Un service d'information voyageur bâti sur une infrastructure à composants

Réunion GtMob – GDR I3

01-02/12/2005, INRETS - Lille

A. Flissi (LIFL/CNRS), C. Gransart (INRETS-LEOST), P. Merle (LIFL/INRIA)

Plan

- Introduction :
  - « service d'information voyageur »
- Réalisation avec MIDP
- Réalisation selon une approche « composants » :
  - contexte
  - composants applicatifs du service
  - support d'exécution
  - nouveaux défis
  - propositions
- Conclusion

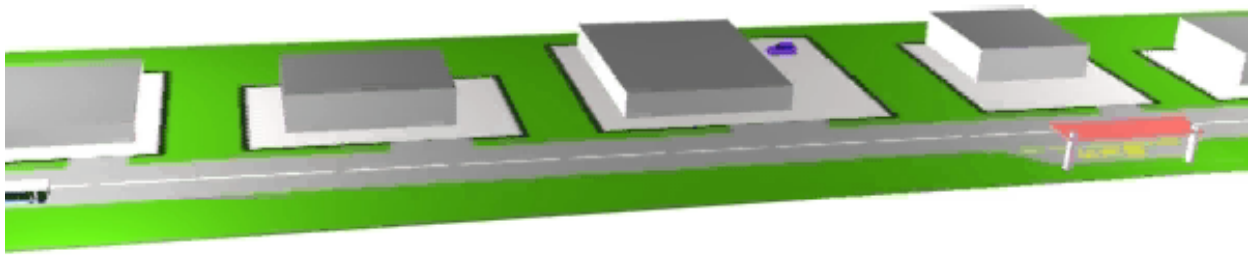
Idée de base

QUESTIONS :

- Où descendre ?
- Comment repérer les arrêts ?
- Trouver la correspondance ?

- Demander au chauffeur
- Rester attentif sur le parcours

Solution



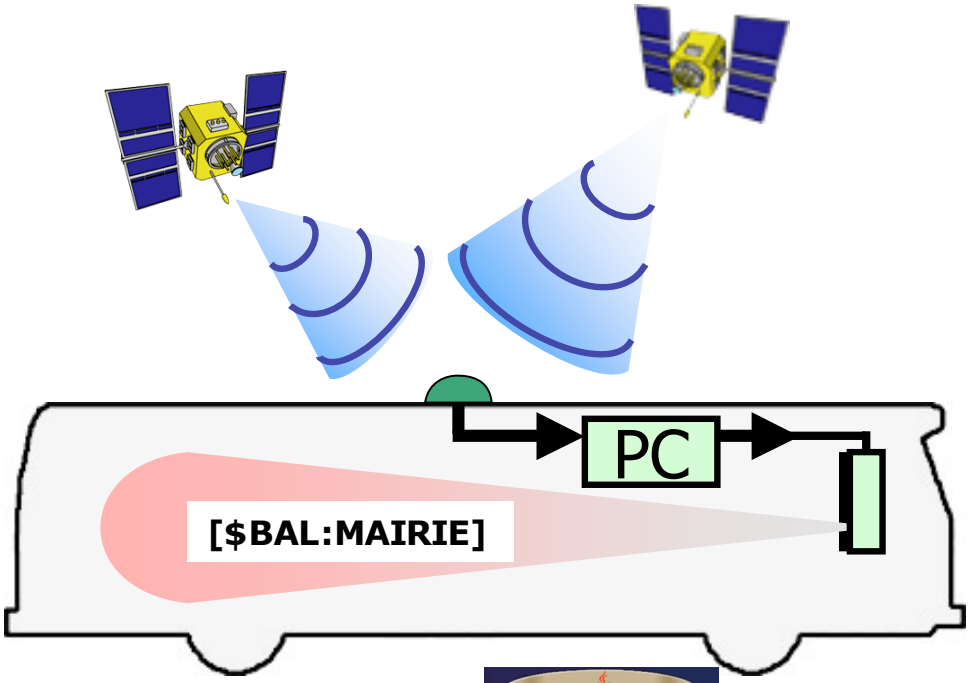
Facile à utiliser

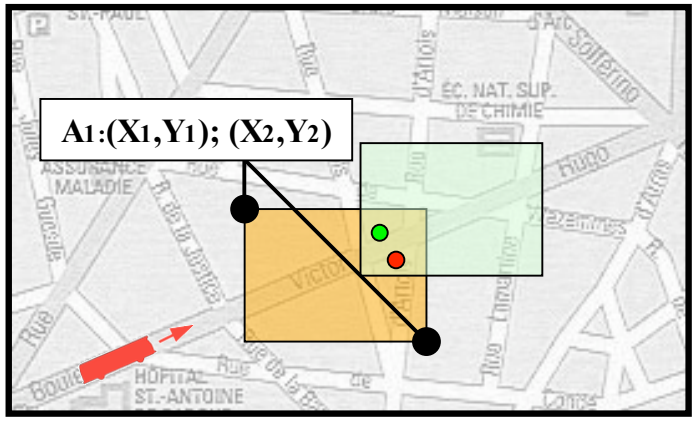
Être prévenu suffisamment tôt pour se préparer à descendre

Information sur le nom de la station

Trouver un moyen simple et discret utilisable également par des personnes à déficience auditive ou visuelle

Fonctionnement



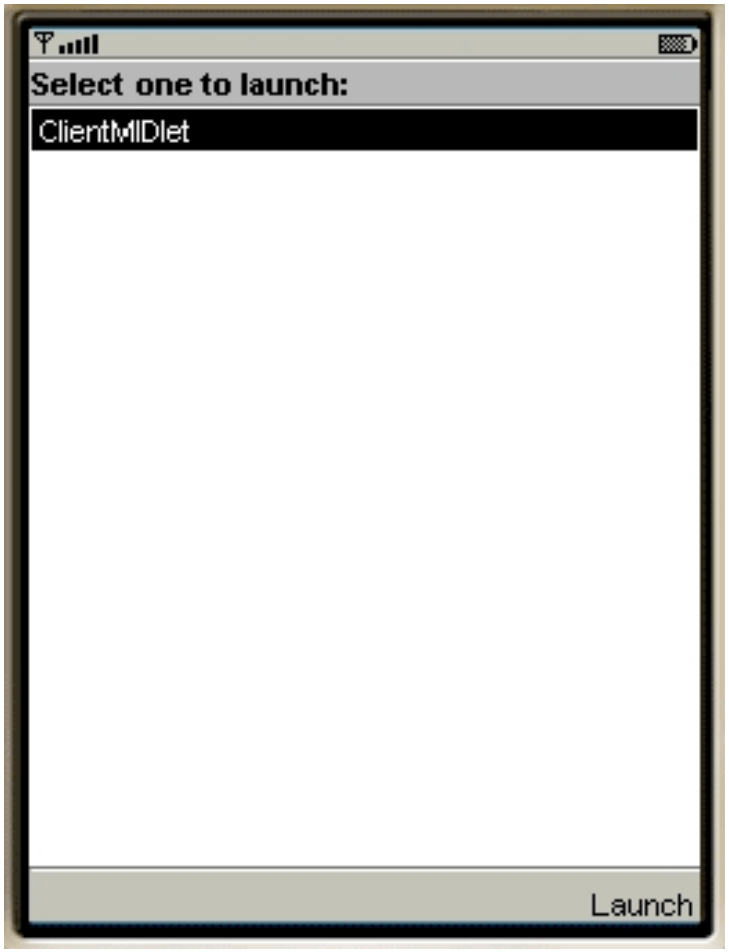



```
<xml version='1.0' encoding ='ISO-8859-1' ?>

<ICAU>
  <localisation sens="0">
   <pointUL>
     <latitude>50.73298</latitude>
     <longitude>3.02658</longitude>
   </pointUL>
   <pointDR>
     <latitude>50.78445</latitude>
     <longitude>3.08565</longitude>
   </pointDR>
  </localisation>

  <info>
    <ligne>12A</ligne>
    <arret>colbert</arret>
  </info>
 </ICAU>
```

Réalisation avec MIDP

- Uniquement TCP et UDP
  - Réalisation d'un protocole propriétaire
- Déploiement à la main
- Téléphone Java + Wi-Fi ???
- Fonctionne sur émulateur

Réalisation avec MIDP

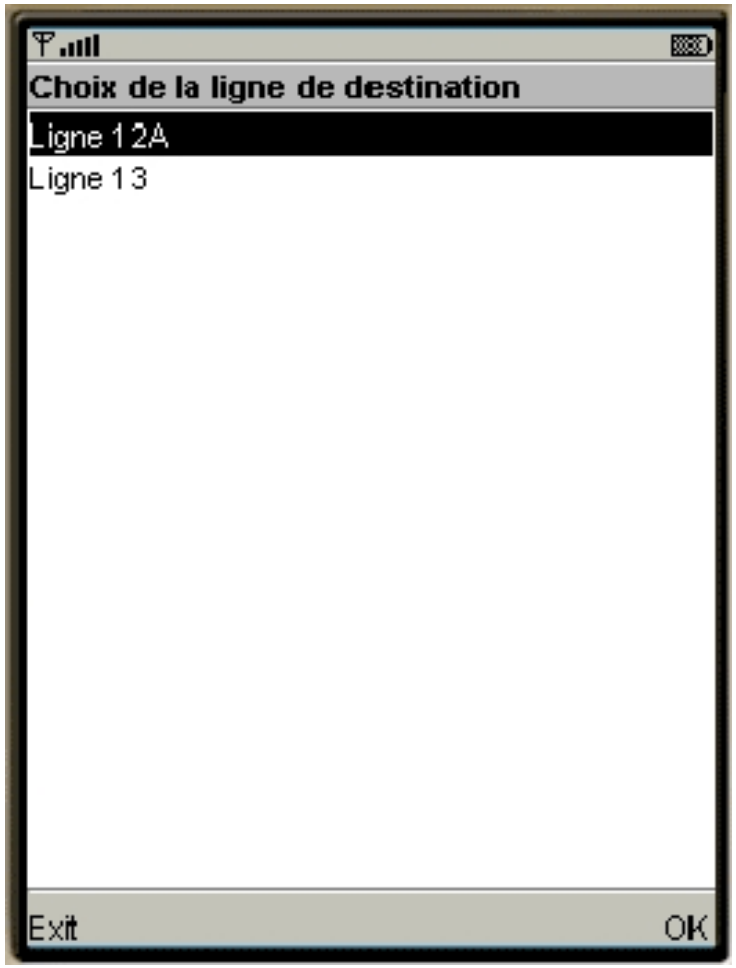


Réalisation avec MIDP

Réalisation avec MIDP

Réalisation avec MIDP

Réalisation avec MIDP



Réalisation avec MIDP

Réalisation avec MIDP

Réalisation avec MIDP

Réalisation avec MIDP

Réalisation avec MIDP

Conception du service à l'aide d'une approche composants

- Conception du service :
  - Service contextuel, i.e. fonction de la localisation géographique (bus et utilisateur)
  - Approche à base de composants logiciels répartis sur différents nœuds :
    - poste de commande central (régie de transport),
    - serveur (bus),
    - terminal utilisateur

