Session-Aware Popularity-based Resource Allocation Across Several Differentiated Service Domains

Paulo Mendes, Henning Schulzrinne, Edmundo Monteiro

Department of Informatics Engineering
University of Coimbra, Coimbra, Portugal
{mendes,edmundo}@dei.uc.pt

Department of Computer Science
Columbia University, New York, USA
{mendes,schulzrinne}@cs.columbia.edu

Abstract The Differentiated Services model (DS) maps traffic into network services that have different quality levels. However, inside each service flows can be treated unfairly, since the DS model has no policy to distribute the service bandwidth between all sessions that compose the service aggregate traffic. Our goal is to study a signaling protocol that fairly distributes the resources reserved for each DS service between scalable multimedia sessions in a multicast network environment, where scalable sources divide session traffic in hierarchical layers, sending each layer to different multicast groups. We present a signaling protocol called *Session-Aware Popularity Resource Allocation* (SAPRA) that allows the distribution of DS services resources along sessions path, based upon the receiver population of each session. Simulations show that SAPRA protocol has small bandwidth overhead, is efficient updating the resources allocated to each session and also supplying receivers with reports about the quality level of their session.[1]

Key words: fairness, multimedia, multicast, differentiated services, scalable video sessions, signaling protocol.

1 Introduction

Almost all multimedia applications in the Internet use video flows with rates that don't change over time, even when dealing with receivers with heterogeneous capabilities. For example, RealNetworks SureStream [18] and Microsoft Windows Media [15] technologies support multi-rate encoding, i.e. the ability to select a video quality when multiple copies encoded at different rates are available. This approach leads to waste of bandwidth in heterogeneous environments, such as the Internet, because sources send copies of the same stream in order to satisfy requests with different quality requirements. This problem can be solved using scalable encoding [24, 9] that divides a video stream into cumulative layers with different rates and importance, with the rate of the stream being achieved adding the rates of all its layers. This approach doesn't waste bandwidth, since sources only send one stream to all receivers, mapping each layer to a different multicast group. Receivers join as many multicast groups as their connection speed allows [12], starting by the most important layer. We use the term *session* to encompass all layers belonging to the same stream, as in the Real-Time protocol (RTP) [22].

Due to their real-time characteristics, multimedia sessions need quality guarantees from the networks. These guarantees can be provided by the DS model [2], which allows network providers to aggregate traffic in different services at the boundaries of their network. Each DS service is based upon a per-hop behavior (PHB) that characterizes the allocation of resources needed to give an observable forwarding behavior (loss, delay, jitter) to the aggregated traffic. However, inside each service sessions can still be treated unfairly, since the DS model has no policy to distribute services bandwidth between all the sessions that compose their aggregate traffic, i.e., there is no *inter-session* fairness inside each DS service.

Therefore, we propose a fair mechanism called *Session-Aware Popularity Resource Allocation* (SAPRA), which provides *inter-session* fairness inside each DS service assigning more bandwidth to sessions with bigger receiver population. In the presence of scalable sessions, SAPRA can also provide *intra-session* fairness, assigning to each layer a drop precedence that matches its importance inside its session. However this is only possible if the DS service has the capability to assign different drop precedences to flows as happens, for example, with DS services based upon the Assured Forwarding PHB [5]. To achieve this goal, SAPRA is composed of an agent and a marker in each DS edge router. The agent computes sessions fair rate considering their number of receivers, while the marker have a per flow action, assigning different drop precedences to each session layer. SAPRA also includes a punishment function and a resource utilization maximization function. The former punishes high-rate sessions - sessions with a rate above their fair rate - in times of congestion, motivating sessions to adapt to the network capacity. The latter complements the fairness policy avoiding waste of resources. The SAPRA agent and marker are described and evaluated in [14]. In this paper we intent to study the SAPRA signaling protocol used by agents to exchange sessions

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

information, allowing a fair distribution of resources in the path of each session.

We present NS [17] simulations that evaluate the SAPRA protocol in a topology with ten DS domains, considering one DS domain per Autonomous Systems (AS). Simulations show that the protocol has small bandwidth overhead, and that it is efficient updating the resources allocated to each session and supplying receivers with reports about the quality level 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 operation and Section 4 presents some simulation results. Finally, Section 5 presents conclusions and future work.

2 Related Work

Resources reserved to services depends on 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 with dynamic SLAs, which can be based upon Bandwidth Brokers [16] or approaches that uses intra-domain signaling protocol, e.g., the Resource Reservation Protocol (RSVP) [3], to reserve resources where they are needed, avoiding over-provisioning along certain branches of a multicast tree. However these approaches don't distribute the service resources reserved 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 [1], but this definition can not exist with discrete set of rates [20]. Sarkar et al. presents a fair algorithm [21] that exists with discrete set of rates, but doesn't consider the population of sessions, and its scalability and efficiency isn't proved in a Internet-like scenario. The population of sessions is considered by Legout et al. [10], 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 [11] 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 and the importance of its layers.

3 SAPRA Protocol

In this section we describe and evaluate the SAPRA signaling protocol. We assume that edge routers have an accurate notion of the resources assigned to each DS service through static or dynamic SLAs, and that the solution implemented in each DS domain to achieve this is a policy issue and is beyond the scope of this paper. We denote the direction of packet traveling from the source toward receivers as downstream and the opposite direction as upstream. SAPRA doesn't demand the separation of multicast and unicast traffic into different DS services, since it handles unicast flows as sessions with one layer and one receiver. However we consider that SAPRA is more properly used to fairly distribute resources between large-scale sessions, i.e. sessions with a large number of receivers. Large-scale sessions, such as video-on-demand and InternetTV, are normally longer - minutes or hours - and fewer - hundreds - than the usual unicast services in the Internet.

