A SIMULATION OF ECN-CAPABLE MULTICAST MULTIMEDIA DELIVERY IN *NS*-2 ENVIRONMENT

Robert R. Chodorek
Department of Telecommunications
The AGH University of Technology
al. Mickiewicza 30
30-059 Kraków
Poland
E-mail: chodorek@kt.agh.edu.pl

KEYWORDS
ns-2, RTP, multicasting, multimedia delivery, congestion control, Explicit Congestion Notification (ECN).

ABSTRACT

Explicit Congestion Notification (ECN) transfers a clear indication of congestion through the network and, therefore, gives a possibility to construct closed-loop control of effective transmission rate. In the paper, a simulation model of ECN-capable multicast multimedia delivery in ns-2 environment is discussed. The model includes complete ECN-capable network infrastructure (sender, receivers, network nodes). Special emphasis is placed on the core of the system, the ECN-capable RTP protocol. Simulation results confirm credibility of the model.

INTRODUCTION

Explicit Congestion Notification (ECN) relies on ability of the network to detect congestion build-up. Opposite to the traditional congestion avoidance methods, based on the packet dropping, network which implements ECN is able to react on incipient stages of congestion.
The ECN-capable IP is described in the RFC 3168 (Ramakrishnan et al. 2001). The ECN-capable IP itself isn't able to provide congestion avoidance, it need support from the upper (transport) layer. ECN-capable TCP congestion avoidance (Ramakrishnan *et al.* 2001) uses ECN bits as a feedback, which inform the sender about possibility of congestion. ECN are looped at the end of connection (to the receiver) and returned to the source in the TCP packet as so-called ECN-Echo flag.
Explicit Congestion Notification was implemented in Berkeley's *ns*-2 simulator (Floyd 1998). Nowadays, ECN-capability in *ns*-2 covers:

- IP header processing conformable to RFC 3168,
- modified RED queue mechanism; where RED gateways set the ECN bit in the packet header, rather than dropping the packet,
- ECN-capable TCP transport protocol.

In result, *ns*-2 allows simulation of reliable unicast connections via TCP (unicast elastic traffic) and there is no possibility of simulation of real-time multimedia transmission (non-elastic traffic, usually multicastly distributed through the network). The aim of the paper is to propose extensions for ns-2, which make possible to simulate an ECN-capable multicast multimedia delivery.
The rest of the paper is organized as follows. Section two proposes the ECN-capable RTP transport protocol. Section three contains an overview of the *ns*-2 simulator. Section four describes the implementation of ECN-capable RTP protocol in *ns*-2 environment. In the fifth section simulation model of ECN-capable receiver-driven layered multicast is presented, while section six analyses the simulation results. Section seven summarizes our experiences.

THE ECN-CAPABLE RTP PROTOCOL

The RTP transport protocol is commonly used for delivery of the multimedia streams, but there is no standardized usage of ECN-based congestion control for RTP yet.

The RTP Transport Protocol

The RTP (Real-time Transport Protocol) version 2 (Frederick et al. 1996) provides end-to-end delivery services for data with real-time characteristics. Applications typically run RTP on top of UDP transport protocol to make use of its multiplexing and checksum services. RTP can provide data transfer to multiple destinations using multicast distribution (depending on underlying network).
The RTP cooperates with the RTCP control protocol. The RTCP protocol provides feedback on the quality of data distribution using report packets. The feedback may be directly useful for control of adaptive encoding. RTCP also carries a persistent transport-level identifier for an RTP source. In multicast sessions, rate of RTCP packet must be scaled to numbers of participants. An optional function o RTCP is to convey minimal session control information.

The Proposition of the ECN-capable RTP protocol

ECN-capable RTP sender sets the ECN bits in the IP header to indicate the ECN-capability of the transport protocol end-point. Packet with ECN-Capable Transport codepoint CE(`0`) (i.e. '`10`') are sent to the receiver(s) using unicast or multicast transmission. If the congestion is build-up, routers on delivery tree will set the Congestion Experienced (CE) codepoint '`11`' in the IP header. In result, RTP receiver gets the information about congestion appearing in at least one network node along delivery tree.

