Thèse de Doctorat de l'université Paris VI
Pierre et Marie CURIE

Spécialité

SYSTÈMES INFORMATIQUES

présentée par

M. Paulo André DA SILVA GONCALVES

pour obtenir le grade de

DOCTEUR de l'université Pierre et Marie CURIE

Communications de Groupes : Analyse et Performances des Mécanismes d'agrégation de l'information de Contrôle

Presentée et soutenue publiquement le 25 juin 2004 devant le jury composé de

M. Nazim AGOULMINE	**Rapporteur**
M. Ken CHEN	**Rapporteur**
Mlle. Dominique GAITI	**Examinateur**
M. Guy PUJOLLE	**Directeur de thèse**
M. Otto Carlos M. B. DUARTE	**Directeur de thèse**
M. José F. de REZENDE	**Directeur de thèse**
M. Khaldoun AL AGHA	**Invité**

Thèse de Doctorat de l'université Paris VI
Pierre et Marie CURIE

Spécialité

SYSTÈMES INFORMATIQUES

présentée par

M. Paulo André DA SILVA GONCALVES

pour obtenir le grade de

DOCTEUR de l'université Pierre et Marie CURIE

Communications de Groupes : Analyse et Performances des Mécanismes d'agrégation de l'information de Contrôle

Presentée et soutenue publiquement le 25 juin 2004 devant le jury composé de

M. Nazim AGOULMINE	**Rapporteur**
M. Ken CHEN	**Rapporteur**
Mlle. Dominique GAITI	**Examinateur**
M. Guy PUJOLLE	**Directeur de thèse**
M. Otto Carlos M. B. DUARTE	**Directeur de thèse**
M. José F. de REZENDE	**Directeur de thèse**
M. Khaldoun AL AGHA	**Invité**

25/06/2004

A mes parents.

Remerciements

Cette thèse intitulée "Communications de Groupes: Analyse et Performances des Mécanismes d'agrégation d'information de Contrôle", a été soutenue le 25 juin 2004 à l'Université Pierre et Marie Curie (Paris VI) devant le jury composé de M. Nazim AGOULMINE, M. Ken CHEN, Mlle. Dominique GAITI, M. Guy PUJOLLE, M. Otto Carlos M. B. DUARTE, M. José F. de REZENDE et M. Khaldoun Al AGHA. Je tiens à exprimer ici toute ma gratitude envers les membres du jury pour leur disponibilité. Plus particulièrement, je suis reconnaissant à:

M. Nazim AGOULMINE, Professeur à l'Université d'Evry Val d'Essone, et M. Ken CHEN, Professeur à l'Université de Paris XIII, d'avoir acceptés d'être rapporteurs de ma thèse.

M. Khaldoun Al AGHA, Maître de Conférences à l'Université de Paris XI, et Mlle. Dominique GAITI, Professeur à l'Université de Technologie de Troyes, qui ont eu l'amabilité de bien vouloir participer à ce jury.

M. Guy PUJOLLE, Professeur à l'Université Pierre et Marie Curie (Paris VI), M. Otto Carlos M. B. DUARTE, Professeur à l'Université Fédérale de Rio de Janeiro, et M. José F. de REZENDE, Professeur à l'Université Fédérale de Rio de Janeiro, qui ont dirigé mes travaux de recherche et m'ont soutenu pendant ces années.

Mes remerciements vont bien entendu à toutes les personnes qui m'ont permis d'achever ce travail, m'entourant de leurs conseils et de leur soutien en toute situations. J'adresse toute ma sympathie aux membres du LIP6.

Je voudrais aussi remercier la CAPES/Brésil pour le support financier de cette thèse.

Je remercie mes parents et mes proches qui m'ont toujours encouragé et soutenu. Enfin, je remercie Muriel Meneveau pour son tendre soutien.

Résumé

Qu'elles soient déjà existantes ou pressenties, les applications multiparties fondées sur un service de multidiffusion requièrent pour certaines le retour d'information des participants qu'elles mettent en rapport. Selon ces applications, les informations générées peuvent être utilisées par une source émettrice pour réaliser du contrôle de flux ou d'erreur, pour adapter (choisir) la méthode de codage, voire même organiser les données en couches cumulatives. Le nombre de participants mis en rapport pouvant atteindre l'ordre de plusieurs milliers, le volume des informations de contrôle retournées entraîne l'implosion de la source pour des sessions de grande taille. Parmi les solutions jusqu'à présent proposées pour répondre aux contraintes liées au facteur d'échelle, l'agrégation fournit un moyen efficient au regard de ces contraintes. L'agrégation consiste à résumer les informations de contrôle retournées par chacun des participants dans des messages uniques. Les informations sont résumées (dans ces messages) de façon à constituer les données d'entrée de nombreux algorithmes utilisés par les sources émettrices. Cela en prenant en considération une éventuelle contrainte temporelle du retour d'information.

La problématique des travaux présentés dans cette thèse est d'améliorer l'efficacité des entités responsables de l'agrégation. Nous traitons plus particulièrement de deux mécanismes : le premier est celui utilisé par un récepteur pour temporiser le retour de ses informations de contrôle. Le second mécanisme est utilisé pour réguler temporellement l'agrégation au niveau des entités qui en sont responsables. La méthode que nous proposons est générique et s'appuie uniquement sur la régulation des temporisateurs (systèmes de temporisation). Notre méthode a la particularité d'être indépendante du type de contenu agrégé et de fonctionner sur une éventuelle sélection de paquets à acheminer. Nous modélisons le système d'agrégation et nous donnons une description formelle de son comportement. Les performances de notre système sont évaluées au travers de simulations. Nous démontrons une efficacité améliorée lorsque un processus d'émission périodique d'informations de contrôle est associé à notre méthode d'agrégation temporelle. Cette thèse inclut également le développement d'un prototype de nœud d'agrégation, pour valider, à partir d'expérimentations, l'aptitude de notre méthode à fonctionner dans des réseaux réels.

Mots-clés :

Communications de groupes, agrégation, efficacité, performance

Abstract

Group applications often require feedback to be sent from group members to a common destination such as a multicast source. The source then uses the information to adjust its transmission rate, encoding scheme, number of layers, etc. The challenge, as the group size grows, is to avoid implosion while maintaining the timely nature of feedback. The problem has been addressed in several ways, including the use of feedback suppression, feedback rate control, and feedback aggregation mechanisms. Aggregation scales better than other implosion avoidance techniques. Moreover, it offers an efficient way to provide the source with summarized information that is useful as input into different algorithms.

In this work, we demonstrate how to increase the efficiency of an aggregator. We show how to tune two mechanisms: "feedback transmission", on the timer settings used by receivers, and "temporal merging", on the timer settings used by an aggregation node. Our approach is generic, relating purely to timer settings, and does not touch on "content merging", or the decision regarding which packets to forward. We provide a formal model of the aggregation system and a formal description of the system behavior as well. We present a simulation study of the aggregator system. We show that periodic feedback transmission coupled with a suitable timer setting at the aggregator increases efficiency. To further demonstrate the feasibility and suitability of our approach, we have implemented an application-level aggregator and performed some live Internet experiments.

Key Words:

Multicast, feedback aggregation, aggregation efficiency, performance

Table of contents

Chapter 1

Introduction

The phenomenal growth of the Internet is based on a simple and robust design philosophy mainly drawn in the 1970s, under the Internet research program of the Defense Advanced Research Projects Agency (DARPA) of US Department of Defense. However, the Internet architecture was not designed to accommodate easily newer and more sophisticated services. Current reality and changing requirements are eating away at the viability of the original Internet architecture. Numerous efforts are being undertaken to align the Internet better with current and future service requirements. Examples include Integrated Service (IntServ), Differentiated Service (DiffServ), IP Security (IPSEC), firewalls, mobile IP, Network Address Translator (NAT) devices, label switching, Virtual Private Networks (VPNs), Web caches, overlay and peer-to-peer networks, and multicast services.

1.1 Motivation and Problem Statement

Multicast communication – the one-to-many or many-to-many delivery of data – is a topic of paramount importance in the research community, among standard groups, and network service providers. The idea behind multicasting is simple; a multicast source has to transmit once instead of n times for n destinations. Packet replication is provided by the network. This largely reduces the number of packets within the network as compared with multiple unicasts. This feature makes the use of the multicast technology very attractive to distribute high-bandwidth media like video to a large number of destinations.

Although multicasting might seem simple, its deployment in the Internet is far behind expectations after more than a decade of important research and development efforts. Commercial and technical barriers have contributed to this scenario.

Many Internet Service Providers (ISPs) have historically not embraced the multicast technology because of weak demand for commercial applications, unknown traffic and routing implications, undefined billing models, and a necessity to maintain the stability of existing customer connections [1]. In addition, the deployment of multicast has not seen any commercial support because of a typical "chicken and egg" situation: application writers were not seen any multicast deployed; the ISPs were not seen any commercial multicast applications; and router vendors were not seen any multicast service demand from ISPs [2]. However, this situation has changed. The deployment of multicast has recently received commercial support from carriers, ISPs, and common operating systems. Much of this is due to an increase interesting in streaming applications for audio and video.

Technical barriers that have de-escalated the multicast deployment are related to the standard IP multicast and modifications in the router fabric. Multicast requires a great amount of state and complexity in both edge and core routers. The provision of multicast has implied changes in an infrastructure that had a lengthy history of only offering best-effort unicast services. This has put in evidence the belief of many in the Internet community that new services demand for some "intelligence" to be pushed into the network. The contribution of the standard IP multicast to de-escalate the multicast deployment is also significant. IP multicast is based on an open service model that leads to a complex architectural design. Nowadays, IP multicast is still plagued with concerns pertaining to membership management, scalability, network management, and support for higher layer functionality such as error, flow and congestion control.

Numerous efforts are being undertaken to make the provision of multicast less complex. Proposals to simplify multicast protocols are based on the fact that the majority of multicast groups have only a single source. In particular, protocols such as Source-Specific Multicast (SSM) [3] exploit this fact, which in turn eliminates the shared tree and rendezvous point complexity. In [1] the authors present results of a month-long multicast traffic measurement study. Their findings confirm that an overwhelming majority of groups are indeed single source; only 1.1 percent of all multicast groups observed during their study had multiple simultaneous sources.

The single biggest step recently in multicast routing and addressing also comes from the recognition that the demand for large-scale multicast is largely for one-to-many or single source. By ignoring the many-to-many case, certain aspects of multicast routing and addressing are easier solved. Express (Explicitly Requested Single Source Multicast) [4, 5] demonstrates these abilities. Express addresses are based on the multicast channel model. A multicast channel is identified by a tuple (S,E), where S is the source address and E is the group address. Using Internet Protocol version 4 (IPv4), the tuple is a 64-bit address. The source can select any lower 32 bits (any group address) for its channel and not collide with another. Express sources transmit to a channel and advertise that channel. Receivers learn about these channels and initiate an Express join. Routers propagate these joins directly toward the source, building a source rooted multicast forwarding tree. The Express model simplifies the complexity of multicast routing and the assignment of multicast addresses as well.

In addition to an easier assignment of multicast addresses and the provision of simpler multicast routing algorithms to structure distribution trees, there are other important issues one must consider to devise an operational and scalable multicast service. Group applications often require feedback to be sent from group members to a common destination such as a multicast source. The source then uses the information to adjust its transmission rate, encoding scheme, number of layers, etc. The challenge, as the group size grows, is to avoid implosion while maintaining the timely nature of feedback. This problem has been addressed in several ways because a single implosion avoidance solution is neither suitable nor necessary for all applications and scenarios. Applications often have different feedback requirements in terms of latency, importance of receivers, and model. The model can be positive acknowledgments (ACKs), negative acknowledgments (NAKs), reports, or hybrid approaches. Solutions for feedback implosion avoidance include the use of feedback suppression (e.g. for NAKs), feedback rate control (e.g. for reports), and feedback aggregation mechanisms (e.g. for ACKs, NAKs, and reports).

Hierarchical aggregation is the idea of using a feedback tree in which internal nodes (aggregators or feedback mergers) are responsible for merging feedback information sent from receivers back to the source. This merging consists of receiving information from numerous downstream nodes and then either selecting from or summarizing this information for retransmission upstream. Aggregation is gaining in importance with recent discussions about multicast congestion control [6, 7, 8], and research efforts in both reliable multicast [9, 10, 11, 12, 13, 14, 15, 16] and source-adaptive layered multicast schemes [17, 18, 19, 20, 21, 22, 23]. The recognition that aggregation will be essential to devise an operational and scalable multicast service is the main motivation of this thesis.

There has been significant previous research on NAK aggregation in the context of reliable multicast. However, very little was done regarding the aggregation of receiver reports [18, 21, 22, 23, 24, 25]. Typically, applications that rely on the reception of reports consider all receivers of equal importance (e.g. Real-time Transport Protocol (RTP) [26]). The aggregation of receivers' reports provides the source with summarized information that represents the whole group. This information is in turn used as input into different algorithms such as the organization of a multicast data stream into useful layers for layered coding schemes and the parametrization of a multicast Forward Error Control (FEC) algorithm. In order to make a proper summarization, it is necessary that an aggregator gets one report from its downstream neighbors before generating upstream feedback. This is difficult to achieve when the source needs to get continuously feedback in short time periods. Feedback transmission, which is the manner in which reports are generated, and temporal merging, which is the manner in which the aggregator decides when information is to be forwarded, affect the efficiency of the aggregation system. The question of how to improve efficiency is the major concern of this thesis. We show how to tune "feedback transmission", on the timer settings used by receivers, and "temporal merging", on the timer settings used by intermediate nodes. Our approach is generic, relating purely to timer settings, and does not touch on "content merging", or the decision regarding which packets to forward.

The quality of a merger scheme is also dependent upon the manner in which information is chosen or amalgamated. This thesis does not address this "*content merging*" problem, whose solutions depend to a great extent on the particular information being merged. This work is restricted to the generic, content-neutral questions related to scheduling the flow of information.

1.2 Contributions

We provide a formal model of the aggregation system and a formal description of the system behavior as well. We present a simulation study of the aggregator system. We show that periodic feedback transmission coupled with a suitable timer setting at the aggregation node increases efficiency. In addition, we improve on other techniques by allowing the deployment of feedback control schemes willing to improve the aggregator efficiency and to reduce network overhead as well.

1.3 Document Outline

The second chapter addresses related work and presents applications that benefit from summarized reports. In the third chapter, we develop a formal model of the aggregator system; we introduce evaluation metrics; we analyze the necessary condition for improving the aggregator efficiency; we provide a formal description of the system behavior; we study feedback transmission and timer settings at the aggregator. The third chapter ends with a simulation study of the aggregator efficiency. In chapter 4, we are interested in real world experiments. We conduct some live Internet experiments to further demonstrate the feasibility and suitability of our approach. In the final chapter, results are summarized.

1.4 Introduction en Français

La croissance phénoménale de l'Internet est basée sur une philosophie de conception simple et robuste qui a été principalement élaborée dans les années 70 dans le cadre du programme de recherche en projets avancés du DARPA (Defense Advanced Research Projects Agency). Cependant, l'architecture Internet n'a pas été conçue pour accommoder facilement des services nouveaux et sophistiqués. Les besoins pour mettre en place ces services mettent en cause la viabilité de l'architecture Internet originale. De ce fait, de nombreux efforts sont entrepris pour que l'Internet s'adapte plus facilement aux besoins courants et futurs des services. Comme exemple, nous pouvons donner le Service Intégré (IntServ), le Service Différencié (DiffServ), la Sécurité IP (IPSEC), les firewalls, la mobilité IP, les dispositifs de traduction d'adresse réseau (NAT), label switching, les Réseaux Privés Virtuels (VPNs), le cache Web, les réseaux overlay et pair à pair, et la communication de groupe.

La communication de groupe - la communication point à multipoint ou multipoint à multipoint - est un sujet d'importance primordiale pour la communauté de recherche, parmi les instances de régulation de l'Internet, et les fournisseurs d'accès Internet. L'idée du multipoint est simple ; une source multipoint doit transmettre une fois au lieu de n fois vers n destinations. Le réseau est responsable pour la duplication des paquets. Ceci réduit énormément le nombre de paquets dans le réseau en comparaison de plusieurs unicasts. Cette particularité fait du multipoint une technologie très attirante pour distribuer la vidéo haut débit à un grand nombre de destinations dans le monde.

Bien que le multipoint puisse sembler simple, son déploiement dans l'Internet est lointain derrière des expectatives après plus d'une décennie d'efforts de recherche et de développements très importants. Les entraves commerciales et techniques ont contribué à ce scénario.

De leur côté, les fournisseurs d'accès Internet n'ont pas historiquement embrassé le multipoint pour plusieurs raisons : la très faible demande pour des applications commerciales ; les implications sur le routage et le trafic n'ont pas été connues ; l'absence de modèles de facturation ; et le besoin de maintenir la stabilité de connexions de clients existants. En outre, le déploiement du multipoint n'a pas vu d'appui commercial à cause d'une impasse : les fournisseurs de logiciels ne voyaient pas de multipoint déployé ; les fournisseurs d'accès Internet ne voyaient pas de demandes pour des applications commerciales multipoint ; et les vendeurs de routeurs ne voyaient pas de demande de service multipoint par les fournisseurs d'accès Internet. Cependant, cette situation a changé. Le déploiement du multipoint a récemment reçu l'appui commercial des fournisseurs d'accès Internet et des principaux systèmes d'exploitation. Tout cela est dû à une augmentation importante de l'intérêt pour des applications d'audio et de vidéo sur l'Internet.

Les entraves techniques qui ont retardé le déploiement du multipoint sont liées à la norme IP multicast aussi bien qu'aux modifications qui étaient nécessaires dans les routeurs. Le multipoint nécessite le maintien d'un nombre important d'états et génère une forte complexité, aussi bien au niveau des routeurs de cœur qu'au niveau des routeurs de bordure. La mise en œuvre du multipoint a impliqué des changements sur une infrastructure connue pour n'offrir que des services unicast du type best effort. Cela a mis en évidence que des nouveaux services exigent une certaine “intelligence” placée dans le réseau. Actuellement, cette idée est encore dans l'esprit d'une partie de la communauté Internet. La contribution de la norme IP multicast pour retarder le déploiement du multipoint est aussi importante. Le IP multicast est basé sur un modèle de service ouvert qui mène à une conception architecturale complexe. Aujourd'hui, le IP multicast a toujours des aspects problématiques par rapport à la gestion de groupes, au passage à l'échelle, à la gestion du réseau, et au support aux couches supérieures pour faciliter le contrôle d'erreur, de flux et de congestion.

De nombreux efforts sont entrepris pour rendre la mise en oeuvre du multipoint moins complexe. Les propositions pour simplifier les protocoles multipoint sont basées sur le fait que la majorité de groupes multipoint n'a qu'une source active. Particulièrement, les protocoles comme le SSM (Source-Specific Multicast) exploitent ce fait qui élimine à son tour l'arbre partagé et la complexité pour établir un point de rendez-vous. Dans [1], les auteurs présentent les résultats d'une étude de mesure de trafic multipoint effectuée pendant un mois. Leurs découvertes confirment qu'une majorité écrasante de groupes a en effet une source ; seulement 1.1 pour cent de tous les groupes multipoint observés pendant leur étude avait des sources multiples simultanées.

Récemment, l'avancée la plus importante dans l'adressage et routage multipoint vient aussi du fait que la demande pour des services multipoint est en grande partie pour des applications avec une seule source. En ignorant le cas "multipoint à multipoint", il y a certains problèmes de routage et d'adressage qui sont plus facilement résolus. L'EXPRESS (Explicitly Requested Single Source Multicast) démontre ces capacités. Les adresses EXPRESS sont basées sur le modèle de canal multipoint. Un canal multipoint est identifié par un tuple (S, E), où S est l'adresse source et E est l'adresse de groupe. En employant l'IPv4 (la version 4 du Protocole Internet), le tuple est une adresse à 64 bits. La source peut choisir n'importe quel 32 bits inférieur (n'importe quelle adresse de groupe) pour son canal de transmission et ne pas entrer en collision avec un autre. Des sources EXPRESS transmettent sur un canal connu par les récepteurs qui commencent un EXPRESS join. Les routeurs propagent les messages EXPRESS join directement vers la source, construisant un arbre multipoint enraciné dans celle-ci. Le modèle EXPRESS simplifie la complexité du routage multipoint aussi bien que l'attribution des adresses multipoint.

L'attribution plus facile d'adresses multipoint et le développement d'algorithmes de routage multipoint plus simples sont importantes pour rendre un service multipoint opérationnel. Néanmoins, il y a encore d'autres facteurs également importants qu'il faut considérer pour que service passe à l'échelle. Qu'elles soient déjà existantes ou pressenties, les applications multiparties fondées sur un service de multidiffusion requièrent pour certaines le retour d'information des participants qu'elles mettent en rapport. Selon ces applications, les informations générées peuvent être utilisées par une source émettrice pour réaliser du contrôle de flux ou d'erreur, pour adapter (choisir) la méthode de codage, voire même organiser les données en couches cumulatives. Le nombre de participants mis en rapport pouvant atteindre l'ordre de plusieurs milliers, le volume des informations de contrôle retournées entraîne l'implosion de la source pour des sessions de grande taille. Jusqu'à présent, plusieurs solutions ont été proposées pour répondre aux contraintes liées au facteur d'échelle. De différentes solutions sont nécessaires car il n'y a pas de solution unique qui soit au même temps nécessaire et appropriée à toutes les applications et scénarios. Les applications ont souvent de contraintes temporelles pour le retour d'information de récepteurs ; de différents besoins par rapport au type d'information retournée (e.g. l'acquittement (ACK), l'acquittement négatif (NAK), le "rapport") et à l'importance de chaque récepteur. Les solutions actuelles pour répondre aux contraintes liées au facteur d'échelle incluent la suppression de messages (e.g. pour les NAKs), le contrôle du débit génération de messages (e.g. pour les rapports) et l'agrégation de messages (e.g. pour les ACKs, NAKs, et rapports).

L'agrégation hiérarchique est l'idée d'utiliser un arbre dans lequel les nœuds internes (nœuds d'agrégation) sont responsables pour résumer les informations retournées par chacun des participants dans de messages uniques. Les nœuds d'agrégation peuvent également faire une éventuelle sélection de paquets à acheminer. L'importance de l'agrégation a été mise en évidence récemment a partir de discussions sur le contrôle de congestion multipoint et d'efforts de recherche dans les domaines du multipoint fiable et de l'adaptation de la source vidéo en couches. La vision de que l'agrégation sera essentielle pour rendre opérationnel un service multipoint qui répond aux contraintes liées au facteur d'échelle, est la motivation principale de cette thèse.

Il y a eu beaucoup de recherche concernant l'agrégation de NAKs dans le contexte du multipoint fiable. Cependant, très a peu été fait quant à l'agrégation des messages du type "rapport". Typiquement les applications qui comptent sur la réception de ces messages considèrent tous les participants du groupe de même importance (e.g. Real-time Transport Protocol (RTP)). L'agrégation de rapports des participants permet la source d'avoir un résume d'informations du groupe entier. Ce résumé à son tour sert comme donnée d'entrée de nombreux algorithmes utilisés par les sources émettrices multipoint. Comme exemple, nous pouvons donner le calcul des débits des couches vidéo et le paramétrage d'algorithmes de FEC (Forward Error Correction). Afin d'obtenir un résumé cohérent d'informations de retour, un nœud d'agrégation doit recevoir l'information de contrôle de tous les participants dont il est responsable. Cette approche devient difficilement réalisable de façon efficace par rapport aux règles d'agrégation temporelle si la fréquence d'arrivée de données de retour à un nœud d'agrégation est élevée.

La problématique des travaux présentés dans cette thèse est d'améliorer l'efficacité des entités responsables de l'agrégation. Nous traitons plus particulièrement de deux mécanismes : le premier est celui utilisé par un récepteur pour temporiser le retour de ses informations de contrôle. Le second mécanisme est utilisé pour réguler temporellement l'agrégation au niveau des entités qui en sont responsables. La méthode que nous proposons est générique et s'appuie uniquement sur la régulation des temporisateurs (systèmes de temporisation). Notre méthode a la particularité d'être indépendante du type de contenu agrégé et de fonctionner sur une éventuelle sélection de paquets à acheminer. Nous modélisons le système d'agrégation et nous donnons une description formelle de son comportement. Les performances de notre système sont évaluées au travers de simulations. Nous démontrons une efficacité améliorée lorsque un processus d'émission périodique d'informations de contrôle est associé à notre méthode d'agrégation temporelle. Cette thèse inclut également le développement d'un prototype de nœud d'agrégation, pour valider, à partir d'expérimentations, l'aptitude de notre méthode à fonctionner dans des réseaux réels.

Chapter 2

Related Work

The idea of deploying in-network nodes capable of merging information sent by the receivers has been used in different contexts, most notably in the context of reliable multicast and source-adaptive layered multicast schemes. Recently, there has been a proliferation of multicast congestion control schemes that consider the use of in-network feedback aggregation. This has been motivated by the recognition that aggregation is beneficial for many applications and provides a powerful way to devise scalable multicast protocols. This chapter gives an overview of feedback scaling mechanisms and the different research areas that exploit aggregation techniques.

2.1 Feedback Scaling Mechanisms

Group applications often require feedback to be sent from group members to a common destination such as a multicast source. Typical feedback messages include positive acknowledgments (ACKs), negative acknowledgments (NAKs) and "reports". ACKs are used to confirm every packet received. NAKs are used to inform the source about which packets were detected to be lost. Reports convey different information about receivers. Examples include, experienced jitter, supported reception rate, and observed loss rate.

The challenge, as the group size grows, is to ensure that the source continuously gets feedback from the group without being overwhelmed by an excessive number of feedback messages. The problem has been addressed in several ways because a single solution is neither suitable nor necessary for all applications and scenarios. Solutions include the use of polling (e.g. for reports), feedback suppression (e.g. for NAKs), feedback rate control (e.g.

for reports), and feedback aggregation mechanisms (e.g. for ACKs, NAKs, and reports).

2.1.1 Polling

Polling was the first mechanism devised to address the feedback implosion problem. Examples include probabilistic querying, querying with randomly delayed responses, and an expanding scoped search [27]. A number of existing solutions is based on hybrid approaches [28, 29, 30]

Probabilistic querying

The probabilistic querying scheme is quite simple. On sending a feedback request to the receivers, the source sets a timer to wait for responses. A receiver responds to the request with a given probability. If no reply is received before the timer goes off, the source then sends another request. The probabilistic scheme is simple to implement but its simplicity is not priceless. Although replying probabilities can be biased in accordance with the importance of the receiver, it is not possible to ensure that the source will get feedback from an intended receiver (*e.g.* the receiver experimenting the worst-case performance). Moreover, it is difficult to set suitable replying probabilities when the group size is not known.

Randomly delayed responses

In this scheme, each receiver delays the answer to a request by a random amount of time. The difficult of this scheme is the choice of the random delay. If delay is too short, feedback implosion can occur. Suitable random delays can be obtained using some knowledge about the group size. The scheme is specially interesting because it allows the source to get responses from all receivers in the multicast group.

Expanding scoped search

The expanding scoped search scheme gradually increases the Time-To-Live (TTL) of the requests sent by the source. In this manner, requests go further and further through the branches of the multicast tree. This scheme does not scale well when a large number of receivers is concentrated at a fixed point of the multicast tree.

2.1.2 Feedback Suppression

Feedback suppression has been examined by several research efforts in the multicast area (e.g., [31, 30, 32, 29]). Suppression mechanisms are used when feedback information is not specific to the receiver which sends the feedback. An example of such a feedback is a NAK. When the receiver sends a NAK by multicast, all other receivers will see the NAK and so do not need to send a NAK if they have lost the same packets. Typically, suppression mechanisms work as follows: time is divided into feedback rounds, which are either implicitly or explicitly indicated to the receivers. At the start of each feedback round, each receiver sets a randomized timer to delay feedback generation. This delay is typically related to both the group size and the one-way delay from the receivers to the source. If the receiver hears feedback from another receiver so that it is unnecessary for it to send its own feedback, it cancels its timer. Otherwise, when the timer goes off, a feedback message is sent.

When the many-to-many delivery of information is not available, feedback suppression mechanisms can still work if receivers unicast feedback to the sender that in turn echoes received information to the group. In such an approach, the amount of time the receivers delay feedback generation is related to the Round-Trip Time (RTT) instead of the one-way delay to the source.

ACK messages generally have receiver-specific information and so suppression mechanisms cannot be employed. Aggregation can be used to merger multiple ACKs into one so that message implosion is unlikely.

2.1.3 Feedback Rate Control

The Real-time Transport Protocol (RTP) [33, 34] provides end-to-end delivery services for data with real-time characteristics, such as audio and video, over unicast or multicast network services. Nowadays, large multimedia sessions over the Internet typically use RTP. RTP runs on the top of UDP (User Datagram Protocol) to take advantage of its multiplexing and checksum services. RTP itself does not provide any mechanism to ensure timely delivery or provide other quality-of-service guarantees, but relies on lower-layer services to do so. It does not guarantee delivery or prevent out-of-order delivery, nor does it assume that the underlying network is reliable and delivers packets in sequence. The sequence numbers included in RTP allow the receiver to reconstruct the sender's packet sequence.

The RTP control protocol (RTCP) is used to monitor the Quality of Service (QoS) and to convey information about the participants in an on-going session. The RTPC protocol defines feedback messages as "reports". They are periodically sent by all the session's participants and carry statistics that can be useful to the application. These statistics include interarrival jitter and fraction of packet lost.

RTCP includes a scaling mechanism to avoid feedback implosion in multicast environments. RTCP changes the rate at which a participant sends reports onto the multicast channel as a function of the number of participants in the session. Since the participants send reports to everyone, each participant can estimate the group size [35].

