Vers l'Internet interplanétaire : mise en œuvre d'un réseau DTN dans un contexte de simulation d'une liaison spatiale

Florian Greff

Mémoire d'ingénieur

Vers l'Internet interplanétaire : mise en œuvre d'un réseau DTN dans un contexte de simulation d'une liaison spatiale

Florian GREFF

Année 2013–2014

Stage de fin d'études réalisé au Centre National d'Études Spatiales
en vue de l'obtention du diplôme d'ingénieur de TELECOM Nancy

Maître de stage : M. François LASSERE

Encadrant universitaire : M. Bertrand PETAT

Déclaration sur l'honneur de non-plagiat

Je soussigné,

Nom, prénom : GREFF, Florian

Élève-ingénieur régulièrement inscrit en 3e année à TELECOM Nancy

Numéro de carte d'étudiant : 30709085

Année universitaire : 2013–2014

Auteur du document, mémoire, rapport ou code informatique intitulé :

Vers l'Internet interplanétaire : mise en œuvre d'un réseau DTN dans un contexte de simulation d'une liaison spatiale

Par la présente, je déclare m'être informé sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

Je déclare en outre que le travail rendu est un travail original, issu de ma réflexion personnelle, et qu'il a été rédigé entièrement par mes soins. J'affirme n'avoir ni contrefait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier texte ou code informatique, dans le but de me l'accaparer.

Je certifie donc que toutes formulations, idées, recherches, raisonnements, analyses, programmes, schémas ou autre créations, figurant dans le document et empruntés à un tiers, sont clairement signalés comme tels, selon les usages en vigueur.

Je suis conscien) que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université, et qu'en cas de manquement aux règles en la matière, j'encourrais des poursuites non seulement devant la commission de discipline de l'établissement mais également devant les tribunaux de la République Française.

Fait à Toulouse, le 29 septembre 2014

Signature :

Mémoire d'ingénieur

Vers l'Internet interplanétaire : mise en œuvre d'un réseau DTN dans un contexte de simulation d'une liaison spatiale

Florian GREFF

Année 2013–2014

Stage de fin d'études réalisé au Centre National d'Études Spatiales
en vue de l'obtention du diplôme d'ingénieur de TELECOM Nancy

Florian GREFF
32 rue des Gravières
54180 HEILLECOURT
+33 (0)6 63 41 85 10
florian.greff@tncy.fr

TELECOM Nancy
193 avenue Paul Muller,
CS 90172, VILLERS-LÈS-NANCY
+33 (0)3 83 68 26 00
contact@telecomnancy.eu

Centre National d'Études Spatiales
18 avenue Édouard Belin
31401 TOULOUSE Cedex 9
+33 (0)5 61 27 31 31

Maître de stage : M. François LASSERE

Encadrant universitaire : M. Bertrand PETAT

Remerciements

Je tiens en premier lieu à exprimer toute ma gratitude M. François LASSERE, Mme. Hélène PASQUIER ainsi que le CNES pour m'avoir accordé ce stage. Rêvant depuis toujours d'espace, je suis heureux d'avoir pu parachever ma formation par la participation à ce projet.

Pour son encadrement et sa confiance, je remercie une fois encore M. François LASSERE, maître de stage. Merci également à l'ensemble des agents CNES avec lesquels j'ai eu la chance de travailler et/ou que j'ai côtoyés – le service DCT/PS/SGE, MM. Patrick GELARD, Emmanuel DUBOIS et Raymond SOUMAGNE – pour leur accueil, leur sympathie et le partage de leur expertise.

Je remercie M. Bertrand PETAT, encadrant TELECOM Nancy, pour ses nombreux conseils et encouragements, ainsi que M. Jean-Marie MOUREAUX et Mme. Catherine LAMBERT pour leur grand soutien dans ma recherche de stage.

Enfin, je remercie ma famille et belle-famille pour leur présence et leur aide durant toute ma formation d'ingénieur.

Résumé

Le stage s'inscrit dans un contexte de mise en place d'un démonstrateur bout-en-bout par le Centre National d'Études Spatiales. Il doit permettre l'évaluation de l'usage d'une architecture réseau nommée DTN (Delay-Tolerant Networking). Le DTN correspond à une pile protocolaire spatiale de niveau 3 et 4 appelée à se développer dans l'avenir, et faisant actuellement l'objet d'une standardisation au niveau des principales agences spatiales mondiales. L'intérêt majeur de ce protocole est de supporter les propriétés difficiles propres aux liaisons spatiales tout en garantissant l'intégrité des données à transmettre. Il s'inscrit également dans le paradigme d'Internet Interplanétaire, en proposant des mécanismes de niveau réseau à des systèmes de communication essentiellement basés sur la liaison de données.

L'objectif du stage est de participer à l'étude technique de ce démonstrateur : état de l'art, implémentation de solutions techniques et mise en œuvre avec l'architecture existante de communication spatiale. Une attention particulière est portée sur l'évaluation de performances des solutions utilisées ou conçues, afin d'aboutir à un retour d'expérience pertinent, identifiant les points d'amélioration à étudier dans le futur. En dehors de sa problématique principale, le stage s'intéresse également à l'évaluation plus globale du DTN. Via la promotion de ses principaux avantages, on peut espérer, un jour, pouvoir connecter Internet à d'autres planètes.

Mots-clés : espace, réseau, DTN, CNES.

Abstract

The internship took place in the context of the implementation of an end-to-end network demonstrator by the "Centre National d'Études Spatiales" (French Space Agency). The purpose was to evaluate the use of a network architecture called DTN (Delay-Tolerant Networking). DTN is related to a space protocol stack set to be developped in the future, and is currently under standardization by the world's major space agencies. A major advantage of this protocol is to withstand the harsh properties of space links while ensuring the integrity of data transmitted. It is also part of the Interplanetary Internet paradigm, proposing network mechanisms to communication systems mainly based on data link.

The objective of the internship is to participate in a technical study of this demonstrator : perform a state of the art, implement technical solutions and apply them to an existing architecture of space communication. Particular attention is focused on performance evaluation of the designed solutions in order to provide relevant feedback, identify areas of improvement to consider in the future. The internship therefore includes a global study of the DTN architecture. Through the promotion of its advantages, we hope to be able to one day, connect the Internet to other planets.

Keywords : space, network, DTN, CNES.

Table des matières

Avant-propos

Le stage s'est déroulé du 1er avril au 30 août 2014 au Centre National d'Études Spatiales à Toulouse. Il s'inscrit dans la problématique globale des communications spatiales, essentielles au bon déroulement des missions spatiales, qu'elles soient pleinement opérationnelles (satellites de télécommunications par exemple) ou scientifiques (exploration, étude de la Terre, ...). Dans ce contexte, on observe également une croissance des télécommunications interplanétaires ou vers l'espace profond, dont les conditions rendent parfois difficile l'usage des méthodes classiques.

En parallèle, pour fournir un accès au réseau Internet à des régions faiblement connectées, une architecture nommée DTN pour *Delay-Tolerant Networking* a été imaginée. En proposant des mécanismes de niveau réseau dédiés au support des conditions de communications difficiles, elle apparaît également comme une réponse aux problématiques propres aux réseaux spatiaux (délais, pertes, ...). Le stage s'inscrit dans le contexte de l'étude de cette architecture innovante, appliquée aux réseaux spatiaux.

Après avoir brièvement présenté la structure d'accueil du stage, nous détaillerons le contexte technique et les motivations entourant la conception de DTN. La troisième partie décrira l'étude technique réalisée, à savoir mise en œuvre d'un réseau DTN bout-en-bout sur protocoles spatiaux, étape par étape. Enfin, nous proposerons un retour d'expérience et une synthèse de l'ensemble du travail réalisé.

Chapitre 1

Présentation de la structure d'accueil

1.1 Le Centre National d'Études Spatiales

(Largement inspiré de Wikipédia, l'encyclopédie libre)

Le **Centre National d'Études Spatiales** (**CNES**) est un établissement public à caractère industriel et commercial (EPIC) français chargé d'élaborer proposer au gouvernement la stratégie spatiale française et de la mettre en œuvre. Il est placé sous la tutelle conjointe des ministères de la Recherche et de la Défense.

Le CNES travaille en collaboration avec Arianespace (Évry) et l'Agence Spatiale Européenne (ESA), sur cinq domaines d'activité :

1. Accès à l'espace (lanceurs) ;
2. Terre, environnement et climat (développement durable) ;
3. Applications grand public ;
4. Science et innovation ;
5. Sécurité et Défense.

Le Centre National d'Études Spatiales est à l'origine de quantité de projets spatiaux, même si ce n'est pas lui qui fabrique les lanceurs ou les satellites. Dans le cas des lanceurs, après avoir conçu la filière Ariane, le CNES agit aujourd'hui comme autorité de conception et de qualification pour le compte de l'État français qui est l'État de lancement. Il joue aussi le rôle d'assistant au maître d'ouvrage, l'Agence Spatiale Européenne, pour les nouveaux développements.

Le CNES dispose d'un **budget de 1,911 milliards d'euros** en 2010, soit le plus important en Europe malgré une relative stagnation au cours des dernières années. Il comprend la part reversée à l'Agence Spatiale Européenne (685 millions d'euros), consacrée essentiellement aux missions scientifiques (astronomie, exploration du système solaire, étude de la Terre) et aux investissements dans les lanceurs. La part investie directement par le CNES porte dans l'ordre d'importance sur les lanceurs et la gestion de la base de lancement de Kourou (324 M€), les missions militaires (269 M€), scientifiques (167 M€), l'étude de la Terre généralement dans le cadre de coopérations binationales (120 M€), les

développements autour des satellites de télécommunication et de navigation par satellite (42 M€).

La localisation du CNES est divisée entre 4 centres (~2 400 employés) :

- le **siège à Paris** (~200 employés) ;
- la **Direction des Lanceurs** à Paris (~200 employés) : assure le développement des lanceurs Ariane et accompagne la phase de production industrielle pour le compte d'Arianespace. Elle prépare également l'avenir en travaillant sur les nouvelles générations de lanceurs et de systèmes de propulsion ;
- le **Centre Spatial Guyanais** (~270 employés) : port spatial de l'Europe, à Kourou ;
- le **Centre Spatial de Toulouse** (~1 700 employés) : lieu du stage (cf. ci-dessous).

Établissement public, le CNES a dès sa création fait un effort d'information et d'animation pour faire connaître et apprécier les activités spatiales par les jeunes et leurs enseignants.

1.2 Le Centre Spatial de Toulouse

(Largement inspiré de Wikipédia, l'encyclopédie libre)

Le Centre Spatial de Toulouse est le centre technique et opérationnel du CNES. Créé en 1968, **plus de 1 700 salariés** y sont occupés au développement de la plupart des travaux de la responsabilité du CNES, à l'exception des lanceurs et de leurs lancements : conception, développement, réalisation en partenariat avec les entreprises industrielles et les laboratoires scientifiques, mise en orbite, contrôle et exploitation des véhicules et systèmes spatiaux, satellites, sondes et ballons. Il mène simultanément une quarantaine de projets spatiaux en coopération avec une vingtaine de pays.

Il accueille également le centre de contrôle du véhicule automatique de transfert européen (ATV), dont le dernier a été lancé le 29 juillet 2014. Ses missions couvrent la plupart des tâches techniques et d'assistance aux scientifiques comme le management des projets, les études de recherche et technologie, les centres d'opération pour les mises à poste et la gestion en orbite, les moyens informatiques et d'études mathématiques, les supports (administration, logistique et communication). Le CNES exerce en outre à partir du Centre Spatial de Toulouse une délégation de maîtrise d'ouvrage pour la Direction Générale de l'Armement.

En plus de ses programmes d'activité, le CNES engage ou développe dans l'établissement toulousain des travaux lui permettant de préparer les évolutions et les projets spatiaux futurs. Certaines de ces activités transversales concernent les plateformes satellites et d'autres les organismes de préparation et d'exploitation de missions pérennes.

C'est au Centre Spatial de Toulouse, au sein du service « Systèmes Sol Génériques » de la sous-direction « Produits et Segments Sol », que s'est déroulé mon stage.

1.3 Présentation du service d'accueil

1.3.1 La sous-direction « Produits et Segments Sol »

La Sous-Direction « Produits et Segments Sol » (DCT/PS) a pour missions :

- de proposer et mettre en œuvre la politique technique des systèmes sol pour les projets orbitaux du CNES ;
- d'assurer la maîtrise d'ouvrage, la maîtrise d'œuvre ou l'ingénierie système des développements des segments sol et le rôle d'autorité de conception dans les domaines suivants :
  - segments sol d'opérations : centres de contrôles, systèmes et interfaces de transmission de données au sol,
  - segment sol utilisateurs : centres de mission, centres de données images et scientifiques, systèmes de traitement, de gestion et de valorisation des données.
- d'apporter son support au développement des systèmes informatiques pour les métiers et les projets : simulateurs, ateliers de conception, ateliers système et d'ingénierie ;
- de contribuer à la mise en œuvre de la politique du CNES pour les données scientifiques ;
- de proposer et conduire les études R&T (Recherche et Technologie) dans son domaine de compétence et les actions visant à rendre disponibles les techniques et technologies nécessaires, de développer et maintenir les compétences et de capitaliser le retour d'expérience ;
- de développer, maintenir et promouvoir l'utilisation par les projets de produits réutilisables, de standards, de techniques d'ingénierie et de favoriser la mutualisation, au sein du CNES ainsi qu'avec les autres agences spatiales ;
- de développer la coopération européenne dans son domaine.

1.3.2 Le service « Systèmes Sol Génériques »

Au sein de la Sous-Direction Produits et Segments Sol, le service Systèmes Sol Génériques (DCT/PS/SGE) exerce son activité dans le domaine des segments sol d'opérations. Il définit, développe et maintient les composants et systèmes logiciels génériques nécessaires en particulier aux systèmes de contrôle et de mission, aux échanges de données et à l'interface informatique avec les stations ainsi qu'aux simulateurs associés. Il contribue à l'élaboration des normes et standards relatifs aux segments sol, à l'interface sol/bord, et il participe aux groupes de standardisation appropriés.

Depuis de nombreuses années, DCT/PS/SGE maintient pour le compte des projets du CNES un **laboratoire Télémesure et Télécommande** (TM/TC). Il est essentiellement axé sur des activités d'étude R&D, de mise en œuvre et d'utilisation des différentes techniques relatives aux protocoles de transmission utilisés dans les systèmes de télémesure et de télécommande tels que spécifiés par les normes et standards du monde spatial, ainsi que des activités de développement, de support et d'expertise aux projets (bord et sol).

Les travaux d'étude, de test/validation et d'expertise effectués par le laboratoire TM/TC de DCT/PS/SGE ont rapidement fait apparaître la nécessité de disposer d'un ensemble de moyens de tests et d'outils de simulation adaptés aux différents cas. Un banc de test désigné **Plate-forme TM/TC CCSDS** a été mis en œuvre sur la base de ces exigences.

Cette plate-forme regroupe aujourd'hui un ensemble d'éléments fonctionnels matériels et logiciels réalisés conformément aux recommandations du CCSDS (Consultative Committee for Space Data System) et spécifications des standards ESA relatives au codage de canal, aux protocoles de télémesure et de télécommande par paquets. Ces différents éléments fonctionnels ont été réalisés soit en interne, soit dans le cadre de contrats industriels de R&T. La plate-forme constitue ainsi une configuration « temps-réel » réellement représentative d'une liaison spatiale.

Chapitre 2

Contexte technique du stage

2.1 Les réseaux spatiaux

2.1.1 Description

L'Internet terrestre tel que nous le connaissons aujourd'hui est structuré par des propriétés connues, telles qu'une forte connectivité, une stabilité importante, une symétrie des connexions. Ces caractéristiques permettent aux protocoles du paradigme TCP/IP de supporter efficacement le réseau, garantissant une transmission fiable tout en proposant des fonctionnalités fluidifiantes (contrôle de congestion, contrôle de flux).

L'étude décrite dans le présent rapport s'intéresse plus particulièrement aux réseaux spatiaux. Par défaut, le réseau spatial définit l'ensemble du chemin technologique permettant à un centre de contrôle de communiquer avec un objet spatial, généralement un satellite en orbite (Figure 2.1). La communication est ensuite structurée par deux voies, la voie TéléCommande (ou TC) et la voie TéléMesure (ou TM). On parle alors de communication TM/TC.

Cependant, les réseaux spatiaux souffrent de contraintes importantes, les classant dans la catégorie des «réseaux difficiles» :

- **connectivité intermittente** : les stations sol ne peuvent pointer en permanence vers le satellite, celui-ci peut être placé sur une orbite non géosynchrone[1]... Le maintien d'une connexion point-à-point (par extension, bout-en-bout) permanente est ainsi, sauf cas particulier, inenvisageable ;
- **taux de pertes important** : pour certaines communications bord[2]/sol, il peut atteindre 10% (on estime à 0,1% en moyenne le taux de pertes dans un réseau terrestre classique) ;
- **asymétrie des connexions** : dans un contexte de communication spatiale, le rapport de liaison montante/descendante peut atteindre les 1/1000. Certaines missions (interplanétaires notamment) impliquent même une unidirectionnalité des liaisons ;
- **délais conséquents** : inférieurs à 1 seconde dans le cadre d'une communication

1. Se dit d'une orbite dont la période est égale à celle de la rotation de la Terre
2. Bord désigne tout ce qui est embarqué dans un véhicule spatial (tel qu'un satellite, une sonde, ...)

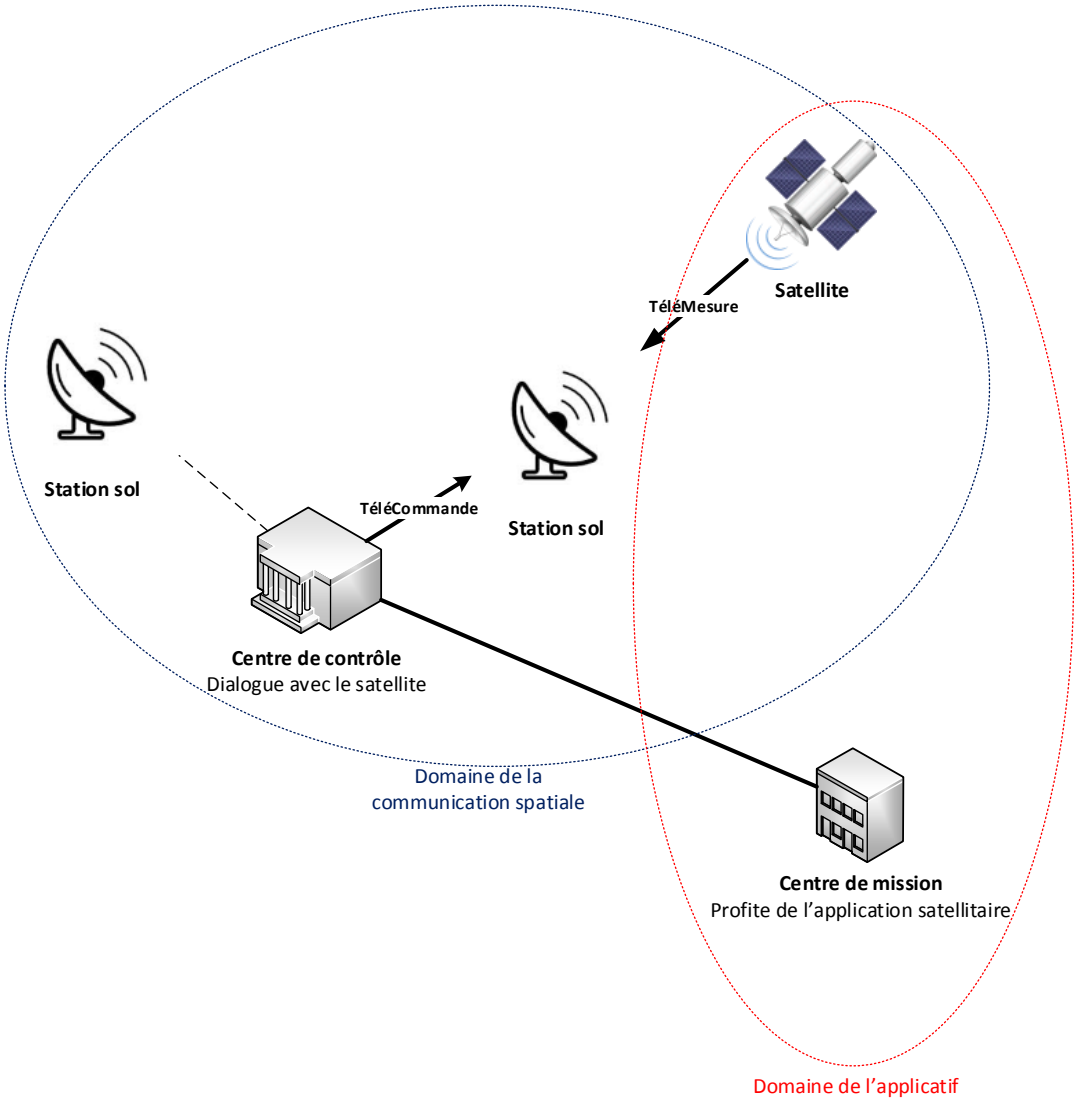


FIGURE 2.1 – Architecture de communication spatiale

avec un satellite en orbite basse, les délais spatiaux restent largement supérieurs aux délais terrestres, atteignant aisément l'heure pour des communications *deep space*[3] (de 3 à 20 minutes environ pour une communication Terre – Mars). Ainsi, chaque transmission est très coûteuse, l'interactivité bidirectionnelle devient rapidement lourde.

Ces caractéristiques rendent impossible l'utilisation directe de la suite protocolaire TCP/IP, ce qui implique la définition et l'implémentation de nouvelles architectures réseaux.

2.1.2 Solutions réseau

Les protocoles utilisés aujourd'hui en opérations spatiales sont essentiellement des protocoles de liaison de données (niveau 2), associés à des modes de transmission physique

3. Vers un espace profond, par exemple vers une sonde lointaine

de type radiofréquence (Figure 2.2). Le contrôle de fiabilité est présent au niveau de ces couches inférieures, car aucun mécanisme de transport tel que TCP n'est utilisé.

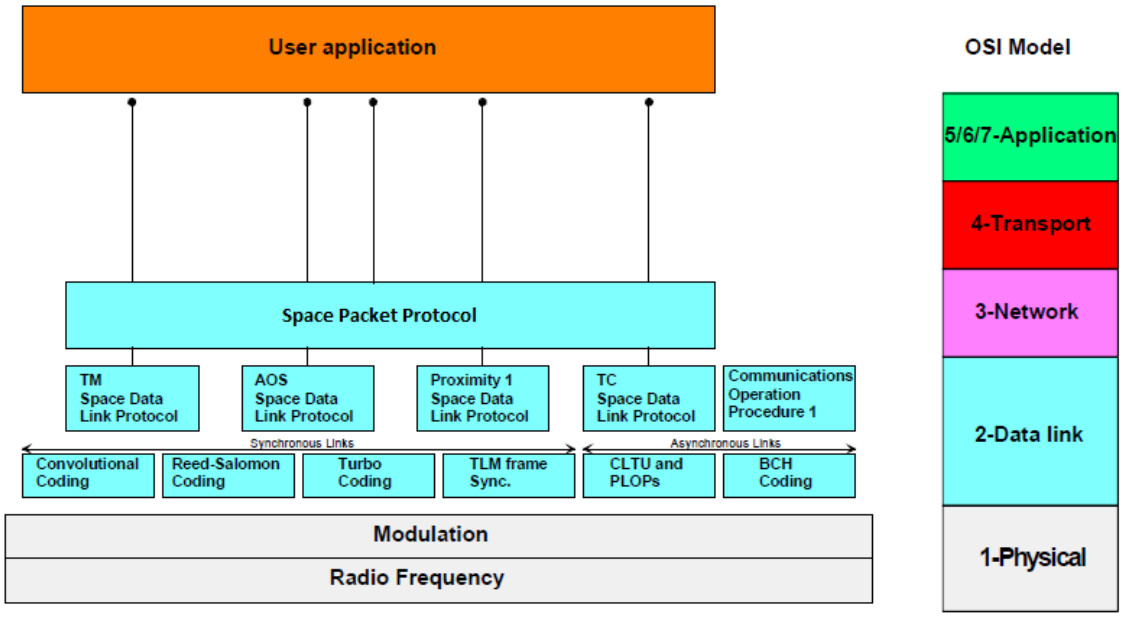


FIGURE 2.2 – Exemple de pile protocolaire pour la TM/TC

En termes applicatifs, le CCSDS (Consultative Committee for Space Data System, consortium international dédié à la standardisation des protocoles et systèmes de communication spatiaux) a défini un standard protocolaire, **CFDP** (CCSDS File Delivery Protocol). L'objectif est de tendre vers une harmonisation des méthodes de transfert de télémesure et télécommande, via un protocole souple et efficace.

Cependant, ce paradigme spatial souffre de quelques défauts actuels :

1. Les applications sont spécifiquement conçues pour supporter les contraintes du réseau spatial. Elles ne sont donc en général pas réutilisables telles quelles dans un autre contexte réseau (autres protocoles) ;
2. Un chemin bout-en-bout est nécessaire à chaque transmission et, en cas de perte, la retransmission s'effectue depuis la source du message ;
3. Un nœud de réseau spatial ne peut communiquer avec un nœud de réseau terrestre que par des mécanismes de tunnels [4] et/ou *proxies* [5] : les protocoles correspondant à ces deux types de réseaux sont incompatibles sans action intermédiaire.

Dans un objectif d'homogénéisation de ce type de communications, l'IRTF DTNRG [6] a conçu une nouvelle architecture réseau, le **DTN** pour *Delay-Tolerant Networking*. DTN s'appuie sur une couche réseau nommée **Bundle Protocol** pour proposer une réponse globale à la problématique des réseaux difficiles. Notamment, les DTN sont caractérisés par leur forte résistance aux délais, aux coupures de connexions, aux asymétries des liaisons.

On se place alors dans une perspective d'évolution des protocoles spatiaux bas niveau vers des protocoles de niveau réseau et transport, dont le DTN fait partie.

4. Encapsulation de données d'un protocole réseau dans un autre
5. Composant logiciel informatique qui joue le rôle d'intermédiaire en se plaçant entre deux hôtes pour faciliter ou surveiller leurs échanges
6. Internet Research Task Force Delay-Tolerant Networking Research Group

Il est à savoir que l'architecture DTN découle du paradigme d'**Internet Interplanétaire**, croisé avec le besoin d'un protocole robuste pour les réseaux dynamiques (MANETs[7] etc.). L'Internet Interplanétaire a été imaginé à la fin du XXème siècle, dans un souci de construction d'une architecture de communication robuste entre la Terre et Mars, qui ne pouvait pas être l'Internet terrestre pour les raisons explicitées plus haut.

A long terme, l'Internet Interplanétaire est imaginé pour interconnecter des régions spatiales (par exemple, l'Internet terrestre, ou un réseau ad-hoc situé sur Mars…) via un « *backbone*[8] spatial », l'ensemble étant fédéré par l'architecture DTN (Figure 2.3). Ainsi, on obtiendrait un réseau dynamique possédant trois propriétés particulièrement intéressantes :

1. **Facilité de la mise en œuvre** : le réseau spatial peut être utilisé de façon transparente par un utilisateur terminal tel qu'un rover, une sonde spatiale… Il « suffit » de le connecter au réseau après l'avoir configuré, ce n'est plus l'ensemble du chemin de communication qui est reconstruit à chaque mission. De même, les applications sont totalement déchargées des contraintes du réseau sous-jacent. Elles deviennent réutilisables, peuvent être conçues par des développeurs sans lien avec le spatial, etc. ;
2. **Haute disponibilité** : par l'utilisation de l'ensemble des objets spatiaux disponibles, la connectivité bout-en-bout devient très forte voire permanente ;
3. **Tolérance aux conditions difficiles** : naturellement, et indépendamment de l'exploitation des deux propriétés ci-dessus, le réseau interplanétaire profite de la robustesse de l'architecture DTN par rapport aux conditions difficiles propres aux réseaux spatiaux (délais, dynamisme des nœuds, …). Ceci sans alourdir l'application utilisant le réseau : le bilan de connexion s'en trouve amélioré.

Cependant, compte tenu de la difficulté de mise en œuvre d'une telle technologie, pour des raisons opérationnelles (les stations sol sont utilisées et on ne peut pas les modifier comme on veut) mais aussi de sécurité (comment laisser un message parmi d'autres transiter sur mon satellite ?), la réalité de cet Internet Interplanétaire n'est pas envisageable à moyen terme. Cela reste néanmoins un vecteur de motivation continu autour des DTNs spatiaux.

7. Mobile Ad hoc NETwork : réseau sans infrastructure dans lequel toutes les stations peuvent être mobiles

8. Historiquement, réseau informatique faisant partie des réseaux longue distance de plus haut débit Internet. Désigne de manière plus générale un réseau central d'interconnexion robuste

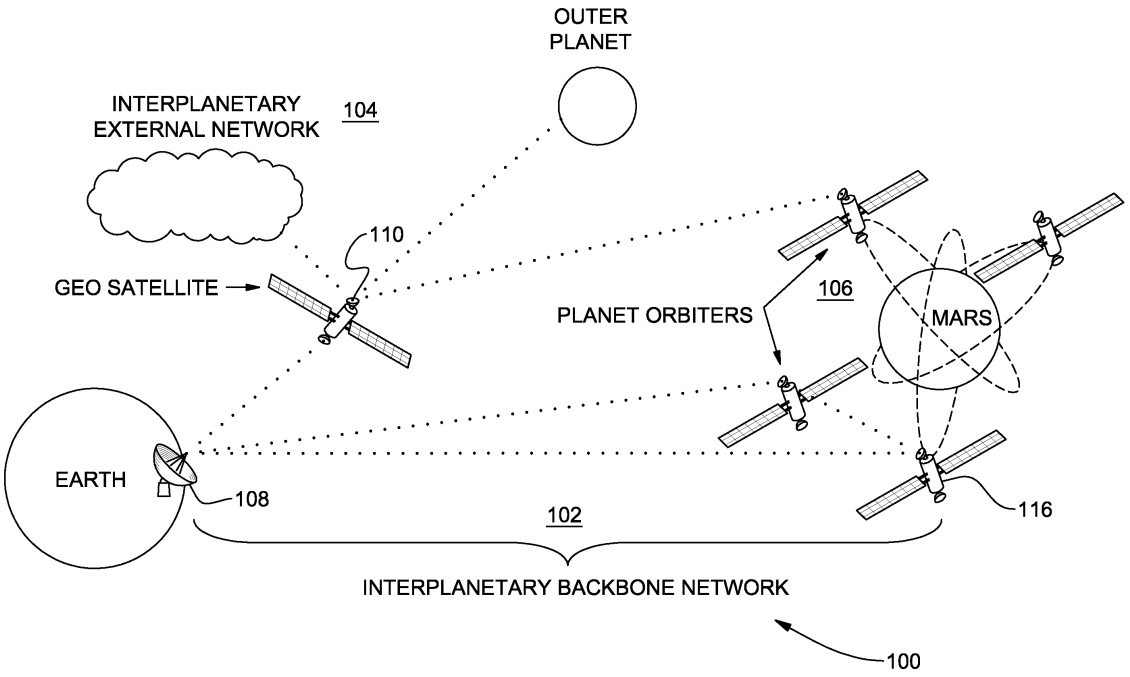


FIGURE 2.3 – Illustration de l'Internet Interplanétaire (source [16])

2.2 L'architecture DTN

2.2.1 Paradigme

L'architecture DTN définit un modèle de surcouche réseau (*overlay*) tolérante aux délais et aux disruptions. Pour ce faire, elle remodélise l'ensemble du réseau de communication comme un réseau de «nœuds DTN», interconnectés à l'aide d'une couche nommée **Bundle** (Figure 2.4). Cette couche se place directement en-dessous de la couche applicative et permet la mise en œuvre du réseau DTN au travers de deux mécanismes : le *store-and-forward* et la couche de convergence.

Premièrement, Bundle implémente un mécanisme de *store-and-forward*, c'est-à-dire un stockage persistant des données au niveau de chaque nœud intermédiaire (Figure 2.5). La retransmission des *bundles* – l'unité de donnée – perdus est alors directement amorcée par le dernier nœud traversé. Ainsi, l'information est acheminée par étapes successives (*hop-by-hop*), grâce au **custody transfert**, c'est-à-dire le transfert de la responsabilité de l'acheminement d'un segment de données à un nœud voisin.

En effet, lors d'une communication Internet classique, la responsabilité de transmission de l'information est conservée par l'application émettrice, ce qui implique que les segments de données sont retransmis depuis ce nœud origine en cas de défaillance d'un élément intermédiaire du réseau. A l'aide du stockage persistant des bundles, un nœud DTN est capable de déléguer cette responsabilité au nœud DTN voisin en même temps que le bundle effectif.

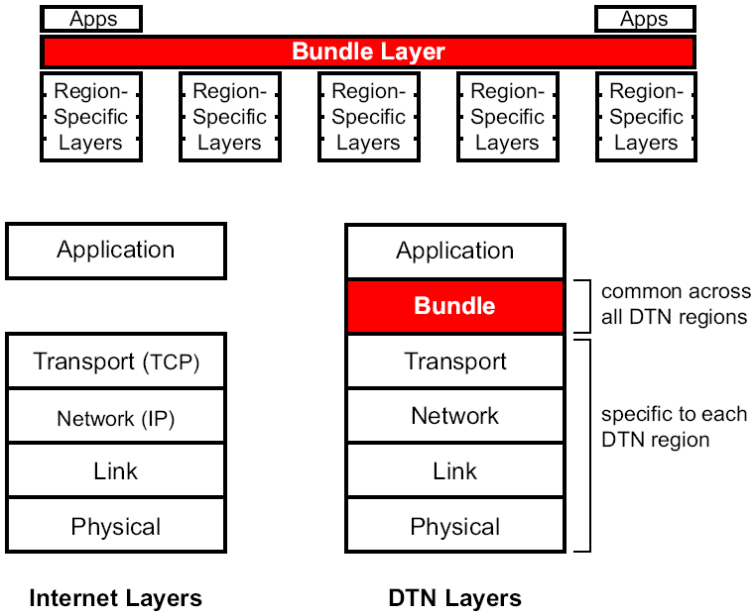


FIGURE 2.4 – La couche d'overlay Bundle (source [21])

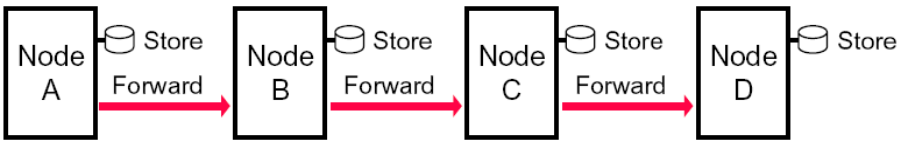


FIGURE 2.5 – Mécanisme de store-and-forward (source [21])

Il s'agit d'un mécanisme essentiel à la tolérance aux conditions difficiles pour trois raisons :

1. Les nœuds intermédiaires étant capables de retransmettre directement l'information, les pertes deviennent beaucoup moins coûteuses pour la communication (pas besoin de parcourir à nouveau l'ensemble du chemin depuis l'application émettrice, en subissant à nouveau les pertes et délais).
2. Lorsque la transmission d'un bundle à un nœud voisin s'est déroulée avec succès, on est certains que la communication bout-en-bout a avancé d'une étape. Cela permet d'acheminer l'information dans des environnements extrêmement contraints. On pensera par exemple aux réseaux radio militaires, évoluant en environnement disruptif (Figure 2.6). Lors de la traversée d'un réseau caractérisé par des pertes conséquentes, il est inconcevable d'espérer une communication ne subissant aucune perte, de l'émetteur jusqu'au récepteur. En revanche, si celle-ci est gérée nœud par nœud, par étape, elle devient possible dès que l'on dispose d'un mécanisme de transmission point-à-point efficace (c'est-à-dire qu'il va permettre, même si cela est long et/ou complexe, de transmettre un bundle à son voisin). Ceci est également applicable aux réseaux spatiaux.

