سلسلة تعلم بسهولة

محاكي الشبكات

Network Simulator (NS2)

تأليف

أمجد محمد عز الدين عبد اللطيف

سلسلة

تعلم بسهولة

محاكي الشبكات NS2

الجزء الثاني

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 ، 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 إنا 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 .

أمجد محمد عز الدين عبد اللطيف
البريد الإلكتروني
amjedns2@gmail.com
السودان – 2009م

,,اللهم لا سهل إلا ما جعلته سهلا وأنت تجعل الحزن إذا شئت سهلاً,,

فهرس المحتويات

مقدمة:

يرتكز هذا الجزء على شرح موضوعين ،يتحدث الموضوع الأول من هذا الجزء عن المهارات الأساسية التي يجب توفرها لكتابة البرامج الكبيرة وتعتبر كمساعد للمبرمج في كتابة برنامجه بأكثر من طريقة .

في الموضوع الثاني من الكتاب يتحدث عن التحليل وطرق جمع المعلومات من الشبكة ويتطرق إلى أهم كلاسين موجودين في المحاكي والذي يتم استخدامهما بكثرة للحصول على معلومات تساعد في حساب متغيرات الأداء Performance Metrics مثل الانتاجية ومقدار التأخير ومقدار استهلاك الشبكة للمصادر المتاحة Utilization وغيرها.

بالنسبة للتحليل فيجب أولاً التعرف على صيغة ملف التحليل Trace file والذي عن طريقه يمكن أيضاً الحصول على نتائج باستخدام طرق الفلترة للملف أو باستخدام برامج جاهزة تقوم بتحليل الملف واستخراج النتائج مباشرة ثم بعد ذلك يأتي دور الرسم البياني الذي يوضح النتائج التي تم الوصول إليها في شكل رسم بياني.

في هذا الكتاب جميع الامثلة المذكورة يتم تطبيقها على نظام لينيكس فيكمن تجربتها في نظام لينيكس أو باستخدام برنامج Vmware الذي يسمح لك بتنصيب لينيكس على ويندوز لمن أراد العمل على بيئة الويندوز كما يوجد برنامج آخر من انتاج شركة Sun Microsystem اسمه VirtualBox أيضاً يمكن التعامل به في ويندوز كذلك يوجد طرق أخرى يمكن العمل بها سواءً في نظام لينيكس أو ويندوز سيتم التطرق لها لاحقاً في هذا الجزء.

الباب الأول

مهارات أساسية في كتابة البرامج

مقدمة

لكي تكون محترفاً في استخدام المحاكي يجب عليك أولاً أن تقوم بتعلم طريقة كتابة برامجه ثم عليك بعد ذلك تعلم كيفية تصميم شبكة ثم تحليلها وبالاضافة لذلك فإن هناك بعض المهارات البرمجية التي إذا قمت بتعلمها فتساعدك على كتابة برامج قوية وسهلة ومفهومة وكذلك تطلعك على طرق مختلفة لكتابة البرامج التي يمكن أن تختصر كثيراً من الأسطر البرمجية هذه المهارات تبدأ بتعلم استخدام كلمة global في دالة finish وكيف يمكن عمل نسخة من الكلاس Simulator ، ثم بعد ذلك يأتي استخدام مفهوم الثوابت والمصفوفات في كتابة البرنامج وكيف تساعد على اختصار كتابة البرنامج.

لتنظيم البرنامج إذا كان حجمه كبيراً فيمكن توزيع البرنامج إلى دوال صغيرة أو إلى ملفات وهي من المهارات التي أدخلها مفهوم Object-oriented (البرمجة الشيئية).

آخر مهارة وهي معروفة في جميع لغات البرمجة وتعتبر من أهم المهارات التي يجب على المبرمج تعلمها واستخدامها وهي التوثيق ، نعم توثيق البرنامج من الأشياء المهمة جداً فهي تساعد المبرمج على تذكر ما قام به فيما بعد وتوضح تسلسل البرنامج بطريقة منظمة ومفهومة وهو ما نريد أن نطبقه في البرامج التي نكتبها باستخدام محاكي الشبكات NS2 فهي من البرامج التي بدون التوثيق تصبح برامج صعبة الفهم بالنسبة للقارئ.

مهارات اساسية في كتابة البرنامج

يوجد العديد من المهارات التي سوف نتطرق لها من خلال المواضيع التي سيتم شرحها والمقصود بالمهارات أي مهارات كتابة البرنامج وسوف نتطرق لأكثر من طريقة لكتابة الكود وذلك لتسهيل عملية الربط ما بين الاجراءات والملفات.