Cadre du travail : les services contextuels

Hors couverture

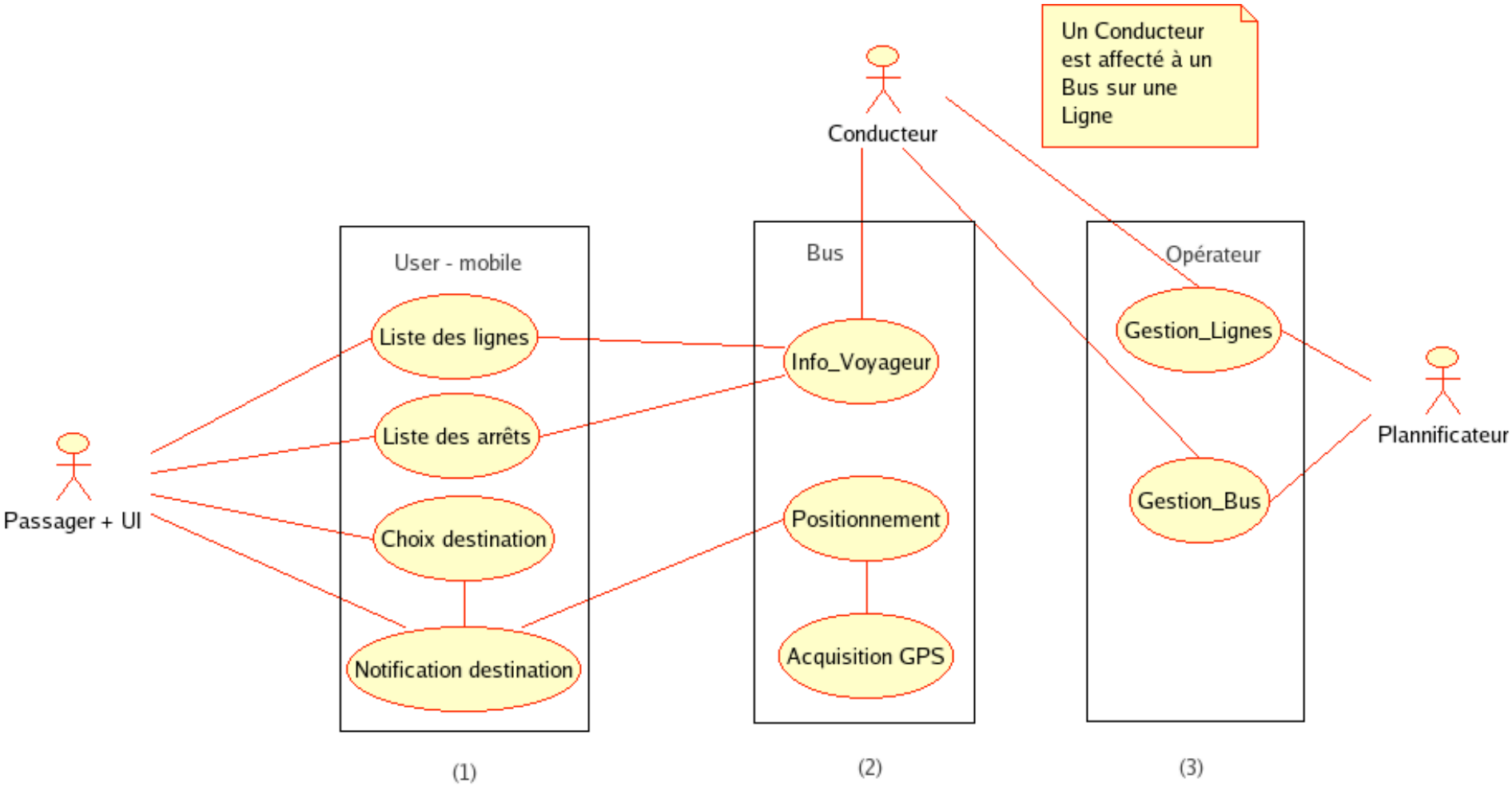
Apparition des services

Utilisation des services

Sauvegarde Offline Désinstallation

Réf: UbiMob'05 : A. Flissi, C. Gransart, P. Merle, Une Infrastructure à Composants pour des Applications Ubiquitaires

Architecture générale du service



Support d'exécution ?

- Challenges :
  - Hétérogénéité (matérielle/logicielle), interopérabilité (multi-vendeurs), répartition
  - Portabilité (infrastructure, applications)
  - Ressources limitées (mémoire, autonomie énergétique…)
  - Découverte dynamique des services contextuels & déploiement automatisé du logiciel
- Réponse :
  - Adaptation (statique) aux capacités des terminaux

=> Middleware à Composants

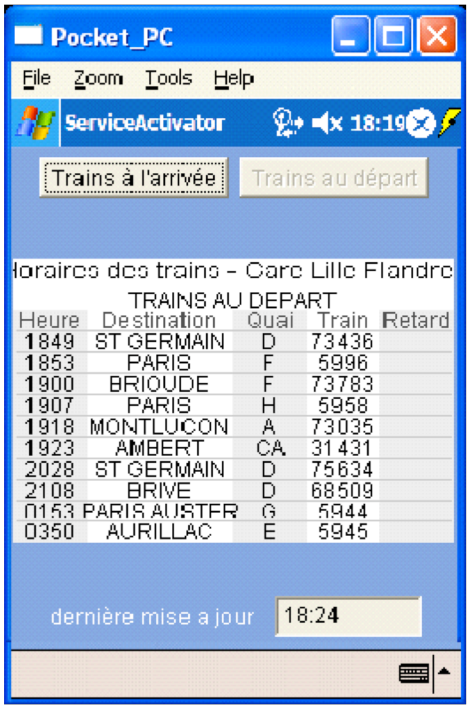
Support d'exécution

- Challenges :
  - Hétérogénéité (matérielle/logicielle), interopérabilité (multi-vendeurs), répartition :
    - modèle de Composants CORBA
  - Portabilité (infrastructure, applications) :
    - implantation en Java
  - Ressources limitées (mémoire, autonomie énergétique…) :
    - mode transmission/réception, protocole multicast, dimensionnement du middleware
  - Découverte dynamique des services contextuels & déploiement automatisé du logiciel :
    - composants « systèmes » dédiés
  - Adaptation (statique) aux capacités des terminaux :
    - => *OpenCCM Mosaïques* : Prototype d'implantation
    - présentation aux utilisateurs des seuls assemblages adaptés

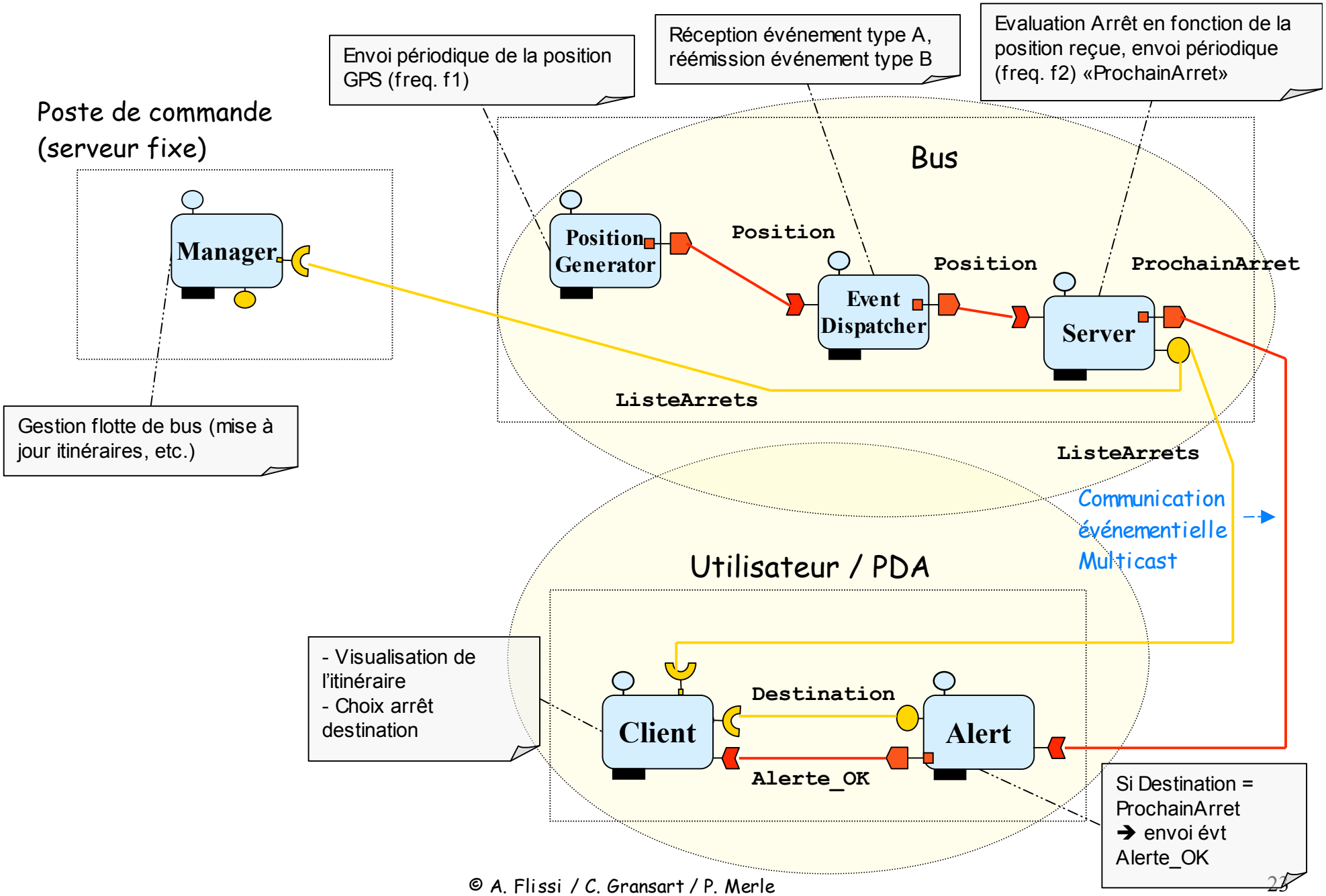
Illustration, capture d'écran (exemple service INRETS / Mosaïques)

