SICAP, A Shared-segment based Inter-domain Control Aggregation Protocol

Rute Sofia[1] Roch Guérin[2] Pedro Veiga[3]

Abstract

Existing Quality of Service models, being well defined in the data path, lack an uniform end-to-end control path mechanism capable of guaranteeing the required resources to bandwidth intensive services, such as video streaming. Current reservation protocols represent partial solutions to this problem, since they are able to provide scalable resource reservation inside routing domains. However, it is primarily between domains that scalability becomes a major issue, since inter-domain links experience large volumes of reservation requests. As a possible solution, we present and evaluate the *Shared-segment based Inter-domain Control Aggregation Protocol* (SICAP), which affords the benefits of shared-segment aggregation, while avoiding its major drawback, namely, its sensitivity to the intensity of requests [9]. We present results of simulations that compare the performance of SICAP against that of the *Border Gateway Reservation Protocol,* (BGRP) which relies on sink-tree aggregation to achieve scalability.

I. Introduction

Quality of Service (QoS) is a field that has given rise to a wide range of works that investigate data path mechanisms. This includes the *Integrated Services* [7] and the *Differentiated Services* [11] models. Nevertheless, no QoS model can be fully deployed without an adequate control path mechanism, capable of providing efficient resource management to the booming and diversified Internet multimedia-based services. Currently, protocols such as the *Resource Reservation Protocol (RSVP)* [8] or the *Yet anothEr Sender Session Internet Reservation protocol* (YESSIR) [6], scale well when used to reserve resources inside regions that share the same routing policies, i.e., *Autonomous Systems* (AS's). However, it is between AS's that scalability becomes a major issue, since inter-domain links are likely to experience high intensity of reservation requests. One might argue that these links can be over-provisioned to eliminate the need for reservations. Still, over-provisioning is not cost-effective for all providers, and furthermore, it requires AS boundary routers (BR's) to be able to cope with high volumes of requests, which translates into significant memory and processing costs.

Control state aggregation is another option that can be used to reduce the information kept in each router along a path: instead of keeping state per request, routers keep only state per group of requests, i.e., per *aggregate*. Hence, the granularity chosen to perform aggregation is a key factor in determining the state reduction that can be achieved. Aggregation could, for instance, be done on the *flow level*, per source and destination IP addresses. But, according to Huston [3] there were around 1.09 billion addresses visible in the Internet routing table in 2001, which translates into 10^{12} possible combinations of active IP addresses and, consequently, with an aggregation scheme that may not scale. Alternatively, aggregation could be based on groups of aggregated IP addresses [14], i.e., *network prefixes*, which could reduce state along a path, but possibly not substantially, since such scheme depends on how addresses are distributed over route prefixes, and on how routes are aggregated through each AS. A far better option is to aggregate reservations at the *AS level*, given that AS's are the basic building block of current Internet routing infrastructure: being the current Internet organized in AS's and considering that the information exchanged by the BGP protocol [15] provides what is necessary to obtain the smallest path according to providers' agreements, aggregation on the AS level seems the most appropriate. From a scalability standpoint, since there are currently 13,000 active AS's on the Internet [1], this represents a much smaller universe than the billions of active IP addresses.

Our goal is two-fold. First, we aim to describe the design of SICAP, a protocol based on a shared-segment aggregation approach, and second, to show that SICAP achieves reasonable performance improvements when compared to BGRP. Hence, the remainder of the report is structured as follows: section II presents related work. Section III gives an operational example of BGRP. Section IV presents the SICAP protocol in detail, and section V gives a comparison of SICAP and BGRP performance. Finally, section VI presents conclusions and future work.

(1) rsofia@seas.upenn.edu, ESE, University of Pennsylvania, Philadelphia, PA 19104-6390 and Department of Informatics, University of Lisbon.
(2) guerin@ee.upenn.edu, ESE, University of Pennsylvania.
(3)pmv@di.fc.ul.pt, Department of Informatics, University of Lisbon.

II. Related work

Pan et al. [5] present an inter-domain signaling protocol, BGRP, which merges requests that have the same destination *AS*, creating aggregates in the form of sink-trees. Pan et al. show that BGRP has good performance when compared with RSVP without aggregation, but they do not provide a comparison of BGRP with other possible inter-domain aggregation mechanisms, partially because no other proposal had been put forward at the time.

Sofia et al. [9] present a comparison of the shared-segment and the sink-tree approach. By means of simulations, they compare algorithms that illustrate the behavior of the two proposals, showing that the shared-segment approach has a higher total state cost than the sink-tree approach, because of its sensitivity to the intensity of requests. However, they also show that the shared-segment approach reduces the number of aggregates created, when compared with the sink-tree approach.

Our work builds on the former work by developing a protocol, SICAP, that implements a number of enhancements to the basic shared-segment algorithms of [9] and that eliminates most if not all of their previous drawbacks. In particular, because SICAP is able to avoid the sensitivity of the shared-segment approach to the intensity of requests, it brings out the full benefits provided by that aggregation approach.

In the next section, we give an example of BGRP, before proceeding with a detailed description of SICAP.

III. BGRP operational example

BGRP is an inter-domain control aggregation protocol that is *sender-initiated* in the sense that the first BR on the path triggers reservation requests. BGRP merges requests that have the same destination AS, creating aggregates shaped as sink-trees, being the destination AS's their roots. This allows BGRP to greatly reduce the amount of state required at BR's along a path, when compared with a mechanism that does not perform aggregation.

