An Investigation of Inter-Domain Control Aggregation Procedures[*]

Rute Sofia[1,2]   Roch Guérin[1]   Pedro Veiga[2]

[1]ESE, University of Pennsylvania
rsofia@seas.upenn.edu,guerin@ee.upenn.edu

[2]DI, University of Lisbon
{rute,pmv@di.fc.ul.pt}

Abstract

Current Quality of Service models such as those embodied in the Differentiated Services proposal, rely on data path aggregation to achieve scalability. Data path aggregation bundles into a single aggregate multiple flows with the same quality requirements, hence decreasing the amount of state to be kept. A similar scalability concern exists on the control path, where the state required to account for individual reservations needs to be minimized. There have been several proposals aimed at control path aggregation, and the goal of this paper is to expand on these works in an attempt to gain a better understanding of the various parameters that influence the efficiency of different approaches. In particular, we focus on inter-domain control aggregation, and compare an Autonomous System (AS) sink-tree based approach with two examples of a shared AS segment based approach, in terms of the amount of state kept, both per AS and per edge router. Our main contributions are in providing a greater understanding into the design of efficient control path aggregation methods.

1. Introduction

Data path *Quality of Service* (QoS) issues are by now reasonably well understood, and a number of different alternatives have been proposed and investigated, e.g., *Integrated Services* (*IntServ*) [3] and *Differentiated Services* (*DiffServ*) [12], each representing a different trade-off in terms of capability and scalability. However, the same understanding is not really available for control path issues. The control path consists primarily of mechanisms for reserving and maintaining the necessary data path resources, as embodied in proposals such as *Internet Streaming Protocol (ST-II)* [5] and *Resource Reservation Protocol (RSVP)* [4], with the latter being the current solution of choice for most new IP services. The main concern of these proposals is their scalability, specially in terms of inter-domain links, that are expected to carry a large volume of individual reservation requests.

Our main motivation is, therefore, to gain a better perspective into the scalability of various control mechanisms, and their ability to handle large reservation volumes. We focus on inter-domain control reservations, as we expect them to be the most stressful in terms of scalability. Our approach is not so much to propose a specific mechanism, but instead to try to gain a basic understanding of factors and parameters that affect the scalability of inter-domain control reservation mechanisms. In particular, we focus on evaluating various *aggregation* techniques that attempt to minimize state and processing due to resource reservation on links connecting different routing domains or *Autonomous Systems* (AS's), i.e., *inter-domain* links. Information provided by the *Border Gateway Protocol* (BGP) [17], the current dynamic solution for inter-domain information exchange, allows the use of different criteria to decide how, when, and where to aggregate reservation requests. Aggregation can, for example, be done on the basis of a single shared AS hop, or on the basis of a shared AS path segment, or simply be based on having the same destination AS, as proposed by Pan et al. [6]. These different options translate into maintaining state at different locations in the network. In general, state needs to be kept for each individual reservation at all aggregation and deaggregation points, while state is kept for aggregate reservations at all the intermediate inter-domain links they traverse. Hence, the goal of a scalable solution is to minimize the overall reservation state in the network, as well as the state that any router needs to maintain.

In addition to the state needed, a scalable solution should also take into account processing and signaling requirements, ensuring that both are kept as low as possible. A related factor is the bandwidth efficiency of a solution, and in particular how often the bandwidth allocated to an aggregate reservation is updated. Ideally, updating bandwidth after every change to the individual reservations of an aggregate would ensure that only the minimum possible quantity

would be allocated. However, this would most likely increase the signaling load significantly. Alternatively, bandwidth allocation could be updated less frequently to minimize signaling overhead. However, this could affect network efficiency by providing some aggregate reservations with more bandwidth than they really need, potentially preventing others from getting the bandwidth they require.

All of the above represent issues that need to be explored, and carrying out such a comprehensive research is clearly beyond the scope of a single paper. In this paper, we concentrate on the aspect of state optimization and consider two representative families of possible algorithms. The first one makes aggregation decisions on the basis of shared AS sink-trees, while the second relies on shared AS path segments. We consider algorithms that belong to each family, and evaluate their cost in terms of state they require both at the AS and router level.

The rest of the paper is organized as follows: Section 2 covers related work. Section 3 describes a generic aggregation scenario and the two aggregation approaches under consideration, and presents a simple analytical model for computing state required by several candidate algorithms. Section 4 is devoted to evaluating, by means of simulations, the performance of the algorithms for different network topologies. Finally, Section 5 summarizes findings and outlines future work.

2. Related Work

Guérin et al. [8] present a survey of possible approaches to aggregate RSVP requests assuming unicast scenarios and covering issues such as RSVP state management and path characterization. The authors propose the use of aggregation *tunnels*, i.e., pipes between entry and exit points of a defined *aggregation region*, i.e., a cloud of routers where regular RSVP messages are ignored. A similar approach is followed by Berson et al. [11]. They consider unicast and multicast scenarios, focusing also on RSVP aggregation within an aggregation region. These two approaches are concerned with RSVP scalability: RSVP requires all the routers on the path of an individual reservation to maintain state dedicated to that reservation. The resulting state quantity can be overwhelming especially for backbone routers that may have to support a large number of simultaneous requests. The two proposals reduce reservation state by aggregating individual requests inside an aggregation region. However, in both proposals, an aggregation region is typically synonymous with an AS. As a result, neither considers the problem of inter-domain control aggregation, which is the focus of this paper.

Pan et al. [6] were the first to explicitly consider the problem of inter-domain aggregation, for which they introduced an inter-domain signaling protocol, the *Border Gate-*

way Reservation Protocol (BGRP). BGRP aggregates control information by merging requests that have the same destination *AS*. For each reservation, BGRP sends a pair of control messages along the path that an aggregate will follow to reach its destination according to BGP rules, i.e., a *sink-tree*. On each AS along the path, edge routers keep information regarding the tree each individual reservation belongs to, its sink or root, and bandwidth to reserve for the tree. Using a tree has the advantage of avoiding *intermediate deaggregation points*, i.e., AS's between source and destination where individual reservations need to be regenerated. Hence, deaggregation takes place only at the destination AS. Pan et al. show that BGRP has good performance when compared with RSVP without aggregation. However, BGRP was not compared to other possible aggregation methods. Thus, assessing its effectiveness as an inter-domain solution remains an open question. Answering such a question and exploring the space of possible approaches is one of the motivations of our work.

