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

Rute Sofia
ESE, University of Pennsylvania, EUA
DI, University of Lisbon, Portugal
Email:rsofia@seas.upenn.edu

Roch Guérin
ESE, University of Pennsylvania
Email:guerin@ee.upenn.edu

Pedro Veiga
DI, University of Lisbon
Email:pmv@di.fc.ul.pt

Abstract— **Existing Quality of Service models are well defined in the data path, but lack an end-to-end control path mechanism that guarantees the required resources to bandwidth intensive services, such as video streaming. Current reservation protocols provide scalable resource reservation inside routing domains. However, it is primarily between such 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 [1]. 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* [2], and the *Differentiated Services* [3] 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)* [4] or the *Yet anothEr Sender Session Internet Reservation protocol* (YESSIR) [5], 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 at the *flow level*, i.e., per source and destination IP addresses. But, according to Huston [6], there were around 1.09 billion addresses visible in the 2001 Internet routing table, which translates into up to 10^{12} possible combinations of active IP addresses, and consequently, such aggregation scheme may not scale. Alternatively, aggregation could be based on groups of aggregated IP addresses [7], 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 the Internet routing infrastructure. From a scalability standpoint, since there are currently 13,000 active AS's in the Internet [8], this represents a much smaller universe than the billions of active IP addresses.

Our goal is two-fold. We first aim at describing 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 paper is organised 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.

II. Related work

Pan et al. [9] present an inter-domain signaling protocol, BGRP, which merges requests that have the same destination *AS*, thus 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. [1] present a comparison of the shared-segment and the sink-tree approaches. By means of simulations, they compare algorithms that illustrate the behavior of the two proposals, showing that the shared-segment approach has a total state cost higher than the one of 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.

0-7803-7710-9/03/$17.00 © 2003 IEEE.

The work presented in this paper builds on the previous one, since it describes a protocol, SICAP, that implements a number of enhancements to the basic shared-segment algorithms [1], eliminating 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 briefly 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 it is the first BR on the path to trigger the establishment of reservations. 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 back a GRAFT message that allocates the necessary resources along the path traversed by the earlier PROBE message. The aggregates created by BGRP have *soft-state*, i.e., their state 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.

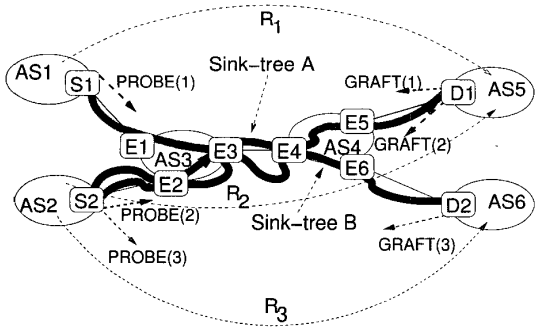


Fig. 1. BGRP example.

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, again originated 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 generic operation of BGRP, and a specific instance where it does not result in the minimum possible amount of state: 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 [1] 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 design 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 in the next sections. Issues such as the intra-domain mechanism to use and its interaction with SICAP, are beyond the scope of this work, since our focus is inter-domain control aggregation[1].

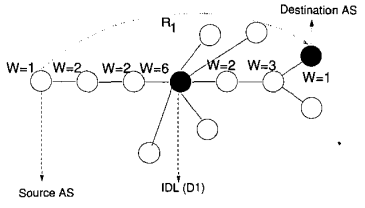
A. Deaggregator Choice Algorithm

SICAP uses an enhanced version of the *Weighted Deaggregation pointS* (WDS) [1] algorithm to decide on 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 *hotspots*, i.e., they might experience higher intensities of requests. For each AS m on a path, WDS computes a weight W_m equal to the number n_m of downstream neighbors of m, i.e., $W_m = n_m$. The AS that yields the largest weight is

