R316-ROM
Ingénierie de la téléphonie sur IP

Sami Evangelista
IUT de Villetaneuse
Département Réseaux et Télécommunications
2024–2025

http://www.lipn.univ-paris13.fr/~evangelista/cours/R316-ROM

Plan

Contenu du module

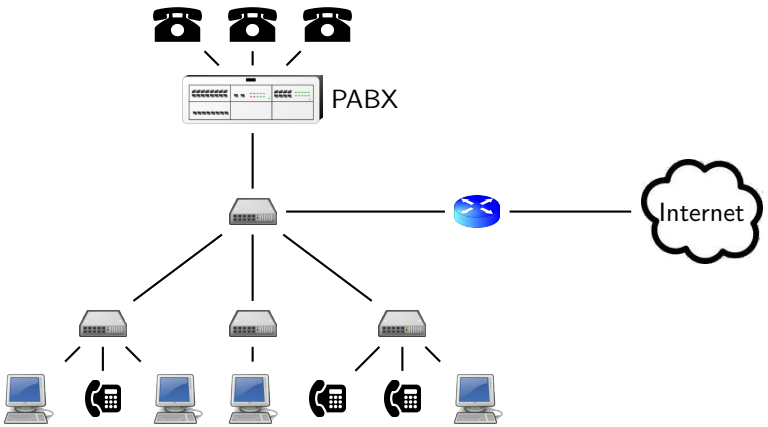
R316-ROM — Ingénierie de la téléphonie sur IP

- Objectif : étude des protocoles et mécanismes de qualité de service utilisés en téléphonie sur IP
- Organisation :
  - 6h de Cours/TD
  - $5 \times$ 3h de TP
  - $1 \times$ 2h de contrôle
- Évaluation : TP notés + contrôle

La ToIP

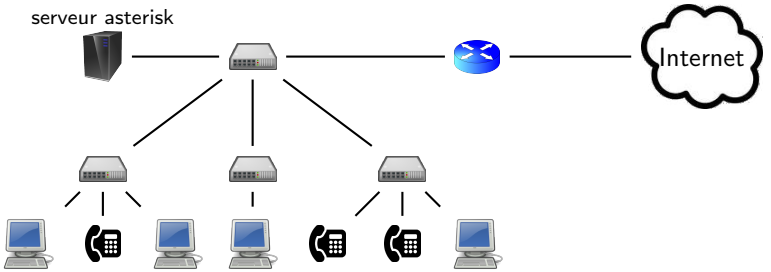
- ToIP = Telephony on IP
- Techniques de téléphonie qui utilisent uniquement le réseau IP (plus de réseau téléphonique).
- Équipements téléphoniques avec une pile TCP/IP.
- Avantages :
  - baisse des coûts (abonnements et communication)
  - simplification du câblage (un seul réseau avec un seul type de câble)
  - flexibilité :
    - mobilité des utilisateurs (grâce aux registres, voir plus loin)
    - travail à distance
    - facilité d'ajout/suppressions de lignes
- Inconvénients :
  - problèmes réseau : latence, baisse de la qualité de la communication
  - pas de service pendant une coupure électrique

Exemple d'architecture hybride



- utilisation de téléphones analogiques (☎) et IP (📞)
- Tous les appels (entrants, sortants et en interne) transitent par le PABX.
- Rôles du PABX :
  - proxy et registrar SIP (voir plus loin)
  - passerelle VoIP (traduction flux analogiques ↔ flux numériques)

Exemple d'architecture 100% IP



- plus de téléphones analogiques
- asterisk = logiciel libre, serveur SIP sous Linux
- Le serveur asterisk remplace le PABX.

Plan

L'architecture SIP

Participants :

- **UA** (User Agent) : logiciel ou équipement de téléphonie
- **Registrars** : les serveurs d'enregistrement des UA
- **Proxys SIP** : intermédiaires entre les UA
- **Serveurs DNS** : renseignent sur les proxys SIP de leur domaine

Les UA et l'adressage SIP

- UA = n'importe quel logiciel (linphone, ekiga, ...) ou téléphone IP qui comprend le protocole SIP.
- Pas skype, par exemple, car il utilise un protocole propriétaire.
- Un UA est identifié par son **URI** (Uniform Ressource Identifier).
- forme générale d'une URI :

  `sip:identifiant[:motdepasse]@où[:port][?paramètres]`
- entre crochets : ce qui est optionnel
- Donc dans la forme la plus simple on a :

  `sip:identifiant@où`
- où peut être :
  - l'IP ou le nom de l'UA
  - l'IP ou le nom de son proxy SIP
  - le nom du domaine de l'UA

Les registrars (serveurs d'enregistrement)

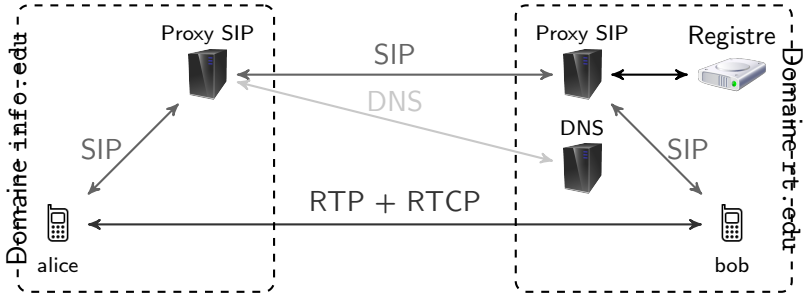
- Les utilisateurs peuvent se connecter avec leurs UA sur différentes machines.
- Problème : comment retrouver son IP pour lui transmettre des appels ?
- La **registrar** maintient une base de données des localisations dans un **registre** sous forme de couples

  (identifiant, IP + port)
- registrar = n'importe quel serveur qui traite les requêtes SIP **REGISTER** des UA
- Un registrar stocke uniquement les localisations de son domaine.

Les proxys SIP

- proxy SIP : intermédiaires entre deux UA qui ne connaissent pas leurs localisations respectives
- Le proxy SIP consulte le registre de son domaine pour récuperer la localisation d'un UA appelé.
- Généralement, le proxy et le registrar sont une seule et même entité.

