{ EPITECH }

Babel

Protocole de communication

Ngonta_e Benlou_d Barrau_a Renaud_j Benram_s Tholim_r

30/04/2011

Il s'agit d'un protocole de communication qui permettra aux deux groupes participant au projet Babel de pouvoir établir une communication Client / Client et une communication Client / Serveur.

Table des matières

I - Introduction **2**

II – Les paquets **2**

III - Communication Client to Server (CCS) **3**

A - Connexion 3

a - Enregistrement 3

b - Authentification 5

B – Gestion de la liste de contact 6

a – Ajout d'un contact 6

b – Retirer d'un contact 7

C – Mise à jour de la liste de contact 7

D – Appeler un contact 8

E – Déconnexion 9

IV - Direct Client to Client (DCC) **9**

A – Préliminaire 10

B – Echange de données 10

C – Interrompre une conversation 10

IV - Représentation numérique des types et des modes 11

A – Mode 11

B – Type de requête 12

C – Type d'erreur 13

D –Type de réponse 14

D –Etat des contacts 15

I - Introduction

BABEL est un projet de communication réseau basé sur le protocole SIP. Il est question pour nous dans ce projet, de développer une application permettant à des utilisateurs d'échanger voix, données et vidéos à partir d'un réseau connecté à internet.
Nous implémenterons un client graphique permettant à l'utilisateur d'accéder aux différentes fonctionnalités de l'application telles que appeler un contact de sa liste de contact, ajouter un ami à sa liste de contacts, retirer un ami, envoyer du texte, etc.
La partie serveur de l'application quant à elle permettra l'enregistrement et l'authentification des utilisateurs, qui devront posséder un compte (login, mot de passe,….) sur notre système pour pourvoir utiliser l'application. Le serveur permettra aussi l'interconnexion des clients distants pour les communications directes de client à client et pour les conférences.
Le protocole de communication que nous entendons établir ici est un protocole binaire et aura pour but de permettre à deux groupes participant au projet BABEL de pouvoir procéder à une communication Client / Client (Direct Client to Client / DCC) ainsi qu'à une communication Client / Serveur (CCS). Ceci permettra lors de la soutenance finale, de tester les projets des deux groupes en faisant jouer tour à tour à chacun le rôle de client et de serveur. La version du protocole que nous vous présentons ici n'est pas définitive et sera rendu plus stable à mesure du développement de l'application.
Nous verrons ici en détails ce que propose notre protocole DCC puis, par la suite, le protocole CCS. Mais avant tout, voyons à quoi ressembleront les paquets qui transiteront.

II - Les paquets

Les différents types d'envoi effectué par le serveur et le client se feront de la manière suivant:

<---------- 4 octets ---------->

<--- 65535 octets --->			
	Mode (1 octet)	Type Réponse (1 octet)	Taille du Paquet (2 octets)
	Data		

Les exemples seront schématisés de la manière suivante:

[*Mode* | *Type Réponse* | *Taille du Paquet* **][** *Data* **]**

Le mode et les types de réponses sont des réponses numériques qui sont définies pour représenter la donnée. C'est l'identité du paquet. Ceux-ci s'écrivent de la forme BBL*_[type | mode]*.

Le premier ensemble entre crochets correspond au header. Il est de quatre octets et comme décrit ci-dessus, celui-ci est constitué de trois parties:

- *Le Mode* : Le mode permet au serveur ou au client d'identifier le type de paquet reçu. Il sera de la forme BBL_MODE*_[mode d'envoi]*. Il informera le destinataire sur la façon de manipuler la donnée. Il sera stocke sur un octet.

- *Le Type de Réponse* : Le type de réponse informe le receveur du paquet sur la nature du paquet. Il permettra de connaître par exemple la confirmation de réception du paquet ou un type d'erreur concernant une information erronée. Il influencera aussi le comportement du destinataire pour la manipulation de la donnée reçus. Il sera stocké sur un octet. Il est de la forme BBL_RPL*_[type de réponse]* pour une réponse positive, de la forme BBL_ERR*_[type d'erreur]* en ce qui concerne les erreurs transitées et pour une simple requête émise, il s'écrit de la forme BBL_[*type*].

- *La Taille du Paquet* : Il est ici spécifié la taille total du paquet envoyé. Les paquets auront pour limite 65535 (header compris). Il sera stocké sur deux octets.

Le deuxième ensemble entre crochets correspond à la donnée, si nécessaire, à transiter entre les différents protagonistes de l'échange. Elle pourrait par exemple correspondre, dans le cas d'une communication entre deux clients, à des données représentant un son enregistré et compressé.

