CONT 3/5 : Protocoles multimédia

Olivier Fourmaux

Version 1.3

CONT : plan du cours 3/5

Introduction

Quels protocoles pour la distribution de contenu ?

- initialement : transfert de fichier informatique
  - protocoles : FTP/TCP
- 1995-2005 : Internet du web + multimédia
  - protocoles : HTTP/TCP pour le web
  - démultiplication des types d'objets
  - apparition du P2P... pour l'échange de fichier
  - **un peu de protocoles spécifiques au multimédia (RTP, RTSP...)**
- 2005-... : Internet de la vidéo
  - **retour de HTTP (HLS, MSSS, HDS, DASH...)**

CONT : plan du cours 3/5

Applications multimédia

Nombreux mécanismes associés à ces applications (ex : pour une vidéo-conférence) :

- établissement d'une session
  - préparation (médias, codecs, protocoles...)
  - annonce et invitation (news, web, e-mail...)
  - initiation (signalisation ou connexion)
- participation
  - **émission** : codage/compression, paquétisation, transmission...
  - **réception** : décodage/décompression, lecture audio/vidéo...

➠ besoin de protocoles spécifiques pour :

- encapsuler les données multimédia
- gérer les participants
- ...

CONT : plan du cours 3/5

RTP

Real-Time Transport Protocol : RFC 1889 (v1) et RFC 3550 (v2)

- distribution de données multimédia de bout-en-bout
- nombreux paramètres pour les applications
  - identification (type de données, participants)
  - numérotation spatiale et temporelle des messages
  - synchronisation de flux
  - surveillance de qualité
  - communication vers un ou plusieurs destinataires

les algorithmes de codage, de synchronisation, de lecture sont implantés au niveau de l'application

RTCP

Real-Time Transport Control Protocol

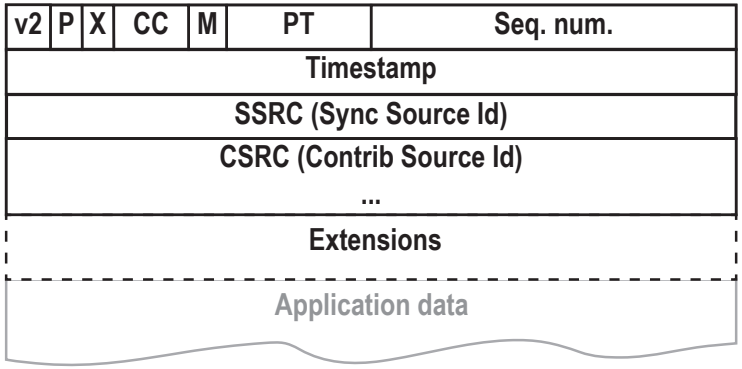
- totalement **intégré à RTP**
  - inclus dans les même documents (RFC 1889 et RFC 3550)
- transmet les **informations de session**
  - synchronisation
  - participants
  - ...
- fournit des **statistiques** sur la qualité de la session
- transmet des informations de contrôle sur la session
  - ex : identifier un participant sur les écrans des autres participants

Transport et RTP

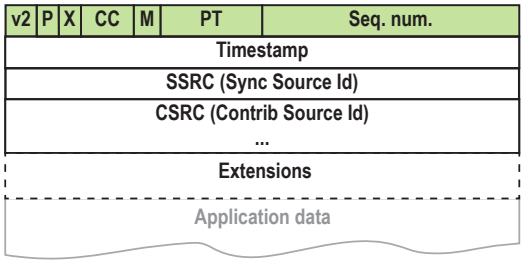
Couche transport associée à RTP/RTCP

- RTP est-il un protocole de transport ?
  - pas de mux/démux
- doit être associé à un protocole de transport classique
  - généralement UDP
  - TCP, DCCP, SCTP...
- configuration classique avec UDP :
  - numéro de port pair pour RTP
  - numéro de port impair suivant pour RTCP
  - **les numéros de port RTP et RTCP doivent être différents**

Format d'un message RTP

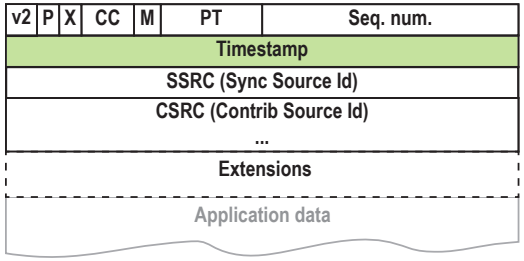


En-tête RTP



- P=1 si bourrage (padding), X=1 si extension présente
- CC : nombre d'identifiants CSRC qui suivent l'entête
- M : marqueur définit et interprété par un profil
- PT : type de données du message RTP (défini par un profil)
- Seq. num. : incrémenté de 1 pour chaque message RTP (valeur initiale aléatoire)

En-tête RTP : estampille temporelle



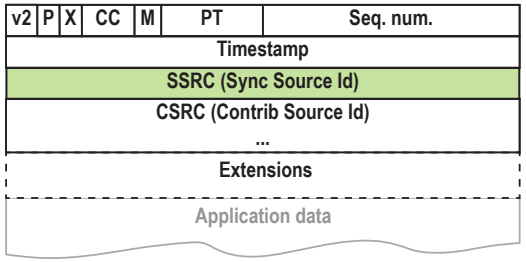
- instant d'échantillonnage du premier octet du message RTP
  - doit être liée à une horloge incrémentée de façon monotone et linéaire
  - fréquence de l'horloge dépendant du type des données (spécifiée par le profil)
  - valeur initiale aléatoire
- instant de présentation
  - calculés à partir d'une origine temporelle, dépend du profil

En-tête RTP : estampille temporelle (suite)

L'estampille temporelle permet la lecture au bon rythme des données multimédia :

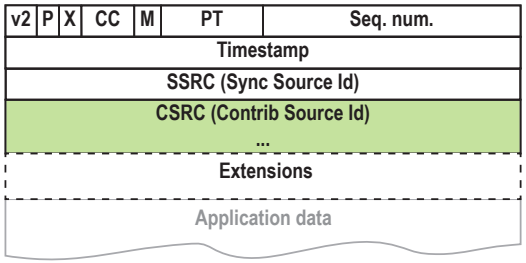
- **résolution suffisante** pour permettre le calcul de la gigue
  - (unité = une trame vidéo) est insuffisant
  - si les messages sont générés régulièrement (ex : 160 échantillons audio), alors l'horloge peut être celle d'échantillonnage (et s'incrémenter de 160 à chaque message)
- **horloge identique** pour tous les messages générés au même moment
  - ex : tous les messages associés à une même trame vidéo auront la même estampille temporelle (mais pas le même numéro de séquence)
- selon le codage la croissance des estampille peut être non monotone
  - ex : messages associés aux trames MPEG interpolées (IPBBBPBBB...)

En-tête RTP : SSRC



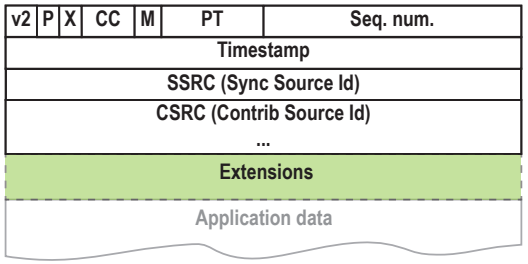
- identifie la source sur laquelle les données du message sont synchronisées
- à chaque SSRC correspond un interval de numéro de séquence
- choisi de manière aléatoire pour ne pas avoir 2 SSRC identiques dans la même session RTP (les implémentations de RTP doivent pouvoir gérer les collisions)
- chaque application peut avoir plusieurs SSRC

En-tête RTP : CSRC



- liste de 0 à 15 items de 32 bits
- identifie les sources qui ont des données dans le message RTP.
- les CSRC sont insérés par les "*mixers*"
- plusieurs utilisations possibles, exemple :
  - identifier les interlocuteurs dans une session de télé-conférence (le mixer indique son SSRC et les SSRC des sources initiales deviendrons des CSRC)

En-tête RTP : extensions



- permet de réaliser une implementation propriétaire
  - dans un cadre expérimental par exemple (pour de tester de nouveaux formats de données)
  - paramètres d'extension définis par l'implementation

Profiles RTP audio et vidéo

Le RFC 3551 défini un profil de base sans négociation de paramètres

PT	encodage	A/V	Clock (Hz)
0	PCMU	A	8000
2	G721	A	8000
3	GSM	A	8000
5	DVI4	A	8000
7	LPC	A	8000
8	PCMA	A	8000
9	G722	A	8000
15	G728	A	8000
26	JPEG	V	90000
31	H261	V	90000...

