

# FOSYMA

Projet Multi Agent

Victor Fleiser - Thomas Marchand

# Index

[Stratégie](#)

[Agents](#)

[Communication](#)

[Position optimale du Silo](#)

[SILO](#)

[Interblocage](#)

[Terminaison](#)

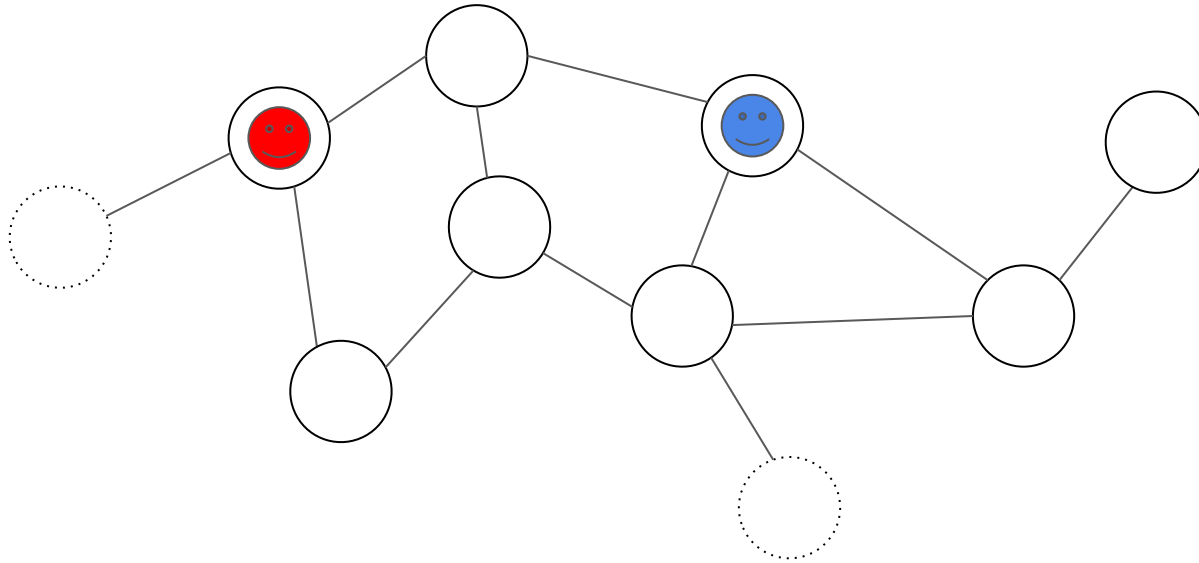
[Code changes](#)

[Extra](#)

# Stratégie

# Stratégie - phase EXPLORATION

Tous les agents découvrent les noeuds ouverts, se partagent les cartes



# Exploration décentralisée

## Avantages

- **Rapide** : Couvrage plus rapide
- **Robuste** : Un agent en panne n'empêche pas les autres agents
- **Simple** : “Trouver le noeud ouvert proche”, déjà implémenté

## Inconvénients

- **Inconvénients très mineurs**

# Stratégie - phase EXPLOITATION

## SILO

Trouve le noeud optimal

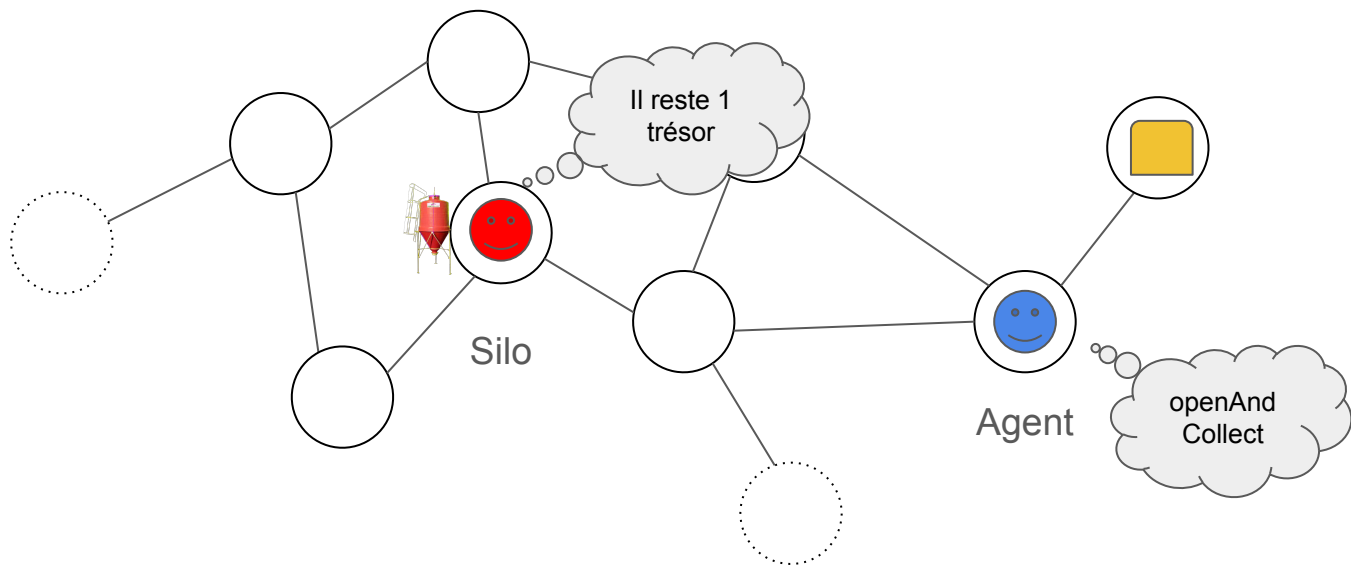
Calcule des missions adaptées aux agents

Obtient tous les trésors

## AGENTS

Trouvent le Silo

Suivent des mission adaptées à leur capacités




# Exploitation centralisée (avec Silo)

## Avantages

- **Vue globale:** Permet d'attribuer les missions avec une bonne efficacité
- **Coordination :** Peut former des équipes

## Inconvénients

- **Point de défaillance unique :** Silo en panne -> 
- **Goulot d'étranglement :** plus il y a d'agents, plus trafic élevé autour du Silo

# Approche hybride

Transition naturelle

Best of both worlds

Vitesse de l'exploration décentralisée

Efficacité de l'exploitation centralisée



# Agents

# Agents - types

- **Collecteur** : Peut explorer, déverrouiller et récupérer les trésors selon ses capacités
- **Explorateur** : Peut explorer et déverrouiller les trésors selon ses capacités.
- **Silo (non central)** : Comme explorateur.
- **Silo (Central, nommé “Silo”)** : Agit comme agent central pendant l’exploitation

# Communication

Agent1

Je veux partager X information à  
Agent2

Agent2

Agent1

Je veux partager X information à  
Agent2

Agent2

Setup



ReceiveXBehaviour

Agent1

Je veux partager X information à  
Agent2 ↓

ShareXBehaviour

Agent2

ReceiveXBehaviour

Agent1

Je veux partager X information à  
Agent2

ShareXBehaviour

X

ReceiveXBehaviour

WaitForAckBehaviour

Agent2

Agent1

Je veux partager X information à Agent2

ShareXBehaviour

WaitForAckBehaviour

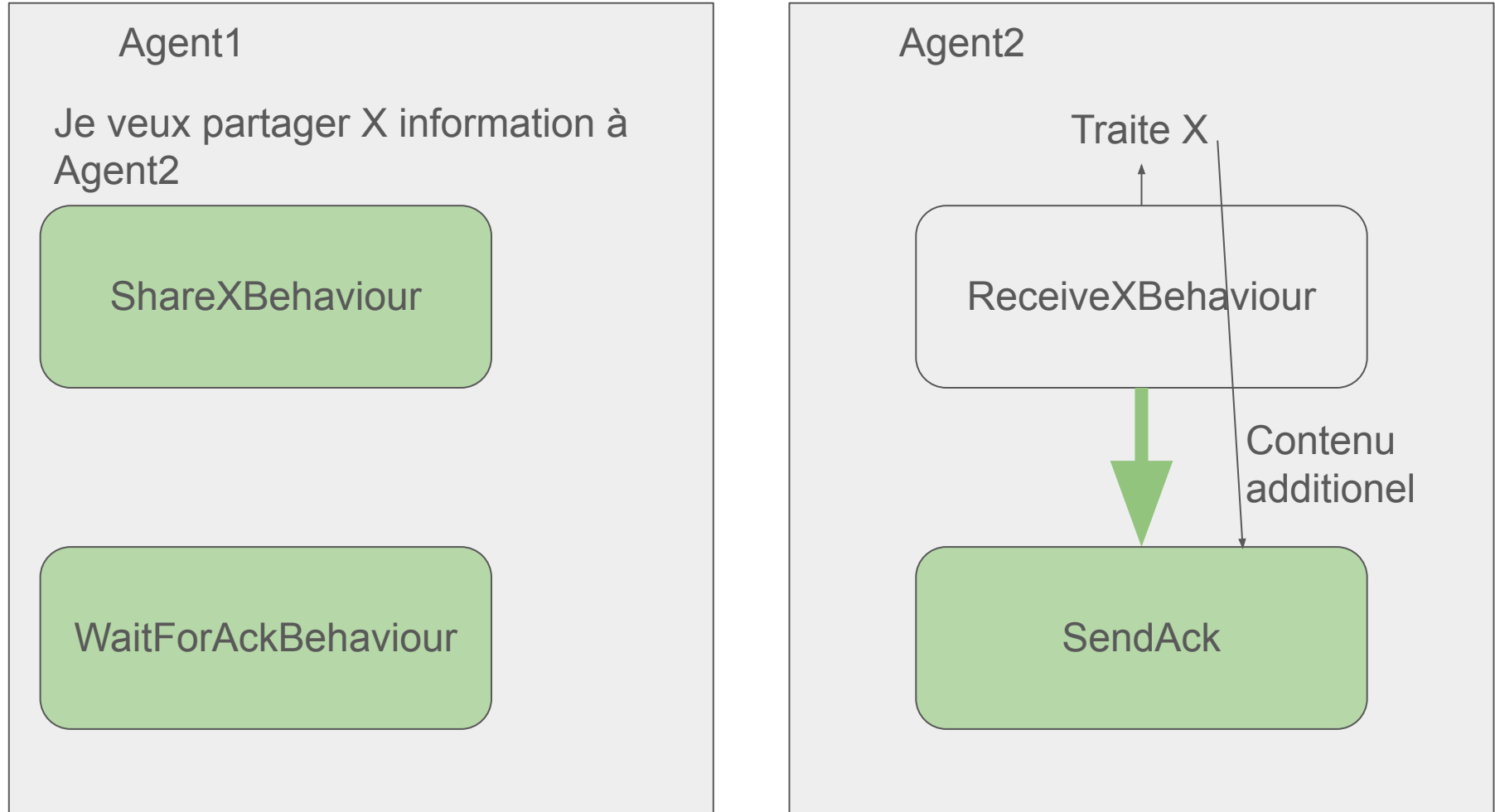
Agent2

Traite X

ReceiveXBehaviour

Contenu  
additionel

SendAck





Agent1

Je veux partager X information à  
Agent2

ShareXBehaviour

WaitForAckBehaviour

Agent2

ReceiveXBehaviour

SendAck

ACK

Contenu  
additionel



Agent1

Je veux partager X information à  
Agent2

ShareXBehaviour

ACK  
Contenu additionel

WaitForAckBehaviour

Agent2

ReceiveXBehaviour

Agent1

Traite réponse



ShareXBehaviour

Agent2

ReceiveXBehaviour

Agent1

Agent2

ReceiveXBehaviour

# Communication - Behaviours et Protocoles

## [SHARE-TOPO]

ShareMapBehaviour -> ReceiveMapBehaviour

## [SHARE-SILO-KNOWLEDGE]

ShareSiloKnowledgeBehaviour -> ReceiveSiloKnowledgeBehaviour

## [STRATEGY]

CommunicateWithSiloBehaviour -> ReceiveCommunicationWithSiloBehaviour

WaitForAckBehaviour

SendAck

# Communication - Behaviours et Protocoles

Autres communications qui n'utilisent pas de ACK pour prioriser la vitesse

## **[MOVE-REQUEST]**

SendMoveRequestBehaviour -> ReceiveMoveRequestBehaviour

## **[MISSION]**

Silo -> ReceiveMissonsBehaviour (Agent)

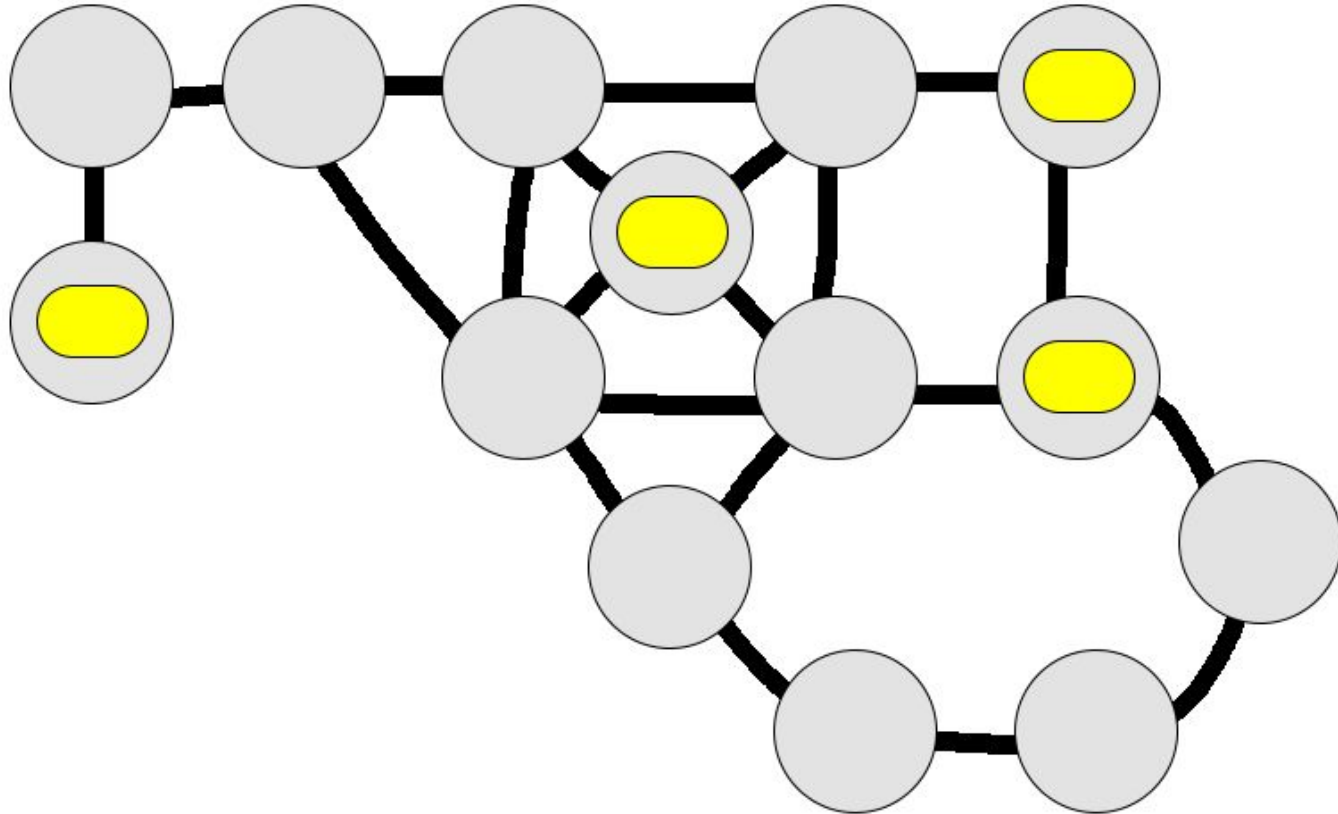
## **[STRATEGY]**

Agent -> ReceiveCommunicationWithSiloBehaviour (Silo)

# Position optimale du Silo

Quelle est la position optimale pour le Silo ?

