AVALUACIÓ DE NOUS ALGORITMES PER A LA CREACIÓ DE CAMINS ENTRE NODES DE LA XARXA TOR

Memòria del projecte final de carrera corresponent als estudis d'Enginyeria Superior en Informàtica presentat per Esteve Alquézar i Mora i dirigit per Sergio Castillo Pérez.

Bellaterra, Febrer de 2010

El firmant, Sergio Castillo Pérez, professor del Departament d'Enginyeria de la Informació i de les Comunicacions de la Universitat Autònoma de Barcelona

CERTIFICA:

Que la present memòria ha sigut realitzada sota la seva direcció per Esteve Alquézar i Mora

Bellaterra, Febrer de 2010

Firmat: Sergio Castillo Pérez

A la meva família.

Agraïments

- Voldria començar donant les gràcies al meu director de projecte Sergio Castillo per haver confiat en mi des d'un principi. Ha sigut un camí llarg però gratificant, tant a nivell acadèmic com a nivell personal.

- Voldria també donar les gràcies als meus amics de tota la vida i als amics i companys de la universitat per tots els moments que hem passat junts i que m'han fet créixer com a persona.

- Finalment, vull agrair a tota la família i en especial als meus pares per tot el suport i tota la comprensió que m'han donat durant tota l'Enginyeria i durant la realització d'aquest projecte, mai sense oblidar l'esforç i constància que han dedicat per poder-me donar una bona educació. GRÀCIES!

Índex

Índex de figures

Capítol 1

Introducció

Actualment, en l'època en que vivim de les noves tecnologies i els avenços en el camp de la informació fan que el volum de dades hagi crescut d'una forma increïblement ràpida. La informació és poder i hi ha molts mecanismes per trasgiversarla. Buscar una informació des d'un punt del planeta o des d'un altre pot donar lloc a resultats manipulats en quan a número de resultats i en quan a la integritat de la mateixa. Hi ha més filtres del que aparentment sembla: governs, organitzacions i altres tipus d'institucions modifiquen la informació o simplement fan que no sigui accessible depenent de circumstàncies temporals i geogràfiques, entre d'altres.

Per aquest motiu, s'han anat desenvolupant eines per tal de preservar l'anonimat en les xarxes de comunicació amb la finalitat de poder-se expressar amb total llibertat, i poder accedir a la informació amb independència d'on estigui localitzat el subjecte.

Aquest projecte està basat en la xarxa d'anonimat Tor, que es basa en una implementació del que s'anomena la segona generació de *onion routing*.

Per situar-nos en el context ens remuntarem a l'origen de les xarxes anònimes. El primer model, i el que ha influenciat més en les comunicacions anònimes, va ser presentat per David Chaum, inventor de diversos protocols criptogràfics i s'anomenava The Mix Network, a l'any 1981. La idea bàsica era establir comunicacions utilitzant una cadena de servidors proxies.

El funcionament a grans trets és el següent: cada missatge es xifra en cada

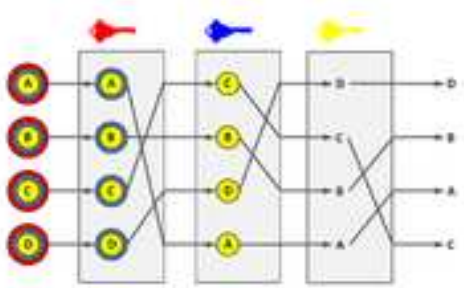

Figura 1.1: Procés de xifratge d'una xarxa Mix Network

proxy utilitzant una clau pública. El resultat del xifratge s'encapsula en una capa (en la figura 1.1 es pot veure en forma de rectangle), i el resultat final és que tenim el missatge en la capa xifrada més interna (en el últim rectangle de més a la dreta). Cada servidor proxy extreu una capa de xifratge per saber a qui li haurà d'enviar el missatge. D'aquesta manera s'evita la possibilitat de poder veure per a quins nodes passarà la informació. Només es coneix el següent salt i el previ.

Més endavant, a l'any 1991, Ptzmann i Waidner van dissenyar un sistema similar però per anonimitzar les converses telefòniques que funcionaven sobre un sistema ISDN (Integrated Services Digital Network). Finalment, diverses idees sobre aquest disseny van ser adaptades per a poder navegar de forma anònima per la Web. És el que es va anomenar Web Mixes.

Goldschlag, Reed i Syverson varen presentar a l'any 1996 la Onion routing. És equivalent i semblant a les xarxes comentades anteriorment Mix Networks però en un altre context. Aquestes es basen en una tecnologia o estratègia d'encaminament de circuits. El seu funcionament és el següent: en comptes d'encaminar cada paquet anònim de forma separada, el primer missatge obre un circuit a través de la xarxa, etiquetant una ruta. Cada missatge que tingui una etiqueta particular serà encaminat per el seu camí predeterminat. Un cop s'ha acabat la transmissió, es pot enviar un missatge amb la finalitat de tancar el camí actual. El principal objectiu és aconseguir que l'anàlisi de tràfic d'aquesta xarxa sigui més difícil i per tant, la correlació d'informació no permet identificar l'emissor ni el tràfic intercanviat.

El primer missatge que s'envia a través de la xarxa està xifrat en diverses ca-

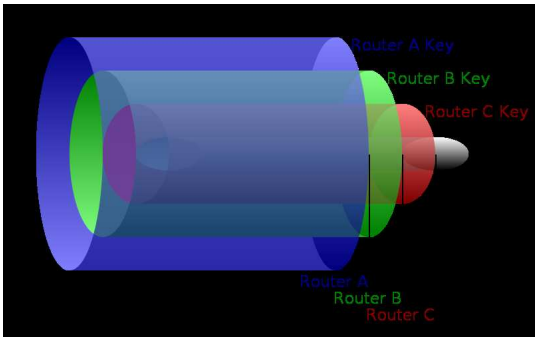

Figura 1.2: Estructura Onion Routing

pes, que només pot ser desxifrat a través d'una cadena de Onion routers utilitzant les seves respectives claus privades. A part, el primer missatge (en la figura 1.2 el podem veure representat de color gris) també conté la clau compartida de l'emissor i els encaminadors i informació sobre l'adreça del següent.

Finalment arribem a la segona generació de Onion router: Tor (The second Onion Routing). Va arribar a l'any 2004 (Dingledine, Mathewson i Syverson) [SGOR04]. Tor retransmet fluxos arbitraris TCP per tota una xarxa de retransmissors. Cal destacar que està especialment ben sintonitzat per treballar amb el tràfic web.

Tor utilitza una arquitectura tradicional de xarxa. Els clients poden crear els seus propis camins escollint un número arbitrari de nodes (per defecte en són tres) a partir d'un directori public contingut en diversos servidors. Actualment el criteri de selecció que s'utilitza per a construcció de circuits es basa principalment en tres factors aconseguint una bona relació entre latència i anonimat. Va ser introduït per Renner i s'explicarà més endavant amb més detall. Considera per una banda el *uptime*: com més temps faci que el node està en funcionament, més probabilitats hi haurà de continuï en funcionament, proporcionant-nos d'aquesta manera estabilitat en la construcció dels circuits. Per altre banda tenim la variable *GeoIP*: si els nodes es seleccionen dintre del mateix país, s'aconseguira reduir la latència, però per contra perdrem nivell d'anonimat. I finalment la variable *bandwidth*: tindran preferència els nodes amb més ample de banda augmentant d'aquesta manera la velocitat de transferència.

A diferència de la primera implementació del Onion routing, Tor utilitza un mecanisme interactiu: primer de tot, el client es connecta amb el primer node i després li demana a aquest node que es connecti en el següent node. El canal bidireccional es fa servir en cada etapa per a intercanviar la clau Diffie-Hellman.

Tor proveeix un mecanisme per a serveis ocults: un servidor amagat obre una connexió anònima i l'utilitza per avisar d'un punt de contacte. Cal remarcar que Tor es va dissenyar amb la idea de tenir una baixa latència tot i que la realitat és una altra. Això és possible mitjançant servidors que corren per tota una sèrie de voluntaris a traves d'Internet.

En resum, podem dir que Tor és una xarxa que proporciona l'anonimat als usuaris vinculats a la xarxa Internet que utilitzen el protocol TCP. La xarxa funciona íntegrament per voluntaris i és actualment la xarxa anònima pública més gran del món, format per aproximadament 1500 nodes amb una capacitat total de 3Gbps i una capacitat de sortida de 1Gbps.

1.1 Motivacions

La principal motivació de la realització d'aquest projecte ha estat que actualment no hi ha cap treball que hagi avaluat els algoritmes de selecció de nodes des d' un punt de vista de la latència i de l'anonimat. En aquest projecte farem propostes de nous algoritmes de selecció de nodes així com les seves avaluacions.

A part, sempre he tingut un gran interès pel món de les xarxes i la seguretat, cosa que m'ha despertat un gran interès i curiositat per aprendre el seu funcionament. Aquest projecte està molt lligat al món de les TIC (Tecnologies de la Informació i Comunicació) en el qual cada dia va prosperant i millorant. M'agradaria dedicar-me professionalment en un futur proper.

1.2 Objectius i abast del problema

L'objectiu d'aquest Projecte Final de Carrera és estudiar i treballar en una sèrie d'experiments per analitzar com afecten diferents algoritmes proposats per a selecció de nodes dels circuits, tan des del punt de vista de l'anonimat com de la latència. Més endavant analitzarem amb profunditat tres criteris de selecció de nodes per tal de crear circuits.

L'abast del problema bàsicament radicarà en aconseguir els millors resultats en quant al balanç entre l'anonimat i latència.

1.3 Organització de la memòria

A continuació expliquem l'organització d'aquesta memòria, donant lloc a una breu descripció dels continguts de cada capítol.

En aquest **primer capítol** farem una petita i breu introducció a aquest projecte final de carrera. S'explicarà l'origen, l'abast, els objectius i les motivacions del projecte.

En el **segon capítol** es comenta l'estudi de viabilitat per realitzar aquest projecte. S'explicaran també els requisits previs per a la realització del projecte, l'estat de l'art del tema, el propi estudi de viabilitat en si i la planificació que es pretén seguir.

En el **tercer capítol** explicarem d'una manera simple els principis de la criptografia per tal de poder entendre millor el funcionament de Tor. Comentarem també l'arquitectura que té associada així com el protocol de control que té per a objectiu controlar la creació de circuits.

En el **quart capítol** explicarem basicament tres coses: per una banda el disseny en la que introduirem el modul TorCtl, escrit en Python que es el nucli principal d'aquest projecte. Per altre banda exposarem l'entorn de desenvolupament i finalment explicarem amb detall la implementació dels diferents algoritmes proposats en aquest projecte.

En el **cinquè capítol** presentarem els resultats empírics resumits i explicats en forma de gràfiques amb intervals de confiança per a cada algoritme proposat. D'aquesta manera podrem observar l'eficiència dels algoritmes des de el punt de vista de la latència en funció de l'algoritme.

En el **sisè capítol** exposarem les conclusions a les quals hem arribat. Explicarem els coneixements adquirits per tal de poder desenvolupar aquest projecte així com els objectius complerts. Al final del tema presentarem les conclusions obtingudes juntament amb les possibles futures millores i ampliacions.

Capítol 2

Estudi de viabilitat

Al llarg d'aquest capítol analitzarem de manera detallada la viabilitat d'aquest projecte des de diversos punts de vista: legal, econòmic i material.

És de vital importància abans de començar un projecte analitzar el que es vol gestionar, millorar i avaluar així com considerar si són viables tots els recursos que aquest impliqui. És per aquest motiu que a continuació en fem un previ anàlisi. És també molt important conèixer alternatives del producte o dels productes que estan en el mercat per tal de conèixer que hi ha desenvolupat, que es podria desenvolupar o que es podria millorar entre altres aspectes. Per això veurem l'estat actual del projecte de software lliure de TOR, que es pot obtenir d'una manera totalment gratuïta i per a diferents plataformes a la seva pàgina oficial: **http://www.torproject.org**

Per a realitzar l'estudi de viabilitat presentat a continuació ens hem basat en el model **IEEE Std 830-1998** com a referència.

2.1 Estat de l'art

Tor es un projecte de software lliure que ajuda a defensar-se contra l'anàlisi de tràfic, una forma de vigilància de la xarxa que amenaça la llibertat personal, la privacitat, la confidencialitat en els negocis i relacions, la seguretat de l'estat. També evita que algú que observi una connexió a Internet aprengui quins llocs

es visiten i a part evita que els llocs que es visitin aprenguin la posició física de l'origen de la connexió. Tot això és possible gràcies a la col·laboració d'usuaris que també busquen anonimat, els quals cedeixen els recursos de forma altruista com ara servidors i ample de banda.

El fet de que la selecció sigui aleatòria implica que hi haurà una separació geogràfica entre nodes. Com a conseqüència es pot donar el fet de tenir altes latències. S'ha de considerar també que a major nombre de nodes que formin el circuit més gran serà l'anonimat, però més gran serà la latència. Donat al caràcter aleatori de la selecció dels nodes, sorgeix un compromís entre la latència i la anonimat.

