Università di Pisa

Facoltà di Ingegneria

Corso di Laurea in
Ingegneria Telecomunicazioni

Tesi di Laurea Specialistica

Studio del protocollo SIP in presenza di collegamenti satellitari

Relatori

Prof. Michele Pagano

Ing. Rosario Garroppo

Ing. Stella Spagna

Candidato

Filippo Venturini

Anno Accademico 2009/2010

TITOLO

Studio del protocollo SIP in presenza di collegamenti satellitari

INDICE

Introduzione

Il protocollo SIP(Session Initial Protocol) rappresenta una delle possibilità per la segnalazione per tecnologie VOIP. Questo lavoro di tesi va a completare quanto svolto da me, per la laurea di I livello sul piano dati. L'obbiettivo d questo lavoro è stato quello di approfondire la conoscenza del simulatore Ns2 utilizzato per le simulazioni, capire la struttura del SIP in tutte le sue componenti e riportare queste conoscenze su uno scenario , in modo da poterne comprendere le prestazioni. Le difficoltà incontrate sono state numerose, soprattutto per la struttura di Ns2 che non mi ha permesso di modellare lo scenario secondo le mie esigenze, e per la scarsa documentazione sul simulatore e sul modulo SIP di Ns2 sviluppato da Rui Prior. Nonostante ciò il lavoro è stato svolto al meglio delle possibilità, e con buoni risultati. L'obbiettivo di questo lavoro di tesi , è stato quello di analizzare le prestazioni del protocollo SIP utilizzato per instaurare, modificare e abbattere sessioni multimediali, su link satellitari ad elevato ritardo. L'elaborato di tesi è organizzato in maniera seguente: il primo capitolo introduce gli aspetti più importanti del protocollo SIP, il secondo descrive il modello di simulazione utilizzato,nel terzo capitolo viene descritto lo scenario simulato con Ns2,e il quarto capitolo descrive i risultati ottenuti nei vari casi persi in analisi.

1 Il protocollo SIP

1.1 Panaromica sul SIP

La telefonia su protocollo di rete(VOIP), è la tecnologia di telecomunicazioni emergente, in una rete mondiale che vede sempre più convergere la infrastruttura dati con le rete di telecomunicazioni, quando fino a qualche anno fa il mondo dati e quello voce erano nettamente separati. L' accesso ad una rete mondiale, ad un basso costo e con elevata qualità, è una condizione essenziale, per lo sviluppo economico globale, e l' aumentare delle quantità di informazione in formato digitale, ha fatto si che sia aumentato sempre di più l'interesse verso la trasmissione di informazioni su reti , come per esempio quella IP. Le reti di base per voce e dati, sono state rimpiazzate da soluzioni ad alta affidabilità capaci di trasportare voce, dati,video e trasmissioni multimediali. Inoltre le soluzioni proprietarie, che all' alba di questa tecnologia , erano state messe in campo, sono state messe da parte, a favore di un' ambiente di sviluppo seppur standardizzato maggiormente inter-operabile. Sebbene per i grandi carrier, il traffico voce rappresenti ancora un grossa fetta dei guadagni, nei mercati maturi(Europa-occidentale,Nord-America,Giappone),la
crescita del mercato voce si attesta attorno al 3-4%, mentre quello dati sta vivendo un crescita molto forte

approssimativamente attorno al 30%. In questo scenario di mercato, l' idea di trasportare la voce sulla rete dati, ha riscontrato sempre più interesse da parte degli operatori del settore, disegnando uno scenario che vede convergere audio, video e dati su un'unica tipologia di rete. Il VOIP ha trovato terreno fertile nella sua crescita, grazie ad alcune caratteristiche tecniche che lo hanno reso particolarmente appetibile:

- Migliore gestione delle risorse
- Unica struttura di rete per voce e dati
- Possibilità di sviluppo di applicazioni per funzionalità Web e Voce

Lo sviluppo di questa tecnologia è stata accompagnata dagli istituti preposti con lo sviluppo di alcuni standard , il cui obbiettivo era di fornire una definizione precisa di tutti gli aspetti tecnici che riguardavano questa tecnologia. Il primo ente a fornire uno standard è l'ITU (International Telecomunication Union), che nel novembre del 1996 propone la sua soluzione per sistemi di comunicazione multimediale a pacchetto sotto il nome di H.323, dove vengono descritti i protocolli per la segnalazione e il controllo della chiamata. Ma questa soluzione presenta limiti evidenti dovuti a gli elevati costi degli elementi architetturali e ai lunghi tempi di ammortizzazione dei costi, e cosi si coglie l' esigenza di creare una soluzione più modulare, che trova vita nel SIP.

Nel marzo del 1999, il gruppo mmusic del IETF e poi aggiornato in seguito nel 2002 nel gruppo sip, viene stilata la RFC 3261 che specifica il protocollo SIP(Session Initial

Protocol). Il SIP è il protocollo di controllo sviluppato a livello di applicazione, che permette di instaurare, modificare e abbattere chiamate o sessioni multimediali. Questo protocollo si presenta molto leggero (almeno nelle prime versioni), capace di garantire affidabilità della segnalazione indipendentemente dal protocollo di trasporto utilizzato (TCP o UDP). Essendo stato sviluppato dal gruppo internet (IETF), molte delle funzionalità sono ereditate da protocolli già esistenti come HTTP 1.1.Esso si basa su un approccio di comunicazione client-server(semantica Request/Response) secondo il quale ad ogni richiesta corrisponde una risposta, utilizza un interpretazione testuale dei messaggi con codifica ascii del carattere e l'utilizzo di codici di risposta propri dell'http. La semplicità di questo protocollo si riscontra in una fase di setup molto veloce, infatti in un solo RTT(Round Trip Time) si riesce a combinare la ricerca della posizione dell' utente, l' invito,e la descrizione delle caratteristiche della chiamata. Il SIP affronta solo il problema di attivare/disattivare una sessione multimediale, le restanti funzionalità (Qos, sicurezza ecc.) sono affidati ad altri protocolli già definiti da IETF.

Ecco una lista dei protocolli impiegati per le altre funzionalità:

-RSVP Resource reservation protocol (RFC 2205) per la prenotazione delle risorse di rete

-RTP Real Time Protocol (RFC 1889/RFC 3550) per il trasporto di informazioni *real time* e riscontro al trasmettitore sulla Qos a livello di rete osservato dal ricevitore

-RTSP Real Time Streaming Protocol (RFC 2326) per il controllo del trasporto di streaming di dati multimediali

-SDP Session Description Protocol (RFC 2327) per la descrizione di sessioni multimediali

Un'altra caratteristica importante è la *personal mobility* , infatti come per l' e-mail nel SIP l' identificativo è dell' utente e non del terminale, quindi l' utente può accedere al servizio da terminali diversi oppure associarlo a diversi terminali con funzioni diverse(es:pc,telefono fisso, telefono SIP ecc.).

Nei successivi paragrafi verranno presentati tutti gli elementi e le procedure che costituiscono questo protocollo

1.2 Funzionalità del protocollo

Per l'instaurazione,la modifica e la terminazione delle comunicazioni multimediali, sono previste dal protocollo cinque funzionalità:

1. User location : determinazione del sistema terminale da usare per la comunicazione
2. User capabilities : determinazione dei media e dei loro parametri descrittivi da utilizzare durante la comunicazione, il sip trasporta sostanzialmente delle informazioni che sono codificate su SDP.
3. User availability : determinazione della volontà del chiamato a voler partecipare alla comunicazione,

determina in altri termini lo stato di un utente (presence service).
4. Call setup : squillo e instaurazione della chiamata in entrambi i lati della comunicazione.
5. Session Management: gestione della sessione, che prevede la modifica dei parametri per una sessione in corso, invocazione di servizi per conto dell'utente e terminazione della sessione.

1.3 Architettura del SIP

In questo paragrafo descriveremo l'architettura SIP cosi come è definita dalla RFC 3261. Gli elementi base si dividono in:

- **User Agent Client** (UAC): entità logica lato cliente che genera e trasmette una nuova richiesta ed utilizza una macchina a stati finiti del terminale.
- **User Agent Server** (UAS): entità logica lato server che genera e trasmette una risposta alla ricezione di una richiesta.
- **Proxy Servers**: è un server di rete che ha funzione di routing a livello applicativo determinando sostanzialmente il next hop (UAS o Proxy) a cui inoltrare la richiesta, può in alcuni casi generare risposte definitive sostituendosi al UAS.

Questi ultimi a loro volta si dividono in 3 categorie:

- **Stateless**: sono quei server che non hanno stato ovvero una volta inoltrata la richiesta, non mantengono più l'informazione cancellando la memoria, l'approccio stateless garantisce l'affidabilità a livello applicativo

solo end-to-end ovvero tra UAC e UAS, è l'UAC che attiva un timer allo scadere del quale se non ha ottenuto la risposta rinvia la richiesta.

- **Stateful**: sono quei server che mantengono le informazioni raccolte nelle richieste e nelle risposte elaborate e li utilizzano per la generazione di messaggi successivi, tra queste informazioni sono presenti i timer per la richiesta che è stata inoltrata, in questo caso l'affidabilità è hop-by-hop, infatti la ritrasmissione sarà gestita dal proxy e non dal UAC. Grazie alla memorizzazione delle informazioni gli stateful possono implementare procedure per l'autenticazione dell'utente che ha trasmesso la richiesta proprio perché mantengo uno stato della richiesta.

- **Transaction Stateful**: il proxy mantiene le informazioni riguardanti la transizione per una durata tale da consentire la conclusione della transazione in atto.

Da questi elementi base derivano altri elementi come:

- **B**ack to **B**ack **U**ser **A**gent (B2BUA): entità logica "bifronte" formata dall'accoppiamento di due UA in modalità "back-to-back", ossia "spalla a spalla". Nel B2BUA, i messaggi SIP in arrivo verso uno dei due UA rilanciano messaggi SIP in uscita dall'altro UA. Il senso di questa operazione è quello di mantenere separati i dialoghi da un lato e dall'altro dell'interfaccia costituita dal B2BUA stesso, ad esempio per implementare servizi di chiamata anonima, oppure per modificare parti del messaggio SIP che non dovrebbero

essere modificabili se non dagli estremi in comunicazione.

- **Registrar server**: è un UAS che ha il compito di accettare le richieste di registrazione fatte dagli utenti, infatti per iniziare una sessione, le entità SIP devono conoscere l'host corrente attraverso cui è raggiungibile l'utente di destinazione. Gli elementi di rete SIP, per "trovare" l'utente, consultano un servizio astratto di localizzazione ("Location Service"), responsabile del binding degli indirizzi relativi ad un particolare dominio. Il processo di Registrazione prevede l'interazione con il Location Service di un dominio, per creare la mappatura tra la SIP URI registrata e un insieme di "Contact Address" definiti dall'utente. La procedura consiste nell'invio della richiesta REGISTER ad un particolare UAS, detto "Registrar", che agisce da "front-end" per le registrazioni relative ad un dominio. Il Registrar è poi consultato dal Proxy Server responsabile per l'instradamento nel particolare dominio. Occorre sottolineare che il Registrar e il SIP Proxy rappresentano entità logiche che, fisicamente,possono coesistere in un solo elemento di rete.

- **Gateway SIP**: è un apparato che serve ad interfacciare utenti SIP ad altri utenti che usano tecnologie diverse dal SIP.

- **Redirect server**: è un particolare UAS con il compito di determinare e comunicare a chi gli ha inoltrato la richiesta il next hop server a cui ridirigere tale richiesta, anche in questo caso tale server utilizza il

servizio di location service di cui abbiamo parlato in precedenza.

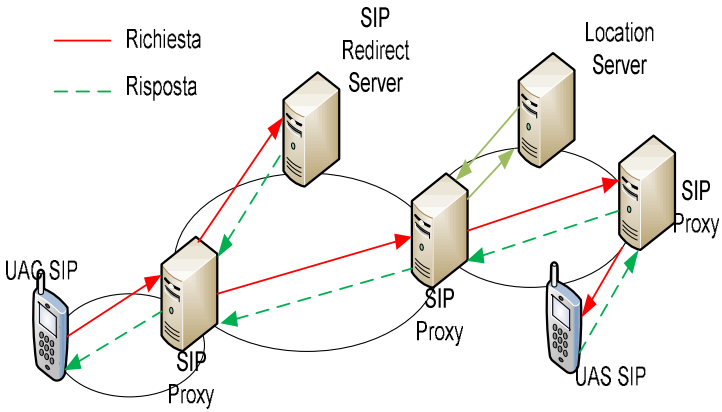

Figura 1.1- Elementi dell'architettura SIP

1.4 Indirizzi SIP

Gli indirizzi SIP presentano una forma del tutto simile a quella utilizzata per la posta elettronica e per altri servizi Internet, quindi segue la definizione di **URL** (Uniform Resource Locators) data nella RFC 1738.

La forma degli URL è stata definita in maniera tale da poter rappresentare un elevato numero di protocolli e di tipologie di risorse disponibili, ed è del tipo:

scheme:specifier

come ad esempio è *http://www.unipi.it* .In questo esempio parola *http* rappresenta il campo *scheme* che in questo caso è il protocollo utilizzato,cioè http,e a seguire troviamo il campo *specifier* ,che contiene il nome del dominio *www.unipi.it.* Il protocollo SIP per le sue caratteristiche legate alla mobilità, di solito fa riferimento agli **URI**(Uniform Resource Identifier), in quanto un indirizzo SIP non è vincolato ad una singola macchina(quindi ad un solo indirizzo IP), ma è associato ad un' entità logica che si muove in diversi punti della rete cambiando la sua posizione geografica.

Ecco il formato del SIP **URI:**

sip:user:password@host:port;option

dove SIP indica il protocollo del servizio a cui si riferisce l'indirizzo , *user* è l'utente del servizio*, password* è l'eventuale password per l'accesso al servizio, *host* è il terminale usato dall' utente per accedere al servizio, e *port* specifica il numero di porta con il quale connettersi per contattare l' utente. Dopo il campo port, possono essere inseriti dei campi opzionali (preceduti dal punto e virgola)per specificare , per esempio, il tipo di terminale o il protocollo di trasporto da usare per la trasmissione dei messaggi di segnalazione. Un importante esempio di flessibilità di questa tipologia di indirizzi, è il seguente:

sip: +5-223-575-2343@gateway.com;user=phone

Come si nota chiaramente il tipo di indirizzo contenuto è del tipo E.164(rete telefonica classica), l' *host* è un gateway verso

una rete PSTN e nel campo opzionale viene indicato che l' utente è un telefono. Da questo esempio si nota come il protocollo SIP sia capace di riportare nel suo formato di indirizzamento anche gli indirizzi E.164, cioè i numeri telefonici utilizzati nelle reti PSTN, favorendo cosi l'integrazione tra queste due tecnologie.

1.5 Metodi SIP

I metodi sono una delle componenti più importanti del protocollo SIP, infatti sono considerati come dei "verbi" da utilizzare per richiedere un'azione specifica da far prendere ad un altro UA o Server. Nella RFC 3261 sono stati definiti 6 metodi, che garantiscono le funzionalità base dei servizi SIP e sono:

- INVITE
- REGISTER
- BYE
- ACK
- CANCEL
- OPTIONS

Tutti i metodi usano dei messaggi formati da una (linea di inizio), uno o più campi di intestazione (*header*), una linea vuota che indica la fine dei campi dell'intestazione e un campo dati opzionale (*body*).

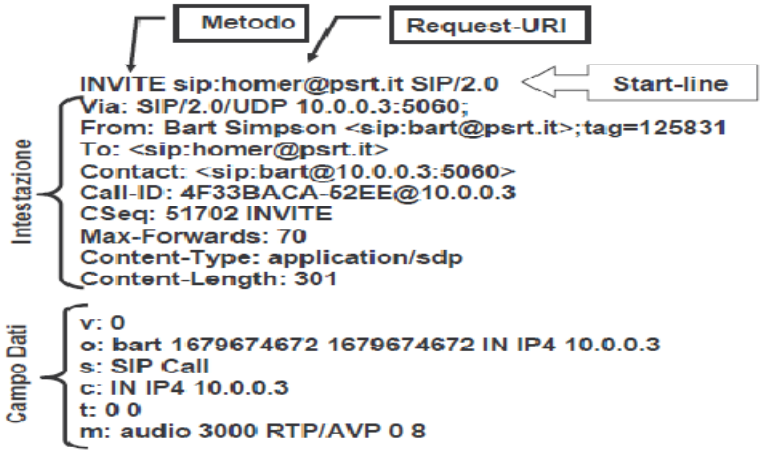

Figura 1.2- Formato del messaggio relativo al metodo INVITE

Un proxy non deve comprendere tutti i metodi definiti, in quanto il suo compito è quello di procedere con l' inoltro di questi messaggi ai server successivi, e in questa maniera è possibile introdurre nuovi metodi senza dover aggiornare i nodi intermedi (*server SIP*) , ma solo modificando gli UA che ne usufruiscono. Questa è un ulteriore prova della flessibilità di questo protocollo, e della sua modularità. Nei prossimi paragrafi verranno presentati in maniera più approfondita i metodi sopraelencati.

