3.E : Protocoles de transport (UDP/TCP) et ICMP

Olivier GLÜCK
Université LYON 1 / Département Informatique
Olivier.Gluck@univ-lyon1.fr
http://perso.univ-lyon1.fr/olivier.gluck

1

Les protocoles de transport : UDP et TCP

2

Le protocole UDP et la rapidité

L'en-tête UDP
Le mode non connecté
Quelles applications utilisent UDP ?

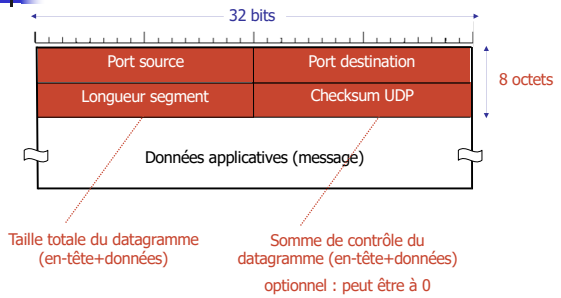
3

Le protocole UDP

- UDP (RFC 768) - User Datagram Protocol
  - protocole de transport le plus simple
  - service de type best-effort (comme IP)
    - les datagrammes UDP peuvent être perdus
    - les datagrammes UDP peuvent arriver dans le désordre
  - mode non connecté : chaque datagramme UDP est traité indépendamment des autres
- Pourquoi un service **non fiable sans connexion** ?
  - simple donc **rapide** (pas de délai de connexion, pas d'état entre émetteur/récepteur)
  - petit en-tête donc économie de bande passante
  - UDP peut émettre aussi rapidement qu'il le souhaite : pas de limite à l'envoi contrairement à TCP (contrôle de congestion)

4

L'en-tête UDP : 8 octets

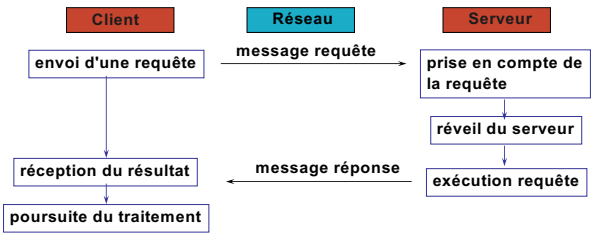


UDP = IP + numéros de port !!

5

UDP est en mode non connecté

Permet d'envoyer rapidement des messages mais sans l'autorisation du destinataire donc aucune garantie sur la réception du message



Exemple de mode non connecté : courrier postal, mail

Tous les protocoles d'Internet sont en mode non connecté SAUF TCP

6

Quelles applications utilisent UDP ?

- Pour les applications qui ont besoin d'envoyer rapidement
  - UDP permet des envois rapides sans limitation mais sans garantie donc sans fiabilité
- Souvent utilisé pour les **applications multimédias**
  - Vidéos, son, musique, streaming, visioconférence, voix sur IP
  - Ces applications sont tolérantes aux pertes/erreurs et sensibles au débit (les données doivent arriver à la bonne vitesse)
- Autres utilisations d'UDP
  - Applications qui envoient peu de données et qui ont besoin de rapidité
  - exemple : **DNS**
- Transfert fiable sur UDP
  - Comme UDP n'apporte aucune garantie, l'application peut ajouter des mécanismes pour réparer les pertes ou erreurs (acquittements…)

7

Lyon 1

Le protocole TCP et la fiabilité

Qu'est-ce que la fiabilité ?
Les mécanismes de la fiabilité
Le mode connecté
L'en-tête TCP
Quelles applications utilisent TCP ?

8

Le protocole TCP et la fiabilité

- Transport Control Protocol (RFC 793, 1122, 1323, 2018, 2581)
- Transport **fiable** en **mode connecté**
  - entre un client et un serveur : (@IP src, port src) --> (@IP dest, port dest)
  - transporte un flot d'octets (ou flux)
    L'application lit/écrit des octets dans un tampon, TCP assure la fiabilité
- Fiabilité : faire en sorte que tout ce qui arrive est exactement ce qui a été envoyé
  - Les données ne doivent pas être perdues : **sans perte**
  - Les données ne doivent pas subir d'erreurs : **sans erreur**
    Une erreur = un bit qui change de valeur pendant le transfert
  - Les données doivent arriver **dans l'ordre**
  - Les données ne doivent pas arriver en double : **sans duplication**

