Special Issue

Multicast Congestion Control for Active Network Services*

ARTURO AZCORRA
Area de Ingeniería Telemática, Univ. Carlos III de Madrid, 28911 Leganés (Madrid), Spain
azcorra@it.uc3m.es

MARÍA CALDERÓN
Facultad de Informática, Univ. Politécnica de Madrid, 28660 Boadilla del Monte (Madrid), Spain
mcalderon@fi.upm.es

MARIFELI SEDANO
Escuela Universitaria Politécnica, Univ. De Alcalá de Henares, 28871 Alcalá de Henares (Madrid), Spain
marifeli@aut.alcala.es

JOSE IGNACIO MORENO
Area de Ingeniería Telemática, Univ. Carlos III de Madrid, 28911 Leganés (Madrid), Spain
jmoreno@it.uc3m.es

Abstract. The growing interest in multicast applications for the Internet, that will increase with the introduction of active network technology, brings out the need for providing effective congestion control mechanisms. This paper studies the problems and specific requirements for multicast congestion control. From this study, a general mechanism is proposed and its advantages and compliance with the requirements are described. The general mechanism is then applied in detail to the Reliable Multicast Active Network Protocol implementation. We conclude that active networks provide good support for congestion control mechanism and that the proposed approach satisfies the stated requirements. However, further work is needed to optimize the values of the system parameters.

1 INTRODUCTION

Active network technology is based on specific processing of packets at network nodes. There are two main approaches for active networks [1]. The first one is the programmable switch, characterized by maintaining the existing packet/cell format, and providing a standard mechanism to support the downloading of programs at routers. The second one is the capsule approach that goes somewhat further. In the capsule approach it is possible to define packets, called capsules, each of which identifies the code to process it. This approach has two variants depending on how the code is installed at the routers. Under the embedded code variant, the code is sent within the capsule itself. Under the demand load variant, if the code is not already in the router, it is requested to the previous router, i.e., the one from which the capsule has been received.

Active networks seek to address the problem of slow network service evolution by building programmability into the network infrastructure itself, thus allowing many new network services to be introduced much more rapidly [2]. An active node runs one (or more) execution environment(s) that interpret received active packets by means of an associated virtual machine [3]. Active network technology does not, at this moment, address some relevant areas in a satisfactory way. Examples of such areas are resource consumption control, network management, security considerations, and in particular, congestion control.

Congestion control is recognized as a conceptual function of the network layer. However, in the Internet the original Source Quench mechanism at the IP layer has not been used. The situation is that each protocol on top of IP may, or may not, implement some congestion control procedure. Currently, most TCP implementations provide a fair and effective technique, but most other protocols on top of IP (UDP, routing protocols, ...) do not incorporate congestion control at all. With the

* This work has been partly supported by CICYT (the Government Commission of Science and Technology) under project TIC97-0929.

introduction of "unfair" TCP implementations, the increase in nonTCP traffic (e.g. internet telephony), and growing usage of multicast IP (e.g. real audio, mbone), the approach of placing congestion control only in TCP is insufficient. The expected situation for active networks is more severe in that their flexibility means that the variety of applications and services will be wider than in the current Internet. For this reason, it seems clear that an appropriate treatment of congestion control is essential for active network services and applications.

Services and applications that require multicast communication are considered an especially relevant problem to study. On the one hand, availability of multicast applications is growing fast. Some examples that may be cited are Software Distribution, Newspaper/Financial Distribution, Online Auctions, Video/Teleconference, Chats, Interactive Group Games, Whiteboard, Computer Supported Cooperative Work, Application sharing or Distri-buted Interactive Simulation. On the other hand, congestion control solutions for multicast traffic are still in a very preliminary stage, while at the same time IETF has explicitly required that any proposal must incorporate congestion control [4]. It is foreseen that the flexibility introduced by active network technology will accelerate the introduction of multicast applications, thus making congestion control a crucial issue.

What we propose is that congestion control be incorporated within active networks instead of being provided end-to-end as is done on the current Internet. We show that active networks allow the implementation of hop-by-hop congestion control mechanisms that will show a better behavior than end-to-end solutions. This is particu-larly true for the case of multicast applications, in which end-to-end congestion control solutions suffer from different drawbacks. We have designed and formally specified a congestion control mechanism for multicast data over active networks. This mechanism has been applied to the Reliable Multicast Active Network Protocol (RMANP) implementation in order to show its application to a concrete case. An overview of its main functions is presented.

2 MULTICAST CONGESTION CONTROL REQUIREMENTS

Any congestion control mechanism must satisfy general requirements, such as reacting to congestion as soon as possible or minimizing the number of lost packets while reacting to congestion. The characteristics of multicast communication imply some additional specific requirements:

Scalability to large numbers of receivers. Some mechanisms suffer from implosion problems when receivers send back to the source congestion indications.

