UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T J.121

(02/2002)

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

SÉRIE J: RÉSEAUX CÂBLÉS ET TRANSMISSION DES SIGNAUX RADIOPHONIQUES, TÉLÉVISUELS ET AUTRES SIGNAUX MULTIMÉDIAS

Services interactifs pour la distribution de télévision numérique

Protocole de contrôle de qualité de la webodiffusion

Recommandation UIT-T J.121

RECOMMANDATIONS UIT-T DE LA SÉRIE J

RÉSEAUX CÂBLÉS ET TRANSMISSION DES SIGNAUX RADIOPHONIQUES, TÉLÉVISUELS ET AUTRES SIGNAUX MULTIMÉDIAS

Recommandations générales	J.1–J.9
Spécifications générales des transmissions radiophoniques analogiques	J.10–J.19
Caractéristiques de fonctionnement des circuits radiophoniques analogiques	J.20–J.29
Equipements et lignes utilisés pour les circuits radiophoniques analogiques	J.30–J.39
Codeurs numériques pour les signaux radiophoniques analogiques	J.40–J.49
Transmission numérique de signaux radiophoniques	J.50–J.59
Circuits de transmission télévisuelle analogique	J.60–J.69
Transmission télévisuelle analogique sur lignes métalliques et interconnexion avec les faisceaux hertziens	J.70–J.79
Transmission numérique des signaux de télévision	J.80–J.89
Services numériques auxiliaires propres aux transmissions télévisuelles	J.90–J.99
Prescriptions et méthodes opérationnelles de transmission télévisuelle	J.100–J.109
Services interactifs pour la distribution de télévision numérique	**J.110–J.129**
Transport des signaux MPEG-2 sur les réseaux par paquets	J.130–J.139
Mesure de la qualité de service	J.140–J.149
Distribution de la télévision numérique sur les réseaux locaux d'abonnés	J.150–J.159
IPCablecom	J.160–J.179
Divers	J.180–J.199
Application à la télévision numérique interactive	J.200–J.209

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

Recommandation UIT-T J.121

Protocole de contrôle de qualité de la webodiffusion

Résumé

La présente Recommandation définit les protocoles qui opèrent entre un serveur et un client et assurent la distribution de programmes audio et télévisuels conformément à la Rec. UIT-T J.120, c'est-à-dire par webodiffusion, sur un réseau IP général, réseau qui n'offre pas de garantie de qualité de service et dans lequel des erreurs de données ou des pertes de paquets peuvent donc se produire. L'utilisation de ces protocoles se traduit par une amélioration de la qualité.

Un mécanisme est défini par lequel un rapport sur la qualité est envoyé depuis un client vers le serveur, rapport qui est utilisé par le serveur pour optimiser la distribution des données.

La présente Recommandation définit des types de paramètres qui doivent être contenus dans le rapport du client destiné au serveur, ainsi qu'un protocole de transmission qui est utilisé pour envoyer le rapport du client au serveur. La présente Recommandation définit aussi la transmission de paramètres associés utiles au serveur pour analyser le rapport du client.

Source

La Recommandation J.121 de l'UIT-T, élaborée par la Commission d'études 9 (2001-2004) de l'UIT-T, a été approuvée le 13 février 2002 selon la procédure définie dans la Résolution 1 de l'AMNT.

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, détermine les thèmes d'étude à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution 1 de l'AMNT.

Dans certains secteurs des technologies de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

NOTE

Dans la présente Recommandation, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un Membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT n'avait pas été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

TABLE DES MATIÈRES

Recommandation UIT-T J.121

Protocole de contrôle de qualité de la webodiffusion

1 Domaine d'application

La réaction du serveur à la réception du rapport du client ne fait pas partie du domaine d'application de la présente Recommandation.

2 Références

La présente Recommandation se réfère à certaines dispositions des Recommandations UIT-T et textes suivants qui, de ce fait, en sont partie intégrante. Les versions indiquées étaient en vigueur au moment de la publication de la présente Recommandation. Toute Recommandation ou tout texte étant sujet à révision, les utilisateurs de la présente Recommandation sont invités à se reporter, si possible, aux versions les plus récentes des références normatives suivantes. La liste des Recommandations de l'UIT-T en vigueur est régulièrement publiée.

2.1 Références normatives

– Recommandation UIT-T H.225.0 (2000), *Protocoles de signalisation d'appel et mise en paquets des trains multimédias dans les systèmes de communication multimédia en mode paquet*.

– Recommandation UIT-T J.120 (2000), *Distribution de programmes radiophoniques et télévisuels sur le réseau IP*.

2.2 Références informatives

– IETF RFC 1889 (1996), *RTP: A Transport Protocol for Real-Time Applications*.

3 Termes et définitions

La présente Recommandation définit les termes suivants:

3.1 protocole de transport en temps réel (RTP, *real-time transport protocol*): protocole pour applications en temps réel défini dans la Rec. UIT-T H.225.0.

3.2 protocole de commande RTP (RTCP, *RTP control protocol*): protocole de commande pour paquets RTP défini dans la Rec. UIT-T H.225.0.

3.3 webodiffusion: définie dans la Rec. UIT-T J.120. Distribution de programmes radiophoniques et télévisuels sur le réseau IP.

4 Abréviations et conventions

4.1 Abréviations

La présente Recommandation utilise les abréviations suivantes:

IP protocole Internet (*Internet protocol*)

RTCP protocole de commande RTP (*RTP control protocol*)

RTP protocole de transport en temps réel (*real-time transport protocol*)

TCP protocole de commande de transmission (*transmission control protocol*)

UDP protocole datagramme d'utilisateur (*user datagram protocol*)

4.2 Conventions

Pour l'implémentation de la présente Recommandation, les termes "DOIT" et "DEVRA" ainsi que "REQUIS" doivent être interprétés comme indiquant un aspect obligatoire de la présente spécification.

Les mots clés utilisés pour définir l'importance d'une prescription particulière dans la présente Recommandation sont résumés ci-dessous.

"DOIT"	Ce mot ainsi que l'adjectif "REQUIS" indiquent que l'article est une obligation absolue de la présente spécification.
"NE DOIT PAS"	Cette expression indique que l'article est une interdiction absolue de la présente spécification.
"DEVRAIT"	Cette expression ainsi que l'adjectif "RECOMMANDÉ" indiquent qu'il peut, dans des circonstances particulières, exister des raisons valables pour ignorer cet article, mais qu'il convient, avant de faire ce choix, de prendre en considération toutes les incidences et d'étudier soigneusement le cas.
"NE DEVRAIT PAS"	Cette expression indique qu'il peut, dans des circonstances particulières, exister des raisons valables pour que le comportement indiqué soit acceptable voire même utile, mais qu'il convient, avant de faire ce choix, de prendre en considération toutes les incidences et d'étudier soigneusement le cas.
"PEUT"	Ce mot ainsi que l'adjectif "FACULTATIF" indiquent que cet article est effectivement facultatif. Un fournisseur peut choisir d'inclure l'article par exemple parce qu'il est exigé sur un marché particulier ou parce qu'il améliore le produit, alors qu'un autre fournisseur peut choisir d'omettre ce même article.

5 Définition du système

Le système et les flux des signaux associés entre le serveur et les clients sont représentés à la Figure 1. Dans ce schéma, le rapport d'émetteur (SR, *sender report*) et le rapport du récepteur (RR) du protocole RTCP sont utilisés pour l'échange d'informations entre le serveur et le client. Le protocole RTCP est fondé sur la transmission périodique de paquets de commande vers tous les participants d'une session, en utilisant le même mécanisme de distribution que celui de paquets de données médias. Le protocole sous-jacent doit assurer le multiplexage des données et des paquets de commande, en utilisant par exemple, des numéros de port distincts avec le protocole UDP.

