Multicast

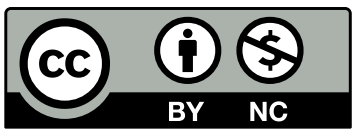
Pierre David
pda@unistra.fr

Université de Strasbourg – Master CSMI

2023 – 2024

Plan

Licence d'utilisation



Plan

Introduction

Besoin : communication 1 $\rightarrow n$ ou $n \rightarrow n$

- applications de partage :
  Exemples :
  - visio-conférence
  - diffusion de flux de type TV
  - tableau blanc partagé
  - édition de texte partagée
- rationalisation de trafics lourds :
  Exemple : diffusion des news, de la TV en direct
- simplicité de configuration :
  Exemple : désigner tous les équipements ayant une certaine propriété (ex: tous les routeurs du réseau local) sans connaître leur adresse précise

Introduction

Problème : comment faire pour que ces flux ne passent pas *n* fois par les mêmes câbles ?

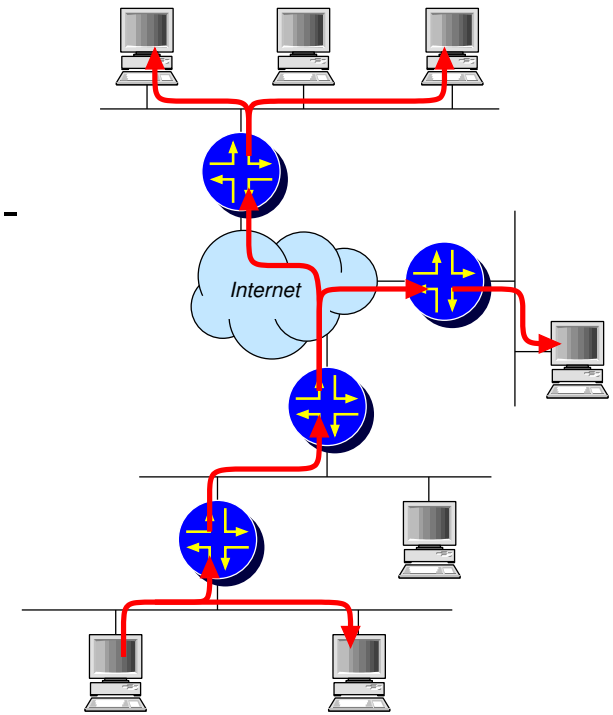
Solution : *multicast* (diffusion multipoint)

- *multicast* = *broadcast* limité à quelques nœuds, mais qui peut franchir les routeurs
- vu d'un émetteur, le *multicast* correspond à un arbre dont il est la racine
- la diffusion ne se fait que vers les sites concernés
  $\Rightarrow$ notion d'appartenance dynamique à un groupe

Introduction

Multicast : émettre un datagramme vers plusieurs destinations en une seule opération

Les applications *multicast* utilisent UDP
(TCP $\Rightarrow$ connexion entre deux parties)



Introduction – MBone

Problème : le *multicast* requérait des extensions dans les couches IP
$\Rightarrow$ déploiement lent

Solution : réseau virtuel « par dessus » Internet
$\Rightarrow$ MBone (pour *Multicast BackBone*)

Le MBone était le support de la recherche en matière :

- de protocoles de routage *multicast*
- de protocoles de transport audio/vidéo/etc
- d'applications nouvelles

Mais le Mbone était aussi utilisé en exploitation, jusqu'à sa disparition.

Introduction – MBone

Historique :

- décembre 1985 : première proposition des spécifications de base *multicast* (protocole IGMP)
- juillet 1986 : IGMP version 0
- novembre 1988 : protocole de routage DVMRP
- août 1989 : IGMP version 1
- 1992 : programme `mrouted` et création du MBone
- mars 1992 : première retransmission (audio) d'une conférence de l'IETF
- fin 1993 : création du FMBone (*French MBone*)
- 1994 : IGMP version 2
- 2002 : IGMP version 3
- 2008 : arrêt du MBone

Introduction – Problèmes à résoudre

Problèmes du multicast :

- comment nommer les destinataires ?
  $\Rightarrow$ notion d'adresse de *groupe multicast*
- comment diffuser sur un réseau Ethernet ?
  $\Rightarrow$ encapsulation dans une trame Ethernet
- comment connaître les destinataires ?
  $\Rightarrow$ appartenance dynamique aux groupes
- comment effectuer une propagation sans cycle ?
  $\Rightarrow$ protocoles de routage

Plan

Principes – Groupes

Adresses *multicast* :

- adresses IP de classe D (224.0.0.0 à 239.255.255.255)
- désigne un groupe de machines
  Une machine appartient à un groupe $\Rightarrow$ elle reçoit tous les datagrammes émis à destination de l'adresse de ce groupe
- certains groupes sont prédéfinis. Par exemple :

224.0.0/24	adresses locales : routage et maintenance
224.0.0.1	toutes les machines sur ce réseau
224.0.0.2	tous les routeurs sur ce réseau
224.0.0.251	multicast DNS (mDNS)
224.0.1/24	trafic de routage propagé sur l'Internet
224.0.1.1	diffusion de l'heure
224.0.1.187	tous les nœuds CoAP
224.2/16	adresses SDP/SAP choisies aléatoirement

Principes – Groupes

L'appartenance d'une machine à un groupe est dynamique.

Le système d'exploitation d'une machine offre aux processus deux opérations de base :

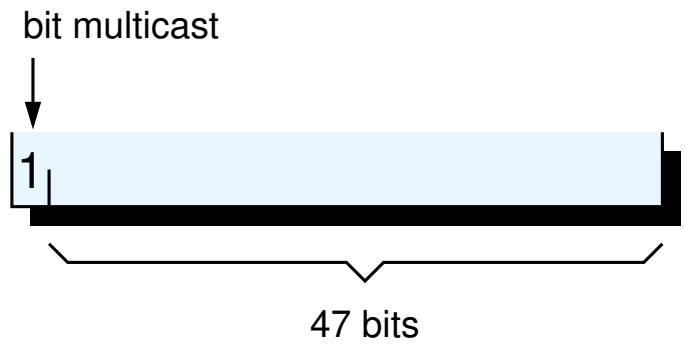
- ajouter un nouveau groupe à la liste des groupes reconnus par cette machine
- retirer un groupe existant de la liste des groupes reconnus par cette machine

Exemple : pour recevoir une conférence IETF, il faut ajouter les groupes 224.0.1.11 (pour le son) et 224.0.1.12 (pour les images) à la liste des groupes reconnus.

Principes – Diffusion sur un réseau Ethernet

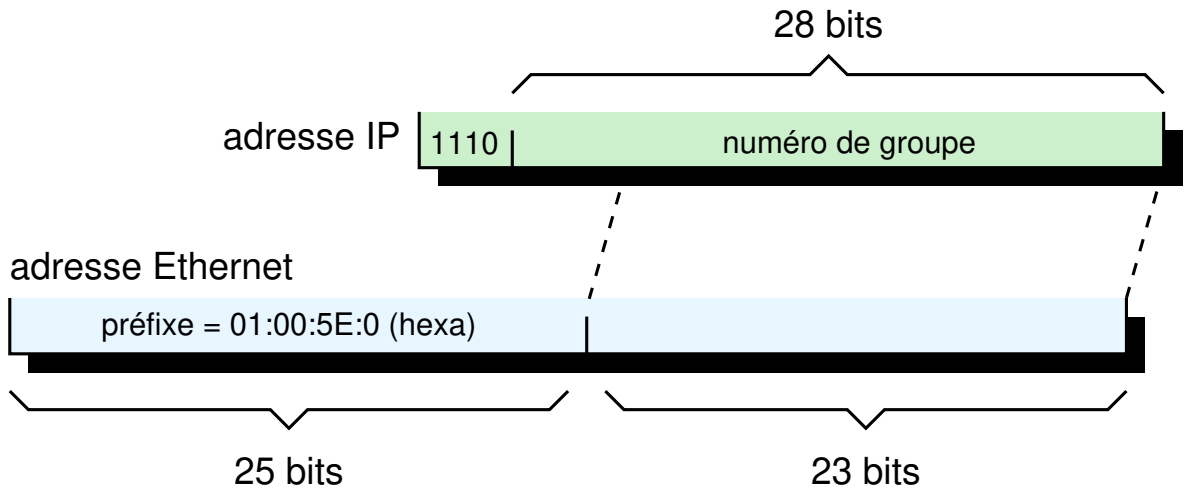
Ethernet permet la diffusion multipoint

$\Rightarrow$ bit 8 dans l'adresse Ethernet (01-00-00-00-00-00)



Principes – Diffusion sur un réseau Ethernet

Codage d'une adresse de groupe *multicast* (adresse IP) dans une adresse Ethernet :



Codage non bijectif $\Rightarrow$ les récepteurs doivent vérifier l'adresse figurant dans l'en-tête du datagramme IP

Pas besoin de ARP pour la classe D !

Plan

Principes

À quels destinataires envoyer un datagramme ?
Séparation du problème en deux sous-problèmes :

- protocole machine/routeur
  ⇒ pour que le routeur apprenne quels groupes le réseau local doit recevoir
  Protocole IGMP (partie « machine/routeur »)
- protocole routeur/routeur
  ⇒ pour que les datagrammes circulent dans l'Internet depuis la source jusqu'au routeur
  Plusieurs protocoles de routage possibles
  ⇒ pas de « meilleur protocole »

Principes – Appartenance aux groupes

IGMP = *Internet Group Management Protocol*

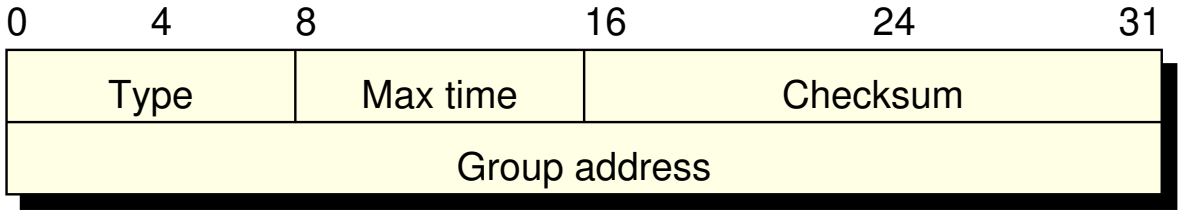
Protocole utilisé pour :

- obtenir ou annoncer l'appartenance à un groupe
  $\Rightarrow$ protocole routeur/feuilles
- propager les informations de certains protocoles de routage (i.e. DVMRP)
  $\Rightarrow$ protocole routeur/routeur

