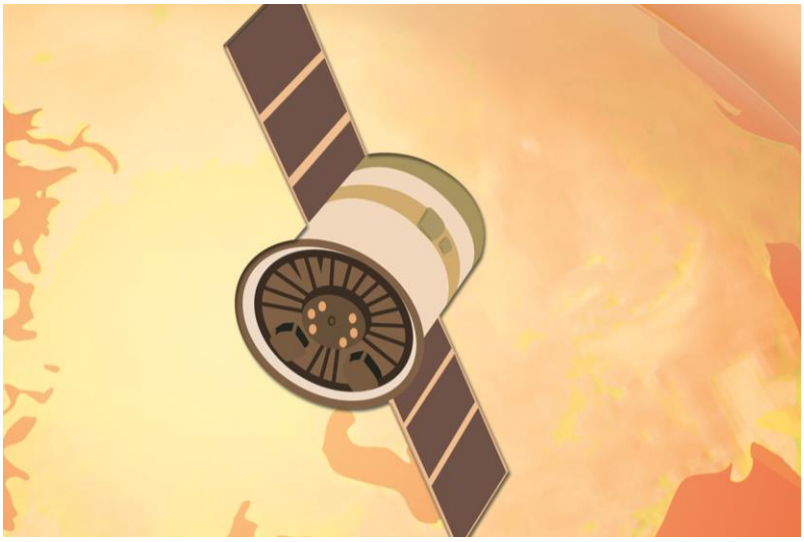
Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg

Bd Pérolles 80
case postale 32
CH-1705 Fribourg
t. +41 (0)26 429 66 11
f. +41 (0)26 429 66 00
www.eia-fr.ch

Projet Intégré

Systèmes embarqués – 2014

Analyse



Cours	Réseaux IP 2
Professeur	D. Gachet, <daniel.gachet@hefr.ch>
Etudiant	David Rossier, <david.rossier@edu.hefr.ch>
Classe	T-2a
Date	20.03.2014
Groupe	e

Hes·so FRIBOURG FREIBURG
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz

Table des matières

1. Introduction

Dans le cadre du cours de Systèmes embarqués 2, nous devons réaliser un projet intégré par groupe de 3 personnes. Notre groupe est composé de Loïc Gremaud, Romain Froidevaux et David Rossier.

Ce projet nous permettra de mettre en application la matière vue lors des cours de théorie et des travaux pratiques. Chaque membre du projet est chargé de tâches bien définies. Ce rapport a pour but d'analyser ces différents points pour faciliter la suite du projet (Architecture, puis conception).

Lors de ce projet, je suis chargé d'effectuer les tâches suivantes :

- Étude du protocole UDP et ICMP (Ping)
- Fonctionnement des jeux :
  - Tic-Tac-Toe
  - Bataille navale
- Gestion des scores

2. Étude des protocoles réseaux

Lors de ce projet, deux cibles devront être capables de jouer à un jeu à travers le protocole UDP/IP. Le but de ce point est d'étudier le fonctionnement et les en-têtes des protocoles Ethernet, IP, ICMP et UDP.

2.1. Ethernet

Notre professeur nous a fourni généreusement un Fast Ethernet Controller (FEC) afin que nous puissions utiliser le protocole Fast Ethernet à l'aide de la cible. Vous trouverez dans le tableau ci-dessous l'utilité de chacune des fonctions.

Fonction	Utilité	Paramètres
void fec_init()	Initialise le Fast Ethernet Controller	Aucun
void fec_start(bool promiscuous, bool broadcast)	Lance le FEC.	Promiscuous : true pour lancer en mode promiscuous Broadcast : true pour accepter les broadcast
void fec_stop()	Arrête le FEC	Aucun
int fec_recv(void* frame, size_t length)	Méthode pour recevoir un paquet ethernet Return : taille du paquet Ethernet reçu	Frame : l'adresse du buffer où stocker le paquet Length : Longueur du paquet
int fec_send(const void* frame, size_t length)	Méthode pour envoyer un paquet par le FEC. Return 0 = success, -1 = error	Frame : Adresse du buffer où est stocké le paquet à envoyer Length : Longueur du paquet
void fec_get_mac_address(uint8_t* mac)	Méthode qui retourne l'adresse MAC du FEC dans la mémoire flash	Mac : pointeur où sera stockée l'adresse MAC
Void Fec_set_mac_address(const uint8_t *mac)	Méthode qui fixe une adresse MAC	Mac : L'adresse MAC.

