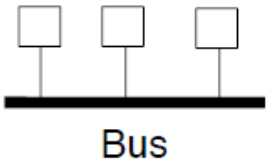
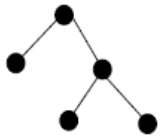
Synthèse IPP

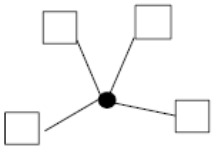
Chapitre 1 : Bases

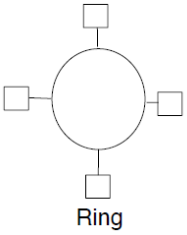
- Introduction
  - Un réseau est un ensemble d'appareils et de logiciels qui vous permettent de transmettre des informations d'un expéditeur à un ou plusieurs destinataire
  - Un réseau est simplement un ensemble de composants qui peuvent être soit logiciel, soit matériel qui mis ensemble peuvent communiquer, ils sont reliés ensemble par des moyens physiques tels que le Wi-Fi ou des câbles RJ 45.
    - On a un réseau dès que l'on connecte plusieurs choses ensemble et qu'elles peuvent communiquer
    - Pas que des réseaux connectés aux ordinateurs
      - Ex : Réseau téléphonique
      - Ex : Réseau radiophonique
        - Envoi d'informations via les ondes
  - Réseaux Actuels (Les réseaux peuvent être classés en différentes catégories) :
    - POTS (Plain Old Telephone System)
    - Téléphonie mobile
    - Réseau de Diffusion (Télévision / Radio)
    - Réseaux Informatiques (Internet, Réseaux propriétaires)
      - On doit respecter un certain ensemble de règles (appelés protocoles)
        - Un émetteur et un receveur s'échange des informations
          - Dialogue entre deux ordinateurs
    - Réseau Postal
      - Contient certaines caractéristiques identiques au réseau informatique
        - Certaines règles
          - Adresse destinataire, Adresse expéditeur
            - Contient une certaine structure
          - Contenu
- Network Classification
  - Basé sur leur couverture géographique
    - 0.1 – 1 m : Internet Bus / Network
    - 10 m – 1km: Local Area Network (LAN)
    - 1 km – 100 km: Metropolitan Area Network (MAN)
    - 100 km -> …: Wide Area Network (WAN)
    - And more …. Satellite networks, Interplanetary network
  - Basé sur leur topologie

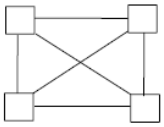
- On peut connecter les réseaux informatiques selon différentes (topologies)
  - Bus : première façon de connecter les ordinateurs (un câble central relie les ordis entre eux)
    - Pose un certain nombre de question
    - Comment les Ordis communiquent
      - Envoi de message si demande de communication
      - Le message naviguait à travers le câble
      - L'autre machine relié au câble est en attente de message
      - Si le message lui est adressé, il le lit





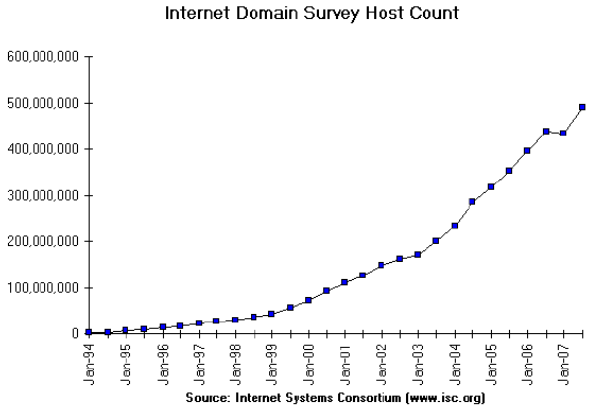






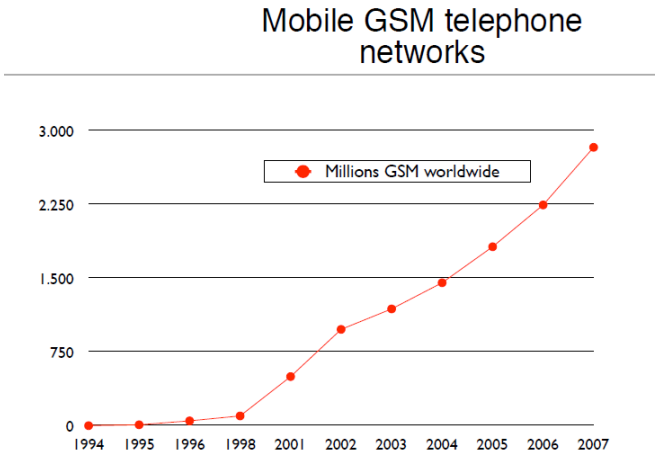
  - En entreprise : l'organisation en étoile est souvent utilisée
    - Un rooter principal est connecté à toutes les autres machines
    - Problème
      - Si le rooter crashe, plus personne ne peut communiquer avec personne
  - Organisation full mesh
    - Organisation la plus efficace
      - Tous les ordinateurs sont connectés à tous les utilisateurs
      - Problème
        - Difficile à mettre en place
        - Le nombre de câbles est quadratique et est donc très couteux
        - Inutilisable pour un grand nombre d'ordis
        - Utilisé par les providers
          - Les serveurs des providers sont connectés de cette façon
  - Problèmes
    - Lorsqu'un message est envoyé comment savoir à qui il est destiné
      - Solution : on donne une adresse à chacune des machines
      - Actuellement ces adresses sont les adresses IP
    - Que se passe-t-il si deux machines parlent en même temps
    - On ne possède pas assez de ressource (bande passante limitée)
    - Sécurité : N'importe qui qui lit le message peut le lire, le modifier
    - Ce problème est encore d'actualité
      - https = httpsecurise
        - On transforme un message lisible en un message illisible pour le commun des mortels
    - Dès que le câble est coupe le réseau se voit partitionner et les deux parties ne peuvent plus communiquer avec personne

- Evolution d'Internet

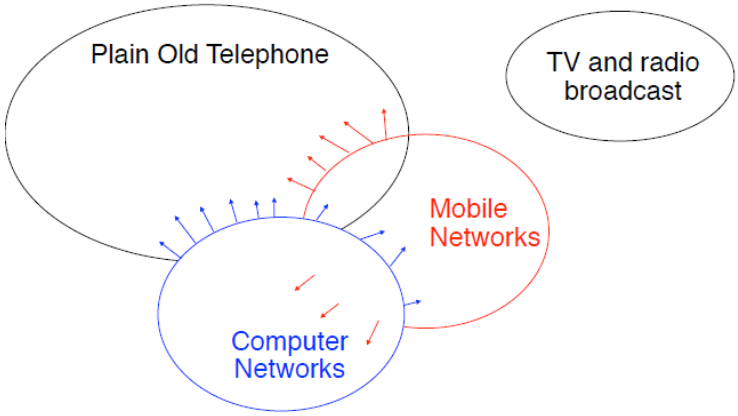


- Internet n'est pas seulement un unique réseau mais plutôt un ensemble de réseau
- Naissance d'internet :
  - D'abord les universités, des chercheurs se sont liées ensemble (ils voulaient un réseau décentralisé (tout ne devait pas avoir un point central (donc pas de structure en étoile))
    - On invente toutes les sortes de protocoles afin de communiquer
    - On va ensuite ouvrir internet à un public plus large
    - Des sociétés commencent alors à apparaitre
    - Pose un certain nombre de problèmes
      - Problème de manque d'adresse IP

- Evolution du Réseau GSM

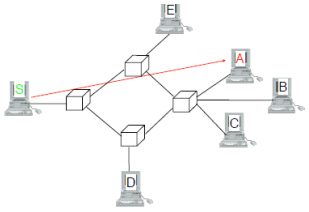


- Evolution des réseaux en général

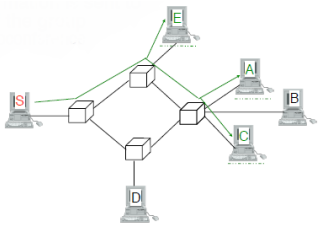


- Le Futur
  - Une grande convergence entre toutes les technologies
    - Triple play
    - Quadruple play
  - Les nouveaux services seront probablement déployés premièrement sur les réseaux de données
    - Les services de télévisions fournis par les opérateurs télécoms
    - Service de données mobiles
    - Service de télévision mobiles
    - Enregistrement audio et vidéo à partir de l'IP
    - Nouveaux services

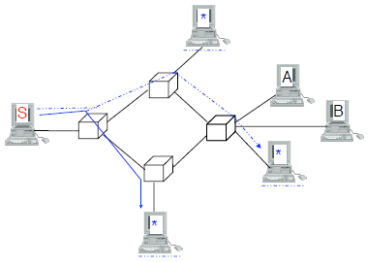
- Les différents modes de transmission
  - Unicast
    - Un expéditeur / un destinataire
    - Le plus largement utilisé sur internet
    - Mode point à point
      - Un envoyeur et uniquement un receveur
      - Dialogue entre deux personnes
      - Exemple : lorsque l'on appelle quelqu'un sur son téléphone, on commence un dialogue un à un
      - Le message atterris lors de l'envoi du message sur un rooter qui s'occupe de de le faire passer de rooter en rooter jusqu'à attendre le pc de destination
        - On essaie de faire en sorte de trouver le meilleur chemin



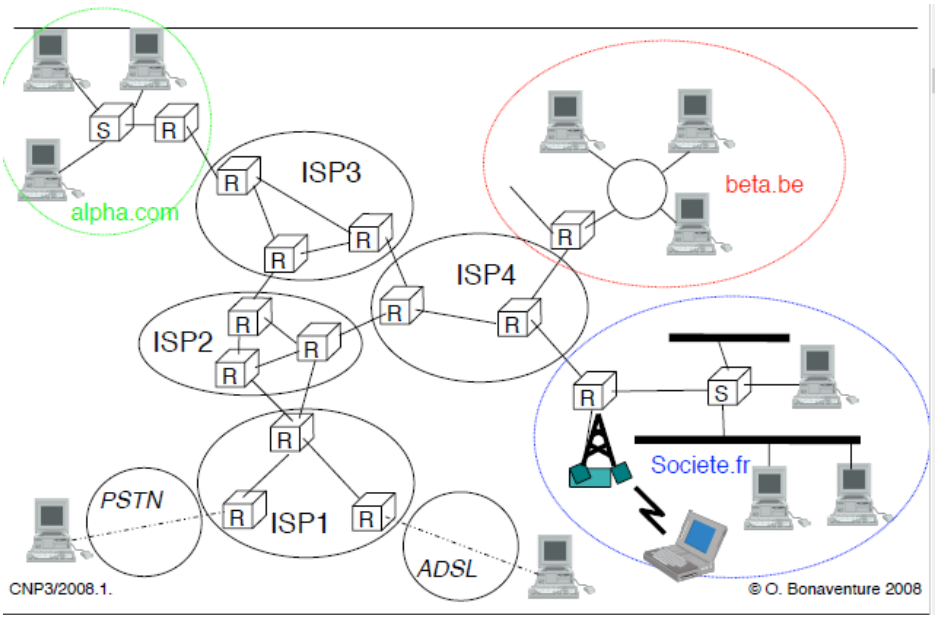
- MultiCast
  - Un expéditeur / un groupe de destinataires
  - La même information est envoyée à tous les membres du groupes
  - Utilisés sur de petits groupes de machines
  - Une information est envoyée à tous les membres du même groupe
    - Exactement ce qu'il se passe lorsque l'on fais de la radio
  - Le message est envoyé une seule fois et ce sont les rooters qui s'occupent de dupliquer le message



- AnyCast
  - L'information est envoyée d'un expéditeur vers un destinataire d'un groupe de destinataires potentiels
  - On a un groupe de machines auxquelles on peut envoyer un message, on envoie le message à une des machines du groupe (on ne sait pas à qui on envoie le message)

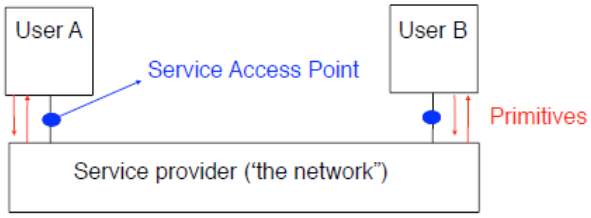


- Un petit aperçu de l'internet



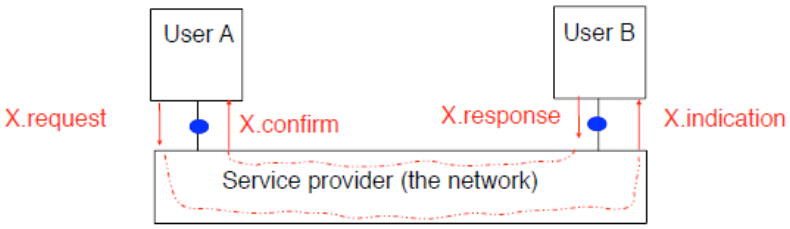
- Concept de Bases
  - Modèle abstrait du comportement du réseau
    - Le réseau est considéré comme une boite boire
    - Les utilisateurs interagissent avec le réseau en utilisant des primitives qui sont échanges à travers un point d'accès de services
    - Différents Service / Protocole
      - Service : le réseau va être responsable de vous fournir un service (mail)
        - Pour pouvoir rendre ce service, on utilise différentes règles appelés protocole
        - TCP orientés connections
        - UDP service sur internet orienté hors connexion
    - Un message pourrait être perdus
      - Les messages passent sur le rooter qui possède chacun de la mémoire (limitée) donc si le rooter est saturé et les messages qui arrivent ne peuvent donc plus être traités et donc peuvent se perdre
        - On essaie de masquer cette non-fiabilité
        - On utilise différents mécanismes
        - On utilise des accusés de réception pour être sûr que le message à bien été envoyé

- Parfois ce manque de fiabilité ne pose pas de problèmes (ex : YouTube, jeu, …), la plupart du temps, les réseaux sont fiables, de plus un mode hors connexion est plus rapide et plus facile), on utilise un mode hors connexion on contraire si l'on a besoin de fiabilité, on utilise un mécanisme connecté (par exemple pour l'envoi de mail, pc Banking, mais le mécanisme est plus lent)



Types des primitives

- Primitive
  - Représentation abstraite de l'interaction entre un utilisateur et son fournisseur de réseau
  - Peut contenir des paramètres tels que la source, la destination et / ou le message (SDU ou Service Data Unit)



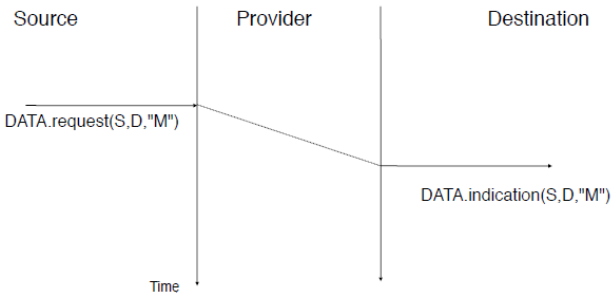
  - X.request
    - Requête d'un utilisateur à un fournisseur d'accès internet
  - X.indication
    - Primitive générée par le fournisseur de réseau vers un utilisateur (souvent liée à une précédente primitive X.request)
  - X.response
    - Primitive utilisée pour répondre à une précédente primitive X.indication
  - X.confirm
    - Primitive générée par un fournisseur d'accès internet vers un utilisateur (liée à une primitive X.response)

Le service sans connexion

- But
  - Permet à un expéditeur d'envoyer rapidement un message à un destinataire
- Principe
  - L'expéditeur place le message de façon à ce qu'il soit transmis à travers une primitive DATA.req et qu'il le donne au fournisseur de réseau. Le fournisseur de

