A Multi-state Congestion Control Scheme for Rate Adaptable Unicast Continuous Media Flows[1]

Pantelis Balaouras, Ioannis Stavrakakis

Communication Networks Laboratory (CNL)
Department of Informatics and Telecommunications
University of Athens (UoA)
15784, Athens, Greece
{P.Balaouras, istavrak} @di.uoa.gr

Abstract. In this paper a multi-state rate adaptation scheme for rate adaptable continuous media (CM) sources is presented; the network bandwidth availability information is assumed to be communicated to the source only indirectly, through a notification of the packet losses. The developed scheme is capable of detecting and responding well to static (due to access bandwidth limitations) and long-term (due to initiation/termination of CM flows) bandwidth availability, as well as locking the rate of a CM flow to an appropriate value. Its performance is compared with that under a classic Additive Increase Multiplicative Decrease algorithm and its shown to perform better in terms of a number of relevant metrics, such as packet losses, oscillatory behavior, and fairness.

1. Introduction and Background

The network layer of Internet currently provides no inherent congestion control. In current IP environments, there is no central entity to dictate an explicit rate for the flows or explicitly indicate the current network state like, for example, in an ATM environment; ECN (Explicit Congestion Notification) based schemes are still far from being adopted and widely deployed [1,2]. Thus, congestion control may be provided through mechanisms implemented either by the transport or the application layers. A flow source would use its own application or transport level information concerning packet losses to infer the state of the network, resulting in a distributed rate control scheme. Different users may infer a different network state. The Transport Control Protocol (TCP) protocol provides congestion control, whereas the User Datagram Protocol (UDP) does not.

One of the emerging services appearing in the Internet is the continuous media (CM) streaming service, such as video streaming, conferencing and "on demand" services. CM applications require more bandwidth, compared to the typical data block

[1] This work has been supported in part by the IST Program "Videogateway" of the EU (IST-1999-10160).

transfer services (e.g., Web access over TCP, file transfer via FTP, e-mail), and are sensitive to the network induced delay, delay jitter and packet loss.

TCP is not suitable for multimedia transmission, because of its window-based congestion scheme [3,4,5]. For CM services UDP is used as the base transport protocol, in conjunction with the Real Time Protocol (RTP) and Real Time Control Protocol (RTCP) [6]. RTP provides end-to-end delivery services for real-time data, such as interactive audio and video. Those services include payload type identification, sequence numbering, time-stamping and delivery monitoring. RTP itself does not provide any mechanism to ensure the timely delivery of media units nor provides for other quality-of-service guarantees, but rather relies on lower-layer services to do so (i.e., integrated or differentiated services applied at the network layer). RTP carries data with real-time properties, whereas RTCP monitors the quality of service of each participant and conveys useful information to all participating entities. Applications that use the UDP transport protocol have to implement *end-to-end congestion control* [7] otherwise (a) the TCP traffic suffers and (b) the network collapses. RTCP may be used as a feedback channel to provide for congestion notification to CM applications that implement an end-to-end congestion control scheme [8,9]. Examples of RTP/RTCP based rate control algorithms, that are TCP-friendly, are [10-14]. Other rate control protocols, such as RAP [15] and TFRCP [16], that are TCP-friendly and do not rely on RTP/RTCP, have been proposed in the literature as well.

Rate control protocols and adaptation mechanisms require sources capable of adjusting their rate on the fly (such as MPEG-4 [17]) or transcoding filters and video gateways [18,19,20] capable of transcoding a high bit-rate input stream (such as MPEG-2 [21]) to a lower bit-rate adjustable output stream. An alternative to rate control and transcoding mechanisms is the multi-layered coding with multicast transfer scheme [22]. Such a scheme allows the receiver to connect to the appropriate multicast channel(s) according to its connection bandwidth availability.

A rate adaptation scheme needs to be communicated the network state through a feedback mechanism. Most rate adaptation schemes use the current packet loss rate to infer whether the network is congested or not, and the rate is either increased or decreased, respectively. The increase function is either additive increase [23] or multiplicative increase [16], whereas the decrease function is multiplicative decrease [8,9,15] or the rate may be set to a value calculated by an equation-based formula [24] which estimates the equivalent TCP throughput [10-14,16]. The various feedback mechanisms are either RTCP-based [8-14] or ACK-based [15,16]. In the first case the feedback delay is at least 5 sec (± 1.5 randomly determined), whereas in the second case it is equal to one round trip time. A congestion control scheme that uses the history of the packet losses has been proposed as well [25].

In this paper a rate adaptation scheme is introduced for an environment supporting CM flows. As discussed briefly below - as well as throughout this paper – the scheme will attempt to take into consideration certain peculiarities associated with a CM-supporting networking environment.

Users may be connected to the network using links of limited and fixed bandwidth (e.g., dial-up users, ADSL users), or practically unlimited bandwidth. The bit-rate of a CM flow is assumed to take any value within a specific range, where a minimum rate is considered in order to ensure a minimum perceptual quality, whereas a maximum

rate represents the upper limit of the source encoder capability or is imposed to prevent unnecessarily high consumption of bandwidth and /or prevent the suppression of other non CM flows, e.g., TCP flows. The (limited) access bandwidth and the minimum and maximum source encoding rates all impose restrictions on the behavior of the source rate adaptation scheme and should be taken into consideration. For instance, probing for more bandwidth availability when the (static) access bandwidth limit is reached should be avoided by an effective rate adaptation scheme.

CM flow initiations and terminations could result in large and lasting bandwidth availability changes. Such type of changes should be distinguished from (small and short-term) typical bandwidth of fluctuations and should trigger a rapid response from the rate adaptable CM flows to avoid excessive losses or bandwidth under-utilization. Because of the nature of the CM applications, decreasing the rate upon congestion by a factor of 2 (as in TCP) should be avoided, as the perceptual QoS would suffer substantially. Fast convergence to a fair bandwidth allocation is desirable after a flow initiation or termination occurs.

The rate adaptation scheme proposed in this paper is designed for a networking environment supporting CM flows and is capable of detecting and responding effectively to static and long-term bandwidth availability. The CM flows are supposed not to be aware of any access bandwidth limitations or the number of co-existing CM flows or the overall network load. The developed scheme detects any access bandwidth limitations and locks the rate of the flow to an appropriate rate, as well as detects lasting load changes (due to the initiation or termination of CM flows) and locks the rate of the CM flows to an appropriate rate. Short-term bandwidth changes are not expected to be present in a (purely) CM supporting networking environment due to the nature of the Adaptable Constant Bit Rate (A-CBR) CM applications considered here. While an extension of this work will consider an environment where A-CBR CM flows coexist with other traffic (such as TCP) which introduces short-term bandwidth changes, the effort there is expected to be focused on the study of the impact of such bursty traffic on the proposed rate adaptation algorithm and the introduction of the adjustments to improve its effectiveness, rather than the redesign of the algorithm to become responsive to such short term bandwidth fluctuations. To follow the latter fluctuations may be impossible due to the granularity and the hysterisis of the encoding process and the rate adaptation time lag, as well as be of negligible impact on the perceived QoS.

