Università degli Studi di Milano

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea in Informatica per le telecomunicazioni

Tesi di laurea

Sistemi mobili resilienti per la diffusione e l'immagazzinamento di dati pericolosi

Candidato:
Marco Guerra
Matricola 725154

Relatore:
Mattia Monga

Correlatore:
Andrea Trentini

Anno Accademico 2013

Indice

Capitolo 1

Introduzione

1.1 Introduzione

Col passare degli anni il ruolo delle telecomunicazioni è sempre più cruciale, al giorno d'oggi infatti la necessità di essere sempre in contatto con il mondo è più che evidente. Ciò può avvenire per svariati modi, dai più futili ai più nobili. La rete ha assunto un peso tale per cui il grado di sviluppo di un paese è determinato anche dalla sue infrastrutture di telecomunicazioni. Così nei paesi in cui non risulta possibile investire abbastanza risorse per la costruzione di tali infrastrutture finisco per divenire isolati al resto del mondo con notevoli ripercussioni negative, tra le quali: impoverimento culturale, impossibilità di ricevere soccorsi dinanzi a malattie, calamità naturali o colpi di stato, ed un enorme difficoltà nel contrastare eventuali censure.

Sebbene ad uno stato non ancora maturo, negli ultimi anni si è iniziato a sviluppare tecnologie in grado di mettere in piedi reti affidabili e al tempo stesso economiche. Ciò dona una possibilità anche a quei paesi in cui la rete rimane ancora un miraggio. Molto famoso è il caso di FabFi, un sistema di mesh networking su grande scala sviluppato da un gruppo di persone le quali sono riuscite a portare la rete in un paese afghano con ben 45 nodi su una distanza massima di 6 km e un throughput di 11.5Mbit/s e in un villaggio keniano, questa volta con 50 nodi su una distanza massima di 3.5 km e un throughput di ben 30Mbit/s. Queste due reti sono state costruite con un budget di appena 200 dollari. Gli esempi non si fermano qui, vi sono infatti moltre altre community (anche italiane) che verrano riprese successivamente. L'utilità della rete è stata messa in evidenzia anche nella vicenda delle rivolte in Egitto un paio di anni fa. Allora il governo decise di tagliare internet, oscurando al mondo interno ciò che realmente stava succedendo. Ciò è stato possibile semplicemente spegneno i nodi principali del paese. Per mezzo di una rete mesh [1] si sarebbe potuto mettere in piedi un sistema in grado di connettere gruppi di persone dotati di un semplice smartphone, di filmare gli avvenimenti e condivederli gli uni con gli altri in modo da avere una

ridondanza di dati tale da non preoccuparsi qualora una o più fonti dovesse essere distrutta e da uploadre su internet in un secondo momento o al tempo stesso, qualora uno o più dispositivi condividessero la propria connessione internet. È infatti possibile costituire una rete semplicemente tra i dispositivi che ne vogliono fare parte usando come nodi i dispositivi stessi. Lo scopo di questa tesi verterà proprio su questo progetto. [**?**]

1.2 Scopo del progetto

Si vuole quindi realizzare una rete decentralizzata di dispositivi in grado di inviare e immagazzinare dati da e verso tutti gli utenti di tale rete. L'esempio dell'Egitto potrebbe risultare esagerato, però lo stesso concetto è applicabile a contesti più piccoli come manifestazioni o sommosse. L'idea alla base è di cercare di connettere tutti i manifestanti ad una rete Internet indipendente, che non utilizzi dei server centrali i quali possono essere spenti in qualsiasi momento da parte di polizia o governi. Una volta connessi tutti i manifestanti, per mezzo di un sistema di mirroring è possibile replicare ciascuna registrazione a ciascun manifestante facente parte della rete. Questa ridondanza di dati rende molto difficile la censura dei contenuti registrati.

Si andrà inizialmente a costruire una rete mesh, cioè una particolare rete il cui compito di trasmettere e ricevere dati viene affidato ad ogni nodo che ne fa parte. In questo tipo di reti risulta particolarmente efficace la loro capacità di reagire e adattarsi dinanzi a malfunzionamenti o disconnessioni da parte di uno o più nodi. Facendo un riscontro con la realtà, è molto probabile che durante una sommossa o una manifestazione non propriamente pacifica, alcuni dispositivi possano avere problemi di questo tipo. è necessario quindi avere un'infrastruttura versatile, che possa garantire la continua comunicazione da e verso tutti i nodi. L'avere un'infrastruttura di rete economica, affidabile e come si vedrà in seguito, slegata dall'hardware rende le reti mesh perfette per un progetto di questo tipo.

Successivamente si implementarà il sistema per la condivisione e la registrazione dei file. Anche a questo livello verrà analizzato e usato solamente sofwtare open-source, per non venire meno ai principi cardini sopra elencati. Si darà maggior peso a tutti quei software indipendenti dalla piattaforma su cui vengono eseguiti, per permetterne l'utilizzo al maggior numero di dispositivi possibili.

Capitolo 2

Stato dell'arte

Verranno in seguito elecanti alcuni dei software presi in esame durante la realizzazione di questo progetto. Tutto il software preso in esame e utilizzato è unicamente open-source. Sebbene molti di essi siano progetti più o meno supportati e con un livello di sviluppo più o meno maturo o addirittura abbandonati, mi è sembrato comunque doveroso citarli anche per dare maggior risalto al tema delle reti libere e alle loro problematiche quali sicurezza e privacy. Le reti libere sono oggetto di molto interesse negli ultimi anni, tanto da dare il via alla nascita di numerose community proprio per promuoverne e facilitarne la diffusione.

2.1 Commotion

2.1.1 Descrizione

Commotion`http://commotionwireless.net/` è un tool open-source per dispositivi wireless, come computer e smartphone, per realizzare reti mesh. Permette inoltre di condividere il proprio punto di accesso Internet con gli atri utenti della rete. Inizialmente nacque come pacchetto aggiuntivo per openWRT, infine crebbe fino a poter essere compilato su diversi dispositivi. Offre uno livello di gestione e configurazione sopra OLSR. La sua interfaccia non è altro che un' estensione del sistema di configurazione di openWRT. L'obbiettivo principale è impedire il monitoraggio o l'interruzione di comunicazioni stabilendo connessioni peer-to-peer.

2.1.2 Livello di sviluppo

Durante il lavoro il software era in versione alpha, privo di documentazione e non disponibile pertanto non è stato possibile effetturare delle prove. Agisce a livello rete, usando il protocollo OLSR. Allo stato attuale delle cose si sono compiuti notevoli passi in avanti, tanto che l'intero progetto è presente su

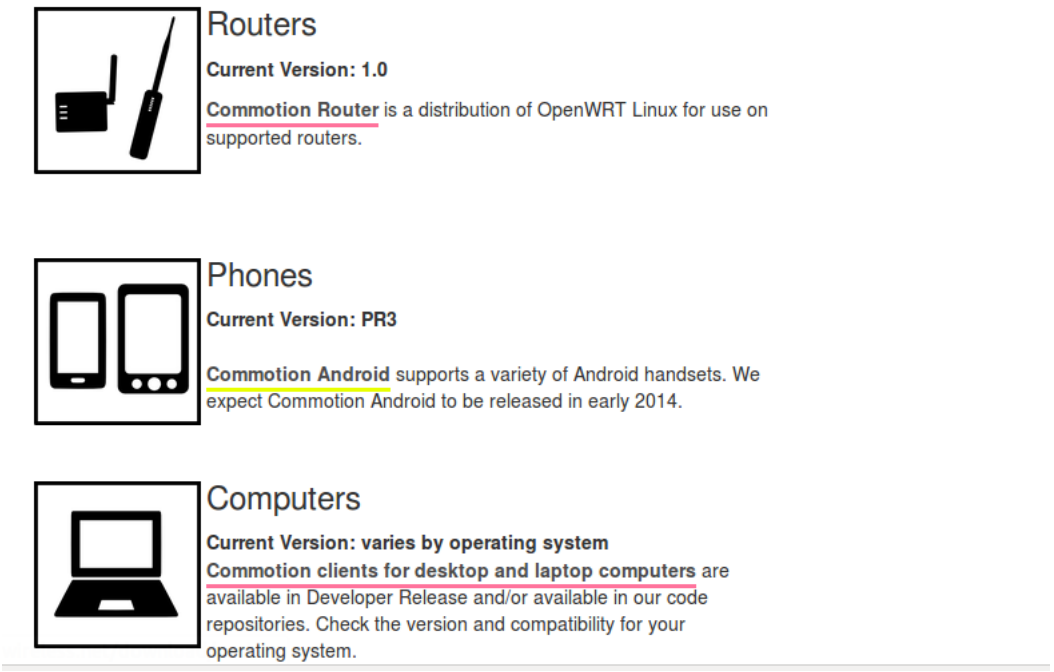

Figura 2.1: `http://commotionwireless.net`

github. Commotion è disponibile per diversi dispositivi e piattaforme tra cui router, smartphone android e computer ed è in continuo aggiornamento.

Rappresenta la più valida alternativa all'intero progetto anche se ha delle lacune in fatto di sicurezza poichè chiunque abbia un router commotion può accedere alla rete. Di seguito viene riportato un interessante articolo su cosa commotion non è in grado di fornire, insieme a delle alternative proposte dagli sviluppatori.
`http://commotionwireless.net/understanding-commotions-warning-label/`

2.1.3 Considerazioni

Al momento presenta delle lacune in fatto di sicurezza, è sconsigliato solamente per applicazioni critiche. Il progetto è sviluppato e seguito da numerose organizzazioni tra cui:

The Open Technology Institute
The Guardian Project
The Serval Project
Wireless Community Elvetica

Update e bugfix frequenti, documentazione del software ancora del tutto assente.

2.2 Freedom Box

2.2.1 Descrizione

Il progetto Freedom box verte sulla realizzazione di un sistema in grado di proteggere la privacy delle comunicazioni e di difendersi dalle intercettazioni. Software per questi scopi sono già esistenti, l'obbiettivo di freedom box è di integrarli in un piccolo plug computer in modo da rendere questi servizi di facile utilizzo. Una volta completato, si avrà un dispositivo in grado di rimpiazzare tutti i vari social network, mantenendo i propri contatti e al tempo stesso salvaguardando la privacy. Si potranno eseguire dei backup e salvare il tutto in un formato cifrato sulle freedom box di amici o comunque associate alla propria. Le freedom box potranno sempre comunicare le une con le altre anche per mezzo di path alternativi, aggirando quindi possibili ostacoli imposti dai provider, firewall o malfunzionamenti sulla rete. Le freedom box potranno inoltre comunicare con altre freedom box o telefoni tramite traffico Voip.

