Entorn de simulació de fallades per a xarxes òptiques de transport

Marc Manzano Castro

Índex

Contents **i**

Introducció **xi**

0.1 Objectius xi

0.2 Abast xi

0.3 Agraïments 1

1 Teoria de xarxes òptiques **3**

1.1 Concepte de Survivability 3

1.2 Introducció 4

1.2.1 L'era de SONET/SDH 5

1.2.2 L'ascens de WDM i DWDM 6

1.2.3 Xarxes òptiques de transport 7

1.2.4 Xarxes multicapa 7

1.3 G/MPLS 8

1.3.1 Generalized MPLS 11

1.4 Enrutament 13

1.4.1 Routing Information Protocol (RIP) 14

1.4.2 Open Shortest Path First (OSPF) 14

1.5 Protecció i Restauració 15

1.5.1 Classificació de models de recuperació de fallades 16

1.5.2 Models de recuperació de fallades de G/MPLS 19

1.5.3 El model de protecció 1+1 20

1.6 Propagació de fallades 21

2 Conceptes i tecnologies de simuladors 23

2.1 Introducció 23

2.2 Principals simuladors de xarxes 25

2.2.1 OPNET 26

2.2.2 NS-2 27

2.2.3 OMNet++ 28

2.2.4 GTNetS 29

2.2.5 Shawn 29

2.3 OMNet++ 30

2.3.1 Instal·lació 32

2.3.2 Configuració i funcionament d'un petit example 34

3 Modelització de la propagació de fallades a una xarxa 37

3.1 Introducció 37

3.2 Model NLDS 37

3.3 Model SIC 40

4 Disseny i implementació 43

4.1 Organització del projecte 43

4.2 Metodologia 45

4.3 Implementació de l'entorn de simulació 47

4.3.1 Primer contacte amb OMNet++: Tic Toc 47

4.3.2 Model TicToc Extès 51

4.3.3 Model NLDS 54

4.4 Implementació del Model NLDS 54

4.4.1 El node: Txc1 56

4.4.2 L'enllaç: FailureChannel 56

4.4.3 La topologia 56

4.4.4 L'administrador: myScenarioManager 59

4.4.5 El recollidor d'estadístiques: statistics 60
4.5 Implementació del Model SIC 61
4.5.1 El node: RSVP_LSR / RSVP_FAILED 62
4.5.2 L'enllaç: FailureChannel 66
4.5.3 La topologia 66
4.5.4 L'administrador: myScenarioManager 67
4.5.5 Component: FailureManager 67
4.5.6 El recollidor d'estadístiques: statistics 68

5 Resultats de la simulació 71
5.1 Simulació amb una topologia de 100 nodes 71
5.1.1 Primera simulació 72
5.1.2 Segona simulació 75
5.2 Simulació amb una topologia de 200 nodes 77
5.2.1 Primera simulació 78
5.2.2 Segona simulació 80

6 Conclusions 83

7 Treball futur 85

Bibliography 89

Índex de figures

1.1 OXC (optical cross-connect) de 3 x 3 amb dos longitud d'ona per fibra 6
1.2 Exemple d'una xarxa lògica definida per sobre d'una xarxa física . . . 7
1.3 Representació d'una xarxa multicapa 8
1.4 Arquitectura MPLS 10
1.5 Capçalera MPLS 11
1.6 Exemple d'OSPF 15
1.7 Components d'un domini de protecció a G/MPLS 17
1.8 Classificació de provisionament de camins 18
1.9 Classificació de d'assignació de recursos 18
1.10 Model de protecció global 20
1.11 Model de protecció revertit 20
1.12 Model de protecció local 21
1.13 Model de protecció 1+1 21

2.1 I·lustració d'un event, d'una activitat i d'un procés 25
2.2 El simulador OMNeT++ en l'entorn Linux 32

3.1 Taula de símbols 38
3.2 El model SIS vist des de un node en concret. 39
3.3 Diagrama d'estats del model SIC 41

4.1 Evolució temporal 44
4.2 Estimació en hores 44

4.3 Repartiment temporal 45
4.4 Visualització de l'execució de la simulació 51
4.5 Topologia del model TicToc Extès 52
4.6 Arquitectura del model NLDS aplicat a OMNet++ 55
4.7 Interfície gràfica de BRITE 57
4.8 Execució topologia Nsfnet14 amb β=0.5 i δ=0.5 60
4.9 Execució topologia Cost266 amb β=0.5 i δ=0.5 61
4.10 Execució topologia de 1000 nodes creada amb BRITE amb β=0.5 i δ=0.5 61
4.11 Components de RSVP_LSR 63
4.12 Component de RSVP_FAILED 65
4.13 RSVP_FAILED i RSVP_LSR després d'afegir el component RouterState a tots dos 66
4.14 Topologia emprada per a testejar el sistema 67
4.15 Topologia de 100 nodes just després del moment en que s'infecten aleatòriament un 20% dels nodes 68
4.16 Topologia de 100 nodes amb uns quants nodes CAIGUTS abans de fer el canvi de mòdul RSVP_LSR per RSVP_FAILED 69
4.17 Topologia de 100 nodes amb uns quants nodes CAIGUTS després de fer el canvi de mòdul RSVP_LSR per RSVP_FAILED 69

5.1 Percentatge de nodes infectats al llarg de la simulació 73
5.2 Número total de connexions creades respecte el número total de connexions caigudes 74
5.3 Rànking dels primers 15 nodes que han tingut una pèrdua de connexions en la simulació 75
5.4 Percentatge de nodes infectats al llarg de la simulació 76
5.5 Número total de connexions creades respecte el número total de connexions caigudes 77
5.6 Percentatge de nodes infectats al llarg de la simulació 79
5.7 Número total de connexions creades respecte el número total de connexions caigudes 80
5.8 Percentatge de nodes infectats al llarg de la simulació 81

5.9 Número total de connexions creades respecte el número total de connexions caigudes 82

Índex de taules

1.1 Tecnologies de commutació de GMPLS 12

Introducció

0.1 Objectius

Amb aquest projecte es vol assolir l'objectiu de construir un entorn de simulació que permeti l'anàlisi de les fallades d'una xarxa òptica de transport intra-domini que proveeix connexions basades en camins lògics (G/MPLS). La simulació es basarà en un model que consideri els elements estàtics de la xarxa (topologia, capacitat total dels enllaços, equipament de transmissió, etc) i la dinàmica emergent del servei prestat (camins lògics creats entre cada parella de nodes, utilització dels recursos que això genera, i la concurrència d'esdeveniments de fallades i el seu impacte sobre les connexions establertes en en aquell mateix instant.

L'entorn permetrà establir criteris per a l'ocurrència d'esdeveniments de les fallades a simular. Aquests criteris es veuran reflectits en forma de paràmetres inicials que es passaran a l'entorn.

A partir d'aquesta proposta principal, s'ha decidit aplicar el desenvolupament de l'entorn de simulació al simulador de xarxes OMNet++. Per tal d'arribar a assolir aquest objectiu final, s'haurà de dedicar tota una etapa del treball a conèixer bé el funcionament del simulador i de les llibreries de protocols implementades per aquest.

0.2 Abast

Finalment, l'entorn de simulació implementat proporcionarà el següent:

- Un entorn de simulació de xarxes òptiques amb interfície gràfica capaç de, a partir de uns certs paràmetres i d'una topologia donada, analitzar al llarg del temps quin efecte tindrien certes fallades en aquella topologia i, així doncs,

determinar quins punts de la nostra xarxa s'hauríen de protegir per tal que una fallada afectés mínimament al servei prestat per aquesta.

- Una sèrie d'eines capaçes de traduir matrius d'adjacència a topologies de xarxa en llenguatge entès per OMNet++ i de definir tot un rang d'IPs per a cadascuna de les interfícies de tots els routers que formen una xarxa.

0.3 Agraïments

Als meus pares, pel seu interés, suport i ànims durant tota la meva formació acadèmica, professional i personal.

A en Juan, per tota l'ajuda que he rebut de part seva, per totes les coses que he aprés i que em queden per aprendre d'ell i per tota la paciència que ha tingut amb mi.

A l'Eusebi, per tranquilitzar-me i donar-me suport quan ho he necessitat.

A tots els membres del laboratori del grup Broadband Communications and Distributed Systems, per l'ajuda rebuda quan la necessitava.

Sense aquestes persones aquest projecte de final de carrera no hagués estat possible; moltes gràcies pel recolzament rebut i sobretot, per creure en mi.

Capítol 1

Teoria de xarxes òptiques

1.1 Concepte de Survivability

A làmbit de les xarxes, *survivability* és el terme utilitzat per descriure la capacitat d'una xarxa per a seguir funcionant i proveïnt serveis durant i després de la detecció d'una fallada. La traducció equivalent al català de *survivability* seria "tol·lerant a fallades". Així doncs quan es produeix una fallada, una xarxa tol·lerant a fallades ha de tenir facilitats alhora de proveïr un o més serveis alternatius. Aquests serveis alternatius poden oferir un rendiment més baix comparat amb el del servei habitual, i per tant ens trobaríem enfront d'un servei degradat respecte l'inicial.

Survivability, com a característica del sistema, necessita ser mesurada. Per mesurar la *survivability* d'una xarxa cal que s'evaluïn els següents punts:

- La freqüència d'ocurrències de condicions anormals a través de l'anàlisi d'esdeveniments de fallades.
- L'impacte causat al rendiment per aquestes condicions en termes de duració de parades.
- L'impacte de les fallades al conjunt del sistema.

La necessitat de tenir una noció de *survivability* no queda clara del tot a primera vista perquè, apart de que hi puguin haver especificacions incorrectes, un sistema deixa de proveir servei només quan es produeix una fallada, i per una gran varietat de sistemes, les fallades poden ser tol·lerades o evitades (per exemple, utilitzant components de alta qualitat). No obstant, com que els recursos són sempre limitats, *survivability* ofereix una compensació entre funcionalitat i el consum de recursos.

Per exemple, si es requereix que una xarxa global internacional tingui una disponibilitat elevada, i existeix el risc de que sigui l'objectiu d'un cert atac, els recursos necessitats per tal d'implementar redundància per protegir el servei de qualsevol tipus d'atac serien prohibitius, degut a què hi haurien grans parts del sistema que hauríen de ser duplicades completament. La compensació de la què es parla doncs, significa que alhora del disseny, s'ha de decidir quin tipus de fallades es consideren irrecuperables i quines altres haurien de ser tractades. La conseqüència és que el cost es redueix però, per altra banda, augmenta el potencial de fallades. Si la taxa de fallades està per sota del que és considerat acceptable, i si el servei alternatiu està preparat per tots els possibles escenaris de fallades, llavors la xarxa serà una *survivable network* sota les condicions donades.

Addicionalment, el terme *survivable network* es defineix al RFC 4427 [1], com una xarxa çapaç de restaurar el tràfic quan es produeix una fallada", mentres que *network survivability* es defineix com "el conjunt de possibilitats que permeten a una xarxa restaurar un tràfic afectat per una fallada. El grau de *survivability* es determina per la capacitat de la xarxa a sobreviure a una o múltiples fallades". Aquestes definicions tenen la virtut de ser simples, però al mateix temps ignoren que el tràfic podria restaurar-se selectivament.

1.2 Introducció

En el decenni de 1980 va començar una revolució en les xarxes de telecomunicacions generada per la utilització d'una nova tecnologia: el cable de fibra òptica. Des de llavors, l'enorme estalvi de costos i l'augment de la qualitat de la xarxa ha portat a molts avenços en les tecnologies necessàries per a les xarxes òptiques, els beneficis de la qual tot just es comencen a valorar.

Les xarxes de telecomunicacions han evolucionat durant un segle d'història dels avenços tecnològics i canvis socials. Les xarxes que una vegada havíen proveït serveis bàsics de telefonia a través d'una xarxa local, avui transmeten l'equivalent a milers d'enciclopèdies per segon. A través d'aquesta història, la xarxa digital ha evolucionat en tres etapes fonamentals: asíncrona, síncrona i òptica.

Les primeres xarxes digitals eren asíncrones. En aquest tipus de xarxes, és l'emissor el que decideix quan s'envia el missatge. Per tant, el receptor no sap mai quan rebrà un missatge i com a conseqüència el missatge ha de dur una informació

intrínseca de quan comença el missatge (el seu inici) i quan termina (el seu final) de manera que el receptor conegui la informació que ha de descodificar. Aquesta tècnica de transimissió de dades freqüentment pot resultar en errors de bit.

Mentre el desplegament de la fibra òptica anava en augment , no existien estàndards que imposessin com els elements de la xarxa havien de modular la senyal òptica. Així doncs un gran nombre de mètodes de propietat van aparèixer, dificultant als proveïdors de telecomunicacions l'interconnexió d'equips entre diferents fabricants.

La necessitat de tenir un estàndard comú per a totes les xarxés òptiques va impulsar la creació de xarxes òptiques síncrones (SONET i SDH). SONET, a més a més d'estandaritzar diferents paràmetres, va definir els tipus de components de la xarxa que es necessitàven, les arquitectures de xarxa que els fabricants podien implementar, i la funcionalitat que cada node havia de dur a terme. A partir d'aquest moment els proveïdors de telecomunicacions podien utilitzar equipament òptic de diferents fabricants amb la confiança de que disposaven d'una mínima interoperabilitat.

Un aspecte de SONET que l'ha permés sobreviure durant una època d'enormes canvis a les necessitats de capacitat de la xarxa és la seva escalabilitat.

1.2.1 L'era de SONET/SDH

Fins fa poc temps, la tecnologia que prevalia a la capa 1 i que era subjacent a les xarxes de transport era SONET/SDH, dos tecnologies orientades a circuits que van ser desenvolupades amb la finalitat de substituir PDH (Plesiochronous Digital Hierarchy). Les xarxes de transport van experimentar una revolució enorme a principis del 1990 gràcies a SONET/SDH.

Així doncs, SONET/SDH es va convertir en la tecnologia fundadora de les xarxes de transport, perquè quan es va introduir, les empreses tot just es trobaven a les etapes inicials del desplegament de les xarxes de fibra òptica. Aquest fet va ajudar a que s'establissin uns estàndards en quant a les interfícies fabricades per diferents empreses.

1.2.2 L'ascens de WDM i DWDM

A finals dels anys 1990 una nova tecnologia va sorgir del no res, just a temps per assegurar que la creixent demanda de capacitat a les xarxes seria satisfeta: WDM (Wavelenght Division Multiplexing). En paraules senzilles, aquesta tecnologia dona la capacitat de transmetre al mateix temps més de una senyal òptica per una mateixa fibra.

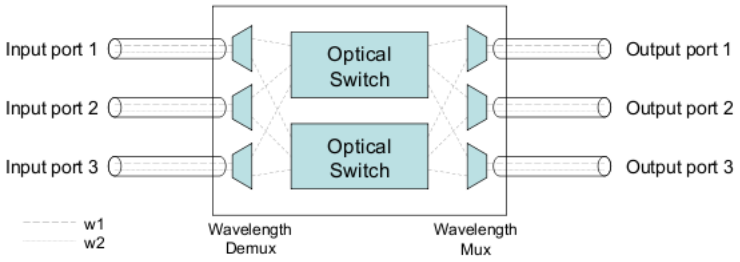

Figura 1.1: OXC (optical cross-connect) de 3 x 3 amb dos longitud d'ona per fibra

Amb aquesta tecnologia, més poden ser multiplexats més de 160 canals en una sola fibra, cadascun d'ells operant a 40 gigabits per segon o més. Ha rebut el nom de *dense WDM* (DWDM) per la seva capacitat de transportar un gran número de canals per fibra.

El servei que una xarxa WDM ofereix a un client, és una connexió lògica entre un origen i un destí, realitzada mitjançant un camí òptic, un *lightpath*. Els *lightpaths* poden ser establerts estàticament abans que la xarxa comenci a funcionar, o dinàmicament, sota demanda. En qualsevol cas, aquest fet dóna molta flexibilitat alhora de dissenyar, fer funcionar i mantenir xarxes. De fet, els *lightpaths* fan possible abstreure la topologia física, ja que per definició es permet que hi hagi múltiples topologies lògiques i concurrents a una mateixa xarxa física. La figura 1.2 mostra com es pot crear una xarxa lògica per sobre d'una xarxa física.

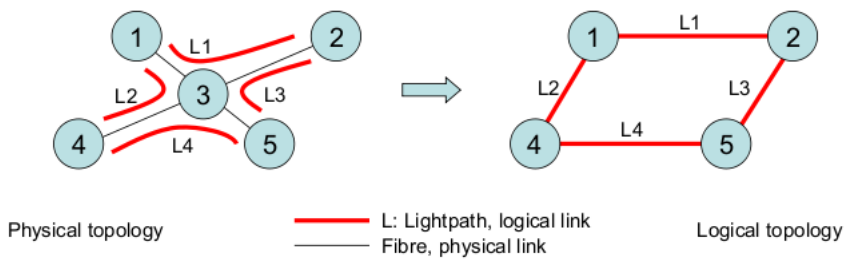

Figura 1.2: Exemple d'una xarxa lògica definida per sobre d'una xarxa física

1.2.3 Xarxes òptiques de transport

Una xarxa òptica de transport (OTN) ha estat definida per l'ITU-T com una xarxa *composta per un conjunt d'elements de xarxes òptiques connectats per enllaços de fibra òptica, amb la capacitat de proveïr transport, multiplexació, encaminament, administració, supervisió i survivability de canals òptics tranportant senyals de clients* [2] [3]. La visió que es té sobre una OTN és que forma part de les xarxes autòmatiques i dinàmiques del futur.

1.2.4 Xarxes multicapa

Les xarxes de transport normalment tenen un disseny basat en capes i rarament es composen d'una sola tecnologia. Basar el disseny en capes és un mètode per reduir la complexitat, pero també és el producte dels esforços que es fan per part de la comunitat científica per tal de preservar la investigació i el desenvolupament de noves tecnologies. Un possible i típic escenari es el mostrat a la figura 1.3

El *Generalized Multiprotocol Label Switching* (GMPLS) ofereix un marc unificat d'administració d'una xarxa pero totes les capes de commutació d'una xarxa. GMPLS utilitza un pla de control unificat i abstrau les diferències existents entre les diferents capes de medi de transmissió, estratègies de multiplexació, provisionament de camins, mecanismes de protecció i restauració, etc. A la següent secció es troba més informació d'aquesta tecnologia.

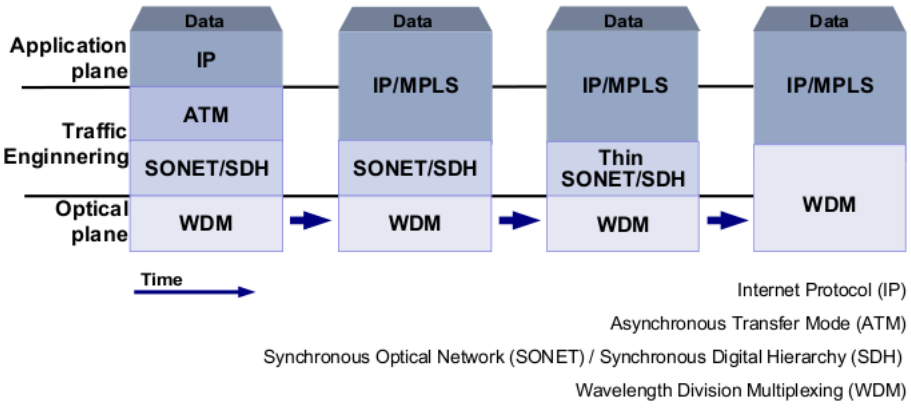

Figura 1.3: Representació d'una xarxa multicapa

1.3 G/MPLS

Multiprotocol Label Switching (MPLS) té dos objectius principals: accelerar l'enviament de paquets, i proveïr d'enginyeria del tràfic a xarxes IP. Per tal d'aconseguir un enviament més ràpid MPLS utilitza un escenari d'intercanvi d'etiquetes en comptes d'adreces per tal de determinar el següent salt per un missatge rebut. Amb l'objectiu de proveir d'enginyeria del tràfic les xarxes IP treballen com a xarxes orientades a la connexió. MPLS separa les capes de control (senyalització i enrutament) i de dades(enviament). Generalized MPLS (GMPLS) extén els components on l'operació d'intercanvi pot ser duta a terme a un sol pla de control.

Multiprotocol Label Switching va sorgir de l'evolució dels protocols d'encaminament i de reenviament(forwarding). MPLS proposa una solució que integra el control de la capa 3 d'enrutament amb la simplicitat de la capa 2 de switching. Bàsicament MPLS proveeix la separació dels components de control i de fordwarding i l'algoritme de forwarding d'intercanvi d'etiquetes [4].

El pla de control té dues funcions principals: descobriment del camí (enrutament), el qual implica crear les taules d'encaminament, i la funció de senyalització (per senyalitzar un camí enrutat). El protocol d'enrutament intercanvia informació amb altres routers per tal de construir i mantenir una taula d'encaminament actualitzada, utilitzant protocols estàndard de la capa 3, tals com Open Shortest Path First (OSPF) [6], Intermediate system to Intermediate System (IS-IS) [5], o Border Gateway Protocol (BGP). La taula de forwarding es manté a través del motor de control el qual és distribuït a tots els nodes de la xarxa mitjançant un protocol de

senyalització, com el Resource Reservation Protocol (RSVP) [7] i [8] o Label Distribution Protocol (LDP) [9].

El component de forwarding està basat en l'algoritme de forwarding d'intercanvi d'etiquetes (el mateix algoritme utilitzat per reenviar paquets als switchs d'ATM i Frame Relay). Els protocols de senyalització i la distribució d'etiquetes permeten la creació de Label Switch Paths (LSP) similars als Asyncronous Transfer Mode (ATM) Virtual Paths i als Virtual Circuits (VPI/VCI).

Una característica important de MPLS és la seva habilitat de fer passar el tràfic IP per un camí definit a través de la xarxa. Per cada servei específic, una taula de Forwarding Equivalence Class (FEC) és creada per representar un grup de fluxos amb els mateixos requeriments d'enginyeria del tràfic.