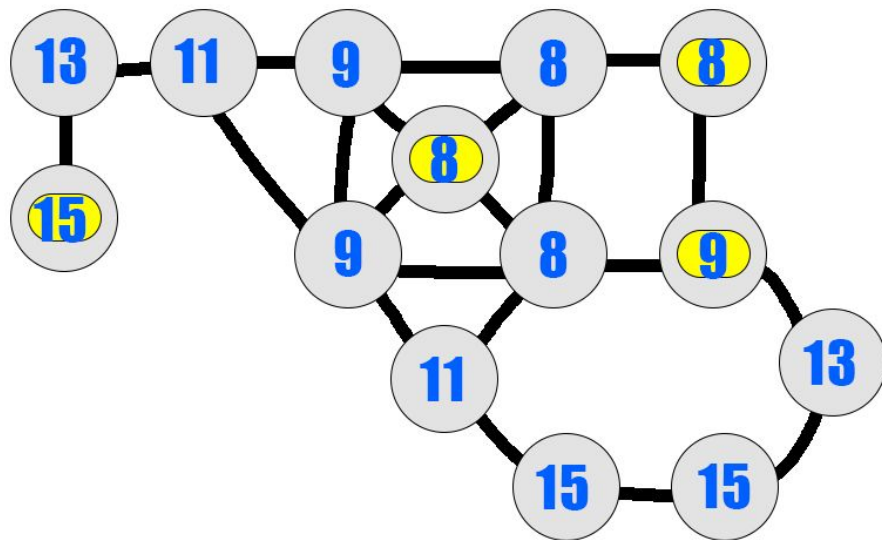
-> on cherche à minimiser un score attribué à chaque noeud





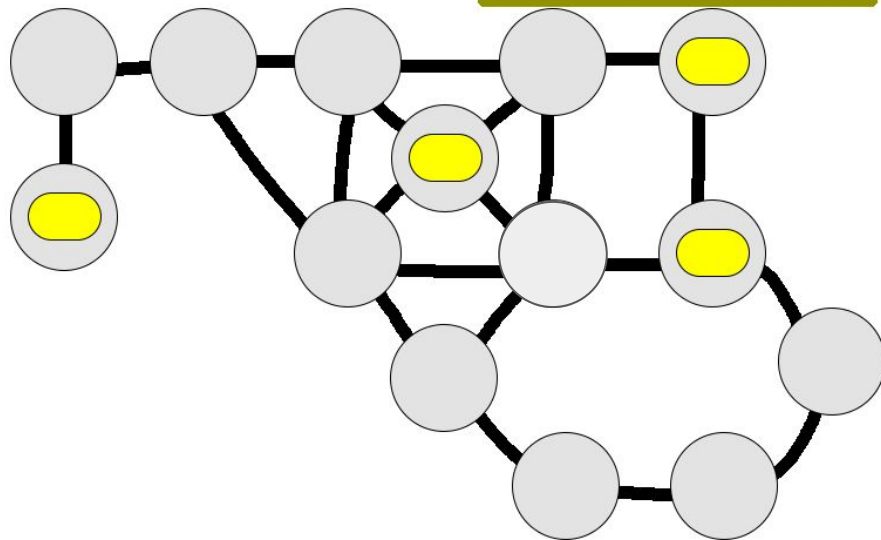
On veut être proche des trésors  
Distance =  $\sum$ distances

## DISTANCE AUX TRESORS



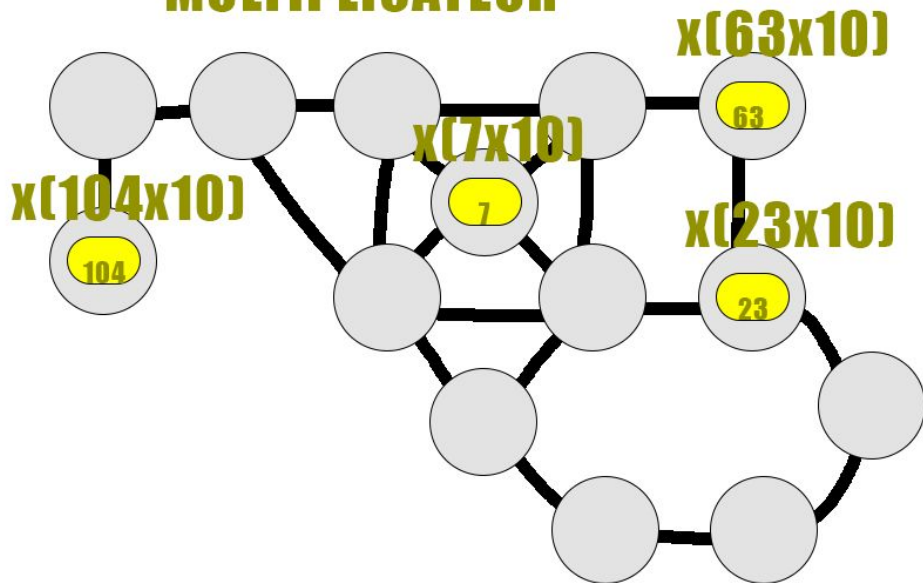
On utilise un multiplicateur pour  
prendre en compte différents critères

$$\text{SCORE} = \text{DISTANCE} \times \text{MULTIPLICATEUR}$$

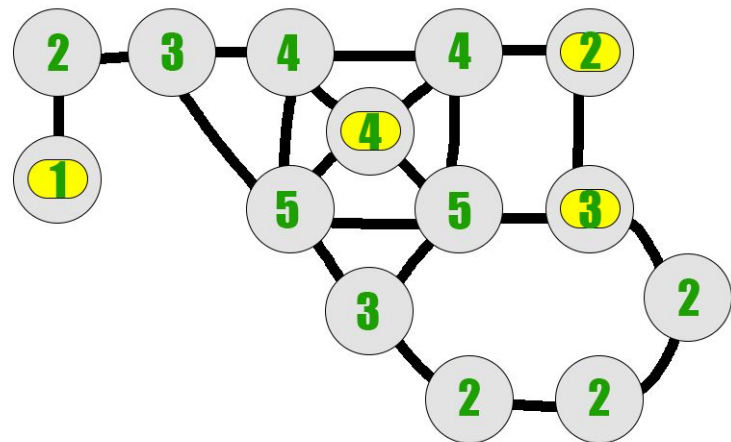


On ne veut pas occuper un trésor  
 Multiplicateur  $\times$  (amount  $\times 10$ )

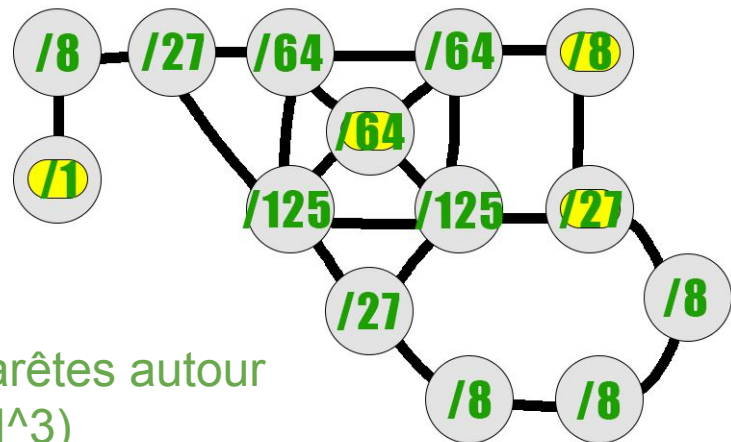
## MULTIPLICATEUR



## NOMBRE D'ARRETES



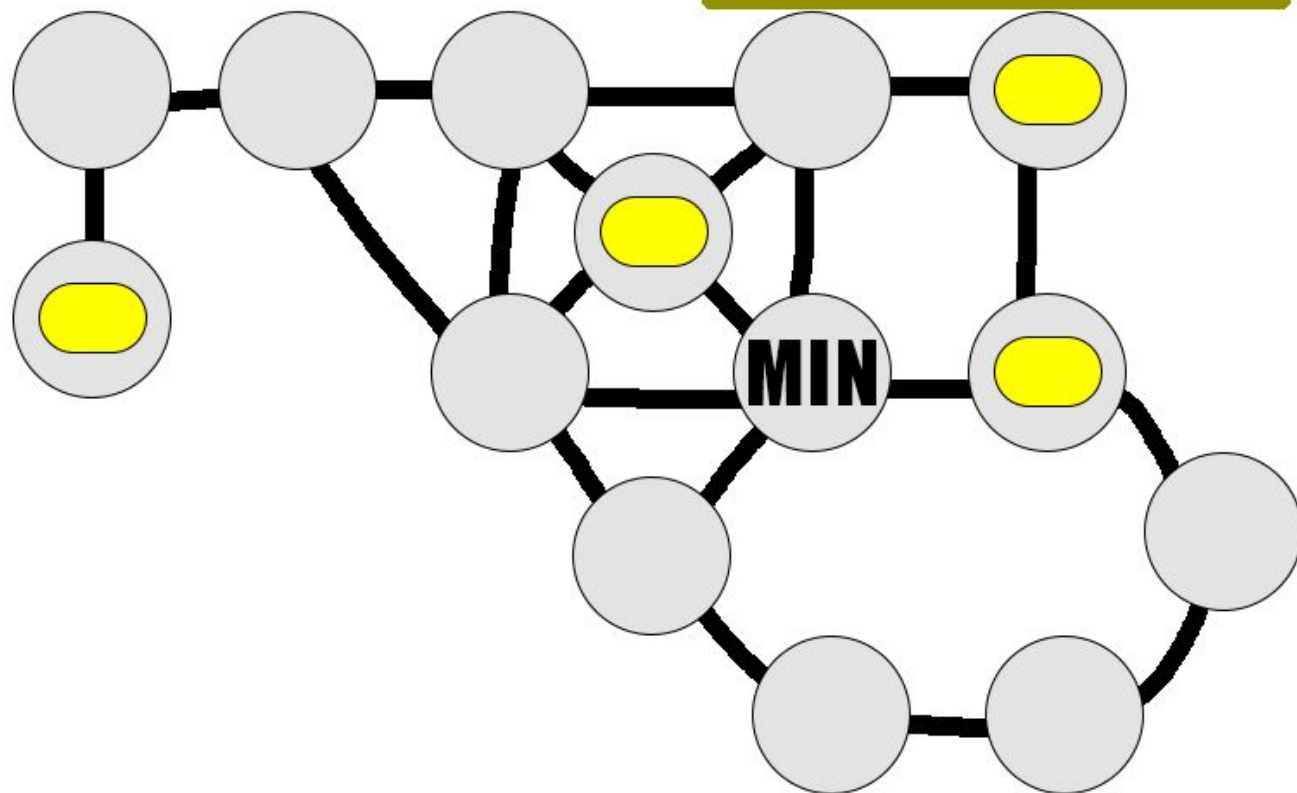
## MULTIPLICATEUR



On veut avoir pleins d'arêtes autour  
 Multiplicateur  $/ (|\text{arêtes}|^3)$



**SCORE = DISTANCE x MULTIPLICATEUR**



SILO

# Silo - General Ticker

onTick :

1. updateOpenNodeStatus
2. updateTreasureStatus
3. updateAssignment
4. calculateTasks
5. tryAssigningTasks

# Silo - 1. updateOpenNodeStatus

Mets à jour les noeuds ouverts du Silo.

CLOSED -> enlevés

OPEN -> ajout dans openNodeStatus

## Silo - 2. updateTreasureStatus

Mets à jour les trésors du Silo.

Regarde ses trésors de exploCoopBehaviour



## Silo - 3. updateAssignment

Mets à jour le statut des trésors/noeuds/agents

Pos	Typ	Amt	Str	Lck	Lock	Assignment	End T	Try	L.Att	L.Upd
44	GOL	15	2	0	Open	available		0		53197
58	DIA	80	2	4	Lock	available		0		52903
5	GOL	25	0	0	Open	available		0		52899
h40	DIA	11	0	0	Open	available		0		53539
90	GOL	25	3	2	Lock	available		0		52699
70	GOL	25	2	3	Lock	available		0		52896
h1	GOL	102	2	3	Open	available		0		53434
95	GOL	20	1	1	Open	available		0		53293
M10	GOL	15	3	2	Open	available		0		53659
M35	GOL	40	3	2	Lock	available		0		53257
10	GOL	10	1	2	Open	available		0		52935

## Silo - 4. calculateTasks (**Missions**)

$$\text{Importance} = \text{Amount} / (\text{nbEssai} + 1)$$

unlockAndCollect	: déverrouillage et collecte d'un trésor	x 2.5
collect	: collecte d'un trésor	x 1.5
collectWithLosses	: collecte avec pertes d'un trésor	x 1.25
unlock	: déverrouillage d'un trésor	x 1

## Silo - 4. calculateTasks (**Missions**)

discoverOpenNode : explorer un noeud ouvert

$$\text{Importance} = 20 / (\text{nbEssai} + 1)$$

explore : exploration de la carte aléatoire

$$\text{Importance} = 1.1$$

## Silo - 4. calculateTasks (Missions)

Task Type	Imp.	Pos	Res	Amt	Str	Lck
unlockAndCollect	200,0	58	DIA	80	2	4
collect	153,0	h1	GOL	102		
collectWithLosse	127,5	h1	GOL	102		
unlockAndCollect	100,0	M35	GOL	40	3	2
unlock	80,0	58			2	4
unlockAndCollect	62,5	90	GOL	25	3	2
unlockAndCollect	62,5	70	GOL	25	2	3
unlock	40,0	M35			3	2
collect	37,5	5	GOL	25		
collectWithLosse	31,3	5	GOL	25		
collect	30,0	95	GOL	20		
unlock	25,0	90			3	2
unlock	25,0	70			2	3
collectWithLosse	25,0	95	GOL	20		
collect	22,5	44	GOL	15		
collect	22,5	M10	GOL	15		
discoverOpenNode	20,0	11				
collectWithLosse	18,8	44	GOL	15		
collectWithLosse	18,8	M10	GOL	15		
collect	16,5	h40	DIA	11		
collect	15,0	10	GOL	10		
collectWithLosse	13,8	h40	DIA	11		
collectWithLosse	12,5	10	GOL	10		
explore	1,0					

# Silo - 5. tryAssigningTasks

Attribue les tâches

## **SOLO**

Pour chaque mission:

    Pour chaque agent:

        S'il peut faire la mission toute seule:

            Mission donnée

# Silo - 5. tryAssigningTasks

## **COLLAB**

Pour chaque groupe d'agent (taille 2 à 4):

Pour chaque mission:

Pour chaque combinaison d'agents:

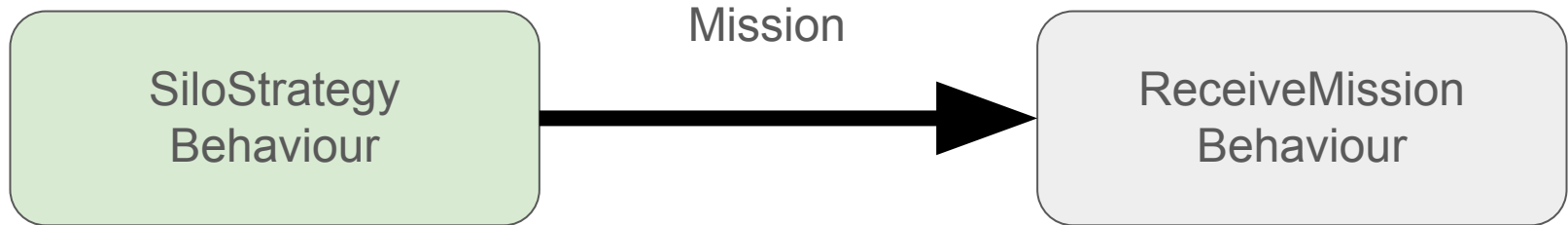
S'ils peuvent faire la mission ensemble:

Mission donnée

## Silo - 5. tryAssigningTasks

Un agent aura toujours au moins une tâche compatible: “explore”.