Les paquets IGMP sont encapsulés dans des datagrammes IP

Principes – Appartenance aux groupes

Partie « protocole routeur/feuilles » de IGMP

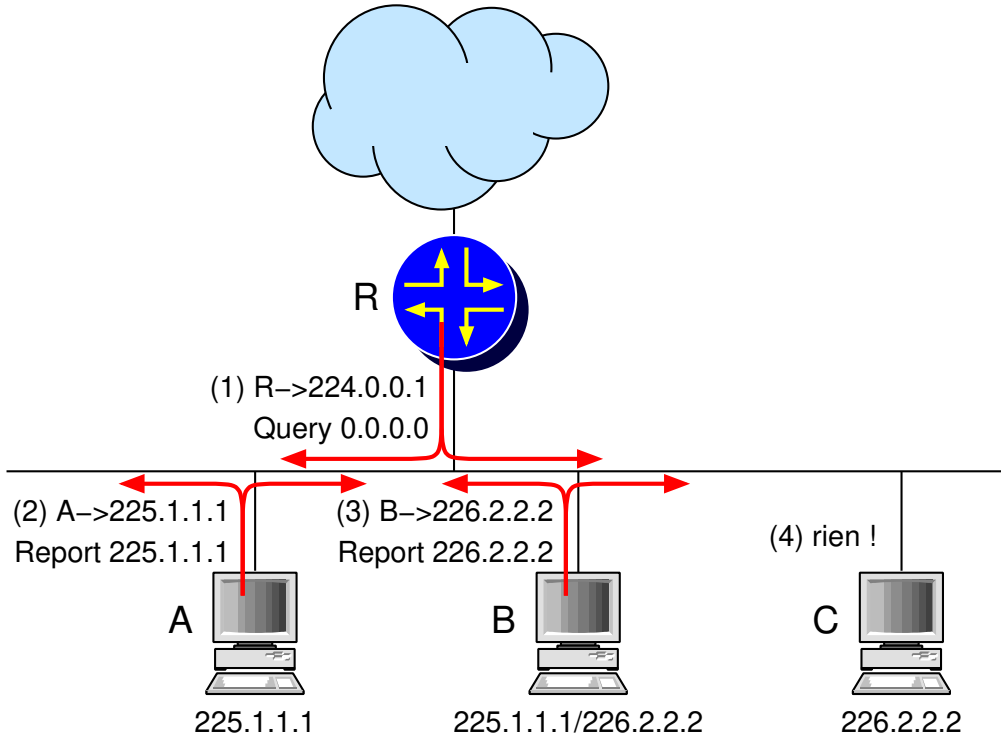


avec :

- *type* = *Membership Query*
  $\Rightarrow$ interrogation périodique par le routeur
- *type* = *Membership Report*
  $\Rightarrow$ annonce d'appartenance par une machine
- *type* = *Leave Group*
  $\Rightarrow$ annonce de retrait d'un groupe par une machine

Principes – Appartenance aux groupes

Interrogation IGMP périodique par le routeur :



Optimisations :

- réponses envoyées après un délai aléatoire borné
- pas de réponse si une autre machine réagit avant
- TTL = 1

Principes – Appartenance aux groupes

Problème : plusieurs routeurs sur un même réseau

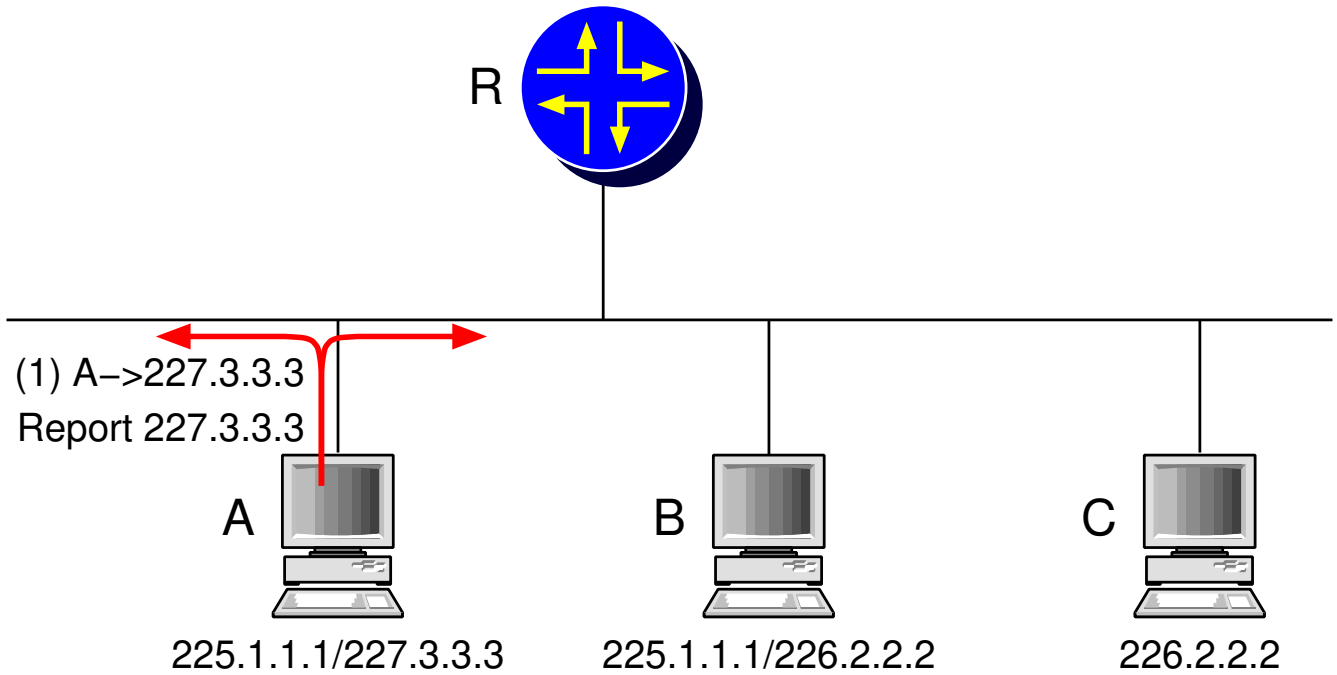
Solution : élection d'un routeur « interrogateur »

Méthode : routeur avec l'adresse IP la plus basse

- tous les routeurs commencent à émettre des interrogations (*Query*)
- si un routeur reçoit un message *Query* en provenance d'une adresse plus petite, il arrête d'émettre
  $\Rightarrow$ mais il continue à écouter
- si un routeur n'a reçu aucun message *Query* au bout d'un certain délai, il recommence à émettre des interrogations

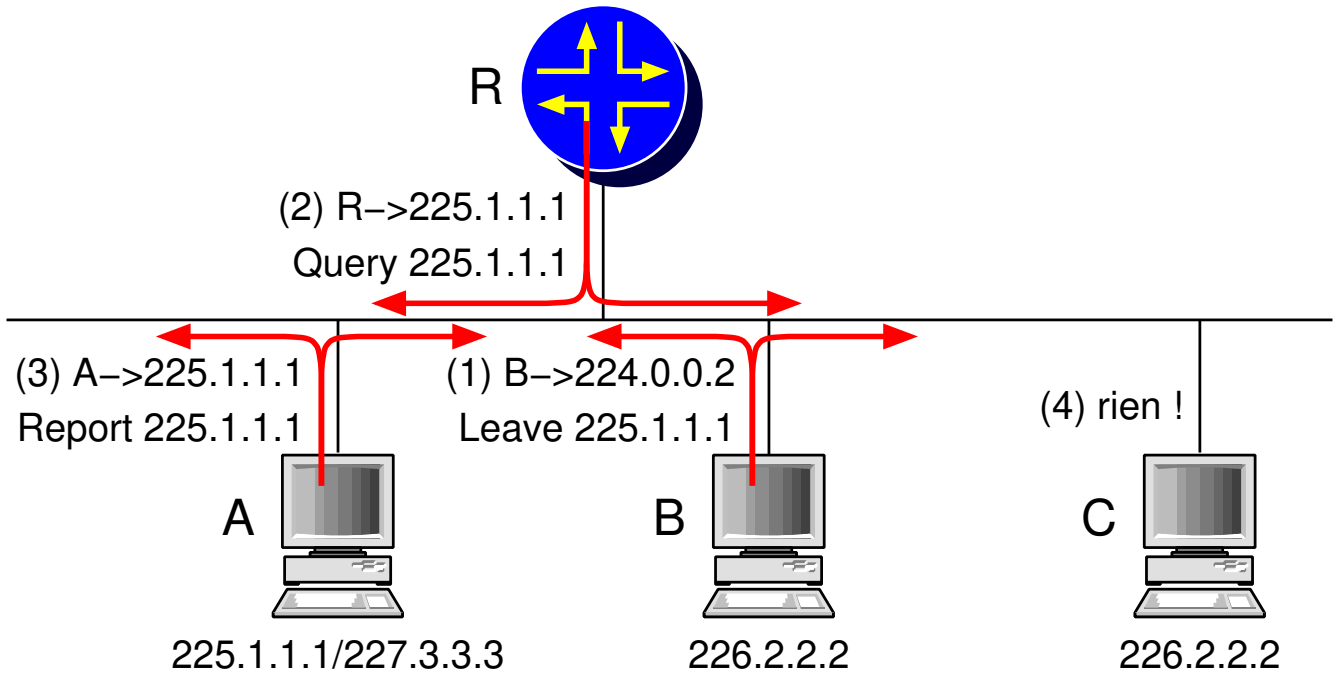
Principes – Appartenance aux groupes

Ajout d'un groupe : A ajoute le groupe 227.3.3.3



Principes – Appartenance aux groupes

Retrait d'un groupe : B retire le groupe 225.1.1.1



Plan

Routage – Algorithmes

Comment propager les datagrammes dans le réseau ?
$\Rightarrow$ le chemin suit un arbre

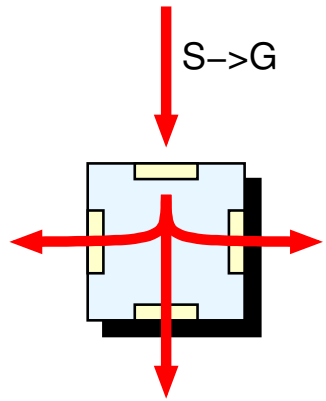
Problème : comment construire l'arbre ?

Plusieurs grandes familles d'algorithmes :

- inondation
- arbres couvrants
- arbres enracinés à la source
  - simples
  - avec élagage
- arbres partagés par un groupe

Inondation

Algorithme de l'inondation (*flooding*) :
Principe : un routeur envoie le datagramme sur toutes les interfaces, sauf sur celle qui a reçu le datagramme.

