Guillaume DAVY – Cédric FOUCAULT – Guilhem GAMARD – Romain VERNOUX

Projet de programmation réseau
Spécification du protocole

Version 1.0 du 9 novembre 2012

Table des matières

1 Introduction

Le présent document décrit le protocole de jeu Bomberman en réseau implémenté par Romain VERNOUX, Guilhem GAMARD, Cédric FOUCAULT et Guillaume DAVY dans le cadre de l'UE PROJET DE PROGRAMMATION RÉSEAU (MPRI 1–21). Ce protocole a été prévu afin d'être aisément extensible, ce qui permettra l'ajout de fonctionnalités supplémentaires si le temps le permet.

La version du protocole présentée dans ce document est la version 1.

Historique

– Version 1.0 : spécification initiale.

2 Déroulement d'une partie

La présente section décrit l'aspect général et les règles du jeu implémenté, comme s'il était exécuté sur une seule machine (par exemple en écran partagé). La communication sur le réseau sera abordée dans les sections 3 et suivantes.

Chaque partie se déroule dans un espace discrétisé, l'aire de jeu étant une matrice de « cases ». Chaque case peut être un et un seul des éléments suivants :

– un bloc destructible ;
– un bloc indestructible ;
– une case libre (aussi appelée case vide).

Une partie comprend également un certain nombre de joueurs et de bombes.

Chaque joueur est caractérisé par une case courante, qui désigne sa position dans l'aire de jeu. Chaque bombe est caractérisée par une case courante et un compte à rebours (entier naturel).

Une *configuration*, aussi appelée *état courant*, est la donnée d'une aire de jeu finie et d'un nombre fini de joueurs et de bombes. La détermination de la configuration initiale d'une partie est documentée en section 5.

Le temps est également discrétisé, le jeu se déroulant en tours successifs. La durée physique d'un tour est un paramètre déterminé à l'avance avec la configuration initiale et est fixée pour toute la partie.

Un tour se déroule comme suit : chaque joueur choisit une action qu'il effectuera pour ce tour, puis la configuration de la partie est mise à jour en fonction de ces actions.

Chaque mise à jour se déroule en deux étapes : effets des actions des joueurs, puis mise à jour des bombes.

Actions des joueurs. Au début d'un tour de jeu, chaque joueur peut choisir d'effectuer une seule des actions suivantes :

- se déplacer dans une direction parmi droite, gauche, haut, bas ;
- poser une bombe ;
- ne rien faire.

Les actions des différents joueurs pour un même tour sont réputées avoir eu lieu en même temps. Chaque action modifie la configuration du jeu comme décrit ci-après.

Pour un joueur dont la case courante est (x, y), l'action de se déplacer à droite (resp. gauche, haut, bas) a pour effet de changer sa position vers la case $(x + 1, y)$ (resp. $(x - 1, y)$, $(x, y - 1)$ et $(x, y + 1)$) si cette dernière est une case libre. Si la case de destination est hors de l'aire de jeu ou n'est pas une case libre, l'action est sans effet.

L'action de poser une bombe ajoute une bombe à la partie, dont la position courante est la position courante du joueur l'ayant posée, et dont le compte à rebours est initialisé à 12. Si la case n'était pas initialement une case libre, l'action est sans effet.

L'action de ne rien faire est sans effet.

Les actions des joueurs prennent effet dans un ordre arbitraire, selon l'implémentation. Une fois les effets des actions de chaque joueur pris en compte, la phase de mise à jour des bombes est lancée.

Mise à jour des bombes. Lors de la phase de mise à jour des bombes, le compte à rebours de chaque bombe est décrémenté.

Ensuite, tant que la configuration comporte au moins une bombe b dont le compte à rebours est égal à zéro, cette bombe explose. Une explosion se déroule comme suit : pour chaque case c sur la même ligne ou la même colonne que b et à une distance inférieure ou égale à 5,

- si c est un bloc destructible, c devient libre ;
- si c est la position courante d'une bombe, le compte à rebours de cette bombe est mis à 0 ;
- si c est la position courante d'un joueur, ce joueur est éliminé de la partie.

