

PPP
Point to Point Protocol

Mise en Œuvre

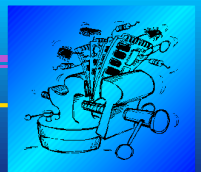
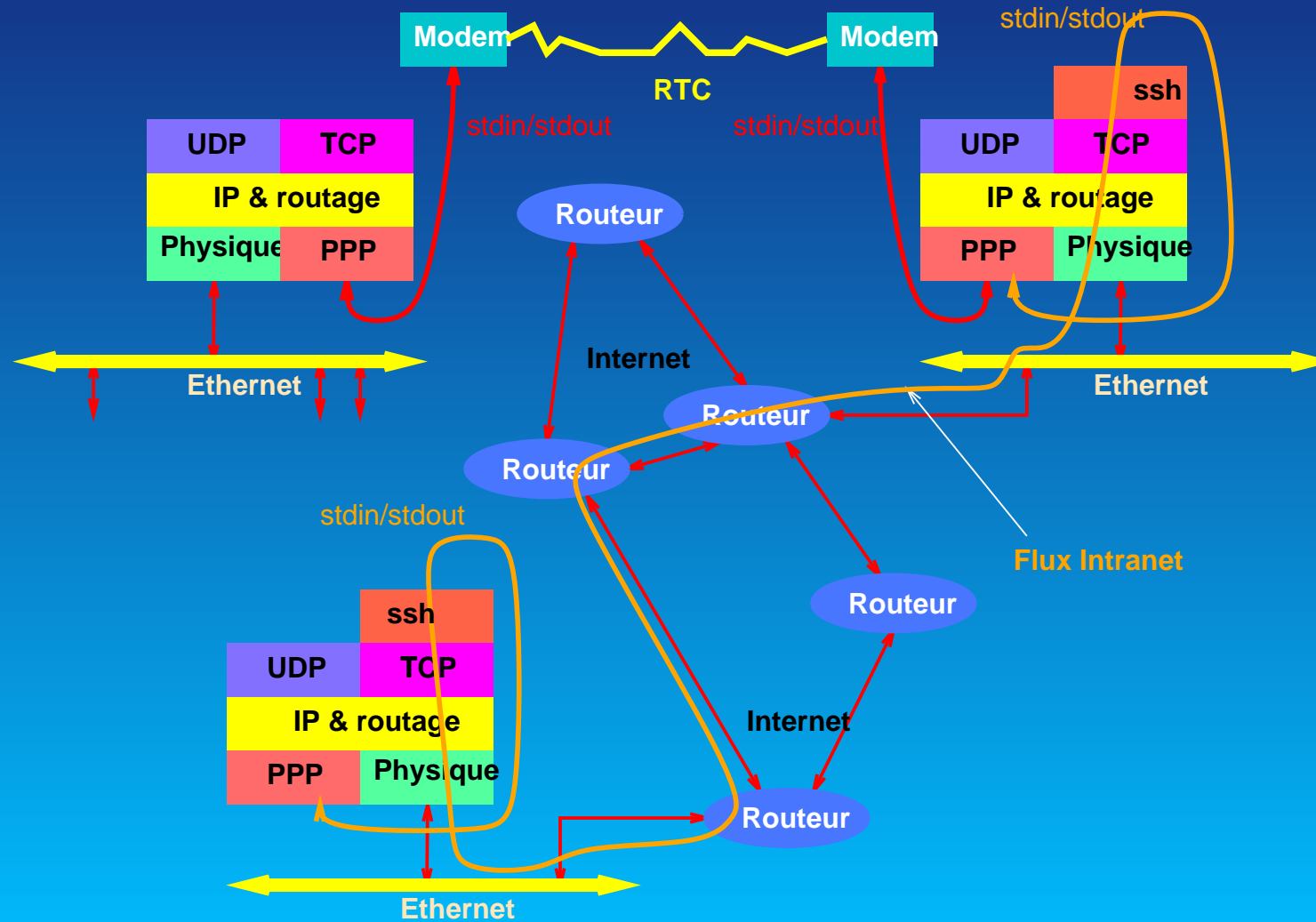
Ronan.Keryell@enst-bretagne.fr

Département Informatique
École Nationale Supérieure des Télécommunications de Bretagne

27–28 janvier 2000

- Besoin d'accès Internet pas cher
- ~→ Utilisation du réseau téléphonique commuté via des modems
- Faire passer le protocole IP (entre autre) dans des liaisons série (radio-amateurs...)
- Développement de protocoles d'encapsulation génériques dont PPP est une *lingua franca*
- Faire des réseaux privés virtuels (Intranet) au dessus d'Internet : le flux d'octet peut être transporté par des applications standard (rsh, ssh,...)
- Comprendre la notion d'encapsulation de protocoles







- Description du protocole PPP
- Modem
- Routage proxyARP et outils d'administration
- Mise en pratique : tunnel dans `ssh`



- Protocole simple développé chez 3Com début 1980 puis inclus dans 4.2BSD et SunOS en 1984
- RFC 1055
- Trames ultra-simples auto-synchronisantes :



-  Si un 0xc0 dans les données ?
 - ▶ Encodage préalable 0xc0 en 0xdb 0xdc
 - ▶  Si 0xdb dans les données ? Encodage en 0xdb 0xde
- Économie possible d'1 de 2 0xc0 consécutifs
- Aucun contrôle, pas d'échange d'adresse des extrémités pourtant utile pour le routage, etc.





