Le transport sur liens satellites

Des PEPs aux DTNs

Lecture given by Emmanuel Lochin

ISAE-SUPAERO

Plan

Problématique de TCP sur liens satellites

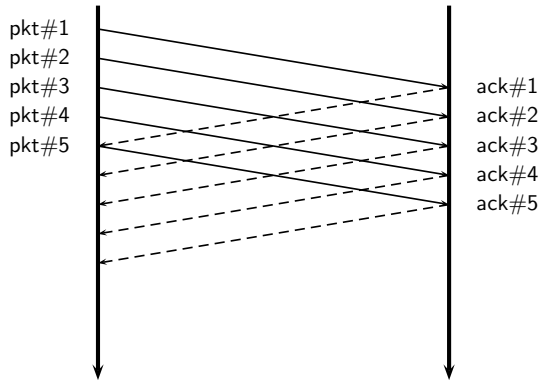
- Prenons le modèle simple du débit TCP suivant : $TCP_{throughput} = \sqrt{\frac{3}{4}} * \frac{MSS}{RTT * \sqrt{p}}$
- Nous avons deux facteurs qui impactent sur le débit moyen : RTT et p
- Un long délai a donc un impact sur les performances de TCP (problème du **self-clocking**)
- Les couches basses sont robustes $\rightarrow$ $BER \approx 10^{-7}$
  - Les erreurs liens résultent de la mobilité (*handover*) voire de conditions météo extrêmes
  - Problème : ce ne sont **pas des pertes de congestion**
- La performance de TCP est limitée par un fort **produit bande passante $\times$ délai**
  - Noté *LBDP : Large Bandwith Delay Product* ou *LFN : Long Fat Network*
- Il y a donc deux freins principaux : le **fort LBDP et les erreurs liens**

TCP NewReno

RFC6582

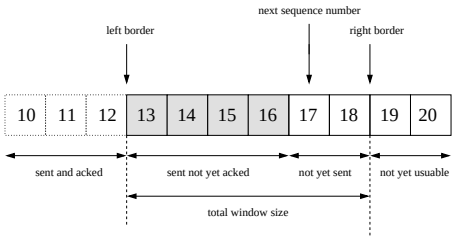
- **Slow-start** ($cwnd < ssthresh$) : $cwnd \leftarrow 2 \times cwnd$ every RTT (or $cwnd + 1$ on every ACK)
- **Congestion avoidance** ($cwnd > ssthresh$) : $cwnd \leftarrow cwnd + 1/cwnd$ on every ACK (or $cwnd + 1$ every RTT)
- Losses
  - **Fast retransmit** $cwnd, ssthresh \leftarrow cwnd/2$ on triple dup-ACK (a.k.a. losses ... or something else ?)
  - **Fast recovery** $cwnd \leftarrow ssthresh + n_{\mathrm{dupack}}$ for isolated losses
  - $cwnd \leftarrow 1, ssthresh \leftarrow cwnd/2$ on timeout
- Generalised $AIMD(a, b)$
  - CA : $cwnd \leftarrow cwnd + a$
  - **Loss** : $cwnd \leftarrow (1 - b)cwnd$
  - NewReno : $a = 1, b = 1/2$

TCP ACK scheme

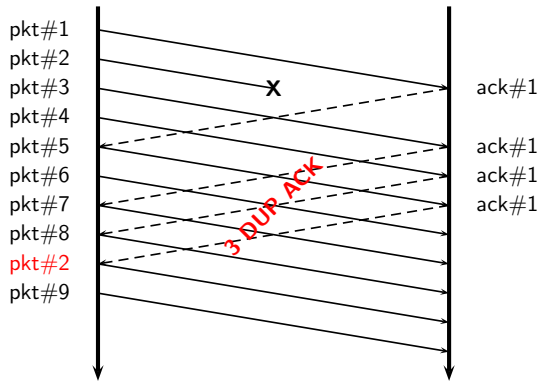


Stratégie pour éviter la congestion : la fenêtre glissante

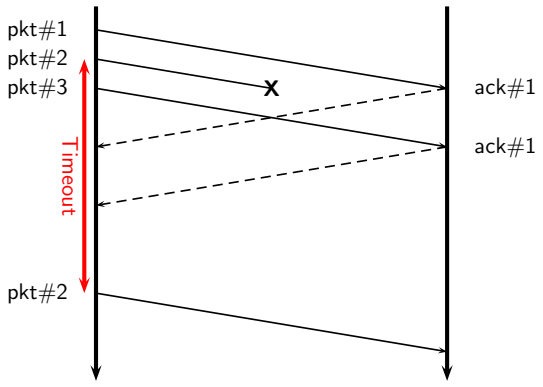
- Utilisation d'une "fenêtre glissante" (`cwnd`)
- `cwnd` indique la quantité de paquets que l'émetteur peut envoyer avant d'attendre des ACKs
- `cwnd` glisse au fur et à mesure que les premiers paquets envoyés sont acquittés (self-clocking)