1- Découverte et déploiement du service

2- Utilisation du service



Modélisation des composants applicatifs



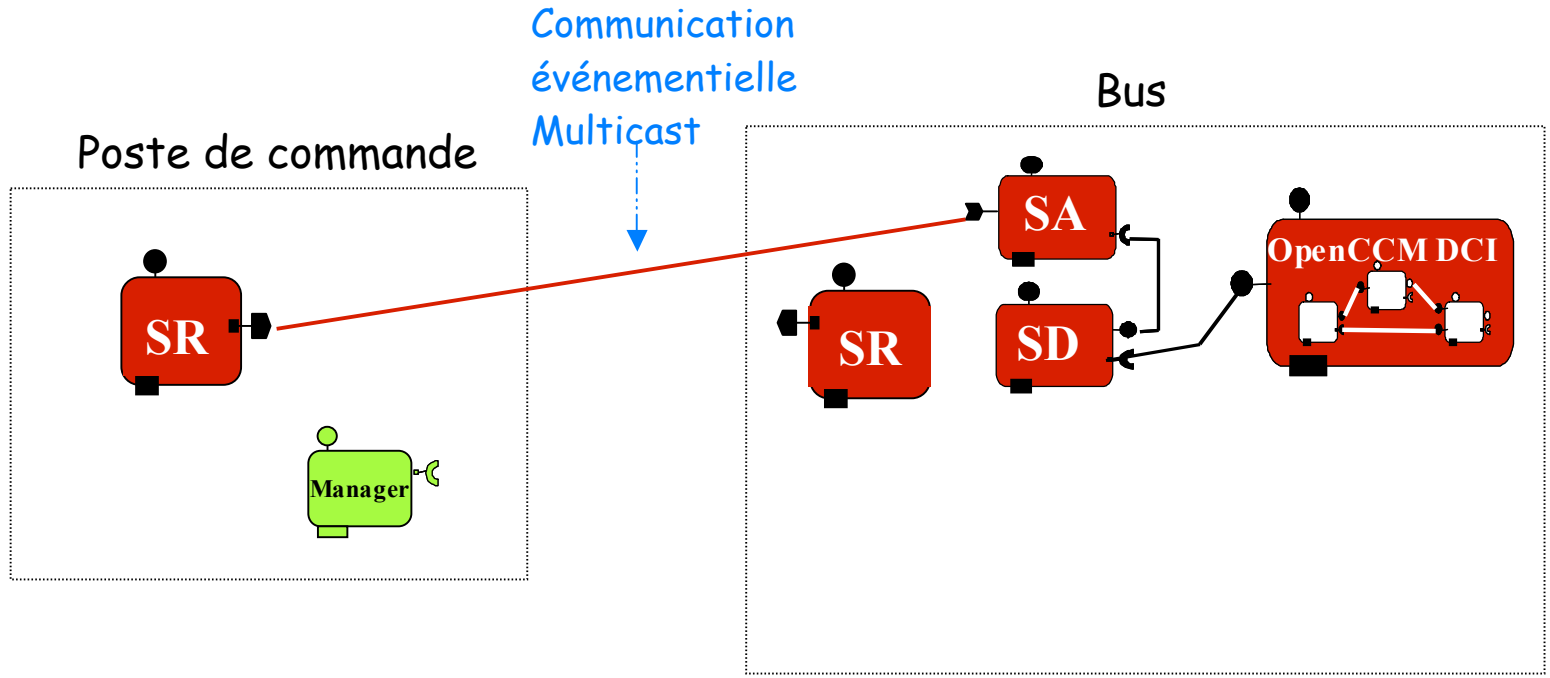
Nouveaux défis

- Déploiement incrémental de services
- Pré-déploiement / installation du middleware
- Dimensionnement du middleware p/r plate-forme cible
- Re-configuration de services
- Applications ubiquitaires & respect de la vie privée

Réponses aux défis (1)

- Déploiement incrémental des différents "services" & composants
  - Serveur central : composant *Manager* préalablement déployé
  - Bus découvre le "service" : déploiement des composants *PositionGenerator*, *EventDispatcher* et *Server*
  - Utilisateur / PDA découvre le "service" : déploiement des composants *Client* et *Alert*.

Déploiement incrémental du service « Alerte Arrêt » (1)

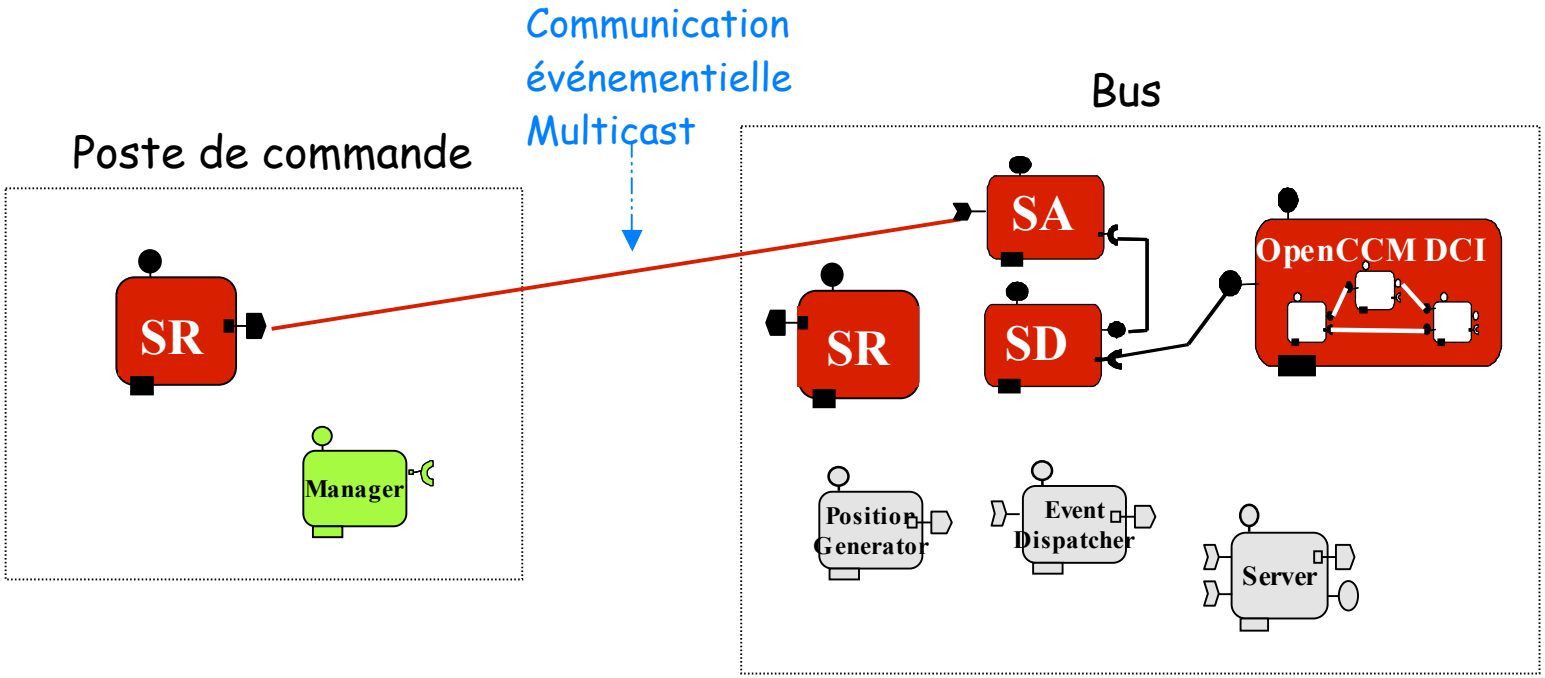


1

- Infrastructure *OpenCCM Mosaïques* (découverte dynamique et déploiement automatique de services) démarrée

- Déploiement préalable du composant *Manager*

Déploiement incrémental du service « Alerte Arrêt » (2)