Règles d'encodage RTP

Règles génériques d'encodage RTP :

- diffusion contrôlée au niveau de la **source**
  - choix d'émission avec différent PT
  - segmentation des données au niveau du codeur
  - utilisation du bit M si besoin
- **multiplexage** de plusieurs types de données
  - interdit dans une session RTP
  - utiliser plusieurs sessions en parallèle
- mécanismes audio et vidéo très différents...

Règles d'encodage audio RTP (1)

Règles de base pour l'encodage audio avec RTP :

- suppression des silence (grâce aux estampilles temporelles)
  - mais ajout possible d'un descripteur de silence *(Confort Noise)*
  - positionnement du marqueur au premier message audio après un silence (pour distinguer le silence d'un problème réseau)
- horloge RTP indépendante du nombre de canaux utilisés et de l'encodage
  - si N canaux alors N échantillons pendant une période
- fréquences d'échantillonnage initiales
  - 8000, 11025, 16000, 22050, 24000, 32000, 44100 et 48000 Hz

Règles d'encodage audio RTP (2)

Sessions multi-canaux : les échantillons d'un même instant doivent être dans le même message RTP en suivant cet ordre

number	channel						
	1	2	3	4	5	6	
stereo	l	r					
3	l	r	c				
4	l	c	r	S			
5	Fl	Fr	Fc	Sl	Sr		
6	l	lc	c	r	rc	S	

l	left
r	right
c	center
S	surround
F	front

Recommandation de fonctionnement pour l'audio RTP

Taille des données audio transportées :

- données audio dans un message : **20 ms** ou **une trame** encodée
  - **l'interval de temps dans un message ➠ délai de bout-en-bout mini.**
  - interval plus long ➠ augmentation du délai et du taux de perte
  - valeurs acceptables : 0 à 200 ms
- encodages basés sur des **échantillons**
  - chaque échantillon codée sur S bits fixes
  - les bits des échantillons sont écrits à la suite
    - si multicanal, les échantillons du même instants sont écrits à la suite dans l'ordre définis
  - durée d'un messages dépend du nombre N d'échantillons
  - les données (S*N bits) doivent être un multiple de l'octet
- encodages basés sur des **trames**
  - encode un intervalle audio de taille fixe prédéfinie
  - codage résultant de taille variable ou fixe
    - plusieurs trames peuvent être intégrées dans un message (nécessite de pouvoir les séparer)
  - en multicanal, range écrit à la suite dans l'ordre défini

Exemple audio échantillonnée : L16

Codage PCM 16 bits

- données = suite de mots de 16 bits
  - entiers signés (en complément à 2)
  - transmis selon l'ordre des octets du réseau *(MSB first)*
- information complémentaires transmises hors-bande (SDP...)
  - fréquence d'échantillonnage
  - nombre de canaux (et ordre si besoin)
  - traitement analogique amont (amplification etc.)

Exemple audio avec trame : GSM

Codage GSM : blocs de 160 échantillons (20 ms) codées en 33 octets

Octet	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
0	1	1	0	1	LARc0.0	LARc0.1	LARc0.2	LARc0.3
1	LARc0.4	LARc0.5	LARc1.0	LARc1.1	LARc1.2	LARc1.3	LARc1.4	LARc1.5
2	LARc2.0	LARc2.1	LARc2.2	LARc2.3	LARc2.4	LARc3.0	LARc3.1	LARc3.2
3	LARc3.3	LARc3.4	LARc4.0	LARc4.1	LARc4.2	LARc4.3	LARc5.0	LARc5.1
4	LARc5.2	LARc5.3	LARc6.0	LARc6.1	LARc6.2	LARc7.0	LARc7.1	LARc7.2
5	Nc0.0	Nc0.1	Nc0.2	Nc0.3	Nc0.4	Nc0.5	Nc0.6	bc0.0
6	bc0.1	Mc0.0	Mc0.1	xmaxc00	xmaxc01	xmaxc02	xmaxc03	xmaxc04
7	xmaxc05	xmc0.0	xmc0.1	xmc0.2	xmc1.0	xmc1.1	xmc1.2	xmc2.0
8	xmc2.1	xmc2.2	xmc3.0	xmc3.1	xmc3.2	xmc4.0	xmc4.1	xmc4.2
9	xmc5.0	xmc5.1	xmc5.2	xmc6.0	xmc6.1	xmc6.2	xmc7.0	xmc7.1
10	xmc7.2	xmc8.0	xmc8.1	xmc8.2	xmc9.0	xmc9.1	xmc9.2	xmc10.0
11	xmc10.1	xmc10.2	xmc11.0	xmc11.1	xmc11.2	xmc12.0	xmc12.1	xcm12.2
12	Nc1.0	Nc1.1	Nc1.2	Nc1.3	Nc1.4	Nc1.5	Nc1.6	bc1.0
13	bc1.1	Mc1.0	Mc1.1	xmaxc10	xmaxc11	xmaxc12	xmaxc13	xmaxc14
14	xmax15	xmc13.0	xmc13.1	xmc13.2	xmc14.0	xmc14.1	xmc14.2	xmc15.0
15	xmc15.1	xmc15.2	xmc16.0	xmc16.1	xmc16.2	xmc17.0	xmc17.1	xmc17.2
16	xmc18.0	xmc18.1	xmc18.2	xmc19.0	xmc19.1	xmc19.2	xmc20.0	xmc20.1
17	xmc20.2	xmc21.0	xmc21.1	xmc21.2	xmc22.0	xmc22.1	xmc22.2	xmc23.0
18	xmc23.1	xmc23.2	xmc24.0	xmc24.1	xmc24.2	xmc25.0	xmc25.1	xmc25.2
19	Nc2.0	Nc2.1	Nc2.2	Nc2.3	Nc2.4	Nc2.5	Nc2.6	bc2.0
20	bc2.1	Mc2.0	Mc2.1	xmaxc20	xmaxc21	xmaxc22	xmaxc23	xmaxc24
21	xmaxc25	xmc26.0	xmc26.1	xmc26.2	xmc27.0	xmc27.1	xmc27.2	xmc28.0
22	xmc28.1	xmc28.2	xmc29.0	xmc29.1	xmc29.2	xmc30.0	xmc30.1	xmc30.2
23	xmc31.0	xmc31.1	xmc31.2	xmc32.0	xmc32.1	xmc32.2	xmc33.0	xmc33.1
24	xmc33.2	xmc34.0	xmc34.1	xmc34.2	xmc35.0	xmc35.1	xmc35.2	xmc36.0
25	Xmc36.1	xmc36.2	xmc37.0	xmc37.1	xmc37.2	xmc38.0	xmc38.1	xmc38.2
26	Nc3.0	Nc3.1	Nc3.2	Nc3.3	Nc3.4	Nc3.5	Nc3.6	bc3.0
27	bc3.1	Mc3.0	Mc3.1	xmaxc30	xmaxc31	xmaxc32	xmaxc33	xmaxc34
28	xmaxc35	xmc39.0	xmc39.1	xmc39.2	xmc40.0	xmc40.1	xmc40.2	xmc41.0
29	xmc41.1	xmc41.2	xmc42.0	xmc42.1	xmc42.2	xmc43.0	xmc43.1	xmc43.2

Autres formats d'encapsulation RTP audio

RFC 2658 RTP PureVoice(tm) Audio
RFC 3389 RTP Comfort Noise (CN)
RFC 3558 RTP for Enhanced Variable Rate Codecs (EVRC)
RFC 4184 RTP for AC-3 Audio
RFC 4348 RTP for VMR-WB Audio Codec
RFC 4352 RTP for AMR-WB+ Audio Codec
RFC 4867 RTP for AMR (Adaptive Multi-Rate) and AMR-WB Audio Codecs
RFC 4598 RTP for Enhanced AC-3 (E-AC-3) Audio
RFC 5188 RTP for the Enhanced Variable Rate Wideband Codec (EVRC-WB)
RFC 5215 RTP for Vorbis Encoded Audio
RFC 5219 A More Loss-Tolerant RTP Payload Format for MP3 Audio
RFC 5391 RTP for ITU-T Recommendation G.711.1
RFC 5404 RTP for G.719
RFC 5574 RTP for the Speex Codec
RFC 5577 RTP for ITU-T Recommendation G.722.1 (wideband)
RFC 5584 RTP for the Adaptive TRansform Acoustic Coding (ATRAC) Family
RFC 5993 RTP for GSM Communications Half Rate (GSM-HR)
RFC 6884 RTP for the Enh. Var. Rate Narrow-Wideband Codec (EVRC-NW)
RFC 7587 RTP for the Opus Speech and Audio Codec

Règles d'encodage vidéo RTP

Règles de base pour l'encodage vidéo avec RTP :

- la plupart des encodages vidéo utilisent une horloge à 90000 Hz
  - identique à celle de MPEG
  - multiple entier de 24 (HDTV), 25 (PAL/SECAM), 29.97 (NTSC) et 30 Hz (HDTV)
  - suffisamment élevée pour calculer la gigue
- estampille temporelle ➠ instant d'échantillonnage de l'image
  - si image sur plusieurs messages : même estampille
- le bit de marquage est utilisé sur le dernier message d'une image

