UPMC – Master informatique – M1-PSTL
Contrôle automatique de trafic ferroviaire

BL/PM

2013-14

Description générale du dispositif On équipe un réseau ferroviaire de capteurs, de feux bicolores et, naturellement, de véhicules ferroviaires.

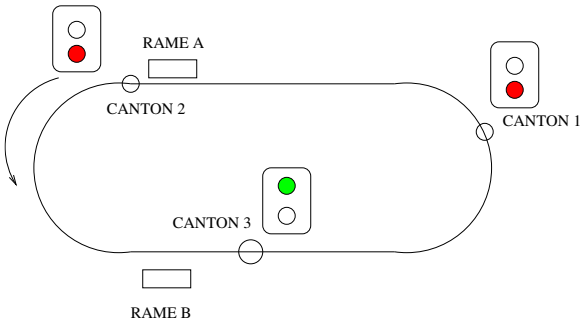
Chaque équipement possède un identificateur. Les véhicules circulent sur le réseau ou sont à l'arrêt. Ils sont capables d'interpréter des ordre de mise en marche ou d'arrêt. Les véhicules en marche peuvent circuler dans un sens ou dans l'autre. Les capteurs émettent une impulsion au passage d'un véhicule. Les feux bicolores sont réalisés par deux *leds* (une rouge, une verte).

Un système central de contrôle reçoit les signaux d'activation des capteurs; il commande les *leds* des feux ainsi que la marche ou l'arrêt des véhicules.

Le système doit assurer la circulation des véhicules sur le réseau en empêchant leurs collisions.

On envisage plusieurs scénarios de circulation des véhicules ferroviaires, en fonction de la configuration du réseau ferré (que nous appelerons sa *topographie*). Ces scénarios définissent la *politique de sécurité* à réaliser.

Scénario 0 Le réseau ferré est un anneau sur lequel les véhicules circulent dans le même sens. On associe un feu à chaque capteur. Dans ce scénario, un capteur représente une limite de canton. Un véhicule ne peut franchir cette limite lorsqu'un autre véhicule est présent sur le canton.



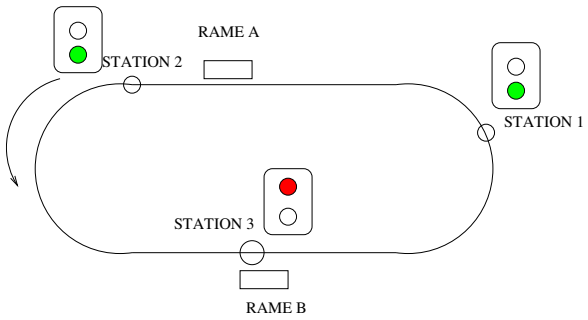
Initialisation: les véhicules sont placés entre deux stations. Il ne doit y avoir qu'un seul véhicule entre deux stations. Tous les feux sont positionnés au vert sauf ceux qui se trouvent derrières les véhicules qui sont positionnés au rouge. On démarre l'ensemble des véhicules.

Politique de sécurité: un véhicule ne franchir une limite de canton lorsque le canton est occupé.

1. Lorsqu'un véhicule active un capteur dont le feu est au vert, le feu est mis au rouge.
2. Lorsqu'un véhicule active un capteur dont le feu est au rouge, le véhicule est arrêté.

3. Un véhicule ne peut redémarrer que si le feu du capteur suivant celui sur lequel il est arrêté est mis au vert.
4. Lorsqu'un véhicule redémarre, le feu du capteur précédent celui qu'il quitte peut être mis au vert.

Scénario 1 Le réseau ferré est un anneau sur lequel les véhicules circulent dans le même sens. On associe un feu à chaque capteur. Dans ce scénario, un capteur représente une station. Un véhicule doit être arrêté lorsqu'il atteint un capteur. Le véhicule stationne un temps arbitrairement déterminé.

