Signaling Protocol for Session-Aware Popularity-based Resource Allocation

Paulo Mendes [*] Henning Schulzrinne [†] Edmundo Monteiro [‡]

Abstract

The Differentiated Services model (DS) maps traffic into services that offer different quality levels. However, flows are treated unfairly in each service, since the DS model lacks a policy to distribute service bandwidth between flows that form the service aggregate traffic. We present a signaling protocol called Session-Aware Popularity-based Resource Allocation (SAPRA) that fairly distributes resources along the path of each session, with an amount of resources proportional to their receiver population. We assume that scalable sessions are layered hierarchically, with each layer sent to a different multicast group. We evaluate the efficiency of SAPRA using theoretical analysis and simulation. [1]

Keywords: Quality of service; multicast; differentiated services; scalable video sessions; fairness; signaling [2].

[*]Department of Computer Science, Columbia University, New York NY 10027-7003. +1 212 939-7000, mendes@cs.columbia.edu and Department of Informatics Engineering, University of Coimbra, pmendes@dei.uc.pt

[†]Department of Computer Science, Columbia University

[‡]Department of Informatics Engineering, University of Coimbra

[1]This work is supported by POSI-Programa Operacional Sociedade de Informação of Portuguese Fundação para a Ciência e Tecnologia and European Union FEDER

[2]Multi-point, multicast service management; QoS management; Multimedia network traffic engineering and optimization

1 Introduction

Current popular Internet multimedia applications, such as RealNetworks SureStream[3] and Microsoft Windows Media [4] stream the same content at different rates, depending on receiver capabilities. However, even if they use multicast, each stream in this multi-rate encoding contains a complete encoding, thus wasting bandwidth. This problem can be solved by using scalable encoding [20, 10]. Scalable encoding divides a video stream into cumulative layers with different rates and importance. The stream rate is then the sum of its layers. Sources send only one stream to all receivers, mapping each layer to a different multicast group. All layers belonging to the same stream form a *session*, as in the Real-Time protocol (RTP) [19].

Due to their real-time characteristics, multimedia sessions need quality guarantees from networks, which can be provided by the DS model [3]. This model allows network providers to aggregate traffic into different services at the boundaries of their network according to per-hop behaviors (PHB) that characterize the allocation of resources needed to give an observable forwarding behavior (loss, delay, jitter) to the service aggregated traffic. However, inside each service, sessions are treated unfairly, since the DS model has no policy to distribute bandwidth between sessions that compose the service aggregated traffic, i.e., services lack *inter-session* fairness.

We propose a fairness protocol called *Session-Aware Popularity-based Resource Allocation* (SAPRA), which provides inter-session fairness inside each service by assigning more bandwidth to sessions with larger audiences. In the presence of scalable sessions, SAPRA also provides *intra-session* fairness, assigning to each layer a drop precedence that matches its importance. But, this is only possible if the service has the capability to assign different drop precedences to flows, as happen with services based upon the Assured Forwarding PHB [6]. To achieve its goal, SAPRA adds agents and markers to edge routers. Agents manage sessions to provide inter-session fairness; markers deal with layers, providing intra-session fairness. In the spirit of the DS model, agents and markers are placed only in edge routers, since they do not need access to the aggregate traffic of each service.

[3]http://www.realnetworks.com/resources/producer
[4]http://www.microsoft.com/windows/windowsmedia/

SAPRA also includes a punishment function and a resource utilization maximization function. The former punishes high-rate sessions, that is, sessions with a rate above their fair rate during periods of congestion, motivating sessions to adapt to the network capacity. The latter avoids waste of resources when there are sessions with rate lower than their fair share of bandwidth, in which case the bandwidth not used is equally distributed between the other sessions. The behavior of agents and markers is described and evaluated in [14]. In this paper, we describe and study the SAPRA signaling protocol used by agents to exchange information about sessions, allowing a fair distribution of resources in the path of each session. The theoretical analysis and simulations show that the protocol has small bandwidth overhead, is efficient in providing agents with accurate information about sessions, and supplies receivers with timely reports about the quality of their sessions.

The remainder of the paper is organized as follows. Section 2 briefly describes work related to SAPRA. In Section 3, we describe and evaluate the SAPRA protocol. Section 4 presents simulation results. Finally, Section 5 presents conclusions and future work.

2 Related Work

The amount of resources reserved for each DS service depend on service-level agreements (SLAs) between domains. The management of multicast traffic can be done with static SLAs for flows with more than one egress router. However, since multicast traffic is both heterogeneous and dynamic, static SLAs needs over-provisioning of resources in the domain. This problem can be solved by dynamic SLAs, which can be managed by bandwidth brokers [15] or approaches that use intra-domain signaling protocols, e.g., the Resource Reservation Protocol (RSVP) [4], to reserve resources where they are needed, avoiding over-provisioning along certain branches of a multicast tree. However these approaches don't distribute reserved service resources across several domains, between sessions that constitute the service aggregate traffic. To solve this issue some proposals present inter-session fairness mechanisms based upon the max-min fairness definition [2], but this definition cannot exist with discrete set of rates [17]. Sarkar et al. present a fairness algorithm [18] that considers discrete set of rates, but does not consider the population of sessions, and its scalability and efficiency is not proved in an Internet-like scenario. The population of sessions is considered by Legout et al. [11], but their proposal doesn't consider intra-session fairness for scalable sources, requires changes in all routers, doesn't maximize resources and doesn't punish high rate flows. Li et al. present [12] another inter-session fairness mechanism, but it is also based upon the max-min fairness definition, assumes only one shared link, and doesn't consider the population of a session, as well as the importance of its layers.