1.5.1 Invite

Il metodo *INVITE* è usato per invitare un utente o un servizio a partecipare ad una sessione. Il corpo del messaggio di questo comando contiene una descrizione della sessione alla quale il chiamato è invitato, la descrizione è fatta utilizzando un protocollo apposito :SDP (Session Description Protocol). Le risposte ad un messaggio *INVITE* devono essere sempre riscontrate con un metodo ACK. Se il messaggio *INVITE* non contiene informazioni SDP, queste devono essere inglobate nel metodo *ACK*. In questo caso però , se il chiamato non supporta le caratteristiche dei media indicate nel messaggio *ACK*, deve richiedere in modo esplicito la chiusura della sessione che si può ritenere già stabilita. La sessione multimediale si considera stabilita dopo lo scambio dei messaggi *INVITE, 200 OK* e *ACK*. Durante la procedura di instaurazione, lo UAC trasmette un messaggio di INVITE, creando un identificatore unico della sessione, indicato nel campo Call-ID dell' intestazione,e serve per individuare i messaggi associati alla particolare sessione. Nello stesso tempo viene inizializzato un contatore Cseq a valori interi, e questo viene incrementato di un unità ogni volta viene formulata una richiesta all'interno della stessa chiamata.

I campi To e From(vedi fig.2) dell'intestazione dei messaggi INVITE contengono le informazioni sugli indirizzi del chiamato e del chiamante, inoltre è presente un etichetta (From Tag), che serve ad individuare in modo univoco un dialogo all' interno di una sessione. Una volta stabiliti, in tutti i messaggi scambiati tra UAC e UAS saranno sempre inseriti i valori di *From tag* , *To tag* e *CALL-ID*, che identificheranno in maniera univoca il dialogo e la sessione. Un nuovo messaggio di

INVITE trasmesso nello stesso dialogo avrà gli stessi valori di *From Tag,To Tag* e *CALL-ID* del messaggio di INVITE iniziale, ma sarà contraddistinto da un *CSeq* differente, in questa maniera lo UAS sarà in grado di distinguere tra un nuovo INVITE e uno semplicemente ritrasmesso. Gli INVITE possono essere ritrasmessi sia per modificare i media coinvolti nella sessione , sia semplicemente per mantenere attiva la comunicazione e le relative informazioni di stato, nel caso in cui nella sessione ci siano dei periodi d inattività che potrebbero far supporre la sua interruzione per motivi sconosciuti. Come si può notare dalla figura , altri campi rivestono un ruolo importante e sono presenti anche in altri metodi diversi dall' INVITE.

Il campo *VIA* è utilizzato per registrare il percorso fatto dalla richiesta SIP, in modo tale che la relativa risposta possa seguire lo stesso percorso. Il primo indirizzo inserito nel campo *VIA* è quello dello UAC che genera la richiesta, poi ogni proxy che effettua l'inoltro aggiunge il proprio indirizzo all' inizio di questo campo, in questo modo quando la richiesta arriva allo UAS,e questo sarà in grado di inviare la risposta al nodo giusto. Un altro campo che merita attenzione è il *CONTACT* ,che contiene l' URI che identifica il terminale che ha trasmesso la richiesta o quello a cui la richiesta è stata trasmessa, a seconda se il campo si trova in un messaggio di richiesta o risposta. Lo scambio di questa informazione attraverso il campo CONTACT permette agli estremi della comunicazione UAC e UAS, di poter successivamente scambiare messaggi in maniera diretta , senza passare per

proxy intermedi. Concludiamo con il campo *Max-Forwards*, che ha una funzione simile a quella del TTL dell'intestazione del pacchetto IP. Il valore di questo campo viene decrementato di una unità da ogni proxy che inoltra il messaggio. Nel caso in cui un proxy riceva un messaggio con questo campo pari a zero, procederà allo scarto e restituirà al terminale che ha generato la richiesta un risposta del tipo *483 Too Many HOPS.* Nella RFC 3261 il valore di partenza di questo campo è pari a 70.

1.5.2 Ack

Questo metodo conferma che lo UAC ha ricevuto una risposta definitiva ad una richiesta di *INVITE.* Il corpo del messaggio ACK contiene la descrizione SDP della sessione, nel caso in cui il relativo messaggio di INVITE sia stato trasmesso senza informazioni nel campo dati. Il campo dati di un *ACK* non può essere utilizzato per cambiare i parametri della sessione stabiliti nel messaggio di *INVITE* .Nei messaggi ACK , il campo CSeq non viene incrementato, ma riporta lo stesso valore numerico del relativo *INVITE,* in questo modo lo UAS può associare il riscontro *ACK* al coretto *INVITE*.

1.5.3 Options

Questo metodo permette ad un chiamante di conoscere i metodi e le caratteristiche dei media supportate dai terminali o da un server , prima di instaurare la sessione. In questo modo il chiamante farà delle richieste che sicuramente il chiamato sarà

in grado di soddisfare, dato che se ne è accertato in una fase precedente.

1.5.4 Cancel

Il metodo *CANCEL* è usato per cancellare una richiesta pendente individuata dai campi *CALL-ID*, *TO TAG*, *FROM TAG* e *CSeq,* senza influenzare le altre richieste pendenti o completate. Questo tipo di metodo è tipicamente utilizzata per cancellare richieste di INVITE, e può essere formulata da uno UA o da un proxy in ogni momento, a condizione che si abbiano le informazioni necessarie per individuare in modo univoco la richiesta da cancellare.

1.5.5 Register

Il metodo *REGISTER* è utilizzato per notificare alla rete SIP il loro attuale punto di contatto, inteso come URI SIP (o indirizzo IP), e l' insieme di indirizzi SIP associati a quel particolare punto di contatto. La registrazione del terminale non è necessaria per abilitare lo UA ad utilizzare i server SIP per le chiamate in uscita, ma risulta fondamentale per ottenere il corretto instradamento delle chiamate in ingresso da parte dei proxy del dominio. Lo UA invia la richiesta all'indirizzo multi cast "SIP.mcst.net"(224.0.1.75) oppure a quello del *registrar*, noto in quanto inserito in quanto inserito tra i parametri di configurazione dello UA. A seguito della trasmissione di un

REGISTER , uno UA può ricevere risposte di conferma,di ridirezione o errore.

Una particolare attenzione va data ad alcuni campi della richiesta. Il campo *To* contiene l'indirizzo dell'utente che si sta registrando, *From* contiene l'indirizzo dell'utente che sta effettuando la registrazione (esso è uguale al campo *To* nel caso in cui la registrazione venga fatta dallo stesso utente che si sta registrando),*Request-URI* contiene il dominio del registrar, *CALL-ID* identifica lo UA, *CSeq* permette di distinguere le richieste provenienti dallo stesso UA, *Contact* permette di specificare l'indirizzo a cui spedire tutte le richieste dirette all'indirizzo SIP specificato nel campo To della richiesta registrar. La durata di default di una registrazione è pari ad 1 ora.

1.5.6 Bye

Il metodo BYE è usato dai terminali , quando si decide di voler abbandonare la sessione. Questo tipo di richiesta , può essere generata solamente dai terminali coinvolti nella sessione, e non da proxy o terminali non coinvolti. Quando chi partecipa ad una chiamata si vede recapitato un BYE, deve cessare di trasmettere informazioni utente (video , dati e voce) verso chi ha generato il messaggio.

1.6 Risposte SIP

Una risposta SIP è un messaggio generato da uno UAS o da un Server, dopo aver ricevuto ed interpretato una richiesta. Il

contenuto informativo di una risposta, è contenuta nello Status Code e Reason Phrase(vedi fig.3). Il primo è un intero di tre cifre che indica il tipo di risposta in linguaggio macchina, il secondo invece serve per dare una breve descrizione testuale comprensibile all'essere umano. Si può notare dalla figura che i campi che identificano la *transaction* (*To, From,CALL-ID e CSeq*) sono copiati dal messaggio di richiesta. Le classi di risposta definite nel protocollo SIP sono sei , di cui cinque sono prese direttamente dal protocollo http. Nei prossimi paragrafi verranno mostrate tutte le classi di risposta ,e i messaggi più importanti di ciascun classe.

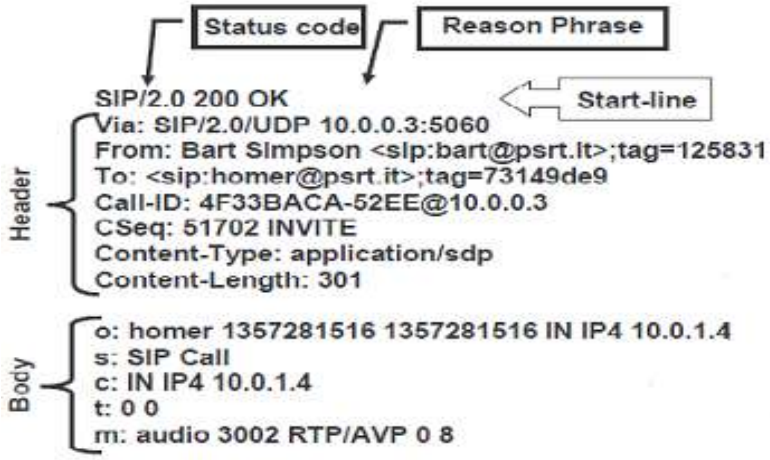

Figura 1.1-Formato dei messaggi di risposta

1.6.1 INFORMATIONAL (1xx)

Queste classe di risposte, serve ai server o ai proxy contattati per indicare che sta effettuando altre azioni e non ha ancora un risposta definitiva. In questo modo , il terminale prende tempo per elaborare la richiesta precedentemente ricevuta, ed evita inutili ritrasmissioni. Ecco alcune risposte appartenenti a questa classe.

- *100 Trying* Indica che la richiesta è stata ricevuta , ma che è ancora in fase di elaborazione
- *180 Ringing* Notifica che la richiesta è stata inoltrata all' utente(sta squillando il telefono).Se l'utente risponde immediatamente,non viene trasmessa questa risposta , ma direttamente il messaggio 200 OK
- *181 Call Is Being Forwarded* Notifica che la chiamata è stata inoltrata verso un altro utente
- *182 Call Queued* La richiesta è stata accodata e sarà elaborata appena finito con le altre in coda.
- *183 Session Progress* Questa risposta è usata per l'apertura di un canale logico per il trasporto di informazioni sullo stato della chiamata, quando la sessione è ancora nella fase di attivazione.

1.6.2 SUCCESS (2xx)

Le risposte appartenenti a questa classe indicano che la richiesta ha avuto successo oppure che è stata accettata.

- *200 OK* Questa risposta può essere usata nel SIP in due casi: per notificare che l'invito a partecipare ad una sessione è stato accettato e per indicare che la richiesta è stata accettata o semplicemente ricevuta
- *202 Accepted* Questo messaggio indica che lo UAS ha ricevuto e compreso la richiesta, ma che essa non può essere autorizzata o elaborata dal server

1.6.3 REDIRECTION (3xx)

Questa tipologie di risposte è utilizzato per fornire servizi di inoltro chiamata da parte di *Server Redirect.*

- *300 Multiple Choices* Questo messaggio contiene diversi campi Contact.La presenza di questi campi indica che il location service ha prodotto diversi utilizzati nella localizzazione dell' utente che si vuole contattare
- *301 Moved Permanently* In questo caso, il campo Contact contiene il nuovo indirizzo permanente del chiamato. Questo messaggio notifica lo UAC che l'indirizzo che riceve può essere registrato nella cache per essere usato per successive chiamate verso quell'utente
- *302 Moved Temporarily* In questo caso l' indirizzo riportato ha una validità limitata

1.6.4 CLIENT ERROR (4xx)

Le risposte che appartengono a questa classe sono date da un server o UAS per indicare che la richiesta non può essere soddisfatta , cosi come è stata presentata. In base al tipo di messaggio di risposta, lo UAC è invitato a modificare alcune parti della richiesta effettuata in precedenza.

- *400 Bad Request* Indica che la richiesta non è stata compresa dal server
- *401 Unauthorized* E' trasmesso da uno UAS per indicare all'utente che affinchè la sua richiesta venga elaborata , è necessaria l'autenticazione
- *482 Loop Detected* Indica che la richiesta è entrata in un loop, e che ha raggiunto di nuovo un server che precedentemente l' aveva inoltrata
- *486 Busy Here* Viene generato da uno UAS per indicare che non è in grado di accettare la richiesta di chiamata . Equivale al tono "occupato" nella rete PSTN

1.6.5 SERVER ERROR (5xx)

Le risposte di questa classe vengono generate dai server quando non sono capaci di elaborare le richieste pur essendo corrette. La ricezione di questo messaggio da parte di uno UAC implica che la richiesta inoltrata è formalmente corretta e che quindi non necessita modifiche per essere ripresentata.

- *500 Server Internal Error* Questa risposta è usata per comunicare che nel server si sono verificati degli errori che non permettono l'elaborazione della richiesta
- *501 Not Implemented* Il server non è capace di elaborare la richiesta in quanto non è supportata

1.6.6 GLOBAL ERROR (6xx)

Un server utilizza risposte di questa classe quando , dopo aver acquisito informazioni definitive su di un particolare utente , ha capito che la richiesta non potrà essere soddisfatta in nessun modo, da nessun server della rete.

- *600 Busy Everywhere* Viene generata solo se il server ha compreso che la richiesta non potrà essere soddisfatta da nessun posto della rete

1.7 Instaurazione di una sessione SIP

Dopo aver affrontato tutti gli aspetti teorici di questo protocollo è utile per capire meglio quanto spiegato nei paragrafi precedenti, vedere nella pratica cosa accade durante l'instaurazione di una sessione SIP. Vedremo per prima la procedura di registrazione che rappresenta una fase importante antecedente la sessione, e poi un esempio di apertura di una sessione SIP.

1.7.1 Registrazione

Nella figura sottostante(fig.4) sono mostrati i messaggi coinvolti nella fase di registrazione. Si nota che l'utente avente indirizzo *Watson@bell-tel.com* si registra al server SIP locale avente indirizzo *bell-tel.com*. Osservando il campo Via dell'intestazione possiamo notare che l'host usato da questo utente è *saturn.bell-tel.com*. Dal campo Contact invece possiamo trovare l'indirizzo che sarà utilizzato dall'utente, cioè *Watson@saturn.bel-tel.com:3890;transport=udp*. Si osserva che al server viene indicato l'indirizzo SIP associato all'utente registrato, ed in particolare , si osserva che l'utente si aspetta i messaggi di segnalazione sulla porta UDP 3890. Dal campo empire si deduce che la registrazione ha una durata di 2 ore(espresso in secondi).Dopo questa procedura , qualunque chiamata diretta a *watson@bell-tel.com* che arriva al server SIP bell-tel.com sarà inoltrata al nuovo indirizzo *Watson@saturn.bell-tel.com* sulla porta 3890 UDP .

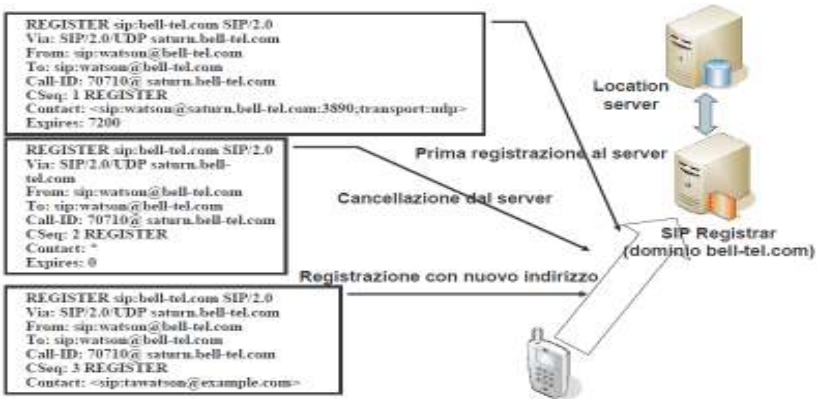

Figura 1.2-Messaggi scambiati durante la registrazione

Nel caso in cui , l'utente Watson voglia essere raggiunto in un altro posto in seguito ad uno spostamento, dovrà prima effettuare la cancellazione della registrazione precedente, e poi effettuare una nuova registrazione col suo nuovo indirizzo. La cancellazione avviene trasmettendo al server una richiesta uguale alla precedente tranne che per i campi *CSeq, Contact* e *Expires*. Il primo viene incrementato di uno, il secondo con il carattere asterisco indica che non ha più indirizzi dove può essere reperibile, il campo *Expires* settato a 0 indica chiaramente al server che l'utente *Watson@bell-tel.com* non deve avere registrazioni sul server. Con il successivo messaggio viene dichiarato il nuovo indirizzo.

