سلسلة تعـــلم بســـهولة

Network Simulator (NS-2)

الجزء الأول

تأليف
أمجد محمد عز الدين عبد اللطيف

سلسلة
تعلم بسهولة

محاكي الشبكات NS2

الجزء الأول

تأليف
أمجد محمد عزالدين عبداللطيف

الحمد لله رب العالمين والصلاة والسلام على أشرف المرسلين سيدنا محمد وعلى آله وصحبه وسلم أما بعد .

أولا أحمد الله أن يسر لي كتابة هذه السلسلة والذي أرجو من الله أن ينفع بها المسلمين.

تقوم هذه السلسلة بشرحه بطريقة تسهل تعلمه وتشرح أساسياته ومفاهيمه ، والتي أسأل الله بها أن تكون عوناً للمهتمين في هذا المجال وأن تكون حلاً لمشكلتهم ورداً على أسئلتهم ، فأرجو من أي شخص يملك معلومة عن هذه المحاكي أن لا يبخل بها علي لكي أضيفها في هذه السلسلة، حتى تعم الفائدة، فأسأل الله أن يوفقني في كتابتها وأن يذكرني فيها ما نسيت وأن يعلمني فيها ما جهلت وما أصبت فمن الله وما أخطأت فمن نفسي والشيطان .... واللهم إنأ نسألك علماً نافعاً ورزقاً حلالاً طيباً وعملا متقبلاً.

والله أسأل أن يوفقنا لما خير وأن يجنبنا ما هو شر وأن يهدينا سبيل الحق الرشاد إنه ولي ذلك والقادر عليه .

--أرحب بالعروض العربية---

أمجد محمد عز الدين عبد اللطيف

البريد الإلكتروني

amjedns2@gmail.com

م2009السودان -

السيرة الذاتية (CV)

http://amjedns2cv.blogspot.com

فهرس المحتويات

مقدمة

هذا الجزء يقوم بشرح المفاهيم الأساسية دون الخوض في التفاصيل المعقدة التي تصعب الفهم والتي تساعد على فهم المحاكي بصورة جيدة ، ويركز على العديد من الأمثلة التي تثبت الفهم وتقويه وتربط المواضيع بعضها ببعض .

في الجزء الأول يستعرض الكتاب أساسيات المحاكي والتي تقوم على TCL ويشرح أهم الأوامر فيها والمستخدمة في المحاكي.

ثم يسرد تسلسل كتابة البرنامج في ns2 وخصائصه وأنواع التنسيقات التي يمكن عملها وكذلك مجموعة الأدوات التي يمكن استخدامها مع المحاكي مثل nam والذي سيتم التطرق إليه لاحقاً بالتفصيل والذي يستخدم لعرض نتائج محاكاة الشبكة بشكل رسومي متحرك .

والآاااااااااااااااان!!!!!

هل أنت جاهز للخوض في عالم ns2

إذا على بركة الله وبسم الله نبدأ الرحلة ,,,,

محاكي الشبكات بين الماضي والحاضر

بدأ هذا المحاكي في عام 1989 بواسطة مشروع VINT والذي هو شكل من أشكال المحاكي Real Network Simulator في جامعة كاليفورنيا في معـــامل بركلـــي Berkeley Lab وهذا هو شعار الجامعة.

ويبلغ عدد المستخدمين لهذا المحاكي في عام 2006 كالتالي : أكثر من 1000 معهد في أكثر من خمسين دولة وأكثر من 10000 مستخدم حول العالم ويصل شهرياً حوالي 300 رسالة على الموقع الرئيسي للجامعة .

Ns home page

http://www.isi.edu/nsnam/ns

كما يوجد عدد من الإصدارات آخرها الإصدارة ns-allinone-2.33 وفي عام 2008 في شهر June تم تطوير ns2 إلى الإصدار الجديد والذي تمت تسميته ns3 والذي إلى الآن قيد التـطوير.

كما يتم عقد ورشة عمل كل سنتين والتي تعقد في اليونـان فـي مدينـة أثينـا ، والتـي بـدأت سنة 2006 وحققت نجاحاً كبيراً ثم عقدت الورشة الثانية في سنة 2008 ، وأطلق عليها WNS2(workshop ns2) ، وكان الهدف الرئيسي لهذه الورشة جلب الباحثين في مجال الشبكات من المجالين الصناعي والأكاديمي ليناقشوا التطور الحاصل ولكي يحددوا الاتجاه المستقبلي لمحاكاة الشبكات network simulation .

أساسيات محاكي الشبكات

- مكونات المحاكي:

يعتبر المحاكي غني جداً بالعديد من مكونات وبروتوكولات الشبكات والتي يتم التعبير عنها بشكل Object ، ويرتكز على عمله داخلياً على لغتين هما OTCL,C++ وكما تلاحظ هناك فرق شاسع بين اللغتين فلغة C++ هي لغة تستخدم Compiler لترجمة أوامرها بينما لغة OTcl فهي لغة تستخدم مفسر interpreter لتفسير أوامرها. فاستخدام لغة C++ لكفاءتها وسرعتها في التنفيذ وكتبت بها البروتوكولات والمكونات الثابتة والتي لا تتغير وتحتاج لسرعة في التنفيذ بينما لغة TCL مع كونها بطيئة في التنفيذ لكن سريعة في التعديل وهي شيء مهم بالنسبة للمستخدم فكتبت بها أوامر المستخدم مما جمع بين سرعة التنفيذ وسرعة التعديل من هاتين اللغتين .