2.2.2 Livello di sviluppo

Allo stato attuale il software non è completo, ma comunque implementa un web proxy per migliorare la sicurezza e la privacy durante la navigazione web, un toolkit contenente un' interfaccia utente(Plinth) e un sistema di autenticazione(MonkeySphere)e dei tool per bloccare banner e i vari sistemi di tracking. Le freedom box possono già comunicare in maniera sicura servendosi di reti TOR. I vari box hanno già pieno supporto hardware.

2.2.3 Considerazioni

Freedom (`http://www.freedomboxfoundation.org/`)box è seguito e sviluppato da programmatori di tutto il mondo, molti dei quali arrivano dalla community di Debian. Bdale Garbee, project leader di Debian, è a capo del progetto e mantiene un repositoy in constante aggiornamento, addirittura più frequenti di commotion.

Sembra un progetto piuttosto ambizioso e a lungo termine. Non è stato possibile testare il tutto poichè legato a dell'hardware apposito sebbene sia possibile scaricare sia l'installer del progetto sia l'interfaccia grafica:

```
git clone git://anonscm.debian.org/freedombox/freedom-maker.git
git clone https://github.com/jvasile/Plinth.git
```

2.3 Serval mesh

2.3.1 Descrizione

Lo sviluppo di Serval mesh (`http://www.servalproject.org/`) verte principalmente due progetti: il primo consiste nella realizzazione di una rete mobile temporanea formata da antenne telefoniche e del tutto autosufficiente, impiegata in situazioni critiche come disastri dovuti a calamità naturali mentre il secondo è un sistema che permette di stabilire connessioni telefoniche tra cellulari dotati di wi-fi o eventualmente dispositivi dedicati, collegati ad una rete mesh. I due progetti sono facilmente integrabili.

2.3.2 Livello di sviluppo

Il software è disponibile solamente su terminali Android. Non è stato ancora testato sui tablet. In futuro verrano realizzati dei porting per IOS e Symbian

2.3.3 Conclusioni

Senza alcun dubbio è un progetto molto interessante, ci sono però due principali problemi: la già limitata durata della batteria dei vari smartphone viene ulteriormente abbattuta. Inoltre tutti i dati sensibili come telefonate e sms viaggeranno liberamente per la rete constituendo una grande minaccia alla privacy. L'applicazione è disponibile sull'app store di google `http://www.servalproject.org/`.

Purtroppo non è possibile scambiare dati all'infuori di messaggi o telefonate, e data la natura del progetto questa applicazione mostra i suoi limiti.

2.4 Open Mesh

2.4.1 Descrizione

Open mesh è una rete privata dei cittadini nata nel gennaio 2012, quando il governo egiziano ha deciso di bloccare l'intera rete internet. Non vi è traccia di ulteriori informazioni, il progetto sembra abbandonato. L'ultimo post nel forum interno risale a oltre due anni fa

2.5 olsrd: Optimized Link State Routing

2.5.1 Descrizione

OLSR[4] (`http://www.olsr.org/`)è un'implementazione di OLSR,un protocollo link state proattivo. All'interno del modello iso/osi si colloca a livello rete, col vantaggio di poter essere installato su molti dispositivi. Non è molto

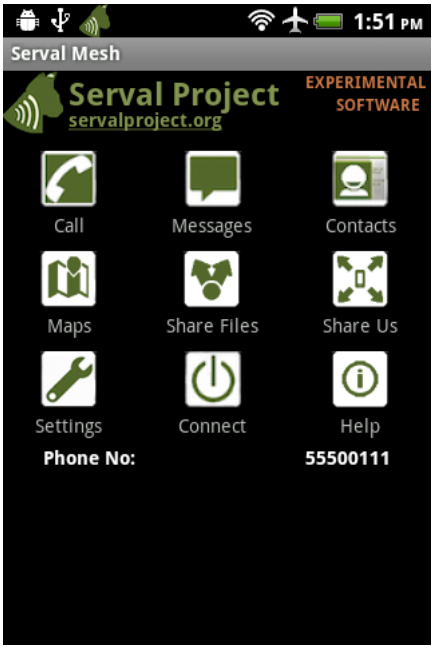

Figura 2.2: `https://play.google.com/store/apps/details?id=org.servalproject&hl=it`

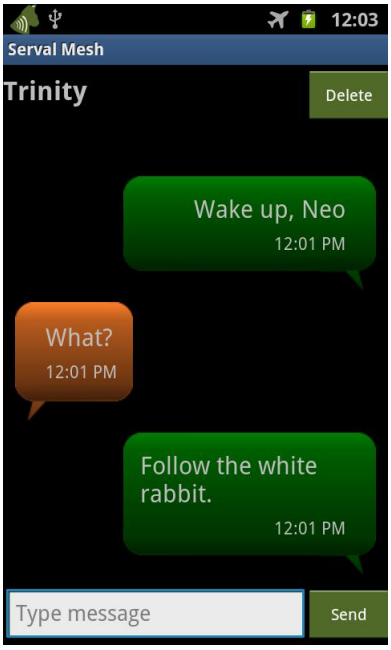

Figura 2.3: `https://play.google.com/store/apps/details?id=org.servalproject&hl=it`

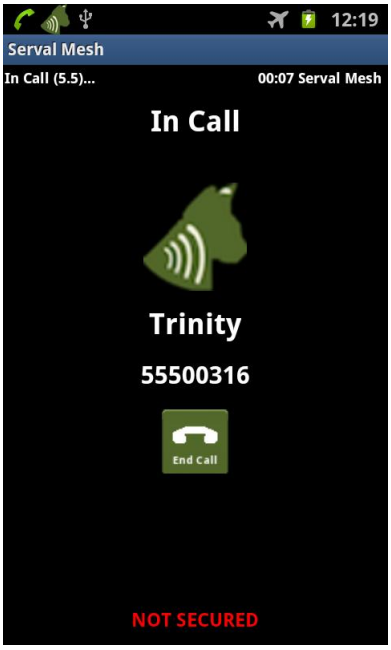

Figura 2.4: `https://play.google.com/store/apps/details?id=org.servalproject&hl=it`

oneroso in termini di consumo di cpu, caratteristica che lo rende pienamente compatibile con tutti quei dispositivi portatili con un'autonomia non troppo longeva e cpu poco potenti. All'attuale livello di sviluppo OLSR è considerato un protocollo molto solido, è comunque in continuo miglioramento e viene regolarmente sottoposto a numerosi stress test con ottimi risultati, arrivando ad essere usato in reti con fino 2000 nodi all'attivo. Dal momento che fa parte dello sviluppo di commotion, esiste una build sviluppata appositamente per android. Tuttavia il link a tale build è risultato essere sempre irraggiungibile.

2.6 B.A.T.M.A.N. (Better Approach To Mobile Ad-hoc Networking)

2.6.1 Descrizione

Lo standard attuale di routing per reti mesh è rappresentato da OLSR (Optimized Link State Routing) che usa un approccio link state: calcola tutti i percorsi e li distribuisce a tutti i nodi della rete. Batman (`http://www.open-mesh.org/projects/open-mesh/wiki`) invece usa un approccio distance vector: Ogni router mantiene solo le informazioni riguardanti il miglior path da se stesso verso gli altri nodi della rete. Grazie a questo

aspetto batman permette di usare router meno performanti e quindi meno costosi. è un protocollo sicuro e open source, e su reti con molti nodi risulta più efficiente di OLSR.

2.6.2 Livello di sviluppo

Negli anni il progetto ha riscosso parecchio interesse, cosìsi sono formati vari rami di sviluppo, tra i principali vi sono: Batmand: é stata la prima implementazione del protocollo. Si tratta di un deamon che opera a livello rete, modificando le tabelle di routing. Tutte le implementazioni di reti mesh sono nate agendo a livello di rete, e la maggior parte di esse sono rimaste fedeli a questo schema. Batman adv: Nel 2007 alcuni sviluppatori cominciarono a sperimentare l'idea a livello data-link, cosìnacque una nuova versione del protocollo, che prese il nome di batman adv (advanced) per distinguerlo dalle altre versioni. All'inizio fu sviluppata sia un'interfaccia virtuale user space sia un modulo del kernel. L'interfaccia virtuale venne successivamente rimossa a causa dell'eccessivo overhead dei nodi. il software è open source, disponibile per linux e come estensione di openWRT.

2.6.3 Considerazioni

è un progetto collaudato e in continuo aggiornamento. è presente molta documentazione.In reti non troppo magliate risulta meno performante di OLSR, ma resta in ogni caso un'ottima alternativa.

2.7 Roofnet

2.7.1 Descrizione

Roofnet (`http://www.comclub.org/roofnet/`) è una rete mesh sperimentale sviluppata dal Massachusetts Institute of Technology che fornisce accesso ad Internet con banda larga a Cambridge e ha all'attivo circa 20 nodi. Per installare RoofNet è necessario Linux, una scheda Atheros AR521x o un Access point Netgear WGT634U e il software scaricabile dal sito. Viene anche fornito un kit al prezzo di 650 dollari comprensivo di un computer con del software preinstallato, una scheda wireless e un'antenna. In questo modo non si ha bisogno di alcuna configurazione, permettendo una facile diffusione. I nodi sono basati su Linux Red Hat 9. Il protocollo di routing è SrcRR, che prende ispirazione da dsr. Anche questo progetto quindi, opera a livello rete.

2.7.2 Livello di sviluppo

Il progetto è datato al 2006 e da allora non si hanno più notizie riguardanti aggiornamenti o nuove features. Si sono ottenute comunque delle ottime

performance: Sul percorso più lungo (4 hop su un area di 1.25 miglia) si ha in media una latenza di circa 12 ms con un throughput di 120 Kb/s.

2.8 Wing

2.8.1 Descrizione

Wing (Wireless Mesh Network for Next-generation Internet) (`http://www.wing-project.org/doku.php`è un progetto sponsorizzato del ministero dell'istruzione italiano e guidato da CREATE-NET (Center for REsearch And Telecomunication Experimentation for NEtworked communities) e Technion (Istrael Institute of Technology). Il team composto da questi due istituti svolge ricerche su algoritmi e protocolli per reti wireless. Parte del software sviluppato è un'estensione di RoofNet.

2.8.2 Livello di sviluppo

Wing aggiunge al progetto originale di RoofNet supporto per multiple radio interfaces, WCETT routing e un assegnamento dei canali autonomo. Oltre alla tecnologia WiFi, sono supportati sia WiMax che UMTS. Il software al momento è parecchio instabile, non ci sono informazioni su futuri aggiornamenti.

2.8.3 Considerazioni

Ottima la possibilità di essere usato anche su reti WiMax e UMTS, si sa poco sugli sviluppi futuri.

