L'Arbre à Palabres

Projet de Programmation Réseau - 2013
Licence Informatique
Université Paris Diderot

Giulio Guerrieri, **Arnaud Sangnier**, Jean-Baptiste Yunès

18 mars 2013

Attention : Prenez le temps de lire attentivement TOUT le document et ce, dans ces moindres détails. Relisez plusieurs fois, même et surtout en groupe. Ne vous lancez pas immédiatement dans la réalisation à moins que vous ne soyez déjà à l'aise avec le développement réseau et la communication interapplications. Faites donc des diagrammes, imaginez des scénarios de communication entre entités, essayez d'en analyser la portée, les problèmes, les informations dont les entités ont besoin au cours de leur vie, etc. Essayez d'abord de vous abstraire des problèmes techniques.

Note : Dans le document le signe ␣ représentera un simple caractère d'espacement (ASCII 32).

Introduction

Le but de ce projet est de programmer différentes entités qui en s'exécutant réalisent un service s'apparentant à des **salons de discussion**.

Le système comportera au moins trois entités : des **villes**, des **cafés** et des **clients**.

Un **client** pourra interroger une ville afin d'en connaître la liste des cafés qui y sont ouverts (consultation de l'annuaire des bistrots). Une fois son choix opéré, le client pourra entrer dans un café, éventuellement saluer (« Bonjour tout le monde », « Salut les aminches ») pour y discuter avec les autres clients. Une discussion ordinaire consiste, comme au comptoir d'un bistrot, à émettre des messages que tout le monde recevra (« Hé, vous avez vu la téloche hier soir, c'était ringard leur truc, non ? », « Qu'est ce qu'il fait moche en ce moment », « Z'avez lu le sujet du projet de programmation réseau ? »). Il sera aussi possible de converser de façon privée avec un client en particulier lorsqu'on est dans un bistrot (« Dis-moi Gonzague, t'as pas 10 balles à me prêter ? », « Rachid, t'as pas vu Robert ? »).

Les **cafés** doivent, comme tout établissement de ce type, s'enregistrer officiellement auprès d'une ville (le débit de bêtises est soumis à règlementation). Un café s'engage aussi à retransmettre divers messages informatifs d'ordre général en provenance de la municipalité (la ville) et qui sont destinés à tous les clients. Les messages informatifs sont normalement archivés par le café et retransmis à tout client qui entre dans le café.

Les **villes** sont organisées en réseau. Une nouvelle ville sera normalement branchée à au moins une ville existante, sauf la première ! Chaque ville devra vérifier régulièrement que les cafés

qui s'y sont déclarés sont toujours en activité car elle n'acceptera qu'un nombre limité de cafés à un instant donné (la surface d'une ville est limitée, et elles ne souhaitent pas non plus trop promouvoir le débit de liquides). La ville pourra, de plus, recevoir des messages à caractère informatif d'ordre général et les retransmettre à des cafés qui les serviront à leur tour à leurs clients. Ces messages informatifs pourront être à diffuser uniquement aux cafés de la ville concernée (c'est la « fête des voisins ») ou à tous les cafés de toutes les villes atteignables à partir de la ville d'origine du message (c'est une « grande brocante régionale »).

Modes conversationnels

Les conversations entre clients (conversations privées) s'effectueront en mode connecté (TCP).

Les conversations entre client et café s'effectueront toujours en mode paquet (UDP).

Les conversations entre villes s'effectueront en mode connecté (TCP).

Les conversations entre ville et cafés s'effectueront elles en mode paquet (UDP).

Les conversations entre client et ville s'effectuera en mode connecté (TCP).

On notera qu'à priori les cafés ne se parlent pas entre eux (concurrence oblige), mais que rien n'empêche un café « pirate » de se faire passer pour un client et diffuser de fausses informations « C'est nul ici, y'a pas d'ambiance, le café est trop cher »).

Description détaillée

Un client

Tout client possède un attribut *nom*.

Un client qui souhaite passer un peu de temps dans un café doit auparavant interroger une ville (identifiée par un numéro ip et un numéro de port TCP) pour :

- en obtenir la liste des cafés ouverts. Le message aura la forme `SHOPLIST`, ce à quoi la ville répondra par une liste de noms de cafés : `200␣SHOPLIST␣`n`␣`$nomcafe_1$`,`$nomcafe_2$`,`$\cdots$ $nomcafe_n$
- suite à quoi (ou alors si le client non standard – un habitué ? – connaît déjà le nom d'un café dans lequel il souhaite se rendre) il en obtiendra les caractéristiques pour s'y rendre en envoyant le message `SHOPINFO␣`*nomcafe*, la ville répondra alors :
  * `200␣SHOPINFO␣`*nomcafe*`,`*ip*`,`*port* dans le cas où le café existe bien,
  * `403␣SHOPINFO␣not␣found` pour indiquer que le café en question n'existe pas ou plus.

