MASTER IN COMPUTER SCIENCE

Téléchargement concurrent via interfaces multiples

Singleton, Thomas

Award date:
2007

Awarding institution:
University of Namur

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique
Année Académique 2006 – 2007

Téléchargement concurrent via interfaces multiples

Thomas Singleton

Mémoire présenté en en vue de l'obtention du grade de licencié en informatique

Téléchargement concurrent via interfaces multiples

Thomas Singleton

Mémoire présenté en vue de l'obtention du grade de licencié en informatique.

Résumé

Français.

La plupart des systèmes informatiques actuels disposent de plusieurs interfaces réseau, mais les protocoles et les modes de communications restent basés sur l'utilisation d'une seule interface à la fois. Ce travail propose d'étudier et d'implémenter une solution permettant de télécharger du contenu en utilisant plusieurs interfaces en même temps. Nous analysons d'abord les solutions existantes dans le domaine et les principes généraux. Ensuite nous présentons un protocole original que nous avons développé pour résoudre le problème. Enfin nous illustrons son fonctionnement dans différents scénarios de téléchargement.

English.

Currently most computers have multiple network interfaces but most of the protocols and communication methods presently in use rely on only one interface at a time. Our essay work examines and implements a novel solution for downloading data using multiple network interfaces simultaneously. We first explore the current technologies avaible and their underlying basic principles. We then present the original protocol designed to solve the issue at stake and demonstrate it's correct functionning using various download use cases.

Remerciements

Je tiens à remercier tous ceux
sans qui ce travail n'aurait pas pu être
mené à bien.

Mon promoteur Mr Schumacher
pour sa disponibilité et ses conseils
avisés.

Mes parents pour leur relecture attentive.

Ma compagne pour son soutien
indéfectible durant mes années
d'études.

Table des matières

1 Introduction

La plupart des équipements informatiques récents comportent plusieurs interfaces réseau. Les ordinateurs portables entre autres ont généralement une interface ethernet filaire et une interface ethernet sans fil. La plupart des serveurs ont au minimum deux interfaces ethernet. Il existe beaucoup de possibilités pour adjoindre une interface supplémentaire à un système n'en possédant qu'une à la base.

Les possibilités offertes par la présence de multiples interfaces sur un même système varient en fonction de son utilisation. Pour reprendre l'exemple cité, la présence de deux interfaces, l'une filaire et l'autre sans fil sur un ordinateur portable augmente la facilité avec laquelle il peut être utilisé dans divers endroits en tirant parti de l'infrastructure présente. Un deuxième exemple pourrait être celui d'un serveur possédant deux interfaces : il va pouvoir destiner l'une d'elles au trafic “normal” et l'autre à son administration à distance sur un réseau séparé. Cela lui permet une plus grande sécurité et fiabilité. Un tel serveur peut aussi utiliser ses deux interfaces à tour de rôle pour répartir la charge de travail entre elles. Un dernier exemple pourait être celui d'une personne travaillant dans un train sur un portable possédant une interface Wifi et une interface GPRS. Dans les gares, il peut profiter d'un accès sans fil proposé là comme dans de nombreux lieux publics et ensuite utiliser la connexion GPRS lors du voyage d'une gare à l'autre.

Si le nombre d'interfaces disponibles par système augmente et que les systèmes qui en possèdent plusieurs deviennent la norme, les modes de communication et les paradigmes des protocoles réseau se basent toujours sur un modèle de connexion point-à-point entre deux interfaces pour transmettre la totalité de l'information lors d'un dialogue entre deux hôtes. On peut observer la même évolution du point de vue du nombre de processeurs dans les systèmes. Alors que la tendance est à la multiplication du nombre de coeurs dans les processeurs actuels, les méthodes de programmation et algorithmes les plus couramment utilisés restent mono-processus.

C'est dans ce contexte que ce mémoire propose une démonstration par l'exemple de l'intérêt que présente l'utilisation par une application de plusieurs interfaces d'un même hôte pour réaliser un échange de données avec un autre

système informatique. Ce mode de fonctionnement fait d'ores et déjà l'objet de brevets déposés par des tiers [PAT].

Le deuxième chapitre de ce mémoire expose plusieurs solutions existantes de communication multi-interfaces, SCTP et deux évolutions de TCP, GridTCP et pTCP. Le troisième chapitre constitue le coeur de ce travail : le protocole original que nous proposons, Multi Interface Transfer Protocol (MITP) et son implémentation de démonstration, réalisée en Python. Ensuite nous généraliserons ce principe afin de proposer une solution applicable à un protocole plus général comme HTTP. L'avant-dernier chapitre décrit plusieurs scénarios de fonctionnement qui permettent d'illustrer le comportement de la solution présentée. Enfin, le dernier chapitre synthétise les résultats de ce travail.

1.1 Conventions

Au préalable, il importe de préciser certains termes et la signification particulière qui leur est accordée dans ce texte.

- Connexion : Une connexion représente un lien logique (ici via un réseau informatique) entre deux interfaces de deux hôtes; ce lien permet d'échanger des données.
- Interface : Une interface est le composant informatique matériel ou virtuel qui relie un hôte à un réseau informatique. Les interfaces d'un hôte peuvent être de deux types (filaire, sans fil) et peuvent utiliser différents protocoles ou technologies de réseau (ethernet, PPP, ...).
- Hôte : Ici, un hôte représente un système informatique capable d'établir des connexions. Un hôte peut posséder plusieurs interfaces, être connecté à plusieurs réseaux et dialoguer avec plusieurs autres hôtes.

Notons aussi qu'il sera question dans notre travail de “division du contenu” à transférer en plusieurs fractions indépendantes. Il est clair que cette opération de division du contenu est déjà effectuée à un niveau “sous-applicatif” par la pile TCP/IP pour les réseaux à commutation de paquets. Ce qui nous préoccupera ici est la division du contenu au niveau applicatif plutôt que dans les mécanismes sous-jacents de connexion.

Illustration des termes définis :

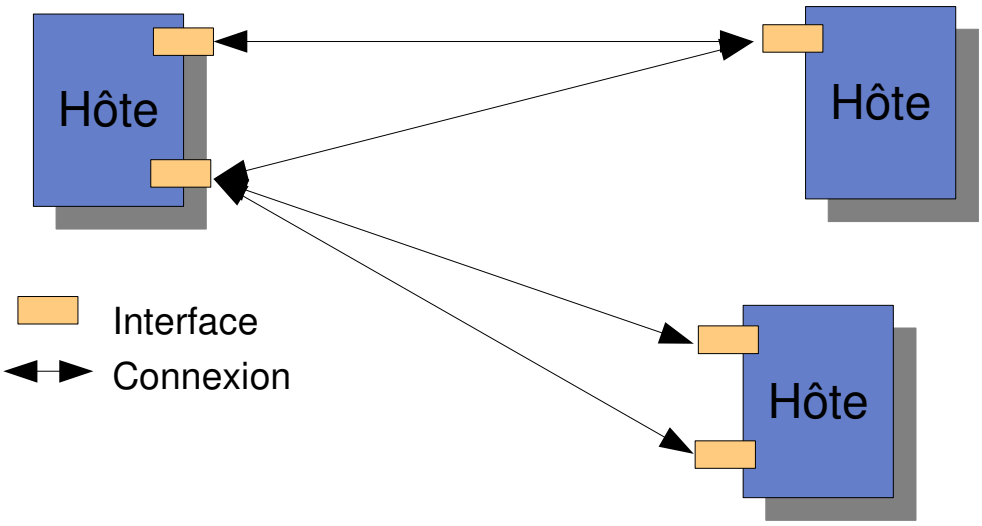


Figure 1 : Hôte – Interface – Connexion

Dans le schéma de la Figure 1, un hôte utilise ses deux interfaces pour se connecter à plusieurs autres hôtes. Chaque connexion se fait entre deux interfaces, il y a donc ici quatre connexions.

2 Etat de l'art

L'utilisation simultanée de plusieurs interfaces réseau pour un échange de données entre deux hôtes peut se faire à plusieurs niveaux. Il est possible de résoudre le problème au niveau du système d'exploitation, dans les couches de base de la pile TCP/IP ou à un niveau supérieur, en gérant les multiples interfaces dans l'application même et en se basant sur la fonction standard de connexion point-à-point offerte par les protocoles comme TCP ou UDP. Notons que lorsqu'on parle ici de connexion point-à-point, il s'agit de connexion interface-à-interface.

Présentons d'abord deux solutions existantes ayant une implémentation de bas niveau. Ces deux solutions définissent de nouveaux protocoles équivalents aux protocoles TCP ou UDP.

2.1 SCTP

SCTP, pour “Stream Control Transmission Protocol”, est un protocole situé au niveau “transport” dans les couches standard du modèle TCP/IP. Ce protocole a été proposé en 2000 par le groupe SIGTRAN de l'IETF dans le RFC 2960 [SCTP1]. SCTP a initialement été développé pour transporter des données de contrôle PSTN sur des réseaux IP.

SCTP se situe au même niveau que TCP et UDP dans le modèle TCP/IP et offre les mêmes garanties que TCP :

- Transfert fiable des données, garantie sur l'absence d'erreurs
- Respect de l'ordonnancement des données, avec une option pour le désactiver si l'utilisateur de SCTP le souhaite

A la différence de TCP, qui est basé sur l'échange d'un flux d'octets bi-directionnel, deux hôtes communiquant avec SCTP échangent un ensemble de messages au travers d'un ou plusieurs canaux. En outre, dans un échange SCTP, les hôtes communicants peuvent utiliser plusieurs interfaces pour réaliser cet échange. En TCP on parle de “connexion”, avec SCTP on parle d'une “association”.

Les associations SCTP présentent plusieurs avantages par rapport à une connexion TCP. L'utilisation de plusieurs interfaces permet une redondance et une résistance aux pannes. L'échange de messages (“message-stream”) plutôt que d'un flux de bits (“byte-stream”) non délimité dispense les utilisateurs d'insérer des limites artificielles dans les données pour pouvoir les séparer au niveau applicatif, couche supérieure, à la réception. Par exemple, pour la transmission de messages de longueur variable, il n'est plus nécessaire de convenir de caractères de contrôle qui signaleraient la fin d'un message et le début du suivant. L'ordonnancement strict entre les messages peut, en outre, être relaxé afin d'éviter les problèmes de délai en cas de retransmission dans un des flux de messages d'une association.

Comparons schématiquement une connexion TCP et une association SCTP.

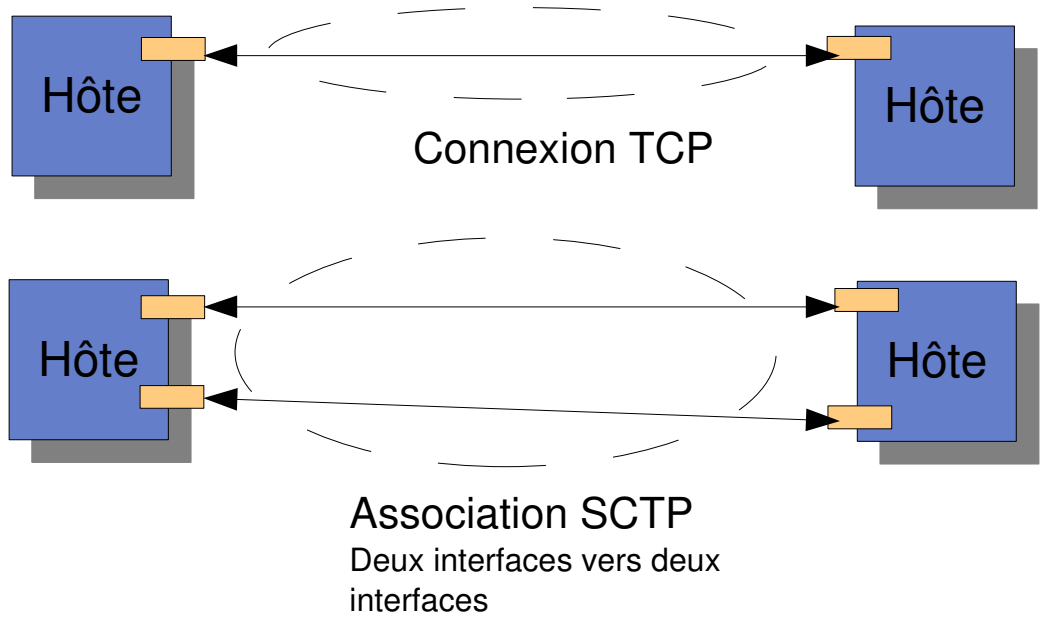


Figure 2 : Connexion TCP – Association SCTP

Remarquons aussi que les associations SCTP ne doivent pas forcément être symétriques du point de vue du nombre d'interfaces entre les deux hôtes.