- طريقة المحاكاة التي يتبعها المحاكي:

محاكي ns2 هو عبارة عن discrete event simulator يعني يستخدم مفهوم الأحداث المتقطعة والتي تعتبر واحدة من طرق المحاكاة ، حيث يتم جدولة الأحداث باستخدام نوع معين من المجدولات بأزمانها ويتم ترتيبها وحسب هذه الأزمان يحصل الحدث ويتم معالجته ويأخذ رقم متفرد ID .
فمثلاً في الزمن 0.2 يتم إرسال حزمة من نوع FTP فعندما تصل ساعة المحاكاة إلى هذا الزمن يتم ربط الإجراء المعالج برقم هذا الحدث ليقوم بمعالجته ، ويسمى الإجراء الذي يقوم بمعالجة الأحداث Handler ، ويحتوى المحاكي على مجموعة من الأصناف Classes التي ترتبط مع بعضها البعض وتساعد في عملية التخاطب ما بين اللغتين.
- كلاس Simulator:
يعتبر هذا الكلاس هو الكلاس الأساسي في المحاكي والذي عن طريقة تتم عملية المحاكاة ويحتوي على مجموعة من الواجهات interfaces التي تهدف إلى تهئية عملية المحاكاة واختيار النوع المناسب من أنواع المجدولات .
عملية المحاكاة عموماً تبدأ بعمل instance من هذا الكلاس يعني نسخة من هذا الكلاس وبعد ذلك نداء العديد من الدوال المندرجة تحته لإنشاء الشبكة وتهيئتها وتنسيقها .

أساسيات برمجة TCL

لغة TCL (Tool Command Language) وهي لغة Script language والتي تم تصنيفها very high level language مثلها مثل اللغات الأخرى التي تندرج تحت هذا التصنيف مثل PHP, JavaScript, Python, Perl وهي عبارة عن لغة تستخدم مفسر لتفسير أوامرها ولكي تدعم مفهوم Object Oriented لذلك تم توسيعها وامتدت إلى إصدارة أخرى وهي Otcl(Object extension of Tcl) وكتبت بها كما ذكر سابقاً أوامر المستخدم ، إذاً فالأوامر والدوال التي يتم نداؤها في المحاكي هي دوال مكتوبة بلغة Otcl وهي تقوم بدورها بالإتصال بلغة C++ ومن ثم تنفيذ أوامر المستخدم.

سوف نذكر فقط الأوامر والتعبيرات التي نحتاج إليها في هذا الكتاب فهي لغة برمجة مستقلة بذاتها وهي تشمل :

- المتغيرات
- التعبيرات الرياضية
- عبارات التحكم
- حلقة التكرار وأنواعها
- الإجراءات procedure.
- التعامل مع الملفات .

أولاً : المتغيرات :

طريقة تعريف المتغيرات في لغة TCL هي :

نضع عبارة set ثم اسم المتغير ثم بعد ذلك القيمة التي نريد وضعها في المتغير وهي إما تكون قيمة أو قيمة من متغير آخر أو قيمة من تعبير رياضي أو سلسلة نصية لأنه لا يوجد نوع بيانات للمتغير وهذه هي ميزة لغات scripting languages فهي لا تحدد نوع معين لنوع البيانات مثلاً integer, float , double, Boolean وغيره بل تقوم بأخذ نوع البيانات حسب القيمة المسندة للمتغير .

مثال :

set a 6	result: a=6
set b 4	result: b=4
set a $b	result: a=4
set a [expr 5+$b]	result: a=9
set a $b.3	result: a=4.3

نلاحظ هنا العبارتين الأولى والثانية تعريف متغيرين a,b وفي الثالثة قمنا بتغيير قيمة a إلى القيمة الموجودة في b وهي 4 وفي التعبير الذي يليه قمنا بوضع تعبير رياضي يضيف 5 إلى قيمة b ويضع الناتج في a وفي آخر تعبير قمنا بوضع قيمة b وألحقنا به كسر 0.3 لتصبح القيمة 4.3 .

ثانياً : التعبيرات الرياضية :

التعبيرات الرياضية للقيام بالعمليات الحسابية من جمع وطرح وضرب وقسمة وغيره من العمليات ويتم تعريف الكلمة المحجوزة بلغة TCL وهي expr وهي اختصار لعبارة expression وتوضع بين القوسين [expr expression] ثم بعدها التعبير الرياضي .

مثال :

```
set b 5
set a [expr 5+$b*2]
```

فهنا قمنا بتعريف متغير b به القيمة 5 ثم المتغير a وبه قيمة حسابية وهي 5 مجموع لها قيمة المتغير b وهي 5 ومضروبه في 2 وتصبح قيمة a تساوي 15.

ثالثاً : عبارات التحكم :

صيغة شرط if :

```
if expr script
```

مثال :

```
set  x 2
if $x>1 {…}
or if $x>1 {…} elseif $x<-3 {…} else {…}
```

رابعاً : حلقة التكرار وأنواعها :

حلقة for, while, foreach:

حلقة for loop :

صيغة الحلقة : for script expr script script

مثال :

```
for {set i 0} {$i<5} {incr i} {…}
```

حلقة while loop:

مثال:

```
while $x > 2 {…}
```

حلقة foreach loop:

وهي حلقة تعمل مع مصفوفة أو سلسلة من العناصر .

صيغة الحلقة: foreach listvar list body

مثال :

```
set total 0
foreach num {1 2 3 4 5} {
set total [expr $total+$num]
}
```

فناتج تنفيذ هذه الحلقة هو أن المتغير total سيأخذ القيمة 15 .

خامساً : الإجراءات Procedure :

تعتبر الإجراءات مثل الدوال في اللغات الأخرى مثل لغة java, c++ وغيرها .

وطريقة تعريف الإجراء هي :

proc name arguments {body}

مثال :

لتعريف إجراء يستقبل قيمة واحدة ويقوم بطرح واحد منها :

```
proc decr x {
expr $x-1
}
decr 3    ; => x=3
```

ولا يوجد عبارة return في الإجراء فهو يقوم بإرجاع القيمة ضمنياً في أخر عبارة

سادساً : التعامل مع الملفات :

طريقة فتح ملف والتعامل معه كالتالي :

```
outputfile [open "testfile" "w"]  set
puts $outputfile "this is line 1"
close $outputfile
```

في السطر الأول عرفنا متغير عادي اسمه outputfile ثم قمنا بفتح ملف اسمه testfile وهو الصلاحية التي نريدها هنا الكتابة w اختصار write .

