UNIVERSITÉ DE ROUEN	ANNÉE 2015-2016
FACULTÉ DES SCIENCES	RÉSEAUX
LICENCE 3 INFORMATIQUE	FICHE DE TD N° 9 (PROJET)

Client/Serveur pour le protocole TFTP (2ème partie)

Pour accélerer le transfert de fichier TFTP deux modifications du protocole ont été proposées :

1. la première modification consiste à augmenter la taille d'un bloc de données (RFC 2348),
2. la seconde consiste à ajouter un mécanisme de fenêtre d'envoi (RFC 7440).

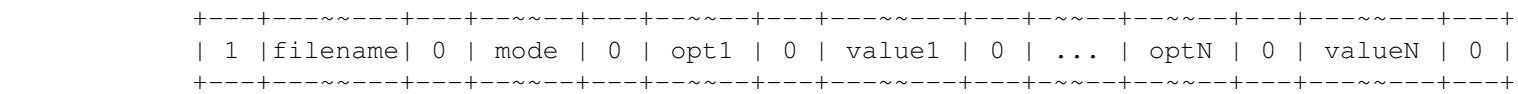
La RFC 2347 permet d'étendre le protocole TFTP en ajoutant une option à la fin d'un paquet RRQ (ou WRQ).

La seconde partie du projet consiste à mettre en oeuvre ces deux modifications et à mesurer les vitesses de transfert obtenu en fonction de la taille de bloc versus taille de fenêtre.

Option d'extension

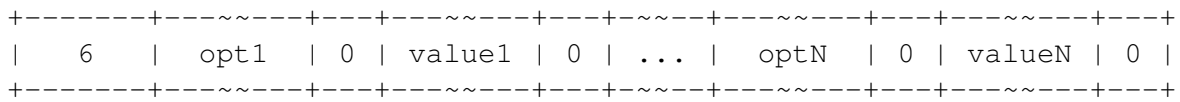
Le méchanisme d'option du protocole TFTP est décrit dans la RFC 2347 (`http://www.rfc-editor.org/rfc/rfc2347.txt`).

Les options sont concaténées au paquet RRQ comme ceci :



Chaque option est suivie d'une valeur. Options et valeurs sont transmises sous la forme de chaînes de caractères ASCII terminées par un zéro.

Un nouveau type de paquet OACK (code 6) permet d'accuser réception des options envoyées par le client.



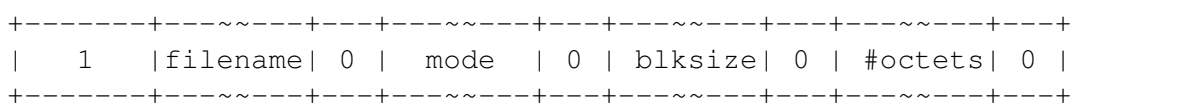
Le serveur n'envoie dans ce paquet que les options qu'il peut prendre en charge. Si la valeur d'une option prise en charge est invalide, la specification de l'option devra indiquer si le serveur doit ommettre l'option dans l'OACK, répondre avec une valeur alternative, ou envoyer un paquet ERROR avec un (nouveau) code d'erreur 8 qui permet de rejeter une option (ce qui entraînera l'interruption du transfert).

Le client accuse réception de l'OACK à l'aide d'un paquet ACK (numéro de bloc 0).

Option blksize

L'option `blksize` permettant de modifier la taille d'un bloc de données est décrite dans la RFC 2348 (`http://www.rfc-editor.org/rfc/rfc2348.txt`).

Un paquet RRQ avec l'option `blksize` a la forme suivante :



où `#octets` est une chaîne de caractère qui indique la taille maximale en octets des données d'un bloc DATA. Cette taille doit être comprise entre 8 et 65464.

Le serveur qui accepte cette option répond avec un paquet OACK indiquant une taille inférieur ou égale à la taille envoyée par le client. Le client doit alors utiliser cette taille ou envoyer un paquet ERROR, avec le code d'erreur 8, pour terminer le transfert.

Option windowsize

L'option `windowsize` permettant de mettre en oeuvre un mécanisme de fenêtre d'envoi est décrite dans la RFC 7440 (`http://www.rfc-editor.org/rfc/rfc7440.txt`).

Un paquet RRQ avec l'option `windowsize` a la forme suivante :

```
   2B     string   1B   string   1B     string     1B   string   1B
+-------+---~~---+----+---~~---+----+-----~~-----+----+---~~---+----+
|   1   |filename|  0 |  mode  |  0 | windowsize |  0 | #blocks|  0 |
+-------+---~~---+----+---~~---+----+-----~~-----+----+---~~---+----+
```

où `#blocks` est une chaîne de caractère qui indique la taille de la fenêtre (nombre de blocs). Cette taille doit être comprise entre 1 et 65535.

Le serveur qui accepte cette option répond avec un paquet OACK indiquant une taille inférieur ou égale à la taille envoyée par le client. Le client doit alors utiliser cette taille ou envoyer un paquet ERROR, avec le code d'erreur 8, pour terminer le transfert.

Le serveur envoi de façon cyclique le nombre de blocs fixé au départ et attend l'ACK du dernier bloc que le client doit envoyer. Si le client reçoie un bloc n+1 alors qu'il n'a pas reçu le précédent bloc n, il envoie l'ACK du dernier bloc reçu dans le bon ordre (n-1). Le serveur doit alors envoyer une fenêtre de blocs à partir du bloc n.

Exercice 1 Décrire l'ensemble des paquets échangés dans le cas où un client demande un fichier `bonjour.txt` à un serveur, le fichier contenant le texte `bonjour !!!\n` suivi de 512 espaces, dans le cas où on utilise :

1) l'option taille de bloc avec la valeur 400 (sans erreurs).

2) l'option taille de bloc avec la valeur 100 et l'option taille de fenêtre à 5 (sans erreurs).

3) l'option taille de bloc avec la valeur 100 et l'option taille de fenêtre à 5 avec le troisième bloc de données qui est perdu.

Exercice 2 Pour chaque type d'échange de messages qui peut être fait en considérant les options, indiquer les messages question/réponse qui peuvent se faire à partir du client et à partir du serveur.

Implantation d'un client/serveur TFTP (2ème partie)

Exercice 3 ***Outils de manipulation de paquets TFTP***

1) Définissez les fonctions suivantes :

```
int tftp_make_rrq_opt(char *buffer, size_t *length, const char *fichier,
                      size_t noctets, size_t nblocs);

int tftp_make_oack(char *buffer, size_t *length, uint16_t bloc,
                      size_t noctets, size_t nblocs);
```

qui créent un paquet tftp pour chaque type en mettant à jour la variable `length` avec la taille des données écrites.
Valeur de retour : 0 si tout s'est bien passé, -1 sinon.

2) Définissez des fonctions permettant de récuperer les informations contenues dans chaque type de paquet TFTP.

Exercice 4 ***Fonctions de Question/Réponse***

Écrire les fonctions question/réponse en prenant en charge les options taille de bloc et taille de fenêtre.

Exercice 5 ***Client/serveur TFTP***

1) Écrire le code d'un client TFTP qui peut réaliser un transfert de type RRQ en prenant en charge les options taille de bloc et taille de fenêtre.

2) Écrire le code d'un serveur TFTP multi-threads qui ne répond qu'à des requêtes de type RRQ en prenant en charge les options taille de bloc et taille de fenêtre.

3) Afficher dans le client le temps de transfert du fichier.

4) Étudier l'évolution de temps de transfert en fonction de la taille du bloc et la taille de la fenêtre.