****

**Ceg 3585**

**Programmation de socket et B8ZS**

****

March 10, 2023

universite d’Ottawa

**Membres du groupe**

Waffo Tony Adams (300310088)

**Assistants d’enseignements**

Félix Nadon

Reda Rahmoune

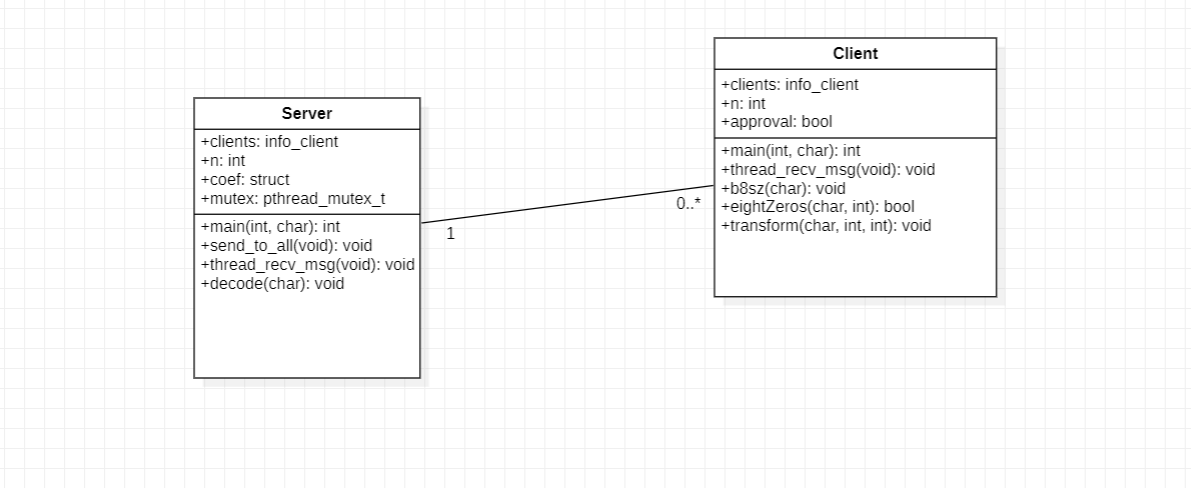
1. **Description brève du but et de la théorie du problème**

Ce laboratoire du cours de CEG3585 a pour but principal d’encoder une série de bit avec l’encodage B8ZS et de la transmettre à la machine serveur à l'aide de l’application client/serveur conçu au tout premier laboratoire. La théorie utilisée dans le cadre de ce laboratoire est basée sur le brouillage qui est une technique qui consiste à remplacer les séquences qui produisent un voltage constant. A cet effet, il est important de connaître la technique B8ZS comme présenté dans le cours. La particularité de ce laboratoire serait de réaliser une programmation qui serait non seulement capable d’encoder une série de bits chez le client, mais aussi de l’envoyer au serveur en utilisant les sockets où il sera décodé avant d’être affiché à l’écran. Pour ce faire, nous utiliserons la programmation C afin d’ajouter du contenu au travail initialement entamé dans le premier laboratoire.

1. **Explication brève de votre algorithme de solution**

En effet, une partie du travail avait déjà été faite dans un des laboratoires précèdent qui implémente les fonctions d’envoie et de réception par l’utilisation des threads. Mais à la suite des enjeux énumérés plus haut auxquels nous devons faire face, la solution consiste non pas à envoyer un message entre les entités du réseau, mais plutôt d’encoder les bits entrés par le client à l'aide du B8ZS, de tel sorte que le serveur puisse le décoder. Ainsi, le client se doit d’entrer une suite de 0 et de 1(sans espacements). A ce niveau, il encode les données en question avec la fonction b8sz () qui non seulement change les «1 » en « + » et « - », mais aussi qui détecte les séries de 8 zéros (avec la fonction booléenne eightZeros() ) et appelle la fonction transform() pour les transformer en «000+-0-+ » ou « 000-+0+- » dépendamment de la polarité  de la dernière impulsion. Une fois le message décodé, le client envoie un message au serveur demandant l’autorisation d'envoie (char authorize[50]= ‘approval request’). Ainsi le serveur se chargera de recevoir le message et d’approuver l’envoie en remplaçant le mot « request » par « confirm ». Ceci fait, le message est renvoyé au client qui va effectivement vérifier la présence du mot clé « confirm » à la suite de la réponse du serveur et lui envoyer en retour le message encodé après avoir assigné « true » a la variable approval. Le message transmis, le récepteur reçoit cette série de +, - et 0 et appelle la fonction decode() qui transforme les «+ » et «-» en 1, et remplace « 000+-0-+ » et « 000-+0+-» en une série de 8 zéros. Une fois décodé, il renvoie un message au client accusant la réception des données