Scénario d'appel entre deux UA



`alice@info.edu` passe un appel à `bob@rt.edu`

1. Établissement de la session
   - utilisation de SIP entre les proxys et les UA
   - Le proxy SIP de `info.edu` interroge le DNS de `rt.edu` pour avoir l'IP du proxy SIP de `rt.edu`.
   - Le proxy SIP de `rt.edu` interroge son registre pour connaître la(es) localisation(s) actuelle(s) de bob.
2. Lors de l'appel
   - transport du flux audio entre les deux UA avec RTP+RTCP
3. Libération de la session
   - utilisation de SIP entre les proxys et les UA

Utilisation de DNS pour localiser un proxy SIP

- Dans le scénario précédent, on a vu qu'un proxy interroge un serveur DNS quand il doit relayer un message SIP au proxy d'un domaine, disons `rt.edu`.
- Dans ce cas, le DNS du domaine `rt.edu` doit définir (au moins) un enregistrement SRV ayant la forme suivante :

  `_sip._prot SRV prio poids port nom-du-proxy-sip`

  avec :
  - `prot` = protocole de transport utilisé par le proxy
  - `prio` = classe de priorité du serveur
  - `poids` = poids relatif du serveur dans sa classe de priorité
  - `port` = numéro de port sur lequel il faut contacter le proxy sip
- Quand un proxy voudra communiquer en UDP avec un proxy de `rt.edu`, il demandera les enregistrements SRV concernant `_sip._udp.rt.edu`.

Utilisation de DNS pour localiser un proxy SIP : exemple

Fichier de la zone `rt.edu` :

```
;;              priorité  poids  port  nom-du-proxy
_sip._udp  SRV  1         2      5060  sip1
_sip._udp  SRV  1         1      5060  sip2
_sip._udp  SRV  2         1      6589  sip3

;;  adresses IP des trois serveurs
sip1  A  1.2.3.1
sip2  A  1.2.3.2
sip3  A  1.2.3.3
```

- Le domaine `rt.edu` a trois proxys SIP utilisant UDP : `sip1`, `sip2` sur le port 5060 et `sip3` sur le port 6589.
- `sip1` et `sip2` ont une priorité de 1 : c'est eux qu'il faut tenter de contacter en premier. Si aucun des deux ne répond, on contacte `sip3`.
- `sip1` et `sip2` ont la même priorité mais pas le même poids. Les poids relatifs indiquent que dans 2 cas sur 3 il faut contacter `sip1` et dans 1 cas sur 3 il faut contacter `sip2`.

Remarque sur les enregistrements DNS de type SRV

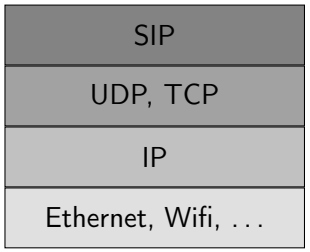
- Dans l'exemple précédent on a vu qu'un enregistrement SRV permet de découvrir l'identité d'un serveur SIP.
- Ce type d'enregistrement a été créé pour découvrir n'importe quel type de service proposé sur un domaine.
- Par exemple, si l'administrateur du domaine veut rendre visibles deux serveurs HTTP et FTP, il peut rajouter dans le fichier de zone :

```
;;              priorité  poids  port  nom de la machine
_http._tcp  SRV  1         1      80    www
_ftp._tcp   SRV  1         1      21    ftp
```

Plan

Présentation de SIP

- SIP = Session Initiation Protocol
- première version de SIP dans la RFC 3261 (juin 2002)
- protocole de signalisation pour l'établissement/libération de sessions interactives entre utilisateurs
- utilisé en téléphonie et plus généralement pour les communications multimédias
- port par défaut = 5060
  (utilisé par tous les équipements SIP : UA, proxys, registrars)
- pile protocolaire :



- (En général c'est plutôt de l'UDP au niveau transport.)

Structure des messages SIP

- structure similaire aux messages HTTP
- Requêtes et réponses ont le même format :
  - 1 ligne avec
    - pour une requête : le type de la requête
    - pour une réponse : le code d'état
  - N ligne(s) d'en-tête avec différents champs
  - 1 ligne vide qui marque la fin de l'en-tête
  - un corps

Types de requête

Les types les plus utilisés :

- **INVITE** = demande d'initiation d'une session
- **ACK** = confirmation des paramètres d'une session
- **REGISTER** = enregistrement de sa localisation auprès d'un registrar
- **BYE** = fin d'une session ($\Leftrightarrow$ un des UA raccroche pendant l'appel)
- **CANCEL** = annulation d'une session ($\Leftrightarrow$ l'appelant raccroche avant que l'appelé ne décroche)

Codes d'état des réponses

Les codes les plus utilisés :

- 1xx = Messages d'information
  - 100 = trying
  - 180 = ringing
- 200 = OK
- 3xx = Messages de redirection
  - 301 = moved permanently (identifiant demandé n'est plus dispo.)
  - 302 = moved temporarily
- 4xx = Erreur client
  - 401 = authorisation requise (p.ex., un registrar refuse l'enregistrement)
  - 404 = utilisateur inexistant
  - 486 = utilisateur occupé
- 5xx = Erreur serveur
  - 500 = erreur interne
  - 503 = service non disponible (p.ex., serveur surchargé)

Les champs d'en-tête

- Chaque champ de l'en-tête a la forme `Champ: Valeur`.
- Champs principaux pour les messages INVITE :
  - **From** — URI de l'appelant
  - **To** — URI de l'appelé
  - **Call-Id** — id. d'un appel
  - **User-Agent** — type de l'UA
  - **Via** — liste des UA/Proxys par lequel le message est passé (IP + ports)
    ⇒ **La réponse au message suivra ce même chemin.**
  - **Content-Type** — type MIME du contenu
  - **Max-Forwards** — nombre max. de proxys par lesquels un message peut transiter (⇒ permet d'éviter les boucles)

Le corps du message

- Le corps du message est optionnel.
- Il contient le descriptif des paramètres de la session :
  - **IP + port à utiliser pour le flux RTP**
  - medias souhaités pour la communication
  - codecs disponibles
  - paramètres des codecs

  ...
- On le trouve principalement dans
  - un message INVITE (param. fournis par l'appelant)
  - un message OK envoyé en réponse à un INVITE (param. fournis par l'appelé)
- Il peut être au format HTML ou SDP (Session Description Protocol).