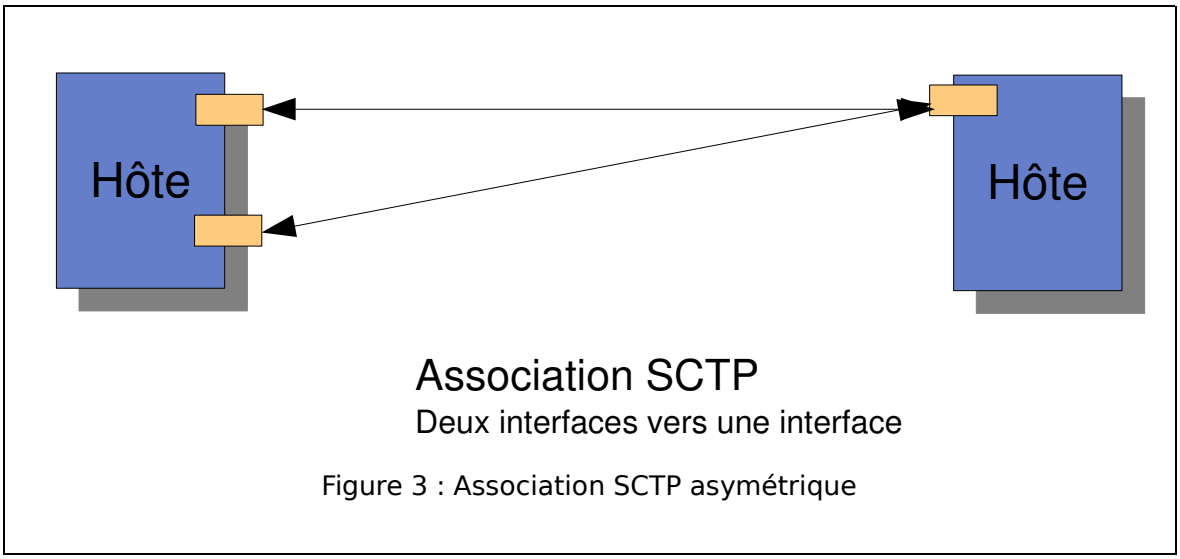


Figure 3 : Association SCTP asymétrique

SCTP rejoint un des objectifs poursuivis dans ce mémoire : l'utilisation simultanée de plusieurs interfaces de connexion pour un même transfert de données. Mais la politique de gestion des différentes interfaces n'étant pas directement définissable par l'utilisateur, l'utilisation du ou des chemins alternatifs doit être explicite dans les appels à la fonction de transmission. Par défaut, les chemins alternatifs ne seront utilisés qu'en cas de problème sur le chemin principal.

2.2 Evolutions TCP

TCP, pour “Transmission Control Protocol”, est un des protocoles au coeur de la pile des protocoles utilisés par les systèmes informatiques actuels pour transmettre de l'information sur des réseaux IP. Il est d'ailleurs synonyme de ce type de réseau, puisqu'on parle de réseau TCP/IP pour indiquer les deux protocoles principaux. Mais TCP est un procole mis au point au début des années 1980. Malgré une remarquable adaptabilité aux changements de technologies et une évolution constante qui font qu'il reste le protocole de base de la majorité des communications sur Internet, il accuse certaines faiblesses pour des réseaux particuliers ou des utilisations spécifiques.

Certaines des améliorations proposées au protocole TCP [TCPRMP] ont pour but de réduire la latence ou d'améliorer le comportement en cas de congestion via de nouveaux algorithmes afin de gérer celle-ci efficacement.

D'autres améliorations concernent l'optimisation de l'utilisation d'une large capacité de bande passante pour des transmissions à longue distance d'une grande quantité de données. C'est le cas de figure, par exemple, pour la transmission de données provenant d'expériences en physique des particules entre le lieu d'expérimentation et le(s) lieu(x) où les données doivent être exploitées [GRIDFTP].

Dans ces cas, l'utilisation d'un seul flux TCP pour transmettre les données pose problème lors d'une éventuelle perte de paquets. Celle-ci est interprétée par l'implémentation TCP de base comme de la congestion en amont, et, en conséquence, le système réduit sa vitesse d'émission. La récupération de la vitesse maximale de transmission peut prendre du temps à cause de l'approche TCP d'augmentation linéaire / diminution multiplicative de la vitesse d'émission en cas de problème.

Plusieurs solutions ont été proposées. En règle générale elles visent des applications particulières de transmission de volumes importants de données d'expériences scientifiques, comme l'exemple utilisant GridTCP cité ci-dessus. L'approche commune de ces différentes propositions se base sur le “striping” des données à transmettre. D'une façon analogue à ce que les systèmes RAID effectuent dans le domaine du stockage (à savoir répartir les données à écrire sur plusieurs disques de façon tournante et considérer l'ensemble de ces disques comme une seule unité logique), les méthodes proposées préconisent la répartition des données à transmettre sur plusieurs flux TCP parallèles de l'émetteur vers le récepteur. Une implémentation simple de ce principe peut être trouvée dans pTCP [PTCP].

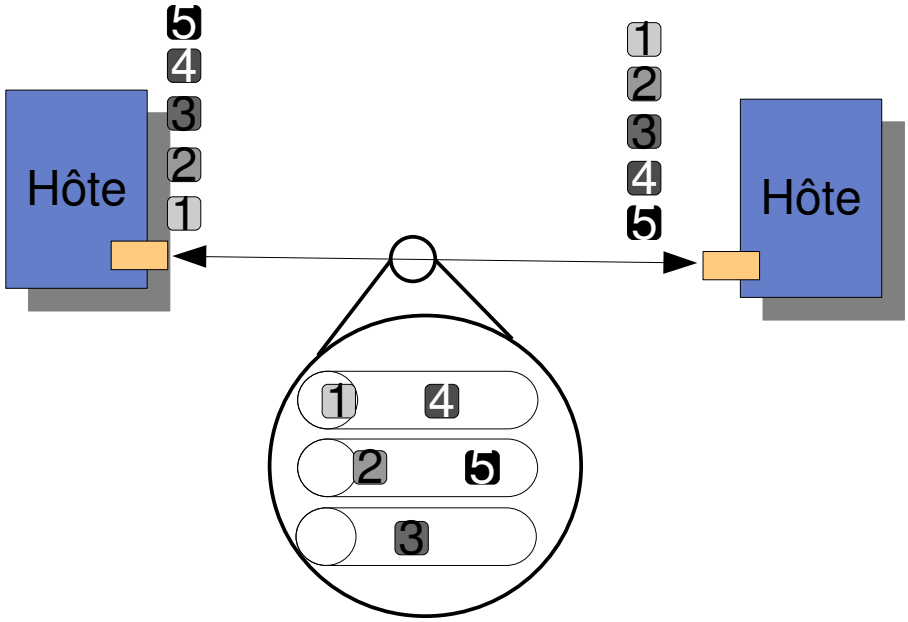


Figure 4 : Flux parallèles

Une connexion logique est composée de plusieurs flux parallèles. Chaque flux reçoit un paquet de façon circulaire (“Round Robin”). Par exemple, dans l'illustration de la Fig.4, la connexion est composée de trois flux, le premier paquet va dans le premier flux, le suivant dans le deuxième et le troisième paquet dans le troisième et dernier flux. Ensuite le quatrième paquet est envoyé de nouveau dans le premier flux et ainsi de suite.

Ces multiples flux parallèles permettent une meilleure utilisation de la bande passante théorique entre les deux hôtes. Mais parfois au prix d'une perte de l'aspect équitable du TCP de base.

3 Multi Interface Transfert Protocol

3.1 Introduction

Multi Interface Transfert Protocol, MITP, est le protocole d'application et l'approche présentée dans ce document pour permettre l'utilisation de plusieurs interfaces d'un même hôte en vue de télécharger des données sur un serveur.

Avant d'expliquer la solution que nous proposons ici pour le téléchargement multi-interfaces, il est nécessaire de présenter les différents modes de transfert de données entre deux hôtes et leurs principales caractéristiques. Par ailleurs, nous verrons comment notre solution s'inspire d'idées déjà adoptées et appliquées en partie.

3.2 Modes de transfert

Etablissons une classification des modes de transfert entre hôtes en fonction du nombre de participants au transfert.

3.2.1 Mode “un-à-un”

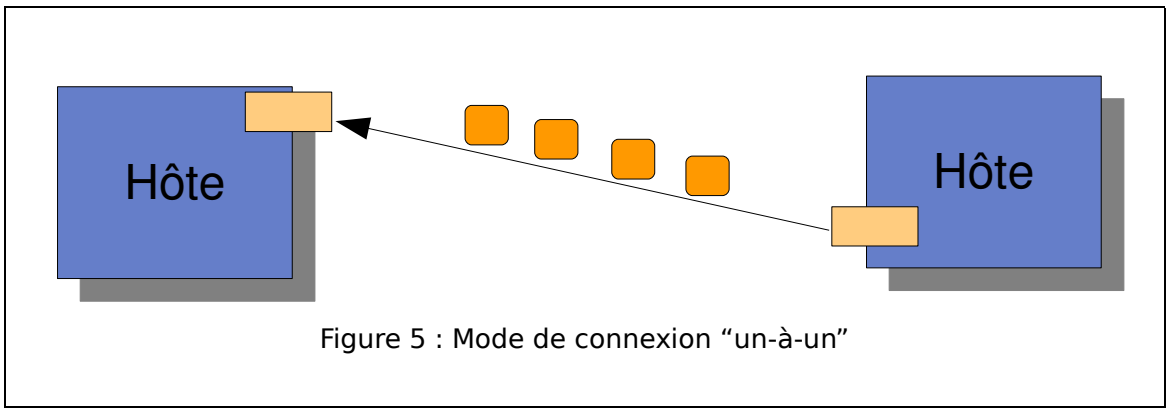


Figure 5 : Mode de connexion “un-à-un”

A l'heure actuelle, le mode “un-à-un”, un émetteur et un récepteur, est le mode le plus usité pour les connexions entre hôtes. Un grand nombre de protocoles se basent sur ce mode. Un dialogue s'établit entre l'émetteur et le récepteur et les données sont envoyées. Le point de contact utilisé par les

hôtes consiste en une et une seule interface. Toutes les données transmises vers le récepteur proviennent du même émetteur pour le transfert complet.