III - Communication Client to Server (CCS)

Cette partie nous permet d'instaurer les différents messages échangés dans un mode TCP instauré entre le Serveur et le Client. Nous commencerons par l'enregistrement / l'authentification pour ensuite passer par la demande d'appel, pour enfin finir par le protocole mis en place pour la mise à jour des listes de contacts.

A - Connexion

a - Enregistrement

Avant toute connexion, un utilisateur nouveau devra s'enregistrer auprès de notre serveur afin de pouvoir se connecter. Le mode d'enregistrement proposera donc au client de fournir quelques informations à propos de celui-ci, afin de pouvoir l'enregistrer dans sa Base De Données.

En entrant dans le mode d'enregistrement, les envois de données sont dans le mode BBL_MODE_REGISTER. Le type ici sera BBL_REGISTER. Les informations transitent de la manière suivante:

- le login:

  Il est demande à l'utilisateur d'entrer un login. Les types d'erreurs possibles sont:

  - BBL_ERR_LOGIN_INVALID : le login envoyé est invalide,
  - BBL_ERR_LOGIN_EXIST: le login est déjà existant.

  Dans ces deux cas, la data du message d'erreur sera vide.

- le mot de passe:

  Il est ensuite demandé à l'utilisateur d'entrer un mot de passe.

  L'erreur possible est la suivante:

  - BBL_ERR_PASS_INVALID : le mot de passe est invalide.

  La data du message d'erreur est aussi vide pour la réponse.

  Dans l'envoi de l'enregistrement, la data se présente de la manière suivante :

login|password

<u>À noter:</u> le symbole '|' représenté ici correspond au caractère '\0' et ceci est valide pour la suite.

Si le client clôt correctement l'enregistrement, une réponse est envoyée du client vers le serveur dans le mode BBL_MODE_REGISTER. Le champ est de type BBL_REGISTER_END.

La data est constituée du login correspondant au client ayant terminé l'enregistrement.

Exemple:

<u>envoi :</u> [BBL_MODE_REGISTER | BBL_REGISTER | 20][Koala|jackdaniels]

Ici, on aperçoit un envoi de header correspondant à l'enregistrement d'un certain *Koala*. La data contient le login de l'utilisateur. La taille du paquet correspond à la taille du header en ajoutant la taille de la data. On a 4 + 5 + 11 = 20.

<u>réception :</u> [BBL_MODE_REGISTER | BBL_REGISTER_END | 4][ ]

La réponse du serveur est le header avec une confirmation.

b - Authentification

Un utilisateur déjà enregistré peut accéder à Babel en s'authentifiant auprès du serveur dans le mode BBL_MODE_AUTH.

Le paquet envoyé possède un type BBL_AUTH. La data est, elle, sous la forme suivant:

login|password

Deux types de réponses sont possibles:

- BBL_RPL_AUTH_OK : authentification réussie,
- BBL_ERR_INVALIDAUTH : authentification échouée.

La data est vide pour ces deux cas.

Sans authentification, l'utilisateur ne pourra accéder à aucune fonctionnalité de BABEL.

Exemple:

<u>envoi :</u> [BBL_MODE_AUTH | BBL_AUTH | 18][Koala|poneyrose]

Envoi d'une tentative de connexion. La chaîne data contient le login *Koala* ainsi que le mot de passe *poneyrose.*

réception : [BBL_MODE_AUTH | BBL_AUTH_OK| 4][ ]

La réponse du serveur est le simple header avec une confirmation de connexion. On peut enfin accéder aux fonctionnalités de Babel.

B - Gestion de la liste de contacts

a - Ajouter un contact

Afin de communiquer, l'utilisateur se doit d'ajouter des contacts à sa liste d'amis. Pour cela, le mode BBL_MODE_ADDFRIEND permet d'identifier le paquet transportant les informations pour cette action. Le type de ce paquet est BBL_ADDFRIEND.

La data contiendra le login du contact à ajouter.

Deux types de réponses sont possibles :

- BBL_RPL_ADDFRIEND_OK : ajout réussi,
- BBL_ERR_NOTEXISTINGLOGIN : le login demandé n'existe pas.

La data est vide pour ces deux type de réponse.

Exemple:

envoi : [BBL_MODE_ADDFRIEND | BBL_ADDFRIEND | 9][Astek]

Envoi d'une tentative d'ajout du contact *Astek*.

réception : [BBL_MODE_ADDFRIEND | BBL_ERR_NOTEXISTINGLOGIN | 4][ ]

La réponse du serveur est le simple header avec une erreur avertissant que le login n'existe pas.

b - Retirer un contact

Si l'on veut supprimer un contact, il existe le mode BBL_MODE_DELFRIEND. Le type envoyé dans le paquet est BBL_DELFRIEND.