Exemple vidéo : H264

H264 spécifie une couche d'abstraction du réseau
(NAL : Network Abstraction Layer)

- paquetisation contrôlée au niveau du codeur
  - découpage des images en tranches *(slices)* autonomes (au niveau du décodage image)
  - puis partition en unités NAL (NALU)
- NALU autonomes (au niveau du décodage transport)
  - entête NALU (1 octet)
  - données NALU (flux d'octets bruts : image ou paramètres H264)
- intégration dans RTP
  - habituellement une NALU $\sim$ 1.4 Ko
  - plusieurs NALU par messages
  - grandes NALU sur plusieurs messages

Autres formats d'encapsulation RTP vidéo

RFC 2435 RTP for JPEG-compressed Video
RFC 2250 RTP for MPEG1/MPEG2
RFC 3640 RTP for Transport of MPEG-4 Elementary Streams
RFC 4425 RTP for Video Codec 1 (VC-1)
RFC 4587 RTP for H.261 Video Streams
RFC 4628 RTP for H.263 Video Streams
RFC 4629 RTP for the 1998 Version of ITU-T Rec. H.263 Video (H.263+)
RFC 5371 RTP for JPEG 2000 Video Streams
RFC 6184 RTP for H.264 Video
RFC 6190 RTP for Scalable Video Coding (extension H264 AVC)
RFC 6416 RTP for MPEG-4 Audio/Visual Streams
RFC 6469 RTP for DV (IEC 61834) Video
RFC 7741 RTP for VP8 Video
RFC 7798 RTP for High Efficiency Video Coding (HEVC)

CONT : plan du cours 3/5

Contrôle de la transmission : RTCP

RTCP permet d'évaluer la qualité de la transmission

- maintient des **informations de session**
  - synchronisation
  - participants
  - ...
- transmission périodique de **messages de contrôle**
  - vers tous les participants d'une session multimédia (même chemins que les messages RTP)
  - multicast ou unicast
- fournit des statistiques
- contrôle le débit d'une session RTP
  - information de retour vers une source *(feedback)*
  - **permet à la source de s'adapter à ses destinataires**

Format des messages RTCP

2	1	5	8	16 bits
V	**P**	**xC**	**PT**	**length**
...				

- **V** = version (2 actuellement)
- **P** = bourrage *(padding)* inclus dans la longueur
  - taille =valeur du dernier octet de bourrage (crypto par bloc)
- **xC** = nombre d'éléments dans ce message RTCP
- **PT** = type de message :
  - 200 = SR *(Sender Report)* : informations par les émetteurs
  - 201 = RR *(Receiver Report)* : statistiques par les récepteurs
  - 202 = SDES *(Source DEScription)* : information sur la source en UTF8 (CNAME, e-mail, numéro de téléphone, localisation...)
  - 203 = BYE : quitter une session RTP
  - 204 = APP *(APPlication)* : fonctions spécifiques de l'application
- **length** = nombre de mots de 32 bits (entête et bourrage inclus)
  - délimitation s'il y a plusieurs messages RTP dans l'entité de transport (datagramme UDP ou autre)

Format des messages SR

2	1	5	8	16 bits
V	**P**	**RC**	**PT**	**length**
SSRC of sender				
sender info ...				
report 1 ...				
report 2 ...				
...				

- **RC** = nombre de **report** inclus dans les message RTCP
- **SSRC** *(Synchronization SouRCe)* = identifiant de l'origine du msg
- 1 * **sender info** spécifique aux messages SR
- RC * **report**

Format des messages SR : sender info

2	1	5	8	16 bits
NTP timestamp, most significant word				
NTP timestamp, least significant word				
RTP timestamp				
sender's packet count				
sender's octet count				

- **NTP** *(Network Time Protocol)* = temps absolu *(wallclock time)*
  - estampille sur 64 bits (32 bits ➠ secondes + 32 bits ➠ fractions de s.)
  - relatif au 1 janvier 1900 à 0h00 (Fin en 2036 !)
- **length** = nombre de mots de 32 bits (entête et bourrage inclus)
- **RTP timestamp** = correspondance RTP du temps NTP pour cette session RTP (avec l'unité de temps spécifique)
- **sender's packet count** = nombre de messages RTP pour la session
- **sender's octet count** = idem pour qtt de données utiles

Format des messages RR

2	1	5	8	16 bits
V	P	RC	PT	length
SSRC of packet sender				
report 1 …				
report 2 …				
…				

- **RC** = nombre de **report** inclus dans les message RTCP
- **SSRC** *(Synchronization SouRCe)* = identifiant de l'origine du msg
- RC * **report**

Format des messages SR et RR : report

2	1	5	8	16 bits
SSRC N (SSRC of source N)				
fraction lost	cumulative number of packets lost			
extended highest sequence number received				
interarrival jitter				
last SR (LSR)				
delay since last SR (DLSR)				

- **SSRC N** = identifiant de la source auquel se rapporte ce **report**
- **fraction lost** = perte depuis le dernier SR/RR
- **cumul. num. of pkt lost** = perte depuis le début de la session
- **ext. hi. seq. num. received** = num. séq. RTP étendu du nombre de cycle
- **interarrival jitter** = écart moyen lissé (temps NTP sur 32 bits)
  - $J_i = J_{i-1} + (|D(i-1,i)| - J_{i-1})/16$ avec $D(i,j) = (R_j - R_i) - (S_j - S_i)$
- **LSR** = date d'émission du dernier SR (32 bits)
- **DLSR** = diff. dernier SR et l'émission du RR (32 bits)

Calcul temporels RTCP

Manipulation du temps NTP sur 64 et 32 bits :

- **T1 = 10 novembre 1995 à 11h33 et 25,125 secondes**
  - **T1 sur 64 bits** *(NTP timestamp)*
    - secondes NTP : `0x b44d b705`
    - fraction NTP : `0x 2000 0000`
  - **T1 sur 32 bits** : `0x b705 2000`
- **T2 = 10 novembre 1995 à 11h33 et 36,500 secondes**
  - **T2 sur 64 bits** *(NTP timestamp)*
    - secondes NTP : `0x b44d b710`
    - fraction NTP : `0x 8000 0000`
  - **T2 sur 32 bits** : `0x b710 8000`
- si SR envoyé à T1, RR recut à T2 avec :
  - **LSR** = `0x b705 2000`
  - **DLSR** = `0x 0005 4000`

➠ RTT ?

RTP/RTCP

Comment partager la bande passante entre RTP et RTSP ?

- règle : limiter le trafic de contrôle à moins de 5%

Comment limiter le trafic RTCP ?

- unicast : calcul direct
- multicast : plus il y a de participants, plus il faut limiter les envois de messages RTCP

Controle de débit RTCP :

- algorithme basé sur l'estimation distribuée du nombre de participants $N(t)$ (en comptant les SSRC des **reports**)
- $D_{RTCP} \leq \frac{0.05 D_{RTP}}{N(t) T_{mesgRTCP}}$
- ajout d'une composante aléatoire pour éviter les synchronisations

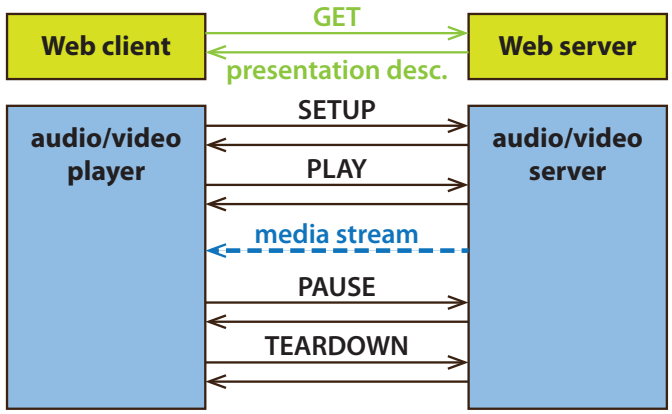
CONT : plan du cours 3/5

RTSP

Real Time Streaming Protocol (RFC 2326)

- contrôle **hors bande** de la diffusion (habituellement TCP port 554)
- identification de la resource via URL (ex : `rtsp://media.upmc.fr:554/videofile`)
- fonctionnalités typiques d'un lecteur vidéo :
  - lecture/pause
  - avance rapide
  - accès à une position temporelle...
- **ne définit aucun mécanismes de codage pour la vidéo/audio**
- **ne définit pas la méthode d'encapsulation des données**
- **n'impose aucun mode de mise en mémoire tampon du lecteur média**

RTSP : fonctionnement



RTSP : commandes

La première ligne d'une requête est une commande :