Il peut aussi attendre au Silo dans ce cas.



# Mission

Dans ExploCoopBehaviour: setup mission

missionType	goToSilo	findSilo	explore	waitForSiloMessage	exploreOpenNodes	unlockTreasure	pickUpTreasure	unlockAndPickUpTreasure
missionDestinations	Silo neighbor nodes	null?	null	Silo neighbor nodes		closestnodes(pos,  agents )	closestnodes(pos,  agents )	closestnodes(pos,  agents )
goalDestination	Closest node in destinations	Random node	Random node	Closest node in destinations	openNode	Random node from destinations	Random node from destinations	Random node from destinations
pathToDestination	Path to closest node	null	null	Path to closest node	null	null	null	null
goalPriority	400	1 + rand(0~100)	0	100	300	650	625	675
isStationary	false	false	false	If at closest node	-	-	false	false
missionTimeout	15s	10s	10s	1s + timeBetweenTaskReset - ((5 - tasksToConsiderPortion) * timeBetweenTasksReset / 5))	distance*500ms + 5s	distance*1s + 5s	distance*1s + 5s	distance*1s + 5s

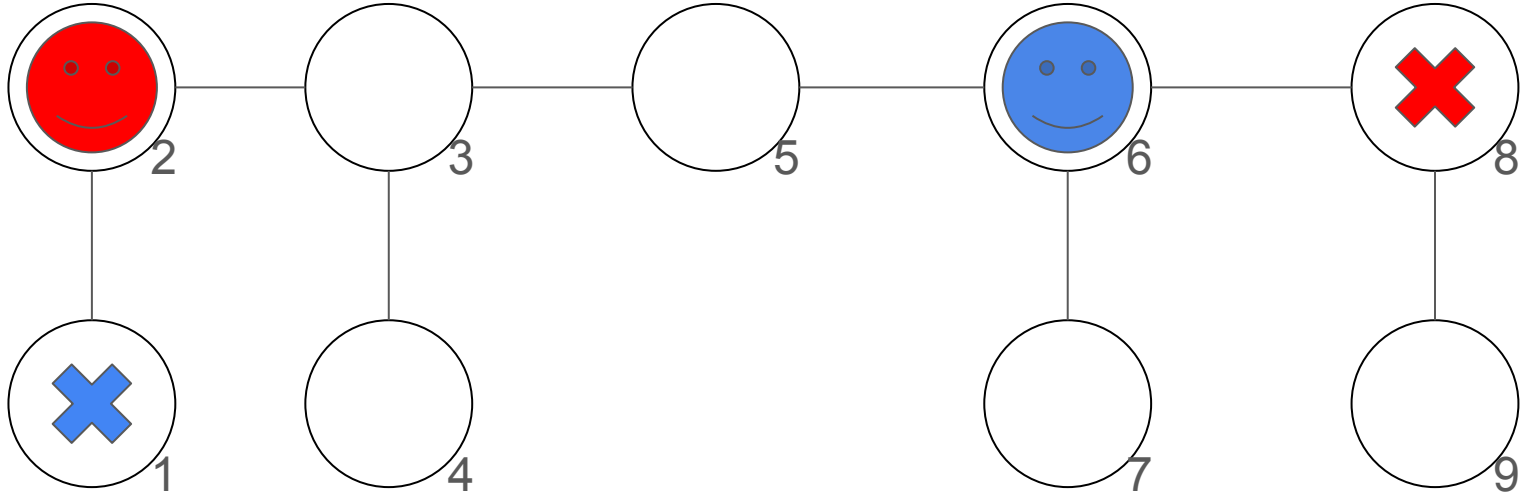


# Interblocage

# Interblocage 1 (cas de base)

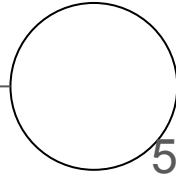
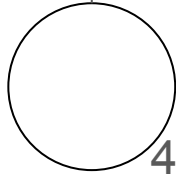
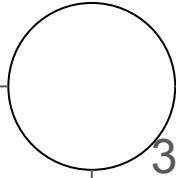
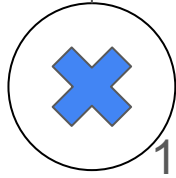
Priority: 600

Priority : 350

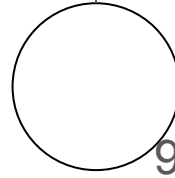
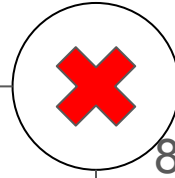
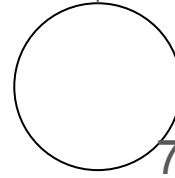
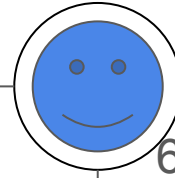


# Interblocage 1 (cas de base)

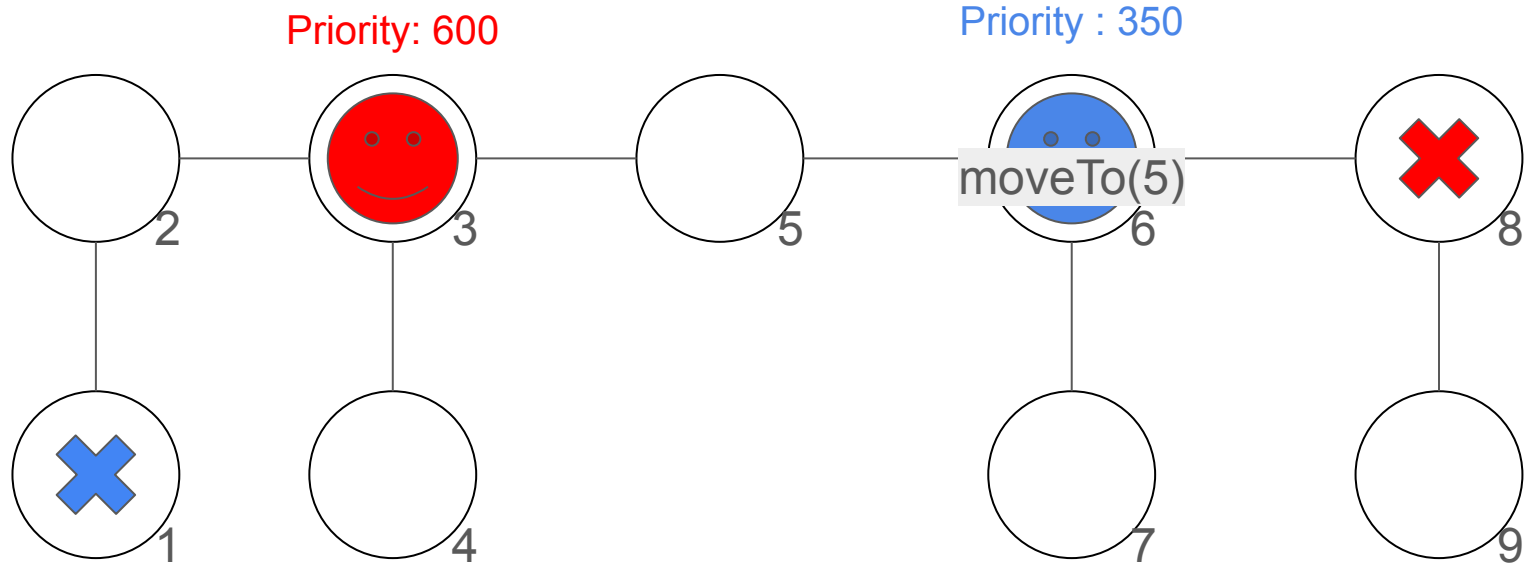
Priority: 600



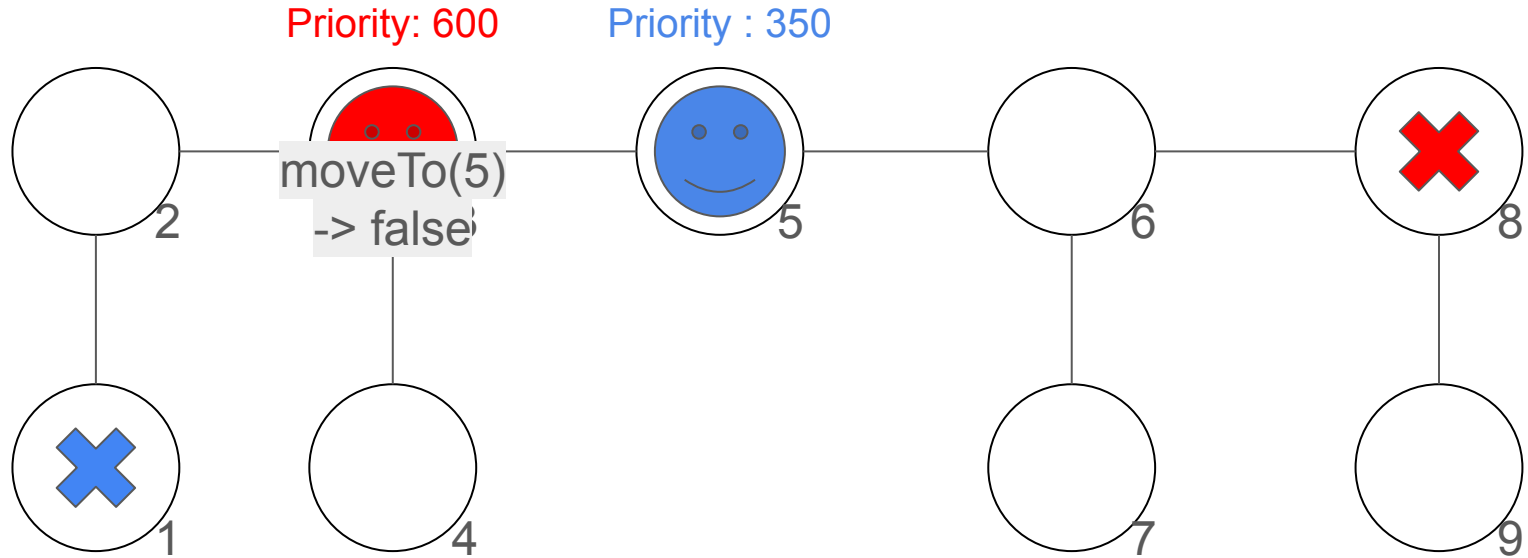
Priority : 350



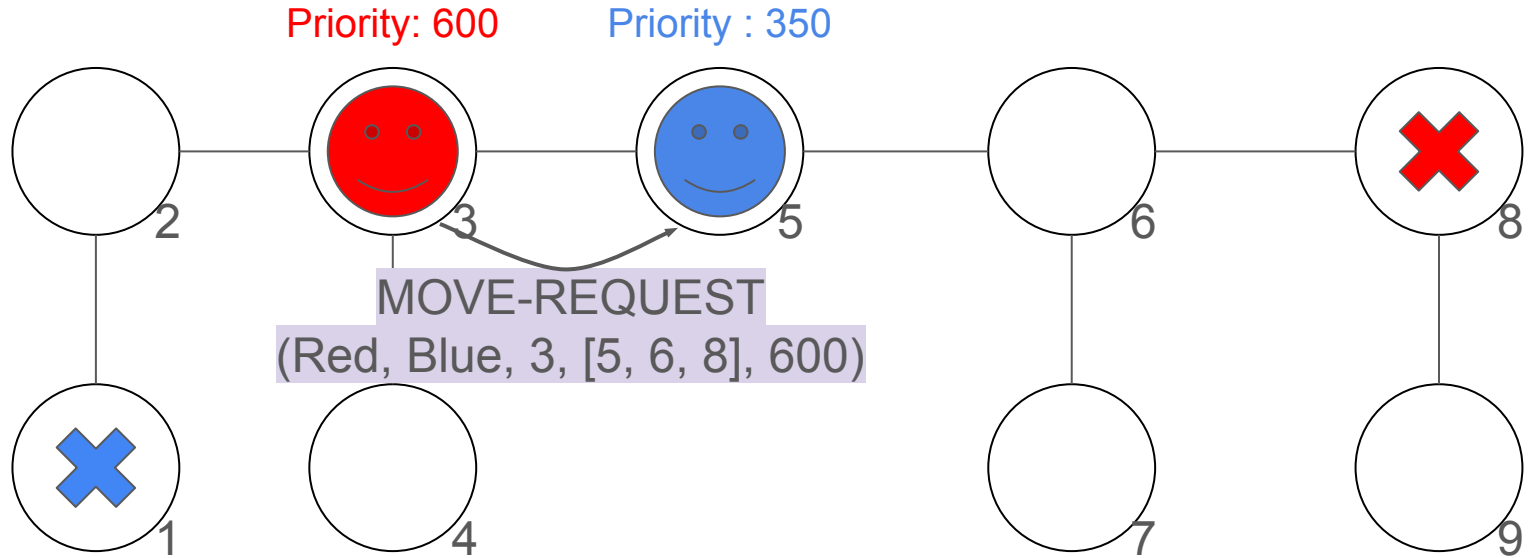
# Interblocage 1 (cas de base)



