route changes were not simulated, because the main focus laid on scalability.

Since on-demand services with guaranteed QoS are not available yet, recommendations from [7] were considered and led to the conclusion to assume a Poisson process for session interarrival (due to independency of users) and a Pareto distributed session duration. The initial example from section 2 of providing QoS for phone calls was also considered so every reservation requested a constant bandwidth of 64 kbit/s. The time

until a new request is issued at an end-system is exponentially distributed, while the duration of a “session” (which denotes an established reservation) is chosen from a pareto distribution $F(x) = P[X \leq x] = 1 - (b/x)^\alpha$ with a mean duration of 180 s by using $F(x) - 0.99b$ ($\alpha = 1.7, b = 126$).

The primary objective of the simulations was to examine the scalability property of the DARIS aggregation principle, but not to find an optimum small amount $\Delta(A_p(u, w))$ of “unused” resources (which can be nevertheless utilized by traditional best-effort services). Therefore, simple heuristics are used for the adaptation of an aggregate’s capacity. An exponential moving weighted average (EWMA) as well as a smoothed standard deviation is used to track the development of $\Sigma(A_p(u, w))$. Furthermore, a smoothed gradient as well as quantization is used to calculate the additional amount of resources that should be used for incrementing the aggregate’s capacity. Quantization and the average value are also used for the hysteresis when the aggregate’s capacity has to be decremented. The main objective was to develop an effective decoupling between reservations and adaptation of the aggregate capacity rather than to design an optimized algorithm for minimizing unused capacity and adaptation frequency. An example of the adaptation of an aggregate’s resources at one DSDM is given in figure 9(a). The rectangular curve shows the capacity that was reserved for the aggregate. The thicker curve below shows the smoothed average of the capacity that is currently used by the included reservations (brighter jagged line).



Figure 9: Simulation results

The simulations ran with different reservation initiation rates: each end-system initiated 1000, 10 000, 50 000, 100 000, 500 000 and 1 000 000 sessions with an associated increase of mean rates of 0.05, 0.5, 2.5, 5, 25, and 50 sessions (or reservations) per second. Every configuration was run with 100 different random seeds of the OMNeT++ built-in linear congruential generator with a cycle length of $2^{31} - 2$. An OMNeT++ tool generated the seeds in order to guarantee that the seed values were far enough apart. Furthermore, the minimum number of reservations that must exist before an aggregate can be created (aggregation threshold k) was varied over the values 2, 4, 8, and 100.

First of all, the total number of transit states per DSDM (i.e., the sum of all transit states after a run) is shown in figure 9(b) for $k = 2$. Transit states stem from reservations

that do not start or end within the own domain. Keeping states for reservations that are initiated or released by end-systems of the own domain cannot be avoided. Consequently, DSDMs 1–4 and 10–13 do not have to keep any transit states, because they are located in stub domains. The diagram shows the results (average of 100 simulation runs) for the configuration 10^6 sessions per end-system with a mean rate of 50 initiated sessions per second. Because the total number of transit states is shown, the result for DSDM 7 includes also the transit reservations before the aggregation started. The diagram shows that the proposed aggregation principle can save several orders of magnitude of states.

Figures 10(a) and 10(b) show the behavior of DARIS within DSDMs 6 and 7 for the different configurations and $k = 2$. The plotted total number of states on the ordinate (logarithmic scale) reflects the sum of states over the complete runtime of the simulation. The mean values of the 100 different runs are shown as well as the absolute deviation. Because a pareto distribution was used for the duration, normality of the mean values was checked by a Ryan-Joiner test. The calculated regression coefficients gave no reason to reject the hypothesis of normality. Consequently, it was assured that the results were following a normal distribution so that a confidential interval could be determined. The 99.99% confidential interval is located very close around the average value.

