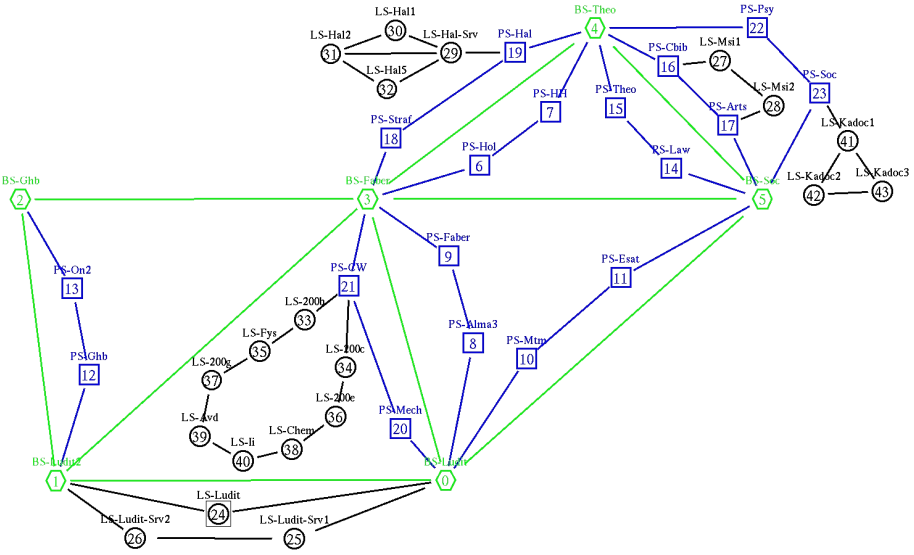
Netwerksimulatie – Verslag 3

Philippe Dellaert
3e Bachelor Computerwetenschappen-Elektrotechniek

Wouter Van Ranst
3e Bachelor Computerwetenschappen-Bedrijfsbeheer

16 mei 2007

1 Vereenvoudigde KULeuvenNet topologie

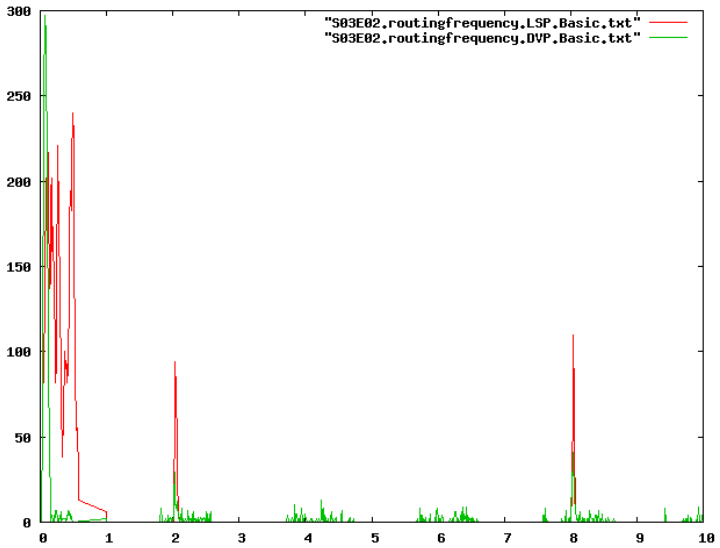


Figuur 1: Vereenvoudigde KULeuvenNet topologie.

In figuur 1 wordt de door ons gemoddeleerde vereenvoudige topologie afgebeeld. Deze is bekomen door de in de opgave vermelde nodes correct met elkaar te verbinden en in `nam` een beetje te slepen met de nodes. We hebben daarnaast ook de backbone nodes groen gekleurd en als zeshoek voorgesteld. De periferie-nodes hebben we in het blauw gekleurd en als vierkanten afgebeeld. De lokale nodes zijn gewone cirkels en in het zwart.