TCP loss detection scheme



Short flow loss detection problem



Variantes TCP

- Le monde TCP n'est pas uniforme, il existe de nombreuses variantes
- L'implémentation par défaut varie suivant les OS
  - FreeBSD, OpenBSD : NewReno
  - Windows $<$ 10 : Compound ($\geq$Vista), Westwood+
  - Windows 10, OSX, GNU/Linux : CUBIC
- CUBIC est plus agressif que NewReno
- Autres contrôles de congestion : Vegas (delay-based), HSTCP, Hybla, TFRC (rate-based), ...
- Autres transports : DCCP, SCTP, LEDBAT, XCP, ...
- Variantes de *slow start*
  - Quick Start : nécessite l'assistance des routeurs (même philosophie que XCP)
  - Jump Start : estime la fenêtre à envoyer en fonction du RTT estimé lors du *three-way-handshake*

La problématique de TCP dans les réseaux à fort produit bande passante $\times$ délai

RFC1072

- Ce produit correspond a une quantité de données transmises mais non acquittée
  - Le délai du produit est donc le *RTT*
- La RFC 1072 qualifie un réseau comme LFN lorsque ce produit est supérieur à 10^5b
  - Lien terrestre optique : 1Gb/s $\times$ 1ms = 10^6b
  - Lien satellite : 1Mb/s $\times$ 500ms =5.10^5b
- TCP possède une fenêtre de taille max 65535 bytes $\rightarrow$ problème lorsque le $BDP > 64K$
  - Problème résolu par l'option **TCP window scale** RFC 1323
  - Activé par défaut dans GNU/LINUX depuis 2.6.8, MAC OSX et Windows depuis Vista
- Selon la RFC 3649 (HighSpeed TCP), un flot TCP NewReno seul sur un lien 10Gb/s et RTT=100ms devrait obtenir une perte d'un paquet après 5.000.000.000 paquets (1,75 h) au départ de la phase de *congestion avoidance*

La problématique des erreurs liens

- TCP ne distingue pas les pertes de congestion des erreurs liens
- En conséquence l'évolution de sa fenêtre est impactée par ces types de pertes : $cwnd \leftarrow cwnd/2$
- TCP Westwood (2001) est une variante TCP qui tente de distinguer ces pertes
  - La distinction s'effectue par estimation de la capacité de bout-en-bout
- Autre possibilité, utiliser le flag ECN pour différencier le lien d'où provient la perte
  - Idée de B. Briscoe : *congestion exposure (ConEx), re-feedback & re-ECN*
  - Problème du management des flags ECN par les routeurs intermédiaires

TCP Westwood $\rightarrow$ détecter les erreurs liens

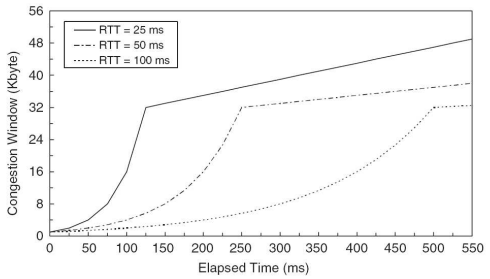
- L'idée clé est de mesurer en continu à la source TCP le débit de la connexion en monitorant les ACKs
- L'estimation est ensuite utilisée pour calculer fenêtre de congestion et le seuil du *slow start* après un épisode de congestion
  - i.e. après 3 ACK duliqués ou RTO
- La stratégie est simple : ne pas automatiquement diviser par deux la fenêtre de congestion après trois ACK dupliqués
- Pour cela TCP Westwood détermine un seuil de *slow start* et une fenêtre de congestion compatibles avec l'utilisation effective de la capacité au moment de la congestion
  - Les auteurs ont baptisé cette phase de *faster recovery*
- Evolution $\rightarrow$ Westwood+
  - amélioration de l'estimation de la capacité

Illustration du problème de TCP sur réseau à fort LBDP

- **Exercice :** prenons le cas réseau d'un lien satellite de capacité 1Mbps et de RTT nominal 500ms. Soit un flot TCP NewReno de MSS 1460B (soit un MTU de MSS+40=1500B), ayant effectué sa phase de *slow start* et en mode *congestion avoidance*. Dans combien de temps celui-ci atteindra la capacité maximum en supposant qu'il soit seul sur le lien et que l'option window scale soit activée ?