3 SAPRA Protocol

In this section we describe and evaluate SAPRA. We assume that edge routers have an accurate notion about resources reserved for each service, based upon static or dynamic SLA, and that mechanisms implemented in each domain are a policy issue and beyond the scope of this paper.

In our model, scalable sources distribute media content in sessions. Each session consists of multiple layers. Each layer is identified by a source-specific multicast (SSM) channel, namely the sender IP address and the destination multicast group [7]. SAPRA does not depend on number of multimedia sources. Multicast branch points can be located in edge and interior routers, but receivers are attached only to edge routers.

SAPRA is most suited for long-lived multicast sessions with large receiver populations such as Internet TV and periodic near-video-on-demand systems [1].

SAPRA also integrates non-scalable sources and unicast traffic. Non-scalable sources are treated like scalable sources with one layer. Unicast flows are treated as sessions with one layer and one receiver.

Not all autonomous systems have to implement SAPRA, as SAPRA signaling messages are exchanged between SAPRA-speaking agents. However, receivers in non-SAPRA domains do not count toward the session population and thus sessions with large numbers of receivers outside of SAPRA domains are left with a smaller share of resources than they deserve.

Receivers can join sessions by, for example, listening to Session Announcement Protocol (SAP) [5] messages. Receivers first join the multicast group for the most important (lowest) layer and then increase their reception quality by joining additional (higher) layers. Each layer requires all layers below it. Agents in leaf edge routers use IGMPv3 "State-Changes" reports, issued when receivers join a layer, to measure the local receiver population for each session.

SAPRA agents and markers allocate resources to sessions, rather than managing multicast groups without concern about their semantic relationship. If the structure of sessions were hidden from agents and markers, agents would not be able to implement inter-session fairness, markers would fail to provide intra-session fairness, and sessions will have lower and less stable quality levels.

We propose two methods that allow agents in leaf routers to collect information about the composition of sessions, using consecutive address ranges and IGMP extensions. We assume SSM, where multicast addresses are locally allocated by each source, from a pool of 2^{24} addresses in IPv4 and 2^{32} addresses in IPv6. For the first method, each sender allocates consecutive multicast addresses to all layers inside a session and keeps a gap of at least one address between sessions. Since receivers join layers sequentially, an agent can detect a new session if its multicast address differs by at least two from any other session address. Alternatively, if the number of layers per session can be bounded, the address bits can be divided into high-order session and low-order layer bits.

For the second method, we add the address of the most important layer to the IGMPv3 [8], using the "auxiliary" data field. The address of the most important layer then identifies the session.

For both methods, agents deduce the relationship between layers in a session by the order receivers join them. Details can be found in [14].

We define downstream as the direction from the data source to the receivers, and upstream as the opposite. Session receivers are attached to leaf routers in the multicast tree. Ingress routers deliver data into a domain, while egress routers transport data from a domain elsewhere. In our model, all leaf routers are egress routers. We also define the *Downstream Fair Rate* (DFR), the *Local Fair Rate* (LFR) and the *Used Fair Rate* (UFR) in a link. The DFR is the fair rate downstream the link, the LFR is the fair rate of the link and the UFR is the lesser of the LFR and the DFR. By using the UFR instead

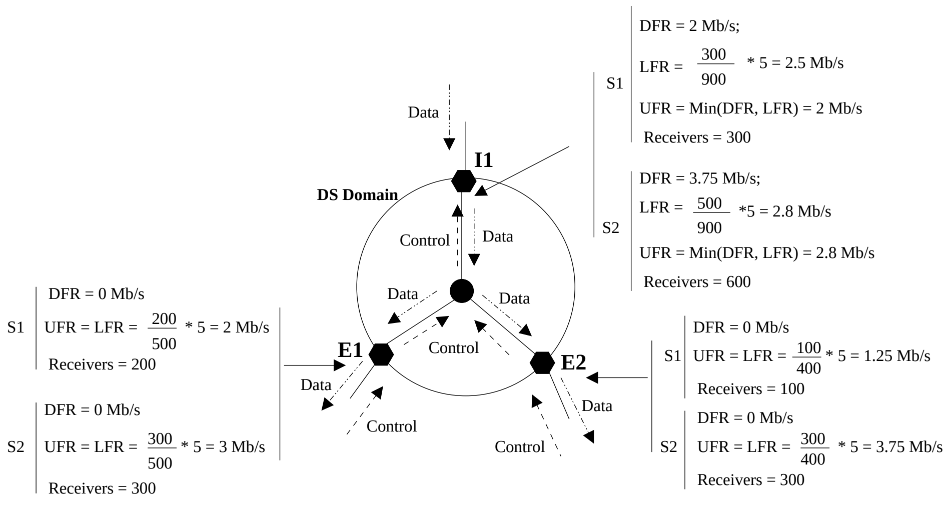

S1
DFR = 2 Mb/s;
LFR = 300/900 * 5 = 2.5 Mb/s
UFR = Min(DFR, LFR) = 2 Mb/s
Receivers = 300
S2
DFR = 3.75 Mb/s;
LFR = 500/900 *5 = 2.8 Mb/s
UFR = Min(DFR, LFR) = 2.8 Mb/s
Receivers = 600
Data
I1
DS Domain
Control
Data
Data
Data
E1
Control
E2
Data
Data
Control
Control
S1
DFR = 0 Mb/s
UFR = LFR = 200/500 * 5 = 2 Mb/s
Receivers = 200
S2
DFR = 0 Mb/s
UFR = LFR = 300/500 * 5 = 3 Mb/s
Receivers = 300
S1
DFR = 0 Mb/s
UFR = LFR = 100/400 * 5 = 1.25 Mb/s
Receivers = 100
S2
DFR = 0 Mb/s
UFR = LFR = 300/400 * 5 = 3.75 Mb/s
Receivers = 300

Figure 1: Fair rate computation in edge routers

of the LFR, sessions that cannot make use of resources further downstream don't waste bandwidth.