2 Routingfrequentie plots

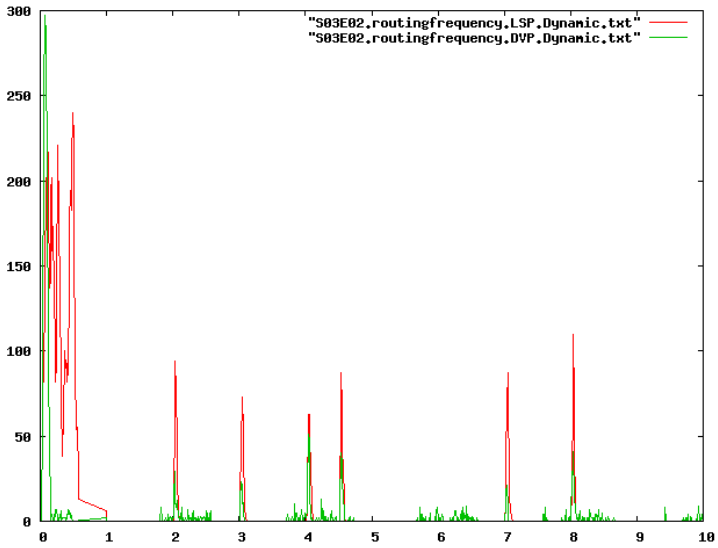
2.1 Basis plot



Figuur 2: Aantal routerings pakketten bij de basis opgave.

In figuur 2 wordt het aantal routerings pakketten voorgesteld in functie van de tijd voor zowel `link state protocol` en `distance vector protocol`. Na twee seconden wordt de link tussen de backbone switch van faber en de backbone switch van sociologie verbroken. Na acht seconden wordt deze link terug geactiveerd. Het valt dadelijk op dat bij het `link state protocol` de initiële piek minder hoog is dan bij het `distance vector protocol`. Echter wordt er wel een langere periode veel routerings pakketten rondgestuurd. Bij het uitvallen van een link zien we bij beide protocolen een piek verschijnen, hier is de piek van het `link state protocol` wel hoger dan die van het `distance vector protocol`. Als laatste valt het op dat het `distance vector protocol` regelmatig nog routering info doorstuurt terwijl het `link state protocol` dit enkel doet op momenten dat het nodig is.

2.2 Dynamisch plot



Figuur 3: Aantal routerings pakketten bij een uitbreiding op de opgave met meer dynamische links.

In figuur 3 wordt het aantal routerings pakketten voorgesteld wanneer er wat meer dynamische links zijn in het netwerk. Nog steeds wordt de link tussen de backbone switch van faber en de backbone switch van socioligie verbroken na twee seconden en terug geactiveerd na acht seconden. Daarnaast wordt nu ook de link tussen de backbone switch van sociologie en de periferie switch van rechten verbroken na vier seconden en terug geactiveerd na zeven seconden. Als laatste wordt ook de link tussen de tweede backbone switch van ludit en de backbone switch van faber verbroken na drie seconden en opnieuw geactiveerd na 4,5 seconde. Wat op deze grafiek duidelijk zichtbaar is, is dat er ook hier weer grote pieken zijn bij het `link state protocol` op het moment dat een link verbroken of opnieuw geactiveerd wordt. Daarnaast zijn de pieken van het `distance vector protocol` ook hier weer lager. Het laatste dat opvalt is dat het `distance vector protocol` nog steeds met dezelfde regelmaat routerings pakketten blijft rondsturen, het aantal onderbroken links heeft hier dus geen invloed op, buiten de eerder vermelde pieken.

3 link state protocol

Bij het `link state protocol` wordt er in elke node een map aangemaakt van het netwerk. Elke node weet dus exact waar in het netwerk een andere node aanwezig is en hoe die te bereiken is. Om deze map aan te maken wordt er gebruik gemaakt van een *flooding* methode. Dit gebeurt in twee belangrijke stappen.

