Programmation de mobiles
Projet

2SN - Parcours R et T

Katia Jaffrès-Runser
Quentin Bailleul

2023-2024

Brève description L'objectif principal du projet est de faire communiquer deux smartphones Android pour qu'ils soient en mesure d'échanger des informations via une communication Bluetooth. Ce sujet vous guide pour mettre en place le socket via une procédure de connexion et l'utiliser pour échanger des données.

Déroulement Vous disposez de 3 séances de projet encadrées pour effectuer ce projet, qui sont précédées de 2 séances de TP. Ces 5 séances sont suivies d'une séance de recette où chaque groupe présente le travail réalisé. Le projet se fait en binôme. Nous vous prêtons pendant toute sa durée un smartphone Android et un câble USB par personne. Merci d'en prendre soin et de nous le retourner lors de la séance de recette.

Ressources Vous disposez d'un cours et de l'expérience du TP pour mettre en oeuvre les principaux outils (création des Activity, Intent, Threads, Handler, ...) nécessaires pour mener à bien ce projet. Le site principal pour les développeurs d'application Android : `https://developer.android.com`.

Test Attention : pour tester votre application, il faudra la déployer sur deux smartphones physiques fournis (pas d'émulateur). Pour que vous puissiez le faire, il faut que le système Android soit paramétré en mode *debug USB*. Pour cela, allez dans les paramètres, puis paramètre du système et appuyez 7 fois sur le numéro de build. Il s'affichera ensuite un message pour vous signaler que vous êtes maintenant développeur Android ;-) Ensuite, revenez en arrière, et vous aurez un nouveau menu : Options développeurs. Dans ce dernier, activez le mode Débogage USB.

Évaluation La dernière séance est dédiée à l'évaluation du projet. La note (14 pts) se décompose comme suit :

[6pts] Le code commenté, rendu le jour de la dernière séance de TP sur Moodle.

[2pts] Un rapport d'au maximum 3 pages qui décrit le travail réalisé, l'architecture de l'application (les activités et les classes implémentées) et le fonctionnement de l'application. La date de rendu du rapport est précisée sur Moodle.

[3pts] Une recette qui se fera en binôme lors de la dernière séance de TP. Le chargé de TP organisera le passage des groupes dans la séance. Il leur demandera une explication du travail réalisé. À la suite de cette explication, le binôme devra réaliser une démonstration du travail effectué. Il est demandé de présenter une version du projet **qui fonctionne**.

[3pts] Réponse aux questions de l'intervenant.

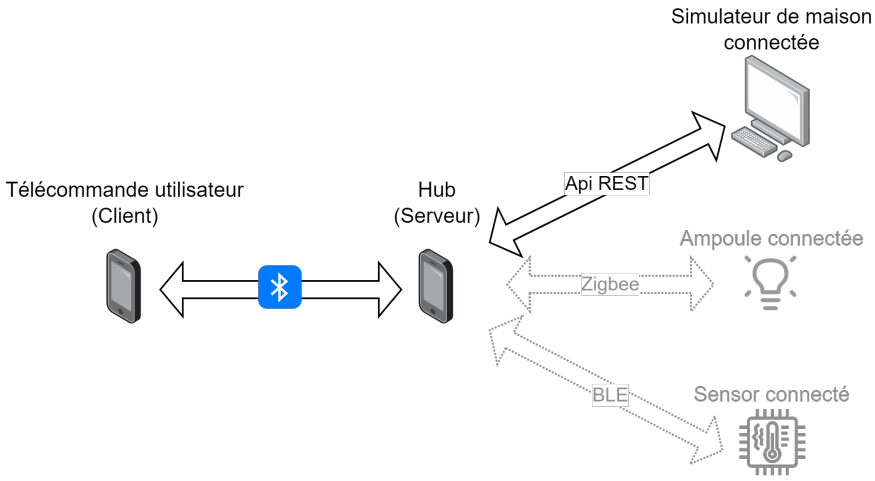

Figure 1: Architecture

1 Contrôle d'une maison connectée à l'aide d'un hub Bluetooth

Ce projet s'appuie sur l'application de contrôle d'une maison connectée réalisée en TP. Il est demandé de créer une application qui s'installera sur deux smartphones Android. Comme illustré par la Fig 1, un des smartphones fera office de télécommande utilisateur. Le deuxième fera office de passerelle entre le Bluetooth et l'API REST (ou bien d'autres technologies utilisées en domotique comme le ZigBee, le 443MHz, le BLE, ...).