Aquest és un element clau que fa que MPLS sigui tan útil. Les FECs permeten a MPLS fer agrupació de fluxos, assignant una sola etiqueta a diferents fluxos amb la mateixa FEC. L'agrupació de fluxos redueix el nombre d'etiquetes necesitat per tal de manipular un determinat grup de paquets, i també redueix la quantitat de control de tràfic necessitat per la distribució d'etiquetes. Aquest fet millora l'escalabilitat i redueix el temps de procés.

Al node d'ingrés d'una xarxa MPLS els paquets que provenent d'IP són examinats. Aquest node, anomenat Label Edge Router (LER), els assigna una etiqueta i els paquets etiquetats són reenviats al llarg del camí, anomenat Label Switch Path (LSP), a on cada Label Switch Router (LSR) fa una decisió de commutació basada en l'etiqueta del paquet. La Label Information Base (LIB) proveeix una etiqueta i una interfície de sortida. La següent figura mostra una arquitectura MPLS:

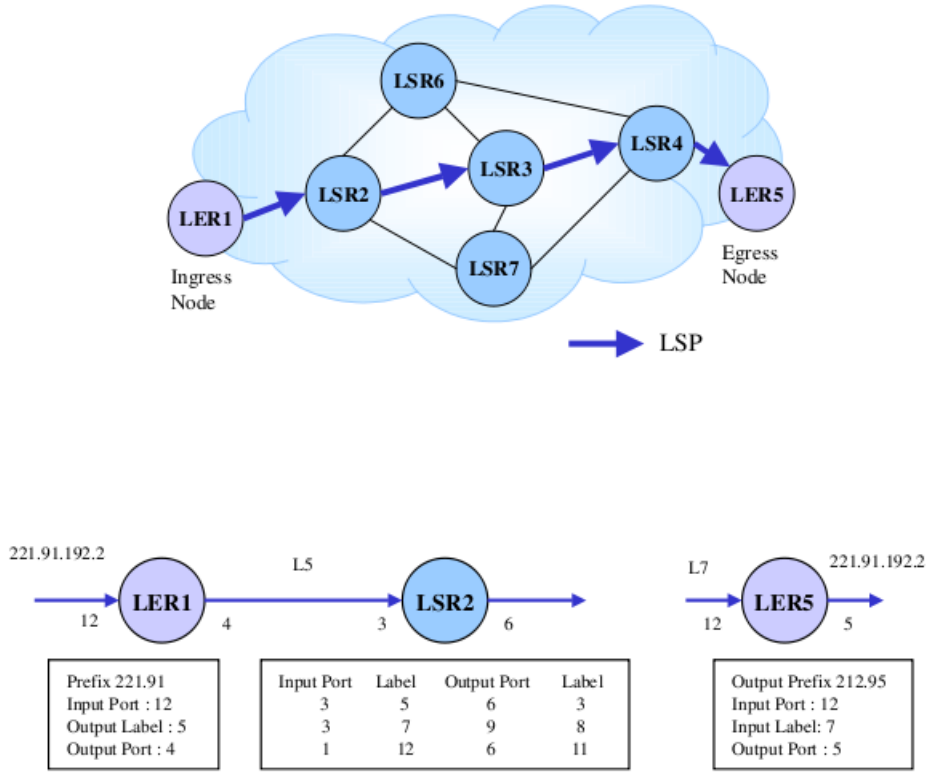

Figura 1.4: Arquitectura MPLS

Un cop es selecciona el LSP i els protocols de senyalització han assignat les etiquetes, aquest LSP pot ésser utilitzat. Quan un paquet IP arriba, el node d'ingrés (LER1) analitza l'adreça IP i el port d'entrada i assigna una etiqueta de sortida 5. El Label Switch Router 2 de la figura 1.4, nomes utilitza el LIB (Label Information Base) per executar el process de reenviament. Un paquet que arriba amb una etiqueta 5 utilitzant un port d'entrada 3 és reenviat al port de sortida 6 amb etiqueta 3. Al node de sortida el LER recupera la direcció IP.

La figura 1.5 mostra la capçalera de MPLS a on el camp per l'etiqueta (20 bits) conté el valor actual de la capçalera MPLS. L'EXP (camp experimental) (3bits) es per a provisionament de QoS (Quality of Service). El camp Time To Live (TTL) (8 bits) proveeix la funcionalitat convecional de TTL a IP. El camp S (stack) (1bit) suporta una pila jerarqueritzada d'etiquetes, que es una seqüencia d'etiquetes del paquet organitzada com una pila LIFO (Last In First Out). La pila d'etiquetes permet als paquests contenir informació de més d'una FEC, i també permet que

aquests viatgin a través de diferents dominis MPLS o segments LSP· Cal remarcar que el processament d'etiquetes és sempre basat en la que està més amunt. Aquesta propietat és essencial a MPLS degut al context de les xarxes òptiques perquè el nombre de longituds d'ona disponibles no és gaire elevat.

Figura 1.5: Capçalera MPLS

Extenent OSPF i IS-IS es permet als nodes intercanviar informació sobre la topologia de la xarxa, disponibilitat de recursos, etc. Aquesta informació és utilitzada pel protocol d'encaminament, en combinació amb el RSVP-TE, per calcular i crear camins subjectats a uns determinats recursos i/o restriccions polítiques. Aquest fet permet a MPLS dur a terme, d'una forma adequada, dos propòsits principals: enginyeria del tràfic i un ràpid re-encaminament.

1.3.1 Generalized MPLS

El Internet Engineering Task Force (IETF) ha extès el protocol MPLS per tal de que inclogui altres mecanismes de commutació mitjançant Generalized MPLS (GMPLS) (A la taula 1.1 trobem un resum d'aquestes tecnologies). El desenvolupament de GMPLS requereix modificacions als protocols de senyalització i d'encaminament actuals. Nous protocols com el Link Management Protocol (LMP), han sigut també implementats per manejar i mantenir la funcionalitat dels plans de controls i de dades entre dos nodes veïns. Els esforços actuals d'estandardització de l'IETF sobre GMPLS són breument mencionats seguidament:

- Un nou Link Management Procotol (LMP) disenyat per problemes d'adreces relacionats amb l'administració dels enllaços a xarxes òptiques.
- Millores als protocols d'encaminament OSPF i IS-IS per tal d'advertir de la disponibilitat de recursos òptics de la xarxa (p.e. banda d'ampla)
- Millores als protocols de senyalització RSVP-TE i CR-LDP amb finalitats d'enginyeria del tràfic que permetin als LSPs ser explícitament especificats a través de la part interior d'una xarxa òptica.

Domini de commutació	**Tipus de tràfic**	**Escenari de reenviament**	**Exemple d'un component**	**Nomenclatura**
Paquets o cel·les	IP o ATM	Etiqueta o connexió virtual de canals (VCC)	Router IP o un switch ATM	Packet switch capable (PSC)
Temps	TDM/SONET	Ranura de temps al repetir un cicle	Digital cross-connect systems (DCS)	TDM switch capable (TSC)
Longitud d'ona	Transparent	Lambda	Dense wavelength-division multiplexing (DWDM)	Lamda switch capable (LSC)
Espai físic	Transparent	Fibra	Optical cross-connect (OXC)	Fiber switch capable (FSC)

Taula 1.1: Tecnologies de commutació de GMPLS

- Millores en l'escalabilitat, com per exemple amb una construcció de LSP jeràrquics.

Hi ha diversos aspectes a tenir en compte si s'utilitza MPLS per controlar una xarxa. Per exemple, l'espai d'etiquetes de MPLS, és comparativament gran (un milió per port), mentres que hi ha un número determinat de lambdas i canals TDM (Time Division Multiplexing). Les xarxes òptiques utilitzaran centenars de fibres paral·leles, cadascuna contenint centenars de lambdas entre un parell d'elements d'una xarxa. Això significa que el número total d'enllaços a xarxes òptiques/TDM pot ser forces ordres de magnitud superior al número d'una xarxa MPLS.

1.4 Enrutament

Els algoritmes d'encaminament poden ser classificats segons diferents característiques:

- **Estàtics o no adaptatius:** Les rutes es calculen d'avantmà i són carregades als nodes durant la seva inicialització i permaneixen invariants al llarg del temps.
- **Dinàmics o adaptatius:** Canvien les seves decisions d'encaminament per a reflexar canvis a la topologia i/o al tràfic. Poden diferir en els instants d'adaptació i en la forma d'obtenir la informació i prendre decisions.
- **Aïllats:** Els nodes basen les seves decisions d'encaminament en informació obtinguda localment.
- **Centralitzats:** Un node de control utilitza la informació obtinguda de tots els nodes de la xarxa i pren les decisions d'encaminament, que posteriorment transmet a la resta de nodes de la xarxa.
- **Distribuïts:** Les decisions d'encaminament es prenen localment en els nodes i es basen en informació que s'obté d'una part (només adjacents) o de la totalitat de la resta de nodes.

A les xarxes actuals l'encaminament és **dinàmic** i **distribuït**.

Els protocols d'enrutament poden ser dividits en dos classes: protocols de gateway interiors (IGP) o protocols de gateway exteriors (EGP). Els primers s'utilitzen a dins de sistemes autònoms (una xarxa de routers sota una administració comuna, com per exemple una xarxa corporativa o una xarxa d'un districte escolar). L'encaminament entre diferents sistemes autònoms es duu a terme mitjançant EGP, dels quals els protocols de gateways frontaner (BGP) en són un exemple. Mentre que els IGPs estan dissenyats per mantenir la informació de la topologia de la xarxa dinàmicament, produir rutes òptimes i respondre a canvis ràpidament, EGPs donen més èmfasi a l'estabilitat i al control administratiu.

Els protocols d'encaminament interior també poden classificar-se de la següent manera:

1. **Vector distància:** Determina la direcció (vector) i la distància des d'un punt fins a qualsevol enllaç de la xarxa. Un exemple d'aquest grup seria el Routing Information Protocol (RIP).

2. **Estat d'enllaç (Link State):** Propaga informació de la topologia de la xarxa a tots els routers. Un exemple d'aquest grup seria el Open Shortest Path First (OSPF).

3. **Híbrid:** Combina aspectes dels algoritmes de Link State i de vector distància.

Aquests tres tipus es distingeixen en les mètriques que utilitzen per a seleccionar les rutes i en la forma d'emmagatzemar i intercanviar les actualitzacions de la taula d'encaminament.

1.4.1 Routing Information Protocol (RIP)

El Routing Information Protocol és un protocol de vector distància per a xarxes petites. Cada router habilitat amb RIP és classificat segons si és actiu o passiu: els actius adverteixen als demés routers de les seves rutes i els passius escolten i van actualitzant la seva taula d'encaminament a partir dels avisos que van rebent. Normalment els routers executen RIP en mode actiu i els hosts en mode passiu.

1.4.2 Open Shortest Path First (OSPF)

Open Shortest Path First és un protocol d'encaminament jeràrquic IGP, que utilitza l'algoritme Dijkstra Link State per a calcular camins el més curts possibles. Es capaç de construir una base de dades Link State (LSDB) idèntica a tots els routers de la zona.

OSPF és probablement el tipus de protocol més utilitzat a les xarxes grans. Pot funcionar amb seguretat utilitzant MD5.

Un exemple de funcionament senzill de OSPF seria el mostrat a la següent figura 1.6:

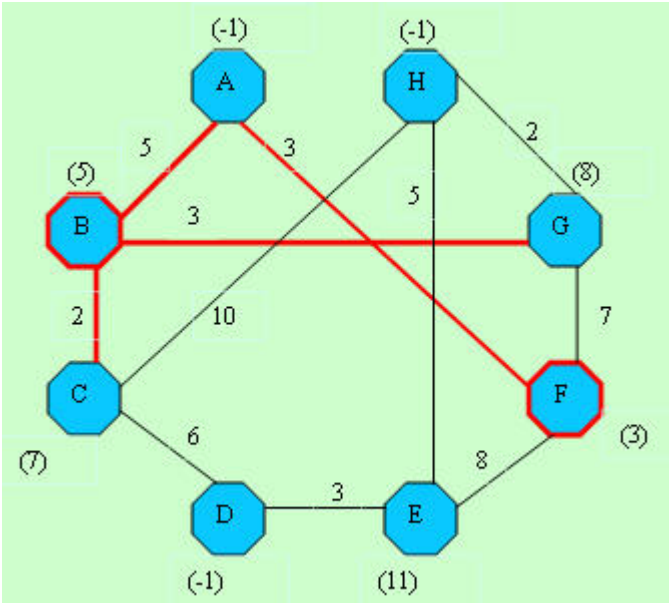

Figura 1.6: Exemple d'OSPF

1.5 Protecció i Restauració

En aquesta secció es presenten els mecanismes de recuperació de fallades que utilitzen G/MPLS.

Un domini de protecció a G/MPLS es defineix com el cojunt de Label Switch Routers (LSRs) pel qual són encaminats un *working path* i el seu corresponent camí de protecció. El domini de protecció per tant, ens indica un *working path* i un *backup path*.

La figura 1.7 mostra un simple domini de protecció de MPLS, format per un Working Label Switch Path, el qual és protegit, i un Backup Label Switch Path (camí de recuperació) per on el tràfic és redirigit un cop s'ha detectat una fallada. Els components Protection Switch LSR (PSL) i Protection Merge LSR (PML) són dos Label Switch Routers amb la funció de protecció. Tots els components que intervenen en el control de fallades de MPLS són definits a continuació:

- *Working o Active Label Switch Path:* Un LSP establert per transportar tràfic des d'un origen LSR a un destí LSR sota condicions normals. En altres pa-

raules, un working LSP és un LSP que conté fluxos de tràfic que requereixen protecció. La porció d'un working LSP que necessita protecció s'anomena segment protegit.

- *Protection or Backup Label Switch Path:* Un LSP establert per transportar el tràfic d'un working path seguint una fallada al working path. Un LSP de protecció ha de protegir tant un segment del working LSP com tot l'LSP sencer.
- *Label Switch Router (LSR):* El terme en MPLS, Label Switch Router (LSR) és utilitzat en aquest document per descriure un node de commutació de circuits com per exemple un optical cross-connect (OXCs) a les xarxes òptiques.
- *Protection Switch LSR (PSL):* Un PSL és un LSR a on s'originen tant el working path com el backup path. Fins que no es té coneixement d'una fallada, el PSL commuta el tràfic protegit des del working path al seu corresponent backup path.
- *Protection Merge LSR (PML):* El PML és l'LSR que finalitza tant el working path com el backup path i ajunta els dos fluxos de tràfic en un sol LSP, o si ell mateix és el node de destí, passa el tràfic a protocols de capes superiors.
- *Selector and Bridge nodes:* A les xarxes GMPLS hi ha una terminologia específica per referir-se als nodes PSL i PML. En aquest cas s'anomenen Bridge i Selector nodes, respectivament.

1.5.1 Classificació de models de recuperació de fallades

A la literatura actual han sigut presentants diferents propostes per a classificar els models de recuperació de fallades. En aquesta secció és presenten 2 models de classificació. El primer d'ells es basa en el nombre de working i backup paths i el segon model es basa en el fet del moment de temps en el qual els backups són calculats.

El model M:N

Probablement el model més conegut de classificació de mètodes de recuperació de fallades sigui el M:N. Està basat en el nombre de backup i working paths protegits. En aquest model M és el nombre de backups LSPs utilitzats per protegir N, els working LSPs. Utilitzant aquest model els possibles models de protecció podrien ser:

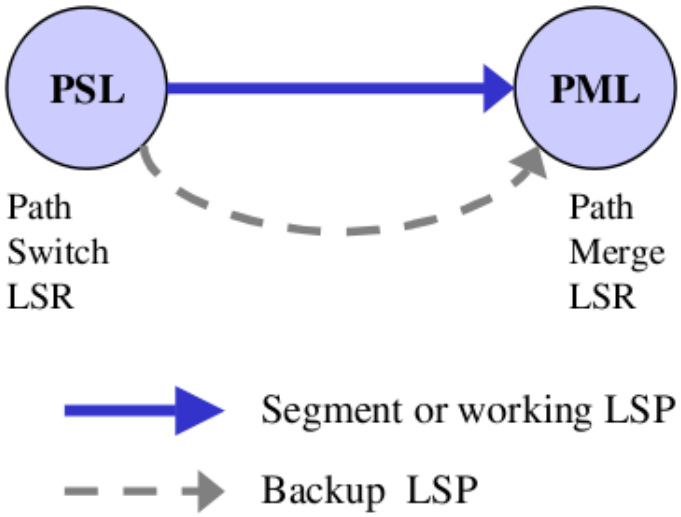

Figura 1.7: Components d'un domini de protecció a G/MPLS

- 1:1: 1 working LSP és protegit/restaurat per un sol backup LSP.
- M:1: 1 working LSP és protegit/restaurat per M backup LSPs.
- 1:N: 1 backup LSP és utilitzat per protegir/restaurar N working LSPs (es comparteixen els backups).
- N:M: N working LSPs són restaurats per M backup LSPs.
- 1:0: No hi ha protecció.
- 1+1: El tràfic és enviat tant pel working LSP com pel backup LSP.

Model de provisionament de camins i d'assignament de recursos

Aquesta manera de classificar els mètodes de recuperació de fallades es basa en el fet del moment de temps en el qual els backups són calculats i l'assignació de recursos.

El provisionament de camins pot ser categoritzat, com es mostra a la figura 1.8, d'acord amb el temps en el qual el backup path és calculat o quan el path és establert i quan els recursos són assignats a aquell path.

Hi ha diferents tipus de mecanismes mencionats a la literatura per reservar recursos. En tots els casos el nivell de reserva de recursos, com es mostra a la figura 1.9, pot ser classificat com a dedicat (com un 1:1, 1+1), compartit (1:N, M:N) o sense reserva (1:0), en el qual només és possible recuperar-se d'una fallada quan els recursos tornen a estar disponibles.

A la restauració compartida és important esmentar que suporta tràfic amb dos tipus de prioritats (baixa i alta).

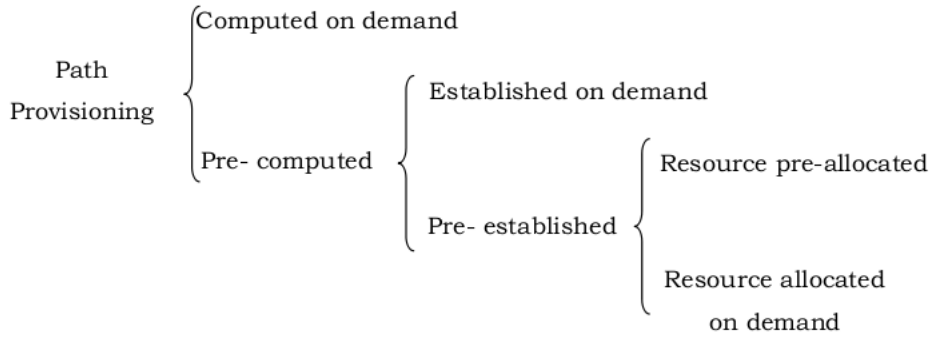

Figura 1.8: Classificació de provisionament de camins

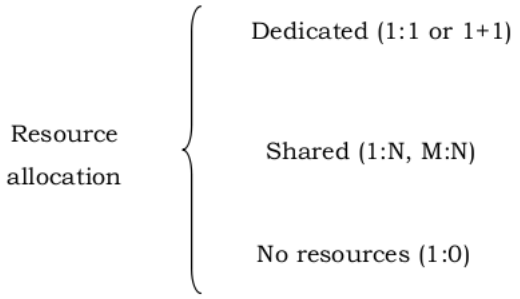

Figura 1.9: Classificació de d'assignació de recursos

En el cas de pre-allocated backup paths, existeix la pregunta de què fer amb les recursos que tenen els backup paths. D'acord amb [11], hi ha dos opcions:

- *Dedicated-resources:* Si els recursos d'un backup path són dedicats, no poden ser utilitzats per a res més excepte transportar tràfic del working path, com per

exemple en el cas de protecció 1+1. Encara que el back up path no transporti tràfic del working path, no és possible utilitzar per transportar altres tipus de tràfic.

- *Extra-traffic-allowed:* Si el backup path només transporta el tràfic del working path quan aquest falla, mentrestant és possible permetre el transport de tràfic extra. Per definició, tràfic extra és aquell tipus de tràfic que pot ser desplaçat en qualsevol moment en que els recursos per on passa siguin necessàris per transportar tràfic del working path.

1.5.2 Models de recuperació de fallades de G/MPLS

En aquest apartat es presenten els principals models de recuperació de fallades. Aquests models primerament van ser definits per MPLS a [11]. Més tard, alguns d'aquests models van ser adaptats a xarxes òptiques (a través del pla de control de GMPLS).

El model global

En aquest model [12], el node d'ingrés agafa la responsabilitat de la recuperació de fallada quan arriba una senyal que indica que s'ha produït una fallada. Aquest mètode requereix l'establiment de backups disjunts per cadascun dels working paths actius.

En aquest model la protecció sempre s'activa al node d'ingrés, independement d'on es produeixi una fallada al llarg de tot el working path. Això implica propagar la informació de que s'ha produit una fallada tot el camí enrere fins el node origen.

El model revertit

La principal idea d'aquest mètode és de revertir el tràfic al punt on s'ha produït una fallada del LSP protegit cap al node d'ingrés mitjançant un Reverse Backup LSP.

El model local

Amb aquest model de recuperació de fallades, la recuperació comença des del punt de la fallada. És un mètode local i transparent al node d'ingrés. L'avantatge principal

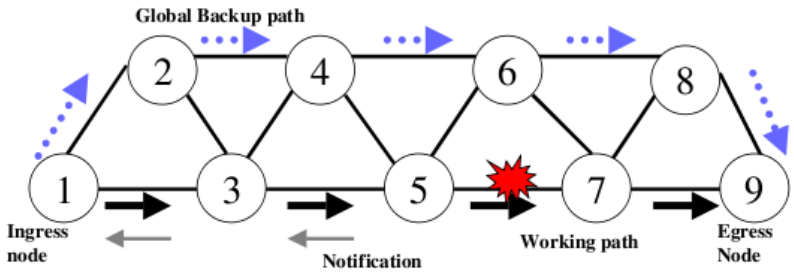

Figura 1.10: Model de protecció global

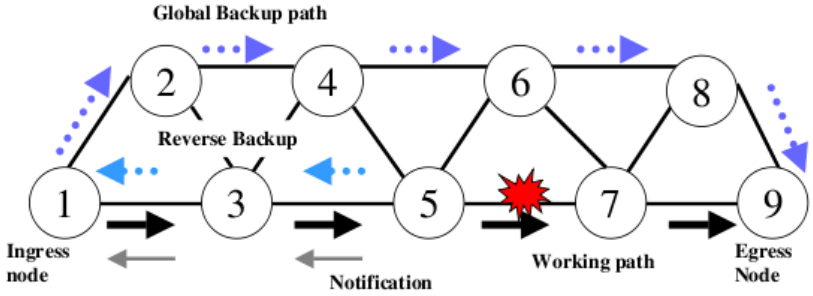

Figura 1.11: Model de protecció revertit

d'aquest model és que ofereix un temps de recuperació més petit que el model global i que evita que es perdin paquets.