1.7.2 Apertura sessione

Nella figura(fig.5) viene mostrato lo scambio di messaggi necessari per aprire una connessione tra il chiamante bia@iet.unipi.it ed il chiamato rossi@ele.unifi.it.

I passi della segnalazione sono i seguenti:

1. Lo UAC dell'utente con indirizzo bia@iet.unipi.it spedisce una richiesta SIP INVITE indicando come destinatario rossi@ele.unifi.it. La richiesta arriverà al proxy SIP del dominio ele.unifi.it
2. Il proxy SIP contatta il Location Service per individuare la posizione corrente di rossi@ele.unifi.it.

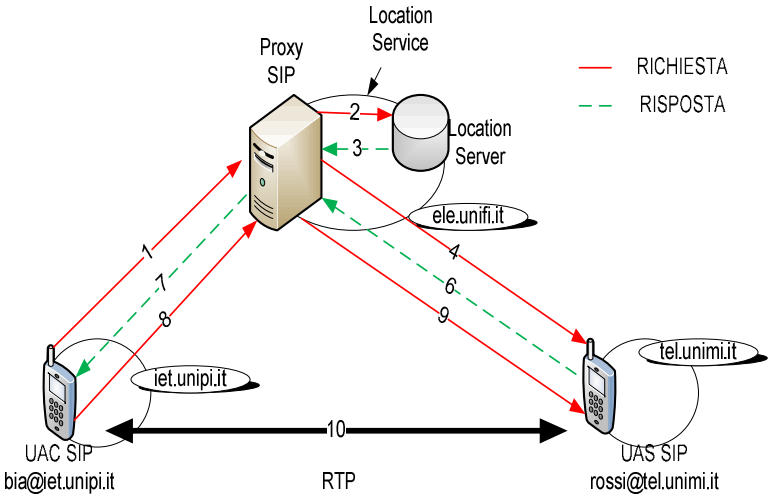

Figura 1.4-Chiamata mediante proxy

3. Il Location Service comunica al proxy SIP che l'utente richiesto è contattabile all'indirizzo rossi@tel.unimi.it.
4. Il proxy SIP inoltra la richiesta proveniente dal UAC verso la nuova posizione del chiamato, cioè invia una richiesta nella quale la prima riga è la seguente:INVITE rossi@tel.unimi.it.
5. Viene notificata la richiesta all'utente , cioè squilla il telefono o appare il messaggio di notifica sul terminale dell'utente chiamato
6. Viene trasmesso il messaggio di risposta *200 OK* che notifica al chiamante che il chiamato è pronto ad

iniziare la sessione SIP. Tale messaggio viene inviato al proxy SIP
7. Il messaggio di risposta viene inoltrato dal proxy SIP al chiamante
8. Il chiamante trasmette verso il proxy SIP il messaggio ACK
9. Il messaggio ACK viene inoltrato dal proxy SIP verso il chiamato
10. Inizia la comunicazione fra i due utenti

Dopo lo scambio dei media, se uno tra il chiamante e il chiamato decide di chiudere la sessione, deve inviare un *BYE* all' altro, e con la ricezione della conferma (messaggio *200 OK*) la sessione si chiude.

2 Il modello di simulazione

2.1 Simulatore NS2

Mi sembra necessario per comprendere cosa è stato fatto nel mio lavoro di tesi, conoscere lo strumento utilizzato per ottenere i risultati che saranno poi l'argomento delle conclusioni.

Il simulatore utilizzato è NS2 (sta per network simulator versione 2), nato come prima versione nel 1989 come variante del simulatore di rete REAL, progettato per studiare le dinamiche di flusso e gli schemi di controllo delle reti dati a commutazione di pacchetto. Il Network Simulator 2 è un simulatore ad eventi discreti(event-driven), cioè vengono eseguite delle azioni solo agli istanti in cui si presentano eventi che modificano lo stato del sistema. Il simulatore possiede una lista di venti marcati con l'istante di simulazione, il centro di elaborazione accede alla lista degli eventi attraverso uno scheduler , che gestisce le operazioni di inserimento e di estrazione dalla lista in base a criteri definiti . Ad ogni passo il centro di elaborazione del simulatore seleziona l'evento dallo scheduler, esegue le azioni associate a tale evento e genera eventualmente nuovi eventi che passa allo scheduler in modo che possano essere aggiunti alla lista , questo meccanismo viene avviato da un insieme di eventi decisi dall'utente.

I punti di forza del simulatore consistono nell'elevata modularità, disponibilità in rete di modelli di simulazione con relativa documentazione e costi ridotti in fase di sviluppo. D'altra parte sono emersi degli svantaggi nel suo utilizzo come la ridotta scalabilità, prestazioni in alcuni casi scadenti, presenza di porzioni di codice non commentati, non testati e incompleti. Ns2 da la possibilità di simulare molte tipologie di reti IP LAN e WAN grazie all'implementazione di protocolli di rete (MAC,routing e trasporto), modelli di sorgenti di traffico (CBR,FTP), meccanismi di gestione delle code (FIFO RED), protocolli e meccanismi wireless 802.11 in modalità sia ad-hoc che infrastructured.

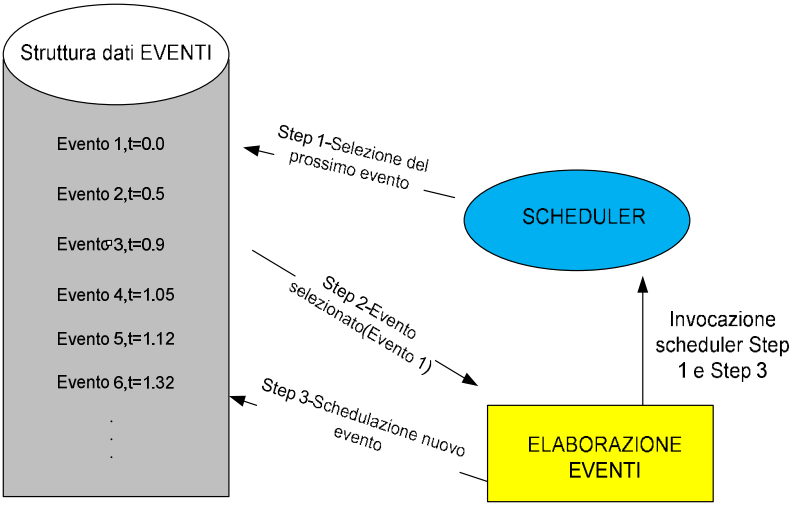

Figura 2.1-Gestione degli eventi in NS2

Per descrivere lo scenario di simulazione, gli elementi coinvolti e il loro comportamento si utilizzano:

- Il C++ come Back-End, per la definizione dei nuovi oggetti,protocolli e frame work, per la manipolazione dei pacchetti a livello di bit e per gestire strutture di dati complesse o voluminose
- OTCL come front-end, utilizzato per la descrizione dello scenario di simulazione, definizione di variabili temporanee e per la scrittura di codice che necessita di essere modificato frequentemente

Un concetto importante in Ns2 è la programmazione per oggetti, una rete è costituita da numerosi componenti di diversa natura che interagiscono tra di loro, e la programmazione orientata all'oggetto ci permette in maniera semplice di costruire e caratterizzare nuovi componenti di rete,collegamenti e protocolli.

Tutti gli oggetti derivano dalla classe TclObject che è la base per tutte le classi(vedi figura). Saranno brevemente descritte le classi principali , necessarie per effettuare una simulazione:

- La classe SIMULATOR è una classe fondamentale per NS2, perché permette di avviare l'intera simulazione. Con il comando *set ns [new simulator]* si crea una variabile per l'intera simulazione e con *$ns run* si avvia la simulazione

- La classe NODE implementa le funzionalità del protocollo IP, ovvero definisce l'indirizzamento,il routing e la consegna delle unità informative al protocollo di trasporto. Sono previsti due tipologie di nodi, unicast node si occupa di gestire pacchetti con un solo mittente ed un solo destinatario, mentre multicast node si occupa di gestire pacchetti con un solo mittente e più destinatari.

  Per creare l'oggetto NODE:
  set node1 [$ns node]
  per la creazione di topologie più grandi (N oggetti NODE) si può utilizzare un ciclo for:
  for {set i 0} {$i < N} {incr i} {
  set node$i [$ns node]
  }
- La classe LINK è l'oggetto che permette il collegamento tra i nodi. Per creare il collegamento fra i nodi specificando il tipo, la capacità, il delay e la disciplina di coda si utilizza il comando:
  $ns duplex-link $node1 $node2 10Mb 1ms DropTail

- La classe AGENT simula il comportamento del protocollo di trasporto di un nodo, si occupa della creazione e della rimozione delle unità informative. L' agent non gestisce dati reali ma solo la dimensione dell'unità informativa
- La classe APPLICATION emula le applicazioni più comuni e le caratteristiche di profili di traffico noti. I dati una vota creati dall'application vengono inviati all'oggetto agent tramite funzioni di interfaccia

le unità dati generate dall'oggetto application vengono passati all'oggetto. In figura possiamo cosi come è costituito lo stack protocollare in NS2
agent tramite funzioni di interfaccia.

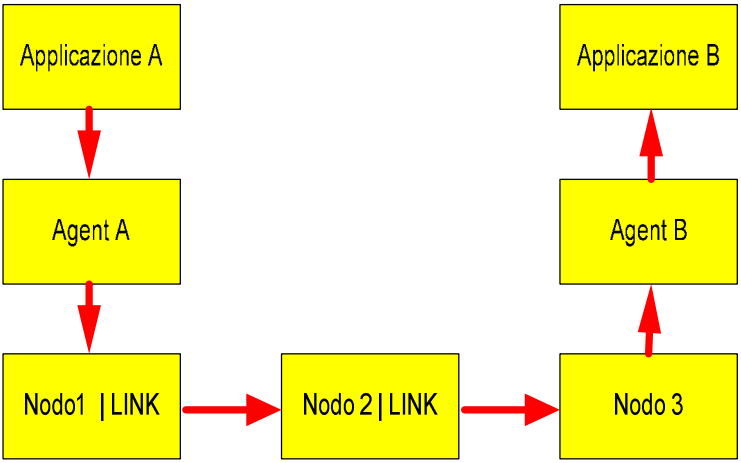

Figura 2.2-Struttura a strati di NS2

2.2 Livelli logici del SIP

Nel primo capitolo ho descritto l'architettura SIP in tutte le sue componenti e procedure. Per comprendere meglio l'obbiettivo del mio lavoro di tesi, e come il protocollo SIP è implementato su NS2,è necessaria una descrizione logica del SIP. Ciascuna entità SIP (ma non tutte)dal punto di vista logico può essere

vista , come formata da più livelli logici stratificati: livello di trasporto,livello Transaction e livello Transaction User (TU). Nei prossimi paragrafi saranno descritti uno ad uno i livelli logici che compongono il SIP.

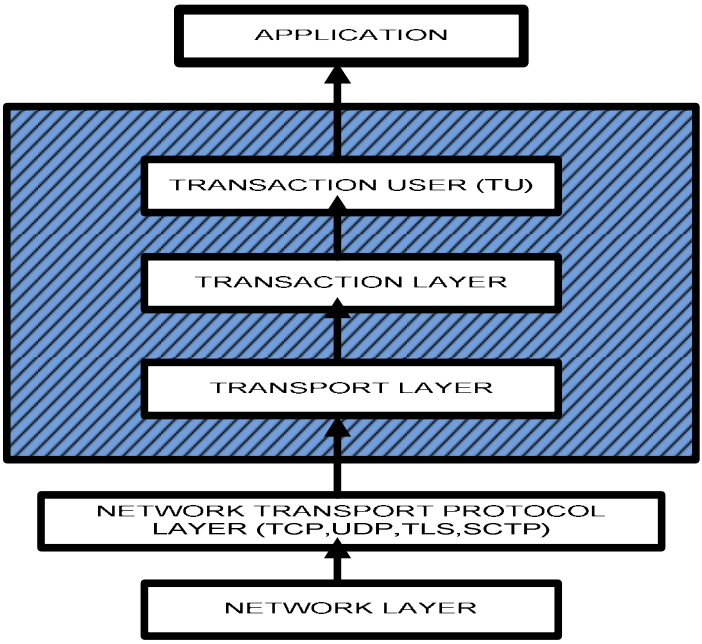

Figura 2.3-Struttura logica di un'entità SIP

2.2.1 Livello logico di trasporto

Il livello di trasporto si occupa della gestione della connessione, dell' inoltro delle richieste e delle risposte sulla rete. In particolare definisce come un client invia delle richieste

e riceve delle risposte, e in maniera duale il comportamento del server. Il livello di trasporto gestisce la richiesta/risposta con la relativa transazione, se il matching è riscontrato ,la richiesta è passata alla transazione , altrimenti viene inoltrata al core dell'entità SIP.

2.2.2 Livello Transaction

Il livello Transaction è costituito da un lato client e un lato server, e lo possiamo trovare all'interno degli User Agent e degli stateful proxy server. La parte client Transaction si occupa di ricevere una richiesta da un elemento che in quell'istante funge da client e fornire la richiesta al server Transaction. Inoltre è incaricato di fornire le risposte al TU,di generare l'ACK per ogni risposta definitiva come i 2XX response nel caso di una richiesta di INVITE e di filtrare ogni ritrasmissione o risposte non conosciute. Alla stessa maniera il server Transaction si occupa di ricevere le richieste dal livello di trasporto e passarle al TU o viceversa,di assorbire in caso di INVITE la richiesta di ACK per ogni risposta definitiva ad eccezione di una risposta 2XX. La risposta 2XX e il suo ACK ricevono un trattamento particolare, la risposta è ritrasmessa solo da un UAS, e il suo ACK generato solamente da l'UAC. Nei prossimi paragrafi, verrà descritto in maniera più particolareggiata il comportamento della macchina a stati del livello Transaction sia dal lato client che quello server, nel caso di metodi INVITE e nonINVITE. Per comprendere meglio

quanto verrà detto , è necessario introdurre i TIMER SIP. Nella tabella sottostante sono presentati i timer cosi come sono definiti nella RFC 3261.

TIMER	VALUE	MEANING
T1	500 ms	RTT Estimate
T2	4 s	The maximum retransmit interval for non-invite requests and INVITE responses
T4	5 s	Maximum duration a message will remain in the network
Timer A	Initially T1	INVITE request retransmit interval , for UDP only
Timer B	64*T1	INVITE transaction timeout timer
Timer C	>3min	Proxy INVITE transaction timeout
Timer D	>32 s for UDP	Wait time for response retransmits
Timer E	Initially T1	non –INVITE request retransmit interval UDP only
Timer F	64*T1	Non-INVITE transaction timeout timer
Timer G	Initially T1	INVITE response retransmit interval
Timer H	64*T1	Wait timer for ACK receipt
Timer I	T4 for UDP	Wait time for ACK retransmits
Timer J	64*T1 for UDP	Wait time for non-INVITE request transmits
Timer K	T4 for UDP, 0 for TCP/SCTP	Wait time for response retransmits

Figura 2.4-Tabella TIMER SIP [RFC 3261]

2.2.2.1 INVITE/non INVITE CLIENT TRANSACTION

La procedura per l' instaurazione di una chiamata prevede un three-way handshake, il client invia l'INVITE, il server invia la risposta,e il client riscontra la risposta con l'ACK. Quando il TU decide di inviare un INVITE, la macchina a stati entra nello stato iniziale "calling". Se è stato utilizzato un protocollo di trasporto non affidabile (per esempio UDP), viene inizializzato il timer A con una valore pari a T1. Allo scadere del timer A, se non è stata ricevuta alcuna risposta, la copia della richiesta viene ritrasmessa passandola al livello logico di trasporto, e il timer viene raddoppiato(Timer A=2*T1). Ogni volta che il timer A scade senza aver ricevuto una risposta ,avviene una ritrasmissione della richiesta e il timer viene reimpostato con un valore doppio del precedente. Per ogni transazione, il client transaction inizializza il timer B con un valore pari 64*T1 secondi , se allo scadere di questo timer si è ancora nello stato iniziale(stato "calling"), si passa l'informazione al TU passando allo stato "terminated " cancellando la transizione. Se quando siamo nello stato "calling" si riceve una risposta provvisoria si passa allo stato "proceeding"; in questo stato il client transaction disattiva le ritrasmissioni, in fatti se arriva una risposta provvisoria 1XX la passa direttamente al livello TU, restando in questo stato, mentre nel caso in cui giounga una risposta con codice compreso tra 300-699 si transita allo stato "completed". Giunti a questo punto il client transaction passa la risposta ricevuta al livello TU, generando un ACK che

tramite il livello di trasporto sarà passato per la trasmissione. Se arriva un risposta di successo 2XX, la macchina a stati passa allo stato “accepted” e la risposta viene notificata al livelllo TU. Quando si giunge allo stato “completed” viene inizializzato il timer D con un valore di 32 secondi nel caso di protocollo di trasporto UDP e 0 secondi nel caso di protocollo TCP. Questo timer serve ad attendere ritrasmissioni che nel caso di passaggio allo stato successivo “terminated” non verrebbero prese in considerazione. Nello stato “completed” ad ogni ricezione di una risposta 300-699 si invia un’ACK al livello di trasporto, e allo scadere del timer D si transita nello stato “terminated”, a questo stato si giunge dallo stato “calling”, “proceeding”e “completed” quando si riceve una risposta definitiva 2XX. Ora consideriamo il lato client transaction nel caso di richieste non-INVITE.