The RTP specification recommends that RTCP traffic be limited to 5% of the overall session bandwidth. Of this 5%, 75% is available to the receivers and the remaining 25% is allocated to the data senders. The period for transmitting RTCP packets for a sender is

$$T = \frac{\text{number of senders}}{0.25 \ \cdot \ 0.05 \ \cdot \ \text{session bandwidth}} (\text{avg. RTCP packet size}). \tag{2.1}$$

The period for transmitting RTCP packets for a receiver is

$$T = \frac{\text{number of receivers}}{0.75 \ \cdot \ 0.05 \ \cdot \ \text{session bandwidth}} (\text{avg. RTCP packet size}). \tag{2.2}$$

The interval between RTCP packets is varied randomly over the range $[0.5, 1.5]$ times the calculated interval to avoid unintended synchronization of all participants [36]. The minimum sending interval of reports is 5 seconds. A reporting interval of 5 seconds and beyond, most likely will severely hamper performance for applications that use RTP. For instance, although all transport protocols used on the Internet need to address congestion control in some way [37], RTP does not include any mechanism to treat congestion. A multicast congestion control based on the reception of RTCP reports will react to congestion over very long time periods.

2.1.4 Feedback Aggregation

Hierarchical aggregation is the idea of using a feedback tree in which internal nodes are responsible for merging feedback information sent from receivers back to the source. This merging consists of receiving information from numerous downstream nodes and then either selecting from or summarizing this information for retransmission upstream. Feedback aggregation largely solves the feedback implosion problem. Moreover, it offers to applications the possibility to benefit from summarized information such as the distribution of the receivers' feedback data.

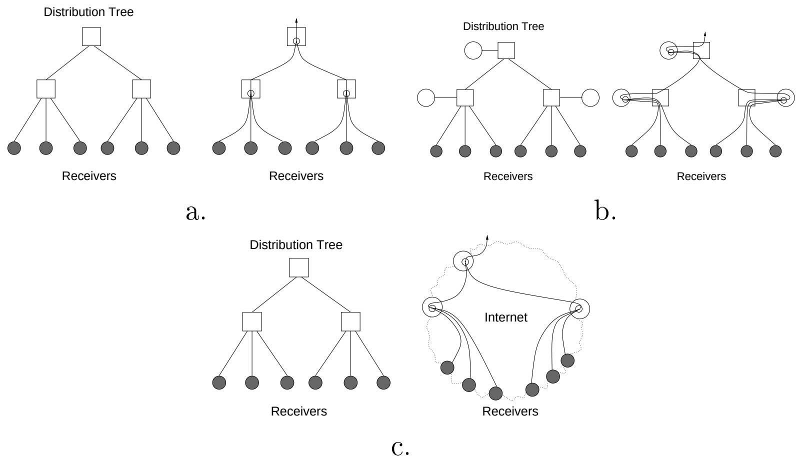


Figure 2.1: Design choices to deploy feedback mergers.

Deployment of Feedback Mergers

Feedback aggregation services can be available through routers or specialized servers. These routers or servers are referred as *feedback mergers* or *aggregators*. Fig. 2.1 shows three design choices for deploying feedback mergers. Fig. 2.1(a) shows the first design choice where routers along the multicast distribution tree are capable of aggregating feedback packets. This design choice suffers the same well-known problems of active networks [38, 39]. Fig. 2.1(b) presents another possibility to deploy feedback mergers. In this case, feedback mergers are co-located with routers along the multicast distribution tree. Servers that are co-located with routers are commonly employed in reliable multicast architectures. Implementation problems that are related to the first design choice is reduced since only minimal router support is required. Fig. 2.1(c) shows the third design choice where feedback mergers are deployed at end systems, and therefore, without requiring changes to existing routers.

Feedback Mergers and Active Networks

Our view is that aggregation nodes are of considerable potential importance for multicast applications but we do not believe that aggregation nodes will be deployed in a basis as proposed in the active network paradigm. Active networks emphasize generality. The deployment of active networks aim at providing a sufficiently flexible platform to solve a great number of problems in today's networks. However, to date, it is not clear whether or how such a platform with the desired level of security, performance and scalability can be engineered or deployed. We believe that the emergence of specialized nodes (maybe proprietary nodes) in the public Internet and the deployment of end-system aggregation nodes provide a viable and attractive alternative for making aggregation available.

2.2 RTCP extensions

The current trend to deploy single-source multicast protocols introduces a problem to RTCP since its scaling mechanism relies upon the many-to-many delivery of information. In [24] extensions to the RTCP protocol are proposed to restore the semantics of RTCP when the many-to-many delivery of information is not available. The authors propose the use of three mechanisms: reflection, summarization, and hierarchical aggregation. *Reflection* is used by the source to allow group members to communicate with each other. Receivers unicast reports to the source, which in turn sends onto the multicast channel any feedback message received. To improve efficiency, *summarization* might be used by the source. In this case, instead of reflecting every received report, the source only send onto the multicast channel summarized information such as the distribution of feedback data. Hierarchical aggregation can be also used to improve scalability. These propositions are currently under consideration in the Internet Engineering Task Force (IETF) as permanent enhancements to RTCP [25].

2.3 Reliable Multicast

Feedback from the multicast group is important to ensure reliable transmission from the sender. To achieve reliability, receivers can use either ACK messages to confirm every packet received or NAK messages to only inform the sender about which packets were detected to be lost. An ACK-based approach is not scalable and can incur a message implosion. A NAK-based approach is preferred by reliable multicast protocols because the number of NAKs is likely to be much less than the number of ACKs, resulting in a more scalable protocol. Even though, a NAK-based approach is prone to message implosion too. This is possible when group size is large and loss probabilities are high. Most of the known reliable multicast protocols treat the NAK implosion problem by either suppressing NAKs using random wait [31, 16] or aggregating NAKs using hierarchies [40, 10, 41, 42, 12, 15, 43].

2.4 Multicast Congestion Control

Congestion is the state of sustained network overload where the demand for network resources (i.e. bandwidth and buffer space in the routers) is close to or exceeds capacity [44]. The most widely used protocol over the Internet, the Transmission Control Protocol (TCP) [45], provides a control mechanism to avoid the congestion collapse. TCP congestion control halves the sending rate in response to congestion. While this has no importance to bulk data transfer, real-time applications can be severely affect as it can noticeable reduce the user-received quality [46]. In addition, TCP ensures data transfer reliability, which is not suitable for real-time applications. To avoid these issues, applications with real-time requirements have been running either on the top of RTP or directly on the top of UDP. However, these protocols do not address congestion control at all and can seriously affect the Internet stability as real-time applications become widely used.

This issue has motivated numerous work on congestion control for non-TCP unicast applications [6, 47, 46, 48, 49, 50, 51]. Their focus is on performing congestion control in a "TCP-friendly" manner. The rate is computed according to some formula such as the TCP equilibrium equation, which relates throughput to the loss and RTT (these schemes are called "equation-based"). Equation-based congestion control protocols, such as TCP-Friendly Rate Control (TFRC) [6], are more appropriate for applications that need to maintain a slowly-changing sending rate, while being responsive to network congestion over longer periods of time. These time periods are on the order of seconds instead of fractions of seconds.

Recently there has been a proliferation of proposals for multicast congestion control (e.g. [52, 53, 6, 7, 8]). The most widely used approach is one in which a multicast source regulates its rate in response to congestion feedback from its receivers. An important issue in designing source-based congestion control mechanisms is to get feedback from receivers in a scalable manner, and then combine them into a single rate decision at the source. Moreover, the rate must be regulated so that it ensures "fair" sharing of bandwidth among competing multicast and unicast sessions [41].

The TCP-Friendly Multicast Congestion Control protocol (TFMCC) [7] belongs to the class of single rate congestion control mechanisms. In TFMCC, each receiver tracks the RTT and the smoothed loss rate, and feeds these values into the TCP equilibrium equation. The results are communicated to the sender using exponentially weighted random timers and biasing feedback in favor of low-rate receivers.

PGMCC [8] also belongs to the class of single rate congestion control mechanisms. It selects the receiver with the worst network conditions as the group representative, called the *acker*. The selection process for the acker determines the fairness of the protocol, and is based on a simplified version of the TCP equilibrium equation. The other receivers send NAKs to the source. Similar to TFMCC, each receiver tracks the RTT and the smoothed loss rate. These values are used as input into the equation, and the results are embedded into ACKs, if receiver is the acker, or NAKs, otherwise. Scalability is achieved by using NAK suppression via randomization and the selection of a single node in charge of sending ACKs. If available, PGMCC also makes use of in-network aggregation. PGMCC is suitable for applications that can cope with larger variations in the sending rate. On the other hand, TFMCC is well suited to applications with more constraints on acceptable rate changes.

Several schemes have been suggested in the literature to support congestion control for layered data [54, 19, 55, 18, 56]. All of them are generally based on estimates done by the receivers on the available data rate. McCanne *et al.* [54] pioneered the study of multicast layered video transmission. They propose the transmission of video layers on separate multicast addresses. Based on congestion control information, a receiver subscribes to the number of layers that can be supported by its individual path from the source. Li *et al.* [19] propose a hierarchical rate control algorithm to manage the adding and dropping of video layers by receivers and an error recovery mechanism using retransmissions to adapt to network congestion. One limitation of these schemes is that receivers are limited to the layers that the source is willing to transmit. Current layered schemes rely on feedback from the multicast group to perform quality adaptation at the source.

Vickers *et al.* [18, 56] propose a source-adaptive layered multicast algorithm in which the transmission of each receiver report is triggered by the reception of a control packet sent by the source. Control packets are transmitted once for every R video packets transmitted, where R is assumed to be a small number such as 15. This assumption in practice leads to excessive control traffic. Scalability is achieved using hierarchical aggregation of reports. Feedback mergers perform content merging to estimate the rate of the video layers. This content merging is based on a combined "goodput" of video traffic received by all the receivers. A proper content merging is achieved when a feedback merger gets one report from each of its downstream neighbor nodes. Neither analysis nor results regarding the merging efficiency of feedback mergers are provided. Also, no justifications about the choice of the tunable parameter R and the timer setting at the merger nodes are presented.

The single-loop packet merging scheme proposed in [22] focus on the content-merging problem. Feedback transmission and temporal merging are similar to the approach in [18]. Analysis is based on the assumption that a merger node performs each merging procedure using one report from each of the its downstream neighbors. The proposed scheme for adapting the quality of layered video shows improvements in the fairness of the delivered service. In [23] it is demonstrated that content merging based solely on the proposed "goodput" quality metric in [18] fails to compute the best set of rates for the layers. In [57] is proposed a content-merging algorithm that considers the density of satisfied users, the weighted bandwidth allocation, and the "goodput" to the receivers. The results show enhancements in the quality and fairness of layered multicast sessions. Feedback transmission and temporal merging are also similar to the approach in [18] but for analysis purposes, each merging procedure is assumed to use one report from each of the downstream neighbors of a merger node.

2.5 Report Aggregation: Applications Uses and Scenarios

One advantage of feedback aggregation is the possibility to collect quickly data from all participants of a multicast group without incurring feedback implosion. Thus, an accurate estimate of the distribution of data from the receivers can be easily built. The distribution of data is useful as input into different algorithms. Following we present some applications that benefit from having access to the distribution of receivers' data.

- The organization of a multicast data stream into useful layers for layered coding schemes:

  The use of cumulatively layered or hierarchical information organization has long been an attractive idea for data delivery. The distribution of packet losses and delay would help to identify what percentage of participants experience various loss and delay levels, and thus how the data streams bandwidth might be divided to suit the group conditions.

- The parameterization of a multicast Forward Error Control (FEC) algorithm:

  An accurate estimate of the distribution of reported losses would help a source, which does not have a global view, to tune the degree of redundancy built in the FEC stream. The distribution helps to identify whether the majority of the group is experiencing high levels of loss, or whether in fact the high loss reports are only from a small subset of the group. Similarly, this information if sent by the source might enable a receiver to make a more informed decision about whether it should leave a group that includes a very high percentage of the worst-case reporters.

- Multicast congestion control and group management:

  A single-rate congestion control protocol that adapts to the slowest receiver may be improper to a large group of heterogeneous receivers. The reason the sending rate may be very low with respect to the bandwidth availability of the majority of group members. In order to enable applications to remove receivers that unduly impair protocol performance, it is necessary to provide the application with information about the degree of heterogeneity of the group. This information can also be used as input into the congestion control algorithm. An approximate distribution of TCP-friendly rates of the receivers, for instance, allows the calculation of the expect improvement in the sending rate when certain receivers are removed from the group. Appropriate actions can in turn take place.

- Multicasting of summarized feedback:

  Report rate control mechanisms such as the used by RTCP can still work properly in single-source environments. For this purpose, receivers must unicast feedback packets to the source, which in turn echoes every received feedback to the group. To improve efficiency, instead of echoing every feedback, the source can send on the multicast channel a summary of feedback data. This summary would be the receivers' data distribution, which would easily provide group information available for all members. When hierarchical aggregation is available, incoming feedback at the source are already summarized. So, the source only needs to echo the received information to the multicast group. The above-mentioned approaches to provide group information available for all members can also benefit other applications running in single-source environments.

2.6 Chapter Summary

In this chapter, we have presented feedback scaling mechanisms and different research areas that exploit aggregation techniques. There has been significant previous research on NAK aggregation in the context of reliable multicast. However, very little was done regarding the aggregation of receiver reports. Typically, applications that rely on the reception of reports consider all receivers of equal importance. The aggregation of receivers' reports provides the source with summarized information that represents the whole group. This information is in turn used as input into different algorithms such as the organization of a multicast data stream into useful layers for layered coding schemes, the rate estimation of video layers for source-adaptive layered video multicast, and the parametrization of a multicast FEC algorithm. In order to make a proper summarization, it is necessary that an aggregator gets one report from its downstream neighbors before generating upstream feedback.

2.7 Résumé du Chapitre

Dans ce chapitre, nous avons présenté les mécanismes de contrôle du retour d'information proposés pour répondre au facteur d'échelle pour les sessions de grande taille. Ce chapitre a offert également un panorama des domaines de recherche qui exploitent les techniques d'agrégation. Ensuite, les applications et les scénarios pour l'agrégation de messages du type "rapport" ont été présentés. Il y a eu beaucoup de recherche concernant l'agrégation de NAKs dans le contexte du multipoint fiable. Cependant, très a peu été fait quant à l'agrégation des messages du type "rapport". Typiquement les applications qui comptent sur la réception de ces messages considèrent tous les participants du groupe de même importance (e.g. Real-time Transport Protocol (RTP)). L'agrégation de rapports des participants permet la source d'avoir un résume d'informations du groupe entier. Ce résumé à son tour sert comme donnée d'entrée de nombreux algorithmes utilisés par les sources émettrices multipoint. Comme exemple, nous pouvons donner le calcul des débits des couches vidéo et le paramétrage d'algorithmes de FEC (Forward Error Correction). Afin d'obtenir un résumé cohérent d'informations de retour, un nœud d'agrégation doit recevoir l'information de contrôle de tous les participants dont il est responsable. Cette approche devient difficilement réalisable de façon efficace par rapport aux règles d'agrégation temporelle si la fréquence d'arrivée de données de retour à un nœud d'agrégation est élevée.

Chapter 3

Aggregator System: Model, Analysis, and Simulations

This work has been inspired by several efforts that exploit report aggregation. Aggregation offers an efficient way to provide the source continuously with summarized information that is useful as input into different algorithms. In order to make a proper summarization, it is necessary that an aggregator gets one report from its downstream neighbors before generating upstream feedback. This is difficult to achieve when the source needs to get continuously feedback in short time periods. Feedback transmission, which is the manner in which reports are generated, and temporal merging, which is the manner in which the aggregator decides when information is to be forwarded, affect the efficiency of the aggregation system. The question of how to improve efficiency is the major concern of this thesis.

3.1 Temporal Merging and Efficiency

Our objective in this section is the formalization of temporal merging and the definition of efficiency. Consider the basic network topology as shown in Figure 3.1. The merger node waits for feedback packets from k receivers. Temporal merging is performed as follows: incoming feedback packets are merged and the resulting packet is sent upstream whenever one of the subsequent two *conditions* holds.

- At least one packet from each of the nodes has arrived. We refer to this as the "*packet condition*".

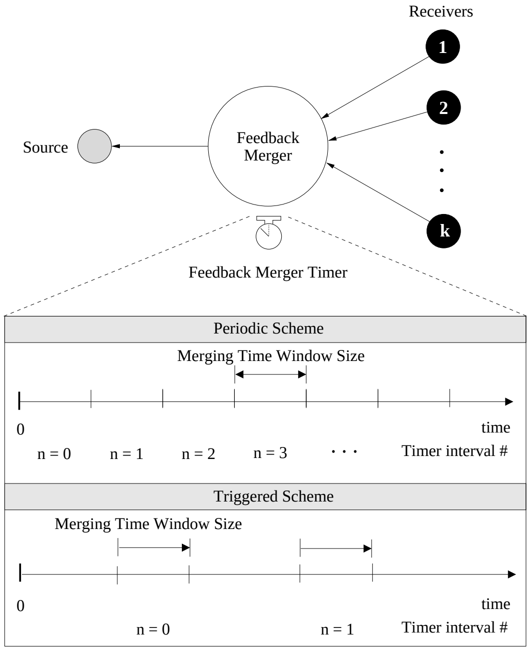


Figure 3.1: Basic network topology and merging timer schematics.

- The merger timer has expired. We refer to this as the "*timeout condition*".

In addition to these conditions, the following advice is applicable: if, prior to merging, more than one packet from the same node arrives, then the previously stored packet is replaced.

The above-mentioned temporal merging is similarly used in research efforts on source-adaptive layered multicast schemes [18, 22, 23, 57]. In addition, it is specially interesting because of the following rationales. *Packet condition* ensures that the content-merging algorithm is aware of feedback from each of the k nodes, satisfying a notion of fairness - each node has an equal right to contribute with information to an aggregation procedure. *Timeout condition* relates to the maximum waiting time required for *packet condition* to hold. The advice of packet replacements favours the most recent received information. However, they entail wasted resources; information previously transmitted is discarded.

An aggregator is inefficient if it generates upstream feedback before having received information from all of its downstream neighbors. It is also inefficient if, prior to merging, it receives multiple reports from the same downstream neighbor. To improve efficiency, we need to reduce packet replacements and aggregations due to *timeout condition* as well. Ultimately, this problem consists of finding a feedback transmission mechanism that, coupled with suitable timer settings at the merger node, makes efficiency better.

3.2 System Model

Having presented temporal merging and identified the need for reducing packet replacements and aggregations due to *timeout condition*, we provide a formal model of the aggregation system. The system model is dependent upon the manner in which the merging timer behaves. We study two schemes for regulating the behavior of the merging timer. *(i)* The merger timer expires at regular prescheduled intervals. We call this the "periodic scheme". *(ii)* The merger timer is set upon arrival of the first feedback packet following a merge. We call this the "triggered scheme". The former is simpler to implement than the latter. Throughout this section, the variables marked with a circumflex are related to the triggered scheme. The model we provide makes no particular assumptions about the amount of feedback packets transmitted (*i.e.* we make no particular assumptions about feedback transmission characteristics).

Consider the merging timer schematics shown in Figure 3.1. Let k be the number of receivers from which the feedback merger is waiting for packets. For a given timer interval n, let ρ_i^n and β_i^n be, respectively, the number of packet arrivals from receiver i and the number of its packets that have been replaced.

Periodic scheme

Under the periodic scheme, a timer interval n occurs even in the absence of feedback. For a feedback merger waiting for packets from k receivers, the number of outgoing packets from the timer interval n is given by

$$\gamma_{out}^n = \max_{i=1,\ldots,k}[\rho_i^n - \beta_i^n], \qquad \gamma_{out}^n \geq 0, \tag{3.1}$$

and the number of incoming packets in the timer interval n is

$$\gamma_{in}^n = \sum_{i=1}^{k} \rho_i^n, \qquad \gamma_{in}^n \geq 0. \tag{3.2}$$

Given a timer interval n, the number of outgoing packets due to *packet condition* is given by

$$\mu_k^n = \min_{i=1,\ldots,k}[\rho_i^n - \beta_i^n], \qquad \mu_k^n \geq 0, \tag{3.3}$$

and the number of outgoing packets attributable to *timeout condition* is

$$\sigma_k^n = \max_{i=1,\ldots,k}[\rho_i^n - \beta_i^n] - \min_{i=1,\ldots,k}[\rho_i^n - \beta_i^n], \qquad \sigma_k^n = 0 \text{ or } 1. \tag{3.4}$$

If timer expiration occurs and there exists at least one feedback packet waiting for merging, then $\sigma_k^n = 1$ else $\sigma_k^n = 0$. Note that the number of outgoing packets from a timer interval n is equal to $\mu_k^n + \sigma_k^n$, the same result as in (3.1).

Proposition 3.2.1 *Performance in a timer interval n is efficient if there are no packet replacements and the amount of packet arrivals from all receivers is the same.*

Proof. To achieve efficient performance in a timer interval n, the following two conditions must hold: $\beta_i^n = 0$ for all $1 \leq i \leq k$, and $\sigma_k^n = 0$. According to Eq. (3.4), it follows that if β_i^n is equal to zero for all $1 \leq i \leq k$, σ_k^n will also be equal to zero if $\max_{i=1,\ldots,k}[\rho_i^n] = \min_{i=1,\ldots,k}[\rho_i^n]$, which implies $\rho_1^n = \rho_2^n = \ldots = \rho_k^n$. Because no ρ_i^n is allowed to be equal to zero in pursuing efficient performance, $\rho_1^n = \rho_2^n = \ldots = \rho_k^n \geq 1$. $\square$

Remark 3.2.2 In order to achieve efficient performance in a timer interval n, the amount of packet arrivals from all receivers must be the same and packet replacements must not occur. Evidently, the transmission of feedback packets needs to be coordinated such that it leads to coordinated packet arrivals at the merger node.

Remark 3.2.3 The periodic scheme allows for more than one merging procedure attributable to *packet condition* per timer interval. The number depends upon timer settings and feedback transmission characteristics.

The preceding proposition allows to start a new proposition that will be ulteriorly useful for further analysis.

Proposition 3.2.4 *If efficient performance is achieved in a timer interval n, then*

$$\frac{\gamma_{out}^n}{\gamma_{in}^n} = \frac{1}{k} \tag{3.5}$$

Proof. This demonstration is simple and we present it here for the sake of completeness. We know that efficient performance implies $\rho_1^n = \rho_2^n = \ldots = \rho_k^n \geq 1$. We can rewrite this as $\rho_1^n = \rho_2^n = \ldots = \rho_k^n \geq x$, where $x \geq 1$. Thus, $\gamma_{in}^n = \sum_{i=1}^k \rho_i^n = kx$. Since $\beta_1^n = \beta_2^n = \ldots = \beta_k^n = 0$ is a necessary condition for efficient performance, $\gamma_{out}^n = \max_{i=1,\ldots,k}[\rho_i^n - \beta_i^n] = x$. $\square$

Triggered scheme

In the triggered scheme[1], at least one feedback packet is necessary in order to start a timer interval. It is easy to argue that for a feedback merger waiting for packets from k receivers, the number of outgoing packets from a timer interval n is

$$\hat{\gamma}_{out}^{n} = \max_{i=1,\ldots,k}[\rho_i^n - \beta_i^n] = 1. \tag{3.6}$$

To see this, consider that the arrival of a feedback packet pertaining to a receiver j sets the merger timer. For such a receiver, $\rho_j^n - \beta_j^n$ is equal to one because a timer interval will be finished after the occurrence of a merging procedure. By way of the same reasoning, it can be easily verified that for a receiver i, where $i \neq j$, $\rho_i^n - \beta_i^n$ is at the most one.

The number of incoming packets in the timer interval n is

$$\hat{\gamma}_{in}^{n} = \sum_{i=1}^{k} \rho_i^n, \qquad \hat{\gamma}_{in}^{n} \geq 0. \tag{3.7}$$

Given a timer interval n, the number of outgoing packets due to *packet condition* is given by

$$\hat{\mu}_k^n = \min_{i=1,\ldots,k}[\rho_i^n - \beta_i^n], \qquad \hat{\mu}_k^n = 0 \ or \ 1, \tag{3.8}$$

and the number of outgoing packets attributable to *timeout condition* is given by

$$\hat{\sigma}_k^n = 1 - \min_{1 \leq i \leq k}[\rho_i^n - \beta_i^n] \ , \qquad \hat{\sigma}_k^n = 0 \ or \ 1, \tag{3.9}$$

It should be noted here that in a timer interval n, if β_i^n is equal to zero for all $1 \leq i \leq k$, then $\hat{\sigma}_k^n$ will be equal to zero if $\min_{1 \leq i \leq k}[\rho_i^n]$ is equal to one. That is, $\rho_1^n = \rho_2^n = \ldots = \rho_k^n = 1$. Thus, regardless of the amount of transmitted feedback packets, the model shows that an efficient performance in a timer interval n is achieved if the number of packet arrivals from all receivers is equal to one.

Remark 3.2.5 Efficient performance in a timer interval n means that

$$\frac{\hat{\gamma}_{out}^{n}}{\hat{\gamma}_{in}^{n}} = \frac{1}{k} \tag{3.10}$$

[1] We emphasize that throughout this thesis, variables marked with a circumflex are related to the triggered scheme.

3.3 Metrics: Fraction of Merges due to *Packet* and *Timeout Conditions*

In this section we define two necessary metrics for the evaluation of the system performance: the fraction of merges due to *packet* and *timeout conditions*.

3.3.1 Periodic scheme

Considering the overall number N of timer intervals, the fraction of merges due to *timeout condition* is given by

$$\alpha_k^{packet} = \frac{\sum_{n=0}^{N-1} \min_{i=1,\dots,k}[\rho_i^n - \beta_i^n]}{\sum_{n=0}^{N-1} \max_{i=1,\dots,k}[\rho_i^n - \beta_i^n]}, \qquad 0 \leq \alpha_k^{packet} \leq 1, \tag{3.11}$$

and the fraction of merges attributable to *timeout condition* is

$$\alpha_k^{timeout} = 1 - \frac{\sum_{n=0}^{N-1} \min_{i=1,\dots,k}[\rho_i^n - \beta_i^n]}{\sum_{n=0}^{N-1} \max_{i=1,\dots,k}[\rho_i^n - \beta_i^n]}, \qquad 0 \leq \alpha_k^{timeout} \leq 1. \tag{3.12}$$

The above-mentioned formulæ hold if for some n, $0 \leq n \leq (N-1)$, there exists an i, $1 \leq i \leq k$, such that $\rho_i^n \geq 1$.

3.3.2 Triggered scheme

Considering the overall number N of *existing* timer intervals, the fraction of merges attributable to *packet condition* is given by

$$\hat{\alpha}_k^{packet} = \frac{1}{N} \sum_{n=0}^{N-1} \min_{1 \leq i \leq k} [\rho_i^n - \beta_i^n] \ , \qquad 0 \leq \hat{\alpha}_k^{packet} \leq 1, \tag{3.13}$$

and the fraction of merges due to *timeout condition* is

$$\hat{\alpha}_k^{timeout} = 1 - \frac{1}{N} \sum_{n=0}^{N-1} \min_{1 \leq i \leq k} [\rho_i^n - \beta_i^n] \ , \qquad 0 \leq \hat{\alpha}_k^{timeout} \leq 1 \ . \tag{3.14}$$

3.4 Metric: Global Filtering Level

The effectiveness of implosion avoidance is commonly measured by the degree of packet suppression. For this reason, we introduce a performance metric related to packet suppression.

Definition 3.4.1 *We define* Filtering Level *in a timer interval n as the number of packets suppressed by the feedback merger over the total number of incoming packets.*

Definition 3.4.2 *We define* Global Filtering Level *as the overall number of packets suppressed by the feedback merger over the overall number of incoming packets.*

Definition 3.4.3 *The* optimal value *of the Global Filtering Level is defined as the value that results from efficient performance.*

3.4.1 Periodic scheme

Under the periodic scheme, the *Filtering Level* is given by

$$\xi^n = 1 - \frac{\gamma_{out}^n}{\gamma_{in}^n}, \qquad 0 \le \xi^n < 1, \tag{3.15}$$

if $\gamma_{in}^n \ge 1$. In generalizing the *Filtering Level* to take into account the overall number N of timer intervals, we get the *Global Filtering Level*

$$\xi_k = 1 - \frac{\sum_{n=0}^{N-1} \gamma_{out}^n}{\sum_{n=0}^{N-1} \gamma_{in}^n}, \qquad 0 \le \xi_k < 1, \tag{3.16}$$

The preceding formula holds if for at least one timer interval n, $0 \le n \le (N-1)$, $\gamma_{in}^n \ge 1$. Substituting equations (3.1) and (3.2) in (3.16), we obtain

$$\xi_k = 1 - \frac{\sum_{n=0}^{N-1} \max_{i=1,\ldots,k}[\rho_i^n - \beta_i^n]}{\sum_{n=0}^{N-1} \sum_{i=1}^{k} \rho_i^n}, \qquad 0 \le \xi_k < 1, \tag{3.17}$$

if for some n, $0 \le n \le (N-1)$, there exists an i, $1 \le i \le k$, such that $\rho_i^n \ge 1$.

Proposition 3.4.4 *The* optimal value *of the Global Filtering Level is*

$$\xi_{k,optimal} = 1 - \frac{1}{k}, \qquad \text{for } k > 1. \tag{3.18}$$

Proof. For every timer interval n having packets within and exhibiting efficient performance, the $\frac{\gamma_{out}^n}{\gamma_{in}^n}$ ratio is equal to $\frac{1}{k}$. So that, the $\frac{\sum_{n=0}^{N-1} \gamma_{out}^n}{\sum_{n=0}^{N-1} \gamma_{in}^n}$ ratio will also be equal to $\frac{1}{k}$. □

3.4.2 Triggered scheme

Under the triggered scheme, the *Filtering Level* is given by

$$\hat{\xi}^n = 1 - \frac{\hat{\gamma}^n_{out}}{\hat{\gamma}^n_{in}}, \qquad 0 \leq \hat{\xi}^n < 1, \tag{3.19}$$

if $\gamma^n_{in} \geq 1$. The *Global Filtering Level* is obtained by generalizing the *Filtering Level* to take into account the overall number N of timer intervals. In this manner, the *Global Filtering Level* is given by

$$\hat{\xi}_k = 1 - \frac{\sum_{n=0}^{N-1} \hat{\gamma}^n_{out}}{\sum_{n=0}^{N-1} \hat{\gamma}^n_{in}}, \qquad 0 \leq \hat{\xi}_k < 1, \tag{3.20}$$

Substituting equations (3.6) and (3.7) in (3.20), we obtain

$$\hat{\xi}_k = 1 - \frac{N}{\sum_{n=0}^{N-1} \sum_{i=1}^{k} \rho^n_i}, \qquad 0 \leq \hat{\xi}_k < 1, \tag{3.21}$$

for N existing timer intervals.