طريقة كتابة برنامج في NS2:

في الفصل السابقة تحدثنا عن أساسيات لغة tcl والتي لا من معرفتها لكي نكتب برامجنا في NS2 وهنا سنكتب بلغة Otcl .

برنامج في NS2 يتكون من الخطوات التالية:

- Create the event scheduler (simulator)
- [Setup tracing]
- Create network topology
- [Setup routing]
- [Insert error modules/network dynamics]
- Create connection (transport)
- Create traffic (application)
- Schedule events
- Start the scheduler

تعتبر الخطوات التي بين الأقواس [] ليست أساسية في كتابة أي برنامج وهي عبارة عن مواضيع متقدمة ولكن ذكرتها للمعرفة فقط ولن تشرحها في هذه الإصدارة .

- Create the event scheduler (simulator)

وهي عبارة عن إنشاء object من كلاس Simulator والذي من خلاله سيتم إجراء عملية المحاكاة ، ويكون بالطريقة التالية :

```
set ns [new Simulator]
```

حيث ns هنا عبارة عن متغير ولا يشترط تسميته بهذا الاسم ولكن لسهولة التعامل معه ولأنه يأخذ اسم المحاكي وهو الشائع في الإستخدام بالنسبة للأمثلة في الكتب، فيمكنك تعريف أي اسمه ترغبه أو تفضله .

- Create network topology

في هذه الخطوة يتم تعريف شكل الشبكة topology وربطها مع بعضها البعض.

■ Create connection (transport)

في هذه الخطوة يتم إنشاء البروتوكول الذي سيتم استخدامه في الإرسال TCP,UDP وتحديد المصدر والهدف أو المصب لأي نوع منهما ويسمى Agent.

■ Create traffic (application)

يتم في هذه الخطوة يتم تعريف تطبيق يقوم بإرسال حزم بشكل معين مثل FTP,Telnet, CBR, VBR…etc وإسناده بالنسبة لمصدر agent.

■ Schedule events

ويتم هنا جدولة الأحداث يعني التطبيقات متى تبدأ ومتى تنتهي ومتى يتم نداء دالة إنهاء المحاكاة وتكون بالصيغة:

$ns at <time> <event>

any legitimate ns/tcl commands <event> :

يعني الحدث هنا ممكن يكون أمر عادي من أوامر ns/tcl ويمكن نداء دالة أو إجراء من خلاله وتكون بين علامتي التنصيص "" ، وإليك بعض الأمثلة :

$ns at 0.3 "$ftp start"

$ns at 1.2 "$ftp stop"

$ns at 1.3 "finish"

حيث دالة finish يتم كتابتها في البرنامج ومحتوياتها سيتم شرحها في المثال العملي بينما تطبيق ftp فيتم تحديد وقت البداية ووقت النهاية كما هو موضح.

■ Start the scheduler

بداية المحاكاة وذلك بالعبارة التالية

$ns run

ولا يوجد عبارة بعدها ومنها يتم بدأ عملية المحاكاة.

إنشاء شبكة وربطها :

الشبكة بطبيعتها تتكون من أجهزة وفي المحاكي يتم تعريف الأجهزة على شكل عقد node ، فعلى سبيل المثال إذا أردنا تعريف شبكة مكونة ثلاثة أجهزة وجهاز router نقوم بكتابتها كالتالي :

```
set pc1 [$ns node]
set pc2 [$ns node]
set pc3 [$ns node]
set router [$ns node]
```

حيث كلمة node هي عبارة عن كلمة محجوزة تخبر المفسر على أن هذا المتغير pc1 هو من نوع الصنف node ، أما عبارة $ns فنحن في الخطوة الأولى قمنا بتعريف متغير اسمه ns وهو عبارة عن object من الكلاس Simulator الذي يحتوى كل التفاصيل المتعلقة بعملية المحاكاة فنقوم باختيار الصنف أو الدالة المناسبة منه وذلك عن طريق $ns class/method فإما كلاس أو دالة .

قمنا حتى الآن فقط بتعريف الأجهزة لكن لم نقم بربطها ، فللقيام بربطها نستخدم الدالة duplex-link والتي تعني وجود رابط من الجهتين رابط من الجهاز الأول إلى الجهاز الثاني ومن الجهاز الثاني إلى الجهاز الأول ، إذا كنا نريد رابط واحد فقط نستخدم الدالة simplex-link وهي تعني رابط واحد فقط ، أما كيفية الكتابة فصيغته على النحو التالي :

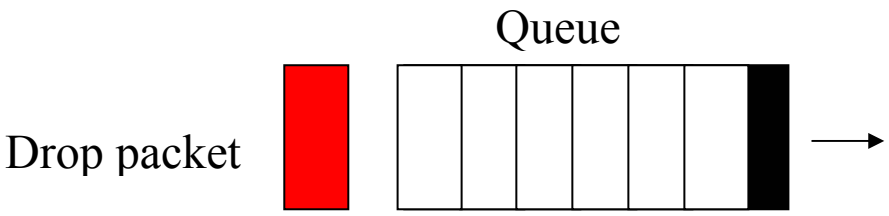
```
$ns duplex-link $n0 $n1 <bandwidth> <delay> <queue>
```

وفي مثالنا هذا

```
$ns duplex-link $pc1 $router 5mb 2ms DropTail
$ns duplex-link $pc2 $router 5mb 2ms DropTail
$ns duplex-link $pc3 $router 5mb 2ms DropTail
```

دالة duplex-link تستقبل مجموعة من parameter وهي سعة الناقل ومقدار التأخير في هذا الناقل ونوع Queue ، ما هو هذا Queue ؟

Queue هو عبارة عن مجموعة من الخوارزميات مثل DropTail, RED, CBQ يمكن الرجوع إلى الإنترنت لمعرفة كيف تعمل هذه الخوارزميات وليس المجال مجال تفصيلها لكن نذكر واحدة فقط وهي DropTail والتي في عملها تعني امتلاء Queue فتقوم هذه الخوارزمية بحذف الحزم من الذيل أو من الآخر ، حسناً نوضحها بشكل آخر:



المربع هو عبارة عن حزمة فهنا queue إذا امتلأ يتم الحذف من الخلف أو من الذيل لذلك سميت هذه الخوارزمية بهذا الاسم، فأي عقدة يكون بها queue فإذا كانت هناك أكثر من حزمة وصلت يتم وضعها قليلاً في الصف queue لكي يتم خدمة الحزمة الحالية ثم بعد ذلك يتم أخذ الحزم من الأمام والتي هي باللون الأسود وإرسالها.
ويمكن ربطها بطريقة أخرى باستخدام دالة simplex-link والتي ستكون 6 عبارات بدل 3 لأنه رابط في الإتجاهين.