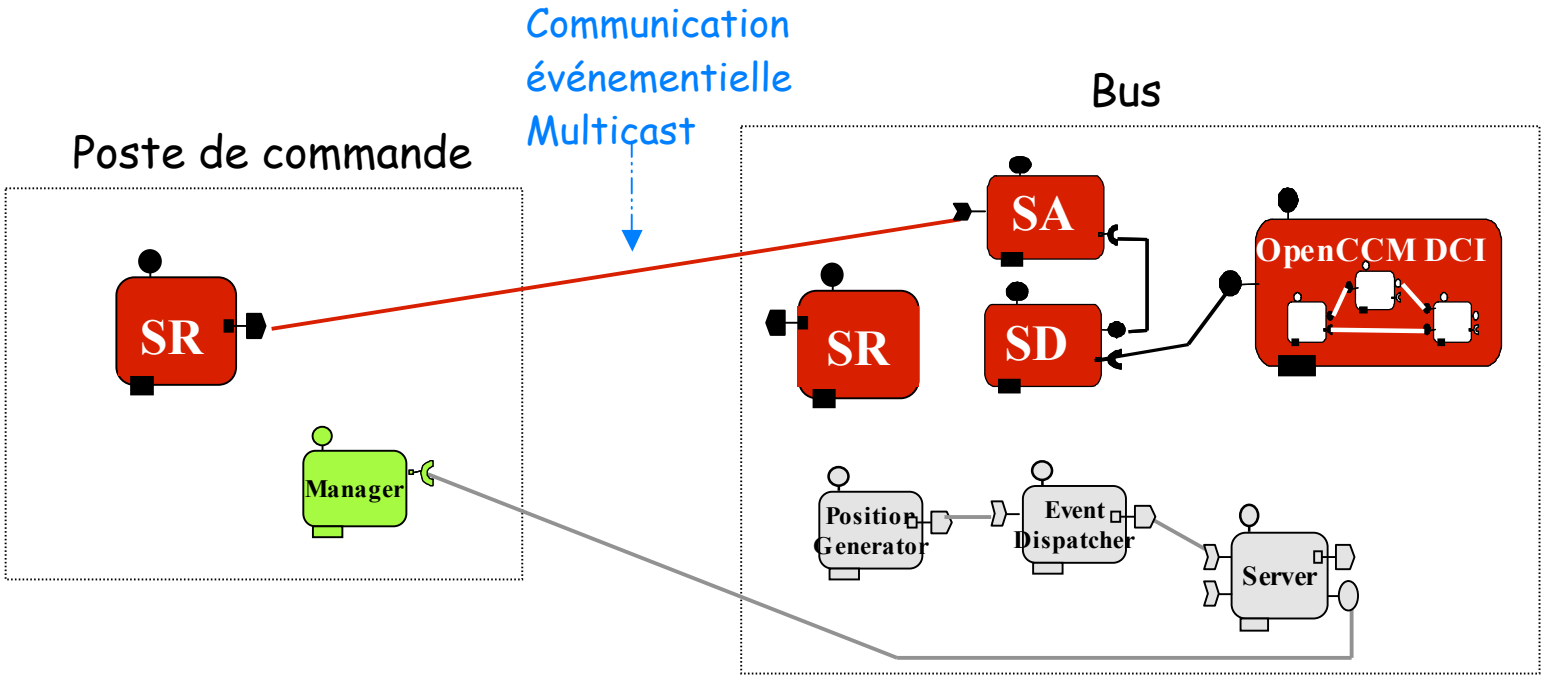
2

- Le bus « découvre » le service

- Déploiement des composants *PositionGenerator* et *Server*, interconnexion avec *Manager*

a) Téléchargement et instanciation des composants (assemblage « mobile »)

Déploiement incrémental du service « Alerte Arrêt » (2)



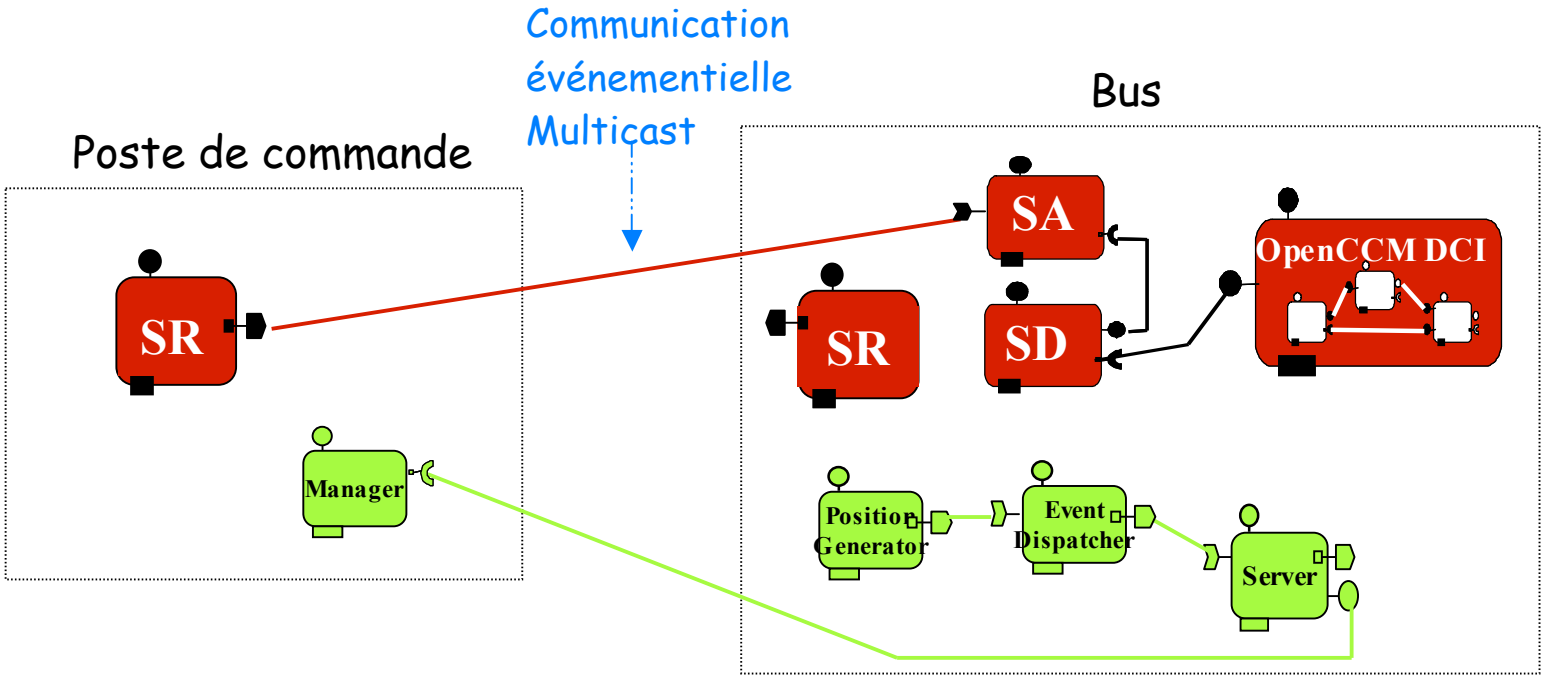
2

- Le bus « découvre » le service

- Déploiement des composants *PositionGenerator* et *Server*, interconnexion avec *Manager*

b) Interconnexion avec l'assemblage « fixe », configuration des propriétés métiers

Déploiement incrémental du service « Alerte Arrêt » (2)



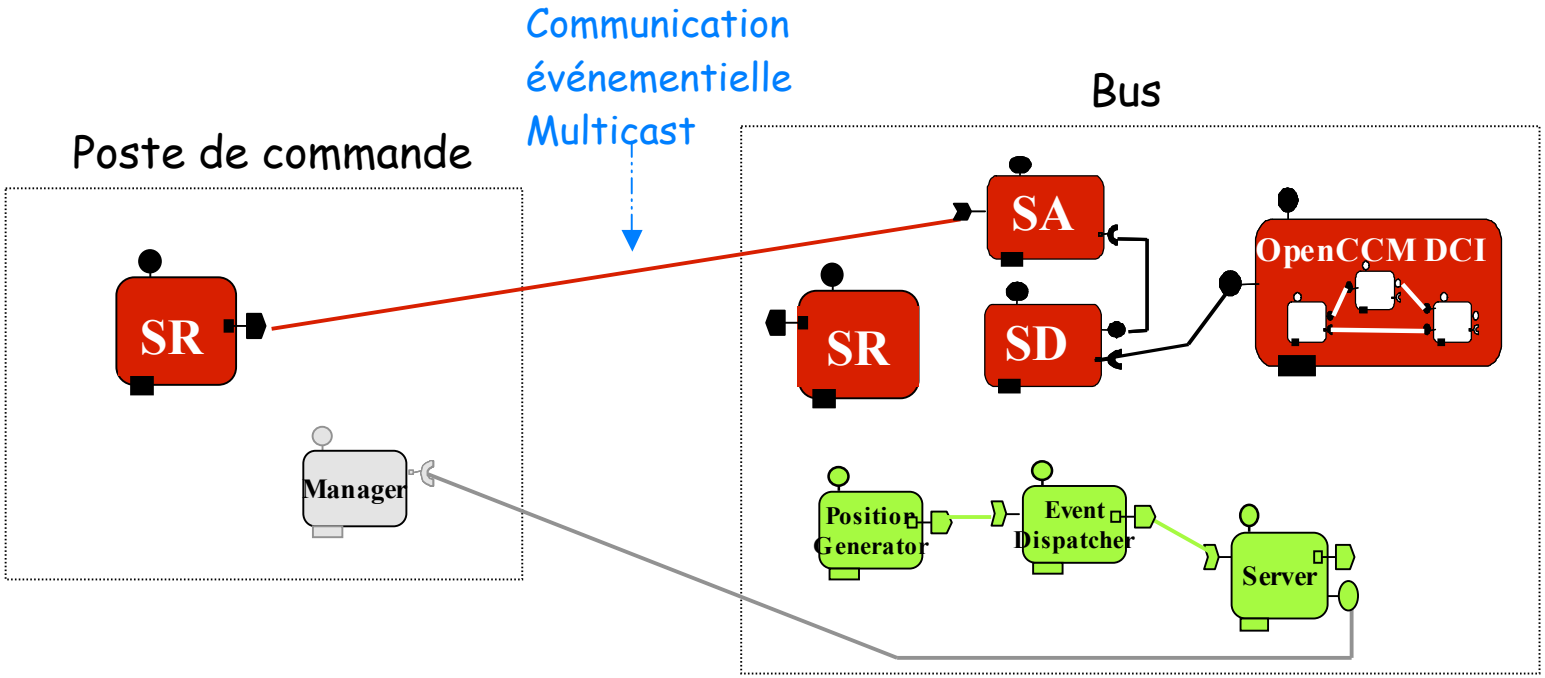
2

- Le bus « découvre » le service

- Déploiement des composants *PositionGenerator* et *Server*, interconnexion avec *Manager*

c) Activation des composants

Déploiement incrémental du service « Alerte Arrêt » (2)



