

Dynamic Aggregation of Reservations for Internet Services

Roland Bless

Institute of Telematics
University of Karlsruhe
76128 Karlsruhe, Germany
Tel.: +49 721 608 6413
Email: bless@tm.uka.de

Abstract—Global availability of on-demand services with an associated guaranteed quality-of-service (QoS) is still missing in the Internet today. The Differentiated Services architecture achieves scalability in the data plane by treating all flows with a specific type of service as one aggregate. In order to provide guaranteed end-to-end reservations, the control plane (e.g., performing admission control and management of resource reservations) has to be scalable as well.

The DARIS Architecture provides this essential linkage between scalability in the packet forwarding path and scalability of QoS management in the Internet. It comprises a novel concept of dynamic and hierarchical aggregation, which acts on the network level of autonomous systems (ASes). This relieves management entities in intermediate ASes of processing load by reducing their managed reservation states as well as the number of signaling messages that have to be processed significantly. Moreover, novel and special support for aggregation by a dedicated signaling protocol mechanism is provided.

Index Terms—Differentiated Services, Resource Management, Reservation Aggregation, Signaling.

INTRODUCTION

The Differentiated Services (often abbreviated as DiffServ or DS) architecture [1] promises a scalable solution to the concurrent need for quality of service (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: in contrast to the former Integrated Services (IntServ) approach [2], where every router distinguishes between individual end-to-end flows (i.e., a so called micro-flow that is characterized by protocol, source and destination IP addresses and port numbers), DiffServ routers in the core network differentiate only between different forwarding classes which are determined by a value in each packet's header [3]. Unavoidable functional complexity such as microflow classification and policing is pushed to the network edges. The number of forwarded flows at the network edge is not as high as in transit networks located in backbone and inner areas of the Internet, where many flows are aggregated into larger flows.

It is envisioned that an early deployment of DiffServ will mainly use statically provisioned resources at first,

whereas dynamically reserved resources for services should be added later on [1]. However, a major missing piece for globally available *on-demand* services with an assured quality-of-service from end to end is a scalable management structure with a corresponding signaling protocol for requesting such services (a control plane) from a domain of a provider. Though there exist schemes for integration of DiffServ and IntServ [4] it is assumed that QoS is provided by a pure DiffServ architecture from end to end for the rest of the paper. The overall aggregation scheme, however, is also applicable to other architectures (e.g., router-based QoS architectures or MPLS).

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 [5], require admission control from end to end. Especially, this also includes the case where service requests have to be exchanged between domains. But per-flow admission control obviously possesses potential scalability problems in the control plane with respect to the numbers of exchanged messages and reservation states. While many existing management approaches mainly concentrate solely on scalability within a single DiffServ domain, only few approaches address global scalability in a multi-domain case. But, as described later, existing approaches are not flexible enough.

In this paper, an approach for a scalable and integrated management architecture is presented. The emphasis of this paper lies on the *DARIS (Dynamic Aggregation of Reservations for Internet Services)* concept to achieve the required global scalability, and, on a related suitable and optimized signaling protocol for this purpose.

Section I describes the overall management architecture as well as fundamental scalability problems in the control plane. The DARIS aggregation principle is described in section I-A. Section II presents the novel signaling protocol DMSP that has special support for aggregation. Simulation results are presented and discussed in section III. The paper closes with a summary and outlook on future work in section IV.

I. THE DARIS MANAGEMENT ARCHITECTURE

The DARIS management architecture follows others approaches that are based on a separate resource manage-

ment entity within a DiffServ domain. Letting dedicated entities instead of routers perform most resource management functions, such as admission control, has several advantages: first, routers are relieved from carrying out these functions in addition to routing and packet forwarding functions. Second, more complex functions such as policy-based admission control or functions that require keeping persistent state (e.g., accounting-related functions) should not or cannot be built into routers due to lack of processing performance for these tasks or limitations of storing persistent states. Third, it allows a clear and better separation of control functions and data packet forwarding functions which can evolve independently from each other (in analogy to the separation of routing and forwarding functions in routers). Usually, interior DS routers of a DiffServ 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.

Well known examples for these approaches are the “Bandwidth Broker” approach [6] or “Reservation Agents” [7]. In these approaches, a logically centralized agent controls resources of a whole domain. It receives requests to provide a certain amount of resources for a particular data flow through its domain. Based on its internal bookkeeping of reserved resources this agent admits or rejects the new request. Basically, this requires knowledge of the internal domain topology, resource capacities and current selected routes. The latter can be provided by passively participating in link state-based interior routing protocols (such as widely used OSPF or IS-IS protocols) and BGP. In case the agent made a positive intra-domain admission control decision it requests resources from the agent of the next adjacent DiffServ domain on the path to the flow’s destination. Thus, 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, it creates or adapts traffic profiles in boundary routers accordingly in order to activate the actual service.

In the DARIS architecture, such a management entity or agent is called *Differentiated Services Domain Manager (DSDM)*, because it performs several additional functions besides “bandwidth brokerage”, which are not described in this paper, because it is focused on aggregation and signaling issues. The basic two plane architecture is shown in figure 1. It is assumed that a larger Autonomous System (AS) may be sub-divided into several DiffServ domains (usually corresponding to routing areas), each controlled by a single DSDM. A policy server may be used to control resources of the whole AS, especially in order to arbitrate resources between DSDMs in parts of shared control such as common backbone areas. DSDMs communicate with each other and with end-systems by a signaling protocol that is called *Domain Manager Signaling Protocol (DMSP)*. Some of its special features are described later in section II.

