ASAWoO

Contributions CASA

- **Gestion des communications REST DTN**
  - Proxy HTTP qui permet d'accéder aux fonctionnalités à travers une couche DTN
  - Implems C3PO et Bundle Protocol
- **FunctionalityManager (local)**
  - Functionality Listener
    - Instanciation automatique des fonctionnalités
    - vs spawn
- **Annuaire de services REST**
  - Lien avec les fonctionnalités

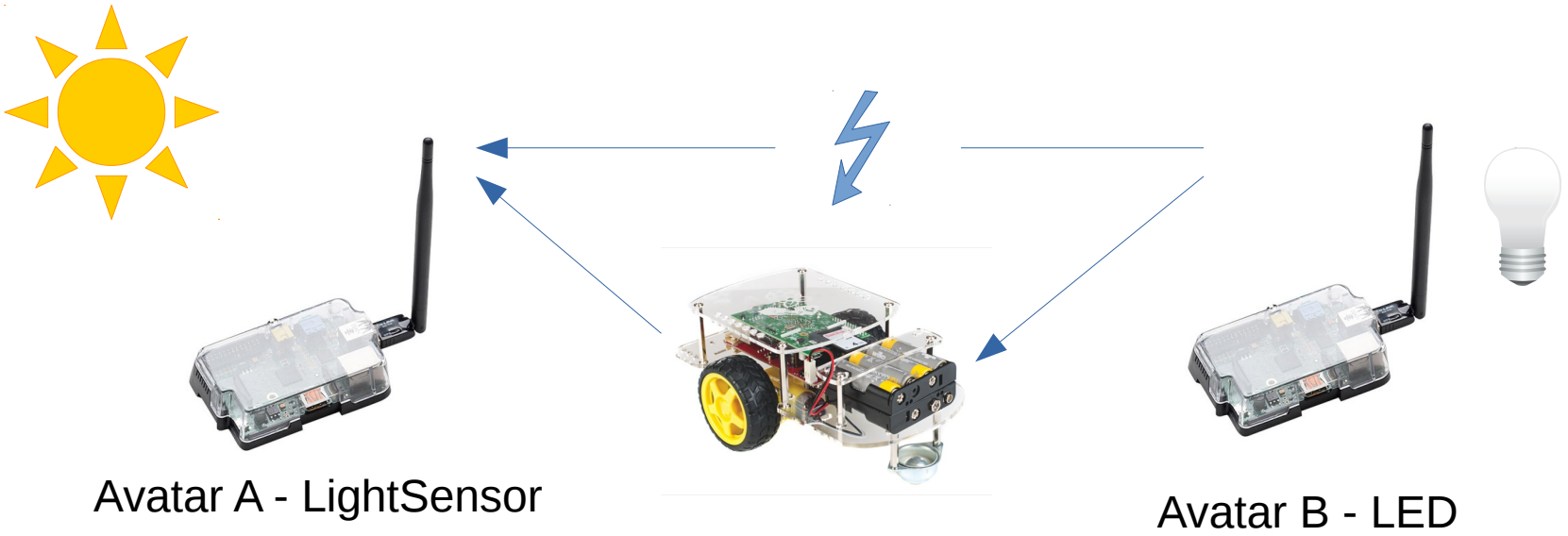
Exemples d'appels via proxy DTN

- `GET http://192.168.10.1:9999/dtn?dtn_uri=http+mcast://asawoo.domains.brightness.LightSensor/light`
- `PUT http://192.168.10.10:9999/dtn?dtn_uri=http://192.168.10.1/raspi/led/status/on`

Scénario de démonstration

- Communication DTN entre avatars
  - Notifier un changement d'état (capteur/actionneur)

Scénario de démonstration



Reste à faire

- **Registre de services « distribué »**
  - Annonces
  - Remote Functionality Manager
- **Génération de proxys pour les fonctionnalités distantes ?**
  - mapping automatique
- **Transformation de requêtes REST pour passer par le proxy DTN (DTN helper)**
- **Support de CoAP**
- **Intégration JS + Python avec Vert.x**

Pile technologique (nouveautés)

- **iPOJO**
  - Modèle à composant
  - Facilite l'instanciation de clients et fournisseurs de services OSGi
- **Vert.X**
  - micro services
  - framework asynchrone
  - Multi-language (Javascript // Python partiellement supporté)
  - serveur et client HTTP (plus léger que D-OSGi)
    - Regarder du côté de Jersey ? (existe connecteur OSGi)

iPOJO - Exemples

- Fournir un service OSGi

```
@Component
@Instantiate
@Provides
@CapabilityProvider(Capabilities = {…})

public class GroveLed implements LedActuator, LedInspector {

...

}
```

iPOJO - Exemples

- ## Se lier à un service

```
public class C3PODTNAdapter implements DTNAdapter, FunctionalityManagerListener {
    @Requires
    private FunctionalityManager functionalityManager;
    @Requires
    private RestServiceRegistry restServiceRegistry;

    @Bind(optional = true)
    void bindMessageService(C3POMessageService messageService) {
        this.msgService = messageService;
        ...
    }
}
```

- ## Cycle de vie

```
public class C3PODTNAdapter implements DTNAdapter, FunctionalityManagerListener {

    @Validate
    public void start() {...}
    @Invalidate
    public void stop() {...}
}
```

