

L<sup>A</sup>T<sub>E</sub>X

Appunti dalle lezioni di:

## **Protocols and Architectures for Space Networks**

Davide Tropea

A.A. 2020/2021



# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	DNT overview . . . . .	3
1.2	Bundle Protocol DTN architecture . . . . .	3
1.2.1	Proactive fragmentation . . . . .	4
1.2.2	Reactive fragmentation . . . . .	4
1.2.3	Late binding . . . . .	4
<b>2</b>	<b>RFC 4838</b>	<b>5</b>
2.1	Introduction . . . . .	5
2.2	Virtual Message Switching using Store-and-Forward . . . . .	5
2.3	Nodes, Endpoints, EIDs and Registrations . . . . .	6
2.3.1	URI Schemes . . . . .	6
2.4	Late Binding . . . . .	6
2.5	Priority Classes . . . . .	7
2.6	Postal-Style Delivery Options and Administrative Records . . . . .	7
2.6.1	Delivery Options . . . . .	7
2.6.2	Administrative Records: Bundle Status Reports and Custody Signals . . . . .	8
2.7	Primary Bundle Fields . . . . .	9
2.8	Routing and Forwarding . . . . .	9
2.9	Fragmentation and Reassembly . . . . .	10
2.10	Reliability and Custody Transfer . . . . .	10
2.11	DTN Support for Proxies and Application Layer Gateways . . . . .	11
2.12	Timestamps and Time Synchronization . . . . .	12
<b>3</b>	<b>RFC 5050 and BPbis draft</b>	<b>13</b>
3.1	Introduction . . . . .	13
3.2	Definitions . . . . .	13
3.3	Bundle Format . . . . .	14
3.3.1	BP Fundamental Data Structures . . . . .	15
3.3.1.1	CRC Type and CRC . . . . .	15
3.3.1.2	Block Processing Control Flags . . . . .	15
3.3.1.3	Bundle Processing Control Flags . . . . .	15
3.3.1.4	Identifiers . . . . .	16
3.3.1.4.1	Endpoint ID . . . . .	16
3.3.1.4.2	Node ID . . . . .	16
3.3.1.5	DTN Time and Creation Timestamp . . . . .	17
3.3.2	Bundle Representation . . . . .	17
3.3.2.1	Bundle . . . . .	17
3.3.2.2	Canonical Bundle Block Format . . . . .	17
3.3.2.3	Primary Block . . . . .	18
3.3.3	Extension Blocks . . . . .	18
3.4	Bundle Fragmentation . . . . .	18
3.5	Administrative Records . . . . .	19
3.5.1	Bundle Status Reports . . . . .	19
3.6	CC's Notes on bundle identification . . . . .	20
<b>4</b>	<b>Satellites Networks</b>	<b>21</b>
4.1	Introduction . . . . .	21
4.1.1	Geostationary Earth Orbit (GEO) . . . . .	21
4.1.2	Low Earth Orbit (LEO) . . . . .	21
4.2	DTN as an evolution of TCP-splitting PEPs . . . . .	22
4.3	DTN and GEO/LEO Satellite Communications . . . . .	23
4.3.1	TATPA Testbed . . . . .	23
4.3.2	GEO satellites with fixed terminals . . . . .	24
4.3.2.1	Ideal channel . . . . .	24
4.3.2.2	Congestion . . . . .	24
4.3.2.3	PER . . . . .	24
4.3.2.4	PER and congestion . . . . .	25
4.3.3	GEO satellite with mobile terminals . . . . .	25
4.3.4	LEO satellites . . . . .	26
4.3.4.1	LEO data mule . . . . .	26

4.3.4.2	LEO Earth Observation <sup>1</sup>	28
4.3.5	Summary	30
<b>5</b>	<b>Interplanetary Networks</b>	<b>31</b>
5.1	Introduction	31
5.1.1	Moon to Earth Communications	31
5.1.2	Mars to Earth Communications	32
5.1.3	Multi-Asset Mars to Earth Communications	33
5.2	Licklider Transmission Protocol (LTP)	35
5.2.1	Introduction and Motivations	35
5.2.2	Terminology	35
5.2.3	LTP Session	36
5.2.4	Timers calculation	36
5.2.5	Data retransmission	37
5.2.6	Problems	37
5.2.6.1	Ideal transmission	37
5.2.6.2	Losses on data segments only	38
5.2.6.3	Losses on data and signaling segments	38
5.2.7	Possible Enhancements	39
5.2.8	Performances Analysis	40
5.2.9	Packet Layer Forward Error Correcting (PL-FEC)	41
5.3	Contact Graph Routing (CGR)	42
5.3.1	Introduction	42
5.3.2	Foundamentals	42
5.3.3	Example	43
5.3.4	Schedule-Aware Bundle Routing (SABR)	44
5.3.5	Unibo-CGR	45
<b>6</b>	<b>Security &amp; QoS</b>	<b>47</b>
6.1	DTN Security	47
6.1.1	Terminology	47
6.1.2	Block-Level Granularity	47
6.1.3	Security Blocks	48
6.2	Quality of Service (QoS)	48
6.2.1	Priority classes	48
6.2.2	General Routing Classification	48
6.2.3	Routing replication schemes in opportunistic environments	49
6.3	Course Conclusions	49
<b>7</b>	<b>Virtualbricks Lab</b>	<b>50</b>
7.1	Check installation	50
7.2	DTN2	50
7.2.1	DTN2hops layout	50
7.2.2	Start the Bundle Protocol	51
7.2.3	dtnd shell	51
7.2.4	default configuration file: /etc/dtn.conf	51
7.2.5	DTNperf	52
7.2.5.1	client	52
7.2.5.2	server	52
7.2.5.3	monitor	52
7.2.5.4	Basic applications	52
7.3	Traffic analysis with Wireshark	53
7.4	Experiment 1: GEO satellites with fixed terminal	53
7.4.1	pre-settings	53
7.4.1.1	TCP buffers	53
7.4.1.2	Channels configuration	54
7.4.1.3	VM2 isolation	54
7.4.2	Background traffic	54
7.4.3	Experiment	54
7.5	Experiment 2: LEO satellite data mule	55

<sup>1</sup>L'esperimento è stato discusso a lezione ed è una versione semplificata rispetto a quella presente nelle slides.

# Chapter 1

## Introduction

### 1.1 DNT overview

La rete internet classica è basata sulle seguenti assunzioni dettate dal protocollo TCP/IP:

- **Connettività end-to-end**: la comunicazione è possibile se esiste un collegamento continuo tra la sorgente e la destinazione;
- **RTT corto**: il recupero dei pacchetti persi è basato su ARQ (automatic repeat request) e cioè sulle ritrasmissioni dalla sorgente;
- **Ridotte perdite di pacchetti**: molte delle quali dovute alla congestione;

Le **Challenged Network** sono utilizzate in contesti in cui una o più di queste assunzioni sono violate. Le **DTN (Delay-/Disruption- Tolerant Networking)** sono le reti utilizzate per far fronte ai problemi delle challenged networks. Ambiti di utilizzo di queste reti sono: stazioni spaziali, militare, collegamenti sottomarini, comunicazioni a seguito di disastri naturali.

### 1.2 Bundle Protocol DTN architecture

Architettura basata sull'introduzione di un livello detto **bundle layer** tra il livello applicativo e quelli inferiori. L'unità della comunicazione è il **bundle**, un pacchetto dati di grandi dimensioni processato dai nodi seguendo la filosofia store-and-forward (i pacchetti ricevuti vengono salvati su una memoria locale ed inviati al nodo successivo non appena possibile).

Nelle reti internet classiche, è possibile creare collegamenti end-to-end poiché i nodi intermedi, tramite regole di routing prestabilite, riescono autonomamente a indirizzare tutti i pacchetti verso la destinazione finale. Caratteristica comune di ciascun router è che all'interno sono utilizzati i medesimi protocolli Internet dello stack OSI-ISO. Nelle reti DTN, la semantica del trasporto end-to-end è confinata all'interno di un **hop DTN**, ognuno dei quali può implementare stack di protocolli differenti. L'hop è il *saltello* che i bundles effettuano tra **due** nodi comunicanti. Un percorso end-to-end nelle DTN è costituito da più hops.

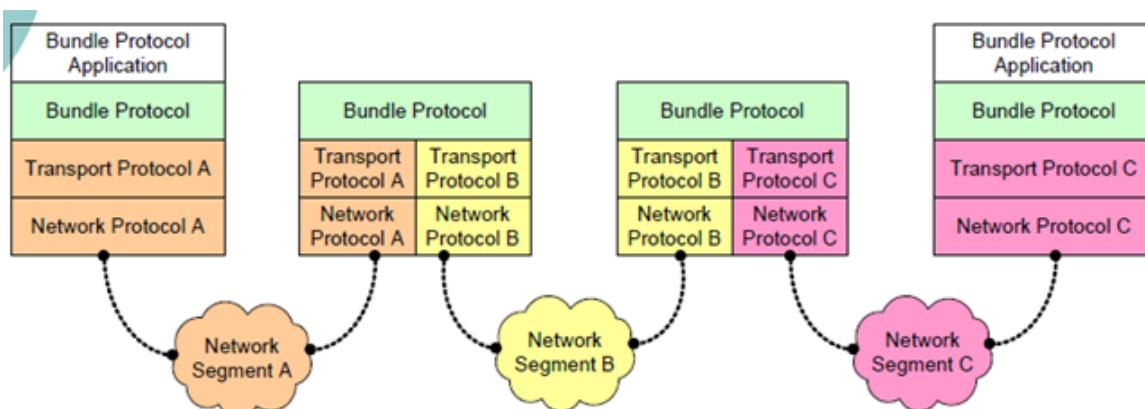


Figure 1.1: bundle layer protocol

Alla ricezione dei bundles ciascun nodo procede a scriverli in memoria locale poiché le condizioni ambientali in cui si trova potrebbero non essere adatte all'immediata trasmissione del bundle al nodo successivo. Inoltre, trovandosi in condizioni con RTT elevato, è più semplice procedere al recupero delle perdite.

Quando un bundle raggiunge la destinazione, un **token di accettazione della custodia** è inviato ai nodi precedenti e ne decreta la fine del cammino. Finché un nodo non riceve il token, continua a ritrasmettere il bundle seguendo un proprio **Retransmitted Timeout (RTO)**. Il bundle è rimosso dalla memoria locale solo se è stata raggiunta la sua naturale scadenza oppure se si è ricevuto un token di accettazione.

Dalle due immagini sottostanti è possibile fare un confronto tra le architetture TCP/IP e DTN. Il collegamento end-to-end interessa i due host agli estremi in TCP/IP, mentre i singoli hop intermedi in DTN. Inoltre, nei routers di TCP/IP è prevista la presenza di soli tre livelli (fisico, collegamento, rete) dello stack OSI-ISO, mentre negli hop DTN si possono anche avere protocolli differenti.

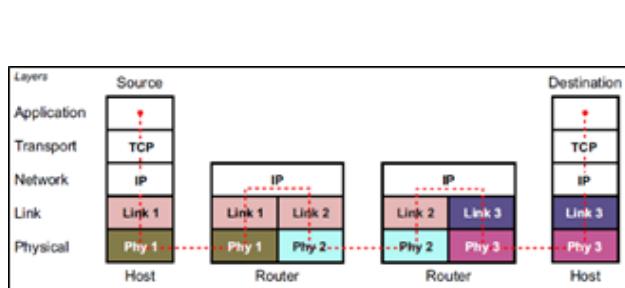


Figure 1.2: TCP/IP

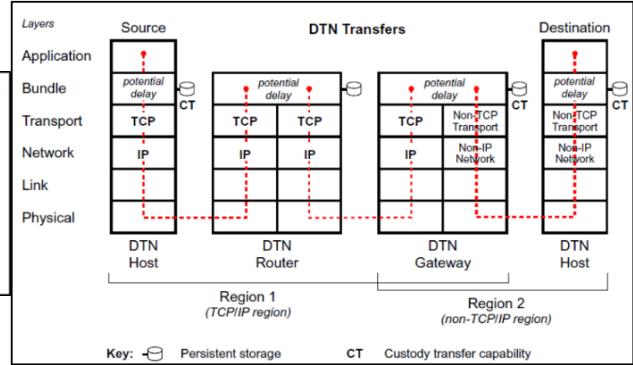


Figure 1.3: DTN

### 1.2.1 Proactive fragmentation

E' una tecnica che permette di suddividere *a priori* i bundle in frammenti multipli con dimensioni compatibili con il **Contact Volume**. Il contact volume è l'ammontare massimo di dati che può essere trasferito tra nodi DTN durante un **Contatto**. E' implementato in ION (udpcl) ma non in DTN2.

### 1.2.2 Reactive fragmentation

E' una tecnica che entra in azione *a posteriori* quando si verifica un'interruzione durante il trasferimento di un bundle. Anziché ritrasmetterlo per intero, il bundle è suddiviso in due **frammenti** al nodo mittente: nel primo sono contenuti i dati già stati trasmessi; nel secondo la rimanente parte da trasmettere non appena il collegamento è ristabilito. Il secondo frammento è trattato dagli altri nodi come un normale bundle ed è possibile che sia frammentato ulteriormente al verificarsi di nuove interruzioni.

La reactive fragmentation è utile per la trasmissione di bundle di grandi dimensioni nelle situazioni in cui le interruzioni sono frequenti e imprevedibili, tipo la comunicazione dei satelliti con i dispositivi mobili. E' implementata in DTN2 e non in ION.

### 1.2.3 Late binding

Nelle reti IP i *nomi human-friendly* dei siti web sono i sinonimi degli *indirizzi* al quel sono collocati, l'associazione è effettuata dal browser web interrogando il DNS di riferimento. Nel bundle protocol l'utilizzo del DNS è impraticabile a causa degli elevati ritardi (o della totale mancanza della connessione) e il routing è basato esclusivamente sugli **Endpoint Identifiers (EID)**. Gli EID identificano uno o più nodi DTN e, non avendo alcun significato topologico, l'associazione EID-destinazione è effettuata solamente durante l'ultimo hop.

Per introdurre il **late binding** (che riprenderemo successivamente) supponiamo di voler inviare un bundle ad un nodo *futuro* e quindi inesistente al momento del primo inoltro: il bundle è instradato sulla base dell'EID finale, ad esempio `[dtn://futurenode.dtn]`; l'ultimo nodo della DTN lo prende in custodia finché un giorno non arriverà un nuovo nodo (con EID `[dtn://futurenode.dtn]`) a reclamarlo.

# Chapter 2

## RFC 4838

### 2.1 Introduction

L'RFC 4838 è il documento più importante sulle DTN in quanto descrive l'architettura DTN Bundle Protocol (BP), è di tipo *informativo* e non definisce degli standard da seguire.

L'architettura si basa sulla rete Internet Interplanetaria, da sempre concentrata sul problema della comunicazione con lo spazio profondo. Tale architettura è tuttavia applicabile a diversi contesti in cui le disconnessioni sono frequenti e i ritardi elevati, relegando il caso dello spazio profondo ad un semplice specializzazione.

Nel BP è prevista l'introduzione di un livello chiamato **bundle layer** nello stack tra il livello applicativo e il livello di trasporto, i nodi che lo implementano sono chiamati **nodi DTN**.

Principi fondamentali dell'architettura:

- utilizzare messaggi di lunghezza variabile, possibilmente lunghi e non di dimensioni limitate;
- migliorare l'interoperabilità tra i nodi utilizzando sintassi e indirizzamenti che supportino un'ampia gamma di convenzioni;
- fornire meccanismi di sicurezza che proteggano l'infrastruttura dall'uso non autorizzato;
- fornire classi, opzioni di consegna e metodi per esprimere la durata utile dei dati per consentire alla rete di soddisfare al meglio le esigenze delle applicazioni che la utilizzeranno;

Le applicazioni che utilizzano la rete devono:

- ridurre al minimo il numero di scambi bidirezionali;
- far fronte ai riavvii al verificarsi di errori lasciando le transazioni in sospeso;
- informare la rete della vita utile e dell'importanza relativa dei dati.

### 2.2 Virtual Message Switching using Store-and-Forward

Un'applicazione DTN installata su un nodo invia messaggi di lunghezza arbitraria denominati **Application Data Unit (ADU)**. Gli ADU, il quale ordine relativo potrebbe non essere mantenuto durante una trasmissione, sono trasformati dal bundle layer in una o più unità di dati chiamati **bundle**.

Il bundle ha una struttura dati ben definita formata da due o più **blocchi**: il **primary block** che funge da *header*; il **payload** con il carico dati utile da trasferire; gli **opzionali** utilizzati per scopi differenti. Il termine blocco è utilizzato al posto di header poiché potrebbe anche non apparire all'inizio del bundle a causa di particolari requisiti di elaborazione (ad esempio, firme digitali).

A seguito di errori durante la comunicazione, i bundles possono essere suddivisi in **frammenti**. Tali frammenti sono trattati come bundle dai nodi successivi e possono essere frammentati ulteriormente oppure riassemblati ovunque nella rete.

I nodi sorgente e destinazione sono identificati tramite gli **Endpoint Identifiers (EID)** che saranno descritti nel seguito. Per il momento è sufficiente sapere che nei bundles ne sono presenti diversi: sorgente; destinazione; **report-to**; **custodian**.

La modalità di trasmissione prevista nei nodi DTN è la **store-and-forward**, che, a differenza di quanto accade in IP (presenza di un collegamento continuo, eliminazione dei pacchetti dopo un breve lasso di tempo), non prevede che i collegamenti siano sempre disponibili

e affidabili. I nodi possono archiviare i bundle per un determinato periodo di tempo (facendoli sopravvivere anche ai riavvii del sistema) e ritrasmetterli non appena se ne presenta l'occasione (il **contatto**). Questo presuppone che lo spazio di archiviazione sia disponibile e ben distribuito in tutta la rete e che lo storage sia sufficientemente persistente e robusto.

Problemi comuni che derivano dall'utilizzo di queste reti ai quali è necessario porre rimedio sono la gestione della congestione e la mitigazione del Denial of Service.

## 2.3 Nodes, Endpoints, EIDs and Registrations

Un **nodo DTN** (o semplicemente *nodo*) è un calcolatore utile a inviare e ricevere bundle, rappresenta *de facto* l'implementazione del bundle layer. Le applicazioni installate su nodi diversi comunicano logicamente scambiandosi gli ADU trasportati all'interno dei bundles.

I nodi DTN possono far parte di gruppi chiamati **DTN Endpoints**. Un bundle è considerato consegnato con successo ad un Endpoint DTN quando un numero minimo di nodi dell'endpoint ha ricevuto il bundle correttamente senza errori. Questo sottogruppo di nodi è chiamato **Minimum Reception Group (MRG)** dell'endpoint. L'MRG di un endpoint può essere composto da un singolo nodo (**unicast**), un sottogruppo di nodi (**anycast**) oppure da tutto il gruppo (**multicast/broadcast**). Chiaramente un singolo nodo può far parte di diversi MRG.

Un **Endpoint Identifier (EID)** è un *nome* espresso utilizzando la sintassi generale degli **Uniform Resource Identifier (URI)** che identifica un Endpoint DTN. Utilizzando l'EID, un nodo è in grado di determinare l'MRG dell'Endpoint DTN. E' richiesto che anche ciascun nodo abbia almeno un EID che lo identifichi univocamente.

Le applicazioni inviano gli ADU ad un particolare EID e, viceversa, possono mettersi in attesa della ricezione di ADU destinate ad un particolare EID. A seconda di come l'EID è costruito, è possibile sfruttarne delle porzioni per diagnostica e scopi di routing.

La volontà di un'applicazione di ricevere degli ADU destinati ad un EID è chiamato **registrazione**. Con essa, l'applicazione comunica al BP che i bundles in arrivo con uno specifico *demux token* sono destinate a lei (funzionamento analogo alle porte TCP).

Le informazioni relative ad una registrazione sono mantenute dal nodo affinché possano sopravvivere agli improvvisi riavvii del sistema (o dell'applicazione stessa). Il tentativo di stabilire una nuova registrazione tuttavia, non è sempre garanzia di successo. Se l'applicazione ad esempio richiede gli ADU destinati ad un EID interpretabile o irraggiungibile, la richiesta fallisce.

### 2.3.1 URI Schemes

La sintassi degli URI è stata progettata per esprimere nomi o indirizzi per un'ampia gamma di scopi e risulta adatta a esprimere nomi per endpoint DTN.

Un URI è formato da due parti: uno **Scheme Name** e uno **Scheme Specific Part (SSP)**. Lo scheme name esprime i vincoli lessicale con i quali rappresentare e interpretare i caratteri dell'SSP.

L'utilizzo degli schemi URI è un concetto chiave nelle DTN poiché garantisce un'ampia flessibilità nella strutturazione e nell'interpretazione degli EID. Essendo dei semplici *nomi*, non è necessario infatti che gli EID abbiano un legame con la topologia della rete.

Come visto in precedenza, un singolo EID potrebbe essere riferito ad un endpoint contenente più di un nodo DTN. E' responsabilità del progettista dello schema dell'URI definire le regole per l'interpretazione dell'SSP di un EID, così da determinare se ci si sta riferendo ad una comunicazione unicast, anycast o multicast tra nodi.

Gli schemi utilizzati sono prevalentemente due:

- **Generale:** [dtn://susy.dtn/ping] in cui dtn:/susy.dtn è l'identificativo del nodo e ping è il *demux token* (equivalente alla porta TCP)
- **NASA:** [ipn:10.3] in cui ipn:10 è l'identificativo del nodo e 3 il *demux token* (equivalente alla porta TCP)

## 2.4 Late Binding

Il protocollo IP adotta una politica di *early binding* in cui si interroga un server DNS per effettuare l'associazione nome-indirizzo. Nell'architettura DTN è invece utilizzato il concetto di *late binding*.

Per *binding* si intende l'interpretazione dell'**SSP** di un EID con lo scopo di portare un messaggio verso una destinazione. Il termine *late* è presente poiché l'associazione EID-destinazione potrebbe non avvenire necessariamente al momento della creazione del bundle. L'EID di destinazione è potenzialmente reinterpretato in ogni hop, quindi l'associazione può avvenire alla fonte, durante il transito, o anche alla destinazione.

Si supponga di voler inviare un bundle ad un dispositivo che non è ancora fisicamente esistente e che quindi, non abbia ancora un indirizzo IP di destinazione; il bundle potrebbe essere instradato sulla base del nome DTN ad un gateway nel quale è memorizzato; quando il device di destinazione si *materializza*, in primo luogo stabilisce una connessione TCP/IP al gateway e successivamente gli comunica il suo nome DTN; poiché il suo indirizzo IP può essere derivato dal campo sorgente dei pacchetti IP in entrata, il bundle protocol associa il nome DTN del bundle in attesa all'indirizzo IP del dispositivo.

In una rete con frequenti disconnessioni, l'utilizzo del late binding risulta vantaggioso poiché l'utilizzo di un routing basato su nomi riduce l'ammontare di informazioni che devono essere propagate attraverso la rete.

## 2.5 Priority Classes

L'architettura DTN offre tre livelli di priorità *relativi* per la consegna degli ADU:

- **bulk**: spediti *con il minimo sforzo*, cioè sono a bassa priorità. Nessun bundle di questa classe è inviato finché non sono spediti prima i bundle di tutte le altre classi destinati alla stessa destinazione e provenienti dalla stessa origine;
- **normal**: spediti prima dei bundles appartenenti alla classe bulk;
- **expedited**: spediti prima dei bundle appartenenti a tutte le altre classi;

Le applicazioni specificano la classe di priorità e la durata per ciascun ADU inviato, questi parametri influenzano la tempestività di consegna. È importante notare che la classe di priorità associata ad un determinato bundle è relativa al solo nodo mittente. Ciò significa ad esempio che, i bundle expedited sono trattati in modo prioritario rispetto ai bundle bulk solamente se questi provengono dallo stesso nodo. La priorità potrebbe essere applicabile o meno su bundles provenienti da diverse sorgenti, ma questo dipende dalle politiche del nodo ricevente.

## 2.6 Postal-Style Delivery Options and Administrative Records

L'architettura DTN supporta molte **Delivery Options** che possono essere selezionate da un'applicazione alla richiesta di trasmissione degli ADU. In aggiunta, sono definite due tipologie di **Administrative Records**: **Status Report** e **Signal**. Questi records sono bundles che forniscono informazioni per la consegna di altri bundle e sono utilizzate in congiunzione con le delivery options.

### 2.6.1 Delivery Options

Le applicazioni che inviano gli ADU possono richiedere una combinazione tra opzioni riportate nel seguito. La combinazione è trasportata in ciascun bundle prodotto e inviato dal bundle layer del nodo DTN mittente. Sono definite otto delivery options.

- **Custody Transfer Requested**: i bundles inviati sono inviati con una maggiore affidabilità usando la procedura di custody transfer. I bundles sono inviati dal bundle layer utilizzando un protocollo di trasmissione affidabile (se disponibile), e la responsabilità di affidabilità può essere trasmessa tra i *custodi* nella rete (ove possibile, si spiegherà meglio il procedimento nei paragrafi successivi);
- **Source Node Custody Acceptance Required**: richiede che il nodo DTN sorgente fornisca la custody transfer per il bundle inviato. Se la custody transfer non è disponibile alla sorgente quando questa delivery option è richiesta, la trasmissione fallisce. Ciò fornisce alle applicazioni un mezzo per insistere affinché il nodo DTN di origine prenda la custodia dei bundle inviati;
- **Report When Bundle Delivered**: richiede che sia generato un (singolo) *Bundle Delivery Status Report* quando l'ADU è consegnato al destinatario previsto. Questa richiesta è anche nota come **return-receipt**;

- **Report When Bundle Acknowledged by Application:** richiede che sia generato un *Acknowledgement Status Report* quando l'ADU è ACK-ato dall'applicazione ricevente. Ciò differisce dalla richiesta di generazione del Bundle Delivery Status Report perché è l'applicazione ricevente a richiederlo. E' utile nei casi in cui le applicazioni agiscono da *application layer gateway* e desiderano indicare lo stato di una operazione protocollare esterna al DTN alla sorgente richiedente;
- **Report When Bundle Received:** richiede che sia generato un **Bundle Reception Status Report** quando ciascun bundle inviato arriva al nodo DTN. E' pensato principalmente per scopi diagnostici;
- **Report When Bundle Custody Accepted:** richiede che sia generato *Custody Acceptance Status Report* quando ciascun bundle inviato è stato accettato usando la custody transfer. E' pensato principalmente per scopi diagnostici.
- **Report When Bundle Forwarded:** richiede che sia generato un *Bundle Forwarding Status Report* quando ciascun bundle inviato lascia un nodo DTN dopo l'inoltro. E' pensato principalmente per scopi diagnostici;
- **Report When Bundle Deleted:** richiede che sia generato un *Bundle Deletion Status Report* quando ciascun bundle inviato è cancellato dal nodo DTN. E' pensato principalmente per scopi diagnostici;

Le prime quattro delivery options sono pensate per l'uso ordinario delle applicazioni. Le altre quattro per uso diagnostico e il loro utilizzo potrebbe risultare limitato in determinate condizioni ambientali.

Le procedure di sicurezza mettono a disposizione tre ulteriori delivery options che possono essere abilitate: *Confidentiality Required*, *Authentication Required*, *Error Detection Required*.

### 2.6.2 Administrative Records: Bundle Status Reports and Custody Signals

I record amministrativi sono utilizzati per segnalare informazioni sullo stato o condizioni di errore relative al bundle layer. Se ne distinguono due tipi: **Bundle Status Report (BSR)**: inviato dal BP all'EID *report-to*; **Custody signal**: inviato dal BP all'EID *current-custodian* (che può anche differire dall'EID di sorgente);

I record amministrativi vengono inviati come bundle con un EID di sorgente impostato su uno degli EID associati al nodo DTN che genera il record amministrativo.

Attualmente sono definiti i seguenti BSR:

- **Bundle Reception:** inviato quando un bundle arriva al nodo DTN;
- **Custody Acceptance:** inviato quando un nodo ha accettato la custodia di un bundle con l'opzione *Custody Transfer Request* abilitata;
- **Bundle Forwarded:** inviato quando un bundle contenente l'opzione *Report When Bundle Forwarded* parte da un nodo DTN dopo essere stata inoltrato;
- **Bundle Deletion:** inviato da un nodo DTN quando un bundle contenente l'opzione *Report When Bundle Deleted* è scartato;
- **Bundle Delivery:** inviato dal nodo DTN destinatario finale quando un ADU compreso nel bundle inviato contenente l'opzione *Report When Bundle Delivered* attiva è utilizzato dall'applicazione;
- **Acknowledged by application:** inviato dal nodo DTN destinatario finale quando un ADU completo compreso nel bundle inviato contenente l'opzione *Application Acknowledgement* attiva è processato dall'applicazione.

In aggiunta agli status reports, il custody signal è definito per indicare lo stato di un custody transfer. E' inviato all'EID current-custodian contenuto nel bundle ricevuto:

- **Custody Signal:** indica che la custodia è stata trasferita con successo. Questo segnale appare come un indicatore booleano, ciò significa che può rappresentare sia il successo che il fallimento del tentativo di trasferimento della custodia.

