Introduction aux réseaux automatisés de minecarts

par Procastin

Introduction

Un réseau automatisé de minecarts est un réseau de minecarts qui relie un ensemble de gares et qui n'a pas besoin de l'action d'un joueur pour fonctionner. On s'interdit, par exemple, les intersections où l'on choisit sa direction à l'aide d'un bouton, puisque celles-ci ont besoin de l'action d'un joueur pour diriger le minecart.

L'automatisation des réseaux de minecarts est un domaine qui n'a été que très peu exploré jusqu'à présent. Pourtant, les applications intéressantes sont nombreuses. Pour n'en citer qu'une, un réseau de minecarts automatisé permet de relier des usines automatiques distantes de milliers de blocs à des systèmes de stockage centralisés.

Le but de cet article est de présenter un ensemble de critères suffisants pour créer un réseau de minecarts automatisé. Dans la première partie, nous expliquerons comment acheminer les minecarts à destination tout en évitant les collisions. Dans la seconde partie, nous nous intéresserons à la latence des réseaux automatiques de minecart.

1 Routage et collisions

On s'intéresse au comportement que doivent avoir les intersections pour arriver à un réseau capable de router les minecarts, sans qu'ils entrent en collision.

1.1 La méthode de routage

Une intersection doit être capable de router correctement chaque minecart qui s'y présente. Pour cela, elle doit disposer d'informations sur la destination du minecart, ou au moins sur la direction qu'il doit prendre localement.

Une première manière de faire serait de piloter les intersections de manière globale, à l'aide d'un système de routage commun à tout le réseau qui communique avec les intersections. Mais un tel système serait fortement limité par l'importante latence des communications entre le système de routage et les intersections.

Pour cette raison, un système distribué est préférable à un système centralisé. On va donc piloter les intersections localement, en faisant voyager l'information de destination avec le minecart. Pour ce faire, on encode l'information de destination du minecart par un bloc que l'on transporte dans un second minecart, muni d'un coffre ou d'un entonnoir, qui précèdera toujours notre premier minecart.

Le travail d'une intersection consiste donc, pour chaque convoi arrivant, à lire [1] le bloc du premier minecart pour identifier la destination puis à diriger le convoi vers la bonne sortie. Sans oublier de réinjecter le bloc dans le minecart de tête car les intersections suivantes auront également besoin de l'information. Par abus de langage, on assimilera le convoi de deux minecarts à un unique minecart par la suite.

Remarque Si l'intersection sait pour chaque bloc, c'est à dire pour chaque destination, quelle est la bonne direction, c'est parce qu'elle a été programmée au moment de sa construction par quelqu'un qui connaissait la topologie de l'ensemble du réseau.

1.2 Le problème des collisions

Ce dont nous avons parlé jusqu'à maintenant marche très bien quand le réseau n'a qu'un utilisateur. Mais, dès lors qu'il y a plusieurs minecarts qui circulent sur le réseau, plusieurs minecarts peuvent vouloir aller au même endroit, au même moment, en passant par le même chemin et, fatalement, ils entrent en collision, s'arrêtent, obstruent le réseau et provoquent des collisions en chaîne.

La solution à ce problème consiste à spécifier un invariant sur le réseau : Il doit toujours y avoir au moins une distance de sécurité de τ secondes entre deux convois. Comme τ est strictement positif, aucune collision n'est possible dès lors que l'invariant est respecté partout sur le réseau. On parle de *réseaux sécurisés* pour désigner les réseaux qui respectent cet invariant sans éjecter de minecart du réseau avant arrivée à destination. Quelle valeur donner à τ en pratique ? La plus petite possible telle que le réseau fonctionne, nous y reviendrons.

Mais est ce possible de maintenir un tel invariant sur l'ensemble du réseau ? Nous allons voir que cela dépend de la topologie du réseau. Pour se représenter le réseau globalement, il faut abstraire le fonctionnement des intersections et des gares, on les considère comme de simples boites noires capables d'avaler et de recracher des minecarts. On peut alors assimiler le réseau à un graphe orienté. Les sommets correspondent aux intersections et aux gares. Les arcs correspondent aux rails qui les relient.