La data contient le login de la personne à retirer de la liste d'amis.

Trois types de réponse sont possibles:

- BBL_RPL_DELFRIEND_OK : contact retiré de la liste,
- BBL_ERR_NOTEXISTINGLOGIN : le login demandé n'existe pas,
- BBL_ERR_NOTLISTFRIEND : le login n'appartient pas à la liste d'amis.

La data reste vide pour ces réponses.

Exemple:

envoi : [BBL_MODE_DELFRIEND | BBL_DELFRIEND | 9][Astek]
Envoi d'une requête de suppression d'*Astek* de la liste de contact.
réception : [BBL_MODE_DELFRIEND | BBL_ ERR_NOTLISTFRIEND | 4][]
La réponse du serveur est le simple header avec une erreur signalant la non appartenance du contact *Astek* de la liste de contact.

C - Mise à jour de la liste de contacts

Afin de pouvoir rafraîchir la liste d'amis pour connaître le statut de chaque contact (connecté, déconnecté...), il existe un mode BBL_MODE_MAJ. La demande est de type BBL_MAJ.

La data est constituée du login de l'expéditeur.

Les différentes réponses possibles sont:

- BBL_RPL_MAJ : envoi la liste des contacts avec les états de chacun d'entre eux,
- BBL_RPL_ENDMAJ : représente le dernier paquet envoyé pour la mise à jour de la liste de contact.

Les paquets reçus du serveur contiennent tous une liste de contacts à mettre à jour. Le type de chacun sera BBL_RPL_MAJ excepté pour le dernier qui lui aura pour type BBL_RPL_ENDMAJ. Dans le cas où le serveur n'a qu'un seul paquet à envoyer, celui-ci aura pour type BBL_RPL_ENDMAJ.

Ces deux types de paquets contiendront la data suivante:

[login|state]*

<u>Nb :</u> '*' désigne la multiplicité possible de cette information dans la data.

Pour l'état du contact, on disposera d'un octet représentant l'état du contact:

- BBL_STATE_CONNECTED : contact connecté,
- BBL_STATE_DISCONNECTED : contact déconnecté,
- BBL_STATE_AWAY : contact connecté mais absent.
- BBL_STATE_BUSY : contact connecté mais occupé.

Il est important de noter que les états enregistrés en permanence dans la base de données sont CONNECTED et DISCONNECTED, les deux autres sont directement gérés au niveau du client lors de la session de l'utilisateur connecté. Certains états pourront être ajoutés à ce niveau pour plus d'intuitivité au niveau de notre interface graphique.

A chaque paquet BBL_RPL_MAJ reçu du serveur, le client renvoie un paquet de type BBL_MAJ jusqu'à réception d'un paquet de type BBL_RPL_ENDMAJ. Si un type autre que BBL_RPL_MAJ interfère, la mise à jour sera interrompue.

Ce mode sera beaucoup apprécié au moment de l'authentification de l'utilisateur ou pour rafraîchir sa liste de contact.

Exemple:

<u>envoi :</u> [BBL_MODE_MAJ | BBL_MAJ | 9][Koala]
Envoi d'une requête de mise à jour de la liste de contact de *Koala*
<u>réception :</u> [BBL_MODE_MAJ | BBL_ RPL_ENDMAJ | 4][Poney|BBL_STATE_CONNECTED| Pedobear|BBL_STATE_DISCONNECTED]
La réponse du serveur est un paquet qui se trouve être le premier mais aussi le dernier paquet envoyé pour mettre à jour la liste de contact de *Koala*. On peut observer que *Poney* est connecté mais que *Pedobear* est déconnecté.

D - Appeler un contact

Pour contacter un contact, le mode BBL_MODE_CALL est mis en place pour obtenir les informations nécessaires pour engager une connexion Client / Client.
Le type correspondant à l'obtention de ces informations est BBL_CALL.

La data est constituée du login du contact à joindre.

Les multiples réponses possibles sont les suivantes :

- BBL_RPL_CALL_OK : le contact est trouvé et est bien connecté. Une conversation peut être entamée. Ce paquet sera accompagnée de l'IP éventuellement NATée du correspondant à joindre dans la data,
- BBL_ERR_NOTEXISTINGLOGIN : le login demandé n'existe pas. La data est vide dans ce cas-là,
- BBL_ERR_AWAY : l'utilisateur que l'on cherche à joindre n'est pas disponible. La data contient le login de la personne pas joignable.

