NSY102
Découverte de services
le protocole mDNS-SD
L'API java JmDNS 3.4.1
Un usage du patron ServiceLocator

Cnam Paris
jean-michel Douin, douin au cnam point fr
18 Mai 2015

Notes de cours

Sommaire

Principale bibliographie utilisée

- A Classification of Service Discovery Protocols
  - http://doc.utwente.nl/54527/1/classification_of_service.pdf
- Survey of Service Discovery Protocols in Mobile Ad Hoc Networks
  - http://www.dis.uniroma1.it/~midlab/articoli/SSDP.pdf
- Apache and Zeroconf Networking
  - http://people.apache.org/~sctemme/ApconEU2005/FR08/FR08_Apache_and_Zeroconf.pdf
- ***JmDNS***
  - ***http://jmdns**.sourceforge.net*
  - ***http://jmdns**.sourceforge.net/apidocs/index.html*
  - *http://jmdns.sourceforge.net/xref-test/javax/jmdns/test/JmDNSTest.html*
  - **http://www.javased.com/ Java Examples : jmdns**
  - **http://grepcode.com/project/repo1.maven.org/maven2/javax.jmdns/jmdns/**
  - **http://sourceforge.net/projects/mdnstools/files/**
- **DNS SRV (RFC 2782) Service Types**
  - http://www.dns-sd.org/ServiceTypes.html

Le contexte

- **Découverte de services**
  - **Wikipedia**
    - Service discovery protocols (SDP) are network protocols which allow automatic detection of devices and services offered by these devices on a computer network.
    - Service discovery requires a common language to allow software agents to make use of one another's services without the need for continuous user intervention.

Préambule : Datagram UDP

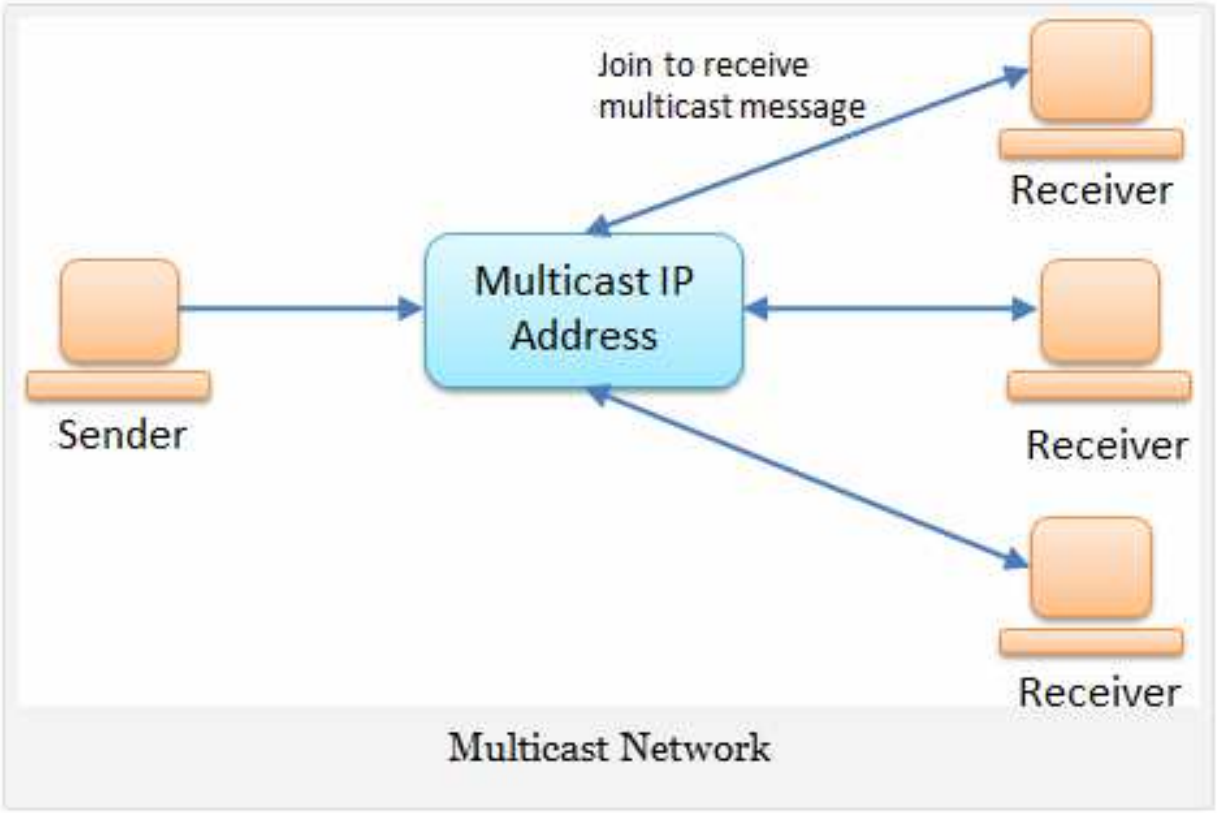
- **http://fr.wikipedia.org/wiki/User_Datagram_Protocol**

  - **Le *User Datagram Protocol***

  - **Le rôle de ce protocole est de permettre la transmission de données de manière très simple entre deux entités, chacune étant définie par une adresse IP et un numéro de port. Contrairement au protocole TCP, il fonctionne sans négociation : il n'existe pas de procédure de connexion préalable à l'envoi des données (le handshaking).**

  - **Donc UDP ne garantit pas la bonne livraison des datagrammes à destination, ni leur ordre d'arrivée. Il est également possible que des datagrammes soient reçus en plusieurs exemplaires.**

Préambule : Multicast IP



- **Src: http://lycog.com/programming/multicast-programming-java/**
  - **En java : quelques lignes, en préambule**

Préambule : en java

- **Deux classes prédéfinies**
  - **java.net.MulticastSocket, java.net.DatagramPacket**

- **Adresse multicast 224.0.0.251** *(224.0.0.1 à 224.0.0.255)*
- **Port 5253** *(au hasard…)*

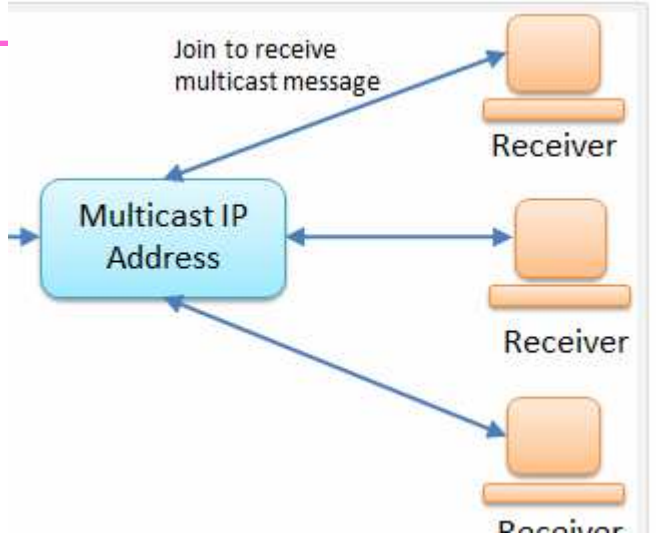
- **Deux classesJava**
  - **La classe MulticastSender**
    - **Envoi de hello à tous …**
  - **La classe MulticastReceiver**
    - **Réception et affichage**

- **Démonstration**
  - **Un émetteur, un récepteur**
  - **Un émetteur plusieurs récepteurs**
  - **Plusieurs émetteurs plusieurs récepteurs**

Le « MulticastSender » 224.0.0.251:5253

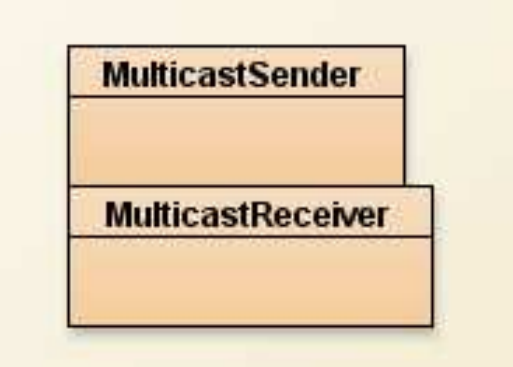

```
import java.net.*;
import java.io.*;

public class MulticastSender{

  public static void main(String[] args) throws IOException{
   int port=5253;
   MulticastSocket socket=new MulticastSocket(port);
   InetAddress group=InetAddress.getByName("224.0.0.251");

   String msg="Hello";
   socket.joinGroup(group);

   DatagramPacket packet;
   packet = new DatagramPacket(msg.getBytes(), msg.length(),
                               group, port);
   socket.send(packet);
   socket.close();
  }
}
```

Le « MulticastReceiver » 224.0.0.251:5253




```
import java.io.*;
import java.net.*;

public class MulticastReceiver {


  public static void main(String[] args) throws
   byte[] inBuf = new byte[256];

   MulticastSocket socket = new MulticastSocket(5253);
   InetAddress address = InetAddress.getByName("224.0.0.251");
   socket.joinGroup(address);
   while (true) {
    DatagramPacket inPacket = new DatagramPacket(inBuf,
                                                 inBuf.length);
      socket.receive(inPacket);
      String msg = new String(inBuf, 0, inPacket.getLength());
      System.out.println("  Reçu depuis " + inPacket.getAddress()
                                          + " msg : " + msg);
    }
  }}
```

Hello , une démo ?



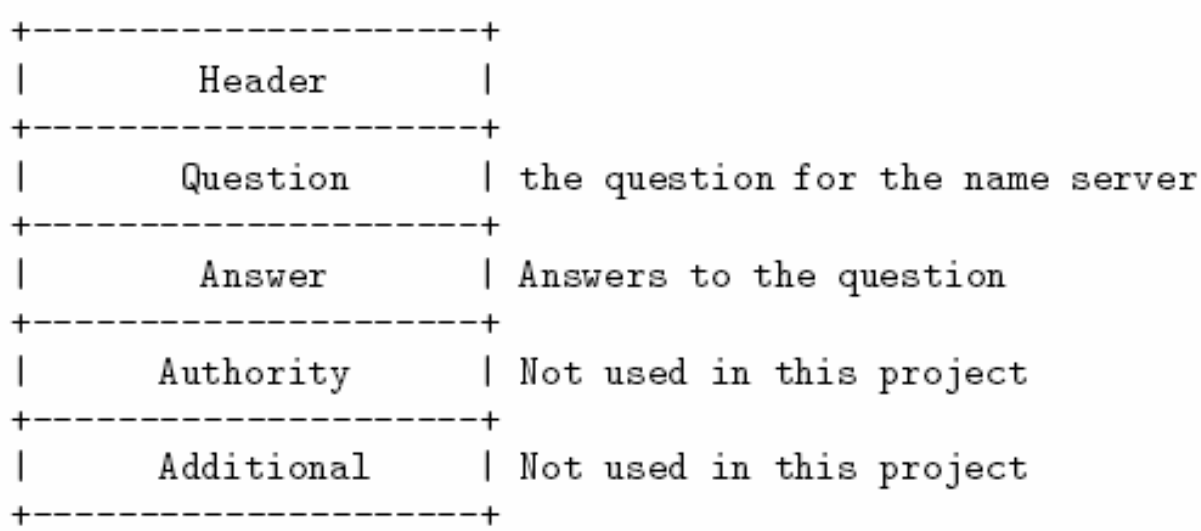

```
D:\FREECOM\NSY102\JmDNS\JmDNS_bluej>java -cp . MulticastReceiver
Depuis /192.168.56.1 msg : Hello
```

- **Hello**
  - **Rappel ? un protocole XXX,**
    - **ce sont des paquets avec un format imposé et (souvent) un port dédié**
    - **Par exemple 5353 pour le protocole mDNS**
    - **Voir en http://jfod.cnam.fr/nsy102/jmdns/jmdns_examples.jar**

mDNS ou DNS structure des paquets

1 DNS Packet Structure

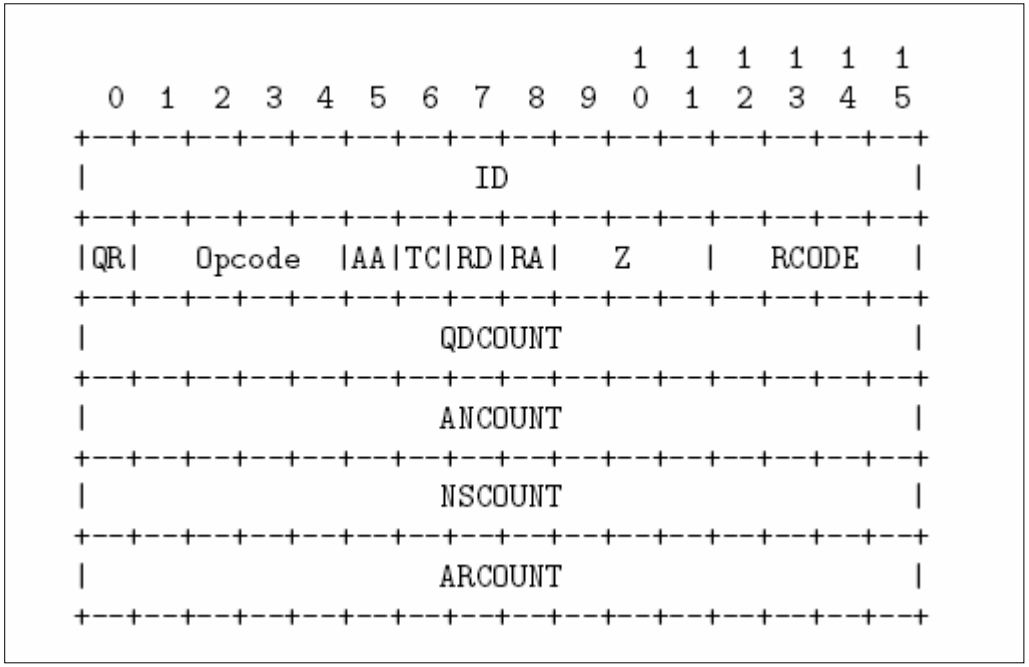
All DNS packets have a structure that is