Hi ha una serie de propostes per a la selecció dels nodes que formaran part del circuit que la seva finalitat és aconseguir una latència més baixa a costa de sacrificar el nivell anonimat. La que s'utilitza actualment, ha estat introduïda per Renner [ARES08] i es basa en tres factors per tal de millorar aquesta relació:

- **Uptime**: aquest factor intenta garantitzar que els nodes per els quals passarà la informació estiguin disponibles i que fallin el menys possible. És a dir, ens garanteix la disponibilitat.

- **GeoIP**: aquest paràmetre s'utilitza per aconseguir una latència més baixa: es triga menys temps a establir canals de comunicació dins una mateixa zona geogràfica. Tindran doncs, més prioritat els nodes de la mateixa zona.

- **Bandwidth**: aquest últim paràmetre ens determinarà que el flux d'informació sigui el més ràpid possible. Tindran preferència els nodes que tinguin més ampla de banda disponible.

A part de la xarxa Tor, podem trobar diferents alternatives pel que fa a la xarxes anònimes. En la introducció s'han esmentat unes quantes, la majoria d'elles estan obsoletes actualment però han sigut una font d'inspiració per desenvolupar posteriorment noves invencions. De les opcions que hem estudiat citarem les que tenen un mínim renom i un número mínim d'usuaris per tal de poder-les avaluar amb sentit crític:

- **Mute**: Es tracta d'un sistema totalment distribuït, anònim i que utilitza xifrat. Com a característica rellevant es que utilitza tècniques d'encaminament basat en el comportament de les formigues; d'aquesta manera va evolucionant en el pas del temps. Es doncs, dinàmic en que es creen canals de transmissió intel·ligents i amb memòria. Els nodes memoritzen el número de paquets per un cert origen A i cap a un cert destí B. Cada cop, el camí va acumulant un major número de transicions amb la qual cosa s'agafa el camí òptim. Utilitza IP's virtuals, és a dir que es tracta d'una comunicació indirecta. Aquest sistema s'utilitza més com a gestor de fitxers, com a substitut de les xarxes P2P. Per a més informació aquí deixem la pàgina web: http://mute-net.sourceforge.net

- **FreeNet**: Aquest sistema és diferent a l'anterior. S'utilitza per als mateixos propòsits que la xarxa Tor, és un model contra la censura en la qual podem estar sotmesos avui en dia per Internet, per exemple per saltar-se el gran *firewall* de Xina [FWLL]. A diferència de Tor, la idea principal del xifrat de la informació de FreeNet no és la seva ocultació sinó poder evitar que sigui censurada. Les característiques importants a remarcar són que ningú sap en un instant de temps que està compartint, ningú sap cap a on va la informació que envia. Els algoritmes d'encaminament són intel·ligents: varien segons la càrrega, segons la latència entre altres característiques. Tot i així la seva essència és la d'un sistema P2P. Per a més informació us podeu dirigir a: http://www.freenetproject.org/

- **GNUNet**: Aquest sistema es basa pràcticament en FreeNet. Introdueix canvis en el sistema d'emmagatzematge global de la informació. Utilitza un sistema de crèdits en el qual els nodes més generosos son recompensats. Com a característiques a remarcar és que la informació està completament distribuïda, és totalment anònima i està xifrada. La informació (cada fitxer) és codificada amb blocs de 1KB anomenats DBlocks. Aquests DBlocks són identificats mitjançant una estructura formada per índexs de tots ells anomenats IBlock. L'usuari només ha de conèixer les paraules que defineixen

aquests fitxer per tal de poder accedir a les dades. Per a més informació us podeu dirigir a: http://gnunet.org/

Podem observar que hi ha diversos criteris gens trivials i que determinaran en gran part els avantatges i desavantatges a l'hora d'implementar qualsevol tipus de xarxa anònima. Entre les característiques més importants ens trobem amb:

- **Transmissió de la informació**: es realitzarà sobre TCP o sobre UDP? les rutes d'encaminament seran estàtiques o seran dinàmiques?
- **Estat de la informació a nivell global**: com accedim al contingut? quines són les fonts en que podem aconseguir la informació? quina distribució de la informació utilitzem: models geogràfics? balanceig de càrrega?

Veiem doncs, que les diferents xarxes alternatives d'anonimat tenen una arquitectura radicalment diferent a la de Tor i amb la conseqüència de que no sigui possible poder realitzar un anàlisis comparatiu. Es per aquest motiu que proposarem els nostres algoritmes per la creació de camins entre nodes de la xarxa Tor.

2.2 Estudi de viabilitat

A continuació descrivim la viabilitat del projecte des de tres punts de vista diferents: el marc legal, el marc de econòmic i el material

2.2.1 Viabilitat legal

Dins la pàgina oficial citada en la secció anterior, se'ns proporciona informació de com utilitzar la xarxa, quins mecanismes calen per tal de poder utilitzar els recursos, com està implementat i fins i tot ens donen la possibilitat de col·laborar-hi ja que es tracte d'un projecte de software lliure amb llicència GPL v2. Aquesta llicència ens permet partir del codi font del projecte i modificar-lo o introduir noves funcionalitats. En cap moment realitzar aquesta acció comporta una despesa econòmica. Únicament s'han de citar els autors originals.

Fins al dia d'avui ningú ha estat denunciat per a utilitzar la xarxa de Tor als E.E.U.U. d'una manera deshonesta, tan si són simples nodes com si són nodes de sortida. Tanmateix, l'organisme encarregat de portar la legalitat de la xarxa Tor es diu Electronic Frontier Foundation [LFAQ09] i comenten que no poden assegurar de que mai ningú pugui ser denunciat ja que s'ha pensat i dissenyat per ser una eina per tal de poder parlar lliurament, respectar la privacitat i els drets humans. Tot això pot comportar problemes legals en segons quins contextos ens situem. Cal deixar ven clar que pel simple fet d'utilitzar Tor, una persona no pot ser denunciada ja que només s'està amagant darrera una xarxa com moltes més n'hi ha. El problema fonamental radica en l'ús que se li doni.

Donat a que el codi de TOR està llicenciat sota la GPL v2, legalment no hi ha cap mena d'impediment per poder alterar el codi font que ens proveïm de base, sempre i quan surtin els noms dels autors. En el nostre cas hem utilitzat el nucli del projecte per desenvolupar una branca paral·lela. Fer un ús de repetidors o utilitzar la xarxa de TOR pot estar restringit segons en quins països.

2.2.2 Viabilitat econòmica

Aquest projecte és totalment viable econòmicament ja que tot el nucli principal es de codi obert i gratuït. No hi ha cap cost per d'adquisició de mòduls o software addicional per el desenvolupament. La única inversió econòmica és una màquina per a poder muntar tot l'entorn de desenvolupament.

2.2.3 Viabilitat material

L'únic material imprescindible per a poder dur a terme el projecte és una màquina amb una connexió a Internet i un entorn de desenvolupament sobre Python. A dia d'avui inclús és possible desenvolupar a través d'un mòbil.

2.3 Planificació temporal del treball

La tècnica de planificació que hem utilitzat per tal d'estructurar les tasques dins del projecte ha estat un diagrama de Gantt ja que aquesta ens permet realitzar la planificació temporal del projecte. Seguidament us la presentem exposant els *milestones* que considerem bàsics, així com el seu corresponent diagrama associat.

2.4 Planificació del projecte

Estudi de la infraestructura de Tor i mètriques anonimat (2 setmanes):

Durant aquesta etapa del projecte estudiarem l'arquitectura de la xarxa TOR i l'algorisme criptogràfic que permet proporcionar anonimat als seus usuaris. Tanmateix, ens centrarem en analitzar quines són les diverses mètriques d'anonimat que la comunitat científica ha proposat [DIAZ02] amb l'objectiu de poder avaluar els algoritmes a implementar. Incloem en aquest període la configuració bàsica del nostre sistema per fer servir la xarxa TOR.

Estudi d'algoritmes actuals (2 setmanes):

En aquest punt profunditzarem en l'aprenentatge dels algoritmes anteriorment citats per extreure'n les seves característiques principals.

Implementació d'algoritmes (3 setmanes):

En aquest punt serà a on s'aplicarà la teoria a la pràctica, desenvolupant els algoritmes alternatius estudiats prèviament.

Estudi de possibles millores (1 setmana):

Aquí ens centrarem en valorar l'estat actual i mirar si les implementacions vistes es poden millorar en algun aspecte. Proposarem una alternativa que proporcioni un compromís entre latència i anonimat acceptable.

Implementació de les millores (3 setmanes):

En el cas de que es vegin millores a fer, serà aquí a on s'implementaran versions més eficients.

Benchmarks (1 setmana):

Aquesta serà la tasca final, que farem una valoració objectiva dels resultats.

Analitzarem els valors numèricament i gràficament.

2.5 Diagrama de Gantt

WBS	Name	Start	Finish	Work	Duration	Slack	Cost	Assigned to
1	Estudi infraestructura tor i mètriques d'anonimicitat	Apr 15	Apr 28	10d	10d		0	
2	Estudi d'algoritmes actuals	Apr 29	May 12	10d	10d		0	
3	Implementació d'algoritmes sense codi font	May 13	Jun 9	20d	20d		0	
4	Estudi possible millora	Jun 10	Jun 16	5d	5d		0	
5	Implementació millora	Jun 17	Jul 14	20d	20d		0	
6	Benchmark	Jul 15	Jul 21	5d	5d		0	
7	Escriptura memòria	Apr 15	Jul 21	70d	70d		0	

Figura 2.1: Taula milestones

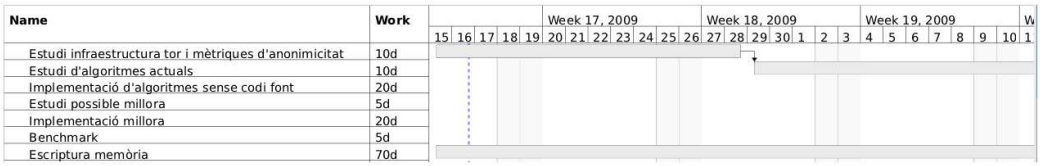

Figura 2.2: Diagrama de Gantt

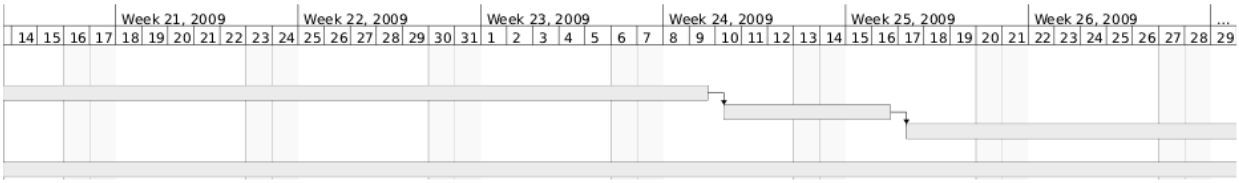

Figura 2.3: Diagrama de Gantt (segona part)

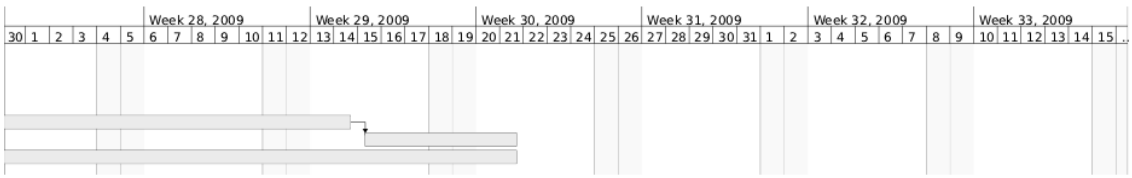

Figura 2.4: Diagrama de Gantt (tercera part)

Capítol 3

Fonaments teòrics

En aquest capítol expliquem els conceptes principals que s'han de saber per tal de poder entendre d'una manera més sòlida i profunda l'arquitectura i la funcionalitat que ens ofereix TOR. Serà per aquest motiu que ens és totalment imprescindible esmentar nocions bàsiques de criptografia. D'aquesta manera el lector es podrà arribar a fer una idea del cost computacional que porta associat el fet de mantenir comunicacions xifrades i anònimes.

Més endavant passarem a descriure el funcionament intern de TOR tot explicant el Tor Control Protocol: és el servei bàsic per a poder interactuar amb aquesta xarxa. Per concloure el capítol, parlarem també sobre dos punts importants en tota xarxa: la congestió i la seguretat.

3.1 Criptografia bàsica

El procediment utilitzat per xifrar dades es realitza mitjançant un algoritme el qual es pot considerar com a una funció matemàtica. Un criptosistema realitza una transformació matemàtica sobre les dades originals. La intercepció d'aquestes dades per un usuari no legítim no podrà recuperar la informació original en un temps computacionalment raonable. Els missatges que s'han de protegir, anomenats textos en clar, es transformen mitjançant aquesta funció o obtenint a la sortida del procés el text xifrat.

El funcionament de TOR tal i com s'ha dit a la introducció, utilitza un canal bidireccional que s'utilitza en cada etapa per tal d'intercanviar una clau Diffie-Hellman. Això és necessari per tal de garantitzar que la informació sigui consistent i no es vegi alterada. El protocol de Diffie-Hellman és el que utilitza TOR. Per tant, per tal de poder entendre amb més profunditat l'arquitectura de TOR és necessari parlar abans sobre el protocol Diffie-Hellman.