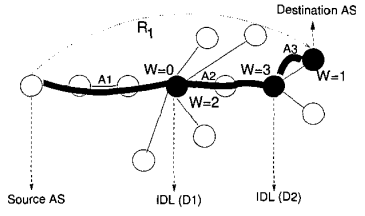
[1]The interaction between intra and inter-domain signaling mechanisms, are being address in the context of the IETF working group *Next Steps In Signaling (NSIS)* [10].

selected as an *intermediate deaggregation location (IDL)*. It should be noticed that the WDS algorithm is not presented here as the optimal (or unique) solution to decide on how to aggregate. Instead, WDS is presented as a possible simple algorithm, that does not require too much information, and that yet allows the shared-segment approach to perform better than the sink-tree approach. Deciding on where to deaggregate is a complex decision that depends mostly on the relationship between neighboring AS's, and it may rely on innumerable parameters: number of downstream AS neighbors, type of AS relationships (peer-to-peer, siblings, provider-client and so on), or even the way that traffic flows. To check if WDS is indeed the best option, there is the need for some thorough research on the subject. Such issue is beyond the scope of this paper.

Fig. 2, where each node represents an AS, and where each line represents an inter-domain link (traffic flows in both ways), 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 IDL's is carried through SICAP messages, which we present next, together with several examples of messaging sequences.

B. Messages

SICAP defines five message types, 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 some other information. **REQ** 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 BR's encountered on the path, which is of variable size, since it depends on the number of routers encountered. According to current Internet statistics [11], the current average path size is of five AS's and the maximum is of eleven AS's. Therefore, the average size of the route record should be about seven and its maximum size twenty, since the first and last AS only contribute with one BR. **RESV** messages are sent upstream by the last-deaggregator of a path to allocate the resources required by a REQ. 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** messages are triggered by the source of a reservation to delete it along a path. **REFRESH** messages update the information regarding reservations along a path. They are sent periodically each T_r [2] seconds by any first-aggregator. Their purpose is to detect inconsistencies, such as loss of messages, node failure, or path changes.

C. SICAP Operation

To illustrate how SICAP works, we use Fig. 3, where ellipses represent different AS's, Si 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 on the establishment of reservation R_1.

1) End-to-End Reservation Establishment: Fig. 4 illustrates the message exchange required to establish R_1. The establishment is triggered with $REQ(1)$, sent from $S1$ to the

[2]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.

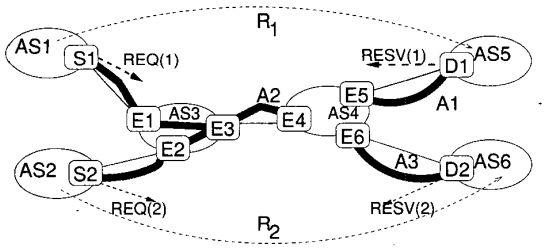
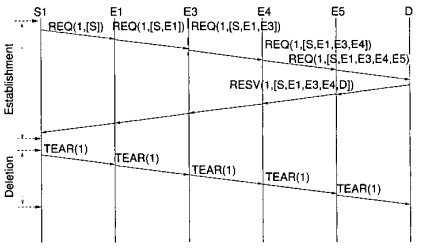
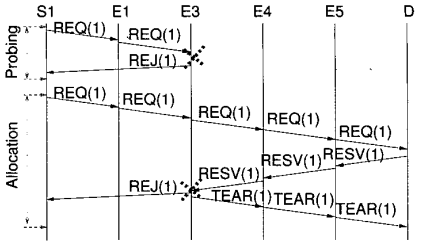


Fig. 3. SICAP example.



