ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
CORSO DI LAUREA IN INGEGNERIA E SCIENZE INFORMATICHE

Il protocollo SCTP e il suo utilizzo nelle reti 4G e 5G

Elaborato in:
RETI DI TELECOMUNICAZIONE

Relatore:
Prof. Franco Callegati

Presentata da:
Paolo Garroni

Anno Accademico 2019/2020

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione a SCTP

SCTP è un protocollo di trasporto per reti a commutazione di pacchetto IP, creato per superare i limiti di TCP e UDP in contesti applicativi con requisiti di alta disponibilità, performance e ordinamento parziale dei dati.

In questo capitolo verranno introdotti il contesto storico, le esigenze e le motivazioni che hanno portato alla creazione di SCTP e verranno descritte le sue caratteristiche ad alto livello. Nei seguenti capitoli verranno discusse le caratteristiche in maggiore dettaglio.

1.1 Il contesto storico: SS7 over IP (SIGTRAN)

SCTP nasce intorno agli anni Duemila, nell'ambito della suite di protocolli *SIGTRAN (SIGnaling TRANsport)*: tali protocolli rispondono all'esigenza di trasportare su rete IP i messaggi di segnalazione *Signaling System No.7 (SS7)*, storicamente utilizzati per il piano di controllo della rete telefonica generale *Public Switched Telephone Network (PSTN)* a commutazione di circuito.

In questa sezione verranno introdotti la rete PSTN e i protocolli SS7 in modo da comprenderne i requisiti tipici della commutazione di circuito che SCTP deve soddisfare nonostante la natura *best-effort* della rete IP sottostante.

1.1.1 Evoluzione della rete PSTN

Le origini della rete PSTN risalgono a fine Ottocento, pochi anni dopo l'assegnazione del brevetto del telefono ad Alexander Graham Bell nel 1876. Da allora la rete è stata in continua evoluzione ed espansione: gli operatori di telecomunicazioni nazionali ed internazionali hanno coperto le zone geografiche in modo capillare sviluppando le proprie reti e l'interconnessione di queste ultime ha realizzato una enorme rete con copertura globale.

L'infrastruttura esistente, grazie alla sua grande diffusione e capillarità del territorio, è stata sfruttata dagli operatori per trasportare oltre ai servizi di telefonia classica anche tutti i servizi digitali. Analogamente, la telefonia mobile in tutte le sue generazioni è sempre stata integrata e interoperabile con la già esistente rete PSTN.

Per quanto si possano considerare le reti telefoniche e le reti a banda larga come oggetti separati e distinti, tale indipendenza e autonomia è logica e non fisica: gli apparati di gestione e smistamento dei flussi di informazione (telefonia classica, internet) sono in effetti oggetti differenti, ma il trasferimento fisico avviene utilizzando le stesse strutture portanti (fibra, rame, ponti radio) e i medesimi apparati di trasmissione. Le reti mono-servizio esistenti sono interdipendenti rispetto alla rete trasmissiva che è una risorsa comune.

Negli ultimi decenni nei livelli gerarchicamente più alti della rete gli apparati sono stati progressivamente sostituiti con apparati in grado di gestire multiple tipologie di servizio: i router IP. Gli apparati dedicati alla gestione di un solo tipo di servizio (telefonia, internet, eccetera), si concentrano invece nelle sole centrali periferiche, dove avviene il processo di separazione dei compiti.

La convergenza delle varie reti esistenti (telefonica fissa, telefonica mobile, Internet a banda larga, eccetera) verso una unica rete IP è una naturale conseguenza della crescita esponenziale di Internet, che ha reso evidente la flessibilità e la capacità di adattamento delle reti IP. Basti pensare alla semplicità con la quale attraverso un unico collegamento a banda larga è ormai possibile usufruire dei più svariati servizi (web-browsing, e-mail, instant-messaging, streaming audio/-video, fonia, videoconferenze, eccetera): questa pluralità di servizi non richiede

alcun intervento sulla rete in quanto il protocollo IP rende disaccoppiata la rete trasmissiva dai servizi offerti.

Gestire e manutenere una unica rete IP che trasporta il traffico delle varie reti mono-servizio apporta indubbi vantaggi per gli operatori in termini di costi e facilità di gestione.

La rete PSTN e quella IP hanno però caratteristiche e logiche di funzionamento molto diverse tra loro.

I servizi di telefonia PSTN rispondono a logiche di instradamento a commutazione di circuito: l'utente è connesso direttamente e in modo fisso alla centrale e durante una telefonata viene instaurato un canale diretto, stabile e dedicato tra due utenti e viene mantenuto per tutta la sua durata, anche nei momenti di silenzio. Ciò garantisce qualità del servizio in termini di affidabilità e latenza minima che è ideale per la telefonia e per altre applicazioni real-time. Il rovescio della medaglia è una gestione più inefficiente e costosa della rete trasmissiva in quanto quando viene instaurato un canale quest'ultimo non può essere utilizzato per trasferire i segnali di altri utenti nei momenti in cui è inutilizzato (per esempio, durante le pause nella telefonata).

Al contrario la commutazione di pacchetto delle reti IP prevede la suddivisione del flusso dati in pacchetti i quali possono arrivare a destinazione seguendo percorsi differenti, eventualmente fuori ordine, eventualmente con perdite, ma occupano risorse solo nel momento di una effettiva necessità e consentono un utilizzo più efficiente e meno costoso della rete trasmissiva.

Si passa cioè da una gestione che prevede risorse preventivamente e staticamente assegnate a una gestione in cui la risorsa trasmissiva cambia alle esigenze specifiche del momento. Tramite IP la rete trasmissiva è in grado di gestire tipologie di traffico multiple in modo efficiente e ciò consente importanti economie di scala e di scopo.

IP è però un servizio best-effort e gestisce tali flussi in modo indifferenziato: è evidente che per i contesti che richiedono garanzia di una certa qualità del servizio ciò rappresenta una importante criticità.

1.1.2 Signaling System No.7 (SS7)

Il **Signaling System No.7 (SS7)** è un insieme di protocolli di segnalazione telefonica sviluppato dall'ITU e diventato di uso standard nella rete PSTN. Creato a fine anni Ottanta si è evoluto nel tempo contestualmente alle tecnologie di rete fissa e mobile.

SS7 viene usato per controllare la maggior parte delle chiamate telefoniche, fisse o mobili, in logica di commutazione a circuito. Consente inoltre numerosi altri servizi tra cui gli SMS.

I protocolli SS7 sono *segnalazioni a canale comune (CSS)*: le informazioni di controllo vengono trasmesse su un canale comune su un mezzo diverso rispetto a quello utilizzato per la voce (all'opposto delle *segnalazioni associate al canale (CAS)*, trasmesse sullo stesso canale e mezzo del segnale fonico). I messaggi SS7 dispongono quindi di linee trasmissive ed apparati appositamente dedicati, che costituiscono una rete di segnalazione fisicamente distinta e parallela alla rete di trasmissione dei segnali fonici.

I messaggi di segnalazione sono per loro natura segnali discontinui: conviene inviarli a pacchetti. La rete di segnalazione SS7 adotta infatti una commutazione di pacchetti per controllare l'instaurazione e la chiusura delle chiamate telefoniche, le quali al contrario avvengono per commutazione di circuito.

La architettura a canale comune ha consentito, oltre alla funzione di controllo delle telefonate, anche lo sviluppo delle *Intelligent Network (IN)*, di cui un esempio è mostrato in fig. 1.1: i sistemi di commutazione *Service Switching Point (SSP)* distribuiti su tutta la rete inviano messaggi di segnalazione lungo la rete dedicata SS7. Tali messaggi di segnalazione sono instradati da nodi *Signal Transfer Point (STP)* e giungono ai nodi centralizzati *Service Control Point (SCP)* dove risiede la logica del servizio e che fungono da interfaccia con i database degli operatori di telecomunicazioni (contenenti per esempio le numerazioni cliente e i dati specifici dei servizi offerti).

Questo tipo di architettura offre all'operatore di rete i mezzi per sviluppare e controllare i servizi più efficientemente: consente di implementare nuove funzionalità rapidamente per fornirle ai clienti nel minore tempo possibile. Nella

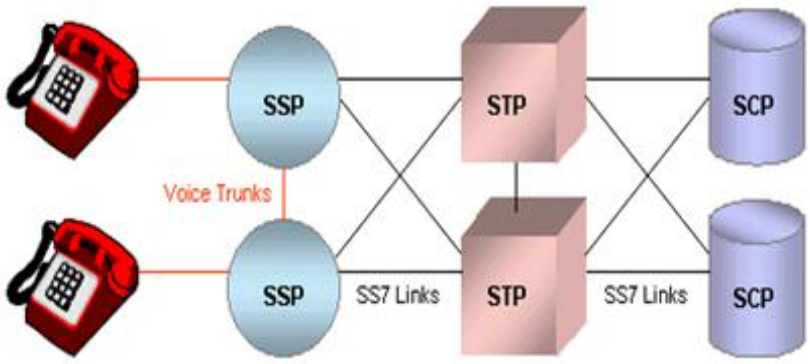

Figura 1.1: Architettura semplificata di una rete Intelligent Network SS7

rete PSTN sono stati implementati in questo modo numerosi servizi supplementari aggiuntivi rispetto alla semplice telefonata: il trasferimento di chiamata, la conversazione tra più utenti, l'avviso di chiamata, la flessibilità delle tariffazioni delle chiamate, i numeri verdi, la portabilità del numero telefonico, il televoto, i servizi di contact-center e altri ancora.

Una volta introdotti, i servizi sono personalizzati facilmente per incontrare le necessità dei clienti.

SS7 opera a commutazione di pacchetto ma utilizza uno stack di protocolli propri dedicati, mostrati in fig. 1.2.

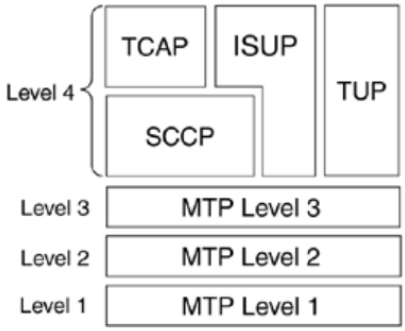

Figura 1.2: Stack dei protocolli SS7

In particolare nei livelli L1, L2 e L3 sono utilizzati una serie di protocolli a cui ci si riferisce collettivamente con il nome *Message Transfer Part (MTP)* che

garantisce un trasporto affidabile, in ordine e senza duplicati dei messaggi tra i nodi della rete SS7.

Il sistema SS7 è stato ampliamente utilizzato nella rete PSTN per svariati anni ed è stato molto apprezzato per il suo funzionamento efficiente e adeguato al suo scopo. Tuttavia negli anni Duemila, con la crescita esponenziale di Internet e la graduale conversione della rete PSTN in una rete IP, emerse la necessità di adattare SS7 al trasporto best-effort della rete IP senza compromettere l'efficienza, affidabilità e alta disponibilità che lo contraddistinguono.

1.1.3 SS7 over IP (SIGTRAN)

SIGTRAN (SIGnaling TRANsport) è un IETF working-group che sviluppò intorno agli anni Duemila una suite di protocolli con lo scopo di adattare i protocolli SS7 della rete telefonica generale PSTN al trasporto su rete IP.

In un certo senso si può quindi considerare la suite di protocolli SIGTRAN come una estensione di SS7 che, pur conservandone la natura di commutazione a circuito e senza comprometterne i requisiti funzionali e di performance, ne consente il trasporto su rete IP ('SS7-over-IP').

Nello specifico gli obiettivi di SIGTRAN erano:

- Definire i requisiti architetturali e di performance necessari a trasportare i protocolli di segnalazione della PSTN su rete IP
- Valutare i protocolli esistenti e se necessario definire un nuovo protocollo di trasporto che ottemperasse ai requisiti definiti
- Definire metodi di incapsulamento dei vari protocolli di segnalazione PSTN

Nel 1999 IETF pubblicò il RFC 2719 [19] che definiva il framework architetturale e i requisiti di performance di SIGTRAN identificando tre componenti necessari alla sua realizzazione:

- Il protocollo di rete IP
- Un protocollo di trasporto comune adeguato ai requisiti del trasporto delle segnalazioni telefoniche

- Un insieme di *adaptation-layer* che supportassero le primitive dei protocolli di segnalazione PSTN e li adattassero al protocollo di trasporto sottostante

Come si può vedere in fig. 1.3, SCTP è il protocollo di trasporto su base IP comune a tutti gli adaptation-layer che fungono da interfaccia per i protocolli SS7 al livello superiore. In questo modo gli operatori possono mantenere l'esistente sistema di segnalazioni SS7 e trasportarlo su rete IP.

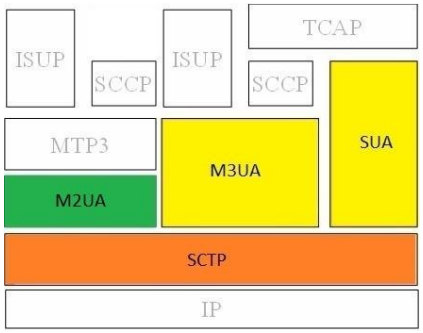

Figura 1.3: Stack dei protocolli SIGTRAN

1.2 Limiti di UDP e TCP

Il protocollo di trasporto comune previsto per la suite di protocolli SIGTRAN doveva essere indipendente dai protocolli di segnalazione PSTN al suo livello superiore e doveva soddisfarne i requisiti stringenti di affidabilità e bassa latenza. Il Working Group valutò i due principali protocolli di trasporto esistenti: UDP e TCP. UDP fu subito scartato perché non soddisfava il requisito di affidabilità e trasporto ordinato. TCP al contrario soddisfava tali requisiti minimi ma aveva diverse limitazioni.

I limiti e le criticità principali di TCP da risolvere con il nuovo protocollo SCTP si possono riassumere nei seguenti aspetti [27] [24]:

- **Necessità di un protocolo affidabile ma orientato ai messaggi**: TCP è al contrario *stream-oriented* e le applicazioni devono aggiungere i propri

marcatori per determinare l'inizio e la fine dei messaggi. Inoltre è necessario l'utilizzo esplicito del flag Push (PSH) per trasferire il messaggio in un tempo ragionevole.

- **Necessità di un protocollo affidabile che consenta il non-ordinamento o l'ordinamento parziale dei dati**: UDP non garantisce né l'affidabilità né l'ordinamento dei dati. TCP al contrario garantisce sia l'affidabilità che l'ordinamento. In alcuni casi però l'ordinamento di TCP è causa di ritardi non necessari: l'ordinamento stretto di TCP è causa di HOL-blocking (vedi sezione 2.9.4) e nel caso di flussi logici di dati distinti tra loro l'ordinamento e i conseguenti ritardi sono ingiustificati

- **Necessità di meccanismi di failure-detection rapida e ottimizzazione dei timer di ritrasmissione**: il tempo necessario a una connessione TCP per rilevare un malfunzionamento della rete è troppo lungo, mancano meccanismi nativi di rilevazione rapida dei malfunzionamenti di rete e di ottimizzazione dei timer di ritrasmissione (nella maggior parte delle implementazioni).

- **Necessità di meccanismi nativi di fail-over**: per raggiungere i requisiti richiesti in termini di alta disponibilità e resilienza ai malfunzionamenti di rete sono necessari meccanismi nativi come il *multi-homing*, ovvero la disponibilità di più interfacce di rete per la stessa connessione tra end-point

- **Necessità di eliminare gli stati *half-open* che creano vulnerabilità agli attacchi SYN-flood**: TCP è relativamente vulnerabile ad attacchi *Denial of Service (DoS)*, in particolare di tipo *SYN flood* nel quale il server viene inondato di richieste SYN inizializzando diverse connessioni per le quali l'attaccante non terminerà mai l'handshake, lasciandole di fatto aperte e congestionando il server

1.3 Caratteristiche di SCTP

La tabella 1.1 pone a confronto le principali caratteristiche di UDP, TCP e SCTP.

Attribute	**TCP**	**UDP**	**SCTP**
Reliability	Reliable	Unreliable	Reliable
Connection Management	Connection-oriented	Connectionless	Connection-oriented
Transmission	Byte-oriented	Message-oriented	Message-oriented
Flow Control	Yes	No	Yes
Congestion Control	Yes	No	Yes
Fault Tolerance	No	No	Yes
Data Delivery	Strictly Ordered	Unordered	Partially ordered
Security	Yes	Yes	Improved

Tabella 1.1: Caratteristiche di TCP, UDP e SCTP a confronto

Come si può vedere SCTP combina caratteristiche di entrambi i protocolli:

- **Orientamento alla connessione** (come TCP)
- **Affidabilità** (come TCP)
- **Controllo di flusso e di congestione** (come TCP)
- **Ordinamento** (come TCP)
- **Orientamento ai messaggi** (come UDP e diversamente da TCP che invece è orientato al flusso di byte)

Inoltre SCTP ha caratteristiche aggiuntive proprie, in grado di sopperire ai limiti evidenziati in sezione 1.2:

- **Ottimizzazione dei timer di ritrasmissione**: TCP consente la configurazione dei timer di ritrasmissione che quindi possono essere ottimizzati (vedi sezione 2.6.3)

- **Multi-homing (fail-over, fault-tolerance)** SCTP prevede l'utilizzo di più interfacce di rete per ciascuno dei due end-point coinvolti nella comunicazione, constituendo quindi diversi *path* possibili. In caso di *failure-detection* il path può essere cambiato consentendo quindi una maggiore resilienza ai malfunzionamenti di rete nelle applicazioni che richiedono alta disponibilità (vedi sezione 2.8.3)

- **Heartbeat (failure-detection)**: SCTP prevede il monitoraggio dello stato di connessione di ogni path attraverso meccanismi di *heartbeat* consentendo la rilevazione precoce dei malfunzionamenti di rete (vedi sezione 2.6.5)

- **Multi-streaming (ordinamento parziale dei flussi logici)**: SCTP consente di suddividere i dati inviati in flussi logici indipendenti tra loro. In questo modo le perdite di dati relativi a un flusso non causano ritardo negli altri flussi che, se ricevuti correttamente, possono essere consegnati immediatamente al livello superiore (problema noto come *HOL-blocking* presente in TCP, vedi sezione 2.9.4)

- **Assenza di stati vulnerabilità ad attacchi DoS SYN-flood**: SCTP, utilizzando un meccanismo basato sui cookie, alloca le risorse solamente quando la procedura di handshake è terminata. Elimina quindi lo stato *half-open* che causa la vulnerabilità di TCP agli attacchi *Denial of Service (DoS)* di tipo *SYN flood* (vedi sezione 2.3.1)

Capitolo 2

Principi di funzionamento di SCTP

2.1 Struttura del pacchetto SCTP

Il pacchetto SCTP, mostrato in fig. 2.1, è composto da

1. **Common Header**: contiene le informazioni di identificazione e di verifica del pacchetto SCTP

2. Uno o più **chunk** 'pezzi' che fungono da contenitori di dati utente o dati di controllo

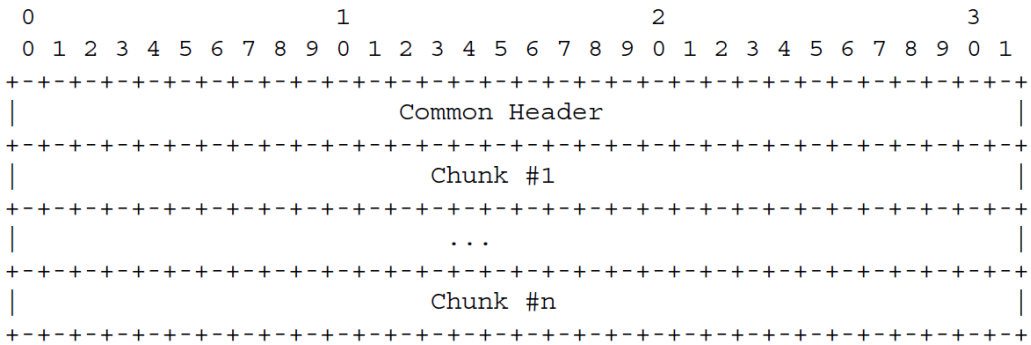

Figura 2.1: Struttura del pacchetto SCTP

2.1.1 Common Header

La Common Header, mostrata in fig. 2.2, contiene le informazioni di identificazione e di verifica del pacchetto SCTP e ha dimensione fissa di 12 byte.

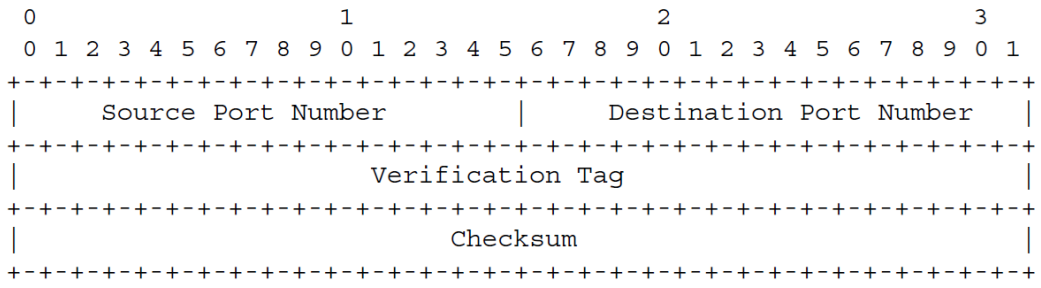

Figura 2.2: Struttura della Common Header

La struttura della Common Header è composta dai seguenti campi:

Source port è un *unsigned integer* di 16 bit che contiene il numero di porta del mittente. Viene utilizzata dal destinatario per identificare l'associazione SCTP al quale il pacchetto appartiene, insieme al numero di porta destinazione e gli indirizzi IP sorgente e destinazione. Il valore 0 è proibito.

Destination port è un *unsigned integer* di 16 bit che contiene il numero di porta del destinatario. Il destinatario lo utilizza per effettuare il de-multiplexing del pacchetto SCTP verso il processo applicativo corretto. Il valore 0 è proibito.

Verification tag è un *unsigned integer* di 32 bit che viene utilizzato da ciascun end-point per identificare la associazione corrente al destinatario ed evitare che pacchetti appartenenti a sessioni precedenti vengano interpretati come parte dell'associazione attuale. Si tratta di un valore random generato e scambiato durante la procedura di associazione (vedi sezione 2.2.2 e sezione 2.3.1) e utilizzato durante tutta la vita dell'associazione.

Checksum è un *unsigned integer* di 32 bit utilizzato per verificare l'integrità del pacchetto in ricezione: il mittente prima di inviare il pacchetto imposta a

0 il campo, calcola il checksum dell'intero pacchetto mediante l'algoritmo *CRC32c* polinomiale per la rilevazione di errori ed inserisce il risultato nel campo lasciando inalterato il resto del pacchetto. In ricezione viene salvato il valore Checksum, impostato a 0 il campo e calcolato nuovamente il checksum mediante il medesimo algoritmo: se i due valori differiscono il pacchetto non è valido e viene ignorato.

2.1.2 Chunk

Un *chunk* 'pezzo', la cui struttura generica è mostrata in fig. 2.3, è un contenitore di dati utente o dati di controllo il cui contenuto e struttura sono identificati dal campo **Type**.

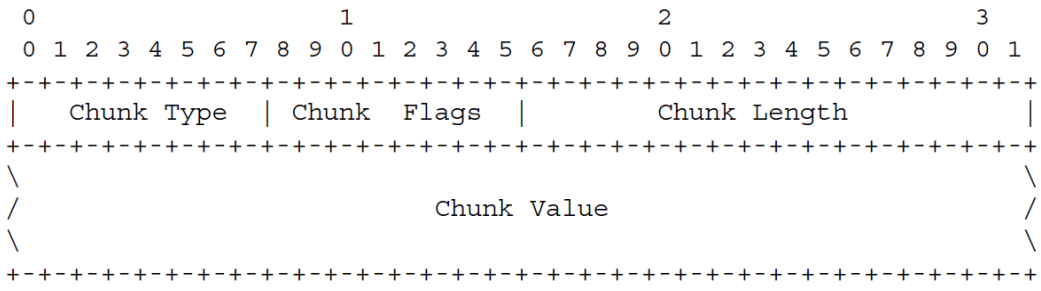

Figura 2.3: Struttura generica di un chunk

La dimensione di ogni chunk è variabile (minimo 4 byte) e specificata nel proprio campo **Length**.

Ad ogni chunk viene aggiunto se necessario un padding di 0 in modo che la dimensione sia sempre un multiplo di 4 byte. Tale padding è escluso dalla dimensione Length specificata.

La struttura dei chunk nel pacchetto SCTP comprende i seguenti campi:

Type è un *unsigned integer* di 8 bit che identifica il tipo di dati contenuti nel chunk. Ad ogni type definito corrisponde una struttura dati specifica contenente varie informazioni di controllo e/o dati utente.

Flags sono 8 bit di segnalazione definiti in modo specifico per ogni Type. Se non specificato diversamente sono impostati a 0 dal mittente e ignorati dal ricevente.

Length è un *unsigned integer* di 16 bit che indica la dimensione in byte del chunk, comprensiva dei campi Type, Flags, Length e Value (il valore minimo è quindi di 4 byte nel caso di Value vuoto). Il valore di Length indica la dimensione effettiva escluso l'eventuale padding finale aggiunto: il protocollo SCTP prevede infatti l'aggiunta di un padding di 0 ad ogni chunk in modo tale che la dimensione totale sia multipla di 4 byte.

Value (Data) è un campo a dimensione variabile (nota al ricevente grazie al campo Length) contenente i dati veri e propri trasportati dal chunk. L'utilizzo e il formato di tale campo è specifico per ogni tipo di chunk.

2.1.3 Parametri dei chunk di controllo

I chunk di controllo possono contenere, nel proprio campo Value, uno o più parametri a dimensione variabile. Tali parametri sono definiti in modo specifico per ogni Type che li supporta ma è prevista una struttura generica di formato **Type-Length-Value (TLV)** comune a tutti i parametri, mostrata in fig. 2.4. Come si può notare la struttura è molto simile a quella utilizzata per i chunk.

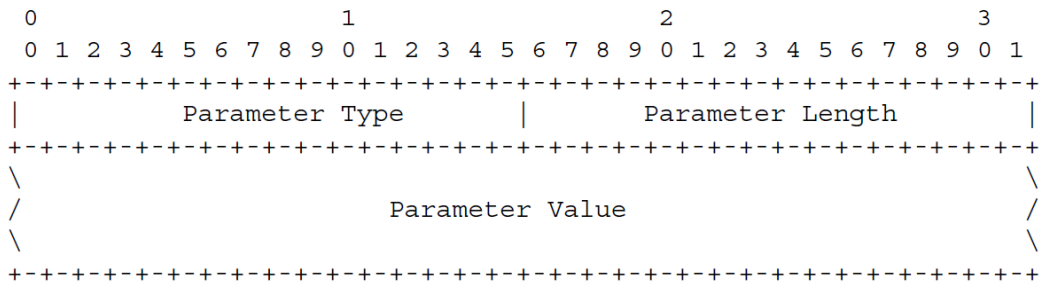

Figura 2.4: Struttura TLV dei parametri dei chunk di controllo

Type è un *unsigned integer* di 16 bit che identifica il tipo di parametro. É da notare che tale valore deve essere univoco tra tutti i chunk. Per esempio, il

Type = 5 è il parametro utilizzato per rappresentare l'indirizzo IPv4: nessun altro parametro potrà utilizzare tale valore per rappresentare un parametro diverso da IPv4 [24, p. 21]

Length è un *unsigned integer* di 16 bit che indica la dimensione in byte del parametro e segue gli stessi principi visti per il campo Length dei chunk

Value è un campo a dimensione variabile (nota al ricevente grazie al campo Length) contenente i dati veri e propri del parametro

2.2 Principali tipologie di chunk

Il valore presente nel campo *Type* di ogni Chunk ne identifica scopo, struttura e contenuto.

La tabella 2.1 elenca le tipologie di Chunk definite da IETF e i relativi Type, cosi come specificate nel RFC [24, p. 17-18]: tra i 256 Type possibili alcuni sono definiti dall'IETF e riservati al protocollo, altri sono riservati per future estensioni IETF e altri ancora sono disponibili per estensioni ed utilizzi personalizzati.

