INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

On-Demand Layer Addition (ODL): Making Multi-Layer Multicast Transmissions Cheaper

Vincent Roca

N° 0239

Février 2000

THÈME 1

Rapport technique

ISSN 0249-0803

On-Demand Layer Addition (ODL): Making Multi-Layer Multicast Transmissions Cheaper

Vincent Roca

Thème 1 — Réseaux et systèmes
Projet PLANETE

Rapport technique n ° 0239 — Février 2000 — 17 pages

Abstract: The context of this work is the multicast transmission of data using multiple multicast groups, one per layer of information. One reason to use multiple groups is to address the potential heterogeneity of receivers and avoid that high-end receivers suffer from a reduced reception rate because of the presence of low-end receivers. But there is a risk that some of the groups are not used. Sending data to a group with no receiver has a cost that is often underestimated: information is kept in routers, periodic tree management traffic is required with both dense and sparse mode protocols, packets are sent on the LAN even if they are dropped by the first-hop router, IGMPv1 has limitations, etc. As IP multicast becomes widely deployed and used, these issues may well compromise its scalability. In this paper we introduce a new protocol, ODL (On Demand Layer Addition), which *enables a source to use only the layers (i.e. groups) that are actually required by the current set of receivers.* We describe its behavior when used with several kinds of packet scheduling schemes (cumulative or not) and different scenarios (one-to-many versus many-to-many). We also show that this protocol can lead to significant savings when used with router filtering techniques, even if a single multicast group is used in that case. We have implemented the ODL protocol, integrated it in a multicast library and we report several experiments that assess its benefits.

Key-words: networks, multicast, multi-layer transmissions, multiple multicast groups, scalability, multicast routing

(Résumé : tsvp)

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN (France)
Téléphone : 04 76 61 52 00 - International: +33 4 76 61 52 00
Télécopie : 04 76 61 52 52 - International: +33 4 76 61 52 52

On-Demand Layer Addition (ODL): une technique pour rendre les communications multipoints multi-couches moins coûteuses

Résumé : Le contexte de ce travail est celui des transmissions de données multi-couches, chaque couche utilisant un groupe multicast distinct. C'est le cas par exemple des codages vidéo hiérarchiques, de l'approche ALC (Asynchronous Layered Coding), etc. Une motivation pour utiliser plusieurs groupes est de pouvoir prendre en compte l'hétérogénéité des récepteurs, et ainsi d'éviter que les récepteurs de bonne qualité ne soient pénalisés car le rythme de transmission se fait sur la base des récepteurs plus lents. Mais il y a un risque que certains de ces groupes ne soient pas utilisés. Transmettre des données à un groupe multicast qui n'est utilisé par aucun récepteur a un coût qui est souvent sous-estimé: des informations d'état sont maintenues par les routeurs, des trafics périodiques de gestion sont nécessaires que ce soit avec les protocoles de routage en mode dense ou épars, des paquets sont transmis sur le LAN même s'ils sont ensuite détruits par le premier routeur, IGMPv1 a des limites, etc. Au fur et à mesure que le déploiement et l'utilisation d'IP multicast progresse, ces limitations peuvent devenir un problème et compromettre le passage à l'échelle. Dans ce document nous introduisons le protocole ODL (On-Demand Layer Addition), qui *permet à une source d'utiliser en permanence uniquement les couches (et donc les groupes) pour lesquelles existe au moins un récepteur.* Nous décrivons son comportement avec plusieurs techniques d'ordonnancement de paquets en couches (cumulatives ou non) et différents scénarios (un vers n ou n vers m). Nous montrons également que ce protocole peut conduire à des gains significatifs lorsqu'il est utilisé avec des techniques de filtrage par les routeurs, même si dans ce cas un seul groupe est utilisé. Nous avons implémenté le protocole ODL, l'avons intégré à une librairie multicast et rendons compte de plusieurs expériences qui établissent ses intérêts.

Mots-clé : réseaux, multipoint, transmissions multi-couches, groupes multipoints multiples, facteur d'échelle, routage multipoint

1 Introduction

1.1 Addressing Heterogeneity

One trend of research in multicast communications is to rely on several multicast groups. This is the case for cumulative layered transmissions used by video applications [5][38]. To accommodate the potential heterogeneity of the receivers in terms of processing power (e.g. a PDA versus a powerful workstation) and/or transmission capabilities, the source uses a layered data coding and transmits each layer in a separate multicast group. Users subscribe to as many groups as possible according to their available bandwidth. If the associated congestion control mechanism indicates that a user should reduce its incoming traffic, then this latter unsubscribes to one or more groups. As soon as all the users behind the bottleneck router have unsubscribed, this branch of the multicast distribution tree is pruned, thereby reducing the load on the router [22][37].

Another example is the multicast distribution of popular files (e.g. the promotion clip of a film, the latest Linux distribution, etc.). As it is assumed that many users will retrieve the file simultaneously, this latter is sent continuously. Using several multicast groups and an appropriate scheduling scheme ([36], MCM [9], ALC [21], RMDP [28]) gives the opportunity to address the heterogeneity of receivers. This class of applications differs from the previous one by the fact that data is known in advance rather than produced in real-time like a video stream.

Finally DSG [3] or GMMG [30][31] can be used for the distribution of the data flows generated on-the-fly where each receiver must receive all the data (unlike layered coding techniques). These schemes also rely on the use of several groups to address the heterogeneity. They can be used with such cooperative work applications as a white-board [39].

These three kinds of applications use a fixed number of multicast groups usually given as an argument at start up. The risk that some of these groups are not used by any receiver is thus non zero, especially with continuous file transmissions.

1.2 The Various Costs Associated to a Multicast Group

Of course the data packets sent to a group with no receiver associated will be dropped by the first-hop multicast router. But there are additional costs that are often underestimated:

- A dense mode protocol like DVMRP [25] or PIM-DM [8] relies on the periodical flooding of data as far as made possible by the TTL. It greatly increases the network traffic and the router load even on branches with no receiver.
- With dense-mode protocols, all the routers, even if they do not belong to the multicast distribution tree, keep information for this group [1]. For instance, a router running PIM-DM keeps a context with the source and group addresses, the state and a timer for each group in "prune state" (i.e. without any underneath receiver) [8]. Section 4.3 gives more details in case of DVMRP/`mrouted`.
- The MOSPF protocol [23] requires that each router of the domain keeps and exchanges state for each multicast group. This information is then used by multicast routers to create the local view of the distribution tree. An unused group will also require such a state.
- Using a sparse-mode protocol like PIM-SM [40] requires using MSDP (Multicast Source Discovery Protocol) [10] to inform remote domains of the presence of a source in the local domain. Whenever a new source becomes active, the local MSDP peer announces its presence (`SA` message) to all directly connected MSDP peers, who then forward the message to other peers. This information is periodically refreshed and may also be cached [1]. This Internet-wide periodic flooding (and state) takes place even if the group has no member!
- And finally a multicast address is reserved while class D addresses are a scarce resource with IPv4.