The *optimal value* of the Global Filtering Level is given by

$$\hat{\xi}_{k,optimal} = 1 - \frac{1}{k}, \qquad \text{for } k > 1. \tag{3.22}$$

3.5 System Behavior

In this section our objective is to provide a formal description of the system behavior.

Proposition 3.5.1 *The Global Filtering Level can assume its optimal value even though performance is inefficient.*

Proof. The Global Filtering Level should be as close as possible to its optimal value in pursuit of efficient performance. However, an optimal value of the Global Filtering Level does not imply efficiency because packet replacements and merges due to *timeout condition* may be occurring. To prove the proposition, it is necessary to recognize that in a timer interval n where both packet replacements and *timeout condition* hold, the only possible situations are as follow:

- **situation 1 -** periodic scheme: $\gamma^n_{in} > k \max_{1 \leq i \leq k}[\rho^n_i - \beta^n_i]$; triggered scheme: $\hat{\gamma}^n_{in} > k$.

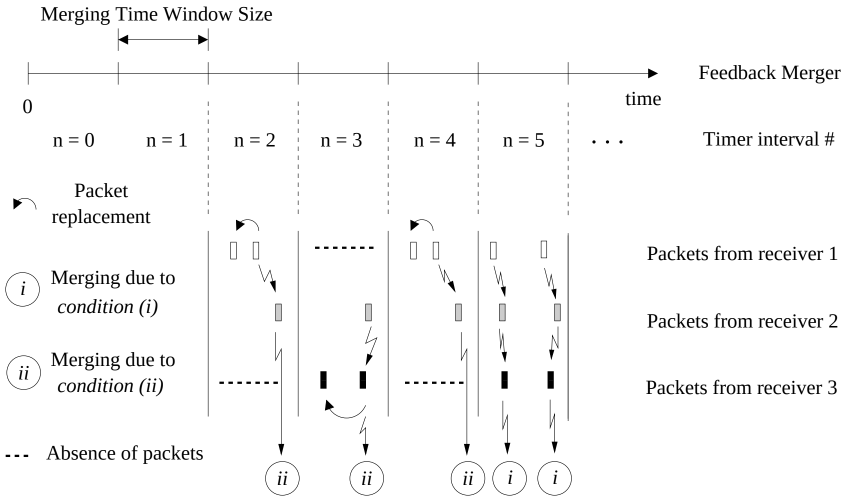


Figure 3.2: Scenario in which the Global Filtering Level assumes its optimal value.

- **situation 2 -** periodic scheme: $\gamma_{in}^n < k \max_{1 \le i \le k}[\rho_i^n - \beta_i^n]$; triggered scheme: $\hat{\gamma}_{in}^n < k$.
- **situation 3 -** periodic scheme: $\gamma_{in}^n = k \max_{1 \le i \le k}[\rho_i^n - \beta_i^n]$; triggered scheme: $\hat{\gamma}_{in}^n = k$.

Whenever both packet replacements and timeout condition occur but **situation 3** holds, $\frac{\gamma_{out}^n}{\gamma_{in}^n} = \frac{1}{k}$. If timer intervals only exhibit such a behavior or only exhibit efficient performance or both, $\frac{\sum_{n=0}^{N-1} \gamma_{out}^n}{\sum_{n=0}^{N-1} \gamma_{in}^n} = \frac{1}{k}$. So that, $\xi_k = \xi_{k,optimal}$. Using the same reasoning, it turns out that $\hat{\xi}_k = \hat{\xi}_{k,optimal}$. □

Figure 3.2 illustrates proposition 3.5.1 when the periodic scheme is used. In relation to the timer intervals 2, 3, and 4, the Global Filtering Level (ξ_3) is equal to its optimal value of 0.66. It should be noted that the occurrence of such a scenario depends to a great extent on feedback transmission characteristics. The coordination of feedback transmission to some level should make such a scenario unlikely.

Proposition 3.5.2 *If there are no packet replacements, the Global Filtering Level will be less than or equal to its optimal value.*

Proof. The equality in the proposition is obtained by considering that merges occur only due to *packet condition* but without packet replacements and so it means efficient performance. To prove the inequality it is sufficient to look into a single timer interval n and demonstrate the following:

- **periodic scheme:** if *timeout condition* holds in a non-empty timer interval n but there are no packet replacements, $\frac{\gamma_{out}^n}{\gamma_{in}^n} > \frac{1}{k}$;
- **triggered scheme:** if *timeout condition* holds but packet replacements do not occur, $\frac{\hat{\gamma}_{out}^n}{\hat{\gamma}_{in}^n} > \frac{1}{k}$.

The scenario under which the periodic scheme operates means that $\gamma_{in}^n < k \max_{1 \leq i \leq k}[\rho_i^n - \beta_i^n]$. From our model, we know that $\gamma_{out}^n = \max_{1 \leq i \leq k}[\rho_i^n - \beta_i^n]$. In this manner, $\frac{\gamma_{out}^n}{\gamma_{in}^n} > \frac{1}{k}$. For the triggered scheme, $\hat{\gamma}_{in}^n < k$ and so $\frac{\hat{\gamma}_{out}^n}{\hat{\gamma}_{in}^n} > \frac{1}{k}$.

□

PROPOSITION 3.5.3 *If there are packet replacements, the Global Filtering Level may assume values greater than its optimal value.*

Proof. Whenever $\hat{\gamma}_{in}^n > k$ for the triggered scheme or $\gamma_{in}^n > k \max_{1 \leq i \leq k}[\rho_i^n - \beta_i^n]$ for the periodic scheme, it means that there is at least one packet replacement. The scenario leads to $\frac{\hat{\gamma}_{out}^n}{\hat{\gamma}_{in}^n} < \frac{1}{k}$ for the triggered scheme and to $\frac{\gamma_{out}^n}{\gamma_{in}^n} < \frac{1}{k}$ for the periodic scheme. So, the Global Filtering Level may assume values greater than its optimal value whenever a combination of timer intervals leads to $\frac{\sum_{n=0}^{N-1} \hat{\gamma}_{out}^n}{\sum_{n=0}^{N-1} \hat{\gamma}_{in}^n} < \frac{1}{k}$ or to $\frac{\sum_{n=0}^{N-1} \gamma_{out}^n}{\sum_{n=0}^{N-1} \gamma_{in}^n} < \frac{1}{k}$ in accordance with the adopted timer scheme.

□

REMARK 3.5.4 Whenever mergers occur due to *packet condition* but there are packet replacements, the Global Filtering Level will be greater then its optimal value.

PROPOSITION 3.5.5 *If the fraction of merges due to* timeout condition *is equal to zero, then the Global Filtering Level will be greater than or equal to its optimal value.*

Proof. Under the condition that the fraction of merges due to *timeout condition* is equal to zero, merges can occur with or without packet replacements but evidently due to *packet condition*. Merges due to *packet condition* and the further fact that there are no packet replacements mean efficient performance and so the Global Filtering Level assumes its optimal value.

To prove the inequality, it is sufficient to demonstrate that, in a single timer interval n, if merging procedures occur due to *packet condition* but there is at least one packet replacement, then $\frac{\gamma_{out}^n}{\gamma_{in}^n} < \frac{1}{k}$.

If performance is efficient, we know that $\frac{\gamma_{out}^n}{\gamma_{in}^n} = \frac{1}{k}$. On the other hand, if merging procedures within the timer interval n occur due to *packet condition* but there is at least one packet replacement, it means that γ_{in}^n is greater than $k \max_{1 \le i \le k}[\rho_i^n - \beta_i^n]$. From our model, we know that $\gamma_{out}^n = \max_{1 \le i \le k}[\rho_i^n - \beta_i^n]$. Thus, $\frac{\gamma_{out}^n}{\gamma_{in}^n} < \frac{1}{k}$. Considering a number N of timer intervals such that for each $0 \le n \le N$, $\gamma_{in}^n \ge k \max_{1 \le i \le k}[\rho_i^n - \beta_i^n]$, $\xi_k \ge \xi_{k,optimal}$. The demonstration for the triggered scheme is similar. □

Remark 3.5.6 A Global Filtering Level that is greater than its optimal value increases efficiency for feedback implosion avoidance, but is inadequate for effective feedback.

3.6 Feedback Transmission

Feedback transmission describes the manner in which receivers decides when to send feedback packets upstream. Feedback transmission affects the manner in which an aggregator receives feedback packets and so it potentially impacts efficiency. To improve efficiency, feedback transmission and packet arrivals at the aggregator need to be coordinated to some extent. Figure 3.3 illustrates these ideas.

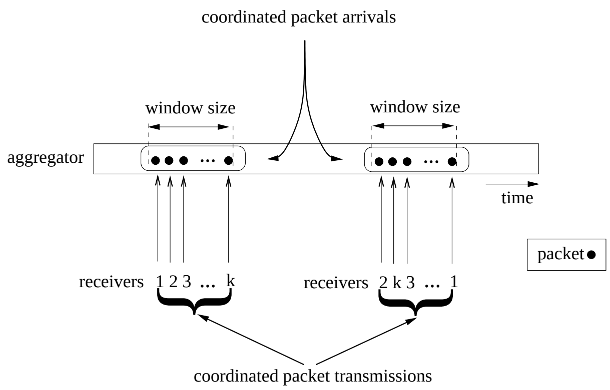


Figure 3.3: Efficient performance: coordinated packet transmission and arrivals.

It should be noted that packet transmission coordination is consonant with an optimal solution for transmitting feedback in which the session bandwidth devoted to the feedback traffic is equally shared among the receivers. So, all the receivers issue reports at the same rate. In this section, we investigate possible solutions to coordinate packet transmission. These solutions in turn must result in some coordinated packet arrivals at the aggregator.

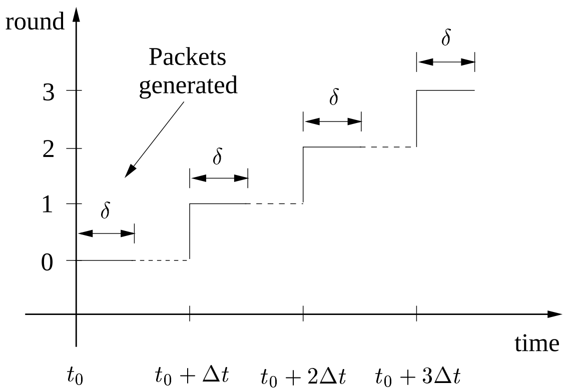


Figure 3.4: Coordinated feedback transmissions.

3.6.1 Exploiting synchronization

Let δ be the amount of time by which the feedback can be delayed. Let Δt be the feedback sending interval. Feedback transmission occurs at rounds. Let r be a feedback round, starting at $t_0 + r\Delta t$, where t_0 is a well-known time[2] of the first round $r = 0$. At each round, each receiver draws a random variable x uniformly distributed in [0,1] and sets its feedback timer to $x\delta$. Upon timer expiration, a feedback packet is issued. The algorithm described spreads receivers' feedback transmission uniformly in $[t_0 + r\Delta t, t_0 + r\Delta t + \delta]$ and so feedback transmission is coordinated. Figure 3.4 illustrates feedback transmission coordination at receivers.

Let us now consider the aggregator. Let d_i be the one-way delay estimate between receiver i and aggregator. Let $\Delta\tau$ be the merging time window size. On receiving the first feedback packet following a merging, the aggregator sets a merging timer. The merging time window should be sized sufficient comfortable to allow that one packet from each of the k receivers arrives. Figure 3.5 illustrates a worst-case scenario that allows us to determine the merging time window size. In this figure, we assume that $k = 2$ and $d_1 < d_2$. Further, we assume that receiver 1 issues a packet at t_0 and receiver 2, which experiments the worst one-way delay, issues its packet at $t_0 + \delta$. In this case, $\Delta\tau$ should be greater than $(\delta - d_1) + d_2$. In generalizing the scenario to take into account k receivers, the merging time window should be sized so that

$$\Delta\tau > (\delta - \min_{1 \leq i \leq k}[d_i]) + \max_{1 \leq i \leq k}[d_i] \tag{3.23}$$

[2]In practice, the start-time needs to *reliably* reach all feedback senders in advance of the indicated time.

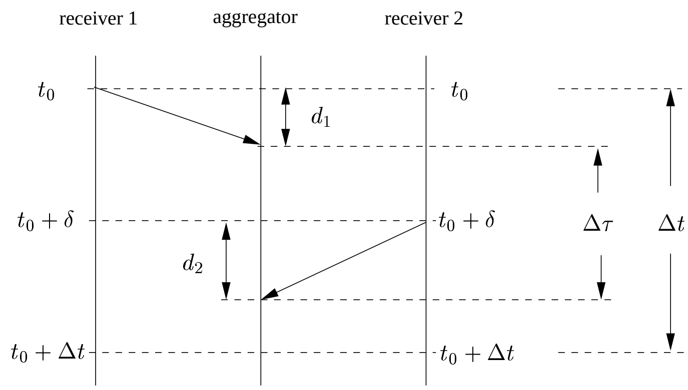


Figure 3.5: Window management.

The merging time window relates directly to the data age and so it should be sized as close as possible to its lower bound to keep the worst-case age to a minimum. Even though, it might be desired to increase the merging time window size by a small amount of time in order to safeguard against variations in the one-way delay estimate.

A conservative lower bound on the feedback sending interval can be obtained by considering the scenario where only the packet from the receiver that experiments the worst one-way delay is not lost but its packet transmission is delayed by δ. In this case, to reduce the probability of merging packets that are related to round $r+1$ in the round r, the lower bound on Δt should be

$$\Delta t > 2\Delta\tau \tag{3.24}$$

The timer settings just described were obtained by considering that the merging timer is set upon receipt of the first packet following an aggregation. Another possibility might exploit the clock synchronization to set the merging timer expiration at regular prescheduled intervals T_{exp}. In this case, the merging timer should expire at

$$T_{exp} = t_0 + r\Delta t + (\delta - \min_{1 \le i \le k}[d_i]) + \max_{1 \le i \le k}[d_i] \tag{3.25}$$

and the lower bound on the feedback interval is given by

$$\Delta t > (\delta - \min_{1 \le i \le k}[d_i]) + \max_{1 \le i \le k}[d_i] \tag{3.26}$$

The algorithm just described makes perfect sense assuming that good estimates of the one-way delay and clock synchronization are available. Estimates of the one-way delay can be done by the aggregator using timestamps in feedback packets. To reduce computation overhead, the aggregator might set $\min_{1 \leq i \leq k}[d_i] = 0$ and only keep track of the maximum observed one-way delay per receiver. It should be noted that using the maximum observed one-way delay might be highly conservative because ninety-nine percent of packets on a connection are guaranteed to suffer a delay smaller than 99-*percentile*[3] delay [58]. In this manner, the term *max* in equations can be expressed as a statistical bound.

Clock synchronization might be achieved using Global Positioning System (GPS) receivers or less accurately, it can be done with the Network Time Protocol (NTP) [59, 60]. In the case of NTP, the errors that accumulate between the stratum-1 server and the local host must be taken into account. An NTP server knows the synchronization error to the stratum-1 server to which it is synchronized. In the same manner, a host can determine the synchronization error to the NTP server to which it is synchronized. The sum of these errors gives the worst-case error in synchronization. Assume that ϵ_i is the worst-case error in synchronization between a local host i, $1 \leq i \leq k$, and the stratum-1 server. Let us now revisit the timer settings.

Assuming that the merging timer is set upon receipt of the first packet following a merging, a conservative merging time window size is given by

$$\Delta\tau > 2 \max_{1 \leq i \leq k}[\epsilon_i] + \delta + \max_{1 \leq i \leq k}[d_i] \tag{3.27}$$

and the lower bound on Δt is given by

$$\Delta t > 2\Delta\tau \tag{3.28}$$

In the case of the merging timer expires at regular prescheduled intervals, the timer might be conservatively scheduled to expire at

$$T_{exp} = t_0 + r\Delta t + \delta + \max_{1 \leq i \leq k}[\epsilon_i + d_i] + \epsilon_{aggregator} \tag{3.29}$$

and the lower bound on the feedback interval is given by

$$\Delta t > \delta + \max_{1 \leq i \leq k}[\epsilon_i + d_i] + \epsilon_{aggregator} \tag{3.30}$$

The solution we have presented results in coordinated packet transmission and coordinated packet arrivals at the aggregator. In practice, however, the performance of such

[3] Percentiles are a popular means of specifying dispersion. The α-percentiles can be estimated by sorting the n observations and taking the $[(n-1)\alpha + 1]th$ element in the ordered set.

a solution is strongly dependent on clock synchronization. For reasons of cost and convenience, it is not possible to equip every computer with GPS receivers. On the other hand, using a less accurate synchronization method as NTP has several issues. NTP may take several minutes or even hours to adjust a system's time to the ultimate degree of accuracy[4]. There are several reasons for this. NTP averages the result of several time exchanges in order to reduce the effects of variable latency, so it may take several minutes for NTP to even reach consensus on what the average latency is. Generally this happens in about 5 minutes [61]. Further, it often takes several adjustments for NTP to reach synchronization. The result is that we should not expect NTP to immediately synchronize two clocks.

Under good conditions on a Local Area Network (LAN) without too many routers or other sources of network delay, synchronization to within a few milliseconds is normal. Anything that adds latency, such as hubs, switches, routers, or network traffic, will reduce this accuracy. The synchronization accuracy on a Wide Area Network (WAN) is typically within the range of 10-100 ms [61]. For the Internet, synchronization accuracy is unpredictable. Furthermore, NTP is not guaranteed to be secure.

3.6.2 Exploiting tight feedback control

An alternative solution to the packet coordination problem is the employment of a polling mechanism on the part of aggregators. Polling mechanisms have already been widely studied in the literature [27, 28, 29, 30]. These mechanisms are commonly used with backoff timers to avoid instantaneous massive responses. However, polling in rounds with backoff response can take a significant time [62]. The aggregator can use shared or individual channels for feedback request as illustrated in Figure 3.6. A shared channel for feedback request reduces transmission overhead but is not priceless. The lost of a single feedback request potentially prevents response from many or all receivers, which in fact will highly degrade efficiency.

Another solution is to have the source transmitting a stimulus message to receivers, which then respond by sending a report. This idea is exploited in [18], where the transmission of each receiver report is triggered by the reception of a control packet multicasted by the source. Stimulus messages are transmitted once for every R video packets transmitted, where R is assumed to be a small number such as 15. This assumption in practice leads to excessive control traffic and makes packet coordination at the aggregator difficult. Growing R helps to reduce overhead and may provide better coordination of packet arrivals at the aggregator. However, as in the case of polling, due to packet loss, not all recipients will receive a stimulus message, potentially degrading efficiency.

[4]Accuracy measures the extent to which a given clock agrees with Coordinated Universal Time (UTC). For example, the clock on a host might be 27.1 msec behind or ahead UTC.

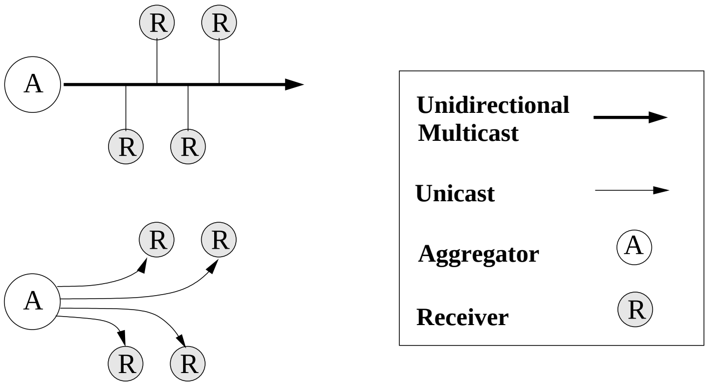


Figure 3.6: Shared and individual channels for feedback request.

In general, schemes that explicitly coordinate the transmissions of all receiver reports tend to generate excessive control traffic. In addition, determining an optimal timer setting at the aggregators may be difficult.

3.6.3 Periodic Feedback

The solutions we have examined require either clock synchronization or a tight control for every feedback transmission. We present a solution to the packet coordination problem that does not rely on such requirements and hence has greater potential for scalability. We propose to use a feedback transmission mechanism whereby a steady state leads to a reasonable efficiency. This mechanism relates purely to the timer settings used by receivers. Feedback transmission proceeds as follows: each receiver regularly transmits consecutive feedback packets spaced apart by Δt. This implicitly coordinates packet arrivals at the aggregation node. The coordination, and consequently, the efficiency using such a mechanism is a compromise between the feedback sending interval and the amount of jitter[5] experienced by each receiver's feedback stream. If the feedback sending interval is too small, a small amount of jitter degrades efficiency. If the feedback sending interval is too large, the efficiency remains immune to jitter, but the extra delay between consecutive reports and the waiting time for merging, may be inadequate to some applications. Clearly, efficiency is also dependent upon packet loss. If loss occurs, then there is really nothing that can be done.

The idea behind our solution is to improve efficiency by allowing the aggregator to take advantage of the timely manner in which feedback are generated. In a realistic scenario, the resulting packet from an aggregation procedure will be issued to some extent in a

[5]The typical cause of jitter is other traffic on a network.

timely fashion, a feature which is particularly interesting for applications that require a more timely feedback. It should be noted that we do not specify the control scheme for regulating the amount of generated feedback traffic. We do not treat this problem is this thesis. However, we will present a potential solution. We consider the feedback transmission mechanism presented here as a basis to built feedback control schemes that are aware of efficient performance at the aggregator but also willing to minimize network overhead.

Feedback loop

The feedback sending interval that meets specified performance requirements can be determined by the aggregation node and may change over time due to dynamic network conditions of delay variability and variations in the number of receivers returning reports. Thus, a mechanism to inform receivers about changes in the feedback sending interval must be employed. A *feedback interval field* in feedback packets can be used by the aggregation node to maintain its upstream neighbor node (*e.g.* the source or another aggregation node) informed about the minimum feedback sending interval that is necessary to improve its performance. When an aggregator receives aggregate reports, the *feedback interval field* of outgoing packets must contain the maximum value between the feedback sending interval informed by its downstream neighbor nodes and the feedback sending interval that will improve its performance. The incoming report at the source will carry the feedback sending interval value to be adopted by the receivers. The source informs the receivers of the new Δt by updating a *feedback interval field* in data packets.

Limited feedback traffic

The amount of generated feedback traffic grows linearly with the number of receivers. However, due to feedback aggregation, the amount of incoming feedback packets at the source is effectively equivalent to the amount of feedback generated by a single receiver. Even though, it might be desirable to limit the feedback traffic to a percentage of the session bandwidth as in RTP. The feedback sending interval to accomplish this purpose is given by

$$T = 100 \;.\; \frac{\text{number of participants . feedback packet size}}{\text{\% of session bandwidth . session bandwidth}} \;. \qquad (3.31)$$

Since an aggregation node knows the number of its descendant receivers, this information can be carried by outgoing feedback packets and aggregated at upstream aggregation nodes to provide the source with a reasonably count of the number of receivers. The new Δt to be adopted is the maximum value between T and the feedback sending interval value carried by incoming reports at the source.

This mechanism for regulating the amount of generated feedback traffic does not require the use of additional control packets, thus reducing overhead.

3.7 Coordination of Packet Arrivals and Window Size

Section 3.6 states that periodic feedback transmission leads to coordinated packet arrivals at the aggregator. In this section, we demonstrate it does indeed. For this purpose, we assume the nonexistence of jitter and packet loss. The existence of jitter coupled with the fact that feedback is sent in a periodic manner means that packet replacements cannot occur. With the further assumption that there is no packet loss, we can determine the sizes for the merging time window that lead to efficient performance. The performance metrics are the Global Filtering Level and the fraction of merges due to *packet condition*.

Consider our basic network topology (Figure 3.1). Each receiver regularly sends consecutive feedback packets spaced apart by Δt. The merging time window size is $\Delta\tau$.

3.7.1 Periodic Scheme

For $k = 2$ and $k = 3$, Figure 3.7 shows steady-state values of ξ_k and $\alpha_k^{timeout}$ as a function of the feedback sending interval. The plot is a scatter plot. For each Δt, different values of ξ_k and $\alpha_k^{timeout}$ are obtained by varying the arrival-times of all receivers' first reports over the interval $[0,\ \Delta t)$. By steady-state values, we mean values computed over a recurring pattern of merges at the merger node. The main conclusions regarding these figures are the following. First, merges due to *timeout condition* lead to a Global Filtering Level that is lower than its optimal value. Second, setting the merging time window size for a multiple of the feedback sending interval ensures that all merges occur due to *packet condition*. That is, assuming that there is no packet loss, efficient performance is achieved if $\Delta\tau = b\Delta t$, where $b = 1, 2, \ldots$. Factor b represents the expected number of feedback packets from each receiver per timer interval. One method of reducing the probability of packet replacements in the presence of packet loss is to make $b = 1$. This choice also reduces the maximum possible waiting time for merging. Hence, we adopt $b = 1$.

Finally, it should be noted that if the feedback sending interval Δt (or the feedback interarrival interval) is around window size $\Delta\tau$, then the performance of the periodic scheme

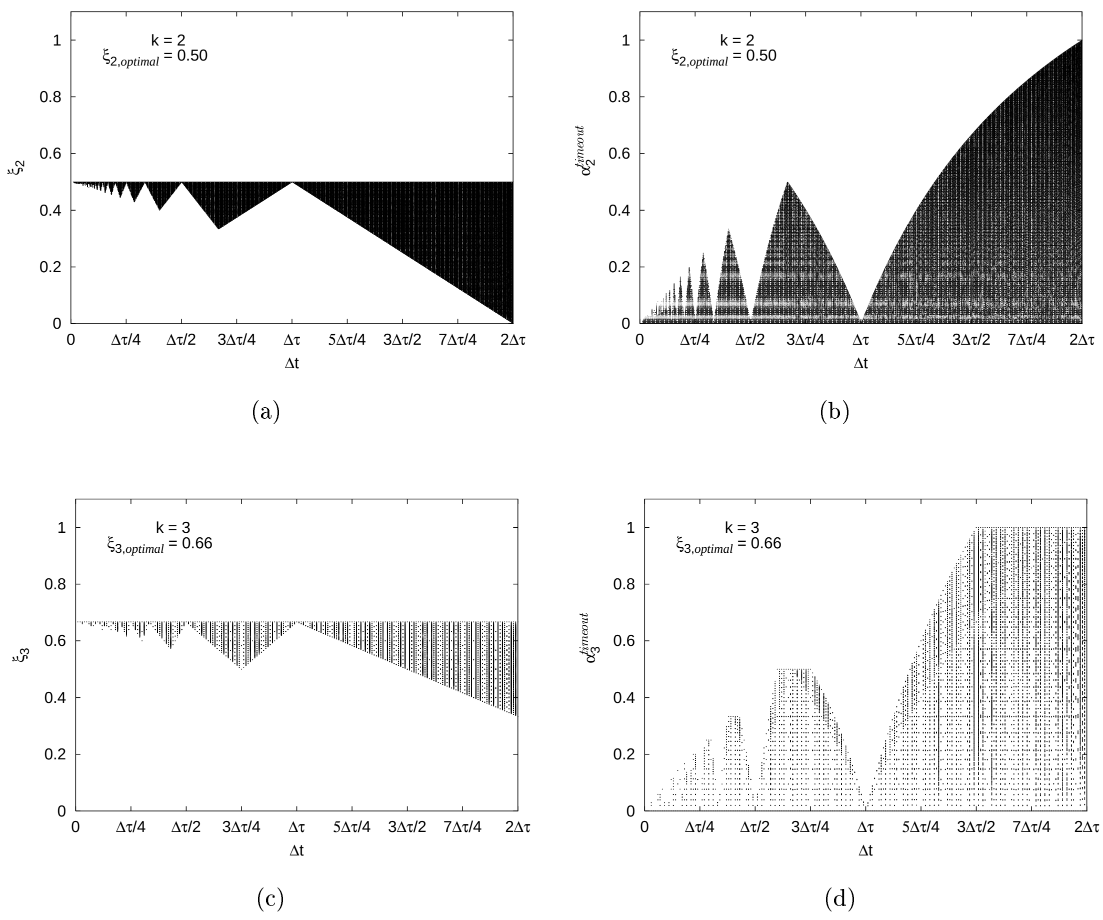


Figure 3.7: Theoretical results for the periodic scheme.

may decline. The occurrence of random network jitter leads to a feedback interarrival interval close to the merging time window size. We should expect, therefore, the periodic scheme to perform well only when the feedback sending interval value is higher than the value of jitter. How much higher the feedback sending interval value should be is investigated in Section 3.8.3.

3.7.2 Triggered Scheme

For $k = 2$, $k = 3$, and $k = 4$, Figure 3.8 shows the steady-state values of $\hat{\xi}_k$ and $\hat{\alpha}_k^{timeout}$ as function of the feedback sending interval. Note that, as highlighted in each plot, efficient performance is achieved when $\Delta t < (\hat{\xi}_{k,optimal})^{-1}\Delta\tau$, thereby providing a lower bound on $\Delta\tau$. In the case of packet losses, the strategy employed to reduce the number of packet replacements is to maintain $\Delta\tau < \Delta t$. Thus, we derive the following inequality:

$$\hat{\xi}_{k,optimal}\Delta t < \Delta\tau < \Delta t \ , \tag{3.32}$$

3.8 Impact of Random Delays

Section 3.7 firmly establishes that periodic feedback transmission leads to coordinated packet arrivals at the aggregator. However, we have pointed out in Section 3.6.3 that jitter experienced by each receiver's feedback stream affects performance. Jitter may potentially induce packet replacements. Another source of inefficiency is a potential variability in the true size of the merging time window. We next explain this issue.

Timers are generally implemented using a periodic tick interrupt. The time interval between consecutive ticks is called system timer resolution. On standard Linux implementations the system timer resolution is 10 ms while on RTLinux (Real Time Linux) it is 1 ms. If a timer is not schedule to go off on a tick boundary, its expiration will be offset by the duration time to the next tick. Further, upon expiration, the signal delivery may be offset by a small time dependent on the system loading. The consequence of an increased merging time window size is the possibility of packet replacements, which also degrades performance. On the other hand, variability in the timer settings of a receiver does not introduce a new problem we need to take into account because the effect of such a variability is jitter in the feedback stream.