- Ancien
- Ne peut faire passer que de l'IP dans la version disponible (pas d'identificateur de protocole)
- Pas d'authentification
- Protocole CSLIP avec compression mais incompatible avec SLIP : le site distant doit savoir si on utilise ou non la compression
- Pas de négociation des paramètres de communication
- Pas de checksum



- Développement sur VAX 4.3BSD (1989), RFC 1171 (1990) puis RFC 1661 (1994)
- Plusieurs types de trames sont possible. Exemple de trames de type HDLC avec leurs propre code de vérification (RFC 1662) :

Drapeau	Adresse	Contrôle	Protocole	Données	FCS	Drapeau
0x7e	0xff	0x03	0xabcd	<i>n</i> octets	16 ou 32 bits	0x7e

-  Si un 0x7e dans les données ?
 - ▶ Encodage préalable 0x7e en 0x7d 0x5e
 - ▶  Si 0x7d dans les données ? Encodage en 0x7d 0x5d, etc. ainsi que pour 0x03 mais aussi éventuellement XON, XOFF... si interceptés (modems)
- Si FCS incorrect : à la poubelle !
- Version compacte :



Protocole	Données	FCS	Drapeau
0xab	n octets	16 bits	0x7e

- Numéro de protocole \rightsquigarrow plusieurs protocoles simultanés (NetBios, AppleTalk, OSI, DECnet, IPX, etc) sans encapsulation
- LCP (Link Control Protocol) permettant aux extrémités de se mettre d'accord sur :
 - ▶ Taille maximale des trames
 - ▶ Échappement de certains caractères du style XON(C-s)/XOFF(C-q) avec différentes méthodes dans chaque direction
 - ▶ Authentification de bas niveau
 - ▶ Détection d'une boucle : un *magic number* tiré au hasard est envoyé. Si on reçoit son propre *magic number*, il y a



probablement un rebouclage

- ▶ Contrôle de qualité
- ▶ Arrêter
- ▶ Compression des paquets
- ▶ ...
- NCP (Network Control Protocol) avec IPCP (RFC-1332) dans le cas d'IP :
 - ▶ Échange et attribution d'adresses
 - ▶ Choix d'un système de compression des entêtes IP
 - ▶ Arrêter
- Possibilité d'agréger plusieurs liens pour augmenter les débits (multi-link)

L'intérêt de PPP est la versatilité, la portabilité et l'information



disponible sur le (non-)fonctionnement.

Nombreux développements (encapsulation dans ISDN, HDLC, X25, SONET,...) cf. les RFC



Utilisation de 2 protocoles :

PAP : (Password Authentication Protocol) un mot de passe est envoyé en clair sur la ligne et est comparé par l'autre à celui enregistré pour ce couple de machines

CHAP : (Challenge Handshake Authentication Protocol)

1. un nombre aléatoire est envoyé par A sur la ligne
2. la machine distante B la hache avec un secret via MD5 et le renvoie sur la ligne
3. A compare le résultat avec le même calcul effectué localement avec le même secret
4. MicroSoft utilise une version spécifique de CHAP avec une autre fonction de hachage

Pour des raisons de sécurités, seule la dernière version est à utiliser (le mot de passe ne passe pas en clair)




- 2 états : mode commande (langage) & mode transmission de données
- Choix de la vitesse, de la parité et du nombre de bits de stop souvent fait par imitation de la liaison avec l'ordinateur ~→ l'ordinateur doit causer au moins une fois au modem !
- Contrôle de flot logiciel (XON/XOFF) (monopolise 2 caractères) ou matériel (CTS/RTS & DTR/DSR) (nécessite 4 fils de plus)



Exemple du Sportster Message Plus.

Une commande est une séquence de caractères suivie par un retour chariot. En général le modem répond OK à une commande

- A : répond
- A/ répète la dernière commande
- +++ + pause sans retour chariot pour passer en mode commande.  si « +++ » apparaît justement dans un flot de données avec une pause... Possibilité de déconnecter le modem ? Dépend du protocole et possibilité de supprimer cette interprétation du + au niveau modem ou de le supprimer de la transmission (via PPP)
- AT suivi d'une commande plus longue + retour chariot. AT tout court permet de vérifier la connexion modem avec OK



- ▶ AT 00 retourne en mode ligne (après +++)
- ▶ ATDT0,0164694992 fait le numéro en fréquence vocale avec une pause « , »
- ▶ at h 0 raccroche
- ▶ at e1 met en marche l'écho des commandes
- ▶ at f0 met en marche l'écho des données transmises
- ▶ Concaténation des commandes possible
at e0f113 supprime tout écho et met le haut-parleur à fond
- ▶ at q0 affiche le résultat des commandes
- ▶ at v1 ... en mode alphanumérique
- ▶ at x4 &a3 ... et en mode très verbeux
- ▶ at sr? affiche le contenu du registre r
- ▶ at sr=n met n dans le registre r : at s0=0 réponse automatique au bout de 0 sonnerie \rightsquigarrow pas de réponse

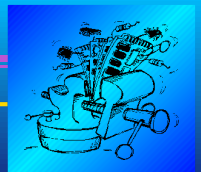


automatique

- ▶ `at s43.4=1 s43.5=1` active le stockage des télécopies et des messages vocaux (pas plus documenté !...)
- ▶ `at z4` réinitialise le modem à partir de la configuration d'usine pour un contrôle de flot matériel
- ▶ `at &b1` la vitesse du port série est fixe et indépendante de celle de la ligne téléphonique
- ▶ `at &c1` le signal DCD indique bien la présence de porteuse
- ▶ `at &d2` le signal DTR a un comportement normal (terminal prêt à recevoir)
- ▶ `at &f1` charge une configuration d'usine avec un contrôle de flot matériel
- ▶ `at &h1` contrôle de flot matériel suivant le signal CTS (modem prêt à envoyer)



- ▶ `at &r2` le signal RTS contrôle la transmission de données du modem au terminal
 - ▶ `at &s1` le modem sort un DSR normal autorisant l'émission par le terminal
 - ▶ `at &i0` pas de contrôle de flux logiciel en réception
 - ▶ `at &k1` utilise la compression de donnée si possible.
Souvent, la compression de `pppd` est meilleure car beaucoup de mémoire et meilleur algorithme de compression. Dépend du PPP distant... mais difficile de changer la configuration du modem une fois PPP démarré
 - ▶ `at &n0&u0` utilise la vitesse téléphonique maximale possible
 - ▶ `at &w0` écrit la configuration courante dans la configuration NVRAM 1
- Lire la doc... malheureusement pour grand public !...



Un argument numérique manquant est compris comme 0



Documentation papier incomplète ou indisponible ? Demandons au Sportster :

- `at $` affiche les options de numérotation
- `at &a` affiche la liste des commandes avec un `&`
- `at s$` liste les registres
- `at i4` donne les paramètres du modem
- `at i5` donne les paramètres stockés dans la NVRAM du modem
- `ati6` et `ati11` donne le diagnostic de la liaison
- `at i7` donne la configuration du produit



La législation française interdit qu'un système informatique appelle trop souvent par erreur un faux numéro

- Liste noire des échecs stockée dans le modem
- Interdit un numéro au bout de 4 échecs
- Affichage de la liste sur Sportster : `ati8`
- Il existe des séquences secrètes mettant à 0 cette liste...
- Problème lors de la mise au point (pas de réponse, erreur de config) \rightsquigarrow numéro innocent en liste noire
- Mon approche : rajouter un suffixe changeant à mon numéro (ignoré par France Télécom). Ex :
0164694992 0001, puis 0164694992 0002, etc. lorsque cela arrive



Grâce à la commande `tip` on peut tester et initialiser la vitesse et certains paramètres :

```
tip -38400 /dev/term/a
```

Pour en sortir retour chariot ~.