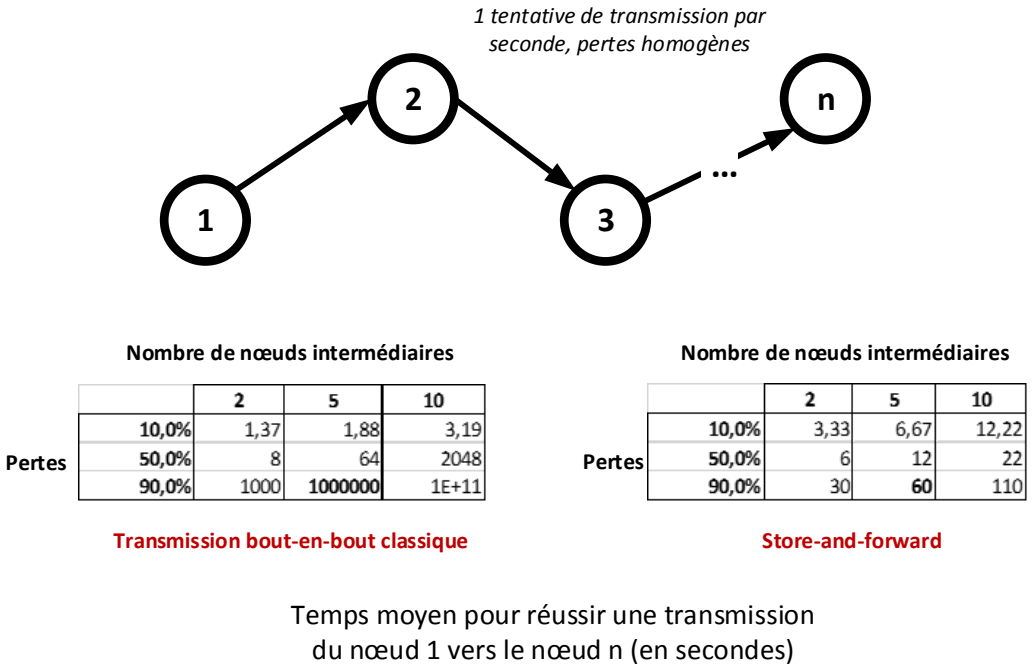


FIGURE 2.6 – Temps moyens de transmission dans un réseau ad hoc disruptif

3. Il n'est plus nécessaire de disposer d'un chemin bout-en-bout pour amorcer une communication. Celle-ci peut progresser à chaque fenêtre de communication (appelée contact, cf. plus bas) entre deux nœuds voisins. La fenêtre globale pendant laquelle des échanges réseaux faisant progresser la communication finale sont possibles est ainsi grandement augmentée (Figure 2.7). De plus, le caractère asynchrone de cette démarche amène la viabilité de nouveaux types de réseaux de communications. Par exemple, à titre d'illustration, une *data mule*[9] qui ferait des voyages entre deux centres de données proposerait une transmission à fort délais mais grand débit tout en étant transparente du point de vue des utilisateurs finaux. Ceci n'est pas possible par l'utilisation unique d'un réseau de type TCP/IP.

Le deuxième mécanisme mis en œuvre est celui de **couches de convergences**. En effet, Bundle Protocol agit comme surcouche réseau et ne gère en aucune façon la transmission effective des bundles au nœud voisin. Pour cela, il utilise une couche de convergence ; celle-ci permet l'exploitation par le nœud DTN de la pile protocolaire sous-jacente (par exemple, TCP/IP), comme l'illustre la (Figure 2.8).

Ainsi, en permettant au réseau DTN de vivre sans tenir compte des contraintes réelles du réseau, les couches de convergences induisent la deuxième caractéristique essentielle des DTNs : la possibilité d'interconnexion de régions [10] hétérogènes (Figure 2.9).

En passant par un réseau overlay tel que DTN, il est possible de faire communiquer des éléments de réseaux différents (protocoles différents) sans devoir utiliser des passerelles. Du point de vue de l'utilisateur final, ce mécanisme est totalement transparent puisque la couche Bundle est commune à tous les nœuds du DTN. Elle agit alors comme une couche réseau, de la même façon que le protocole IP permet d'interconnecter des ma-

9. Véhicule transportant physiquement un support de stockage entre deux régions éloignées, dans le but de créer un lien de communication effectif
10. Réseau ou sous-réseau disposant d'une cohérence en termes de politique protocolaire

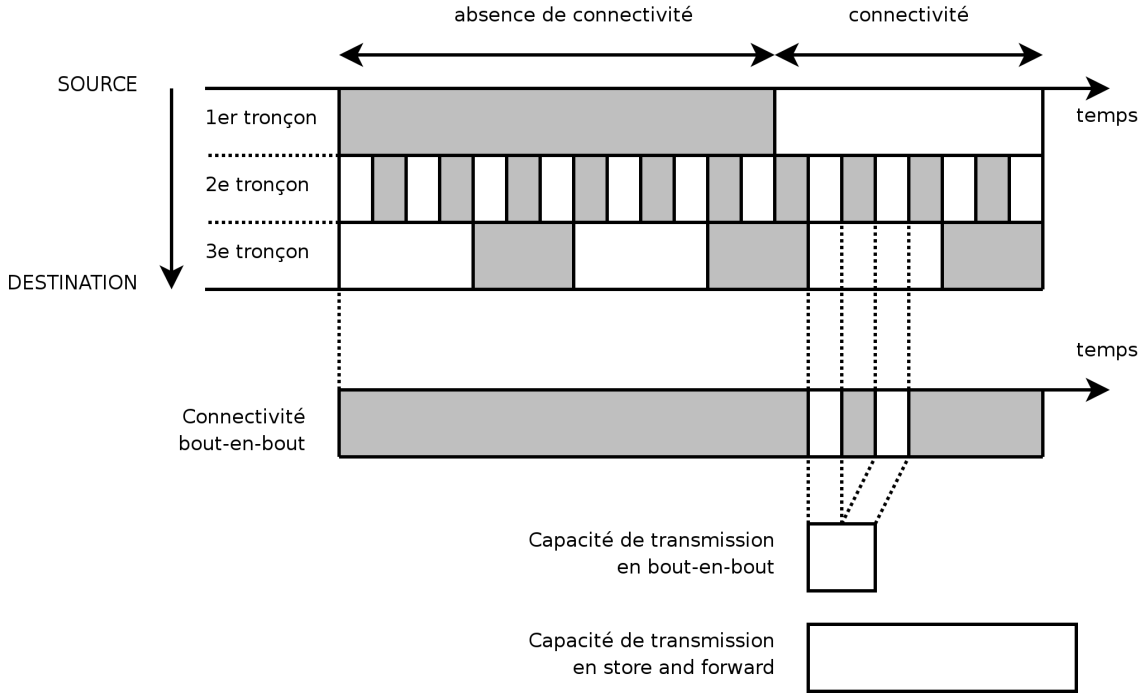


FIGURE 2.7 – Transmission DTN versus TCP/IP (source [14])

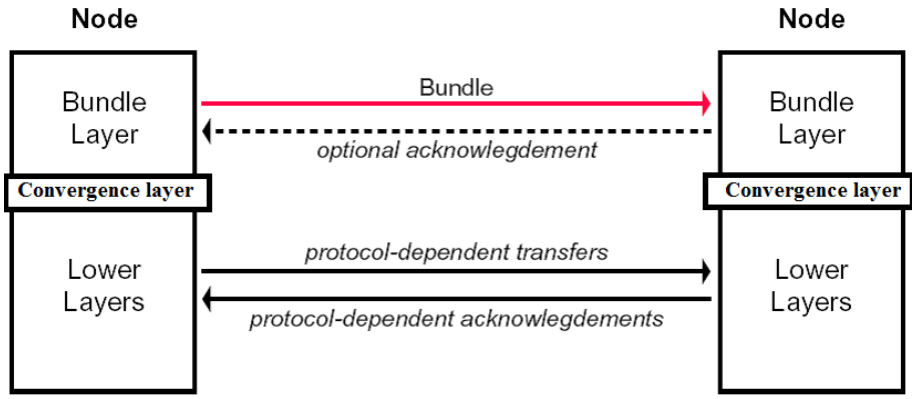


FIGURE 2.8 – Transmission effective d'un bundle grâce à la couche de convergence (source [21])

chines transmettant leurs paquets via des supports physiques différents : Wi-Fi, Ethernet... (Figure 2.10)
A l'usage, la couche de convergence est confondue avec la couche Bundle.

Tenant compte de ces deux caractéristiques, les applications potentielles sont nombreuses et concernent à la fois les réseaux terrestres classiques, réseaux mobiles, spatiaux… La suite de notre étude se concentrera essentiellement sur les réseaux spatiaux.

Enfin, il est à noter que, seule, l'architecture DTN ne propose aucun mécanisme de fiabilisation bout-en-bout ni de remise en ordre des bundles à l'arrivée. Ceci doit être géré par les couches supérieures. La fiabilisation point-à-point est quant à elle possible par le **custody transfert** décrit plus haut, bien que cela soit optionnel selon la RFC [11] (source [17]).

11. Les **Requests For Comments** (**RFC**), littéralement «demandes de commentaires», sont une série numérotée de documents officiels décrivant les aspects techniques d'Internet, ou de différents matériels informatiques

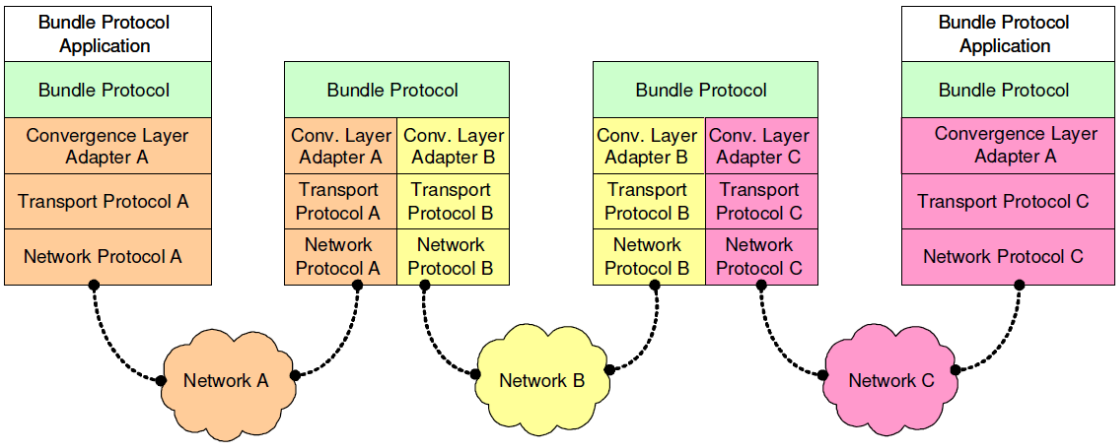


FIGURE 2.9 – Fédération de régions hétérogènes par la couche Bundle (source [11])

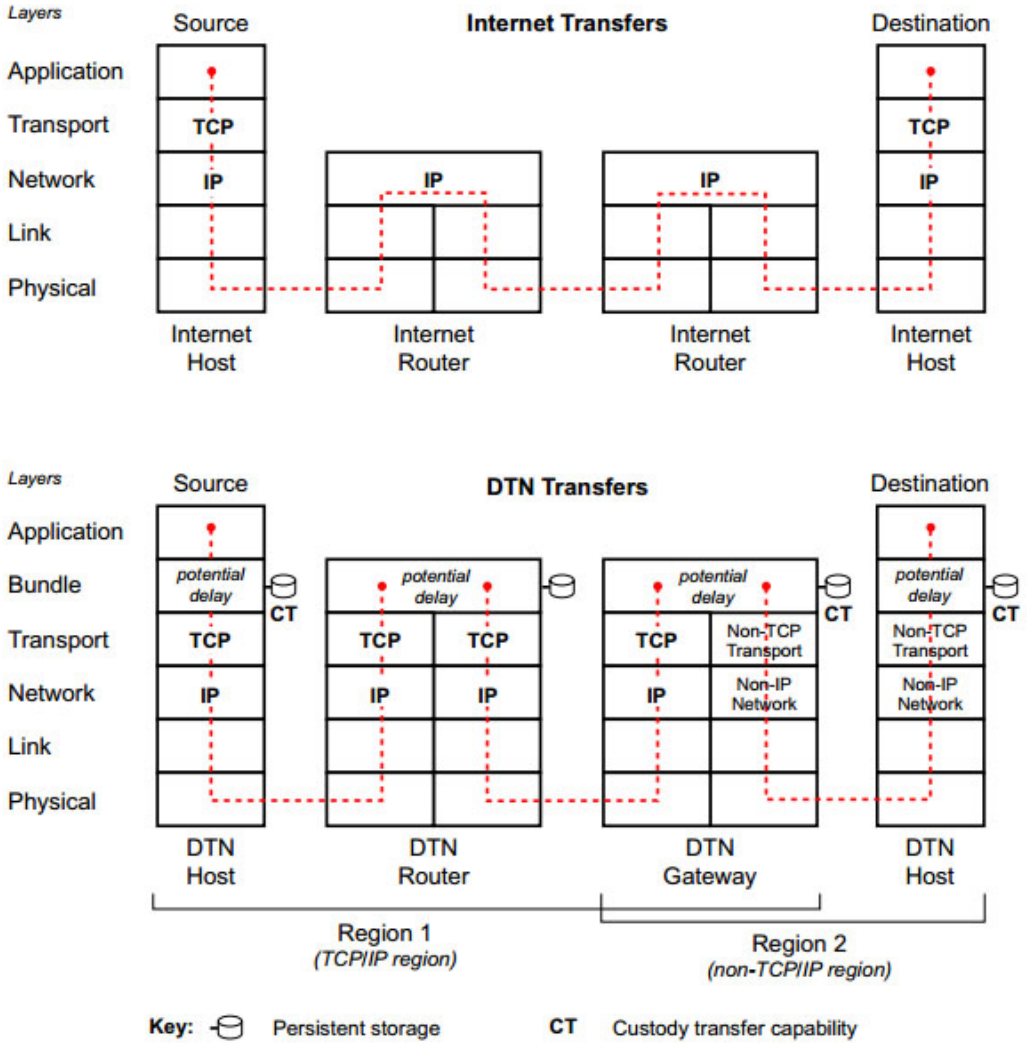


FIGURE 2.10 – Fédération des protocoles de bas niveau par les couches IP et DTN (source [21])

2.2.2 Mise en œuvre des réseaux DTN

La mise en place d'un réseau DTN est effectuée par la définition des nœuds et des liaisons qui leur permettent de communiquer.

Chaque nœud dispose d'un **EID** (**Endpoint IDentifier**) unique ainsi que d'un type parmi trois (source [21] :

- **host** : il s'agit des nœuds terminaux, capables de générer et d'absorber des bundles dans une application ;
- **router** : nœud participant au routage des bundles à l'intérieur d'une région DTN ;
- **gateway** : participe au routage des bundles entre deux régions DTN différentes (même principe qu'une *Border Gateway* entre deux Systèmes Autonomes dans le monde IP).

L'EID reprend le principe des URIs [12] pour construire un identifiant unique du nœud DTN, de la forme `{Region ID ; Regional Node ID}`. Le Region ID est unique pour une région donnée, tandis que le Regional Node ID est unique pour un nœud donné au sein d'une région. Plusieurs nœuds DTN peuvent ainsi posséder le même Regional Node ID tant qu'ils sont situés dans des régions DTN différentes.

L'intérêt de cet adressage est sa résolution dite de *late binding* (liaison tardive). Dans un réseau Internet, l'URI d'un objet est résolue en *early binding* (soit avant le début de la transmission, cf. Figure 2.11), à l'aide d'un serveur DNS [13]. Dans un réseau DTN, la résolution de l'EID est effectuée **au fur et à mesure** que l'on en a besoin (Figure 2.12). Par exemple, le Regional Node ID d'un nœud d'une région distante est résolu par la Gateway DTN à l'entrée de ladite région. Ce phénomène allège le réseau de toute communication concernant la résolution à distance et/ou la mise à jour des EID, ce qui est essentiel au bon fonctionnement de la couche Bundle en environnement difficile.

L'allocation des adresses des nœuds BP doit se faire au travers des organismes SANA (Space Assigned Numbers Authority) et IANA (Internet Assigned Numbers Authority) afin de garantir leur unicité.

Enfin, la notion structurant la définition des liaisons est la notion de **contact**. Un contact correspond à la possibilité pour deux objets réseau de se transmettre de l'information à travers un lien.

On définit trois types de contacts dans les réseaux DTN (source [20]) :

- **permanent** : toujours disponible. Une coupure est tellement rare qu'elle est considérée comme une panne ;
- **à la demande** : semblable au contact permanent, il est toujours disponible à condition de l'activer via une requête ;
- **intermittent** : disponible dans certains intervalles de temps. On distingue alors les contacts :

12. Uniform Resource Identifier, courte chaîne de caractères identifiant une ressource physique ou abstraite sur un réseau
13. Domain Name System

– planifiés : on connaît avec exactitude les moments de contacts. Par exemple, les contacts entre une station sol et un satellite en orbite sont déterminés par orbitographie,
– prévisibles : on tente de prédire les prochains contacts en fonction des précédents,
– opportunistes : impossibles à prévoir.

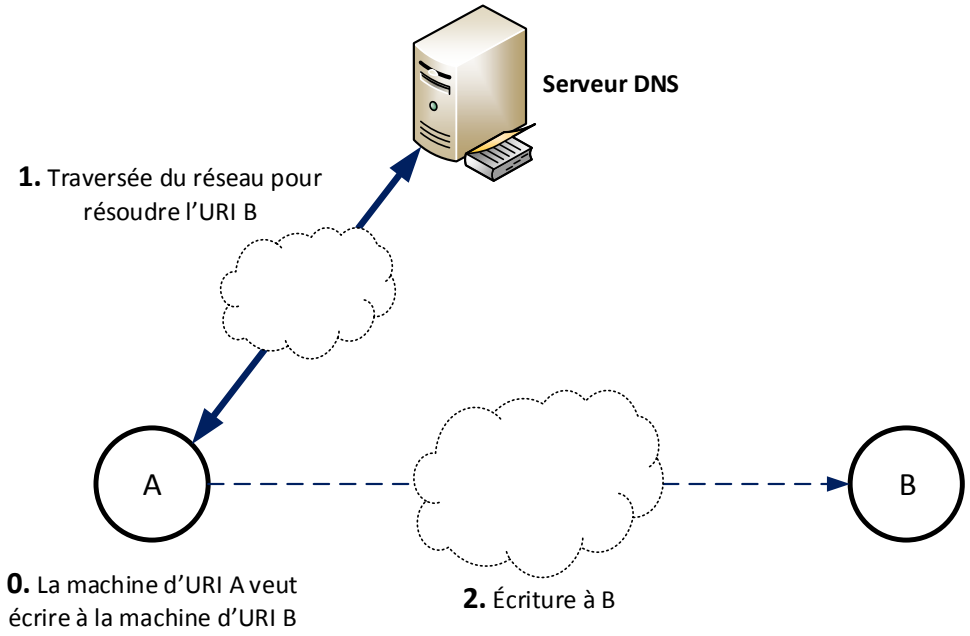


FIGURE 2.11 – Exemple de résolution d'adresse *early binding*

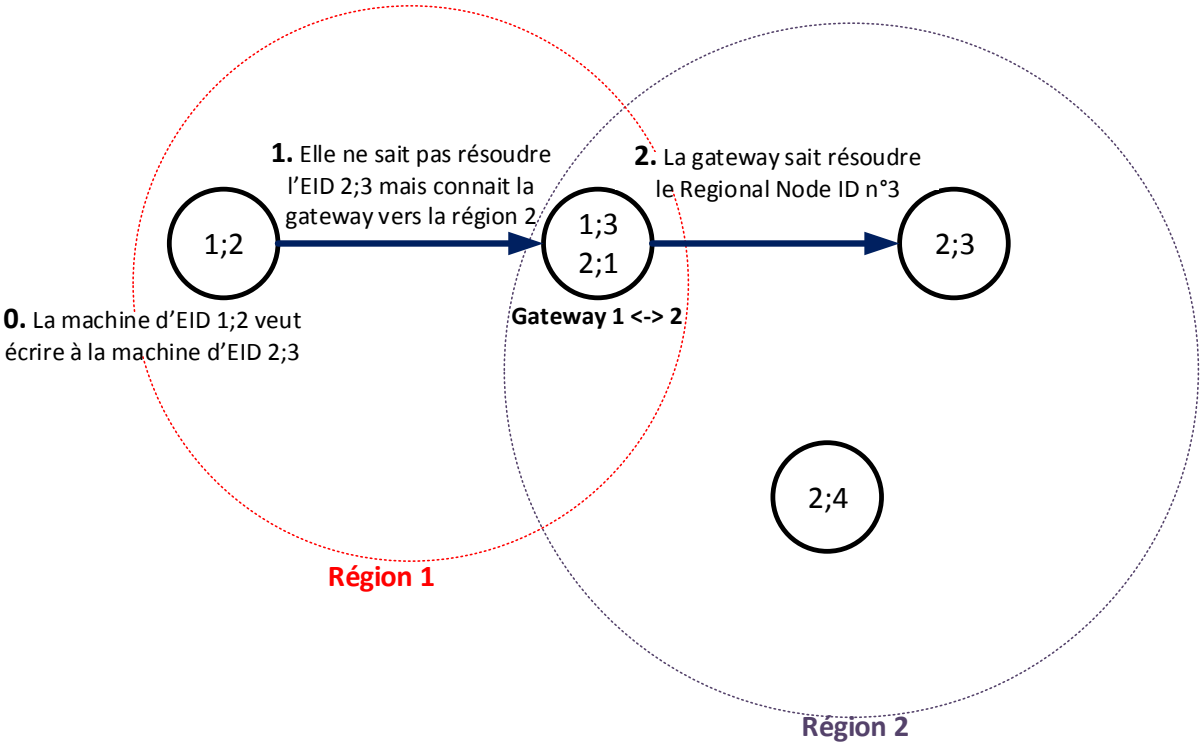


FIGURE 2.12 – Exemple de résolution d'adresse *late binding*

2.2.3 Routage dans le DTN

L'algorithme de routage utilisé pour acheminer l'information à travers un DTN n'est pas spécifié dans la RFC. Les régions DTN d'un même réseau pouvant posséder des propriétés très hétérogènes, il convient à l'implémentation logicielle de sélectionner l'algorithme le plus efficace en fonction de ces propriétés. On distingue cependant deux catégories de routage :

1. Le routage **proactif** : les routes sont calculées a priori et stockées dans des tables. Elles sont mises à jour de façon asynchrone à l'aide de messages spécifiques échangés entre les routeurs (*Link State Advertisement* ou LSA). Ainsi, chaque nœud à une connaissance complète de la topologie du réseau, ce qui lui permet de connaître la meilleure route vers chaque destinataire, à l'aide d'une méthode de calcul du plus court chemin (Dijkstra par exemple). Ce type de routage convient aux réseaux très stables tels l'Internet terrestre (routage OSPF [14] par exemple) mais supporte mal le passage à l'échelle, surtout dans les réseaux mobiles et/ou ad hoc où le changement fréquent de topologie induit une inondation du réseau et la conservation temporaire de routes obsolètes ;
2. Le routage **réactif** : les routes sont calculées au moment où l'information doit être transmise. Ce type de routage gère mieux le caractère dynamique du réseau mais la méconnaissance de certains liens aboutit généralement à emprunter des routes plus coûteuses en moyenne (en termes de bilan de liaison).

14. Open Shortest Path First

Les protocoles de routage les plus utilisés dans les réseaux DTN sont les suivants (sources [12] et [15]).

Le *flood* ou inondation

Le nœud copie l'information et la transmet à l'ensemble des nœuds voisins, et ainsi de suite jusqu'à l'arrivée de l'information. Cette méthode gère parfaitement le caractère dynamique du réseau, avec l'avantage d'être légère en termes de ressources de calcul. En revanche, elle a tendance à surcharger le réseau et générer des boucles, c'est pourquoi elle supporte très mal le passage à l'échelle.

DTLSR (Delay-Tolerant Link State Routing)

Protocole de routage proactif basé sur des LSA, proche des protocoles de routages dans l'Internet. Il a cependant la particularité d'être tolérant aux délais, puisqu'il a été spécifiquement conçu pour fonctionner au sein de DTN terrestres dédiés à l'interconnexion de régions en développement.

PRoPHET (Probabilistic Routing Protocol using History of Encounters and Transitivity)

Comme son nom l'indique, PRoPHET constitue une approche probabilistique du routage réactif. Le calcul d'une route est basé sur l'historique des rencontres et la transitivité. Ces deux paramètres indiquent l'efficacité d'une route sous la forme d'une probabilité de transmettre le message au destinataire ; la meilleure route est ensuite sélectionnée.

On considère que les paires de nœuds se rencontrant fréquemment ont une plus grande chance de transmettre le message. Ceci est ensuite affiné par la prise en compte de la transitivité : si un nœud B rencontre fréquemment deux nœuds A et C non ou peu liés, alors B peut servir de relais pour permettre la transmission d'un message de A vers C (ou inversement).

Contact Graph Routing (CGR)

CGR est un algorithme de routage dynamique initialement dédié aux réseaux spatiaux (source [10]), en cours de standardisation auprès du CCSDS. Il part de l'hypothèse que chaque contact est planifié selon un **graphe de contacts**, connu de l'ensemble des nœuds. La route est calculée au moment de transmettre un message, à l'aide de l'état du graphe de contacts et de la configuration du nœud à l'instant t. Si une route statique est indiquée en entrée de CGR, elle est directement utilisée par le nœud. Enfin, si aucune route ne peut être calculée, il est possible de définir une passerelle par défaut, celle-ci se chargera de transmettre le message sur la bonne route.

CGR est tolérant à la disruption, la connaissance du graphe de contacts lui permettant de définir facilement un nouveau chemin si le premier n'a pas été efficace.

L'utilisation de cet algorithme nécessite de connaître précisément l'évolution des contacts avec le temps, ainsi que leurs caractéristiques (délai, bande passante…). Il est particulièrement adapté aux réseaux spatiaux où l'évolution spatiale de chaque objet est prévisible (par télémesure et/ou orbitographie), avec l'avantage de ne pas augmenter la charge du réseau par des échanges entre les routeurs.

En conclusion, on notera la complexité des méthodes de routage supportant le passage à l'échelle d'un réseau DTN, relativement aux méthodes typiques de l'Internet terrestre.

2.2.4 Licklider Transmission Protocol

Licklider Transmission Protocol (LTP) est un protocole de transport point-à-point, imaginé pour la transmission d'un message en conditions difficiles, par exemple une communication spatiale (source [18]). UDP ne permet pas de transport fiable, TCP demande des échanges initiaux et finaux trop coûteux – surtout sur un lien à fort délai – et ne supporte pas les pertes élevées, c'est pourquoi il était nécessaire de concevoir un nouveau mécanisme de transport pour les réseaux difficiles.

LTP s'intercale entre la couche Bundle – plus précisément la couche de convergence – et les couches réseau sous-jacentes, sous la forme d'un **moteur LTP**. Chaque nœud dispose de son moteur LTP :

- à l'émission, le moteur LTP agrège les bundles en sortie sous la forme d'un **bloc LTP** (optimisation de l'utilisation du lien). Une fois le bloc constitué, celui-ci est découpé en segments (Figure 2.13) qui sont envoyés par le réseau vers un port défini ;
- à la réception, le moteur LTP reçoit l'ensemble des segments sur le port défini, reforme le bloc LTP puis le transmet à la couche supérieure.

Un bloc LTP est constitué de deux parties (Figure 2.13) :

- la partie **rouge** : données dont la livraison est garantie par un système d'acquittements (fonctionnement similaire au protocole TCP) ;
- la partie **verte** : transmission non fiable de type *best-effort*, les segments perdus ne sont pas retransmis (fonctionnement similaire au protocole UDP).

La taille de chaque partie est variable, elle peut être nulle.

Ainsi, après l'envoi d'un segment rouge, le moteur LTP se positionne en attente d'un acquittement de la réception. En cas de perte, le moteur entre en pause le temps d'un *timer* [15] au lieu de stopper la transmission, puis reprend celle-ci dès que cela est possible. Ce fonctionnement le rend très tolérant aux conditions difficiles (pertes notamment). De plus, LTP est par nature tolérant aux délais (timers potentiellement très larges).

La fin de la partie rouge est indiquée par un segment *End Of Red Part* (EORP). La transmission du bloc est finalisée par un mécanisme de type « 3-Way Handshake » [16] (Figure 2.14) :

15. Minuteur
16. Selon le protocole de communication TCP, une connexion entre deux hôtes s'établit en trois étapes : c'est le « three-way handshake »

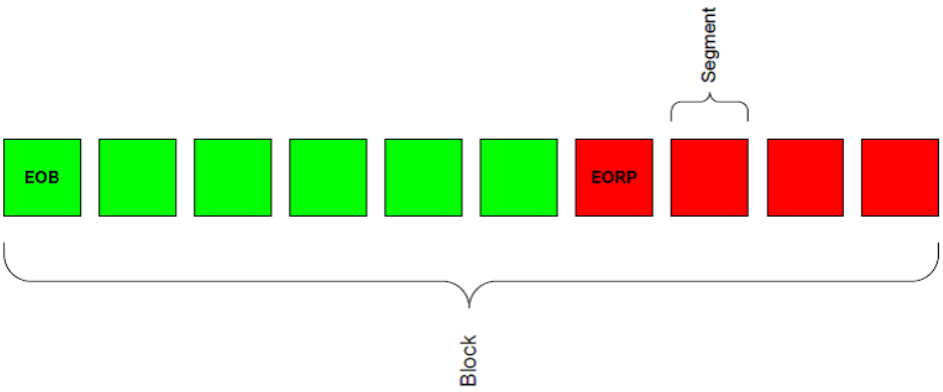


FIGURE 2.13 – Segmentation d'un bloc LTP

1. End Of Red Part (EORP) : indique la fin de la partie rouge ;
2. Report Segment : indique ce qui a été correctement reçu ;
3. Report Acknowledgment : finalise la transmission si le Report Segment a indiqué que l'ensemble de la partie rouge a été reçue.

La fin du bloc (correspondant également à la fin de la partie verte) est, quant à elle, indiquée par un segment *End of Block* (Figure 2.14).

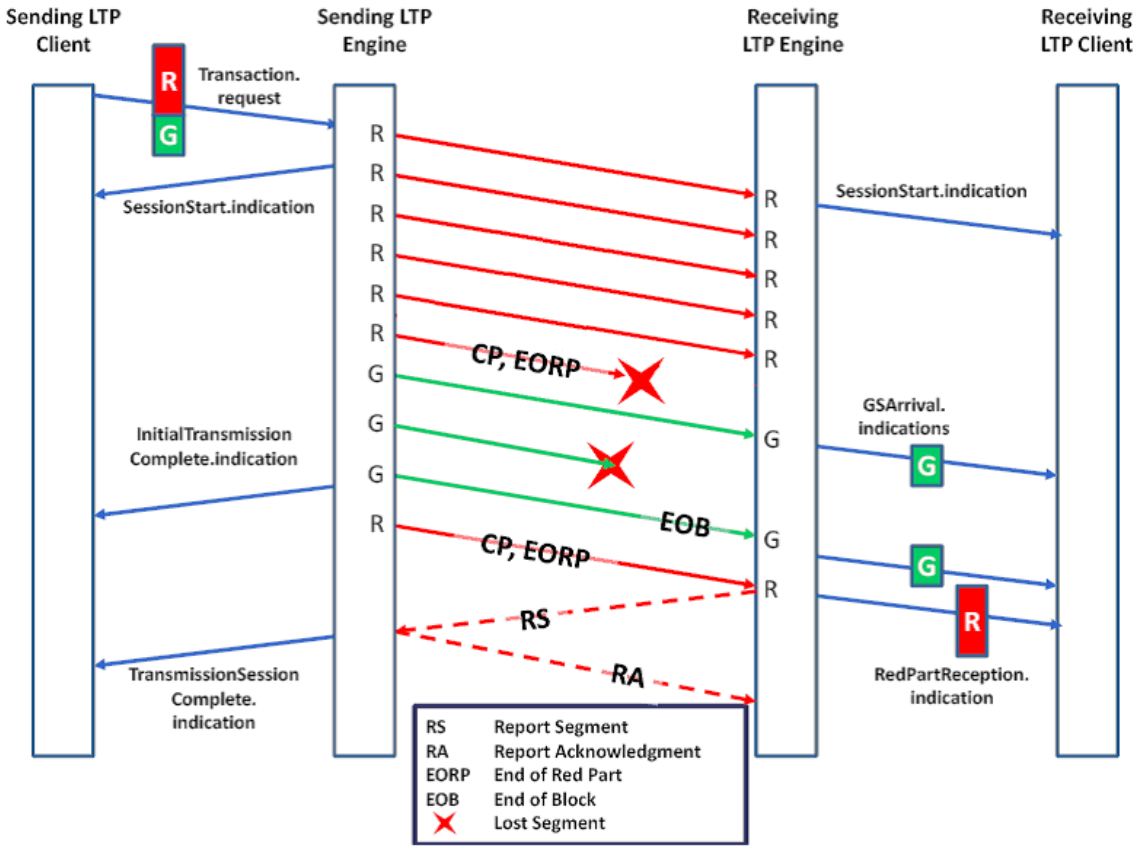


FIGURE 2.14 – Illustration du protocole LTP (source [9])

Chaque moteur LTP possède un certain nombre de **sessions**. Une session correspond à la possibilité d'envoyer un bloc à travers le réseau. Un contrôle de flux est ainsi réalisé par le paramétrage du nombre de sessions ouvertes autorisées en simultané. Chaque entête

de segment contient le numéro de la session dont il est originaire, le récepteur est alors capable de réassembler correctement les blocs, malgré l'existence de plusieurs sessions : il s'agit d'une forme de multiplexage [17] temporel.