To establish a reservation, BGRP uses a *two-phase* mechanism: in the first phase, the path is probed with a PROBE message sent from the *first-aggregator*, i.e., the first egress router on the path, to the *last-deaggregator,* i.e., the last ingress router on the path. In the second phase, the last-deaggregator uses the information gathered by the PROBE to choose the aggregate into which it will merge the reservation. It then sends a GRAFT message that allocates the necessary resources along the path traversed by the earlier PROBE message. Using such scheme avoids the consequences of having to deal with *asymmetric paths*, i.e., following a path from AS B to AS A that is different from the path followed when going from AS A to AS B: probing the path prior to allocating resources, ensures that a reservation is established on the reverse path earlier probed from sender to destination AS.

The aggregates created by BGRP have *soft-state,* i.e., their information is periodically refreshed by BR's through the use of REFRESH messages. BGRP also uses optional TEAR messages, that routers can send to explicitly remove reservations.

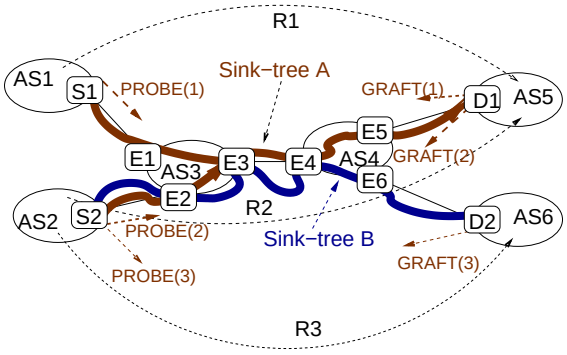


Fig. 1. BGRP example.

To illustrate how BGRP works, we use the scenario shown in Fig. 1, where R_1 represents a reservation request originated in AS 1 and destined to an end-host in AS 5. When router $S1$ receives R_1, it sends $PROBE(1)$, which contains the request identifier R_1, the source identifier $S1$, the identifier of R_1 destination, the bandwidth requirement $b_{\{S1,E1\}}$, where $\{i, j\}$ represents the link between BR's i and j, and an empty route record. $PROBE(1)$ goes through $E1$, $E3$, $E4$, and $E5$, each of which inserts its identifier in the route record. $PROBE(1)$ stops in case of error, or when it reaches the last-deaggregator, $D1$. If it fails to reach $D1$, an ERROR message is sent back to $S1$ by the router where the failure occurred. If it reaches $D1$, this router replies with $GRAFT(1)$, which contains the same information as the $PROBE(1)$, along with a label A that uniquely identifies the sink-tree whose root is $D1$. $GRAFT(1)$ will establish

the reservation along $E1$, $E3$, $E4$, and $E5$. If a request R_2, which is originated in AS 2 and also destined to an end-host in AS 5 arrives at $S2$, then this router sends $PROBE(2)$, containing the identifier R_2 and bandwidth $b_{\{S2,E2\}}$. When this message arrives at $D1$, it replies with $GRAFT(2)$, that will increment in $b_{\{S2,E2\}}$ units the bandwidth allocated to the tree A, until $E3$. From $E3$ to $S2$, $GRAFT(2)$ triggers the creation of a new branch of A, allocating for it $b_{\{S2,E2\}}$ units.

Let's now suppose that a request R_3, originating again in AS 2, is destined to AS 6. When $PROBE(3)$ reaches $D2$, the last-deaggregator on the path of R_3, this router triggers the creation of a new sink-tree, B, that extends all the way to $S2$ and is independent of the tree A even over their common segments. This simple example illustrates both the operation of BGRP and a specific instance where it does not result in the minimum possible amount of state. In the example, routers $S2$, $E2$, $E3$, and $E4$ have to keep state for trees A and B, even though both trees share that segment of the path. The shared-segment approach developed in [9] and on which SICAP relies, is an attempt at further reducing the amount of state in the presence of such shared path segments. In the next section, we describe the design of SICAP and how this protocol is able to take advantage of shared path segments to reduce the number of aggregates created.

IV. SICAP Design and Operation

SICAP, like BGRP, is sender-initiated and uses a two-phase mechanism to establish reservations. The information collected during the probing phase is used to decide how to aggregate, as explained next. The information collected during the probing phase is used to decide how to aggregate, as explained next.

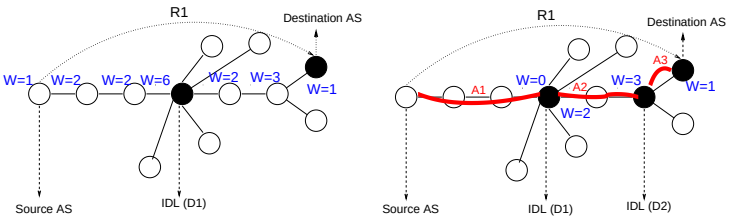
A. Deaggregator Choice Algorithm

SICAP uses an enhanced version of the *Weighted Deaggregation pointS* (WDS) [10] algorithm to decide how to aggregate. WDS assumes that AS's with a large number of downstream neighbor AS's are more suitable as aggregate end points, since those AS's are more likely to be reservation *hot spots*, i.e., they might experience higher intensities of requests. For each AS m of a path, WDS computes a weight W equal to the number of downstream neighbors of m, n_m:

$$W_m = n_m, \forall m \tag{1}$$

There are two particular 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 $n_m = 1$. The AS that yields the largest weight is selected as an *intermediate deaggregation location (IDL)*.

Fig. 2, where each node represents an AS, exemplifies how WDS works. When the last-deaggregator at the destination AS receives request R_1, it computes the weight of each AS on the path. As shown in Fig. 2 (a), the AS yielding the heaviest weight is D1, which becomes the first IDL. To increase the probability that requests coming from different source AS's will use aggregates already established, the process is repeated recursively between each IDL and the destination AS. Fig. 2 (b) shows the second and final iteration for the segment between D1 and the destination. Therefore, in the given example, WDS triggers the creation of three different aggregates: the first extends from the source AS to D1; the second extends from D1 to D2; the third goes from D2 to the destination AS.