Selective reaction. In multicast, a single loss in a LAN or router will be perceived as n losses (one at each of the n downstream receivers). It is important that the congestion control distinguishes whether two indications correspond to a single loss or whether they really correspond to two different losses.

Loss-tolerance. A single packet loss, or a set of isolated losses, should not be treated as conventional congestion [5]. Existing studies over Mbone [6] have shown that the probability of each and every packet being lost at some point of the distribution tree is very large for groups of sparse receivers.

Fast response. Because the network multiplies the traffic injected by the source along the distribution tree, the fast response requirement is particularly relevant in multicast traffic.

Heterogeneity. In multicast communications there is diversity in the type and capabilities of receivers, and in the characteristics of the paths (bandwidth and delay) that communicate the source with the receivers. This situation brings up the problem of internal-fairness, in which it is considered unfair to the group that a slow receiver, or a receiver connected through a slow path, slows down throughput to all the receivers.

Multiple data sources. When multicast communication is combined with some degree of reliability, most solutions propose local recovery at other points besides the source. This implies that retransmitting systems should also be congestion controlled, in addition to the source.

Reaction time estimation. Most congestion control algorithms need an estimation of the round trip time in order to parameterize different transitions of the protocol entity. An estimation of the time under which the network should react to congestion actions is needed, for example, to distinguish congestion from a deadline or receiver that left the group. The estimation of the Round Trip Time (RTT), either average or to the most distant receiver, is more difficult to obtain in multicast than in the unicast case.

Any congestion control proposal for multicast communication should satisfy the above requirements, in addition to the general ones established for unicast. Under the functional point of view, there are no known differences between the multicast and unicast case, the same basic functions being required:

- How an entity increases throughput to provide the maximum available bandwidth to the application, under no congestion.
- Why an entity decides that there is congestion. This may be detected by its own local information, or be-cause it receives an explicit indication from another entity.
- How an entity communicates to other entities that it has detected congestion.

- What actions does an entity take to recover the congestion situation.
- How an entity decides that the congestion situation has been tackled with.

The hop-by-hop congestion control mechanism that we propose will fulfill these functions at the different entities of the active network, attempting to satisfy the general congestion control requirements as well as the ones specific to multicast communications.

3 CURRENT APPROACHES TO MULTICAST CONGESTION CONTROL

One of the most difficult problems encountered in the design of end-to-end multicast congestion control is scalability. Currently, there are two main approaches to provide acceptable scalability: 1) sender-oriented algorithms, in which the source controls the rate of injected traffic based on indications sent by the receivers. 2) receiver-oriented algorithms, in which the receivers control the rate at which they accept data.

Among the sender-oriented algorithms the many different proposals may be roughly divided in three types. The first type consists of selecting a set of representative receivers (e.g. [7]) that will send immediate indications, while the non-representative ones will perform probabilistic suppression. This approach will expose different reaction times depending on the number of selected representatives, and their topological location. The second type proposes the construction of a tree, where some receivers act as tree nodes and others as tree leaves. Each node acts as an intermediate retransmission point, serving the requests received from its descendants. Nodes also perform congestion control by monitoring the congestion level of its descendants. In [8] this is done using a dynamic congestion window (TCP like) updated with received ACKs and NACKs from the descendants. A summary of the congestion reports is aggregated along the tree in order to inform the sender. The main drawbacks are: 1) the delay from the congestion detection instant at the SA to the instant at which the sender reduces its rate may be too long. 2) the construction of a tree of end-systems will not in general make advantage of the topol-º ogy of the physical distribution tree. The third type consists on a collection of independent techniques to reduce the implosion of congestion notifications. A fairly interesting one is [5] based on that the receiver performs probabilistic suppression of the indications sent towards the source, in order to avoid implosion. The drawback of this approach is that congestion indication is delayed, increasing the reaction time to cope with the congestion.

Receiver-oriented algorithms organize transmitted data into layers, associating one multicast address to each layer. Every receiver may control the accepted rate by subscribing to more or less multicast addresses. The receiver will monitor the loss rate. When losses are detected, the receiver will leave a multicast group, and if there are no losses the receiver will join an additional group. Notice that it is required that all receivers in a congested subtree agree on what group should be abandoned and at what instant. The main drawbacks of this technique are that the reaction time is dependant on the propagation of control information at the multicast routing level, and that it is not possible to control the rate of the source (i.e. the source could be creating congestion in the first transmission hop). Some proposals ([9,10,11]) apply this approach to unreliable data distribution, satisfying the heterogeneity requirement by allowing receivers with different characteristics to get different quality. Other authors [12, 13] apply it to reliable data distribution. In this case, all data must be received and therefore it is only useful for bulk transfers in which receivers with higher throughput will complete the transfer in shorter time.