On notera que LTP permet uniquement un transport point-à-point du message, le protocole est donc souvent considéré comme un protocole de niveau 2 (liaison de données). Une extension, LTP-Transport (LTP-T), autorise le transport bout-en-bout de l'information, mais étant donné son faible intérêt pour les réseaux DTN, où la gestion du transport bout-en-bout à un niveau supérieur à Bundle apparaît plus pertinente, nous ne la détaillerons pas davantage.

2.2.5 Fragmentation

Bundle Protocol propose un mécanisme de fragmentation afin d'optimiser l'exploitation des contacts actifs. Il propose deux catégories de fragmentation :

1. **Proactive** : la fragmentation est effectuée *a priori*, lorsque la durée de contact restante ne permet pas l'envoi d'un bundle entier. Ce dernier est alors fragmenté en deux (ou plusieurs) bundles de taille plus petite, à chacun desquels un entête est apposé. Cela permet de profiter pleinement des disponibilités des contacts intermittents, à condition que ceux-ci soient planifiés. Le réassemblage est effectué à l'aide de paramètres *timestamp* et *offset* situés dans les entêtes Bundle ;
2. **Réactive** : utilisée en cas de rupture non prévisible de contact, *a posteriori*. Grâce à l'entête Bundle, le récepteur a conscience de n'avoir reçu qu'une partie du bundle. Il transmet alors ce fragment de bundle au nœud suivant (en indiquant qu'il s'agit d'un fragment) et informe l'émetteur originel de cette situation. Ce dernier pourra alors envoyer le fragment manquant lors du prochain contact, sans avoir à retransmettre l'ensemble du bundle. Cette solution semble idéale au sein d'un réseau instable, mais son implémentation sur cas réel souffre de deux problèmes :
   - elle induit un overhead [18] conséquent ;
   - elle autorise un nœud intermédiaire à modifier l'entête du bundle (pour indiquer qu'il s'agit d'un fragment), ce qui est peu acceptable pour des raisons de sécurité : l'intégrité n'est plus garantie.

2.2.6 Implémentation du protocole

Un bundle est composé d'un *Primary Block* (l'en-tête Bundle) et d'un *Payload Block*, littéralement la charge utile, c'est-à-dire l'unité de données sous-jacente. Le Primary Block reprend des champs équivalents à ceux d'IP : version, source, destination... Il ajoute cependant de nouveaux champs spécifiques au protocole Bundle (cf. Annexe A).

Il est à noter que de nombreux champs sont codés en **SDNV** (Self Delimiting Numeric Value), ce qui leur confère une taille variable donc mieux optimisée : le MSB (Most

17. Technique consistant à faire passer plusieurs informations à travers un seul support de transmission
18. Combinaison d'excès de temps de calcul, mémoire, bande passante, ou autre ressource requis pour atteindre un objectif donné

Significant Bit ou bit de poids fort) du dernier octet est positionné à 0, alors que celui des octets précédents est positionné à 1. De même, les champs indiquant un nœud DTN (source, destination, etc.) sont en réalité des pointeurs vers une entrée de dictionnaire. Le dictionnaire est un tableau contenu dans l'en-tête ; ce mécanisme permet de réduire la taille moyenne de l'en-tête en évitant de renseigner plusieurs fois la même valeur dans des champs distincts.

Enfin, il existe des extensions d'en-tête, pour la sécurité par exemple.

2.2.7 État de l'art

Actuellement, les DTNs spatiaux s'inscrivent dans le cadre des objectifs du CCSDS. L'un d'entre eux est la standardisation de l'architecture DTN adaptée aux contraintes du spatial (*Near Earth*[19] et *Deep Space*). Dans ce contexte, la NASA est la principale force de proposition, mais d'autres agences internationales s'y intéressent de près au travers d'expérimentations.

A titre d'illustration, la première expérimentation connue d'un DTN spatial a eue lieu avec la sonde Deep Impact en octobre 2008 (Deep Impact Networking ou DINET)(source [3]). En octobre 2012, le commandant de l'ISS[20] Sunita Williams a pu manipuler à distance un petit robot de type « Lego Mindstorms » situé au centre européen d'opérations spatiales en Allemagne, au cours d'une expérience exploitant le protocole DTN (source [3]). D'autres expérimentations similaires ont été menées par la NASA, la JAXA (agence spatiale japonaise), ainsi que d'autres agences spatiales.

En termes d'implémentations logicielles, il en existe aujourd'hui quelques-unes mais les principales sont les deux suivantes (source [1]) :

- **DTN2** : développé par le DTNRG, cette implémentation en C++ est dédiée à l'expérimentation et la validation du protocole, ainsi qu'à l'apprentissage. Elle est essentiellement destinée aux réseaux terrestres bien qu'elle comporte depuis peu une implémentation de LTP. Plusieurs modes de routage sont disponibles. Sa nature pédagogique rend son utilisation simple d'accès et bien documentée. DTN2 reste toutefois limité dans une optique d'utilisation opérationnelle ;
- **ION** : développée en C par le Jet Propulsion Laboratory (NASA), **Interplanetary Overlay Network** est l'implémentation de Bundle Protocol la plus aboutie dans un contexte de communication spatiale. Elle est construite pour les réseaux spatiaux et sa configuration est orientée en conséquence (routage CGR par exemple). De plus, elle possède une implémentation de LTP et de CFDP.

Il n'y a, à ce jour, aucune utilisation opérationnelle connue des DTNs, seulement des expérimentations.

19. Objets situés dans le voisinage de la Terre
20. Station Spatiale Internationale (International Space Station)

2.3 Objectifs du stage

Dans sa mission de Recherche et Technologie (R&T), le CNES souhaite participer à l'expérimentation d'une communication DTN entre un centre de contrôle CNES et la Station Spatiale Internationale (ISS), dont la partie bord serait gérée par la NASA. L'objectif à terme est de réaliser un système démontrant la possibilité pour le CNES de faire transiter des communications DTN spatiales sur son réseau de stations. Il n'est en effet pas prévu à court terme d'installer une pile DTN sur les stations CNES, mais ce dernier souhaite conserver son potentiel d'interopérabilité des moyens sol vis-à-vis des autres agences spatiales, y compris dans le cas de communications DTN mais sans modification de l'existant.

Dans un premier temps, on cherche à réaliser l'expérimentation illustrée par la Figure 2.15 (les protocoles inconnus seront expliqués au fur et à mesure).

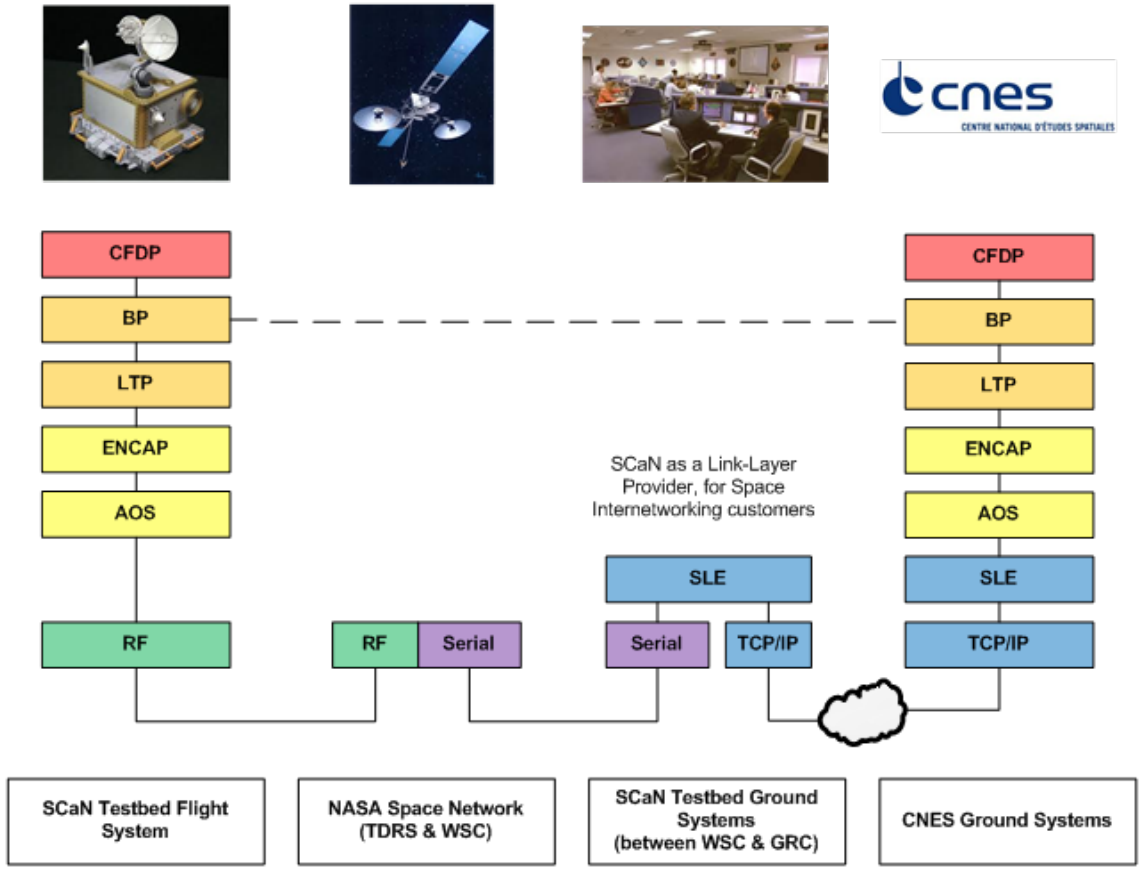


FIGURE 2.15 – Démonstrateur DTN bout-en-bout en collaboration avec la NASA

Le stage s'inscrit dans le contexte du développement de la partie sol de cette expérimentation. Il consiste à l'installation des piles protocolaires ION, leur validation, puis la conception d'une solution permettant leur utilisation dans un contexte réel (sur protocoles spatiaux). Enfin, l'intégration de l'ensemble sur la plate-forme de simulation TM/TC permettra de valider la partie sol de l'expérimentation future.

Une attention supplémentaire est portée sur la modélisation des éléments bord, car mon travail doit permettre au CNES de simuler la partie « NASA » de l'expérimentation en toute autonomie, afin de tester sa ou ses solution(s) sol.

Chapitre 3

Déroulement du stage

3.1 Organisation

Afin de répondre aux objectifs du stage, il a été décidé de l'organiser selon la liste de tâches suivantes :

1. État de l'art des DTNs, appréhension du contexte et découverte du laboratoire TM/TC ;
2. Expérimentation de DTN2 pour se familiariser avec ce type de piles protocolaires ;
3. Installation et expérimentation d'ION ;
4. Si cela est possible, modification du code source d'ION afin de le faire fonctionner au-dessus des piles protocolaires spatiales ENCAP et AOS (voir Figure 2.15) ;
5. Installation sur la plate-forme laboratoire de simulation puis validation de la solution technique réalisée.

En termes de livrables, et par l'intermédiaire du présent rapport, le stage doit aboutir à la fourniture :

- d'une synthèse concernant l'installation et l'exploitation des implémentations logicielles utilisées ;
- d'une solution technique pour l'intégration de DTN sur les couches protocolaires spatiales ;
- d'un retour d'expérience sur l'évolution des systèmes existants et l'impact de la technologie étudiée.

Enfin, le stage a été réalisé en autonomie, tout en étant supporté à trois niveaux :

- bien entendu, par l'encadrement régulier par mon maître de stage ;
- par des échanges avec d'autres services CNES travaillant ou ayant travaillé sur DTN (notamment, le service « Applications et Réseaux par Satellite ») ;
- par des échanges avec des acteurs externes (voir plus loin).

Les parties suivantes, organisées par objectifs et, par défaut, dans l'ordre chronologique, s'intéressent au détail du travail réalisé.

3.2 État de l'art

La première étape du stage a consisté en la réalisation d'un état de l'art sur les DTNs : lectures de rapports, de standards, documentation sur le web. Le descriptif de l'architecture DTN présenté en partie 2.2 résulte essentiellement de la synthèse de ces documents. De plus, des échanges avec mon maître de stage m'ont permis de mieux appréhender le contexte opérationnel des communications spatiales (cf. partie 2.1).

Cette étape était plus complexe que prévu, à cause du caractère très expérimental et ciblé (lié à des infrastructures réseaux « atypiques ») des DTNs. Heureusement, l'existence de rapports de stages précédents ([12] et [15]) fut une amorce appréciable à ce travail. De plus, ma présence à des réunions de travail sur DTN, organisées par un autre service du CNES, fut un moyen d'échange enrichissant sur cette technologie.

3.3 Mise en place de l'environnement de travail

3.3.1 Problématique

En parallèle de l'état de l'art, j'ai pris en main la machine sur laquelle je devais réaliser le début de mon travail technique. Il s'agissait d'une machine virtuelle [1] Linux, disponible sur un serveur distant administré par un tiers.

A l'utilisation rapide du système, il s'est avéré que celui-ci ne satisfaisait pas deux contraintes majeures :

1. Nous souhaitions utiliser des machines virtuelles pour l'expérimentation réseau, il était donc indispensable de pouvoir en créer, dupliquer et supprimer en toute autonomie ;
2. L'absence de connexion à Internet rendait l'installation de certains paquets impossible.

J'ai alors formulé la proposition d'installer un serveur de machines virtuelles au sein du laboratoire, dédié à l'expérimentation réseau ainsi qu'à la recette de logiciels (on peut facilement installer un OS [2] cible sur une VM). L'intérêt éprouvé pour cette manipulation me pousse à y consacrer une partie.

1. Illusion d'un appareil informatique créée par un logiciel d'émulation (anglais *virtual machine*, abr. **VM**)
2. *Operating System* ou Système d'Exploitation

3.3.2 Mise en place

Avant toute chose, nous avons identifié les besoins (cf. ci-dessus). Pour des raisons de souplesse, il a été décidé d'utiliser un hyperviseur [3] de niveau 2, c'est-à-dire installé par-dessus un OS « physique ». Le dernier critère était la gratuité du système, pour une installation facilitée de l'ensemble.

Le choix de l'hyperviseur était important car celui-ci allait conditionner l'ensemble de notre potentiel en termes d'expérimentations réseaux. Deux d'entre eux ont retenu notre attention, par rapport aux critères ci-dessus :

- **Virtualbox** : publié par Oracle, ce logiciel libre de virtualisation est l'un des plus connus. Il est facile d'utilisation, stable. En revanche, il est moyennement rapide car il émule le processeur, et ses capacités de mise en réseau virtuel des VMs sont limitées (uniquement création d'un réseau local virtuel).
- **Virtualbricks** : logiciel libre réalisé par l'Université de Bologne. Il a à l'origine été spécifiquement créé pour expérimenter DTN sur des machines virtuelles (source [19]). Il réutilise les principes d'un hyperviseur stable - en l'occurrence, QEMU/KVM - en le couplant avec un logiciel de création de réseaux virtuels (VDE, Virtual Distributed Ethernet)(cf. Figure 3.1). Ses atouts sont :
  - outils d'expérimentation réseau : on peut créer la topologie réseau souhaitée entre les VMs, paramétrer les liaisons grâce au module «**NetEmu**» (délais, pertes...) ;
  - performance en mode KVM, qui virtualise directement le processeur (technologie Intel VT, source [4]).

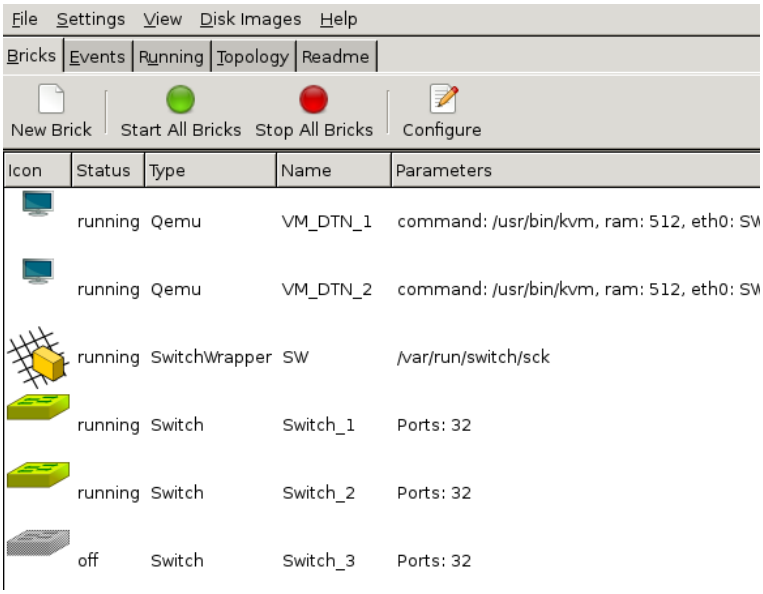


FIGURE 3.1 – Capture d'écran de Virtualbricks

Naturellement, Virtualbricks semblait adapté à nos besoins mais, à la première utilisation, a montré de forts signes d'instabilité. Heureusement, un contact direct avec les développeurs du logiciel ont permis de résoudre les problèmes, nous offrant la possibilité de pro-

3. Plate-forme de virtualisation

fiter du potentiel de ce logiciel. En attendant, Virtualbox a été utilisé pour les premières installations, les deux hyperviseurs sont donc disponibles sur la machine « serveur ».

Les machines virtuelles fonctionnent sous Debian 7.4 pour Virtualbricks, Ubuntu 14.04 pour Virtualbox. Elles sont reliées au réseau par l'intermédiaire de l'hyperviseur et accessibles depuis un poste bureautique par un client SSH [4]. La Figure 3.2 illustre l'environnement de travail mis en place.

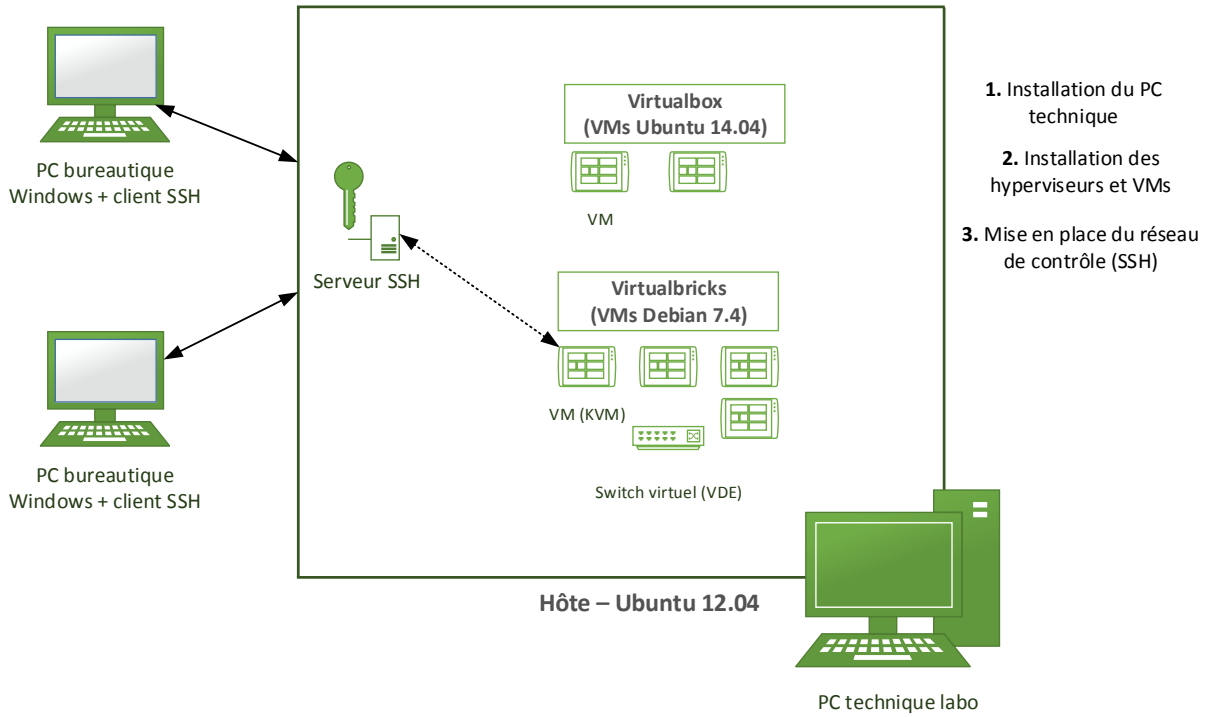


FIGURE 3.2 – Mise en place de l'environnement de travail

Pour des raisons de clarté vis-à-vis de la problématique du stage, les informations plus détaillées sur l'installation de l'ensemble et l'accès aux VMs à distance sont disponibles en Annexe B.

3.4 Exploitation de DTN2

Avant de débuter la phase de codage pour répondre à la problématique du projet, il fallait que je me familiarise avec l'implémentation ION de DTN. Dans un premier temps, l'expérimentation de DTN2 était un bon moyen de prendre en main ce type de logiciel, avec une courbe de démarrage plus lisse.

La méthodologie utilisée fut la suivante (voir aussi la Figure 3.3) :

1. Utiliser progressivement DTN2 jusqu'à mettre en évidence les mécanismes génériques de DTN ;

4. *Secure SHell* (SSH) est à la fois un programme informatique et un protocole de communication, qui permettent d'accéder à une machine distante de façon sécurisée

2. Utiliser progressivement ION jusqu'à maîtriser son utilisation de base et mettre en évidence les mécanismes génériques de DTN ;
3. Expérimenter ION sur des scénarii-clés, relatifs à des scénarii réels, afin de valider leur faisabilité.

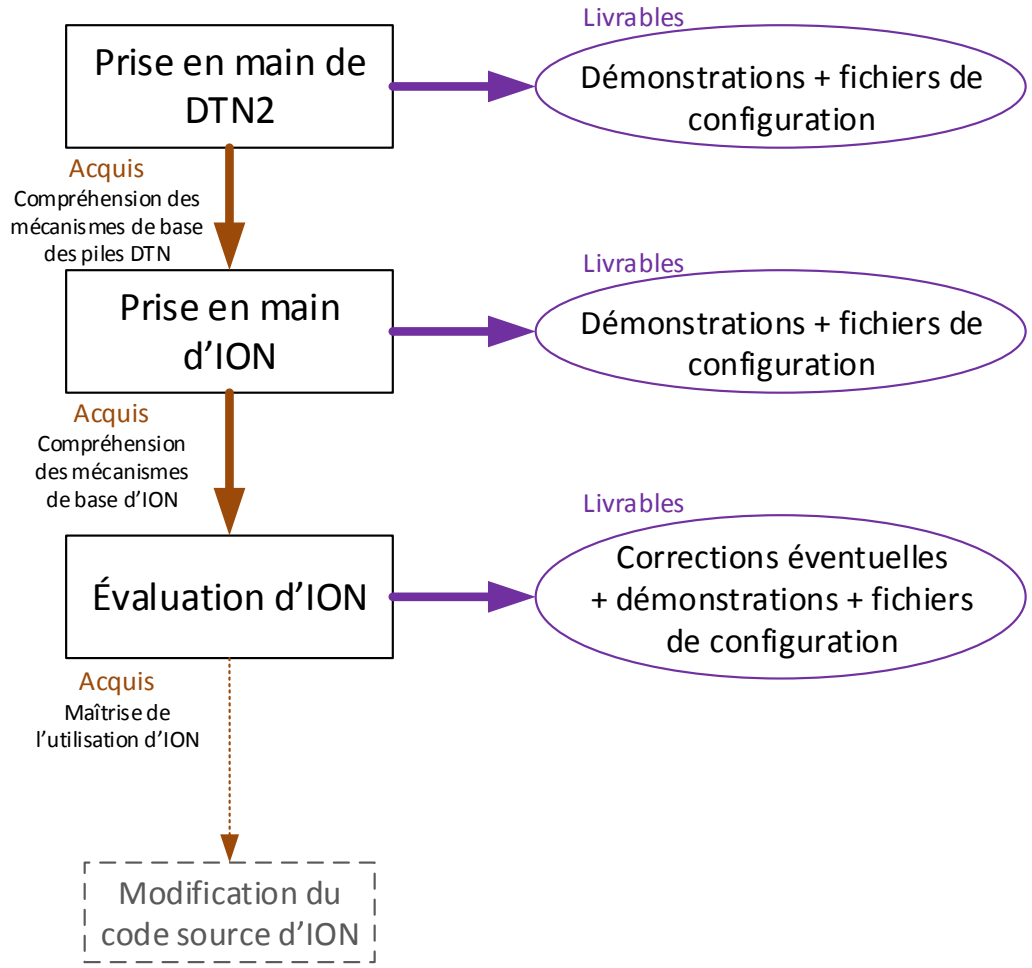


FIGURE 3.3 – Méthodologie utilisée pour la prise en main des implémentations logicielles

3.4.1 Présentation

DTN2, défini par l'organisme de recherche IRTF DTNRG, est le nom de l'implémentation de référence de l'architecture DTN. Simple et robuste, il met en œuvre cette dernière de façon générique. Il est donc utilisable dans un contexte d'exploitation mais peu adapté aux réseaux difficiles tels que les réseaux spatiaux. Il reste un bon moyen de débuter avec la mise en œuvre d'un réseau DTN. La suite de ce rapport s'attache à présenter le mode de fonctionnement de ce logiciel ainsi que l'expérimentation réalisée (version 2.9.0 de DTN2). Son installation, aisée, est détaillée par la documentation.

DTN2 fonctionne comme un démon, c'est-à-dire qu'il s'exécute en arrière-plan plutôt que

sous le contrôle direct de l'utilisateur. Codé en C++, il s'appuie sur le *framework* [5] Oasys, qui fournit une interface entre le code de DTN2 et le système d'exploitation sous-jacent. Enfin, DTN2 est paramétré via un fichier de configuration TCL [6] et propose les fonctionnalités suivantes :

- une **API** [7] permettant de concevoir des applications utilisant le DTN (C, Perl, Python et TCL). Elle permet, basiquement, d'envoyer un paquet de données applicatives dans un bundle DTN2 ;
- utilisation de **stockage persistant** pour le « store-and-forward » des bundles. Par défaut, Berkeley Database est utilisé, mais il possible de spécifier un stockage dans un fichier, en mémoire vive, ou encore dans une base de données SQL (SQLite, MySQL) ;
- **protocoles de routage** nombreux : DTN2 supporte à la fois des algorithmes de routage internes et externes. Les premiers visent à faciliter la gestion d'un système autonome en automatisant au maximum la construction des tables de routage, les seconds peuvent être vus comme des interfaces normalisées d'échange d'informations entre deux systèmes autonomes. Les principaux protocoles supportés sont :
  - routage statique,
  - flood,
  - PRoPHET,
  - DTLSR,
  - TCS routing (pour les Tetherless Computing Architecture, permettant notamment d'interconnecter des régions privées de l'Internet à l'aide de « data mules », voir source [13]),
  - RAPID (Resource Allocation Protocol for Intentional DTN) : routage externe,
  - HBSD (History Based Scheduling and Drop) : routage externe.
- **couches de convergence** incluant TCPCL (TCP Convergence Layer), LTPCL, Bluetooth CL ;
- possibilité de créer des **tunnels DTN2**.

3.4.2 Mise en œuvre

L'objectif de cette étape était d'appréhender le principe de mise en œuvre de DTN2 : un fichier de configuration et un démon. En effet, nous verrons plus tard qu'ION utilise la même mécanique, dans un registre plus complexe en termes de configuration.

La Figure 3.4 schématise le cycle d'utilisation de DTN2.

5. Ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou partie d'un logiciel

6. Tool Command Language

7. Une **interface de programmation** (anglais *Application Programming Interface*, abr. **API**) est un ensemble normalisé de classes, méthodes ou fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels

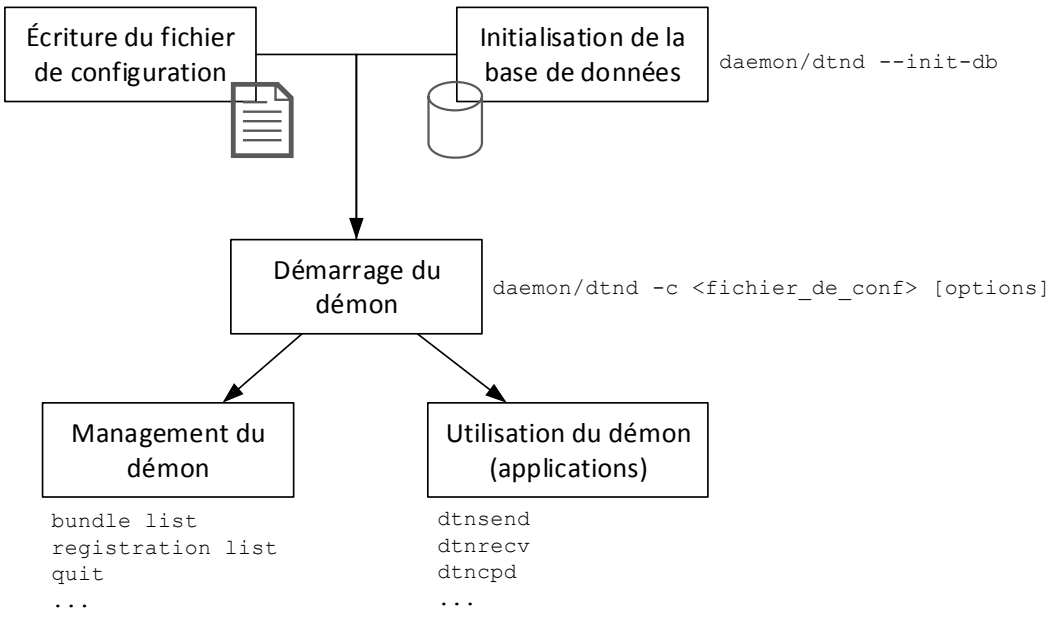


FIGURE 3.4 – Cycle d'utilisation de DTN2 pour une machine donnée

L'Annexe C illustre la structure par défaut du fichier de configuration, ainsi que les principales commandes utilisées (pour des informations plus détaillées, se référer à la documentation officielle [2]).

Dans DTN2, les nœuds sont identifiés par des EID/URI de la forme `dtn://<host>.dtn/service`. La définition de ces différentes URI permet donc également d'effectuer du filtrage de trafic en ouvrant le nœud pour un service ou non. Notons la présence du service « * », il sert de service par défaut pour la réception de bundles dont le nœud courant n'est pas le destinataire final.

Ainsi, notre étude consiste à faire dialoguer deux machines à travers une machine intermédiaire. Après avoir pris en main le logiciel pour une communication locale à une machine puis entre deux machines, l'étude illustrée par la Figure 3.5 a été mis en place. Virtualbricks est utilisé pour modéliser le réseau sous-jacent. Les VMs sont des clones qui ne diffèrent que par leur fichier de configuration. Enfin, le routage utilisé est un routage statique.

Les tests sont effectués avec à l'aide des applications suivantes :

`dtnsend [opts] -s <local_eid> -d <dest_eid> -t <type> -p <payload>`
Permet d'envoyer le contenu d'un fichier (`-t f`) ou un message (`-t m`) par Bundle Protocol au nœud distant d'EID dest_eid (attention de bien préciser les services). Les options sont nombreuses (pour les lister, taper `dtnsend -h`).

`dtnrecv [opts] <local_eid>`
Permet de se mettre en écoute des bundles envoyés par l'émetteur à l'aide de dtnsend, entrant sur l'EID <local_eid>, puis affiche leur contenu. Même remarque que ci-dessus concernant les options.

`dtncp [-A api IP address] [-B api port] [-D] [-expiration sec] <filename> <destination_eid> <remote-name>`
Permet d'envoyer un fichier par Bundle Protocol.

`dtncpd [-A api_IP_address] [-B api_port] [ directory ]`
Permet de se mettre en écoute des fichiers envoyés par dtncp, éventuellement de spécifier leur dossier de destination.

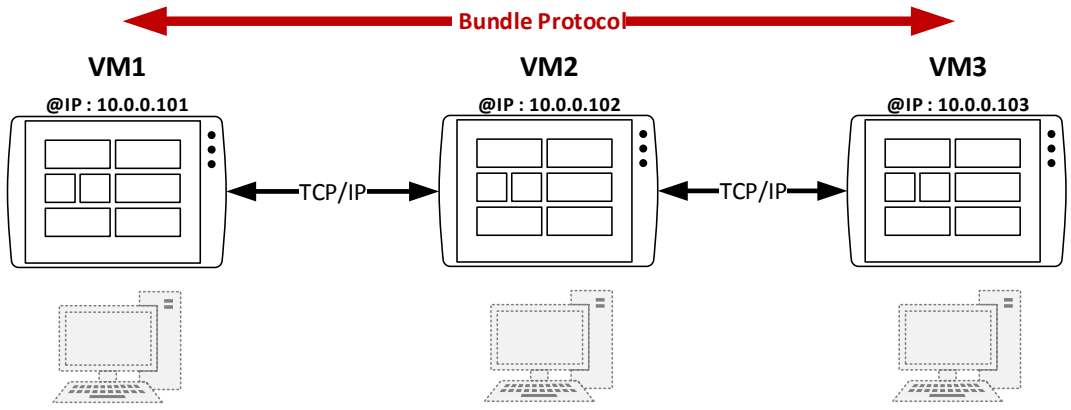


FIGURE 3.5 – Scénario d'étude de DTN2

En résumé, on remarque que la transmission des bundles est effectivement gérée de façon transparente par le démon. Une fois qu'un bundle est disponible sur le nœud destinataire final, il est possible de le consommer avec une application, mais cela n'est pas nécessaire à sa réception dans le dossier regroupant les bundles. On remarque également que la transmission est résistante aux déconnexions, puisque l'on peut observer les bundles en transit sur le nœud 2 lorsque le nœud 3 n'est pas présent sur le réseau.