1. **Document de conception en utilisant les diagrammes UML**



1. **Captures d’écrans de la démonstration de l’application**

Ci-dessous correspondent les captures d’écran de la connexion des clients au serveur et d’affichage des données encodées et décodées à l’écran.

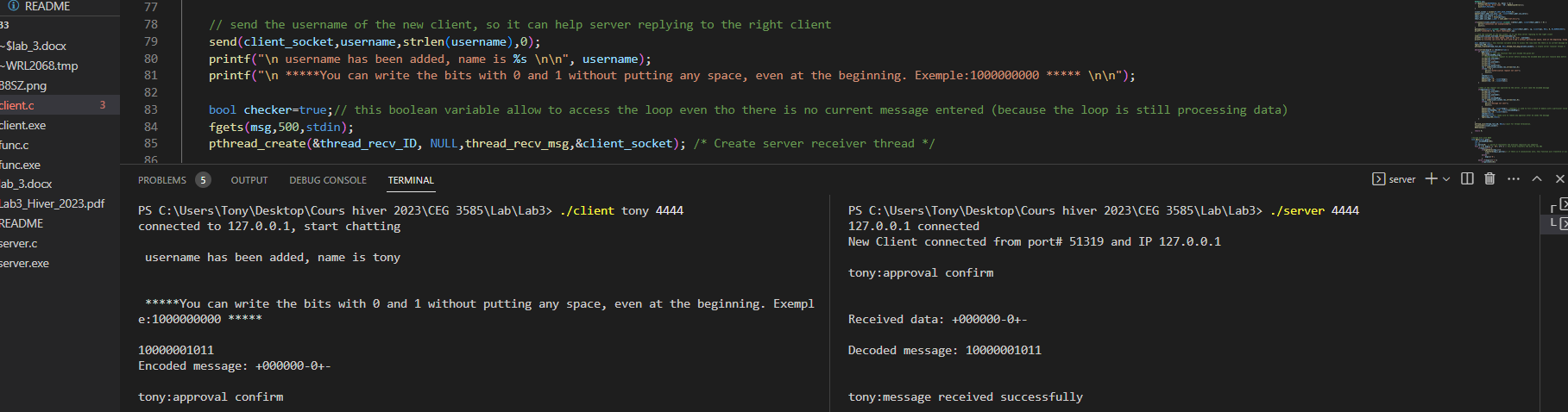


Figure 1 Connexion avec le serveur et envoie des données

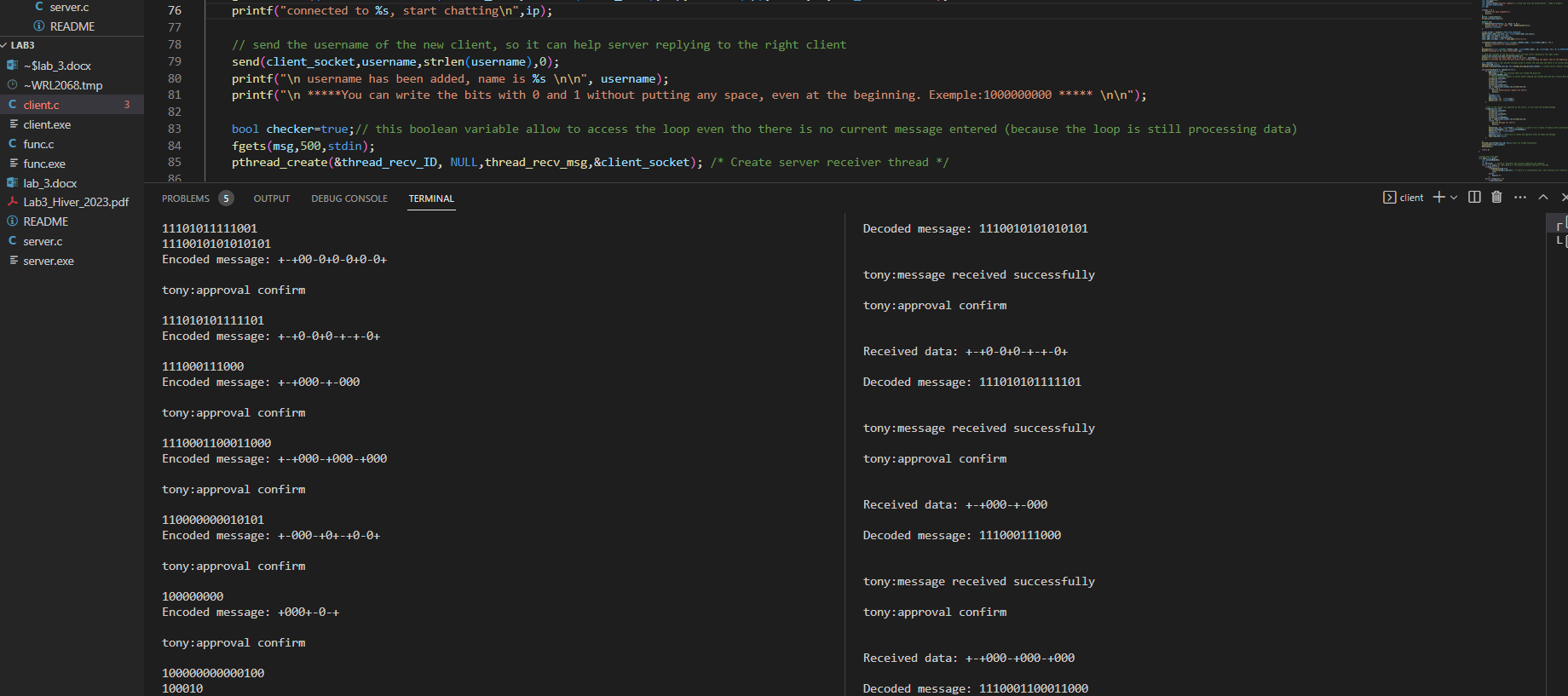


Figure 2

1. **Discussion**

Tout comme au dernier laboratoire, la programmation des sockets a été indispensable pour établir la communication dans le réseau. Une fois de plus, l’utilisation des threads s’est avérée importante, et nos connaissances du cours nous ont grandement aidé à aboutir à notre fin. Le code a bien été implémenté dans son ensemble et des défis majeurs n’ont pas été rencontrés. Toutefois, nous avons eu un petit problème au niveau de l’utilisation des fonctions send et recv. En effet nous essayions d’envoyer la demande d’autorisation, de recevoir la réponse du serveur et d’envoyer le message encodé au sein de la même itération dans la boucle while. Chose qui portait problème puisque le serveur ne recevait jamais le message encodé. Pour remédier à cette situation nous avons dû réaliser une programmation de telle sorte que cette action soit faite en deux itérations. Il était donc important de s’assurer que le programme n’oblige pas le client a re-saisir les données en incluant certaines variables booléennes tels que « checker ».

1. **Conclusion**

En définitive, nous pouvons affirmer que ce laboratoire a été une réussite puisque les objectifs ont été atteints. Il m’a largement instruit en ce qui concerne l’importance du décodage dans un réseau; d’où l’utilisation du brouillage B8ZS. Il a donc permis de passer en revue la méthode de conversion qui a été montrée en classe et de l’intégrer dans une communication réseau; ce qui augmente la complexité. Ainsi, en plus d’avoir mis en pratique mes connaissances sur la programmation des sockets en langage C, j’ai aussi mis en avant différentes fonctions de traitement de chaîne de caractère tout en approfondissant mes connaissances s’agissant des méthodes d’encodage et de décodage.