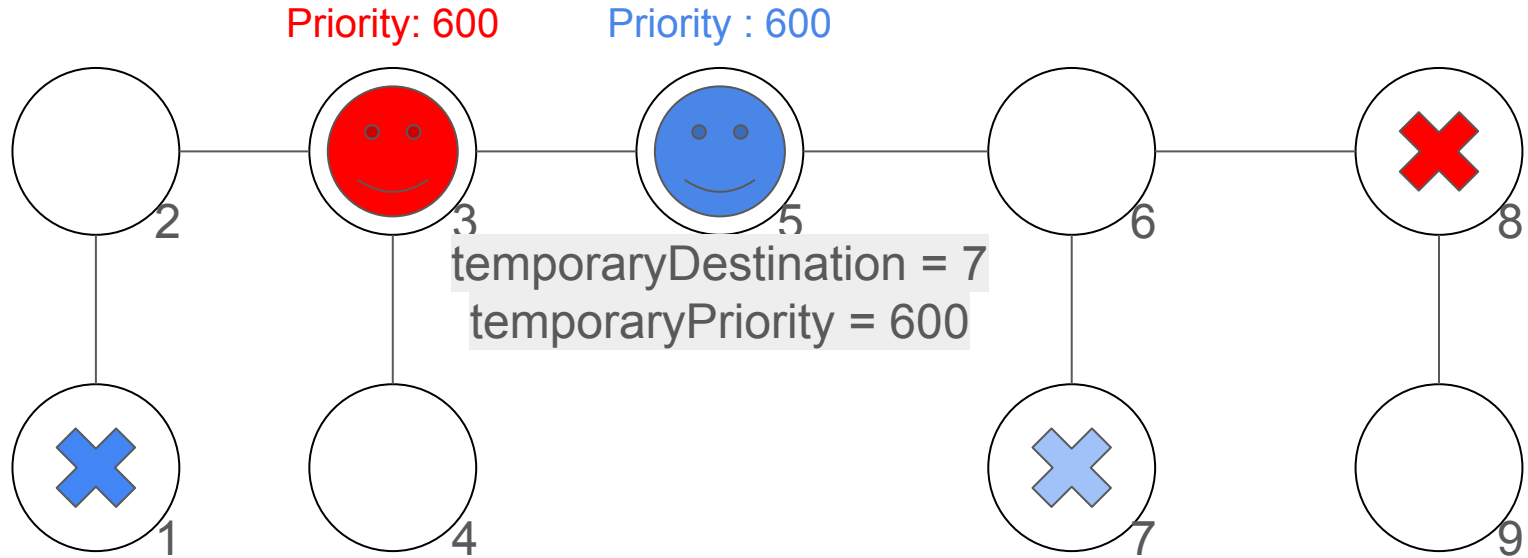
# Interblocage 1 (cas de base)



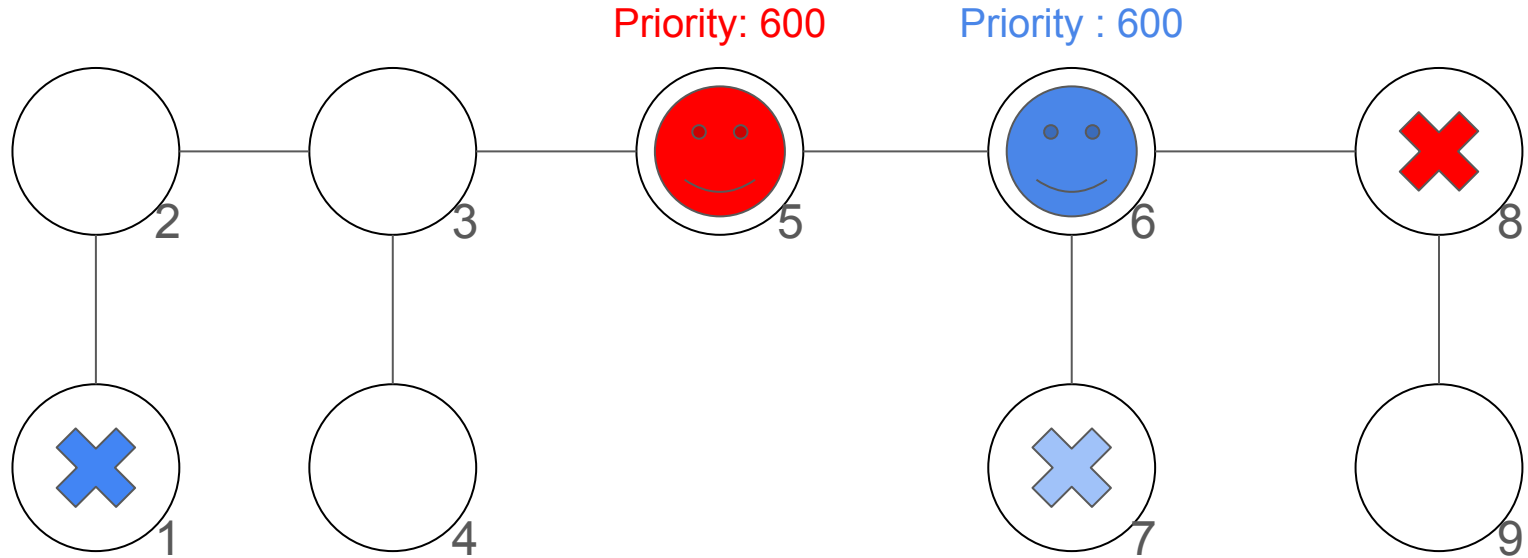
# Interblocage 1 (cas de base)



# Interblocage 1 (cas de base)

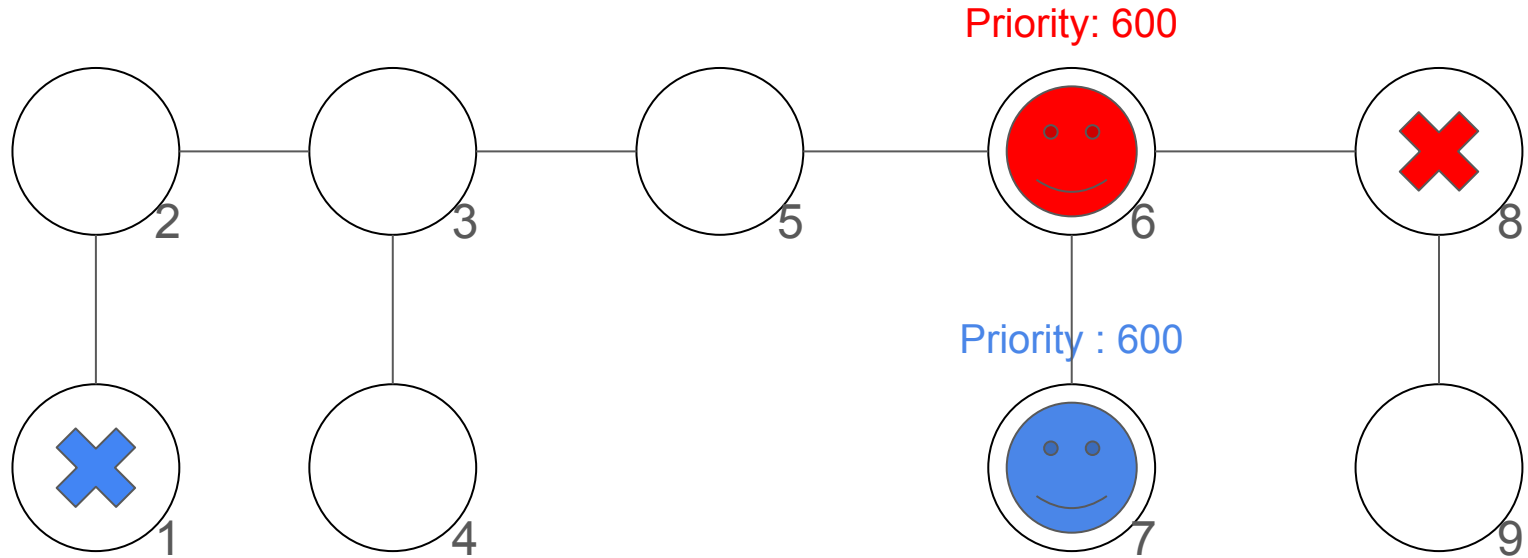


# Interblocage 1 (cas de base)

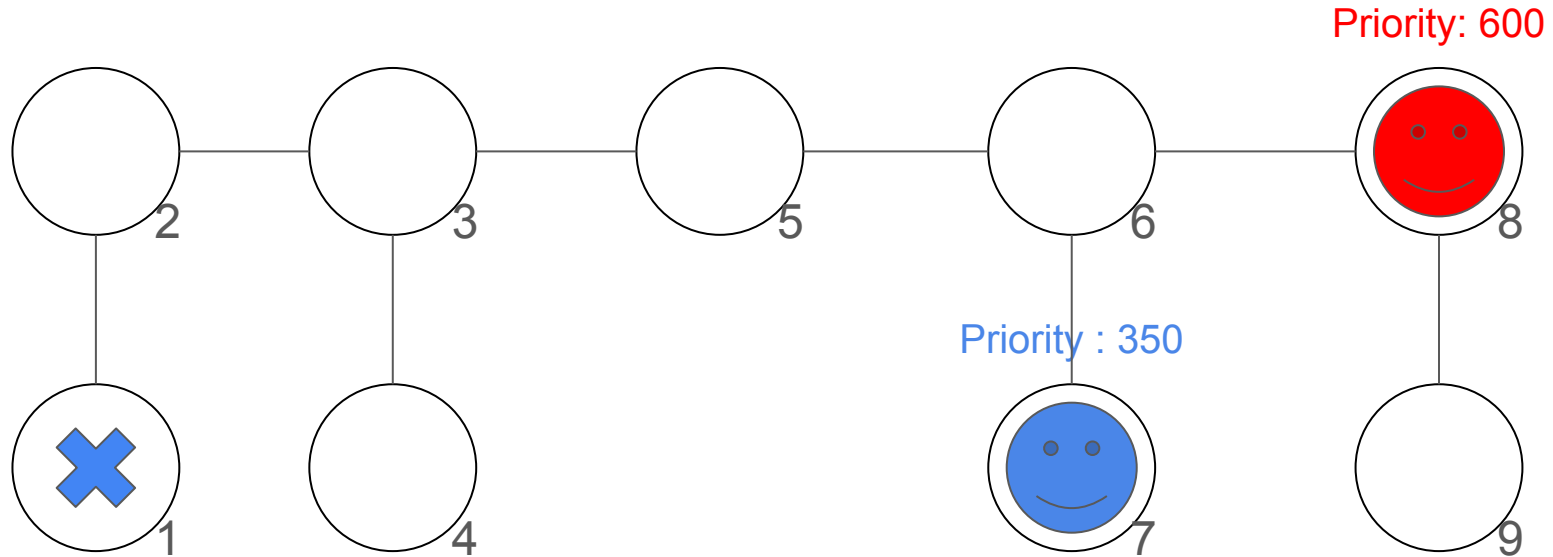




# Interblocage 1 (cas de base)

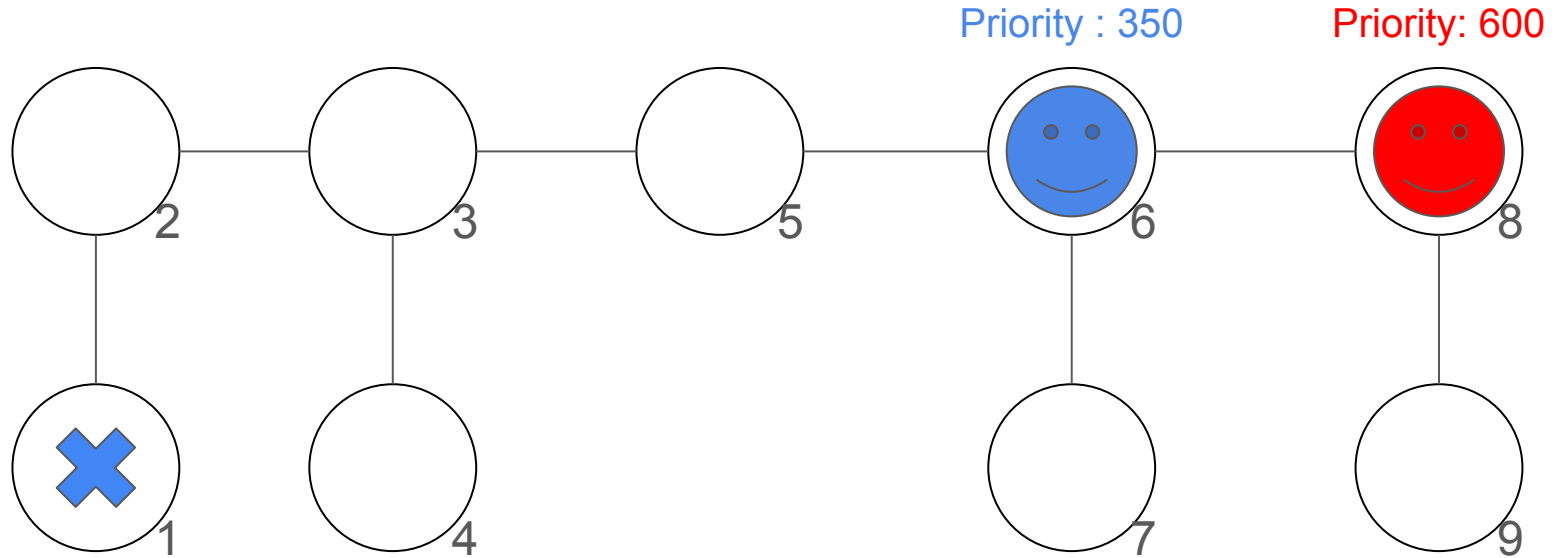


# Interblocage 1 (cas de base)

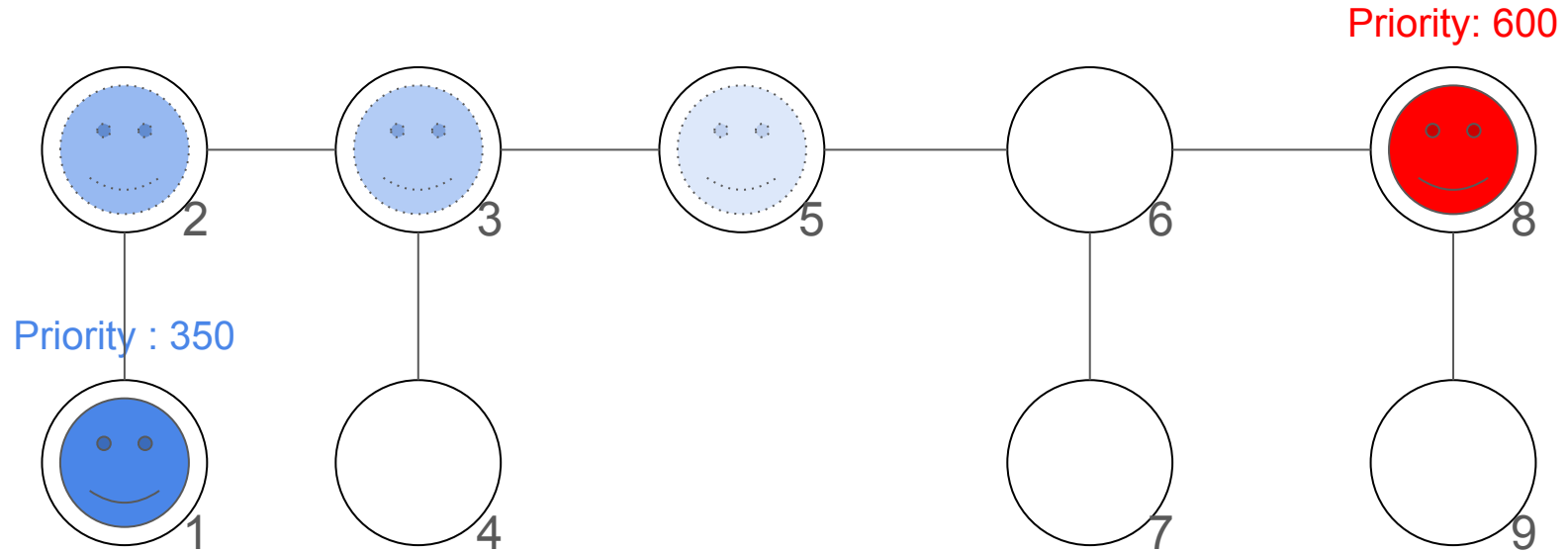


L'agent repart tout de suite sans envoyer de ACK.  
Fonctionne bien quand la vitesse d'exéc est similaire entre les deux agents.  
(Le cas ici, sinon on aura d'autres MoveRequest)