ملاحظات هامة:

- لغة tcl حساسة جداً بالنسبة للفراغات يعني أنه لازم يكون ما بين كل parameter والآخر مسافة فراغ واحدة فقط وإلا فيسعطي خطأ عند تنفيذ البرنامج.
- بالنسبة لأسماء الخوارزميات يجب أن تكتب بصورة صحيحة فهنا نوع DropTail يجب أن يكتب بهذه الطريقة وإلا أيضاً سيعطي خطأ في التنفيذ فالحروف الكبيرة ليست كالصغيرة يعنى case sensitive.
- أيضاً يجب كتابة الرموز ms, mb مع القيمة .

تعريف التطبيقات والمضيفات agent and application:

بعد تكوين الشبكة وربطها أصبح من الممكن الآن تشغيل بعض التطبيقات لكي تقوم بإرسال الحزم وللقيام بذلك يتم على خطوتين أولاً : تعريف المضيف الذي سيستضيف التطبيق بالإضافة إلى أنه يمثل طريقة الإرسال إما TCP,UDP فنبدأ بتعريف المضيف agent والذي يحتوي على مصدر ومصب.

لتعريف TCP agent نقوم بالآتي :

set tcp [new Agent/TCP]

في هذه الخطوة عرفنا متغير tcp من النوع Agent/TCP والتي كما ذكرنا سابقاً طريقة الإرسال أو البروتوكول الذي يتم عن طريقه الإرسال ونسميه بالمصدر .

set tcpsink [new Agent/TCPSink]

نقوم بتعريف المصب بالنسبة للمصدر tcp وهو tcpsink فالمصب TCPSink هو خاص فقط بـ TCP .

$ns attach-agent $n0 $tcp

إلى هنا قمنا بتعريف المصدر والمصب ولم نقم بإسنادها لعقدة معينة ، فيتم إسنادها عن طريق الدالة attach-agent وتستقبل العقدة والمصدر، وهنا قمنا باسناد المصدر إلى العقدة n0 .

$ns attach-agent $n1 $tcpsink

قمنا باسناد المصب للعقدة n1.

$ns connect $tcp $tcpsink

يجب إجراء عملية اتصال بين كل مصدر ومصب لكي يتم التفريق بين المصادر الموجودة في الشبكة ولتمييزها عن بعضها البعض وذلك عن طريق دالة connect لتربط المصدر المصب .

set ftp [new Application/FTP]

$ftp attach-agent $tcp

في هذه الخطوة يتم اسناد تطبيق معين إلى TCP ليقوم بإرسال الحزم ، وفي مثالنا عرفنا تطبيق FTP وقمنا بإسناده إلى المصدر tcp عن طريق دالة attach-agent.

ولتعريف UDP agent نقوم بنفس الخطوات كالأتي :

```
set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n0 $udp
$ns attach-agent $n1 $null
$ns connect $udp $null
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
```

نلاحظ المصب لكل من TCP,UDP ففي TCP استخدم المصب TCPSink بينما مع UDP فاستخدم المصب Null لماذا؟

سؤال ؟ ألم تعرف الإجابة عنه بعد ...

هذا هو مفهوم Connection oriented, Connectionless في الشبكات. ماذا!! لا تعرفه لماذا! حسناً سأشرح لك هذا المفهوم ببساطة مفهوم Connection oriented يعني أنه إذا تم إرسال حزمة من طرف إلى طرف آخر فيجب على الطرف الآخر الرد وإرسال حزمة تؤكد من أن الحزمة التي أرسلت قد وصلت ثم يقوم بإرسال الحزمة الثانية وهكذا..., وهذا ما يعمل به TCP أبسط مثال لهذا المفهوم التليفون العادي أو الموبايل .
أما مفهوم Connectionless فهو يستخدمه UDP ويعني أن الطرف الأول يقوم بإرسال حزمة وليس على الطرف الآخر الرد وأبسط مثال له البريد الإلكتروني .
التطبيقات الموجودة في المحاكي تنقسم إلى قسمين :

- Simulation application
مثل FTP, Telnet, VoiceOverIP, Audio,…etc وهي التطبيقات التي يتم استخدامها في الشبكة الحقيقية .

- Traffic generator
مثل CBR, VBR, Exponential, Pareto,…etc وهي التطبيقات التي لا توجد في الحقيقة ولكن هي عبارة حركة traffic يتم إنشاؤه بطريقة معينة يتم تحديدها ويوجد بها عدد من التوزيعات التي يتم استخدامها في مجال المحاكاة مثل Exponential وغيرها .

مثال عملي :

مثالنا يتكلم عن شبكة مكونة من جهازين يقوم الجهاز الأول بإرسال حزم FTP للجهاز الآخر ويقوم الجهاز الآخر بإرسال حزم CBR للجهاز الأول .