SAPRA also doesn't restrict the location of multicast branch points, which can exist either in edges and interior routers, as shown in Fig. 1. Since agents need to distinguish sessions inside a DS aggregate and markers have a per layer action, they are placed only in edge routers. Therefore SAPRA doesn't imply alterations to DS interior nodes, being in conformity with the DS model. SAPRA contributes to an end to end fair allocation of DS service resources even if some DS domains don't implement SAPRA, since those domains are transparently crossed by SAPRA messages. However the population of sessions that have receivers in those domains isn't increased and so those sessions have a lower share of resources in the path to their source.

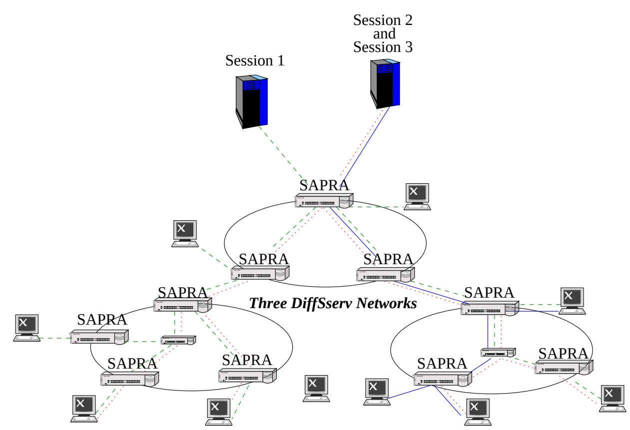

Figure 1. SAPRA location in a DS multicast tree.

In what concerns end hosts, SAPRA doesn't restrict the number of multimedia *sources* that can exist in the same host, which we name *sender*. When sources are scalable, we assume that each one generates a multimedia session with several layers, each layer identified by a Source-Specific Multicast (SSM) channel [6], i.e., the sender IP address and destination multicast group. Receivers can get information about sessions using, for example, the Session Announcement Protocol (SAP) [4]. A receiver can join more than one session at the same time, even if those sessions belong to the same sender. To join a session, receivers starts joining the SSM channel of the most important layer. Receivers can try to increase their reception quality joining more layers, always from the most important to the less important one.

In what concerns the network, SAPRA agents in leaf edge routers collect information about the local receiver population of each session from IGMPv3 "State-Changes" reports, when receivers join the most important layer of their session. To achieve a fair distribution of resources, edge routers keep information about each session. This information is kept by agents, outside the data path, while markers only keep, in the data path, information required to mark packets. SAPRA has to be a session-based mechanism and not only multicast-based, since hiding session information from DS edge routers results in *intra-session* unfairness, higher quality oscillations and lower quality for all receivers. The session aware characteristic of SAPRA is explained in detail in [14], where we also present two examples of mechanisms for agents in leaf routers to collect session information, namely the number of layers in each session and the importance of each layer inside the session. In the first method, each sender allocates consecutive multicast addresses to all layers inside a session and keeps one address gap between sessions. With SSM this method doesn't bring any address allocation problem, since each source is responsible for resolving address collisions between all the channels - 2^{24} in IPv4 and 2^{32} in IPv6 - they create. With this method, DS-edge routers identify as belonging to the same session all layers that receivers join with consecutive SSM channels. In the second proposed method the *auxiliary* data field of IGMPv3 [7] reports can be used to include the multicast address of the most important layer - which identifies the session - in reports about other layers. So, DS-edge routers explicitly know at what session each layer belongs to. Routers that don't implement SAPRA ignore the *auxiliary* data field as is done in the current IGMPv3 implementations. In both proposed methods, routers know the relationship between layers in a session by the order receivers use to join sessions' layers. Receivers are motivated to join layers from the most important to the less important one, because less important layers are useless without the most important ones in the re-construction of the session's multimedia stream.

Based upon the information (receiver population and fair rate of each session) collected using the SAPRA signaling protocol from downstream neighbors, agents compute the local fair rate that each session is entitle to. Fig. 2 illustrates how agents do this computation for two sessions, S_1 and S_2, in a DS domain with three edge routers. As we described in [14], agents compute each session local fair rate in a link - *Local Fair R*ate (*LFR*) - based upon the receiver population that the session has in the link, assigning more resources to sessions with higher number of receivers. Since in this example all links have 5 Mb/s, the *LFR* of S_1, for example, in E_2 is 1.25 Mb/s, because S_1 has 100 receivers and the receiver population in the link is 400. The fair rate that effectively is used for a session in a link - *Used Fair Rate* (*UFR*) - is the minimum value between its *LFR* and its fair rate downstream this link - *Downstream Fair Rate* (*DFR*) -, avoiding wasting downstream resources. In this example, the *DFR* in the egress routers in zero, since there aren't links downstream from those routers. This way the *DFR* isn't used in the egress routers to estimate the *UFR* of each session, which becomes equal to *LFR*. In the ingress router, the receiver population of a session is the sum of all receivers that session has in each egress router and its *DFR* is the maximum value of the *UFR* that a session has in each egress router. This maximum value guarantees the satisfaction of the heterogeneous requirements of all receivers that the session has in different egress routers, without leading to a congestion situation beyond the domain since packets will be dropped in egress routers where the session has lower *UFR*. Like was mention before, we assume that each DS domain has policies to assign resources to each DS service in order to avoid congestion its interior routers. Therefore S_1, for example, has in the ingress router an estimated population of 300 receivers (200 from E_1 and 100 from E_2), a *DFR* of 2 Mb/s (maximum value between the *UFR* in E_1 and E_2) and a *UFR* of 2 Mb/s, which is the minimum value between the *LFR* (2.5 Mb/s) and the *DFR*. If the egress routers were connected to several ingress routers, the fair rate of each session, in these egress routers, would have been estimated in the same way as it was for the ingress router in this example.