**Gli administrative records devono necessariamente fare riferimento al bundle ricevuto.**

I BSR sono una caratteristica essenziale delle DTN poiché permettono ai ricercatori e controllori di tracciare un determinato bundle (vedi esempio sotto).

La distinzione tra l'EID di *report-to* e *sorgente* è molto sottile, poiché un bundle può essere tracciato anche utilizzando un nodo esterno che funge da controller. Quando l'EID *report-to* coincide con quello della *sorgente*, il campo **delivered (Dlv)** nello SR funziona come *ricevuta di ritorno* che può essere usata dall'applicazione DTN per imporre l'affidabilità end-to-end. La vera affidabilità end-to-end non è garantita dal solo BP, ma custode-per-custode.

TABLE I: EXCERPT OF A CSV LOG FILE: STATUS REPORTS TRIGGERED BY BUNDLE “3970.44” SENT FROM VM1 TO VM2.

RX TIME	Report_SRC	Report TST	Rep. SQN	Rep. Type	Bndl_SRC	Bndl TST	Bndl SQN	Bndl FO	Bndl FL	Dlv	Ct	Rev	Fwd	Del
0	dtn://vm1.dtn	3970	45	S_R	dtn://vm1.dtn/dtnperf/src_10860	3970	44	0	0		3970			
3.85	dtn://vm2.dtn	3973	42	S_R	dtn://vm1.dtn/dtnperf/src_10860	3970	44	0	0			3973		
3.93	dtn://vm1.dtn	3974	49	S_R	dtn://vm1.dtn/dtnperf/src_10860	3970	44	0	0					3974
4.46	dtn://vm2.dtn	3973	44	S_R	dtn://vm1.dtn/dtnperf/src_10860	3970	44	0	0		3973			
4.47	dtn://vm2.dtn	3973	45	S_R	dtn://vm1.dtn/dtnperf/src_10860	3970	44	0	0	3973				

## 2.7 Primary Bundle Fields

La struttura di un Bundle (definita interamente nell'RFC 5050) è così composta:

- **Primary Block:** è come un header, deve *necessariamente* essere replicato in tutti i fragments ed è composto da:
  - **Creation Timestamp:** concatenazione dell'ora di creazione del bundle e di un numero di sequenza incrementale in modo tale da garantire che il CT sia univoco per ogni ADU proveniente dalla stessa fonte;
  - **Lifespan:** la lifespan (durata) di un bundle è espressa come l'offset rispetto al suo tempo di creazione. Quando un bundle viene archiviato nella rete, può essere cancellato dopo che ha raggiunto la sua durata limite;
  - **Class of Service Flags:** indicano le opzioni di consegna e la classe prioritaria del bundle;
  - **Source EID:** EID del nodo sorgente (il primo);
  - **Destination EID:** EID del/dei nodo/nodi destinatari;
  - **Report-to EID:** EID del nodo al quale dovrebbe essere inviato il report;
  - **Custodian EID:** EID del custode corrente del bundle (se ne esiste uno);
- **Payload block;**
- **Optional (Extension) blocks;**

## 2.8 Routing and Forwarding

I nodi in una rete DTN potrebbero essere interconnessi utilizzando tecnologie di reti sotostanti differenti, una rete DTN è descritta in modo astratto utilizzando un *multigrafo* (un grafo in cui i vertici possono essere interconnessi con più di un bordo). Gli archi del grafo sono variabili nel tempo poiché dipendono da ritardo, capacità e direzionale (capacità = 0 significa che non c'è collegamento tra due nodi).

L'intervallo di tempo durante il quale due nodi sono connessi è chiamato **contatto**; la velocità del trasferimento tra i due nodi è chiamata **capacità**; il prodotto tra la capacità e il contatto è chiamato come **volume di contatto** e rappresenta la massima quantità di dati che è possibile inviare in uno specifico contatto. Se i contatti e i relativi volumi sono noti a priori, si possono effettuare scelte di routing e forwarding intelligenti. Viceversa, se tali dati non sono noti e durante il percorso che un bundle dovrebbe seguire sono previste perdite, i calcoli diventano particolarmente impegnativi (si entra nell'ambito delle *opportunistic networks* e altre proposte di routing, molte delle quali di scarso interesse).

Si distinguono diversi tipi di contatto:

- **Persistent Contacts:** anche detti *always-on*, sono sempre disponibili, cioè, non è necessaria alcuna inizializzazione della connessione per istanziare il contatto;
- **On-Demand Contacts:** è richiesta l'esecuzione di alcune azione per la creazione di un'istanza, dopodiché sono trattati come persistent contract fino al momento in cui sono terminati;

- **Intermittent**: suddivisi a loro volta in:

- **Scheduled Contacts**: è un accordo per stabilire un contatto in un determinato momento e per una determinata durata. Un esempio di questo tipo di contatto è il collegamento con un satellite in orbita terrestre bassa;
- **Opportunistic Contacts**: non sono programmati e si presentano inaspettatamente. Un esempio potrebbe essere un aeromobile fuori programma che sorvola il cielo e si mette a disposizione di instaurare delle comunicazioni;
- **Predicted Contacts**: non sono basati su una pianificazione fissa. In base a cronologie di contatti osservati in passato è possibile stimare tempi probabili in cui il contatto potrebbe avvenire.

## 2.9 Fragmentation and Reassembly

La **frammentazione** e il **riassembaggio** sono funzioni progettate per migliorare l'efficienza dei trasferimenti dei bundle. Il loro obiettivo è garantire che il volume di contatto sia interamente utilizzato (proactive fragmentation) e che siano evitate le ritrasmissioni di bundles già parzialmente inviati (reactive fragmentation)

- **Proactive Fragmentation**: il nodo DTN suddivide il blocco dati dell'applicazione in blocchi più piccoli, ciascuno di questi blocchi è trasmesso come se fosse un bundle indipendente che può a sua volta essere frammentato. In questo caso, i nodi destinatari finali sono responsabili dell'estrazione dei blocchi più piccoli dai bundles in arrivo e del loro riassemblaggio per poter ricreare correttamente l'ADU. Il termine *proattiva* è dovuto al fatto che, nei contesti in cui è utilizzata tale tecnica, il volume di contatto è noto a priori.
- **Reactive Fragmentation**: i nodi DTN che condividono un arco nel grafo DTN possono frammentare un bundle in modo *cooperativo* se questo è inviato solo parzialmente. Il ricevente modifica il bundle ricevuto per indicare che è un frammento e in seguito lo inoltra normalmente. Il mittente può apprendere che solo una parte del bundle che ha tentato di inviare in precedenza è stata in realtà consegnata, quindi, invia la parte restante al prossimo contatto. Il termine *retroattiva* è dovuto al fatto che il processo di frammentazione si verifica dopo che ha avuto luogo un tentativo di trasmissione.

Si noti come l'utilizzo di tali tecniche comporta problemi di sicurezza, in particolare di integrità e autenticazione dei bundles. Altre problematiche già trattate sono: la proactive fragmentation non è implementata in DTN2; la reactive fragmentation non è implementata in ION.

## 2.10 Reliability and Custody Transfer

Il servizio base del bundle layer ricorda il protocollo UDP, i messaggi sono inviati con una priorità associata (non è garantito tuttavia che tale priorità sia rispettata) e non è prevista l'invio di un ACK dal destinatario alla sorgente. Per implementare un trasferimento affidabile (come in TCP) sono presenti due opzioni: l'**ACK end-to-end** e la **custody transfer**.

L'**ACK end-to-end** è un classico acknowledgement che una sorgente richiede alla destinazione per avere conferma che il bundle sia correttamente ricevuto, le applicazioni sui nodi possono richiederlo se la loro implementazione lo prevede.

La custody transfer è una feature dell'architettura DTN, consiste nell'abilitare l'opzione di **Custody Transfer Request** presente nella struttura del bundle. Abilitando tale opzione, si trasferisce la responsabilità del trasferimento affidabile degli ADU nei bundles attraverso i nodi nella rete. Per trasferimenti unicast ciò comporta in genere lo spostamento dei pacchetti (in termini di metrica di instradamento) *vicino* alla destinazione finale e la ritrasmissione degli stessi quando necessario. I nodi che lungo il percorso seguito dai bundles ne accettano la custodia si chiamano **custodi**, e, lo spostamento fisico di un bundle da un custode ad un altro, si chiama appunto custody transfer.

L'architettura DTN non prevede che tutti i nodi accettino il trasferimento della custodia, per cui esso non è da intendersi come un meccanismo hop-by-hop. Per esempio, alcuni nodi potrebbero avere lo storage sufficiente a salvare dei bundles e agire come custodi, ma potrebbero non offrire il servizio a causa di congestioni della rete oppure poiché lavorano a bassa potenza.

L'esistenza dei custodi può alterare le decisioni di routing, in alcune circostanze infatti, potrebbe convenire spostare un bundle da un custode ad un altro anche se questo si trova più lontano rispetto ad un altro nodo raggiungibile (in termini di distanza metrica).

La custody transfer è utilizzata inoltre per aumentare (anche se di poco) l'affidabilità delle consegne dei messaggi. Quando l'opzione è abilitata, il bundle layer fornisce un *timeout addizionale*, un *meccanismo di ritrasmissione* e un *meccanismo di segnali ACK custodian-to-custodian*. Se l'opzione invece risulta disabilitata, il timeout e il meccanismo di ritrasmissione non sono forniti dal bundle layer, e la consegna del bundle dipende esclusivamente dai protocolli di trasferimento affidabile implementati a livelli inferiori.

Quando un nodo accetta la custodia di un bundle che contiene l'opzione *Custody Transfer Request* abilitata, invia un segnale **Custody Transfer Accepted** al nodo identificato dall'EID *Current Custodian* contenuto nel *primary block*. Prima dell'inoltro del bundle al nodo successivo, l'EID *Current Custodian* è aggiornato (per trasferimenti unicast) con quello del nodo corrente.

Quando un'applicazione richiede che un ADU sia consegnato con il custody transfer, la richiesta è *consultiva*. In alcune circostanze, la sorgente del bundle per il quale è stata richiesta la custody transfer, potrebbe non essere in grado di fornire questo servizio. Il bundle in questione, potrebbe in seguito attraversare diversi nodi DTN prima di ottenere un custode. In questa situazione, l'EID *Current Custodian* è marcato ad un endpoint nullo. Nei casi in cui l'applicazione desidera che la sorgente prenda in custodia il bunde, può abilitare l'opzione *Source Node Custody Acceptance Required*. Ciò potrebbe risultare utile se le applicazioni desiderano avere per un bundle una *catena* di custodia continua.

Nelle reti DTN in cui uno o più hop custodian-to-custodian sono strettamente unidirezionali, il meccanismo di custody transfer potrebbe essere alterato a causa dell'impossibilità a ricevere un segnale di accettazione della custodia (e qualsiasi altra informazione). Questa situazione non significa necessariamente che il bundle si perda, i nodi dall'altro lato dell'hop possono continuare a trasferire la custodia per poi consegnare il bunde alla destinazione corretta. In questa circostanza la sorgente richiede un BSR e ricevendolo, giunge erroneamente all'conclusione che il bundle sia stato scartato e non consegnato. Sebbene questo problema non possa esser risolto completamente, sono forniti meccanismi per aiutare ad interpretare meglio informazioni apparentemente errate. Ciò viene eseguito dal nodo mittente prima che il bundle sia inviato in un collegamento unidirezionale usando una variante del BSR. Questi tipi di report sono forniti se il bundle in questione richiede il report abilitando l'opzione **Report When Bundle Forwarded**.

Nelle versioni future del bundle protocol, il meccanismo di custody transfer sarà rimpiazzato da un altro meccanismo chiamato **bundle-in-bundle encapsulation**.

## 2.11 DTN Support for Proxies and Application Layer Gateways

Uno degli obiettivi delle reti DTN è quello di fornire un metodo comune per l'interconnessione di gateway e proxy al livello di applicazione. Nei casi in cui si voglia rendere un'applicazione Internet classica tollerante ai ritardi, è possibile creare proxy locali per trarre i benefici delle capacità di comunicazione messe a disposizione dalle DTN. Rendere tali proxy compatibili con le DTN riduce il lavoro dello sviluppatore, poiché non deve preoccuparsi di implementare il routing e la gestione dell'affidabilità, permettendo alle applicazioni TCP/IP di operare su una DTN.

Quando le DTN sono utilizzate per realizzare una forma di incapsulamento di altri protocolli, possono essere costruite reti di overlay comprese di gateway a livello di applicazione. Ciò permette ai gateway remoti di segnalare il successo o il fallimento di operazioni non basate sul protocollo DTN semplicemente interpretando gli ADU ricevuti. Senza questa capacità, tali indicatori dovrebbero essere implementati dalle applicazioni stesse in modi non standard.

E' bene specificare che, sebbene sia possibile permettere alle applicazioni di funzionare sulle DTN, queste dovranno comunque essere sottoposte a lunghi ritardi e avere una interazione minima. Si pensi ad un SYN TCP, se incapsulato in un bundle e inoltrato utilizzando le DTN si aspetterebbe un ACK nel giro di breve tempo ma il contesto applicativo non lo permette. L'inoltro di file è invece possibile senza problemi.

## 2.12 Timestamps and Time Synchronization

L’architettura DTN è strettamente legata alla sincronizzazione temporale tra i nodi per quattro motivi: identificazione di bundle e frammenti; effettuare il routing sfruttando contatti scheduled o predicted; scadenza dei bundle; scadenza delle registrazioni delle applicazioni.

L’identificazione e la scadenza dei bundles sono implementate posizionando all’interno di ciascuno di essi il *creation timestamp* (momento di creazione) e, in un apposito campo, il tempo di scadenza (espresso come offset dal creation timestamp in secondi). I timestamp di origine dei bundles in arrivo sono resi disponibili alle applicazioni che ricevono gli ADU grazie a qualche chiamata di sistema. Ciascun set di bundles corrispondente ad un ADU deve contenere un timestamp univoco per l’EID del mittente.

L’**EID**, il **timestamp**, e il **numero di sequenza** sono i tre parametri che identificano univocamente un bundle *non frammentato*. I bundle frammentati si distinguono utilizzando due ulteriori campi che sono l’**offset** e la **lunghezza del frammento**. Tale identificazione è usata per diversi scopi, tra cui il custody transfer e il riassemblaggio dei frammenti di bundles. E’ bene notare che per il riassemblaggio sono utilizzati l’offset e la lunghezza dei dati amministrativi (non la lunghezza del frammento).

Il tempo è utilizzato anche per il processo di registrazione, quando un’applicazione infatti desidera ricevere degli ADU destinati ad un particolare EID, tale registrazione è mantenuta per un periodo finito di tempo (che potrebbe essere fissato dall’applicazione). Per le registrazioni multicast, l’applicazione può specificare un range di tempo o un *intervallo di interesse* durante il quale ricevere bundles. In questo modo, tutto il traffico inviato nell’intervallo ad un EID specifico, è consegnato all’applicazione (a meno che tale traffico non sia scaduto).

# Chapter 3

## RFC 5050 and BPbis draft

### 3.1 Introduction

Sin dal momento della sua pubblicazione, il Bundle Protocol è stato implementato in diversi linguaggi di programmazione e reso disponibile per un'ampia varietà di piattaforme. Queste implementazioni hanno individuato opportunità per rendere il protocollo più efficace e semplice da usare.

L'**RFC 5050** (rilasciato nel 2007) contiene i dettagli implementativi della *versione 6* del Bundle Protocol (**BPv6**). Anch'esso come il 4838 è un RFC sperimentale, quindi, fornisce alla comunità di internet un protocollo da poter adottare, non imponendo alcuno standard.

Sebbene l'RFC 5050 rappresenti ad oggi l'implementazione ufficiale, è in corso di pubblicazione da parte dell'*Internet Engineering Task Force (IETF)* un nuovo draft che andrà a rimpiazzarlo. Tale draft, noto col nome di **BPbis**, descrive la *versione 7* del Bundle Protocol (**BPv7**) ed è destinato a diventare un RFC standard.

La chiarezza del BPbis è influenzata negativamente dalle numerose riscritture in corso, per cui la versione a cui si farà riferimento è la *26* (l'ultima disponibile è la *28*). Durante il capitolo saranno evidenziate le differenze tra BPbis e l'RFC 5050, con eventuali commenti e valutazioni del professore (CC).

Il Bundle Protocol (collocato nello stack di protocolli come mostrato in figura 1.1) utilizza i protocolli *nativi* di trasporto e/o di rete sottostanti per le comunicazioni all'interno di una rete. Il livello al quale si trovano i protocolli sottostanti è chiamato (nel contesto del draft) **Convergence Layer**, mentre l'interfaccia tra il BP e uno specifico protocollo sottostante è chiamata **Convergence Layer Adapter**.

### 3.2 Definitions

Si elencano nel seguito le definizioni dei termini usati nel draft:

- **Bundle**: è l'unità dati del DTN Bundle Protocol. Ciascuno di essi è composto da due o più **blocchi** di dati.
- **Block**: è una struttura dati che contribuisce a formare correttamente un bundle.
- **Application Data Unit (ADU)**: è l'unità dati dell'applicazione. Lo scopo della trasmissione di un bundle (non frammentato) è trasferire gli ADU tra i nodi.
- **Payload**: è il carico dati utile contenuto all'interno di un bundle. Per un bundle che non è un frammento, il payload è un ADU.
- **Partial Payload**: è un payload che comprende o i primi  $N$  o gli ultimi  $N$  byte di un bundle di dimensione  $M$ , tale che  $0 < N < M$ . Si noti che ogni partial payload è un payload, quindi può essere a sua volta suddiviso in altri partial payload.
- **Fragment**: è un bundle il cui payload contiene un partial payload.
- **Bundle node**: (o semplicemente **node**) è un'entità che può inviare e/o ricevere bundles. È composto concettualmente da tre elementi: il *bundle protocol agent*, una serie di zero o più *convergence layer adapters*, e un *application agent*.
- **Bundle protocol agent (BPA)**: è il componente del nodo che fornisce i servizi ed esegue le procedure del Bundle Protocol.

- **Convergence layer adapter (CLA)**: sono i componenti del nodo che inviano e ricevono i bundle per conto del BPA utilizzando i servizi messi a disposizione da qualche stack di protocolli supportato dalla rete nel quale il nodo è collocato.
- **Application Agent (AA)**: è il componente del nodo che utilizza i servizi del BP per effettuare la comunicazione per qualche scopo dell’utente. E’ composto da due elementi: l’*administrative element* e l’*application-specific element*.
- **Application-specific Element**: è il componente dell’AA che costruisce, richiede la trasmissione, accetta la consegna ed elabora gli ADU.
- **Administrative element**: è il componente dell’AA che costruisce, richiede la trasmissione, accetta la consegna ed elabora tutti gli *Administrative Records* (definiti sotto), inclusi gli *status report*.
- **Administrative record**: è un’ADU particolare scambiata tra gli *Administrative Element* degli AA nei nodi. L’unico Administrative Record definito in questo documento è lo **status report** (discusso in seguito).

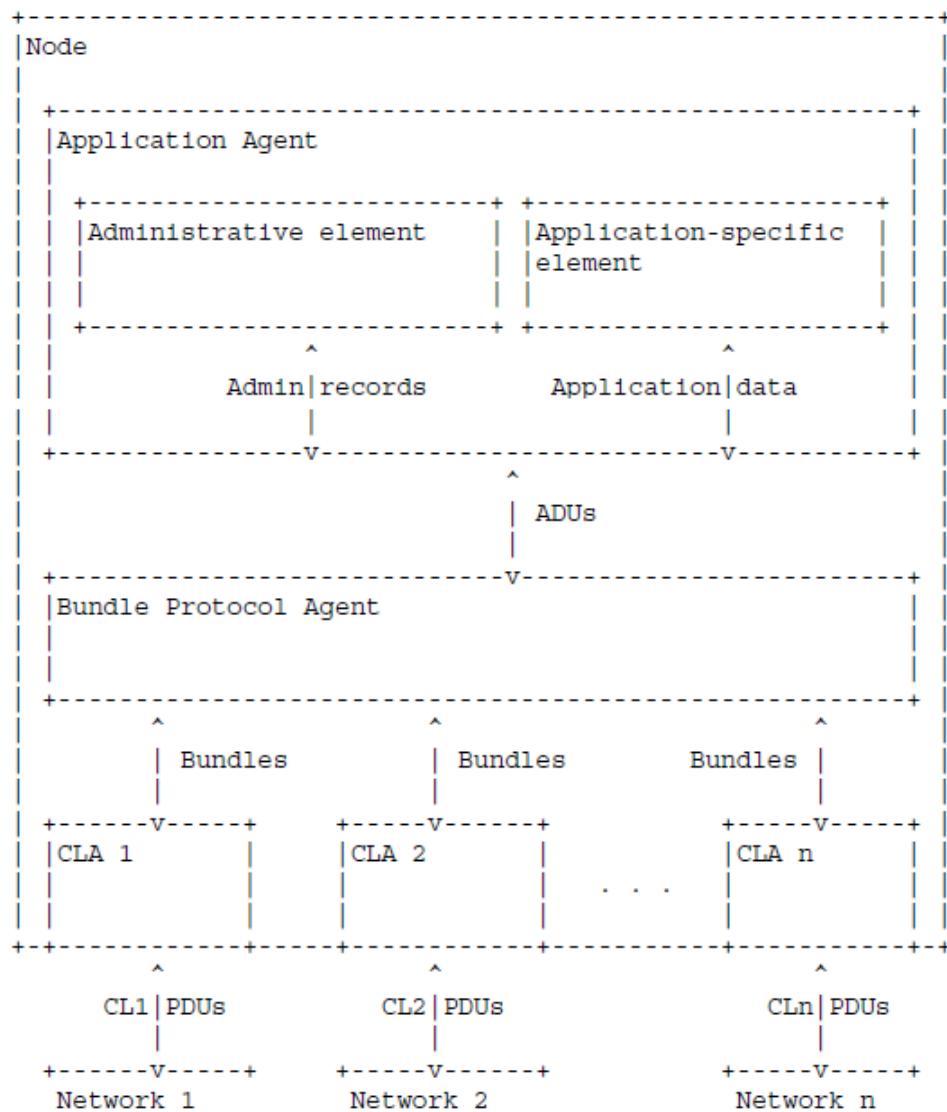


Figure 3.1: Components of a Bundle Node

- **Bundle Endpoint**: (o semplicemente **endpoint**) è un insieme di zero o più nodi identificati univocamente tramite un **Endpoint bundle ID** (o semplicemente **EID**).
- **Singleton endpoint**: è un endpoint formato da *un solo* nodo. Ogni nodo deve necessariamente far parte di almeno un singleton endpoint.

### 3.3 Bundle Format

Il formato dei bundle è conforme alla **Concise Binary Object Representation (CBOR)**. Ciascuno di essi è costituito da una sequenza concatenata di almeno due blocchi, rappresentati come CBOR array di lunghezza indefinita (tabella 3.1). Il primo e l’ultimo elemento dell’array sono rispettivamente il **primary bundle block** e il **payload block**. Nel mezzo sono presenti uno o più **canonical bundle blocks** (elementi addizionali).



Table 3.1: Rappresentazione grafica di un bundle

Il primary block ha un formato *specifico*, tutti gli altri blocchi hanno la stessa forma *canonica*. Una implementazione del Bundle Protocol può:

- scartare tutte le sequenze di byte non conformi alle specifiche del BP;
- accettare le sequenze di byte non conformi alle specifiche e trasformarle in strutture adatte prima di processarli. Le procedure per realizzare tale trasformazione esulano dallo scopo di questo draft.

CC

La maggior differenza che si riscontra tra l'RFC 5050 e il BPbis è il formato del bundle: **Self Delimiting Numerical Values (SDNV)** nell'RFC 5050; **CBOR** nel BPbis.

### 3.3.1 BP Fundamental Data Structures

CC

Nel paragrafo 4.1 del draft BPbis sono anticipate alcune strutture dati molto discutibili in termini di chiarezza.

#### 3.3.1.1 CRC Type and CRC

Il **CRC type** è un CBOR `unsigned integer`, presente in ciascun blocco, che ammette solo i seguenti valori:

- **0** indica *non è presente alcun CRC*;
- **1** indica *è presente uno standard X-25 CRC-16*;
- **2** indica *è presente uno standard CRC32C (Castagnoli) CRC-32*

Il **CRC** è omesso se il CRC type è 0, altrimenti è rappresentato come `CBOR byte string` di due (CRC type = 1) o quattro byte (CRC type = 2). La sequenza di byte è costituita da `unsigned integer` (16 e 32 bit rispettivamente) nel *network byte order*.

Si noti che una protezione più robusta dell'integrità dei dati nel BP può essere fornita, a seconda della necessità, mediante *Block Integrity Blocks* come definito nel **Bundle Security Protocol (BPSEC)**.

CC

I CRC sono usati, come al solito, per verificare l'integrità dei dati. Il loro utilizzo in BPbis va a colmare, sotto questo punto di vista, una delle lacune presenti nell'RFC 5050.

#### 3.3.1.2 Block Processing Control Flags

I **Block Processing Control Flags** sono bit che esprimono proprietà dei singoli blocchi dei bundle. Sono contenuti nell'header di tutti i blocchi *canonici* a cui si riferiscono.

Bit	BPbis	RFC5050 (when different)
0	block must be replicated in every fragment	
1	transmission of a status report is requested if block can't be processed	
2	bundle must be deleted if block can't be processed	Last block
3	Reserved	
4	block must be removed from bundle if it can't be processed	Block was forwarded without being processed
5	Reserved	Block contains an EID
6	Reserved	Notpresent
7-63	Unassigned	

#### 3.3.1.3 Bundle Processing Control Flags

I **Bundle Processing Control Flags** sono bit che esprimono proprietà dell'intero bundle (non dei singoli blocchi). Sono contenuti all'interno del primary block.

Bit	BPbis	RFC5050 (when different)
0	Bundle is a fragment	
1	Payload is an administrative record	
2	Bundle must not be fragmented	
3	Reserved	Custody Transfer requested
4	Reserved	Destination Endpoint is a singleton
5	Application acknowledgement is requested	
6	Reserved	
7	Reserved	(Cardinal) Priority (now in QoS extensions)
8	Reserved	(Cardinal) Priority (now in QoS extension)
9-13	Reserved	
14	Bundle reception status reports requested	
15	Reserved	Bundle custody acceptance s. rep. requested
16	Bundle forwarding status reports requested	
17	Bundle delivery status reports requested	
18	Bundle deletion status reports requested	
19-20	Reserved	
21-63	Unassigned	

### 3.3.1.4 Identifiers

#### 3.3.1.4.1 Endpoint ID

La destinazione dei bundles (*bundles endpoint*) è identificata da una stringa di testo chiamata **Endpoint IDs (EID)**. Ogni EID è un **Uniform Resource Identifier (URI)** caratterizzato da una struttura generale:

scheme name		scheme-specific part (SSP)
-------------	--	----------------------------

Lo **scheme name** è un insieme di regole sintattiche e semantiche che spiegano come analizzare e interpretare l'**SSP**. Ogni *scheme name* usato per formare un EID deve essere aggiunto in un registro gestito dall'IANA chiamato **Registry of URI Scheme Code Numbers**. L'associazione di un *URI scheme code number* ad uno specifico *scheme name* aiuta a rappresentare in modo più compatto gli EID nei blocchi dei bundles. Si noti che l'insieme dei possibili schemi ammissibili è illimitato. Ogni entry contenuta nel registro deve contenere un riferimento ad un documento che definisce come l'**SSP** di ciascun URI deve essere interpretato e codificato per la trasmissione.

Ciascun EID è rappresentato come un **CBOR array** di due dimensioni. Il primo elemento è un **CBOR unsigned integer** che identifica lo *scheme name* definito nel registro dell'IANA. Il secondo è la rappresentazione CBOR dell'**SSP**.

Gli schemi adottati per la rappresentazione sono due:

- **dtn:** dtn://susy.dtn/ping in cui:
  - susy.dtn è il *node name*;
  - ping è il *demux token* (equivalente alla porta TCP/UDP);
- **ipn:** ipn:3.2 in cui:
  - 3 è il *node number*;
  - 2 è il *service number* (equivalente al demux token di DTN);

In entrambi i casi, l'endpoint nullo è rappresentato con **dtn:none**

CC

La logica dietro lo schema ipn approvato dalla NASA è quello di risparmiare bit e riutilizzare i numeri per l'identificazione delle risorse spaziali, questo tuttavia è soggetto a errori.

La dualità dello schema ha causato molti problemi di interoperabilità e ha ostacolato la fusione delle comunità di ricerca spaziali e terrestri.