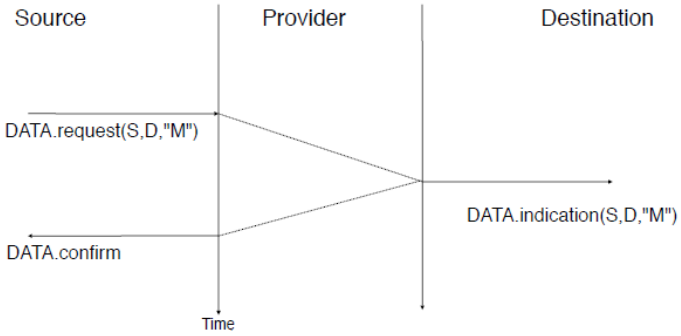
réseau transfère le message et le délivre au destinataire en utilisant une primitive DATA.ind

- Utilisation
  - Utile pour envoyer de courts messages (e.g. post office)
- Avantages
  - On ne réserve pas de ressource
  - On dépose juste le message sur le réseau (Risque de pertes du message)

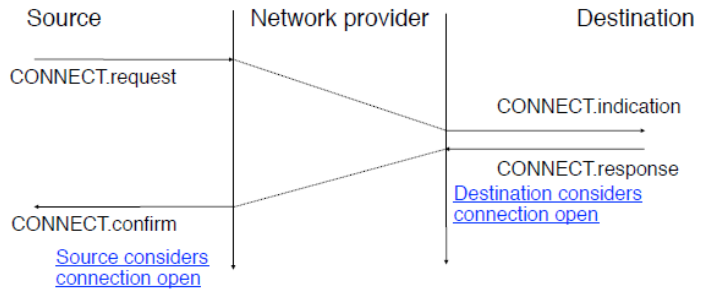


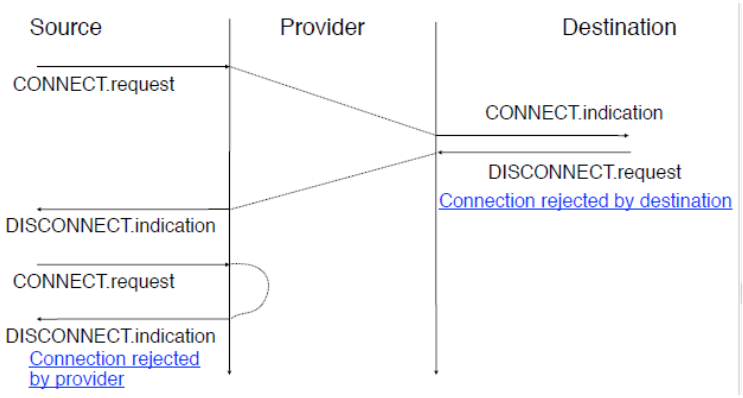
Variants of connectionless service

- Confirmation
  - Une primitive DATA.confirm est délivrée par le fournisseur à l'expéditeur pour confirmer que le message est bien arrivé à destination
- Fiabilité
  - Un service sans connexion fiable (pas d'erreurs)
  - Un service sans connexion non fiable (des erreurs sont possibles)
- Protection contre les erreurs de transmission
  - Le service peut ou peut ne pas détecter/corriger les erreurs
- Protection contre les pertes
  - Le service peut ou ne peut pas perdre les messages
- In sequence delivery
  - Pas de garantie
  - In sequence delivery pour tous les messages envoyés par une source

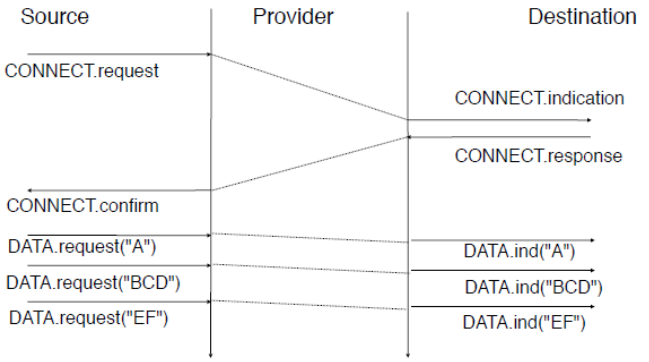


- Connection-oriented service
  - But : crée un lien logique (connexion) entre deux utilisateurs pour leur permettre d'echanger efficacement des messages
  - Phases principales du service
    - Etablissement d'une connexion
      - On réserve les ressources nécessaires
    - Transfert de données
      - Les deux utilisateurs peut envoyer et recevoir des messages sur la connexion
    - Fin de la connexion
      - On libère toutes les ressources
  - Utilisation
    - Utile quand deux utilisateurs doivent tous les deux
      - Echanger un grande nombre de messages
      - Ont besoin d'une structure d'échange (e.g. téléphone)





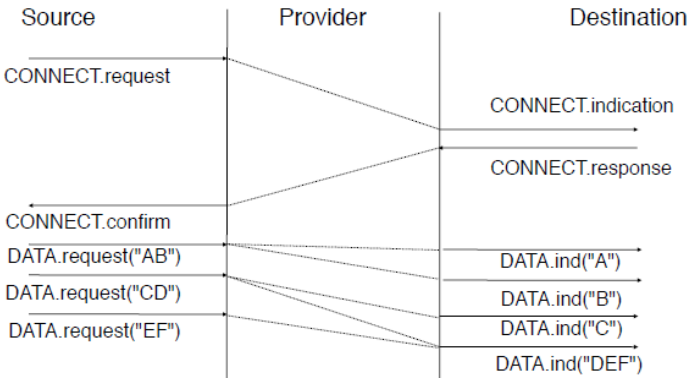
- Data transfer : message mode



Provider delivers one Data.ind for each Data.req

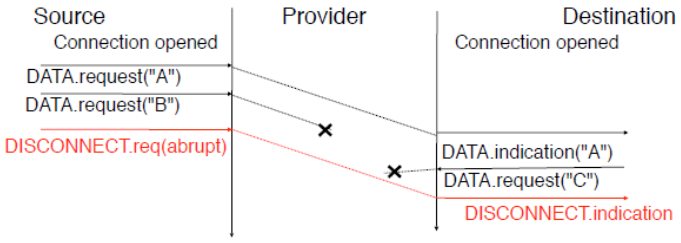
- Data transfer : stream mode

The providers delivers a stream of characters from source to destination

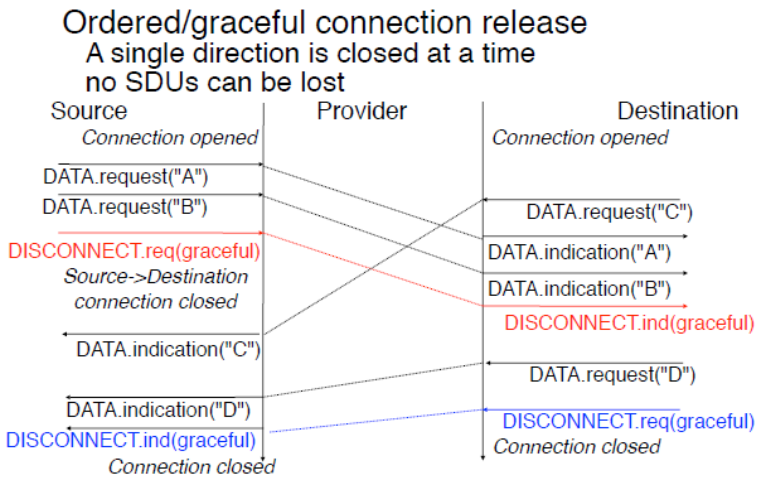


- Connection release

Abrupt release
SDUs can be lost during connection release

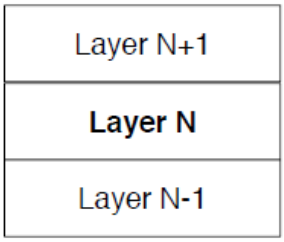


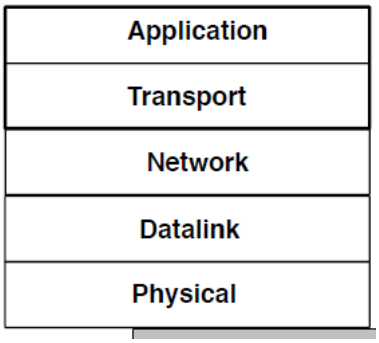
Such an abrupt connection release can be caused by the network provider or by the users



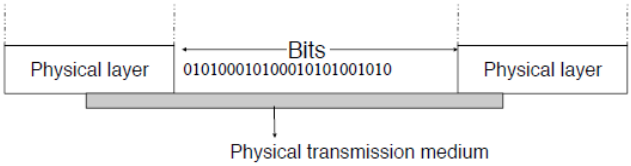
- Caractéristiques du service orienté connection
  - Caractéristiques possibles
    - Transmission bi directionnelle
      - Les deux utilisateurs peuvent envoyer et recevoir des SDU's
    - Livraison fiable
      - Les SDU's sont délivrés en sequence
      - Aucun SDU ne peut être perdus
      - Aucun SDU ne peut être corrompus
    - Mode message ou mode « stream »
      - Connection release
        - Se ferme habituellement quand le fournisseur est forcé de publié une connexion
        - Abrupt or graceful quand l'utilisateur demande la fin de la connexion

- Layered reference models
  - Problème : Comment est t-il possible de raisonner à propos de système complexe comme un réseau informatique ou Internet ?
  - Solution :
    - Diviser le réseau en couche
    - La couche N fournis un service bien définis au couche N+1 en utilisant le service obtenu par la couche N-1

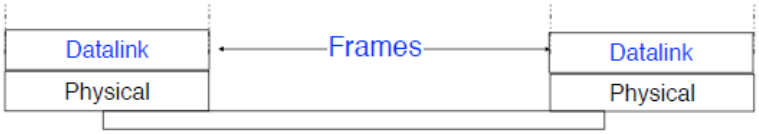




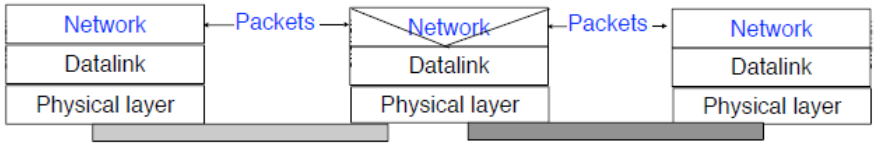
- La couche physique
  - But : Transmet des bits entre 2 appareils physique connectés
  - Service fournis par une couche physique
    - Transmission bit par bit et réception
    - Service non-fiable
      - Le destinataire peut décoder un 1 quand le destinataire envoie un 0
      - Quelques bits transmis peuvent être perdus
      - Le destinataire peut décoder plus de bits que les bits qui ont été envoyés par l'expéditeur



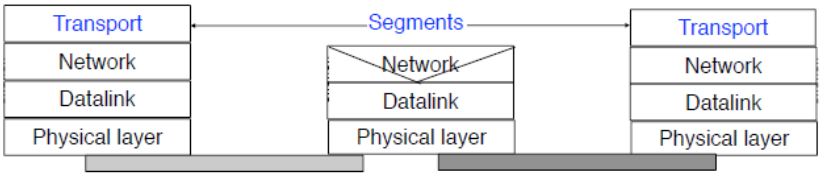
- The datalink layer
  - But : fournir un service qui permettait l'échange de « frames »
    - « frame » : groupe structuré de bits
    - Supporte les réseaux locaux
  - Services
    - Service orienté connexion fiable
    - Service sans connexion non fiable



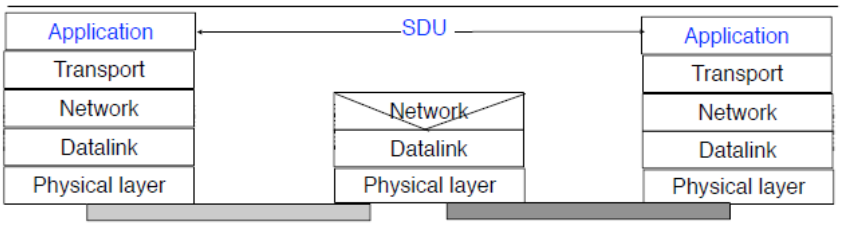
- The Network layer
  - But
    - Permet aux informations d'être échanger entre les hébergeurs qui ne sont pas attachés au même lien physique de taille moyenne en utilisant des relais
    - L'unité d'informations dans la couche réseau est appelé un paquet
  - Services
    - Service sans connexion non-fiable (internet)
    - Service orienté connexion fiable



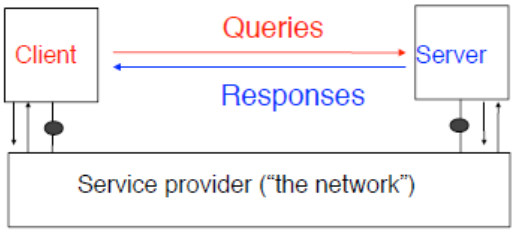
- The Transport Layer
  - But : assure un échange fiable de données entre "endsystems" même si la couche réseau ne fournis pas un service fiable
  - Services
    - Service sans connexion non fiable
    - Service orienté connexion fiable



- The application layer
  - But : échange d'informations utiles entre applications en relayant sur la couche de transport qui cache la complexité du réseau
  - Unit of information
    - Service Data Unit, SDU

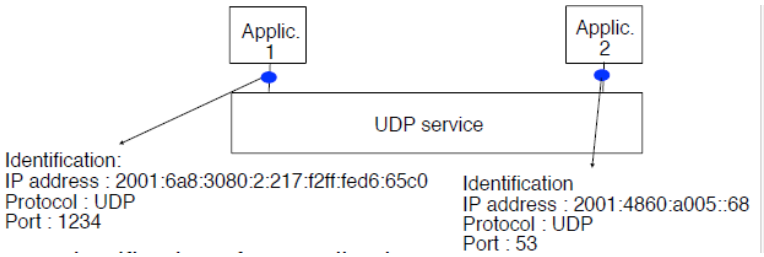


- The client server model
  - Client : interagis avec le serveur à travers la couche de transport et envoie des « queries » ou des commandes
  - Server : Réponds aux « queries » reçue de la part du client. Exécute les commandes du client. De nombreux clients peuvent utilisés le même serveur
  - Le client et le serveur interagissent avec le fournisseur de service. Le client et le serveur doivent parler le même langage.
    - Application-level protocol : ensemble de règles syntaxiques et sémantiques qui définissent les messages échanger entre le client et le serveur ainsi que leur classement



- Les services du transport sur l’internet
  - Sur internet on utilise deux services de transport différent :
    - Le service fournis par l’« User Datagram Protocol » (UDP)
      - Service sans connexion non fiable avec détection d’erreurs
    - Le service fournis par le « Transmission Control Protocol » (TCP)
      - Service orienté connexion fiable avec flux de bits

- UDP service
  - Identification d’une application
    - Adresse IP + UDP + numéro de port
  - Caractéristiques du service UDP
    - Service sans connexion
    - Non fiable
      - Les messages peuvent être perdus
      - Les erreurs de transmission peuvent être détectées mais pas récupérées
      - La séquence n’est pas préservée
    - Taille maximum des messages : presque 64 Kbytes