Impact du RTT sur TCP

- La réactivité de TCP diminue au fur et à mesure que le RTT augmente



Où sont les freins ?

- L'établissement de connexion : *three-way-handshake*
  - Pas de données utiles transférées
  - Problème accru pour les clients web qui multiplient le nombre de connexions
- Le *slow start* et la valeur initiale du *slow start*
- La phase de *congestion avoidance* qui croit de 1/W par RTT

Solutions possibles

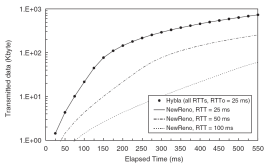
- Rendre plus aggressive la phase CA (CUBIC, HTCP, ...) $\rightarrow$ nécessaire mais pas suffisant
- Mitiger la dépendance du RTT (TCP Hybla)
- Changer la valeur initiale du SS
- Utiliser un SS plus agressif (e.g. *jump start*)
- Faire du TCP pacing
  - Afin d'éviter les débordements de tampon (on change la sporadicité du trafic)
  - Afin d'atteindre plus rapidement la capacité (RAPID, Initial Spreading, ...)

Sur le TCP/ACK pacing

- **V. Konda and J. Kaur, "RAPID : Shrinking the Congestion-control Timescale", IEEE Infocom 2009**
- **R. Sallantin et al., "Initial spreading : A fast Start-Up TCP mechanism", IEEE LCN 2013**
- **D. Wei et al., "TCP Pacing Revisited", 2006**

TCP-Hybla

- Supprimer la dépendance du RTT
- Accroitre la réactivité de TCP en prenant un RTT de référence (RTT_O)
- *slow start* : `cwnd = cwnd + 2^k - 1`
- *congestion avoidance* : `cwnd = cwnd + k^2/cwnd`
  - rappel NewReno : `cwnd = cwnd + 1/cwnd`
- `k = RTT/RTT_O`



Source

- **C. Caini and R. Firrincieli, "TCP Hybla : a TCP Enhancement for Heterogeneous Networks", in International Journal of Satellite Communications and Networking, 2004**

D'autres variantes TCP pour le satellite

- TCP Peach (2001)
  - changement des phases de *slow start* et *fast recovery* en *sudden start* et *rapid recovery*
  - introduction de *dummy segments* pour sonder la capacité disponible
  - l'idée est que ces *dummy segments* atteignent la destination seulement si le chemin n'est pas congestionné (marqués non-prioritaire dans le champs TOS)
  - problème nécessite la prise en compte du champ TOS par les routeurs du chemin
- TCP Noordwijk (2009)
  - projet ESA, baptisé selon la localisation du centre technique de l'agence spatiale européenne situé à Noordwijk, Pays-Bas
  - définition d'un TCP spécifique au DVB-RCS
  - objectif : trouver une alternative à TCP NewReno et TCP Vegas (opérant pour ce dernier correctement lorsqu'en isolation) intégré dans les I-PEPs

Autres propositions

- SCPS-TP (Space Communications Protocol Standards - Transport Protocol)
  - Extensions transport pour TCP et UDP en environnement spatial ⇒`http://www.scps.org/`
  - Protocole propriétaire
- DARTS (Data Acceleration and Reduction Technology)
  - Contrôle de congestion rate-based basé sur SCPS-TP
- Amélioration d'autres protocoles (**e.g.** DCCP) pour lien satellite
- Des variantes comme TCP-CUBIC se comportent mieux sur lien long-délai

Bibliographie

- **T. Iyer et al., "Data Acceleration and Reduction Technology", International Workshop on Satellite and Space Communications, 2009**
- **G. Sarwar et al., "Performance of VoIP with DCCP for Satellite Links", IEEE ICC 2009**

Bilan

- Il existe donc des alternatives protocolaires pour liens satellites
- Problème : un déploiement de bout-en-bout n'est pas envisageable et potentiellement dangereux pour l'équilibre de l'internet
- A vous de me dire pourquoi ... indice : repartez de l'équation de base de TCP et comparez l'impact avec celle d'Hybla