#### 3.3.1.4.2 Node ID

Nonostante i nodi siano distinti dagli endpoint (l'endpoint è un insieme di zero o più nodi) si utilizzano gli EID per identificarli. Si ricorda che all'interno dell'AA di ogni nodo è presente l'Administrative Element che scambia Administrative Records con gli altri nodi. Ogni nodo è strutturalmente e permanentemente registrato nel singleton endpoint al quale

gli administrative records ricevuti da altri nodi sono consegnati. Questo endpoint, chiamato **Administrative Endpoint**, è perciò univocamente e permanentemente associato al nodo. L'EID del nodo Administrative Endpoint è il **Node ID**.

CC

La spiegazione fornita dal draft, per quanto corretta, non è molto chiara. Uno sviluppo logico sarebbe dovuto iniziare dalla definizione di un identificatore per i nodi, per poi essere esteso agli endpoint identifiers, invece di fare il contrario.

### 3.3.1.5 DTN Time and Creation Timestamp

Il **DTN time** è un CBOR `unsigned integer` che indica il numero di millisecondi passati dal 01/01/2000 (scala **UTC: Universal Time Coordinated**).

CC

L'utilizzo dei millisecondi al posto dei secondi è uno dei vantaggi che il BPbis ha rispetto all'RFC5050. Il tempo restituito dalla funzione `time()` di UNIX si riferisce al 01/01/1970. Il DTN time può essere derivato da questo sottraendo 946684800 secondi.

Il **Creation Timestamp** è un CBOR `array` di due dimensioni contenuto in ogni bundle. Il primo elemento è il **bundle creation time**, cioè, il *DTN time* al quale la richiesta di trasmissione è stata ricevuta e, di conseguenza, il bundle è stato creato. Il secondo elemento è il **sequence number** (rappresentato come CBOR `unsigned integer`), cioè, l'ultimo valore del contatore incrementale (dal momento in cui è stata ricevuta la richiesta di trasmissione) gestito dal nodo sorgente. Il contatore è incrementato ogni volta che l'ora corrente avanza di un millisecondo.

Per i nodi in possesso di orologi locali poco accurati è raccomandato settare il *creation time* a zero e non resettare mai a zero il *sequence number*.

La creazione, in uno stesso nodo, di due bundle distinti con stesso *node ID* sorgente e stesso *creation timestamp* può portare a comportamenti inaspettati e/o scarse performance della rete. La combinazione di *node ID* e *creation timestamp* serve ad identificare una richiesta di trasmissione e fare in modo che l'applicazione ricevente possa inviare correttamente gli ACK.

## 3.3.2 Bundle Representation

### 3.3.2.1 Bundle

I Bundles sono rappresentati come CBOR `indefinite-length array`. Il primo elemento è la rappresentazione CBOR del Primary Block, tutti gli altri elementi sono invece la rappresentazione CBOR dei Canonical Blocks. L'ultimo blocco del bundle è seguito da un CBOR `break stop code` che termina l'array.

Ad ogni blocco è associato un **block number** (non presente nell'RFC5050) che lo identifica univocamente all'interno del bundle. Il primary block ha implicitamente *block number 0*, il payload 1, gli altri blocchi hanno un block number (specificato esplicitamente nell'header) non correlato all'ordine con cui sono presenti nel bundle.

### 3.3.2.2 Canonical Bundle Block Format

Tutti i blocchi, ad eccezione del primary block, sono **Canonical Blocks** rappresentati come CBOR `array` di 5 (se CRC=0) o 6 (altrimenti) elementi. I campi dei canonical blocks sono elencati di seguito nell'ordine nel quale *devono* apparire:

Other (i.e. Canonical) blocks fields
Block type
Block number
Block Processing Control Flags
CRC type
Block-type-specific data
CRC (optional)

Il **block type** può essere: 1 per indicare che il blocco è il payload (come detto in precedenza); 2-9 sono valori riservati; 192-255 sono disponibili per uso privato e/o sperimentale; gli altri valori sono riservati per utilizzi futuri.

### 3.3.2.3 Primary Block

Il **primary block** contiene le informazioni necessarie a inoltrare il bundle verso la sua destinazione. E' rappresentato come **CBOR array** ed è immutabile, cioè, il contenuto dei suoi campi restano identici dalla creazione, alla consegna del bundle. I campi del primary block sono elencati di seguito nell'ordine nel quale *devono* apparire:

Primary Block Field
Version
Bundle Processing Control Flags
CRC type
Destination EID
Source node EID
Report-to EID
Creation Timestamp
Lifetime (in ms)
Fragment offset (solo se il bundle è un fragment)
Total Application Data Unit Length (solo se il bundle è un fragment)
CRC (optional)

### 3.3.3 Extension Blocks

Gli **extension block** sono tutti i blocchi che non siano il primary e il payload. Poiché non tutti sono definiti nel Bundle Protocol, i nodi non devono necessariamente prevedere implementazioni di procedure per la loro gestione. E' possibile quindi che un nodo riceva un bundle con un extension block e che non sia in grado di processarlo. In questo caso, il valore presente nel campo *Block Processing Control Flags* indica l'azione che il BPA deve intraprendere. All'interno del draft sono presenti i seguenti extension block:

- **Previous node (Type 6):** identifica il nodo che ha inviato il bundle nell'hop (non è necessariamente il nodo sorgente) al nodo in cui il bundle attualmente risiede. E' usato per prevenire il *ping-pong* nel routing.
- **Bundle Age (Type 7):** contiene il numero di millisecondi che sono passati tra il tempo di creazione del bundle e il tempo in cui è stato inoltrato l'ultima volta. E' pensato per aiutare a stabilire la scadenza del bundle nei nodi sprovvisti di un orologio accurato.
- **Hop Count (Type 10):** è pensato per eliminare i bundle che hanno superato un certo numero limite di hops.
- **Manifest (Type 13):** non ne ha parlato a lezione e non c'è scritto niente.
- **Metadata (Type 14):** contiene informazioni riguardanti il payload. E' utile a proteggere (o non proteggere) la riservatezza dei dati in modo più flessibile.
- **Data Label (Type 15):** utile per le decisioni di inoltro.

## 3.4 Bundle Fragmentation

In alcune situazioni risulta vantaggioso per il BPA ridurre la grandezza dei bundle prima di inoltrarli (ad esempio perché è prevista una durata del contatto troppo breve). La grandezza può essere ridotta *frammentando* i bundles. Un bundle con payload di  $M$  bytes è suddivisibile in due **fragmentary bundles**: entrambi hanno *source node ID* e *creation timestamp* dell'originale, i payload contengono invece rispettivamente i primi  $N$  byte e gli  $M - N$  rimanenti (tale che  $0 < N < M$ ). I frammenti sono bundle e come tali possono essere suddivisi a loro volta dal BPA, a meno che non sia indicato nel primary block un flag che lo vietи.

Se necessario, il riassemblaggio degli ADU a partire dai frammenti è effettuato nel gruppo di nodi endpoint, sebbene possa avvenire anche in un altro nodo del percorso.

CC

La frammentazione è una tecnica molto invitante da utilizzare, ma presenta molti svantaggi nascosti e dovrebbe essere usata solo quando strettamente necessario.

## 3.5 Administrative Records

Gli **administrative records** sono ADU standard usate per fornire alcune delle funzionalità del Bundle Protocol. Ognuno di essi è costituito da un (*unsigned integer*) **record type code** e dal **record content** (in type-specific format). L'unico *record type code* valido è quello relativo allo *status report* ma tipi addizionali possono essere definiti in documenti supplementari del DTN BP.

Record Type	BPbis	RFC5050
1	Status Report	
2	Reserved for future use	Custody signal
>2	Reserved for future use	

CC

La **Custody Option**, presente nell'RFC 5050, è stata rimossa in BPbis e sostituita con la **Bundle in Bundle Encapsulation (BiBE)**.

### 3.5.1 Bundle Status Reports

La trasmissione di un **Bundle Status Report**, sotto specifiche condizioni, è un'opzione che può essere invocata quando è richiesta la trasmissione di un bundle. I report forniscono informazioni utili sui bundles che viaggiano nella rete tra cui: avvisi di ricezione, inoltro, consegna finale ed eliminazione. Sono inoltrati all'EID *report-to* e, anche se l'opzione è abilitata, l'effettiva trasmissione non è obbligatoria. Gli status report sono rappresentati come CBOR array di 6 dimensioni (se il bundle è un frammento) o 4 (altrimenti).

Field	Subfield	content
Status information	Received Forwarded Delivered Deleted	yes/no,timestamp (optional) yes/no,timestamp (optional) yes/no,timestamp (optional) yes/no,timestamp (optional)
SR reason code		vedi tabella precedente
Referred bundle identifiers	Source Generation time Sequence number [Fragment offset] [Fragment length]	

Il primo elemento dello status report array è lo **Status Information** rappresentato come CBOR array di almeno 4 dimensioni contenenti informazioni riguardanti (in questo ordine):

- Reporting node received bundle
- Reporting node forwarded the bundle
- Reporting node delivered the bundle
- Reporting node deleted the bundle

Ciascun elemento del campo *status information* è rappresentato da un CBOR array di 2 o 1 dimensioni: il primo elemento è un CBOR Boolean che indica se l'opzione è abilitata o meno; il secondo è un timestamp opzionale che indica il momento di abilitazione.

Il secondo elemento dello status report array è lo **Status Report Reason Code**, rappresentato come CBOR *unsigned integer*, che indica il valore dello **status indicator**. Gli status indicator validi sono registrati nel *IANA Bundle Status Report Reason Codes registry* del *Bundle Protocol Namespace*. I valori iniziali del registro sono elencati qui di seguito, è da tenere presente però che ulteriori reason code possono essere definiti in specifiche supplementari del protocollo DTN:

Type	BPbis	RFC5050
0	No additional information	
1	Lifetime expired	
2	Forwarded over unidirectional link	
3	Transmission canceled	
4	Depleted storage	
5	Destination endpoint ID unavailable	
6	No known route to destination from here	
7	No timely contact with next node on route	
8	Block unintelligible	
9	Hop limit exceeded	Reserved
10	Traffic pared (e.g., status reports)	Reserved
>10	Unassigned/Reserved	

Il terzo elemento dell’array è il *source node ID* identificante il nodo di origine del bundle il cui stato viene segnalato.

Il quarto elemento dell’array è il *creation timestamp* del bundle il cui stato viene segnalato.

Il quinto elemento dell’array è presente se e solo se il bundle il cui stato viene segnalato conteneva l’offset di un frammento. Se presente deve essere l’*offset del frammento* del suddetto bundle rappresentato come CBOR *unsigned integer*.

Il sesto elemento dell’array è presente se e solo se il bundle il cui stato viene segnalato conteneva l’offset di un frammento. Se presente deve essere la *lunghezza del payload* del suddetto bundle rappresentato come CBOR *unsigned integer*.

### 3.6 CC’s Notes on bundle identification

Un bundle, che non sia un frammento, è identificato in un administrative record da tre variabili:

- Source EID
- Bundle’s creation time
- Sequence number

Un frammento (quindi un bundle con il fragment bit negli administrative record flags settato a 1) è identificato aggiungendo due ulteriori variabili:

- Fragment offset
- Fragment length (non la lunghezza totale del payload originale)

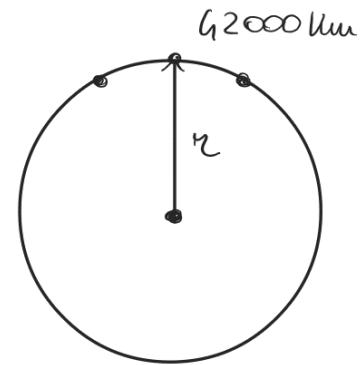
RAGGIO ORBITALE  $\rightarrow$  62000 Km

RAGGIO EQUATORE  $\rightarrow$  6378

GEO ALTITUDE  $\rightarrow$  35.700 Km

## Chapter 4

# Satellites Networks



### 4.1 Introduction

#### 4.1.1 Geostationary Earth Orbit (GEO)

La **Geostationary Earth Orbit (GEO)** è un'orbita ellittica situata 35,786 km sopra l'equatore. Un oggetto posto in tale orbita ha un periodo di rotazione uguale a quello della Terra, perciò, tale oggetto (in movimento) appare fisso ad un osservatore terrestre. Satelliti per le comunicazioni e meteorologici sono spesso posizionati in orbita geostazionaria poiché, così facendo, le antenne terrestri con le quali comunicano non devono modificare la loro posizione per seguirli e possono rimanere fisse ad osservare un singolo punto nel cielo. È possibile ricavare il raggio uguagliando la forza di attrazione gravitazionale alla forza centripeta:

Per far ruotare un satellite  
attorno ad un corpo si:  
usa la forza centripeta

$$F_{\text{newton}} = G \frac{m_{\text{sat}} m_{\text{earth}}}{r^2}$$

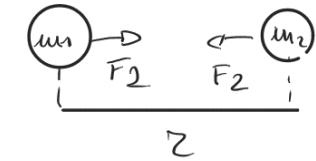
$$F_{\text{centripetal}} = m_{\text{sat}} \omega^2 r$$

$$F_{\text{newton}} = F_{\text{centripetal}}$$

$$G \frac{m_{\text{earth}}}{r^2} = \omega^2 r$$

$$\frac{\mu}{r^3} = \omega^2$$

$$r = \sqrt[3]{\frac{\mu}{\omega^2}}$$



$$F_1 = F_2 = G \frac{m_1 \times m_2}{r^2}$$

dove  $G$  è la costante gravitazionale universale e  $\mu = G m_{\text{earth}}$  il parametro gravitazionale standard della Terra.

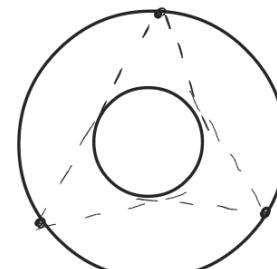
Per coprire l'intero suolo terrestre sono sufficienti tre satelliti in orbita geostazionaria a  $120^\circ$  di inclinazione l'uno dall'altro. Se da un lato risulta vantaggioso posizionare satelliti in questa orbita, dall'altro si deve tenere conto degli svantaggi. L'elevata distanza terra-satellite comporta un'attenuazione dei segnali non indifferente e i ritardi sono elevati. Il tempo di propagazione terra-satellite è di circa 125 millisecondi, ciò vuol dire che il *round trip time* (RTT) si aggira sul mezzo secondo. Decrescendo inoltre l'angolo di inclinazione antenna-terra al decrescere della latitudine, le regioni polari risultano difficili da coprire.

#### 4.1.2 Low Earth Orbit (LEO)



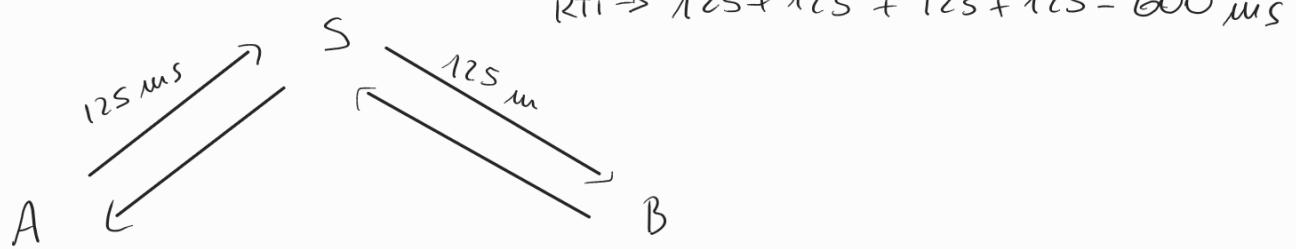
La **Low Earth Orbit (LEO)** è l'insieme di orbite ellittiche situate tra i 160 e i 2000 km sopra l'equatore. Un oggetto posto in tale orbita ruota almeno 11.25 volte al giorno attorno alla Terra, perciò, tale oggetto appare come un punto in rapido movimento ad un osservatore terrestre. Molti dei satelliti artificiali sono posizionati in orbita LEO poiché, oltre ad avere un minor costo di lancio, garantiscono bassa attenuazione e tempo di propagazione dei segnali. Viaggiando a velocità elevate, per avere una buona copertura globale, è necessario costruire una costellazione di decine di satelliti. In un'orbita LEO **olare** è possibile coprire l'intera superficie globale utilizzando un solo satellite, chiaramente, la copertura non sarà simultanea in tutte le aree.

Vantaggi  $\rightarrow$  no TRAIUNO necessario  
3 satelliti per coprire la terra

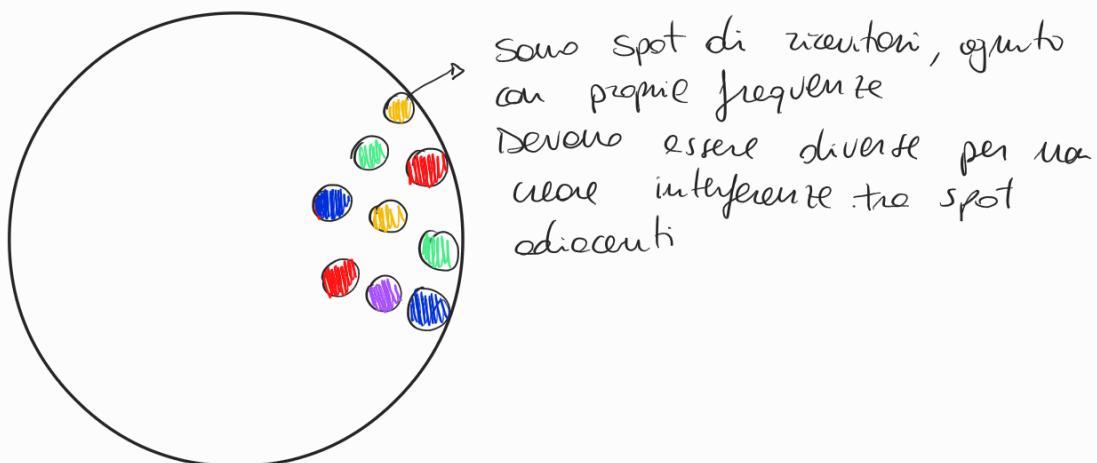
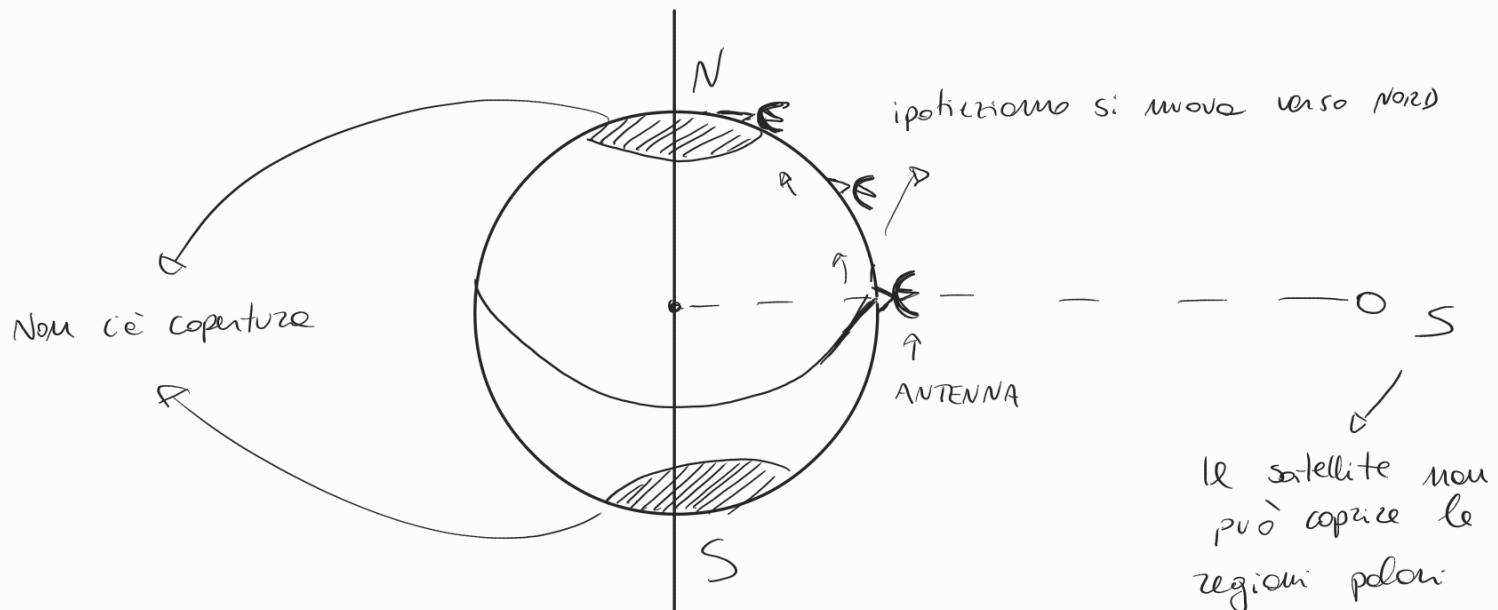


Svantaggi  $\rightarrow$  grandi distanze, tenere di più area molto secondo  
L'inclinazione dell'antenna diminuisce con la latitudine  
Poco copertura dei poli

Perde mille secondi di RTT?

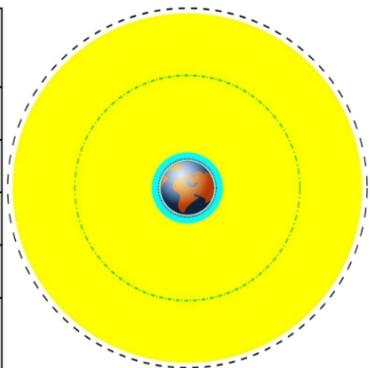


CAMBIO ANGOLO CON LATITUDINE



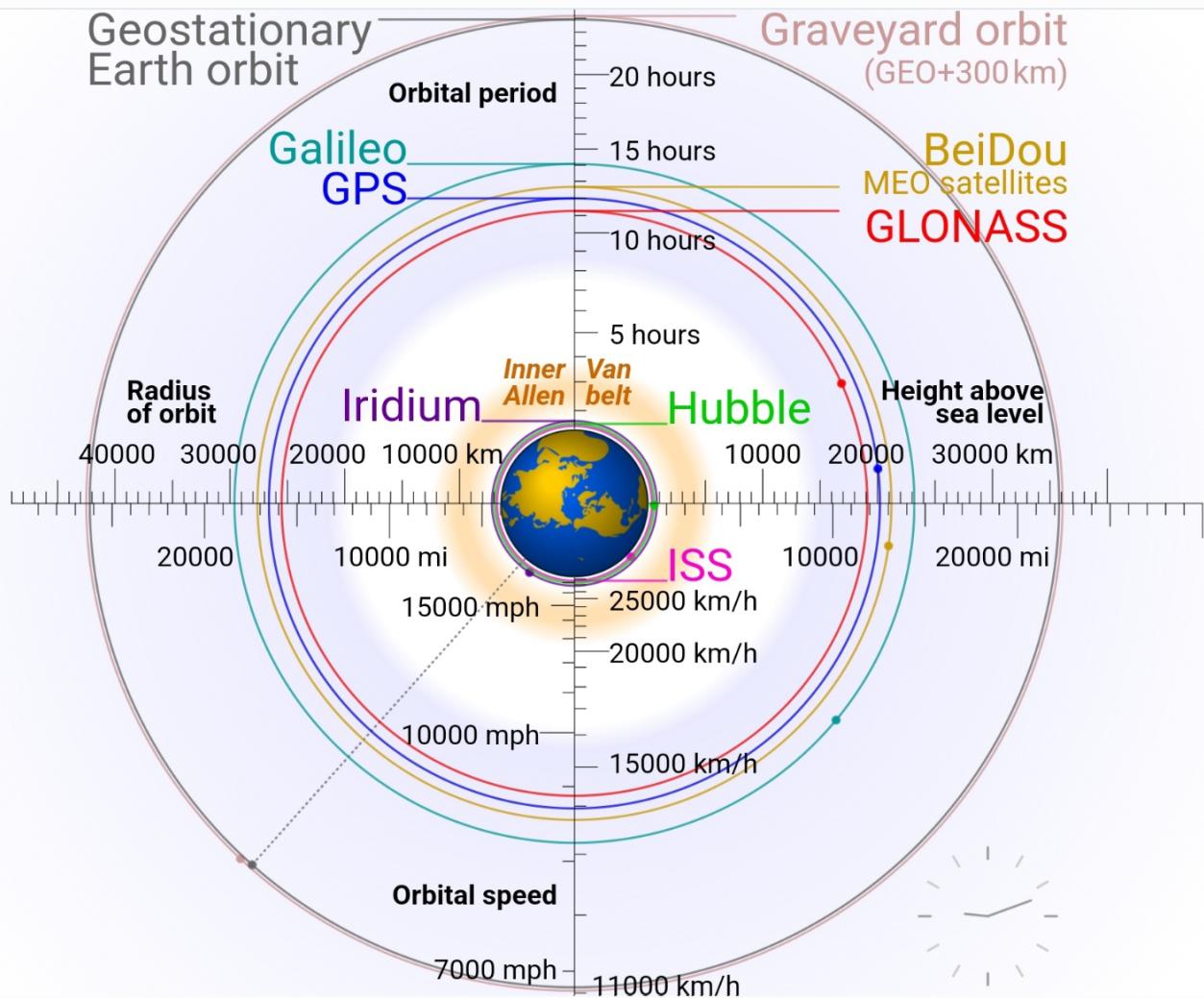
# Distances from Earth

English	Marker	Distance above earth (km)	Distance from center of earth (km)
<u>Earth</u>	Blue/brown image	0	6370
<u>Low Earth Orbit (LEO)</u>	Cyan area	160 to 2,000	6,530 to 8,370
<u>Medium Earth Orbit (MEO)</u>	Yellow area	2,000 to 34,780	8,370 to 41,150
<u>International Space Station (ISS)</u>	Red dotted line	370	6,741
<u>Global Positioning System (GPS) satellites</u>	Green dash-dot line	20,230	26,600
<u>Geostationary Orbit (GEO)</u>	Black	35,794	42,164



[https://en.wikipedia.org/wiki/Low\\_Earth\\_orbit](https://en.wikipedia.org/wiki/Low_Earth_orbit)

I dispositivi GPS fanno più velocità vicini



Satelli GEO devono essere lungo l'equatore  
I satelli LEO sono lungo i poli

## \* LEO

Con un orbita polare, basta un satellite per coprire la terra.

Ogni 12 ore coprono lo stesso area

Nel caso di download di info da un satellite, servono le BASE STATION, che sono geolocalizzate in diverse longitudini.

Possiamo vedere un satellite di questo tipo per circa 5-9 minuti, proprio perché sono molto veloci.

---

Adesso sta nascendo una comunicazione "ottica" tramite laser, molto veloci, ma ci sono problemi come le nuvole. Possono comunicare con questo metodo anche satelliti LEO e GEO

## 4.2 DTN as an evolution of TCP-splitting PEPs

Le comunicazioni satellitari sono soggette ai problemi presentati nelle sezioni delle orbite LEO e GEO. Una possibile soluzione è la modifica del livello di trasporto nello stack di protocolli, soluzione che, sebbene possibile, non è adatta all'utilizzo generale di Internet. Dal momento che i client satellitari sono una nicchia per i fornitori di contenuti, non esiste un reale vantaggio per tali provider di modificare lo stack standard dei protocolli per offrire una migliore Quality of Service (QoS).

Per utilizzare varianti del protocollo di trasporto adatte ai collegamenti satellitari, si utilizzano i PEPs. I **Performance Enhancing Proxies (PEPs)** (letteralmente: proxies che migliorano le prestazioni) sono nodi intermedi inseriti ad uno solo (**integrated PEP**, figura 4.1b) o, più frequentemente, ad entrambi (**distributed PEPs**, figura 4.1a) gli estremi della connessione. I PEPs, basati sulla tecnica di TCP-splitting, suddividono la connessione TCP in due (integrated) o tre (distributed) connessioni TCP, così facendo è possibile utilizzare varianti del protocollo TCP adatte ai collegamenti satellitari. L'introduzione dei PEPs fornisce un'effettiva soluzione che porta dei vantaggi per l'end user, d'altro canto però introduce anche svantaggi relativi alla sicurezza (incompatibilità con IPsec) violando la semantica end-to-end garantita dal livello di trasporto.

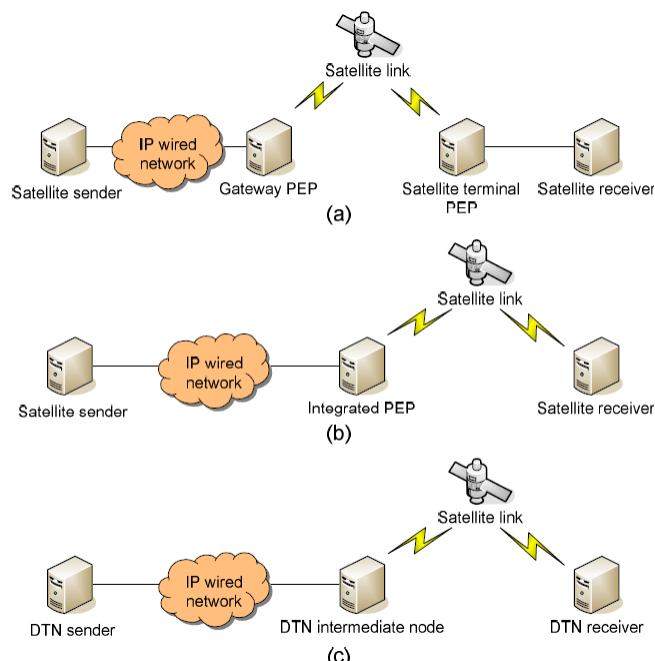


Figure 4.1: PEP and DTN architecture comparison: a) distributed PEP; b) integrated PEP; c) DTN network.

