1 OBJECTIF

Ce document est rédigé pour fixer le cadre des *travaux pratiques – application réseau* pour l'année 3IF de l'INSA. L'objectif premier est de permettre aux étudiants d'acquérir et d'exercer les connaissances nécessaires pour la réalisation d'une application basée sur INET.

2 SYSTEME DE VIDEO A LA DEMANDE : DESCRIPTION GENERALE

Un *système de vidéo à la demande IP* est défini comme étant un système intégrant :

- un ou plusieurs serveurs connectés à des périphériques d'acquisition/stockage d'images ; ces images sont partagées sous format de flux multimédias sur un réseau à des clients ayant le droit de les utiliser ;
- des clients interconnectés à travers un réseau ainsi que leurs logiciels, accédant aux flux multimédias partagés par les serveurs. Les flux sont fournis d'une façon concurrente et indépendante, en utilisant l'architecture INET.
- un réseau permettant le transfert des flux multimédias

L'architecture très générale d'un tel système est la suivante :

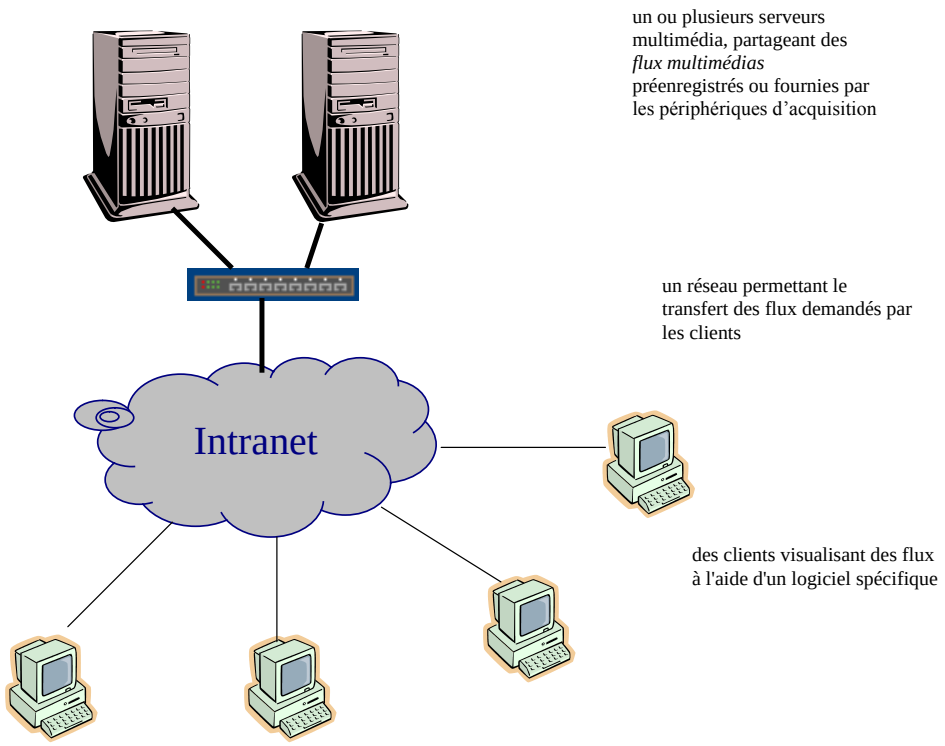

Figure 1 Architecture d'un système de vidéo à la demande IP

Dans un scénario habituel, chaque client potentiel a la possibilité de choisir et de visualiser un ou plusieurs flux en utilisant un catalogue des flux disponibles dans le système (partagés par les serveurs).

Chaque flux peut être visualisé par plusieurs clients simultanément, et chaque client peut visualiser plusieurs flux en parallèle.

3 ACCES AU CATALOGUE

Chaque serveur envoie le contenu de son catalogue (la description des flux qu'il partage, format fichier texte) à la demande, en utilisant http. Il envoie aussi le contenu de ce catalogue en utilisant des messages multicast à des intervalles de temps préétablis.