1.5.3 El model de protecció 1+1

Aquest model de restauració de fallades utilitza dos working paths. En aquest cas, el PML/Selector LSR està supervisant el millor working path (per exemple, el que tingui millor senyal). Després d'una fallada, el PML/Selector detecta que només hi ha un path i selecciona aquest path com el seu nou working path. Aquest mètode és ràpid, no té cap pèrdua de paquets i elimina el temps de notificació de fallades, amb l'inconvenient de que consumeix molts recursos ja que tots dos paths han d'estar sempre reservats. A més a més, un PSL/Bridge LSR també ha de ser configurat perquè sigui capaç d'enviar tràfic per dos camins.

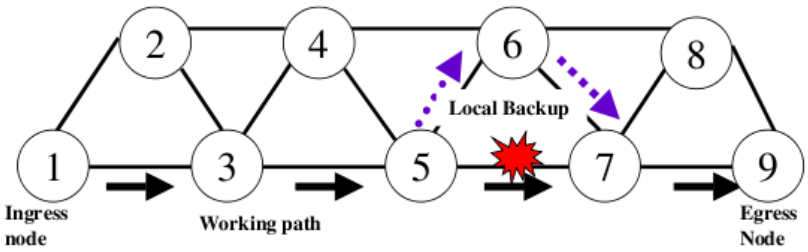

Figura 1.12: Model de protecció local

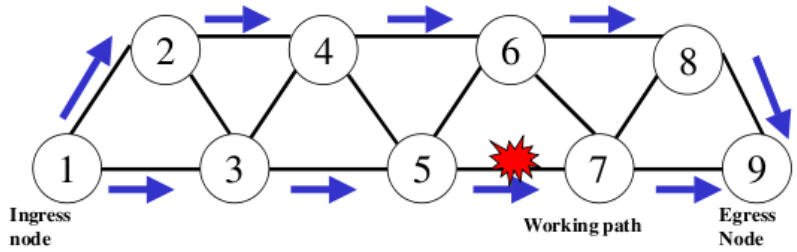

Figura 1.13: Model de protecció 1+1

1.6 Propagació de fallades

Els virus de computadors avui en dia encara formen part d'un tema important de les xarxes i dels sistemes. Els mecanismes de defensa existents normalment es basent en l'exploració de signatures de virus. Mentre que aquests mecanismes poden prevenir i detectar la propagació de virus coneguts, han demostrat no ser de gran ajuda alhora de dissenyar sistemes de defensa globals òptims. La recent proliferació d'atacs de negació del servei (DDoS) en conjunt amb la propagació vírica agreuja el problema. La magnitud d'una propagació vírica significa que els atacs DDoS que van agafats a la cua dels virus i cucs poden aparèixer en una escala sense precedents i són, així doncs, especialment perjudicials. Amb l'excepció d'un parell d'estudis de model especialitzats, la majoria encara continua desconeixent les característiques de propagació de virus de computadors i els factors que els influencien. La pregunta clau en tots aquests casos és la següent: *El virus/cuc viurà per sempre al llarg de tota la xarxa, o morirà?*

En aquest marc, és interessant saber quines respotes i han de donar les tèniques de modelatge per aquest tipus de sistemes, més concretament respostes a:

- *Com es propaga un virus a través de la xarxa?* Per a contestar aquesta pregunta es necessita un model anàlitic general de propagació vírica, el qual pot conèixer l'impacte sobre qualsevol topologia subjacent sense estar limitat per qualsevol tipus d'assumpcions d'aquesta.

- *Existeix un llindar d'epidèmia, i si existeix, com es pot trobar?* El llindar d'epidèmia és una condició que enllaça les característiques dels virus amb les de la topologia. Així doncs, si aquesta condició es satisfà, la infecció viral morirà al llarg del temps.

- *Si la condició del llindar d'epidèmia és satisfeta, amb quina velocitat la xarxa acabarà desinfectant-se?*

- *Quin node és preferible immunitzar?* Hauria de ser el node amb el grau nodal més elevat? O el que té una puntuació més gran segons PageRank?

Al capítul 3 d'aquest document s'aprofundeix més en aquest tema, i es proposen algunes respostes a aquestes preguntes.

Capítol 2

Conceptes i tecnologies de simuladors

2.1 Introducció

D'acord amb Shannon [14], la simulació digital per ordinador és el procés de dissenyar un model d'un sistema real i dur a terme experiments amb aquest model en un ordinador digital amb el propòsit específic d'experimentar. Si ens basem en la taxonomia donada a [15], la simulació digital per ordinador es pot dividir en tres categories: (1) Monte Carlo, (2) continua i (3) d'esdeveniments discrets. La simulació de Monte Carlo és un mètode al qual un problema inherentment no probabilístic es soluciona mitjançant un procés estocàstic; la representació del temps no és explícita. A una simulació continua, les variables són funcions continues (per exemple un sistema d'equacions diferencials). Si el valor de les variables canvia a instants precisos de temps, la simulació és d'esdeveniments discrets.

Com ja s'ha comentat, una simulació implica la modelització d'un sistema. Un sistema és una part del món que nosaltres triem per considerar-la com un tot, separada de la resta del món per varies consideracions, un tot que triem perquè conté una sèrie de components, cadascun d'ells caracteritzat per un conjunt seleccionat de dades i models, i per accions que poden implicar ell mateix (un component) amb altres components. El sistema pot ser real o imaginari i ha de rebre entrades i/o produir sortides per el seu entorn.

Un model és una abstracció d'un sistema que intenta replicar algunes propietats d'aquell sistema. La col·lecció de propietas que un model intenta replicar (amb

el propòsit de donar respostes a determinades preguntes sobre el sistema) han d'incloure l'objectiu de modelatge. La importància de l'objectiu de modelatge no pot ser exagerada; una formulació correcte és l'objectiu essencial per a qualsevol estudi amb simulació satisfactori. Només a través de l'objectiu és possible assignar un significat a qualsevol simulació donada. Partint que per definició un model és una abstracció, hi ha detalls que existeixen al sistema però que no tenen representació al model. Per tal de justificar el nivell d'abstracció, les suposicions del model han d'anar pel mateix camí que el seu objectiu.

Segons Nance [16], un model està composat per objectes i les relacions que hi ha entre ells. Un objecte pot ser qualsevol cosa caracteritzada per un o més atributs els quals tenen uns valors assignats. Els valors assignats als atributs han d'ajustar la manera de representar-se als llenguatges convencionals d'alt nivell.

Hi ha dos conceptes que són de vital importància a la simulació d'esdeveniments discrets, aquests són el *temps* i l'*estat*. La següent classificació ens mostra conceptes fonamentals per tal de descriure'ls:

- Un instant és un valor de temps del sistema a on el valor de com a mínim un atribut d'un objecte pot ser alterat.
- Un interval és la duració que hi ha entre dos instants successius.
- Un **span** és la successió contigua d'un o més intervals.
- L'estat d'un objecte és la enumeració de tots els valors dels atributs de l'objecte en un instant determinat.

Aquestes definicions són la base dels següents conceptes, els quals són àmpliament utilitzats:

- Una activitat és l'estat d'un objecte durant un interval.
- Un event és un canvi a l'estat d'un objecte, i que té lloc a un instant. Un event es diu que és determinat si la única condició per a que l'event tingui lloc pot ser expressada com una funció de temps. Altrament, és diu que és contingent.
- L'activitat d'un objecte és l'estat d'un objecte entre dos events describint els successius canvis d'estat de l'objecte.
- Un procés són tots els estats successius que pot tenir un objecte al llarg d'un **span**.

Aquests conceptes poden ser observats a la figura:

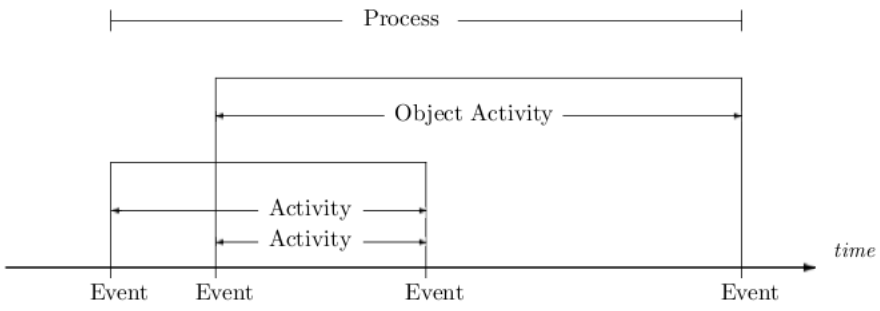

Figura 2.1: I·lustració d'un event, d'una activitat i d'un procés

Finalment, l'activitat i el procés formen la base dels següents tres conceptes d'una simulació d'esdeveniments discrets:

- A una visió global de la programació d'events, el modelador identifica quan han de tenir lloc les accions al model.
- A una visió global de l'exploració d'activitats, el modelador identifica per què les accions tenen lloc al model.
- A una visió global de la interacció de processos, el modelador identifica els components d'un model i descriu la seqüència d'accions entre cadascun d'ells.

2.2 Principals simuladors de xarxes

Els simuladors s'han vingut utilitzant des de fa temps en l'àmbit de la recerca i ocupen un lloc privilegiat com a eina per a estudiar protocols, problemes d'allotjament de recursos, aplicacions i, en general, sistemes complexos el comportament dels quals ha de ser estimat o verificat. Els simuladors són útils per a treballar amb tecnologies que encara no han sigut construides, però també quan els estudis que es duen a terme impliquen xarxes que van més enllà del que es pot arribar a disposar amb els test beds, ja sigui pel tamany, pel cost econòmic o per altres raons tècniques.

Com que els simuladors han de cumplir amb la missió de reproduir aspectes rellevants de la realitat, han de ser capaços d'enfrontar-se amb els avenços i les

transformacions que les xarxes duen a terme al llarg del temps.

Normalment un simulador és una peça complexa de software, i en varis aspectes semblant a una eina de desenvolupament de software i, com que n'hi ha forces, escollir l'adequat per a un projecte pot ser la primera decisió difícil de prendre. El criteri per decidir quin utilitzar pot variar segons els següents aspectes: suport d'abstracció d'un model, el llenguatge de programació, entorn d'execució i grau de dinamisme (si els programes són interpretats o compilats), integració amb altres eines (amb altres simuladors), disponibilitat d'un bon suport a l'usuari i una bona documentació, maduresa de la implementació, disponibilitat del codi font o arquitectura. Alguns d'aquests aspectes poden entrar en conflicte amb d'altres, per tant finalment s'ha de tenir en compte prioritats.

Seguidament s'analitzen diversos simuladors de xarxes open-source basant-se en els següents aspectes:

- escalabilitat,
- suport a una bona enginyeria del software, i
- suport al tractament de les dades recollides.

2.2.1 OPNET

OPNET és un respectat simulador d'esdevenimets discrets i de propòsit general que disposa d'una interfície gràfica i mòduls per a tasques com el modelatge de xarxes, planejament i anàlisis. No és un simulador disponible gratuïtament i tampoc és open-source. Utilitza un modelatge orientat a objectes jerarqueritzat que intenta ajustar-se a l'estructura d'una xarxa real, equipament i protocols. Suporta xarxes inhalàmbriques. A [17] el producte és exposat amb la capacitat d'administrar diversos centenars de nodes, mentre que al fulletó del producte s'afirma que pot arribar fins a milers [18].

En les seves últimes versions és capaç de simular models en paral·lel, cosa que millora molt la seva escalabilitat. Com que es tracta d'un producte prou madur, disposa d'una extensa documentació, exemples, una llibreria de protocols i de dispositius de xarxa. L'usuari pot desenvolupar nous models combinant el llenguatge C/C++ amb diagràmes de transició d'estats.

2.2.2 NS-2

NS és extensament utilitzat a la comunitat de recerca en l'àmbit de xarxes. Va començar al 1989 com una variació d'un simulador que hi havia prèviament anomenat REAL. La seva versió actual s'anomena *ns-2* i és disponible a [19]. Disposa d'un suport substancial per a la simulació de transport (p.e. TCP) i protocols de sessió, algoritmes d'enrutament unicast i multicast, i diversos protocols a nivell d'aplicació i per a xarxes inhalàmbriques (ad hoc, local i satèlit). *ns-2* maneja topologies arbitràries, compostes per routers i enllaços.

Fidel a la idea de ser una eina unificada per a la comunitat de recerca de xarxes, inclou una gran varietat de contribucions d'usuaris. *ns-2* ha servit com a base per a moltes extensions tals com SensorSim per a xarxes de sensors i *pdns* per a simulacions paral·lelitzades i distribuides.

Generadors de xarxes tals com Tiers o GT-ITM poden ser emprats per a la producció autòmatica de topologies de xarxes arbitràries amb diverses propietats definides per l'usuari. Un animador de xarxes anomenat NAM fa possible el fet que es pugui visualitzar el comportament de la xarxa simulada amb la generació de dades mentre es duu a terme. *ns-2* disposa també d'una interfície d'emulació que permet la interacció amb el tràfic del món real, com la injecció de tràfic.

ns-2 utilitza C++ com el seu llenguatge d'implementació, mentre que la configuració (i la definició de l'escenari) es fa amb un derivat de Tcl anomenat OTcl, executat a un entorn interpret de commandes. A [17] l'autor comenta que l'esquema que duu a terme *ns-2* per a l'implementació de les seves simulacions (basat en scripts senzills), complica el fet d'afegir nous components, incrementa la corva d'aprenentatge i fa que el debug sigui molt complicat. A més a més la documentació acostuma a no ésser actualitzada respecte el software.

ns-2 pot arribar a manjear uns quants milers d'elements a una xarxa simulada, tot i que no és pràctic treballar amb xarxes gaire grans [20]; consumeix massa memòria i no és suficientment ràpid.

Respecte a la recolecta i l'afegiment de dades, *ns-2* ofereix dos mecanismes bàsics per a dur a terme aquestes funcions. El primer s'anomena *trace* i, simplement, permet fer logging a un arxiu o directament a la consola mentre la simulació s'està duent a terme. El segon s'anomena monitor i és capaç de desar informació dels esdeveniments, tals com l'arribada i la sortida de paquets, els paquets perduts,

el retràs experimentat, etc.

A la pràctica, l'usuari de *ns-2* generalment ha de preparar programes especials o scripts per a processar i resumir les dades obtingudes amb el simulador per tal de dur a terme un anàlisis exhaustiu.

2.2.3 OMNet++

Tal i com s'anuncia a la seva plana web [21], OMNet++ és un simulador d'esdeveniments discrets modern, basat en components, modular i de codi obert amb un fort suport de GUI i un kernel de simulació incrustat. La seva apliació principal és a l'àrea de la simulació de xarxes de comunicacions, tot i que és possible simular processos d'altres tipus d'arees. El seu llenguatge d'implementació es C++ i ofereix una llibreria de classes per dur a terme simulacions en aquest llenguatge. El producte és gratuït per un ús acadèmic, d'altres utilitzacions requereixen d'una llicència comercial.

El pas de missatges és el mecanisme de comunicació principal emprat pels components del simulador, la qual cosa simplifica executar simulacions en entorns paral·lelitzats o distribuïts. Comparat amb *ns-2* és una eina força jove (el seu desenvolupament va començar al voltant de 1998) i, per això, disposa de menys mòduls i protocols implementats.

A OMNet++, un model d'una xarxa consisteix en entitas aniuades jeràrquicament anomenades mòduls. Els mòduls es comuniquen mitjançant pas de missatges, on els missatges poden contenir qualsevol tipus d'informació. Els mòduls poden enviar els missatges directament a la destinació final o a través d'un camí, a través de portes i connexions, les quals poden tenir assignades propietats tals com ample de banda, retràs, o taxa d'error per bit. Els mòduls poden tenir paràmetres que poden servir per a personalitzar un comportament determinat del model, per a crear diferents topologies, i per comunicació entre mòduls com a variables compartides. L'usuari ha de proveïr el nivell més baix de la jerarquia de mòduls, el qual conté els algoritmes del model.

OMNet++ utilitza dos llenguatges, un per escriure el comportament del model i l'altre per controlar la simulació. El primer d'ells és C++ i el segon és anomenat NED. Els arxius escrits en NED descriuen la topologia de la xarxa.

OMNet++ utilitza una llibreria Tcl per tal de proveïr l'usuari d'una interfícia gràfica. Aquest simulador és elogiat per el seu disseny, fàcil d'utilitzar i amb flexibilitat, especialment si es compara amb *ns-2*. Aquesta evidència és raonable tenint en compte que és un producte molt més nou que no pas *ns-2*. S'espera una bona escalabilitat d'OMNet++ ja que suporta simulacions paral·leles i distribuides degut a que utilitza Message Passing Interface (MPI) per comunicar processos lògics (LP). En quant a les capacitats per recollir estadístiques OMNet++ és similar a *ns-2*, tot i que si es fa un cop d'ull a la documentació, es veu que en aquest àmbit està menys especialitzat.

2.2.4 GTNetS

GTNetS (Georgia Tech Network Simulator) va ser presentat al 2003 amb l'avantatge de permetre simulacions a gran escala que podien ser creades fàcilment amb les eines de les que es diposàven en aquella època. Va ser dissenyat per a suportar simulacions paral·leles i distribuides. També, és interessant saber que l'autor de GTNetS va començar a implementar aquest simulador després d'haver implementat el *pdns*, la distribució paral·lela de *ns*.

El simulador utilitza C++ com a llenguatge d'implementació i simulació, i per tant, la simulació ha de ser conduïda per un programa principal escrit en aquest mateix llenguatge. GTNetS proveeix d'un gran nombre de protocols implementats.

Degut a que és un simulador força nou, l'autor en persona adverteix que certes característiques poden ser immadures encara, o fins i tot perdudes. A [17], l'autor dóna resultats de simulacions de xarxes de més de 48000 nodes en un entorn distribuit que consisteix en 32 sistemes amb un total de 128 processadors. En el paper [24] es conclou que tan *pdns* com GTNetS tenen un rendiment més o menys similar. GTNetS disposa d'un gran nombre de funcions per tal d'assistir a l'usuari alhora de la recol·lecció d'estadístiques.

2.2.5 Shawn

Shawn és un nou simulador dissenyat per treballar amb grans xarxes de sensors inhalàmbrics, amb centenars i centenars de nodes. L'increment d'escalabilitat s'assoleix mitjançant la utilització de models abstractes.

Els autors de *Shawn* creuen convenient simular l'efecte causat per un fenòmen i no el fenòmen en sí mateix. Per exemple, en comptes de simular una capa completa de MAC incloent-hi el model de propagació per radio, els seus efectes són modelats (p.e. corrupció i pèrdua de paquets). D'aquesta manera les simulacions s'executen molt més ràpidament, però detalls sobre la funció i el comportament de la capa física o dels paquets són impossibles d'obtenir.

La seva implementació és en C++, el qual també pot fer-se servir per tal d'implementar els seus models. Com que l'escalabilitat és el seu atractiu més important, no és sorprenent que d'acord amb els seus autors, el seu rendiment és excel·lent. Una comparació amb *ns-2* es presenta a [25]. Per exemple, una xarxa amb 2000 nodes simulats a *ns-2* es requereixen més de 25 hores, mentre que a *Shawn* només es necessiten 19 segons. La comparació en el consum de memòria és impressionant també: *ns-2* utilitza més de 220MB de RAM, mentre que *Shawn* n'utilitza menys de 9. Respecte a la recol·lecció d'estadístiques no hi ha cap referència ni a [25] ni a la guia dels desenvolupadors.

2.3 OMNet++

El simulador OMNeT++ aporta un entorn de simulació d'esdeveniments discrets. Això significa que no només pot executar simulacions de xarxes d'ordinadors, encara que és el seu ús principal. Podem trobar simulacions de xarxes TCP/IP, protocols com Ethernet o Token Ring, dissenys de xarxes ad-hoc i infraestructures, etc.

Com ja s'ha esmentat abans, OMNeT++ fa servir una estructura modular. Els mòduls es programen en llenguatge C++, mentre que per unir-los entre sí i fer un model, es fa servir un llenguatge propi d'alt nivell: NED. L'avantatge de la programació modular és que es pot reutilitzar codi per a altres mòduls i/o projectes.

Els mòduls es classifiquen en simples i compostos. Els mòduls compostos són formats per conjunts de mòduls simples, o d'altres compostos. En els mòduls simples és on es programa el comportament en C++.

Una punt fort d'aquest simulador resideix en que es poden fer servir un potent entorn gràfic per mostrar les simulacions. És bastant intuïtiu i ens pot fer servir per saber com està la xarxa en qualsevol moment determinat. Per contra, les simulacions fetes amb entorn gràfic són més lentes que si les executem des de comandes.

Al finalitzar les simulacions, si s'ha definit un mòdul d'estadístiques, OMNeT++ s'haurà encarregar de guardar els valors de la simulació en un o varis fitxers. Amb les utilitats que porta, el simulador permet mostrar gràfiques molt completes dels resultats.

Finalment, cal destacar que OMNeT++ suporta execucions amb paral·lel. Per això, és necessari que estigui configurat les llibreries MPI. Els programes no s'han de programar per fer una execució paral·lela. OMNeT++ té tècniques que permeten que es pugui realitzar una paral·lelització.