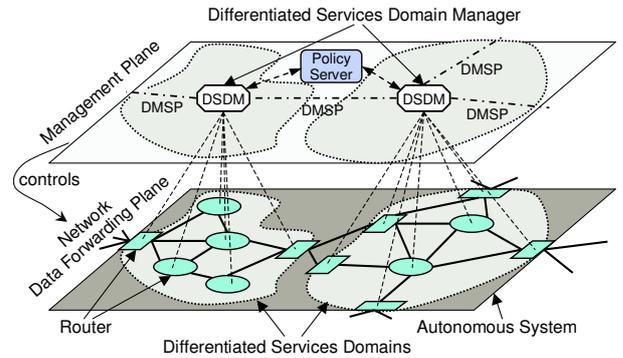


Fig. 1. Separation of control and data forwarding level

Although 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 the *control plane*. This is illustrated by the following example. If one imagines that global telephone calls over IP use Differentiated Services (e.g., some EF-based service) to ensure a common QoS level, resources for every call have to be admitted and reserved from end to end on demand. As shown in figure 2, those ASes which aggregate traffic (cf. ASes A and B) have to process the sum of all signaling messages and to manage state for every reservation. Therefore, per micro-flow reservations are usually not considered to be scalable.

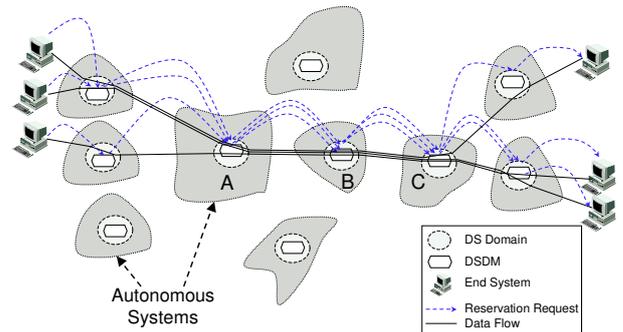


Fig. 2. Scalability problem in the control plane

The scalability problems in the control plane can also be solved by using aggregation. For instance, the DSDM in AS A decides to combine the three existing reservations into a single aggregate. In this case, the aggregate spans ASes A, B and C, that lie on the common path of all three reservations. The DSDM in AS B can now delete the three established states for the single reservations and has to manage a state for the new aggregate. A new reservation that traverses the same partial path can then use the already established aggregate if there is enough capacity in it to integrate the additional flow. DSDM in AS A can directly forward the reservation request to the aggregation endpoint in AS C, therefore AS B can also save processing of the signaling messages.

Other approaches that considered scalability in the inter-domain cases concentrated mainly on aggregation along sink-based trees [8], [9], i.e., aggregation towards

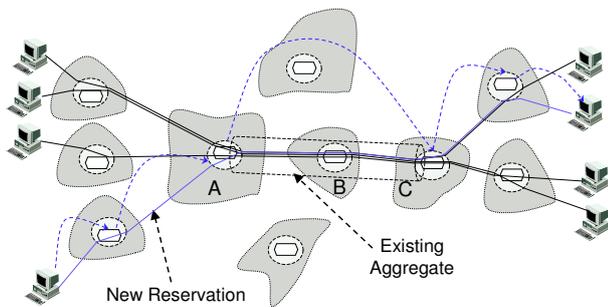


Fig. 3. Scalability problem solved by using aggregates

the same destination (network or end-system). The latter is a rather rare situation where many end-systems require reserved resources towards the same receiver or destination network. The QBone Bandwidth Broker architecture [10] also allows aggregation, but only from edge network to edge network, and hierarchical aggregation was not considered.

The DARIS aggregation principle, that is presented in the next section, is more flexible, because it allows creation of aggregates between arbitrary domains and also allows to nest aggregates in a fully hierarchical manner.

A. The DARIS aggregation principle

First of all, it has to be discussed *what* can be aggregated. The flexibility of the DiffServ architecture allows every provider to use different codepoints and PHBs to realize a certain per-domain behavior (PDB). For end-to-end service provisioning multiple domains have to agree on a common service definition and its guaranteed parameters (for instance, a service that offers a guaranteed bandwidth within a specific interval). Therefore, the only possible unit of aggregation for the inter-domain case can be service-specific resources of a commonly defined service. It has to be mentioned that the described aggregation is different from possibly existing DiffServ aggregates between adjacent domains. Usually these aggregates have to be maintained independently of the reservation aggregates.

As noted before, aggregation of reservations is possible between arbitrary DiffServ domains. It is assumed that every DSDM has knowledge of the inter-domain routing table (e.g., as passive BGP peer) from which an internal AS graph representation is created. In this graph, a node represents an AS and an edge a link between ASes. A DSDM also records every established reservation in this graph along the reservation's path, i.e., every node also contains a set of reservations that traverse this AS or end in this AS. Surely, this graph represents only the view of this specific DSDM. Therefore, it only sees directed paths from its own node to other ASes, and, it only records reservations for which it receives requests.