Quelques notes supplémentaires :

- le **custody transfer** fiable (avec acquittement) n'est pas implémenté d'après la documentation (source [2]) ;
- pour demander un acquittement bout-en-bout avec dtnsend, il faut utiliser l'option `-D` et non `-e` comme indiqué dans la documentation ;
- la documentation décrit la commande `dtndcpd` alors qu'il s'agit de la commande `dtncpd` (cf. ci-dessus). De plus, le démon plante si cette commande est la première à être lancée par le serveur, car son EID n'est pas encore défini dans le registre et qu'il n'est pas possible de le spécifier en argument de cette commande.

Enfin, nous avons noté que la documentation de DTN2 était très riche, ce qui constitue un bon point pour celui qui veut rapidement mettre en place un réseau DTN «classique».

3.5 Exploitation d'ION

3.5.1 Présentation

ION est l'implémentation de DTN la plus complète dans un contexte de communication spatiale. Ses caractéristiques sont les suivantes :

- **couches de convergence** : LTPCL, TCPCL, UDPCL, DCCPCL [8] ;
- couches supérieures (non obligatoires) : **CFDP** (cf. partie 2.1.2), AMS [9], DTPC (cf. partie 3.9), DCCP, BSS [10] ;
- routage : uniquement **CGR** (cf. partie 2.2.3) ;
- de même que DTN2, quelques applications de base ainsi qu'une **API**.

De plus, ION utilise l'adressage **IPN** (InterPlanetary Network) pour identifier les nœuds DTN. L'EID IPN est de la forme `ipn:<node_id>.<service_id>`, où les IDs sont des nombres strictement positifs. Typiquement, le `<node_id>` d'un satellite correspond à son Spacecraft ID assigné par SANA. L'intérêt de cet adressage est avant tout de travailler sur des identifiants beaucoup plus courts, ce qui permet de réduire la taille des entêtes Bundle afin qu'ils soient mieux adaptés aux contraintes spatiales (notamment, ressources limitées à bord). Il existe cependant un module d'ION permettant l'adressage DTN2 afin d'interconnecter les deux systèmes, mais aucun mécanisme de tunnel.

On constate ainsi qu'ION est bien dédié au spatial, notamment par son protocole de routage exploitant uniquement des contacts planifiés.

De façon analogue à DTN2, le fonctionnement d'ION s'appuie sur plusieurs fichiers de configuration, exploités par un démon. Les fichiers de configuration synthétisent des commandes qu'il est possible de fournir directement à un module de configuration. Les fichiers de configuration disponibles en Annexe D peuvent faire office d'exemple.

Les principaux modules utilisés sont les suivants :

- **ionadmin** : configuration du « Node ID » ainsi que du graphe des contacts (ainsi que les bandes passantes et délais associés) ;
- **ltpadmin** : configuration du moteur LTP. Permet d'ajouter des *spans* (sessions) entre deux machines. Il s'agit de l'entité fondamentale définissant la possibilité pour deux moteurs LTP distants de dialoguer entre eux. Cette configuration est essentielle puisqu'elle va induire le potentiel du moteur LTP en termes de contrôle de flux. De plus, on définit dans cette commande l'utilisation par LTP des protocoles de plus bas niveau, à l'aide d'un programme appelé **Link Service** ;
- **bpadmin** : configuration de la couche Bundle (interfaces réseaux et protocoles associés) ;
- **cfdpadmin** : configuration du démon CFDP ;
- **ipnadmin** : configuration du routage statique.

8. Datagram Congestion Control Protocol Convergence Layer
9. Asynchronous Messaging Service
10. Bundle Streaming Service

Les fichiers de configuration correspondant à chaque module sont de type `<nom>rc` (par exemple, ltprc, bprc, etc.). En cas d'interrogation, ne pas hésiter à se référer au manuel de chaque module et/ou fichier de configuration, car ceux-ci sont bien détaillés.

A noter qu'il est possible de composer l'ensemble dans un seul fichier de configuration, soit directement (voir exemples en Annexe D), soit par l'intermédiaire de la commande suivante :

```
ionscript -i host#.ionrc -p host#.ipnrc -l host#.ltprc -b host#.bprc -O host#.rc
```
Compile les fichiers de configuration ionrc, ipnrc, ltprc et bprc de l'host# dans un fichier unique host#.rc

Le processus inverse (décomposition) est effectué de cette façon :

```
ionscript -i host#.ionrc -p host#inrc -l host#ltprc -b host#bprc -I host#rc
```

Pour terminer, il ne reste plus qu'à démarrer le démon. Si l'on utilise qu'un seul fichier de configuration global, ce qui est clairement le plus pratique, on exécute la commande suivante :

```
ionstart -I <nom_fichier_conf>.rc
```

Une fois cela fait, les applications peuvent être utilisées. Le démon peut ensuite être stoppé par la commande `ionstop` ou la commande `killm`. Pour des informations sur l'une de ces commandes, taper `<nom_commande> --help`

3.5.2 Mise en place de l'étude

Deux précédents stages effectués au CNES (sources [12] & [15]) s'étaient intéressés à l'évaluation d'ION selon un scénario avancé. Ce scénario s'inscrit dans le cadre d'une approche d'hybridation satellite/terrestre pour la télédétection. Un satellite effectue des mesures puis les transmet à un centre de contrôle (Figure 3.6).

Dans l'objectif d'augmenter le temps de visibilité [11] du satellite, on souhaite lui permettre de communiquer avec plusieurs stations lors de son orbite. L'étude consiste alors à caractériser le basculement du satellite, c'est-à-dire la commutation dynamique effectuée entre les deux stations (Figure 3.7).

Dans ce contexte, l'architecture DTN mis en place par ION apporte la surcouche nécessaire :

- à l'interconnexion entre le domaine spatial et le domaine terrestre ;
- au choix du chemin correspondant à la position du satellite.

Malheureusement, des bugs couplés à un manque de temps pour leur résolution n'ont pas permis à l'époque de constater le bon déroulement du scénario. Après discussion, il m'a donc semblé pertinent de réaliser mon évaluation d'ION par l'étude de ce scénario.

11. On dit qu'un satellite est «visible» par une station lorsque sa position en orbite lui permet de communiquer avec ladite station

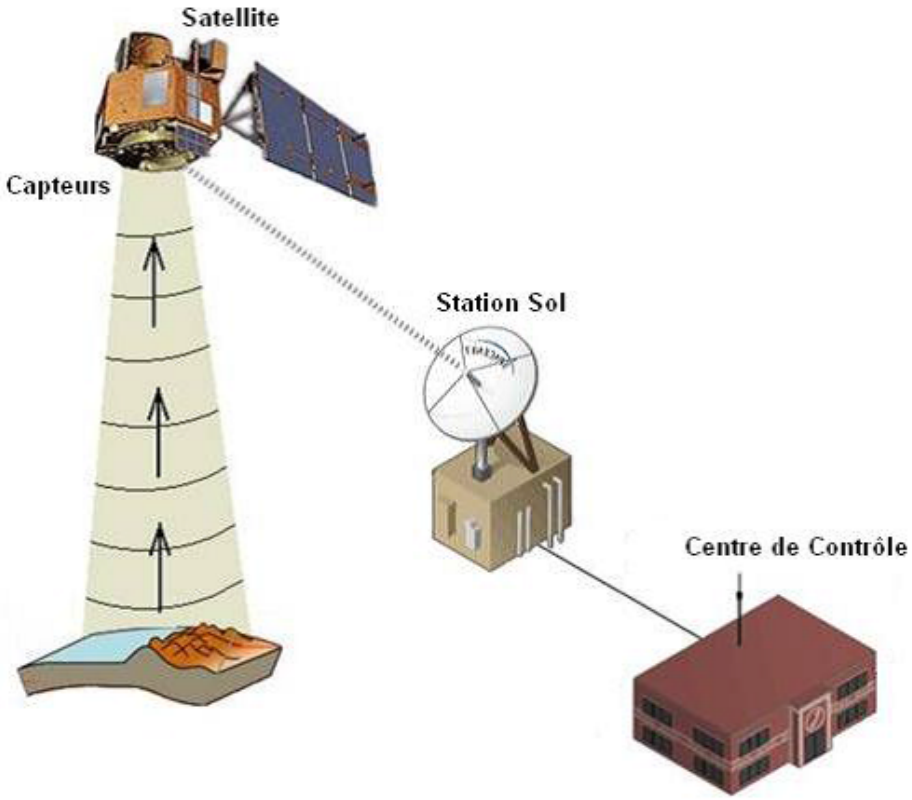


FIGURE 3.6 – Lien bord/sol type pour la télédétection

En revanche, la caractérisation mathématique et exhaustive des mécanismes ayant déjà été réalisée lors de ces stages, il ne nous a pas semblé nécessaire de les réitérer dans le détail.

Dans ce but, il a été décidé de procéder par étapes, en augmentant au fur et à mesure la complexité des scénarii :

1. Communication locale ;
2. Communication entre deux machines puis via un nœud intermédiaire ;
3. Communication entre deux machines sur lien stressé ;
4. Étude du basculement du satellite.

Les deux premières étapes constituent essentiellement des étapes de prise en main du logiciel et ne seront pas décrites dans le présent rapport. Elles ont néanmoins permis de mettre en évidence les caractéristiques rencontrées au préalable à l'utilisation de DTN2 («store-and-forward» notamment, cf. partie 3.4.2).

Communication entre deux machines sur lien stressé

Cette étape intermédiaire a pour but de caractériser le comportement d'ION sur un lien très stressé (pertes, délais, …), afin de pouvoir se consacrer par la suite à l'étude du basculement pur, sans se soucier d'éventuels problèmes causés par le caractère difficile des liaisons spatiales.

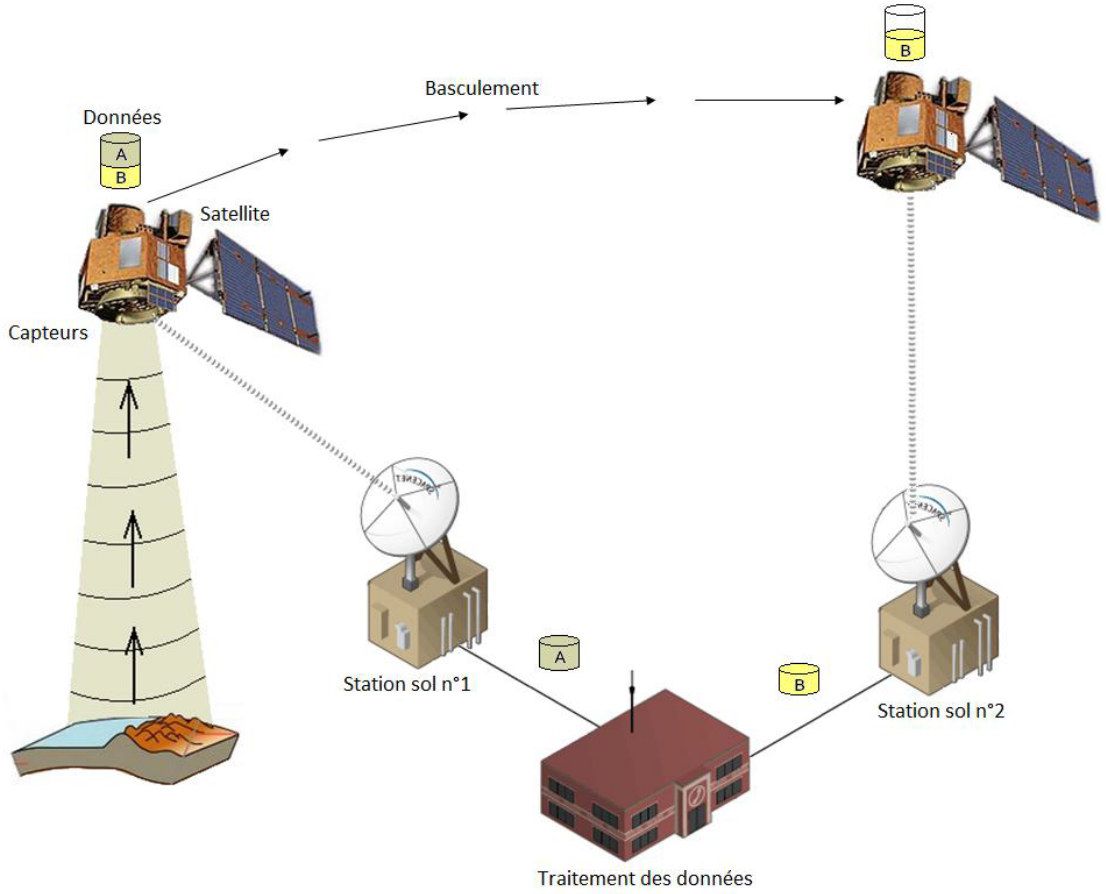


FIGURE 3.7 – Basculement du satellite

Ainsi, nous mettons en œuvre une liaison telle qu'illustrée par la Figure 3.8. Pour ce faire, nous construisons un réseau « physique » à l'aide de Virtualbricks (Figure 3.9).

Dans un premier temps, les commandes suivantes sont utilisées pour les tests :

`bpsource <destinationEndpointId> ['text'] [-t TTL]`
Ouvre un shell permettant la saisie de texte, le texte saisi est ensuite envoyé au destinataire spécifié en argument.

`bpsink <ownEndpointId>`
Se met en écoute.

L'analyse de l'échange réseau est effectuée avec l'aide du logiciel Wireshark. Ce même logiciel sera utilisé pour chaque test décrit dans la suite du présent rapport.

NB : dans tous les cas, le service « 0 » (EID de type `x.0`) ne doit JAMAIS être utilisé par une application, car le démon ION s'en sert déjà.

On constate que la transmission s'effectue malgré les caractéristiques du lien. En cas de déconnexion à l'improviste, LTP se place dans un mode d'attente puis reprend la transmission dès le rétablissement du lien (Figure 3.10). On met ainsi en évidence sa forte tolérance à la disruption/aux délais.

L'étape suivante consiste à évaluer le protocole CFDP (voir partie 2.1.2) implémenté dans

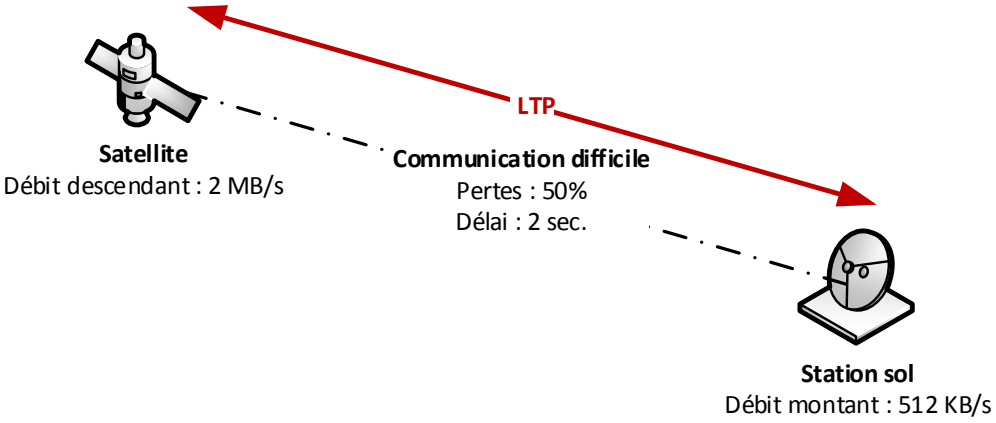


FIGURE 3.8 – Scénario d'étude d'ION sur lien stressé

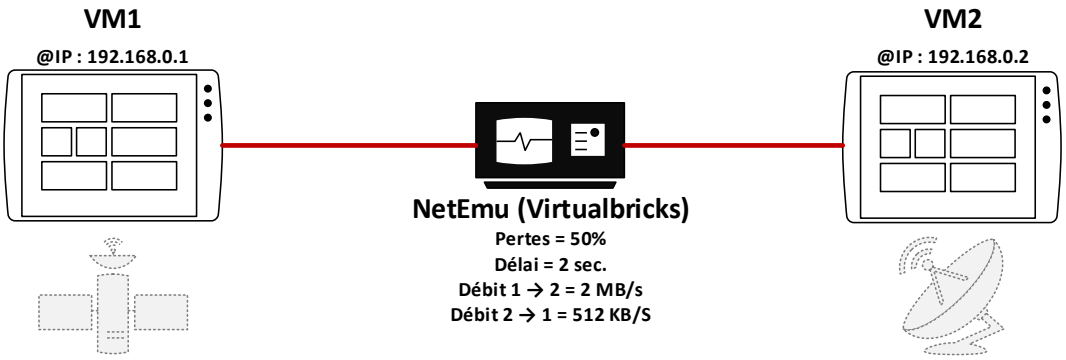


FIGURE 3.9 – Réseau construit pour l'étude du lien stressé

ION. En effet, nous avons vu en partie 2.3 (objectifs du stage) que le démonstrateur bout-en-bout à expérimenter devait utiliser CFDP pour le transfert des fichiers du centre de contrôle vers le satellite (Figure 3.11). ION implémente CFDP mais uniquement sa classe 1, *unreliable transfert* (source [10]), ce qui signifie qu'il n'y a pas de système d'acquittements au niveau CFDP : la fiabilité du transfert doit donc être gérée par les couches inférieures, ce qui est notre cas à l'aide de LTP.

Pour utiliser CFDP, il faut procéder comme suit :

1. Ouvrir les endpoints `x.64` et `x.65` dans la configuration des nœuds terminaux (bpadmin), ils correspondent aux EID utilisés par CFDP dans ION ;
2. Dans ce même fichier de configuration, ajouter une partie «cfdpadmin» entre «bpadmin» et «ipnadmin» (Figure 3.12). Ceci fait démarrer le démon CFDP au démarrage du nœud ;
3. Écrire un petit fichier de configuration `fichier_de_conf` au niveau de l'émetteur, paramétrant l'envoi du fichier (Figure 3.13) ;

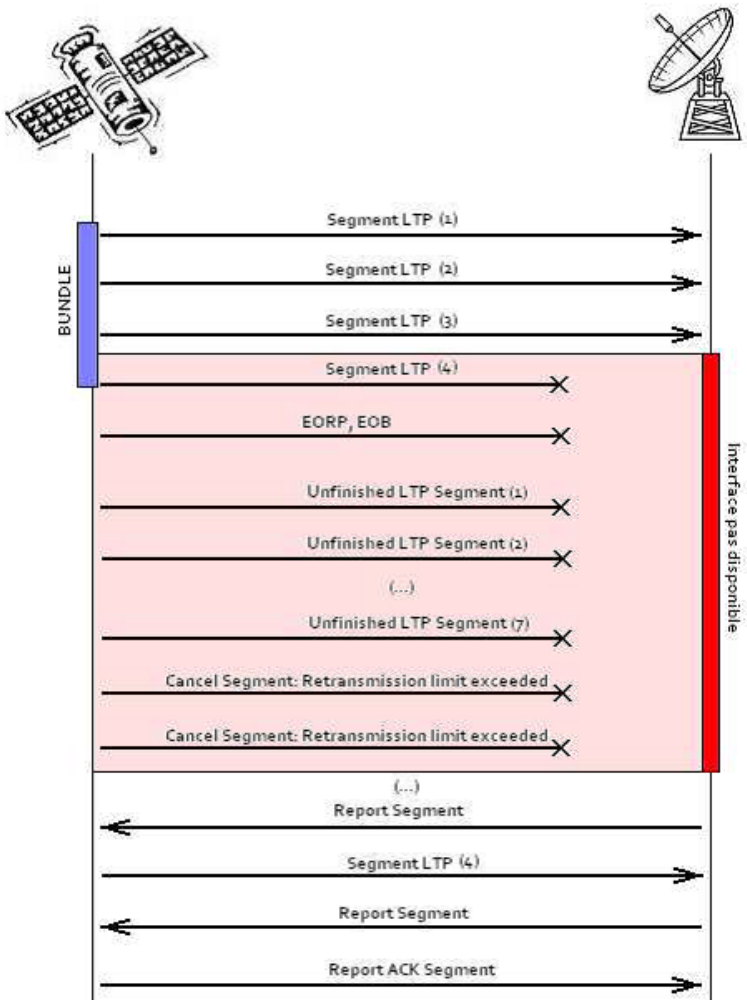


FIGURE 3.10 – Chronogramme d'échanges réseaux pendant la coupure du lien spatial

4. Envoyer le fichier à l'aide de la commande `cfdptest <fichier_de_conf>`

Nous transférons des fichiers de 300KB, 650KB et 10MB. La suite de cette partie synthétise les résultats obtenus lors de nombreux tests : diminution des pertes, changements de configuration LTP, utilisation de CFDP ou non, variation des tailles d'images envoyées...

Premièrement, nous constatons que la communication peut se bloquer en cas de pertes importantes. Ceci est dû à une configuration incorrecte du moteur LTP. En effet, nous avons vu qu'il était possible de paramétrer les sessions LTP dans le fichier de configuration ltprc, en spécifiant notamment leur nombre maximal en simultané ainsi que la taille maximale de leur bloc. Cette configuration est **essentielle** au bon fonctionnement du protocole. En effet, si la capacité du moteur LTP est trop faible comparée aux propriétés de la liaison, la communication risque fortement d'entrer dans une boucle de blocage, ou du moins dans un mode de transmission extrêmement lente.
L'augmentation du nombre de sessions et/ou de la taille des blocs permet de multiplier les « essais de transmissions », de mettre en parallèle les timers et ainsi drastiquement augmenter la capacité de transmission du moteur LTP. Dans notre cas, nous souhaitons rester dans une dynamique « temps réel » et ne pas attendre davantage d'agrégation au niveau du bloc LTP, c'est pourquoi nous préférons travailler sur un grand nombre de sessions. Notons toutefois que le nombre de sessions ne doit pas dépasser une certaine limite, au risque là encore de bloquer la transmission, cette fois à cause d'une sur-

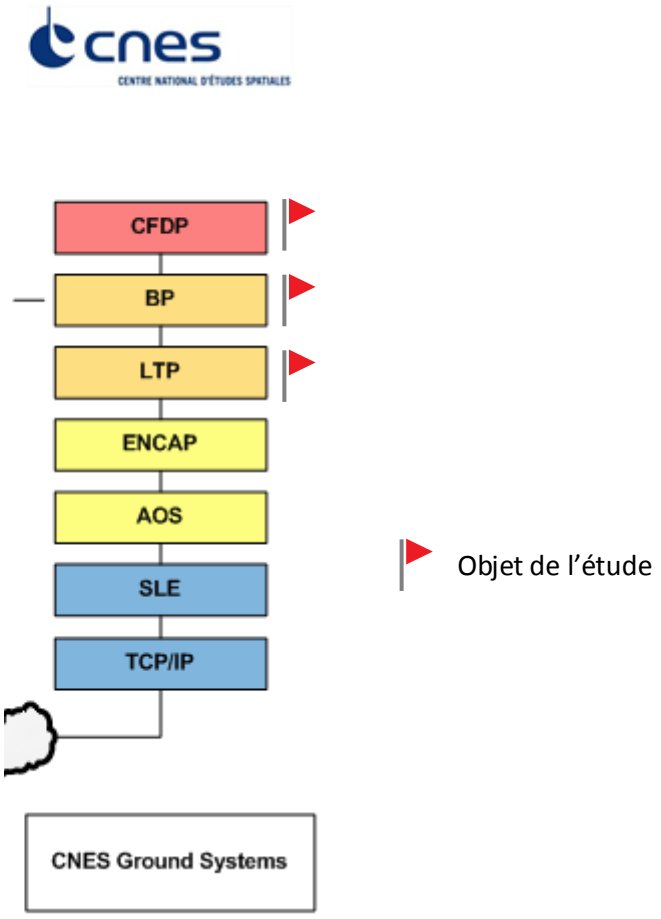


FIGURE 3.11 – Pile protocolaire pour le centre de contrôle : étude des couches hautes

charge de la mémoire vive. Un tableau de calcul, disponible dans les sources d'ION (`ION-LTP-configuration`), permet de connaître les bons paramètres à communiquer au moteur LTP ;

Deuxièmement, **les contacts intermittents sont bien gérés** mais doivent fournir des fenêtres de transmission suffisamment importantes pour que celle-ci puisse s'effectuer. Cela paraît logique mais, si le lien est trop intermittent et difficile (délais !), le moteur LTP n'aura jamais d'occasion de transmettre un bloc avec succès.

Troisièmement, **le protocole CFDP fonctionne correctement**, il a permis de transférer avec succès des images de plus de 10MB.

Enfin, bien que, malgré les caractéristiques du lien, le transfert effectif de l'image se soit effectué dans un temps raisonnable (il subit de façon quasi linéaire les pertes), il n'en est rien de l'arrêt de la transmission, qui peut aisément prendre dix fois plus de temps, alors que l'image est déjà présente, intègre, dans le dossier de destination. Ceci est dû aux nombreuses pertes des *Report Segment* et *Report Acknowledgment Segment*, faisant souvent croire à LTP que le lien est brisé, impliquant son arrêt, sa mise en attente, etc. Il serait intéressant de caractériser plus finement l'impact des pertes sur le temps de transmission d'un fichier par CFDP, cela sera rappelé en partie 3.9.

```
[…]
## end bpadmin

## begin cfdpadmin
1
s 'bputa'   #démarre le démon CFDP
e 1   #optionnel (log)
w 1   #optionnel (debugging)
## end cfdpadmin

##begin ipnadmin
[…]
```

FIGURE 3.12 – Paragraphe à ajouter au fichier de configuration ION pour utiliser CFDP

```
d  2        #destination number
f  local_file_path
t  remote_file_path
&     #send
q     #quit
```

FIGURE 3.13 – Exemple de fichier de configuration CFDP pour cfdptest

Maintenant que la communication sur lien stressé est correctement maîtrisée, nous pouvons passer à l'étude du basculement du satellite.

Étude du basculement du satellite

Nous mettons en œuvre le scénario illustré par la Figure 3.7. Pour cela, nous réalisons tout d'abord un réseau « physique » simple dans Virtualbricks : toutes les machines sont branchées au même switch virtuel (Figure 3.14). Le réseau overlay est ensuite configuré via ION, de telle sorte qu'il représente le scénario étudié (Figure 3.15). Le centre de contrôle **(4)** est lié de façon permanente aux stations sol **(2)** et **(3)**. Comme il s'agit d'un réseau terrestre classique, le protocole TCP/IP est utilisé. Le satellite **(1)**, quant à lui, est lié de façon intermittente aux stations **(2)** et **(3)** (le lien actif change toutes les 30 secondes).

Nous utilisons les commandes `bping` et `bpecho` pour le test de notre architecture. Ces commandes s'utilisent de façon analogue à `bpsource` et `bpsink`, à ceci près que leur fonction est d'envoyer un « ping » [12] toutes les secondes. Le ping résultant est équivalent à un ping ICMP, mais au niveau de la couche Bundle.

12. Nom d'une commande informatique permettant de tester l'accessibilité d'une autre machine à travers un réseau

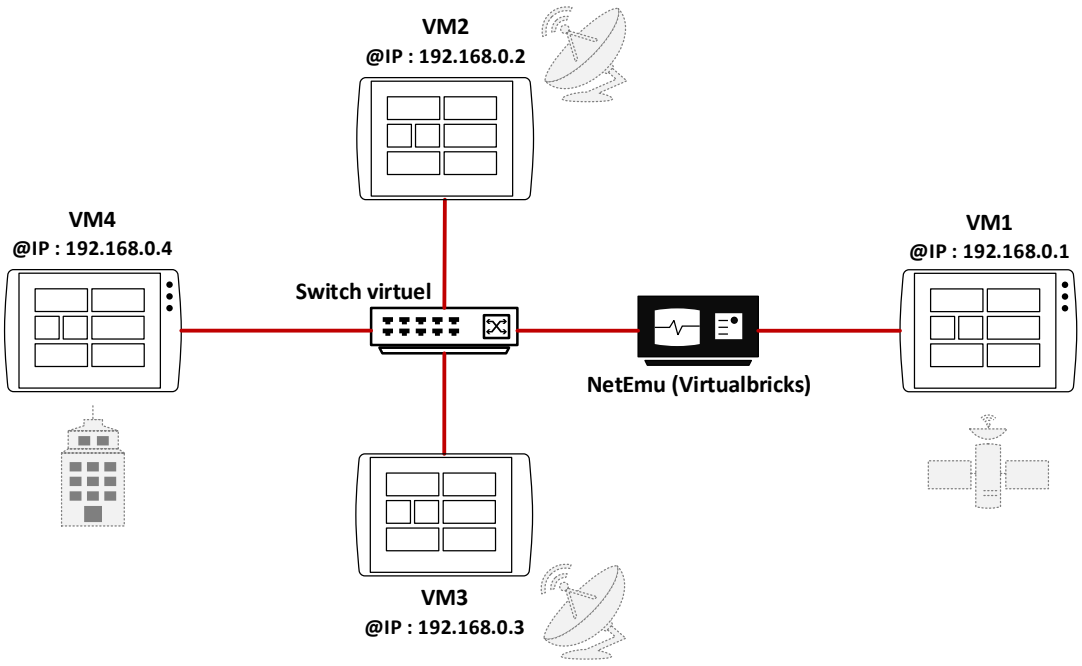


FIGURE 3.14 – Premier réseau construit pour l'étude du basculement

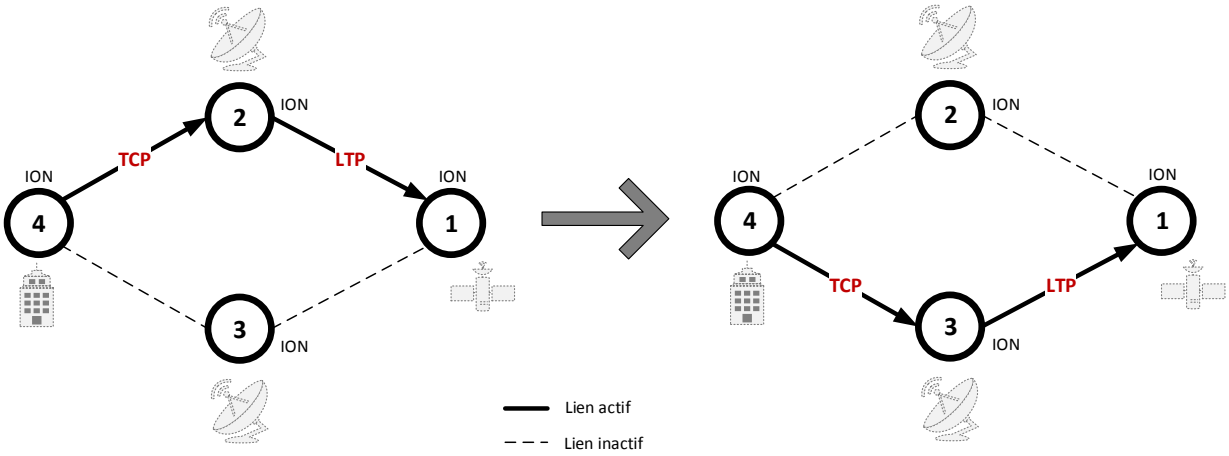


FIGURE 3.15 – Réseau overlay ION pour le scénario de basculement

Les résultats attendus sont alors les suivants :

1. Le centre de contrôle envoie les pings via la station **(2)**, puis le satellite répond par le même chemin. A 30", le chemin change et passe par la station **(3)** (Figure 3.15), et inversement toutes les 30 secondes ;
2. En cas de lien terrestre brisé, le centre de contrôle envoie l'ensemble des pings à l'autre station, même si cela va induire des délais compte tenu du caractère intermittent du lien spatial ;
3. Si un ping est reçu lors du basculement, la réponse emprunte l'autre chemin ;
4. Si le lien spatial est encore actif pour un temps inférieur au délai de propagation, le ping emprunte l'autre chemin, ainsi le satellite pourra le recevoir directement plutôt que de devoir attendre le prochain basculement ;
5. Si des deux liens spatiaux est brisé, le ping est transmis à l'autre station en passant par le centre de contrôle, de sorte qu'il « fasse le tour ».

Résultat 1. Nous avons constaté un premier basculement à 30", puis le centre de contrôle **(4)** continue d'envoyer tous ses bundles à la station **(3)**, au lieu de rebasculer vers la station **(2)**. De fait, lors des intervalles où la station **(3)** n'est pas liée au satellite, elle conserve temporairement les bundles puis les transmet d'une seule traite dès le rétablissement du contact. Après avoir tenté pendant un certain temps de résoudre ce problème, nous avons contacté les développeurs principaux d'ION. Nous supposions une erreur provenant de CGR c'est pourquoi, avec leur aide, nous avons utilisé des outils de *debugging* avancés afin de mettre en évidence ladite erreur :

1. Positionnement du flag `CGR_DEBUG` à 1. Cela permet d'obtenir des traces de CGR (à chaque fois qu'il effectue un choix) en sortie standard lors de l'exécution du logiciel. A l'heure où j'écris ce rapport, le flag ne peut être positionné à 1 qu'en modifiant directement sa valeur dans le code source (l. 16 du fichier bp/ipn/ipnfw.c). A terme, il doit être possible de le modifier à la configuration du compilateur (option de la commande `configure` à l'installation) ;
2. Utilisation d'un programme dédié, `cgrfetch`, simulant l'envoi de bundles dans le but de récupérer les informations de routage CGR correspondant. Ce programme s'utilise sous la forme `cgrfetch {[OPTIONS] DEST-NODE } | -l`. Il génère un fichier JSon [13], visualisable à l'aide d'une page web (contrib/cgr-viewer) sous la forme d'un graphe (Figure 3.16).

Ces outils ont permis à M. Scott Burleigh, un des développeurs principaux d'ION, de comprendre l'erreur et la corriger. Le problème provenait du fichier `bp/cgr/libcgr.c`. La version corrigée a été incluse à notre version de travail mais, à l'heure où j'écris ce rapport, n'est pas présente dans la version publique. Après recompilation du logiciel, **nous obtenons les résultats escomptés concernant la commutation de chemin de la part du centre de contrôle (4)**.