- **Src:** http://www.ccs.neu.edu/home/amislove/teaching/cs4700/fall09/handouts/project1-primer.pdf

Paquet DNS

2 DNS Headers

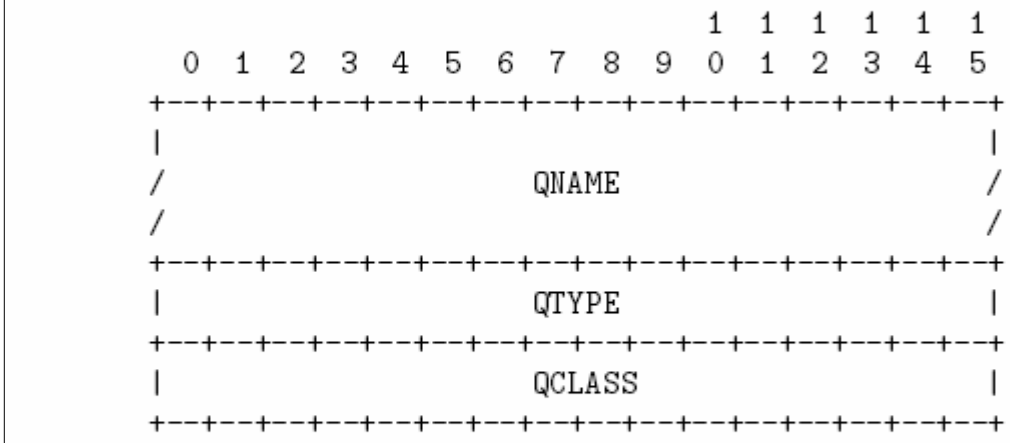


- **Description de chaque champ :**
- **http://www.ccs.neu.edu/home/amislove/teaching/cs4700/fall09/handouts/project1-primer.pdf**

Paquet DNS, champ *question*

3 DNS Questions

A DNS question has the format



Where each of these fields is as described below:

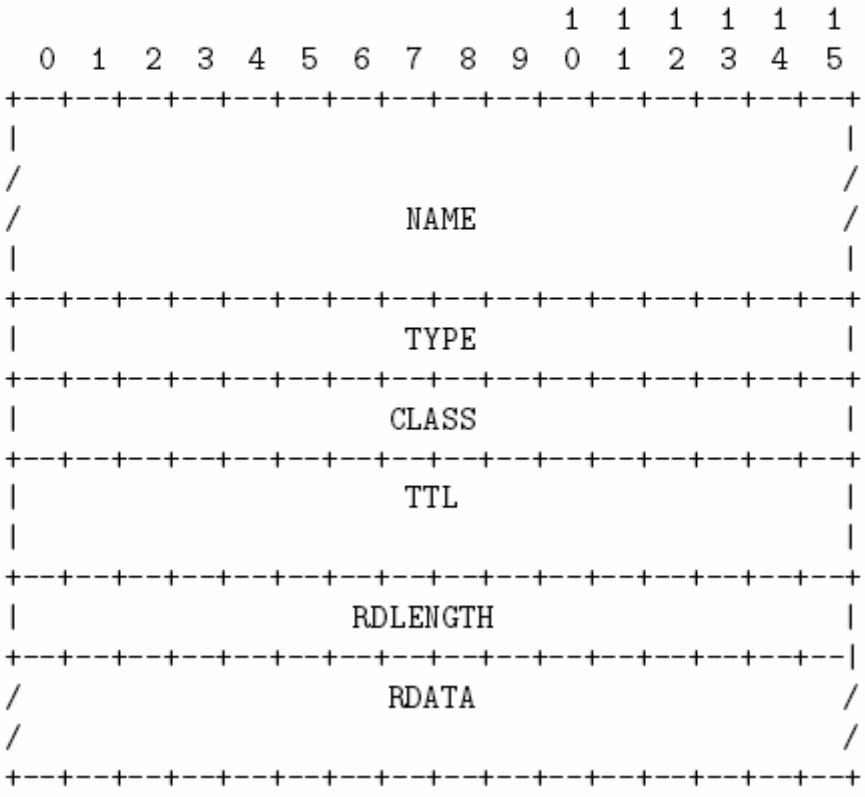
QNAME A domain name represented as a sequence of labels, where each label consists of a length octet followed by that number of octets. The domain name terminates with the zero length octet for the null label of the root. See the DNS Example query below.

- http://www.ccs.neu.edu/home/amislove/teaching/cs4700/fall09/handouts/project1-primer.pdf

Paquet DNS, champ *answer*

4 DNS Answers

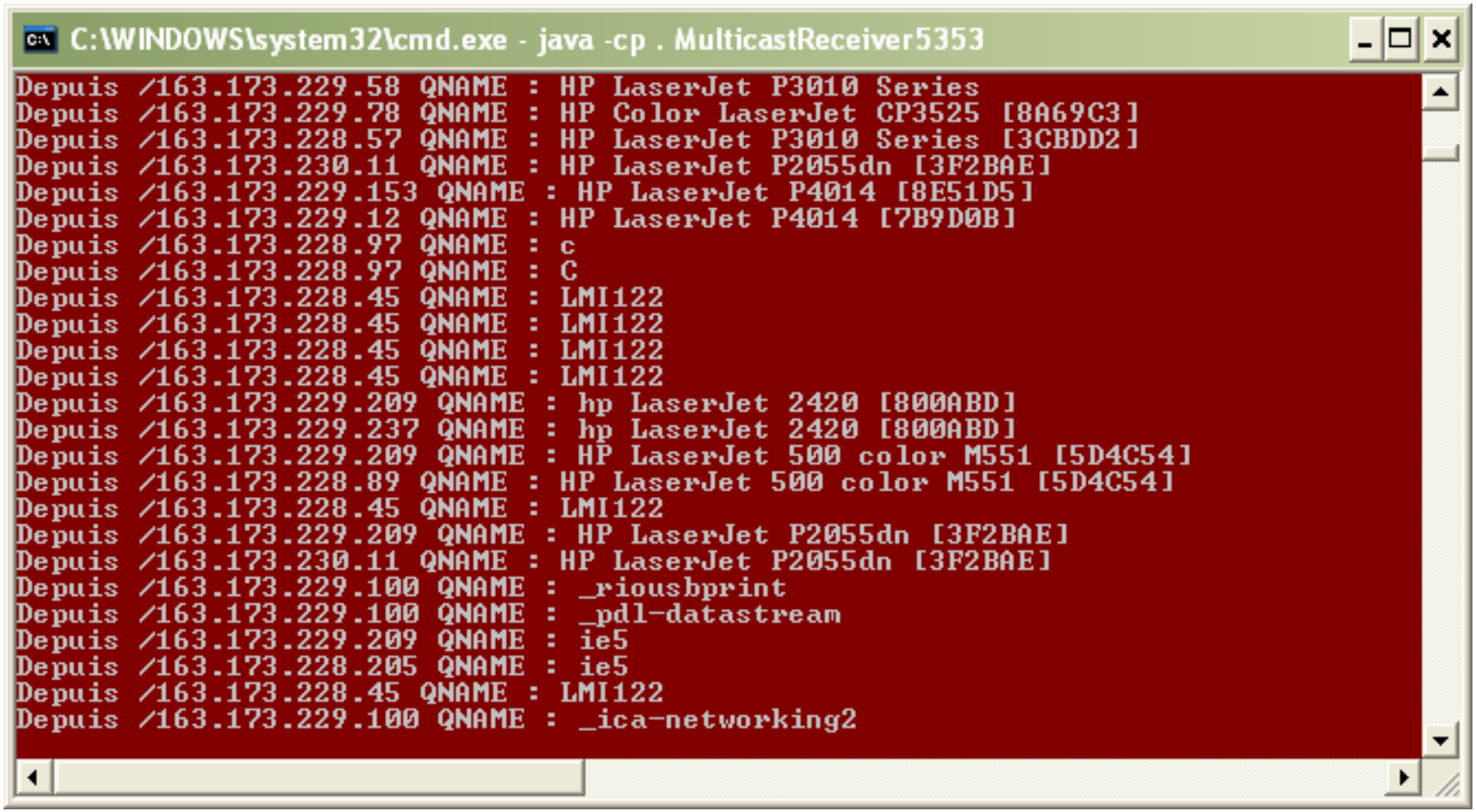
A DNS answer has the format



- http://www.ccs.neu.edu/home/amislove/teaching/cs4700/fall09/handouts/project1-primer.pdf

mDNS 224.0.0.251:5353

- **Démonstration**
  - **Afficher le champ QNAME du champ question des paquets DNS à cette adresse sur son poste de travail**
  - **Modification de la classe MulticastReceiver …**



JmDNS ?

- **Une API Java toute prête**
  - **ServiceInfo : une classe java associée au paquet DNS**
  - **Découverte de services**
    - **Un thread d'écoute des trames au bon protocole**
  - **Enregistrement de services**
    - **Diffusion du service**

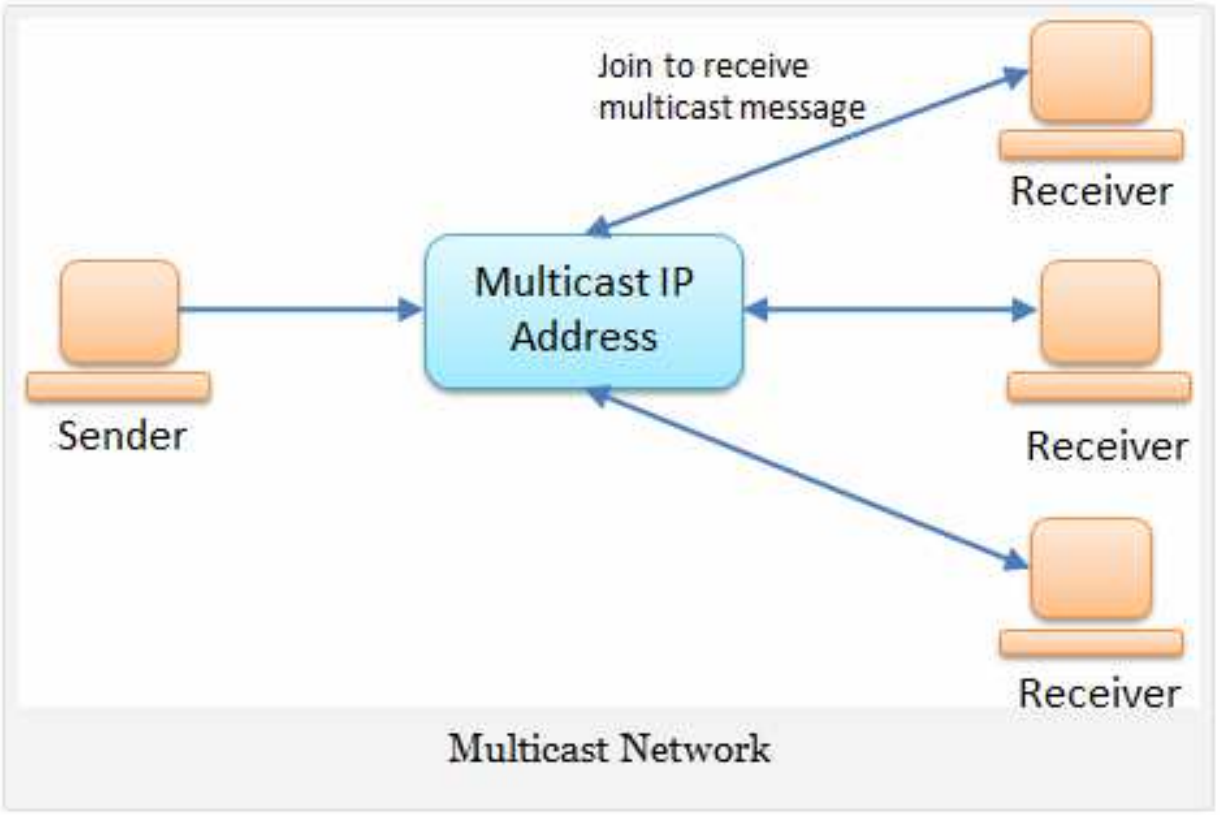
- **Vérifions que JmDNS utilise bien ce port et cette adresse …**
  - **Port : `5353`**
  - **Adresse : `224.0.0.251`**
  - http://jmdns.sourceforge.net/clover/javax/jmdns/impl/constants/DNSConstants.html

API jmdns, DNSConstants

```java
/**
 * DNS constants.
 *
 * @author Arthur van Hoff, Jeff Sonstein, Werner Randelshofer, Pierre Frisch, Rick Blair
 */
public final class DNSConstants {
    // http://www.iana.org/assignments/dns-parameters

    // changed to final class - jeffs
    public static final String MDNS_GROUP                   = "224.0.0.251";
    public static final String MDNS_GROUP_IPV6              = "FF02::FB";
    public static final int    MDNS_PORT                    = Integer.parseInt(System.getProperty("net.mdns.port", "5353"));
    public static final int    DNS_PORT                     = 53;
    public static final int    DNS_TTL                      = 60 * 60;                                                    //
    // public static final int DNS_TTL = 120 * 60; // two hour TTL (draft-cheshire-dnsext-multicastdns.txt ch 13)
```