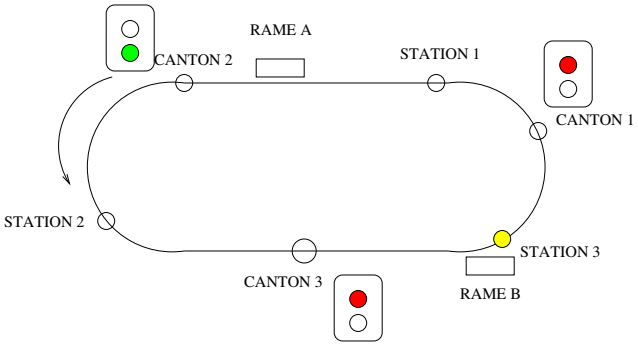


Initialisation: les véhicules sont placés entre deux stations. Il ne doit y avoir qu'un seul véhicule entre deux stations. Tous les feux sont positionnés au vert sauf ceux qui se trouvent derrières les véhicules qui sont positionnés au rouge. On démarre l'ensemble des véhicules.

Politique de sécurité: un véhicule ne peut quitter une station lorsque la station suivante est occupée.

1. Lorsqu'un véhicule active un capteur, il est arrêté et le feux correspondant est mis au rouge.
2. Un véhicule ne peut redémarrer que si le feu du capteur suivant celui sur lequel il stationne est au vert.
3. Lorsqu'un véhicule redémarre, le feu du capteur précédent celui qu'il quitte peut être mis au vert.

Scénario 2 Le réseau ferré est un anneau sur lequel les véhicules circulent dans le même sens. Dans ce scénario, les capteurs sont utilisés pour diviser le réseau en *cantons*. Ces capteurs sont dits *en limite de canton*. Dans chaque canton, on place un (et un seul) capteur qui représente une station. À chaque capteur en limite de canton est associé un feu. Un véhicule doit être arrêté lorsqu'il active le capteur d'une station. Il stationne un temps arbitrairement déterminé.

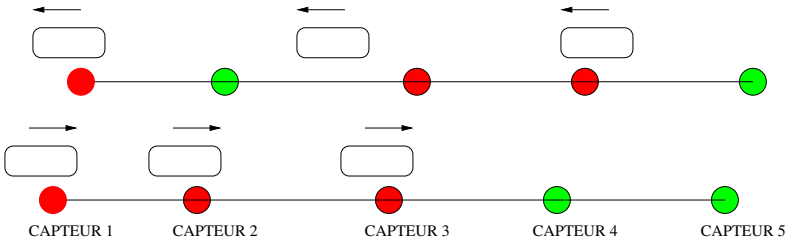


Initialisation: les véhicules sont placés entre une station et une limite de canton. Il ne doit y avoir qu'un seul véhicule sur chaque canton. Les feux correspondant au limites de cantons qui se trouvent derrière un véhicule sont mis au rouge. Les autres feux sont mis au vert. On démarre l'ensemble des véhicules.

Politique de sécurité: il ne doit y avoir qu'un seul véhicule au plus sur chaque canton.

1. Lorsqu'un véhicule active un capteur en limite de canton:
   (a) Si le feu associé est au rouge, le véhicule doit être arrêté.
   (b) Si le feu est au vert, il doit être mis au rouge.
2. Lorsqu'un véhicule est arrêté en limite de canton et que le feu associé passe au vert, le véhicule doit redémarrer et le feu doit repasser au rouge.
3. Lorsqu'un feu est mis au rouge, le feu associé au capteur en limite du canton précédent peut être mis au vert.

Scémario 3 Le réseau est un segment qui représente une ligne à vois unique. Les véhicules y circulent dans un sens ou dans l'autre, amis, à un moment donné, tous les véhicules circulent dans le même sens. Ils font la navette d'un côté à l'autre de la ligne. Les capteurs représentent des stations. On associe un feu à chaque capteur. Les véhicules doivent être arrêtés lorsqu'ils passent sur un capteur et le feu correspondant est mis au rouge. Ils stationnent un temps arbitraire. Lorsque tous les véhicules sont stationnés d'un côté de la ligne, le sens de la marche est inversé et le traffic reprend.



Initialisation: les véhicules sont placés entre deux stations. Il ne doit y avoir qu'un seul véhicule entre deux stations. Tous les feux sont positionnés au vert. On démarre l'ensemble des véhicules, dans le même sens.