El protocol Diffie-Hellman permet l'intercanvi secret de claus entre dos punts que no han tingut un previ contacte. A més, destaca perquè s'utilitza sobre canals insegurs i no requereix cap tipus d'autenticació, tot i que en proveeixen les bases. La principal seguretat d'aquest algoritme radica en la gran dificultat de calcular logaritmes discrets en un cos finit.

Per explicar-ho de manera teòrica, procedirem a fer un seguit de definicions (veure figura 3.1):

Sigui $Alice$ i Bob cada un dels punts en quals es vol intercanviar les claus i un adversari E.

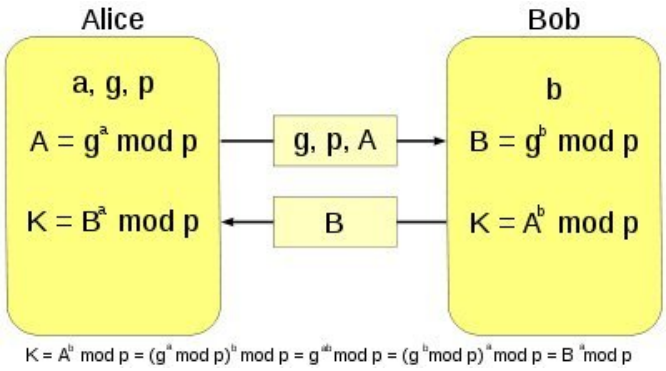

Figura 3.1: Algoritme Diffie-Hellman

En primer lloc s'estableix un número primer p i un generador g. Aquests són públics, coneguts per tothom (l'adversari està inclòs). En segon lloc, l'extrem $Alice$ escull un número a l'atzar, després calcula l'equivalència matemàtica mostrada a la figura 3.1, i aquest resultat l'envia cap a Bob. Mentrestant, l'extrem Bob escull un número a l'atzar, després calcula la mateixa equivalència matemàtica mostrada a la figura 3.1, i aquest resultat l'envia cap a $Alice$.

Es gràcies a aquestes equivalències que les dos parts la puguin calcular. El nostre adversari E no en té suficient en conèixer p, g, A i B. Necessita també un dels dos valors privats: a o b. També pot provar d'invertir la funció. Però la gràcia d'aquest algoritme és que calcular a donat b és el problema del logaritme discret, que es computacionalment costos. Tanmateix aquest protocol és sensible a atacs actius de l'estil *man-in-the-middle*. Aquests tipus d'atacs consisteixen en que l'intrús s'interposa entre els dos punts de comunicació i d'aquesta manera es capaç de llegir i alterar les dades que flueixen pel canal tot suplantant la identitat:

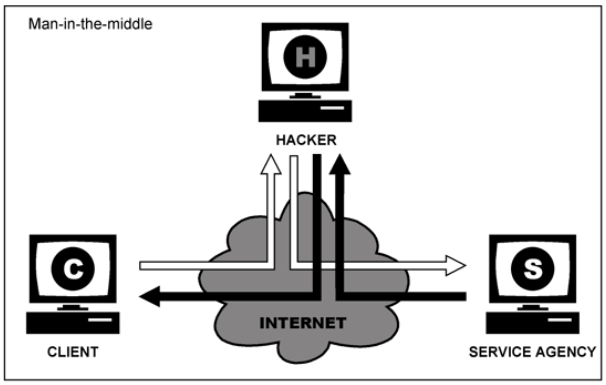

Figura 3.2: Atac Man-In-The-Middle

Al no tenir cap tipus d'autenticació, el nostre atacant E, es podria fer passar per $Alice$ o per Bob aconseguint d'aquesta manera acordar una clau amb cada participant i retransmetre les dades entre ells. Estaria escoltant la comunicació en els dos sentits.

A part del protocol Diffie-Hellman explicarem d'una manera breu un altre protocol que està molt estés en tota la xarxa Internet i particularment també a la xarxa Tor: **SSL** (*Secure Sockets Layer*).

SSL és un protocol criptogràfic que proporciona comunicacions segures per a una xarxa. Ens proporciona l'autenticació i privacitat de la informació entre extrems sobre Internet mitjançant l'us de criptografia. Generalment, només el servidor es autentificat mentre que el client es manté sense autentificar ja que l'autenticació mútua requereix d'un desplegament d'infraestructura de claus públiques per als clients. Aquest protocol implica una serie de fases bàsiques:

- Negociació entre les parts de l'algoritme que es faran servir en la comunicació.
- Intercanvi entre claus públiques i l'autenticació basada en certificats digitals.
- Xifrat del tràfic basat en xifrat simètric.

Durant la primera fase la primera fase, el client i el servidor negocien quins algoritmes criptogràfics s'utilitzaran. Les implementacions actuals ens proporcionen les següents opcions:

- Per a la criptografia de clau pública podem utilitzar els diferents algoritmes: *RSA, Diffie-Hellman* o *DSA*.
- Per al xifrat simètric podem utilitzar els diferents algoritmes: *RC2, RC3, IDEA, DES, Triple DES* o *AES*.
- Amb funcions hash: *MD5* o *SHA*.

A grans trets, el funcionament és el següent:

- En client envia un missatge especial anomenat *ClientHello* especificant una llista de conjunt de xifrats, mètodes de compressió i la versió del protocol SSL més alta permesa.
- Després, el client rep un registre *ServerHello*, en el que el servidor escull els paràmetres de connexió a partir de les opcions ofertades en el punt anterior per el client.
- Quan els paràmetres de la connexió són coneguts per les dues bandes, comencen un intercanvi de certificats.
- El servidor pot requerir un certificat al client perquè la connexió sigui mútuament autentificada.
- El client i el servidor negocien una clau simètrica secreta anomenada *master secret* possiblement fent servir el resultat d'un intercanvi *Diffie-Hellman* comentat anteriorment.

Fixem-nos que aquest protocol ens serveix per a poder utilitzar tot tipus de comunicacions segures. Es per aquest motiu que Tor utilitza per a cada node, una connexió segura. I tal i com s'ha dit amb anterioritat la clau escollida es el resultat d'aplicar el protocol de Diffie-Hellman. També hem d'esmentar que a part del SSL explicat ara mateix, també s'utilitza el TLS, que no es res més que una versió posterior i millorada del SSL però la idea principal és la mateixa.

3.2 Arquitectura de TOR

Tal i com s'ha dit anteriorment, la xarxa Tor és una xarxa que funciona per capes. Cada usuari és un *Onion Router* (OR) que executa un procés servidor amb privilegis d'usuari que per defecte escolta al port 9051, i accepta tant connexions locals com remotes. Requereix una prèvia autenticació per posteriorment poder interactuar amb el servei i el tenim implementat per múltiples arquitectures i sistemes operatius. D'aquesta manera podem utilitzar aquesta xarxa en pràcticament qualsevol tipus de màquina: un ordinador personal, una PDA o inclús un mòbil.

Cada OR manté una connexió xifrada amb els altres OR's utilitzant TLS. Per altra banda, cada usuari executa un *software* amb la funció de *proxy* o passarel·la que s'anomena *Onion Proxy* (OP). Mitjançant aquest software cada node podrà obtenir un llistat públic de nodes, establir circuits a través de la xarxa i manejar connexions. Donat que un dels serveis més utilitzats a Internet es el World Wide Web i donat que al llarg del temps hi ha hagut una evolució molt ràpida de llenguatges que ens permeten accedir a dades sensibles, s'han desenvolupat proxies amb unes característiques interessants tals com: no fan caching, tenen capacitats avançades de filtratge, modifiquen els headers HTTP i eviten anuncis, entre altres característiques. El software que es proporciona per defecte és el *privoxy* [PRIV]. Tanmateix cal remarcar que no es condició *sine qua non* utilitzar privoxy per tal d'interactuar amb la xarxa de TOR.

Aquests OP accepten fluxos TCP i s'encarreguen de gestionar-los a través de l'establiment de circuits. Al final de cada circuit creat per Onion Proxies trobem el Onion Router, que s'encarregarà de connectar amb el destinatari final i enviar les dades. Podem veure a continuació una figura il·lustrativa:

En aquesta figura podem observar els diferents tipus de nodes: els intermitjos (Onion Proxy) i els de sortida (Onion Router). Només en l'últim node la informació no està xifrada, sempre i quan no es demani de forma explícita, per exemple per autenticar-se en una pàgina segura. En tots els altres nodes la informació va xifrada.

Això s'aconsegueix ja que cada node de sortida té una clau d'identificació i una clau Onion. La primera clau ens servirà per signar certificats TLS i per signar

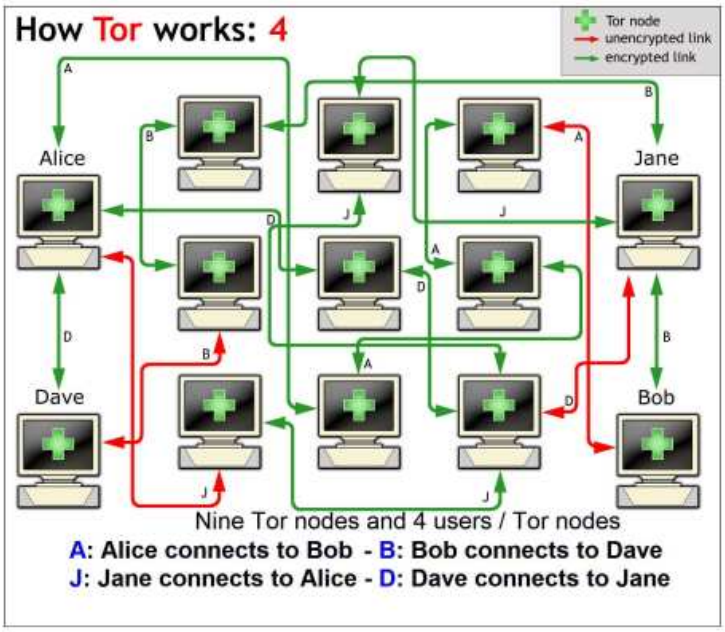

Figura 3.3: Establiment de connexions

els *routers descriptor*. La segona clau ens servirà per desxifrar les peticions dels usuaris per a la construcció de circuits. Les claus van canviant de forma regular per tal de limitar l'impacte de possibles compromisos.

3.2.1 TCP: Tor Control Protocol

En aquesta secció ens centrarem en les possibilitats que ens brinda el Tor Control Procotol. Tal i com s'ha comentat, es tracta d'un servei que està a l'espera de peticions amb el qual podrem, entre d'altres coses crear circuits (per defecte de tres nodes). Però d'on obtenim la llista de nodes (routers descriptor) i les seves característiques? A continuació us posarem un exemple il·lustratiu de com obtenir el llistat del nodes a partir del directori públic mitjançant la comanda GETINFO ns/all:

Veiem doncs que ens apareix una extensa llista on s'utilitzen dues línies per a definir cada node. Ens proporciona informació tal com el nick, el idHash, el uptime, la seva direcció IP pública i número de port de control que té, entre d'altres

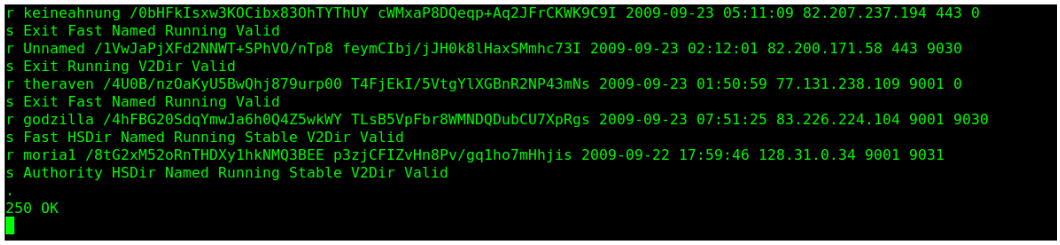

Figura 3.4: Obtenció dels nodes del llistat públic

característiques. Tota persona que col·labori com a node dintre de la xarxa Tor es veurà en principi reflexat a la llista del directori públic. Diem en principi ja que en realitat hi ha més nodes dels apareixen al llistat públic. Hi ha xarxes que tenen llistes actualitzades dels nodes que apareixen en aquest llistat, denegant l'accés i fent impossible l'encaminament cap a Internet. Un gran exemple podria ser el gran *firewall* de xina, que per qüestions socio-polítiques imposen restriccions mitjançant gestors de continguts, entre d'altres mecanismes. El mateix govern no se n'amaga i proporciona una *URL* per tal d'executar un test sobre un domini per saber si està restringit: **http://www.greatfirewallofchina.org/test/**. Tot i així un usuari pot aconseguir d'una manera no gens trivial nodes de sortida que no estiguin reflexats a la llista pública.

A partir d'aquest llista, ja podrem construir els nostres circuits segons les característiques que creiem oportunes. És important remarcar que tots els circuits que s'estableixen estan orientats a connexió. Si es vol crear un canal no orientat a connexió s'haurà d'encapsular dintre de segments TCP, per exemple, per realitzar trucades VoIP.

