

L041

Rapport de projet



utbm
université de technologie
Belfort-Montbéliard

Table des matières

Rappel du sujet :	3
Approche du projet :	4
Identification d'un transfert et interception d'un portique	4
Déplacement d'un portique	5
Godlike.....	6
Espace de stockage.....	6
Affichage.....	7
Jeu de test et lecture de fichier externe	9
Pour le futur du programme	10

Rappel du sujet :

Sur une plateforme logistique, 3 types de transporteurs se présentent :

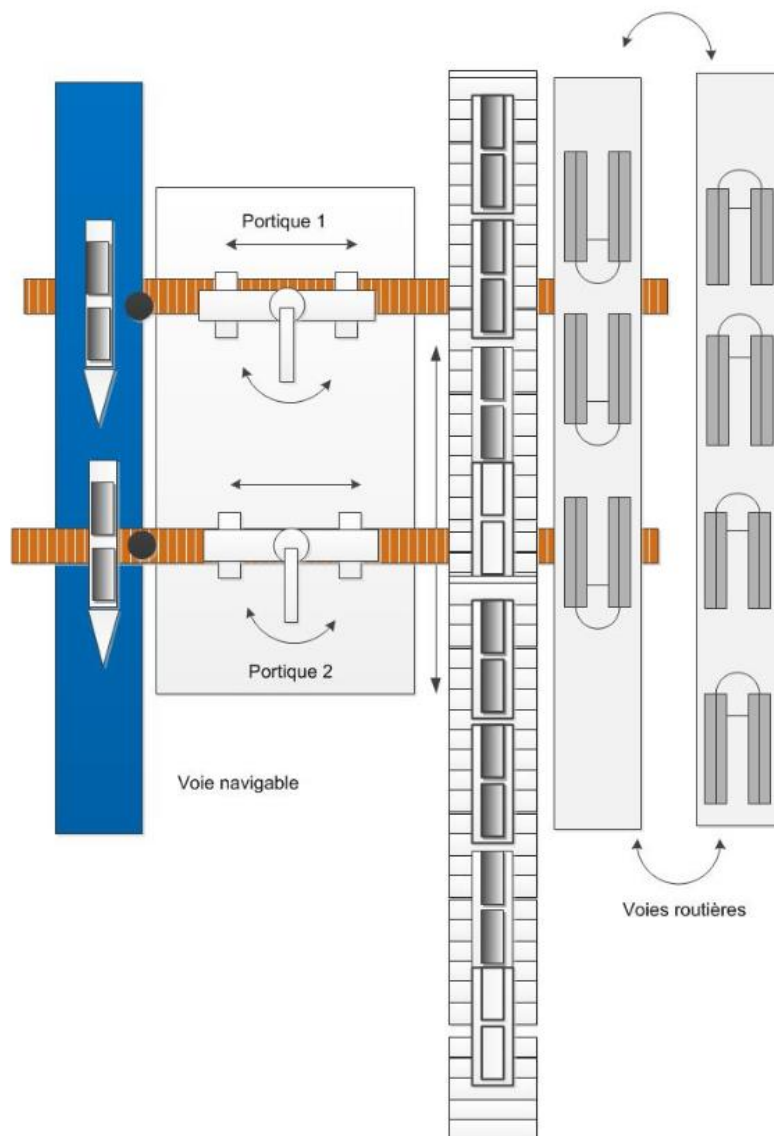
- Des péniches
- Des trains
- Des camions

Chaque transporteur possède une destination ainsi qu'un nombre n d'emplacements qui peuvent accueillir des conteneurs.

Chaque conteneur possède une destination et est positionné sur un transporteur.

Le but du projet est de synchroniser deux portiques qui devront déplacer les conteneurs d'un transporteur à un autre de façon à ce que la destination du conteneur corresponde à celle du transporteur sur lequel il est placé.

Les deux portiques doivent pouvoir accéder à tous les transporteurs sans avoir à se croiser.