Vert.x

- Serveur HTTP (Routes / URI REST)

```
public class RestServiceRegistryImpl implements RestServiceRegistry {
    @Requires
    private Vertx vertx;
    @Requires
    private Router router;

    @Validate
    private void start() {
      router.get("/registry").handler(routingContext -> {
        String json = Json.encodePrettily(getAllFunctionalities());
        routingContext.response()
          .setStatusCode(HttpResponseStatus.OK.code())
          .putHeader(HttpHeaders.CONTENT_TYPE, "application/json")
          .putHeader(HttpHeaders.CONTENT_LENGTH, Integer.toString(json.length()))
                              .write(json)
                              .end();
      });

      router.post("/registry/:functionality/:method").handler(this::addService)

      vertx.createHttpServer().requestHandler(router::accept).listen(httpPort);
    }

}
```

Vert.x

- Client HTTP
  - Exécution de requêtes

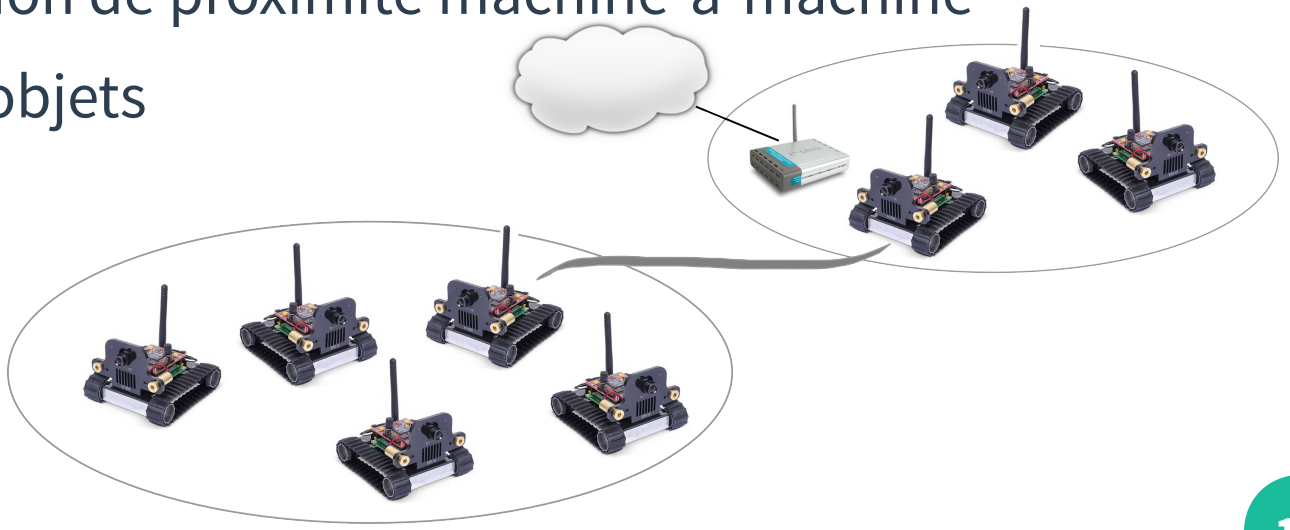
```
HttpClient client = vertx.createHttpClient();

// execute request every 3s
vertx.setTimer(3000, l -> {
            client.get(DTNConstants.DTN_DEFAULT_HTTP_PORT, "localhost",
                        "/registry", response -> {
                response.bodyHandler(buffer -> {
                    logger.info("Response length: "+buffer.length());
                    logger.info("Response body : "+buffer.toString());
                });
            }).end();

        });
```

Ressources

- iPOJO
  - http://felix.apache.org/documentation/subprojects/apache-felix-ipojo.html
- Vert.x 3
  - http://vertx.io/
  - http://vertx.io/docs/vertx-core/java/
  - https://github.com/vert-x3/vertx-examples

Connectivité intermittente

- **Les objets peuvent être accessibles par intermittence pour des raisons**
  - d'économie d'énergie (mise en sommeil)
    - Plages horaires d'activité
  - de faible portée des interfaces de communication sans fils
    - Communication de proximité machine-à-machine
    - Mobilité des objets



Verrous scientifiques

- **Le Web actuel**
  - repose sur des modèles client/serveur et requête/réponse
  - suppose une connectivité de bout-en-bout et une communication synchrone
- **Les protocoles de communications tolérant les ruptures de connectivité**
  - reposent sur le principe du « *store and forward* » ou « *store, carry and forward* »
  - introduisent un asynchronisme et pas de connectivité de bout-en-bout

☛Rendre les protocoles du Web asynchrones et tolérants les ruptures de connectivité

Verrous scientifiques

- **COAP : protocole applicatif conçu pour l'Internet des objets**
  - Couche HTTP-like (RESTful)
  - Couche message légère
  - Transport UDP + fiabilisation (ack, retransmissions)
- **CoAP : bon candidat pour les DTN...**
  - Asynchrone
  - Extension OBSERVE
  - Minimise les échanges de bout-en-bout (crucial pour les DTN)
- **... mais**
  - Suppose UDP/IP
  - Pas de modèle « *store, carry and forward* »

☛ **Implanter COAP sur une couche DTN**