Tableau 1 : fonctions fast ethernet controller

Dans le tableau ci-dessous, nous pouvons voir l'en-tête d'une trame Ethernet. Elle nous sera utile pour définir le destinataire et compléter le paquet.

Preamble	Destination MAC address	Source MAC address	Type/Length	User Data	Frame Check Sequence (FCS)
8 bytes	6 bytes	6 bytes	2 bytes	46 – 1500 bytes	4 bytes

Tableau 2: Header Ethernet

Le paquet Ethernet peut contenir des en-têtes IP dans le paquet (User Data). Nous verrons dans le point suivant le contenu de ceux-ci.

2.2. IP

Dans le tableau ci-dessous, nous pouvons voir la taille et le contenu des paquets IPv4. Lors de ce projet, nous allons utiliser uniquement des adresses IPv4, bien qu'il soit possible de modifier son fonctionnement pour qu'il gère également les adresses IPv6.

<table>
<tr><th colspan="6">32 bits</th></tr>
<tr><td>Version
(4 bits)</td><td>Header
Length
(4 bits)</td><td colspan="2">Type of service
(8 bits)</td><td colspan="2">Total Length
(16 bits)</td></tr>
<tr><td colspan="3">Identification
(16 bits)</td><td>Flags
(3 bits)</td><td colspan="2">Fragment offset
(13 bits)</td></tr>
<tr><td colspan="2">Time to Live
(8 bits)</td><td colspan="2">Protocol
(8 bits)</td><td colspan="2">Header Checksum
(16 bits)</td></tr>
<tr><td colspan="6">Source address
(32 bits)</td></tr>
<tr><td colspan="6">Destination address
(32 bits)</td></tr>
<tr><td colspan="6">Payload</td></tr>
</table>

Tableau 3: IP Header

Voici la description des champs :

- Version (4 bits) : version du protocole IP.
- Header Length (4 bits) : nombre de mots de 32 bits composant l'en-tête.
- Type of service (8 bits) : indique la façon selon laquelle le paquet doit être traité.
- Total Length (16 bits): indique la taille totale du paquet en octets. La taille de ce champ étant de 2 octets, la taille totale du paquet ne peut dépasser 65536 octets.
- Flags : utilisé pour la fragmentation
- Fragment offset : utilisé pour la fragmentation
- Time To Live (8 bits) : indique le nombre maximal de routeurs à travers lesquels le paquet peut passer. Ainsi ce champ est décrémenté à chaque passage dans un routeur, lorsque celui-ci atteint la valeur critique de 0, le routeur détruit le datagramme. Cela évite l'encombrement du réseau par les paquets perdus.
- Protocole (8 bits) : permet de définir le protocole de couche supérieur. Quelques valeurs qui peuvent nous intéresser :
  - ICMP : 1
  - TCP : 6
  - UDP : 17
- Header checksum (16 bits) : permet de contrôler l'intégrité de l'en-tête afin de déterminer si celui-ci n'a pas été altéré pendant la transmission.

- Source address (32 bits) : représente l'adresse IP de la machine émettrice, il permet au destinataire de répondre
- Destination address (32 bits) : adresse IP du destinataire du message

Les paquets envoyés contiendront peu de données, il ne sera donc pas nécessaire de gérer la fragmentation des paquets. Les paquets IP peuvent contenir des paquets de type ICMP et UDP, que nous détaillerons dans les points suivants.

2.3. ICMP

Les paquets ICMP seront utilisés pour envoyer et recevoir des pings afin de vérifier le bon fonctionnement du réseau entre nos cibles (ou un autre appareil se trouvant sur le réseau). Le protocole ICMP se trouve dans le payload du paquet IP.

Dans le tableau ci-dessous, nous pouvons observer le contenu des en-têtes ICMP.