Politique de sécurité: un véhicule ne peut quitter une station lorsque la station qui se trouve devant lui, dans le sens de la marche, est occupée. Un véhicule qui atteint un capteur en fin de ligne ne peut redémarrer que lorsque le sens de la marche a été inversé.

1. Lorsqu'un véhicule active un capteur, il est arrêté et le feux correspondant est mis au rouge.
2. Un véhicule ne peut redémarrer que si le feu du capteur qui se trouve devant lui, dans le sens de la marche, est au vert.
3. Un véhicule qui atteint un capteur à l'extrémité de la ligne ne peut redémarrer qu'en inversant le sens de sa marche.
4. Si la ligne possède n capteurs, numérotés de 1 à n de manière consécutive, et k véhicule, le sens de la marche ne peut être inversé que si ou bien les k stations de 1 à k sont occupées ou bien les k stations de n à $n-k-1$ sont occupées.

Description du système Le système final visé comporte les matériels suivants:

- un circuit ferroviaire électrique alimenté par rails;
- des ampoules *reed* pour les capteurs de passage de véhicules;
- des couples de *leds* rouge et verte pour les feux de signalisation;
- des locomotives digitales (3) pour les véhicules;

- un module dcc *sprog* pour relayer les commandes de marche et d'arrêt vers les véhicules;
- un micro-contrôleur Arduino pour relayer les commandes d'allumage et d'extinction des *leds* ainsi que les signaux d'activation des ampoules `reed`.

Le système comporte les deux composants logiciels: nous appelons le premier *moniteur* et le second *contrôleur*. Le premier est l'interface de pilotage des équipements. Il est relié au module *sprog* et au micro-contrôleur par des liaisons série (USB). Le second est le système chargé d'assurer la politique sécurité du traffic en décidant des ordres de marche et d'arrêt des véhicules et de la couleur des feux. Ces deux composants communiquent par une liaison TCP/IP selon le protocole PCF spécifié page . Chaque capteur, feu (couple de *leds*) et véhicule possède un identificateur unique pour chaque catégorie d'équipement.

Le composant *moniteur* est chargé de

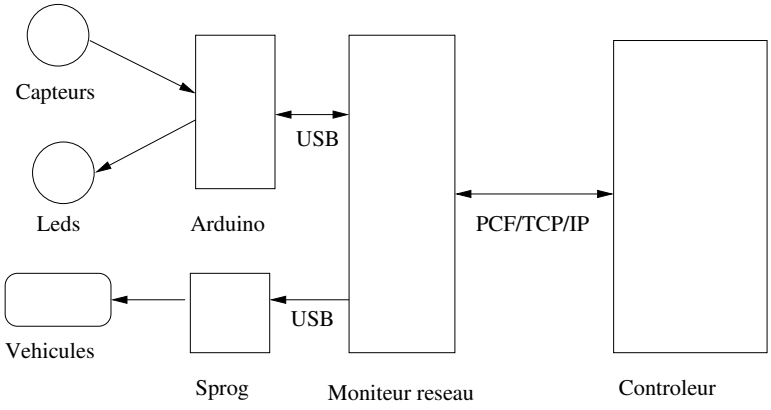
- relayer vers le *contrôleur* les signaux d'activation des capteurs qu'il reçoit du micro-contrôleur. Il transmet l'identificateur du capteur activé;
- relayer vers le micro-contrôleur les commandes de changement de la couleur des feux de signalisation et vers le module *sprog* les commandes de mise en marche ou d'arrêt des véhicules. Il reçoit du *contrôleur* les identificateurs des équipements concernés ainsi que la nature des actions à effectuer.

  Le composant *moniteur* peut également *complètement simuler* l'activité du réseau physique en engendrant de lui-même les messages à transmettre au contrôleur. Une telle version du composant sera fournie pour le développement du contrôleur, avant que celui-ci ne soit éprouvé sur un réseau physique.

Le module *contrôleur* doit assurer la politique de sécurité du trafic du réseau ferrovaire, étant donné la *topographie* du réseau, le *scénario* de circulation des véhicules et la *configuration initiale* du traffic. Ces trois éléments font l'objet d'une négociation avec le composant *réseau* qui constitue la première phase de l'échange entre ces deux composants. Il a l'initiative de la mise en route du sysème, une fois la configuration initiale convenue et mise en place.