3.2.2 Mode “Plusieurs-à-un”

Ici, il y a plusieurs émetteurs et un récepteur. A la différence du mode “un-à-un”, les données proviennent de plusieurs émetteurs. Ce fonctionnement n'est possible que dans le cas où la source des données est partagée entre plusieurs émetteurs possibles. Par exemple, si le récepteur souhaite télécharger un fichier et qu'il est disponible chez plusieurs émetteurs, il peut en télécharger des fractions différentes chez chacun. Cela peut améliorer le taux et le temps de transfert. Par contre, le récepteur utilise la même interface pour dialoguer avec tous les émetteurs. Il faut aussi que le protocole utilisé pour le transfert des données supporte la division du contenu à un niveau “sub-atomique”, c'est à dire que le contenu puisse être divisé en plusieurs fractions indépendantes. C'est le cas par exemple pour le FTP ou HTTP, mais pas pour le NNTP. En effet il n'est pas possible de télécharger un message[1] en plusieurs parties distinctes, même s'il est présent sur plusieurs serveurs différents. Nous reprendrons plus loin cette idée de division du contenu à transférer en plusieurs fractions.

Par ailleurs, on peut aussi envisager le mode “plusieurs-à-un” avec un seul émetteur et plusieurs récepteurs simultanés. C'est le concept de “multicast” développé pour les réseaux IP.

[1]Le terme “message” est à comprendre dans le sens d'article Usenet transféré par NNTP

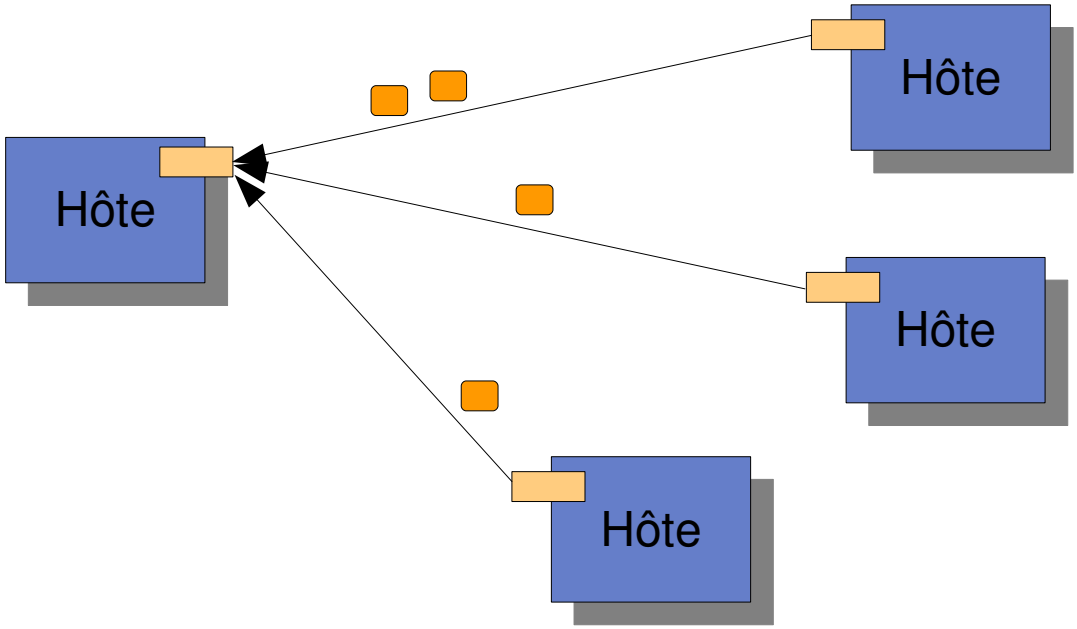


Figure 6 : Mode de connexion “Plusieurs-à-un”

3.2.3 Mode “Plusieurs-à-plusieurs”

Dans ce cas de figure, plusieurs émetteurs et plusieurs récepteurs sont impliqués. Ce système est exploité dans les applications et les protocoles P2P (Peer-to-Peer) [P2P1]. Le principe général est que plusieurs émetteurs disposent du contenu à transférer et plusieurs récepteurs téléchargent ce contenu. Quand un récepteur a terminé de télécharger le contenu, il devient à son tour émetteur.

Les protocoles et réseaux P2P impliquent que chaque fichier échangé peut être divisé par le protocole et les applications en plusieurs fractions indépendantes échangées au fur et à mesure de leur disponibilité sur les récepteurs (voir division sub-atomique de la page précédente). Les hôtes communicants sont donc à la fois récepteurs de contenu et émetteurs de contenu. Comme le système abandonne la dichotomie générale entre serveurs (émetteurs) et clients (récepteurs) dans la relation de transfert de contenu, les

hôtes participant à ce type d'échange sont donc des “pairs” au sens “d'égaux entre eux”, d'où l'acronyme P2P Peer-to-Peer – Pair-à-Pair. Les réseaux P2P présentent un certain nombre d'avantages, la communautarisation des ressources étant le principal, mais ils ont aussi des inconvénients. Une discussion plus approfondie sur les réseaux et protocoles P2P sort toutefois du cadre de ce mémoire.

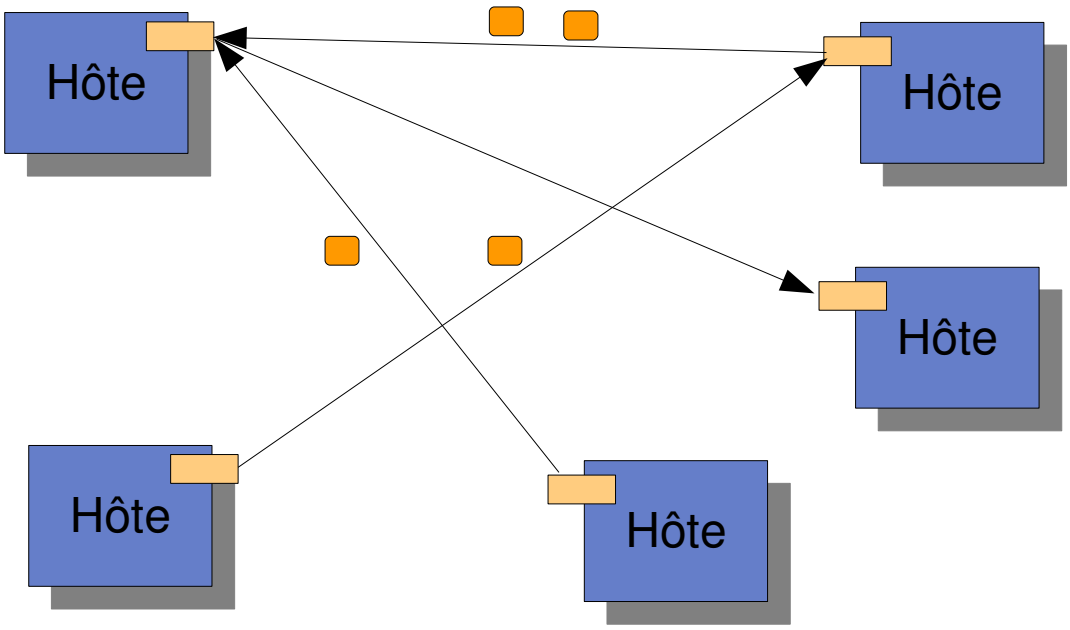


Figure 7 : Mode de connexion “Plusieurs-à-plusieurs”

Pour plus de détails, le lecteur intéressé peut se référer au groupe de travail Internet2 s'occupant du sujet [P2P2].

3.3 Principes de base

Les deux idées de base de MITP sont :

1. La division du contenu source en fractions indépendantes au niveau applicatif.
2. L'utilisation de plusieurs chemins (via des interfaces distinctes) pour accéder au contenu source.

Ces deux principes sont indissociables. En effet, il n'est pas intéressant de faire emprunter plusieurs chemins à un ensemble indivisible, sauf pour obtenir une meilleure résistance aux erreurs. Réciproquement, diviser en plusieurs fractions logiques un contenu qui sera transféré par le même canal a peu d'intérêt d'un point de vue applicatif. Rappelons que les réseaux basés sur la commutation de paquets divisent déjà les informations à transférer en plusieurs petites unités (“Protocol Data Unit”) pour leur faire emprunter le canal de communication. Mais cette division étant effectuée par les protocoles de bas niveau, les applications manipulent des unités de données indépendantes de la division par les couches inférieures.

3.3.1 Division du contenu

Comme dit plus haut, l'idée de diviser le contenu source existe dans d'autres protocoles. Cela permet de reprendre, entre autres, un téléchargement interrompu. Avec une division du contenu en fractions indépendantes, il est possible de les répartir via l'une ou l'autre interface, ce qui ouvre un certain nombre de possibilités et d'avantages que nous détaillerons plus loin.

La politique d'assignation des fractions du contenu à télécharger est définissable par le client. On peut imaginer une répartition équitable entre les interfaces, une répartition aléatoire, une répartition non fixée (chaque interface prenant une nouvelle fraction à chaque fois qu'elle a fini la précédente), ou alors une métrique de répartition en fonction du coût de chaque chemin depuis l'interface ou de la vitesse de l'interface.

Les figures 8 à 11 illustrent graphiquement les principales politiques.

- Répartition non fixée (Fig.8) : chaque interface consomme une fraction dès qu'elle a fini la précédente, jusqu'à ce que toutes les fractions aient été téléchargées.
- Répartition fixée (Fig.9) : chaque interface se voit attribuer un sous-ensemble des fractions à télécharger.
- Répartition dynamique (Fig.10) : en présence d'un chemin plus rapide que l'autre, la première interface consomme toutes les fractions restantes avant que la deuxième n'ait pu terminer le téléchargement de sa fraction initiale.

- Répartition dynamique (Fig.11) : si un chemin devient inaccessible (dans cet exemple, le deuxième), la première interface reprend la fraction non-terminée et poursuit avec le reste des fractions jusqu'à la fin.

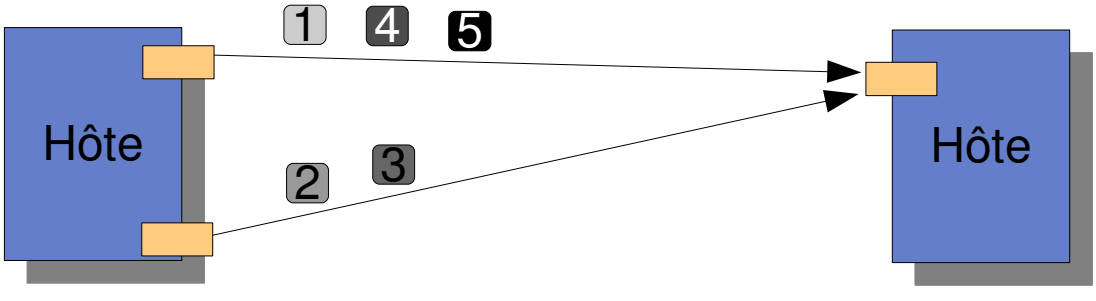


Figure 8 : Répartition des fractions non fixée

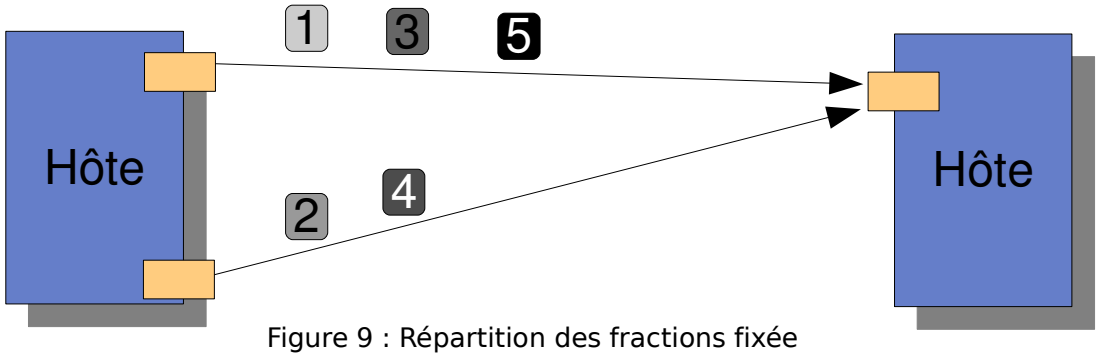
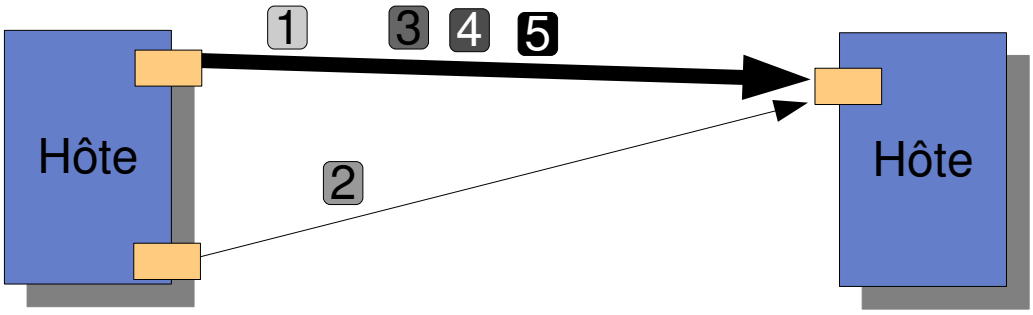