1) *Establishing Aggregates*: When a new reservation request is received, the DSDM checks whether there exist already other reservations along the path towards the destination. Thus, the number of existing reservations is checked in each node on the reverse path, i.e., backwards

from the destination to the own node. If there exist at least k reservations, a new aggregate can be created. This aggregate will cover the capacity of k reservations plus the resources of the newly requested reservation. Furthermore, the capacity of the aggregate should be increased by an additional amount in order to include future reservations without having to increase the aggregate's capacity first. The additional but possibly 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, because longer aggregates save more states and signaling messages. However, only aggregates of at least two hops make sense, because creating a very short reservation aggregate between two directly adjacent ASes does not have any benefit, but only additional overhead. Therefore, an aggregate must comprise at least one more node except starting point and end point.

In the following, the notation $A_p(u, w)$ denotes a reservation aggregate along path p (a sequence of edges) with node u as starting point and node w as end point of the aggregate. The value of $A_p(u, w)$ is the capacity of the aggregate. $\Sigma(A_p(u, w))$ denotes the sum of resources of all flows that are aggregated in $A_p(u, w)$, while $\Delta(A_p(u, w)) := A_p(u, w) - \Sigma(A_p(u, w))$ denotes the amount of "unused" resources in the aggregate $A_p(u, w)$, i.e., these resources are not actually reserved by currently aggregated flows.

The aggregate is installed along path p by allocating the additional 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 . Initially, only directly adjacent DSDMs communicate with each other. By successful establishment of the aggregate a new trust relationship has also been built between the starting point and end point through using a chain of trust between adjacent DSDMs. Therefore, after aggregate creation the request for establishing the single reservation can be directly forwarded from the starting point u to w .

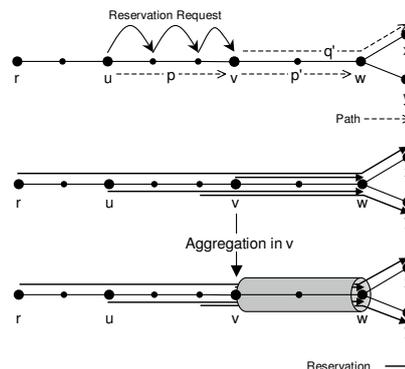


Fig. 4. Creation of an aggregate by the DSDM in AS v

Figure 4 shows an example where the DSDM in AS v decides to create an aggregate along path p' upon receiving a new reservation request for a flow from u to w . Three

existing flows and one new flow are included into an aggregate $A_p'(v, w)$.

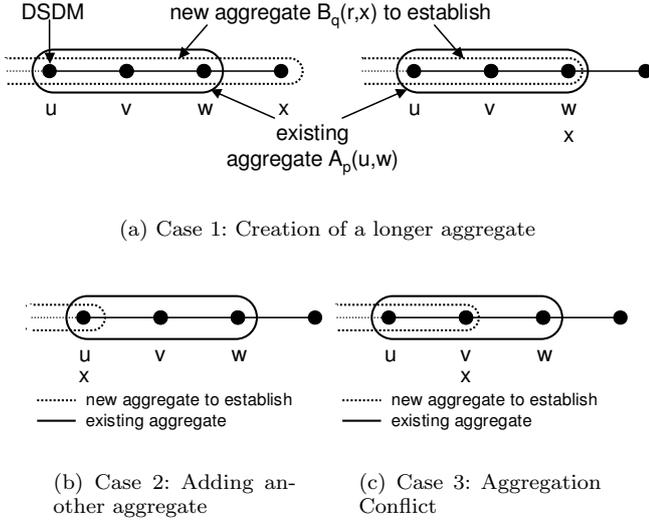


Fig. 5. Different cases of creating a new aggregate

However, some cases (which are illustrated in figure 5) have to be considered when new aggregates are created. Aggregate $A_p(u, w)$ is already established when a new aggregate $B_q(r, x)$ should be created. The latter would cover some reservations that were already aggregated in $A_p(u, w)$. In case 1 (cf. fig. 5(a)) a longer aggregate should be created (r lies before u and x behind w or falls 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, because it will save more states and messages. 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 $A_p(u, w)$. In 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 $B_q(r, x)$ contains reservations that were already aggregated, they are not known at the DSDM in x 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.

As mentioned in case 1, in the DARIS approach, aggregates can be hierarchically nested. Two slightly different types of nested aggregates are depicted in figure 6. Nested aggregates are visible within a DSDM at a starting point if two aggregates have the same starting point but different end points (cf. fig. 6(a)). In other cases, aggregate nesting is not visible for a starting point of an aggregate, if other DSDMs downstream embedded the