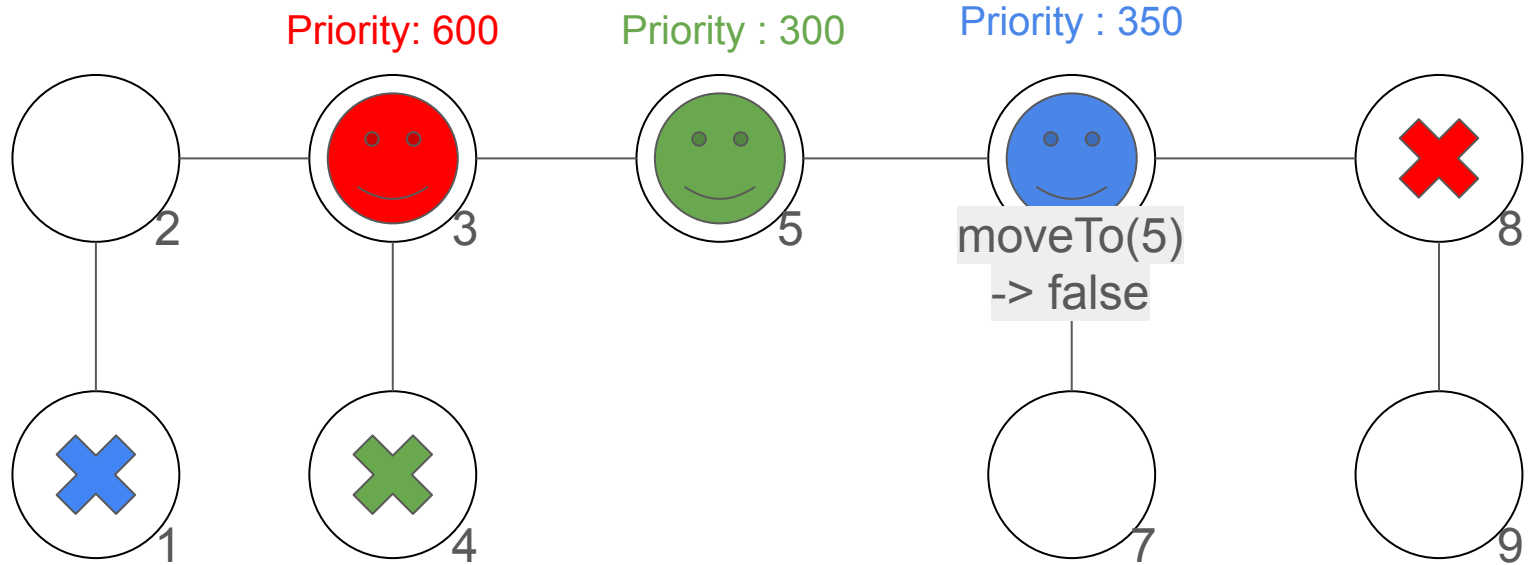
# Interblocage 1 (cas de base)



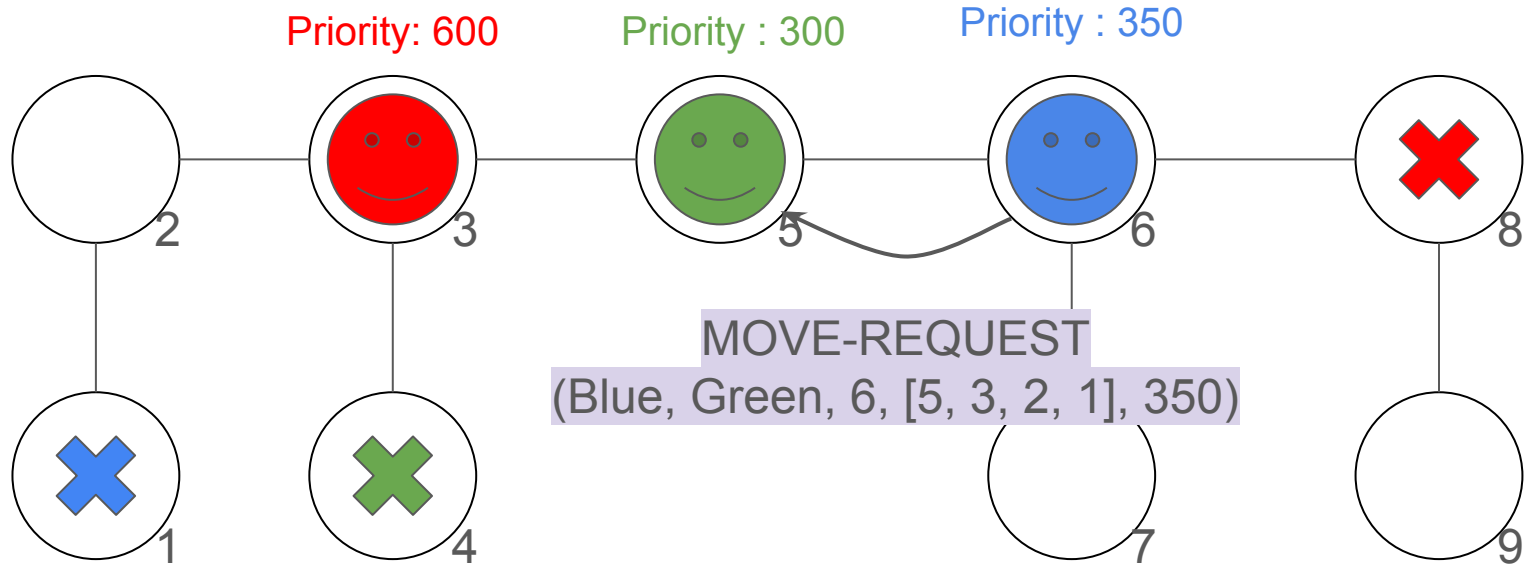
# Interblocage 1 (cas de base)



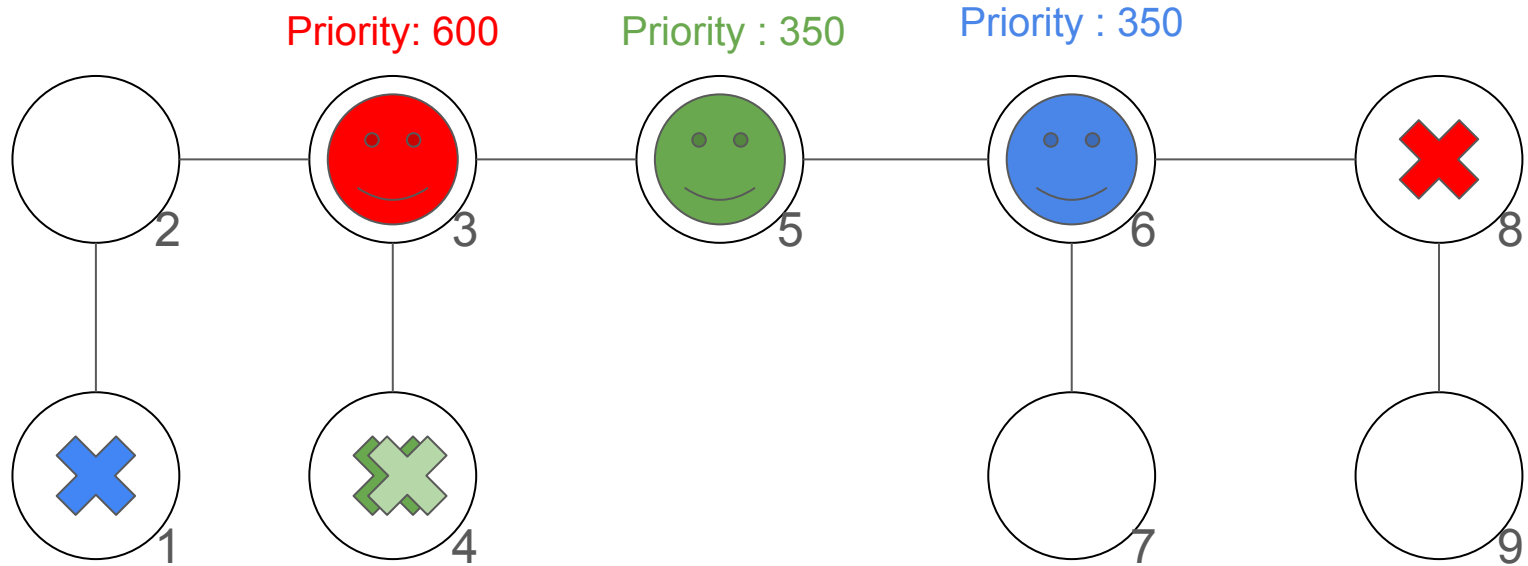
## Interblocage 2 (agents au milieu)



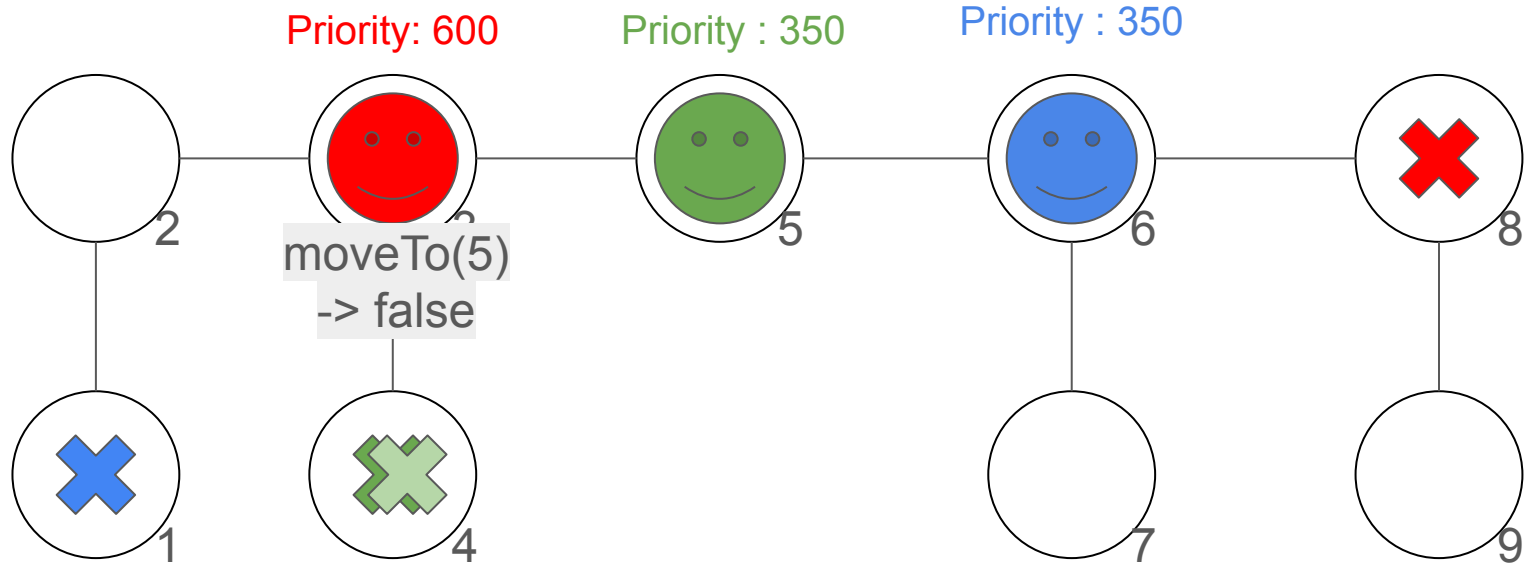
## Interblocage 2 (agents au milieu)