32 bits		
Type	Code	Header checksum
Rest of Header		

Tableau 4 : ICMP Header

Voici la description des champs :

- Type (8 bits) : Type de message ICMP. Quelques valeurs importantes pour le PING :
  - 0 : Echo Reply
  - 3 : Destination Unreachable
  - 8 : Echo Request
  - 11 : Time Exceeded
- Code (8 bits) : Sous type (donne des précisions)
- Header Checksum (16 bits) : permet de contrôler l'intégrité de l'en-tête afin de déterminer si celui-ci n'a pas été altéré pendant la transmission.
- Rest of Header : Utilisé selon le type et le code

2.4. UDP

Afin de communiquer entre elles pour jouer en réseau, les cibles utiliseront le protocole IP/UDP. Pour mieux comprendre son fonctionnement, il est important de connaitre les en-têtes d'un paquet UDP, afin d'envoyer avec succès les données d'un point à l'autre.

Dans le tableau ci-dessous, vous trouverez le contenu des en-têtes UDP :

32 bits	
Port Source (16 bits)	Port Destination (16 bits)
Length (16 bits)	Header Checksum (16 bits)
Données (longueur variable)	

Tableau 5 : UDP Header

Voici la description des champs :

- Port Source (16 bits) : Le port source du paquet (définit l'application source)
- Port Destination (16 bits) : Le port de destination du paquet (définit l'application de destination)
- Length (16 bits) : Longueur du paquet UDP
- Header Checksum (16 bits) : permet de contrôler l'intégrité de l'en-tête afin de déterminer si celui-ci n'a pas été altéré pendant la transmission.

Il sera important de définir un port spécifique pour nos jeux, en dehors de la plage de ports réservés (0 à 49152). Il faudra donc choisir un port entre 49153 et 65535.

2.5. Format des données transmises

Afin de pouvoir communiquer d'une cible à une autre, il sera nécessaire de définir un format précis pour pouvoir lire les informations fournies par UDP sur l'autre cible.

Il existe plusieurs manières d'effectuer cette opération en utilisant le moins de place possible, par exemple en sérialisant les données à envoyer. Pour l'instant, les seuls cas de transferts par UDP se font lors des jeux.

Un format possible :

G : int ID_JEU, W : boolean win, LS : int last_shot, POS : (int X_POS, int Y_POS)

3. Fonctionnement global

Lors de ce projet, je serai chargé d'implémenter le modèle et les différents contrôleurs de notre application. Dès lors, il m'est apparu important d'introduire toutes les interactions possibles entre les différentes interfaces.