In this section, we study the impact of jitter and timer variability on the aggregator efficiency. Let δ be a positive bounded variable representing the variation in the merging time window size. Let l and m represent two receivers. Let J_l^{max} and J_m^{max} be, respectively,

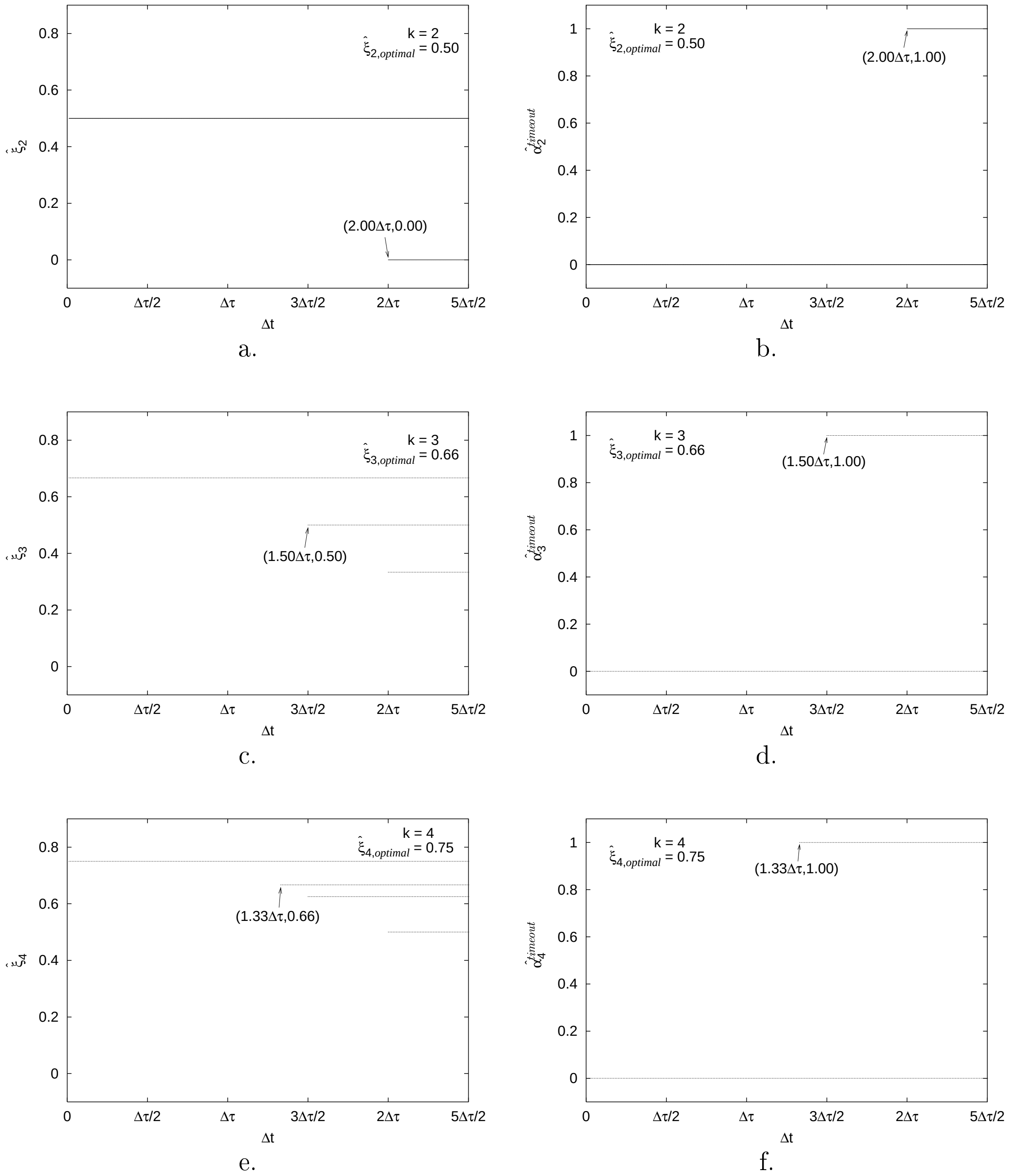


Figure 3.8: Theoretical results for the triggered scheme.

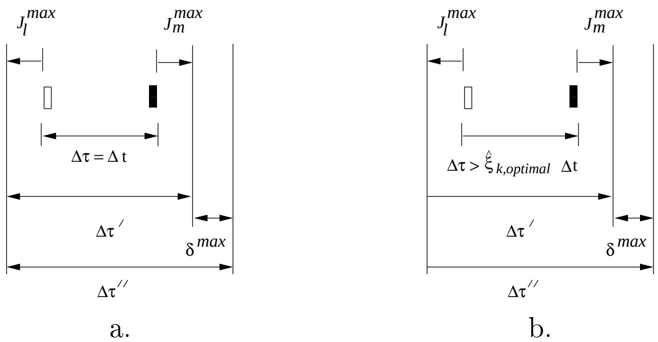


Figure 3.9: Worst case of delay variability for each timer scheme.

a statistical bound of jitter experienced by the feedback stream of receivers l and m. We assume a low variance among the statistical bound of jitter in the feedback flow of receivers. Although this assumption is not realistic, it is sufficient to provide us insights into the impact of delay variability on the aggregator efficiency. However, a more realistic study through network experiments will be provided in the next chapter. We analyze a scenario in which a packet from receiver l arrives at the beginning of a timer interval n and a packet from receiver m arrives at the end of the same timer interval. As regards both the periodic and triggered schemes, Figures 3.9(a) and 3.9(b), respectively, illustrate the worst case of delay variability that occurs in this scenario.

3.8.1 Periodic Scheme

In order for the merging scheme to work well, Δt must be sized sufficiently large with respect to jitter and window size variability. We approximate the cumulative jitter and variability as follows: to take jitter into account, the merger node should choose a merging time window size $\Delta\tau' = \Delta t + J_l^{max} + J_m^{max}$, as seen in Figure 3.9.a.. As the true window size can differ from the chosen value due to timer variability, the maximum possible window size is $\Delta\tau'' = \Delta\tau' + \delta^{max}$. We make no assumptions about which feedback source will be either receiver l or receiver m in each timer interval, so we make the conservative approximation $J_l^{max} + J_m^{max} = 2\max_{i=1,\ldots,k}[J_i^{max}]$. We wish for the influence of random delays to be an insignificant portion of $\Delta\tau''$. Thus Δt should be sized to be a large ratio of these delays. We define the ratio a as follows:

$$a = \frac{\Delta t}{\delta^{max} + 2\max_{i=1,\ldots,k}[J_i^{max}]}. \tag{3.33}$$

The performance is given in terms of packets replacements and merges attributable to *packet condition*. In Section 3.8.3, we investigate the performance of the aggregator as a function of ratio a. It should be noted that ratio a is determined by the aggregator and cannot be individually chosen by the receivers.

The occurrence of random delays does not change the relation between Δt and $\Delta\tau$. It does, however, introduce a lower bound on the feedback sending interval in order to achieve a desirable performance.

3.8.2 Triggered Scheme

From Figure 3.9(b), it follows that, due to jitter, the merger node should choose a merging time window size $\Delta\tau^{'}$ such that

$$\Delta\tau^{'} > \hat{\xi}_{k,optimal}\Delta t + J_l^{max} + J_m^{max}. \tag{3.34}$$

Due to timer variability, the maximum possible window size is

$$\Delta\tau^{''} = \Delta\tau^{'} + \delta^{max}. \tag{3.35}$$

To reduce the probability of packet replacements, we should make

$$\Delta\tau^{''} < \Delta t. \tag{3.36}$$

By substituting $\Delta\tau^{''}$ (3.35) in Eq. (3.36), we obtain

$$\Delta\tau^{'} + \delta^{max} < \Delta t. \tag{3.37}$$

Using $\Delta\tau^{'} = \hat{\xi}_{k,optimal}\Delta t + J_l^{max} + J_m^{max} + \theta$ as a solution for Eq. (3.34), where θ is a positive constant, Eq. (3.37) can be rewritten as

$$\Delta t > k(J_l^{max} + J_m^{max} + \delta^{max} + \theta) \tag{3.38}$$

We consider θ as small as possible. Thus, the product $k\theta$ is negligible. We make no assumptions about which feedback source will be either receiver l or receiver m in each timer interval, so we make the conservative approximation $J_l^{max} + J_m^{max} = 2\max_{i=1,\ldots,k}[J_i^{max}]$. Therefore, Δt should be sized to be a ratio of $k(\delta^{max} + 2\max_{i=1,\ldots,k}[J_i^{max}])$. Thus, we define the ratio d as follows:

$$d = \frac{\Delta t}{k(\delta^{max} + 2\max_{i=1,\ldots,k}[J_i^{max}])}. \tag{3.39}$$

Section 3.8.3 investigates the performance of the aggregator as a function of ratio d. The ratio d is also determined by the aggregation node.

The occurrence of varying delays changes the relation between Δt and $\Delta\tau$ and also introduces a lower bound on the feedback sending interval in order to achieve a desirable performance. The merging time window size we will adopt in our study is $\Delta\tau = \hat{\xi}_{k,optimal}\Delta t + 2\max_{i=1,\ldots,k}[J_i^{max}]$.

3.8.3 Results

Simulations and network experiments are of extreme importance to assess how new protocols and mechanisms perform in today's networks (LANs, WANs, Internet). In this section, we perform simulations to study how random delays affect the efficiency of the aggregation system under the triggered and periodic schemes. Simulations are used to point out certain aspects that are difficult to study in uncontrolled environments as the Internet. Our simulation results also give insights into a lower bound on the feedback sending interval that dampens the effects of random delays and thus improves performance.

The simulation results we present in this section were derived using the *ns* network simulator [63]. We have developed a modified version of the simulator for our study. We use as performance metrics the fraction of merges due to *packet condition*, packet replacements, and the Global Filtering Level. The performance is presented in function of the ratios a (periodic scheme) or d (triggered scheme) that we have previously defined. We adopt a 99% confidence level. It should be noted that in most cases confidence intervals shown in the figures are very small and can be seen as a single dot.

Simulation Scenarios

The network topology considered is shown in Figure 3.1. Each link has 5 Mbps of bandwidth and a 2 ms delay. We have performed a set of simulations for $k = 10$ and $k = 100$. In each simulation, each receiver sends approximately 600 feedback packets. The start time of each receiver is uniformly chosen at random from the interval $[0, \Delta t)$. The merging window size variation δ of each timer interval is uniformly chosen at random from the interval $[0, 10]$ ms. After sending the first feedback packet, each receiver i sends its packets spaced apart by an interval uniformly chosen at random from the interval $[\Delta t - J_i^{max}, \Delta t + J_i^{max}]$. This procedure induces maximum jitter of J_i^{max} in the feedback stream of receiver i. In the simulations, J_i^{max} is identical for all receivers. We have performed simulations for $J_i^{max} =$ [10 ms, 20 ms, 30 ms].

Periodic Scheme

Figure 3.10 shows the results for $k = 10$ and $k = 100$, with $\delta^{max} = 10\ ms$ and $J_i^{max} = 30\ ms$ for all $i \in [1, k]$. From Figure 3.10(a), we note that the fraction of merges due to *packet condition* is greatly influenced by random delays when ratio a is low. This influence increases along with the number of receivers. The reason is as follows: the occurrence of random delays may induce the arrival of two packets from the same receiver within the same timer interval. If the second packet arrives immediately after a merging procedure that has been completed due to *packet condition*, but before a timer expiration, a merging procedure due to *timeout condition* will occur. If the second packet arrives before *packet condition* holds, a packet replacement will take place. After any of these situations occurs within a timer interval, the subsequent timer interval may lack packets from the concerned receiver, which induces a merging procedure due to *timeout condition*. Therefore, as k increases, there is a greater probability that both the number of merges attributable to *timeout condition* and the number of packet replacements will also increase. Figure 3.10(b) confirms the effect of delay variability on the number of packet replacements as k increases. The Y-axis represents the percent of the total number of packets sent that have been replaced. We observe an average number of packet replacements equal to 3.6 when $k = 10$ and $a = 10$. As $k = 100$ and $a = 10$, the average number of packet replacements is 267. Increasing the value of a reduces the side effect of random delays.

Figure 3.10(c) shows the Global Filtering Level for $k = 10$ and $k = 100$. We note that the values are very close to the optimal Global Filtering Level. The more the ratio a is increased, the more the Global Filtering Level decreases towards $\xi_{k,optimal}$. For low values of a, the Global Filtering Level is less than $\xi_{k,optimal}$ because of the following: if, on the one hand, merges due to *timeout condition* lead to a Global Filtering Level less than $\xi_{k,optimal}$, on the other hand, packet replacements lead to a Global Filtering Level greater than $\xi_{k,optimal}$. Thus, merges due to *timeout condition* strongly impact the Global Filtering Level than do packet replacements.

Tables 3.1 and 3.2 summarize the outcomes of simulation studies with different J_i^{max} values. We note that the number of packet replacements is low in all cases. In particular, we note that the number of merges due to *packet condition* is greatly influenced by varying delays when $k = 100$. Increasing the ratio a reduces the side effect of varying delays. However, this implies an increase in the feedback sending interval. Assuming reasonably performance, if a feedback latency on the order of a few seconds is needed to meet the application requirements, both varying delays and k should be low. In practice, this fact might limit the applicability of the periodic scheme.

The minimum feedback sending interval that satisfies specified performance require-

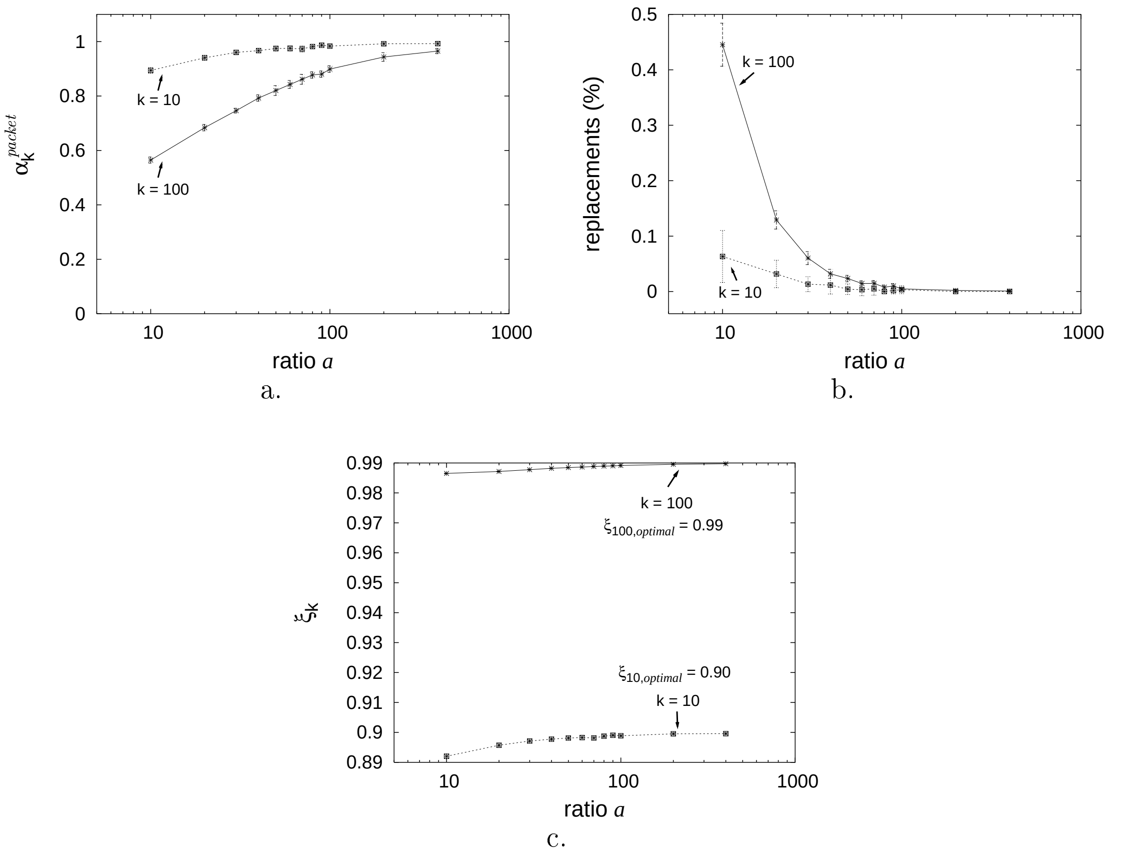


Figure 3.10: Periodic scheme ($\delta^{max} = 10\ ms$ and $J_i^{max} = 30\ ms$ for all $i \in [1, k]$): fraction of merges due to *packet condition*; percentage of packet replacements; and the Global Filtering Level.

Table 3.1: Summary of the periodic scheme results ($\delta^{max} = 10\ ms$).

		periodic scheme			
		J_i^{max} for all $i \in [1, k]$			
		$10\ ms$		$20\ ms$	
k	a	$\alpha_k^{cond(i)}$	$repl.(\%)$	$\alpha_k^{cond(i)}$	$repl.(\%)$
10	30	0.942 ± 0.004	0.005 ± 0.008	0.961 ± 0.007	0.010 ± 0.014
10	100	0.982 ± 0.001	0.000 ± 0.000	0.985 ± 0.008	0.000 ± 0.000
10	200	0.989 ± 0.004	0.000 ± 0.000	0.990 ± 0.006	0.000 ± 0.000
100	30	0.702 ± 0.007	0.042 ± 0.005	0.745 ± 0.008	0.054 ± 0.007
100	100	0.862 ± 0.005	0.004 ± 0.003	0.892 ± 0.008	0.005 ± 0.003
100	200	0.920 ± 0.010	0.001 ± 0.001	0.936 ± 0.007	0.001 ± 0.002

Table 3.2: Summary of the periodic scheme results ($\delta^{max} = 10\ ms$).

		periodic scheme	
		J_i^{max} for all $i \in [1, k]$	
		$30\ ms$	
k	a	$\alpha_k^{cond(i)}$	$repl.(\%)$
10	30	0.960 ± 0.007	0.013 ± 0.014
10	100	0.983 ± 0.006	0.003 ± 0.007
10	200	0.992 ± 0.006	0.000 ± 0.000
100	30	0.746 ± 0.009	0.060 ± 0.012
100	100	0.899 ± 0.012	0.005 ± 0.003
100	200	0.943 ± 0.016	0.002 ± 0.002

ments can be determined by the merger node. To see this, consider the following: assume that $k = 10$, the maximum jitter observed is 10 ms, $\delta^{max} = 10\ ms$, and that the current feedback sending interval is 3 seconds. Under these conditions, $a = 100$. For this scenario, the expected performance of merging procedures in terms of merges due to *packet condition* is around 98%. Now assume that the maximum jitter observed increases to 20 ms and that the performance of 98% must be maintained. From Table 3.1, we verify that this performance requirement can be met if a is equal to 100, which implies a minimum feedback sending interval of 5 seconds. In practice, a mechanism to inform receivers about the new feedback sending interval must be employed. One solution that does not require additional control packets is the message piggybacking we have presented in Section 3.6.3.

Triggered Scheme

Figure 3.11 shows the results for $k = 10$ and $k = 100$, with $\delta^{max} = 10\ ms$ and $J_i^{max} = 30\ ms$ for all $i \in [1, k]$. We note that the triggered scheme performs very well even in the presence of varying delays. The behavior of the packet replacements is shown in Figure 3.11(a). The Y-axis represents the percent of the total number of packets sent that have been replaced. We note an average value of 3.12 packet replacements when $k = 10$ and $d = 1$ ($\Delta t = 700\ ms$). When $k = 100$ and $d = 0.1$ ($\Delta t = 700\ ms$), we note the average number of 306 packet replacements. Thus, with a fixed feedback sending interval, the number of packet replacements increases as k increases. The reason is as follows: varying delays may induce the occurrence of more than one packet being received from the same receiver during the same timer interval. Thus, as k increases, the probability of packet replacements also increases. Aside from this fact, the triggered scheme still provides a low number of packet replacements in relation to the total number of packets sent.

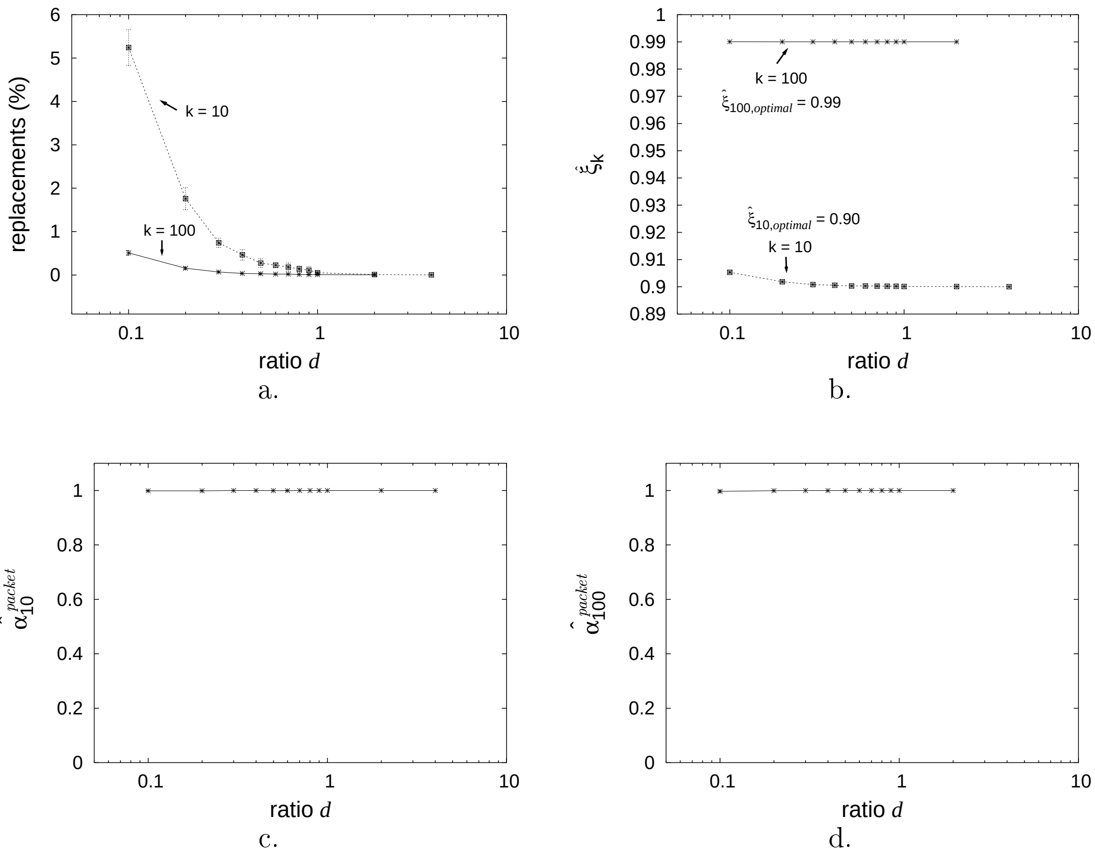


Figure 3.11: Triggered scheme ($\delta^{max} = 10\ ms$ and $J_i^{max} = 30\ ms$ for all $i \in [1, k]$): percentage of packet replacements; the Global Filtering Level; and fraction of merges due to *packet condition* for $k = 10$ and $k = 100$.

Table 3.3: Summary of the triggered scheme results ($\delta^{max} = 10\ ms$).

		triggered scheme			
		J_i^{max} for all $i \in [1, k]$			
		$10\ ms$		$20\ ms$	
k	d	$\hat{\alpha}_k^{cond(i)}$	$repl.(\%)$	$\hat{\alpha}_k^{cond(i)}$	$repl.(\%)$
10	0.1	0.998 ± 0.002	3.308 ± 0.336	0.997 ± 0.003	4.854 ± 0.216
10	0.3	0.999 ± 0.001	0.465 ± 0.094	0.999 ± 0.001	0.624 ± 0.129
10	1.0	1.000 ± 0.000	0.057 ± 0.033	0.999 ± 0.001	0.076 ± 0.040
10	2.0	1.000 ± 0.000	0.025 ± 0.022	1.000 ± 0.000	0.022 ± 0.019
100	0.1	0.998 ± 0.001	0.357 ± 0.024	0.998 ± 0.002	0.474 ± 0.046
100	0.3	0.999 ± 0.001	0.046 ± 0.012	0.999 ± 0.001	0.078 ± 0.019
100	1.0	0.999 ± 0.001	0.004 ± 0.003	1.000 ± 0.000	0.008 ± 0.002
100	2.0	1.000 ± 0.000	0.002 ± 0.002	1.000 ± 0.000	0.003 ± 0.005

Figure 3.11(b) shows the Global Filtering Level for $k = 10$ and $k = 100$. We see values that are very close to the optimal Global Filtering Level. The more the value of d increases, the more the Global Filtering Level decreases towards $\hat{\xi}_{k,optimal}$. For low values of d, the Global Filtering Level is greater than $\hat{\xi}_{k,optimal}$ for the following reason. Recall that merges due to *timeout condition* lead to a Global Filtering Level less than $\hat{\xi}_{k,optimal}$ and that packet replacements lead to a Global Filtering Level greater than $\hat{\xi}_{k,optimal}$. From Figure 3.11(c) and Figure 3.11(d), we note that the number of merging procedures due to *timeout condition* is approximately zero, which highlights the effect of packet replacements on the Global Filtering Level.

Tables 3.3 and 3.4 summarize the results of simulation studies for different J_i^{max} values. These results show that the triggered scheme provides reasonably performance even in the event of random delays.

As explained in Section 3.8.3, the minimum feedback sending interval that satisfies specified performance requirements can be determined by the merger node. We now provide an example using the triggered scheme. Assume that $k = 10$, the maximum jitter observed is 10 ms, $\delta^{max} = 10\ ms$, and that the current feedback sending interval is 300 ms. Hence, $d = 1$. Under this scenario, the expected performance of merging procedures in terms of merges due to *packet condition* is maximized. On the other hand, the expected performance in terms of packet replacements is around 0.06%. Now assume that the maximum jitter observed increases to 30 ms and that the same performance must be maintained. From Table 3.4, we verify that this performance requirement can be met if d is equal to 1, which implies a minimum feedback sending interval of 700 ms.

Table 3.4: Summary of the triggered scheme results ($\delta^{max} = 10\ ms$).

		triggered scheme	
		J_i^{max} for all $i \in [1,k]$	
		$30\ ms$	
k	d	$\hat{\alpha}_k^{cond(i)}$	$repl.(\%)$
10	0.1	0.999 ± 0.009	5.243 ± 0.412
10	0.3	0.999 ± 0.001	0.741 ± 0.108
10	1.0	1.000 ± 0.000	0.052 ± 0.026
10	2.0	1.000 ± 0.000	0.010 ± 0.021
100	0.1	0.997 ± 0.004	0.510 ± 0.056
100	0.3	1.000 ± 0.000	0.069 ± 0.019
100	1.0	0.999 ± 0.001	0.008 ± 0.004
100	2.0	0.999 ± 0.001	0.004 ± 0.004

3.9 Chapter Summary

In this chapter we have presented a formal study of the aggregation system. Based on our system model, we have introduced evaluation metrics and formally described the system behavior. We have adopted an approach for transmitting feedback that, coupled with suitable timer settings at the merger node, increases efficiency. In our approach, each receiver regularly transmits consecutive feedback packets spaced apart by Δt.

Timer settings at the aggregator were obtained for two timer schemes: periodic and triggered. Under that periodic scheme, the merging time window size ($\Delta\tau$) should be equal to the feedback sending interval (Δt). The occurrence of varying delays does not change the relation $\Delta t = \Delta\tau$. It does introduce, however, a lower bound on the feedback sending interval in order to achieve a desirable performance. In practice, we should expect notable system performance when the feedback sending interval is very large with respect to jitter. Our results suggest that, in practice, a feedback sending interval on the order of several seconds or even tens of seconds is necessary to achieve reasonable efficiency. These feedback latencies might be inadequate for several applications and so the applicability of the periodic scheme is reduced. Our results also show that, for a given feedback sending interval, the probability of generating an aggregate report that is based on information from all descendant receivers can be dramatically reduced as the number of descendant receivers grows. Although the periodic scheme has performance flaws, we should observe in practice very low percentages of packet replacements. In our study, the highest percentage of packet replacements was around 0.5%.

For the triggered scheme, timer settings at the aggregator were first obtained disregarding random delays. The occurrence of varying delays changes the relation between Δt and $\Delta\tau$. Also, it introduces a lower bound on the feedback interval in order to achieve a desirable performance. The original merging time window size obtained disregarding random delays needs to be increased by a component that is dependent upon jitter experimented by each receiver's feedback flow. The approach we adopted for the timer adaptation was pertinent to our intent of performance evaluation under the assumption that the statistical bound of jitter estimates for each incoming feedback flow at the aggregator exhibit low variance. We have reasonably increased the merging time window size by $2\max_{i=1,\ldots,k}[J_i^{max}]$. Our findings show that the triggered scheme performs much better than the periodic scheme. A feedback sending interval of only 700 ms has provided very good performance in all scenarios we have studied. The performance in the worst-case simulated scenario was then obtained using a merging time window size of around 25.1 times the statistical jitter bounds (i.e., 753 ms/30 ms). In other words, the original merging time window size ($\Delta\tau = (1-\frac{1}{100}).700\ ms = 693\ ms$) was incremented by nearly 8.7%. In practice, we should expect that an aggregator operating under the triggered scheme will be capable of summarizing feedback considering information from an overwhelming majority of its downstream neighbors.

3.10 Résumé du Chapitre

Dans ce chapitre nous avons présenté une étude formelle du système d'agrégation. Basé sur notre modèle de système, nous avons proposé de métriques d'évaluation de performances et nous avons formellement décrit le comportement du système. Nous avons proposé une approche pour le retour d'information de contrôle qui associé à un réglage approprié de l'agrégation temporelle, augmente l'efficacité. Dans notre approche, les transmissions des paquets de retour sont programmées à des intervalles réguliers. Cet intervalle est, pour chaque récepteur, égal à Δt.