Résultat 2. En cas de défaillance d'une station, le centre de contrôle continue de lui envoyer des bundles pendant un certain temps (10 fois) puis, ne recevant pas d'acquittement, arrête l'envoi. On note que le nœud préfère arrêter temporairement l'envoi de bundles plutôt que de leur faire emprunter un autre chemin (dans notre cas, les envoyer

13. JavaScript Object Notation est un format de données textuel et générique

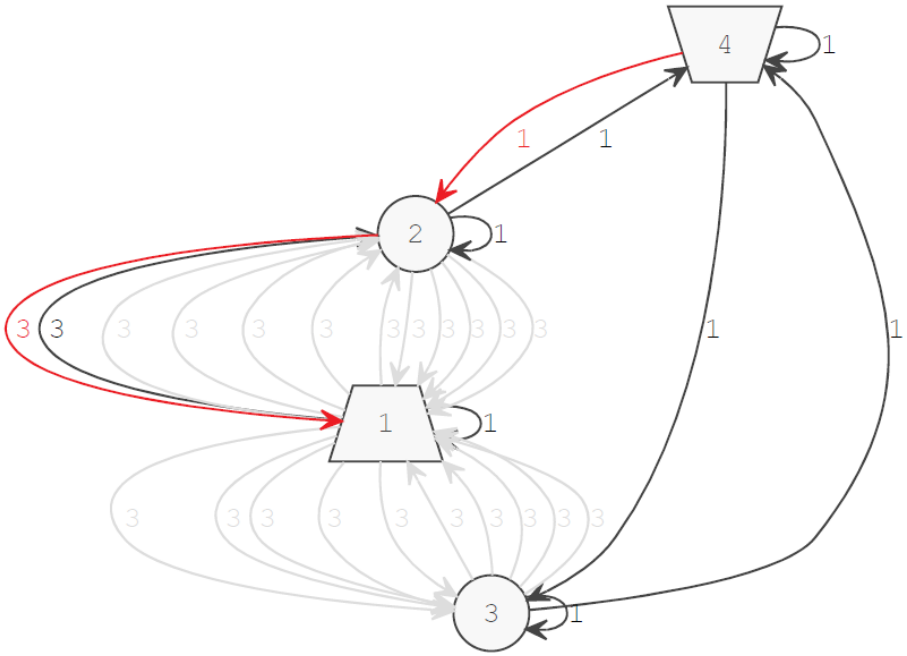


FIGURE 3.16 – Exemple de résultat produit par cgrfetch

à la deuxième station). Ceci montre une des limites de CGR, qui s'adapte mal aux pannes des nœuds et, par extension, aux contacts peu et/ou non planifiés. Cependant, dans le contexte de la communication spatiale, la redondance des équipements permet de garantir leur fiabilité, ce qui tamponne cette faiblesse. CGR reste donc bien adapté au spatial, où l'on peut planifier correctement les contacts, mais peu adapté à d'autres contextes.

Résultat 3. Correctement observé. La requête ping et sa réponse sont totalement indépendants du point de vue de la couche Bundle et des couches sous-jacentes. Leur chemin peut donc être décorrélé.

Résultat 4. Non observé. En effet, les caractéristiques des liaisons – distance en secondes-lumière, débit estimé – définies dans le fichier de configuration (ionadmin) sont utilisées par CGR, mais uniquement pour le calcul du plus court chemin. Elles lui permettent de sélectionner le chemin de meilleure qualité globale mais pas d'anticiper le délai de propagation précédant la disparition d'un contact.

Résultat 5. Après réflexion, il n'était pas logique de nous attendre à ce résultat. En effet, LTP est conçu pour être très robuste aux délais et/ou coupures de connexions. Par essence, il n'a donc aucune raison d'abandonner l'utilisation d'une liaison lorsque celle-ci devient indisponible. Cependant, on pourrait considérer qu'un lien spatial ne doit plus être emprunté si son temps d'indisponibilité a dépassé un seuil donné. Or, il n'existe pas dans LTP de mécanisme permettant de tuer une session au bout d'un certain temps d'inactivité. Pour ce faire, il faut passer par la couche Bundle, en paramétrant la durée de vie maximale (Time-To-Live) des bundles sur le réseau. Si cette durée est dépassée, la couche Bundle est capable de mettre fin aux sessions LTP correspondantes. Le bundle peut alors emprunter un autre lien… sauf si le premier est toujours censé être présent d'après le graphe de contacts (cf. paragraphe précédent).

Une fois cette expérimentation menée, nous avons peaufiné notre scénario en construisant un réseau « physique » (toujours avec Virtualbricks) plus proche du réseau réel, à savoir une séparation du réseau en deux sous-réseaux ainsi que des liens cohérents avec

le graphe de contact (Figure 3.17). Nous avons, là encore, observé qu'ION interconnectait avec succès les deux régions très hétérogènes que sont la région spatiale et la région terrestre.

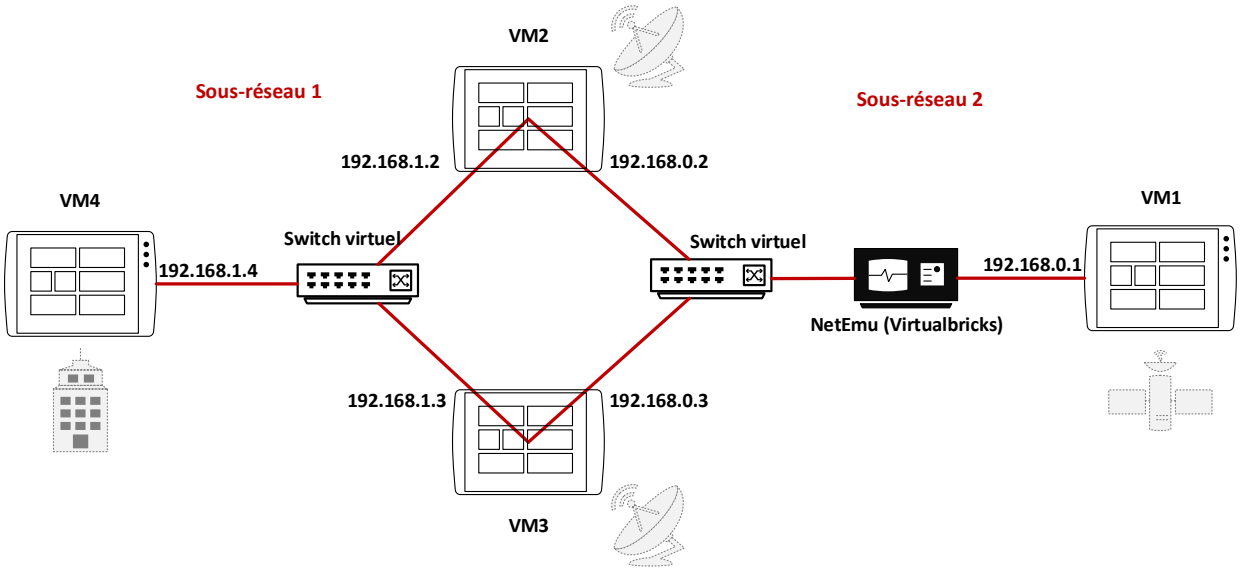


FIGURE 3.17 – Deuxième réseau construit pour l'étude du basculement

L'étude du basculement étant terminée, nous avions l'assurance de maîtriser suffisamment ION pour pouvoir passer à la phase de codage.

3.6 Mise en œuvre d'ION sur protocoles spatiaux

3.6.1 Motivation

Comme nous l'avons vu, le protocole LTP implémenté dans ION fonctionne par-dessus UDP via le **Link Service** UDPLS. Le Link Service à utiliser est indiqué au moment de la configuration des « spans » (ltpadmin). Cette indication fait référence à un programme du même nom, qui se charge de transmettre effectivement les segments LTP à travers le réseau réel.

La liste des implémentations existantes est visible dans le dossier `ltp` du code source d'ION. On remarque un dossier `udp` contenant notamment les fichiers sources `udplsi` et `udplso`, mais également d'autres implémentations : AOS, DCCP, SDA [14], ...

Rappelons que l'objectif du stage est de faire fonctionner le réseau DTN par-dessus la pile protocolaire qui sera effectivement utilisée en opération (Figure 2.15). On observe que, côté bord, la couche AOS s'exécute directement au-dessus de la couche physique (RF pour RadioFréquence). En effet, la liaison de données est assurée par ce protocole, les couches sol inférieures (SLE - TCP/IP) concernent un système de passerelles qui sera détaillé en partie 3.8. En résumé, nous considérons dans un premier temps la communication bout-en-bout centre de contrôle <–> ISS selon la pile protocolaire illustrée par la Figure 3.18.

14. Service Data Aggregation

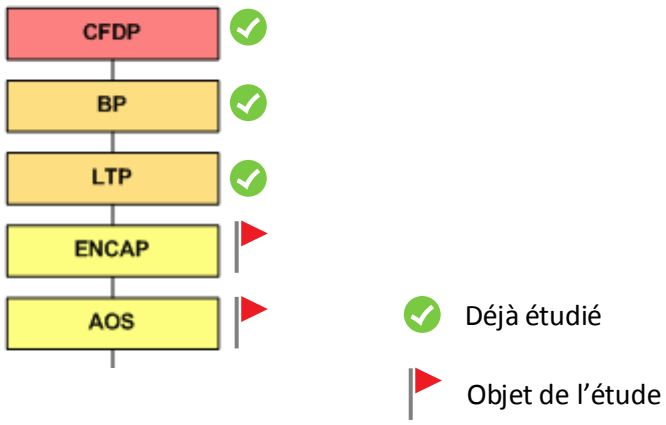


FIGURE 3.18 – Pile protocolaire pour la liaison de données CC <–> ISS

Dans ce but, nous sommes forcés de modifier le code source d'ION afin d'implémenter notre propre version de Link Service : ENCAP – AOS. Après une brève présentation de ces deux protocoles, nous expliquerons la méthode utilisée pour les implémenter dans ION.

3.6.2 Le protocole ENCAP

Encapsulation Service est un standard CCSDS permettant, comme son nom l'indique, l'encapsulation de PDUs [15] de haut niveau dans un paquet sobre contenant quelques informations utiles. Ce paquet peut ensuite être directement envoyé à son destinataire à l'aide d'un protocole de liaison de données.

Basiquement, l'en-tête d'Encapsulation contient quelques champs, pour une taille totale comprise entre 1 et 8 octets (Figure 3.19, source [8]) :

ENCAPSULATION PACKET HEADER

PACKET VERSION NUMBER 3 bits	PROTOCOL ID 3 bits	LENGTH OF LENGTH 2 bits	USER DEFINED FIELD 0 or 4 bits	PROTOCOL ID EXTENSION 0 or 4 bits	CCSDS DEFINED FIELD 0 or 2 octets	PACKET LENGTH 0 to 4 octets
'111'	'XXX'	'00'	0 bits	0 bits	0 octets	0 octets
'111'	'XXX'	'01'	0 bits	0 bits	0 octets	1 octet
'111'	'XXX'	'10'	4 bits	4 bits	0 octets	2 octets
'111'	'XXX'	'11'	4 bits	4 bits	2 octets	4 octets

FIGURE 3.19 – En-tête ENCAPsulation (source [8])

– Packet Version Number (3 bits) : toujours égal au triplet «111» ;

15. Protocol Data Unit : unité de données protocolaire (segment, paquet, ...)

- Protocol ID (3 bits) : identifiant du protocole supérieur. La liste des IDs officiels est disponible sur le site de SANA [5] ;
- Length of Length (2 bits) : indique la taille du champ Packet Length. Permet d'optimiser légèrement la taille de l'en-tête, en disposant d'une taille de Packet Length variable ;
- User Defined Field (4 bits) : champ réservé à l'utilisateur, *optionnel* ;
- Protocol ID Extension (4 bits) : utilisé si le champ Protocol ID est positionné à «110». Identifie le protocole supérieur sur 4 bits. *Optionnel* ;
- CCSDS Defined Field (2 octets) : réservé pour l'avenir, *optionnel* ;
- Packet Length (0 à 4 octets) : taille de la PDU encapsulée dans cet en-tête.

Au total, on remarque une grande redondance dans les champs pour peu d'informations utiles. Les champs « Protocol ID » et « Packet Length » restent néanmoins intéressants s'ils ne sont pas déjà fournis par la couche inférieure, ce qui est notre cas. Actuellement, les utilisations opérationnelles de ce protocole sont rares. Le Space Packet (voir Figure 2.2) est plus conventionnel.

3.6.3 Le protocole AOS

Advanced Orbiting Systems Space Data Link Protocol est également un standard CCSDS, correspondant à un protocole de liaison de données dédié au spatial (source [6]).

En pratique, les trames AOS sont de taille fixe, définie pour la mission. Elles contiennent un en-tête dont une partie est de taille variable, ainsi qu'un trailer de contrôle optionnel (Figure 3.20).

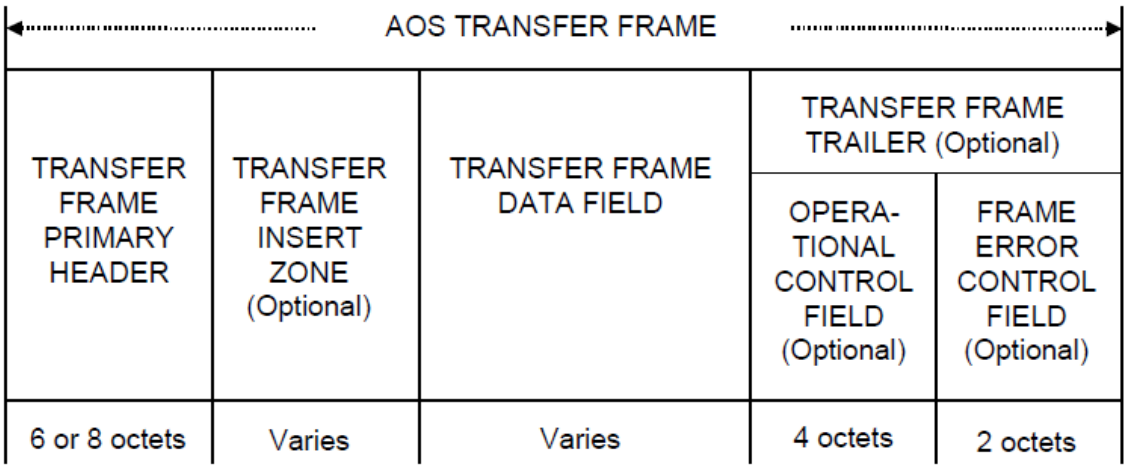


FIGURE 3.20 – Trame AOS (source [6])

Nous nous intéresserons uniquement à l'en-tête primaire dans la suite de cette étude, car les parties optionnelles ne sont pas retenues pour l'expérimentation future.

L'en-tête primaire contient notamment des informations relatives au canal physique utilisé (« master channel »), au canal virtuel éventuel (« virtual channel »), au numéro de trame, selon les champs suivants (Figure 3.21) :

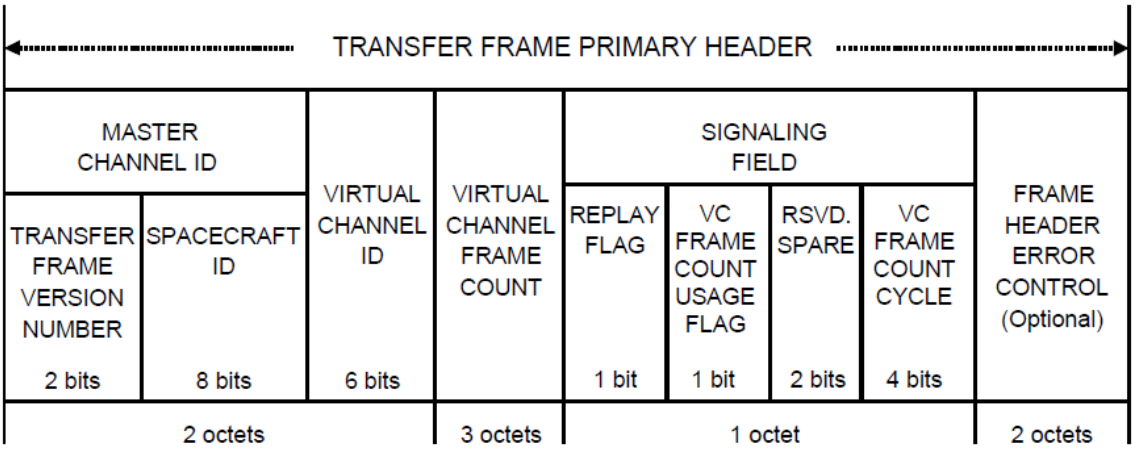


FIGURE 3.21 – Trame AOS (source [6])

- Transfer Frame Version Number (2 bits) : toujours égal au couple «01» ;
- Spacecraft ID (8 bits) : constitue le « Master Channel ID » avec le Transfer Frame Version Number ;
- Virtual Channel ID (6 bits) : identifie le canal virtuel au sein du canal physique ; «000000» si un seul canal virtuel est utilisé, «111111» si l'on a affaire à une trame «idle» [16] ;
- Virtual Channel Frame Count (3 octets) : numéro de la trame ;
- Replay Flag (1 bit) : si positionné à 1, indique que la trame est une copie retransmise d'une trame précédente ;
- VC Frame Count Usage Flag (1 bit) : indique si le champ VC Frame Count Cycle est utilisé ;
- Rsvd. Spare (2 bits) : réservé pour l'avenir ;
- VC Frame Count Cycle (4 bits) : incrémenté de 1 quand le Virtual Channel Frame Count est arrivé en bout de cycle (trame numéro 16 777 216). Non utilisé si le numéro de trame n'excède pas ce nombre ;
- Frame Header Error Control : champ de contrôle d'intégrité de la trame. *Optionnel*.

Comme nous venons de l'indiquer, il est ainsi possible de séparer virtuellement les flux de données à l'aide des canaux virtuels (maximum 64). On note toute de même une certaine redondance, notamment au niveau des champs de type « Frame Count », mais cela a des causes historiques : l'en-tête a été étendu alors qu'il était déjà utilisé en opérations.

3.6.4 Mise en œuvre dans ION

La méthode utilisée est la suivante :

1. Mettre en place un environnement de développement ;
2. Repérer où s'effectue l'utilisation par LTP du service UDP/IP ;
3. Définir les modifications à effectuer pour remplir notre objectif ;

16. Désigne une PDU vide de données, inactive

4. Implémenter ENCAP ;
5. Implémenter AOS.

Mettre en place un environnement de développement

Étant donné que nous voulions modifier le code source d'ION, il nous fallait un moyen de développer de façon sûre et tracée. Pour cela, nous avons mis en place un serveur Git [17], gestionnaire de gestion de versions décentralisé, sur le PC technique du laboratoire. A l'initialisation, le serveur Git a été nourri avec les sources d'ION, plus les corrections évoquées en partie 3.5.2. Des informations supplémentaires sur l'installation et la configuration du serveur Git sont disponibles en Annexe F.

Repérer où s'effectue l'utilisation par LTP du service UDP/IP

Le code d'ION est peu documenté et très peu commenté, j'ai donc dû opérer une phase importante de rétro-ingénierie sur l'ensemble du code source. Un document présent dans le code source d'ION, «ION.pdf» [10], m'a servi de point d'appui. Ce document dégrossit l'organisation du code source et permet d'identifier les rôles de chaque paquet constituant le logiciel.

De ce travail résulte que les Link Services utilisés par LTP se trouvent dans le paquet du même nom (ltp). Pour chaque Link Service, on dispose de 3 fichiers sources :

- un pour le Link Service Output (par exemple, `udplsi.c`) ;
- un pour le Link Service Input (par exemple, `udplso.c`) ;
- un header (par exemple, `udplsa.h`) pour l'importation des fonctions extérieures et la définition de constantes.

Basiquement, pour UDPLS, les programmes «LSI» et «LSO» créent une socket [18], la connectent à la machine distance (paramètres définis dans ltpadmin), puis bouclent sur un algorithme de communication avec le moteur LTP. En sortie, les segments à envoyer sont dépilés du moteur LTP puis fournis à la socket UDP (Figure 3.22). En entrée, les segments reçus sont empilés (Figure 3.23). Le moteur LTP gère lui-même la gestion plus haut niveau des segments, on observe donc un réel découplage entre les deux couches, ce qui est véritablement pratique lorsque l'on a besoin de changer une fonctionnalité.

Définir les modifications à effectuer pour remplir notre objectif

Nous prenons la décision de nous baser sur UDPLS et le modifier pour construire notre propre Link Service : **CNESLS**. L'ensemble de la liaison de données AOS sera donc empilée sur UDP/IP, ce qui n'est pas problématique dans un premier temps puisque nous souhaitons simplement tester notre implémentation. De plus, il suffit de remplacer les

17. http ://www.git-scm.com
18. Élément logiciel servant d'interface par laquelle un développeur exploitera facilement les services d'un protocole réseau

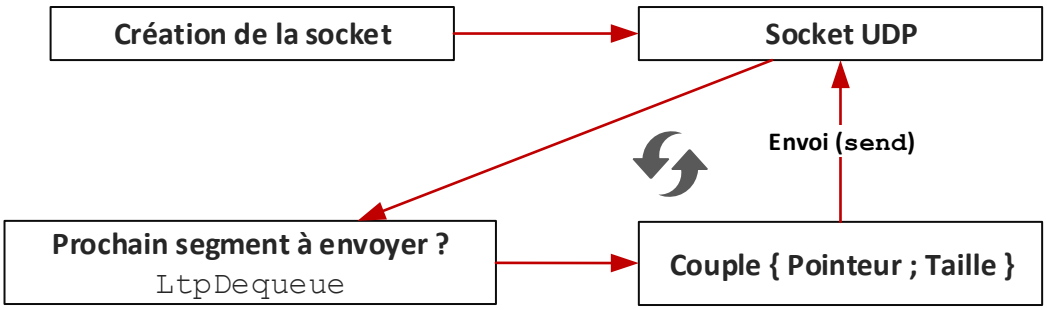


FIGURE 3.22 – Fonctionnement du programme UDPLSO

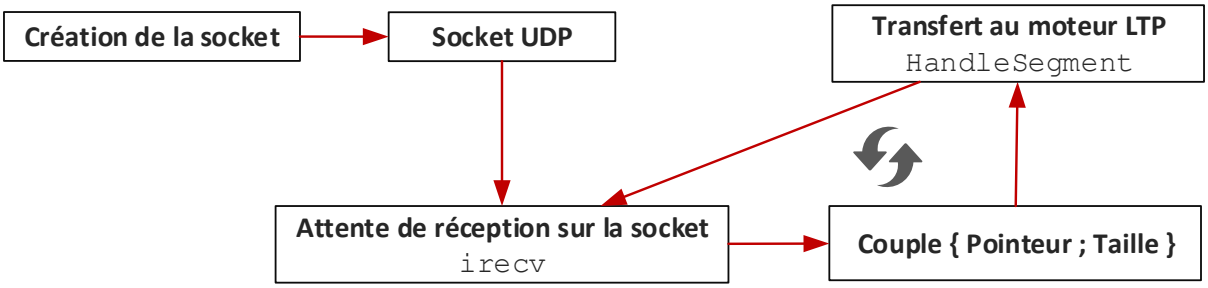


FIGURE 3.23 – Fonctionnement du programme UDPLSI

fonctions de l'API socket par des fonctions d'une API relative à un autre mode de transmission pour modifier le protocole sous-jacent à AOS.

Ainsi, nous cherchons à modifier les *buffers* de données en transit entre la socket UDP et le moteur LTP, dans le but de leur enlever ou ajouter nos en-têtes. Deux solutions principales s'offrent à nous : créer un nouveau buffer en y déversant les données de l'ancien (+ modifications), ou travailler directement sur le buffer original. Nous avons constaté que la gestion de la mémoire dans ION était très complexe à mettre en œuvre car elle utilisait des fonctions propres. Nos essais se basant sur la première méthode ne se sont pas avérés concluants à cause de cela, c'est pourquoi nous avons décidé de sélectionner la seconde.

Les Figures 3.24 et 3.25 illustrent l'algorithme qui a été mis en œuvre par-dessus UDPLS pour créer CNESLS. Notamment, une bibliothèque de fonctions en C relatives à la manipulation des en-têtes ENCAP et AOS a été créée, ce qui permet leur réutilisation potentielle dans d'autres contextes. L'ensemble des fichiers de CNESLS sont disponibles dans le dossier `sources-ion/ltp/cnes`.

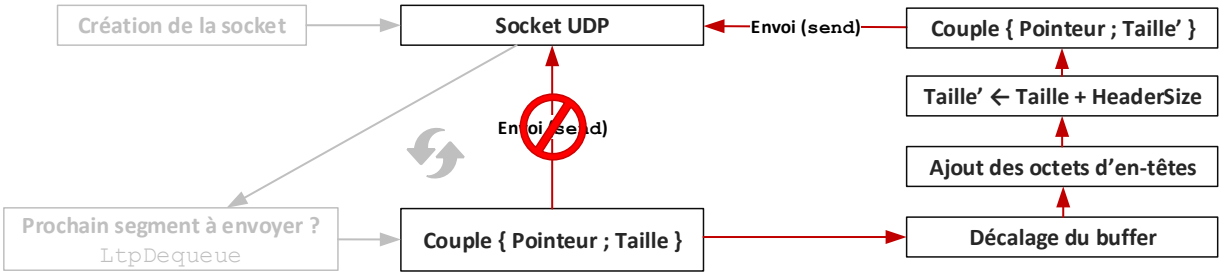


FIGURE 3.24 – Fonctionnement du programme CNESLSO

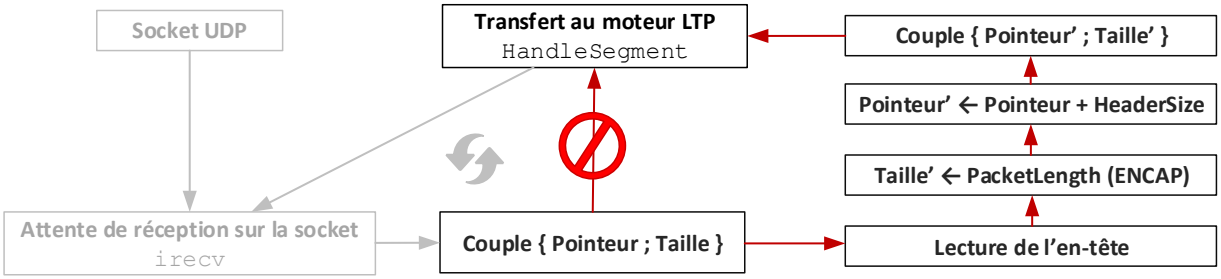


FIGURE 3.25 – Fonctionnement du programme CNESLSI

Implémenter ENCAP

Voici comment les valeurs de champs de l'en-tête ENCAP sont définies dans notre implémentation :

- Packet Version Number (3 bits) : toujours égal au triplet «111» ;
- Protocol ID (3 bits) : identifiant de LTP soit «001» (source [5]) ;
- Length of Length (2 bits) : dépendant de la valeur de Packet Length. Utilisé en réception pour connaître la taille de l'en-tête ;
- User Defined Field (4 bits) : non utilisé ;
- Protocol ID Extension (4 bits) : non utilisé ;
- CCSDS Defined Field (2 octets) : non utilisé ;
- Packet Length (0 à 4 octets) : taille du segment retournée à son dépilement (fonction `LtpDequeue`). Utilisé en réception pour connaître la taille du segment LTP encapsulé.

Implémenter AOS

Voici comment les valeurs de champs de l'en-tête AOS sont définies dans notre implémentation :

- Transfer Frame Version Number (2 bits) : toujours égal au couple «01» ;

- Spacecraft ID (8 bits) : identifiant du nœud dans ION, aussi égal à l'identifiant du moteur LTP ;
- Virtual Channel ID (6 bits) : «000000» car on utilise un seul canal virtuel pour le moment ;
- Virtual Channel Frame Count (3 octets) : numéro du segment retourné par le moteur LTP ;
- Replay Flag (1 bit) : 0 car non utilisé ;
- VC Frame Count Usage Flag (1 bit) : 0 car VC Frame Count Cycle n'est pas utilisé
- RSVD. SPARE (2 bits) : non utilisé ;
- VC Frame Count Cycle (4 bits) : non utilisé ;
- Frame Header Error Control : non utilisé ;

Nous avons vu que la taille des trames AOS devait être fixe. Dans notre cas, nous avons défini une taille de trame de 1275 octets. De plus, la partie caudale de la trame qui n'est pas remplie par la PDU sous-jacente est bourrée avec des 0. Bien évidemment, l'ensemble de ces valeurs pourra aisément être affiné en fonction du contexte opérationnel.

3.7 Architectures ION pour stations sol non-DTN

En dehors de l'étude de réseaux « full-DTN » tels que présentés dans les parties précédentes, une de mes missions consistait à étudier l'impact de la technologie sur un réseau de stations non-DTN. En effet, comme nous l'avons vu en partie 2.3, le CNES souhaite conserver son potentiel d'interopérabilité de moyens sol vis-à-vis d'agences spatiales mettant en œuvre un DTN bout-en-bout. Cela doit pouvoir se faire sans modification de l'existant, il faut donc étudier les mécanismes qui permettraient à un centre de contrôle DTN de profiter d'un groupe de stations sol sans que celles-ci ne connaissent Bundle Protocol ni aucun protocole lié. Cela rentre également dans le cadre de l'expérimentation en collaboration avec la NASA, bien qu'une solution soit déjà proposée (passage par passerelles SLE, voir partie 3.8). La suite de la réflexion se positionne du côté du centre de contrôle.

La première méthode qui vient à l'esprit est l'**utilisation des stations sol sous forme de proxies transparents**. Cela consiste à considérer que la communication avec le satellite est point-à-point, alors que l'on écrit en réalité à la station sol. Cette dernière se contente de transférer le message au satellite sans fournir de mécanisme de niveau Bundle (Figure 3.26).

Cependant, on imagine facilement que cette méthode induit des problèmes de routage dans le cas d'un groupe de stations sol. En effet, si la communication est vue depuis les extrémités comme une communication point-à-point, comment peut-on spécifier plusieurs adresses IP destinataires et donc plusieurs chemins ?

Spécification de plusieurs chemins logiques

Pour qu'ION puisse disposer de plusieurs chemins, il est nécessaire de définir les stations comme des nœuds du réseau DTN, même si ce n'est pas le cas. On les inclut donc au

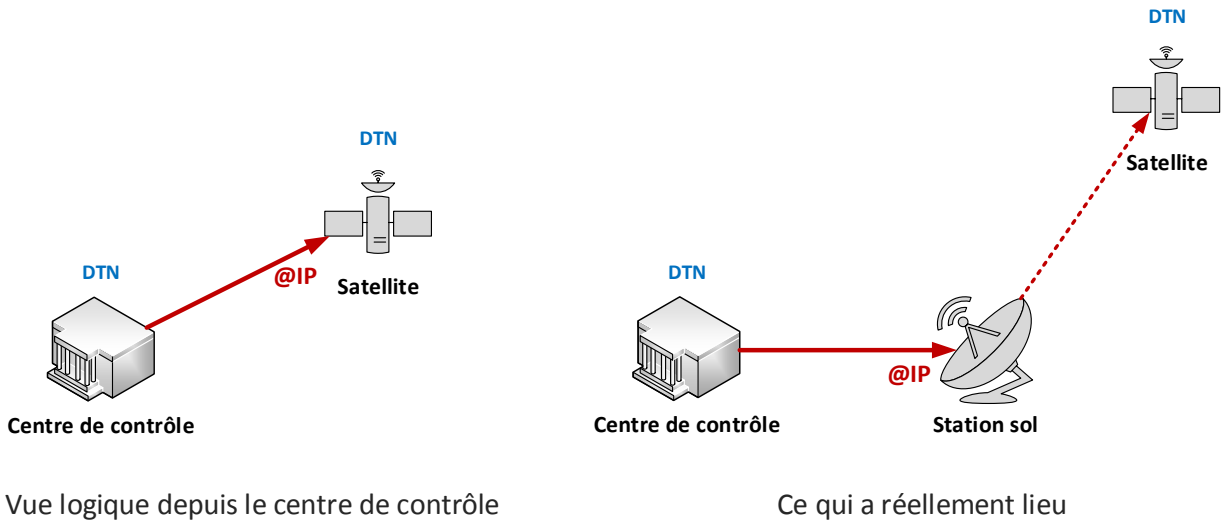


FIGURE 3.26 – Utilisation d'une station par un mécanisme de proxy

graphe de routage (ionadmin). Cependant, cela ne suffit pas car on souhaite dialoguer avec le satellite à l'aide d'une pile protocolaire spatiale (LTP/ENCAP/AOS), que la station ne peut générer. On ne peut donc pas définir les liens centre de contrôle <–> station comme des liens TCP/IP classiques (utilisation de la couche de convergence TCPCL), car la pile protocolaire spatiale ne serait pas présente (Figure 3.27). La solution que je propose alors est de **faire fonctionner la pile spatiale (CNESLS) par-dessus une socket TCP** (actuellement UDP, voir Figure 3.27). Au final, ION est capable d'écrire à plusieurs stations en fonction du temps, tout en réalisant une communication AOS bout-en-bout.

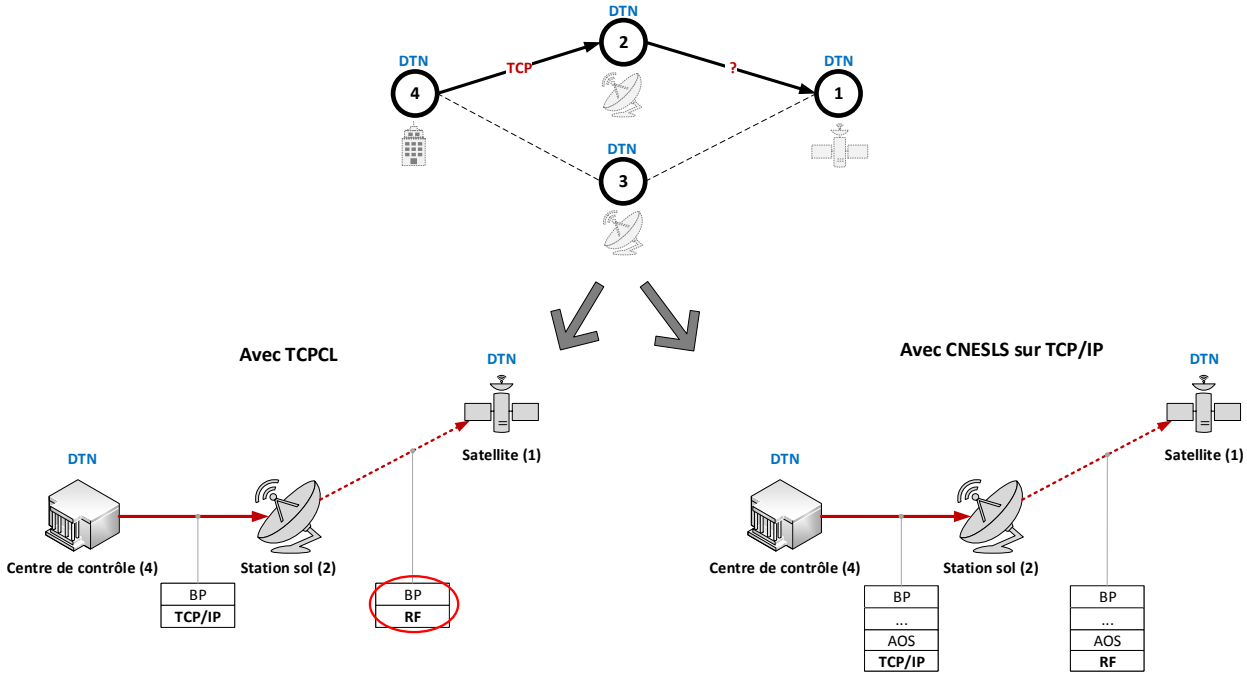


FIGURE 3.27 – Déclaration des stations non-DTN dans le graphe de routage

En pratique, les mécanismes de routage sont réalisés à un niveau inférieur à ION, par des passerelles transparentes. Cela réutilise l'idée illustrée par la figure 3.26, à ceci près que

l'on écrit à une passerelle à la place d'une station sol. La passerelle se charge ensuite de la commutation de stations de façon transparente (Figure 3.28).