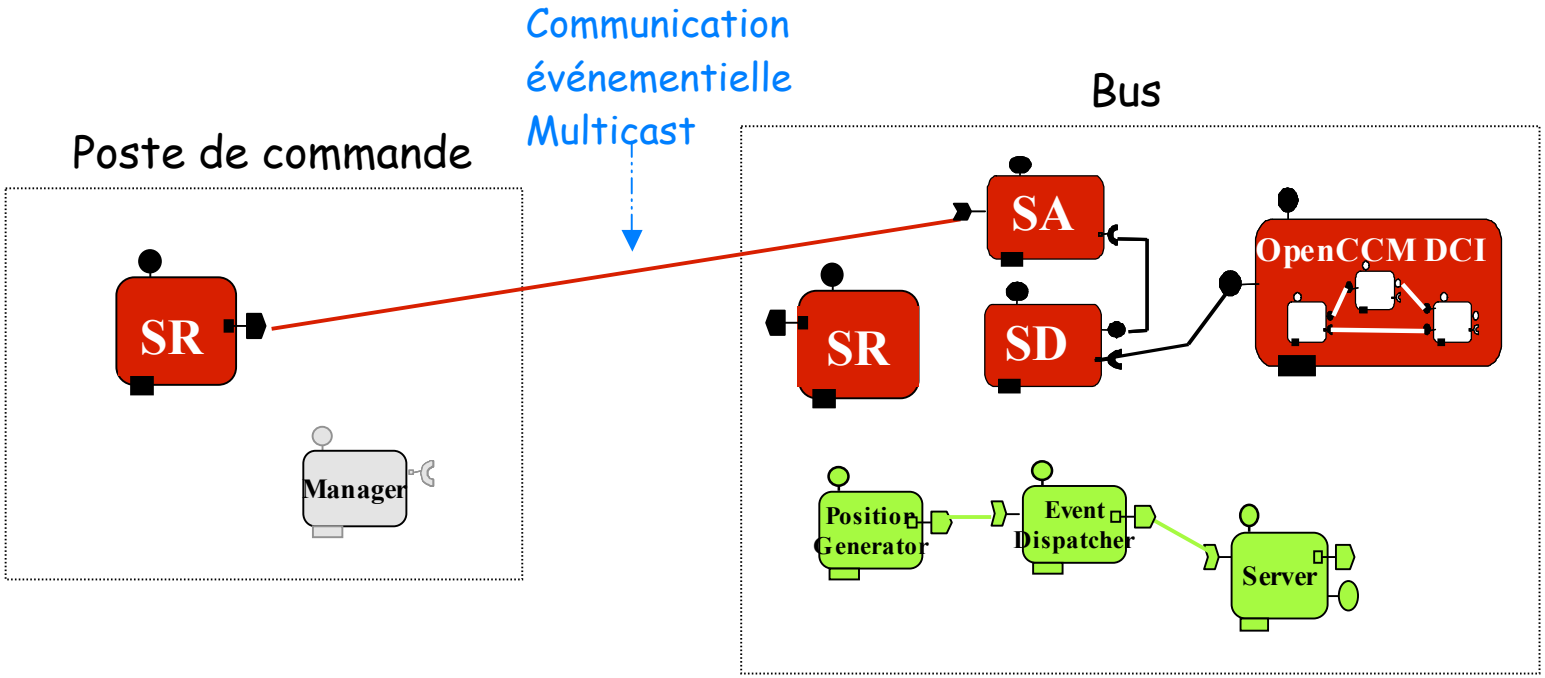
2

- Le bus « découvre » le service

- Déploiement des composants *PositionGenerator* et *Server*, interconnexion avec *Manager*

d) Désactivation composants

Déploiement incrémental du service « Alerte Arrêt » (2)



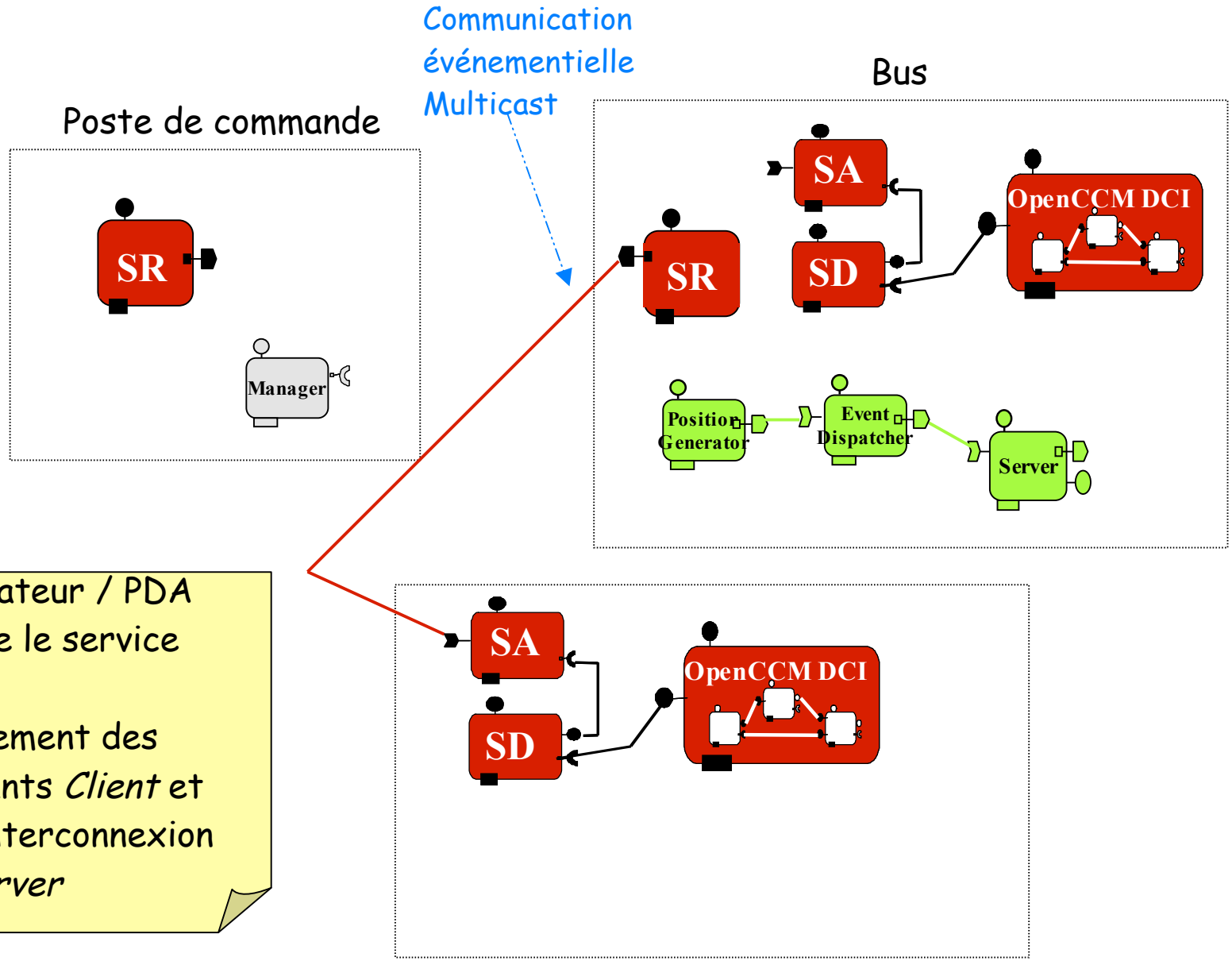
2

- Le bus « découvre » le service

- Déploiement des composants *PositionGenerator* et *Server*, interconnexion avec *Manager*

e) Suppression interconnexion composants

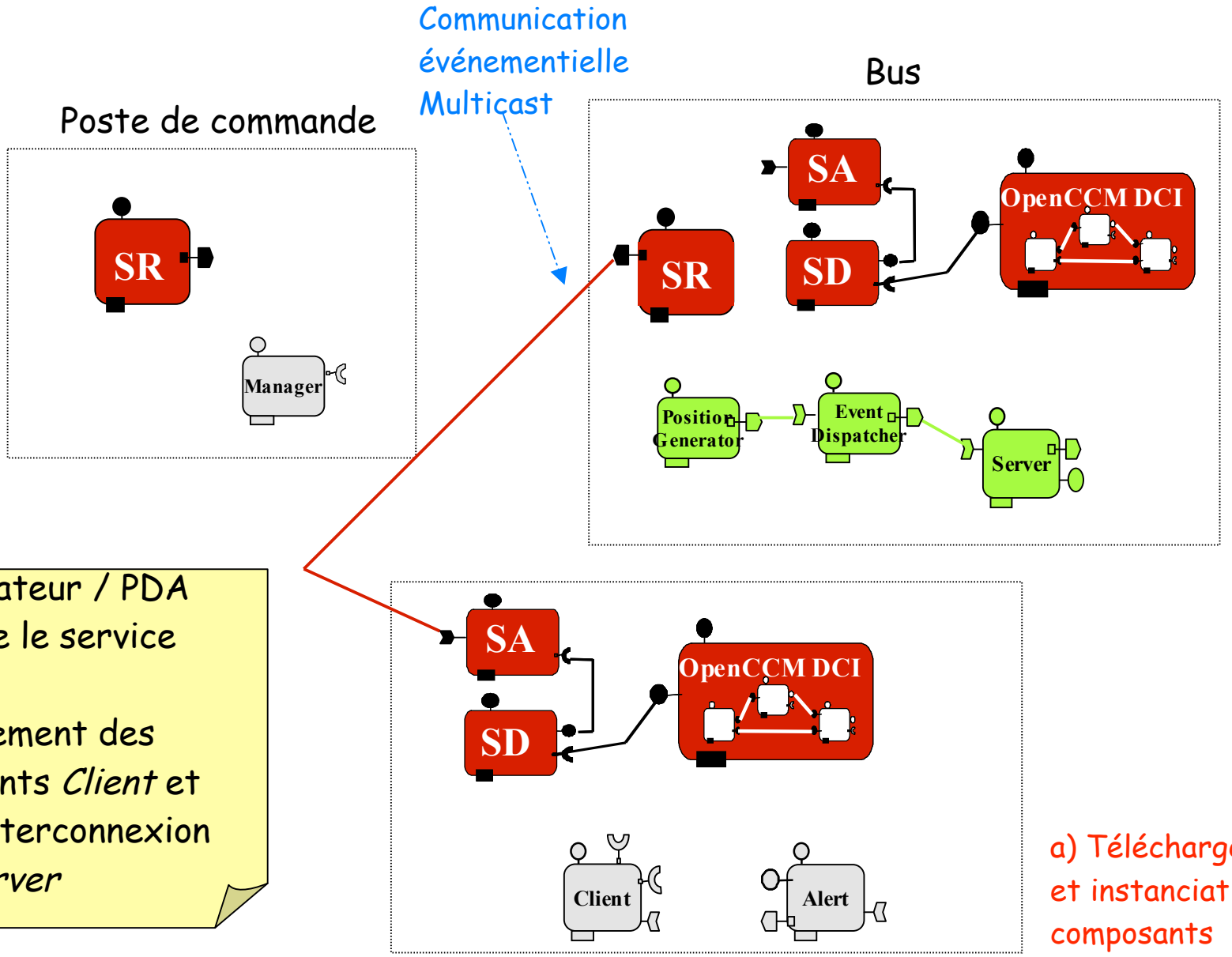
Déploiement incrémental du service « Alerte Arrêt » (3)