Les temporisations au niveau d'un nœud d'agrégation ont été obtenues pour deux schémas : périodique et à déclenchement. Dans l´approche périodique, la fenêtre d'agrégation ($\Delta\tau$) doit être égale à l'intervalle (Δt) du retour d'information de récepteurs. La gigue ne remet pas en question la relation entre $\Delta t = \Delta\tau$. Il introduit, pourtant, une borne inférieure pour l'intervalle de retour d'information afin d'obtenir une performance souhaitée. En pratique, nous devrions nous attendre à de performances significatives du système quand l'intervalle de retour d'information est très important par rapport à la gigue. Nos résultats suggèrent qu'en pratique, un intervalle de retour d'information de l'ordre de quelques secondes voire dizaines de secondes est nécessaire pour obtenir une efficacité raisonnable. Ces délais pourraient s'avérer inappropriés pour plusieurs applications et ainsi l'applicabilité de l'approche périodique est réduite. Nos résultats démontrent aussi que, pour un intervalle de retour d'information donné, la probabilité de produire un rapport agrégé qui prenne en considération tous les récepteurs dont un nœud d'agrégation est responsable, peut être drastiquement réduite lorsque que le nombre de récepteurs descendant augmente. Bien que l'approche périodique soit faillible, nous devrions observer en pratique un très faible pourcentage de remplacements de paquets. Dans notre étude, le plus haut pourcentage de paquets remplacés que nous avons observé était d'environ 0.5%.

En ce qui concerne le schéma à déclenchement, la temporisation dans le nœud d'agrégation a été dans un premier temps obtenue sans tenir en compte des délais aléatoires. Ces délais modifient la relation entre δt et $\delta\tau$. Ils introduisent également une borne inférieure pour l'intervalle du retour d'information afin d´obtenir des performances souhaitées. La fenêtre d'agrégation obtenue sans prendre en compte les délais aléatoires doit être augmentée d'une valeur qui dépend de la gigue observée par le flux de retour d'information de chaque récepteur. L'approche que nous avons retenue pour adapter le temporisateur répondait à notre but, en supposant que la borne statistique de la gigue estimée pour chaque flux de retour d'information a une faible variance. Nous avons augmenté de façon raisonnable la largeur de la fenêtre d'agrégation de $\max_{i=1,..,k}[J_i^{max}]$. Les résultats obtenus démontrent que le schéma à déclenchement a des performances nettement supérieures à celles du schéma périodique. Un intervalle de retour d'information de seulement 700 ms nous a fourni de performances très importantes dans tous les scénarios étudiés. Ces performances dans le scénario simulé de pire de cas possible ont été obtenues en utilisant une fenêtre d'agrégation 25 fois supérieure à la borne statistique de la gigue (c´est-à-dire, 753 ms/30 ms). En d'autres termes, la largeur originale de la fenêtre d'agrégation ($\Delta\tau = (1 - \frac{1}{100}).700\ ms = 693\ ms$)) a été augmentée d´environ de 8.7%. En pratique, nous devrions nous attendre a ce qu´un noeud d'agrégation basé sur le schéma à déclenchement soit capable de résumer les informations de retour en prenant en compte une majorité importante de récepteurs dont il est responsable.

Chapter 4

Network Experiments

In the previous chapter we have studied the impact of varying delays on the aggregator performance using simulations. In this chapter, however, we are interested in network experiments. Our first intent is to get insights into jitter performing measurements in a real-world environment. Thereafter, we perform a number of network experiments that aim at providing insights into the aggregator efficiency in a more realistic scenario.

4.1 Experimental Setting

All of our experiments are set using hosts located at Paris (French) or Rio de Janeiro (Brazil) or both. The main reason for this restriction is the difficult to obtain access to hosts in other places. While there may be limitations to studying two places, our setup offers the possibility to perform inner-city and intercontinental experiments. In addition, our experimental setup coupled with tests performed at different times and days offers a large and varied scenario for our research. So we believe the research reported in this chapter is pertinent despite being limited to two places.

We obtained access to one or more hosts in the following domains: *gta.ufrj.br*, *lip6.fr*, *noos.fr*, and *free.fr*. The first domain pertains to the Network and Automation Group (*GTA*) at the Federal University of Rio de Janeiro in Brazil. The domain *lip6.fr* pertains to the *Lip6* laboratory at the Pierre and Marie Curie University in Paris, France. *GTA* and *Lip6* have high speed Internet connections. *NOOS* is a TV and Internet cable-operator in France. The host located at the *NOOS* domain has the upstream and downstream bandwidth, respectively, limited to 128 kbps and 512 kbps. *FreeFR* is an Internet Service

Provider in France. The host located at this domain uses a dial-up V.90 modem connection.

4.2 Jitter Measurements

We have performed a great number of jitter measurements at different times and days using as *source-destination* of packets *GTA-NOOS*, *Lip6-NOOS*, *GTA-FreeFR*, *Lip6-FreeFR*, and *FreeFR-NOOS*. Here we report the detailed behavior for a number of experiments; the behavior of the other was similar. Hosts on Lip6 and GTA could not be used as destination of packets due to firewall rules. For each experiment, we attempt to trace the path from the source to the destination of packets using the *traceroute*[1] utility. *Traceroute* sends a sequence of datagrams with Time-To-Live (TTL) set to 1,2, etc., and uses the ICMP "time exceeded" messages that are returned to determine routers along the path. Three datagrams are sent at each TTL setting and a line is printed showing the TTL, address of the router and RTT of each test. If answers come from different routers, the address of each responding system is printed. If there is no response within 5 seconds, a "*" is printed for that test.

4.2.1 Iperf

Our objective here is to perform active measurements using the *iperf* utility [64]. Using *iperf*'s verbose option we collected measurements of jitter between two hosts. Iperf creates a constant bit rate UDP stream. The default datagram size is 1470 bytes. We use a datagram flow of 300 kbps whenever the destination host is placed at the NOOS domain, otherwise a datagram flow of 25 kbps is used. *Iperf* is set to print a jitter estimate every 1 second. Jitter calculations are continuously computed by the destination host, as specified by RTP in RFC 1889 [65]. Basically, the sender records a 64 bit timestamp in the datagram. The destination host computes the relative transit time as (*destination's receive time - sender's send time*). The sender and the destination hosts do not need to have synchronized clocks because any difference is subtracted out in the jitter calculation. Jitter is the smoothed mean of differences between consecutive transit times.

Formally, jitter estimates are constructed by timestamps as follow: let t_i be the timestamp of the ith packet, *i.e.* the time packet was generated by the client. Let r_i be the time packet i is received by the server. Let $D(i-1,i) = (r_i - r_{i-1}) - (t_i - t_{i-1})$ be a sample of jitter calculated whenever a reception of a pair of packets. An estimate of jitter is given by

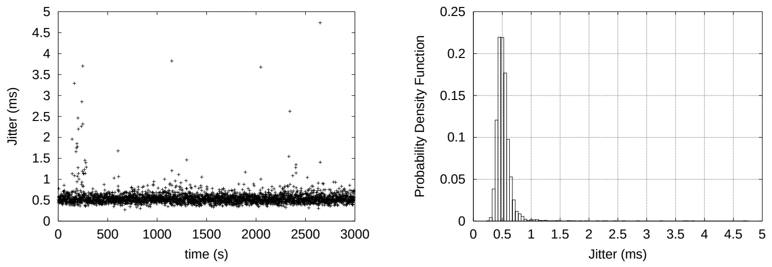
[1]For a survey of network measurement tools including *traceroute* see the Cooperative Association for Internet Data Analysis (CAIDA). Web page http://www.caida.org/Tools/

$$J_i = (1-u)J_{i-1} + u(|D(i-1,i)|) \qquad (4.1)$$

where u is a fixed constant. Thus J_i is a smoothed average of the jitter samples $D(1,2)$, ..., $D(i-1,i)$. The estimate places more weight on the recently observed jitters than on the observed jitters of the distant past. The fixed constant u is equal to $1/16$. This value gives a good immunity to short bursts while maintaining a reasonable rate of convergence [66]. This form of estimate is similarly used to estimate round-trip times in TCP [58].

Inner-Paris

Figure 4.1(a) shows a scatter plot of jitter measurements from Lip6 (132.227.110.56) to NOOS (81.65.248.127) on February 22, 2004 (a Sunday) at 05:42am (CET[2]). What is more important is that the experiment was run in the early morning on a weekend. So, we were expecting to measure low jitter, and thus low queuing delays. Absolutely, we note that most of jitter estimates are less than 1 ms. The maximum jitter estimate is around 5 ms. The distribution of jitter measurements is shown in Figure 4.1(b). The traceroute of the path *Lip6-NOOS* is presented in Table 4.1. Some reasons for the absence of RTT estimates concerning to the host number 8 follow: packets to/from the hop number 8 were lost; the host has not responded in time; or the host has not responded. The last reason is plausible because routers in the Internet can be configured to limit the generation rate of ICMP messages or even to not send ICMP "time exceeded" messages at all.



(a) (b)

Figure 4.1: Iperf results from Lip6 to NOOS on February 22, 2004 at 05:42am (CET). (a) scatter plot of jitter estimates. (b) density of jitter estimates.

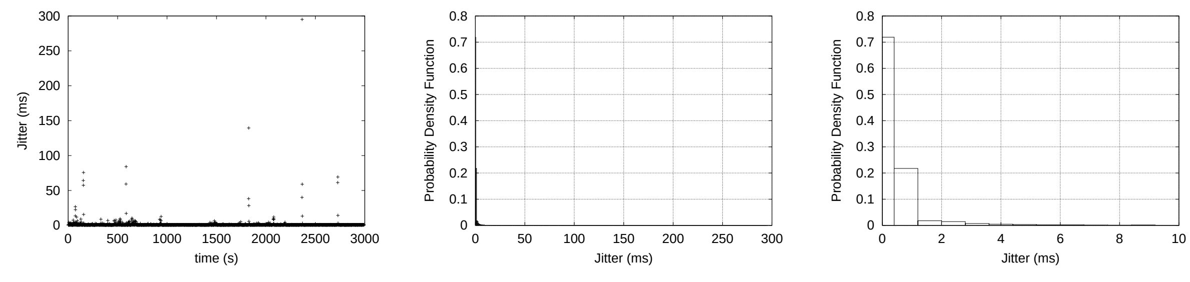
Figure 4.2(a) shows a scatter plot of jitter measurements from Lip6 (132.227.110.56) to NOOS (81.65.249.206) on February 23, 2004 (a Monday) at 01:56pm (CET). What is more

[2] CET (Central Europe Time) is 1 hour ahead of Coordinated Universal Time (UTC). Note that during summer daylight saving time/summer time is observed, and CEST (Central Europe Summer Time) is used instead.

Table 4.1: Traceroute output from Lip6 to NOOS on February 22, 2004 at 05:42am (CET).

Hop	Host	RTT (ms)	RTT (ms)	RTT (ms)
1	132.227.74.254 (132.227.74.254)	5.785	2.241	2.152
2	r-olympe (132.227.109.254)	2.251	4.585	2.163
3	r-jusren.reseau.jussieu.fr (134.157.254.126)	2.594	2.997	2.500
4	rap-jussieu.rap.prd.fr (195.221.127.181)	62.237	5.767	3.973
5	gw-rap.rap.prd.fr (195.221.126.78)	112.598	183.721	108.420
6	193.51.181.102 (193.51.181.102)	58.855	35.640	21.818
7	nri-b-pos12-0.cssi.renater.fr (193.51.180.158)	9.830	8.474	4.693
8		*	*	*
9	gama-vlan406.noos.net (212.198.0.131)	5.828	4.679	5.138
10	CMTS-Mon32A.noos.net (212.198.1.237)	5.897	6.711	4.856
11	m127.net81-65-248.noos.fr (81.65.248.127)	14.321	20.829	20.276

important is that the experiment was run in a daytime on a weekday. So, we were expecting to measure higher jitter. Figure 4.2(b) shows the resulting distribution and Figure 4.2(c) highlights the distribution from 0 to 10 ms. A general observation is that although there are estimates on the order of hundred of milliseconds, more than 99% jitter estimates are lower than 2 ms. The maximum jitter measurement is around 300 ms. Table 4.2 presents the traceroute output for this experiment. Once again, there are no RTT estimates for host number 8. This leads us to suspect of an ICMP rate-limiting mechanism or even an explicit configuration to not generate ICMP messages.



(a) (b) (c)

Figure 4.2: Iperf results from Lip6 to NOOS on February 23, 2004 at 01:56pm (CET). (a) scatter plot of jitter estimates. (b) density of jitter estimates. (c) zoom on the density of jitter estimates.

Table 4.2: Traceroute output from Lip6 to NOOS on February 23, 2004 at 01:56pm (CET).

Hop	Host	RTT (ms)	RTT (ms)	RTT (ms)
1	132.227.74.254 (132.227.74.254)	2.720	2.245	2.102
2	r-olympe (132.227.109.254)	2.822	2.692	2.171
3	r-jusren.reseau.jussieu.fr (134.157.254.126)	2.761	2.993	2.085
4	rap-jussieu.rap.prd.fr (195.221.127.181)	4.633	4.104	2.967
5	gw-rap.rap.prd.fr (195.221.126.78)	2.726	5.650	15.580
6	193.51.181.102 (193.51.181.102)	6.757	3.732	4.969
7	nri-b-pos12-0.cssi.renater.fr (193.51.180.158)	3.727	4.341	4.132
8		*	*	*
9	gama-vlan406.noos.net (212.198.0.131)	5.434	4.930	4.985
10	CMTS-Mon32A.noos.net (212.198.1.237)	10.952	6.590	7.605
11	m127.net81-65-248.noos.fr (81.65.248.127)	19.594	13.382	23.663

Figure 4.3(a) shows a scatter plot of jitter measurements from Lip6 (132.227.110.56) to FreeFR (213.228.54.116) on February 26, 2004 (a Thursday) at 02:43am (CET). We were initially expecting to measure low jitter, and thus low queueing delays. It turned out, however, that it is not the case. We note that most of jitter estimates are around 20 ms. The maximum jitter estimate is around 170 ms. The distribution of jitter measurements is shown in Figure 4.3(b). A possible explanation for high jitter estimates is as follows: jitter is generally caused by congestion in the IP network. In this experiment, however, the dial-up connection used by the destination of packet has potentially a great contribution. In Point-to-Point Protocol (PPP) encapsulation, jitter is almost always due to serialization delay. As shown in Table 4.3, the amount of serialization delay (the time it takes to actually place the bits onto an interface) introduced on a low-speed link can be significant. The fact that the destination of packets has a reception rate limited to about 33.6 kbps[3] might significantly contribute to inflate jitter estimates. Table 4.4 shows RTT estimates for the experiment. We note that RTT estimates from the host at Lip6 to the host at FreeFR (hop 11) is very high with respect to estimates concerning to the other hops.

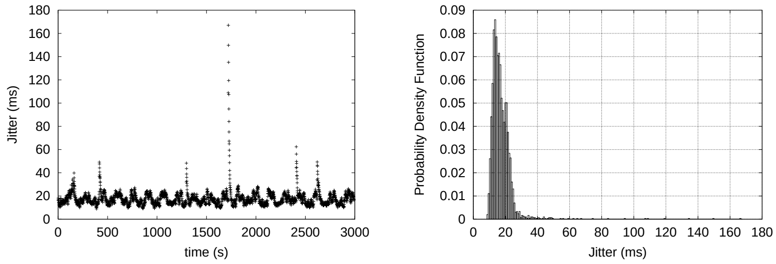
Remote Continents

Figure 4.4(a) shows a scatter plot of jitter measurements from GTA (146.164.69.28) to NOOS (81.65.248.127) on February 22, 2004 (a Sunday) at 06:40am (CET). What is more

[3] V.90 modem technologies allow a maximum downstream rate of 56 kbps. The upstream rate is limited to 33.6 kbps.

Table 4.3: Serialization delay for various frame sizes on low-speed links.

	1 byte	64 bytes	128 bytes	256 bytes	512 bytes	1024 bytes	1500 bytes
33.6 kbps	238 us	15.2 ms	30.5 ms	61 ms	121.9 ms	243.8 ms	357.1 ms
56 kbps	143 us	9 ms	18 ms	36 ms	72 ms	144 ms	214 ms



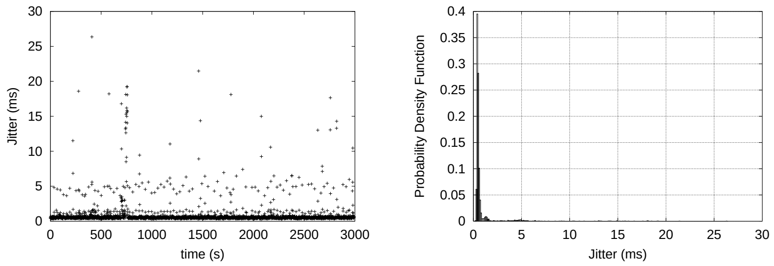
(a) (b)

Figure 4.3: Iperf results from Lip6 to FreeFR on February 26, 2004 at 02:34am (CET). (a) scatter plot of jitter estimates. (b) density of jitter estimates.

Table 4.4: Traceroute output from Lip6 to FreeFR on February 26, 2004 at 02:34am (CET).

Hop	Host	RTT (ms)	RTT (ms)	RTT (ms)
1	132.227.74.254 (132.227.74.254)	6.082	2.301	2.226
2	r-olympe (132.227.109.254)	2.351	3.054	2.433
3	r-jusren.reseau.jussieu.fr (134.157.254.126)	3.026	2.847	2.395
4	rap-jussieu.rap.prd.fr (195.221.127.181)	3.528	4.078	8.600
5	gw-rap.rap.prd.fr (195.221.126.78)	4.491	3.748	4.001
6	193.51.181.102 (193.51.181.102)	4.114	3.461	8.043
7	nri-b-pos12-0.cssi.renater.fr (193.51.180.158)	4.269	3.487	4.011
8	free-telecom.sfinx.tm.fr (194.68.129.223)	4.532	5.570	5.410
9	cbv-6k-2-a0.routers.proxad.net (212.27.32.195)	28.099	20.515	8.518
10	nas-cbv-4-a8.routers.proxad.net (213.228.2.68)	8.353	6.050	5.325
11	nas-cbv-4-213-228-54-116.dial.proxad.net (213.228.54.116)	358.691	149.012	222.194

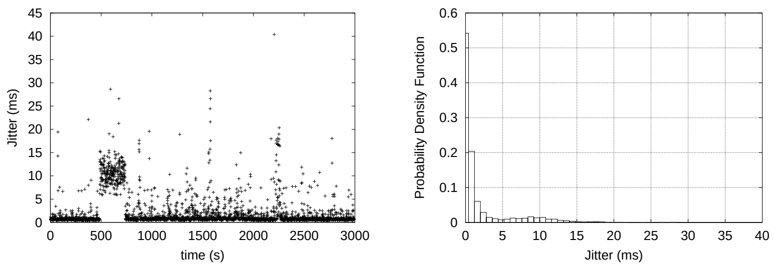
important is that during the experiment, it is late night on a weekend in South America and early morning on a weekend in Europe[4]. The output of the traceroute (see Table 4.5) indicates that the packets follow to North America before reaching Europe. During experiments, it is late night in North America. We were expecting that routers on the path would not be overloaded and so jitter estimates would be low. The distribution of jitter is shown in Figure 4.4(b). We note that the important majority of jitter estimates is lower than 1 ms.



(a) (b)

Figure 4.4: Iperf results from GTA to NOOS on February 22, 2004 at 06:40am (CET). (a) scatter plot of jitter estimates. (b) density of jitter estimates.

Figure 4.5(a) shows a scatter plot of jitter measurements from GTA (146.164.69.28) to NOOS (81.65.250.155) on Mars 05, 2004 (a Friday) at 05:17pm (CET). What is more important is that during the experiment, it is daytime on a weekday in both South America and Europe. In addition, it is also daytime in North America. Consequently, we expect that routers on the path carry a larger fraction of traffic from remote continents. The distribution of jitter is shown in Figure 4.5(b). We note that most jitter estimates are less than **12 ms**. Table 4.6 presents the traceroute output for this experiment.



(a) (b)

Figure 4.5: Iperf results from GTA to NOOS on Mars 05, 2004 at 05:17pm (CET). (a) scatter plot of jitter estimates. (b) density of jitter estimates.

[4]The relative time between continents is based on the time in cities involved in the experiment.

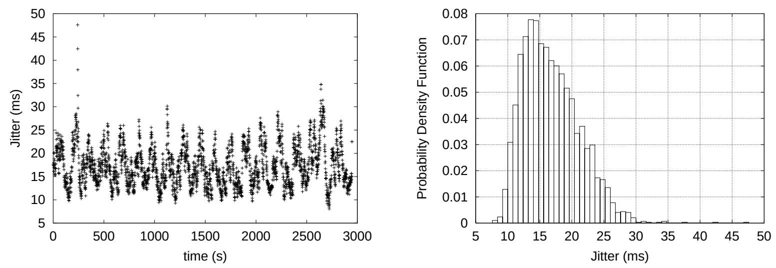
Table 4.5: Traceroute output from GTA to NOOS on February 22, 2004 at 06:40am (CET).

Hop	Host	RTT (ms)	RTT (ms)	RTT (ms)
1	angra-1 (146.164.69.1)	0.613	0.386	0.292
2	bcast-ext (146.164.69.254)	1.154	0.926	0.861
3	rt-nce2.ufrj.br (146.164.1.5)	2.027	2.661	2.388
4	200.20.96.58 (200.20.96.58)	3.704	2.205	2.320
5	200.179.69.29 (200.179.69.29)	3.037	2.162	2.647
6	ebt-G8-0-core01.rjo.embratel.net.br (200.244.63.203)	3.619	3.008	2.725
7	200.244.140.233 (200.244.140.233)	2.654	2.755	3.032
8	so-2-1.hsa2.Miami1.Level3.net (64.156.8.33)	110.607	110.945	111.037
9	ge-6-1-1.mp2.Miami1.Level3.net (64.159.1.101)	111.451	111.021	110.988
10	so-0-1-0.bbr2.Washington1.Level3.net (64.159.0.230)	136.987	136.815	136.575
11	so-0-0-0.mp2.London2.Level3.net (212.187.128.133)	209.247	209.222	209.805
12	so-3-0-0.mp2.Paris1.Level3.net (212.187.128.37)	215.652	215.827	215.406
13	ge-11-2.ipcolo1.Paris1.Level3.net (212.73.240.117)	216.813	216.425	216.432
14	unknown.Level3.net (212.73.204.110)	216.442	216.469	216.057
15	195.132.16.77 (195.132.16.77)	215.672	216.064	215.729
16	195.132.16.85 (195.132.16.85)	216.590	217.141	216.552
17	gama-vlan406.noos.net (212.198.0.131)	216.357	216.582	216.804
18	CMTS-Mon32A.noos.net (212.198.1.237)	217.232	216.943	217.274
19	m127.net81-65-248.noos.fr (81.65.248.127)	230.527	239.777	224.939

Table 4.6: Traceroute output from GTA to NOOS on Mars 05, 2004 at 05:17pm (CET).

Hop	Host	RTT (ms)	RTT (ms)	RTT (ms)
1	angra-1 (146.164.69.1)	0.628	0.272	0.213
2	bcast-ext (146.164.69.254)	0.953	0.776	0.816
3	rt-nce2.ufrj.br (146.164.1.5)	4.027	2.871	3.313
4	200.20.96.58 (200.20.96.58)	3.652	39.599	2.928
5	200.179.69.29 (200.179.69.29)	3.091	3.229	3.240
6	ebt-G4-0-core03.rjo.embratel.net.br (200.244.63.236)	3.772	3.522	2.679
7	200.244.140.237 (200.244.140.237)	4.309	3.983	2.475
8	so-2-1.hsa2.Miami1.Level3.net (64.156.8.33)	111.760	111.478	111.471
9	ge-6-0-1.mp2.Miami1.Level3.net (64.159.0.21)	112.413	113.675	113.261
10	so-0-1-0.bbr2.Washington1.Level3.net (64.159.0.230)	137.804	137.068	138.361
11	so-0-0-0.mp2.London2.Level3.net (212.187.128.133)	209.869	210.476	210.130
12	so-1-0-0.mp1.Paris1.Level3.net (212.187.128.41)	216.596	217.060	216.886
13	ge-10-2.ipcolo1.Paris1.Level3.net (212.73.240.101)	219.176	217.463	217.020
14	unknown.Level3.net (212.73.204.110)	228.824	216.688	217.145
15	195.132.16.77 (195.132.16.77)	217.891	217.131	216.987
16	195.132.16.85 (195.132.16.85)	217.727	217.120	217.467
17	gama-vlan406.noos.net (212.198.0.131)	217.758	217.456	216.965
18	CMTS-Mon32A.noos.net (212.198.1.237)	221.147	223.551	223.980
19	m155.net81-65-250.noos.fr (81.65.250.155)	224.944	228.761	228.147

Figure 4.6(a) shows a scatter plot of jitter measurements from GTA (146.164.69.28) to FreeFR (213.228.54.116) on February 26, 2004 (a Thursday) at 03:41am (CET). What is more important is that during the experiment, it is late night on a weekday in both South America and Europe. So, we not expect to get jitter estimates very different from estimates that we obtained in the experiment *Lip6-FreeFR* (see Figure 4.3). The distribution of jitter estimates is shown in Figure 4.6(b). In effect, we do not observe much difference. Jitter estimates are concentrated in the interval [10, 30] ms. The results suggest that jitter estimates are inflated because of the serialization delay problem, as discussed so far, and not because it is an intercontinental experiment. Table 4.7 shows the output of the traceroute utility for the experiment. The absence of some RTT estimates may be due to packet loss or an ICMP rate-limiting mechanism.



(a) (b)

Figure 4.6: Iperf results from GTA to FreeFR on February 26, 2004 at 03:41am (CET). (a) scatter plot of jitter estimates. (b) density of jitter estimates.

4.2.2 Independent implementation

In this section we closely examine jitter estimates and interarrival time of reports using an independent implementation. Our implementation offers the possibility to gather data using different report sizes and different report sending intervals as well. The server process awaits the client to connect using TCP. The control connection is used by the server to inform the client the size of reports, the interval between consecutive reports, and the number of reports to be issued. The client process accepts and handles the commands issued by the server. Reports are sent using UDP. After sending the requested number of reports, the client asks the server whether additional reports are necessary to complete the desired number of measurements. This procedure is due to occasional lost of reports. Thereafter, the client proceeds according to the server's response.

We define **probe** as a single measurement of jitter or interarrival interval for a given report size. The measurement (one probe) involves two consecutive received reports in order

Table 4.7: Traceroute output from GTA to FreeFR on February 26, 2004 at 03:41am (CET).

Hop	Host	RTT (ms)	RTT (ms)	RTT (ms)
1	angra-1 (146.164.69.1)	0.498	0.232	0.181
2	rt-ct-bloco-H.ufrj.br (146.164.5.193)	1.421	*	1.539
3	* rt-nce2.ufrj.br (146.164.1.5)		2.391	2.061
4	200.20.96.58 (200.20.96.58)	2.795	2.085	*
5	200.179.69.29 (200.179.69.29)	2.505	2.678	2.637
6		*	*	*
7	ebt-P13-0-core01.spo.embratel.net.br (200.230.1.122)	9.089	*	*
8	ebt-P6-0-intl03.spo.embratel.net.br (200.230.0.13)	10.973	11.035	11.243
9	sl-gw11-orl-2-3.sprintlink.net (144.232.236.129)	146.826	146.994	*
10	* sl-bb21-orl-0-0.sprintlink.net (144.232.2.233)		146.978	135.116
11	* sl-bb22-orl-15-0.sprintlink.net (144.232.2.150)		146.917	146.237
12	sl-bb20-atl-10-2.sprintlink.net (144.232.19.129)	322.280	218.894	234.444
13	sl-bb20-nyc-10-0.sprintlink.net (144.232.8.89)	165.327	165.195	153.214
14	sl-bb22-lon-12-0.sprintlink.net (144.232.9.162)	219.355	*	*
15	sl-bb20-lon-12-0.sprintlink.net (213.206.128.52)	308.905	259.331	219.753
16	sl-bb21-par-14-0.sprintlink.net (213.206.129.70)	251.950	227.239	226.434
17	sl-gw11-par-15-0.sprintlink.net (217.118.224.47)	286.473	*	226.927
18	* sle-iliad-2-0.sprintlink.net (217.118.239.150)		233.874	233.678
19	nas-cbv-4-a8.routers.proxad.net (213.228.2.68)	234.825	234.386	233.896
20	nas-cbv-4-213-228-54-116.dial.proxad.net (213.228.54.116)	381.074	439.120	439.967

of sequence. This in turn means that we avoid to make estimates when packet replication, out-of-order delivery, or gaps (packet loss) in the feedback flow occur. We discuss these different factors that we consider to gather data.

- Packet replication: the delivery of multiple copies of the same packet. It occurs rarely and it is possibly due to link-level retransmissions. These retransmissions are generally considered "pathological" network behavior and so we discard replicated packets.

- Out-of-order delivery: the result of route alternation. Route alternation is the tendency for the path between two hosts to change over time, typically oscillating over a small set of possibilities. The result is often out-of-order packet delivery. Paxson has shown that route oscillations is common on the Internet [67]. His findings indicate that the majority (91%) of Internet routes persist for hours. However, there are still a significant number of routes that alternate more quickly. Out-of-order delivery is common during route alternation because packets are frequently sent closer together. We expect that the time spacing between consecutive issued feedback packets will be at least on the order of hundreds of milliseconds, contributing to reduce the occurrence of out-of-order delivery. The same expectation helps to explain why we prefer to avoid jitter estimates using out-of-order packets. Consider the following example. Suppose that two consecutive packets are issued spaced apart by 500 ms. Assuming that they arrive out-of-order and spaced apart by 10 ms, the jitter sample is this case is equal to 510 ms. This is enormous and leads to a short burst in jitter estimates. To avoid this, we choose do not take out-of-order packets into account in jitter estimates. Even though bursts in jitter estimates may occur.

  In practice, an aggregator must discard out-of-order packets. In this manner, an aggregation procedure does not use information that is older than information used in a previous aggregation procedure. Additionally, if out-of-order packets arrive during the same merging procedure, an aggregator will only use the most recent information. This is in accordance with the packet replacement advice we have presented so far in the Section 3.1.