Exemple de message INVITE


```
1  INVITE sip:411@ideasip.com SIP/2.0
2  CSeq: 1 INVITE
3  Via: SIP/2.0/UDP 194.254.173.6:5060
4  Via: SIP/2.0/UDP 157.12.54.87:5060
5  Via: SIP/2.0/UDP 54.21.4.7:5060
6  User-Agent: Ekiga/4.0.1
7  From: <sip:sami@194.254.173.6>
8  Call-ID: 54d5b754-cdbe-e611-885f
9  To: <sip:411@ideasip.com>
10 Content-Length: 458
11 Content-Type: application/sdp
12 Max-Forwards: 70
13
14 v=0
15 o=- 1481542778 1 IN IP4 194.254.173.6
16 s=Ekiga/4.0.1
17 c=IN IP4 194.254.173.6
18 t=0 0
19 m=audio 54678 RTP/AVP 116 0 8 101
20 a=sendrecv
21 a=rtpmap:116 Speex/16000/1
22 a=rtpmap:8 PCMA/8000/1
23 a=rtpmap:101 telephone-event/8000
24 a=fmtp:101 0-16,32,36
25 ...
```

En-tête (lignes 1 à 12)

ligne 1 — ligne de requête avec type de la requête, URI de l'appelé et num. de version SIP

lignes 3–5 — le INVITE a été émis par l'UA 194.254.173.6:5060. Il est ensuite passé par les proxys 157.12.54.87:5060 et 54.21.4.7:5060

lignes 7–9 — URI de l'appelant et de l'appelé

ligne 11 — format du contenu du message = SDP

Corps (lignes 14 et suivantes)

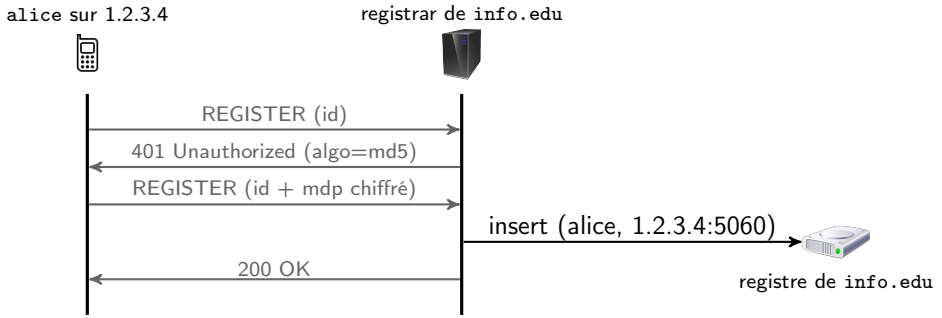
ligne 17 — IP à utiliser pour le flux RTP = 194.254.173.6

ligne 19 — port UDP à utiliser pour le flux RTP = 54678

ligne 20 et suivantes — autres info. RTP (media utilisés, codecs, ...)

Scénario SIP 1 — Enregistrement d'un UA

- `alice@info.edu` s'enregistre auprès de son registrar
- Quand a lieu l'enregistrement ? À l'ouverture du softphone, au branchement du téléphone IP, ...

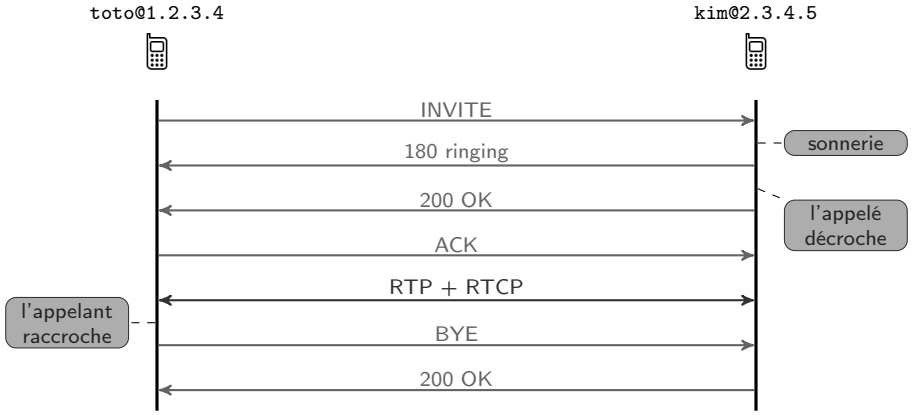


- Un 1er message REGISTER contient l'identifiant.
- Le serveur refuse et envoie un algo de chiffrement (md5 ici).
- Un 2ème message REGISTER contient identifiant + mot de passe crypté.

Scénario SIP 2 — Appel direct

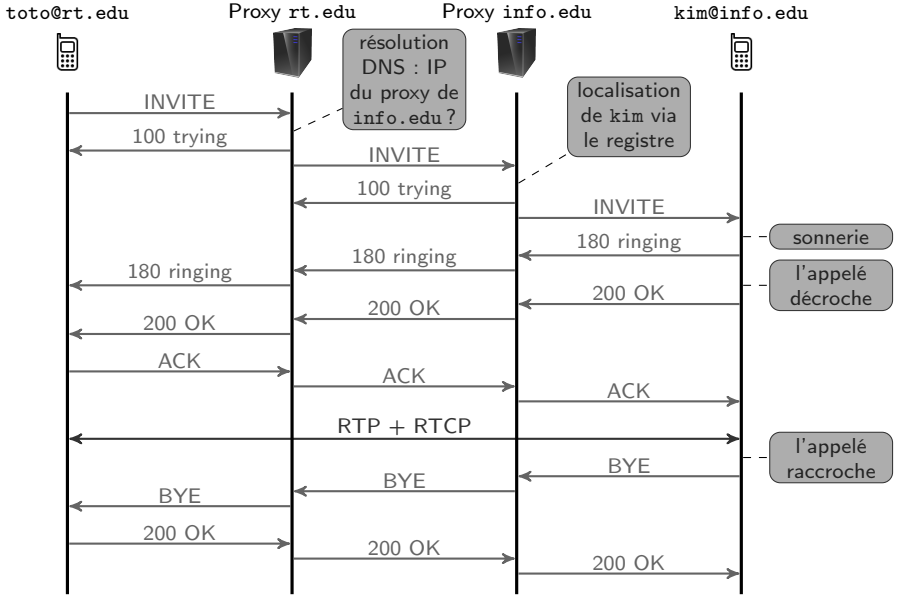
- `toto@1.2.3.4` appelle `kim@2.3.4.5`

  (Pas nécessaire de passer par un proxy si on a l'IP de l'UA destinataire.)