Approche du projet :

Identification d'un transfert et interception d'un portique

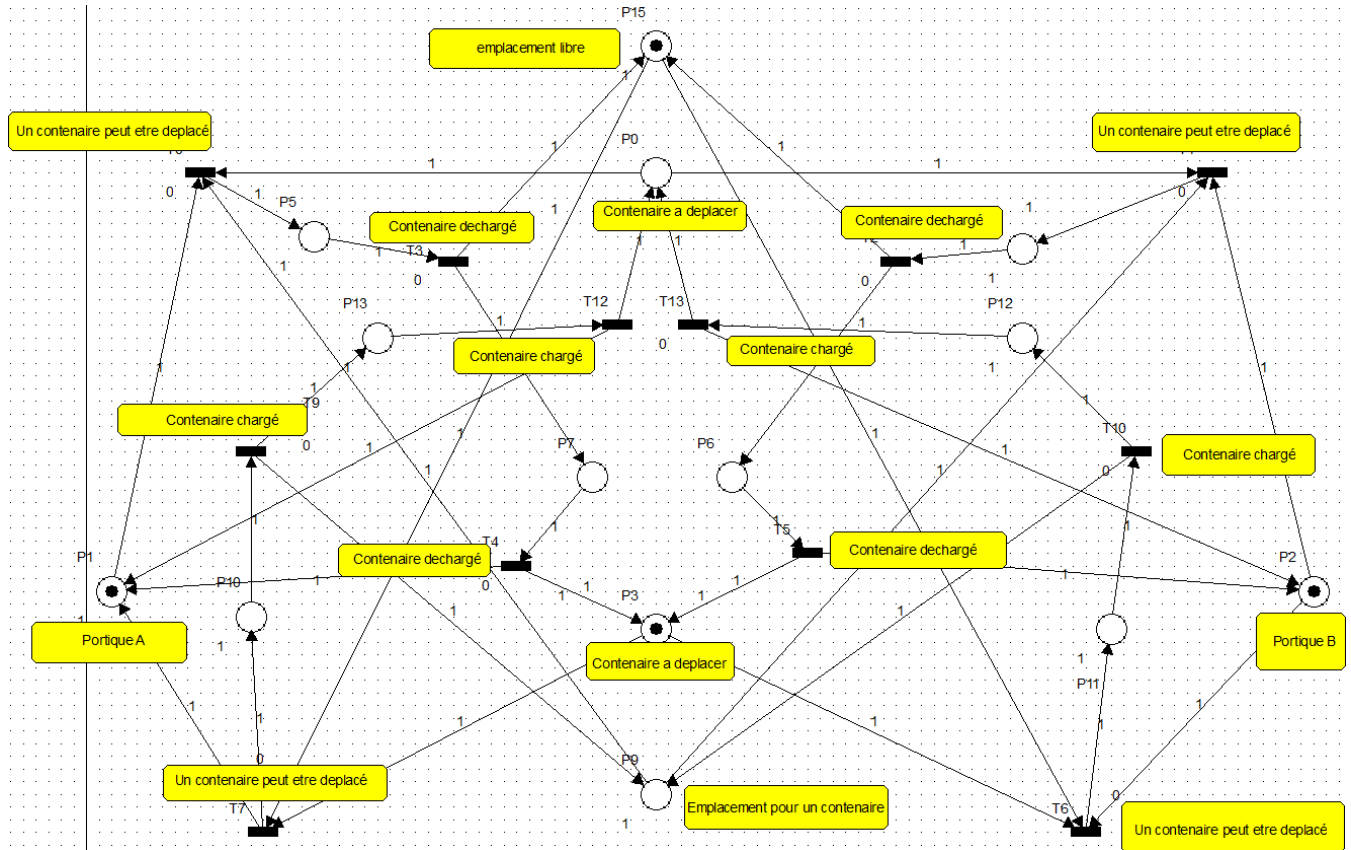


Figure 0-1 Représentation par réseau de pétri de la discussion entre deux transporteurs pour le transfert de conteneurs