البرنامج كاملاً :

```
set ns [new Simulator]
set f [open out.nam w]
$ns namtrace-all $f
proc finish {} {
    global ns f
    $ns flush-trace
    close $f
    # run animation
    puts "running nam..."
    exec nam out.nam &
    exit 0
}
set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0 $n1 5mb 2ms DropTail
set tcp [new Agent/TCP]
set tcpsink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n1 $tcpsink
$ns connect $tcp $tcpsink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0.1 "$ftp start"
$ns at 1.5 "$ftp stop"
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
```

```
set null [new Agent/Null]
$ns attach-agent $n0 $null
$ns connect $udp $null
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$ns at 0.0 “$cbr start”
$ns at 1.5 “$cbr stop”
$ns at 1.5 “finish”
$ns run
```

```
set ns [new Simulator]
set f [open out.nam w]
$ns namtrace-all $f
```

السطر الأول قمنا بتعريف object من كلاس Simulator.
السطر الثاني قمنا بفتح ملف اسمه out.nam وهو امتداد خاص ببرنامج Nam الذي سيعرض الشبكة في شكل رسومي سنتطرح إليه لاحقاً .
السطر الثالث استخدمنا دالة namtrace-all والتي تقوم بوضع جميع أحداث المحاكاة التي تمت محاكاتها في صيغة يتعامل معها برنامج Nam ليقوم بعرضها.

```
proc finish {} {
    global ns f
    $ns flush-trace
    close $f
    # run animation
    puts "running nam..."
    exec nam out.nam &
    exit 0
}
```

هذا هو الإجراء الذي يتم نداؤه في نهاية عملية المحاكاة والذي يقوم بإغلاق الملفات التي تم فتحها وتنظيف الذاكرة والخروج .

تعريف الإجراء يكون بعبارة proc ثم اسم الإجراء وبعده المتغيرات التي يستقبلها وهنا لا يوجد .
السطر الأول من الإجراء يوجد عبارة global ، هنا يتم استخدام هذه العبارة لأنه في لغة tcl المتغيرات الموجودة خارج الإجراء لا يمكن رؤيتها داخل الإجراء إلا بإخبار الإجراء بأن هذا المتغير هو متغير معرف خارجه ويتم ذلك عن طريق عبارة global.
السطر الثاني : عبارة $ns flush-trace وهي تقوم بتفريغ الذاكرة من المتغيرات التي تم استخدامها في المحاكاة لتحرير المساحة المستخدمة.
السطر الرابع عبارة عن دالة إغلاق الملف الذي تم فتحه في بداية البرنامج.
السطر الخامس عبارة طباعة عبارة على الشاشة باستخدام دالة puts .
السطر السادس عبارة عن دالة تقوم بتنفيذ الأمر الذي بعدها في shell أو terminal أو Dos، وهنا قمنا بتنفيذ برنامج nam ومرر الملف الذي تمت الكتابة فيه وهو out.nam.
ما تبقى من البرنامج تم شرحه بالتفصيل سابقاً .

ملاحظات هامة:

- الحروف الكبيرة ليست كالحروف الصغيرة case sensitive .
- مكان الإجراء finish لا يشترط في بداية البرنامج ويمكن كتابه في أي مكان لأن المجدول يقوم بترتيب الأحداث زمنياً ثم تنفيذها.

NS2 Formatting التنسيق:

سنتحدث عن التنسيق من ناحية :

- Color
- Node manipulation
- Topology layout
- Misc

أولاً : الألوان Color:

إسناد اللون فهو إما أن يكون لون عقدة node أو لون رابط link أو لون الحزم packets :

إسناد اللون لعقدة node :

```
$node color red
```

إسناد اللون لرابط link:

```
$ns duplex-link-op $n0 $n1 color green
```

عبارة duplex-link تستخدم لعملية ربط العقد كما أسلفنا سابقاً ، بإضافة عبارة op والتي تعني خيارات options مثل اللون والعنوان label.

إسناد اللون للحزم packets:

يتم أولاً تعريف الألوان التي ستستخدم في بداية البرنامج كالآتي:

```
$ns color 1 red
$ns color 2 green
$ns color 3 yellow
```

نقوم باعطاء الألوان أرقاماً مثل 1،2،3 وهي غير ثابتة فيمكن وضع أي رقم ، أما عن كيفية الاسناد فيتم اسناده للمضيف agent وذلك عن طريق المتغير fid_ وهو اختصار flow id وذلك عن طريق الأمر التالي :

```
$tcp set fid_ 2
$udp set fid_ 3
```

هنا يتم اسناد لون الحزم المتعلقة بالمضيف tcp اللون رقم 2 الذي تم تعريفه أول البرنامج وفي مثالنا رقم 2 هو اللون الأخضر green.

ثانياً : Node Manipulation:

معالجة العقدة تكون بعمليتين الأولى تغيير الاسم (label) والثانية تغيير الشكل (shape) ، تغيير الاسم هو لتوضيح معنى العقدة في برنامج nam مثلاً pc, router وكذلك الأمر بالنسبة للشكل لتمييز العقد بأشكالها فمثلاً شكل router يختلف عن شكل host (pc) وهكذا وطريقة كتابتها كالأتي :

```
$node label “server”
```

$node2 label “host1”
أيضاً يمكن وضع عبارة أو توضيح label فوق الرابط لتوضيح مثلاً سعة الرابط وسرعته أو نوعه مثلاً ويكون ذلك بالأمر التالي :
$ns duplex-link-op $node $nod2 label “Ethernet link”
أما الشكل shape فهو إما أن يكون circle, box, hexagon :
$node shape box
$node2 shape circle

ثالثاً : Topology layout:
المقصود بمصطلح layout يعني تنظيم شكل الشبكة في برنامج nam ويكون ذلك كالآتي :

$ns duplex-link-op $n0 $n1 orient left
$ns duplex-link-op $n0 $n1 orient right-up
$ns duplex-link-op $n0 $n1 orient down
$ns duplex-link-op $n0 $n1 orient 60deg