1.3 Goals of the ODL Protocol

Minimizing the number of multicast groups used is all the more important as these applications are to become popular. In this paper we introduce a new protocol, ODL (On Demand Layer addition), that is well suited to these layered transmission schemes. It relies on the presence of a *receiver-oriented* congestion control scheme for layered transmissions (e.g. RLM [22], RLC [37]) where each receiver chooses to add or drop a layer within the fixed set of available layers. ODL's goal is to *enable the source to use only the layers that are actually required by the current pool of receivers.*

We also show that this protocol can lead to significant savings when used with router filtering techniques like [20] or TUF [7] even if a single multicast group is used in that case.

The rest of the paper is organized as follows: Section 2 details the ODL protocol. Section 3 studies the benefits of ODL with non cumulative and router-based filtering schemes. Then section 4 introduces its implementation, discusses timer initialization, and presents experimental results. Section 5 discusses related work. Finally we conclude.

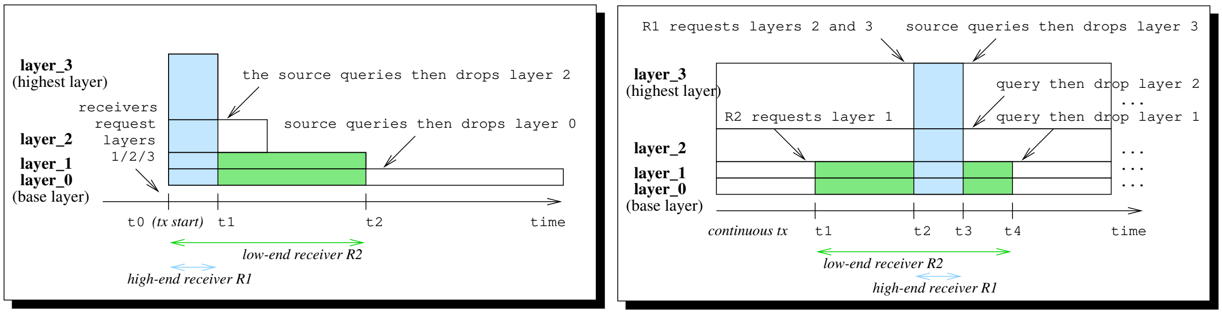
2 The ODL Protocol

In this section we assume (1) that each layer is carried on a *separate multicast group*. Therefore *the source* adds or drops a layer by sending or not packets to a particular multicast address. After the source stops sending new packets to a group, the soft-state kept by multicast routers (section 4.3) for this group slowly disappears (e.g. `mrouted` uses a default 5 minutes timer). *A receiver* adds or drops a layer by joining or leaving the associated multicast group. We also assume (2) that transmissions are *cumulative*, i.e. receiving layer i requires receiving all the layers below. Section 3 analyzes the use of ODL when these two assumptions are not met.

2.1 Sketch of the protocol

ODL is an *end-to-end protocol*, working in a request/response mode. Therefore ODL has no requirement neither on the router functionalities nor on the routing and IGMP protocols in use.

ODL relies on both the source and the receiver sides: it is the *receiver*'s responsibility to ask for additional layers when appropriate (e.g. if his congestion control module says so) and to respond to `QUERY` messages. It is up to the *source* to create additional layers when asked to do so by a receiver. Of course he can refuse (e.g. if his maximum number of layers has been reached). The source also checks periodically that each multicast group is used by one or more receivers by sending `QUERY` messages. If nobody responds, then this (now useless) layer is dropped.



(a) Synchronous start

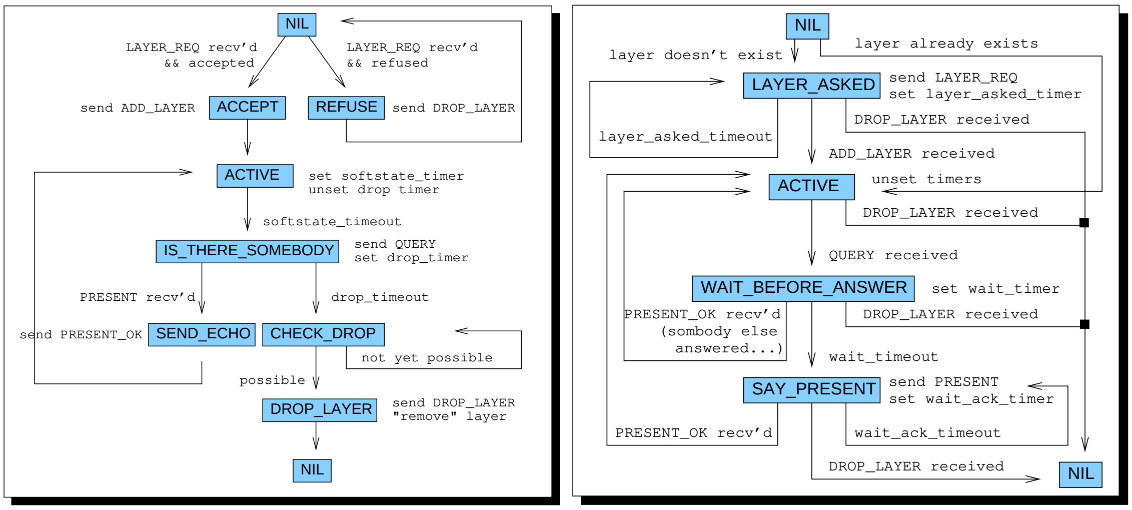
(b) Asynchronous start

Figure 1: Transmissions on each layer with a synchronous or asynchronous start of the receivers; with ODL only the grey areas are used.

Figures 1 (a) and (b) compare layer management at the source with and without the ODL protocol. There are two receivers, a high-end receiver R_1 who subscribes to all four layers, and a low-end receiver R_2 who only subscribes to the first two layers. We consider two kinds of applications:

- an application doing a one time file transfer (figure 1 (a) where MMG [4] is assumed): here all the receivers must be ready before the transmission starts (synchronous start).
- an application doing continuous file transfers (figure 1 (b)): here receivers can arrive at any time (asynchronous start).

From Figure 1, we see that using ODL enables the source to remain as close as possible to the actual requirements of the receiver set. For instance layers 2 and 3 are quickly dropped after that R_1 has finished receiving the whole file. The periods when the various layers are indeed used are represented in (dark/light) grey. The white areas correspond to a useless presence of the layers (no receiver any more) when ODL is not used. Note that so far we did not assumed any latency between the time the last receiver leaves and the time it is detected by the source. This is discussed in section 4.2.



(a) Source

(b) Receiver

Figure 2: (Simplified) Finite State Machines for each layer controlled by ODL.