Une première activité permet de sélectionner le rôle désiré (client ou serveur) comme illustré par la Fig.2. Une fois un bouton pressé, la connexion Bluetooth est mise en place.

Lorsque la connexion est établie, le client affiche une activité qui liste l'ensemble des appareils connectés au smartphone du serveur avec l'interface présentée dans le sujet de TP. Quand le client demande d'allumer ou d'éteindre un appareil, une requête est envoyée via Bluetooth au serveur.

Lorsque le serveur est connecté au client, il affiche aussi l'activité de monitoring vue en TP (avec des boutons non-pressables) qu'il rafraîchit périodiquement. De plus, le serveur envoie via Bluetooth périodiquement les données nécessaires au client pour créer son affichage. Si le client demande d'allumer ou d'éteindre un appareil, le serveur transfère l'action via l'API REST. Une fois l'action effectuée, il envoie au client les informations mises à jour.

2 Bluetooth sur Android

Pour interagir avec la maison connectée depuis le client, il faut échanger des informations entre les téléphones. Cet échange se fait par l'ouverture d'un socket Bluetooth. Un des smartphones prend le rôle d'un serveur Bluetooth ; l'autre, celui d'un client. L'ouverture du socket de communication passe par une étape de connexion. Cette étape suppose que les deux téléphones sont déjà appairés physiquement entre eux via Bluetooth. On ne vous demande donc pas de programmer cet appairage, vous pouvez le réaliser "à la main" depuis le menu Bluetooth des téléphones.

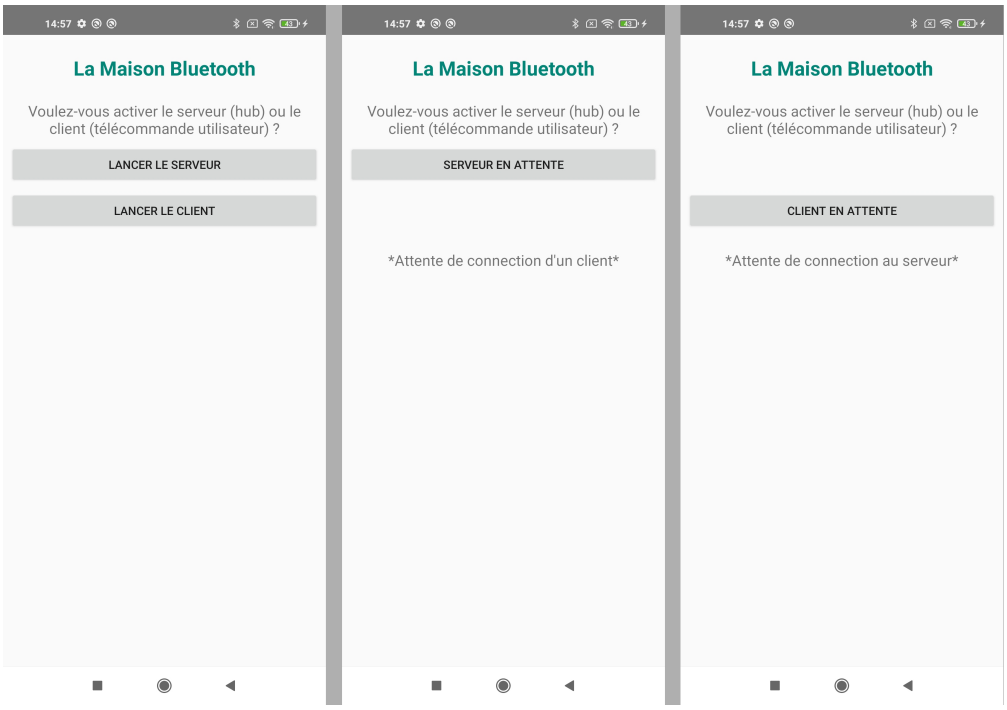

Figure 2: Écran d'invite de connexion (gauche), en cours de connexion si serveur (centre) et en cours de connexion si client (droite)