المهارة الأولى استخدام عبارة global:

في دالة finish لاحظ يوجد عبارة global لتوضيح أن المتغيرات هذه هي متغيرات معرفة خارج الاجراء وإذا لم نقم بهذه العملية سوف يعطي البرنامج رسالة خطأ أثناء التنفيذ (لأن الكود هو لغة OTCL وهي لغة Scripting لذلك التنفيذ يكون سطراً سطراً فمثلاً إذا كان الخطأ في السطر السابع سيتم تنفيذ الأسطر الستة الأولى وعند السابع ستظهر رسالة الخطأ) والتي سيقول ان المتغير الذي تم استخدامه داخل هذا الاجراء غير معروف مثال عملي على ذلك :

```
set ns [new Simulator]
set message  "finish program"
proc finish {} {
global ns  message
$ns flush-trace
puts $message
}
```

في هذا المثال لو تم مسح عبارة global ns message سيعطي البرنامج خطأ بأن المتغير ns غير معرف داخل الاجراء باعتباره الذي سينفذ أولاً.

عبارة global هنا تشمل ns object والمتغيرات الاخرى ، ونحن نعرف ان كل الدوال تندرج تحت ns object والذي تم تسميته هنا في المثال ns.

يوجد دالة في المحاكي تسمى instance يتم استعمالها مع classes ويمكن استخدامها مع الكلاس Simulator لعمل نسخة من الكلاس Simulator الموجود في البرنامج الحالي بدلاً من استخدام عبارة global وذلك بالطريقة التالية:

```
set ns [new Simulator]
proc finish {} {
set ns_ [Simulator instance]
$ns_ flush-trace
}
```

في المثال تم عمل نسخة من الكلاس Simulator وهو نفس Object الخارجي ns ، ففي المثال تغير اسم المتغير من ns إلى ns_ فالمقصود أنه يمكن اخبار الاجراء بأن متغير الكلاس Simulator معرف داخل الاجراء بطريقتين الأولى باستخدام عبارة global والثانية بعمل نسخة من كلاس Simulator الخارجي.

المهارة الثانية استخدام الثوابت والمصفوفات في كتابة البرنامج:

فهي تساعد على فهم البرنامج بطريقة أفضل وتزيد من سرعة التعديل بالنسبة للثوابت فهي لا تختلف عن المتغيرات العادية في لغة TCL مثال:

```
set numbers  10
set num 2
for {set i 0} { $i < $numbers} {incr i} {
puts "the multiplication table of $num *  $i = [expr $i * $num]"
}
```

لاحظ في هذا المثال عملية التحكم بالبرنامج سهلة جدا فإذا أردنا أن نعدل البرنامج لحساب مثلاً جدول الضرب بالنسبة للعدد 3 فما علينا سوى تغيير قيمة المتغير num.

بالنسبة للمصفوفات فيوجد نوع جميل من المصفوفات ومستخدم بكثرة ويسمى associative array والتي تقوم بتعريف عناصرها بواسطة فهرس مثال على ذلك:

```
set num("first")  1
set num("second")  2
set sum [expr $num("first")+ $num("second")]
puts "the sum = $sum"
```

لنقم بتجربة ذلك في مثال عملي :

```
set ns [new Simulator]
#open nam file
set nf [open out.nam w]
$ns namtrace-all $nf
#set variables of topology
set lanNodes  5
set link("bandwidth")   5mb
set link("delay")   2ms
set link("queue")  DropTail
```

```
#define two routers
set router0  [$ns node]
set router1  [$ns node]
#link two routers
$ns duplex-link $router0 $router1 2mb 2ms DropTail
#create and connect nodes with routers
for {set i 0}{$i < $lanNodes}{incr i} {
    set n($i) [$ns node]
    set n([expr $i+5])  [$ns node]
    $ns duplex-link $n($i) $router0 $link("bandwidth") $link("delay") $link("queue")
$ns duplex-link $n([expr $i+5]) $router1 $link("bandwidth") $link("delay") $link("queue")
}
```

داخل الحلقة قمنا بتعريف العقد من 0-4 لتمثيل أجهزة الشبكة الأولى وربطها مع router0 والعقد من 5-9 لتمثيل أجهزة الشبكة الثانية باستخدام صيغة expr والتي تستخدم للقيام بعملية رياضية وهنا قمنا بزيادة رقم العقدة 5.