Problème : comment se connecter sur un modem en attente de connexion et bloquant le terminal avec DCD pour le configurer ou utiliser le modem temporairement en sortie ?

Utiliser la « porte de derrière » qui ignore DCD :

```
tip -38400 /dev/cua/a
```



Il y a principalement 2 possibilités pour connecter un modem à un serveur :

- Contrôle direct par PPP : le démon PPP tourne tout le temps et attend une connexion
 - ▶ C'est simple (utilisé au CRI)
- Contrôle par un `getty` avec une mire de login. Il faut donc d'abord se logger puis lancer PPP
 - ▶ Fournit un premier niveau d'authentification : le mot de passe du login. Souvent seul système d'authentification utilisé par les fournisseurs d'accès...
 - ▶ C'est plus compliqué mais peut aussi être automatisé
 - ▶ Certains `getty` reconnaissent les trames PPP et lancent PPP automatiquement (Linux)
 - ▶ Le modem peut servir à se connecter depuis un Minitel ou un



terminal connecté à un modem puisqu'il y a une mire de login



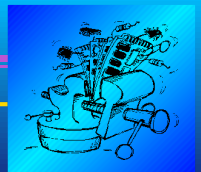
Sur Solaris, afficher les services de login avec `pmadm -l` et selon le besoin :

- Supprimer le login sur le port qui va accueillir PPP si besoin est avec

```
pmadm -r -p zsmon -s ttya
```

- Ou bien installer un login depuis le modem avec par exemple

```
pmadm -a -p zsmon -s ttya -i root -fu -v 1 \  
-m 'ttyadm -b -h -d /dev/term/a -l minitel -s /usr/bin/login \  
-S n -m ldterm,ttcompat -p 'moret minitel login: ''
```



- Rajouter dans `/etc/ttydefs` une entrée

```
minitel:38400 evenp hupcl:38400 evenp hupcl::38400
```

pour permettre de se logger depuis un Minitel. Pour avoir tous les caractères 8 bits depuis un autre ordinateur, envoyer un BREAK et éventuellement lancer PPP si besoin de PPP



- Mettre dans /etc/issue une séquence de login avec un passage en 80 colonnes pour Minitel (`^[:1}`) au début du style 1}

Bienvenue au Centre de Recherche en Informatique
de l'Ecole des Mines de Paris

Taper 2 retours chariots si vous venez d'un minitel...

PoUr_PASSer_eNtre_SePt_et_HUit_BitS_eNVoyer_UN_+BreAK+...

La dernière ligne est lisible aussi bien en 8 bits qu'en 7 bits
parité paire (Minitel)



Il existe de nombreux logiciels permettant de faire du PPP

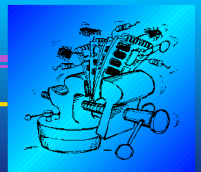
- Logiciels intégrés au système d'exploitation (Solaris2, Windows95-98,...)
- MacPPP pour MacIntosh
- pppd qui existe sur la plupart des Unix
- ... et d'autres : <http://cs.uni-bonn.de/ppp/faq.html>

C'est pppd qui est utilisé au CRI et qui est présenté ici sur Solaris avec un modem Sportster MessagePlus.

Il existe aussi du matériel permettant de relier directement des lignes ou des modems à un réseau Internet typiquement utilisé par les fournisseurs d'accès.



- Contient en fait plusieurs logiciels :
 - ▶ `pppd` le démon lui-même contrôlant la ligne
 - ▶ `chat` qui cause avec le modem ou les logins pour les initialisations
 - ▶ `pppstats` pour obtenir des statistiques sur `pppd`
 - ▶ de nombreux scripts pour démarrer ou arrêter les connexions, faire du rappel automatique, etc.
 - ▶ des modules rajoutés au noyau pour relier le flot IP au démon `pppd`
- Fonctionnalités intéressantes :
 - ▶ Fonctionne aussi sur liaisons synchrones
 - ▶ Notion de « plugins » chargeables pour modifier le comportement de `pppd`
 - ▶ Fonctions « *hook* » dans le code pour appeler du code



personnel

▶ IPv6

`ftp://cs.anu.edu.au/pub/software/ppp`



`pppd [tty_name] [speed] [option]`

`<local_IP_address>:<remote_IP_address>` impose une ou 2 des adresses spécifiées

`defaultroute` rajoute le pair comme route par défaut dans la mesure où c'est l'accès Internet principal

`proxyarp` fait croire à son réseau que la machine distante est locale

`netmask <n>` prend `<n>` comme netmask de l'interface

`mru <n>` (Maximum Receive Unit) demande au pair de ne pas envoyer de paquet dépassant la taille `<n>`. Compromis à trouver entre économie d'entête et monopolisation de la ligne le temps de transmettre un paquet. Typiquement 296 pour les liaisons lentes

`mtu <n>` idem mais en transmission



`pass-filter <expression>` choisit les paquets à transmettre, les autres partent à la poubelle

`ms-dns <addr>` fournit 1 ou 2 adresses de DNS à un client Microsoft

`ms-wins <addr>` fournit 1 ou 2 adresses de WINS (Windows Internet Name Services) à un client Microsoft

`bsdcomp <nr>,<nt>` comprime les paquets avec des codes jusqu'à <nr> bits en réception et <nt> bits en émission (compress)

`deflate <nr>,<nt>` un autre système de compression plus efficace (gzip)

`auth` demande que le pair s'authentifie

`call <name>` lit les options depuis `/etc/ppp/peers/<name>`

`connect <script>` exécute <script> pour se connecter qui contient typiquement un chat



`disconnect <script>` exécute `<script>` à la déconnexion

`modem` attend sur DCD et contrôle DTR en début et fin de connexion

`crtscts` utilise le contrôle de flot matériel, indispensable pour ne pas perdre trop de caractères

`local` ne contrôle pas de modem (ignore DCD et ne contrôle pas DTR). typiquement pour envoyer le flux réseau dans un autre tuyau (Intranet via `rsh` ou `ssh`,...)