Capitolo 3

Descrizione architettura

Al giorno d'oggi sono molteplici i dispotivi portatili in grado di effettuare delle registrazioni e di connettersi a delle reti wireless. Si presuppone che l'utente abbia tale dispositivo, compatibile con il software che verrà trattato successivamente. Mettere in comunicazione tutti questi dispositivi costuisce il primo step: La realizzazione della rete. Si ha qundi bisogno di una rete inanzitutto wireless e decentralizzata: Una rete mesh[1]. Queste due caratteristiche portano ad avere due grandi vantaggi: In primo luogo una rete di questo tipo è molto economica, infatti non sono necessari né cavi né server. In secondo luogo c'è il fattore affidabilità: Data l'assenza server centrali, tutti i nodi della rete sono direttamente collegati gli uni con gli altri, ognuno dei quali funge da ripetitore. Se un nodo dovesse guastarsi, la rete non ne risentirebbe poiché riorganizzerebbe il routing bypassando il guasto, in modo da instradare comunque i dati. Dal momento che tutti parlano con tutti, se un nodo ha a disposizione un'accesso ad Internet, può condividere la propria connessione col resto della rete. Grazie a queste particolarità sono partiti molti progetti per rendere Internet usufruibile anche dai paesi del mondo più poveri e isolati. Il seguente articolo Building a Rural Wireless Mesh Network mostra come è stato possibile portare internet in un villaggio africano. Costituita la rete, è necessario implementare un sistema che permetta ai dispositivi di poter inviare e ricevere file. Tutto ciò può essere fatto tramite un file system distribuito[2] o per mezzo di un torrent[3]. Per problemi che verranno illustrati successivamente, si è scelto un file system distributo. Per nostra fortuna, la rete è piena di programmi di questo tipo, per di più open source. Ai fini del progetto, si cerca un file system con le seguenti caratteristiche:

- Mirroring il più esteso possibile
- Possibilità di lavorare anche offline
- Bandwidth optimized

- Poco esigente dal punto di vista hardware, in modo da aumentare la scalabilità per più dispositivi
- Ben supportato, con un community attiva

Poichè la rete è pensata per accettare un numero variabili di dispositivi, una volta scelto il file system è necessario configurarlo dinamicamente a seconda del numero dei dispositivi connessi in un dato momento. Per far ciò si avrà bisogno di implementare un meccanismo di discovery e di assegnamento degli indirizzi[3]. Arrivati a questo punto non si dovrà fare altro che installare gli opportuni programmi per quanto riguarda la registrazione audio e video.

3.1 Problemi aperti

Allo stato attuale del software disponibile i principali problemi sono legati a diversi fattori. In primo luogo, come si vedrà successivamente in dettaglio, la banda a disposizione è veramente poca e, al crescere del numero dei dispositivi connessi(e quindi alla ridondanza dei dati), le performance precipitano. In secondo luogo, i file system distribuiti ad oggi disponibili non sono pensati per avere a monte una rete così versatile. é necessario infatti aggiornare ogni file di configurazione su ogni dispovitivo ogni volta che avvengono dei cambiamente nella rete. A livello implementativo, ciò si riduce a fermare il programma in esecuzione, aggiornarlo con i nuovi parametri e riavviarlo. Di conseguenza anche le registrazioni verranno temporaneamente fermate per una durata almeno pari al tempo necessario per il discovery dei nuovi dispositivi e il rimontaggio del file system distribuito. Un ulteriore problema è dovuto al fatto che non tutti i dispositivi hanno uguale importanza. Il modo più sicuro per assegnare indirizzi ip è tramite l'utilizzo di un server dhcp. Per la natura stessa del progetto è impensabile assegnare un indizzo statico ad ogni dispositivo. Ciò significa che si avranno una o più macchine server a fronte di tante altre client, ed è chiaro che riuscire a bloccare tali macchine significa bloccare l'intero sistema. Lo stesso discoso è applicabile a qualsiasi programma si userà per montare il file system. A causa della limitata banda a disposizione non è possibile aumentare la ridondanza delle macchine server, per lo meno su reti non cablate.

Capitolo 4

La rete mesh

4.1 Protocolli di rete

Esistono due tipi di protocolli di routing

4.1.1 Distance vector

Distance vector [6]è un algoritmo distributo che permette ad ogni router di costruire una tabella di routing contenente il costo e la distanza(in termini di hop) del path per raggiungere tutti gli altri nodi della rete. Inizialmente ogni router conosce solo l'identità dei router a cui esso è direttamente collegato, informazioni solitamente inserire dall'amministratore di rete. A degli intervalli di tempo regolari, ogni nodo invia la propria tabella di routing ai suoi nodi adiacenti. Se una distanza riportata dovesse essere minore di quella corrente, la tabella di routing verrà aggiornata con il nuovo path. Iterando l'operazione, ogni router potrà determinare il cammino minino per raggiungere tutti gli altri nodi della rete.

Vantaggi e svantaggi di distance vector

Questo tipo di algoritmo è molto semplice da realizzare. Grazie alle minime informazioni scambiate tra i nodi, la rete non viene sovraccaricata.
I nodi non conoscono l'intera topologia della rete, ma solo il modo migliore per arrivare da un punto all'altro e più la rete è grande, più è difficile individuare eventuali problemi.
Il problema principale è dato dalla possibile comparsa di cicli dovuti a qualche variazione della rete: Se in una rete formata dai nodi A-B-C-D, il collegamento A-B dovesse guastarsi, B cercherebbe di instradare i vari dati ad A passando attraverso C, poiché dalla sua tabella di routing, C dista due hop da A (ma deve passare per B). In ogni caso però, il collegamente A-B è interrotto e quindi A non può essere raggiunto nemmeno da C. Questo problema è noto come *count-to-infinity*.

4.1.2 Link state

[7]Proprio come Distance vector, iniziamente ogni nodo conosce solo le informazioni degli altri router a lui collegati direttamente. A regolari intervalli di tempo, ogni nodo invia in broadcast a tutti gli altri nodi della rete le informazioni relative alla propria identità e dei suoi cammini adiacenti. Ogni nodo aggiornerà la propria tabella di routing solo se i numeri di sequenza dei pacchetti ricevuti saranno più recenti di quelli correnti. Una volta che le tabelle di routing conterranno l'intera topologia della rete, ogni router eseguirà un altro algoritmo (algoritmo di Dijkstra)per determinare i cammini minimi.

Vantaggi e svantaggi di link state

È più difficile da realizzare. Contrariamente all'algoritmo di distance vector, è molto difficile trovare dei loop ed in ogni caso è possibile fermarli. È adatto a gestire reti di grandi dimensioni ed è più veloce nel memorizzare la topologia delle rete e i suoi cambiamenti.

4.2 Protocolli di routing per reti mesh

È possibile dividere gli algoritmi appena descritti in due ulteriori categorie sulla base del momento in cui vengono calcolati i percorsi.

4.2.1 Algoritmi proattivi

Un protocollo proattivo mantiene aggiornati tutti i cammini ad intervalli di tempo regolari, a prescindere se verranno utilizzati o meno. Così facendo, si avrà sempre a disposizione un cammino pronto per ogni richiesta di instradamento a scapito di immettere traffico sulla rete anche quando non sarebbe richiesto.

4.2.2 Algoritmi reattivi

Un protocollo reattivo calcola i cammini solo quando un pacchetto deve essere trasmesso, in questo modo il traffico sulla rete viene notevolmente ridotto ma i tempi di consegna del pacchetto stesso risultano più lunghi. Questo tipo di protocollo si presta meglio per applicazioni con una banda limitata, come il progetto preso in esame.

4.2.3 Algoritmi ibridi

Un protocollo ibrido unifica i vantaggi dei protocolli proattivi con quelli reattivi. Ogni nodo calcola proattivamente solo i cammini verso i suoi nodi

vicini, mentre i più lontani vengono calcolati reattivamente, solo quando richiesto.

4.3 Installazione pacchetti necessari

Si è optato per l'utilizzo di batman-adv [3]sia perchè viene incontro alle esigenze del progetto sia per le sue particolari feature in costante sviluppo e miglioramento.

Ciò che contraddistingue maggiormente batman-adv è che non opera a livello rete come gli altri algoritmi ma a livello data-link. Questo significa che all'effettivo tutti i nodi risultato collegati fisicamente gli uni con gli altri portando un notevole vantaggio soprattutto in fase di configurazione col crescere del numero dei nodi poichè non affetti da eventuali cambiamenti della topologia della rete. Grazie a questa particolarità, un nodo può entrare a fare parte della rete anche se sprovvisto di un indirizzo ip, sebbene ne avrà comunque bisogno se dovrà implementare dei servizi aggiuntivi (o esserne parte), per lo meno a livello rete o superiore. Al di sopra di batman-adv è quindi possibile usare qualsiasi tipo di protocollo.

Batman ha inoltre una serie di feature aggiuntive molto utili, prima fra tutte *batctl*. batctl è un tool che offrè funzionalità di monitoraggio e configurazione. Nel corso del progetto questo tool verrà usato più volte, sia in fase di configurazione delle interfacce di rete sia per misurazioni di prestazioni.

Per questi motivi la scelta del protocollo di routing è ricaduta su batman: un protocollo open-sorce stabile, in continuo aggiornamento e con delle feature particolarmente interessanti, completamente slegato dall'hardware e adatto ad essere integrato col altri protocolli.

4.4 Installazione pacchetti necessari

È possibile scaricare batman-adv e batctl dal proprio gestore di pacchetti, ma è consigliabile scaricare le ultime versioni aggiornate al seguente indirizzo:

```
http://www.open-mesh.org/projects/open-mesh/wiki/Download
```

Una volta scaricati gli archivi, si procede alla loro estrazione

```
tar -zxvf batman-adv-2013.1.0.tar.gz
tar -zxvf batctl-2013.1.0.tar.gz
```

A questo punto, per procedere con l'installazione è sufficiente digitare

```
make
```

e

```
make install
```

in ognuna delle due directory.

Sono disponibili package sulla stragrande maggioranza delle distribuzioni linux, in particolare la versione disponibile su ubuntu risulta essere aggiornata almeno alla penultima versione attuale. Per quanto riguarda debian invece il package risulta ormai obsoleto, tanto che mancano diverse funzionalità aggiuntive come batctl. C'è da dire che non si sono riscontrati problemi con versioni diverse del software.