Table 1: ODL signalization messages.

type	sent by	used by	kind of transmission	description
INFO_REQ	receiver	source	unicast to source	a receiver asks for information
INFO	source	receivers	multicast on layer 0	information on all active layers
LAYER_REQ	receiver	source	unicast to source	a receiver asks for a new layer
ADD_LAYER	source	receivers	multicast to layer 0	a new layer has been added
DROP_LAYER	source	receivers	mcast on target layer	the target layer is dropped
QUERY	source	receiver	mcast on target layer	does anybody receive this layer?
PRESENT	receiver	source	unicast to source	yes, this receiver uses the layer
PRESENT_OK	source	receivers	mcast on target layer	acknowledges PRESENT message

2.2 Detailed Description for the Sending Side

We first consider the sending side and we assume (1) there is only one source and (2) that receivers are not on the same host as the source. In particular it means that the source is not a receiver.

At session start, the source uses a single layer. We assume that the address of the associated multicast group has been communicated to receivers independently (`sdr`, web, mail). *With an application where asynchronous starts are possible, this base layer (layer 0) is permanent*, i.e. not controlled by ODL. The reason is that it is also used as a signalization channel by the source. On the contrary, *all layers above 0 are under ODL's control*, and they only exist when at least one receiver has joined the group. With an application where synchronous start is required, the base layer can be controlled by ODL since layer 0 has no interest once the last receiver has left.

To know if there is at least one receiver for a given layer, the source periodically transmits a `QUERY` message on the target group. Because transmissions are not reliable, the `QUERY` message is sent several times. If no answer, i.e. `PRESENT` message, is received after a given time, the layer is dropped. Therefore the source issues a final `DROP_LAYER` message and then avoids sending any packet on this group anymore. The associated distribution tree will slowly disappear from the multicast routers as their soft-state times out. On the contrary if the source receives a PRESENT message, then the layer is kept and an acknowledgement (PRESENT_OK message) is echoed to the group. In order to detect duplicates, QUERY messages include an identifier that is written back in PRESENT and PRESENT_OK messages.

ODL uses two timers at the sending side:

- *softstate_timer:* periodicity of the QUERY messages.
- *drop_timer:* waiting time before dropping a layer once the QUERY has been sent.

We discuss the value to use for these two timers in section 4.

Resolution of Conflicts when Dropping a Layer

Using a cumulative scheduling scheme means that layer i can only be dropped if there is no layer $> i$. In practice conflicts often take place. In figure 3 we assume that the last receiver leaves at time t_0. For each layer this departure is detected at the next query stage. Because the layer's timers are not synchronized, layers 0 and 1 must wait that layer 2 has been dropped before being dropped in their turn.

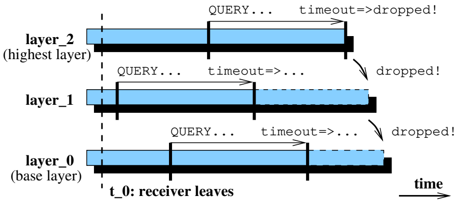


Figure 3: Interdependence of layers with a cumulative scheme; lower layers drop is delayed by the presence of layer 2.

2.3 Detailed Description for the Receiving Side

A receiver joining a session first asks for information by sending on the base layer (supposed known) an INFO_REQ message. The answer, the INFO message, informs each receiver of the current situation: number of layers, features of each layer, multicast address and port number for each layer, etc. The receiver can then join one or more layers depending on what the congestion control module says (congestion control is out of the scope of this paper). If no INFO message is received after a given time, then a new INFO_REQ message is sent. Of course after a given number of unsuccessful tries the receiver gives up.

If a receiver wants to benefit from an additional layer not yet available, he sends a LAYER_REQ message to the source using unicast. It is very important to avoid using multicast transmissions at the receiver in order to limit the multicast load to the network to the strict minimum: the data flows. Then the source replies with an ADD_LAYER message if he agrees, with a DROP_LAYER message otherwise.

Another major task of a receiver is to answer to QUERY messages. For the sake of scalability, the PRESENT message is not returned immediately but after a random time. At timeout and if no other message has been received, then the receiver sends a PRESENT message and waits for an acknowledgement. If no PRESENT_OK message is received after a given time, then a new PRESENT message is sent. After a given number of unsuccessful tries the receiver drops the layer.

2.4 The Case of Multiple Sources

The case of many-to-many transmissions is more complex. In particular the sources can be heterogeneous and they will not all offer the same number of transmission layers. Therefore *ODL must enable an individual treatment of each source.* To do so, the sender's private address (address/port number) of a message is always communicated to the receiver. If the multicast library (section 4.1) always reads ODL messages using the **recvfrom** or **recvmsg** socket system calls, then the sender's address is known and the receiver can reply in unicast[1] [32].

[1] Another solution is to assign a private identifier to each source. But this solution requires (1) a coordination among all of them in order to warrant unicity and (2) that a private address is known and stored along each identifier to enable unicast transmissions. Therefore we prefer the former solution.

The behavior of ODL in presence of several sources is the following:

- a layer addition request is directed to a particular source by specifying its private address and using unicast.
- a layer addition request may also be directed to all the sources by multicasting it to the base layer. But this solution is *not recommended at all* as it creates a new receiver-rooted multicast tree (at least with DVMRP).
- a source can deny a layer addition request for any reason (e.g. if he does not have sufficient resources). If the request was sent to him in unicast, then the source replies with a `DROP_LAYER` directed to the receiver.
- a source adding a layer multicasts this announcement to the base layer. Sources can use a random delay before sending the `ADD_LAYER` message if their number justifies the use of a suppression technique.

2.5 The Case of Receivers on the Same Host as a Source

A receiver on the same host as a source can usually receive data at full transmission speed and therefore requests all the layers. This is a problem if no remote receiver uses the upper layers since all the multicast groups are used anyway. The same situation occurs if the source is also a receiver. To avoid it, the source uses a TTL of 0 for layers where all the receivers are local. Therefore packets never leave the host. Whenever there is at least one foreign receiver, the TTL value specified at application start is used instead. The algorithm is the following:

```
on receiving a LAYER_REQ message for a new layer:
        if (the message comes from the local host)
                add layer in ``local transmission mode'' (i.e. with a TTL of 0);
                send ADD_LAYER with a TTL of 0 and switch to ``ACTIVE'' state;
        else
                do the normal processing (ADD_LAYER...);
                switch to ``ACTIVE'' state;

on receiving a PRESENT message:
        if (the message comes from the local host)
                note that there is at least one local receiver;
                do nothing else yet...
        else
                do the normal processing (PRESENT_OK...);
                switch to ``ACTIVE'' state;

on drop_timeout:
        if (there is a local receiver)
                go into ``local transmission mode'' (i.e. with a TTL of 0);
        else
                drop the layer;
```

3 Using ODL in Other Circumstances

In this section we examine how ODL can be used with two other solutions that address the heterogeneity problem: non cumulative multi-layer scheduling and router based filtering.

3.1 The Case of Non Cumulative Packet Scheduling Schemes

So far we assumed that a cumulative scheduling was used. This is not compulsory and ODL can be used with non cumulative approaches too. For instance, with DSG (Destination Set Grouping) [3], data is sent over separate multicast groups at different rates and a receiver chooses *the* appropriate group according to its capabilities. ODL is still applicable. To receive data faster, a receiver changes (rather than adding) of layer. If the new layer is not already active, then he first asks for it. Because it generates some delay, the receiver keeps on listening to the previous layer until the new one is available.