3

- L'utilisateur / PDA découvre le service
- Déploiement des composants *Client* et *Alert*, interconnexion avec *Server*

Déploiement incrémental du service « Alerte Arrêt » (3)

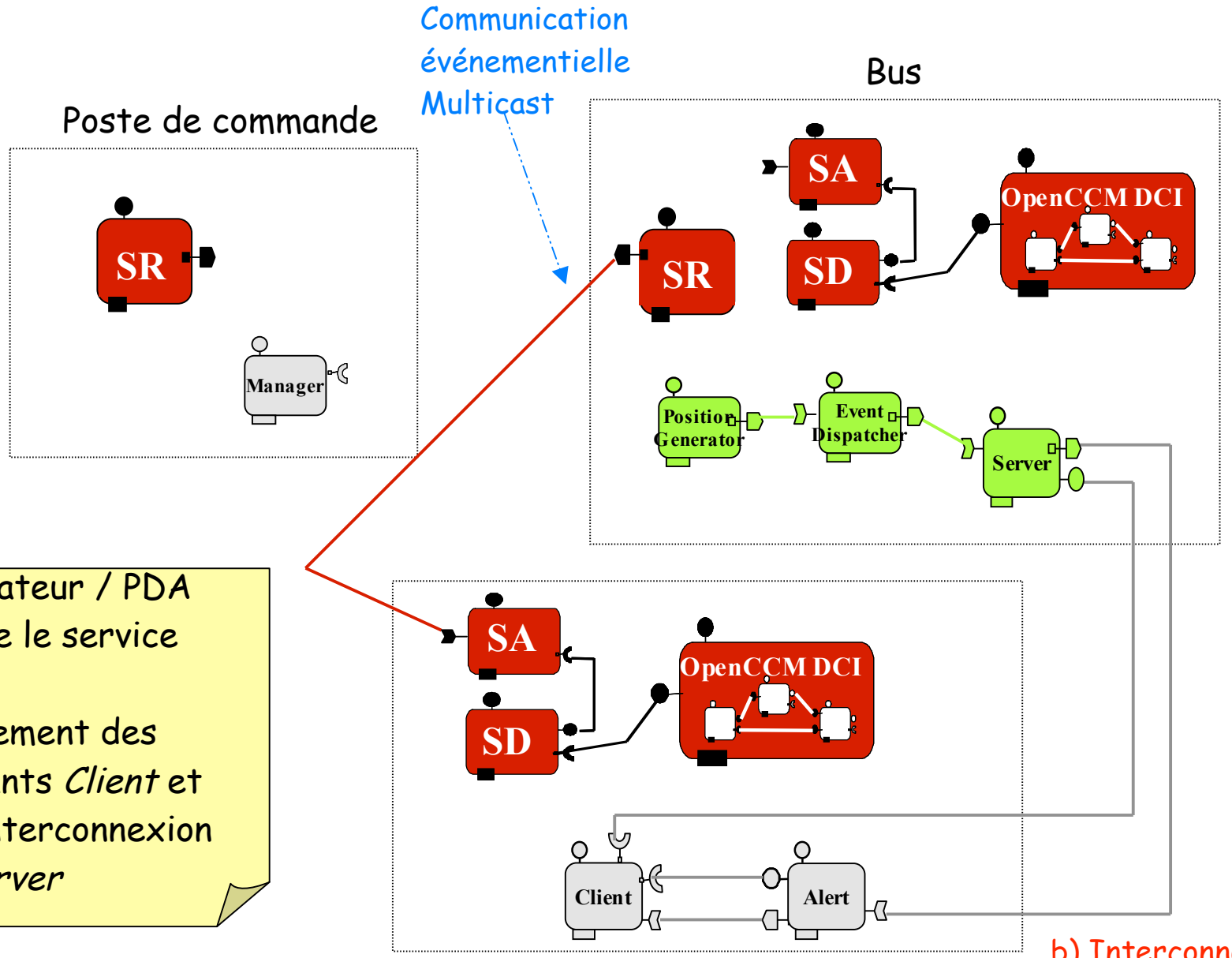


3

- L'utilisateur / PDA découvre le service
- Déploiement des composants *Client* et *Alert*, interconnexion avec *Server*

a) Téléchargement et instanciation des composants (assemblage « mobile »)

Déploiement incrémental du service « Alerte Arrêt » (3)

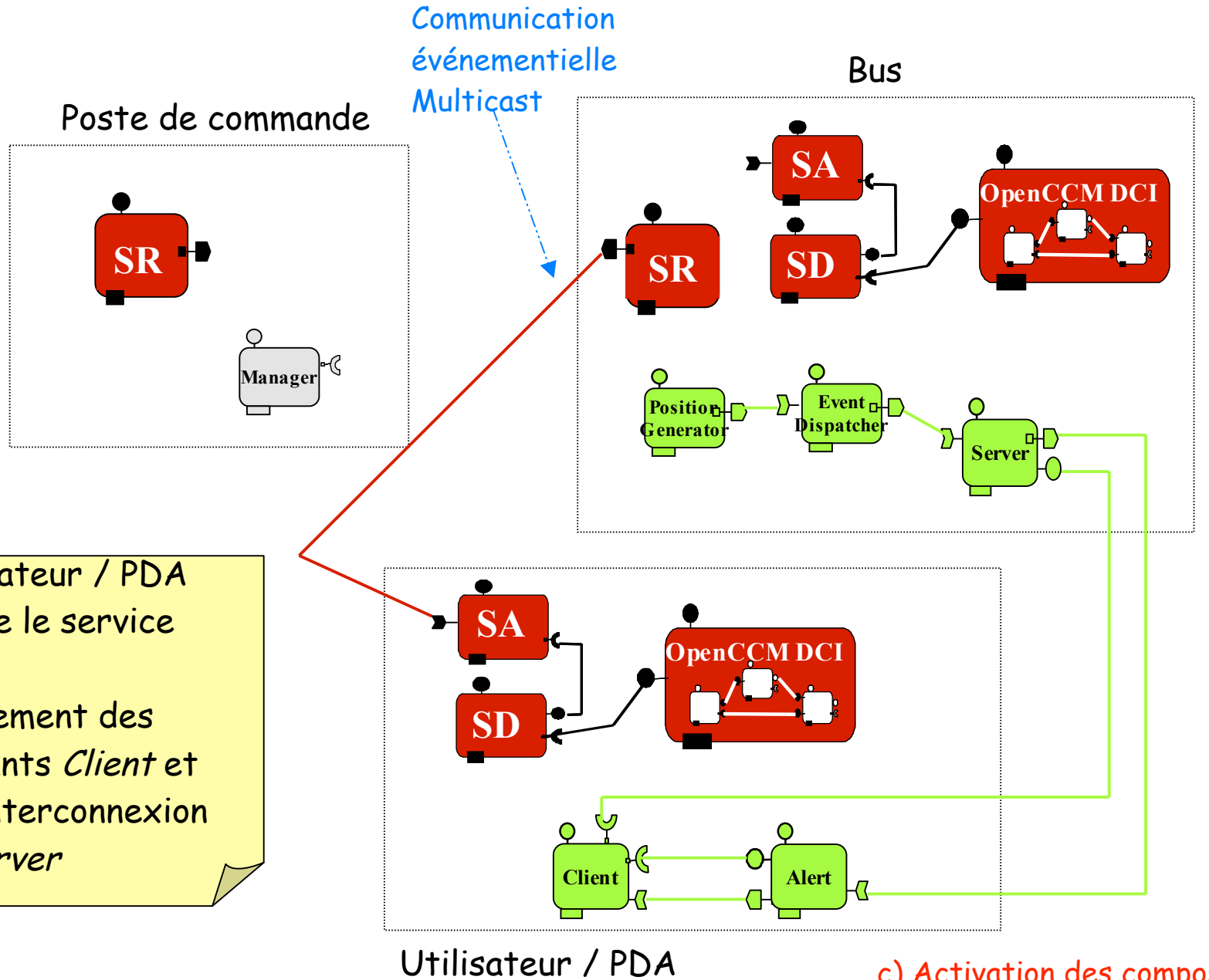


3

- L'utilisateur / PDA découvre le service
- Déploiement des composants *Client* et *Alert*, interconnexion avec *Server*

b) Interconnexion avec l'assemblage « fixe », configuration

Déploiement incrémental du service « Alerte Arrêt » (3)



3

- L'utilisateur / PDA découvre le service
- Déploiement des composants *Client* et *Alert*, interconnexion avec *Server*

c) Activation des composants

Réponses aux défis (2, 3)

- Pré-déploiement/installation du middleware lui-même :

  Q:

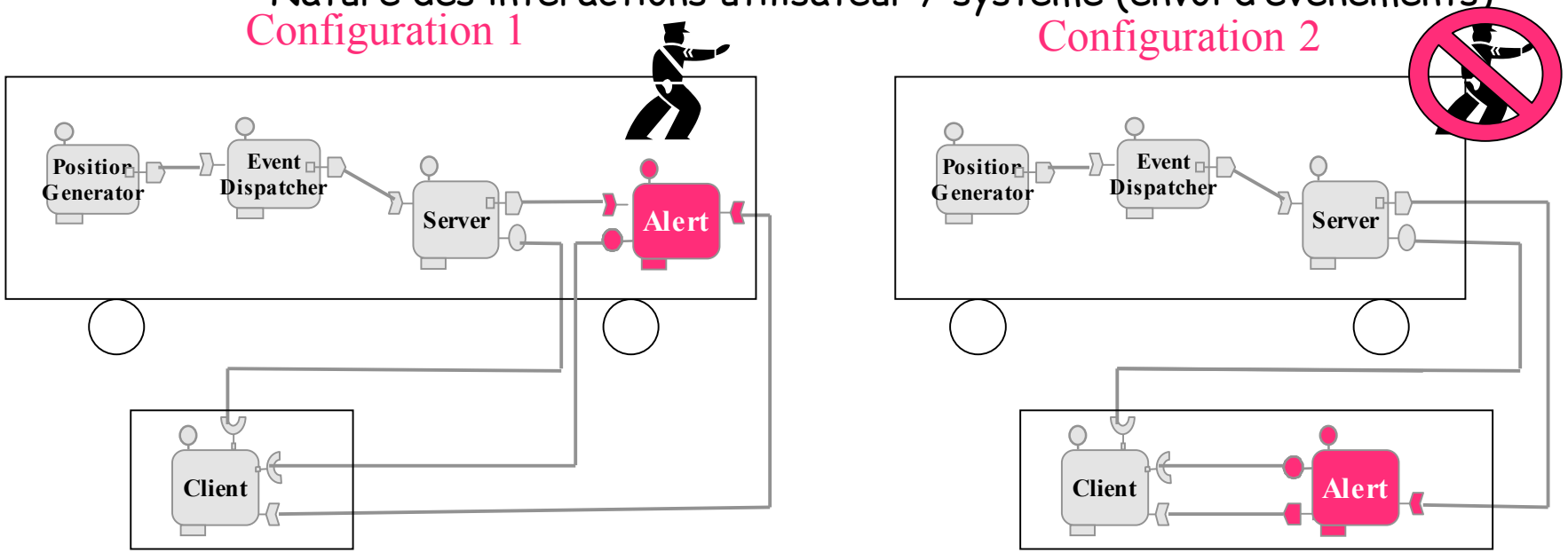
  Comment l'intergiciel de communication est-il déployé/installé sur le terminal utilisateur final ?