Figure 9 : Répartition des fractions fixée



Firgure 10 : Répartition des fractions non fixée, différentes vitesses d'interfaces

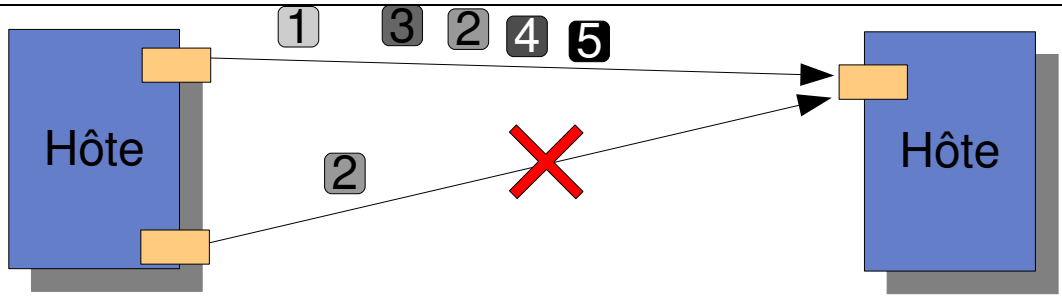


Figure 11 : Répartition des fractions, échec d'une interface

3.3.2 Utilisation des interfaces

MITP utilise plusieurs connexions à l'hôte source via plusieurs interfaces de l'hôte récepteur. On peut ainsi combiner les ressources et la bande passante des différentes interfaces. Chaque connexion vers l'émetteur pour transférer une fraction est indépendante et s'effectue en fonction de la politique d'assignation des fractions aux interfaces définie par l'utilisateur.

3.4 Protocole

Les fonctionnalités nécessaires pour une implémentation de ces deux principes n'étant présentes dans aucun protocole actuel, nous avons décidé de créer un protocole ad-hoc pour la démonstration par l'exemple de la faisabilité des idées présentées ci-dessus. Ce protocole, MITP, possède trois phases :

- Initialisation
- Transfert
- Clôture

Chaque phase correspond à l'envoi d'une ou plusieurs “requêtes texte” du client vers le serveur via une connexion TCP. Les requêtes ont toutes la forme :

“méthode ou mot clé” “paramètre1” “paramètre2” ... “paramètreN”

Dans les descriptions suivantes, les chaînes de caractères entre chevrons dénotent une variable paramétrisable.

- Initialisation :

Le récepteur se connecte à l'émetteur via une des interfaces disponibles (configurables) pour demander des informations sur le fichier (taille ...) et négocier les paramètres de connexion.

  - Récepteur :

  INFG <fichier demandé> <taille d'une fraction élémentaire>

  La première requête vers le serveur consiste à négocier le transfert d'un fichier en spécifiant la taille voulue des fractions logiques que l'on va recevoir par la suite.

  - Emetteur :

  Le serveur va répondre à cette requête d'initialisation soit avec un message d'erreur, soit avec une série d'informations sur le contenu à transférer, à savoir :

    - La taille totale du contenu
    - Le nombre de fractions (en fonction de la taille demandée par le client)
    - Un checksum de contrôle sur le fichier complet

INFS OK “taille du fichier” “nombre de fractions” “checksum fichier”

En cas d'erreur, le serveur répondra par un simple message avec un code spécifiant le type d'erreur s'étant produite :

INFS KO “code d'erreur”

- Transfert :

Une fois la phase de négociation terminée, chaque interface peut se connecter à l'émetteur pour télécharger une fraction du fichier. Les fractions sont accompagnées d'un checksum. Ce checksum vérifie le transfert de chaque fraction sans attendre la fin du téléchargement. Ainsi on peut recommencer le transfert s'il y a eu une erreur ou assigner la fraction à une autre interface au cas où l'on aurait remarqué qu'une interface particulière produit trop d'erreurs.

  - Récepteur :

  GETG <numéro de fraction>

  La requête pour demander une fraction du fichier à télécharger.

  - Emetteur :

  Le serveur va répondre avec un message contenant le checksum et les données correspondantes à la fraction de fichier à transférer.

  GETS “checksum fraction” “data”

- Clôture :

Une fois toutes les fractions correctement téléchargées sur le récepteur, il peut clôturer la connexion et permettre ainsi à l'émetteur de recycler les ressources.

  - Récepteur :

  FING

  - Emetteur :

  FINS

3.5 Fonctions

Les différentes actions de haut niveau à réaliser sont donc :

- Serveur
  - Gérer la connexion et le protocole
  - Diviser le fichier source
- Client
  - Gérer les connexions et le protocole
  - Gérer la politique d'attribution des fractions (fixée ou non) et son paramétrage
  - Réceptionner les fractions et les vérifier
  - Réassembler les fractions en un fichier final et le vérifier

Chaque action de haut niveau correspond à un module différent. On aura donc un module serveur principal accompagné d'un module d'abstraction du fichier/fractions. Pour le client, il faudra un module de gestion générale et une abstraction pour les connexions ainsi qu'un module de gestion des fractions et d'attribution/répartition des fractions.

3.5.1 Serveur

Le serveur doit être capable de comprendre les requêtes provenant des clients et de les transformer en demandes pour le module de gestion de fichiers. Ensuite si nécessaire, il faut renvoyer la fraction demandée au client via la connexion.

Détaillons les fonctions par modules

- Module principal
  - Accepter les connexions
  - Traiter les requêtes
    - Initialisation
    - Demande de fraction
    - Clôture
  - Terminer les connexions

- Interagir avec le module de gestion de fichier source

- Module fichier source
  - Gérer l'abstraction de fichier du système d'exploitation
  - Lire et transmettre les fractions de fichier au module principal

Graphiquement

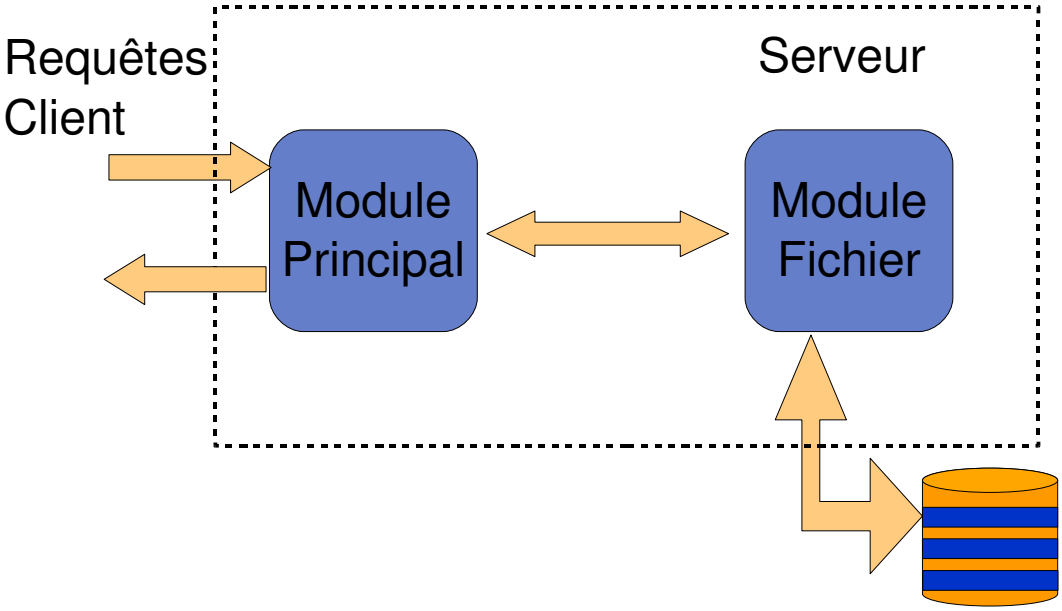


Figure 12 : Fonctions serveur

3.5.2 Client

Le client doit établir les connexions vers le serveur pour initialiser le téléchargement et ensuite demander les différentes fractions du fichier à télécharger en fonction de la politique de répartition définie par l'utilisateur. Enfin, il faut clôturer les connexions et vérifier le téléchargement.

Définition des fonctions par module :

- Module principal
  - Gestion du protocole
  - Initialisation
  - Demande de fraction
  - Clôture
- Module connexions
  - Etablissement
  - Vérification
  - Fin
- Module parties
  - Configuration de la politique
  - Gestion des attributions
- Module fichier
  - Réception des fractions
  - Réassemblage final