4.5 Creazione della rete

Come prima cosa occorre disattivare il network manager per evitare che interferisca con la rete che si andrà a creare. Come verrà spiegato successivamente, quando si riattiveranno le interfacce di rete e si abiliterà la nuova interfaccia bat0, il network manager cercherà di usare la sua interfaccia prefefinita, tipicamente wlan0 o eth1 disconnettendosi dalla rete. Dal momento che si tratta di un componente non necessario, disabilitandolo si evita di incappare in questo tipo di problema.

```
# service network-manager stop
```

È necessario inoltre caricare nel kernel il seguente modulo

```
# modprobe batman-adv
```

A questo punto è possibile creare la rete. Batman-adv permette di utilizzare qualsiasi interfaccia, persino quella bluetooth. L'interfaccia che sarà maggiormente presa in considerazione è la wireless, la più adatta al progetto. È opportuno quindi individuarne il nome

```
# iwconfig
```

In questo caso è wlan0, come si può vedere dall'output

```
lo        no wireless extensions.

virbr0    no wireless extensions.

vboxnet0  no wireless extensions.

wlan0     IEEE 802.11bgn  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=15 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
```

```
eth0      no wireless extensions.
```

Fatto ciò, si può procedere alla creazione della rete

```
# ifconfig wlan0 down
# ifconfig wlan0 mtu 1528
# iwconfig wlan0 mode ad-hoc
# iwconfig wlan0 channel 1
# iwconfig wlan0 enc off
```

E le assegnamo un nome

```
# iwconfig wlan0 essid nome_rete_mesh
```

A questo punto bisogna assegnare al protocollo l'interfaccia di rete che dovrà usare

```
# batctl if add wlan0
```

L'interfaccia wlan0 dovrebbe essere stata aggiunta e risultare ancora inattiva

```
batctl if
```

Come ultimo passo è necessario assegnare gli indirizzi (manualmente o tramite dhcp) e attivare wlan0. Dal momento che batman-adv lavora a livello data-link, è sufficiente impostare l'indirizzo di bat0: Un'interfaccia virtuale creata dal protocollo sul quale verrano inviati i dati.

```
# ifconfig wlan0 up
# ifconfig bat0 up
# ifconfig bat0 192.168.2.1 netmask 255.255.255.0 up
```

Questo procedimento deve essere eseguito su tutti i nodi della rete. Se tutto è andato a buon fine è possibile verificare la presenza dei vari nodi

```
# batctl o
```

Un esempio di output

```
[B.A.T.M.A.N. adv 2013.1.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    1.456s   (226) 00:15:af:88:52:dd [     wlan0]: 00:15:af:88:52:d
```

E pingarli per verificarne la connessione attraverso una funzionalità di batctl

```
# batctl ping 00:15:af:88:52:dd
```

Restituirà

```
PING 00:15:af:88:52:dd (00:15:af:88:52:dd) 20(48) bytes of data
20 bytes from 00:15:af:88:52:dd icmp_seq=1 ttl=50 time=2.22 ms
20 bytes from 00:15:af:88:52:dd icmp_seq=2 ttl=50 time=4.05 ms
20 bytes from 00:15:af:88:52:dd icmp_seq=3 ttl=50 time=2.31 ms
20 bytes from 00:15:af:88:52:dd icmp_seq=4 ttl=50 time=2.25 ms
20 bytes from 00:15:af:88:52:dd icmp_seq=5 ttl=50 time=2.20 ms
20 bytes from 00:15:af:88:52:dd icmp_seq=6 ttl=50 time=3.61 ms
20 bytes from 00:15:af:88:52:dd icmp_seq=7 ttl=50 time=2.22 ms
...
```

Batman-adv considererà un nodo inattivo solo quando il suo campo last-seen raggiungerà un valore di 200 secondi.

4.6 Assegnazione automatica degli indirizzi

Allo stato attuale delle cose la rete è up e perfettamente funzionante. Si è dovuto però settare gli indirizzi ip di ciascuna manualmente, operazione che poco si addice ad una rete di questo tipo. È impensabile infatti, che ciascun utente debba accordarsi a priori sull'indirizzo che dovrà usare. In questo caso entra in gioco il protocollo dhcp, che permette di assegnare automaticamente a qualsiasi dispositivo facente parte della rete un indirizzo ip nel range specificato.

4.6.1 Funzionamento

dhcp [5]non è altro che un protocollo client-server. Tramite lo scambio di quattro messaggi è possibile per il client accordare un indirizzo ip in maniera automatica:

1. Il client manda in broadcast un DHCP DISCOVER con source address l'indirizzo 0.0.0.0 . Il messaggio contiene inoltre un transaction id per accoppiare ad ogni richiesta la relativa risposta.

2. Uno o più server possono rispondere con un DHCP OFFER il quale contiene otre allo stesso transaction id del dhcp discover l'indirizzo ip proposto, l'indizzo della maschera e la durata della validità di tale ip.

3. A questo punto il client da conferma al server per mezzo di un DHCP REQUEST. Questo messaggio contiene semplicemente gli stessi parametri del dhcp offer.

4. Infine il server manda un'ulteriore conferma al client tramite un DHCP ACK.

Questo sistema di doppia conferma è stato pensato principalmente come soluzione a due possibili problemi: Supponendo di avere qualche tipo di

problema sulla ricezione di DHCP OFFER il client rimanderà un DHCP DISCOVER e il server cercherà di assegnargli un nuovo indirizzo ip. In questo caso avremmo quindi uno spreco di indirizzi. Ma se al server non arriva un DHCP REQUEST, assegnerà al client lo stesso indirizzo. Nel caso di più dhcp server sulla rete, ciascunt client riceverà più DHCP OFFER. Fra questi dovrà confermarne solo uno per mezzo di un DHCP REQUEST. La destinazione di questo messaggio è ancora broadcast, al fine di inviare la comunicazione a tutti i server della rete.

4.7 Installazione e configurazione di dhcp server

Per prima cosa è necessario scaricare dhcp server dal proprio gestore dei pacchetti

```
# apt-get install isc-dhcp-server
```

A questo punto verrà creato il file di configurazione /etc/dhcp/dhcpd.conf È necessario editare le seguenti voci:

```
  # gedit /etc/dhcp/dhcpd.conf
```

```
ddns-update-style none;
```

ddns-update ha tre possibili valori:

- ad-hoc: deprecato
- interim: permette al server dhcp di aggiornare il server dns ogni volta che viene rilasciato un indirizzo ip in modo da sapere a quali computer sono associati gli indirizzi ip dati. Affinché ciò funzioni è necessario che il server dns sia di tipo dinamico
- none: se il server dns non è dinamico o non si vuole usufruire della precedente feature bisognerà settare questo valore a none

```
authoritative;
```

senza questa direttiva i client non sarebbero in grado di ottenere un indirizzo ip dopo aver cambiato la subnet fino a quando i loro vecchi indirizzi non saranno scaduti. Infine si imposta una subnet con relativa netmask e il range di indirizzi da assegnare

```
subnet 192.168.0.0 netmask 255.255.255.0 {
## dhcp start  and end IP range ##
range 192.168.0.11 192.168.0.100;
option routers 192.168.0.1;
}
```

È possibile inoltre settare il lease time degli indirizzi assegnati modificando la seguente voce:

```
default-lease-time 3600;
```

Come ultimo passo bisogna impostare su quale interfaccia gli indirizzi verranno assegnati. Si usera l'interfaccia virtuale bat0 creata da batman adv. Si deve quindi editar il seguente file

```
/etc/default/isc-dhcp-server
```

nel seguente modo

```
INTERFACES="bat0"
```

Non rimane altro che avviare il server

```
# service isc-dhcp-server start
```

4.8 Ottenere un indirizzo ip

Per ottere un nuovo indirizzo è necessario inanzitutto essere connessi alla rete, successivamente digitare:

```
dhclient -r
```

Per rilasciare l'attuale indirizzo ip e fermare il processo dhclient e

```
dhclient bat0
```

per richiedere il nuovo indirizzo sull'interfaccia bat0. È possibile verificare il corretto funzionamento lanciando il comando

```
ifconfig bat0
```

L'output ottenuto dovrebbe essere di questo tipo

```
ifconfig bat0
bat0      Link encap:Ethernet  IndirizzoHW 52:43:8d:74:8f:d4
          indirizzo inet:192.168.0.11  Bcast:192.168.0.255  Maschera:255.255.255
```

dove l'indirizzo di bat0 deve essere compreso nel range settato all'interno del file (server side)

```
/etc/dhcp/dhcpd.conf
```

Capitolo 5

Il file system distribuito

5.1 Tahoe

Tahoe [1]è un file system distribuito open source ad alta affidabilità e sicurezza.

> *Tahoe-LAFS is a Free and Open cloud storage system. It distributes your data across multiple servers. Even if some of the servers fail or are taken over by an attacker, the entire filesystem continues to function correctly, including preservation of your privacy and security.*

La maggior parte degli storage provider non fanno altro che sincronizzare sui loro server una directory locale selezionata dall'utente. Ogni provider garantisce la sicurezza sui dati a loro inviati, sui quali hanno tutti i permessi di lettura e scrittura. In realtà ciò che garantiscono è il limitato uso di questo potere. Usando Tahoe è possibile ottenenere i seguenti benefici:

- I file vengono criptati prima di essere memomorizzati su disco, quindi anche prima di essere mandati in cloud. In questo modo nessuno potrà avere accesso a questi dati senza il proprio consenso.
- Si può configurare il livello di mirroring su più storage provider, in modo da avere sempre la possibilità di recuperare i propri dati.
- È possibile massimizzare lo spazio disponibile fra tutti gli storage provider in modo da avere un unico e grande servizio di cloud.

La questione delle sicurezza è cruciale per ogni servizio di storage. La maggior parte di essi dichiara i propri sistemi sicuri,promettendo che i dati personali non verranno letti o modificati in alcun modo, pur avendonone la possibilità. Va inoltre tenuta in considerazione la possibilità della diffusione o modificati di tali dati a causa di bug o attacchi esterni. Con tahoe, il service provider non ha proprio modo di leggere o modificare alcun tipo di

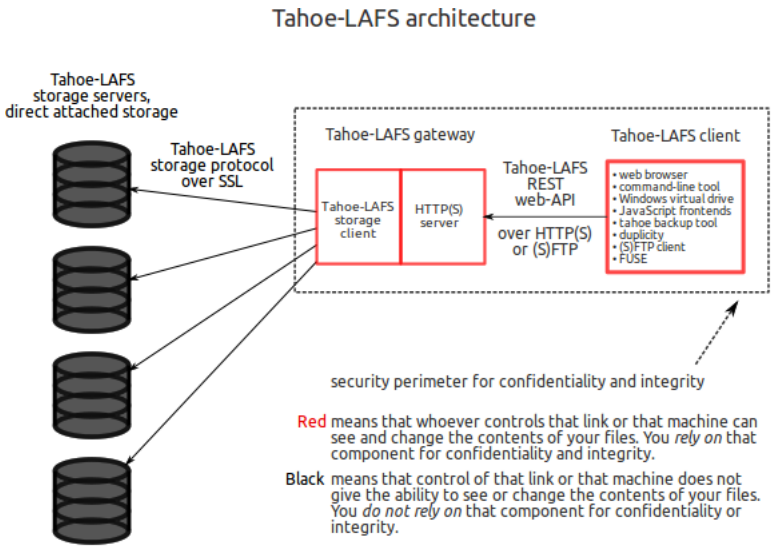

Figura 5.1: `https://tahoe-lafs.org/trac/tahoe-lafs`

dato. Per questo motivo in tahoe viene definito il concetto di *Provider-independent security*. Grazie a questa funzionalità si può fare a meno di criptare i dati prima di archiviarli. fonte figura?