- commandes principales (obligatoires)
  - **OPTIONS** : indique les commandes disponibles
  - **SETUP** : début d'une session RTSP et allocation des ressources coté serveur
  - **PLAY** : démarrage de la transmission d'un flux
  - **TEARDOWN** : arrêt définitif avec libération des ressources
- commandes additionnelles (optionnelles)
  - **DESCRIBE** : description technique d'un média (recom.)
  - **PAUSE** : arrêt temporaire de la transmission d'un flux (recom.)
  - **ANNOUNCE** : changement dans la description d'un média (SDP)
  - **RECORD** : demande d'enregistrement d'un flux
  - **REDIRECT** : redirection du client vers un nouveau serveur
  - **GET_PARAMETER** : récupération de paramètres de transmission
  - **SET_PARAMETER** : contrôle du codage ou du périphérique

RTSP : réponses

La première ligne d'une réponse est un *status code* suivi d'un texte :

- 1xx *(Informational)* : requête reçue, processus en cours
  - `100 Continue`
- 2xx *(Success)* : action comprise et acceptée
  - `200 OK`
  - `201 Created`
- 3xx *(Redirection)* : d'autres actions doivent être réalisées
  - `300 Multiple Choices`
  - `302 Moved Temporarily`
  - `305 Use Proxy`
- 4xx *(Client Error)* : valide mais problématique à cause du client
  - `400 Bad Request`
  - `415 Unsupported Media Type`
  - `453 Not Enough Bandwidth`
- 5xx *(Server Error)* : valide mais problématique du côté serveur
  - `500 Internal Server Error`
  - `503 Service Unavailable`

RTSP : exemple

```
C->W: GET /concert.sdp HTTP/1.1
      Host: www.upmc.fr

W->C: HTTP/1.1 200 OK
      Content-Type: application/x-rtsl

      <session>
        <track src="rtsp://live.upmc.fr/concert/audio">
      </session>

C->M: DESCRIBE rtsp://live.upmc.fr/concert/audio RTSP/1.0
      CSeq: 1

M->C: RTSP/1.0 200 OK
      CSeq: 1
      Content-Type: application/sdp
      Content-Length: 44

      v=0
      o=- 2890844526 2890842807 IN IP4 132.227.24.202
      s=RTSP Session
      m=audio 3456 RTP/AVP 0
      a=control:rtsp://live.upmc.fr/concert/audio
      c=IN IP4 224.2.0.1/16

C->M: SETUP rtsp://live.upmc.fr/concert/audio RTSP/1.0
      CSeq: 2
      Transport: RTP/AVP;multicast

M->C: RTSP/1.0 200 OK
      CSeq: 2
      Transport: RTP/AVP;multicast;destination=224.2.0.1;
          port=3456-3457;ttl=16
      Session: 0456804596

C->M: PLAY rtsp://live.upmc.fr/concert/audio RTSP/1.0
      CSeq: 3
      Session: 0456804596

M->C: RTSP/1.0 200 OK
      CSeq: 3
      Session: 0456804596
```


CONT : plan du cours 3/5

SDP

Session Description Protocol (RFC 4566)

- **description des sessions** multimedia pour leur initialisation
- présente les détails du média à transmettre/recevoir aux participants
  - adresses, identifiants, codecs, métadata, etc.
- ce n'est pas un protocole de transmission/transport
  - **seulement de la description**
- peut être utilisé sur un protocole de transport quelconque
- peut être intégré à tout protocoles d'initialisation/signalisation
  - RTSP, SIP, etc.
- **format standard** de présentation
  - indication de contenu d'une description SDP avec `Content-Type: application/sdp`
  - la description SDP est une suite de ligne de texte de type `<type>=<value>`

SDP : exemple

```
v=0
o=fourmaux 2990844526 2990842807
    IN IP4 132.227.63.51
s=SDP Seminar
i=A Seminar on SDP
u=http://www.upmc.fr/sem/sdp.pdf
e=olivier.fourmaux@upmc.fr
c=IN IP4 224.2.17.12/127
t=2973397496 2973404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
```

Types SDP possibles :

```
v=  (protocol version)
o=  (originator and session identifier)
s=  (session name)
i=* (session information)
u=* (URI of description)
e=* (email address)
p=* (phone number)
c=* (connection information)
b=* (0 or more bandwidth information lines)
t=  (time the session is active)
r=* (0 or more repeat times)
z=* (time zone adjustments)
k=* (encryption key)
a=* (0 or more session attribute lines)
   0 or more media descriptions
     m=  (media name and transport address)
     i=* (media title)
     c=* (connection information)
     b=* (0 or more b.w. information lines)
     k=* (encryption key)
     a=* (0 or more media attribute lines)
```

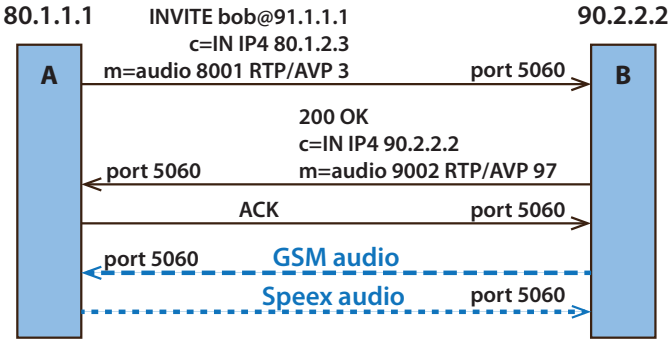

CONT : plan du cours 3/5

SIP

Session Initiation Protocol (RFC 3261)

- mise en place d'appel
  - **notification** (intention d'établir un appel)
  - **négociation** (codages, média...)
  - **terminaison** (fin de l'appel/session)
- utilisateurs identifiés par noms et/ou e-mails (pas de numéros de téléphone)
- correspondance entre identifiants et adresses IP
- gestion des appels
  - **ajout de nouveaux média** pendant l'appel
  - **changement de codage** en cours
  - **invitation** de participants
- protocole de base de l'IMS (IP Multimedia Subsystem du 3GPP)

SIP : fonctionnement



- **négociation** du codec (réponse 606 `not acceptable` avec la liste des codec supportés)
- **rejet** d'appel (réponse `busy`, `gone`, `payment`, `forbidden`...)
- **transmission** (données multimédia envoyées avec RTP ou un autre protocole)

SIP : exemple

```
INVITE sip:auto@localhost SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:13764;rport
Max-Forwards: 70
Contact: <sip:matthew@127.0.0.1:13764>
To: "sip:auto@localhost"<sip:auto@localhost>
From: "Olivier"<sip:olivier@upmc.fr>;tag=5c7cdb68
Call-ID: NmNhYWNhMjY0Y2MOOT.
CSeq: 1 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY,
   MESSAGE, SUBSCRIBE, INFO
Content-Type: application/sdp
User-Agent: eyeBeam release 1014g stamp 44944
Content-Length: 389

v=0
o=- 7 2 IN IP4 132.227.61.199
...
```

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 127.0.0.1:13764;;rport=13764;received=127.0.0.1
To: "sip:auto@localhost" <sip:auto@localhost>;tag=tCAED3F5A
From: "Olivier"<sip:olivier@upmc.fr>;tag=5c7cdb68
Call-ID: NmNhYWNhMjY0Y2MOOTc4YTI2MzgzZTNlYTRhZTMxNTE.
CSeq: 1 INVITE
Contact: <sip:127.0.0.1:5060>
Content-Type: application/sdp
Content-Length: 271

v=0
o=olivier 1208261984604 1208261984604 IN IP4 127.0.0.1
...
```

```
ACK sip:127.0.0.1:5060 SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:13764;rport
Max-Forwards: 70
Contact: <sip:matthew@127.0.0.1:13764>
To: "sip:auto@localhost"<sip:auto@localhost>;tag=tCAED3F5A
From: "Olivier"<sip:olivier.@upmc.fr>;tag=5c7cdb68
Call-ID: NmNhYWNhMjY0Y2MOOTc4YTI2MzgzZTNlYTRhZTMxNTE.
CSeq: 1 ACK
Content-Length: 0
```

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 127.0.0.1:13764;rport=13764;received=12
To: "sip:auto@localhost" <sip:auto@localhost>
From: "Olivier"<sip:olivier@upmc.fr>;tag=5c7cdb68
Call-ID: NmNhYWNhMjY0Y2MOOTc4YTI2MzgzZTNlYTRhZTMxNTE.
CSeq: 1 INVITE
Content-Length: 0
```


SIP : conversion de nom et localisation

Comment faire correspondre l'identifiant SIP (nom/email) à l'adresse IP de l'appelé ?

- l'appelé peut être mobile
- il peut obtenir une adresse IP dynamique/privée
- il peut avoir plusieurs appareils IP (ordinateur, tablette, smartphone... etc.)

Avec quelle flexibilité ?