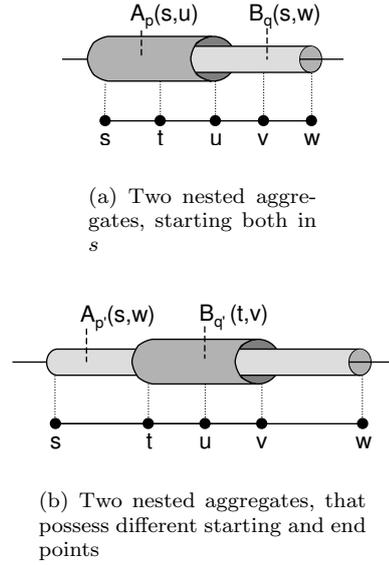


Fig. 6. Different kinds of nested aggregates

aggregate into a larger one (cf. fig. 6(b)). As illustrated by case 1, hierarchical aggregates are important in order to let longer (and more efficient) aggregates evolve over time and to preserve the independence of the aggregation decision within DSDMs. However, there is a small chance that nested longer aggregates cannot be created for reservations that had an earlier conflict during establishment of the enclosing shorter aggregate. As simulations showed, these “indirect conflicts” are rare events, but nevertheless they have to be handled by DSDMs.

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 comprising a different set of reservations.

2) *Changing and Terminating Aggregates*: The capacity of an aggregate may need to be adapted over time. Especially if a new reservation should be included 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 include future reservations without having to increase the aggregate’s capacity first.

Similarly, if an aggregated reservation is terminated, 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 a few minutes.

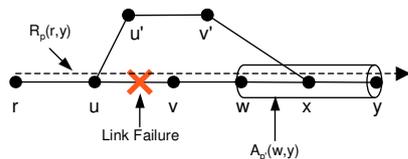


Fig. 7. Inter-domain routing changes may cause new conflicts

Inter-domain route changes are not easy to handle, because they can 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 deleted all state information about R_p already it cannot access state information about 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.

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 adaptation intervals.

II. THE SIGNALING PROTOCOL DMSP

Because aggregation causes also some overhead in signaling, one objective of the DARIS architecture was to reduce this overhead. In this section, some of the specialized and novel protocol mechanisms are presented.

First of all, it has to be noted, that signaling for resources of unicast flows in DiffServ networks must be sender-initiated. The reservation request must start at the sender and traverse the path downstream to the receiver. During this first pass, admission control takes place and resources are pre-reserved. The response is forwarded backwards (upstream) from the receiver back to the sender. During this second pass, resources are actually reserved and unnecessary pre-allocated resources released. Furthermore, traffic conditioning mechanisms have to be adjusted at this second pass. Adjusting them downstream would cause packet loss and QoS violations, e.g., if the rate of a traffic shaper at an egress boundary router is increased before the corresponding traffic profile had been updated in the next downstream ingress router. Nevertheless, DMSP supports receiver-initiated reservations by sending such a request directly to the sender which then initiates the actual reservation and sends back the result to the initiator.

The signaling sequence for the establishment of an aggregate is shown in figure 8. In step (1) the sender sends a *SrvEstReq* message to its corresponding DSDM in order to establish a new reservation for a specific service. The DSDM in r now wants to create a new aggregate and signals an *AggEstReq* message (step (2)) to its neighbor in u . The aggregate is established in the following steps

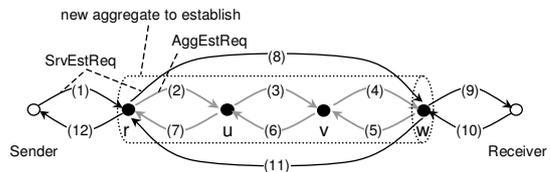


Fig. 8. Establishing an aggregate

(3)–(7) before the *SrvEstReq* gets directly forwarded to the aggregation end point w in step (8). Finally, the aggregate end point forwards the request towards the receiver in step (9). The subsequent response (10) is also directly forwarded from the aggregate end point w to the aggregate starting point r (11), which then forwards it to the sender that initiated the request.

To avoid several trials when establishing an aggregate, the protocol supports requests that carry parameter intervals. Thus, a lower bound of the requested resources for an aggregate A_p would be $\Sigma(A_p(u, w))$ and the upper bound $A_p(u, w)$. Consequently, any value that falls within the requested bounds will be accepted by the initiator. Additional capacity is reserved if possible, and the *SrvEstReq* will not fail if there were enough resources for the new single flow but not for the additional requested aggregate resources $\Delta(A_p(u, w))$.

Once an aggregate is established and a new reservation should be integrated into the existing aggregate, the residual capacity of the latter may not be sufficient to cover the resources of the new flow. In this case, the aggregate's capacity (i.e., its reserved resources) has to be increased at first. Furthermore, if the aggregate is embedded into another aggregate, the latter may also have to be increased. For these situations, a special support through the signaling protocol DMSP has been developed.