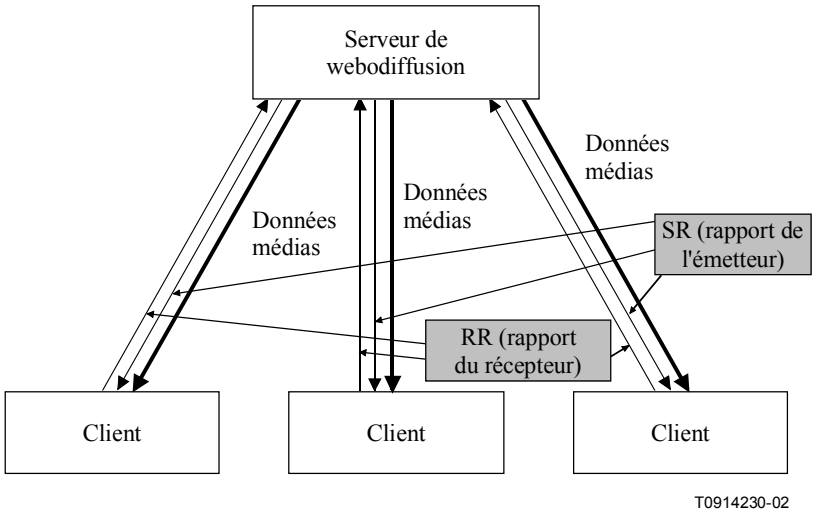


Figure 1/J.121 – Schéma bloc du système de contrôle de qualité pour la webodiffusion

5.1 Procédure utilisée au niveau du serveur

5.1.1 Transmission des informations d'aide aux mesures

Tout d'abord, le serveur envoie des informations d'aide aux mesures qui incluent un temps de référence et des informations concernant les paquets transmis vers le client. Ces informations sont utilisées par le client pour élaborer le rapport du client et par le serveur pour analyser le rapport du client. Bien que l'intervalle de transmission pour cette information ne soit pas défini dans la présente Recommandation, le paragraphe 7 spécifie l'intervalle de transmission recommandé. Cette information est acheminée dans un paquet SR (*sender report*: rapport d'émetteur) défini au § 6.1. Le paquet SR est envoyé en utilisant un numéro de port différent de celui des données médias (vidéo/audio). Le numéro de port est donné par le déclenchement d'une session du protocole J.120, ce numéro est égal au numéro de port RTP de données média plus 1.

5.1.2 Réception du rapport du client

Le serveur doit toujours se trouver dans un état où il est en mesure de recevoir le paquet de rapport provenant du client.

5.1.3 Distribution des données médias

Le serveur doit assurer la distribution optimale des données médias (audio/vidéo) compte tenu des résultats de l'analyse du paquet rapport provenant du client. La méthode effectivement utilisée pour la commande de la distribution dépend de l'implémentation et n'entre pas dans le domaine d'application de la présente Recommandation.

5.2 Procédure au niveau du client

5.2.1 Monitorage de la qualité de transmission

Le client surveille les conditions sous lesquelles les données médias sont reçues et mesure la qualité de la transmission. Les paramètres faisant l'objet des mesures sont décrits au § 6.2. Le client doit déterminer ces paramètres à partir des informations d'aide aux mesures précitées provenant du serveur.

5.2.2 Transmission vers le serveur du rapport du client

La qualité de transmission mesurée au niveau du client est communiquée au serveur dans un rapport du client. Ce rapport doit être acheminé sous forme d'un paquet RR (rapport du récepteur) défini au § 6.2. Le paquet RR est envoyé en utilisant le numéro de port donné par le déclenchement d'une session du protocole J.120.

6 Définition d'un paquet rapport

6.1 Rapport d'émetteur (SR)

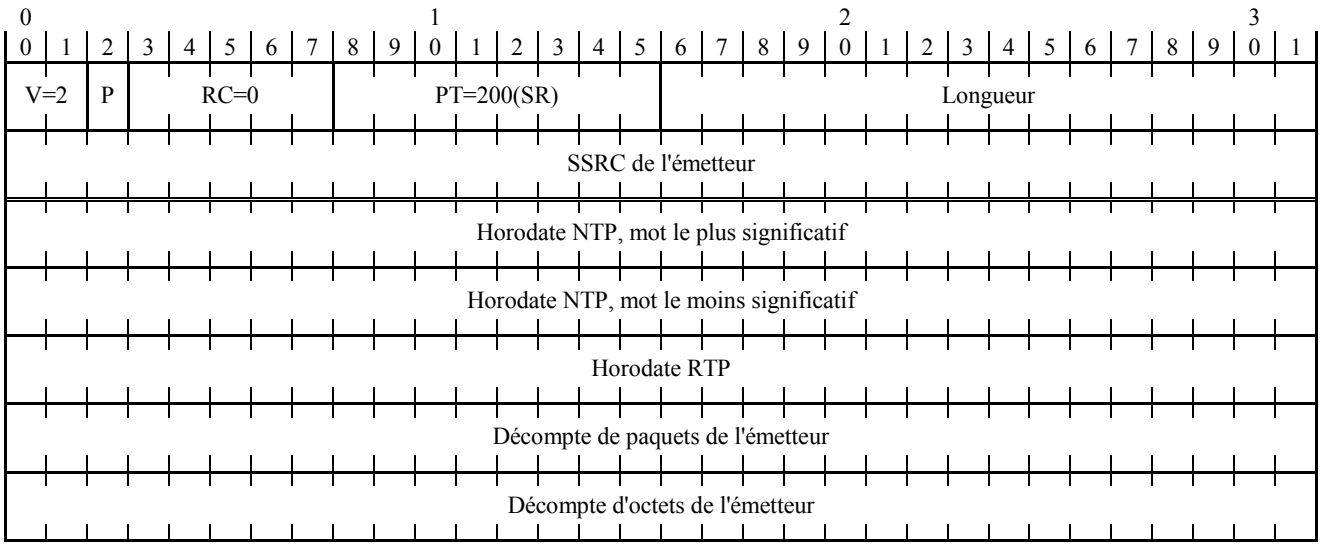


Figure 2/J.121 – Format du paquet rapport d'émetteur (SR)

Le paquet rapport d'émetteur se compose de deux sections: la première section, l'en-tête, occupe 8 octets, les champs ont les significations suivantes:

version (V): 2 bits. Ce champ identifie la version du protocole RTP, qui est la même dans les paquets RTCP que dans les paquets données RTP. La version définie dans la présente Recommandation est deux (2).

remplissage (P, *padding*): 1 bit. Si le bit de remplissage est positionné, le paquet RTCP contient à la fin quelques octets supplémentaires de remplissage qui ne font pas partie des informations de commande. Le dernier octet de remplissage indique le nombre d'octets de remplissage qu'il faut ignorer. L'opération de remplissage peut être imposée par certains algorithmes de cryptage qui nécessitent des tailles de bloc fixes. Dans un paquet RTCP composite, le remplissage ne doit être requis qu'au niveau du dernier paquet simple car le paquet composite est crypté comme un tout.

décompte de rapports (RC, *report count*) de réception: 5 bits. Ce champ donne le nombre de blocs de rapport de réception contenus dans le paquet considéré. La valeur zéro est valide.

type de paquet (PT, *packet type*): 8 bits. Ce champ contient la constante 200 qui indique qu'il s'agit d'un paquet RTCP SR.

longueur: 16 bits. Il s'agit de la longueur du paquet RTCP considéré, en nombre de mots de 32 bits moins un, y compris l'en-tête et le remplissage éventuel. (Le décalage de 1 fait que zéro est une longueur valide et permet d'éviter les éventuelles boucles infinies lors de l'analyse d'un paquet RTCP composite; par ailleurs, le fait de compter le nombre de mots de 32 bits permet d'éviter un contrôle de validité sur les multiples de 4.)