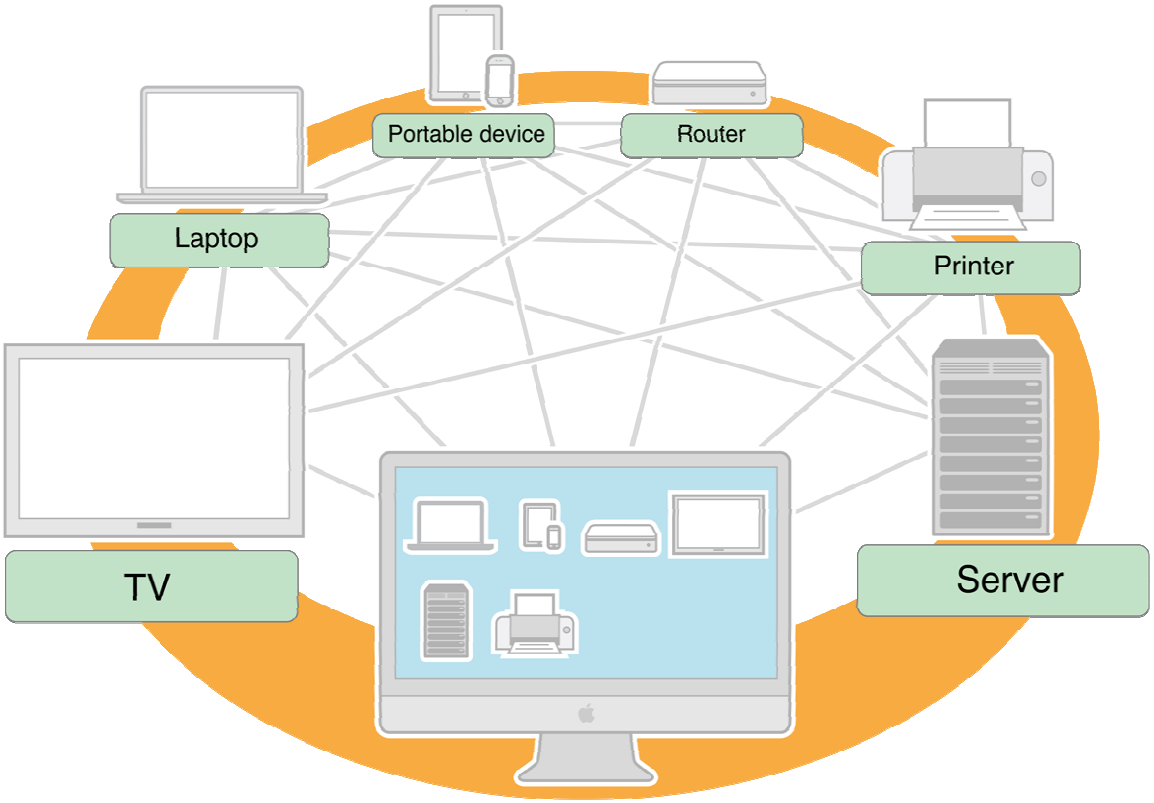
- http://jmdns.sourceforge.net/clover/javax/jmdns/impl/constants/DNSConstants.html
- **Fin du préambule…**

mDNS-SD / JmDNS



- **mDNS-SD : Un protocole, des données échangées**
- **jmDNS : une implémentation en java**

Bonjour Apple



- https://developer.apple.com/library/prerelease/ios/documentation/Cocoa/Conceptual/NetServices/Introduction.html
- https://developer.apple.com/library/ios/documentation/Networking/Conceptual/dns_discovery_api/Introduction.html

Avertissement, sur ces notes de cours

- **Approche de mDNS-SD par l'API Java**
  - **Pas de présentation du protocole …**
  - **Ni des couches ISO …**
  - **Ni du 802.11ac …**

- **Au sommaire**
  - **Le contexte par un exemple**
  - **Découverte des services**
  - **Installation d'un service**
  - **Découverte des types**
  - **Un exemple de capteurs**
  - **Le patron service locator**
  - **Un TP induit**

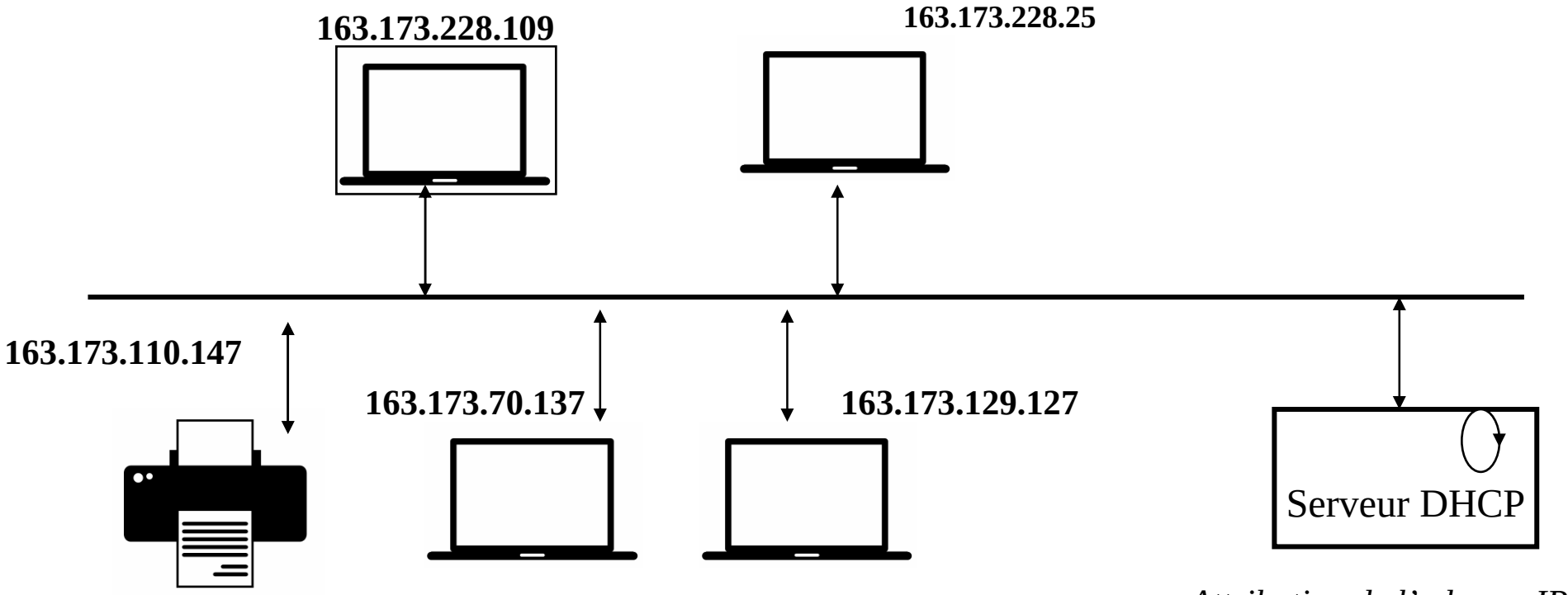
Un exemple …

- **Chaque imprimante connectée possède un service web**
  - **Un service de commande de « toner » par exemple**
    - **http://163.173.228.120:80**

  - **Cf. les types :** http://www.dns-sd.org/ServiceTypes.html

- **Un certain type de périphérique délivre de la vidéo**

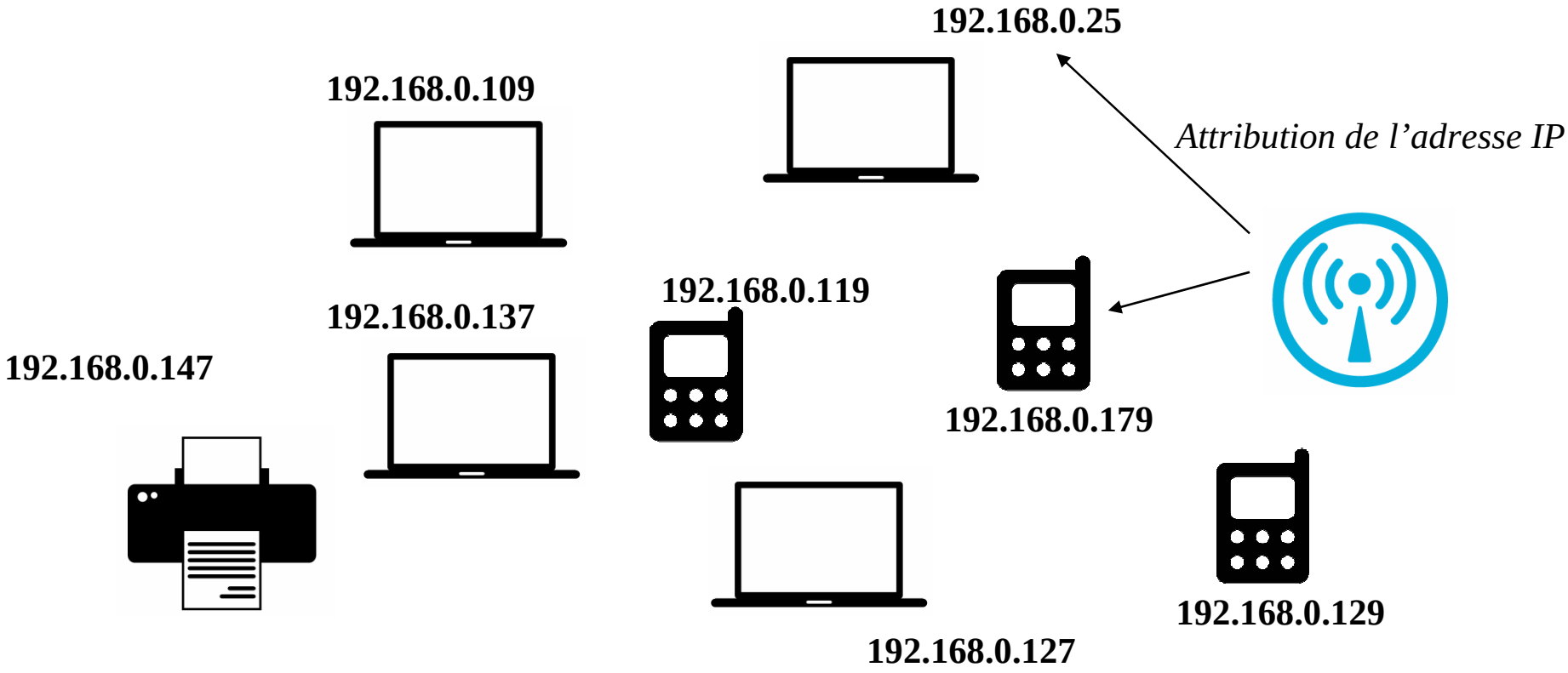
Contexte … bien connu



Attribution de l'adresse IP

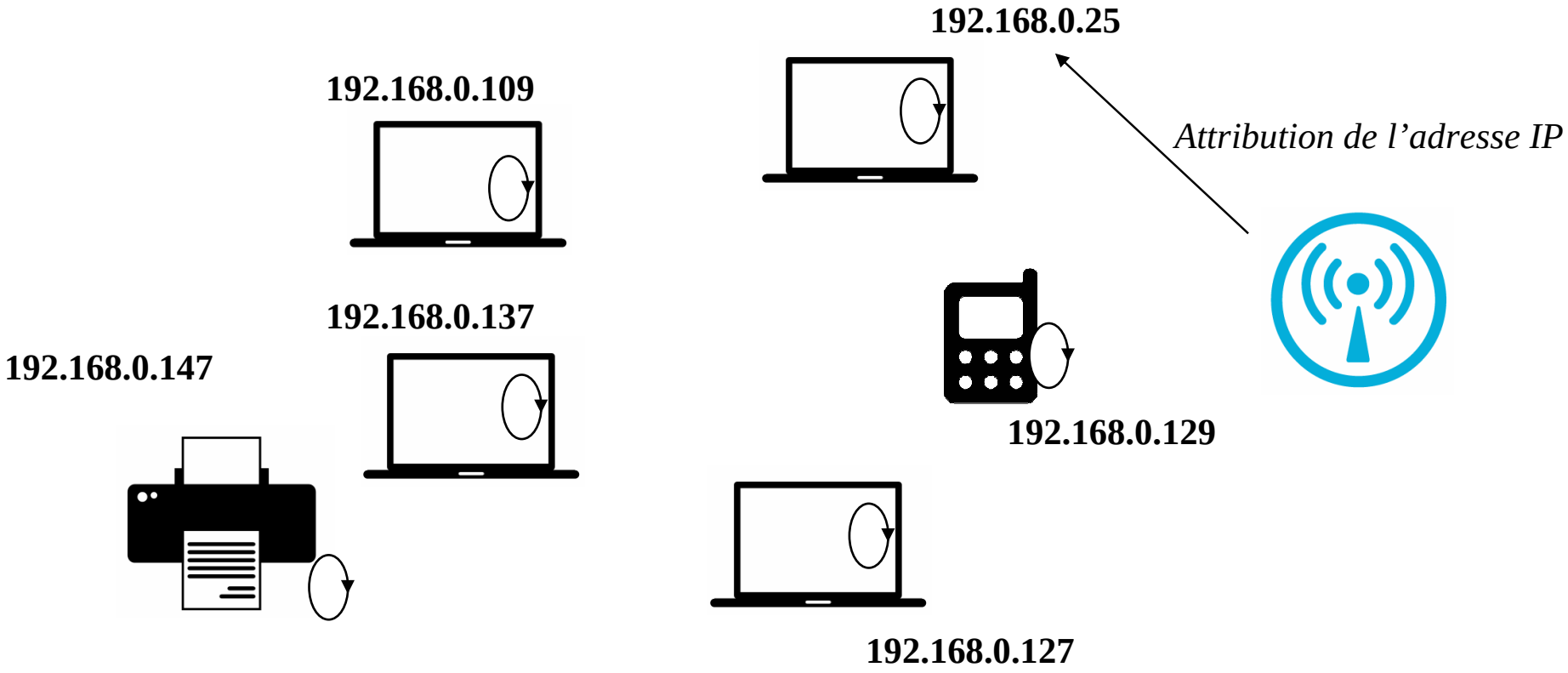
- **Contexte :**
  - **un réseau 163.173.xxx.xxx**
  - **Chaque machine reliée possède une adresse IP,**
    - **adresse attribuée par le serveur DHCP**