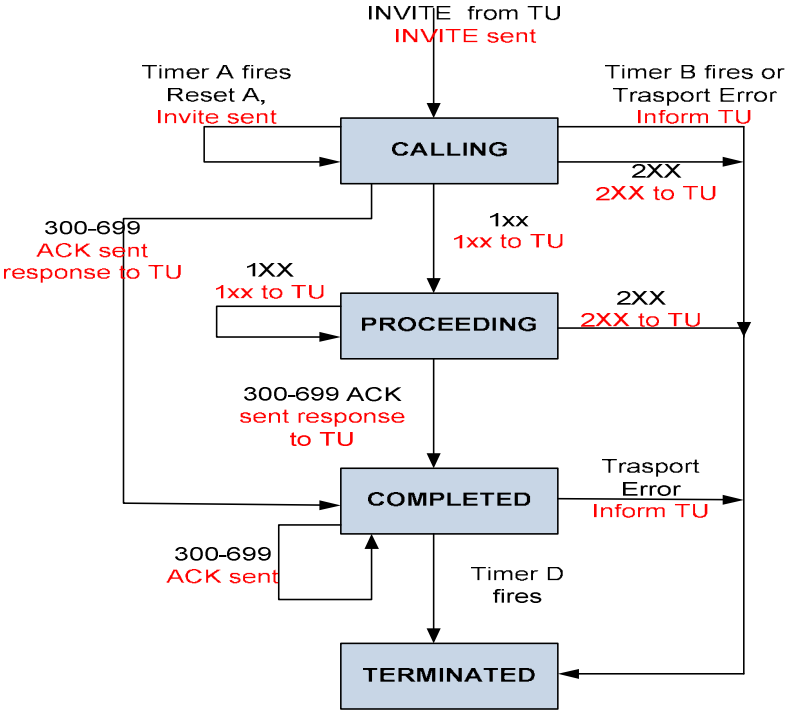

Figura 2.5-INVITE CLIENT TRANSACTION

Quando il TU inizia una nuova transizione con una richiesta che non sia un INVITE o un ACK si entra nello stato "Trying", viene settato il timer F (timeout) con valore di 64*T1 secondi e la richiesta viene passata al livello di trasporto per la trasmissione. Le ritrasmissioni in questo stato vengono regolate dal timer E, che inizialmente viene settato ad un valore pari a T1, se non si ottiene risposta alla sua scadenza, la richiesta viene ritrasmessa e il timer E viene portato ad un valore pari a min(2*T1,T2); l'incremento segue questa regola andando a raddoppiare il T1 per ogni ritrasmissione: min(4*T1), min

(6*T1, T2) ritrasmissione:min(4*T1,T2), e cosi via. Il valore di T1 è di 500 ms mentre il valore di T2 è 4 sec. Se siamo ancora nello stato "Trying" e il timer F scade si informa il TU e si transita allo stato "terminated". Se invece è stata ricevuta una risposta provvisoria 1XX, questa sarà passata al TU e si salta allo stato "proceeding";se invece è una risposta definitiva 200-699, la risposta sarà passata al TU transitando allo stato "completed".

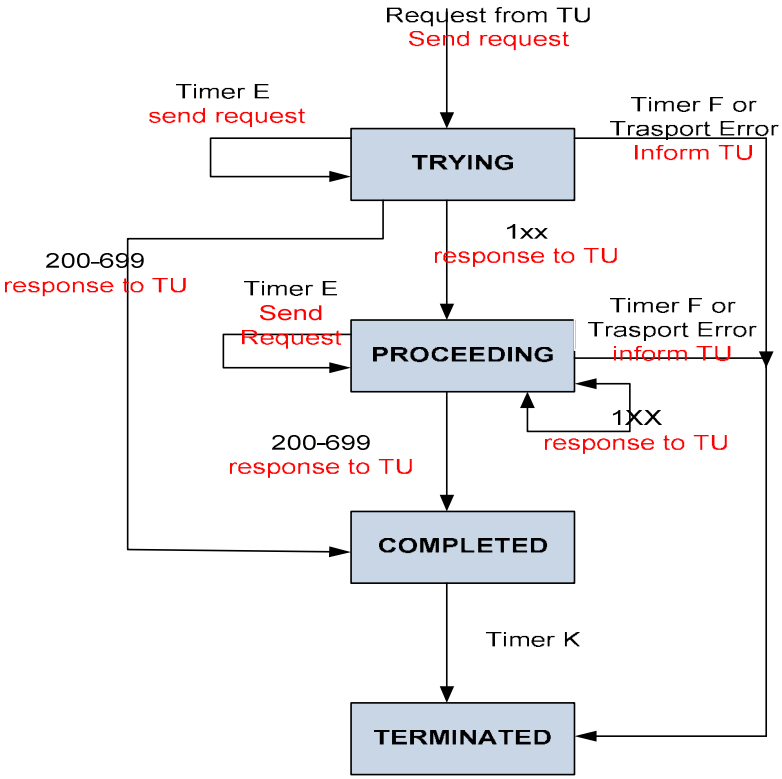

Figura 2.6-nnINVITE CLIENT TRANSACTION

Giunti allo stato “proceeding” ci si rimane in due casi, il primo è lo scadere del timer E a cui segue una ritrasmissione della richiesta tramite il livello logico di trasporto , il secondo è la ricezione di una risposta 1XX che viene passata al TU. Si passa allo stato “completed” se si riceve una risposta definitiva 200-699, con l’azione di notifica al TU. Nel caso ci si trovi nello stato “trying” o “proceeding”e scada il timer F o si riscontri un errore sul livello logico di trasporto, si passa allo stato “terminated” con la notifica al TU. Giunti allo stato “completed” viene settato il timer K(T4 secondi) , con il compito di riassorbire eventuali ritrasmissioni. Una volta scaduto il timer K, si passa allo stato “terminated” e la transazione viene cancellata.

2.2.2.2 INVITE/non INVITE SERVER TRANSACTION

Quando il TU riceve una richiesta, viene creata un’istanza del server transaction che si occuperà della consegna delle richieste e della trasmissione affidabile delle risposte. Alla ricezione di una richiesta INVITE si entra nello “stato proceeding”, e nel caso in cui il TU sarà in grado di dare una risposta definitiva o provvisoria entro 200 ms, viene generato un messaggio *100 Trying* . Questo messaggio ha la funzione di fermare il timer che il lato client ha inizializzato al momento dell’invio

dell'INVITE. Nello stato "proceeding" le risposte provvisorie 101-199 trasmesse dal TU vengono passate al livello logico di trasporto senza alcun passaggio di stato; se avviene una richiesta di ritrasmissione quando siamo ancora nello stato "proceeding", la copia della richiesta provvisoria più recente ricevuta dal TU viene passata al livello logico di trasporto per la ritrasmissione. La ricezione dal Tu di una risposta 2XX nello stato "proceeding" , provoca il passaggio della risposta al livello logico di trasporto e il transito nello stato "terminated" con la con la conseguente cancellazione della transazione. Se nello stato "proceeding", il TU passa una risposta con codice 300-699, la risposta viene inoltrata al livello di trasporto per la trasmissione e si salta nello stato "completed" con l'avvio del timer G impostato ad un valore di T1 secondi. Appena arrivati in questo stato viene settato il timer H con valore pari a 64*T1 secondi, la funzione del timer è quella di terminare le ritrasmissioni di tutte le transizioni, il suo valore è stato scelto come il timer B, tempo per il quale il lato client ritenta l'invio della richiesta. Per quanto riguarda il timer G , se questo scade la risposta è passata al livello di trasporto ed il timer verrà settato a min(2*T1,T2) con l 'incremento visto per il timer E fino ad un valore massimo che è dato dal timer H. Se mentre si è nello stato "completed" viene ricevuto un ACK, si passa allo stato "confirmed".

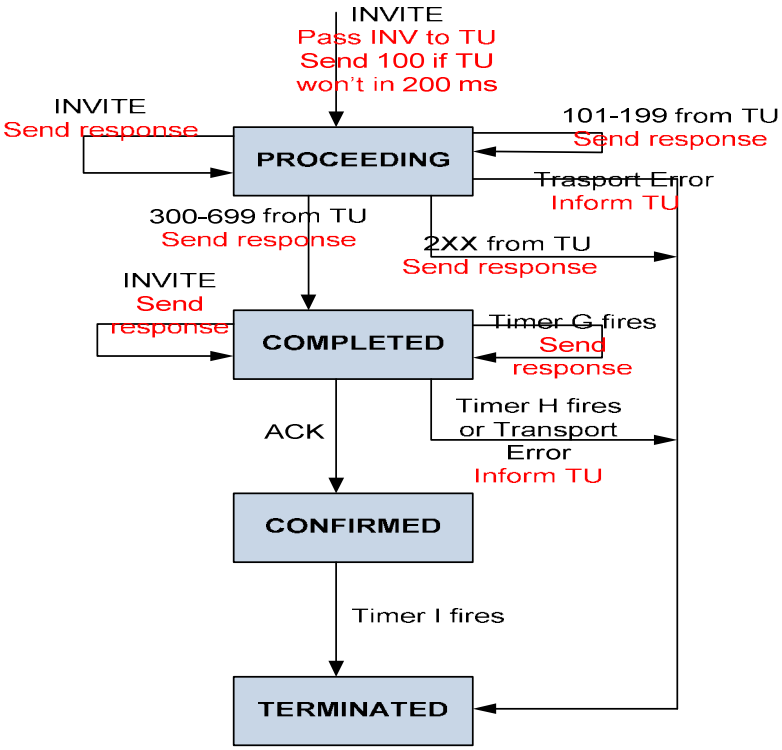

Figura 2.7-INVITE SERVER TRANSACTION

Lo scadere del timer H nello stato "completed" ci dice che l'ACK non è stato ricevuto, quindi si passa allo stato "terminated", con la notifica al livello TU. Il compito dello stato "confirmed" è quello di assorbire tutti gli ACK ritrasmessi. Se da TU si ottiene una risposta 2XX nello stato "proceeding", si passa allo stato "accepted" con l'invio della risposta al livello logico di trasporto. Nello stato "terminated"si arriva allo scadere del timer I che assorbe tutte le

ritrasmissioni. La macchina a stati per il non-INVITE server transaction è rappresentata in figura 2.7. La macchina viene istanziata dal TU, non appena si riceve una richiesta diversa da INVITE e ACK, e si entra nello stato "trying". Da qui si passa allo stato "proceeding" se si riceve dal TU una risposta provvisoria oppure allo stato "completed" se si riceve sempre dal TU una risposta 200-699 che verrà inoltrata al livello logico di trasporto. Dallo stato "completed" allo scadere del timer J, il cui compito è di assorbire le ritrasmissioni, si passa allo stato "terminated".

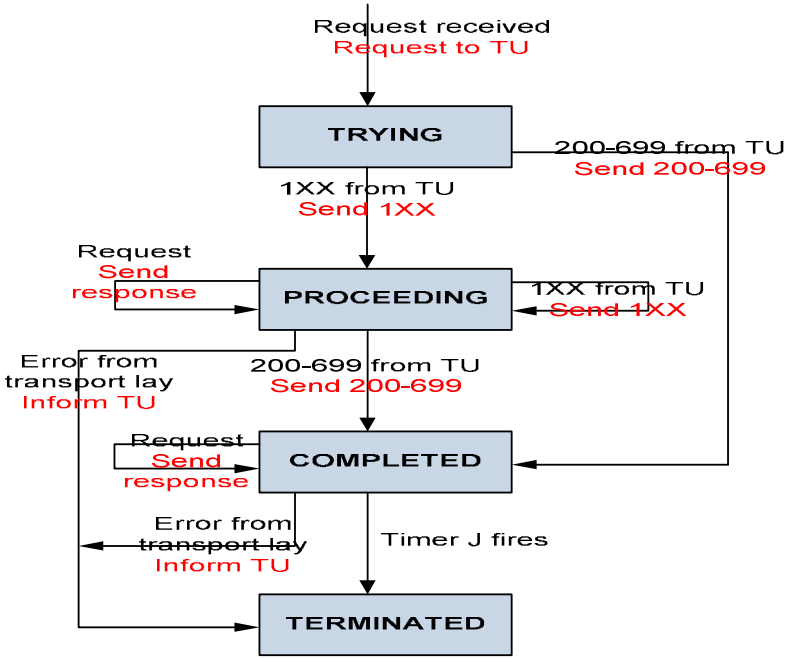

Figura 2.7-nnINVITE SERVER TRANSACTION

2.2.3 Transaction User (TU)

Il TU rappresenta il core di ogni entità SIP e può essere paragonato alla coscienza di una persona che decide quando e se effettuare una richiesta. E' proprio il TU ad inviare la richiesta SIP, creando l'istanza del client transaction e decidendo il protocollo di trasporto, l'indirizzo IP e la porta da utilizzare per l'invio. Il livello logico TU, si occupa anche di eliminare le transazioni al momento della ricezione della relativa risposta. Infatti quando il client cancella la transizione, richiede al server di fermare il processing, ritornando allo stato precedente all'inizio della transizione e generando un messaggio di errore per la specifica transizione(metodo CANCEL).

2.3 Meccanismi di Local Overload

In questo paragrafo verranno descritti i meccanismi di controllo dell'overload che verranno utilizzati nel corso delle simulazioni

2.3.1 Sistema una coda FIFO

Il primo sistema da noi preso in considerazione e la coda FIFO, la disciplina di coda piu semplice, dove il primo messaggio ad essere arrivato e anche il primo messaggio ad essere servito. Come tutti i buffer fisicamente realizzabili, la coda FIFO e caratterizzata da una dimensione massima oltre la quale i messaggi non potranno piu essere elaborati e andrà incontro ad una situazione di congestione. Per questo è stato pensato di

dividere il buffer in delle zone con l'intento di dare al server un segnale su quello che accade all'interno del buffer. Le zone del buffer sono marcate da due soglie:

• soglia alta *THHIGH* : ha la funzione di notificare che il numero di messaggi in accodamento sta saturando il buffer e che se non verrà preso un qualche provvedimento si andrà incontro alla totale saturazione del sistema. Nella nostra proposta è stato scelto di accodare i messaggi nonINVITE in modo da far concludere le sessioni instaurate, mentre e stato scelto di scartare i messaggi di INVITE che avrebbero avuto la funzione di aumentare il carico,dando vita alla instaurazione di nuove sessioni.
• Soglia bassa *THLOW:* ha la funzione di notificare che il numero di messaggi e sceso al di sotto del livello di guardia e che sostanzialmente si e fuori dalla situazione di overload, in questo caso tutti i messaggi verranno accodati nel buffer.

2.3.2 Sistema ad una coda con priorità

Il sistema ad una coda con priorità che sostanzialmente è stato il sistema preso in considerazione per le simulazioni del Local Overload di questo elaborato di tesi, ribadisce nuovamente il solito concetto di dare una priorità di elaborazione ai messaggi nonINVITE. La struttura del buffer è unica, l'accodamento dei messaggi è quindi misto. La disciplina di coda cambia a seconda dello stato in cui si trova il buffer passando da una disciplina FIFO in situazione di normalità ad una disciplina

LIFO quando siamo in stato di Overload. Anche in questo caso la variazione di stato è gestita dalle soglie THHIGH e THLOW:

• soglia alta THHIGH: ha la funzione di notificare che il buffer è in saturazione ed attiva il meccanismo di Local Overload, accodando il messaggio di INVITE in fondo al buffer, mentre se il messaggio in arrivo è un nonINVITE questo viene posto in testa al buffer in modo da poter esser elaborato per primo (disciplina LIFO).

• Soglia bassa THLOW: ha la funzione di notificare che si è ritornati ad una situazione di normalità, i messaggi vengono accodati secondo la disciplina FIFO.