(a) First IDL (b) Second IDL

Fig. 2. WDS example.

The information used to make decisions on where to place IDLs is carried through SICAP messages, which we present next together with several examples of messaging sequences.

B. Messages

SICAP defines five message types, which carry both basic and additional information. all of which contain a request identifier, the request destination, the bandwidth required, the type of message, and a timestamp. Additionally, each message might carry other information, as described next:

- **REQ**. These messages are sent by first-aggregators to probe network resources. Along the path, each BR adds its identifier to the REQ message. When a REQ reaches a destination AS, it carries the list of BRs traversed, the *route record*, and resources required. The route record has a variable size, since it depends on the number of routers crossed. According to current Internet statistics [13], the current average path size is of five AS's and the maximum is of eleven AS's. Therefore, in average the route record will have a size of seven, and a maximum size of twenty, since the first and last AS only contribute with one BR each.
- **RESV**. RESV messages are sent upstream by the last-deaggregator of a path, as a reply to a received REQ message and to allocate the required resources. The RESV contains the information of the corresponding REQ, and an aggregate label that identifies the aggregate into which the reservation will be merged.
- **ERROR** messages are used in case of reservation failure. If a reservation is rejected, an error message of sub-type REJ is sent upstream, to notify the first-aggregator of the rejection. If a reservation fails, not due to resources or link failure, but because the corresponding aggregate state was deleted, a generic ERROR message is sent downstream, to notify the next router in the path that the reservation should be retried.
- **TEAR**. TEAR messages are triggered by the source of a reservation to delete it along a path. The TEAR travels downstream, deleting the corresponding reservation at each BR.
- **REFRESH**. These messages update the information regarding reservations along a path. They are sent periodically each T_r seconds by any first-aggregator, and travel downstream until they reach a last-deaggregator. They carry the same information that a RESV message contains, and their purpose is to detect inconsistencies, such as message loss, node failures, or path changes. By default, T_r is set to 30s, since this is the default value for the BGP timer *KeepAlive*. If a router does not receive a REFRESH message for an aggregate after 90s, the default value of the BGP timer *HoldTime*, it will delete the aggregate state.

These five different types of messages are used in different situations, to which we provide examples next.

C. SICAP Operation

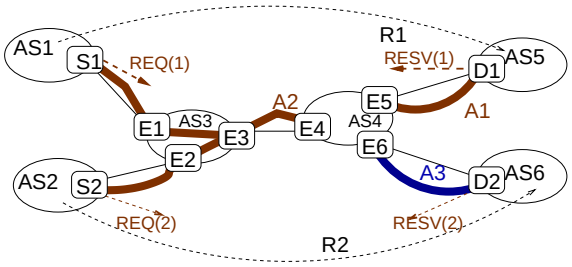


Fig. 3. SICAP example.

To illustrate how SICAP works, we use the scenario of Fig. 3, where ellipses represent different AS's, S_i is the first-aggregator and Di is the last-deaggregator on the path of reservation R_i. The figure shows two reservations: R_1 is a reservation between an end-host in AS 1 and an end-host in AS 5, and R_2 is a reservation between an end-host in AS 2 and an end-host in AS 6. We consider three scenarios: the first deals with the establishment of reservations R_1 and R_2, the second describes the deletion of reservation R_1, and the third illustrates a possible exchange of error messages in the case of a failure of reservation R_1.

1) End-to-End Reservation Establishment: To establish a reservation, SICAP uses a pair of REQ/RESV messages: each time a reservation request arrives to a first-aggregator, the SICAP protocol sends a REQ to probe the path, until it reaches a last-deaggregator. REQ messages are sent downstream, hop-by-hop. Similarly to the BGRP case, this represents the first phase of the reservation establishment, which ends when a REQ message either reaches a last-deaggregator, or when it reaches a point of failure. For the latter case, a REJ message is sent upstream directly to the first-aggregator: since the reservation has not been established yet, intermediate routers do not need to be notified that the reservation was rejected.

The reception of a REQ message by a last-deaggregator triggers the second phase of a reservation establishment. The last-deaggregator starts by choosing as an IDL the AS of the path that holds the heaviest W_m. It then looks for an aggregate that ends in the current AS, and that either starts or crosses the IDL AS. If such an aggregate exists, its resources are updated by a RESV message sent by the last-aggregator: at each BR, the RESV triggers the creation or the update of the aggregate. When the RESV arrives at the aggregate source, this router will calculate again which is the next intermediate deaggregation location, and the aggregate that provides access to that AS.

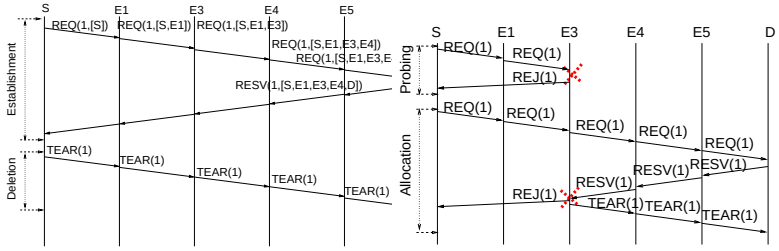


Fig. 4. SICAP messaging.