3. Control Aggregation Issues and Definitions

In this section, we introduce some terminology and concepts related with inter-domain control aggregation. Throughout the document, an *aggregation region* or *domain* is synonymous with an AS. An *ingress router* is a router placed at the *edge* of an AS, crossed by traffic that enters the AS. Similarly, an *egress router* is a router placed at the edge of an AS, but crossed by traffic that exits the AS. Requests having in common some path characteristics and crossing the same egress router can be bundled together in an *aggregate*. For instance, requests going to the same destination can be aggregated together hence being treated as a single request by edge routers along a path. An aggregate is named *originating* if it starts in the current AS; it is named *ending* if it ends in the current AS, having therefore to be deaggregated; it is named *transient* if it is just passing by the current AS. Consequently, aggregates are characterized by their starting and ending AS's. An *aggregator* is a process in charge of processing and possibly merging requests as they leave an AS, hence positioned at egress routers. A *deaggregator* is a process in charge of splitting ending aggregates into requests, hence positioned at ingress routers. *Merging* of aggregates takes place when aggregates that cross different ingress routers and the same egress router at an AS, have the same aggregation requisites (for instance, share a path segment). Such is the case exemplified in Fig.1, where aggregate A2 is merged into aggregate A1. An *AS hop* represents an inter-domain hop. An AS j is *downstream* of an AS i if it is between i and a destination AS; AS j is *upstream* of AS i if it is between a source AS and i.
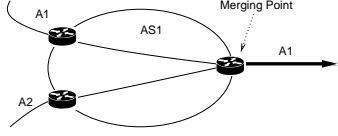
Figure 1. Merging of aggregates.

3.1. Generic Model

To describe and compare the aggregation approaches, we use the generic aggregation model illustrated in Fig. 2 (a), where AS's 1 to P represent source AS's and $P + K + 1$ to $P + K + N$ destination AS's. Between source and destination AS's, there is a segment with K AS's. For ease of understanding and visualization, traffic flows only from left to right. We assume that each of the source AS's wants to establish Y individual reservations with each of the destination AS's. In the selected topology, all paths have a size of $K + 1$ AS hops, where K is a variable that can be set to reflect a typical AS hop count, e.g., based on a given AS path size distribution. To obtain realistic values for K, we use the values collected by Telstra [13] and presented in [10][1]. This model, an AS-level *dumbbell* [2] topology, is simple and sufficient to explain the state accounting methodology we use, since state along any path differs as a function of an AS location: sources, destinations, and intermediate AS's.
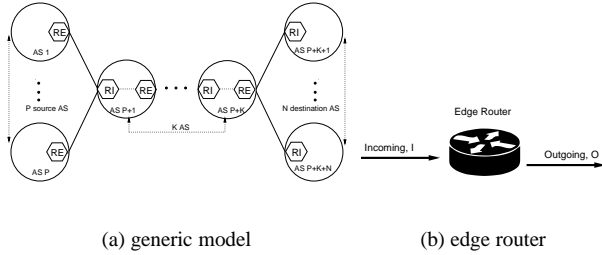

(a) generic model   (b) edge router

Figure 2. State accounting.

State represents information about reservations that routers along a path need to store. Hence, from a router perspective, state is associated with the interfaces crossed by the requests, as illustrated in Fig. 1 (b).

In our model, source AS's (1 to P) keep only state related with outgoing reservations. Destination AS's ($P + K + 1$ to

---

[1]This data is based on BGP measurements obtained from five major operators in 2001 and gathered from a total of 60978 AS. Among other facts, it shows that the current maximum AS path has a size of 10 AS's. So, K is taken to be less than ten, since the biggest path in our scenario has $K + 1$ AS's.

$P + K + N$) keep only state due to incoming reservations. State accounting for AS's $P + 1$ to $P + K$ is more complex and depends on the aggregation approach used.

We consider that a request, whether individual or aggregate, occupies one unit of state per interface that the reservation crosses: if x requests entering an RE will be merged into one outgoing aggregate, the corresponding state is $x + 1$ units; if an ending aggregate at an RI contains x requests, the resulting state is $1 + x$ units; if an aggregate is transient, it requires 2 units of state per RI and per RE it crosses. Hence, the average reservation state, Q, for an edge router i is given by Eq. 1 (a). I_i represents state due to incoming reservations and O_i state due to outgoing reservations.

$$(a): Q_i = I_i + O_i \qquad (b): S_m = \sum_{i=0}^{n} Q_i \qquad (1)$$

Correspondingly, the average state S_m for an AS m (Eq. 1 (b)), is simply obtained by summing the state of its n edge routers. Both S_m and Q_i are relevant performance measures for a given aggregation scheme. Tracking state at the AS level gives an overall measure of performance, while tracking it at the router level can help identify variations in state that routers are required to maintain. For example, an aggregation scheme could achieve a low AS level state quantity by having state concentrated at a few routers. In the next sections, we illustrate state accounting for several aggregation methods.

3.2. Sink-Tree AS Based Approach

Fig. 3 displays an example of state accounting for the scenario of Fig. 2 (a) when using a sink-tree AS based approach, as proposed in BGRP. Requests that share a destination AS are aggregated in the form of a sink-tree whose root is the destination AS.
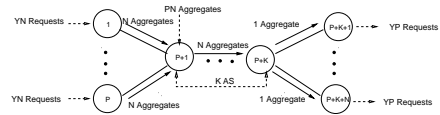

Figure 3. Accounting, sink-tree.