3.1 STRUCTURE D'UN FICHIER CATALOGUE

ServerAddress: *server_address* CRLF

ServerPort: *server_port* CRLF

Object ID=*id* name=*name* type=*type* address=*address* port=*port* protocol=*protocol* ips=*ips* CRLF

…

Object ID=*id* name=*name* type=*type* address=*address* port=*port* protocol=*protocol* ips=*ips* CRLF

CRLF

Le paramètre *protocol* peut avoir l'une des valeurs : TCP_PULL, TCP_PUSH, UDP_PULL, UDP_PUSH, MCAST_PUSH.

Le paramètre *ips* représente le nombre d'images par seconde.

Le paramètre *type* représente le format d'images transférées, et peut avoir l'une des valeurs : BMP, JPEG. (*Votre projet doit prendre en compte au moins le format jpeg.*)

La durée de vie d'un catalogue est infinie (il ne change pas dans le temps).

Exemple :

ServerAddress: *134.213.23.12* CRLF

ServerPort: *8080* CRLF

Object ID=*1* name=*video1* type=*BMP* address=*134.213.23.12* port=*8012* protocol=*TCP_PULL* ips=*0.5* CRLF

Object ID=*2* name=*video2* type=*JPEG* address=*224.100.100.100* port=*2001* protocol=*MCAST_PUSH* ips=*2* CRLF

Object ID=*3* name=*video3* type=*BMP* address=*134.213.23.12* port=*8013* protocol=*UDP_PUSH* ips=*2* CRLF

CRLF

3.2 COMMUNICATION HTTP POUR RECEVOIR LE CATALOGUE

Le serveur écoute sur un port prédéfini. Le client lui envoie une requête http (GET) pour demander le fichier catalogue.txt (défini dans la section précédente).

Le serveur renvoie comme réponse le fichier catalogue.

4 ACCES A UN FLUX MULTIMEDIA

Pour simplifier le problème, on suppose que le flux multimédia ne contient que des images, en format BMP ou JPEG. Le serveur envoie les images l'une après l'autre, en ajoutant des informations concernant leur position dans le flux.

La communication se fait en plusieurs modes :

- *Pull* : le client envoie une requête pour chaque image
- *Push* : le serveur envoie image après image vers le client, sans attendre qu'il les demande

Les protocoles utilisés : TCP, UDP, IP (Multicast)

4.1 TCP-PULL

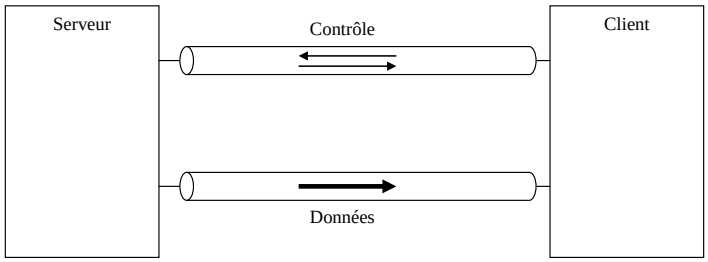

Figure 2

Le client se connecte au serveur en utilisant TCP pour réaliser une connexion de contrôle. Le serveur est défini par le paramètre *address* d'un objet dans le catalogue. Le port utilisé par le serveur pour l'écoute est aussi spécifié dans le catalogue (le paramètre *port*).

Le client envoie une commande (chaîne de caractères) pour l'objet *ID* ; la commande doit contenir le port utilisé par le client pour recevoir les données ; la commande se termine avec les caractères CR LF ;

```
GET ID CRLF
LISTEN_PORT client_port CRLF
CRLF
```

Le serveur se connecte (TCP) sur le port *client_port* du client

Le client envoie une commande pour recevoir une image (sur le canal de contrôle) :

```
GET IMAGE_ID CRLF
CRLF
```

Si *IMAGE_ID* = -1 le serveur envoi l'image en cours.