Fig. 4 shows the message exchange required to establish R_1, for the scenario illustrated in Fig. 3. To start the establishment of R_1, $S1$ sends $REQ(1)$ to the end-host in AS 5. $REQ(1)$ contains the reservation identifier R_1 and its bandwidth requirement, $b_{\{S,E1\}}$, where $\{i,j\}$ represents the link between routers i and j. $REQ(1)$ travels through $E1$, $E3$, $E4$, and $E5$, which add their identifiers to the route record of the message. When $D1$ receives $REQ(1)$, it realizes that the request ends in AS 5 and therefore, uses the information collected to choose the aggregate that R_1 will be merged into. Because there is no adequate aggregate, $D1$ triggers the creation of a new aggregate, A_1, and selects $E5$ as its starting point. To establish A_1, $D1$ sends $RESV(1)$, requesting $b_{\{S,E1\}}$ on each link of the reverse path provided by $REQ(1)$. When $RESV(1)$ arrives at $E5$, the aggregate label is reset and $RESV(1)$ is sent to the previous-hop, $E4$. $E4$ looks for an aggregate that might carry $R1$ until $S1$. Not finding any, $E4$ triggers the creation of another aggregate, A_2, that extends all the way from $S1$ to $E4$, and updates the aggregate label in $RESV(1)$ to A_2. If $RESV(1)$ succeeds in reaching $S1$, then the reservation is established.

Let us now consider a request R_2 originating in AS 2 and destined to AS 6: when $D2$ receives $REQ(2)$, it triggers the creation of aggregate A_3 to $E6$, and sends $RESV(2)$ to establish the reservation. When $RESV(2)$ arrives at $E4$, this router realizes that R_2 can be merged into the existing aggregate A_2, and therefore simply updates the resources of A_2. However, because A_2 heads towards AS 1 and not AS 2, a new aggregate branch A_4 is created from $E3$ to $S2$. This branch is directly merged into A_2 at $E3$, so that $E3$ is not an IDL where individual reservation state would have to be kept, but simply a merging point. This example shows how the shared-segment aggregation approach can reduce the number of aggregates created, therefore reducing state along a path: from AS 2, two reservations, R_1 and R_2, which have different destination AS's, reuse the same aggregate over the path segment they share.

2) Reservation Deletion: The explicit deletion of a reservation, whether it is individual or aggregate, is carried out by a TEAR message. The deletion of an individual reservation is done from the first-aggregator to the last-deaggregator as a consequence of an end-host request, and it represents an update of resources to an established aggregate. However, the deletion of an aggregate A is only triggered by its source, when its bandwidth is equal to zero, $B_A = 0$, i.e., when the aggregate is *empty*. The TEAR message is then sent downstream, either until it reaches the destination of the aggregate, or until it reaches a BR where $B_A > 0$, which implies that reservations are being merged into the aggregate A, and therefore, the TEAR stops.

To delete R_1, $S1$ sends $TEAR(1)$, which carries the reservation identifier R_1 and also $b_{\{S,E1\}}$. Between $S1$ and $E4$, each router decreases the bandwidth of A_2 in $b_{\{S,E1\}}$ units. When $TEAR(1)$ reaches $E4$, the aggregate field is reset, and $TEAR(1)$ is forwarded to the next-hop, $E5$. This router knows that R_1 is mapped to aggregate A_1 and therefore updates the aggregate label of $TEAR(1)$ to A_1. $E5$ then decreases the bandwidth of A_1 by $b_{\{S,E1\}}$ units.

3) Reservation Failure: As shown in Fig. 4 (b), a reservation failure can occur in either any of the two phases of the establishment of R_1. We first consider a failure during the probing phase of R_1, and assume that when $REQ(1)$ reaches $E3$, this router realizes that there are not enough resources to satisfy $R1$. Hence, $REQ(1)$ is stopped and $REJ(1)$ is

sent towards $S1$ to notify this router of the failure, so that it can release the associated resources: BR's between $S1$ and $E3$ do not have yet any state related with R_1, since the failure occurred during the probing phase. Such is the case of $E1$, which simply passes $REJ(1)$ to $S1$. We now consider the case of a failure during the allocation phase of R_1, and assume that when $RESV(1)$ reaches $E3$, this router notices that there are insufficient resources on link $\{E1, E3\}$. As a result, $E3$ not only sends a $REJ(1)$ message towards S_1, but it also needs to delete the partially established reservation towards $D1$. This is accomplished by sending a $TEAR(1)$ message towards $D1$.

4) IDL State Management: In the shared-segment aggregation approach, and as explained in the previous section, aggregates might not extend all the way until the destination of some of the individual reservations they carry. Instead, they may end at an IDL AS. At IDL's, reservation requests have to be switched from an ending aggregate at the ingress router, to a new aggregate at the egress router. Therefore, aggregators[1] at an IDL have to keep track of the mapping between individual reservations and aggregates. One way to achieve this is to keep each reservation identifier and resources at the aggregator. However, this solution incurs a significant overhead in the amount of state that must be kept [9]. SICAP avoids this state penalty by keeping track of the mapping between aggregates and reservations at the level of destination AS's, rather than explicitly mapping individual reservations to aggregates. In other words, SICAP maintains per aggregate a list of the destination prefixes advertised by the AS's an aggregate provides access to. As an example of how such information can be used to efficiently manage reservations, we address again the scenario illustrated in Fig. 3. During the establishment of R_1, and when $REQ(1)$ arrives at $D1$, this router looks for the most specific advertised prefix that matches R_1 destination address. $D1$ then inserts the found prefix(es) in the $RESV(1)$ message. When this message arrives at $E5$, SICAP updates the list of destination prefixes of A_1, adding to that list the prefix(es) contained in $RESV(1)$. When R_1 gets torn down and $TEAR(1)$ arrives at $E5$, this router simply looks up the most specific prefix that matches the destination address carried by $TEAR(1)$, at the set of destination prefixes kept per aggregate, and finds out that A_1 contains the most specific match. Therefore A_1 resources can be updated without mapping explicitly R_1 to A_1. The state cost of this solution depends mostly on the number of prefixes each AS advertises. Broido et al. [1] present measurements of the Internet routing table, where from a possible universe of 12,399 AS's, the majority of AS's advertised a maximum of 99 prefixes[2], which is a reasonable number, when compared to the much larger number of individual reservations crossing BR's.