Since each source AS is generating Y individual reservations for each destination AS, we have $Y * N$ individual requests per source AS. Also, each source AS creates N aggregates, because aggregation is based on destination AS's. Therefore, there is a total of $P * N$ aggregates entering AS $P + 1$. At this AS, merging of the $P * N$ aggregates takes place, resulting in a total of N outgoing aggregates. Deaggregation occurs only at destination AS's, where incoming aggregates are deaggregated into individual reservation requests.

Tab. 1 details state kept in each AS, showing state at both ingress and egress routers, and Eq. 2 gives the global state count for a sink-tree aggregation approach in this particular scenario.

$$S_1 = PN(2Y + 4) + 4NK - 2N \qquad (2)$$

3.3. Shared AS Segment Based Approach

Aggregation decisions made by this approach have as basis the existence of an AS path segment common to an aggregate and to an arriving request. In contrast, the sink-tree approach requires a shared segment that extends all the way to the destination AS. In the shared segment approach reservation requests can be assigned to any aggregate with an ending point upstream of their destination AS. If no such aggregate exists, a new one is created, not necessarily extending all the way to the destination AS. The motivation for such flexibility is that shorter aggregates may accommodate more easily additional future requests. On the one hand, by aggregating reservation requests that share only a path segment, we expect to minimize the number of aggregates in use and hence, global state. On the other hand, this process can result in having multiple deaggregation points, each contributing with state, wiping out the advantage of reducing the number of aggregates.
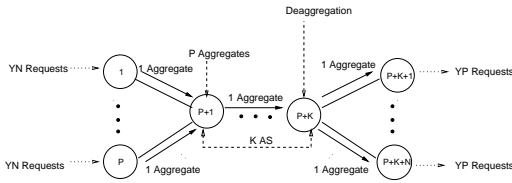

Figure 4. Accounting, shared segment.

Fig. 4 exemplifies accounting for the scenario illustrated in Fig. 2 (a), when AS $P + K$ is the intermediate deaggregation point. Tab. 2 shows state per AS when using the shared segment approach, while Eq. 3 gives global state, S_2.

$$S_2 = 4YPN + 4P + 2N + 4K - 6 \qquad (3)$$

In order to understand possible variations and also explore how to choose an optimal deaggregation point, we introduce next two algorithms based on the shared segment approach, and compare them with a sink-tree based algorithm, namely, BGRP.

Biggest Possible Shared Segment (BPS) This algorithm is triggered each time a new reservation request arrives. It then checks whether or not an adequate aggregate exists. By adequate we mean that the aggregate has to have in common with the request a segment with a pre-determined size. For instance, if a request has only one AS hop in common with the aggregate, then aggregating them might bring only a small advantage.
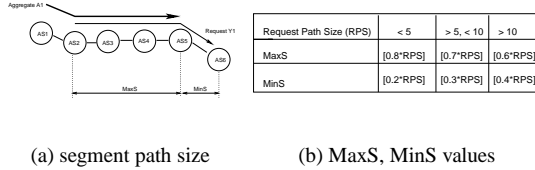

(a) segment path size   (b) MaxS, MinS values

Figure 5. BPS parameters MaxS, MinS.

Choosing a specific aggregate is, therefore, made based on the Maximum Shared Segment (MaxS) shared with the request. It has also to take into consideration the Minimum Surplus Segment (MinS), which corresponds to the remaining path segment from the end point of the chosen aggregate to the destination AS of the request. Fig. 5 (a) displays a diagram with MaxS and MinS segments. An individual request, Y1, originated in AS 2 and ending in AS 6, that has a possible aggregate candidate A1. A1 starts in AS 1 and ends in AS 5, three AS hops ahead of AS 2. MaxS is the segment between AS 2 and AS 5. MinS is the segment from AS 5 to AS 6. Fig. 5 (b) displays values for MaxS and MinS, where $[x]$ represents the closest integer to x. These values were chosen having in mind a typical path size, based on the path distribution mentioned in Section **??**. If there is no aggregate with the required characteristics, a new one is created with a size equal to MaxS.

Tab. 3 details state kept per AS when using BPS in the context of the topology of Fig. 2 (a).

The placement of the intermediate deaggregation point, AS D, is influenced by the size of the segment between source and destination AS's. Hence, the value of K is important to total state, S_3, as indicated by Eq. 4.

$$S_3 = \begin{cases} 4YPN + 4P - 2N + 2NK + 2K - 2, & K + 1 < 5 \\ 4YPN + 4P - 8N + 2NK + 2K + 4, & 5 \le K + 1 \le 10 \end{cases} \qquad (4)$$

BPS has the advantage of choosing deaggregation points that take into account the path size of requests. Hence, it is sensitive to the order and characteristics of requests that trigger the establishment of aggregates. For example, if the first requests arriving to an aggregator have small path sizes, the corresponding initial aggregates will tend to have small path sizes also, which will lead to a small number of aggregates, but possibly to many deaggregation points. However,

Table 1. Global state, sink-tree approach.

AS	1	(...)	P	P+1	P+2	(...)	P+K	P+K+1	(...)	P+K+N
Ingress (IN,OUT)	-	-	-	NP, NP	N, N	N, N	N, N	1, YP	1, YP	1, YP
Egress(IN,OUT)	YN, N	YN, N	YN,N	NP, N	N, N	N, N	N, N	-	-	-
Total	(Y+1)N	(Y+1)N	(Y+1)N	3NP+N	4N	4N	4N	YP+1	YP+1	YP+1

Table 2. Global state, shared segment approach.

AS	1	(...)	P	P+1	P+2	(...)	P+K	P+K+1	(...)	P+K+N
Ingress (IN,OUT)	-	-	-	P,P	1,1	1,1	1,YPN	1, YP	1, YP	1, YP
Egress(IN,OUT)	YN, 1	YN, 1	YN,1	P,1	1,1	1,1	YPN, N	-	-	-
Total	YN+1	YN+1	YN+1	3P+1	4	4	2YPN+N+1	YP+1	YP+1	YP+1

Table 3. BPS state per AS.