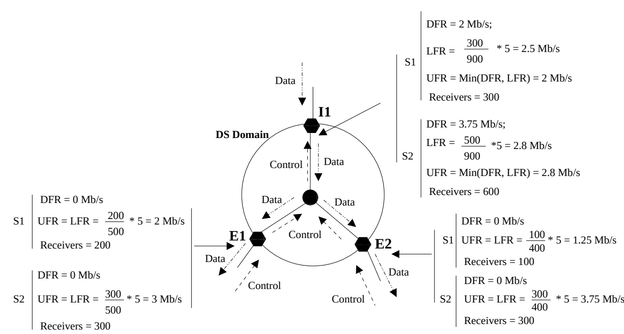

Figure 2. Example of fair rate estimation in edge routers.

The *UFR* of each session is passed to the markers of downstream links that have traffic from that session, being used to mark packets IN or OUT of profile, depending if the session estimated rate is lower or higher than their *UFR*. Traffic of new sessions is marked as OUT of profile while markers don't have information about those sessions. This might happen when the multicast protocol is faster propagating group membership information than the SAPRA protocol propagating session information.

Next we explain in more detail the operation of the SAPRA protocol.

3.1 Overview of Protocol Operation

The main goal of SAPRA protocol is to allow each agent to collect downstream information of sessions in order to fairly distribute the local DS service bandwidth between them. To achieve this, SAPRA defines an *UPDATE* message that is sent by each agent to its upstream neighbors with information about the receiver population and *UFR* of each session. Each upstream neighbor only receives information about sessions whose sources are upstream from it.

SAPRA also aims to allow agents on leaf routers to have an updated notion of the minimum *UFR* that sessions, with local receivers, have in the path from their sources. With this information, agents on leaf routers can provide receivers with periodic information about their session fair share of the network resources, allowing receivers to adapt to quality oscillations. To achieve this, the SAPRA protocol defines a *SYNC* message that agents send to their downstream neighbors with the minimum *UFR* that sessions have in the path since their sources. Each downstream neighbor only receives information about sessions that have receivers downstream from it.

Since SAPRA protocol is only used to exchange information between agents in neighbor edge routers, agents don't need to have global network knowledge, which increases SAPRA scalability. Edge routers use Border Gateway Protocol's (BGP) [19] routing information to forward *UPDATE* messages, while *SYNC* messages use the previously collected routing information and traverse exactly the reverse path that *UPDATE* messages took. SAPRA control messages run over TCP, increasing the protocol robustness eliminating the need to implement fragmentation and re-transmission.

The management of the moment when agents send SAPRA messages, could be done as follows: agents could only sent a message after receiving a message of the same type, i.e, agents would send a *SYNC* per session after receiving a *SYNC* from the upstream neighbor in the path to the session source, and they would send an *UPDATE* per session after receiving *UPDATE* messages from all its downstream neighbors where that session is present. However, this approach has two main disadvantages: First, *UPDATEs* can be delayed if agents take a long time to gather information about each session from all their downstream neighbors. This can happen, for example, if there are different number of routers between the agent and its neighbors, or if neighbors send information about the same session at different times; second, agents should include in each *UPDATE* information of several sessions in order to reduce the protocol overhead . However *UPDATEs* can only be sent after agents have received downstream information about all sessions, which delays even more the propagation of sessions information upstream.

Therefore, SAPRA uses a lightweight approach where agents send messages periodically, independently of the reception of messages from their neighbors, simplifying the protocol. In order to decrease the time needed to propagate messages in the tree, the periodic sending interval should be short and it shouldn't be synchronized in all agents: the non synchronization of agents allows them to collect as much information as possible about sessions, before sending their next message, reducing the overhead. To achieve this, a random value is added to the sending period. Since the random value should be small to avoid delaying messages, its default interval is [-0.1, +0.1].

3.1.1 Update messages

When agents receive the first join from a local receiver or when they receive the first *UPDATE* from a downstream neighbor, they start to send *UPDATEs* periodic, with a default interval of 1 s. Each *UPDATE* includes information about all non-idle sessions, i.e., sessions that have a high variation of their receiver population and/or *UFR* since the last time they were included in an *UPDATE*, considering the total receiver population and total *UFR* in the upstream link. So, if an agent only have idle sessions, it doesn't send any *UPDATE*. However when a session ends, its information (no receivers and zero *UFR*) is always included in the next *UPDATE* allowing the immediate release of unnecessary state in upstream agents. The required minimum variation has the default value of 25%. This showed to be a good compromise between the number of *UPDATEs* and the propagation of accurate information to allow a fair allocation of resources between sessions. This restriction allows us to use short sending intervals, improving the protocol accuracy. Considering that sessions are normally long, intervals of 1 s are lower enough to propagate significant changes in sessions population, which should varies more significantly in the beginning and end of sessions.

Sessions have a hard-state in agents, since they are started and ended with the reception of *UPDATE*s. However they also have a soft-state property, in order to deal with routing changes and link failures. To implement this, the state of each session is refreshed by periodic *complete UPDATE*s. We name them *complete*, because they include information about all sessions, even if their state don't change for more than the minimum required. These *complete UPDATE*s are sent periodic with a default interval of 30 s, and the session state has to be refreshed every 90 s, in order not to be deleted. The former default value is equal to the BGP *keep_alive_time* variable, while the latter is equal to the BGP *hold_time* variable. The utilization of these values establishes a relationship between SAPRA refreshment of sessions state in agents and BGP refreshment of routing information between the edge routers where those agents are.

When agents receive *complete UPDATE*s, they only update sessions whose *DFR* varies by more than the minimum required. This allows the detection of sessions with little downstream state variation, avoiding unnecessary computation that don't lead to a significant change of local state. Similar procedure isn't done with the number of receivers, since this would lead to an incorrect knowledge about the downstream receiver population of sessions.