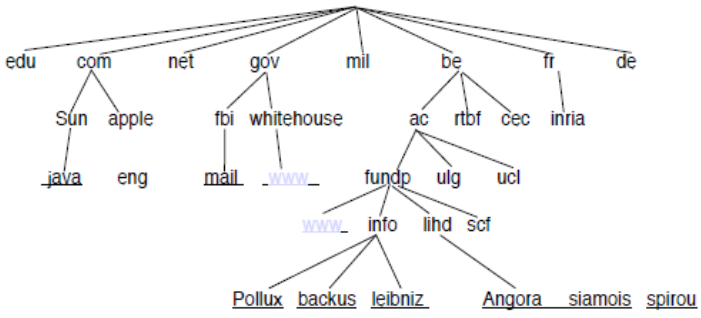


- TCP service
  - Identification d'une application
    - Adresse IP + UDP + numéro de port
  - Caractéristiques du service TCP
    - Orienté connexion
    - Bidirectionnel
    - Fiable
    - Flux de byte
    - Fin de connexion
      - Abrupt si initié par le fournisseur de service
      - Doux ou abrupt si initié par l'utilisateur

- Names and addresses
  - Adresse du serveur
    - Adresse IP de l'hébergeur sur lequel le service de numéro de port est en cours de fonctionnement (TCP ou UDP)
      - Généralement numéro de port connus
    - Inconvénients
      - Difficile de se rappeler une adresse IP pour un être humain
    - Idée
      - Remplace l'adresse IP par un « hostname »
      - Plus facile pour les humains
        - Mais l'adresse IP est nécessaire pour contacter le serveur
      - Comment transformer une adresse IP en un « hostname »
        - Fichier Hosts.txt
          - Contient une table des noms d'adresses qui doit être mises à jour régulièrement
          - Ne peut pas être utilisés dans un large réseaux

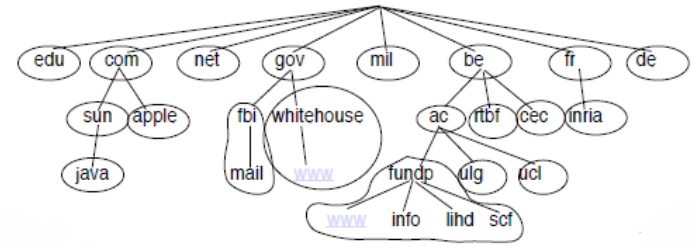
```
#
# Internet host table
#
127.0.0.1       localhost
138.48.32.99    babbage
138.48.32.100   leibniz
138.48.32.1     routeur
138.48.32.92    corneille
138.48.32.107   backus
138.48.20.152   arzach
138.48.32.137   almin01
138.48.32.170   duke
```

- Hostnames
  - Nécessaire
    - Les « hostsnames » doivent être uniques
  - Comment parvenir cela de manière proportionnée
    - En introduisant une hiérarchie
    - Chaque « hostname » est composé de deux parties
      - Nom du domaine (globalement unique)
      - Hostname (unique au sein d'un même domaine)
  - Comment distribuer de manière unique des noms de domaines
    - En introduisant une hiérarchie
    - Un petit nombre de noms de domaines de haut niveau
    - Au sein de chaque domaine de haut niveau, on alloue un second niveau de nom de domaines
    - Au sein de chaque second niveau de domaines, on alloue un troisième niveau de noms de domaines et de « hostsnames »



- Comment traduire les noms dans des adresses
  - Comment traduire efficacement les « host name »
    - En utilisant une base de données centralisée
      - Il y a plus de 1 billion d'host names actuellement
    - En utilisant une base de données distribuée
      - DNS : Domain NameSystem
      - Relie la hiérarchie des noms de domaines

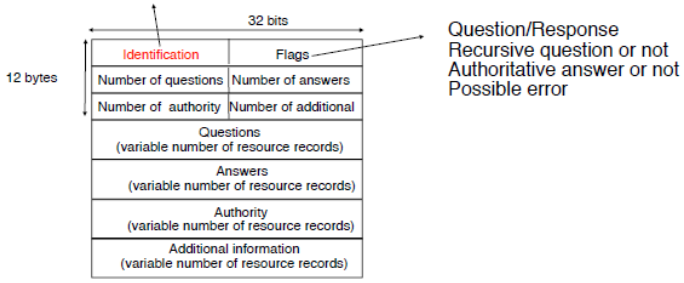
- Il y a un serveur qui doit demander de traduire les hosts names au sein de ce domaine



- DNS resolver
  - Pour traduire les noms et adresses, nous avons besoin de l'implémentation d'un DNS
    - Afin de connaitre la liste actuelle des adresses IP du serveur de routing et d'implémenter un protocole DNS et de de traverser la hiérarchie des noms de domaine.
    - Solution
      - DNS Resolver
        - Un résolveur pour un ensemble d' « endhosts »
        - Maintenir les listes des adresses IP et des serveurs de routing à jour
        - Implémenter un protocole DNS
      - « Endhosts »
        - Seulement capable d'envoyer des requêtes DNS au résolveur
        - Doit connaitre les adresses IP des DNS résolveurs les plus proches

- DNS : optimisations
  - Réduire le risque d'échecs
    - Différents « root-servers »
    - Le serveur DNS faisant autorité pour chaque domaine
    - Chaque « endhosts » peut envoyer des « queries » à différents résolveurs
  - Des performances améliorées
    - Eviter d'envoyer plusieurs fois la même « query »
    - Cache mémoires dans les résolveurs DNS contenant :
      - Traduction récentes des name-addresses
      - Les adresses des serveurs DNS récemment contactés
  - Protocole DNS
    - Fonctionne généralement sous UDP
    - Parfois également utilisés sur TCP

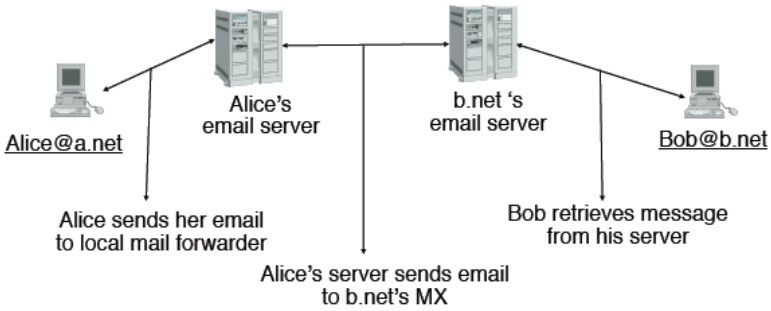
- DNS : message format
  - Chaque requête DNS contient un nombre qui sera retourné en réponse par le serveur pour permettre au client de faire correspondre sa requête



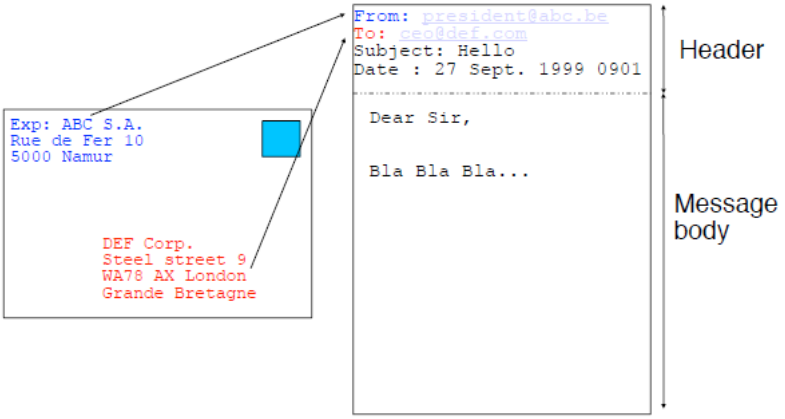
- DNS : ressource records
  - Chaque message DNS est composé de ressources d'enregistrement encodé en tant que TLV

< Name, Value, Type, TTL>
Types de RR
Lifetime of the RR in server's cache
A (Address)
Name is a hostname and Value an **IPv4 address**
AAAA (Address)
Name is a hostname and Value an **IPv6 address**
NS (NameServer)
Name is a domain name and Value is the hostname of the DNS server responsible for this domain
MX (Mail Exchange)
Name is a domain name and Value is the name of the SMTP server that must be contacted to send emails to this domain
Type CNAME
Alias

- Email
  - Modèle simplifié
    - Alice envoie un email à Bob
    - On utilise un DNS pour trouver l'adresse des serveurs mails
    - Lorsque le service reçoit le mail
      - Le serveur va transmettre le mail au serveur du destinataire
      - Le DNS va traduire et trouver l'adresse du destinataire
      - La boite mail est un peu comme une boite aux lettres
      - Le contenu du mail est toujours traduit sous son format ASCII



- Email message format
  - Format du « header »
    - Contient uniquement des caractères US-ASCII (codés sur 7bits)
    - Au moins 3 lignes qui finissent avec « <CRLF> »
      - From : sender@domain
      - To : recipient@domain (Personne Principale)
      - Date : <creation date of message>
    - Champs optionnels
      - Subject : subject of message
      - Cc : copy@domain (Personne secondaire)
        - Ou CCI (Permet de cacher ces destinataires-là aux autres)
      - Message-ID : <number@domain>
      - Received : information on path followed by message
      - In-reply-to : <message-ID>
    - Le “header” se termine avec une ligne vide (<CRLF>)
    - Est formé de 12 bytes
      - On sait exactement comment les coder
      - Chaque question ou réponse va avoir la même forme / structure



- MIME
  - Les emails ont été faits pour le code US-ASCII
    - Comment faire pour envoyer des messages plus complexes
    - On convertis le contenu extrêmement riche en contenu ASCII
      - Peu importe le type de contenu (Power Point, …)
    - Permet de mettre plusieurs objets différents au sein d’un même mail
  - Multipurpose Internet Mail Extensions
    - Amélioration du format de message des emails
    - Contraintes :
      - Doit rester compatible avec les anciens serveurs de mail
        - La plupart d’entre eux ne supporte que les caractères US-ASCII et les courts messages

- Doit supporter le code non Anglais
  - Les caractères envoyés doivent dépassés les 7 bits du code US-ASCII
- Doit supporter plusieurs formats en un seul message
  - Le corps du message, les pièces jointes
- Doit permettre d'envoyer des sources audio, vidéo
  - On doit donc identifier le type de contenu

- Solution :
  - Ajouter de nouveaux champs optionnels dans le « header »
    - MIME-Version :
      - La version de MIME utilisée pour encoder le message
        - Version actuelle : 1.0
    - Content-Description :
      - Un commentaire décrivant le contenu du message
    - Content-type (type/encoding) :
      - Le type d'informations au sein du message
        - Text, image, vidéo, application, multipart
      - Encoding of content :
        - Text/plain, image/gif, audio/basic, video/mpeg
        - Multipart/alternative
          - Message contenant plusieurs fois la même information avec différents encodage
        - Multipart/mixed
          - Message contenant plusieurs informations de différents types
    - Content-Transfer-Encoding :
      - Comment le message a-t-il été encodé
    - Content-Id :
      - Un identifiant unique pour le contenu
  - Ajouter des champs optionnels dans le message

Characters sets and content encoding

- Comment supporter des ensembles de caractères plus riche
  - Content-Type : text/plain ; charset=us-ascii
- Comment encoder des données non textuelles
  - Les données doivent être encodés dans le code caractères US-ASCII codés sur 7bits et en Base64
    - Pg 42 syl
    - Contient uniquement 64 caractères du code ASCII
    - 256 paquets différents de 8 bits
  - Chaque caractère est encodé sur 6 bits
    - Ex : 24 bits du message initial => 4 caractères ASCII
  - Le caractère spécial « = » est utilisé pour le remplissage

- On n'a pas toujours de multiples de 3 octets (comment fait t'-on pour passer de 8 bits à 6 bits)
    - On utilise le symbole « = »
        - On ajoute des zéros et lors de la traduction 0 équivaut à « = »
        - Trois cas possible
            - 3*q + 2
            - 3*q + 1
            - 3*q

            « Ou q représente le reste »
        - Attention erreur possible dans les slides (lire le syllabus)

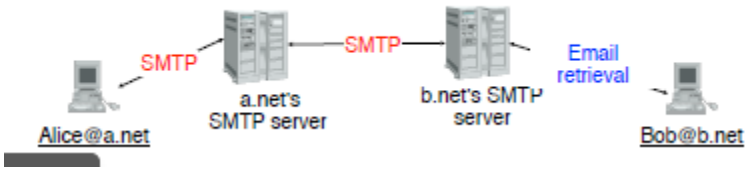
- Comment placés différents contenus dans un simple message
    - On a besoin d'un délimiteur entre les différents types de contenus placés à l'intérieur du message (body)
        - Suite de caractères non visibles dans le mail
        - On précise dans le header quel est le séparateur
            - C'est l'application qui choisit le séparateur le plus souvent

```
Date: Mon, 20 Sep 1999 16:33:16 +0200
From: Nathaniel Borenstein <nsb@bellcore.com>
To: Ned Freed <ned@innosoft.com>
Subject: Test
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="simple boundary"

preamble, to be ignored

--simple boundary
Content-Type: text/plain; charset=us-ascii

partie 1

--simple boundary
Content-Type: text/plain; charset=us-ascii

partie 2
--simple boundary
```

Email transmission

- SMTP (Simple Mail Transfer protocol
    - Utilise le service TCP
    - Adresse du serveur SMTP
        - Adresse IP du serveur + TCP + numéro de port : 25
    - Modèle client-serveur
        - Le serveur attend les emails à livrer, relayer
        - Le client envoie des emails à travers le serveur
    - Application-level protocol
        - Le client ouvre la connexion TCP
        - Le client envoie des commandes composées de :
            - Command parameter <CRLF>
                - HELO
                - MAIL FROM
                - RCPT TO

        - DATA
        - QUIT
    - Le serveur répond avec des messages d'une ligne
        - Numeric code comment (text) <CRLF>
            - 250 OK
            - 221 closing
    - Trois phases au SMTP
        - Établir une connexion SMTP
            - Connexion TCP établie lors d'une requête du client
            - Salutations de la part du serveur
            - Commande HELO de la part du client
        - Transférer le message
            - MAIL FROM : <user@domaine>
            - RCPT TP : <user@domaine>
            - DATA
        - Fin de la connexion SMTP
            - QUIT
            - Messages de fermeture de la part du serveur
            - La connexion TCP est fermée



- Retrieval of email messages
    - Dans le temps :
        - La destination est toujours connectée à internet
            - Les adresses email sont de type : username@hostname
            - Quand un email arrive, il est enregistré dans un fichier qui appartient à l'utilisateur (par exemple : /var/mail en Unix)
    - De nos jours :
        - La plupart des réseaux ont un ou peu de serveurs SMTP utilisés pour récupérer des emails mais aussi pour détecter les spams, les virus
        - Les utilisateurs récupèrent leurs emails depuis ce serveur
            - Post Office Protocol (POP)
            - Internet Mail Access Protocol (IMAP)
            - Webmail

- POP
    - But
        - Permettre aux utilisateurs authentifiés de récupérer des mails depuis le serveur
    - Opération

- POP utilise le service TCP
- L'adresse du serveur POP
  - Host Address + TCP + port number: 110
- Le client envoie des commandes
  - Les commandes sont des lignes en ASCII se terminant par le caractère <CRLF> (USER, PASS, STAT, RETR, DELE, QUIT)
- Le serveur répond avec
  - OK si la commande fut un succès
    - Les messages email suivent plusieurs réponses « ok »
  - ERR s'il y a une erreur