5.1.1 Funzionamento

Una *Storage grid* è formata da vari *Storage server*. Un gateway comunica con questi server e fornisce loro accesso al file system tramite procolli HTTP(S), SFTP o FTP. La funzione di gateway può essere svolta da dei nodi client(che si comportano come client verso gli storage servers), da processi o altri programmi che si connettono a un gateway come per esempio Web browser, client SFTP o FTP. Gli utenti non si affidano agli storage server nè per l'*integrità* nè per la *confidenzialità* dei propri dati ma al gateway che fornisce funzionalità di criptazione e integrity-check. In una tipica implementazione il gateway risiede sulla propria macchina, pertanto è su questa che viene affidata l'integrità e la confidenzialità. Proprio Per questi motivi un server non può nè leggere nè modificare alcun file. Si affida al server solamente la disponibilità dei dati. Una possibile alternativa consiste nell'usare un gateway su una macchina remota e di connettersi ad essa tramite HTTPS o SFTP. Così facendo l'amministratore di tale macchina avrà accesso a tutti i dati, ma l'utente potrà accedere al file system anche da dispositivi privi dell'istallazione di un gateway come per esempio uno smartphone.

I dati vengono condivisi N volte su almeno H storage server affinchè possano essere recuperati da ognuno di questi K server (di default N = 10,

H = 7 e K = 3). In questo modo solamente un guasto di H-K+1 server può rendere i dati irrecuperabili.

5.2 Controllo degli accessi

Controllo degli accessi

Quando si effettua un upload di un file, si ha la possibilità di rendere tale file:

- Immutable: Dopo essere stato uploadato un file di questo tipo non può essere più modificato.
- Mutable: Dopo essere stato uploadato un file di questo tipo può essere modificato solo dagli utenti aventi i permessi di lettura e scrittura.

Un utente con i permessi di lettura e scrittura su un file mutable può passare ad un altro utente i suoi stessi permessi, oppure concedere solamente l'accesso in sola lettura. Un utente abilitato alla sola lettura invece, potrà passare solamente permessi di questo tipo.

5.3 Installazione e configurazione

La versione più recente di tahoe è disponibile al seguente indirizzo

`https://tahoe-lafs.org/source/tahoe-lafs/releases/allmydata-tahoe-1.10.0.zip`

È necessario installare *python2.7*o superiore

`http://www.python.org/download/releases/2.7.4/`

Per procedere con l'installazione è sufficiente scompattare l'archivio e digitare

```
python setup.py build
```

Il programma in sé non fa altro che creare, avviare e stoppare dei nodi (*nodes*). Ogni nodo legge e scrive dei file, sono collocati inoltre in opportune directory, ognuna delle quali contenente un file di configurazione chiamato tahoe.cfg.*Storage nodes* e *client nodes* formano una *grid*. Gli *introducer nodes* servono per far comunicare i vari nodi gli uni con gli altri. Per creare una propria *grid* sono necessari degli storage e introducer node. Tahoe mette a disposizione una *public test grid* come piattaforma di prova. Si avrà bisogno solamente di creare un client node.

```
bin/tahoe create-client
```

Un client node non condivide alcuno spazio con gli altri nodi. Verrà creata la directory /.tahoe contente il seguente file di configurazione tahoe.cfg

```
# -*- mode: conf; coding: utf-8 -*-

# This file controls the configuration of the Tahoe node that
# lives in this directory. It is only read at node startup.
# For details about the keys that can be set here, please
# read the 'docs/configuration.rst' file that came with your
# Tahoe installation.


[node]
nickname =
web.port = tcp:3456:interface=127.0.0.1
web.static = public_html
#tub.port =
#tub.location =
#log_gatherer.furl =
#timeout.keepalive =
#timeout.disconnect =
#ssh.port = 8022
#ssh.authorized_keys_file = ~/.ssh/authorized_keys

[client]
# Which services should this client connect to?
introducer.furl =
helper.furl =
#key_generator.furl =
#stats_gatherer.furl =

# What encoding parameters should this client use for uploads?
#shares.needed = 3
#shares.happy = 7
#shares.total = 10

[storage]
# Shall this node provide storage service?
enabled = false
#readonly =
reserved_space = 1G
#expire.enabled =
#expire.mode =
```

```
[helper]
# Shall this node run a helper service that clients can use?
enabled = false

[drop_upload]
# Shall this node automatically upload files created or modified in a local directo
enabled = false
# To specify the target of uploads, a mutable directory writecap URI must be placed
# in 'private/drop_upload_dircap'.
local.directory = ~/drop_upload
```

5.4 Public test grid

Chiamata anche Pubgrid, è uno spazio di archiviazione in remoto fornito da tahoe per testare il suo corretto funzionamento. Proprio per questo motivo lo spazio è limitato e non dovrebbe essere usato come storage di dati importati. Per connettersi al pubgrid è necessario editare le seguenti voci all'interno del file di configurazione tahoe.cfg

```
[node]
nickname = un nome qualsiasi

[client]
introducer.furl = pb://hckqqn4vq5ggzuukfztpuu4wykwefa6d@publictestgrid.dnsd.info:502
```

A questo punto si può lanciare tahoe, sono necessari i permessi di root

```
# bin/tahoe start
```

Si è quindi connessi alla grid. È possibile inoltre connettersi ad un server web che mette a disposizione un'interfaccia grafica per la gestione dei file tramite il seguente indirizzo

```
http://127.0.0.1:3456
```

5.5 File system manipulation

Tahoe mette a disposizione una serie di comandi che forniscono le seguenti funzionalità:

- list
- upload
- download
- unlink

- rename
- mkdir

Ogni nodo ha un proprio indirizzo contenuto in /.tahoe/node.url . Tutti i comandi usano questo file per sapere quale nodo usare, se si vuole infatti usare lo stesso su una macchina diversa è sufficiente creare la directory /.tahoe e incollare questo file. In questo modo si usarà il nodo in questione invece di quello locale. L'architettura di tahoe è sostanzialmente un insieme di directory e file che formano un grafo. Per usare tutto ciò è necessario fissare un punto di partenza dal momento che non esiste alcuna directory radice. Per creare uno *starting point* bisogna creare un *alias*. Tutti gli alias risiederanno in directorybase/private/aliases. Usando il comando tahoe ls si avrà la lista di tutti i file a partire dall'alias di default: tahoe. È possibile specificare un alias in particolare: tahoe ls nomealias. Scrivere quindi tahoe ls tahoe avrà quindi lo stesso effetto di tahoe ls. A questo punto quindi si dovrà creare uno starting point e settarlo come scelta di default

```
# bin/tahoe create-alias tahoe
```

Ora è possibile uplodare e scaricare file

```
# bin/tahoe cp nome_file tahoe:nome_file
```

viceversa, per effettuare un download

```
# bin/tahoe get nome_file nome_nuovo_file
```

Per un elenco di tutti i comandi si veda

```
# bin/tahoe --help
```

Una breve lista:

```
Administration
    create-node                 Create a node that acts as a client, server or
                                both.
    create-client               Create a client node (with storage initially
                                disabled).
    create-introducer           Create an introducer node.
    create-key-generator        Create a key generator service.
    create-stats-gatherer       Create a stats-gatherer service.

Controlling a node
    start                       Start a node (of any type).
    stop                        Stop a node.
    restart                     Restart a node.
    run                         Run a node synchronously.
```

```
Debugging
    debug                       debug subcommands: use 'tahoe debug' for a list.

Using the filesystem
    mkdir                       Create a new directory.
    add-alias                   Add a new alias cap.
    create-alias                Create a new alias cap.
    list-aliases                List all alias caps.
    ls                          List a directory.
    get                         Retrieve a file from the grid.
    put                         Upload a file into the grid.
    cp                          Copy one or more files or directories.
    rm                          Unlink a file or directory on the grid.
    unlink                      Unlink a file or directory on the grid (same as
                                rm).
    mv                          Move a file within the grid.
    ln                          Make an additional link to an existing file or
                                directory.
    backup                      Make target dir look like local dir.
    webopen                     Open a web browser to a grid file or directory.
    manifest                    List all files/directories in a subtree.
    stats                       Print statistics about all files/directories in a
                                subtree.
    check                       Check a single file or directory.
    deep-check                  Check all files/directories reachable from a
                                starting point.
```

5.6 Configurazione della propria grid

Per configurare un nodo è necessario modificare alcuni file nella sua base directory che vengono letti quando si avvia il nodo. Per questo motivo, dopo aver modificato questi file è necessario riavviare il nodo. Un nodo può essere client/server, introducer, statistic gatherer o key generator. Un nodo client/server fornisce i seguenti servizi

- web-API service
- SFTP service
- FTP service
- drop-upload service
- helper service

- storage service

```
[node]
```

```
nickname = (UTF-8 string, optional)
```

È il nome del nodo che verrà visualizzato. Se non viene fornito questo valore sarà settato come unspecified

```
web.port = (string, optional)
```

Setta la porta sulla quale il web server del nodo deve stare in ascolto. Al momento della creazione di un nodo, tahoe setterà questo valore con la porta 3456

```
web.static = (string, optional)
```

Questo valore è un percorso di una directory, contenente i file per il caricamento della parte statica del sito web

```
tub.port = (integer, optional)
```

Controlla su quale porta il nodo accetta connessioni foolscap. Se non viene fornito il nodo chiederà al kernel una delle porte disponibili. La porta sarà salvata in un file separato chiamato chiamato *client.port* o *introducer.port* per poi essere riutilizzata in futuro.

```
tub.location = (string, optional)
```

Si si sta usando un server node, è necessario settare questo valore per ricevere le connessioni degli altri nodi. Ogni nodo genera un *FURL*: Una stringa contenente dei parametri di connessione da passare all'introducer. Questa stringa è situata in propria-base-directory/private/storage.furl. Se non viene fornita, il nodo cercherà di usare strumenti come *ipconfing* per determinare un set di indirizzi ip sul quale può essere raggiunto dagli altri nodi. Questo campo include inoltre la porta sul quale tcp resta in ascolto, la stessa specificata in tub.port o assegnata dal kernel qualora non sia stata specificata. .... Questa volta si ha bisogno di un nodo che faccia da client e server allo stesso tempo, è sufficiente digitare

```
# bin/tahoe create-node
```

e un introducer, che si dovrà creare in una nuova base directory

```
mkdir new_dir
cd new_dir
# bin/tahoe create-introducer ~/new_dir
```

a questo punto si fa partire l'introducer

```
# bin/tahoe start ~/new_dir
```

All'interno di questa directory verrà creato un file chiamato *introducer.furl*, contenente l'url a cui tutti gli altri nodi si dovranno connettere. Si dovranno quindi editare le seguenti voci in *tahoe.cfg* di ogni nodo presente nella grid.

```
[client]introducer.furl= url dell'introducer
[node]nickname= un nome qualsiasi
```

Con l'attuale configurazione non è ancora possibile effettuare alcun upload o download, ci sono ancora tre parametri da impostare:

```
# What encoding parameters should this client use for uploads?
shares.needed =
shares.happy =
shares.total =
```

Ogni volta che un file è caricato nella grid, questi tre valori permettono di impostare i parametri di criptaggio:

- *shares.total* Indica il numero di condivisioni che saranno create all'upload del file
- *shares.needed* Indica il numero di condivisioni necessarie per recuperare il file
- *shares.happy* Indica il numero di server che devono essere disponbili affinchè un file possa essere caricato

Settare *shares.needed = 1* significa fare semplicente N condivisioni, quindi un semplice mirroring del file. Per un corretto upload, *shares.happy* non deve essere maggiore rispetto al numero di nodi della grid. Settare *shares.happy* ¡= *shares.needed* è permesso, ma così si perderà la ridondanza dei file qualora qualche server dovessere cadere.