## Interblocage 2 (agents au milieu)

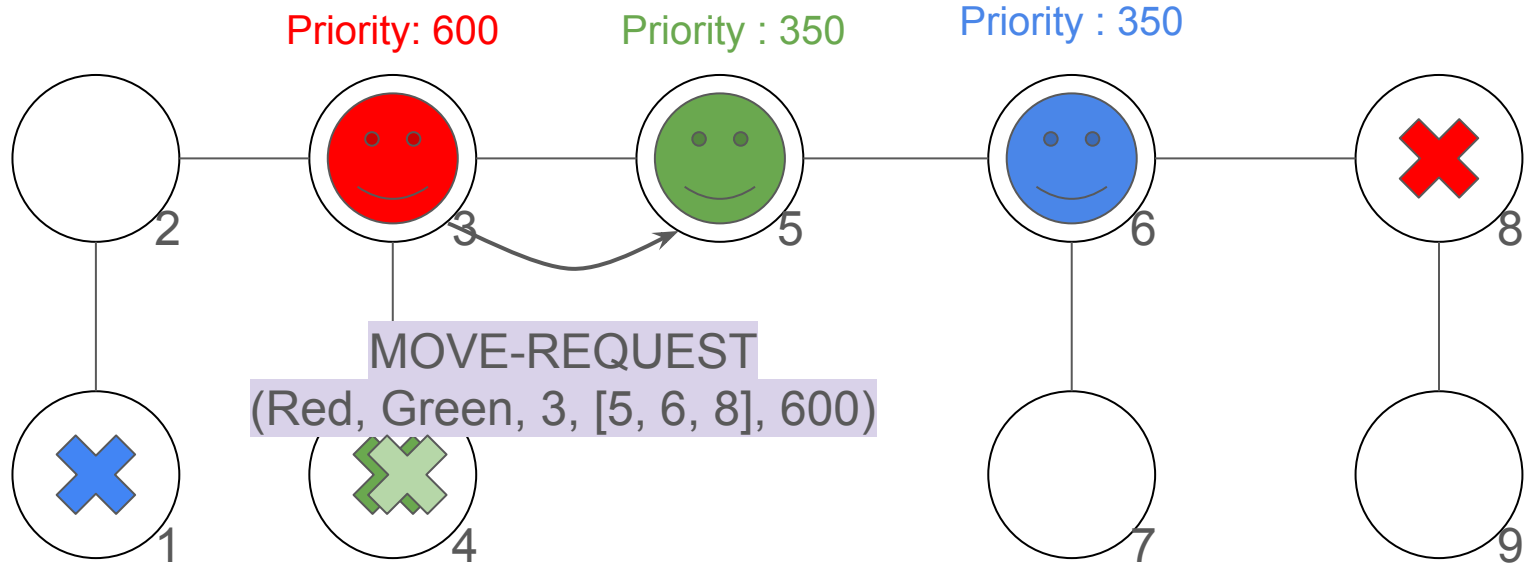


## Interblocage 2 (agents au milieu)

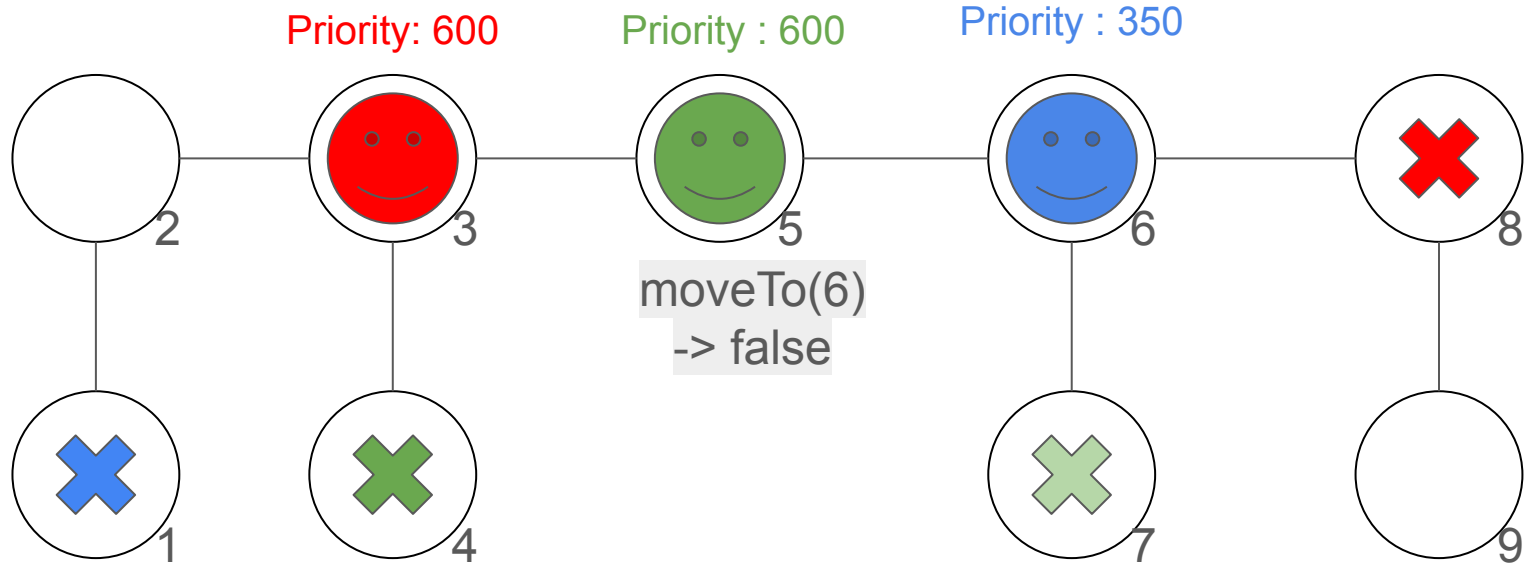




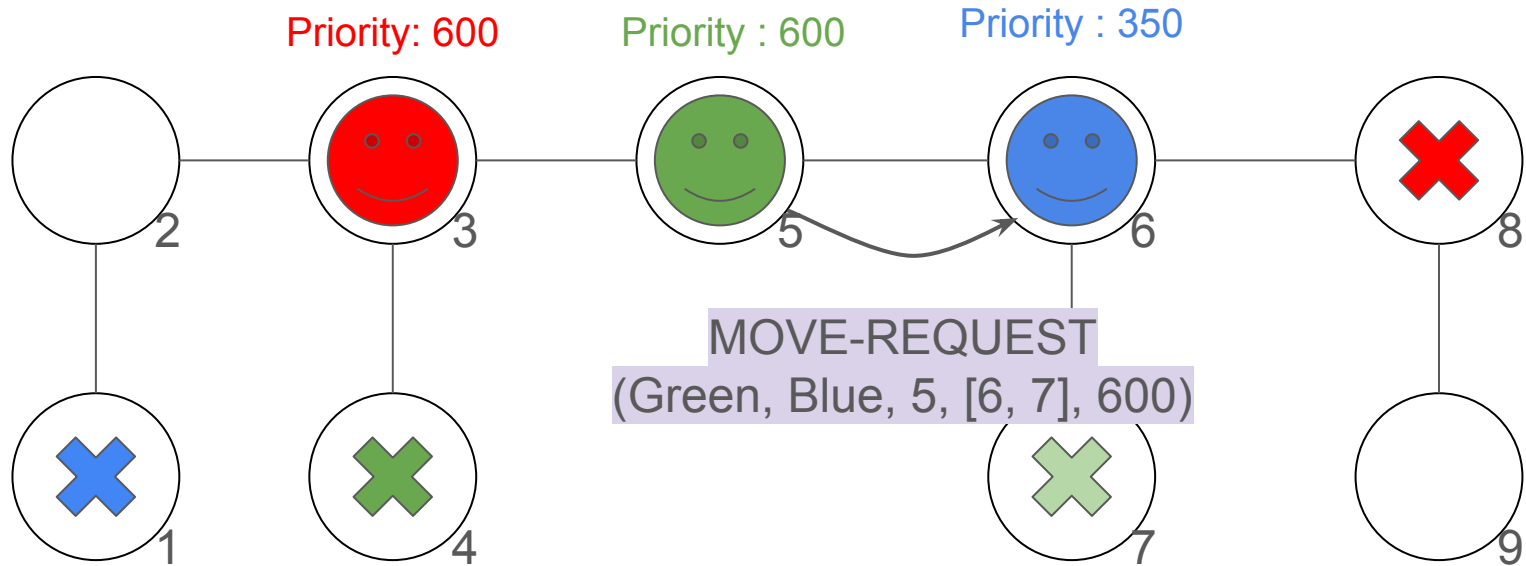
## Interblocage 2 (agents au milieu)



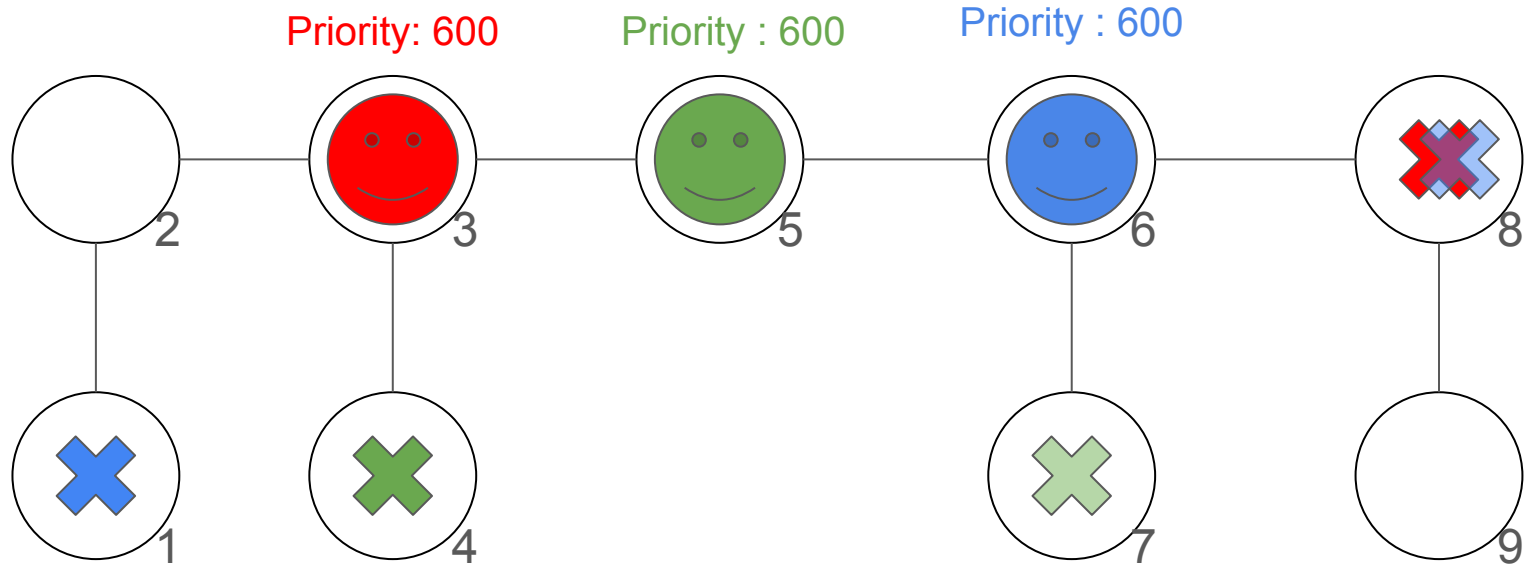
## Interblocage 2 (agents au milieu)



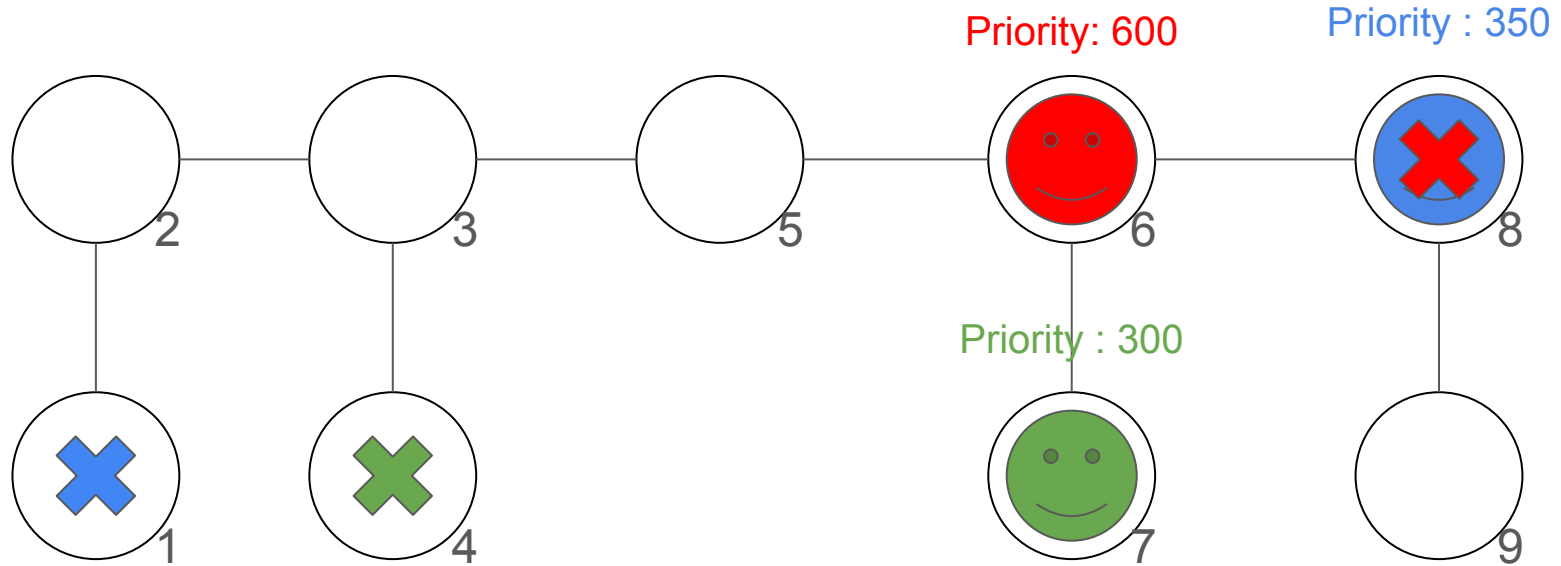
## Interblocage 2 (agents au milieu)



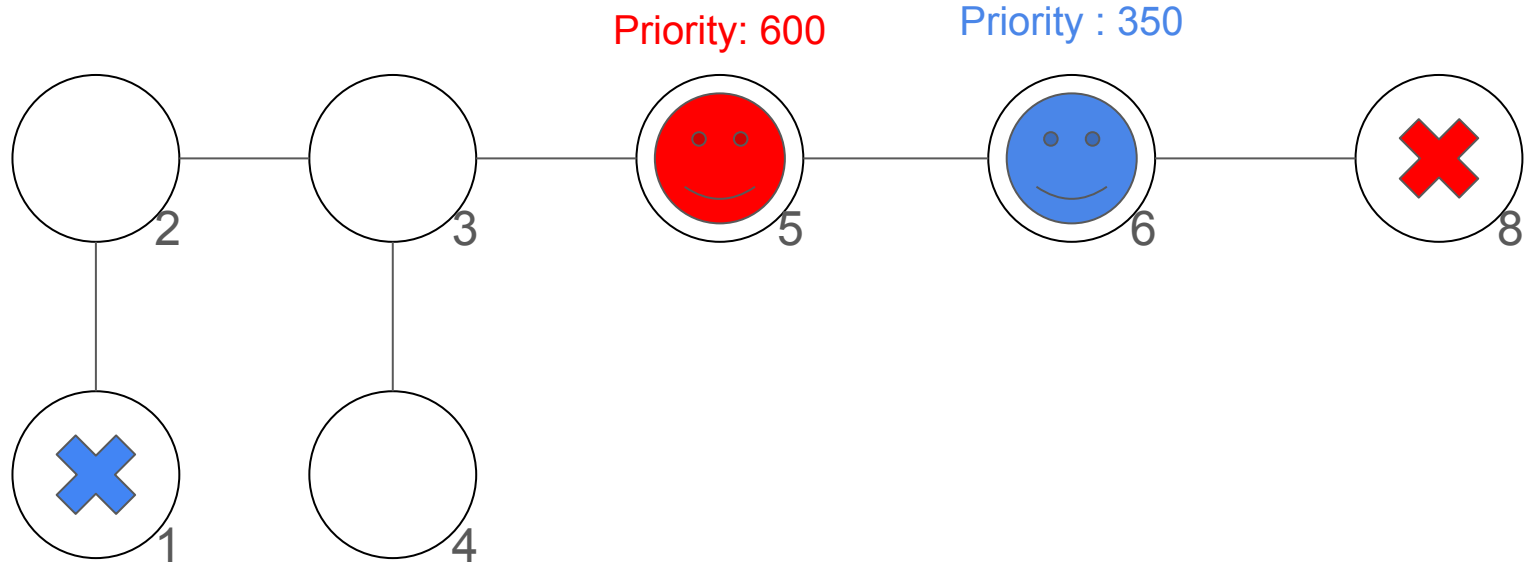
## Interblocage 2 (agents au milieu)