The general ECN-capable receiver behaviour, recommended in RFC 3168, suggests that response to obtained CE codepoint must be essentially the same as response to a single dropped packet. RTP generally reacts on packet drops by modification of statistics in RTCP reports. Such a reaction is too weak to ensure effective congestion avoidance. Therefore we propose, that if receiver obtains congestion indication (a single CE packet), the feedback to the sender will be sent. The data receiver informs the data sender, when a CE packet has been received, by sending a new type of RTCP message - ECN-feedback (EFB).

EFB message conveys the information mandatory to RTCP message, as defined in (Frederick et al. 1996), and, additionally, includes reception report blocks (one for each of the synchronisation sources from which this receiver has received RTP data packets). Each reception report block provides ECN-feedback information.

According to the location of traffic controller in end-systems, ECN-capable congestion control can be performed in sender-driven or receiver-driven manner. Sender-driven control can utilise EFB message. If the EFB message is received, the RTP sender will send via API, to the sender application, the request of reducing effective transmission rate (e.g. through change of compression level). Lack of congestion also can be reported to the sender application.

Explicit congestion notification can be also send by RTP receiver to receiver application, via API interface. Receiver application can hold the EFB messages transfer and perform congestion control itself (receiver-driven control).

THE BERKELEY'S *NS*-2 SIMULATOR

The discrete-event *ns*-2 simulator, developed in U. C. Berkeley (Fall and Vradhan 2002), is an multi-protocol network simulator, which both delivers tools for simulation of standard network protocols (IP, IPv6, TCP, UDP, RTP, HTTP, FTP, etc.) and offers an infrastructure for design and deployment of new network protocols. The simulator supports many network technologies, including LANs, WANs, wired and wireless networks, QoS assurance, etc.

The *ns*-2 is an object oriented simulator, written in C++, with an OTcl interpreter as a front-end. Typically, network protocols are simulated using both programming languages. Protocol processing mechanisms are usually written in C++, while the experiment (topology, configuration, etc.) is described in OTcl.

Protocols in *ns*-2 are represented by objects, which are constructed as derived classes of the base *Agent* class (written in C++). If the protocol supports session mechanisms, the derived classes of base *Session* class (also written in C++) will be constructed. Above C++ classes are mirrored into a similar class hierarchy within the OTcl interpreter. For example, the RTP implementation in *ns*-2 environment consists of three C++ classes:

- *RTPAgent* – RTP protocol processing mechanisms,
- *RTCPAgent* – RTCP protocol processing mechanisms,
- *RTPSession* – the RTP session management.

C++ classes are closely mirrored by corresponding objects in the OTcl class hierarchy, respectively, Agent/RTP, Agent/RTCP and Session/RTP.

IMPLEMENTATION OF ECN-CAPABLE RTP PROTOCOL IN *NS*-2 ENVIRONMENT

The proposed ECN-capable RTP is build on the basis of the existing RTP simulation module. Six new objects, three written in C++ (*RTPECNAgent*, *RTCPECNAgent*, *RTPECNSession*) and three written in OTcl (Agent/RTP/ECN, Agent/RTCP/ECN, Session/RTP/ECN), are derived from standard RTP classes described above.

The *RTPECNAgent*

The RTP protocol mechanisms are represented in the *ns*-2 simulator by the *RTPAgent* class. Applications can access RTP agents via the *sendmsg*() function in C++ (or via the *send* or *sendmsg* methods in OTcl). Prior to internal data processing (mainly buffering and segmentation), new packets are created (using the *makepkt*() method). Each packet contains a monotonically increasing sequence number and an RTP timestamp. Packets are sent to the network node via the *sendpkt*() method and transferred by the network. The upstream *RTPAgent* receives packets from its network node using the *recv*() method. It's worth remarking that application data are typically represented by their size and transmission time and they are not actually transferred in the simulator.

```
class RTPECNAgent : public RTPAgent {
 public:
    RTPECNAgent();
    virtual void recv(Packet* p, Handler*);
    virtual int command(int argc,
                        const char*const* argv);
    virtual void sendmsg(int nbytes,
                         const char *flags = 0);
 protected:
    virtual void sendpkt();
    virtual void makepkt(Packet*);
    RTPECNSession* session_;
};
```

Figure 1: The *RTPECNAgent* Definition.