`xonxoff` utilise le contrôle de flot logiciel XON/XOFF

`persist` survit à une fin de connexion. Typiquement pour un serveur

`passive` `pppd` essaye d'établir la connexion et attend si cela échoue. Typiquement pour les accès en entrée

`silent` encore plus discret car attend sans essayer d'établir une connexion



`demand` initialise le lien à la demande seulement

`idle <n>` déconnexion automatique au bout de `<n>` secondes d'inactivité

`active-filter <filter-expression>` choisit les paquets devant déclencher une connexion à la demande

`pty <commande>` alloue un *pty* pour le flux PPP et y connecte `commande`. Intérêt pour les tunnels :

```
pppd pty 'ssh -t server.my.net pppd'
```

`notty` n'utilise pas de *pty* pour le flux PPP

`debug` sort dans `syslogd` les paquets de contrôle

`record <filename>` enregistre tout le flux PPP. Traitement avec le programme `pppdump`



Un exemple simple sur deauville dans /etc/ppp/chap-secrets :

```
# Secrets for authentication using CHAP
# client          server  secret                      IP addresses
deauville ecuelles un vrai secret deauville
ecuelles deauville AH, encore un ! ecuelles
```

- Le symétrique pour le champs adresse doit se trouver sur ecuelles
- Permet d'accepter ou d'attribuer une adresse IP en fonction du nom de client donné. Cf la syntaxe dans le man
- Possibilité de mettre des options spécifiques par nom de connexion avec -- à la fin d'une ligne



Système de script d'interaction générale (cf aussi expect et dip)

```
chat [ options ] script
```

- f <chat file> où lire la configuration du modem. Par exemple
/etc/ppp/attente_MessagePlus
- t <timeout> temps avant d'abandonner. Utile pour les connexions lentes
- r <report file> où mettre ce qui est reçu, stderr par défaut envoyé par pppd dans /etc/ppp/connect-errors. Cf mot-clé REPORT
- e affiche ce qu'envoie chat. Cf ECHO
- v mode verbeux via syslog
- T <string> remplace \T dans le script par <string>. Utile pour paramétrer avec un numéro de téléphone, etc.



-U <string> remplace \U dans le script par <string>

Cf le man chat



- Script = couples attendu/envoi

```
ogin:-BREAK-ogin: ppp sword: hello2u2
```

```
ogin:--ogin: ppp sword: hello2u2
```

```
ABORT BUSY ABORT 'NO CARRIER' '' ATZ OK ATDT5551212 CONNECT
```

Les chaînes envoyées sont suivies d'un retour chariot

- Attention a ne pas mettre de mot de passe dans un script en ligne... ps auxww ou ps -ef
- Le code de retour d'erreur permet d'identifier l'ABORT qui a eu lieu entre autres



SAY Initialisation du modem USRobotics Sporster MessagePlus :

Réinitialise le modem à partir de la configuration

d'usine pour un contrôle de flot matériel :

at z4

Écho local des commandes mais pas des données :

OK at e1 f1

Numérote en fréquence vocale :

OK at t

Mode bien verbeux :

OK at q0 v1 x4 &a3

Vitesse DTE constante :

OK at &b1

Vitesse maximale :

OK at &n0 &u0

DCD normal :

OK at &c1



```
# DTR normal :  
OK at &d2  
  
# Le modem contrôle DSR :  
OK at &s1  
  
# Contrôle de flot CTS normal par le modem :  
OK at &h1  
  
# Contrôle de flot RTS :  
OK at &r2  
  
# Pas de contrôle de flot logiciel :  
OK at &i0  
  
# Met la compression si demandée :  
OK at &k1  
  
SAY Initialisation terminée.
```



Low BandWidth X proxy

```
lbxproxy [:<display>] [option]
```

- On crée localement un faux terminal X
- Son trafic est récupéré et retransmis au vrai serveur X en LBX par lbxproxy qui est moins gourmand
- Le client n'a donc pas besoin d'être modifié

```
xterm -display ecuelles:0
```

devient

```
lbxproxy :1 -display ecuelles:0
```

```
xterm -display :1
```

ssh Fait aussi de la compression



Après décompaction du .tar.gz récupéré depuis
<http://www.cri.ensmp.fr/~keryell/systeme/PPP> par exemple :

- `./configure`
- `make`
- passer root et faire `make install`



- Ajouter dans `/etc/syslog.conf` (bien noter le TAB en milieu de ligne...):

```
# Mets du debug bien verbeux, utile pour PPP :  
*.emerg;*.alert;*.crit;*.err;*.warning;*.notice;*.info;*.debug /var/adm/messages
```

- Faire un `kill -1 'cat /etc/syslog.pid'`
- Récupérer le `/etc/ppp` depuis la page
<http://www.cri.enscm.fr/~keryell/systeme/PPP>
- Adapter les fichiers de configuration, particulièrement `/etc/ppp/chap-secrets`
- Récupérer et installer le fichier `/etc/init.d/pppd`
- Faire un lien avec `ln -s ../init.d/pppd /etc/rc3.d/S90pppd`
- Lancer le tout avec un `/etc/init.d/pppd start` pour les essais..



Il y a aussi des exemples dans la distribution de `pppd` dans les répertoires `etc.ppp/` et `scripts/` ainsi que le fichier `SETUP`.

- La liste des fichiers utilisés au CRI :

`attente_avec_chat` un chat script initialisant le modem en attente utilisé par `ppp-on-dialer`

`auth-down` exécuté lors de la fin de la connexion authentifiée

`auth-up` exécuté lors du démarrage de l'authentification

`chap-secrets` contient les secrets CHAP. Seul root doit pouvoir lire ce fichier !

`ip-down` exécuté lors de la fin d'IP

`ip-up` exécuté lors du démarrage d'IP

`options` contient les options communes à tous les `pppd`

`pap-secrets` contient les secrets PAP (attention, ce n'est pas un protocole très sécurisé, à éviter). Seul root doit pouvoir lire ce



fichier !

ppp-off arrête pppd

ppp-on démarre pppd

ppp-on-dialer lance chat

- Les commandes en -up et -down écrivent une entrée dans /var/log/ppp-log.
- pppd initialise des variables d'environnement pour ces scripts : PPPLOGNAME, CONNECT_TIME, BYTES_RCVD, PEERNAME, SPEED,...