  R:

  - Approche « statique » :
    - Imposer son standard (!), partenariats avec constructeurs
  - Approche « dynamique »
    - Téléchargement (e.g. via HTTP) et auto-installation à partir de fichiers CAB, JAR ou EXE sur le terminal,
    - Pré-installation *Plug&Play* sur périphériques de stockage externe (clef USB, carte mémoire SD, etc.)
- Dimensionnement pour petits équipements :
  - Réduction taille archives composants applicatifs (~50%),
  - ~40% archives runtime OpenCCM (classes générées souches et squelettes, conteneur, analyseur XML, etc.).

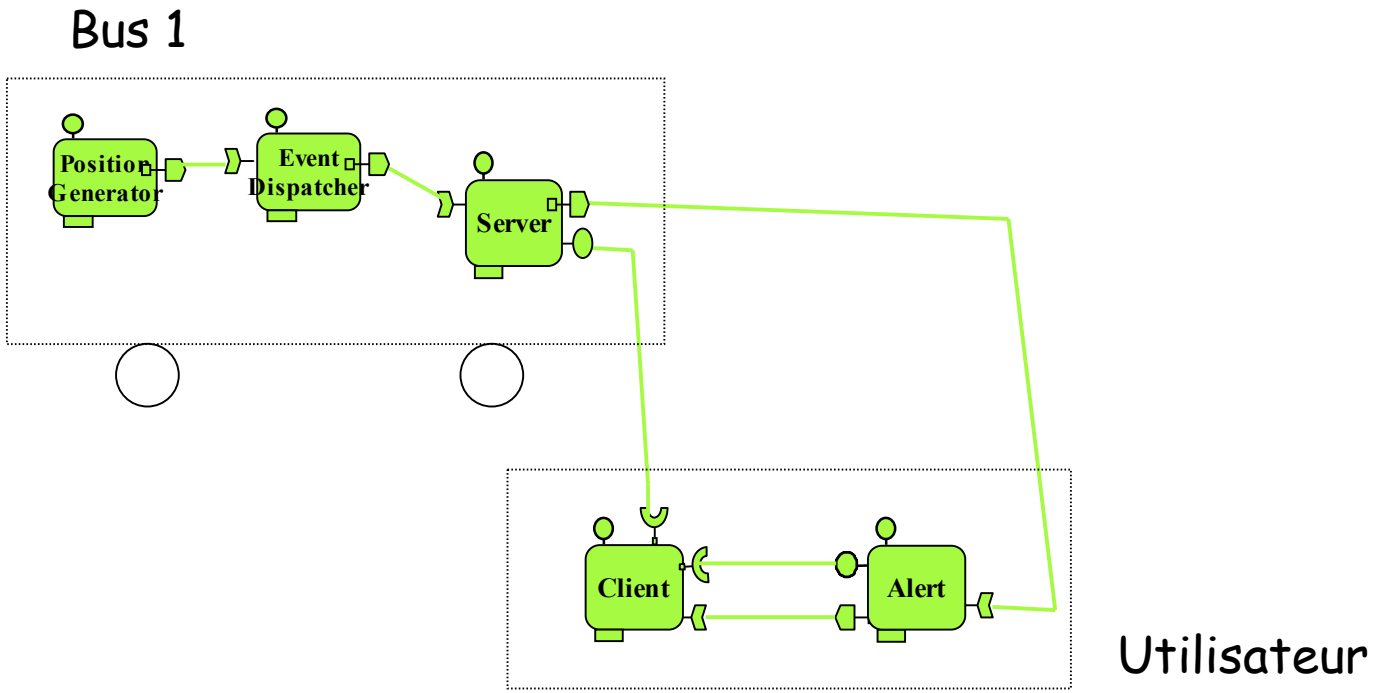
Réponses aux défis (4)

- Ubiquité & respect de la vie privée :
  - Méthodologie de conception des applications / configuration des assemblages via :
    - Composants de traitement déployés et s'exécutant sur les terminaux mobiles (e.g. composant *Alert*).
    - Nature des interactions utilisateur / système (envoi d'événements)



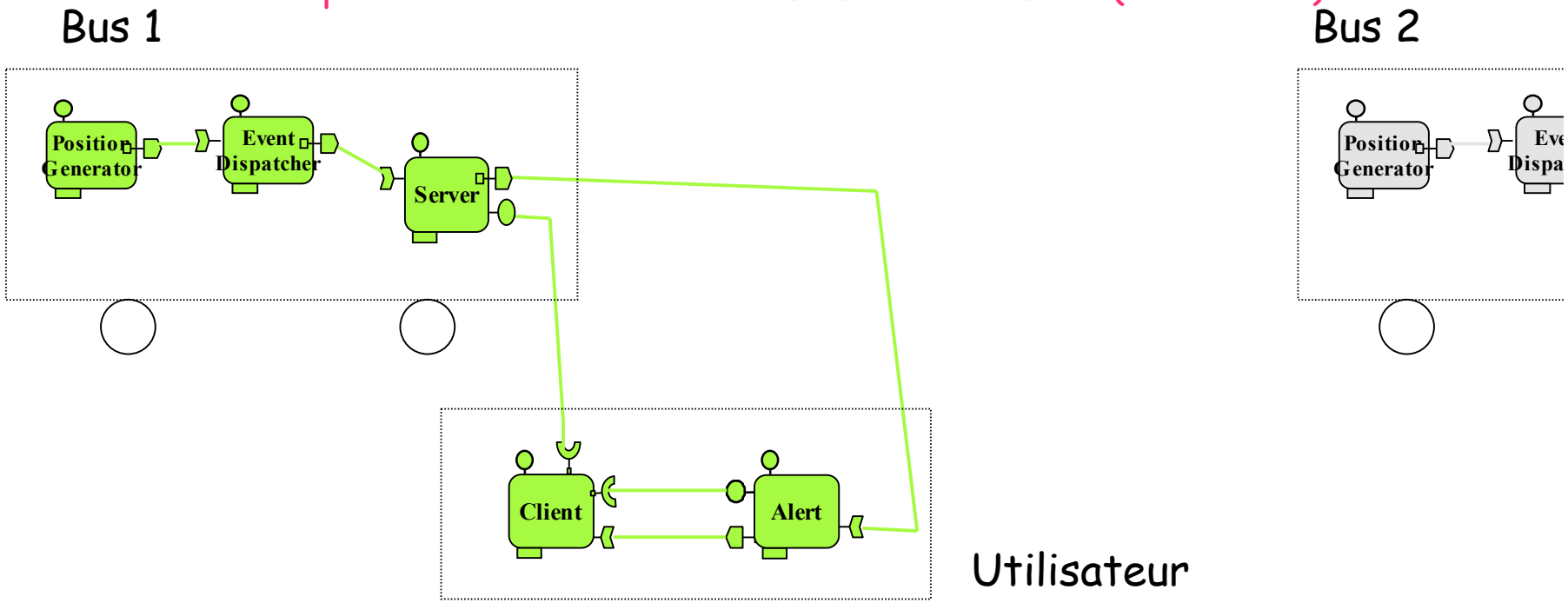
Réponses aux défis (5)

- Problème : le voyageur doit prendre une correspondance
- Solution : re-configuration de services
  - service en cours d'exécution



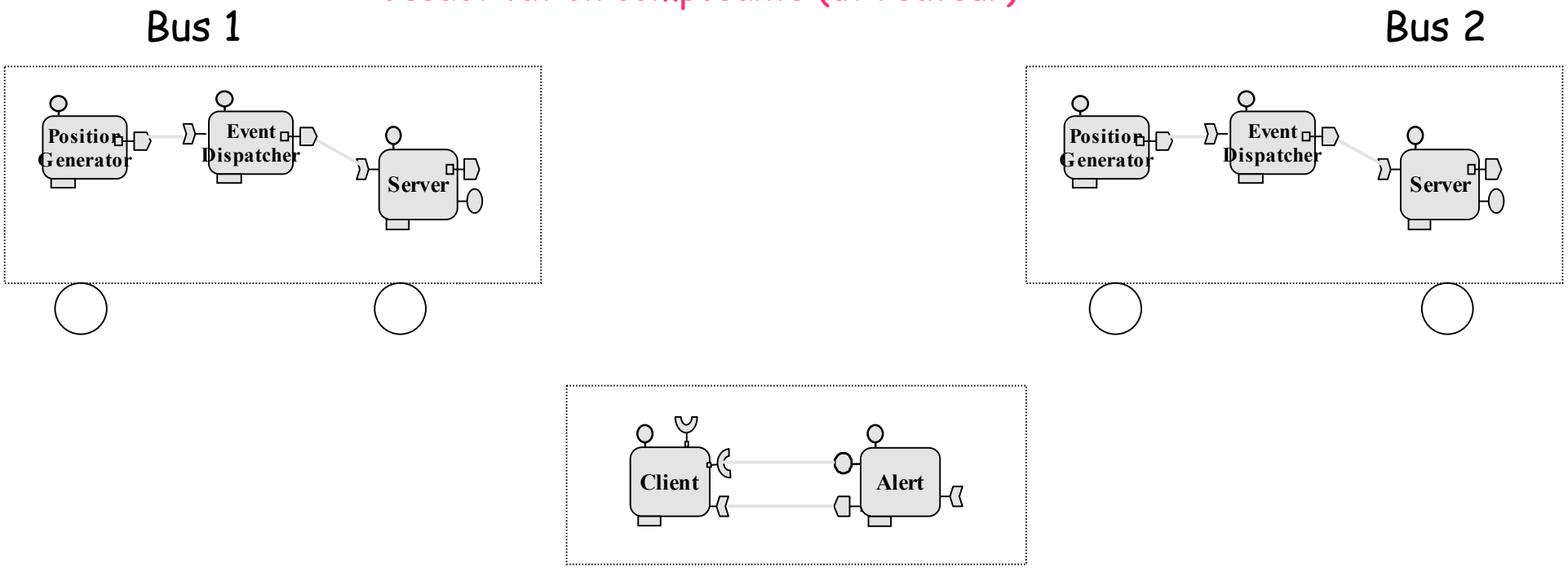
Réponses aux défis (5)