The *RTPECNAgent* class overrides a number of *RTPAgent* methods (see Figure 1) and introduces ECN-capability into RTP transport protocol. The new *makepkt*() method creates RTP packet and sets CE(0) codepoint in the IP header to indicate the ECN-capability of the RTP protocol. Such a method of the IP header processing is often used in *ns*-2 simulator (e.g. by the ECN-capable TCP protocol).

Packets are sent to the network node via the modified *sendpkt*() method of an *RTPECNAgent* (the sender). Thus ECN-capable RTP packets are sent to the receiver (or receivers) using unicast (or multicast) transmission. As in the case of "standard" (non- ECN-capable) transmission, the *recv*() method of an *RTPECNAgent* (the receiver) is invoked by upstream nodes when sending a packet.

The modified *recv*() method analyses ECN bits. When the *RTPECNAgent* receives a data packet with the ECN bit set in the packet header, the receiver performs one of the following actions:

- the *RTPECNAgent* informs receiver application via OTcl callback function (receiver-driven congestion avoidance),
- the *RTPECNAgent* sends ECN feedback to the sender using *RTCPECNAgent* class (sender-driven congestion avoidance),
- do nothing (no congestion avoidance).

The performed action is defined by two flags in the Agent/RTP/ECN object. The *receiver_driven* flag indicates (by being set) the receiver-driven mode of congestion avoidance and the *sender_driven* flag indicates the sender-driven one. Default values for both flags are "false" and indicate no congestion avoidance. If both flags are set to "true", both sender and receiver will be notified about possibility of congestion (not recommended, for statistical purposes only).

The *RTCPECNAgent* and the *RTPECNSession*

The RTCP protocol monitors and controls RTP session. In the *ns*-2 simulator, the RTCP protocol functionality was divided between two objects: the *RTCPAgent* class and the *RTPSession* class. The *RTCPAgent* class implements the RTCP packet processing, while the *RTPSession* class implements the RTP session management.
The RTCP protocol behaviour is based on the periodic transmission of control packets. The packet is prepared by the *RTPSession* in response to the *RTCPAgent* request. The *RTCPAgent* class simulates the transmission and reception of the RTCP packets using, respectively, the *sendpkt*() and *recv*() methods.

```
class RTCPECNAgent : public RTCPAgent {
 public:
    RTCPECNAgent();
    int command(int argc,
                const char*const* argv);
 protected:
    void sendpkt();
    RTPECNSession* session_;
    double rtcp_efb_interval(int members,
              int senders, double rtcp_bw);
    int multicast_ECN_timer_;
};

class RTPECNSession : public RTPSession {
 public:
    RTPECNSession();
    int command(int argc,
                  const char*const* argv);
    int build_report(int bye);
protected:
    int build_efb();
};
```

Figure2: The *RTCPECNAgent* and *RTPECNSession* Definition.

The *RTCPECNAgent* class and the *RTPECNSession* class (Figure 2) introduces ECN-capability into RTCP protocol. The *RTCPECNAgent* class implements the new *sendpkt*() method, which is able to send additional type of report – the ECN-feedback (EFB) report. The EFB report is transmitted to the sending *RTPECNAgent*, when the *RTPECNAgent* receives a data packet with the ECN notification and the *sender_driven* flag in the Agent/RTP/ECN object is set. The EFB report is prepared by the *RTPECNSession* class on the *RTCPECNAgent* request. Each RTCP report is generated by the modified *build_report*() method of *RTPECNSession* class. If the RTCP report is EFB report, the *build_efb*() method is called by the *build_report*().
The EFB report sending mode depends on the type of RTP transmission (unicast or multicast). When the *RTPECNAgent* receives the ECN notification during the unicast transmission, the EFB report is immediately send to the sender. During multicast transmission, such a report generation policy may cause cumulative multiplication of EFB reports (so-called feedback implosion) and, in result, the network at the sender site becomes congested from the cumulative back-traffic from the receivers. Therefore, the interval between reception of ECN notification and EFB transmission is generated by the *rtcp_efb_interval*() method from the uniform distribution on $[0; \phi]$, where ϕ depends on the total number of receivers in the multicast group.