In eerste instantie zal elke node controleren welke nodes aan zichzelf verbonden zijn. Eenmaal deze informatie is verwerkt, zal elke node naar elk van zijn buren een boodschap sturen die de *link-state advertisement* wordt genoemd. Deze boodschap bestaat uit een identificatie gedeelte vanuit welke node het komt, identificatie van alle verbonden nodes en sequentie nummer dat verhoogt elke keer de node de boodschap uitstuurt. Als deze boodschap toekomt op een buur node, zal deze tweede node dit doorsturen naar alle nodes die hieraan verbonden zijn, behalve de node waarvan de boodschap komt. De tweede node zal dan bijhouden dat het die bepaalde boodschap al heeft doorgestuurd, dus als die nog eens toekomt, wordt die niet meer naar iedereen doorgestuurd. Zo zal de boodschap zich verspreiden over het hele netwerk zonder oneindig door te gaan.

Eenmaal alle nodes hun boodschap hebben doorgestuurd naar hun buren, en al deze boodschappen zich verspreid hebben over het netwerk, stopt het versturen van boodschappen en wordt er geen routering info meer verstuurd. Elke node stelt nu aan de hand van de binnengekomen boodschappen een map op van het netwerk en weet op die manier hoe een andere node te bereiken is.

Als er een link uitvalt tussen twee nodes, zullen beide nodes opnieuw hun boodschap doorsturen naar iedere buur. Aangezien de sequentie nummer verhoogt is, sturen die buren dat ook weer door, zodat wederom heel het netwerk op de hoogte gesteld wordt van de veranderingen in de links.

Deze uitleg verklaart ook waarom er gedurende een periode van bijna een seconde een hoop routerings pakketten zijn bij het `link state protocol`. Alle boodschappen moeten zich verspreiden over het hele netwerk, en dat kan soms even duren. Ook verklaart de uitleg de hoge pieken wanneer er een link verbroken wordt, aangezien wederom de boodschappen van de nodes waartussen de link verbroken is doorheen het netwerk verspreid moeten worden.

4 distance vector protocol

Bij het `distance vector protocol` wordt er berekend via een algoritme, dat afhangt van welke implementatie men gebruikt, wat het snelste pad is tussen twee nodes. Het algoritme berekent dan een kost voor elke link tussen de verschillende nodes en kan rekening houden met afstand, bandbreedte, ingestelde waarden, etc. Hoe hoger de kost van een link, hoe minder kans dat deze gebruikt zal worden.

Één van de belangrijke verschillen met het `link state protocol` is dat

bij het `distance vector protocol` elke node periodisch informatie doorstuurt naar de verbonden buren. Daardoor kunnen de buren hun informatie ook aanpassen en eventuele veranderingen doorsturen naar hun eigen buren. Op die manier wordt enkel de nodige informatie doorgestuurd in plaats van alle informatie. In het begin zorgt dit wel voor een grotere piek aangezien er veel informatie tegelijk verspreid wordt, maar deze piek is van kortere duur dan bij het `link state protocol`.

Als er een verandering is in de status van een link, zullen de verbonden nodes direct deze informatie doorsturen naar hun buren en de routerings tabellen zullen eventueel aangepast worden om andere routes te gebruiken.

5 Vergelijking

Uiteraard zijn er grote verschillen tussen onze gesimuleerde versie en de werkelijkheid. Een van de grote verschillen is natuurlijk het verschil in de links. In onze simulatie is alles 10Mb met een delay van 10ms. In de werkelijkheid zijn de backbone links 1Gb en hangt de delay van verschillende factoren af. Ook de periferie links en de lokale links verschillen. Dit zorgt er natuurlijk voor dat de routering van het verkeer helemaal anders kan zijn aangezien sommige links misschien veel meer geprefereerd worden als andere.

Daarnaast zijn er in de werkelijkheid ook meer dan 500 vlan's. Al deze vlan's zorgen voor een extra overhead die het hele netwerk zal beinvloeden. Deze zijn echter wel nodig om de verschillende onderdelen van de KULeuven van elkaar af te schermen.