In this paper, a multi-state congestion control scheme for rate adaptable unicast CM flows is presented. This scheme is suitable for an environment in which the network bandwidth availability information is communicated to the source only indirectly through a notification of the packet losses that the specific source suffers over a time interval. This environment is present, for example, when CM flows are transmitted by using RTP/UDP/IP protocols. The proposed scheme aims to provide a finer rate adaptation decision space for the source by introducing a number of flow states and then basing the rate adaptation decision on the current source state in addition to the current packet loss feedback information. The introduction of states allows for an effective “summarization” of the recent bandwidth availability and adaptation history. The resulting multi-state rate adaptation algorithm has a larger decision space compared to traditional schemes, and responds well to a diversity of network situations, such as

those outlined above, while exhibiting good fairness characteristics. The behavior of the proposed scheme is examined in an environment where only A-CBR CM (RTP/UDP/IP) flows are transmitted (i.e., free of TCP flows), as it may be the case under the forthcoming differentiated services.

2. Description of the Algorithm

A networking environment supporting N RTP/UDP/IP CM flows is considered. The RTCP protocol is used as a feedback notification mechanism as specified by RFC 1889 [7]. The source and receiver(s) of each flow exchange RTCP Sender and Receiver Reports (SR and RR, respectively). The sources are assumed to be capable of adapting their encoding and transmission rates on the fly, as it is the case. The source of a flow is not aware of the topology, the links, the available bandwidth and the current number of CM flows transmitted over the network. The only available information is the one carried by the RTCP RR reports. One RR report is sent every T seconds. In this paper T is set to be equal to 5 sec, as specified by the RFC 1889.

A decision module - located at the source - is responsible for deciding on the appropriate transmission rate to which the source should adapt. Rate adaptation decisions are taken at the *adaptation points*. An adaptation point occurs when a RR arrives at the source of the flow or a time-out timer expires indicating that at least one RR report is excessively delayed or lost. The time-out timer is initially set to the value of `timeout_period(i)` – for flow i – which is set to a value $T_0(i)$ greater than T to absorb (in part) the network introduced delay jitter. This counter is reset to $T_0(i)$ each time a RR is received and to $T_1(i)$, $T<T_1(i)<T_0(i)$, each time a timeout occurs. This subsequent reduction in the time-out period will lead to a reduction of the time to the next adaptation point if this is due to a time-out and, thus, will lead to a faster and more effective rate adaptation.

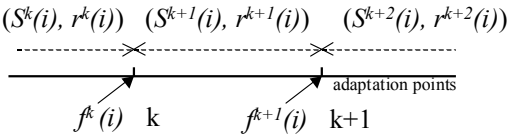

Fig. 1. State and rate transitions.

Let $f^k(i)$ denote the network state feedback associated with the k^{th} adaptation point of flow i. $f^k(i) \equiv \varnothing$ (i.e., no value) if the k^{th} adaptation point is due to a timer expiration and $f^k(i) \equiv p^k(i)$ when it is due to the arrival of a RR; $p^k(i)$ denotes the packet loss rate derived from the RR received at the k^{th} adaptation point. Let $S^k(i)$ and $r^k(i)$ denote the state – to be defined later – and rate of flow i, just before k^{th} adaptation point. At the k^{th} adaptation point a state transition (adaptation) from $S^k(i)$ to state $S^{k+1}(i)$ and rate transition from $r^k(i)$ to $r^{k+1}(i)$ will occur, effective just after the k^{th} adaptation point and remaining effective until just before the $(k+1)^{st}$ (Fig.1). In addition to the received feedback $f^k(i)$, the next state $S^{k+1}(i)$ depends on $S^k(i)$ as well as a number of other

quantities defined at each adaptation point providing useful history information. Such quantities are the recent average packet loss rate $p^k_h(i)$ – defined precisely later -, the number of successive, $I^k_{succ}(i)$, and total, $I^k_{total}(i)$, visits to the current state $S^k(i)$, for certain states, and the binary lock decision function $B_{Lock}(i)$ that returns 1, if rate locking is decided. The next rate $r^{k+1}(i)$ depends on the next state $S^{k+1}(i)$ decided to be visited, the current rate $r^k(i)$, and the feedback $f^k(i)$.

Let $\overline{S}$ denote the state space of the process $\{S^{k+1}(i)\}_{k\geq 0}$; $\overline{S}$= {`just_intro`, `decr_due_nofeedback`, `increase`, `decrease`, `lock`, `rmax`,`rmin`, `fast_increase`, `fast_decrease`, `check_for_bw`, `low_increase`, `low_decrease`, `low_lock`, `low_check_for_bw`}. A brief description of the states and the motivation behind their introduction are presented next.

Upon initiation, a flow enters state `just_intro`, allowing the new source to follow slightly different increase/decrease policy for a given feedback compared to existing flows. This is necessary since the initial rate of a flow is randomly selected within a range and thus different level of up or down adjustments are needed compared to older flows which have already adjusted their rate and need to respond to bandwidth availability changes.

If a rate adaptation point is due to the time-out timer expiration it is considered that at least a feedback is lost or excessively delayed and the `decr_due_nofeedback` state is visited while the rate is properly decreased.

States `rmax` and `rmin` are defined to capture the state of a source that attempted to select a rate beyond the corresponding rate limits. When a flow is in one of these states the source will apply such a control that the flow rate remain in the range $[r_{min}(i), r_{max}(i)]$. Also, visiting the state `rmin` once is an indication of a highly congested network, whereas visiting this state more times is an indication of persisting congestion, probably due to access network limitation.

States `increase` and `decrease` are defined for the normal case in which the flows compete with each other in order to share the available bandwidth. The flow enters state `increase` in order to *increase* its rate, whereas it enters state `decrease` to *decrease* the rate in response to detecting some minor congestion.

State `fast_increase` is for the case that the algorithm estimates that the network is rather seriously under-utilised, possibly due to bandwidth release caused by a CM flow termination(s). The transition to this state is accompanied with a rate *increase* that is considerably greater than that of state `increase`. State `fast_decrease` is for the case that the algorithm detects rather serious network congestion due to, for instance, the initiation of some new CM flow(s) and results in a rate dec*rease* that is more drastic than that of state `decrease`.

The algorithm is capable of detecting situations where the flows oscillate around the fairness point or around its access bandwidth limit and could optionally lock the rate. The state `lock` is defined for this purpose. As long as the flow is in state `lock` the rate is *maintained*. After spending some time in state `lock`, the state `check_for_bw` is visited and bandwidth probing rate increases take place, every time this state is (re)visited.