SSRC (*synchonization source*): 32 bits. Il s'agit de l'identificateur de source de synchronisation pour l'émetteur du paquet SR considéré.

La deuxième section, les informations de l'émetteur, occupe 20 octets et figure dans chaque paquet de rapport d'émetteur. Elle récapitule les transmissions de données faites par cet émetteur. La signification des champs est la suivante:

horodate NTP (*network time protocol*): 64 bits. Ce champ indique l'heure lue sur l'horloge murale où ce rapport a été envoyé de sorte qu'associé avec les horodates renvoyés dans les rapports de réception des autres récepteurs, il permet de mesurer le temps aller-retour vers ces récepteurs. Les récepteurs doivent savoir que la précision de la mesure de l'horodate peut être beaucoup moins bonne que la résolution de l'horodate NTP. L'incertitude sur la mesure de l'horodate n'est pas indiquée car elle peut ne pas être connue. Un émetteur qui peut garder la trace du temps écoulé mais qui ne connaît pas l'heure lue sur une horloge murale, peut se servir du temps écoulé jusqu'à ce qu'il rejoigne la session. Ce temps est supposé être inférieur à 68 ans, le bit supérieur sera ainsi égal à zéro. Il est permis d'utiliser l'horloge d'échantillonnage pour évaluer le temps écoulé lu sur une horloge murale. Un émetteur qui n'a aucune notion de l'heure indiquée par l'horloge murale ou qui n'a pas la notion du temps écoulé peut mettre l'horodate NTP à zéro.

horodate RTP: 32 bits. Cette horodate correspond à l'horodate NTP (ci-dessus), mais elle est exprimée avec les mêmes unités et a le même décalage aléatoire que les horodates RTP des paquets de données. Cette correspondance peut servir à la synchronisation intra et intermédia des sources dont les horodates NTP sont synchronisées; elle peut aussi servir aux récepteurs indépendants du média à évaluer la fréquence nominale de l'horloge RTP. A noter que dans la plupart des cas, cette horodate sera différente de l'horodate RTP d'un paquet de données adjacent quelconque. Elle est plutôt calculée à partir de l'horodate NTP correspondante en utilisant la relation qui existe entre le compteur d'horodate RTP et le temps réel et qui est mise à jour périodiquement en contrôlant l'heure lue sur l'horloge murale à un instant d'échantillonnage.

décompte de paquets de l'émetteur: 32 bits. Il s'agit du nombre total de paquets de données RTP envoyés par l'émetteur depuis le début de la transmission jusqu'à l'instant où le paquet SR considéré a été produit. Ce décompte est réinitialisé si l'émetteur change d'identificateur SSRC.

décompte d'octets de l'émetteur: 32 bits. Il s'agit du nombre total d'octets de charge utile (c'est-à-dire n'incluant pas d'en-tête ou ne servant pas au remplissage) transmis par l'émetteur dans les paquets de données RTP depuis le début de la transmission jusqu'à l'instant où ce paquet SR a été produit. Ce décompte est réinitialisé si l'émetteur modifie son identificateur SSRC. Ce champ peut servir à évaluer le débit moyen de données de charge utile.

6.2 Rapport du récepteur

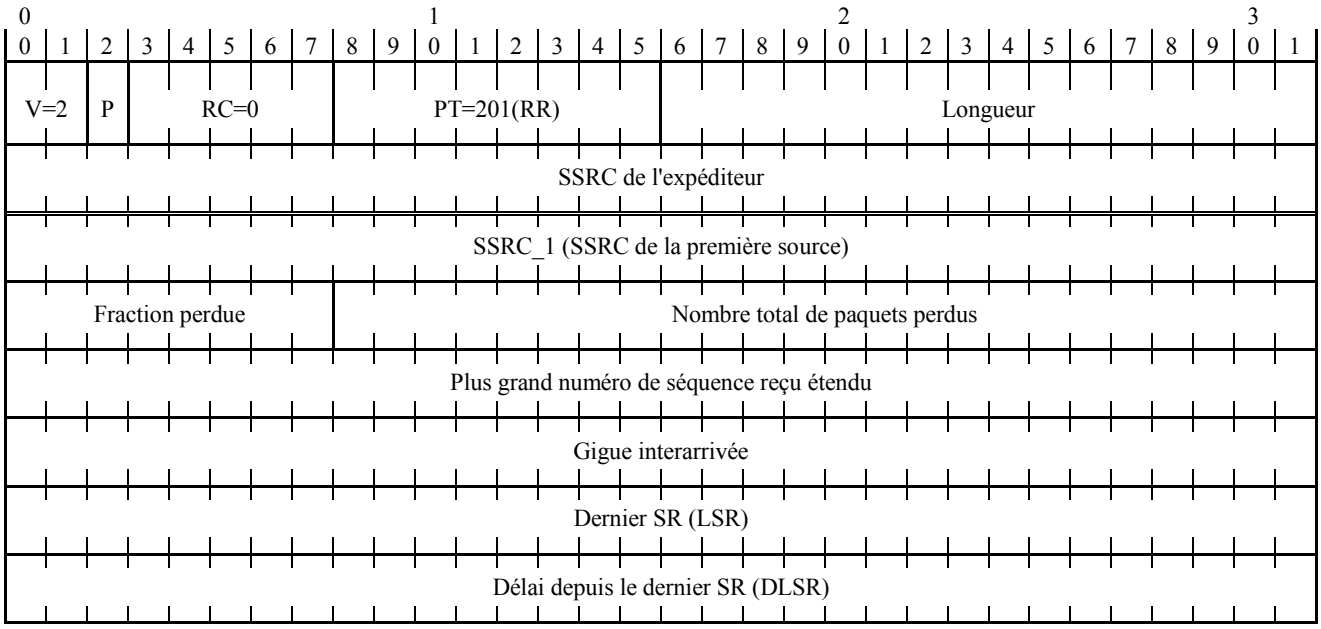


Figure 3/J.121 – Format du paquet rapport du récepteur (RR)

Le paquet rapport du récepteur est constitué de deux sections. La première section, l'en-tête, qui a une longueur de 8 octets, est la même que celle du paquet SR sauf que le champ type de paquet contient la constante 201. Les champs sont définis au § 6.1.

La deuxième section contient un nombre quelconque ou nul de blocs de rapport de réception qui dépend du nombre d'autres sources entendues par ce récepteur depuis le dernier rapport. Chaque bloc de rapport de réception comporte des statistiques sur la réception des paquets RTP provenant d'une même source de synchronisation. Les récepteurs ne gardent pas en mémoire les statistiques lorsqu'une source change d'identificateur SSRC suite à une collision. Ces statistiques sont les suivantes:

SSRC_n (identificateur de la source): 32 bits. Il s'agit de l'identificateur SSRC de la source sur laquelle portent les informations figurant dans ce bloc rapport de réception.

fraction perdue: 8 bits. Il s'agit de la fraction des paquets de données RTP provenant de la source SSRC_n qui ont été perdus depuis l'envoi du précédent paquet SR ou RR, exprimée par un nombre à virgule fixe, la virgule binaire étant placée au bord gauche du champ. (Cela revient à prendre la partie entière après avoir multiplié la fraction perdue par 256.) Cette fraction est définie comme étant le nombre de paquets perdus divisé par le nombre de paquets prévus (voir la définition dans le paragraphe qui suit). Si la perte est négative en raison de paquets en double, la fraction perdue est mise à zéro. On notera qu'un récepteur ne peut pas dire si des paquets ont été perdus après le dernier paquet reçu, et qu'aucun bloc du rapport de réception ne doit être émis pour une source donnée si tous les paquets envoyés par cette source pendant le dernier intervalle de rapport ont été perdus.