Als laatste grote verschil is er het verkeer. Wij hebben in onze simulatie één FTP verbinding aangemaakt en gebruikt. In werkelijkheid is het netwerk veel zwaarder belast met duizenden connecties van verschillende aard. Zowel UDP als TCP verbindingen worden constant aangemaakt of verbroken. Dit verschil hangt natuurlijk ook samen met het verschil in capaciteit. Als we dit zouden proberen te implementeren in de netwerk simulator, dan zou onze trace file ongeloofelijk groot worden wegens de hoeveelheid data die er door het netwerk gaat op zo'n korte periode.

6 Voorstel

Op het moment zijn er enkele problemen met de huidige opstelling van KULeuvenNet. Alle routering gebeurt in de core en dit zorgt voor twee problemen. Allereerst moeten daardoor alle vlans ook door de core gaan omdat anders hun routering niet werkt. Daarnaast worden pc's ook krachtiger en krachtiger, waardoor het gevaarlijk kan zijn als een ongewenst programma ineens 100 000'den pakketten per seconden naar een router begint te sturen. De routers zijn namelijk verre van krachtig genoeg om deze hoeveelheid te kunnen verwerken en zullen daardoor in problemen komen.

Daarom heeft Herman Moons een nieuw voorstel gedaan voor een nieuwe

topologie van het netwerk. Hierbij zouden de periferie L2-switchen vervangen woorden door L3-switchen waarbij dus ook al routering gebeurt. Daardoor wordt de zware belasting weggehaald van de core L3-switchen. De vlan's kunnen dan ook door de periferie switchen worden verwerkt. En een laatste gevolg, als een ongewenst programma probeert om een heel grote hoeveelheid pakketten te sturen naar een L3-switch, zal enkel een deel van de periferie onbereikbaar worden, aangezien er enkel een periferie switch wordt aangevallen. De core zelf ondervindt hiervan geen problemen en kan blijven functioneren. Het probleem doet zich dus voor op een veel kleinere schaal dan nu het geval zou zijn.

Als laaste zouden ook de links in de core, die nu 1Gbps zijn, vervangen worden door 10Gbps links. Dit zorgt ervoor dat het netwerk genoeg capaciteit heeft voor de toekomst.