5.7 Configurazione della propria grid

Per configurare un nodo è necessario modificare alcuni file nella sua base directory che vengono letti quando si avvia il nodo. Per questo motivo, dopo aver modificato questi file è necessario riavviare il nodo. Un nodo può essere client/server, introducer, statistic gatherer o key generator. Un nodo client/server fornisce i seguenti servizi

- web-API service
- SFTP service

- FTP service
- drop-upload service
- helper service
- storage service

```
[node]
```

```
nickname = (UTF-8 string, optional)
```

È il nome del nodo che verrà visualizzato. Se non viene fornito questo valore sarà settato come unspecified

```
web.port = (string, optional)
```

Setta la porta sulla quale il web server del nodo deve stare in ascolto. Al momento della creazione di un nodo, tahoe setterà questo valore con la porta 3456

```
web.static = (string, optional)
```

Questo valore è un percorso di una directory, contenente i file per il caricamento della parte statica del sito web

```
tub.port = (integer, optional)
```

Controlla su quale porta il nodo accetta connessioni foolscap. Se non viene fornito il nodo chiederà al kernel una delle porte disponibili. La porta sarà salvata in un file separato chiamato chiamato *client.port* o *introducer.port* per poi essere riutilizzata in futuro.

```
tub.location = (string, optional)
```

Si si sta usando un server node, è necessario settare questo valore per ricevere le connessioni degli altri nodi. Ogni nodo genera un *FURL*: Una stringa contenente dei parametri di connessione da passare all'introducer. Questa stringa è situata in propria-base-directory/private/storage.furl. Se non viene fornita, il nodo cercherà di usare strumenti come *ipconfing* per determinare un set di indirizzi ip sul quale può essere raggiunto dagli altri nodi. Questo campo include inoltre la porta sul quale tcp resta in ascolto, la stessa specificata in tub.port o assegnata dal kernel qualora non sia stata specificata. .... Questa volta si ha bisogno di un nodo che faccia da client e server allo stesso tempo, è sufficiente digitare

```
# bin/tahoe create-node
```

e un introducer, che si dovrà creare in una nuova base directory

```
mkdir new_dir
cd new_dir
# bin/tahoe create-introducer ~/new_dir
```

a questo punto si fa partire l'introducer

```
# bin/tahoe start ~/new_dir
```

All'interno di questa directory verrà creato un file chiamato *introducer.furl*, contenente l'url a cui tutti gli altri nodi si dovranno connettere. Si dovranno quindi editare le seguenti voci in *tahoe.cfg* di ogni nodo presente nella grid.

```
[client]introducer.furl= url dell'introducer
[node]nickname= un nome qualsiasi
```

Con l'attuale configurazione non è ancora possibile effettuare alcun upload o download, ci sono ancora tre parametri da impostare:

```
# What encoding parameters should this client use for uploads?
shares.needed =
shares.happy =
shares.total =
```

Ogni volta che un file è caricato nella grid, questi tre valori permettono di impostare i parametri di criptaggio:

- *shares.total* Indica il numero di condivisioni che saranno create all'upload del file
- *shares.needed* Indica il numero di condivisioni necessarie per recuperare il file
- *shares.happy* Indica il numero di server che devono essere disponbili affinchè un file possa essere caricato

Settare *shares.needed = 1* significa fare semplicente N condivisioni, quindi un semplice mirroring del file. Per un corretto upload, *shares.happy* non deve essere maggiore rispetto al numero di nodi della grid. Settare *shares.happy* ¡= *shares.needed* è permesso, ma così si perderà la ridondanza dei file qualora qualche server dovessere cadere.

5.8 Vulnerabilità

A questo punto ci si può rendere facilmente conto che gli introducer sono nodi chiave rispetto ai soli nodi client. Più introducer si avranno e più il sistema sarà meno vulnerabile. Ai fini del progetto si è usato un solo

introducer per non complicare troppo il codice. In un sistema abbastanza magliato è lecito pensare che ad ogni introducer venga assegnato un indirizzo ip staticamente, proprio per il ruolo chiave. Più introducer si avranno e più saranno gli indirizzi che andranno a costruire il furl nel file di configurazione di ciascun dispositivo. Tuttavia nulla vieta di sviluppare un sistema di assegnazione degli indizzi dinamico, a fronte però di un notevole aumento di complessità del codice e un notevole calo di performance direttamente proporzionale al numero dei dispositivi connessi alla rete. Dovendo testare sia il corretto funzionamento del file system distribuito che il sistema di assegnazione dinamico degli indizzi ai client e tenendo conto del limitato numero di dispositivi in dotazione si è optato per questa configurazione.

Capitolo 6

Registrazione audio e video

Arrivati a questo punto del progetto non rimane altro che installare i programmi necessari per le registrazioni. Motion è ottenibile digitando dalla shell

```
# apt-get install motion
```

La particolarià di motion risiede soprattutto nei suoi molteplici parametri di configurazione. La banda disponibile è un aspetto di notevole rilevanza dal momento che si potrebbero avere una grande mole di utente connessi, intenti a caricare e scaricare grandi quantità di dati. Per questi motivi occorre configurare motion occuratamente al fine di minimizzare la grandezza delle registrazioni. Il file di configurazione sarà collocato nella propria home. Il primo accorgimento da tenere in considerazione risiede nel fatto che non è necessario registrare di continuo. I tempi morti occupano spazio, banda e non interessano a nessuno. sarebbe opportuno quindi catturare solamente i momenti in cui succede qualcosa. Ciò è possibile editanto la seguente riga di motion.conf

```
# Always save images even if there was no motion (default: off)
output_all off
```

Esistono molti altri parametri che agiscono direttamente o indirettamente sulla qualità e quindi sullo spazio occupato. Ai fini di questo progetto ci si limiterà agli snapshot e alla estensione come file jpeg. Ultima ma non meno importante è la directory di destinazione dei file in output. Tale directory verrà impostata nella home dell'utente

```
target_dir ~/motion_movies
```

Sebbene sia consigliato usare un path assoluto, grazie a questo metodo si evita di dover creare un ulteriore parametro di configurazione per l'utente, in più si ha la certezza di avere i permessi di scrittura in questo spazio di lavoro.

Capitolo 7

Automatizzazione dei processi

7.1 Creazione della rete

Si hanno due tipi di configurazioni:

- script_securemesh_server
- script_securemesh_client

script_securemesh_server dovrà essere lanciato sulla macchina che ospiterà il nodo introducer di tahoe e il server dhcp, mentre script_securemesh_client su tutti i dispositivi che si connetteranno alla rete.

Salvo queste differenze gli script sono molto semplici: Entrambi gli script devono essere eseguiti con i privilegi di root.

7.2 Script macchina server

Cominciamo ad analizzare lo script più importante

Per prima cosa abbiamo 2 variabili da impostare

```
int\_wifi=wlan0
bat0\_ip=192.168.0.10
```

int_wifi rappresenta l'interfaccia wireless presente sulla macchina. Verrà in seguito disattivata per evitare problemi di configurazione. Solitamente è rappresentata da wlan0 o eth1. A bat0_ip verrà invece impostato un indirizzo ip in modo statico. Tale indirizzo sarà quello del server dhcp.

Si disabilita il network manager per evitare che interferisca con la rete mesh, e si procede alla creazione della stessa.

```
service network-manager stop
```

Si carica il modulo di batman-adv

```
modprobe batman-adv
```

e si disabilità l'interfaccia wireless

```
ifconfig ${int\_wifi} down
```

A questo punto si può procedere alla creazione della rete. Verrà creata una rete ad-hoc non criptata di nome securemesh

```
ifconfig ${int\_wifi} mtu 1528
iwconfig ${int\_wifi} mode ad-hoc
iwconfig ${int\_wifi} channel 1
iwconfig ${int\_wifi} enc off
iwconfig ${int\_wifi} essid securemesh
```

Si configura la propria interfaccia wireless per essere usata con batman-adv, che in ogni caso ne creerà una propria virtuale chiamata bat0.

```
batctl if add ${int\_wifi}
```

Si fa partire il server dhcp e si riabilità l'interfaccia wireless

```
service isc-dhcp-server start
ifconfig ${int\_wifi} up
```

Non rimane altro che assegnare un ip a bat0. Gli indizzi dei vari client verranno rilascia su questa interfaccia, come specificato nell'apposito file di configurazione

```
ifconfig bat0 up ${bat0\_ip}
ifconfig ${int\_wifi} up

Infine si starta tahoe e l'introducer

\begin{verbatim}
~/allmydata-tahoe-1.10.0/bin/tahoe start ~/introducer/

~/allmydata-tahoe-1.10.0/bin/tahoe start
```

e si crea la nuova directory /tmp/mount_dir che verrà usata come punto di mount dello spazio condiviso fra tutti i dispositivi all'interno del file system

```
mkdir /tmp/mount\_dir
```

Vengono fatte partire sia la registrazione audio

```
~/securemesh/script/parallel\_commands
```

A questo punto il controllo viene passato al seguente script, comune a tutti i dispositivi

```
~/securemesh/script/update\_config/nmap\_script
```

Una piccola parentesi per quanto riguarda lo script

```
~/securemesh/script/parallel\_commands
```

il è contenuto il seguente codice, che permette di lanciare due comandi sullo stesso terminale:

```
for cmd in "$@"; do {
  echo "Process \"$cmd\" started";
  $cmd & pid=$!
  PID\_LIST+=" $pid";
} done

trap "kill $PID\_LIST" SIGINT

echo "Parallel processes have started";

wait $PID\_LIST

echo
echo "All processes have completed";
```

I comandi che si andranno a lanciare nel prossimo script saranno motion per quanto riguarda la registrazione video e arecord per l'audio

```
parallel\_commands "arecord $(date '+%Y-%m-%d\_%H-%M-%s.wav')" "motion"
```

7.3 Script macchina client

ToDo

7.4 Script automatico

Rappresenta la logica di controllo dell'intero progetto. Lo script controlla in un'intervallo di 30 seconda se ci sono variazioni sul numero di nodi connessi alla rete. Se non è avvenuto nessun cambiamento motion e arecord continuano a registrare e il timer viene resettato, in caso contrario le registrazioni devono essere fermate insieme a tahoe. Devono inoltre essere aggiornati il file di configurazione tahoe.cfg con il corretto numero di nodi. Una volta fatto ciò, tahoe e le registrazioni vengono riavviate.