La bombe b est ensuite retirée de la partie.

Il est à noter que ce comportement entraîne des réactions en chaîne : l'explosion d'une bombe peut entraîner une autre explosion. Toutes les explosions incidentes sont prises en compte pendant le même tour que l'explosion initiale.

En outre, lorsqu'une bombe est posée, son compte à rebours est décrémenté une première fois pendant le même tour. Enfin, une bombe dont le compte à rebours est à 1 au début d'un tour explosera lors de ce tour, puisque la décrémentation des compteurs a lieu avant la prise en compte des explosions.

Ces détails ont peu d'importance pour le rendu du jeu final (la durée d'un tour étant très courte), mais seront cruciaux pour de garantir la bonne synchronisation des différents joueurs dans la section 6.

Si, à la fin d'un tour, tous les joueurs ont été éliminés sauf un, ce dernier est le gagnant. Si tous les joueurs ont été éliminés, la partie est nulle.

3 Conventions et architecture générale

Dans la suite de ce document, le mot *entier* (nombre entier) désignera une suite de quatre octets représentant un entier non signé codé en *little endian*. En outre, sauf indication contraire, un octet isolé doit s'interpréter comme non signé.

Le protocole est architecturé selon un modèle *client-serveur*. Le protocole réseau utilisé est TCP au-dessus d'IPv4. Le port habituellement utilisé est le $42\,042$. Chaque joueur est un client.

Dans l'architecture proposée, un client est amené à communiquer avec deux types de serveur :
- le serveur *lobby* ;
- le serveur *partie*.

Le serveur *lobby* fait office de bureau d'accueil : il propose une liste de parties en attente de joueurs. Un client peut alors choisir de rejoindre une partie existante en se connectant au serveur *partie* correspondant ou d'en créer une nouvelle. Les règles de communication entre les clients et le serveur *lobby* sont données en section 4.

Chaque partie de jeu est gérée par un serveur *partie* dédié (les différents serveurs pouvant être sur un même serveur physique), ainsi, une partie donnée est supportée par un canal de communication réservé. La connexion entre un client et un serveur partie met en jeu deux protocoles de communication : le protocole avant le début du jeu (exposé en section 5) et le protocole en cours de jeu (détaillé en section 6).

Le premier protocole spécifie le comportement du serveur tant qu'une partie est en attente de joueurs : celui-ci informe les clients du nombre de joueurs qui ont rejoint la partie. Lorsqu'une partie est complète, le serveur déclenche le début du jeu en envoyant aux clients les informations d'initialisation requises. Puis, le second protocole entre en jeu. Le serveur sert alors de relais pour les actions des joueurs. Chaque client envoie au serveur les actions effectuées par son joueur, et reçoit en retour les actions

effectuées par l'ensemble des joueurs. La mise à jour de l'état courant et l'affichage sont dévolus aux clients.

Au cours de toutes ces phases, la communication s'effectue à l'aide de *paquets*. Un paquet désigne ici l'unité sémantique de base du protocole, et non un paquet au sens de TCP/IP. Chaque paquet est constitué comme suit :

- un entier codant la longueur du paquet (cet entier exclu) ;
- un octet codant le *type* du paquet ;
- des données de taille variable, selon le type du paquet.

Ainsi les données, requêtes et réponses transitent entre clients et serveur, encodées sous forme de paquets. Le serveur et les clients envoient et reçoivent des paquets en boucle ; ces paquets sont interprétés en fonction de leurs types suivant la présente spécification.

Dans la suite de ce document, lorsqu'un paquet de type n sera décrit, le fait que le paquet débute par un entier codant sa longueur et un octet codant son type sera sous-entendu. On ne décrira que la structure des données de taille variable.

Il est possible qu'un paquet reçu soit plus long que la description qui en est faite ici. Dans un tel cas, tous les octets en excédent doivent être ignorés. Symétriquement, chaque implémentation est libre de placer des données supplémentaires à la fin d'un paquet, pourvu que le champ *longueur* du paquet les prenne en compte.