9

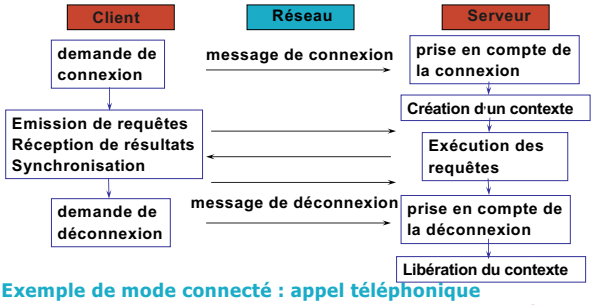
Les mécanismes de la fiabilité

- Fiabilité : **sans perte, sans erreur, dans l'ordre, sans duplication**
- En cas de perte ou erreur, il faut retransmettre si on n'a pas reçu d'acquittement (ACK) au bout d'un certain temps
- Pour détecter une perte ou une duplication, il faut des ACK et numéroter les messages et les ACK
- Pour détecter les erreurs, on utilise les checksum
- Pour retransmettre, il faut conserver les messages envoyés qui n'ont pas encore été acquittés
- Mécanismes : **retransmissions, timeout, ACK, stockage des messages non acquittés, checksum, numérotation des messages et des acquittements**

10

TCP est en mode connecté

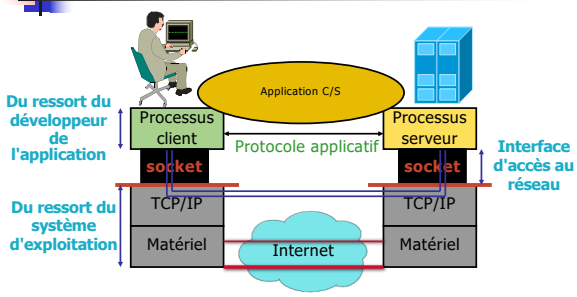
Permet d'envoyer des messages avec fiabilité mais limitation du débit à l'envoi : contrôle de flux et contrôle de congestion



Exemple de mode connecté : appel téléphonique
TCP est le seul protocole d'Internet en mode connecté

11

Les sockets, interface d'accès au réseau

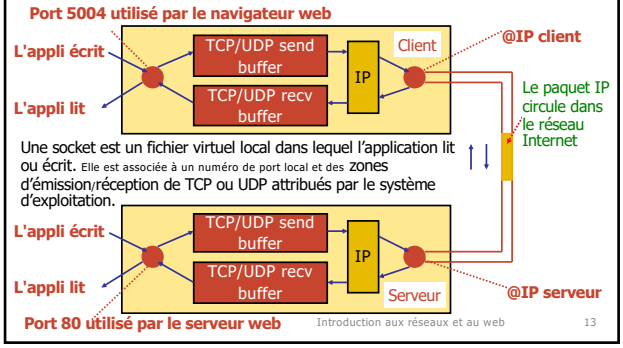


Une socket : interface locale à l'hôte, créée par l'application, contrôlée par l'OS
Porte de communication entre le processus client et le processus serveur

12

Les sockets et les numéros de port

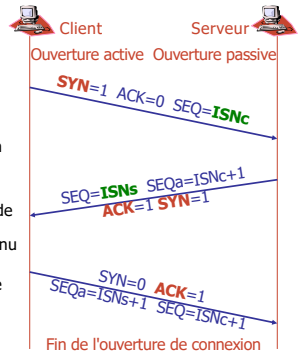
- Un échange Client/Serveur = **(@IP_src,port_src,@IP_dest,port_dest)**



13

Etablissement d'une connexion TCP

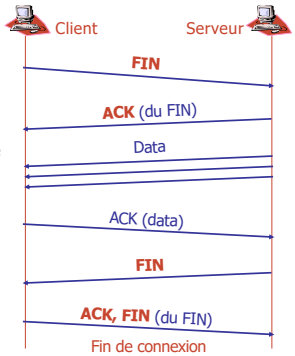
- Connexion en trois phases
  - 1 - demande d'ouverture par le client (SYN), choix ISNc
  - 2 - acceptation par le serveur (SYN+ACK), allocation des tampons, choix ISNs
  - 3 - le client acquitte l'acceptation (ACK)
- Mode Client/Serveur
  - Le TCP du client fait une demande d'ouverture de connexion vers le port du serveur qui doit être connu à l'avance
  - Le TCP du serveur est en attente des demandes d'ouverture de connexion en provenance des clients