En tots els casos, amb independència del criteri de selecció de nodes, el procés és sempre el mateix: un cop s'han agafat els nodes desitjats per a tal de crear un circuit (o diversos circuits), per a cada node es negocia una clau compartida utilitzant l'algoritme Diffie-Hellman comentat en el punt anterior. La idea és dir-li en el nostre primer node del circuit que es connecti en el següent node que volem per a la construcció del circuit. I anirem iterant fins a trobar l'últim node: el de

sortida. Serà aquest últim l'encarregat d'efectuar el forwarding cap al servidor a on l'usuari hagi fet la petició. Cada node de la xarxa TOR pot ser utilitzat per múltiples usuaris, amb la qual cosa ens podem trobar en que un node pot formar part de n circuits. El circuit pot ser tant llarg com es desitgi. Com més llarg sigui el circuit, més anonimat tindrem, però més temps trigarem a fer totes les negociacions. A part que tindrem més possibilitats que el circuit es trenqui per la possible no disponibilitat d'algun node en algun moment. Les ordres que s'envien d'un node en un altre són simples tot seguint l'estàndard RFC 2234 [RFC2234]. Això ens permetrà inclús fer proves manuals des d'una línia de comandes. Les ordres les podrem separar en dos grans grups:

- Events síncrons: aquest grup d'ordres s'executarà quan nosaltres li diguem. Podem controlar perfectament quan succeeixen aquests events. Alguns exemples d'aquestes ordres són: *GETINFO*: així és com obtenim la llista de nodes del servidor públic. *ATTACHSTREAM*: vincular un stream a un circuit.

- Events asíncrons: en canvi aquest grup d'ordres s'executarà en funció de com estigui el sistema. No podem controlar quan succeeixen aquests events. A més, no només la nostre màquina els pot causar. Qualsevol *host* que estigui a la xarxa ens pot desencadenar aquest tipus d'events. Alguns exemples d'aquestes ordres són: *CircStatus*: notificacions de canvis d'estat d'un circuit.

Veiem un exemple de com es realitza una petició en un *site* web de forma anònima mitjançant aquestes simples comandes:

```
SETEVENTS EXTENDED CIRC STREAM
250 OK
650 STREAM 9240 SENTCONNECT 754 www.conspiracyplanet.com:80
650 STREAM 9241 NEW 0 obamanati.info:80
650 STREAM 9241 REMAP 0 174.132.115.60:80 SOURCE=CACHE
650 STREAM 9241 SENTCONNECT 754 174.132.115.60:80
650 CIRC 764 EXTENDED VSvTZGO7UPj4yh8,xanadu PURPOSE=GENERAL
```

Figura 3.5: Interactuació amb el port de control de TOR

Podem trobar totes les comandes disponibles en la especificació del Tor Control Protocol:

https://git.torproject.org/checkout/tor/master/doc/spec/control-spec.txt. En la *release* estable, la creació de circuits es basa segons els criteris introduïts per Renner i comentats en el capítol anterior. Aquests criteris són tres: l'ample de banda, localització geogràfica i el temps que fa que un node existeix en la xarxa.

3.2.2 Congestió i seguretat

Dos conceptes importants en tota xarxa son la **congestió** i la **seguretat**. És impossible implementar una xarxa sense congestió però si que es pot intentar erradicar el problema de la millor manera. El mateix ens passa amb la seva seguretat. Ara passarem a explicar-los per aquesta xarxa particular amb més detall:

- **Congestió**: és un fenomen produït quan en una part de la xarxa o en la seva totalitat se li ofereix més tràfic del que pot processar. Una de les causes més freqüents són: insuficiència en els commutadors i insuficiència de capacitat de càlcul en els nodes.

  Cada node té una configuració sobre quin ample de banda brinda a la resta de la xarxa amb la qual cosa sembla que d'aquesta manera es pugui resoldre el problema. Però tot i així, tot i tenint l'ampla de banda limit ens hem de preocupar per la congestió, ja sigui perquè sorgeix de forma accidental o ja sigui perquè ha estat causada de forma intencionada. Si suficients usuaris agafen el mateix node de sortida per a la creació de circuits, pot ser que aquest quedi totalment saturat. Serà mala sort. Però si que un usuari malintencionat pot enviar un fitxar molt llarg a través de la xarxa cap a un servidor web que corre i després negar-se a llegir qualsevol dels bytes de la web. Aquests colls d'ampolla es poden propagar per a tota la xarxa. Per solucionar aquests problemes, no cal tornar a implementar les finestres del *TCP/IP* ja que com que Tor funciona sobre TCP, ens garanteix que el lliurament de la informació estarà ordenada. Aquest problema el tractem en dos nivells:

  - Nivell de circuit:

Per tenir un control sobre la utilització d'ampla de banda per a un circuit utilitza dues finestres. La primera ens serveix per a saber quanta informació es capaç de processar el OR. La segona ens serveix per a saber quanta informació podem enviar a fora de la xarxa.

Cada finestra s'inicialitza en un valor determinat. Quan s'envien dades es decrementa la finestra corresponent. Quan un OR ha rebut suficient informació envia el que es diu un *relay sendme* cap al node, amb un identificador del circuit a zero. Quan un OR rep un relay sendme comença a augmentar el tamany de la finestra. Passa el mateix amb un node intermig

– Nivell de fluxos:

  El control en el nivell de fluxos es similar a l'utilitzat per als circuits. Tant els nodes intermitjos com els nodes de sortida utilitzen el relay sendme per a implementar un control aïllat per cada flux.

- **Seguretat**:

  En tota xarxa la seguretat és un aspecte clau a tenir en compte. No només ens interessa que ens brindi anonimat. Necessitem també una seguretat.

  Des de sempre, tot tipus de xarxa ha tingut més o menys problemes en aquest aspecte. És un fet inevitable. Quan implementem una xarxa ja estem establint comunicacions i d'alguna manera o altre es poden veure afectades a diversos nivells. En el cas Tor, tenint en compte que es una xarxa que funciona a nivell d'aplicació, és a dir, per sobre TCP, tindrem els problemes associats en la capa TCP i d'afegit els que pugui aportar Tor. Veiem a continuació d'una forma breu els principals problemes que es poden desencadenar dintre de la xarxa Tor [SGOR04]:

  - Denegació de Servei: conegut també per *Denial Of Service*. El principal d'aquest tipus d'atacs es causar que un servei o un recurs sigui inaccessible als usuaris legítims, provocant generalment la pèrdua de connectivitat de la xarxa per el consum d'ample de banda de la víctima.

    Un dels típics atacs que s'han registrat a la xarxa de Tor consisteix a forçar els OR's perquè realitzin operacions criptogràfiques computacionalment costoses. Per exemple, un atacant pot simular diversos inicis de negociacions de claus forçant que el OR faci càlculs criptogràfics recurrents causant d'aquesta manera que el seu processador treballi al límit. Actualment no hi ha la manera per fer front a aquest atac.

  - Abús de les polítiques de sortida: Donada la naturalesa de la xarxa Tor, quan un client realitza una petició a qualsevol *host* causa que aquest no detecti el client, només el node de sortida. És per aquest motiu que hi ha d'haver restriccions en els nodes de sortida. Cada node de sortida té la configuració de la seva política de seguretat. Per exemple, per defecte es denega el *SMTP*, sinó fos així seria el lloc idoni per a poder fer *spam*, per exemple. Segons els responsables de Tor, a dia d'avui encara no han trobat cap indici d'abús sobre els nodes de sortida. Tot

i així és un factor a tenir molt en compte.

– Directori de serveis: Tal i com hem comentat amb anterioritat, la funció del directori de serveis és la de proveir al client tots els nodes públics disponibles de la xarxa Tor per tal de que a partir d'aquest creïn els seus circuits.

  Si un atacant controla un servidor de directori, pot presentar la informació d'una manera distorsionada. Per exemple podria mostrar només els nodes que ell té sota control, aconseguint d'aquesta manera que els circuits que l'usuari crei el node de sortida estarà probablement controlat per l'atacant i d'aquesta manera és possible, depenent si la connexió final no va sobre *SSL* o *TLS* que pugui realitzar una plena captura de les dades.

3.3 Conclusions

Hem començat fent una breu introducció al món de la criptografia que és totalment imprescindible per tal de poder entendre amb més profunditat l'interior de Tor.

A continuació hem pogut veure en profunditat la capa més baixa de comunicació: el Tor Control Protocol i el mecanisme que utilitza per comunicar-se amb els altres *hosts* així com la construcció de circuits. Hem esmentat les principals comandes que ens brinda aquesta arquitectura classificats en dos grans grups: els events síncrons i els events asíncrons. Finalment hem parlat sobre la congestió i seguretat.

Capítol 4

Disseny i implementació

Aquest capítol l'hem dividit en dues parts. En la primera part anomenada disseny explicarem al lector el TorCtl, que és un mòdul de Python que implementa el Tor Control Protocol. Tanmateix explicarem les diferents eines i utilitats que hem anat utilitzant al llarg d'aquest projecte així com el disseny dels diferents algoritmes proposats en forma de pseudocodi, sense entrar en detalls.

En la segona part anomenada implementació passarem a explicar la basant comuna de tots els algoritmes. Els descriurem amb precisió, cadascun en un apartat diferent. En aquesta secció comentarem els principals problemes que ens hem trobat durant el desenvolupament algorítmic. Conclourem el capítol exposant les conclusions que hem obtingut.

4.1 Disseny

Durant el desenvolupament d'aquest projecte hem utilitzat un servidor de control de versions SVN. D'aquesta manera podem veure d'una forma ràpida, clara i precisa els canvis que s'han anat fent en el codi. Tindria més sentit si hi haguessin més desenvolupadors en aquest projecte. Tanmateix l'emmagatzemament és més òptim ja que realitzem còpies incrementals. Com a IDE hem utilitzat Eclipse SDK versió 3.3.2 amb un plug-in anomenat *Python Development 1.4.4.2636* ja que tot el desenvolupament està fet en Python. És un entorn amigable en el qual ja hi

estàvem familiaritzats i d'aquesta manera s'agilitzen les tasques.

En el capítol anterior hem parlat sobre el procés que segueix cada node per a interactuar dins la xarxa Tor. Tot i que es poden construir circuits a través d'un terminal mitjançant comandes, la forma més habitual es dissenyar programes que automatitzin aquesta tasca i poder canviar valors a temps real: número total de nodes per als circuits, procedència geogràfica dels nodes de sortida, entre moltes altres possibles opcions. Aquests programes s'ajudaran de mòduls o llibreries per tal de poder desenvolupar d'una manera més fàcil i ràpida. A continuació us explicarem el mòdul que hem utilitzat per tal de desenvolupar els algoritmes proposats al llarg del projecte.

Tot i que el codi font del *kernel* o nucli del Tor Control Process està implementat en llenguatge *C*, el TorCtl, que és el modul que hem utilitzat nosaltres, està desenvolupat en *Python* [TPPT09]. Òbviament necessita que estigui corrent el procés de Tor per a poder enviar i rebre ordres.

TorCtl ens dona suport per a la creació de camins amb diverses restriccions en els nodes i en la selecció de circuits així com obtenir estadístiques del que fem. Podem desenvolupar a dos nivells d'abstracció:

- En la primera capa d'interactuacio s'utilitza la classe TorCtl. Típicament ho farem important el TorCtl.TorCtl i creant una connexió TorCtl.Connection i estenent de TorCtl.EventHandler. Aquesta ultima classe fa de pont entre el procés de Tor i la resta de classes. S'encarrega de gestionar els possibles events, tant síncrons com asíncrons, que hi pugui haver.

- En la segona capa s'utilitza la classe PathSupport. Ho farem important el TorCtl.PathSupport i instanciant o estenent de PathSupport.PathBuilder, que aquesta ultima s'estén de TorCtl.EventHandler. Aquesta classe es l'encarregada de gestionar la construcció de circuits, fluxos i de poder associar fluxos a circuits. Les classes PathSupport.NodeRestrictor i PathSupport.PathRestrictor ens ajudaran a sotmetre aquests *streams* i circuits a les polítiques que creiem oportunes.

En el nostre cas, hem optat per desenvolupar els algoritmes mitjançant la pri-

mera capa. L'avantatge de treballar en aquesta capa és que és molt més flexible i es poden implementar coses que treballant a la segona capa ja estan implementades d'una manera que potser no ens es eficient. El desavantatge es que s'han d'implementar coses que possiblement ja les trobaríem implementades o ens costarien menys desenvolupar-les.

A continuació, en les següents subseccions, passem a descriure d'una manera esquemàtica el disseny dels tres algoritmes principals i finalment el de millora.