3.1.2 Sync messages

Agents in leaf edge routers receive in *SYNC*s the minimum *UFR* that each session has in its path, which enables them to send to local receivers a report with the fair rate of their sessions. Receivers can use this information to adjust their quality reception. As an example of a simple adaptation mechanism, receivers could join additional layers while the monitored rate remains below their session minimum *UFR*. In a more aggressive approach, receivers could join an extra layer even if the monitored rate becomes higher than the minimum *UFR*, trying to get extra resources. In this case, if losses increase, receivers should leave the less important layer and only join it again after receiving a report with a higher *UFR*.

Agents start sending *SYNC*s when they receive the first *UPDATE* from one of their downstream neighbors, if their router has local sources. Otherwise, agents only start sending *SYNC*s when they receive the first *SYNC* from one of their upstream neighbors. After this, agents send *SYNC*s periodically with a default interval of 1 s.

The fair rate value included in *SYNC*s for each session, depends on the source location of that session. If the source is local, that value is the *UFR* in the link were the *SYNC* is going to be sent. If the source is located upstream, that value is the minimum of the local *UFR* and upstream fair rate. In both cases the fair rate is only included in the *SYNC* if it's different from the one sent in the last message, which allows the use a short sending interval.

3.2 Example of Protocol Operation

Fig. 3 illustrates how SAPRA works, using two sessions S_1 and S_2. In this example, we assume that all DS domains links have 10 Mb/s and that links between routers 1 and 2, routers 5 and 8 and routers 4 and 6 have bandwidth of 10 Mb/s, 3 Mb/s and 4 Mb/s, respectively. In the rest of this paper we represent an edge router i as r_i, the agent on that router as a_i and the link between two edge routers r_i and r_j as $link_{(r_i,r_j)}$. In Fig. 3 the notation $U@s$ represents the moment when an *UDPATE* packet is sent. For example, $U8@14s$ means that the UPDATE number 8 was sent at second 14. The notation $Y@s$ has an equal meaning for *SYNC* messages.

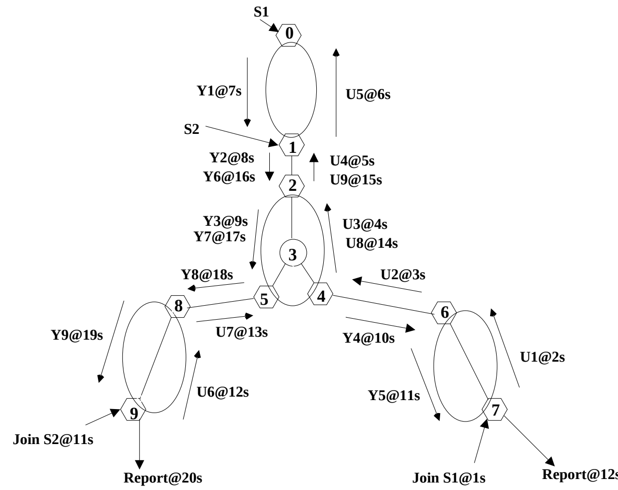

Figure 3. Example of the protocol operation.

When receivers join S_1, a_7 sends an *UPDATE* (U_1) toward a_0. When a_1 receives the *UPDATE* U_4 , it also starts sending *SYNC*s, since it has a local source. However since this local source isn't the source of the session in U_4, no *SYNC* is sent at second 6 by a_2. The first *SYNC* is sent by a_0 toward a_7, at second 7, with 4 Mb/s as S_1 minimum *UFR* on the tree - *UFR* for S_1 on $link_{(r_4,r_6)}$. Receivers on r_7 get a reports 12 s after joined S_1.

When receivers join S_2, a_9 sends an *UPDATE* (U_6) toward a_1, whose router has S_2 source directly connected. No *UPDATE* is sent from a_1 to a_0 since S_2 is not present in $link_{(r_0,r_1)}$ and the receiver population and *UFR* of S_1 didn't change in that link. At second 8, a_1 sends a *SYNC* (Y_2) with 3 Mb/s as S_2 minimum fair rate on the tree - *UFR* for S_2 on $link_{(r_5,r_8)}$. That information is received by a_9, on *SYNC* (Y_9), 8 s after receives had joined S_2. No *SYNC* is sent from a_2 toward a_7, passing through a_4, since S_2 isn't present in that branch and S_1 didn't change upstream.

If we decrease the bandwidth of $link_{(r_1,r_2)}$ to 5 Mb/s, the *LFR* for the two sessions would be 2.5 Mb/s, which is lower than the *DFR* of S_1 and S_2 - 4 Mb/s and 3 Mb/s, respectively. In this case, a_1 would have sent 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 - variation higher than the minimum required - and would have included in *SYNC* Y_6, not only the *UFR* of S_2 but also of S_1, which are both 2.5 Mb/s. Therefore a_2 would also have sent a *SYNC* to a_4, in addition to the one that sends to a_5. A *SYNC* would have been sent from a_4 to a_6, since the minimum value between the *UFR* (4Mb/s) and the upstream fair rate (2.5 Mb/s) is different from the value sent in the previous *SYNC* (4 Mb/s). This way S_1 receivers on r_7 would have 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 on receivers population or fair rates. However at second 30 (sending interval of *complete UPDATE*s), agents send an *UPDATE* with information about all sessions they have, to refresh the state of all sessions.

3.3 Protocol evaluation

We analyze the required stored state, the bandwidth overhead and the delay between the moment when sessions change and the moment their resources are updated in all agents and their receivers get a report.

Although not directly related with the protocol, we briefly refer to stored state since this is important to the overall scalability of SAPRA. All the required state is stored outside the data path, in agents, keeping in the data path only the *UFR* of each session to allow markers to mark packets. SAPRA scalability is increased because the stored state is of order *O(N)*, where N is the number of sessions. Although sessions are multicast, the stored state doesn't depend on the number of receivers, i.e., doesn't depend on the network size.

