

M1 Informatique, Réseaux

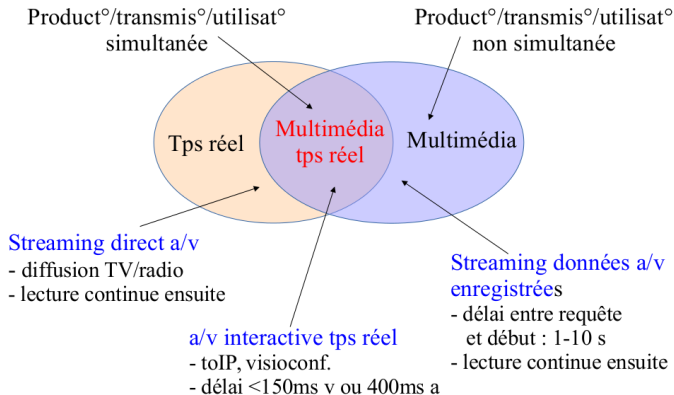
Cours 9 : Réseaux pour le multimédia

Olivier Togni
Université de Bourgogne, IEM/LE2I
Bureau G206
olivier.togni@u-bourgogne.fr

1^{er} mars 2021

Multimédia et temps réel

Application réseau multimédia = application réseau qui manipule audio ou vidéo



Vidéo

- ▶ transmission consomme beaucoup de bande passante $\Rightarrow$ compression
 - ▶ redondance spatiale et/ou
 - ▶ redondance temporelle
- ▶ serveurs stockent différentes version compressées adaptées aux débits



Audio

- ▶ Conversion analogique/numérique

Ex. PCM : 8000 échantillons/s codés sur 8 bits $\Rightarrow$ débit=64 Kb/s

- ▶ débit proportionnel à échantillonnage et codage $\Rightarrow$
 - ▶ compromis entre qualité et bande passante utilisée
 - ▶ compression pour transport

Ex. CD audio : PCM 44100 échantillons/s codés sur 16 bits $\Rightarrow$ 705,6 Kb/s en mono ou 1,411 Mb/s en stéréo

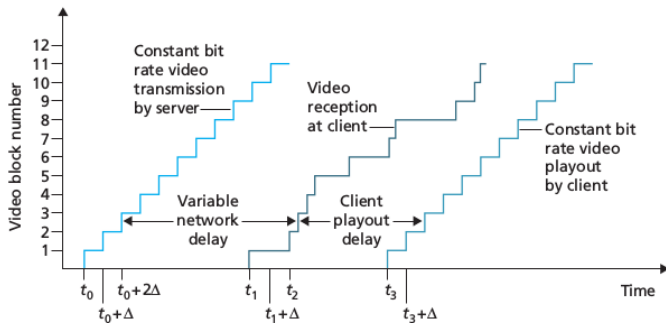
Streaming de vidéo enregistrée

3 catégories :

- ▶ sur UDP
- ▶ sur HTTP (et donc TCP)
- ▶ sur HTTP adaptatif

Buffer coté client pour atténuer variations de délai (gigue)

Délai de lecture



Streaming sur UDP

- ▶ serveur transmet avec débit = débit de lecture du client
Ex. paquets de 8000 bits, lecture à 2 Mb/s $\Rightarrow$ 1 paquet toutes les 4ms
- ▶ utilise un protocole adapté au transport de données a/v : RTP
- ▶ problèmes :
 - ▶ débit du réseau variable donc pb de lecture en continu
 - ▶ nécessite serveur de contrôle multimédia (ex. RTSP)
 - ▶ les firewall bloquent en général UDP

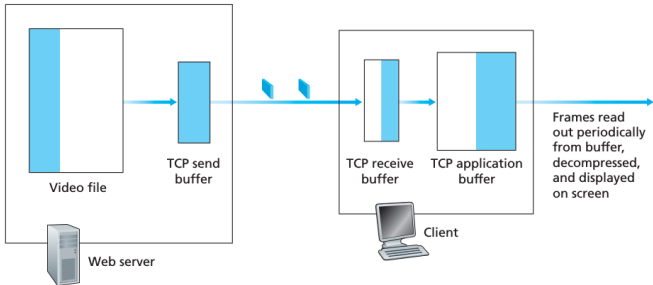
Streaming sur HTTP

- ▶ vidéo stockée comme les autres objets par le serveur web, accès par URL
- ▶ client fait requête GET sur URL
- ▶ serveur envoie le fichiers vidéo dans msg HTTP de réponse aussi vite que possible (dépend de TCP)
 - variation de débit due à contrôle de congestion et variation de délai si retransmissions
 - + traverse bien les firewall et NAT
 - + pas besoin de serveur de contrôle de média