  À la suite de quoi, la conversation sera terminée.

Une fois que le client a pu récupérer les données associées à un café, il peut alors s'y rendre. Aucune procédure d'enregistrement particulière n'est requise pour entrer dans un café, il suffit simplement de rejoindre le groupe de diffusion adéquat (numéro normalement renvoyé par la ville à la suite d'un `SHOPINFO␣`*nom*) puis d'écouter ou d'émettre sur le port adéquat. Les actions possibles sont :

- multidiffusion d'un message de salutation `HELLO␣`*nomclient*`,`*ip*`,`*port message*. Normalement, un client ne devrait pas envoyer plus d'une fois ce message. À la suite de ce message, le café peut lui renvoyer en UDP une rafale de paquets contenant divers messages informatifs

d'ordre général qui auront tous la forme `100␣NEWS␣`*date*`␣`*message*, après un certain délai (laissé à la libre d'appréciation du client) il s'arrêtera d'écouter des messages sur le *port* indiqué dans le message et se mettre en écoute (UDP) sur le port standard du café,
- multidiffusion à la cantonade d'un message ordinaire de la forme `MESSAGE␣`*nom*$_{client}$ *message*. Un tel message ne provoquera pas de réponse particulière associée au niveau du protocole,
- un demande de conversation privée avec un autre client et qui nécessite alors le déroulement d'un sous-protocole :
  * multidiffusion préalable du message `HEY␣`*nomclient*$_{source}$`␣TO␣`*nomclient*$_{destination}$ qui indique qu'on souhaite attirer l'attention du client *nomclientdestination*,
  * si celui qui est concerné décide de ne pas rester sourd, il répondra aussi en multidiffusion un message de la forme `300␣HEY␣`*nomclient*$_{source}$`␣FROM␣`*nomclient*$_{destination}$`,`*ip*`,`*port* signifiant par là qu'on peut converser en TCP sur l'adresse et le port indiqué, la conversation privée étant décrite plus loin.
  * si celui qui est concerné décide de rester sourd, l'initiateur de la demande de conversation privée ne doit pas en être bloqué pour autant...

Example de conversation privée entre `achille` et `latortue`

On considère ici que c'est `achille` qui a initié la conversation privée, *i.e.* il a multidiffusé dans le café le message `HEY␣achille␣TO␣latortue`, lequel a répondu quelque chose comme `300␣HEY␣achille␣FROM␣latortue,192.168.0.17,45678`. Ainsi `latortue` se met en attente d'une connexion entrante TCP à partir de laquelle la communication s'effectuera alternativement de l'un à l'autre en commençant par un message de `achille` à `latortue`. Les messages sont de deux types :
- `MESSAGE␣`*message* pour envoyer un simple contenu,
- `SHUTUP` pour indiquer à l'autre partie que l'on souhaite arrêter la conversation privée, suite à quoi plus aucun message ne circulera et la connexion sera close par les deux parties.

Durant une conversation privée, les deux parties ignoreront purement et simplement les messages habituellement multidiffusés dans le café (deux ivrognes conversant ensemble sont sourd au reste du monde).

Un café

Tout café possède un *nom* et un *port* principal (choisi arbitrairement).

Un café, pour s'exécuter normalement, devra préalablement s'enregistrer auprès des autorités municipales en envoyant à l'entité ville correspondante un message `NEWSHOP␣`*nomcafe*`,`*port* indiquant qu'il souhaite s'ouvrir en utilisant le *port* principal spécifié ; la ville lui renverra alors, au choix :
- `200␣NEWSHOP␣`*ip* pour lui signifier que sa demande est acceptée et que le groupe de multidiffusion *ip* lui est attribué ;
- `404␣NEWSHOP␣existing␣name` pour lui signifier que sa demande est rejetée car un tel *nom* est déjà attribué à un café ;
- `500␣NEWSHOP␣city␣full` pour lui signifier que sa demande est rejetée car la ville possède déjà bien assez de cafés actifs. Le café est alors encouragé à réitérer sa demande plus tard (ceci afin de ne pas décourager l'esprit d'entreprise).