I valori Type sono definiti in modo tale che i due bit più significativi diano informazioni su come SCTP deve comportarsi nel caso in cui il Type non venga riconosciuto [24, p. 18]:

- 00 $\Rightarrow$ non processare il pacchetto e scartalo
- 01 $\Rightarrow$ non processare il pacchetto, scartalo e rispondi con un pacchetto SCTP contenente un Chunk ERROR con causa 'Unrecognized Chunk Type'
- 10 $\Rightarrow$ scarta il chunk e processa il resto del pacchetto
- 11 $\Rightarrow$ scarta il chunk, processa il resto del pacchetto e e rispondi con un pacchetto SCTP contenente un Chunk ERROR con causa 'Unrecognized Chunk Type'

Un principio analogo viene seguito per i Type dei parametri opzionali nei Chunk di controllo [24, p. 20].

Value	Abbreviation	Description
0	DATA	Payload data
1	INIT	Initiation
2	INIT ACK	Initiation acknowledgement
3	SACK	Selective acknowledgement
4	HEARTBEAT	Heartbeat request
5	HEARTBEAT ACK	Heartbeat acknowledgement
6	ABORT	Abort
7	SHUTDOWN	Shutdown
8	SHUTDOWN ACK	Shutdown acknowledgement
9	ERROR	Operation error
10	COOKIE ECHO	State cookie
11	COOKIE ACK	Cookie acknowledgement
12	ECNE	Explicit congestion notification echo (reserved)
13	CWR	Congestion window reduced (reserved)
14	SHUTDOWN COMPLETE	Shutdown complete
15-62	N/A	available
63	N/A	reserved for IETF-defined Chunk Extensions
64-126	N/A	available
127	N/A	reserved for IETF-defined Chunk Extensions
128-190	N/A	available
191	N/A	reserved for IETF-defined Chunk Extensions
192-254	N/A	available
255	N/A	reserved for IETF-defined Chunk Extensions
255	N/A	IETF-defined chunk extensions

Tabella 2.1: Valori Type e rispettivi chunk

In seguito verranno approfonditi i principali tipi di Chunk che hanno particolare rilevanza nel funzionamento del protocollo, rimandando alla consultazione del RFC [24] per le rimanenti tipologie.

2.2.1 Chunk DATA (Type = 0)

Il DATA Chunk (Type = 0) è utilizzato per il trasporto di messaggi utente o loro frammenti ed'è di fondamentale importanza per alcuni aspetti del protocollo come il multi-streaming e la consegna ordinata di chunk e messaggi utente.

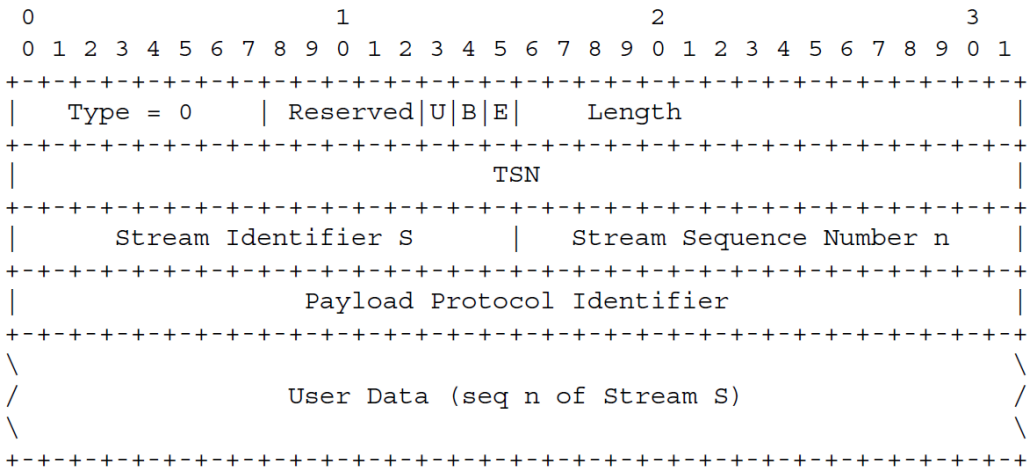

Figura 2.5: Struttura del chunk DATA (Type = 0)

La struttura del DATA Chunk, mostrata in figura fig. 2.5, è composta come segue:

Type impostato a 0 per identificare il Chunk Type DATA.

Flag Il campo flag è organizzato come segue:

Reserved i primi 5 bit del campo flag sono posti a 0 dal mittente e vengono ignorati dal ricevente.

U (Unordered) se posto a 1 indica che il Chunk non fa parte di uno stream ordinato, pertanto il campo Stream Sequence Number non ha significato e in ricezione il chunk deve essere consegnato immediatamente

al livello superiore senza alcun ordinamento. Se un messaggio utente non ordinato è frammentato tutti i suoi frammenti devono avere U posto a 1.

B (Beginning) se posto a 1 indica che il Chunk è il primo frammento di un messaggio utente.

E (Ending) se posto a 1 indica che il Chunk è l'ultimo frammento di un messaggio utente.

Length specifica la dimensione del chunk dal Type fino all'ultimo byte di User Data (escludendo quindi l'eventuale padding aggiunto al termine). Per esempio, un chunk DATA che trasporta un messaggio di un solo byte avrà Lenght 16 + 1 = 17 byte.

TSN (Transmission Sequence Number) è un *unsigned integer* di 32 bit che identifica il chunk consentendone la conferma di ricezione selettiva e quindi evitandone la ri-trasmissione.

Stream Identifier S è un *unsigned integer* di 16 bit che identifica lo stream al quale il chunk appartiene.

Stream Sequence Number n è un *unsigned integer* di 16 bit che identifica la posizione del frammento di dati utente inviato (User Data) all'interno dello stream di cui fa parte (identificato dallo Stream Identifier S). Se un messaggio utente è frammentato da SCTP i suoi chunk faranno riferimento allo stesso Stream Sequence Number n e avranno un ordine definito dal TSN.

Payload Protocol Identifier è un *unsigned integer* di 32 bit che identifica il protocollo applicativo di livello superiore al quale il pacchetto SCTP è diretto. Questo valore non è né utilizzato né alterato da SCTP che si limita a riceverlo dal livello superiore, trasportarlo e consegnarlo al livello superiore in ricezione. É però inserito in ogni chunk per permettere alle entità della rete e all'endpoint destinatario di identificare il tipo di dati che contiene.

User Data è un campo a dimensione variabile (noto grazie al campo Length) che contiene il payload di dati utente da trasferire. L'implementazione SCTP provvede a rendere la sua dimensione multipla di 4 byte aggiungendo un padding di 0.

2.2.2 Chunk INIT (Type = 1)

Il chunk INIT (Type = 1) è utilizzato per richiedere l'inizializzazione di una associazione SCTP tra due end-point. Da specifiche, un pacchetto SCTP che contiene un chunk INIT per essere considerato valido non deve contenere altri chunk.

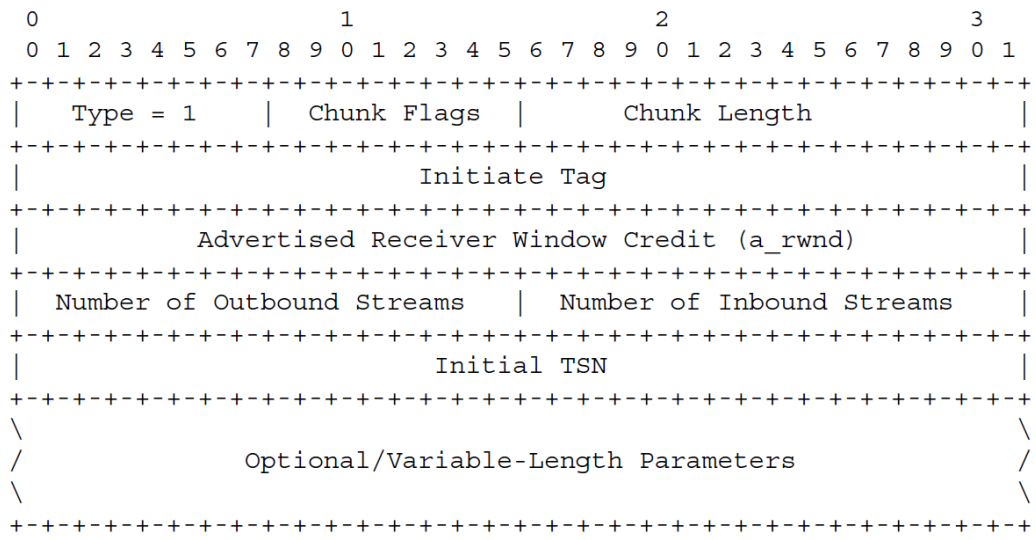

Figura 2.6: Struttura del chunk INIT (Type = 1)

La struttura del chunk DATA, mostrata in fig. 2.6, è composta come segue:

Type impostato a 1 per identificare il chunk come INIT.

Flags impostati a 0 dal mittente ed ignorati dal ricevente.

Initiate Tag è un *unsingned integer* di 32 bit diverso da 0 che l'end-point genera in modo random. Tale numero verrà inserito dall'end-point peer nel campo Verification Tag di ogni chunk DATA che invia nel corso dell'associazione

corrente. In questo modo l'end-point identifica l'associazione corrente e si evita che pacchetti appartenenti ad associazioni precedenti vengano interpretati come parte dell'associazione attuale. É importante che tale valore sia randomizzato per evitare attacchi di tipo *man-in-the-middle* e *sequence-number* e che, per tutta la durata dell'associazione, rimanga invariato [24, p. 72].

Advertised Receiver Window Credit (a_rwnd) è un *unsingned integer* di 32 bit attraverso il quale l'end-point comunica al destinatario la dimensione in byte del buffer che ha dedicato alla ricezione dei dati per l'associazione.

Number of Outbound Streams (OS) è un *unsingned integer* di 16 bit diverso da 0 attraverso il quale l'end-point comunica al suo peer quanti stream in uscita richiede di creare per l'associazione.

Number of Inbound Streams (MIS) è un *unsingned integer* di 16 bit diverso da 0 attraverso il quale l'end-point comunica al suo peer il massimo numero di stream in ingresso che consente di creare per l'associazione.

Initial TSN (I-TSN) è un *unsingned integer* di 32 bit attraverso il quale l'end-point comunica il Transmission Sequence Number (TSN) iniziale che userà per l'associazione.

Parametri opzionali del chunk INIT

Il chunk INIT prevede la possibilità di aggiungere, in seguito ai parametri obbligatori specificati sopra, diversi parametri opzionali a lunghezza variabile con formato Type-Length-Value (TLV).

Alcuni esempi sono i parametri **IPv4 (Type = 5)**, **IPv6 (Type = 6)** e **HostName Address (Type = 11)**, mostrati rispettivamente in fig. 2.7, fig. 2.8 e fig. 2.9.

Nel caso di multi-homing l'end-point può notificare i propri indirizzi IPv4 e IPv6 che vuole rendere parte dell'associazione (oltre all'indirizzo di provenienza del chunk INIT che ne fa parte di default). Per far ciò è possibile aggiungere uno o più parametri di tipo IPv4 (Type = 5), IPv6 (Type = 6) oppure un Host Name

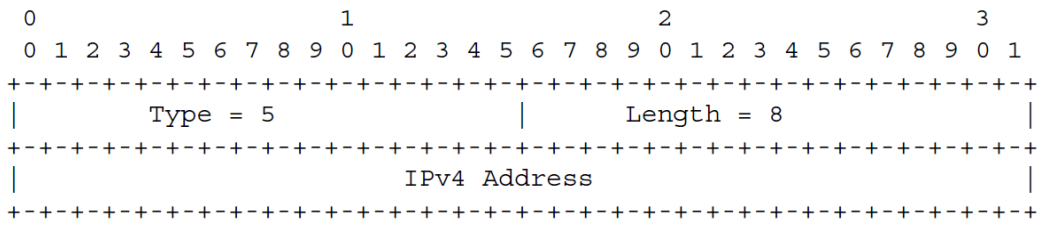

Figura 2.7: Parametro IPv4 (Type = 5) del chunk INIT

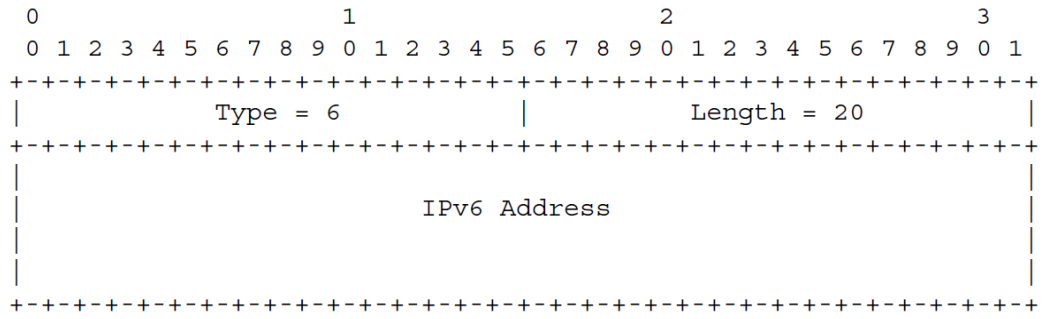

Figura 2.8: Parametro IPv6 (Type = 6) del chunk INIT

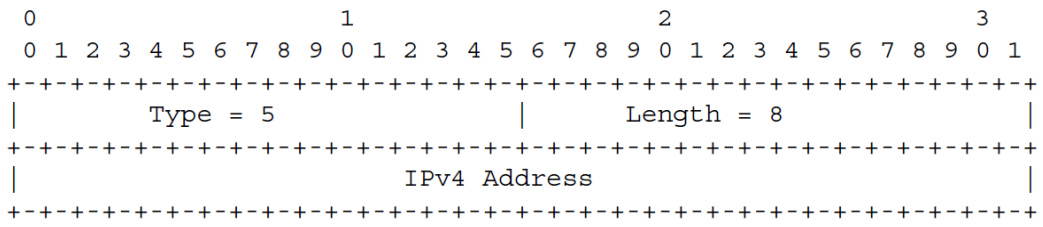

Figura 2.9: Parametro Host Name Address (Type = 11) del chunk INIT

Address (Type = 11). Nel caso del Host Name Address (Type = 11) l'end-point specifica solamente il nome di dominio ed'è compito dell'end-point che lo riceve risolverlo in indirizzi di rete e registrarli come parte dell'associazione.

Un altro parametro opzionale è il **Cookie Preservative (Type = 9)** che consente di incrementare la durata di validità del cookie di stato dell'associazione attraverso il campo *Suggested Cookie Life-Span Increment (msec.)*.

Altri parametri opzionali sono definiti dal RFC [24][p. 27-30], al quale si rimanda per approfondimenti.

2.2.3 Chunk INIT-ACK (Type = 2)

Il chunk INIT-ACK è utilizzato dal destinatario di un chunk INIT per confermarlo. Da specifiche, un pacchetto SCTP che contiene un chunk INIT-ACK per essere considerato valido non può contenere altri chunk.

La struttura del chunk INIT-ACK è speculare rispetto al chunk INIT pertanto è ridondante analizzarla. Una differenza importante rispetto a quest'ultimo è il parametro aggiuntivo **State Cookie (Type = 7)**. Tale parametro è obbligatorio in un chunk INIT-ACK e deve contenere:

- un timestamp del momento in cui è stato creato il cookie
- un tempo di vita specificato per il cookie
- le minime informazioni necessarie a ricreare il **Transmission Control Block (TCB)**, ovvero la struttura dati nella quale un end-point memorizza tutte le informazioni di stato e operative necessarie per gestire una determinata associazione
- un **Message Authentication Code (MAC)** che viene generato utilizzando le minime informazioni individuate per ricreare il TCB e una chiave privata segreta dell'end-point

Lo State Cookie è necessario per evitare lo stato *half-open* che espone al rischio di attacchi DoS SYN-flood: l'end-point che invia il chunk INIT-ACK infatti eliminerà tutte le strutture dati relative all'associazione subito dopo l'invio e sarà in grado di ricostruirle, previa verifica dell'integrità di sorgente e di contenuto utilizzando il MAC, grazie al cookie che verrà restituito dall'end-point peer nel chunk COOKIE-ECHO (vedi sezione 2.3.1 per maggiori dettagli).

2.2.4 Chunk SACK (Type = 3)

Il chunk SACK è utilizzato per:

1. confermare la ricezione della sequenza di chunk DATA correttamente e senza interruzioni fino ad un certo TSN, detto **Cumulative Ack TSN**

2. informare sulle sequenze di chunk ricevute correttamente successive al Cumulative Ack TSN (quindi che seguono una o più interruzioni della sequenza), dette **Gap Ack Blocks**. Ne fornisce il numero totale e per ciascuna di esse la posizione relativa di inizio e fine rispetto al Cumulative Ack TSN

3. informare sul numero di chunk duplicati ricevuti a partire dall'ultimo SACK inviato, e fornirne i TSN

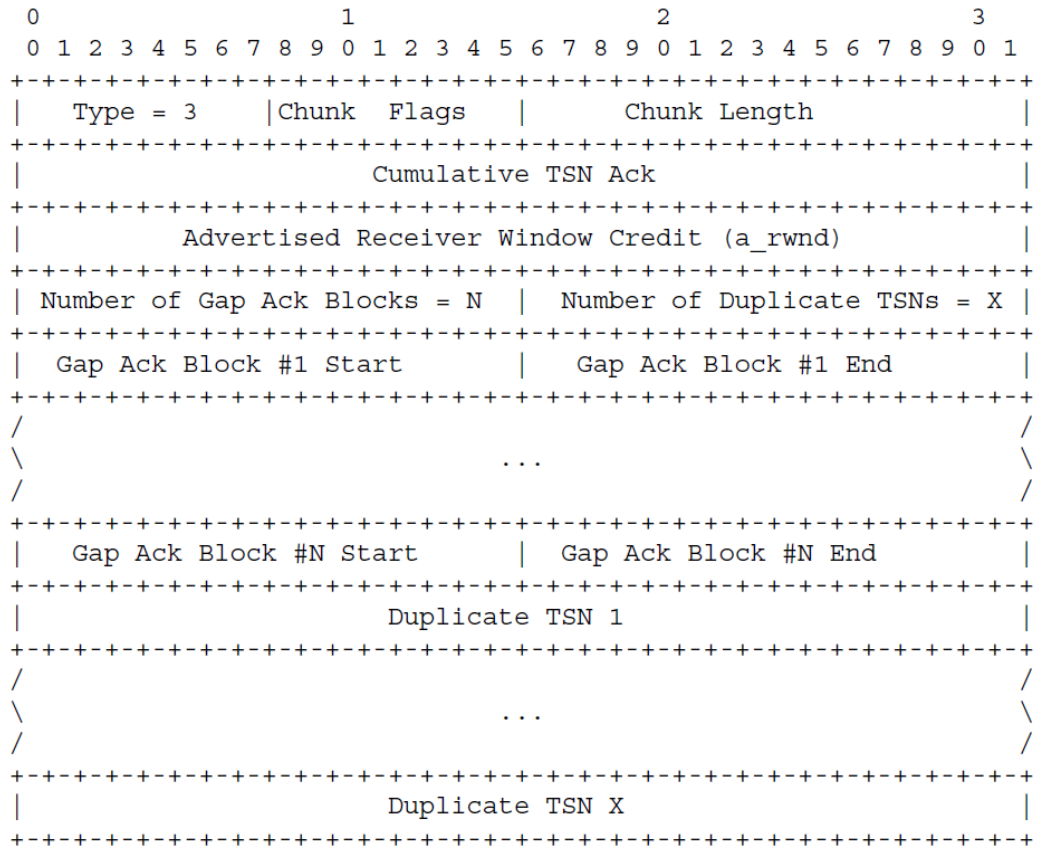

Figura 2.10: Struttura del chunk SACK (Type = 3)

La struttura del chunk SACK, mostrata in figura fig. 2.10, è composta come segue:

Type impostato a 3 per identificare il chunk come SACK

Flags impostati a 0 dal mittente ed ignorati dal ricevente

Cumulative TSN Ack è un *unsingned integer* di 32 bit che contiene l'ultimo TSN della sequenza di TSN ricevuti senza interruzioni. Serve a confermare che la sequenza di TSN è stata correttamente ricevuta fino a quel punto. Nella fase iniziale quando nessun TSN è ancora stato ricevuto viene impostato al valore Initial TSN - 1, dove Initial TSN è fornito dall'end-point peer durante la procedura di associazione e più precisamente nel chunk INIT

Advertised Receiver Window Credit (a_rwnd) è un *unsingned integer* di 32 bit attraverso il quale l'end-point comunica al destinatario la dimensione in byte del buffer che ha dedicato per la ricezione dei dati dell'associazione. Questo parametro, già presente nel chunk INIT, può essere ampliato durante la vita dell'associazione ma non ristretto [24, p. 27]

Number of Gap Ack Blocks è un *unsingned integer* di 16 bit che contiene il numero di sequenze di chunk ricevute correttamente ma successive al Cumulative TSN Ack (quindi dopo una o più interruzioni)

Number of Duplicate TSNs è un *unsingned integer* di 16 bit che contiene il numero di chunk (o più precisamente di TSN) duplicati ricevuti a partire dall'ultimo SACK inviato

In seguito a tali campi obbligatori, nel caso in cui siano presenti dei *Gap Ack Block*, per ognuno di essi vengono inseriti:

- la posizione relativa rispetto al Cumulative Ack TSN del primo TSN della sequenza ricevuto correttamente (**Gap Ack Block Start**)
- la posizione relativa rispetto al Cumulative Ack TSN dell'ultimo TSN della sequenza ricevuto correttamente (**Gap Ack Block End**)

Infine, se presenti, vengono inseriti tutti i **Duplicate TSN**, ovvero i TSN duplicati ricevuti successivamente all'invio dell'ultimo SACK.

Per maggiori dettagli sull'utilizzo del chunk SACK per le conferme selettive si veda sezione 2.6.2.

2.3 Associazioni SCTP

SCTP, come TCP, è orientato alla connessione. Ciò significa che, prima di poter comunicare tra loro, due end-point devono completare una procedura di *handshake* nella quale si comunicano i rispettivi parametri necessari ad instaurare un canale di comunicazione affidabile full-duplex su una rete inaffidabile *best-effort* come la rete IP.

La differenza sostanziale introdotta da SCTP è il *multi-homing*: il canale di comunicazione instaurato da SCTP viene instaurato tra una o più interfacce di rete di un end-point client e una o più interfacce di rete di un end-point server. Di conseguenza, non si tratta di una relazione uno-a-uno come in TCP ma piuttosto di una relazione molti-a-molti. Questo è il principale motivo per cui in SCTP si parla di *associazione* anziché di *connessione* come in TCP.

2.3.1 Inizializzazione: 4-way handshake

Il protocollo di handshake a 4 vie di SCTP tra un end-point client A e un end-point server B è riassunto in fig. 2.11.

Le fasi previste sono le seguenti:

1. **INIT**

   (a) il client invia un chunk INIT (vedi sezione 2.2.2) inserendo, tra le altre, anche le seguenti informazioni:

   - **Initiate Tag**: un numero intero positivo a 32 bit diverso da 0, generato in maniera random, con il quale il client identifica la associazione corrente: il server lo inserirà nel campo Verification Tag in tutti i successivi pacchetti dell'associazione
   - **Advertised Receiver Window Credit (a_rwnd)**: la dimensione in byte del priprio buffer di ricezione
   - **Number of Outbound Streams (OS)**: il numero di stream in uscita che richiede di creare

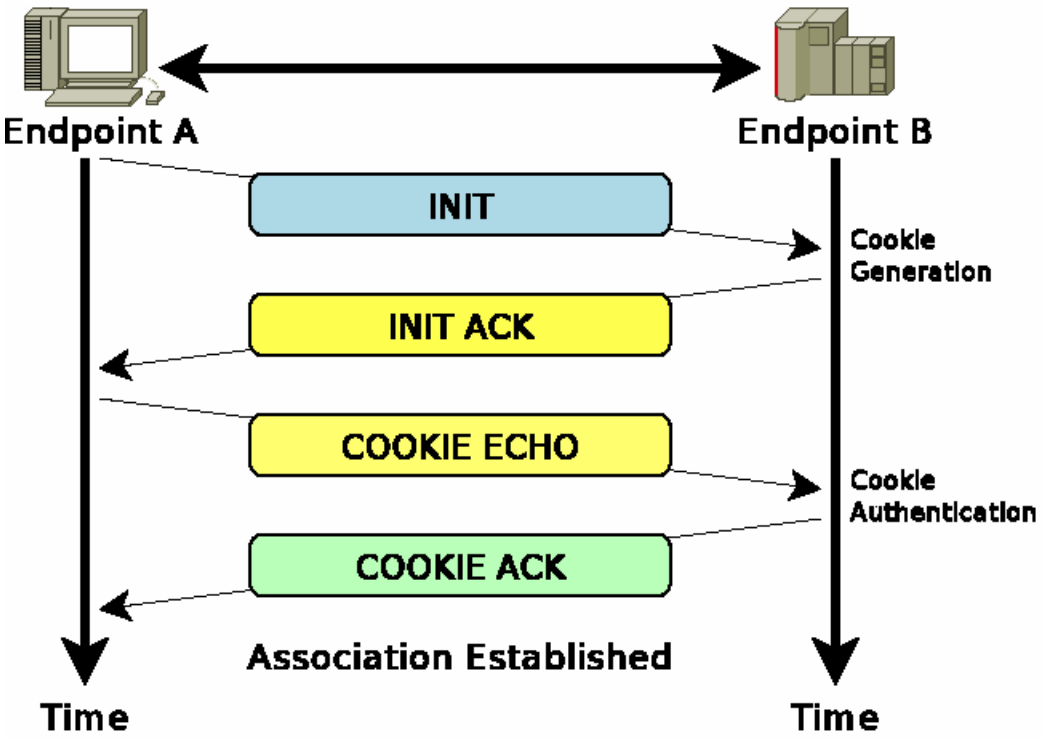

Figura 2.11: Inizializzazione dell'associazione: 4-way handshake

- **Number of Inbound Streams (MIS)**: il massimo numero di stream in ingresso che consente di creare
- eventuali indirizzi di rete aggiuntivi oltre al proprio nel caso si usufruisca del multi-homing

(b) fa partire un timer *T1-init*

(c) entra nello stato COOKIE-WAIT

2. **INIT-ACK**

(a) il server riceve il chunk INIT del client e risponde all'indirizzo sorgente di tale pacchetto con un chunk INIT-ACK (vedi sezione 2.2.3) speculare ad INIT, contenente le informazioni analoghe del server e alcuni parametri aggiuntivi:

- **Initiate Tag**: generato in modo analogo a quanto visto per il client, è un numero con il quale il server identifica la associazione corrente, che il client inserirà nel campo Verification Tag in tutti i successivi pacchetti dell'associazione
- **Verification Tag**: l'Initiate Tag del client
- **Advertised Receiver Window Credit (a_rwnd)**: la dimensione in byte del proprio buffer in ricezione
- **Number of Outbound Streams (OS)**: il numero di stream in uscita che richiede di creare
- **Number of Inbound Streams (MIS)**: il massimo numero di stream in ingresso che consente di creare
- eventuali indirizzi di rete aggiuntivi oltre al proprio nel caso si usufruisca del multi-homing
- **State Cookie**: cookie contenente le informazioni minime necessarie a ricreare il **Transmission Control Block (TCB)**[1], e un **Message Authentication Code (MAC)** che viene generato utilizzando le minime informazioni individuate per ricreare il TCB e una chiave privata segreta dell'end-point

(b) Libera tutte le risorse allocate per la connessione (evitando il rischio di attacchi DoS SYN-Flood, vedi sezione 2.3.1)

3. **COOKIE-ECHO**

(a) il client riceve il chunk INIT-ACK e ferma il timer *T1-init*

(b) esce dallo stato COOKIE-WAIT

(c) Invia un chunk COOKIE-ECHO al server nel quale inserisce il **Cookie** ricevuto nel campo State Cookie del chunk INIT-ACK. Il client può eventualmente aggiungere dei chunk DATA al pacchetto contenente il chunk COOKIE-ECHO, purché siano successivi ad esso e nello stesso

[1] struttura dati nella quale un end-point memorizza tutte le informazioni di stato e operative necessarie per gestire una determinata associazione

pacchetto. Non deve però inviare ulteriori pacchetti fino alla ricezione del chunk COOKIE-ACK del server

(d) fa partire un timer *T1-cookie*

(e) entra nello stato COOKIE-ECHOED

4. **COOKIE-ACK**

(a) il server riceve il chunk COOKIE-ECHO

(b) verifica l'integrità di sorgente e fonte del cookie utilizzando il MAC e ricostruisce il TCB dell'associazione

(c) entra nello stato ESTABLISHED: a questo punto per il server l'associazione è effettivamente stabilita

(d) invia un chunk COOKIE-ACK con il quale conferma l'avvenuta associazione al client. Tale chunk è vuoto e non contiene nessuna altra informazione. Altri chunk DATA o SACK possono essere inseriti nel pacchetto, purché successivi al COOKIE-ACK

(e) Alla ricezione del COOKIE-ACK, il client ferma il timer *T1-cookie* ed entra nello stato ESTABLISHED. A questo punto l'associazione è effettivamente stabilita in entrambi gli end-point

Si noti che è obbigatorio che i chunk INIT e INIT-ACK siano i soli chunk nel pacchetto di cui fanno parte: nelle rispettive fasi nessun altro dato viene scambiato tra gli end-point (vedi sezione 2.2.2 e sezione 2.2.3).