nombre total des paquets perdus: 24 bits. Il s'agit du nombre total de paquets de données RTP provenant de la source SSRC_n qui ont été perdus depuis le début de la réception. Par définition, il s'agit du nombre de paquets prévus moins le nombre de paquets effectivement reçus, où le nombre de paquets reçus comprend les paquets en retard et les paquets en double. Par conséquent, les paquets qui arrivent en retard ne sont pas comptés comme des paquets perdus, et la perte peut être négative s'il y a des paquets en double. Le nombre de paquets prévus est défini comme le dernier numéro de séquence reçu étendu (défini plus loin) moins le numéro de séquence initial reçu. Ce nombre peut se calculer par la méthode donnée à l'Appendice III.

plus grand numéro de séquence reçu étendu: 32 bits. Les 16 bits inférieurs contiennent le plus grand numéro de séquence reçu dans un paquet de données RTP provenant de la source SSRC_n, et les 16 bits les plus significatifs étendent ce numéro de séquence par le nombre correspondant de cycles de numéros de séquence, qui peut être mis à jour au moyen de l'algorithme donné à l'Appendice II. On notera que des récepteurs distincts dans une même session produiront des extensions du numéro de séquence différentes si les heures auxquelles ils ont été activés sont très différentes.

gigue interarrivée: 32 bits. Il s'agit d'une évaluation de la variance statistique du temps interarrivée des paquets de données RTP, mesurée en unités d'horodate et exprimée par un entier non signé. La gigue interarrivée J est définie comme étant l'écart moyen (valeur absolue lissée) de la différence D d'espacement de deux paquets au niveau du récepteur par rapport à l'émetteur. Comme le montre l'équation ci-dessous, la différence D est équivalente à la différence de "temps de transit relatif" pour les deux paquets; le temps de transit relatif est la différence entre l'horodate RTP d'un paquet et l'heure indiquée par l'horloge du récepteur au moment de l'arrivée du paquet, mesurée dans la même unité.

Si *Si* est l'horodate RTP du paquet *i*, et *Ri* l'heure d'arrivée en unités d'horodate RTP du paquet *i*, alors pour les deux paquets *i* et *j*, D peut s'exprimer comme suit:

$$D(i+j) = (Rj - Ri) - (Sj - Si) = (Rj - Sj) - (Ri - Si)$$

Chaque fois qu'un paquet de données *i* est reçu en provenance de la source SSRC_n, la gigue interarrivée est recalculée en fonction de la différence *D* s'appliquant à ce paquet et au paquet $i - 1$ le précédant dans l'ordre d'arrivée (pas nécessairement le paquet le précédant dans la séquence), selon l'équation:

$$J = J + \frac{|D(i-1,i)| - J}{16}$$

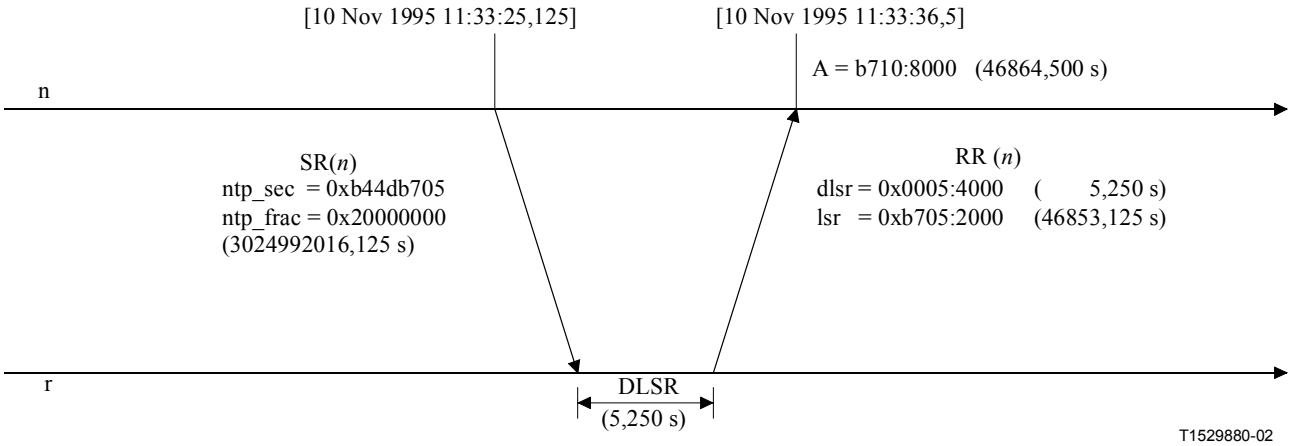
Chaque fois qu'un rapport de réception est transmis, la valeur courante de *J* est échantillonnée.

Le calcul de la gigue est prescrit ici pour permettre aux contrôleurs indépendants du profil de faire des interprétations valides des rapports venant de différentes implémentations. Cet algorithme est l'estimateur optimal du premier ordre et le paramètre de gain 1/16 donne un bon facteur de réduction du bruit tout en conservant une vitesse de convergence raisonnable. L'Appendice V contient un exemple d'implémentation.

horodate du dernier paquet SR (LSR, *last SR time-stamp*): 32 bits. Il s'agit des 32 bits du milieu parmi les 64 de l'horodate NTP (comme expliqué au A.4/H.225.0: ordre des octets, alignement et format temporel) qui figurent dans le dernier paquet RTCP SR (rapport d'émetteur) reçu en provenance de la source SSRC_n. Si aucun paquet SR n'a encore été reçu, le champ est mis à zéro.

délai depuis le dernier paquet SR (DLSR, *delay since last SR*): 32 bits. Il s'agit du délai, exprimé dans l'unité 1/65536 secondes, entre la réception du dernier paquet SR envoyé par la source SSRC_n et l'envoi de ce bloc de rapport de réception. Si aucun paquet SR n'a encore été reçu de la source SSRC_n, le champ DLSR est mis à zéro.

Désignons par SSRC_r le récepteur qui envoie ce rapport de réception. La source SSRC_n peut calculer le temps de transmission total vers le récepteur SSRC_r en enregistrant l'heure A à laquelle ce bloc de rapport de réception est reçu. Elle calcule le temps total aller-retour A – LSR en utilisant le champ LSR (horodate du dernier paquet SR), et en soustrayant ensuite le champ DLSR pour aboutir au temps de transmission aller-retour (A – LSR – DLSR). Cela est illustré à la Figure 4. Ce délai peut servir de mesure approximative de la distance à un groupe de récepteurs, même si certaines liaisons présentent des délais très asymétriques.



A	0xb710:8000	(46864,500 s)
– DLSR	0x0005:4000	(5,250 s)
– LSR	0xb705:2000	(46853,125 s)
= délai	0x0006:2000	(6,125 s)

Figure 4/J.121 – Exemple de calcul de temps de transmission aller-retour

7 Intervalle de transmission RTCP

Le protocole RTP est conçu pour permettre à une application de s'adapter automatiquement à des tailles de session allant de quelques participants à plusieurs milliers. Par exemple, dans une audioconférence, le trafic de données est intrinsèquement autolimitant car uniquement une ou deux personnes parleront à un instant donné, de sorte qu'avec une distribution par multidiffusion, le débit de données sur une liaison quelconque restera relativement constant et indépendant du nombre de participants. En revanche, le trafic de commande n'est pas autolimitant. Si les rapports de réception en provenance de chaque participant étaient envoyés à un débit constant, le trafic de commande croîtrait linéairement avec le nombre de participants. Par conséquent, le débit doit être diminué proportionnellement.