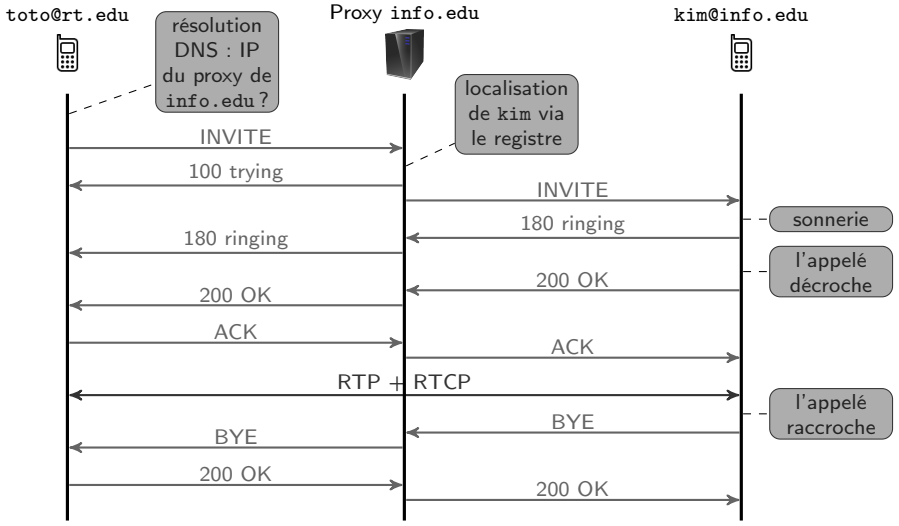
Scénario SIP 3 — Appel passant par deux proxys

- `toto@rt.edu` appelle `kim@info.edu` en passant par son proxy



Scénario SIP 4 — Appel passant par un proxy

- `toto@rt.edu` appelle `kim@info.edu` en contactant directement le proxy de `info.edu`



Plan

Rappels sur NAT

- NAT = Network Address Translation
- technique utilisée pour résoudre la pénurie d'adresses IP en séparant des réseaux privés (p.ex., à l'IUT) et le réseau public
- sur un réseau privé : IP privées inutiles sur le réseau public
- Plages des adresses privées :
  - 10.0.0.0/8
  - 172.16.0.0/12
  - 192.168.0.0/16
- Un réseau privé est derrière une passerelle NAT (généralement la passerelle par défaut du réseau) qui fait la translation entre les deux types d'adresse.
- L'interface de la passerelle qui la relie à l'extérieur a une IP publique.

Fonctionnement de la passerelle NAT (1/2)

1 Pour les paquets sortants, la passerelle :
  1.1 modifie l'IP et le port source (privés) par son IP publique et par un nouveau numéro de port qu'elle choisit (p.ex., aléatoirement dans une plage donnée) ;
  2.2 et mémorise l'association (IP + port privés, IP + port publics) dans une table de correspondance.

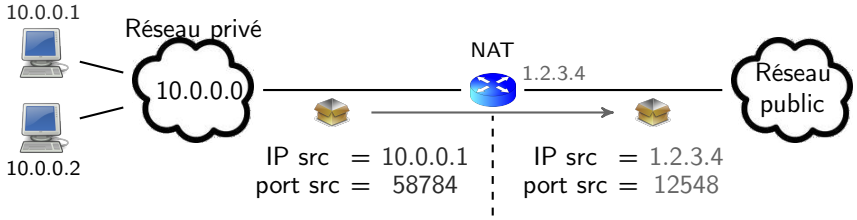


Table de correspondance

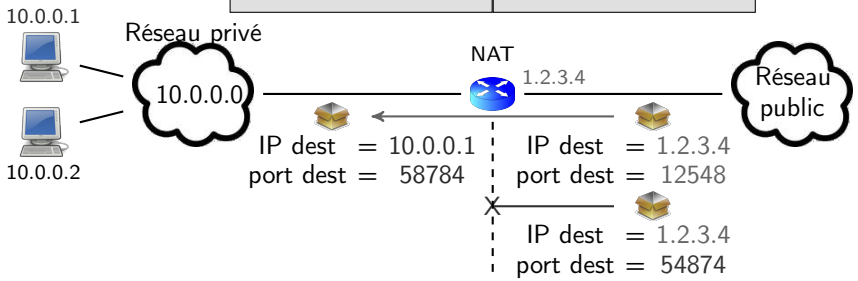
IP + port privés	IP + port publics
10.0.0.1:58784	1.2.3.4:12548

Fonctionnement de la passerelle NAT (2/2)

2 Pour les paquets entrants, la passerelle cherche dans sa table.
  2.1 si association trouvée : IP et port destination modifiés
  2.2 sinon : paquet jeté