هذا فقط مثال بسيط يوضح كيفية استخدام هذا النوع من المصفوفات في كود خاص بالمحاكي لكن أغلب استخدامه يكون غالباً في أكواد الشبكات اللاسلكية wireless فلو قمت بفتح أي كود خاص بالشبكات اللاسلكية ستجد فيه استخدام هذا النوع من المصفوفات ولا يوجد كود كهذا حقيقة لكنه اسهل في الفهم وصعب في الكتابة لانه كثير.

```
proc  finish { } {
        global ns nf
        close $nf
        $ns flush-trace
        exec nam out.nam
        exit
}
$ns at  5 "finish"
$ns run
```

باستخدام المصفوفات والمتغيرات يجعل البرنامج أكثر مرونة فمثلاً إذا أردنا زيادة عدد الأجهزة في الشبكتين إلى 10 ما علينا فقط إلا تغيير قيمة المتغير lanNodes ، أيضاً يمكن قراءة البرنامج بصورة أوضح في ربط الشبكة بالـ router0 يمكن قراءة السطر ومعرفة bandwidth , delay and queue من شكل السطر.

المهارة الثالثة توزيع البرنامج في أكثر من دالة أو ملف :

وذلك بأن تكتب الكود المختص بعملية انشاء شكل الشبكة في إجراء (دالة) لكي تسهل من فهم البرنامج والتي تدعم مفهوم من مفاهيم لغات البرمجة يعرف Readability فالبرنامج إذا لم يكن أي شخص غير كاتبه يستطيع فهمه أو متابعة تنفيذه يعتبر برنامج ينقصه هذا المفهوم وهذا هو الفارق الذي تنافست فيه لغات البرمجة.

إذاً يفضل في كتابة برامج محاكي الشبكات Ns2 أن تراعى هذه النقطة ويتم ذلك بكتابة الكود المختص مثلاً بإنشاء شكل الشبكة في اجراء ثم عملية انشاء traffic معين واسناده إلى عقدة في اجراء آخر وغيره.

الصيغة العامة لتعريف الاجراء:

```
proc proc_name {parameters } {
#code of procedure
}
```

يمكن أيضاً توزيع البرنامج على أكثر من ملف مثلاً يمكن وضع أكثر الدوال استخداماً في ملف وندائه في البرامج بدلاً من كتابته من جديد وهو مفهوم أيضاً معروف في لغات البرمجة يسمى reusability يعنى اعادة الاستخدام ، ويتم عمل ذلك باستخدام الكلمة المحجوزة source (التي تماثل في لغة الجافا import وفي لغة C/C++ include) ثم اسم الملف ثم بعدها مباشرة يمكن نداء اي متغير أو اجراء موجودة في هذا الملف.

الصيغة العامة :

```
source procs.tcl
set avg  [getAverage 10  22 32 54]
```

حيث الاجراء getAverage موجود في الملف procs.tcl.

يمكن نداء اي دالة موجودة في هذا الملف بعد عبارة source وهذا بالطبع يفيد كثيرا في جعل الدوال أو الاجراءات المستخدمة بكثرة توضع في ملف يتم استخدامه في الأكواد بدون الحوجة لكتابتها من جديد.

المهارة الرابعة توثيق البرنامج :

وهي من المهارات المهمة جداً بالنسبة لكاتب البرنامج لكي يتذكر البرنامج بسرعة ولقارئ الكود لكي يفهم الكود بطريقة أفضل ، ففي اللغات الأخرى تركز على هذه المهارة وتعطي أكثر من طريقة لكتابة التوثيق حيث يمكن كتابة سطر واحد فقط أو كتابة أكثر من سطر فمثلاً في لغة جافا يمكن التوثيق بطريقتين :

يمكن استخدام // لكتابة سطر واحد فقط أو /* */ لكتابة أكثر من سطر.

يتم التوثيق في لغة OTCL بطريقة واحدة وهي عن طريق الرمز # مثلاً:

```
#define nodes
set n0 [$ns node]
set n1 [$ns node]
#connect two nodes
$ns duplex-link $n0 $n1 2mb 2ms DropTail
```

الباب الثاني

الحصول على معلومات من الشبكة

مقدمة

الشبكة بطبيعتها تختلف من منطقة لأخرى وكل شبكة تأخذ متغيرات أو تهئية بطريقة معينة تناسب وضعها ، فيوفر محاكي الشبكات NS2 أكثر من طريقة لتهئية مكونات الشبكة والتي بقيمها تؤثر على أداء الشبكة.