Utile pour le debug ou sur une ligne avec une mire de login.

Par exemple :

- Via un `tip` sur le modem local se connecter sur l'ordinateur d'accès distant (`ATDnuméro`)
- Arriver sur la machine devant faire tourner le `pppd` distant
- Lancer à distance `pppd` en mode passif depuis le shell
- Lancer le `pppd` local depuis `tip` grâce à la commande retour chariot `~C`
- Si pas via `tip`, il devrait être possible de lancer la connexion via un `cua` puis lancer `pppd` sur le `tty` dual et se déconnecter de `cua` avec comme effet de basculer la transmission sur `pppd` et son `tty`



```
deauville-keryell > netstat -r
```

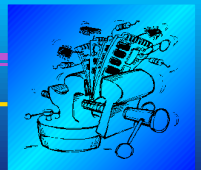
Routing Table:

Destination	Gateway	Flags	Ref	Use	Interface
-----	-----	-----	-----	-----	-----
localhost	localhost	UH	02930354		lo0
ville-saint-jacques	ecuelles	UGH	0	0	
ecuelles	deauville	UH	3	7	ppp0
ensmp-fbleau2	deauville	U	3	49140	le0
BASE-ADDRESS.MCAST.NET	deauville	U	3	0	le0
default	routeur-172	UG	0	4874	

- Une route a été rajoutée par pppd pour atteindre ecuelles en passant par le lien PPP ppp0
- Une route a été rajoutée par /etc/ppp/ip-up pour atteindre ville-saint-jacques derrière ecuelles connecté à un réseau Ethernet par :

```
/usr/sbin/route add ville-saint-jacques ecuelles 2
```

- Au niveau d'ecuelles pppd a rajouté la route vers deauville et

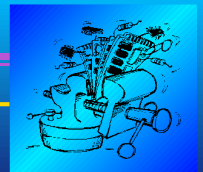


/etc/ppp/ip-up a rajouté la route vers ville-saint-jacques :

```
ecuelles-keryell > netstat -r
```

Routing tables

Destination	Gateway	Flags	Refcnt	Use	Interface
ville-saint-jacques	ecuelles	UH	0	0	le0
localhost	localhost	UH	0	0	lo0
deauville	ecuelles	UH	10	51912	ppp0
ecuelles	ecuelles	UH	19	54969	le0
default	deauville	UG	2	32191	ppp0



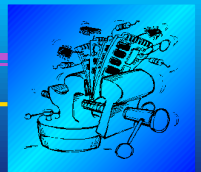
Comment faire pour qu'une autre machine puisse joindre ecuelles et villes-saint-jacques sans modifier la table de routage de toutes les machines du site ?

- En mettant au niveau du protocole ARP de traduction numéros IP → Ethernet
- deauville répond à tout le monde qu'ecuelles a le numéro Ethernet de deauville et récupère ses paquets. La table de routage n'a plus qu'à œuvrer

```
deauville-keryell > arp -a
```

```
Net to Media Table
```

Device	IP Address	Mask	Flags	Phys Addr
-----	-----	-----	-----	-----
ppp0	ecuelles	255.255.255.255		
le0	chailly	255.255.255.255		08:00:20:10:d5:c6
le0	chailly-qe1	255.255.255.255		08:00:20:10:d5:c6
le0	ville-saint-jacques	255.255.255.255	SP	08:00:20:85:f0:79
le0	ecuelles	255.255.255.255	SP	08:00:20:85:f0:79



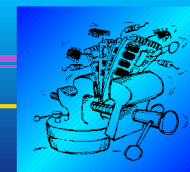
```
le0    deauville                255.255.255.255 SP    08:00:20:85:f0:79
le0    BASE-ADDRESS.MCAST.NET  240.0.0.0          SM    01:00:5e:00:00:00
```

- La déclaration de ville-saint-jacques est rajoutée dans /etc/ppp/ip-up par

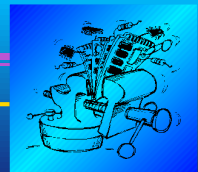
/usr/sbin/arp -s ville-saint-jacques 8:0:20:85:f0:79 pub
- Attention à la durée de vie des caches ARP lors de la mise au point : Sun=20 mn, Cisco=4 h...



