# “Protocole RINGO”
# Projet de Programmation Réseaux

**License Informatique - Université Paris Diderot**

**2016**

*Attention :* Prenez le temps de lire attentivement TOUT le document et ce dans ses moindres détails. Relisez plusieurs fois, même 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 inter-applications. 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). De plus les messages circulant sont indiqués entre crochets, **les crochets ne faisant pas partie du message**. À la fin du document le format des parties constituant les messages est décrit de façon précise.

## Introduction

Le but de ce projet est de programmer un protocole de communication basé sur une communication en anneau (*ring* en anglais). L'idée est que des entités communiquent entre elles par UDP et la topologie du

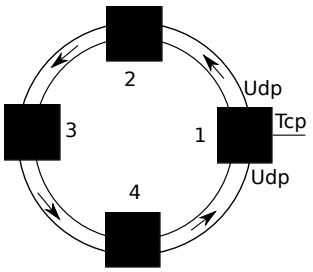


FIGURE 1 – Schéma d'un anneau

réseau est un anneau. La circulation des messages sur un anneau se fait de façon unidirectionnelle. Ainsi sur l'exemple d'anneau représenté par la Figure 1, l'entité 1 envoie ses messages sur le port UDP d'écoute de l'entité 2, l'entité 2 envoie ses messages sur le port UDP d'écoute de l'entité 3, etc. Chaque entité peut de plus attendre une connexion sur un port TCP pour permettre l'insertion d'un nouveau noeud dans l'anneau.

Le but de ce projet sera de programmer des entités pour réaliser ce protocole de communication en anneau, de programmer des applications utilisant ce protocole mais également d'inventer et de programmer de nouvelles applications.

Il sera également demandé de gérer des entités doubleurs servant à dédoubler des anneaux comme c'est le cas de l'entité 1 sur l'anneau représenté par la Figure 2.

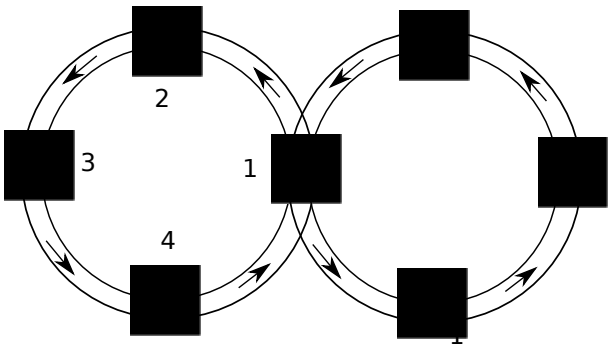


FIGURE 2 – Schéma d'un anneau avec un doubleur

# Description détaillée du protocole

## Les entités

Les caractéristiques d'une entité sont les suivantes :

- Un identifiant (d'au plus 8 caractères)
- Un port d'écoute pour recevoir les messages UDP de l'entité précédente sur l'anneau (inférieur à 9999)
- Un port TCP
- L'adresse IP de la machine suivante sur l'anneau
- Le port d'écoute UDP de la machine suivante sur l'anneau
- Une adresse IPv4 de multi-diffusion (qui servira pour signaler la panne du réseau)
- Un port de multi-diffusion (inférieur à 9999)

Ainsi une entité attend les messages du réseau sur son port UDP et elle transmet (les messages à transmettre) à la machine suivante sur l'anneau sur le port correspondant. Comme dit précédemment le port TCP sert pour des connexions courtes dans le temps, lorsqu'une nouvelle entité sert à s'insérer dans l'anneau ou lorsque l'on souhaite dupliquer un anneau. Toutes les entités d'un anneau partagent la même adresse et le même port de multi-diffusion qui sera utilisé lorsqu'une entité voudra signaler aux autres qu'elle a détecté une panne sur l'anneau.

## Insertion dans un anneau

Nous allons maintenant décrire comment se déroule l'insertion d'une nouvelle entité dans un anneau. Supposons par exemple qu'une entité souhaite s'insérer dans l'anneau décrit à la Figure 1 entre l'entité 1 et l'entité 2. Pour se faire il commencera par se connecter au port TCP de l'entité 1, celle-ci lui enverra alors le message suivant [`WELC␣ip␣port␣ip-diff␣port-diff\n`] où `ip` et `port` sont les caractéristiques de l'entité 2 sur laquelle la nouvelle entité devra transmettre ces messages et `ip-diff` et `port-diff` sont les caractéristiques de multi-diffusion à utiliser en cas de panne. La nouvelle entité répondra alors avec un message de la forme [`NEWC␣ip␣port\n`] avec son adresse IP et son numéro de port d'écoute UDP. Suite à quoi, l'entité 1 répondra par le message [`ACKC\n`] signalant qu'à partir de maintenant elle enverra à cette nouvelle entité les messages à transmettre, puis elle fermera la connexion TCP. À partir du moment où elle envoie le message [`ACKC\n`], l'entité 1 enverra désormais les messages à transmettre à la nouvelle entité et non plus à l'entité 2.