Tal i com indica el seu autor András Varga, les principals característiques del programa són:

- Llibreria del nucli de simulació
- Compilador pel llenguatge NED (nedc).
- GUI (Interfície gràfica) per l'execució de la simulació, crea un executable (Tkenv)
- Interfície via comandes per l'execució de la simulació (Cmdenv)
- Eina gràfica de vectors de sortida (Plove)
- Eina gràfica per visualitzar resultats escalars (Scalars)
- Utilitats d'ajuda al desenvolupament:
  - Generador de llavors de nombres aleatoris (seedtool)
  - Creació de makefiles automàtic (opp_makemake)
  - Generació de documentació del model (opp_nedtool)
  - Editor gràfic de fitxers NED (gned)
  - Etcètera ...
- Documentació, exemples, etc.

La documentació que s'inclou amb el simulador és molt completa i molt útil, especialment el manual d'usuari [22], ja que repassa tots els aspectes del simulador, i és una gran eina a l'hora d'aprendre a programar amb ell.

A més a més, gràcies al gran suport que rep aquest simulador, existeix una llista de correu en la pàgina oficial en que es pot consultar tot tipus de preguntes i respostes, en les quals hi apareix el propi autor del programa.

Les utilitats del simulador són una gran eina de treball, i d'una qualitat molt alta. Realitzen la seva funció perfectament, integrant-se fàcilment amb l'entorn, i són senzilles de fer servir.

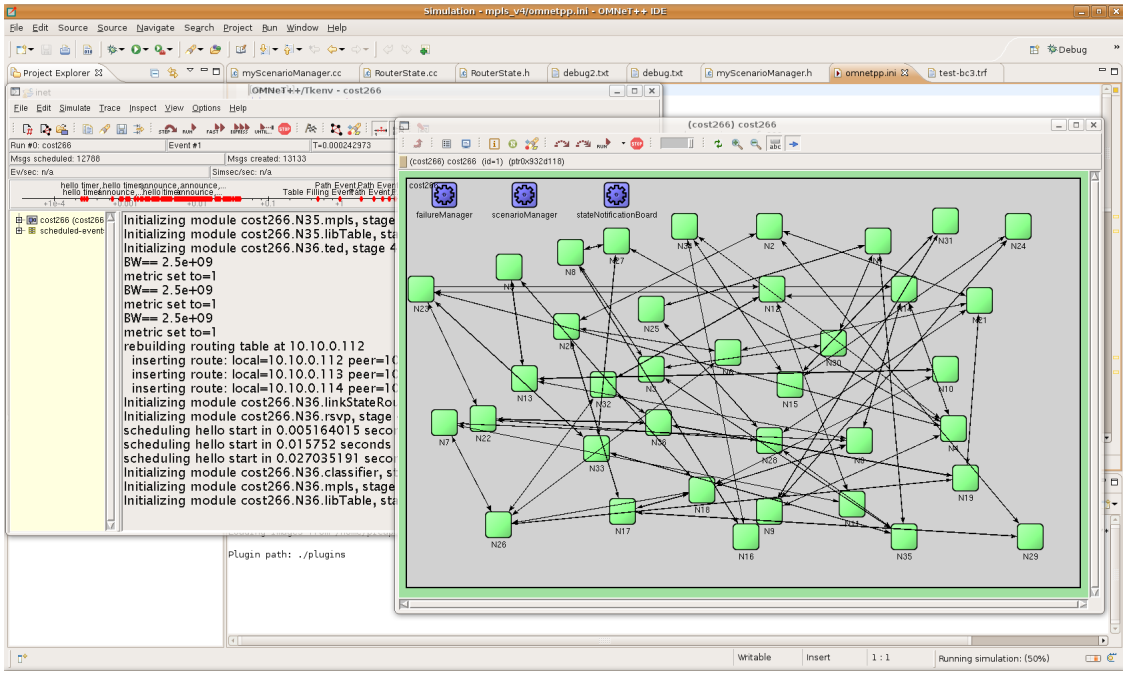

Figura 2.2: El simulador OMNeT++ en l'entorn Linux

2.3.1 Instal·lació

OMNeT++ és una aplicació que s'ha d'executar en un entorn UNIX. Segons el sistema operatiu en que estem treballant, triarem una versió o una altra:

- Per sistemes UNIX, ens serveix des de versions de GNU/Linux (Fedora, Ubuntu, etc...), Solaris, i també Mac Os X. En aquest cas ens haurem de baixar el codi font de l'aplicació i compilar-lo (és possible que ens demani varis paquets

per poder fer la instal·lació.). També es poden trobar versions compilades per alguna versió de Linux, encara que no estan mantenides pel creador

- Per versions Windows, hi ha dues alternatives. La primera és fer servir un entorn UNIX que funcioni sota Windows, en aquest cas es diu Cygwin. D'aquesta manera disposarem de les comandes típiques d'entorns UNIX en Windows. La segona manera és integrant-ho amb Microsoft Visual C++, encara que les ordres per compilar i executar seran un pèl diferents. En tots dos casos, s'ha de baixar la versió en binari del simulador, e instal·lar-la.

A l'hora de realitzar el projecte, hem escollit la versió UNIX per treballar. OMNeT++ té una llicència Acadèmica (i en el nostre cas el podem fer servir lliurement), i a més a més estem desenvolupant en un entorn de codi lliure i per tant no disposem de cap restricció en quant a llicències privatives.

La versió que hem fet servir ha estat la 4.0, i s'ha desenvolupat en un Ubuntu 8.04 - *Hardy Heron* -. En el moment del redactat d'aquest document, l'última versió estable del programa és la 4.0.

La instal·lació d'OMNeT++ ja s'ha documentat en altres projectes de final de carrera. En aquest cas, citaré una instal·lació en UNIX la qual ja va ser explicada en el projecte en que es va desenvolupar l'algorisme AntNet-QoS [23], però actualitzada.

Instal·lació pas a pas:

1. Descarga de la pàgina del simulador (http://www.omnetpp.org) el codi font de la versió desitjada (en el nostre cas, la versió 4.0).

2. Descompressió de l'arxiu en format tar.gz amb el codi font del simulador. Comanda: "tar zxvg omnetpp-4.0.tgz"

3. Moure la carpeta extreta al directori d'inici del nostre usuari (ex:/home/usuari)

4. Configuració de les variables: PATH, LD_LIBRARY_PATH i TCL_LIBRARY_PATH. Depenent de la versió de UNIX (Linux), ens caldrà modificar el fitxer ocult ".bash_profile" o ".bashrc" en l'intèrpret de comandes bash:

```
export PATH=$PATH:~/omnetpp-4.0/bin
export LD_LIBRARY=~/omnetpp-4.0/bin
export TCL_LIBRARY_PATH=~/omnetpp-4.0/lib
```

5. Ens situem al directori omnetpp-4.0

6. Per comprovar si podem compilar correctament, executem l'ordre: #./configure.

7. Per pantalla anirà comprovant si disposem dels paquets necessaris. Normalment, fan falta els paquets Tcl i Tk en versions binàries i de desenvolupament, i altres paquets per l'entorn gràfic com BLT,... El paquet MPI és opcional si volem executar simulacions paral·leles.

8. Un cop tinguem els paths correctes i no faltin paquets necessaris, el compilem amb: #make all (amb make simple pot donar errors de compilació).

9. Si tot ha funcionat correctament, ja tindrem el programa a punt per utilitzarlo. Per provar si funciona, dins del directori sample, hi ha un script ./rundemo que ens mostrarà exemples del programa.

2.3.2 Configuració i funcionament d'un petit example

En el punt anterior hem parlat de fitxers amb llenguatge NED i amb llenguatge C++. Els fitxers NED defineixen l'estructura de la xarxa (enllaços, portes d'entrades i de sortides, mentre que els fitxers C++ és el codi dels mòduls simples. Els dos tipus els podem escriure amb qualsevol editor de text.

Per poder compilar un programa, un cop disposem dels fitxers codi font, el primer que cal fer és crear un Makefile. Un makefile és un fitxer en que s'inclou totes les comandes necessàries per tal de poder compilar i enllaçar, obtenint l'executable de la nostra aplicació.

En UNIX, creem el nostre makefile amb la comanda:

```
#opp_makemake
```

Aquesta commanda admet diversos paràmetres. Els més útils són:

- -f: força a generar un nou makefile, encara que ja hi hagi algun.
- -h: ens ofereix totes les opcions
- -u: ens permet triar si volem executar en un entorn gràfic o comandes (defecte gràfic):

```
-u Tkenv / -u Cmdenv
```

- -o: podem definir un nom propi en l'executable.

Un cop generat el Makefile, podem començar amb compilar (UNIX):

```
#make
```

El primer que realitza és comprovar els fitxers NED amb un compilador propi que porta (ja que és un llenguatge creat per OMNeT++). Un cop compilats, llavors executa el compilador g++ (per defecte, encara que pot ser un altre) i compila els fitxers .cpp. Un cop finalitzat i si ha anat bé, ens haurà generat uns fitxers .o (fitxers objectes, en Windows són .obj). Només queda enllaçar-los i generarà el nostre executable.

Per executar el nostre programa, simplement la cridem pel seu nom. Si hem triat un entorn gràfic, ens engegarà la finestra principal, i altres finestres (dibuix de la xarxa,...).

Si ens cal netejar i eliminar l'executable i els fitxers objectes, la següent comanda realitza aquesta funció:

```
#make clean
```

Capítol 3

Modelització de la propagació de fallades a una xarxa

3.1 Introducció

El problema de la propagació de virus ha atret molt d'interés per part de la comunitat científica. Entre tots els models proposats per a propagacions virals, hi ha hagut dos que han tingut una acceptació força àmplia. El primer d'ells, anomenat el model SIS, considera que els individus poden romandre en dos estats diferents: susceptible (S) o infectat (I). Un individu susceptible pot infectar-se en contacte amb qualsevol altre individu infectat, i després curarse ell mateix amb una certa probabilitat per tal de ser susceptible una altra vegada. El segon model, anomenat SIR, és similar al primer amb la única diferència que un cop curat, un individu és considerat esborrat (R) de la població i immune a altres infeccions. En aquest apartat es considera només el primer model, tot i que tots dos són importants.

3.2 Model NLDS

Partint del punt que cap mètode anterior a aquest ha estat capaç de centrar-se en llindars d'epidèmies per a grafs reals, el model que es descriu a continuació de propagació de virus, no depèn de la precènsia d'una connectivitat homogènia ni de qualsevol tipus de topologia. S'assumeix una xarxa no dirigida connectada G = (N,E), on N és el número de nodes de la xarxa i E el conjunt d'enllaços. S'assumeix també una taxa universal d'infecció β per cada aresta connectada a un node infectat i una taxa de mort de virus δ per a cada node infectat.

Aquest model treballa amb petits i discrets passos de temps Δt, amb $\Delta t \to 0$. Els mateixos resultats es poden aconseguir mitjançant un cas continuu. Durant cada interval de temps Δt, un node infectat i intenta infectar als seus veïns amb probabilitat β. Al mateix temps, i es pot curar amb probabilitat δ.

Aquest procés pot ser modelat com una cadena de Markov (una *cadena de Markov* consisteix en una sèrie d'esdeveniments on la probabilitat que un esdeveniment succeeixi, només depèn de l'esdeveniment immediatament anterior) amb 2^N estats. Cada estat de la cadena de Markov correspon a una possible configuració d'un sistema de N nodes, on cadascun d'ells pot estar en dos possibles estats (susceptible o infectat), el que comporta que puguin haver-hi 2^N possibles configuracions. La configuració al temps $t+1$ només depèn de la configuració al temps t; així doncs, es tracta d'una cadena de Markov.

Symbol	Description
$\mathcal{G}$	An undirected connected graph
N	Number of nodes in $\mathcal{G}$
E	Number of edges in $\mathcal{G}$
$\mathbf{A}$	Adjacency matrix of $\mathcal{G}$
$\mathbf{A}'$	The transpose of matrix $\mathbf{A}$
β	Virus birth rate on a link connected to an infected node
δ	Virus death rate on an infected node
t	Time stamp
$p_{i,t}$	Probability that node i is infected at t
$\vec{\mathbf{P}}_\mathbf{t}$	$\vec{\mathbf{P}}_\mathbf{t} = (p_{1,t}, p_{2,t}, \ldots, p_{N,t})'$
$\zeta_{i,t}$	Probability that node i does not receive infections from its neighbors at t
$\lambda_{i,A}$	The ith largest eigenvalue of $\mathbf{A}$
$\vec{\mathbf{u}}_{\mathbf{i,A}}$	Eigenvector of $\mathbf{A}$ corresponding to $\lambda_{i,A}$
$\vec{\mathbf{u}}'_{\mathbf{i,A}}$	Transpose of $\vec{\mathbf{u}}_{\mathbf{i,A}}$
$\mathbf{S}$	The "system" matrix describing the equations of infection
$\lambda_{i,S}$	The ith largest eigenvalue of $\mathbf{S}$
s	Score $s = \beta/\delta \cdot \lambda_{1,A}$
η_t	Number of infected nodes at time t
$\langle k \rangle$	Average degree of nodes in a network
$\langle k^2 \rangle$	Connectivity divergence (sum of squared degrees)

Figura 3.1: Taula de símbols

Aquesta cadena de Markov també te un estat absorbent, quan tots els nodes es troben a l'estat de susceptible. Per arribar a aquest estat es pot fer des de qualsevol estat de la cadena de Markov, implicant que aquest estat es repetirà amb probabilitat 1 per un període llarg de temps. No obstant, tant s'hi podria arribar en un període de temps molt curt com en un període equivalent a l'edat de l'univers (en

el qual ens trobaríem en el cas en el que l'epidèmia mai mor).

La aproximació òbvia per a resoldre la cadena de Malkov esdevé inviable per a un número elevat de N, degut al creixement exponencial de mida que experimenta la cadena. Per fer front a aquesta dificultat sustituïm aquest problema per l'equació següent, on s'observa la probabilitat que un node i estigui infectat al temps t ($p_{i,t}$):

$$1 - p_{i,t} = (1 - p_{i,t-1}) + \tau_{i,t} + \delta p_{i,t-1}\tau_{i,t}$$

on:

$$i = 1 \cdots N$$

i on:

$$\tau_{i,t} = \prod_{j:neighbour of i} (1 - \beta * p_{j,t-1})$$

Aquesta equació representa el model NLDS (non-linear dynamical system), i la figura 3.2 mostra el seu diagrama de transició:

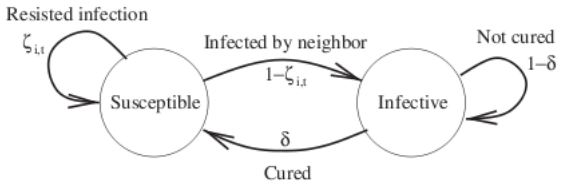

Figura 3.2: El model SIS vist des de un node en concret.

Per tant, a NLDS es denota la probabilitat que un node i s'infecti al temps t com a $p_{i,t}$. Es denota $\tau_{i,t}$ com la probabilitat que un node i no rebrà cap infecció per part dels seus veïns al següent instant de temps. Això passa si cada veí o està desinfectat, o està infectat però falla alhora de contagiar el virus amb una probabilitat (1-β).

En aquest model s'assumeix que les probabilitats $p_{j,t-1}$ són independents entre elles.

Un node i és sa a qualsevol temps t si no va rebre cap infecció dels veïns al temps t i i va ser desinfectat al temps *t-1*, o va ser infectat al *t-1*, pero va ser curat al temps t.

Així doncs, donada una topologia de xarxa i uns valors particulars de β i δ, podem resoldre l'equació d'adalt númericament per tal d'obtenir l'evolució temporal de la població infectada. Aquesta evolució ve donada per:

$$\eta_t = \sum_{i=1}^{N} p_{i,t}$$

Arribats a aquest punt, s'ha explicat detalladament tot el model NLDS matemàticament. Per tal de simular aquest model en un computador aquest variarà lleugerament, quedant de la següent manera:

- Es defineix un valor per a β i δ.
- Es comença cada simulació amb un conjunt aleatori de nodes infectats donada una topologia.
- Durant cada instant de temps, un node infectat intenta infectar a cadascun dels seus veïns amb una probabilitat β. A més a més, cada node infectat es pot curar amb una probabilitat δ. Un intent d'infecció a un node ja infectat no ha de tenir cap efecte.

3.3 Model SIC

Per dur a terme aquest projecte i passar de l'àmbit matemàtic a l'àrea de les xarxes òptiques, i més concretament a MPLS, s'ha fet una reestructuració del model SIS/NLDS per tal d'obtenir un model òptim per al treball amb xarxes MPLS i l'establiment de camins entre diferents nodes.

El model SIC consta de tres estats:

- S: susceptible
- I: infectat
- C: caigut

Amb el següent diagrama d'estats:

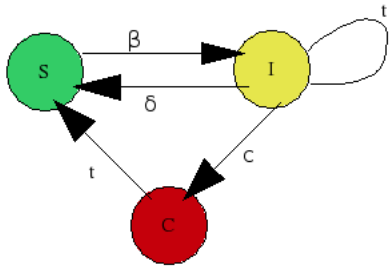

Figura 3.3: Diagrama d'estats del model SIC

El diagrama de la figura 3.3 s'explica a continuació:

- Al començament del funcionament del sistema tots els nodes es troben en estat SUSCEPTIBLE, i s'escull aleatòriament els nodes que s'infecten.
- Un node pot passar de l'estat SUSCEPTIBLE a l'estat INFECTAT amb probatilitat β (el mateix paràmetre que al model NDLS).
- En el cas que un node arriba a ser INFECTAT, s'engega un *time to repair* diferent per a cada node. Un cop aquest temporitzador arriba a zero, mitjançant una probabilitat δ (la mateixa que al model anterior), es determina si el node passa a SUSCEPTIBLE. Si no passa a SUSCEPTIBLE (per tant es cumpleix 1-δ), mitjançant una altra probabilitat c es determina si el node passa a CAIGUT. Si no passa a CAIGUT (per tant es cumpleix 1-δ i 1-c) el node torna a engegar el temporitzador de l'estat INFECTED, i per tant fins que no torni a acabar aquest, romandrà en el mateix estat.
- Quan un node arriba a l'estat de CAIGUT significa que només podrà recuperarse, i passar a l'estat SUSCEPTIBLE mitjançant un altre *time to repair* diferent de l'estat INFECTAT.

Així doncs, arribats a aquest punt i resumint una mica els punts descrits prèviament, en el model SIC els nodes disposaran de quatre paràmetres: β, δ, *time*

to repair a l'estat I i *time to repair* a l'estat C.

El rol que juga cada estat en aquest model està lligat a MPLS, i per cada estat trobem les següents tasques:

- Quan un node es troba en estat SUSCEPTIBLE el node funciona amb total normalitat, sempre que no tingui un node INFECTAT o CAIGUT com a veí.
- Quan un node es troba en estat INFECTAT deixa passar les connexions existents que tenia fins al moment just d'infectar-se. En canvi, no deixa crear connexions noves a través seu implicant a tots els seus enllaços.
- Quan un node es troba en estat CAIGUT totes les connexions que passen per aquell node es perden i n'impedeix crear de noves fins que no passa a SUSCEPTIBLE.

Capítol 4

Disseny i implementació

Al llarg d'aquest capítol s'explica l'evaluació temporal que s'ha seguit per adquirir els coneixements necessaris per tal de realitzar el projecte, quines han estat les metodologies emprades per a dur a terme la seva implementació i, finalment, com s'ha estructurat l'entorn de simulació.

4.1 Organització del projecte

Per tal d'assolir els objectius del projecte, s'han hagut de realitzar tasques complementàries com per exemple, una formació acadèmica en l'àmbit de les xarxes òptiques, entre d'altres.

Aquestes tasques exposades a continuació s'han dut a terme en l'ordre temporal que indica la figura 4.1:

	SETMANA			
abril	1	2	3	4
Lectura de conceptes fonamentals de simuladors d'esdeveniments discrets				
Lectura de surveys de comparatives de simuladors de xarxes				
Assistència a la classe de l'assignatura de Xarxes del Màster IIA				
Lectura sobre conceptes de xarxes òptiques				
maig	1	2	3	4
Assistència a la classe de l'assignatura de Xarxes del Màster IIA				
Lectura de la documentació del simulador OMNet++				
Lectura sobre camins virtuals (G/MPLS)				
Lectura sobre protecció de xarxes				
Exposició oral sobre OMNet++ al Màster IIA				
juny	1	2	3	4
Lectura de la documentació del simulador OMNet++				
Implementació d'un model senzill				
Lectura del paper de Propagació d'errors a través d'una xarxa de transport				
Implementació d'un model senzill del model NLDS				
Disseny de l'entorn de simulació per a xarxes MPLS.				
Escriptura de la memòria				
Implementació de l'entorn de simulació de fallades per a xarxes òptiques				
juliol	1	2	3	4
Lectura de la documentació del simulador OMNet++				
Implementació de l'entorn de simulació de fallades per a xarxes òptiques				
Escriptura de la memòria				
Anàlisi de resultats				

Figura 4.1: Evolució temporal

Segons el patró que ha seguit la figura 4.1, a la imatge 4.2 trobem una estimació temporal en hores de la durada del projecte:

	Hores
Hores d'estudi d'OMNet++	80
Hores d'estudi previ	40
Hores de disseny	20
Hores d'implementació	120
Hores d'anàlisi de resultats	60
Hores de redacció de la memòria	40
Total:	360

Figura 4.2: Estimació en hores

I en el gràfic de la imatge 4.3 es veu el repartiment de cada part en el temps total del projecte:

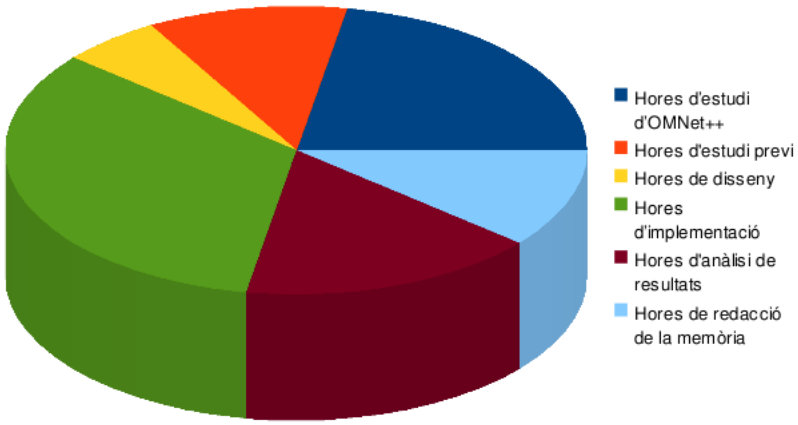

Figura 4.3: Repartiment temporal

Com es pot observar, el percentatge més gran se l'enduu la tasca d'implementació, seguida d'aprop per la tasca d'estudi del simulador OMNet++.