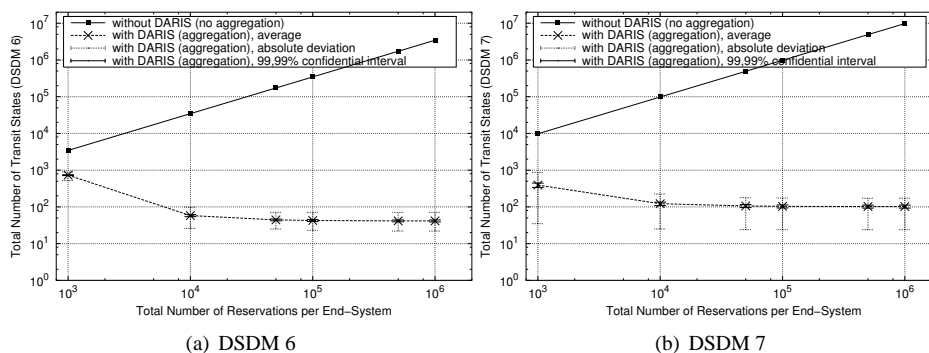
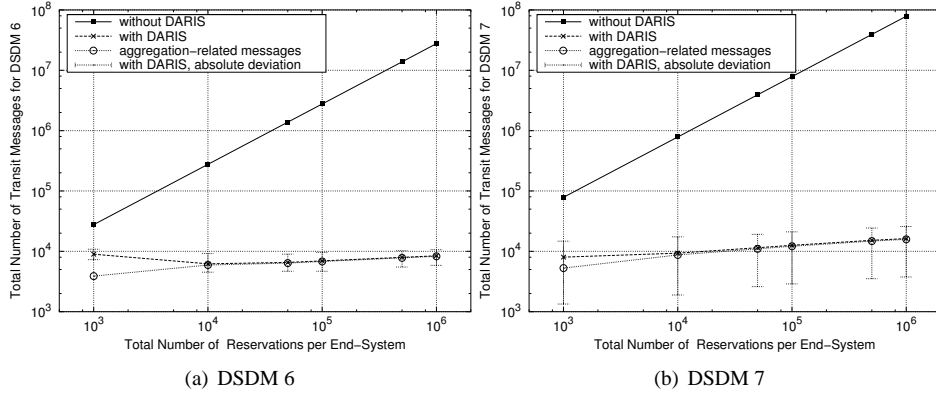


Figure 10: Total number of reservation states for $k = 2$

Due to the symmetric topology results for DSDMs 5, 8 and 9 are similar to those of DSDM 6. In summary, the overhead due to own and foreign aggregates (aggregates initiated by other DSDMs are contained in the number of transit reservations) is very small and the number of states can be reduced to a scalable and manageable amount with the DARIS concept.

The influence of the aggregation threshold k was also examined. While runs with $k \in \{2, 4, 8\}$ show similar results, significant state reduction for $k = 100$ is only achieved in cases of higher reservation initiation rates. This is due to the fact that the high reservation threshold is only reached if there exist simultaneous reservations along the same path. On the one hand, it follows that a reservation threshold should be kept low to allow early aggregate formation. On the other hand it confirms that aggregation works even better in highly dynamic environments with high intensity of reservation fluctuation.

In the next step, the total number of signaling messages has to be examined. Figures 11(a) and 11(b) depict the number of transit messages and aggregation-related signaling



(a) DSDM 6 (b) DSDM 7

Figure 11: Total number of transit messages $k = 2$

messages (averaged over 100 runs). Similarly, DARIS reduces the number of resource-related signaling messages to a scalable amount. The overhead caused by aggregation is small compared to the number of transit messages without aggregation. When aggregation is used, it can be observed that most transit messages are aggregation-related, because the difference between both curves is very small (at logarithmic scale). Moreover, it is obvious that with an increasing number of sessions the total number of aggregation-related messages must also increase slowly: the aggregates have to be adapted more often over the longer runtime. Moreover, simulations showed also that aggregation conflicts occur only in early phases of aggregate creation. After most aggregates have been established the situation remains mostly stable.