Pour réaliser ce projet, j'ai choisi de donner le plein pouvoir à mes transporteurs, ce sont mes transporteurs qui vont choisir des déplacements de leurs conteneurs. Chaque transporteur aura donc deux processus (un pour chaque portique) et va s'occuper d'appeler les portiques. Lorsque l'un des processus aura réussi à intercepter un portique libre, le portique sera mis en attente d'instruction. Le processus du transporteur va alors vérifier si un déplacement d'un de ses conteneurs est disponible, si c'est le cas une structure est remplie avec les informations nécessaires (identifiant du transporteur appelant, identifiant du conteneur à déplacer, identifiant du transporteur receveur et identifiant de l'emplacement à desservir). Une fois les informations remplies, le portique en est notifié, il récupère les informations et lance la phase de transfert du conteneur.

PS Un système de sémaphore a été mis en place sur l'emplacement du conteneur à déplacer et sur l'emplacement visé. L'emplacement du conteneur à déplacer est libéré après le chargement et l'emplacement visé est libéré après le déchargement.

Déplacement d'un portique

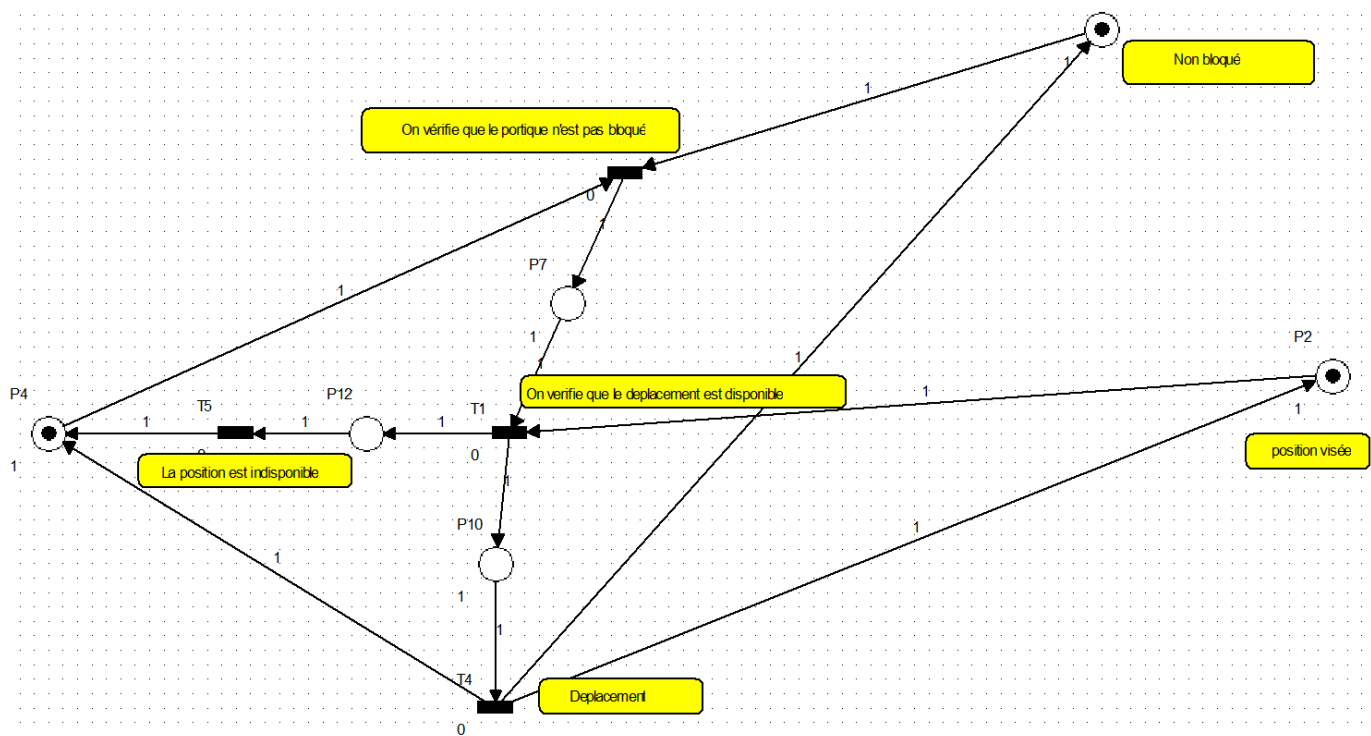


Figure 0-2 Représentation par réseau de pétri du déplacement d'un portique vers une unité