Fig. 1 illustrates how agents compute the UFR of two sessions, S_1 and S_2, in a domain with three edge routers. This computation requires information about sessions population and DFR, which agents collect from downstream neighbors using the SAPRA signaling protocol. As we described in [14], agents compute the LFR by assigning more resources to sessions with larger receiver population. For example, assuming that all links have 5 Mb/s, the LFR of S_1 is 1.25 Mb/s in E_2, because S_1 has 100 receivers and the total population in the link is 400.

In this example, the UFR of each session is equal to its LFR at egress routers, since its DFR is zero. At the ingress router, the population of a session is the sum of its receivers in each egress router and its DFR is the maximum value of its UFR in each egress router. This maximum value satisfies the heterogeneous requirements of receivers downstream different egress routers without congesting links downstream the domain, since packets will be dropped at egress routers in conformity with the UFR that session has at those routers. As mentioned before, we assume that each domain has policies to assign resources to each service in order to avoid congestion in interior routers.

At the ingress router, S_1, for example, has a population of 300 receivers, a DFR of 2 Mb/s and an UFR of 2 Mb/s, which is the minimum value between LFR (2.5 Mb/s) and DFR. The estimated UFR of each session is passed to the markers in the downstream link, being used to mark packets in or out of profile, depending if the session estimated rate is lower or higher than its UFR.

3.1 Overview of Protocol Operation

Agents use two messages, UPDATE and SYNC, to propagate information. UPDATE messages deliver the UFR and population size of each session to upstream neighbors, allowing agents along the path to compute fair rates. SYNC messages propagate information about session quality to downstream neighbors. When session listeners get a SYNC message, they can adapt to quality changes.

Messages are exchanged only between neighboring agents, increasing scalability. For robustness, SAPRA runs over TCP, eliminating the need to implement fragmentation and re-transmission. UPDATEs are forwarded using Border Gateway Protocol's (BGP) [16] routing information and SYNCs follow the path of the UPDATEs in reverse, similar to RSVP PATH and RESV messages.

SAPRA agents send messages periodically, not synchronized to receiving messages from elsewhere, gathering up information during the interval. Periodic messages are suppressed if there has been no significant change in the information to be propagated. Sending intervals are short to reduce the time it takes for resource allocations to converge. Sending intervals are not synchronized; each nominal interval is varied by a small random about of 10%.

As an alternative, agents could send messages only after receiving a message of the same type, a SYNC from the upstream neighbor in the path to the session source or an UPDATE from each downstream neighbor where that session is present. However, this approach increases UPDATEs propagation delay since agents can take a long time to gather information about each session from all downstream neighbors. This happens, for example, when there is a different number of routers between the agent and each neighbor, or when neighbors send information about the same session at different times.

3.1.1 UPDATE messages

UPDATE messages are sent with a minimum nominal interval of 1 s. They are triggered when an agent receives the first join from local receivers or when it receives the first UPDATE from a downstream neighbor. Each message includes information about non-stationary sessions, i.e., sessions whose population or UFR has changed significantly since the last time they were included in an UPDATE, considering the total population and total UFR in the upstream link where the message is going to be sent. So, the agent doesn't send any UPDATE if there are only stationary sessions. But, when a session ends, its information (no receiver, UFR of zero) is always included in the next UPDATE allowing the immediate release of unnecessary state in upstream agents. Minimum state variation is set to 25%, which appears to be a good compromise between the number of UPDATEs and the propagation of accurate information to allow a fair allocation of resources. This restriction allows the use of 1 s intervals, improving the protocol accuracy: since sessions are long and their population normally varies significantly only when they begin and end, intervals as short as 1 s are enough to propagate significant changes in the state of sessions.

Sessions have hard-state, since state starts and ends with the reception of UPDATEs. However it is also refreshed by periodic complete UPDATEs that include information about all sessions. This soft-state property makes SAPRA cope with routing changes and link failures, so the default values for the interval of complete UPDATEs and for the time agents wait before deleting state were chosen based upon BGP variables: the former is 30 s, the value of the BGP `keep_alive_time` variable, and the latter 90 s, the value of the BGP `hold_time` variable. When agents receive complete UPDATEs, they only update the state of those sessions whose DFR varies more than a set minimum to avoid computations that do not affect the local resource distribution. However, the population estimate is always updated, to avoid cumulative rounding errors.