Another scalability issue is the protocol bandwidth overhead, which is significantly reduced, since *UPDATEs* aren't sent if there are only idle sessions, *SYNCs* aren't sent if there are no sessions with a different fair rate (minimum of *UFR* and upstream fair rate) from the one sent in the previous message, and SAPRA messages are only sent by each agent to their neighbors. The latter property makes SAPRA overhead independent of the network size, being only dependent on the number of sessions and the sending intervals. Therefore we analyze the protocol scalability in a extreme scenario where the sending interval of both messages is short - 1 s - and the number of sessions is high - 1000. We assume that all sessions are present in all links, that each session has a rate of 1.8 Mb/s distributed in 3 layers, and that they are very dynamic with significant changes every second. Since the information about each session occupies 48 bytes in *UPDATE*s and 12 bytes in *SYNCs*, and since we have 1000 sessions, the control information between every two agents has a rate of 480 kb/s. Since the data rate of all sessions in each link is 1800 Mb/s, the ratio between control and data bandwidth is 0.0003 (0.03%). Even in the present of low-rate sessions SAPRA still have an insignificant overhead. For example, if we assume sessions with 64 Kb/s and the same 3 layers, the protocol overhead is 0.75%.

Although SAPRA has an insignificant bandwidth overhead, the protocol needs to be fast updating sessions information in all agents and supplying receivers with updated reports. The time required to do this depends on the number of agents in each session path. In the worst case, the time required to update sessions information is given by Eq. 1 and the time for receivers to get a report by Eq. 2, where n_a is the number of agents and a_s the sending interval in each agent[2]. The worst case happens when an *UPDATE* message is the first one to arrive to each agent, because only at this point the agent starts sending messages, and so this *UPDATE* message will be delayed in each agent for the totality of the sending interval. However, the time required to update sessions information and notify receivers is shorter when all agents already started sending messages. This is illustrated in Fig. 3, where the reception delay of *SYNC* Y_5 correspond to the worst case, while the reception delay of Y_9 is shorter since several agents already had started sending messages when receivers joined session S_2.

$$(n_a - 1)\, a_s \quad (1)$$

$$(2\,(n_a - 1) + 1)\, a_s \quad (2)$$

So, in the worst case, for a path with six DS and thus 12 agents, the time required to update sessions information is 11 s and receivers get the first report 23 s after they joined their session. We don't consider paths longer than six DS domains since the percentage of those paths is very low, e.g., 0.5% for seven domains and 0.1% for eight domains, regarding BGP data obtained from five major operators in November 2001 and related with 60978 different ASs [25], and assuming one DS domain per AS. This mean that, utmost 11 s after changes happen, the *UFR* of each session in each link is update to the lowest value in their path and that utmost after 23 s the upstream fair rate of each session, in each link toward their receivers, becomes equal to the session minimum *UFR*.

Besides this, receivers are notified faster if changes lead to a decrease in their session minimum *UFR*, because agents only include in a *SYNC* the minimum value between the *UFR* and the upstream fair rate if this value is different from the one sent in the previous *SYNC*, and because in each stabilized network branch the upstream fair rate of sessions in each link is equal to their minimum *UFR* on the branch[3]. This means that if changes decrease the *UFR* of a session in a link below the upstream fair rate, the agent includes this new minimum *UFR* in the next *SYNC*, and so receivers are notified in the next report. However if the *UFR* increases to values higher than the upstream fair rate, the agent doesn't include information about this session in the next *SYNC*, because the upstream fair rate, which is lower than the new *UFR,* was already sent in the previous *SYNC*. However, if the agent is the last one before the session source, it includes the new *UFR* in the next *SYNC*, since this is the minimum *UFR* in the session path.

On the one hand, delayed notifications of increasing fair rates lead to a higher stability since receivers only join layers when the resources allocated to their session are already updated in the entire path, avoiding leaving

[2] Assuming the same sending interval for *UPDATEs* and *SYNC*s.

[3] If changes happen in different branches at the same time, the stabilization of fair rates is achieved first in each branch and only then in the entire network.

latency problems [12]. On the other hand, faster notifications of decreasing fair rates optimize resource utilization in the network, since receivers can more quickly leave layers, reducing network traffic.

4 Simulations

In this section we present simulations that show the protocol ability to allocate resources between sessions in multicast trees passing through several DS domains. Simulations showing SAPRA agent and marker operation in each DS edge router are presented in [14].

We use a scenario with ten DS and one DS domain per AS as shown in Fig. 4, where the longest path has six DS domains - based upon BGP data obtained from AS 1221 [25]. We use a topology with sources (S_1 to S_4) in different DS domains, multicast branches with different number of domains, DS domains with multicast branches in ingress and interior routers, receivers in ingress and egress routers and egress routers only with local receivers or 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 obtained from AS 1221 BGP data to decide upon receivers location. Hence, we place 26%, 40%, 26%, 9% and 2.5% of receivers in DS domains at a distance of two, three, four, five and six domains from their source, respectively. In Fig. 4, $R_x(Y)$ is a notation for Y receivers of session S_x .

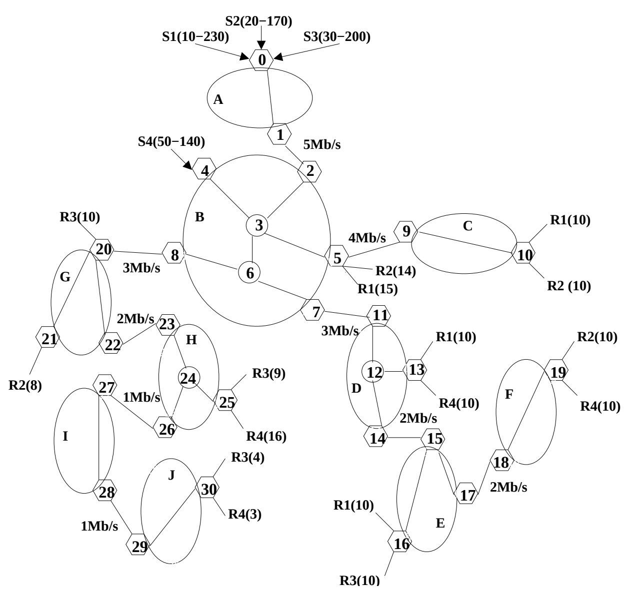

Figure 4. Simulation scenario.