There are two major differences yet:

- the ODL signalization must be *sent to all the layers* rather than only on the base layer.
- dropping or adding a layer is immediate as there is no layer dependency.

3.2 The Case of Router Based Filtering Schemes

Another emerging class of solutions to address the problem of heterogeneous multicast transmissions is to rely on *router based filtering* [7][20]. These techniques discard packets from flows whose bandwidth exceed their fair share. The TUF proposal [7] adds a tag to each packet sent to identify the "priority level" (or drop precedence). [20] also advocates the use of filtering within routers but without any involvement by either the source or the receivers. Instead per-flow state must be kept in each router. Of course both approaches require extensions to (at least some key) multicast routers.

With router based filtering, a single multicast group is used. Therefore ODL does not lead to any benefit from the multicast distribution tree and address management points of view (section 1). Yet ODL is still interesting. Indeed, ODL enables a source to know the highest requirements of the set of receivers. Therefore the source will not generate a high-rate traffic if only low-end receivers exist which can save much bandwidth within the backbone network. Without ODL, this extra traffic would only be dropped at the bottleneck routers who may be far away.

But a major difference exists: ODL works on *discrete rate levels* (i.e. layers) whereas filtering schemes lead to continuous rates. We believe this is not a problem since many applications and scheduling schemes can only handle discrete transmission rates (e.g. layered video codecs, multi-quality speech coding, etc.).

Figure 4 illustrates the use of ODL with router based filtering like TUF, i.e. where the most useful data packets (lower layers) are transmitted preferentially. We assume there is a single receiver whose reception rate is represented by the curve. According to this reception rate, the receiver answers to ODL queries saying how many layers are indeed useful. The source adjusts its transmission rate accordingly by dropping or adding layers. The resulting traffic sent by the source is represented by the grey areas. The situation is similar with multiple receivers as the source considers the highest reception rate in that case.

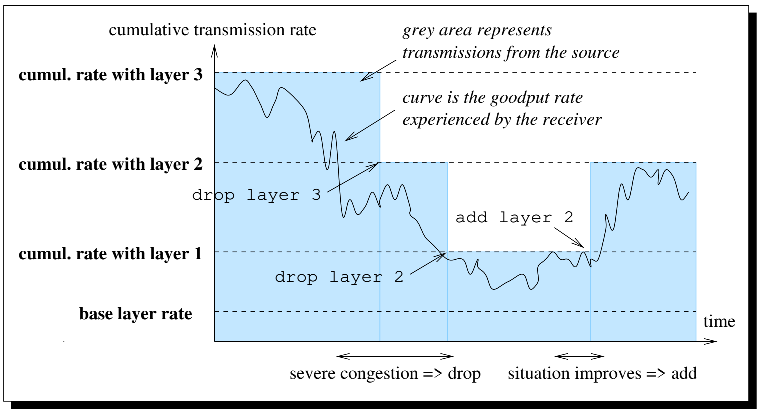


Figure 4: Using ODL with a router-based filtering scheme like TUF.

4 Implementation and Evaluation

This section describes an implementation of the ODL protocol, an evaluation of the various protocol timers, and experiments we carried out.

4.1 The ODL Building Block

We have implemented the ODL protocol as a building block [41] that we integrated to the MCL library [32]. Three functions are exported:

- a function to process ODL timers, called periodically,

- a function to process incoming ODL signalization messages, and
- a function to request a new layer (i.e. to trigger a LAYER_REQ message), used by the congestion control module.

This building block also requires several functions provided by the library: a function to send ODL messages, to add or drop a layer (ODL only performs the signalization part), etc.

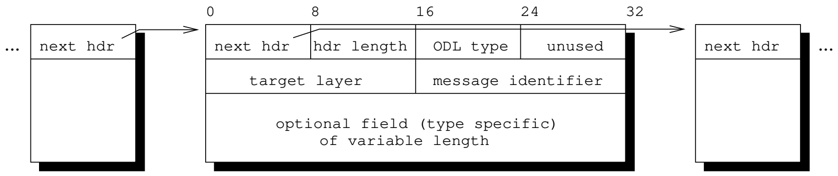


Figure 5: ODL extension header format.

The ODL messages are piggy-backed to data packets thanks to the concept of *extension header* (Figure 5) used for instance by IPv6. Any number of ODL (or non ODL) headers can be attached to a given packet as long as the MTU is not exceeded. This solution greatly reduces the transmission cost for downward signalization (source to receivers). Upward Signalization (receivers to source) is anyway limited by feedback cancellation techniques (section 2.3).

4.2 Analytical Analysis of the Timer Values

The values of the softstate_timer and drop_timer are essential since they control the ODL overhead, the discovery of unused layers, and the robustness in front of losses (section 2.2). This is the goal of this section.

4.2.1 The DROP Timer

Let us begin with the drop_timer. Its initial value depends on the following parameters:

- the RTT to the farthest receiver: max_RTT, because of the request/response exchange.
- the upper bound a receiver can wait before answering to QUERY message (section 2.3): $T_{maxwait}$. Because of its importance, $T_{maxwait}$ has a fixed known value.

A possible value is therefore:

$$T_{drop} = RF * (max\_RTT + T_{maxwait}) \tag{1}$$

where $RF \geq 1$ is the Robustness Factor. Because max_RTT is in practice rather difficult to evaluate[2], we use a "reasonable" fixed value instead and use the following adaptive (multiplicative increase/additional decrease) scheme:

```
the source issues a DROP_LAYER message with a new identifier;
if (the source receives a  LAYER_REQ message with the same identifier
    within two max_RTT)
        // the drop_timer value was too short, so increase RF...
        RF = min(RF * constant1, MAXIMUM_RF_VALUE);
else
        // the drop_timer value was correct, so decrease RF...
        RF = max(RF - constant2, MINIMUM_RF_VALUE);
```

Another factor exists: the loss probability of QUERY, PRESENT and PRESENT_OK messages for a given receiver. Because the QUERY message is sent several times, because a PRESENT message lost may be replaced by a PRESENT message from another receiver, we do not take this factor specifically into account in equation (1). In case of high losses, we merely rely on the adaptive algorithm for the RF constant.

[2] Several reliable multicast protocols require the evaluation of the RTT to each receiver [14][2][12]. We try to avoid this extra complexity by proposing a simpler adaptive algorithm.