Graphiquement

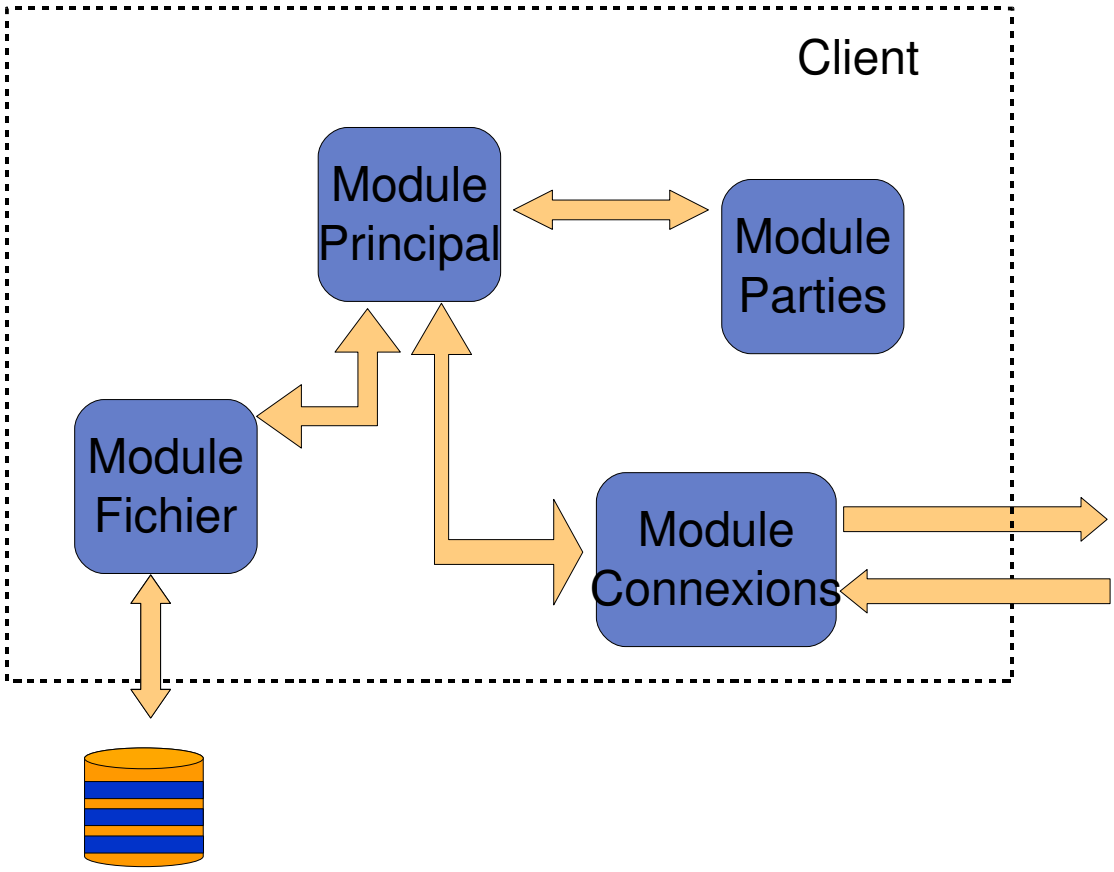


Figure 13 : Fonctions client

3.6 Exemple de session MITP

Client Serveur

1. Connexion au serveur via un socket
2. Demande d'informations sur le fichier (INFG)
3. Réponse du serveur (INFS)
4. Division du fichier en fractions, sur base de la taille négociée
5. “Tant que” toutes les fractions n'ont pas été téléchargées
6. Demander la prochaine fraction à télécharger au gestionnaire d'attribution
7. Connexion au serveur via un socket de l'interface attribuée
8. Demande de la fraction (GETG)
9. Demande de la fraction au gestionnaire de fichier
10. Réponse avec la fraction (GETS)
11. Ecriture de la fraction téléchargée vers le gestionnaire de fractions
12. Fin “Tant que”
13. Demande de clôture
14. Clôture (FING)
15. Clôture (FINS)
16. Ecriture du fichier et vérification

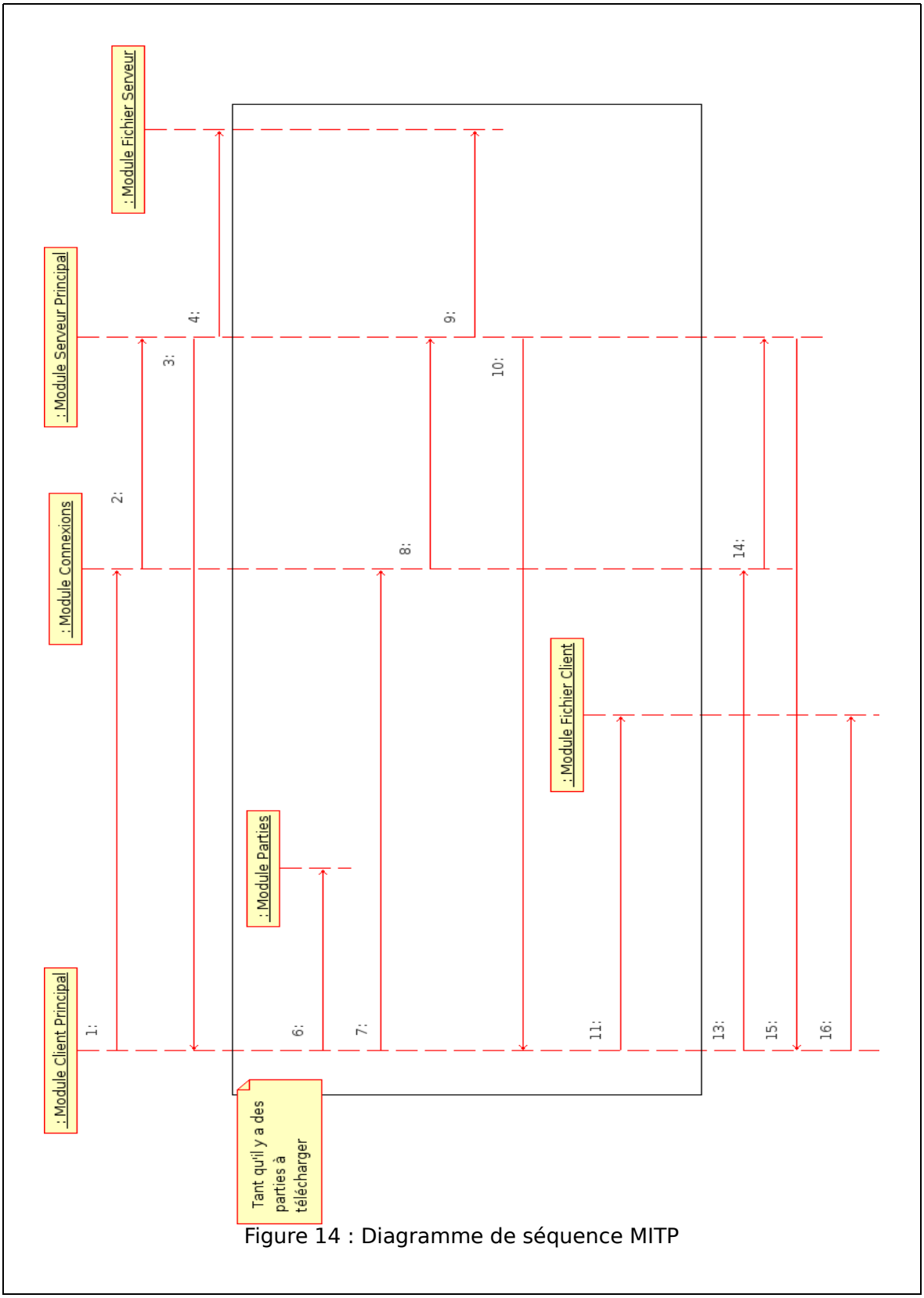


Figure 14 : Diagramme de séquence MITP

3.7 Généralisation HTTP

Comme décrit précédemment, un des désavantages de la solution proposée est l'utilisation d'un protocole ad hoc et la nécessité de faire dialoguer le client avec un serveur comprenant ce protocole. Cette partie-ci du travail montre comment il est possible de transplanter les idées de division du contenu et d'utilisation de plusieurs interfaces à un protocole plus général.

Le protocole HTTP (HyperText Transfert Protocole) est le protocole standard de transfert d'informations sur le World Wide Web, utilisé entre autres pour consulter les pages web. Par la suite ce protocole a été repris comme support pour toute une série d'autres protocoles de plus haut niveau qui encapsulent leurs messages dans des requêtes HTTP (SOAP par exemple).

HTTP fonctionne sur base d'une série de requêtes/réponses standardisées entre un client et un serveur. La principale requête pour rapatrier du contenu est la requête “GET” dont voici un exemple simple :

```
GET /index.html HTTP/1.1
Host : www.example.org
```

Le format général des requêtes étant :

```
<method> <url> <version>
<header1>
<header2>
...
<headerN>
<ligne vide
<corps optionnel>
```

Dans l'exemple, la méthode est “GET”, l'URL “index.html” et la version 1.1 d'HTTP est utilisée. De plus un header “Host” indique sur quel serveur la ressource identifiée par l'URL est demandée.

Un header nous intéresse tout particulièrement. Il s'agit du header “Range” défini par la norme HTTP. Ce header permet d'ajouter à une requête de contenu une ou plusieurs contraintes sur la plage de données que contiendra la réponse. Comme tous les objets transférés par HTTP sont des suites d'octets, le serveur doit pouvoir diviser le contenu demandé selon la ou les contraintes spécifiées dans le header. Le serveur peut aussi ignorer la requête. Les plages de données peuvent être définies en spécifiant une plage complète (de l'octet N à l'octet M) ou en omettant le début ou la fin, dans le sens “tout jusqu'à” ou “tout à partir de”.

Voici le premier exemple modifié pour ne demander en réponse que les 10 premiers bytes du contenu :

```
GET /index.html HTTP/1.1
Host : www.example.org
Range : bytes=0-10
```

En utilisant ce principe, combiné au fait que le protocole HTTP est “stateless” (c'est-à-dire que le serveur ne conserve pas d'état et traite chaque connexion indépendamment de la suivante), nous sommes capables d'utiliser le protocole HTTP pour recréer le schéma MITP.

La phase d'initialisation est remplacée par une requête HTTP “HEAD” qui donne la taille du contenu. Cette taille permet d'initialiser le client et de diviser le travail entre les interfaces. Les différentes fractions de contenu sont transférées avec des requêtes “GET” jusqu'à la complétion du fichier à télécharger. Notons que le protocole HTTP ne fournit pas de checksum avec chaque réponse. Il n'est pas nécessaire de clôturer la session comme dans MITP parce que la division du fichier du côté serveur est effectué à la volée pour chaque requête. La Fig.15 présente les différentes couches qui sont utilisées :

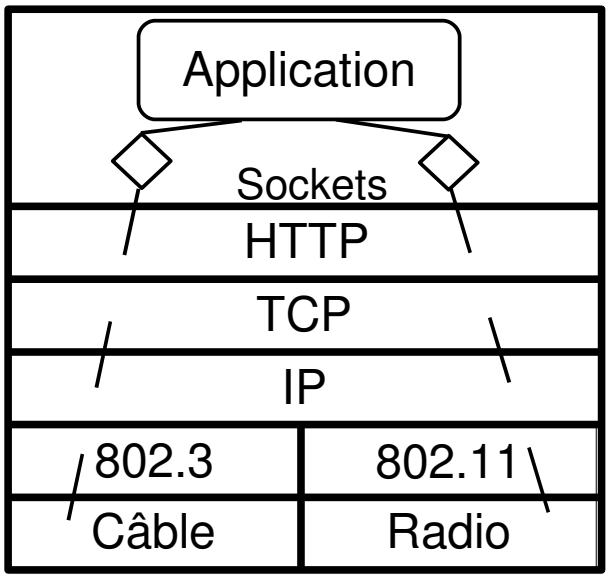


Figure 15 : Modèle généralisation HTTP

Comparons l'exemple de session MITP avec un téléchargement HTTP

Client

Serveur

1. Connexion au serveur via un socket
2. Demande d'informations sur le fichier (requête HEAD)
3. Réponse du serveur
4. “Tant que” toutes les fractions n'ont pas été téléchargées
5. Demander la prochaine fraction à télécharger au gestionnaire d'attribution
6. Réponse avec la fraction à télécharger
7. Connexion au serveur via un socket de l'interface attribuée
8. Demande de la fraction (requête GET avec header Range)
9. Réponse avec la fraction de flux
10. Ecriture de la fraction téléchargée vers le gestionnaire de fractions
11. Fin “Tant que”
12. Ecriture du fichier et vérification

Notons l'absence de clôture à la fin de la session.