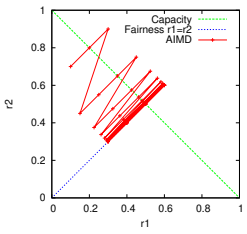
Rappel et réponse : TCP et l'équité

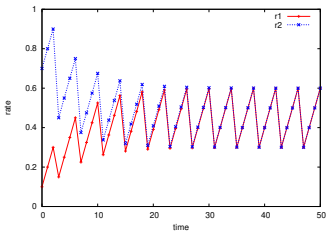
- TCP utilise l'algorithme AIMD pour partager équitablement les ressources
- Il existe d'autres algorithmes d'équité, exemple :
  - Multiplicative Increase, Multiplicative Decrease
  - Additive Increase, Additive Decrease
- Soit le débit $r_i(t)$ d'un emetteur r au temps t et $c(t)$ représentant l'indication binaire de congestion. Si l'on se restreint à un contrôle linéaire, la fonction de mise à jour du débit avec AIMD peut être exprimée de la façon suivante :

$$r_i(t+1) = \begin{cases} \alpha_{inc} + \beta_{inc}.r_i(t) & \text{si} \quad c(t) = 0 \\ \alpha_{dec} + \beta_{dec}.r_i(t) & \text{si} \quad c(t) = 1 \end{cases} \qquad (1)$$

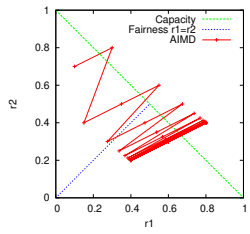
  - Pour AIMD : $\alpha_{inc} > 0; \beta_{inc} = 1; \alpha_{dec} = 0; 0 < \beta_{dec} < 1$
  - Pour MIMD : $\alpha_{inc} = 0; \beta_{inc} > 1; \alpha_{dec} = 0; 0 < \beta_{dec} < 1$
  - Pour AIAD : $\alpha_{inc} > 0; \beta_{inc} = 1; \alpha_{dec} < 0; \beta_{dec} = 1$
  - Pour MIAD : $\alpha_{inc} = 0; \beta_{inc} > 1; \alpha_{dec} < 0; \beta_{dec} = 1$

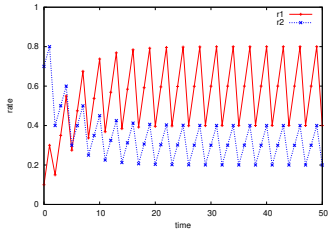
TCP vs TCP





TCP vs Hybla





- Solution : il faut conditionner l'utilisation de ces protocoles spécifiquement aux liens satellites
- Comment ? En scindant la connexion par le biais d'un **proxy**

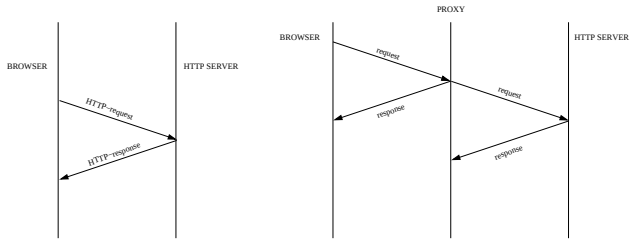
Les proxys

- Principe : connexion en 2 temps ⇒on se connecte au serveur puis à l'Internet
- Peut faire office de cache pour les pages web récemment consultées (le cache est paramétrable)
- Peut demander une authentification de l'utilisateur avant la sortie vers l'extérieur
- Fonctionnement identique à une authentification du type de connexion Internet
- On peut demander une connexion pour chaque site !
- Fonction de filtrage : peut interdire la connexion en fonction de mots inappropriés, analyser la présence de virus, contrôler la durée, l'heure de connexion et de déconnexion

- Il faut un proxy par application : proxy-web, proxy-ftp, proxy-snmp, ...
  - Aujourd'hui, beaucoup de proxies sont en fait des multi-proxy capables de gérer la plupart des applications courantes
- Le proxy peut alors modifier les informations à envoyer au serveur pour une application donnée

Exemple de cache Web

- Optimisation des requêtes HTTP



A propos de la fonction d'enregistrement

- Le serveur garde une trace détaillée de toutes les informations qui le traversent (fichier de log)
  - selon la loi européenne de conservation des traces celle-ci est bornée entre 6 mois minimum et 24 mois maximum
- Le serveur enregistre la trace des requêtes effectuées par tous les clients utilisant le proxy et notamment :
  - Identification du client
  - Dates et heures de connexion
  - Liens et ressources consultés
  - Taille et temps de téléchargement, etc

Proxy transparent

- Le client possède une configuration IP standard
- Chaque requête web est automatiquement et de manière transparente renvoyée vers le proxy par le routeur/firewall
- Le serveur proxy relaye la requete sur le serveur destinataire et stocke l'information dans son cache
- Le client a alors l'illusion d'une connexion directe

Proxy socks

- Socks est un protocole proxy générique pour TCP/IP
  - Standard IETF, RFC 1928
- Opère au niveau des couches transport et application
- Certains clients web (Firefox) implémentent un proxy-socks
- Il existe aussi une implémentation Linux