AS	1	(...)	P	P+1	(...)	Deaggregator, D	(...)	P+K+1	(...)	P+K+N
Ingress (IN,OUT)	-	-	-	P,P	1,1	1,YPN	N, N	1, YP	1, YP	1, YP
Egress(IN,OUT)	YN, 1	YN, 1	N,1	P,1	1,1	YPN,N	N, N	-	-	-
Total	YN+1	YN+1	N+1	3P+1	4	2YPN+N+1	4*N	YP+1	YP+1	YP+1

Table 4. WDS state per AS.

AS	1	(...)	P	P+1	P+2	(...)	P+K	P+K+1	(...)	P+K+N
Ingress (IN,OUT)	-	-	-	P,P	1,1	1,1	1,YPN	1, YP	1, YP	1, YP
Egress(IN,OUT)	YN, 1	YN, 1	YN,1	P,1	1,1	1,1	YPN, N	-	-	-
Total	YN+1	YN+1	YN+1	3P+1	4	4	2YPN+N+1	YP+1	YP+1	YP+1

if initial requests have large path sizes, then the path size of initial aggregates will also be large. This implies the use of less deaggregation points, but possibly ending up with a larger number of aggregates.

Segment with 'Weighted' Deaggregation Point (WDS) This algorithm intends to remedy the deficiency of BPS in limiting aggregate path sizes due to the path size of the first requests received, by including information on the likelihood that a given AS will be a termination point for many future requests. Specifically, WDS assumes that AS's with a larger number of downstream neighbor AS's are more likely to be deaggregation points. This makes such AS's better candidates for being the end-point of an aggregate, and is combined with the distance from the aggregation point when deciding how to create new aggregates. In other words, the aggregator computes a weight, W_m, for each AS m of each path request, based on the number of downstream AS neighbors and the distance from the aggregator to AS m. It then chooses as deaggregation point the AS with the biggest weight. Eq. 5 defines W_m, where n_j represents a downstream neighbor of m and d represents the distance from the origin AS to m, given in AS hops:

$$W_m = \sum_j n_j * d, \forall j, m \qquad (5)$$

There are two special cases for the algorithm. The first occurs when two AS's yield the same weight value. In this case, the algorithm chooses the AS nearest to the destination. The second occurs when the destination AS is a *leaf*, i.e., it has no downstream neighbors. For this case, the algorithm assumes that $n_j = 1$.

Tab. 4 displays state kept per AS for WDS in the scenario of Fig. 2 (a), when AS K is the only intermediate deaggregation location, chosen because it yields the largest weight. Total state for WDS, S_4, is given in Eq. 6.

The major drawback of WDS is that it has to know beforehand the number of downstream neighbors for each AS. Even so, WDS has the advantage of choosing deaggregation points that are less sensitive to the characteristics of individual requests and the order in which they are received.

$$S_4 = 4YPN + 4P + 2N + 4K - 6 \qquad (6)$$

3.4. A Comparison Example

In this section we present a simple comparison of BGRP, BPS, and WDS in terms of state they require for Fig. 2 (a) scenario. We aim to show changes in state due to the variation of the number of sources, destinations, and the size of the segment between sources and destinations, K. K is taken to be between 1 and 10, while P, N, and Y are taken to be between 1 and B, a large quantity. The possible combinations of P, N, Y, and K in terms of the maximum and the minimum values of Eq. 2, Eq. 4, and Eq. 6 are displayed by rows in Tab. 5

Row one represents a configuration with one source and destination AS, as well as one request only. This is a very simple configuration, just used to exemplify the behavior of the algorithms for configurations with small number of source and destination AS's, as well as low intensity of requests. For this case, the three algorithms show similar performance, which means that none of the varied parameters, in small quantity, has significant impact on state the algorithms require. When the number of requests is consider-

Table 5. Global state comparison, 1<K<10.

Row	P	N	Y	K	BGRP	BPS	WDS
1	1	1	1	1	$S_1 = 8$	$S_3 = 8$	$S_4 = 8$
				10	$S_1 = 44$	$S_3 = 44$	$S_4 = 44$
2	1	1	B	1	$S_1 \approx 2B$	$S_3 \approx 4B$	$S_4 \approx 4B$
				10	$S_1 \approx 2B$	$S_3 \approx 4B$	$S_4 \approx 4B$
3	1	B	1	1	$S_1 \approx 8B$	$S_3 \approx 4B$	$S_4 \approx 6B$
				10	$S_1 \approx 44B$	$S_3 \approx 16B$	$S_4 \approx 6B$
4	1	B	B	1	$S_1 \approx 2B^2 + 6B$	$S_3 \approx 4B^2$	$S_4 \approx 4B^2 + 2B$
				10	$S_1 \approx 2B^2 + 42B$	$S_3 \approx 4B^2 + 12B$	$S_4 \approx 4B^2 + 2B$
5	B	1	1	1	$S_1 \approx 6B$	$S_3 \approx 8B$	$S_4 \approx 8B$
				10	$S_1 \approx 6B$	$S_3 \approx 8B$	$S_4 \approx 8B$
6	B	1	B	1	$S_1 \approx 2B^2 + 4B$	$S_3 \approx 4B^2 + 4B$	$S_4 \approx 4B^2 + 4B$
				10	$S_1 \approx 2B^2 + 4B$	$S_3 \approx 4B^2 + 4B$	$S_4 \approx 4B^2 + 4B$
7	B	B	1	1	$S_1 \approx 6B^2 + 2B$	$S_3 \approx 4B^2 + 4B$	$S_4 \approx 4B^2 + 6B$
				10	$S_1 \approx 6B^2 + 38B$	$S_3 \approx 4B^2 + 16B$	$S_4 \approx 4B^2 + 6B$
8	B	B	B	1	$S_1 \approx 2B^3 + 4B^2 + 2B$	$S_3 \approx 4B^3 + 4B$	$S_4 \approx 4B^3 + 6B$
				10	$S_1 \approx 2B^3 + 4B^2 + 38B$	$S_3 \approx 4B^3 + 16B$	$S_4 \approx 4B^3 + 6B$

ably increased (row two), state increases, for any of the algorithms. However, BGRP requires less state than the other algorithms: this happens because BGRP keeps state due to individual reservations only at source and destination AS's, while BPS and WDS add to this the cost of keeping state due to individual reservations in one intermediate deaggregation point. When K changes from 1 to 10, the amount of state for any of the algorithms remains the same, which means that the value of K in this particular configuration does not influence the global state required by any of the algorithms.