4.2.2 The SOFTSTATE Timer

Because transmissions are faster on higher layers than on lower ones, it seems natural to check the presence of receivers more frequently on higher layers, i.e. to use a smaller value for the `softstate_timer`: $T_{ss}(k)$. We can define a cost function for layer $k > 0$, $Cost(k)$, as the average number of packets sent between the time the last receiver leaves and the time when ODL detects that this layer is no longer used. Therefore:

$$Cost(k) = \frac{(T_{ss}(k) + T_{drop})}{2} * rate(k)$$

where $rate(k)$ is the transmission rate for this layer. We want to keep $Cost(k)$ below a fixed threshold for all layers: $Cost_1$. From the previous equation we find:

$$T_{ss}(k) = \frac{2 * Cost_1}{rate(k)} - T_{drop}$$

We don't want $T_{ss}(k)$ to become negative (which means that the target cost is not feasible for this layer), so we define a minimum value: min_T_{ss}. Therefore:

$$T_{ss}(k) = max(min\_T_{ss}; \frac{2 * Cost_1}{rate(k)} - T_{drop}) \quad (2)$$

Other parameters may affect the `softstate_timer`. In particular we can make it depend on the *frequency receivers leave a layer*, which in turn depends on two parameters:

- in case of a (continuous or not) file transfer application, the time required to retrieve the whole file: $T_{retrieve}(k, file\_size)$. Its value depends on the packet scheduling scheme, the layer k, and the file size. In case of data flows generated on-the-fly, this parameter is of course meaningless.
- the network dynamicity: In an unstable network where congestion quickly appears, receivers frequently join or leave a layer. In that case, the source should check frequently if the groups are still used or not.

If we neglect the second factor (difficult to evaluate and highly variable), we can introduce a correction to equation (2) as follows:

$$T_{retrieve}(k, file\_size) = \frac{file\_size}{c\_rate(k)}$$

$$T_{ss}(k, file\_size) = max(min\_T_{ss}; min(\frac{2 * Cost_1}{rate(k)} - T_{drop}; \frac{file\_size}{c\_rate(k)})) \quad (3)$$

With a power of 2 rate distribution (i.e. send 1 packet/tick on layer 0, 1 on layer 1, 2 on layer 2, 4 on layer 3, etc.): $rate(k) = 2^{k-1}$ and the cumulative rate for layers 0 to k is: $c\_rate(k) = 2^k$.

Finally it is meaningful to require that $T_{ss}(k)$ remains *below a given maximum time*, max_T_{ss}, especially on the lower layers. Indeed if we no longer consider the cost in terms of packets but in terms of resources required on each multicast router (section 4.3), then ODL should ensure that an unused multicast group can not be used longer than max_T_{ss} seconds. Thus:

$$T_{ss}(k) = min(max\_T_{ss}; max(min\_T_{ss}; \frac{2 * Cost_1}{rate(k)} - T_{drop})) \quad (4)$$

4.2.3 Relationship with IGMP

An important factor is the IGMP version in use at the receivers. With IGMPv1, a receiver leaving a group does not try to inform anybody. Therefore data sent to this group keeps flowing until the local multicast router discovers there is no receiver any more at the next `Membership QUERY`. The default query period, T_{IGMP_query}, is 125 seconds [11]. ODL can discover the departure of the last receiver before IGMPv1 if:

$$T_{ss}(k, file\_size) + T_{drop} < T_{IGMP\_query} \quad (5)$$

If inequation (5) is false, then ODL does not help to reduce the `LEAVE` latency of IGMPv1. This is an additional reason to define a max_T_{ss} parameter smaller than 125 seconds.

The situation is different with IGMPv2. Here the last local host leaving a group issues a `LEAVE` message to inform the local multicast router. This router then performs a `Membership QUERY`, and if nobody answers, sends a *PRUNE* message upwards. ODL cannot react faster than IGMPv2, especially if the fast group management mechanism [29] is implemented.

Note that backward compatibility issues lead to using IGMPv1 mechanisms if on the local network an IGMPv1 host or router exists [11]. Because of that, fast group management may not always be usable.

4.3 The Memory Cost of a Multicast Group with Dense-Mode Routing Protocols

In order to assess the *memory cost* of a group in each multicast router, we analyzed the `mrouted` (version 3.8) distribution. We found that a group leads to the allocation of (more than) 100 bytes of state information, no matter whether there are receivers beneath this router or not[3].

The dump of the forwarding cache table of a multicast router of the University of Paris 6 (1/6/00) indicated the presence of 195 entries for 28 multicast groups (January 6th, 2000) for a total of $28 * 88 + 195 * 12 = 4804$ bytes. If this is easily manageable, the multiplication of multi-layer multicast sessions (and more generally of multicast applications) and the common *wrong* idea that an unused multicast group has little associated cost may well make the situation become quickly serious. For instance let's suppose that 1000 sites (which is not a lot for the whole Internet) perform large scale continuous multicast disseminations, each of them using a fixed number of 5 layers. Then each multicast router within their scope will require: $1000 * 5 * (88 + 12) = 500,000$ bytes, *even if the router is not on the path to a receiver!* By using ODL only the layers having at least one receiver will be used at any time. Other potential layers are only deployed on-demand and do not incur any cost as long as they are not used.

Even if we only considered the DVMRP protocol here, per-group state is required with other protocols as well (section 1). In addition to the memory costs analyzed here, there is the cost of the flood-and-prune activity typical to dense-mode protocols. This cost will slowly disappear as sparse-mode protocols become more widely deployed, but in the meantime this is an additional reason to use ODL.

4.4 Experiments with ODL

We have carried out several experiments in order to assess:

- the overhead introduced by the ODL signalization,
- the validation of ODL timer values, and
- the gains, in terms of reduced data transmissions.

4.4.1 Test setup

The tests include a single source who performs a one-time file transfer, uses five layers and a cumulative GMMG scheduling scheme. Note that the results would be the same with any other cumulative scheme. The average cumulative throughput is respectively 5, 10, 20, 40 and 80 kbytes/s. A 400 kbytes file is submitted by the applications in 50 ADUs of 8 kbytes each. There are two receivers: a high-end receiver, R_1, who subscribes to all five layers, and a low-end receiver, R_2, who only subscribes to the first three layers. We assume a synchronous start so the situation is similar to that of Figure 1 (a). Three hosts connected to an Ethernet LAN are used, one for the source and the `tcpdump` tool, the two others for the receivers. The heterogeneity is simulated by specifying the maximum number of layers asked by each receiver.

4.4.2 Experimental Results

Figures 6 and 7 show the resulting traffic at the source respectively without and with ODL. The protocol parameters are the following: $Cost_1 = 40$ packets, $min\_T_{ss} = 0.2$ s, $max\_T_{ss} = 20$ s, $T_{maxwait} = 0.4$ s and $RF = 1$. The internal ODL clock frequency is 10Hz. In Figure 7, the vertical lines at time $0.17s$ (hardly visible) illustrate the addition of all five layers, and the remaining five vertical lines after time 11s the drop of each layer. The isolated points with small error bars illustrate the sending of a `QUERY` message on a layer.

[3]More precisely information is stored in a gtable structure (88 bytes), one per group, plus a list of stable structures (12 bytes each), one per multicast source, and a list of ptable structures (16 bytes each), one per prune message received from a downstream router.