Il ne dispose pour assurer la politique de sécurité que des données d'activation des capteurs transmises par le *moniteur* et de sa capacité de requérir auprès du *moniteur* un changement de la couleur des feux ainsi que la marche et l'arrêt des véhicules.

Le développement du contrôleur est la mission confiée.



PCF: protocole de contrôle ferroviaire L'échange des messages est asynchrone. Il est à l'initiative du contrôleur ou du réseau. Les messages sont au format XML tel que définit par la DTD donnée page 8. L'élément racine des messages échangés est `pcf`: on parlera de *message* ***`pcf`***. L'élément racine `pcf` a deux attributs:

- `reqid` un identificateur (voir ci-dessous);
- `type` qui a pour valeur `request` ou `answer` selon qu'il s'agit d'une requête ou d'une réponse. L'attribut peut également prendre la valeur `advise` lorsqu'il s'agit d'un simple message d'information.
  L'émetteur des requêtes engendre les identificateurs et gère pour son compte leur unicité. Une réponse doit contenir l'identificateur de la requête à laquelle elle répond. Une réponse contenant un identificateur inconnu **doit** être ignorée. L'identificateur d'un message de type `answer` **peut** reprendre celui de la requête ou de la réponse qui a provoqué son émission; il **peut** également utiliser un identificateur unique.

Lorsque le type d'un message `pcf` est `request` nous parlerons d'une *requête* ***`pcf`*** ou, plus simplement de *requête*; lorsque le type est `answer` nous parlerons d'une *réponse* ***`pcf`*** ou, plus simplement de *réponse*. Les messages `pcf` ne contiennent qu'un seul élément. Selon l'élément contenu dans le message `pcf` et la nature du message (requête ou réponse), nous parlons de *requête* ***`eltname`*** ou de *réponse* ***`eltname`***; où `eltname` est le nom de cet élément.

Éléments contenus dans un message `pcf`

- `info` cet élément est principalement destiné aux messages d'informations. En réponse, il peut être utilisé pour accepter ou refuser une requête. En requête, elle est utilisée pour signaler un incident de fonctionnement, ce qui est redondant avec l'attribut `info`. L'élément `info` **doit** contenir un attribut `status` qui prend la valeur `ok` ou `ko`. Il **peut** transporter un message.
- `hello` requête d'initialisation d'une session.
  L'élément `hello` **doit** indiquer un attribut `id` qui permet d'identifier l'émetteur de la requête;
- `olleh` réponse d'acquittement de la requête `hello`.
  L'élément `olleh` **peut** indiquer un attribut `id` qui permet d'identifier l'émetteur de la réponse;
- `bye` requête ou réponse d'information de fin de session. L'émetteur d'un message `bye` **doit** fermer sa connection après l'émission du message;
- `topography` requête destinée à déterminer la topographie du réseau utilisé. La topographie est décrite sous la forme du graphe orienté des capteurs présents sur le réseau. Le graphe est donné comme une liste de noeuds (voir élément `edges`).
  Lorsque cette requête est émise par le réseau, elle **doit** contenir la description de sa topographie. Le contrôleur **doit** alors émettre une réponse `advise` avec un statut `ok` ou `ko` selon qu'il accepte ou non la topographie proposée. Si la réponse `advise` a le statut `ok`, la topographie du réseau est dite *déterminée*.
  Lorsque cette requête est émise par le contrôleur, elle **doit** contenir un graphe vide (liste vide). Le réseau **doit** alors émettre une requête `topography` qui **doit** contenir la description de son graphe de capteurs.
- `lights` requête ou réponse permettant de définir où sont placés les feux. Le placement des feux est indiqué par une liste de feux: voir élément `light`.
- `scenario` requête permettant de définir quel scénario de circulation va être utilisé. Le nom (ou numéro) du scénario **peut** être indiqué avec l'attribut `id`.
  Si, dans une requête, l'identificateur est indiqué, le récepteur de la requête doit comprendre que l'émetteur lui indique quel scénario il veut utiliser. Auquel cas, le récepteur **doit** émettre une réponse `advise` avec un statut `ok` ou `ko` selon qu'il accepte ou non le scénario proposé. Si le statut de la réponse

est `ok`, le scénario est dit *défini*.
Si, dans une requête, l'identificateur n'est pas indiqué, le récepteur doit comprendre que l'émetteur demande quel scénario il faut utiliser. Auquel cas, le récepteur **doit** émettre soit une réponse `scenario` contenant le scénario qu'il veut utiliser soit une réponse de fin de connection.