From a global perspective, there are only three configurations where BGRP requires more state than the other algorithms: rows three, seven, and eight. These three rows represent configurations with a large number of destination AS's, hence representing specific cases where sink-tree based aggregation is not optimized. As for the behavior of the other algorithms under these configurations, we can observe that WDS is the only algorithm that is not influenced by the variation of K.

It should be noticed that the configuration presented in row four also contains a large number of destination AS's, as well as many requests, but BGRP requires less state than the other algorithms. This happens because BPS and WDS require state of individual reservations at intermediate deaggregators.

From this particular comparison, we can infer some preliminary conclusions. First, the intensity of requests is a factor of major importance for the performance of aggregation procedures: any of the algorithms suffers considerable performance variations when increasing the intensity of requests. Second, the sink-tree approach appears to be less sensitive to configurations with higher intensity of requests, which indicates that the use of intermediate deaggregation points result in high state cost along a path. Third, the shared-segment approach requires less state due to aggregates, since it reduces the number of aggregates generated. In the next section, we present simulations that explore in further detail the behavior of BGRP, BPS, and WDS.

4. Performance Evaluation

The analytical model presented in section **??** demonstrates state accounting by means of a particular scenario, highlighting the impact that specific factors (number of sources, destinations, and path size) have on the overall state of a network. However, when considering heterogeneous networks such as the Internet, there are other factors that might influence the state to be kept, such as the traffic distribution, or the average duration of reservations. Hence, to understand the behavior of the algorithms under realistic scenarios, we carry out simulations using *ns2* [15]. We model the arrival of requests as a Poisson[2] process with exponential distributed lifetime mean σ. To capture the influence of the average duration of requests in the overall state, we use three different types of requests: *short-lived* reservation requests (SLR) with an exponential average duration of 20 s, *long-lived* requests (LLR) with an exponential average duration of 120 s, and a mix of 50% SLR, 50% LLR, that stands for a particular example of mixed traffic (MLR). To distribute the requests across topologies and since there is no current information regarding traffic distribution in the Internet, we apply two different distribution methods: an homogeneous and a *hotspot* method. In the former, source AS's are chosen randomly and destinations are placed according to the distribution of addresses by AS distance mentioned in section **??**. In the latter, we use the concept of *hotspot*, i.e., an AS with higher incidence of traffic than the others.

In order to make a consistent comparison of the algorithms, we keep the average number of requests per second in the system (*intensity of requests*) constant, while varying the average duration of requests, according to the traffic intensity formula for an $M/M/\infty$ model [7]. We use this model, since for the case of state accounting, blocking overhead is negligible. For each simulation, state accounting is done both on the AS and edge router level by

---

[2]We chose a Poisson process to model the arrival of requests since it is known to describe well user *session* arrivals, as mentioned in [16, 14].

collecting statistics dynamically for incoming and outgoing reservations: minimum, average and maximum values are updated each time the corresponding variable changes. Also, to achieve statistically meaningful results, each experiment has been repeated several times using different random number seeds. Further and more detailed results can be found in [9].
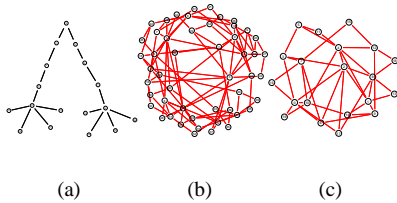

Figure 6. Used topologies.

4.1. Tree Topology

With this first experiment, we aim to highlight the behavior of each algorithm in two extreme cases: a scenario with a large number of destination AS's, unfavourable for BGRP, since there is one source sending several requests to a large number of different destination AS's, and a scenario with one destination AS only, very favourable for BGRP. Hence, we use the particular tree topology illustrated by Fig. 6 (a). Changing the roles of each node yields either a source-tree or a sink-tree scenario.

Source-Tree In the source-tree experiment, a unique source AS, node 0, sends requests to every other node, according to the homogeneous distribution method. Fig. 7 plots minimum, average, and maximum state per AS, at ingress and egress, when requests are of type SLR and the intensity of requests is of 5000. If we look at the upper chart for each algorithm, we can easily identify the source and destinations: the source keeps state only at the egress, while destinations keep state only at the ingress. For the cases of BPS and WDS, we also identify the intermediate deaggregation points, AS's 4 and 8. Comparing the lower chart of each of the algorithms, we see that BGRP requires twice the units of state of BPS or WDS for the aggregates it creates. However, even in this unfavourable scenario and from a global perspective, BGRP is the algorithm that requires the least state, because it only deaggregates at the destination. BPS and WDS add to this the cost of having intermediate deaggregation points in AS 4 and 8: these AS's keep state due to ending aggregates but also due to their mapped individual requests, increasing the global state required.