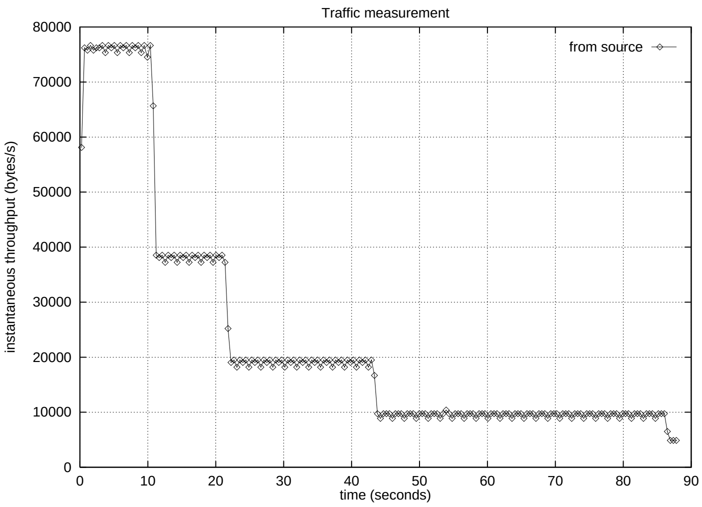


Figure 6: Traffic at the source without ODL.

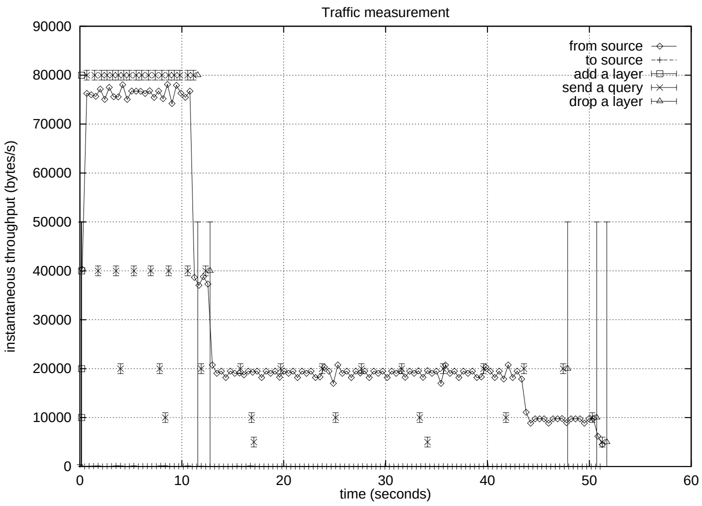


Figure 7: Traffic at the source with ODL.

We see that ODL periodic checks are efficient: the higher the level, the faster the receiver departure is detected and transmissions stopped. Layer 3 is dropped only a few seconds after that R_1 has left (at time 10.9s). An analysis of `tcpdump` traces show that only 64 data packets have been sent during this interval on layer 3, which remains in line with the $[0;\ 2 * Cost_1]$ target[4]. Layers 0 and 1 are also dropped far before their natural end after that R_2 has left (at time 43.5s). Here 65 and 74 packets are respectively sent on layers 1 and 0 after the departure of R_2. Once again it remains inferior to the $2 * Cost_1$ maximum.

Table 2: Detailed measures without and with ODL, $Cost_1 = 40$ packets.

	without ODL	with ODL
Total traffic from source (data+ODL, including UDP/IP hdr)	2,176,710 bytes, or 3,951 pkts	1,631,425 bytes, or 3,061 pkts
Total traffic to source (ODL replies, including UDP/IP hdr)	N/A	2,720 bytes, or 48 pkts
Total ODL overhead	N/A	8,776 bytes, or 0.54%
Gains made possible by ODL	N/A	25.1% (bytes), or 22.5% (pkts)

[4]Cost1 is the average cost, not the maximum cost.

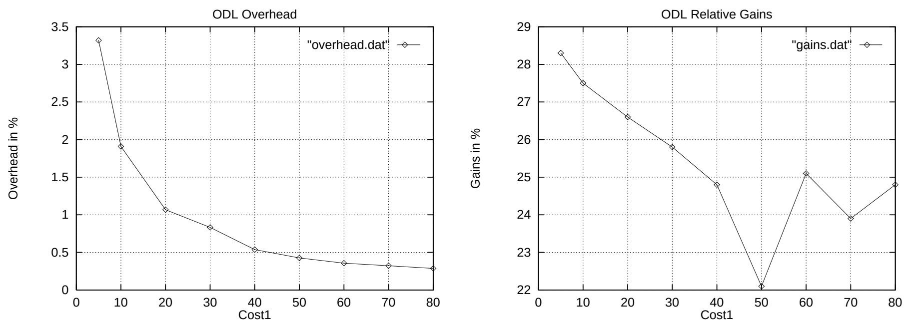
From table 2 we see that *ODL reduced by 5.1% the total number of bytes on the LAN and by 22.5% the total number of packets*. Of course this result greatly depends on the scenario in use: number and features of the set of receivers.

At the same time *ODL's overhead* (i.e. the number of bytes introduced for ODL signalization) *remains very low* as it only represents 0.54% of the total traffic (in bytes). This is largely due to the fact that ODL messages sent by the source are often piggy-backed into data packets[5]. On the reverse path a distinct ODL packet is sent each time. Yet the traffic remains rather low because of feedback cancellation techniques.

4.4.3 Impacts of the $Cost_1$ Parameter

We carried out the same experiments while varying the $Cost_1$ parameter in the [5; 80] interval. The overhead introduced by ODL signalization and the benefits of using ODL are shown in figures 8(a) and 8(b). As the $Cost_1$ parameter increases, the overhead goes asymptotically to zero since fewer queries/replies are generated. This overhead is anyway very small, below 1% of the total amount of data sent if $Cost_1 > 20$ packets. Having more than two receivers would create a slightly higher number of ODL replies, but it would be limited by ODL's feedback cancellation techniques.

The "gains" curve is more complex. As expected, in the [5; 50] interval the smaller the $Cost_1$ parameter, the faster the detection of unused layers, and the smaller the number of unused packets sent. But the results are more chaotic in the [60; 80] interval. The reason is the following: queries are sent every max_T_{ss} seconds on layer 0 when $Cost_1 \geq 60$ packets. It turns out that a query is sent on layer 0 *immediately after* the departure of receiver R_2. Therefore the source is immediately informed and does not have to wait an additional max_T_{ss} period which saves many transmissions (in fact layer dependencies (section 2.2) may delay a bit the effective drop of layer 0).



(a) Overhead (b) Gains

Figure 8: ODL's overhead versus gains as a function of the $Cost_1$ parameter.