- Packet loss: an indication of persistent congestion. This situation can lead to short bursts in jitter estimates. Since we are not interested in exploring the worst case jitter, we avoid to make jitter estimates using packets that are not consecutive.

In our implementation, we change the gain parameter u used to construct jitter estimates. *Iperf* utility sets the gain parameter u to $1/16$. Given that the aggregator will not get very frequent jitter measurements and thus old measurements are likely to be outdated, a higher value of $u = 1/4$ is used.

Feedback packets are generally very short [18, 22, 23, 57] but the resulting data from an aggregation procedure may require longer packets (for instance, when aggregation results in the feedback data distribution). We wish for avoiding fragmentation of reports. IP fragmentation occurs when a router must forward a datagram onto an interface with a maximum transmission unit (MTU) smaller than the size of the packet. Fragmentation puts additional burden on routers, which must expend resources to split the packets, as well as end hosts, which must hold state waiting for all fragments and then reassemble the original packet after all fragments have arrived. If a single fragment of the fragment series is lost in transit, the other fragments are useless since the original packet cannot be reassembled. Because the Internet standard for MTU is 576 bytes, fragmentation is unlikely using a Maximum Segment Size[5] (MSS) of 548 bytes, 8 bytes of UDP header, and 20 bytes of IP datagram header. In this manner, we restrict the size of reports in our evaluation.

Methodology

We have measured jitter and interarrival interval of reports using different UDP datagram sizes: 23 bytes, 100 bytes, 200 bytes, 300 bytes, 400 bytes, and 500 bytes. For each datagram size, we have performed 1,000 probes. The sending interval of reports was set to 500 ms. The set of probes with a given datagram size takes approximately 8 minutes. The whole set of probes is about 50 minutes. The exact duration of the probes is dependent upon network conditions of packet loss.

Inner-Paris

Figures 4.7 to 4.9 show measurements we collected from Lip6 to NOOS on April 02, 2004 (a Friday) at 05:33pm (CEST). The traceroute from source to destination is shown in Table 4.8. The lost of reports was not observed during measurements. Figures 4.7(a) and 4.7(b) show, respectively, a scatter plot of interarrival interval and the resulting probability density function. We note that the interarrival interval of reports is normally distributed with mean equal to the sending interval of reports.

Figure 4.8 shows the behavior of jitter estimates in about 50 minutes. It should be noted that jitter is not dependent upon packet size in ours tests. The differences we observe in jitter for the different packet sizes is due to varying network conditions over time. We note that although jitter estimates are very noise, they are relative low with few points lying above the lower band of the majority of data. During measurements for the 400-

[5]The Maximum Segment Data is equivalent to the maximum amount of user data in the packet.

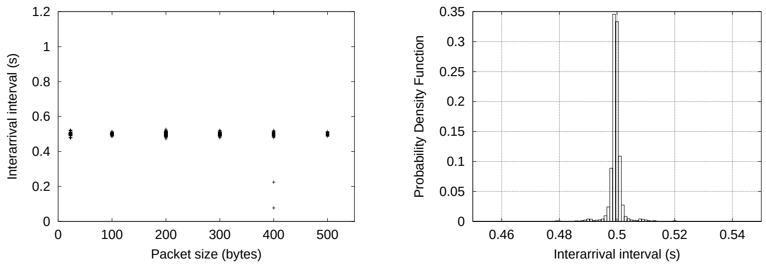
Table 4.8: Traceroute output from Lip6 to NOOS on April 02, 2004 at 05:33pm (CEST).

Hop	Host	RTT (ms)	RTT (ms)	RTT (ms)
1	132.227.74.254 (132.227.74.254)	2.735	5.587	9.324
2	r-olympe (132.227.109.254)	5.945	2.306	2.190
3	r-jusren.reseau.jussieu.fr (134.157.254.126)	2.407	2.312	6.424
4	rap-jussieu.rap.prd.fr (195.221.127.181)	4.723	3.857	8.073
5	gw-rap.rap.prd.fr (195.221.126.78)	4.004	4.086	4.456
6	193.51.181.102 (193.51.181.102)	3.956	3.381	3.448
7	nri-b-pos12-0.cssi.renater.fr (193.51.180.158)	4.393	3.575	3.579
8	noos.sfinx.tm.fr (194.68.129.253)	4.231	4.958	4.247
9	cartier-vlan406.noos.net (212.198.0.130)	4.802	4.343	3.999
10	CMTS-Mon32A.noos.net (212.198.1.237)	5.575	7.053	6.611
11	m155.net81-65-250.noos.fr (81.65.250.155)	15.764	13.161	14.453

byte reports, we note a burst in jitter. However, this abrupt growing quickly decreases. In general, jitter burst tracks imminent or transient congestion while packet loss tracks persistent congestion [33].

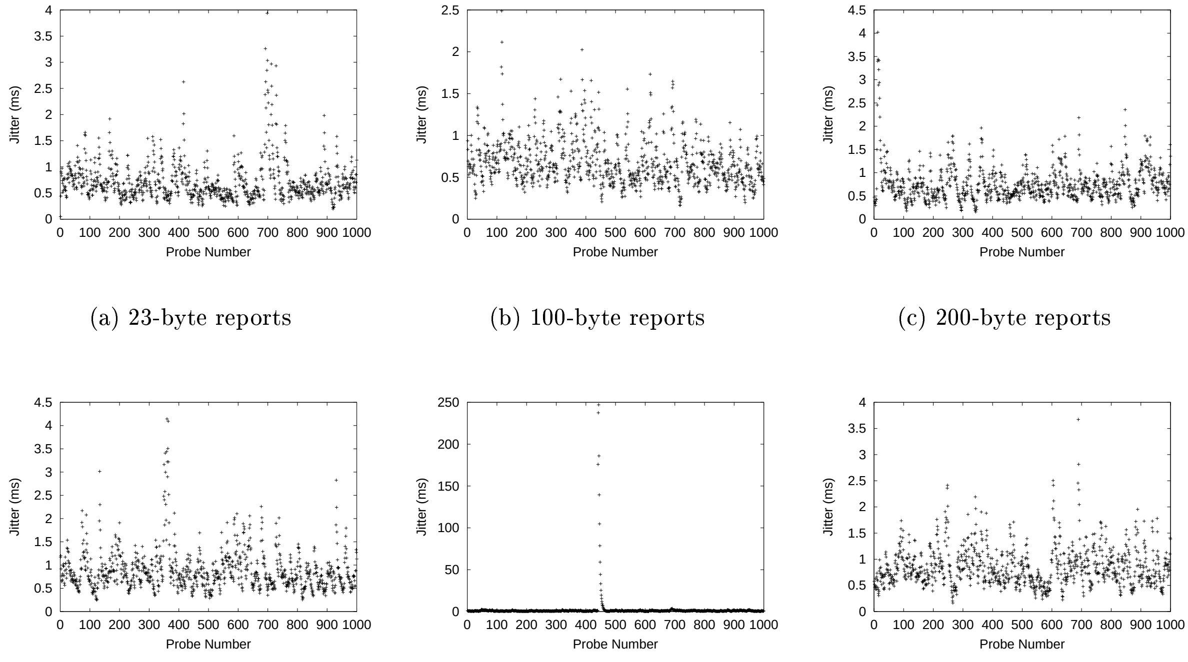
The distribution of jitter estimates is heavy-tailed and so observations from the tail are rare. Figure 4.9(a) shows the probability density function of jitter estimates in the experiment using reports of 100 bytes. We note that most jitter estimates are lower than 1.5 ms. Figure 4.9(b) shows the probability density function of jitter estimates in the the experiment for reports of 400 bytes. Network conditions have introduced more jitter during these experiments. We note that most jitter estimates are less than 2 ms as seen in Figure 4.9(c), which highlights from 0 to 2.5 milliseconds the distribution presented in 4.9(b).

Figures 4.10 to 4.12 show measurements we collected from Lip6 to FreeFR on April 06, 2004 (a Tuesday) at 06:11am (CEST). The traceroute from source to destination is shown in Table 4.9. The graphs show significant differences with respect to previous results. First, as seen in Figure 4.10(b), the interarrival interval is not normally distributed. The gaps in the distribution is potentially due to the serialization-delay issue we have discussed. Second, jitter estimates show much more dynamic network conditions as shown in Figure 4.11. Even though, report loss was not observed during the measurements. Finally, all the distributions of jitter estimates exhibit heavy-tail but present longer tail. Figure 4.12 shows jitter distribution with respect to experiments using reports of 200 and 400 bytes.



(a) (b)

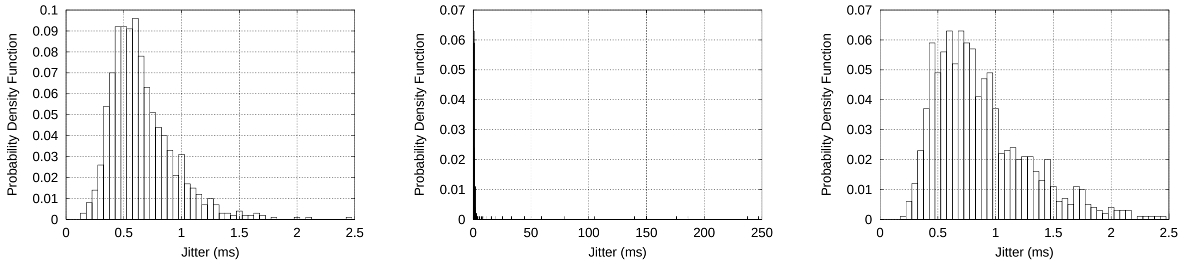
Figure 4.7: Lip6-NOOS on April 02, 2004 (a Friday) at 05:33pm (CEST). (a) scatter plot of interarrival intervals and (b) resulting distribution.



(a) 23-byte reports (b) 100-byte reports (c) 200-byte reports

(d) 300-byte reports (e) 400-byte reports (f) 500-byte reports

Figure 4.8: Jitter estimation in accordance with the probe number. Lip6-NOOS on April 02, 2004 (a Friday) at 05:33pm (CEST).

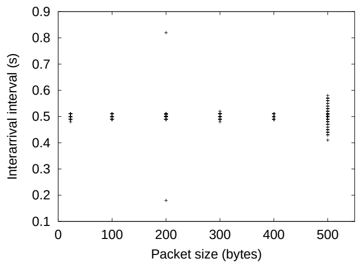


(a) 100-byte reports (b) 400-byte reports (c) 400-byte reports

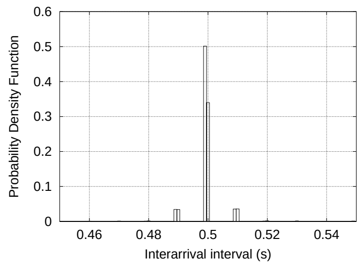
Figure 4.9: Jitter Distributions. Lip6-NOOS on April 02, 2004 (a Friday) at 05:33pm (CEST)

Table 4.9: Traceroute output from Lip6 to FreeFR on April 06, 2004 at 06:11am (CEST).

Hop	Host	RTT (ms)	RTT (ms)	RTT (ms)
1	132.227.74.254 (132.227.74.254)	2.792	2.306	2.104
2	r-olympe (132.227.109.254)	2.253	2.243	2.166
3	r-jusren.reseau.jussieu.fr (134.157.254.126)	6.547	2.572	2.726
4	rap-jussieu.rap.prd.fr (195.221.127.181)	32.826	3.914	4.525
5	gw-rap.rap.prd.fr (195.221.126.78)	5.893	3.990	11.325
6	193.51.181.102 (193.51.181.102)	4.457	3.578	5.379
7	nri-b-pos12-0.cssi.renater.fr (193.51.180.158)	3.826	4.673	4.255
8	free-telecom.sfinx.tm.fr (194.68.129.223)	4.320	4.359	3.632
9	cbv-6k-2-a0.routers.proxad.net (212.27.32.195)	4.840	4.434	4.064
10	nas-cbv-3-a8.routers.proxad.net (213.228.2.67)	5.870	5.426	4.549
11	nas-cbv-3-213-228-56-61.dial.proxad.net (213.228.56.61)	139.596	126.653	115.798

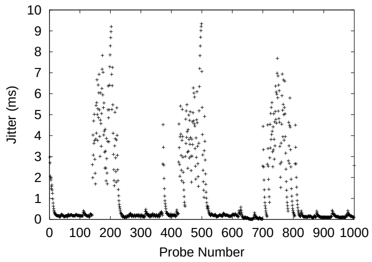


(a)

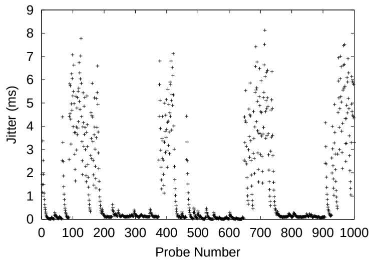


(b)

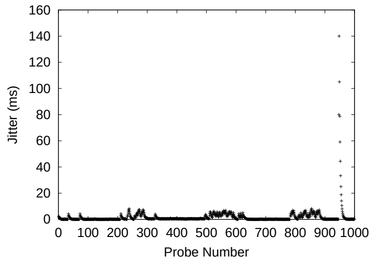
Figure 4.10: Lip6-FreeFR on April 06, 2004 (a Tuesday) at 06:11am (CEST). (a) scatter plot of interarrival intervals and (b) resulting distribution.



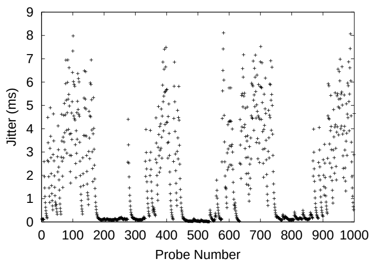
(a) 23-byte reports



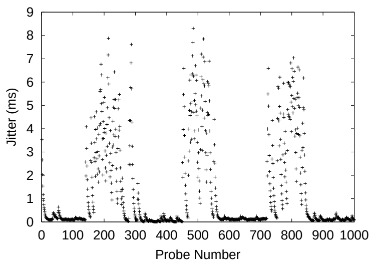
(b) 100-byte reports



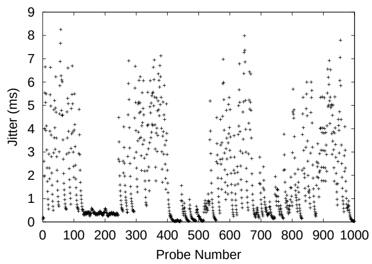
(c) 200-byte reports



(d) 300-byte reports

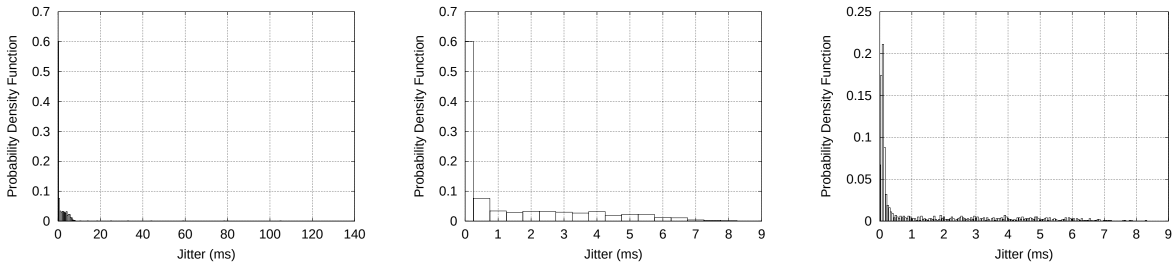


(e) 400-byte reports



(f) 500-byte reports

Figure 4.11: Jitter estimate x probe number. Lip6-FreeFR on April 06, 2004 (a Tuesday) at 06:11am (CEST).



(a) 200-byte reports (b) 200-byte reports (c) 400-byte reports

Figure 4.12: Jitter Distributions. Lip6-FreeFR on April 06, 2004 (a Tuesday) at 06:11am (CEST).

Figures 4.13 to 4.15 show measurements we collected from FreeFR to NOOS on April 06, 2004 (a Tuesday) at 05:05am (CEST). Only two reports were lost during measurements. The traceroute from source to destination is shown in Table 4.10. We note higher RTT values, which may indicate that routers through the path *FreeFR-NOOS* operate under high traffic conditions.

The experiments also reveal interesting results. Figure 4.13(a) is a scatter plot of interarrival intervals. We note that the interarrival of reports can highly skew from the original report sending interval. However, these events were rare and probably were the result of congestion. The distribution of interarrival intervals in Figure 4.13(b) is atypical but without gaps. A potential reason for such a behavior is the serialization-delay issue associated with network congestion.

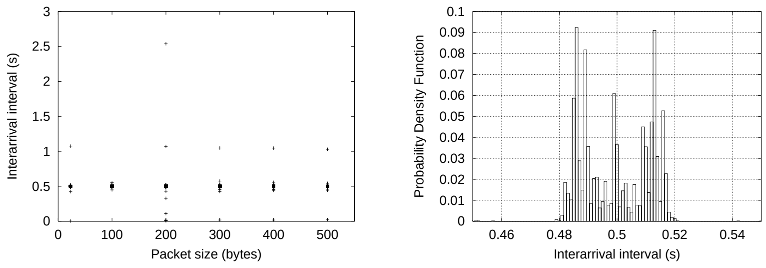
Figure 4.14 shows the behavior of jitter estimates in about 50 minutes. Except during the experiment using 23-byte reports, all the other experiments reveal jitter measurements lying around or above 10 ms. Figure 4.15 shows the distribution of jitter estimates collected during the tests with reports of 100 and 200 bytes.

Remote Continents

Figures 4.16 to 4.18 show measurements we collected from GTA to NOOS on Mars 07, 2004 (a Sunday) at 12:20pm (CET). Only nine reports were lost during measurements. The traceroute from source to destination is shown in Table 4.11. Figures 4.16(a) and 4.16(b), respectively show the scatter plot of interarrival intervals and the resulting distribution. Figure 4.17 shows the behavior of jitter estimates during the experiment. We note that jitter estimates are very low although bursts in jitter are observed in some probes. In particular,

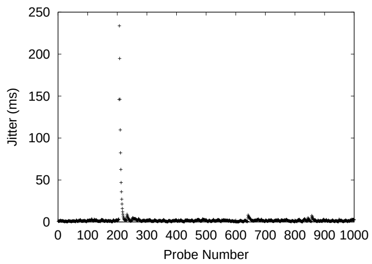
Table 4.10: Traceroute output from FreeFR to NOOS on April 06, 2004 at 05:05am (CEST).

Hop	**Host**	**RTT (ms)**	**RTT (ms)**	**RTT (ms)**
1	192.168.254.254 (192.168.254.254)	128.629	119.782	119.920
2	213.228.2.126 (213.228.2.126)	119.906	119.864	119.916
3	213.228.3.238 (213.228.3.238)	119.971	119.878	119.895
4	195.132.16.77 (195.132.16.77)	119.993	119.879	119.907
5	195.132.16.85 (195.132.16.85)	119.930	119.853	119.909
6	212.198.0.130 (212.198.0.130)	119.926	119.854	119.952
7	212.198.1.237 (212.198.1.237)	119.890	119.832	119.910
8	81.65.249.7 (81.65.249.7)	130.254	129.897	129.870

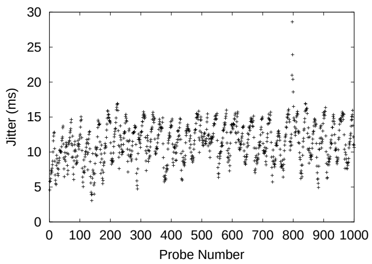


(a) (b)

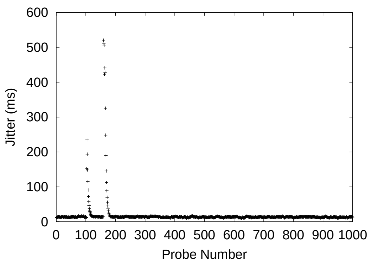
Figure 4.13: FreeFR-NOOS on April 06, 2004 (a Tuesday) at 05:05am (CEST). (a) scatter plot of interarrival intervals and (b) resulting distribution.



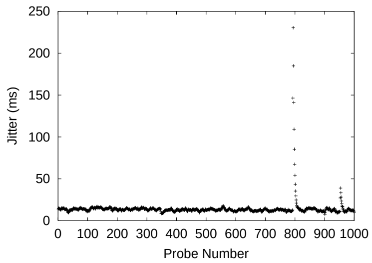
(a) 23-byte reports



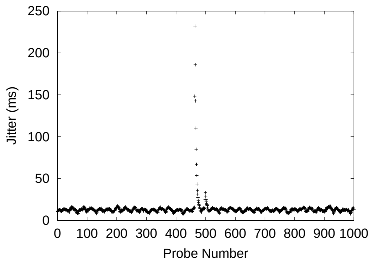
(b) 100-byte reports



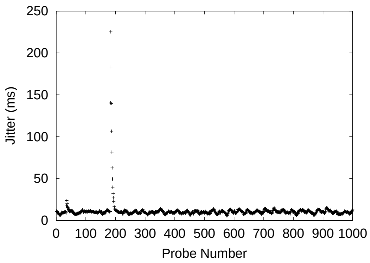
(c) 200-byte reports



(d) 300-byte reports

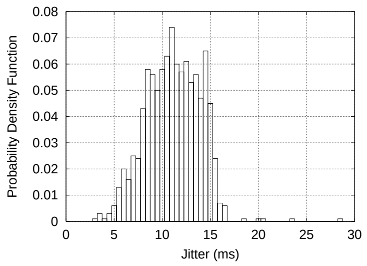


(e) 400-byte reports

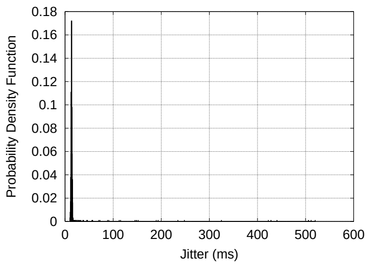


(f) 500-byte reports

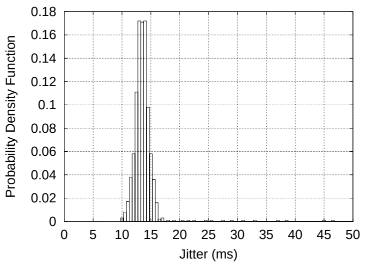
Figure 4.14: Jitter estimate x probe number. FreeFR-NOOS on April 06, 2004 (a Tuesday) at 05:05am (CEST).



(a) 100-byte reports



(b) 200-byte reports



(c) 200-byte reports

Figure 4.15: Jitter Distributions. FreeFR-NOOS on April 06, 2004 (a Tuesday) at 05:05am (CEST).

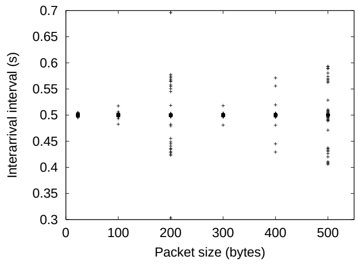
jitter estimates shown in Figure 4.17(f) exhibit periods of relative stability followed by short-duration bursts. The relative constant interval between bursts may be due to global traffic synchronization on a link. Figure 4.18 shows the distribution of jitter estimates collected during experiments with reports of 100 and 500 bytes. We note low jitter estimates and heavy-tailed distributions.

Table 4.11: Traceroute output from GTA to NOOS on Mars 07, 2004 at 12:20pm (CET).

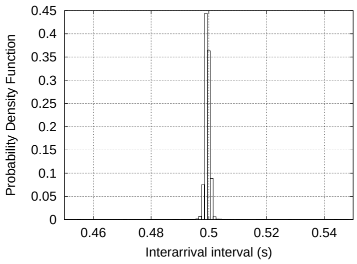
Hop	Host	RTT (ms)	RTT (ms)	RTT (ms)
1	angra-1 (146.164.69.1)	0.668	0.357	0.278
2	fundao (146.164.69.254)	1.001	0.883	0.848
3	rt-nce2.ufrj.br (146.164.1.5)	1.514	1.563	1.553
4	200.20.96.58 (200.20.96.58)	1.936	2.393	1.993
5	200.179.69.29 (200.179.69.29)	2.488	2.587	2.716
6	ebt-G4-0-core03.rjo.embratel.net.br (200.244.63.236)	2.577	3.046	2.652
7	200.244.140.237 (200.244.140.237)	2.380	2.499	2.149
8	so-2-1.hsa2.Miami1.Level3.net (64.156.8.33)	110.697	111.088	111.201
9	ge-6-0-1.mp2.Miami1.Level3.net (64.159.0.21)	111.253	111.156	111.391
10	so-0-1-0.bbr2.Washington1.Level3.net (64.159.0.230)	136.483	136.293	136.588
11	so-0-0-0.mp2.London2.Level3.net (212.187.128.133)	209.136	209.341	209.795
12	so-1-0-0.mp1.Paris1.Level3.net (212.187.128.41)	215.600	225.120	216.147
13	ge-10-2.ipcolo1.Paris1.Level3.net (212.73.240.101)	215.799	215.758	215.955
14	unknown.Level3.net (212.73.204.110)	215.815	215.842	215.927
15	195.132.16.77 (195.132.16.77)	215.995	216.233	215.843
16	195.132.16.85 (195.132.16.85)	216.669	216.356	216.223
17	gama-vlan406.noos.net (212.198.0.131)	216.541	216.626	217.036
18	CMTS-Mon32A.noos.net (212.198.1.237)	217.270	216.820	217.865
19	m155.net81-65-250.noos.fr (81.65.250.155)	228.139	228.009	226.512

Figures 4.19 to 4.20 show measurements we collected from GTA to NOOS on April 01, 2004 (a Thursday) at 04:16pm (CEST). Only thirty-two reports were lost during measurements. The traceroute from source to destination is shown in Table 4.12. Jitter estimates reveal a great surprise here; although experiments were conducted during daytime on a weekday, jitter estimates are very low. In all cases, most jitter estimates are less than 2 ms.

Figures 4.22 to 4.24 show measurements we collected from GTA to FreeFR on April 06, 2004 (a a Tuesday) at 05:12am (CEST). Only ten reports were lost during measurements. The traceroute from source to destination is shown in Table 4.13. Figure 4.22 shows a scatter plot of interarrival intervals and the resulting distribution. In particular, an atypical

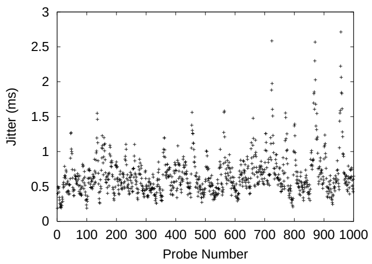


(a)

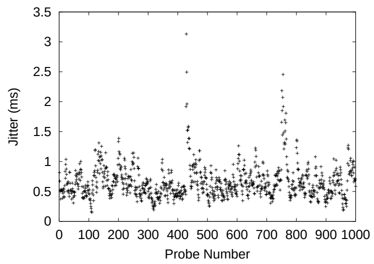


(b)

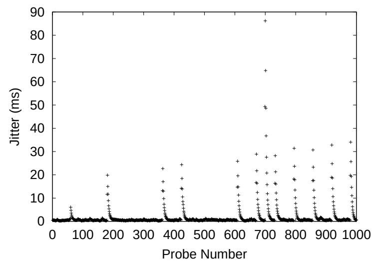
Figure 4.16: GTA-NOOS on Mars 07, 2004 (a Sunday) at 12:20pm (CET). (a) scatter plot of interarrival intervals and (b) resulting distribution.



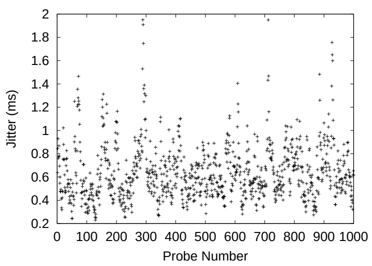
(a) 23-byte reports



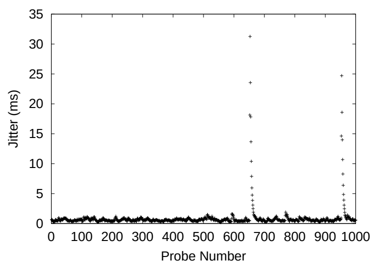
(b) 100-byte reports



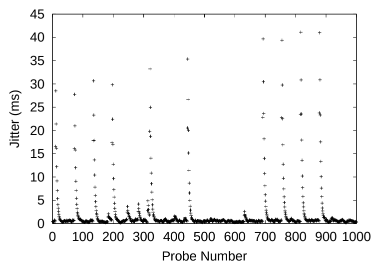
(c) 200-byte reports



(d) 300-byte reports

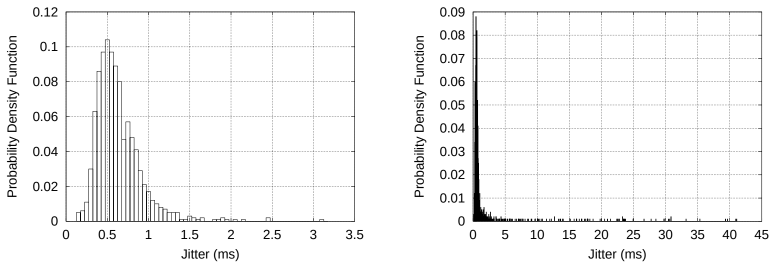


(e) 400-byte reports



(f) 500-byte reports

Figure 4.17: Jitter estimate versus probe number. GTA-NOOS on Mars 07, 2004 (a Sunday) at 12:20pm (CET).

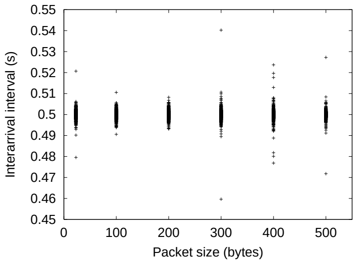


(a) 100-byte reports (b) 500-byte reports

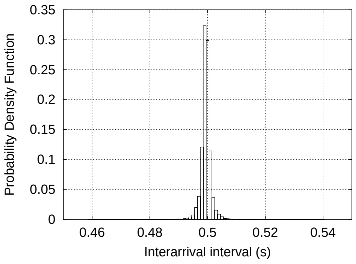
Figure 4.18: Jitter Distributions. GTA-NOOS on Mars 07, 2004 (a Sunday) at 12:20pm (CET).