7.5 Script automatico

Rappresenta la logica di controllo dell'intero progetto. Lo script controlla in un'intervallo di 30 seconda se ci sono variazioni sul numero di nodi connessi alla rete. Se non è avvenuto nessun cambiamento motion e arecord continuano a registrare e il timer viene resettato, in caso contrario le registrazioni devono essere fermate insieme a tahoe. Devono inoltre essere aggiornati il file di configurazione tahoe.cfg con il corretto numero di nodi. Una volta fatto ciò, tahoe e le registrazioni vengono riavviate.

7.6 Configurazione fuse

Si può accede a tahoe tramite fuse in 2 modi: tramite login o tramite un file di configurazione. Il secondo modo risulta più appropriato poichè rende la procedura automatica, una volta configurato il tutto a dovere.

Assumendo che tahoe si trovi nella propria home, in una directory chiamata BASEDIR si dovrà creare il seguente file

```
BASEDIR/private/accounts
```

il cui contenuto sarà rispettivamente nome utente, password e l'URI della directory condivisa. Output d'esempio per un utente generico:

```
generic\_user generic\_password URI:DIR2:as7rukf3ysxeeloa5t73trt6e4:b6pn54dpf2vzc5er
```

7.6.1 Configurazione accesso sftp

Proprio come ssh, si avrà bisogno di una coppia di chiavi. La prima volta che un client comunchierà con un dato server, memorizzerà la sua host-key, che dovrà essere la stessa anche per le connessioni future.

Si avrà bisogno di openSSH, liberamente scaricabile dal proprio gestore dei pacchetti.

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

Si procede alla creazione della chiave

```
cd BASEDIR
ssh-keygen -f private/ssh\_host\_rsa\_key
```

A questo punto si deve abilitare il server sftp editanto il tahoe.cfg, aggiungendo le seguenti voci:

```
[sftpd]
enabled = true
port = tcp:8022:interface=127.0.0.1
host\_pubkey\_file = private/ssh\_host\_rsa\_key.pub
host\_privkey\_file = private/ssh\_host\_rsa\_key
accounts.file = private/accounts
```

7.7 Configurazione fuse

Si procede quindi all'analisi del codice:

```
touch /tmp/temp.txt
sed '/shares.total =.*/d' ~/securemesh/script/update\_config/concat1.txt &> /tmp/ter
cp /tmp/temp.txt ~/securemesh/script/update\_config/concat1.txt
```

Queste 3 stringhe non fanno altro che ripristinare il file di configurazione di tahoe nel caso in cui lo script dovesse essere stato interotto in un punto critico, al fine di evitare problemi legati a dati sporchi. Per tale motivo risiede fuori dal loop centrale.

tramite nmap scansiono la rete nel range di indizzi disponibili tramite dhcp, e per ognuno di essi controllo che facciano uso della porta 44655 usata da tahoe. In questo modo è possibile capire quanti nodi sono connessi alla rete.

```
nmap -p 44655 192.168.0.10-100 > /tmp/test
OUTPUT=$(grep -c '44655/tcp' /tmp/test)

echo "Vecchio share.total = $OLD\_SHARES";
echo "Nuovo share.total = $OUTPUT";
```

Ottenuti questi valori, si accede al file system distribuito tramite fuse

```
echo user | sshfs user@127.0.0.1: /tmp/mount\_dir -o password\_stdin -p 8022
```

dove

- /tmp/mount_dir è il punto di mount della uri condivisa

- -o password_stdin è un riferimento al nome utente e password contenuti nel file /.tahoe/private/accounts
- -p 8022 è il numero della porta usata

Da qui in poi si entra nel loop infinito. La condidizione di uscita settata a true è voluta poichè il progetto è pensato per lanciare le registrazioni solamente in fase di start-up e di dimenticarsene in seguito.

```
OLD\_SHARES=$(cat ~/.tahoe/tahoe.cfg | grep -o 'shares.total = [[:digit:]]\{1,3\}')
OLD\_SHARES=${OLD\_SHARES##*= }
```

Si va inanzitutto a salvare l'attuale valore di shares.total che verrà in seguito confrontato con i nuovi valori. Da qui è possibile capire se sono avvenuti dei cambiamente circa il numero di nodi all'iterno della rete con la seguente condizione di if

```
if [ "$OLD\_SHARES" != "$OUTPUT" ]; then
```

A questo punto se nulla è cambiato aspetto lo scadere di un timer, per poi ripere questo controllo. Se invece la situazione è cambiata:
stoppo sia motion che arecord

```
#kill $(pgrep motion)
#kill $(pgrep arecord)
```

smonto la directory mount al file system

```
umount /tmp/mount\_dir
```

e mi ricostruisco quello che sarà il nuovo file di configurazione tahoe.cfg aggiornato con i valori attuali.

```
echo "shares.total = $OUTPUT" >> ~/securemesh/script/update\_config/concat1.txt
cat ~/securemesh/script/update\_config/concat1.txt ~/securemesh/script/update\_confi
#ripristino concat1 per il prossimo ciclo
sed '$d' ~/securemesh/script/update\_config/concat1.txt > /tmp/temp.txt
cp /tmp/temp.txt ~/securemesh/script/update\_config/concat1.txt
```

Non resta altro che restartare tahoe

```
~/allmydata-tahoe-1.10.0/bin/tahoe restart
```

Rimontare mount_dir

```
echo user | sshfs user@127.0.0.1: /tmp/mount\_dir -o password\_stdin -p 8022
```

e rilanciare motion e arecord

```
xterm -e ./parallel\_commands "arecord $(date '+%Y-%m-%d\_%H-%M-%s.wav')" "motion" &
```

Capitolo 8

Misurazioni

8.1 Introduzione

ToDo

8.2 Livello data-link

La seguente prova è stata effettuata con 2 dispositivi, sia a breve che a media distanza con dei muri nel mezzo per mezzo di una funzionalità d batctl che permette di pingare a livello data-link

```
batctl ping 00:1e:52:81:44:84
```

dove 00:1e:52:81:44:84 è il MAC address del dispositivo in questione. L'output ottenuto:

```
20 bytes from 00:1e:52:81:44:84 icmp_seq=1 ttl=50 time=0.56 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=2 ttl=50 time=0.68 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=3 ttl=50 time=2.29 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=4 ttl=50 time=0.57 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=5 ttl=50 time=0.61 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=6 ttl=50 time=0.61 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=7 ttl=50 time=4.13 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=8 ttl=50 time=1.02 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=9 ttl=50 time=0.63 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=10 ttl=50 time=0.59 ms
...
```

La latenza media risulta essere di 1.169 ms. Ad una prima analisi sembra un valore piuttosto alto, soprattutto perchè siamo in presenza di una connessione diretta fra 2 dispositivi posti a pochi centrimetri di distanza. C'è anche da dire però che i dispositivi in questione non sono macchine dedicate

e il ricevitore wireless di un pc non può essere comparato ad una antenna di un router in termini di prestazioni.

La situazione non differesce con i due dispositivi posti ad una distanza maggiore e con dei muri di mezzo:

```
20 bytes from 00:1e:52:81:44:84 icmp_seq=1 ttl=50 time=0.58 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=2 ttl=50 time=0.93 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=3 ttl=50 time=0.62 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=4 ttl=50 time=0.57 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=5 ttl=50 time=4.81 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=6 ttl=50 time=0.63 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=7 ttl=50 time=0.52 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=8 ttl=50 time=3.44 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=9 ttl=50 time=0.96 ms
20 bytes from 00:1e:52:81:44:84 icmp_seq=10 ttl=50 time=0.61 ms
```

Questa volta la media è di 1.367 ms, un risultato in linea con il precende.

Testiamo ora il meccanismo di discovery con due soli dispositivi, verrà usata anche qui una funzionalità di batctl

```
batctl o
```

il comando è stato lanciato 10 volte per ottere diversi campioni di output:

```
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.620s   (251) 00:15:af:88:52:dd [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.428s   (251) 00:15:af:88:52:dd [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.480s   (251) 00:15:af:88:52:dd [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.828s   (251) 00:15:af:88:52:dd [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.972s   (251) 00:15:af:88:52:dd [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.924s   (251) 00:15:af:88:52:dd [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.884s   (251) 00:15:af:88:52:dd [     wlan0]: 00:15:af:88:52:d
```

```
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.796s   (251) 00:15:af:88:52:dd [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.696s   (251) 00:15:af:88:52:dd [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.568s   (251) 00:15:af:88:52:dd [     wlan0]: 00:15:af:88:52:d
```