Contexte : des machines reliées entre elles



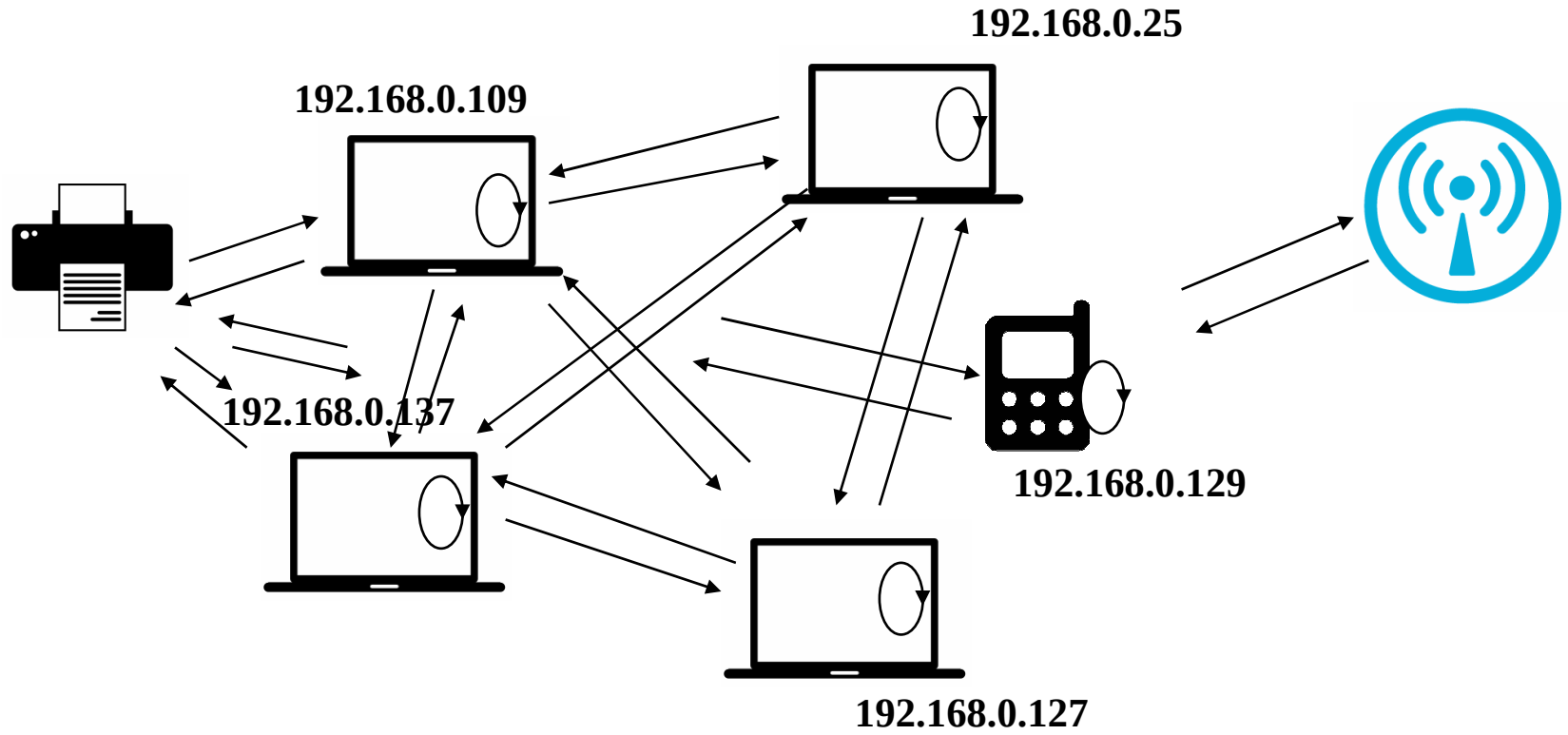
- **Contexte en WiFi, réseau local : idem au réseau filaire**
  - **un réseau 192.168.0.xxx**
  - **Chaque pc ou mobile possède une adresse IP attribuée par l'antenne**

Question : chaque machine possède un service web 1/2



- **Découverte du service de chacun, mais comment ?**
  - **Quelle url ?, quel port ?**

Réponse : découverte 2/2



- **Une solution : diffusion multicast de l'adresse IP, du port …**
  - **La découverte des services de chacun,**
    - **Diffusion Multicast de son URL adresse IP + port et du texte**
    - **mDNS**

Zeroconf

- **Le bien nommé**
  - **Wikipedia :**
    - **Zero Configuration Networking (Zeroconf) est l'appellation générique d'un ensemble de protocoles permettant de créer automatiquement un réseau IP utilisable sans configuration particulière ou serveurs dédiés.**
  - **Résolution de noms**
    - **Multicast DNS (mDNS) est un protocole utilisant des datagrammes**
    - **Un client mDNS**
      - **Obtention de l'adresse IP d'un périphérique à partir de son nom**
      - **Nom unique (numéroté si homonyme)**
  - **Recherche de services**
    - **DNS-SD (Service Discovery) utilise mDNS**
    - ***UPnP (SSDP), SLP …***

Enregistrement de type SRV (RFC 2782)

- **http://fr.wikipedia.org/wiki/Enregistrement_de_service**
  - http://www.dns-sd.org/ServiceTypes.html

- **Un enregistrement de service avec les champs suivants**
  - **Service :** *_http*
  - **Protocole :** *_tcp* **ou** *_udp*
  - **Nom.de.domaine**
  - **TTL :** *durée de validité Time-To-Live de la réponse en secondes*
  - **Classe :** *IN pour internet*
  - **Priorité :** *priorité du serveur cible, valeur faible priorité élevée*
  - **Poids :** *poids relatif donné pour les services identiques, (permet un choix aléatoire)*
  - **Port :** *numéro de port*
  - **Cible :** *nom du serveur qui fournit le service*

  - *Note :*
    - *une corrélation sera faite avec l'API JmDNS,*
      - *notamment à la création du service et la classe ServiceInfo*

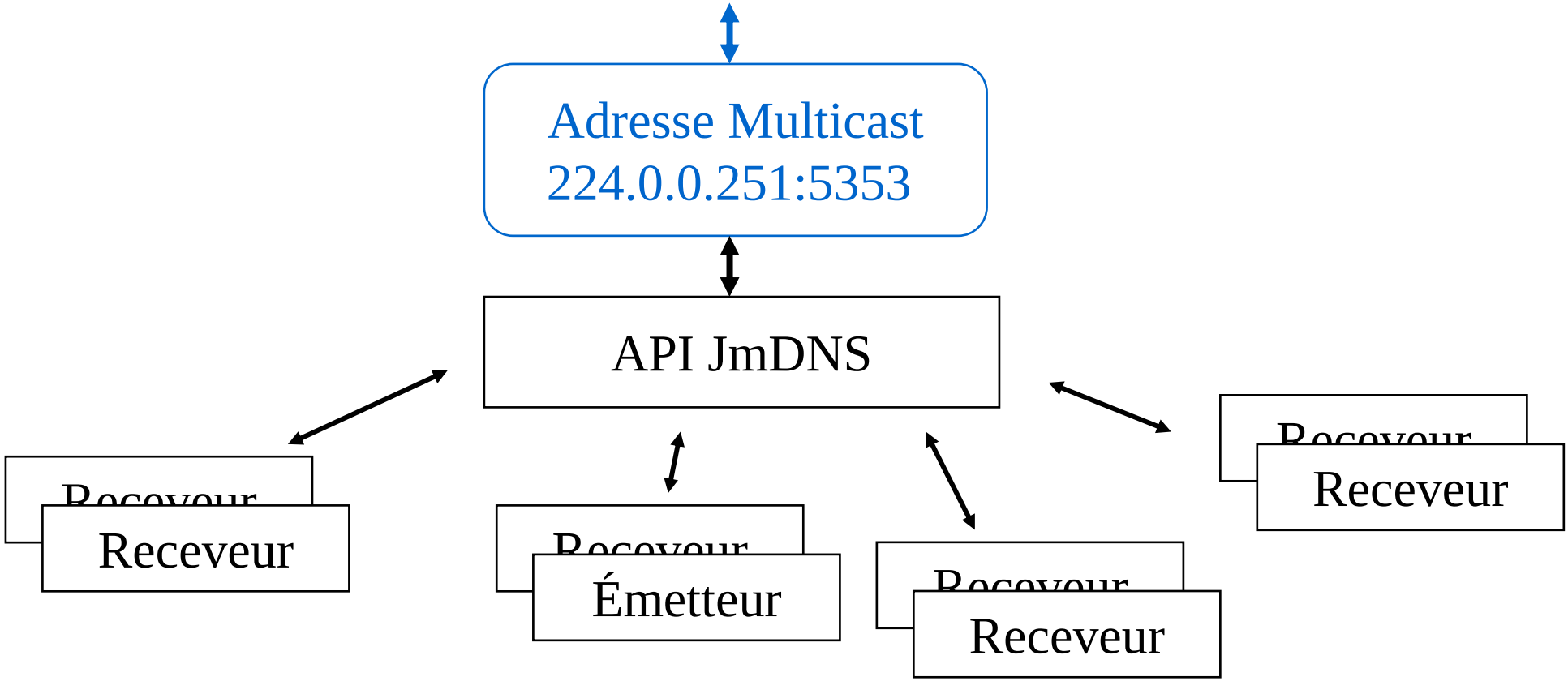
Quelques règles :

- **Nom de service: nom unique priorité 0 poids 0**
- **Par convention numérotation des services aux mêmes caractéristiques**
  - **Attention aux caractères tels que le '.' Pour les noms des services …**

API Java JmDNS

- **http://jmdns.sourceforge.net/**
- **API javadoc**
  - **http://jmdns.sourceforge.net/apidocs/index.html**
- **Sommaire**
  - **Contexte**
  - **Découverte des services**
    - **Selon une convention de noms de types**
      - **Exemple, services web : "_http._tcp.local."**
    - http://www.dns-sd.org/ServiceTypes.html
  - **Création et enregistrement des services**

API jmdns



- **Un singleton**
  - **Une API écrite en java**
  - **Adéquation Datagramme ←→ POO**

Architecture : Paquetages JmDNS 3.4.1

JmDNS 3.4.1

All Classes

Packages
javax.jmdns
javax.jmdns.impl
javax.jmdns.impl.constants
javax.jmdns.impl.tasks
javax.jmdns.impl.tasks.resolver
javax.jmdns.impl.tasks.state

javax.jmdns

Interfaces
JmDNS.Delegate
JmmDNS
JmmDNS.Factory.ClassDelegate
NetworkTopologyDiscovery
NetworkTopologyDiscovery.Factory.Clas
NetworkTopologyListener
ServiceListener
ServiceTypeListener

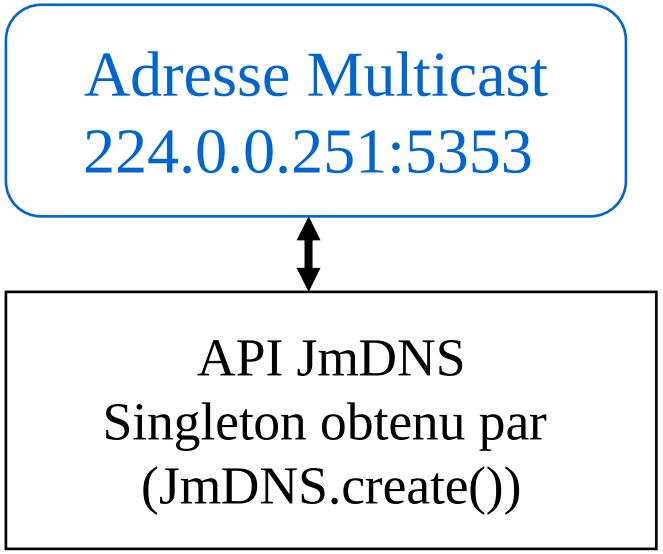
Classes
JmDNS
JmmDNS.Factory
NetworkTopologyDiscovery.Factory
NetworkTopologyEvent
ServiceEvent
ServiceInfo

Enums
ServiceInfo.Fields

- **Patron fabrique est utilisé**
  - **Covariance en sus**
- **JmDNS jmdns = JmDNS.create();**
  - **return new JmDNSImpl();**

Initialisation, un schéma

```
JmDNS jmdns = null;

try{
   jmdns = JmDNS.create();

   …

}catch(Exception e )
  …
}finally{
   jmdns.close();
}
```

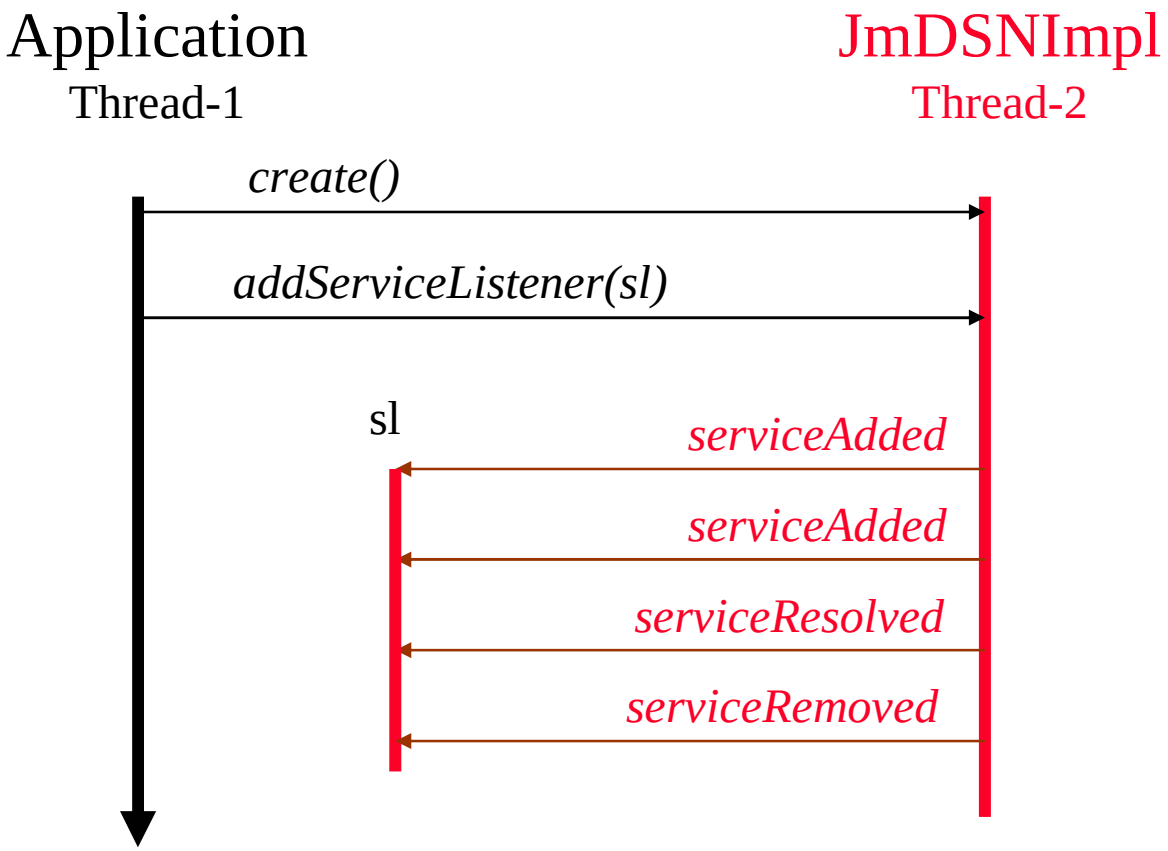


Découverte de service

- **Un écouteur/observateur auprès d'une instance JmDNS**
  - **ServiceListener**

```
public class …

    – JmDNS jmdns = JmDNS.create();
    – jmdns.addServiceListener("_http._tcp.local.", new LogListener());


   static class LogListener implements ServiceListener {
     @Override
     public void serviceAdded(ServiceEvent event) {

     }

     @Override
     public void serviceRemoved(ServiceEvent event) {

     }

     @Override
      public void serviceResolved(ServiceEvent event) {
   }
 }
```