States low_increase, low_decrease, low_lock, and low_check_for_bw are defined to allow for a milder reaction, which is more appropriate when it is highly possible that the congestion is due to access network limitations, near the $r_{min}(i)$ value.

Under no feedback ($f^k(i) = \emptyset$) - that is the time-out timer expires - the state decr_due_nofeedback is visited from any state and the rate is decreased as described in section 3. If the number of consecutive visits to this state exceeds a threshold (*max_cons_times_in_decrnofeed(i)*) then the state rmin is visited.

2.1 State Transitions under Zero Packet Loss Rate Feedback ($f^k(i)=p^k(i) = 0$)

Under zero-Packet Loss Rate (zero-PLR) feedback the state transitions shown in Table 1 take place under the conditions specified. Such transitions are typically followed by a rate increase, as determined by the state visited and described in section 3. Some of the state transitions are self-justifiable in view of the comments provided for the various states in the previous section.

When a transition to a state is accompanied by a rate increase that results in a rate exceeding $r_{max}(i)$, then the rate is set to $r_{max}(i)$ and an instantaneous transition to state rmax occurs. State rmax may be visited from any state and transitions to rmax are not shown on Table 1 for simplicity.

Since the initial rate of a new flow (entering state just_intro) is randomly selected, a zero-PLR feedback implies that that selection has most likely been very conservative and, thus a transition to state fast_increase (as opposed to increase or low_increase) occurs.

Since state decr_due_no_feedback is visited upon time-out timer expiration (indicating delayed or lost feedbacks) and the rate is decreased, a zero-PLR feedback indicates that the previous rate decreases that occurred (upon visiting this state) have been rather excessive and a transition to state fast_increase occurs to allow for a fast rate "correction" (increase).

Upon zero-PLR feedback the algorithm (process) remains in (makes a transition to) state increase, unless this type of transition has occurred for a number (equal to *go_for_fast_increase(i)*) of consecutive times. When the latter occurs it is inferred that a faster increase rate is needed and a transition to state fast_increase occurs. *go_for_fast_increase(i)* is a function that depends on the current rate $r^k(i)$; the larger $r^k(i)$ the larger its value, implying that a lower rate flow enters state fast_increase sooner than a higher rate one.

In order to maintain some smoothness in the process and avoid frequent rate oscillations the state increase (as opposed to fast_increase) is visited from states fast_decrease upon zero-PLR feedback.

Upon zero-PLR feedback the process moves from state decrease to increase (self-explanatory) unless a decision to lock the rate is taken (see section 4) in which case the state lock is visited.

Upon zero-PLR feedback the process remains in state lock, unless this type of transition has occurred for a number (equal to *go_for_check_bw(i)*) of consecutive times, in which case the state check_for_bw is visited.

Table 1. State transitions under the zero-PLR feedback (except `rmax`).

Current state	Next state	Condition
`just_intro`	`fast_increase`	always
`decr_due_nofeed.`	`fast_increase`	always
`rmax`	`rmax`	always
`rmin`	`increase`	C_4: $I^k_{total}(i) <$ *times_in_rmin(i)*,e.g., 3
	`low_increase`	C_5: $I^k_{total}(i) =$ *times_in_rmin(i)*
`increase`	`increase`	C_3:$I^k_{succ}(i) <$ *go_for_fast_increase(i)*
	`fast_increase`	C_2: $I^k_{succ}(i) =$ *go_for_fast_increase(i)*
`fast_increase`	`fast_increase`	always
`fast_decrease`	`increase`	always
`decrease`	`increase`	C_1:not $B_{Lock}(i)$
	`lock`	not C_1 : $B_{Lock}(i)$
`lock`	`check_for_bw`	C_8: $I^k_{succ}(i) >$ *go_for_check_bw(i)*, e.g., 8
	`lock`	not C_8
`check_for_bw`	`check_for_bw`	C_9: $I^k_{succ}(i) <$ *go_to_unlock(i)*, e.g., 4
	`increase`	not C_9
`low_decrease`	`low_increase`	always
`low_increase`	`increase`	C_6:$I^k_{succ}(i)>$*escape_low_states(i)*,e.g.,20
	`low_increase`	not C_6
`low_lock`	`low_check_for_bw`	C_{10}:$I^k_{succ}(i)>$*go_for_low_check_bw(i)*
	`low_lock`	not C_{10}
`low_check_for_bw`	`low_increase`	C_{11}: $I^k_{succ}(i) >$ *go_to_low_unlock(i)*, e.g., 6
	`low_check_for_bw`	not C_{11}

Upon zero-PLR feedback the process remains in state `check_for_bw`, unless this type of transition has occurred for a number (equal to *go_to_unlock(i)*) of consecutive times, in which case the state `increase` is visited and the flow is unlocked.

Upon zero-PLR feedback the process moves to state `low_increase` or `increase` depending on the total number of visits to `rmin` since the initiation of the flow. If this number is equal to or greater (less) than *times_in_rmin(i)* then state `low_increase` (`increase`) is visited. Visiting `rmin` for a number of times implies that it is likely that the available bandwidth is only slightly greater than $r_{min}(i)$ and that thus a lower rate increase that is associated with state `low_increase` would be more appropriate.

The collection of states with the prefix low form a sub-space (referred to as low-state subspace) which is entered by the process only from state `rmin` under zero-PLR feedback and provided that *times_in_rmin(i)* threshold is exceeded. While in the low-space subspace, the process evolves in a similar manner under zero-PLR feedback as described earlier, with minor decision threshold (parameter) adjustments as captured by the introduced parameters ("history" counters). The only exit point

from this subspace is from state `low_increase` under zero-PLR feedback and provided that this state has been visited for a number (equal to *escape_low_states(i)*) of times, in which case there appears to be room for higher rate increase and the state `increase` is visited.

2.2 State Transition under Nonzero Packet Loss Rate Feedback ($f^k(i)=p^k(i) > 0$).

Under nonzero Packet Loss Rate (nonzero-PLR) feedback the state transitions shown on Table 2 take place under the specified conditions. Such transitions are followed by a rate decrease, as determined by the state visited and described in section 3. Some of the state transitions are self-justifiable in view of the comments provided for the various states in section 1. When a transition to a state is accompanied by a rate decrease that results in a rate below $r_{min}(i)$, then the rate is set to $r_{min}(i)$ and an instantaneous transition to state `rmin` occurs. State `rmin` may be visited from any state and transitions to `rmin` are not shown on Table 2 for simplicity.