Table 4.12: Traceroute output from GTA to NOOS on April 01, 2004 at 04:16pm (CEST).

Hop	**Host**	**RTT (ms)**	**RTT (ms)**	**RTT (ms)**
1	angra-1 (146.164.69.1)	0.725	0.309	0.406
2	bcast-ext (146.164.69.254)	0.990	0.840	0.725
3	rt-nce2.ufrj.br (146.164.1.5)	2.846	2.192	7.284
4	200.20.96.58 (200.20.96.58)	8.187	5.322	3.141
5	200.179.69.29 (200.179.69.29)	6.532	5.608	5.339
6	ebt-G10-0-core01.rjo.embratel.net.br (200.244.63.235)	149.180	195.046	7.217
7	200.244.140.233 (200.244.140.233)	5.225	5.264	7.269
8	so-2-1.hsa2.Miami1.Level3.net (64.156.8.33)	113.904	116.054	112.567
9	ge-6-0-1.mp2.Miami1.Level3.net (64.159.0.21)	113.831	113.266	113.592
10	so-0-1-0.bbr2.Washington1.Level3.net (64.159.0.230)	139.025	139.840	138.972
11	212.187.128.133 (212.187.128.133)	212.590	211.447	211.169
12	212.187.128.41 (212.187.128.41)	217.402	221.975	225.680
13	212.73.240.101 (212.73.240.101)	219.849	216.985	223.863
14	212.73.204.110 (212.73.204.110)	223.308	218.212	226.169
15	baffin-ge-0300.noos.net (195.132.16.77)	224.296	221.078	219.864
16	* * *			
17	cartier-vlan406.noos.net (212.198.0.130)	217.394	219.065	220.598
18	CMTS-Mon32A.noos.net (212.198.1.237)	223.640	218.308	218.211
19	m155.net81-65-250.noos.fr (81.65.250.155)	229.122	228.210	236.315

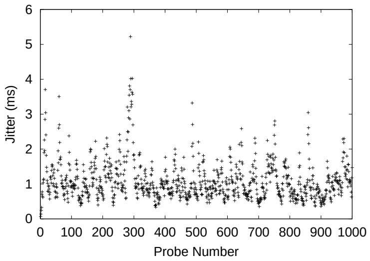


(a)

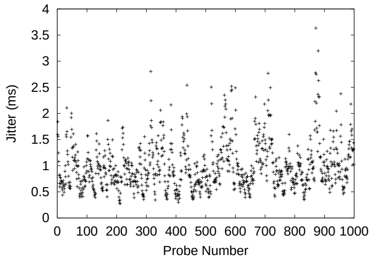


(b)

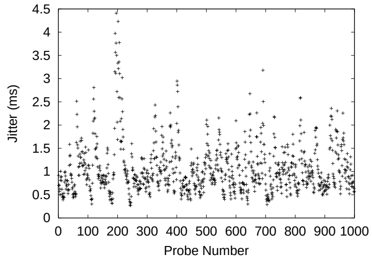
Figure 4.19: GTA-NOOS on April 01, 2004 (a Thursday) at 04:16pm (CEST). (a) scatter plot of interarrival intervals and (b) resulting distribution.



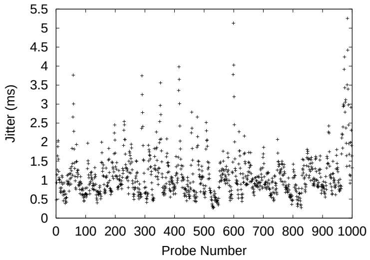
(a) 23-byte reports



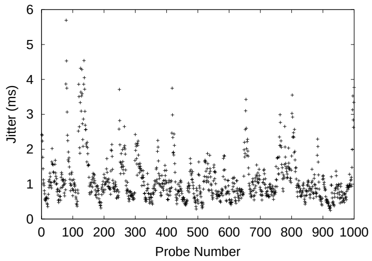
(b) 100-byte reports



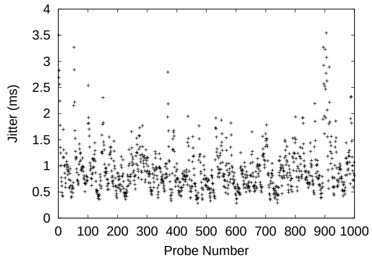
(c) 200-byte reports



(d) 300-byte reports

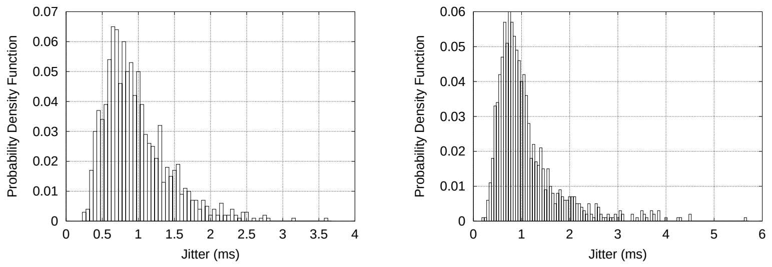


(e) 400-byte reports



(f) 500-byte reports

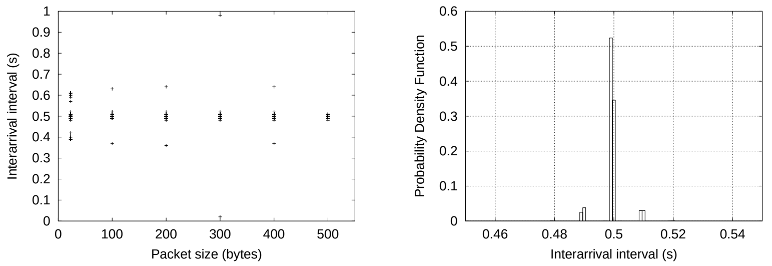
Figure 4.20: Jitter estimate in accordance with probe number. GTA-NOOS on April 01, 2004 (a Thursday) at 04:16pm (CEST).



(a) 100-byte reports (b) 400-byte reports

Figure 4.21: Jitter Distributions. GTA-NOOS on April 01, 2004 (a Thursday) at 04:16pm (CEST).

distribution is observed again. The behavior of jitter estimates over time is presented in Figure 4.23. Between the probe numbers 400 and 650, during measurements with reports of 23 bytes, we note that jitter measurements exhibit periods of relative stability followed by short-duration bursts. A relative constant interval between bursts is observed again. Figure 4.24 shows the distribution of jitter estimates for experiments with reports of 23, 400, and 500 bytes. We note that most jitter estimates are low and that the distributions present very long tails.



(a) (b)

Figure 4.22: GTA-FreeFR on April 06, 2004 (a a Tuesday) at 05:12am (CEST). (a) scatter plot of interarrival intervals and (b) resulting distribution.

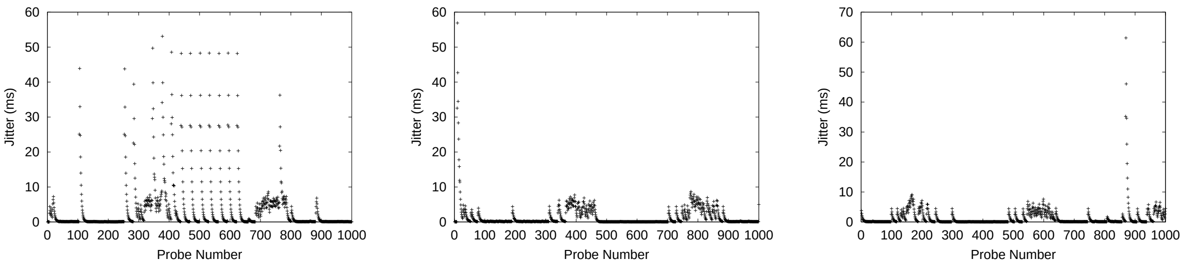
4.2.3 Comments

We finish our study with brief comments concerning our results.

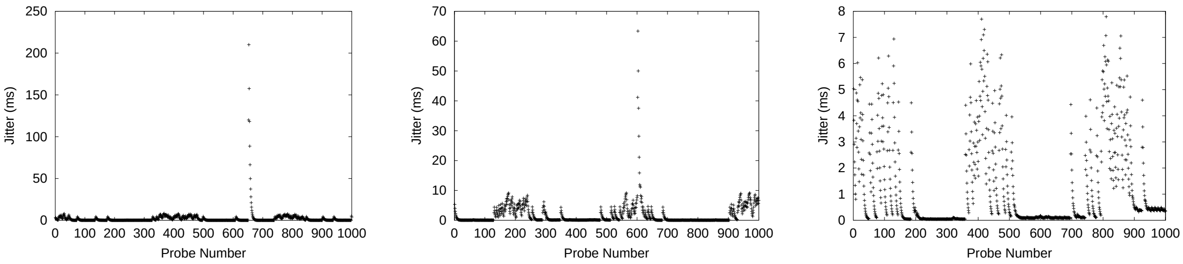
- There is a definite trend revealed by our results. They suggest that a report sending

Table 4.13: Traceroute output from GTA to FreeFR on April 06, 2004 at 05:12am (CEST).

Hop	Host	RTT (ms)	RTT (ms)	RTT (ms)
1	angra-1 (146.164.69.1)	0.727	0.325	0.270
2	bcast-ext (146.164.69.254)	1.135	0.937	0.910
3	rt-nce2.ufrj.br (146.164.1.5)	2.997	2.617	2.239
4	200.20.96.58 (200.20.96.58)	4.302	2.771	2.330
5	200.179.69.29 (200.179.69.29)	2.837	3.183	2.818
6	ebt-G8-0-core01.rjo.embratel.net.br (200.244.63.203)	95.792	107.849	202.709
7	200.244.140.229 (200.244.140.229)	2.745	3.105	3.972
8	sl-gw13-orl-2-0.sprintlink.net (160.81.90.165)	136.299	134.324	134.410
9	sl-bb21-orl-14-0.sprintlink.net (144.232.2.187)	265.636	151.571	204.522
10	sl-bb22-orl-15-0.sprintlink.net (144.232.2.150)	134.535	135.419	126.021
11	sl-bb20-atl-10-2.sprintlink.net (144.232.19.129)	289.828	223.817	123.167
12	sl-bb20-nyc-10-0.sprintlink.net (144.232.8.89)	152.310	142.001	157.079
13	sl-bb22-lon-12-0.sprintlink.net (144.232.9.162)	206.335	205.264	205.372
14	sl-bb20-lon-12-0.sprintlink.net (213.206.128.52)	224.496	205.209	206.549
15	sl-bb21-par-14-0.sprintlink.net (213.206.129.70)	213.219	213.466	212.618
16	sl-gw11-par-15-0.sprintlink.net (217.118.224.47)	213.028	212.905	213.109
17	sle-iliad-2-0.sprintlink.net (217.118.239.150)	220.548	221.501	221.322
18	nas-cbv-3-a8.routers.proxad.net (213.228.2.67)	220.728	*	221.793
19	nas-cbv-3-213-228-56-61.dial.proxad.net (213.228.56.61)	353.375	347.902	337.670

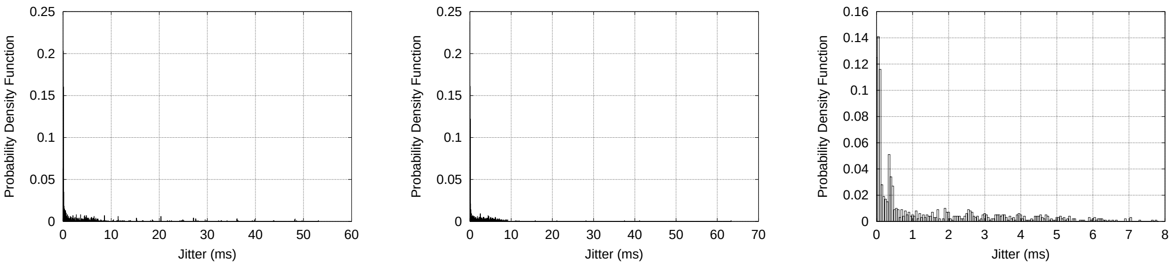


(a) 23-byte reports (b) 100-byte reports (c) 200-byte reports



(d) 300-byte reports (e) 400-byte reports (f) 500-byte reports

Figure 4.23: Jitter estimate versus probe number. GTA-FreeFR on April 06, 2004 (a a Tuesday) at 05:12am (CEST).



(a) 23-byte reports (b) 400-byte reports (c) 500-byte reports

Figure 4.24: Jitter Distributions. GTA-FreeFR on April 06, 2004 (a a Tuesday) at 05:12am (CEST)

interval of 500 ms is stable enough to be used in many situations in practice. In all the experiments with our independent implementation, the interarrival interval of reports was very close to the report sending interval. This characteristic is important because the aggregator depends on it to improve efficiency.

- The distribution of jitter estimates in the experiments has typically presented heavy-tail characteristics. In simple terms, it means that jitter can be very large but such an event is rare.

- Jitter estimates from our experiments are not intended to be a statistical sample of the Internet but they are probably typical of many paths.

4.3 An Application-Level Aggregator

In this section, we study the impact of a real-world experiment on the aggregator efficiency. We have implemented an application-level aggregator that serves as a proof-of-concept prototype. The current version of our prototype is based on an *open-loop* feedback approach. In practice, this means that the aggregator cannot inform the receivers about changes in the reporting interval. Although this is not perfect, we hope that it is sufficient to demonstrate the feasibility and suitability of our timer setting approaches. In addition, our implementation demonstrates that aggregators can be easily deployed at end-systems and so without requiring changes to existing network infrastructure.

4.3.1 Timer settings at the aggregator

So far, we have investigated the relation between the feedback sending interval and the merging time window size in the presence of random delays. For the periodic scheme, the occurrence of varying delays does not change the relation $\Delta t = \Delta \tau$. This relation is so used in the experiments with the periodic scheme. For the triggered scheme, the occurrence of varying delays changes the relation between Δt and $\Delta \tau$. The original merging time window size obtained when there are no random delays needs to be increased by a component that is dependent upon jitter experimented by each receiver's feedback flow. Under the assumption that the statistical bound of jitter estimates for each incoming feedback flow at the aggregator exhibit low variance, we have fairly increased the merging time window size by $2\max_{i=1,\dots,k}[J_i^{max}]$. In a real-world scenario, however, jitter estimates for each incoming feedback flow at the aggregator may highly diverge. The problem would consist of summarizing the set of jitter estimates into a single number that will be used to adapt the merging time window size. Statistical techniques [68] can be used to obtain such a number but we wish for avoiding complexity. In addition, there is a need to limit the merging time window size to some value. We solve these issues in a simple and effective manner. Our simulation results for the triggered scheme (see Section 3.9) have shown that, in the worst-case simulated scenario, the system efficiency is improved if the original merging time window size is incremented by around 8.7%. In this manner, we make $\Delta\tau = \hat{\xi}_{k,optimal}\Delta t + 2\max_{i=1,\ldots,k}[J_i^{max}]$ but limited to $1.1\hat{\xi}_{k,optimal}\Delta t$.

4.3.2 Performance metric: representativeness

We need to evaluate to what extent an aggregator can generate upstream feedback that represents the whole set of its descendant receivers. We introduce then a new performance metric. We define *representativeness*, which is referred as *rp-x*, as the percentage of merging procedures that have used information from at least $100x$% of the aggregator's descendant receivers. For instance, an *rp-0.98*= 0.99 means that 99% of upstream feedback information is based on feedback information from at least 98% of the aggregator's descendant receivers. It should be noted that *rp-1* is equivalent to the percentage of merging procedures due to *packet condition*.

4.3.3 Detailed experimental setting

As said so far, all the experiments are set using hosts located at Paris (French) and Rio de Janeiro (Brazil). In the experiments throughout this section, the aggregator is located at Paris. Receivers are emulated by independent processes that are distributed over 5 hosts: 2 hosts in Rio de Janeiro and the other 3 hosts in Paris. Figure 4.25 shows the detailed experimental setting and how processes are spread over the different hosts. All hosts use *Linux* as operating system.

4.3.4 Experimental Results

We have performed a number of experiments using 500 and 1,000 emulated receivers. Each experiment takes around 20 minutes. Reporting intervals of 500 ms and 2,000 ms were used. The lowest reporting interval was selected based on findings in Section 4.2.2. The Central Process Unit (CPU) load of the aggregator during experiments was at most 4%. Although this load is not significant for an application-level aggregator, lower loads can be achieved if the aggregation module is built in the kernel. The experiments we made between Monday 05 April, 2004 at 10:00 CEST and Friday 28 May, 2004 at 03:00 CEST.

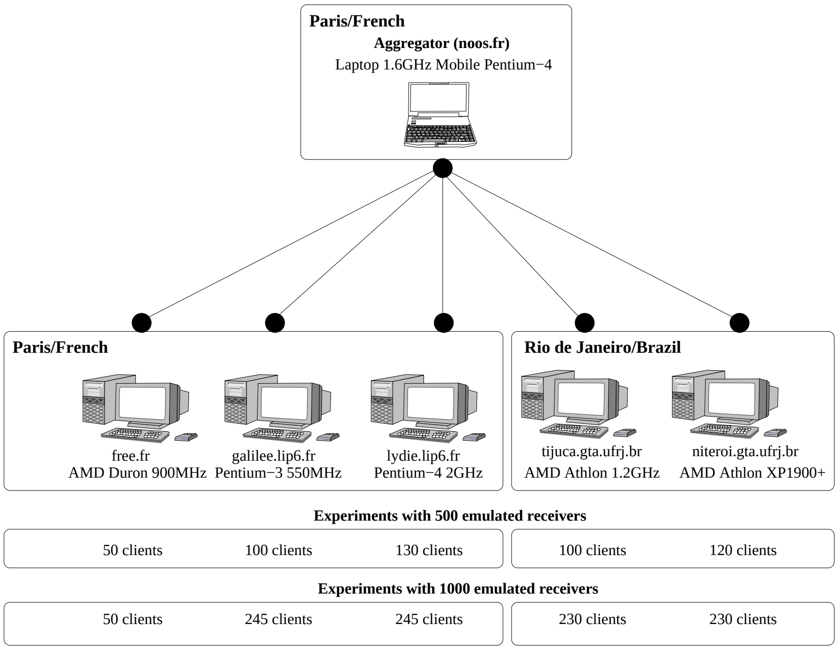


Figure 4.25: Detailed experimental setup.

Periodic scheme

Figure 4.26 shows experimental results for the periodic scheme when the aggregator is responsible for continuously merging packets from 500 receivers. The reporting interval in the experiments is set to 500 ms. It is important to note that up to 1.44% of packet loss is observed in the experiments. Packet loss militates against efficiency and so our results are influenced in an unfair way. The number of merging procedures in the experiments is around 2,600. This fact coupled with a number of packet losses that varies from 6,743 to 17,308 reduces the probability of merging procedures due to *packet condition*. We note that in most experiments around 20% of merges occur due to *packet condition* (*i.e. rp-1*). Performance exhibits some stability from *rp-0.90* on. Representativeness *rp-0.90* is around 90% in most experiments. This means that 90% of merging procedures were performed considering information of at least 450 receivers. We note at most 0.54% of packet replacements.

Figure 4.27 shows experimental results for the periodic scheme when the aggregator is responsible for continuously merging packets from 1,000 receivers. The reporting interval in the experiments is set to 2,000 ms. We observe performance improvements with respect to the previous experiments. This is due to the longer reporting interval and a more favorable condition of packet loss. Merges due to *packet condition* is around 50% and packet replacements are at most 0.04%. Performance exhibits some stability from *rp-0.98* on. Representativeness *rp-0.98* is around 85% in most experiments. This means that 85% of merging procedures were performed considering information of at least 490 receivers. We note at most 0.04% of packet replacements.

Triggered scheme

Figure 4.28 shows experimental results for the triggered scheme when the aggregator is responsible for continuously merging packets from 500 receivers. The reporting interval in the experiments is set to 500 ms. Although results are negatively affected by packet loss, performance gains of the triggered scheme are notable. In particular, experiment 3 observes fewer packet losses and so it exhibits the best performance. This experiment has 71.5% of merges due to *packet condition*. In addition, representativeness *rp-99* is nearly 96%, that is, 96% of merging procedures have accessed information from at least 495 receivers. Representativeness *rp-90* is nearly 100% in all experiments. The worst-case percentage of packet replacements is close to 6.8%. This value is somewhat inflated. The reason packet losses make the aggregator to wait for information that will not arrive. This in turn maximizes the waiting time to aggregate and gives the opportunity to more than one packet from some receivers to arrive before merging. So that, packet replacements are more likely. We expect so a lower percentage of packet replacements in better conditions of packet loss. As a final observation, our experimental results make evident the suitability and effectiveness of our timer setting approaches to enhance the aggregator efficiency.

4.4 Chapter Summary

This chapter was dedicated to real-world network experiments. Network experiments are very important to assess how new protocols and mechanisms perform in today's networks. We have collected jitter estimates from inner-Paris and intercontinental experiments as well. The results have provided us with insights into jitter and interarrival interval characteristics of feedback at an aggregator. We have developed the prototype of an aggregation node and presented a study of the impact of real-world experiments on the aggregator efficiency. The experimental findings are fairly in accordance with our expectation. The demonstration of how to achieve an improved system efficiency was done.

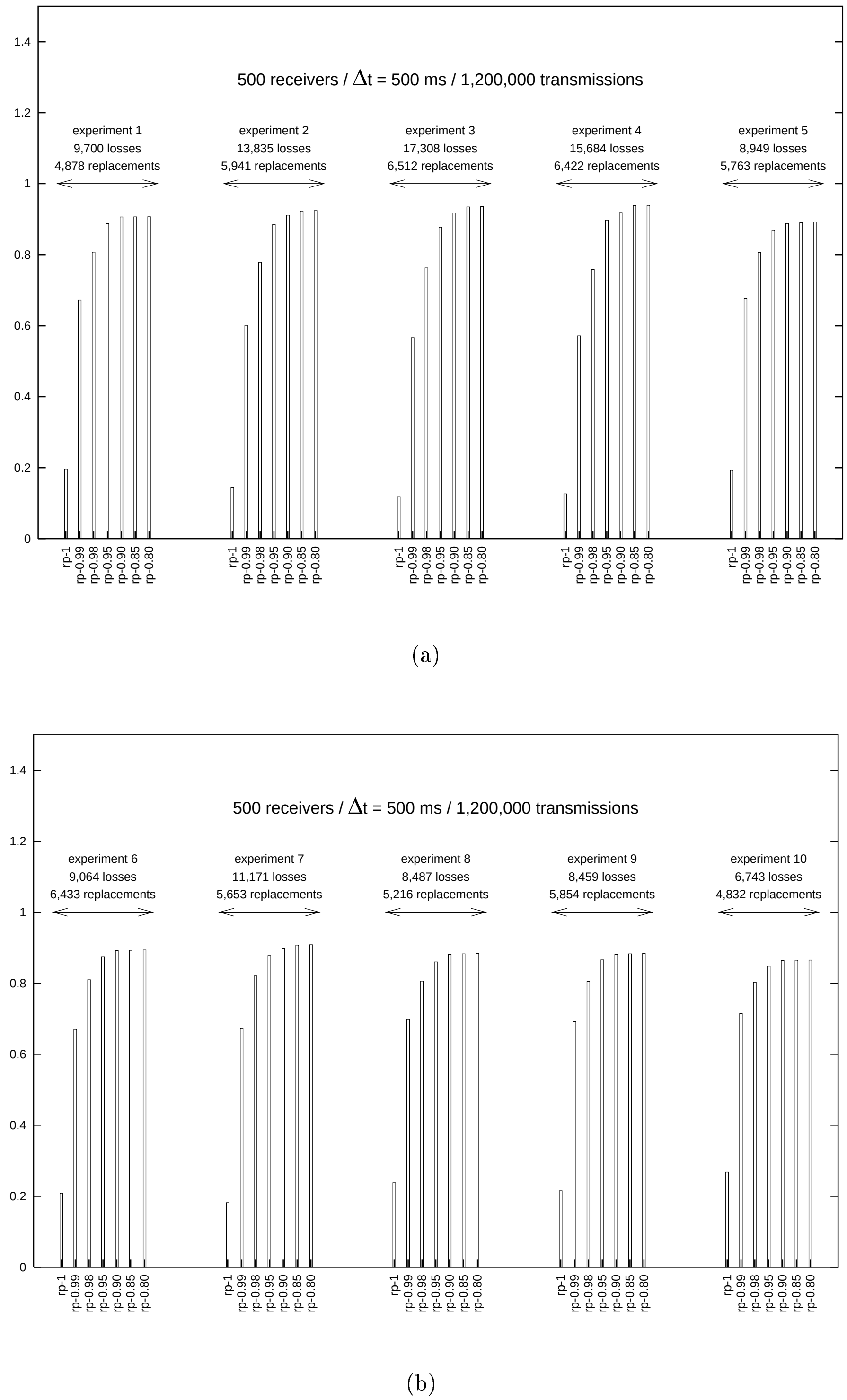
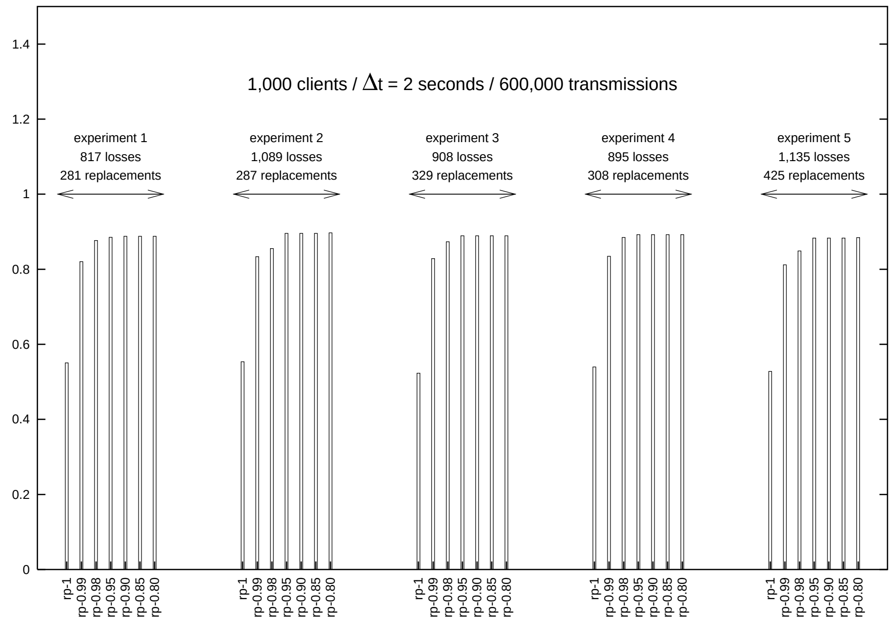
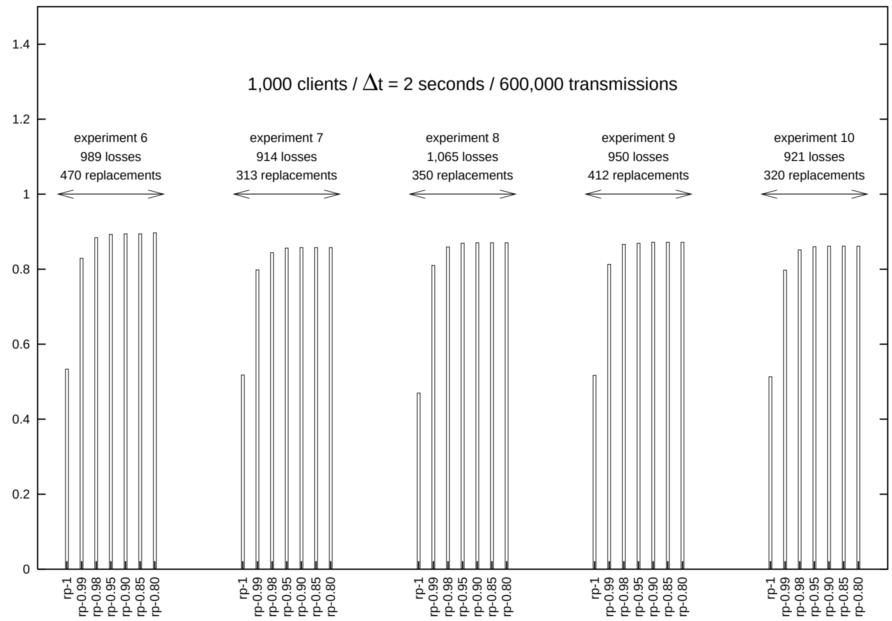


Figure 4.26: Experimental results for the periodic scheme. 500 emulated receivers and reporting interval of 500 ms.

