16 TCP – Les protocoles d'application

16.1 Intégration verticale

L'architecture de la famille TCP s'est construite au fur et à mesure de l'introduction de nouveaux protocoles et de la modification, ou de l'abandon, d'anciens protocoles. Certaines décisions, certaines tendances, sont toutefois constantes tout au long de l'évolution de la famille :

- ✧ au-delà de la couche transport, une seule couche, la couche application, embrasse toute la problématique du traitement des informations; il est donc possible d'intégrer plusieurs traitements et d'obtenir des gains de performances de façon analogue, mais sur une plus grande échelle, à celle permise par l'intégration des opérations de transcodage, de compression et de chiffrement au sein de la couche de présentation du modèle OSI.
- ✧ les couches sont constituées de protocoles autonomes, sans éléments communs entre eux (services, interfaces ou autres); il est donc possible de proposer une interface, des services et une mise en oeuvre limité aux seules fonctions offertes par le protocole, sans coût supplémentaire induit par la généralité imposé par un cadre normatif;
- ✧ toute application peut atteindre directement le protocole, la couche, qui l'intéresse; il n'est donc pas nécessaire de descendre la chaîne protocoles ce qui permet un gain de performance ainsi que l'accès des primitives de très bas niveau, même depuis la couche application.

Il en résulte une architecture très différente, généralement plus simple, que celle du modèle OS

La simplicité architecturale est cependant atteinte au prix de difficultés supplémentaires devant être traités au sein de la couche application :

- L’absence de services communs nécessite la reprogrammation de ceux-ci dans chaque protocole en ayant l'usage; il s'ensuit une augmentation de l'effort de mise en oeuvre, des sources d’incohérence, etc.; par exemple, ftp et smtp ont du traiter indépendamment des questions d’encodage et de compression.
- L’absence d’interfaces normalisées pour chacune des couches empêche de substituer facilement l’un pour l’autre deux protocoles de même niveau au service d’un protocole de niveau supérieur; par exemple, deux protocoles de transfert de fichiers ont du être développés selon qu’ils utilisent TCP ou UDP : ftp et tftp.
- L’absence de planification globale a nécessité le développement de protocoles complémentaires pour pallier à plusieurs lacunes, notamment en ce qui concerne l’adressage; pour ne pas compromettre la bases installée des équipements dotés de TCP, ces protocoles ont nécessairement dû être ajoutés au sein de la couche application (en utilisant la possibilité de court-circuiter au besoin la couche transport, voire la couche réseau); il s’ensuit un accroissement considérable des interactions à haut niveau pour permettre la gestion et la configuration des protocoles de bas niveau; par exemple la prolifération des protocoles de supports tels icmp, igmp, bootp, dhcp, dns, etc. qui conditionne d’autres protocoles de haut niveau sans avertissement.

Il en résulte que chacun des protocoles d’application doit compléter, pour lui-même, les services plus ou moins complets normalement fournis par une couche transport : :

- type de service
- qualité du service
- transfert de données
- interface personne-machine
- gestion de la connexion
- expedited delivery
- status reporting
- sécurité

16.2 Définition des services

On retrouve dans l’architecture TCP l’équivalent des points de services et des points de connexions, limités toutefois à la couche transport : il s’agit des ports (TSAP) et des sockets (TCEP).

L’intervalle de définition des numéros de port (typiquement 1..65535) est divisé en deux sous-intervalles : les ports bien connus (typiquement 1..5000) et les ports clients (typiquement 5000..65535). Parmi les

premiers les numéros 1 à 1023 sont réservés par l'IANA (trouver le sens de l'acronyme, voir rfc 1700) et plusieurs sont liés à des protocoles officiels, normalisés. Les ports clients sont alloués à la demande lors du traitement d'une requête de service spécifique.

Chaque socket possède un identificateur unique qui renvoie à une adresse de socket composée de l'adresse IP suffixée du numéro de port. L'indirection supplémentaire mise en oeuvre par l'identificateur permet de modéliser les connexions multiples sur un même socket. La table de correspondance est gérée par le serveur.

Les sockets peuvent recevoir les commandes suivantes :

... voir commandes de socket TA 6.1, p. 487

Il existe aussi un autre mode de gestion des connexions au sein de la famille TCP, les streams. Ils sont adressables de façon analogue aux sockets mais sont dotés d'un jeu de commandes mieux adaptés au traitement transactionnel :

... voir commandes de streams : HA 11.2.1

16.3 Adressage des services

Une requête de service nécessite l'ouverture préalable d'un socket identifié. Celui-ci fait référence à un numéro de socket composé d'une adresse IP (désignant le serveur auquel est adressée la requête) et d'un numéro de port (déterminant la nature de la requête, son protocole), le socket désignant lui-même le point de connexion.

Il faut résoudre deux problèmes :

- ✧ comment déterminer l'adresse IP du serveur auquel on veut adresser la requête;
- ✧ comment obtenir un numéro de port unique.