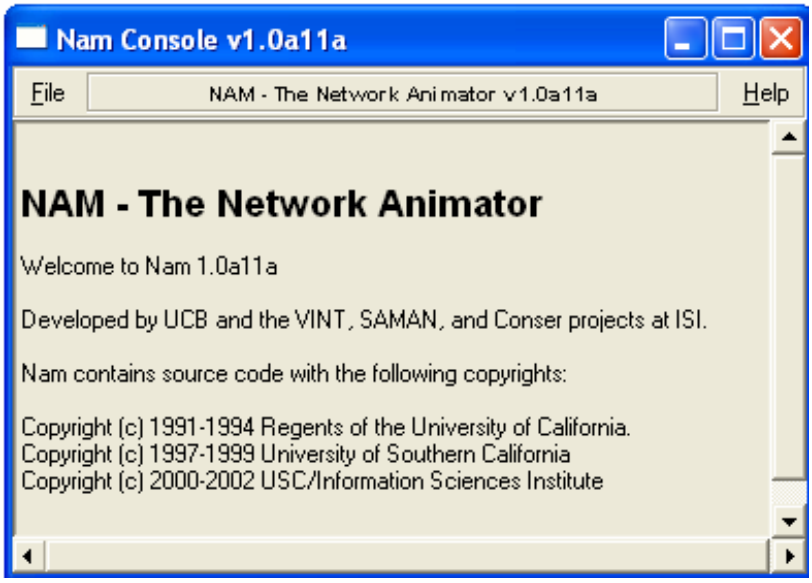
يتم تحديد التوضع أو المكان الذي ستوضع به العقدة n1 بالنسبة للعقدة n0 باستخدام الدالة orient ويمكن وضع الاتجاهات (up, down, right, left, right-up, right-down, left-up, left-down, 60deg) وإذا لم تحديد أي layout فيستم تحديد الوضع الإفتراضي .

رابعاً : Misc:
وهي عبارة بعض الدوال تستخدم في التنسيق سنشرح إثنين منها وهما:
trace-annotate , set-animation-rate
الدالة الأولى trace-annotate وهي دالة تستخدم لتوضيح وإضافة بعض التعليقات في برنامج nam أثناء التنفيذ وصيغتها كالآتي :
$ns at 3.5 “$ns trace-annotate \” tcp start send packet\””
ويعني ذلك في الزمن 3.5 ستقوم هذه الدالة بطباعة عبارة tcp start send packet في الجزء السفلي من برنامج nam.
الدالة الثانية وهي set-animation-rate وهي تستخدم لزيادة أو نقصان سرعة العرض في برنامج nam وصيغتها كالتالي :
$ns at 1.0 “$ns set-animation-rate 0.1ms”

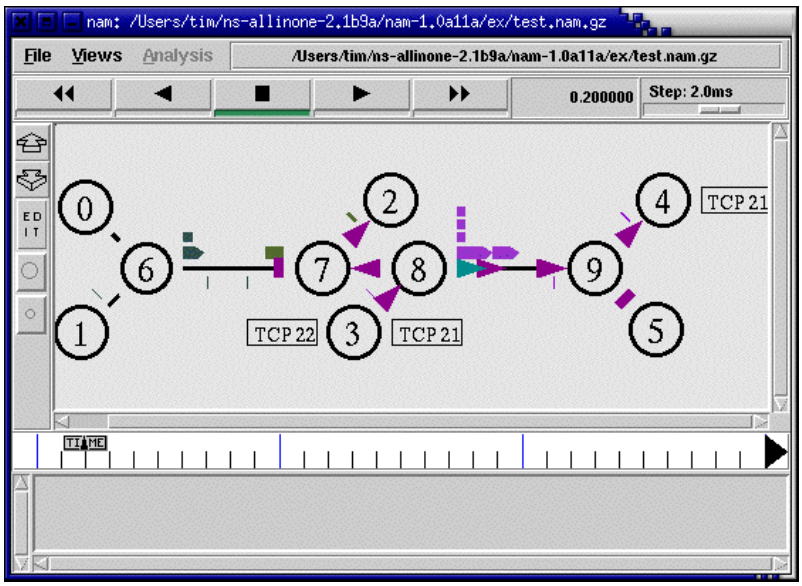
استخدام برنامج Network Animator(NAM):

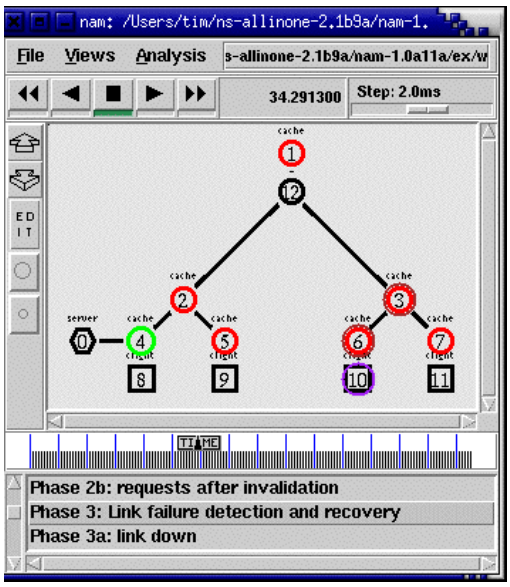
يتم استخدام هذا البرنامج لعرض ملف تم كتابته بصيغة معينة خاصة به ليقوم لعرض نتائج المحاكاة بصورة حركية بحيث يتم رؤية النتائج بصورة متحركة وهذا يساعد على الفهم .