(a) reservation establishment and deletion



(b) reservation failure

Fig. 4. SICAP messaging.

destination end-host in AS 5. $REQ(1)$ contains the reservation identifier R_1 and its bandwidth requirement, $b_{\{S1,E1\}}$. $REQ(1)$ travels through $E1$, $E3$, $E4$, and $E5$, which add their identifiers to the route record of the message. When $D1$ receives $REQ(1)$, it realises 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 yet, $D1$ 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 $b_{\{S1,E1\}}$ 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 realises that R_2 can be merged into 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 branch of A_2 is created from $E3$ to $S2$. 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 an individual reservation is done from the first-aggregator to the last-deaggregator as a consequence of an end-host request, and it triggers an update to the aggregate(s) that carry the reservation until its destination.

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 in $b_{\{S,E1\}}$ 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 realises 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 sends back $REJ(1)$ 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 not enough resources on link $\{E1, E3\}$, to satisfy R_1. As a consequence, $E3$ 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 sending a $TEAR(1)$ message towards $D1$.

D. 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[3] 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 [1]. 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. [8] 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[4], which is a reasonable number, when compared to the much larger number of reservations crossing BR's.

Another issue related to the use of intermediate deaggregation locations is the processing cost of the lookup that has to be performed in order to find the aggregate that carries an already established reservation, at each intermediate aggregator. It is our believe that such cost is of minor significance for the global performance of a solution, since in average there are at most three intermediate aggregators, considering that an AS path has a maximum size of eleven AS's [12]. We consider that this hypothesis has to be further analysed, in real-scale environments. Hence, we leave it for now as future work.

V. BGRP AND SICAP PERFORMANCE COMPARISON

There are several measures of efficiency that can be used to evaluate the ability of an inter-domain signaling protocol in reducing storage 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 paper, we assume the context of the regular mode of operation for both BGRP and SICAP, for which the bandwidth of an aggregate is updated per individual request, i.e., an aggregate's bandwidth is equal to the sum of the bandwidth of its reservations. As a consequence of the update per individual reservation, both protocols achieve 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.

To analise state, we used the network simulator ns2 [13] and re-run a simulation scenario first introduced in [1]. Such scenario helped to detect previously the shared-segment weaknesses. Its re-enactment will help to determine if SICAP is able to reduce state by not keeping information about individual reservations at IDL's. The scenario uses the 50 node AS-level topology illustrated in Fig. 5, and a distribution of requests where each node has the same probability of being a source, and where destinations are placed according to a distribution of addresses based on AS distance [11]. The arrival of requests is modeled as a Poisson process with mean holding time of σ. The results[5] presented in Tab. I comprise the minimum, maximum and average state values, calculated within a 95% confidence interval. 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, 50% of short-lived requests and 50% of long-lived requests. The results 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 state ratio $\frac{SICAP}{BGRP}$ holds approximately the same value when the duration of requests changes, showing that the duration affects both protocols in a similar manner. 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. Fig. 6 shows the difference in average state for different intensities. Each bar represents the average state value that a protocol requires for a particular type of requests, and for a particular intensity.

[3]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.

[4]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%.

[5]Detailed results can be found in [14].

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.