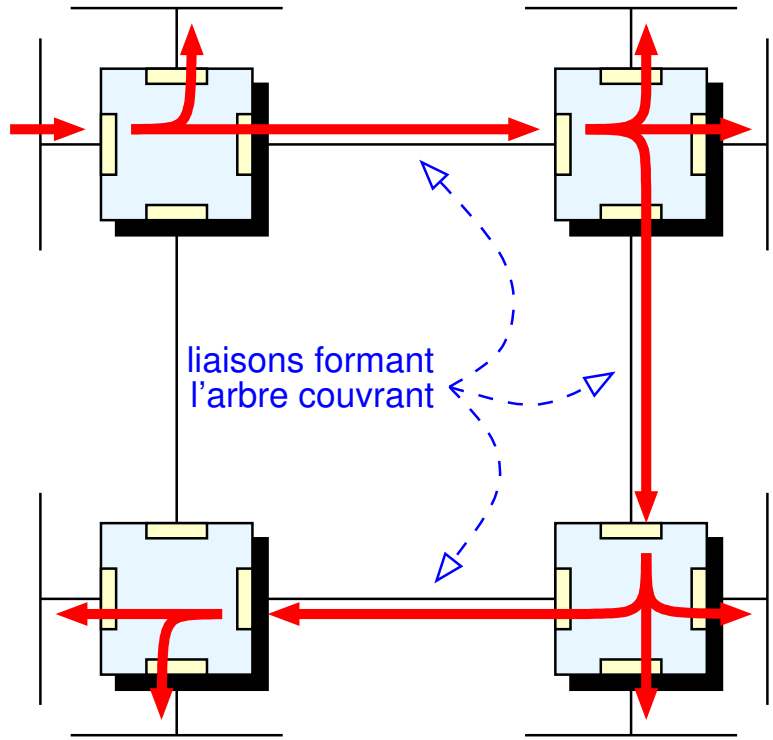


Avantage : simple
Inconvénients :

- il faut conserver trace des derniers datagrammes reçus (car il peut y avoir des cycles dans le graphe)
  $\Rightarrow$ il faut beaucoup de mémoire sur les routeurs
- le datagramme est envoyé systématiquement
  $\Rightarrow$ pas d'élagage (*pruning*) de l'arbre

Arbre couvrant

Algorithme de l'arbre couvrant (*spanning tree*) :
Principe : former un arbre couvrant à partir du graphe de tous les routeurs



Arbre couvrant

Avantages :

- simple
- peu de ressources sur les routeurs

Inconvénients :

- un seul arbre couvrant pour tout l'Internet
  $\Rightarrow$ pas de routage en fonction de la source
- surcharge des liaisons de l'arbre
  $\Rightarrow$ centralisation sur l'arbre
- pas très efficace
  $\Rightarrow$ ne choisit pas toujours le chemin le plus court
- le datagramme est envoyé systématiquement
  $\Rightarrow$ pas d'élagage (*pruning*) de l'arbre

Reverse Path Broadcasting

Algorithme *Reverse Path Broadcasting*

Principe :

- chaque routeur a une table de routage *unicast*
  $\Rightarrow$ le routeur connaît le *meilleur chemin* pour S
- si un datagramme S$\rightarrow$G arrive par l'interface correspondant au meilleur chemin
  $\Rightarrow$ il est transmis à toutes les autres interfaces
- si un datagramme S$\rightarrow$G arrive par une autre interface
  $\Rightarrow$ il est ignoré

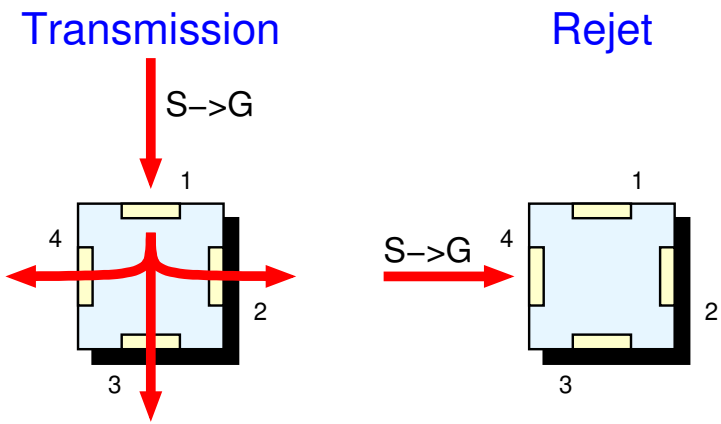
Reverse Path Broadcasting

Exemple : la table de routage *unicast* indique

Pour aller à	Passer par
S	interface 1

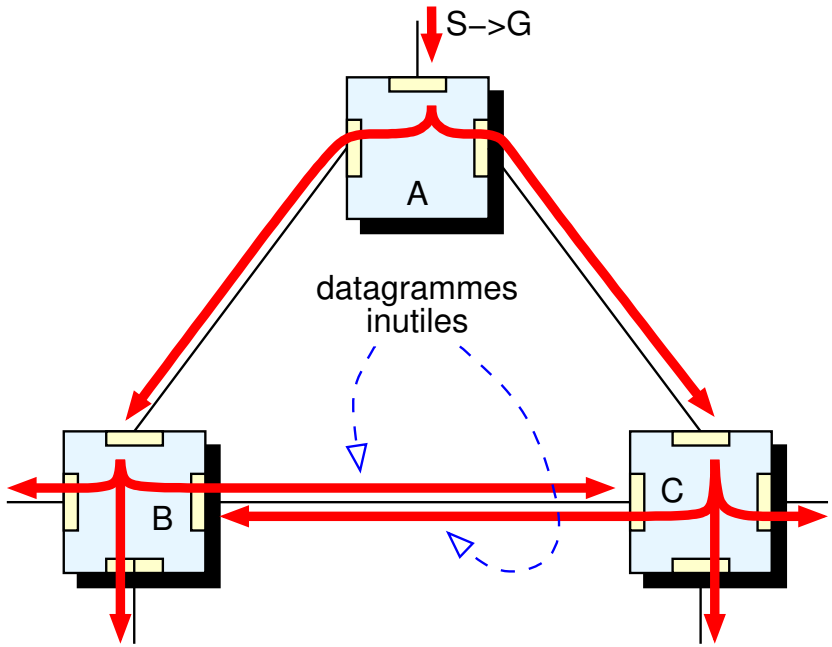
Deux cas :

- si un datagramme *multicast* émis par S arrive par l'interface 1, il est retransmis
- si un datagramme *multicast* émis par S arrive par une interface autre que 1, il est ignoré



Reverse Path Broadcasting

Amélioration possible :



Principe : si B connaît la table de routage de C, il sait qu'il n'est pas le « meilleur chemin » C $\rightarrow$ S : B ne transmet pas à C

Idem pour C.

Reverse Path Broadcasting

Avantages :

- simple
- efficace au niveau des routeurs
  $\Rightarrow$ peu de ressources sur chaque routeur
- efficace au niveau du réseau
  $\Rightarrow$ utilisation du « meilleur chemin »

Inconvénients :

- le datagramme est envoyé systématiquement
  $\Rightarrow$ pas d'élagage (*pruning*) de l'arbre

Truncated Reverse Path Broadcasting

Algorithme *Truncated Reverse Path Broadcasting*

Similaire à l'algorithme précédent, mais sans diffusion vers les réseaux *feuilles* si ce n'est pas nécessaire

Le datagramme est donc :

- envoyé systématiquement à tous les routeurs
- envoyé lorsque nécessaire sur les réseaux feuilles

$\Rightarrow$ résoud une partie du problème !

Reverse Path Multicast

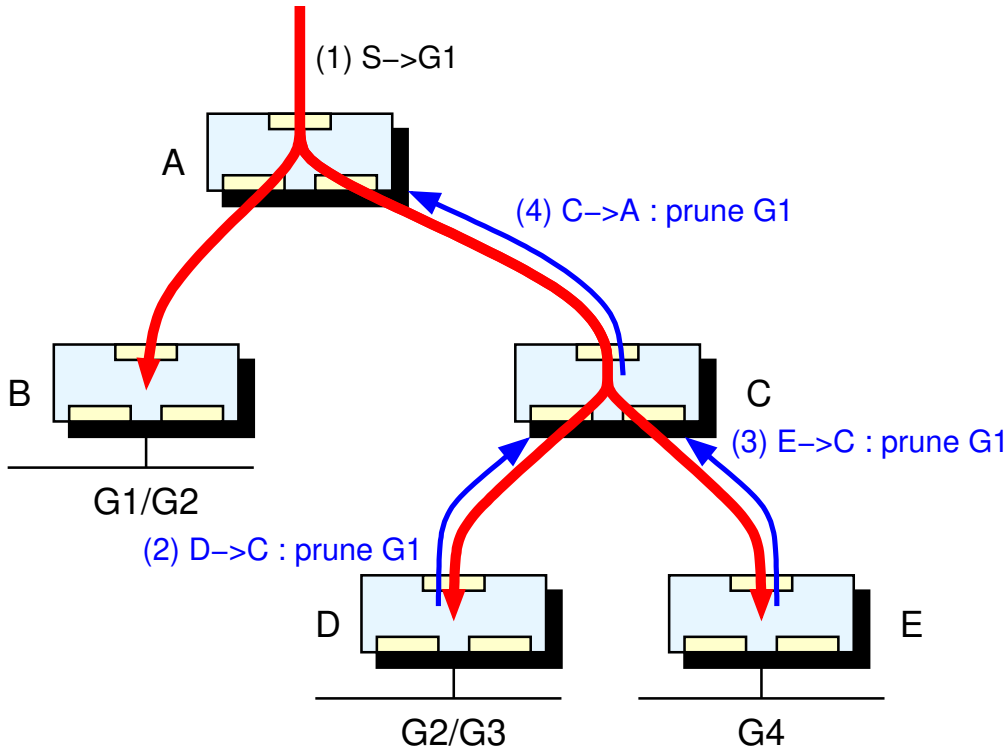
Algorithme *Reverse Path Multicast* (cf protocole DVMRP)
Principe :

- lorsqu'un datagramme S$\rightarrow$G est reçu, il est envoyé sur tous les nœuds par l'algorithme TRPB
- si un routeur n'a pas de feuille appartenant au groupe G, il envoie un message d'élagage (*prune*) au routeur précédent (en amont)
- si un routeur reçoit des messages d'élagage de tous les routeurs situés en aval, il envoie un message d'élagage au routeur amont
- les informations d'élagage doivent être expirées périodiquement, et on recommence alors au début

DVMRP ajoute la possibilité de greffes (*grafts*) : un routeur qui a récemment envoyé un message d'élagage pour un groupe G peut demander à remettre la branche dans l'arbre de diffusion s'il reçoit une adhésion pour G

Reverse Path Multicast

Exemple :



Reverse Path Multicast

Avantages :

- simple
- prend en compte les appartenances aux groupes
  $\Rightarrow$ n'utilise pas trop de ressources réseau

Inconvénients :

- envoi général lors du premier datagramme
  $\Rightarrow$ attention aux changements d'échelle !
- consomme des ressources sur les routeurs
  $\Rightarrow$ une entrée par couple (S,G) actif

L'algorithme RPM (implémenté par le protocole DVMRP) est à l'heure actuelle le plus répandu

Arbres partagés par un groupe

Algorithme des arbres partagés par un groupe

Principe : définir un arbre couvrant pour l'ensemble des membres d'un groupe
$\Rightarrow$ diffusion faite sans tenir compte de l'adresse de la source

Avantages :

- moins de ressources dans les routeurs
- ne requiert pas la collaboration de *tous* les routeurs du réseau
- pas d'inondation périodique sur le réseau

Inconvénients :

- concentration et congestion sur l'arbre couvrant
- peut conduire à des routages inutilement longs

Plan

Routage – Protocoles

Comment ces différents algorithmes sont-ils utilisés ?
$\Rightarrow$ protocole de routage multicast

Un protocole de routage multicast :

- implémente un ou plusieurs algorithmes de diffusion
- maintient des informations spécifiques au *multicast*
- utilise une table de routage *unicast*
- définit un format de paquets

Spécialisation des protocoles en deux catégories :

- mode *dense* (*dense mode*)
- mode *clairsemé* (*sparse mode*)

Routage en mode dense

Utilisation du mode *dense* :

- beaucoup de bande passante, *ou*
- grande densité de destinataires

Idée :

- on part du principe que tous les nœuds sont intéressés par tous les groupes
  $\Rightarrow$ on propage vers tous les nœuds par défaut
- si un nœud ne veut pas d'un groupe, il réagit
  $\Rightarrow$ envoi d'un message d'élagage

Mode *dense* adapté aux réseaux de sites
Protocoles DVMRP, MOSPF, PIM-DM

Routage en mode clairsemé

Utilisation du mode *clairsemé* :

- peu de bande passante, *et*
- destinataires éventuellement nombreux, mais très dispersés

Idée :

- on part du principe qu'aucun nœud n'est intéressé par un groupe
  $\Rightarrow$ on n'envoie rien par défaut
- si un nœud veut appartenir à un groupe, il envoie explicitement un message d'adhésion

Mode *clairsemé* adapté aux réseaux à longue distance
Protocoles PIM-SM et CBT
Attention : protocoles en cours de développement !