Pour chaque session, on suppose que le trafic de données est soumis à une limite globale appelée "largeur de bande de session" à répartir entre les participants. Cette largeur de bande peut être réservée et la limite imposée par le réseau, ou peut être uniquement une part raisonnable. La largeur de bande de session peut être choisie en fonction de certains coûts ou d'une connaissance à priori de la largeur de bande de réseau disponible pour la session. Elle est quelquefois indépendante du codage du média, mais le choix du codage peut être limité par la largeur de bande de session. Le paramètre largeur de bande de session devrait être fourni par une application de gestion de session lorsqu'il y a invocation d'une application média, mais les applications média peuvent également fixer une valeur par défaut déterminée à partir de la largeur de bande de données utilisée par un seul émetteur pour le codage retenu pour la session. L'application peut également imposer des limites de largeur de bande fondées sur les règles de portée de multidiffusion ou sur d'autres critères.

Pour les calculs de largeur de bande pour le trafic de commande et pour le trafic de données il faut tenir compte des protocoles des couches Transport et Réseau inférieures (UDP et IP par exemple) car c'est cela que le système de réservation des ressources devra connaître. L'application serait également censée connaître lequel des protocoles est utilisé. Les en-têtes de niveau liaison ne sont pas inclus dans le calcul car le paquet sera encapsulé avec différentes en-têtes niveaux de liaison au fur et à mesure de sa progression dans le réseau.

Le trafic de commande doit être limité à une faible fraction connue de la largeur de bande de session. Faible de manière à ce que la fonction principale du protocole de transport qui est d'acheminer les données, ne soit pas affectée; connu de sorte que le trafic de commande puisse être inclus dans la spécification de largeur de bande attribuée à un protocole de réservation des ressources, et de sorte que chaque participant peut calculer de manière indépendante la part qui lui est attribuée. Il est proposé de fixer à 5% la fraction de la largeur de bande de session attribuée au protocole RTCP. La valeur de ce paramètre et les autres constantes pour le calcul d'intervalle n'ont pas un caractère critique, mais tous les participants à la session doivent utiliser les mêmes valeurs afin que l'on puisse calculer le même intervalle. Par conséquent, ces constantes devraient être fixées pour un profil particulier.

L'algorithme décrit à l'Appendice IV respecte les objectifs précités. Il permet de calculer l'intervalle entre l'envoi de paquets RTCP composites afin de répartir la largeur de bande de trafic de commande attribuée entre les participants. On obtient ainsi pour une application donnée, une réaction rapide dans le cas de petites sessions où, par exemple, l'identification de tous les participants est importante, tout en pouvant s'adapter automatiquement à des sessions plus importantes. L'algorithme présente des caractéristiques suivantes:

- aux émetteurs est attribué collectivement au moins 1/4 de la largeur de bande de trafic de commande de sorte que, dans les sessions où il y a un grand nombre de récepteurs mais un faible nombre d'émetteurs, les participants nouvellement connectés recevront plus rapidement l'identificateur CNAME pour les sites émetteurs;

- l'intervalle entre les paquets RTCP doit être supérieur à 5 secondes au moins afin d'éviter un dépassement de la largeur de bande autorisée en présence de rafales de paquets RTCP, lorsque le nombre de participants est faible et que le trafic n'est pas lissé selon la loi des grands nombres;
- l'intervalle entre les paquets RTCP varie de manière aléatoire dans la fourchette [0,5; 1,5] fois l'intervalle calculé pour éviter une synchronisation intempestive de tous les participants. Le premier paquet RTCP envoyé après avoir rejoint une session est également retardé à cause de la variation aléatoire du demi-intervalle minimal RTCP dans le cas où l'application a démarré simultanément dans plusieurs sites, par exemple, lorsqu'elle est déclenchée par une annonce de session;
- une estimée dynamique de la taille moyenne de paquets RTCP composites est calculée, y compris tous ceux qui ont été reçus et envoyés, pour s'adapter automatiquement aux modifications du volume d'informations de commande transporté.

Cet algorithme peut être utilisé pour des sessions dans lesquelles tous les participants sont autorisés à émettre. Dans ce cas, le paramètre largeur de bande de session est le produit de la largeur de bande de chaque émetteur multiplié par le nombre de participants et la largeur de bande RTCP représente 5% de cette valeur.

Appendice I

Algorithmes

Le présent appendice contient des exemples représentés en code C relatifs à certains aspects des algorithmes pour émetteurs et récepteurs RTP. Il peut exister d'autres méthodes d'implémentation qui, dans certains contextes, sont plus rapides ou qui présentent d'autres avantages. Les présentes notes d'implémentation sont données uniquement à titre d'information et sont destinées à préciser la spécification du protocole RTP.