4.4.4 Discussion of the results

We can draw the following conclusions:

- *the smaller the $Cost_1$ parameter, the higher the chances of a fast detection of unused layers.* But it remains *probabilistic*, and using longer query periods (i.e. high $Cost_1$) also increases the chances of unpredictable results.
- There is a *strong analogy with the signal sampling theory.* Unfortunately in our case the maximum "signal" frequency and thus the optimal sampling frequency are not known!
- Anyway the *gains made possible by ODL are significant*, even if they greatly depend on the configuration. On a LAN we observed an average 25% reduction in the number of bytes sent. With a WAN configuration (e.g. R_1 on a remote site) the local gains (reduced source/LAN/mrouter load) remain the same. In addition there can be some reduction of the number of packets received by the remote site (depending on the IGMP version), of the forwarding cache table, of the tree management traffic, etc.

[5]More precisely the first ODL header is sent immediately in a dedicated packet (total of 48 bytes, including the UDP/IP headers), and additional transmissions (for reliability) are piggy-backed in data packets and only add 8 bytes each.

5 Related Work

5.1 Dynamic source-oriented adaptation

In this paper we assumed that the decision to receive or not a layer was receiver-oriented, the source(s) having defined a fixed set of available layers. ODL is merely a feedback protocol that enables the source(s) to adapt the number of layers effectively used. An opposite approach is to enable the source(s) to adapt the features of each layer in real time.

[38] describes two variants of the SAMM (Source Adaptive Multi-layered Multicast) algorithm. The source adjusts dynamically the number (like ODL) and the individual rate of each layer of a layered video stream thanks to feedback information sent by the receivers (and possibly routers). This work differs from ODL from many respects: it mixes both the adaptation and congestion control functions, it requires that routers implement some form of priority-based packet discarding function, it requires the presence of feedback mergers, one variant also requires that routers monitor and report the available bandwidth, and finally the authors only consider the case of a single video source.

[18] describes a particular case of the DSG scheme that uses only two groups: a base group (B_group), and an improved group (V_group). The V_group transmission rate is determined by maximizing a receiver satisfaction metric called IRF (Inter-Receiver Fairness). The evaluation of IRF requires some feedback from the set of receivers. [19] extends this work by considering the original DSG scheme ($n > 2$ groups) but it relies on a homogeneous ATM networks offering an ABR service. Our approach is both more restricted in its goals (we only consider the use of each layer) and more general (it is not limited to a particular transmission scheme). Note that ODL is no longer required with the DSG/IRF mechanism.

ODL also shares some similarities with RTCP (RTP Control Protocol) [34]. If both protocols return feedback information to the source(s), RTCP goals are to “monitor the quality of service” and to “convey information about the participants in an on-going session. The latter aspect of RTCP may be sufficient for loosely controlled sessions [...] but it is not necessarily intended to support all of an application's control communication requirements”. ODL only addresses one particular issue not covered by RTCP, so both protocols can (will?) be used together as they bring separate benefits.

[16] gives a taxonomy of feedback protocols for multicast sessions. ODL is close to the “polled” category except that the feedback content is generic rather than individualized.

5.2 Membership size estimation

Several mechanisms have been proposed to estimate the size of a multicast session. [5] introduces a probabilistic polling method through several polling rounds, each with a higher reply probability. [24] describes a single polling round technique. [13] adds refinements to these works.

ODL shares some similarities with these works. Yet ODL does not care about the exact number of members and only returns a binary answer: “yes there is at least one receiver”, or “no there is nobody”. ODL is therefore much simpler.

In fact ODL can be associated with a protocol giving an estimation of the number of receivers (e.g. if it is already available because it is required by another building block). As long as this estimation is above a given threshold, ODL is useless. As soon as it falls below this threshold, there is a risk that the statistical approach is erroneous, and therefore ODL is enabled for a rapid and accurate behavior when the last receiver leaves.

5.3 State aggregation and administrative scoping

Another kind of solution, complementary to ODL, to improve IP multicast scalability is the aggregation of multicast forwarding state. [35] promotes a novel data structure to better aggregate ranges of multicast addresses within a router's forwarding table. The authors also note that an appropriate address allocation scheme would greatly help improving this aggregability.

Because the TTL (Time To Live) header field has limitations, there is some work on administrative scoping [15]. The MZAP (Multicast-Scope Zone Announcement Protocol) enables the definition of (hierarchical) multicast scope zones of any size to better control the dissemination of multicast traffic, which is another way to improve IP multicast scalability.

6 Conclusions and Future Work

We have introduced a new end-to-end protocol, ODL, that enables a source to be informed in real-time of the presence of receivers and of their reception possibilities. With this information the source generates a data flow that is really used by the current set of receivers.

ODL is *very beneficial to multi-layer multicast transmission schemes*, be they cumulative (e.g. a hierarchically coded video stream) or not (e.g. DSG). ODL avoids the use of groups with no receiver attached, whose cost is often under-estimated: state kept within multicast routers (e.g. with `DVMRP/mrouted` 100 bytes of state information are allocated per group in *all the* multicast routers), periodic tree management traffic (flood/prune of dense-mode protocols, `SA` messages of MSDP, etc.), and router load. This is of much importance to guaranty the *scalability of IP multicast* as it becomes more widely deployed and used. To avoid compromising this goal, the protocol is designed so that the feedback is always unicast to the source(s), rather than multicast.

ODL is *beneficial to router based filtering techniques* as well, another class of solutions to address the heterogeneity of receivers, even if a single multicast group is used in that case.

For a good behavior (timer values), ODL requires some information from the other components and the application: packet scheduling scheme, transmission rates, application type, file size. This is in line with the ILP (Integrated Layer Processing) paradigm [6].

We have implemented the ODL protocol, integrated it within the MCL multicast library [32], and made several experiments to assess its benefits. The corresponding building block is freely distributed on the author's home page.

Some aspects have not been covered by the tests. We can mention the impacts on the scheduling scheme efficiency, of the join latency when dynamically adding a layer. Loosing some initial packets can compromise an efficient reconstruction of ADUs. Using FEC [27] and delaying the data packet transmissions on a newly added group are two complementary solutions. Another item is the analysis of the feedback cancellation technique efficiency in front of a very high number of receivers. Complementary schemes (e.g. developed for RTP [33]) could be used. Finally we are currently working on a "reverse-IGMP" (local) solution where a source queries its first-hop router to get group membership information (he knows which group is pruned). We are studying its feasibility and scope; e.g. it requires new extensions to IGMPv3, it may also depend on the multicast routing protocol in use and whether the intra or inter-domain is considered, ODL is still required to enable a receiver to ask for an additional layer. Anyway we believe that both approaches are complementary. ODL is *always applicable* and *immediately* as it does not depend on any protocol deployment. This is another illustration of the end-to-end argument.

Acknowledgments

The author thanks Lorenzo Vicisano for his comments and the idea of "reverse-IGMP".