Protocole DVMRP

DVMRP = *Distance Vector Multicast Routing Protocol*

- Protocole le plus ancien (1992), implémenté par le programme `mrouted` sous Unix
- DVMRP utilise IGMP pour transporter ses messages
- La version 3 de DVMRP utilise l'algorithme RPM (*Reverse Path Multicast*) avec les greffes
- DVMRP calcule et propage sa propre table de routage *unicast* à l'aide d'une méthode proche de RIP : il ne consulte pas la table « naturelle » du routeur
  $\Rightarrow$ différencie les trafics *unicast* et *multicast* et permet l'utilisation de tunnels (cf MBone)

Protocole DVMRP

DVMRP nécessite deux tables :
La table de routage *unicast* donne le meilleur chemin pour chaque source :

Source	Masque	Routeur précédent	Distance	Temps
128.1.0.0	255.255.0.0	193.51.24.1	5	200
128.2.0.0	255.255.0.0	193.51.24.1	2	100
128.3.0.0	255.255.0.0	193.51.24.5	4	150

La table de diffusion *multicast* donne la structure de l'arbre :

Source	Groupe	Temps	Parent	Fils
128.1.0.0	224.1.1.1	200	i1 prune	i2p i3p
	224.2.2.2	100	i1	i2p i3
	224.3.3.3	150	i1	i2
128.2.0.0	224.1.1.1	150	i2	i1p i3

Protocole PIM-DM

PIM-DM = *Protocol Independent Multicast – Dense Mode*

Le *Independent* signifie que PIM est indépendant du protocole de routage *unicast*.

PIM-DM est comparable à DVMRP

Protocole PIM-DM

Comme DVMRP, PIM-DM utilise :

- un algorithme proche du *Reverse Path Multicast*
- des messages d'élagage (*prune*) ou de greffe (*graft*)

À l'inverse de DVMRP, PIM-DM :

- utilise son propre format de messages (DVMRP utilise IGMP)
- ne construit pas sa table de routage *unicast*, mais utilise celle présente dans le routeur
- envoie les datagrammes sur toutes les interfaces (DVMRP ne l'envoie pas aux interfaces qui ne sont pas le « meilleur chemin ») $\Rightarrow$ raison : indépendant du routage *unicast*

Protocole PIM-SM

PIM-SM = *Protocol Independent Multicast – Sparse Mode*

Rappel : PIM-SM est adapté aux réseaux à longue distance :

- bande passante rare
- membres de groupes nombreux et disséminés

PIM-SM est différent de PIM-DM, mais en reprend certaines caractéristiques :

- il est indépendant du protocole de routage *unicast*
- il utilise le même format de messages

PIM-SM utilise l'encapsulation, l'algorithme d'arbre enraciné à la source, ou l'algorithme d'arbre partagé

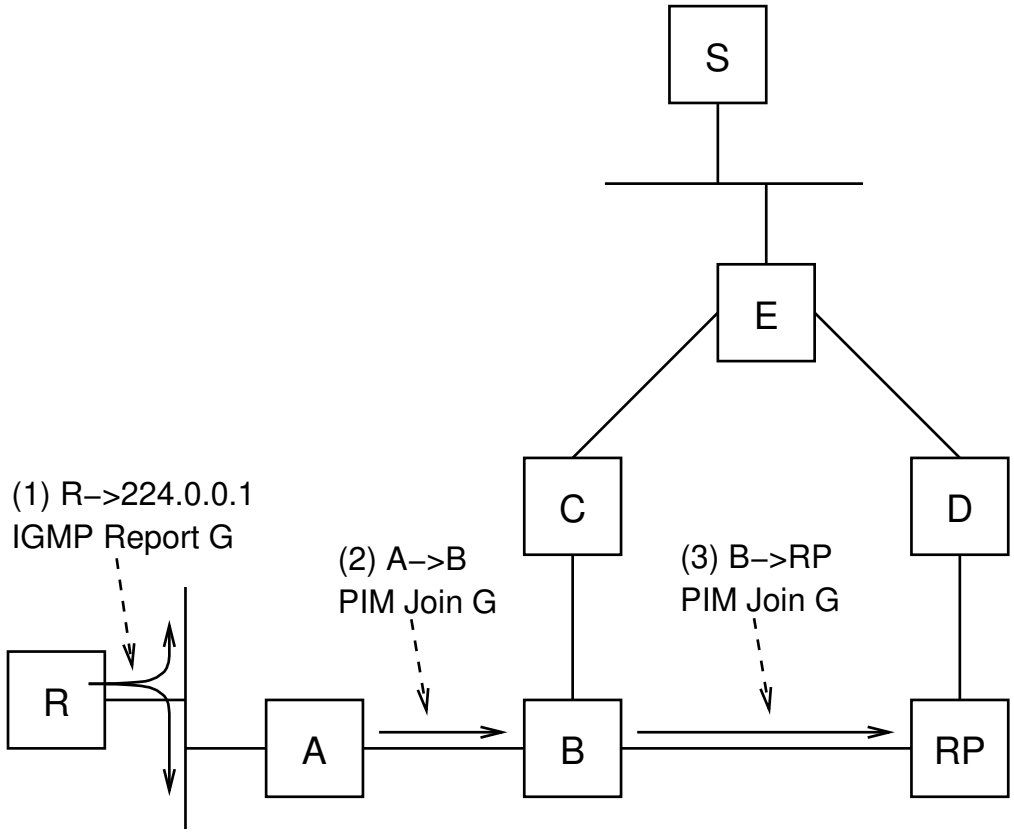
Protocole PIM-SM

Notion de *point de rendez-vous* (RP) :

- routeur qui recueille les demandes de réception et d'émission pour un groupe donné
- il y a un seul et un seul RP par groupe
- un RP peut être le point de rendez-vous de plusieurs groupes

Protocole PIM-SM

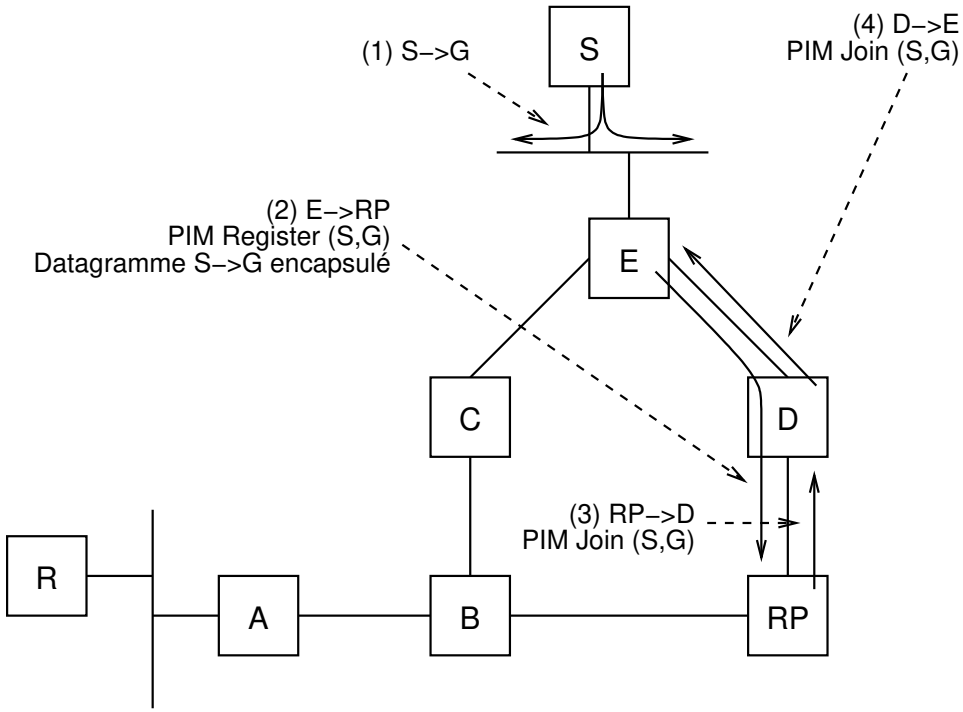
Adhésion d'un récepteur R à un groupe G :



Chaque routeur intermédiaire ajoute le groupe G à sa liste de groupes.

Protocole PIM-SM

Émission vers G par une nouvelle source S :



RP décapsule le datagramme S→G et le propage.
Tant que E n'a pas reçu le message *Join*, E encapsule les datagrammes S→G dans des messages *Register*.
Lorsque l'émetteur est enregistré, il y a un chemin de S vers R (S,E,D,RP,B,A,R)

Protocole PIM-SM

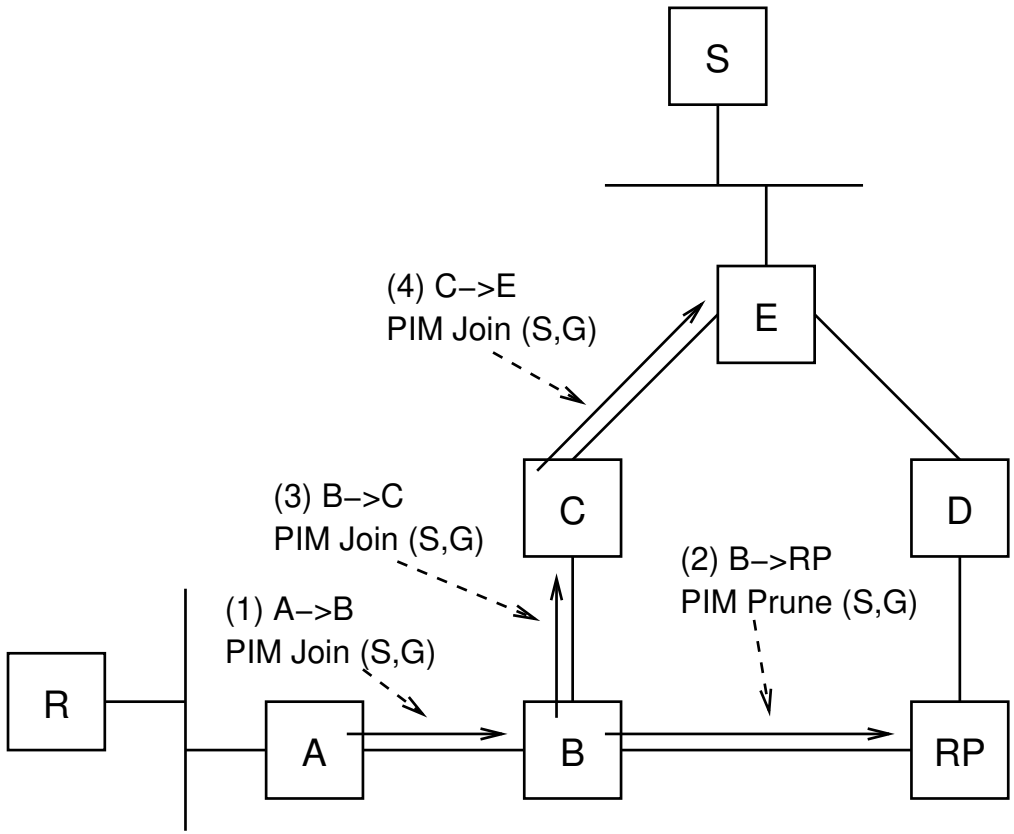
Une fois l'émetteur enregistré :

- soit E continue à utiliser le chemin passant par RP
  $\Rightarrow$ algorithme de l'*arbre partagé par un groupe*

  Mais problèmes :
  - route non optimale
  - congestion près de RP
- soit un routeur de récepteur décide de demander une diffusion par le « meilleur chemin »
  $\Rightarrow$ algorithme de l'*arbre enraciné à la source*
  Le changement est décidé si le trafic le justifie

Protocole PIM-SM

Changement du routage S→G pour R :



Protocole PIM-SM

Il y a un ensemble de RP pour tous les groupes.