Les définitions suivantes sont utilisées dans tous les exemples; par soucis de clarté et de concision, la structure n'est valable que pour les architectures à 32 bits dans lesquelles l'octet plus significatif est placé en premier (dite "big-endian" en anglais). On suppose que les champs binaires sont conditionnés de manière serrée, les bits de plus fort poids étant placés en premier, sans remplissage additionnel. Il sera nécessaire néanmoins d'apporter certaines modifications pour obtenir une implémentation portable.

```
/*
   * rtp.h  --  Fichier d'en-têtes (RFC XXXX)
   */
#include <sys/types.h>

/*
   * Les définitions de types sont valables pour les architectures à 32 bits et
   * pourront devoir être ajustées pour des architectures à 16 ou à 64 bits.
   */
typedef unsigned char  u_int8;
typedef unsigned short u_int16;
typedef unsigned int   u_int32;
typedef short int16;

/*
   * Version actuelle du protocole.
   */
#define RTP_VERSION    2
```

```
#define RTP_SEQ_MOD (1<<16)
#define RTP_MAX_SDES 255          /* longueur maximale du texte pour le SDES */

typedef enum {
     RTCP_SR   = 200,
     RTCP_RR   = 201,
     RTCP_SDES = 202,
     RTCP_BYE  = 203,
     RTCP_APP  = 204
} rtcp_type_t;

typedef enum {
     RTCP_SDES_END   = 0,
     RTCP_SDES_CNAME = 1,
     RTCP_SDES_NAME  = 2,
     RTCP_SDES_EMAIL = 3,
     RTCP_SDES_PHONE = 4,
     RTCP_SDES_LOC   = 5,
     RTCP_SDES_TOOL  = 6,
     RTCP_SDES_NOTE  = 7,
     RTCP_SDES_PRIV  = 8
} rtcp_sdes_type_t;

/*
   * en-tête de données RTP
   */
typedef struct {
     unsigned int version:2;           /* version de protocole */
     unsigned int p:1;                 /* indicateur de remplissage */
     unsigned int x:1;                 /* indicateur d'extension d'en-tête */
     unsigned int cc:4;                /* décompte CSRC */
     unsigned int m:1;                 /* bit de marqueur */
     unsigned int pt:7;                /* type de charge utile */
     u_int16 seq;                      /* numéro de séquence */
     u_int32 ts;                       /* horodate */
     u_int32 ssrc;                     /* source de synchronisation */
     u_int32 csrc[1];                  /* liste CSRC optionnelle */
} rtp_hdr_t;

/*
   * Mot d'en-tête commun RTCP
   */
     typedef struct {
     unsigned int version:2;          /* version de protocole */
     unsigned int p:1;                /* indicateur de remplissage */
     unsigned int count:5;            /* varie en fonction du type de paquet */
     unsigned int pt:8;               /* RTCP type de paquet */
     u_int16 length;                  /* longueur de paquet en mots, sans ce
                                      /* mot */
} rtcp_common_t;

/*
   * Masque "big-endian" pour la version, couple bit-type de paquet
   */
#define RTCP_VALID_MASK (0xc000 | 0x2000 | 0xfe)
#define RTCP_VALID_VALUE ((RTP_VERSION << 14) | RTCP_SR)

/*
   * Bloc de rapport de réception
   */
typedef struct {
     u_int32 ssrc;                     /* source de données faisant l'objet du
                                       /* rapport */
```

```
     unsigned int fraction:8;           /* fraction perdue depuis le dernier
                                        /* rapport SR/RR */
     int lost:24;                       /* nombre cumuli de paquets perdus (avec
                                        /* signe) */
     u_int32 last_seq;                  /* numéro de la dernière séquence
                                        /* étendue */
     u_int32 jitter;                    /* gigue interarrivée*/
     u_int32 lsr;                       /* dernier paquet SR en provenance de
                                        /* cette source */
     u_int32 dlsr;                      /* temps écoulé depuis le dernier
                                        /* paquet SR*/
} rtcp_rr_t;

/*
   * élément SDES
   */
typedef struct {
     u_int8 type;                       /* type d'élément (rtcp_sdes_type_t) */
     u_int8 length;                     /* longueur de l'élément (en octets) */
     char data[1];                      /* texte, sans terminaison nulle */
     } rtcp_sdes_item_t;

/*
   * Un paquet RTCP
   */
typedef struct {
     rtcp_common_t common;              /* en-tête commun*/
     union {
       /* rapport d'émetteur (SR) */
       struct {
            u_int32 ssrc;                 /* émetteur produisant ce rapport*/
            u_int32 ntp_sec;              /* horodate NTP */
            u_int32 ntp_frac;
            u_int32 rtp_ts;               /* horodate RTP */
            u_int32 psent;                /* paquets envoyés */
            u_int32 osent;                /* octets envoyés */
            rtcp_rr_t rr[1];              /* liste de longueur variable */
       sr;

       /* rapport de réception (RR) */
       struct {
            u_int32 ssrc;               /* récepteur produisant ce rapport*/
            rtcp_rr_t rr[1];            /* liste de longueur variable */
       } rr;

       /* description de la source (SDES) */
       struct rtcp_sdes {
            u_int32 src;                /* premier SSRC/CSRC */
            p_sdes_item_t item[1];      /* liste des éléments SDES */
       sdes;

       /* BYE */
       struct {
            u_int32 src[1];             /* liste des sources */
            /* ne peut exprimer le texte de fin pour cette raison */
       } bye;
     } r;
} rtcp_t;

typedef struct rtcp_sdes rtcp_sdes_t;

/*
   * Information d'état par source
   */
```

```
typedef struct {
     u_int16 max_seq;                  /* numéro de séq. Le plus élevé
                                       /* visible */
     u_int32 cycles;                   /* décompte décalé de cycles de numéro
                                       /* de séquence */
     u_int32 base_seq;                 /* numéro de séquence de base*/
     u_int32 bad_seq;                  /* dernier numéro de "mauvaise"
                                       /* séq. + 1 */
     u_int32 probation;                /* paquets de séq. jusqu'à ce que la
                                       /* source soit valide */
     u_int32 received;                 /* paquets reçus */
     u_int32 expected_prior;           /* paquets attendus pendant dernier
                                       /* intervalle */
     u_int32 received_prior;           /* paquets reçus pendant dernier
                                       /* intervalle */
     u_int32 transit;                  /* temps de transm. relatif du paquet
                                       /* précédent*/
     u_int32 jitter;                   /* gigue estimée */
     /* ... */
} source;
```

Appendice II

Contrôles de validité des en-têtes RTP

Un récepteur RTP devrait contrôler la validité de l'en-tête RTP des paquets entrants car ils peuvent être cryptés ou provenir d'une application différente avec une erreur d'adresse. De même, lorsque le cryptage est activé, un contrôle de validité des en-têtes est nécessaire pour s'assurer que les paquets entrants ont été correctement décryptés, bien qu'un contrôle de validité avec un résultat "mauvais" (type de charge utile inconnu par exemple) ne signifie pas nécessairement que le décryptage a échoué. Seuls les contrôles de validité faibles sont possibles sur un paquet de données RTP provenant d'une source jusque-là non perçue:

- le champ version RTP doit être égal à 2;
- le type de charge utile doit être connu; en particulier, il ne doit pas être égal à SR ou à RR;
- si le bit P est positionné, le dernier octet du paquet doit contenir un décompte d'octets valide, en particulier, inférieur à la longueur totale du paquet moins la taille de l'en-tête;
- le bit X doit être égal à zéro si le profil ne spécifie pas que le mécanisme d'extension d'en-tête peut être utilisé. Dans les autres cas, le champ longueur d'extension doit être inférieur à la taille totale du paquet moins la longueur de l'en-tête fixe et le remplissage;
- la longueur du paquet doit correspondre au décompte CC et au type de charge utile (lorsque les charges utiles ont une longueur connue).

Les trois derniers contrôles sont quelque peu complexes et ne sont pas toujours réalisables, et de ce fait il ne reste que les deux premiers qui occupent uniquement quelques bits. L'identificateur SSRC dans le paquet est celui qui a été précédemment reçu, le paquet étant alors probablement valide et le contrôle pour s'assurer que le numéro de séquence se trouve bien dans la fourchette prévue permet une validation plus poussée. Si l'identificateur SSRC n'a pas été précédemment reçu, les paquets de données transportant cet identificateur peuvent être considérés comme non valides et ceci jusqu'à ce qu'un petit nombre de ces paquets parviennent avec des numéros de séquences consécutifs.

L'utilitaire `update_seq` présenté ci-dessus permet de s'assurer qu'une source est déclarée valide uniquement après que des paquets `MIN_SEQUENTIAL` aient été reçus en séquence. Il valide également le numéro de séquence `seq` d'un paquet nouvellement reçu et actualise l'état de séquence de la source du paquet dans la structure vers laquelle `s` pointe.

Lorsqu'une nouvelle source est perçue pour la première fois, c'est-à-dire, lorsque son identificateur SSRC ne figure pas dans le tableau et que l'état par source lui est attribué, s→probation devrait être le nombre de paquets séquentiels exigé pour déclarer qu'une source est valide (paramètre (MIN_SEQUENTIAL) et s→max_seq initialisé à seq-1 s→probation indique que la source n'est pas encore valide de sorte que l'état peut être effacé après une brève temporisation.

Après qu'une source soit considérée valide, le numéro de séquence est considéré comme étant valide s'il n'est pas supérieur de MAX_DROPOUT ni inférieur de MAX_MISORDER par rapport à s→max_seq. Si le nouveau numéro de séquence est supérieur à s→max_seq de max_seq modulo la fourchette de numéro de séquence RTP (16 bits), mais est inférieur à max_seq, il est réinitialisé et le décompte (décalé) de cycles de numéros de séquence est incrémenté. Une valeur de un est renvoyée pour indiquer un numéro de séquence valide.

Dans les autres cas, la valeur zéro est renvoyée pour indiquer que le contrôle de validation a échoué et le mauvais numéro de séquence est mémorisé. Si le paquet suivant reçu transporte le numéro de séquence supérieur suivant, il est considéré comme le début valide d'une nouvelle séquence de paquet supposé être dû à une interruption longue ou à un redémarrage de la source. Etant donné que plusieurs cycles de numéros de séquence complets ont été manqués, les statistiques de perte de paquets sont réinitialisées.