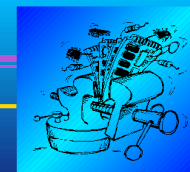

```
ecuelles-keryell > Dec 4 20:29:02 ecuelles pppd[271]: pppd 2.2.0 started by keryell, uid 151
Dec 4 20:29:08 ecuelles chat[273]: send (at&f^M)
Dec 4 20:29:08 ecuelles chat[273]: expect (OK)
Dec 4 20:29:08 ecuelles chat[273]: at&f^M^M
[...]
Dec 4 20:30:03 ecuelles chat[273]: CONNECT -- got it
Dec 4 20:30:03 ecuelles chat[273]: send (^M)
Dec 4 20:30:03 ecuelles pppd[271]: Serial connection established.
Dec 4 20:30:04 ecuelles pppd[271]: Using interface ppp0
Dec 4 20:30:04 ecuelles pppd[271]: Connect: ppp0 <--> /dev/ttyb
Dec 4 20:30:04 ecuelles pppd[271]: sent [LCP ConfReq id=0x1 <mru 296> <auth chap 05> <magic 0xe8b9f20f> <pcomp> <accomp>]
Dec 4 20:30:04 ecuelles pppd[271]: rcvd [LCP ConfReq id=0x2 <mru 296> <auth chap 05> <magic 0x140ecca9> <pcomp> <accomp>]
Dec 4 20:30:05 ecuelles pppd[271]: sent [LCP ConfAck id=0x2 <mru 296> <auth chap 05> <magic 0x140ecca9> <pcomp> <accomp>]
Dec 4 20:30:05 ecuelles pppd[271]: rcvd [LCP ConfAck id=0x1 <mru 296> <auth chap 05> <magic 0xe8b9f20f> <pcomp> <accomp>]
Dec 4 20:30:05 ecuelles pppd[271]: sent [CHAP Challenge id=0x1 <1da2e8395e4bb68f42ee37f30a18d419a3db75c604e73b095a889ec14b92b9cac48bb>]
Dec 4 20:30:05 ecuelles pppd[271]: rcvd [CHAP Challenge id=0x2 <a55a220f2b4d9c858eec2e4f87a23af3810367b5de2994f765c11bf45cbb69b9e2f0c>]
Dec 4 20:30:05 ecuelles pppd[271]: sent [CHAP Response id=0x2 <6b71ca24006172815d7c79668510569f>, name = ecuelles]
Dec 4 20:30:05 ecuelles pppd[271]: rcvd [CHAP Response id=0x1 <e37984dee266abd795fe19dc8f41e463>, name = deauville]
Dec 4 20:30:05 ecuelles pppd[271]: sent [CHAP Success id=0x1 Welcome to ecuelles.]
Dec 4 20:30:05 ecuelles pppd[271]: rcvd [CHAP Success id=0x2 Welcome to deauville.]
Dec 4 20:30:05 ecuelles pppd[271]: Remote message: Welcome to deauville.
Dec 4 20:30:05 ecuelles pppd[271]: sent [IPCP ConfReq id=0x1 <addr 192.54.172.207> <compress VJ 0f 01>]
Dec 4 20:30:05 ecuelles pppd[271]: sent [CCP ConfReq id=0x1 <bsd v1 12>]
Dec 4 20:30:05 ecuelles pppd[271]: rcvd [IPCP ConfReq id=0x2 <addr 192.54.172.242> <compress VJ 0f 01>]
Dec 4 20:30:05 ecuelles pppd[271]: sent [IPCP ConfAck id=0x2 <addr 192.54.172.242> <compress VJ 0f 01>]
Dec 4 20:30:05 ecuelles pppd[271]: rcvd [CCP ConfReq id=0x3 < 18 04 78 00> <bsd v1 15>]
Dec 4 20:30:05 ecuelles pppd[271]: sent [CCP ConfRej id=0x3 < 18 04 78 00>]
Dec 4 20:30:05 ecuelles pppd[271]: rcvd [IPCP ConfAck id=0x1 <addr 192.54.172.207> <compress VJ 0f 01>]
Dec 4 20:30:06 ecuelles pppd[271]: local IP address 192.54.172.207
Dec 4 20:30:06 ecuelles pppd[271]: remote IP address 192.54.172.242
Dec 4 20:30:06 ecuelles pppd[271]: Setting interface mask to 255.255.255.0
```



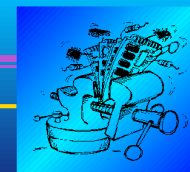
```
Dec 4 20:30:06 ecuelles pppd[271]: found interface le0 for proxy arp
Dec 4 20:30:06 ecuelles pppd[271]: rcvd [CCP ConfAck id=0x1 <bsd v1 12>]
Dec 4 20:30:06 ecuelles pppd[271]: rcvd [CCP ConfReq id=0x4 <bsd v1 15>]
Dec 4 20:30:06 ecuelles pppd[271]: sent [CCP ConfAck id=0x4 <bsd v1 15>]
Dec 4 20:30:06 ecuelles pppd[271]: Compression enabled
Dec 4 20:45:05 ecuelles pppd[271]: rcvd [CHAP Challenge id=0x3 <a03ec612cf4d89b75080d9f63e85e7d04149aac6a6cf07b69f47a4464fa309294c705
Dec 4 20:45:05 ecuelles pppd[271]: sent [CHAP Response id=0x3 <945178be9608fd06ca0d62f707842b39>, name = ecuelles]
Dec 4 20:45:06 ecuelles pppd[271]: rcvd [CHAP Success id=0x3 Welcome to deauville.]
Dec 4 20:45:06 ecuelles pppd[271]: Remote message: Welcome to
deauville.
[...]
```



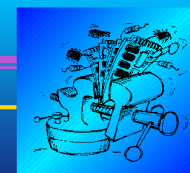
```
Dec  4 14:39:28 deauville pppd[3921]: rcvd [LCP TermReq id=0x2]
Dec  4 14:39:28 deauville pppd[3921]: LCP terminated by peer
Dec  4 14:39:28 deauville pppd[3921]: sent [LCP TermAck id=0x2]
Dec  4 14:39:31 deauville pppd[3921]: Connection terminated.
Dec  4 14:39:31 deauville pppd[3921]: Connection terminated.
Dec  4 14:39:37 deauville chat[4425]: send (at&f^M)
Dec  4 14:39:37 deauville chat[4425]: expect (OK)
Dec  4 14:39:37 deauville chat[4425]: ^M
Dec  4 14:39:37 deauville chat[4425]: OK
Dec  4 14:39:37 deauville chat[4425]: -- got it
[...]
Dec  4 14:39:43 deauville chat[4425]: ate0^M^M
Dec  4 14:39:43 deauville chat[4425]: OK
Dec  4 14:39:43 deauville chat[4425]: -- got it
Dec  4 14:39:43 deauville chat[4425]: send (^M)
Dec  4 14:39:43 deauville pppd[3921]: Serial connection established.
Dec  4 14:39:44 deauville pppd[3921]: Using interface ppp0
Dec  4 14:39:44 deauville pppd[3921]: Connect: ppp0 <--> /dev/ttyb
Dec  4 14:39:44 deauville pppd[3921]: Connect: ppp0 <--> /dev/ttyb
[...]
Dec  4 20:29:25 deauville pppd[3921]: rcvd [LCP ConfReq id=0x1 <mru 296> <auth c
hap 05> <magic 0xe8b9f20f> <pcomp> <accomp>]
Dec  4 20:29:25 deauville pppd[3921]: sent [LCP ConfReq id=0x2 <mru 296> <auth c
```



