Rapport de projet de Réseau

Pierre Monnin & Thibaut Smith

Année universitaire 2013-2014

Sommaire

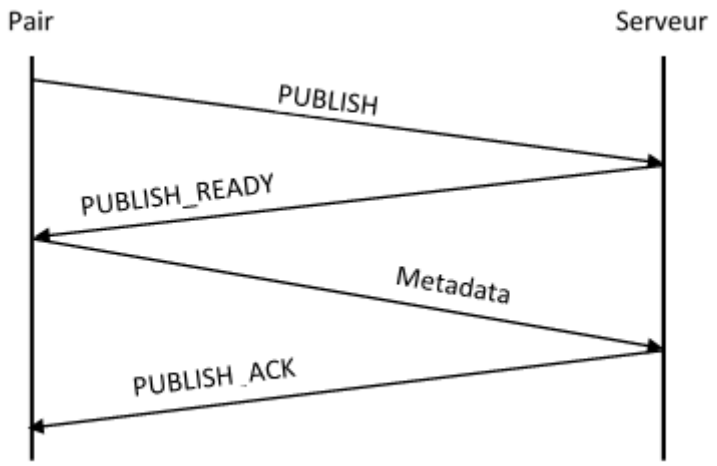

FIGURE 1 – Protocole de publication

1 Descriptions des protocoles applicatifs

1.1 Protocole de publication (pair-serveur)

Le protocole de publication est présenté sur la figure 1. Il commence par une requête `PUBLISH` de la part du client. Celle-ci annonce la volonté de ce pair de réaliser une publication sur le serveur central. Afin d'éviter de gêner l'écoute UDP, le serveur réalise une autre socket sur un port aléatoire généré par le système d'exploitation et répond, depuis ce port, `PUBLISH_READY`. Le client peut alors envoyer au serveur sur ce port les métadonnées du fichier sous la forme d'une structure `Metadata`. Une fois que le serveur a reçu cette structure, il renvoie un message `PUBLISH_ACK` au client terminant le processus de publication.

Le serveur continue à travailler pour ajouter le fichier à ses fichiers disponibles. Ce processus est décrit au paragraphe 3.

1.2 Protocole de recherche (pair-serveur)

Ce protocole est consultable sur la figure 2. Pour débuter l'échange, le client envoie une requête `SEARCH`. A nouveau, pour éviter de gêner l'écoute UDP avec d'autres messages, le serveur répond `SEARCH_READY` depuis un port aléatoire généré par le système. Le client écrit alors au serveur sur ce port pour lui envoyer le mot-clef de recherche saisi par l'utilisateur. Le serveur recherche alors des torrents correspondant au mot-clef. Il renvoie alors au client le nombre de résultats à recevoir suivi de ces résultats (un par message) s'il en existe.

1.3 Protocole de récupération d'un fichier (pair-pair)

Pour récupérer un fichier, le client commence par envoyer une requête `GET` suivie du SHA du fichier à récupérer sur le pair distant. Celui-ci recherche dans la liste des fichiers disponibles celui dont le SHA correspond à la valeur envoyée. S'il existe et est toujours disponible sur le disque, le

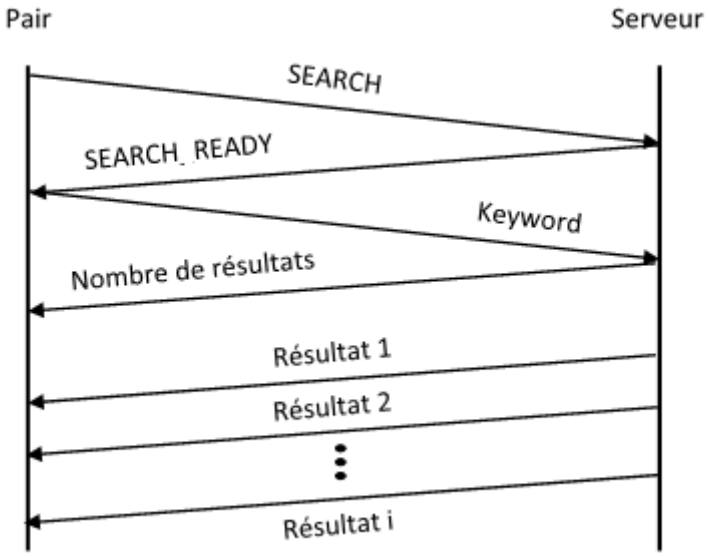

FIGURE 2 – Protocole de recherche

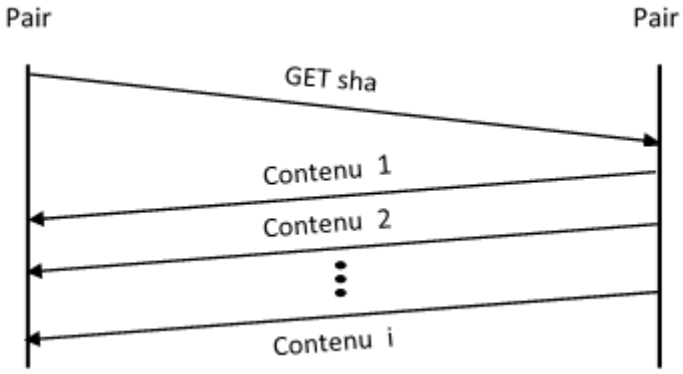

FIGURE 3 – Protocole de récupération d'un fichier

serveur envoie le contenu au client. Il est découpé selon la limite du protocole TCP. Le protocole global est visible sur la figure 3.