4.2 Metodologia

Per a la realització de l'entorn de simulació m'he basat en la metodologia *d'extreme programming*, que és una metodologia àgil per al desenvolupament de software eficient i efectiu. Vam escollir aquesta metodologia ja que està orientada al canvi constant de funcionalitats i requeriments, que va en funció de les necessitats del client i l'evolució del projecte. La metodologia fomenta que ser capaç d'adaptar un projecte als canvis constants en qualsevol punt de vida del projecte és una millor aproximació i més realista que intentar definir tots els requisits i funcionalitats de forma fixa al principi del projecte i invertir esforços a posteriori per realitzar-ne les modificacions pertinents.

Hi ha cinc principis que defineixen *d'extreme programming*:

- Simplicitat: Es simplifica el disseny per afavorir el desenvolupament i facilitar el manteniment. També s'aplica la simplicitat a la documentació, de forma que és documenti la funcionalitat exclusiva del codi comentat.
- Comunicació: Com més simple sigui el codi més comunica a la resta de desenvolupadors o als membres que s'hi afegeixin posteriorment, i encara més si la

documentació és concreta i bàsica sobre les funcionalitats del codi. Tot i que la metodologia descriu treball en parelles, en el nostre cas la feina ha estat individual però amb constant comunicació amb l'equip de desenvolupadors de RedIRIS que han anat examinant el codi.

- Feedback: Com que el client està integrat en el projecte, la seva opinió sobre l'estat del projecte és directa i constant. A més, la comunicació és constant i el client veu el resultat en tot moment, sol·licitant els canvis si són necessaris sense haver de refer tot una part. També permet que indiqui als desenvolupadors les parts essencials amb les que vol centrar el seu projecte perquè així hi dediquin més atenció.

- Coratge: Es parla de coratge quan es vol fer referència a la valentia per implementar les característiques que requereix el client en un moment determinat i no basar-ho tot en una visió més flexible que pugui permetre més flexibilitat i funcionalitats en un futur.

- Respecte: En el nostre cas no s'utilitza clarament però defineix en el respecte entre programadors per tal de consensuar solucions i no xafar el treball realitzat per la resta de companys.

Les característiques i avantatges de l'ús d'*d'extreme programming* són les següents:

- Desenvolupament iteratiu i constant: petites millores, una rere l'altre, sotmeses a proves de funcionalitat.

- Proves continues i específiques: per cada petita millora es realitza la prova de funcionament i a mida que es van afegint noves funcionalitats es segueixen repetint les proves sobre les parts anteriors per avaluar si han perdut efectivitat.

- Integració constant amb el client: La comunicació és constant i li permet al client mantenir-se informat en tot moment de l'evolució del projecte i fer-hi les indicacions necessàries.

- Correccions constants de codi: al realitzar proves de forma regular es detecten els errors ràpidament i es poden solucionar amb més freqüència.

- Refactorització del codi: capacitat per reescriure petites parts del codi que no afectin a la seva funcionalitat final però que en millorin l'eficiència a partir dels errors trobats en les constants proves

- Simplicitat del codi: cada part té una funcionalitat concreta i senzilla de forma que es pot realitzar amb més precisió la detecció i posterior correcció d'errors i és més fàcil d'entendre la funcionalitat de cada part.

- Propietat compartida del codi: Al ser una metodologia normalment aplicada en petits grups de programadors, tots ells estan al corrent de les modificacions i les funcionalitats de cada part del codi, poden aportar el seu gra de sorra a cada punt.

4.3 Implementació de l'entorn de simulació

En el capítol 2 ja s'ha definit a fons tota la funcionalitat del simulador OMNet++. Ha quedat clar que aquest simulador aporta principalment una bona interfície gràfica per interactuar amb l'usuari (la qual cosa el fa molt atractiu), una llibreria del nucli de simulació i un compilador per al llenguatge descriptor de topologies NED.

Les llibreries de protocols implementades per aquest simulador es poden trobar a un subapartat de la pàgina web oficial d'aquest [21]. Per a la realització d'aquest projecte, amb l'objectiu de simular fallades a xarxes òptiques de transport, la llibreria de protocols escollida ha sigut *INET framework* [26], la qual és definida com a un paquet de simulació de xarxes de comunicacions open-source per a OMNet++. El paquet *INET framework* conté models per a un conjunt de protocols d'internet, tals com: UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, IEEE 802.11, MPLS, OSPF, entre d'altres. El manual d'usuari de la llibreria es pot trobar a [27], amb una llista de totes les classes implementades i la seva funcionalitat.

Per començar a agafar traça amb l'IDE (Integrated Development Environment) del simulador, es va decidir començar a programar un petit exemple que es pot trobar explicat en forma de tutorial a [28].

4.3.1 Primer contacte amb OMNet++: Tic Toc

Aquest apartat es basa en l'exemple de simulació Tictoc, el qual es pot trobar a la carpeta samples/tictoc del directori d'instal·lació de OMNet++, per tal que el lector pugui probar immediatament com funciona l'exemple. No obstant, és aconsellable que l'usuari segueixi els primers passos explicats al tutorial de la web [28], amb el fí de familiaritzar-se més amb l'entorn de treball de OMNet++.

Degut a que l'àrea en la qual té més aplicació OMNet++ és en la simulació de xarxes de telecomunicacions, l'exemple triat per implementar com a primera presa de contacte forma part d'aquesta àrea.

Inicialment, es disposa d'una xarxa que consisteix en dos nodes. Els nodes han de dur a terme una tasca senzilla: un dels nodes crea un paquet, i ambdós nodes continuen passant-se el mateix paquet eternament. Els nodes s'anomenen tic i toc.

Els passos a seguir per a dur a terme la implementació són els següents:

1. Crear un directori anomenat "tictoc", i fer cd a aquell directori.
2. Descriure la xarxa que utilitzarem d'exemple creant un arxiu de topologia (NED). Un arxiu de topologia és un arxiu de text que identifica els nodes de la xarxa amb els enllaços que els connecten. Es pot crear amb qualsevol editor de text i anomenarem aquest fitxer "tictoc1.ned":

```
simple Txc1
{
    gates:
        input in;
        output out;
}

network Tictoc1
{
    submodules:
        tic: Txc1;
        toc: Txc1;
    connections:
        tic.out --> {  delay = 100ms; } --> toc.in;
        tic.in <-- {  delay = 100ms; } <-- toc.out;
}
```

L'arxiu es llegeix més fàcilment si es comença pel final. Diu el següent:

- TicToc1 és una xarxa, que està formada per dos submòduls, tic i toc. tic i toc són instàncies del mateix tipus de mòdul anomenat **Txc1**. Es connecta la porta de sortida de tic (anomenada out) amb la porta d'entrada de toc (anomenada in), i viceversa. En aquest enllaç que uneix ambdós nodes hi haurà un retràs de propagació de 100ms en tots dos sentits.
- **Txc1** és un mòdul simple (la qual cosa significa que ha d'estar implementat en C++). **Txc1** té una porta d'entrada anomenada in, i una porta de sortida anomenada out.

3. Arribats a aquest punt, es necessita implementar la funcionalitat del mòdul simple **Txc1**. Ho durem a terme mitjançant la classe **txc1.cc**:

```
#include <string.h>
#include <omnetpp.h>


class Txc1 : public cSimpleModule
{
  protected:
    // The following redefined virtual function holds the algorithm.
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};

Define_Module(Txc1);

void Txc1::initialize()
{
    // Am I Tic or Toc?
    if (strcmp("tic", getName()) == 0)
    {
        // create and send first message on gate "out". "tictocMsg" is an
        // arbitrary string which will be the name of the message object.
        cMessage *msg = new cMessage("tictocMsg");
        send(msg, "out");
    }
}

void Txc1::handleMessage(cMessage *msg)
{
send(msg, "out");
}
```

El mòdul simple **Txc1** es representa mitjançant la classe en C++ **Txc1**, la qual ha de ser derivada de **cSimpleModule** i ha de ser registrada a OMNet++ amb la macro **Define_Module()**. Es redefineixen dos mètodes de **cSimpleModule**: initialize() i handleMessage(). Ambdós són cridats des del kernel del simulador: el primer d'ells només un cop i el segon cada cop que un missatge arribi al mòdul.

A initialize() es crea l'objecte missatge **(cMessage)**, i s'envia per la porta out. Com que aquesta porta està conectada a la porta d'entrada de l'altre node, el simulador del kernel enviarà el missatge a l'altre node després d'un retràs de 100ms. Aquest altre node rebrà el missatge a handleMessage() i el tornarà a enviar al seu homònim, resultant en un procés d'un continu ping-pong.

Els missatges i els esdeveniments (timers, timeouts) són tots representats mitjançant la classe **cMessage** a OMNet++. Un cop s'envia o es programa un missatge, el kernel del simulador els mantindrà a la seva llista "d'esdeveniments programats"o de "esdeveniments futurs"fins que els arribi l'hora de ser rebuts pel destí via handleMessage().

Cal remarcar que en aquest exemple no hi ha cap tipus de mecanisme per tal d'aturar l'execució, sino que continuarà infinitament fins que es faci un STOP al GUI. No obstant, seria fàcil posar-ne algun tal com: un contador màxim de paquets enviats o un temps de simulació de CPU.

4. Ara ha arribat el moment de crear el Makefile el qual ajudarà alhora de compilar i enllaçar el programa per crear l'executable tictoc:

   ```
   $ opp_makemake
   ```

   Aquesta commanda hauria d'haver creat un Makefile al directori actual.

5. Per compilar i enllaçar aquesta simulació senzilla hem d'escriure la següent commanda:

   ```
   $ make
   ```

6. Per últim, és necessari crear un fitxer omnetpp.ini, el qual és l'encarregat de dir al simulador del kernel quina xarxa es vol simular (ja que podem tenir diferents topologies de xarxa a un mateix model). El nostre arxiu .ini contindrà el següent:

   ```
   [General]
   network = tictoc1
   ```

7. Un cop completats els passos anteriors, el programa s'executa escribint la següent commanda:

   ```
   $ ./tictoc
   ```

8. Prémer el botó de Run a la barra d'eines per tal de començar la simulació. S'hauria de veure els nodes tic i toc intercanviant missatges entre ells. La següent figura mostra el resultat de l'execució:

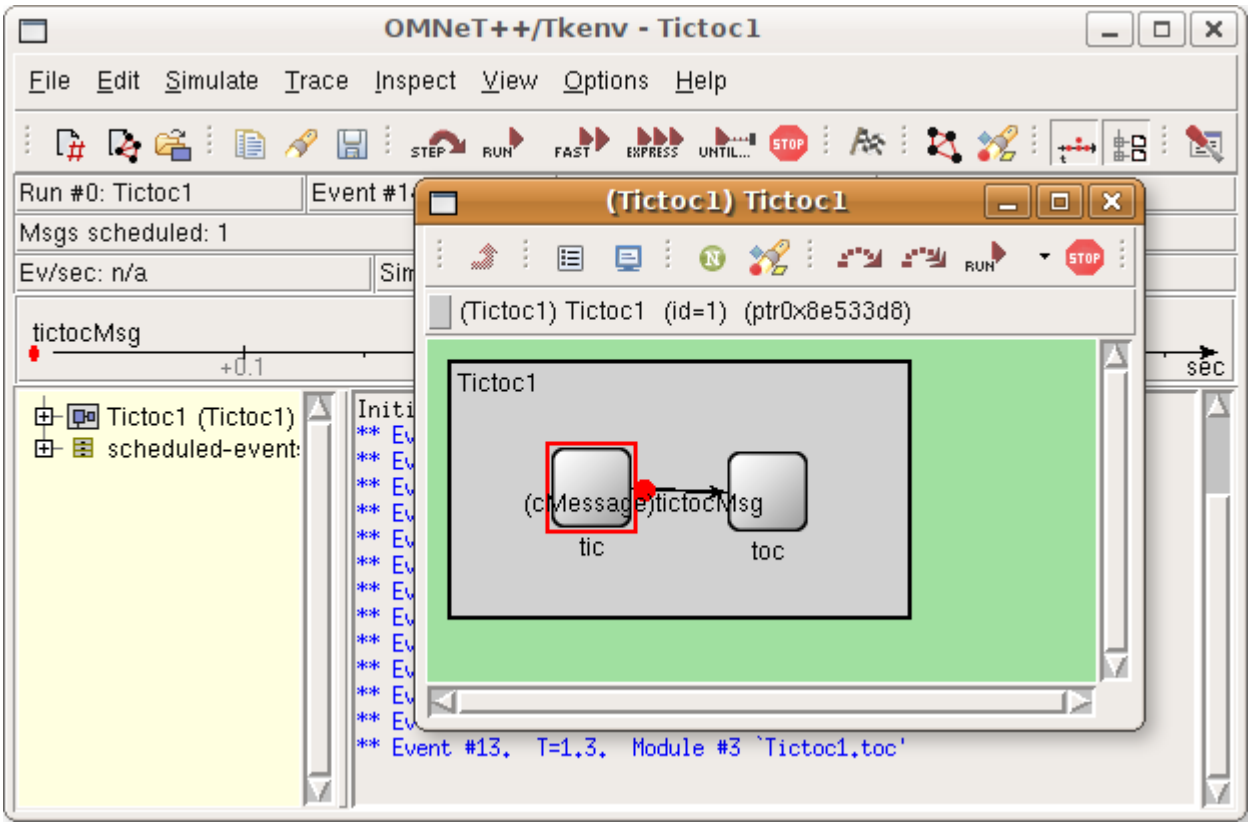

Figura 4.4: Visualització de l'execució de la simulació

9. Per sortir de la simulació cal prémer el botó STOP i després anar a FILE i EXIT.

Un cop entés aquest primer exemple, el pas següent ha estat l'ampliació d'aquest.

4.3.2 Model TicToc Extès

Primerament s'han hagut de definir les funcionalitats que es vol que el model contingui:

- Una topologia on els nodes només poden enviar missatges als seus veïns.
- Un mecanisme pel qual es pugui administrar el comportament del tràfic de

la xarxa. Aquest mecanisme ha de fer arribar als nodes les tasques a dur a terme.

- Un tipus d'enllaç que pugui fallar.

Seguidament es detalla com han sigut incorporades al model TicToc.

Topologia

Per satisfer aquesta funcionalitat, s'ha implementat una topologia de 4 nodes, tal i com es pot veure a la figura 4.5. Aquesta topologia permet que els nodes no es vegin a tots entre ells, i per tant detectar una fallada quan un node intenta enviar un missatge a un destí que no es veí seu.

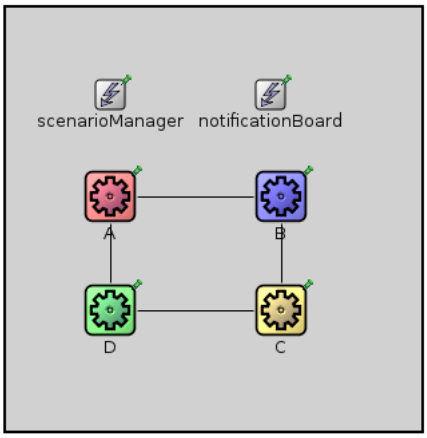

Figura 4.5: Topologia del model TicToc Extès

Com es pot observar, no només apareixen els 4 nodes a la nostra xarxa, sino que també apareixen dos mòduls més: *scenarioManager* i *notificationBoard*. Tots dos s'expliquen a continuació.

Component myScenarioManager

La segona funcionalitat a implementar s'ha dut a terme mitjançant la derivació d'un mòdul proveït per la llibreria INET. Aquest mòdul s'anomena *ScenarioManager* i la seva funció és controlar els experiments amb simulacions. Permet programar esdeveniments a diferents instans de temps especificats, que poden ser des de canviar el valor d'un paràmetre fins a crear o eliminar connexions entre dos nodes, de manera

que l'usuari pugui observar el comportament de la xarxa en la transició d'esdeveniments.

El motiu que ha dut a que es derivi *ScenarioManager* i que s'obtingui *myScenarioManager* ha sigut principalment que, aquesta classe per definició, llegeix les commandes de control d'un fitxer XML passat com a paràmetre, cosa que no interessava en aquest moment.

Per tal que *myScenarioManager* pugui administrar el comportament de la xarxa i, per tant, tingui accés a tots els nodes en qualsevol instant de temps, a la xarxa s'ha afegit un mòdul anomenat *NotificationBoard*. La funció de *NotificationBoard* és produir diferents tipus d'interrupcions degudes a diferents tipus d'esdeveniments i que, tot mòdul que estigui esperant una interrupció en concret sigui avisat, amb la finalitat que dugui a terme una tasca derivada de l'esdeveniment que ha provocat la interrupció.

Mitjançant aquest mecànisme, es dota a *myScenarioManager* amb la capacitat de controlar i administrar la xarxa, tot des del mateix mòdul.

Les tasques que *myScenarioManager* espera que duguin a terme els nodes són tan senzilles com simplement dir a un node origen que enviï un missatge a un origen destí i que, si aquest destí no és veí seu, tregui un missatge per pantalla on digui que no ha estat possible l'enviament del missatge.

Fins ara, no s'ha tingut en compte la qualitat de l'enllaç que separa dos nodes. Al següent apartat es té en compte aquest fet.

Component FailureChannel

Els enllaços predefinits de la llibreria d'OMNet++ (**cChannel**) ja disposen d'una sèrie de paràmetres configurables per l'usuari:

- Bit error rate.
- Packet error rate.
- Delay.
- Data rate.

No obstant, cap d'ells modela el comporament que es vol obtenir d'un canal, el qual es vol que estigui o en funcionament o caigut. Per tant, per afegir el paràmetre *failure* a un canal, s'ha hagut de derivar la classe *cDatarateChannel* amb l'obtenció de *FailureChannel*, un canal amb una probabilitat configurable per l'usuari de que l'enllaç, en un moment determinat, estigui en funcionament o no.

Per introduir el canal amb probabilitat de fallar, s'ha hagut de substituir a l'apartat de connexions del .NED el següent:

```
A.port++ <--> { delay = default(0.0)s } <--> B.port++;
```

Pel següent:

```
A.port++ <--> FailureChannel {  failure = default(0.0); } <--> B.port++;
```

Així doncs, i després de l'execució d'algunes provés de simulació, s'ha acabat la implementació del model TicToc Extès.

4.3.3 Model NLDS

Un cop finalitzada la primera etapa en la qual l'objectiu era familiaritzar-se amb l'entorn de treball d'OMNet++, el següent pas és implementar el comporament d'una xarxa descrit pel model NLDS explicat al capítol 3. Amb aquesta tasca, s'acabarà de conèixer a fons tota l'estructura que envolta el kernel del simulador i de la llibreria INET. A la secció 4.4 s'explica detalladament tot el procés per a la implementació d'aquest model a OMNet++.

4.4 Implementació del Model NLDS

Anteriorment ja hem parlat del model NLDS [10]. L'objectiu d'implementar el comportament d'aquest model a NLDS, m'ha donat la oportunitat de crear un projecte des de zero en un entorn de treball inicialment desconegut per mi com era OMNet++.

Partin del punt en què ja es té un mínim coneixement de l'estructura que ha de seguir un model per ser simulat a OMNet++, es vol construir un sistema que, donat uns valors passats com a paràmetre per β i δ, i donada una topologia es pugui simular el comportament explicat a [10].

Així doncs, l'arquitectura del sistema ha quedat definida de la següent manera, tal i com es veu a la imatge 4.6.

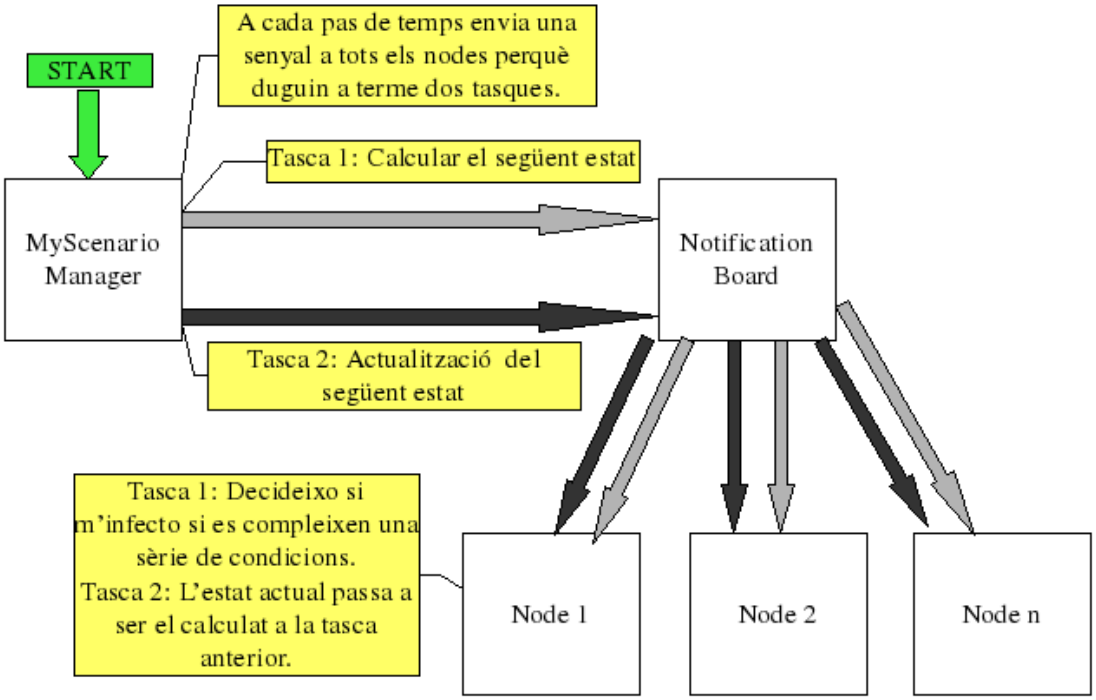

Figura 4.6: Arquitectura del model NLDS aplicat a OMNet++

La imatge 4.6 mostra el comportament que ha de seguir el sistema. Un cop inicialitzat aquest, s'escull un nombre aleatori de nodes que s'infecten, per tal de propagar l'error. *myScenarioManager* programa dos events a cada instant de temps de la simulació: un de càlcul del següent estat i l'altre d'actualització de l'estat. D'aquesta manera s'eviten problemes amb els que seria fàcil trobar-se, com el de solapament d'estats. Els dos events programats per *myScenarioManager* van dirigits a tots els nodes de la xarxa, els quals, seguint els patrons definits a 3.2, acaben executant les tasques. Durant cada tasca de càlcul, un node infectat intenta infectar a cadascun dels seus veïns amb una probabilitat β. A més a més, cada node infectat es pot curar amb una probabilitat δ. Un intent d'infecció a un node ja infectat no ha de tenir cap efecte.