Protezione da attacchi DoS SYN-flood

Il 4-way handshake di SCTP risolve una nota vulnerabilità di sicurezza di TCP relativa al suo stato di connessione *half-open*: TCP infatti alloca le risorse e le strutture dati per la connessione alla ricezione del segmento SYN, rimanendo poi in attesa dell'ACK finale da parte del client. Se tale ACK non arriva il server rimane in attesa in uno stato half-open, riservando le risorse per tale connessione per un certo tempo.

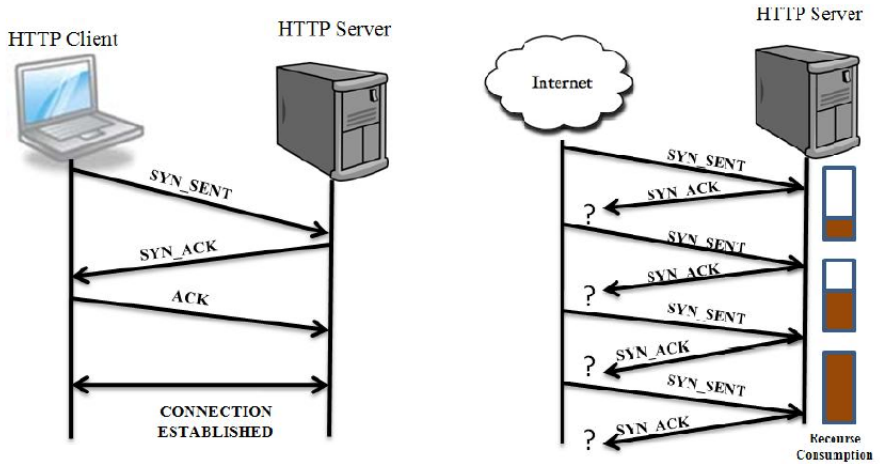

Figura 2.12: Vulnerabilità di TCP agli attacchi DoS SYN-flood

Nel caso degli attacchi SYN-flood della famiglia Denial Of Service (DoS) vengono inviate una quantità massiccia di richieste SYN, eventualmente da indirizzi IP sempre diversi e falsificati, senza mai terminare l'handshake con l'ACK finale. In questo modo il server continua ad allocare risorse per le connessioni half-open fino ad esaurimento, come mostrato in fig. 2.12.

SCTP risolve tale problema grazie allo **State Cookie** generato dal server nella fase di INIT-ACK: il server infatti include nel cookie le informazioni necessarie a ricreare il *Transmission Control Block (TCB)* dell'associazione[2]. Tale cookie viene inoltre autenticato con un MAC generato utilizzando una chiave segreta privata del server.

Il server, una volta generato lo State Cookie, lo inserisce nel chunk INIT-ACK inviato al client dopodiché libera tutte le risorse e le strutture dati allocate per l'associazione. In questo modo, in caso di eventuale attacchi SYN-flood, è protetto dal sovraccarico dato dall'allocazione delle risorse.

Se e solo se il client confermerà la associazione con il chunk COOKIE-ECHO il server allocherà le risorse necessarie. Sarà in grado di farlo utilizzando il cookie restituitogli dal client, la cui integrità di sorgente e contenuto è verificabile

[2]struttura dati contenente tutte le informazioni di stato e operative necessarie all'associazione

grazie al MAC inserito e il cui contenuto consente la ricreazione del TCB per l'associazione.

Come si può notare in SCTP non esiste lo stato *half-open* presente in TCP: la associazione viene effettivamente creata solamente quando entrambi gli end-point hanno inviato le rispettive conferme.

SCTP è quindi, al contrario di TCP, intrinsecamente protetto da attacchi DoS SYN-flood.

2.3.2 Terminazione della associazione

Una associazione SCTP può terminare per ABORT o per SHUTDOWN:

ABORT è una terminazione immediata con perdita dei dati rimasti in sospeso.

- Un endpoint che ha intenzione di abortire una associazione può inviare un chunk ABORT al suo endpoint peer. Il chunk ABORT deve essere l'unico presente nel pacchetto e deve contenere il Verification Tag del destinatario ed eventualmente una causa di errore (come per esempio 'User-Initiated Abort' nel caso l'aborto sia richiesto dal livello superiore)
- L'endpoint che riceve il chunk ABORT non deve rispondere a tale pacchetto. Una volta controllato il Verification Tag si limita a rimuovere le risorse allocate per l'associazione ed eventualmente notificare la causa di terminazione al livello superiore

SHUTDOWN prevede la consegna dei dati rimasti in sospeso prima della terminazione

- Il livello superiore, utilizzando la primitiva SHUTDOWN resa disponibile dal protocollo, può richiedere il *graceful shutdown* dell'associazione. Ricevuta la chiamata a tale primitiva dal livello superiore, l'endpoint entra in stato SHUTDOWN-PENDING e non accetta più

dati dal livello superiore. Si limita a inviare i dati rimasti in sospeso e ad attendere la loro conferma da parte dell'endpoint peer, eventualmente procedendo con la loro ri-trasmissione se necessario (come previsto nel normale scambio dati)

- Una volta ricevute le conferme di tutti i chunk inviati, l'endpoint invia un chunk SHUTDOWN inserendo nel campo Cumulative TSN Ack l'ultimo TSN sequenziale ricevuto dall'endpoint peer, avvia un timer *T2-shutdown* ed entra nello stato SHUTDOWN-SENT. Se il timer raggiunge un tempo prestabilito mentre l'endpoint è in tale stato quest'ultimo ripete l'invio del chunk SHUTDOWN, aggiornando il campo Cumulative TSN Ack con l'ultimo TSN ricevuto dal peer. Questo procedimento assicura, a differenza di TCP, che non esistano associazioni *half-open* dovute a chiusure dell'associazione non sincronizzate. Tuttavia la ritrasmissione del chunk SHUTDOWN viene effettuata un numero massimo di volte dopodiché SCTP segnala al livello superiore che l'endpoint peer non è raggiungibile
- Quando il server riceve il chunk SHUTDOWN entra nello stato SHUTDOWN-RECEIVED e smette di accettare nuovi dati. Tramite il campo Cumulative TSN Ack verifica che tutti i chunk inviati siano stati ricevuti dal client. Se risultano chunk non ricevuti procede con il loro reinvio finché non risultano tutti ricevuti
- Il client in stato SHUTDOWN-SENT che riceve ulteriori chunk DATA dal server ripete l'invio del chunk SHUTDOWN aggiornando il campo Cumulative TSN Ack con l'ultimo TSN ricevuto dal peer
- Quando il server verifica che tutti i chunk inviati al client sono stati ricevuti invia un chunk SHUTDOWN-ACK ed entra nello stato SHUTDOWN-ACK-SENT
- L'endpoint che riceve il chunk SHUTDOWN-ACK risponde con un chunk SHUTDOWN-COMPLETE e termina definitivamente l'associazione rimuovendo le risorse e le strutture dati allocate

- Il server che riceve lo SHUTDOWN-COMPLETE termina anch'esso la associazione e rimuove le strutture dati allocate (solamente se, per quella associazione, si trova nello stato SHUTDOWN-ACK-SENT, altrimenti il pacchetto viene ignorato) [24, p. 107-110]

A differenza di TCP, SCTP non supporta uno stato *half-open* nel quale un endpoint può continuare ad inviare dati mentre il suo peer è in stato CLOSED: quando in una associazione viene effettuato lo SHUTDOWN ciascuno dei due peer smetterà di accettare nuovi dati dal livello superiore e si limiterà ad inviare i dati rimasti in sospeso.

2.4 Frammentazione dei messaggi utente

2.4.1 Lunghezza massima del pacchetto SCTP

Un pacchetto SCTP può includere uno o più chunk fino al raggiungimento della dimensione massima del pacchetto, salvo alcune eccezioni (alcune tipologie di Chunk devono essere da specifiche gli unici chunk nel pacchetto) [24, p. 15].

In una associazione SCTP, in particolare in caso di multi-homing, possono esistere diversi percorsi tra le varie interfacce di rete coinvolte, pertanto possono esistere diversi MTU [3]. Il datagramma IP contenente il pacchetto SCTP deve avere dimensione totale minore o uguale alla ***Path Maximum Transmission Unit (P-MTU)*** **dell'associazione SCTP**, ovvero il minimo MTU esistente per tutti i percorsi possibili tra le interfacce della associazione. La dimensione massima del pacchetto SCTP deve conformarsi a tale limite [24, p. 91,100].

2.4.2 Frammentazione e ricostruzione dei messaggi utente

SCTP prevede che un messaggio di dati utente di uno stream possa essere frammentato in diversi Chunks di Type DATA qualora la sua dimensione ecceda

[3] Per *Maximum Transmission Unit (MTU)* 'Unità Massima di Trasmissione' si intende la dimensione massima del pacchetto di livello 3 (IP) trasmissibile senza sua frammentazione durante il percorso (per esempio, in caso di trasporto 1500 Byte se trasportato su Ethernet).

il limite imposto dal P-MTU dell'associazione SCTP.

Questa funzionalità non è garantita dal protocollo ma dalle singole implementazioni. Il RFC [24, p. 91-92] infatti stabilisce che:

- il riassemblamento di dati frammentati deve essere sempre garantito in ricezione [24, p. 75]
- le varie implementazioni possono decidere di supportare o meno la frammentazione dei dati in uscita
- Se una implementazione non supporta la frammentazione, nel caso in cui il pacchetto SCTP ecceda la dimensione consentita, il pacchetto deve essere scartato senza inviarlo restituendo un errore al livello superiore

La frammentazione di un messaggio utente e la sua ricostruzione avviene utilizzando i campi TSN, S, B e E del chunk DATA (vedi sezione 2.2.1):

- L'endpoint mittente frammenta il messaggio utente in DATA chunk di dimensione adeguata
- A tutti i chunk del messaggio viene assegnato lo stesso Stream Identifier S
- Ad ogni DATA chunk viene assegnato un TSN sequenziale che ne identifica l'ordine
- Il primo chunk del messaggio viene identificato impostanto il flag $B = 1$, l'ultimo impostando il flag $E = 1$, consentendo quindi al destinatario di riconoscere l'inizio e la fine del messaggio (vedi fig. 2.13)

2.5 Ordine

Nella trasmissione a commutazione di pacchetto IP ciascun pacchetto può seguire percorsi diversi in rete e giungere fuori sequenza in ricezione o andare perso.

B	E	Description
1	0	First piece of a fragmented user message
0	0	Middle piece of a fragmented user message
0	1	Last piece of a fragmented user message
1	1	Unfragmented message

Table 1: Fragment Description Flags

Figura 2.13: Flag B e E e loro significato

SCTP, analogamente a TCP, introduce meccanismi per rendere la trasmissione di dati ordinata e affidabile nonostante la natura best-effort della rete IP.

L'orientamento ai messaggi e la natura multi-streaming di SCTP rendono i suoi meccanismi di ordinamento inevitabilmente differenti e più complessi di TCP e consentono una maggiore flessibilità. In particolare SCTP prevede due livelli logici di ordinamento, con ruoli indipendenti e complementari:

1. Ordinamento della sequenza completa di chunk che prescinde dallo stream di appartenenza

2. Ordinamento dei messaggi relativamente al proprio stream di appartenenza

2.5.1 Ordine dei chunk (TSN)

Ogni end-point SCTP identifica tramite un numero sequenziale ogni chunk inviato nel corso di una associazione, a prescindere dallo stream di appartenenza. Tale numero identificativo viene inserito nel campo **Transmission Sequence Number (TSN)** del chunk DATA (vedi sezione 2.2.1).

Durante l'inizializzazione della associazione i due end-point dichiarano al peer il primo TSN che utilizzeranno nel campo **Initial TSN** dei chunk INIT e INIT-ACK (vedi sezione 2.2.2).

Il ruolo del TSN è per certi versi molto simile al *Sequence Number* di TCP, con la differenza che il TSN identifica la posizione relativa di un chunk nella sequenza di invio, anziché la posizione relativa nel flusso di byte come in TCP.

Grazie al TSN SCTP può:

- ricostruire la sequenza ordinata di chunk DATA in ricezione
- determinare eventuali mancate ricezioni e inviare conferme di ricezione selettive al peer tramite chunk SACK

2.5.2 Ordine dei messaggi (SID, SSN)

SCTP garantisce l'ordinamento dei messaggi relativamente allo stream di appartenenza. Inoltre, prevede la possibilità di contrassegnare uno o più messaggi come non ordinati (cioè da consegnare immediatamente al livello superiore).

Per ordinare i messaggi SCTP utilizza i seguenti campi del chunk DATA (sezione 2.2.1):

- **Stream Identifier S**: identifica lo stream al quale il chunk appartiene
- **Stream Sequence Number n**: identifica la posizione del frammento di dati utente inviato (User Data) all'interno dello stream di cui fa parte (identificato dallo Stream Identifier S). Se un messaggio utente è frammentato da SCTP i suoi chunk faranno riferimento allo stesso Stream Sequence Number n e avranno un ordine definito dal TSN
- **Unordered (U)**: se posto a 1 indica che il messaggio non fa parte di uno stream ordinato, pertanto il campo Stream Sequence Number non ha significato e in ricezione il messaggio deve essere consegnato immediatamente al livello superiore senza alcun ordinamento, non appena completo (se il messaggio è frammentato in più chunk essi saranno comunque ordinati grazie al campo TSN, ma il messaggio in sé non ha ordine rispetto ad altri messaggi dello stesso stream)

2.5.3 Ordinamento parziale e confronto con TCP

TCP prevede l'ordinamento totale di un unico flusso di byte utilizzando il *Sequence Number*. SCTP al contrario è orientato ai messaggi e suddivide lo-

gicamente i messaggi in stream, pertanto il suo ordinamento è più complesso e flessibile:

- L'ordine della intera sequenza di chunk della associazione è definito dai TSN
- Ogni messaggio è composto da uno o più chunk e appartiene a uno stream, identificato dallo Stream Identifier. I messaggi di uno stream possono essere:
  - ordinati rispetto agli altri messaggi dello stesso stream utilizzando lo Stream Sequence Number
  - non ordinati e consegnati immediatamente al livello superiore una volta ricevuti correttamente, se il flag U è impostato a 1

SCTP prevede quindi diverse tipologie di ordinamento:

- Ordinamento totale (un unico stream di messaggi ordinati)
- Ordinamento parziale (diversi stream, i cui messaggi sono ordinati relativamente ai messaggi dello stesso stream)
- Non ordinamento (contrassegnamento di uno o più messaggi come non ordinati, cioè da consegnare al livello superiore non appena completi)

In particolare, l'ordinamento parziale è una caratteristica strettamente correlata alla natura multi-streaming di SCTP e assume importanza fondamentale nel protocollo in quanto consente di superare i limiti di TCP riguardo al fenomeno di *HOL-blocking* (vedi sezione 1.2 e sezione 2.9.4).

2.6 Affidabilità

SCTP introduce diversi meccanismi per rendere il trasferimento dati affidabile nonostante la natura inaffidabile del protocollo IP. Molti dei meccanismi di affidabilità di SCTP derivano o evolvono concetti già presenti in TCP.

In particolare SCTP garantisce l'affidabilità combinando i seguenti meccanismi:

- Validazione del pacchetto (Checksum CRC32c e Verification Tag)
- Conferme selettive (SACK)
- Ritrasmissioni (RTO e Fast-retransmit)
- Failure-detection (Endpoint-failure-detection e Path-failure-detection)

2.6.1 Validazione del pacchetto (Checksum e Verification Tag)

Checksum CRC32c

Ogni pacchetto SCTP contiene nella Common Header (vedi sezione 2.1.1) un campo **Checksum** di 32 bit utilizzato per verificare in ricezione l'integrità dell'intero pacchetto (sia header che payload):

- il mittente prima di inviare il pacchetto imposta a 0 il campo, calcola il checksum dell'intero pacchetto mediante l'algoritmo *CRC32c* polinomiale per la rilevazione di errori ed inserisce il risultato nel campo lasciando inalterato il resto del pacchetto
- In ricezione viene salvato il valore Checksum, impostato a 0 il campo e calcolato nuovamente il Checksum mediante il medesimo algoritmo: se i due valori differiscono il pacchetto non è valido e viene ignorato

Il meccanismo di checksum utilizzato da SCTP è molto simile a quello già presente in TCP e svolge lo stesso compito, tuttavia SCTP adotta una dimensione maggiore (32 bit in SCTP, 16 bit in TCP) e un algoritmo differente (CRC32c).

Verification Tag

Ogni pacchetto SCTP contiene nella Common Header un campo Verification Tag (vedi sezione 2.1.1) che serve per identificare la associazione attuale ed evitare

che pacchetti appartenenti a sessioni precedenti vengano interpretati come parte dell'associazione attuale:

- Durante l'inizializzazione della associazione ciascun end-point genera un valore random diverso da 0 e lo inserisce nel campo Initiate Tag del chunk INIT (o INIT-ACK) per fornirlo all'end-point peer (vedi sezione 2.2.2 e sezione 2.3.1)
- Durante la vita dell'associazione il numero fornito dall'end-point peer rimane invariato e viene inserito nel campo Verification Tag di ogni chunk DATA inviatogli
- L'end-point che riceve un pacchetto SCTP verifica tramite tale valore se il pacchetto appartiene o meno alla associazione corrente. Se il Verification Tag non corrisponde il pacchetto viene scartato automaticamente

2.6.2 Conferme selettive (SACK)

SCTP prevede la conferma di ricezione selettiva e la ritrasmissione dei soli chunk mancanti attraverso il chunk SACK (vedi sezione 2.2.4), che contiene tutte le informazioni necessarie allo scopo:

- **Cumulative Ack TSN**: ultimo TSN della sequenza di chunk ricevuta correttamente, completa e senza interruzioni
- **Gap Ack Blocks**: sequenze di chunk ordinate ricevute correttamente ma successive al Cumulative Ack TSN (quindi dopo una o più interruzioni della sequenza). Ne viene fornito il numero totale **Number of Gap Ack Blocks** e per ognuna di esse viene fornita la posizione di inizio (**Gap Ack Block Start**) e di fine (**Gap Ack Block End**) relativamente al Cumulative Ack TSN
- **Duplicate TSNs**: sono i TSN duplicati ricevuti a partire dall'invio dell'ultimo SACK, che possono essere indice di una mancata ricezione del SACK da parte dell'end-point peer. Ne viene fornito il numero totale nel campo

Number of Duplicate TSNs e per ognuno di essi ne viene fornito il TSN nei campi **Duplicate TSN**

Un esempio del funzionamento del Duplicate TSN potrebbe essere il seguente: se il TSN=19 fosse stato ricevuto tre volte (ovvero con due duplicati), l'end-point invierebbe un chunk SACK contenente due Duplicate TSN=19 e azzererebbe la lista dei duplicati. Se venisse ricevuto nuovamente un chunk con TSN=19, l'end-point inserirebbe nel successivo chunk SACK un solo Duplicate TSN=19.

Un esempio di sequenza di chunk con interruzioni e relativi parametri SACK è mostrata in fig. 2.14.

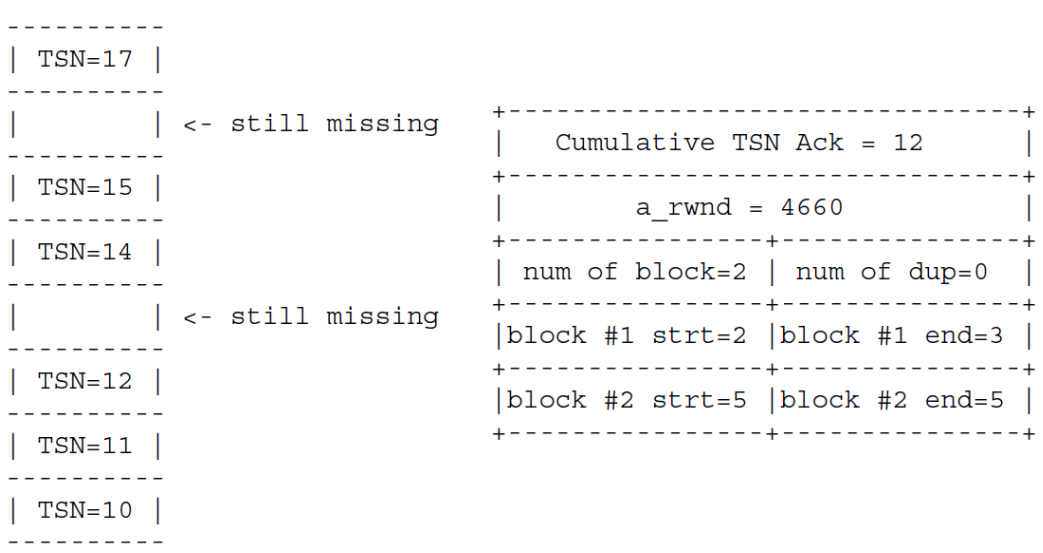

Figura 2.14: Esempio di utilizzo SACK

I meccanismi descritti derivano dai concetti già presenti in TCP di ACK cumulativo (comportamento di default) e Selective Acknowledgement (modifica specificata nel RFC 2018 [20]). In SCTP tuttavia le conferme e le ritrasmissioni selettive sono previste come comportamento di default.

I chunk SACK, oltre a garantire la ritrasmissione selettiva, sono strettamente correlati alla stima del *Round Trip Time (RTT)*, al conseguente calcolo del *Retransmission Timeout (RTO)* (vedi sezione 2.6.3) e ai controlli di flusso e di congestione (vedi sezione 2.7).

Si noti che i chunk SACK non devono essere per forza inviati immediatamente alla ricezione dei chunk DATA e neanche ad ogni pacchetto SCTP ricevuto. Gli implementatori possono decidere in autonomia quanto frequentemente e con quanto ritardo dalla ricezione inviare SACK di conferma, purché rispettino alcune specifiche [24, p. 77-79]:

- un SACK deve essere inviato almeno ogni 2 pacchetti SCTP ricevuti e con un ritardo di al massimo 500ms dalla ricezione
- alla ricezione di un pacchetto SCTP con soli chunk DATA duplicati è obbligatorio inviare immediatamente un SACK (ciò normalmente accade quando uno o più SACK inviati vengono persi con conseguente timeout RTO e ritrasmissione)
- se il pacchetto contiene anche chunk DATA nuovi oltre che duplicati è comunque consigliato l'invio immediato di un SACK

2.6.3 Ritrasmissioni (RTO e Fast-retransmit)

I meccanismi che SCTP utilizza per avviare la ritrasmissione dei chunk mancanti sono i seguenti:

- **Retransmission timeout (RTO)**: il timeout si verifica allo scadere del timer T3-rtx che pone un limite massimo al tempo necessario per la conferma dei chunk in-flight. Allo scadere del timer la mancata conferma viene interpretata come perdita dei chunk non ancora confermati, i quali vengono ritrasmessi appena possibile
- **Fast Retransmit on Gap Reports**: quando uno o più chunk vengono segnalati come mancanti da 3 SACK consecutivi SCTP li interpreta come persi e li ritrasmette appena possibile, senza aspettare la scadenza del timer T3-rtx (timeout RTO)

Retransmission Timeout (RTO) e timer T3-rtx

Come TCP, anche SCTP utilizza un timer come meccanismo principale per garantire la trasmissione affidabile di pacchetti su un canale inaffidabile. Superato il tempo limite imposto dal timer viene avviata la ritrasmissione dei pacchetti non confermati, assicurando quindi la consegna in caso di assenza di feedback da parte dell'end-point peer.

In SCTP il timer di ritrasmissione viene chiamato **T3-rtx** e il rispettivo evento di timeout che innesca la ritrasmissione è detto **Retransmission Timeout (RTO)**.

In caso di multi-homing viene mantenuto un timer T3-rtx con rispettivo RTO per ognuno dei path dell'associazione.

Il calcolo del RTO avviene in modo simile a TCP sulla base del **Round Trip Time (RTT)** misurato tra l'invio di un pacchetto e la ricezione della rispettiva conferma. Per effettuare tale calcolo vengono mantenute dagli end-point due variabili:

- **SRTT (smoothed round-trip time)**
- **RTTVAR (round-trip time variation)**

Il calcolo del RTO avviene secondo le seguenti regole [24, p. 83-84]:

- Finché il RTT non è stato misurato, il RTO viene impostato al valore *RTO.Initial* (raccomandato 3 secondi)
- Alla prima misurazione R del RTT si impostano:
  - SRTT = R
  - RTTVAR = R / 2
  - RTO = SRTT + 4 $\times$ RTTVAR
- Alle successive misurazioni R del RTT, con R' la misurazione precedente e valori raccomandati RTO.Alpha = 1/8 e RTO.Beta = 1/4, si impostano:
  - RTTVAR = (1 - RTO.Beta) $\times$ RTTVAR + RTO.Beta $\times$ $|$SRTT - R'$|$

  - SRTT = (1 - RTO.Alpha) $\times$ SRTT + RTO.Alpha $\times$ R'
  - RTO = SRTT + 4 $\times$ RTTVAR

- Come avviene anche in TCP, la misurazione del RTT per ogni destinazione viene effettuata non più di una volta per ogni *round-trip*, scegliendo per la misurazione un pacchetto in uscita che non sia stato ritrasmesso

- Per prevenire timeout non necessari o tempi di ritrasmissione troppo lunghi vengono definiti rispettivamente un limite minimo **RTO.Min** (raccomandato 1 secondo), ed opzionalmente un limite massimo **RTO.Max** (raccomandato 60 secondi). Il calcolo del RTO deve in tal caso rispettare le seguenti condizioni:

  - RTO = $max(\text{RTO}, \text{RTO.Min})$
  - RTO = $min(\text{RTO}, \text{RTO.Max})$

Il timer T3-rtx per un dato path viene gestito secondo le seguenti regole [24, p. 85]:

- Se il timer non è avviato, viene avviato:

  - al primo chunk DATA inviato in quel path (anche nel caso si tratti di una ristrasmissione)
  - alla ricezione di un SACK il cui Cumulative TSN Ack indica uno o più TSN mancanti nella sequenza

- Il timer viene fermato se alla ricezione di un chunk SACK tutti i chunk in attesa di conferma sono confermati

- Il timer viene riavviato se alla ricezione di un chunk SACK il minore TSN in-flight viene confermato ma restano dei chunk in attesa di conferma

Alla scadenza del timer T3-rtx si verifica un evento di **Retransmission TimeOut (RTO)**, gestito come segue [24, p.86 87]:

- I parametri *ssthresh* e *cwnd* del path nel quale il timer è scaduto vengono aggiornati secondo le regole previste dal controllo di congestione (vedi sezione 2.7.2)
- Il timer RTO del path nel quale il timer è scaduto viene raddoppiato (**exponential back-off mechanism**), con eventualmente RTO.max come limite superiore (consigliato 60 secondi). Alle successive misurazioni del RTT il RTO verrà nuovamente calcolato secondo le regole viste sopra
- Viene avviata la ritrasmissione di tutti i chunk in-flight nel path in questione al momento dello scadere del timer:
  - tale ritrasmissione in caso di multi-homing viene effettuata in un path alternativo, se disponibile. I timer già avviati dei path non vengono fermati
  - a seconda dei parametri *P-MTU* (vedi sezione 2.4.1) e *cwnd* (vedi sezione 2.7) del path scelto, vengono inviati i suddetti chunk in uno o più pacchetti non appena possibile
  - Se il timer T3-rtx nel nuovo path scelto è inattivo, viene fatto partire utilizzando il rispettivo RTO. Se invece il path scelto non è cambiato (per esempio, se è l'unico disponibile) e il timer è inattivo, viene fatto partire solo dopo aver raddoppiato il tempo RTO

2.6.4 Path Heartbeat

Durante la vita dell'associazione (stati COOKIE-ECHOED e/o ESTABLISHED), l'endpoint monitora periodicamente tutti gli indirizzi dell'end-point peer inviando dei chunk HEARTBEAT la cui ricezione deve essere confermata con dei chunk HEARTBEAT-ACK.