Théorème 1.1 *Tout réseau sécurisé est eulérien, c'est à dire qu'il est nécessaire que toute intersection ait au moins autant de sorties que d'entrées.*

Preuve Par l'absurde, supposons que l'on est dans un réseau sécurisé qui n'est pas eulérien, montrons que l'on arrive à une contradiction. Comme ce réseau n'est pas eulérien, il possède une intersection qui est telle que $d^- > d^+$, où d^- et d^+ sont respectivement le nombre d'entrées et de sorties de l'intersection. Pour respecter l'invariant, l'intersection doit envoyer au plus un minecart toutes les τ secondes sur chacune de ses sorties. Elle peut donc évacuer au plus d^+ minecarts par intervalle de τ secondes. Or si elle reçoit, sur chacune de ses entrées, un minecart toutes les τ secondes, elle en aura d^- à évacuer par intervalle de τ secondes, comme $d^- > d^+$ ce n'est pas possible en respectant l'invariant. Si elle ne les évacue pas c'est qu'elle les stocke, mais le flux d'entrée peut continuer indéfiniment et, dans ce cas, le nombre de minecarts stockés ne

1. En utilisant les mêmes techniques que pour le tri automatisé d'items.

va jamais cesser de croître, puisqu'il arrive plus de minecarts qu'il est possible d'en évacuer. Le stockage ayant nécessairement une capacité limitée, il finira par déborder et il devra éjecter des minecarts du réseau. Ce réseau n'est donc pas sécurisé, on arrive bien à la contradiction que l'on cherchait.

Corollaire 1.2 *Tout réseau sécurisé est fortement connexe et chaque intersection du réseau a exactement autant d'entrées que de sorties.*

Preuve Dans un graphe eulérien orienté, chaque sommet a un degré entrant égal à son degré sortant. Ce résultat est déjà connu, une preuve classique est basée sur le principe de Dirichlet.

Corollaire 1.3 *Il existe un modèle pour les intersections tel que pour qu'un réseau constitué uniquement de telles intersections soit sécurisé il faut et il suffit que ce réseau soit eulérien*

Preuve Le *il faut* est donné par le TODO. Pour le *il suffit*, explicitons un modèle pour les intersections tel qu'un réseau eulérien constitué de telles intersections est nécessairement sécurisé. On garde les mêmes significations pour d^- et d^+. Procédons par induction sur la structure du réseau pour définir ce modèle. On interdit aux gares d'envoyer un nouveau minecart sur le réseau tant que la voie ferrée par laquelle il doit s'insérer est déjà utilisée dans l'intervalle de τ secondes courant. Notre hypothèse d'induction est que le réseau est sécurisé jusqu'à une intersection telle que $d^- = d^+$, intersection non comprise. On peut en déduire que le flux entrant de cette intersection n'excède pas d^- minecarts par intervalle de τ secondes. Puisque l'intersection dispose de d^- sorties, il suffit qu'elle assigne, à chacune de ses entrées, une sortie et elle disposera d'une quantité de sorties suffisantes pour ne pas évacuer plus d'un minecart par intervalle de τ secondes et par sortie. L'invariant se propage, un tel réseau est bien sécurisé.

1.3 Exemples de réseaux sécurisés

On se pose maintenant la question de la forme qu'il convient de donner à un réseau sécurisé, étant donné un ensemble de gares à relier. Les principaux critères de choix sont les suivants : longueur totale du réseau, nombre et taille des intersections, distance entre les gares via le réseau.

Un réseau sécurisé *circulaire* est tel que toutes les intersections ont une entrée, une sortie et sont des gares. Si l'on résout le problème du voyageur de commerce pour trouver le plus court réseau circulaire possible, on obtient un réseau de longueur totale minimale. En plus, ce réseau ne contient pas d'intersection complexe. Il est donc très peu coûteux en pratique. Par contre, la distance entre 2 gares, même proches à vol d'oiseau, peut être très importante. La FIGURE 1 est un exemple de réseau sécurisé circulaire.

Si l'on veut que les villes proches puissent toujours être reliées rapidement, le plus simple est de concevoir un réseau qui ne comporte que des tronçons bidirectionnels. C'est à dire un réseau tel que pour chaque voie ferrée qui part d'une première intersection pour aller vers une deuxième, il y a une autre voie ferrée qui part de la deuxième pour aller vers la première, ce sont ces couples de voies qui sont appelés tronçons bidirectionnels. Pour que la longueur totale d'un tel réseau n'explose pas, on aura intérêt à ce qu'il ait une forme *arborescente*, c'est à dire