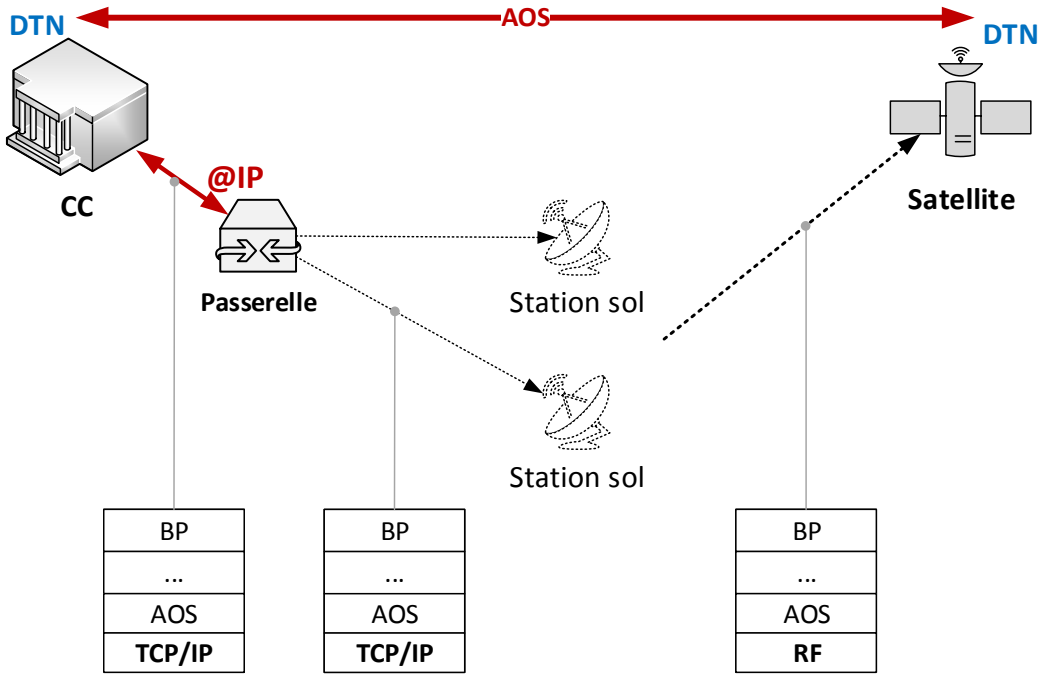


FIGURE 3.28 – Commutation bas niveau par une passerelle

Spécification de plusieurs chemins physiques

Nous avons vu que les solutions actuelles déchargeaient ION de la responsabilité de commutation de stations (choix de la station). Cependant, il reste important de pouvoir différencier les propriétés de chaque chemin : délai, bande passante, ... Ce afin que le moteur LTP situé au niveau du centre de contrôle réagisse correctement à la difficulté de la liaison utilisée pour communiquer avec le satellite à un instant donné.

Cela peut être réalisé en **différenciant virtuellement les contacts en fonction de la station sol utilisée**. Par exemple, lors d'un basculement, on définit un nouveau contact dans ION. Le nœud destinataire reste le même (le changement de chemin est transparent) mais les propriétés sont différentes.

Pour valider cette méthode, nous avons réalisé une petite expérience. On considère une communication centre de contrôle <–> satellite point-à-point, dont la liaison spatiale dispose de caractéristiques évoluant au fil du temps, en fonction du chemin sol emprunté (encore une fois, celui-ci est transparent du point de vue du centre de contrôle). Au niveau physique, le changement effectif de chemin est réalisé par des événements Virtualbricks. On construit 3 chemins physiques aux propriétés différentes, puis on active l'un ou l'autre en fonction du temps (Figure 3.29) : cela représente la commutation de chemin effectuée de façon transparente.

Dans CGR, chaque nouvelle ouverture de chemin est représentée par un contact. Les événements Virtualbricks, quant à eux, permettent de démarrer ou stopper des éléments

du réseau à un instant précis (Figure 3.30). Pour information, ils permettent également l'exécution d'un script shell sur une VM.

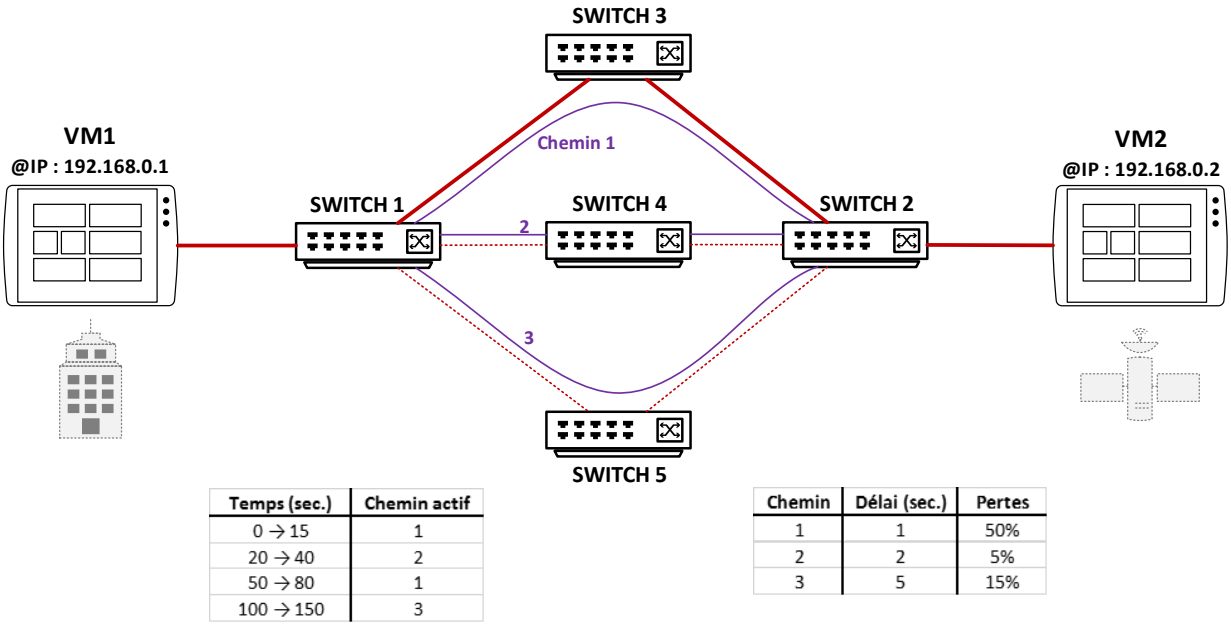


FIGURE 3.29 – Réseau construit pour l'étude de la commutation de chemins

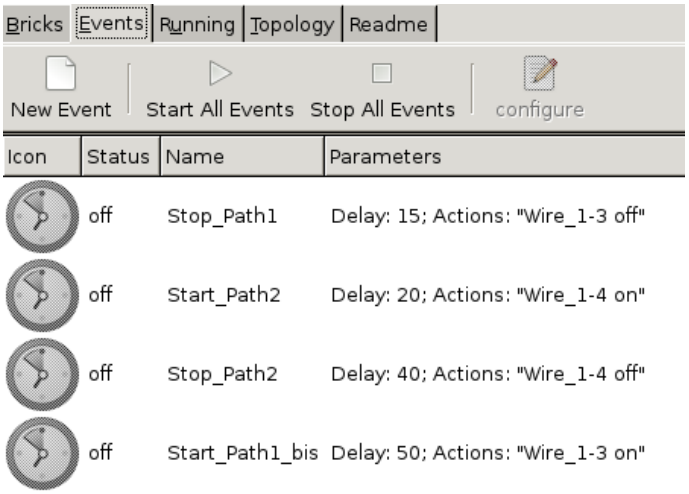


FIGURE 3.30 – Les événements dans Virtualbricks (capture d'écran)

Enfin, on utilise, comme d'habitude, `bping` et `bpecho` pour la communication DTN. On constate que l'ensemble des pings arrivent à destination, parfois avec un fort délai (non anticipation du délai avant coupure du contact, voir partie 3.5.2). Les pings arrivent également dans le désordre, ce qui est normal puisque la couche Bundle ne gère pas la remise en ordre des bundles. Cependant, nous avons ainsi pu gérer correctement la commutation de stations, sans inclure ces dernières à notre topologie CGR.

3.8 Mise en œuvre par passerelles SLE

3.8.1 Présentation

Le **Space Link Extension** est un jeu de services visant à étendre les services bord/sol CCSDS jusqu'à la partie sol, c'est-à-dire accessibles depuis un centre de contrôle. Son but principal est de permettre l'interopérabilité entre agences : un centre de contrôle utilise les moyens sols d'une autre agence tout en profitant d'une communication bord/sol standard avec le satellite.

Le CCSDS a édité un certain nombre de recommandations définissant le SLE, au travers de 4 services principaux (source [7]) :

- **RAF** (**Return All Frames**) : permet à l'utilisateur distant d'avoir accès à toutes les trames reçues par la station depuis le satellite ;
- **RCF** (**Return Channel Frames**) : permet à l'utilisateur distant d'avoir accès à toutes les trames reçues par la station via un canal spécifique (Master Channel ou Virtual Channel) ;
- **FCLTU** (**Forward Command Link Transmission Unit**) : permet à l'utilisateur d'envoyer des trames de télécommande au satellite via la station distante ;
- **ROCF** (**Return Operational Control Fields**) : permet à l'utilisateur distant d'avoir accès aux champs de contrôle opérationnels (contrôle de la transmission des télécommandes).

L'utilisation de SLE est composée de deux parties :

- une capacité utilisateur (« **user** »), correspondant à la possibilité pour un centre de contrôle d'utiliser une station compatible SLE d'une autre agence ;
- une capacité fournisseur (« **provider** »), correspondant à la possibilité pour une station sol de fournir une liaison TM/TC SLE à un centre de contrôle d'une autre agence.

Il présuppose également l'utilisation du protocole bord/sol CCSDS (trames et paquets).

La mise en œuvre est opérée par l'installation du service SLE dans les stations et centres de contrôle, soit en « natif », soit par un système de passerelles auxquelles la production du service SLE est déléguée. La première solution étant coûteuse et complexe à mettre en place, c'est actuellement par des passerelles que les éléments sol du CNES utilisent et fournissent les services SLE. En configuration « provider », ces passerelles peuvent être interfacées avec des simulateurs de télémesure. En configuration « user », elles s'interfacent au centre de contrôle.

3.8.2 Expérimentation

Rappelons que, dans l'expérimentation que nous cherchons à réaliser, le centre de contrôle sera CNES mais la station sol sera NASA : SLE permettra le couplage entre ces deux éléments (Figure 3.31).

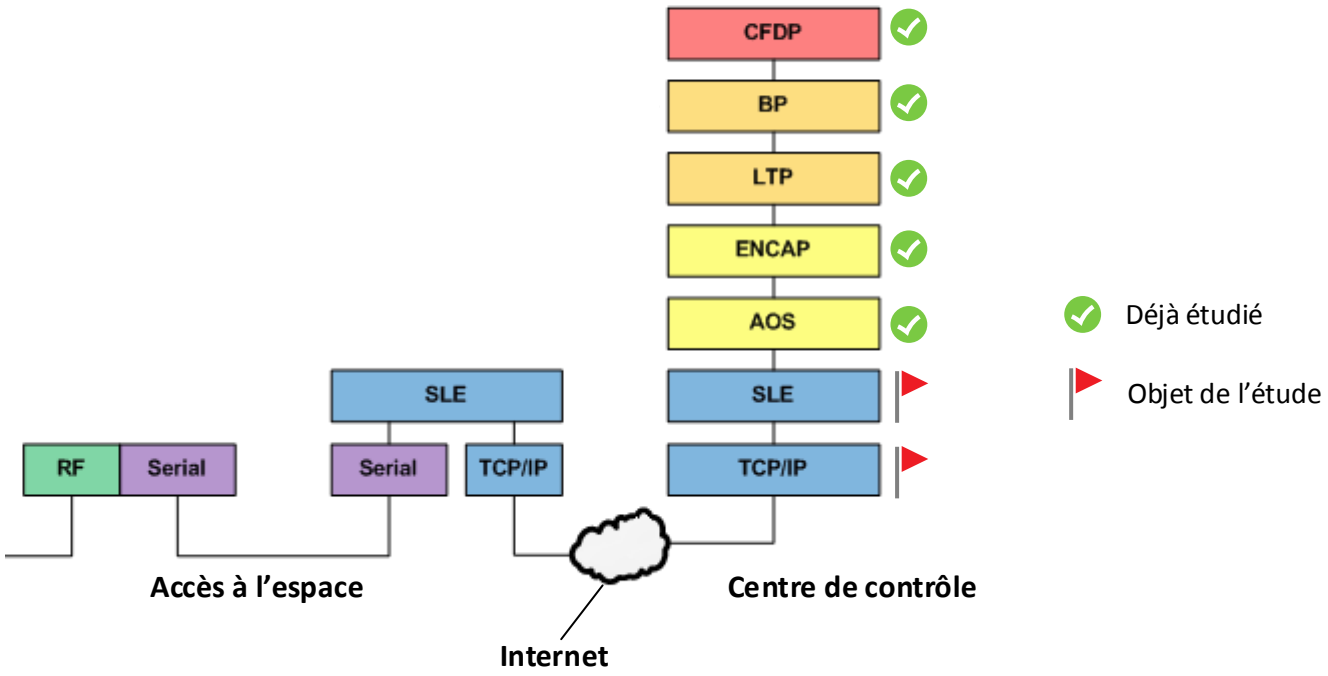


FIGURE 3.31 – Pile protocolaire pour la liaison centre de contrôle <–> station NASA

Nous étudions donc la faisabilité de l'utilisation de SLE par une pile protocolaire ION, en deux étapes :

1. Couplage ION – passerelle user : dialogue entre ION et un simulateur de TM pré-existant ;
2. Dialogue entre ION et une machine virtuelle simulant une télémesure DTN.

La configuration des deux passerelles (user et provider) ne sera pas détaillée dans ce rapport car il s'agissait de configurations conventionnelles. Pour en savoir plus, se référer au manuel utilisateur.

Couplage ION - passerelle user

La tâche consiste à modifier notre Link Service CNESLS (cf. partie 3.6) pour le faire utiliser la passerelle user à l'envoi. Pour cela, on utilise une interface spécifique nommée **BLOC-CNES** : le centre de contrôle ouvre une connexion TCP avec la passerelle, sur le port *bloc_tc*, puis lui envoie ce qu'il veut faire passer en SLE, encapsulé dans un en-tête BLOC-CNES. La passerelle contrôle la conformité de l'en-tête et, le cas échéant, transmet le message à la passerelle provider via SLE (Figure 3.32). Une seconde connexion sur le port *bloc_tm* permet de recevoir les TM transmises par la passerelle provider.

Pour réaliser notre expérience, nous modifions le code source de la passerelle, de sorte que l'en-tête BLOC-CNES ne soit plus utilisé dans la communication « Application – Passerelle user » : aucun ajout par l'émetteur et aucune action de contrôle et de décapsulation par le récepteur.

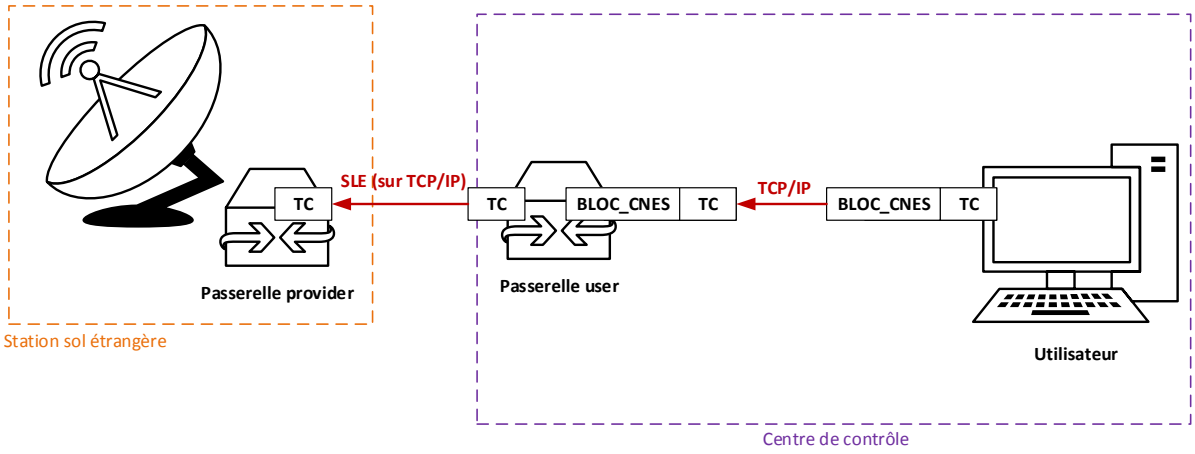


FIGURE 3.32 – Envoi d'une TC par une station sol étrangère à l'aide de SLE

A titre informatif, les fichiers modifiés sont les suivants :

- `PL_EnvoyerTrameBlocCnes` ;
- `PL_TraiteTCBlocCnes` ;
- `PL_LireMessageBlocCnes` ;
- `PL_LireTCBlocCnes`.

A chaque fois, on retire les actions d'ajout ou de retrait d'en-tête et on remplace la taille de trame normalement contenue dans l'en-tête par **1275** (notre taille fixe de trame AOS, voir partie 3.6.4). Enfin, on modifie CNESLS et le fichier de configuration ION situé sur la VM « centre de contrôle » :

- **CNESLS** : utilisation d'une socket TCP au lieu d'UDP : remplacement du paramètre `SOCK_DGRAM` par `SOCK_STREAM` à la création de celle-ci ;
- **fichier de configuration ION** : changement du port distant indiqué en paramètre de CNESLS dans `ltpadmin`. Ainsi, le span ne pointe plus vers l'adresse IP de la station mais vers l'adresse IP de la passerelle, sur le port *bloc_cnes*.

Ainsi, lors de l'envoi de la première trame AOS, notre centre de contrôle virtuel va naturellement initier une connexion TCP avec la passerelle, puis dialoguer avec elle selon le même protocole (Figure 3.33).

Dialogue avec un satellite simulé

L'objectif est d'utiliser l'une de nos machines virtuelles ION pour simuler la TM reçue du satellite. Pour ce faire, nous utilisons l'interface **SOSIE**, conçue pour permettre à la passerelle provider d'obtenir de la TM d'un simulateur externe. Dans notre cas, le « simulateur » est la VM correspondant au satellite, la TM étant constituée par des pings.

Les échanges entre le simulateur et la passerelle provider se font également sur TCP/IP, à ceci près que c'est la passerelle qui initie la connexion pour la voie TM **et** la voie TC. De

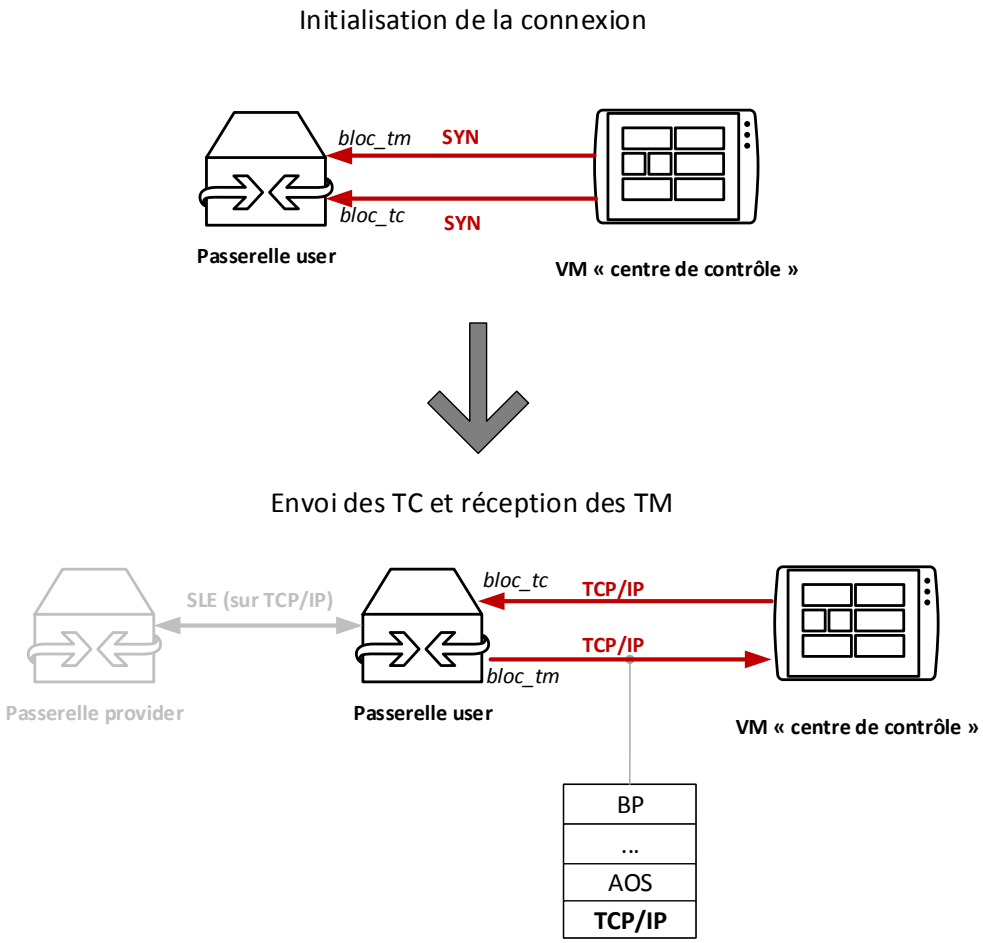


FIGURE 3.33 – Utilisation de l'interface BLOC-CNES côté user

plus, un en-tête contenant la taille de la trame doit être ajouté, il permet la synchronisation des flux entre la passerelle et la machine externe.

On effectue alors les actions suivantes :

- modification de `ltpadmin` sur la VM « satellite » pour que les sessions entrantes et sortantes utilisent respectivement les ports *tc_sosie* et *tm_sosie* ;
- modification de CNESLS pour que la socket utilise le protocole TCP/IP (voir plus haut) ;
- modification de CNESLSO pour qu'il se positionne en attente de demande de connexion au lieu d'en initier une, lorsque le port *tm_sosie* est utilisé et uniquement dans ce cas (la configuration SOSIE ne servant qu'aux tests et pas en conditions réelles) ;
- configuration sur la passerelle provider de l'en-tête SOSIE évoqué plus haut : on définit en-tête de 2 octets, contenant la taille de trame et ne faisant pas partie des données utiles ;
- modification de CNESLSO et CNESLSI pour qu'en mode SOSIE (utilisation des ports *tm_sosie* et *tc_sosie* pour la communication), l'en-tête contenant la taille de trame (1275) soit respectivement ajouté et coupé.

Le couplage entre notre VM « satellite » simulatrice de TM et la passerelle provider correspond alors à ce qui est illustré par la Figure 3.34.

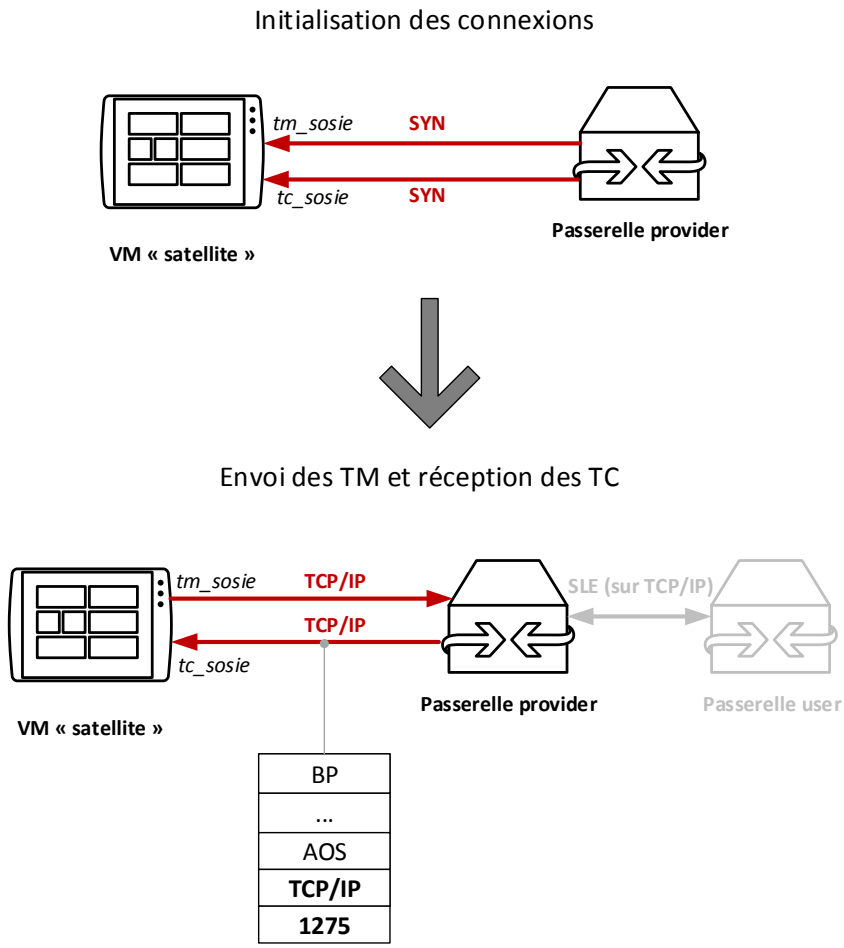


FIGURE 3.34 – Utilisation de l'interface SOSIE côté provider

Conclusion

Cette partie fut longue car la configuration des passerelles SLE est difficile à appréhender. Après une période de *debugging*, nous avons réussi à mettre en place un échange (pings) entre 2 VMs à travers le réseau SLE. La transmission d'un fichier via CFDP a également été réalisée avec succès.

La procédure de mise en place est la suivante :

1. Démarrage du simulateur de satellite (VM1) ;
2. Démarrage de la passerelle provider, qui se connecte au simulateur ;
3. Démarrage de la passerelle user (nécessite les deux premières étapes) ;
4. Démarrage du centre de contrôle (VM2), qui se connecte à la passerelle user.

Ainsi, l'ensemble de la pile protocolaire du segment sol (Figure 2.15) a été étudiée et la faisabilité du couplage des différents éléments la constituant a été démontrée et caractérisée.

3.9 Perspectives (fin août / septembre)

A l'heure où j'écris ce rapport, le stage doit encore se poursuivre durant un mois et demi.

Nous avons donc identifié un certain nombre de points à étudier avant sa fin :

- caractériser la transmission SLE en environnement difficile (lien spatial simulé) ;
- étudier l'implémentation de DTPC présente dans ION. DTPC (Delay-Tolerant Payload Conditioning) est une surcouche à Bundle Protocol, gérant la remise en ordre des bundles ainsi que le contrôle de flux. Elle décharge ainsi l'application de ces responsabilités. Une implémentation existe apparemment dans ION, nous souhaitons la tester ;
- caractériser plus finement la communication LTP sur lien stressé (cf. partie 3.5.2), notamment le délai de « finalisation » de la transmission, lorsque les pertes sont très importantes ;
- expérimenter de nouveaux scénarii pour tester CGR plus en profondeur ;
- étudier l'API ION permettant de construire ses propres applications utilisant la couche Bundle ;
- étudier la possibilité de transporter plusieurs segments LTP dans une même trame AOS (actuellement un seul, avec bourrage si nécessaire).

En fonction de l'avancée globale, des tâches supplémentaires pourront être accomplies afin d'élargir les perspectives accordées par le projet.

Bilan de stage

Ce stage de fin d'études au CNES a contribué à la mise en place d'un démonstrateur DTN bout-en-bout en collaboration avec la NASA, dans un contexte de simulation et d'expérimentation. Après avoir effectué une synthèse des propriétés de l'architecture DTN et de l'existant en termes d'implémentation, j'ai pris en main avec succès la pile logicielle dédiée aux communications DTN spatiales, Interplanetary Overlay Network. Les expérimentations menées ont permis de mettre en évidence les propriétés majeures de DTN, à travers des scénarii correspondant à des communications spatiales réelles. Un travail de *debugging* a abouti à la correction de problèmes de routage rencontrés à l'utilisation de l'algorithme CGR. Une implémentation de la pile protocolaire réellement utilisée en opération a ensuite été réalisée au sein d'ION. Enfin, l'adaptation de ce travail et du système d'interopérabilité SLE a permis un couplage de ces deux éléments, représentatif du futur démonstrateur :

- d'un côté, un centre de contrôle virtuel utilisant CNESLS et une passerelle SLE pour fournir une communication DTN sur AOS bout-en-bout avec le satellite ;
- de l'autre, un simulateur de satellite utilisant le système SOSIE du jeu de passerelles SLE. Il est cohérent avec le centre de contrôle en termes de protocoles réseaux et permettra au CNES de tester sa ou ses solution(s) sol en toute autonomie.

La Figure 3.35 synthétise les étapes d'une communication simulée. Le stage a globalement consisté à réaliser ce chemin bout-en-bout, en tenant compte de l'existant, des standards et des spécifications du projet. L'attention portée sur la modularité des différents composants de la solution technique (CNESLS, codes sources modifiés, ...) lui confère un certain potentiel évolutif. Étant donné que le stage s'inscrivait comme une étape charnière du projet, à mi-chemin entre les évaluations d'ION effectuées en amont et la concrétisation du projet en aval, l'ensemble du travail a dû être réalisé en ayant toujours sa réutilisabilité à l'esprit.

En conclusion, nous avons pu apprécier les caractéristiques de l'architecture DTN (LTP inclus), à savoir sa forte tolérance aux conditions difficiles ainsi que sa capacité à faire converger les réseaux hétérogènes. Nos expérimentations ont mis en valeur l'intérêt de DTN dans un contexte spatial, à condition que l'algorithme de routage soit adapté à ces conditions, ce qui est le cas avec CGR, basé sur la planification de contacts.

Malheureusement, nous nous sommes aussi rendus compte que le passage à l'échelle d'une telle technologie est fortement demandeuse de systèmes de gestion encore absents. En effet, ont essentiellement été étudiés jusqu'à présent les aspects protocolaires purs de DTN, qu'il s'agisse de l'architecture globale ou de son application au spatial. Les lacunes concernent principalement deux points liés au graphe de contacts :

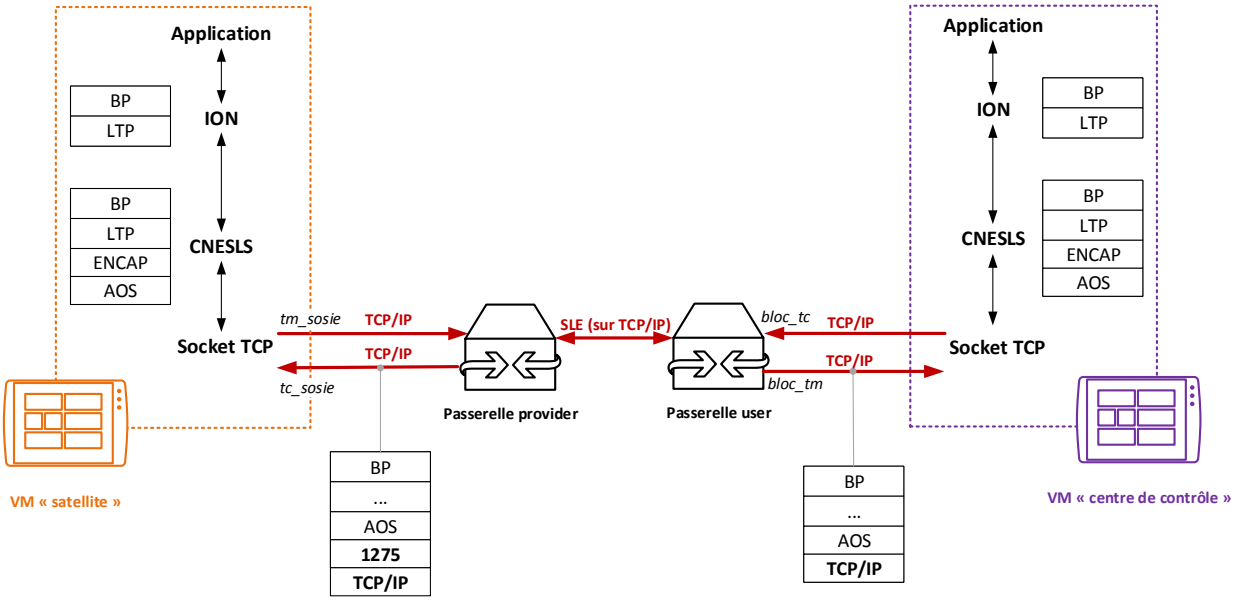


FIGURE 3.35 – Schéma récapitulatif du chemin de communication bout-en-bout

1. **L'inscription des contacts** *a priori* puis l'édition de la topologie au fur et à mesure de la vie des objets spatiaux. Un exemple simpliste : si un satellite est visible une fois par jour par une station, il faut inscrire 730 contacts dans le fichier de configuration pour couvrir une période d'un an. Si l'on passe à plusieurs centaines de satellites et plusieurs dizaines de stations, le nombre de lignes se compte en millions, ce qui n'est pas viable, surtout si chaque objet spatial doit avoir une topologie cohérente avec celle de ses voisins parfois étrangers. Sans compter que si les contacts ne peuvent pas être planifiés à long terme (ce qui est le cas en réalité), il est nécessaire de redémarrer ION à chaque édition du graphe de contacts ;
2. **L'appréhension du temps** : dans ION, les données temporelles des contacts peuvent être indiquées de façon absolue ou relative – par rapport à l'instant de démarrage du démon ION. Si la deuxième solution est plus pratique dans un contexte d'expérimentation (pas besoin de modifier la valeur du temps à chaque nouveau test), elle n'est pas viable dans un contexte opérationnel, car elle nécessite une synchronisation au niveau du démarrage des nœuds du réseau. L'utilisation de CGR en pratique ne peut donc se faire qu'avec des valeurs temporelles absolues, ce qui nécessite là encore une synchronisation globale des nœuds, mais cette fois de leurs horloges internes.