Tutti i path vengono verificati durante l'intera vita dell'associazione, anche i path inattivi a causa di timeout ripetuti e mancate conferme.

Il livello superiore può però disabilitare esplicitamente l'heartbeat su uno o più path tramite apposite primitive.

Chunk HEARTBEAT e HEARTBEAT-ACK

I chunk HEARTBEAT (Type = 4) contengono, come parametri opzionali nel proprio campo Value, un timestamp che identifica il momento dell'invio e l'indirizzo destinazione che si vuole verificare.

I chunk di conferma HEARTBEAT-ACK (Type = 5) contengono invece una copia delle informazioni ricevute, lasciata inalterata.

Si noti che il timestamp inviato dal client viene restituito nuovamente nel chunk HEARTBEAT-ACK dal server: utilizzando questo valore l'endpoint è in grado di stimare il RTT del path e aggiornare il RTO.

I chunk HEARTBEAT e HEARTBEAT-ACK hanno quindi una duplice funzione:

- monitoraggio dello stato di raggiungibilità dell'end-point peer in un determinato path, marcandolo come inattivo in caso di multiple mancate conferme (vedi sezione 2.6.5)
- aggiornamento del RTT e del RTO per tale path in condizioni di normalità (vedi sezione 2.6.3)

Periodo di Heartbeat

L'intervallo periodico di invio di chunk HEARTBEAT è specifico per ogni path: il RFC [24, p. 103] raccomanda di aggiungere un parametro *HB.interval* al RTO calcolato per quel path (vedi sezione 2.6.3), applicando un jittering del $\pm 50\%$ del RTO e con *exponential backoff* in caso di mancata ricezione della conferma.

Il periodo di Heartbeat *HB.interval* è definito dalle varie implementazioni che possono anche scegliere di renderlo modificabile da parte del livello superiore.

Possono essere consentite al livello superiore anche l'attivazione/disattivazione di heartbeat verso uno specifico indirizzo e l'invio di HEARTBEAT on-demand. Per tutte queste funzioni sono disponibile delle primitive apposite definite dal protocollo [24, p. 102].

Nell'ottimizzazione del periodo di heartbeat o nella sua disattivazione è bene considerare il suo impatto:

- Aumentando l'intervallo di heartbeat aumenta anche il tempo necessario a individuare la non-raggiungibilità dell'end-point peer
- Riducendo troppo l'intervallo di heartbeat si rischia di interpretare dei ritardi fisiologici come malfunzionamento del path
- In assenza di heartbeat in un path (che è presente di default ma può essere disattivato dal livello superiore chiamando una apposita primitiva), gli errori verranno riscontrati solamente tramite l'invio di chunk DATA (ciò può causare problemi, per esempio nel caso di un chunk ABORT andato perso che causa il fallimento della chiusura dell'associazione)

Il meccanismo heartbeat di SCTP deriva dal keep-alive di TCP che però è prevista come funzionalità opzionale. SCTP prevede tale meccanismo di default in modo da rilevare il prima possibile eventuali non-raggiungibilità e garantire, con l'integrazione del multi-homing (vedi sezione 2.8), l'alta disponibilità richiesta dai contesti applicativi di SCTP (vedi sezione 1.2).

Rilevazione di path inattivi

I chunk HEARTBEAT non confermati dall'indirizzo destinatario incrementano il contatore di errori del path in questione e, al raggiungimento del valore limite *Path.Max.Retrans* il path viene considerato inattivo.

La trasmissione di HEARTBEAT periodica continua anche per i path inattivi, solamente non viene incrementato il contatore di errori in caso di mancata risposta [24, p. 102].

Alla ricezione di un HEARTBEAT-ACK il path in questione viene considerato attivo e il relativo contatore di errori viene azzerato.

2.6.5 Failure detection

SCTP prevede meccanismi di monitoraggio e rilevazione rapida degli errori dello stato di connessione dell'intera associazione e dei singoli path. Ciò assicura una gestione rapida in caso di non-raggiungibilità con cambio di path primario non appena venga rilevata la sua inattività.

Endpoint Failure Detection prevede la chiusura di una associazione al raggiungimento di un certo numero di ritrasmissioni consecutive non confermate

- Ogni endpoint in SCTP mantiene un contatore del numero totale di ritrasmissioni consecutive non confermate verso l'endpoint associato. Tale conteggio comprende tutte le interfacce in caso di multi-homing, quindi un unico contatore per l'intera associazione. Nel conteggio sono considerati anche le ritrasmissioni non confermate di chunk HEARTBEAT
- Il contatore viene resettato ogni volta che viene ricevuto una conferma (SACK o HEARTBEAT-ACK) da parte dell'endpoint peer
- Se il valore del contatore eccede il parametro del protocollo *Association.Max.Retrans* l'endpoint deve considerare il suo peer irraggiungibile, terminare l'invio di pacchetti verso di esso e impostare la associazione allo CLOSED [24, p. 100]

Path Failure Detection prevede la chiusura di un singolo path al raggiungimento di un certo numero di ritrasmissioni di chunk DATA o mancate conferme di chunk HEARTBEAT

- un endpoint mantiene un contatore di ritrasmissioni per ognuno degli indirizzi di rete dell'endpoint destinazione coinvolti nell'associazione, quindi un contatore per ogni path dell'associazione in caso di multi-homing. Tale conteggio considera anche le mancate conferme degli HEARTBEAT inviati
- Il reset del contatore viene effettuato alla ricezione della conferma di un TSN non ancora confermato o di un HEARTBEAT-ACK

- Se il contatore eccede il parametro del protocollo *Path.Max.Retrans* l'endpoint deve considerare il path in questione inattivo e notificare il livello superiore [24, p. 101]
- Se il path primario dell'associazione diventa inattivo l'endpoint può eleggere come primario un path alternativo verso un'altra interfaccia di rete destinazione, se esiste ed'è attivo. Se sono presenti più path alternativi possibili ne deve essere scelto e utilizzato uno per volta

2.7 Controllo di flusso e della congestione

I controlli di flusso e della congestione di SCTP si basano sugli stessi meccanismi di TCP [11] opportunamente adattati.

2.7.1 Controllo di flusso

Il controllo di flusso ha come obiettivo prevenire la perdita di pacchetti dovuta a overflow dei buffer di ricezione dei destinatari.

In modo identico a TCP, SCTP ottiene il controllo di flusso grazie alla variabile **Receiver advertised window size (rwnd)**. Tale variabile viene mantenuta da ogni end-point per ogni associazione SCTP e rappresenta il numero di byte disponibili nel buffer di ricezione: viene comunicato all'end-point peer che lo utilizza come limite superiore di byte *in-flight* (inviati e non ancora confermati tramite SACK).

Inizialmente il valore di rwnd viene scambiato tra gli end-point attraverso il chunk INIT e INIT-ACK, dopodiché viene aggiornato durante la vita dell'associazione attraverso i chunk SACK (vedi sezione 2.2.4).

2.7.2 Controllo della congestione

Il controllo della congestione ha come obiettivo la prevenzione della perdita di pacchetti dovuta alla saturazione della capacità dei router nella rete.

SCTP segue lo stesso approccio *non-greedy* di TCP, ovvero la riduzione del *rate* di invio alla rilevazione di eventi interpretati come perdita dei pacchetti:

- Retransmission Timeout (RTO)
- 3 SACK consecutivi che segnalano come mancanti gli stessi TSN

L'assunzione di base è che gli eventi di perdita dei pacchetti siano indice di congestione di rete.

Come avviene in TCP, anche in SCTP gli endpoint mantengono una serie di variabili per stimare il livello di congestione di rete e adattare opportunamente il rate di invio:

- **Congestion control window (cwnd)**: valore in byte utilizzato per regolare il rate di invio: ogni end-point può avere al più $min(\text{cwnd}, \text{rwnd})$ byte in-flight, ovvero inviati ma non ancora confermati tramite SACK. cwnd viene aumentato ad ogni round-trip e diminuito in caso di eventi di perdita (sintomo di congestione)
- **Slow-start threshold (ssthresh)**: valore in byte che determina la soglia di passaggio dalla fase di *slow-start* (incremento del rate di invio esponenziale) a quella di *congestion-avoidance* (incremento del rate di invio lineare) per un determinato path
- **Partial Bytes Acked (partial_bytes_acked)**: utilizzata per facilitare l'aggiustamento della cwnd durante la fase di congestion-avoidance, tiene traccia della quantità di byte confermati in ricezione a partire dall'ultimo incremento di cwnd

A differenza di TCP, SCTP supporta il multi-homing pertanto gli end-point SCTP devono mantenere la tripletta di variabili *(cwnd, ssthresh e partial_bytes_acked)* per ognuna delle interfacce destinazione dell'associazione (a differenza della variabile rwnd che è unica per ogni end-point). Si noti che delle variabili viste partial_bytes_acked è l'unica non presente anche in TCP.

In estrema sintesi, l'approccio utilizzato da TCP ed ereditato da SCTP prevede a regime l'aumento additivo del rate di invio in assenza di eventi di perdita e il decremento moltiplicativo in caso di perdita di pacchetti (*Additive Increse Multiplicative Decrease (AIMD)*) [11].

Più precisamente il controllo di congestione SCTP si compone di tre fasi, che verranno analizzate nel dettaglio nel seguito del capitolo:

- **Slow Start** a inizio trasmissione o dopo eventi di timeout RTO, SCTP sonda la capacità di rete con crescita del rate di invio esponenziale ad ogni round-trip.
- **Congestion avoidance** quando la finestra di congestione aumenta oltre la soglia sstrhesh, la crescita del rate di invio diventa lineare. SCTP esce da questa fase in caso di chunk segnalati come mancanti da 3 SACK consecutivi, passando alla fase di Fast Recovery, oppure in seguito a timer RTO, tornando alla fase di slow-start.
- **Fast Recovery** In seguito a chunk segnalati come mancanti da 3 SACK consecutivi SCTP imposta la soglia ssthresh a metà della finestra di congestione e continua con lo stesso incremento lineare previsto nella fase congestion-avoidance, ma evitando che eventuali nuovi eventi di fast-recovery riducano ulteriormente cwnd e sstresh. SCTP esce da questa fase alla completa ricezione dei chunk mancanti, tornando alla fase di congestion-avoidance, o in seguito a timer RTO, tornando alla fase di slow-start.

Slow Start

L'algoritmo di slow-start viene utilizzato per sondare la capacità della rete ad inizio trasmissione o in caso di *Retransmission Timeout (RTO)* (vedi sezione 2.6.3).

A inizio trasmissione, o in seguito ad un lungo periodo di inattività, la finestra di congestione viene inizializzata come segue:

$$\text{cwnd} = min(4 \times \text{P-MTU}, max(2 \times \text{P-MTU}, 4380 \text{ bytes}))$$

Per P-MTU si intende la *Path-Maximum-Transmission-Unit* (vedi sezione 2.4.1).

La fase di slow-start prevede che alla ricezione di ogni SACK che incrementi il numero di Cumulative TSN Ack (vedi sezione 2.6.2), la finestra di congestione

cwnd sia incrementata come segue (dove per #byteAcked si intende la somma dei byte di tutti i chunk DATA confermati dal SACK, compresi i Gap Block):

$$\text{cwnd} = min(\text{P-MTU}, \text{\#byteAcked})$$

Più precisamente il RFC [24, p. 96] specifica che, per motivi di sicurezza, tale incremento della cwnd deve avvenire solo se tutte le seguenti condizioni sono vere (altrimenti va ignorato):

1. La cwnd corrente è completamente utilizzata
2. Il SACK avanza il punto di Cumulative TSN Ack
3. Il mittente non è in Fast-recovery

Analogamente a quanto avviene in TCP, tale incremento della cwnd equivale ad un raddoppiamento del rate di invio ad ogni *round-trip*, con conseguente crescita esponenziale.

A inizio trasmissione ssthresh può essere impostato alto a piacere, per esempio uguale alla dimensione della Receiver advertised window size (rwnd) [24, p. 96]. In caso di eventi di perdita come timeout RTO o chunk segnalati come mancanti da 3 SACK consecutivi, ssthresh viene aggiornata:

$$\text{ssthresh} = max(\text{cwnd}/2, 4 \times \text{P-MTU})$$

Lo stesso aggiornamento di ssthresh viene attuato ogni RTO in caso il mittente non stia più inviando chunk DATA.

L'aggiornamento della cwnd è invece diverso in base all'evento di perdita che lo causa:

- In seguito ad un timeout RTO è prevista l'entrata nella fase di slow-start dopo una drastica riduzione della finestra di congestione:

$$\text{cwnd} = \text{P-MTU}$$

- Se l'evento di perdita è dovuto a chunk segnalati come mancanti da 3 SACK consecutivi è previsto il passaggio alla fase di Fast Recovery dopo un aggiornamento della finestra di congestione meno drastico:

$$\text{cwnd} = \text{ssthresh}$$

In assenza di eventi di perdita, il rate di invio cresce fino al raggiungimento della soglia ssthresh: a questo punto SCTP entra nella seguente fase di *congestion-avoidance*.

Congestion avoidance

In questa fase SCTP incrementa la Congestion control window (cwnd) linearmente di un P-MTU (vedi sezione 2.4.1) ogni *round-trip*. Esistono diversi modi per raggiungere tale obiettivo. Lo standard SCTP raccomanda un proprio algoritmo che fa uso della variabile **partial_bytes_acked**:

- Inizialmente:

$$\text{partial\_bytes\_acked} = 0$$

- Nella fase di congestion-avoidance (cwnd $>$ ssthresh), ogni volta che viene ricevuto un SACK che incrementa il Cumulative TSN Ack, partial_bytes_acked viene aggiornato come segue (dove per #byteAcked si intende la somma dei byte di tutti i chunk DATA confermati dal SACK, compresi i Gap Block):

$$\text{partial\_bytes\_acked} \mathrel{+}= \#byteAcked$$

- Se partial_bytes_acked $\geq$ cwnd e i byte-in-flight $\geq$ cwnd:

$$\text{cwnd} \mathrel{+}= \text{P-MTU}$$

$$\text{partial\_bytes\_acked} \mathrel{-}= \text{cwnd}$$

- Quando tutti i dati trasmessi sono stati confermati:

$$\text{partial\_bytes\_acked} = 0$$

- Allo stesso modo di quanto avviene in slow-start, quando il mittente non invia chunk DATA in un path, la sua cwnd viene aggiornata ogni RTO:

$$\text{ssthresh} = max(\text{cwnd}/2, 4 \times \text{MTU})$$

In sintesi, durante la fase di congestion-avoidance, SCTP incrementa la cwnd di 1 P-MTU alla ricezione di un SACK che incrementi il Cumulative Ack TSN a condizione che i partial_bytes_acked e i byte in-flight siano maggiori o uguali a cwnd. Ciò equivale ad un incremento lineare del rate di invio (1 P-MTU ogni round-trip).

La fase di congestion-avoidance continua con un incremento lineare della cwnd fino ad un evento di perdita, interpretato come congestione della rete.

Quando si verificano eventi di perdita come timeout RTO o chunk segnalati come mancanti da 3 SACK consecutivi (o nel caso il mittente non stia più inviando chunk DATA nel path), ssthresh viene aggiornata allo stesso modo visto nella fase di slow-start:

$$\text{ssthresh} = max(\text{cwnd}/2, 4 \times \text{P-MTU})$$

In caso di timeout RTO è prevista la ripartenza della fase di slow-start dopo una drastica riduzione della finestra di congestione:

$$\text{cwnd} = 1\text{P-MTU}$$

Se invece l'evento di perdita è dovuto a chunk segnalati come mancanti da 3 SACK consecutivi è previsto il passaggio alla fase di Fast Recovery dopo un aggiornamento della finestra di congestione meno drastico:

$$\text{cwnd} = \text{ssthresh}$$

Fast Retransmit on Gap Reports e Fast Recovery

In modo simile a quanto avviene in TCP Reno, SCTP prevede la ritrasmissione rapida (cioè senza aspettare la scadenza del timeout RTO) di tutti i chunk che vengono segnalati come mancanti da 3 SACK consecutivi.

In seguito all'avviamento della ritrasmissione rapida, SCTP aggiorna le variabili come segue:

$$\text{ssthresh} = max(\text{cwnd}/2, 4 \times \text{MTU})$$

$$\text{cwnd} = \text{ssthresh}$$

$$\text{partial\_bytes\_acked} = 0$$

SCTP entra quindi nella fase di **fast-recovery** e continua con lo stesso incremento lineare previsto nella fase congestion-avoidance ma evitando che eventuali nuovi eventi di ritrasmissione rapida riducano ulteriormente cwnd e sshtresh.

SCTP esce dalla fase di fast-recovery:

- alla ricezione della conferma di tutti i TSN ritrasmessi: torna alla fase di congestion-avoidance
- allo scadere di un timer RTO: torna alla fase di slow-start dopo aver aggiornato cwnd e ssthresh:

$$\text{cwnd} = \text{P-MTU}$$

$$\text{ssthresh} = max(\text{cwnd}/2, 4 \times \text{P-MTU})$$

2.8 Multi-homing

Una delle principali novità introdotte da SCTP rispetto a TCP è il *multi-homing*: un end-point in una associazione SCTP viene definito *multi-homed* quando può essere raggiunto attraverso più di un indirizzo di rete.

2.8.1 Dichiarazione degli indirizzi di rete

Quando due end-point instaurano una associazione si scambiano i chunk INIT (client) e INIT-ACK (server). Gli indirizzi utilizzati per inviare i pacchetti contenenti tali chunk sono di default considerati parte dell'associazione pertanto se l'associazione non è multi-homed non è necessario altro.

In caso di multi-homing però, gli end-point possono dichiarare i rispettivi indirizzi di rete aggiuntivi ai quali sono raggiungibili e che vogliono rendere parte

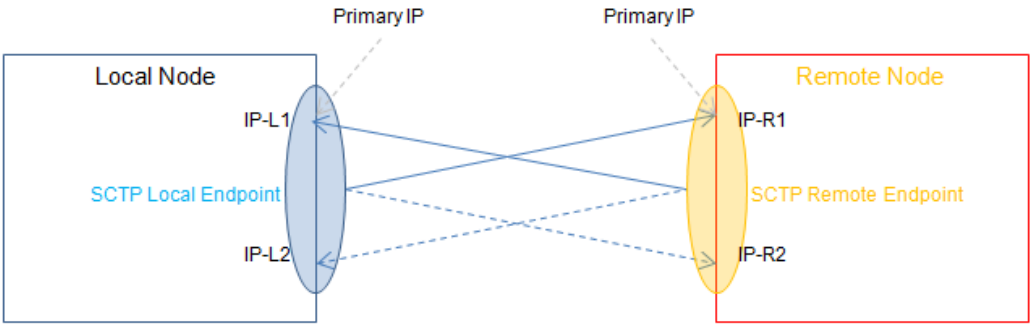

Figura 2.15: Esempio di associazione multi-homing SCTP

dell'associazione. Per far ciò vengono utilizzati i parametri opzionali IPv4, IPv6 e Host Name Address dei chunk INIT e INIT-ACK.

In particolare un end-point può dichiarare gli indirizzi aggiuntivi fornendo uno o più indirizzi IPv4 e/o IPv6 oppure in alternativa un Host Name Address (nome di dominio).

Nel caso di indirizzi IPv4 e/o IPv6 l'end-point si limiterà a registrare nell'associazione la coppia (IP, Source Port) per ciascun indirizzo, leggendo l'IP dal parametro opzionale del chunk e la porta sorgente dalla Common Header del pacchetto SCTP. Nel caso venga invece fornito un nome di dominio è compito dell'end-point che lo riceve risolverlo in indirizzi di rete per registrarli come parte dell'associazione. Questa ultima opzione, sebbene comporti un overhead aggiuntivo dovuto alla risoluzione DNS del nome di dominio, assicura una maggiore probabilità che l'associazione funzioni correttamente in presenza di un NAT tra i due end-point [24, p. 28-29]. Inoltre l'utilizzo di Host Name Address comporta un potenziale problema di sicurezza per il server: la risoluzione DNS lo lascia infatti in uno stato *half-open*. Pertanto sebbene normalmente la risoluzione DNS avviene immediatamente alla ricezione del nome di dominio nel caso dei server può essere posticipata alla ricezione del chunk COOKIE-ACK [24, p. 58-59].

2.8.2 Path primario e path secondari

Il multi-homing implica che un end-point, per raggiungere un end-point peer multi-homed, abbia a disposizione diversi indirizzi destinazione e quindi diversi *path* 'percorsi'.

Di questi path uno verrà scelto come primario (*primary-path*) e sarà utilizzato per l'invio di tutti i nuovi chunk DATA, mentre i rimanenti (*alternate-path*, o *backup-path*) vengono utilizzati solamente per le ritrasmissioni.

In ricezione l'implementazione SCTP rimane in ascolto su ciascuno degli indirizzi specificati, che può ricevere pacchetti dall'end-point peer in qualsiasi momento durante la vita dell'associazione.

Gli indirizzi del primary-path sono normalmente gli stessi utilizzati per instaurare l'associazione. Il livello superiore può però richiedere l'utilizzo di un indirizzo sorgente e/o destinazione diverso per rendere un determinato path primario in qualsiasi momento con apposite primitive rese disponibili dal protocollo [24, p.114]. Ciò rende possibile per esempio il cambio periodico del primary-path utilizzando i vari indirizzi disponibili in modalità round-robin, reso disponibile in alcuni sistemi operativi.

Il primary-path può cambiare solo in caso di richiesta esplicita da parte del livello superiore o in caso di *Path-failure* (vedi sezione 2.6.5), con conseguente *fail-over* (vedi sezione 2.8.3).

In condizioni di normalità, ogni nuovo chunk DATA viene inviato sul *primary-path* mentre eventuali ritrasmissioni dovrebbero essere inviate su un *alternate path* attivo, se disponibile. Anche in caso di timeout, i nuovi chunk DATA continuano ad essere inviati sul *primary-path*.

Si noti che ciò non vale per altri tipi di chunk: il chunk INIT-ACK per esempio deve sempre essere inviato allo stesso indirizzo dal quale è provenuto il chunk INIT. Analogamente, il chunk HEARTBEAT-ACK deve sempre essere inviato allo stesso indirizzo dal quale è provenuto il chunk HEARTBEAT. Anche il chunk SACK richiede di norma di essere inviato allo stesso indirizzo dal quale è provenuto il chunk DATA, tuttavia esiste una eccezione degna di nota: la ricezione di chunk DATA duplicati può significare che i relativi SACK non riescano ad arri-

vare a destinazione, in tal caso è consigliato l'utilizzo di un path alternativo per i rispettivi SACK.

2.8.3 Failover

Il primary-path può cambiare solo in caso di richiesta esplicita da parte del livello superiore o in caso di *Path-failure* (sezione 2.6.5): ciò si verifica nel caso in cui nel primary-path sia stato raggiunto il limite massimo di ritrasmissioni e/o mancate conferme di HEARTBEAT, chiamato *Path.Max.Retrans*. In questo caso viene effettuato il *fail-over*: l'endpoint può eleggere come primario un path alternativo, se esiste ed'è attivo.

Se sono presenti più path alternativi possibili ne deve essere scelto e utilizzato solamente uno per volta [24, p. 101] e la scelta di quale alternate-path utilizzare è lasciata agli implementatori consigliando di scegliere la coppia (sorgente, destinazione) più divergente possibile da quella che ha causato il path-failure [24, p. 87-88].

2.8.4 Utilità del multi-homing e ambiti di utilizzo

In molte applicazioni che utilizzano la rete è di grande importanza mantenere path di comunicazione affidabili e resilienti ai malfunzionamenti della rete.

Come è noto le reti a commutazione di pacchetti IP forniscono un servizio di tipo *best-effort*, cioè senza alcuna garanzia di effettiva consegna al destinatario, né tantomeno di qualità del servizio. I protocolli di routing riescono generalmente a trovare path alternativi in caso di malfunzionamenti lungo una certa rotta, ma i tempi necessari per tale convergenza sono inaccettabili per molte applicazioni.

Oltre ai possibili malfunzionamenti della rete, bisogna considerare possibili malfunzionamenti delle interfacce di rete che in caso di host *single-homed* determinano la totale irraggiungibilità e quindi la caduta del servizio.

La ridondanza dei path e delle interfacce di rete sono quindi un aspetto molto importante per le applicazioni che richiedono alta disponibilità.

Il multi-homing consente di utilizzare per la stessa associazione diverse interfacce di rete e diversi path offrendo una intrinseca maggiore resilienza ai malfunzionamenti delle interfacce di rete e delle reti locali o core utilizzate per la comunicazione. Un host infatti è in grado di utilizzare per la stessa associazione interfacce di diverso tipo (per esempio, fibra, wired Ethernet e Wifi), connesse a diverse network IP che a loro volta possono essere connesse a differenti network backbone. Ciò apre diverse possibilità di configurazione per ottenere alti livelli di disponibilità a seconda delle necessità specifiche.

Sebbene il principale scopo del multi-homing in SCTP sia aumentare la disponibilità di una associazione, il multi-homing ha ambiti di utilizzo molto più ampi. Per esempio, può essere utilizzato per il bilanciamento della trasmissione dei dati in parallelo sui vari path disponibili (*load-sharing*) o la selezione dinamica del path più efficiente disponibile (*hand-over*) in contesto mobile.

2.8.5 Limiti del multi-homing SCTP

Il multi-homing è stato implementato in SCTP per fare fronte alle necessità di alta disponibilità ed affidabilità richieste per il trasporto dei protocolli Signaling System No. 7 (SS7, vedi sezione 1.1.2). Il focus del multi-homing SCTP è stato quindi posto fin dal principio unicamente sulla ridondanza e conseguente maggiore disponibilità della associazione.

In particolare lo scopo del multi-homing SCTP è unicamente la gestione efficente delle ritrasmissioni e il *fail-over* in caso di errori ripetuti sul primary-path.

Altri ambiti di utilizzo del multi-homing, quali la gestione dinamica delle interfacce di rete dell'associazione in mobilità (*hand-over*) o l'aumento delle prestazioni tramite concorrenza delle trasmissioni (*load-sharing*) non sono nativamente supportate da SCTP ma possono essere ottenute tramite opportune estensioni.

Per esempio, una caratteristica del multi-homing SCTP è che non consente di default la gestione dinamica degli indirizzi di rete: gli indirizzi di rete coinvolti nella associazione devono essere dichiarati durante l'inizializzazione e non è previsto nessun meccanismo per cambiare dinamicamente tale set di indirizzi al variare delle condizioni (per esempio, nel caso in cui una interfaccia di rete viene

attivata e sarebbe desiderabile poterla aggiungere alla associazione SCTP esistente e utilizzarla come primary-path). Tali meccanismi di *handover* dinamico sono desiderabili in alcuni contesti come quello mobile (per esempio, passare con il proprio smartphone da Wifi a 4G durante una transazione bancaria su internet o durante una chiamata VoIP: vorremmo che la connessione rimanesse attiva pur cambiando interfaccia). Per ottenere tali funzionalità è stata definita una estensione di SCTP standardizzata da IETF: il Dynamic Host Resolution (DAR, vedi sezione 2.8.6) [25].

Un'altra caratteristica del multi-homing SCTP è che un solo path viene utilizzato per la trasmissione dati: il primary-path. In condizioni di normalità gli alternate-path vengono utilizzati solamente per le ritrasmissioni e il primary-path viene cambiato solo in caso di fail-over o di cambio esplicito richiesto dal livello superiore. In alcuni ambiti di utilizzo potrebbe essere desiderabile la trasmissione concorrente dei dati o dei meccanismi di alternanza dei path utilizzati come primary, ai fini della distribuzione della trasmissione dati sui diversi path disponibili (*load-sharing*). SCTP non è facilmente estendibile in questo contesto: il trasferimento di dati simultaneo su diversi path aumenta la necessità di ri-ordinamento dei pacchetti da parte del ricevente con conseguente degrado delle performance. Gli algoritmi di controllo della congestione SCTP infatti sono derivati da TCP (vedi sezione 2.7) e perdono di efficacia al crescere del ri-ordinamento in ricezione [16]. Di conseguenza, per ottenere la funzionalità di load-sharing in SCTP si rendono necessarie modifiche alla gestione del buffer in uscita e di adattamenti ai meccanismi di controllo della congestione. Sebbene siano state effettuate proposte [12] non esistono ancora estensioni per il load-sharing divenute standard.