La suite du sujet décrit brièvement les principales étapes à réaliser pour implémenter la communication Bluetooth avec l'API standard d'Android. Pour plus de détail, référez-vous à la documentation officielle : `https://developer.android.com/guide/topics/connectivity/bluetooth`

2.1 Connexion

La connexion doit s'effectuer dans un ordre précis. Le serveur doit commencer à écouter pour une connexion entrante puis seulement après le début de l'écoute, le client peut commencer sa procédure de connexion.

Pour obtenir le socket de communication, le serveur utilise un objet `BluetoothServerSocket` qui attend la connexion d'un client pour retourner le socket de communication Bluetooth. C'est la méthode `listenUsingRfcommWithServiceRecord("MonServeur", mon_UUID)`; qui retourne l'objet `BluetoothServerSocket`. Ce `BluetoothServerSocket` est ensuite mis en écoute avec la méthode `accept()`. Quand une requête de connexion arrive d'un client, ce `BluetoothServerSocket` retourne le socket de communication de type `BluetoothSocket`. Ce socket est prêt à être manipulé pour échanger des messages avec le client.

Le `UUID` est un identifiant unique connu par le serveur et le client d'une même application. Cet `UUID` doit être généré en ligne et définit comme une constante dans vos classes.

Le client recherche avec le `BluetoothAdapter` l'ensemble des téléphones appairés disponibles avec `getBondedDevices()`. Après avoir sélectionné le `Device` du serveur, il demande l'accès au socket de ce `Device` avec `createRfcommSocketToServiceRecord(mon_UUID)` qui retourne un objet `BluetoothSocket`.

Pour établir la connexion avec ce `BluetoothSocket`, il faut utiliser la méthode `connect()`.

Remarque : Ce `BluetoothSocket` sera manipulé par la suite via un thread pour réaliser l'envoi et la réception des messages dans le reste de l'application. Pour que les données reçues dans le thread soient exploitables par une activité dans le thread graphique principal, il faudra lui associer un `Handler`.

Attention : Il est conseillé d'arrêter le *BluetoothServerSocket* une fois la connexion établie. Ceci évitera d'accepter d'autres connexions et de créer des sockets supplémentaires.

2.2 Communication

Une fois la connexion initialisée et le socket obtenu, on peut définir une nouvelle classe qui hérite de Thread pour lire un `InputStream` avec la méthode `read(byte[] buffer)` et écrire dans un `OutputStream` avec la méthode `write(byte[] buffer)` dans le socket. Pour lire des données issues du socket dans le thread, on le fera dans la méthode run(), et pour écrire des données, on créera une méthode write(byte[] buffer). Ce thread, ainsi créé, aura une portée globale (static) mais sera privé.

Si plusieurs activités ont besoin de ce thread, on peut utiliser le patron de conception du singleton en Java pour qu'elles y aient accès.

Pour résumer, à ce stade vous avez :

- Une activité qui gère la connexion Bluetooth. Cette activité utilise deux threads pour connecter le client et le serveur BT. Une fois que le client et le serveur sont connectés, ces deux threads s'arrêtent.
- Un troisième thread static est alors instancié (et non lancé) par l'activité de connexion Bluetooth pour permettre les échanges de messages via le socket Bluetooth.

Le socket Bluetooth ainsi créé sera utilisé (via le thread static) pour envoyer des informations entre le client et le serveur.

3 Y'a plus qu'à

Fusionner l'application du TP avec votre socket Bluetooth ...