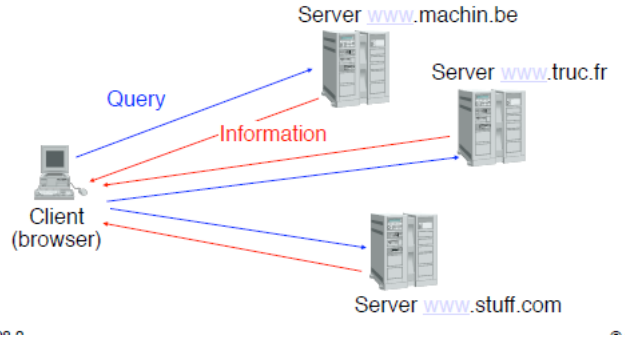
▪ Les trois phases du protocole

- Autorisation (Vérifier les informations de l'utilisateur)
  - USER <username>
  - PASS <password>
- Transaction (Récupération et suppression des messages)
  - STAT
    - Liste des « headers » des messages enregistrés
  - RETR <n>
    - Récupération du nième message
  - DELE <n>
    - Le nième message est marqué pour la suppression
- Mise à jour (Fin de la phase de récupération)
  - Les messages marqués pour la suppression sont supprimés du serveur
  - La connexion TCP est fermée

• World Wide Web

▪ But

- Permettre aux navigateurs de naviguer entre les documents hypertexte sauvegardés sur de multiples serveurs



- Les 5 éléments clés du « www » sont
  - Un système d'adressage qui permet d'identifier n'importe quel document sauvegardé sur le serveur
    - URL

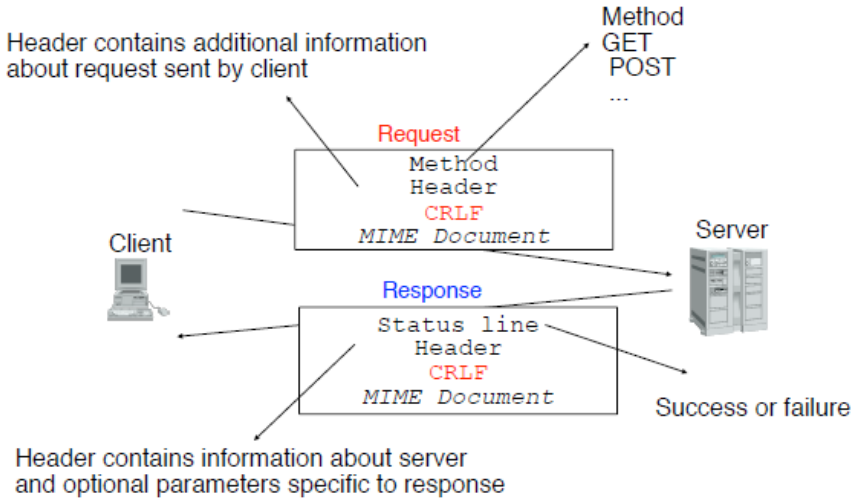
- Un langage hypertextuel qui permet de facilement écrire des documents avec des liens hypertexte
  - HTML
- Un protocole de niveaux d'applications efficace et léger permettant de s'échanger des documents
  - HTTP
- Des serveurs
- Des clients (ici des navigateurs)

Uniform Resource Locator (URL)

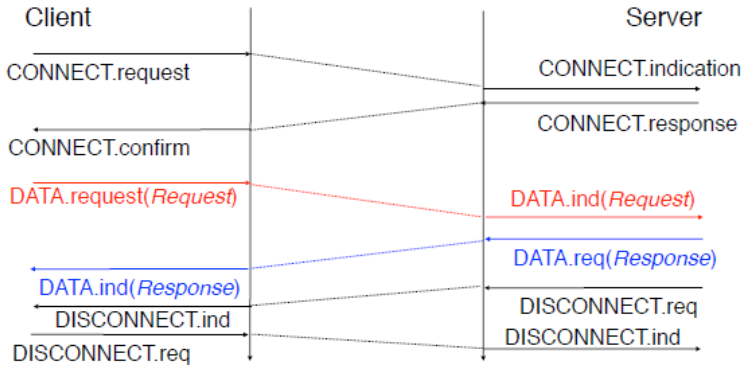
- Syntaxe générique : <protocol>://<document>
  - « protocol » est utilisé pour récupérer les documents depuis le serveur
    - Le protocole le plus commun étant http
  - « document »indique le serveur et la localisation du document
  - <user> :<password>@<server> :<port>/<path>
    - <user> nom d'utilisateur optionnel
    - <password> mot de passe optionnel
    - <machine> nom d'hébergeur ou adresse IP du serveur qui héberge le document
    - <port> Numéro de port optionnel
    - <path> emplacement du document sur le serveur

Information transfer www

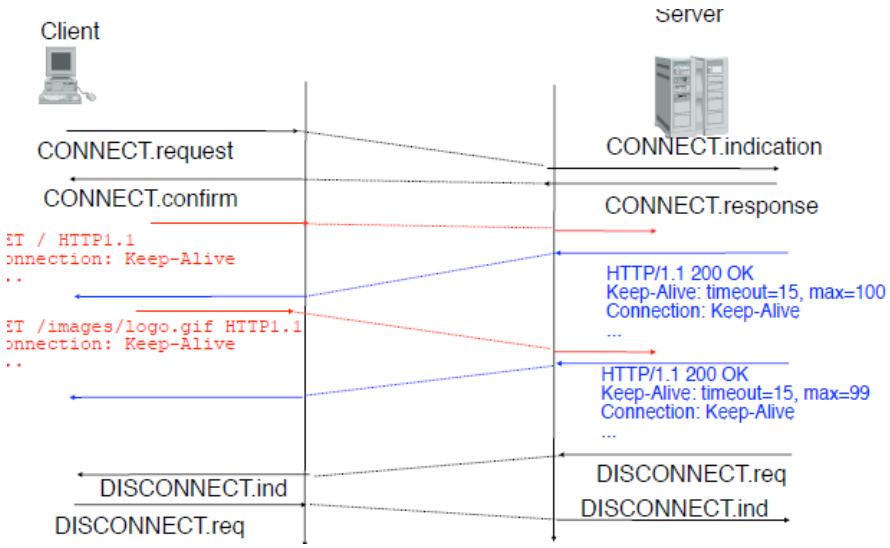
- HTTP
  - HTTP est un protocole sans états qui ne maintient aucun état d'une requête à une autre
    - Ne garde pas les infos entre les requêtes
    - Plus simple à implémenter / gérer
    - Pour garder les infos : on utilise des cookies
    - POP, FTP, SMTP sont des exemples de protocoles avec états



- HTTP 1.0 : connexion non persistante
  - Principe
    - Repose sur le service TCP (port par défaut : 80)
    - Le client envoie des requêtes, le serveur répond
  - Une connexion TCP unique utilisée pour transmettre
  - Un seul document



- HTTP 1.1
  - Utilise une connexion TCP unique mais persistante
    - Cette connexion TCP peut être utilisée pour plusieurs requêtes et pour les réponses correspondantes
    - Reste tout de même sans états
    - Keep alive : temps que le timeout n'est pas dépassée, on ne coupe pas la connexion (se réinitialise à chaque demande)
      - On a nombre limites de documents disponibles au téléchargement



- Méthodes
  - GET
    - Méthode utilisée pour demander un « document » stocké sur le serveur
      - GET <document> HTTP/1.0
  - POST
    - Méthode utilisée pour envoyer un « document » sur le serveur
      - Le document est une partie de la requête et encodé en tant que document « MIME »
- HTTP : les headers de requêtes
  - Permet de rajouter de l'information sur le client ou sur la requête
    - Host : <name>
      - Nom du serveur où le document est enregistré
    - Authorization
      - Permet d'exécuter des contrôles d'accès
    - If-Modified-Since : <date>
      - Le serveur va envoyer le document demandé uniquement si le document est plus récent que la date
    - Referer : <url>
      - Information indiquant que l'URL visitée par le client avant sa requête
    - User-Agent : <agent>
      - Information précisant le type de navigateur utilisé par le client
- HTTP : les lignes de statut
  - Format : Version_HTTP Code Comment
  - Échecs / Succès
    - 1xx
      - Pour information
    - 2xx
      - Succès

- 3xx
  - Redirection
  - LA requête n'a pas pu être traitée par le serveur local et à été envoyée à un autre serveur
- 4xx
  - Erreur coté client
  - Erreur de syntaxe, URL inatteignable, non autorisée
- 5xx
  - Erreur coté serveur
  - Erreur interne, méthode non implémentée sur le serveur

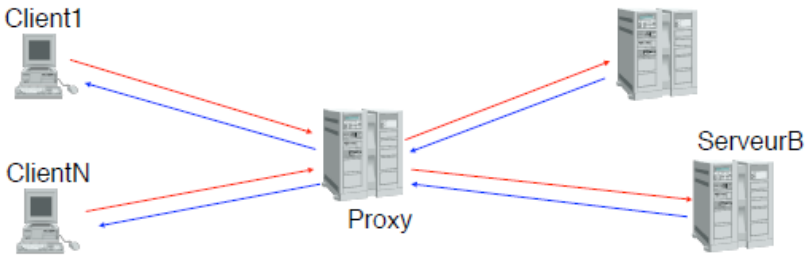
« Le premier chiffre indique s'il y a erreur ou succès »

« Les deux autres chiffres permettent de détailler le message »

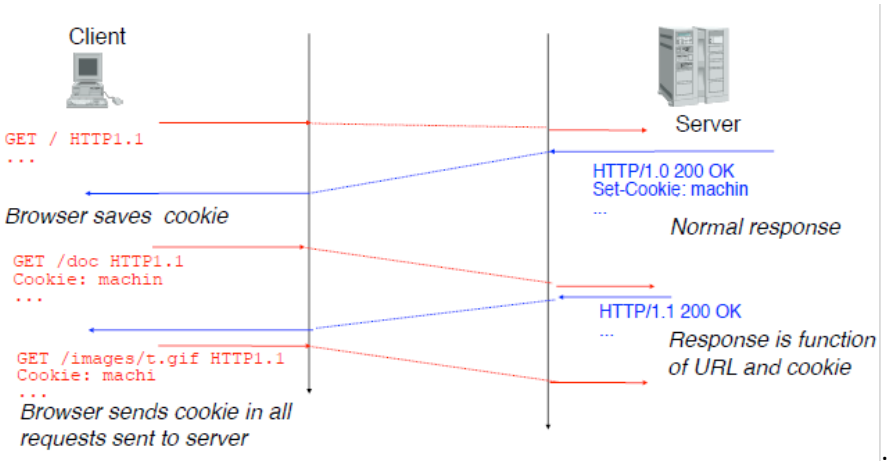
- HTTP les « headers »de réponse
  - Information optionnelle à propos du serveur, de la réponse ou du document attaché à la réponse
    - Date du document attaché à la réponse
    - Serveur (Nom et version du serveur http utilisé)
    - Content-* (« MIME header » attaché au document)

Improving performances

- Observation
  - De nombreuses pages sont demandées à de multiples reprises ou depuis des noms d'hôtes très proches
  - Solution
    - Des caches locaux pour chaque client
      - Le header if-modifies-since peut également aidé
    - Un cache pour de multiples hotes

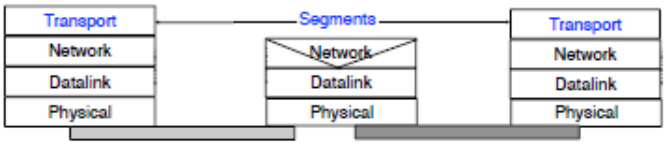


- HTTP Cookies
  - Permettent de différencier une connexion d'une autre connexion
  - 1 cookie par site
  - Lier au site et non au serveur



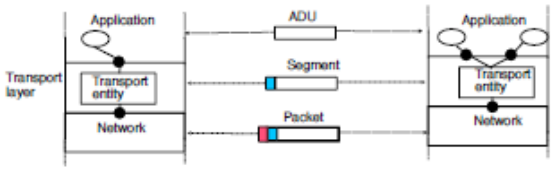
Chapitre 3 : Transport Layer

- The transport layer
  - But
    - Améliore le service fournis par la couche réseau pour permettre de le rendre utilisable par les applications
  - Les services de la couche transport
    - Service sans connexion non-fiable
    - Service orienté connexion fiable

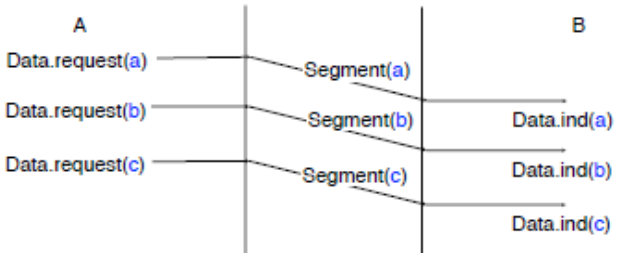


  - Les problèmes à résoudre à l'aide de la couche transport
    - La couche transport doit permettre à deux applications d'échanger de l'information
      - Cela nécessite une méthode permettant d'identifier les applications
    - La couche transport doit être utilisable par les applications
      - Détecter les erreurs de transmission
      - Corriger les erreurs de transmission
      - Récupérer les paquets perdus et les paquets dupliquer
      - Différents types de services
        - Sans connexion
        - Orienté connexion

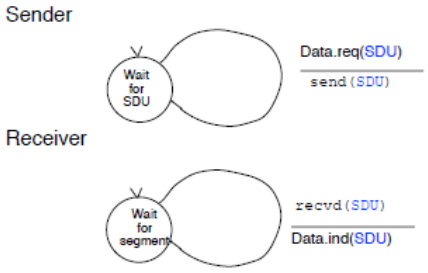
        - Question-réponse
    - Organisation interne
        - La couche transport utilise le service fournis par la couche réseau
        - Deux couches réseaux s'échange des segments



- Transport layer protocols
    - Comment puis-je fournir un service fiable au niveau de la couche de transport
    - Protocol 1 (Base)
        - Dès l'arrivée d'un data.request (SDU), l'entité de transport envoie un segment contenant ce SDU via la couche réseau (send(SDU))
        - Dès l'arrivée du contenu de l'un des paquets de la couche réseau, l'entité de transport délivre le SDU trouvé dans le paquet à l'utilisateur en utilisant data.ind(SDU)



        - Issue
            - Que se passe-t-il si le destinataire est plus lent que l'expéditeur
        - Protocol 1 (As FSM)

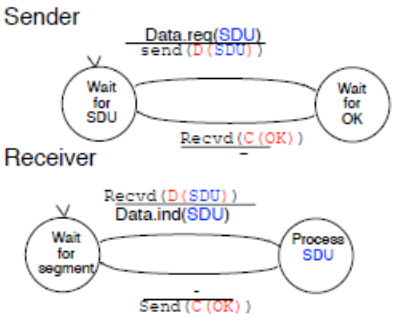


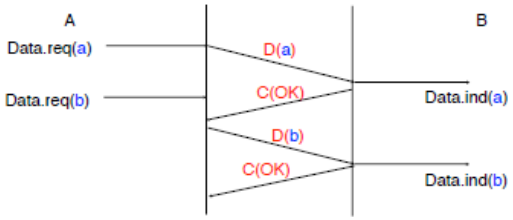
        - Protocole 2
            - Principe

- Utilise un segment de contrôle (OK) qui est envoyé par le destinataire après avoir effectué le segment reçu
- Créer une boucle de réponse entre l'expéditeur et le destinataire

- Conséquence
  - Deux types de segments
    - Data Segment content le SDU
      - Notation : D(SDU)
    - Control Segment
      - Notation : C(OK)

  - Format des segments
    - Au moins un bit dans le header du segment est utilisé pour indiquer le type du segment





    - L'expéditeur peut uniquement envoyé des segments lorsqu'il y est autorisé » par le destinataire

- Protocole 3
  - Erreurs de Transmission
    - Quelles erreurs de transmission doit on considérer sur la couche de transport ?
      - Les erreurs de transmission naturelles sur la couche physique
        - Erreurs aléatoires isolées
          - Un bit est inversé sur le segment
        - Erreur de débordement aléatoire
          - La plupart des bits dans le groupe se sont inversées
  - Les informations envoyées sur le réseau peuvent êtres corrompues pour d'autres raisons que par des erreurs de transmission

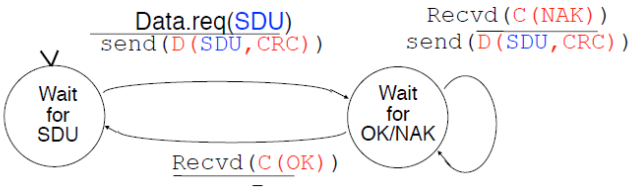
- Ces attaques peuvent être effectuées en utilisant des protocoles de sécurité spéciaux ainsi que des protocoles en dehors de la couche de transport.