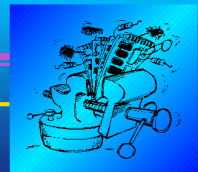
```
hap 05> <magic 0x140ecca9> <pcomp> <accomp>]
Dec  4 20:29:25 deauville pppd[3921]: sent [LCP ConfAck id=0x1 <mru 296> <auth c
hap 05> <magic 0xe8b9f20f> <pcomp> <accomp>]
Dec  4 20:29:25 deauville pppd[3921]: rcvd [LCP ConfAck id=0x2 <mru 296> <auth c
hap 05> <magic 0x140ecca9> <pcomp> <accomp>]
Dec  4 20:29:25 deauville pppd[3921]: sent [CHAP Challenge id=0x2 <a55a220f2b4d9
c858eec2e4f87a23af3810367b5de2994f765c11bf45cbb69b9e2f0c0f0fd6458018d1b49fdbbc2d5
ae2>, name = deauville]
Dec  4 20:29:25 deauville pppd[3921]: rcvd [CHAP Challenge id=0x1 <1da2e8395e4bb
68f42ee37f30a18d419a3db75c604e73b095a889ec14b92b9cac48bb4055f>, name = ecuelles
]
Dec  4 20:29:25 deauville pppd[3921]: sent [CHAP Response id=0x1 <e37984dee266ab
d795fe19dc8f41e463>, name = deauville]
Dec  4 20:29:26 deauville pppd[3921]: rcvd [CHAP Response id=0x2 <6b71ca24006172
815d7c79668510569f>, name = ecuelles]
Dec  4 20:29:26 deauville pppd[3921]: sent [CHAP Success id=0x2 Welcome to deau
ville.]
Dec  4 20:29:26 deauville pppd[3921]: CHAP peer authentication succeeded for ecu
elles
Dec  4 20:29:26 deauville pppd[3921]: CHAP peer authentication succeeded for ecu
elles
Dec  4 20:29:26 deauville pppd[3921]: rcvd [CHAP Success id=0x1 Welcome to ecue
lles.]
```



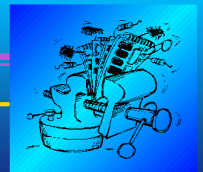
```
Dec  4 20:29:26 deauville pppd[3921]: Remote message: Welcome to ecuelles.
Dec  4 20:29:26 deauville pppd[3921]: sent [IPCP ConfReq id=0x2 <addr 192.54.172
.242> <compress VJ 0f 01>]
Dec  4 20:29:26 deauville pppd[3921]: sent [CCP ConfReq id=0x3 <deflate 15> <bsd
v1 15>]
Dec  4 20:29:26 deauville pppd[3921]: rcvd [IPCP ConfReq id=0x1 <addr 192.54.172
.207> <compress VJ 0f 01>]
Dec  4 20:29:26 deauville pppd[3921]: sent [IPCP ConfAck id=0x1 <addr 192.54.172
.207> <compress VJ 0f 01>]
Dec  4 20:29:26 deauville pppd[3921]: rcvd [CCP ConfReq id=0x1 <bsd v1 12>]
Dec  4 20:29:26 deauville pppd[3921]: sent [CCP ConfAck id=0x1 <bsd v1 12>]
Dec  4 20:29:26 deauville pppd[3921]: rcvd [IPCP ConfAck id=0x2 <addr 192.54.172
.242> <compress VJ 0f 01>]
Dec  4 20:29:26 deauville pppd[3921]: found interface le0 for proxy ARP
Dec  4 20:29:26 deauville pppd[3921]: local  IP address 192.54.172.242
Dec  4 20:29:26 deauville pppd[3921]: local  IP address 192.54.172.242
Dec  4 20:29:26 deauville pppd[3921]: remote IP address 192.54.172.207
Dec  4 20:29:26 deauville pppd[3921]: remote IP address 192.54.172.207
Dec  4 20:29:26 deauville pppd[3921]: rcvd [CCP ConfRej id=0x3 <deflate 15>]
Dec  4 20:29:26 deauville pppd[3921]: sent [CCP ConfReq id=0x4 <bsd v1 15>]
Dec  4 20:29:27 deauville pppd[3921]: rcvd [CCP ConfAck id=0x4 <bsd v1 15>]
Dec  4 20:29:27 deauville pppd[3921]: BSD-Compress (15/12) compression enabled
Dec  4 20:29:27 deauville pppd[3921]: BSD-Compress (15/12) compression enabled
```



```
Dec  4 20:44:26 deauville pppd[3921]: sent [CHAP Challenge id=0x3 <a03ec612cf4d8
9b75080d9f63e85e7d04149aac6a6cf07b69f47a4464fa309294c7057a4567d3e563d37489753>,
name = deauville]
Dec  4 20:44:26 deauville pppd[3921]: rcvd [CHAP Response id=0x3 <945178be9608fd
06ca0d62f707842b39>, name = ecuelles]
Dec  4 20:44:26 deauville pppd[3921]: sent [CHAP Success id=0x3 Welcome to deau
ville.]
Dec  4 20:44:26 deauville pppd[3921]: CHAP peer authentication succeeded for ecu
elles
Dec  4 20:44:26 deauville pppd[3921]: CHAP peer authentication succeeded for ecu
elles
[...]
```



```
deauville-keryell > ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
    inet 127.0.0.1 netmask ff000000
le0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
    inet 192.54.172.242 netmask fffffff0 broadcast 192.54.172.255
ppp0: flags=28d1<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST,UNNUMBERED> mtu 296
    inet 192.54.172.242 --> 192.54.172.207 netmask fffffff0
```



- `pppstats` donne des informations sur le fonctionnement de `pppd` :

```
deauville-keryell > pppstats -w 2 -r
```

IN	PACK	VJCOMP	RATIO	UBYTE		OUT	PACK	VJCOMP	RATIO	UBYTE
3341987	99884	93862	1.91	4374113		7630353	110408	95638	2.04	11561351
923	23	21	1.80	1158		644	22	21	2.12	932
612	17	15	1.93	853		646	20	17	1.80	788
513	19	17	1.76	569		2949	26	23	1.33	2917

- `ping` pour tester le fonctionnement et la latence :

```
deauville-keryell > ping -sv ecuelles
```

```
PING ecuelles: 56 data bytes
```

```
64 bytes from ecuelles (192.54.172.207): icmp_seq=0. time=235. ms
```

```
64 bytes from ecuelles (192.54.172.207): icmp_seq=1. time=212. ms
```

```
64 bytes from ecuelles (192.54.172.207): icmp_seq=2. time=180. ms
```

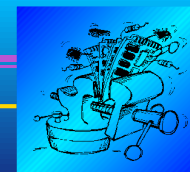
```
^C
```

```
----ecuelles PING Statistics----
```

```
3 packets transmitted, 3 packets received, 0% packet loss
```

```
round-trip (ms)  min/avg/max = 180/209/235
```

- `tcpdump` et `etherfind/snoop` pour voir passer les paquets (mais



pas sur le lien ppp0 lui-même...)