Nei *distributed PEPs* le connessioni TCP presenti sono tre: satelliteSender-PEPgateway; PEP-PEP; satelliteTerminalPEP-satelliteReceiver. La prima e l'ultima, solitamente, sono realizzate su cavi fisici, per cui è possibile utilizzare il protocollo TCP ordinario (ad esempio NewReno). Quella intermedia utilizza invece versioni di TCP differenti (o un altro protocollo di trasporto) specializzate per connessioni satellitari.

Negli *integrated PEP* le connessioni TCP presenti sono solamente due: tra il satellite mittente e l'integrated PEP si utilizza una connessione su cavo con TCP standard, per il collegamento satellitare si utilizza una versione di TCP modificata.

Per entrambe le architetture è possibile mostrare un'architettura DTN corrispondente che utilizza un Convergence Layer Adapter per TCP. Nel seguito si focalizza l'attenzione sull'architettura integrated, che meglio si adatta ad un confronto diretto.

Una rete DTN che corrisponde ad un integrated PEP è mostrata in figura 4.1c, i corrispondenti stack di protocolli invece, sono in figura 4.2.

Dalla comparazione tra le diverse architetture si evidenziano i seguenti punti salienti:

- Entrambe hanno due connessioni al livello di trasporto, la primo con cavo fisico, la seconda con satellite;
- Entrambe possono usare varianti di TCP adatte ai collegamenti satellitari;
- La soluzione DTN non è poi così trasparente: il Bundle Protocol deve essere necessariamente installato sugli end-nodes;
- La tecnica di TCP-splitting viola la semantica end-to-end di TCP perché i PEPs intermedi devono necessariamente operare a livello di trasporto e applicazione, mentre

lo stack di protocolli riserva queste funzionalità solo all'end-node. In DTN questo inconveniente è evitato, poiché il ruolo di TCP è ridefinito dall'inserimento del Bundle Protocol;

- La tecnica di TCP-splitting è incompatibile con IPsec;

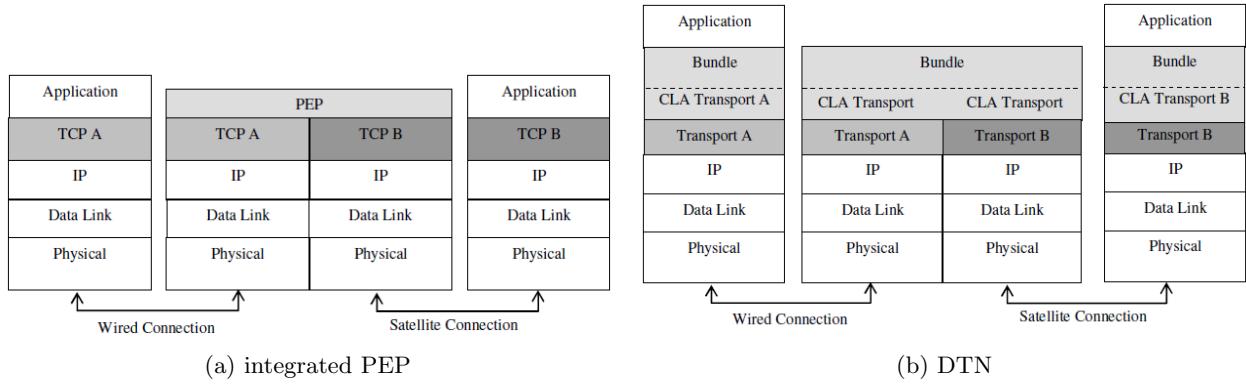


Figure 4.2: PEP and DTN protocol stack comparison

### 4.3 DTN and GEO/LEO Satellite Communications

L'architettura DTN può essere usata per: satelliti GEO con terminali fissi, GEO con terminali mobili, satelliti LEO. Questi tre scenari sono analizzati separatamente nel seguito. Nei satelliti GEO con terminali fissi, generalmente è presente una connessione end-to-end continua, alternative alle DTN come PEPs sono disponibili e offrono buone performance. Nel secondo scenario con i terminali in movimento, l'utilizzo delle DTN è una possibile soluzione, ma i vantaggi rispetto ai PEPs devono essere valutati attentamente. Il terzo scenario presenta l'ambiente di sfida più difficile, poiché è caratterizzato da connessioni end-to-end intermittentи o, nel caso più estremo, totalmente assenti (ad esempio quando i dati sono trasferiti da una sorgente ad una destinazione che non saranno mai visibili allo stesso tempo).

Senza perdita di generalità, ci si focalizza inizialmente sul caso di utente fisso connesso a Internet attraverso un satellite GEO. I ritardi sono elevati (RTT=600ms), è possibile che la rete si congestioni e i segmenti possano perdere a causa di errori sul canale satellitare (i pacchetti con uno o più bit corrotti sono scartati). Questo scenario, è il meno favorevole alle DTN perché, senza mobilità dei terminali, si può ragionevolmente assumere che il collegamento satellitare presenti poche interruzioni e un cammino continuo con gli end-nodes sia sempre disponibile. E' stato mostrato che l'approccio con le DTN rappresenta comunque una soluzione competitiva rispetto a PEPs. A seguire si presentano i risultati addizionali ottenuti con versioni più recenti della stessa testbed.

#### 4.3.1 TATPA Testbed

La **Testbed on Advanced Transport Protocols and Architectures (TATPA)** in figura 4.3 riproduce le caratteristiche di una rete eterogenea che include un collegamento satellitare. È basata su un insieme di PC Linux con kernel 2.26 patchato con il Multi TCP package che implementa TCP-Hybla.

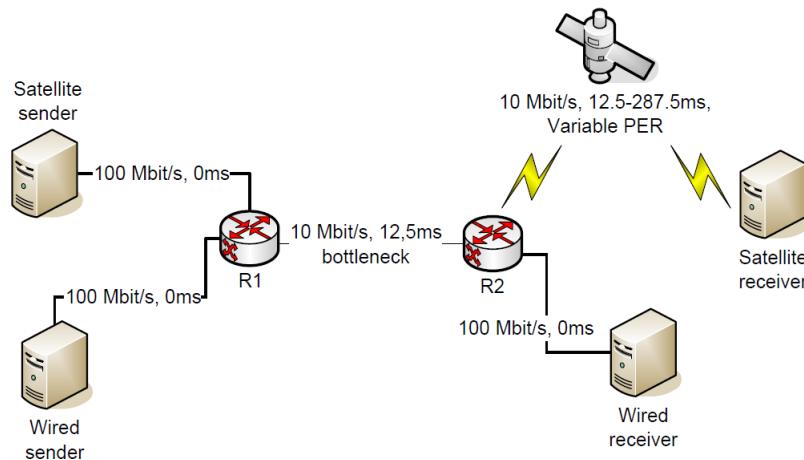


Figure 4.3: Logical layout of the TATPA testbed

Un’implementazione Linux del TCP-splitting PEP, chiamata **PEPsal**, può essere abilitata nel router R2 in conformità alla topologia in figura 4.1b. TCP A e TCP B della figura 4.2a sono rispettivamente nelle versioni NewReno e Hybla. La configurazione DTN corrispondente (figura 4.1c) è ottenuta installando l’implementazione DTN2 sul **satellite-sender**, sul **satellite-receiver** e sul router R2 (che in questo caso funge da nodo DTN intermedio). Per equità nella comparazione con PEPsal, Trasport A e Transport B della figura 4.2b sono rispettivamente NewReno e Hybla. Il *satellite emulator* simula un ritardo (di 287.5ms one-way per GEO) e un **Packet Error Rate (PER, 0% o 1% nei test)**. L’interruzione del canale satellitare è emulata dalle tracce reali. Per la valutazione delle performance DTN è stato usato il tool DTNperf\_2 incluso nel package DTN2

#### 4.3.2 GEO satellites with fixed terminals

Il test valuta le performance ottenibili su 180 secondi di unità dati satellitari trasferite in termini di **goodput**, cioè, l’ammontare di dati del livello applicativo trasferite per unità di tempo. Le tecniche comparate sono: end-to-end NewReno, end-to-end Hybla, PEPsal e DTN. Gli ambienti operativi, al crescere del livello di sfida sono: ideale, congestionato, non-zero PER e non-zero PER congestionato. I quattro scenari sono analizzati singolarmente e fanno tutti riferimento alla figura 4.4.

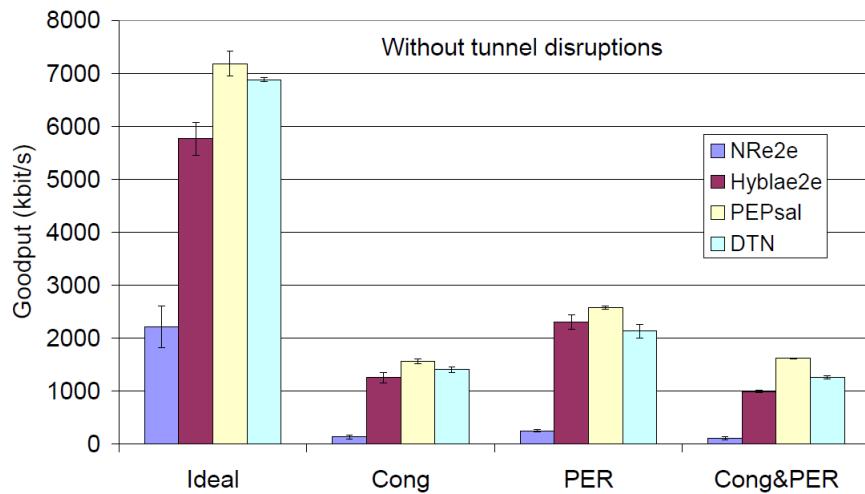


Figure 4.4: GEO satellite with fixed users (no disruptions); goodput of a single satellite connection (RTT=600ms); averaged values, 90% confidence intervals

##### 4.3.2.1 Ideal channel

Questo caso base prevede solo un lungo RTT: 25ms sul collegamento R1-R2, 575ms sul collegamento satellitare, per un totale di 600ms. NewReno non è in grado di sfruttare completamente la banda satellitare di 10 Mbit/s. Hybla funziona meglio (5.8Mbit/s) poiché è specificatamente progettato per RTT lunghi. Le performance migliori sono ottenute con PEPsal, che trae il vantaggio sia di Hybla che di un RTT ridotto sulla connessione satellitare. DTN raggiunge le medesime performance di PEPsal, per via della similarità tra le due architetture.

##### 4.3.2.2 Congestion

Questo scenario mostra il problema dell’*RTT unfairness*, tipico quando si combinano collegamenti cablati a collegamenti satellitari. Il trasferimento dati satellitare (RTT=600ms) è gravemente compromesso da 5 RTT brevi (25ms) della rete IP cablata (cioè sul collegamento R1-R2). L’impatto su NewReno è drammatico, infatti il goodput è vicino allo zero. Viceversa, le altre tecniche si avvicinano al *maximum fair share*, ovvero il collo di bottiglia dato da banda/numeroConnessioni ( $10\text{Mbps} / 6 = 1.66 \text{ Mbit/s}$ ) che si può considerare come target ideale. Questo dovrebbe essere attribuito per Hybla al suo controllo di congestione migliore, mentre per PEPsal e DTN all’isolamento del collegamento satellitare dal collegamento cablato della rete IP.

##### 4.3.2.3 PER

Lo scenario è simile a quello del canale ideale, qui però il collegamento satellitare è affetto da una grande perdita di pacchetti (PER=1%) (ad esempio dovute a condizioni di propagazione

pessime). Sebbene l'origine del problema sia differente, i risultati sono abbastanza vicini al caso di congestione: NewReno è prossimo allo zero, le altre tecniche sono migliori anche se la banda totale disponibile non è raggiunta a causa dell'elevato PER. A differenza del caso con congestione però, PEPsal e DTN hanno performance migliori, dovute principalmente all'utilizzo di una variante di TCP ottimizzata per le connessioni satellitari, e non per l'isolamento dalla rete IP cablata.

#### 4.3.2.4 PER and congestion

E' presente contemporaneamente congestione e PER. I risultati, vicini al caso di sola congestione, mostrano come questa sia dominante sul degrado delle prestazioni.

#### 4.3.3 GEO satellite with mobile terminals

Il test, simile al precedente, introduce la mobilità dei satelliti che porta all'interruzione del canale. I vantaggi dell'utilizzo delle DTN dipendono dalla durata e dalla frequenza di tale interruzione. Con il Bundle Protocol che utilizza un TCP CLA, e con l'implementazione di riferimento DTN2, si deve distinguere tra interruzioni *short* ( $\leq 30s$ ) e *long* ( $> 30s$ ). La soglia di 30 secondi è impostata in accordo all'implementazione standard di DTN2, anche se l'RFC non specifica questo valore.

Le interruzioni short non necessitano di una risposta dal BP e sono direttamente gestiti dalla ritrasmissione TCP. Al contrario, le interruzioni long fanno sì che il BP chiuda la connessione TCP interrotta e faccia una serie di tentativi per aprire una nuova connessione TCP. La chiusura forzata della connessione TCP a sua volta innescava la reactive bundle fragmentation, quindi, il bundle è suddiviso in due frammenti: il primo è costituito dai dati inviati correttamente prima dell'interruzione, il secondo dai dati rimanenti da inviare non appena una nuova connessione TCP è stabilita.

Disruption	Lenght	TCP (e-2-e or with PEP)	DTN Bundle protocol with TCP CLA
Short	$< 30s$ (DTN2 default)	Tackled by TCP	Tackled by TCP
Long	$>30s$ (DTN2 def.) and $< 1200 s$ (TCP Linux def.)	Tackled by TCP	Tackled by DTN
Very long	$> 1200 s$ (TCP Linux def.) $< 24 h$ (DTN2 def.)	TCP failure	Tackled by DTN
Extremely long	$> 24 h$ (DTN2 def.)	TCP failure	DTN failure

Le gallerie delle autostrade forniscono un pratico esempio di interruzioni indotte dalla mobilità dei terminali. Si focalizza l'attenzione sulla ferrovia *Direttissima* Bologna-Firenze lunga 96km, con 33 gallerie per una lunghezza complessiva di 33km, pari al 39% del totale. Assumendo una velocità costante di 120km/h, si hanno 30 interruzioni short e 3 interruzioni long (214s, 92s, 553s).

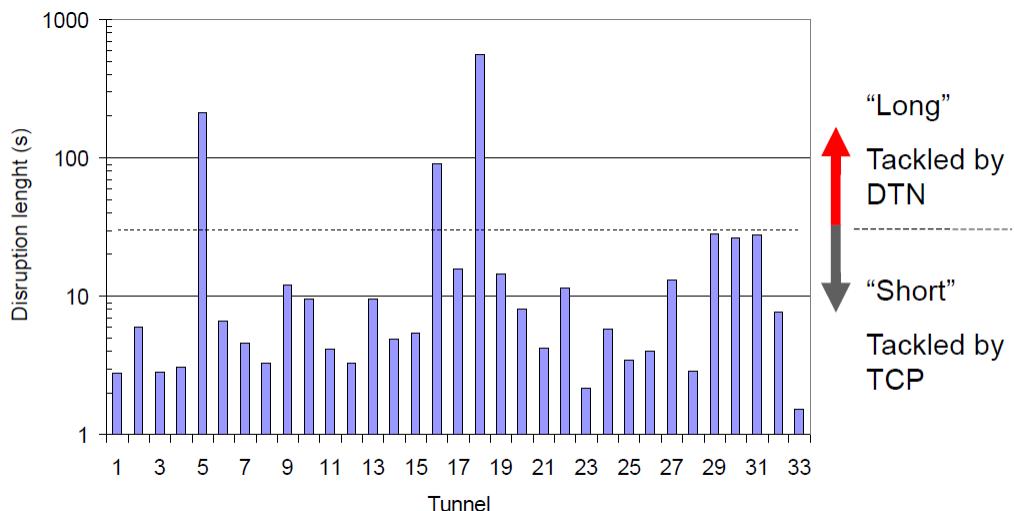


Figure 4.5: Disruption lengths caused by tunnels in the Bologna-Florence "Direttissima" railway line, assuming a train speed of 120 km/h; the dotted line is the 30s threshold

Lo scenario del test prevede il trasferimento file da un server ad un satellite a bordo di un treno in movimento da Bologna a Firenze. La topologia della rete e le tecniche comparate sono le medesime del precedente paragrafo, con l'aggiunta della mobilità del terminale. I risultati sono mostrati in figura 4.6.

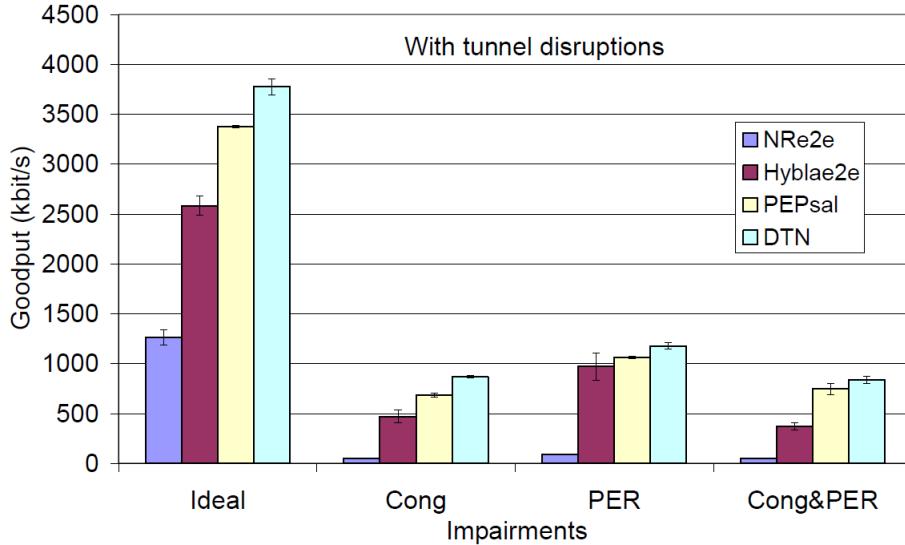


Figure 4.6: GEO satellite with mobile users (disruptions due to railway tunnels); goodput of a single satellite connection (RTT=600ms); averaged values, 90% confidence intervals.

Comparando questi risultati con quelli ottenuti per terminali fissi, si giunge a due conclusioni. Primo, le performance peggiorano in tutti i casi, il che risulta abbastanza ovvio dall'indisponibilità del canale satellitare per il 39% del tempo a causa delle gallerie. Meno evidente è che a questo tempo bisogna aggiungere il *restart delay*, dopo il quale TCP e il BP sono in grado di riavviare la trasmissione a seguito dell'interruzione. Si aggiunge inoltre il tempo necessario a TCP per raggiungere lo *steady state*. Più è lungo l'RTT, peggiore è questo effetto. La seconda conclusione è che il comportamento qualitativo è lo stesso del caso senza interruzioni, stavolta però, è DTN ad avere (lievi) migliori performance rispetto a PEPsal.

Considerando una velocità del treno di 60km/h, l'interruzione più lunga è di 1200s e coincide con la *maximum tolerable disruption lenght* del TCP Linux di default. In questo caso, tutte le tecniche tranne DTN abortirebbero il trasferimento del file.

In conclusione, con i terminali mobili, le DTN possono risultare vantaggiose anche in termini di goodput, vantaggi che però dipendono, come già detto, dalla durata e dalla frequenza dell'interruzione. Per le *very long*, le DTN sono decisamente migliori.

#### 4.3.4 LEO satellites

Dai paragrafi precedenti, è noto come sia necessaria una costellazione di satelliti LEO per fornire un'adeguata copertura del segnale su tutto il globo e di conseguenza una connessione continua. Utilizzando un singolo satellite, si ottiene una connessione intermittente, ed è questo il caso su cui ci si focalizza poiché di maggior interesse dal punto di vista delle DTN. Si consideri un satellite LEO e una stazione a terra: il satellite passa sopra la stazione stabilendo una comunicazione (*contact*) solo per brevi periodi (*contact window*). La breve finestra di contatto e la banda limitata, incidono sul **contact volume**, ovvero i dati utili che è possibile inviare. Se un file di grandi dimensioni (immagine) non può essere inviato durante un singolo passaggio, è necessario suddividerlo in segmenti ed inviarlo in passaggi consecutivi. In questa situazione, è possibile usare la *proactive fragmentation* del Bundle Protocol, la quale, suddivide il payload in più bundle di grandezza prefissata (e si fa in modo che coincida con il contact volume).

##### 4.3.4.1 LEO data mule

Si consideri lo scenario in figura 4.7: la **Source Station (SS)** e la **Destination Station (DS)** sono entrambe stazioni di terra sufficientemente distanti da non trovarsi contemporaneamente nell'area di copertura del satellite LEO. In altre parole, non c'è mai connessione diretta tra sorgente e destinazione. Questa situazione è la più favorevole per le DTN, poiché, l'assenza di una connessione end-to-end previene l'instaurazione di connessioni TCP (o TCP-like) tra le stazioni di terra. Inoltre, l'assenza di connessioni end-to-end rende impossibili trasferimenti UDP sprovvisti di storage *in-network*. L'unico approccio possibile è quello dello

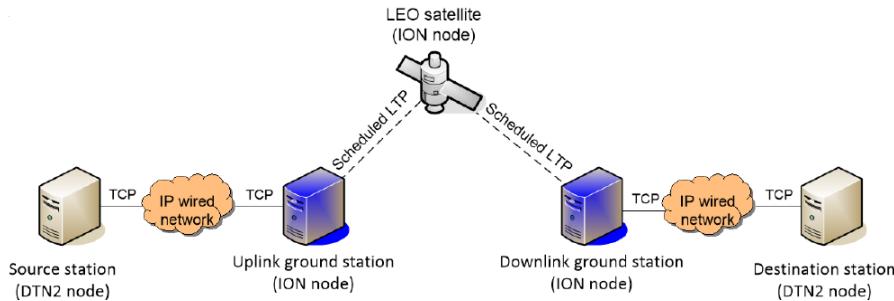


Figure 4.7: LEO satellite: data mule

store-and-forward di grandi quantità di dati sul satellite, schematizzabile in tre fasi: (1) la **Uplink Ground Station (UGS)** è connessa al satellite e i dati sono spostati a bordo di esso; (2) né la UGS né la **Downlink Ground Station (DGS)** sono connessi al satellite e tutti i dati sono salvati nella sua memoria locale (persistente); (3) la DGS è connessa e i dati sono scaricati dal satellite.

L'esperimento condotto prevede il trasferimento di un file di 20MB, suddiviso in 100 bundles da 50kB ciascuno, i risultati sono illustrati in figura 4.8.

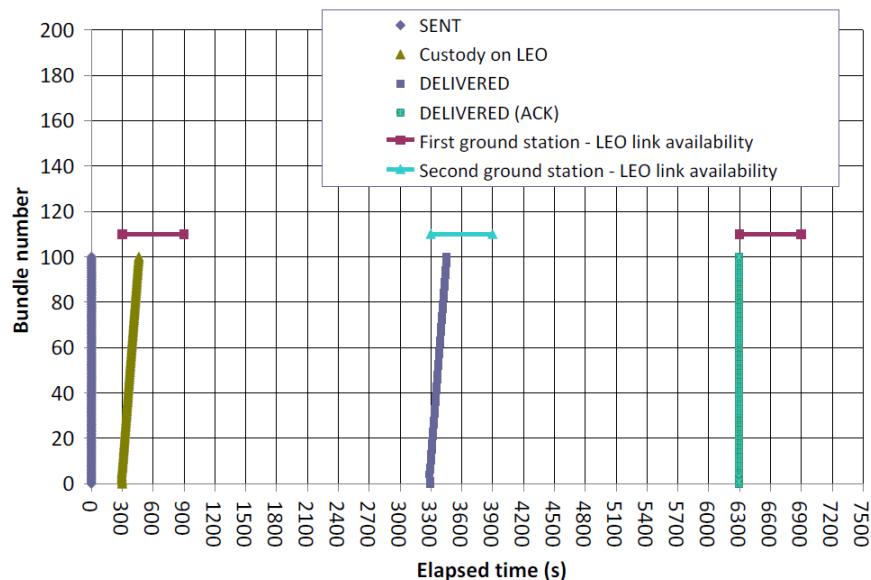


Figure 4.8: LEO satellite: 20 MB file transfer; 100 bundles, 50 kB each; DTN BP Status report logs collected by DTNperf server

Il grafico mostra il numero di bundles trasferiti nel tempo tra il satellite e le due stazioni di terra, quindi, non riporta i trasferimenti nei tratti SS-UGS, DS-DGS e viceversa essendo situati su collegamenti cablati:

- generazione bundles:** i 100 bundles sono generati quasi istantaneamente dalla SS (infatti la linea appare verticale al tempo t=0), presi in carico dal BP e consegnati alla UGS;
- contatto UGS-satellite:** i 100 bundles sono inviati dalla UGS al satellite;
- satellite isolato** dalle due stazioni di terra mentre gira attorno alla terra, i bundles sono salvati nella sua memoria locale;
- contatto satellite-DGS:** i 100 bundles sono scaricati dal satellite alla DGS e consegnati alla DS. Subito dopo, la DS invia 100 ACK alla DGS che, a sua volta, invia al satellite (non illustrato nel grafico);
- satellite isolato** dalle due stazioni di terra mentre gira attorno alla terra, gli ACK sono salvati nella sua memoria locale;
- contatto UGS-satellite:** gli ACK sono istantaneamente scaricati dal satellite alla UGS (infatti la linea appare verticale al tempo t=6300) e in seguito consegnati alla SS;

Sebbene applicato all'ambiente satellitare, questo è un tipico esempio di **data-mule communication**, un task per cui il Bundle Protocol è stato progettato. L'unica alternativa alle DTN in questo scenario sarebbe l'upload/download manuale di file o un'applicazione specifica che includa le funzionalità del BP.

#### 4.3.4.2 LEO Earth Observation<sup>1</sup>

Si consideri lo scenario in figura 4.9a: si vuole inviare una fotografia scattata dal satellite LEO all'**ESA Mission Control Center** sfruttando tre stazioni di terra (GS1, GS2, GS3) situate a diverse longitudini. All'interno di ciascun nodo è presente l'ipn (EID adottato dalla NASA nella implementazione ION) corrispondente. Le linee tratteggiate indicano collegamenti spaziali intermittenti, quelle continue invece i collegamenti terrestri non intermittenti.

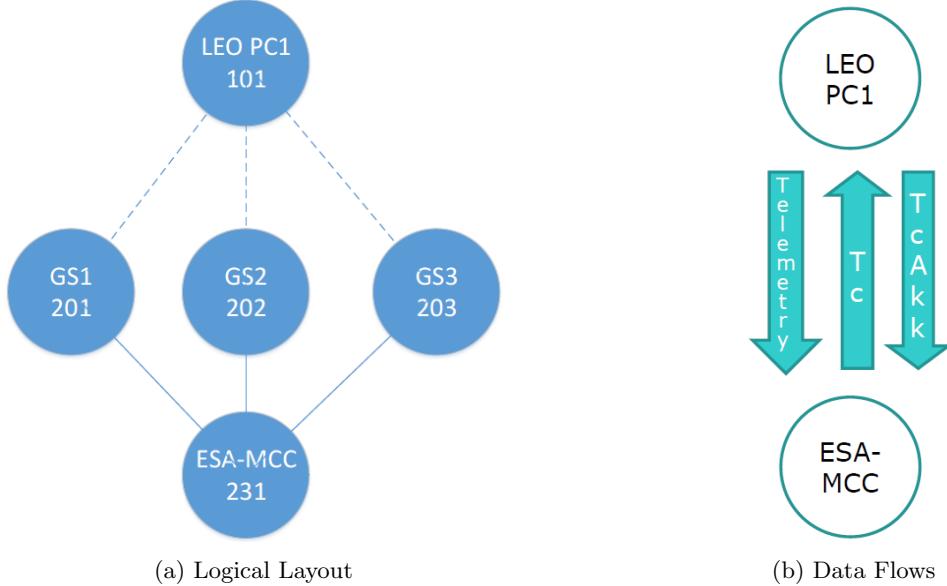


Figure 4.9: Scenario A

In figura 4.9b sono mostrati i flussi dati che satellite e MCC scambiano tra loro. La **Telemetry** (non è la fotografia) contiene dati generati sul satellite LEO, utili all'MCC a monitorarne costantemente lo stato (tipo la temperatura rilevata dal satellite), generalmente sono bundles di piccole dimensioni. In direzione opposta ci sono i **TeleCommands**, ordini che l'MCC impedisce al satellite (tipo, fai una fotografia quando sorvoli una determinata regione). Infine ci sono i **TeleCommands Acknowledgement** per confermare la ricezione dei Tc dal satellite all'MCC. Nel data flow è omesso lo schema di download dell'immagine presente nell'esperimento originale, in quanto troppo complesso. Le istanze DTN presenti nei due nodi comunicanti sono complessivamente 6:

- **MCC:** 2 server (ricevente) uno per i flussi di Telemetry, uno per i flussi di TcACK. 1 client (mittente) per il flusso di Tc.
- **LEO:** 2 client (mittenti) uno per i flussi di Telemetry, uno per i flussi di TcACK. 1 server (ricevente) per i flussi di Tc.