- Comment détecter ces erreurs de transmission
  - Principe
    - L'expéditeur ajoute un peu d'information de contrôle à l'intérieur du segment.
      - L'information de contrôle est calculée sur l'entièreté du segment et placée dans l'en-tête ou au bout du segment.
      - Le destinataire vérifie que l'information de contrôle reçue est correcte en la recalculant.

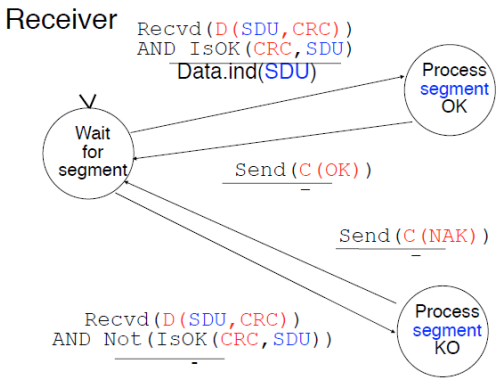
  - Parité des bits
    - Solution simple afin de détecter les erreurs de transmission
      - Utilisée sur les lignes séries à faible vitesse
    - Parité impaire
      - Pour chaque groupe de n bits, l'expéditeur calcule le n + énième bit de sorte que le groupe n + 1 contient un nombre impair de bits mis à 1
  - Internet Checksum
    - Motivation
      - Les protocoles internet sont implémentées dans les logiciels et nous aimerions bien avoir des algorithmes efficaces afin de détecter les erreurs de transmission et qui sont faciles à implémenter.
    - Solution
      - Internet checksum
        - Voir annexe prise de notes.
  - Comportement du destinataire
    - Si le checksum est correct
      - Envoie un segment de contrôle (OK) à l'expéditeur afin de :
        - Confirmer la réception du segment de données
        - Permettre à l'expéditeur d'envoyer le segment suivant.
      - Si le checksum est incorrect
        - Le contenu du segment est corrompus et doit être jeté

- L'expéditeur envoie un segment de contrôle spécial (NAK) à l'expéditeur afin de lui demander de renvoyé les données corrompues du segment de données

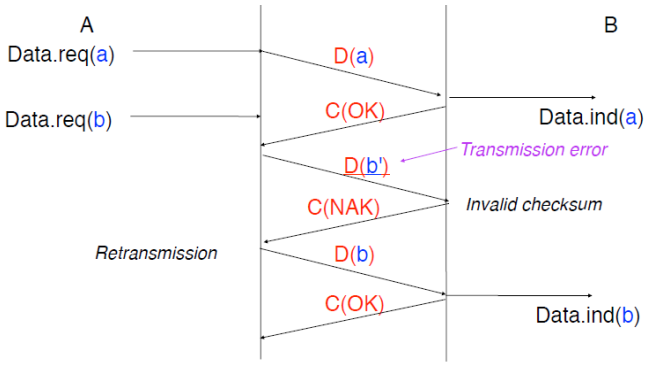
- Protocole 3a
  - Algorithme Expéditeur



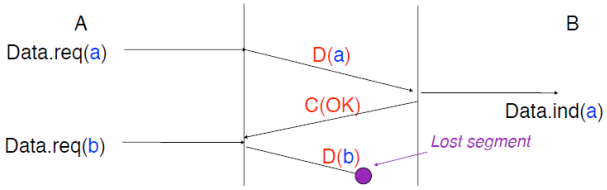
  - Algorithme Destinataire



  - Exemple

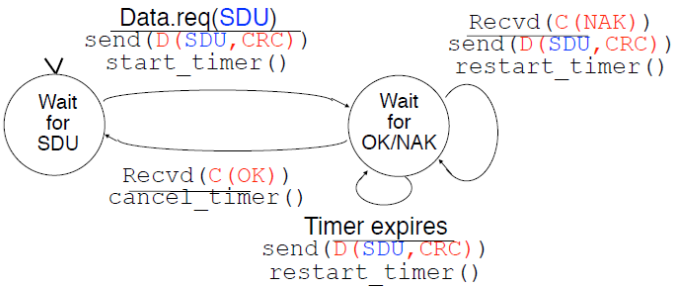


  - Problème

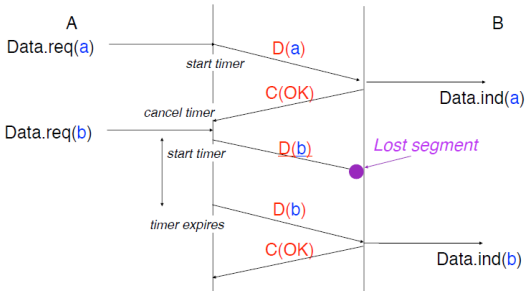


DEADLOCK

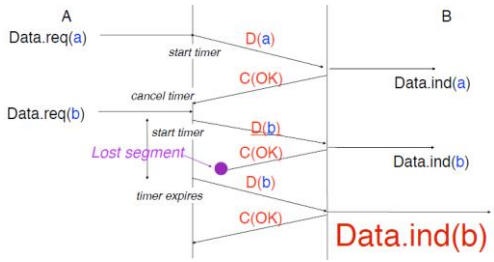
- Protocole 3b
  - Modification de l'expéditeur
    - On ajoute un timer de retransmission afin de renvoyer le segment perdu après un certain temps



  - Pas de modification du destinataire
  - Exemple



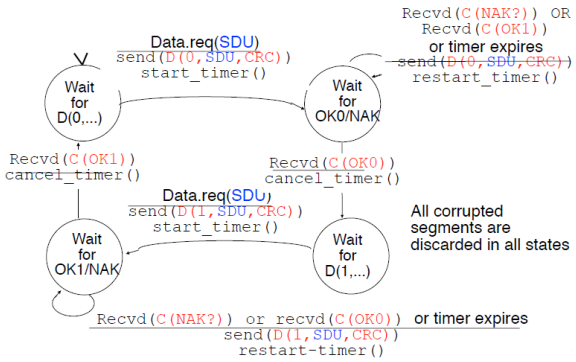
  - Problème



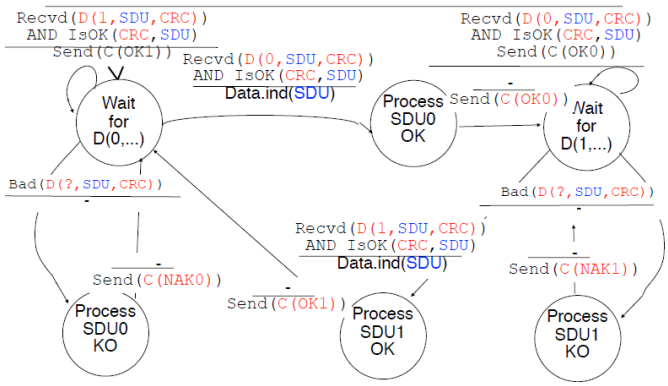
- Protocole d'alternance de bit
  - Principe de la solution
    - Ajouté un numéro de séquence à chaque segment de données envoyés par l'expéditeur
      - À chaque réception, le destinataire va envoyer un accusé de réception
    - En regardant à ce numéro de séquence, le destinataire peut vérifier s'il a déjà reçu ce segment.
    - Lorsqu'il reçoit un message de la couche réseau et lorsque le CRC est ok il indique qu'il attend le message avec le numéro correspondant et dès qu'il est ok on le distribue à la couche application

- Une fois que l'on traite les messages possédant le numéro correspondant, on attend les numéros suivant.
  - On peut recevoir plusieurs messages portant le même numéro
    - On ne le distribue pas car on l'a déjà distribué précédemment
    - Cela est surement dû à une perte de l'accusé de réception
      - On renvoie l'accusé de réception correspondant

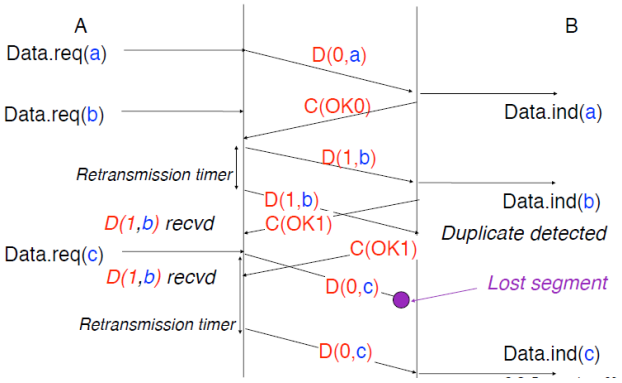
- Contenu de chaque segment
  - Segment de données
  - Segment de contrôle
- Problème
  - On envoie pour chaque problème un NAK
    - Numéro de NAK => correspond au numéro du message
- Algorithme expéditeur



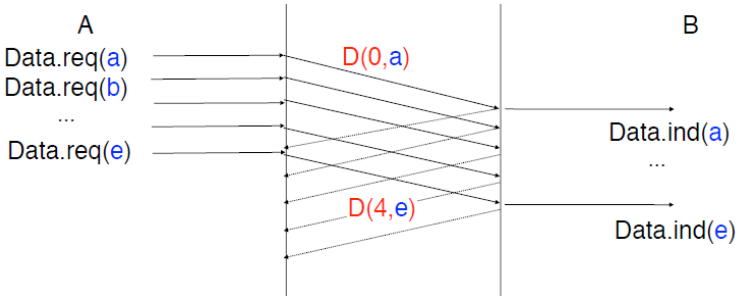
- Algorithme destinataire



- Exemple



- Performance de l'alternance de bits
  - Est calculée en fonction de la bande passante ainsi que du temps de parcours
    - Pour chaque envoi de message on doit attendre son accusé de reception
    - On attends donc la plupart du temps
  - Comment améliorer ces performances
    - Utiliser un pipeline
      - L'expéditeur devrait être autorisé à envoyer plus d'un segment en attendant une confirmation du destinataire

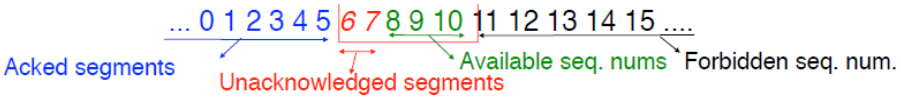


  - On ajoute un numéro de séquence à l'intérieur de chaque segment
    - Chaque segment de données contient son propre numéro de séquence
    - Chaque segment de contrôle indique le numéro de séquence du segment de données en cours de vérification (OK/NAK)
  - Pour l'expéditeur
    - On a besoin d'assez de buffers afin de sauvegarder les

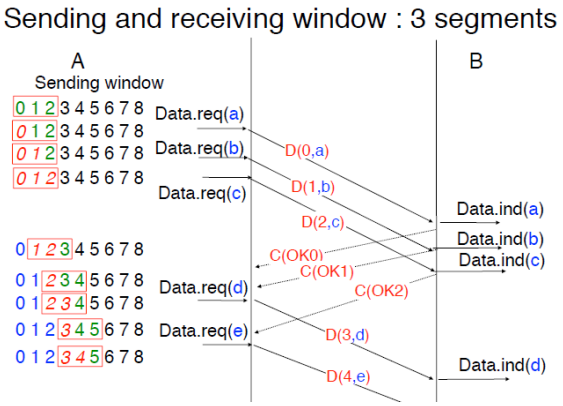
segments de données qui n'ont pas encore été reconnus afin de les retransmettre si nécessaire

- Pour le destinataire
  - On a besoin d'assez de buffers pour stocker les segments de données supplémentaires

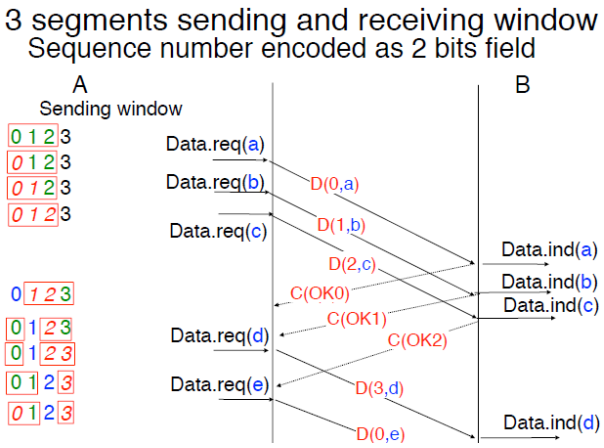
- Comment empêcher le buffer de déborder
  - Sliding Windows
    - L'expéditeur garde une liste des segments qu'il est autorisé à envoyer
    - Types de segments :
      - Acked segments : segments dont on a déjà reçu le message et dont on a déjà reçu un accusé de réception
      - Unacknowledge segments : message en attente d'un accusé de réception
      - Available sequence nums : placé dans la sliding windows



- Le destinataire garde également une fenêtre contenant les numéros de séquence acceptable
- Le destinataire et l'expéditeur doivent utiliser des fenêtres compatibles
  - En l'occurrence, la fenêtre de réception doit être plus grande que celle d'expédition
- Lors de l'arrivée d'un accusé de réception, la fenêtre avance
- Lorsque j'arrive au dernier numéro de séquence, je retourne au premier numéro de s équence



- Encoder les numéros de séquence
  - Problème : combien de bits avons-nous dans l'en-tête du segment afin d'encoder ce numéro de séquence
  - Solution : Placer à l'intérieur de chaque segment transmis son numéro de séquence modulo 2N
  - Le même numéro de séquence pourra être utilisé pour différents segments

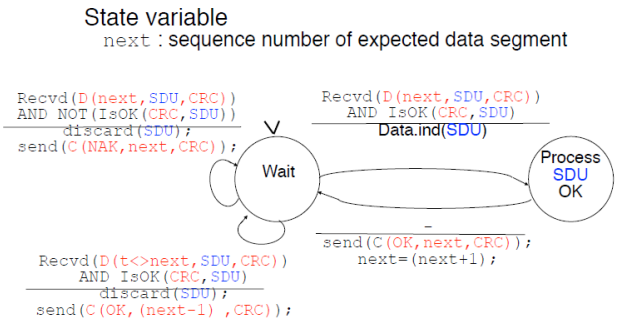


- Mode de transfert fiable avec une fenêtre déroulante
  - Go-Back-N
    - Implémentation simple, en particulier du coté de la réception, le débit sera limité lorsqu'une perte apparait
  - Selective Repeat
    - Plus difficile à implémenter mais le débit restera haut même en cas de perte