2.8.6 Cenni sull'estensione Dynamic Host Reconfiguration (DAR)

Il *Dynamic-Host-Reconfiguration (DAR)* è una estensione di SCTP standardizzata da IETF che consente di realizzare *handover* dinamico degli indirizzi di rete, particolarmente utile in contesti mobile. Verranno qui accennati brevemente i principi alla base del suo funzionamento, per approfondimenti si rimanda al RFC [25].

DAR consente l'aggiunta o l'eliminazione dinamica di indirizzi di rete e il cambiamento dinamico del primary-path durante la vita di una associazione SCTP. In particolare un end-point grazie al DAR è in grado, senza dover re-inizializzare l'associazione, di:

- aggiungere dinamicamente un indirizzo di rete all'associazione
- eliminare dinamicamente un indirizzo di rete dall'associazione
- richiedere all'endpoint peer di utilizzare un determinato indirizzo di rete come primary-path

Per raggiungere tali obiettivi DAR introduce nuovi tipi di chunk detti **Address Configuration chunks**:

- **Address Configuration Change Chunk (ASCONF, Type = 0xC1)** attraverso il quale comunicare le modifiche degli indirizzi di rete dell'associazione, mostrato in fig. 2.16
- **Address Configuration Acknowledgment (ASCONF-ACK, Type = 0x80)** che conferma le modifiche richieste dal chunk ASCONF, mostrato in fig. 2.17

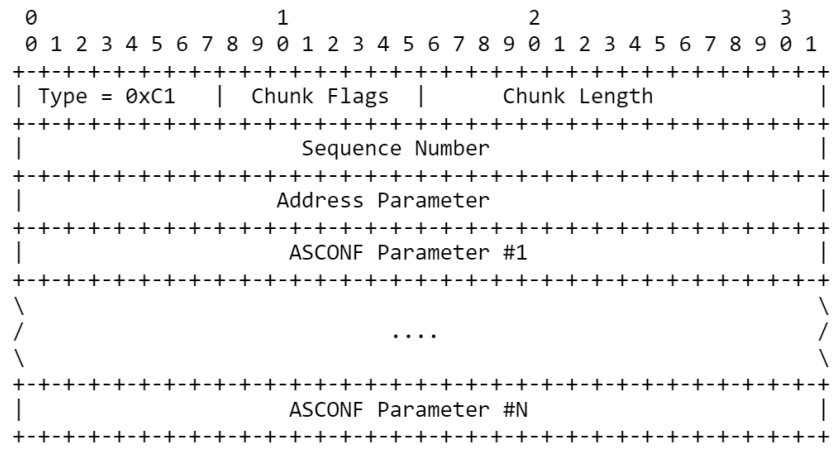

Figura 2.16: Struttura chunk ASCONF (Type = 0xC1)

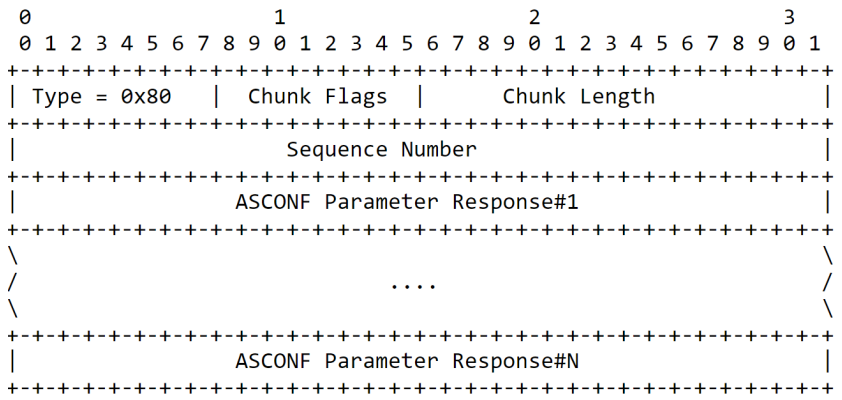

Figura 2.17: Struttura chunk ASCONF-ACK (Type = 0x80)

Address Configuration Parameters	Parameter Type	Chunk container
Add IP Address	0xC001	ASCONF
Delete IP Address	0xC002	ASCONF
Set Primary Address	0xC004	ASCONF, INIT, INIT-ACK
Error Cause Indication	0xC003	ASCONF-ACK
Success Indication	0xC005	ASCONF-ACK
Adaptation Layer Indication	0xC006	INIT, INIT-ACK
Supported Extensions	0x8008	INIT, INIT-ACK

Tabella 2.2: Parametri TLV introdotti dal DAR e rispettivi chunk contenitori

I chunk ASCONF e ASCONF-ACK devono essere obbligatoriamente autenticati secondo quanto previsto nel RFC 4895 [21] in modo tale da limitare i rischi di *association-hijacking* [25, p. 30-33]. Se vengono ricevuti non autenticati devono essere scartati dal ricevente.

DAR introduce inoltre diversi parametri di formato Type-Length-Value (TLV) per le varie operazioni disponibili, da utilizzare sia nei propri chunk ASCONF e ASCONF-ACK che nei chunk INIT e INIT-ACK a seconda del tipo. La tabella tabella 2.2 mette in correlazione i parametri disponibili con i relativi chunk in cui devono essere inseriti.

Il chunk ASCONF, oltre ai ben noti campi Type, Flags, Length dell'header

previsti per tutti i chunk (vedi sezione 2.1.2), contiene nel proprio campo Value i seguenti campi:

Sequence Number identifica il numero di sequenza del chunk ASCONF (analogamente al TSN dei chunk DATA)

Address Parameter è un parametro TLV di tipo IPv4 o IPv6 (vedi sezione 2.2.2) appartenente al mittente del chunk ASCONF e già parte dell'associazione

ASCONF Parameters sono uno o più parametri di tipo Type-Length-Value (TLV) definiti appositamente per il ASCONF dal DAR (vedi tabella 2.2)

I parametri ASCONF Parameter utilizzati per richiedere le modifiche contengono, oltre alle informazioni relative alla modifica stessa (per esempio, l'indirizzo da aggiungere o rimuovere), anche un campo **ASCONF-Request Correlation ID** che identifica univocamente la richiesta di modifica.

Ogni richiesta, prima di essere attuata, deve essere confermata tramite un chunk ASCONF-ACK contenente a sua volta per ogni richiesta di modifica ricevuta un parametro che indica successo o una causa di errore (vedi tabella 2.2).

I parametri di risposta contengono, analogamente ai parametri di richiesta, un campo **ASCONF-Response Correlation ID**: attraverso di esso è possibile mettere in correlazione la risposta con la richiesta alla quale si riferisce, il cui ASCONF-Request Correlation ID corrisponde.

In caso di mancata ricezione della conferma per le modifiche richieste non è possibile attuare le modifiche. Verrà invece utilizzato un meccanismo di timeout per re-inviare il pacchetto ASCONF fino ad ottenere la conferma.

2.9 Multi-streaming

Il concetto di *multi-streaming*, mostrato in fig. 2.18, si riferisce alla suddivisione dei dati trasmessi in *stream* 'flussi' logici distinti unidirezionali, trasmessi in parallelo sullo stesso canale e utilizzando la stessa associazione tra end-point.

Viene quindi utilizzata la stessa associazione tra end-point per diversi flussi logici che vengono multiplexati dal mittente e de-multiplexati dal ricevente.

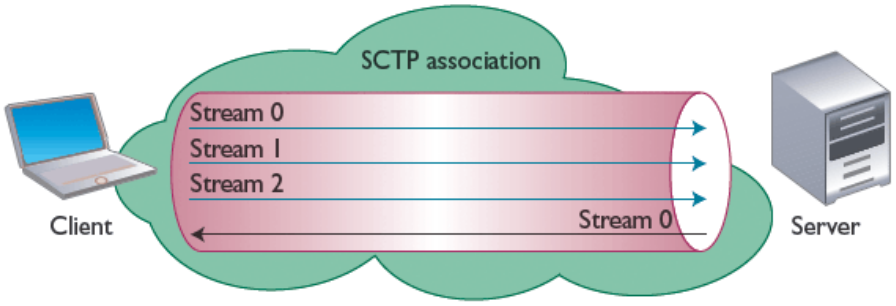

Figura 2.18: Concetto di multi-streaming

Il multi-streaming rappresenta, insieme al multi-homing, una delle principali caratteristiche distintive di SCTP rispetto a TCP. La sua implementazione è piuttosto semplice grazie alla struttura data ai pacchetti e ai chunk SCTP: il protocollo è stato infatti pensato fin dal principio per offrire questa funzionalità.

2.9.1 Dichiarazione del numero di stream

Il numero degli stream utilizzati viene dichiarato durante l'iniziazione della associazione (vedi sezione 2.3.1) dal client nel chunk **INIT** e dal server nel chunk **INIT-ACK** (vedi sezione 2.2.2). I campi utilizzati per entrambi i chunk sono gli stessi:

- **Number of Outbound Streams** dichiara il numero di stream in uscita che l'end-poind richiede di creare
- **Number of Inbound Streams** dichiara il numero massimo di stream in ingresso che l'end-point consente di creare

Al termine della inizializzazione della associazione ogni end-point sa quanti stream vuole creare, quanti al massimo ne può supportare, quanti il peer ne vuole creare e quanti al massimo il peer ne può supportare: verranno creati gli stream richiesti dal mittente se non eccedono il limite massimo imposto dal ricevente (altrimenti verrà utilizzato il numero di stream consentiti dal ricevente). Tali stream rimangono validi per tutta la durata di vita dell'associazione.

2.9.2 Stream ordinati

La fig. 2.19 mostra un esempio di multi-streaming con stream ordinati. I triangoli rappresentano i chunk DATA e i colori identificano gli stream. I rettangoli con sfondo giallo rappresentano i buffer dei rispettivi end-point e il rettangolo con sfondo azzurro l'associazione SCTP.

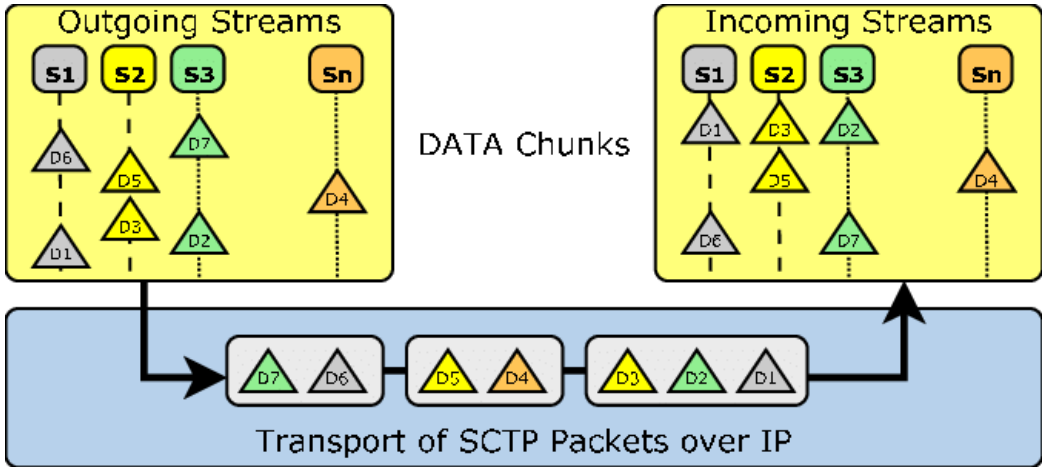

Figura 2.19: Esempio di multi-streaming con stream ordinati

Il multi-plexing dei messaggi utente (o loro frammenti) appartenenti a diversi stream avviene semplicemente tramite il bundling di diversi chunk DATA nello stesso pacchetto SCTP. Infatti un pacchetto SCTP può contenere diversi chunk DATA fino a raggiungimento del limite imposto dal P-MTU (vedi sezione 2.4.1) e ogni chunk DATA (vedi sezione 2.2.1) ha:

- **Stream Identifier**: identifica univocamente uno stream dell'associazione. Si noti che l'attribuzione di uno Stream Identifier valido è obbligatorio per ogni chunk DATA [24, p. 88]

- **Stream Sequence Number**: identifica univocamente il numero di sequenza del messaggio all'interno del proprio stream (parte da 0 a inizio associazione)

Ogni messaggio utente viene quindi assegnato ad uno stream e i messaggi utente dello stesso stream sono consegnati in modo ordinato e affidabile al li-

vello superiore, in modo indipendente da ciò che avviene negli altri stream (per esempio, in caso di dati mancanti e ritrasmissioni).

Si noti che i messaggi utente possono essere contenuti in un unico chunk o frammentati in più chunk (vedi sezione 2.4): in questo ultimo caso tali chunk avrebbero stesso Stream Identifier, stesso Stream Sequence Number e diverso TSN (il quale identifica l'ordine dei chunk), mentre l'inizio e la fine del messaggio sono determinati grazie ai flag Beginning (B) e Ending (E) del chunk DATA.

2.9.3 Stream non ordinati

L'ordinamento dei messaggi utente non è necessario per tutte le applicazioni. Talvolta l'ordinamento rappresenta un inutile overhead.

SCTP consente di utilizzare uno stream senza ordinamento: in ogni chunk DATA che non necessita di ordinamento viene impostato a 1 il flag **Unordered (U)** (vedi sezione 2.2.1 e sezione 2.5.2). Ciò indica al ricevente che il chunk non fa parte di uno stream ordinato, pertanto il campo Stream Sequence Number non ha significato e in ricezione il messaggio di cui il chunk fa parte deve essere consegnato immediatamente al livello superiore non appena completo.

Se un messaggio utente non ordinato è frammentato in diversi chunk DATA tutti i suoi frammenti devono avere U posto a 1. I frammenti del messaggio saranno comunque ordinati utilizzando il campo TSN presente e l'inizio e la fine del messaggio verranno individuati in ricezione grazie ai flag Beginning (B) e Ending (E) del chunk DATA (vedi sezione 2.2.1). A non essere ordinato è quindi il messaggio in relazione allo stream a cui appartiene, non il messaggio al suo interno (che verrà composto dai relativi chunk in ordine e solo allora consegnato al livello superiore).

2.9.4 Utilità del multi-streaming e contesti di utilizzo

Il multi-streaming elimina la necessità di aprire una connessione tra end-point per ogni flusso logico di dati (come avviene in TCP), utilizzando invece la stessa

associazione SCTP per diversi flussi logici che vengono multiplexati dal mittente e de-multiplexati dal ricevente.

SCTP risolve inoltre un problema noto di TCP, denominato **Head-Of-Line (HOL) blocking**: dato che in TCP esiste un solo stream, una interruzione nella sequenza fa si che la consegna a livello applicativo di tutti i segmenti successivi venga ritardata fino all'arrivo del segmento mancante.

SCTP è esente da tale problema in quanto i flussi di dati sono logicamente separati: eventuali incompletezze riscontrate in uno stream non ritardano la consegna ai livelli superiori degli altri stream completi.

La fig. 2.20 mostra un esempio pratico: i chunk mancanti sono il 9, il 4 e il 22. Sebbene i rispettivi stream siano in attesa di tali chunk mancanti per la consegna ordinata al livello superiore, il chunk 11 può essere consegnato al livello superiore immediatamente.

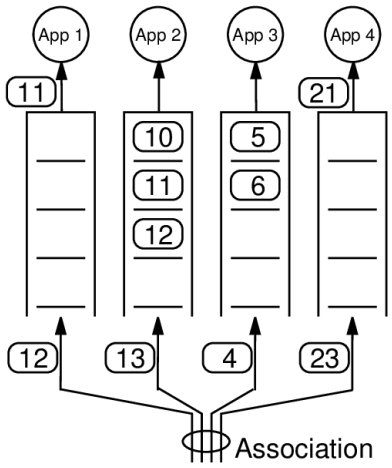

Figura 2.20: Esempio di come SCTP sia esente dal fenomeno di HOL-blocking

Come è noto TCP fornisce una consegna ordinata e affidabile, mentre UDP fornisce una consegna non ordinata e non affidabile. Esistono però applicazioni che necessitano di affidabilità pur non necessitando di alcun ordinamento o necessitando di un ordinamento solamente parziale.

La segnalazione in telefonia necessita dell'ordinamento dei soli messaggi appartenenti alla stessa risorsa (per esempio, la telefonata di un utente), mentre i

messaggi di segnalazione relativi ad altre risorse possono (e dovrebbero) essere consegnati in modo indipendente. In questi casi l'HOL-blocking di TCP causa un ritardo non necessario e inaccettabile in termini di prestazioni. Si pensi per esempio al caso in cui il primo segmento di una sequenza di messaggi di segnalazione vada perso: in tal caso tutti i successivi messaggi dovrebbero attendere la ritrasmissione ma questo ritardo è assolutamente superfluo e degrada le prestazioni. Inoltre il multi-streaming prevede che gli stream siano parte della stessa associazione e quindi soggetti allo stesso rate di invio e controllo di congestione, riducendo l'overhead. Questi aspetti hanno un impatto diretto sulle prestazioni dei messaggi di segnalazione.

Gli ulteriori ambiti in cui il multi-streaming può risultare utile sono svariati. Per esempio, può risultare utile per minimizzare la latenza in applicazioni di streaming o real-time, per creare meccanismi di priorità e trattamenti preferenziali tra vari stream o ancora per il trasferimento di gruppi di files in modo parallelo e indipendente tra loro ma tramite la stessa associazione. Per esempio SCTP, se sostituito a TCP nei protocolli HTTP e FTP (dove spesso vengono trasferiti numerosi file per ogni sessione), ridurrebbe l'overhead dato dall'instaurazione e la gestione di connessioni TCP multiple ed eliminerebbe l'HOL-blocking. Tale sostituzione non è però di fatto mai avvenuta perché l'introduzione di SCTP richiederebbe la modifica del codice sorgente di un enorme numero di applicazioni, sviluppate e testate per TCP.

Capitolo 3

Cenni di utilizzo di SCTP nelle precedenti generazioni 2G e 3G

Le generazioni di telefonia mobile 2G e 3G continuano ad utilizzare la logica di commutazione a circuito tipica della rete PSTN. Come visto in sezione 1.1.3 la realizzazione della commutazione a circuito e di tutti i servizi supplementari di telefonia avviene utilizzando il sistema di segnalazione SS7 che, a partire dagli anni Duemila, è stato esteso con la suite di protocolli SIGTRAN per consentirne il trasporto su base IP grazie ad una serie di protocolli di interfaccia detti adaptation-layer e un protocollo di trasporto comune che soddisfasse i requisiti di SS7: SCTP.

Alla nascita di SCTP negli anni Duemila la tecnologia 2G era già implementata e di uso comune mentre la 3G era nel suo periodo iniziale.

Nello stesso periodo in cui l'utilizzo di SIGTRAN veniva gradualmente introdotto nella rete PSTN, esso è stato introdotto anche nelle esistenti reti 2G e nelle nascenti 3G. In particolare è stato introdotto nella porzione di rete core (che si interfaccia con la PSTN) e nella porzione di *mobile-backhaul* tra le stazioni base che gestiscono gli apparati radio (celle) e le corrispondenti interfacce di ingresso alla rete core.

Per *mobile-backhaul*, come mostrato in fig. 3.1, si intende l'intersezione tra una rete mobile la corrispondente rete core di trasporto.

Trattandosi di una rete di interfaccia, la rete backhaul comprende aspetti corre-

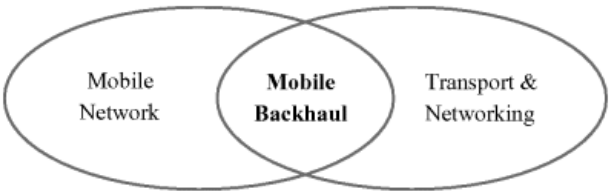

Figura 3.1: Concetto di mobile backhaul

lati alla rete mobile e aspetti correlati alla rete core a commutazione di pacchetto. La sua architettura ha impatto indiretto sul funzionamento di tutta la rete e la sua efficienza è di fondamentale importanza. Si noti che le connessioni tra le stazioni base della rete mobile e le rispettive interfacce di ingresso alla rete core sono state negli ultimi decenni un numero in continua crescita a causa dell'aumentare del numero di celle. Al crescere della capacità della rete e al diminuire delle dimensioni delle celle ha assunto sempre maggiore importanza il trasporto efficiente, affidabile e in grado di trasportare connessioni multiple della mobile backhaul per massimizzare l'efficienza e ridurre i costi degli operatori.

3.1 Rete 2G (GSM)

La fig. 3.2 mostra l'architettura semplificata della rete 2G (GSM). Tralasciando i dettagli tecnici che esulano dallo scopo di questa tesi, si individuano i seguenti componenti:

- **Mobile Station (MS)** è il dispositivo client GSM (per esempio, il classico cellulare quando utilizza la rete 2G)
- **Base Station Transceiver (BTS)** è composto dall'antenna e da tutti gli apparati correlati alla trasmissione/ricezione delle onde radio di una cella, gestisce la comunicazione con le varie MS attive nella cella e si interfaccia con un controller BSC

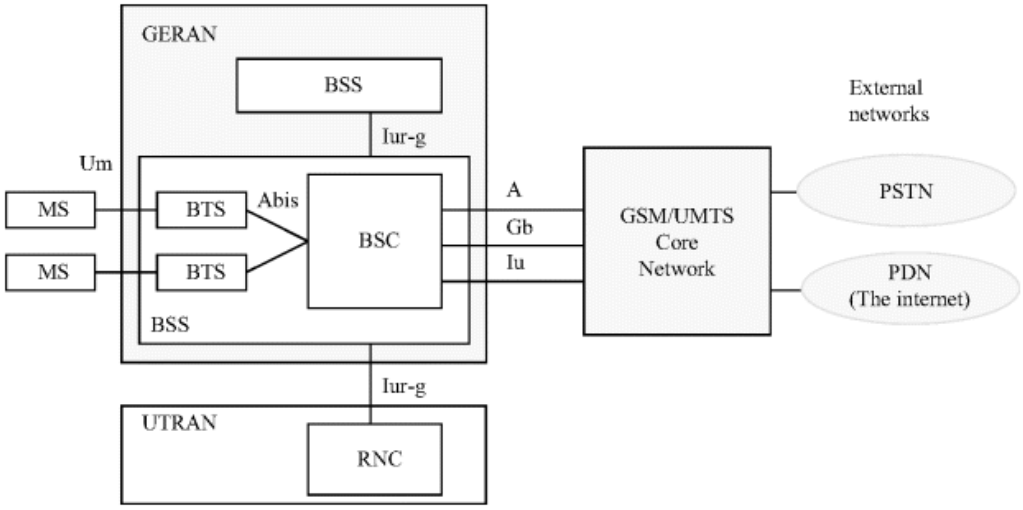

Figura 3.2: Architettura semplificata della rete 2G (GSM)

- **Base Station Controller (BSC)** è il controller che gestisce diversi BTS e si interfaccia con gli altri controller 2G (BSC) e 3G (RNC) e con la rete core GSM
- **Base Station Subsystem (BSS)** è la porzione della rete di accesso GSM con un controller BSC che gestisce diversi BTS e si interfaccia con la rete core GSM
- **GSM EDGE Radio Access Network (GERAN)**: è la rete di accesso GSM, composta da diversi BSS
- **Radio Network Controller (RNC)** è il controller delle stazioni radio base 3G, analogamente al BSC per le stazioni radio base 2G
- **UMTS Terrestrial Radio Access Network (UTRAN)** è la rete di accesso 3G (UMTS)

Le interfacce della rete mobile backhaul, visibili in fig. 3.2, sono:

- **A**: tra il controller BSC e la rete core 2G a commutazione di circuito. Inizialmente basata su multiplazione a divisione di tempo TDM, è stato ag-

giunto in seguito il funzionamento over-IP nella Release 7 del 3GPP del 2007

- **Gb**: tra il controller BSC e la rete core 2G a commutazione di pacchetto
- **Iu**: tra il controller BSC e la rete core 3G UMTS (se presente), introdotta nella Release 5 del 3GPP del 2002

Nella rete GSM i protocolli SIGTRAN, basati su trasporto SCTP, sono utilizzati nella **interfaccia A** (vedi fig. 3.2) tra la rete core GSM e ogni controller di stazione base BSC [18]. Attraverso tale interfaccia i messaggi di segnalazione entrano nella rete core GSM e vengono inoltrati nella rete PSTN.

3.2 Rete 3G (UMTS)

La fig. 3.3 mostra l'architettura semplificata della rete 3G (UMTS). Tralasciando i dettagli tecnici che esulano dallo scopo di questa tesi, si individuano i seguenti componenti:

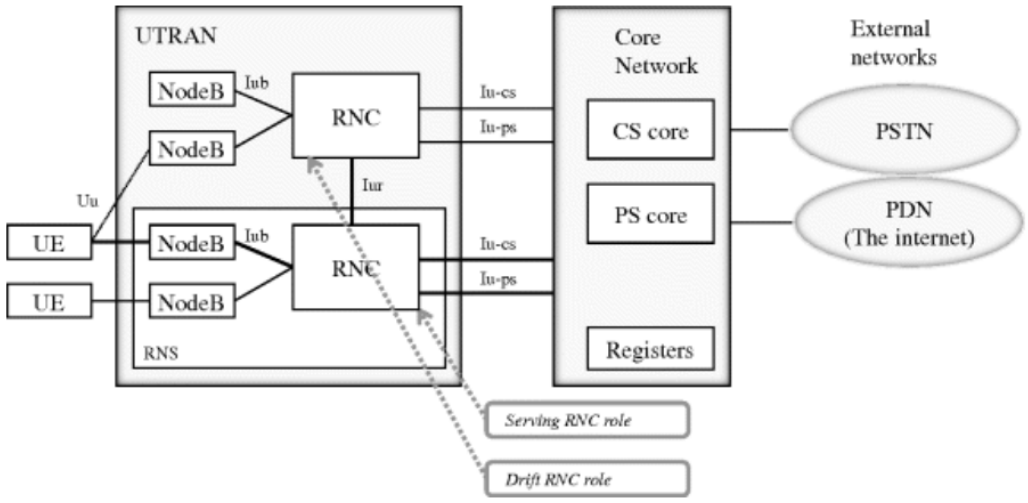

Figura 3.3: Architettura semplificata della rete 3G (UMTS)

- **User Equipment (UE)** è il dispositivo client UMTS (per esempio, il classico cellulare quando utilizza la rete UMTS)

- **NodeB** è composto dall'antenna e da tutti gli apparati correlati alla trasmissione/ricezione delle onde radio di una cella, gestisce la comunicazione con le varie UE attive nella cella e si interfaccia con un controller RNC
- **Radio Network Controller (RNC)** è il controller che gestisce diversi NodeB e si interfaccia con gli altri controller 3G (RNC) e 2G (BSC, vedi fig. 3.2) e con la rete core UMTS
- **Radio Network Subsystem (RNS)** è la porzione della rete di accesso UMTS con un controller RNC che gestisce diversi NodeB e si interfaccia con la rete core UMTS
- **UMTS Terrestrial Radio Access Network (UTRAN)**: è la rete di accesso UMTS, composta da diversi RNS

Le interfacce della rete mobile backhaul e UTRAN, visibili in fig. 3.3, sono:

- **Iu**: è il nome con cui ci si riferisce collettivamente alle interfacce **Iu-cs** (traffico a commutazione di circuito) e **Iu-ps** (traffico a commutazione di pacchetto) che collegano il controller RNC alla rete core UMTS
- **Iur (e Iur-g)**: connette RNC vicini. Esiste una versione analoga di questa interfaccia, chiamata **Iur-g** (vedi fig. 3.2), che connette un controller RNC (3G) con un controller BSS (2G)
- **Iub**: connette il controller RNC con le stazioni radio NodeB

Tutte le interfacce sopra riportate gestiscono, con stack di protocolli diversi, sia i messaggi utente (user-plane) che quelli di segnalazione (control-plane) [18].

Per tali interfacce il supporto over-IP è stato aggiunto nella Release 5 del 3GPP del 2002: prima di allora erano trasportate su un protocollo chiamato *Asynchronous Transfer Mode (ATM)* (a commutazione di circuito virtuale e trasmissione di cella, ovvero incapsulando i dati in unità, dette celle, di lunghezza fissa anziché in pacchetti a lunghezza variabile come avviene in IP) [18].

Il protocollo di segnalazione (control-plane) utilizzato nell'interfaccia Iu, tra i controller RNC e la rete core UMTS, è il *Radio Access Network Application Part*

(RANAP). Nella versione IP di UMTS prevista a partire dalla Release 5 del 3GPP del 2002 RANAP è mappato sui protocolli SIGTRAN e utilizza quindi SCTP come protocollo di trasporto (per entrambe le interfacce Iu-cs e Iu-ps). Attraverso tale interfaccia i messaggi di segnalazione entrano nella rete core UMTS e vengono inoltrati nella rete PSTN [18].