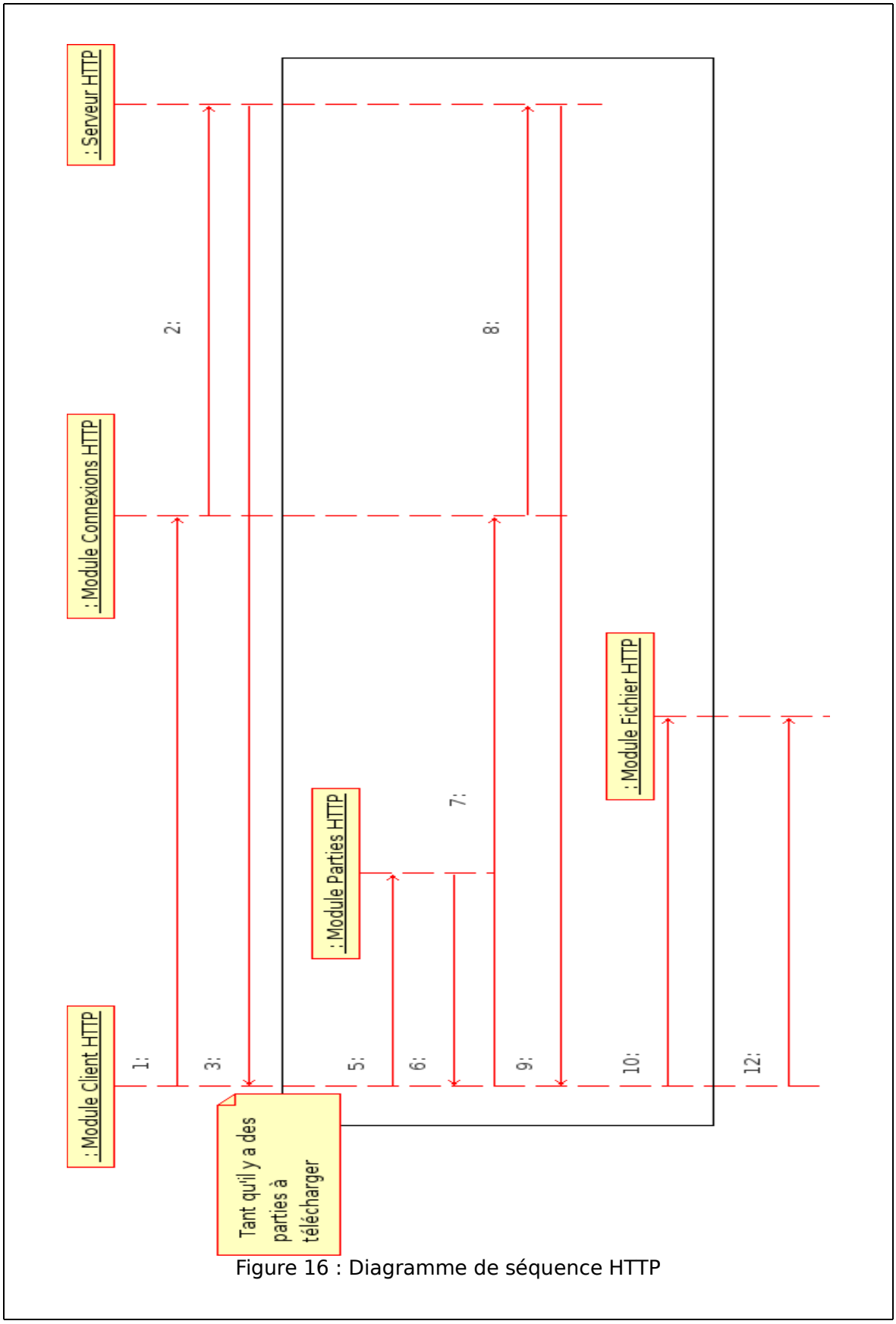


Figure 16 : Diagramme de séquence HTTP

3.8 Analyse critique / Performances

Les principaux avantages de l'utilisation de plusieurs interfaces d'un même hôte pour le téléchargement de données, telle que proposée par ce travail, sont :

- Une meilleure utilisation de la capacité en terme de bande passante des différentes interfaces. En regroupant plusieurs interfaces, on additione la bande passante de chacune d'elles pour former une sorte de pseudo-interface dont la vitesse théorique combinée est celle de l'ensemble des interfaces qui la composent.

- L'autre avantage est la résistance aux pannes d'une interface particulière. Si une interface défaille, les autres peuvent reprendre le travail sans qu'une intervention de l'utilisateur soit nécessaire et ainsi poursuivre le téléchargement jusqu'à sa complétion.

- Enfin, grâce à la possibilité de définir une politique de gestion des interfaces, une gestion fine et précise de la charge de chaque interface peut être mise en place en fonction des contraintes propres aux réseaux et à l'environement dans lequel elles évoluent.

Du côté des désavantages, citons :

- Dans le démonstrateur proposé, la complexité de gestion des différentes interfaces se situe au niveau de l'application. Il faut tenir compte, à l'instar de la programmation multi-thread par rapport à la programmation mono-thread, des problèmes généraux liés à la programmation concurrente, par exemple des accès multiples à des données partagées, la synchronisation entre les différentes parties, ...

- Par ailleurs, l'utilisation de plusieurs interfaces sur les protocoles de transport existant (point-à-point et mono-interface) implique que les protocoles d'application qui seront utilisés ne doivent pas comporter de contrôle “out-of-band” ou d'authentification. En effet, une fois le téléchargement commencé, les requêtes peuvent provenir d'une autre interface que celle qui a initié le téléchargement.

4 Démonstration

Cette partie présente différents scénarios d'utilisation du démonstrateur et une comparaison avec une technologie concurrente (SCTP) afin d'illustrer le fonctionnement des idées proposées par ce mémoire. Les résultats seront présentés sous forme de captures de paquets réalisées sur un réseau de test décrit ci-dessous. Ces captures de paquets ont été faites avec le programme d'analyse de réseau informatique Ethereal. Sauf mention contraire, les captures ont été effectuées sur le client.

4.1 Environnement de test

Les scénarios d'utilisation ont été joués sur un réseau informatique réel basé sur la technologie ethernet filaire et sans-fil (voir Fig.17 pour le schéma). Le client possède deux interfaces réseau, une sur chaque technologie d'accès différente. Chaque interface possède sa propre adresse IP et est reliée à son réseau d'accès particulier. Les paquets transitant par le réseau sans-fil sont traités par un access point pour être retransmis sur le réseau filaire. Le serveur, lui, est connecté au réseau filaire. Le serveur n'utilise qu'une seule interface pour recevoir les connexions. Le type de transfert est donc “plusieurs-à-un” tel qu'exposé dans la section 3.2

Liste des adresses IP participant aux captures :

IP1 : 172.16.100.193 ou 172.16.28.60 (Client – Ethernet)

IP2 : 172.22.252.249 (Client – Sans fil)

IP3 : 172.16.20.14 (Serveur – Ethernet)

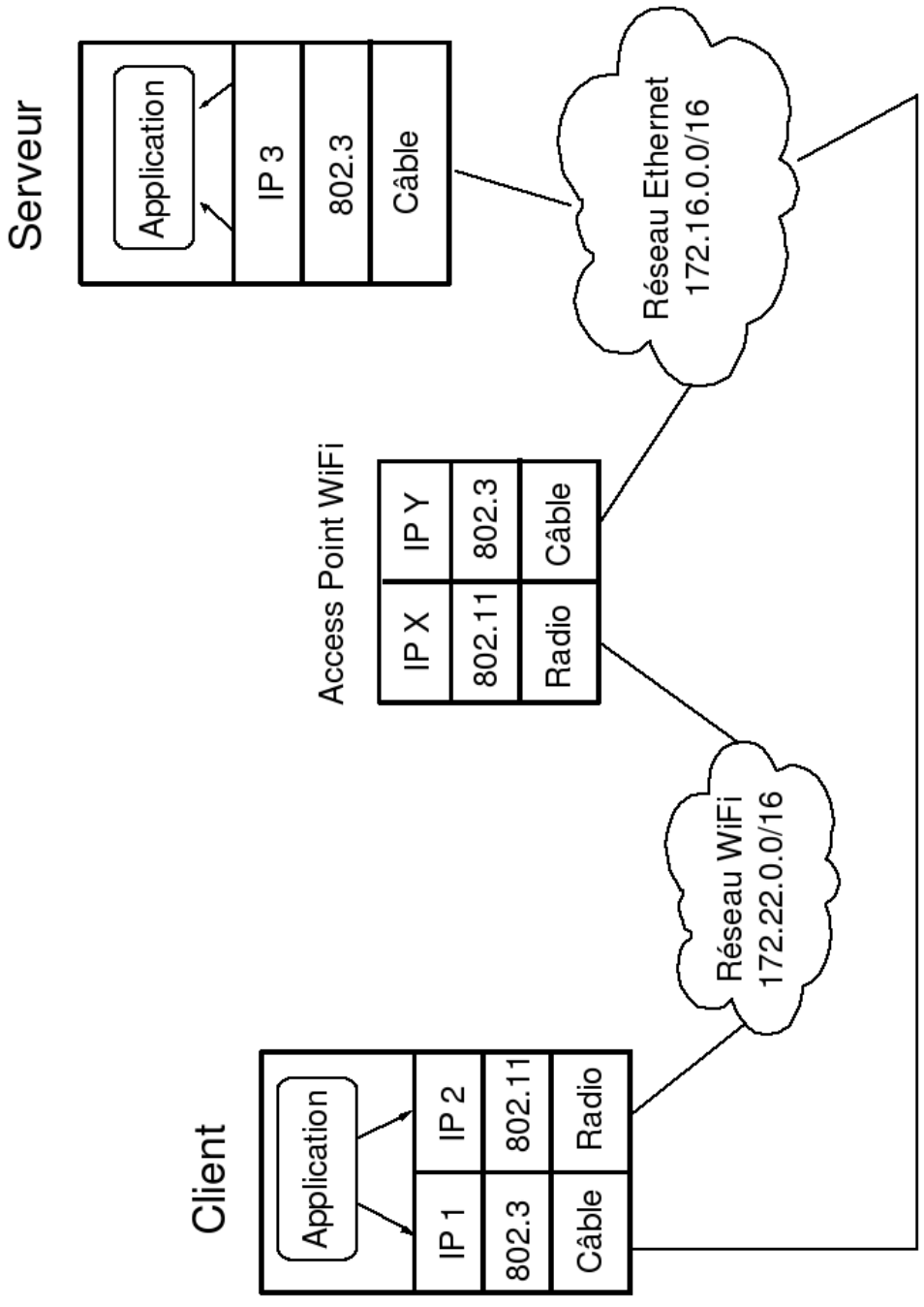


Figure 17 : Environnement de test

4.2 Scénarios

Pour illustrer les scénarios, certains paquets capturés seront présentés avec leur instant de capture, leur numéro d'ordre, leur source, leur destination et les données qu'ils transportaient. L'instant d'arrivée du paquet sur la machine qui réalise la capture est exprimé en secondes écoulées depuis le début de la capture.

Les captures de trafic réseau complètes étant composées d'un grand nombre de paquets, il serait impossible de reproduire complètement l'échange d'informations qui est effectué lors d'un scénario. L'ensemble des paquets capturés lors des scénarios peuvent être consultés dans les annexes.

4.2.1 Scénario 1

Ce premier scénario consiste en une connexion simple et sans erreurs utilisant les deux interfaces et une répartition dynamique des fractions entre elles.

MITP

Temps	Ordre / Total	Source	Destination	Data
1.648130	10 / 1073	172.16.100.193	172.16.20.14	INFG tmp.toto 10000
1.655456	12 / 1073	172.16.20.14	172.16.100.193	INFS OK 0001048576 0105 38f6...
1.693760	17 / 1073	172.16.100.193	172.16.20.14	GETG 1
1.693887	18 / 1073	172.22.252.249	172.16.20.14	GETG 0
2.291522	611 / 1073	172.16.100.193	172.16.20.14	GETG 58
2.302570	631 / 1073	172.22.252.249	172.16.20.14	GETG 60
3.030719	1063 / 1073	172.16.100.193	172.16.20.14	FING
3.031017	1064 / 1073	172.16.20.14	172.16.100.193	FINS

On peut voir la demande de connexion “INFG” avec le nom de fichier et la taille des fractions, la négociation des paramètres avec le serveur, suivie des demandes de fraction “GETG” réparties aléatoirement entre les deux interfaces. Dans ce scénario, il y a 105 fractions à transférer. Les deux interfaces ayant un débit plus ou moins équivalent, la liaison filaire ne prend pas totalement le dessus sur la liaison sans-fil. Le transfert se termine par la demande de fin de connexion “FING”.