De plus, nous avons noté une redondance certaine au niveau des mécanismes de fiabilisation : acquittements CFDP, Red Part LTP, custody transfert BP, codes correcteurs en couche 2. Bien que ces mécanismes soient situés à des niveaux différents, il serait probablement nécessaire de mieux cadrer leur utilisation opérationnelle, afin de limiter l'overhead.

Ainsi, l'application de DTN au spatial, via CGR, **nécessite de concevoir un système de management fiable** qui supporte le passage à l'échelle de l'algorithme de routage. La standardisation de cet ensemble n'est pas suffisamment mature pour envisager à moyen terme une utilisation globale de DTN sous la forme d'un « Internet Interplanétaire ».

Néanmoins, si l'on garde ces réserves à l'esprit, on peut d'ores et déjà considérer l'intérêt d'une utilisation opérationnelle d'ION dans un contexte de communication spatiale bout-

en-bout (routage complexe mis à l'écart étant donné les problèmes évoqués ci-dessus). Les principaux avantages que je souhaite mettre en avant sont les suivants :

- protocoles « natifs » de tolérance aux délais (LTP, store-and-forward), déchargement des protocoles de niveau 2 des responsabilités liées au transport ;
- interconnexion facilitée des régions hétérogènes : plus besoin de passerelle ni de tunnel ; un seul pivot, ION. De plus, sa structure autorise l'implémentation de nouveaux protocoles, on est donc certains qu'ION peut rester seul « interlocuteur » ;
- déchargement des applications des contraintes liées au réseau et harmonisation de l'utilisation du réseau spatial, comme IP au niveau d'Internet ;
- mise à contribution d'autres objets spatiaux pour augmenter le temps de visibilité virtuelle (sous-entendu, directe ou indirecte). Pour des petits réseaux (en nombre de nœuds), routage dynamique facile à mettre en place et qui utilise efficacement et de façon transparente l'ensemble des objets de communication (stations, satellites relais) ;.
- préparation au futur en disposant déjà de la base sur laquelle s'appuierait une mise en œuvre globale de DTN.

En mettant à contribution un certain nombre d'objets spatiaux, on pourra communiquer avec l'espace non seulement sans interruption (Figure 3.36), mais également avec une fiabilité complète.

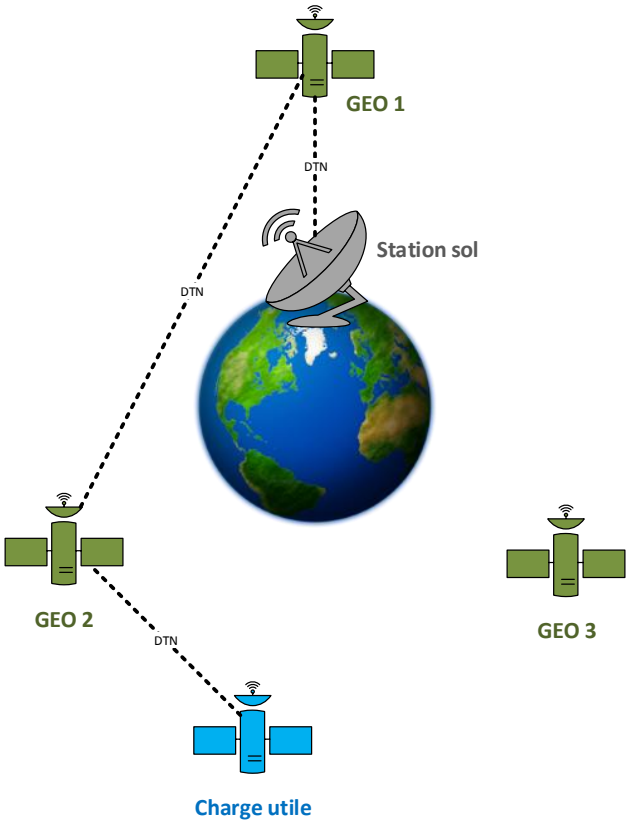


FIGURE 3.36 – Mise à contribution de satellites géostationnaires pour communiquer indirectement avec un satellite

Le travail que j'ai réalisé a participé à l'étude de faisabilité de cette architecture, pour sa future promotion auprès des opérationnels. D'un point de vue personnel, je suis fier d'avoir su mettre à profit mes connaissances pour proposer des solutions techniques pertinentes au sein de ce contexte innovant, et espère pouvoir faire carrière dans le domaine des communications spatiales. Outre les nombreuses compétences que j'ai acquises ou approfondies (développement, implémentation de protocoles réseaux, prise en main de logiciels, architecture réseau, utilisation de notices techniques, analyse de journaux de bord, etc.), ce stage fut une première expérience de longue durée avec des responsabilités d'ingénieur. De façon générale, j'ai pu affiner mon sens didactique, ma capacité à apprécier les délais et les difficultés techniques, et prendre confiance en ma capacité à proposer des idées nouvelles. La nécessité de dialoguer en anglais avec des acteurs externes fut également très formatrice, aussi bien au niveau de la langue pure que d'un point de vue de la réalisation de projet.

En résumé, j'ai dû composer avec différentes ressources matérielles, intellectuelles et humaine, dans un objectif et des intérêts communs. C'est, je crois, l'essence de notre métier.

Bibliographie / Webographie

[1] Delay Tolerant Networking Research Group, Code. 23

[2] Delay Tolerant Networking Research Group, Dtn2Documentation. 31, 32

[3] Internet Interplanétaire, Wikipédia. 23

[4] Kernel-based Virtual Machine, Wikipédia. 27

[5] Space Assigned Number Authority (SANA), Website. 46, 50

[6] AOS Space Data Link Protocol, Blue Book, 2003. 46, 47, 68

[7] Space Link Extension Services, Blue Book, 2005. 55

[8] Encapsulation Service, Blue Book, 2006. 45, 68

[9] Licklider Transmission Protocol for CCSDS, Red Book, 2012. 21, 67

[10] S. Burleigh. ION Design and Operation, 2013. 19, 37, 48

[11] R. Firrincieli C. Caini. Application of Contact Graph Routing to LEO Satellite DTN communications, 2012. 15, 67

[12] A. Belén Uña Cidón. Architecture DTN pour la télédétection. Master's thesis, INP ENSEEIHT, 2011. 19, 26, 34

[13] S. Keshav D. Kroeker. A Tetherless Computing Architecture. 30

[14] T. Le Ravallec D. Schrifve. Les réseaux DTN. 14, 67

[15] C. Gauthier. Architecture DTN fiabilisée pour la télédétection par satellite. Master's thesis, ISAE ENSICA, 2012. 19, 26, 34

[16] K. Jerelt H. El-Damhougy. Interplanetary communications network, interplanetary communications network backbone and method of managing interplanetary communications network, 2006. 11, 67

[17] S. Burleigh K. Scott. "Bundle Protocol Specification", RFC 5050, 2007. 14

[18] S. Farrell M. Ramadas, S. Burleigh. "Licklider Transmission Protocol - Specification", RFC 5326, 2008. 20

[19] D. Lacamera P. Apollonio, C. Caini. Virtualbricks, a virtual testbed manager for scientific research on networking : application to DTN satellite. 27

[20] A. Hooke L. Torgerson R. Durst K. Scott K. Fall H. Weiss V. Cerf, S. Burleigh. "Delay-Tolerant Networking Architecture", RFC 4838, 2007. 16

[21] F. Warthman. Delay-Tolerant Networks, 2003. 12, 14, 15, 16, 67

Liste des illustrations

Liste des acronymes

AOS	Advanced Orbiting Systems Space Data Link Protocol
API	Application Programming Interface
BP	Bundle Protocol
CCSDS	Consultative Committee for Space Data Systems
CGR	Contact Graph Routing
CNES	Centre National d'Études Spatiales
CNESLS	CNES Link Service
DCCP	Datagram Congestion Control Protocol
DNS	Domain Name System
DTN	Delay-Tolerant Network(ing)
DTNRG	Delay-Tolerant Networking Research Group
DTPC	Delay-Tolerant Payload Conditioning
ENCAP	Encapsulation Service
EID	Endpoint Identifier
IANA	Internet Assigned Numbers Authority
ION	Interplanetary Overlay Network
IP	Internet Protocol
IPN	Interplanetary Network Addressing
IRTF	Internet Research Task Force
ISS	International Space Station
JPL	Jet Propulsion Laboratory
LSA	Link State Advertisement
LSI	Link Service Input
LSO	Link Service Output
LTP	Licklider Transmission Protocol
NASA	National Aeronautics and Space Administration
PDU	Protocol Data Unit
RFC	Request For Comments
SANA	Space Assigned Numbers Authority
SDA	Service Data Aggregation
SLE	Space Link Extension
TC	TéléCommande
TCP	Transport Control Protocol
TCPCL	TCP Convergence Layer
TM	TéléMesure
UDP	User Datagram Protocol
UDPLS	UDP Link Service
URI	Uniform Resource Identifier
VM	Virtual Machine

Annexes

Annexe A

En-tête Bundle Protocol (Primary Block)

Version (1 byte)	Bundle Processing Control Flags (SDNV)
Block Length (SDNV)	
Destination Scheme Offset (SDNV)	Destination SSP Offset (SDNV)
Source Scheme Offset (SDNV)	Source SSP Offset (SDNV)
Report-To Scheme Offset (SDNV)	Report-To SSP Offset (SDNV)
Custodian Scheme Offset (SDNV)	Custodian SSP Offset (SDNV)
Creation Timestamp (SDNV)	
Creation Timestamp Sequence Number (SDNV)	
Lifetime (SDNV)	
Dictionary Length (SDNV)	
Dictionary (byte array)	
Fragment Offset (SDNV, optional)	
Application data unit length (SDNV, optional)	

Annexe B

Installation de Virtualbricks

L'installation de Virtualbricks doit se faire par le paquet récupéré sur le dépôt officiel. Les codes sources téléchargeables sur Internet (Launchpad notamment) ne sont pas à jour et truffés de bugs.

Installation (sur Debian/Ubuntu) :

1. Ajouter les lignes suivantes au fichier `/etc/apt/sources.list` :

   ```
   deb http://cnrl.deis.unibo.it/repo/ RELEASE main
   deb-src http://cnrl.deis.unibo.it/repo/ RELEASE main
   ```

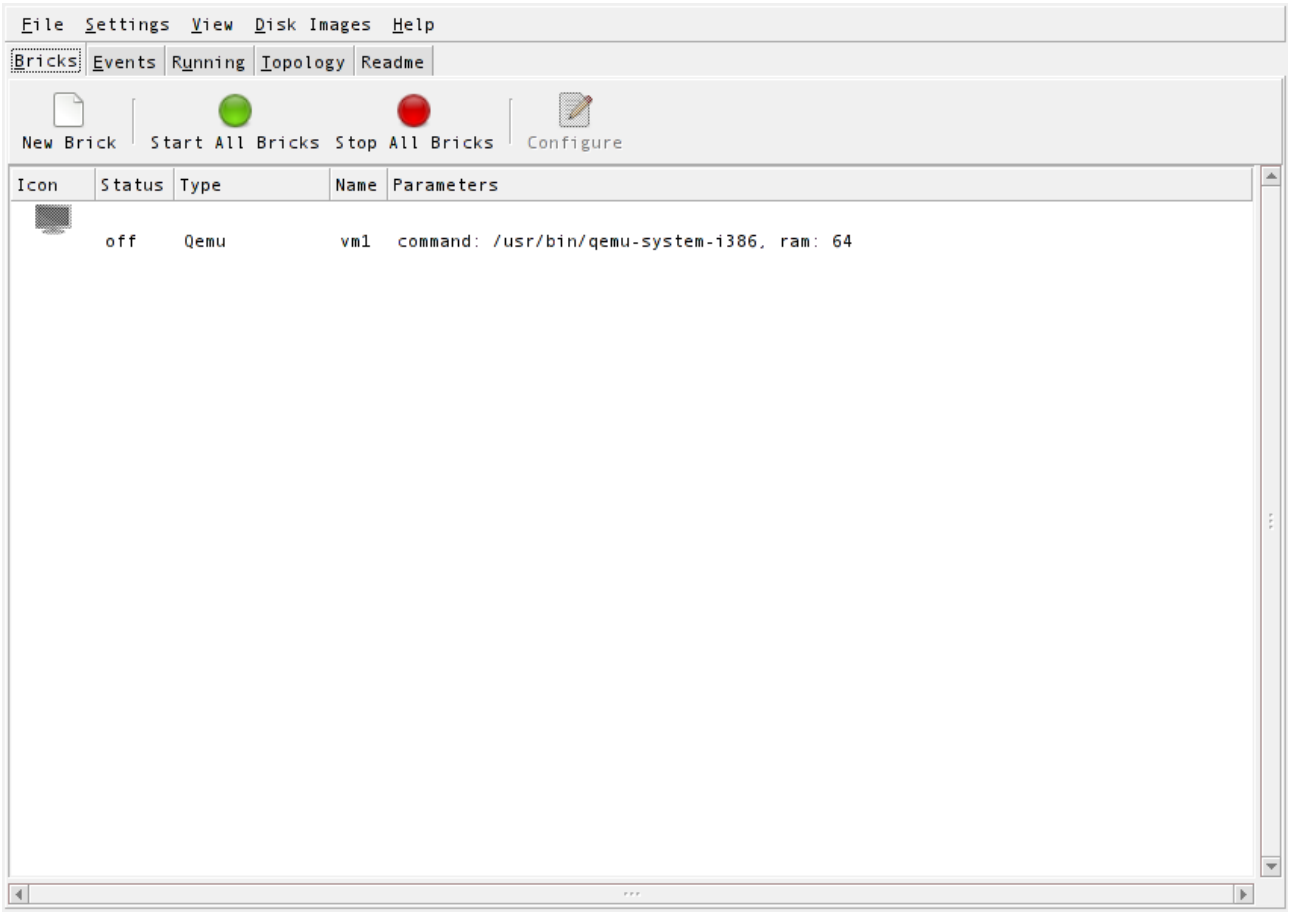
   Remplacer `RELEASE` par le nom de la distribution. Les distributions actuellement supportées sont les suivantes : Debian wheezy, Debian sid, Ubuntu precise, Ubuntu quantal, Ubuntu raring, Ubuntu saucy.
2. Pour télécharger la clé publique GPG correspondant au dépôt (optionnel), exécuter la commande suivante : `wget -O - http://cnrl.deis.unibo.it/repo/signing_key.pub | sudo apt-key add -`
3. Mettre à jour la liste des dépôts : `sudo apt-get update`
4. Installer le paquet Virtualbricks : `sudo apt-get install virtualbricks`

Je conseille de toujours lancer le logiciel en ligne de commande et de le quitter avec la commande `quit`, dans le shell Virtualbricks.

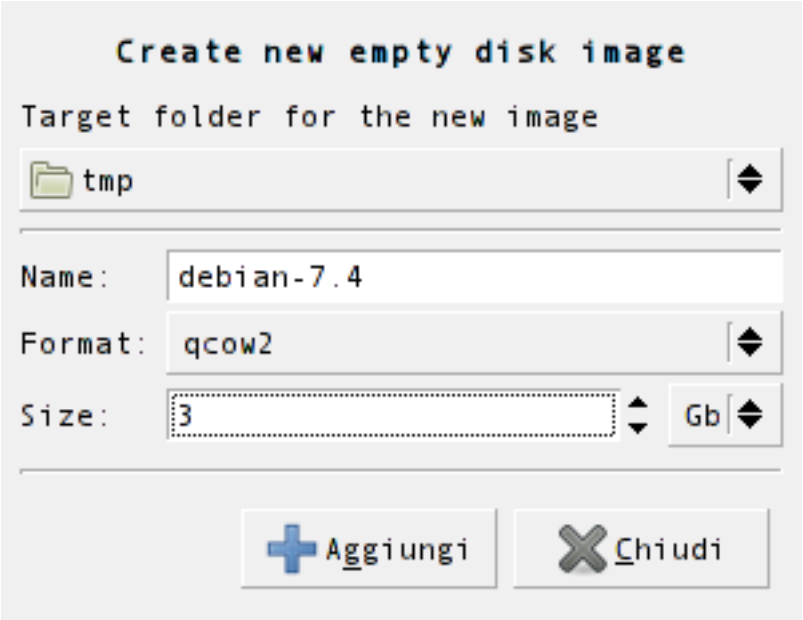
Les pages suivantes constituent un tutoriel d'installation d'une VM et de sa configuration réseau. Ils m'ont été transmis par M. Marco Giusti, développeur de Virtualbricks. Je me permets de les joindre à cette annexe afin de regrouper l'ensemble des documents.

Tutoriel 1

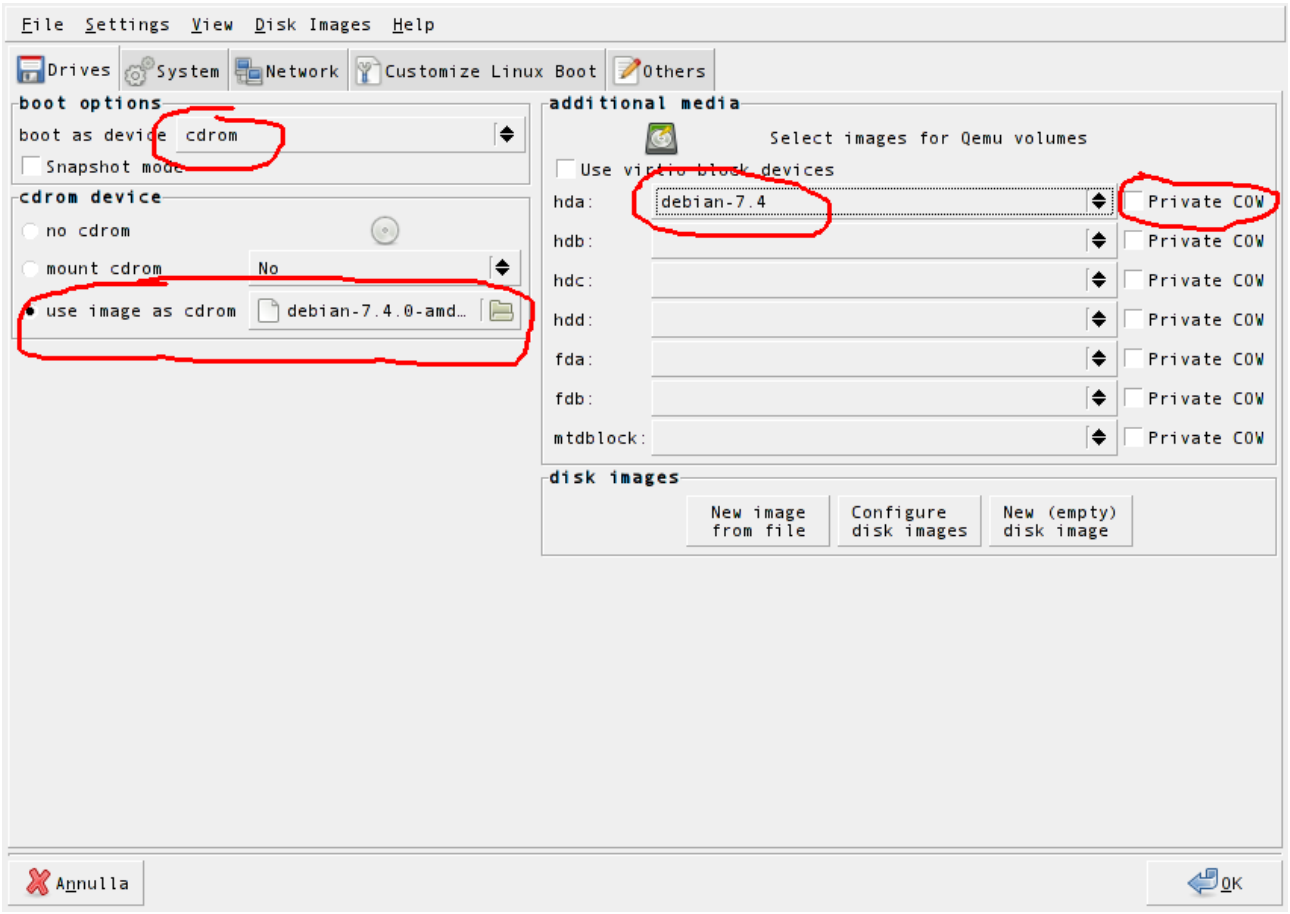
Création et configuration d'une nouvelle machine virtuelle et installation ex-novo de Debian



Créez un nouveau projet et ajoutez une nouvelle machine virtuelle.

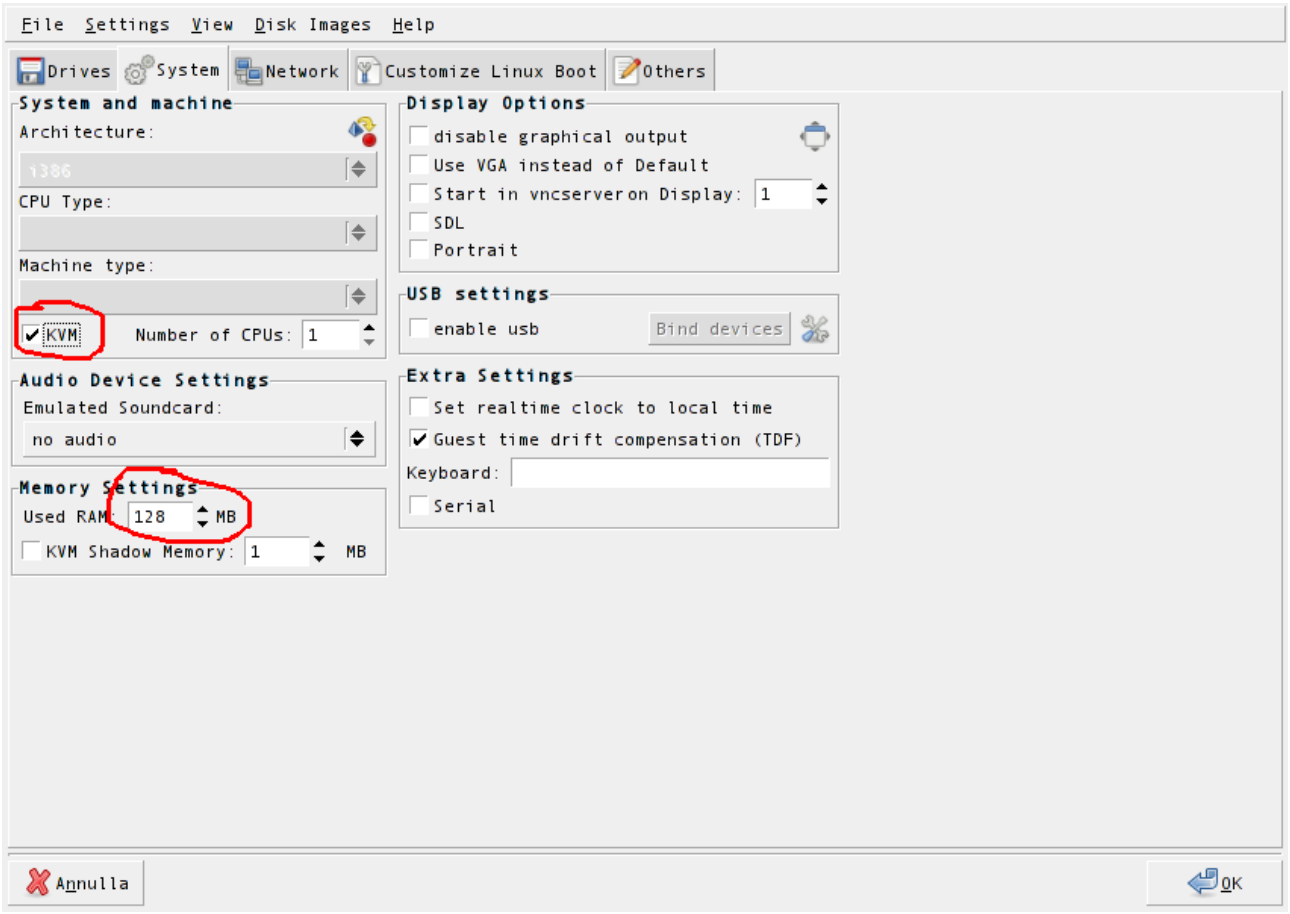


Ajoutez un nouveau disque. Sélectionnez le format « qcow2 ».



Organisez les réglages comme ci-dessus. Ces réglages vaudront seulement pour le premier lancement. Ceux-ci permettront de lancer le système d'installation de Debian.

Dans le réglage « hda », il faut mettre le disque précédemment créer.

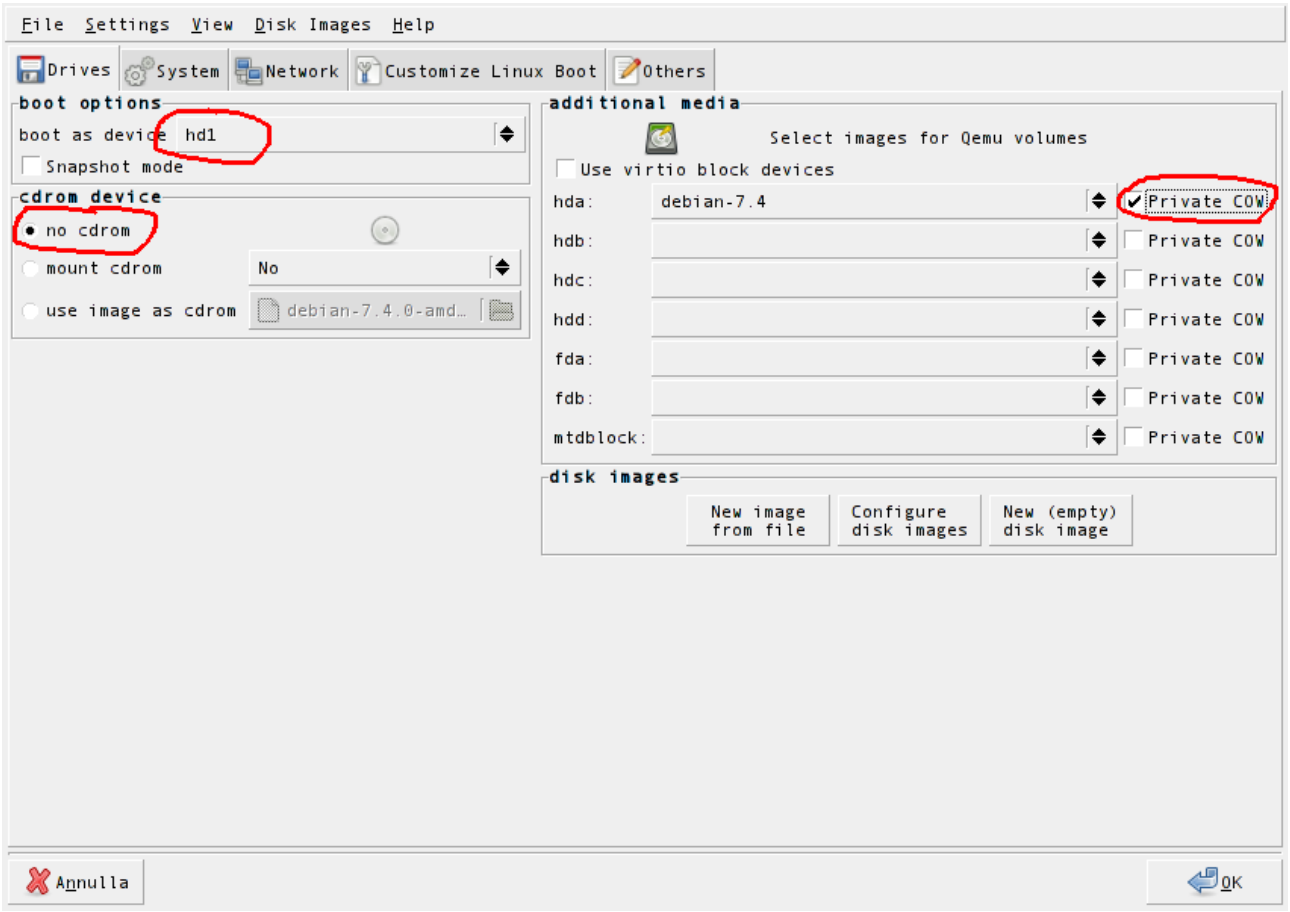


Sélectionnez le réglage « kvm ». Si un message d'erreur apparaît, s'assurer que le module du kernel « kvm-intel » ou « kvm-amd » a été chargé.
La quantité de mémoire nécessaire pour l'installation est relativement basse, la réglez selon la nécessité.
À ce niveau les réglages sont terminés.

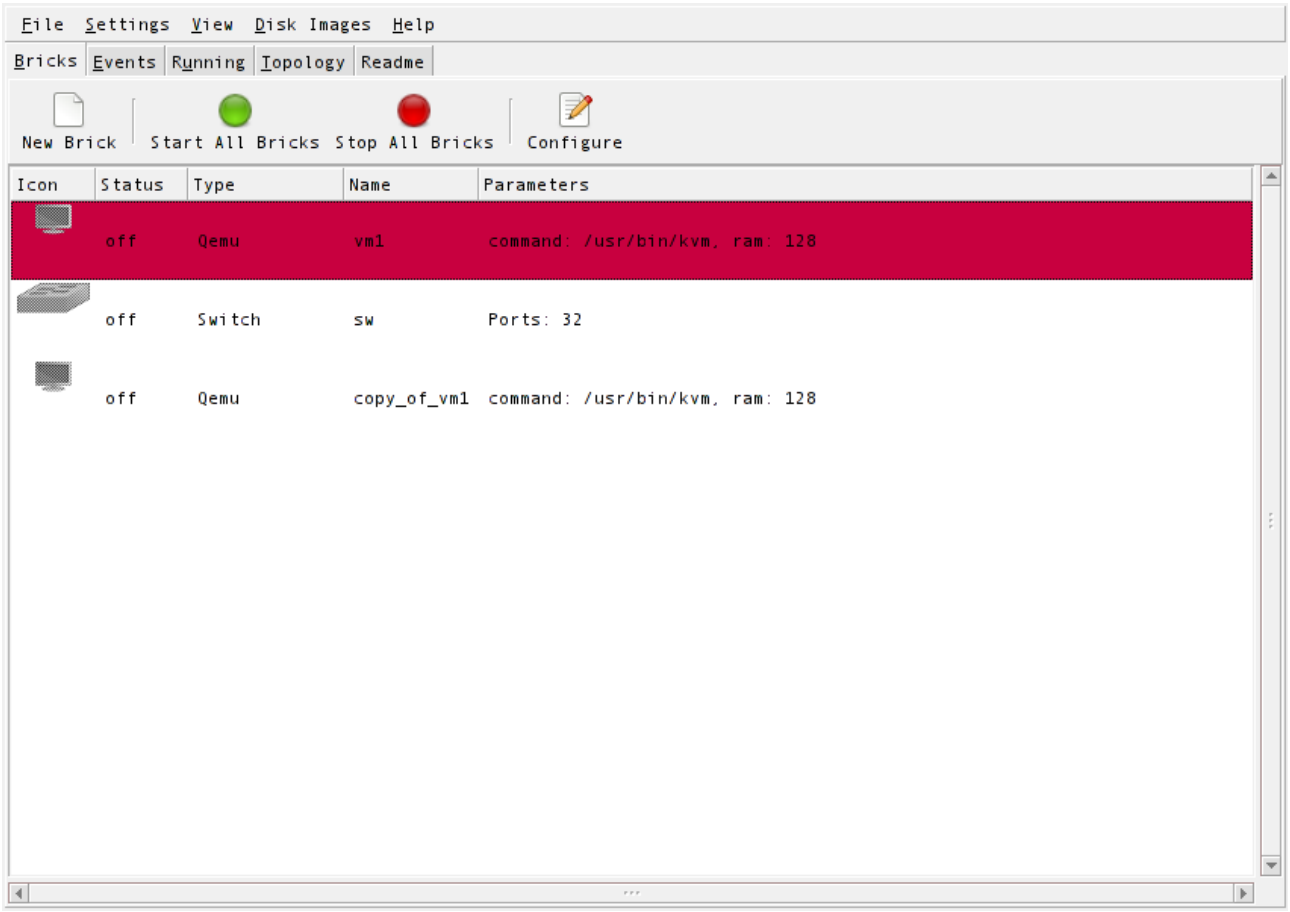
Procédez avec l'installation normale du système d'exploitation.