« Architecture » en asynchrone



- **Application, découverte des services**
  - **Deux threads, mécanisme du « callback » : un classique**

ServiceListener un exemple

- **Un écouteur/observateur auprès d'une instance JmDNS**
  - **JmDNS jmdns = JmDNS.create();**
  - **jmdns.addServiceListener("_http._tcp.local.", new LogListener());**

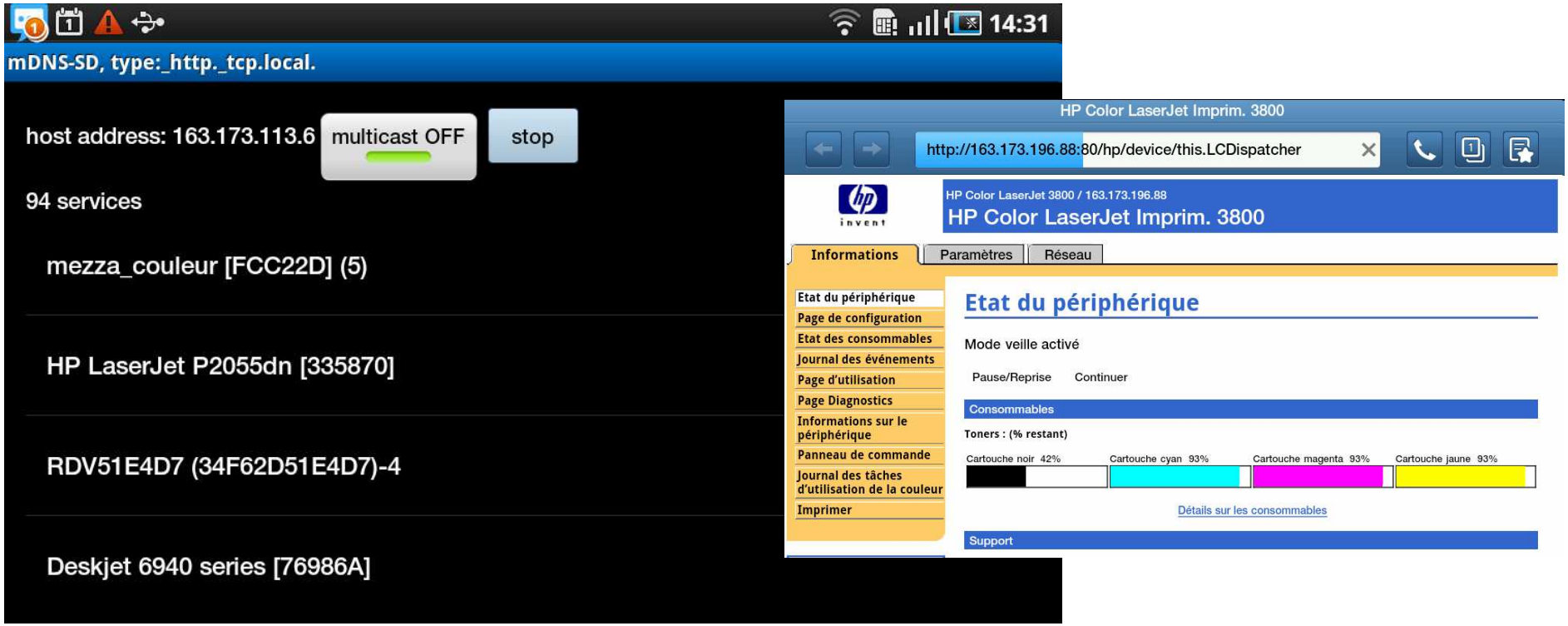
```
static class LogListener implements ServiceListener {
    @Override
    public void serviceAdded(ServiceEvent event) {
      System.out.println("Service added   : " + event.getName() + "." + event.getType());
    }

    @Override
    public void serviceRemoved(ServiceEvent event) {
      System.out.println("Service removed : " + event.getName() + "." + event.getType());
    }

    @Override
     public void serviceResolved(ServiceEvent event) {
      System.out.println("Service resolved: " + event.getInfo());
    }
}
```

Résultat au Cnam en mars 2015

```
Service added   : Brother HL-5250DN series [00807782748f]._http._tcp.local.
Service resolved: [ServiceInfoImpl@16366669 name: 'Brother HL-5250DN series [00807782748f]._http.
Service added   : HP LaserJet 500 color M551 [5D4C54]._http._tcp.local.
Service resolved: [ServiceInfoImpl@12989962 name: 'HP LaserJet 500 color M551 [5D4C54]._http._tcp
Service added   : HP Color LaserJet CP3525 [8AC9FF]._http._tcp.local.
Service resolved: [ServiceInfoImpl@29668982 name: 'HP Color LaserJet CP3525 [8AC9FF]._http._tcp.l
Service added   : NAS-AKOKA._http._tcp.local.
Service added   : HP Color LaserJet CP3525 [8BF199]._http._tcp.local.
Service resolved: [ServiceInfoImpl@18475065 name: 'HP Color LaserJet CP3525 [8BF199]._http._tcp.l
Service added   : HP Color LaserJet CP3525 [8A69C3]._http._tcp.local.
Service resolved: [ServiceInfoImpl@12263427 name: 'HP Color LaserJet CP3525 [8A69C3]._http._tcp.l
Service added   : lmi19._http._tcp.local.
Service resolved: [ServiceInfoImpl@25960696 name: 'lmi19._http._tcp.local.' address: '/163.173.22
Service added   : HP LaserJet P2055dn [3CCCA0]._http._tcp.local.
```

- **Tous les périphériques possédant un service http**
  - **Le premier : est une imprimante Brother HL-5250DN**
    - **http://163.173.229.204:80**
      - **Depuis un navigateur**

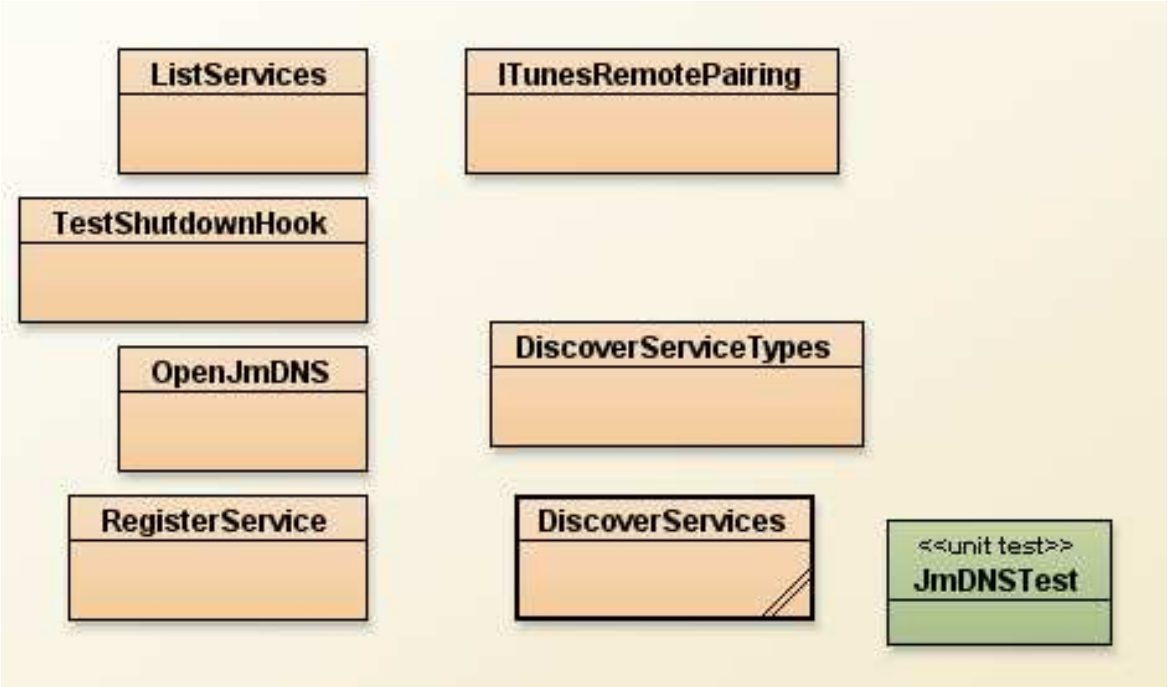
Sur votre mobile Android, les services http accessibles



- **Au click sur l'item HP Laser, la page est affichée**
  - **Ici un service web de configuration,**
    - **Autres services web : autres url**
- https://play.google.com/store/apps/details?id=com.grokkt.android.bonjour

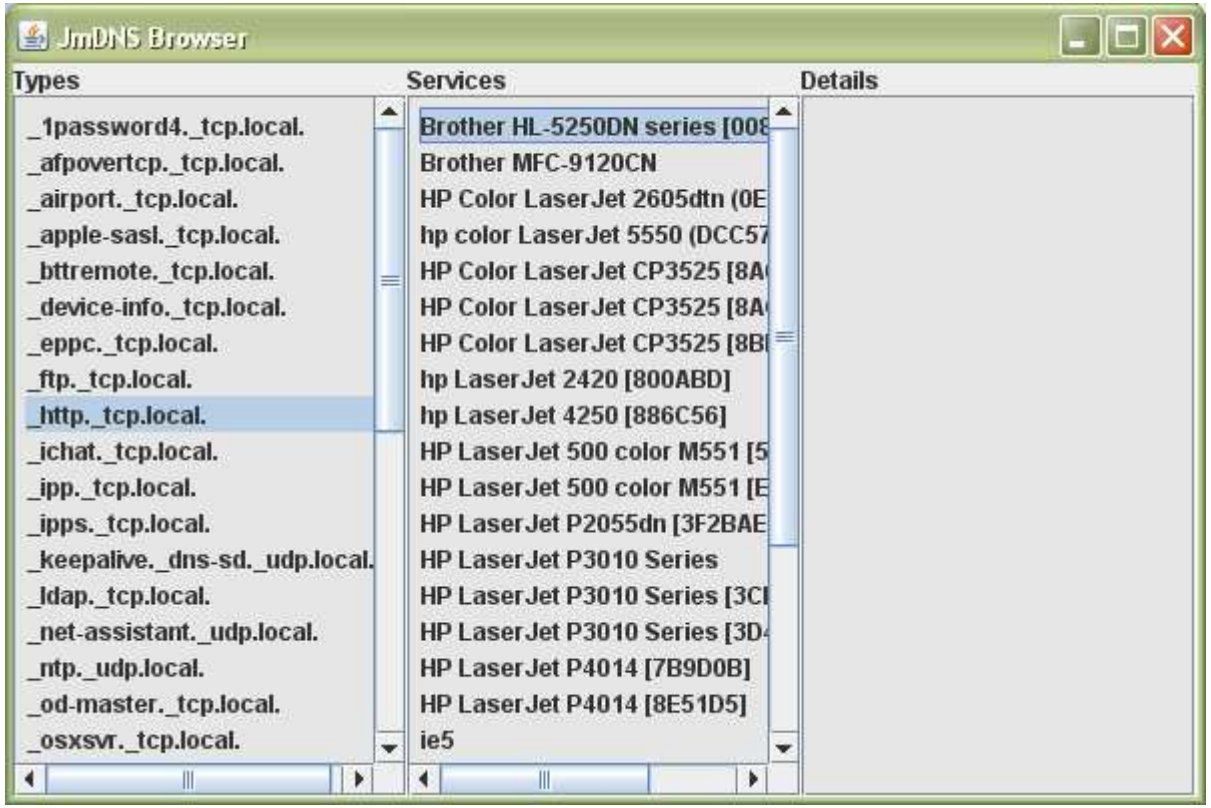
Démonstration

- **Bluej**
  - **Projet bluej ici : http://jfod.cnam.fr/nsy102/jmdns/jmdns_examples.jar**
    - **package examples**
      - **La librairie JmDNS est ici http://jfod.cnam.fr/nsy102/jmdns/jmdns.jar**



Démonstration : un browser j2se prêt à l'emploi

- **https://github.com/jadahl/jmdns/tree/master/src**
  - **Auteurs : Arthur van Hoff, Werner Randelshofer**
  - **Cf. en annexe**



Initialisation de JmDNS

```
JmDNS jmdns = null;
try{
  jmdns =  JmDNS.create();

  jmdns = JmDNS.create(InetAddress.getLocalHost());

  jmdns = JmDNS.create(InetAddress.getLocalHost(), "unNom");


…
…
…
…
}finally{
  jmdns.close();
}
```

Quels sont les services ? ServiceInfo

- `// en mode synchrone`
- `// ServiceInfo[] services = jmdns.list("_http._tcp.local.");`

```
try {
  jmdns = JmDNS.create();
  ServiceInfo[] services = jmdns.list("_http._tcp.local.");

  for(ServiceInfo si : services){
    System.out.println(si.getURLs()[0]);
  }

  } finally {
    if (jmdns != null) jmdns.close();
  }
```

http://jmdns.sourceforge.net/apidocs/index.html
http://jmdns.sourceforge.net/apidocs/javax/jmdns/ServiceInfo.html