Si ha un discovery medio di 0.72 ms.
Verrà ripetuto il test con un dispositivo aggiuntivo:

```
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.696s   (248) 00:15:af:88:52:dd [     wlan0]: 00:1e:52:81:44:8
00:1e:52:81:44:84    0.236s   (227) 00:1e:52:81:44:84 [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.112s   (247) 00:15:af:88:52:dd [     wlan0]: 00:1e:52:81:44:8
00:1e:52:81:44:84    0.740s   (226) 00:1e:52:81:44:84 [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.604s   (249) 00:15:af:88:52:dd [     wlan0]: 00:1e:52:81:44:8
00:1e:52:81:44:84    0.308s   (223) 00:1e:52:81:44:84 [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.196s   (251) 00:15:af:88:52:dd [     wlan0]: 00:1e:52:81:44:8
00:1e:52:81:44:84    0.068s   (219) 00:1e:52:81:44:84 [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.024s   (251) 00:15:af:88:52:dd [     wlan0]: 00:1e:52:81:44:8
00:1e:52:81:44:84    0.676s   (216) 00:1e:52:81:44:84 [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.744s   (251) 00:15:af:88:52:dd [     wlan0]: 00:1e:52:81:44:8
00:1e:52:81:44:84    0.540s   (214) 00:1e:52:81:44:84 [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.396s   (252) 00:15:af:88:52:dd [     wlan0]: 00:1e:52:81:44:8
00:1e:52:81:44:84    0.192s   (213) 00:1e:52:81:44:84 [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.232s   (253) 00:15:af:88:52:dd [     wlan0]: 00:1e:52:81:44:8
```

```
00:1e:52:81:44:84    0.080s   (213) 00:1e:52:81:44:84 [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.088s   (251) 00:15:af:88:52:dd [     wlan0]: 00:1e:52:81:44:8
00:1e:52:81:44:84    0.836s   (212) 00:1e:52:81:44:84 [     wlan0]: 00:15:af:88:52:d
[B.A.T.M.A.N. adv 2012.2.0, MainIF/MAC: wlan0/4c:eb:42:9d:09:e0 (bat0)]
  Originator      last-seen (#/255)           Nexthop [outgoingIF]:   Potential next
00:15:af:88:52:dd    0.632s   (250) 00:15:af:88:52:dd [     wlan0]: 00:1e:52:81:44:8
00:1e:52:81:44:84    0.468s   (211) 00:1e:52:81:44:84 [     wlan0]: 00:15:af:88:52:d
```

Questa volta il tempo medio di discovery è di soli 0.37ms. Sebbene 3 dispositivi non siano sufficienti a determinare un caso statistico, si può comunque affermare che il numero di dispositivi non influenza (almeno in maniera percettibile) le prestazioni a questo livello di rete.

8.3 Livello rete

A livello rete le prestazioni cambiano notevolmente, purtroppo in peggio. Il test è stato effettuato con soli 2 dispositivi connessi a una distanza davvero ravvicinata. Per prima cosa è necessario ottenere un indirizzo ip dal server dhcp. La procedura dura in media qualche secondo come si può vedere dal log.

```
tail -f /var/log/syslog
```

l'output:

```
Mar 27 21:12:12 marco-U36SG dhcpd: DHCPDISCOVER from 02:b7:82:f0:dc:ee via bat0
Mar 27 21:12:13 marco-U36SG dhcpd: DHCPOFFER on 192.168.0.34 to 02:b7:82:f0:dc:ee (g
Mar 27 21:12:14 marco-U36SG dhcpd: DHCPDISCOVER from 02:b7:82:f0:dc:ee (guerro-MacBo
Mar 27 21:12:14 marco-U36SG dhcpd: DHCPOFFER on 192.168.0.34 to 02:b7:82:f0:dc:ee (g
Mar 27 21:12:19 marco-U36SG dhcpd: DHCPDISCOVER from 02:b7:82:f0:dc:ee (guerro-MacBo
Mar 27 21:12:19 marco-U36SG dhcpd: DHCPOFFER on 192.168.0.34 to 02:b7:82:f0:dc:ee (g
Mar 27 21:12:19 marco-U36SG dhcpd: DHCPREQUEST for 192.168.0.34 (192.168.0.1) from (
Mar 27 21:12:19 marco-U36SG dhcpd: DHCPACK on 192.168.0.34 to 02:b7:82:f0:dc:ee (gue
```

Fattò ciò è possibile avvalersi della funzionalità ping per testare la latenza dei pacchetti. che corrisponde all'indirizzo assegnato da dhcp

```
ping 192.168.0.34
```

l'output:

```
PING 192.168.0.34 (192.168.0.34) 56(84) bytes of data.
64 bytes from 192.168.0.34: icmp_req=1 ttl=64 time=2.07 ms
64 bytes from 192.168.0.34: icmp_req=2 ttl=64 time=2.09 ms
64 bytes from 192.168.0.34: icmp_req=3 ttl=64 time=2.09 ms
64 bytes from 192.168.0.34: icmp_req=4 ttl=64 time=2.04 ms
64 bytes from 192.168.0.34: icmp_req=5 ttl=64 time=2.09 ms
...
...
...
...
64 bytes from 192.168.0.34: icmp_req=30 ttl=64 time=3.86 ms
64 bytes from 192.168.0.34: icmp_req=31 ttl=64 time=5.24 ms
64 bytes from 192.168.0.34: icmp_req=32 ttl=64 time=2.08 ms
64 bytes from 192.168.0.34: icmp_req=33 ttl=64 time=3.61 ms
64 bytes from 192.168.0.34: icmp_req=34 ttl=64 time=2.09 ms
64 bytes from 192.168.0.34: icmp_req=35 ttl=64 time=4.85 ms

--- 192.168.0.34 ping statistics ---
35 packets transmitted, 35 received, 0% packet loss, time 34039ms
rtt min/avg/max/mdev = 2.018/2.653/6.566/1.093 ms
```

come si può notare dalle statistiche finali

```
--- 192.168.0.34 ping statistics ---
35 packets transmitted, 35 received, 0% packet loss, time 34039ms
rtt min/avg/max/mdev = 2.018/2.653/6.566/1.093 ms
```

Si ha un round trip time di circa 2.5 secondi, decisamente troppi per una connessione diretta fra soli due dispositivi. é da notare la differenza di circa 1 secondo di latenza tra il livello rete e data-link.

Purtroppo il vero punto debole si è rilevato essere proprio il file system condiviso con delle performance davvero basse. Al posto delle registrazioni si sono caricariti direttamente dei file all'interno del file system a scopo dimostrativo, per avere delle statistiche di rirefimento. Ci si è posizionati all'interno del punti di mount:

```
cd /tmp/mount_dir
```

tramite il comando dd sono stati generati automaticamente in blocco dei file

```
root@marco-U36SG:/tmp/mount_dir# dd if=/dev/random of=/tmp/mount_dir/immagine4 count
0+1 record dentro
0+1 record fuori
20 byte (20 B) copiati, 0,00133066 s, 15,0 kB/s
```

```
root@marco-U36SG:/tmp/mount_dir# dd if=/dev/random of=/tmp/mount_dir/immagine5 count
0+2 record dentro
0+1 record fuori
25 byte (25 B) copiati, 12,7966 s, 0,0 kB/s

root@marco-U36SG:/tmp/mount_dir# dd if=/dev/random of=/tmp/mount_dir/immagine6 count
0+3 record dentro
0+1 record fuori

32 byte (32 B) copiati, 29,4644 s, 0,0 kB/s
root@marco-U36SG:/tmp/mount_dir# dd if=/dev/random of=/tmp/mount_dir/immagine7 count
0+4 record dentro
0+1 record fuori
41 byte (41 B) copiati, 51,4252 s, 0,0 kB/s
root@marco-U36SG:/tmp/mount_dir# dd if=/dev/random of=/tmp/mount_dir/immagine8 count
0+5 record dentro
0+1 record fuori

54 byte (54 B) copiati, 54,2965 s, 0,0 kB/s
root@marco-U36SG:/tmp/mount_dir# dd if=/dev/random of=/tmp/mount_dir/immagine9 count
0+6 record dentro
0+1 record fuori

63 byte (63 B) copiati, 70,9131 s, 0,0 kB/s
root@marco-U36SG:/tmp/mount_dir# dd if=/dev/random of=/tmp/mount_dir/immagine9 count
0+7 record dentro
0+1 record fuori
57 byte (57 B) copiati, 102,375 s, 0,0 kB/s
root@marco-U36SG:/tmp/mount_dir# dd if=/dev/random of=/tmp/mount_dir/immagine10 coun
0+8 record dentro
0+1 record fuori

76 byte (76 B) copiati, 59,5999 s, 0,0 kB/s
root@marco-U36SG:/tmp/mount_dir# dd if=/dev/random of=/tmp/mount_dir/immagine11 coun
0+9 record dentro
0+1 record fuori
80 byte (80 B) copiati, 141,502 s, 0,0 kB/s

root@marco-U36SG:/tmp/mount_dir# dd if=/dev/random of=/tmp/mount_dir/immagine12 coun
0+10 record dentro
0+1 record fuori
90 byte (90 B) copiati, 162,151 s, 0,0 kB/s
```

Per copiare un file da 90 byte su due soli dispositivi si è dovuto aspettare circa 160 secondi. Allo stato attuale del software risulta quindi impensabile usare l'intero sistema per lo scopo prefissato.

Capitolo 9

Conclusioni

Al livello attuale di sviluppo del software, il sistema non funziona come dovrebbe. Il problema principale è dovuto alla limitata banda a disposizione. Dai test prima riportati risulta impensabile per l'intero progetto trovare un reale campo di utilizzo. Purtroppo tutti i file system distributi disponibili non sono stati progettati per avere a monte reti di questo tipo. Oltre a questo c'è da fare un'altra considerazione: tahoe, come del resto gli altri file system, devono sapere preventivamente la topologia della rete e, qualora questa dovesse cambiare, è necessario andare ad aggiornare tutti i file di configurazione su ciascun nodo e riavviare tahoe. Riavviare tahoe significa stoppare le registrazioni e smontare il punto di mount. Per quanto questo processo possa essere automatizzato, in una rete abbastanza magliata questa operazione risulta essere molto onerosa in termini prestazionali. Una possibile alternativa potrebbe essere l'uso di un sistema di mirroring come torrent, in cui si invia semplicemente i dati ad ogni utente della rete senza il bisogno di saperne il numero esatto e quindi evitando il riavvio. Presi singolarmente, rete e file system risultano essere molto stabili discorso che però non può essere fatto per la loro integrazione. Le funzionalità a livello 2 di batman-adv risultano essere molto interessanti e potenti, in questo progetto però perdono di significato dal momento che tutto ciò che viene implentato al di sopra ha bisogno del livello rete.

Commotion sembra aver intrapreso la giusta direzione, sebbene prevedendo l'uso di router e antenne dedicate, dispositivi la cui mancanza in questo progetto si è sentita ma che purtroppo non si sposano bene con il suo scopo. Durante il lavoro, commotion ha compiuto passi da gigante passando da una build alfa con notevoli problemi a tutta una serie di software disponibili per più dispositivi e piattaforme. Ciò che manca ancora, è la possibilità di condividere dati. Sarà interessante osservarne i nuovi sviluppi.

A prescindere dal risultato ottenuto, questo progetto è servito ad entrare in contatto con questa tipologia di reti sconosciute ancora a molti. Sono stati eseguiti numerosi lavori di configurazione al fine di ottenere la migliore

integrazione possibile fra i vari strati del progetto, questo ha permesso di ottenere nuove conoscenze di rete e scripting per quanto riguarda i sistemi linux.

Bibliografia

[1] Tim Kindberg e George Coulouris Jean Dollimore. Distributed systems: Concepts and design, 2005.