Le serveur envoie l'image sur le canal de données ; le format des messages :

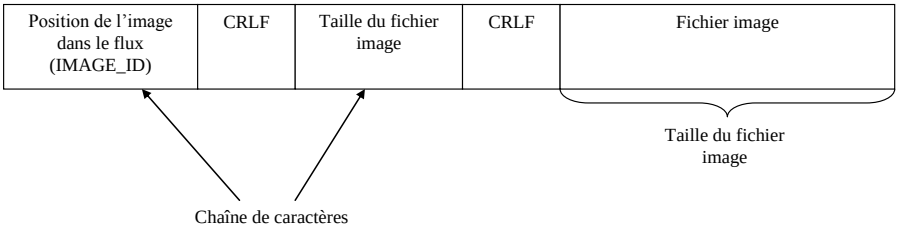

Figure 3

Pour finir la connexion, le client envoie le message END sur la connexion de contrôle :

```
END CRLF
CRLF
```

Dans ce cas, les deux connexions (contrôle et données) sont fermées par le serveur.

4.2 TCP-PUSH

Le client se connecte au serveur en utilisant TCP pour réaliser une connexion de contrôle.

Le client envoie une commande (chaîne de caractères) pour l'objet *ID* ; la commande doit contenir le port utilisé par le client pour recevoir les données ; la commande se termine avec les caractères CR LF ;

```
GET ID CRLF
LISTEN_PORT listen_port CRLF
CRLF
```

Le serveur se connecte (TCP) sur le port *listen_port* du client

Le client demande au serveur d'envoyer des images (sur le canal de contrôle) :

```
START CRLF
CRLF
```

Le serveur commence à envoyer des images sur le canal de données (au même format que 4.1, voir Figure 3)

Le client peut demander au serveur de faire une pause (sur le canal de contrôle) :

```
PAUSE CRLF
CRLF
```

Le démarrage après la pause se fait avec la commande START.

Pour finir la connexion, le client envoie le message END sur la chaîne de contrôle :

```
END CRLF
CRLF
```

Dans ce cas, les deux connexions (contrôle et données) sont fermées.

4.3 UDP-PULL

Les deux canaux de communication (contrôle et données) utilisent UDP.

Dans ce cas le fichier image est transféré en utilisant plusieurs fragments.

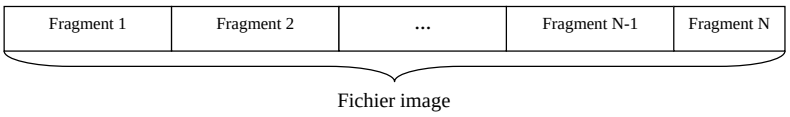

Figure 4 Fragmentation d'un fichier image[1]

Le client envoie une commande (chaîne de caractères) pour l'objet *ID* ; la commande doit contenir le port utilisé par le client pour recevoir les données ; la commande se termine avec les caractères CR LF ;

```
GET ID CRLF
LISTEN_PORT port CRLF
FRAGMENT_SIZE taille CRLF
CRLF
```

Le client envoie une commande pour recevoir une image ; IMAGE_ID représente la position de l'image dans le flux ; si IMAGE_ID = -1, le serveur envoie l'image en cours : FRAGMENT_SIZE représente la taille de fragments désirée par le client.

```
GET IMAGE_ID CRLF
CRLF
```

[1] Le dernier fragment n'a pas obligatoirement la même taille que les autres.

