Réseaux

Couche Application

E. Jeandel

1 Généralités

Couche application
- Dernière couche du modèle OSI et TCP/IP
- Échange de *messages* entre processus

Protocole

Un protocole de niveau application doit spécifier
- le type et le contenu des messages échangés
- Les règles déterminant quand ces messages doivent être envoyés/reçus
- Agent utilisateur (*user agent*) : interface entre l'utilisateur et le protocole (ex : navigateur)

Principe client/serveur

En général un protocole applicatif utilisera un système client/serveur
- Le client initie la connexion, demande un service
- Le serveur fournit un service Exemples :
  - Web : navigateur/serveur Web
  - Envoi de mail : MTA qui envoie/MTA qui reçoit

Quelquefois, chaque participant est simultanément client et serveur
- FTP

Pair à Pair
- Simultanément client et serveur
- Le serveur ne sont typiquement pas toujours allumés ;
- Trafic symétrique
- Passage à l'échelle plus aisé

Caractéristiques
- Tolérance aux pertes ou non
- Délais faibles
- Grande Bande passante ou non
  - Application *élastique* : peut utiliser la bande passante si elle en a.

Exemples
- Transfert de fichiers ?
- Multimedia ?
- VoIP ?
- Mail ?
- XDMCP/VNC ?
- Consultation d'une page web ?
- Jeux vidéos ?
- Messagerie instantanée ?

Questions à se poser
- Structuration des données ? (texte (XML ?) /binaire) ?
- Interaction ?
- Erreurs ?

Structuration des données (TCP)
- En TCP, les données ne sont par organisées en paquet
- Nécessité une analyse syntaxique/sémantique
- Le protocole doit être structuré
  - HTTP : texte, une entête par ligne, puis une saut de ligne
  - XMPP (Jabber) : Flux XML : lire jusqu'au tag fermant correspondant
  - Protocoles binaires (par ex) : commencer par un champ indiquant la longueur

Structuration des données (UDP)
- Organisation en paquet
- Protocole non fiable
- L'application doit continuer à fonctionner si certains paquets sont perdus.
  - Multimédia : I-frames et P-frames

Interactions
- Éviter d'avoir trop de messages
  - Le temps d'AR est souvent le facteur limitant
- Mécanisme pour signaler le type de réponse :
  - Code d'erreur (ex : HTTP)
  - Balise spéciale (ex : XMPP)

2 BOOTP/DHCP

BOOTP/DHCP

- Configurer une machine (Adresse IP, masque du réseau, etc)
- Fonctionne en UDP (68/67 pour client/serveur)
- BOOTP : Liste d'adresse statique. Lien vers un fichier de configuration à télécharger
- DHCP : Allocation dynamique : "location" d'une adresse avec un bail qui peut expirer

Principe

Quatre étapes :

- Le client envoie un paquet `DHCPDISCOVER` pour trouver le serveur DHCP en broadcast
  - Peut demander une certaine IP
  - Peut demander des paramètres utiles : routeur, masque, nom de domaine…
- Le serveur trouve une adresse IP pour le client, et lui envoie un paquet `DHCPOFFER`
  - Le serveur utilise le champ `CHADDR` (Client Hardware Address) pour déterminer l'IP
- Le client envoie un `DHCPREQUEST` en *broadcast* pour accepter l'offre
  - Permet de prévenir *tous* les serveurs DHCP
- Finalement le serveur DHCP envoie un `DHCPACK` qui contient tout les paramètres que le client lui a demandé

Principe (Suite)

Principe du bail :

- Le client essaie de renouveler le bail (`DHCPREQUEST`)
- Deux fois : après la moitié du temps et les 7/8 du temps.
- Si le serveur ne répond pas, ou qu'il répond `DHCPNACK`, le client doit recommencer (en broadcast)

Paramètres utiles (RFC 2132)

- *subnet mask*
- *time offset* fuseau horaire (décalage avec UTC)
- *router*
- *domain name server* (DNS)
- *host name*
- *domain name*
- *Default IP TTL*
- *static route*
- *SMTP server*

3 DNS

3.1 Principe

DNS (Domain Name System)

- Donner des noms plutôt que des IPs à des machines
- Organisé sous forme d'arbre
  - Chaque noeud a un nom (ex : univ-mrs.frr)
  - Les feuilles de l'arbre correspondent à une vraie machine
  - Un *domaine* est un sous-arbre

DNS (Organisation)

Division en *zones*

- Une zone est un ensemble de noeuds.
- Chaque zone possède un serveur officiel (*authoritative server*, autorité) pour la zone
- Le serveur officiel gère tous les noms de la zone (il peut ne pas faire partie de la zone)

Règles :

- Tout noeud appartient à une zone
- Aucun noeud n'appartient à plus d'une zone

Résolution DNS

Du point de vue du client :

- Le client contacte un serveur DNS
- Le serveur DNS lui répond
- Protocole DNS
- UDP 53 pour le serveur
- Port source du client variable

Résolution DNS

Du point de vue d'un serveur, trois cas

- Si je suis le serveur officiel de l'adresse, je renvoie la réponse
- S'il s'agit d'une adresse dans un de mes sous-domaines, j'interroge le sous-domaine
- Sinon, j'interroge un serveur *racine*

Le serveur racine connaît, pour chaque domaine de premier niveau (TLD) un serveur officiel à contacter.