We assume that the DS service capacity in each edge router link is equal to the link bandwidth, and that intra-domain links have enough bandwidth to avoid congestion since the relation between SAPRA and the reservation protocol used in each DS domain is beyond the scope of this paper. We also assume that each queue has a size of 64 packets, which is the default value in Cisco IOS 12.2 [23] and that interior routers have FIFO queues, since these are the simplest ones. We use data packets with a size of 1000 bytes since this is a value between the MTU of dial-up connections - 576 bytes - and the MTU of Ethernet and high speed connections - 1500 bytes. Since control packet are mainly exchanged between routers, *UPDATE* and *SYNC* packets have a size of 1500 bytes, which is the default MTU in Cisco routers. These values for data and control packets reduce the number of headers, routing decisions, protocol processing and hosts device interrupts, without increasing fragmentation.

We use four sessions: S_1 from second 10 to 230, S_2 from second 20 to 170, S_3 from second 30 to 200 and S_4 from second 50 to 150. Each session has a rate of 1.8 Mb/s divided into three layers. The most important layer has 303 kb/s, the medium 606 kb/s and the less 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 [8]. The simulations last 240 s simulations, which is enough to identify transient state when sessions start and end and to show that sessions remain stable in between. With longer simulations sessions remain stable for a longer time, since the time needed for their state to stabilize doesn't change. Results of an 600 s simulation can be found in [13]. We assume that receivers join when sessions start and leave when they end, since this should be the common behavior of viewers of large-scale sessions.

First we analyze the protocol bandwidth overhead. In this simulation a total of 375 *UPDATEs* and 271 *SYNCs* were exchanged between 27 agents. Since each *UPDATE* has 48 bytes and each *SYNC* has 12 bytes, *UPDATEs* have a rate of 0.6 Kb/s and *SYNCs* of 0.1 Kb/s among all agents during the 240 s simulation. Since the four sessions have a total data rate of 7.2 Mb/s, the total control bandwidth overhead is approximate 0.010%.

Next we analyze how fast sessions resources are updated and receivers get network reports, namely the first report. For this, we use the information[4] dumped every 5 s by S_1 receivers in the edge routers 5, 10, 13 and 16, as shown in Fig. 5. Results for S_2, S_3 and S_4 can be found in [13].

Receivers on r_5 get their first network report at second 17, i.e., 7 s after they had joined S_1, which complies with Eq. 2. Using this equation, receivers on r_{10} and r_{13} should get their first report at second 21 and receivers on r_{16} at second 25. However, these receivers get their first report earlier, because some agents in the path to S_1 source already had started sending messages. In the case of receivers on r_{10}, they get the first report at second 19, because when a_5 sends a report to its receivers at second 17, it also sends a *SYNC* toward a_{10}. This agent receives the *SYNC* one second after that, due to the sending in-

[4] The session rate monitored by receivers and the fair rate they collect from *SYNCs*.

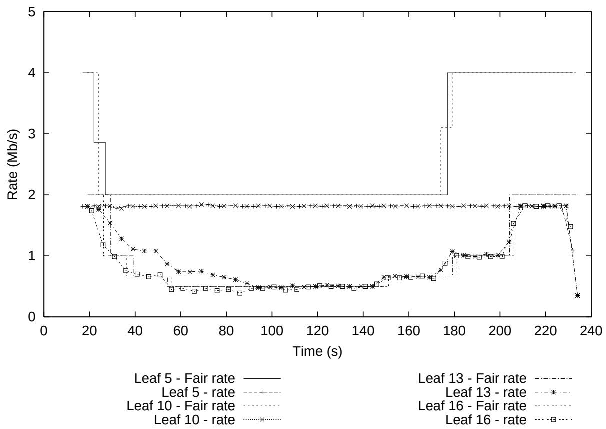

Figure 5. S_1 rate and *UFR* viewed by its receivers.

terval of a_9, and sends a report to its receivers at second 19. In the case of receivers on r_{13} and r_{16}, they get a report at second 19 and 21, because a *SYNC* is sent by a_2 and a_{11}, respectively. These results show that the delay of the first report is lower or equal to the one given by Eq. 2, depending upon the number of agents on the path that already started sending messages.

After receiving the first report, receivers get reports when the allocated resources change. The reception moment depends on the number of agents that already started sending messages, the receivers position in the tree and if resources increases or decreases. As an example, Fig. 5 shows when S_1 receivers get reports after receivers have joined S_2 on r_5, r_{10} and r_{19} at second 20 and leaving at second 170. Table 1 shows these results more clearly.

	router 5	router 10	router 13	router 16
S2 join	27s	24 s	29 s	25 s
S2 leave	177 s	179 s	179 s	181 s

Table 1. Moment when S_1 receivers get reports.

After receivers joined S_2, S_1 *UFR* decreases to 1 Mb/s in $link_{(r_{14},r_{15})}$. Since this value is lower than the upstream fair rate of 2 Mb/s, which was sent in the previous *SYNC*, a_{14} includes this value in the next *SYNC*, and so S_1 minimum *UFR* is reported first to r_{16} receivers than to r_{13} receivers, as shown in table 1. However, if $link_{(r_7,r_{11})}$ had lower capacity, decreasing again S_1 *UFR*, r_{13} receivers would have received that new minimum value before r_{16} receivers, since the *SYNC* that would have been sent by a_7 would have arrived first to r_{13} receivers than to r_{16} receivers.