Exemple:

envoi : [BBL_MODE_CALL | BBL_CALL | 9][Poney]
Envoi d'une requête pour appeler *Poney*.
réception : [BBL_MODE_CALL | BBL_ CALL | 14][192.68.1.3]
La réponse du serveur est un paquet informant l'adresse IP (nattée) de la personne que l'on cherche à joindre dans la data.

E - Déconnexion

En ce qui concerne la déconnexion, le mode BBL_MODE_DISCONNECTION permet d'informer le serveur d'une déconnexion de l'utilisateur. Le type envoyé sera BBL_DISCONNECTION.

La data contiendra le login de la personne qui se déconnecte.

Pour comprendre comment s'opère une discussion, la partie suivante traitera des différents paquets pouvant être échangés.

IV - Direct Client to Client (DCC)

Comme expliquer dans la partie II de ce protocole, les échanges Client / Client se font en UDP. Avant d'entamer une conversation, pour que le destinataire à qui l'on veut parler sache l'identité de la personne voulant la joindre, il y a une étape préliminaire. On découpe donc un appel en deux étapes : les préliminaires et l'échange de donnée.

A - Préliminaire

Lorsque le client obtient du serveur l'IP éventuellement NATée du contact à joindre grâce au mode BBL_MODE_CALL, il tente une connexion auprès de celui-ci. Lorsque la connexion est établie, le client envoie un paquet de préliminaire présentant ce dernier au destinataire du paquet.

Le mode pour les préliminaires est BBL_MODE_PRELIMINARY. Le type envoyé pour se présenter est BBL_PRELIMINARY.

La data sera constituée du login de l'expéditeur du paquet.

Tant qu'il n'y a pas eu de réponse venant du contact, aucun autre échange ne sera possible pour l'utilisateur.

Les réponses à cette tentative d'échange sont les suivantes:

- BBL_RPL_PRELIM_OK : confirme la connexion et permet l'échange de donnée avec le contact. BBL_ERR_ REFUSED : la connexion est refusée, pas d'échange possible, la connexion est fermée.

La data contiendra le login de la personne acceptant la connexion.

La connexion étant ainsi établie et acceptée, le client peut procéder désormais à l'échange des données.

Exemple:

envoi : [BBL_MODE_PRELIMINARY| BBL_PRELIMINARY | 9][Koala]
Envoi d'une requête pour l'étape préliminaire de la communication entre *Koala* et *Poney*.
réception : [BBL_MODE_PRELIMINARY | BBL_RPL_PRELIM_OK | 4][Poney]
La réponse du client *Poney* est positive et la data contient le login de celui-ci.

B - Echange de données.

Le client sera capable de communiquer avec son interlocuteur à l'aide des paquets qui seront introduits dans cette partie. Pour ainsi faire, un mode BBL_MODE_DATA est mis en place pour échanger des données.

Sachant que plusieurs types de données sont capables de transiter, elles possèdent toutes la même structure mise a part la partie type qui diffère:

- BBL_DATA_VOICE : envoi d'une donnée audio,
- BBL_DATA_MSG : envoi d'une donnée texte,
- BBL_DATA_VIDEO : envoi d'une donnée vidéo.

La partie data contiendra les données voulant être envoyées.

Exemple:

envoi : [BBL_MODE_DATA| BBL_DATA_VOICE| ***X***][***data***]

Envoi d'une donnée correspondant à un envoi de donnée audio. ***data*** représente la donnée audio que l'on veut faire transiter (on ne peut représenter une donnée audio à l'écrit d'où la substitution par ***data***) et ***X*** représentant la taille total du paquet.

C - Interrompre une conversation

Pour interrompre une discussion, le client aura la possibilité d'envoyer, avant d'interrompre la connexion, un paquet possédant le mode BBL_MODE_DISCONNECTION. Le paquet envoyé est similaire à celui vu lors du CCS. Le type est le même (BBL_DISCONNECTION) ainsi que le contenu de la data qui contiendra toujours le login de l'expéditeur.

V - Représentation numérique des types et des modes

Dans cette partie, nous allons associer la valeur numérique pour chacun des modes, types, requête et états. Nous associerons, si nécessaire, un message en anglais pour chacun.

A - Modes

000 BBL_MODE_REGISTER

Le mode pour l'enregistrement d'un utilisateur.

001 BBL_MODE_REGISTER_END

Le mode pour désigner la fin d'un enregistrement.