## Messages circulant sur l'anneau

Afin que les messages circulant sur l'anneau ne tournent pas indéfiniment chaque message aura un identifiant `idm` spécifique et **unique** encodé sur 8 octets et toute entité recevant un message déjà transmis ne devra pas le retransmettre à son successeur sur l'anneau. Tous les messages circulant sur l'anneau auront une taille maximale de 512 octets. Si un message est mal formé, alors il ne sera pas retransmis.

Sur l'anneau il y aura deux types de message : les messages des applications installées sur l'anneau et les messages de gestion du protocole.

Les messages d'application auront la forme [`APPL␣idm␣id-app␣message-app`] où `id-app` est l'identifiant de l'application codé sur 8 octets et `message-app` est le message dont ce sert l'application. Une entité qui reçoit un tel message, si elle ne supporte pas l'application concernée, transmet le message à l'entité suivante (sauf si elle a déjà vu passée le même message). On verra par la suite un exemple d'applications.

Parmi les messages de gestion du protocole, il y a par exemple le message [`WHOS␣idm`]. Lorsqu'une entité reçoit ce message, elle le transmet le cas échéant à l'entité suivante et ensuite elle envoie un message de la forme [`MEMB␣idm␣id␣ip␣port`] où elle précise son identité, son adresse IP et son port d'écoute (ce message faisant le tour de l'anneau). Ces deux messages font chacun le tour de l'anneau.

Il y a également les messages de déconnexion d'une entité. Lorsqu'une entité souhaite sortir de l'anneau elle envoie sur l'anneau un message [`GBYE␣idm␣ip␣port␣ip-succ␣port-succ`] où `ip` et `port` sont l'IP et le port d'écoute UDP de l'entité souhaitant sortir et `ip-succ` et `port-succ` sont l'IP et le port d'écoute UDP de l'entité suivante dans l'anneau. Ce message circule sur l'anneau jusqu'à arriver au prédécesseur de l'entité souhaitant sortir de l'anneau. Celui-ci valide alors la sortie en envoyant un message [`EYBG␣idm`], Il signifie ainsi à l'entité sortante que maintenant les messages à transmettre seront transmis à son successeur dans l'anneau. L'entité a alors quitté l'anneau à ce moment. Tant qu'elle n'a pas reçu le message `EYBG`, l'entité sortante doit continuer à assumer son rôle de transmission de message dans l'anneau.

Une entité peut également vouloir tester si un anneau est encore bien connecté. Pour cela elle peut envoyer un message de la forme [`TEST␣idm␣ip-diff␣port-diff`] où `ip-diff` et `port-diff` sont les IP et port de multi-diffusion de l'anneau testé. Ce message devra faire le tour de l'anneau testé, si au bout d'un certain temps (qu'il vous faudra choisir en faisant attention à ne pas prendre une valeur trop petite) elle ne reçoit pas son message, elle peut signaler à tout le monde que selon elle l'anneau est cassé en envoyant le message [`DOWN`] sur l'adresse et le port de multi-diffusion de l'anneau. Une entité qui reçoit un tel message se termine. Une entité qui reçoit le message `TEST` et qui ne se trouve pas sur l'anneau testé, c'est à dire qui se trouve sur un anneau ayant d'autres caractéristiques de multi-diffusion ne retransmettra pas ce message. Ce dernier cas peut arriver en cas de duplication d'anneau.