Per donar la sensació de separació entre nodes SUSCEPTIBLES i nodes INFECTATS s'ha relacionat el color verd amb els primers i el color vermell amb els segons.

Seguidament es detallarà cada component del sistema NLDS implementat.

4.4.1 El node: Txc1

El node implementat pel model NLDS ha estat l'utilitzat per al model TicToc Extès, però afegint-li tres paràmetres: un identificador del node, el grau nodal i l'estat del node (0=SUSCEPTIBLE, 1=INFECTAT), tal i com es veu al codi .NED següent:

```
simple Txc1
{
     parameters:
        int id;
        int degree;
        int state = default(0);
        @display("i=block/routing");
    gates:
        inout  port[degree] @loose;
}
```

4.4.2 L'enllaç: FailureChannel

L'enllaç utilitzat és exactament el mateix que el definit a la secció anterior 4.3.2.

4.4.3 La topologia

La topologia del sistema NLDS s'ha volgut que formi part del conjunt de paràmetres que l'usuari pot escollir abans d'iniciar la simulació. Un servidor ja disposava d'un conjunt de topologies representant algunes xarxes reals. No obstant, per tal de crear topologies amb una quantitat elevada de nodes s'ha fet servir un generador de topologies (BRITE), i un programa en Java que genera, a partir d'una matriu d'adjacències, un arxiu .NED amb la topologia de la xarxa a simular definida correctament, tal i com espera rebre-la OMNet++.

BRITE

BRITE (Boston university Representative Internet Topology gEnerator), és un generador de topologies que, a partir d'una sèrie de paràmetres introduïts a la seva interfície gràfica, entre els que aparèix el número de nodes, genera un arxiu amb una matriu d'adjacència.

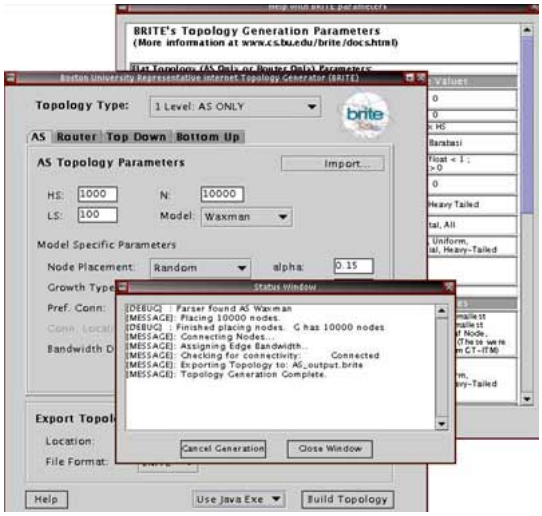

Figura 4.7: Interfície gràfica de BRITE

Cal remarcar que aquest graf generat per BRITE no sempre és un graf connectat.

NedGenerator

A partir d'una matriu d'adjacència, el programa NedGenerator amb l'ajuda d'una llibreria de tractament de grafs (gentraf [30]), genera un arxiu .NED referent a la topologia.

NedGenerator ha de rebre com a paràmetre el tipus de node (mòdul que implementa la funcionalitat d'un node, en aquest cas Txc1) i el tipus d'enllaç utilitzat (en aquest cas FailureChannel) amb el valor de data rate per als enllaços.

Un exemple de la tasca de NedGenerator es pot observar a continuació. A

partir d'una simple matriu d'adjacències:

```
[nodenum]
6

[capacity]
# 1 2 3 4 5
0 1 0 0 0 0 # 0
1 0 1 1 0 0 # 1
0 1 0 0 1 0 # 2
0 1 0 0 1 0 # 3
0 0 1 1 0 1 # 4
0 0 0 0 1 0 # 5
```

Genera el següent arxiu .NED:

```
import  inet.base.NotificationBoard;

network test_bc3
{
@display("bgb=700,500");
submodules:
N0: Txc1 {
parameters:
id=0;
degree=4;
@display("i=block/cogwheel,blue;p=0,182");
}
N1: Txc1 {
parameters:
id=1;
degree=3;
@display("i=block/cogwheel,blue;p=276,190");
}
// Fins a 5 nodes
notificationBoard : NotificationBoard {
parameters:
  @display("i=block/cogwheel,blue;p=56,24");
}
scenarioManager : myScenarioManager {
parameters:
```

```
script = xmldoc("scenario.xml");
@display("i=block/cogwheel,blue;p=176,24");
}
connections:
N0.port++ <--> FailureChannel { dataratechannel = 2000; \
failure = default(0.0); } <--> N1.port++;
N1.port++ <--> FailureChannel { dataratechannel = 2000; \
failure = default(0.0); } <--> N2.port++;
// etc...
```

Per a la simulació d'aquest model, s'han arribat a utiltizar topologies de fins a 1000 nodes.

4.4.4 L'administrador: myScenarioManager

El component que crea tots els esdeveniments que es produeixen a la xarxa i que decideix quan s'han de produir és *myScenarioManager*. Com ja s'ha explicat a la secció 4.3.2, *myScenarioManager* utilitza el mòdul Notification Board per tal d'avisar, síncronament, a tots els nodes cada cop que han de realitzar una tasca. No obstant, se li han afegit certs elements per què s'adeqüés a la funcionalitat esperada al model NLDS:

- Un paràmetre amb el número total de Time Steps que es volen realitzar. En aquest cas aquest valor també defineix el número de cops que un node intentarà curar-se o infectar als seus veïns, en cas de que estigui INFECTAT.
- Un paràmetre que defineix el tant per cent de nodes infectats al principi de la simulació, per tal de començar a propagar la infecció.

En el següent tros de codi, *myScenarioManager* dispara les dues interrupcions dirigides als nodes. Com es pot observar, primer sempre cridarà la interrupció de càlcul, i després la d'actualització:

```
void myScenarioManager::handleMessage(cMessage *msg)
{
nb->fireChangeNotification(CALCULATION_EVENT ,
(const cPolymorphic*)msg->getContextPointer());
nb->fireChangeNotification(UPDATE_STATE_EVENT ,
(const cPolymorphic*)msg->getContextPointer());
s->writeStats(top);
}
```

La tercera crida que apareix al codi és dirigida al mòdul d'estadístiques, explicat a continuació.

4.4.5 El recollidor d'estadístiques: statistics

La funció del mòdul recollidor d'estadístiques és la de, cada cop que tots els nodes actualitzen el seu estat, fer un recorregut per tota la topologia per saber quin tant per cent dels nodes de tota la xarxa es troben infectats a cada *time step* de la simulació. Les dades obtingudes són escrites a un arxiu de text preparades per a un tractament posterior.

Finalment, a les imatges següents es pot observar el sistema NLDS implementat a OMNet++ executant-se en diferents topologies.

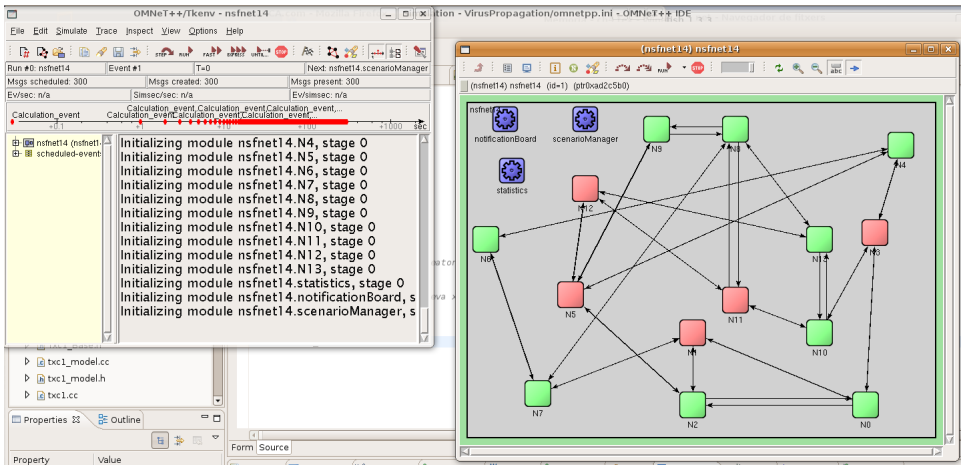

Figura 4.8: Execució topologia Nsfnet14 amb β=0.5 i δ=0.5

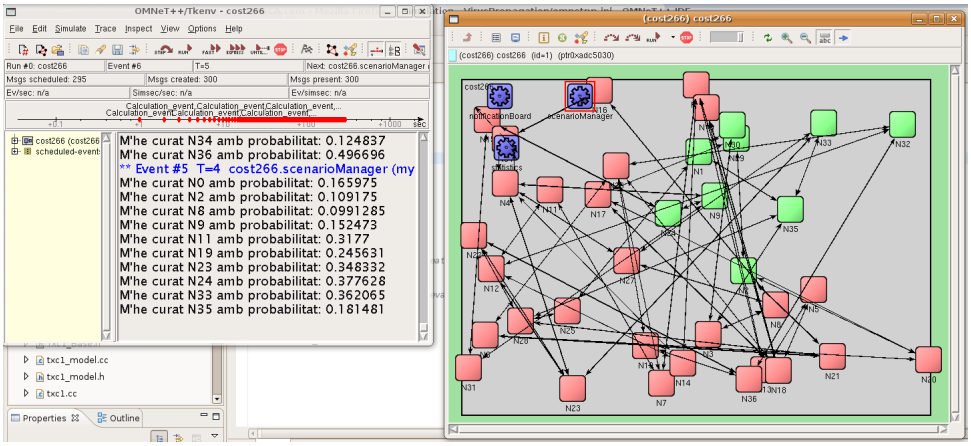

Figura 4.9: Execució topologia Cost266 amb β=0.5 i δ=0.5

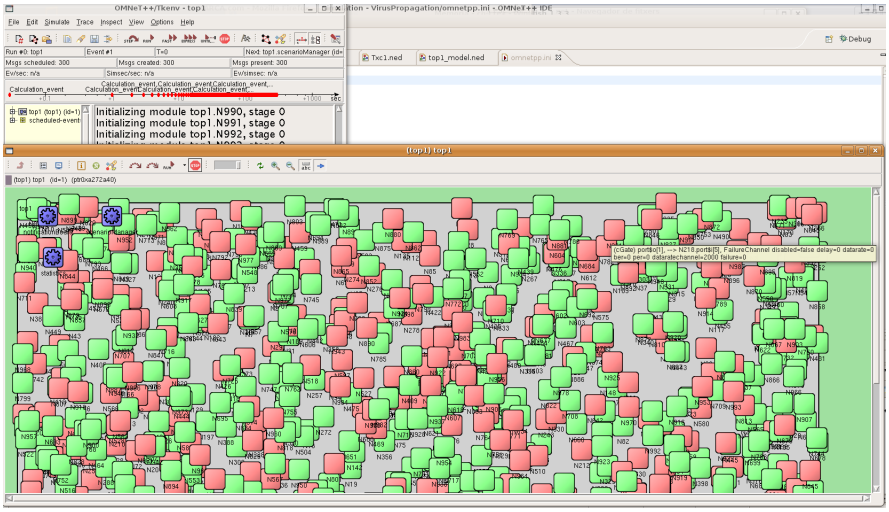

Figura 4.10: Execució topologia de 1000 nodes creada amb BRITE amb β=0.5 i δ=0.5

4.5 Implementació del Model SIC

Al capítol 3, a la secció 3.3, s'ha definit un model de propagació de fallades de tres estats (SUSCEPTIBLE, INFECTAT i CAIGUT) capaç de ser aplicat a una xarxa funcionant amb MPLS. En aquesta secció es detallen tots els components que s'han

emprat per tal de dur a terme la implementació del model SIC a OMNet++.

L'arquitectura que segueix el model SIC és idèntica a la del model implementat NLDS 4.6. El comportament que segueix aquest model es defineix a 3.3.

En el model SIC, al igual que al model NLDS, s'ha associat un color per a cada estat possible del node: l'estat SUSCEPTIBLE és verd, l'INFECTAT el taronja i el CAIGUT el vermell. Com a paràmetres, a part dels 3 que ja s'empràven al model NLDS (β, δ i la topologia), el model SIC en necessita dos més:

- El Minimum Time To Repair (MTTR) des de l'estat INFECTAT.
- El Minimum Time To Repair (MTTR) des de l'estat CAIGUT.

Així doncs, a continuació es descriu cadascun dels components que formen part del sistema SIC implementat. Cal recordar que la llibreria INET-framework disposa del protocol MPLS implementat juntament amb tot un conjunt de components dedicats a aquest protocol.

4.5.1 El node: RSVP_LSR / RSVP_FAILED

Amb la finalitat de poder simular els tres estats diferents del model SIC, aquest sistema consta de dos tipus de nodes de la llibreria d'INET-framework:

- RSVP_LSR: Quan el node es troba en SUSCEPTIBLE o INFECTAT, utiltiza aquest mòdul. Si el node acaba passant a l'estat CAIGUT, aquest node RSVP_LSR és substituit a la xarxa mitjançant *FailureManager* per un node RSVP_FAILED. Es defineix com a node *sa* aquell que està a l'estat SUSCEPTIBLE o INFECTAT i que, per tant, utilitza el mòdul RSVP_LSR.
- RSVP_FAILED: Quan el node es troba en estat CAIGUT, utilitza aquest altre mòdul. Un cop s'acava el temporitzador i el node passa a SUSCEPTIBLE, el node és substituit a la xarxa mitjançant *FailureManager* per un node RSVP_LSR. Es defineix com a node *caigut* aquell que està a l'estat CAIGUT i que, per tant, utilitza el mòdul RSVP_FAILED.

RSVP_LSR

El mòdul escollit per a dur a terme la funció de node "en funcionament"a la xarxa ha estat el mòdul compost RSVP_LSR. RSVP_LSR, al ser un mòdul compost (format

per un cojunt de mòduls simples) no disposa d'una classe en C++ que descrigui el seu comportament. Està format per un conjunt de components que, tots junts, fan possible la simulació del comportament d'un router per a xarxes MPLS. Aquest conjunt de components es poden veure a la figura 4.11.

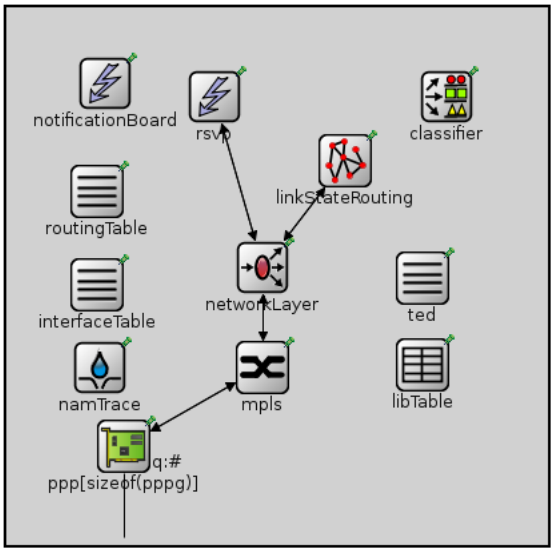

Figura 4.11: Components de RSVP_LSR

Com es pot observar, els components estan comunicats entre ells. Seguidament es descriu breument la funcionalitat de cada membre de RSVP_LSR:

- *NotificationBoard*: El fet que RSVP_LSR sigui un mòdul força complex, ja que acull totes les funcionalitats d'un router MPLS, fa necessàri la incorporació d'una *NotificationBoard* a cadascun dels routers. Tal i com s'ha vist a la secció anterior, la funció de la *NotificationBoard* és de produir diferents tipus d'interrupcions degudes a diferents tipus d'esdeveniments i que, tot mòdul que estigui esperant una interrupció en concret sigui avisat, amb la finalitat que dugui a terme una tasca derivada de l'esdeveniment que ha provocat la interrupció. En aquest cas, els routers es comuniquen entre ells informació referida a l'estat dels enllaços, entre altres coses.

- *RoutingTable*: Al tractar-se d'una xarxa amb MPLS funcionant sobre IP, cada router necessita tenir una taula d'encaminament pròpia amb informació de les direccions IP de les seves interfícies i de les interfícies veïnes. Aquesta taula, necessària a cada node, no es crea dinàmicament sino que cada node la llegeix d'un arxiu .rt prèviament passat com a paràmetre. Aquest arxiu té el següent aspecte:

```
# definició de cada interfície del node amb la IP associada
ifconfig:
name: ppp0 inet_addr: 10.10.0.1 MTU: 1500 Metric: 1
ifconfigend.

#taula d'encaminament
route:
10.10.0.2 10.10.0.2 255.255.255.255 H 0 ppp0
routeend.
```

El fet que la taula s'hagi de passar manualment a cada router ha complicat el fet de treballar amb topologies grans (més de 100 nodes). Així doncs, s'ha optat per crear aquests arxius dinàmicament a partir de la matriu d'adjacència de la topologia passada com a paràmetre. Aquesta eina implementada s'anomena *RTGenerator.java* i ha ajudat moltíssim a alleugerir la tasca de configuració del sistema.

- *InterfaceTable*: Aquest submòdul simplement té la funció d'emmagatzemar la informació llegida de l'arxiu .rt.
- *TED (Traffic Engineering Database)*: Conté la informació de la topologia de la xarxa, així com l'ample de banda lliure a cada enllaç a cada instant de temps.
- *LinkStateRouting*: Implementa un protocol senzill d'encaminament link state.
- *RSVP*: Implementa el protocol de senyalització RSVP per MPLS. Aquest submòdul és l'encarregat de rebre la informació quan es vol crear un camí, i de dur a terme aquesta tasca.
- *MPLS*: Com el seu propi nom diu, implementa el protocol MPLS.
- *LIBTable*: Guarda informació de les etiquetes dels camins.

RSVP_FAILED

El mòdul escollit per a dur a terme la funció de node caigut a la xarxa ha estat el mòdul RSVP_FAILED. Aquest mòdul simplement consta d'un component anomenat "dummy"que no fa res. Quan un node passa a CAIGUT, l'administrador d'esdeveniments (*myScenarioManager*) avisa a *FailureManager*, el qual crea un objecte RSVP_FAILED i substitueix el node que funcionava correctament a la xarxa (RSVP_LSR) pel nou objecte creat. D'aquesta forma s'aconsegueix simular el comportament que hauria de tenir un node caigut.

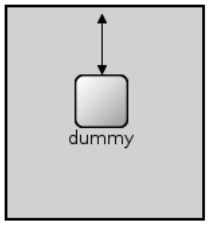

Figura 4.12: Component de RSVP_FAILED

Amb l'objectiu d'implementar el comportament del model SIC, cal afegir a RSVP_LSR i RSVP_FAILED un component extra *RouterState*.

RouterState

Aquest component és el responsable de guardar l'estat actual de cada router i de, a cada interrupció de càlcul del següent estat, interactuar amb els veïns de diferent manera segons l'estat en el que es trobi i seguint la pauta marcada pel model SIC.

De la mateixa manera, també té la responsabilitat d'avisar l'administrador d'esdeveniments (*myScenarioManager*) cada cop que un node passa a estat CAIGUT i s'ha de fer la substitució dinàmica del router RSVP_LSR pel RSVP_FAILED, o passa a l'estat SUSCEPTIBLE i s'ha de fer la substitució inversa.

Finalment els dos nodes queden de la següent manera:

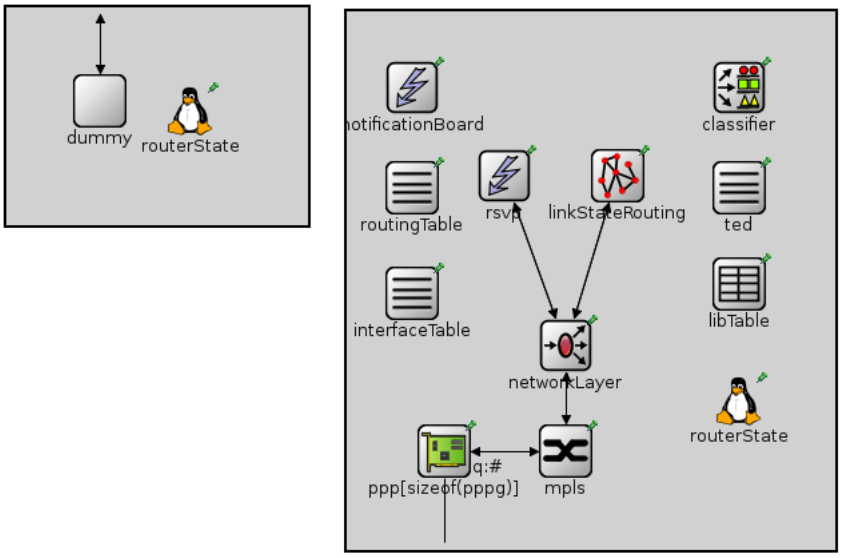

Figura 4.13: RSVP_FAILED i RSVP_LSR després d'afegir el component RouterState a tots dos

4.5.2 L'enllaç: FailureChannel

L'enllaç utilitzat és exactament el mateix que el definit a la secció anterior 4.3.2.

4.5.3 La topologia

Al igual que al model NLDS, en aquest model la topologia també s'ha volgut que formés part del conjunt de paràmetres. Per tal de crear topologies s'ha fet servir el mateix sistema que l'explicat anteriorment. Amb un generador de topologies i amb una sèrie d'eines s'ha pogut obtenir finalment les topologies desitjades. La única diferència, en quant al model NLDS, és que en aquest model els nodes són del tipus RSVP_LSR/FAILED, i abans eren Txc1.