Reverse proxy

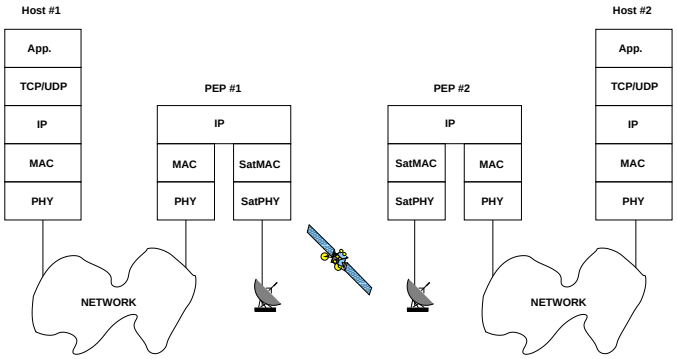
- Proxy inversé : permettre à des clients externes d'utiliser un service interne
- Achemine des requêtes extérieures vers les machines du réseau interne
- Garde les propriétés d'un proxy
- Un relais inverse n'a d'utilité que s'il apporte des fonctionnalités :
  - d'authentification, mots de passe certificats
  - de chiffrement
  - de filtrage applicatif

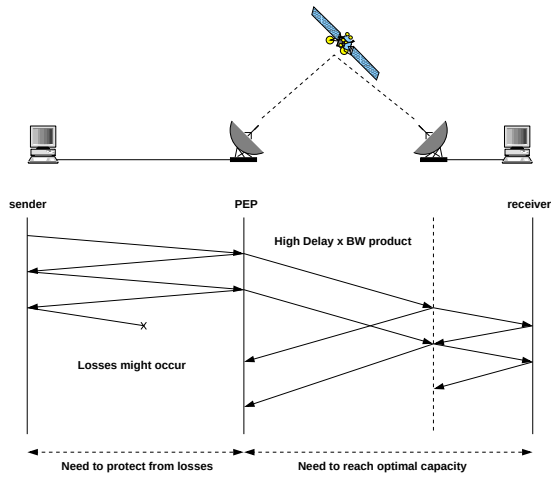
Performance Enhancing Proxy (PEP)

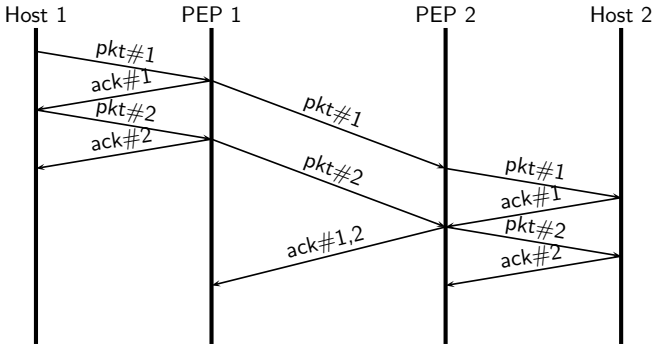
- Performance Enhancing Proxies sont des standards definis dans la RFC 3135 (Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations)
- Un PEP peut soit "couper" ou soit "snooper" une connexion :
  - SPLIT : scinde le réseau en deux parties : le proxy prétend être en face du point de terminaison de la connexion dans chaque direction
  - SNOOP : le proxy prend le contrôle sur les transmissions des segments TCP dans les deux sens par filtrage des ACKS et reconstruction (similaire au **spoofing**)

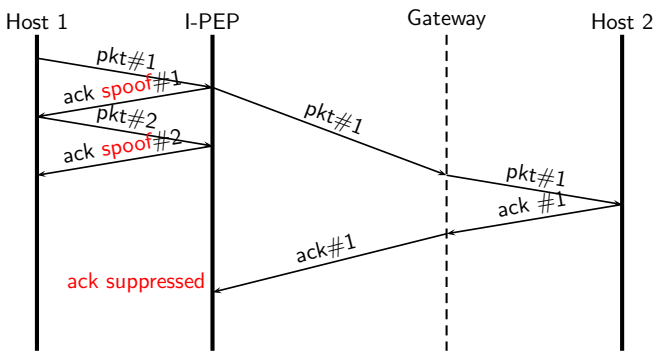
- Un PEP est soit autonome (**Integrated PEP**), soit distribué (**Distributed PEP**) :
  - **Integrated PEP (I-PEP)** : ils tournent en un seul point tandis que les PEPs distribués (**Distributed PEP**) sont présents aux deux côtés du lien, qui est à l'origine de la dégradation des performances
  - Les PEPs commerciaux agissent comme des "boites noires" utilisant des protocoles (propriétaires ou non) et qui encapsulent le protocole entrant