D. Enhancements

This report deals with SICAP in terms of its basic operational behavior, with the purpose of analyzing the ability of SICAP to establish and manage reservations, while achieving state scalability. However, SICAP can be enhanced to support some particular situations, only adding some minor modifications, which we explain in this section.

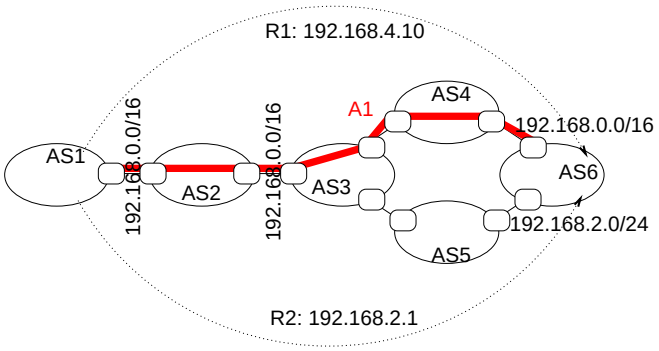
1) Dealing with Bi-directional Reservations: As mentioned before, the two-step reservation establishment scheme that SICAP and BGRP use avoids dealing with asymmetric paths when requests are sent from sender to destination. However, some Internet services, such as *Voice over IP*, *VoIP*, might require reservation updates on behalf of receivers, i.e., a *bi-directional* reservation behavior. Those updates have to travel on the reverse path of the reservation, but current Internet routing does not guarantee symmetric paths. We give an example where a sender-initiated VoIP reservation R_1 has already been established, and is mapped to an aggregate A_1. If R_1 receiver asks for an update of R_1, a message will have to be sent from R_1 receiver to R_1 sender. This might be a problem, since there is no guarantee whatsoever that the request sent by the receiver will travel on the reverse path of A_1. For such a scenario, SICAP uses a similar solution to the one provided by the use of the PHOP object in RSVP: SICAP keeps for each aggregate and per BR the identifier of the previous router on the aggregate path. But, this is not sufficient to ensure that receiver-based requests can be correctly redirected when they reach the starting point of an aggregate which is not a first-aggregator, i.e., an aggregator at an IDL: when the reservation request arrives at the corresponding aggregate source, it has to determine which is the appropriate previous-hop on the reservation path, i.e., the adequate intermediate deaggregator. Thus, at the last-deaggregator, each individual reservation is associated with the identifiers of the intermediate deaggregators crossed on the reservation path. In [10], we reached the conclusion that WDS gives rise in average to three IDL's. Consequently, and if SICAP associates each reservation with the identifiers of the deaggregators crossed in its path only at last-deaggregators, it can support bi-directional reservations while avoiding possible scalability issues.

[1]Note that except for the deaggregator in the destination AS, deaggregators do not need to keep track of individual requests, since no reservation or forwarding state is maintained at the deaggregator.

[2]Their measurements of the Internet routing table in December 2001, show that from a possible universe of 12,399 AS's, 40% announced only one prefix. These prefixes however, represented only 4.9% of 102,394 prefixes. The data also sustains that the number of AS's advertising over 100 prefixes is only 1%.

2) Dealing with IP Address Space Overlapping: Both BGRP and SICAP aim at reducing the state cost when compared with a solution that does not perform aggregation. Therefore, and as mentioned, because we are comparing BGRP and SICAP in terms of their regular operation mode, we concentrate on the state cost of SICAP and how it compares to that of BGRP. But, another key factor that may affect the scalability of a control aggregation solution is the frequency of updates to an aggregate, i.e., the *signaling load*. In the regular operation mode, both BGRP and SICAP attain the same signaling load, which is not reduced when compared to a solution that does not perform aggregation, because they update aggregates *per* individual reservation. A possible way of reducing the signaling load of an aggregation solution is to perform *over-reservation*, i.e., provide an aggregate with more bandwidth than the required at a certain instant, thus reducing the frequency of messages exchanged and accordingly, the required processing in BR's, at the expense of allocating resources in advance to aggregates. Besides the need for resources in advance, an over-reservation mechanism also requires a way to identify an already established and sufficiently provisioned aggregate as the adequate one to merge the reservation into: such identification is easily performed by SICAP, since its regular operation mode already requires the mapping of each aggregate with a set of the destination network prefixes the aggregate provides access for. BGRP also supports this enhancement, named *quiet grafting* and briefly mentioned by Pan et al. [5], but further examined for BGRP by Nikolouzou et al. [2], [4].

We are currently evaluating several over-reservation schemes not only to be used with SICAP, but also with BGRP. However, and because in the context of this report we are evaluating the basic operation of these protocols, in this section we simply address a situation that may give rise to inconsistencies in the case of over-reservation, due to the overlapping of destination IP prefixes, and how SICAP deals with it. That situation is illustrated in Fig. 5, where R_1 and R_2 are reservation requests originated in AS1 but with IP destination addresses 192.168.4.10, and 192.168.2.1, respectively. The destination AS for both these requests is AS 6, which advertises two prefixes for two different AS paths: 192.168.0.0/16 for the AS path $\{AS1, AS2, AS3, AS4, AS6\}$, and 192.168.2.0/24 for the AS path $\{AS1, AS2, AS3, AS5, AS6\}$. We assume that an aggregate $A_1$is already in place from AS 1 to AS 6, and that the prefix 192.168.0.0/16 is mapped to A_1. In the regular SICAP operation mode, R_1 would be mapped into A_1, and R_2 would originate the creation of a new aggregate on the AS path $\{AS1, AS2, AS3, AS5, AS6\}$, when the respective REQ messages would reach the last-deaggregator. However, with over-reservation and assuming that A_1 has sufficient resources to automatically provision $R_1$and R_2 at AS 1, the overlapping of the prefixes advertised by AS 6 would generate an error situation: in AS 1, R_1 would be correctly merged into A_1. However, because in AS 1 the prefix mapped to A_1, 192.168.0.0/16 is the most specific match to the destination address of R_2, this reservation would also be merged into A_1, when in fact it should follow a different path.