Le serveur envoie l'image en plusieurs fragments ; chaque fragment a un identifiant correspondant à la position du fragment dans le fichier image ; le format d'un paquet UDP contenant un fragment :

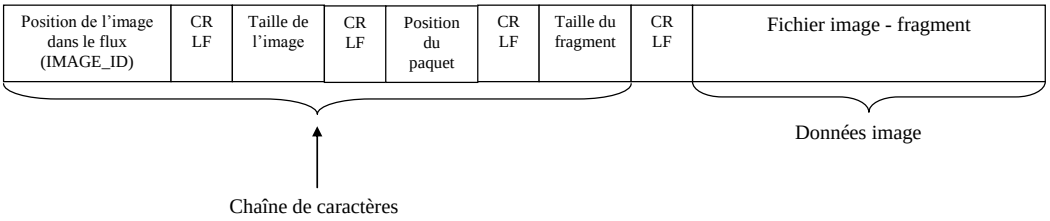

Figure 5 Message contenant un fragment d'image

Le paramètre *position du paquet* représente l'adresse du premier octet du fragment dans l'image. Une image est affichée après que tous ces fragments sont réceptionnés. Il est possible qu'un certain nombre de fragments soit perdus. Dans ce cas on suppose qu'une image est complètement réceptionnée quand le dernier fragment reçu appartient à une autre image.

Pour fermer la connexion, le client envoie le message END :

```
END CRLF
CRLF
```

4.4 UDP-PUSH

Le client envoie une commande (chaîne de caractères) pour l'objet *ID* ; la commande doit contenir le port utilisé par le client pour recevoir les données ; la commande se termine avec les caractères CR LF ;

```
GET ID CRLF
LISTEN_PORT port CRLF
FRAGMENT_SIZE taille CRLF
CRLF
```

Le client demande au serveur d'envoyer des images :

```
START CRLF
CRLF
```

Le serveur commence à envoyer des images sur le port de données du client (le format : Figure 5)

Le client peut demander au serveur de faire une pause (sur le port de contrôle) :

```
PAUSE CRLF
CRLF
```

Pour finir la communication, le client envoie le message END sur la connexion de contrôle :

```
END CRLF
CRLF
```

Le serveur ne peut pas toujours savoir si le client utilise encore un flux qu'il à demandé (le message END peut être perdu). Pour savoir si le client utilise le flux, le serveur attend des messages de la part du client à des intervalles de temps inférieurs à 60 secondes. Ces messages (envoyés par le client) ont le format :

```
ALIVE ID CRLF
LISTEN_PORT port CRLF
CRLF
```

4.5 MCAST_PUSH

Dans ce cas le client ne peut pas envoyer des messages au serveur. Le client se connecte à l'adresse IP multi-destination et reçois des fragments d'image de la même manière que la communication UDP.

4.6 MESSAGES ERRONES

Si le serveur reçoit un message erroné, il n'envoie pas de message d'erreur.

5 DESCRIPTION DU CLIENT

Les fonctionnalités de base du logiciel client :

- visualiser le contenu disponible sur un serveur accessible (catalogue) ;
- permettre d'ajouter l'adresse du serveur fournisseur de catalogues ;
- permettre la sélection d'un flux, la visualisation de ce flux (en utilisant la méthode de communication adéquate), la fermeture d'un flux, l'arrêt sur une image ;
- permettre la visualisation de plusieurs flux en parallèle (optionnel) ;

6 DESCRIPTION DU SERVEUR

Les fonctionnalités de base du logiciel serveur :

- fournir le contenu disponible via un fichier catalogue ;
- envoyer les images constituant un flux, en utilisant la méthode de communication adéquate ;
- permettre la visualisation de plusieurs flux en parallèle (optionnel) ;

7 RESULTATS ATTENDUS

A. Manuel utilisateur pour l'application serveur (fichier PDF, HTML ou TXT) ;

B. Code source du serveur (fichier ZIP), exécutable et instructions pour la construction de l'exécutable (fichier TXT).

8 NOTATION

Les points suivants seront évalués :

Manuel utilisateur (**2p**)
Facilité d'installation, jeu de données de test (**2p**)
Accès au catalogue (http – **4p**)
Partage d'un flux :
TCP Push (**4p**)
TCP Pull (Optionnel **– 1p**)
UDP Push (Optionnel **– 1p**)
UDP Pull (**4p**)
IP Mcast (Optionnel **– 2p**)