It is highly desirable to rapidly detect the introduction of a new CM flow in the network and adapt the flows' rate in order to avoid excessive packet losses and provide bandwidth to the new flow. The initial rate of the new flow may be too high for a loaded network. In this case the flows will experience packets losses due to the congestion introduced by the new flow. In order to distinguish whether the congestion is due to the initiation of a new flow or not, the reported packet loss rate $p^k(i)$ is compared to a recent average packet loss rate. Let $p^k{}_h(i)$ denote this average, and let $diff^k(i)$ denote the difference between $p^k(i)$ and $p^{k-1}{}_h(i)$:

$$diff^k(i) = [p^k(i) - p^{k-1}{}_h(i)]^+$$

If $diff^k(i)$ is greater than $p_{new_flow}(i)$, then it is considered that the current packet losses ($p^k(i)$) are significantly higher than the recent average losses ($p^{k-1}{}_h(i)$), they are attributed to the introduction of a new flow and state `fast_decrease` is visited. Otherwise, it is considered that packet losses occur due to the increases of the CM flows and the state `decrease` is visited. The value used for $p_{new_flow}(i)$ is 0.04. When packet losses occur ($p^k(i) >0$), the recent average packet loss rate $p^k{}_h(i)$ is updated as follows:

$$p^k{}_h(i) = c_h * p^{k-1}{}_h(i) + (1-c_h) * p^k(i), \quad (c_h = 0.875)$$

Upon nonzero-PLR feedback the process moves from states `just_intro`, `decr_due_nofeedback`, `increase`, `decrease`, `fast_increase`, `fast_decrease` and `rmax` to state `decrease` if $diff^k(i)) \le p_{new_flow}(i)$ and to state `fast_decrease` otherwise. When in states `lock` or `check_for_bw` the process maintains its state under nonzero-PLR unless $diff^k(i)) > p_{new_flow}(i)$ in which case the state `fast_decrease` is visited.

Under nonzero-PLR the process moves from state `low_increase` to state `low_increase`, unless a decision to lock the rate is taken ($B^L_{Lock}(i)$ function returns 1) and then the state `low_lock` is visited. If $diff^k(i)) \le p_{new_flow}(i)$ the process remains in state `low_lock` (no rate change) under nonzero-PLR feedback unless this type of transition has occurred for a number (equal to *cancel_low_locking(i)*) of consecutive times, in which was a wrong locking rate inferred and the state

low_decrease is visited (rate unlocking). If $diff^k(i)) > p_{new_flow}(i)$ then the process moves from state low_lock to state low_decrease. Finally, the process moves under nonzero-PLR feedback from state low_check_for_bw to either low_lock if $diff^k(i)) \leq p_{new_flow}(i)$ or the state low_decrease otherwise.

Table 2. State transitions under the nonzero-PLR feedback (except rmin).

Current state	Next state	Condition
just_intro, decr_due_nofeed., increase, rmax, decrease, fast_increase, fast_decrease	decrease	C_{12} : $diff(i) \leq p_{new_flow}(i)$
	fast_decrease	not C_{12}: $diff(i) > p_{new_flow}(i)$
lock, check_for_bw	lock	C_{12}
	fast_decrease	not C_{12}
low_decrease	low_decrease	always
low_increase	low_lock	C_{13}:B^L_{Lock}(i)
	low_decrease	not C_{13}
low_lock	low_lock	C_{12} AND C_{14} ; C_{14}: $I^k_{succ}(i) > cancel_low_locking(i)$)
	low_decrease	not (C_{12} AND C_{14})= (not C_{12}) OR (not C_{14})
low_check_for_bw	low_lock	C_{12} : $diff(i) \leq p_{new_flow}(i)$
	low_decrease	not C_{12}
rmin	rmin	always

3. The Distance Weighted Additive Increase, Loss Rate Based Multiplicative Decrease Scheme

The proposed multi-state rate adaptation algorithm belongs in the class of Additive Increase Multiplicative Decrease (AI-MD) scheme [23]. In accordance with the AI-MD characteristic of the proposed scheme, the new rate $r^{k+1}(i)$ is given in terms of the previous rate $r^k(i)$ by:

$$r^{k+1}(i) = r^k(i) + \alpha(i), \quad \text{for the increase case}$$
$$r^{k+1}(i) = r^k(i) * \beta(i), \quad \text{for the decrease case}$$

It should be noted that the increase step $\alpha(i)$ and the decrease factor $\beta(i)$ are not fixed – as in the classical AI-MD case – but the former depends on the state visited $S^{k+1}(i)$, $r^k(i)$, $r_{max}(i)$ and $r_{min}(i)$ and the latter depends on the state visited $S^{k+1}(i)$, $r_{min}(i)$ and $p^k(i)$. To capture the key fact that the increase step depends on the distance between, $r^k(i)$ and $r_{max}(i)$ and that the decrease factor is shaped by the received packet losses, the proposed scheme is referred to as the *Distance-weighted Additive Increase, Loss rate-based Multiplicative Decrease* (D.AI-L.MD) rate adaptation scheme.

(i). In the case that the visited state is state `increase` or `check_for_bw`, the increase step $\alpha(i)$ has two attributes: the base increase rate $Incr(i)$ in kbps and the rate distance factor $dist_weight(r^k(i))$. That is,

$$\alpha(i) = dist_weight(r^k(i)) * Incr(i),$$
$$dist_weight(r^k(i)) = (r_{max}(i) - r^k(i)) / (r_{max}(i) - r_{min}(i))$$

Note that $dist_weight(r(i))$ expresses the distance of the current rate from $r_{max}(i)$. If $r^k(i) \rightarrow r_{min}(i)$ then $dist_weight(r^k(i)) \rightarrow 1$, whereas if $r^k(i) \rightarrow r_{max}(i)$, then $dist_weight(r^k(i)) \rightarrow 0$. If flows i and j are such that $r_{max}(i) = r_{max}(j)$, $r_{min}(i) = r_{min}(j)$ and $r^k(i) > r^k(j)$, then $dist_weight(r^k(i)) < dist_weight(r^k(j))$. This means that flows with lower rate are increasing at a faster pace than flows with higher rate, therefore the convergence time to fairness is expected to improved.

(ii). If the visited state is state `fast_increase`, the increase step $\alpha(i)$ has also a third attribute denoted by $d(i)$. That is,

$$\alpha(i) = d(i) * dist_weight(r^k(i)) * Incr(i)$$

When state `fast_decrease` is visited from another state, $d(i) = d_{init}(i) = 5$; $d(i)$ is decreased by one at each adaptation point at which the flow remains in state `fast_increase`. The reason for this decrease is that each time a fast increase occurs, the probability of having packet losses in the next time interval is expected to increase also. Thus, the next fast increase would be more conservative. When $d(i)$ reaches zero it is reset to the value $d_{init}(i)$ plus the number of successive visits to state `fast_increase`. The reason is that successive visits to state `fast_increase` indicates that most likely there is available bandwidth for the flow. Since, $dist_weight()$ reduces the increase step $\alpha(i)$ as the rate increases, $d(i)$ is increased in order to compensate for this reduction, after a large number of successive visits to state `fast_increase`.

