TP 3 de programmation Android

Résumé

L'objectif de ce troisième TP est d'étudier les étapes de la lecture de vidéo en "streaming".
Une adresse utile : `http://developer.android.com/guide/components/index.html`
En local : `/mnt/n7fs/ens/tp_cregut/android-sdk-linux_86/docs/offline.html`

1 Media Player

Terminez le TP 2 au moins jusqu'en 3.4 (lecture vidéo avec MediaPlayer et cycle de vie). Vous avez a priori jusqu'ici réalisé les étapes suivantes :

- Transfert de vidéo sur la carte SD.
- Lecture de vidéo par Intent.
- Transformation du Layout (boutons play/pause et SeekBar).
- Association de Listeners aux boutons pour lire la vidéo.
- Etude du cycle de vie de l'application. Lire à ce sujet la page de la doc (`http://developer.android.com/reference/android/app/Activity.html`) et le schéma du cycle de vie de l'application. Grâce au LogCat et aux différentes fonctions **onCreate()**, **onPause()**, **onResume()**, étudiez ce qu'il se passe lorsque vous démarrez l'application, la quittez, y revenez etc.

2 Streaming HTTP

Dans la version précédente, on ne peut lire que des vidéos présentes en local sur la carte SD virtuelle. Nous allons maintenant faire en sorte que le lecteur puisse ouvrir des vidéos accessibles à distance, via le protocole HTTP ou le protocole de streaming RTSP.

2.1 Configuration

Pour que votre application puisse accéder à Internet, il faut modifier la variable d'environnement http_proxy. Pour cela, commencez par identifier votre shell. Placez-vous dans votre répertoire "home/login", et dans un terminal, tapez **echo $SHELL**

- Si le résultat est **/bin/bash**, faites **gedit .bashrc** et ajoutez : **export http_proxy=proxy.enseeiht.fr :3128**
- Si le résultat est **/bin/csh**, faites **gedit .cshrc** et ajoutez : **setenv http_proxy proxy.enseeiht.fr :3128**

Dans le fichier AndroidManifest.xml, ajoutez une **uses-permission** correspondant à internet.

2.2 Etude des états

Modifiez enfin votre activité afin qu'elle puisse, selon ce que vous tapez dans le composant EditText, récupérer la vidéo (mediaPlayer.setDataSource(...)) soit sur internet (si le texte commence par www), soit sur la carte SD.

Afin d'implémenter ce filtrage, une solution (parmi d'autres) est d'utiliser : **url.startsWith("http ://www")**.

Mesurez enfin grâce au LogCat les temps de réponse aux différentes instructions du MediaPlayer, et en particulier celui de la commande **mediaPlayer.prepare()**. Pourquoi la transition vers l'état **prepare** dure si longtemps ?

3 Streaming RTSP/RTP

Le protocole RTSP offre des fonctionnalités de streaming plus avancées que le HTTP : accès alétoire fin, contrôle plus fin du débit, transport séparé des différentes pistes média (e.g. audio et vidéo) avec la possibilité de les synchroniser à la réception etc. Il est notamment intéressant d'étudier les échanges entre le serveur diffusant la vidéo et le client qui la lit. Pour cela, nous allons simuler au niveau local ces interactions. Trois terminaux assureront les différents rôles :

- Un client qui cherche à lire une vidéo
- Un serveur capable de la diffuser
- Un pilote pour commander ou administrer le serveur

Commencez par récupérer l'adresse IP de votre machine (commande **ifconfig** dans un terminal, chercher "inet adress"). Vous pouvez éventuellement utiliser **localhost** : 127.0.0.1

Ensuite effectuez les étapes suivantes :

- Terminal 1 (serveur) : `vlc --intf telnet --rtsp-host ip_de_la_machine:8554` (remplacer l'ip par votre ip). Cela démarre le serveur vlc.
- Terminal 2 (pilote) : `telnet localhost 4212` (mot de passe : admin), puis `new test vod enabled` et `setup test input /home/login/chemin_de_la_video/samplevideo.3gp` (en remplaçant le chemin par votre chemin). Cela va lancer la diffusion.
- Terminal 3 (client) : `vlc --extraintf=rtsp:logger --verbose=2 rtsp://ip_de_la_machine:8554/test 2>vlc-log.txt` (en remplaçant l'ip par votre ip). Cela va lancer la lecture, et rediriger la sortie vers le fichier vlc-log.txt.

Pour vérifier que VLC a bien ouvert les ports et est en écoute sur ces ports, vous pouvez utiliser la commande `netstat -pl | grep vlc`.

Pendant la transmission, des messages sont échangés entre le client et le serveur. Ceux-ci utilisent les protocoles :

- RTP (Realtime Transport Protocol), utilisant lui-même UDP, pour le transport des données multimédias (les paquets audio ou vidéo compressées)
- RTSP pour la signalisation (c'est à dire les commandes play/pause etc.)
- RTCP (Realtime Transport Control Protocol) pour l'échange de statistiques de transmission. Les statistiques RTCP rassemblent pour une connexion média des informations telles que le nombre d'octets et de paquets transmis, le nombre de paquets perdus, la gigue, et le Round-Trip delay (temps d'aller-retour, très utile en visioconférence). Une application peut utiliser ces informations pour gérer la QoS, pour contrôler ou limiter le débit, changer de codec ...

Lancez la vidéo, et effectuez les actions suivantes : pause, play, accès aléatoire. Fermez ensuite vlc, et étudiez les échanges entre le client et le serveur en faisant : `grep rtsp vlc-log.txt`. Essayez en particulier de répondre aux questions suivantes :

- Identifier le nombre de pistes (réponse à la requête Describe)
- Identifier l'encodage de chaque piste.
- Identifier la réponse à l'accès aléatoire (requête Play)
- Identifier le port de réception des flux pour chaque piste (réponse à Setup). Vérifiez également avec netstat les couples de ports RTP/RTCP ouverts par le client VLC, un couple pour chaque piste.

Des informations supplémentaires sur ces échanges peuvent être trouvées sur : `http://camss65.free.fr/projet/projets%20S9/TP_Multimedia/TP1_camille_manano.pdf`

De nombreuses vidéos l'usage des terminaux mobiles sont acessibles sur YouTube en RTSP : `http://m.youtube.com/watch?hl=en&client=mv-google`.

Malheureusement, le pare-feu de l'ENSEEIHT nous empêche d'y accéder.