A Code

```
#########################################
# EXERCISE SETUP                        #
#########################################

# Basename for the generated files
set filenameBase "S03E02"

# Setting type of routing
# link state protocol
#set routingType "lsp"
# distance vector protocol
set routingType "dvp"

# More dynamics
set dynamicNetwork "1"

#########################################
# SIMULATOR STARTUP                     #
#########################################

# Create simulator
set ns [new Simulator]

#########################################
# ROUTING SETUP                         #
#########################################

# Setting routing protocol
if { $routingType == "lsp" } {
        $ns rtproto LS
} elseif { $routingType == "dvp" } {
        $ns rtproto DV
}

#########################################
# COLOR SETUP                           #
#########################################

$ns color 0 Red

#########################################
# FILES                                 #
#########################################

# Trace file
set tf [open $filenameBase.$routingType.dynamic.$dynamicNetwork.out.tr w]
$ns trace-all $tf
```

```
# Nam tracefile
set nf [open $filenameBase.$routingType.dynamic.$dynamicNetwork.out.nam w]
$ns namtrace-all $nf

########################################
# NODE SETUP                           #
########################################

# Backbone nodes
set bsludit     [$ns node]
set bsludit2    [$ns node]
set bsghb       [$ns node]
set bsfaber     [$ns node]
set bstheo      [$ns node]
set bssoc       [$ns node]

# Periphery nodes
set pshol       [$ns node]
set pshh        [$ns node]
set psalma3     [$ns node]
set psfaber     [$ns node]
set psmtm       [$ns node]
set psesat      [$ns node]
set psghb       [$ns node]
set pson2       [$ns node]
set pslaw       [$ns node]
set pstheo      [$ns node]
set pscbib      [$ns node]
set psarts      [$ns node]
set psstraf     [$ns node]
set pshal       [$ns node]
set psmech      [$ns node]
set pscw        [$ns node]
set pspsy       [$ns node]
set pssoc       [$ns node]

# Local nodes
set lsludit     [$ns node]
set lsluditsrv1 [$ns node]
set lsluditsrv2 [$ns node]
set lsmsi1      [$ns node]
set lsmsi2      [$ns node]
set lshalsrv    [$ns node]
set lshal1      [$ns node]
set lshal2      [$ns node]
set lshal5      [$ns node]
set ls200b      [$ns node]
set ls200c      [$ns node]
set lsfys       [$ns node]
set ls200e      [$ns node]
```

```
set ls200g      [$ns node]
set lschem      [$ns node]
set lsavd       [$ns node]
set lsii        [$ns node]
set lskadoc1    [$ns node]
set lskadoc2    [$ns node]
set lskadoc3    [$ns node]

# Creating backbone links
$ns duplex-link $bsludit        $bsludit2       10Mb 10ms DropTail
$ns duplex-link $bsludit        $bssoc          10Mb 10ms DropTail
$ns duplex-link $bsludit        $bsfaber        10Mb 10ms DropTail
$ns duplex-link $bsludit2       $bsfaber        10Mb 10ms DropTail
$ns duplex-link $bsludit2       $bsghb          10Mb 10ms DropTail
$ns duplex-link $bsghb          $bsfaber        10Mb 10ms DropTail
$ns duplex-link $bsfaber        $bstheo         10Mb 10ms DropTail
$ns duplex-link $bstheo         $bsfaber        10Mb 10ms DropTail
$ns duplex-link $bsfaber        $bssoc          10Mb 10ms DropTail
$ns duplex-link $bstheo         $bssoc          10Mb 10ms DropTail
$ns duplex-link $bssoc          $bstheo         10Mb 10ms DropTail

# Creating periphery links
$ns duplex-link $pshol          $bsfaber        10Mb 10ms DropTail
$ns duplex-link $pshol          $pshh           10Mb 10ms DropTail
$ns duplex-link $pshh           $bstheo         10Mb 10ms DropTail
$ns duplex-link $psalma3        $bsludit        10Mb 10ms DropTail