Note :

- Il faut que chaque serveur DNS connaisse un moyen d'accéder à un serveur racine
- Voir par exemple `http://www.iana.org/about/popular-links/`

Interrogation

Un serveur A peut interroger un serveur B de deux manières :

- Itérative : A demande à B quel est le prochain serveur à contacter pour me rapprocher du serveur officiel
- Récursive : A demande à B de faire la résolution DNS.

En pratique, les interrogations vers un serveur *racine* sont itératives, les autres récursives.

Exemple

Je (moi) cherche à obtenir l'IP de `www.lif.univ-mrs.fr`

- Ma machine contacte le serveur DNS `147.94.64.180`
- Celui-ci contacte un serveur root, par exemple `192.5.5.241`(`f.root-servers.net`)
- Le serveur root lui dit que pour contacter `.fr`, il faut s'adresser à `192.134.0.129`(`c.nic.fr`)
- Le serveur `147.94.64.180` demande donc à `192.134.0.129` de faire la résolution DNS pour lui
- `192.134.0.129` sait que `univ-mrs.fr` est en particulier géré par `193.51.208.1`(`dns.inria.fr`) et lui demande de travailler
- `193.51.208.1` sait que `lif.univ-mrs.fr` est géré par `139.124.1.1`(`ns1.univmed.fr`) et lui passe donc la requête
- `139.124.1.1` connaît l'IP de `www.lif.univ-mrs.fr`, la transmet à `193.51.208.1` qui la transmet à `192.134.0.129` qui la transmet à `147.94.64.180`, qui enfin le donne à ma machine

Délégation

- En fait, les champs DNS permettant de déléguer ne contiennent pas une adresse IP, mais un nom d'hôte…
- Cercle vicieux ?
- En pratique `dns.inria.fr` connaît (et renvoie) l'IP de `ns1.univmed.fr`
- Une zone connaît l'adresse IP de tous les hôtes auxquelles elles délèguent..

La pratique

- Tout nom de domaine doit être enregistré dans au moins *deux* serveurs
- *primary name server* et *secondary name server*

Système de cache :

- Pour éviter trop de volume d'informations
- TTL (Time to Live) fixé par le serveur officiel
- Flag dans le protocole DNS pour indiquer qu'on est le serveur officiel.

Voir le cours d'Outils de l'Internet pour user et abuser du DNS..

3.2 Contenu

Contenu

Une ligne dans une liste d'enregistrement DNS est de la forme suivante

```
ac.    172800  IN      NS      b.ns13.net.
```

- Nom de domaine
- TTL
- IN en général (pour INternet)
- Type d'enregistrement
- Contenu

Plusieurs types d'enregistrements

- Le DNS ne sert pas qu'à transformer des noms en IP

Types d'enregistrements

- A : Address
- AAAA (IPV6)
- NS : Name Server (Délégation)
- CNAME : Canonical Name
- SOA : Start of Authority
- TXT
- MX : Mail eXchanger
  - Contient un entier spécifiant la priorité

Exemple

```
gmail.com.       86345   IN   SOA    ns1.google.com. dns-admin.google.com.
                                     1433624 21600 3600 1209600 300
gmail.com.       245     IN   TXT    "v=spf1 redirect=_spf.google.com"
gmail.com.       3545    IN   MX     5 gmail-smtp-in.l.google.com.
gmail.com.       3545    IN   MX     10 alt1.gmail-smtp-in.l.google.com.
gmail.com.       3545    IN   MX     20 alt2.gmail-smtp-in.l.google.com.
gmail.com.       245     IN   AAAA   2a00:1450:8002::53
gmail.com.       243     IN   A      209.85.227.19
gmail.com.       243     IN   A      209.85.227.83
gmail.com.       345545  IN   NS     ns1.google.com.
gmail.com.       345545  IN   NS     ns2.google.com.

;; AUTHORITY SECTION:
gmail.com.       345545  IN   NS     ns1.google.com.
gmail.com.       345545  IN   NS     ns2.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.  328073  IN   A      216.239.32.10
ns2.google.com.  328073  IN   A      216.239.34.10
```

3.3 DNS et DHCP

DNS et DHCP

Comment interagissent DNS et DHCP dans un réseau local ?