⇒ utilisé par beaucoup de serveurs de média (ex. youtube)

Streaming sur HTTP

Bufférisation + préchargement



HTTP adaptatif

- ▶ DASH (Dynamic Adaptive Streaming over HTTP)
- ▶ chaque vidéo codée sous différentes versions avec différents débits (et qualités) à différentes URLs
- ▶ serveur maintient fichier manifeste contenant la liste des versions et leur débit de lecture)

HTTP adaptatif

- ▶ Client : 1 requête GET pour chaque morceau (chunk) de vidéo
- ▶ mesure la bande passante en réception $\Rightarrow$ adaptation de débit coté client
- ▶ si bande passante diminue, choix des morceaux suivants à débit moindre
- ▶ souvent même fonctionnement pour le son $\Rightarrow$ choix dynamique des morceaux de vidéo et son puis synchronisation locale

Réseaux de diffusion de contenus

- ▶ CDN (Content Delivery Network)
- ▶ des millions de vidéos sont distribuées à la demande chaque jours dans l'Internet $\Rightarrow$ besoin d'infrastructure performante chez les gros distributeurs
- ▶ 2 stratégies :
 - ▶ près des utilisateurs (ex. Akamai)
 - ▶ près des PoP des ISP

Fonctionnement des CDN

- ▶ chaque vidéo n'est pas forcément sur chaque serveur
- ▶ fonctionnement similaire à un cache : si pas sur le serveur, chargement en cache. Si plus de place en cache, suppression vidéo la moins récemment utilisée
- ▶ choix du cluster par
 - ▶ redirection DNS,
 - ▶ géographique,
 - ▶ mesure périodique des délais et taux de perte vers DNS locaux
 - ▶ anycasting

CDN : exemple de Youtube

- ▶ racheté par Google en 2006
- ▶ environ 50 des clusters de Google délivrent les vidéos demandées sur Youtube
- ▶ utilise le réseau privé de google entre les Data Centers
- ▶ choix du cluster en fonction du RTT + équilibrage de charge
- ▶ streaming HTTP sans DASH

Voix sur IP

- ▶ rappel : IP = Best effort (délai acheminement et absence perte non garantis)
- ▶ techniques d'amélioration au niveau de la couche application
- ▶ transport par UDP car TCP trop long si retransmission
- ▶ ex. débit source 8000 octets/s, 1 morceau envoyé toutes les 20 ms, mis dans segment UDP

Délai, gigue, taux de perte

- ▶ **délai de bout en bout** : non perçu en dessous de 150 ms ; acceptable jusqu'à 400 ms. Si > 400 ms, le client ne va pas considérer le paquet (paquet perdu)
- ▶ **gigue** : doit être prise en compte et peut être annulée par séquençement, horodatage et lecture différée
- ▶ **taux de perte** : suivant codage, taux entre 1 et 20 % acceptable, surtout si redondance des données

Forward Error Correction (FEC)

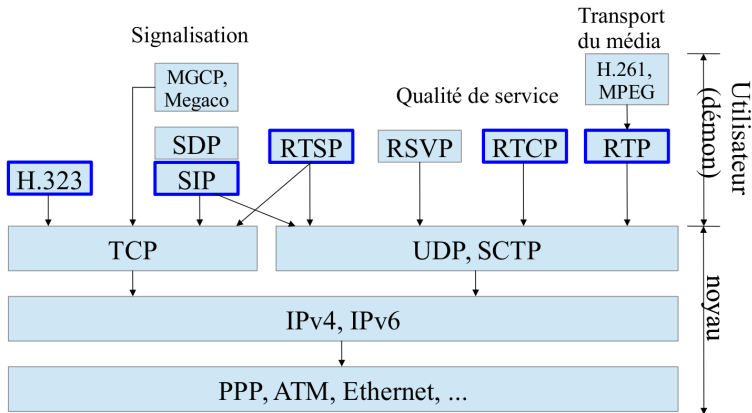
- ▶ ajouter des informations redondantes dans les paquets de sorte à pouvoir reconstruire (une approximation d')un paquet perdu
- ▶ méthode 1 : morceau redondant envoyé après n morceaux (XOR des n morceaux) $\Rightarrow$ reconstruction possible si une perte parmi les $n + 1$ paquets
- ▶ méthode 2 : version basse résolution du paquet $n - 1$ ajoutée au paquet n
- ▶ méthode 3 : entrelacement des morceaux avant envoi pour que deux morceaux originellement consécutifs soient assez distants
- ▶ méthode 4 : masquage d'erreur en remplaçant paquets perdus par paquets similaires.