(iii). If the visited state is state `low_increase` or `low_check_for_bw` then the rate is increased by a constant step. That is,

$$a(i) = incr_for_low(i)$$

A typical value of $incr_for_low(i)$ is 1kbps. Table 3 summarizes the different increase function steps $\alpha(i)$.

Table 3. Increase step function $\alpha(i)$.

Visited State	***Increase step α(i)***
`increase, check_for_bw`	$\alpha(i) = dist_weight(r^k(i)) * Incr(i)$
`fast_increase`	$\alpha(i) = d(i) * dist_weight(r^k(i)) * Incr(i)$
`low_increase,low_check_for_bw`	$\alpha(i) = incr_for_low(i)$

The decrease factor $\beta(i)$ has two attributes. The first attribute is the value $(1-p^k(i))$, which is a common attribute in the decrease factors when the visited states are `decrease`, `fast_decrease`, and `low_decrease`. The rate $r^k(i) * (1-p^k(i))$ is roughly the rate under which no packet loss would occur, provided that packet losses occur at a constant rate $p^k(i)$ over the interval between two consecutive adaptation points. In part, because this in not likely in most cases, packet losses will continue to occur even if the new rate is set to $(1-p^k(i)) * r^k(i)$. Therefore, a second attribute denoted

by $decr_parameter(i)$ is introduced to intensify the rate decrease. The value of $decr_parameter(i)$ depends on the state visited, as shown below.

(i). If the visited state is `decrease`, $decr_parameter(i)$ is less than 1; a typical value used is 0.98 but other values may be more appropriate depending on the environment.

(ii). If the visited state is `low_decrease`, $decr_parameter(i)$ is set to 1. The reason is that since the rate in states `low_decrease` and `low_increase` is near $r_{min}(i)$, further rate decrease is likely to reduce the rate to $r_{min}(i)$ which is not desirable.

(iii). If the visited state is `fast_decrease`, then it is likely that a new flow is initiated, as indicated earlier. In this case, the parameter $threshold_{decr}(i)$ with a value less than one and a typical value 0.9 is used:

(a) if $p^k(i) < 1\text{-}threshold_{decr}(i)$, then $decr_parameter(i)$ is set to $threshold_{decr}(i)$. Multiplicative reduction by $threshold_{decr}(i)*(1-p^k(i))$ is a considerable reduction. The reason for this drastic decrease is that first, it reduces the probability of packet losses during the next interval and second, enables a new flow (which is likely to exist) to compete for more bandwidth. Otherwise, the new flow's rate will be kept around its initial rate.

(b) if $p^k(i) \geq 1\text{-}threshold_{decr}(i)$, the factor $\beta(i)$ is set to $(1-p^k(i))$.

In the case the visited state is state `decr_due_nofeedback`, the factor used is $param_{nofeed}(i)$,with typical value of 0.75. Table 4 summarizes the different values of the multiplicative decrease factor.

Table 4. Multiplicative factor $\beta(i)$.

Visited State	*Multiplicative factor $\beta(i)$*
`decrease`	$\beta(i) = decr_param(i)*(1-p^k(i))$; $decr_param(i) = 0.98$
`low_decrease`	$\beta(i) = (1-p^k(i))$
`fast_decrease`	if $(1-p^k(i)) > threshold_{decr}(i)$ {$\beta(i) = threshold_{decr}(i)*(1-p^k(i))$} else {$\beta(i) = (1-p^k(i))$}
`decr_due_nofeed.`	$\beta(i) = param_{nofeed}(i)$

4. Rate Locking/Unlocking Procedure

The D.AI-L.MD scheme ensures that each flow converges to the fairness point, since it is an AI-MD scheme [23]. When a fair rate or the access bandwidth capacity (ceiling) is reached, an effective rate adaptation scheme should try to (temporarily) lock the flow's rate. This locking should not be permanent but should be reconsidered periodically by employing a proper bandwidth probing mechanism.

To lock the rate, a good estimation mechanism of the current fair rate (or the effective access bandwidth ceiling) should be employed. Since the flow's rate follows a pattern of increases and decreases, a good estimate of the locking rate should be based on recent rates (as opposed to the current rate only), which should be properly

weighted. Let *local_min(i)* denote the rate just before the most recent transition from state `decrease` to state `increase` and let

$sm_local_min^{m+1}(i) = f_2 * sm_local_min^{m}(i) + (1-f_2) * local_min(i)$, where $f_2=0.9$

denote the smoothed running average of the *local_min(i)* rates. If the current rate is above the fairness point it will be decreasing (non-monotonically in general) to reach the fairness point, as the AI-MD characteristic of the algorithm guarantees; during this period both *local_min(i)* and *sm_local_min(i)* will be monotonically non-increasing. Similarly, if the current rate is below the fairness point, it will be increasing (non-monotonically in general) and during this period *local_min(i)* and *sm_local_min(i)* will be monotonically non-decreasing. When the rate reaches the fair rate level (fairness point), both the current rate and *local_min(i)* will oscillate around it. After some time period – which depends on the value of f_2 – *sm_local_min(i)* will reach the current *local_min(i)* and this marks the time instant when the rate is locked ($B_{Lock}(i)$ returns 1). The greater the value of f_2 the more accurate the locking rate is expected to be (that is, closer to the fairness point) and the longer time it would take to lock. The locking rate is not the one in effect at the locking time instant but rather it is a smoothed rate based on recent actual rates (and not the *local_min(i)* rates), given by

$$aver_rate_recent^{k+1}(i) = f_r * aver_rate_recent^{k}(i) + (1-f_r) * r^k(i) \qquad (1)$$

with a selected value for $f_r = 0.67$. It should be noted that locking the rate near the fairness point is possible for flows with unlimited access bandwidth or flows whose access bandwidth is higher than the fairness point. Flows whose access bandwidth is below the fairness point are locked near their ceiling rate.

As mentioned earlier, it may take a relatively long time to identify the locking time instant if the procedure mentioned above is followed. To reach a rate locking decision faster for flows whose ceiling is close to $r_{min}(i)$ (such as for dial-up users) and avoid harming rate oscillations and packet losses, a different rate locking strategy is followed for such flows. The rate is locked ($B^L_{Lock}(i)$ returns 1) to the *aver_rate_recent(i)* rate (as in (1)) as soon as more than a small number (equal to `go_for_low_locking(i)` and typically three) transitions occur from state `low_increase` to `low_decrease`.