Work reported on in a recent paper [14] has begun to exploit the advantages of active networks to support congestion control, although restricted to the simpler unicast case. In his article, Faber shows how the throughput of a TCP connection may be improved 18% under bursty traffic conditions. Some limitations of his approach are that it generates one congestion indication for each lost data packet, and that it does not react to losses in the non-active nodes.

4 MULTICAST CONGESTION CONTROL OVER ACTIVE NETWORKS

The congestion control mechanism proposed is designed for an active network formed by a set of active systems interconnected by non-active internetworks. A non-active internetwork may be a direct link, a LAN or a WAN that provides unreliable unicast and multicast capsule delivery. Therefore, the mechanism is designed to work under the situation in which not all the network nodes are active, and it is prepared to react to losses within the connecting internetworks.

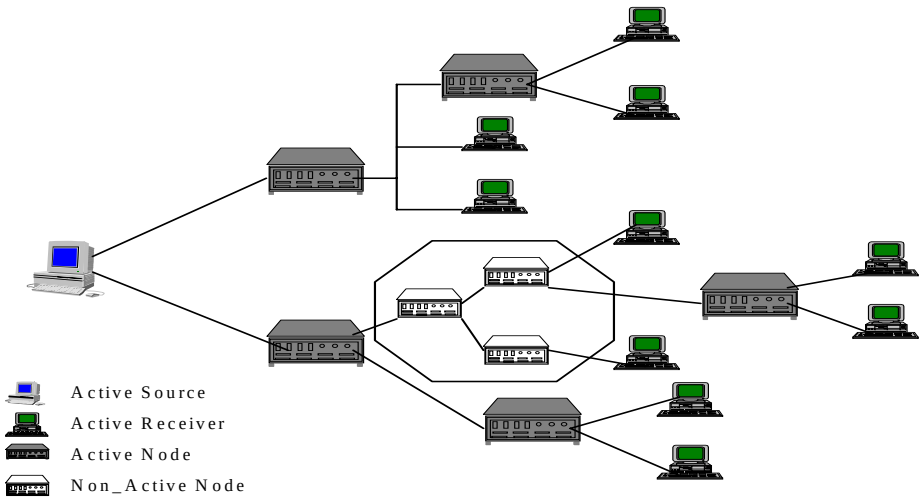

Figure 1. Control Tree Structure

A multicast session is structured as a tree of active nodes formed by the multicast routing algorithm. The tree root is the session active source and the tree leaves are the session active receivers (see Figure 1). Each active node has one upstream interface and one or more downstream interfaces. Through each downstream interface, an active node may reach one or more direct descendants (either active nodes or receivers).

The proposed congestion control works hop-by-hop. An overview of the functions performed by each system under the proposed mechanism follows:

The Source

- Controls the value of the session transmission data rate, stamping each outgoing data capsule its the current value.
- Periodically multicasts the value of the session minimum_rate (fixed by the application).
- Periodically requests through all the tree feedback state information (RTT and highest acknowledged sequence number). The source will receive state information from all its direct descendants and from all active nodes that have receivers as direct descendants.
- When it receives a feedback state information answer, it updates its state information registers. If the highest acknowledged sequence number received increments the global one, then the source will increase its output rate.
- Whenever it receives congestion indications, it applies congestion control actions.
- Whenever it detects that feedback state information is not flowing (severe congestion), it reduces the output rate to the minimum defined value.

Each Active Node

- Forwards capsules downstream (as well as caching them for potential retransmission), in the no-congestion situation.
- When it receives a state information request, it immediately sends a state information report towards the source and also forwards the request to all its direct descendants.
- When it receives a feedback state information answer, it updates its state information registers.
- When it receives a retransmission request, it retransmits the capsules from its cache.
- Whenever it detects severe capsule loss (in its queue or in the preceding internetwork) it notifies the congestion upstream.
- In a congestion situation, it controls the value of the session transmission data rate, stamping each outgoing data capsule with its current value.
- Whenever it receives congestion indications, it applies congestion control actions and forwards the indication upstream.
- When it detects that feedback state information is not flowing (severe congestion), it reduces its output rate to the minimum defined value and indicates this situation towards the source.

- When congested, it retransmits lost capsules from its cache (through the appropriate downstream interface) at a controlled rate, and holds newly-received capsules in a queue for subsequent forwarding.
- When congestion decreases it will forward new capsules (from its cache) at an increasing output rate, until the cache is empty.

Each Receiver

- Controls the reception rate. If it is below the session minimum_rate, it notifies a "leave" and quits the session.
- It controls received capsule sequence numbers. In case of loss, it requests retransmission from a previous node and if loss is severe additionally indicates congestion.
- When it receives a feedback state information request, it immediately sends an answer towards the source.