Table de correspondance

IP + port privés	IP + port publics
10.0.0.1:58784	1.2.3.4:12548

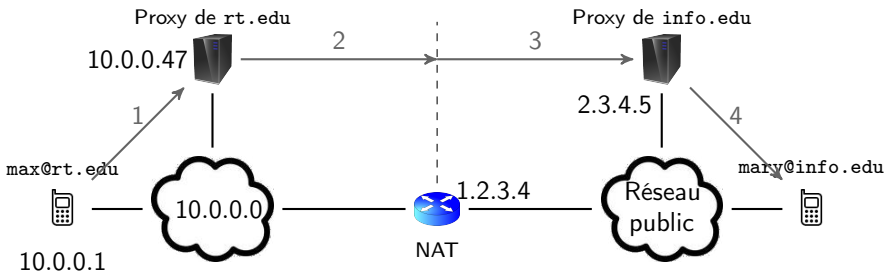


Remarques sur le fonctionnement de la passerelle NAT

- Certaines passerelles NAT ne font pas de translation de port : elles changent uniquement l'adresse IP.
- La sortie d'un paquet ouvre le port choisi par la passerelle.
- Les lignes de la table ont une durée de vie limitée (quelques minutes).
- La passerelle joue le rôle de filtre : elle
  - laisse entrer les paquets envoyés en réponse à des paquets sortis du réseau ;
  - et bloque les autres.
- Si un serveur se trouve sur le réseau privé et doit pouvoir être accessible depuis l'extérieur il faut rajouter statiquement une ligne dans la table.
  - Ex : l'administrateur ajoute (priv. = 10.0.0.2:80, pub. = 1.2.3.4:80) pour rendre le serveur web sur 10.0.0.2 accessible depuis l'extérieur.
    opération possible sous Linux avec l'outil iptables (voir TP)

Problème d'utilisation du NAT avec SIP (1/2)

`max.rt.edu` appelle `mary.info.edu`.



- La passerelle ne modifie pas l'en-tête et le contenu SIP !
- Dans le message SIP Invite 3 :
  - source IP et UDP = 1.2.3.4:48789 (port choisi par la passerelle NAT)
  - Dans l'en-tête SIP : `Via: 10.0.0.1:5060` et `Via: 10.0.0.47:5060`
  - Dans le corps SIP : IP + port pour le flux RTP = 10.0.0.1:8420

⇒ La réponse au INVITE ne pourra pas parvenir au proxy de `rt.edu`.

⇒ Même si la réponse au INVITE arrivait à max, le flux RTP envoyé par mary serait envoyé sur 10.0.0.1:8420

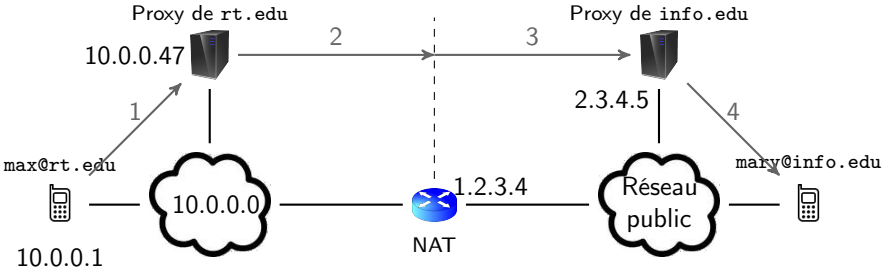
Problème d'utilisation du NAT avec SIP (2/2)

- L'exemple précédent montre qu'on doit résoudre deux problèmes :
  1. La réponse au INVITE doit arriver au proxy de `rt.edu`.
  2. Le flux RTP doit pouvoir arriver à max.
- Pour le problème 1 : utilisation de l'option received/rport.
- Pour le problème 2 : nombreuses solutions. Nous allons étudier STUN.

L'option received/rport

- Quand un proxy SIP va recevoir un INVITE, il :
  1. compare l'IP+port source à IP+port du dernier champ Via
  2. si $\neq$ alors il y a un NAT entre les 2 $\Rightarrow$ il rajoute à ce dernier champ Via
     - l'option `received=<ip-source>`
     - l'option `rport=<port-source>`
- Ce sont les valeurs des options `received` et `rport` qui seront utilisées pour la réponse au INVITE.

L'option received/rport — Exemple



	IP + port sources	Champs Via (en-tête SIP)
1	10.0.0.1:5060	Via : 10.0.0.1:5060
2	10.0.0.47:5060	Via : 10.0.0.1:5060 Via : 10.0.0.47:5060
3	1.2.3.4:48789	Via : 10.0.0.1:5060 Via : 10.0.0.47:5060
4	2.3.4.5:5060	Via : 10.0.0.1:5060 Via : 10.0.0.47:5060 ;received=1.2.3.4 ;rport=48789 Via : 2.3.4.5:5060

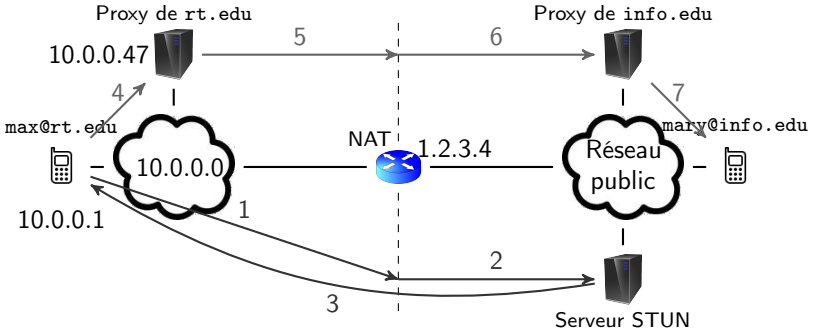
Le proxy de `info.edu` pourra envoyer sa réponse à 1.2.3.4:48789.

STUN

- STUN = Simple Traversal of UDP through NAT
- Un serveur STUN permet au client de découvrir son IP et son port publics.
- On trouve plein de serveurs STUN libres d'utilisation sur Internet.
- Fonctionnement (simplifié) :
  1. Le client envoie un paquet UDP au serveur STUN.
  2. Le serveur STUN répond en plaçant dans sa réponse l'IP et le port du client.
- Remarques :
  - Le serveur STUN ne doit pas être sur le réseau privé du client.
  - L'IP et le port du client sont placés à l'intérieur du message STUN ($\Rightarrow$ non modifiés par le NAT lors du retour).
  - L'envoi du paquet STUN par le client permet d'ouvrir le port public alloué par le NAT.
  - STUN ne marche pas avec certains types de NAT : les NAT symétriques qui attribuent des ports publics en fonction de l'IP de destination.