14

Fermeture d'une connexion TCP

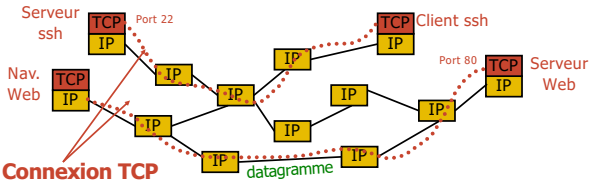
- Fermeture négociée
  - 1 - demande de fin de connexion (FIN) par une des extrémités
  - 2 - acquittement du FIN (ACK) mais mise en attente de la demande (Le serveur a encore des données non transmises)
  - 3 - envoi des données en attente
  - 4 - acquittement des données (ACK)
  - 5 - acceptation de la fin de connexion par le serveur (FIN)
  - 6 - acquittement de la fin de connexion (ACK)



15

Caractéristiques du protocole TCP

Couche transport : communications entre applis



- TCP - protocole de transport **de bout en bout**
  - uniquement présent **aux extrémités**
  - transport **fiable** de **segments** (mode **connecté**)
  - protocole complexe (retransmission, gestion des erreurs, séquencement, …)

16

Quelles applications utilisent TCP ?

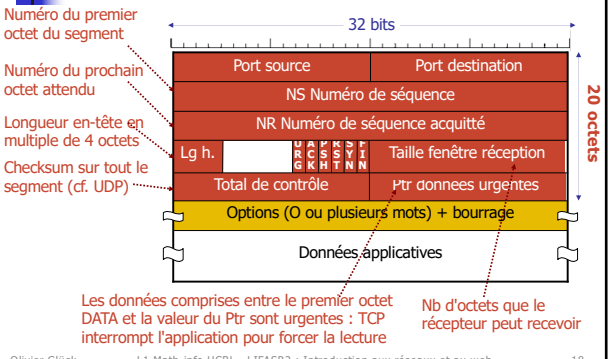
Toutes celles qui ne peuvent pas se passer de la fiabilité c'est à dire presque toutes !

- Le web (**HTTP**)
- La connexion à distance (**telnet**, **ssh** et **X**)
- Le courrier électronique (**SMTP**, **POP**, **IMAP**, **Webmail**)
- Le transfert de fichiers (**FTP**)
- L'accès aux fichiers distants (**NFS**, **SMB**)
- L'annuaire fédérateur (**LDAP**)

Les applications multimédias et le DNS n'utilisent pas TCP

17

L'en-tête TCP : 20 octets



18

L'en-tête TCP (2)

- Numéro de séquence NS (émission)
  - comptabilise les **octets** depuis le début de la connexion
  - ISN : numéro de séquence initial, valeur "aléatoire" acquittée lors de l'établissement de la connexion
  - le numéro de séquence du premier octet transmis est ISN+1 puis NS=ISN+nb_octets_transmis+1
- Numéro de séquence NR (réception)
  - le récepteur renvoie le numéro du prochain octet attendu soit NS_reçu+taille_données_reçues

19

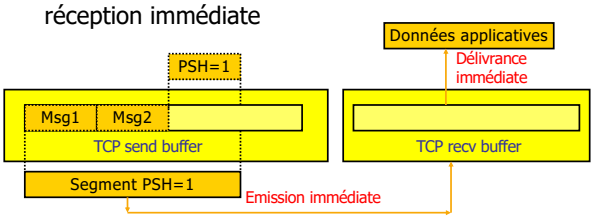
L'en-tête TCP (3)

- Les 6 drapeaux
  - **URG** : valide le champ "Ptr données urgentes"
  - **ACK** : valide le champ NR
  - **PSH** : PUSH indique au récepteur de délivrer immédiatement les données en attente sur le récepteur
    - TCP peut attendre d'avoir suffisamment de données avant de constituer un fragment (efficacité du protocole)
    - exemple : retour chariot (CR) dans un terminal virtuel
  - **RST** : demande au destinataire de réinitialiser la connexion ou rejet d'une demande de connexion
  - **SYN** : demande de connexion (échange des ISN)
  - **FIN** : demande de déconnexion (le destinataire n'est pas obligé de s'exécuter : fermeture négociée)

20

Le bit PUSH

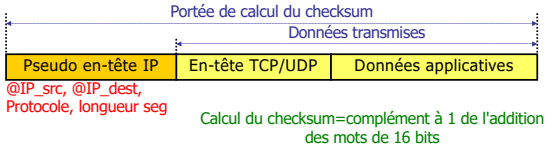
- Pour optimiser la transmission, par défaut TCP attend que le tampon d'émission soit plein pour constituer un segment (groupage de messages)
- Le bit PUSH sert à demander la transmission et réception immédiate