Le portique a donc pour rôle le transfert d'un conteneur d'un transporteur à un autre. La contrainte étant que l'on a deux portiques et qu'ils ne doivent pas se croiser.

Pour répondre à cette contrainte, j'ai choisi de représenter ma zone d'activité par une grille, les voies de chaque transporteur étant définies, la taille sera fixe en largeur, mais variable en hauteur de façon à pouvoir modifier le nombre de transporteurs. Dans cette grille j'ai choisi d'y représenter la position des emplacements sur lesquels pourront être posés des conteneurs sur un repère orthonormé (x,y). Les portiques prenant toute la largeur de la grille n'auront besoin de gérer seulement leur déplacement en y. Le repère y ne servira que plus tard pour l'affichage.

Le déplacement du portique va se dérouler en deux phases : une phase de déchargement (le portique va alors récupérer le conteneur à déplacer) et une phase de chargement où le portique va placer le conteneur sur l'emplacement indiqué du transporteur correspondant).

Pour chacune des phases, le portique va déterminer s'il doit augmenter ou réduire sa position en y pour atteindre son checkpoint (son point de chargement ou de déchargement). Il va alors progresser d'unité en unité en vérifiant à chaque fois si le second portique n'est pas déjà présent sur la position souhaitée. Si le second portique est sur la position souhaitée, le portique va alors bloquer le libre déplacement du second portique et le pousser au fur et à mesure de son déplacement. Une fois le checkpoint atteint et l'action de chargement ou de déchargement effectuée, le second portique retrouve alors son libre déplacement.

PS Pour éviter toute situation d'interblocage, un système de sémaphore a été mis en place sur la variable indiquant si le portique est bloqué ou non.

Godlike

La fonction Godlike est une fonction qui réagit dans le cas où plus aucun transporteur ne peut desservir de conteneur. La fonction va permettre de débloquer la situation. Pour déterminer si on doit activer la fonction on procède de la manière suivante :

Lorsqu'un portique est appelé, mais que le processus ne trouve pas de conteneur à déplacer, le processus incrémente une variable, le signale au processus de la fonction Godlike puis se met en attente. À chaque signal, la fonction Godlike vérifie le nombre de processus mis en attente. Si tous les processus sont mis en attente alors la fonction se charge dans un premier temps de vérifier si un camion en file d'attente peut libérer la situation.

On va d'abord chercher à voir si un camion peut être rempli de façon à privilégier ainsi le départ d'un camion, puis par la suite voir si un camion peut compléter l'emplacement d'un autre transporteur avec le conteneur qu'il transporte. Si c'est le cas, le camion en question remplace alors automatiquement un camion présent sur la plateforme, si ce n'est pas le cas on entame un programme plus poussé :

On va chercher à faire partir le conteneur le plus apte. La fonction va d'abord faire une liste des transporteurs possédants le plus de conteneurs leur appartenant. Si plusieurs transporteurs sont trouvés, c'est celui qui aura le moins de conteneurs ne lui appartenant pas qui sera sélectionné. Si encore une fois plusieurs transporteurs sont trouvés, ce sera le premier de la liste qui sera sélectionné.

Le transporteur sélectionné sera alors vidé des conteneurs qui ne lui correspondent pas (par placement de ceux-ci dans un espace de stockage) puis le transporteur partira.