$ns duplex-link $psalma3        $psfaber        10Mb 10ms DropTail
$ns duplex-link $psfaber        $bsfaber        10Mb 10ms DropTail
$ns duplex-link $psmtm          $bsludit        10Mb 10ms DropTail
$ns duplex-link $psmtm          $psesat         10Mb 10ms DropTail
$ns duplex-link $psesat         $bssoc          10Mb 10ms DropTail
$ns duplex-link $psghb          $bsludit2       10Mb 10ms DropTail
$ns duplex-link $psghb          $pson2          10Mb 10ms DropTail
$ns duplex-link $pson2          $bsghb          10Mb 10ms DropTail
$ns duplex-link $pslaw          $bssoc          10Mb 10ms DropTail
$ns duplex-link $pslaw          $pstheo         10Mb 10ms DropTail
$ns duplex-link $pstheo         $bstheo         10Mb 10ms DropTail
$ns duplex-link $psarts         $bssoc          10Mb 10ms DropTail
$ns duplex-link $psarts         $pscbib         10Mb 10ms DropTail
$ns duplex-link $pscbib         $bstheo         10Mb 10ms DropTail
$ns duplex-link $psstraf        $bsfaber        10Mb 10ms DropTail
$ns duplex-link $psstraf        $pshal          10Mb 10ms DropTail
$ns duplex-link $bstheo         $pshal          10Mb 10ms DropTail
$ns duplex-link $psmech         $bsludit        10Mb 10ms DropTail
$ns duplex-link $psmech         $pscw           10Mb 10ms DropTail
$ns duplex-link $bsfaber        $pscw           10Mb 10ms DropTail
$ns duplex-link $pspsy          $bstheo         10Mb 10ms DropTail
$ns duplex-link $pspsy          $pssoc          10Mb 10ms DropTail
$ns duplex-link $bssoc          $pssoc          10Mb 10ms DropTail
```

```
# Local links
$ns duplex-link $lsludit        $bsludit        10Mb 10ms DropTail
$ns duplex-link $lsludit        $bsludit2       10Mb 10ms DropTail
$ns duplex-link $lsluditsrv1    $bsludit        10Mb 10ms DropTail
$ns duplex-link $lsluditsrv1    $lsluditsrv2    10Mb 10ms DropTail
$ns duplex-link $lsluditsrv2    $bsludit2       10Mb 10ms DropTail
$ns duplex-link $lshalsrv       $pshal          10Mb 10ms DropTail
$ns duplex-link $lshalsrv       $lshal1         10Mb 10ms DropTail
$ns duplex-link $lshalsrv       $lshal2         10Mb 10ms DropTail
$ns duplex-link $lshalsrv       $lshal5         10Mb 10ms DropTail
$ns duplex-link $lshal1         $lshal2         10Mb 10ms DropTail
$ns duplex-link $lshal5         $lshal2         10Mb 10ms DropTail
$ns duplex-link $pscw           $ls200c         10Mb 10ms DropTail
$ns duplex-link $ls200c         $ls200e         10Mb 10ms DropTail
$ns duplex-link $ls200e         $lschem         10Mb 10ms DropTail
$ns duplex-link $lschem         $lsii           10Mb 10ms DropTail
$ns duplex-link $lsii           $lsavd          10Mb 10ms DropTail
$ns duplex-link $lsavd          $ls200g         10Mb 10ms DropTail
$ns duplex-link $ls200g         $lsfys          10Mb 10ms DropTail
$ns duplex-link $lsfys          $ls200b         10Mb 10ms DropTail
$ns duplex-link $ls200b         $pscw           10Mb 10ms DropTail
$ns duplex-link $pssoc          $lskadoc1       10Mb 10ms DropTail
$ns duplex-link $lskadoc1       $lskadoc2       10Mb 10ms DropTail
$ns duplex-link $lskadoc2       $lskadoc3       10Mb 10ms DropTail
$ns duplex-link $lskadoc3       $lskadoc1       10Mb 10ms DropTail
$ns duplex-link $psarts         $lsmsi2         10Mb 10ms DropTail
$ns duplex-link $lsmsi2         $lsmsi1         10Mb 10ms DropTail
$ns duplex-link $lsmsi1         $pscbib         10Mb 10ms DropTail

#########################################
# NAM SETUP                             #
#########################################

# Backbone node labels
$bsludit label BS-Ludit
$bsludit2 label BS-Ludit2
$bsghb label BS-Ghb
$bsfaber label BS-Faber
$bstheo label BS-Theo
$bssoc label BS-Soc

# Backbone node colors
$bsludit color Green
$bsludit2 color Green
$bsghb  color Green
$bsfaber  color Green
$bstheo  color Green