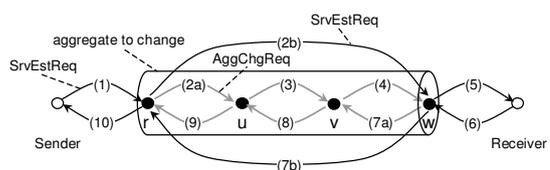


Fig. 9. Integrating a new reservation into an existing aggregate by increasing its capacity

In these cases, parallelism is used to reduce the reservation setup time. Figure 9 shows an example for this situation. After the *SrvEstReq* is received (1), the DSDM in r forwards it directly (strictly speaking after a successful local admission control) to the aggregate end point w (2b). After performing a local admission control at w forwarding of the *SrvEstReq* message is suspended until the *AggChgReq* message to increase the aggregate arrives at w (4). This can be considered as an optimistic strategy assuming that increasing the aggregate will be successful. If the aggregate cannot be increased by the required amount, r simply sends a *SrvAbortReq* message to w , thereby canceling the waiting *SrvEstReq* message. Thus, two signaling processes are started at w in parallel: increasing the

aggregate capacity by sending a *AggChgReq* message to u (2a) and forwarding the *SrvEstReq* message (2b) to the aggregate end point. After the increment of the aggregate was considered successful at w , the *SrvEstReq* message is forwarded towards the receiver (5). At this point it is possible to let the response *AggChgRsp* wait at w until the response for the single reservation arrives (6). Depending on the providers policy, incrementing the aggregate may be likewise canceled when the response (6) for the single flow is negative. Alternatively, a successful increment of the aggregate may be preserved even when establishing the reservation for the single flow fails afterwards. The former option will lead to a parallel sending of the *AggChgRsp* (7a) and the *SrvEstRsp* (7b) message, while for the latter option *AggChgRsp* will be sent back earlier. However, the *SrvEstRsp* will always wait at r on the corresponding *AggChgRsp* to arrive, before it is finally forwarded to the sender (10).

In order support the withholding of a message until the required message arrives, DMSP supports so-called “forward waiting conditions” and “reply waiting conditions”. These can be seen as simple references to other messages whose arrival trigger the continuation of the suspended message processing. It is sufficient to support only one “forward waiting condition” and one “reply waiting condition” per message, because there exists at most one superior aggregate.

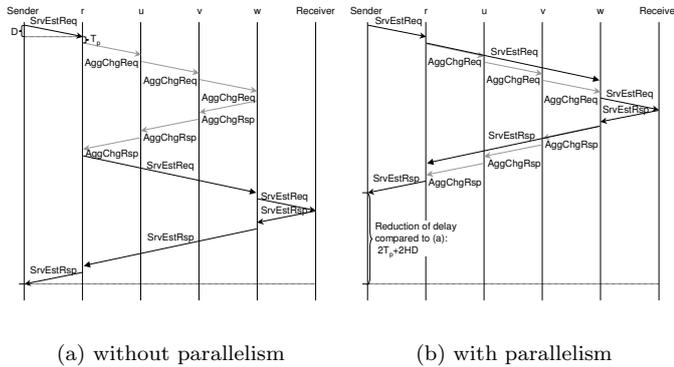


Fig. 10. Timed sequence diagrams of signaling sequence from figure 9 without and with parallelism, D Delay, T_p Processing Time, H Number of Hops between DSDMs.

Figure 10 illustrates the advantage of this solution for the example given in figure 9. The total time for the parallel version of the signaling sequence is clearly shorter than the non-parallel version (assuming that partial delays and processing times are equal in both cases). One can coarsely quantify the gain: D denotes the average delay between two directly adjacent DSDMs or between a DSDM and an end system respectively, T_p denotes the mean processing time within a DSDM (neglecting the processing times in end systems for the sake of simplicity). Similarly, it is assumed that D and T_p are equal for all systems. The number of involved DSDMs is $H - 1$ (H is the number of

hops). Then the total time for the complete sequence of figure 10(a) can be calculated as:

$$T_{\text{Normal}} = 2(H + 2)(D + T_p) + 2HD \quad (1)$$

and likewise for the sequence of figure 10(b):

$$T_{\text{Parallel}} = 2(H + 1)(D + T_p) + 2D \quad (2)$$

The achieved gain (reduction of setup time) is the difference between equations 1 and 2, and thus $T_{\text{Normal}} - T_{\text{Parallel}} = 2T_p + 2HD$. Therefore, the savings will be more than one round trip time and they will increase with a growing length of the path.

III. EVALUATION

First of all, the applicability of the DARIS aggregation principle was examined. It is the question whether current AS paths in the Internet are long enough for aggregation. For this purpose, several BGP table dumps from the routeviews project [11] were used and converted into a graph representation. Subsequently, for every AS that was not a pure transit AS the shortest path lengths to every other AS were calculated and counted. The result is depicted as complementary distribution function $P(X > x) = 1 - P(X \leq x)$ in figure 11: 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 (thus saving states and message processing in at least two ASes). Therefore, enough aggregation possibilities are given in the real Internet topology.