Anche il protocollo di segnalazione (control-plane) utilizzato nell'interfaccia Iur tra RNC vicini, chiamato *Radio Network Subsystem Application Part (RNSAP)*, viene mappato sui protocolli SIGTRAN a partire dalla dalla Release 5 del 3GPP, utilizzando quindi SCTP a livello di trasporto [18].

Nella rete UMTS, sempre a partire dalla dalla Release 5 del 3GPP, SCTP viene utilizzato come protocollo di trasporto anche per il protocollo di segnalazione *Node-B Application Part (NBAP)* utilizzato nell'interfaccia Iub tra le stazioni radio NodeB e i rispettivi controller RNC [18]. La stessa associazione SCTP viene utilizzata sia per il traffico uplink che downlink. Inoltre le *bearer*[1] vengono mappate su uno stream in uplink e uno stream in downlink [18].

L'operatività di NBAP è critica per il NodeB: se la connessione IP viene persa grazie alle caratteristiche di *failure-detection* di SCTP (vedi sezione 2.6.5) vengono avviate le procedure di recovery del NodeB, come per esempio una riconfigurazione o un restart, nell'arco di tempo di alcuni secondi (da pochi secondi a decine di secondi, a seconda delle configurazioni). Grazie alle caratteristiche di SCTP viene quindi limitato l'eventuale tempo di caduta di servizio dei NodeB [18].

[1] canali logici associati a una certa *Quality of Service (QoS)* e offerti dal Layer 2 ai livelli superiori per il trasferimento dei dati

Capitolo 4

Utilizzo di SCTP nelle reti 4G e 5G

4.1 Rete 4G

Per comprendere la rete 4G, è utile introdurre inanzitutto la nomenclatura utilizzata per le sue componenti principali.

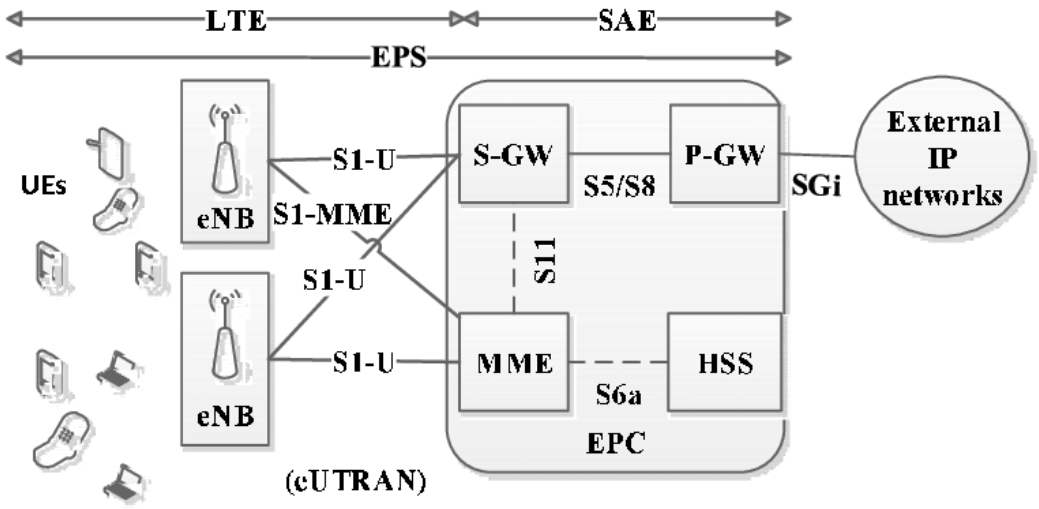

Figura 4.1: Architettura semplificata della rete 4G

L'architettura della rete 4G, mostrata in fig. 4.1, viene chiamata anche chiamata anche **Evolved Packet System (EPS)** ed è costituita dalle seguenti componenti fondamentali:

System Architecture Evolution (SAE) è la rete 4G non-radio, che comprende:

- **Evolved Packet Core (EPC)**: rete core 4G, che comprende diverse entità elencate in sezione 4.1.1
- porzione di rete backhaul

Long Term Evolution (LTE) o Evolved UTRAN (E-UTRAN) : è la rete di accesso radio, che comprende:

- **eNodeB (eNB)**: stazioni radio 4G contenenti le antenne e gli apparati strettamente connessi ad essi
- **User Equipment (UE)**: dispositivi mobile client 4G
- porzione di rete backhaul

La rete di quarta generazione apporta alcune importanti innovazioni:

All-over-IP: a differenza delle generazioni precedenti supporta esclusivamente la commutazione di pacchetto abbandonando quindi definitivamente la commutazione di circuito. Il suo scopo è quindi garantire un flusso dati IP in mobilità e senza soluzione di continuità tra i dispositivi mobili 4G degli utenti UE e la rete *Packet Data Network (PDN)*[1]. I servizi forniti con commutazione di circuito nelle generazioni precedenti possono essere supportate grazie all'uso di **bearer**[2] con bit-rate garantito

Interoperabilità con diverse reti di accesso: oltre che con la rete di accesso E-UTRAN (4G) supporta l'interoperabilità con le precedenti GERAN (2G) e UTRAN (3G) e altre tecnologie non-3GPP quali WiMAX, WLAN e reti fisse

Flat-architecture: nella rete di accesso LTE non esiste un controller centralizzato come nelle generazioni precedenti. L'intelligenza è decentralizzata nei

[1] rete a commutazione di pacchetto IP, per esempio Internet

[2] canale logico che garantisce una certa *Quality of Service (QoS)* per un flusso di pacchetti IP tra un dispositivo mobile e la rete PDN

eNodeB, che sono connessi tra loro e alla rete core EPC. Tale decentralizzazione riduce notevolmente i tempi di instaurazione della connessione e di handover, critici per le applicazioni real-time, inoltre riduce i costi ed elimina la necessità di un controller centralizzato ad alta disponibilità che costituisce un single-point-of-failure

Separazione tra control-plane e data-plane: le entità che gestiscono le segnalazioni di controllo e il traffico dati utente sono distinte in modo da essere scalabili indipendentemente e più facilmente ottimizzabili. La separazione consente inoltre di spostare le entità preposte allo user-plane il più vicino possibile alla rete di accesso in modo da ridurre la latenza, mantenendo invece le entità di controllo più centralizzate

4.1.1 Evolved Packet Core (EPC)

L'architettura della rete core EPC è composta da diverse entità, mostrate in fig. 4.2.

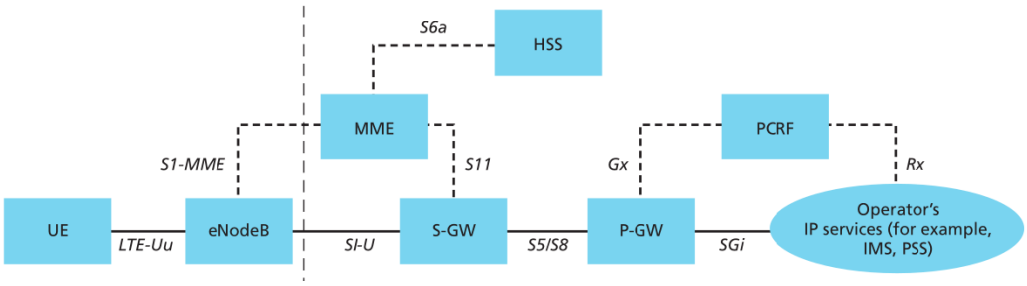

Figura 4.2: Principali entità della rete 4G

Packet Data Network GateWay (P-GW) è il gateway di interfaccia con la *Packet Data Network (PDN)* per il trasporto dei dati utente (data-plane). É responsabile inoltre dell'allocazione degli indirizzi IP per i dispositivi utente UE, del filtraggio dei pacchetti IP downlink nelle varie *bearer* in base alle QoS previste e del *Policy Control Enforcement Function (PCEF)* [3].

[3] addebito in base al flusso dati e alle regole reperite dal PCRF

Policy Control and Charging Rules (PCRF) è la funzione che contiene le regole di addebito e di autorizzazione a determinate QoS per gli utenti sottoscrittori del servizio, utilizzate dal gateway P-GW per il controllo e l'addebito dei flussi dati.

Serving GateWay (S-GW) è il gateway di interfaccia con la rete di accesso radio E-UTRAN che instrada i dati utente (data-plane) dai eNodeB al P-GW. Inoltre, funge da punto di ancoraggio per i dati utente di un determinato UE in movimento tra diversi eNodeB in modo da preservarli durante il passaggio.

Mobility Management Entity (MME) è la funzione che gestisce i messaggi di signaling (control-plane) tra i dispositivi utente UE e la rete core EPC: localizza e autentica gli UE in modo da stabilire quale gateway S-GW e P-GW debba occuparsi del flusso dati.

Home Subscriber Server (HSS) è il database contenente le informazioni dei clienti sottoscrittori del servizio necessarie per gestirne il flusso dati (profilo QoS), restrizioni per il roaming, PDN al quale è abilitato ad accedere, eccetera. Gestisce inoltre informazioni dinamiche di supporto ad altre funzioni, come per esempio l'identità del MME che sta gestendo l'utente e le chiavi di sicurezza per l'autenticazione (*Authentication Center (AUC)*).

4.1.2 Utilizzo del protocollo SCTP nella rete 4G

La fig. 4.3 mostra la rete di accesso E-UTRAN e la rete backhaul che connette quest'ultima con la rete core.

Comparando le architetture della rete di accesso 3G UTRAN (vedi fig. 3.3) e 4G E-UTRAN (vedi fig. 4.3), si possono notare alcune importanti differenze:

- E-UTRAN collassa tutte le funzioni della rete di accesso in un singolo nodo: eNodeB. Le funzioni di controllo sono decentralizzate e spostate più vicino agli utenti: non c'è bisogno di un controller centrale come in UTRAN (RNC) e quindi i requisiti di performance real-time prima richiesti a tutta la

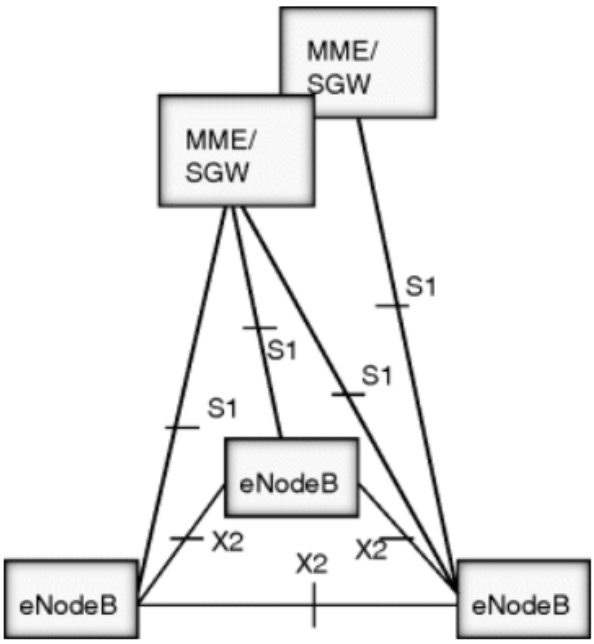

Figura 4.3: E-UTRAN e mobile-backhaul 4G

rete backhaul sono ora richiesti ai soli eNodeB. La rete backhaul è in questo modo notevolmente più semplice ed efficiente

- E-UTRAN introduce il collegamento logico diretto tra le stazioni radio (eNodeB), che non esiste nelle reti 2G e 3G e ottimizza l'hand-over tra essi (l'hand-over resta comunque possibile tramite l'interfaccia S1 in sua assenza, ma in questo modo è ottimizzato)

Una altra importante differenza è che, mentre nelle reti di precedenti generazioni il supporto a IP era una opzione tra altre possibili (per esempio, ATM in 3G), in LTE IP è l'unica opzione possibile.

Come si può vedere in fig. 4.3 ogni eNodeB può interfacciarsi con diversi nodi della rete core (cioè diversi MME e diversi SGW) e questa ridondanza consente un aumento della resilienza contro eventuali malfunzionamenti dei nodi della rete core.

Le interfacce utilizzate nella rete di accesso E-UTRAN e nella rete backhaul, mostrate in fig. 4.3, sono:

- **S1** interfaccia la rete E-UTRAN con la rete core EPC e si riferisce collettivamente a due interfacce:
  - **S1-MME** è l'interfaccia utilizzata per i messaggi di signaling (control-plane) tra ogni nodo radio eNodeB e i corrispondenti nodi di controllo MME nella rete core
  - **S1-U** è l'interfaccia utilizzata per i dati utente (data-plane) tra ogni nodo radio eNodeB e i corrispondenti gateway SGW nella rete core
- **X2** è l'interfaccia utilizzata nel collegamento degli eNodeB con gli eNodeB vicini e viene utilizzata sia per il data-plane che per il control-plane (con stack di protocolli diversi). Questa interfaccia è rilevante per l'*hand-over* del UE da un eNodeB ad un altro in mobilità. X2 semplifica e ottimizza l'handover ma non è strettamente necessaria ad esso: l'hand-over può avvenire anche attraverso l'interfaccia S1 nel caso una interfaccia X2 tra i due eNodeB non sia disponibile. Inoltre si tenga presente che X2 è una interfaccia logica: in una rete backhaul 4G reale è probabile che per le interfacce X2 il traffico venga instradato attraverso nodi di aggregazione (L2 o IP) nella rete di accesso, in quanto realizzare collegamenti fisici diretti tra ogni eNodeB risulterebbe molto costoso. Il volume di traffico correlato alle interfacce X2 non è molto elevato, ma la sua funzionalità ha importante ricadute sulla performance della rete in particolar modo per la gestione degli hand-over

Gli stack di protocolli utilizzati per tali interfacce è mostrato in fig. 4.4.

Come si può vedere in fig. 4.4, SCTP viene utilizzato nelle interfacce S1-MME e X2 (control-plane) per trasportare i messaggi di segnalazione rispettivamente tra eNodeB e rete core (MME) e tra eNodeB vicini tra loro. In tali interfacce ogni coppia di nodi di rete (rispettivamente (eNodeB, MME) per l'interfaccia S1-MME e (eNodeB, eNodeB) per l'interfaccia X2) instaura una singola associazione SCTP. I messaggi di segnalazione dei vari UE vengono separati logicamente grazie al multi-streaming mentre il multi-homing consente la ridondanza del trasporto di rete tra le multiple interfacce IP degli end-point, se presenti [1] [2].

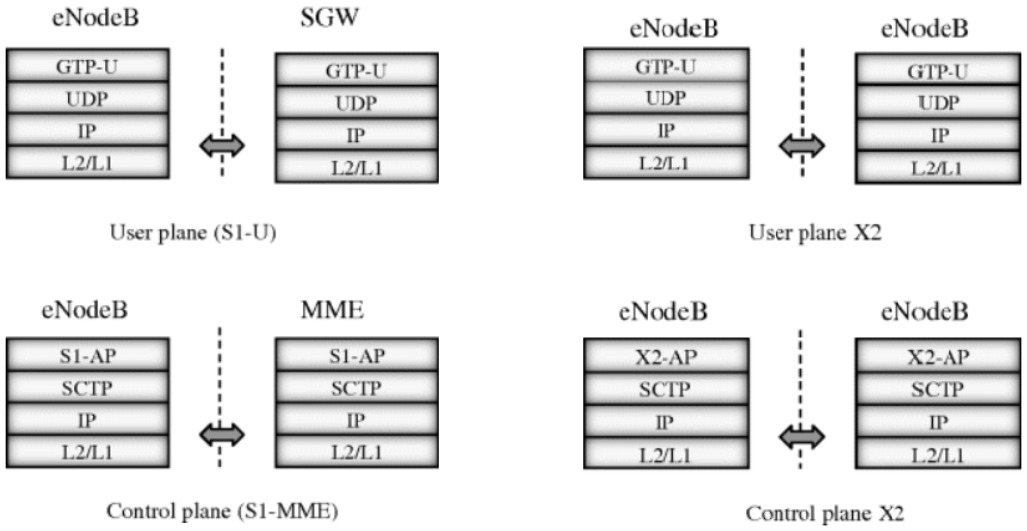

Figura 4.4: Stack di protocolli previsti per le interfacce S1 e X2

4.2 Rete 5G

Lo sviluppo della quinta generazione di reti cellulari 5G è stato motivato da diverse criticità:

- Rapido aumento del traffico dati (a causa per esempio di streaming video e del numero crescente di applicazioni in cloud)
- Rapido aumento del numero di dispositivi utente (smartphone, tablet, smartwatch, *Internet of Things (IoT)*), che tenderà a saturare le capacità di banda e il numero di dispositivi gestibili dagli operatori
- Necessità di tecnologie più efficienti dal punto di vista energetico e poco costose per gli operatori (e di conseguenza per gli utenti)
- Richiesta di caratteristiche avanzate di performance della rete per i servizi del futuro (veicoli a guida autonoma, robot a comando remoto, telemedicina, realtà virtuale, e molto altro)

Di conseguenza gli obiettivi che la rete 5G si prefigge di ottenere sono molto ambiziosi [15]:

- Data-rate di picco
  - Downlink: 20 Gbit/s
  - Uplink: 10 Gbit/s
- Latenza user-plane
  - eMBB (enhanced Mobile Broadband): 4ms
  - URLLC (Ultra-Reliable Low-Latency Communication): 1ms
- Latenza control-plane: 10-20ms per il passaggio idle $\rightarrow$ active
- Numero di dispositivi gestibili: circa un milione per km²
- Riduzione del consumo energetico: fino al 90% in meno rispetto alla 4G per ogni bit trasmesso

I casi d'uso della nuova rete 5G comprendono diversi domini applicativi in svariati settori, che si possono sintetizzare in tre macro-categorie [14], mostrate in fig. 4.5.

Enhanced Mobile Broadband (eMBB) il grande aumento di capacità di banda disponibile in mobilità renderà possibile applicazioni multimediali sempre più complesse, compresi streaming audio/video ad alta risoluzione, video 3D, videogiochi e applicazioni per il lavoro in cloud, realtà aumentata e altro ancora. Questo processo, già iniziato con le reti 4G, sarà destinato ad evolvere ulteriormente grazie ad un notevole aumento di performance della rete.

Ultra-Reliable and Low-Latency Communications (URLLC) le ridotte latenze e l'alta affidabilità e disponibilità della rete renderanno possibili scenari di utilizzo nuovi, come veicoli a guida autonoma, robot a controllo remoto (controllo industriale, operazione chirurgiche real-time e altro ancora) e numerose applicazioni correlate all'*Internet of Things (IoT)*. Queste caratteristiche saranno la base della quarta rivoluzione industriale.

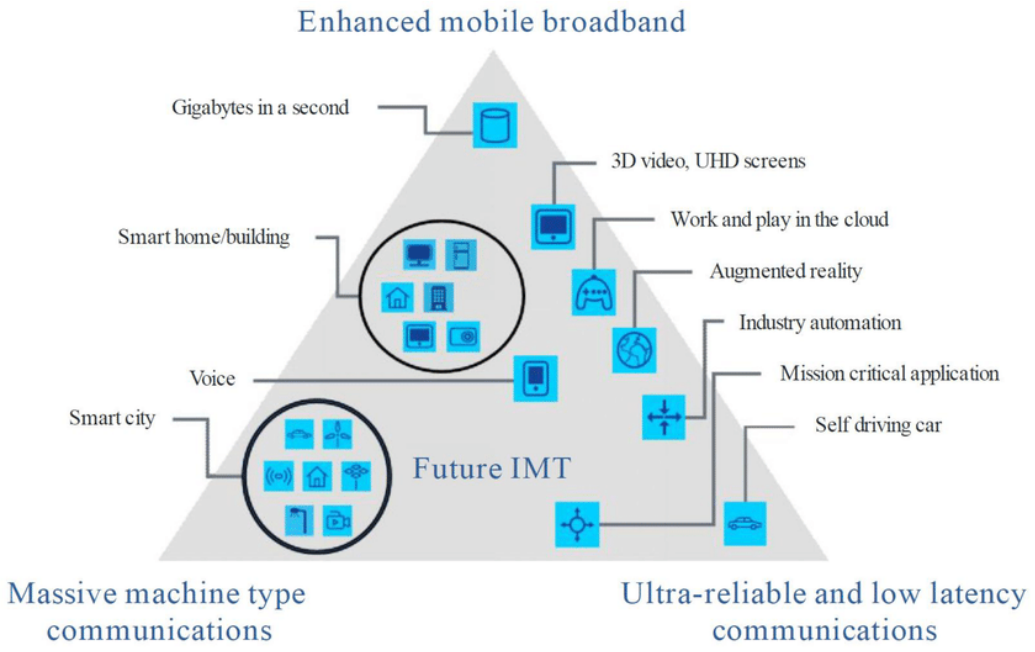

Figura 4.5: Casi d'uso 5G

Massive Machine-Type-Communications (mMTC) - Massive-Connectivity la rete consentirà la connettività ad un numero di dispositivi molto maggiore rispetto alle generazioni precedenti. Ciò consentirà la distribuzione dell'intelligenza e il dispiegamento di un numero enorme di dispositivi IoT e sensori che comunicheranno in rete tra loro, trasformando le città, le case, le industrie come oggi le conosciamo e fornendo nuove possibilità di sviluppo grazie all'Intelligenza Artificiale e al Machine Learning.

4.2.1 5G key enablers

Con il termine *5G key enablers* si intendono un insieme di tecnologie di fondamentale importanza per il raggiungimento degli obiettivi della rete 5G. Di seguito ne descriveremo brevemente i principali.

5G New Radio (5G-NR)

La principale tecnologia chiave della rete 5G è una nuova tecnologia radio, chiamata **5G New Radio (5G-NR)**, che utilizza un range di frequenze molto più ampio rispetto a quelle utilizzate dalle generazioni precedenti. In particolare il 3GPP TS.38.101-1 [4] individua due bande per il 5G-NR: **FR1** (410 MHz – 7125 MHz) e **FR2** (24250 MHz – 52600 MHz).

A frequenze radio più elevate corrisponde una portata più ridotta e di conseguenza celle di dimensione minore. Per non limitare la portata del servizio la rete 5G prevede quindi diverse tipologie di celle, associate a tipi di antenna, bande di frequenza, capacità di copertura e di penetrazione negli edifici differente. I dispositivi 5G utilizzeranno la tipologia più performante disponibile in base alla loro posizione (quindi la banda di frequenza più alta disponibile).

La fig. 4.6 da una idea generale delle aree di copertura e degli ambiti di utilizzo di tre diverse tipologie di cella 5G.

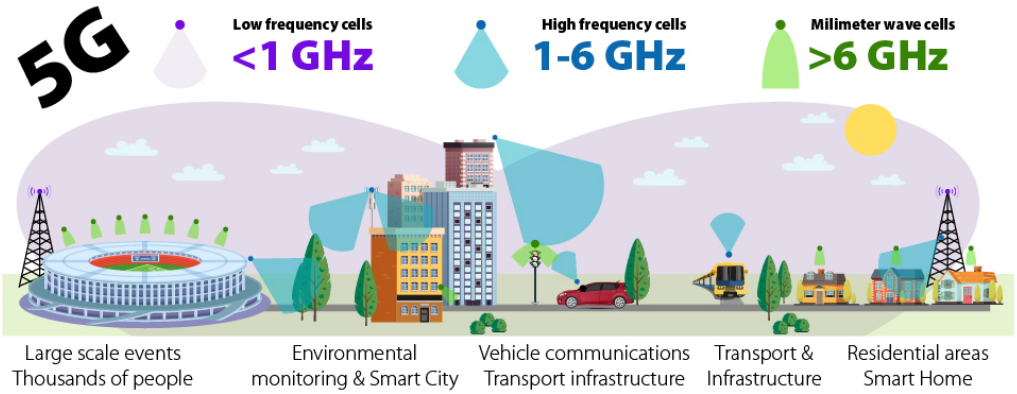

Figura 4.6: 5G Spectrum

Low-band (<1 GHz) è la banda a più ampia copertura (chilometri) e a maggiore capacità di penetrazione negli edifici, utilizzata anche dalle precedenti generazioni. A ciò corrisponde una minore velocità: le performance che questa banda può offrire sono quindi solo lievemente migliorative rispetto alle performance di LTE (800Mhz, velocità di picco intorno ai 100Mbps)

Mid-band (1-6 GHz) ha minore copertura e capacità di penetrazione rispetto alla low-band, ma con l'utilizzo di tecnologie come *Massive-MIMO*[4] e *beam-forming*[5] è in grado di creare flussi diretti verso gli utenti limitando le interferenze. Le velocità di picco raggiungibili sono maggiori, intorno a 1Gbps.

High-band (>6 GHz) è la banda delle *millimiter-waves (mmWave)* che consente capacità molto maggiori rispetto alle precedenti generazioni, con velocità di picco intorno ai 10Gbps e latenze estremamente ridotte. Allo stesso tempo la copertura e la capacità di penetrazione negli edifici sono molto ridotte: per ottenere una buona copertura è necessario impiegare un numero molto maggiore di celle. Tante piccole stazioni base a basso consumo energetico devono essere dispiegate, anche internamente ad edifici, e la tecnica del *beam-forming* assume un ruolo essenziale per limitarne le interferenze.

End-to-End Network Slicing

L'**End-to-End Network Slicing** consiste nella realizzazione, su una determinata infrastruttura di rete reale, di una serie di reti logiche/virtuali tra loro indipendenti e che funzionino in parallelo, a efficienza piena e senza interferenze come se avessero ognuna una rete fisica dedicata.

Ogni *slice* 'fetta' della rete reale è paragonabile quindi a tutti gli effetti a una rete completa *end-to-end*, creata appositamente per soddisfare i requisiti di *Quality of Service (QoS)* del proprio dominio applicativo, come mostrato in fig. 4.7.

Data la grande varietà di domini applicativi previsti per la rete 5G, ognuno con requisiti di QoS diversi, questa tecnologia assume una fondamentale importanza.

La realizzazione di queste reti logiche end-to-end orientate ai servizi è possibile grazie due tecnologie di fondamentale importanza:

[4]**Massive Multiple Input and Multiple Output (Massive-MIMO)**: è una tecnologia radio che utilizza un grande numero di antenne (anche centinaia) sia sul lato emittente sia sul lato ricevente, allo scopo di migliorare le prestazioni del canale di comunicazione

[5]**Beam-forming**: tecnologia che consente di concentrare e direzionare il segnale radio

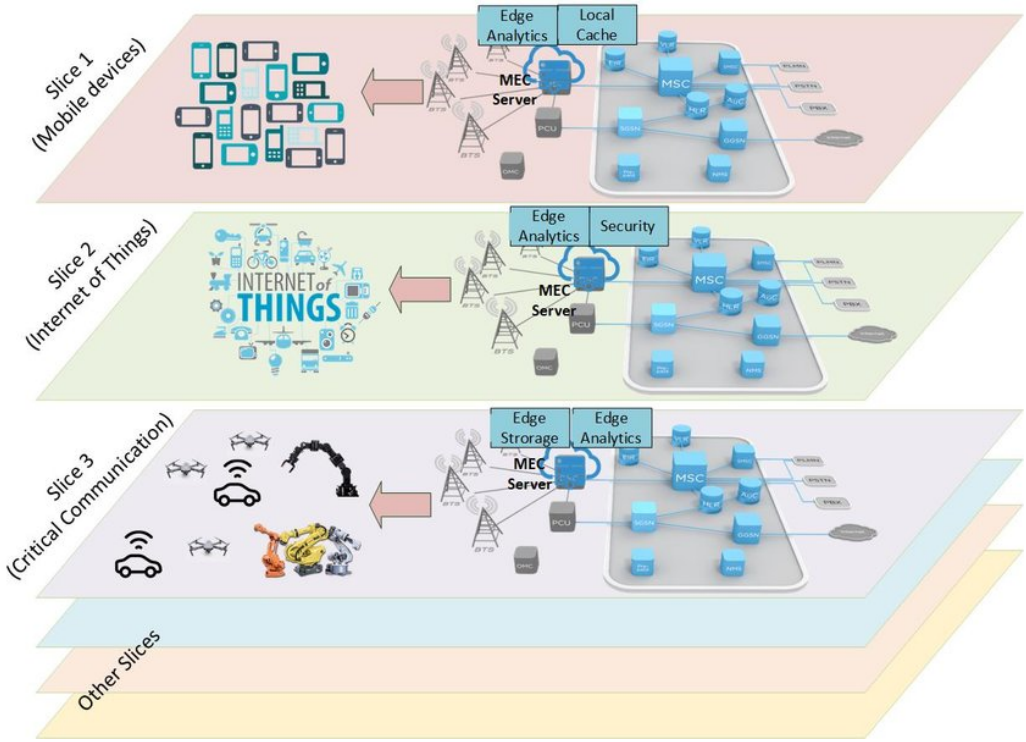

Figura 4.7: 5G End-to-End Network Slicing

Software Defined Networking (SDN) architettura di rete che prevede separazione tra control-plane e data-plane con centralizzazione dell'intelligenza di rete (control-plane) in uno o più controller separati. Ciò facilita l'amministrazione e la configurazione dell'intera rete e soprattutto ne consente la programmabilità e automazione attraverso software eseguiti dal controller che riflettono le logiche definite dall'operatore (*Network Automation*). La rete diventa una entità logica con funzionamento programmabile in modo indipendente dalla rete fisica sottostante.