4.2. Sink-Tree

Let us now illustrate the behavior of the algorithms in a sink-tree scenario, where the only destination AS is node 0. Nodes 1 to 16 represent sources requesting reservations to node 0, according to the homogeneous traffic distribution method. Fig. 8 depicts minimum, average and maximum state values across the topology for BGRP, BPS, and WDS, when requests are of type SLR and the average number of requests in the system is 5000.
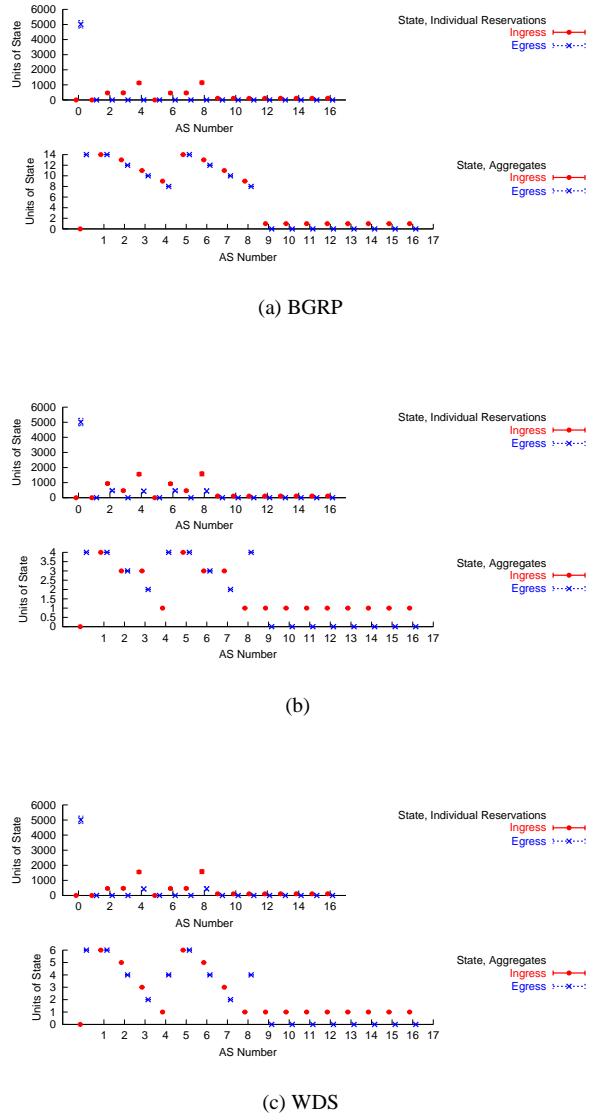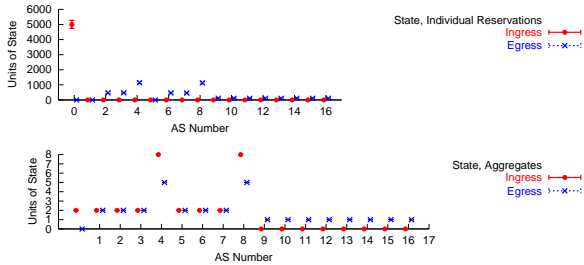

(a) BGRP

(b)

(c) WDS

Figure 7. Source-tree, $\sigma = 20s$, **5000 requests.**

Similarly to the source-tree scenario, we spot sources, destinations and intermediate deaggregation locations by

looking at the charts that plot state due to individual reservations for each algorithm. However, in terms of aggregates, there is a major difference for this scenario: while BPS still chooses intermediate deaggregation points, WDS chooses as only deaggregation points the destination. Hence, BGRP and WDS require the same state. Another major difference from the previous experiment is that BPS and WDS do not reduce the number of agregates when compared to BGRP. This is in compliance with the fact that a sink-tree scenario is a best-case for BGRP.

(a) BGRP
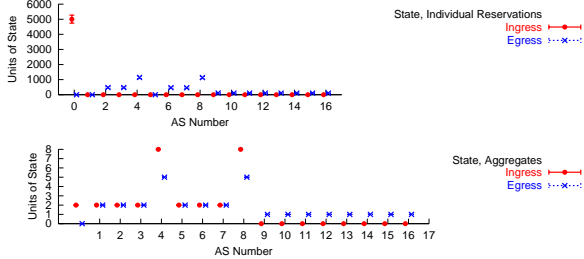
(b) BPS
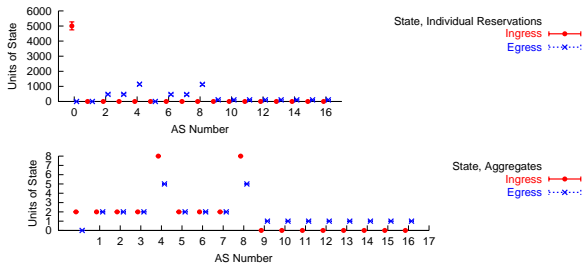
(c) WDS

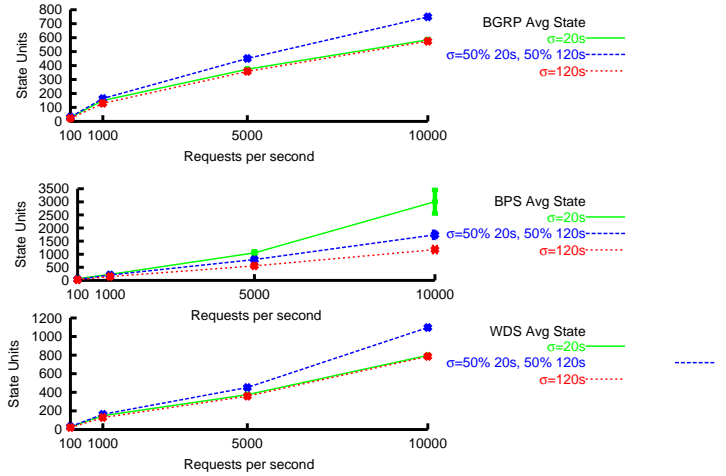**Figure 8. Sink-tree, $\sigma = 20s$, 5000 requests.**

Figure 9. State variation, HT scenario.

4.3. Internet-like Topologies

In this section, we use simulations to investigate the performance of the algorithms on Internet-like topologies. We first devise a scenario where requests are homogeneously distributed across a chosen topology. The two topologies used in this section were generated by BRITE [1], a topology generator with the ability to create AS level topologies.

Homogeneous Traffic (HT) This experiment exemplifies BGRP, BPS, and WDS behavior in the topology illustrated in Fig. 6 (b), where there are 50 nodes, each representing a different AS. Requests are generated according to the homogeneous method.