Application de STUN à la ToIP

- STUN va permettre à l'UA d'ouvrir un port UDP public pour le flux RTP.
- Le port et l'IP publics sont ensuite placés dans le corps du INVITE.



1 req. STUN — IP src = 10.0.0.1 et port src = 10000 (port RTP privé)
2 req. STUN — IP src = 1.2.3.4 et port src = 24045 (port RTP public)
3 rép. STUN — contenu : IP = 1.2.3.4 et port = 24045
4–7 INVITE — contenu SDP :

```
c=IN IP4 1.2.3.4
m=audio 24045 RTP/AVP 116 0 8 101
```

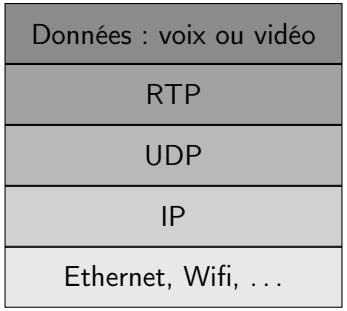
Plan

Utilisation de RTP et RTCP

- Une fois l'appel initié, SIP n'est plus utilisé durant l'appel.
- C'est RTP qui va permettre de transporter le flux audio entre les deux UA.
- RTCP est utilisé en complément de RTP pour contrôler la qualité de la communication.
- RTCP fournit périodiquement un rapport aux UA.

Présentation de RTP

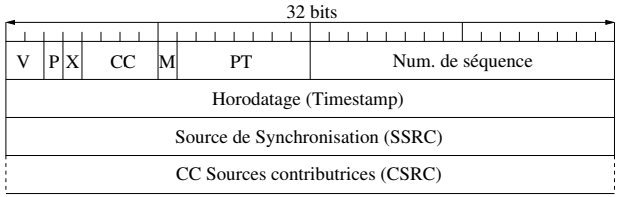
- RTP = Real-time Transport Protocol
- version initiale : janvier 1996 (RFC 1889)
- version actuelle : juillet 2003 (RFC 3550)
- utilisé par les applications ayant des contraintes temporelles fortes (téléphonie, vidéo, jeux vidéo, ...)
- pile protocolaire :



Fonctions principales de RTP

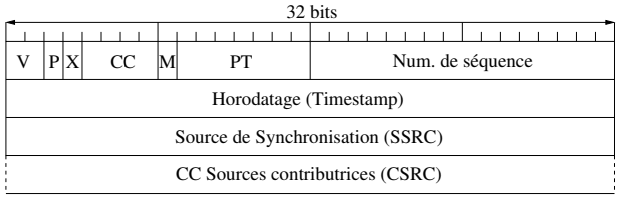
- Réordonancement des paquets.
  - Le réseau IP ne garantit pas que ordre de réception = ordre d'émission.
  - déséquencement en ToIP ⇒ mots restitués dans le désordre
- Synchronisation des flux vidéo et audio.
  - utilisation d'estampilles temporelles pour indiquer l'instant d'émission du paquet

L'en-tête RTP (1/2)



- **V** (numéro de version) = 2 actuellement
- **P** (Padding) = 1 si octets de bourrage dans les données
  - si P = 1, le dernier octet des données donne le nb. d'octets de bourrage
- **X** = 1 s'il y a une eXtension d'en-tête (utilisée par certains codecs)
- **CC** (CSRC Count) = nombre de CSRC dans l'en-tête (0 en pratique)
- **M** (Marker) = signification dépendante du PT
- **PT** (Payload Type, 7 bits) = type du codec utilisé (ex : 80 pour le G711)
- **Num. de séquence** (16 bits) = compteur incrémenté de 1 entre chaque paquet. utilisé :
  - par RTP pour réordonner les paquets
  - et par RTCP pour compter les pertes

L'en-tête RTP (2/2)



- **Timestamp** (32 bits) = estampille temporelle. utilisée :
  - par RTP pour la synchronisation voix-image
  - et par RTCP pour calculer la gigue.

  L'unité pour l'estampille et la période d'échantillonage. Par exemple :
  - Avec la plupart des codecs on a une fréquence d'échantillonnage de 8kHz.
  - Une estampille de 14 240 correspond à un temps de $\frac{7840}{8\times10^3} = 1.78$ s.

  L'estampille du 1er paquet RTP est généralement choisie aléatoirement.
- **SSRC** (Synchronisation SouRCe, 32 bits) = identifie la source de la synchronisation. Choisi aléatoirement.
- **CCSRC** (Contributing SouRCe, CC mot(s) de 32 bits) = identifient les sources contributrices (souvent CC = 0)

RTP en audio+vidéo

- Dans le cas d'un appel audio+vidéo on a 2 flux RTP sur 2 ports différents ($\Rightarrow$ chacun ouvert avec STUN ou équivalent si l'on a du NAT).
- C'est dans le corps du message INVITE que l'on précise les deux ports :

```
m=audio 49170 RTP/AVP ...
m=video 51372 RTP/AVP ...
```

- On utilise les estampilles temporelles des deux flux pour synchroniser la voix et l'image.

Le codec G711

- C'est le codec le plus utilisé.
- choix par défaut sur la plupart des équipements et logiciels de ToIP
- normalisé par l'UIT-T (Union internationale des télécommunications) dans les années 1970
- période d'échantillonnage de 125 μs
- 256 niveaux d'échantillonnage codés sur 8 bits

$\Rightarrow$ débit du codec $= \frac{8}{125 \times 10^{-6}} = 64 kb/s$

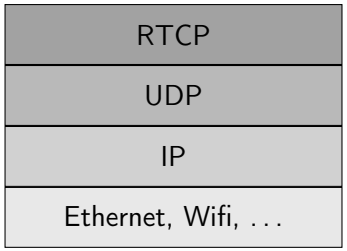
Autres codecs

Quelques codecs de l'UIT-T avec leur MOS (Mean Opinion Score), une note moyenne entre 0 et 5 attribuée par un panel d'utilisateurs.