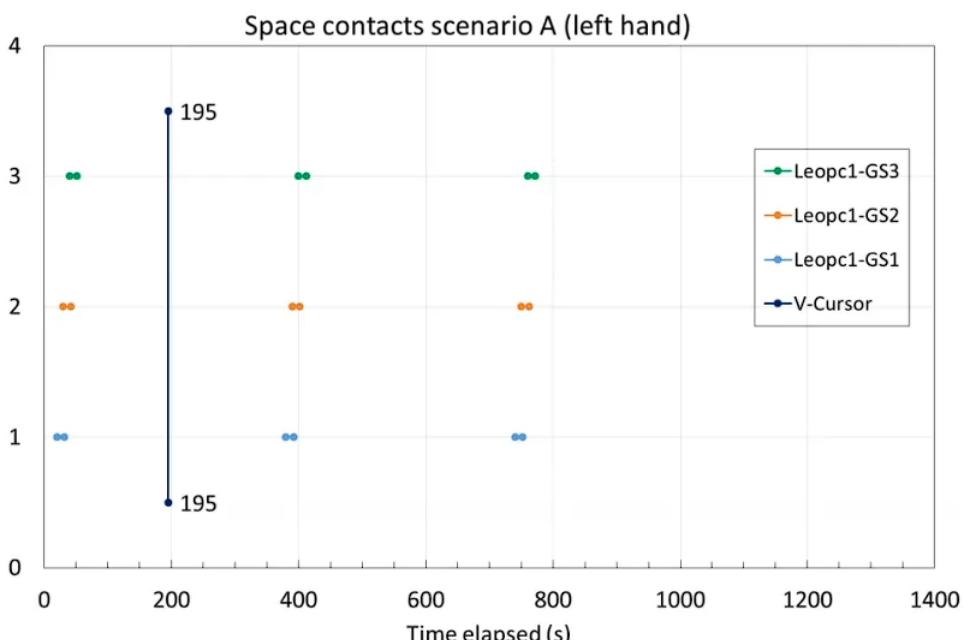


Figure 4.10: Scenario A: Contacts (time scaled: s instead of minutes)

<sup>1</sup>L'esperimento è stato discusso a lezione ed è una versione semplificata rispetto a quella presente nelle slides.

In figura 4.10 sono illustrati i contatti (molto brevi) tra le GS e il satellite LEO: il contatto della GS1 è immediatamente seguito dal contatto con GS2 e, a sua volta, da quello con GS3. Questo scenario non denota una situazione ottimale poiché, trovandosi le tre GS a longitudini simili, per lunghi periodi di tempo non è possibile comunicare con il satellite. Il segmento con 195 agli estremi è un cursore utile nel file originale dei dati ad ottenere maggiori informazioni.

I risultati dell'esperimento sono stati ottenuti utilizzando DTNperf e sono illustrati in figura 4.11. Gli status report contenuti in ogni riga del file sono stati in seguito elaborati per ottenere una rappresentazione grafica (figura 4.12) dalla quale si estrapolano le considerazioni seguenti.

D	E	F	G	H	I	J	K	L	M	N	O	P
Report_SC	Report_Ty	Bndl_SRC	Bndl_TST	Bndl_SQN	Bndl_FO	Bndl_FL	Div	Ct	Rcv	Fwd	Del	Reason
1	Report_SC	Report_Ty	Bndl_SRC	Bndl_TST	Bndl_SQN	Bndl_FO	Bndl_FL	Div	Ct	Rcv	Fwd	Del
2	1	STATUS_R	ipn:231.1268	569783274	1	0	0	569783277	569783277			no addition
3	1	STATUS_R	ipn:101.1267	569783274	1	0	0	569783278	569783278			no addition
4	2	STATUS_R	ipn:101.1267	569783274	2	0	0	569783278	569783278			no addition
5	2	STATUS_R	ipn:102.1266	569783274	2	0	0	569783310	569783310			no addition
6	1	STATUS_R	ipn:102.1266	569783274	1	0	0	569783314	569783314			no addition
7	1	STATUS_R	ipn:231.1268	569783274	1	0	0		569783275			no addition
8	1	STATUS_R	ipn:101.1267	569783274	1	0	0		569783278			no addition
9	2	STATUS_R	ipn:101.1267	569783274	2	0	0		569783278			no addition
10	1	STATUS_R	ipn:102.1266	569783274	2	0	0		569783307			no addition
11	9	STATUS_R	ipn:102.1266	569783274	1	0	0		569783307			no addition
12	1	STATUS_R	ipn:102.1266	569783277	1	0	0	569783310	569783310			no addition
13	1	STATUS_R	ipn:102.1266	569783277	1	0	0		569783306			no addition
14	1	STATUS_R	ipn:101.1267	569783278	1	0	0	569783279	569783279			no addition
15	3	STATUS_R	ipn:102.1266	569783278	1	0	0	569783310	569783310			no addition
16	1	STATUS_R	ipn:101.1267	569783278	1	0	0		569783279			no addition
17	2	STATUS_R	ipn:102.1266	569783278	1	0	0		569783307			no addition
18	1	STATUS_R	ipn:102.1266	569783280	1	0	0	569783318	569783318			no addition
19	1	STATUS_R	ipn:102.1266	569783280	1	0	0		569783308			no addition
20	1	STATUS_R	ipn:101.1267	569783282	1	0	0	569783283	569783283			no addition
21	4	STATUS_R	ipn:102.1266	569783282	1	0	0	569783310	569783310			no addition
22	1	STATUS_R	ipn:101.1267	569783282	1	0	0		569783283			no addition
23	3	STATUS_R	ipn:102.1266	569783282	1	0	0		569783307			no addition
24	1	STATUS_R	ipn:102.1266	569783283	1	0	0	569783311	569783311			no addition
25	2	STATUS_R	ipn:102.1266	569783283	1	0	0		569783311			no addition
26	1	STATUS_R	ipn:231.1268	569783286	1	0	0	569783287	569783287			no addition
27	1	STATUS_R	ipn:101.1267	569783286	1	0	0	569783290	569783290			no addition
28	2	STATUS_R	ipn:101.1267	569783286	2	0	0	569783290	569783290			no addition

Figure 4.11: Scenario A: results (status reports collected by DTNperf in a .csv file)

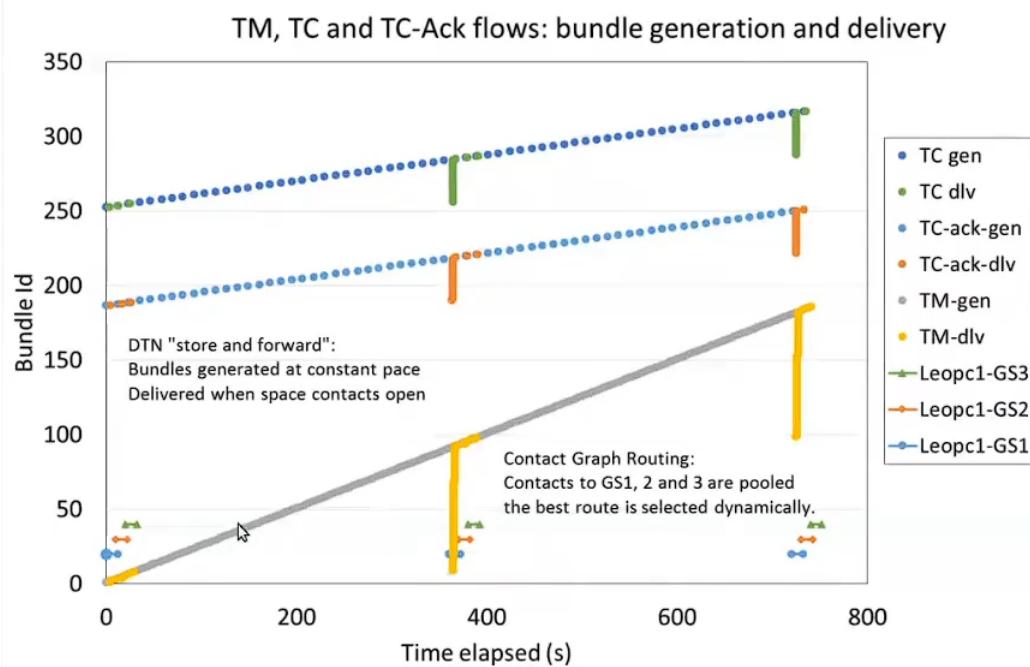


Figure 4.12: Scenario A: analysis of flows

Si consideri inizialmente il flusso dati relativo alla Telemetria (segmento più in basso):

1. al tempo  $t=0$ , inizia la generazione dei bundles di telemetria, nello stesso istante inizia il primo contatto. I bundles generati sono inviati subito dal satellite e consegnati al MCC fino al termine del contatto con GS3.
2. Non ci sono contatti con le GS, la generazione dei bundles di telemetria prosegue e questi sono memorizzati nella memoria persistente del satellite.
3. Avviene il contatto con GS1, tutti i bundle di telemetria memorizzati sono inviati al MCC, il trasferimento è rapido poiché tali bundles sono di piccole dimensioni. Una

volta terminato il trasferimento dei bundles in coda, prosegue il trasferimento dei bundle generati fino alla fine del contatto con GS3 (come in 1).

4. la situazione 2-3 si ripete.

L'invio dei bundles dal satellite al MCC è più complicato di quanto sembri poiché, essendo i contatti molto ravvicinati tra loro, il trasferimento inizia sfruttando la GS1 e termina con la GS3. Per decidere a quale GS inviare i bundles è necessario che al Bundle Layer sia presente un algoritmo di instradamento chiamato **Contact Graph Routing**.

Per gli altri due segmenti (Tc in alto, TcACK in mezzo), la situazione è analoga a quella appena descritta: i bundles generati quando il satellite risulta invisibile alle GS, sono memorizzati nella sua memoria locale e poi inviati al primo contatto disponibile.

#### 4.3.5 Summary

		<b>DTN</b>	<b>PEP</b>	<b>e2e TCP (advanced)</b>	<b>E2e TCP (NewReno)</b>
<b>GEO scenarios</b>	GEO fixed terminals	Yes	Yes	Yes with limits	No
	GEO mobile terminals	Yes	Yes	Yes with limits	No
<b>LEO scenarios</b>	LEO Earth observation	Yes	No	No	No
	LEO data mule	Yes	No	No	No

Figure 4.13: The more challenging the scenario, the better for DTN!

# Chapter 5

## Interplanetary Networks

### 5.1 Introduction

Le connessioni interplanetarie costituiscono l'ambiente di sfida più estremo per le DTN. Le sfide alle quali sono sottoposte comprendono: ritardi di propagazione elevatissimi, connessioni intermittenti dovuti al movimento orbitale dei pianeti e alte probabilità di perdita dei dati durante il loro percorso. L'unico vantaggio che si può ricavare da un ambiente del genere è il fatto che le orbite seguono percorsi deterministici, quindi, come per i satelliti LEO, è possibile prevedere le durate dei contatti e i contact volumes.

#### 5.1.1 Moon to Earth Communications

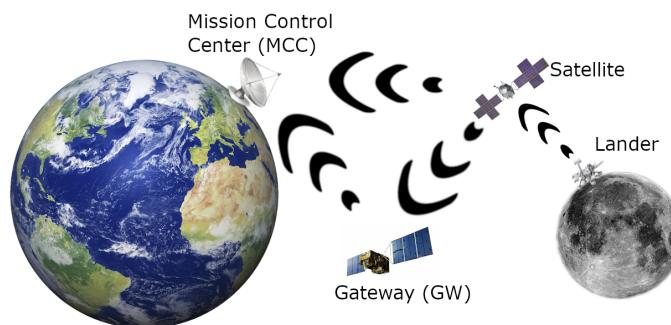


Figure 5.1: Moon to Earth Scenario

Si consideri l'esempio in figura 5.1: si vuole scaricare sulla Terra, al **MCC**, un'immagine (8 bundles) scattata sul lato nascosto della Luna. Sul suolo lunare è situato un **lander** e in orbita (sempre lunare) c'è un **satellite** (simile ai satelliti LEO terrestri) con il quale comunica a intermittenza. Il satellite lunare può inoltre comunicare con un **Gateway** e direttamente con il **MCC** quindi, a seconda delle situazioni in cui si trova, effettua scelte dinamiche per l'invio dei bundles. GW e MCC sono collegati con una connessione stabile e continua: tutti i bundles ricevuti dal GW sono immediatamente consegnati al MCC. I risultati dell'esperimento sono illustrati in figura 5.2

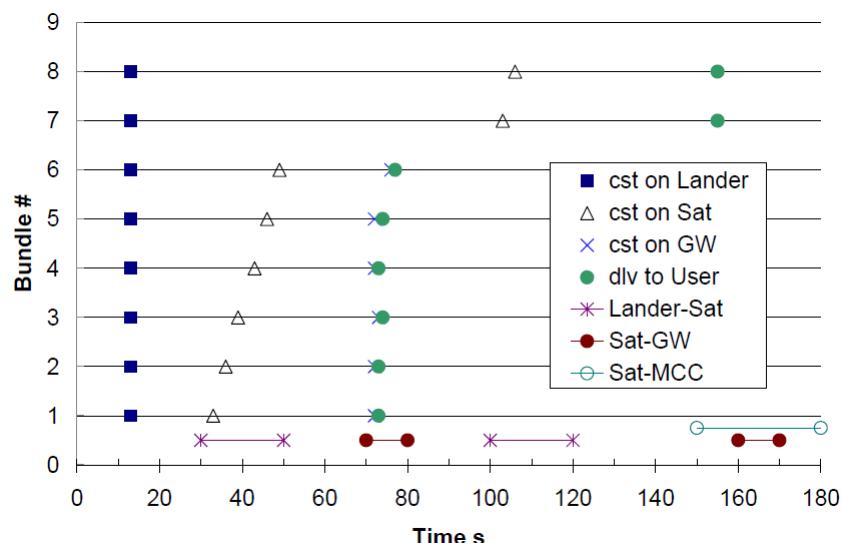


Figure 5.2: Moon to Earth Results

Al tempo  $t=10$  circa, gli 8 bundles sono generati istantaneamente e salvati nella memoria locale del lander (nessun contatto è disponibile). A  $t=30$  si verifica il contatto lander-satellite, la durata dello stesso è lunga abbastanza da permettere il trasferimenti di soli 6 bundles (il satellite ne acquisisce la custodia). A  $t=70$  si verifica il contatto satellite-gateway, i 6 bundles sul satellite sono trasferiti al gateway (che ne acquisisce la custodia) e, istantaneamente, al MCC (i pallini sovrastano le x perché il trasferimento è quasi istantaneo). A  $t=100$  si verifica il contatto lander-satellite, durante il quale sono trasferiti i due bundles rimanenti (il satellite ne acquisisce la custodia). A  $t=150$  si verifica il contatto satellite-MCC e il trasferimento è completato. Si noti che in questo ultimo contatto se ne sovrappone un secondo satellite-gateway in cui non succede nulla. Questo esperimento serve a dimostrare che l'algoritmo Contact Graph Routing effettua sempre la scelta migliore del nodo al quale inviare i bundles (non sempre è conveniente utilizzare il gateway).

### 5.1.2 Mars to Earth Communications

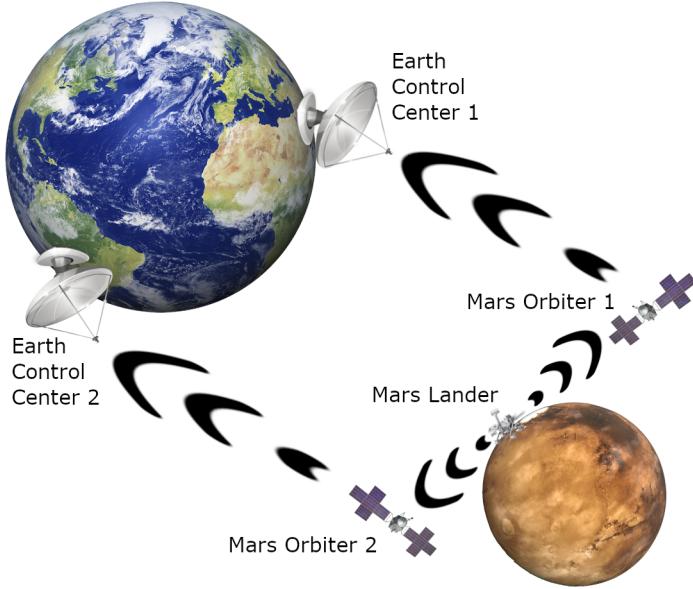


Figure 5.3: Mars to Earth Scenario

Si consideri ora lo scenario in figura 5.3: si vuole inviare una fotografia (10 bundles di 50kB ciascuno) scattata dal lander su Marte, alla Terra utilizzando due Mars Orbiter e due Earth Control Center. Il trasferimento, costituito da due hops, può avvenire utilizzando indipendentemente uno dei due orbiter e uno dei due control center. Su tutti i collegamenti il protocollo di trasporto utilizzato al posto di TCP è **LTP (Licklider Transmission Protocol)**. L'esperimento si concentra sull'analisi delle ritrasmissioni dei segmenti LTP in ambienti con ritardi di propagazione elevati (si assume un RTT earth-orbiter di 720 secondi, cioè 12 minuti).

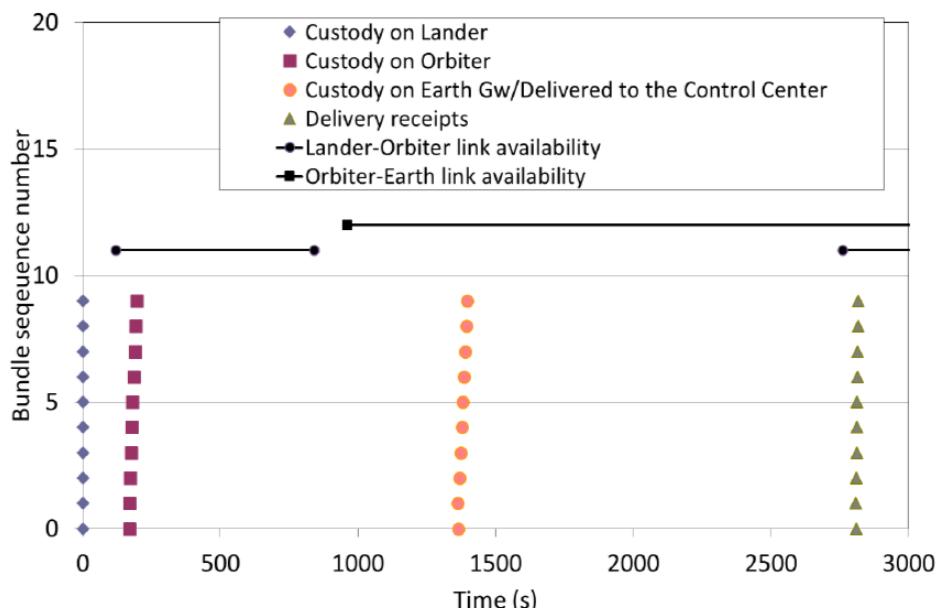


Figure 5.4: Mars to Earth Results (PER = 0%)

Il primo test (figura 5.4) non prevede una perdita di pacchetti durante il tragitto (**PER = 0%**). Al tempo  $t=0$  i bundles sono generati nel lander e, non essendoci orbiter visibili, sono memorizzati all'interno della sua memoria locale. Poco tempo dopo, si verifica il contatto lander-orbiter e tutti i bundles sono trasferiti su esso. A  $t=1000$  si verifica il contatto orbiter-earth e, nello stesso istante, inizia il trasferimento dall'orbiter al control center. Dal grafico si può vedere che i bundles iniziano ad essere recapitati solo a  $t=1360$ , esattamente 6 minuti dopo l'inizio del trasferimento (tempo che coincide con  $RTT/2$ ), a causa dell'elevato tempo di propagazione. Ricevuti i bundles, il control center invia all'orbiter le delivery receipt (non illustrato nel grafico) che a sua volta consegna al lander non appena risulta visibile ( $t=2700$  circa). Da questo esperimento è possibile vedere i vantaggi dell'utilizzo di LTP a scapito di TCP nel caso del canale ideale: i bundles durante il contatto orbiter-earth sono consegnati in  $RTT/2 = 360$  secondi pari al tempo di propagazione del segnale. Non è possibile ottenere un risultato migliore di questo, infatti, se si fosse utilizzato TCP: il 3-way-handshake utilizzato per instaurare la connessione prevede l'invio del SYN e l'attesa del SYN-ACK. Il tempo impiegato per questa operazione è pari a un RTT completo, durante il quale però non è trasferito alcun dato utile (i.e. 12 minuti buttati nel cesso).

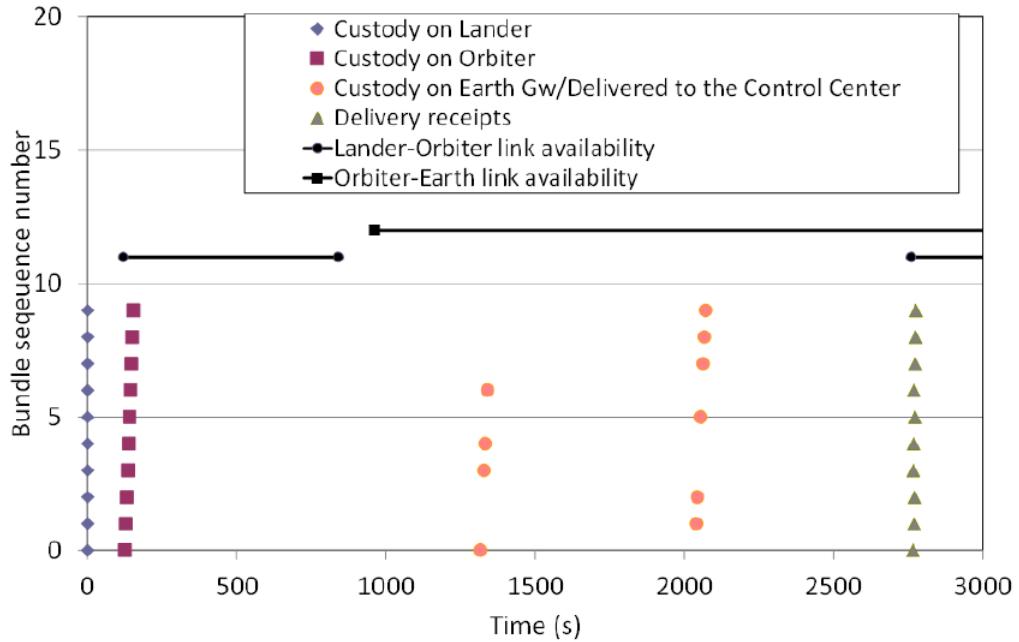


Figure 5.5: Mars to Earth Results (PER = 3%)

Il secondo test (figura 5.5) prevede **PER = 3%**. Utilizzando il protocollo LTP al livello di trasporto, ogni bundle è incapsulato all'interno di un segmento LTP. Tali segmenti LTP sono suddivisi in più pacchetti inviati per mezzo di UDP (si approfondirà LTP nelle sezioni successive, si assume tuttavia che non ci siano perdite di pacchetti che compromettano l'integrità di un segmento intero). La situazione è simile alla precedente ma si verificano delle perdite, per cui, durante il contatto orbiter-earth solo 4 bunnles sono trasferiti correttamente e, i rimanenti 6 sono trasferiti dopo circa un RTT completo. Dopo aver ricevuto i 4 bundles, il control center invia all'orbiter le delivery receipt corrispondenti (tempo necessario per il trasferimento earth-orbiter:  $RTT/2$ ). L'orbiter a questo punto si rende conto che 6 bundle non sono stati consegnati correttamente, quindi, avvia la ritrasmissione (tempo necessario per il trasferimento orbiter-earth:  $RTT/2$ ). Il tempo di circa 12 minuti tra la consegna dei primi 4 bundles e i rimanenti 6 è, in questo scenario, il tempo minimo necessario ottenibile in un protocollo con trasmissione affidabile (quale è LTP).

### 5.1.3 Multi-Asset Mars to Earth Communications

Lo scenario in figura 5.6a rappresenta un caso di missione di esplorazione Marziana. Sono presenti: (1) Un lander Marziano che ha l'obiettivo di inviare alla Terra dati scientifici (immagini) e di telemetria; (2) Due orbiter Marziani; (3) Tre stazioni di Terra; (4) Un satellite GEO chiamato EDRS (European Data Relay Satellite) che funge da gateway; (5) Due nodi di destinazione Terrestri chiamati ESA-MCC (European Space Agency Mission Control Center) ed ESA-PDGS (European Space Agency Payload Data Ground Segment) che hanno rispettivamente i compiti di accettare i dati di telemetria e scientifici.

Il lander su Marte è collegato agli orbiter e alle GS tramite collegamenti intermittenti e con larghezze di banda differenti (linee tratteggiate: banda normale, punteggiate: banda limitata), ma non al gateway EDRS.

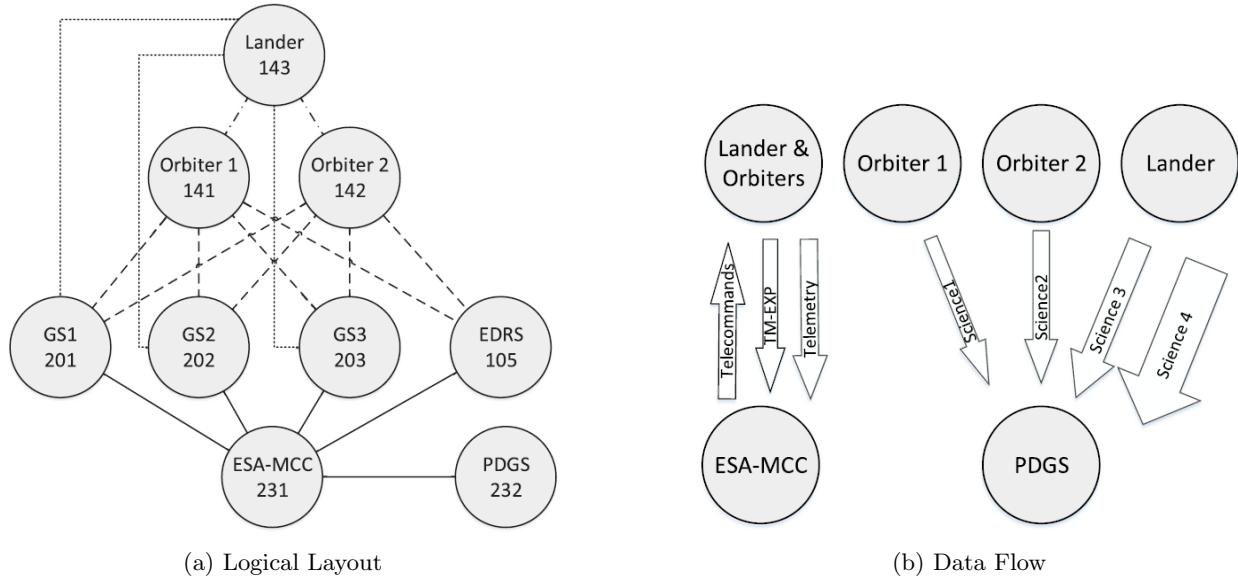


Figure 5.6: Scenario C

Facendo riferimento alla figura 5.6b si può notare come il MCC invii telecomandi al lander e agli orbiter, i quali inviano in direzione opposta dati di telemetria con differenti priorità (expedited e normal). Sono presenti inoltre quattro flussi di dati scientifici con priorità bulk, inviati dagli spacecraft al PDGS. Le istante DTNperf in esecuzione contemporanea sono in totale diciannove: 13 client che generano 13 flussi differenti, 5 server e 1 monitor. Ad ogni esecuzione dell'esperimento sono raccolti 55000 status report.