(a)

Fig. 5. Overlapping IP Prefix situation.

To avoid this situation, we follow the solution [4] used in the context of BGRP quiet-grafting mechanism. That solution is to exclude possible prefixes that are contained in a most specific match, but that are not a match to the required IP destination address. In the given example, that would mean to exclude the subnet 192.168.2.0/24 from 192.168.0.0/16, thus avoiding the incorrect mapping of A_1 to the prefix 192.168.2.0/24 in AS 1. The drawback of this solution is the need to search for more specific prefixes that are not a match to a certain IP address, each time a request reaches a

last-deaggregator, and the consequent computational cost.

V. BGRP AND SICAP PERFORMANCE COMPARISON

There are several possible measures of efficiency that can be used to evaluate the ability of an inter-domain signaling protocol in reducing memory occupancy and processing cost at BR's. This cost is related to the number of aggregates that are maintained and to how often their *state* and bandwidth needs to be updated, which translates into the *bandwidth efficiency* and consequent *signaling load* of a solution. In this report, we assume the context of the regular mode of operation for both BGRP and SICAP, where the bandwidth of an aggregate is updated per individual request, so that both protocols attain the same signaling load. Therefore, the performance parameter left to focus on is state, since this is the only efficiency parameter where these protocols may differ.

To quantify state, we consider that a reservation occupies one unit of state each time it crosses a BR interface: if x individual reservations are mapped into one aggregate, the corresponding state is $x + 1$ units; when an aggregate that contains x reservations is deaggregated, the state occupied is $1 + x$; if an aggregate is in transit when crossing an AS, it requires four units of state, two at the ingress and two at the egress BR.

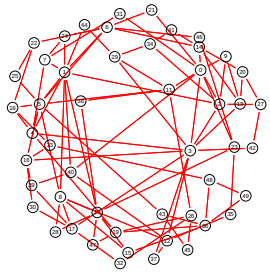


Fig. 6. Internet-like topology

To analyze state, we run an experiment first introduced in [9], so we can see if and how SICAP is able to reduce state by not keeping information about individual reservations at IDL's. The experiment uses the AS level topology with 50 nodes illustrated in Fig. 6, and a distribution of requests where each node has the same probability of being a source, and destinations are placed according to a real distribution of addresses based on AS distance [12]. The arrival of requests is modeled as a Poisson process with mean inter-arrival time of σ. In order to exemplify three possible cases of requests, and to achieve a consistent comparison of the performance of the protocols, the duration of requests was varied while keeping the system load constant. Three scenarios were considered: short-lived requests, with an average duration of 20s; long-lived requests, with an average duration of 120s; mixed traffic, from which we generated three possible sub-types: a mix of 20% of short-lived and 80% of long-lived requests; a mix of 50%/50% and a third mix, 80% of short-lived requests and 20% of long-lived requests.

The results presented in Tab. I comprise the minimum, maximum and average state values, calculated within a 95% confidence interval, for an intensity of 5,000 requests. They show that SICAP consistently outperforms BGRP, which confirms the former's ability to reduce state by lowering the number of aggregates created, since the state associated with individual requests is the same for both protocols. The columns containing the state ratio $\frac{SICAP}{BGRP}$ and the state difference $BGRP - SICAP$, show how the duration of requests affect both protocols. For example, the average state ratio is approximately 0.62 for the different types of requests. It should be noticed however, that state varies as a function of the duration of individual requests: short-lived requests require more average state than any of the other types. This phenomenon is merely a consequence of the increased "load" associated with shorter duration requests, i.e., in order to keep the intensity of requests constant while varying the duration, it is necessary to generate more short-lived requests than either mixed or long-lived.

To see in detail how the intensity of requests affects state, the experiment was repeated for an intensity of 10,000 requests, and Tab. I holds the results, which are again consistently lower for SICAP than for BGRP. The state ratio $\frac{SICAP}{BGRP}$ is again representative of how state varies and is usually higher than the ratio obtained for an intensity of 5,000 requests, as a consequence of having more requests per unit of time, i.e., a broader universe. The column presenting the state difference $BGRP - SICAP$ shows better the state reduction achieved by SICAP when compared with BGRP, and is higher for the intensity of 10,000 requests than for the intensity of 5,000. This state reduction does not increase proportionally to the intensity of requests, since it is only related with the number of aggregates created: while the state due to individual requests is the same for both protocols, SICAP reduces the number of aggregates created.

To present a global perspective of how state is affected when the intensity of requests varies, the experiment was repeated for different intensities and Fig. 7 shows the results. Each bar represents the average state value that a protocol requires for a particular type of requests, and for a particular intensity. Note that the difference of state between BGRP and SICAP does not grow proportionally to the intensity of requests, because that difference is only due to the number of aggregates created. However, the difference remains significant in terms of scalability, since state due to individual reservations is only kept at the end-points of a path, but state due to aggregates is kept in each BR crossed.