An important aspect of the proposed mechanism is that congestion indications contain the requested rate. This allows all incoming congestion indications to be filtered (both to forward them, and to locally react to them) with a rate higher or equal to the one currently in use in the node. Therefore, a node will decrease its transmission rate when a more restrictive congestion indication is received. A node will gradually increase its transmission rate while no congestion symptoms are detected.

The advantages of the proposed mechanism over active networks, as related to the requirements from section 2, are:

1. Congestion indications are filtered by the active nodes based on the stamped rate. This avoids multiple reactions to a single congestion instance as well as an implosion of congestion indications.

2. Isolated capsule losses are not interpreted as conventional congestion, and are locally recovered by the closest upstream active node that has cached the lost capsules.

3. Congestion is detected at the closest downstream active node by sequence number control.

4. Explicit congestion indications are used in order to propagate reaction to all nodes, from the congestion point towards the source, as fast as possible.

5. Each active node reacts locally to congestion by reducing its output rate, and sending an explicit congestion indication upstream. Therefore, each node will only need to queue packets received from the instant of its own reaction up to the instant of reaction of its parent active node. In this way, all active nodes try to collaborate to absorb the overflow that occurs from the congestion detection instant up to the congestion reaction instant at the source.

6. Retransmitted data is subject to congestion control both at the source and at intermediate active nodes.

7. The feedback state information requests allows a calculation of the highest RTT, in an integrated manner, while avoiding implosion.

8. The absence of feedback state information allows the detection of congestion even in the case in which congestion itself causes the loss of congestion indications.

In order to make the behavior of the proposed mechanism more concrete it has been applied to provide conges-tion control in the RMANP implementation . In order to give an understanding of the complete system, a brief description of the RMANP protocol is given below (a more detailed description of RMANP may be found in [15]).

5 OVERVIEW OF THE RMANP PROTOCOL

RMANP provides different multicast distribution services over active networks. It provides reliable, time-restricted reliable, and unreliable transfer modes for open, controlled and closed receiver groups. RMANP is essentially a sender-oriented protocol, but receivers are also responsible for requesting data retransmissions (NACKs). Each RMANP session provides a multicast communication between one sender and many receivers.

The main features of RMANP are: ACK fusing.– Consists of the sending of just one ACK from a given active node towards the source of each "n" ACKs received. The new ACK carries the fused information of all "n" ACKs; NACK Filtering.– Is performed at active nodes in order to send just one NACK towards the source per data capsule lost. This is, they remember the data already requested, and when a NACK is received it is forwarded only if it asks for different data; Data caching.– If there is space available, active nodes store capsules in addition to forwarding them across the network. Removal of stored capsules in an active node is triggered when all the direct descendants of the node have confirmed the reception of the capsule, or after a given time has passed; Intermediate sequence control.– Active nodes detect gaps in capsule sequence numbers and they generate retransmission requests accordingly. This mechanism is intended to anticipate retransmission requests that would anyway be generated later on at receiver sites; Local recovery.– When a capsule loss occurs, retransmission will be executed at the nearest active node that has cached the lost capsule in order to bring retransmission points closer to the place where the loss occurs; Retransmissions with restricted scope.–

Active nodes forward a retransmission capsule only on interfaces for which retransmission requests for that capsule were received. This feature reduces bandwidth waste and prevents the use of resources at nodes and receivers that did not have trouble receiving this capsule; Retransmission Filtering.– Active nodes use filtering techniques to prevent multiple retransmissions of the same capsule, if it has been requested in parallel by a given set of receivers or active nodes which can be reached via the same network interface.

6 APPLICATION OF THE PROPOSED MECHANISM TO RMANP

The mechanism is formally specified using *Statecharts* [16], but a thorough presentation is not possible in this article. For this reason we present an overview of the main functions. We will begin by describing the behavior of an active node, to discuss afterwards the functions specific to the source and to the receivers. Finally, the algorithm used to compute an estimation of the Round Trip Time to the most distance receiver is presented.

Congestion Control at Active Nodes

A node implements a cache to attempt to store all forwarded capsules that have not been acknowledged by all its direct descendants. The node implements one input FIFO queue, *new_queue*, associated to the upstream interface. The node implements one FIFO queue, ret_queue_i, associated to each downstream interface. Figure 2 represents these flow structures at the node and the paths followed by data capsules. The usage of these queues will be described when studying the behavior of the node.

Here the node behavior is formalized using extended automata. There is one automaton for the node global state (*new_automaton*), depicted in Figure 3, and one automaton for each downstream interface (*ret_automaton*), depicted in Figure 4. The initial state of the *new_automaton* is *No_congestion*, and the initial state of each *ret_automaton* is *No_retransmission_pending*.