Démonstration / exécution

```
http://163.173.228.30:80
http://163.173.230.243:8080/
http://163.173.229.214:80
http://163.173.230.8:80
http://163.173.230.99:80
http://163.173.229.3:80
http://163.173.230.11:80
http://163.173.229.204:80
http://163.173.228.57:80
http://163.173.229.241:80
http://163.173.228.19:80
http://163.173.229.58:80
http://163.173.228.205:80
http://163.173.228.119:80
http://163.173.229.2:80
http://163.173.229.102:80
http://163.173.229.153:80
```

Quels sont les types ? ServiceTypeListener

- **JmDNS jmdns = JmDNS.create();**

// en mode asynchrone

- **// jmdns.addServiceTypeListener(new TypeListener());**

```
static class TypeListener implements ServiceTypeListener {


  @Override
  public void serviceTypeAdded (ServiceEvent event) {
    System.out.println("Service type added: " + event.getType());
  }

  @Override
  public void subTypeForServiceTypeAdded (ServiceEvent event) {
    System.out.println("SubType for service type added: " +
                          event.getType());
  }

}
```

Démonstration / exécution

```
Service type added: _smb._tcp.local.
Service type added: _ssh._tcp.local.
Service type added: _sftp-ssh._tcp.local.
Service type added: _ldap._tcp.local.
Service type added: _xgrid._tcp.local.
Service type added: _rfb._tcp.local.
Service type added: _od-master._tcp.local.
Service type added: _apple-sasl._tcp.local.
Service type added: _servermgr._tcp.local.
Service type added: _net-assistant._udp.local.
Service type added: _workstation._tcp.local.
Service type added: _pcast._tcp.local.
Service type added: _osxsvr._tcp.local.
Service type added: _afpovertcp._tcp.local.
Service type added: _printer._tcp.local.
Service type added: _pdl-datastream._tcp.local.
Service type added: _http._tcp.local.
Service type added: _telnet._tcp.local.
Service type added: _ipp._tcp.local.
Service type added: _ftp._tcp.local.
Service type added: _nvstream._tcp.local.
```

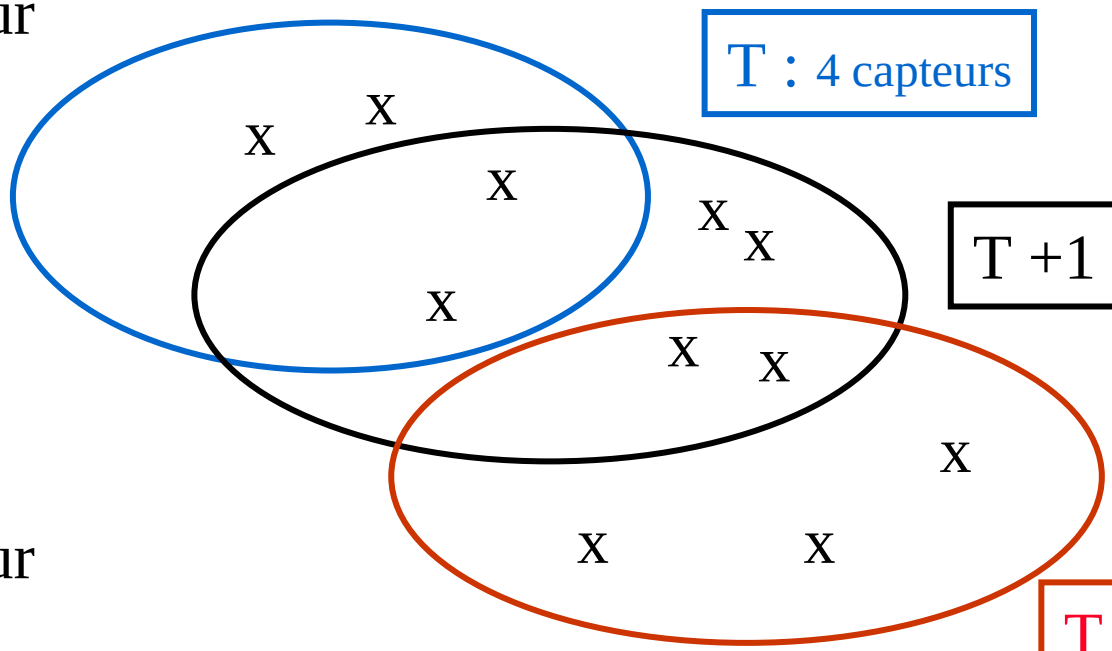
Un exemple…

- **Contexte:**
  - Une machine vient d'être installée sur le réseau,
  - Ce périphérique contient un serveur Web,
  - Ce serveur Web délivre la valeur d'un capteur de température …
    - Un exemple est ici
      - http://jfod.cnam.fr/ds2438/ds2438/
  - Ce capteur fait partie d'un réseau de capteurs de température…

- **Séquence :**
  - Découverte de l'URL dynamiquement
    - Le type est : _ds2438. _tcp.local.
  - Connexion avec ce serveur HTTP/TCP à cette URL
    - Obtention de la valeur
  - Application :
    - Calcul d'une nouvelle moyenne à chaque ajout ou retrait de périphérique

Schéma du « réseau »

x : un capteur
une URL



x : un capteur
une URL

T : Changement d'état estampillé

- **Un réseau qui évolue …**
  - **le nombre de capteurs à un instant T**

Obtention d'un service et connexion : schéma de programmes … un exemple

ExecutorService executor = Executors.newSingleThreadExecutor();

AtomicInteger number = new AtomicInteger(0);

// une classe interne …

```
static class DS2438ServiceListener implements ServiceListener {
    @Override
    public void serviceAdded(ServiceEvent event) {
      number.incrementAndGet();
    }

    public void serviceRemoved(ServiceEvent event) {
        number.decrementAndGet();
    }

    public void serviceResolved(ServiceEvent event) {
      executor.submit(new DS2438Connection(event.getInfo()));
    }
}
```

```
static class DS2438Connection implements Runnable {

  public void run(){
    // connexion avec le serveur associé
}
```

executor.shutdown();

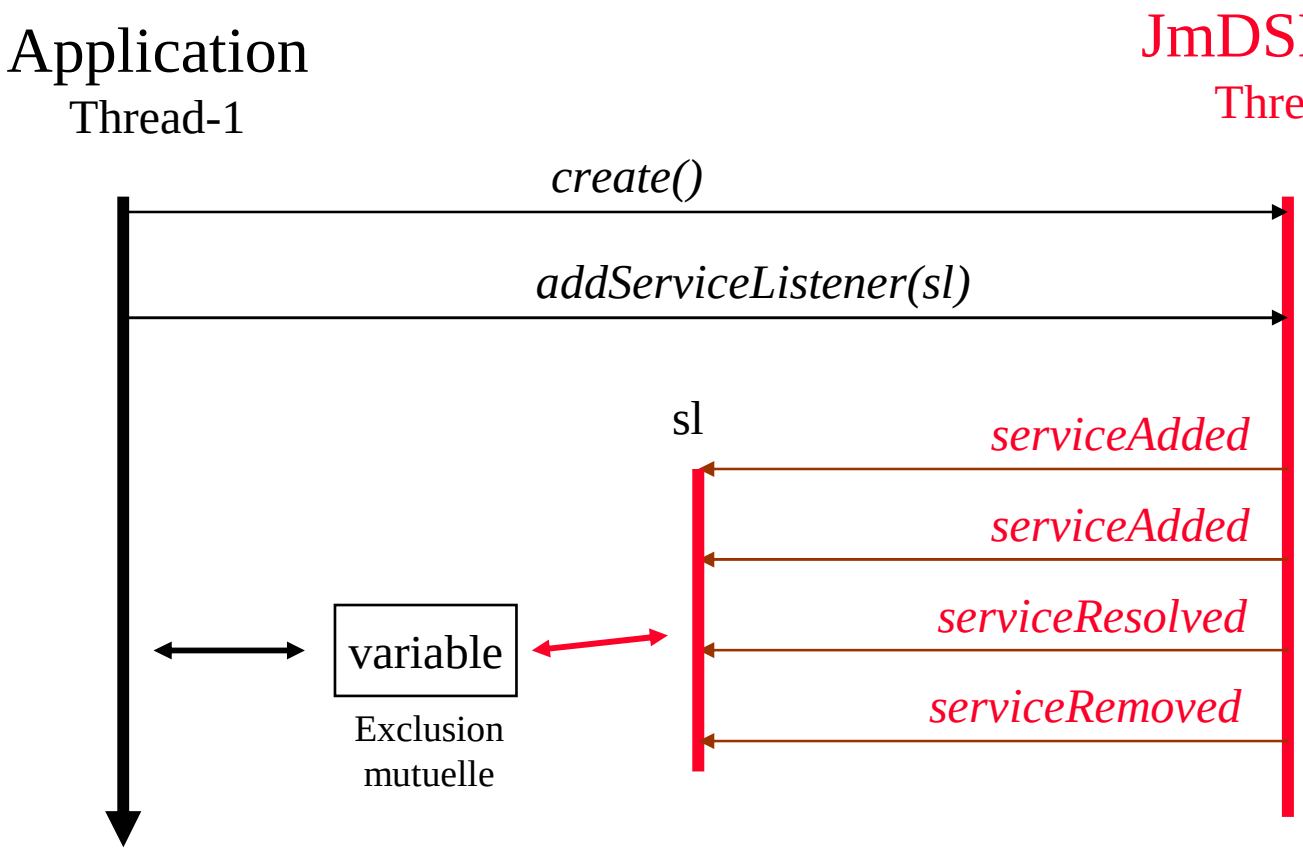
Schéma de programmes … précaution

Hypothèse : **Méthodes appelées par JmDNSImpl depuis un autre Thread**

```
static class SampleListener implements ServiceListener {
   @Override
   public void serviceAdded(ServiceEvent event) {
      le plus court en temps d'exécution est requis
   }

   public void serviceRemoved(ServiceEvent event) {
      le plus court en temps d'exécution est requis
   }

   public void serviceResolved(ServiceEvent event) {
      le plus court en temps d'exécution est requis

   }
}
```

Cf. en annexe les précautions à prendre pour du swing et android

« Architecture » en asynchrone, bis



- **Application, découverte des services**
  - **Deux threads, mécanisme du « callback » : un classique**
  - **Voir java.util.concurrent.atomic.***
  - **et Collections.synchronizedXXX**

Publication, création d'un service

```
JmDNS jmdns = JmDNS.create();

ServiceInfo info;
String type = "_http._tcp.local."  ;
String nom = "un_nom_unique_de_service",
info = ServiceInfo.create(type, nom, 8080, "index.html");

jmdns.registerService(info);

// paramètres supplémentaires
Map<String, String> map = new HashMap<,>();
map.put("cle", "valeur");
info = ServiceInfo.create(type, "un_nom_de_service", 8080, map);

jmdns.registerService(info);
```

Une autre création

```
JmDNS jmdns = JmDNS.create();

ServiceInfo info;
String type = "_http._tcp.local."  ;



// poids et priorité de 0
info = ServiceInfo.create(type, "un_nom_de_service", 8080, 0, 0,
  "index.html");



jmdns.registerService(info);
```

Encore une création d'un service

```
jmdns = JmDNS.create(address, name);
ServiceInfo serviceInfo =
  ServiceInfo.create(Constants.SERVICE_TYPE,
              Constants.SERVICE_NAME,
              Constants.ZEROCONF_PORT,
              "mon service");

HashMap<String, String> deviceInfoMap = new HashMap<,>();

deviceInfoMap.put(Constants.KEY_DEVICE_NAME,getDeviceName());

deviceInfoMap.put(Constants.KEY_DEVICE_ID, getDeviceId()); // ...

serviceInfo.setText(deviceInfoMap);

jmdns.registerService(serviceInfo);
```

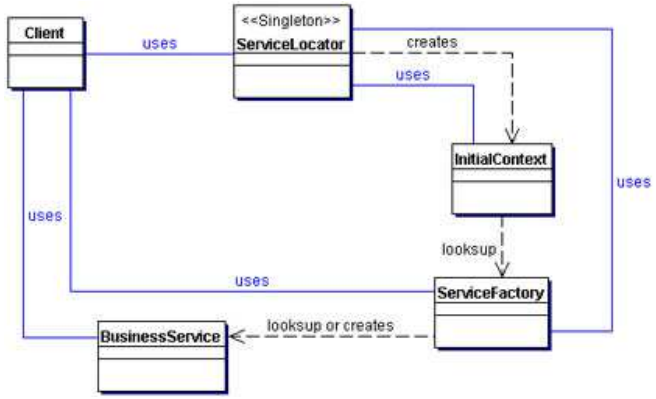
Les constantes instructives

```
// http://www.iana.org/assignments/dns-parameters

// changed to final class - jeffs
public static final String MDNS_GROUP                   = "224.0.0.251";
public static final String MDNS_GROUP_IPV6              = "FF02::FB";
public static final int    MDNS_PORT                    = Integer.parseInt(System.getProperty("net.mdns.port", "5353"));
public static final int    DNS_PORT                     = 53;
public static final int    DNS_TTL                      = 60 * 60;                                          // default one hour TTL
// public static final int DNS_TTL = 120 * 60; // two hour TTL (draft-cheshire-dnsext-multicastdns.txt ch 13)

public static final int    MAX_MSG_TYPICAL              = 1460;
public static final int    MAX_MSG_ABSOLUTE             = 8972;

public static final int    FLAGS_QR_MASK                = 0x8000;                                           // Query response mask
public static final int    FLAGS_QR_QUERY               = 0x0000;                                           // Query
public static final int    FLAGS_QR_RESPONSE            = 0x8000;                                           // Response

public static final int    FLAGS_AA                     = 0x0400;                                           // Authorative answer
public static final int    FLAGS_TC                     = 0x0200;                                           // Truncated
public static final int    FLAGS_RD                     = 0x0100;                                           // Recursion desired
public static final int    FLAGS_RA                     = 0x8000;                                           // Recursion available

public static final int    FLAGS_Z                      = 0x0040;                                           // Zero
public static final int    FLAGS_AD                     = 0x0020;                                           // Authentic data
public static final int    FLAGS_CD                     = 0x0010;                                           // Checking disabled
```