L'adressage IP est directement lié à la topologie du réseau et aux fonctions d'aiguillage. Il est en grande partie responsable de l'efficacité de l'aiguillage de la famille TCP et un des facteurs du succès de cette famille. Il ne convient cependant pas, en général, aux besoins de la couche application. La relocalisation d'un serveur, la modification de la structure du réseau ou la reconfiguration d'un aiguilleur peuvent induire un changement d'adresse IP nécessaire au respect des contraintes d'aiguillage. Un tel changement ne doit pas avoir d'impact au niveau de la couche application. La situation engendrée est d'autant plus instable, voire chaotique, que le nombre d'équipements du réseau est grand. Un système d'adressage littéral a donc rapidement été proposé, en marge de la couche transport, par le biais du protocole applicatif DNS qui donne accès à un répertoire dynamique permettant d'établir la correspondance entre l'adresse littérale et l'adresse IP. Voilà notre premier problème

résolu (nous étudierons bientôt plus en détails le protocole DNS, voir chapitre 5).

L'obtention d'un numéro de port spécifique est réalisée comme suit. En fonction de la nature du service, le requérant utilise un numéro de port « bien connu » sachant qu'un serveur échantillonne de façon systématique de tous ses ports « bien connus ». À la réception de la requête de service sur le port bien connu, le serveur détermine alors un nouveau numéro de port parmi les ports clients. Il renvoie ensuite une réponse au requérant en indiquant l'identificateur et le numéro de socket. Cette pratique permet au serveur de n'être à l'écoute que des seuls ports actifs (ports bien connus et ports clients alloués).

16.4 Priorisation des services

La couche de transport met à disposition un mécanisme limité de priorisation d'acheminent normalement utilisé par les protocoles d'applications pour transmettre les commandes (information de contrôle) du protocole, particulièrement pour le traitement des situations exceptionnelles et des erreurs. Un champ booléen « urgent data » est présent dans chaque unité de transport et induit le traitement suivant lorsqu'il est activé :

- ✧ émission : arrêt d'accumulation et transmission immédiate de l'unité urgente;
- ✧ réception : arrêt d'accumulation et signalisation (interruption) immédiate du processus désigné par le numéro de socket (ou le numéro de stream); il appartient alors au processus de traiter, ou non, l'interruption et l'unité urgente.; le traitement lui-même n'est pas modélisé au sein de TCP.

16.5 Protocoles complémentaires

Fonctions jugées nécessaires a posteriori.

La rançon du succès : impossibilité de modifier les couches basses sans compromettre la base installée.

adressage :

- ✧ liaison : ARP, RARP
- ✧ réseau : IGMP, BOOTP, DHCP, ICMP
- ✧ transport : DNS

16.5.1 DNS

Initialement, la résolution des adresses littérales en adresses IP était réalisée localement par chaque équipement grâce à un fichier téléchargé sur une base quotidienne depuis le site de coordination d'internet (NIC). Cette approche était condamnée en raison même du succès d'internet. Le protocole DNS est apparu pour résoudre ce problème et a été développé à partir des objectifs suivants (extraits de la RFC) :

> *The primary goal is a consistent name space which will be used for referring to resources. In order to avoid the problems caused by ad hoc encodings, names should not be required to contain network identifiers, addresses, routes, or similar information as part of the name.*
>
> *The sheer size of the database and frequency of updates suggest that it must be maintained in a distributed manner, with local caching to improve performance. Approaches that attempt to collect a consistent copy of the entire database will become more and more expensive and difficult, and hence should be avoided. The same principle holds for the structure of the name space, and in particular mechanisms for creating and deleting names; these should also be distributed.*
>
> *Where there tradeoffs between the cost of acquiring data, the speed of updates, and the accuracy of caches, the source of the data should control the tradeoff.*
>
> *The costs of implementing such a facility dictate that it be generally useful, and not restricted to a single application. We should be able to use names to retrieve host addresses, mailbox data, and other as yet undetermined information. All data associated with a name is tagged with a type, and queries can be limited to a single type.*
>
> *Because we want the name space to be useful in dissimilar networks and applications, we provide the ability to use the same name space with different protocol families or management. For example, host address formats differ between protocols, though all protocols have the notion of address. The DNS tags all data with a class as well as the type, so that we can allow parallel use of different formats for data of type address.*
>
> *We want name server transactions to be independent of the communications system that carries them. Some systems may wish to use datagrams for queries and responses, and only establish virtual circuits for transactions that need the reliability (e.g., database updates, long transactions); other systems will use virtual circuits exclusively.*
>
> *The system should be useful across a wide spectrum of host capabilities. Both personal computers and large timeshared hosts should be able to use the system, though perhaps in different ways.*

La nécessité de répartir à la fois les données et leur gestion sur la base d'une division de l'espace de dénomination s'est donc rapidement imposée. Les critères de division ont cependant été délibérément ignorés afin de garantir une plus grande souplesse. Il est possible d'utiliser des

critères géographiques, politiques, économiques ou toute autre combinaison de critères pourvue qu'elle assure une délégation efficace et non ambiguë de l'inscription et de la mise à jour des noms.