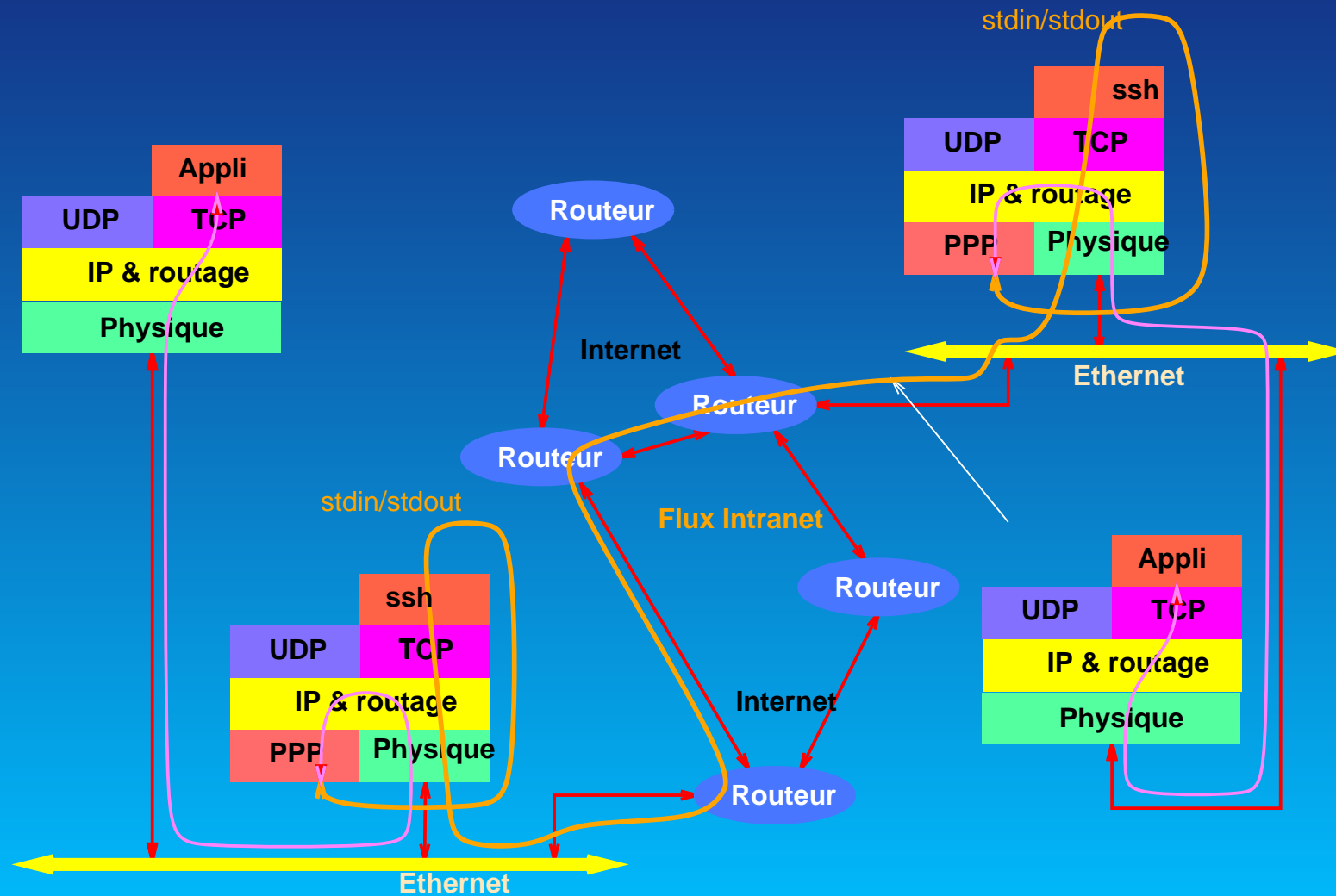
- traceroute pour voir par où (ne) passent (pas) les paquets


```
chailly-keryell > traceroute ecuelles
traceroute to ecuelles (192.54.172.207), 30 hops max, 40 byte packets
 1  deauville (192.54.172.242)  1 ms *  1 ms
 2  ecuelles (192.54.172.207) 235 ms 284 ms 278 ms
```



- PPP : passer protocole réseau sur flux d'octets
- Dépasser la simple liaison modem
- Idée : faire passer le flux dans une liaison quelconque





- rsh
- telnet
-  ssh
- Utilisations par des pirates
 - ▶ HTTP
 - ▶ FTP
 - ▶ SMTP
 - ▶ DNS
 - ▶ Fragments TCP secondaires si non filtrés par coupe-feux
 - ▶ ...



- Faire tourner un exécutable à distance en détournant *stdin* et *stdout* de manière cryptée depuis la machine *A* :

```
ssh B exécutable
```

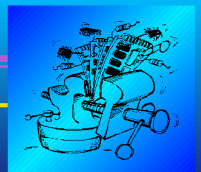
- Relier 2 `pppd` via `ssh/sshd`

```
pppd pty 'ssh -t B pppd'
```

- Mettre en place les clés d'authentification entre les 2 `ssh`
- Voir le texte du TP...



- PPP permet de faire beaucoup de choses
 - Les modems sont de vrais ordinateurs avec leur langages
 - C'est compliqué car tout est possible
 - Les Mac et PC arrivent avec une configuration standard simple à utiliser
 - Possibilité de créer des tunnels : pare-feu, intranets,...
- ~> Travaux pratiques



List of Slides

- 1 Introduction
- 2 Principe
- 3 Plan
- 4 SLIP
- 6 PPP
- 10 Authentification dans PPP
- 11 Modem modernes asynchrones
- 12 Commandes compatibles Hayes
- 17 Documentation en ligne du modem
- 18 Liste noire
- 19 Comment parler au modem ?
- 20 Connexion du modem
- 22 Contrôle du login
- 23 Accès compatible Minitel



25	Logiciels disponibles
26	Logiciel pppd
28	Quelques options de pppd
32	Fichiers d'authentification
33	chat
35	Fonctionnement de chat
36	Exemple de fichier chat
38	lbp proxy — Compression externe
39	Compilation
40	Mise en place
41	Contenu de /etc/ppp
43	Utilisation via une connexion
44	Routage
46	ProxyARP
48	Exemple de connexion
50	Exemple de connexion (dual)
54	Contrôle des paramètres
55	Autres outils d'information
57	Tunnel PPP



59	Transport du tunnel PPP
60	Tunnel PPP dans <code>ssh</code>
61	Conclusion
62	Table des matières