$bssoc   color Green

# Backbone node shapes
```

```
$bsludit shape hexagon
$bsludit2 shape hexagon
$bsghb shape hexagon
$bsfaber shape hexagon
$bstheo shape hexagon
$bssoc shape hexagon

# Backbone link colors
$ns duplex-link-op $bsludit $bsludit2 color Green
$ns duplex-link-op $bsludit $bssoc  color Green
$ns duplex-link-op $bsludit $bsfaber color Green
$ns duplex-link-op $bsludit2 $bsfaber color Green
$ns duplex-link-op $bsludit2 $bsghb  color Green
$ns duplex-link-op $bsghb $bsfaber color Green
$ns duplex-link-op $bsfaber $bstheo  color Green
$ns duplex-link-op $bstheo  $bsfaber  color Green
$ns duplex-link-op $bsfaber  $bssoc  color Green
$ns duplex-link-op $bstheo $bssoc  color Green
$ns duplex-link-op $bssoc  $bstheo  color Green

# Periphery node labels
$pshol  label PS-Hol
$pshh label PS-HH
$psalma3 label PS-Alma3
$psfaber label PS-Faber
$psmtm label PS-Mtm
$psesat label PS-Esat
$psghb label PS-Ghb
$pson2 label PS-On2
$pslaw label PS-Law
$pstheo label PS-Theo
$pscbib label PS-Cbib
$psarts label PS-Arts
$psstraf label PS-Straf
$pshal label PS-Hal
$psmech label PS-Mech
$pscw label PS-CW
$pspsy label PS-Psy
$pssoc label PS-Soc

# Periphery node colors
$pshol color Blue
$pshh color Blue
$psalma3 color Blue
$psfaber color Blue
$psmtm color Blue
$psesat color Blue
$psghb color Blue
$pson2 color Blue
$pslaw color Blue
```

```
$pstheo color Blue
$pscbib color Blue
$psarts color Blue
$psstraf color Blue
$pshal color Blue
$psmech color Blue
$pscw color Blue
$pspsy color Blue
$pssoc color Blue

# Periphery node shapes
$pshol  shape square
$pshh  shape square
$psalma3  shape square
$psfaber  shape square
$psmtm  shape square
$psesat  shape square
$psghb  shape square
$pson2  shape square
$pslaw  shape square
$pstheo  shape square
$pscbib  shape square
$psarts  shape square
$psstraf  shape square
$pshal  shape square
$psmech  shape square
$pscw  shape square
$pspsy  shape square
$pssoc  shape square

# Periphery link colors
$ns duplex-link-op $pshol  $bsfaber  color Blue
$ns duplex-link-op $pshol  $pshh  color Blue
$ns duplex-link-op $pshh  $bstheo  color Blue
$ns duplex-link-op $psalma3  $bsludit  color Blue
$ns duplex-link-op $psalma3  $psfaber  color Blue
$ns duplex-link-op $psfaber  $bsfaber  color Blue
$ns duplex-link-op $psmtm  $bsludit  color Blue
$ns duplex-link-op $psmtm  $psesat  color Blue
$ns duplex-link-op $psesat  $bssoc  color Blue
$ns duplex-link-op $psghb  $bsludit2  color Blue
$ns duplex-link-op $psghb  $pson2  color Blue
$ns duplex-link-op $pson2  $bsghb  color Blue
$ns duplex-link-op $pslaw  $bssoc  color Blue
$ns duplex-link-op $pslaw  $pstheo  color Blue
$ns duplex-link-op $pstheo  $bstheo  color Blue
$ns duplex-link-op $psarts  $bssoc  color Blue
$ns duplex-link-op $psarts  $pscbib  color Blue
$ns duplex-link-op $pscbib  $bstheo  color Blue
$ns duplex-link-op $psstraf  $bsfaber  color Blue
```

```
$ns duplex-link-op $psstraf  $pshal  color Blue
$ns duplex-link-op $bstheo  $pshal  color Blue
$ns duplex-link-op $psmech  $bsludit  color Blue
$ns duplex-link-op $psmech  $pscw  color Blue
$ns duplex-link-op $bsfaber  $pscw  color Blue
$ns duplex-link-op $pspsy  $bstheo  color Blue
$ns duplex-link-op $pspsy  $pssoc  color Blue
$ns duplex-link-op $bssoc  $pssoc  color Blue

# Local node labels
$lsludit label LS-Ludit
$lsluditsrv1 label LS-Ludit-Srv1
$lsluditsrv2 label LS-Ludit-Srv2
$lsmsi1  label LS-Msi1
$lsmsi2  label LS-Msi2
$lshalsrv label LS-Hal-Srv
$lshal1 label LS-Hal1
$lshal2 label LS-Hal2
$lshal5 label LS-Hal5
$ls200b label LS-200b
$ls200c label LS-200c
$lsfys label LS-Fys
$ls200e label LS-200e
$ls200g label LS-200g
$lschem label LS-Chem
$lsavd label LS-Avd
$lsii label LS-Ii
$lskadoc1 label LS-Kadoc1
$lskadoc2 label LS-Kadoc2
$lskadoc3 label LS-Kadoc3

# Local node colors
$lsludit color Black