21

Le contrôle d'erreur dans UDP et TCP

- Deux objectifs
  - vérifier que les données transmises n'ont pas été altérées
  - garantir que les données sont transmises au bon destinataire --> rajout d'un pseudo en-tête IP pour le calcul du checksum (qui est non transmis)



22

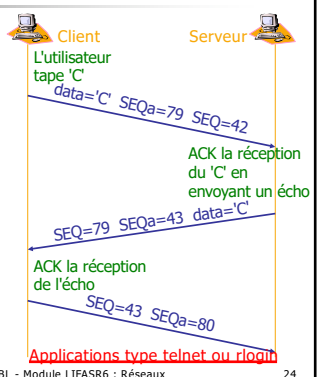
Exemple de calcul de checksum

- Calcul du checksum par additions des mots de 16 bits complémentées à 1
- Exemple : checksum sur 3 mots de 16 bits
  - 0110011001100110
  - 0101010101010101
  - 0000111100001111
- Somme des deux premiers mots
  - 1011101110111011
- Addition du troisième mot
  - 1100101011001010
- Complément à 1
  - 0011010100110101 (= le champ checksum)
- Si pas d'erreur, la somme de tous les mots de 16 bits reçus doit faire 1111111111111111

23

Numéro de séquence et ACK

- Principe du *piggybacking* : un segment peut contenir des données et acquitter un segment précédent
- SEQ=numéro du premier octet dans le segment depuis l'ouverture de la connexion
- SEQa=numéro du prochain octet attendu
- L'acquittement d'un octet acquitte tous les octets précédents



24

Le numéro de séquence initial (ISN)

- Chaque entité communique son ISN à l'autre à l'ouverture de la connexion
- Il permet de distinguer les octets de deux connexions successives utilisant les mêmes adresses de transport
  - ouverture de la connexion (@IP1,p1,@IP2,p2)
  - échanges de segments
  - fermeture de la connexion (@IP1,p1,@IP2,p2)
  - ouverture de la connexion (@IP1,p1,@IP2,p2)
  - …
- Un même ISN est tiré toutes les 4h30 environ

25

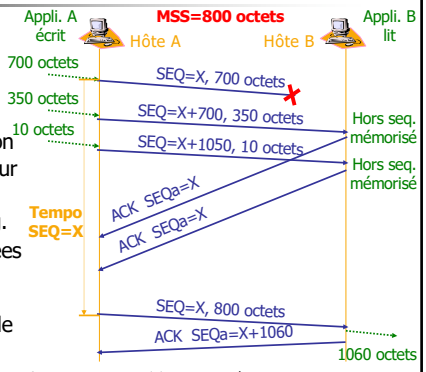
Taille des segments TCP

- Souplesse de TCP :
  - l'application lit/écrit dans un tampon
  - TCP décide quand envoyer/restituer un segment (si PSH n'est pas positionné)
- Idée :
  - TCP a intérêt d'envoyer des segments de taille maximale (limitation de l'overhead lié à la taille de l'en-tête)
  - fragmentation IP coûteuse --> éviter la fragmentation
  - la taille max. d'un segment TCP est de 64Ko (à cause d'IP)
- MSS : Maximum Segment Size (sans en-tête)
  - à l'ouverture de la connexion, chaque entité peut annoncer (option TCP) son MSS à l'autre, en fonction de la MTU de son réseau (MSS=MTU-40) (20,IP+20,TCP)
  - par défaut, MSS=536 octets

26

Politique de retransmissions

- Les pertes de segment sont détectées par absence d'ack positif à expiration d'un temporisateur sur l'émetteur
- 1 tempo. par seg.
- Toutes les données reçues hors séquence sont mémorisées par le destinataire



27

Envois des ACK (RFC 1122, 2581)

- Idée : essayer de ne pas envoyer d'ACK sans données, faire des acquittements cumulés

Evénement sur le récepteur	Action du récepteur
Arrivée d'un segment, dans l'ordre (sans trou), les octets précédents ayant étés acquittés	ACK mis en attente. Envoi de l'ACK au bout de 500ms s'il n'y a pas eu de données à envoyer entre temps
Arrivée d'un segment, dans l'ordre (sans trou), un ACK d'un segment précédent est déjà en attente	Envoi immédiat d'un ACK qui acquitte l'ensemble (ACK cumulé)
Arrivée d'un segment, dans le désordre (avec un numéro de séquence supérieur à celui attendu : création d'un trou)	Envoi d'un ACK dupliqué : si le récepteur attendait 120, il renvoie un ACK avec SEQa=120
Arrivée d'un segment qui remplit partiellement ou complétement un trou	Envoi immédiat d'un ACK