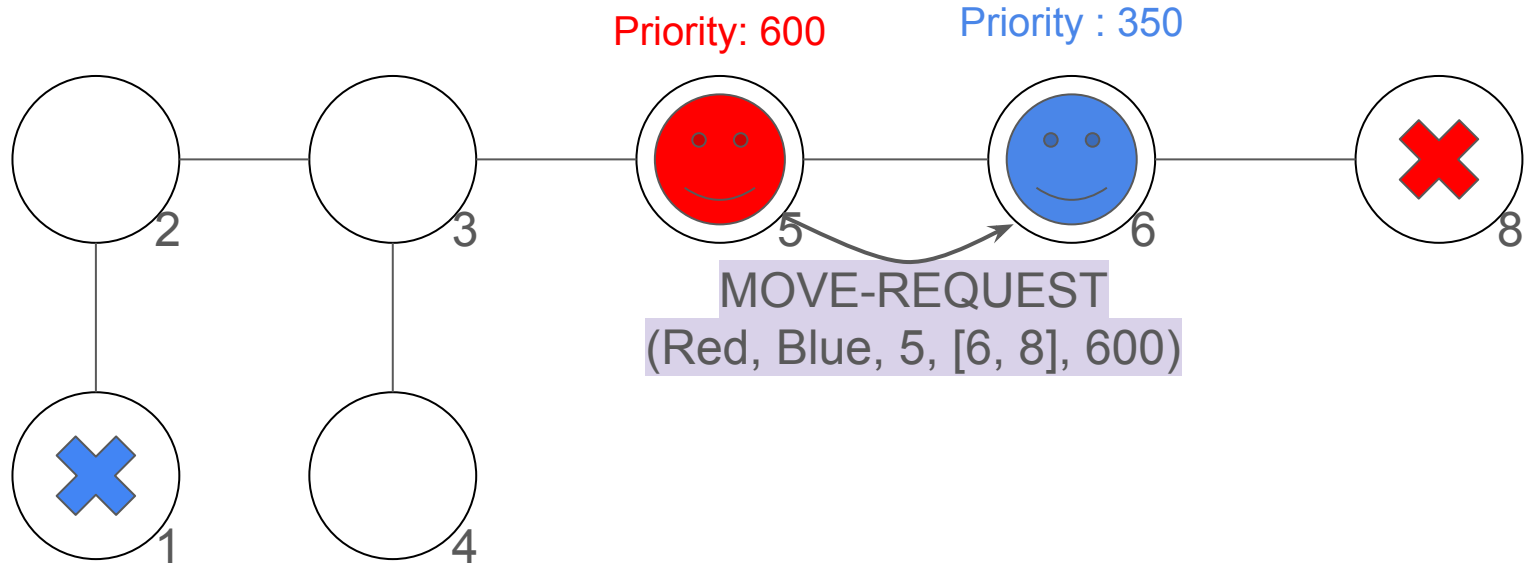
## Interblocage 2 (agents au milieu)



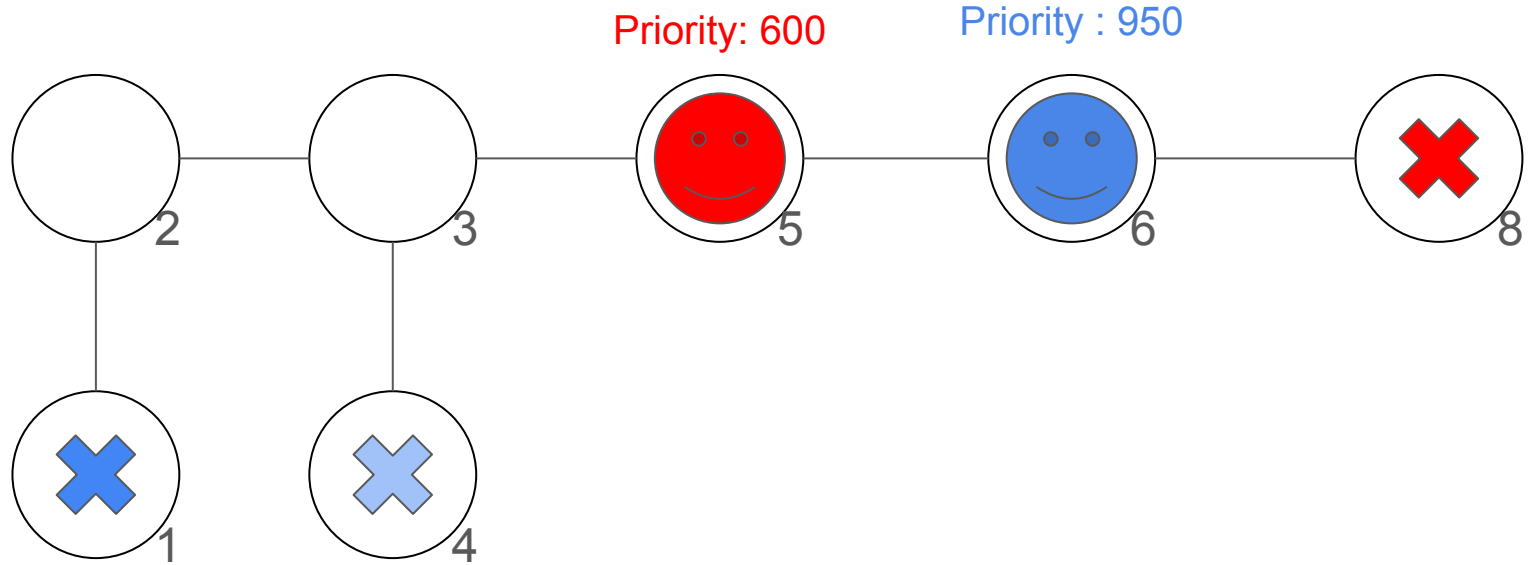
## Interblocage 3 (impasse)



## Interblocage 3 (impasse)

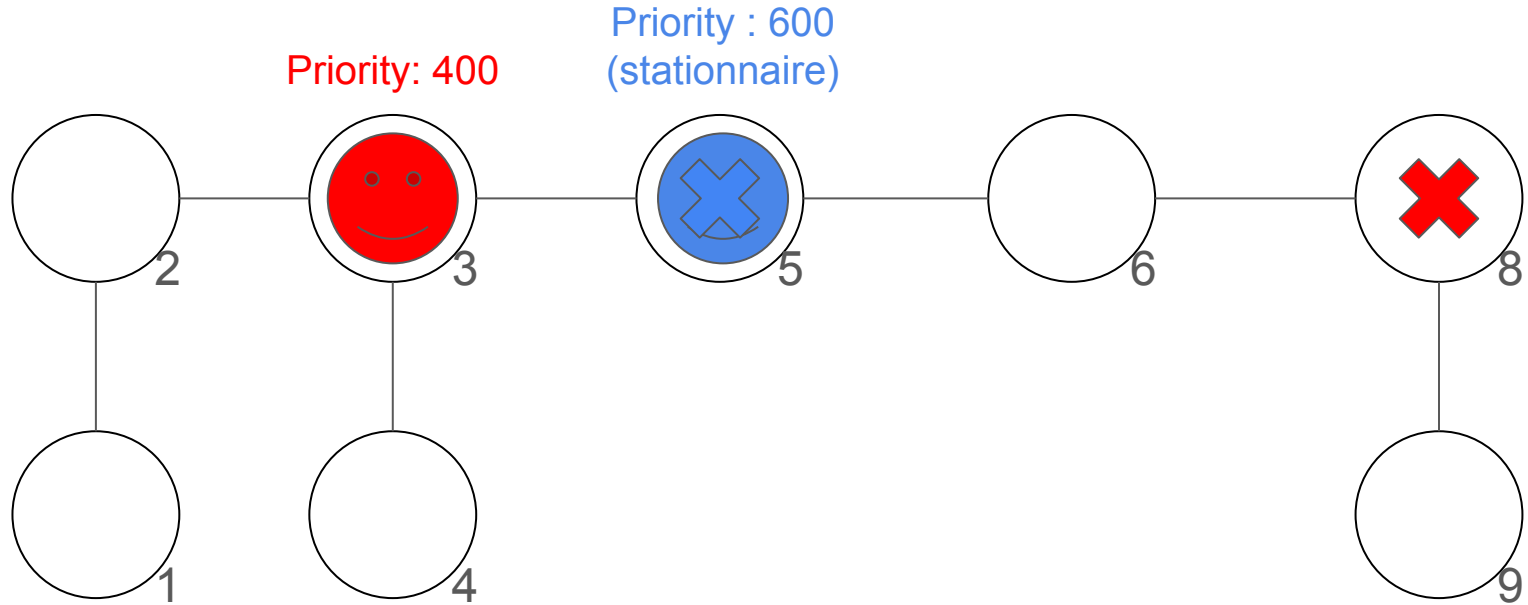


## Interblocage 3 (impasse)

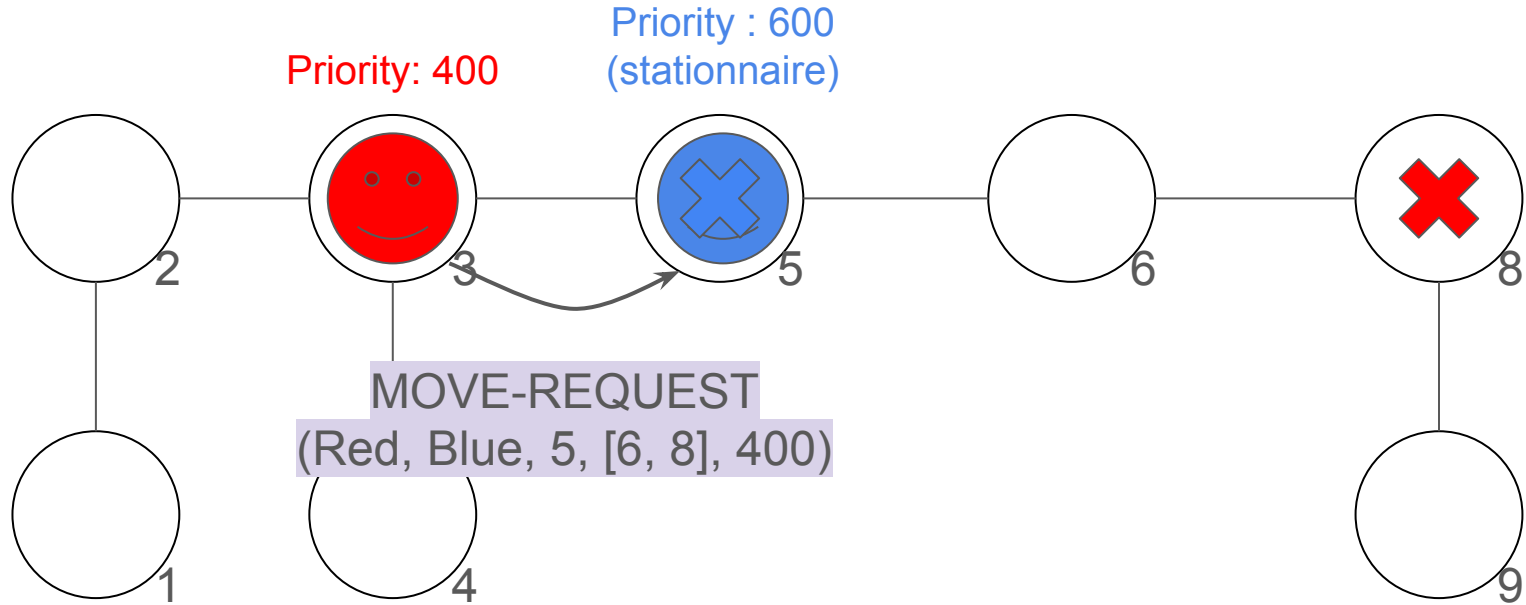




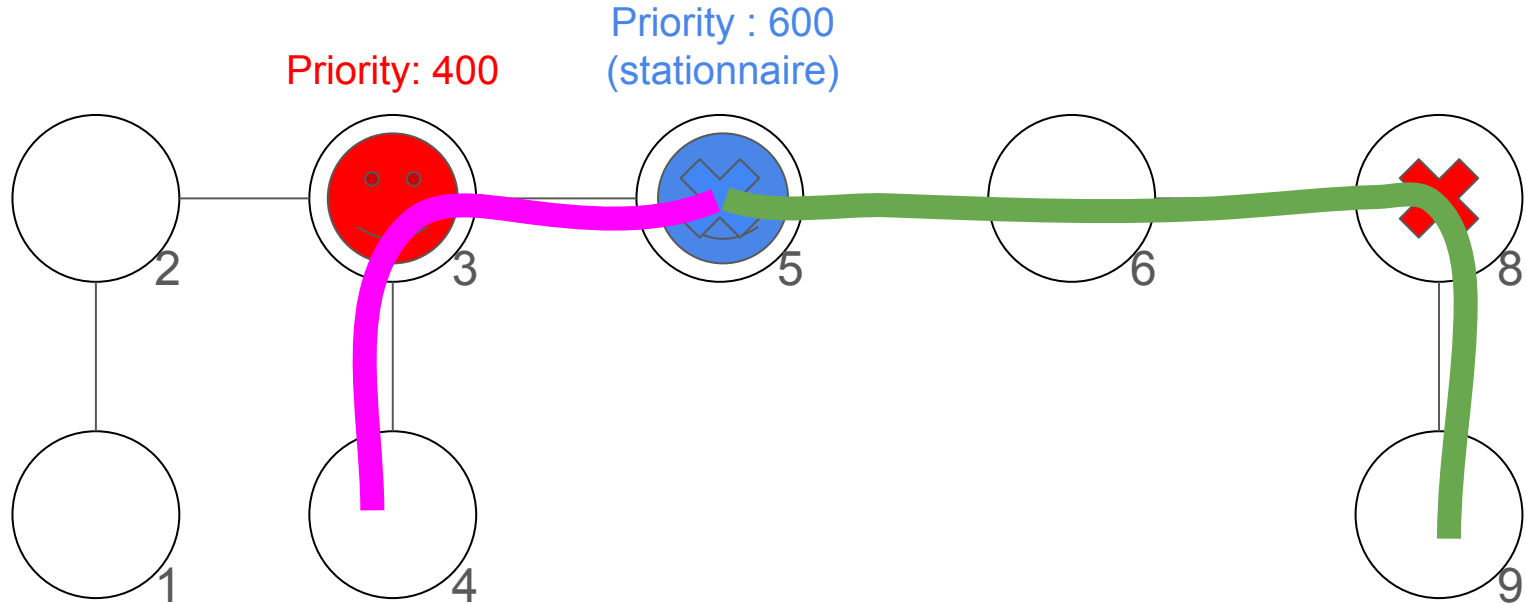
## Interblocage 4 (agent stationnaire)