2.4 Meccanismo TU

Il meccanismo TU valuta il traffico del server attraverso i due parametri presi in considerazione precedentemente ovvero mediante il calcolo del rate delle sessioni in fase di servizio (μs) e del Backload, la differenza consiste nelle variabili che entrano in gioco per il calcolo dei due parametri. Nel meccanismo TU il rate delle sessioni in fase di servizio è dato dal numero totale di messaggi di INVITE accettati in un certo istante temporale Tm. La differenza sostanziale con il rate calcolato mediante l'approccio Transaction consiste nel fatto che quando il traffico aumenta il meccanismo TU tende a sovrastimare il rate a causa del fatto che il tempo per servire un

INVITE aumenta determinando la ritrasmissione dell'INVITE stesso, ritrasmissioni di cui il TU tiene conto. Per quanto riguarda il calcolo del Backload nel meccanismo TU, è stato scelto di calcolare Nproc come il numero dei messaggi nonINVITE in coda sul numero medio di messaggi per sessione. Anche per il meccanismo TU sarà calcolata la stima del ritardo di coda del proxy (dq), come somma degli INVITE accettati e Nproc normalizzati a μs.

2.5 OC-PROXY

In questo paragrafo vengono descritte le funzioni che implementano il proxy OC , che presenta le soluzione di controllo dell'overload. Il proxy OC nel nostro scenario è il proxy BD, a cui fanno riferimento gli UA che ricevano le chiamate. Andiamo dunque ad analizzare le funzioni membro della classe oc-proxy, daremo uno sguardo alle funzioni membro che gestiscono il sistem a coda dell' oc-proxy, queste sono: recv, handle e checkQueue.

OCProxy::recv : la funzione *recv* fa un primo controllo sul messaggio arrivato, incrementando delle variabili che tengono conto della tipologia del messaggio, *inviteArrived_* per i messaggi INVITE e *nninviteArrived_* per i messaggi nn INVITE, successivamente fa un controllo tramite il metodo bind sullo script tcl per determinare a quale tipologia di sistema a coda si fa riferimento. Nel nostro caso valutando solo il caso del sistema ad una coda, viene attivata la funzione *checkQueue(p).*

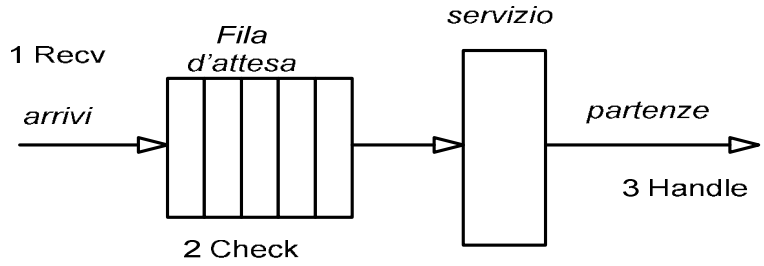

Figura 2.8-Sistama a coda OC-PROXY

OCProxy::checkQueue : la funzione *checkQueue* implementa il meccanismo di gestione per il sistema ad una coda FIFO, gli eventi che si possono verificare sono 4:

1. Buffer pieno: in questo caso viene settata una variabile di controllo *exceeded* ad 1 e viene richiamata la funzione **action0(p)** tramite la quale il messaggio viene scartato e vengono incrementati dei contatori *INVITEdrop_*, *nnINVITEdrop_* che quantificano il numero di INVITE/nnINVITE scartati. Adesso consideriamo i casi in cui il Buffer non è pieno ed introduciamo l' utilizzo di due soglie che ci permetteranno di gestire meglio la coda:
2. numero messaggi maggiore della *TH_HIGH_* : viene settata la variabile *invTHExceeded* ad 1 e viene richiamata la funzione **action1(p)** tramite la quale se il messaggio di richiesta da processare è un INVITE , questo viene scartato e viene incrementato il contatore *INVITEdrop_*, se il messaggio di richiesta è nnINVITE, questo viene solo accodato.
3. Numero messaggi al di sotto della *TH_LOW_*: la variabile di controllo *invTHExceeded* viene settata a 0 e viene richiamata la funzione **action2(p)** tramite la quale il messaggio da processare viene accodato

indipendentemente se questi è un INVITE o un nn INVITE.

4. Ultimo caso se il numero dei messaggi è tra la *TH_LOW* e la *TH_HIGH* : viene fatto un controllo sulla variabile *invThExceeded*, se questa è settata ad 1 viene richiamata la funzione **action1(p)** che abbiamo descritto sopra , se *invThExceeded* è settata a 0 si richiama la funzione **action2(p)** che abbiamo detto accoda il messaggio , indipendentemente se esso sia INVITE o nn INVITE.

OCProxy::checkQueue_prio: la funzione c*heckQueue_prio* implementa sostanzialmente la nostra proposta di Local Overload che consiste in caso di overload nel dare una priorità di elaborazione e quindi di servizio ai messaggi di nnINVITE in modo da far concludere le sessioni già instaurate, accodando il messaggio secondo una disciplina di coda LIFO (last in first out), ponendolo in testa al buffer. Per quanto riguarda i messaggi di INVITE vengono trattati in modo consueto ovvero mediante l'inserimento in coda al buffer. La coda a priorità anche se fisicamente considerata come unica, mantiene una gestione separata dei messaggi INVITE/nnINVITE, tramite la costruzione di due variabili *INVITEqueue_* e *nnINVITEqueue_* ("sottocode") si ha il controllo sui messaggi accodati:

INVITEqueue=inviteArrived−INVITEServiced−INVITEdrop

nnINVITEqueue=nninviteArrived−nnINITEServiced−nnINVITEdrop

Anche in questo caso gli eventi riscontrabili sono 4:

1. Buffer pieno : caso in cui il sistema è completamente pieno o caso in cui una delle due "sottocode" è piena, successivamente si richiama la funzione **action0(p)** per mezzo della quale si scarta il messaggio d'INVITE o di nnINVITE, incrementando gli opportuni contatori.

2. numero dei messaggi maggiore della *TH_HIGH_* : in questo caso viene settata *invThExceeded* ad 1 e richiamata la funzione **action3(p)** , questa è l'azione che gestisce il meccanismo di local overload, nello specifico se il messaggio è un INVITE, questo viene accodato in fondo al buffer, se questo è un nnINVITE la funzione inserisce il messaggio in testa al buffer.

3.numero dei messaggi al di sotto della *TH_LOW_* : la variabile di controllo *invThExceeded* è settata a 0 e viene richiamata la funzione **action2(p)** tramite la quale il messaggio viene accodato in fondo al buffer.

4.numero dei messaggi compreso tra la *TH_LOW_* e la *TH_HIGH_*: in questo caso se la variabile *invThExceeded* è settata a 0 viene ripresa la funzione **action2(p)** sopra esplicata. Se la variabile *invThExceeded* è settata a 1 si richiama la funzione **action3(p)** il cui funzionamento è stato esplicato nel caso 1.

OCProxy::handle: la funzione handle gestisce il centro di servizi dei sistemi a coda ed in particolar modo effettua il servizio in base alla tipologia di messaggio , se questo è un INVITE viene incrementata la variabile *INVITEServiced_* nel caso di nnINVITE viene incrementata la variabile *nnINVITEServiced_* , e il messaggio essendo stato servito viene eliminato dal buffer.

La gestione dei tempi di servizio è affidata alla funzione **OCPRoxy::timerExpiredMIS**:
Il messaggio in testa al buffer viene prelevato, e nel caso sia un messaggio nnINVITE, la funzione gli attribuisce un time di processing pari a 5ms. Se invece si tratta di un messaggio d'INVITE viene richiamata la funzione **OCProxy::calcola _Tproc()** tramite la quale il tempo di processing del messaggio è una variabile gaussiana di media 30 ms e deviazione standard 5 ms. Dopo aver assegnato il time processing al messaggio , questo viene passato alla funzione handle. Nel caso in cui la coda è vuota il time processing del messaggio sarà di default pari a 10 microsecondi .

OCProxy::calcola() : L' OC-proxy ha il compito di monitorare le risorse di sistema quali tasso di servizio, sessioni in fase di servizio e le sessioni in fase di attesa. La funzione calcola() determina sostanzialmente il Backload tramite questi parametri che, appositamente schedulati in delle liste, caratterizzano dei feedback che verranno inviati alle sorgenti per adattare il proprio rate per l'algoritmo sviluppato dal gruppo di Schulzrinne (TU-approach). In formule, per il calcolo del backload secondo il TUapproach il tasso di servizio (mu) è calcolato come rate degli INVITE accettati:

$$muTU = \frac{INVITEarrived - INVITEdrop}{TMbase}$$

Tm_Base è definito nell'header della classe sip-timers con un valore di default pari a 10 sec.
Il calcolo del Backload secondo l'approccio TU è dato dalla formula:

$$\text{BackTU} = \frac{\text{INVITEqueue} + \text{nnINVITEqueue}}{\text{Lsess} - 1}$$

dove Lsess sta ad indicare il numero medio di messaggi per sessione ed è dato dalla formula:

$$\text{Lsess} = \frac{\text{INVITEservice} + \text{nnINVITEservice}}{INVITEaccepted}$$

dove:

$$INVITEaccepted = \text{inviteARRIVED} - \text{INVITEdrop}$$

Per quanto riguarda l'aggiornamento dei feedback che vengono inoltrati alle sorgenti, **OCProxy::newOCTU,** è stata implementata una lista (**double OCProxy::mean_list(list<double> lista**)) a finestra mobile di dimensione pari a 20 che restituisce la media dei valori in ingresso, questi valori sono fb_TU per il meccanismo TU.
In formule fb_TU è :

$$\text{fbTU} = \left| \frac{\text{muTU} * \left(1 - \left| \frac{\text{dqTU} - 0.2}{\text{TMbase}} \right| \right)}{10 * \text{norm}} \right|$$

dove dqTU è il ritardo di accodamento stimato per il meccanismo TU, ed il valore 10*norm rappresenta un fattore di normalizzazione affinché l'algoritmo sia allocato tra le sorgenti attive, considerando una distribuzione equa delle risorse.

Vediamo ora come alcuni parametri relativi all'OCPoxy e il loro valore nella nostra simulazione:

-**Agent/OCProxy set unacoda_**: attiva il sisetma ad una coda
-**Agent/OCProxy set dim_MAX_**: setta la dimensione di messaggi che il buffer può contenere
-**Agent/OCProxy set TH_HIGH_**: è il valore di soglia alta , di cuia bbiamo spiegato il funzionamento in precedenza. Questo valore è ottenuto come l' 80% dell *dim_MAXINV_*, nel caso in cui il valore non fosse un intero , verrà arrotondato per eccesso all'intero superiore o per difetto all'intero inferiore a seconda che si superi o meno il decimale 5.
-**Agent/OCProxy set TH_LOW_**: è il valore della soglia bassa , ottenuto come il 60% *dim_MAXINV_*, anche per questo setting si è scelto di arrotondare il valore ottenuto, qualora non fosse un intero, come visto per la *TH_HIGH_*.

Agent/SIPProxy set OC_off : questo setting abilita i meccanismi di distributed overload, presi in considerazione, nei proxy della rete.
Agent/SIPProxy set TU_approach : attiva l'approccio TU.
Agent/OCProxy set dim_MAXINV_ : setta la dimensione massima di messaggi INVITE che il buffer può contenere
Agent/OCProxy set dim_MAXnnINV_ : setta la dimensione massima di messaggi nnINVITE che il buffer può contenere, visto che le simulazioni sono state effettuate con un sistema di segnalazione di 7 messaggi la dimensione di questo setting dipende dal valore del setting attribuito a set dim_MAXINV_, ovvero:

$$MAXnnINV = 6 * MAXINV$$

```
set serveraddrPR(10) [$n($nn(10)) node-addr]
   set sipPR(10) [new Agent/OCProxy proxy(10).com]
   $ns attach-agent $n($nn(10)) $sipPR(10)
   DNSGod register proxy proxy(10).com $serveraddrPR(10)
   $sipPR(10) set recordeRoute_ 1

$sipPR(10) set unacoda_ 1
$sipPR(10) set TH_HIGH_ 36
$sipPR(10) set TH_LOW_ 0
$sipPR(10) set dim MAX  36
```

Figura 2.9- Codice settaggio OC-Proxy

2.6 Modifiche al modulo SIP

In questo paragrafo saranno descritte le modifiche effettuate su alcuni file(sip-proxy.cc,sip-timers.cc e sip-trans.cc) del modulo SIP per l'implementazione dei sistemi di Local e Distribuited Overload

2.6.1 SIP-PROXY.CC

Le modifiche sostanziali che riguardano questo file consistono nel modo in cui le sorgenti riescono ad adattare il proprio traffico tramite l'utilizzo dei feedback messi a disposizione dalla funzione **OCProxy::newOCTU,** andiamo dunque a vedere nel dettaglio la funzione void SIPProxy::recv.
void SIPProxy::recv: innanzitutto viene creata una variabile x uniformemente distribuita tra 0,1 che ci permette di avere un

criterio di decisione secondo il quale si effettua lo scarto del messaggio con l'obiettivo appunto di diminuire il traffico. Per verificare se tale messaggio è uno dei papabili allo scarto, si effettua un controllo sul messaggio, se questi è un INVITE ed il meccanismo richiamato da script Tcl tramite il metodo bind è TU, si estrae dal campo VIA il valore oc_TU_, a questo punto se la variabile x risulta essere maggiore del valore oc_TU_ e l'uri del dominio nel campo TO è diverso da "proxy(10).com" si procede con lo scarto, altrimenti il pacchetto sarà servito.

2.6.2 SIP-TIMERS.CC

Nel file SIP-TIMERS.cc è stata introdotta la nuova classe SIPTimerMIS derivata in modo pubblico da TimerHandler. Questa classe ha il compito di gestire i timer dell'oc-proxy, vediamo nello specifico le funzioni membro:
void SIPTimerMIS::start : con questa funzione vengono inizializzati il timer di misura (Tm), il timer di controllo (Tc) e il timer di servizio (Thandle), per convenzione è stato scelto Tm=10s, Tc=10*Tm mentre Thandle è stata impostata a 10 μs.
void SIPTimerMIS::restart : la funzione membro restart ha il compito di incrementare i timer Tm e Tc, sostanzialmente ad ogni suo stanziamento viene raddoppiato il loro valore.
void SIPTimerMIS::expire : expire richiama la funzione timerExpiredMIS definita in modo protected nella classe OCPROXY, tramite tale funzione si richiama la funzione restart che, come abbiamo visto, ha il compito di incrementare i timer Tm e Tc.

2.6.3 SIP-TRANS.CC

Vediamo ora quali sono le modifiche apportate al file sip-trans.cc, partiamo con le modifiche alla classe SIPTransLayer, questa classe gestisce le transazioni degli elementi sip nello specifico gestisce le richieste e le risposte tra i livelli logici. Innanzitutto vengono costruite delle liste in cui vengono raccolte tutte le transazioni, in **list<SIPTransaction *> CltTr,** vengono raccolte tutte le transazioni INVITE/nnINVITE CLIENT, in **list<SIPTransaction *> SrvTr** si hanno le transazioni che si riferiscono ai nnINVITE SERVER ed infine in **list<SIPTransaction *> SrvTrInv** abbiamo le transazioni degli INVITE SERVER.

int SIPTransLayer::returnSERVER : questa funzione restituisce il numero delle transazioni SERVER.
int SIPTransLayer::returnCLIENT : restituisce il numero delle transazioni CLIENT.
void SIPTransLayer::fromTU : tramite questa funzione si procede al riempimento della lista riguardante le transazioni clientTransaction, in particolare se si riceve dal livello TU un messaggio lato client, ed il messaggio è un INVITE si stanzia un oggetto della classe CltINVITETrans, se il messaggio è un nnINVITE si stanzia un oggetto della classe CltnonINVITETrans, dopodiché il messaggio viene accodato alla fine della lista CltTr.
void SIPTransLayer::updateServiceRate : non viene richiamata.
void SIPTransLayer::updateArrivalRate : non viene richiamata.
void SIPTransLayer::newSrvTrans : aggiorna le transazioni lato server, nello specifico se il messaggio è un INVITE si stanzia un oggetto della classe SrvINVITETrans ed il

messaggio viene accodato in fondo alla lista SrvTrInv, se il messaggio è un nnINVITE si stanzia un oggetto della classe SrvNonINVITETrans ed il messaggio viene accodato nella lista SrvTr

3 Scenario di simulazione

Dopo aver affrontato nei capitoli precedenti le questioni riguardanti il protocollo SIP e il modello di simulazione, in questo capitolo descrivo lo scenario di simulazione che è stato costruito con il simulatore NS-2.