3.1.2 SYNC messages

SYNC messages are sent with a minimum nominal interval of 1 s. They are triggered when an agent with local sources receives the first UPDATE from a downstream neighbor, or when an agent without local sources receives the first SYNC from an upstream neighbor. A SYNC carries information about sessions present in the downstream link where the message is being sent. If the source of a session is local, that information is the UFR of that session in the downstream link, otherwise it is the minimum value between the local and upstream UFR. In both cases, the fair rate is only included if it is different

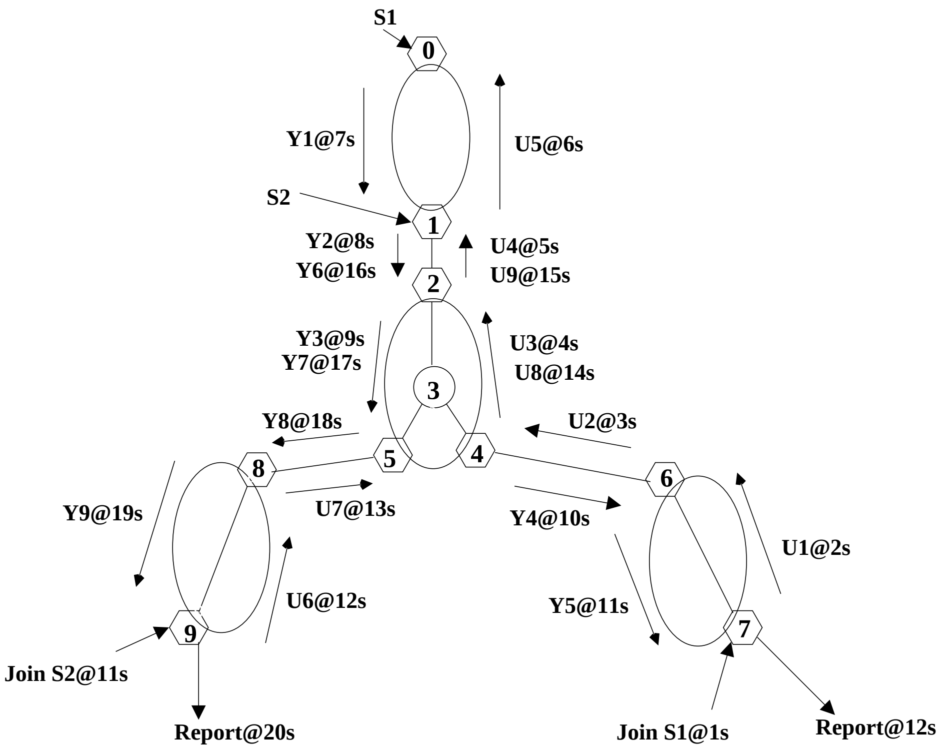


Figure 2: Protocol operation

from the one sent in the previous SYNC, which allows the use of sending intervals of 1 s.

3.2 Example of Protocol Operation

Fig. 2 illustrates how SAPRA works using two sessions, S_1 and S_2. We assume that intra-domain links have 10 Mb/s of bandwidth and inter-domain links between routers 1 and 2, routers 5 and 8, and routers 4 and 6 have 10 Mb/s, 3 Mb/s, and 4 Mb/s of bandwidth, respectively. We label edge router i as r_i, the agent on that router as a_i, and the link between r_i and r_j as (r_i, r_j). In Fig. 2, the notation $U_n@t$ indicates that the nth UDPATE is being sent at time t; $Y_n@t$ has a similar meaning for SYNCs. For example, $U_8@14s$ means that the eighth UPDATE was sent at second 14.

When receivers join S_1, a_7 sends U_1 toward a_0. When a_1 receives U_4, it is triggered to start sending SYNCs, since it has a local source; but since the source isn't from S_1, a_1 doesn't send a SYNC at second 6. The first SYNC is sent by a_0 toward a_7 at second 7, with 4 Mb/s as minimum UFR of S_1 on the tree, the UFR on (r_4, r_6). Receivers on r_7 get a reports 11 s after joined S_1.

When receivers join S_2, a_9 sends U_6 toward a_1, whose router is directly connected to the source of S_2. No UPDATE is sent from a_1 to a_0 since S_2 is absent from (r_0, r_1), and the state of S_1 remains the same. At second 8, a_1 sends Y_2 with 3 Mb/s as minimum UFR of S_2 on the tree — UFR on (r_5, r_8). That information is received by a_9, on Y_9, 8 s after receives joined S_2. No SYNC is sent from a_2 toward a_7, through a_4, since S_2 is absent in that branch and S_1 upstream state remains the same.

Considering an hypothetical scenario, if we had used a bandwidth of 5 Mb/s to (r_1, r_2), the LFR for the two sessions would be 2.5 Mb/s, which is lower than the DFR of S_1 and S_2. For this case, a_1 would send an UPDATE to a_0 at second 16 indicating a decrease of UFR of S_1 from 4 Mb/s to 2.5 Mb/s and would include in Y_6 the UFR of S_2 and S_1, both equal to 2.5 Mb/s. Therefore, a_2 would send also a SYNC to a_4, in addition to the one sent to a_5. Since the minimum value between the UFR (4 Mb/s) and the upstream fair rate (2.5 Mb/s) in a_4 is different from the value sent in the previous SYNC (4 Mb/s), a SYNC would be sent to a_6. This way, S_1 receivers on r_7 would get a second report at second 20, updating the UFR from 4 Mb/s to 2.5 Mb/s. After second 20, no other message is exchanged, because there are no changes in population or fair rates. However agents send a complete UPDATE at second 30, the default sending interval for those messages.

3.3 Protocol evaluation

To evaluate the protocol, we analyze the memory it requires to store state, its bandwidth overhead, and the time it takes to update UFR of all sessions in all agents as well as to notify receivers about their session quality, when changes happen in the state of sessions.

For N sessions, SAPRA agents need to store $O(N)$ amount of state. In particular, the stored state does not depend on the number of receivers, increasing SAPRA scalability.