## Interblocage 4 (agent stationnaire)

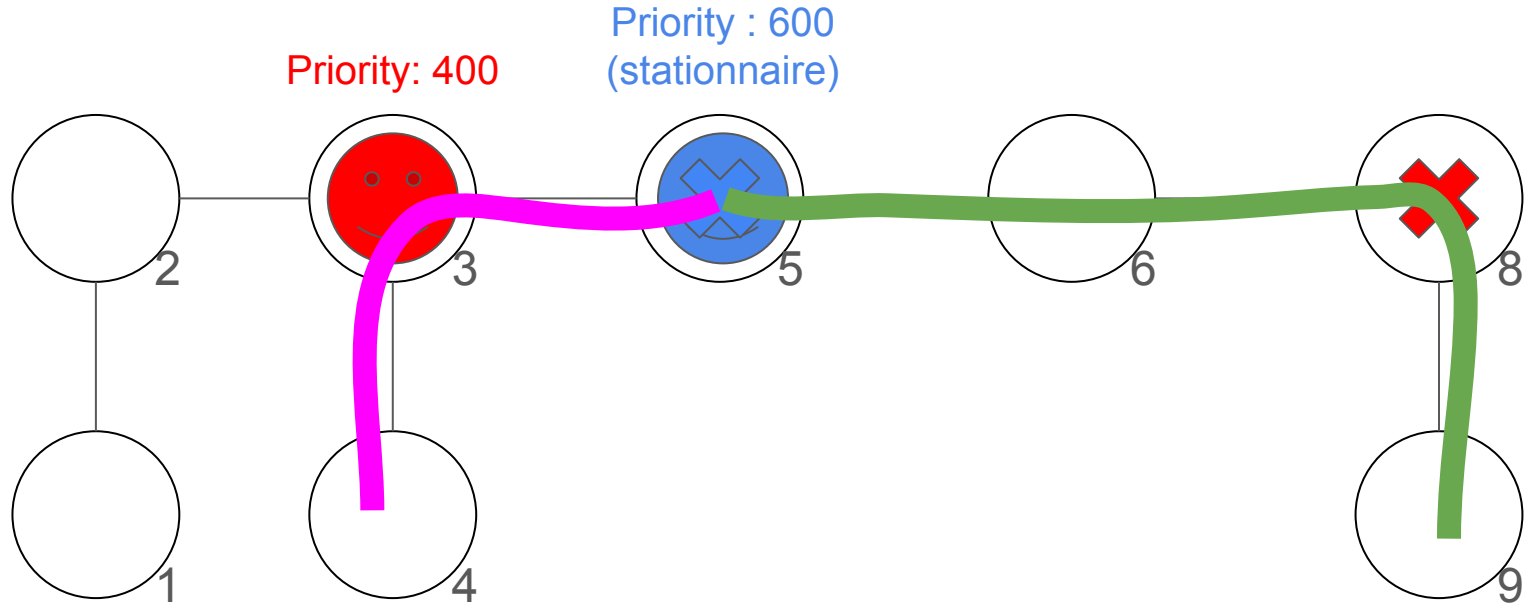


## Interblocage 4 (agent stationnaire)



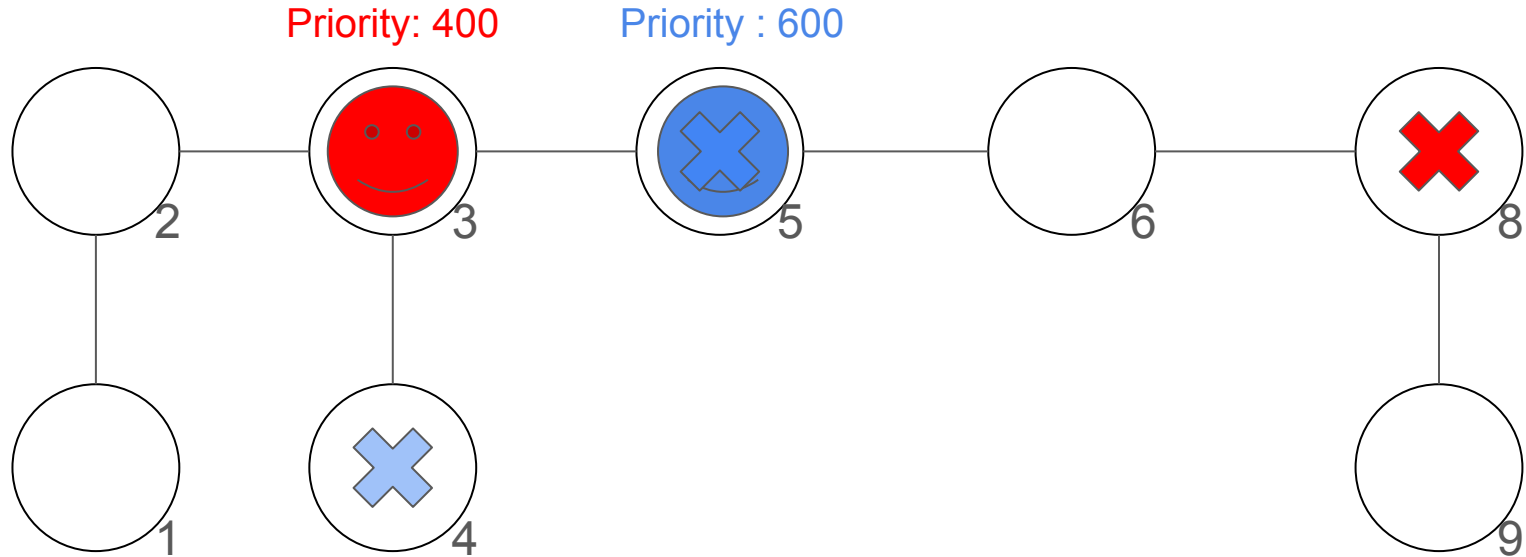
Calcule l'intersection la plus proche du côté de l'envoyeur ainsi que l'intersection la plus proche sans passer vers l'envoyeur

## Interblocage 4 (agent stationnaire)



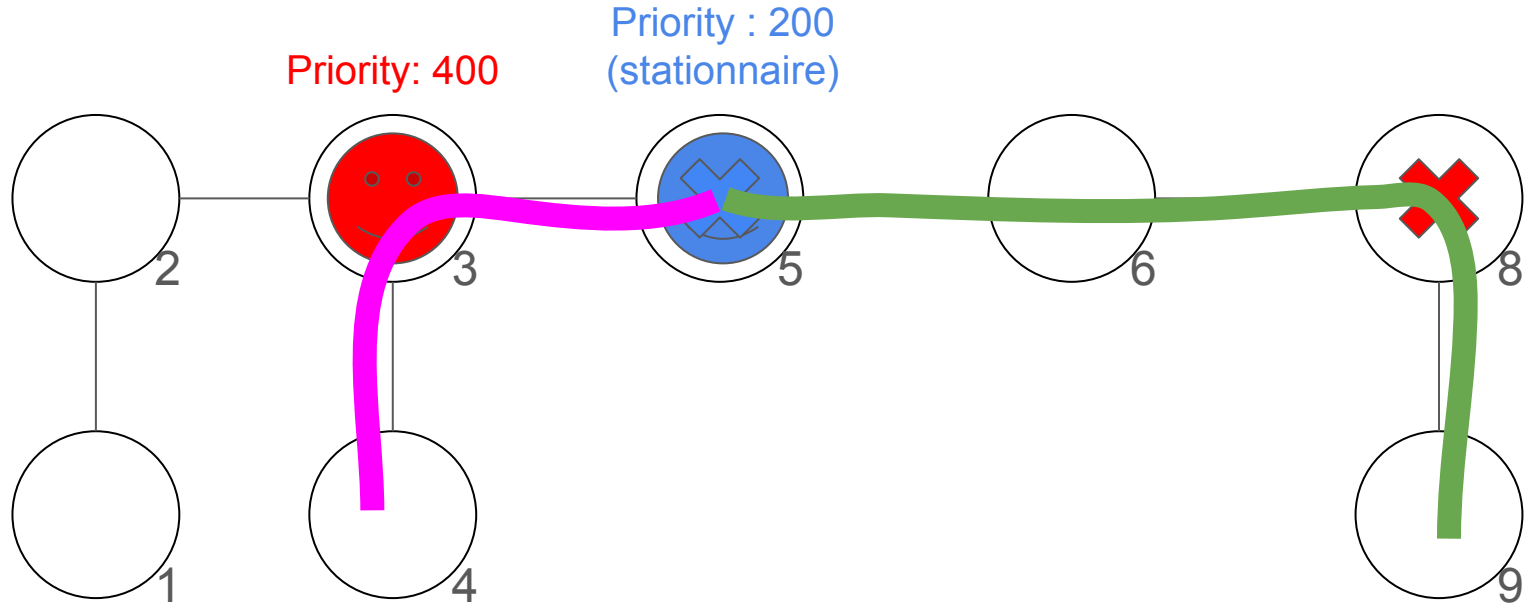
Puis il choisit celle qui est **la plus avantageuse pour lui** si il a la priorité, ou celle **la plus avantageuse pour l'autre** si il n'a pas la priorité

## Interblocage 4 (agent stationnaire)



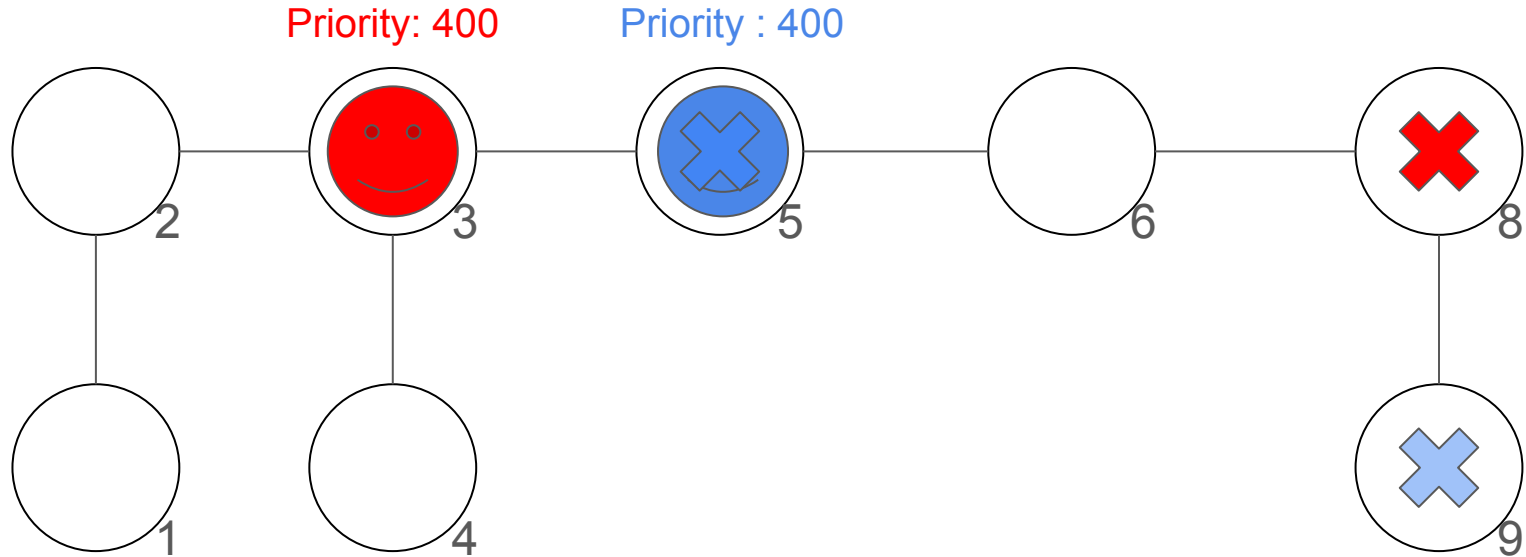
Puis il choisit celle qui est **la plus avantageuse pour lui** si il a la priorité, ou celle **la plus avantageuse pour l'autre** si il n'a pas la priorité

## Interblocage 4 (agent stationnaire) (bis)



Puis il choisit celle qui est **la plus avantageuse pour lui** si il a la priorité, ou celle **la plus avantageuse pour l'autre** si il n'a pas la priorité

## Interblocage 4 (agent stationnaire) (bis)



Puis il choisit celle qui est **la plus avantageuse pour lui** si il a la priorité, ou celle **la plus avantageuse pour l'autre** si il n'a pas la priorité

# Priorités :

0 : pas de but

100 : en communication, ne bouge pas pour rester dans le champ

350 : aller vers un noeud ouvert pendant l'exploitation

400 : aller vers le SILO

600 : aller vers un trésor pour confirmer sa location/contenu

625 : aller vers un trésor pour le collecter

650 : aller vers un trésor pour l'ouvrir

675 : aller vers un trésor pour l'ouvrir et le collecter

800 : pousse toi je suis le SILO et je vais vers ma nouvelle location

900 : pousse toi c'est sans issue (SILO bloque)

925 : pousse toi c'est sans issue (golem bloque)

950 : pousse toi c'est sans issue (autre)



# Terminaison

# Fin de l'exploration (transition)

Quand

- La carte d'un agent n'a plus de noeuds ouverts.

Ou

- Plus de noeuds ouverts accessibles (l'agent ne voit pas de chemin sans obstacles jusqu'au restant des noeuds ouverts).

# Fin de l'exploitation

Quand le Silo calcule qu'il ne reste plus assez de tâches suffisamment importantes (et que les tâches ne sont pas assignées, ou tous les trésors sont vides, ou les agents (Silo exclus) n'ont pas les capacités requises)

-> affichage dans le terminal (on a laissé le code continuer à s'exécuter mais c'est ici qu'on considère l'exécution terminée)

# Terminaison - les limites

Si les agents ne connaissent pas la positions du Silo, ils peuvent prendre du temps

Peut finir prématurément:

- S'il reste des trésors qui n'ont pas été détectés par les agents (par exemple trésors transportés par des golems)
- Pour les trésors avec une petite valeur (ex: 1) après peu d'essais

Peut ne pas finir:

- Si la position des trésors changent constamment (les *attempts* sont reset) où qu'un coffre est impossible à ouvrir

Ralenti

- Peut être lent pour la collaboration: il faut que les agents corrects soient à côté du Silo (finis par se produire mais peut prendre du temps si on n'a pas de chance et qu'un agent est envoyé explorer)

# Améliorations potentielles

Silo : Lorsque plusieurs agents peuvent faire une tâche -> prendre le plus efficace

Silo : utiliser le fait que le champ de communication peut être  $>1$  pour avoir des agents qui attendent/communiquent sans être collés au Silo

Rendre le code plus robuste au temps d'exécutions en utilisant moins/pas des timers pour les différentes parties du code.

Meilleur algorithme de noeud optimal pour le Silo prenant en compte les potentiels routes avec un trafic élevé.

Code changes

# Code changes

Agents “explorer” peuvent maintenant participer aux ouvertures de coffres

Le noeud optimal du Silo évite maintenant les noeud trésors

Silo prend en compte les missions assignées et les capacités des agents avant de dire “Finished”

Changement de timings (Reset task time 5->15, considerSiloLostAfter : 20000->40000)

maxGroupSize: 3->4 pour ouvrir les coffres à 4

# Code changes

Importance des noeuds ouverts réduit plus rapidement ( $20/\text{attempts}+1 \rightarrow 20/\text{attempts}^2+1$ )

Ajout compteur stopExploringFactor pour mission “explore”.  $\text{factor}/100$ ,  $\text{factor} \in [1,90]$  qui augmente  $\rightarrow$  réduit nombre d’explore pour favoriser les agents autour du Silo pour mission collectives au fur et à mesure que l’exécution progresse

Le silo compare sa position optimale à la position courante ET son ancienne position optimale pour regarder si elle est correcte.

Prints plus lisibles



Extra

# Partage complet des cartes

ShareMapBehaviour envoie myMap complet.

## Avantages

- **Simple** : pas besoin de calculer la différence
- **Robuste** : Si on partage tout, dans le cas où un partage échoue, une autre partage arrange le problème.
- **Correction** : Si un agent avait une erreur (ne devrait pas arriver ici), le partage peut le corriger

## Inconvénients

- **Coût message** : Le message est plus gros
- **Lourd** : La fusion d'une grande carte est plus coûteuse

Dans notre cas, on a trouvé la robustesse plus importante.