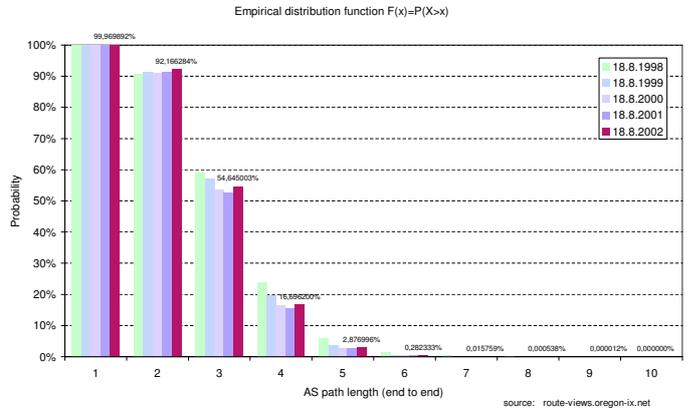


Fig. 11. Empirical distribution of end-to-end AS path lengths in the Internet

The DARIS principle was simulated (including the special signaling support by DMSP) by using the OMNeT++ simulation environment [12]. In order to evaluate the DARIS aggregation a simple topology as depicted in figure 12 was used at first. This topology consists of 13 DSDMs, where DSDMs 5 and 6 have to handle messages from DSDMs 1 and 2, as well as 3 and 4 respectively. Similarly, DSDM 7 has to handle 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. 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 with a mean duration of 180s. Every reservation requested a constant bandwidth of 64kbit/s. Adaptation of an aggregate’s capacity is controlled by a combination of several mechanisms that are not described in detail here. However, an exponential moving weighted average 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 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. An example of the adaptation of an aggregate’s resources at DSDM 1 is given in figure 13.

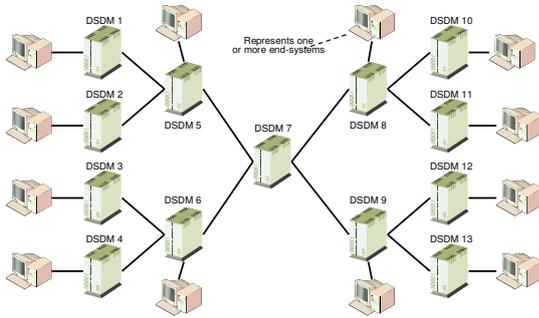


Fig. 12. A simple simulation topology

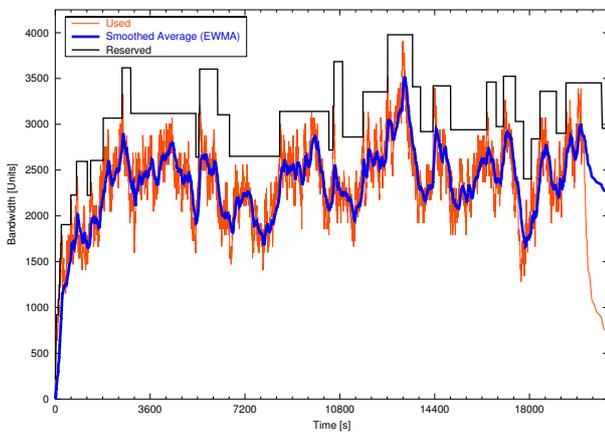


Fig. 13. Adaptation of an aggregate’s capacity

The simulations ran with different reservation initiation rates: each end-system initiated 1000, 10000, 50000, 100000, 500000 and 1000000 sessions with associated increasing mean rates of 0.05, 0.5, 2.5, 5, 25, and 50 sessions (or reservations) per second. Every configuration was run with 50 different random seeds. Furthermore, the minimum number k of reservations that must exist before an aggregate can be created was varied over the values 2, 4, 8, and 100.

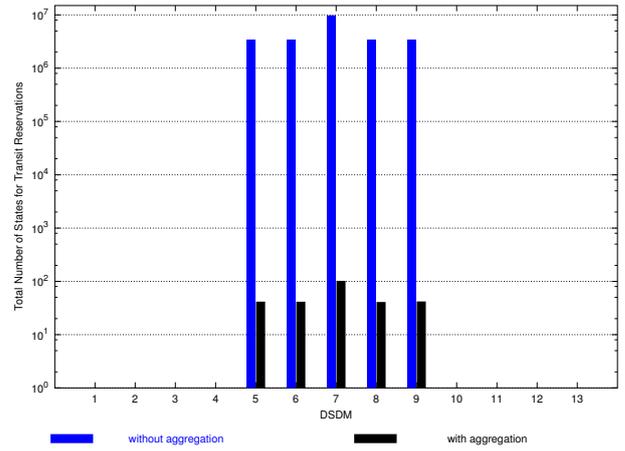


Fig. 14. Number of transit states per DSDM

First of all, the number of transit states per DSDM is shown in figure 14 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 terminated by end-systems of the own domain cannot be avoided on principle. Consequently, DSDMs 1–4 and 10–13 do not have to keep any transit states, because they are stub domains. The diagram shows the averaged results for the configuration 1000000 sessions per end-system with a mean rate of 50 initiated sessions per second. By using the DARIS aggregation principle, several orders of magnitude of states can be saved.

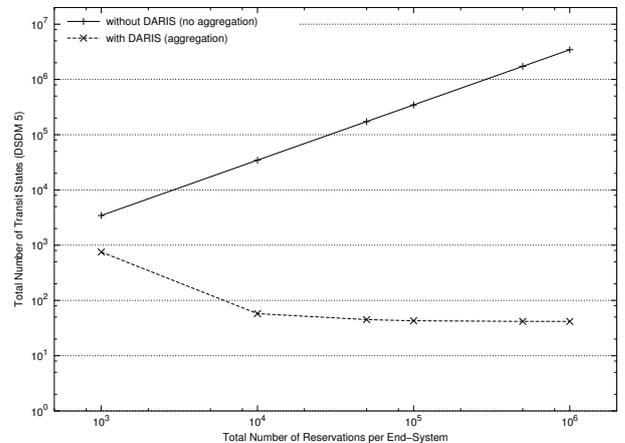


Fig. 15. Number of states within DSDM 5

Figures 15 and 16 show the behavior of DARIS within DSDMs 5 and 7 for the different configurations and $k = 2$. The plotted number of states reflects the total number of states over the complete runtime of the simulation. The overhead due to own and foreign aggregates (aggregates initiated by other DSDMs) is very small and the number of states can be reduced to a scalable and manageable amount by the DARIS concept.