The protocol bandwidth overhead is significantly reduced, because messages are not sent in every sending interval and they are only exchanged between neighbors. This means that the overhead is independent from the network size, being dependent only on the number of sessions and sending intervals. Therefore, we analyze the protocol scalability for a worst-case scenario where the sending interval of both messages is short (1 s) and the number of sessions is high (1,000). We assume that all sessions are present on all links, that each session has three layers with total rate of 1.8 Mb/s, and that their population is very dynamic, with significant changes occurring every second. Since the information about each session occupies 48 bytes in UPDATEs and 12 bytes in SYNCs, the control information between any two agents consumes 480 kb/s. Due to this and the data rate of all sessions in each link (1,800 Mb/s), the ratio between control and data bandwidth is of 0.03%. Even in the presence of low-rate sessions the overhead is still insignificant: for example, if we assume sessions with 64 kb/s, the protocol overhead is 0.75%.

The time taken by SAPRA to update UFR of all sessions in all agents depends on the number of agents in each session path and if agents were already triggered to start sending messages. Time required to notify receivers also depends on one additional factor: if the minimum UFR of the session, in the path to its source, increases or decreases.

The maximum time to update UFR of a session is given by $(n_a - 1)\, a_s$, and the maximum time to deliver a report to its receivers by $(2\,(n_a - 1) + 1)\, a_s$, where n_a is the number of agents in the session path and a_s the sending interval. This maximum time occurs only during the first update, because the first UPDATE triggers all agents to start sending messages and so it is delayed by the total a_s in each agent. After this, the delay is lower than a_s. So, for a path with six domains and 12 agents, the maximum time required to update UFR of sessions is 11 s and receivers get their first report 23 s after joining their session. We assume paths with lengths up to six domains since the percentage of longer paths is very low. For example, a November 2001 study of 60,978 different Autonomous Systems (AS) [21] showed that only 0.5% of paths cross seven domains and 0.1% eight domains.

However, receivers are notified faster if the minimum UFR of their session is decreased by an agent that is not the agent closest to the source of their session, because that agent immediately sends a SYNC and so n_a is lower than the number of agents in the session path. This is possible because in each stabilized network branch the upstream fair rate in each link is equal to the minimum UFR on the branch. If changes happen in different branches at the same time, stabilization of fair rates is achieved first in each branch and only afterwards in the entire network.

When the UFR decreases below the upstream fair rate, the agent includes it in the next SYNC.

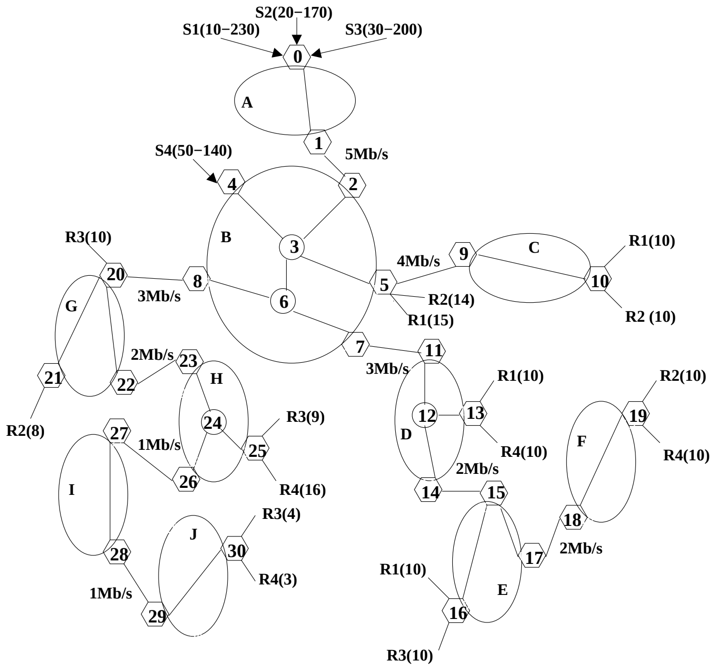


Figure 3: Simulation scenario

Therefore, if the UFR remains higher than the upstream fair rate in all agents, the new minimum UFR is only establish by the agent closest to the source, and so the notification of receivers is slower if the UFR of sessions increase. On the one hand, slower notification of higher minimum UFR leads to a higher stability, since receivers only join layers when resources of their session are updated in the entire path. On the other hand, faster notification of lower minimum UFR optimize resource utilization, since receivers leave layers more quickly, reducing network traffic.

4 Simulations

In this section we present simulations that show the protocol ability to allocate resources between sessions in multicast trees across several domains. We use a scenario with ten domains, one domain per AS as shown in Fig. 3, where the longest path has six domains. We used the ns simulation tool.

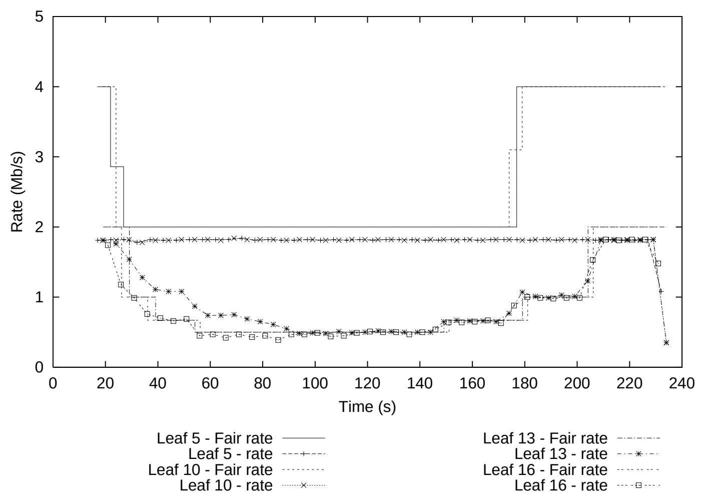
The topology has four sessions, S_1 to S_4, in different domains, multicast branches across different number of domains, domains with multicast branches in ingress and interior routers, receivers in ingress and egress routers, and egress routers only with local receivers and also with downstream links.