Un routeur élu est chargé de :

- surveiller les RP
- constituer l'ensemble de RP
- distribuer périodiquement les adresses des RP

Protocole CBT

CBT = *Core Based Trees*

CBT est comparable à PIM-SM :

- mode *clairsemé*
- indépendant du protocole de routage *unicast*
- apprentissage du noyau
- algorithme *arbre partagé par un groupe*
- un routeur central (*core router*) par groupe

Protocole CBT

En revanche CBT diffère de PIM-SM :

- CBT n'utilise que l'algorithme d'*arbre partagé*
  Il ne peut pas passer à l'algorithme *arbre enraciné à la source*
  Raison : l'arbre enraciné à la source demande trop de ressources dans les routeurs
  $\Rightarrow$ CBT supporte mieux la montée en charge
- le *core router* est toujours la racine de l'arbre d'un groupe
  $\Rightarrow$ le point de rendez-vous de PIM-SM n'est pas toujours la racine
- CBT est symétrique (pas de différence entre nouvel émetteur et nouveau récepteur)

Plan

MBone

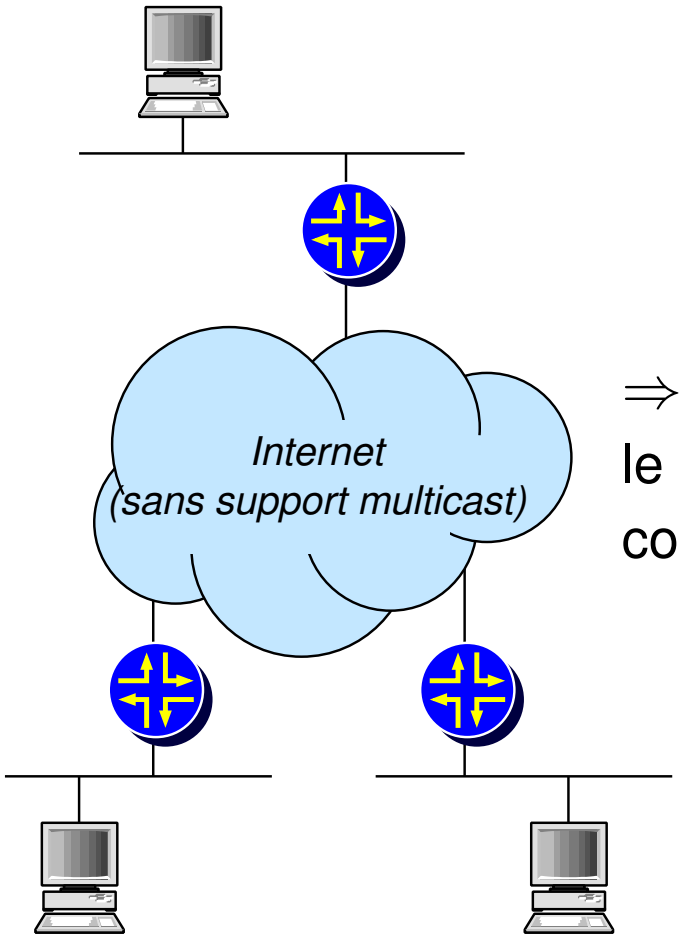
Historique :

- 1992 : création, 40 réseaux dans 4 pays
- fin 1993 : création du FMBone (*French MBone*)
- novembre 1994 : concert des Rolling Stones
- novembre 1996 : l'élagage devient obligatoire
- mars 1997 : 3400 réseaux dans 25 pays
- 2008 : fin du MBone

MBone = support expérimental de la recherche sur le *multicast*
$\Rightarrow$ effectivement utilisé par tous les publics

MBone

Support de la diffusion dans un environnement qui ne supporte pas le *multicast* :

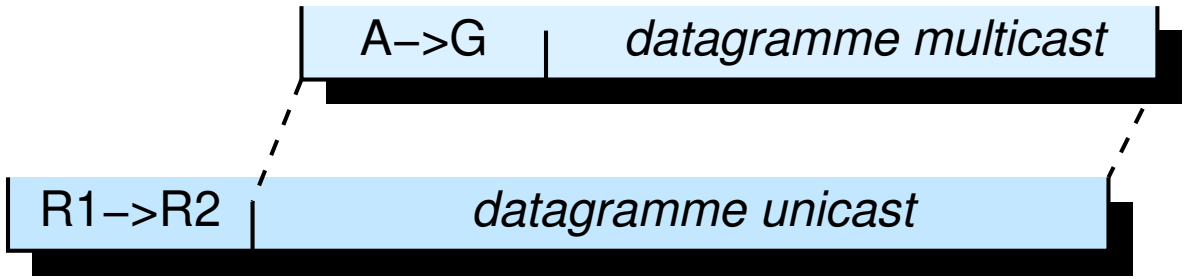


$\Rightarrow$ présence « d'îlots » qui comprennent le *multicast*, séparés par un réseau qui ne comprend pas le *multicast*.

MBone

Lorsque le datagramme IP émis par A traverse le routeur R1, celui-ci l'encapsule dans deux datagrammes IP à destination de R2 et R3.

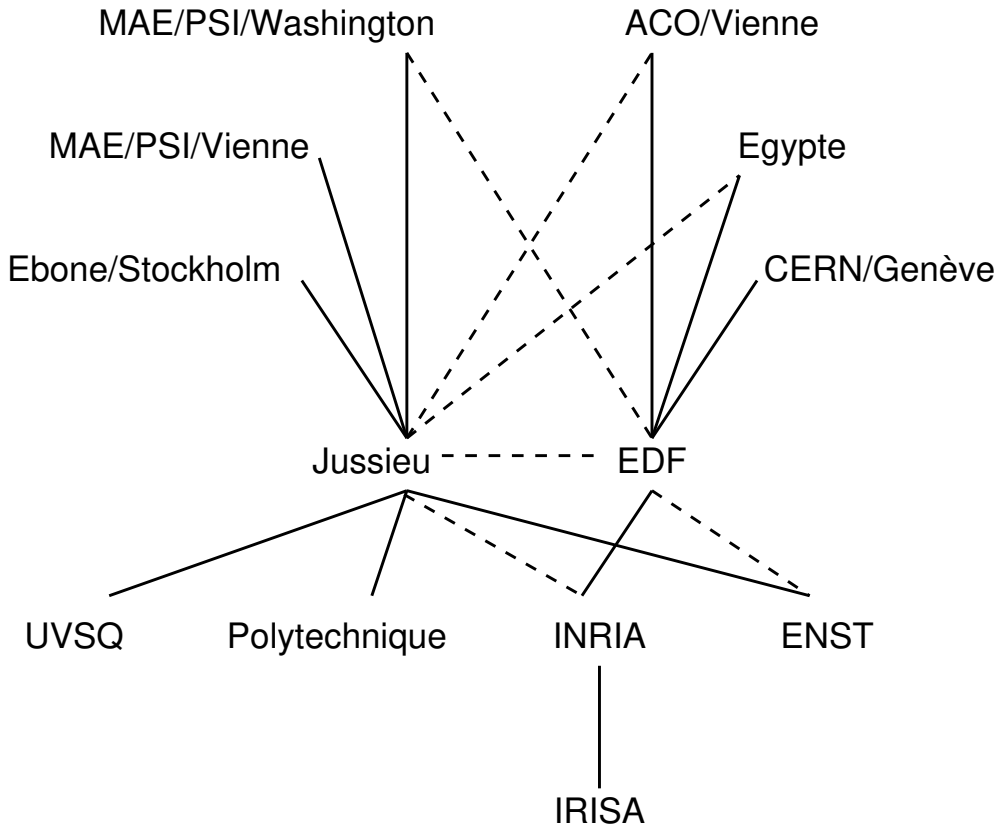
Exemple pour l'encapsulation vers R2 :



$\Rightarrow$ on parle de *tunnels multicast*

MBone

MBone est un réseau virtuel par dessus l'Internet
Exemple (fantaisiste) pour le FMBone :



Les pointillés correspondent aux liaisons (tunnels) de secours.

MBone

Les tunnels sont configurés explicitement par les administrateurs des deux nœuds.

Quatre paramètres pour chaque tunnel :

- adresse IP de l'autre extrémité du tunnel
- « coût » de la liaison : utilisé pour le calcul des chemins
  $\Rightarrow$ intéressant pour avoir des tunnels de secours
- TTL seuil : rejet de tout paquet entrant avec TTL < seuil
  $\Rightarrow$ barrières à la sortie d'un campus, d'un pays, etc.
- bande passante utilisable
  $\Rightarrow$ pour limiter les dégats !

MBone

Le MBone n'est plus utilisé depuis 2008 :

- le *multicast* n'est pas utilisé à grande échelle dans l'Internet
  - les débits dans l'Internet se sont accrus
  - le *streaming* vidéo a remplacé la diffusion d'événements en *multicast*
- le support du *multicast* est assuré en natif par certains réseaux
  - diffusion d'images de disque dur dans des salles de ressources
  - diffusion du flux « télévision » dans les réseaux des opérateurs « grand public » jusqu'à la « box » des clients

Aujourd'hui, le *multicast* est restreint au seul domaine contrôlé par une entité

Plan

Programmation

Programmation *multicast* : simple

Plusieurs additions à l'interface de programmation :

- demander la réception d'un groupe *multicast*
- quitter un groupe *multicast*
- spécifier le TTL des datagrammes *multicast* en émission
- bouclage automatique des datagrammes *multicast* (réception des datagrammes émis localement)

Programmation

Ajouter ou quitter un groupe *multicast* :

```
setsockopt (s, IPPROTO_IP, IP_ADD_MEMBERSHIP,
               &imr, sizeof imr)

setsockopt (s, IPPROTO_IP, IP_DROP_MEMBERSHIP,
               &imr, sizeof imr)
```

Avec :

```
struct ip_mreq {
    struct in_addr imr_multiaddr ;
    struct in_addr imr_interface ;
} ;
struct ip_mreq imr ;
```

Initialisation de la structure :

```
imr.imr_multiaddr.s_addr =
         htonl (inet_addr ("225.1.2.3")) ;
imr.imr_interface.s_addr = htonl (INADDR_ANY) ;
```

Programmation

Spécifier le TTL des datagrammes *multicast* :

```
unsigned char ttl = 1 ;
setsockopt (s, IPPROTO_IP, IP_MULTICAST_TTL,
               &ttl, sizeof ttl)
```