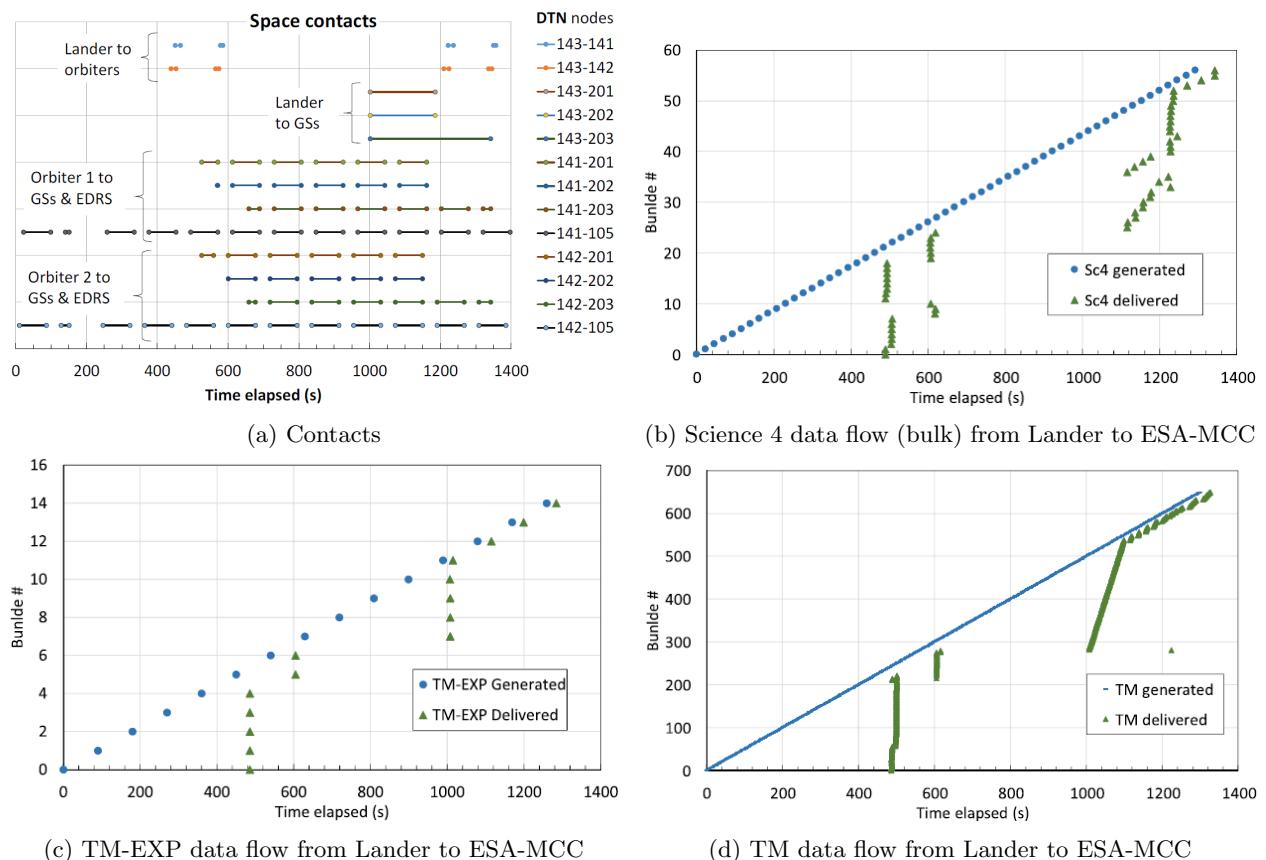


Figure 5.7: Scenario C

In figura 5.7a sono raffigurati i contatti che si verificano tra i vari nodi, i quali mettono in evidenza la complessità del problema di routing. Il data flow bulk in figura 5.7b ha priorità minima, i bundles hanno un rate di generazione più elevato e dimensione maggiore. Le consegne avvengono in maniera disordinata per via della scarsa capacità di banda tra il lander e gli altri nodi. La somma di tutti i volumi di contatto è di poco superiore al traffico generato. In figura 5.7c è illustrato il flow expedited con priorità massima. I bundles hanno rate di generazione breve e sono consegnati in ordine, tuttavia è possibile notare dei burst a causa delle frequenti mancanze di connessione. In figura 5.7d è illustrato il flow normal, nel quale si può notare, ancora una volta, come la consegna dei bundles avvenga in modo disordinato (sintomo di prestazioni degradate). L'esperimento si conclude con la consegna di tutto il traffico previsto nei tempi prefissati (1440s) grazie al fix introdotto in ION 3.0.6b.

## 5.2 Licklider Transmission Protocol (LTP)

### 5.2.1 Introduction and Motivations

Il **Licklider Transmission Protocol (LTP)**, descritto negli RFC 5325 (motivations), 5326 (specifications) e 5327 (security extensions), è un protocollo progettato per garantire affidabilità basata sulle ritrasmissioni (ARQ) su canali caratterizzati da RTT estremamente elevati e/o frequenti interruzioni della connessione. Le comunicazioni spaziali forniscono un chiaro esempio di questo tipo di ambienti e, infatti, LTP è principalmente rivolto a supportare la trasmissione affidabile *long-haul* (a lungo raggio) nello spazio interplanetario.

LTP è usato come Convergence Layer nel Bundle Protocol e si interfaccia sul protocollo UDP o su protocolli analoghi. I punti cardine sono:

- servizio di consegna sia di tipo affidabile che non affidabile;
- riduzione della conversazionalità (**chattiness**) tra entità comunicanti;
- mancanza di una procedura per l'instaurazione di una connessione (handshake);
- mancanza dei classici controlli di flusso e di congestione a retroazione; l'invio dei dati è **rate based**;
- **unidirezionalità logica.** Collegamenti bidirezionali sono possibili tramite instaurazione di sessioni multiple;
- organizzazione dei dati in blocchi (anche di grandi dimensioni).

### 5.2.2 Terminology

- **Engine LTP:** nodo della rete sul quale è implementato il protocollo LTP;
- **Block:** Array di dati applicativi ottenuti dal protocollo del livello superiore (BP) e da trasmettere tramite un'istanza del client LTP. Ciascun blocco è formato da:
  - **Red-Part:** porzione del blocco la cui trasmissione deve avvenire **con affidabilità**, realizzata con ritrasmissioni e acknowledgment (analogamente al TCP).
  - **Green-Part:** porzione del blocco con semantica *best effort*, ovvero consegna **senza affidabilità** e quindi senza meccanismi di ritrasmissione (analogamente all'UDP).

Un singolo blocco può essere costituito da Red Part, Green Part o entrambe. Se presente la Green, è posta subito dopo la Red. La presenza contemporanea di Green e Red Part causa numerosi svantaggi ed è preferibile avere monocromia nel blocco (la possibilità di averle entrambe in un singolo blocco sarà infatti rimossa in futuro).

- **Session:** Attività del protocollo LTP condotta logicamente tra due LTP Engine con lo scopo di trasmettere **un blocco**. Il flusso dati in una sessione è unidirezionale: il dati viaggiano dal mittente al ricevente mentre gli ACK viaggiano in direzione opposta;
- **Segment:** Unità dati della trasmissione LTP utilizzata per la suddivisione di un blocco. È la struttura dati trasmessa da un LTP Engine ad un altro nel corso di una sessione. Ciascun segmento LTP ha un tipo diverso a seconda del contenuto trasportato:
  - **Data Segments:** viaggiano dal mittente al ricevente e trasportano i dati utili per i quali si è avviata la connessione;
  - **Report Segment:** viaggia dal ricevente al mittente e trasporta gli ACK relativi ai soli segmenti ricevuti correttamente;
  - **Report-Acknowledgement:** viaggia dal mittente al ricevente e contiene l'ACK di un Report Segment;
  - **Session Management Segments:** possono essere generati sia dal mittente che dal ricevente e possono essere di due sottotipi:
    - \* **Cancellation:** il mittente inizializza la procedura di cancellazione della sessione al ricevente. All'interno di questo segmento, un byte è riservato al motivo della cancellazione;
    - \* **Cancellation-acknowledgement:** è l'ACK relativo al segmento Cancellation;

Ogni segmento può essere inoltre etichettato come:

- **End Of Block (EOB):** ultimo segmento appartenente al blocco originale;

- **End Of Red-Part (EORP)**: ultimo segmento appartenente alla Red Part del blocco;
- **Check Point (CP)**: segmento che sollecita l'engine ricevente all'invio di un **Reception Report**;
- **Bundle Aggregation**: è il processo di creazione di un blocco LTP. Il CLA del nodo prende i bundles provenienti dal BP layer e li aggrega all'interno di un blocco LTP. L'aggregazione procede finché una tra le due soglie di **aggregation size limit** o **aggregation time limit** è raggiunta.
- **Block Segmentation**: è il processo di suddivisione dei blocchi LTP in segmenti LTP. L'entità del protocollo LTP divide il blocco in un numero di segmenti, la cui dimensione massima dipende dal protocollo del livello sottostante.

### 5.2.3 LTP Session

Una **sessione LTP** è l'attività svolta dal protocollo mediante la quale due Engine si scambiano **un blocco**. Per ogni blocco da trasmettere è inizializzata una nuova sessione. Per il trasferimento di più blocchi contemporaneamente è possibile instaurare **sessioni parallele**. Il numero massimo di sessioni parallele è settato nei file di configurazione del protocollo e agisce come una sorta di controllo del flusso.

L'interazione tra mittente e destinatario avviene senza negoziazione, quindi, alla ricezione del primo segmento del blocco, il destinatario inizia una sessione di ricezione. Per il trasferimento di un blocco sono previsti i seguenti passaggi:

1. Il mittente suddivide il blocco in segmenti e li invia logicamente al destinatario LTP, passando fisicamente per gli strati inferiori (e viceversa una volta arrivati a destinazione).
2. L'ultimo segmento red viene marcato come EORP e CP.
3. Viene avviato un timer per la EORP, quindi può essere ritrasmessa automaticamente se la risposta non è stata ricevuta.
4. L'ultimo segmento del blocco è marcato come EOB.

I segmenti green vengono immediatamente consegnati al servizio del client una volta arrivati al destinatario, mentre per i segmenti red la consegna avviene solo dopo aver ricevuto tutto il blocco. Una volta che il destinatario riceve tutti i segmenti red della trasmissione, viene inviato un Report Segment che indica la ricezione completa. Il Report Segment è immediatamente trasmesso al mittente, e un timer di ritrasmissione viene avviato. Se il “Report Acknowledgment” di conferma non viene ricevuto prima del timeout, il Report Segment viene ritrasmesso. Il mittente riceve il Report Segment, disattiva i timer per la ritrasmissione dell'EORP e invia al destinatario il segmento Report Acknowledgment. In assenza di perdite, la trasmissione della red-part è quindi terminata lato mittente e la sessione viene chiusa. Il destinatario riceve il segmento Report Acknowledgment e disattiva i timer per la ritrasmissione del Report Segment. La sessione di ricezione della red part è terminata e la sessione è chiusa anche lato ricevitore. La chiusura delle sessioni implica la liberazione dei buffer della sessione in entrambi gli engine, che si rendono disponibili ad una eventuale nuova sessione. Come detto in precedenza possono esserci più sessioni in parallelo, a discrezione dell'utente, in relazione anche alla memoria fisica a disposizione per i buffer (spesso limitata in ambito spaziale).

### 5.2.4 Timers calculation

LTP è progettato per sfruttare al meglio i contatti tra nodi, per i quali si conoscono gli instanti di inizio/fine trasmissione e la massima velocità di trasferimento dati ottenibile (*velocità nominale*). Tali informazioni sono note a tutti i nodi della rete DTN grazie al **contact plan** e sono utilizzate dal protocollo LTP per schedulare la trasmissione dei dati. La velocità nominale di un contatto regola la ”velocità” con cui i bundles passano, nello stack di protocolli, dal BP layer all'LTP layer. A differenza di TCP inoltre, LTP **evita il burst** dei segmenti, infatti, tra l'istante di invio di un segmento e l'istante di invio del successivo, deve trascorrere un intervallo di tempo, anch'esso regolato dalla velocità nominale.

Non essendoci negoziazioni, LTP deve calcolare accuratamente i timer di ritrasmissione. Se il tempo calcolato è troppo breve, questo comporta una ritrasmissione non necessaria. Se il tempo calcolato è al contrario eccessivo, il tempo totale di consegna aumenta e, quindi, è

ritardato anche il momento in cui vengono rilasciate le risorse. Nelle comunicazioni spaziali ciò può avere pesanti ripercussioni sul goodput, a causa della mancanza di buffer liberi per accogliere nuovi blocchi. Se una sessione è in corso al momento della chiusura del canale (fine della finestra di trasmissione), non ha senso re-inviare i segmenti di segnalazione non confermati al loro scadere previsto, in quanto il canale radio non sarebbe disponibile. Per questo motivo tutti i timer LTP vengono **congelati** al momento della chiusura di una finestra e riabilitati all'apertura della finestra successiva: i segmenti LTP sono inviati **solo** durante i contatti.

### 5.2.5 Data retransmission

Se durante una sessione l'engine riceve un Checkpoint, esso invia al mittente un **Reception Report**, con il quale indica tutti i segmenti red ricevuti correttamente prima del checkpoint stesso. La perdita di uno o più segmenti red, dato che deve essere garantita la loro consegna, innesca il meccanismo di ritrasmissione. Il Reception Report è normalmente inviato in un singolo Report Segment e al suo interno vi sono i **Reception Claim**, ovvero gli intervalli contigui di byte red ricevuti correttamente.

### 5.2.6 Problems

Questo capitolo ha lo scopo di valutare gli effetti negativi prodotti dalla perdita di segmenti dati e segmenti di segnalazione (report e checkpoint) appartenenti alla red part del blocco. Lo scopo finale è quello di suggerire alcuni possibili miglioramenti in caso di alte percentuali di errore sul canale, mantenendo la compatibilità con gli standard. Poiché non sono soggetti a ritrasmissioni, i segmenti green non sono di interesse per quest'analisi. Si considerino sottintese le fasi per un invio in cui nel nodo mittente i bundle vengono aggregati in blocchi LTP, i quali vengono successivamente segmentati e i segmenti ottenuti vengono messi in coda per l'invio. Con lo scopo di semplificare la lettura, conviene definire le seguenti metriche:

- **Delivery Time**: tempo totale di consegna di un bundle, da intendersi come tempo impiegato dal momento in cui un'applicazione del mittente (posta al livello Applicazione al di sopra del Bundle layer) decide di trasferire un'informazione (ad esempio un file o stringhe di testo) e il momento in cui tale informazione viene consegnata alla corrispondente applicazione del destinatario.
- **Penalty Time**: tempo aggiuntivo al Delivery Time, causato dalla perdita di uno o più segmenti.

#### 5.2.6.1 Ideal transmission

Trasmissione in cui tutti i segmenti vengono ricevuti senza perdite. In questa situazione (figura 5.8), non essendoci perdite e ritrasmissioni, il Penalty Time è nullo.

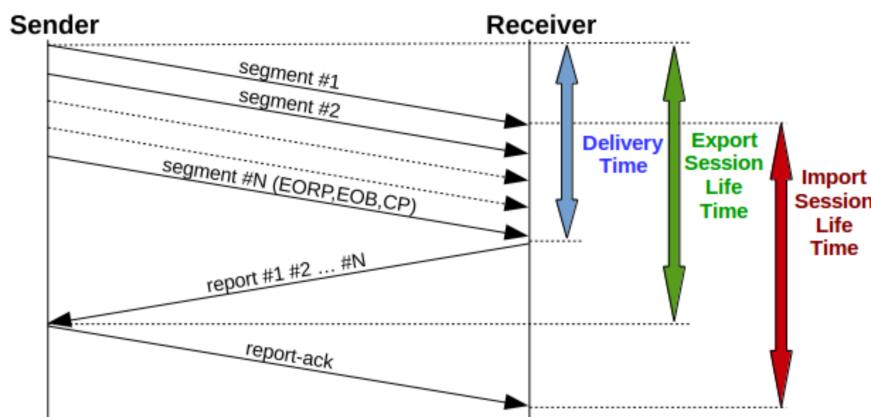


Figure 5.8: ideal session

1. il mittente invia tutti i segmenti (l'ultimo con l'indicazione di End of Red Part, End of Block e Checkpoint);
2. il destinatario riceve correttamente tutti i segmenti e invia un Report Segment che conferma i segmenti ricevuti;
3. il mittente riceve il Report Segment, invia un Report Ack e Termina;

- il destinatario riceve il Report Ack e termina;

Essendo nelle applicazioni spaziali di norma il RTT molto maggiore della somma dei tempi di invio dei segmenti e del ritardo introdotto dal processore, il Delivery Time coincide nel caso ideale con RTT/2, cioè con il tempo di propagazione (minimo teorico assoluto). Lato mittente nel caso ideale si viene a conoscenza della corretta ricezione dopo un RTT, che è ancora una volta il minimo teorico nel caso in cui si voglia l'affidabilità. La durata della sessione è di un RTT, ma lato mittente si apre subito e si chiude all'arrivo del Report Segment. Lato destinatario si apre all'arrivo del primo segmento e si chiude all'arrivo del Report Ack.

#### 5.2.6.2 Losses on data segments only

Trasmissione in cui tre segmenti dati vengono persi (figura 5.9).

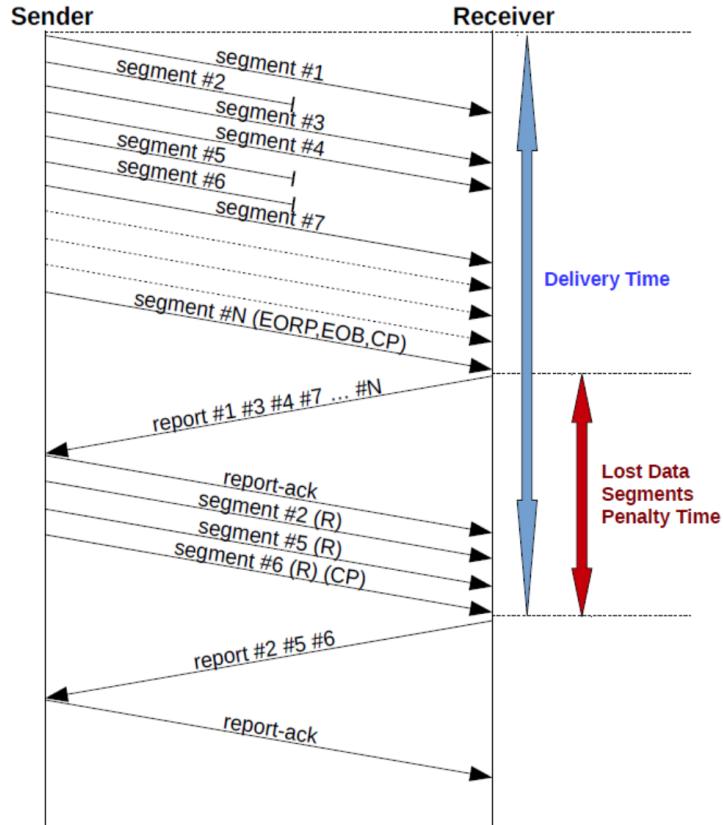


Figure 5.9: sessions with losses on data segments only

- Il blocco LTP è suddiviso in N segmenti che vengono inviati sequenzialmente;
- L'ultimo segmento è etichettato come Check-Point (CP) per richiedere al ricevente il Report Segment (RS);
- Il RS inviato funge da ACK per i segmenti ricevuti (1,3,4,7). I segmenti 2, 5 e 6 sono andati perduti durante il trasferimento, per cui non sono confermati;
- Il mittente invia il Report ACK (RA) relativo al RS;
- Dopo il RA sono ritrasmessi i segmenti precedentemente perduti (2,5,6). Il segmento 6 è etichettato come CP;
- L'RS finale con il relativo RA conclude la sessione;

Assumendo un tempo di ritrasmessione dei segmenti persi  $T_x \ll RTT$ , il tempo perso è pari ad un RTT indipendentemente dal numero di segmenti persi (perché una volta ricevuto il RS, il mittente sa esattamente quali segmenti ritrasmettere).

#### 5.2.6.3 Losses on data and signaling segments

Trasmissione in cui vengono persi segmenti dati e di segnalazione (figura 5.10).

- Il blocco LTP è suddiviso in N segmenti che vengono inviati sequenzialmente;
- I segmenti 2, 3 ed N (quest'ultimo col CP) vengono persi;

3. L'intera sessione rimane in stallo finché il CP non è ritrasmesso (tempo perso pari ad un **RTO** Retransmission Time Out del CP);
4. Solo dopo aver ricevuto il Report, il mittente si rende conto che i segmenti 2 e 3 non sono stati consegnati, per cui li ritrasmette (tempo perso pari ad un RTT, come nel caso del paragrafo precedente);

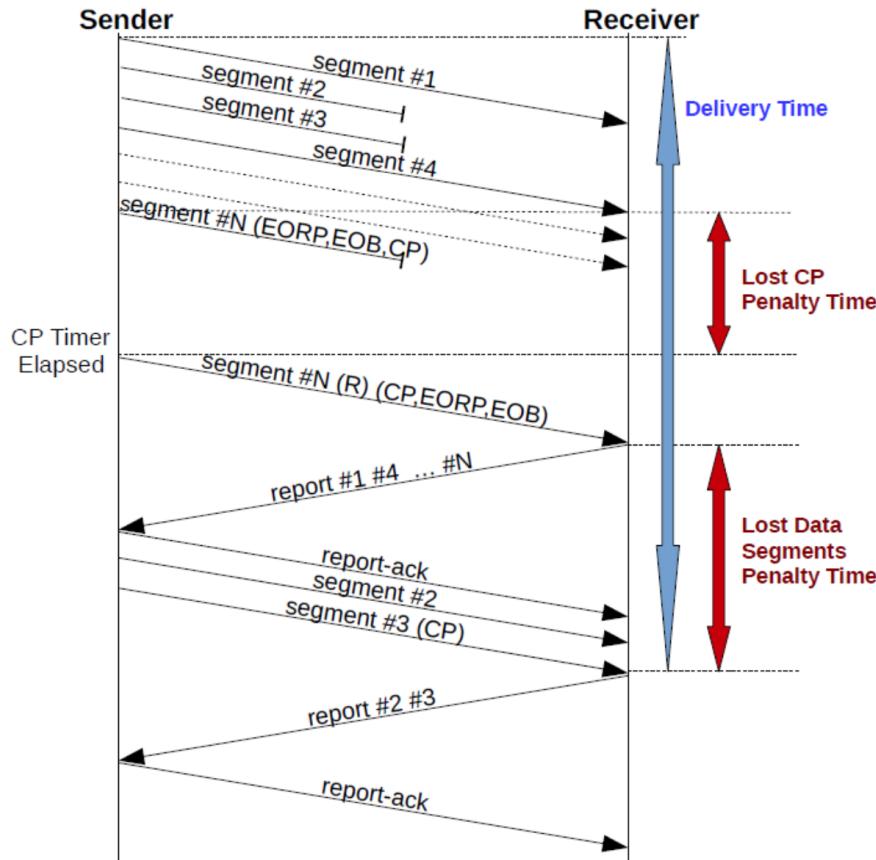


Figure 5.10: session with losses on data and signaling segments

In questo caso, è possibile vedere come il tempo necessario alla ritrasmissione dei segmenti dati è sommato al tempo necessario alla ritrasmissione del CP. Ciò avviene anche nel caso in cui vengano persi durante il trasferimento altri signaling segments.

### 5.2.7 Possible Enhancements

Riassumendo quanto visto nel paragrafo precedente:

- Nel caso ideale i segmenti raggiungono la destinazione in RTT/2 e la sessione completa dura in totale un RTT;
- Per la perdita di uno o più segmenti dati (anche se già ritrasmessi), il Penalty Time è pari ad un RTT;
- Per ogni segmento di segnalazione perso, il Penalty Time è sempre maggiorato di un RTO;

Per ottenere miglioramenti delle performance è logico pensare ad una protezione dei segmenti di segnalazione. Per realizzare tale protezione è possibile utilizzare tecniche di **ridondanza** oppure introdurre un **closing state**.

Gli approcci di ridondanza proposti sono basati sulla ripetizione (N volte) di alcuni segmenti di segnalazione, in particolare dei checkpoint. Le ripetizioni considerate sono di due tipi:

- **Burst** di N1 checkpoint, ovvero N1 invii replicati e consecutivi dei checkpoint;
- **Spread** di N2 checkpoint, ovvero invio di N2 checkpoint equidistanti temporalmente all'interno di un intervallo di RTT;

Le due tecniche possono essere combinate senza interferire tra di loro, risultando in N2 burst di N1 segmenti di segnalazione ripetuti.

Un altro miglioramento possibile è l'introduzione di un nuovo stato, detto closing, in cui la sessione entra prima di essere chiusa definitivamente e durante il quale è in grado di rispondere solo ai Report Segment. Questo evita al destinatario di dover esaurire tutti i tentativi di ritrasmissione prima di poter chiudere la sua sessione.

### 5.2.8 Performances Analysis

I tool usati sono DTNperf\_3, Virtualbricks e macchine virtuali con sistema operativo Debian 7 con installato ION 3.4.1. Il testbed è costituito da due macchine virtuali, connesse attraverso un emulatore di canale (Netemu) nel quale è possibile stabilire i tempi di propagazione one-way e la PER (Packet Error Ratio). Per interpretare al meglio i risultati, conviene cercare di evitare interferenze da sessioni LTP parallele. Per questo motivo è preferibile inviare un bundle per volta. Il principio di funzionamento di DTNperf\_3 può essere riassunto nelle seguenti 4 fasi:

1. Il client, in esecuzione sul nodo mittente, genera un bundle e lo trasmette al server;
2. il bundle viene consegnato al server, in esecuzione sul nodo di destinazione;
3. viene inviato un ACK al client;
4. il client riceve l'ACK e ricomincia dal punto 1;

Il ciclo si interrompe se è stata inviata la quantità di dati richiesta inizialmente, oppure, si è esaurito il tempo massimo dato come parametro di input.

Il Delivery Time può essere calcolato dagli Status Report “delivered”, che contengono il timestamp di generazione del bundle e il timestamp di consegna del bundle. Questa operazione è molto semplice ma richiede che le macchine in cui sono in esecuzione il client e il server di DTNperf\_3 abbiano gli orologi di sistema perfettamente allineati. Nei test sono stati considerati 4 scenari LTP:

- **Originale**, così come fornito da ION 3.4.1;
- **Closing State**;
- Burst (con lunghezza dei burst=3 per tutti i segmenti di segnalazione) più Closing State (**Burst 3** in figura);
- Spread (con NRIP=3) più Closing State (**Distributed 3** in figura);

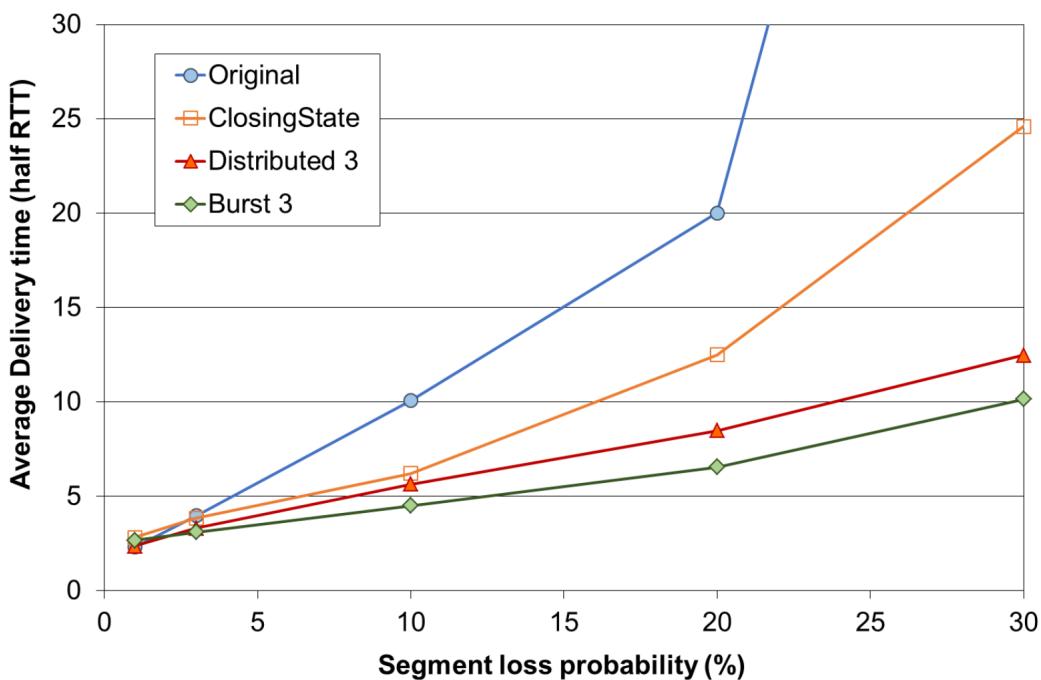


Figure 5.11: Risultati dei test in termini di Delivery Time medio in funzione della PER (Packet Error Ratio)

I risultati in figura 5.11 confermano che, se il canale presenta delle perdite indipendenti, il burst risulta la più efficace delle tecniche implementate. Il vantaggio introdotto dalle nuove tecniche non è significativo (in termini di media) per valori di PER inferiori al 10%, perché le problematiche da risolvere riguardano solo le situazioni in cui si hanno delle perdite, come la perdita del Report Ack finale (nel caso del Closing State) o anche la perdita di altri segmenti di segnalazione (nel caso di Burst e Spread). Ad ogni modo si può notare che i miglioramenti introdotti hanno un impatto significativo a partire da valori di PER minori (già a partire dal 3%).