Exemple de Skype

Protocoles propriétaires et paquets de données et contrôle cryptés

- ▶ client skype dispose de beaucoup de codecs différents pour encoder à différents débits et qualités (souvent 16000 échantillons/s pour la voix)
- ▶ audio et vidéo transportés par UDP, messages de contrôle par TCP (aussi données si firewall bloque UDP)
- ▶ récupération sur perte par FEC pour voix + vidéo
- ▶ système en partie pair à pair :
 - ▶ 2 types de pairs (super ou ordinaire) organisés en réseau
 - ▶ traversé de NAT : si deux pairs communicant sont derrière un NAT, passent chacun par un super pair qui va choisir un relais
 - ▶ index des associations (nom utilisateur, adresse IP) distribuée sur les super pairs
- ▶ si plus de 2 participants, flux audio envoyés à l'initiateur qui les combine et envoie le signal combiné à chaque participant

Protocoles pour applications interactives



RTP

- ▶ Real Time Protocol (RFC 3550) : acheminer des flux de données qui ont des propriétés temps réel
- ▶ Situé au niveau application, utilise TCP ou UDP (plus souvent UDP) et les modes unicast ou multicast
- ▶ RTCP (Realtime Transport Control Protocol, RFC 3550) : contrôler les flux acheminés par RTP
- ▶ 2 Intermédiaires : les **mixeurs** (regrouper plusieurs flots de plusieurs applications en un seul flot conservant le même format) et les **translateurs** (changer le format du codage, ex : de MPEG vers H.261)
- ▶ ne permet pas de réserver des ressources réseau, ni apporter une fiabilité, ni garantir délai de livraison

RTP : format des paquets

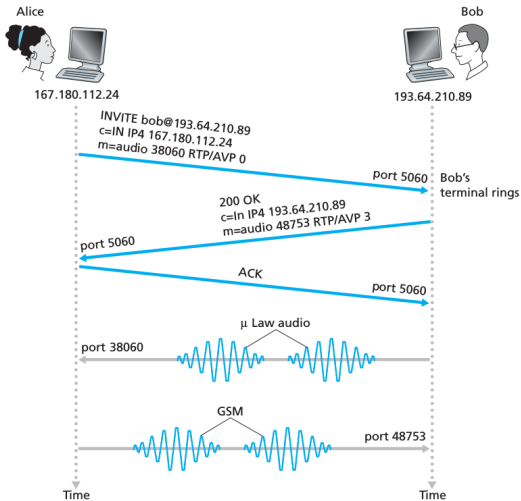
En-tête RTP sur 12 octets (+ options), dont :

- ▶ payload type (7bits) = type de codage (PCM = 0, GSM = 3, MPEG = 14, ...),
- ▶ sequence number (16 bits) = n° de séquence, +1 à chaque nouveau paquet
- ▶ Timestamp (32 bits) = horodatage (heure d'échantillonnage du 1er octet du paquet)
- ▶ Synchronization Source identifier (SSRC, 32 bits) : n° aléatoire identifiant la source

SIP

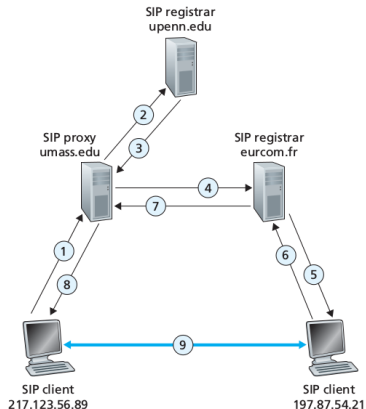
- ▶ Session Initiation Protocol, RFC 3261, RFC 5411)
- ▶ mécanismes d'établissement d'appel entre un appelant et un appelé sur réseau IP
 - ▶ l'appelant peut notifier son souhait de démarrer l'appel
 - ▶ les participant peuvent se mettre d'accord sur les codages utilisés
 - ▶ permet aux participants de pouvoir terminer un appel
- ▶ mécanismes pour que l'appelant détermine l'adresse IP de l'appelé (adr dynamique ou multiples IPs)
- ▶ mécanismes de gestion de l'appel : ajout nouveau flux pendant l'appel, changement encodage, transfert d'appel, ...