3.1 Topologia

Nella topologia(vedi figura 3.1) sono presenti dieci domini (colore blu), ognuno dei quali caratterizzato da:

- 198 User Agent con range di indirizzi [x.y.2 – x.y.199]
- un server Proxy con indirizzo [x.y.0]
- un router con indirizzo [x.y.1]

Ciascuno di questi domini è collegato attraverso un link satellitare ad router che fa parte di un dominio esterno(colore rosso), che rappresenta il dominio degli interlocutori costituito da:

- 1988 User Agent con range di indirizzi [10.y.2-10.y.1990]
- un server Proxy (denominato Proxy BD) con indirizzo [10.y.0]
- un router con indirizzo [10.y.1]

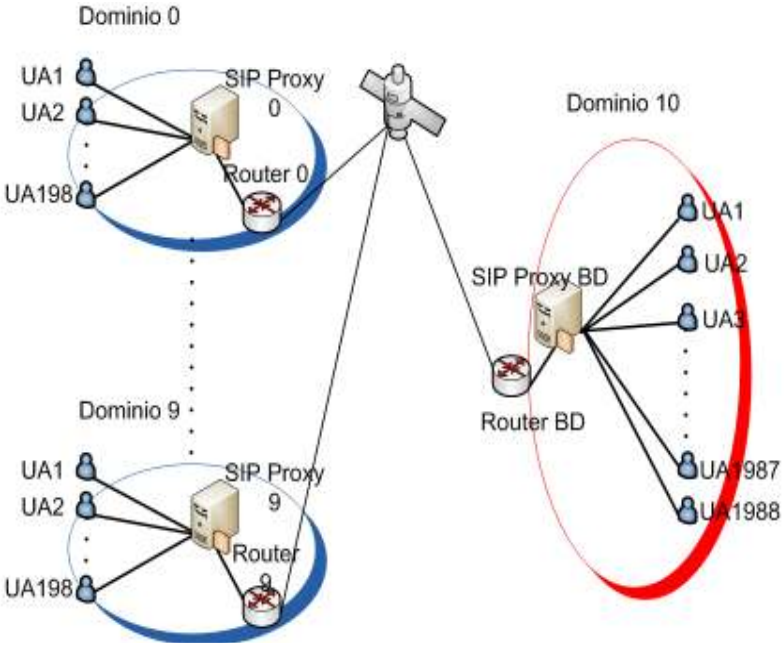

Figura 3.1-Topologia dello scenario simulato

Il piano di indirizzamento usato è di tipo gerarchico del tipo [x.y.z], dove :

- x indica il dominio
- y indica il sottodominio
- z indica il nodo.

Ns-2 offre una struttura gerarchica limitata, infatti abbiamo a disposizione:

- 10 bit per i domini , con cui si possono implementare al massimo 1023 domini

- 11 bit per i cluster, , con cui si possono implementare al massimo 2048 cluster
- 11 bit per i nodi, , con cui si possono implementare massimo al massimo 2048 nodi

Di seguito è mostrato il frammento di codice , utilizzato per realizzare la nostra topologia:

```
# Topology information :
lappend domain 11
AddrParams set domain_num_ $domain

lappend cluster 1 1 1 1 1 1 1 1 1 1 1
AddrParams set cluster_num_ $cluster

lappend eilastlevel 200 200 200 200 200 200 200 200 200 200 2000
AddrParams set nodes_num_ $eilastlevel
```

Figura 3.2-Codice settaggio topologia

Che indica che la topologia è costituita da 11 domini, con ognuno all'interno un cluster(non ci sono sottodomini), e ciascun dominio è costituito da 200 nodi, tranne l' ultimo che ne contiene 2000.

Tutti i link all'interno dei domini hanno una capacità pari a 10 Mbps e disciplina di coda Droptail(quando la coda è piena , si scartano i pacchetti). I router della topologia sono dei semplici nodi, che non hanno alcuna funzione specifica.

I SIP PROXY da 0 a 9 , sono quelli forniti dal modulo SIP di Rui Prior , ognuno di questi proxy si occupa delle registrazioni degli User Agent interni al proprio al dominio,e alle loro richieste. La capacità di questi proxy è stata stimata in 50 calls/s. La durata della simulazione è di 1500 secondi. Ora passeremo in rassegna in maniera più particolareggiata alcuni elementi che compongono la topologia.

3.2 User Agent

Il simulatore Ns-2 è in grado di simulare il comportamento degli User Agent(UA) da entrambi i lati UAC e UAS.

```
32    #User Agent del dominio 5
33    for {set i 802} {$i < 1000} {incr i} {
34        set sipU($i) [new Agent/SIPUA U($i) proxy(4).com]
35        $ns attach-agent $n($i) $sipU($i)
36        $sipU($i) set-proxy $serveraddrPR(4)
37    }
```

Figura 3.3-Codice settaggio proxy

Con l'utilizzo del modulo Sip creato da Rui Prior, è possibile studiare la fase di segnalazione tra due UA durante l'instaurazione della chiamata , fase costituita da 7 messaggi. Ogni User Agent , dopo essere stato creato (vedi. Figura 3) deve effettuare una procedura di registrazione con la quale notifica al Proxy server la sua presenza all'interno del dominio in cui si trovano. In questa modo ogni sua richiesta passerà da il Proxy all' interno del proprio dominio, e il Proxy server venuto a conoscenza della sua esistenza saprà dove indirizzare i messaggi indirizzati al particolare User Agent. Alla fase di

registrazione segue la fase di segnalazione, che serve per instaurare in maniera corretta lo scambio dei media , che costituisce il fulcro dello chiamata.
La fase di segnalazione inizia non appena l' UAC invia una richiesta di INVITE al UAS. Ciascun elemento sul percorso dell' invite , conferma la ricezione e il passaggio di questo messaggio con un 100Trying. Alla ricezione dell' INVITE l' UAS risponde con un 180Ringing , che indica che si sta avvertendo l'utente (il telefono squilla), e con l' invio del 200OK accetta l' apertura della sessione, la ricezione dell'ACK determina l'apertura in maniera corretta della sessione.

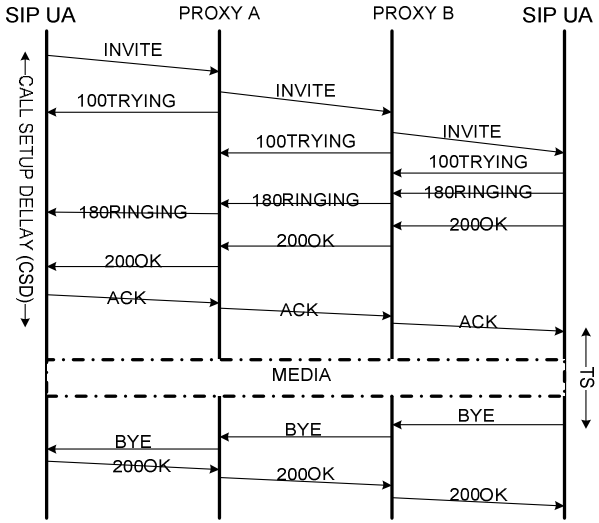

Figura 3.4-Segnalazione SIP

Una volta aperta la sessione, avviene lo scambio dei media, che non riguarda la fase di segnalazione, dopo di che la sessione deve essere chiusa con un messaggio di BYE, la fase di

segnalazione termina con la ricezione del 200OK, che notifica l'avvenuta ricezione del BYE.

Analizzando il modulo di Ns2 che costituisce l' UA, si può notare che oltre alla generazione dei messaggi coinvolti nella segnalazione, l' UA si occupa anche di altre importanti operazioni, le principali sono:

-Procedura di registrazione
-Gestione dei Timer SIP
-Recupero di situazioni critiche come: Miss-matching dei messaggi, e INVITE FAILED.

Nel caso particolare del mio scenario di simulazione, sono state apportate delle modifiche al codice(file ns-sip.Tcl) in modo che il comportamento degli UA segua queste linee:

- Alla notifica di un INVITE FAILED ,dopo un' attesa pari a Ti(il cui valore è estratto in modo casuale da una variabile esponenziale di valore medio 10 secondi), viene generato un nuovo messaggio di INVITE
- Alla notifica del termine di una sessione con successo (evento BYE-OK), dopo un' attesa pari a Ti (il cui valore è estratto in modo casuale da una variabile esponenziale di valore medio 10 secondi)viene aperta una nuova chiamata con lo scambio tra chiamante e chiamato rispetto alla sessione precedente.
- Alla notifica dell' apertura della sessione , dopo un' attesa pari a Ts(il cui valore è estratto in modo casuale da una variabile esponenziale di valore

medio 30 secondi), il chiamato genera un Bye per chiudere la sessione(vedi fig.4).

L' obbiettivo di questo lavoro di tesi è di analizzare le prestazioni dello scenario costruito in una situazione di stress, e capire l'impatto che possono avere i TIMER SIP su link ad elevati ritardo come quello satellitare. Quindi si sono rese necessarie alcune modifiche , come quelle descritte sopra, per sottoporre tutte le componenti dello scenario a situazioni di stress per tutta la durata della simulazione.

3.3 Link Satellitari

I link che collegano i router 0:9 con il router BD, rappresentano dei link geo-satellitari.
Purtroppo la non interoperabilità tra i moduli SIP, e quello satellitare , ci ha costretto ad una semplificazione nella modellazione dei link, ma siamo comunque riusciti a rappresentarne le caratteristiche principali. Il mio studio riguarda esclusivamente il piano di segnalazione, quindi molti aspetti disponibili sui moduli satellitari di NS2, non ci interessavano. Quello che interessa e che siamo riusciti a fare, è imporre sui link satellitari elevati ritardi, propri dei link geo-satellitari, in modo da poter comprendere l'impatto che questi assieme ai TIMER SIP hanno sulle prestazioni dello scenario. Per caratterizzare i link mi sono affidato a quanto ottenuto attraverso misurazioni effettuate in dipartimento (vedi Performance Evaluaion of Moip Applictions over Satellite : An Experimental Study).

Per quanto riguarda il ritardo sul link, i risultati ottenuti da queste misurazioni hanno mostrato la seguente situazione (vedi tabella 3.5):

Packet Size (byte)	Mean RTT (ms)	RTT (ms) 99% Conf.Interval
1500	2853	[2679,3027]
200	1334	[1296,1327]
73	1256	[1206,1306]
60	1225	[1195,1255]

Figura 3.5-Tabella misurazioni

Data la dimensione dei pacchetti coinvolta nella segnalazione SIP, cosi come sono stati costruiti nel modulo SIP di Rui Prior,è stata preso come riferimento per il RTT un valore pari 1400 ms, e quindi un ritardo one way pari a 700 ms
I restanti parametri dei link satellitari sono i seguenti:

- Ritardo One Way 700 ms
- Capacità del link : 2048 Kbps
- Disciplina di coda : DropTail

Altri parametri come la Packet Loss o il Jitter sono stati trascurati , dati i valori irrisori assunti.

3.4 Proxy

Per quanto riguarda i proxy utilizzati nello scenario è necessario specificare che i proxy dei domini blu , sono quelli definiti nel modulo SIP di Rui Prior e non hanno subito

modifiche. Questi proxy hanno il compito di inoltrare i messaggi ricevuti dagli User Agent in maniera corretta, e lavorano con disciplina di coda FIFO. Il proxy utilizzato nel dominio rosso , denominato proxy BD è un proxy modificato che implementa soluzioni di Local e Distribuited Overload, come descritto nei precedenti paragrafi.

3.5 Procedure adottate nella simulazione

Se ci fermassimo alla topologia descritta nei paragrafi precedenti, avremmo un notevole numero di nodi incapaci di interagire tra di loro.
Il modulo di Ns2 adottato per le simulazioni ci permette di dare vita a questi nodi, attraverso i metodi SIP. Con le procedure descritte nei paragrafi successivi, viene imposto ai nodi di fare particolari azioni ad un determinato istante di simulazione.
Le azioni sono riconducibili a 3 metodi SIP:
REGISTER,INVITE e BYE.

3.5.1 Registrazioni

La procedura di registrazione di tutti gli User Agent presenti nello scenario , è stata eseguita in maniera scaglionata in modo tale da non sovraccaricare i link. Sono stati utilizzati 20 processi di registrazione, ciascuno dei quali registra 200 User Agent . Il primo processo di registrazione inizia al secondo 2 di simulazione, e gli altri processi seguono ad intervalli di 5 secondi.
Il numero di utenti che effettuano le chiamate non è uguale per tutte le simulazioni, nonostante questo per ogni simulazione effettuata si registrano tutti gli User Agent creati nello scenario.

```
proc REGI1 { } {
global sipU
for {set i 2} {$i < 200} {incr i} {
        $sipU($i) register
     } }
```

Figura 3.6- Frammento codice processo di registrazione

3.5.2 Invite

Dopo aver effettuato la registrazione , necessaria per l'instradamento corretto dei messaggi, è necessario per l'instaurazione di una chiamata l'invio di un messaggio di INVITE. Sono state utilizzate dieci procedure, una per ogni dominio dei chiamanti. Il tempo di start dei processi dei processi di INVITE è fissato al 150esimo secondo di simulazione.

```
proc invitare1 {} {
global ns sipU
for {set i 2} {$i < 72} {incr i} {
set T_now [$ns now]
set Ts 0
set c [expr $i+2000]
set Tstart [expr $Ts+$T_now]
$ns at $Tstart  "$sipU($i) invite U($c) proxy(10).com"

}
}
```

Figura 3.7- Frammento codice processo inviti

Entrati nel processo, in maniera iterativa viene estratto il tempo

attuale della simulazione (T-now-i, con i si identifica la sessione), a cui viene sommato un tempo pari a Ti-i (generato in maniera casuale da una variabile esponenziale di valor medio 10 secondi), la somma di questi due tempi sarà i tempo in cui viene inviato l'invite (T-INVITE-i).

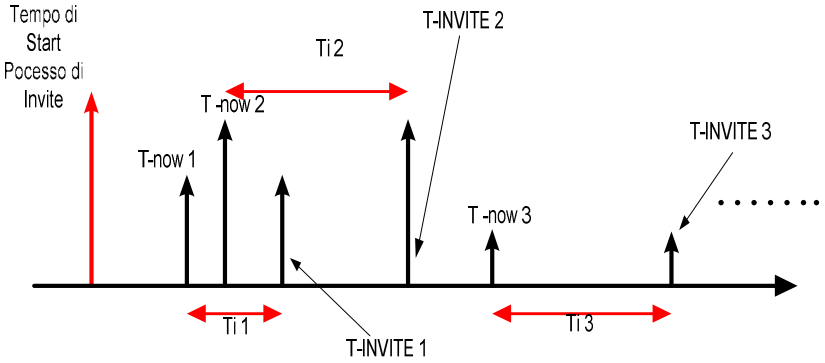

Figura 3.8- Asse dei tempi processo di invite

3.5.3 BYE

La gestione dei BYE avviene in maniera completamente automatica, non è stata necessario l'utilizzo di alcun processo nello script TCL. Infatti non appena la sessione viene aperta con successo, viene estratto un tempo TS (estratto in maniera casuale da una variabile esponenziale di valore medio 30 secondi) dopo il quale il chiamato invia il BYE per chiudere la chiamata precedentemente aperta. Come già detto in precedenza quando la sessione viene chiusa (il 200OK conferma la ricezione del BYE), dopo un'attesa pari a Ti viene aperta una nuova sessione con lo scambio tra chiamato e chiamante.

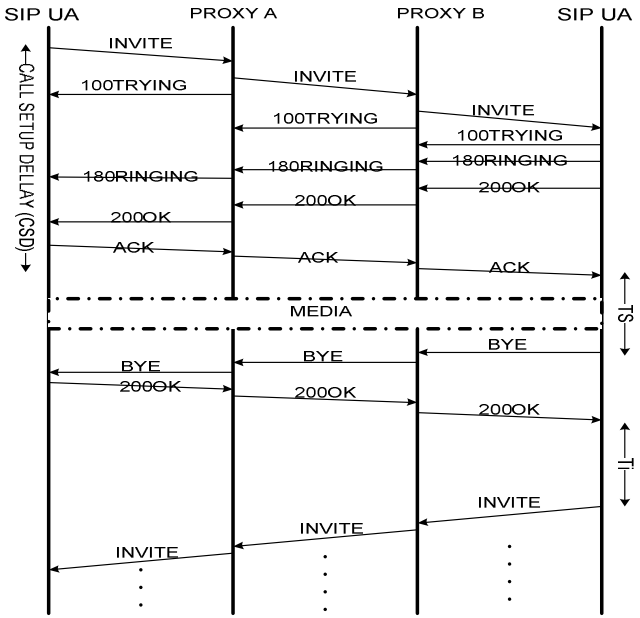

Figura 3.9- Scambio di messaggi tra SIP UA

4 Risultati della simulazione

In questo capitolo verranno presentati i risultati delle simulazioni tramite grafici e si giungerà alle conclusioni.