- `init` requête destinée à déterminer la configuration initiale du système. La configuration initiale du système est donnée par l'indication de la position des véhicules entre deux capteurs (voir élément `position`).
  Lorsque cette requête est émise avec un ensemble de positions non vide, le récepteur de la requête **doit** émettre une reponse `advise` avec le statut `ok`, s'il accepte la configuration proposée, ou avec le statut `ko` s'il la refuse.
  Lorsque cette requête est émise avec un ensemble de positions vide, le récepteur de la requête doit comprendre que l'émetteur lui demande de proposer une configuration initiale. Le récepteur **peut** alors émettre une requête `init` contenant une configuration initiale non vide.
  Lorsqu'une requête proposant une configuration non vide a reçu une réponse positive (`advise/ok`), que la topographie est déterminée et le scénario défini, le système est *initialisé*.

  > **Il faut faire l'hypothèse** que le réseau ne propose ou n'accepte que des configurations initiales qu'il peut réaliser.

- `start` requête destinée à démarrer le fonctionnement du système. Cette requête ne peut être émise que par le contrôleur. Elle **doit** être comprise par le réseau comme la demande de mettre en marche les véhicules présents sur le réseau. Cette requête **ne doit pas** être acceptée et réalisée tant que le système n'a pas été initialisée. Le réseau **doit** émettre une réponse `advise` avec le statut `ok` ou `ko` selon qu'il accepte et réalise ou non la requête.

- `up` requête signalant l'activation de capteurs. Elle ne doit être émise que par le réseau. Elle contient l'identification des capteurs activés (voir élément `capteur`). Cette requête **peut** être acquittée par le contrôleur.

- `set` requête de commande des feux de signalisation ou des véhicules ferroviaires. Elle indique l'identificateur des équipements visés ainsi que les informations utiles à l'exécution de la commande. Cette requête est émise par le contrôleur pour piloter les feux, la marche ou l'arrêt des véhicules. Elle est émise par le réseau lorsqu'un véhicule demande à redémarrer. Pour les détails sur ces modalités, voir les éléments `light` et `train`.

Éléments décrivant un équipement

- `capteur` admet deux attributs:
  - `id` qui donne l'identificateur du capteur;
  - `type` qui peut prendre la valeur `canton` ou `station`.
- `light` admet deux attributs:
  - `id` qui donne l'identificateur du feu;
  - `color` qui peut prendre la valeur `red` ou `green`. Il indique quelle *led* doit être allumée, l'autre devant être alors éteinte.
- `train` admet trois attributs:
  - `id` qui donne l'identificateur du véhicule;

- `action` qui peut prendre la valeur `stop` ou `start`.
  Les valeurs `start` et `stop` sont utilisées par le contrôleur pour mettre en marche (`start`) ou arrêter un véhicule (`stop`).
- `dir` qui peux prendre les valeurs `forward` ou `backward` selon que le véhicule doit circuler dans un sens ou dans l'autre. Cet attribut optionnel a pour valeur par défaut `forward`.

Élements pour la topographie d'un réseau

- `edges` élément contenant la description d'un nœud du graphe des capteurs du réseau. Il contient un élément `capteur`, la liste de ses prédécesseurs (élément `in`) qui correspondent aux arrêtes entrantes et la liste de ses prédécesseurs (élément `out`) qui correspondent à ses arrêtes sortantes;
- `in` contient une liste de capteurs;
- `out` contient une liste de capteurs.

Les listes de capteurs oeuvent être vides l'une ou l'autre, mais pas l'une et l'autre.

Élément pour l'initialisation

- `position` contient trois éléments:
  - `before` le capteur avant le véhicule;
  - `train` le véhicule;
  - `after` le capteur après le véhicule.