SIP : appel vers IP connue



SIP : translation de nom et localisation

- ▶ utilise des entités SIP :
proxies et registrars
- ▶ messages Register envoyés
par utilisateur SIP à son
Registrar quand appli SIP
démarrée



WebRTC

- ▶ API javascript (IETF et W3C)
- ▶ débute en 2011 mais rapidement intégrée aux navigateurs
- ▶ Architecture triangulaire pour mise en relation et échange tps réel entre pairs sans modules propriétaires, utilise le codec VP8
- ▶ Concurrent de CU-RTC-Web de Microsoft qui utilise le codec H.264 propriétaire

Qualité de Service

- ▶ Internet fonctionne en mode *Best effort*
- ▶ Service différencié avec qualité garantie de plus en plus nécessaire
- ▶ 3 approches :
 1. utiliser au mieux l'internet actuel
 2. différenciation de service
 3. Qualité de service par connexion

Comparaison des 3 approches

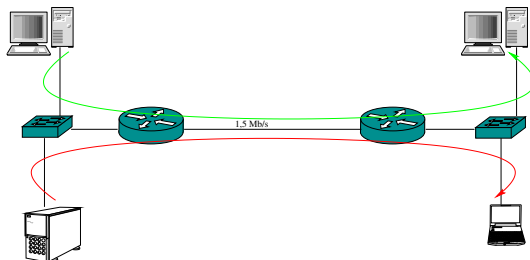
Approche	Granularité	Garantie	Mécanismes	Complexité	Etat
Utilisation au mieux	traitement égalitaire des trafics	pas ou peu	CDN, provisionnement de ressources	mini	actuel
différentiation de service	par classe	pas ou peu	marquage de paquets, régulation, ordonnancement	moyenne	peu déployé
QoS par connexion	flux traités différemment	grande	marquage de paquets, régulation, ordonnancement et admission d'appel	légère	très peu

Dimensionner les réseaux best effort

- ▶ But : provisionner suffisamment de capacité pour atteindre les performances requises par les applications
- ▶ provisionnement de bande passante (liens) ou de dimensionnement de réseau (plus complexe)
- ▶ besoin de :
 - ▶ modèles de demandes de trafic entre les terminaux
 - ▶ besoins de performances standards (ex : pour la VoIP, $Pr(\text{délai} > \text{seuil}) < \alpha$)
 - ▶ modèles de prédiction de performances et techniques d'allocation de bande passante

Proposer plusieurs classes de service

- ▶ Diviser le trafic en classes
- ▶ Proposer différents niveaux de services à ces différentes classes
- ▶ exemple : deux sessions se "partagent" un lien à 1,5 Mb/s



Proposer plusieurs classes de service

- ▶ **marquage de paquets** pour permettre aux routeurs de distinguer les paquets des différentes classes
- ▶ **isolation de trafic** entre les classes
 - ▶ régulation de trafic
 - ▶ allocation au niveau liaison (ordonnancement)
- ▶ mais en utilisant les ressources de façon efficace

Ordonnancement

- ▶ **FIFO** : paquet à transmettre sélectionné dans le même ordre que leur arrivée
- ▶ **File de priorité** : les paquets sont classifiés à l'arrivée dans des files suivant leur marque (le champs ToS d'IPv4 par exemple)
- ▶ **permutation circulaire** (round robin)
- ▶ **file d'attente équitable pondérée** (weighted fair queuing) : classe i de poids w_i est garantie de recevoir une fraction de service d'au moins $w_i / \sum w_j$

Régulation par seau percé

- ▶ régulation de
 - ▶ **débit moyen** : limitation du nombre de paquets sur une période (mais sur quel interval ?)
 - ▶ **débit crête** : limitation du nombre de paquets sur une période courte
 - ▶ **taille de rafale** : limitation du nombre de paquets sur une période très courte
- ▶ seau percé : seau contenant au max b jetons qui sont ajoutés à un débit de $r \Rightarrow$ limite le débit moyen
- ▶ seau percé + file équitable pondérée = délai max en file d'attente calculable

Différentiation de Service

- ▶ **Diffserv, RFC 2475** : traitement de différentes classes de trafic dans l'internet de façon scalable
- ▶ classification + marquage des paquets en bordure de réseau (hôte source Diffserv ou 1er routeur Diffserv par lequel le trafic passe) en utilisant le champs DS des paquets IPv4 ou IPv6
- ▶ forwarding (coeur du réseau) : chaque routeur a un comportement par saut suivant la marque du paquet (indépendant de la source de celui-ci)
 - ▶ **débit d'expédition garanti** par classe au niveau de chaque routeur
 - ▶ **bande passante et buffering garantis** par classe

Qualité de service par connexion

- ▶ **Réservation de ressource** : permet de garantir qu'un appel aura les ressources nécessaires à la QoS désirée
- ▶ **admission d'appel** : permet aux appels de demander et réserver leur ressources (blocage si ressources insuffisantes)
- ▶ **signalisation d'appel** : permet de réserver les ressources au niveau de chaque routeur du chemin source-destination
- ▶ **internet** : protocole RSVP (Ressource Reservation Protocol, RFC 2205) situé au niveau de la couche transport (modèle IntServ ou Service intégré de l'IETF)
- ▶ ajoute des informations d'état dans chaque routeurs ⇒ problème de passage à l'échelle