$lsluditsrv1 color Black
$lsluditsrv2 color Black
$lsmsi1  color Black
$lsmsi2  color Black
$lshalsrv color Black
$lshal1 color Black
$lshal2 color Black
$lshal5 color Black
$ls200b color Black
$ls200c color Black
$lsfys color Black
$ls200e color Black
$ls200g color Black
$lschem color Black
$lsavd color Black
$lsii color Black
$lskadoc1 color Black
```

```
$lskadoc2 color Black
$lskadoc3 color Black

# Local link colors
$ns duplex-link-op $lsludit  $bsludit  color Black
$ns duplex-link-op $lsludit  $bsludit2  color Black
$ns duplex-link-op $lsluditsrv1 $bsludit  color Black
$ns duplex-link-op $lsluditsrv1 $lsluditsrv2  color Black
$ns duplex-link-op $lsluditsrv2 $bsludit2  color Black
$ns duplex-link-op $lshalsrv  $pshal  color Black
$ns duplex-link-op $lshalsrv  $lshal1  color Black
$ns duplex-link-op $lshalsrv  $lshal2  color Black
$ns duplex-link-op $lshalsrv  $lshal5  color Black
$ns duplex-link-op $lshal1  $lshal2  color Black
$ns duplex-link-op $lshal5  $lshal2  color Black
$ns duplex-link-op $pscw  $ls200c  color Black
$ns duplex-link-op $ls200c  $ls200e color Black
$ns duplex-link-op $ls200e  $lschem color Black
$ns duplex-link-op $lschem  $lsii color Black
$ns duplex-link-op $lsii  $lsavd color Black
$ns duplex-link-op $lsavd  $ls200g color Black
$ns duplex-link-op $ls200g  $lsfys color Black
$ns duplex-link-op $lsfys  $ls200b color Black
$ns duplex-link-op $ls200b  $pscw color Black
$ns duplex-link-op $pssoc  $lskadoc1  color Black
$ns duplex-link-op $lskadoc1  $lskadoc2  color Black
$ns duplex-link-op $lskadoc2  $lskadoc3  color Black
$ns duplex-link-op $lskadoc3  $lskadoc1  color Black
$ns duplex-link-op $psarts  $lsmsi2  color Black
$ns duplex-link-op $lsmsi2  $lsmsi1  color Black
$ns duplex-link-op $lsmsi1  $pscbib  color Black

#########################################
# FTP CONNECTION BETWEEN CW AND LAW   #
#########################################

# Starting TCP Agent
set tcp [new Agent/TCP]
#set tcp [new Agent/TCP/Newreno]
#set tco [new Agent/TCP/Vegas]
$ns attach-agent $pscw $tcp

# Starting Sink Agent
set sink [new Agent/TCPSink]
$ns attach-agent $pslaw $sink

# Connecting TCP agent to Sink agent
$ns connect $tcp $sink

# Setting connection parameters
```

```
$tcp set fid_ 0
#$tcp set packetSize_
#$tcp set window_
#$tcp set rate_

# Starting FTP Connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp

#######################################
# BREAKING LINKS                      #
#######################################

# Connection between Sociology and Faber
$ns rtmodel-at 2.0 down $bsfaber $bssoc
$ns rtmodel-at 8.0 up $bsfaber $bssoc

if { $dynamicNetwork == "1" } {
        # Connection between Sociology and Law
        $ns rtmodel-at 4.0 down $bssoc $pslaw
        $ns rtmodel-at 7.0 up $bssoc $pslaw
        #Connection between BS Faber, BS ludit2
        $ns rtmodel-at 3.0 down $bsludit2 $bsfaber
        $ns rtmodel-at 4.5 up $bsludit2 $bsfaber
}

#######################################
# TIMING                              #
#######################################

# FTP Connection timing
$ns at 1.0 "$ftp start"
$ns at 9.9 "$ftp stop"

# End of simulation
$ns at 10.0 "finish"

#######################################
# FUNCTIONS                           #
#######################################

# Finish function
proc finish {} {
        #finalize trace files
        global ns nf tf filenameBase routingType dynamicNetwork
        $ns flush-trace
        close $tf
        close $nf

        #exec nam $filenameBase.$routingType.dynamic.$dynamicNetwork.out.nam &
```

```
        exit 0
}

#########################################
# RUNNING SIMULATOR                     #
#########################################

# Running the network simulator
$ns run
```