Tab. 6 details state per AS and per router in terms of minimum, average, and maximum for the three algorithms, when the intensity of requests is of 5000. Comparing the performance of the three algorithms in terms of state required by the different types of requests, we see that request duration does influence state kept. MLR requests demand the highest state values for BGRP and WDS. BPS, however, is more sensitive to SLR requests. A hypothetical explanation for this behavior is the way requests are generated, on the one hand, and BPS sensitivity to the path size of first requests, on the other: to keep the intensity of requests constant, we generate more SLR requests than either MLR or LLR for the whole simulation duration. Due to the way sources and destinations are placed in the topology, a larger number of requests has more probability of creating more diversified path sizes. Hence, there might be higher probability that the first requests arriving to an AS have different path sizes, thus creating more state variability for BPS. This hypothesis is currently being analysed.

From a general perspective and comparing the ratios $\frac{BPS}{BGRP}$ with $\frac{WDS}{BGRP}$, we observe that state required by BGRP and WDS is the same, which implies that WDS reduces significantly the number of aggregates created. BPS, however, requires much more units of state than the other two algorithms. To better perceive the influence of the intensity of requests in the performance of the algorithms, we repeat this experiment, while increasing the intensity of requests. The variation of average state for different values of σ is plotted in Fig. 9. Comparing the three charts of the figure, it is visible that BGRP and WDS require the same average state up to an intensity of 5000 requests. However, when the number of requests increases to 10000, WDS state also increases, especially for MLR requests.

To understand better the impact of the intensity of requests in state required, we present next two hotspot cases for the topology illustrated in Fig. 6 (c), a source and a destination hotspot.
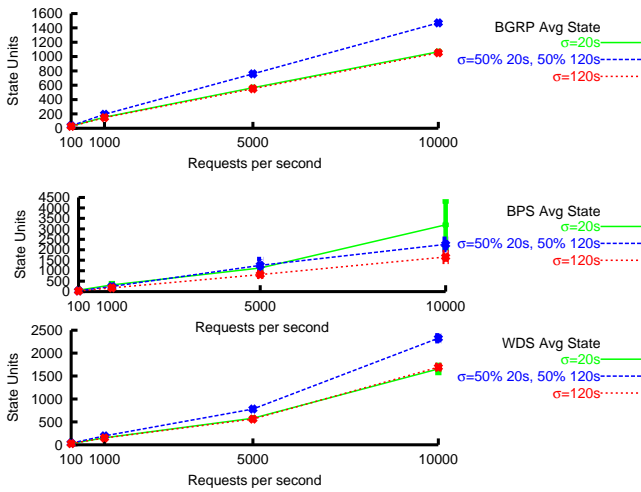

Figure 10. State variation, SH.

Source Hotspot (SH) The source hotspot scenario is emulated by having 60% of the requests starting in a random hotspot AS. Remaining requests are created by the homogeneous traffic distribution method. This experiment represents a possible source-tree scenario in a realistic environment. We repeated this experiment while varying the intensity of requests, to observe state variation. Fig. 10 depicts the average state required per AS. A first observation is that global state is higher than in the homogeneous scenario, which is due to the fact that the topology is smaller than the previous one, increasing the average state per AS. A second observation is that while BGRP average state curves for each request type evolve linearly with the increase of the
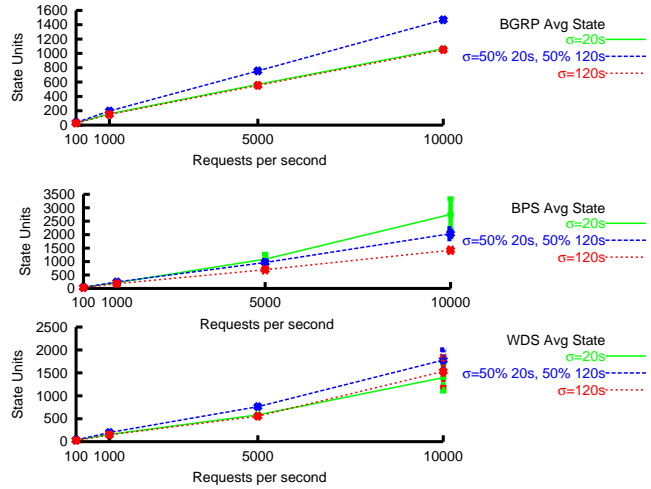

Figure 11. State variation, DH.

number of requests, for WDS, state varies more abruptly with the intensity of requests, once again especially for MLR requests, the type of requests that diversifies more the choice of path sizes. Also, BPS performance is now closer to the performance of WDS, even though BPS has more state variability in terms of maximum, minimum and average, especially for requests of type SLR. This implies that BPS and WDS have in average the same number of intermediate deaggregators, for this scenario.

Destination Hotspot (DH) In this scenario, a node chosen randomly receives 60% of the requests. The remaining 40% requests are placed by using the homogeneous traffic distribution. We repeated the simulation while varying the intensity of requests and Fig. 11 plots the results obtained in terms of state variation. In comparison to the previous scenario, there is a decrease of average state for WDS and BPS, while BGRP requires approximately the same state. For instance, when there are 10000 requests in the system, BGRP requires approximately 1600 state units both for the source and the destination hotspot scenario, when requests are of type MLR, while for both SLR and LLR requests it requires approximately 1000 state units. As for WDS, the state required by requests of type SLR and LLR in both scenarios is similar to the state required by BGRP, while for MLR requests WDS requires slightly more state in the source hotspot scenario than in the destination one.

5. Summary and Conclusions

In this paper, we investigated different aggregation approaches for inter-domain control aggregation, with the aim

Table 6. Average state for the homogeneous traffic scenario, 5000 requests.