I risultati sono state ottenuti con l' ausilio del simulatore di reti Ns2, i risultati sono stati poi "ripuliti" con script AWK e trasformati in grafici con il programma MATLAB. Prima di addentrarci nei risultati è necessario capire quali indici prestazionali sono stati misurati della segnalazione SIP nello scenario di simulazione.

4.1 I parametri prestazionali

La telefonia su IP, deve sempre confrontarsi con la rete telefonica tradizionale, sia in termini di affidabilità sia per la qualità del segnale trasportato.
Un parametro importante nella misura della QOS (Quality of Service) del servizio è il Call Setup Delay (CSD).
In letteratura esistono diverse definizioni di questo parametro, alcune anche molto differenti tra loro. Nel mio studio dato che l'analisi riguarda strettamente il piano di segnalazione, la definizione considerata è la seguente:

-Call Setup Delay (CSD) è considerato come la differenza tra l'istante in cui l' UAS riceve l' ACK(con cui si chiude la fase di instaurazione della sessione) e il tempo nel quale l' UAC ha inviato l' INVITE

In sintesi questo parametro ci dice quanto tempo si impiega ad instaurare una chiamata, quindi minore è il suo valore migliore è la qualità del servizio offerto.

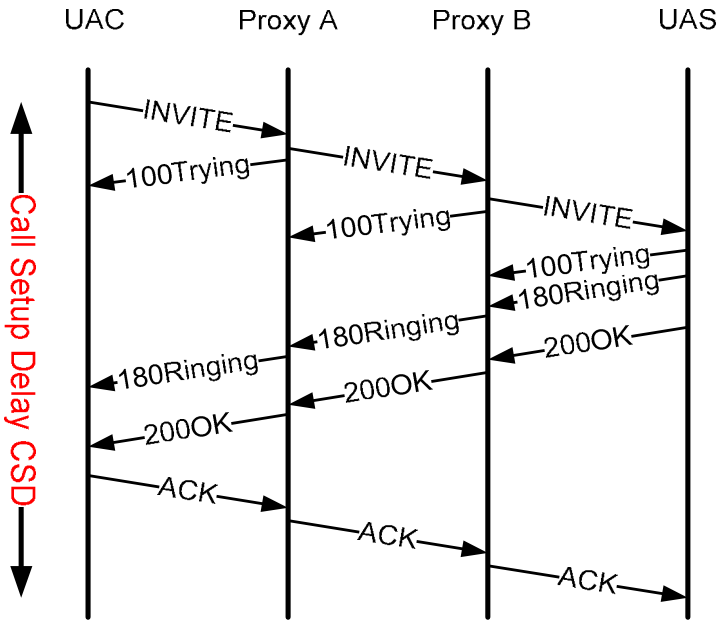

Figura 4.1- Definizione di CSD

Il secondo parametro preso in considerazione non è un vero proprio indice prestazionale come il precedente, ma è il numero di sessioni al secondo che vengano istaurate in maniera corretta. Per ottenere questo parametro ho considerato tutte le sessioni che si instaurano e divise per la durata della simulazione meno il tempo della fase di registrazione.

$$\frac{\#\text{sessioniOK}}{s} = \frac{\#\text{sessioniok per durata della simulazione}}{1250}$$

Per considerare una sessione non instaurata in maniera corretta devono avvenire due eventi:

- Scade il Timer B, e l' INVITE fallisce (evento INVITE FAILED)
- La sessione si apre in maniera corretta, ma con un CSD maggiore di 10 secondi

L'ultimo parametro considerato deriva dal precedente, ed è la percentuale di chiamate fallite, cioè che non si sono instaurate in maniera corretta.

4.2 Analisi delle simulazioni

L 'obbiettivo del mio lavoro è quello di valutare l'impatto che l'alto ritardo imposto dai link satellitari ha sui *Timer SIP,* e quindi capire come l'interazione tra i timer e l'elevato ritardo si rifletta sulle prestazioni della segnalazione SIP.

Quindi per prima cosa vedremo come si comporta lo scenario descritto nel capitolo precedente senza alcuna modifica, poi modificando i Timer SIP, vedremo come varieranno le

prestazioni, e in aggiunta vedremo anche il comportamento di alcuni meccanismi di overload su questo scenario.

I valori di prova per il Timer A sono:

-500ms , valore di default definito nella RFC 3261

-800 ms , valore di poco superiore al ritardo one-way del link satellitare

-1100 ms , valore molto al di sopra del ritardo one-way del link

Le simulazioni sono state effettuate in varie situazioni di traffico, variando il numero di utenti per dominio che effettua chiamate. L' equazione sottostante descrive il traffico offerto in termini di INVITE quindi di sessioni, *n* è il numero di domini coinvolti nelle comunicazioni ed è sempre fissato a un valore pari a 10, mentre *m* è il numero di utenti attivo per dominio.

Facendo variare il parametro, sono state create varie situazioni di traffico offerto. I parametro Ts è la durata media di una sessione ed è modellato con una variabile esponenziale di valore medio 30 secondi,e Ti è il tempo di interarrivo tra una chiamata e l'altra modellato con una variabile esponenziale di valore medio di 10 secondi.

$$\lambda = \frac{m * n}{Ts + Ti}$$

4.2.1 Sistema a coda FIFO

Il primo caso come detto , ci serve per capire una situazione importanti. Lo scenario è uguale a quello nel capitolo precedente, non ci sono meccanismi di overload attivi, tutti i proxy lavorano con disciplina FIFO, e i Timer SIP sono quelli della RFC 3261. Vediamo cosa accade per come sono impostati i Timer SIP nella RFC 3261 in questo scenario. L' UAC invia l'INVITE all' UAS(i nodi intermedi sono omessi nella figura per semplicità), ma questo messaggio non fa in tempo ad arrivare dato la distanza da percorrere su link satellitare, che il Timer A(valore di default 500 ms) scade e l'UAC ritrasmette il messaggio di INVITE, e raddoppia il Timer A (1 secondo).

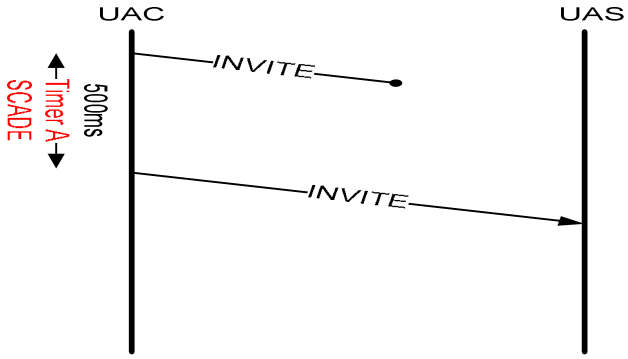

Figura 4.2- Scadenza del Timer A

Quindi in questa situazione abbiamo una probabilità di ritrasmissione del messaggio di INVITE pari ad 1, cioè la ritrasmissione avviene sempre. Vediamo ora dai grafici le

prestazioni di questo scenario. Cosi come per gli altri scenari che verranno analizzati, le prestazioni sono state misurate al variare delle chiamate al secondo offerte. Infatti sulle ascisse sono presenti diversi valori di Calls per second(CPS) ottenuti variando il numero di utenti che effettuano chiamate per ciascun dominio chiamante. Il primo grafico ci mostra i valori del CSD ;come si nota dal grafico ci si attesta su valori di CSD che possono garantire un buon servizio fino a 12 calls/sec. Infatti se si considera che un valore "buono" di CSD per chiamate internazionali(evento probabile su scenari con link satellitari) è 8 secondi , allora si capisce come per questo scenario il numero di chiamate che si riesce a smaltire offrendo un buon servizio è 12 chiamate al secondo. Per valori di chiamate al secondo superiori le prestazioni interini di CSD si degradano molto rapidamente.

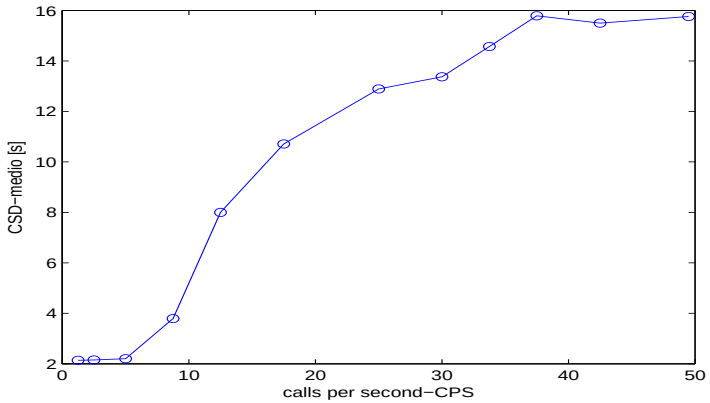

Figura 4.3- Grafico del CSD, coda senza priorità

Il secondo grafico riguarda le prestazioni del SIP Proxy BD, infatti per come è stato strutturato lo scenario, tutto il traffico passa inevitabilmente per quel proxy. Quindi attraverso questi grafici si analizzano le prestazioni del SIP Proxy BD, che è un Proxy OC descritto nel capitolo 2. Tutti i parametri di configurazione del Proxy BD sono stati impostati per ottimizzare le performance e quindi di ridurre il tempo di attesa dei messaggi all'interno della coda del proxy , che lavora con disciplina FIFO.

Il grafico mostra un andamento perfettamente prevedibile dalla conoscenza delle prestazione del proxy BD. Il proxy mostra prestazioni crescenti in termini di chiamate che riesce a instaurare fino a valori di 8 chiamate al secondo. Da quel valore in poi le prestazioni cominciano a decrescere per giungere a valori irrisori a partire da 13 chiamate al secondo. Le prestazioni si attestano su valori inferiori al valore teorico che ci si dovrebbe attendere, questo fatto è certamente da attribuire alla numerose ritrasmissioni di cui abbiamo parlato prima, che il proxy si trova a gestire oltre al normale traffico per la segnalazione SIP.

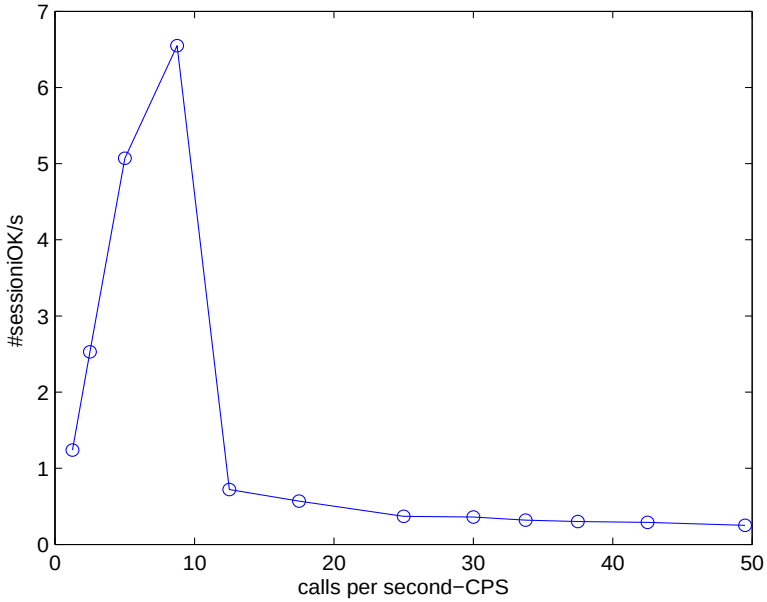

Figura 4.4- Andamento #sessioniOK/S, coda senza priorità

Il terzo grafico è legato a quanto detto per il grafico precedente, infatti fino a 8 chiamate al secondo le perdite sono quasi nulle in quanto il proxy riesce a smaltire il traffico offerto. Poi come nel grafico precedente si avevano valori di sessioni al secondo sempre minori, nel terzo grafico la percentuale di sessioni che non vanno a buon fine cresce in maniera molto rapida fino ad arrivare a valori che superano il 90%. Per quanto riguarda la percentuale di sessione fallite, posso dire che il proxy Bd mostra valori accettabili fino a quando si trova a gestire un

numero massimo di chiamate che non supera 8 chiamate al secondo.

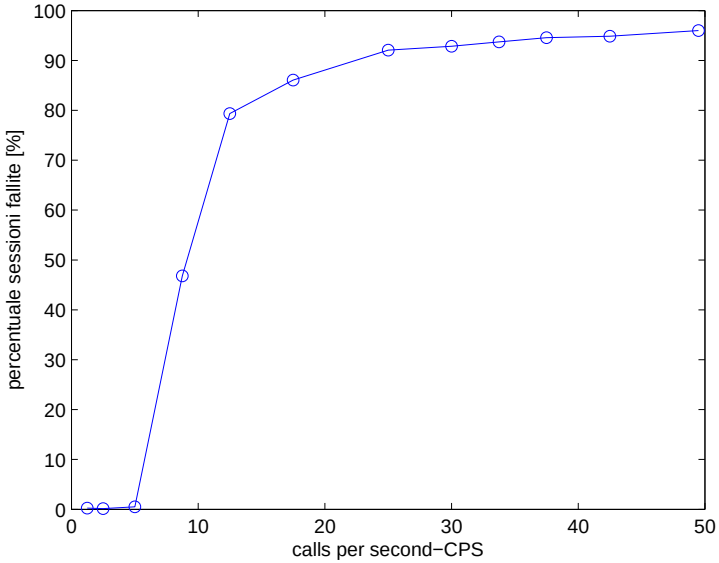

Figura 4.5- Andamento percentuale sessioni fallite, coda senza priorità

4.2.2 Risultati con coda a priorità

Il paragrafo precedente mi è servito per comprendere la situazione di partenza dello scenario che ho costruito, e quindi vedere in quali punti si poteva operare per migliorare le prestazioni. Non vi è dubbio che il punto cruciale su cui si può lavorare è il SIP Proxy Server BD, infatti su gli altri elementi dello scenario sia per il ruolo che svolgono nell'architettura

SIP sia per le possibilità di modifica che il modulo SIP di Rui Prior offre, non è possibile operare per migliorare gli indici prestazionali che ho considerato per il mio lavoro. Da qui l'idea di operare sia sui Timer Sip in modo da ridurre le ritrasmissioni e di migliorare la gestione dei messaggi da parte del proxy BD. Per fare ciò quindi è stato necessario modificare il codice di Ns2 (nello specifico il file *sip-timers.h*) e utilizzare il sistema a coda con priorità nel proxy BD che implementa una soluzione di "Local Overload"(vedi figura sottostante).

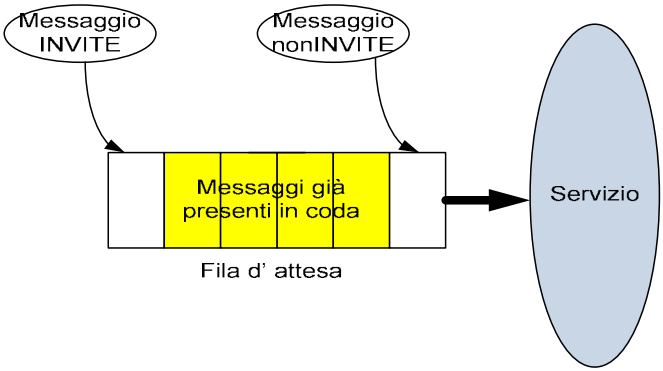

Figura 4.6- Sistema Local Overload

Il principio su cui si basa il Local Overload è quello di dare priorità a messaggi NonINVITE rispetto a messaggi INVITE, in modo da favorire sessioni già in fase avanzata di instaurazione rispetto a quelle in fase iniziale. Con questo *"modus operandi"* quindi ci si aspetta un numero minore di chiamate al secondo servite, ma un Call Setup Delay minore.

Con l'ausilio dei grafici andiamo a vedere le prestazioni dello scenario in termini di CSD. Quello che si nota è un' andamento

nettamente differente rispetto alla coda senza priorità, e per valori del Timer T1 pari a 500 ms (curva verde) prestazioni decisamente confortanti che si attestano per tutti i valori di traffico offerto sotto la soglia degli 8 secondi.