- en fonction du temps et du lieu
- en fonction de l'état de l'appel et/ou de l'appelée (transfert d'appel, etc.)

➠ ce service est assuré par des serveurs SIP :

- serveur **SIP Registrar**
- serveur **SIP Proxy**

SIP : Registrar

Au démarrage du client SIP de l'utilisateur :
➠ émission d'un message `REGISTER` vers le **serveur SIP Registrar** de l'utilisateur.

```
REGISTER sip:upmc.fr SIP/2.0
Via: SIP/2.0/UDP 132.227.61.205:5061;rport;branch=z9hG4bKDC8595CD770E4317ACBC3B64CF5016C
From: Olivier <sip:olivier@upmc.fr>;tag=1516659370
To: Olivier <sip:olivier@upmc.fr>
Contact: "Olivier" <sip:olivier@132.227.61.205:5061>
Call-ID: 46E1C3CB36304F84A020CF6DD3F96461@Verso.com
CSeq: 37764 REGISTER
Expires: 1800
Max-Forwards: 70
User-Agent: LIP6-SIP-Phone v0.9
Content-Length: 0
```


SIP : Proxy

L'utilisateur envoie un message `INVITE` vers son **Proxy SIP**

- indique l'adresse SIP du destinataire (`sip:olivier@upmc.fr`)

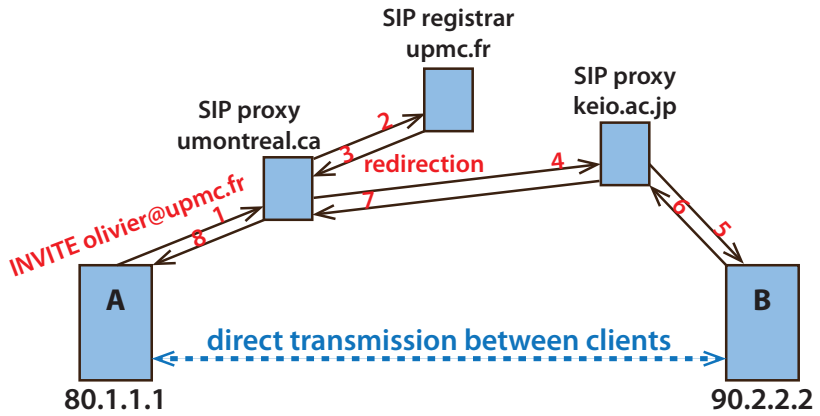
Le serveur Proxy SIP est responsable de l'acheminement des messages SIP

- éventuellement à travers plusieurs proxys.
- l'appelé envoie la réponse à travers le(s) même proxy(s)

➠ ensuite, le proxy de l'appelé renvoie une réponse SIP à l'appelant contenant son l'adresse IP

Le service fournis par le proxy SIP est similaire à celui d'un serveur DNS local

SIP : exemple



CONT : plan du cours 3/5

CONT : plan du cours 3/5

Début du multimédia sur Internet

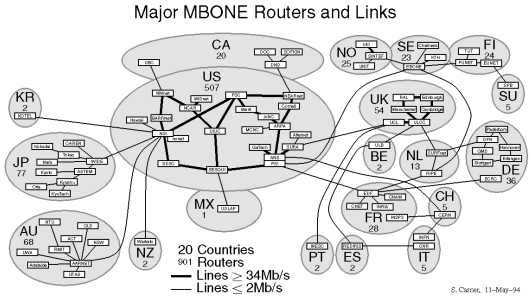
avant 1995 : Internet académique
- début des recherches sur la vidéo

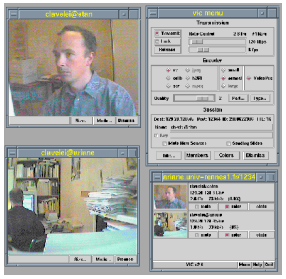
1995 - 2005 : Internet du web
- 1^{re} génération d'applications commerciales (téléchargement progressif via un lecteur dédié)

2005 - 2010 : début de l'Internet de la vidéo
- 2^{me} génération d'applications commerciales avec RTSP, et surtout RTMP (**flash** standard de facto)

2010... : Internet massivement pour la vidéo
- 3^{me} génération d'applications commerciales avec le **HTTP Streaming**...

Recherche sur la vidéo sur Internet - années 1990





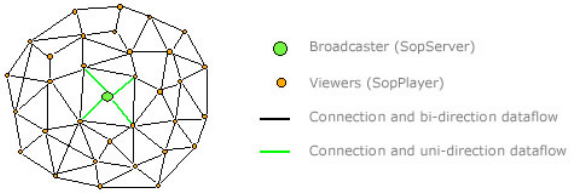
- support de la qualité de service dans le réseau
  - signalisation, réservation de ressources et ordonnancement de paquet (RSVP/IntServ)
  - prioritisation de trafic (DiffServ)
- communications multicast

➠ application cible : **vidéoconférence multi-partenaire**

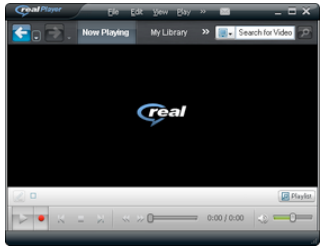
Recherche sur la vidéo sur Internet - années 2000

Difficulté à introduire de nouveaux mécanismes dans les réseaux

- applications adaptatives
- approches P2P
  - multicast applicatif
  - BitTorrent
  - Coolstreaming, SopCast, PPLive, PPStream...



1re génération d'app. vidéo commerciales sur Internet (1)



- lecture vidéo simple
- intégration avec le navigateur web limitée
- protocoles et serveurs propriétaires
- pas assez de contenu disponible sur Internet
- pas assez d'accès Internet à haut débit

➠ **période de l'internet du web (1995-2005)**

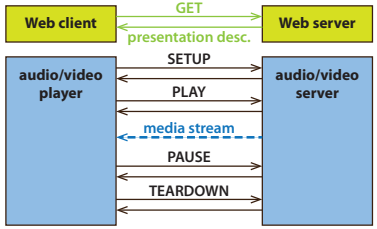
1re génération d'app. vidéo commerciales sur Internet (2)

- téléchargement complet
  - objets récupéré via HTTP puis lecture via *player* dédié
    - ✘ pas de *pipelining*
- téléchargement **progressif**
  - *metafile* récupérée et transmise au *player* dédié
  - le *player* établis une communication directe vers le serveur web
    - ✘ HTTP transfert les données aussi rapidement que possible
    - ✘ transmission importante même si arrêt de la lecture
    - ✘ inadaptation à l'usage de l'audio/vidéo (stop, lecture, accès direct)
    - ✘ surcoût de TCP
    - ✘ pas d'adaptation aux conditions du réseau

2^{me} génération d'app vidéo commerciales sur Internet (1)

Protocoles de streaming dédiés

- protocoles avec un état par session
- séparation des connexion contrôle et données
  - contrôle : lecture, pause, retour arrière, avance rapide...
  - données : UDP ou TCP (mais pas HTTP)
- exemple : RTSP, RTMP...



2^{me} génération d'app vidéo commerciales sur Internet (2)

Flash : technologie principale pour le décodage de la vidéo

- *player* de facto coté client (95% PC, tous les OS, tous les navigateurs)
- environnement de programmation multi-plateforme (Action script)
- contenu multimédia riche (avec bonne intégration au navigateur)
- video FLV : codec Sorenson, VP6 puis H264
- protocole RTMP
  - ✘ problèmes de sécurité, filtrage des ports non standard
  - ✘ servers web plus rentables que serveurs dédiés

Actuellement 2010 ➠

Nouveaux écrans : smart TV, smartphones, tablettes, consoles de jeux...

- nouveaux besoins (encodages multiples : au moins 4 types d'écran)
- nouvelles plateformes où Flash n'est pas forcément disponible (iOS, Android, consoles...)