SIMULATION MODEL OF ECN-CAPABLE RECEIVER-DRIVEN LAYERED MULTICAST

As an example of possible application of ECN-capable RTP transmission, a simulation model of ECN-capable receiver-driven layered multicast was developed. The exemplary solution combines network-based explicit congestion notification with receiver-driven layered multicast scheme.

Layered Multicast Transmission

Layered encoding assumes that high-quality data stream is encoded into several complementary substreams (layers). Usually layers are encoded in cumulative manner, as the complete base stream and some supplementary streams.
Receiver-driven layered multicast scheme (Kim and Ammar 2001; McCanne et al. 1996) requires layered encoding - input multimedia stream is compressed into several substreams with different QoS requirements. Typically base stream (the lowest picture quality with the minimal bandwidth requirements) and several other streams to improve quality are used. Layers (substreams) are strictly synchronised and simultaneously transmitted through the network as separate multicast groups. Receivers can individually subscribe or unsubscribe to the appropriate multicast group to achieve the best quality signal that the network can deliver. Layers can be joining/leaving only in order of their relevance.

A Proposition of ECN-Capable Receiver-Driven Layered Multicast Multimedia Delivery

ECN-capable receiver-driven layered multicast combines explicit congestion notification with the layered multicast scheme. Decisions about changing group membership and, in result, about changing effective transmission rate, are taken based on ECN notification. Such a transmission scheme forms a close loop control from point of congestion to the receiver and to the point of congestion again.

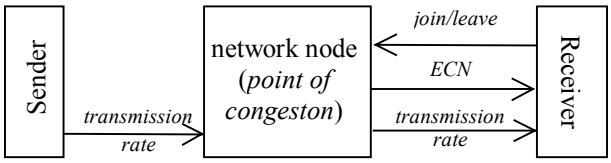

Figure 3: Block Diagram of ECN-Capable Receiver-Driven Control System (Structural Approach).

If the dangerous of congestion appears in the network node, ECN will be transmitted from the possible point of congestion to the receiver (Figure 3). Based on obtained congestion notification, the receiver takes an autonomous decision about leaving multicast groups and, in result, about reducing effective transmission rate.
Lack of ECN implies small possibility of congestion. Receiver responds to lack of ECN by increasing effective transmission rate. This is accomplished by joining the multicast group that transmits the supplementary substream.

Implementation of ECN-Capable Receiver-Driven Layered Multicast in *ns*-2 Environment

A system able to provide ECN-capable layered multicast consists of an ECN-capable sender, set of ECN-capable network nodes and an ECN-capable receiver.
The sender node is built using the sender application able to perform layered transmission, connected to the node via ECN-capable RTP protocol. Because the ECN-capable congestion avoidance is performed in receiver-driven manner, the *sender_driven* flag is set to "false".
The ECN-capable network nodes are the part of *ns*-2. Each ECN-capable network node marks packet according to RED algorithm (instead of packet dropping). Because of non-stability of classic RED queue (packets are marked based on current average queue size of all streams and, in result, receivers frequently change received layer), multilevel RED queue is used. Thus, in congested link, high rate stream that produce congestion is marked first, and then medium rate stream is marked. High priority low rate base stream is marked only when average RED queue size is very large. Therefore, in the case of congestion, streams that minimize network load in congested link are promoted.

```
# create ECN-capable RTP
set rtp1 [new Agent/RTP/ECN]
$ns attach-agent $n1 $rtp1

# create ECN-capable application
set app1 [new Application/LM/ECN]
$app1 attach-agent $rtp1
$rtp1 set receiver_app $app1

$rtp1 set reciver_driven true
```

Figure 4: A Fragment of the OTcl Script Defining the ECN-Capable Layered Multicast Receiver.