28

Valeur du temporisateur RTO

- RTO : Retransmission Time Out
- Impact de cette valeur sur les performances
  - valeur trop petite : des retransmissions inutiles ont lieu
  - valeur trop grande : attente trop importante entre deux retransmissions
- TCP fait une estimation du RTT (temps aller/retour) et ajuste **dynamiquement** la valeur du temporisateur en fonction de cette estimation
  - utilisation d'une option TCP :
    - l'émetteur met la valeur de son horloge dans l'option
    - le récepteur fait écho de cette valeur dans l'ACK
    - l'émetteur fait la différence entre son horloge et cette valeur à la réception de l'ACK

29

Retransmissions rapides *(Fast Retransmit)*

- Idée : il est de toute façon pénalisant d'attendre l'expiration du RTO pour retransmettre
  - quand le récepteur reçoit des données hors-séquence, il renvoie immédiatement un ACK indiquant les données qu'il attend
  - si l'émetteur a reçu 3 ACK dupliqués (4 ACK avec le même numéro de séquence attendu), il retransmet sans attendre l'expiration du RTO
- Suppositions de cas de perte :
  - expiration du RTO --> pas d'ACK dupliqués, congestion dans le réseau (plusieurs segments sont perdus)
  - ACK dupliqués --> peu de segments sont perdus (un ACK dupliqué signifie la réception d'un segment)

30

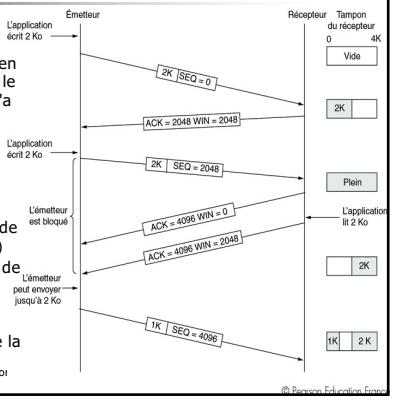
Algorithme de Nagle

- Idée : il est pénalisant d'envoyer des segments contenant 1 seul octet de données (par ex. terminal virtuel)
- principe de Nagle : si l'appli. écrit octet par octet
  - envoi du premier octet dans un segment
  - accumulation dans le tampon des octets suivants tant que le premier octet n'est pas acquitté
  - envoi des octets accumulés dans un seul segment
  - attente de l'acquittement pour envoyer le segment suivant…
- Peut être désactivé dans certains cas (X-Window : mouvements de souris saccadés)

31

Gestion de la fenêtre

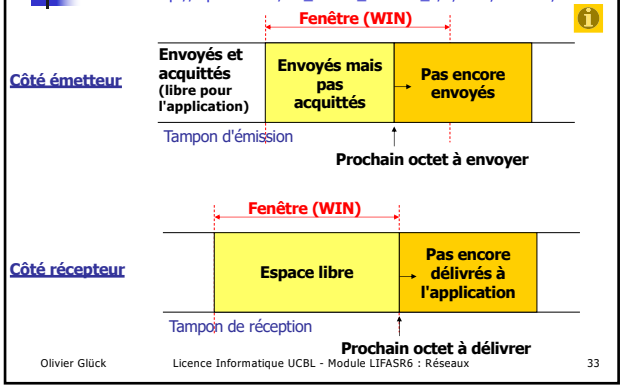
- Contrôle de flux TCP :
  - l'émetteur ne doit pas en envoyer de données si le tampon de réception n'a pas l'espace libre correspondant
- Quand l'émetteur est bloqué, il peut :
  - envoyer des données urgentes (interruption de l'application réceptrice)
  - envoyer des segments de 1 octet pour obliger le récepteur à envoyer SEQa et WIN et maintenir l'état actif de la connexion



32

Contrôle de flux TCP

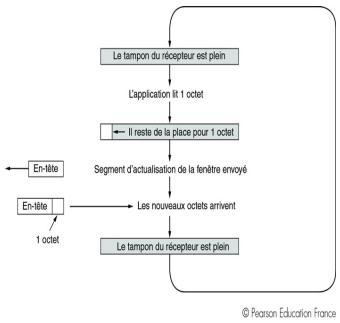
http://wps.aw.com/aw_kurose_network_2/0,7240,227091-,00.html