## Duplication

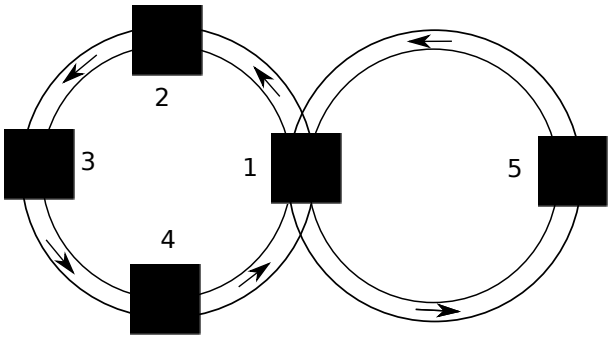


FIGURE 3 – Schéma d'un anneau avec un doubleur

Comme dit précédemment, il est également possible de doubler un anneau pour obtenir des anneaux comme celui représenté à la Figure 2 ou à la Figure 3. Pour obtenir cette dernière topologie, supposons que le nous sommes dans le cas de la topologie présentée à la Figure 1, l'entité 5 qui souhaite dédoubler un anneau commence par se connecter au port TCP de l'entité 1. Cette dernière lui envoie alors le message de la forme [`WELC␣ip␣port␣ip-diff␣port-diff\n`] comme dans le cas d'une insertion. La différence est que l'entité 5 répond alors avec un message de la forme [`DUPL␣ip␣port␣ip-diff␣port-diff\n`] où `ip` est son adresse IP, `port` son port d'écoute et `ip-diff` et `port-diff` sont les caractéristiques de multi-diffusion du nouvel anneau. L'entité 1 répondra alors avec un message [`ACKD\n`] signalant à l'entité 5 qu'elle accepte son rôle de doubleur.

Lorsqu'une entité devient doubleur elle ne peut plus accepter d'insertion de connexion ni de duplication, ainsi si elle reçoit une connexion sur son port TCP elle répond alors par un message [`NOTC\n`] pour refuser toute demande. De plus, une entité doubleur qui reçoit par multi-diffusion le message [`DOWN`] ne devra pas se terminer, elle devra seulement arrêter de transmettre les messages sur l'anneau cassé.

Une entité doubleur recevant des messages devra les transmettre sur les deux anneaux auxquels elle appartient sauf bien entendu les messages déjà vus.

# Exemple d'applications

Le but est finalement de faire tourner des applications sur le protocole décrit précédemment. Nous décrivons deux exemples d'applications à implémenter.

## Diffusion de messages

Cette première application est simple, elle sert juste à transmettre un message à tout le monde. Pour se faire, l'entité enverra un message de la forme [`APPL␣idm␣DIFF####␣size-mess␣mess`] où `size-mess` est la taille du message et `mess` est le message. Un tel message fera le tour de l'anneau.

## Transfert de fichiers

La deuxième application est une application de transfert de fichiers. Nous supposerons que les fichiers sur l'anneau ont un nom unique. Une entité souhaitant faire une requête de fichiers enverra le message [`APPL␣idm␣TRANS###␣REQ␣size-nom␣nom-fichier`] où `size-nom` est la taille du nom de fichier et `nom-fichier` est le nom du fichier qu'elle souhaite obtenir. Ce message fera le tour de l'anneau jusqu'à rencontrer une entité qui a le fichier `nom-fichier`. Cette entité ne passera pas le message à l'entité suivante et commencera le transfert de fichiers. (Si le message `REQ` revient à son émetteur, cela signifie donc que le fichier n'est pas présent sur l'anneau).

Elle commencera par envoyer un message de la forme :

[`APPL␣idm␣TRANS###␣ROK␣id-trans␣size-nom␣nom-fichier␣nummess`]

signifiant qu'elle va commencer un transfert dont l'identité pour le transfert est `id-trans` du fichier `nom-fichier` et que ce transfert sera composé de `nummess` messages. Elle envoie ensuite `num-mess` messages numérotés de 0 à *nummess* − 1 de la forme suivante :

[`APPL␣idm␣TRANS###␣SEN␣id-trans␣no-mess␣size-content␣content`]

où `id-trans` est l'identité de la transaction, `no-mess` est le numéro du message de transaction allant de 0 à *nummess* − 1, `size-content` est la taille des données envoyées dans ce message et `content` sont les données. Ces messages feront le tour de l'anneau jusqu'à arriver à l'entité qui a émis la requête qui ne les retransmettra pas. Si jamais les messages arrivent dans le désordre, l'entité qui a fait la requête n'est pas supposée reconstruire le fichier. Elle devra refaire la requête pour avoir le fichier.

## Spécification de la forme des messages

Afin que vos projets puissent communiquer entre eux, il est important que tous les messages respectent une forme bien précise. Nous décrivons dans cette partie de façon plus détaillée le format de chacun des messages.

Comme vous avez pu le remarquer, les quatre premiers octets d'un message serviront à encoder dans une chaîne de caractères son type. Les types des messages étant : `WELC`, `NEWC`, `ACKC`, `APPL`, `WHOS`, `MEMB`, `GBYE`,`EYBG`, `TEST`, `DOWN`, `DUPL`, `ACKD`.

Nous rappelons que pour les messages tournant sur l'anneau, leur taille ne devra jamais excéder 512 octets.