Due to the symmetric topology results for DSDMs 6,8 and 9 are similar to those of DSDM 5.

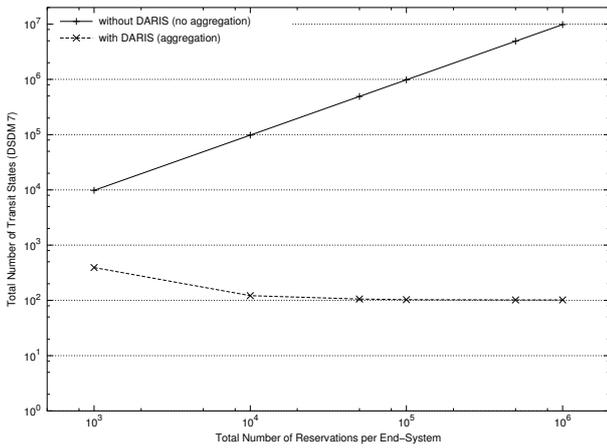


Fig. 16. Number of states within DSDM 7

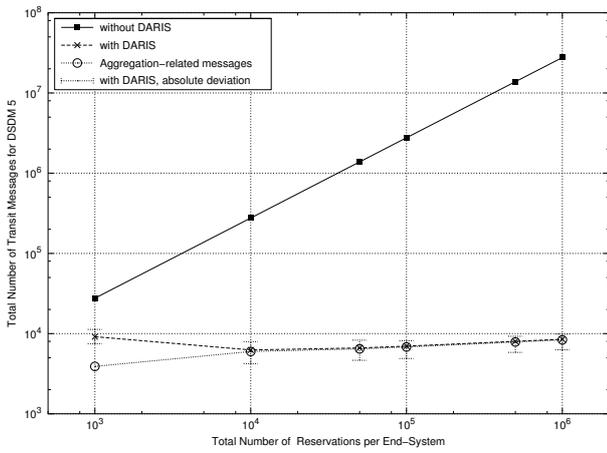


Fig. 17. Number of messages within DSDM 5

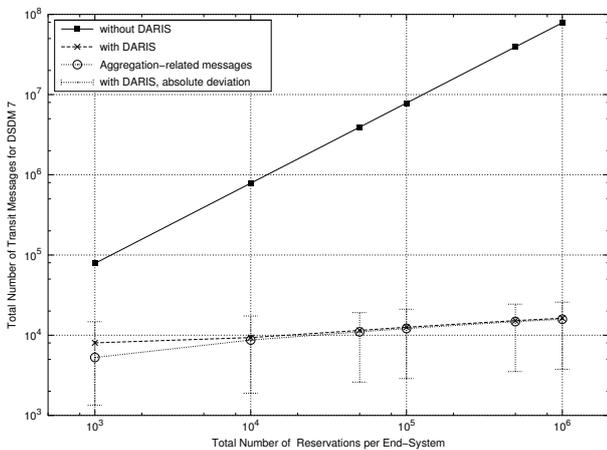


Fig. 18. Number of messages within DSDM 7

In the next step, the number of signaling messages has to be examined. Figures 17 and 18 depict the number of transit messages and aggregation-related signaling messages. Similarly, DARIS reduces the number of transit DMSP messages to a scalable amount. The overhead caused by aggregation is small compared to the number of transit messages without aggregation. It is obvious that with an

increasing number of sessions the number of aggregation-related messages must also increase slowly: the aggregates have to be adapted more often over the longer runtime.

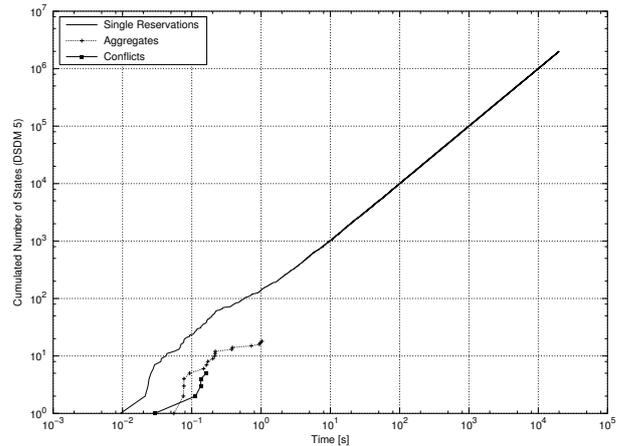


Fig. 19. Development of states and conflicts over time in DSDM 5

So what about the earlier mentioned aggregation conflicts? They occur only in early phases of aggregate creation. After most aggregates have been established the situation stays stable. Figure 19 shows the development over time in DSDM 5 for $k = 8$ and 50000 sessions.