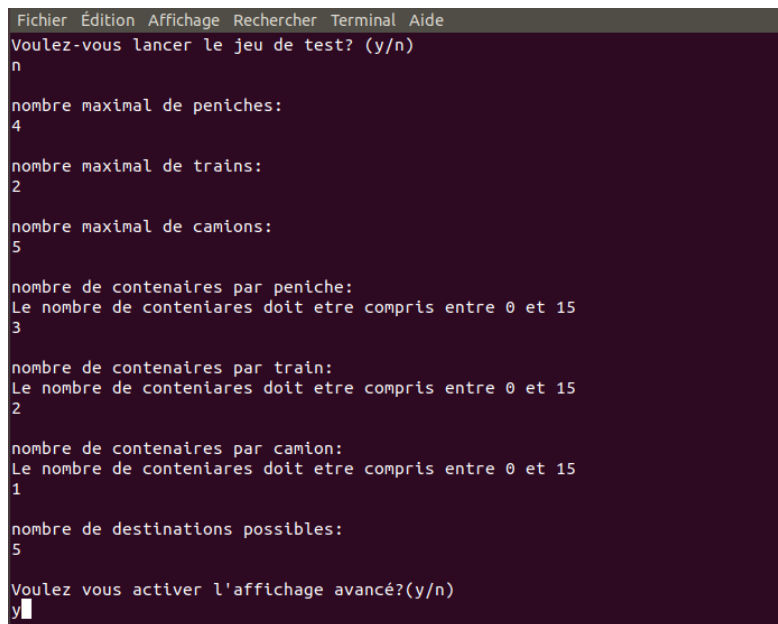
Espace de stockage

L'espace de stockage a pour but de rester vide le plus souvent possible c'est pourquoi il est rempli en dernier recours et est déchargé en premier. En effet, la zone de stockage marche de la même manière que les transporteurs classiques à la différence qu'il est prioritaire. C'est-à-dire que lorsqu'un portique va se libérer, c'est l'espace de stockage qui va vérifier s'il peut déplacer un de ses conteneurs et c'est seulement si ce n'est pas le cas que les processus des transporteurs vont pouvoir faire leur travail.

File de camions

Une file de camions est disponible. Cette file n'est pas directement accessible par les portiques. Elle entre en jeu en cas d'interblocage (cf. Godlike) ou lorsqu'un camion s'en va. Lorsqu'un camion s'en va, le camion en tête de file prend directement sa place.

Affichage



```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
Voulez-vous lancer le jeu de test? (y/n)
n

nombre maximal de péniches:
4

nombre maximal de trains:
2

nombre maximal de camions:
5

nombre de conteneurs par péniche:
Le nombre de conteneurs doit être compris entre 0 et 15
3

nombre de conteneurs par train:
Le nombre de conteneurs doit être compris entre 0 et 15
2

nombre de conteneurs par camion:
Le nombre de conteneurs doit être compris entre 0 et 15
1

nombre de destinations possibles:
5

Voulez vous activer l'affichage avancé?(y/n)
y
```

Figure 0-3 Capture d'écran du paramétrage

L'affichage commence par le paramétrage, ensuite deux affichages sont disponibles :

- Un affichage classique listant les différents transporteurs avec leurs conteneurs (représentés par des nombres qui indiquent leur destination) ainsi qu'un affichage des actions en cours
- Un affichage plus poussé permettant d'observer le mouvement en direct des portiques. Seuls les conteneurs sont représentés (encore une fois par des nombres indiquant leur destination) avec les portiques (représenté par une ligne de '#').