- L'implémentation d'un PEP est soit symétrique soit asymétrique :
  - Les PEP symétriques ont un comportement identique dans les deux directions, les actions entreprises par le PEP sont alors indépendantes de l'origine de l'interface physique des paquets entrants
  - Dans le cas asymétrique, les PEPs opèrent différemment dans chaque direction et peuvent donc améliorer les performances de façon asymétrique









- Favorise les transferts longs
- Augmente le débit vu du côté émetteur pour le trafic court
- L'avantage du débit est du côté de la communication *spoofed* et moins du côté récepteur
- Avantage d'utilisation pour les ISPs
- Inconvénient de la passerelle qui se retrouve à devenir un second goulot d'étrangelement si le trafic à mémoriser augmente drastiquement

Pour plus d'information

- **J. Ishac and M. Allman, "On the performance of TCP spoofing in satellite networks," in Military Communications Conference, IEEE MILCOM 2001**

PEPs commerciaux

Name	Mentat SkyX	ViaSat xPEP	MediaSputnik 2402	PEPSal
Type	Commercial box	Commercial box	Commercial Box	GPL
System	Linux	Linux and Solaris	Linux	Linux
TCP variants	XTP, SCPS-TS	TCP-XL	SCPS-TP	all TCP, TCP-Hybla (default)
Flow control	rate/window-based	window-based	rate-based	window-based
Error recovery	SNACK	ECN	SNACK	SNACK, F-RTO, ECN
Fast Start (RFC1644)	Yes	Yes		Yes
Compression	Yes	Yes	Yes	No
Prefetching	Yes	No		No

Pourquoi utiliser les PEPs ?

- Amélioration du délai de transfert des connexions de bout-en-bout (TCP ACK spacing)
  - En particulier TCP pacing
- Meilleure gestion de la communication et des retransmissions sur le lien satellite
  - Permet d'encapsuler le trafic (TCP Hybla, proxy SCPS)
- Meilleure fiabilisation par l'introduction de mécanismes de redondance (FEC)
- En bref : permet de micro-gérer la connexion TCP sur le lien long-délai

Implications de leur utilisation

- Argument du End-to-End
- Securité
- Routage asymétrique
- Passage à l'échelle
- Alternatives possibles avec des tunnels. Ex : RBSCP (Rate Based Satellite Control Protocol) solution propriétaire CISCO
  - Voir : `http://bit.ly/1HnC5ip`

Delay Tolerant Networking

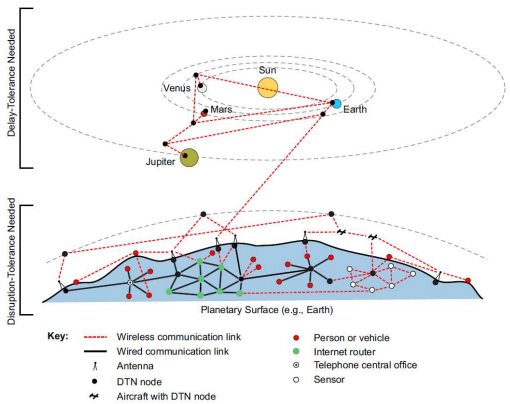
Définition

- Un DTN est un réseau capable de gérer des communications :
  - Longue distance (*deep space networks*) et long délai
    - ⋆ Equateur → GEO ≈ 250ms RTT
    - ⋆ Terre → Lune ≈ 1sec OWD
    - ⋆ Terre → Mars ≈ 15min OWD
  - Avec une connectivité intermittente
  - Sans hypothèse sur la pérénnité temporelle du chemin
- Les DTNs ont été développés pour les communications inter-planétaire
- Ce concept est également utilisé sur des petits réseaux terrestes dynamiques à forte déconnexions
  - Dans ce cas on parle plutôt de réseaux et communications opportunistes

Quel protocole de transport pour le DTN ?

- Ces caractéristiques nécessitent le déploiement d'une architecture adapatée
- Pourquoi ne pas utiliser la pile TCP/IP de bout-en-bout en adaptant le RTO par exemple ?
  - Le délai peut varier de quelques minutes à des heures, voire des jours
  - Problème : TCP est conçu pour négocier une connection bout-en-bout et d'acquitter les segments transmis un à un
- Solution : utilisation d'une approche *store and forward*, point à point, avec délégation de l'assurance de la transmission
- L'avantage du DTN est qu'il supporte l'interconnexion de réseaux hétérogènes en se chargeant du lien long délai et à fort déconnexion les séparant
  - Exemple d'un LAN terrestre connecté à un LAN martien