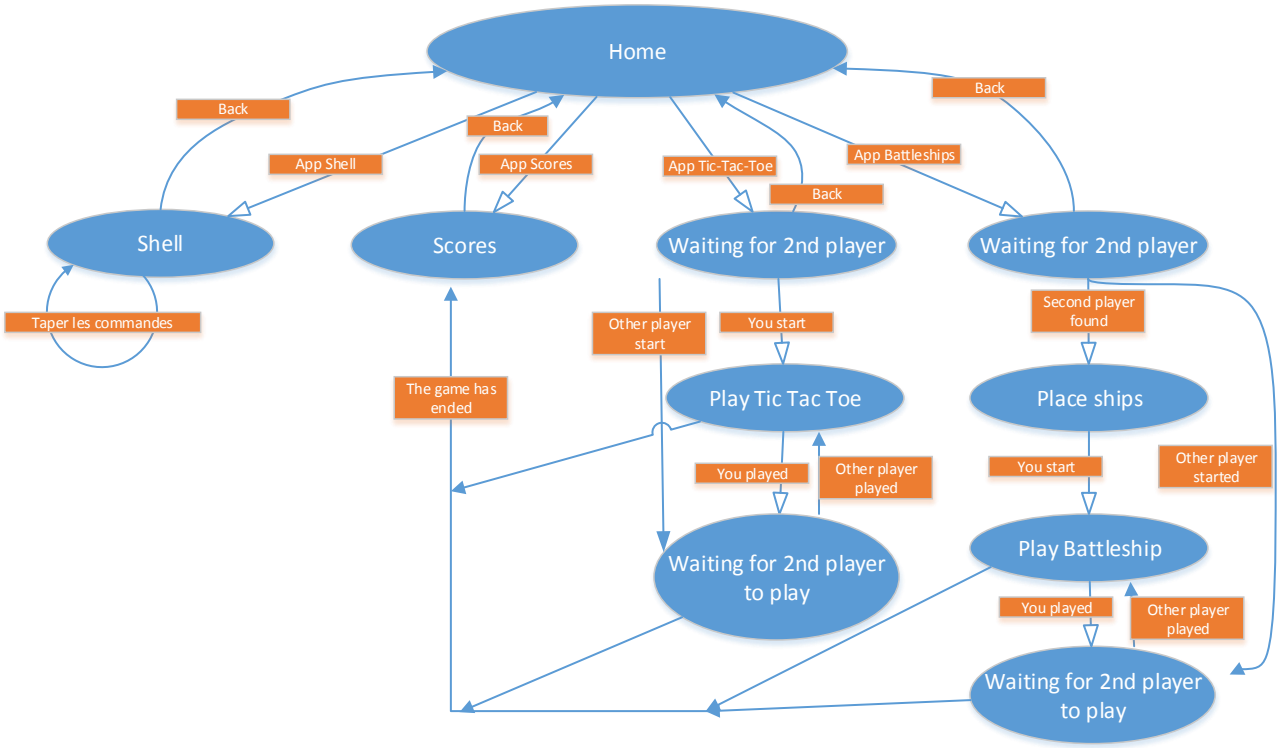


Figure 1: Fonctionnement global

3.1. Interfaces

De nombreuses interfaces permettront de définir l'endroit où nous nous trouvons dans l'application.

3.1.1. Home

C'est l'interface initiale sur laquelle on arrive lorsque l'application se lance. Le joueur pourra alors choisir une application à lancer.

3.1.2. Scores

Il s'agit de l'interface sur laquelle on peut voir les scores. Il est également possible de rejouer, si l'action qui y mène vient de l'un des deux jeux.

3.1.3. Shell

Il s'agit de l'interface qui permet d'utiliser une commande Shell. Cette dernière permet de configurer certaines fonctionnalités de l'application par l'intermédiaire d'une ligne de commande.

3.1.4. Waiting 2nd player (Tic-Tac-Toe)

Cette interface permet d'attendre que le deuxième joueur (sur une autre cible) soit effectivement connecté pour attendre la suite du jeu.

3.1.5. Waiting 2nd player (Bataille navale)

Cette interface permet d'attendre que le deuxième joueur (sur une autre cible) soit effectivement connecté pour attendre la suite du jeu.

3.1.6. Play Tic-Tac-Toe

Cette interface permet à un joueur de sélectionner puis de valider une case pour le Tic-Tac-Toe.

3.1.7. Waiting for 2nd player to play

Cette interface permet à un joueur d'attendre que le joueur adverse joue son coup.

3.1.8. Place ships

Cette interface permet à un joueur de sélectionner puis de valider les positions de ses bateaux.

3.1.9. Play Battleships

Cette interface permet à un joueur de sélectionner puis de valider une case où il souhaite tirer.

3.2. Modèle

Il sera nécessaire de stocker plusieurs données pour garantir le bon fonctionnement de l'application. Nous verrons dans ce point desquels il s'agit.

3.2.1. Interface

Il sera important de garder une information concernant l'interface courante, afin de simplifier le bon fonctionnement des événements repérés sur l'écran tactile LCD.

3.2.2. Shell

Il faudra également garder un historique des dernières commandes lancées afin de pouvoir maintenir l'affichage de la console à jour.

3.2.3. Scores

Le score sera également enregistré dans le modèle.

3.2.4. Jeux

Dans la bataille navale comme dans le Tic-Tac-Toe, l'état de la grille sera enregistré à tout moment dans le modèle. Ces informations sont indispensables pour que le jeu se passe correctement. On notera également le joueur qui devra jouer le prochain tour, et éventuellement le nombre de coups touchés, manqués et tirés pour la bataille navale.

3.3. Contrôleurs

Pour gérer le fonctionnement de l'application, il sera nécessaire d'utiliser plusieurs contrôleurs.

Les contrôleurs minimums sont les suivants, il est possible que nous en ajoutions d'autres par la suite si l'implémentation le requiert (exemple : Chronomètre, thermomètre)

3.3.1. Changer d'interface

Ce contrôleur permet de modifier l'interface actuelle à afficher.

3.3.2. Commencer un jeu

Initialise la grille du jeu choisi dans le modèle. Définit aléatoirement le joueur qui commence la partie.

3.3.3. Valider une case

Met à jour le modèle de la grille pour le jeu en cours, en y ajoutant le statut de la nouvelle case.

Ce contrôleur devra également vérifier l'état du jeu pour savoir s'il y a eu victoire ou non.

