Pour en savoir plus sur la Compression (suite)

Algorithme de compression :

-20 octets (en grisé page 6) inchangés (identiques dans tous les paquets d'une connexion TCP) sont remplacés par 1 octet ("connection number").
- "total lenght" est donné par le LCP.
- "header checksum" inutile (plus de header).
- si envoi de la différence par rapport au paquet précédent (au lieu de la valeur), "packet ID", "sequence number", "ack number", "window" et "urgent pointer" tiennent dans 1 octet (chacun!).
- comme tous ces champs ne changent pas de valeur à chaque paquet, le premier octet est un masque qui indique les valeurs modifiées et donc, les champs optionnels qui suivent. Cet octet contiendra aussi les flags non grisés page 6.
- si la valeur est prévisible (par exemple +1 pour le "packet ID") elle n'est pas envoyée.
- seul le "checksum TCP" est envoyé inchangé (2 octets).

De plus on garde toujours le "connexion number" du dernier paquet envoyé donc pas besoin d'envoyer cette valeur si c'est la même.
Encore un octet de gagné...

ANNEXE

Pour en savoir plus sur la Compression

Principe :

1) Si un paquet
    - n'est pas TCP
    - est un fragment de datagramme TCP
    - a certain bits de contrôle positionnés

   il est envoyé inchangé avec le champ protocole "type_ip".

2) Si une connexion TCP s'ouvre, le 1er paquet est envoyé (presque) inchangé avec le champ protocole "uncompressed_tcp" après copie des en-têtes IP et TCP dans une table des connexions ouvertes.
   Le champ "protocol" de l'en-tête IP contient alors le "connection number" (index dans la table). Le destinataire copie alors les en-têtes dans sa table avec le même index et remet le bon "protocol".

3) Si un paquet d'une connexion ouverte doit être envoyé, l'en-tête est comparée avec celle de la table et le paquet part (protocole "compressed_ip") avec une en-tête compressée comprenant les champs absolu ment nécessaires et le "connection number" s'il est différent du dernier paquet envoyé.
   Le destinataire fait le travail inverse.

Des paquets "uncompressed_tcp" sont aussi envoyés sur une connexion ouverte pour synchronisation.

C'est

☐ déjà

☐ enfin

terminé(❊)

(❊) cocher la case correspondante

jt@ccr.jussieu.fr

Les RFCs

- Low-Cost IP Connectivity
- PPP White Paper

- rfc 1055 : slip
- rfc 1144 : Compressing TCP/IP Headers

- rfc 1547 : PPP Requirements
- rfc 1548 : PPP LCP
- rfc 1570 : PPP LCP Extensions
- rfc 1549 : PPP HDLC Framing
- rfc 1333 : PPP LCP LQM
- rfc 1334 : PPP Authentication
- rfc 1332 : PPP INCP (NCP IP)
- rfc 1376 : PPP DNCP (NCP DecNet)
- rfc 1377 : PPP OSINLCP (NCP OSI)
- rfc 1378 : PPP ATCP (NCP AT)
- rfc 1552 : PPP IPXCP (NCP IPX)

- rfc 1471->1474 : MIBs PPP

Conclusion suite

Exemples d'implémentations PPP :

- UNIX : si pas inclus dans la distribution
  les versions "domaine public" de
  PPP ne semblent pas faciles
  à installer.
  Morning Star PPP payant
- MAC : ncsa/telnet gratuit
  MacTCP payant
- PC : ncsa/telnet gratuit
  pc/tcp payant

Extensions prévisibles de la norme PPP :

- coupure ligne si inactivité
- re-numérotation si paquet à envoyer
- journalisation des evénements.

Disponible sur ftp.jussieu.fr (/pub/doc/jussieu/slip-ppp.ps)

Conclusion

1) On peut faire de l'IP (ou AT) à bas prix
   - Sur RTC, avec modems actuels :
     - v32/v42bis -> ~2 Koctets/sec
     - v32bis/v42bis -> ~3 Koctets/sec
   - Donc,
   - - telnet, ftp... (traffic "léger")
     - possible à 9600 bauds (2400?)
   - - X, mosaic... (traffic assez "lourd")
     - à 14400 bauds (28800)
     - (Low Bandwidth X améliorera X/RTC)
   - - NFS, station sans disque... (UDP non compressé) difficile (timeout!)