VI. SUMMARY AND CONCLUSIONS

In this report, we first described the design and operation of an inter-domain aggregation control protocol, SICAP, which performs shared-segment aggregation of reservation requests. We then compared the performance of SICAP with

TABLE I

AVERAGE STATE, INTENSITY OF 5000 REQUESTS

σ	SCOPE	VALUE	BGRP (Avg/95% CI)			SICAP (Avg/95% CI)			$\frac{SICAP}{BGRP}$	$BGRP - SICAP$
20s	AS	Min	250.70	248.35,	253.04	148.00	146.67,	149.32	0.59	102.70
		Avg	361.92	360.22,	363.62	225.21	223.95,	226.48	0.62	136.70668
		Max	473.71	472.15,	475.28	314.30	312.95,	315.65	0.66	159.412
	Router	Min	62.67	62.09,	63.26	37.00	36.67,	37.33	0.59	25.675
		Avg	90.48	90.06,	90.90	56.30	55.99,	56.62	0.62	34.17667
		Max	118.43	118.04,	118.82	78.58	78.24,	78.91	0.66	39.853
50% 20s	AS	Min	272.40	269.01,	275.79	161.43	158.44,	164.41	0.59	110.972
50% 120s		Avg	356.92	355.23,	358.61	220.45	218.77,	222.13	0.62	136.46696
		Max	441.68	439.52,	443.83	285.29	283.20,	287.38	0.65	156.388
	Router	Min	68.10	67.25,	68.95	40.36	39.61,	41.10	0.59	27.743
		Avg	89.23	88.81,	89.65	55.11	54.69,	55.53	0.62	34.11674
		Max	110.42	109.88,	110.96	71.32	70.80,	71.84	0.65	39.097
20% 20s	AS	Min	276.44	273.63,	279.26	164.88	162.45,	167.32	0.6	111.56
80 % 120s		Avg	356.62	354.97,	358.28	220.70	219.35,	222.06	0.62	135.91984
		Max	435.56	433.27,	437.86	280.66	278.72,	282.60	0.64	154.904
	Router	Min	69.11	68.41,	69.81	41.22	40.61,	41.83	0.6	27.89
		Avg	89.16	88.74,	89.57	55.18	54.84,	55.52	0.62	33.97996
		Max	108.89	108.32,	109.46	70.17	69.68,	70.65	0.64	38.726
20 % 120s	AS	Min	264.86	262.88,	266.83	157.33	156.04,	158.62	0.59	107.528
80% 20s		Avg	358.85	356.92,	360.77	222.44	221.11,	223.76	0.62	136.41068
		Max	453.73	451.40,	456.07	296.23	293.98,	298.47	0.65	157.504
	Router	Min	66.21	65.72,	66.71	39.33	39.01,	39.65	0.59	26.882
		Avg	89.71	89.23,	90.19	55.61	55.28,	55.94	0.62	34.10267
		Max	113.43	112.85,	114.02	74.06	73.50,	74.62	0.65	39.376
120 s	AS	Min	277.32	275.50,	279.15	165.18	163.80,	166.56	0.6	112.144
		Avg	355.53	353.34,	357.71	219.79	218.14,	221.45	0.62	135.73284
		Max	431.64	429.09,	434.19	277.32	274.83,	279.82	0.64	154.316
	Router	Min	69.33	68.88,	69.79	41.30	40.95,	41.64	0.6	28.036
		Avg	88.88	88.33,	89.43	54.95	54.53,	55.36	0.62	33.93321
		Max	107.91	107.27,	108.55	69.33	68.71,	69.95	0.64	38.579

TABLE II

AVERAGE STATE, INTENSITY OF 10000 REQUESTS

σ	SCOPE	VALUE	BGRP (Avg/95% CI)			SICAP (Avg/95% CI)			$\frac{SICAP}{BGRP}$	$BGRP - SICAP$
20s	AS	Min	444.04	442.83,	445.25	314.74	313.43,	316.04	0.71	129.30
		Avg	581.25	580.42,	582.08	426.95	426.13,	427.77	0.73	154.29
		Max	713.71	711.74,	715.69	550.58	548.64,	552.52	0.77	163.13
	Router	Min	111.01	110.71,	111.31	78.68	78.36,	79.01	0.71	32.33
		Avg	145.31	145.11,	145.52	106.74	106.53,	106.94	0.73	38.57
		Max	178.43	177.93,	178.92	137.65	137.16,	138.13	0.77	40.78
50% 20s	AS	Min	469.01	464.89,	473.13	333.11	329.51,	336.70	0.71	135.90
50% 120s		Avg	572.91	569.72,	576.10	419.04	416.19,	421.88	0.73	153.88
		Max	674.24	671.75,	676.72	511.32	508.92,	513.73	0.76	162.91
	Router	Min	117.25	116.22,	118.28	83.28	82.38,	84.18	0.71	33.98
		Avg	143.23	142.43,	144.03	104.76	104.05,	105.47	0.73	38.49
		Max	168.56	167.94,	169.18	127.83	127.23,	128.43	0.76	40.73
20% 20s	AS	Min	473.97	470.88,	477.06	336.60	334.53,	338.66	0.71	137.38
80 % 120s		Avg	571.28	569.64,	572.91	417.17	415.67,	418.68	0.73	154.10
		Max	664.29	661.92,	666.66	501.38	499.01,	503.75	0.75	162.91
	Router	Min	118.49	117.72,	119.27	84.15	83.63,	84.67	0.71	34.34
		Avg	142.82	142.41,	143.23	104.29	103.92,	104.67	0.73	38.53
		Max	166.07	165.48,	166.66	125.35	124.75,	125.94	0.75	40.73
20 % 120s	AS	Min	459.12	455.62,	462.63	326.07	323.68,	328.46	0.71	133.05
80% 20s		Avg	574.93	571.94,	577.92	420.90	418.42,	423.39	0.73	154.02
		Max	687.42	685.42,	689.41	524.34	522.36,	526.31	0.76	163.08
	Router	Min	114.78	113.91,	115.66	81.52	80.92,	82.11	0.71	33.26
		Avg	143.73	142.99,	144.48	105.23	104.60,	105.85	0.73	38.51
		Max	171.85	171.35,	172.35	131.08	130.59,	131.58	0.76	40.77
120 s	AS	Min	475.92	474.34,	477.50	337.84	336.10,	339.57	0.71	138.08
		Avg	572.24	569.77,	574.72	418.03	415.44,	420.62	0.73	154.21
		Max	663.79	659.34,	668.25	500.96	496.41,	505.50	0.75	162.84
	Router	Min	118.98	118.58,	119.38	84.46	84.02,	84.89	0.71	34.52
		Avg	143.06	142.44,	143.68	104.51	103.86,	105.16	0.73	38.55
		Max	165.95	164.83,	167.06	125.24	124.10,	126.38	0.75	40.709