4.1.1 Algoritme aleatori

El primer algoritme proposat de tots consisteix en crear un circuit amb una selecció totalment aleatòria sense tenir en compte cap tipus de variable (uptime, geoIP, etc). La única restricció serà que l'últim node sigui de sortida. És amb aquest algoritme que aconseguirem el màxim anonimat possible, ja que no ens basarem en cap tipus de paràmetres per a realitzar la selecció de nodes. Per contra serà la selecció menys favorable des de el punt de vista de rendiment en quant a la latència, ja que ens trobarem que al fer la selecció totalment aleatòria podem agafar amb la mateixa probabilitat un node que tingui una connexió de 20MB/s o una connexió que funcioni per dial-up de 5KB/s. Al no tenir tampoc en compte el uptime, podem seleccionar nodes que la seva estabilitat a la xarxa no sigui molt gran causant la ruptura del circuit. Aquest fet implica que augmenti la probabilitat d'haver de crear més circuits. A continuació us mostrem el pseudocodi associat:

```
llistaDeNodes = obtenirNodesXarxaTor()
per i ← x fins longitud( llistaDeNodes ) fer
    si i.nodeSortida = Cert llavors
        llistaNodesSortida.afegir[i]
    fi si
    si i.nodeSortida = Fals llavors
        llistaNodesIntermitjos.afegir[i]
   fi si
fi per
index ← random( 1..longitud( llistaNodesSortida ) )
nodeSortida ← nodeSortida[index]
llistaNodesIntermitjos ← barrejar(llistaNodesIntermitjos)
mentres numMaxNodes > 1 fer
   llistaFinalNodes ← afegir(llistaNodesIntermitjos[numMaxNodes])
   numMaxNodes ← numMaxNodes - 1
fi mentres
tancarCircuitsPrevis()
circuitID ← extendreCircuit( 0, llistaFinalNodes )
```

D'una manera més visual, si considerem un cas hipotètic que tots el països del món tenen nodes de sortida de Tor, un possible circuit seria el de la següent figura:

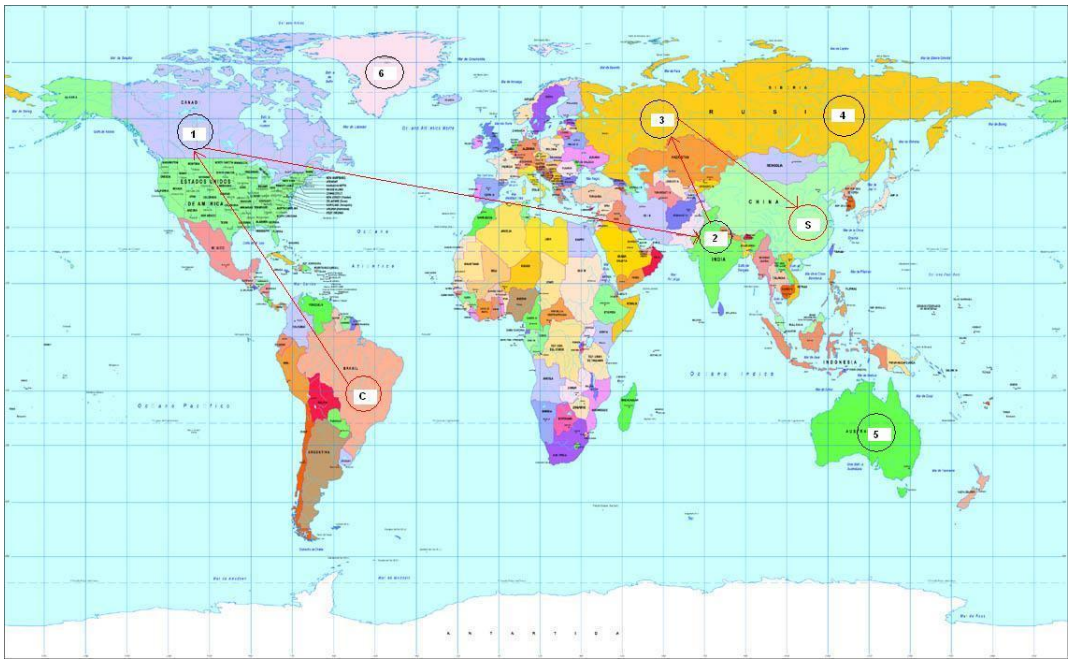

Figura 4.1: Establiment d'un circuit aleatori

En la figura 4.1 il·lustrem un exemple d'un circuit on el client es localitza al Brasil i està disposat a realitzar una connexió a Xina primer passant per Canada, després de per la Índia i finalment per Rusia.

4.1.2 Algoritme GeoIP

El segon algoritme proposat només té un paràmetre en compte, que és la localització geogràfica. Introduint aquest canvi respecte el primer algoritme, aconseguirem per una banda tenir una latència teòrica més baixa però com a contra el nostre anonimat es veurà afectat. Hi hauran menys nodes totals per fer les seleccions ja que els nodes seran del mateix país. A partir d'aquest criteri de selecció es farà exactament el mateix que en el primer cas. Crearem un circuit dels quals un d'ells serà el node de sortida. A continuació us mostrem de manera simplificada el pseudocodi associat:

```
llistaDeNodes = obtenirNodesXarxaTor()
ourCountry ← obtenirPais( nostreIPPublica )
per i ← x fins longitud( llistaDeNodes ) fer
    si i.nodeSortida = Cert llavors
      si i.pais = ourCountry llavors
       llistaNodesSortida.afegir[i]
      fi si
    fi si
    si i.nodeSortida = Fals llavors
      si i.pais = ourCountry llavors
       llistaNodesIntermitjos.afegir[i]
      fi si
   fi si
fi per
index ← random( 1..longitud( llistaNodesSortida ) )
nodeSortida ← nodeSortida[index]
llistaNodesIntermitjos ← barrejar(llistaNodesIntermitjos)
mentres numMaxNodes > 1 fer
   llistaFinalNodes ← afegir(llistaNodesIntermitjos[numMaxNodes])
   numMaxNodes ← numMaxNodes - 1
fi mentres
tancarCircuitsPrevis()
circuitID ← extendreCircuit( 0, llistaFinalNodes )
```

A continuació us mostrem una il·lustració per veure la construcció del circuit sobre nodes del mateix país:

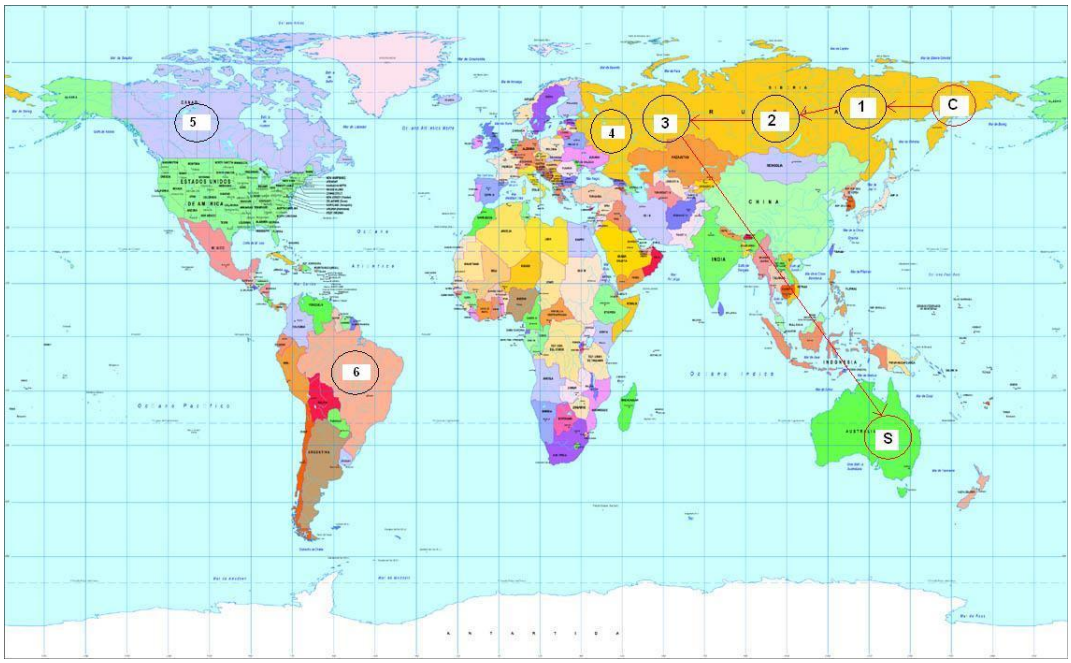

Figura 4.2: Establiment d'un circuit aleatori dintre el mateix país

4.1.3 Algoritme GeoIP millorat

El tercer algoritme proposat consisteix en que el primer node que seleccionem sigui del mateix país que la nostra direcció IP pública. S'utilitza l'estratègia del segon algoritme. El segon node (en el cas d'haver-hi N, s'agafen N-2) és totalment aleatori utilitzant l'estratègia del primer algoritme. I finalment, el node de sortida serà del mateix país en el qual hem fet la petició. És doncs un híbrid entre els dos primers algoritmes però es més anònim que l'anteriorment comentat ja que el destí final pot variar a cada petició (considerem tots els països diferents a on esta implantat Tor). Perquè el lector es faci una idea de quina és la variació respecte els dos algoritmes predecessors, a continuació us el presentem en forma de pseudocodi. Cal remarcar que s'utilitza tot el pseudocodi de l'algoritme anterior i aquí nomes us mostrarem l'ampliació.

```
procediment streamStatusEvent(self, event):
//comprovem si hi ha un nou flux de dades ( stream )
si event.status = NEW || event.status = NEWRESOLVE llavors
    llistaStreams ← guardarNouStream( event.streamID )
    buscarNodeAleatoriAmbCodiPais( event.hostDestinacio )
    //ara procedim a vincular el nou stream amb un dels circuits
    attachStream( event.streamID, circuitID )
fi si
fi procediment
```

A continuació us mostrem la il·lustració

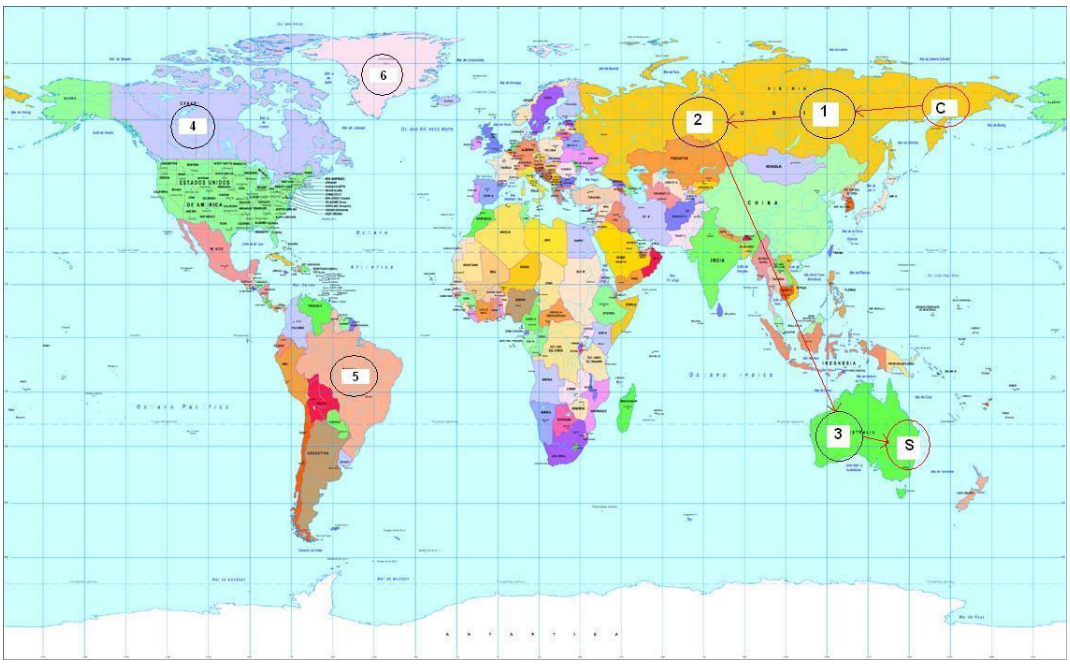

Figura 4.3: Establiment d'un circuit aleatori dintre el mateix país però millorat

4.1.4 Proposta de millora: Algoritme Graf amb latències

Aquesta proposta de millora s'inspira en l'algoritme aleatori però ampliat. Consisteix en construir circuits segons el mateix criteri del primer algoritme: agafarem els nodes de forma aleatòria. A diferencia d'abans, per cada circuit que vulguem crear en crearem molts més. Per què ho fem això? doncs perquè aplicarem un algoritme per tal d'obtenir el camí de cost mínim en quant a latències sobre els circuits creats addicionals. Ens quedarem doncs, amb el millor circuit que ens retorni l'algoritme de cost mínim. Fixem-nos en el fet de com més temps estigui executant-se l'algoritme, més òptim serà el camí ja que s'hauran avaluat més possibilitats i els camins resultants tindran uns pesos en els arcs més petits. El punt òptim seria que el graf format per tots els nodes de la xarxa Tor tingués la màxima connectivitat: K_n, on n és el número de nodes. Després d'aplicar l'algoritme, el nostre candidat l'emmagatzemarem en una llista de circuits. Teòricament, després d'aplicar l'algoritme de cost mínim, el candidat final serà el circuit més eficient en quant a la latència. Per contra, computacionalment serà l'algoritme que tindrà una complexitat temporal i espacial més gran. Perquè quedi més il·lustrat el funcionament intern, a continuació us mostrem el pseudocodi associat:

```
procediment obtenirLlistaNodes()
llistaDeNodes = obtenirNodesXarxaTor()
per i ← x fins longitud( llistaDeNodes ) fer
    si i.nodeSortida = Cert llavors
        llistaNodesSortida.afegir[i]
    fi si
    si i.nodeSortida = Fals llavors
        llistaNodesIntermitjos.afegir[i]
   fi si
fi per
index ← random( 1..longitud( llistaNodesSortida ) )
nodeSortida ← nodeSortida[index]
llistaNodesIntermitjos ← barrejar(llistaNodesIntermitjos)
mentres numMaxNodes > 1 fer
   llistaFinalNodes ← afegir(llistaNodesIntermitjos[numMaxNodes])
   numMaxNodes ← numMaxNodes - 1
fi mentres
tancarCircuitsPrevis()
circuitID ← extendreCircuit( 0, llistaFinalNodes )
fi procediment
circuitsAavaluar = obtenirLlistaNodes()
circuitOptim = floydWarshall( circuitsAavaluar )
```