يمكن أيضاً عن طريق هذا البرنامج إنشاء شبكة بدلاً من تعلم لغة TCL لكتابة البرنامج في NS2 وشكل البرنامج يظهر في الشكل التالي :

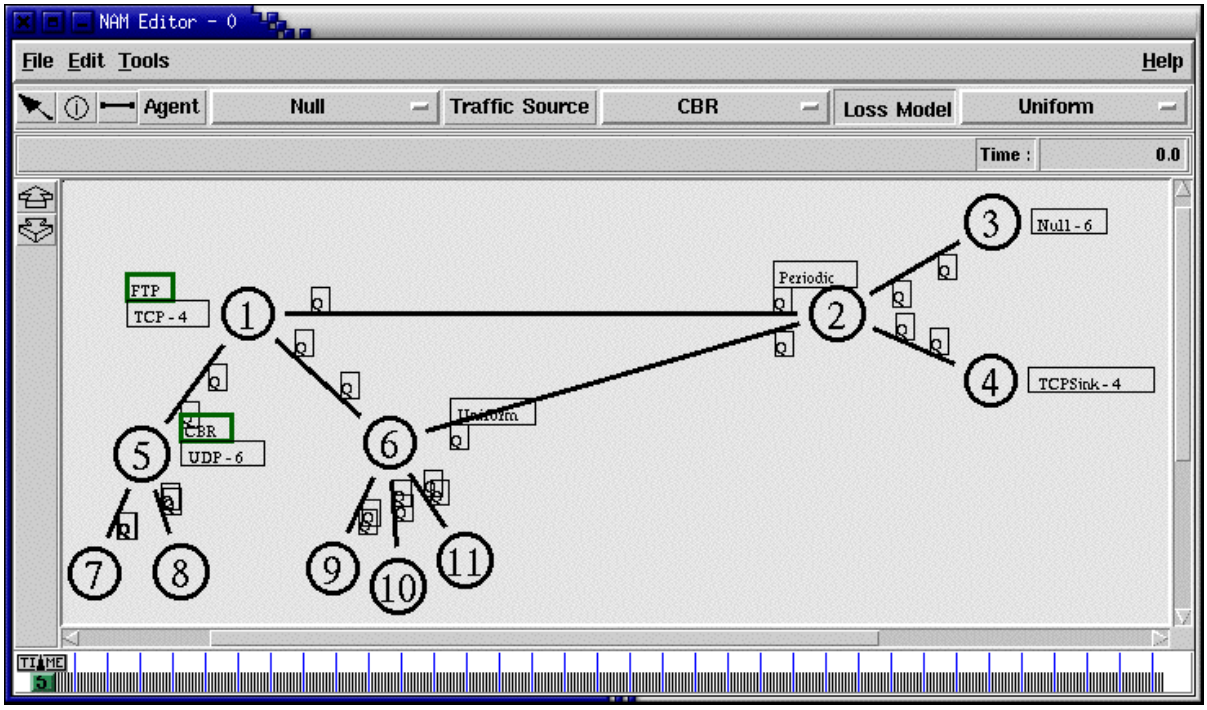


وهذه صورة أخرى لعرض نتائج محاكاة بشكل رسومي متحرك :

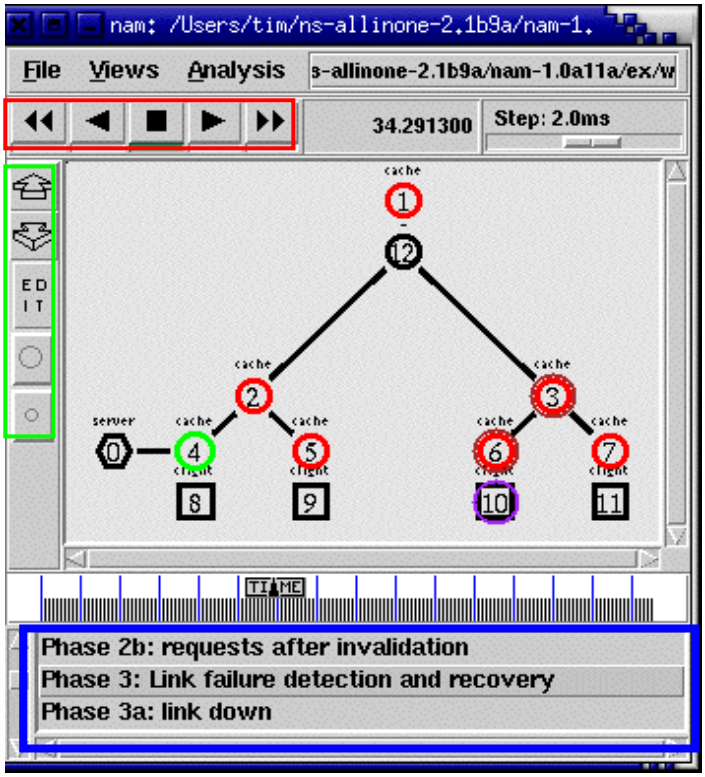




وهذه صورة أخرى للبرنامج والتي توضح كيفية إنشاء شبكة عن طريقه بدون تعلم لغة TCL:



حسناً الآن سنتطرق إلى تعريف الأزرار الموجودة في البرنامج في كلا الواجتين سنبدأ بالواجهة الأولى :



اللون الأحمر:

الجزء المحدد بإطار أحمر يحتوى على أزرار ليست غريبة عنا فنحن نراها في أي مسجل أو مشغل أصوات وهنا المقصود بها تشغيل المحاكاة فكما قلنا أنه يقوم بعرض الشبكة بشكل رسومي ويمكن أن نقول فيلم فبزر التشغيل يتم تشغيل المحاكاة وبزر الإيقاف يتم إيقاف المحاكاة وكذلك الأمر بالنسبة لبقية الأزرار والتي تعني تقدم إلى البداية أو النهاية ويوجد زر تشغيل معاكس للاتجاه حيث يقوم بعرض الفيلم أقصد فلم المحاكاة من نهايته إلى بدايته.