Codec	Débit (kb/s)	MOS
G711	64	4,1
G722	64	4,5
G726	32	3,85
G728	16	3,61
G729	8	3,92

Présentation de RTCP

- RTCP = Real-time Transport Control Protocol
- défini dans les mêmes RFC que RTP
- toujours utilisé conjointement avec RTP
- permet aux UA de contrôler la qualité de la communication
- pile protocolaire :



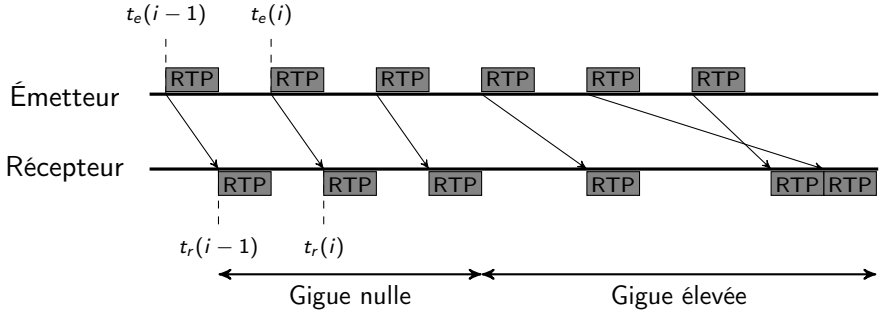
- RTCP n'encapsule pas de données.

Principe de RTCP

- Durant l'appel, les UA s'échangent périodiquement des **rapports** RTCP.
- Ces rapports fournissent des infos sur la qualité de la communication.
- Informations principales :
  - taux de perte (reconstitué à partir des Num. de séquence RTP)
  - gigue (voir mode de calcul plus loin)
- Le trafic RTCP ne doit pas occuper plus de **5%** de la bande passante de la session (trafic RTP + RTCP).

Calcul de la gigue par RTCP

gigue = variation dans le temps d'acheminement des paquets



- $GI(i)$ = Gigue instantanée en i
  $GI(i) = |t_r(i) - t_r(i-1)| - |t_e(i) - t_e(i-1)|$
- $G(i)$ = Gigue en i
  $G(i) = \frac{1}{16} \times GI(i) + \frac{15}{16} \times G(i-1)$
  (moyenne mobile exponentielle de coefficient 1/16)

Utilisation des rapports RTCP

- Et les rapports RTCP, qu'en fait-on ?
- Ce sont aux UA d'interpréter ces rapports et de les traiter de manière adéquate.
- Par exemple :
  1. Durant une session audio+vidéo, les rapports RTCP montrent un taux de perte élevé.
  2. Hypothèse : pertes dûes à une congestion du réseau.

  $\Rightarrow$ L'UA renégocie, pendant l'appel, les paramètres de la session en
  - 2.1 désactivant la vidéo
  - 2.2 utilisant un codec audio de moindre qualité
- Les paramètres de session (utilisation de la vidéo ou pas, codec audio à utiliser, ...) peuvent être renégociés lors d'une session par l'envoi d'un message SIP RE-INVITE.

Plan

La ToIP, une application temps-réel

- Critères de qualité de service (QoS) les plus importants en ToIP :
  - gigue
    - causes : congestion du réseau, paquets suivant des chemins différents, ...
    - conséquences : variations dans le débit de la conversation
  - délai d'acheminement de bout en bout (délai entre l'instant de production d'un octet par l'UA émetteur et celui ou il est restitué par l'UA récepteur)
    - délai max. pour une bonne qualité : 100 ms
    - délai max. tolérable : $300 - 500$ ms
    - Au-delà de 500 ms, ce n'est plus du temps réel.
- Critères secondaires :
  - bande passante
    - faible débit requis pour transporter de la voix
  - fiabilité
    - faible perte de paquets acceptable (baisse temporaire de la qualité)
- Nous allons voir quelques mécanismes de QoS pour la ToIP :
  - aux extrémités (sur les UA) : choix d'UDP vs TCP pour RTP, utilisation de mémoires tampon
  - dans le réseau : mécanismes de QoS

Choix d'UDP plutôt que TCP pour transporter la voix

UDP est plus adapté pour les applications temps-réel :

- en-tête de 8 o. pour UDP vs 20 pour TCP
- temps de traversée de la couche TCP $>$ temps de traversée de la couche UDP $\Rightarrow$ augmentation du délai d'acheminement de bout en bout avec TCP
- L'algorithme de contrôle de la congestion de TCP diminue radicalement le débit d'envoi en cas d'erreur $\Rightarrow$ le récepteur est alors en situation de "famine", en attente de données RTP.
- Les mécanismes de correction d'erreur avec envoi d'acquittements peuvent introduire des délais supplémentaires.
- L'avantage de TCP est la correction des erreurs mais dans le cas de la voix nous avons vu que celles-ci ne sont pas si graves.

Principe du contrôle de la congestion de TCP

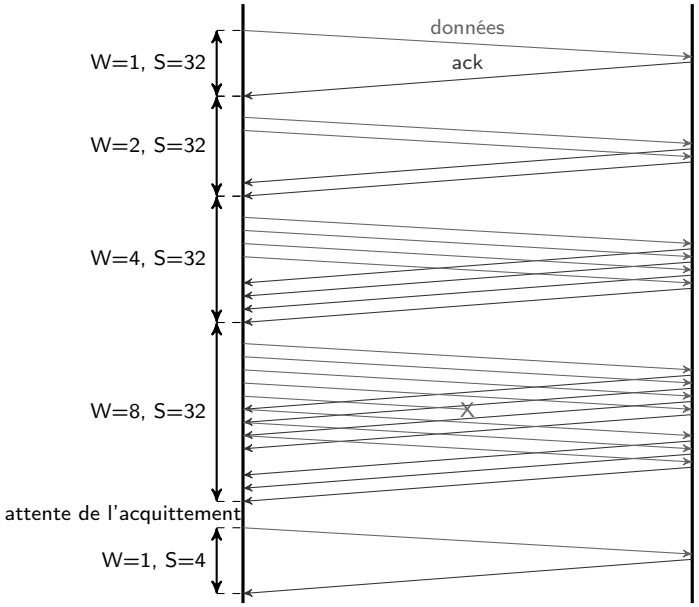
- en situation de congestion :
  - Les mémoires des routeurs sont saturées (modèle du seau percé).
  - Les routeurs détruisent les paquets qu'ils ne peuvent plus mémoriser.