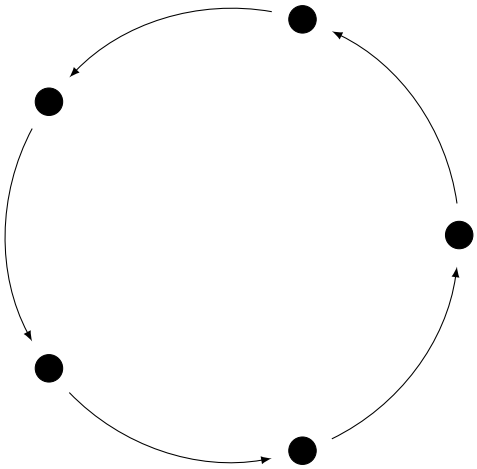

FIGURE 1 – Un réseau sécurisé circulaire comprenant 5 gares

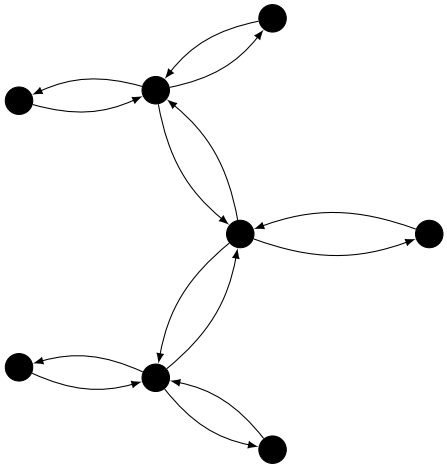

FIGURE 2 – Un réseau sécurisé arborescent comprenant 5 gares et 3 intersections supplémentaires

à ce que les tronçons bidirectionnels forment un arbre. Pour trouver le plus court de ces réseaux, on pourra résoudre le problème de l'arbre de Steiner rectilinéaire. On pourra aussi préférer un réseau arborescent dont la longueur totale est plus importante mais qui permet des communications plus rapides. La FIGURE 2 est un exemple de réseau sécurisé arborescent.

Bien évidemment, au delà de ces 2 exemples, on peut imaginer de nombreuses autres topologies intéressantes pour un réseau sécurisé.

2 Accessibilité et latence

On considère à présent un réseau sécurisé composé uniquement d'intersections qui ont autant d'entrées que de sorties et qui, à chaque intervalle de τ secondes, les mettent en bijection pour gérer le routage. On doit maintenant s'assurer que la manière dont on évite les collisions n'empêche pas les minecarts d'arriver à destination et, si possible, d'y arriver rapidement.

2.1 Le problème des conflits

Simultanément, à chaque intervalle de τ secondes, chaque minecart souhaite être dirigé dans une direction en particulier, déterminée par sa destination. D'ores et déjà, on peut observer que ce ne sera pas possible pour tous les minecarts entrants dès lors que plusieurs minecarts veulent être redirigés vers une même sortie. Autrement dit, pour chaque intersection et à chaque intervalle de τ secondes, un couplage parfait n'existe pas nécessairement entre entrées et sorties. On cherche malgré tout un couplage maximum :

— Pour chaque sortie source de conflit, l'un des minecarts en conflit est assigné.
— Les minecarts restants sont assignés aux sorties vers lesquelles aucun minecart ne voulait être redirigé.

Le temps que met un minecart à aller d'un point à un autre du réseau ne peut donc être borné indépendamment du trafic, il dépend du nombre

de déviations qu'aura emprunté le minecart par rapport à sa trajectoire optimale.

2.2 Conflits et accessibilité

Dans l'hypothèse où le réseau est surchargé, n'est il pas possible qu'un minecart n'allant pas toujours où il veut soit indéfiniment prisonnier d'un cycle du réseau ? On doit s'assurer que tous les minecarts vont bien arriver à destination en temps fini. Un *réseau valide* est un réseau sécurisé tel que tous les minecarts présents sur le réseau arriveront en temps fini à leur destination. Comme le réseau est fortement connexe, le minecart ne peut pas se retrouver à une intersection depuis laquelle il n'y a aucun chemin vers sa destination. C'est donc uniquement un problème de résolution des conflits, c'est à dire un problème de sélection, pour chaque intersection et pour chaque intervalle de temps τ, du couplage maximum parmi tous les couplages maximums possibles.

Une première idée pour aboutir à un réseau valide serait de mettre en place un système de priorités statiques, de type *priorité à droite*, pour arbitrer les conflits. En fait, ça ne peut pas fonctionner : Supposons qu'il arrive à une intersection deux flux continus de minecarts par deux entrées et que tous ces minecarts souhaitent emprunter la même sortie, passage obligé pour leurs destination. Avec un système de priorité statique, le même flux sera toujours prioritaire et, puisqu'il est continu, il monopolisera systématiquement la sortie de l'intersection. Les minecarts arrivant de l'autre flux ne pourront donc jamais passer et n'arriveront jamais à leur destination.

Il faut donc que les priorités soient dynamiques. Si on fait un système de type *feux de circulation*, où le couplage maximum prioritaire change périodiquement, rien ne garantit qu'un minecart effectuant un cycle infini dans le réseau ne soit pas en phase avec le feu et arrive systématiquement au mauvais moment. On pourrait associer une priorité à chaque minecart, la stocker dans son minecart d'informations et la faire augmenter quand le minecart n'est pas choisi à une intersection afin qu'il finisse toujours par être sélectionné. L'ennui c'est que l'information transportée doit être minimale pour permettre un traitement rapide. Au final, la meilleure solution est encore l'aléatoire.