Une fois enregistré le café doit alors écouter des messages provenant sur son *port* principal (en UDP) sur lequel transitent donc les messages en provenance de clients (voir la description des clients et leur mode conversationnel), mais aussi sur son *port* principal (en TCP) pour les messages en provenance de la ville. Ces derniers peuvent être :

- `ALIVE`, auquel cas il doit répondre sans tarder `200␣ALIVE` ; ceci afin de signifier qu'il est en activité,
- `BROADCAST␣message` auquel il ne répond pas mais dont il extrait le *message* pour le multi-diffuser aux clients et en en profitant pour les conserver afin de les diffuser individuellement aux clients qui entreront plus tard dans le café (voir la description des clients).

Lorsqu'un café décide de fermer (il n'est pas spécifié dans ce document comment ni quand cela se produit) il envoie un message de la forme `CLOSE` à tous ses clients ainsi qu'à la ville.

Une ville

Toute ville possède un *nom* ainsi qu'une collection d'adresses autorisant la multidiffusion (collection fournie à son lancement, par example *via* un fichier de configuration).

Lors de la création d'une ville, celle-ci peut-être rattaché à d'autres villes existantes ou être autonome ; si elle est autonome une future ville pourra ou non s'y rattacher selon le caractère permanent de son autonomie (option au lancement). Le rattachement à un ville consiste à s'insérer dans un anneau : *i.e.* les villes attachées sont toutes reliées entre elles comme sur un cercle : la première à la seconde, la seconde à la troisième, ..., la dernière à la première. L'insertion dans l'anneau consiste pour la ville qui s'insère à ouvrir une connexion permanente TCP vers la ville cible (paramètre au lancement) qui sera donc son successeur dans l'anneau, puis à diffuser sur le nouveau brin de l'anneau le fait que désormais la nouvelle ville prend place (dans l'anneau) avant la cible ; l'information d'un nouvel arrivant dans le cercle des villes circulera de proche en proche de sorte que l'anneau se « refermera » correctement.

La ville et l'anneau (fable de Jean de La Fontaine ?)

Supposons, par exemple, qu'un anneau soit constitué des villes v_1, v_2 et v_3, où v_1 est connectée v_2 qui est connectée v_3 laquelle est connectée à v_1. Si v_4 (la nouvelle ville) s'insère en s'attachant à v_2 elle établit une connexion vers v_2 (ainsi v_4 est connectée à v_2) puis v_2 envoie un message à v_3 pour dire que v_4 est avant v_2, lequel transmet ce message à v_1 qui s'aperçoit qu'il faut donc rompre l'anneau qui l'amène vers v_2 et se connecter à v_4. L'anneau est alors v_1, v_4, v_2, v_3. Pour la suppression d'une ville dans un anneau, l'algorithme est du même ordre : on fait circuler un message pour indiquer comment refermer correctement l'anneau.

La gestion de l'anneau utilise alors les messages suivants :

- la ville qui désire s'insérer se connecte à la ville à laquelle elle se rattache en envoyant le message `INSERT␣`*nomville*`,`*ip*`,`*port*. À ce message deux réponses possibles :
  - `200␣INSERT` pour indiquer que l'insertion est acceptée et en cours ; auquel cas la nouvelle ville doit attendre la fermeture de l'anneau,
  - `400␣INSERT` pour signifier que la ville cible ne souhaite pas l'attachement.

  La ville cible fait alors circuler `SUBSTITUTE␣`*nomville*$_1$`,`*ip*$_1$`,`*port*$_1$`␣FOR␣`*nomville*$_2$`,`*ip*$_2$`,`*port*$_2$`␣`*ttl* sur l'anneau indiquant qu'il faut refermer l'anneau sur la *ville*$_1$ et non plus la *ville*$_2$. Ce message ne provoquera aucune réponse. Le dernier argument servira à ne pas faire tourner l'ordre indéfiniment sur l'anneau.

- puis le message de fermeture qui circulera de proche en proche jusqu'à la ville concernée par la fermeture de l'anneau.
- une ville qui désire s'enlever de l'anneau, utilisera le message initial `REMOVE␣`*nomville*`,`*ip*`,`*port* qui provoquera la réponse `200␣REMOVE` et supprimera par circulation du message de mise à jour adéquat la ville de l'anneau.

Les messages « ordinaires » des villes

Tout d'abord une ville peut recevoir un message `NEWSHOP␣`*nomcafe*`,`*port*, il a déjà été précisé à la section décrivant un café comment la ville doit répondre.