2) PPP s'impose sur RTC (sécurité!)

3) PPP pour toute connexion point à point

4) SLIP si on ne peut pas faire autrement (et en attendant)

5) Toute nouvelle installation en PPP même si on n'en utilise pas tout de suite toutes les fonctionnalités

Comparaison (C)SLIP - PPP

	slip	cslip	ppp
simplicité	oui	oui	
diffusion	partout	courant	fréquent
robustesse			oui
transparence	oui, si...	oui, si...	oui
taille trame	1006	1006	1500 nég
connexion	immédiat	immédiat	nég
overhead	1 à 2 oct	1 à 2 oct	4 à 8 oct
détect. erreurs			oui
détect. boucle			oui
LQM			oui
authentification			oui
adresse réseau	fixée	fixée	nég
réseaux	IP	IP	tous (?)
compression IP		VJ	VJ
extensibilité			oui

PPP

Avantages :

- diagnostic en cas d'échec à la connexion
- surveillance de la ligne (détection de boucles, LQM)
- détection d'erreurs (FCS)
- trame normalisée HDLC (1500 octets. Négociable)
- transparence totale
- compressions (LCP, VJ)
- authentification possible
- négociation d'adresses réseau
- "overhead" faible si toutes compressions
- multiprotocole (IP, AT...)
- MIBs

Inconvénients :

- complexe (portage, installation)
- phase de connexion plus "délicate" et plus longue (négociations)
- pas (encore) très répandu

Les options

1 : obsolète

2 : type de compression des en-têtes
    TCP/IP.
    La seule reconnue : Van Jacobson
    Idem CSLIP
    Défaut : pas de compression

3 : négociation de l'adresse IP.
    Si la zone données = 0.0.0.0
        => demande d'adresse
    sinon
        => adresse imposée.
    Si 2 demandes d'adresses ou
    2 adresses imposées
        => échec.
    Défaut : pas de négociation

Un exemple au hasard : IPCP

Les options de configuration

Le champ données des paquets IPCP "configure..." contient une suite d'options de format :

type	longueur	données

- type
  - 1 : IP addresses
  - 2 : IP compression protocol
  - 3 : IP address

- longueur totale de l'option (>= 2)

- données selon type

C'est donc tout ce qui se négocie en IPCP

Les paquets NCP (protocole 0x8021)

code	ident	longueur	données

- codes :
  - établissement et configuration de la liaison
    - 1 : configure request
    - 2 : configure ack
    - 3 : configure nack
    - 4 : configure reject

  - coupure de la liaison
    - 5 : terminate request
    - 6 : terminate ack

  - surveillance et gestion de la liaison
    - 7 : code reject

- ident : un numéro requête-réponse
- longueur des données éventuelles
- données : selon code

Similaire à LCP

Les NCPs

Un par protocole.

Sont aujourd'hui RFCisés :

IPCP : IP Control Protocol
DNCP : DecNet Control Protocol (ph. 4)
OSINLCP : OSI Network Layer Control Protocol
ATCP : AppleTalk Control Protocol
IPXCP : IPX Control Protocol

+ ceux à venir.

Authentification (suite)

CHAP :

- protocole 0xC223
- paquet :

code	identifier	total length
data		

codes

1 : challenge
2 : response
3 : success
4 : failure

- MD5 est le seul algorithme actuel

- Trois temps :
  1) envoi d'une chaîne
  2) réception du résultat de la fonction MD5 (secret, chaîne)
  3) comparaison => ack ou nack

- périodique

Authentification

PAP :

- protocole 0xC023
- paquet :

code	identifier	total length
ID length	identification	
PW length	password	

codes

1 : authenticate request
2 : authenticate ack
3 : authenticate nack

identification et mot de passe
de connexion PPP (et non un "login")

- Une fois en début de connexion

Link Quality Monitoring

- si demandé lors du LCP

- envoi d'un paquet avec protocole 0xC025
  tous les n centièmes de secondes.
  n = champ data du paquet LCP de type 4

- contenant
  1) des variables spécifiques
  2) des variables SNMP
     nombre paquets IN
     nombre paquets OUT
     nombre octets IN
     nombre octets OUT
     etc...