While in state `lock` the algorithm moves to state `fast_decrease` and, thus, unlocks under a high nonzero-PLR feedback ($diff^k(i) > p_{new_flow}(i)$). After spending some time in state `lock`, state `check_for_bw` is visited, after zero-PLR feedback, from which the algorithm may unlock the rate or eventually return to state `lock`.

Since the procedure for visiting state `low_lock` is not as reliable as the one mentioned earlier and the locking rate may be incorrect, a procedure to unlock the rate is introduced in an earlier section 2.2. It is reminded that from state `low_lock` the state `low_decrease` is visited (unlocking) either if packet losses occur during the first `cancel_low_locking(i)`) intervals (typically two) after the locking decision or high nonzero-PLR feedback is received.

5. Simulations Results

The network simulator (ns) [26] – with some developed RTP/RTCP support enhancements – and the topology shown in Figure 2 are employed in the simulations. Nine sources and receivers are defined. Each source s_i transmits an A-CBR flow i over RTP/RTCP to its associated receiver r_i. Receivers r_1 and r_2 are connected at 64 kbps - simulating dialup users connected at this rate; receivers r_3 and r_4 are connected at 0.5 Mbps - simulating ADSL users connected at this rate; receivers r_5 to r_9 are connected at 1.5 Mbps, simulating LAN or ADSL users connected at this rate. Flows 1-4 are access network limited, whereas flows 5-9 are only backbone network limited. The backbone links are full duplex, and their bandwidth capacity is 4 Mbps. The available memory sizes is equal to 25 packets for the interfaces that connect the routers and 2 packets size for the interfaces that connect sources and receivers to routers (1 and 4). The routers are drop-tail routers. The RTCP Sender and Receiver Reports (SR, RR) are sent every five (5) seconds. No TCP flows are assumed to be present to demonstrate the behavior of the proposed multi-state rate adaptation algorithm in an environment it is primarily designed for.

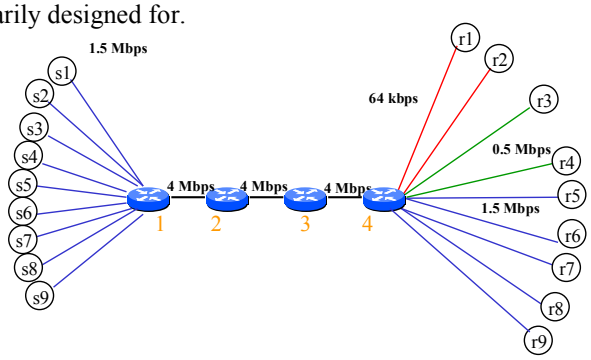

Fig. 2. Network topology and access configuration.

In all simulation configurations, the flows are initiated and terminated according to Table 5. Flow 6 is terminated at the 1200th second, whereas flow 9 is initiated at the 1800th second to demonstrate the response of the simulated schemes to flow initiation/termination. The RTP packet size is common for all configurations and flows and set equal to 1000 bytes. The simulation duration is 2000 seconds. The following parameters are used: $r_{min}(i)$=56 kbps, $r_{max}(i)$=1.2 Mbps, $Incr(i)$ = 30 kbps for all flows i, assuming that the sources are not aware of any access capacity limitation.

Table 5. Start/stop times of flows and rates.

flow	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
start time in sec.	0	1.33	2.17	3.22	4.5	5.17	6.3	7.1	1800
end time in sec.	2000	2000	2000	2000	2000	1200	2000	2000	2000
initial rate (kbps)	56	128	440	550	400	600	1,180	1.200	600

Two different schemes are simulated and compared in order to demonstrate the performance of the proposed D.AI-L.MD scheme. The first scheme is the classic AI-MD

scheme with constant increase step of 30 kbps and decrease factor of 0.97. This factor is close to the corresponding factor for an average packet loss rate of 1% of the D.AI-L.MD scheme (0.98*(1-0.01)). A decrease factor less than this value, e.g., 0.85 or 0.5, leads to a more aggressive decrease, which may be appropriate for data but not for CM flows. Figure 3a illustrates the adaptation behavior of the AI-MD scheme.

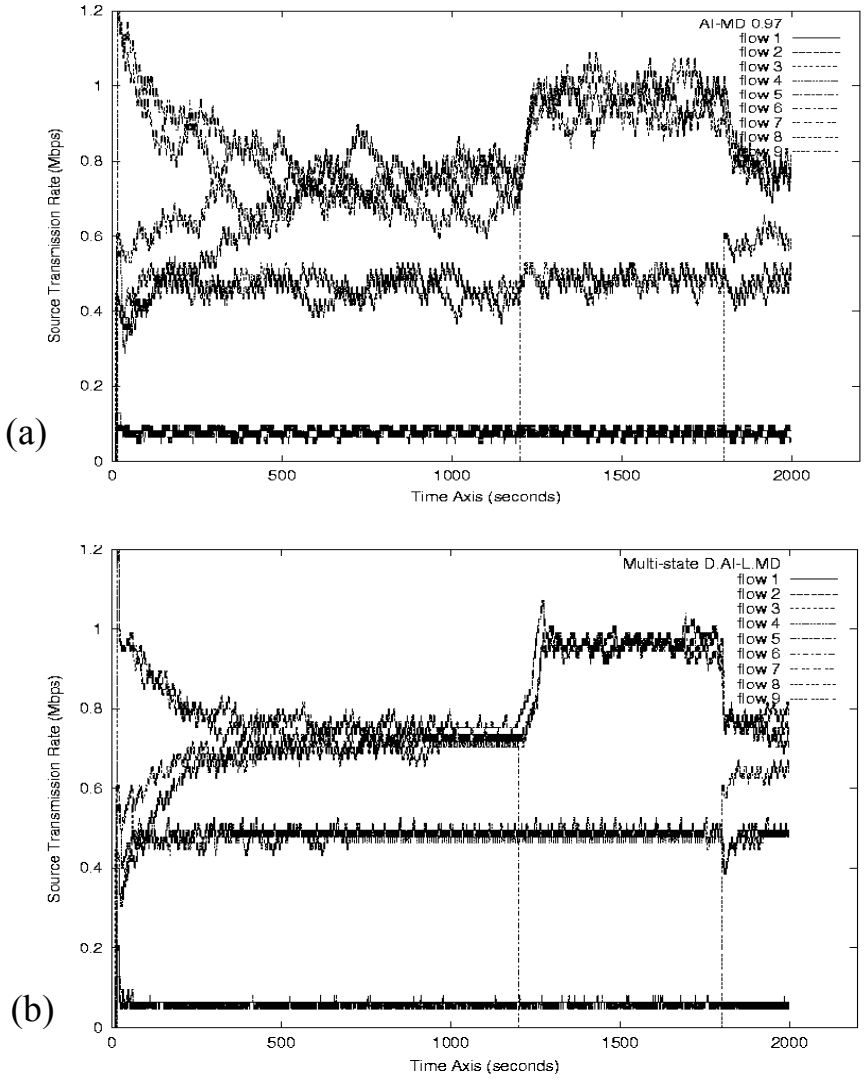

Fig. 3. Flows adaptation in AI-MD and D.AI-L.MD schemes.