A continuació us mostrem una il·lustració del funcionament de l'algoritme de proposta de millora:

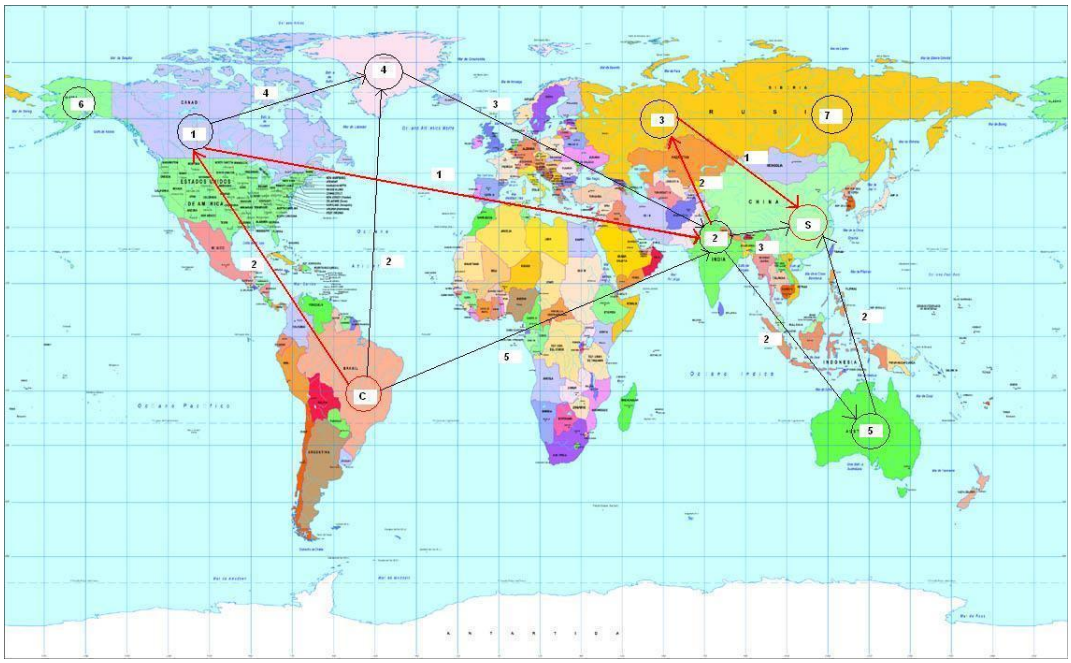

Figura 4.4: Establiment d'un circuit aleatori dintre el mateix país però millorat

Fixem-nos en la figura 4.4 a on surt representat un graf amb múltiples camins. En aquest exemple podem observar que els nodes 6 i 7 no estan connectats ja que no hi ha hagut temps a que l'algoritme hagi calculat les seves latències. Finalment podem observar el resultat final d'aplicar l'algoritme de cost mínim, el camí de color vermell.

4.2 Implementació

La implementació d'aquest projecte han estat bàsicament tres algoritmes de selecció de nodes i un últim que representa una millora substancial respecte els tres anteriors. Els nodes, tal i com s'ha dit amb anterioritat, són obtinguts d'una llista pública de nodes Tor. Cada node tindrà una sèrie de característiques diferents: a quin continent està, quin ample de banda té, quan temps fa que el node està en funcionament, si és un node entremig o és un node de sortida, entre d'altres.

En els tres algoritmes, el número de nodes per defecte són de tres, tal i com està implementat en el mòdul de Tor. Però a part, tenim la possibilitat d'ampli-

ar el número de nodes per tal de poder fer posteriorment proves i presentar les gràfiques.

En tots els algoritmes hi ha una manera de procedir igual que passem a descriure a continuació.

Creem un *socket* que l'utilitzarem per connectar-nos al port de control. Realitzarem aquesta vinculació. Després passarem pel procés d'autenticació i finalment definirem una classe que es cridarà cada cop que hi hagi events, ja siguin síncrons o asíncrons. Aquesta última classe, anomenada *MyHandler(TorCtl.EventHandler())* és la part comuna més important i els seus mètodes són:

- El mètode *circStatusEvent(self, circEvent)*: en aquest mètode ens encarregarem de controlar l'estat del circuit. A grans trets, si el circuit el tenim en un estat *launched*, l'afegirem al nostre diccionari de circuits juntament amb el seu *timestamp* per tal de poder obtenir dades temporals amb posterioritat. Per contra, si tenim el circuit en un estat de *failed* o *closed* el que farem serà suprimir-lo de la nostre llista de circuits.

- El mètode *streamStatusEvent(self, event)*: en aquest mètode ens encarregarem d'un cop estan construïts els circuits, gestionar els fluxos que generarem al fer qualsevol petició utilitzant els recursos de la xarxa Tor. A grans trets, quan tenim un event *new* o *newresolve* el que farem serà també gravar el temps d'arribada i posteriorment realitzarem una vinculació entre el flux que s'acaba de generar en un circuit existent determinat. Per altre banda, quan ens trobem en events del tipus *failed* o *closed* procedirem a suprimir-lo del diccionari.

  Respecte al desenvolupament dels algoritmes, hem tingut algun moment d'entrebanc, sobretot el tercer i la proposta de millora. Hem de tenir en compte que, tot i que el desenvolupament ha estat realitzat en un entorn amigable, el *TorFlow* utilitza *threads* i aquest fet fa que a vegades no sigui trivial poder seguir el flux d'execució i limitar-nos a entendre la situació mitjançant missatges per consola.

  Al principi és difícil triar el camí, ja que un cop descarregat el codi font

del *TorFlow* veiem que té un tamany considerable ja que ens ofereix moltes possibilitats. El fet de triar unes classes i no unes altres comporta unes coses o unes altres que a mida que va avançant el projecte ens anem adonant.

Dins el codi font de *TorFlow* ens hem ajudat d'un paquet que s'anomena *Torify*. Aquesta utilitat forma part del paquet d'utilitats per a mesurar rendiments de la xarxa en diferents condicions. Concretament *Torify*, a través de SOCKS 4.5, realitza una connexió local cap al port de control de Tor, podent mesurar els temps de *streams*.

Un cop explicada els problemes comuns dels algoritmes, anem a explicar les particularitats de cadascun d'ells.

4.2.1 Algoritme aleatori

Respecte a la implementació d'aquest algoritme, tal i com s'ha dit a la secció anterior de disseny, al no considerar cap variable a l'hora de seleccionar els nodes, implica que l'estabilitat a la xarxa no sigui molt gran, causant d'una manera habitual la ruptura del circuit i dificultant-ne la seva avaluació. Aquest fet implica que augmenti la probabilitat d'haver de crear més circuits.

Realment és l'algoritme més fàcil d'implementar però per contra el que més costa realitzar la construcció dels circuits i també en la que un cop s'han creat els circuits, els circuits fallin. Això és degut a diversos factors que sorgeixen al no tenir en compte aquesta avaluació.

4.2.2 Algoritme GeoIP

La implementació d'aquest algoritme és potser un pèl més complicada que l'anterior però a l'hora de construir els circuits per avaluar el rendiment, no falla tant com en el primer algoritme.

4.2.3 Algoritme GeoIP millorat

Aquesta implementació al principi semblava que no seria tant diferent a les altres anteriors. Conceptualment no ho és però si que hi ha petits però importants canvis en la part del codi. El principal problema és l'assignació a temps real del node de sortida segons les restriccions. Tot i que a la pràctica, l'estabilitat dels circuits no varia molt respecte el segon algoritme.

4.2.4 Proposta de millora: Algoritme Graf amb latències

Durant la implementació d'aquest últim algoritme hem tingut la necessitat d'avaluar les diferents alternatives que hi ha de trobar els costos mínims d'un graf. Ens hem decantat per utilitzar l'algoritme de *Floyd-Warshall*, ja que aquest ens permet optimitzar grafs dirigits. D'aquesta manera, hem utilitzat estructures per tal de representar arcs ja que el temps d'un node A a un node B pot ser diferent dels temps d'un node B a un node A. Hem valorat altres algoritmes, com el de *Dijkstra* però té el problema que només serveix per a grafs amb arestes. Si que és cert que també l'haguéssim pogut utilitzar, però haguéssim perdut precisió en els resultats al tenir només en compte només un sentit.

4.3 Conclusions

Al llarg d'aquest capítol hem pogut exposar al lector l'entorn que s'ha utilitzat per tal de dur a terme el desenvolupament del projecte.

A continuació s'han descrit les diferents alternatives de disseny que hi ha per a poder construir tot tipus d'algoritmes de selecció de nodes.

Per finalitzar s'han exposat un per un els diferents algoritmes que havíem pensat juntament amb una proposta de millora i la seva corresponent implementació. Notem que els quatre algoritmes presentats són diversos criteris

de selecció que hem decidit però tanmateix se'n poden crear infinits més. Les possibilitats hi són. Caldrà tenir en compte en qualsevol implementació els avantatges i desavantatges que aquest comporta. Entre les més importants ens hauríem de qüestionar si serà útil, si ens aportarà un rendiment superior i si serà computacionalment tractable a temps real.

Capítol 5

Experiments i resultats

En aquesta secció presentarem els resultats aconseguits segons els objectius que teníem fixats a priori.

Començarem aquest capítol descrivint al lector l'entorn de proves que s'ha realitzat per tal d'obtenir els resultats: el tipus de màquina, les xarxes, els horaris en que s'han efectuat i els número de nodes, entre d'altres.

Finalment passarem a presentar els resultats mitjançant una gràfica per a cada algoritme proposat juntament amb els seus intervals de confiança respectius.

5.1 Experiments

Les proves que hem fet sobre els algoritmes proposats les hem realitzat sobre diferents criteris, tals com xarxes, número de nodes i experimentant amb la longitud dels circuits. A continuació s'explica amb més precisió.

- Xarxes: hem realitzat les proves en la xarxa de la *Universitat Autònoma de Barcelona*, de la *Universitat Politècnica de Catalunya*, de la *Universitat de Barcelona*, en el centre cívic de Sarrià *Casa Orlandai* i en dues línies ADSL de diferents proveïdors. Sabem que tot i fer

les proves en diferents xarxes, en el fons són segurament massa semblants. En el cas de les universitats, absolutament totes utilitzen els recursos de l'anella científica.

- Horaris: és important considerar llençar els mateixos testos en diferents franges horàries. Hi ha algoritmes en que es veu molt més afectat aquest factor, com per exemple el *GeoIP* i n'hi ha d'altres que pràcticament no s'aprecia la diferència, com per exemple l'*aleatori*.
- Sistema Operatiu: hem utilitzat sempre el mateix sistema operatiu: Linux version 2.6.27.25-78.2.56.fc9.i686
- Número de nodes: tal i com podrem observar posteriorment en les gràfiques, hem fet els testos sobre circuits de tres nodes (per defecte en les implementacions), sobre quatre i cinc.

Per a realitzar les proves, tal i com hem comentat ara mateix, les hem sotmès a una serie de paràmetres per tal de poder obtenir les dades de la manera més objectiva possible.

Hem realitzat diferents combinacions entre els paràmetres exposats abans. Sobre una mateixa màquina, hem llançat un *script* automatitzat sobre diferents xarxes i en intervals de temps diferents. Sorprenentment, els millors resultats han estat obtinguts en general, des de la xarxa de la *Universitat Politècnica de Catalunya*, seguidament de la *Universitat Autònoma de Barcelona* i de la *Universitat de Barcelona*. La variació no ha estat molt significativa entre les xarxes universitàries si ho comparem amb els resultats obtinguts en línies ADSL convencionals.

Hem procedit en adquirir les dades per cada xarxa (en total tres mostres, una per cada interval de temps diferent) i al final hem fet una mitjana que és la que en la següent secció us presentem mitjançant gràfiques.

Un aspecte comú en totes els experiments es que si provem de construir circuits de longitud superior a vuit nodes, el sistema triga molt, de l'ordre de minuts.

Per a poder realitzar les proves temporals, ens hem concentrat en mesurar únicament el temps que triguen els diferents circuits en descarregar per complert la següent pàgina:

https://git.torproject.org/checkout/tor/master/doc/spec/control-spec.txt.

Cal remarcar que aquesta pagina es estàtica ja que ens interessa que sempre hi hagi la mateixa quantitat de dades a transmetre.

5.2 Presentació dels resultats

Al llarg d'aquesta secció anirem comentant cadascun dels algoritmes vistos anteriorment. Presentarem la informació en forma de gràfiques juntament amb els seus intervals de confiança.