- http://jmdns.sourceforge.net/clover/javax/jmdns/impl/constants/DNSConstants.html

Le patron ServiceLocator

- **Contexte:**
  - La recherche et la création de services sur un réseau nécessitent des interfaces souvent complexes. De multiples API sont utilisés.
- **Le patron Service Locator**
  - se comporte comme une façade et masque la complexité sous jacente
  - http://www.oracle.com/technetwork/java/servicelocator-137181.html



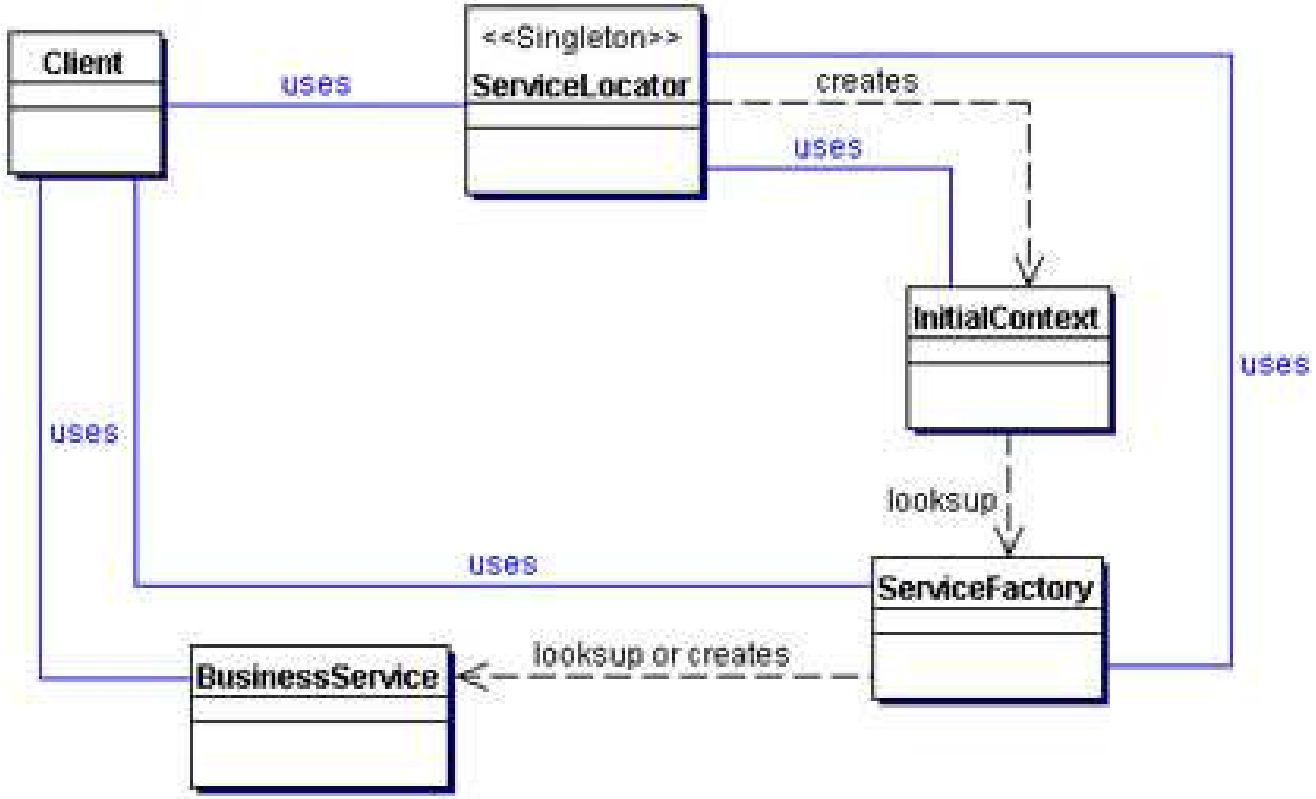
Page suivante

Le patron Service Locator

- **Le patron Service Locator**
  - **Localiser, sélectionner un service selon les besoins d'une application**
    - **accès de manière uniforme**
      - **Par exemple (DNS, LDAP, JNDI API)**
  - **Un Singleton en général …**
    - **Une table, un cache, un espace de nommage**

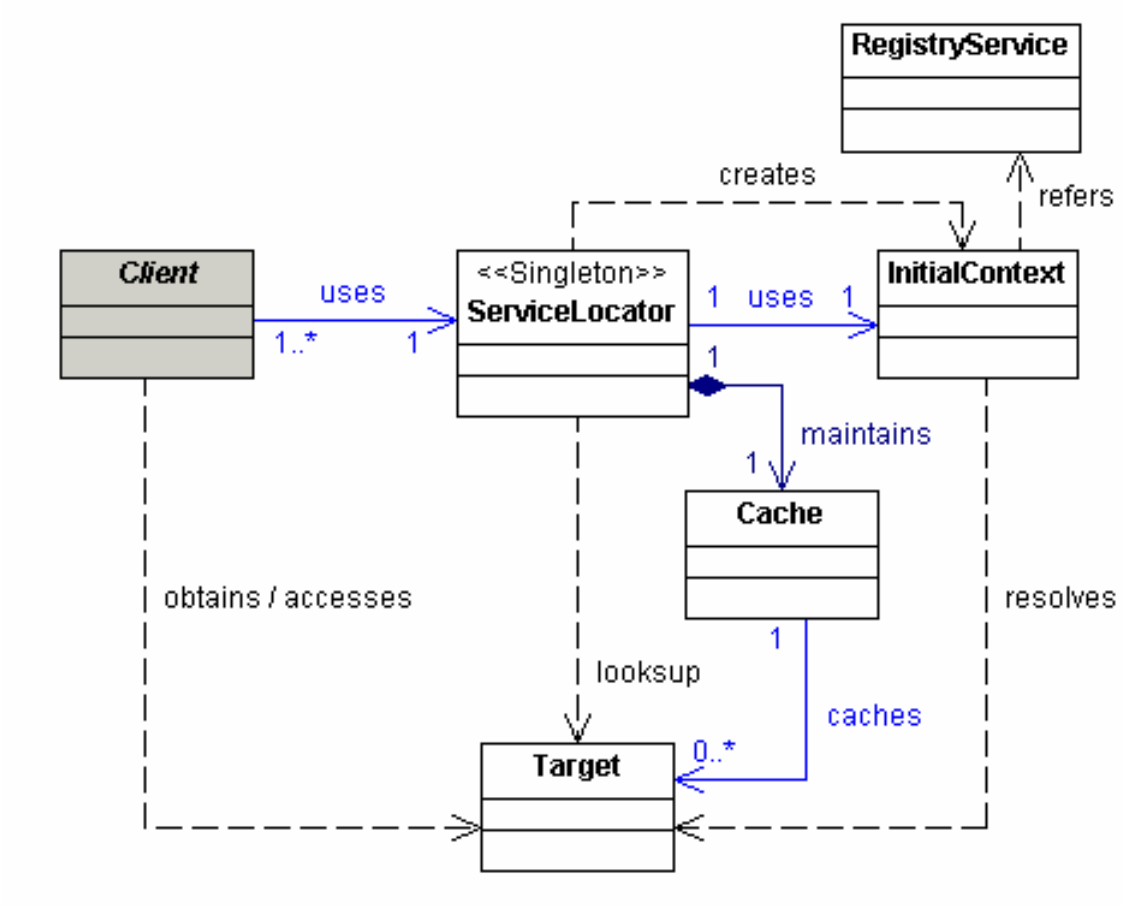
- Diapositives extraites en grande partie du cours de NFP121
  - http://jfod.cnam.fr/NFP121/ cours sur l'injection de dépendance

Service Locator : une façade



- **BusinessService ... as jmdns ....**

Patron ServiceLocator



- **Source : http://www.corej2eepatterns.com/Patterns2ndEd/ServiceLocator.htm**

ServiceLocator, jmdns un schéma possible

```
public final class ServiceLocator implements ….{
  private Map<String,XXX> map; // la table

  private static ServiceLocator instance;
  static{
    instance = new ServiceLocator(); // Cf. Singleton
  }

  public static ServiceLocator getInstance(){
    return instance;
  }


  public Object lookup(String name) throws Exception{
    XXX data = map.get(name);
    …
    return data;
  }
```

ServiceLocator jmdns un schéma possible

```
 private ServiceLocator() {
    this.map = Collections.synchronizedMap(new HashMap<String, XXXX >()); // concurrence
}
```

```
  private class XXXXListener implements ServiceListener {

     public void serviceAdded(ServiceEvent event) {
         map.put(event.getName(), xxxx);
      }

      public void serviceRemoved(ServiceEvent event) {
         map.remove(event.getName(), xxxx);
      }

      public void serviceResolved(ServiceEvent event) {
        // accès aux info, via event
       }
    }
```

Les constantes instructives

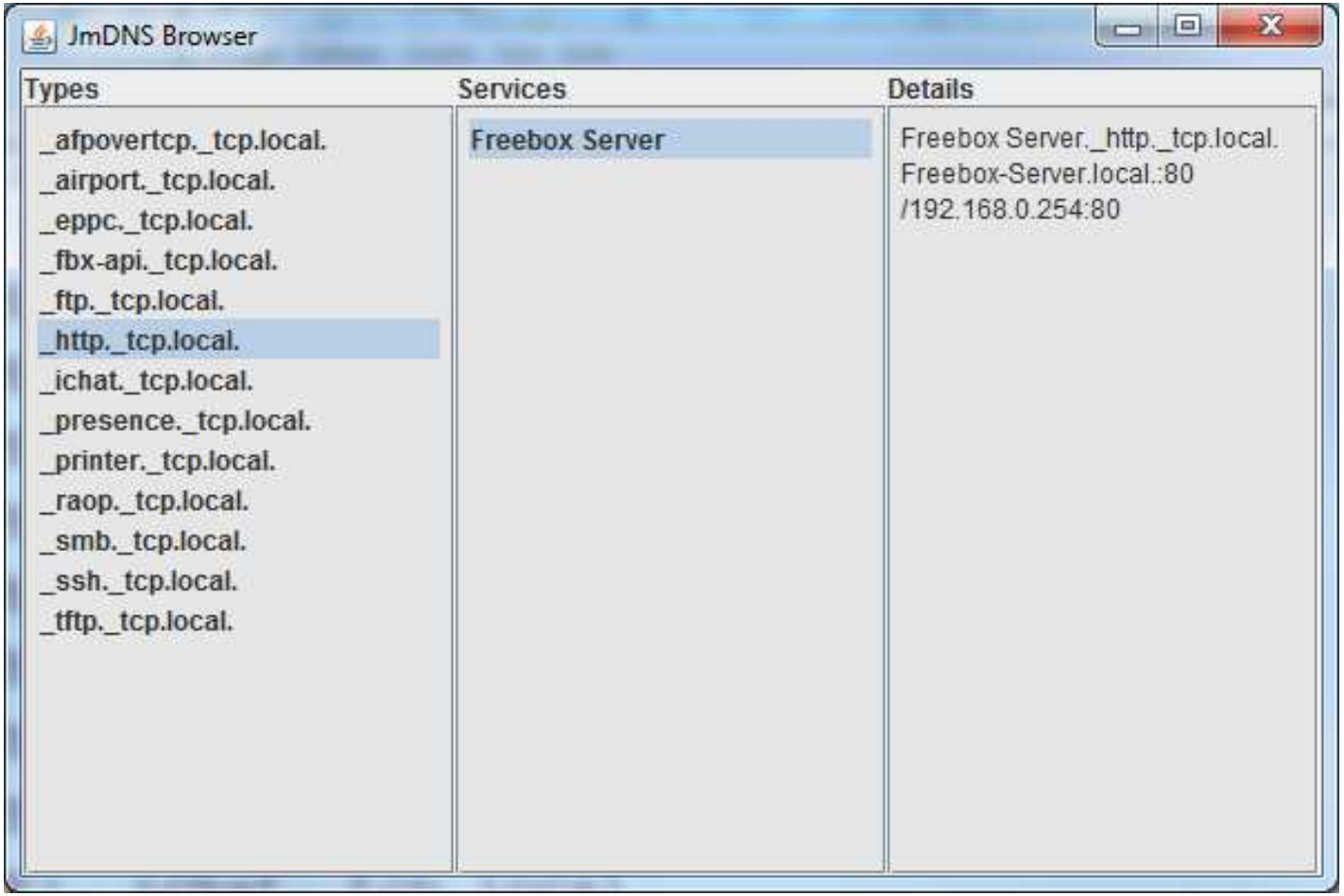
```
// Time Intervals for various functions

public static final int    SHARED_QUERY_TIME              = 20;                    // milliseconds before send shared query
public static final int    QUERY_WAIT_INTERVAL            = 225;                   // milliseconds between query loops.
public static final int    PROBE_WAIT_INTERVAL            = 250;                   // milliseconds between probe loops.
public static final int    RESPONSE_MIN_WAIT_INTERVAL     = 20;                    // minimal wait interval for response.
public static final int    RESPONSE_MAX_WAIT_INTERVAL     = 115;                   // maximal wait interval for response
public static final int    PROBE_CONFLICT_INTERVAL        = 1000;                  // milliseconds to wait after conflict.
public static final int    PROBE_THROTTLE_COUNT           = 10;                    // After x tries go 1 time a sec. on probes.
public static final int    PROBE_THROTTLE_COUNT_INTERVAL  = 5000;                  // We only increment the throttle count, if the previous incremen
public static final int    ANNOUNCE_WAIT_INTERVAL         = 1000;                  // milliseconds between Announce loops.
public static final int    RECORD_REAPER_INTERVAL         = 10000;                 // milliseconds between cache cleanups.
public static final int    RECORD_EXPIRY_DELAY            = 1;                     // This is 1s delay used in ttl and therefore in seconds
public static final int    KNOWN_ANSWER_TTL               = 120;
public static final int    ANNOUNCED_RENEWAL_TTL_INTERVAL = DNS_TTL * 500;         // 50% of the TTL in milliseconds

public static final long   CLOSE_TIMEOUT                  = ANNOUNCE_WAIT_INTERVAL * 5L;
public static final long   SERVICE_INFO_TIMEOUT           = ANNOUNCE_WAIT_INTERVAL * 6L;

public static final int    NETWORK_CHECK_INTERVAL         = 10 * 1000;             // 10 secondes
```