Tutoriel 2

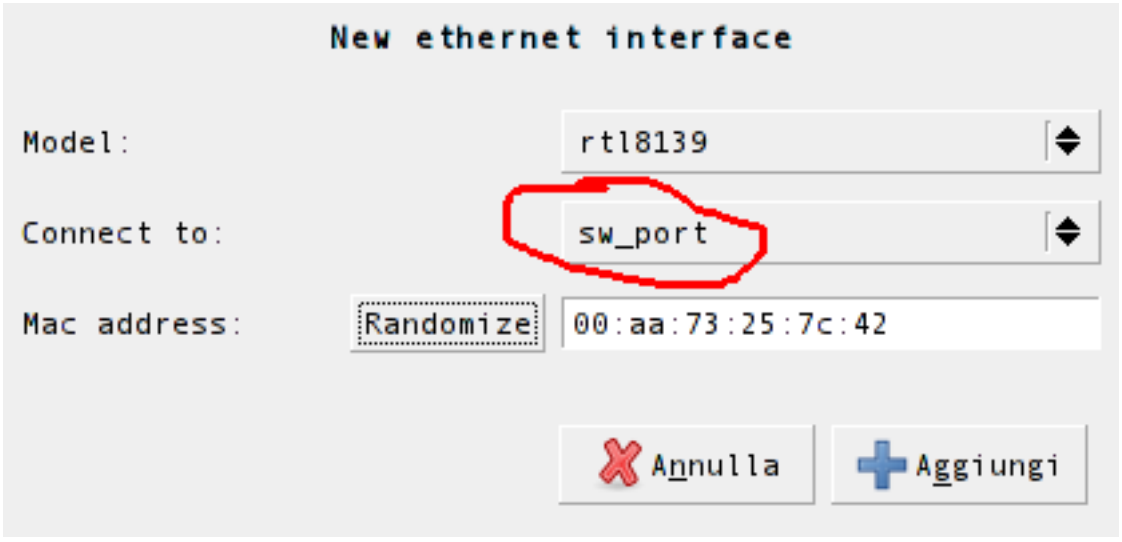
Création d'un simple réseau entre deux machines virtuelles

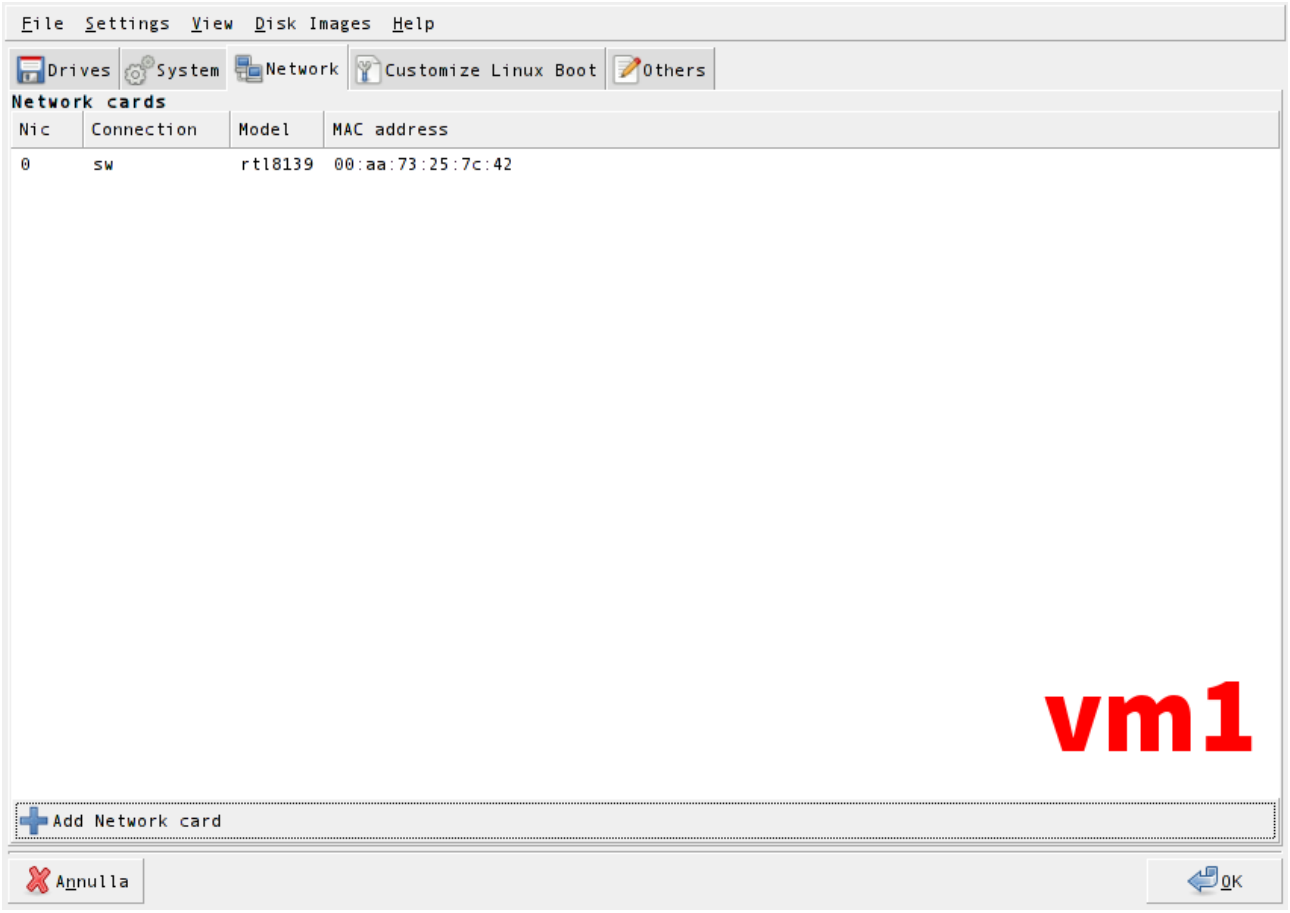


Partez du projet précédemment créé.
Organisez les réglages comme ci-dessus. A ce niveau, le disque d'installation de Debian n'est plus nécessaire.
Sélectionnez le réglage « private cow » pour éviter que deux machines virtuelles entrent en conflit et écrivent sur le même fichier.

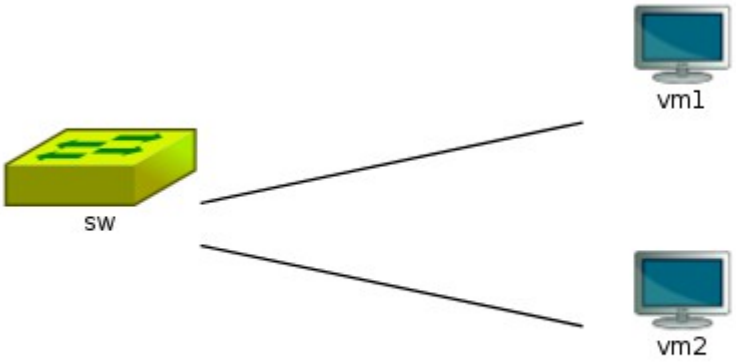


Ajoutez un switch et copiez la machine virtuelle.

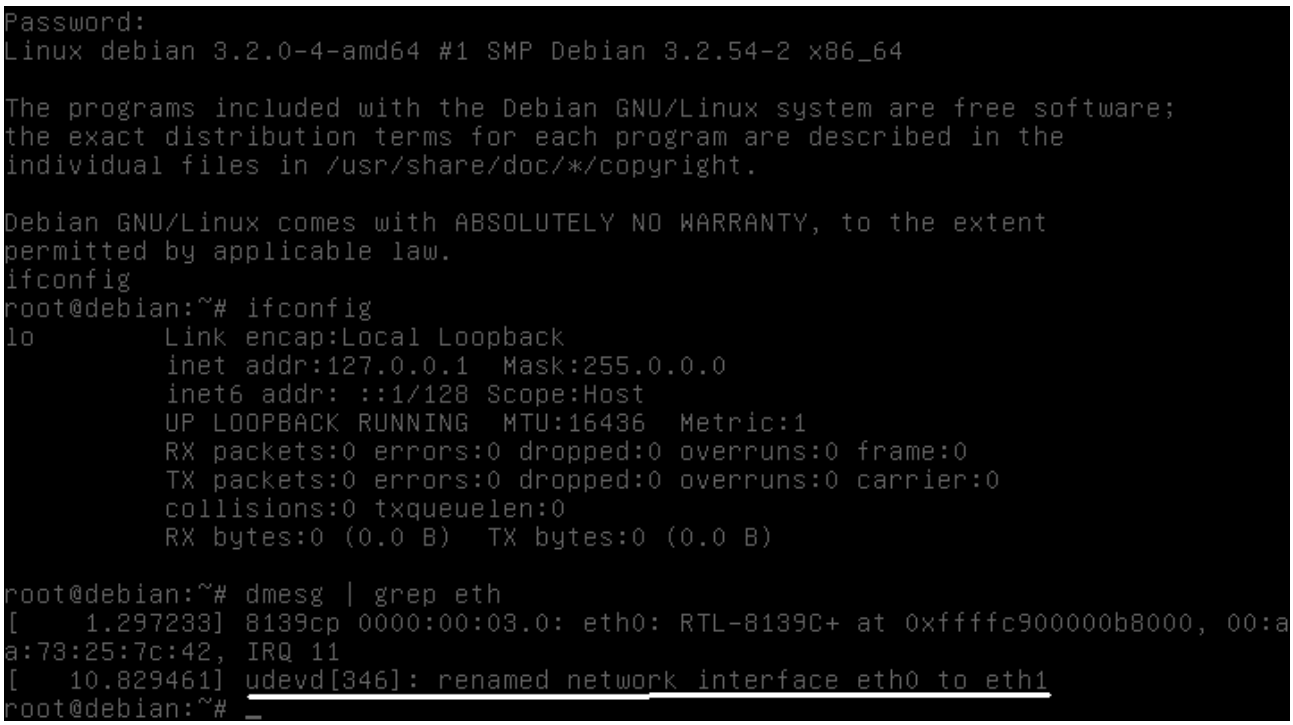




Ajoutez une nouvelle carte réseau à la première machine virtuelle et connectez la au switch.
Répétez l'opération avec la deuxième machine virtuelle.



La topologie devrait se montrer comme l'image ci-dessus.
Mettez en marche les machines virtuelles et préparez-vous à la configuration du réseau.



Faites attention car le logiciel « udevd » pourrait renommer la carte réseau.
Modifiez le fichier « /etc/network/interfaces » comme ci-dessous :

```
auto lo
iface lo inet loopback

auto eth1
iface eth1 inet static
address 192.168.1.1
netmask 255.255.255.0
```

Activez la carte réseau que vous venez de configurer à travers les commandes suivantes:

```
ifup eth1
```

Ajoutez la ligne suivante au fichier « /etc/hosts » :

```
192.168.1.2 vm2
```

Installez le logiciel « openssh-server » :

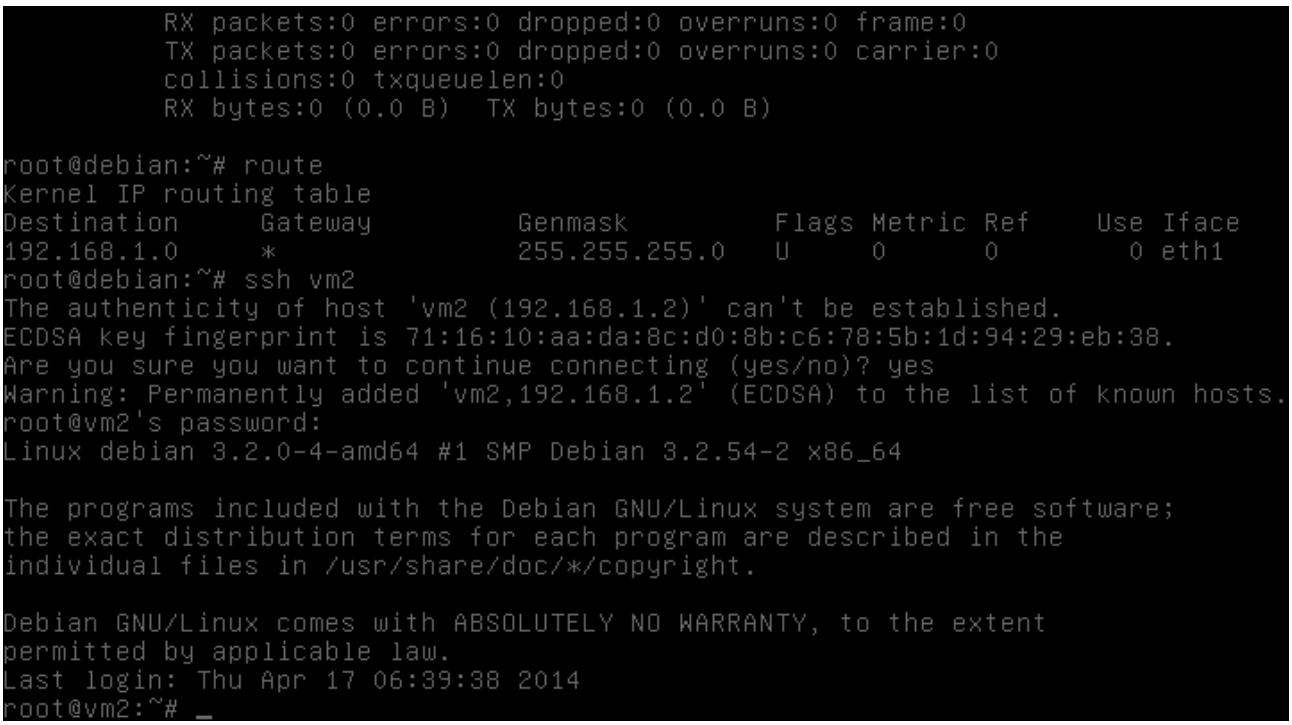
```
apt-get install openssh-server
```

Renommez la machine virtuelle en modifiant le fichier « /etc/hostname ».

Répétez toutes ces opérations pour la deuxième machine virtuelle en faisant attention de changer l'adresse IP et les références à la première machine.

```
root@debian:~# ifconfig
eth1      Link encap:Ethernet  HWaddr 00:aa:73:25:7c:42
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::2aa:73ff:fe25:7c42/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:468 (468.0 B)
          Interrupt:11 Base address:0x8000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@debian:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth1
root@debian:~# _
```

```
root@debian:~# ifconfig
eth1      Link encap:Ethernet  HWaddr 00:aa:3e:d2:8b:8a
          inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::2aa:3eff:fed2:8b8a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:238 (238.0 B)
          Interrupt:11 Base address:0xa000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@debian:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth1
root@debian:~# _
```



Maintenant les deux machines virtuelles sont configurées et communiquent entre elles.
Au prochain redémarrage les machines virtuelles seront déjà configurées et prêtes à communiquer entre elles.

Annexe C

Structure d'un fichier de configuration DTN2

Daemon Console Configuration

...

Storage configuration

configuration du type de stockage
configuration des dossiers de stockage persistant
...

Routing configuration

configuration du protocole de routage

```
route set type [ static | flood | … ]
```

configuration de l'EID du nœud local

```
route local_eid ''dtn://name.dtn''
```

...

TCP convergence layer configuration (ou autre couche de convergence)

ajout des interfaces

```
interface add [name] [CL]
```

ex : `interface add tcp0 tcp`

ajout des liaisons

```
link add <name> <nexthop> <type> <clayer> [args…]
```

ex : `link add myLink 10.0.0.102 ONDEMAND tcp`

configuration du routage statique

```
route add <dest> <link|peer>
```

ex : `route add dtn://remote_node.dtn/service myLink`

...

Autres paragraphes de configuration (non utilisés par défaut)

...

Annexe D

Fichiers de configuration ION (basculement du satellite)

ION1 (satellite)

```
# ion1 configuration file
# run with: ionstart -I ion.rc

## begin ionadmin
1 1 ""
e 1
s

#Link 1-2
a contact +0 +30 2 1 512000
a contact +0 +30 1 2 2000000
a contact +60 +90 2 1 512000
a contact +60 +90 1 2 2000000
a contact +120 +150 2 1 512000
a contact +120 +150 1 2 2000000
a contact +180 +210 2 1 512000
a contact +180 +210 1 2 2000000
a contact +240 +270 2 1 512000
a contact +240 +270 1 2 2000000
a contact +300 +36000 2 1 512000
a contact +300 +36000 1 2 2000000

#Link 1-3
a contact +30 +60 3 1 512000
a contact +30 +60 1 3 2000000
a contact +90 +120 3 1 512000
a contact +90 +120 1 3 2000000
a contact +150 +180 3 1 512000
a contact +150 +180 1 3 2000000
a contact +210 +240 3 1 512000
a contact +210 +240 1 3 2000000
a contact +270 +300 3 1 512000
a contact +270 +300 1 3 2000000

#Link 2-4
a contact +0 +36000 2 4 10000000
a contact +0 +36000 4 2 10000000

#Link 3-4
a contact +0 +36000 3 4 10000000
a contact +0 +36000 4 3 10000000

#Loopback
a contact +0 +36000 1 1 10000000
a contact +0 +36000 2 2 10000000
a contact +0 +36000 3 3 10000000
a contact +0 +36000 4 4 10000000

#Ranges
a range +0 +36000 1 2 3
a range +0 +36000 1 3 3
a range +0 +36000 2 4 1
a range +0 +36000 3 4 1
a range +0 +36000 1 1 1
a range +0 +36000 2 2 1
a range +0 +36000 3 3 1
a range +0 +36000 4 4 1
```

```
m production 10000000
m consumption 10000000
## end ionadmin
################################################################################
#
## begin ltpadmin
1 400
e 1

a span 2 200 200 1400 1400 1 'udplso 192.168.0.2:1113'
a span 3 200 200 1400 1400 1 'udplso 192.168.0.3:1113'

s 'udplsi 192.168.0.1:1113'
## end ltpadmin
################################################################################
#
## begin bpadmin
1
e 1

a scheme ipn 'ipnfw' 'ipnadminep'

a endpoint ipn:1.0 x
a endpoint ipn:1.1 x
a endpoint ipn:1.2 x
a endpoint ipn:1.64 x
a endpoint ipn:1.65 x

a protocol ltp 1400 100
a protocol tcp 1400 100

a induct ltp 1 ltpcli
a induct tcp 192.168.0.1:4556

a outduct tcp 192.168.0.1:4556 tcpclo
a outduct ltp 2 ltpclo
a outduct ltp 3 ltpclo

s
## end bpadmin
################################################################################
#
## begin cfdpadmin
1
s 'bputa'
e 1
## end cfdpadmin
################################################################################
#
## begin ipnadmin
e 1
a plan 1 tcp/192.168.0.0:4556
a plan 2 ltp/2
a plan 3 ltp/3
## end ipnadmin
```

ION2 (station sol)

```
# ion2 loopback configuration file
# run with: ionstart -I ion.rc

## begin ionadmin
1 2 ""
e 1
s

#Link 1-2
a contact +0 +30 2 1 512000
a contact +0 +30 1 2 2000000
a contact +60 +90 2 1 512000
a contact +60 +90 1 2 2000000
a contact +120 +150 2 1 512000
a contact +120 +150 1 2 2000000
a contact +180 +210 2 1 512000
a contact +180 +210 1 2 2000000
a contact +240 +270 2 1 512000
a contact +240 +270 1 2 2000000
a contact +300 +36000 2 1 512000
a contact +300 +36000 1 2 2000000

#Link 1-3
a contact +30 +60 3 1 512000
a contact +30 +60 1 3 2000000
a contact +90 +120 3 1 512000
a contact +90 +120 1 3 2000000
a contact +150 +180 3 1 512000
a contact +150 +180 1 3 2000000
a contact +210 +240 3 1 512000
a contact +210 +240 1 3 2000000
a contact +270 +300 3 1 512000
a contact +270 +300 1 3 2000000

#Link 2-4
a contact +0 +36000 2 4 10000000
a contact +0 +36000 4 2 10000000

#Link 3-4
a contact +0 +36000 3 4 10000000
a contact +0 +36000 4 3 10000000

#Loopback
a contact +0 +36000 1 1 10000000
a contact +0 +36000 2 2 10000000
a contact +0 +36000 3 3 10000000
a contact +0 +36000 4 4 10000000

#Ranges
a range +0 +36000 1 2 3
a range +0 +36000 1 3 3
a range +0 +36000 2 4 1
a range +0 +36000 3 4 1
a range +0 +36000 1 1 1
a range +0 +36000 2 2 1
a range +0 +36000 3 3 1
a range +0 +36000 4 4 1
```

```
m production 10000000
m consumption 10000000
## end ionadmin
################################################################################
#
## begin ltpadmin
1 200
e 1

a span 1 200 200 1400 1400 1 'udplso 192.168.0.1:1113'

s 'udplsi 192.168.0.2:1113'
## end ltpadmin
################################################################################
#
## begin bpadmin
1
e 1

a scheme ipn 'ipnfw' 'ipnadminep'

a endpoint ipn:2.0 x
a endpoint ipn:2.1 x
a endpoint ipn:2.2 x
a endpoint ipn:2.64 x
a endpoint ipn:2.65 x

a protocol ltp 1400 100
a protocol tcp 1400 100

a induct ltp 2 ltpcli
a induct tcp 192.168.0.2:4556 tcpcli
a induct tcp 192.168.1.2:4556 tcpcli

a outduct ltp 1 ltpclo
a outduct tcp 192.168.0.2:4556 tcpclo
a outduct tcp 192.168.1.4:4556 tcpclo

s
## end bpadmin
################################################################################
#
## begin cfdpadmin
1
s 'bputa'
e 1
## end cfdpadmin
################################################################################
#
## begin ipnadmin
e 1
a plan 1 ltp/1
a plan 2 tcp/192.168.0.2:4556
a plan 4 tcp/192.168.1.4:4556
## end ipnadmin
```

ION3 (station sol)

```
# ion3 loopback configuration file
# run with: ionstart -I ion.rc

## begin ionadmin
1 3 ""
e 1
s

#Link 1-2
a contact +0 +30 2 1 512000
a contact +0 +30 1 2 2000000
a contact +60 +90 2 1 512000
a contact +60 +90 1 2 2000000
a contact +120 +150 2 1 512000
a contact +120 +150 1 2 2000000
a contact +180 +210 2 1 512000
a contact +180 +210 1 2 2000000
a contact +240 +270 2 1 512000
a contact +240 +270 1 2 2000000
a contact +300 +36000 2 1 512000
a contact +300 +36000 1 2 2000000

#Link 1-3
a contact +30 +60 3 1 512000
a contact +30 +60 1 3 2000000
a contact +90 +120 3 1 512000
a contact +90 +120 1 3 2000000
a contact +150 +180 3 1 512000
a contact +150 +180 1 3 2000000
a contact +210 +240 3 1 512000
a contact +210 +240 1 3 2000000
a contact +270 +300 3 1 512000
a contact +270 +300 1 3 2000000

#Link 2-4
a contact +0 +36000 2 4 10000000
a contact +0 +36000 4 2 10000000

#Link 3-4
a contact +0 +36000 3 4 10000000
a contact +0 +36000 4 3 10000000

#Loopback
a contact +0 +36000 1 1 10000000
a contact +0 +36000 2 2 10000000
a contact +0 +36000 3 3 10000000
a contact +0 +36000 4 4 10000000

#Ranges
a range +0 +36000 1 2 3
a range +0 +36000 1 3 3
a range +0 +36000 2 4 1
a range +0 +36000 3 4 1
a range +0 +36000 1 1 1
a range +0 +36000 2 2 1
a range +0 +36000 3 3 1
a range +0 +36000 4 4 1
```

```
m production 10000000
m consumption 10000000
## end ionadmin
################################################################################
#
## begin ltpadmin
1 200
e 1

a span 1 200 200 1400 1400 1 'udplso 192.168.0.1:1113'

s 'udplsi 192.168.0.3:1113'
## end ltpadmin
################################################################################
#
## begin bpadmin
1
e 1

a scheme ipn 'ipnfw' 'ipnadminep'

a endpoint ipn:3.0 x
a endpoint ipn:3.1 x
a endpoint ipn:3.2 x
a endpoint ipn:3.64 x
a endpoint ipn:3.65 x

a protocol ltp 1400 100
a protocol tcp 1400 100

a induct ltp 3 ltpcli
a induct tcp 192.168.0.3:4556 tcpcli
a induct tcp 192.168.1.3:4556 tcpcli

a outduct ltp 1 ltpclo
a outduct tcp 192.168.0.3:4556 tcpclo
a outduct tcp 192.168.1.4:4556 tcpclo

s
## end bpadmin
################################################################################
#
## begin cfdpadmin
1
s 'bputa'
e 1
## end cfdpadmin
################################################################################
#
## begin ipnadmin
e 1
a plan 1 ltp/1
a plan 3 tcp/192.168.0.3:4556
a plan 4 tcp/192.168.1.4:4556
## end ipnadmin
```

ION4 (centre de contrôle)

```
# ion4 configuration file
# run with: ionstart -I ion.rc

## begin ionadmin
1 4 ""
e 1
s

Link 1-2
a contact +0 +30 2 1 512000
a contact +0 +30 1 2 2000000
a contact +60 +90 2 1 512000
a contact +60 +90 1 2 2000000
a contact +120 +150 2 1 512000
a contact +120 +150 1 2 2000000
a contact +180 +210 2 1 512000
a contact +180 +210 1 2 2000000
a contact +240 +270 2 1 512000
a contact +240 +270 1 2 2000000
a contact +300 +36000 2 1 512000
a contact +300 +36000 1 2 2000000

#Link 1-3
a contact +30 +60 3 1 512000
a contact +30 +60 1 3 2000000
a contact +90 +120 3 1 512000
a contact +90 +120 1 3 2000000
a contact +150 +180 3 1 512000
a contact +150 +180 1 3 2000000
a contact +210 +240 3 1 512000
a contact +210 +240 1 3 2000000
a contact +270 +300 3 1 512000
a contact +270 +300 1 3 2000000

#Link 2-4
a contact +0 +36000 2 4 10000000
a contact +0 +36000 4 2 10000000

#Link 3-4
a contact +0 +36000 3 4 10000000
a contact +0 +36000 4 3 10000000

#Loopback
a contact +0 +36000 1 1 10000000
a contact +0 +36000 2 2 10000000
a contact +0 +36000 3 3 10000000
a contact +0 +36000 4 4 10000000

#Ranges
a range +0 +36000 1 2 3
a range +0 +36000 1 3 3
a range +0 +36000 2 4 1
a range +0 +36000 3 4 1
a range +0 +36000 1 1 1
a range +0 +36000 2 2 1
a range +0 +36000 3 3 1
a range +0 +36000 4 4 1
```

```
m production 10000000
m consumption 10000000
## end ionadmin
################################################################################
#
## begin ltpadmin

## end ltpadmin
################################################################################
#
## begin bpadmin
1
e 1

a scheme ipn 'ipnfw' 'ipnadminep'

a endpoint ipn:4.0 x
a endpoint ipn:4.1 x
a endpoint ipn:4.2 x
a endpoint ipn:4.64 x
a endpoint ipn:4.65 x

a protocol tcp 1400 100

a induct tcp 192.168.1.4:4556 tcpcli

a outduct tcp 192.168.1.2:4556 tcpclo
a outduct tcp 192.168.1.3:4556 tcpclo
a outduct tcp 192.168.1.4:4556 tcpclo

s
## end bpadmin
################################################################################
#
## begin cfdpadmin
1
s 'bputa'
e 1
## end cfdpadmin
################################################################################
#
## begin ipnadmin
e 1
a plan 2 tcp/192.168.1.2:4556
a plan 3 tcp/192.168.1.3:4556
a plan 4 tcp/192.168.&.4:4556
## end ipnadmin
```

Annexe E

Fichiers de configuration ION (passerelles SLE)

Fichiers de configuration correspondant à l'expérimentation illustrée par la Figure 3.35 (passage par réseau SLE) ; à noter qu'un mot de synchronisation de 4 octets est ajouté à la pile protocolaire entre le satellite et la passerelle SLE provider.

Les valeurs des ports actuels dans CNESLS sont les suivantes :

- SosieTc : 10001 ;
- SosieTm : 10000 ;
- BlocCnesTc : 3020 ;
- BlocCnesTm : 3011.

Enfin, le fichier de configuration SOSIE situé sur la passerelle provider est :

```
ENTETE 2 NON
LONGUEUR 0 16 0
```

ION1 (satellite)

```
# ion1 configuration file
# run with: ionstart -I ion.rc

## begin ionadmin
1 1 ""
e 1
s

#Link 3-1
a contact +0 +36000 3 1 200000
a contact +0 +36000 1 3 200000

#Link 1-2
a contact +0 +36000 2 1 512000
a contact +0 +36000 1 2 2000000

#Link 2-4
a contact +0 +36000 2 4 10000000
a contact +0 +36000 4 2 10000000

#Ranges
a range +0 +36000 1 1 1
a range +0 +36000 2 2 1
a range +0 +36000 3 3 1
a range +0 +36000 4 4 1
a range +0 +36000 1 2 1
a range +0 +36000 1 3 1
a range +0 +36000 2 4 1

m production 10000000
m consumption 10000000
## end ionadmin
##############################################################################
#
## begin ltpadmin
1 400
e 1

a span 2 200 200 1200 1200 1 'cneslso 132.149.40.190:10000'

s 'cneslsi 132.149.40.111:10001'

## end ltpadmin
##############################################################################
#
## begin bpadmin
1
e 1

a scheme ipn 'ipnfw' 'ipnadminep'

a endpoint ipn:1.0 x
a endpoint ipn:1.1 x
a endpoint ipn:1.2 x
a endpoint ipn:1.64 x
a endpoint ipn:1.65 x

a protocol ltp 1275 50
a protocol tcp 1400 100

a induct ltp 1 ltpcli
```

```
a induct tcp 132.149.40.111:4556 tcpcli

a outduct ltp 2 ltpclo
a outduct tcp 132.149.40.111:4556 tcpclo

s
## end bpadmin
################################################################################
#
## begin cfdpadmin
1
s 'bputa'
e 1
w 1
## end cfdpadmin
################################################################################
#
## begin ipnadmin
e 1
a plan 2 ltp/2
a plan 1 tcp/132.149.40.111:4556
## end ipnadmin
```

ION2 (centre de contrôle)

```
# ion2 configuration file
# run with: ionstart -I ion.rc

## begin ionadmin
1 2 ""
e 1
s

#Link 1-2
a contact +0 +36000 2 1 512000
a contact +0 +36000 1 2 200000

#Ranges
a range +0 +36000 1 2 1
a range +0 +36000 2 2 1

m production 10000000
m consumption 10000000
## end ionadmin
###############################################################################
#
## begin ltpadmin
1 400
e 1

a span 1 200 200 1275 1275 1 'cneslso 132.149.40.127:3020'

s 'cneslsi 132.149.40.127:3011'
## end ltpadmin
###############################################################################
#
## begin bpadmin
1
e 1

a scheme ipn 'ipnfw' 'ipnadminep'

a endpoint ipn:2.0 x
a endpoint ipn:2.1 x
a endpoint ipn:2.2 x
a endpoint ipn:2.64 x
a endpoint ipn:2.65 x

a protocol ltp 1275 50
a protocol tcp 1400 100

a induct ltp 2 ltpcli
a induct tcp 132.149.40.119:4556 tcpcli

a outduct ltp 1 ltpclo
a outduct tcp 132.149.40.119:4556 tcpclo

s
## end bpadmin
###############################################################################
#
## begin cfdpadmin
1
s 'bputa'
e 1
w 1
```

```
## end cfdpadmin
################################################################################
#
## begin ipnadmin
e 1
a plan 1 ltp/1
a plan 2 tcp/132.149.40.119:4556
## end ipnadmin
```

Annexe F

Serveur Git d'hébergement du code

Un serveur Git est disponible sur le PC technique installé au laboratoire. Il contient le code à jour comprenant les sources d'ION (version 3.2.0) ainsi que :

- les modifications apportées par le JPL à `cgrfetch` (application d'un patch `cgrfetch.diff`) ;
- les modifications apportées par le JPL à `libcgr.c` (cf. partie 3.5.2) : remplacement du fichier C ;
- le dossier `ltp/cnes` contenant CNESLS.

Seules les sources sont enregistrées, les fichiers compilés ou temporaires sont ignorés. Le dossier `ion.git` est un dossier *bare* (ne contient pas les fichiers effectifs mais seulement l'historique des changements) contenu dans le répertoire personnel d'un utilisateur nommé `git`, situé dans `/var/git`. Il est accessible en mode *read-only* (transmission du code au sein du CNES) et en mode *read-and-write* (pour les développeurs) :

- **read-only** : `git://<ip_pc_technique>/ion.git` (protocole Git) ;
- **read-and-write** : `ssh://git@<ip_pc_technique>/var/git/ion.git` (protocole SSH).

L'authentification SSH est effectuée par mot de passé (celui de l'utilisateur `git`), mais un système de clés RSA pourra être mis en place si besoin.

Enfin, le shell de l'utilisateur `git` est restreint au shell Git : il est donc impossible d'effectuer des actions sur le système, en dehors des commandes Git relatives à ce projet.

Pour **compiler le code**, exécuter `./configure && make && sudo make install`. En cas de problème avec les versions d'`aclocal`, exécuter la commande `autoreconf --force --install` à la racine des sources (nécessite les paquets `automake` et `libtool`).

Annexe G

Études supplémentaires

I Délais importants avec LTP (ordre de la seconde), même sur liens non stressés

Cela est dû au mécanisme d'aggrégation de LTP dans ION, dont la limite **temporelle** ne peut être inférieure à 1 seconde (on spécifie également une limite **en octets** dans le fichier de configuration).

II Utilisation de CNESLS sur plusieurs sous-réseaux

On ne peut *bind* qu'une seule interface d'entrée (CNESLSI) dans le fichier de configuration ION. Ainsi, si l'on souhaite configurer un nœud intermédiaire, possédant plusieurs interfaces dans des sous-réseaux différents, il est nécessaire :

- soit de *bind* CNESLSI sur l'adresse 0.0.0.0 (`ANY_ADDR`) pour qu'il écoute sur toutes les interfaces en même temps : problème éventuel de sécurité et/ou manque de contrôle des flux entrants ;
- soit configurer le routage IP pour que l'interface sur laquelle est *bind* CNESLSI soit accessible depuis n'importe quel sous-réseau, par transitivité : oblige à configurer le routage IP ce qui alourdit considérablement la configuration tout en étant plus ou moins contraire à la philosophie dynamique de DTN.

Nous avons donc modifié le code source de CNESLSI pour qu'il soit possible de *bind* plusieurs instances sur des interfaces différentes. Du point de vue du fichier de configuration (dans ltpadmin) :

- **avant** : `s 'cneslsi <@IP:port>'`
- **maintenant** : `s 'cneslsi <@IP1:port1> <@IP2:port2> [...]'`

Le code ainsi implémenté modifie une partie importante de CNESLSI. Il peut encore être amélioré mais a passé les tests avec succès. A voir si cette modification doit être effectuée dans le code de base (UDPLSI), ce qu'en pensent les développeurs principaux d'ION...

III Étude du transport de plusieurs segments LTP dans une unique trame AOS

Il ne nous a pas semblé utile d'implémenter cette fonctionnalité pour le moment. Il faudra peut-être la réaliser en fonction de l'implémentation choisie par la NASA.

IV Caractérisation de la transmission SLE sur lien stressé

Il n'a pas été possible de mettre en place cette expérimentation, compte tenu de l'environnement technique actuel du laboratoire.