Les deux affichages sont rafraichis par un système de sémaphores. Le processus de rafraichissement de l'affichage est débloqué par sémaphore lors d'une action importante (déplacement d'un portique, chargement ou déchargement d'un conteneur).

```

Fichier  Édition  Affichage  Rechercher  Terminal  Aide

T0 dest:2    T1 dest:3    T2 dest:1    T3 dest:0
  2          3          2          0
  2          3          1          0
 -1          3          -1         0
 -1          3          0          0
  1          0          0          -1

T4 dest:1    T5 dest:1    T6 dest:3    T7 dest:2
  3          -1         3          2
                 3          2
                 3          2
                 -1         2

Transporteur qui part
dechargement terminé
chargement terminé
deblocage des process
dechargement terminé

Peniche  Train  Camion

```

Figure 0-4 Capture d'écran du premier affichage

Le premier se contente de lister les conteneurs et se sert d'un tableau de string rempli par la fonction DisplayInfo() pour savoir quoi afficher. Il m'a été utile pour vérifier plus facilement le déplacement des conteneurs alors que le second m'a permis de vérifier le bon déplacement des portiques.

```

Fichier  Édition  Affichage  Rechercher  Terminal  Aide

-1  2
-1  2
 0  2    1    3 1
# # # # # # # # # # # # # # # #
 0  3    2    0 3
    -1

 3  3    -1 1  -1-1-1-1-1
# # # # # # # # # # # # # # # #
-1  0    1    2 0  -1-1-1-1-1
 1  3    2
 1  2    1    2 3
#

```

Figure 0-5 Capture d'écran du second affichage

Le second affichage est plus complexe et utilise une matrice d'une taille générée de façon modulable en fonction du nombre de transporteurs et du nombre de conteneurs que chaque transporteur possède. Cette matrice est initialisée à chaque début de rafraichissement à -2. Chaque coordonnée de la matrice représente une coordonnée de la zone industrielle. Les transporteurs sont alors parcourus un à un, remplissant la matrice par le numéro de destination de chaque conteneur aux coordonnées (x,y) du conteneur.

La même chose est réalisée pour la liste d'attente des camions ainsi que l'espace de stockage. Pour finir, la ligne correspondant à la position y de chaque portique est remplie par des '#'.

Enfin, un printf est appliqué pour chaque ligne et chaque colonne de la matrice en partant du haut de celle-ci en se déplaçant vers la droite et en faisant un '\n' à chaque fin de ligne. Si le contenu de la matrice est '-2', un espace est affiché, sinon c'est le contenu de la matrice qui l'est.

PS La valeur '-2' est utilisée, car la valeur '-1' est déjà utilisée pour un emplacement vide.

Jeu de test et lecture de fichier externe

Pour réaliser des fichiers de jeux de test j'ai pensé à un format de texte à respecter :

```
4; //Nb peniches
2; //Nb Camions
4; //Nb Trains
5; //Nb contenaire par peniche
5; //Nb contenaire par Train
1; //Nb conteneurs par camion
5; //Numero destination maximale
01;05;03;02;03;01;02;03;04;05;04;02;05;03 //Ids destination (Peniche,camion,train, camion en attente)
02;05;04;03;-1; //a partir de cette ligne ids destination de chaque contenaire par transporteur
04;03;02;01;-1;
02;01;05;04;-1;
01;05;04;03;-1;
05;
04;
03;
02;
-1;03;05;01;02;
-1;01;02;03;04;
03; //A partir de cette ligne ids destination des camions en attente
01;
04;
01;
```

Figure 0-6 Capture d'écran du fichier texte du jeu de test

Le programme se charge ensuite de récupérer les informations et de les attribuer aux transporteurs si l'utilisateur a choisi d'utiliser un jeu de test.

Pour le futur du programme

Encore quelques fonctionnalités et améliorations peuvent être apportées au projet, notamment le respect de l'ordre des départs des différents transporteurs (un train ne peut pas partir si un train est en place devant lui).

Une meilleure interface graphique pourrait être envisageable. Par exemple un ajout de couleurs sur la seconde interface graphique pour mieux distinguer les différents transporteurs comme pour la première interface graphique.

De plus, les couleurs du texte qui s'affiche ne correspondent pas aux types de transporteurs indiqués en bas de l'écran d'affichage numéro 1. Les couleurs du texte ont été générées aléatoirement pour mieux distinguer les différents rafraichissements d'affichage.