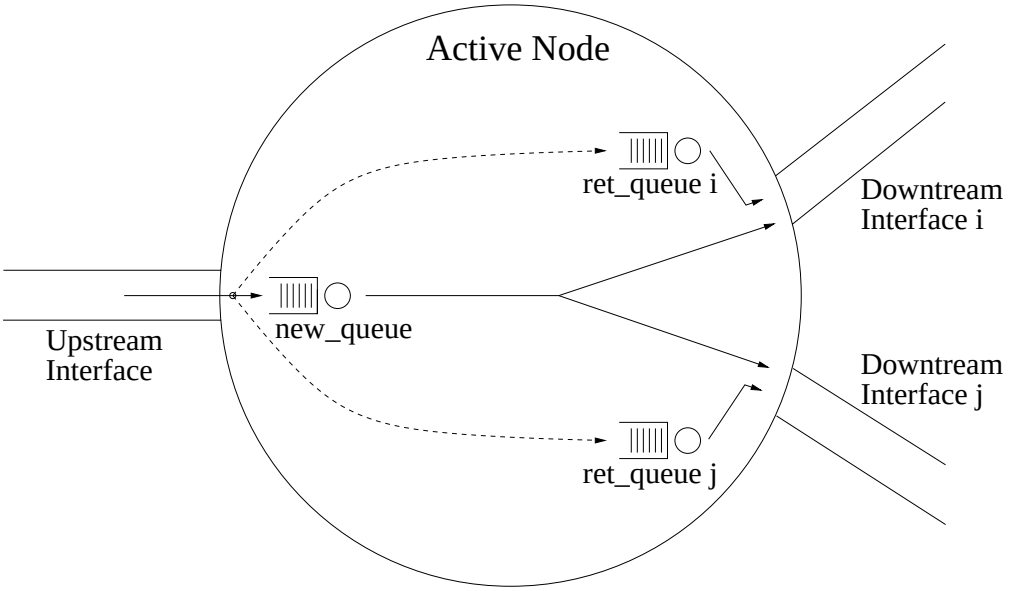

Figure 2. Flow Structure of an Active Node

Because of the design of the transitions, not all the state combinations are reachable. When the *new_automaton* is in the *No_congestion* state or in the *Leaving_congestion* state, all the *ret_automaton* will be in the *No_retransmission_pending* state. Conversely, when any *ret_automaton* is in the *Retransmission_pending* state, the *new_automaton* will be in the *Congestion* state.

In the *No_congestion* state, the node will forward (and try to cache) all received data capsules to all downstream interfaces. Before forwarding, the node will check if its sequence number is one more than the previous capsule (to send a NACK) and it will also record the *Stamped-Rate* of the last forwarded capsule. If the received capsule does not have the expected sequence number, the node will also send a NACK upstream to request retransmission of all lost capsules. If the number of lost capsules is higher than the *G* parameter (and the received capsule is not marked), the node will also send a piggybacked congestion indication upstream, and will mark the forwarded data capsule as having already indicated congestion. The *G* parameter is used to select the threshold of the loss-tolerance requirement. The congestion indication is used to satisfy the fast reaction requirement. The mark of the forwarded data capsule is used to avoid a reaction to congestion downstream of the point where congestion is actually occurring.