Ces données supplémentaires peuvent être utilisées pour le débogage ou pour implémenter des extensions spécifiques à une implémentation. Il conviendra de s'assurer que tout client/serveur gérant une extension devra se conformer au protocole décrit dans le présent document si il communique avec un serveur/client respectant ce protocole à la lettre (ne produisant pas de paquets avec des données supplémentaires).

4 Protocole du lobby

En premier lieu, les clients se connectent au serveur d'accueil, connaissant son adresse IP et le port à utiliser. Ce serveur envoie régulièrement son état à tous les clients connectés, c'est-à-dire la liste des parties en attente et le nombre de joueurs actuellement connectés à chacune d'entre elles.

Pour cela, il utilise un paquet de type 1 contenant les données suivantes :

- un entier codant le nombre de parties en attente ;
- une concaténation des informations correspondant à chaque partie :
  - un entier d'identifiant de partie
  - quatre octets codant l'IP du serveur de jeu
  - deux octets codant le port sur lequel se connecter
  - un entier codant le nombre de joueurs connectés à la partie
  - un entier codant le nombre maximum de joueurs de la partie

Les octets des adresses IP sont codés dans l'ordre de lecture. Ainsi, le codage de l'adresse $192.168.1.2$ commencera par un octet de valeur 192 et finira par un octet de valeur 2. Le port, codé sur deux octets, est en little endian.

Le client peut alors rejoindre directement une partie en se connectant [1] à l'adresse et au port correspondant, attendre, ou créer une nouvelle partie. Dans ce dernier cas, le client envoie un paquet de type 15, sans aucune donnée, au serveur. Celui-ci crée alors une nouvelle partie par le moyen de son choix, mais n'est pas tenu de retourner des informations au client.

5 Protocole avant le début du jeu

Dès qu'au moins un client est connecté à un serveur de jeu, ce dernier envoie régulièrement un résumé de sa situation à tous les clients connectés. Il utilise pour cela un paquet de type 21 avec les données suivantes :

- un entier codant le nombre de joueurs connectés ;
- un entier codant le nombre maximum de joueurs de la partie.

Dès que le nombre maximal de joueurs est atteint, la partie commence. Le serveur envoie alors à chaque client un paquet d'initialisation personnalisé, de type 32, contenant les données suivantes :

- un entier codant le numéro du joueur dans la partie ;
- un entier codant le nombre total k de joueurs ;
- un entier codant la durée d'un tour (en millisecondes) ;
- un entier codant la largeur n de la carte ;
- un entier codant la hauteur m de la carte ;
- une concaténation de $n \times m$ descriptions de cases ;
- une concaténation de k paires d'entiers, codant les positions initiales des joueurs.

L'état initial est entièrement déterminé par le serveur avant d'être envoyé aux clients. Le serveur garantit qu'il envoie le même paquet d'initialisation à tous les clients, excepté pour le numéro du joueur.

État initial de l'aire de jeu. Les $n \times m$ descriptions de cases codent l'état initial de l'aire de jeu. Chaque description de case est un octet codant le contenu d'une case suivant le tableau ci-après.

Valeur de l'octet	Contenu de la case
00	Case libre
01	Bloc destructible
02	Bloc indestructible
03	Bombe

Une case « bombe » (03) désigne une case libre se trouvant être la position d'une bombe. Ainsi, le jeu démarre avec autant de bombes que de cases 03 dans la description de l'aire de jeu initiale, dont les positions sont exactement ces cases 03, et dont les comptes à rebours sont initialisés à 12. Une aire de jeu peut ne contenir initialement aucune bombe.

1. TCP/IPv4

Outre fait qu'elles sont les positions de départ des bombes, les cases 03 sont des cases libres ordinaires, elles n'ont aucune utilité particulière en cours de partie.

Les descriptions de cases sont données de gauche à droite et de haut en bas. Ainsi, la première case représente le coin supérieur gauche de l'aire de jeu (coordonnées $(0, 0)$), la $n-$ième le coin supérieur droit $(n, 0)$, et la $n \times m-$ième le coin inférieur droit (n, m).