33

Syndrome de la fenêtre stupide

- Problème lié au fait que l'application réceptrice lit (vide le tampon) octet par octet
  - l'émetteur ne peut alors envoyer que des petits segments
- Solution de Clark :
  - le récepteur n'annonce la réouverture de la fenêtre que lorsqu'une taille suffisante est disponible --> minimum entre MSS et taille_tampon/2



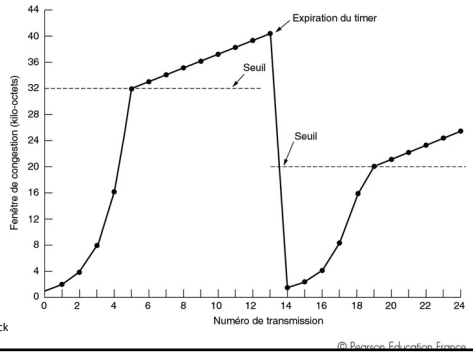
34

Contrôle de congestion dans TCP

- *Congwin* : taille de la fenêtre de congestion
- *Recvwin* : taille de la fenêtre de réception
- La fenêtre d'émission (quantité d'octets que l'émetteur peut envoyer) est *Min(Recvwin,Congwin)*
- *Threshold* : seuil d'évitement de congestion qui définit la limite entre les deux phases suivantes
  - *slow start* : *Congwin* augmente exponentiellement tant que
    - *Threshold* n'est pas atteint
    - une perte se produit
  - *congestion avoidance* : Congwin augmente linéairement tant que pas de perte
- au départ, Threshold=64Ko et Congwin=1*MSS

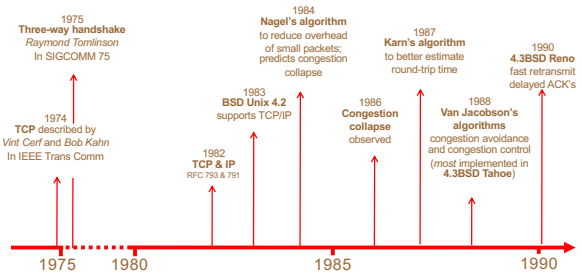
35

Contrôle de congestion dans TCP



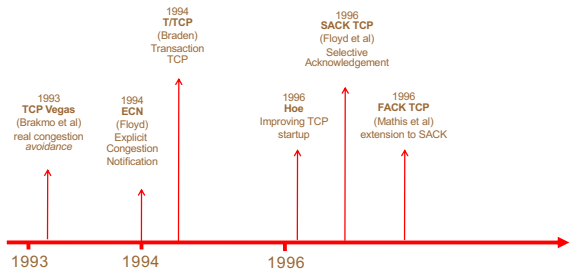
36

Evolutions de TCP (1)



37

Evolutions de TCP (2)



38

Lyon 1

Protocoles de contrôle de l'Internet et utilitaires réseaux

ICMP
ping et traceroute
ARP et RARP
BOOTP et DHCP
Fichiers de config. et commandes UNIX

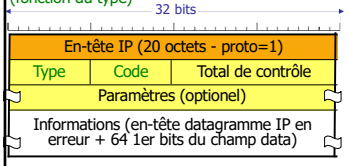
39

ICMP - Internet Control Message Protocol

RFC 792

- Protocole de messages de contrôle de l'Internet
  - échange de messages entre routeurs : signaler une erreur réseau, demande d'information d'état, tests
  - utilisé par des utilitaires (*ping*, *traceroute*, *Network Time Protocol*)
  - permet de pallier au manque de service d'IP

Le champ Code : code d'erreur (fonction du type)



Le champ Type :
0 : réponse d'Echo
3 : destination inconnue
4 : limitation du débit par la source
5 : redirection (ICMP redirect)
8 : demande d'Echo
11 : expiration de délai (TTL=0)
12 : en-tête IP invalide
13/14 : requête/réponse d'horodatage
17/18 : requête/réponse de netmask

40

ICMP - Types de message

- réponse/demande d'Echo : utilisé par *ping*
- réponse/demande d'horodate : idem mais heures incluses pour mesures de performances
- destination inconnue : un routeur ne parvient pas à localiser la destination, problème de fragmentation (bit DF=1), …
- délai expiré : paquet éliminé car TTL a atteint 0 (boucle, congestion, …)
- en-tête IP invalide : la valeur d'un champ IP a une valeur illégale
- ICMP redirect : envoyé par un routeur à un nœud d'extrémité pour signaler une meilleure route (évite la mise à jour manuelle de toutes les tables de routage quand ajout d'un routeur…)
- ralentissement de la source : contrôle de congestion (mais quasiment plus utilisé car génère du trafic supplémentaire -> congestion au niveau TCP)
- autres messages : *www.iana.org/assignments/icmp-parameters*