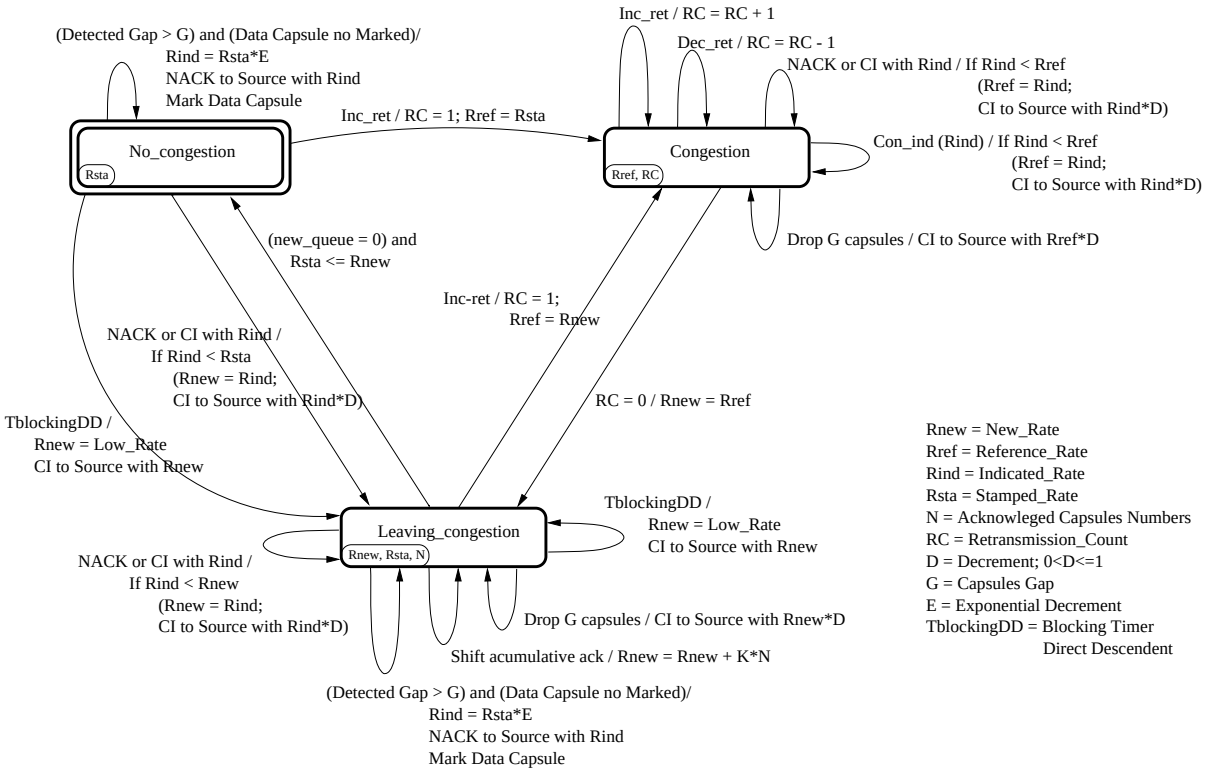

Figure 3. Extended Automaton of the new_automaton

When a node in the *No_congestion* state receives a retransmission request, but no congestion indication, it will pass to the *Congestion* state and the incoming downstream interface will pass to the *Retransmission_pending* state. In this case, the retransmission is made at the current data rate (the one registered at the node) in order to empty the *ret_queue*, but without worsening a **possible** congestion situation.

While in the *Congestion* state the node does not forward **new** data capsules but queues them in the *new_queue*. It only serves retransmissions that are locally cached, or that are received while in this state. In this state, the node classifies a received capsule as **new** if its sequence number is higher than the highest which has previously been processed at the node. Otherwise, the capsule is classified as a **retransmission**, and is queued to the appropriate *ret_queue(s)* based on the standard RMANP retransmission records.