The ECN-capable layered multicast receiver consists of two components, the ECN-capable application, able to perform layered multicast reception (Application/LM/ECN object in OTcl – see Figure 4), and the ECN-capable RTP protocol (Agent/RTP/ECN object in OTcl). Both components are connected using *attach-agent* method for data transmission and *receiver_app* method to assign callback function.
The *receiver_driven* flag set to "true" enables the receiver-driven ECN-capable congestion avoidance.

SIMULATION RESULTS

Congestion in the network is observed as buffers overflow and, in result, as packet losses. Therefore commonly used metric to monitoring the network for congestion is the packet loss ratio - the percentage of all packets discarded for lack of buffer space.

Configuration of the experiment

The network topology used in simulation is shown in Figure 5. The sender is connected to the router A with a link at 10 Mbps bandwidth and 1 ms delay. Receivers R1, R2 and R3 are connected to the router B through 5 ms delay link at 2 Mbps, 0.8 Mbps and 0.4 Mbps, respectively. Router A use the classical Drop Tail queue management. Router B use ECN-capable RED mechanism with three levels for active queue management. Routers are connected with a link at 2 Mbps bandwidth and 10 ms delay.

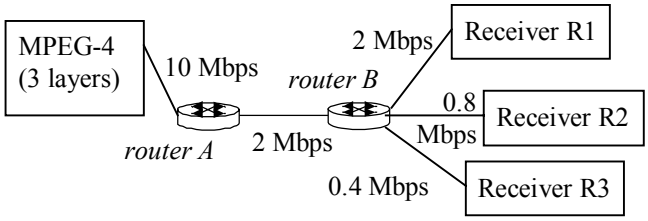

Figure 5: The Network Topology Used in Simulation

Layered videos are based on ten MPEG-4 traces, encoded with high, medium and low picture quality. Video traces are generated from movies (*Jurassic Park*, *Mr Bean*, *Die Hard*, *Star Trek*, *Star Wars*, *Starship Troopers*), sport event recorded from German cable TV (*Formula 1*), static camera located in lecture room (*cam1*) or in office (*cam2*) and from parking security camera (*cam3*). Each test sequence, used in the experiment, consists of 15 000 video frames at 25 Hz and represents 10 mins of video. Properties of used video traces can be found in (Fitzek and Reisslein 2001).
The video stream is distributed from MPEG-4 sender to all receivers (MPEG receiver 1 through 3) using multicast layered transmission. Receiver connects to the multicast group, which transmit the high priority stream (containing layer 1), and, optionally, to one or two supplementary groups (layer 2 and layer 3). Supplementary groups carry low priority streams.

Results

Receiver R1
It was empirically proved that all high quality traces could be transmitted via 2 Mbps link without packet drops. In result, receiver R1, connected via 2 Mbps (non-congested) link is able to receive full video information (layer 1 to 3).

Results obtained for receiver R1 show that transmission is stable, no packet drops are observed. Because link isn't congested, ECNs are not observed.

Receiver R2

Receiver R2 is connected to router B via lightly congested link. For a great amount of tested video sequences, R2 is able to receive layer 2. If the video source is characterised by low-detail, slowly dynamic content, R2 can connect to layer 3. Such a situation was observed for e.g. for *cam1* video source (typical "talking head").

Table 1: Statistical Properties of Packet Loss Ratio Obtained for Receiver R2

video source	high priority stream			low priority streams		
	min	max	mean	min	max	mean
Jurassic Park	0%	1%	0.025%	0%	13%	0.2%
the others	0%	0%	0%	0%	0%	0%

For almost all test video sequences, transmission between sender and R2 is stable, no packet drops are observed. A low packet loss ratio is observed only for 1 in 10 video sequences (*Jurassic Park* – see Table 1). Moreover, all observed packet losses occurs only between 10 and 12 second of transmission and they are caused by unexpected scene change (a dynamic scene follows the slowly one). Generally, no ECNs are observed at the high priority stream and relatively small amount of ECN marked packet are observed for low priority stream. In result, the ECN-capable layered multicast allows reception of the best quality signal that the network can deliver (layers 2 or 3) without packet drops.