Table 1 also illustrates that r_{10} receivers get a report with S_1 minimum *UFR* in their multicast branch (2 Mb/s) first than r_5 receivers. This happens since S_1 minimum *UFR* is decreased by a_5. Even though, Fig. 5 shows that r_5 receivers get at second 21 a report with a S_1 *UFR* of 2.8 Mb/s. This transient value corresponds to S_1 minimum *UFR* in the branch upstream r_5 before second 21, but it isn't the minimum *UFR* in the network, since S_1 *UFR* on $link_{(r_5,r_9)}$ (2 Mb/s) wasn't yet taken into account upstream r_5. More specifically, when a_5 gets a *SYNC* at second 21, it sends S_1 *UFR* of 2.8 Mb/s to its receivers. However, a_5 had already estimated an *UFR* of 2 Mb/s on $link_{(r_5,r_9)}$ after receiving an *UPDATE* from a_{10}, and so, it sends at second 22 a *SYNC* toward a_{10} with S_1 *UFR* of 2 Mb/s, since this value is lower than the upstream fair rate (2.8 Mb/s) and it's different from the fair rate previously sent (4 Mb/s). At second 22, a_5 also sends an *UPDATE* with S_1 *UFR* of 2 Mb/s toward the source, and so r_5 receivers get, at second 27, a report with S_1 upstream fair rate of 2 Mb/s. When a_5 receives this *SYNC*, it doesn't sent another *SYNC* toward a_{10}, since the minimum value between the *UFR* and the upstream fair rate value doesn't change. This exchange of messages is illustrated in Fig. 6.

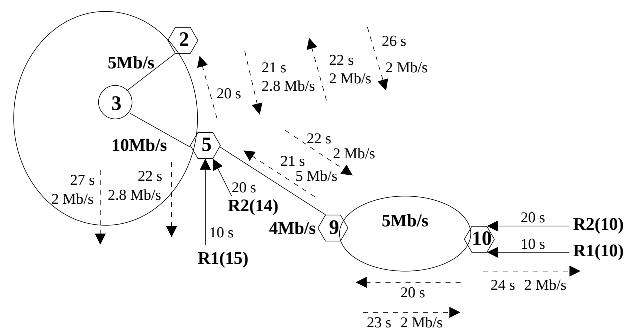

Figure 6. Messages after receivers joined S_2.

After receivers left S_2, S_1 *UFR* becomes, in each link, higher than S_1 upstream fair rate. Therefore no agent includes S_1 information in *SYNCs*, except for the last agent before the source, and so receivers near S_1 source get the new report first than receivers downstream, as is shown in table 1.

Simulation results after receivers join and leave S_2, show that reports about the decrease of the minimum *UFR* of S_1 arrive to receivers earlier than reports about its increase. For example, r_{16} receivers get a report 5 s after S_2 join and only 11 s after S_2 leave and r_{10} receivers get a report 4 s after S_2 join and 9 s after S_2 leave. The report received by r_5 and r_{13} receivers, due to a decrease of the minimum *UFR* of S_1 has the same delay has the report due to an increase - 7 s for r_5 receivers and 9 s for r_{13} receivers. This happens because both reports are originated by a *SYNC* sent by the same agent near the session source, since these receivers are upstream the agent that decreased the minimum *UFR* of S_1.

Another evidence is that the reporting delay increases downstream in a order of $O(n_a\, a_s)$, where a_s is one in these simulations. For example, r_5 is at a distance of 4 edge routers from the source, r_{10} and r_{13} at a distance of 6 edge routers and r_{16} at a distance of 8. Hence, S_1 receivers on r_{10} and r_{13} have the same reporting delay and the receivers on r_{10} and r_{16} get a report 2 s after the receivers on r_5 and r_{13}, respectively.

Results presented in Fig. 5 also show that the SAPRA protocol, agents and markers work properly, since S_1 rate is always below its minimum *UFR*. The exception is the rate monitored by r_{13} receivers, from seconds 29 to 89. This happens because the sum of *UFR* of all sessions on $link_{(r_7,r_{11})}$ is always lower than the link capacity, as can be seen in Fig. 7. Therefore, a certain percentage

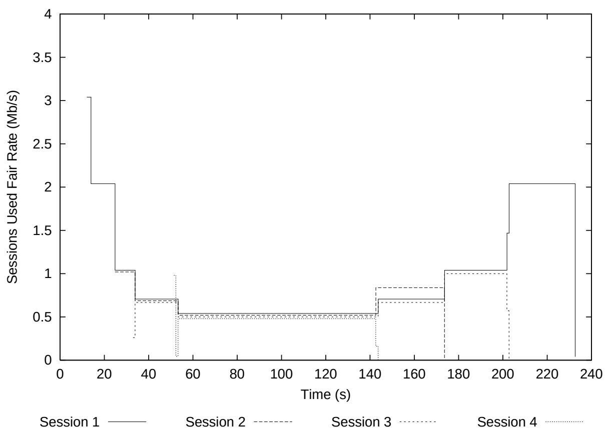

Figure 7. *UFR* between routers 7 and 11.

of OUT packets from all sessions pass through r_7 allowing r_{13} receivers to get a rate higher than the minimum UFR. This doesn't happen with r_{16} receivers, because $link_{(r_{14},r_{15})}$ don't have enough bandwidth to encompass the UFR of all sessions.

However, Fig. 5 also shows that S_1 rate on r_{13} decreases from seconds 29 to 89. This is due to the punishment mechanism, implemented in the marker, that starts filtering S_1 layers after this session had been identified as a *high-rate session* as soon as $link_{(r_7,r_{11})}$ become congested. *High-rate session*s are sessions that have a rate higher than their *UFR* plus their share of the available bandwidth, which exists because other sessions have their *UFR* decreased to match their *DFR*, releasing local resources. In each link, the available bandwidth is divided in equal parts between all non *high-rate sessions*.