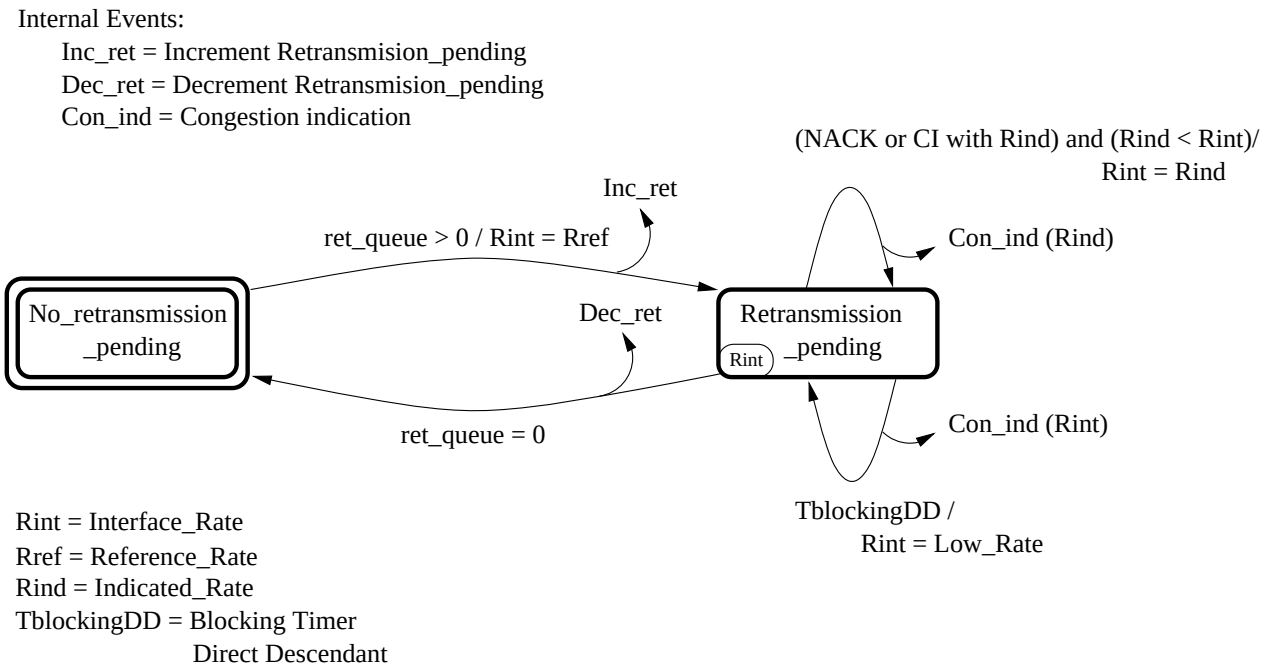

Figure 4. Extended Automaton of the ret_automaton

Once all the *ret_queues* are empty, the node passes to *Leaving_congestion*. In this state, the behavior is similar to the *Congestion* state, but as no retransmissions are pending, the node will proceed to forward the data from the *new_queue* through all the downstream interfaces. In the *Leaving_congestion* state, the rate is increased, beginning at the rate in use when the state was entered, in an additive fashion proportional to the sequence number acknowledged by all the descendants. The objective is to empty the *new_queue* and reach a balance between the arrival and the transmission rates.

Notice that the sporadic loss causes the node to go into the *Congestion* state, but it does not indicate congestion upstream, and it does not reduce its output rate. This means that it will attempt to perform a local fast recovery of the loss and then return to the *No_congestion* state (through the *Leaving_congestion* state). The objective is to recover the sporadic loss but without reacting as it would in a conventional congestion situation.

The congestion indication always contain a rate (*Indicated_Rate*) which informs the receiving node about the highest rate that it should use. The transmission of a congestion indication may be caused by four different situations:

- The node detects a gap (by an incoming non-marked data capsule) in the sequence numbering higher than *G* capsules. Notice that this may only occur in the *No_congestion* or *Leaving_congestion* states. In this case, the *Indicated_Rate* is the stamped rate (the one stamped on the last received capsule, not on the current one) multiplied by an exponential parameter *E* (lesser than 1, possibly 0.5).
- The node is discarding incoming new data capsules due to overflow of the *new_queue*. In this case, the *Indicated_Rate* will be the current rate (depending on the state, it is the *Reference_Rate*, or the N*ew_Rate*) multiplied by an adjusting parameter *D* (lesser than or equal to 1, possibly 0.95).
- The node receives a congestion indication with an *Indicated_Rate* below the current rate (depending on the state, it is the *Stamped_Rate*, the *Reference_Rate*, the *New_Rate, or the Interface_Rate*). In this case, the forwarded *Indicated_Rate* will be the received one multiplied by the adjusting parameter *D*.
- One of the $Tblocking_{DD}$ timers elapses. Each active node maintains one $Tblocking_{DD}$ timer for each of its direct descendants. Each timer is reset when the node receives feedback state information from the associated direct descendant. If one of these timer elapses, the active node will decrease its current rate to the *minimum_rate*, and this is the value included in the congestion indication sent. The reason for this reaction is that the absence of feedback state information is interpreted as severe congestion.

When a node receives a congestion indication it will be ignored if the Indicated_Rate is below the current rate (see the previous point). Otherwise, the node will fix its current rate (*Interface_Rate* of the incoming interface or

New_Rate if in the *Leaving_congestion* state) to the indicated rate.

Congestion Control at Receivers

Receivers do not have descendants, and therefore do not incorporate any of the node functions associated to downstream interfaces. However, an RMANP receiver controls (by time-out) if a retransmission request has been served in order to resend it, and also controls the number of successive retransmission requests (nack_c) issued for a given capsule. This introduces another situation in which a congestion indication is sent: a receiver will send a piggybacked congestion indication for each successive retransmission request sent. The *Indicated_Rate* will be calculated based on the *Stamped_Rate* of the last capsule received in sequence before the first loss, using the expression: $Indicated\_Rate = Stamped\_Rate * E^{nack\_c}$.

The receivers also control their sustained average received rate, in order to satisfy the heterogeneity requirement. The receiver self-imposes an internal fairness policy by leaving the session if the measured rate is below the session minimum_rate.

Congestion Control at the Source

The source does not have an ancestor, and therefore it does not incorporate any of the node functions associated to the upstream interface. The source will exert backpressure on the application to balance the Data_Request primitive rate and the current output rate.

Under very severe congestion, the congestion indications will be lost. This situation will by detected because feedback state information is not received from one, or more, of its direct descendants (like active nodes do). Its reaction will be to reduce its output rate to the *minimum_rate* parameter (fixed by the application).

Estimation of Round Trip Time

Multicast protocols for large groups need an estimation of the RTT in order to distinguish between a spurious congestion situation, and a permanent situation (receiver leave, down link, ...). The removal of dead receivers (or subtrees) for “long” silences is already included in the RMANP protocol, although the calculation of the RTT was not performed.