En el model SIC, cada topologia ha de dur associat un arxiu .trf. Aquest arxiu conté informació dels camins que seran creats i alliberats al sistema.

Per a la realització de proves de creació de camins i el seu posterior alliberament, s'ha utilitzat la topologia senzilla mostrada a la figura 4.14.

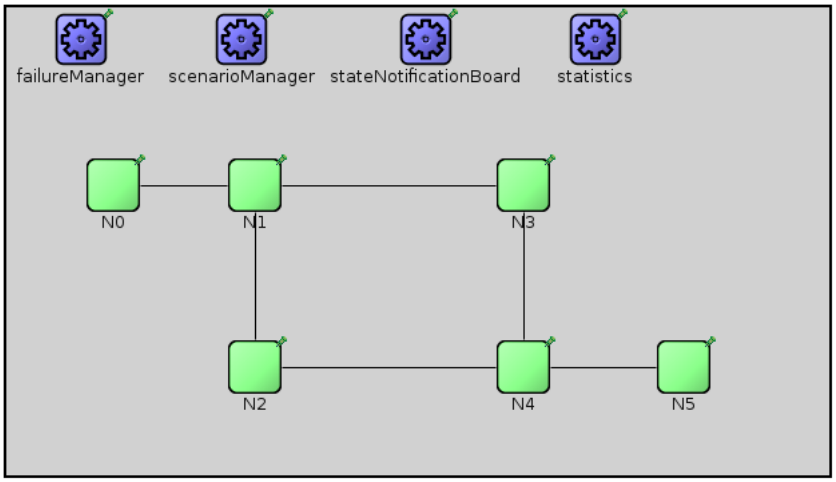

Figura 4.14: Topologia emprada per a testejar el sistema

Un cop el sistema ha funcionat amb aquesta topologia 4.14, s'ha passat a testejar el sistema implementat amb topologies més grans.

4.5.4 L'administrador: myScenarioManager

A més a més de les funcions descrites anteriorment, en el model SIC l'administrador *myScenarioManager* agafa les responsabilitats següents:

- Creació i l'alliberament de camins quan toca (segons el temps establert al fitxer .trf). Un camí només es pot crear mitjançant el fitxer .trf, en canvi, es pot alliberar o bé quan ho diu el fitxer de tràfic o bé quan un node passa a estat CAIGUT i totes les connexions que passàven per aquell node han de ser alliberades.
- Avisar el component *FailureManager* perquè realitzi els canvi d'un node sa per un node caigut quan és necessàri.
- Remoure el node caigut i posar el node sa que hi havia anteriorment quan és necessari.

4.5.5 Component: FailureManager

Aquest component duu a terme la tasca de substitució d'un node que funciona correctament (amb el mòdul RSVP_LSR) per un que no fa res i que simula l'estat de

caigut (amb el mòdul RSVP_FAILED), i la substitució inversa també. Aquesta tasca que a primera vista pot semblar senzilla, ha dut forces problemes degut a que a l'hora de fer l'intercanvi de connexions entre ambdós nodes (el node caigut passa a estar connectat amb tots els veïns del node sa, i el sa passa a no estar connectat amb ningú) encara quedava algun paquet a la xarxa dirigit al node sa. Per tal de solucionar aquest problema s'ha hagut de deixar un interval de temps entre que el node INFECTAT passa a estat CAIGUT i és substituït.

4.5.6 El recollidor d'estadístiques: statistics

El mòdul d'estadístiques de SIC, a més a més de calcular quants nodes hi ha INFECTATS a cada instant de temps (tal i com es feia al model NLDS), calcula el número de connexions caigudes a cada instant de temps. Es defineix com a connexió caiguda aquella que encara té temps de vida però, degut a que un node pel qual passava ha passat a estat CAIGUT, s'ha hagut d'alliberar.

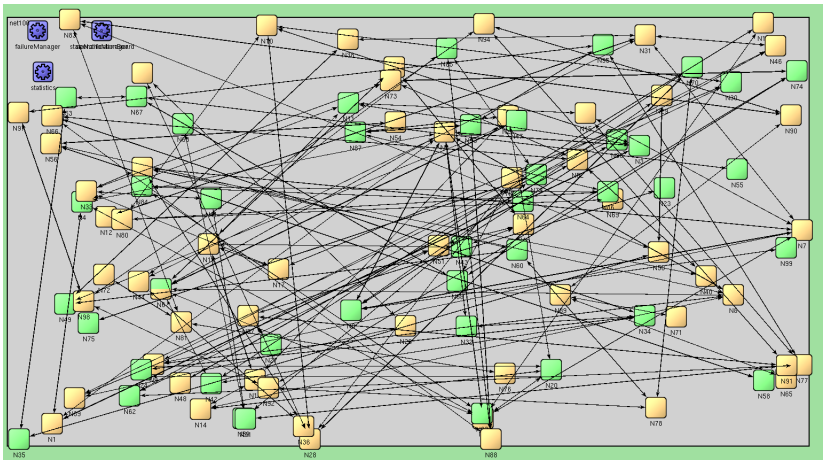

Figura 4.15: Topologia de 100 nodes just després del moment en que s'infecten aleatòriament un 20% dels nodes

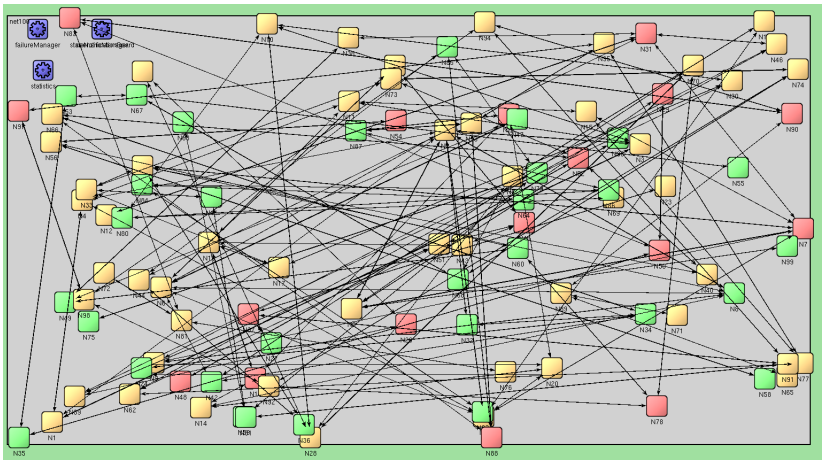

Figura 4.16: Topologia de 100 nodes amb uns quants nodes CAIGUTS abans de fer el canvi de mòdul RSVP_LSR per RSVP_FAILED

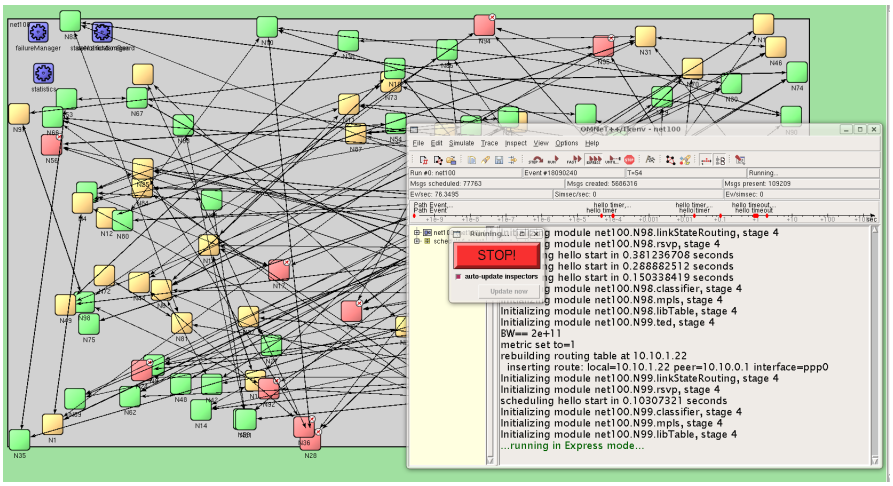

Figura 4.17: Topologia de 100 nodes amb uns quants nodes CAIGUTS després de fer el canvi de mòdul RSVP_LSR per RSVP_FAILED

Capítol 5

Resultats de la simulació

En el capítol anterior s'ha explicat el disseny i la implementació del model SIC desenvolupat amb el simulador OMNet++. En aquest apartat es reflecteixen els resultats obtinguts en diverses simulacions.

5.1 Simulació amb una topologia de 100 nodes

Els dos primers experiments que s'han dut a terme han estat realitzats sobre una topologia de 100 nodes generada amb BRITE. Aquesta topologia mostra les característiques següents:

- Disposa d'un diàmetre de 12 nodes.
- El grau nodal mínim és 1, el màxim 8 i la mitjana és de 2,780 .

En quant al arxiu de tràfic emprat:

- L'origen i el destí de les connexions és aleatori i tots tenen la mateixa propabibilitat d'aparèixer.
- La matriu de tràfic és aleatòria en la proporció relativa i és uniforme. Cap node té mes importància que els altres.
- La mitjana del temps de duració de les connexions és 10.

També cal dir que els resultats obtinguts són la mitja de 5 execucions per resultat.

5.1.1 Primera simulació

La primera simulació executada sobre aquesta topologia disposa del conjunt de paràmetres següents:

- β=0,085
- δ=0,050
- c=0,1
- Un MTTR de l'estat INFECTED per a tots els nodes igual a 2.
- Un MTTR de l'estat CAIGUT per a tots els nodes igual a 5.
- Un percentatge de nodes infectats inicials del 10%.

D'aquesta simulació s'han extret els resultats que es mostren a continuació.

En la Fig. 5.1 es mostra l'evolució temporal del percentatge de nodes que es troben en un estat determinat a cada instant de temps. A partir d'un subconjunt inicial del 10%, que comencen infectant-se aleàtoriament, s'observa com el percentatge de nodes infectats s'estabilitza cap al t=20 en un 40% aproximadamet i, per tant, la xarxa mai es cura. També es pot observar com els nodes caiguts van en funció de les pujades i baixades de la corva dels infectats, per exemple, a l'instant de temps t=150 hi ha un determinat pic de nodes infectats i en conseqüència, a l'instant t=152 s'observa que la corva dels nodes caiguts augmenta. Aquest interval (en aquest cas de 2) depèn dels valors donats per a MTTR.

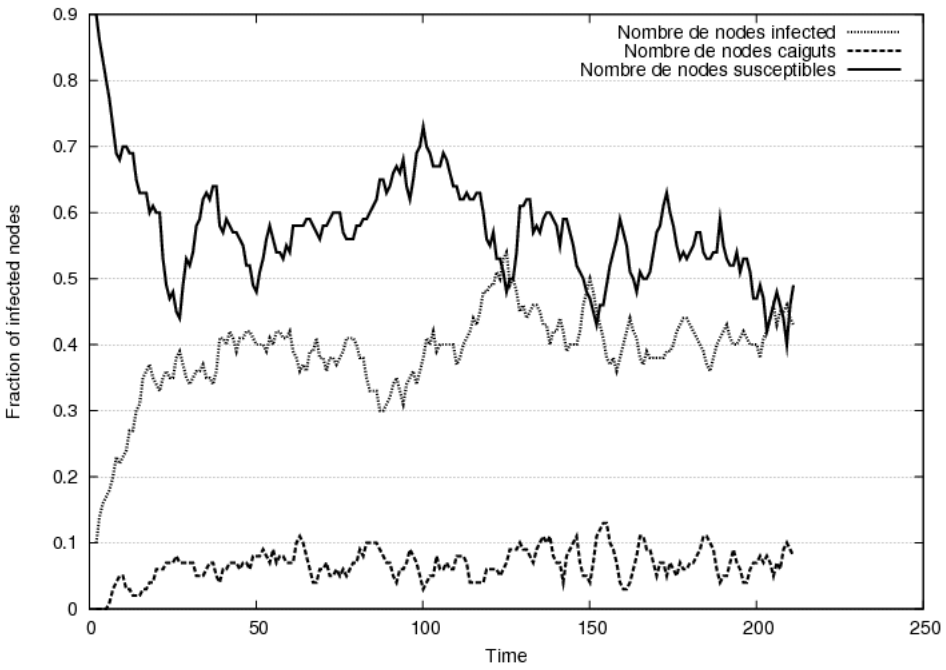

Figura 5.1: Percentatge de nodes infectats al llarg de la simulació

En la Fig. 5.2 es mostra l'evolució temporal del total de les connexions creades i caigudes. L'arxiu de tràfic utilitzat per aquesta topologia conté 50.000 connexions. A primera vista es detecta que en el total de la simulació hi ha hagut un nombre molt petit de connexions creades (gairebé el 5%). Com es pot observar a la Fig. 5.1, al principi de la simulació (abans que s'infectin el 10% de nodes, t=1), tots els nodes es troben en l'estat SUSCEPTIBLE. Com que aquest estat no te cap tipus de restricció alhora de crear camins, és normal que la corva de creació de camins al principi creixi exponencialment. Si es segueix comparant amb la Fig. 5.1, es veu que, un cop el tant per cent de nodes infectats s'estabilitza al voltant del 40%, les connexions creades commencen a minvar. Un node, quan es troba a l'estat INFECTAT, no permet la creació de nous camins, només permet el pas dels camins que ja havien estat creats abans de ser infectat. Per tant, degut que a partir d'un cert temps la xarxa conté al voltant d'un 60% dels nodes en estat SUSCEPTIBLE, s'explica que les connexions que es creen són poques.

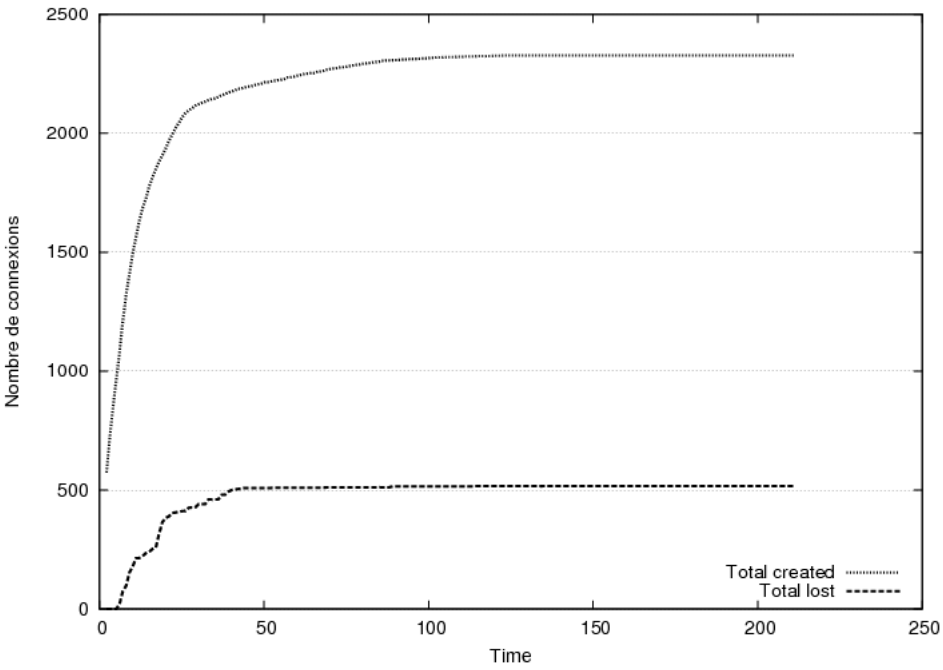

Figura 5.2: Número total de connexions creades respecte el número total de connexions caigudes

Respecte a les connexions caigudes la Fig. 5.2, s'observa que té el mateix comportament que les connexions creades i, un cop aquestes s'estabilitzen, les connexions caigudes també.

En la Fig. 5.3 es pot veure un rànking ordenat dels nodes que han perdut més connexions al llarg de la simulació. Aquesta estadística podria servir alhora de buscar zones per a ser immunitzades i per tant, reduir les connexions perdudes.

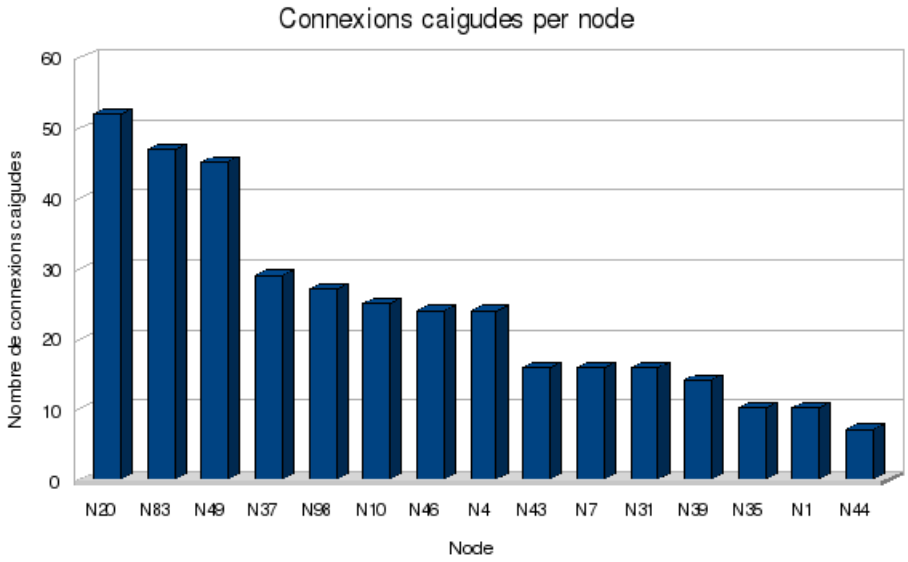

Figura 5.3: Rànking dels primers 15 nodes que han tingut una pèrdua de connexions en la simulació

5.1.2 Segona simulació

La segona simulació executada sobre aquesta topologia disposa del conjunt de paràmetres següents:

- β=0,017
- δ=0,01
- c=0,1
- Un MTTR de l'estat INFECTED per a tots els nodes igual a 2.
- Un MTTR de l'estat CAIGUT per a tots els nodes igual a 5.
- Un percentatge de nodes infectats inicials del 10%.

D'aquesta simulació s'han extret els resultats que es mostren a continuació.

En el gràfic 5.4 es mostra l'evolució temporal del percentatge de nodes que es troben o INFECTATS o CAIGUTS a cada instant de temps. En aquest cas, els valors escollits per a β, δ i c han propiciat la curació total de la xarxa a un t al

voltant de 75. Igual que a la simulació anterior, es comença amb un 10% de nodes infectats i aquest valor es va reduïnt lleugerament fins al punt en que la xarxa es cura.

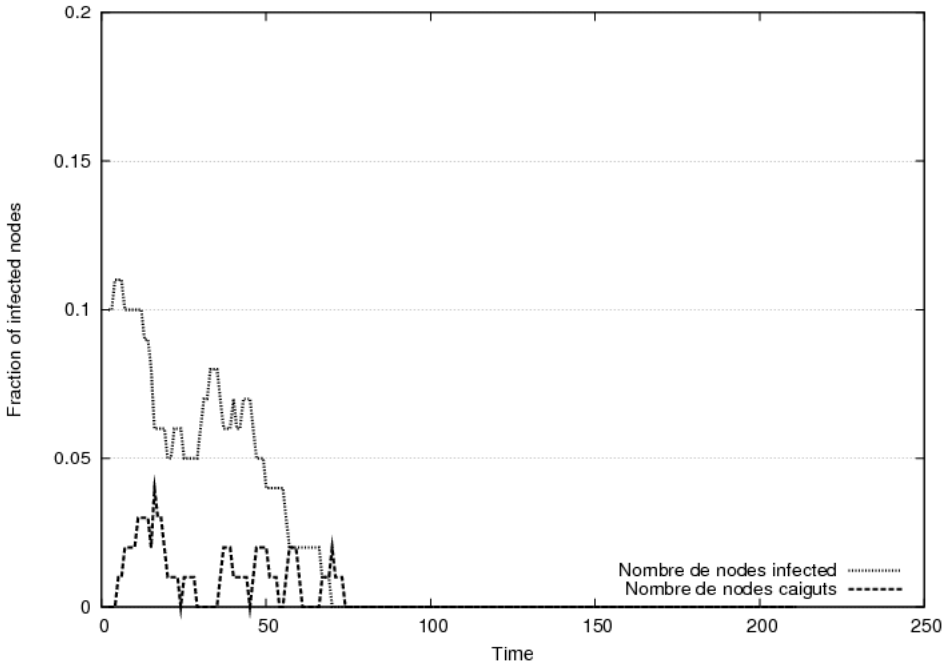

Figura 5.4: Percentatge de nodes infectats al llarg de la simulació

En aquesta segona simulació per a la topologia de 100 nodes, és interessant observar el gràfic de creació i caiguda de connexions 5.5. Es pot observar que la creació de camins és gairebé contínua (exceptuant els primers intervals de temps que la xarxa encara disposa d'un 5-6% de nodes infectats). També es denota la corva de les connexions caigudes, la qual és gairebé despreciable.

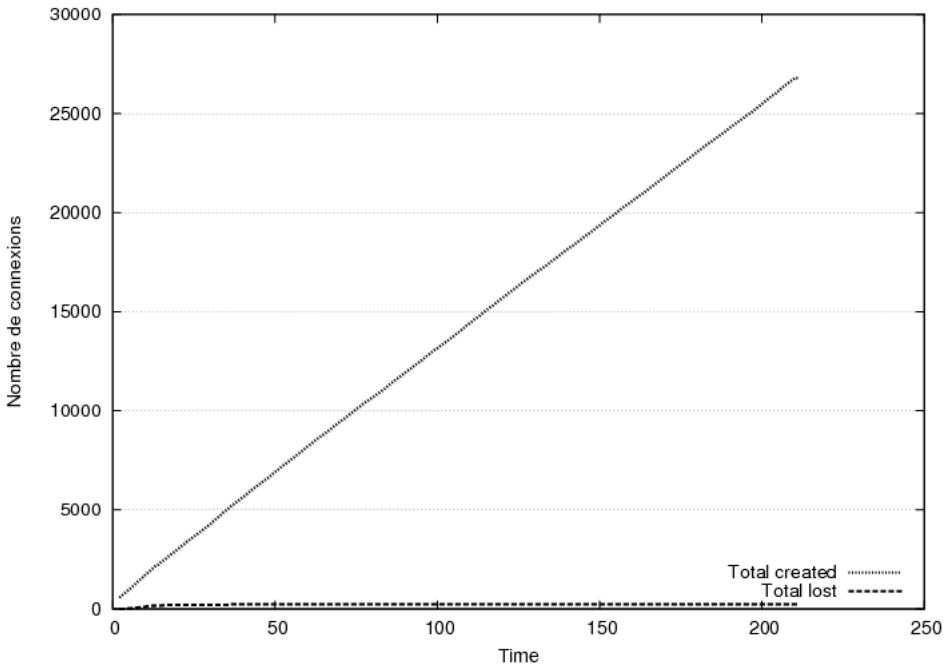

Figura 5.5: Número total de connexions creades respecte el número total de connexions caigudes

Així doncs, amb una topologia de 100 nodes s'ha pogut comprovar l'evolució de la xarxa quan aquesta es cura i quan no.