41

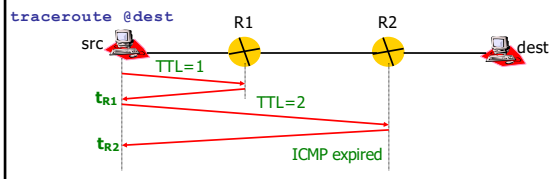
L'utilitaire ping

- Ping : envoi d'un écho, attente de réponse, mesure du temps aller-retour
  - teste l'accessibilité d'une destination **de bout en bout**
  - évaluation de performances
  - la réponse doit parvenir avant 20 secondes
- Exemples :
  - `ping 127.0.0.1` : permet de tester la pile TCP/IP locale (en loopback)
  - `ping mon@IP` : permet de vérifier la configuration réseau locale de la station
  - `ping @default-routeur` : permet de tester la configuration du sous-réseau et de la passerelle
  - `ping @dest` : permet de tester un chemin de bout en bout

42

L'utilitaire traceroute

- Permet de trouver pas à pas le chemin pour atteindre une destination
  - envoi d'un paquet IP avec TTL=1
  - attend ICMP délai expiré
  - envoi d'un paquet IP avec TTL=2, ...



43

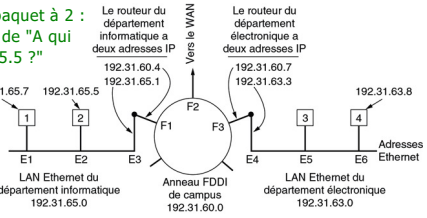
ARP - Address Resolution Protocol

RFC 826

- Problème : les équipements de liaison (cartes réseau...) ne comprennent pas les adresses IP mais utilisent des adresses physiques (MAC)
- Besoin d'associer @MAC <--> @IP

1 veut envoyer un paquet à 2 : diffusion sur le LAN de "A qui appartient 192.31.65.5 ?"

2 répond : "A moi, je suis E2"



44

ARP - Fonctionnement (1)

- Si la machine source et destinataire sont sur le même réseau (par ex. de 1 vers 2)
  - 1 - requête ARP (broadcast MAC)
  - 2 - réponse ARP (le destinataire a reçu le broadcast et s'est reconnu, il envoie son @MAC)
  - 3 - la source peut envoyer ses données vers le destinataire (adresse MAC destination connue)
- Si elles ne sont pas sur le même réseau (par ex. de 1 vers 4)
  - la diffusion ne passe pas le routeur
  - résolution de proche en proche : 1 envoie les données à 192.31.65.1 (ARP pour trouver E3), le routeur info envoie les données à 192.31.60.7 (ARP pour trouver F3), le routeur élec envoie les données à 4 (ARP pour E6)

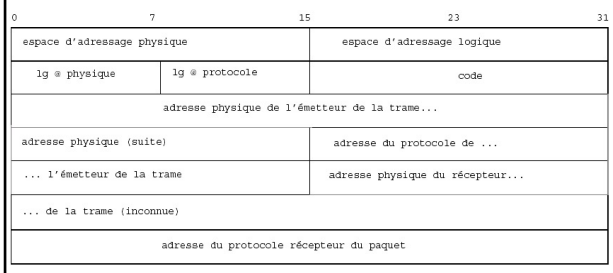
45

ARP - Fonctionnement (2)

- Optimisations
  - Cache ARP : le résultat de chaque résolution est conservé localement pour les émissions suivantes
  - la correspondance (@IP, @MAC) de l'émetteur sont inclus dans la requête ARP pour que le récepteur, voire toutes les machines qui reçoivent le broadcast, mettent à jour leur cache
- Proxy ARP : une machine qui répond à une requête à la place du destinataire (qui ne reçoit pas le broadcast)
  - nécessaire si la route (adresse de la passerelle) pour atteindre le destinataire n'est pas connue