Un exemple complet

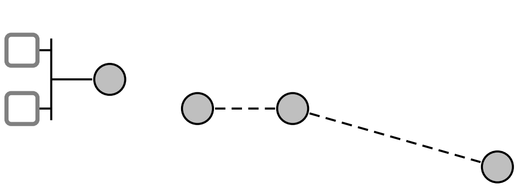


Source : Forrest Warthman, "Delay- and Disruption-Tolerant Networks (DTNs) - A Turorial", available online

Les challenges du DTN

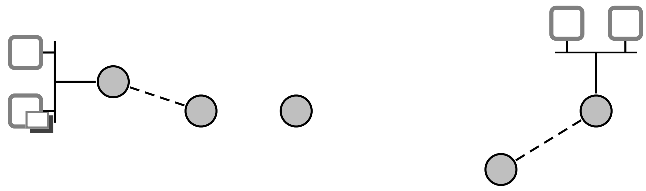
- Une communication sur un réseau DTN doit :
  - Etre fiable
  - Gérer la perte de la connexion
  - Gérer une connexion de mauvaise qualité
  - Gérer une connexion à RTT élevé
- L'architecture du réseau doit :
  - Gérer des unités de transmissions atomiques (*bundle data*) et asynchrones
  - Stocker les données dans l'attente d'une transmission
  - Gérer différentes couches protocolaires en fonction de la région DTN traversée
  - Optimiser la capacité de transmission
  - Gérer un plan d'adressage et de routage propre

Principe du Store and Forward



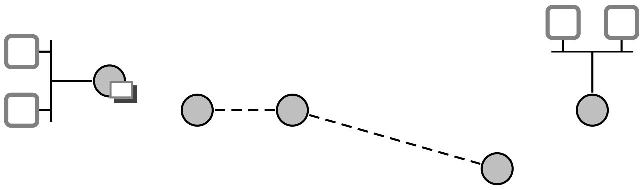
Soit la topologie ci-dessus où les lignes en pointillées représentent les contacts entre les nœuds

Principe du Store and Forward



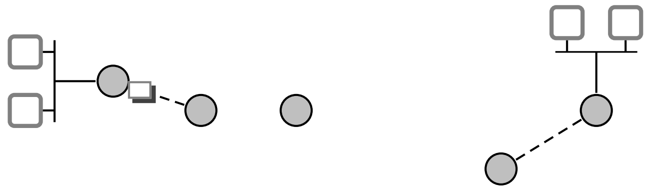
A t_1 un bundle est prêt à être émis

Principe du Store and Forward



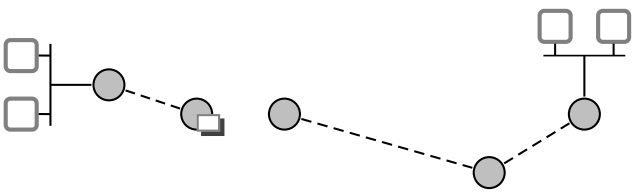
A t_2 il est stocké sur la passerelle dans l'attente d'un contact

Principe du Store and Forward



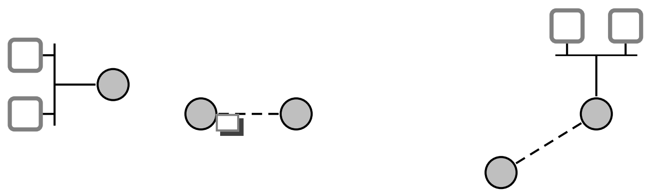
A t_3 il y a contact, le bundle est transmis

Principe du Store and Forward



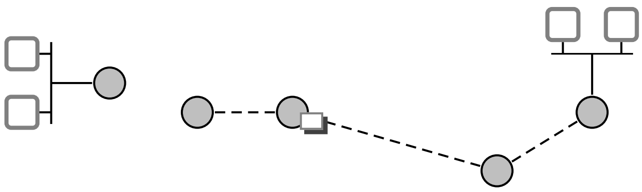
A t_4 il est stocké dans le nœud intérmédiaire dans l'attente de contact

Principe du Store and Forward



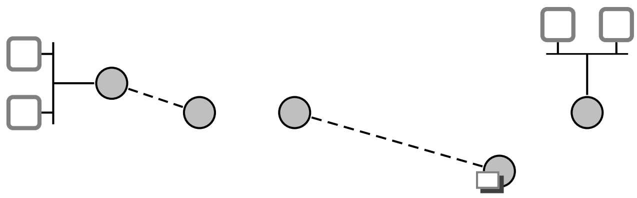
Cette opération continue de proche en proche ...