Des valeurs types de paramètres sont données, sur la base d'un temps maximal pendant lequel l'ordre est mauvais égal à 2 secondes à 50 paquets/seconde et d'une interruption maximale d'une minute. Le paramètre d'interruption MAX_DROPOUT devrait occuper une petite partie de l'espace numéro de séquence de 16 bits pour garantir avec une probabilité acceptable, que les nouveaux numéros de séquence après un redémarrage ne se trouvent pas dans la fourchette acceptable de numéros de séquence avant le redémarrage.

```
void init_seq(source *s, u_int16 seq)
{
     s->base_seq = seq - 1;
     s->max_seq = seq;
     s->bad_seq = RTP_SEQ_MOD + 1;
     s->cycles = 0;
     s->received = 0;
     s->received_prior = 0;
     s->expected_prior = 0;
     /* autre initialisation */
}

int update_seq(source *s, u_int16 seq)
{
     u_int16 udelta = seq - s->max_seq;
     const int MAX_DROPOUT = 3000;
     const int MAX_MISORDER = 100;
     const int MIN_SEQUENTIAL = 2;
     /*
     * La source n'est pas valide jusqu'à ce que des paquets MIN_SEQUENTIAL
     * avec des numéros de séquence consécutifs aient été reçus.
     */
     if (s->probation) {
     /* le paquet est en séquence */
       if (seq == s->max_seq + 1) {
          s->probation--;
          s->max_seq = seq;
          if (s->probation == 0) {
           init_seq(s, seq);
           s->received++;
           return 1;
          }
       }
```

```
        else {
           s->probation = MIN_SEQUENTIAL - 1;
           s->max_seq = seq;
        }
        return 0;
      }
      else if (udelta < MAX_DROPOUT) {
        /* dans l'ordre, avec un intervalle autorisé gap */
        if (seq < s->max_seq) {
        /*
        * Numéro de séquence rebouclé - comptage d'un autre cycle de 64K.
        */
           s->cycles += RTP_SEQ_MOD;
        }
        s->max_seq = seq;
      } else if (udelta <= RTP_SEQ_MOD - MAX_MISORDER) {
        /* saut important du numéro de séquence */
        if (seq == s->bad_seq) {
           /*
           * Deux paquets en séquence -- suppose que l'autre côté
           * a redémarré sans nous aviser, il faut donc uniquement resynchroniser
           * (c'est-à-dire, prétend qu'il s'agissait du premier paquet).
           */
           init_seq(s, seq);
        }
        else {
           s->bad_seq = (seq + 1) & (RTP_SEQ_MOD-1);
           return 0;
        }
      } else {
        /* paquet dupliqué ou réordonné */
      }
      s->received++;
      return 1;
}
```

Le contrôle de validité peut être rendu plus puissant, mais il faut pour cela plus de deux paquets en séquence. Les inconvénients tiennent à ce qu'un grand nombre de paquets initiaux seront éliminés et que le taux de perte élevé de paquets pourrait alors empêcher toute validation. Toutefois, comme la validation de l'en-tête RTCP est relativement puissante, si un paquet RTCP est reçu en provenance d'une source avant les paquets de données, le comptage pourrait être ajusté de sorte qu'il faudrait que deux paquets soient en séquence. Si une perte de données initiales pendant quelques secondes peut être tolérée, on pourra pour une application choisir d'éliminer tous les paquets de données en provenance d'une source jusqu'à ce qu'un paquet RTCP valide ait été reçu de la source en question.

En fonction de l'application et du codage, les algorithmes peuvent utiliser les informations supplémentaires dont ils disposent concernant le format de charge utile pour effectuer une validation plus poussée. Pour les types de charge utile dans lesquels l'incrément d'horodate est le même pour tous les paquets, les valeurs d'horodate peuvent être déduites des précédents paquets reçus en provenance de la même source en utilisant la différence de numéro de séquence (en supposant qu'aucune modification de type de charge utile n'a été apportée).

Un puissant contrôle par "raccourci" est possible étant donné que les quatre premiers octets de l'en-tête d'un nouveau paquet de données RTP nouvellement reçus seront avec une forte probabilité, les mêmes que ceux du paquet précédent provenant de la même SSRC à ceci près que le numéro de séquence aura été augmenté de un.

De même, un cache à entrée unique peut être utilisé pour des consultations SSRC plus rapides dans des applications où les données sont en général reçues en provenance d'une source à la fois.

Appendice III

Détermination du nombre de paquets RTP attendus ou perdus

Afin de calculer le taux de perte de paquets, le nombre de paquets attendus et réellement reçus de chaque source doit être connu, en utilisant pour cela l'information d'état par source définie dans `struct` source référencée via le pointeur `s` dans le code ci-dessous. Le nombre de paquets reçus est simplement le décompte des paquets au fur et à mesure qu'ils arrivent, en incluant les éventuels paquets en retard ou les paquets en double. Le nombre de paquets attendus peut être calculé par le récepteur comme étant la différence entre le numéro de séquence le plus élevé reçu (`s->max_seq`) et le premier numéro de séquence reçu (`s->base_seq`). Etant donné que le numéro de séquence occupe seulement 16 bits et se rebouclera, il est nécessaire de compléter le numéro de séquence le plus élevé par le nombre (décalé) de rebouclages de numéros de séquence (`s->cycles`). Le décompte des paquets reçus et le décompte de cycles est effectué dans l'utilitaire de contrôle de validité de l'en-tête RTP decrit à l'Appendice II.

```
extended_max = s->cycles + s->max_seq;
expected = extended_max – s->base_seq + 1;
```

Le nombre de paquets perdus est défini comme étant le nombre de paquets attendus moins le nombre de paquets réellement reçus:

```
lost = expected – s->received;
```

Etant donné que ce nombre est acheminé sur 24 bits, il devrait être limité à 0xffffff au lieu de le reboucler à zéro.

La fraction des paquets perdus au cours du dernier intervalle de rapport (étant donné que les précédents paquets SR ou RR ont été envoyés) est calculée à partir des différences entre le nombre de paquets attendus et le nombre de paquets reçus pendant l'intervalle considéré, où `expected_prior` et `received_prior` sont les valeurs sauvegardées lorsque le précédent rapport de réception a été produit:

```
expected_interval = expected – s->expected_prior;
s->expected_prior = expected;
received_interval = s->received – s->received_prior;
s->received_prior = s->received;
lost_interval = expected_interval – received_interval;
if (expected_interval == 0 || lost_interval <= 0) fraction = 0;
else fraction = (lost_interval << 8) / expected_interval;
```

La fraction résultante est un nombre de 8 bits à virgule fixe, la virgule binaire se trouvant au bord gauche.

Appendice IV

Calcul de l'intervalle de transmission RTCP

La fonction ci-après calcule l'intervalle de temps, en secondes, entre la transmission de paquets RTCP consécutifs. Cette fonction devrait être appelée après l'envoi d'un paquet RTCP composite pour calculer le délai au bout duquel le paquet suivant devrait être envoyé. Cette fonction devrait être également appelée pour calculer le délai avant l'envoi du premier paquet RTCP après un démarrage plutôt que d'envoyer le paquet immédiatement. Cela évitera la présence de rafales de paquets RTCP si une application est démarrée simultanément dans un grand nombre de sites, par exemple suite à l'annonce d'une session.