First trials to use a real Internet topology were made. Once more the routeviews archive of BGP table dumps was used to extract an undirected AS graph. Although the BGP table represents always a specific view of a particular AS, it can be assumed that this is the most complete topology information at AS level about the Internet, and, a lot more real than any randomly generated “Internet-like” topology. Due to memory limitations only a topology of 1999 (of August 18th) could be used so far. It consists of 5561 ASes from which 4196 are stub ASes, i.e., they are not able to (single-homed ASes) or are not configured to forward transit traffic. Because 64 ASes were classified as pure transit ASes, all other 5497 ASes had end-systems to initiate reservations. Every end-system initiated 5000 reservations with a mean rate of one reservation every 3 seconds. k was set to 2 and the same parameters for the pareto distribution as within the small topology were used.

Figure 20 shows the preliminary results of such a simulation. The DSDM indices were sorted by decreasing number of states. Due to the high number of stub ASes, only 1365 ASes carry transit traffic. In the average savings through applying the DARIS principle lie also in the range of one to two orders of magnitude. However, because of the equally distributed reservations, there are hardly 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 AS. Nevertheless, further simulations have to be made to understand the aggregation behavior better in real 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

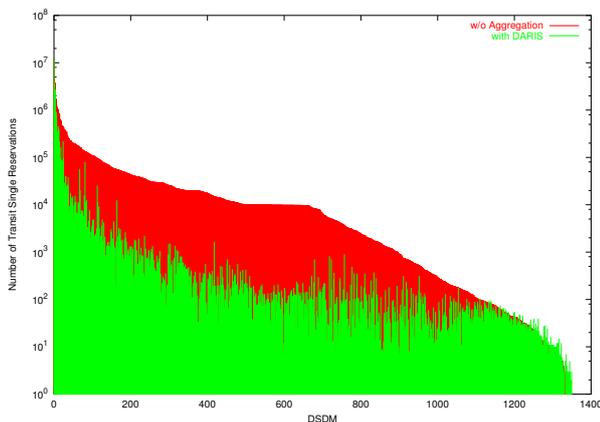


Fig. 20. Transit states per DSDM of an Internet topology from August 18th, 1999

other AS. Currently, the whole simulation is ported to 64-bit platforms due to the 4GB limit per process of traditional 32-bit platforms.

IV. SUMMARY AND OUTLOOK

Global availability of on-demand QoS-based services is one desirable functionality for a next generation Internet. Especially, scalability at the inter-domain level with respect to the number of states and signaling messages of potential solutions has to be considered. Dynamic Aggregation of Reservations for Internet Services (DARIS) is a means to couple aggregation techniques of the data path with those in the control plane. DARIS allows a very flexible and fully hierarchical aggregation of resource reservations. A novel signaling mechanism is used to reduce the setup time for new reservations in combination with existing aggregates. The simulations confirm that DARIS reduces the number of reservations states and messages considerably, even in a real Internet topology.

Currently, extensive efforts are made to examine the behavior of the DARIS concept for more current Internet topologies. The memory limit of 32-bit architectures for a single process is the main problem here, so a 64-bit version is currently built. Furthermore, the effects of using

DMSP's waiting conditions on the average delay may be quantified in future simulation scenarios.

One problem 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.

REFERENCES

- [1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC 2475 (Informational), Dec. 1998.
- [2] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633 (Informational), June 1994.
- [3] F. Baker, D. Black, S. Blake, and K. Nichols, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC 2474 (Standard), Dec. 1998.
- [4] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, M. Speer, B. Braden, B. Davie, E. Felstaine, and J. Wroclawski, "A Framework For Integrated Services Operation Over Diffserv Networks," RFC 2998 (Informational), Nov. 2000.
- [5] B. Davie, A. Charny, F. Baker, J. Bennet, K. Benson, J.-Y. L. Boudec, A. Chiu, W. Courtney, S. Davari, V. Firoiu, C. Kalmanek, K. Ramakrishnam, and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)," RFC 3246, Mar. 2002.
- [6] V. Jacobson, K. Nichols, and L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet," RFC 2638 (Informational), July 1999.
- [7] O. Schelén and S. Pink, "Resource sharing in advance reservation agents," *Journal of High Speed Networks*, vol. 7, no. 3-4, 1998, special Issue on Multimedia Networking.
- [8] —, "Aggregating resource reservations over multiple routing domains," in *Proceedings of IFIP Sixth International Workshop on Quality of Service (IWQOS'98)*, IFIP. IEEE, May 1998.
- [9] P. Pan, E. L. Hahne, and H. Schulzrinne, "BGRP: Sink-Tree-Based Aggregation for Inter-Domain Reservations," *Journal of Communications and Networks*, vol. 2, no. 2, pp. 157-167, June 2000, special Issue on QoS in IP Networks.
- [10] B. Teitelbaum and P. Cimento, "QBone Bandwidth Broker Architecture," <http://qbone.internet2.edu/bb/bboutline2.html>, June 2000, working Document of the QBone Bandwidth Broker Work Group.
- [11] D. Meyer, "Route views project page," <http://www.routeviews.org/>, May 2002.
- [12] A. Varga, "OMNeT++ Discrete Event Simulation System 2.3," <http://www.omnetpp.org/>, Technical University of Budapest, Sept. 2003.