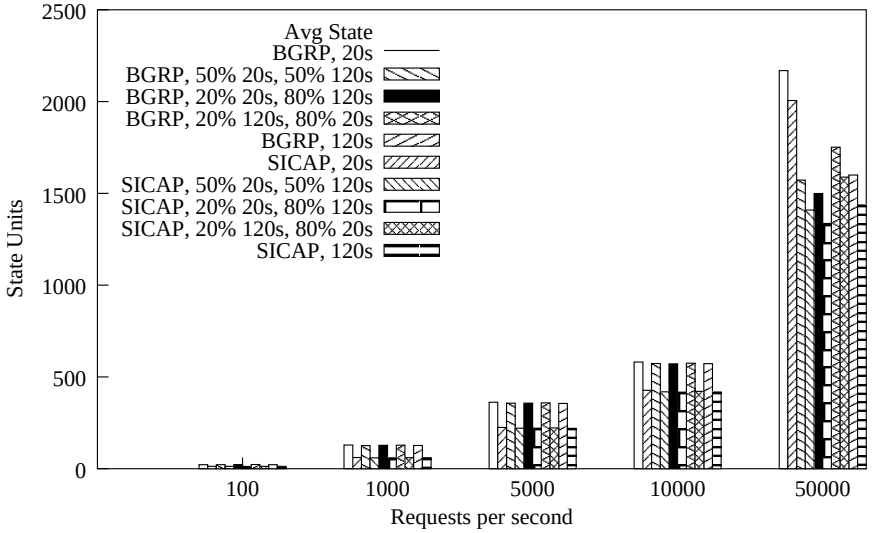


Fig. 7. State variation

that of BGRP in terms of their ability to reduce the amount of state that needs to be maintained along a path, and showed that even though both protocols achieve good performance, SICAP has consistently lower state requirements than BGRP. This is of importance not so much to offer a better performing alternative to BGRP, but to quantify the performance improvements that might still be available.

Because neither of the base version of SICAP or BGRP addresses the issue of reducing the signaling load when compared to a mechanism that does not perform aggregation, this is an area where both protocols are in need of further improvements. As a possible solution, we are currently studying several over-reservation mechanisms in both the context of SICAP and BGRP, with the purpose of evaluating their performance in terms of signaling load reduction, and bandwidth efficiency.

REFERENCES

[1] A. Broido, E. Nemeth, and K. Claffy. Internet Expansion, Refinement, and Churn. Technical report, CAIDA, 2002.

[2] E. Nikolouzou, P. Sampatakos, L. Dimopoulou, I. Venieris, M. Winter, B. Koch, T. Engel, S. Salsano, V. Genova, F. Ricciato, and G. Eichler. BGRPP: Performance evaluation of the proposed Quiet Grafting mechanisms. *Internet Draft*, July 2002. draft-nikolouzou-bgrpp-sim-00.txt.
[3] G. Huston. Analyzing the Internet's BGP Routing Table. Technical report, January 2001.
[4] IST Aquila Project. Public Deliverable D1203, Final System Specifications. Technical report, http://www-st.inf.tu-dresden.de/aquilhttp://www-st.inf.tu-dresden.de/aquila/files/public-deliverables.htm, April 2002.
[5] P. Pan, E. Hahne, and H. Schulzrinne. The Border Gateway Reservation Protocol (BGRP) for Tree-Based Aggregation of Inter-Domain Reservations. *Journal of Communications and Networks*, June 2000.
[6] P. Pan and H. Schulzrinne. Yessir: A simple reservation mechanism for the internet. *8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 98*, July 1998.
[7] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. *Request for Comments 1663, Internet Engineering Task Force*, June 1994.
[8] R. Braden, L. Zhang, and S. Jamin. Resource Reservation Protocol (RSVP) - version 1, Functional Specification. *Request for Comments 2205, Internet Engineering Task Force*, September 1997.
[9] R. Sofia, R. Guérin, and P. Veiga. An Investigation of Inter-Domain Control Aggregation Procedures. In *ICNP'02*, November 2002.
[10] R. Sofia, R. Guérin, and P. Veiga. An Investigation of Inter-Domain Control Aggregation Procedures. Technical report, ESE, University of Pennsylvania, July 2002. available at http://einstein.seas.upenn.edu/mnlab/publications.html.
[11] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. *Request for Comments 2475, Internet Engineering Task Force*, December 1998.
[12] S. Uhling and O. Bonaventure. Implications of Interdomain Traffic Characteristics on Traffic Enginering. Technical report, University of Namur, June 2001.
[13] Telstra. BGP Table Report. http://bgp.potaroo.net/, February 2001.
[14] Y. Rekhter and T. Li. An Architecture for IP Address Allocation with CIDR. *Request for Comments*, September 1993.
[15] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). *Request for Comments 1771, Internet Engineering Task Force*, March 1995.