### 5.2.9 Packet Layer Forward Error Correcting (PL-FEC)

Con i miglioramenti introdotti nei capitoli precedenti, l'ARQ del protocollo LTP è spinto ai limiti teorici massimi raggiungibili. In situazioni in cui però il ritardo di propagazione è elevato (tipo i 3-20 minuti Terra-Marte) è bene considerare delle alternative. Una tra le possibili alternative è l'utilizzo del **Packet Layer Forward Error Correcting (PL-FEC)**. Per comprendere questo meccanismo si pensi ai codici di errore utilizzati nel Physical Layer delle trasmissioni terrestri. Tali codici sono bit di ridondanza aggiunti ai bit del messaggio originale, grazie ai quali è possibile risalire all'informazione corretta e verificare che non si siano verificati errori dovuti (principalmente) al rumore. PL-FEC funziona in modo simile trattando i segmenti LTP come se fossero i bit del messaggio.

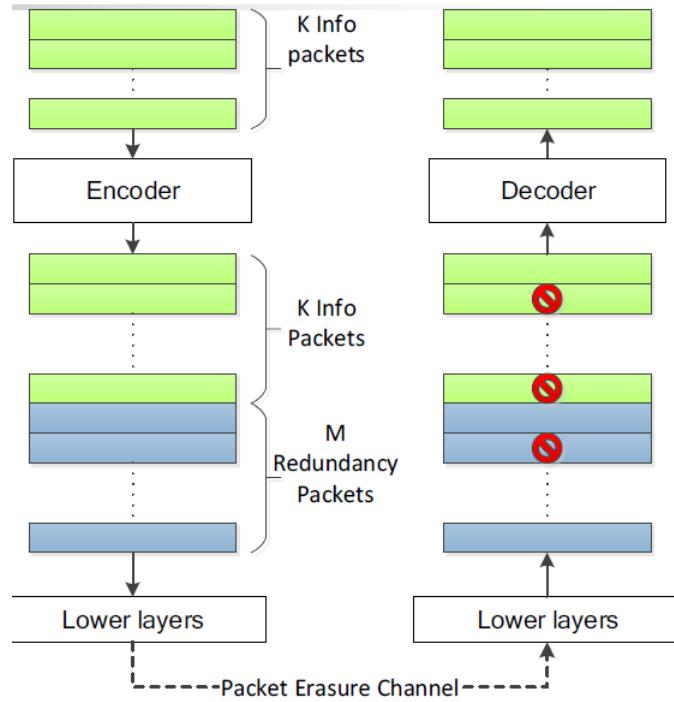


Figure 5.12: An alternative to ARQ: PL-FEC

Facendo riferimento alla figura 5.12:

1.  $K$  segmenti LTP (o **pacchetti info**) contenenti l'informazione originale sono passati ad un encoder;
2.  $M$  pacchetti di ridondanza sono aggiunti dall'encoder per andare a formare la **codeword** completa;
3. i  $K + M$  pacchetti sono inviati e durante il transito alcuni di questi (appartenenti sia a  $K$  che a  $M$ ) sono perduti;
4. Sia  $P$  il numero totale di pacchetti persi, se vale  $P \leq M$  allora è possibile ricostruire correttamente i  $K$  pacchetti info grazie al decoder;

Utilizzando PL-FEC, il meccanismo di ritrasmissione è innescato solo nel momento in cui non è possibile decodificare correttamente una codeword. Di contro, aumenta la complessità generale e il consumo di banda. Carlo Caini e Tommaso De Cola hanno messo in dubbio l'idea che ARQ sia la sola possibile soluzione da utilizzare in LTP, giungendo alle seguenti conclusioni:

- ARQ lavora *a posteriori*: reagisce alle perdite inviando "spare parts" (pezzi di ricambio) solo quando necessario. Ciò risulta buono se il RTT è corto e la larghezza di banda scarsa;
- PL-FEC lavora *a priori*: anticipa le possibili perdite che si potrebbero verificare introducendo le "spare parts". Ciò risulta buono se il RTT è lungo e la larghezza di banda abbondante;
- Sebbene LTP standard è adatto ai satelliti e ai collegamenti Terra-Luna, potrebbe non esserlo ai collegamenti interplanetari;

Si è liberi di essere in accordo o in disaccordo con tale linea di pensiero: se si dovesse esplorare una savana con un'automobile, converrebbe portare i pezzi di ricambio in anticipo, o richiederli e aspettarli solo al momento del bisogno?

## 5.3 Contact Graph Routing (CGR)

### 5.3.1 Introduction

Il routing in Internet può essere considerato analogo alla pianificazione di un viaggio su strada. I collegamenti (strade) formano gli archi del grafo; gli host e i routers (città e incroci) formano i vertici; i costi sono associati all’attraversamento di ciascuno degli archi e il problema è trovare la strada a costo minimo da un vertice ad un altro nel grafo. Per l’Internet terrestre, questo modello funziona bene perché entrambe le topologie di strade e rete sono *time insensitive*. Le connessioni tra host e router sono generalmente continue e di capacità teoricamente illimitata, almeno per la durata di ogni singolo attraversamento del grafo, così come è molto probabile che le autostrade siano aperte a tutti i potenziali conducenti per tutta la durata di ogni singolo viaggio. Ma in una rete spaziale basata su DTN, le connessioni tra i nodi possono apparire e scomparire in orari programmati e potrebbe non esserci mai una connettività continua dalla sorgente di un bundle, fino alla sua destinazione in qualsiasi momento.

Il Contact Graph Routing è invece simile alla prenotazione di voli aerei. Un volo di una singola compagnia aerea costituisce il transito da un aeroporto a un altro ed è caratterizzato dall’ora di partenza, dall’ora di arrivo e dal numero di passeggeri che l’aereo può trasportare. Il problema è selezionare, per ogni viaggiatore, una sequenza di voli che anticipi il più possibile l’orario di arrivo a destinazione, indipendentemente dagli aeroporti sulla rotta. Gli aeroporti vincolano la selezione dei voli (un viaggiatore non può atterrare a Nashville e poi decollare da Francoforte) ma non costituiscono i vertici del grafo. I voli sono i vertici e le connessioni tra questi sono gli archi, cioè i periodi di tempo durante i quali un viaggiatore in arrivo su un volo deve attendere prima di partire con il successivo.

Allo stesso modo, un contatto in una rete DTN costituisce la trasmissione da un nodo a un altro ed è caratterizzato da un inizio, una fine e una capacità. Il problema è selezionare, per ogni bundle, una sequenza di contatti che anticipi il più possibile il tempo finale di arrivo a destinazione, indipendentemente da quali nodi si trovano sul percorso. I nodi vincolano la selezione dei contatti (un bundle non può essere ricevuto da un nodo A e successivamente trasmesso dal nodo B) ma non costituiscono i vertici del grafo. **I contatti sono i vertici del grafo e gli archi sono i periodi di tempo in cui un bundle risiede nella memoria di un nodo**, in attesa della successiva opportunità di trasmissione.

### 5.3.2 Fundamentals

Il **Contact Graph Routing (CGR)** è un sistema di routing dinamico che calcola percorsi attraverso una topologia variabile nel tempo di contatti schedulati in una rete basata sull’architettura DTN. Può essere applicato con successo non solo nell’Internet interplanetario, ma anche a comunicazioni satellitari LEO, poiché in entrambi i casi la disponibilità dei collegamenti è nota a priori. Tuttavia, questa conoscenza perfetta non riduce la complessità dei calcoli del percorso, poiché CGR deve considerare che i collegamenti tra i nodi della rete cambiano nel tempo. La strategia di base di CGR è quella di sfruttare il fatto che, poiché le operazioni di comunicazione spaziale sono pianificate in dettaglio per ogni missione, le rotte tra una qualsiasi coppia di nodi, possono essere dedotte da quei piani piuttosto che essere scoperte attraverso un dialogo attivo.

Alla base del CGR c’è il **contact plan** (piano dei contatti), un elenco ordinato temporalmente delle modifiche pianificate che si verificano nella topologia della rete. Le voci in questo elenco sono i *contatti*; ciascuno di essi indica che la trasmissione dal nodo X al nodo Y alla velocità di dati nominale  $R$  inizierà all’istante  $T_1$  e terminerà all’istante  $T_2$ . Questa affermazione definisce implicitamente anche il **contact volume**, che è la quantità massima di dati che possono essere trasferiti durante il contatto, data dal prodotto tra lunghezza del contatto e la velocità di trasmissione nominale:  $R(T_2 - T_1)$ .

Ogni nodo utilizza i contatti nel contact plan per creare una sorta di *tavella di routing* ovvero, una ”*lista di liste di rotte*”. Ad ogni possibile nodo di destinazione nella tabella è associata una lista di rotte. Ogni rotta nell’elenco per il nodo D identifica un percorso al nodo di destinazione D, dal nodo locale, che inizia con la trasmissione a uno dei vicini del nodo locale nella rete. Il primo nodo a ricevere dati dal nodo locale è chiamato **entry node** della rotta. La voce nell’elenco delle rotte per ogni vicino contiene il percorso migliore che inizia con la trasmissione a quel vicino. Ad ogni rotta sono inoltre associati: tutti gli altri contatti che costituiscono i segmenti del percorso end-to-end, il costo stimato e il tempo di *forfait*, cioè, l’ultimo istante entro il quale il bundle deve essere stato inoltrato all’entry node

della rotta per avere qualche possibilità di attraversarla. Per calcolare una nuova lista di rotte per il nodo D:

- Si costruisce un grafo astratto, orientato e aciclico dei contatti in cui: la radice è il contatto del nodo locale con se stesso, il vertice terminale è il contatto del nodo D con se stesso e gli altri vertici sono contatti che contribuiscono a formare percorsi end-to-end dal nodo locale a D.
- Si effettuano una serie di ricerche sul grafo utilizzando l'algoritmo di Dijkstra. Ad ogni ricerca si trova la rotta a costo minimo che parte dal nodo locale e termina al nodo D. Ogni volta che viene calcolato un percorso, questo è aggiunto all'elenco delle rotte del nodo. Il contatto iniziale del percorso è poi rimosso dal grafo prima di procedere con la ricerca del percorso successivo. La serie di ricerche termina quando non viene prodotto nessun nuovo percorso.

Tra le rotte utilizzabili, scegliamo quella con il costo più basso e mettiamo in coda il bundle per la trasmissione all'entry node di quella rotta. Se l'elenco dei bundle in coda per la trasmissione su un percorso non è vuoto al momento in cui viene raggiunto il tempo di forfait del percorso, è necessario calcolare nuovi percorsi per tutti quei bundle.

Quindi, quando un bundle deve essere trasmesso dal nodo A al nodo Q, consultiamo l'elenco delle rotte per il nodo Q ma, alcune di queste potrebbero essere inutilizzabili perché:

- una rotta potrebbe essere temporaneamente non disponibile perché la trasmissione all'entry node è *bloccata* a causa della perdita della connessione;
- il tempo di consegna nel migliore dei casi su un percorso potrebbe essere maggiore del tempo di vita del pacchetto e, di conseguenza, il pacchetto verrebbe eliminato prima della consegna;
- la "capacità residua" del contatto iniziale sulla rotta (la capacità che non è stata ancora allocata ai bundle con priorità più alta) potrebbe non essere sufficiente a contenere il bundle;

Si noti che quest'ultimo controllo è una sorta di congestion control embrionale: una rotta è considerata inutilizzabile se il suo primo contatto è già completamente "prenotato", causando il re-indirizzamento del pacchetto su rotte meno congestionate.

Il vantaggio principale del CGR è che, come il routing Internet, può essere eseguito con grande sicurezza, poiché si basa su informazioni accurate sulla topologia della rete. La differenza è che:

- La topologia su cui si basa il routing non è la topologia corrente attualmente conosciuta, ma piuttosto una topologia anticipata variabile nel tempo.
- Poiché i cambiamenti nella topologia della rete sono programmati nel corso della pianificazione delle missioni, le informazioni su tali modifiche possono essere propagate molto prima che si verifichino. Proprio come in Internet, la comprensione da parte di ogni nodo della topologia della rete in qualsiasi momento è quasi sempre corretta: mentre la propagazione delle informazioni sui cambiamenti della topologia della rete è lenta, ma comunque "più veloce" della velocità con cui si verificano i cambiamenti stessi.

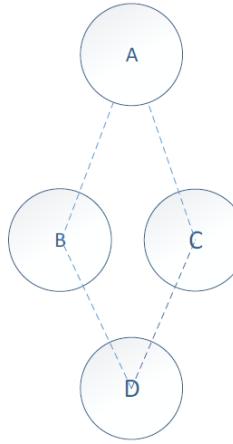
### 5.3.3 Example

Si consideri la topologia della rete presente in figura 5.13a.

Se A è il nodo sorgente e D il nodo destinazione, ci sono solo due possibili strade "geografiche" per congiungerli: attraverso B oppure attraverso C. Tuttavia, ogni collegamento tra i nodi è intermittente, per cui il problema non è trovare la migliore strada "geografica", quanto piuttosto trovare la miglior serie di contatti da poter sfruttare.

A partire dal contact plan in figura 5.13b è possibile costruire il grafo in figura 5.14 nel quale, per semplicità, sono illustrati solo i (6 su 12) contatti utili a far viaggiare i bundles da *alpha* a *beta* in modo unidirezionale (in direzione opposta non interessa ai fini della spiegazione). Si supponga di voler inviare un bundle generato al tempo  $t = 0$  e si assuma che il trasferimento tra nodi avvenga in maniera istantanea (no traffico di background), quale tra le opzioni disponibili sarebbe la migliore? Per trovare la risposta si devono valutare tutte le opzioni possibili:

- (1-3) 1000s per passare da A a B, 100s per passare da B a D, totale 1100s;



(a) Topology

Contact	Sender	Receiver	From time (s)	Until time (s)	Rate (kbps)
1	A	B	1000	1100	1000
2	B	A	1000	1100	1000
3	B	D	1100	1200	1000
4	D	B	1100	1200	1000
5	A	C	1100	1200	1000
6	C	A	1100	1200	1000
7	A	B	1300	1400	1000
8	B	A	1300	1400	1000
9	B	D	1400	1500	1000
10	D	B	1400	1500	1000
11	C	D	1500	1600	1000
12	D	D	1500	1600	1000

(b) Contact scheme

Figure 5.13: Network topology example

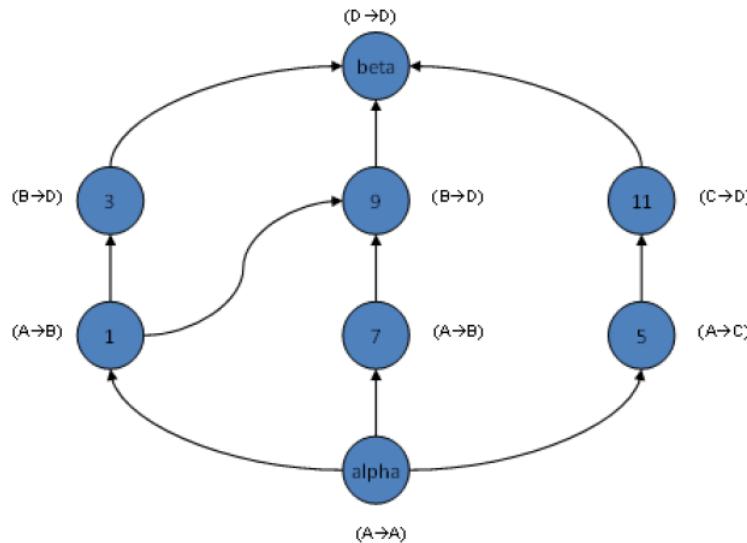


Figure 5.14: Solution

- (1-9) 1000s per passare da A a B, 400s per passare da B a D, totale 1400s;
- (7-9) 1300s per passare da A a B, 100s per passare da B a D, totale 1400s;
- (5-11) 1100s per passare da A a C, 400s per passare da C a D, totale 1500s.

L'algoritmo di Dijkstra è di tipo *greedy* e sceglie come nodo successivo al nodo corrente, quello collegato con l'arco di costo minimo, senza tenere conto dei nodi già visitati. La prima soluzione prodotta dall'algoritmo è il percorso (1-3) che risulta effettivamente il migliore.

### 5.3.4 Schedule-Aware Bundle Routing (SABR)

Lo **Schedule-Aware Bundle Routing (SABR)** è l'ultima versione di CGR standardizzata dal CCSDS (Consultative Committee for Space Data Systems) e completamente implementata in ION (dalla versione 3.7.0). Il funzionamento di SABR è riassunto in tre fasi distinte e schematizzato in figura 5.15:

1. **Route Computations:** in questa fase (**bundle independent**), a partire dalle informazioni contenute nel contact plan, viene costruito il grafo e lanciato su di esso l'algoritmo di Dijkstra per selezionare tutte le rotte a costo minimo dal nodo sorgente, al nodo destinazione. L'esecuzione dell'algoritmo è un'operazione piuttosto pesante, per cui, **non è ripetuta per tutti i bundle**;
2. **Route Validation:** in questa fase (**bundle dependent**) sono utilizzate sia informazioni contenute nel contact plan, sia informazioni locali al nodo. La validazione consiste nel selezionare **per ogni bundle**, tra tutte le rotte ricavate dalla fase precedente, quelle in grado di portarlo alla sua destinazione prima della scadenza. Le informazioni locali comprendono le code di output dell'host e informazioni relative ai singoli bundle, in particolare, dimensione, lifetime e priorità;

3. **Route Selection:** in questa fase, **per ogni bundle**, è selezionata la rota migliore tra quelle elencate nella fase precedente.

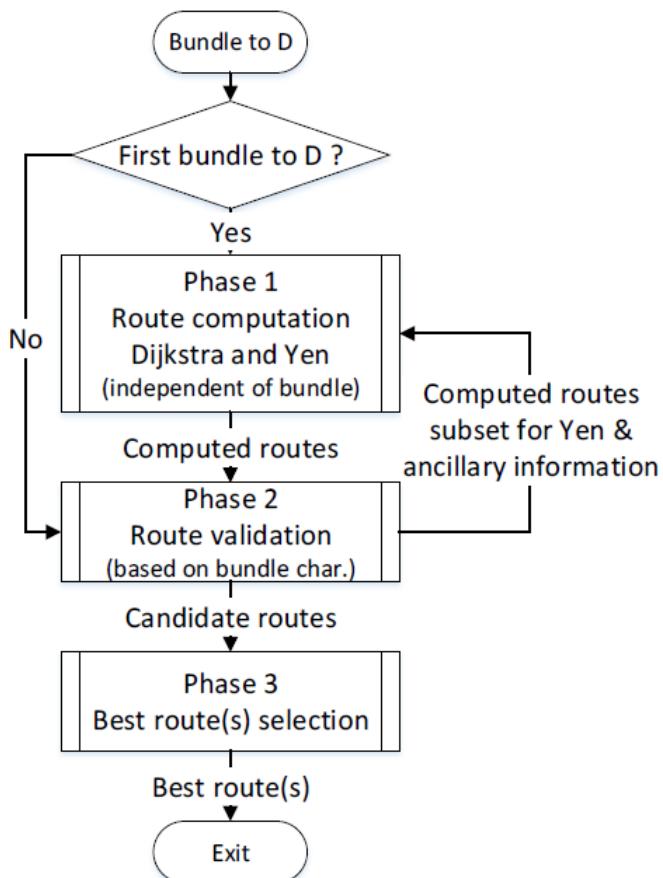


Figure 5.15: SABR

SABR è un algoritmo di routing **non ottimo** e **non scalabile** (la complessità della fase 1 aumenta esponenzialmente col numero di contatti nel contact plan) che fornisce il giusto *trade-off* tra complessità e accuratezza. Per motivi di sicurezza inoltre, **la rotta completa è ricalcolata da zero in ogni nodo** ed è quindi usata solo per selezionare il nodo dell'hop successivo.

### 5.3.5 Unibo-CGR

**L’Unibo-CGR** è una implementazione sperimentale del CGR sviluppata all’Università di Bologna. Si contraddistingue per essere *research-friendly* poiché fornisce l’implementazione SABR di CGR come da specifiche, arricchendola di alcuni miglioramenti opzionali. L’implementazione consiste in un core completamente indipendente dal Bundle Protocol che supporta interfacce multiple (ION bpv6, bpv7, DTN2/DTNME).

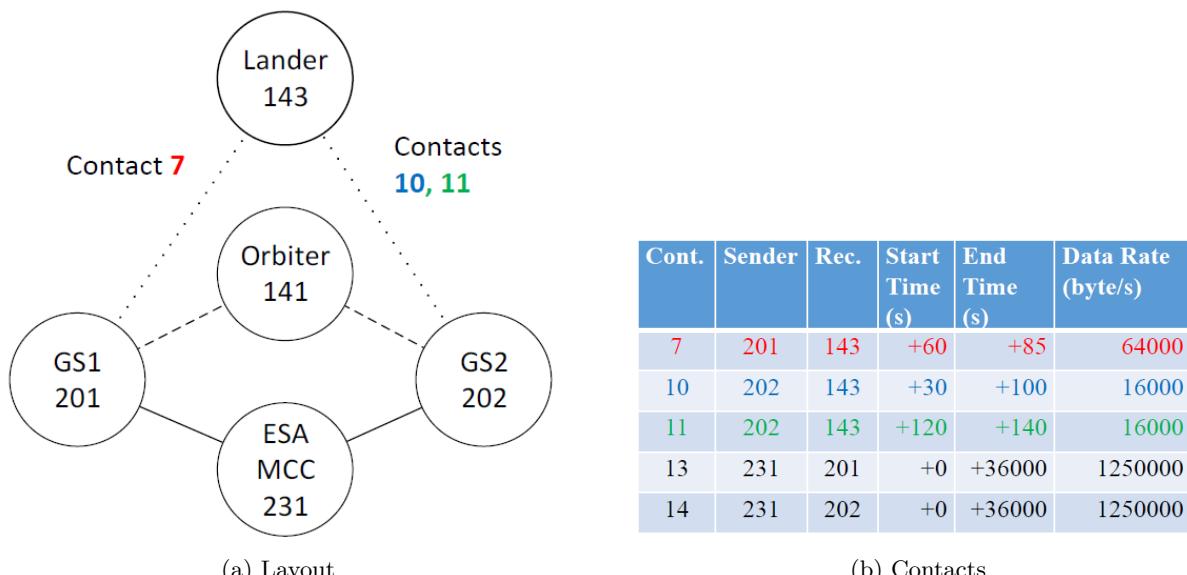


Figure 5.16: Simplified Scenario C

Per dimostrare l'utilità di due miglioramenti (One-route-per-neighbor e queue-delay) presenti in Unibo-CGR è stato effettuato un test in un ambiente completamente ION e uno in cui invece l'ESA MCC eseguiva DTN2 anziché ION (DTN2 non supporta i collegamenti intermittenti) su una rete con la topologia in figura 5.16a. Lo scenario è una versione semplificata dello scenario C già incontrato in precedenza. I contatti continui sono rappresentati dalle linee continue (gli archi tra MCC e GS sono rispettivamente i contatti 13 e 14), quelli intermittenti da linee punteggiate e quelli degli orbiter da linee tratteggiate (non presenti nel contact plan in quanto non essenziali). Tutti i contatti sono bidirezionali, ma la particolarità sta nel fatto che il contatto 7 è nidificato all'interno del contatto 10, il che rappresenta un ambiente di sfida particolarmente difficile poiché porta alla creazione di loops. Se i dati arrivano alla GS1, la cosa ovvia da fare sarebbe utilizzare il contatto 7 per mandarli al lander. Tuttavia il contatto 7 apre dopo 60s mentre il contatto 10 apre dopo 30s. GS1 potrebbe pensare che sia meglio, secondo le sue informazioni locali, inviare i bundles sfruttando il contatto 10, poiché arriverebbero prima. GS1 quindi invia i bundles all'orbiter, che li recapita poi a GS2. GS2, una volta ricevuti i bundles, potrebbe pensare secondo le sue informazioni locali che sia meglio inviare i bundles sfruttando il contatto 7. Poiché è previsto un meccanismo anti ping-pong che le impedisce di inviare i bundles nella direzione in cui gli sono arrivati (quindi all'orbiter), GS2 manda i bundles a GS1 tramite l'MCC, e il loop si ripete. Il tutto avviene perché GS1 e GS2 potrebbero avere opinioni diverse su chi si trovi nella posizione migliore per inviare i bundles al lander. Durante il test sono stati inviati 20 bundles dal MCC al Lander.

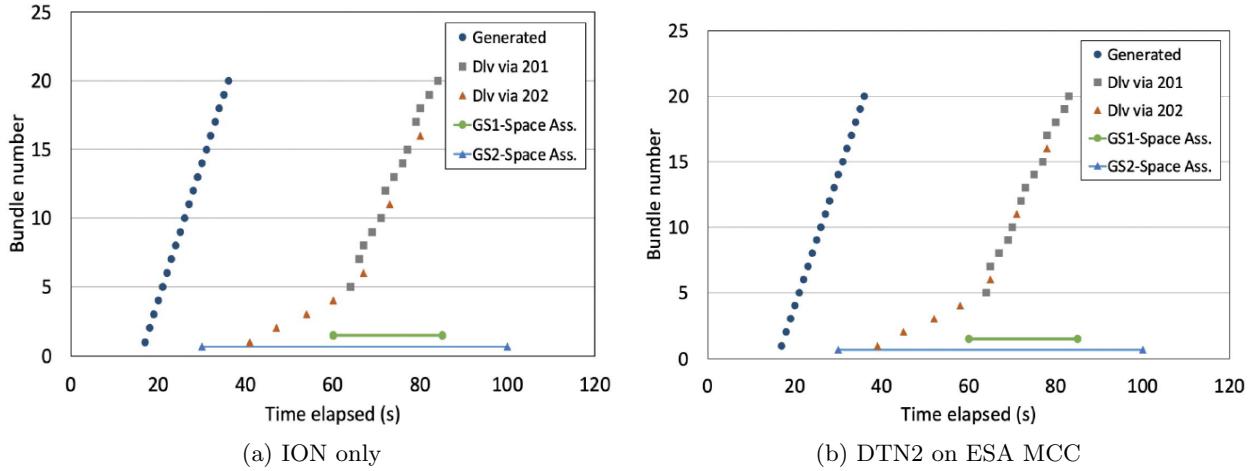


Figure 5.17: Simplified Scenario C results

Dai risultati in figura 5.17a si può notare come CGR sia in grado di consegnare correttamente i bundles dal MCC al lander utilizzando entrambe le GS in modo concorrente. Quando i contatti sono nidificati si può notare un'alternanza di consegna che dipende dalla banda associata al contatto: per ogni bundle consegnato da GS2, ne vengono consegnati 4 da GS1 perché la banda di GS1 è 4 volte la banda di GS2. In figura 5.17b sono mostrati i risultati ottenuti utilizzando ION e DTN2, questi sono praticamente identici al caso precedente e ciò dimostra l'efficacia di Unibo-CGR a prevenire l'effetto di ping pong che si potrebbe verificare combinando le due implementazioni nella stessa rete (utilizzando solo ION o solo DTN2 sui nodi, non si verifica ping pong).

# Chapter 6

## Security & QoS

### 6.1 DTN Security

Il tema della sicurezza nelle reti DTN è trattato all'interno dell'RFC 6257 (2011), che descrive il Bundle Security Protocol (BSP). Una versione semplificata chiamata BPSec è in corso di stesura da parte dell'IETF DTN Working Group (DTNWG), è destinata a diventare un RFC in futuro<sup>1</sup> ed è quello che è trattato nel seguito.

BSP definisce alcune estensioni di sicurezza per il Bundle Protocol per andare a supportare: hop-by-hop authentication, end-to-end integrity and authentication, end-to-end confidentiality e il trattamento differente per bundle payload e altri bundle dati. Per soddisfare i requisiti di sicurezza, all'interno della struttura dei bundle sono inseriti dei **Security Block**. Tuttavia, molti sono i problemi che affliggono tali estensioni, dalla eccessiva complessità (la regola Keep It Simple Stupid KISS è stata largamente violata) alla scarsa compatibilità con la bundle fragmentation e il routing. Per quest'ultima in particolare, le destinazioni di sicurezza diverse dalla destinazione rendono impossibile instradare i pacchetti sulla base dell'EID di destinazione.