Les options

1 : 1500 octets par défaut

2 : liste des codes qui seront préfixés (transparence des caractères de contrôle en asynchrone). Défaut : <0x21

3 : Password Authentication Protocol ou Challenge Handshake Authentication Protocol. Défaut : pas de Protocole

4 : Link Quality Protocol. Défaut : rien

5 : Magic Number : détection de boucle. Défaut : 0x0000

7 : Compression du champ "protocol" Défaut : pas de compression

8 : Suppression des champs "address" et "control". Défaut : pas de compression

9 : nul, 16bits ou 32bits FCS. Défaut : 16 bits

10 : code de "bourrage". Défaut 0xFF.

13 : rappel automatique après authentification. Défaut : sans.

15 : plusieurs paquets PPP dans une tra me. Défaut = 1.

Les options de configuration LCP

Le champ données des paquets LCP "configure..." contient une suite d'options de format :

type	longueur	données

- type
  - 1 : maximum receive unit
  - 2 : async control character map
  - 3 : authentication protocol
  - 4 : quality protocol
  - 5 : magic number
  - 7 : protocol field compression
  - 8 : address and control field compression
  - 9 : FCS alternatives
  - 10 : self describing padding
  - 13 : callback
  - 15 : compound frames

- longueur totale de l'option (>= 2)

- données selon type

C'est donc tout ce qui se négocie en LCP

Les types de paquets LCP

1 : envoi des options non standards
(1er paquet envoyé)
2 : tout accepté
3 : toutes options connues mais valeurs
refusées
4 : options inconnues ou négociation
refusée
5, 6 : fermeture propre
7 : réception d'un code inconnu
8 : protocole inconnu ou non supporté
9, 10 : test ligne, debug, performances…
11 : paquet à mettre à la poubelle
12 : envoi d'un code d'identification LCP
13 : temps de connexion restant

Les paquets LCP (protocole 0xC021)

code	ident	longueur	données

- codes :

  établissement et configuration de la liaison
    1 : configure request
    2 : configure ack
    3 : configure nack
    4 : configure reject

  coupure de la liaison
    5 : terminate request
    6 : terminate ack

  surveillance et gestion de la liaison
    7 : code reject
    8 : protocol reject
    9 : echo request
    10 : echo reply
    11 : discard request

  extension récentes
    12 : identification
    13 : time remaining

- ident : un numéro requête-réponse
- longueur des données éventuelles
- données : selon code

Un paquet PPP

Encapsulation :

protocole	datagramme	bourrage

- protocole (16bits) : type de datagramme
par exemple :
    0x21: IP
    0x29 : AT
    0xC021 : LCP
    0x8021 : IPCP
- datagramme 0 à 1500 octets ; négociable
- bourrage éventuel

La couche LCP

Un automate complexe : 9 états,
16 événements, 13 actions, des "timers"
et des compteurs...

HDLC

La trame :

7E	FF	03	data	FCS	7E

- flag 7E (facultatif si trames se suivent)
- address (toujours FF = toutes stations)
- control (toujours 03 = Unnumbered Info)
  avec PPP "address" et "control" peuvent
  être supprimés après négociation
- données
- frame check sequence
- flag

Pour une liaison asynchrone, les données ne devant pas être (mal) interprétées sont codées par :
- un préfixe ESC (0x7D)
- inversion du bit 6

Ainsi, par exemple :