Bouclage automatique des datagrammes *multicast* (pour recevoir les datagrammes envoyés localement) :

```
unsigned char o = 1 ;
setsockopt (s, IPPROTO_IP, IP_MULTICAST_LOOP,
               &o, sizeof o)
```

Programmation

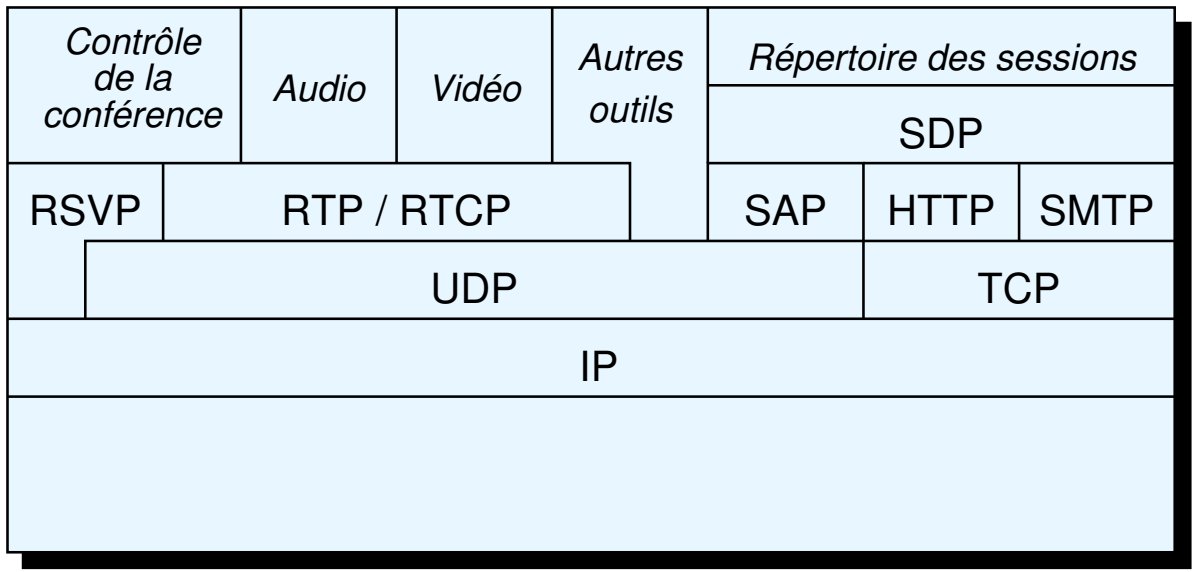
Émission des datagrammes *multicast* : `sendto`
Réception des datagrammes *multicast* : `recvfrom`
$\Rightarrow$ comme n'importe quelle application utilisant UDP

Attention toutefois :

- il faut que l'interface accepte les datagramme *multicast* (configuration avec `ifconfig`)
- il faut savoir où router les datagrammes *multicast* (configuration avec `route`)

Plan

Protocoles applicatifs



SAP + SDP

Problème : comment connaître les groupes *multicast* (audio et vidéo) et les numéros de port pour la conférence XYZ ?

Solutions :

- SAP = *Session Announcement Protocol*
  $\Rightarrow$ propage à intervalles réguliers des annonces
- SDP = *Session Description Protocol*
  $\Rightarrow$ annonces propagées par SAP (ou SMTP, ou HTTP) : coordonnées des conférences

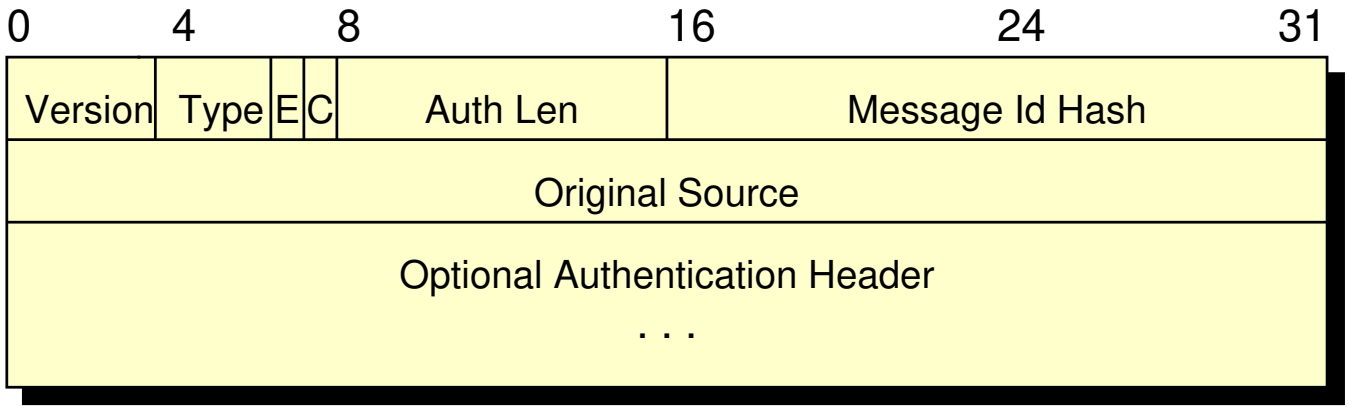
SAP – Session Announcement Protocol

But : mécanisme de propagation d'informations

- à intervalles réguliers (durée dépendante du nombre d'annonces à faire (mais $\leq$ 300 s)
- à une adresse *multicast* précise
  $\Rightarrow$ 224.2.127.254
- à un port UDP précis
  $\Rightarrow$ 9875

SAP – Session Announcement Protocol

Format des paquets SAP version 1 :



Avec :

- *Type* = 0 (annonce de session) ou 1 (suppression de session)
- *E* = 1 si le paquet SDP est crypté
- *C* = 1 si le paquet SDP est compressé
- *Message Id Hash* = clef, unique, pour une source
- *Authentication* : contient la signature du paquet

SAP – Session Announcement Protocol

Notes :

- si le chiffrement est utilisé, des champs sont ajoutés
- l'outil `sdr` implémente la version 0 du protocole :
  Version = *Type* = *E* = *C* = *AuthLen* = 0
  MessageIdHash = *OriginalSource* = 0

SDP – Session Directory Protocol

Les annonces SDP sont diffusées :

- périodiquement avec SAP
- par courrier électronique (SMTP)
- par le Web (HTTP)
- par d'autres canaux...

$\Rightarrow$ SDP est un format d'annonces *en clair*

SDP – Session Directory Protocol

Les annonces SDP sont constituées d'une succession de champs *clef* = *valeur*

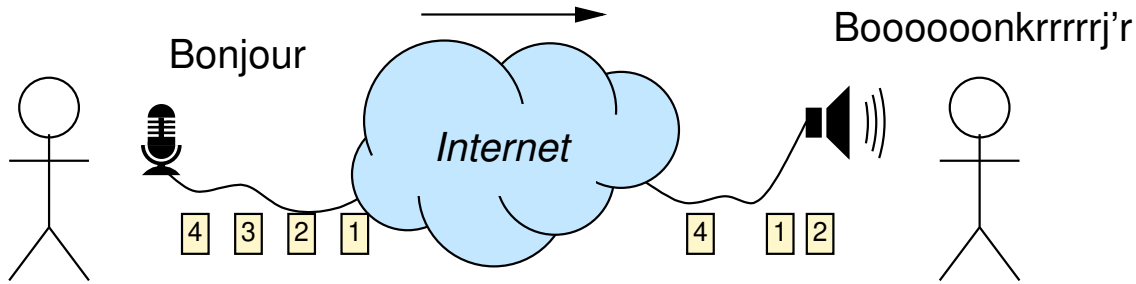
Exemple :

```
v=0
o=pda 2890844526 2890842807 IN IP4 130.79.90.217
s=Mon clip video
e=pda@unistra.fr
c=IN IP4 230.1.2.3/127
t=2873397496 2873404696
m=audio 3456 RTP/AVP 0
m=video 2232 RTP/AVP 31
m=whiteboard 32416 UDP WB
```

RTP – Real Time Protocol

Internet repose sur le principe du *best effort*

Comment contrôler le débit des flux « temps-réel » ?



$\Rightarrow$ problème identique en *unicast* et en *multicast*

Vers un début de solution : RTP

RTP – Real Time Protocol

RTP = *Real-time Transport Protocol*

Principes :

- pas de réémission des paquets perdus !
  si donnée critique (audio) : redondance dans l'application
  (reconstruction du paquet n à partir de $n-1$ et $n+1$)
  si donnée non critique (vidéo), tant pis !
- chaque paquet porte une estampille (date d'émission)
- chaque récepteur informe les participants de la qualité de réception
  $\Rightarrow$ l'émetteur peut agir (ex: réduire la fréquence d'échantillonage)
  $\Rightarrow$ permet aux autres récepteurs de savoir s'ils sont les seuls à avoir un problème ou non

$\Rightarrow$ RTP n'offre pas de garantie de débit

$\Rightarrow$ RTP ne fait qu'offrir aux applications le moyen de reconstituer (partiellement) le flux des données

RTP – Real Time Protocol

Possibilités supplémentaires :

- traduction
  $\Rightarrow$ traduction d'un flot de données
- mixage
  $\Rightarrow$ regrouper plusieurs flots de données en un seul

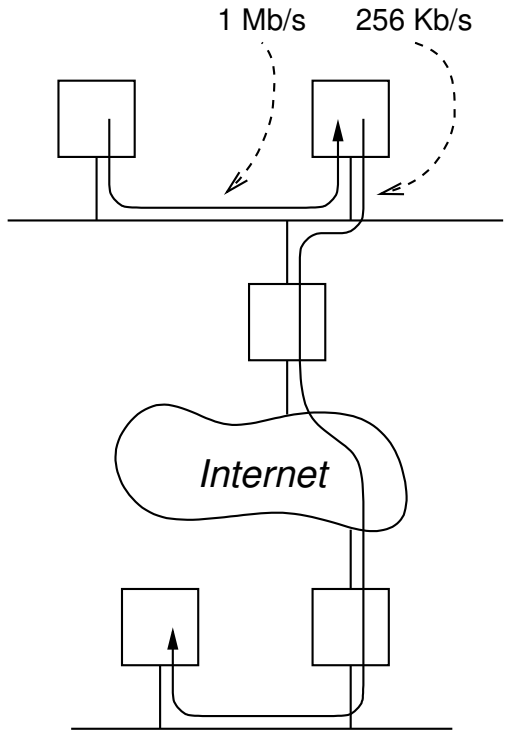
$\Rightarrow$ utilisation essentiellement pour l'adaptation aux réseaux à faible débit

RTP – Real Time Protocol

Traduction : par exemple, diminution du débit utilisé par une application par réduction de la quantité d'information transmise