The second scheme is the proposed multi-state rate adaptation algorithm (D.AI-L.MD). The following parameters of the D.AI-L.MD scheme are used: $d_{init}(i)=5$,

$param_{nofeed}(i)=0.75$, $decr_param(i)=0.98$, $threshold_{decr}(i)=0.9$, $max_cons_times_in_decrnofeed(i)=3$, $incr_for_low(i)=1$ kbps, $p_{new_flow}(i)$ = 0.06 for flows 1-2, $p_{new_flow}(i)$ = 0.04 for flows 3-9, $times_in_rmin(i)=3$, $go_for_check_bw(i)=8$, $go_to_unclock(i)=4$, $go_for_low_check_bw(i)=10$, $cancel_low_locking(i)=2$, $go_to_low_unlock(i)=6$, $escape_low_states(i)=20$. For the function *go_for_fast_increase(i)* = $fi_min(i)+(1-dist_weight(r^k(i))*(fi_max(i)-fi_min(i))$, the values of 2 and 12 are used for parameters $fi_min(i)$ and $fi_max(i)$, respectively. Figure 3b illustrates the adaptation behavior of the D.AI-L.MD scheme.

In view of the perceptual QoS requirement of CM flows it is highly desirable that the rate adaptations are smooth and as infrequent as possible. Oscillatory rate adaptation behavior leads to highly fluctuating perceived quality that may not be tolerated by the end users. The source also benefits from the fewer and smoother adaptations since it is less resource demanding. The congestion scheme that utilises the AI-MD scheme presents the largest deviation of the current rates from their running long-term rate average (Figure 3a), compared to proposed D.AI-L.MD scheme (Figure 3b). The latter scheme presents smoothed long-term oscillatory behavior. The AI-MD scheme induces a greater number of adaptations than that induced by the D.AI-L.MD scheme (Table 6).

Table 6. Occurred adaptations in AI-MD and D.AI-L.MD schemes.

Flows	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Total**
AI-MD	397	396	397	396	396	236	396	396	39	3049
D.AI-L.MD	218	197	180	229	371	202	361	356	30	2144

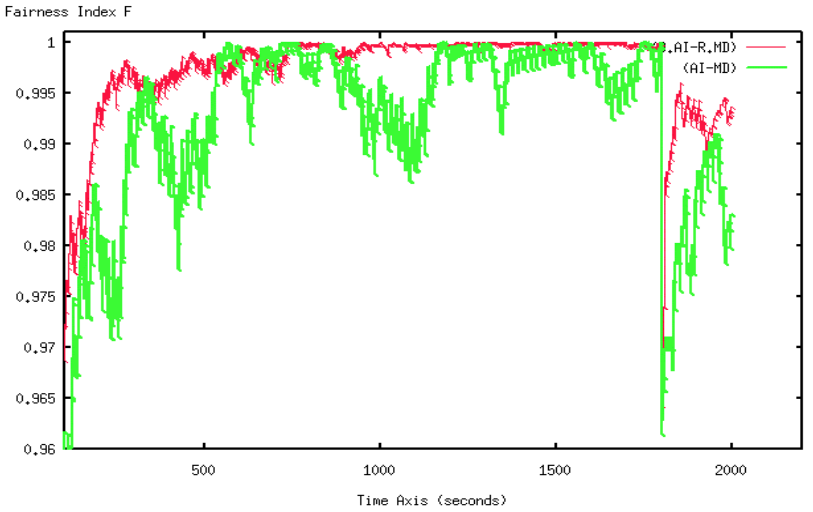

Fig. 4. Fairness index F in AI-MD and D.AI-L.MD schemes.

Fairness and convergence to fairness are requirements set when multiple flows (users) share multiple resources [23]. The fairness index F, presented in [23], is used to

measure the fairness among the A-CBR CM flows. This metric is not meaningful for the limited-access bandwidth flows, i.e., the flows 1-4, and thus, the fairness index F calculation is based on flows 5-9: $F^k = (r^k(5)+ r^k(6)+ r^k(7)+ r^k(8)+ r^k(9))^2 / 5*(r^k(5)^2 + r^k(6)^2 + r^k(7)^2 + r^k(8)^{2+} r^k(9)^2)$.

In Figure 4, the line y=1 indicates the optimal fairness. The time required to reach this line and the achieved level are measures of the convergence to fairness and fairness. As illustrated in Figure 4, the D.AI-L.MD scheme requires less time to converge to the fairness point, as well as presents smaller deviation from it, which enables a more accurate locking.

Table 7. Mean long-term packet loss rate (%).

Flows	AI-MD	D.AI-L.MD
1	16.522	1.657
2	16.857	1.935
3	1.156	0.483
4	1.215	0.562
5	0.711	0.275
6	0.730	0.282
7	0.726	0.272
8	0.757	0.301
9	2.00	0.485

Table 8. Mean conditional packet loss rate (%).

Flows	AI-MD	D.AI-L.MD
1	16.609	4.570
2	16.389	3.877
3	1.674	0.933
4	1.587	0,922
5	1.288	0.894
6	1.266	0.687
7	1.215	0.690
8	1.256	0.758
9	2.005	0.936

Table 9. Total received Kbytes.

Flows	AI-MD	D.AI-L.MD
1	15,700	15,250
2	15,698	15,197
3	115,710	119,679
4	114,442	118,792
5	189,403	192,803
6	97,801	102,036
7	205,214	200,336
8	213,642	204,078
9	14,405	15,767
Total	982,015	983,938

Table 7 shows the *mean long-term packet loss rate*, that is the total number of lost packets over the total number of transmitted packets during the simulation time. Table 8 shows the *mean conditional packet loss rate*, that is the mean value of the reported packet loss rate averaged over the intervals during which nonzero losses occurred. The mean conditional packet loss rate shows the average level of losses when they occur. The D.AI-L.MD scheme presents significantly lower loss rates than the AI-MD scheme, especially for flows 1 and 2, which transmit at a low rate due to the access network limitations.

Statistics concerning the bytes sent and received are gathered. Table 9 presents the total received Kbytes during the simulation period. Figure 5 illustrates the throughput, that is the received RTP packets, over the time window of 1000-1500 seconds. The throughput during this time interval is typical for the overall simulation time. As observed from Table 9 and Figure 5, the D.AI-L.MD scheme is slightly better than the AI-MD scheme in terms of throughput.