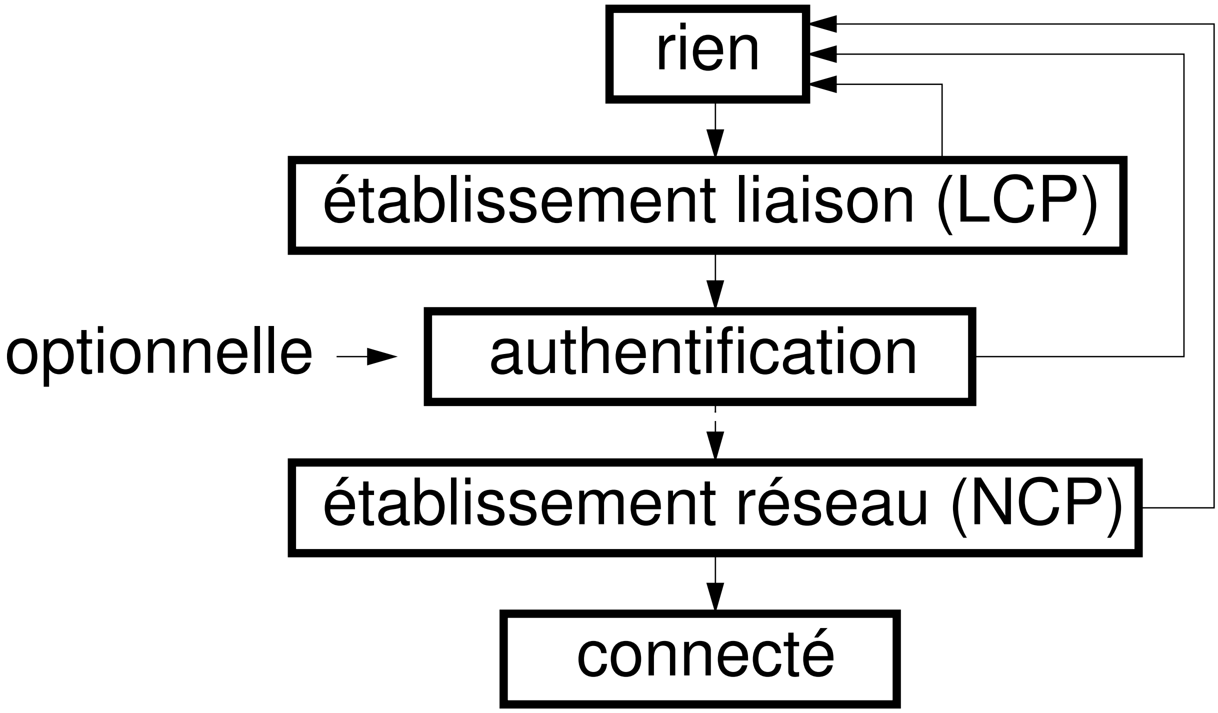
0x7E -> 0x7D 0x5E ; 0x7D -> 0x7D 0x5D
0x11 -> 0x7D 0x31 ; 0x13 -> 0x7D 0x33

PPP

Perkins 1990, revu en 92 et 93, maj en 94

1) une encapsulation standard : HDLC
2) un protocole de contrôle
   "liaison de données" : LCP
3) authentification optionelle
   PAP et/ou CHAP
4) des protocoles de contrôle
   "réseau" : NCP

Les phases d'une connexion PPP :



rien

établissement liaison (LCP)

optionnelle →

authentification

établissement réseau (NCP)

connecté

(C)SLIP

Avantages :