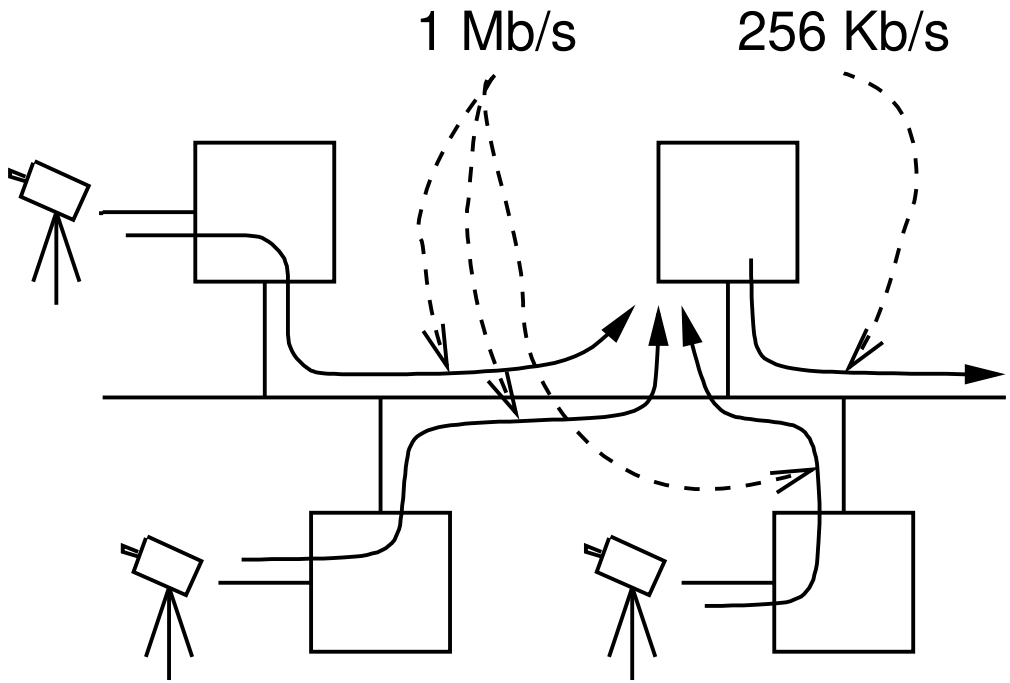
Exemple type : réduction de la qualité d'un signal audio avant la transmission sur un réseau à bas débit

$\Rightarrow$ présence d'une machine *traductrice*



RTP – Real Time Protocol

Mixage : combiner plusieurs flots de données en un seul.
Exemple type : combiner 3 flux vidéo en un seul flux (trois fenêtres de tailles plus petites)



$\Rightarrow$ présence d'une machine *table de mixage*

RTP – Real Time Protocol

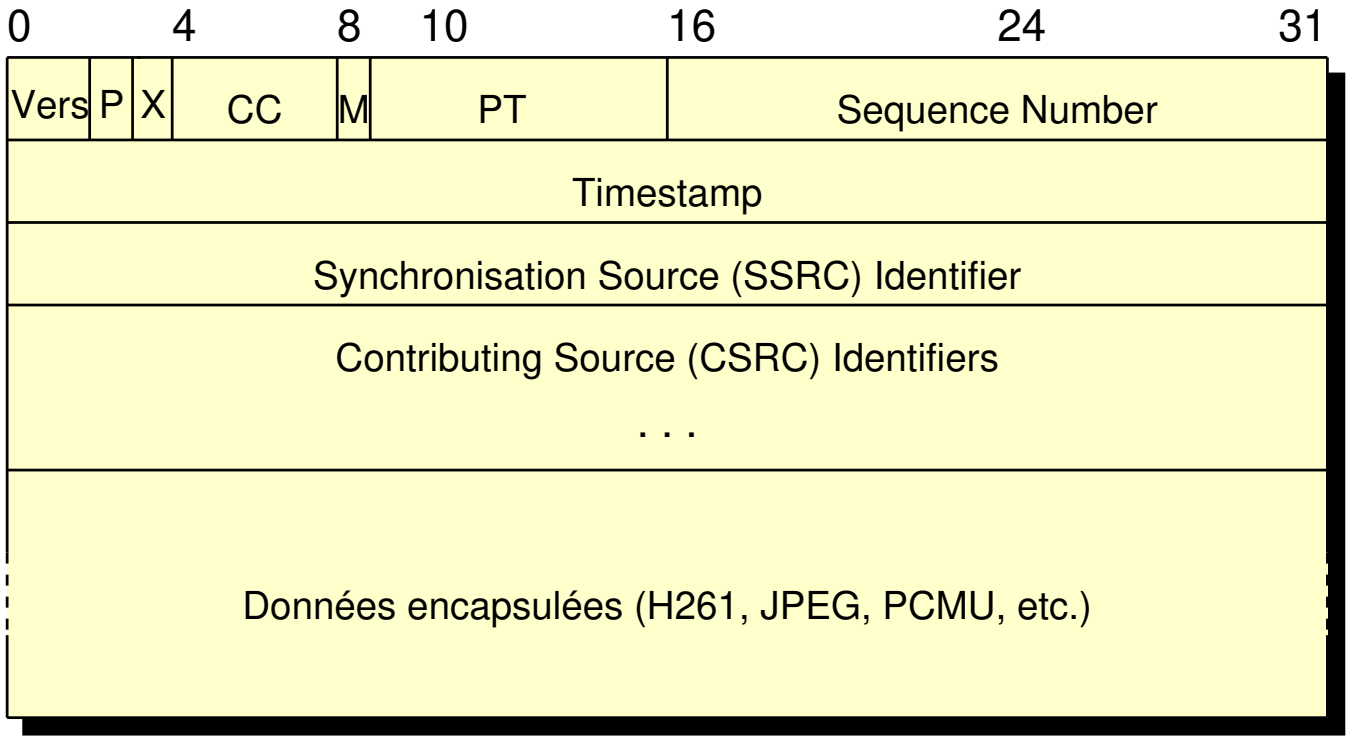
Deux protocoles dans RTP :

- RTP proprement dit : le flux de données
- RTCP (*RTP Control Protocol*) : propagation des informations de gestion du flux
  - annonce par les émetteurs de ce qui a été envoyé (nombre d'octets et de paquets émis, heure)
  - annonce par les récepteurs de ce qu'ils ont reçu (pourcentage de paquets reçus, dernier numéro de paquet reçu, etc.
  - annonce par les émetteurs de ce qui est envoyé (nom de la source, adresse e-mail, etc.)
  - annonce du retrait d'une source
  - messages spécifiques des applications

Si RTP utilise le port UDP n (avec n pair), RTCP utilise le port UDP $n + 1$

RTP – Real Time Protocol

Format des paquets RTP :



RTP – Real Time Protocol

Avec RTP, on sait maintenant comment contrôler le débit des applications « temps-réel » :

- la destination sait reconstruire le flot de données (ordre des paquets, et synchronisation temporelle)
- la source peut adapter son débit à la qualité du réseau (en fonction des rapports des destintaires)

$\Rightarrow$ l'intelligence est aux extrémités
$\Rightarrow$ distribution toujours *best effort*
$\Rightarrow$ aléas de la diffusion sur Internet

Il faudrait un moyen de *contraindre* le réseau à traiter spécialement les paquets « temps réel »
$\Rightarrow$ protocole RSVP

RSVP – Resource ReSerVation Protocol

RSVP indique aux routeurs un flux à traiter spécialement

RSVP permet de réserver des ressources dans les routeurs le long d'un chemin. Par exemple :

- bande passante sur les liens
- file d'attente spécialisée pour un flux
- délai d'acheminement
- taux de perte
- etc.

RSVP ne spécifie pas ce que sont les ressources à réserver
$\Rightarrow$ RSVP est « seulement » un mécanisme pour acheminer les demandes le long d'un chemin
$\Rightarrow$ la QoS demandée est transportée de manière opaque

RSVP – Resource ReSerVation Protocol

RSVP est adapté à la diffusion *multicast*
mais RSVP peut aussi servir pour le cas « *unicast* »

La réservation est effectuée par le(s) récepteur(s)
$\Rightarrow$ tous les récepteurs ne veulent pas (ou ne peuvent pas s'offrir) une bonne qualité de service

RSVP est indépendant du protocole de routage : le routage peut changer, pas la réservation
$\Rightarrow$ RSVP découvre le chemin
$\Rightarrow$ RSVP diffuse périodiquement la réservation
$\Rightarrow$ expiration des réservations dans les routeurs

RSVP – Resource ReSerVation Protocol

Sept types de messages RSVP :

- Messages de chemin :
  - recherche de chemin
  - erreur de chemin (réponse à une demande)
  - libération de chemin
- Messages de réservation :
  - demande de réservation
  - confirmation de réservation
  - erreur de réservation (réponse à une demande)
  - libération de réservation

RSVP – Resource ReSerVation Protocol

Principes :

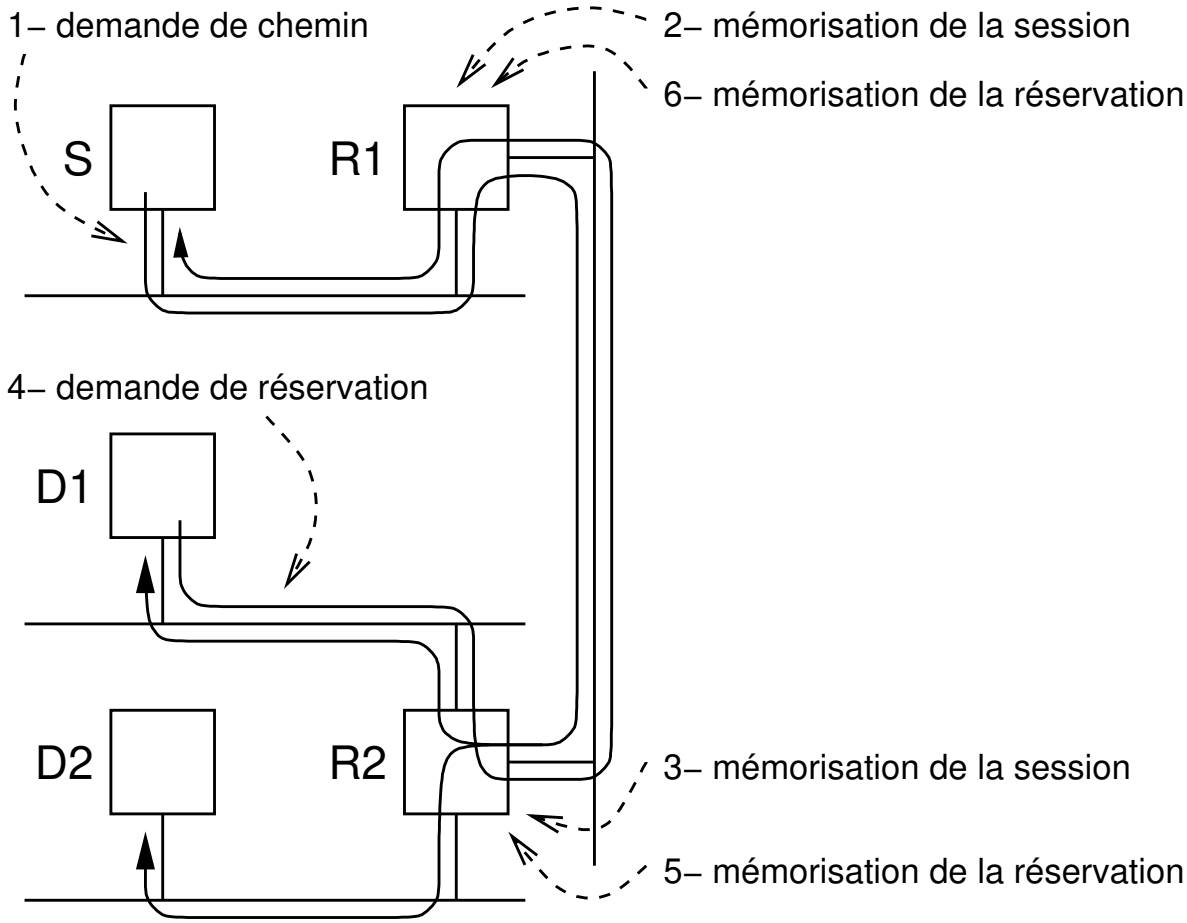
1. la source émet périodiquement des messages RSVP de recherche de chemin à l'adresse du groupe
2. une application destinataire reçoit un tel message
3. l'application destinataire envoie un message RSVP de réservation
4. la source reçoit ce message

Observations :

- la source attend l'étape 4 pour envoyer les données, ou
- tout ceci se fait en parallèle avec le flux de données.