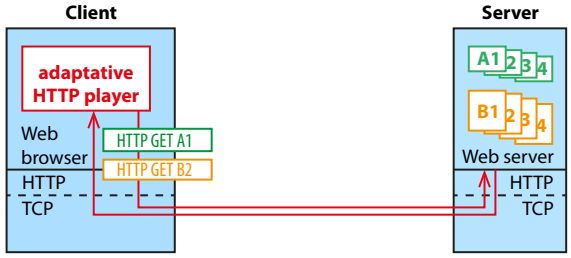
3ème génération : HTTP Streaming

Adaptation de la distribution à l'Internet (plutôt que l'inverse)

- architecture **orientée client** (client avec état, serveur sans état)
  - serveur standard : serveurs web
  - protocole standard : HTTP
  - état de la session et régulation du flux du coté du client
- vidéo **découpée** en multiples *chunks*
  - un *chunk* démarre sur une image de référence
  - téléchargement progressif de chaque *chunk* via HTTP
  - lecture séquentielle des *chunk* ➠ vidéo continue
- adaptation aux différents débits
  ➠ **adaptative HTTP streaming**

CONT : plan du cours 3/5

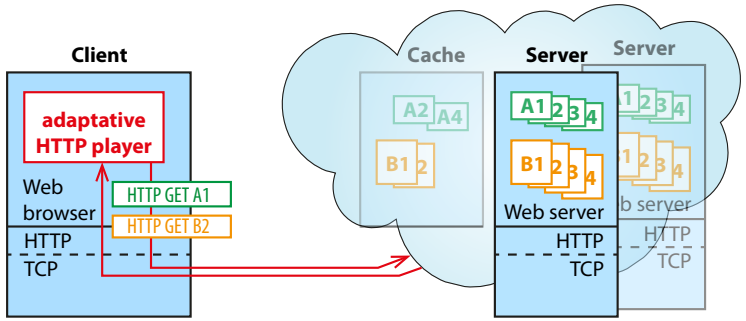
Protocole de découpage multi-chunk pour HTTP



Mécanismes de découpage en multiple *chunk* pour HTTP

- encodage à différents niveaux de qualité
- adaptation par le client : demande du *chunk* avec l'encodage adapté
  ➠ **tous les chunks encodent les mêmes segments de vidéo**

Avantages du streaming HTTP adaptatif



Motivation pour une large adoption

- orienté client ➠ commutation serveur/CDN
- passage des pare-feu et des boîtiers intermédiaires
- réutilise les infrastructure CDN existantes

Exemples de protocoles de streaming HTTP adaptatif

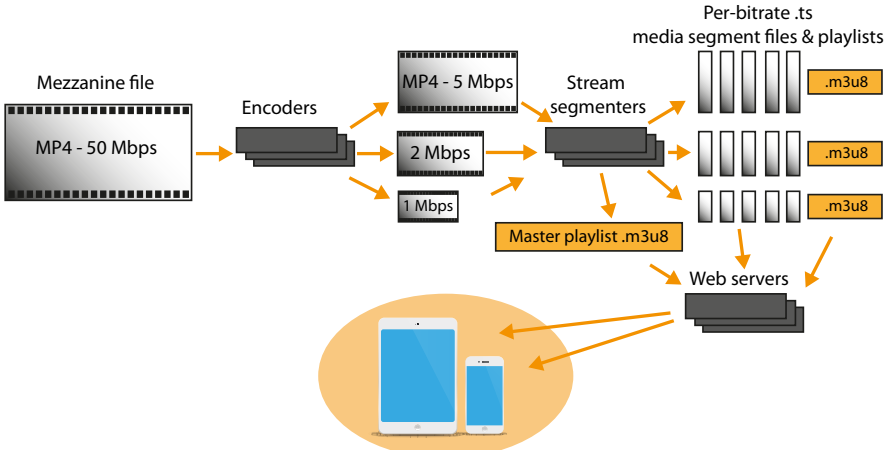
- Apple HLS : HTTP Live Streaming
- Microsoft IIS Smooth Streaming
- Adobe HDS : HTTP Dynamic Streaming
- MPEG DASH : Dynamic Adaptive Streaming over HTTP

Apple HLS (1)

HTTP Live Streaming (2009)

- segmentation du contenu en petits **objets HTTP** standards (10s)
- chargement initial d'une liste de lecture **M3U étendues**
- composants de la solution Apple :
  - **serveurs d'encodage** : H.264 + MP3, HE-AAC ou AC-3 dans un conteneur MPEG transport stream (`.ts`)
  - **serveurs de segmentation** : découpage en petits fragments de taille identique (+ créantion des indexes `.m3u8`)
  - **serveurs de diffusion** : serveur HTTP standard
  - **clients** : après avoir chargé les indexes, doivent pouvoir ré-assembler le flux adéquat
- DRM possibles
- **format publique** (draft IETF informational RFC)
  ➠ large support (la plupart des clients et serveurs multimédia implémentent HLS)

Apple HLS (2)



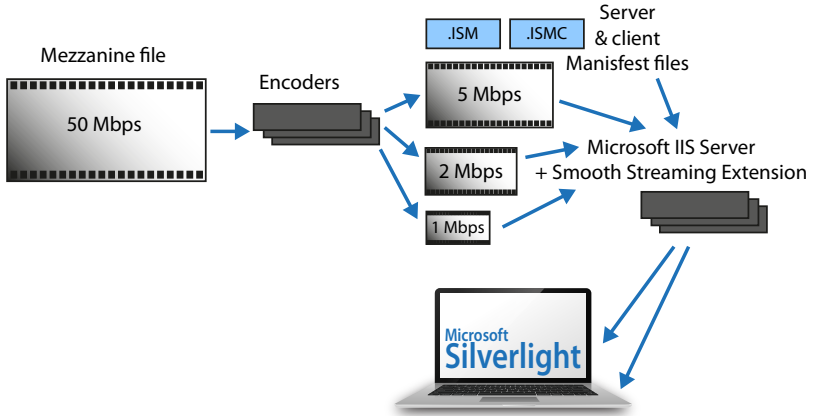
Mécanismes de base :

- récupération des listes de lectures principales et spécifiques
- HTTP GET sur les différents segments `.ts`

Microsoft Smooth Streaming (1)

- vidéo "live" et à la demande avec Silverlight
- composants de la solution Microsoft :
  - **encodeurs** : Microsoft Expression ou autre (VC1, MP4...)
  - **serveur** : Microsoft IIS avec extension Smooth Streaming du IIS media service
  - **clients** : nécessite le player Microsoft Silverlight
- fichiers associés :
  - contenu intégré dans un conteneur ISO base media (MPEG-4 part 12) adapté au streaming (gestion de cache et accès aléatoire)
  - **ISM** : fichier "manifest" décrivant les fichiers média (codecs, débits et temps)
- requêtes de type : `http://video.foo.com/NBA.ism/QualityLevels(400000)/Fragments(video=610275114)`

Microsoft Smooth Streaming (2)



Mécanismes de base :

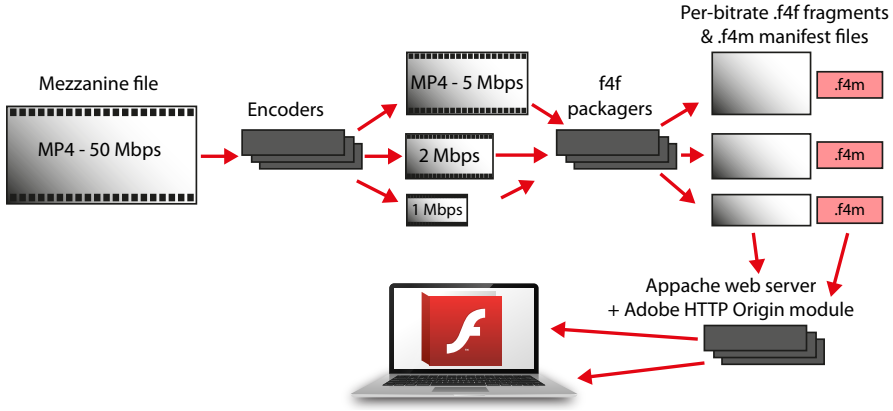
- récupération du fichier "manifest" pour le client (`.ismc`)
- HTTP GET sur les différents fragments (url avec les informations de débit et de temps)

Adobe HDS (1)

HTTP Dynamic Streaming

- vidéo "live" et à la demande avec un **client Flash via HTTP**
- composants de la solution Adobe :
  - **File Packager** : gère la fragmentation (6s) et les fichiers F4F
  - **HTTP Orgin Module** : module pour les serveurs HTTP Apache
  - **OSMF Media Players** (inclus dans les lecteurs Flash)
- fichiers associés :
  - **F4F** : contenu FLV ou MP4 intégré dans un conteneur ISO base media (MPEG-4 part 12) adapté au streaming (gestion de cache et accès aléatoire)
  - **F4M** : fichier "manifest" décrivant les fichiers F4F assemblés (codecs, résolutions, débits et DRM)
- requêtes de type : `http://www.example.com/media/http-dynamic-StreamingSeg1-Frag1`

Adobe HDS (2)



Mécanismes de base :

- récupération du fichier "manifest" `.f4m`
- HTTP GET sur les différents fragments (url avec les informations de débit et de temps)