The simple topology was small enough to examine and understand the aggregation principle in detail. Thereafter, simulations comprising a *real Internet topology* were carried out. This should be a lot more real than any randomly and artificially generated “Internet-like” topology. Again, the routeviews archive of BGP table dumps [9] was used to extract an undirected AS graph. Due to memory limitations of the simulation hardware platform only a topology of 1999 (of August 18th) could be used so far. However, some of the fundamental properties [6] should be already present. The topology consists of 5587 ASes of which 4222 are stub ASes, i.e., they are not able to (as single-homed ASes) or are not configured to forward transit traffic. Because 64 ASes were classified as pure transit ASes, all other 5523 ASes had end-systems to initiate reservations. Every end-system initiated 5000 reservations with a mean rate of one 64 kbit/s reservation every 3 seconds. k was set to 2 and the same parameters for the pareto distribution as within the small topology were used (mean duration of 180 s, simulated time: roughly 4 hours).

Figure 12(a) shows the results of such a simulation. It displays the total number of transit states for reservations. The DSDMs were sorted with decreasing number of states (the abscissa shows a such sorted DSDM index). Due to the high number of stub ASes, only 1365 ASes carry transit traffic. Again, these simulations underline that improvement of one to two orders of magnitude can be achieved with DARIS. Only DSDMs with 100 or less states observe additional overhead caused by aggregation. However, in this order of magnitude the latter does not impose real problems. Figure 12(b) shows similar results for the total number of transit messages.

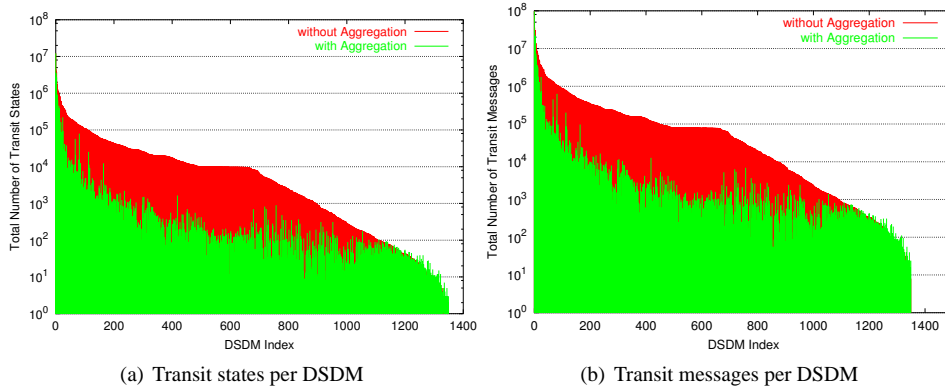


Figure 12: Evaluation in an Internet topology from 1999/08/18 (more than 5500 nodes)

In this scenario a maximum nesting level of 3 (i.e., 4 nested aggregates) could be observed which is one level more than in the previously described small topology. Moreover, only 25 indirect conflicts were counted. Several runs showed comparable results. Because of the equally distributed reservations, there are barely established aggregates from end to end. Therefore, some highly connected ASes can reduce their transit states only to a very small degree, because there are no aggregates that include them as intermediate ASes. Splitting such large ASes internally into separate domains with several DSDMs can significantly reduce the load by distribution and achieve the desired scalability again.

4. Related Work

Other approaches that considered scalability in the inter-domain case mainly concentrated on aggregation along sink-based trees (e.g., Reservation Agents and BGRP) [11, 10], i.e., aggregation towards the same destination (network or end-system). Though the latter is a viable approach for routing and packet forwarding, it represents a rather rare situation for resource reservations in which many end-systems or sources require reserved resources towards the same receiver or destination network. The DARIS approach is much more flexible, because it can establish aggregates along shared AS path segments out of reservations with different source and destination ASes. The same limitations apply to AQUILA [5], because it uses a BGRP derivative. The SICAP approach [13, 12], however, extends the BGRP concept by “shared-segment aggregation” thus overcoming the limitation of sink-based aggregation. In [13, 12] it is shown that shared-segment aggregation outperforms sink-based aggregation with respect to the number of managed states. In contrast to the herein described DARIS approach, SICAP does not allow to nest aggregates (they are only concatenated), and, it uses a different algorithm to decide on how to aggregate. But hierarchical aggregation allows to create longer aggregates and will thus save even more state information.