Elle doit lorsque c'est pertinent (en profitant d'un message à envoyer à un café ou autre chose, ou lorsqu'un client en demande la liste, etc) vérifier que les cafés préalablement enregistrés sont toujours en activité. Pour cela elle utilise un message `ALIVE` tel que décrit dans la section décrivant un café.

Elle doit aussi répondre correctement aux clients désireux d'entrer dans un café soit en répondant à `SHOPLIST` ou à `SHOPINFO␣`*nomcafe* tel que décrit à la section sur le client.

Par ailleurs, il est autorisé à ce que certaines entités (non spécifiées dans ce document) puissent demander à une ville de diffuser un message informatif d'ordre général à tous les cafés qui lui sont connectés ou par extension à toutes les villes et tous leurs cafés connectés. La réception d'un message `CAST␣LOCAL␣`*message* et `CAST␣GLOBAL␣`*message* provoquera le diffusion associée. Dans le premier cas, un message `NEWS␣`*message* sera diffusé à l'ensemble des cafés connectés à la ville, dans le second cas, il faudra ensuite faire circuler la demande de diffusion sur toutes les villes de l'anneau *via* un message `CAST␣GLOBAL␣FROM␣`*nomville*`␣`*ttl* (le dernier argument servant à ne pas faire tourner l'ordre indéfiniment sur l'anneau).

Le format des messages

Sauf dans certains cas de messages TCP (tiens ! lequels ?), les messages sont normalement limités à 256 octets de long.

Le format des messages est assez naturel et surtout il devrait vous permettre de vous passer lors des phases de tests de l'écriture de certaines entités en utilisant `telnet` ou `nc` par exemple. Ainsi le message `123␣TRUC␣BIDULE` signifie que les caractères suivant sont envoyés `1` (ASCII 49) suivi de `2`, puis `3`, puis un caractère d'espacement (ASCII 32), puis `T`, ...,`C`, puis un caractère d'espacement (ASCII 32), puis `B`, ..., `E` **et pour finir** le caractère ASCII 33 (qui sert de marqueur de fin de message, et qui est particulièrement utile pour les communication TCP mais qui ne doit **jamais** être oublié). Les caractères sont tous codés sur un seul octet en utilisant la table ASCII.

Pour les identificateurs tels que *nomcafe*, *nomville* ou *nomclient* il ne peuvent être composés que de caractères alphanumériques (majuscules ou minuscules) de la table ASCII standard (ce sont les seuls caractères acceptés, aucun symbole ni caractère accentué ne l'est).
`sataneprojetdereseauquinouscasselespieds` est acceptable, `Génial ce projet!` est refusé.

Les numéros *ip* sont codés par leur représentation sous la forme d'adresses numériques pointées *i.e.* `192.168.10.23` (avec la représentation de l'octet de poids fort de l'adresse IPv4 en

premier). Aucune autre représentation ne doit être acceptée. Précisons qu'il s'agit bien dans notre exemple du code ASCII du caractère `1` suivi de celui du caractère `9`, etc.

Les numéros de ports et les nombres sont aussi de représentés *via* leur forme décimale naturelle *i.e.* `51234` où chaque chiffre est encodé par l'octet dont la valeur est le code ASCII du caractère correspondant.

Les dates sont de représentation libre mais ne doivent pas contenir de caractère d'espacement.

Gestion des erreurs

Un maximum d'erreurs devront être détectées, toutefois il est possible d'être strict quant à l'application des règles décrites ou plus lâche. Le mode strict bien que conforme est souvent trop restrictif, il empêche souvent des client un peu mal écrits de fonctionner dans les scénarios interprocessus ; il est donc fréquent d'autoriser à interpréter certaines commandes mal formées. Évidemment, c'est à la libre appréciation des programmeurs, lesquels prennent l'entière responsabilité de fournir des applications qui potentiellement ouvrent la porte à des bugs...

Mode pédant

Tout message mal formaté doit être irrémédiablement ignoré ; mais cela ne signifie pas qu'une erreur ne puisse être renvoyée. S'il est prévu qu'une réponse doit normalement être renvoyée, alors le message `501␣SYNTAX` sera renvoyé et la partie correspondante du protocole annulée, les parties s'engageant alors à recommencer leur dialogue sur de bonnes bases.

Toute erreur empêchant la continuation normale des dialogues doit être détectée tant que possible et des décisions raisonnables prises essayant de maintenir un service normal. Si le service ne peut être maintenu, il faut alors arrêter si possible en prévenant les parties concernées.