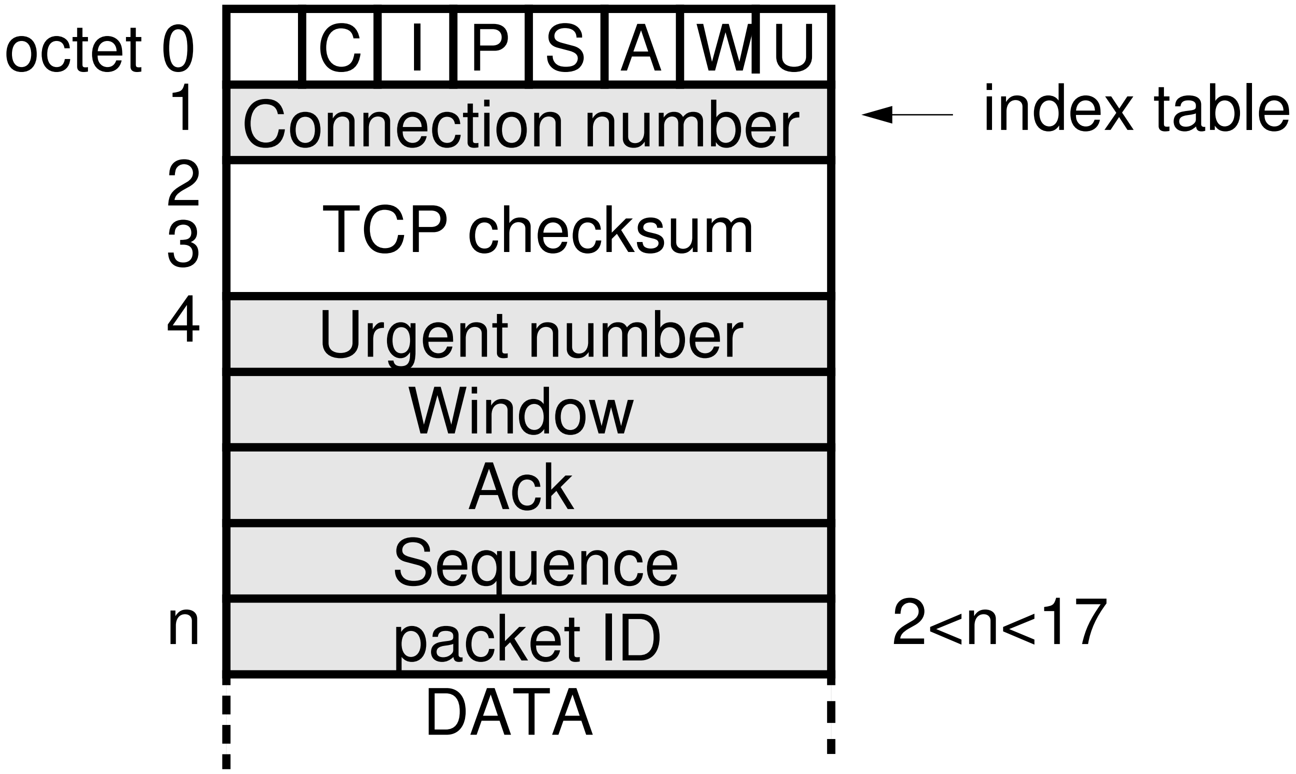
- très simple
- répandu
- transparent (sauf si contrôle de flux Xon-Xoff entre modems)
- "overhead" pratiquement nul

Inconvénients :

- pas de surveillance de la ligne
- aucune détection d'erreurs (doivent être détectées et corrigées par les couches supérieures => plus tard)
- taille trame fixe à 1006 octets
- aucune sécurité (authentification)
- pas de négociation d'adresses
- pas de négociation de la compression VJ
- pas d'extensions possibles
- IP uniquement

Compression (suite)

- Pour tous les datagrammes d'une connexion TCP, infos inchangées ou redondantes ou prévisibles ou compressibles : reste 3 à 16 octets (voire annexe).