#### 6.1.1 Terminology

- **Security Block:** estensione di sicurezza presente in un bundle.
- **Security-aware Nodes:** nodi DTN in grado di processare Security Blocks. Altri nodi sono "trasparenti" per quanto riguarda la sicurezza.
- **Waypoints:** qualsiasi nodo DTN intermedio diverso dalla destinazione.
- **Security source:** nodo che aggiunge un Security Block ad un bundle. Solitamente coincide con la sorgente del bundle, ma non necessariamente.
- **Security destination:** nodo che processa il Security Block di un bundle. In BPSec è riportato che tale destinazione deve coincidere con la destinazione del bundle (che è univoca) per risolvere alcuni problemi di routing derivanti dall'utilizzo di BSP. Tuttavia, anche i nodi intermedi possono essere in grado di processare Security Blocks.
- **Security Operation:** l'applicazione di un dato servizio di sicurezza a un obiettivo di sicurezza. Un'operazione di sicurezza è implementata da un Security Block.
- **Security Service:** processo che fornisce una certa protezione ad un target di sicurezza.
- **Security Target:** il blocco all'interno di un bundle che riceve un servizio di sicurezza come parte di un'operazione di sicurezza.

#### 6.1.2 Block-Level Granularity

I servizi di sicurezza di BPSec devono consentire, a diversi blocchi all'interno di un bundle, di applicare diversi servizi di sicurezza. I blocchi all'interno di un bundle rappresentano diversi tipi di informazioni. Il primary block contiene informazioni di identificazione e routing. Il payload trasporta i dati applicativi. Gli extension blocks trasportano una varietà di dati che possono aumentare o annotare il carico utile o altrimenti fornire le informazioni necessarie per la corretta elaborazione di un bundle lungo un percorso. Pertanto, l'applicazione di un unico livello e tipo di sicurezza a un intero pacchetto non riesce a riconoscere che i blocchi in un pacchetto rappresentano diversi tipi di informazioni con diverse esigenze di sicurezza. Ad

---

<sup>1</sup>Ad oggi sono tre i draft in corso e che diventeranno RFC: BPbis, BPsec e TCPCLv4 (TCP Convergence Layer Protocol)

esempio, un blocco del payload potrebbe essere crittografato per proteggerne il contenuto e un blocco di estensione contenente informazioni di riepilogo relative al payload potrebbe essere firmato per l'integrità ma non crittografato per fornire ai waypoint l'accesso ai dati relativi al payload senza fornire l'accesso al payload.

### 6.1.3 Security Blocks

BPbis definisce due tipi di security block:

- **Block Integrity Block (BIB, ex PIB):** Il BIB è utilizzato per garantire l'integrità dei suoi target di sicurezza (in chiaro). Le informazioni di integrità nel BIB possono essere verificate da qualsiasi nodo lungo il percorso del bundle dalla security source alla destinazione del bundle. I waypoint aggiungono o rimuovono BIB dai bundles in conformità con la loro politica di sicurezza. I BIB non vengono mai utilizzati per la protezione dell'integrità del testo cifrato fornito da un BCB. Poiché la politica di sicurezza sui nodi BPsec può differire per quanto riguarda la verifica dell'integrità, i BIB non garantiscono l'autenticazione hop by hop.
- **Block Confidentiality Block (BCB, ex PCB):** Il BCB indica che i target di sicurezza sono stati crittografati alla security source per proteggere il loro contenuto durante il transito. Il BCB viene de-crittografato dai nodi di accettazione della sicurezza nella rete, fino alla destinazione del bundle inclusa, come una questione di politica di sicurezza. I BCB forniscono inoltre meccanismi di protezione dell'integrità per il testo cifrato che generano.

## 6.2 Quality of Service (QoS)

### 6.2.1 Priority classes

L'RFC 5050, come già visto, definisce tre classi di priorità: bulk, normal ed expedited. Tali classi implicano una qualche forma di schedulazione basata sulla priorità all'interno dei router DTN, ma solo parzialmente sono supportati da DTN2 (flag di priorità possono essere settati, ma i router DTN non se ne preoccupano). La NASA ha ribattezzato le tre classi come **ordinali** e ha raccomandato l'introduzione di una granularità molto più fine per soddisfare i requisiti più complessi dello spazio profondo. Per differenziare tra loro i bundles con priorità expedited, sono stati introdotti 255 livelli (da 0 a 254, il 255 è riservato alla custody signal) di priorità **ordinali** (100 indica un bundle più urgente di 99, e così via). Inoltre è stato aggiunto un ulteriore e particolare livello chiamato **critical**, da utilizzare solo in situazioni di emergenza. Il nodo che lo riceve infatti deve inviare una copia del bundle a tutti i suoi nodi vicini che hanno una qualche plausibile prospettiva di poter inoltrare il bundle verso la sua destinazione finale senza restituirlo al nodo locale. Ogni bundle "critico" deve essere inoltrato **al massimo una volta** da ciascun nodo perché, se usato senza criterio, porterebbe in breve tempo al collasso dell'intera rete; Questo comportamento è noto col nome di **controlled flooding** (inondazione controllata).

Le classi di priorità aggiunte fanno parte delle estensioni del Bundle Protocol e sono descritte nel dettaglio nel draft Bundle Protocol Extended Class Of Service (ECOS). Si noti che tali estensioni sono implementate sotto forma di extension block da associare ai bundles, per cui i nodi che non prevedono la loro (non obbligatoria) gestione potrebbero ignorarli.

### 6.2.2 General Routing Classification

Alla luce di quanto visto per i bundle critici, è possibile fare una classificazione generale del routing:

- **Static:** in cui la rotta da seguire è sempre la stessa e non cambia nel tempo. È realizzato attraverso tabelle di routing statiche;
- **Dynamic:** (CGR) in cui la rotta da seguire cambia nel tempo. Si possono distinguere due tipologie:
  - **Forwarding schemes:** è il funzionamento classico di CGR che prevede l'invio di una copia del bundle all'entry node della rotta selezionata.
  - **Replication schemes:** è il comportamento adottato da CGR per gestire i bundle critical. Prevede l'invio di una copia del bundle a ciascun vicino del nodo locale.

Se da un lato aumenta la probabilità e la velocità di consegna del bundle alla destinazione finale, comporta un aumento del traffico di dati ridondanti in tutta la rete. Risulta particolarmente adatto alle opportunistic networks.

### 6.2.3 Routing replication schemes in opportunistic environments

Nelle DTN terrestri, non è possibile sfruttare il contact plan nel routing, perché i contatti avvengono in maniera casuale. Per questo tipo di ambienti, noti col nome di **opportunistic environments** sono presenti molte proposte su come effettuare il routing, delle quali si elencano le cinque principali:

- **Epidemic Routing:** è simile al comportamento di CGR per l'instradamento dei bundle critical. La differenza è che il nodo, prima di inviare il bundle, *chiede* agli altri nodi se non lo abbiano già ricevuto. In caso negativo, il bundle viene inviato, altrimenti no (si evita quindi l'invio di repliche). È altamente affidabile ma richiede molte risorse.
- **ProPHET:** Utilizza la non completa casualità dei contatti, come spesso accade negli scenari reali, per replicare e passare bundle solo se il nodo incontrato ha sufficienti possibilità per consegnarli a destinazione.
- **Spray-and-Wait:** *Spruzza* un numero limitato di copie nella rete, quindi *attende* finché uno di questi nodi non incontra ("contatti") la destinazione.
- **MaxProp:** è basato sull'assegnazione delle priorità alla pianificazione dei bundle trasmessi ai nodi DTN e dei bundle da eliminare. Le priorità sono basate sulle probabilità di percorso calcolate sulla base di dati storici e su meccanismi complementari. È stato valutato su una vera rete di bus DTN.
- **RAPID:** Altra variante sul tema, valutata anche su rete bus.

## 6.3 Course Conclusions

Le DTN possono giocare un ruolo chiave nelle comunicazioni spaziali poiché, a differenza del passato, non sono più necessarie specifiche soluzioni per la loro implementazione. Attraverso le DTN, le reti spaziali potrebbero diventare solo una parte di un Internet più grande di quello odierno. In passato è stato spesso provato che le tecnologie utilizzate nello spazio, sono molto utili anche sulla terra: perché questo non dovrebbe essere vero anche per le DTN?

# Chapter 7

## Virtualbricks Lab

### 7.1 Check installation

Non avviare tutti i briks in una volta, procedere all'attivazione singolarmente **in questo ordine**:

1. **switch** a scelta: l'icona dovrebbe diventare colorata;
2. **channel emulator** a scelta: l'icona dovrebbe diventare colorata;
3. **switchwrapper**: se non si avvia, il control network non è configurato a dovere;
  - controllare con il comando `ifconfig` se il tap 1.0.0.1 è attivo (dovrebbe esserlo);
  - controllare che ci siano tutti undici i punti;
4. **VM** a scelta: se non parte potrebbe essere necessario aggiungere l'user al gruppo kvm (vedi punto 10 installazione); ripetere i punti 10 e 11;
5. **ssh**: provare a loggare in una macchina virtuale utilizzando il comando `ssh root@10.0.0.1x` dove **x** rappresenta il numero della macchina virtuale nella quale si tenta di accedere.  
La password è **foobar**

Se tutti i passi precedenti sono eseguiti senza riscontrare errori, la configurazione è corretta e si può procedere con gli esperimenti.

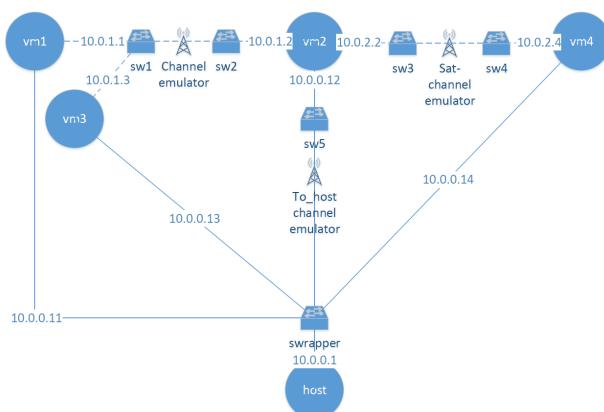
### 7.2 DTN2

E' una piattaforma sperimentale per testare le features del Bundle Protocol su una implementazione reale, può anche essere usata per esperimenti reali. La piattaforma è relativamente semplice da usare poiché presenta solo un file di configurazione per nodo (`/etc/dtn.conf`) e fornisce alcune delle features avanzate del Bundle Protocol come reactive fragmentation, metadata e *opportunistic* links. Tra gli svantaggi è da notare come il suo sviluppo sia fermo alla versione 2.9 rilasciata nel 2012 e come presenti scarso supporto al BSP (Bundle Security Protocol).

#### 7.2.1 DTN2hops layout

La testbed utilizzata durante il corso prevede:

- 4 macchine virtuali (nodi DTN) controllati via ssh dall'host;
- 2 data networks: 10.0.1.x e 10.0.2.x
- 1 control network: 10.0.0.x



### 7.2.2 Start the Bundle Protocol

- **dtnd [-d] [-o <logfile>] [-t] [-c <configfile>]**

Avvia il Bundle Protocol sul nodo nel quale è richiamato. Se lanciato senza alcun argomento è mostrato a schermo il log. Tipicamente il BP è avviato come demone (**-d**) e il log trascritto su un file di output (**-o <logfile>**). L'opzione **-t** effettua la pulizia del database del nodo in cui il demone è lanciato, è consigliato farlo solo all'inizio di ogni nuovo esperimento. È possibile inoltre specificare un file di configurazione esterno mediante **-c <configfile>**.

- **telnet localhost 5050**

Una volta avviato il demone, è possibile interfacciarsi con esso tramite connessione telnet non cifrata a localhost:5050.

### 7.2.3 dtnd shell

La shell è altamente sensibile agli input errati. Se il comando usato non è tra quelli disponibili, o è digitato in modo sbagliato, è necessario usare la combinazione di tasti **ctrl+]** per sbloccarla. A seguito di questa operazione, digitare **quit** e riavviare il demone del BP.

- **help**

Mostra a schermo i comandi validi.

- **exit**

Chiude la shell.

- **link dump**

Mostra a schermo i collegamenti disponibili verso gli altri nodi e il loro stato. La verifica dello stato è effettuata mediante SYN-ACK tra i nodi.

- **route dump**

Mostra a schermo la tabella di routing con i collegamenti disponibili verso gli altri nodi.

- **bundle list**

Mostra a schermo tutti i bundles contenuti nel database locale del nodo (con relativa sorgente e destinazione). I bundles qui elencati possono essere eliminati se e solo se è specificata l'opzione **-t** all'avvio del BP, alla loro scadenza naturale o manualmente.

- **storage set tidy**

Elimina tutti i bundles salvati nel database locale del nodo.

- **bundle dump <bundle\_n>**

Visualizza il contenuto del bundle specificando il sequence number ricavato con **bundle list**.

- **bundle info <bundle\_n>**

Visualizza i parametri associati al bundle. È qui possibile vedere il Global ID, identificativo univoco, così formato: **<source>, <creation\_timestamp.sequence\_number>**.

### 7.2.4 default configuration file: /etc/dtn.conf

- **set type berkeleydb**

Il database di default è il Berkeley.

- **route local\_eid "dtn://<hostname>.dtn"**

Assegna al nodo un identificativo univoco (EID).

- **link add <ltvm> <ip> ALWAYSON|ONDEMAND|OPPORTUNISTIC|SCHEDULED tcp|udp**

Aggiunge un nuovo collegamento di nome **<ltvm>** con il nodo situato all'indirizzo **<ip>**. La modalità può essere in **tcp** o **udp** nel tipo specificata.

- **route add dtn://<hostname>.dtn/\* <ltvm>**

Poiché il BP è situato sopra il livello di trasporto, si serve di TCP/UDP per comunicare con gli altri nodi. Tramite questa riga di testo si comunica al BP che i bundles con destinazione **<hostname>** devono essere instradati usando il collegamento **<ltvm>**.

### 7.2.5 DTNperf

DTNperf ha tre modalità operative: client, server e monitor. Il client genera e invia i bundles, il server li riceve e il monitor raccoglie gli status report in un file csv. Il client, il server e il monitor corrispondono rispettivamente agli indirizzi *from*, *to* e *reply-to*.

#### 7.2.5.1 client

```
dtnperf --client -d <dest_eid> [-T <s> | -D <num> | -F <filename>]
[-W <size> | -R <rate>]
[options]
```

Il client ha tre modalità di trasmissione mutualmente esclusive tra loro di inviare bundles al server.

- **time-mode (-T)**: una serie di bundles con payload dummy della dimensione desiderata (opzione -P) sono generati e inviati al BP.
- **data-mode (-D)**: è uguale alla time-mode, con l'eccezione che la generazione dei bundles si arresta dopo che una quantità data di dati è inviata al BP.
- **file-mode (-F)**: è inviato un file, non payload dummy. E' possibile suddividere il file in bundles di una dimensione desiderata.

Indipendentemente dalla modalità di trasmissione, sono disponibili due politiche di gestione della congestione:

- **rate-base (-R)**: Genera traffico nella rete con rate costante. Il server non risponde a questo traffico (comportamento simile a UDP).
- **window-based (-W)**: La finestra di congestione W rappresenta il numero massimo di bundles in-flight (cioè non ACK-ppati). Il meccanismo è simile alla finestra di congestione di TCP, con la differenza che: W è di dimensione fissa, i bundles in-flight possono essere non-consecutivi e non ci sono ritrasmissioni (gli ACK sono usati solo per innescare la trasmissione di un nuovo bundle).

#### 7.2.5.2 server

```
dtnperf --server [options]
```

Avvia un'istanza del server in grado di gestire client multipli.

#### 7.2.5.3 monitor

```
dtnperf --monitor [options]
```

Avvia un'istanza del monitor in grado di ricevere e collezionare gli status report. Può essere lanciato su un nodo esterno oppure sullo stesso nodo in cui è presente un client.

#### 7.2.5.4 Basic applications

- `dtnperf --client -d dtn://vm2.dtn -T15 -W1`  
invia alla vm2 bundle con dimensione di default (50kB) per 15 secondi, uno alla volta. -W1 impone che ci sia solo un bundle in volo: prima di procedere alla generazione e trasmissione di  $B_i$ , è necessario aspettare l'ACK di  $B_{i-1}$ .
- `dtnperf --client -d dtn://vm4.dtn -D100k -P50k -R1b -l 1000`  
invia alla vm4 due bundles (con payload dummy) di 50 kilobytes (per un totale di 100kb) al tasso di 1 bundle/s impostando la scadenza a 1000 secondi dal momento di generazione.
- `dtnperf --client -d dtn://vm4.dtn [...] -C -f -r --monitor dtn://vm2.dtn`  
invia alla vm4 dei bundles con la custody option attiva e invia gli status report (forwarded and received status) al monitor avviato sulla vm2.
- `dtnperf --client -d dtn://vm2.dtn -F picture.jpg -P 100kB -W4`  
invia un file utilizzando bundle con payload di 100kb alla volta. E' molto utile nei casi in cui si riesca a far coincidere la dimensione con il contact volume. E' un metodo da utilizzare come alternativa alla proactive fragmentation. Si noti inoltre che la segmentazione del file è qui effettuata a livello di applicazione da DTNperf, mentre la bundle fragmentation è effettuata al Bundle Layer.

### 7.3 Traffic analysis with Wireshark

Wireshark è un software utile per effettuare analisi del traffico di rete. Per capire meglio il funzionamento delle DTN può essere utile effettuare tali analisi su connessioni basilari, come quella proposta nel seguito.

1. **VM1:** `tcpdump -i ens4 -w outfile.dump &`

Con questo comando si avvia sul nodo (*vm4* in questo caso) lo sniffing del traffico in entrata e uscita dall’interfaccia `ens4` relativa al BP installato (reperibile tramite `ifconfig`). Tutto il traffico è trascritto su un file di output e, per convenienza, eseguito in background.

2. **VM4:** `dtnperf --server`

3. **VM1:** `dtnperf --client -d dtn://vm4.dtn -D10k -P5k -l 1000`

4. **VM1:** `fg ; ctrl+c`

Il processo in background è richiamato in foreground e successivamente interrotto. All’interruzione è possibile trovare nella directory corrente il file di output `outfile.dump`.

5. **VM4:** `scp root@10.0.0.11:/root/ion/*.dump ./`

Il file di output è inviato dalla *vm1* all’host dal quale è possibile analizzare il traffico utilizzando wireshark.

### 7.4 Experiment 1: GEO satellites with fixed terminal

L’esperimento proposto è relativo al **solo** caso **congestionato** di comunicazione tra terminale **fisso** e satellite **GEO**. La topologia di rete è illustrata in figura 4.3: la *vm1* è il sender terrestre, il *channel emulator* rappresenta il canale R1-R2 terrestre, il *sat channel emulator* rappresenta il collegamento satellitare R2-receiver, la *vm4* è il receiver satellitare. La *vm2* e la *vm3* sono nodi a terra che servono a simulare la congestione della rete cablata.

#### 7.4.1 pre-settings

##### 7.4.1.1 TCP buffers

I buffer di invio e ricezione di TCP sono ingranditi alla dimensione di 8MB con l’esecuzione, in ciascuna VM, dello script `tcp_sat_settings.sh`:

```
# wmem/rmem large
sysctl -w net.ipv4.tcp_wmem='8388608 8388608 8388608'
sysctl -w net.ipv4.tcp_rmem='8388608 8388608 8388608'
sysctl -w net.ipv4.tcp_congestion_control=hybla
# wmem/rmem short
#sysctl -w net.ipv4.tcp_wmem='4096 16384 16384'
#sysctl -w net.ipv4.tcp_rmem='4096 16384 16384'
#Ready to use
#sysctl -w net.ipv4.tcp_congestion_control=cubic
#sysctl -w net.ipv4.tcp_congestion_control=reno
#sysctl -w net.ipv4.tcp_window_scaling=1
#sysctl -w net.ipv4.tcp_sack=1
#sysctl -w net.ipv4.tcp_timestamps=1
sysctl -w net.ipv4.route.flush=1
```

Quindi:

1. `scp tcp_sat_settings.sh root@10.0.0.11:/root/ion/tcp_sat_settings.sh`
2. `chmod +x tcp_sat_settings.sh`
3. `./tcp_sat_settings.sh`

#### 7.4.1.2 Channels configuration

I canali della testbed sono adattati allo scenario in esame:

- **channel emulator:**

- buffer lenght: 75000
- propagation delay: 12
- bandwidth: 1250000

- **sat channel emulator:**

- buffer lenght: 75000
- propagation delay: 285
- bandwidth: 1250000

Per verificare che i settaggi siano stati effettivamente applicati, è consigliato effettuare un ping da *vm1* a *vm2* e *vm4* e accertarsi che l'RTT sia quello desiderato.

#### 7.4.1.3 VM2 isolation

I collegamenti *ltvm1* e *ltvm3* devono essere chiusi, per farlo è necessario commentare nel file */etc/dtn.conf* la riga corrispondente. Il file di configurazione risulta strutturato nel modo seguente:

```
#link add ltvm1 10.0.1.1 ALWAYSON tcp
#link add ltvm3 10.0.1.3 ALWAYSON tcp
link add ltvm4 10.0.2.4 ALWAYSON tcp

#route add dtn://vm1.dtn/* ltvm1
#route add dtn://vm3.dtn/* ltvm3
route add dtn://vm4.dtn/* ltvm4
```

#### 7.4.2 Background traffic

Per simulare il traffico di background presente nella rete cablata terrestre, si utilizzano *vm2* come server IP e *vm3* come client IP. Nello specifico si utilizza il comando *iperf* per stabilire cinque connessioni parallele, durante le quali è inoltrato traffico dummy. I comandi per avviare client e server IP sono eseguiti ogni volta che si vuole provare l'esperimento con una versione TCP/DTN differente:

1. **VM3:** *iperf -s*
2. **VM2:** *iperf -c 10.0.1.2 -P5 -t1200*

Si noti che, per ottenere risultati significativi, la durata specificata deve essere sufficientemente lunga a coprire il periodo di tempo in cui *vm1* e *vm4* comunicano tra loro.

#### 7.4.3 Experiment

L'esperimento consiste nella valutazione del goodput ottenuto al receiver satellitare utilizzando tre versioni differenti di TCP (hybla, reno, cubic) e DTN.

Nel confronto delle versioni TCP *vm4* è il server, mentre *vm1* è il client (*-Z <version>*) che invia pacchetti dummy per un totale di 10000 byte:

1. **VM4:** *iperf -s*
2. **VM1:** *iperf -c 10.0.2.4 -n 10000 -Z hybla|reno|cubic*

Per DTN:

1. **VM4:** *dtnperf --server*
2. **VM1:** *dtnperf --client -d dtn://vm4.dtn -T20 -P1M -W2*

Per assicurarsi che l'esperimento sia effettivamente riuscito, è necessario confrontare i risultati ottenuti con quelli presenti in figura 4.4 per il caso congestionato.

## 7.5 Experiment 2: LEO satellite data mule

L'esperimento proposto è relativo al caso di satelliti **LEO** utilizzati in modalità **data mule**. La topologia della rete è illustrata in figura 4.7: la *vm1* è il nodo sorgente, il *channel emulator* (nel seguito CE) è il collegamento UGS-LEO, il *sat channel emulator* (nel seguito SATCE) è il collegamento LEO-DGS, la *vm4* è il nodo ricevente e la *vm2* è il satellite LEO.

Destinazione e sorgente non sono mai visibili contemporaneamente al satellite, per cui si parte da una **situazione iniziale con CE e SATCE spenti** (manualmente da virtual-bricks). Si noti che il satellite, qui rappresentato da *vm2*, ha il demone del BP in esecuzione, ma non un dtnperf client/server. Esso agisce come nodo intermedio nella rete tra *vm1* e *vm4* inoltrando i bundles verso le rispettive destinazioni. Si noti inoltre che non vi è alcuna distinzione tra sender-UGS e receiver-DGS, in quanto il trasferimento su questi collegamenti è istantaneo e di scarsa rilevanza per l'esperimento. A seguire sono illustrati i passi effettuati:

1. **VM4:** `dtnperf --server`
2. **VM1:** `dtnperf --client -d dtn://vm4.dtn -D100k -P50k -W2 -l 1000`  
il client *vm1* genera e invia due bundle al BP in direzione *vm4*, in seguito resta in attesa dei due ACK corrispondenti. Poiché allo stato attuale il nodo è isolato, è possibile vedere i bundle nel DB della *vm1* (`telnet localhost 5050; bundle list`).
3. **Virtualbricks:** il CE è riattivato (i.e. il satellite *vm2* è visibile a *vm1*). Dopo un breve intervallo di tempo, i bundle sono automaticamente trasferiti da *vm1* a *vm2*.
4. **Virtualbricks:** il CE è disattivato (i.e. il satellite *vm2* è in rotazione attorno al globo e invisibile sia a *vm1* che a *vm4*). E' possibile vedere i bundles memorizzati nel DB della *vm2* (`telnet localhost 5050; bundle list`).
5. **Virtualbricks:** il SATCE è riattivato (i.e. il satellite *vm2* è visibile da *vm4*). Dopo un breve lasso di tempo i bundle sono automaticamente trasferiti da *vm2* a *vm4* (il DB di *vm4* è vuoto perché avendo ricevuto i suoi bundles, li ha già eliminati). **Istantaneamente**, *vm4* invia i due ACK previsti a *vm2*.
6. **Virtualbricks:** il SATCE è disattivato (i.e. il satellite *vm2* è in rotazione attorno al globo e invisibile sia a *vm1* che a *vm4*). E' possibile vedere i bundles relativi agli ACK memorizzati nel DB della *vm2* (`telnet localhost 5050; bundle list`).
7. **Virtualbricks:** il CE è riattivato (i.e. il satellite *vm2* è visibile a *vm1*). Dopo un breve intervallo di tempo, i bundles degli ACK sono automaticamente trasferiti da *vm2* a *vm1* e la comunicazione termina.

# Bibliography

- [1] N. ALESSI, S. BURLEIGH, C. CAINI, AND T. DE COLA, *Ltp robustness enhancements to cope with high losses on space channels*, in 2016 8th Advanced Satellite Multimedia Systems Conference and the 14th Signal Processing for Space Communications Workshop (ASMS/SPSC), IEEE, 2016, pp. 1–6.
- [2] N. ALESSI, C. CAINI, T. DE COLA, S. MARTIN, AND J. P. MAYER, *Dtn performance analysis of multi-asset mars-earth communications*, International Journal of Satellite Communications and Networking, (2014).
- [3] P. APOLLONIO, C. CAINI, AND V. FIORE, *From the far side of the moon: Delay/disruption-tolerant networking communications via lunar satellites*, China Communications, 10 (2013), pp. 12–25.
- [4] P. APOLLONIO, C. CAINI, M. GIUSTI, AND D. LACAMERA, *Virtualbricks for dtn satellite communications research and education*, in International Conference on Personal Satellite Services, Springer, 2016, pp. 76–88.
- [5] E. J. BIRRANE AND K. MCKEEVER, *Bundle Protocol Security Specification*, Internet-Draft draft-ietf-dtn-bpsec-25, Internet Engineering Task Force, Dec. 2020. Work in Progress.
- [6] S. BURLEIGH, *Bundle protocol extended class of service (ecos)*, draft-irtf-dtnrg-ecos-05, (2013).
- [7] S. BURLEIGH, C. CAINI, J. J. MESSINA, AND M. RODOLFI, *Toward a unified routing framework for delay-tolerant networking*, in 2016 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), 2016, pp. 82–86.
- [8] S. BURLEIGH, K. FALL, AND E. J. BIRRANE, *Bundle Protocol Version 7*, Internet-Draft draft-ietf-dtn-bpbis-26, Internet Engineering Task Force, 7 2020. Work in Progress.
- [9] S. BURLEIGH, M. RAMADAS, S. FARRELL, ET AL., *Licklider transmission protocol-motivation*, IETF Request for Comments RFC, 5325 (2008).
- [10] C. CAINI, H. CRUICKSHANK, S. FARRELL, AND M. MARCHESE, *Delay-and disruption-tolerant networking (dtn): an alternative solution for future satellite networking applications*, Proceedings of the IEEE, 99 (2011), pp. 1980–1997.
- [11] C. CAINI, A. d'AMICO, AND M. RODOLFI, *Dtnperf\_3 at work: aims and use*, in Proceedings of the 8th ACM MobiCom workshop on Challenged networks, 2013, pp. 53–56.
- [12] C. CAINI, R. FIRRINCIELI, T. DE COLA, I. BISIO, M. CELLO, AND G. ACAR, *Mars to earth communications through orbiters: Delay-tolerant/disruption-tolerant networking performance analysis*, International Journal of Satellite Communications and Networking, 32 (2014), pp. 127–140.
- [13] V. CERF, S. BURLEIGH, A. HOOKE, L. TORGERSON, R. DURST, K. SCOTT, K. FALL, AND H. WEISS, *Rfc 4838, delay-tolerant networking architecture*, IRTF DTN Research Group, (2007).
- [14] S. FARRELL, H. WEISS, S. SYMINGTON, AND P. LOVELL, *Bundle Security Protocol Specification*. RFC 6257, May 2011.
- [15] M. RAMADAS, S. BURLEIGH, S. FARRELL, ET AL., *Licklider transmission protocol-specification*, IETF request for comments RFC, 5326 (2008).
- [16] K. SCOTT, S. BURLEIGH, ET AL., *Rfc 5050: bundle protocol specification*, IRTF DTN Research Group, (2007).