Le champ *ttl* de certains messages est utilisé pour éviter que les messages concernés tournent trop longtemps dans l'anneau. En effet tout message circulant dans l'anneau est potentiellement « dangereux » puisqu'une mauvaise programmation pourrait consister à laisser un message tourner indéfiniment. Ce champ doit être employé de la manière suivante. Le premier émetteur du message le positionne à une valeur raisonnable (par exemple 10 ou 20, suffisant pour les tests), et chaque entité désireuse de retransmettre le message devra auparavant décrémenter cette valeur avant l'envoi. Si la valeur tombe à 0, le message doit être purement et simplement ignoré et non retransmis ; mais un message d'erreur doit être affiché par l'entité en question.

Mode lâche

Dans ce mode (de programmation) on s'autorise à interpréter certains messages alors que l'on ne devrait pas. Que chacun prenne de l'initiative si'l en a le courage ; mais qu'il assume aussi...

Réalisation

Ce qu'il faut programmer

La réalisation se fera nécessairement en C ou en Java, et possiblement dans les deux (ce qui serait un très bon exercice et un point qui sera particulièrement apprécié lors de l'évaluation [1]). Un groupe **d'au moins deux étudiants et d'au plus trois** (ni plus ni moins) devra réaliser au moins le strict minimum c'est-à-dire un client et un café fonctionnels (on doit pouvoir lancer plusieurs clients et un café). Ce strict minimum ne garantit pas la note de 10, mais simplement que la réalisation peut être considérée comme sérieusement recevable par les examinateurs.

Bien entendu, il faut réaliser une ville, même si la gestion de l'anneau peut, dans un premier temps, être omise. La gestion de l'anneau sera considérée comme un plus très intéressant.

D'autres entités non spécifiées comme les envoyeurs de messages informatifs seront considérés avec attention. Ainsi qu'une extension du protocole que vous pourriez proposer et que les enseignants s'autorisent à rajouter dans le projet. Toute critique sera bien entendu considérée avec attention, et notamment les réflexions en termes de sécurité (comment ne pas usurper la place d'un café, garantir qu'un client est bien qui il affirme être, etc).

Il est impératif de respecter scrupuleusement la spécification fournie dans le sujet ainsi que les formats des messages. Toute violation sera jugée très défavorablement ! Il vous sera de plus fourni un exécutable d'une ville, d'un café et d'un client pour que vous puissiez tester vos implémentations. Une bonne façon de faire est également de faire communiquer les entités de différents groupes de projet entre elles.

Nous mettrons en place un forum sur didel pour que vous puissiez communiquer entre vous et avec nous, par exemple pour faire part d'imprécisions dans le sujet, ou pour signaler que vous avez une entité qui tourne quelque part et ainsi donner l'opportunité à vos collègues de s'y connecter.

Dans tous les cas,

ne restez pas bloqués, demandez de l'aide à vos enseignants ! ! ! ! !

Aucun enseignant ne refusera de vous répondre. N'attendez pas la veille de la soutenance ou les quelques jours qui précèdent pour vous lancer dans le travail, prenez de l'avance, cela vous laissera de la marge pour résoudre les problèmes les plus importants.

Attention : vous pourriez être tentés de vouloir réaliser une interface graphique. Je tiens à préciser que ce serait une grave erreur, en effet vous aller perdre à coup sûr votre énergie sur un aspect de la solution qui n'a rien à voir avec le réseau. Nous considérons que la partie importante est la communication réseau, pas l'aspect final des applications. Toutefois, un développement raisonnable devrait permettre d'obtenir des affichages (en mode texte) qui ne se chevauchent pas trop (il y a peu, voire pas d'asynchronisme, dans les entités). En tous cas, venir à la soutenance avec une interface par-dessus une application qui ne fonctionne pas ne vous attirera que les foudres des examinateurs ; à l'inverse pas d'interface mais de quoi faire communiquer des applications tel qu'indiqué et vous serez couverts de louanges !

1. Attention : ne pas le faire dans les deux langages n'est pas une faute qui sera mal vue ! C'est de le faire dans les deux qui est un plus.

Diviser pour régner

La réalisation du projet se fera par groupe **d'au mois deux étudiants et d'au plus trois** (cette règle ne souffrira aucune exception, un peu d'effort de la part de chacun pour ne pas prendre que des amis proches dans son groupe devrait aider). Et bien entendu, chacun dans un groupe devra travailler et il n'est pas exclu que dans un cas flagrant de participations proche de zéro qu'une note différente du reste du groupe soit délivrée.