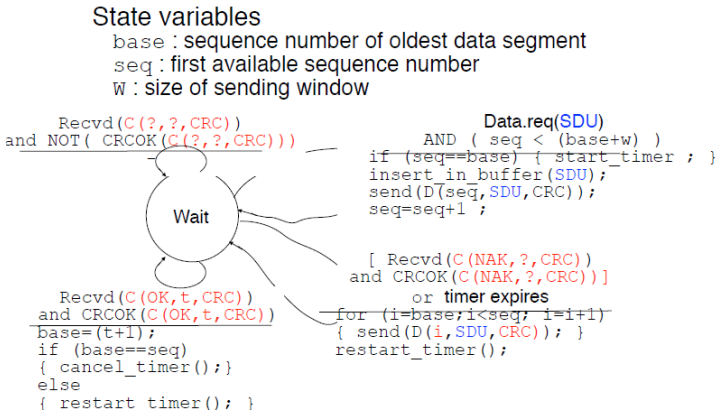
GO-BACK-N

- L'algorithme du destinataire doit être le plus simple possible
- Destinataire

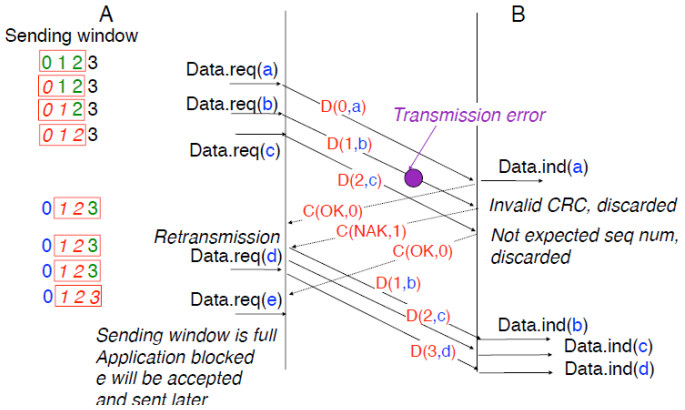
- Il n'accepte que les segments de données ayant un numéro de séquence consécutif
  - Lors de la réception d'un segments de données, on envoie :
    - OKX signifie que tous les segments de données jusqu'à ainsi que X ont été reçu correctement
    - NAKX signifie que le segment de données contenant le numéro de séquence X contient une erreur où à été perdus

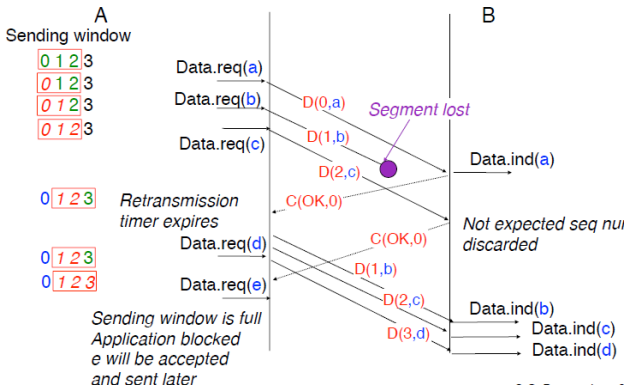


- Expéditeur
  - Se base sur un timer de retransmission afin de détecter les segments perdus
  - Lors de l'expiration du temps de retransmission ou lors de l'arrivée sur segment NAK, on retransmet tous les segments de données non vérifiés
    - L'expéditeur peut donc renvoyer un segment qui a déjà été recu mais qui est hors de la séquence à la destination



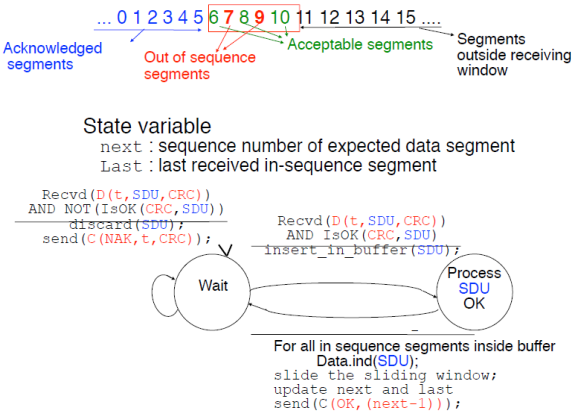
- Exemple





Selective Repeat

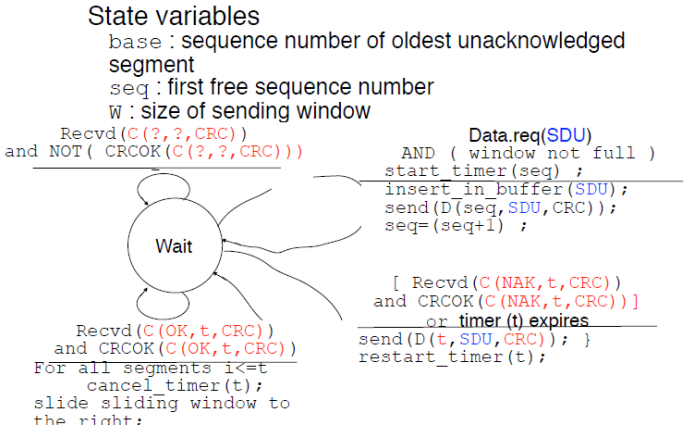
- Destinataire
  - Utilise un buffer afin de stocker les segments reçus en dehors de la séquence et réordonner leur contenu



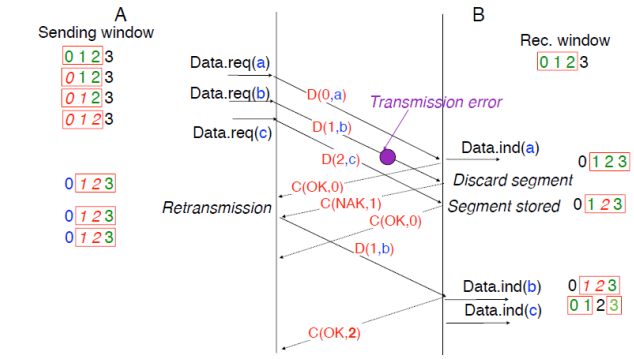
  - Sémantique des segments de contrôle
    - OKX
      - Les segments jusqu'à ainsi qu'au numéro de séquence X ont été correctement reçus
    - NAKX

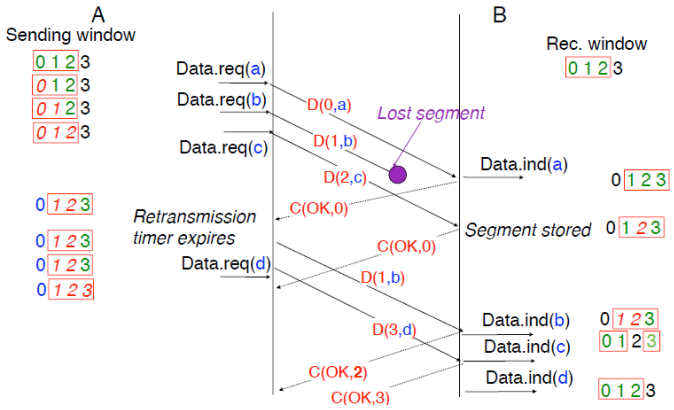
- Le segment avec le numéro de séquence X contient une erreur

- Expéditeur
  - Lors de la détection d'un segments erronés ou lors d'un segment perdus, l' expéditeur renverra uniquement ce segment
    - On aura donc besoin d'un timer de retransmission par segment



- Exemple



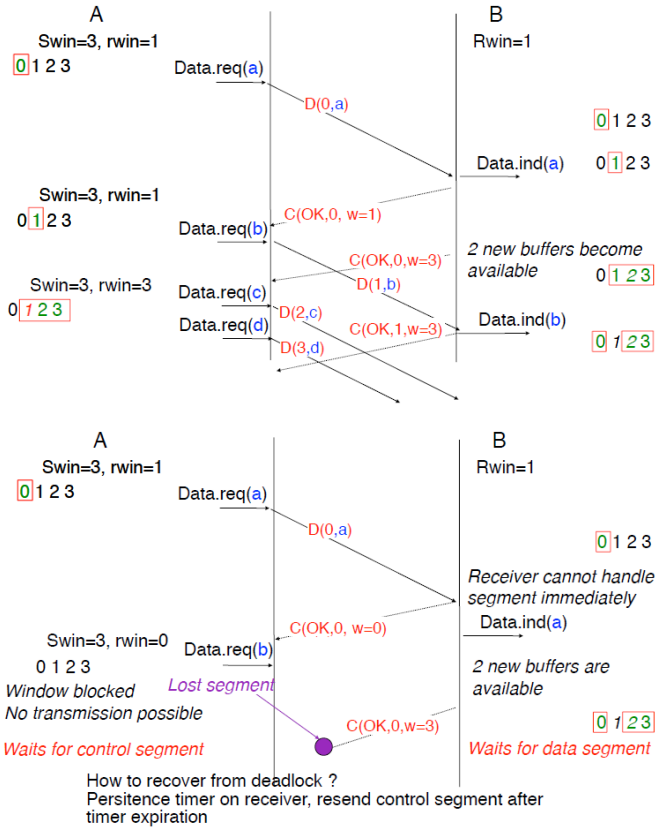


Buffer Management

- Une entité de transport peut supporter plusieurs connexions de transport en même temps
  - Comment peut-on partager les buffers disponibles parmi ces connexions
  - Le nombre de connexions change avec le temps

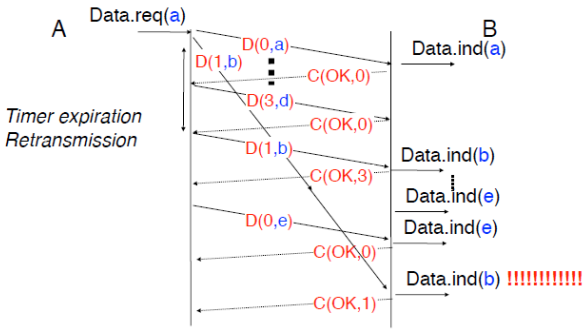
- Certaines connexions nécessitent de grands buffers tandis que d'autres en ont besoin de plus petits

- On va ajuster la taille de la fenêtre du destinataire en fonction du montant de buffer du receveur et permettre au destinataire de prévenir l'expéditeur de la taille de son buffer.
  - On va posséder une nouvelle information : win (indique la taille actuelle du buffer)
  - Changements au niveau de l'expéditeur
    - Swin = taille maximum des fenêtres de l'envoyeur
    - Rwin = taille maximum des fenêtres du receveur À n'importe quel moment, l'expéditeur est autorisé à envoyer des segments de données dont la séquence est inclue entre min(rwin, swin)
      - C'est la plus petite qui définit la taille éffective
      - Il se peut que la taille d'une des fenêtres passe à 0



Duplication et ordonnancement

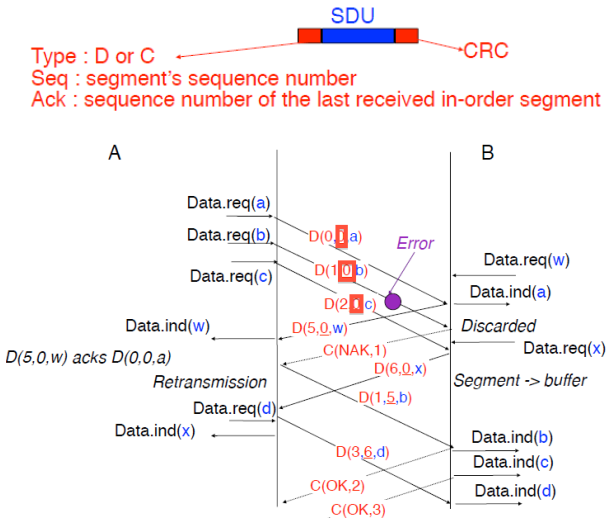
- Problème : un segment tardif pourrait être confondus avec un segment valide



- Comment gérer cela ?
  - Possible à condition que les segments ne restent pas toujours à l'intérieur du réseau
  - Contrainte au niveau de la couche réseau
    - Un paquet ne peut pas rester à l'intérieur du réseau plus de MSL secondes
  - Principe de la solution
    - Seul un segment portant le numéro de séquence X ne peut être retransmis durant MSL secondes

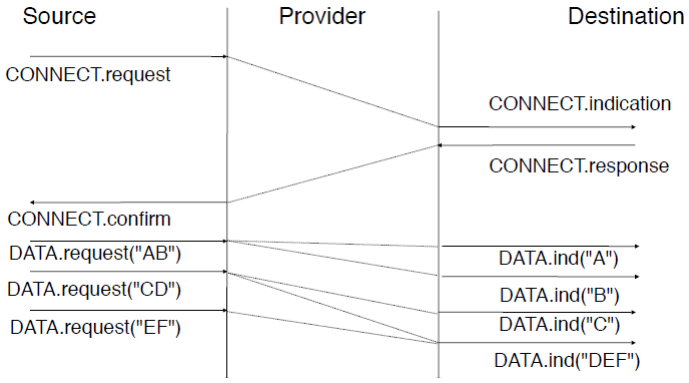
Bidirectionnal Flow

- Comment permettre aux deux hôtes de transmettre des données ?
  - Chaque hôte envoie et des segments de données et des segments de contrôle
  - Piggybacking
    - Placer des champes de contrôle à l'intérieur des segments de données comme on le veut de sorte que le segment de données peut porter des informations de contrôle.
    - Réduit la surchauffe de retransmission



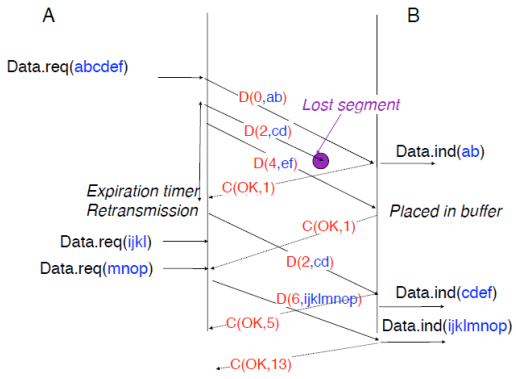
Data transfer : stream mode

- Les fournisseurs livrent un flux de caractères de la source à la destination



Byte stream service

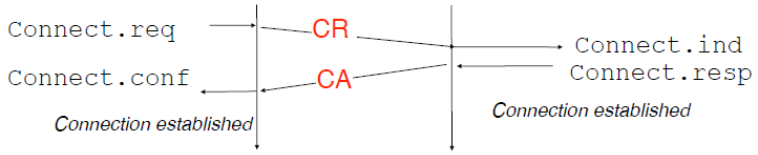
- Comment fournir un flux de byte
  - L'expéditeur divise le flux de bytes en segments
  - LE destinataire livre la charge utile ou les numéros reçus dans la séquence à son utilisateur
  - Généralement chaque octet du flux de byte à son propre numéro de séquence. L'en-tête du segment contient le numéro de séquence du premier byte de la charge utile.
  - On donne un numéro de séquence par caractère
    - On reçoit toutes les lettres envoyées dans le bon ordre
      - Le fait que cela soit envoyé par paquet importe peu car seul l'ordre est important
  - Buffer du côté du receveur
    - Plus simple
    - Plus rapide



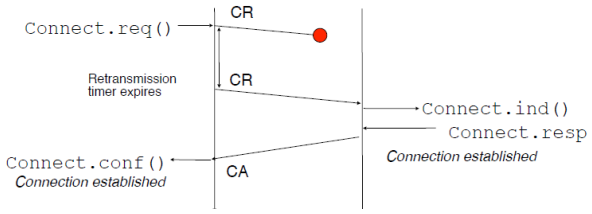
Transport connection establishment

- Comment ouvrir une connexion de transport entre deux entités de transport
  - La couche de transport utilise la couche imparfaite qu'est la couche réseau
    - Des erreurs de transmission sont possible
    - Des segments peuvent être perdus
    - Des segments peuvent être réordonnés