Les paramètres ont les significations suivantes:

rtcp_bw: largeur de bande RTCP cible, c'est-à-dire, la largeur de bande totale qui sera utilisée pour les paquets RTCP par tous les participants à la session, en octets par seconde. Cela devrait représenter 5% du paramètre "largeur de bande de session" indiqué à l'application lors du démarrage.

senders: nombre d'émetteurs actifs depuis l'envoi du dernier rapport, déduit de l'élaboration des rapports de récepteurs pour le paquet RTCP considéré. Nous inclus, si nous avons également émis pendant cet intervalle.

members: il s'agit du nombre estimé de membres participant à la session, y compris nous-mêmes, doit être incrémenté si l'on constate qu'il y a de nouveaux participants à la session à partir de la réception de paquets RTP ou RTCP, et décrémenté lorsque des participants quittent la session (via RTCP BYE) ou que leur temporisation d'état a expiré (la valeur de 30 minutes est recommandée). Lors du premier appel, le paramètre devrait avoir la valeur 1.

we_sent: il s'agit d'un indicateur qui est positionné à Vrai si des données ont été envoyées pendant les deux derniers intervalles RTCP. Si cet indicateur est positionné à Vrai, le paquet composite RTCP qui vient d'être envoyé contenait un paquet SR.

packet_size: taille du paquet composite RTCP qui vient d'être envoyé, en octets, y compris l'encapsulation réseau (par exemple, 28 octets pour l'UDP sur IP).

avg_rtcp_size: pointeur vers l'estimateur de taille de paquet composite RTCP; initialisé et actualisé par cette fonction pour le paquet qui vient d'être envoyé, et également mis à jour par une ligne de code identique dans l'utilitaire de réception RTCP pour chaque paquet RTCP reçu en provenance d'autres participants à la session.

initial: il s'agit d'un indicateur qui est positionné à Vrai pour le premier appel au démarrage afin de calculer le temps au bout duquel le premier rapport devrait être envoyé.

```
#include <math.h>

     double rtcp_interval(int members,
     int senders,
     double rtcp_bw,
     int we_sent,
     int packet_size,
     int *avg_rtcp_size,
     int initial)
     {
     /*
     * Temps minimal entre paquets RTCP provenant de ce site (en secondes).
     * Ce temps évite l'accumulation des rapports lorsque les sessions sont
     * petites et que la loi des grands nombres ne permet pas de lisser le
     * trafic. Il empêche également l'intervalle entre rapports de devenir
```

```
* ridiculement faible pendant des interruptions transitoires dans le cas
* par exemple d'une partition du réseau.
*/
double const RTCP_MIN_TIME = 5.;
/*
* Fraction de la largeur de bande RTCP que l'émetteur actif
* doit partager (cette fraction a été choisie de manière telle que dans
* une session type avec un ou deux émetteurs actifs, le temps de rapport
* calculé sera approximativement égal au temps minimal de rapport,
* de sorte qu'il ne sera pas nécessaire de ralentir l'émission des rapports
* de récepteur). La fraction pour le récepteur doit être égale à 1 moins la
* fraction pour l'émetteur.
*/
double const RTCP_SENDER_BW_FRACTION = 0.25;
double const RTCP_RCVR_BW_FRACTION = (1-RTCP_SENDER_BW_FRACTION);
/*
* Gain (constante de lissage) pour le filtre passe-bas qui fait
* l'estimation de la taille moyenne des paquets RTCP (voir la
* référence Cadzow).
*/
double const RTCP_SIZE_GAIN = (1./16.);

double t; /* interval */
double rtcp_min_time = RTCP_MIN_TIME;
int n;   /* nombre de membres pour le calcul */

/*
* Le tout premier appel après le démarrage de l'application utilise la
* moitié du délai minimal pour effectuer une notification plus rapide
* tandis qu'il autorise un certain délai avant l'établissement du rapport
* pour la randomisation et pour connaître les autres sources, de sorte que
* l'intervalle entre rapports convergera vers la valeur d'intervalle
* correcte plus rapidement. La taille moyenne RTCP est initialisée à
* 128 octets, ce qui est une valeur prudente (on suppose que chacun des
* autres produits des rapports SR et non pas des rapports RR: 20 IP + 8 UDP
* + 52 SR + 48 SDES CNAME).
*/
if (initial) {
rtcp_min_time /= 2;
*avg_rtcp_size = 128;
}

/*
* S'il y avait des émetteurs actifs, leur attribuer au moins une partie
* minimale de la largeur de bande RTCP. Dans les autres cas, tous les
* participants partagent la largeur de bande RTCP de manière égale.
*/
n = members;
if (senders > 0 && senders < members * RTCP_SENDER_BW_FRACTION) {
if (we_sent) {
rtcp_bw *= RTCP_SENDER_BW_FRACTION;
n = senders;
} else {
rtcp_bw *= RTCP_RCVR_BW_FRACTION;
n -= senders;
}
}

/*
* Mise à jour de l'estimée de la taille moyenne par
* la taille du paquet de rapport qui vient d'être envoyé.
*/
*avg_rtcp_size += (packet_size - *avg_rtcp_size)*RTCP_SIZE_GAIN;
```

```
/*
* Le nombre effectif de sites multiplié par la taille moyenne des paquets
* est égal au nombre total d'octets envoyés lorsque chaque site envoie un
* rapport. Si l'on divise cette valeur par la largeur de bande effective,
* on obtient l'intervalle de temps pendant lequel ces paquets doivent être
* envoyés afin de respecter l'objectif de largeur de bande, avec un minimum
* imposé. Pendant cet intervalle de temps, on envoie un rapport de sorte
* que ce temps soit également l'intervalle de temps moyen entre rapports.
*/
t = (*avg_rtcp_size) * n / rtcp_bw;
if (t < rtcp_min_time) t = rtcp_min_time;

/*
* Pour éviter les rafales de trafic issues d'une synchronisation
* intempestive avec d'autres sites, un intervalle de rapport suivant réel
* est pris sous forme d'un nombre aléatoire uniformément distribué
* entre 0,5*t et 1,5*t.
*/
return t * (drand48() + 0.5);
}
```

Appendice V

Estimation de la gigue interarrivée

Les éléments de code ci-dessous implémentent l'algorithme donné au § 6.2 pour calculer une estimée de la variance statistique du temps interarrivée de données RTP à insérer dans le champ gigue interarrivée des rapports de réception. Les entrées sont `r->ts`, l'horodate provenant du paquet entrant, et l'arrivée, le temps courant exprimé dans les mêmes unités. Ici, le pointeur `s` pointe sur l'état pour la source; `s->transit` conserve le temps de transit relatif pour le paquet précédent, et `s->jitter` conserve la gigue estimée. Le champ gigue du rapport de réception est mesuré en unités d'horodate et exprimé sous forme d'un entier sans signe, mais l'estimée de gigue reste en virgule flottante. A chaque fois qu'un paquet de données arrive, l'estimée de la gigue est actualisée:

```
int transit = arrival - r->ts;
int d = transit - s->transit;
s->transit = transit;
if (d < 0) d = -d;
s->jitter += (1./16.) * ((double)d - s->jitter);
```

Lorsqu'un bloc de rapport de réception (vers lequel `rr` pointe) est généré pour ce membre, l'estimée de la gigue courante est retournée:

```
rr->jitter = (u_int32) s->jitter;
```

De même, l'estimée de la gigue peut être maintenue sous forme d'un entier, mais rapportée pour réduire l'erreur d'arrondi. Le calcul est le même sauf pour la dernière ligne:

```
s->jitter += d - ((s->jitter + 8) >> 4);
```

Dans ce cas, l'estimée est échantillonnée pour le rapport de réception comme suit:

```
rr->jitter = s->jitter >> 4;
```

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	**Réseaux câblés et transmission des signaux radiophoniques, télévisuels et autres signaux multimédias**
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	RGT et maintenance des réseaux: systèmes de transmission, circuits téléphoniques, télégraphie, télécopie et circuits loués internationaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	Réseaux de données et communication entre systèmes ouverts
Série Y	Infrastructure mondiale de l'information et protocole Internet
Série Z	Langages et aspects généraux logiciels des systèmes de télécommunication

Imprimé en Suisse
Genève, 2002