  L'élément `position` indique qu'il faut placer le véhicule identifié par l'élément `train` entre les deux capteurs mentionnés.
  Le placement décrit n'est réalisable que si le capteur `after` est successeur du capteur `before` dans le graphe topographique des capteurs du réseau. Lorsqu'un placement n'est pas réalisable, le récepteur d'une requête contenant un tel élément **doit** refuser la requête.

DTD

```
<!DOCTYPE pcf [

<!ELEMENT pcf
         (hello|olleh|scenario|topography|init|start|up|set|info|bye)>
<!ATTLIST pcf reqid ID #REQUIRED>
<!ATTLIST pcf type (request|answer|advise) #REQUIRED>


<!ELEMENT hello EMPTY>
<!ATTLIST id CDATA #REQUIRED>

<!ELEMENT olleh EMPTY>
<!ATTLIST id CDATA #IMPLIED>

<!ELEMENT scenario EMPTY>
<!ATTLIST scenario id CDATA #IMPLIED>

<!ELEMENT topography (edges*)>

<!ELEMENT edges (capteur,in,out)>
<!ELEMENT in (capteur*)>
<!ELEMENT out (capteur*)>

<!ELEMENT lights (light*)>

<!ELEMENT init (position*)>
<!ELEMENT position (before,train,after)>
<!ELEMENT before (capteur)>
<!ELEMENT after (capteur)>

<!ELEMENT up (capteur+)>

<!ELEMENT set (train|light)+>

<!ELEMENT capteur EMPTY>
<!ATTLIST capteur id CDATA #REQUIRED>
<!ATTLIST capteur type (canton|station) #IMPLIED>

<!ELEMENT light EMPTY>
<!ATTLIST light id CDATA #REQUIRED>
<!ATTLIST light color (red|green) #IMPLIED>

<!ELEMENT train EMPTY>
<!ATTLIST train id CDATA #REQUIRED>
<!ATTLIST train action (start|stop) #IMPLIED>
<!ATTLIST train dir (forward|backward) #IMPLIED>

<!ELEMENT start EMPTY>

<!ELEMENT info CDATA?>
```

```
<!ATTLIST info status (ok|ko) #REQUIRED)

<!ELEMENT bye EMPTY>

]>
```

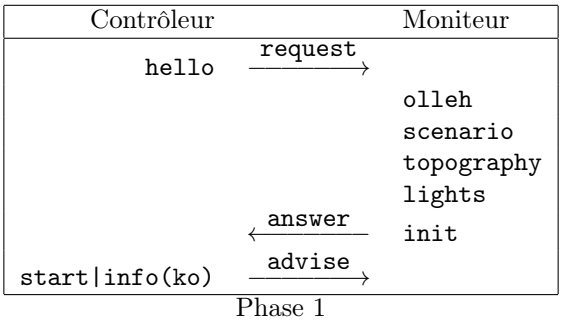
Un modèle client/serveur On propose une architecture du système sur le modèle client/serveur:

- le *moniteur* est le serveur;
- le *contrôleur* est le client.

Les composants échangent des messages PCF. Il y a deux phases aux échanges:

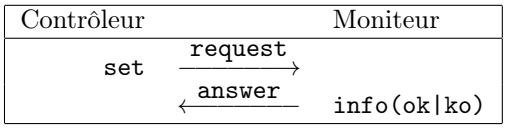
1. une phase d'initialisation qui a pour but de déterminer la configuration du réseau physique, le scénario de trafic à exécuter et la position initiale des véhicules. Elle se termine par la mise en route du trafic, si tout s'est bien passé.
2. une phase d'exploitation qui consiste en une suite d'échange assurant l'exécution du scénario.

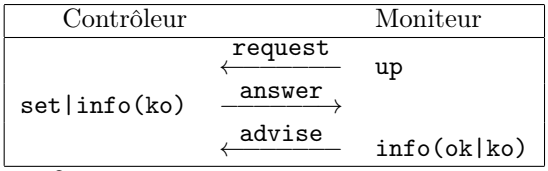
La phase 1 est l'échange séquentiel suivant:



Phase 1

La phase 2 est la répétition des échanges suivants: le premier correspond au redémarrage d'un véhicule, le second à l'arrêt.





Phase 2