octet 0 | | C | I | P | S | A | W | U

1 Connection number ← index table

2
3 TCP checksum

4 Urgent number

Window

Ack

Sequence

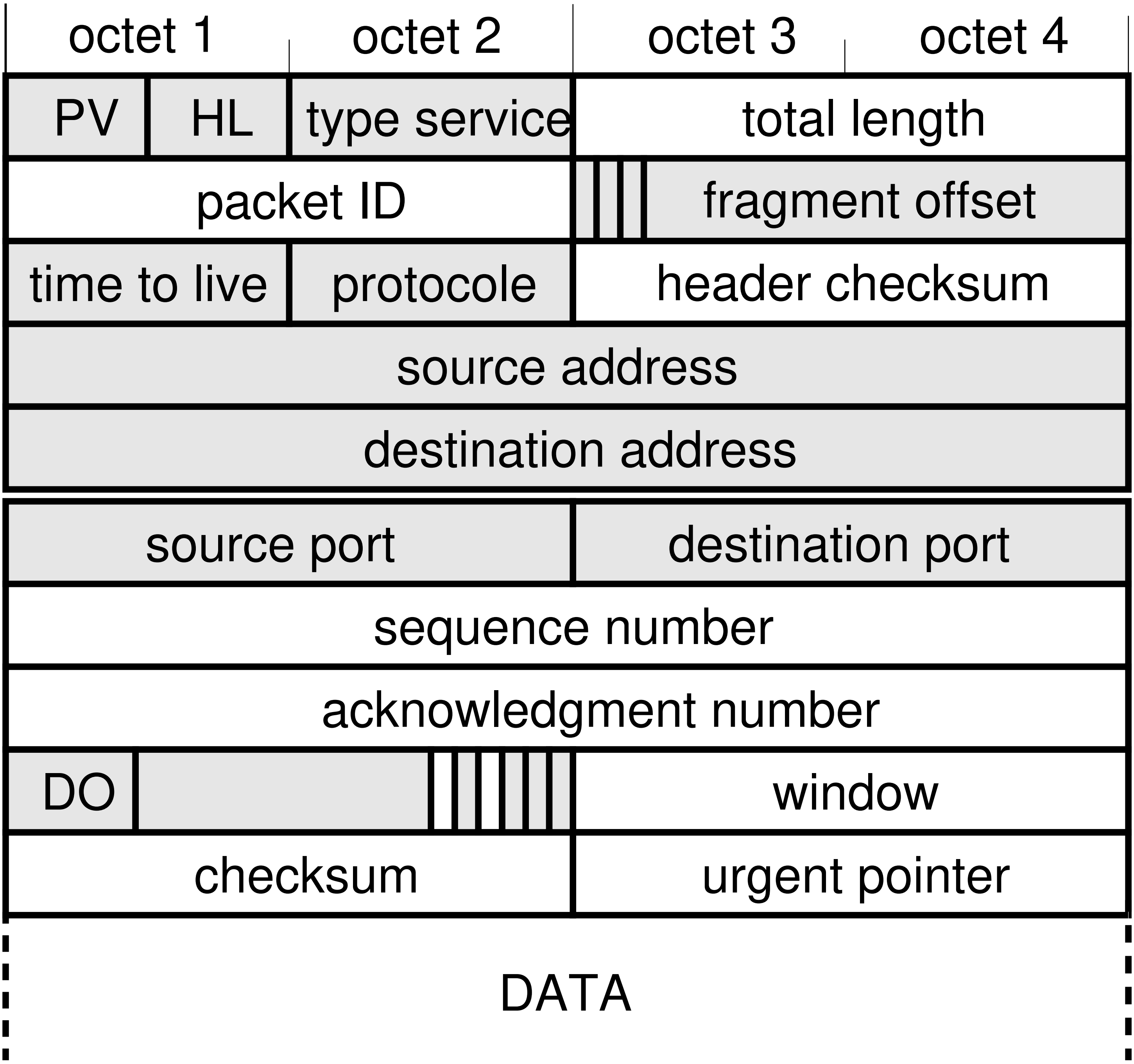
n packet ID 2<n<17

DATA

- 3 formats de datagrammes :
  1) type_ip => ne rien faire
  2) uncompressed_tcp => copier en-têtes dans la table des connexions
  3) compressed_tcp => remplacer (compléter) les en-têtes par (à partir) de l'index dans la table

La compression

- En-tête TCP + IP = 40 octets



octet 1 octet 2 octet 3 octet 4

PV HL type service total length

packet ID fragment offset

time to live protocole header checksum

source address

destination address

source port destination port

sequence number

acknowledgment number

DO window

checksum urgent pointer

DATA

jt@ccr.jussieu.fr

CSLIP

- trames idem SLIP

- compression des en-têtes
  (dite compression "Van Jacobson"
  ou "VJ")
  rfc 1144 [1990]