- Problème : le voyageur doit prendre une correspondance
- Solution : re-configuration de services
  - service en cours d'exécution
  - réception événement « VOUS ÊTES ARRIVÉS » (utilisateur)



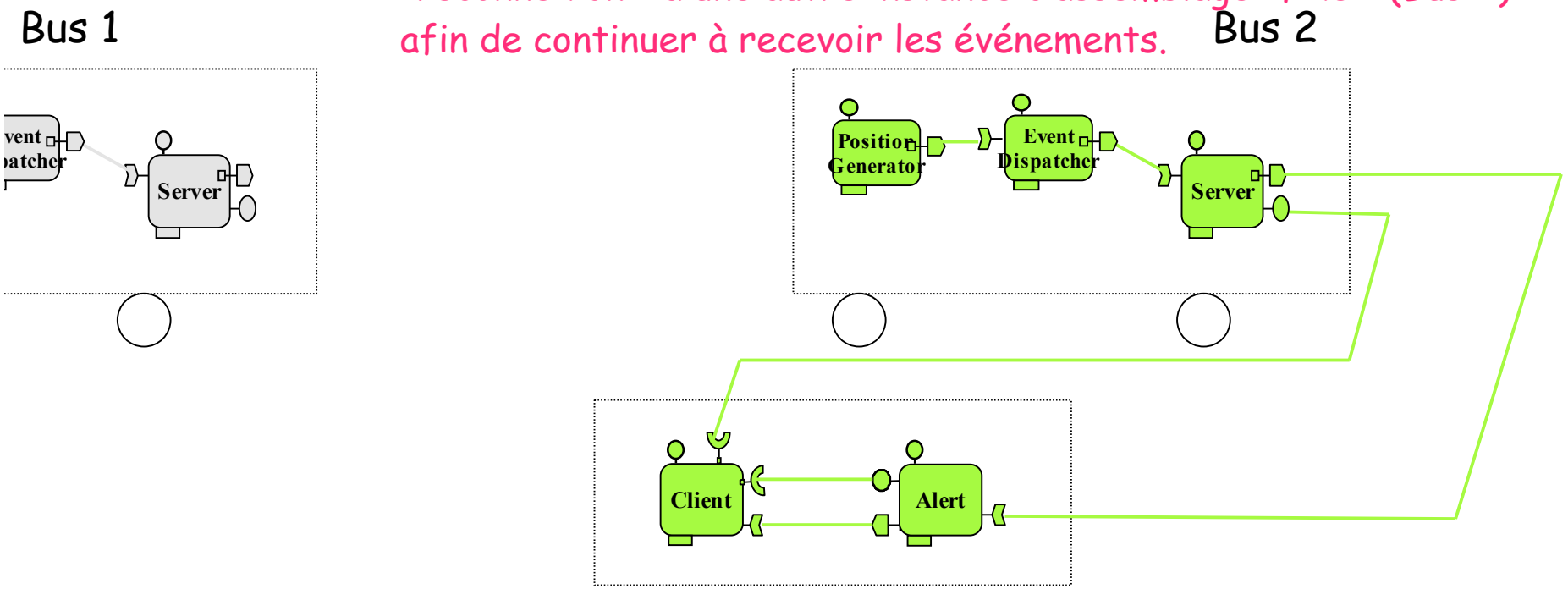
Réponses aux défis (5)

- Problème : le voyageur doit prendre une correspondance
- Solution : re-configuration de services
  - suppression des connexions avec assemblage Bus 1,
  - désactivation composants (utilisateur)



Réponses aux défis (5)

- Problème : le voyageur doit prendre une correspondance
- Solution : re-configuration de services
  - reconfiguration (e.g. propriétés métiers) composants utilisateur
  - "reconnexion" à une autre instance d'assemblage "fixe" (Bus 2) afin de continuer à recevoir les événements.



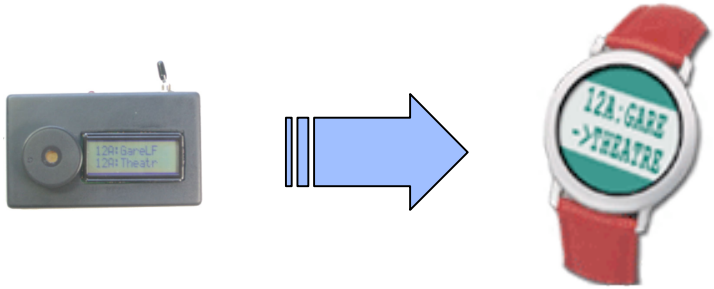
Conclusion (1)

- Approche à base de composants logiciels
  - Méthodologie de conception des applications :
    - Composants/assemblages fixes & mobiles
  - Support d'exécution :
    - Middleware à composants
    - Prototype d'implantation : *OpenCCM Mosaïques*
- Exemple ubiquitaire simple mais nombreux défis à relever !
    - Déploiement incrémental,
    - installation du middleware,
    - dimensionnement / plate-forme cible,
    - reconfiguration dynamique,
    - ubiquité & respect vie privée
    - ...

Conclusion (2)

Une démonstration du projet sur la ligne de bus 12A de Lille a montré la faisabilité du système (version boîtier)

Évolution du système vers un bracelet montre



Possibilité d'adapter le service dans le métro en disposant des balises fixes d'émission dans les stations.

Références

- Dépôt de brevet en septembre 2004
  - réf ALL / HC / 49.946 (INRETS)

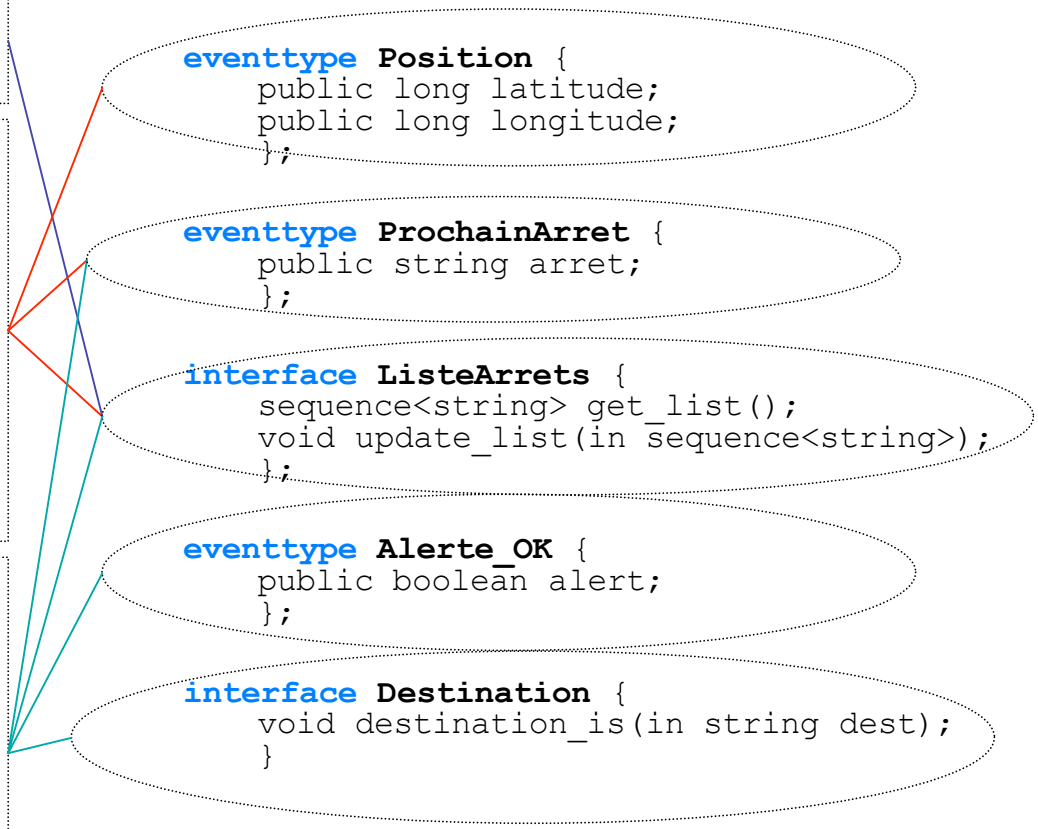
- J. Rioult, C. Gransart, S. Ambellouis, « *Zut, j'ai loupé mon arrêt !* » Un nouveau service d'aide aux déplacements, Cité-TIC, Rennes, 2004.

Annexe : Définition des composants du service et interfaces (IDL)

```
component Manager {
    uses ListeArrets listeArrets;
    };
```

```
component PositionGenerator {
    publishes Position current_position;
    };
component EventDispatcher {
    consumes Position current_position;
    publishes Position current_position;
    };
component Server {
    consumes Position current_position;
    publishes ProchainArret prochainArret;
    provides ListeArrets listeArrets;
    };
```

```
component Alert {
    provides Destination destination;
    consumes ProchainArret prochaineArret;
    publishes Alerte_OK to_users;
    };
component Client {
    uses ListeArrets listeArrets;
    uses Destination destination;
    consumes Alerte_OK fromAlerte;
    }
```



```
eventtype Position {
    public long latitude;
    public long longitude;
    };
```

```
eventtype ProchainArret {
    public string arret;
    };
```

```
interface ListeArrets {
    sequence<string> get_list();
    void update_list(in sequence<string>);
    };
```

```
eventtype Alerte_OK {
    public boolean alert;
    };
```

```
interface Destination {
    void destination_is(in string dest);
    }
```