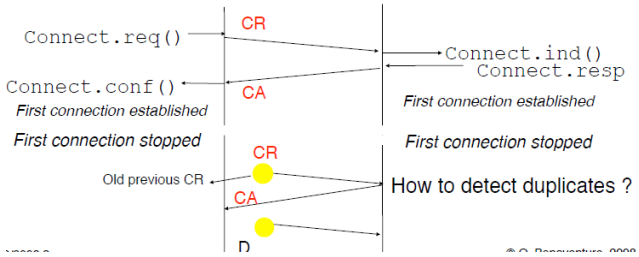
- Des segments peuvent être duppliqués
- Hypothèse
  - Une connexion de transport simple doit être établie entre deux entités de transport
- Solution simple
  - 2 segments de contrôle
    - CR sera utilisé afin de demander l'établissement d'une connexion
    - CA sera utilisé afin de confirmer l'établissement d'une connexion



- Comment gérer les erreurs de transmission et les pertes
  - Les segments de contrôle doivent être protégés par CRC ou par checksum
  - Le timer de retransmission est utilisé afin de se protéger face aux pertes de segments



- Comment gérer les paquets dupliqués ou différés
  - Un CR dupliqué ne devrait pas mener à l'établissement de deux connections de transport au lieu d'une

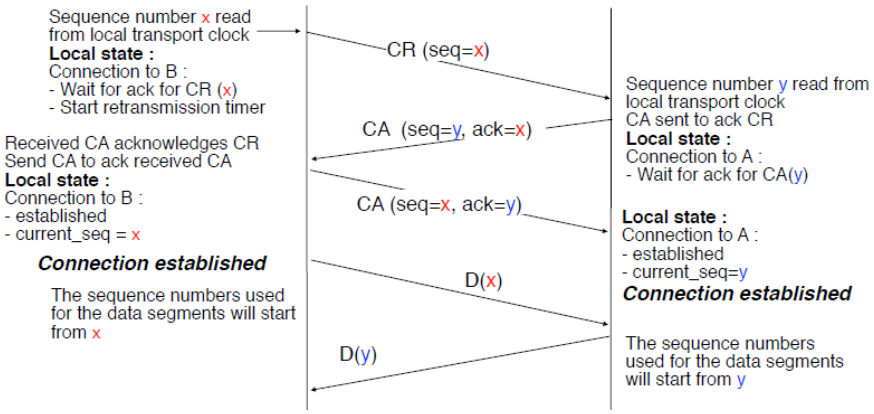


  - La couche réseau garantit par son protocole son organisation interne que son paquet et sa duplication ne vont pas vivre éternellement à l'intérieur du réseau.
    - Aucun paquet ne va survivre plus de MSL secondes à l'intérieur du réseau
  - Les entités de transport se fie à une horloge locale afin de détecter les duplications de demande d'établissement de connexion.
- Horloge de transport
  - Maintenue par chacune des entités de transport
    - 2^K * clock cycle >> MSL

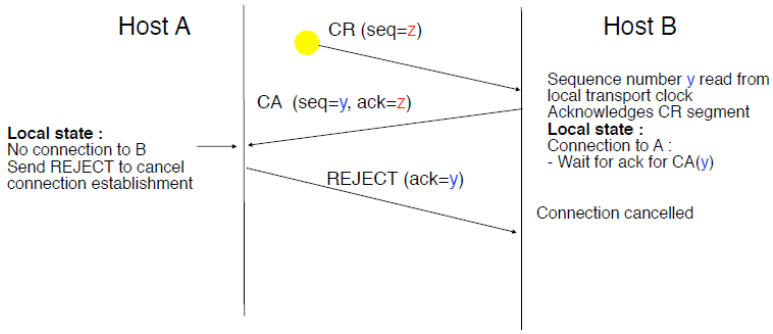
- Doit continuer de compter même si l'entité de transport s'arrête où se redémarre
- Les horloges de transport ne sont pas synchronisées
    - Aussi bien avec une autre horloge de transport ou avec le temps réel

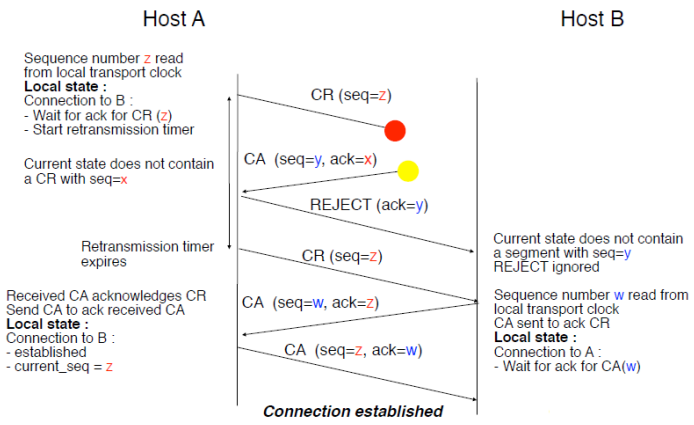
Three Way Handshake

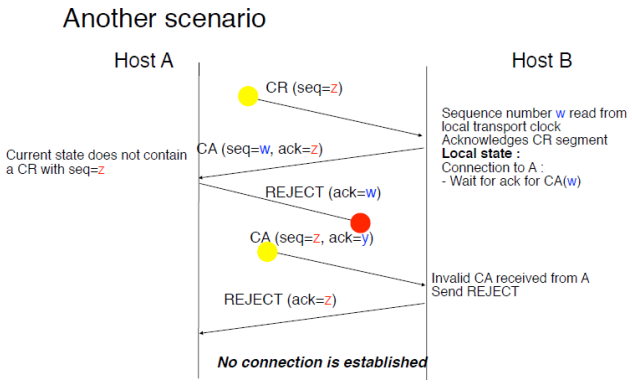
- Cas general



- Que se passe-t-il dans le cas de CR dupliqués

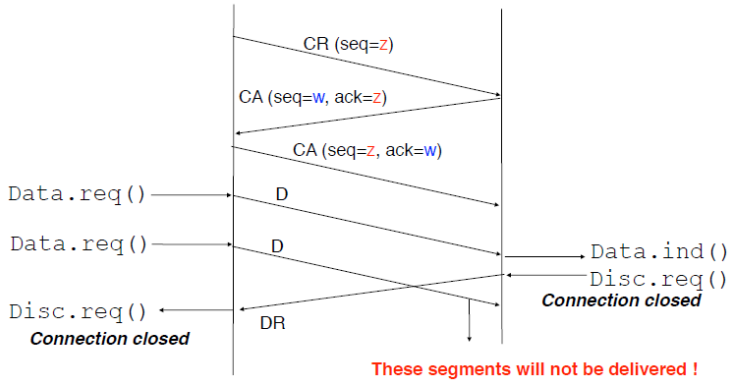






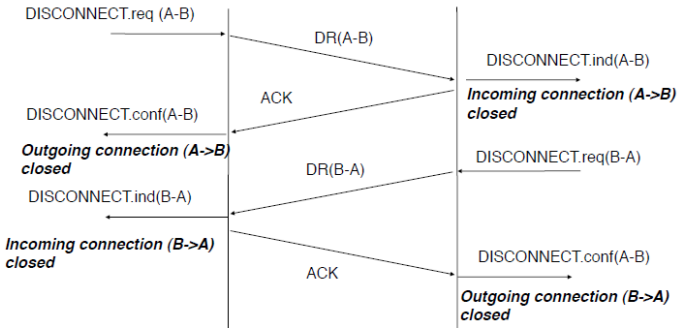
Connection Release

- Une connection de transport peut être utilisée dans les deux directions
- Types de fin de connexion
  - Une fin de connexion abrupte
    - L'une des entités de transport ferme les deux directions du transfert de données
    - Peut mener à la perte de données



    - Une entité de couche de transport pourrait elle-même être forcée à mettre fin à la connexion de transport
      - Le même segment de données a été transmis de multiples fois sans recevoir une confirmation
      - La couche réseau rapporte que l'hôte de destination n'est plus atteignable
      - L'entité de la couche de transport n'a pas assez de ressources disponibles afin de supporter cette connexion
  - Une fin de connexion gracieuse
    - Chaque entité de transport ferme sa propre direction de transfert de données
    - La connexion sera fermée lorsque toutes les données auront été correctement livrées
      - Lorsque A finis de parler à B il dit je finis de parler dans le sens A -> B
        - Par contre il accepte encore les messages de B
        - On peut tjr faire passer les messages dans l'autres sens
      - Lorsque B finis de parler il ferme le sens B -> A

- Lorsque les deux sens sont fermés alors la déconnexion est totale
- En pratique, au moment de la déconnexion A et B envoie le dernier numéro de séquence qui à été envoyé
    - De telle sorte que B sache quel est le dernier message qui devrait être distribué à la couche de l'application
        - Pour prévenir le message de se perdre



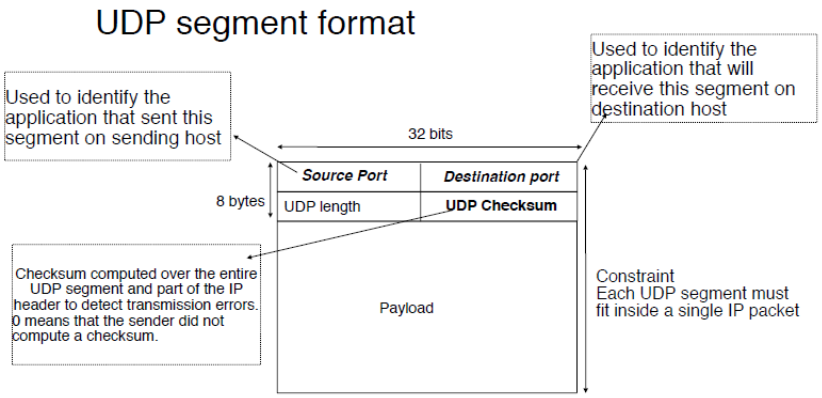
Reliability of the transport layer

- Limitations
    - La couche de transport fournis un transfert de données fiable durant la durée de vie de la connexion de transport
        - Si une connexion est fermée gracieusement alors toutes les données envoyées sur cette connexion ont été reçues correctement.
        - Le transfert de données peut être non fiable tant le cas où la connexion se termine de manière abrupte
- La couche de transport ne se récupère pas d'elle-même lors d'une fin de connexion abrupte
    - Solutions possibles
        - L'application réouvre la connexion et redémarre le transfert de données
        - Couche Session
        - Transaction en cours

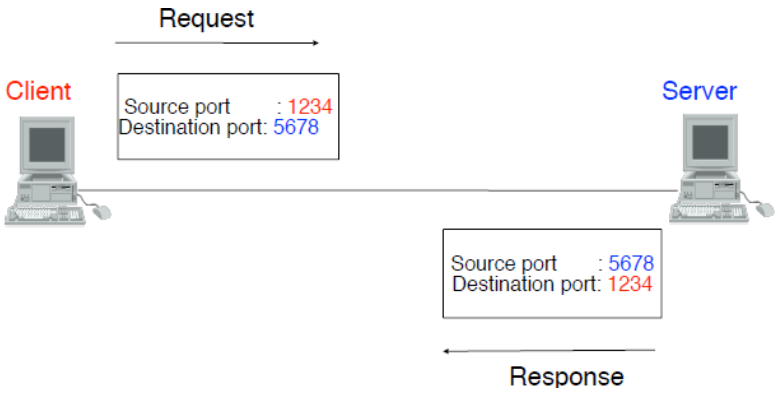
UDP : a simple connectionless transport protocol

- User Datagram Protocol (UDP)
    - Le protocole de transport le plus simple
    - But : permet aux applications d'échanger de petits SDU's en se fiant sur le service IP
        - Sur la plupart des systèmes d'opérations, envoyé des paquets d'IP bruts demande des privilèges spéciaux tandis que n'importe qu'elle autre application pourrait utiliser directement le service de transport
    - Contrainte : l'implémentation de l'entité de transport UDP devrait rester le plus simple possible
- Quels mécanismes au sein de l'UDP
    - Identification de l'application
        - Plusieurs applications fonctionnant sur le même hébergeur devraient pouvoir utiliser le service UDP

        - On doit attribuer un port à chaque programme
        - Par défaut, pour les serveurs Web, on utilise le port 80
    - Solution
        - Un port source pour identifier les applications expéditrices
        - Un port de destination pour identifier les applications destinataires
        - Chaque segment UDP contient le port source ainsi que le port de destination
    - Détection des erreurs de transmission
- UDP Segment Format
    - Décomposer en deux parties
        - L'en-tête
        - Le message en tant que tel (PayLoad)



- Utilisation des ports UDP

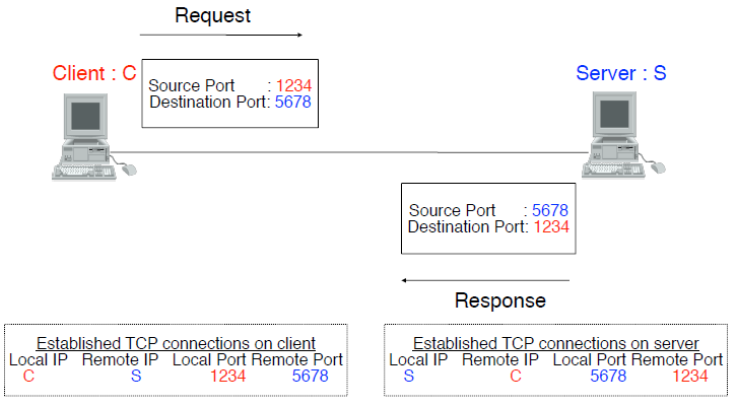


- Limitations du service UDP
    - La taille maximale des SDU's d'UDP dépend de la taille maximale des paquets IP
    - Service sans connexion non-fiable
        - Des SDU's peuvent être perdus mais les erreurs de transmission peuvent être détectées
    - L'UDP ne préserve pas l'ordonnancement
    - L'UDP ne détecte pas ou ne prévient pas la duplication
- Utilisation de UDP

- Application de Demande-réponse, où les requêtes ainsi que les réponses sont courtes et des délais courts sont requis ou utilisés dans des environnements LAN.
- Transferts multimédias où une livraison fiable n'est pas nécessaire et des retransmissions causeraient de trop longs délais.