Principe du Store and Forward



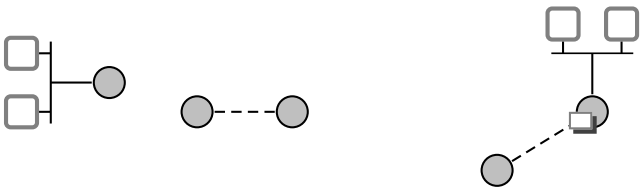
Cette opération continue de proche en proche ...

Principe du Store and Forward



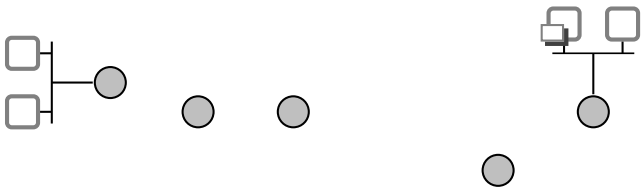
Cette opération continue de proche en proche ...

Principe du Store and Forward



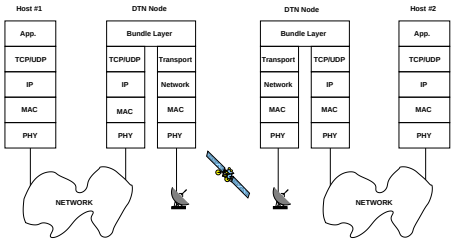
... jusqu'à atteindre la destination

Principe du Store and Forward



... jusqu'à atteindre la destination

L'architecture en couche du DTN et la couche Bundle



- Le stockage des *bundles* s'effectue dans la couche du même nom
- Cette couche comprend une sous-couche appelée *convergence layer* (non représentée) qui s'insère entre la couche transport et la couche *bundle*
- La *convergence layer* permet de faire le lien entre les différents protocoles de transport utilisés de part et d'autre
  - Exemple : permet de réceptionner les segments TCP et de les stocker dans l'attente de la réception complète d'un *bundle*

Custody Transfers

- DTN supporte la retransmission nœuds à nœuds des données perdues ou corrompues des couches transport et *bundle*
- Cependant, puisqu'il n'y a aucune connectivité de bout-en-bout, la fiabilité de bout-en-bout est transposée nœuds à nœuds
- La couche *bundle* permet la retransmission suivant le principe nommé *custody transfer* (CT)
- Ce mode permet l'établissement d'une horloge de retransmission dans l'attente d'un acquittement de bonne réception du paquet par un nœud (pas forcément le suivant)
- La valeur de retransmission est calculée localement ou distribuée lors du premier parcours du chemin
- Le paquet reste mémorisé par la couche *bundle* tant qu'un des nœuds suivant sur le chemin n'accèpte pas le mode CT (par retour d'acquittement) ou dans l'attente d'un acquittement
- Le CT améliore la fiabilité mais ne garantie pas la délivrance $\rightarrow$ seul un acquittement de la destination finale vers la source le permet (*return receipt*)

Classes de service de la couche bundle

- *custody transfer* : délégation de transmission (acquittement de proche en proche)
- *return receipt* : confirmation par la destination finale de la bonne réception des données à la source (acquittement de bout-en-bout)
- *priority of delivery : bulk, normal, expedited*
- *TTL* : bien évidemment, ne peut-être calculé en nombre de saut. C'est donc une valeur temporelle. Une bonne valeur est soit de prendre le temps de transit estimé jusqu'à la destination plus une certaine marge d'erreur

Protocoles de transport pour le DTN

- Licklider Transmission Protocol (LTP)
  - Introduit deux classes de paquets (rouges et verts)
  - Paquets rouges : transfert fiable, nécessite un acquittement
  - Paquets verts : délivrance immédiate sans ACK
- Saratoga
  - Protocole basé sur UDP/IP
  - Cherche à transmettre le maximum de données au débit du lien
  - Utilise SNACK
- CCSD File Delivery Protocol
  - Utilise NAK et codes FEC

Pour conclure

Figure 1 – CGBA5 onboard ISS being attended to by Astronaut Terrence W. Wilcutt (from [15]).

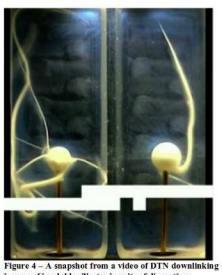


Figure 4 – A snapshot from a video of DTN downlinking images of insoluble silicates in spite of disruptions.

Source : Jenkins, A. ; Kuzminsky, S. ; Gifford, K.K. ; Pitts, R.L. ; Nichols, K., "Delay/Disruption-Tolerant Networking : Flight Test Results from the International Space Station", IEEE Aerospace Conference, 2010

Fin

Emmanuel Lochin `http://personnel.isae.fr/emmanuel-lochin/`