Network Function Virtualization (NFV) consiste in una o più macchine virtuali, eseguite su server standard ad alta capacità, in cloud o su hardware appositi, che eseguono una versione virtuale delle funzioni di rete normalmente eseguite da hardware specializzati. Elaborazioni tradizionalmente residenti nei nodi di rete possono essere eseguite in modo centralizzato ed efficiente

sulla macchina virtuale, alleggerendo il carico elaborativo sui nodi di rete. Si riduce quindi il costo di gestione della rete (meno hardware specializzato, meno carico elaborativo nella rete stessa). Il vantaggio maggiore di questa tecnologia è pero che, se utilizzato in sinergia con il SDN, consente di realizzare reti virtuali ad-hoc in modo indipendente dalla infrastruttura di rete reale sottostante.

Grazie al SDN e alle NFV il gestore della rete fisica può coordinare in modo dinamico e automatizzato la distribuzione complessiva delle risorse tra gli slice allocando banda dove e quando serve sulla base delle caratteristiche e delle richieste del servizio.

La virtualizzazione delle funzioni di rete consente al gestore di abbattere i costi di infrastruttura virtualizzando le funzioni di controllo e mantenendo semplice e poco costosa l'architettura di rete reale. La sofisticazione della rete avviene sfruttando il software anziché l'hardware.

Dal punto di vista del business, ogni *slice* può essere amministrato da un operatore virtuale *(Mobile Virtual Network Operator (MVNO)*: il gestore dell'infrastruttura affitta le risorse fisiche agli MVNO che condividono la stessa rete reale ma realizzano le proprie *network-slice* secondo i requisiti dei servizi offerti agli utenti. Si noti che ciò consente alla rete 5G di essere utilizzata come *Internet Service Provider (ISP)* su infrastruttura mobile dagli operatori virtuali MVNO: questi ultimi potranno offrire agli utenti i propri servizi, tra cui il semplice l'accesso a Internet, sfruttando la slice di rete che affittano dal gestore.

Edge computing

Il concetto di *edge-computing*, mostrato in fig. 4.8, prevede di decentralizzare le risorse necessarie alle applicazioni (calcolo, storage, contenuti) in server dedicati il più possibile vicino agli utenti, tipicamente nella stessa rete dell'operatore o ai suoi confini. In un certo senso si può pensare all'edge computing come a un cloud-computing distribuito (talvolta definito *fog* 'nebbia', proprio per sottolineare la maggiore distribuzione rispetto al *cloud* 'nuvola') che fornisce un *execution-*

environment dedicato a una applicazione in diverse locazioni prossime agli utenti.

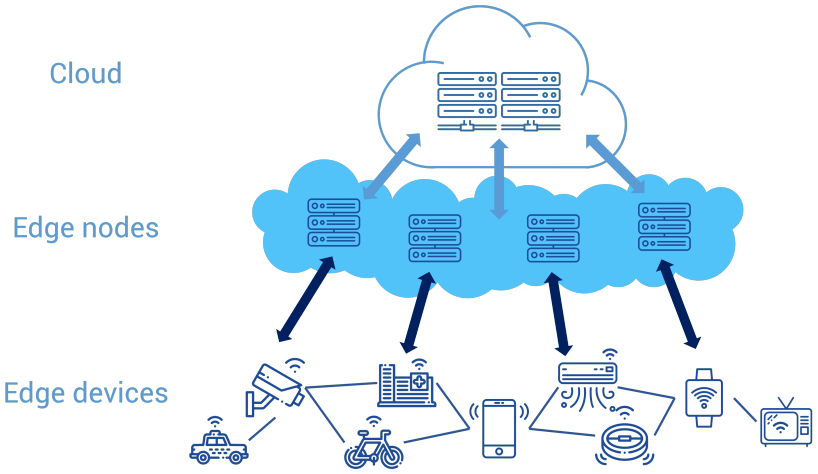

Figura 4.8: Concetto di Edge-computing

Per esempio, le risorse possono essere poste nelle aziende, nelle fabbriche, nelle case, nei veicoli. Tali server edge dedicati possono essere forniti e gestiti dall'operatore principale o da altri provider.

Grazie all'edge computing il fornitore del servizio applicativo può creare una gerarchia di server che riflette i livelli di calcolo/storage/memoria richiesti dall'applicazione, gestita su una unica rete o più precisamente sullo stesso *network-slice* (vedi sezione 4.2.1).

In questo modo si riducono notevolmente la latenza, la necessità di calcolo nei device utente e la congestione di rete relativa al traffico dati, riducendo i costi del gestore della rete e migliorando le performance per i fornitori del servizio con conseguente migliore esperienza degli utenti.

Service Based Architecture (SBA)

Rispetto alla rete core 4G (EPC), la rete core 5G (5GC) è suddivisa in tante network function virtuali NFV, adottando una logica a microservizi chiamata **Service Based Architecture (SBA)**, mostrata in fig. 4.9.

Le NFV all'interno della SBA comunicano tra loro attraverso interfacce RESTful-API basate su HTTP/2 e JSON ed esposte su un message-bus comune chiamato **Service Based Interface (SBI) Message Bus**.

I micro-servizi realizzati dalle NFV sono stati progettati secondo la logica **Control and User Plane Separation (CUPS)**, ovvero separando le funzioni di controllo da quelle relative ai dati utente.

4.2.2 Architettura della rete 5G

La fig. 4.9 mostra l'architettura semplificata della rete 5G. Come si può vedere in figura, l'architettura è suddivisa in una **rete core (5GC)** realizzata con architettura SBA (vedi sezione 4.2.1), una **rete di accesso radio (NG-RAN)** contenente stazioni **radio di nuova generazione (gNB)**, i **dispositivi utente (UE)** e la rete di backhaul.

Le principali NFV previste dalla nella rete core 5GC sono:

Access and Mobility Function (AMF) (control-plane) è la funzione di controllo dei messaggi di signaling tra le stazioni radio gNB e la rete core 5GC. Gestisce la registrazione e autenticazione (utilizzando la funzione AUSF), l'identificazione (utilizzando la funzione UDM) e la mobilità degli UE. É l'analogo della funzione MME della EPC 4G.

Session Management Function (SMF) (control-plane) Instaura e gestisce le sessioni utente. Alloca gli indirizzi IP per gli UE, seleziona e controlla la UPF più appropriata per il traffico dati (eventualmente la più vicina all'utente in caso di edge-computing), comunica alla AMF le informazioni di policy, addebitamento e QoS. É l'analogo delle funzioni svolte collettivamente da MME, SGW-C and PGW-C per la gestione delle sessioni nella EPC 4G.

User Plane Function (UPF) (user-plane) è responsabile dell'instradamento dei pacchetti, funge da *mobility-anchor* e *IP-anchor* nei confronti di Internet e gestisce la QoS prevista per l'utente. Si connette con il rispettivo *Point of Presence (POP)* e può opzionalmente integrare un firewall o NAT. É l'a-

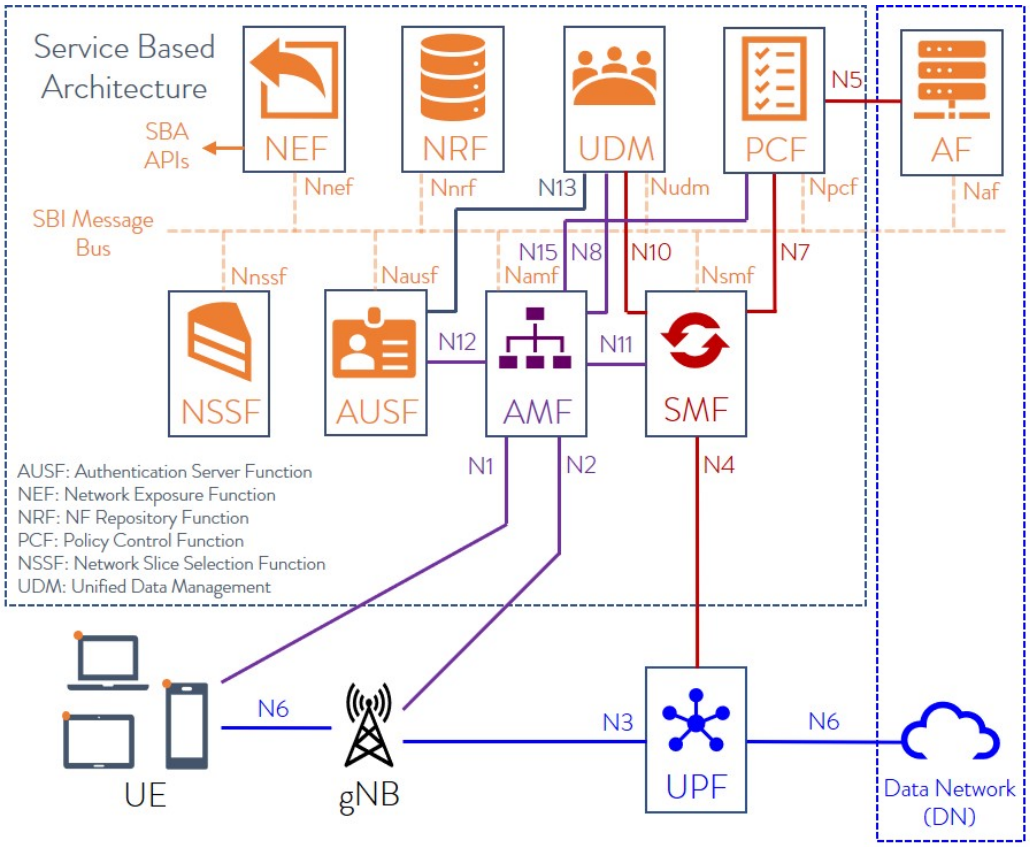

Figura 4.9: 5G Service Based Architecture (SBA)

nalogo delle funzioni di data-plane svolte collettivamente da SGW e PGW nella EPC 4G.

Policy Control Function (PCF) (control-plane) fornisce le regole di policy alle altre funzioni di controllo (comprese quelle relative a network-slicing, roaming, mobility-management e addebito). Accede alle informazioni sulla sottoscrizione del servizio dalla funzione UDM per le relative decisioni sulle policy. É l'analogo della funzione PCRF della EPC 4G.

Authentication Server Function (AUSF) (control-plane) autentica gli UE e gestisce le rispettive chiavi di autenticazione. É l'analogo delle funzioni di autenticazione svolte dal HSS della EPC 4G.

Unified Data Management (UDM) (control-plane) contiene le informazioni sugli utenti sottoscrittori del servizio e i loro profili, genera credenziali di autenticazione, identifica gli utenti e gestisce le rispettive autorizzazioni in merito ai servizi sottoscritti. Svolge parte delle funzioni relative alla gestione dei profili utente della HSS nella EPC 4G.

Application Function (AF) (control-plane) gestisce l'influenza dell'applicazione sull'instradamento del traffico, interagisce con la funzione PCF per il controllo delle policy, accede alla funzione NEF per il reperimento di risorse. É l'analogo della funzione AF nella EPC 4G.

NF Repository Function (NRF) (control-plane) effettua la registrazione e la ricerca dei servizi in modo che le NFV della SBA possano contattarsi a vicenda, mantiene i profili delle NFV e le corrispondenti istanze disponibili. Non è presente una funzione analoga nel EPC 4G.

Network Exposure Function (NEF) (control-plane) funge da interfaccia tra l'interno e l'esterno della rete 5GC attraverso l'esposizione di RESTful-API: fornisce un meccanismo per esporre in modo sicuro servizi, eventi e funzionalità della rete core 5GC e per fornire informazioni in modo sicuro da una applicazione esterna alla rete 5GC. Non è presente una funzione analoga nel EPC 4G.

Network Slice Selection Function (NSSF) (control-plane) seleziona l'insieme di istanze di network-slice che devono servire l'UE e determina quale AMF utilizzare. Non è presente una funzione analoga nel EPC 4G.

4.2.3 Strategie di realizzazione della rete 5G

Come per le generazioni precedenti, anche 5G è composta da una propria tecnologia radio, rete di accesso, rete di backhaul e rete core.

A differenza delle generazioni precedenti, dove la rete di accesso e la rete core dovevano essere necessariamente della stessa generazione, nel processo di standardizzazione della rete 5G è stata prevista la possibilità di integrare 4G e 5G

in diverse configurazioni. Gli operatori possono scegliere quindi diverse strategie di implementazione, illustrate in fig. 4.10.

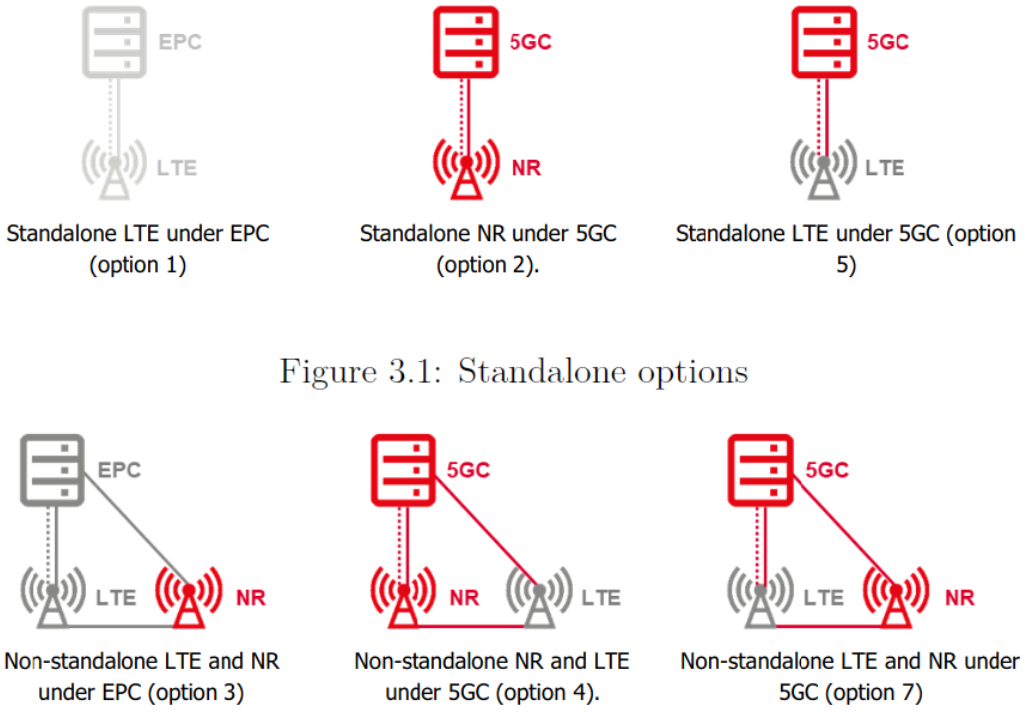

Figura 4.10: Modalità di implementazione della rete 5G: SA e NSA

Stand-Alone (SA) prevede l'utilizzo di una sola generazione nella rete di accesso, che si occupa quindi sia del control-plane che dello user-plane. L'utente passa da una tecnologia all'altra tramite hand-over con continuità del servizio come già avviene nelle tecnologie precedenti.

Non Stand-Alone (NSA) prevede l'utilizzo di entrambe le generazioni nella rete di accesso. Più precisamente, entrambe le generazioni vengono utilizzate per lo user-plane mentre una sola di esse gestisce il control-plane. Sfruttando la dual-connectivity i dispositivi degli utenti possono quindi ricevere separatamente i flussi dati 4G e 5G e aggregarli.

Come si è visto in sezione 4.2.1 la rete core 5GC è stata progettata con logica a microservizi invocati attraverso API standardizzate e con una logica di separazione tra le funzioni di control-plane e di user-plane (CUPS).

Una possibile strategia di evoluzione della attuale rete 4G verso la futura 5G consiste nell'introdurre tale cambio di paradigma nella rete 4G esistente, come mostrato in fig. 4.11:

- introducendo la logica CUPS tramite re-ingegnerizzazione delle funzioni EPC in modo da separare il control-plane dal data-plane (per esempio, suddividendo il SGW in SGW-C e SGW-U e il PGW in PGW-C e PGW-U)
- mappando le funzioni EPC 4G nelle corrispondenti funzioni SBA 5GC e adottandone la logica a microservizi

Seguendo questa strategia è possibile realizzare le funzioni dell'architettura SBA della rete 5GC attraverso le funzioni EPC 4G già esistenti. La rete risultante non è del tutto equivalente alla 5GC, in quanto alcune componenti della 5GC non hanno un corrispettivo nella EPC. É però considerato un possibile punto di partenza per la realizzazione della rete 5G sfruttando l'infrastruttura già esistente [17].

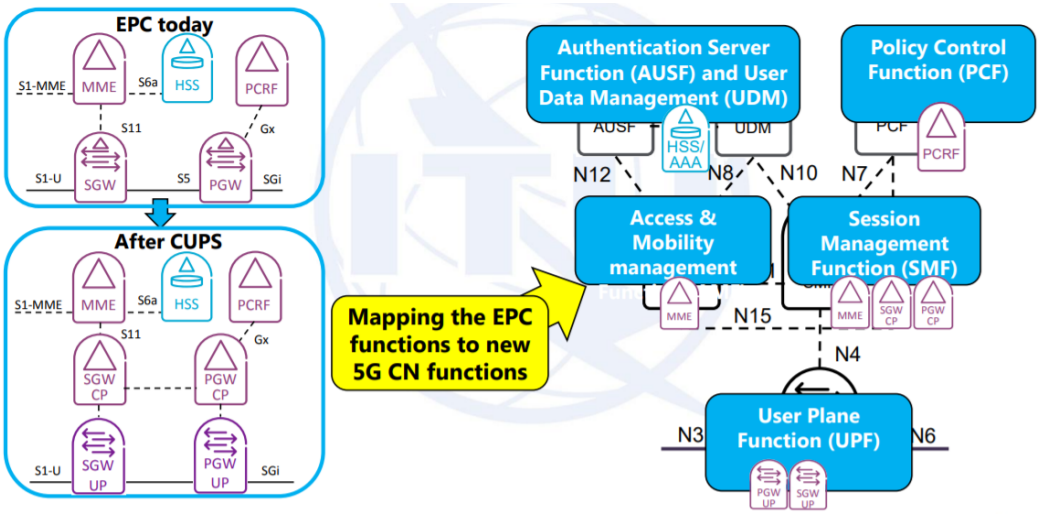

Figura 4.11: Strategia di evoluzione della rete 4G verso la nuova 5G

4.2.4 Utilizzo del protocollo SCTP nella rete 5G

La fig. 4.12 mostra l'architettura semplificata della rete 5G, in particolare la rete di accesso NG-RAN e la rete backhaul che la connette con la rete core.

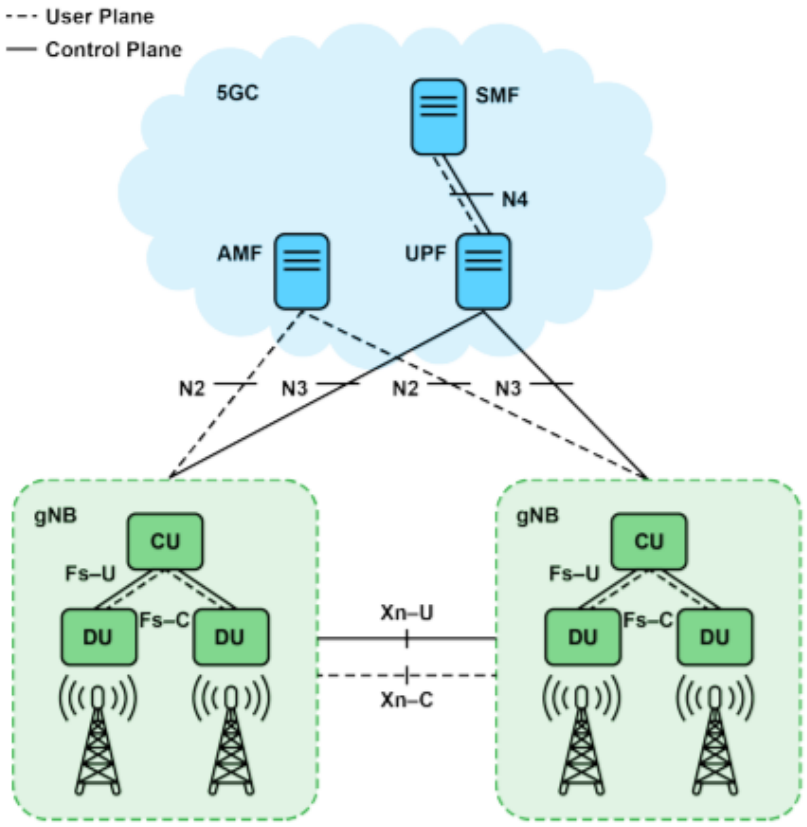

Figura 4.12: Architettura semplificata della rete 5G

Come si può vedere in fig. 4.12 ogni gNB può interfacciarsi con diversi nodi della rete core (cioè diversi AMF e diversi UPF) e questa ridondanza consente un aumento della resilienza contro eventuali malfunzionamenti dei nodi della rete core.

L'utilizzo di SCTP nella rete core 5GC è molto simile a quella vista nella rete 4G (vedi sezione 4.1.2), viene infatti utilizzato come protocollo di trasporto nelle seguenti interfacce:

- **N2** (anche chiamata NG-C nel 3GPP TS 138.412 [8]) è l'interfaccia utilizzata per i messaggi di signaling (control-plane) tra le stazioni radio gNB e la rete core 5GC, in particolare la funzione AMF (registrazione, autenticazione e mobilità degli UE). Per tali procedure è definito un protocollo di livello applicativo apposito **NG Application Protocol (NGAP)** (NG) [7], trasportato su SCTP/IP [8].

- **Xn-C** è l'interfaccia utilizzata per i messaggi di signaling (control-plane) tra stazioni radio gNB vicine tra loro. Tali procedure di segnalazione è definito un protocollo applicativo apposito **Xn Application Protocol (XnAP)** [3], trasportato su SCTP/IP [9].

Una novità rispetto alla E-UTRAN 4G, che utilizza una *flat-architecture*, è l'utilizzo di *split-architecture* nel gNB, con una unità di controllo centrale *Central Unit (CU)* e diverse unità distribuite *Distributed Unit (DU)*: SCTP viene utilizzato anche nell'**interfaccia F1-C** all'interno del gNB per i messaggi di signaling (control-plane) tra l'unità centrale gNB-CU e quelle distribuite gNB-DU [6].

Per ogni coppia di nodi di rete delle interfacce menzionate (rispettivamente (gNB, AMF) per l'interfaccia N2, (gNB, gNB) per l'interfaccia Xn-C e (gNB-CU, gNB-DU) per l'interfaccia F1-C) possono essere instaurate una o più associazioni SCTP.

Nel caso di associazioni multiple l'utilizzo delle singole associazioni può essere ristretto ad alcuni dei messaggi di signaling previsti (al contrario viene utilizzata la singola associazione per tutti i messaggi di signaling).

La possibilità di associazioni multiple tra coppie di nodi è una novità del 5G: come visto in sezione 4.1.2 nel 4G era previsto l'utilizzo di una unica associazione per coppia di nodi.

Analogamente a quanto avviene nella rete 4G, i messaggi di segnalazione dei vari UE vengono separati logicamente grazie al multi-streaming mentre il multi-homing consente la ridondanza del trasporto di rete tra le multiple interfacce IP degli end-point [8] [9] [6].

Capitolo 5

Esempio di traffico SCTP nel piano di controllo 5G

In questo capitolo verrà analizzato il traffico SCTP di un piano di controllo 5G implementato da Daniele Rossi nell'ambito della sua tesi di laurea magistrale [26]. La cattura Wireshark di tale implementazione è stata gentilmente offerta ai fini della analisi del protocollo SCTP in questa tesi.

Il core della rete 5G è stato virtualizzato con il software *Open5GS* e ogni componente è in esecuzione su una macchina virtuale distinta. La porzione di rete gNB e UE viene simulata dal software *UERANSIM*.

La cattura del traffico è stata effettuata nella macchina virtuale corrispondente alla AMF e contiene il traffico da e verso essa.

In particolare verrà di seguito analizzato il traffico SCTP corrispondente all'interfaccia N2 tra AMF e gNB, i cui corrispondenti indirizzi IP sono mostrati in tabella 5.1.

Entità	**IPv4**
AMF	10.250.2.85
gNB	10.250.2.165

Tabella 5.1: Indirizzi IPv4 corrispondenti a AMF e gNB nella cattura

La cattura del traffico TCP/HTTP/JSON tra AMF e le altre NFV della SBA, ovvero relativo al *Service Based Interface (SBI) Message Bus* (vedi sezione 4.2.1), è escluso dalla trattazione per semplicità di esposizione e per scarsa rilevanza ai fini del protocollo SCTP.

Il test effettuato prevede:

1. Registrazione e instaurazione della sessione di un UE
2. Ping a Google
3. Abbattimento del collegamento (necessario in quanto il simulatore non implementa ancora la disconnessione dell'UE: è necessario effettuare il kill dei processi con conseguente caduta dell'UE e del gNB)

No.	Time	Source	Destination	Protocol	Length	Info
22	10.643301	10.250.2.165	10.250.2.85	SCTP	82	INIT
23	10.643567	10.250.2.85	10.250.2.165	SCTP	306	INIT_ACK
24	10.644217	10.250.2.165	10.250.2.85	SCTP	278	COOKIE_ECHO
25	10.644336	10.250.2.85	10.250.2.165	SCTP	50	COOKIE_ACK
26	11.621610	10.250.2.165	10.250.2.85	NGAP	110	NGSetupRequest
27	11.621747	10.250.2.85	10.250.2.165	SCTP	62	SACK
28	11.622469	10.250.2.85	10.250.2.165	NGAP	118	NGSetupResponse
29	11.622679	10.250.2.165	10.250.2.85	SCTP	62	SACK
30	14.018739	10.250.2.165	10.250.2.85	NGAP/NAS-5GS	146	InitialUEMessage, Registration request
38	14.029658	10.250.2.85	10.250.2.165	NGAP/NAS-5GS	146	DownlinkNASTransport, Authentication request
39	14.217777	10.250.2.165	10.250.2.85	NGAP/NAS-5GS	146	UplinkNASTransport, Authentication response
48	14.225700	10.250.2.85	10.250.2.165	NGAP/NAS-5GS	126	DownlinkNASTransport
49	14.305341	10.250.2.165	10.250.2.85	NGAP/NAS-5GS	190	UplinkNASTransport
65	14.315764	10.250.2.85	10.250.2.165	NGAP/NAS-5GS	242	InitialContextSetupRequest
67	14.406436	10.250.2.165	10.250.2.85	NGAP	98	InitialContextSetupResponse
68	14.608436	10.250.2.85	10.250.2.165	SCTP	62	SACK
69	14.608920	10.250.2.165	10.250.2.85	NGAP/NAS-5GS	122	UplinkNASTransport
70	14.609442	10.250.2.85	10.250.2.165	NGAP/NAS-5GS	142	DownlinkNASTransport
71	14.812682	10.250.2.165	10.250.2.85	SCTP	62	SACK
79	20.240480	10.250.2.85	10.250.2.165	SCTP	98	HEARTBEAT
80	20.241074	10.250.2.165	10.250.2.85	SCTP	98	HEARTBEAT_ACK
85	26.640480	10.250.2.85	10.250.2.165	SCTP	98	HEARTBEAT
86	26.640819	10.250.2.165	10.250.2.85	SCTP	98	HEARTBEAT_ACK
87	28.954449	10.250.2.165	10.250.2.85	NGAP/NAS-5GS	150	UplinkNASTransport
116	29.019715	10.250.2.85	10.250.2.165	NGAP/NAS-5GS	226	PDUSessionResourceSetupRequest
121	29.110712	10.250.2.165	10.250.2.85	NGAP	118	PDUSessionResourceSetupResponse
127	29.312478	10.250.2.85	10.250.2.165	SCTP	62	SACK
135	34.544442	10.250.2.85	10.250.2.165	SCTP	98	HEARTBEAT
136	34.544961	10.250.2.165	10.250.2.85	SCTP	98	HEARTBEAT_ACK
137	35.280754	10.250.2.165	10.250.2.85	SCTP	54	SHUTDOWN
138	35.280824	10.250.2.85	10.250.2.165	SCTP	50	SHUTDOWN_ACK
139	35.281151	10.250.2.165	10.250.2.85	SCTP	50	SHUTDOWN_COMPLETE

Figura 5.1: Traffico SCTP nella cattura Wireshark analizzata

La fig. 5.1 mostra l'intera cattura di traffico SCTP: si possono notare la fase di inizializzazione dell'associazione con 4-way-handhsake, una serie di messaggi

NGAP (inseriti in chunk DATA SCTP), le conferme di ricezione SACK, diversi HEARTBEAT e HEARTBEAT-ACK e infine la terminazione dell'associazione con 3-way-handshake.