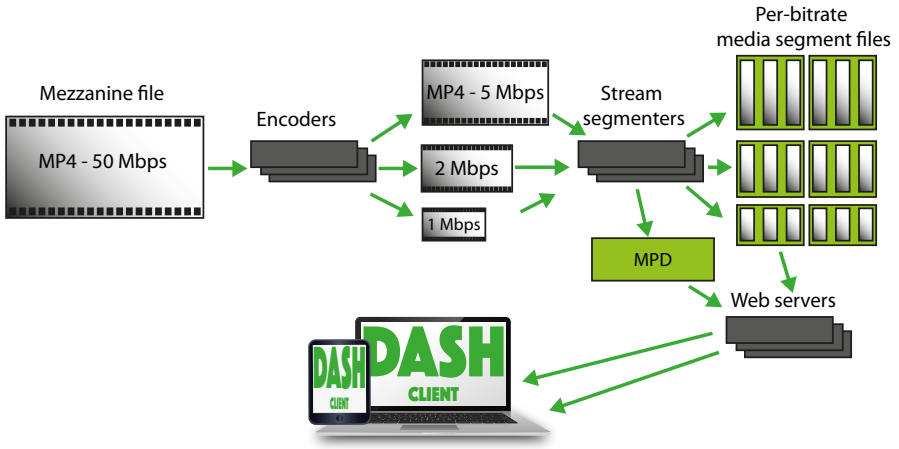
CONT : plan du cours 3/5

MPEG DASH ISO/IEC 23009-1

MPEG's Dynamic Adaptative Streaming over HTTP (2012)

- motivation
  - HTTP est nécessaire (caches/CDN, NAT/Firewalls, adaptation à l'état du réseau...)
  - nombreux types d'équipements
  - nombreuses solutions de streaming sur HTTP
- principes
  - transmission de petits morceaux de vidéo via HTTP
  - pas de modification de l'infrastructure de distribution
  - contrôle et réassemblage par le client
  - adaptation au client et aux conditions du réseau
- cîble : OTT sur tout équipements

DASH : architecture



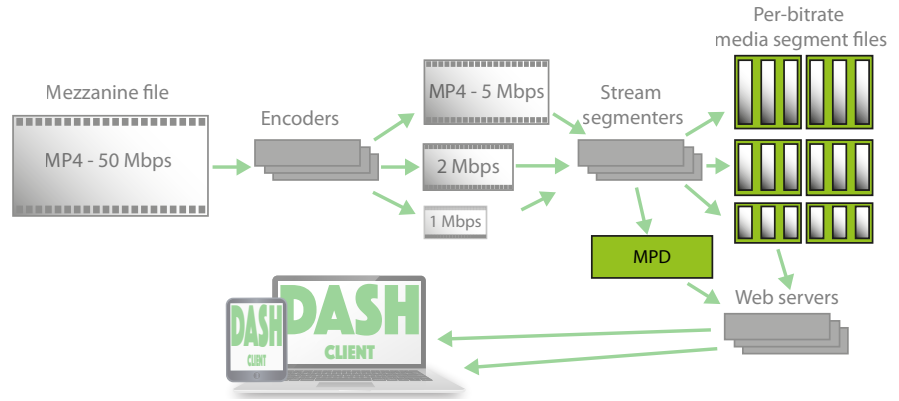
Mécanismes de base :

- récupération du fichier "manifest" MPD
- HTTP GET sur les différents segments nécessaires

DASH : principes de base

- ce que DASH **n'est pas** :
  - protocole, codec, middleware, spécification du client...
- DASH est un catalyseur :
  - définit les **formats** pour permettre une diffusion performante sur Internet
- DASH permet :
  - la réutilisation des technologies existantes (conteneurs, codecs, DRM...)
  - l'utilisation de l'infrastructure Web (cache/CDN)
  - l'amélioration de l'expérience de l'utilisateur (démarrage rapide, limitation du rebuffering)
  - l'adaptation aux besoins de l'utilisateurs, la capacité de ses équipements et à l'état du réseau
  - le contrôle au niveau des clients (différenciation)
  - la commutation de qualité (switching) imperceptible
  - les flux "live" ou pré-enregistré (DVD-like)
  - l'interopérabilité entre flux
  - l'intégration des techniques existantes

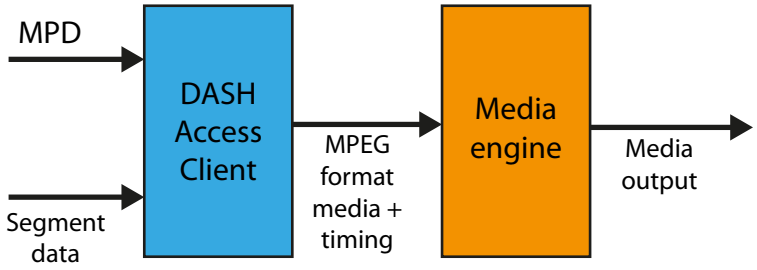
DASH : ce qui est spécifié



- présentation des données sur le serveur HTTP
- fichier MPD (Media Presentation Description)
- structure des URL pour retrouver les segments via HTTP

DASH : classification des informations

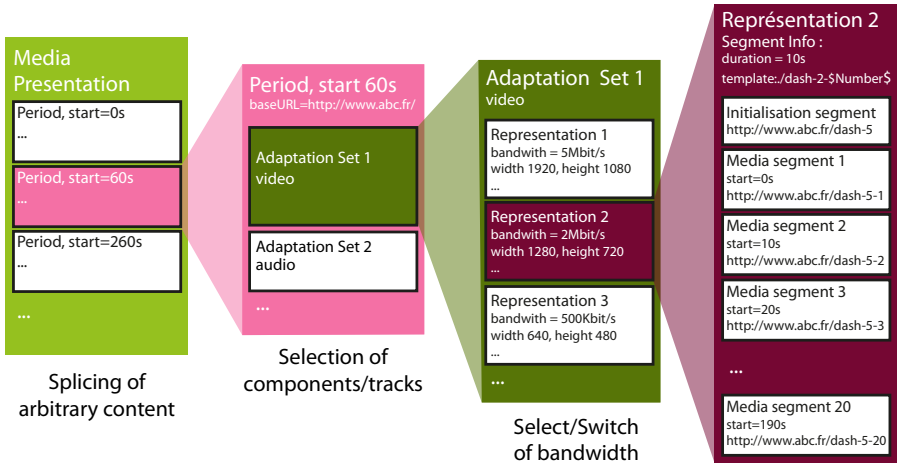
- MPD et informations indexées pour le client DASH
  - cœur des spécifications DASH
- initialisation et flux de contenu pour le lecteur
  - réutilisation des containers existants



DASH : MPD

- information pour sélectionner les éléments des différentes représentations :
  - débit, résolution, codec, language, DRM...
- informations temporelles et d'accès :
  - URL et position dans chaque segment accessible (byte-range)
  - disponibilité des segments (début/fin entemps absolu)
  - temps de présentation initial et durée relative des segments
  - délais recommandé de lecture (live)
- information de découpage et de commutation entre les représentations

DASH : Stucture MPD



DASH : exemple de MPD pour vidéo segmentée

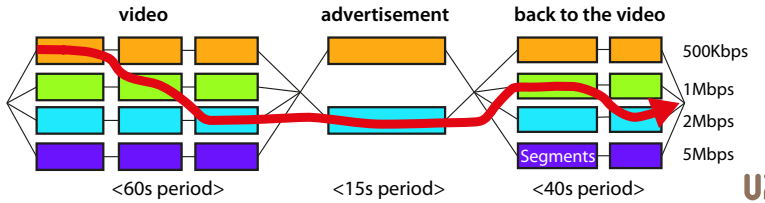
```
<MPD xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011" profiles="urn:mpeg:dash:profile:isoff-main:2011" type="static" mediaPresent
  <BaseURL>http://www-itec.uni-klu.ac.at/ftp/datasets/mmsys12/BigBuckBunny/bunny_10s/</BaseURL>
  <Period start="PT0S">
    <AdaptationSet bitstreamSwitching="true">
      <Representation id="0" codecs="avc1" mimeType="video/mp4" width="320" height="240" startWithSAP="1" bandwidth="45373">
        <SegmentBase>
          <Initialization sourceURL="bunny_10s_50kbit/bunny_50kbit_dash.mp4"/>
        </SegmentBase>
        <SegmentList duration="10">
          <SegmentURL media="bunny_10s_50kbit/bunny_10s1.m4s"/>
          <SegmentURL media="bunny_10s_50kbit/bunny_10s2.m4s"/>
          <SegmentURL media="bunny_10s_50kbit/bunny_10s3.m4s"/>
            ...
          <SegmentURL media="bunny_10s_50kbit/bunny_10s60.m4s"/>
        </SegmentList>
      </Representation>
      <Representation id="1" codecs="avc1" mimeType="video/mp4" width="320" height="240" startWithSAP="1" bandwidth="88482">
        <SegmentBase>
          <Initialization sourceURL="bunny_10s_100kbit/bunny_100kbit_dash.mp4"/>
        </SegmentBase><SegmentList duration="10">
          <SegmentURL media="bunny_10s_100kbit/bunny_10s1.m4s"/>
          <SegmentURL media="bunny_10s_100kbit/bunny_10s2.m4s"/>
            ...
        ...
      <Representation id="19" codecs="avc1" mimeType="video/mp4" width="1920" height="1080" startWithSAP="1" bandwidth="3792491">
        <SegmentBase>
          <Initialization sourceURL="bunny_10s_8000kbit/bunny_8000kbit_dash.mp4"/>
        </SegmentBase>
        <SegmentList duration="10">
          <SegmentURL media="bunny_10s_8000kbit/bunny_10s1.m4s"/>
          <SegmentURL media="bunny_10s_8000kbit/bunny_10s2.m4s"/>
            ...
          <SegmentURL media="bunny_10s_8000kbit/bunny_10s60.m4s"/>
        </SegmentList>
```


