CS378 - Mobile Computing

Connecting Devices
Connecting Devices

• How to pass data between devices?
  – Chat
  – Games
  – Driving

• Options:
  – Use the cloud and a service such as Parse
  – your own server
Connecting Devices

• For devices that are close options include:
  – NFC, near field communication (very close)
  – Bluetooth

• Other options:
  – via a network both devices are on
  – peer to peer wireless connections
Network Service Discovery

• NSD allows users of app to identify other devices on a local network
• Apps that use NSD broadcast their name and connection information to the local network
• Scan from other applications doing the same thing
• [Register] -> Discover -> Connect
NSD Basics

• Step 1 - Register Your Service on the local Network
• Create a NsdServiceInfo object
• Provides information to other devices on network
  – other devices use it to decide if they want to connect to your service or not
Create a NsdServiceInfo Object

```java
public void registerService(int port) {
    // Create the NsdServiceInfo object, and populate it.
    NsdServiceInfo serviceInfo = new NsdServiceInfo();

    // The name is subject to change based on conflicts
    // with other services advertised on the same network.
    serviceInfo.setServiceName("NsdChat");
    serviceInfo.setServiceType("_http._tcp.");
    serviceInfo.setPort(port);
    ....
}
```
NsdServiceInfo Object

• Service name is NsdChat
• name visible to any other devices on network using NSD to look for local services
• Must be unique or Android steps in and resolves conflict
• NsdChat, NsdChat(1), NsdChat(2), ...
Service Type

• Service type: "_http._tcp." _<protocol>_ _<transport_layer>

• List of service types used by discovery service protocols (NSD - Android, Bonour - iO)  

• International Assigned Numbers Authority

• [http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml](http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml)
Port

• int port sent to method that builds NsdServiceInfo object
• avoid hard coding port number
• Use next available prot
• System service exists to get the next available port number
NSD Basics

• Step 2 - Implement the RegistrationListener

• so Android OS can inform our app of success or failure of service registration and unregistration
Implement the RegistrationListener

```java
public void initializeRegistrationListener() {
    mRegistrationListener = new NsdManager.RegistrationListener() {

        @Override
        public void onServiceRegistered(NsdServiceInfo NsdServiceInfo) {
            // Save the service name. Android may have changed it in order to
            // resolve a conflict, so update the name you initially requested
            // with the name Android actually used.
            mServiceName = NsdServiceInfo.getServiceName();
        }

        @Override
        public void onRegistrationFailed(NsdServiceInfo serviceInfo, int errorCode) {
            // Registration failed! Put debugging code here to determine why.
        }
    }
}
```
NSD Basics

• Step 3 - Register the service with NSD

```java
public void registerService(int port) {
    NsdServiceInfo serviceInfo = new NsdServiceInfo();
    serviceInfo.setServiceName("NsdChat");
    serviceInfo.setServiceType("_http._tcp.");
    serviceInfo.setPort(port);

    mNsdManager = Context.getSystemService(Context.NSD_SERVICE);

    mNsdManager.registerService(
        serviceInfo, NsdManager.PROTOCOL_DNS_SD, mRegistrationListener)
```
NSD Basics

• Step 4 - Discover Services on Network
• Application listens to service broadcasts sent via the local network
• Determine what services available
  – filter out things app cannot work with
• Possible you only want to use service from your own app running on another device
NSD Service Discovery

• Set up a discovery listener
• call discoverServices() method
NSD Service Discovery

- Set up a discovery listener

```java
public void initializeDiscoveryListener() {

    // Instantiate a new DiscoveryListener
    mDiscoveryListener = new NsdManager.DiscoveryListener() {
        // Called as soon as service discovery begins.
        @Override
        public void onDiscoveryStarted(String regType) {
            Log.d(TAG, "Service discovery started");
        }
    };
}
```
NSD Service Discovery

```java
@override
public void onServiceFound(NsdServiceInfo service) {
    // A service was found! Do something with it.
    Log.d(TAG, "Service discovery success" + service);
    if (!service.getServiceType().equals(SERVICE_TYPE)) {
        // Service type is the string containing the protocol and
        // transport layer for this service.
        Log.d(TAG, "Unknown Service Type: " + service.getServiceType());
    } else if (service.getServiceName().equals(mServiceName)) {
        // The name of the service tells the user what they'd be
        // connecting to. It could be "Bob's Chat App".
        Log.d(TAG, "Same machine: " + mServiceName);
    } else if (service.getServiceName().contains("NsdChat")){
        mNsdManager.resolveService(service, mResolveListener);
    }
}
```

More methods for Service Lost, Discovery Stopped, and failures of discovery
NSD Service Discovery

• After creating the listener call discoverServices

```java
mNsdManager.discoverServices(
    SERVICE_TYPE, NsdManager.PROTOCOL_DNS_SD, mDiscoveryListener)
```
NDS Basics

• Step 5, After Discovery -> Connect
• Determine Connection information with call to resolveService() method

• Another Listener
  – NsdManager.ResolveListener
  – NsdManager will call back with an NsdServiceInfo object with connection information
  – recall, app that registers services created NsdServiceInfo object
NSD Connect to Discovered Service

NsdManager.ResolveListener to connect

```java
public void initializeResolveListener() {
    mResolveListener = new NsdManager.ResolveListener() {

        @Override
        public void onResolveFailed(NsdServiceInfo serviceInfo, int errorCode) {
            // Called when the resolve fails. Use the error code to debug.
            Log.e(TAG, "Resolve failed" + errorCode);
        }
    }
}
```
NSD Connect to Discovered Service

NsdManager.ResolveListener to connect

```java
@Override
public void onServiceResolved(NsdServiceInfo serviceInfo) {
    Log.e(TAG, "Resolve Succeeded. " + serviceInfo);

    if (serviceInfo.getServiceName().equals(mServiceName)) {
        Log.d(TAG, "Same IP.");
        return;
    }
    mService = serviceInfo;
    int port = mService.getPort();
    InetAddress host = mService.getHost();
```
NDS Basics

• Step 6 - Unregister service when application closes

```java
protected void onPause() {
    if (mNsdHelper != null) {
        mNsdHelper.tearDown();
    }
    super.onPause();
}
```

// NsdHelper's tearDown method
```java
public void tearDown() {
    mNsdManager.unregisterService(mRegistrationListener);
    mNsdManager.stopServiceDiscovery(mDiscoveryLister);
}
```
Peer to Peer with Wi-Fi

• Alternative to Cloud and Network Service
  Discover

• Not required to connect to network or hotspot

• Interact with near by devices
  – range greater than Bluetooth
  – Code wise, as involved as NSD