- pas de négociation SLIP/CSLIP
(la connexion est établie dans l'un
OU l'autre mode)

- parfois l'appelé détecte le type
de connexion et s'y adapte

SLIP

Écrit par Rick Adams [BSD 4.2, 1984]
Défini comme un "non-standard" par
le rfc 1055 [1988]
Inclus dans la plupart des Unix depuis

Protocole ultra-simple :
- 2 caractères spéciaux
  END (0xC0) et ESC (0xDB)
- si un des 2 apparaît dans la trame
  => il est remplacé par 2 octets :
  0xC0 -> 0xDB 0xDC
  0xDB -> 0xDB 0xDD

La Trame :

0xC0	Datagramme IP	0xC0

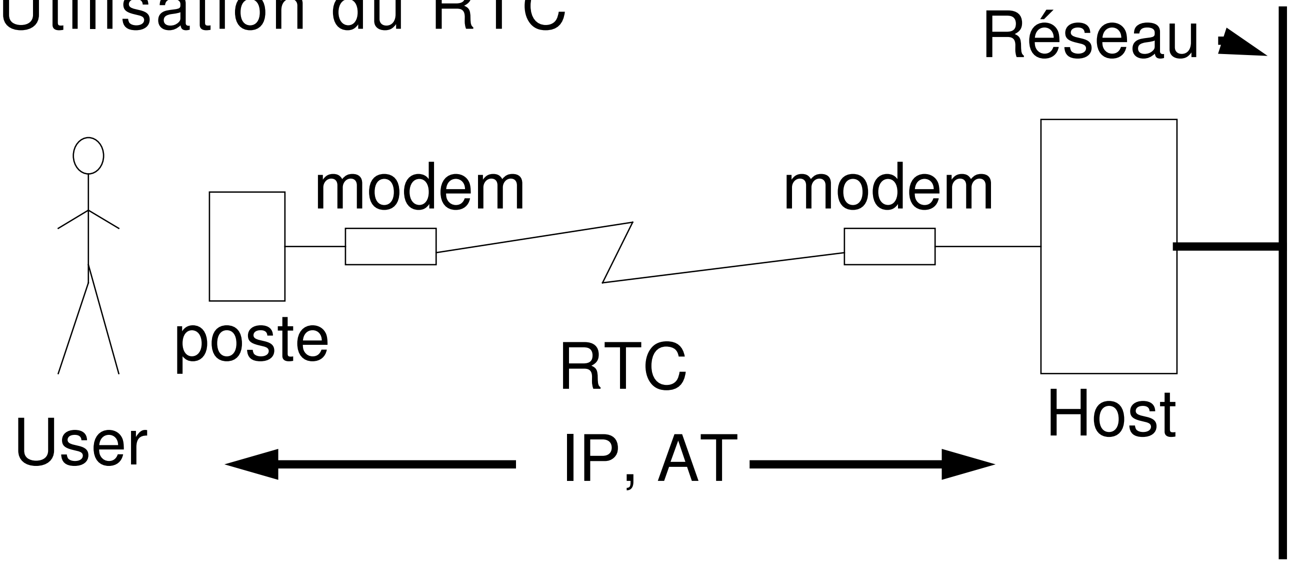
- premier octet facultatif
  (si trames se suivent)
- 0 à 1006 octets de données

Présentation

Pourquoi utiliser un protocole point à point
- kermit, zmodem, tip/cu... => telnet, ftp
- + smtp, pop, nntp, mosaic...

avec l'apparition de modems rapides et pas (trop) chers.

Utilisation du RTC



Réseau

modem

modem

poste

RTC

Host

User

IP, AT

Indépendance protocole niveau supérieur
Indépendance support

réseau	IP, AT...
liaison	SLIP, PPP...
physique	LS, RTC...

PLAN

1) Présentation

2) SLIP

3) CSLIP

4) PPP

5) Comparaison

6) Conclusion

Les protocoles de communication point à point

SLIP et PPP

Jacky Thibault
CCR -Jussieu