5.2 Simulació amb una topologia de 200 nodes

Els següents experiments que s'han dut a terme han estat realitzats sobre una topologia de 200 nodes. Aquesta topologia mostra les característiques següents:

- Disposa d'un diàmetre de 12 nodes.
- El grau nodal mínim és 1, el màxim 13 i la mitjana és de 3,430.

En quant al arxiu de tràfic emprat:

- L'origen i el destí de les connexions és aleatori i tots tenen la mateixa propabibilitat d'aparèixer.
- La matriu de tràfic és aleatòria en la proporció relativa i és uniforme. Cap node té mes importància que els altres.

- La mitjana del temps de duració de les connexions és 10.

També cal dir que els resultats obtinguts són la mitja de 5 execucions per resultat.

5.2.1 Primera simulació

La primera simulació executada sobre aquesta topologia disposa del conjunt de paràmetres següents:

- β=0,085
- δ=0,050
- c=0,1
- Un MTTR de l'estat INFECTED per a tots els nodes igual a 2.
- Un MTTR de l'estat CAIGUT per a tots els nodes igual a 5.
- Un percentatge de nodes infectats inicials del 20%.

D'aquesta simulació s'han extret els resultats que es mostren a continuació.

Primerament trobem la Fig. 5.6 on es motra el tant per cent de nodes INFECTATS i CAIGUTS a cada interval de temps de la simulació. En aquest cas, amb els valors donats de β, δ i c la xarxa, un cop infectada amb el 20% dels nodes, no es cura mai i es manté sobre una mitjana del 40% dels nodes infectats i un 10-12% dels nodes caiguts.

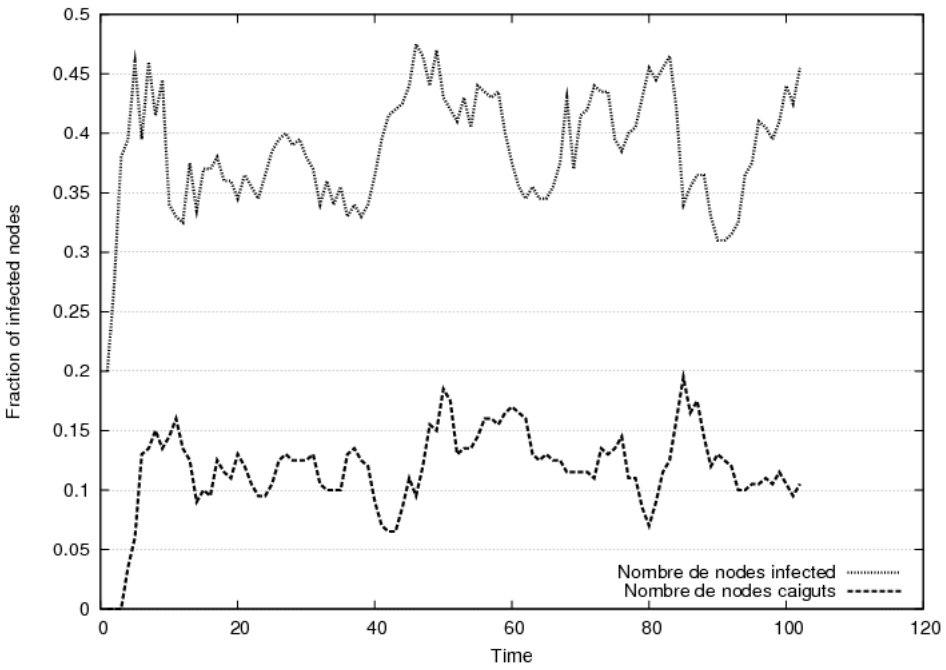

Figura 5.6: Percentatge de nodes infectats al llarg de la simulació

En quant a les connexions creades i caigudes que es mostren a la Fig. 5.7, es pot observar que el comportament que segueixen les dues corves és similar. En aquest cas, igual que a la primera simulació amb la topologia de 100 nodes,el número total de connexions creades a la simulació no arriba al 5%. Aquest fet s'explica amb la Fig. 5.6, de la qual podem extreure que a partir d'un instant de temps al voltant de t=10 el tant per cent de nodes susceptibles (i per tant que poden crear camins) és del 48-50%, valor que resulta ser insuficient per mantenir una mínima connectivitat a la xarxa.

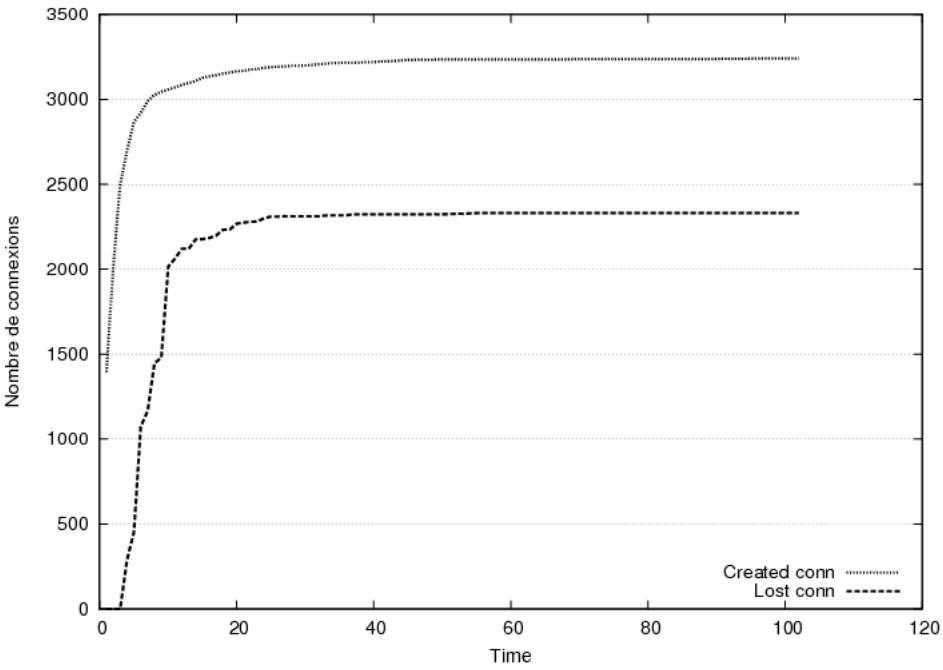

Figura 5.7: Número total de connexions creades respecte el número total de connexions caigudes

5.2.2 Segona simulació

La segona simulació executada sobre aquesta topologia disposa del conjunt de paràmetres següents:

- β=0,051
- δ=0,030
- c=0,1
- Un MTTR de l'estat INFECTED per a tots els nodes igual a 2.
- Un MTTR de l'estat CAIGUT per a tots els nodes igual a 5.
- Un percentatge de nodes infectats inicials del 20%.

D'aquesta simulació s'han extret els resultats que s'observen a continuació.

En la Fig. 5.8 s'observa com el comportament final de la xarxa és similar al de la simulació anterior, amb la diferència que, degut als paràmetres, el tant per cent de nodes infectats s'estabilitza al voltant d'un 10%, mentre que el dels nodes caiguts és d'un 5%.

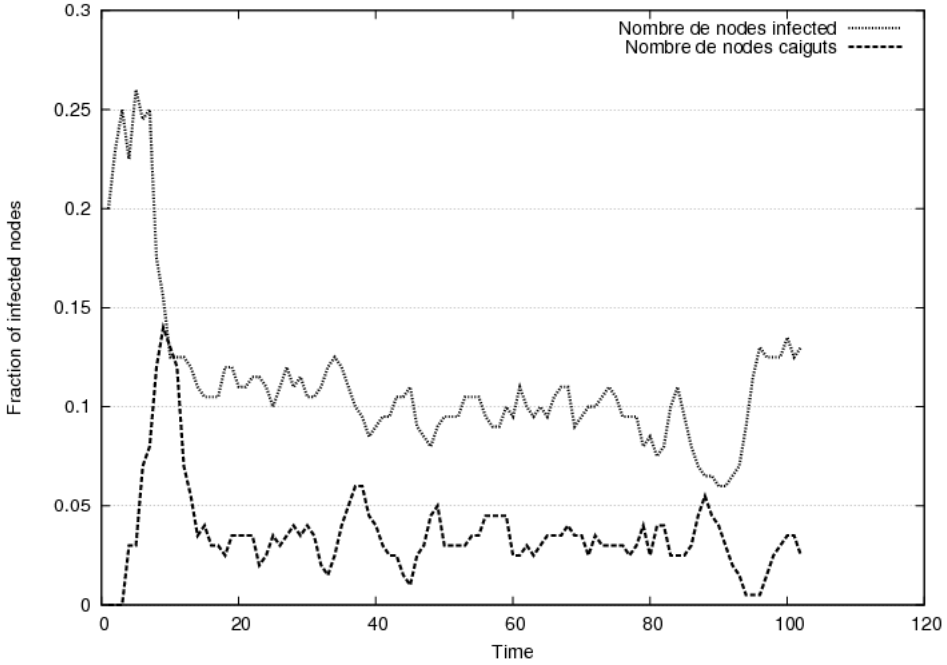

Figura 5.8: Percentatge de nodes infectats al llarg de la simulació

La Fig. 5.9 resulta ser diferent de 5.7 i 5.2 (a on la xarxa tampoc s'acabava de curar mai). El fet que provoca que tot i la xarxa quedar infectada constantment, al fer-ho amb un valor força baix com és un 10& pels nodes infectats i un 5% pels caiguts, permet a la xarxa disposar d'una mínima connectivitat entre els seus nodes, cosa que permet que la creació total de camins sigui d'un 32% del total, valor que difereix del observats a les altres dos figures esmentades. Respecte a les connexions caigudes, s'observa que la corva augmenta lleugerament arribant a un valor final de gairebé una quarta part del total de les connexions creades.

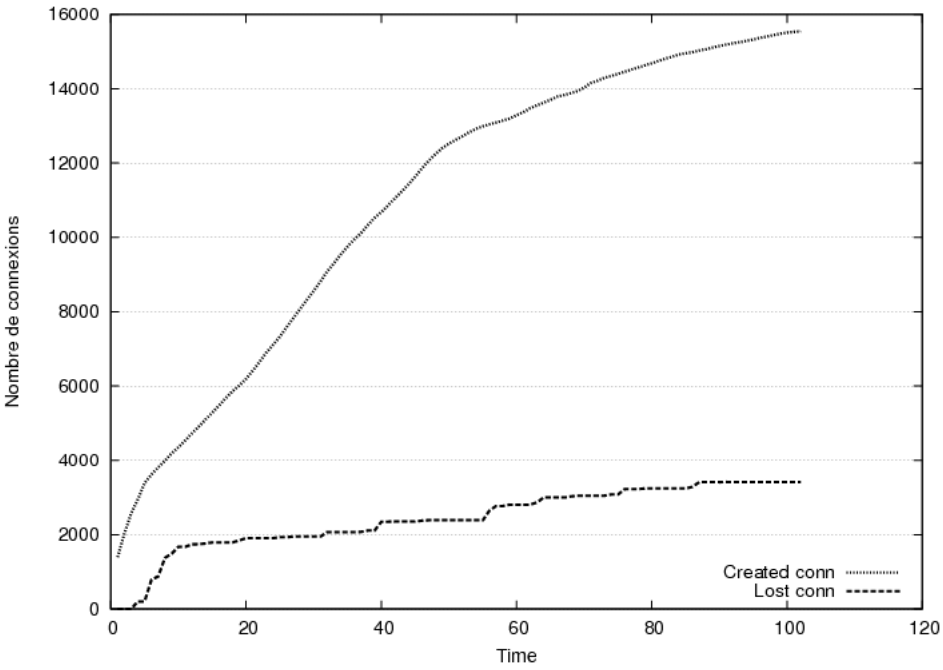

Figura 5.9: Número total de connexions creades respecte el número total de connexions caigudes

Capítol 6

Conclusions

En aquest capítol s'expressen les contribucions que ha donat el desenvolupament d'aquest projecte final de carrera.

A l'inici del treball es va establir una metodologia de programació i planificació a seguir. Es van posar uns certs límits de temps per assolir un conjunt d'objectius, que gràcies a l'ajuda i els ànims tant de tutors com familiars i amics, s'han pogut complir.

Durant el transcurs d'aquest projecte final de carrera, s'han adquirit coneixements de:

- Conceptes i tecnologies de simulació d'esdeveniments discrets.
- Teoria de xarxes òptiques de transport.
- C++: s'ha aprés a llegir codi no escrit per un mateix i a valorar el fet que un codi ho estigui, degut a que s'han hagut de revisar molts arxius de codi font d'OMNet++.
- Java: S'ha après a utilitzar Genèrics.
- Perl
- Shell
- Profundització en coneixements de Linux/Unix
- OMNet++: s'ha llegit tota la documentació d'usuari amb l'objectiu final de poder desenvolupar un projecte mitjançant aquest simulador.

.

Finalment s'ha estat capaç de construir un entorn de simulació de fallades per a xarxes òptiques, i més concretament per a l'àrea de propagació de fallades a una xarxa. Degut a que aquest camp és molt extens i avui en dia tot just comença a ser estudiat, els meus resultats de les simulacions representen només una aplicació de totes les que hi poden arribar a haver en aquesta àrea.

Alhora d'implementar el model SIC s'han trobat forces dificultats per falta de documentació en protocols implementats per OMNet++.

Així doncs, les contribucions del projecte final de carrera al camp de la propagació de fallades a xarxes és la següent:

- Un anàlisi, disseny i implementació d'un model de propagació de fallades per a xarxes òptiques de transport: SIC.
- La implementació d'aquest model en OMNet++, i totes les eines necessàries per fer funcionar aquest correctament.
- Un conjunt de resultats de simulacions, els quals poden ser el punt de partida per a futurs estudis, o simplement una demostració que de que el camp de la recerca en l'àmbit de xarxes pot arribar a ser molt apassionant.

Capítol 7

Treball futur

Alguns problemes que han aparegut mentre avançava el desenvolupament es mostren aquí com a possibles millores per al sistema implementat, ja que o bé per falta de temps no s'han pogut dur a terme o bé no s'han contemplat fins a la finalització del treball. Dividirem aquesta secció en dos parts:

1. **Millores al codi**

Optimització del codi

Una possible tasca a realitzar, seria l'optimització de tot el codi implementat. Alhora de guardar informació necessària s'han fet servir molts contenidors genèrics que, amb una mica més de temps, s'haguéssin pogut convertir tots a arbres de cerca, fent així que la cerca de qualsevol tipus de dada sigués molt més ràpida.

Adaptació del codi per una execució en paral·lel

Degut a que el simulador emprat per a la implementació del sistema SIC suporta una execució paral·lelitzada i distribuïda, una possible futura millora seria l'adaptació del codi implementat per tal que sigui capaç d'executar-se en paral·lel. El motiu que ha fet pensar en aquesta opció ha sigut que, alhora d'executar les simulacions amb topologies grans (més de 100 nodes), la màquina on s'ha dut a terme tot el projecte semblava que es quedés un pèl escassa de

recursos.

Creació dinàmicament de les taules d'encaminament dels nodes i del tràfic referit a una topologia

Pensant en futur, aquest entorn de simulació de fallades per a xarxes òptiques, hauria de contenir el mínim d'eines addicionals possibles tals com el generador de les taules d'encaminament dels nodes a partir de la topologia, o el generador de l'arxiu que conté el tráfic. Per tant, una millora seria la creació d'un mòdul pel sistema que, dinàmicament, al executar ja creés tot el necessàri per al funcionament del sistema.

2. Aspectes de la xarxa

Ampliació de possibles fallades del sistema

En aquest projecte, per raons temporals, només s'han pogut implementar fallades a nodes. Tot i que de manera implícita, si un node falla els enllaços que l'envolten també, una possible millora seria la possibilitat que el sistema permetés definir enllaços que podrien caure sense la necessitat que el node adjacent estigués caigut també.

Protecció i Restauració

També seria interessant implementar un mòdul que fes possible aplicar tècniques de protecció i restauració pel nostre sistema.

Immunització

Un aspecte que aniria molt lligat a la propagació de virus a través d'una xarxa seria la immunització. Es tractaria d'implementar un mòdul capaç de detectar el node per on passen més connexions del sistema i immunitzar-lo d'acord amb diferents tècniques. de manera que el tant per cent de connexions caigues al final de la simulació seria més baix.

Tipus de servei de tràfic

L'entorn de treball, en un futur, seria molt profitós si permetés la definició de diferents tipus de servei de tràfic, és a dir, tràfic amb diferents prioritats.

Routing en entorns d'Epidèmia

Finalment, també es podria arribar a implementar un protocol d'encaminament en entorns afectats per una epidèmia. L'algoritme hauria de garantir una QoS segons la prioritat del tràfic afectat per una fallada.

Bibliografia

[1] E. Mannie, D. Papadimitriou. Recovery **Protection and Restoration** Terminology for Generalized Multi-Protocol Label Switching (GMPLS). IETF, Març 2006. RFC 4427.

[2] ITU-T G.709, **Interfaces for the Optical Transport Network** , March 2003.

[3] ITU-T G.872, **Architecture of Optical Transport Networks** , November 2001.

[4] B. Davie and Y. Rekhter. **MPLS Technology and Applcations** Morgan kaufmann publisher Inc. ISBN 1-55860-656-4, Maig 2000.

[5] D. Oran. OSI **IS-IS Intra-domain Routing Protocol** RFC 1142, Febrer 1990.

[6] Moy, J., **OSPF Version 2** , IETF RFC 2328, Abril 1998.

[7] R. Braden, L. Zhang, S. Berson, S. Herzog, and S.Jamin. **Resource ReSerVation Protocol (RSVP)** IETF RFC

[8] Daniel O. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. **RSVP-TE: Extensions to RSVP for LSP Tunnels**. IETF RFC 3209, Desembre 2001. 2205, Septembre 1997.

[9] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas. **LDP specification**. IETF RFC 3036, January 2001.

[10] Chakrabarti, D., Wang, Y., Wang, C., Leskovec, J., and Faloutsos, C. 2008. **Epidemic thresholds in real networks** ACM Trans. Inform. Syst. Secur. 10, 4, Article 13 (January 2008), 26 pages. DOI = 10.1145/1284680.1284681 http://doi.acm.org/ 10.1145/1284680.1284681

[11] V. Sharma, B. M. Crane, S. Makam, K. Owens,C. Huang, F. Hellstrand, J. Weil, L. Andersson, B. Jamoussi, B. Cain, S. Civanlar, A. Chiu. **Framework for MPLS-Based Recovery** RFC3469. Febrer 2003.

[12] Changcheng Huang, Vishal Sharma, Ken Owens, Srinivas Makam, **Building reliable MPLS Networks using a path protection mechanism** , IEEE Communications Magazine, Març 2002

[13] Hernest H. Page, Jr. **Simulation Modeling Methodolgy: Principles and Etiology of Decision Support** Blacksburg, Virginia 1994.

[14] Shannon, R.E **Systems Simulation: The Art and Science** Prentice-Hall, Englewood Cliffs, NJ 1975.

[15] Nance, R.E. **Proceedings of the Second ACM SIGPLAN History of Programming Languages Conference** Cambridge (1993), MA, April 20-23, Reprinted in ACM SIGPLAN Notices, 28(3), pp. 149-175.

[16] Nance, R.E. **The Time and State Relationships in Simulation Modeling** Cambridge (1993), Communications of the ACM, 24(4), pp. 173-179, April 1981

[17] G. Di CAro **Analysis of simulation environments for mobile ad hoc networks** Technical Report No. IDSIA-24-03. Dalle Molle Institute for Artificial Intelligence, December 2003

[18] OPNET (Juny 2009). **http://www.opnet.com** Juny 2009.

[19] The network simulator ns-2 (Juny 2009). **http://www.isi.edu/nsnam/ns/**

[20] S. Gadde, J. Chase, and A. Vahdat. **Coarse-Grained Network Simulation for Wide-Area Distributed Systems** in Proceedings of Communication Networks and Distributed Systems Modeling and Simulation, Gener 2002

[21] OMNET++ (Juny 2009). **http://www.omnetpp.org** Juny 2009.

[22] OMNET++ User Manual (Juny 2009). **http://www.omnetpp.org/doc/manual /usman.html** Juny 2009

[23] GUADALL, C. **Incorporació de Qualitat de Servei a l'algorisme d'encaminament AntNet** Septembre 2004

[24] R. Fujimoto,et al. **Large-Scale Network Simulation: How Big? How Fast** In Proceedings of the 11TH IEEE/ACM International Symposium on Modeling, Analysis and aggressive use of distributed simulations to model Simulation of Computer Telecommunications Systems abstraction simplifications. (MASCOTS'03), 2003.

[25] A. Kröller, et al. **Shawn: A new approach to simulating wireless sensor networks** , presentat al Design, Analysis, and simulation of Distributed Systems 2005 (SpringSim) Febrer 2005

[26] INET framework, **http://inet.omnetpp.org/**

[27] INET framework, **http://inet.omnetpp.org/doc/INET/neddoc/index.html**

[28] OMNet++, Tic Toc Tutorial **http://www.omnetpp.org/doc/omnetpp40/tictoc-tutorial/**

[29] BRITE, **http://www.cs.bu.edu/brite/**

[30] gentraf, **www.vtpi.org/gentraf.pdf**