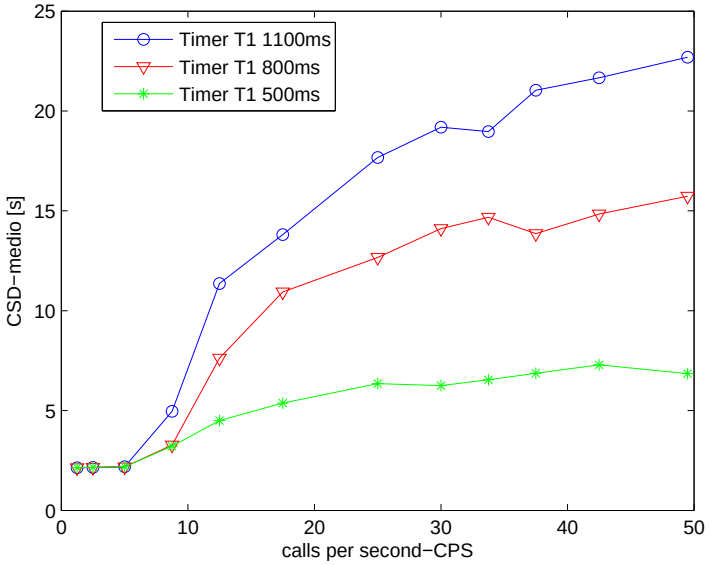

Figura 4.7- Grafico del CSD, coda con priorità

Le altre due curve presentano un andamento simile tra loro , ma su valori differenti. L'aspetto più importante, che merita di essere sottolineato, è che un aumento del Timer SIP che doveva far diminuire le ritrasmissioni e quindi tenere il CSD su valori accettabili , non ha l'effetto sperato, segno che le prestazioni del proxy BD assieme agli alti ritardi del link satellitare ci portano sempre ad avere un numero considerevole

di ritrasmissioni, e quindi con Timer SIP i valori di CSD della sessione si alzano.

Gli altri indici prestazionali misurati dipingono una situazione ben chiara,le prestazioni del Sip Proxy BD sono perfettamente in linea con quello che ci si aspettava dal meccanismo di Local Overload. Da notare che nel caso di Timer A posto a 1100 ms , le prestazioni per entrambi i parametri sono peggiori rispetto agli altri timer. Con il meccanismo a priorità si nota come per tutto i valori di traffico offerto si riesca a mantenere un livello di prestazioni costante senza i "crolli" del caso senza priorità.

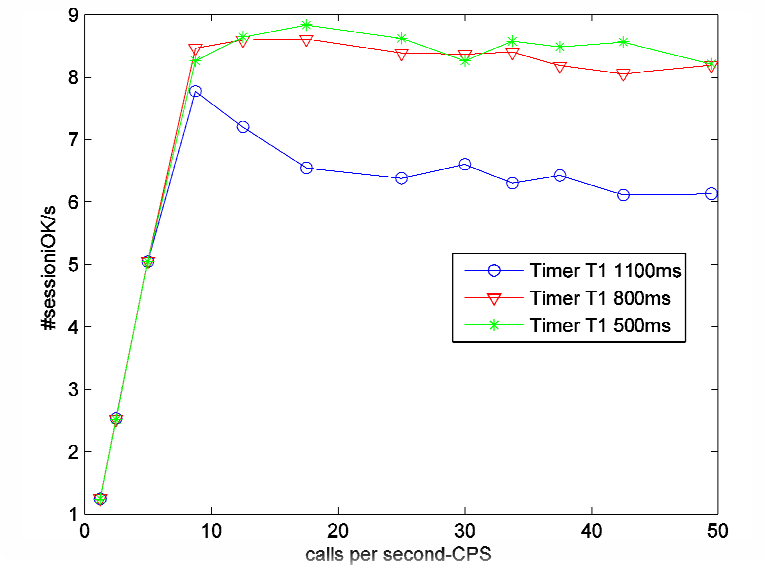

Figura 4.8- Andamento #sessioniOK/S, coda con priorità

Anche per quanto riguarda la percentuale di sessioni perse il meccanismo di Local Overload permette di avere percentuali minori persino della metà, e una crescita molto più graduale, segno che attraverso questo meccanismo lo scenario riesce a smaltire con più facilità le sessioni offerte.

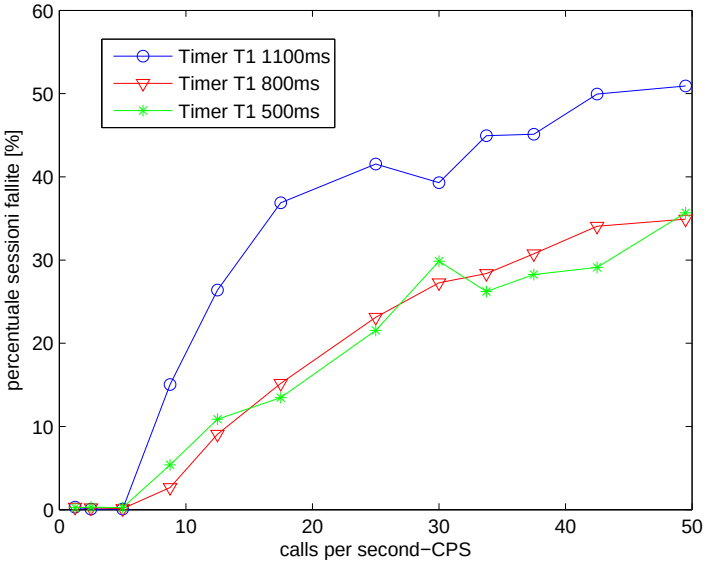

Figura 4.9- Andamento percentuale sessioni fallite, coda con priorità

4.2.3 Risultati con meccanismo TU

Il meccanismo TU dovrebbe evitare situazioni gravi di congestione per gli elementi SIP dello scenario, attraverso un sistema di feedback e di shaping del traffico. Le prestazioni del Distribuite Overload in scenari ad elevato ritardo sono molto al di sotto di quelle registrate in altri scenari caratterizzati da ritardi molto più contenuti. Per quanto riguarda i valori di CSD, le prestazioni non sono molto lontane da quelle registrate negli scenari precedenti.

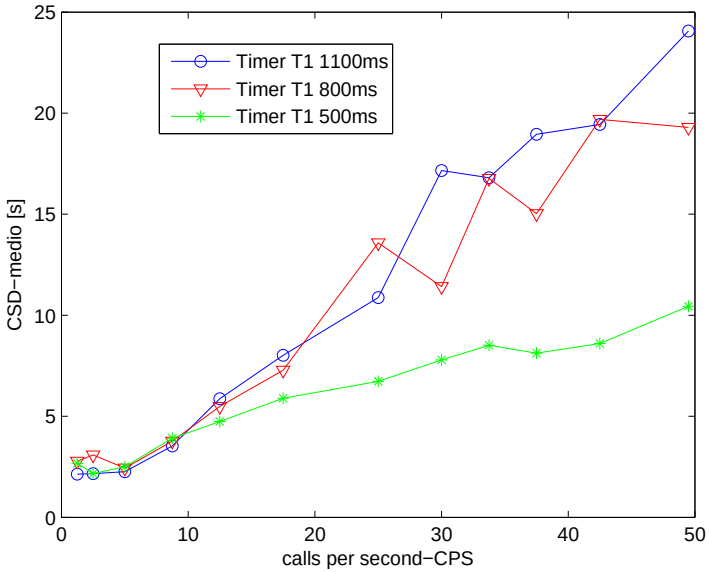

Figura 4.10- Grafico del CSD, coda meccanismo TU

I risultati in termini di sessioni al secondo e percentuali di sessioni fallite , invece risultano molto peggiori rispetto agli altri scenari. Il meccanismo TU risulta del tutto inadatto in questo scenario per i noti problemi di sovrastima del rate e per gli levati ritardi dello scenario che rallentano il sistema di feedback. I grafici che seguano mostrano una situazione desolante con valori inaccettabili per entrambi i parametri.

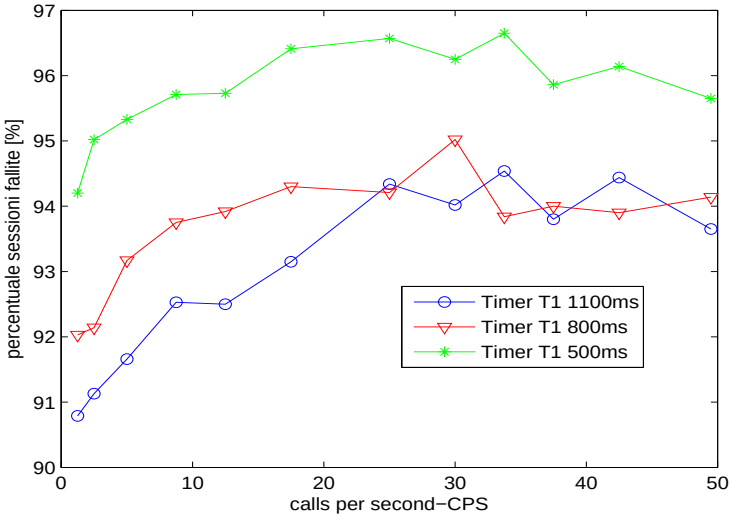

Figura 4.11- Andamento #sessioniOK/S, meccanismo TU

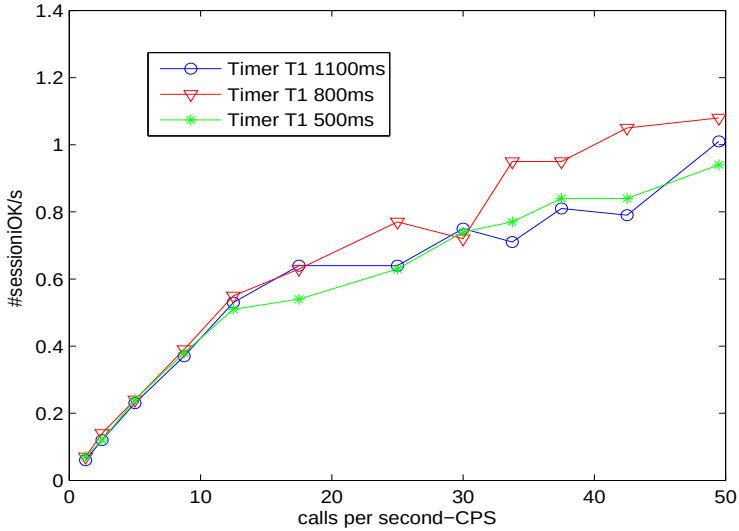

Figura 4.12- Andamento percentuale sessioni fallite, meccanismo TU

4.2.4 Scenari a confronto

Per comprendere meglio le differenze fra i vari scenari ho deciso di aggiungere un paragrafo dedicato ai confronti fra i vari i vari scenari analizzati. Il confronto è stato effettuato tra il meccanismo a priorità e il meccanismo TU, con Timer A posto a 500ms come da RFC 3261.

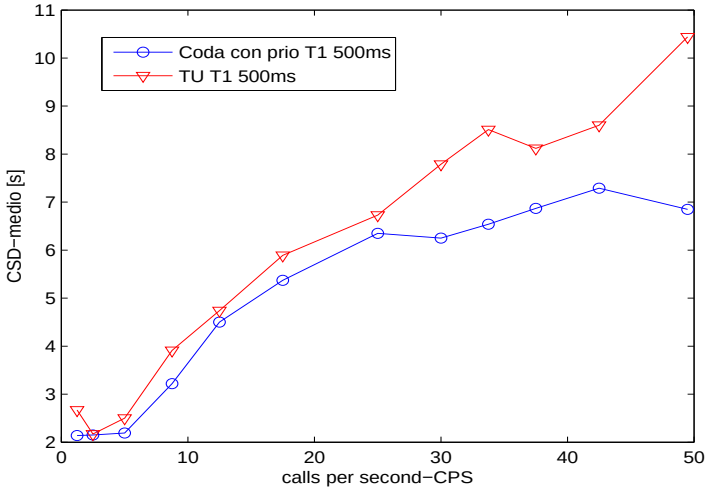

Figura 4.13- Grafico del CSD, sistemi a confronto

Se per quanto riguarda l'andamento del CSD non si riscontrano differenze sostanziali, per gli altri due parametri misurati il confronto risulta impari, e nettamente a favore del meccanismo a priorità.

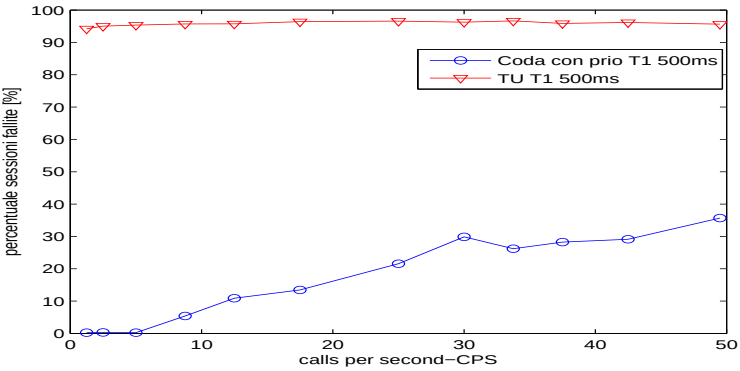

Figura 4.14- Andamento #sessioniOK/S, sistemi a confronto

Il meccanismo di Local Overload risulta molto costante per tutti i valori di traffico offerto , e su valori nettamente migliori.

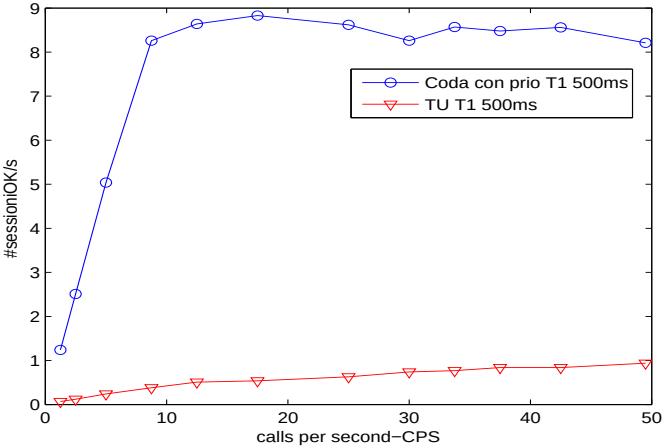

Figura 4.15- Andamento percentuale sessioni fallite, sistemi a confronto

Conclusioni

Il lavoro svolto ha confermato come il ritardo del link satellitare rappresenti l'ostacolo maggiore per il corretto settaggio di una comunicazione. Oltre ad incidere nel tempo di setup di una chiamata, cioè sul Call Setup Delay(CSD) il ritardo provoca un numero assai elevato di ritrasmissioni che vanno anche incidere sulle prestazioni degli elementi dello scenario, che si trovano a gestire un numero inaspettato di messaggi.

Per diminuire il numero di ritrasmissioni, si è deciso di modificare il Timer A rispetto ai valori di default definiti nella RFC 3261. Il Timer A è fondamentale per le ritrasmissioni, e quindi si è deciso di operare proprio su quello.

L' utilizzo di un Timer A molto elevato (1100 ms) con un valore molto maggiore rispetto al ritardo del link non ha migliorato le prestazioni, facendo aumentare anche la percentuale di sessioni fallite. Mentre un valore che ha ottenuto risultati interessanti è 800 ms, infatti se per il CSD ha ottenuto valori di poco superiori rispetto al valore di default, per gli altri due parametri misurati i risultati si sono mostrati molto buoni , e competitivi con il Timer A a 500 ms. Oltre a lavorare sui timer , abbiamo provato ad utilizzare sistemi di controllo dell'overload su questo scenario.

I sistemi di Local Overload si sono mostrati molto performanti e i risultati ottenuti non sono molto diversi da quelli ottenuti in scenari senza link satellitari. Dall'altra parte invece il sistema di Distribuited Overload, il meccanismo TU ha mostrato risultati scadenti su tutti i parametri misurati, mostrando cosi limiti importanti su link ad elevato ritardo.

Bibliografia

[1] Rosario G.Garroppo “Appunti di progetto e simulazione di reti di telecomunicazioni: Network Simulator vers.2 e sue applicazioni”

[2] Rosario G.Garroppo “Voice over IP: aspetti architetturali e di progettazione”

[3] Rosario G.Garroppo, Stefano Giordano, Saverio Niccolini, Stella Spagna “A prediction-based overload control algorithm for Sip servers”

[4] D.Adami, Rosario G.Garroppo, S.Giordano “Perfomance Evaluation of Moip Applications over Satellite: An Experimental Study”

[5] The VINT Project “NS, notes and documentation”

[6] V.Y.H Kueh, R.Tafazolli, B.G.Evans “Perfomance of VOIP Call Set-up Over Satellite-UMTS using SIP”

Ringraziamenti

Desidero ringraziare il prof. Rosario Garroppo e l'ing. Stella Spagna per il supporto e la disponibilità dimostrate in questo periodo di lavoro.

Un ringraziamento particolare va a Matteo, amico di lunga data con cui ho condiviso questo bellissimo periodo di studi,e che non mi ha mai fatto mancare il suo appoggio. Infine il grazie più sentito va ai miei genitori, a mia sorella Giada, ai miei nipoti e zii, che mi hanno sempre sostenuto in questi anni di studio.