References

[1] K. Almeroth, "The evolution of multicast: from the MBone to inter-domain multicast to Internet2 deployment", *IEEE Network Special Issue on Multicasting, January/February 2000.*

[2] Adamson, Macker, "The Multicast Dissemination Protocol (MDP)", *Work in Progress, <draft-macker-rmt-mdp-00.txt>, October 1998.*

[3] M. Ammar, L. Wu, "Improving the Performance of Point to Multi-Point ARQ Protocols through Destination Set Splitting", *Proceedings of IEEE INFOCOM'92, May 1992.*

[4] S. Bhattacharyya, J. Kurose, D. Towsley, R. Nagarajan, "Efficient multicast flow control using multiple multicast groups", *Proceedings of IEEE INFOCOM'98, February 1998.*

[5] J. Bolot, T. Turletti, "Scalable feedback control for multicast video distribution in the Internet", *Proceedings SIGCOMM'94, September 1994.*

[6] D. Clark, D. Tennenhouse, "Architectural considerations for a new generation of protocols", *Proc. ACM SIGCOMM'90, September 1990.*

[7] A. Clerget, "A Tag-based UDP Multicast Flow Control Protocol", INRIA Technical Report 3728, July 1999.

[8] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, A. Helmy, D. Meyer, L. Wei, "Protocol Independent Multicast Version 2 Dense Mode Specification", *Work in Progress, <draft-ietf-pim-v2-dm-03.txt>, June 1999.*

[9] M. Donahoo, M. Ammar, E. Zegura, "Multiple-Channel Multicast scheduling for scalable bulk-data transport", *Proceedings of IEEE INFOCOM'99, March 1999.*

[10] D. Farinacci, Y. Rekhter, P. Lothberg, H. Kilmer, J. Hall, "Multicast Source Discovery Protocol (MSDP)", *Work in Progress, <draft-farinacci-msdp-*.txt, June 1998.*

[11] W. Fenner, "Internet Group Management Protocol, Version 2", *RFC 2236, November 1997.*

[12] S. Floyd, V. Jacobson, S. McCanne, C. Liu, L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing", *Proceedings ACM SIGCOMM'95, 1995.*

[13] T. Friedman, D. Towsley, "Multicast session membership size estimation", *Proceedings of IEEE INFOCOM'99, March 1999.*

[14] M. Handley, B. Whetten, R. Kermode, S. Floyd, L. Vicisano, "The Reliable Multicast Design Space for Bulk Data Transfer", *Work in Progress <draft-ietf-rmt-design-space-00>, June 1999.*

[15] M. Handley, D. Thaler, R. Kermode, "Multicast-Scope Zone Announcement Protocol (MZAP)", *Work in progress, <draft-ietf-mboned-mzap-06.txt>, December 1999.*

[16] M. Hofmann, J. Nonnenmacher, J. Rosenberg, H. Schulzrinne, Zone Announcement Protocol (MZAP)", *Work in progress, <draft-ietf-mboned-mzap-06.txt>, December 1999.*

[17] K. Hua, S. Sheu, "Skyscrapper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems", *Proceedings ACM SIGCOMM'97, September 1997.*

[18] T. Jiang, E. Zegura, M. Ammar, "Inter-Receiver fair multicast communication over the Internet", *Proceedings of NOSSDAV 99, June 1999.*

[19] T. Jiang, M. Ammar, E. Zegura, "On the use of Destination Set Grouping to improve Inter-Receiver Fairness for multicast ABR sessions", *Proceedings of IEEE INFOCOM'00, March 2000.*

[20] M. Luby, L. Vicisano, T. Speakman, "Heterogeneous multicast congestion control based on router packet filtering", *Work in Progress, presented at RMRG meeting, Pisa, June 1999.*

[21] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, J. Crowcroft, "Asynchronous Layered Coding; a scalable reliable multicast protocol", *Internet draft in preparation, <draft-ieft-ALC-v01-spec-rev-01>, reference to be completed when released...*

[22] S. McCanne, V. Jacobson, M. Vetterli, "Receiver-driven layered multicast", *Proceedings ACM SIGCOMM'96, October 1996.*

[23] J. Moy, "Multicast extensions to OSPF", *Request For Comments, RFC 1584, March 1994.*

[24] J. Nonnenmacher, E. Biersack, "Optimal multicast feedback", *Proceedings of IEEE INFOCOM'98, February 1998.*

[25] T. Pusateri, "Distance Vector Multicast Routing Protocol", *Work in Progress, <draft-ietf-idmr-dvmrp-v3-06.txt>, March 1998.*

[26] I. Rhee, S. Joshi, M. Lee, S. Muthukrishnan, V. Ozdemir, "Layered Multicast Recovery", *Technical Report TR-99-09, NCSU, Computer Science Dept., February 1999.*

[27] L. Rizzo, L. Vicisano, "Effective erasure codes for reliable computer communication protocols", *ACM Computer Communication Review, V.27, N.2, April 1997.*

[28] L. Rizzo, L. Vicisano, "Reliable Multicast Data Distribution Protocol based on software FEC techniques", *Fourth IEEE Workshop on the Architecture and Implementation of High Performance Communcaiotn Systems (HPCS'97), Greece, June 1997.*

[29] L. Rizzo, "Fast group management in IGMP", *Proceedings of HIPPARCH'98, London, 1998.*

[30] V. Roca, "Packet scheduling for heterogeneous multicast transmissions", *Protocols for High Speed Networks (PfHSN'99), August 1999.*

[31] V. Roca, "Analysis of Several Scheduling Algorithms for the Heterogeneous Multicast Distribution of Data Flows Generated on the Fly", *Work in Progress, available on the author's home page, October 1999.*

[32] V. Roca, "A library for heterogeneous multicast distributions: concepts, architecture and use", *Work in Progress, available on the author's home page, January 2000.*

[33] J. Rosenberg, H. Schulzrinne, "Sampling of the Group Membership in RTP", *Request for Comments, RFC 2762, February 2000.*

[34] Schulzrinne, Casner, Frederick, Jacobson, "RTP: A Transport Protocol for Real-Time Applications", *Work in Progress, <draft-ietf-avt-rtp-new-04.txt>, June 1999.*

[35] D. Thaler, M. Handley, "On the aggregatability of multicast forwarding state", *Proceedings of IEEE INFOCOM'00, March 2000.*

[36] L. Vicisano, "Notes on a cumulative layered organisation of data packets across multiple streams with different rates", *University College London Computer Science Research Note RN/98/25, Work in Progress, May 1998.*

[37] L. Vicisano, L. Rizzo, J. Crowcroft, "TCP-like congestion control for layered multicast data transfer", *Proceedings of IEEE INFOCOM'98, February 1998.*

[38] B. Vickers, C. Albuquerque, T. Suda, "Adaptive Multicast of Multi-Layered Video: Rate-Based and Credit-Based Approaches", *Proceedings of IEEE INFOCOM'98, February 1998.*

[39] "wb - LBNL Whiteboard Tool", *available at URL http://www-nrg.ee.lbl.gov/wb/*

[40] L. Wei, D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, " Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification", *Work in Progress, <draft-ietf-pim-v2-sm-01.txt>, November 1999.*

[41] B. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, "Reliable multicast transport building blocks for one-to-many bulk data tranfer", *Work in Progress, <draft-ietf-rmt-buildingblocks-00.txt>, June 1999.*

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399