  $\Rightarrow$ Lorsque la couche TCP envoie un paquet, aucun acquittement n'est reçu.
- Mais la congestion n'est pas la seule cause possible à la non réception d'un acquittement. Autre cause possible : erreur de transmission (même si la probabilité est faible).
- Principe du contrôle de la congestion de TCP :
  - Si un paquet n'est pas acquitté (dans les temps), c'est parce qu'il y a de la congestion.
  - Pour réduire la congestion, TCP va réduire radicalement le débit avant de le réaugmenter progressivement.

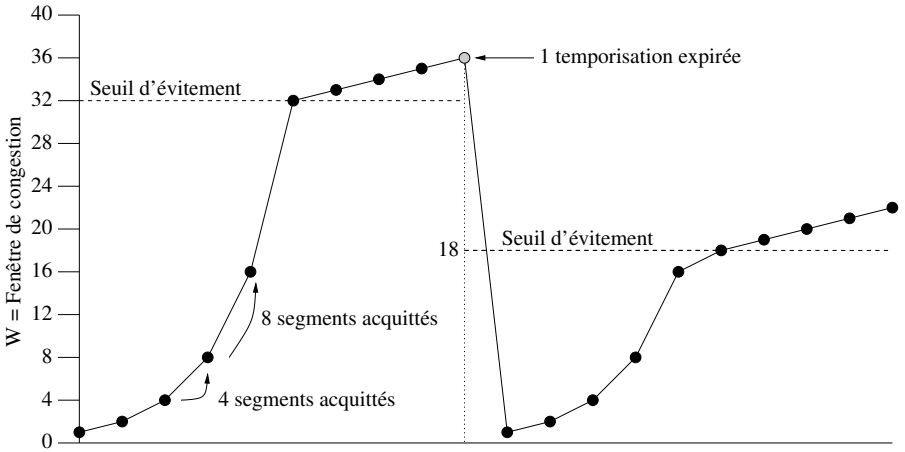
Algorithme de contrôle de la congestion de TCP

- TCP utilise deux variables pour contrôler la congestion :
  - W : fenêtre de congestion = nb. max de segments que l'on peut envoyer en rafale (sans acquittement).
  - S : seuil d'évitement de la congestion
- au début de la connexion, W = 1 et S = paramètre de l'OS (généralement 32 ou 64)
- quand tous les segments de la fenêtre sont acquittés :
  - si $W < S \Rightarrow W = \min(S, 2 \times W)$
  - si $W \geq S \Rightarrow W = W + 1$
- quand un segment n'est pas acquitté à temps :
  - $S = W/2$
  - $W = 1$

Scénario d'échange TCP



Évolution de la fenêtre de congestion



Les tampons en émission et en réception

2 tampons (zones mémoire) sont utilisés par les UA :

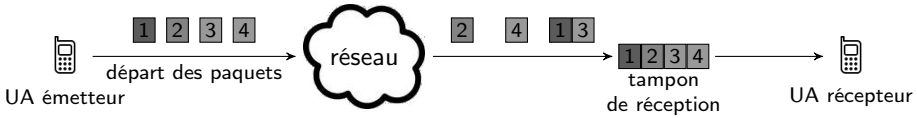
- en émission
- en réception

Le tampon en émission

- Son utilité : grouper les octets produits par le codec pour les envoyer en blocs.
- Quelle doit être sa taille (= taille des données RTP émises) ?
  - trop petit ⇒ mauvaise utilisation de la bande passante
  - trop grand ⇒ augmente le délai d'acheminement de bout en bout
- Cette taille est donc un compromis entre le taux d'utilisation de la bande passante et l'augmentation du délai d'acheminement.
- En pratique, on utilise des tampons de 100–500 o., ce qui correspond à environ 10–60 ms. de voix avec le G711.

Le tampon en réception

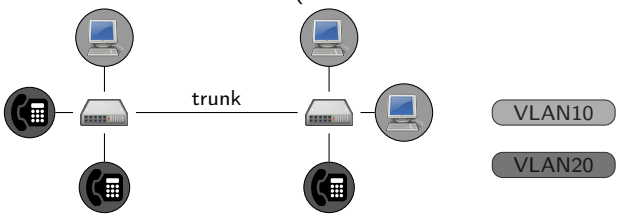
- Son utilité :
  - compenser la gigue
  - corriger les déséquencements



- À l'arrivée du 1er paquet dans le tampon (2 dans notre exemple), le récepteur déclenche une temporisation.
- Les paquets sont placés dans le tampon selon leurs num. de séquence RTP.
- Le contenu du tampon est ensuite restitué à l'UA récepteur
  - dès qu'il est plein ;
  - ou dès que la temporisation arrive à expiration (les paquets arrivant trop tard sont considérés comme perdus).

QoS de niveau 2 : séparation des flux grâce aux VLAN

- VLAN = réseaux locaux virtuels
- Permet de segmenter un réseau physique en plusieurs réseaux logiques.
- Exemple : un réseau avec 2 VLAN (10 = données et 20 = téléphonie).



- Quel intérêt pour la ToIP ? Donner une priorité aux flux
- priorité = 3 bits de l'étiquette de VLAN visible sur les liaisons trunk :

DA	SA	Id de prot.	**Pri.**	DEI	VLAN-id	DL/EType	Données + Bourrage	FCS
6 o.	6 o.	2 o.	**3 bits**	1 bit	12 bits	2 o.	46–1 500 o.	4 o.

← étiquette de 4 octets →

- Les switchs retransmettent les trames par priorité décroissante.
- On peut attribuer une priorité de 7 pour le VLAN 20 et 1 pour le VLAN 10.

⇒ Si la liaison entre les deux switchs est engorgée par un transfert de données, les paquets RTP seront tout de même retransmis.

- Voir le TP 3.