The proposed mechanism to calculate the RTT is based on cumulative hop-by-hop requests/responses of an estimation of the RTT. This is done using the feedback procedure used to detect severe congestion. The source periodically requests feedback by setting a flag in data capsules. When a node receives such a request, it will respond with an ACK capsule containing (in addition to the current highest sequence number), the current estimation of the node RTT. As the data capsule is also forwarded downstream, the node will later receive responses from its descendants. Each node (and the source) will add the response time of each descendant to its incoming indication of RTT, and will store the highest resulting value. Receivers will always send their responses with RTT set to 0.

Notice that in order to associate requests and their corresponding responses in order to calculate the response time, requests and responses carry sequence numbers.

7 CONCLUSION AND FUTURE WORK

It is considered essential to provide robust and fair congestion control mechanisms for active network services, to avoid the problems that are foreseen in the Internet. This is particularly true for the case of multicast applications and services, because of the multiplication of injected traffic by the network itself.

Active network technology has several advantages in the implementation of multicast congestion control mechanisms. It has been show how these advantages can be exploited by presenting a congestion control mechanism for multicast traffic over active networks that does not suffer from the drawbacks found with end-to-end approaches. The mechanism has been applied to the RMANP implementation, showing the feasibility of its detailed design. The mechanism works in the presence of non-active internetworks and is resilient to severe congestion that causes loss of all congestion indications.

Further work is required to provide appropriate values for the different configuration parameters. Computer simulations would be useful to predict the system behavior under different circumstances in order to tune the proposed mechanism.

Finally, this mechanism has been designed for a source based tree, overlaid over the routing tree set up by the conventional multicast routing algorithm used. It would be more convenient for group communications that the congestion control used instead an overlaid shared acyclic graph (e.g. as is used by RSVP). The modifications required on the mechanism for this improvement appear to be minor, but thourough research and testing would be required before reaching a final result.

8 REFERENCES

[1] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall and G.J. Minden. A Survey of Active network Research. IEEE Comunications Magazine, pp. 80-86, January 1997.

[2] D. Wetherall, U. Legedza and J. Guttag. Introducing New Internet Services: Why and How. IEEE Network, Special Issue: Active and Programmable Networks, 12(3):12-19, May/June 1998.

[3] K. Calvert, ed. Architectural Framework for Active Networks Draft. AN Architecture Working Group, July 1998.

[4] A. Mankin, A. Romanow, S. Bradner and V. Paxson. IETF Criteria for Evaluating Reliable Multi-cast Transport and Application Protocols. RFC 2357, June 1998.

[5] C T. Montgomery. A Loss Tolerant Rate Controller for Reliable Multicast. Technical Report: NASA-IVV-97-011, August 1997.

[6] Yajnik, J. Kurose and D. Towsley. Packet Loss Correlation in the Mbone Multicast Network. IEEE Global Internet Conference, December 1996.

[7] D. DeLucia and K. Obraczka. A Multicast Congestion Control Mechanism for Reliable Multicast. IEEE ISCC'98, Athens, Greece, June-July 1998.

[8] I. Rhee, N. Ballaguru and G. N. Rouskas. MTCP: Scalable TCP-like Congestion Control for Reli-able Multicast. Technical Report TR-98-01, Department of Computer Science, North Carolina State University, January 1998.

[9] S. McCanne, V. Jacobson and M. Vetterli. Receiver-driven Layered Multicast. Proc. of SIGCOMM'96, pp. 117-130, August 1996.

[10] L. Wu, R. Sharma and B. Smith. Thin Streams: An Architecture for Multicasting Layered Video. Proc. of NOSSDAV '97, 1997.

[11] T. Turletti, S. Parisis and J. Bolot. Experiments with a Layered Transmission Scheme over the Internet. Technical report RR-3296, INRIA, November 1997.

[12] L. Vicisano, L. Rizzo and J. Crowcroft. TCP-like congestion control for layered multicast data transfer. Proc. of INFOCOM'98, San Francisco, California, p. 996, March/April 1998.

[13] S. Bhattacharyya, J. Kurose, D. Towsley and R. Nagarajan. Efficient Multicast Flow Control using Multiple Multicast Groups. Proc. of INFOCOM'98, San Francisco, California, April 1998.

[14] T. Faber. ACC: Using Active Networking to Enhance Feedback Congestion Control Mechanisms. IEEE Network, Special Issue: Active and Programmable Networks, 12(3):61-65, May/June 1998.

[15] M. Calderón, M. Sedano, A Azcorra and C. Alonso. Active network Support for Multicast Appli-cations. IEEE Network, Special Issue: Active and Programmable Networks, 12(3):46-52, May/June 1998.

[16] D. Harel. Statecharts: A Visual Formalism for Complex Systems. Science of Computer Program-ming, Vol. 8, pp. 231-274, 1987.