Voilà maintenant les caractéristiques pour chacun des champs contenus dans les messages :

- `ip`, `ip-diff` et `ip-succ` seront codés sur 15 octets et contiendront la chaîne de caractères correspondant à l'adresse IPv4. Par exemple, si l'adresse IP est 127.0.0.1, alors `ip` contiendra la chaîne `127.000.000.001` .
- `port` , `port-diff` et `port-succ` seront codés sur 4 octets et contiendront la chaîne de caractères correspondant au numéro de port.
- `idm` contient l'identité d'un message et sera codé sur 8 octets. Rappelons que chaque message généré pour circuler sur l'anneau aura une identité unique. On a les mêmes contraintes pour le champ `id-trans`.
- `id-app` est l'identité d'une application représentée par une chaîne de caractères codée sur 8 octets.
- `id` sera codé sur 8 octets et contiendra une chaîne de caractères.
- `size-mess` est une chaîne de caractères de 3 octets. Si par exemple la taille du message est 3, alors le champ `size-mess` correspondant vaudra `003`.
- `size-nom` est une chaîne de caractères de 2 octets. Si par exemple la taille du nom est 4, alors le champ `size-nom` correspondant vaudra `04`.
- `num-mess` est le nombre de messages, il sera codé en little-endian sur 8 octets. Le même critère s'applique au champ `no-mess`
- `size-content` est une chaîne de caractères de 3 octets. Si par exemple la taille du contenu est 56, alors le champ `size-content` correspondant vaudra `056`.

## Réalisation

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 pris en compte lors de l'évaluation). Un groupe devra programmer au moins tout le protocole et les deux exemples d'application.

Il est impératif de respecter scrupuleusement la spécification fournie dans le sujet ainsi que les formats de message. Toute violation sera jugée très défavorablement !

Il est vivement recommandé de proposer des extensions du projet, par exemple en proposant des nouvelles applications, cela peut-être un chat, des nouveaux services pour les entités, etc. Sur ces points **à vous de jouer** ! Ces applications devront être décrites de façon précise au moment du rendu du projet.

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 se connecter à elle. Ou encore pour préciser vos extensions. **Toute demande sur le projet devra passer par le forum.** Prenez garde à ne pas vous fier à la rumeur, la seule source d'information fiable sera le forum sur Didel ! Au moindre doute, n'hésitez pas à y poster votre question !

La communication verbale entre groupes est non seulement autorisée mais encouragée, cependant il est **strictement interdit** d'échanger du code ; ceci serait considéré comme plagiat et par conséquent jugé sévèrement. Les discussions doivent donc seulement porter sur le fonctionnement du protocole et son

interprétation, ou les formats des messages ; il vaut donc mieux éviter de donner trop d'indications sur la façon de coder les fonctionnalités.

La **notation finale** prendra en compte le fait que des groupes auront réussi à faire communiquer leurs applications entre elles. Il faudra donc penser à en faire la démonstration. D'une certaine manière, ceux qui collaborent dans les limites indiquées ci-dessus seront récompensés ; pour ceux qui décident de faire quelque chose dans leur coin et qui ressemblerait vaguement à ce qui est décrit, ce sera l'inverse, la notation sera revue à la baisse. Un **programme qui s'exécute n'est pas suffisant** ! Vous **devez** écrire un programme qui se comporte comme indiqué ; et s'il ne communique pas avec le programme écrit par d'autres, c'est que vous n'avez pas compris ce qu'est la programmation réseau.

Pour la soutenance, il sera **nécessaire de prévoir un mode de fonctionnement verbeux** dans lequel suffisamment d'informations seront disponibles à l'écran pour comprendre ce qui se passe (affichage des messages circulant, etc).

Vos programmes devront nécessairement pouvoir être exécutés sur les machines des salles de TP de l'UFR. Toute solution ne respectant pas ce critère sera jugée invalide.

Il n'est pas demandé de fournir une interface graphique ; il est même recommandé de s'abstenir d'en faire une (quoique vous en pensiez). Cela ne vous apportera rien, vous fera perdre du temps et n'impressionnera en rien les examinateurs ; et surtout **ce n'est absolument pas dans le programme de cet enseignement** par conséquent, nous insistons : **pas d'interface graphique**.

Votre projet devra bien entendu être robuste (c'est à dire sans erreur) et devra être capable de gérer des messages erronés sans planter.

La réalisation du projet se fera par groupe **d'au moins 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 des étudiants d'un même groupe n'aient pas la même note au final.

La composition des groupes devra être envoyée par mail à `sangnier@liafa.univ-paris-diderot.fr` avant **le Vendredi 25 Mars 2016 23h59**. Toute personne n'ayant pas soumis de groupe avant cette date prend le risque de ne pas avoir de note au projet.

Le projet devra être rendu sur Didel la veille de la soutenance, dont la date vous sera fournie ultérieurement. Des informations sur la soutenance (ordre de passage et instructions) seront aussi fournies plus tard sur Didel.