SCTP

Temps	Ordre / Total	Source	Destination	Data
0.000000	1 / 122	172.22.252.249	172.16.20.14	INIT
0.001785	3 / 122	172.16.20.14	172.22.252.249	INIT_ACK
4.817109	99 / 122	172.16.20.14	172.22.252.249	DATA
5.804147	117 / 122	172.22.252.249	172.16.20.14	SHUTDOWN
5.807394	121 / 122	172.16.20.14	172.22.252.249	SHUTDOWN_ACK

Dans le cas de SCTP on peut voir qu'une seule interface (celle sans fil) s'occupe du transfert de données et que la deuxième interface n'intervient pas dans ce scénario puisqu'il n'y a pas de coupure du lien sur la première interface.

4.2.2 Scénario 2

Ce deuxième scénario consiste en une connexion simple utilisant les deux interfaces et une répartition fixée des fractions. La première interface transfère la première moitié des fractions, la seconde le reste.

MITP

Grâce aux numéros de fractions dans les requêtes “GETG” l'on note que la répartition est bien telle qu'annoncée. Dans ce scénario, il y a 11 fractions à transférer.

L'ordre des requêtes avec leur source et la fraction téléchargée:

Ordre / Total	Source	fraction
19 / 1037	172.22.252.249	0
20 / 1037	172.16.100.193	5
215 / 1037	172.22.252.249	1
217 / 1037	172.16.100.193	6
404 / 1037	172.22.252.249	2
405 / 1037	172.16.100.193	7
580 / 1037	172.22.252.249	3
611 / 1037	172.16.100.193	8
770 / 1037	172.22.252.249	4
875 / 1037	172.16.100.193	9
976 / 1037	172.16.100.193	10

SCTP

SCTP ne supportant pas de répartition explicite entre les interfaces, ce scénario ne saurait être envisagé.

HTTP

A titre d'illustration de la généralisation envisagée à la section 3.7, ce scénario de téléchargement sera aussi effectué avec un client HTTP utilisant le header “Range”. Le contenu à transférer mesure 20382 bytes et la taille de la fraction élémentaire à été arbitrairement fixée à 1000 bytes.

On peut observer dans les paquets capturés l'initialisation avec une requête “HEAD” suivie de 21 requêtes “GET”. Dans cette implémentation de test, la taille des fractions n'est pas négociée mais fixée par le client.

Ordre / Total	Requête	Header “Range”
4 / 221	HEAD	N/A
11 / 221	GET	0-1000
24 / 221	GET	1001-2000
...	...	...
205 / 221	GET	19001-20000
215 / 221	GET	20001-20382

Scénario 2.1

Ce troisième scénario, très semblable au deuxième, consiste en une connexion simple utilisant les deux interfaces et une répartition fixée des fractions. La répartition est du type alternative, un paquet sur deux est assigné à une interface, l'autre moitié est assignée à la deuxième interface.

MITP

Comme pour le scénario 2, on peut observer, grâce aux numéros de fractions dans les requêtes, que la répartition est bien telle qu'annoncée.

SCTP

SCTP ne supportant pas de répartition explicite entre les interfaces, ce scénario ne peut pas être envisagé.

4.2.3 Scénario 3

Ce quatrième scénario est un peu plus complexe : la connexion ethernet filaire va être volontairement et artificiellement limitée en bande passante de telle façon que le débit de l'interface sans fil soit largement supérieur. La répartition dynamique des fractions devrait donc faire en sorte que la deuxième interface transfère la majorité des fractions.

MITP

On constate que la deuxième interface prend en effet en charge la plupart des fractions. Dans ce scénario, il y a 105 fractions à transférer. La répartition par interface observée est la suivante :

Source : 172.16.100.193	Source : 172.22.252.249
1,2,4,6	0,3,5,7 à 104

Sur le graphe de la Fig.18, on peut voir que la courbe pointillée qui représente la bande passante utilisée par l'interface filaire est limitée après quelques requêtes et que c'est l'interface sans-fil (courbe pleine) qui termine le transfert.

SCTP

SCTP se comporte ici de la même manière que dans le premier scénario. En effet, une limitation de la vitesse n'est pas interprétée comme un erreur, et donc seule une interface est utilisée, sans basculer sur l'interface de secours.

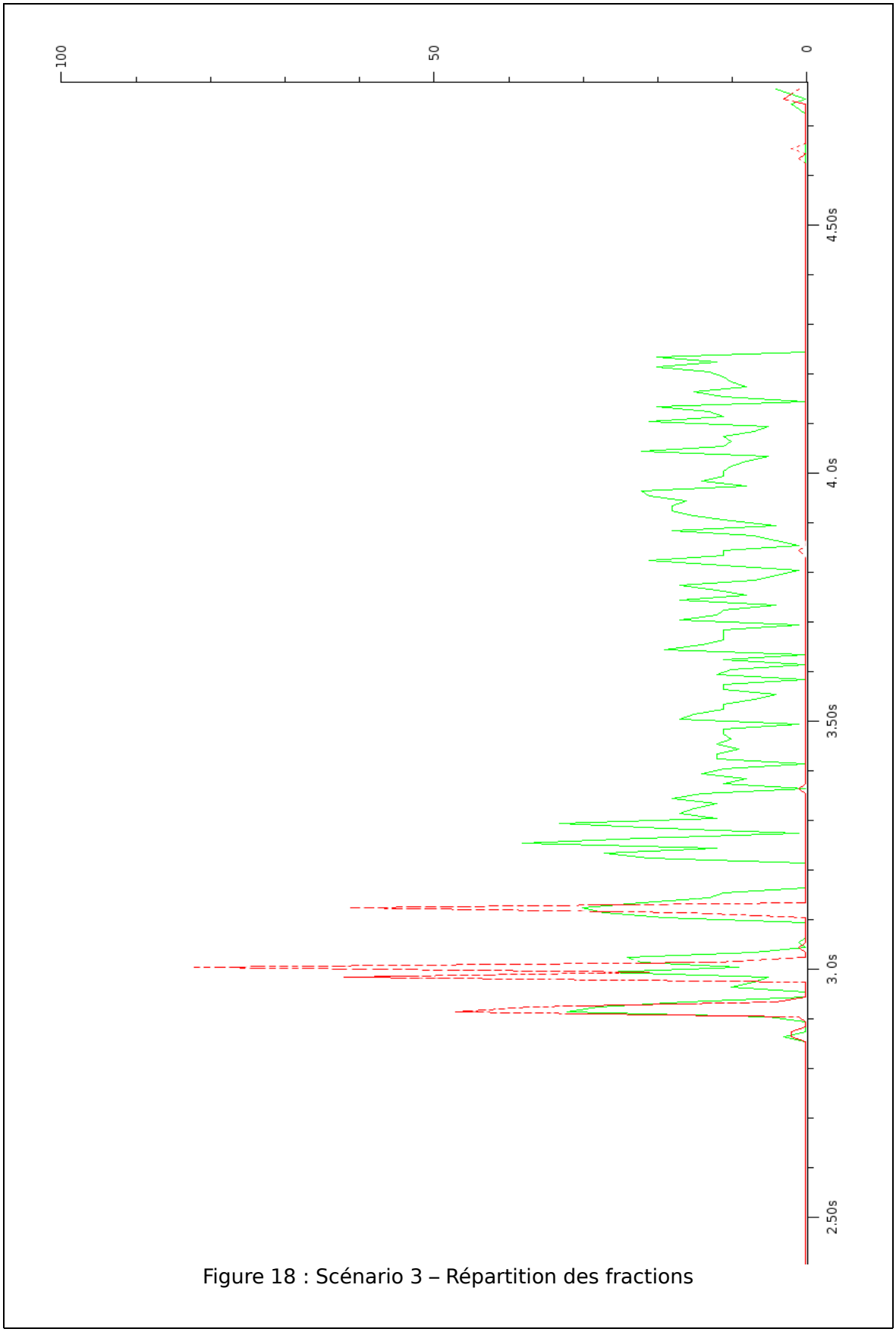


Figure 18 : Scénario 3 – Répartition des fractions

4.2.4 Scénario 4

Ce dernier scénario illustre le comportement des deux protocoles en cas d'erreurs sur une des interfaces. Durant le transfert, on va provoquer une déconnexion de l'interface filaire en débranchant le câble. L'interface sans fil devrait reprendre la connexion et terminer l'envoi des données.

MITP

On peut constater qu'après un certain paquet, il y a une erreur TCP indiquant la perte de connexion sur l'interface filaire. La partie en erreur est donc remise dans l'ensemble des parties téléchargeables et l'interface sans fil la prend en charge. Elle termine ensuite le reste des parties et clôture la connexion.

SCTP

Temps	Ordre / Total	Source	Destination	Data
8.063932	10 / 246	172.16.28.60	172.16.20.14	INIT
8.064169	11 / 246	172.16.20.14	172.16.28.60	INIT_ACK
9.464083	47 / 246	172.16.28.60	172.16.20.14	DATA
76.726405	102 / 246	172.16.20.14	172.22.252.249	HEARTBEAT
76.726448	103 / 246	172.22.252.249	172.16.20.14	HEARTBEAT_ACK
81.065365	179 / 246	172.22.252.249	172.16.20.14	DATA
84.668273	244 / 246	172.22.252.249	172.16.20.14	SHUTDOWN
84.669106	245 / 246	172.16.20.14	172.22.252.249	SHUTDOWN_ACK

On peut observer que le protocole réagit à la perte de connexion sur l'interface principale et termine le transfert via l'autre interface. Un fois l'erreur détectée, SCTP vérifie que l'autre interface de l'association SCTP fonctionne avec la requête “HEARTBEAT”. Notons, grâce aux temps de réception des paquets, qu'il y a un certain délai de réaction avant de basculer sur l'autre interface.

Sur la Figure 19, on peut noter, comme pour MITP dans le scénario 4, que le trafic en trait pointillé s'arrête et que le trafic en trait plein prend la main.