Due to the lack of information about traffic distribution among different ASs, we use the distribution of addresses by AS distance [21] to decide upon receiver locations. Hence, we place 26%, 40%, 26%, 9%, and 2.5% of receivers in domains at a distance of two, three, four, five, and six domains from their source, respectively. In Fig. 3, $R_x(Y)$ denotes Y receivers of session S_x.

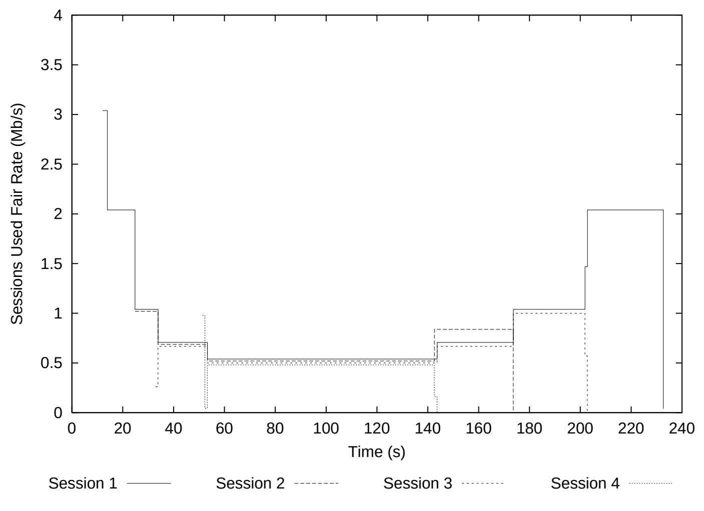
We assume that service capacity in inter-domain links is equal to the link bandwidth and that intra-domain links have enough bandwidth to avoid congestion. We also assume that each queue has a size of 64 packets, which is the default value in Cisco IOS 12.2, and that interior routers have FIFO queues, since these are the simplest ones. We use data packets with 1,000 bytes since this is a value between the MTU of dial-up connections, 576 bytes, and the MTU of ethernet and high speed connections, 1,500 bytes. Since control messages are mainly exchanged between routers, UPDATE and SYNC packets have a size of 1,500 bytes, which is the default MTU in Cisco routers.

Session S_1 spans seconds 10 through 230, S_2 20 to 170, S_3 30 to 200, and S_4 50 to 150. Each session has three layers with total rate of 1.8 Mb/s. The most important layer consumes 303 kb/s, the medium importance one 606 kb/s, and the least important one 909 kb/s. Although SAPRA can deal with any number of layers, three layers provide a good quality/bandwidth trade-off and additional layers only provide marginal improvements [9]. The simulation covers 240 s, which is enough to identify transient state when sessions start and end. To simplify the presentation, we assume that all receivers for a session join and leave simultaneously. For longer simulations, sessions remain stable for a longer time and the time needed for their state to stabilize does not change. Results for 600 s simulations are available in [13]. We use the session rate monitored by receivers and the minimum UFR they receive in reports from agents in leaf routers. dumped every 5 s by S_1 receivers on r_5, r_{10}, r_{13}, and r_{16} (Fig. 4 a)) and the UFR of sessions on (r_7, r_{11}) (Fig. 4 b)) to analyze SAPRA efficiency. Results for S_2, S_3, and S_4 are available in [13]. Simulation results show that SAPRA is efficient keeping the rate of sessions below their UFR, and updating the UFR of all sessions in all agents and notifying receivers.

Fig. 4 a) shows that the rate of S_1 is kept below its minimum UFR, with the exception of the rate monitored by r_{13} receivers from seconds 29 to 89. This happens because the UFR of all sessions on



(a)



(b)

Figure 4: a) S_1 rate and UFR; b) UFR of all sessions on (r_7, r_{11})

(r_7, r_{11}) is always lower than the link capacity, as is shown in Fig. 4 b), and so a percentage of out packets from all sessions pass through r_7 allowing r_{13} receivers to get a rate higher than the minimum UFR. Nevertheless, Fig. 4 a) also shows that S_1 rate on r_{13} decreases in that interval. This is due to the punishment mechanism that starts filtering layers of S_1 when this session is identified as a *high-rate session* during the congestion of (r_7, r_{11}). *High-rate sessions* are sessions with a rate higher than their UFR plus their share of the available bandwidth. Bandwidth is available, because other sessions have a UFR lower than their LFR, to match their DFR, releasing local resources. For each link, the available bandwidth is divided in equal parts between all sessions with rate above their UFR.