002 BBL_MODE_AUTH

Le mode pour l'authentification d'un utilisateur.

003 BBL_MODE_ADDFRIEND
Le mode pour ajouter un contact à sa liste.

004 BBL_MODE_DELFRIEND
Le mode pour supprimer un contact de sa liste.

005 BBL_MODE_MAJ
Le mode pour mettre à jour la liste de contacts.

006 BBL_MODE_CALL
Le mode pour une tentative d'appel à un contact.

007 BBL_MODE_DISCONNECTION
Le mode pour annoncer la déconnexion d'un client.

008 BBL_MODE_PRELIMINARY
Le mode préliminaire pour pouvoir identifier l'appelant pour un client.

009 BBL_MODE_DATA
Le mode pour annoncer un échange de données entre deux clients.

B - Type de requête

020 BBL_LOGIN
Une requête pour enregistrer un login.

021 BBL_PASSWORD
Une requête pour enregistrer un mot de passe.

022 BBL_REGISTER_END
Une requête pour indiquer la fin d'un enregistrement.

023 BBL_AUTH
Une requête pour indiquer la tentative d'authentification auprès du serveur.

024 BBL_ADDFRIEND
Une requête pour indiquer l'ajout d'un contact à sa liste.

025 BBL_DELFRIEND
Une requête pour indiquer la suppression d'un contact de sa liste.

026 BBL_MAJ
Une requête pour demander une mise à jour de sa liste de contact.

027 BBL_CALL
Une requête pour tenter d'appeler un contact.

028 BBL_DISCONNECTION
Une requête pour annoncer la déconnexion d'un client.

029 BBL_PRELIMINARY
Une requête pour indiquer une introduction à une conversation afin de présenter l'identité du client appelant auprès du contact appelé.

030 BBL_DATA_VOICE
Une requête pour désigner l'envoi d'une donnée audio.

031 BBL_DATA_MSG
Une requête pour désigner l'envoi d'une donnée texte.

032 BBL_DATA_VIDEO
Une requête pour désigner l'envoi d'une donnée vidéo.

C - Type d'erreur

040 BBL_ERR_LOGIN_INVALID "You have an invalid login."
Une erreur signalant un login invalide syntaxiquement.

041 BBL_ERR_LOGIN_EXIST "<login> exist currently."
Une erreur signalant un login déjà existant.

042 BBL_ERR_REGISTER_END "You are not currently registered."
Une erreur signalant un échec au niveau de l'enregistrement.

043 BBL_ERR_INVALIDAUTH "Bad login or password"

Une erreur signalant une authentification invalide.

044 BBL_ERR_NOTEXISTINGLOGIN "<login> does not exist."
Une erreur signalant un login inexistant

045 BBL_ERR_NOTLISTFRIEND "<login> is not in your friend list."
Une erreur signalant un login n'appartenant pas à liste de contact de l'utilisateur.

046 BBL_ERR_AWAY "<login> is not busy."
Une erreur signalant que le contact que l'on cherche à joindre est occupé.

D - Type de réponse

060 BBL_RPL_LOGIN "<login> is valid."
Le login entré est valide.

061 BBL_RPL_PASS_OK "Password is valid."
Le mot de passe entré est valide.

062 BBL_RPL_REGISTER_END "You have been registered successfully."
L'enregistrement a bien été pris en compte.

063 BBL_RPL_AUTH_OK "You can log in."
L'authentification est acceptée.

064 BBL_RPL_ADDFRIEND_OK "<login> has been added successfully."
Le contact a bien été ajouté à la liste de contact.

065 BBL_RPL_DELFRIEND_OK "<login> has been deleted successfully."
Le contact a bien été supprimé de la liste.

066 BBL_RPL_MAJ
Indique un paquet pour la mise à jour de la liste de contact contenant des contacts et leurs états respectifs.

067 BBL_RPL_ENDMAJ "You're contact list has been updated."
Tout comme BBL_RPL_MAJ, mais en plus de cela il indique que c'est le dernier paquet pour la mise à jour des contacts.

068 BBL_RPL_CALL_OK
Le contact est joignable. On peut désormais établir une connexion.

069 BBL_RPL_PRELIM_OK "Call accepted."
Le contact confirme la connexion avec son interlocuteur. La conversation peut commencer.

E - Etat des contacts

080 BBL_STATE_CONNECTED
Le contact est connecté.

081 BBL_STATE_DISCONNECTED
Le contact est déconnecté.

082 BBL_STATE_AWAY
Le contact est absent / occupé.