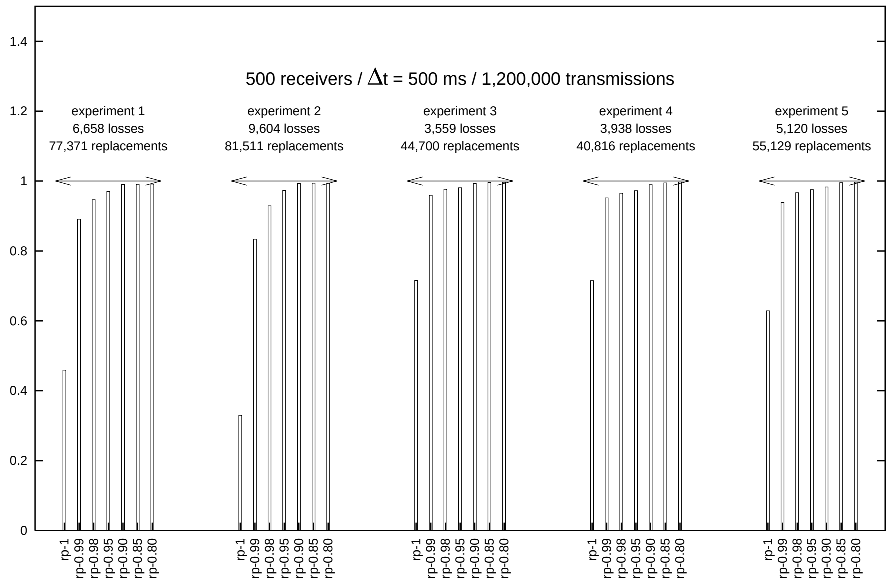


(a)

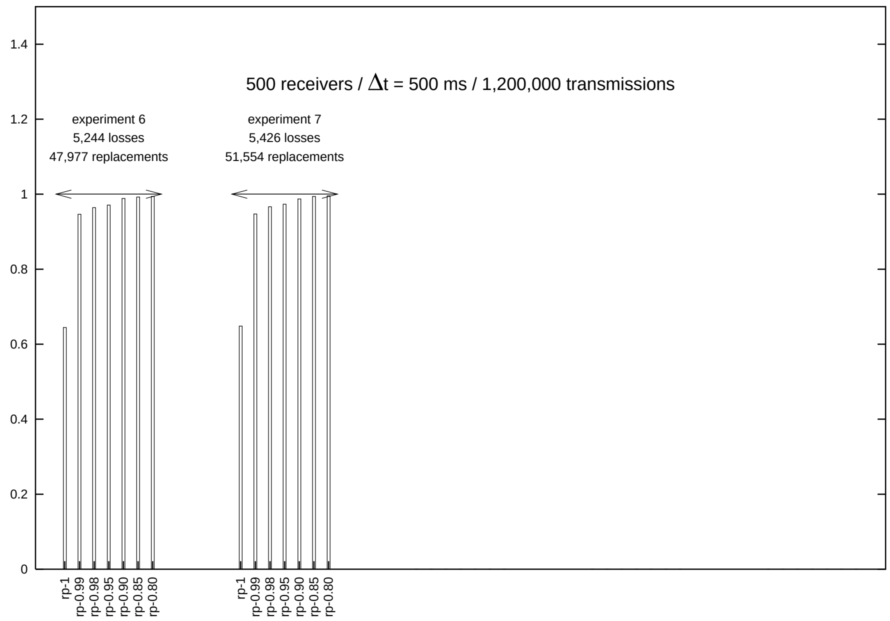


(b)

Figure 4.27: Experimental results for the periodic scheme. 1,000 emulated receivers and reporting interval of 2,000 ms.



(a)



(b)

Figure 4.28: Experimental results for the triggered scheme. 500 emulated receivers and reporting interval of 500 ms.

4.5 Résumé du Chapitre

D'expérimentations réelles sont importantes pour évaluer les performances des nouveaux protocoles et mécanismes qui sont proposés pour l'Internet. Nous avons dédié ainsi ce chapitre à une analyse pratique des nos solutions au travers d'expérimentations réelles. Nous avons collecté des statistiques par rapport à la gigue et à l'intervalle du retour d'information. Ces statistiques font référence à des études à l'intérieur de Paris aussi bien qu'entre continents (l'Amérique du sud et l'Europe). Dans nos études avec le schéma à déclanchement, nous avons améliorer le calcule de la taille de la fenêtre d'agrégation. Nous avons adopté $\Delta\tau$ égal à $\hat{\xi}_{k,optimal}\Delta t + 2\max_{i=1,\ldots,k}[J_i^{max}]$ borné par $1.1\hat{\xi}_{k,optimal}\Delta t$. Les expérimentations nous ont permis de démontrer une efficacité importante du système d'agrégation lorsque les techniques proposées dans cette thèse sont utilisées.

Chapter 5

Conclusions

Many applications benefit from summarized information of feedback data. This information is useful as input into different algorithms at the source such as the organization of a multicast data stream into useful layers for layered coding schemes and the parametrization of a multicast FEC algorithm. A proper summarization of information is achieved when an aggregator gets one report from its downstream neighbors before generating upstream feedback. This is difficult to achieve when the source needs to get continuously feedback in short time periods. Feedback transmission, which is the manner in which reports are generated, and temporal merging, which is the manner in which the aggregator decides when information is to be forwarded, affect the efficiency of the aggregation system. This work was concerned with the efficiency of an aggregation node. We have demonstrated how to tune feedback transmission, on the timer settings used by receivers, and temporal merging, on the timer settings used by an aggregation node.

We have started our treatment with a formal study of the aggregation system. Based on our system model, we have introduced evaluation metrics and formally described the system behavior. Our model was constructed with no assumptions about feedback transmission characteristics. The effort to construct such a model was compensated because it has made easier the description of the system behavior under different conditions.

We have studied feedback transmission and presented a solution to coordinate packet arrivals at an aggregator. The solution does not require neither clock synchronization nor tight feedback control. This allows the deployment of feedback control schemes willing to improve the aggregator efficiency and to reduce network overhead as well. In our approach, feedback transmission occurs at regular intervals. Each receiver regularly transmits consecutive feedback packets space apart by Δt. Jitter experienced by each receiver's feedback stream affects efficiency. However, the side effects of jitter are reduced with a suitable value for the reporting interval.

We have studied two schemes for regulating the behavior of the merging timer: periodic and triggered. The periodic scheme is simpler to implement but its simplicity is not priceless. We have demonstrated that the triggered scheme outperforms the periodic scheme. Under the periodic scheme, the merging time window size ($\Delta\tau$) should be equal to the feedback sending interval (Δt). The occurrence of varying delays does not change the relation $\Delta t = \Delta\tau$. It does introduce, however, a lower bound on the feedback sending interval in order to achieve a desirable performance. In practice, notable system performance is achieved when the reporting interval is very large with respect to jitter. Our results show that a reporting interval on the order of several seconds or even tens of seconds is necessary to achieve reasonable efficiency. These feedback latencies might be inadequate for several applications and so the applicability of the periodic scheme is reduced.

For the triggered scheme, the occurrence of varying delays changes the relation between Δt and $\Delta\tau$. Also, it introduces a lower bound on the feedback interval in order to achieve a desirable performance. The original merging time window size obtained disregarding random delays needs to be increased by a component that is dependent upon jitter experimented by each receiver's feedback flow. The approach we adopted for the timer adaptation was pertinent to our intent of performance evaluation under the assumption that the statistical bound of jitter estimates for each incoming feedback flow at the aggregator exhibit low variance. We have reasonably increased the merging time window size by $2\max_{i=1,\dots,k} J_i^{max}$. Our findings show that the triggered scheme performs much better than the periodic scheme. A feedback sending interval of only 700 ms has provided very good performance in all scenarios we have studied. The triggered scheme achieves better performance in terms of aggregation due to *packet condition* at the price of an increased number of packet replacements.

We have dedicated a chapter to real-world network experiments. Network experiments are very important to assess how new protocols and mechanisms perform in today's networks. We have collected jitter estimates from inner-Paris and intercontinental experiments as well. The results have provided us with insights into jitter and interarrival interval characteristics of feedback at an aggregator. We have developed the prototype of an aggregation node and presented a study of the impact of real-world experiments on the aggregator efficiency. Timer settings for the triggered scheme were reviewed to bound the size of merging time window. We have set $\Delta\tau$ to $\hat{\xi}_{k,optimal}\Delta t + 2\max_{i=1,\dots,k}[J_i^{max}]$ but it was limited to $1.1\hat{\xi}_{k,optimal}\Delta t$. In practice, $\Delta\tau$ can be reduced to $\Delta t + 2\max_{i=1,\dots,k}[J_i^{max}]$. The experimental findings are fairly in accordance with our expectation. The demonstration of how to achieve an improved system efficiency was done.

In addition, our implementation has demonstrated that aggregators can be easily deployed at end-systems. An application-level aggregator that is co-located with a multicast source provides a rapid way for making aggregation available. We have performed experiments with up to 1,000 emulated receivers. Despite this fact, we believe that it is possible to tune the aggregator system to support 10,000 receivers.

Our view is that feedback mergers are of considerable importance for multicast. Aggregation largely solves the feedback implosion problem and gives to many applications the possibility to benefit from having access to summarized feedback. Our hope is that, in contributing to a more solid understanding of aggregation performance, this work contributes to a more solid development of aggregation-based multicast applications as well.

5.1 Conclusion en Français

Nombreuses sont les applications à profiter de la nature résumée des données de retour. Par exemple, les applications de distribution de vidéo en couches sont souvent associées à des algorithmes de calcul des débits des couches à la source. Un autre exemple est le choix des paramètres d'algorithmes de FEC pour le multipoint. Une façon d'obtenir un résumé cohérent de toutes les données de retour sans impliquer une surcharge de calcul à la source est au travers de nœuds d'agrégation. Néanmoins, il a été prouvé que cette approche devient difficilement réalisable de façon efficace par rapport aux règles d'agrégation temporelle si la fréquence d'arrivée de données de retour à un nœud d'agrégation est élevée. On fait la distinction entre "la transmission de données de retour", correspondant à la manière dont les rapports sont générés par les récepteurs, et "l'agrégation temporelle", qui est la façon choisie par un nœud d'agrégation pour remonter l'information. Cette thèse s'est concentrée sur l'efficacité atteinte par un nœud d'agrégation. Nous avons démontré comment obtenir une efficacité importante au travers de la régulation du retour d'information de récepteurs et de l'agrégation temporelle dans les nœuds d'agrégation.

Dans un premier temps, nous avons analysé formellement le mode de fonctionnement d'un système d'agrégation. Pour ce faire, un certain nombre de métriques ont été définies tout en gardant une totale indépendance entre le modèle proposé et les caractéristiques des informations de retour. Avec cette analyse, il a été possible de décrire d'une manière simple le comportement du système dans des différentes conditions.

Ensuite, nous avons proposé une méthode de coordination des arrivées des paquets de retour à un nœud d'agrégation. Cette solution possède la particularité de n'exiger ni la synchronisation des horloges ni un contrôle strict sur l'instant exact de l'envoie des information de retour. Dans notre approche, les transmissions des paquets de retour sont programmées à des intervalles réguliers. Cet intervalle est, pour chaque récepteur, égal à Δt. Naturellement, la gigue observée pour chaque flux d'information est néfaste pour l'efficacité globale du système. Grâce à une méthodologie rigoureuse de sélection de l'intervalle d'émission des rapports, les effets secondaires de la gigue peuvent être réduits de manière significative.

Nous avons étudié deux schémas de régulation de temporisateurs : périodique et à déclenchement. Le schéma périodique est plus simple à implémenter cependant moins efficace. Cette approche atteint sont efficacité maximale lorsque la taille de la fenêtre d'agrégation ($\Delta\tau$) est identique à l'intervalle de retour d'information (Δt). Malgré le fait que la gigue ne change la relation $\Delta t = \Delta\tau$, elle peut modifier la borne inférieure de l'intervalle d'envoie d'informations de retour. En pratique, une bonne efficacité est assurée si l'intervalle Δt est très large par rapport à la gigue. Nos résultats ont montré qu'on peut obtenir une efficacité importante du système pour des valeurs de Δt supérieures à quelques secondes voire dizaines de secondes. Ces valeurs peuvent être trop élevées pour certaines classes d'applications très sensibles au délai, ce qui réduit les applications pratiques du schéma périodique.

Pour le schéma à déclenchement, la présence de gigue change la relation entre Δt et $\Delta\tau$. De plus, il introduit une borne inférieure pour l'intervalle Δt. La largeur de la fenêtre d'agrégation doit maintenant prendre en compte la gigue observée par le flux de retour d'information de chaque récepteur. Afin de proposer une solution qui respect cette exigence, nous avons augmenté la fenêtre d'agrégation de $2\max_{i=1,\dots,k} J_i^{max}$. Le schéma à déclenchement donne de meilleurs résultats que le schéma périodique. A titre d'exemple, un intervalle d'à peine 700 ms atteint déjà des performances très satisfaisantes dans les scénarios étudiés.

Nous avons dédié ainsi une partie de cette thèse à une analyse pratique des nos solutions au travers d'expérimentations réelles. Les mesures que nous avons effectuées sont des statistiques d'expérimentations à l'intérieur de Paris et entre continents (l'Amérique du sud et l'Europe). Plus spécifiquement, ces statistiques font référence à la gigue et à l'intervalle Δt. Dans cette étude, nous avons ajusté la borne du temporisateur théoriquement calculé pour le schéma à déclenchement. Nous avons adopté $\Delta\tau$ égal à $\hat{\xi}_{k,optimal}\Delta t + 2\max_{i=1,\dots,k}[J_i^{max}]$ borné par $1.1\hat{\xi}_{k,optimal}\Delta t$. En pratique, $\Delta\tau$ est simplement égal à $\Delta t + 2\max_{i=1,\dots,k}[J_i^{max}]$ Ces expérimentations nous ont permis de démontrer une efficacité importante du système d'agrégation lorsque les techniques proposées dans cette thèse sont utilisées.

De plus, dans la partie d'expérimentations, nous avons démontré qu'il est bien possible de déployer les nœuds d'agrégation dans les systèmes terminaux. L'avantage de cette approche est la collocation d'un nœud d'agrégation avec une source multipoint, ce qui permet la mise en place plus rapide des systèmes d'agrégation. Nous avons effectué de testes réelles avec 1000 récepteurs au maximum. Néanmoins, notre implémentation peut supporter facilement

10000 récepteurs.

En conclusion, nous sommes convaincus que l'agrégation d'informations de retour est une fonction de grande importance dans les communications de groupes à large échelle, où l'implosion de paquets à la source est un des problèmes fondamentaux. L'objective de cette thèse étant de contribuer à une meilleure compréhension de mécanismes d'agrégation, nous croyions avoir également contribué au développement d'applications multipoint basées sur l'agrégation de données.

Publications

[1] P. A. da Silva Gonçalves, J. F. de Rezende, and O. C. M. B. Duarte, *Réseaux de Capteurs, Book Chapter in Internet Ambiant.* Hermes Science Publications, 2004.

[2] P. A. da Silva Gonçalves, J. F. de Rezende, O. C. M. B. Duarte, and G. Pujolle, "Improving feedback merging for source-adaptive layered multicast schemes." To appear in the Kluwer Cluster Computer Journal (Special Issue with Selected Papers from Networking'2002).

[3] P. A. da Silva Gonçalves, M. Dias de Amorim, J. F. de Rezende, and O. C. M. B. Duarte, "Agregação e supressão de feedback de grupos multiponto heterogêneos," in *Proceedings of WQoSM'2002*, (Rio de Janeiro, Brazil), Nov. 2002.

[4] P. A. da Silva Gonçalves, M. Dias de Amorim, J. F. de Rezende, and O. C. M. B. Duarte, "Qos mechanisms for multilayered multicast applications," in *Tenth International Conference on Telecommunications Systems*, (Monterey, USA), Oct. 2002.

[5] P. A. da Silva Gonçalves, J. F. de Rezende, O. C. M. B. Duarte, and G. Pujolle, "Optimal feedback for quality source-adaptive schemes in multicast multi-layered video environments," in *Lecture Notes in Computer Science - Networking'2002*, no. 2345, pp. 563–574, May 2002.

References

[1] R. Beverly and K. Claffy, "Wide-area IP mulicast traffic characterization," *IEEE Network*, vol. 17, no. 1, pp. 8–15, Jan./Feb. 2003.

[2] I. Brown, J. Crowcroft, M. Handley, and B. Cain, "Internet multicast tomorrow," *The Internet Protocol Journal*, vol. 5, no. 4, Dec. 2002.

[3] J. Holbrook and B. Cain, "Source-specific multicast for ip," Internet Draft (Work in Progress), Oct. 2003.

[4] H. Holbrook and D. Cheriton, "IP multicast channels: Express support for large-scale single-source applications," in *Proceedings of ACM SIGCOMM'99*, Sept. 1999.

[5] K. Yano and S. McCanne, "The breadcrumb forwarding service: A synthesis of PGM and EXPRESS to improve and simplify global IP multicast," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 30, no. 2, Apr. 2000.

[6] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proceedings of ACM SIGCOMM'2001*, (San Diego, CA), Aug. 2001.

[7] J. Widmer and M. Handley, "Extending equation-based congestion control to multicast applications," in *Proceedings of ACM SIGCOMM'2001*, (San Diego, CA), Aug. 2001.

[8] L. Rizzo, "pgmcc: a TCP-friendly single-rate multicast congestion control scheme," in *Proceedings of ACM SIGCOMM'2000*, (Stockholm, Sweden), pp. 17–28, Aug. 2000.

[9] P. Radoslavov *et al.*, "A comparison of application-level and router-assisted hierarchical schemes for reliable multicast," in *IEEE INFOCOM'2001*, (Anchorage, AK), pp. 22–26, Apr. 2001.

[10] B. Whetten and G. Taskale, "An overview of reliable multicast transport protocol II," *IEEE Network*, vol. 14, no. 1, pp. 37–47, Feb. 2000.

[11] A. P. Markopoulou and F. A. Tobagi, "Hierarchical reliable multicast: Performance analysis and placement of proxies," *Networked Group Communications 2000*, pp. 27–36, Dec. 2002.

[12] S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya, "Reliable multicast transport protocol (RMTP)," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 407–421, Apr. 1997.

[13] S. Floyd *et al.*, "A reliable multicast framework for lightweight sessions and application level framing," in *Proceedings of ACM SIGCOMM'95*, (Cambridge, MA), Aug. 1995.

[14] H. W. Holbrook, S. H. Singhal, and D. R. Cheriton, "Log-based receiver-reliable multicast for distributed interactive simulation," in *Proceedings of ACM SIGCOMM'95*, (Cambridge, MA), Aug. 1995.

[15] R. Yavatkar, J. Griffioen, and M. Sudan, "A reliable dissemination protocol for interactive collaborative applications," in *Proceedings of ACM Multimedia'95*, (San Francisco, CA), pp. 333–343, 1995.

[16] S. Ramakrishnan and B. N. Jain, "A negative acknowledgement with periodic polling protocol for multicast over LANs," in *Procedings of IEEE INFOCOM'87*, pp. 502–511, Mar. 1997.

[17] B. J. Vickers, C. Albuquerque, and T. Suda, "Source-adaptive multilayered multicast algorithms for real-Time video distribution," *IEEE/ACM Transactions on Networking*, vol. 8, no. 6, pp. 720–733, Dec. 2000.

[18] B. J. Vickers, C. Albuquerque, and T. Suda, "Adaptive multicast of multi-layered video: Rate-based and credit-based approachs," in *Proc. IEEE INFOCOM'98*, (San Francisco, CA), Apr. 1998.

[19] X. Li, S. Paul, and M. H. Ammar, "Layered video multicast with retransmission (LVMR): Evaluation of hierarchical rate control," in *Proc. IEEE INFOCOM'98*, (San Francisco, CA), pp. 1062–1072, Apr. 1998.

[20] P. A. da Silva Gonçalves, J. F. de Rezende, and O. C. M. B. Duarte, "An active service for multicast video distribution," *Journal of the Brazilian Computer Society*, vol. 7, no. 1, pp. 43–51, July 2000.

[21] M. D. de Amorim, O. C. M. B. Duarte, and G. Pujolle, "Application-aware multicast," in *IEEE Globecom*, (San Antonio, Texas), Nov. 2001.

[22] M. D. Amorim, O. C. M. B. Duarte, and G. Pujolle, "Single-loop packet merging for receiver oriented multicast multi-layered video," in *Proc. International Conference in Computer Communications*, (Tokyo, Japan), Sept. 1999.

[23] M. D. de Amorim, O. C. M. B. Duarte, and G. Pujolle, "Multi-criteria arguments for improving the fairness of layered multicast applications," in *Lecture Notes in Computer Science*, no. 1815, pp. 1–10, May 2000.

[24] J. Chesterfield and E. M. Schooler, "An extensible rtcp control framework for large multimedia distrinutions," in *Proceedings of IEEE Internation Symposium on Netwrok Computing and Applications*, (Cambridge, MA), Apr. 2003.

[25] J. Chesterfield, E. M. Schooler, and J. Ott, "Rtcp extensions for single-source multicast sessions with unicast feedback," Internet Draft (Work in Progress), Apr. 2002.

[26] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications." Request for Comments 1889, Internet Engineering Task Force, Jan. 1996.

[27] R. Yavatkar and L. Manoj, "Optimistic strategies for large-scale dissemination of multimedia information," in *ACM Multimedia'93*, (Anaheim, CA), pp. 1–8, 1993.

[28] J. Bolot, T. Turletti, and I. Wakeman, "Scalable feedback control for multicast video distribution in the internet," in *Proceedings of ACM SIGCOMM'94*, (London, U.K.), pp. 58–67, Sept. 1994.

[29] J. Nonnenmacher and E. W. Biersack, "Scalable feedback for large groups," *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 375–386, June 1999.

[30] T. Fuhrmann and J. Widmer, "On the scaling of feedback algorithms for very large multicast groups," *Computer Communications*, vol. 24, no. 5-6, pp. 539–547, Mar. 2001.

[31] S. Floyd *et al.*, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 784–803, Dec. 1997.

[32] M. Handley, "Session directories and scalable internet address allocation," in *SIGCOMM'98*, (Vancouver, B.C., Canada), pp. 105–116, Sept. 1998.

[33] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications.* Internet RFC 3550, July 2003.

[34] H. Schulzrinne and S. Casner, *RTP: Profile for Audio and Video Conferences with Minimal Control.* Internet RFC 3551, July 2003.

[35] T. Friedman and D. Towsley, "Multicast session membership size estimation," in *Proceedings of IEEE INFOCOM'99*, (New York, USA), Mar. 1999.

[36] S. Floyd and V. Jacobson, "The synchronization of periodic routing messages," *IEEE/ACM Transactions on Networking*, vol. 2, no. 2, pp. 122–133, Apr. 1994.

[37] S. Floyd, *Congestion Control Principles.* Internet RFC 2914, Sept. 2000.

[38] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network architecture," *Computer Communication Review*, vol. 26, no. 2, Apr. 1996.

[39] D. L. Tennenhouse *et al.*, "A survey of active network research," *IEEE Communications Magazine*, pp. 80–86, Jan. 1997.

[40] T. Speakman *et al.*, "PGM reliable transport protocol specification," Internet RFC 3208, Dec. 2001.

[41] S. K. Kasera *et al.*, "Scalable fair reliable multicast using active services," *IEEE Network*, vol. 14, no. 1, pp. 48–57, Feb. 2000.

[42] L. wei H. Lehman, S. J. Garland, and D. L. Tennenhouse, "Active reliable multicast," in *IEEE INFOCOM'98*, (San Francisco, CA), pp. 581–589, Apr. 1998.

[43] S. Paul, K. K. Sabnani, and D. M. Kristol, "Multicast transport protocols for high speed networks," in *Proceedings of ICNP*, (Boston, Massachusetts), Oct. 1994.

[44] P. Gevros, J. Crowcroft, P. Kirstein, and S. Bhatti, "Congestion control mechanisms and the best effort service model," *IEEE Network*, vol. 15, no. 3, pp. 16–26, May/June 2001.

[45] L. L. Peterson and B. S. Davie, *Computer Networks: A Systems Approach.* San Francisco, California: Morgan Kaufmann Publishers, 1996.

[46] D. Tan and A. Zakhor, "Real-time internet video using error relilient scalable compression and TCP-friendly transport protocol," *IEEE Transactions on Multimedia*, May 1999.

[47] R. Rejaie, M. Handley, and D. Estrin, "Layered quality adaptation for internet video streaming," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, Dec. 2000.

[48] R. Rejaie, D. Estrin, and M. Handley, "Quality adaptation for congestion controlled video playback over the internet," in *Proc. ACM SIGCOMM'99*, (Cambridge, MA), Sept. 1999.

[49] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A model based TCP-friendly rate control protocol," in *Proceedings of NOSSDAV'99*, 1999.

[50] D. Sasalem and H. Schulzrinne, "The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme," in *Proceedings of NOSSDAV'98*, 1999.

[51] S. Jacobs and A. Eleftheriads, "Providing video services over networks without service guarantees," in *World Wide Web Consortium Workshop on Real-Time Multimedia and the Web*, 1996.

[52] S. Bhattacharyya, J. Kurose, and D. Towsley, "The lost path multiplicity problem in multicast congestion control," in *IEEE INFOCOM'99*, (New York, NY), Mar. 1999.

[53] J. Golestani and K. Sabnani, "Fundamental observations on multicast congestion control in the internet," in *Proceedings of IEEE INFOCOM'99*, (New York, USA), Mar. 1999.

[54] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proc. SIGCOMM'96*, (Stanford, CA), pp. 117–130, Aug. 1996.

[55] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like congestion control for layered multicast data transfer," in *IEEE INFOCOM'98*, (San Francisco, CA), Apr. 1998.

[56] B. J. Vickers, C. Albuquerque, and T. Suda, "Source adaptive multi-layered multicast algorithms for real-time video distribution," *IEEE/ACM Transactions on Networking*, vol. 8, no. 6, pp. 720–733, Dec. 2000.

[57] M. D. de Amorim, O. C. M. B. Duarte, and G. Pujolle, "Application-Aware multicast," in *Proc. GLOBECOM'2001*, (San Antonio, Texas), Nov. 2001.

[58] S. Keshav, *An Engineering Approach to Computer Networking*. Addison-Wesley Professional Computting Series, 1997.

[59] D. L. Mills, A. Thyagarajan, and B. C. Huffman, "Internet timekeeping around the globe," in *Proceedings of Precision Time and Time Interval Applications and Planning Meeting*, pp. 365–371, Dec. 1999.

[60] D. L. Mills, *Network Time Protocol (Version 3) Specification, Implementation and Analysis*. Internet RFC 1305, Mar. 1992.

[61] D. Deeths and G. Brunnet, "Using NTP to controland synchronize system clocks - part I: Introduction to NTP," Sun Microsystems – URL: http://www.sun.com/solutions/blueprints/0701/NTP.pdf, July 2001.

[62] J. Gemmell, T. Montgomery, T. Speakman, and N. Bhaskar, "The PGM relible multicast protocol," *IEEE Network*, vol. 17, no. 1, pp. 16–22, Jan./Feb. 2003.

[63] S. Bajaj *et al.*, "Improving simulation for network research," tech. rep., University of Southern California, Department of Computer Science, Mar. 1999.

[64] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "Iperf," NLANR – URL: http://dast.nlanr.net/Projects/Iperf/, Mar. 2003.

[65] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*. Internet RFC 1889, Jan. 1996.

[66] J. A. Cadzow, *Foudations of Digital Signal Processing and Data Analysis*. New York, New York: Macmillan, 1987.

[67] V. Paxson, "End-to-end routing behavior in the internet," *IEEE/ACM Transactions on Networking*, vol. 5, no. 3, no. 3, pp. 601–615, 1996.

[68] R. Jain, *The Art of Computer Systems Performance Analysis*. Wiley Professional Computting, 1991.

List of acronyms

ACK	*Positive Acknowledgement*
CET	*Central Europe Time*
CEST	*Central Europe Summer Time*
CPU	*Central Process Unit*
DARPA	*Defense Advanced Research Projects Agency*
EXPRESS	*EXPlicitely REquested Single Source*
FEC	*Forward Error Control*
FreeFR	*Internet Service Provider in French*
GTA	Grupo de Teleinformática e Automação - Network and Automation Group (Brazil)
ICMP	*Internet Control Message Protocol*
IETF	*Internet Enginneering Task Force*
IGMP	*Internet Group Management Protocol*
IP	*Internet Protocol*
IPV4	*Internet Protocol version 4*
ISP	*Internet Service Provider*
LAN	*Local Area Network*
LIP6	Laboratoire d'Informatique de PARIS VI (French)
NAK	*Negative Acknowledgement*
NOOS	*Cable TV and Internet operator in French*
NS	*Network Simulator*
QoS	*Quality of Service*
RFC	*Request For Comments*
RTCP	*RTP Control Protocol*
RTP	*Real-time Transport Protocol*
RTT	*Round-Trip Time*
SSM	*Source-Specific Multicast*
TCP	*Transmission Control Protocol*
TTL	*Time-To-Live*

UDP	*User Datagram Protocol*
WAN	*Wide Area Network*

List of figures

List of tables

Communications de Groupes :
Analyse et Performances des Mécanismes
d'agrégation de l'information de Contrôle

Résumé Qu'elles soient déjà existantes ou pressenties, les applications multiparties fondées sur un service de multidiffusion requièrent pour certaines le retour d'information des participants qu'elles mettent en rapport. Selon ces applications, les informations générées peuvent être utilisées par une source émettrice pour réaliser du contrôle de flux ou d'erreur, pour adapter (choisir) la méthode de codage, voire même organiser les données en couches cumulatives. Le nombre de participants mis en rapport pouvant atteindre l'ordre de plusieurs milliers, le volume des informations de contrôle retournées entraîne l'implosion de la source pour des sessions de grande taille. Parmi les solutions jusqu'à présent proposées pour répondre aux contraintes liées au facteur d'échelle, l'agrégation fournit un moyen efficient au regard de ces contraintes. L'agrégation consiste à résumer les informations de contrôle retournées par chacun des participants dans des messages uniques. Les informations sont résumées (dans ces messages) de façon à constituer les données d'entrée de nombreux algorithmes utilisés par les sources émettrices. Cela en prenant en considération une éventuelle contrainte temporelle du retour d'information.

La problématique des travaux présentés dans cette thèse est d'améliorer l'efficacité des entités responsables de l'agrégation. Nous traitons plus particulièrement de deux mécanismes : le premier est celui utilisé par un récepteur pour temporiser le retour de ses informations de contrôle. Le second mécanisme est utilisé pour réguler temporellement l'agrégation au niveau des entités qui en sont responsables. La méthode que nous proposons est générique et s'appuie uniquement sur la régulation des temporisateurs (systèmes de temporisation). Notre méthode a la particularité d'être indépendante du type de contenu agrégé et de fonctionner sur une éventuelle sélection de paquets à acheminer. Nous modélisons le système d'agrégation et nous donnons une description formelle de son comportement. Les performances de notre système sont évaluées au travers de simulations. Nous démontrons une efficacité améliorée lorsque un processus d'émission périodique d'informations de contrôle est associé à notre méthode d'agrégation temporelle. Cette thèse inclut également le développement d'un prototype de nœud d'agrégation, pour valider, à partir d'expérimentations, l'aptitude de notre méthode à fonctionner dans des réseaux réels.

Mots clés Communications de groupes, agrégation, efficacité, performance

---

Group Communications:
Analysis and Performance of
Feedback Aggregation Mechanisms

Abstract is on page 9
Keywords Multicast, feedback aggregation, aggregation efficiency, performance