Fig. 4 b) also shows that SAPRA is able to distribute resources in each link, decreasing UFR of sessions to release resources for new sessions or increasing them to use resources made available by sessions that ended. The only irregular behavior is the initial variation of the UFR of S_4, which shows up since a_7 gets information about receivers of S_4 on r_{13} and r_{19} at different moments: at second 51, a_7 has only knowledge about receivers on r_{13}, because the UPDATE from a_{13} arrives to a_7 first than the one from a_{19}. At that time the LFR of S_2, S_3 and S_4 is 0.6 Mb/s and the one of S_1 is 1.2 Mb/s, since S_1 has 20 receivers, and S_2 and S_3 have 10 receivers each; however S_1 has DFR of 0.6 Mb/s, making 0.6 Mb/s available to increase the UFR of S_2 and S_3 to 0.667 Mb/s, their DFR, and S_4 to 1.06 Mb/s. At second 52, a_7 gets information about only one S_4 receiver on r_{19}, because only one receiver joined S_4 when a_{19} sent an UPDATE on second 50; therefore, the UFR of S_4 is reduced to 0.06 Mb/s, since this is its UFR on link (r_{14}, r_{15}). After second 53, a_7 already knows about all S_4 receivers on r_{19} and so the UFR of S_4 increases to 0.5 Mb/s.

Next, we analyze the efficiency of SAPRA updating UFR of all sessions in all agents and notifying receivers when S_1 and S_2 receivers join and leave their sessions. When receivers join S_1, receivers closer to the source take relatively more time to get their first report than receivers closer to the leaves, in proportion to their distance to the source, since it is their first UPDATE that triggers upstream agents.

When receivers join and leave S_2, the time that SAPRA takes to notify S_1 receivers also depends on the variation of the minimum UFR of S_1, in addition to the session path length to their leaf router. Table 1 presents more clearly the times seen in Fig 4 a).

	router 5	router 10	router 13	router 16
S2 join at second 20	27 s	24 s	29 s	25 s
S2 leaveat second 170	177 s	179 s	179 s	181 s

Table 1: Moment when receivers are notified about S_1 minimum UFR in their path from the source

This table shows that reports notifying a decrease of the minimum UFR, when S_2 starts, arrive to receivers earlier than reports notifying an increase, when S_2 ends. Receivers on r_5 and r_{13} are exceptions, because they are located upstream the links where the minimum UFR of S_1 is decreased, (r_5, r_9) and (r_{14}, r_{15}) respectively, and so their notifications are always generated by a SYNC sent by the agent closest to the source. Since r_{10} is downstream (r_5, r_9) and r_{16} is downstream (r_{14}, r_{15}), receivers on those routers are notified about a decrease of the minimum UFR first than r_5 and r_{13} receivers, respectively. For example, after receivers joined S_2 on r_{19}, the UFR of S_1 decreases to 1 Mb/s in (r_{14}, r_{15}). Since this value is lower than the upstream fair rate (2 Mb/s), a_{14} includes 1 Mb/s as the minimum UFR of S_1 in the next SYNC.

Although r_5 receivers are notified about the minimum UFR of S_1 at second 27, Fig. 4 shows that they get a transient UFR of 2.8 Mb/s at second 21. This happens because changes occur at the same time in the branch from the source to r_5 and in the branch from r_5 to r_{10}, when receivers joined S_2 on r_5 and r_{10}, which means that the stabilization of UFR of S_1 is achieved first in each branch and only after in the entire S_1 path to r_{10}. This transient value represents the minimum UFR in the branch upstream r_5 before second 21, but it isn't the minimum UFR in the entire S_1 path to r_{10}, since the UFR of S_1 on the branch from r_5 to r_{10} wasn't considered yet upstream of r_5. Fig. 5 shows how the minimum UFR of S_1 is stabilized in the entire path to r_{10} after being computed in each branch.

When a_5 gets a SYNC at second 21 with an UFR of 2.8 Mb/s, it sends a notification to receivers on r_5, since this is the current minimum UFR of S_1 in the path from the source. At that time, S_1 has an UFR of 2 Mb/s on the branch from r_5 to r_{10}, which correspond to the UFR computed by a_5 on (r_5, r_9) after received an UPDATE from a_{10}. At second 22, a SYNC is sent toward a_{10} with an UFR of 2 Mb/s, since this value is lower than the upstream fair rate on a_5 (2.8 Mb/s) and it is different from the fair rate previously sent (4 Mb/s). At the same time, a_5 also sends an UPDATE with an UFR of

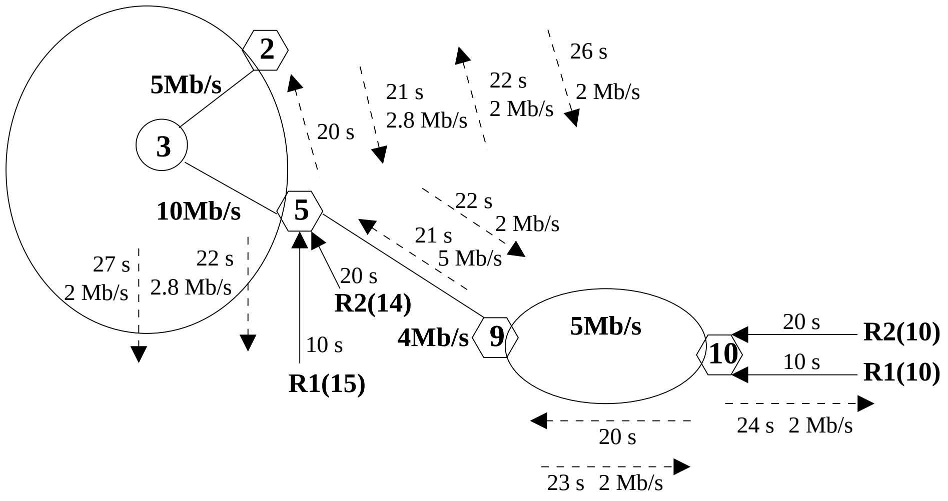


Figure 5: Messages after joining S_2

2 Mb/s toward the source and so r_5 receivers get, at second 27, a report with the final minimum UFR for S_1 (2 Mb/s). When a_5 receives this SYNC, it doesn't sent another SYNC toward a_{10}, because this value is equal to the one sent in the previous SYNC.