5.2.1 Algoritme aleatori

En aquesta gràfica representem el temps de descàrrega del contingut d'una pàgina web estàtica:

https://git.torproject.org/checkout/tor/master/doc/spec/control-spec.txt

D'aquesta manera ens assegurem que les gràfiques siguin més fidels. Passem a mostrar la gràfica:

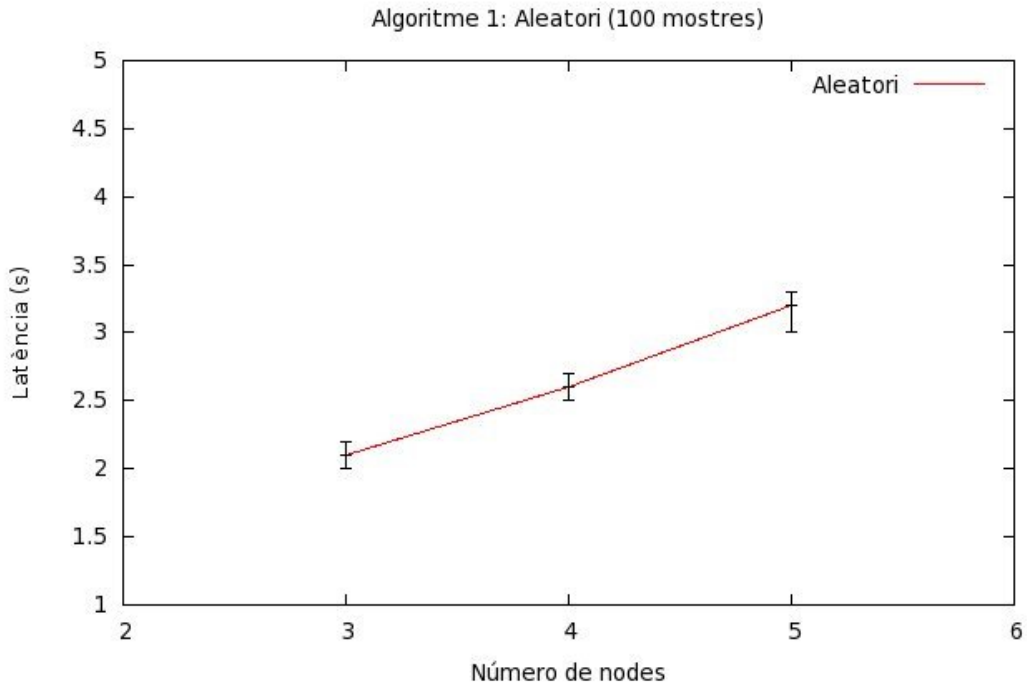

Figura 5.1: Relació entre número de nodes i latència

5.2.2 Algoritme GeoIP

Curiosament, les observacions empíriques extretes d'aquest algoritme no són molt bones. Totes les proves s'han fet desde Espanya.

- Per una banda la quantitat total de nodes espanyols que formen part de la xarxa Tor és extremadament petita. Consta d'un total de 8. Aquest resultat tant baix condiciona als resultats ja que la majoria de cops els circuits que s'estableixen són exactament els mateixos.
- Per altre banda la teoria apunta que si els nodes són del mateix país, la latència serà més baixa. Doncs bé, els nodes espanyols són extremadament lents.

Un cop feta aquesta observació, procedim a mostrar la gràfica:

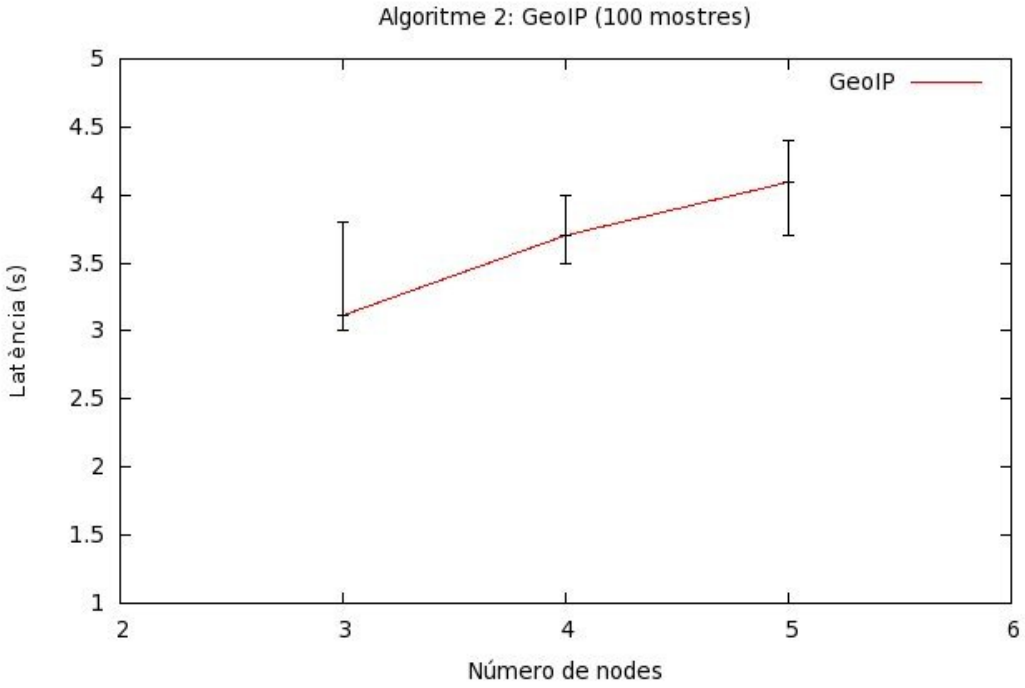

Figura 5.2: Relació entre número de nodes i latència

5.2.3 Algoritme GeoIP millorat

Les observacions empíriques d'aquest algoritme no són molt satisfactòries però es pot observar una relació de latència *versus* número de nodes superior al algoritme anterior. A continuació us mostrem la gràfica:

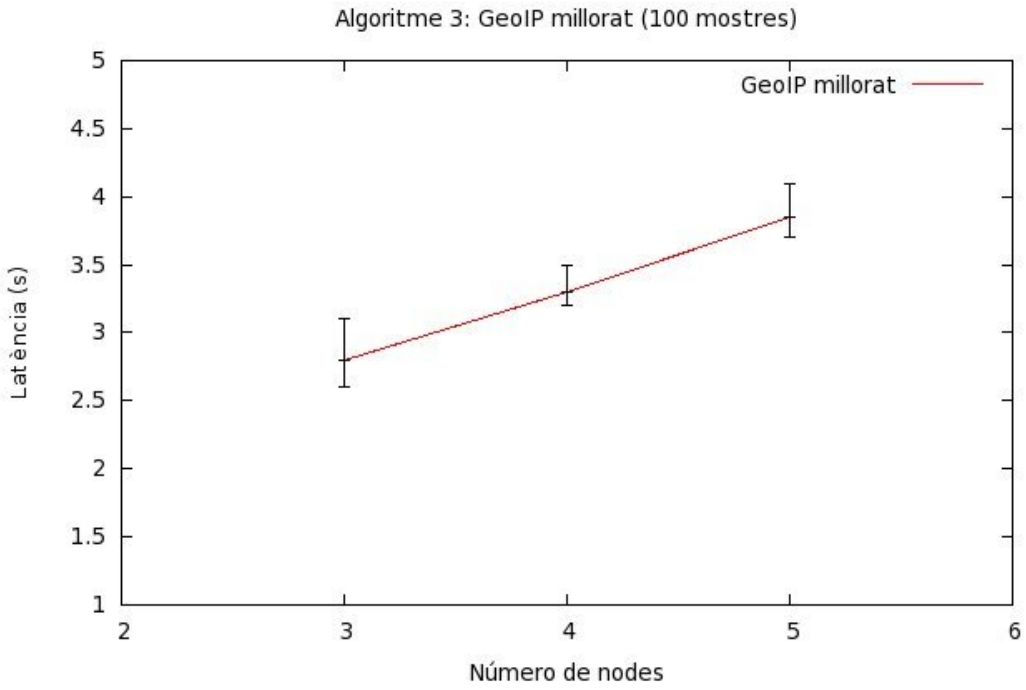

Figura 5.3: Relació entre número de nodes i latència

5.2.4 Proposta de millora: Algoritme Graf amb latències

Els resultats de la implementació d'aquest algoritme de proposta de millora han estat els més satisfactoris. Tal i com s'ha comentat amb anterioritat en el capítol de disseny, la idea principal és anar creant una estructura de graf a temps real i anar calculant les latències entre cada parell de nodes. Després aplicarem l'algoritme de cost mínim per tal de reduir la latència. En els resultats exposats a continuació, hem deixat executant l'algoritme 120 segons abans de fer les peticions. Teòricament. en un temps infinit tindríem un graf complet amb totes les latències avaluades. Donat que aquest fet és totalment impossible, ens hem vistos obligats a realitzar un retall temporal. Podem afirmar segons els resultats extrets que a partir dels 50 segons d'execució de l'algoritme, en general, hi ha suficient informació sobre les

latències per a la creació de nous *streams*. No obstant, a partir d'aquest número de segons, tot i tenir més possibilitats, l'error numèric no difereix tant. A continuació us mostrem la gràfica corresponent:

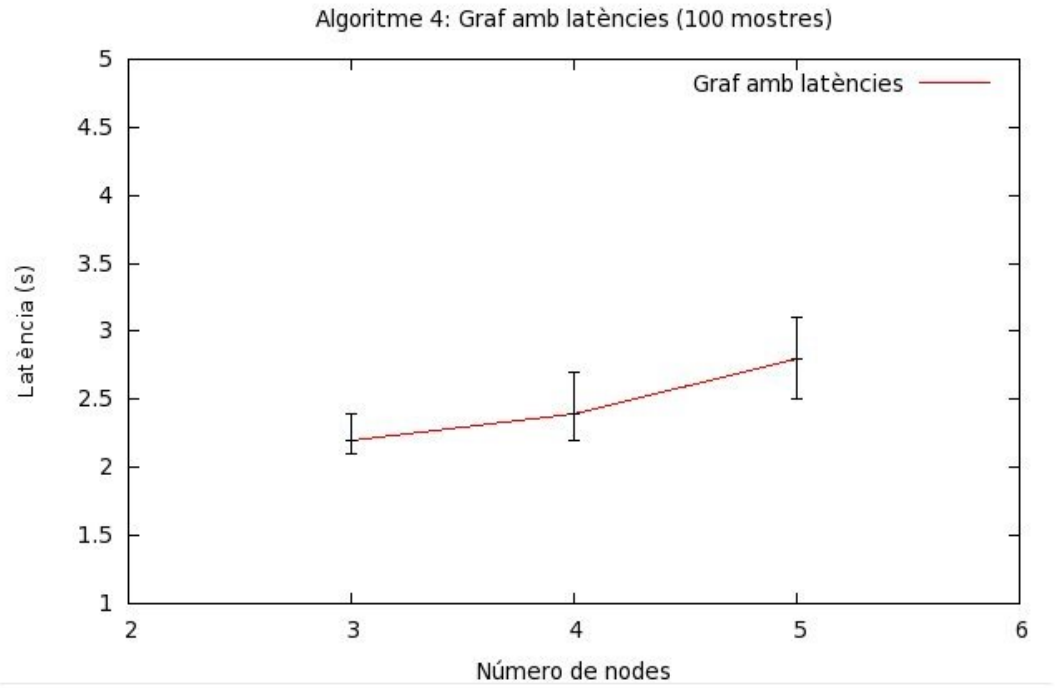

Figura 5.4: Relació entre número de nodes i latència

5.3 Conclusions

En aquest capítol hem exposat al lector d'una manera precisa els resultats pràctics en forma de gràfiques.

Observant les gràfiques mostrades al llarg d'aquest capítol podem concloure que:

- L'algoritme aleatori és un dels que millors resultats ens ha donat en termes de latència.

- La segona proposta, tot i aparentment hauria de ser donar més bons resultats que el primer, això no és aixi ja que tal i com hem comentat anteriorment, els nodes els hem provat desde Espanya. I el rendiment d'aquests nodes no és gens bo.
- En el tercer algoritme, podem observar que el seu rendiment en quan a latència està situat entre els dos anteriorment comentats. És més eficient que l'algoritme GeoIP però no tant com l'aleatori. Tot i que teòricament no hauria de ser aixi.
- En l'algoritme de proposta de millora, podem observar que per a tres nodesté un rendiment bastant bo, deixant enrera l'algoritme GeoIP i el GeoIP millorat. Però quan comencem a augmentar el número de nodes, es posiciona el primer de forma significativa.

Capítol 6

Conclusions

En aquest capítol, exposarem les conclusions finals que hem extret al llarg de tot aquest projecte. Primer de tot començarem per explicar els coneixements adquirits, a continuació comentarem de tots els objectius marcats *a priori* quines metes hem assolit. Després exposarem les conclusions personals obtingudes sobre els algoritmes que hem desenvolupat i sobre el projecte de *TorFlow*. Per finalitzar el capítol transmetrem idees sobre possibles millores i amplicacions que es podrien realitzar arrel d'aquest projecte.