σ	SCOPE	VARIABLE	BGRP (Avg/ 95% CI)			BPS (Avg/ 95% CI)			WDS (Avg/ 95% CI)			$\frac{BPS}{BGRP}$	$\frac{WDS}{BGRP}$
20s	AS	Min	272.79	271.67,	273.91	746.62	710.26,	782.99	273.24	272.08,	274.40	2.74	1.00
		Avg	373.95	373.03,	374.87	1052.27	995.32,	1109.21	374.44	373.43,	375.45	2.81	1.00
		Max	475.39	473.08,	477.70	1391.16	1305.41,	1476.91	476.44	473.83,	479.04	2.93	1.00
	Router	Min	34.10	33.96,	34.24	93.33	88.78,	97.87	34.16	34.01,	34.30	2.74	1
		Avg	46.74	46.63,	46.86	131.53	124.42,	138.65	46.81	46.68,	46.93	2.81	1
		Max	59.42	59.13,	59.71	173.90	163.18,	184.61	59.55	59.23,	59.88	2.93	1
50% 20s 50% 120s	AS	Min	381.35	378.30,	384.41	644.48	618.53,	670.43	382.48	379.33,	385.64	1.69	1.00
		Avg	450.91	448.17,	453.65	793.14	759.83,	826.44	452.35	449.39,	455.31	1.76	1.00
		Max	520.49	517.93,	523.04	941.26	902.86,	979.66	522.46	519.89,	525.04	1.81	1.00
	Router	Min	47.67	47.29,	48.05	80.56	77.32,	83.80	47.81	47.42,	48.20	1.69	1
		Avg	56.36	56.02,	56.71	99.14	94.98,	103.31	56.54	56.17,	56.91	1.76	1
		Max	65.06	64.74,	65.38	117.66	112.86,	122.46	65.31	64.99,	65.63	1.81	1
120 s	AS	Min	298.06	295.12,	301.01	448.43	428.09,	468.78	298.62	295.66,	301.58	1.50	1.00
		Avg	358.13	355.98,	360.28	552.50	528.60,	576.40	358.71	356.53,	360.88	1.54	1.00
		Max	418.70	416.43,	420.97	657.83	627.72,	687.94	419.53	417.17,	421.89	1.57	1.00
	Router	Min	37.26	36.89,	37.63	56.05	53.51,	58.60	37.33	36.96,	37.70	1.5	1
		Avg	44.77	44.50,	45.03	69.06	66.07,	72.05	44.84	44.57,	45.11	1.54	1
		Max	52.34	52.05,	52.62	82.23	78.47,	85.99	52.44	52.15,	52.74	1.57	1

of gaining greater insight into the scalability of different inter-domain aggregation procedures. As utility function, we considered the minimization of state kept per AS and per edge router. We evaluated two basic aggregation approaches, sink-tree and shared segment based, and three algorithms derived from these approaches. BGRP follows the sink-tree approach, and BPS as well as WDS follow the shared path segment approach.

A major conclusion to draw from this investigation is that the sink-tree approach represents a reasonable solution in terms of minimizing state maintained. It is also of reasonable complexity and of low sensitivity to the intensity of requests. However, it does not optimize the number of aggregates it creates, since algorithms based on the shared segment approach reduce significantly the number of aggregates when compared with BGRP.

As ongoing work, we are investigating whether the use of multi-level aggregation can lower the sensitivity of the shared-segment approach to traffic intensity. We are also devising an algorithm based on the shared segment approach that avoids keeping state of individual reservations in intermediate deaggregation locations, and we are investigating the bandwidth efficiency and overall signaling load of these different approaches.

References

[1] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byersers. BRITE: An Approach to Universal Topology Generation. In *MASCOTS '01*, Aug. 2001.

[2] Anja Feldmann, Anna C. Gilbert, Polly Huang, and Walter Willinger. Dynamics of IP traffic: A study of the role of variability and the impact of control. *SIGCOMM'99*, 1999.

[3] Bob Braden, David Clark, and Scott Shenker. Integrated Services in the Internet Architecture: an Overview. *Request for Comments 1633, Internet Engineering Task Force*, June 1994.

[4] Bob Braden, Lixia Zhang, and Sugih Jamin. Resource Reservation Protocol (RSVP) - version 1, Functional Specification. *Request for Comments 2205, Internet Engineering Task Force*, Sept. 1997.

[5] Luca Degrossi and Louis Berger. Internet Stream Protocol Version 2 (ST2). *Request for Comments 1819, Internet Engineering Task Force*, Aug. 1995.

[6] Ping Pan, Ellen Hahne, and Henning Schulzrinne. The Border Gateway Reservation Protocol (BGRP) for Tree-Based Aggregation of Inter-Domain Reservations. *Journal of Communications and Networks*, June 2000.

[7] Raj Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, Inc., 1991.

[8] Roch Guérin, Shai Herzog, and Stephen Blake. Aggregating RSVP-Based QoS Requests. *Internet Draft, Internet Engineering Task Force*, Sept. 1997. Work in Progress.

[9] Rute Sofia, Roch Guérin, and Pedro Veiga. An Investigation of Inter-Domain Control Aggregation Procedures. Technical report, ESE, University of Pennsylvania and DI, University of Lisbon (reference TR-08-7), July 2002.

[10] Steve Uhling and Olivier Bonaventure. Implications of Inter-domain Traffic Characteristics on Traffic Enginering. Technical report, University of Namur, June 2001.

[11] Steven Berson and Subramaniam Vincent. Aggregation of Internet Integrated Services State. *Internet Draft, Internet Engineering Task Force*, Aug. 1998. Work in Progress.

[12] Steven Blake, David Black, Mark Carlson, Elwyn Davies, Zheng Wang, and Walter Weiss. An Architecture for Differentiated Services. *Request for Comments 2475, Internet Engineering Task Force*, Dec. 1998.

[13] Telstra. BGP Table Report. http://bgp.potaroo.net/, Feb. 2001.

[14] Vern Paxson and Sally Floyd. Why We Don't Know How to Simulate the Internet. *Winter Simulation Conference*, 1997.

[15] VINT Project. *The ns Manual*. UC Berkeley, LBL, USC/ISI, Xerox Parc, Sept. 2001.

[16] Walter Willinger and Vern Paxson. Where Mathematics meets the Internet. *Notices of the American Mathematical Society, 45(8)*, Aug. 1998.

[17] Yakov Rekhter and Tony Li. A Border Gateway Protocol 4 (BGP-4). *Request for Comments 1771, Internet Engineering Task Force*, Mar. 1995.