As a router-based management approach RSVP aggregation [1] does not define the actual locations of aggregator and deaggregators and is a more static than dynamic approach. The QBone Bandwidth Broker architecture [14] shows similarities with the DARIS architecture, too. It also allows aggregation (“core tunnels”), but only from edge network to edge network. Hierarchical aggregation was not elaborated or considered in detail.

5. Summary and Outlook

Dynamic Aggregation of Reservations for Internet Services (DARIS) applies aggregation techniques in the control plane to achieve the necessary scalability, especially for inter-domain reservations. It allows a very flexible, fully hierarchical, scalable and efficient aggregation of resource reservations. A careful analysis revealed a few special cases that have to be considered at aggregate creation. The simulations confirm that DARIS reduces the number of reservations states and messages considerably, even if applied in real Internet topologies. Hence, DARIS is an important element to enable a scalable management of on-demand reservations with QoS guarantees per end-to-end flow.

In mobile environments, the QoS management must adapt reservations to changing data paths. Therefore, a special support for anticipated inter-domain handovers as well as for micro-mobility is currently added to the DARIS architecture. Anticipated handovers allow to reserve resources along potential new paths in advance, so that QoS can be provided more seamlessly for handovers.

Moreover, further efforts are made to examine the behavior of the DARIS concept for more current Internet topologies. A lot of memory is required for such large “real” topologies, because every DSDM has to manage a routing table with paths to every other AS. Currently, the simulation is ported to 64-bit platforms due to the 4 GB limit per process of traditional 32-bit platforms.

A general issue with aggregation is that not the aggregating and de-aggregating ASes profit from their initiated aggregates, but all intermediate ASes within an aggregate. Therefore, some incentive for aggregate creation has to be given. This may lead to novel cost models that have to be developed for aggregation concepts in the future.

References

- [1] F. Baker, C. Iturralde, F. L. Faucheur, and B. Davie. Aggregation of RSVP for IPv4 and IPv6 Reservations. RFC 3175, Sept. 2001.
- [2] S. Blake et al. An Architecture for Differentiated Services. RFC 2475, Dec. 1998.
- [3] Q. Chen et al. The Origin of Power Laws in Internet Topologies Revisited. In *Proceedings of INFOCOM 2002*, volume 4, Piscataway, NJ, USA, June 2002. IEEE.
- [4] B. Davie et al. An Expedited Forwarding PHB (Per-Hop Behavior). RFC 3246, Mar. 2002.
- [5] T. Engel et al. AQUILA: Adaptive Resource Control for QoS Using an IP-Based Layered Architecture. *IEEE Communications Magazine*, 41(1):46–53, Jan. 2003.
- [6] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On Power-Law Relationships of the Internet Topology. In *Proceedings of SIGCOMM 1999*, pages 251–262. ACM, Sept. 1999.
- [7] S. Floyd and V. Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, Aug. 2001.
- [8] V. Jacobson, K. Nichols, and L. Zhang. A Two-bit Differentiated Services Architecture for the Internet. RFC 2638, July 1999.
- [9] D. Meyer. Route views project page. <http://www.routeviews.org/>, May 2002.
- [10] P. Pan, E. L. Hahne, and H. Schulzrinne. BGRP: Sink-Tree-Based Aggregation for Inter-Domain Reservations. *Journal of Communications and Networks*, 2(2):157–167, June 2000.
- [11] O. Schelén and S. Pink. Aggregating resource reservations over multiple routing domains. In *Proceedings of IWQOS'98*. IFIP, IEEE, May 1998.
- [12] R. Sofia, R. Guerin, and P. Veiga. An investigation of inter-domain control aggregation procedures. In *10th IEEE ICNP'02*, Nov. 2002. Paris, France.
- [13] R. Sofia, R. Guerin, and P. Veiga. SICAP, A Shared-segment Inter-domain Control Aggregation Protocol. In *HPSR'03*, June 2003. Torino, Italy.
- [14] B. Teitelbaum and P. Cimento. QBone Bandwidth Broker Architecture. <http://qbone.internet2.edu/bb/bboutline2.html>, June 2000.