When S_2 ends, the UFR of S_1 becomes higher than its upstream fair rate in all links, and so only the agent closest to the source includes the new UFR of S_1 in a SYNC. Therefore, receivers closer to the source are notified first, as shown in table 1, and the notification time increases downstream in a order of $O(n_a, a_s)$, where n_a is the number of agents and a_s is 1 s. So, results presented in this section confirm the theoretical analysis of SAPRA.

5 Conclusion and Future Work

This paper describes and evaluates SAPRA, a protocol that allows the distribution of service resources along each session path proportional to the audience size of each session. SAPRA notifies receivers about the fair share of their sessions, allowing them to adapt to quality variations. We simulated a topology with ten domains, where the longest path has six domains. Theoretical analysis and simulation results show that SAPRA has a small bandwidth overhead, and that the protocol is efficient in updating session resources and sending notifications to receivers when sessions change significantly.

When the state of sessions changes, receivers get reports very quickly if the minimum fair rate of their session decreases. That time becomes longer if the minimum fair rate increases, because in this case a SYNC message is sent only by the agent closest to the source after session state was updated in all agents. Results also show that the time to notify receivers about increasing fair rates grows linearly with the number of agents in the path.

Although SAPRA aims mainly to distribute services resources with fairness, it also provides information about the quality of sessions and their population, which can be used to settle SLAs in each domain.

We plan to study a receiver-driven adaptation mechanism, where receivers adapt to bandwidth changes when they receive network reports. This synchronization helps to solve the leave latency problem and improves the adaptive mechanism stability. Being based upon the fair rate of sessions, the adaptive mechanism should solve unfairness between them. We will also study SAPRA in mobile environments, where the location of receivers change frequently.

References

[1] Charu C. Aggarwal, Joel L. Wolf, and Philip S. Yu. Design and analysis of permutation-based pyramid broadcasting. *Multimedia Systems*, 7(6):439–448, 1999.

[2] Dimitri Bertsekas and Robert Gallager. *Data Networks*. Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service. Request for Comments 2475, Internet Engineering Task Force, December 1998.

[4] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation protocol (RSVP) – version 1 functional specification. Request for Comments 2205, Internet Engineering Task Force, September 1997.

[5] M. Handley, C. Perkins, and E. Whelan. Session announcement protocol. Request for Comments 2974, Internet Engineering Task Force, October 2000.

[6] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured forwarding PHB group. Request for Comments 2597, Internet Engineering Task Force, June 1999.

[7] H. Holbrook and B. Cain. Source-specific multicast for IP. Internet Draft, Internet Engineering Task Force, March 2001. Work in progress.

[8] H. Holbrook and B. Cain. Using IGMPv3 for source-specific multicast. Internet Draft, Internet Engineering Task Force, March 2001. Work in progress.

[9] Jun ichi Kimura, Fouad A. Tobagi, Jose-Miguel Pulido, and Peder J. Emstad. Perceived quality and bandwidth characterization of layered mpeg-2 video encoding. In *Proc. of SPIE International Symposium*, Boston, Massachussetes, USA, September 1999.

[10] Mathias Johanson. Scalable video conferencing using subband transform coding and layered multicast transmission. In *Proc. of International Conference on Signal Processing Applications and Technology (ICSPAT'99)*, Orlando, Florida, USA, November 1999.

[11] Arnaud Legout, Joerg Nonnenmacher, and Ernst Biersack. Bandwidth allocation policies for unicast and multicast flows. In *Proc. of the Conference on Computer Communications (IEEE Infocom)*, New York, March 1999.

[12] Xue Li, Sanjoy Paul, and Mostafa Ammar. Multi-session rate control for layered video multicast. In *Proc. of Multimedia Computing and Networking*, San Jose, California, USA, January 1999.

[13] Paulo Mendes. "SAPRA: Session-Aware Popurality Resource Allocation fairness protocol". http://www.cs.columbia.edu/˜mendes/sapra.html.

[14] Paulo Mendes, Henning Schulzrinne, and Edmundo Monteiro. Session-aware popularity resource allocation for assured differentiated services. In *Proc. of the Second IFIP-TC6 Networking Conference*, page 9, Pisa, Italy, May 2002.

[15] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the internet. Request for Comments 2638, Internet Engineering Task Force, July 1999.

[16] Y. Rekhter and T. Li. A border gateway protocol 4 (BGP-4). Request for Comments 1771, Internet Engineering Task Force, March 1995.

[17] Dan Rubenstein, Jim Kurose, and Don Towsley. The impact of multicast layering on network fairness. In *Proc. of SIGCOMM Symposium on Communications Architectures and Protocols*, Cambridge, Massachussetes, USA, September 1999.

[18] Saswati Sarkar and Leandros Tassiulas. Fair allocation of discrete bandwidth layers in multicast networks. In *Proc. of the Conference on Computer Communications (IEEE Infocom)*, Tel Aviv, Israel, March 2000.

[19] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. Request for Comments 1889, Internet Engineering Task Force, January 1996.

[20] David Taubman and Avideh Zakhor. Multirate 3-D subband coding of video. *Journal of IEEE Transactions on Image Processing*, 3(5):572–588, September 1994.

[21] Telstra. "AS 1221 BGP statistics". http://www.telstra.net/ops/bgp/bgp-active.html.