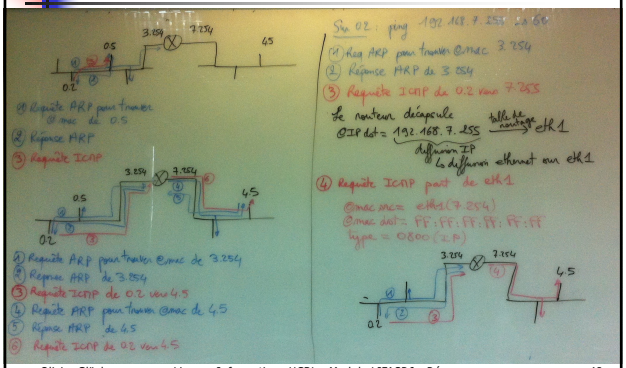
46

ARP - Format du paquet



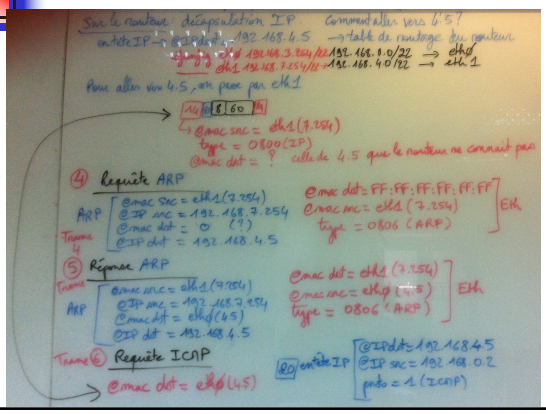
47

ARP - ICMP



48

ARP - ICMP



RARP - Reverse ARP

RFC 903

- ARP : @IP->@MAC RARP : @MAC->@IP
- "Mon @MAC est xx:xx:xx:xx:xx:xx. Quelqu'un connaît-il mon @IP ?"
- permet à un hôte de récupérer son @IP au démarrage par interrogation d'un serveur RARP
  - stations sans disque
  - imprimantes,...
- Même fonctionnement, même format de paquet
- Obsolète car désormais remplacé par BOOTP ou DHCP qui peuvent rendre le même service et ne nécessite pas un serveur RARP sur chaque réseau (*broadcast* MAC limité)

BOOTP (*bootstrap*) - Principe

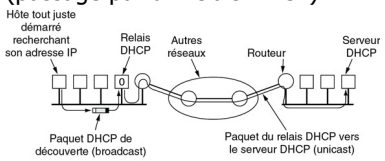
RFC 951, 1048, 1084

- Protocole d'amorçage du réseau au dessus de UDP (les diffusions passent les routeurs)
- le serveur informe la machine qui démarre de
  - son @IP, @IP du serveur de fichiers qui contient son image disque, @IP du routeur par défaut, masque de sous-réseau
- Inconvénient : les tables de correspondances sont statiques (configurées manuellement)
- Pour y remédier, BOOTP est devenu DHCP : *Dynamic Host Configuration Protocol*

DHCP - Principe

RFC 2131, 2132

- Configuration manuelle ou assignation dynamique des adresses IP
- Un serveur spécifique s'occupe d'assigner des configurations réseaux aux hôtes qui en font la demande
- Le serveur n'est pas nécessairement sur le même réseau (passage par un relais DHCP)



DHCP - Fonctionnement

- DHCP - économie d'adresses IP : quand un hôte quitte le réseau, il restitue son adresse
- Les messages DHCP (au dessus d'UDP)
  - **DHCPDiscover** : diffusion du client pour que les serveurs DHCP actifs répondent en fournissant une @IP
  - **DHCPOffer** : offre des serveurs (réponse à DHCPDiscover)
  - **DHCPRequest** : après avoir sélectionné une offre, le client émet une requête d'affectation d'@ au serveur élu
  - **DHCPAck** : le serveur renvoie une config. Réseau et une durée de validité (*lease time*)
  - **DHCPNAck** : refus d'un renouvellement par le serveur
  - **DHCPRelease** : résiliation du bail avant échéance par le client

Quelques fichiers de config. UNIX

- `/etc/hosts` : association locale nom/@IP
- `/etc/resolv.conf` : @ des serveurs de noms, noms de domaines
- `/etc/protocols` : association nom de protocole, numéro de protocole, liste d'alias

```
icmp    1     ICMP
tcp     6     TCP
```

- `/etc/services` : association nom de service, numéro de port/protocole, liste d'alias

```
ftp     21/tcp     FTP
ssh     22/UDP
```

- `/etc/inetd.conf` : association entre nom de service et exécutable réalisant le service

Quelques commandes UNIX

- `ping` : teste l'accessibilité d'une destination
- `traceroute` : renvoie la route prise par les paquets pour atteindre une destination
- `arp` : visualiser/modifier le cache ARP
- `host` : interroger un serveur de noms
- `netstat` : obtenir des statistiques sur le nombre de paquets, les erreurs, les collisions, une interface, une table de routage, les sockets ouvertes, …
- `tcpdump` : visualiser des informations qui passent par l'interface réseau d'une machine