SAPRA is able to distribute resources in each link, decreasing sessions *UFR* to release resources for new sessions or increasing them to use resources made available by ended sessions, as is illustrated on Fig. 7 by the *UFR* variation on $link_{(r_7,r_{11})}$. The only irregular behavior in Fig. 7 is the initial variation of S_4. The *UFR* of this session should be 0.5 Mb/s, which is its *DFR* - *UFR* on $link_{(r_{14},r_{15})}$. However at second 51, r_7 has only knowledge about S_4 receivers on r_{13}, and so S_4 has a *LFR* on $link_{(r_7,r_{11})}$ equal to sessions S_2 and S_3, i.e., 0.6 Mb/s, while S_1 *LFR* is 1.2 Mb/s. But since S_1 has a *DFR* of 0.6 Mb/s, its *UFR* becomes equal to 0.6 Mb/s and the available 0.6 Mb/s is used to increase the *UFR* of other sessions, which means that S_2 and S_3 *UFR* increase to 0.667 Mb/s - their *DFR* - and S_4 keeps the remaining available bandwidth, being its *UFR* increased to 1 Mb/s. At second 52, r_7 receives the first *UPDATE* with information about S_4 receivers on r_{19}. At this point S_4 has a *UFR* of 0.06 Mb/s on $link_{(r_{14},r_{15})}$, since only one receiver had joined when a_{19} sent an *UPDATE* at second 50. Therefore the *UFR* of S_4on $link_{(r_7,r_{11})}$ *is* 0.06 Mb/s, since its *LFR* on that link is higher than that value (0.66 Mb/s, corresponding to 11 receivers). After second 53, all S_4 receivers have joined the session and so S_4 *UFR* on $link_{(r_{14},r_{15})}$ increases to 0.5 Mb/s and consecutively also on $link_{(r_7,r_{11})}$.

5 Conclusion and Future Work

This paper describes and evaluates the *Session-Aware Popularity Resource Allocation (*SAPRA*)* protocol that allows the distribution of DS services resources along session path, based upon the receiver population of each session. SAPRA protocol also provides agents on leaf routers with updated information about the minimum fair rate that sessions have in the network, allowing them to send to receivers information about their session fair share of the network resources. This allows receivers to adapt to quality oscillations.

Simulations were made with a topology having ten DS domains, where the longest path includes six DS domains. Simulation results show that SAPRA has small bandwidth overhead, and that the protocol is efficient updating sessions resources and sending updated reports to receivers, when sessions significantly change. Namely, the required time for receivers to get the first report is equal to the value given by Eq. 2, or lower than that when there are agents in the path to their source that are already sending messages. When sessions state change, receivers get reports very quickly if the minimum fair rate that their session use in the network decreases. The time to receive reports is longer if that minimum fair rate increases, because in this case a *SYNC* is only sent by the agent nearest the source after the sessions state had been update in all agents. However, simulations show that the maximum notification delay is of 23 s, when the path length has six DS domains. Results also show that the report delay increases linearly with the number of agents in the path.

Since SAPRA distributes bandwidth between sessions based upon their receiver population, the motivation to use multicast increases. Besides this, SAPRA can help in the management of DS domains providing information about the quality level and receiver population of sessions. This information can be used, for example, to dynamically distribute resources between DS services in a domain. Furthermore, if sources use dynamic hierarchical schemes, they could employ the minimum fair rate of their sessions to adapt their layers rates.

As future work we'll study a receiver-driven adaptive mechanism, where receivers adapt to quality oscillations based upon the minimum fair rate used by their sessions. On the one hand, being based upon sessions minimum fair rate in the network the adaptive mechanism should solve unfairness between sessions and the leave latency problem described by McCanne et al. in [12]. On the other hand, the adaptive mechanism stability can be improved, since it will only be triggered by *SYNCs*. We'll use this adaptive mechanism to study the sharing of DS services resources between UDP and TCP traffic. We'll also study the use of SAPRA in mobile environments, where receivers location change very often between edge routers.

References

1. Dimitri Bertsekas and Robert Gallager. *Data Networks.* Prentice-Hall, Englewood Cliffs, New Jersey, 1987.
2. 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.
3. R. Braden, 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.
4. M. Handley, C. Perkins, and E. Whelan. Session announcement protocol. Request for Comments 2974, Internet Engineering Task Force, October 2000.
5. J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured forwarding PHB group. Request for Comments 2597, Internet Engineering Task Force, June 1999.
6. H. Holbrook and B. Cain. Source-specific multicast for IP. Internet Draft, Internet Engineering Task Force, March 2001. Work in progress.
7. H. Holbrook and B. Cain. Using IGMPv3 for source-specific multicast. Internet Draft, Internet Engineering Task Force, March 2001. Work in progress.
8. 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.
9. 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)*, Orlando, Florida, USA, November 1999.
10. 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.
11. 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.
12. Steven McCanne, Van Jacobson, and Martin Vetterli. Receiver-driven layered multicast. In *Proc. of SIGCOMM Symposium on Communications Architectures and Protocols*, pages 117–130, Palo Alto, California, USA, August 1996.
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. Microsoft. "Windows Media Techologies". http://www.microsoft.com.
16. 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.
17. NS. "Network simulator". http://www.isi.edu/nsnam/ns/.
18. RealNetworks. "Real system producer". http://www.realnetworks.com.
19. Y. Rekhter and T. Li. A border gateway protocol 4 (BGP-4). Request for Comments 1771, Internet Engineering Task Force, March 1995.
20. 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.
21. 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.
22. 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.
23. Cisco Systems. "Cisco IOS quality of service solutions configuration guide. release 12.2".
24. 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.
25. Telstra. "AS 1221 BGP statistics". http://www.telstra.net/ops/bgp/bgp-active.html.