As far as the time to recover the bandwidth released by a terminated flow (responsiveness) is concerned, the AI-MD scheme reaches the bandwidth capacity faster- due to the constant increase step – than the D.AI-L.MD scheme (Figure 6). The D.AI-L.MD scheme seizes the bandwidth at a slower pace than the AI-MD scheme, because of the distance-weight additive increase feature of the scheme. The responsiveness in the D.AI-L.MD scheme depends on the distance of the flow's rate from the $r_{max}(i)$. The larger the distance (relatively low rate flows), the faster the response. The less the

distance (relatively high rate flow, as in the simulation case), the slower the response. Despite the slower response to the bandwidth release for relatively high rate flows, the D.AI-L.MD scheme is better than the AI-MD in terms of fairness, convergence to fairness, packet losses, throughput and oscillatory behavior.

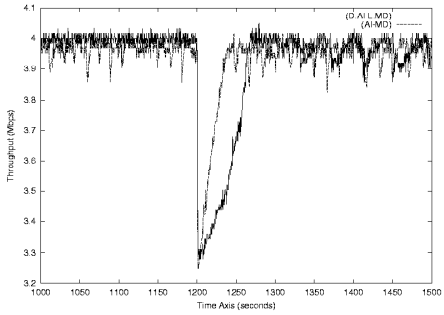

Fig. 5. Throughput of the schemes.

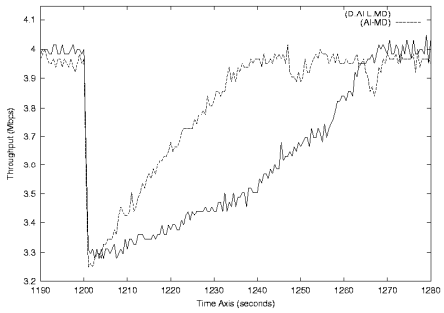

Fig. 6. Response to bandwidth availability.

6. Conclusions

In this paper, a multi-state congestion control algorithm for rate adaptable unicast CM flows in a TCP-free environment has been presented. This scheme (a) provides a finer rate adaptation decision space compared to traditional schemes, (b) enables rate locking, and (c) responds well to a diversity of network situations, such as limited access bandwidth and initiation/termination of CM flows. The proposed D.AI-L.MD scheme presents significantly less packet loss rates and oscillatory behavior compared to the AI-MD scheme. It also presents faster and closer convergence to the fairness point. The AI-MD scheme presents better responsiveness to bandwidth availability in comparison to the D.AI-L.MD scheme. Further work is in progress to further investigate the performance of the proposed scheme and extend it for environments in which both TCP/IP and A-CBR CM flows co-exist in the access and/or core network.

References

[1] Floyd, S., "TCP and Explicit Congestion Notification". ACM Computer Communication Review, V. 24 N. 5, October 1994, p. 10-23.

[2] RFC 2481. Ramakrishnan, K.K., and Floyd, S., "A Proposal to add Explicit Congestion Notification (ECN) to IP". January 1999.

[3] V. Jacobson, "Congestion avoidance control". In Proceedings of the SIGCOMM '88 Conference on Communications Architectures and Protocols (1988).

[4] RFC-2001. “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms”.
[5] RTP Frequently Asked Questions: http://www.cs.columbia.edu/~hgs/rtp/faq.html .
[6] RFC 1889.RTP: A Transport Protocol for Real-Time Applications.
[7] Floyd, S., and Fall, K., “Promoting the Use of End-to-End Congestion Control in the Internet”. IEEE/ACM Transactions on Networking. August 1999.
[8] I. Busse, B. Defner, H. Schulzrinne, “Dynamic QoS Control of Multimedia Application based on RTP”. May 1995.
[9] J. Bolot,T. Turletti, “Experience with Rate Control Mechanisms for Packet Video in the Internet”. ACM SIGCOMM Computer Communication Review, Vol. 28, No 1, pp. 4-15, Jan. 1998.
[10] D. Sisalem, F. Emanuel, H. Schulzrinne, “The Loss-Delay Based Adjustment Algorithm: A TCP-Friendly Adaptation Scheme”. 1998.
[11] D. Sisalem and A. Wolisz, “LDA+: Comparison with RAP, TFRCP”. IEEE International Conference on Multimedia (ICME 2000), July 30 - August 2, 2000, New York.
[12] D. Sisalem and A. Wolisz, “MLDA: A TCP-friendly congestion control framework for heterogenous multicast environments”. Eighth International Workshop on Quality of Service (IWQoS 2000), 5-7 June 2000, Pittsburgh.
[13] D. Sisalem and A. Wolisz, “Constrained TCP-friendly congestion control for multimedia communication” tech. rep., GMD Fokus, Berlin Germany, Feb. 2000.
[14] D. Sisalem, A. Wolisz, “Towards TCP-Friendly Adaptive Multimedia Applications Based on RTP”. Fourth IEEE Symposium on Computers and Communications (ISCC'1999), (Red Sea, Egypt), July 1999.
[15] R. Rejaie, M. Handley, D. Estrin, “An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet”. Proc. INFOCOMM 99, 1999.
[16]J. Padhye, J. Kurose, D. Towsley, R. Koodli, “A Model Based TCP-Friendly Rate Control Protocol”. Proc. IEEE NOSSDAV'99 (Basking Ridge, NJ, June 1999).
[17] ISO/IEC 14496. MPEG-4.
[18] J. Pasquale, G. Polyzos, E. Anderson, and V. Kompella, “Filter propagation in dissemination trees: trading off bandwidth and processing in continuous media networks”. Proc. of NOSSDAV'93.
[19] E. Amirm, S.McCanne, H.Zhang, “An application-level video gateway”. Proc. ACM Multimedia ’95, San Francisco, Nov. 1995.
[20] IST VideoGateway project: http:/www.optibase.com.
[21] ISO/IEC 13818. “MPEG-2 - Generic coding of moving pictures and associated audio information”.
[22] L. Vicisano, J. Crowcroft and L. Rizzo, “TCP-like Congestion Control for Layered Multicast Data Transfer”. Proc. InfoCom'98, San Francisco, March/April 1998.
[23] R. Jain, K. Ramakrishnan, and D.-M. Chiu. “Congestion avoidance in computer networks with a connectionless network layer.” Tech. Rep. DEC-TR-506, Digital Equipment Corporation, Aug. 1987.
[24] J. Padhye, V. Firoio, D. Towsley, J. Kurose, “Modelling TCP Throughput: a Simple Model and its Empirical Validation”. Proceedings of SIGCOMM 98.
[25] T. Kim, S. Lu and V. Bharghavan, “Improving Congestion Control Performance Through Loss Differentiation”. International Conference on Computers and Communications Networks '99, Boston, MA. October 1999.
[26] The network simulator (ns): http://www.isi.edu/nsnam/ns/ns-build.html