Il devra également envoyer l'information par UDP à la cible adverse.

3.3.4. Terminer un jeu

Met à jour le score en y ajoutant une victoire ou une défaite. S'il y a eu égalité, le score ne change pas.

4. Règles des jeux

Dans ce projet, nous avons décidé d'implémenter deux jeux : La bataille navale et le Tic-Tac-Toe. Dans ce point, nous allons établir les règles de chacun des jeux.

4.1. La bataille navale

Ci-dessous, quelques informations supplémentaires sur le jeu de la bataille navale :

4.1.1. Spécification pratiques

Après avoir étudié les dimensions de l'écran tactile LCD, nous avons opté pour une grille de jeu de 6x6 cases, et de trois bateaux : Un de 2x1, et deux de 3x1.

Les bateaux pourront être orientés horizontalement ou verticalement sur la grille.

Le jeu se joue au tour par tour, un joueur après l'autre. Chaque joueur se trouvant sur une cible. Les deux cibles seront reliées entre elles par un câble RJ45, et s'échangeront les informations du déroulement du jeu à l'aide d'un protocole UDP/IP (comme expliqué au point 2.2 et 2.4).

4.1.2. Préparation de la partie

Lorsque les deux joueurs ont lancé le jeu, ils pourront sélectionner un bateau, choisir l'orientation du bateau avec un bouton, puis sélectionner sa position sur la grille (à l'aide de l'écran tactile). Un bouton permettra de valider la position des bateaux.

Une fois tous les bateaux placés sur la grille (pour chacun des joueurs), le jeu commence. Le joueur qui commence est tiré au sort de manière aléatoire.

4.1.3. Déroulement de la partie

L'interface montrera une grille 6x6 contenant les coups tirés (touchés ou non) respectivement les coups tirés par l'adversaire et la position des bateaux selon le tour du joueur. Elle indiquera également le joueur qui doit jouer, ainsi que le nombre de coups touchés et manqués par le joueur qui joue son tour.

La grille permettra, lorsqu'il s'agit du tour d'un joueur, de sélectionner la case où il veut tirer. Un bouton permettra de valider le tir, respectivement, d'abandonner la partie (lorsque c'est à l'autre joueur de jouer).

4.1.4. But du jeu

Le premier joueur à avoir coulé (touché chaque position d'un bateau) tous les bateaux adverses gagne la partie.

4.2. Tic-Tac-Toe

Ci-dessous, quelques informations supplémentaires sur le jeu du Tic-Tac-Toe :

4.2.1. Spécification du jeu

Le jeu du Tic-Tac-Toe se joue sur une grille de 3x3 cases.

Le jeu se joue au tour par tour, un joueur après l'autre. Chaque joueur se trouvant sur une cible. Les deux cibles seront reliées entre elles par un câble RJ45, et s'échangeront les informations du déroulement du jeu à l'aide d'un protocole UDP/IP.

4.2.2. Déroulement de la partie

L'interface graphique sur l'écran tactile LCD montrera une grille 3x3, contenant les coups déjà joués, représentées par des croix et des ronds. Chaque joueur joue l'un après l'autre.

La grille permet de sélectionner, lorsque le tour actuel est celui du joueur, la case sur laquelle il veut positionner sa prochaine pièce, puis il devra la valider à leur d'un bouton sur l'écran.

4.2.3. But du jeu

Le premier joueur à avoir aligné 3 ronds, respectivement croix, horizontalement, verticalement ou en diagonale, gagne la partie. S'il n'y a aucun vainqueur une fois les 9 cases utilisées, la partie est déclarée nulle.

5. Gestion des scores

Nous avons trouvé intéressant de donner la possibilité de voir le score actuel entre deux cibles (Victoires, défaites). Nous avons donc décidé d'implémenter une interface sur l'écran LCD permettant d'afficher le score.

Le joueur pourra voir le nombre de parties qu'il a gagné, et le nombre de parties qu'il a perdu.

6. Conclusion

Cette analyse m'a permis de mieux comprendre la difficulté de créer un paquet réseau à partir de zéro. Chaque nouveau protocole nécessitera une encapsulation / décapsulation avant lecture / écriture sur le médium.

Elle m'a également permis de bien définir le contexte et les fonctionnalités de toute l'application.

Fribourg, le 23 mars 2014,

David Rossier