Positions initiales des joueurs Le dernier champ décrit de ce paquet, une concaténation de paires d'entiers, décrit les positions initiales des joueurs. Une paire d'entiers est une concaténation de deux entiers. Pour chaque joueur, son abscisse (colonne) initiale est donnée par le premier entier de la paire, et son ordonnée (ligne) initiale par le second.

Le serveur garantit que les positions initiales des joueurs sont mutuellement distinctes, qu'elles sont distinctes des positions initiales des bombes, et qu'elles sont toutes des cases libres.

Les positions de départ sont données dans l'ordre des identifiants de joueur : la première paire donne la position initiale du joueur 1, la deuxième la position du joueur 2, ..., la $k-$ième la position du joueur k.

Le serveur garantit que les identifiants de joueur sont exactement les entiers de 1 à k. Ainsi, l'encodage des positions initiales des joueurs a toujours du sens.

6 Protocole en cours de partie

À partir de ce point, le jeu suit la sémantique décrite en section 2. Comme mentionné en section 3, le serveur ne sert maintenant que de relai pour les actions des joueurs, l'ensemble de la mise à jour de l'état de la partie est dévolu aux clients. Ces derniers doivent donc suivre précisément les règles données précédemment, afin de garantir que chaque joueur soit dans la même configuration que les autres.

Encodage des actions. Chaque action sera encodée sur un octet, de la façon suivante.

Valeur de l'octet	Action correspondante
00	Erreur
01	Joueur éliminé
16	Ne rien faire
17	Déplacement à droite
18	Déplacement en haut
19	Déplacement à gauche
20	Déplacement en bas
32	Poser une bombe

Déroulement d'un tour. La partie se déroule comme une succession de tours.

Du point de vue du client, un tour se déroule comme suit :
- collecter l'entrée du joueur (clavier, manette, IA...) ;
- envoyer l'action correspondante (paquet de type 42) ;
- recevoir les actions des joueurs (paquet-réponse de type 42) ;
- mettre à jour la configuration en fonction de la réponse ;
- tour suivant (boucler).

En particulier, le client doit se fier au serveur y compris pour sa propre action au tour courant.

Du point de vue du serveur, le tour se déroule comme suit :
- collecter les entrées des joueurs, soit pour chaque paquet client de type 42 reçu :
  - si le numéro du tour du paquet est égal au numéro du tour courant et que le serveur n'a pas encore associé d'action à ce client, enregistrer cette entrée
  - sinon, ignorer le paquet ;
- attendre que la durée du tour soit écoulée ;
- envoyer les entrées à chaque joueur (paquet-réponse de type 42) ;
- tour suivant (boucler).

Si la durée d'un tour est écoulée avant que tous les joueurs n'aient envoyé une action, les joueurs n'ayant rien envoyé sont réputés ne rien faire (code 16).

Encodage des paquets de type 42. Les paquets de type 42 sont utilisés pour spécifier les actions des joueurs. Dans un sens, le client spécifie son action pour le tour courant à l'aide d'un paquet 42. Dans l'autre, le serveur transmet les actions de chaque joueur à chaque client *via* un paquet de type 42.

Lorsque le client envoie un paquet de type 42 pour spécifier son action pour le tour courant, les données du paquet sont organisées comme suit :
- un entier codant le numéro du tour courant ;
- un octet codant l'action effectuée par le joueur.

Lorsque le serveur envoie un paquet-réponse de type 42 pour spécifier les actions de chaque joueur pour le tour courant, les données du paquet sont organisées comme suit :
- un entier codant le numéro du tour courant ;
- un entier codant le nombre de joueurs de la partie ;
- pour chaque joueur, dans l'ordre des identifiants de joueur, un octet codant son action.

Le nombre de joueurs dans la partie est constant, il s'agit du nombre de joueurs au début de la partie. En d'autres termes, les joueurs éliminés sont comptés tout de même. Lorsqu'un joueur est éliminé de la partie (par l'explosion d'une bombe), l'octet codant son action peut avoir n'importe quelle valeur. Les actions des joueurs éliminés doivent être ignorées, mais elles sont systématiquement envoyées pour garder la même structure de paquet tout au long de la partie.

En outre, le numéro du tour est défini comme zéro au premier tour, puis incrémenté après chaque tour.