6.1 Coneixements adquirits

Aquest projecte m'ha servit per entendre d'una manera profunda la xarxa de Tor, tant a nivell teòric com a nivell pràctic. Al principi no va ser gens fàcil, ja que m'he hagut de familiaritzar amb un nou llenguatge que no havia vist fins al moment: Python. Aquest llenguatge, com tots, tenen les seves coses bones i dolentes, però una vegada ens hi acostumem veiem que és molt potent ja que es tracta d'un llenguatge imperatiu i funcional de manera simultània. A part podem trobar informació en abundància a la xarxa així com àmplies llibreries de desenvolupament.

Per altra banda, donat que es partia d'un codi font obert i que aquest projecte ha estat desenvolupat a partir d'aquest codi, la primera tasca fonamental i prioritària ha estat entendre bé quines accions realitzava el codi. Hi ha numeroses parts que no estan molt ben documentades i/o la *Application Programming Interface* està desactualitzada. Això comporta a que el desenvolupament s'allargui ja que ens trobem amb parts que no són gaire intuïtives.

Tanmateix, per a realitzar les implementacions, es poden fer a diversos nivells que la llibreria de codi obert ens proporciona. Valorar els avantatges i els desavantatges que té implementar algoritmes en una capa o en una altre capa ha estat una tasca complexa però al mateix moment quan es convergeix en una solució, és reconfortant.

Hem aprés com comunicar-nos via telnet amb el procés de Tor i poder tenir un absolut control sobre ell: obtenir els nodes, crear els circuits pertinents i fins i tot reproduir la funcionalitat dels dos primers algoritmes a mà, gràcies a principalment un document on s'especifica el protocol de Tor.

Un altre factor important que m'ha donat uns coneixements addicionals, ha estat el fet de tenir un criteri més sòlid i ampli a l'hora de fer una *suite* de testos així com la manera de presentar les dades en forma de taula o gràfic de la manera més objectiva possible juntament amb els errors de precisió.

Finalment, he adquirit coneixements addicionals gràcies a gent del irc-oftc.net (Internet Relay Chat), en el canal de Tor channel, on hi havia els principals desenvolupadors del projecte Torflow. Allà m'han donat algunes idees i m'han explicat certs conceptes que estan previstos per desenvolupar en les pròximes *releases*: usabilitat en els *add-on* dels navegadors, optimització del nucli de Tor, traduccions i generació de documentació tant per els usuaris com pels desenvolupadors i un llarg etcètera.

6.2 Objectius complerts

En el projecte s'havia definit com a objectiu principal tres algoritmes de selecció de nodes dels quals s'ha pogut realitzar perfectament la seva implementació funcional. A part, s'han intentat millorar els tres algoritmes anteriors mitjançant la implementació d'un quart, que radica bàsicament en un hibrid dels tres anteriors. Tal i com s'ha comentat anteriorment, en el capítol quart, tots els algoritmes tenen la possibilitat d'estendre el circuit tant com es vulgui.

Un altre objectiu principal a part de la tasca pròpia de desenvolupar era la manera com obtenir les dades temporals i com poder-les mostrar en forma de gràfics sense que suposés una gran complicació per a la interpretació del lector.

6.3 Conclusions obtingudes

Al principi, no hagués dit mai que estigués tant evolucionat el projecte Torflow. He quedat gratament sorprès que entre molts col·laboradors s'hagi pogut aconseguir un producte que, tot i que encara li queda una llarga línia de desenvolupament, en el moment actual podem afirmar que ens ofereix moltes possibilitats i que funciona de forma correcte. Pot ser que aparentment no ho sembli, però no es trivial fer el disseny i la implementació d'una xarxa amb aquestes característiques que hem anat veient al llarg d'aquesta memòria. Sí que és cert que la idea principal està basada en implementacions de xarxes anònimes prèvies però tot i així no li treu mèrit.

És també curiós veure que l'activitat d'aquesta xarxa pot variar moltissim en funció del país a on s'estigui o inclús nodes d'altres països que responen en general d'una manera més ràpida que nodes del mateix país.

Al llarg del desenvolupament d'aquest projecte ens hem anat fixant de forma general amb l'origen dels nodes que ofereixen la seva col·laboració, tant

com en nodes interns com en nodes de sortida. El resultat és que pràcticament més del setanta per cent del conjunt de nodes pertanyen als Estats Units i a Alemanya. Lògicament n'hi ha de molts altres països però la seva presència a la xarxa no és tant significativa. Per posar un simple exemple: a Espanya, el número total de nodes que figuren al llistat públic es de vuit. És una xifra realment molt baixa que no ens ofereix un gran ventall de possibilitats. Però com tot projecte, tot requereix el seu temps. Gràcies a la història de la xarxa Tor ens podem arribar a fer una idea de l'evolució creixent que ha tingut els últims temps tenint en compte que la seva existència és de fa pocs anys.

També s'ha de dir que el nucli principal de Tor, el TorCtl, m'ha deixat gratament sorprès en quant a les possibilitats que ens brinda i és relativament fàcil que un desenvolupador s'implementi els seus propis algoritmes de selecció de camins. Hi ha diverses funcions i porcions de codi a mode d'exemple que poden servir com a inspiració. A part, té un bon sistema per emmagatzemar els temps de creació de circuits. Tot i ser un codi plenament funcional en el qual s'ha explotat tota potència de llenguatge, hi ha parts que haurien d'estar més degudament comentades ja que a vegades no es fàcil intuir que està realitzant exactament.

Respecte les conclusions sobre les implementacions, podem arribar a concloure que el tercer algoritme, el GeoIP millorat és el que té la millor relació entre el temps que es triga en desenvolupar i els resultats que ens ofereix.

La implementació del graf amb latències és costós d'implementar i sinó ens esperem molta estona a que es vagin calculant les diferents latències entre nodes, no obtindrem uns bons resultats.

L'algoritme aleatori és molt fàcil i ràpid d'implementar però clar, sempre ens porta a l'obtenció de resultats imprevisibles, causant que no es pugui confiar plenament.

Per acabar les conclusions, hem de comentar que la segona implementació, la GeoIP, no ens ha donat bons resultats. Tanmateix hauríem d'haver provat

des de diferents països per tal de poder jutjar d'una forma més justa. El problema ha estat també que no es tant fàcil poder establir connexions *Virtual Private Network* als diferents països per tal de poder alterar la nostre direcció IP pública.

Les característiques que hem trobat més rellevants dels algoritmes presentats en aquest projecte es troben resumits en la seguent taula:

	Comparativa entre algoritmes		
	Latència	Anonimat	Dificultat implementació
Algoritme 1	ALTA	EXCEL·LENT	TRIVIAL
Algoritme 2	ACCEPTABLE	REGULAR	FÀCIL
Algoritme 3	ACCEPTABLE	REGULAR	NORMAL
Algoritme 4	BAIXA	MOLT BO	COMPLICAT

6.4 Futures millores i ampliacions

Tot projecte sempre es pot millorar de forma significativa en diversos aspectes. En un nivell teòric una de les millores que es podrien fer seria buscar expressions matemàtiques per tal de poder calcular la entropia teòrica per a cadascun dels algoritmes. D'aquesta manera, el capítol de la presentació dels resultats quedaria més complet. Al llarg d'aquest projecte, hem intentat fer una adaptació de la mètrica d'anonimat que que es fa servir en l'article de Claudia Díaz [DIAZ02]. Malauradament la cerca d'aquestes expressions no són gens trivials, ja que s'han de considerar moltes variables. Queda doncs, pendent per a un futur treball.

En un nivell pràctic, el cas d'aquest projecte, es podria millorar el software dotant-lo d'un motor d'intel·ligència artificial per tal de que anés aprenent sobre les accions que l'usuari acostuma a parametritzar per configurar el seu anonimat.

Com que tot projecte informàtic, si funciona i té alguna utilitat, l'acaben utilitzant tard o d'hora els usuaris finals. Per aquest motiu es podria desenvolupar un *add-on* per al navegador diferent al que hi ha ara, no només amb més possibilitats, sinó també que tingués un entorn més amigable i més intuïtiu.

Tot això sense deixar de banda la interfície gràfica, que estaria molt bé que, mitjançant wxPython tenir la possibilitat de poder interactuar d'una forma directe i intuïtiva amb el motor de Tor, sense haver de conèixer els detalls. Actualment, el programa que podem trobar que s'acosta més en aquesta descripció s'anomena *Vidalia*.

Si la gent es preocupa per a poder expressar amb total llibertat els seus pensaments, més gent utilitzara aquesta xarxa i més eficient serà en termes d'anonimat. Hi haurà més nodes per a poder seleccionar i es podran triar millors camins per a un moment concret. El fet de que en una xarxa hi hagi un augment de la conectivitat voldrà dir que va adquirint més resó social i que hi ha més usuaris que la utilitzen. Però això ja no és una millora de Tor, és una futura millora que necessitem que el major número possible d'usuaris col·laborin: la unió fa la força!

Bibliografia

[CLARK07] Clark, J.; Oorschot, P. C. van; Adams, C.: Usability of anonymous web browsing: an examination of Tor interfaces and deployability In the Proceedings of the 3rd Symposium on Usable Privacy and Security (SOUPS '07), Pittsburgh, Pennsylvania, July 2007, p41-51.

[DIAZ02] Diaz, C.; Seys, S.; Claessens, J.; B. Preneel: Towards measuring anonymity In the Proceedings of Privacy Enhancing Technologies Workshop (PET 2002), April 2002.

[MCCOY08] McCoy, D.; Bauer, K.; Grunwald, D; Kohno, T.; Sicker, D.: Shining Light in Dark Places: Understanding the Tor Network In the Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008), Leuven, Belgium, July 2008, p63-76.

[ERLIER07] Erlier, L.; Syverson, P: Improving Efficiency and Simplicity of Tor circuit establishment and hidden services In the Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007), Ottawa, Canada, June 2007.

[ARES08] Panchenko, A.; Pimenidis, L.; Renner, J.: Performance Analysis of Anonymous Communication Channels Provided by Tor. In Proceedings of the Third International Conference on Availability, Reliability and Security (ARES 2008), Barcelona, Spain, March 2008. IEEE Computer Society Press.

[NDSS08] Snader, R.; Borisov, N.: A Tune-up for Tor: Improving Security and Performance in the Tor Network In the Proceedings of the Network and Distributed Security Symposium - NDSS'08, February 2008.

[TCP09] Dingledine, R.; Mathewson, N.: Tor Control Protocol Specifications <https://git.torproject.org/checkout/tor/master/doc/spec/control-spec.txt>

[FSC09] fscked.org <http://fscked.org/projects/torctl>

[TPPT09] Tutorial de Python <http://mundogeek.net/tutorial-python/>

[LFAQ09] Legal FAQ for Tor Relay Operators <http://www.torproject.org/eff/tor-legal-faq.html>

[TPS09] Dingledine, R.; Mathewson, N.: Tor Path Specifications <https://git.torproject.org/checkout/tor/master/doc/spec/path-spec.txt>

[MPRY07] Perry, M.: Securing the Tor Network <http://www.blackhat.com/presentations/bh-usa-07/Perry/Presentation/bh-usa-07-perry.pdf>

[JREN07] Johannes Renner. Performance-Improved Onion

[PRIV] Privoxy: Filtering Web Proxy, http://www.privoxy.org

[GUSAGR02] Gummadi, K.P.; Saroiu, S.; Gribble, D.: "King: Estimating Latency between Arbitrary Internet End Hosts", in Proceedings of the SIGCOMM Internet Measurement Workshop (IMW 2002), Marseille, France, November 2002

[BEFEK00] erthold, O.; Federrath, H.; K¨opsell, S.: "Web MIXes: A system for anonymous and unobservable Internet access," in Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, H. Federrath, Ed. Springer-Verlag, LNCS 2009, July 2000, pp. 115–129.

[RFC2234] Request For Comments www.ietf.org/rfc/rfc2234.txt

[SGOR04] http://www.torproject.org/tor-design.pdf

[NETX] http://networkx.lanl.gov/index.html

[FWLL] http://www.greatfirewallofchina.org/test/

Signat:

Bellaterra, Febrer de 2010

Resumen

En este proyecto se presenta un estudio y una implementación sobre estrategias de selección de nodos de la red TOR. Para esto en una primera parte teorica se da a conocer todo lo que comportan las redes anónimas y su implementación en diferentes entornos. A continuación se estudian los requerimientos y la viabilidad del proyecto. Finalmente exponemos el diseño y desarrollo de los distintos algoritmos propuestos juntamente con las pruebas realizadas y las conclusiones a las que se ha llegado.

Resum

En aquest projecte es presenta un estudi i una implementació sobre estratègies de selecció de nodes de la xarxa TOR. Per això es mostra en una primera part teòrica on es dóna a conèixer tot el que comporten les xarxes anònimes i la seva implantació en diferents entorns. A continuació s'estudien els requeriments i viabilitat del projecte. Finalment exposem el disseny i desenvolupament dels diferents algoritmes proposats juntament amb les proves realitzades i les conclusions a les que s'ha arribat.

Abstract

This project presents a study and an implementation on selection strategies TOR network nodes. For this reason, we present in a first theoretical part wich is made known all that behave anonymous networks and it's implementation in different environments. Here we study the requirements and viability of the project. Finally we discuss the design and development of individual proposed algorithms together with the testing results and conclusions we have reached.