- http://jmdns.sourceforge.net/clover/javax/jmdns/impl/constants/DNSConstants.html

La suite : Outils …

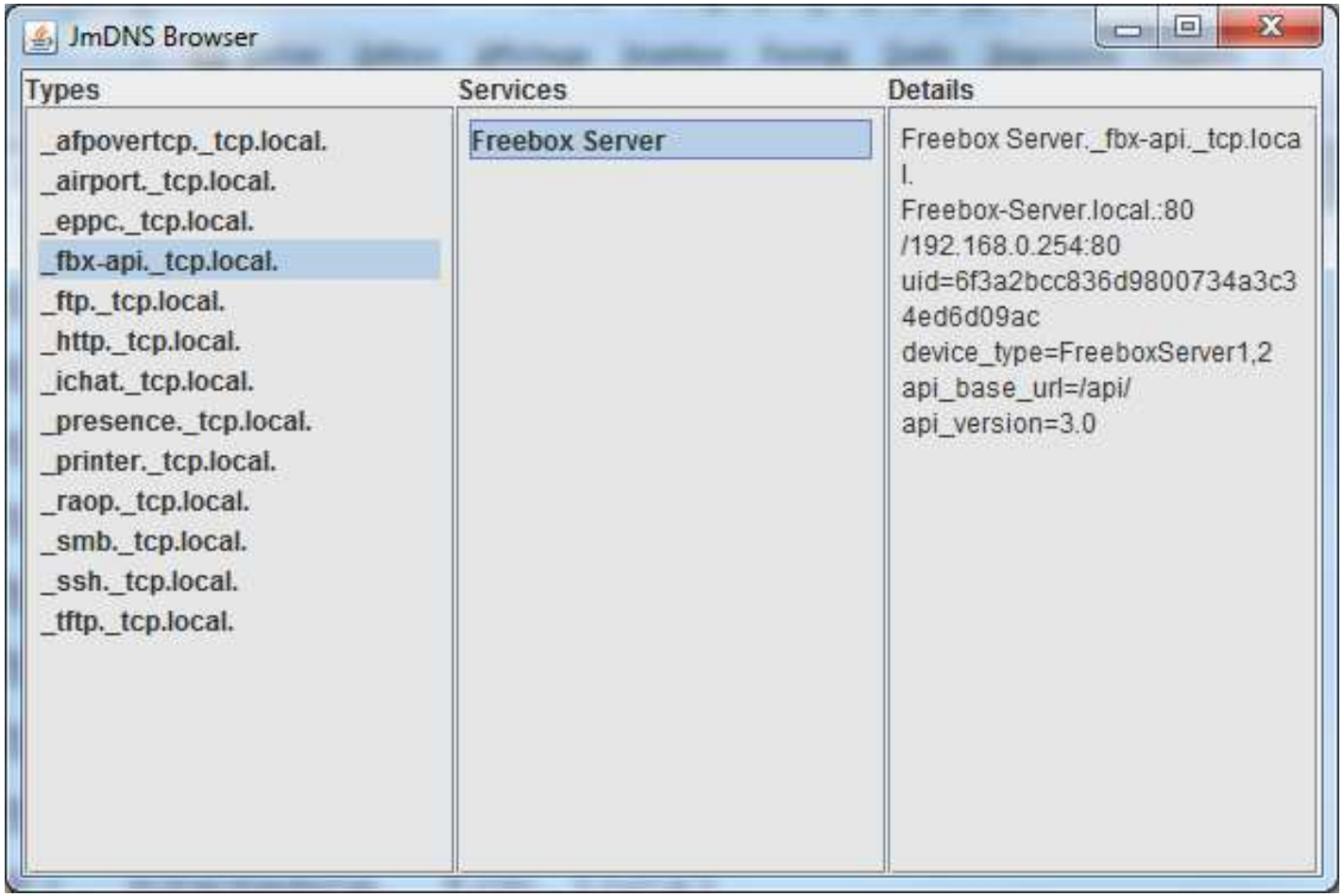
- **Une interface swing**
- **En Annexe**
  - **Règle de programmation avec deux threads**
  - **JINI**
- **…**

Un Browser



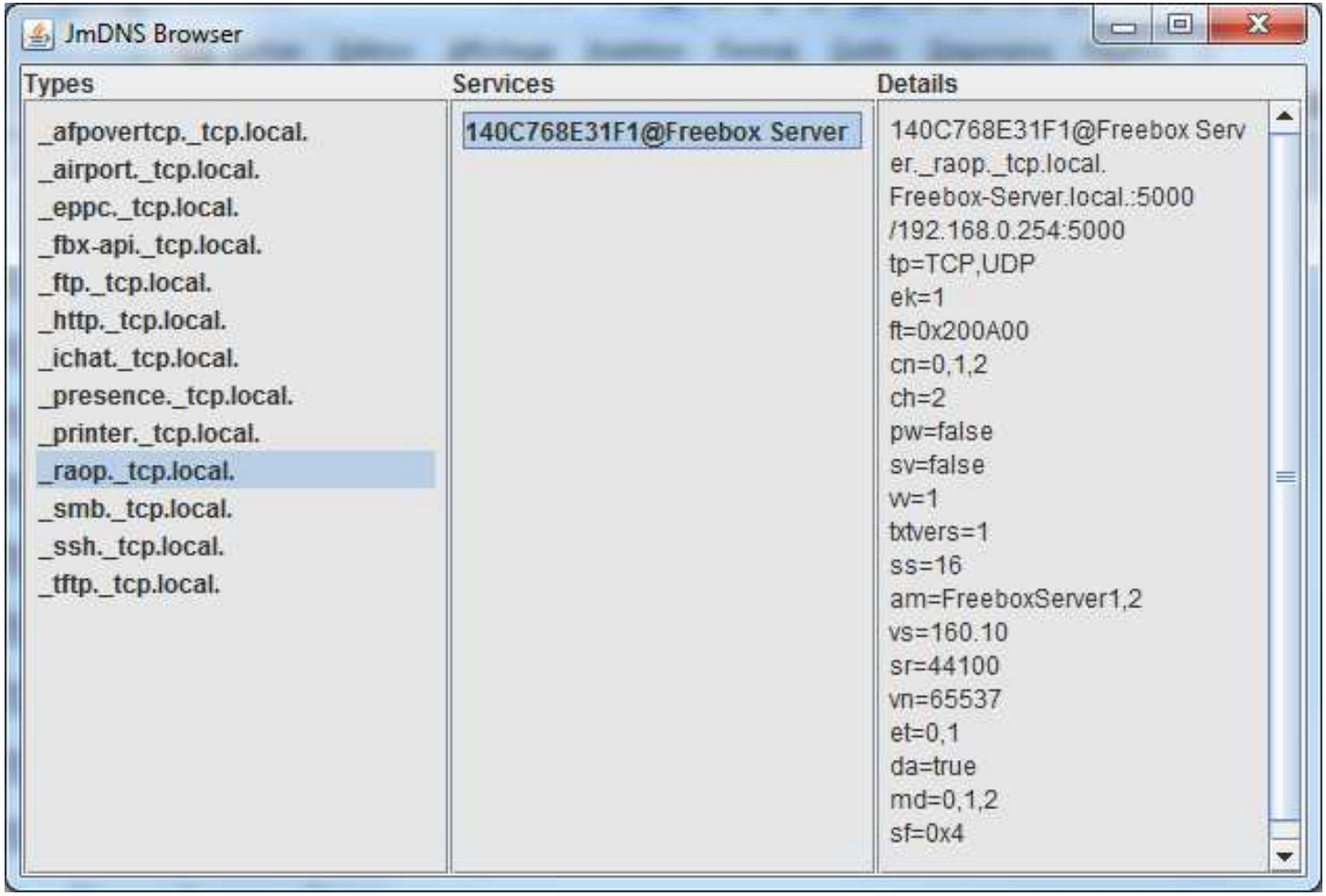
- **Ici une freebox**
  - **Le service de configuration**
- **https://github.com/jadahl/jmdns/tree/master/src**
  - **Auteurs : Arthur van Hoff, Werner Randelshofer**

Browser freebox



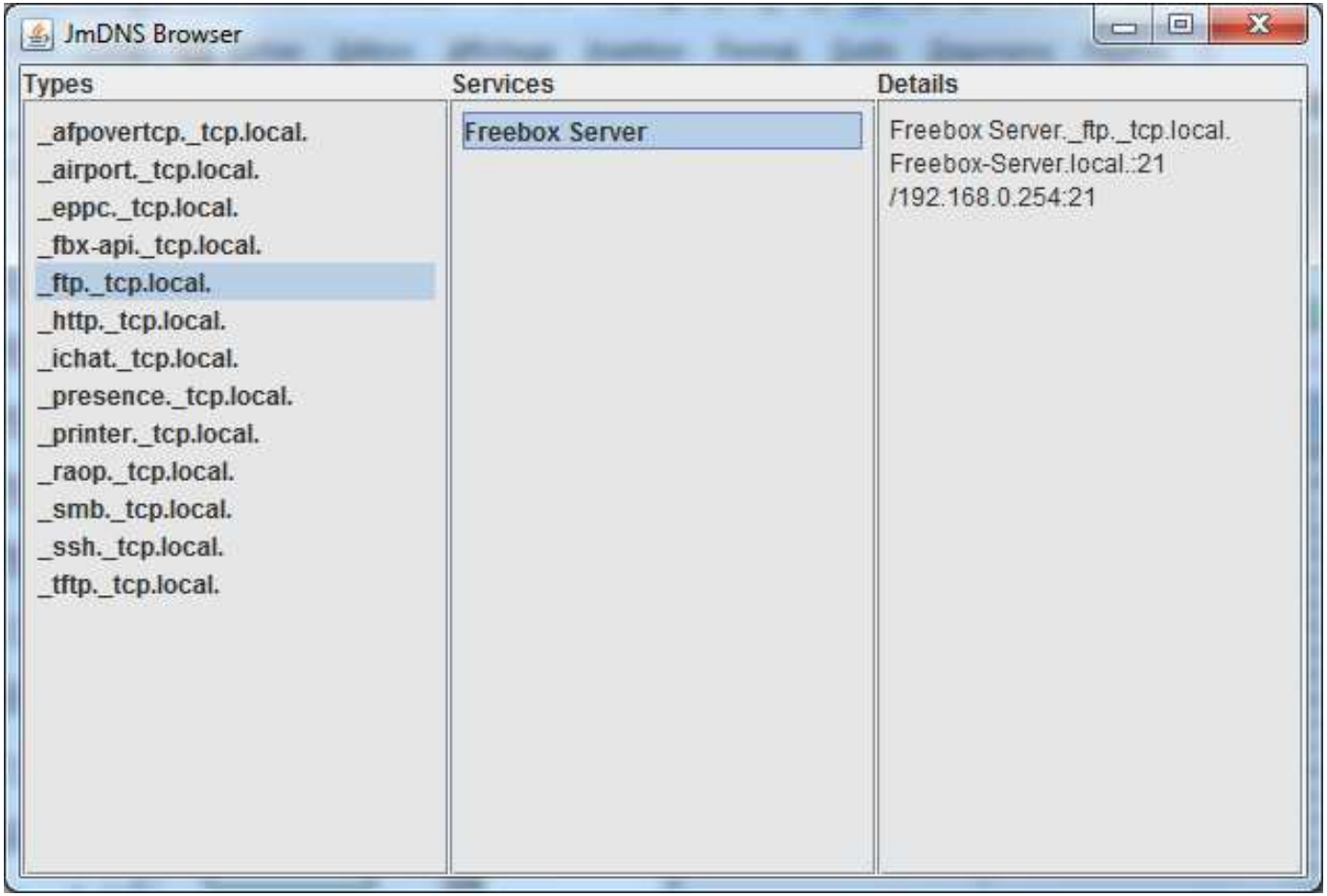
- **_fbx-api**
  - Type « propriétaire »

Browser freebox



- **_raop**

Browser freebox



- **_ftp**
  - **Un serveur ftp**
  - http://www.dns-sd.org/ServiceTypes.html

Conclusion

Annexes

- **Mise en œuvre swing et Android**

- **JINI**
  - **https://river.apache.org/doc/spec-index.html**

Annexe mise en œuvre en swing

- Implémentation des listener
  - Prendre en compte le mécanisme du callback (deux Threads)

    - En swing :
    - http://gfx.developpez.com/tutoriel/java/swing/swing-threading/

```
public void serviceAdded(ServiceEvent event) {
     final String name = event.getName();
     SwingUtilities.invokeLater(new Runnable() {
        public void run() {
               // appel d'une méthode de l'application
         }
     });
  }

public void serviceRemoved(ServiceEvent event) {
   idem
public void serviceResolved(ServiceEvent event) {
   idem
```

Annexe mise en œuvre sous android

- **Sous Android, à l'aide d'un handler* :**
  - https://developer.android.com/training/multiple-threads/communicate-ui.html

```
public void serviceAdded(ServiceEvent event) {
      final String name = event.getName();
      handler.sendEmptyMessage(ADDED);
}
public void serviceRemoved(ServiceEvent event) {
   … handler.sendEmptyMessage(REMOVED);
public void serviceResolved(ServiceEvent event) {
   … handler.sendEmptyMessage(RESOLVED);
```

Le « handler » en attribut de l'activité

```
• private Handler handler = new Handler(){
•         @Override
•         public void handleMessage(Message msg) {
```

Swing comme android un seul Thread est agréé pour accéder à l'interface utilisateur

JINI … précurseur, trop en avance ?

- **A partir de la diapositive 38, support NSY102_JINI.pdf**