Receiver R3

Receiver R3 is connected to router B through heavy congested link. R3 is able to receive layer 1 and, if the video trace is small enough, can connect to layer 2.

Table 2: Statistical Properties of Packet Loss Ratio Obtained for Receiver R3

video source	high priority stream			low priority streams		
	min	max	mean	min	max	mean
Jurassic Park	0%	4%	0.1%	0%	45%	0.99%
cam3	0%	0%	0%	0%	8.9%	0.15%
Die Hard	0%	0%	0%	0%	4.7%	0.07%
Formula 1	0%	0%	0%	0%	1.1%	0.02%
Star Wars	0%	2%	0.035%	0%	19%	0.3%
the others	0%	0%	0%	0%	0%	0%

For half of test video sequences, transmission between sender and R3 is stable, no packet drops are observed (Table 2). High priority stream (layer 1) is usually transmitted error-free, only for two video sequences (*Jurassic Park* and *Star Wars*) small packet drops are detected. Low priority stream (layer 2 and layer 3) has incidental packet drops, but average packet loss ratio never exceeds 1%. A small amount of ECNs appears for layer 1 and larger for layer 2.

CONCLUSIONS

This paper presents a multimedia delivery system able to use ECN-capability of IP protocol. The core of the proposed multicast multimedia delivery is the extended RTP transport protocol. The main innovation introduced to the RTP protocol is its ability to use congestion information carried by ECN bits. Proposed solution supports both types of congestion control: the sender-driven and the receiver-driven one.

On the basis of the proposed definition of ECN-capable RTP protocol, new *ns*-2 RTP module has been created. This module was used as a part of an ECN-capable receiver-driven layered multicast transmission system.

The simulation model was tested in various network environments and in different network conditions. The exemplary results are shown in the paper. The behaviour of the simulations conform to expectations - experiments show that the use of ECN-capable multicast transmission allows achieving stable transmission of the layered video stream, with no packet loss or low packet loss ratio.

Implemented modules may be used for further research on ECN-capable congestion avoidance for multimedia communication.

ACKNOWLEDGEMENTS

This research was supported by State Committee for Scientific Research (KBN) under grant 7 T11D 012 20.

REFERENCES

Fall, K. and K. Vradhan. 2002. "The ns Manual." URL http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf. (Apr.).

Fitzek, F.H.P. and M. Reisslein. 2001. "MPEG-4 and H.263 Video Traces for Network Performance Evaluation". *IEEE Network* 15, No.6 (Nov./Dec.), 40-54.

Floyd, S. 1998. "ECN implementations in the ns simulator". URL http://www.aciri.org/floyd/papers/ecnsims.ps. (Dec.).

Frederick, R.; V. Jacobson; H. Schulzrinne; and S. Casner. 1996. "RTP: A Transport Protocol for Real-Time Applications". RFC 1889. (Jan).

Kim, T. and M.H. Ammar. 2001. "A Comparison of Layering and Stream Replication Video Multicast Schemes". In *Proceedings of 2001 NOSSDAV* (Port Jefferson, NY, Jun.25-26).

McCanne, S.; V. Jacobson; and M. Vetterli. 1996. "Receiver Driven Layered Multicast". In *Proceedings of 1996 ACM SIGCOMM* (Stanford, CA, USA, Oct. 28-30). 117-130.

Ramakrishnan, K.; S. Floyd; and D. Black. 2001. "The Addition of Explicit Congestion Notification (ECN) to IP". RFC 3168. (Sep).

AUTHOR BIOGRAPHY

ROBERT R. CHODOREK was born in Łagów, Poland, in June 1966. He received the M.S. degree in electrical engineering from the Kielce University of Technology, Kielce, Poland, in 1990, and the Ph.D. degree in computer sciences from the AGH University of Technology, Cracow, Poland, in 1996. He is currently an assistant professor at the AGH University of Technology in Cracow, Poland. His research interests include modeling and performance evaluation of telecommunication networks, in particular high speed networking for multimedia communications.