RSVP – Resource ReSerVation Protocol

Exemple : demande de réservation par D1



Note : si D2 demande une réservation, R2 fusionne les demandes.

RSVP – Resource ReSerVation Protocol

Identification d'une session :

- adresse IP de destination
- numéro de protocole (TCP ou UDP ou...)
- numéro de port

$\Rightarrow$ utilisé comme une clef pour retrouver l'état associé à un chemin dans un routeur

$\Rightarrow$ obligatoire dans tous les messages RSVP

RSVP – Resource ReSerVation Protocol

Identification d'un flux à réserver :

- sous-ensemble des datagrammes d'une session
  $\Rightarrow$ notion de filtre :
  - une adresse source (IP + port) précise
  - n'importe quelle adresse source
- partage ou non de la ressource entre les sources
- QoS demandée (non encore normalisée)

sélection des sources	**partage de la ressource**	
	non	**oui**
explicite	*fixed-filter*	*shared-explicit*
n'importe	—	*wildcard-filter*

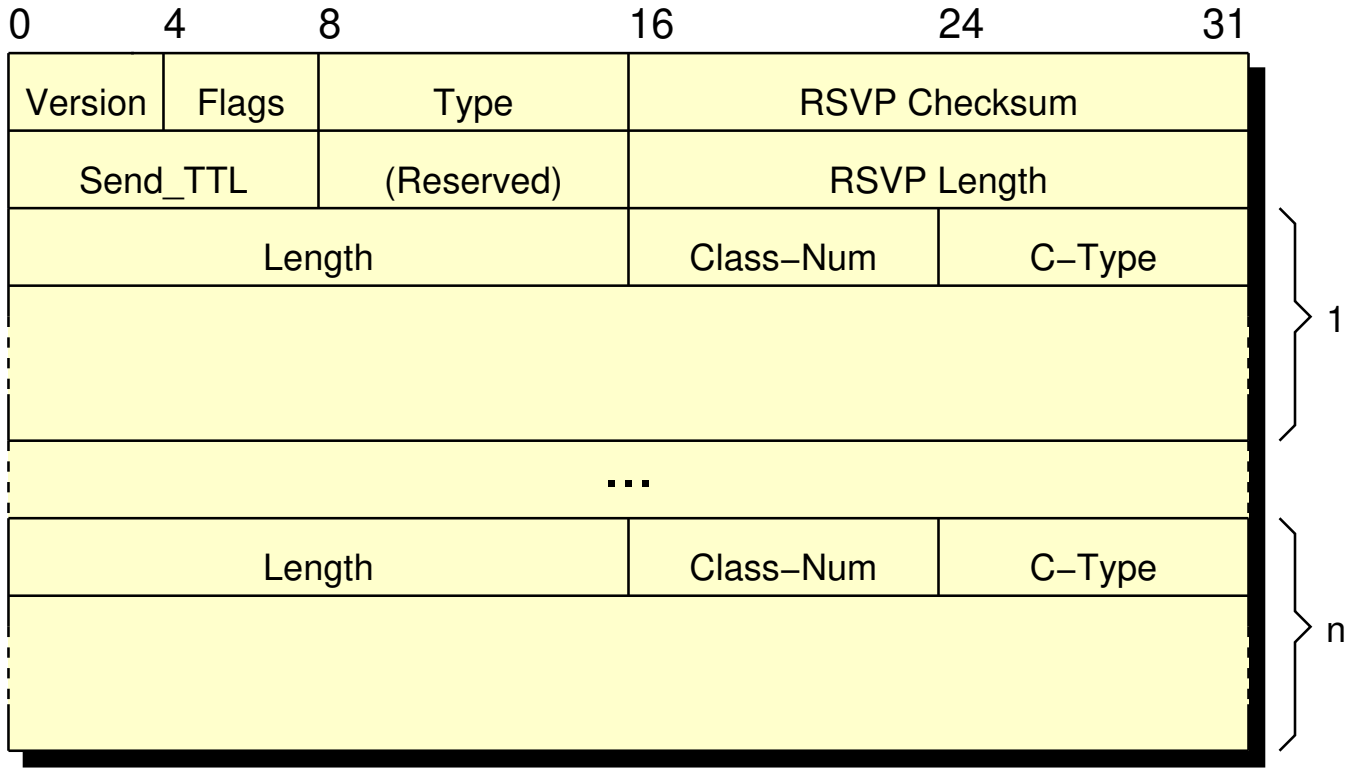
RSVP – Resource ReSerVation Protocol

Exemples :

- *fixed-filter* :
  - application point à point (*unicast*)
  - retransmission d'une conférence (source unique)
  - télésurveillance (multiples sources visionnées simultanément)
  - diffusion d'émission de télévision
- *shared-explicit* et *wildcard-filter* :
  - audio-conférence, où tout le monde ne parle pas en même temps

RSVP – Resource ReSerVation Protocol

Format des messages RSVP :



Principe : une en-tête commune, plus des objets (1 à *n*) ayant la même structure.

Note : les messages RSVP sont encapsulés directement dans des datagrammes IP

RSVP – Resource ReSerVation Protocol

Quelques objets RSVP :

- *SESSION* : identification de session
  $\Rightarrow$ adresse IP destination + protocole + port
- *RSVP_HOP* : adresse d'un routeur intermédiaire
  $\Rightarrow$ adresse IP + interface
- *STYLE* : style de réservation
  $\Rightarrow$ *fixed-filter*, *shared-explicit* ou *wildcard-filter*
- *TIME_VALUE* : période de rafraichissement
  $\Rightarrow$ temps en millisecondes
- *FILTER_SPEC* : datagrammes à traiter spécialement
  $\Rightarrow$ adresse source (IP + port)
- *SENDER_TEMPLATE* : identification de l'émetteur
  $\Rightarrow$ adresse source (IP + port)
- *SENDER_TSPEC* : caractéristiques du flux
  $\Rightarrow$ non encore défini

RSVP – Resource ReSerVation Protocol

Quelques exemples de messages RSVP :

- demande de chemin :
  - en-tête RSVP
  - objet *SESSION*
  - objet *RSVP_HOP* (changé par chaque routeur)
  - objet *TIME_VALUES*
  - objet *SENDER_TEMPLATE*
  - objet *SENDER_TSPEC*
- demande de réservation :
  - en-tête RSVP
  - objet *SESSION*
  - objet *RSVP_HOP* (changé par chaque routeur)
  - objet *STYLE* (ex : *fixed-filter*)
  - objet *FLOW_SPEC*
  - objet *FILTER_SPEC*
  - objet *FILTER_SPEC*

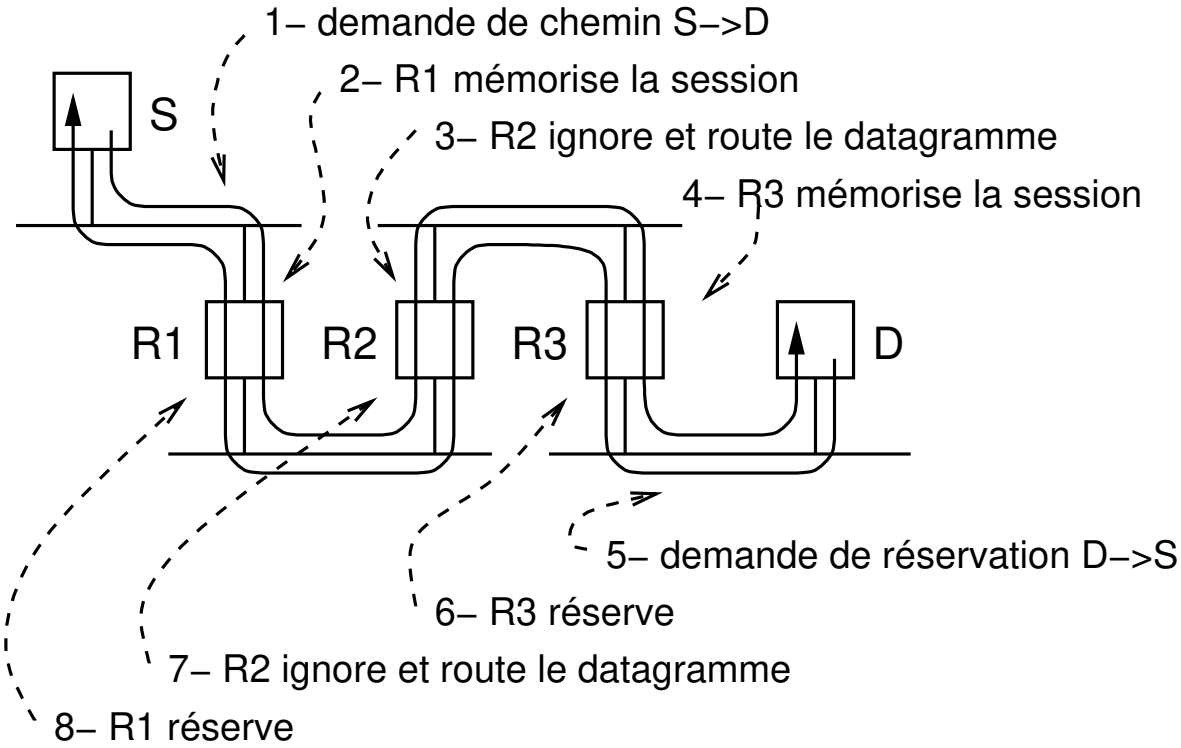
RSVP – Resource ReSerVation Protocol

Quelques exemples de messages RSVP (suite) :

- libération de chemin
  - en-tête RSVP
  - objet *SESSION*
  - objet *RSVP_HOP* (changé par chaque routeur)
  - objet *SENDER_TEMPLATE*
  - objet *SENDER_TSPEC*

RSVP – Resource ReSerVation Protocol

Déploiement de RSVP : pendant encore un certain temps, il y aura des routeurs qui ne comprendront pas RSVP (ex: ici R2) :



Ici, R2 route les datagrammes sans les consulter

$\Rightarrow$ R2 ne réserve pas de ressource

$\Rightarrow$ dégradation de la QoS

Note : RSVP est encore en phase de définition...

Plan

Conclusion

Depuis 1992 : évolution majeure de l'Internet

Plusieurs évolutions :

- définition de la classe D
  notion de groupes d'abonnés
- développement d'algorithmes de diffusion
  expérimentation avec le Mbone
- développement d'applications
  audio, vidéo, tableau blanc, édition de texte, etc.
- apparition de flux « temps réel »
  protocoles RTP et RSVP