- Configurer le serveur DHCP pour qu'il prévienne chaque machine du serveur DNS local
- Permettre au serveur DHCP d'ajouter des enregistrements au serveur DNS
- Utilise le protocole DNS, avec une signature chiffrée (voir cours de sécurité)

4 Divers

4.1 Messageria électronique

Messagerie électronique

Plusieurs fonctionnalités, par plusieurs applications

- MUA (Message User Agent) : Client Mail (`mutt`, Thunderbird, etc)
- MDA (Message Delivery Agent) : Dépose dans la boîte aux lettres
- MTA (Message Transfert Agent) : Transfère les messages d'un serveur à un autre

Seule partie intéressante : le MTA

Pour le reste, voir les protocoles POP3 ou IMAP.

SMTP (Simple Mail Transfer Protocol, RFC 821)

- TCP, port 25
- Protocole textuel, facile à comprendre

Si `bob@kelso.info` veut envoyer un mail à `john@dorian.info` :

- Il contacte son serveur SMTP, et lui donne le message à envoyer
  - Le serveur SMTP de `kelso.info` n'accepte que des messages à destination ou en provenance de `kelso.info`
- Celui-ci cherche le serveur SMTP correspondant à `dorian.info`
  - Enregistrement DNS de type `MX`. A défaut, de type `A`.
  - Priorité sur le serveur à contacter.
- Puis lui envoie le message. Le serveur SMTP correspondant à `dorian.info` se charge de place le message dans la bonne boîte.

Le serveur SMTP de `kelso.info` se *charge* du message. Pas besoin de resté connecté en attendant qu'il soit envoyé.

4.2 HTTP

HTTP

- Utilisé pour accéder aux pages Web
- TCP, Port 80
- Protocole textuel
- Entêtes dans les questions et réponses
- Utilisation de codes d'erreur

Exemple

```
GET /~ejeandel/index.html HTTP/1.1
Host: www.lif.univ-mrs.fr

→

HTTP/1.1 200 OK
Date: Sat, 20 Nov 2010 15:02:59 GMT
Server: Apache
Last-Modified: Mon, 25 Oct 2010 14:17:59 GMT
ETag: "28c6d9d7-1014-4cc59197"
Accept-Ranges: bytes
Content-Length: 4116
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
...
```

Principes

- Au début, une connexion par fichier à télécharger
  - Ralentissement dû à la phase de connexion et au Slow Start de TCP
- 3 solutions :
  - `Connection: Keep-Alive`
  - Pipelining : Envoyer les requêtes sans attendre les réponses
  - Plusieurs connexion simultanées sur le serveur

Gérer la surcharge

Eviter de surcharger une machine avec tout le trafic (ex : `google.com`).

Solutions :

- Round-Robin DNS : permuter les enregistrements DNS à chaque requête
  - N'équilibre pas la charge
  - Problème avec le cache
- Reverse proxy. Une machine reçoit les connexions et répartit la charge
  - Transparent pour le client
  - Avec une redirection 30x

Pas uniquement pour HTTP

4.3 Messagerie instantanée

Protocoles

Plein de protocoles, incompatibles

- AIM
- ICQ
- MSN
- XMPP (Jabber, gmail)
- Yahoo ! Messenger

XMPP (Extensible Messaging and Presence Protocol)

- Utilise XML
- TCP, Port 5222
- Décentralisé (pas un seul serveur)
- Adresses en `username@domain`
- Protocole moderne
  - Serveur à contacter pour `bob@kelso.info` : Enregistrement DNS de type `SRV` sur `_xmpp-server._tcp.kelso.info`
- Communication serveur-serveur différente (port 5269)

Principe

Si `bob@kelso.info` veut parler avec `john@dorian.info` :

- Chacun se connecte à son serveur XMPP
- Bob prévient son serveur qu'il veut parler à John
- Le serveur transmet le message à l'autre serveur :
  - Si aucun des deux serveurs ne bloquent l'autre..
- Le serveur transmet le message à John s'il est connecté, et le garde en mémoire pour le livrer plus tard sinon

Exemple

```
<message type='chat' id='1' to='john@dorian.info'>
  <composing xmlns='http://jabber.org/protocol/chatstates'/>
</message>
<message type='chat' id='2' to='john@dorian.info'>
  <active xmlns='http://jabber.org/protocol/chatstates'/>
  <body>
  Bonjour
  </body>
</message>
<message type='chat' id='3' to='john@dorian.info'>
  <active xmlns='http://jabber.org/protocol/chatstates'/><
/message>
```

Exemple (cont'd)

```
<message type='chat' id='1' to='john@dorian.info'
                            from='bob@kelso.info/Office'>
  <composing xmlns='http://jabber.org/protocol/chatstates'/>
</message>
<message type='chat' id='2' to='john@dorian.info'
                            from='bob@kelso.info/Office'>
  <active xmlns='http://jabber.org/protocol/chatstates'/>
  <body>
  Bonjour
  </body>
</message>
<message type='chat' id='3' to='john@dorian.info'
                            na./from='bob@kelso.info/Office'>
  <active xmlns='http://jabber.org/protocol/chatstates'/>
</message>
```