5.1 Inizializzazione 4-way dell'associazione

La fig. 5.2 mostra la sequenza di pacchetti relativi alla inizializzazione *4-way* della associazione tra gNB e AMF (vedi sezione 2.3.1).

No.	Time	Source	Destination	Protocol	Length	Info
22	10.643301	10.250.2.165	10.250.2.85	SCTP	82	INIT
23	10.643567	10.250.2.85	10.250.2.165	SCTP	306	INIT_ACK
24	10.644217	10.250.2.165	10.250.2.85	SCTP	278	COOKIE_ECHO
25	10.644336	10.250.2.85	10.250.2.165	SCTP	50	COOKIE_ACK

Figura 5.2: Traffico SCTP di inizializzazione dell'associazione

5.1.1 INIT (gNB $\rightarrow$ AMF)

Il pacchetto contenente il chunk INIT, mostrato in fig. 5.3, viene inviato da gNB ad AMF. In figura si possono vedere tutti i campi spiegati in sezione 2.2.2.

In particolare, si può notare che gNB richiede 10 stream in uscita e comunica la propria disponibilità ad accettarne fino ad un massimo di 65535 in ingresso.

gNB non specifica alcun indirizzo aggiuntivo tramite gli appositi parametri TLV, pertanto non richiede la funzionalità del multi-homing.

Gli unici parametri TLV aggiunti sono:

Supported address types parameter gNB comunica ad AMF che supporta indirizzi di tipo IPv4 e IPv6, tuttavia non fornendo indirizzi aggiuntivi l'unico utilizzato nell'associazione sarà l'indirizzo IPv4 sorgente dal quale gNB invia al pacchetto ad AMF

ECN parameter gNB comunica il suo supporto per l'*Explicit Congestion Notification (ECN)* (vedi [24, p. 138]), funzionalità già supportata da TCP che

permette la notifica di congestione di rete esplicita (nel caso entrambi gli end-point e la rete sottostante supportino tale funzionalità)

Forward TSN supported parameter gNB comunica il suo supporto per la non-affidabilità selettiva (*Partial Reliability Extension* [23]) [1]

```
> Frame 22: 82 bytes on wire (656 bits), 82 bytes captured (656 bits)
> Ethernet II, Src: fa:16:3e:73:f7:d1 (fa:16:3e:73:f7:d1), Dst: fa:16:3e:89:f3:69 (fa:16:3e:89:f3:69)
> Internet Protocol Version 4, Src: 10.250.2.165, Dst: 10.250.2.85
v Stream Control Transmission Protocol, Src Port: 54040 (54040), Dst Port: 38412 (38412)
    Source port: 54040
    Destination port: 38412
    Verification tag: 0x00000000
    [Association index: 65535]
    Checksum: 0xe2422ca2 [unverified]
    [Checksum Status: Unverified]
  v INIT chunk (Outbound streams: 10, inbound streams: 65535)
    v Chunk type: INIT (1)
        0... .... = Bit: Stop processing of the packet
        .0.. .... = Bit: Do not report
      Chunk flags: 0x00
      Chunk length: 36
      Initiate tag: 0xcbb02015
      Advertised receiver window credit (a_rwnd): 106496
      Number of outbound streams: 10
      Number of inbound streams: 65535
      Initial TSN: 490992593
    v Supported address types parameter (Supported types: IPv6, IPv4)
      v Parameter type: Supported address types (0x000c)
          0... .... .... .... = Bit: Stop processing of chunk
          .0.. .... .... .... = Bit: Do not report
        Parameter length: 8
        Supported address type: IPv6 address (6)
        Supported address type: IPv4 address (5)
    v ECN parameter
      v Parameter type: ECN (0x8000)
          1... .... .... .... = Bit: Skip parameter and continue processing of the chunk
          .0.. .... .... .... = Bit: Do not report
        Parameter length: 4
    v Forward TSN supported parameter
      v Parameter type: Forward TSN supported (0xc000)
          1... .... .... .... = Bit: Skip parameter and continue processing of the chunk
          .1.. .... .... .... = Bit: Do report
        Parameter length: 4
```

Figura 5.3: Pacchetto contenente il chunk INIT (gNB → AMF)

[1] tramite il chunk *FORWARD-TSN* il mittente può richiedere al ricevente l'avanzamento forzato del Cumulative Ack TSN, ignorando eventuali chunk mancanti nella sequenza fino al TSN specificato.

5.1.2 INIT-ACK (gNB ← AMF)

AMF risponde al chunk INIT di gNB con un pacchetto contenente un chunk INIT-ACK, mostrato in fig. 5.4 (vedi sezione 2.2.3).

```
> Frame 23: 306 bytes on wire (2448 bits), 306 bytes captured (2448 bits)
> Ethernet II, Src: fa:16:3e:89:f3:69 (fa:16:3e:89:f3:69), Dst: fa:16:3e:73:f7:d1 (fa:16:3e:73:f7:d1)
> Internet Protocol Version 4, Src: 10.250.2.85, Dst: 10.250.2.165
v Stream Control Transmission Protocol, Src Port: 38412 (38412), Dst Port: 54040 (54040)
    Source port: 38412
    Destination port: 54040
    Verification tag: 0xcbb02015
    [Association index: 65535]
    Checksum: 0x43a73126 [unverified]
    [Checksum Status: Unverified]
  v INIT_ACK chunk (Outbound streams: 30, inbound streams: 10)
    v Chunk type: INIT_ACK (2)
        0... .... = Bit: Stop processing of the packet
        .0.. .... = Bit: Do not report
      Chunk flags: 0x00
      Chunk length: 260
      Initiate tag: 0x328ffd2d
      Advertised receiver window credit (a_rwnd): 106496
      Number of outbound streams: 30
      Number of inbound streams: 10
      Initial TSN: 1887880873
    v State cookie parameter (Cookie length: 228 bytes)
      v Parameter type: State cookie (0x0007)
          0... .... .... .... = Bit: Stop processing of chunk
          .0.. .... .... .... = Bit: Do not report
        Parameter length: 232
        State cookie: 68b844fb43df727d22ec106ec5acd9a6d4852b5800000000…
    v ECN parameter
      v Parameter type: ECN (0x8000)
          1... .... .... .... = Bit: Skip parameter and continue processing of the chunk
          .0.. .... .... .... = Bit: Do not report
        Parameter length: 4
    v Forward TSN supported parameter
      v Parameter type: Forward TSN supported (0xc000)
          1... .... .... .... = Bit: Skip parameter and continue processing of the chunk
          .1.. .... .... .... = Bit: Do report
        Parameter length: 4
```

Figura 5.4: Pacchetto contenente il chunk INIT-ACK (gNB ← AMF)

Come si può notare in figura AMF richiede 30 stream in uscita e comunica la propria disponibilità ad accettarne fino ad un massimo di 10 in ingresso: l'associazione avrà quindi 10 stream in direzione gNB → AMF e 30 in direzione gNB ← AMF.

Anche AMF come gNB non specifica alcun indirizzo aggiuntivo, pertanto l'associazione non farà uso di multi-homing.

AMF invia a gNB lo *State cookie* (vedi sezione 2.3.1).

Gli unici parametri aggiuntivi, oltre allo State cookie, sono *ECN* e *Forward TSN supported* per dichiarare rispettivamente il supporto alla notifica di congestione di rete esplicita e alla non-affidabilità selettiva analogamente a quanto visto nel chunk INIT.

5.1.3 COOKIE-ECHO (gNB → AMF)

gNB re-inoltra ad AMF lo State cookie, ricevuto nel chunk INIT-ACK, con il chunk COOKIE-ECHO, mostrato in fig. 5.5. Ciò consente a AMF di ricostruire le strutture dati relative all'associazione precedemente eliminate (in modo da evitare lo stato *half-open* che crea vulnerabilità agli attacchi SYN-flood come visto in sezione 2.3.1).

```
> Frame 24: 278 bytes on wire (2224 bits), 278 bytes captured (2224 bits)
> Ethernet II, Src: fa:16:3e:73:f7:d1 (fa:16:3e:73:f7:d1), Dst: fa:16:3e:89:f3:69 (fa:16:3e:89:f3:69)
> Internet Protocol Version 4, Src: 10.250.2.165, Dst: 10.250.2.85
v Stream Control Transmission Protocol, Src Port: 54040 (54040), Dst Port: 38412 (38412)
    Source port: 54040
    Destination port: 38412
    Verification tag: 0x328ffd2d
    [Association index: 65535]
    Checksum: 0x6ee949fd [unverified]
    [Checksum Status: Unverified]
  v COOKIE_ECHO chunk (Cookie length: 228 bytes)
    > Chunk type: COOKIE_ECHO (10)
      Chunk flags: 0x00
      Chunk length: 232
      Cookie: 68b844fb43df727d22ec106ec5acd9a6d4852b5800000000…
```

Figura 5.5: Pacchetto contenente il chunk COOKIE-ECHO (gNB → AMF)

5.1.4 COOKIE-ACK (gNB ← AMF)

AMF utilizza lo State cookie ricevuto nel chunk COOKIE-ECHO per ricostruire le strutture dati relative alla associazione e alloca le risorse necessarie (vedi sezione 2.3.1).

A questo punto l'associazione è stabilita e AMF invia a gNB il chunk di conferma COOKIE-ACK come mostrato in fig. 5.6.

```
> Frame 25: 50 bytes on wire (400 bits), 50 bytes captured (400 bits)
> Ethernet II, Src: fa:16:3e:89:f3:69 (fa:16:3e:89:f3:69), Dst: fa:16:3e:73:f7:d1 (fa:16:3e:73:f7:d1)
> Internet Protocol Version 4, Src: 10.250.2.85, Dst: 10.250.2.165
v Stream Control Transmission Protocol, Src Port: 38412 (38412), Dst Port: 54040 (54040)
    Source port: 38412
    Destination port: 54040
    Verification tag: 0xcbb02015
    [Association index: 65535]
    Checksum: 0x0637ecf1 [unverified]
    [Checksum Status: Unverified]
  v COOKIE_ACK chunk
    > Chunk type: COOKIE_ACK (11)
      Chunk flags: 0x00
      Chunk length: 4
```

Figura 5.6: Pacchetto contenente il chunk COOKIE-ACK (gNB ← AMF)

5.2 Chunk DATA per il trasporto di messaggi NGAP

Come si può vedere in fig. 5.1 l'associazione SCTP viene utilizzata per trasportare una sequenza di messaggi NGAP all'interno dei chunk DATA.

Tutti i messaggi sono contenuti in un singolo chunk DATA, senza frammentazione (flag B e E entrambi impostati a 1, vedi sezione 2.4).

Gli stream utilizzati sono due: lo stream 0 viene utilizzato per i messaggi NGAP tra gNB e AMF non correlati al UE (fig. 5.7) mentre lo stream 1 viene utilizzato per i messaggi NGAP correlati al UE (fig. 5.8). Ciò è in accordo alle specifiche dell'interfaccia N2 [8] che prevedono di utilizzare uno Stream separato per i messaggi di signaling relativi a ciascun UE, mentre i messaggi non correlati a UE vengono mantenuti in stream appositi.

sctp.data_sid == 0

No.	Time	Source	Destination	Protocol	Length	Info
26	11.621610	10.250.2.165	10.250.2.85	NGAP	110	NGSetupRequest
28	11.622469	10.250.2.85	10.250.2.165	NGAP	118	NGSetupResponse

Figura 5.7: SCTP/NGAP tra gNB e AMF non correlato al UE (Stream 0)

La fig. 5.9 mostra un esempio di chunk DATA utilizzato per il trasporto di un messaggio NGAP, in particolare messaggio 2 dello stream 1 (SID=1, SSN=2).

sctp.data_sid == 1

No.	Time	Source	Destination	Protocol	Length	Info
30	14.018739	10.250.2.165	10.250.2.85	NGAP/NAS-5GS	146	InitialUEMessage, Registration request
38	14.029658	10.250.2.85	10.250.2.165	NGAP/NAS-5GS	146	DownlinkNASTransport, Authentication request
39	14.217777	10.250.2.165	10.250.2.85	NGAP/NAS-5GS	146	UplinkNASTransport, Authentication response
48	14.225700	10.250.2.85	10.250.2.165	NGAP/NAS-5GS	126	DownlinkNASTransport
49	14.305341	10.250.2.165	10.250.2.85	NGAP/NAS-5GS	190	UplinkNASTransport
65	14.315764	10.250.2.85	10.250.2.165	NGAP/NAS-5GS	242	InitialContextSetupRequest
67	14.406436	10.250.2.165	10.250.2.85	NGAP	98	InitialContextSetupResponse
69	14.608920	10.250.2.165	10.250.2.85	NGAP/NAS-5GS	122	UplinkNASTransport
70	14.609442	10.250.2.85	10.250.2.165	NGAP/NAS-5GS	142	DownlinkNASTransport
87	28.954449	10.250.2.165	10.250.2.85	NGAP/NAS-5GS	150	UplinkNASTransport
116	29.019715	10.250.2.85	10.250.2.165	NGAP/NAS-5GS	226	PDUSessionResourceSetupRequest
121	29.110712	10.250.2.165	10.250.2.85	NGAP	118	PDUSessionResourceSetupResponse

Figura 5.8: SCTP/NGAP tra gNB e AMF correlato al UE (Stream 1)

```
> Frame 65: 242 bytes on wire (1936 bits), 242 bytes captured (1936 bits)
> Ethernet II, Src: fa:16:3e:89:f3:69 (fa:16:3e:89:f3:69), Dst: fa:16:3e:73:f7:d1 (fa:16:3e:73:f7:d1)
> Internet Protocol Version 4, Src: 10.250.2.85, Dst: 10.250.2.165
v Stream Control Transmission Protocol, Src Port: 38412 (38412), Dst Port: 54040 (54040)
    Source port: 38412
    Destination port: 54040
    Verification tag: 0xcbb02015
    [Association index: 65535]
    Checksum: 0xfb705e02 [unverified]
    [Checksum Status: Unverified]
  v SACK chunk (Cumulative TSN: 490992596, a_rwnd: 106496, gaps: 0, duplicate TSNs: 0)
    > Chunk type: SACK (3)
    > Chunk flags: 0x00
      Chunk length: 16
    > Cumulative TSN ACK: 490992596
      Advertised receiver window credit (a_rwnd): 106496
      Number of gap acknowledgement blocks: 0
      Number of duplicated TSNs: 0
  v DATA chunk(ordered, complete segment, TSN: 1887880876, SID: 1, SSN: 2, PPID: 60, payload length: 163 bytes)
    > Chunk type: DATA (0)
    v Chunk flags: 0x03
        .... 0... = I-Bit: Possibly delay SACK
        .... .0.. = U-Bit: Ordered delivery
        .... ..1. = B-Bit: First segment
        .... ...1 = E-Bit: Last segment
      Chunk length: 179
    > Transmission sequence number: 1887880876
      Stream identifier: 0x0001
      Stream sequence number: 2
      Payload protocol identifier: NGAP (60)
      Chunk padding: 00
> NG Application Protocol
```

Figura 5.9: Messaggio NGAP trasportato da chunk DATA (SID = 1, SSN = 2)

5.3 Conferme selettive SACK

I chunk SACK vengono inseriti sia negli stessi pacchetti contenenti chunk DATA sia come unici chunk del pacchetto SCTP (nel caso in cui il ricevente non abbia dati aggiuntivi da trasmettere).

Un esempio di chunk DATA contenente un chunk SACK è mostrato in fig. 5.9.

Nella cattura non sono presenti chunk duplicati o persi, pertanto non ci sono esempi di *Gap Ack Blocks*, *Duplicate TSNs* e conseguenti ritrasmissioni (vedi sezione 2.2.4 e sezione 2.6.2).

5.4 Heartbeat

Durante la vita dell'associazione vengono inviati diverse volte dei chunk HEARTBEAT per confermare lo stato di attività del path (vedi sezione 2.6.4), visibili in fig. 5.1.

Le fig. 5.10 e fig. 5.11 mostrano rispettivamente un esempio di HEARTBEAT e la corrispondente risposta HEARTBEAT-ACK. Entrambi contengono un parametro *Heartbeat Info* che identifica l'HEARTBEAT inviato e corrisponde nel rispettivo HEARTBEAT-ACK.

```
> Frame 79: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
> Ethernet II, Src: fa:16:3e:89:f3:69 (fa:16:3e:89:f3:69), Dst: fa:16:3e:73:f7:d1 (fa:16:3e:73:f7:d1)
> Internet Protocol Version 4, Src: 10.250.2.85, Dst: 10.250.2.165
v Stream Control Transmission Protocol, Src Port: 38412 (38412), Dst Port: 54040 (54040)
    Source port: 38412
    Destination port: 54040
    Verification tag: 0xcbb02015
    [Association index: 65535]
    Checksum: 0x759affff [unverified]
    [Checksum Status: Unverified]
  v HEARTBEAT chunk (Information: 48 bytes)
    > Chunk type: HEARTBEAT (4)
      Chunk flags: 0x00
      Chunk length: 52
    v Heartbeat info parameter (Information: 44 bytes)
      > Parameter type: Heartbeat info (0x0001)
        Parameter length: 48
        Heartbeat information: 0200d3180afa02a500000000000000000000000000000000…
```

Figura 5.10: Esempio di HEARTBEAT

```
> Frame 80: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
> Ethernet II, Src: fa:16:3e:73:f7:d1 (fa:16:3e:73:f7:d1), Dst: fa:16:3e:89:f3:69 (fa:16:3e:89:f3:69)
> Internet Protocol Version 4, Src: 10.250.2.165, Dst: 10.250.2.85
v Stream Control Transmission Protocol, Src Port: 54040 (54040), Dst Port: 38412 (38412)
    Source port: 54040
    Destination port: 38412
    Verification tag: 0x328ffd2d
    [Association index: 65535]
    Checksum: 0xee78ab1d [unverified]
    [Checksum Status: Unverified]
  v HEARTBEAT_ACK chunk (Information: 48 bytes)
    > Chunk type: HEARTBEAT_ACK (5)
      Chunk flags: 0x00
      Chunk length: 52
    v Heartbeat info parameter (Information: 44 bytes)
      > Parameter type: Heartbeat info (0x0001)
        Parameter length: 48
        Heartbeat information: 0200d3180afa02a500000000000000000000000000000000…
```

Figura 5.11: Esempio di HEARTBEAT-ACK

5.5 Terminazione 3-way dell'associazione

La fig. 5.12 mostra la sequenza di pacchetti relativi alla terminazione *3-way* (o *graceful shutdown*) dell'associazione (vedi sezione 2.3.2).

No.	Time	Source	Destination	Protocol	Length	Info
137	35.280754	10.250.2.165	10.250.2.85	SCTP	54	SHUTDOWN
138	35.280824	10.250.2.85	10.250.2.165	SCTP	50	SHUTDOWN_ACK
139	35.281151	10.250.2.165	10.250.2.85	SCTP	50	SHUTDOWN_COMPLETE

Figura 5.12: Sequenza di pacchetti SCTP di terminazione dell'associazione

Come accennato ad inizio capitolo, nel test è stato necessario l'abbattimento del collegamento tramite kill dei processi in quanto il simulatore non implementa ancora la disconnessione dell'UE. Ciò provoca la caduta del gNB che quindi inizializza la procedura di graceful shutdown:

- gNB chiama la procedura SHUTDOWN e smette di accettare dati di invio dal livello superiore. Invia i chunk rimasti in sospeso (in questo caso nessuno) e invia il chunk SHUTDOWN, mostrato in fig. 5.13, per avvertire AMF della richiesta di chiusura della associazione. Il chunk SHUTDOWN contiene il *Cumulative TSN Ack* che notifica ad AMF quale è l'ultimo TSN ricevuto correttamente da gNB.

- AMF riceve il chunk SHUTDOWN, smette di accettare dati di invio dal livello superiore e procede con l'invio dei chunk rimasti in sospeso e non confermati dal gNB (in questo caso nessuno). Tramite il *Cumulative TSN Ack* ricevuto AMF verifica la ricezione di tutti i chunk e invia la conferma a gNB con il chunk SHUTDOWN-ACK, mostrato in fig. 5.14

- gNB riceve il chunk SHUTDOWN-ACK che conferma l'avvenuta ricezione della sequenza di chunk da parte di AMF, procede a chiudere l'associazione liberando le risorse allocate e invia il chunk SHUTDOWN-COMPLETE di conferma chiusura a AMF, mostrato in fig. 5.15

- AMF riceve il chunk SHUTDOWN-COMPLETE e procede anch'essa a chiudere l'associazione liberando le risorse allocate

```
> Frame 137: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
> Ethernet II, Src: fa:16:3e:73:f7:d1 (fa:16:3e:73:f7:d1), Dst: fa:16:3e:89:f3:69 (fa:16:3e:89:f3:69)
> Internet Protocol Version 4, Src: 10.250.2.165, Dst: 10.250.2.85
v Stream Control Transmission Protocol, Src Port: 54040 (54040), Dst Port: 38412 (38412)
    Source port: 54040
    Destination port: 38412
    Verification tag: 0x328ffd2d
    [Association index: 65535]
    Checksum: 0x3f1f2cad [unverified]
    [Checksum Status: Unverified]
  v SHUTDOWN chunk (Cumulative TSN ack: 1887880878)
    > Chunk type: SHUTDOWN (7)
      Chunk flags: 0x00
      Chunk length: 8
      Cumulative TSN Ack: 1887880878
```

Figura 5.13: Chunk SHUTDOWN

```
> Frame 138: 50 bytes on wire (400 bits), 50 bytes captured (400 bits)
> Ethernet II, Src: fa:16:3e:89:f3:69 (fa:16:3e:89:f3:69), Dst: fa:16:3e:73:f7:d1 (fa:16:3e:73:f7:d1)
> Internet Protocol Version 4, Src: 10.250.2.85, Dst: 10.250.2.165
v Stream Control Transmission Protocol, Src Port: 38412 (38412), Dst Port: 54040 (54040)
    Source port: 38412
    Destination port: 54040
    Verification tag: 0xcbb02015
    [Association index: 65535]
    Checksum: 0x3fbece93 [unverified]
    [Checksum Status: Unverified]
  v SHUTDOWN_ACK chunk
    > Chunk type: SHUTDOWN_ACK (8)
      Chunk flags: 0x00
      Chunk length: 4
```

Figura 5.14: Chunk SHUTDOWN-ACK

```
> Frame 139: 50 bytes on wire (400 bits), 50 bytes captured (400 bits)
> Ethernet II, Src: fa:16:3e:73:f7:d1 (fa:16:3e:73:f7:d1), Dst: fa:16:3e:89:f3:69 (fa:16:3e:89:f3:69)
> Internet Protocol Version 4, Src: 10.250.2.165, Dst: 10.250.2.85
v Stream Control Transmission Protocol, Src Port: 54040 (54040), Dst Port: 38412 (38412)
    Source port: 54040
    Destination port: 38412
    Verification tag: 0x328ffd2d
    [Association index: 65535]
    Checksum: 0x494ce992 [unverified]
    [Checksum Status: Unverified]
  v SHUTDOWN_COMPLETE chunk
    > Chunk type: SHUTDOWN_COMPLETE (14)
    > Chunk flags: 0x00
      Chunk length: 4
```

Figura 5.15: Chunk SHUTDOWN-COMPLETE

Bibliografia

[1] 3rd Generation Partnership Project (3GPP). Evolved Universal Terrestrial Radio Access Network (E-UTRAN); S1 signalling transport. Technical specification TS 36.412, December 2010.

[2] 3rd Generation Partnership Project (3GPP). Evolved Universal Terrestrial Radio Access Network (E-UTRAN); X2 signalling transport. Technical specification TS 36.422, December 2010.

[3] 3rd Generation Partnership Project (3GPP). 5G; NG-RAN; Xn Application Protocol (XnAP). Technical specification TS 138.423, September 2018.

[4] 3rd Generation Partnership Project (3GPP). NR; User Equipment (UE) radio transmission and reception; Part 1: Range 1 Standalone. Technical specification TS 38.101-1, June 2019.

[5] 3rd Generation Partnership Project (3GPP). System architecture for the 5G System (5GS). Technical specification TS 23.501, March 2019.

[6] 3rd Generation Partnership Project (3GPP). 5G; NG-RAN; F1 signalling transport. Technical specification TS 138.472, September 2020.

[7] 3rd Generation Partnership Project (3GPP). 5G; NG-RAN; NG Application Protocol (NGAP). Technical specification TS 138.413, Jenuary 2020.

[8] 3rd Generation Partnership Project (3GPP). 5G; NG-RAN; NG signalling transport. Technical specification TS 138.412, September 2020.

[9] 3rd Generation Partnership Project (3GPP). Technical Specification Group Radio Access Network; NG-RAN; Xn signalling transport. Technical specification TS 138.422, March 2020.

[10] Tara Ali-Yahiya. *Understanding LTE and its Performance*. Springer-Verlag New York, 1st edition, 2011. ISBN: 978-1-4419-6457-1.

[11] M. Allman et al. TCP Congestion Control. RFC 2581, IETF Network Working Group, April 1999.

[12] Amer P. et al. Load Sharing for the Stream Control Transmission Protocol (SCTP). Internet-Draft (Intended status: Experimental), IETF Network Working Group, June 2016.

[13] 5G PPP Architecture Working Group. View on 5G architecture. White paper, February 2020.

[14] ITU-R. IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond. Recommendation M.2083-0, Semptember 2015.

[15] ITU-R. Minimum requirements related to technical performance for IMT-2020 radio interface(s). Report M.2410-0, November 2017.

[16] Iyengar J. R. and others. Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths. *IEEE/ACM Transactions on Networking*, 14(5):951–964, 2006. doi:10.1109/TNET.2006.882843.

[17] Joe Wilke, Ericsson. 5G Network Architecture and FMC. `https://www.itu.int/en/ITU-T/Workshops-and-Seminars/201707/Documents/Joe-Wilke-%205G%20Network%20Architecture%20and%20FMC.pdf`, 2017. Visitato il 03/12/2020.

[18] Juha Salmelin, Esa Metsälä. *Mobile Backhaul*. Wiley, 2012. ISBN: 9781119974208.

[19] Ong L. et al. Framework Architecture for Signaling Transport. RFC 2719, IETF Network Working Group, October 1999.

[20] Mathis M. et al. Stream Control Transmission Protocol. RFC 2018, IETF Network Working Group, October 1996.

[21] Tuexen M. et al. Authenticated Chunks for the Stream Control Transmission Protocol (SCTP). RFC 4895, IETF Network Working Group, August 2007.

[22] Georg Mayer. RESTful APIs for the 5G Service Based Architecture. *Journal of ICT Standardization*, 6(1):101–116, 2018.

[23] Stewart R. et al. Stream Control Transmission Protocol (SCTP) Partial Reliability Extension. RFC 3758, IETF Network Working Group, May 2004.

[24] Stewart R. et al. Stream Control Transmission Protocol. RFC 4960, IETF Network Working Group, September 2007.

[25] Stewart R. et al. Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. RFC 5061, IETF Network Working Group, September 2007.

[26] Daniele Rossi. Implementazione Cloud del piano di controllo della rete 5G. Tesi di Laurea Magistrale, Alma Mater - Università di Bologna, Dicembre 2020.

[27] Seth T. et al. Performance Requirements for Signaling in Internet Telephony. Internet-Draft, IETF, November 1998.