Théorème 2.1 *Pour qu'un réseau sécurisé soit valide en probabilités il suffit que pour toute intersection du réseau, pour tout conflit possible à cette intersection et pour tout couplage maximum solutionnant le conflit, on ait une probabilité $p > 0$ que le couplage soit choisi.*

Preuve On peut voir le graphe du réseau comme une chaîne de Markov, l'état courant est la position du minecart. Comme le graphe est fortement connexe, il existe un chemin de l'état courant vers la destination du minecart. Ce chemin est bien de probabilité non nulle puisque, pour chaque intersection du parcours, la probabilité d'aller vers le sommet suivant du chemin est $p > 0$ s'il y a un conflit, 1 sinon. Ainsi, la destination est un état accessible et récurrent de la chaîne de Markov. Le minecart arrive donc à destination en temps fini avec probabilité 1.

2.3 Latence

On sait que si on attend assez longtemps notre minecart arrivera à destination mais on est en droit de se demander si le temps à attendre est raisonnable, étant donné la topologie du réseau.

En fait, il ne faut pas perdre de vue que la surcharge du réseau est un évènement exceptionnel. Le plus souvent il n'y a aucun de conflit et le minecart arrive à destination par le chemin le plus court. Même dans un réseau très fréquenté, on peut n'avoir en moyenne $O(1)$ conflits pendant le parcours de n intersections et le temps total du parcours sera asymptotiquement du même ordre de grandeur que l'optimal. Si ça ne suffit pas, on peut encore réduire l'écart en moyenne avec l'optimal en ajoutant des boucles sur elles même aux intersections, c'est à dire des stockages tampon, et en les associant en priorités aux minecarts ne pouvant être assignés à la sortie souhaitée.

Les choses se compliquent quand le réseau est tellement fréquenté que les intersections sont presque toutes saturées en permanence, on a alors $O(n)$ conflits pendant le parcours de n intersections. La latence du réseau augmente alors fortement, le temps de voyage moyen dépend de la topologie du réseau et des probabilités de résolution des conflits. Dans le cas de probabilités uniformes, si M minecarts veulent franchir une même intersection dans un réseau saturé, on peut être certain que le temps moyen pour la franchir sera au moins multiplié par M.

Remarque Si la latence est plus critique sur certains axes du réseau que sur d'autres, les probabilités uniformes ne sont probablement pas le meilleur choix.

2.4 Limites d'un réseau

Quand un réseau est saturé il n'y a rien à faire sinon attendre qu'il ne le soit plus. Souvent, ce n'est pas très grave tant que ça n'arrive que ponctuellement. Par contre, si c'est toujours le cas parce que des minecarts en grand nombre sont continuellement injectés, on a atteint les limites du réseau. Quelles sont alors les solutions ?

Une première solution consiste à repérer les goulots d'étranglements et à ajouter des voies. Une manière simple de garder un réseau sécurisé est de les ajouter par 2, une dans chaque sens. On peut alors passer à un formalisme avec des poids entiers sur les arcs, ça ne change pas grand chose. Une deuxième solution est d'augmenter la fréquence de fonctionnement du réseau, c'est à dire de diminuer τ en changeant le modèle des intersections. Il faut alors reconstruire toutes les intersections, il vaut donc mieux anticiper le volume du trafic d'un réseau dès le moment de sa conception.

2.5 Exemples d'intersections

Quel modèle choisir pour ses intersections ? Plus l'on veut une fréquence élevée, c'est à dire τ petit, plus il sera difficile et coûteux d'implémenter les intersections. En effet, ces dernières vont devoir être capable de lire les blocs et de les réinjecter, de traduire en terme de destinations, de choisir un couplage maximum, d'acheminer chaque minecart vers la sortie correspondante et ce en au plus τ secondes. En particulier, cette fréquence dépendra légèrement du nombre de gares du réseau, puisque le système de lecture sera d'autant plus lent qu'il devra pouvoir distinguer de blocs.

Conclusion

Il est permis d'espérer que cette contribution théorique débouchera sur de véritables réseaux fonctionnels dans minecraft. Au delà des problèmes techniques de chargement des zones du

jeu où se trouvent le réseau, le problème majeur restant est celui de la tolérance aux fautes des utilisateurs : Un joueur qui quitte son minecart en plein milieu des voies peut casser le fonctionnement de tout le réseau.

Table des matières