L'espace de dénomination a donc été naturellement organiser en arborescence. Un nœud de l'arborescence est dit domaine principal s'il prend à charge l'homologation des noms sous-jacents (éventuellement par délégation tout en respectant la structure arborescente de l'espace de dénomination). Ainsi, un nom n'appartiendra toujours qu'à un seul domaine principal, même si les informations le concernant peuvent être dupliquées sur plusieurs sites pour des raisons d'efficacité. Un serveur d'homologation est désigné pour chaque domaine principal, ainsi qu'un serveur de relève.

Pour mettre en place ces règles d'exploitation, il a été décidé de répartir les tâches de la façon suivante (extraits de la RFC) :

> *The system administrators provide:*
> *- The definition of zone boundaries .*
> *- Master files of data.*
> *- Updates to master files.*
> *- Statements of the refresh policies desired.*
>
> *The domain system provides:*
> *- Standard formats for resource data.*
> *- Standard methods for querying the database.*
> *- Standard methods for name servers to refresh local data from foreign name servers.*

La mise en oeuvre de l'espace de dénomination fait appel à trois composants (extraits de la RFC) :

> *The DOMAIN NAME SPACE and RESOURCE RECORDS, which are specifications for a tree structured name space and data associated with the names. Conceptually, each node and leaf of the domain name space tree names a set of information, and query operations are attempts to extract specific types of information from a particular set. A query names the domain name of interest and describes the type of resource information that is desired. For example, the Internet uses some of its domain names to identify hosts; queries for address resources return Internet host addresses.*
>
> *NAME SERVERS are server programs which hold information about the domain tree's structure and set information. A name server may cache structure or set information about any part of the domain tree, but in general a particular name server has complete information about a subset of the domain space, and pointers to other name servers that can be used to lead to information from any part of the domain tree. Name servers know the parts of the domain tree for which they have complete information; a name server is said to be an AUTHORITY for these parts of the name space. Authoritative information is organized into units called ZONEs, and these zones can be automatically distributed to the name servers which provide redundant service for the data in a zone.*

> *RESOLVERS are programs that extract information from name servers in response to client requests. Resolvers must be able to access at least one name server and use that name server's information to answer a query directly, or pursue the query using referrals to other name servers. A resolver will typically be a system routine that is directly accessible to user programs; hence no protocol is necessary between the resolver and the user program.*

Il existe donc trois catégories d'interactions :

- ✧ entre le résolveur et le serveur (pour obtenir de l'information relativement à une adresse littérale ou IP)
- ✧ entre le serveur et sa base locale de dénomination (afin d'en l'exploitation)
- ✧ entre deux serveurs : (afin d'échanger leurs bases de dénomination ou afin de relayer une requête localement non satisfaite)

Voici un aperçu des requêtes (la définition exhaustive de celles-ci et des protocoles sous-jacents est présentée dans le rfc 1035) :

- ✧ A *(address)* : obtenir une adresse à partir l'autre;
- ✧ NS *(name server)* : obtenir l'adresse du serveur responsable de l'homologation de l'adresse.
- ✧ CNAME *(canonical name)* : le nom de référence (canonique) de l'équipement à l'adresse indiquée;
- ✧ PTR *(pointer record)* : requête de gestion pour le sous-domaine arpa;
- ✧ HINFO *(host information)* : information descriptive concernant l'équipement à l'adresse indiquée;
- ✧ MX *(mail exchange record)* : données d'acheminement du courriel destiné à l'équipement à l'adresse indiquée;
- ✧ AXFR *(request for zone transfer)* : échange des données entre serveurs;
- ✧ ANY *(request for all records)* : toute l'information disponible concernant l'équipement à l'adresse indiquée.

Les réponses sont systématiquement retournées avec un champ TTL *(time to live)* qui borne la durée de validité de l'information transmise; il appartient au requérant de la redemander avant l'échéance.

Le service DNS utilise a priori le protocole de transport UDP mais TCP aussi disponible si la réponse dépasse de 512 octets (la limite du MTU d'UDP) et pour la synchronisation entre serveurs (requêtes AXFR). Dans le premier cas, il appartient au requérant de ré-émettre sa requête lorsqu'elle lui parvient tronquée!

En pratique, l'espace de dénomination est présentement divisé en deux secteurs : la première organisation effective du domaine : domaines génériques (3 car) et domaines nationaux (2 car; iso3166). Les domaines génériques, initialement américains, répondent à une

classification par secteur d'activité. Les domaines nationaux sont sous-catégorisés de façon variable selon le pays.