اللون الأخضر:

الجزء المحدد باللون الأخضر يحتوى على خمسة أزرار من فوق إلى تحت :

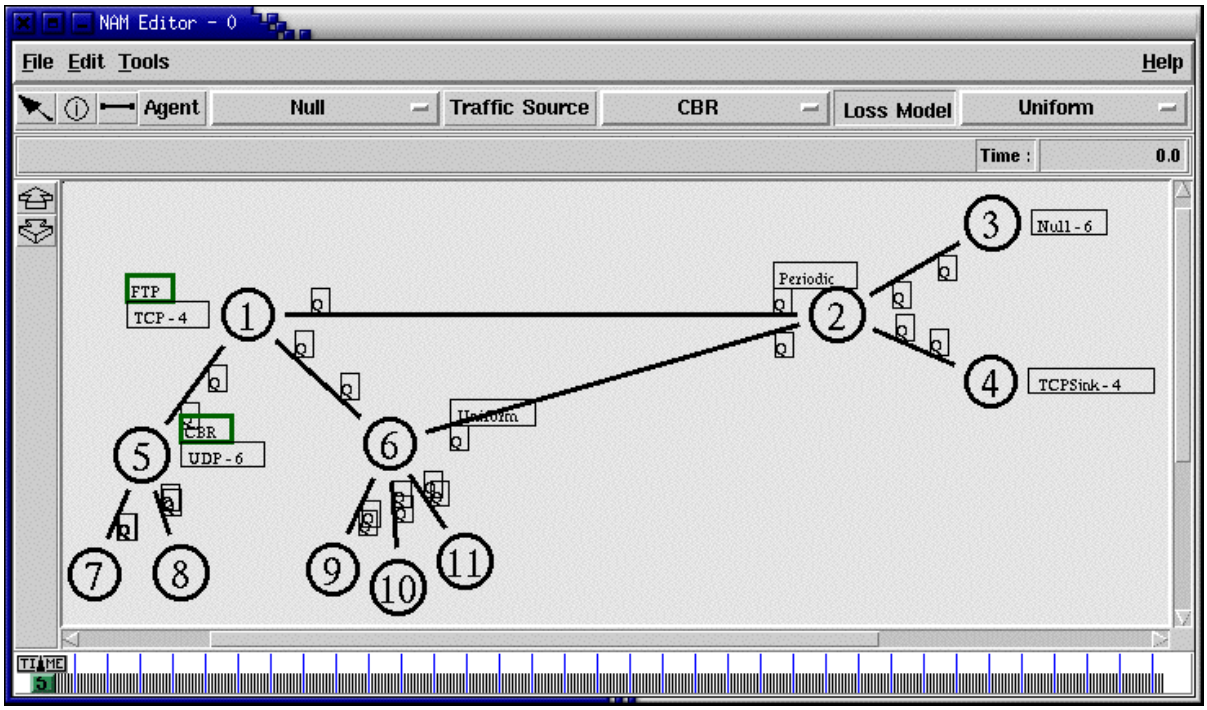
أول زرين وشكلهما شكل سهم لأعلى وأسفل ويقومان بتكبير وتصغير حجم نافذة العرض التي بها الشبكة.

أما الزر الثالث والذي مكتوب عليه كلمة Edit ويتم استعماله في حالة تعديل شكل العرض يعنى إعادة ترتيب العقد بتغيير أماكنها وذلك بالضغط عليها والتحريك إلى أي مكان.

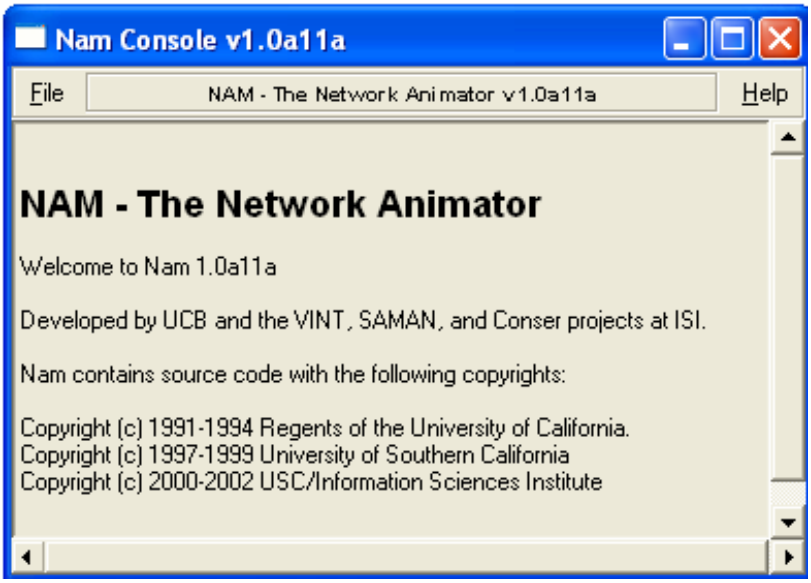
أخر زرين يقومان بتكبير أو تصغير حجم العقد.

اللون الأزرق:

هذا الجزء يستخدم لإظهار التعليقات على شكل المخرجات في زمن معين ويتم ذلك باستخدام دالة trace-annotate التي سبق شرحها ، فالتعليق الذي يكتب في هذه الدالة يظهر في هذا الجزء عن الزمن الذي تحديده.



وهذه الواجهة الثانية التي نستخدمها في بناء شبكة بدون استخدام لغة TCL ويتم فتح هذه النافذة عن طريق فتح برنامج Nam وذلك عن طريق فتح cmd في ويندوز أو فتح terminal في لينيكس وكتابة nam.exe(windows) or nam(linux) فتظهر الواجهة التالية :



هذه الواجهة في ويندوز ثم من قائمة File نختار new.

بعد ذلك يتم الضغط على زر العقدة ثم الضغط في مساحة الرسم ثم زر الخط أو الرابط والضغط على العقدة الأولى والسحب حتى العقدة الثانية حتى يتم رسم شكل الشبكة الذي نريده.

بقية تفاصيل هذا البرنامج سأقوم بشرحها بالتفصيل إن شاء الله في إصدارة أخرى لأنها تتطلب ذكر بعض المواضيع المتقدمة قليلاً.

نهاية الجزء الأول

وقريباً إنشاء الله الجزء الثاني.....

مراجع هذه السلسلة

-The ns Manual,
http://www.isi.edu/nsnam/ns/ns-documentation.html
- Ns distribution download
- http://www.isi.edu/nsnam/ns/ns-build.html
- Installation problems and bug-fix
- http://www.isi.edu/nsnam/ns/ns-problems.html
- Ns-users mailing list
- Ns-users@isi.edu
- See http://www.isi.edu/nsnam/ns/ns-lists.html
- Archives from above URL
- Lots of slides displayed later