2 Algorithme d'un pair et structures de données utilisées

2.1 Structures de données utilisées

Sur un pair, nous avons besoin de stocker les fichiers publiés par ce pair sur le serveur. Pour cela, nous utilisons une hashmap contenant des listes chaînées pour gérer les collisions sur une même clef. Cette hashmap contient des éléments typés par la structure `LocalFile`. Celle-ci contient les métadonnées du fichier (structure `Metadata`) ainsi que son emplacement sur le disque dur (sous la forme d'une chaîne de caractères).

2.2 Algorithmes

Remarque : l'ensemble des phases d'attente sont soumises à un timeout. Quand celui-ci expire, un message est affiché à l'utilisateur l'invitant à recommencer l'opération lancée.

Algorithme général

1. Création d'un thread servant d'écoute TCP (voir algorithme de partage)
2. Si l'utilisateur demande à envoyer un fichier, exécution de l'algorithme de publication
3. Si l'utilisateur demande à effectuer une recherche, exécution de l'algorithme de recherche

Algorithme de partage

1. Création d'une socket d'écoute sur le port `PORT_ON_CLIENT`. Celle-ci est ajoutée à un `select` ce qui nous permet de gérer plusieurs téléchargements simultanés
2. Lors d'une demande de connexion d'un client : on parcourt la liste des fichiers publiés et on récupère la structure de type `LocalFile` dont le SHA correspond à celui envoyé par le pair distant.
3. On crée une nouvelle de socket de dialogue pour le client, qu'on écoutera via l'instruction `select`
4. Sur cette socket, on commence à envoyer le fichier au fur et à mesure en fonction des limites du protocole TCP
5. Lorsque l'envoi est terminé, on ferme la socket et on la retire du `select`

Algorithme de publication

1. Récupération auprès de l'utilisateur du chemin du fichier à envoyer
2. Vérification que le fichier existe
3. Construction des métadonnées (extension récupérée automatiquement, calcul du SHA sur le contenu du fichier, mots-clefs récupérés par une saisie utilisateur)
4. Envoi d'une demande de publication `PUBLISH` au serveur
5. Attente d'une réponse `PUBLISH_READY`
6. Envoi des métadonnées du fichier au serveur sur le nouveau port d'origine de `PUBLISH_READY`
7. Attente de l'acquittement `PUBLISH_ACK`
8. Retour à l'utilisateur

Algorithme de recherche et de récupération de fichier

1. Récupération par saisie utilisateur du mot-clef de recherche
2. Envoi d'une requête `SEARCH`
3. Attente de réception de la réponse `SEARCH_READY`
4. Envoi du mot-clef de recherche au serveur sur le port d'origine de la réponse `SEARCH_READY`
5. Attente de réception du nombre de résultats trouvés
6. Si celui-ci est différent de 0 :
   (a) Attente de réception de chaque résultat
   (b) S'il manque un résultat, afficher un message d'erreur et inviter l'utilisateur à recommencer sa requête
7. Si l'utilisateur choisit de télécharger un fichier en réponse de cette recherche, nous envoyons un `GET sha(fichier)` au pair dont l'adresse est contenue dans le résultat de la requête. Le téléchargement se met alors en marche et lorsqu'on reçoit le signal de fin de fichier, nous fermons la connexion.

3 Algorithme du serveur et structures de données utilisées

3.1 Structures de données utilisées

Pour stocker les informations sur le serveur, nous utilisons une hashmap constituée à l'intérieur de listes chaînées pour gérer les collisions. La hashmap utilise comme clef un hashage sur les mots-clefs associés à un fichier.

Lorsqu'un pair envoie un fichier au serveur, pour chaque mot-clef de recherche, on réalise le hashage et on range les métadonnées avec l'adresse IP du pair dans la liste chainée correspondant à l'entrée de la hashmap. On utilise pour cela une structure appelée `Torrent` qui contient les métadonnées envoyées par le client et l'adresse IP du pair obtenue via le code.

3.2 Algorithmes

Algorithme général du serveur :

1. Ouverture de la connexion UDP sur `PORT_ON_SERV`
2. Attente de connexion
3. Si on reçoit une requête `PUBLISH` : exécution de l'algorithme de publication
4. Si on reçoit une requête `SEARCH` : exécution de l'algorithme de recherche

Algorithme de publication

1. Réponse `PUBLISH_READY` au client depuis un autre port généré par le système (pour ne pas gêner l'écoute UDP)
2. Réception des métadonnées du client
3. Envoi de l'acquittement `PUBLISH_ACK`
4. Pour chaque mot-clef de recherche des métadonnées :
   (a) Hash du mot-clef
   (b) Ajout des métadonnées et de l'adresse IP du pair (sous la forme d'une structure `Torrent`) à la liste contenue dans l'entrée de la hashmap correspondant au hash du mot-clef

Algorithme de recherche

1. Réponse `SEARCH_READY` au client depuis un autre port généré par le système (pour ne pas gêner l'écoute UDP)
2. Réception du mot-clef de recherche du client
3. Hash de mot-clef
4. Parcours de la liste contenue à `hashmap[hash_keyword]` : pour chaque entrée, si le mot-clef est contenu dans les tags des métadonnées, incrémenter le compteur de résultats
5. Envoi du nombre de résultats au client
6. Si le nombre de résultats est différent de 0 :

(a) Se repositionner sur la liste à `hasmap[hash_keyword]`

(b) La parcourir en envoyant au client chaque entrée où le mot-clef est contenu dans les tags de la métadonnée.