DASH : exemple de MPD pour vidéo non segmentée

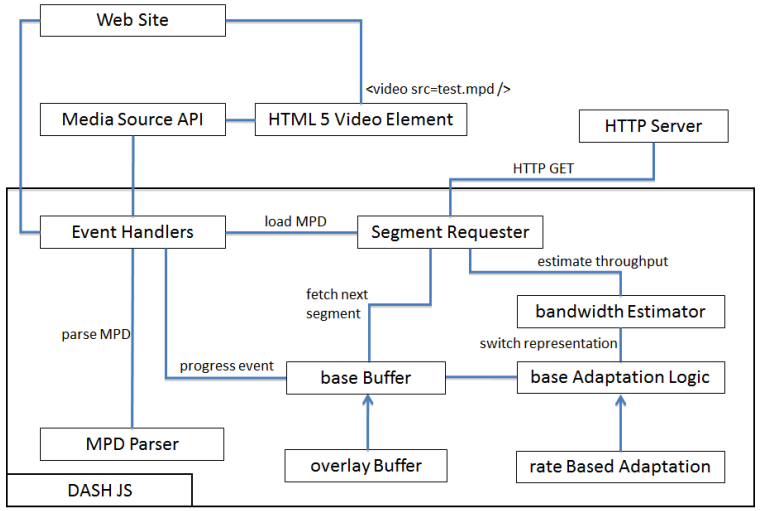
```
<MPD xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011" profiles="urn:mpeg:dash:profile:isoff-main:2011" type="static" me
  <BaseURL>http://www-itec.uni-klu.ac.at/ftp/datasets/mmsys12/BigBuckBunny/bunny_10s/</BaseURL>
  <Period start="PT0S">
    <AdaptationSet bitstreamSwitching="true">
      <Representation id="0" codecs="avc1" mimeType="video/mp4" width="320" height="240" startWithSAP="1" bandwidth="4537
        <SegmentBase>
          <Initialization sourceURL="bunny_10s_50kbit/bunny_50kbit_dashNonSeg.mp4" range="0-864"/>
        </SegmentBase>
        <SegmentList duration="10">
          <SegmentURL media="bunny_10s_50kbit/bunny_50kbit_dashNonSeg.mp4" mediaRange="865-57231"/>
          <SegmentURL media="bunny_10s_50kbit/bunny_50kbit_dashNonSeg.mp4" mediaRange="57232-114771"/>
          <SegmentURL media="bunny_10s_50kbit/bunny_50kbit_dashNonSeg.mp4" mediaRange="114772-171506"/>
            ...
          <SegmentURL media="bunny_10s_50kbit/bunny_50kbit_dashNonSeg.mp4" mediaRange="3344628-3383017"/>
        </SegmentList>
      </Representation>
      <Representation id="1" codecs="avc1" mimeType="video/mp4" width="320" height="240" startWithSAP="1" bandwidth="8848
        <SegmentBase>
          <Initialization sourceURL="bunny_10s_100kbit/bunny_100kbit_dashNonSeg.mp4" range="0-866"/>
        </SegmentBase><SegmentList duration="10">
          <SegmentURL media="bunny_10s_100kbit/bunny_100kbit_dashNonSeg.mp4" mediaRange="867-112515"/>
          <SegmentURL media="bunny_10s_100kbit/bunny_100kbit_dashNonSeg.mp4" mediaRange="112516-229893"/>
            ...
        ...
      <Representation id="19" codecs="avc1" mimeType="video/mp4" width="1920" height="1080" startWithSAP="1" bandwidth="3
        <SegmentBase>
          <Initialization sourceURL="bunny_10s_8000kbit/bunny_8000kbit_dashNonSeg.mp4" range="0-870"/>
        </SegmentBase>
        <SegmentList duration="10">
          <SegmentURL media="bunny_10s_8000kbit/bunny_8000kbit_dashNonSeg.mp4" mediaRange="871-6888335"/>
          <SegmentURL media="bunny_10s_8000kbit/bunny_8000kbit_dashNonSeg.mp4" mediaRange="6888336-12889377"/>
          <SegmentURL media="bunny_10s_8000kbit/bunny_8000kbit_dashNonSeg.mp4" mediaRange="281330526-282757919"/>
        </SegmentList>
```

DASH : mécanismes coté client

- processus client :
  - téléchargement du **fichier MPD**
  - téléchargement **segment par segment** en fonction de la lecture
  - **détermination du débit disponible**
- facteur de choix pour la représentation :
  - état des tampons **mémoires**
  - état du **réseau**
  - état de **l'équipement**
  - changement de résolution de l'utilisateur



DASH : exemple de client



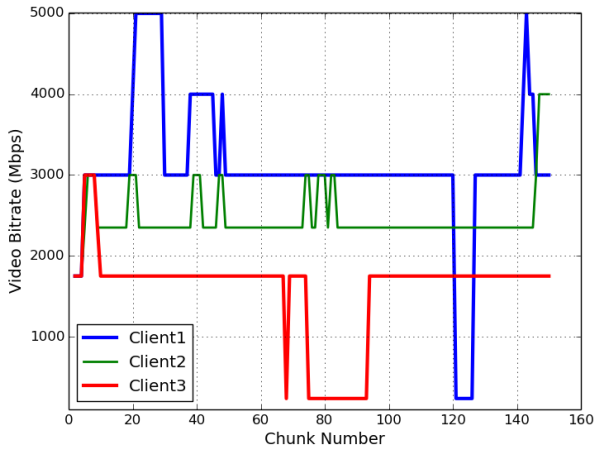
B. Rainer, S. Lederer, C. Müller & C. Timmerer, A Seamless Web Integration of Adaptive HTTP Streaming.

In Proc. European Signal Processing Conference 2012, Bucharest.

DASH : commutation et segments

- supports pour la commutation
  - **alignement des segments** (permet un décodage non recouvrant entre différentes représentations)
  - **point d'accès du flux** (stream access points - SAP - position dans un segment où la commutation est préférable)
  - **encodages adaptés** à la concaténation de différentes présentation (structures similaires)
- formats des segments
  - **ISO base media FF** et **MPEG-2 TS**
  - possibilité d'intégrer d'autres formats
  - indépendant des codec

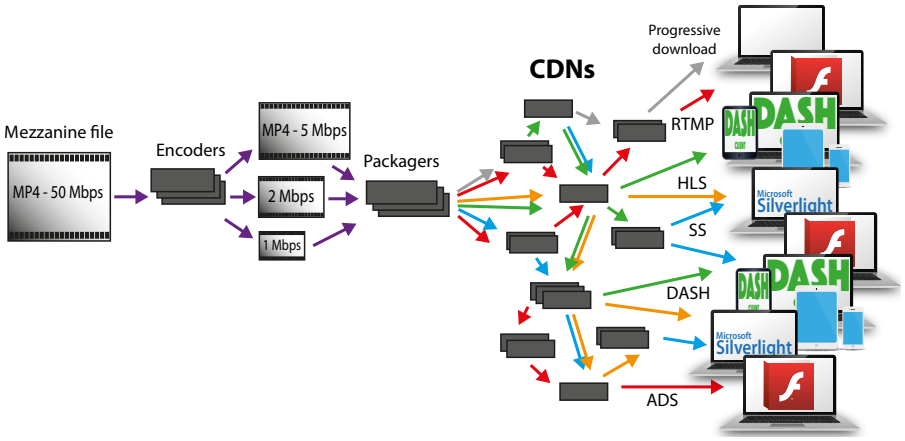
DASH : stabilité



Courbe réalisée par Rabee Mustapha Abuteir (mustapha.abuteir@lip6.fr)

CONT : plan du cours 3/5

Distribution vidéo actuelle



Trafic de l'Internet video

A world full of elephants

- croissance du traffic ➠ 95% vidéo !
  - vidéo : X 100
  - autres applications : X 10
- le segment HTTP (chunk) est le nouveau datagramme
  - support de HTTP par tout les équipements du réseau
- garantir la QoE
  - pas de support universel de la QoS dans l'Internet
  - adaptation aux besoin par mécanismes adaptatifs multi-débits
  - **le CDN est l'élément clé pour optimiser la performance**