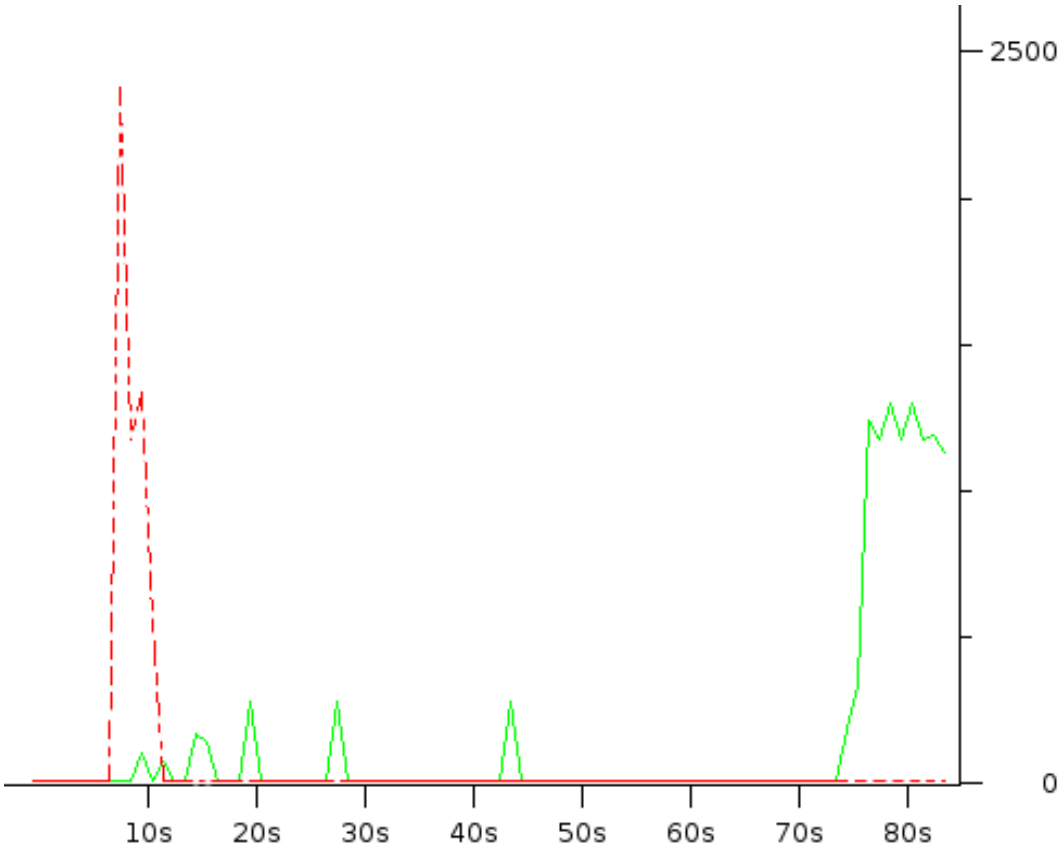


Figure 19 : Scénario 4 – SCTP – Failover

5 Conclusion

Ce travail s'est efforcé de démontrer l'intérêt de l'utilisation de plusieurs interfaces réseaux pour transférer du contenu entre deux hôtes d'un réseau informatique. L'introduction a rappellé que le concept même de l'utilisation de plusieurs interfaces n'était pas très répandu, malgré les avantages et une “applicabilité” théorique importante aux systèmes informatiques existants.

La présentation de l'état de l'art a montré qu'il existe d'autres approches et solutions dans le domaine de l'utilisation simultanée de multiples interfaces, comme SCTP, GridTCP et pTCP. Mais leur principal inconvénient est le manque de contrôle fin sur l'utilisation des interfaces. Cette absence de capacité à définir une politique avancée de gestion des interfaces pose problème pour des scénarios d'utilisation plus complexes. Néanmoins les idées de base des solutions existantes sont exploitées et affinées dans la solution préconisée par ce mémoire.

Après un bref rappel des modes de transfert entre hôtes sur un réseau informatique, nous avons développé les deux idées de la méthode proposée : la division en fractions du contenu à transférer et l'utilisation de plusieurs interfaces selon une politique définissable par l'utilisateur.

Afin d'illustrer l'applicabilité de la méthode proposée, nous avons créé une application de démonstration basée sur un nouveau protocole appellé MITP. Ce protocole est composé de plusieurs requêtes simples pour l'initialisation de la connexion, la définition des paramètres et le transfert du contenu. Nous avons ensuite proposé une généralisation des principes mis en oeuvre dans MITP au protocole HTTP. Elle témoigne du fait que ces idées pourraient être appliquées plus généralement à des protocoles supportant les requêtes de contenus partiels et qui ne s'appuient pas, pour fonctionner correctement, soit sur un contrôle “out-of-band” soit sur une procédure d'authentification.

Ensuite, la présentation de différents scénarios de transfert, à la fois avec MITP et une technologie existante, a permis de valider les idées proposées dans ce texte et de constater les déficiences annoncées des technologies actuelles du point de vue de la gestion de la politique d'attribution des fractions aux interfaces.

5 Conclusion

Pour conclure, si cette recherche permet de constater les avantages de l'utilisation du transfert multi-interfaces elle met aussi en lumière la complexité accrue placée dans l'application, en comparaison des méthodes classiques de transfert point-à-point.

6 Glossaire

Checksum

Un checksum, somme de contrôle en Français, est une technique de vérification de l'intégrité d'une donnée.

Ethernet

Ethernet est une famille de technologies utilisées pour les couches de bas niveau de la pile des protocole Internet. Ethernet est principalement utilisé pour les réseaux locaux.

FTP

File Transfer Protocol. FTP est un protocol applicatif permettant de transférer et de manipuler (copie, effacement, permissions, ...) des fichiers entre deux hôtes connectés à un réseau TCP/IP et qui possédent les applications adéquates.

GPRS

General Packet Radio Service est un service qui peut être utilisé par les réseaux GSM pour transférer des informations autres que la voix et qui utilise le modèle de commutation de paquets plutôt que la commutation de circuits.

HTTP

Hypertext Transfer Protocol. HTTP est un protocole applicatif utilisé pour transférer de l'information sur le World Wide Web. Les navigateurs tels qu'Internet Explorer ou Mozilla Firefox utilisent HTTP pour télécharger les données et les présenter à l'utilisateur.

IETF

Internet Engineering Task Force. L'IETF développe et définit les standards Internet en collaboration avec le W3C et l'ISO. L'IETF est composé uniquement de volontaires et ses membres n'ont pas de statut formel.

6 Glossaire

IP

Internet Protocol. IP est le protocole de base de la couche réseau dans la pile des protocoles Internet. IP assure le routage des paquets dans ce type de réseau.

MITP

Multi Interface Transfer Protocol. MITP est le protocole défini dans ce travail. Il permet l'utilisation de plusieurs interfaces d'un même hôte pour transférer de l'information.

Multicast

Multicast est une technique de dissémination d'information sur réseau informatique. L'émetteur ne doit envoyer qu'une seule fois l'information et celle-ci est alors propagée par les intermédiaires jusqu'aux destinataires, quel que soit leur nombre et leur localisation. L'émetteur n'a pas à contacter chaque récepteur individuellement pour lui transmettre les données.

NNTP

Network News Transfer Protocol. NNTP est un protocol applicatif utilisé pour lire et poster des articles sur Usenet ainsi que par les serveurs Usenet pour s'échanger les articles.

Out-of-band

Dans le cadre des réseaux informatiques, Out-of-band désigne les communications de contrôle qui se déroulent dans un canal spécifique et en-dehors d'une connexion déjà établie pour les données utiles.

P2P

Peer-to-Peer. P2P désigne un réseau informatique où la charge en terme de bande passante et de puissance de calcul est répartie entre les hôtes connectés au réseau plutôt que concentrée dans quelques hôtes qui servent le contenu aux autres. Les réseaux P2P abandonnent le modèle traditionnel de distribution de contenu entre client et serveur pour un modèle où tous les participants à un réseau P2P sont susceptibles d'être client ou serveur.

6 Glossaire

Paquet

Dans le contexte des réseaux informatiques, un paquet est une unité indépendante et formatée de données transitant sur un réseau. Un paquet comporte un préambule, un corps et une terminaison.

PSTN

Public Switched Telephone Network. PSTN désigne les réseaux mondiaux publics de communication téléphonique.

RAID

Redudant Array of Inexpensive Drives. RAID désigne l'ensemble des technologies de stockage qui permettent de répliquer une information sur plusieurs disques. Cette réplication permet d'améliorer les performances et/ou la tolérance aux pannes.

RFC

Request for Comments. Les RFC sont des documents publiés par l'Internet Society et qui contiennent des propositions de protocoles, des informations concernant de nouvelles recherches ou des discussions méthodologiques liées aux technologies mises en oeuvre sur Internet. La plupart des standards utilisés actuellement sur Internet sont définis par un RFC.

SCTP

Stream Control Transmission Protocol. SCTP est un protocole de la couche transport dans la pile des protocoles Internet. SCTP propose la plupart des fonctionnalités de TCP en ce qui concerne les garanties de transmissions. SCTP a été spécifiquement conçu pour le transport des données de réseau PSTN sur réseau IP, fonction pour laquelle TCP présentait des lacunes.

SIGTRAN

SIGTRAN est un groupe de travail de l'IETF qui s'occupe de l'élaboration et de la définition des protocoles de transmission des informations de contrôle PSTN sur réseau IP.

SOAP

SOAP est un protocole applicatif permettant l'échange de messages XML entre hôtes sur un réseau informatique. SOAP est un protocol de base dans les applications Web.

6 Glossaire

Striping

Dans le contexte du stockage de données, le striping est la division d'un ensemble de données en plusieurs fractions qui seront stockées sur des supports physiques différents.

TCP

Transmission Control Protocol. TCP est un protocole de base dans la pile des protocoles Internet. TCP est un protocole de la couche transport. Il permet de connecter deux hôtes via un réseau informatique en offrant de multiples garanties sur la transmission correcte des informations. La plupart des applications Internet utilisent TCP comme protocole de transport (HTTP, FTP, Email, ...)

TCP/IP

Transmission Control Protocol/Internet Protocol. TCP/IP désigne les deux protocoles de base qui forment la pile des protocoles Internet et est utilisé comme raccourci de langage pour cette même pile.

UDP

User Datagram Protocol. UDP est un protocole de base dans la pile des protocoles Internet. UDP est un protocole de la couche transport. Il permet à deux hôtes connectés à un réseau IP d'échanger des messages appellés “Datagram”. UDP ne fournit pas les garanties de transmission de TCP et est surtout utilisé pour les applications contraintes par le temps telles que la téléphonie IP ou les jeux.

UML

Unified Modeling Language. UML est un language standardisé de spécification informatique basé sur un ensemble de notations graphiques qui permettent de créér un modèle abstrait d'un objet ou d'un système.

6 Glossaire

Index du glossaire

Index des figures

Bibliographie

[GRIDFTP] : **Data Management and Transfer in High Performance Computational Grid Environments**. B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke. *Parallel Computing Journal*, Vol. 28 (5), May 2002, pp. 749-771.

[HTTP] : **Hypertext Transfer Protocol – HTTP/1.1**, RFC 2616

[P2P1] : **Core Concepts in Peer-to-Peer (P2P) Networking**. D. Schoder and K. Fischbach, *P2P Computing: The Evolution of a Disruptive Technology*, Idea Group Inc, Hershey. Chapitre 1.

[P2P2] : **Internet2 Peer-to-Peer WG**, http://p2p.internet2.edu/, Dernière consultation le 15/08/2007.

[PAT] : **Dynamic Use of Multiple IP Network Interfaces in Mobile Devices for Packet Loss Prevention and Bandwith Enhancement**, Y. Raziq, *US Patent 2007140256*, 21/06/2007.

[PTCP] : **pTCP: An End-to-End Transport Layer Protocol for Striped Connections**. H.-Y. Hsieh and R. Sivakumar. *IEEE International Conference on Network Protocols (ICNP)*, Paris, France, November 2002

[PYTHONHTTP] : **HTTP Protocol Client**. Python Library Reference, Release 2.5, Section 18.7.

[PYTHONSERVER] : **SocketServer -- A framework for network servers**. Python Library Reference, Release 2.5, Section 18.18.

[PYTHONSOCK] : **socket -- Low-level networking interface**. Python Library Reference, Release 2.5, Section 17.2.

[SCTP1] : **Stream Control Transmission Protocol**, RFC 2960

[SCTP2] : **An Introduction to the Stream Control Transmission Protocol**, RFC 3286

[SCTP3] : **Sockets API Extensions for Stream Control Transmission Protocol (SCTP)**. R. Stewart, Q. Xie, L. Yarroll, K. Poon, M. Tuexen. *Transport Area Working Group sctpsocket Internet-Draft*. Version 14.

Bibliographie

[TCPRMP] : **A Roadmap for Transmission Control Protocol (TCP) Specification Documents**, RFC4614