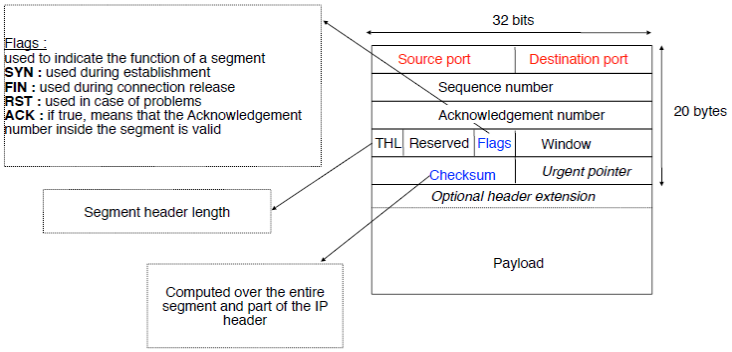
TCP : a reliable connection oriented transport protocol

- TCP
  - Transmission Control Protocol
  - Fournis un service de flux de bytes fiable
  - Caractéristiques du service TCP
    - Connexion TCP
    - Le transfert de données est fiable
      - Pas de pertes
      - Pas d'erreurs
      - Pas de dupplications
    - Le transfert de données est bidirectionnel
    - TCP se fie sur le service IP
    - TCP ne supporte que l'UNICAST
- Connexion TCP
  - Comment identifier une connexion TCP
    - Adresse de l'application source
      - Adresse IP de l'hébergeur source
      - Le numéro de port TCP de l'application sur l'hébergeur source
    - Adresse de l'application de destination
      - Adresse IP de l'hébergeur de destination
      - Le numéro de port de TCP de l'application sur l'hébergeur de destination
  - Chaque segment TCP contient l'identification de la connexion à laquelle il appartient

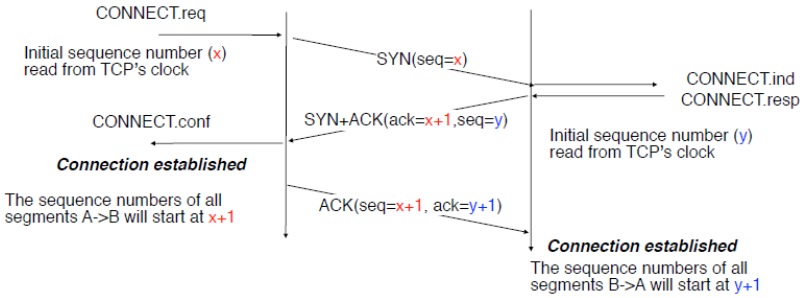


- Format du Segment

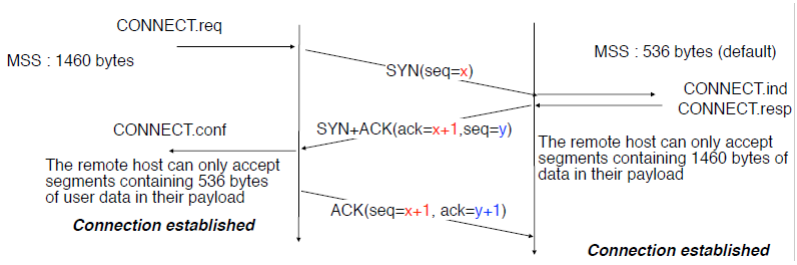
Single segment format



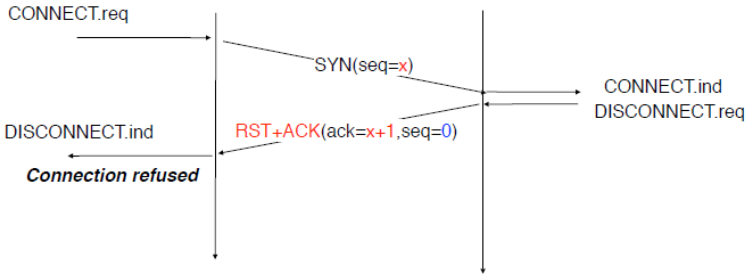
- Établissement de la connexion TCP
  - Three Way Handshake



    - Remarques
      - Le positionnement du drapeau SYN dans le segment consumme un numéro de séquence
      - Le flag ACK n'est positionné seulement lorsque le champ de vérification contient une valeur valide
      - La recommandation par défaut pour l'horloge TCP doit être incrémentée par 1 au moins après 4 microsecondes et après chaque établissement d'une connexion TCP
  - Option de négociation
    - Lors de l'ouverture de la connexion, il est possible de négocier à propos de l'utilisation des extensions TCP
      - Ces options sont encodées à l'intérieur d'une partie optionnelle de l'en-tête TCP

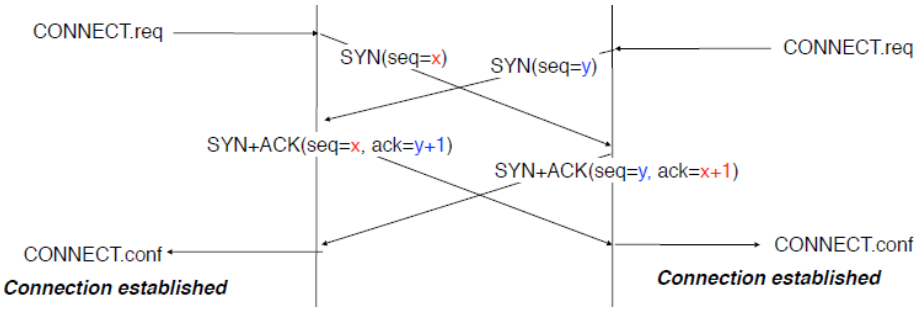


- Rejet de l'établissement de la connexion

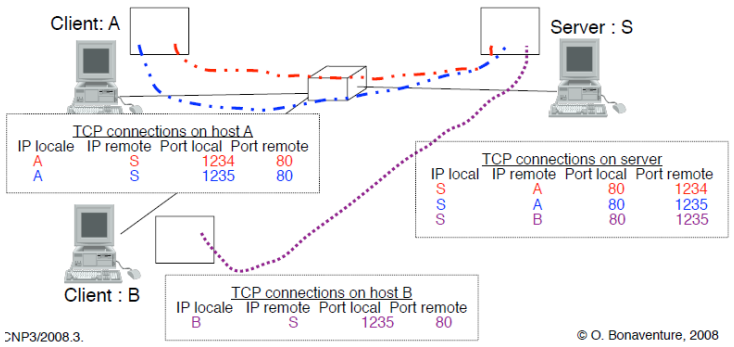


A TCP entity should never send a RST segment upon reception of another RST segment

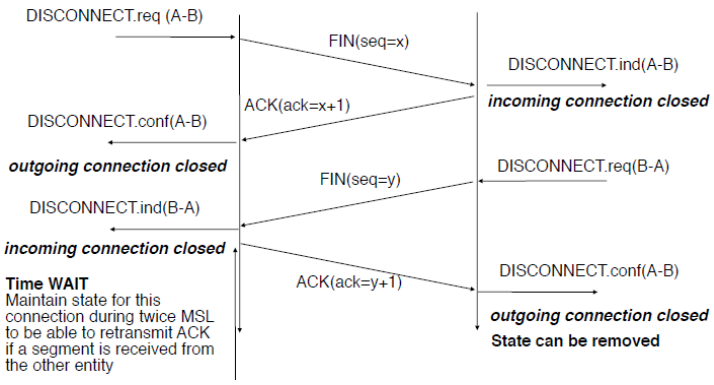
- Établissements simultanés



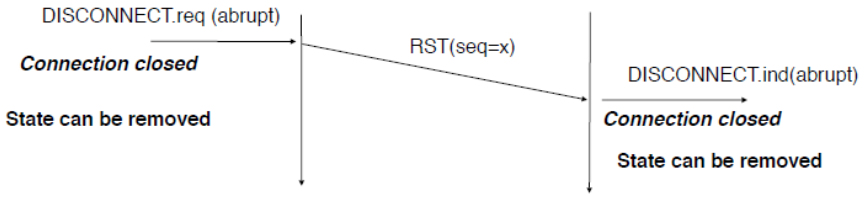
- Comment ouvrir différentes connexions TCP au même moment



- Fin de connexion
  - Fin gracieuse

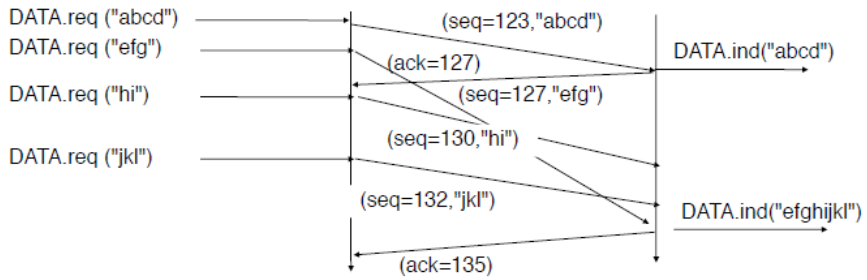


- Fin abrupte

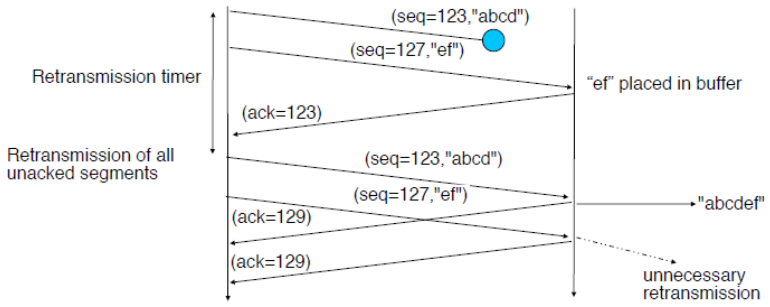


  - Remarques
    - Des segments de données peuvent être perdus durant une telle fin abrupte
    - Aucune entité ne doit attendre dans un état TIME_WAIT après une telle fin
      - Dans tous les cas, n'importe quel segment reçu lorsqu'il n'y a aucun état provoque la transmission d'un segment RST.

- Chaque segment TCP contient
  - 16 bits pour le checksum
    - Utilisé pour détecter les erreurs de transmission affectant la charge de travail utile
  - 32 bits de numéro de séquence (un byte = un numéro de séquence)
    - Utilisé par l'expéditeur pour délimiter les segments à envoyer
    - Utilisé par le destinataire afin de réorganisé es segments reçus
  - 32 bits de numéro de vérification
    - Utilisé par le destinataire pour faire parvenir au numéro de séquence, le prochain byte attendus (dernier byte reçu dans la séquence + 1)

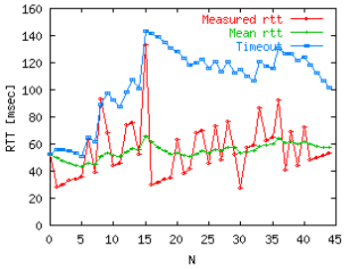


- Comment gérer les pertes de segments
  - TCP utilise un timer de retransmission
    - Si le timer de retransmission arrive à expiration, TCP effectue un GO-Back-N et retransmet tous les segments non vérifiés
    - Généralement un seul timer de retransmission fonctionne à un moment donner

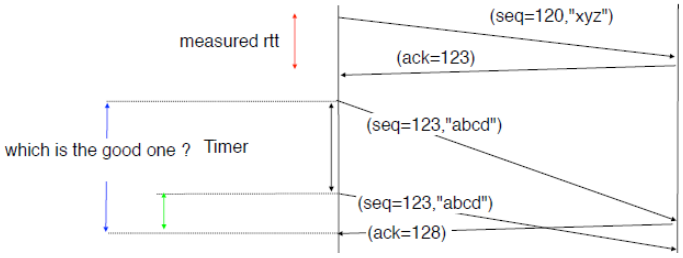


  - Timer de retransmission
    - Comment le calculer
      - Le temps de parcours peut changer fréquemment durant la durée de vie de la connexion TCP
      -

TCP's retransmission timer
One timer per connection
timer = mean(rtt) + 4*std_dev(rtt)
Estimation of the mean
est_mean(rtt) = (1-α)*est_mean(rtt) +α*rtt_measured
Estimation of the standard deviation of the rtt
est_std_dev=(1-β)*est_std_dev+β*|rtt_measured - est_mean(rtt)|

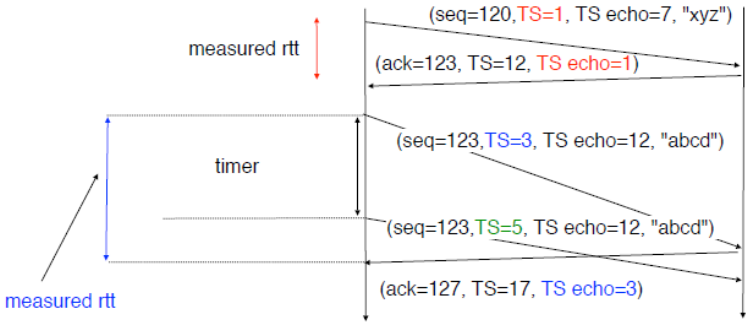


    - Estimation du temps de parcours
      - Comment mesurer le rtt après retransmission

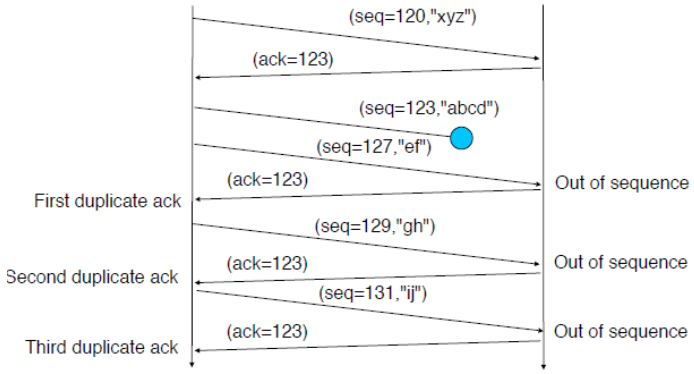


      - Solutions

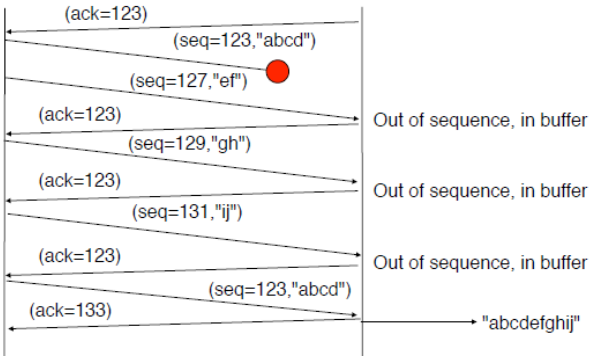
- Karn / Partridge
  - Ne pas mesurer le rtt de retransmission des segments
- Amélioration du Karn Partridge
  - Ajout d'un timestamp dans chaque segment envoyé

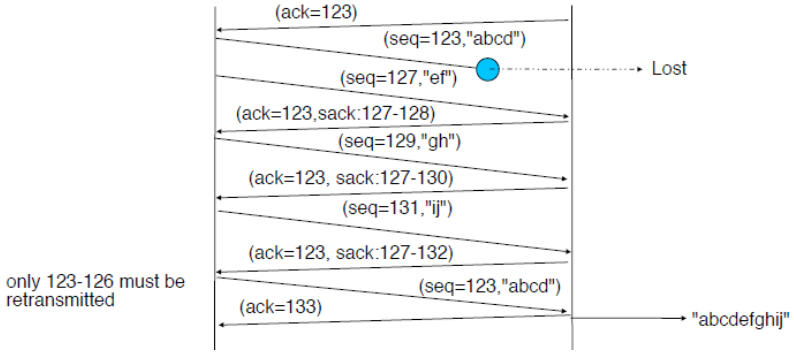


- Amélioration du transfert de données fiable
  - Comment améliorer la réaction dû aux pertes de segments
    - De destinataire TCP devrait envoyer un ACK à chaque fois qu'un segment hors séquence est reçu
    - Heuristique : un segment est considéré comme perdus après l'avoir dupliqué trois fois



  - Comment transmettre à nouveau les segments perdus
    - Lors de la réception de trois ACK dupliqués, on retransmet le premier segment non vérifié (qui ne possède pas de ACK)
      - Moyen de retransmission rapide utilisé par la plupart des implémentations TCP





The network layer

- Le service de la couche réseau sur internet
  - Service sans connexion non fiable
    - Les paquets peuvent être modifiés
      - Perdus à cause d'interférences électromagnétiques
    - Les paquets peuvent être perdus
    - Les paquets peuvent souffrir d'erreurs de transmission
    - L'ordre des paquets n'est pas préservé
    - Les paquets ne peuvent pas être dupliqués
    - La taille des paquets est limitée à 64 KBytes
    - Problèmes de décalage d'horloge
      - Risque de perte
  - Comment construire un service utilisable par les applications