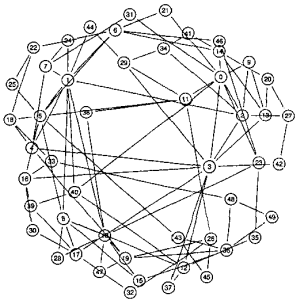


Fig. 5. Topology

TABLE I

STATE, INTENSITY OF 5,000 REQUESTS

σ	SCOPE	VALUE	BGRP (Avg/ 95% CI)			SICAP (Avg/95% CI)			$\frac{SICAP}{BGRP}$
20s	AS	Min	250.70	248.35,	253.04	148.00	146.67,	149.32	0.59
		Avg	361.92	360.22,	363.62	225.21	223.95,	226.48	0.62
		Max	473.71	472.15,	475.28	314.30	312.95,	315.65	0.66
	Router	Min	62.67	62.09,	63.26	37.00	36.67,	37.33	0.59
		Avg	90.48	90.06,	90.90	56.30	55.99,	56.62	0.62
		Max	118.43	118.04,	118.82	78.58	78.24,	78.91	0.66
50% 20s	AS	Min	272.40	269.01,	275.79	161.43	158.44,	164.41	0.59
50% 120s		Avg	356.92	355.23,	358.61	220.45	218.77,	222.13	0.62
		Max	441.68	439.52,	443.83	285.29	283.20,	287.38	0.65
	Router	Min	68.10	67.25,	68.95	40.36	39.61,	41.10	0.59
		Avg	89.23	88.81,	89.65	55.11	54.69,	55.53	0.62
		Max	110.42	109.88,	110.96	71.32	70.80,	71.84	0.65
120 s	AS	Min	277.32	275.50,	279.15	165.18	163.80,	166.56	0.6
		Avg	355.53	353.34,	357.71	219.79	218.14,	221.45	0.62
		Max	431.64	429.09,	434.19	277.32	274.83,	279.82	0.64
	Router	Min	69.33	68.88,	69.79	41.30	40.95,	41.64	0.6
		Avg	88.88	88.33,	89.43	54.95	54.53,	55.36	0.62
		Max	107.91	107.27,	108.55	69.33	68.71,	69.95	0.64

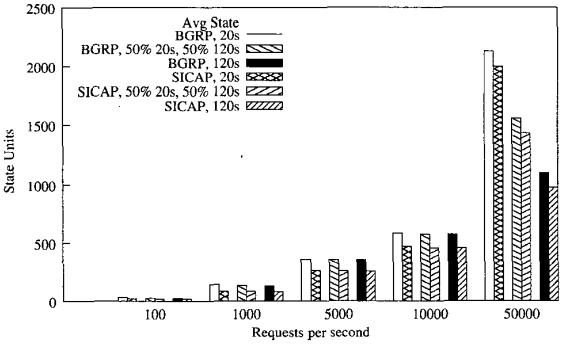


Fig. 6. State variation.

VI. SUMMARY AND CONCLUSIONS

In this paper, 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 their performance in terms of the only performance parameter where their regular operation mode differs, i.e., their ability to reduce the amount of state that needs to be maintained. We showed, by means of simulations, 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. However, neither SICAP nor BGRP addresses the scalability issue brought up by the required signaling load, when compared to a mechanism that does not perform aggregation. One possible solution to this problem might be the use of *over-reservation*, i.e., to provide each aggregate with more bandwidth than the required at a particular instant, to reduce the signaling load. We are currently evaluating several over-reservation mechanisms, both in the context of SICAP and of BGRP, in terms of the signaling load reduction, and the blocking probabilities achieved.

REFERENCES

[1] R. Sofia, R. Guérin, and P. Veiga, "An Investigation of Inter-Domain Control Aggregation Procedures," in *ICNP'02*, Nov. 2002.

[2] 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.

[3] 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*, Dec. 1998.

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

[5] 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.

[6] G. Huston, "Analyzing the Internet's BGP Routing Table," Tech. Rep., Jan. 2001.

[7] Y. Rekhter and T. Li, "An Architecture for IP Address Allocation with CIDR," *Request for Comments*, Sept. 1993.

[8] A. Broido, E. Nemeth, and K. Claffy, "Internet Expansion, Refinement, and Churn," CAIDA, Tech. Rep., 2002.

[9] 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.

[10] I. W. G. NSIS, "Next Steps in Signaling Working Group Charter." [Online]. Available: http:www.ietf.org/html.charters/nsis-charter.html

[11] S. Uhling and O. Bonaventure, "Implications of Interdomain Traffic Characteristics on Traffic Enginering," University of Namur, Tech. Rep., June 2001.

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

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

[14] R. Sofia, R. Guérin, and P. Veiga, "SICAP, A Shared-segment based Inter-domain Control Aggregation Protocol," University of Pennsylvania, Tech. Rep., Oct. 2002, available at http://einstein.seas.upenn.edu/mnlab/publications.html.