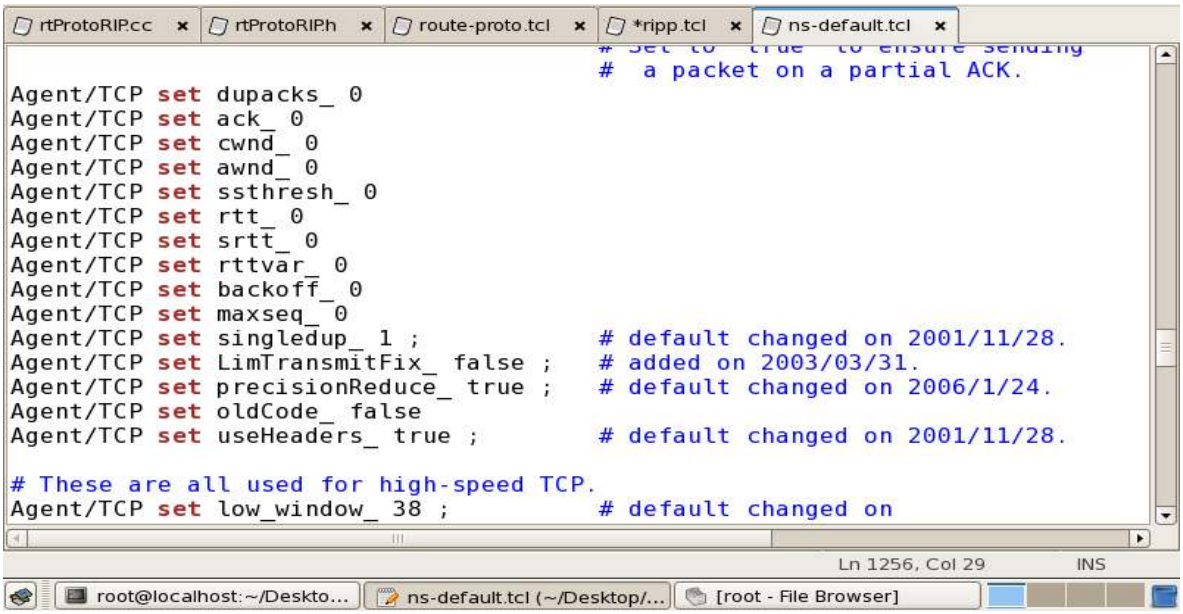
تعتبر أدوات مراقبة الشبكة من الأدوات المهمة جداً لمدير الشبكة والتي يستطيع من خلالها تحديد أداء الشبكة ومعرفة مشاكلها وتسهيل حلها ، ويوجد طرق عدة لمراقبة الشبكة monitoring في الواقع ، ويقوم المحاكي بتوفير آلية للحصول على المعلومات التي يحتاجها مدير الشبكة لمعرفة أداء الشبكة عن طريق متغيرات تتغير قيمها اثناء التنفيذ وقراءة هذه القيم يساعد في الحصول على معلومة تساعد في قياس أداء الشبكة التي تقاس بناء على معايير ومقاييس يتم بواسطتها قياس الأداء وتسمى Performance Metrics ،وسيتم شرح كيفية تطبيق عملية المراقبة للشبكة عن طريق المحاكي في لاحقاً في هذا الباب.

متغيرات الشبكة Network Parameter :

بعد أن تعلمنا في الجزء الاول كيفية كتابة البرنامج ، وقمنا بتطبيق الشبكة التي نريد تصميمها نأتي الآن لنقوم بتغيير بعض سلوك الشبكة وذلك يتم بتغيير متغيرات أو معاملات تسمى Parameters والتي تؤثر على شكل المخرجات وتعتبر من الأشياء المهمة جداً والتي يجب معرفتها في المحاكي والتي من خلالها نستطيع أن نحصل على معلومات عن الشبكة والتي سنستخدمها في عملية التحليل في الأبواب القادمة.

عملية تحليل الشبكة هي من العمليات الاساسية لكل مشروع فاي مشروع في الغالب يكون عبارة عن اقتراح جديد أو module جديد او موجود مسبقاً وتم تحسينه ونريد اختباره عن طريق المحاكي ففي هذه الحالة يجب أن نقوم بتهئية الشبكة على حسب الدراسة ويكون ذلك بالتغيير في هذه المعاملات.

المحاكي NS2 هو محاكي غني جداً بالمعاملات والتي يقوم بوضعها وتخزينها في ملف واحد ويقوم باسترجاع قيم هذه المعاملات منه اثناء تنفيذ البرنامج ، يوجد هذا الملف في المسار ns2.31/tcl/lib/ns-default.tcl وشكله موضح بالصورة التالية :



لاحظ معي شكل المتغيرات تأخذ السيناريو التالي

Agent/TCP set rtt_ 0

معناها ضع القيمة الافتراضية بالنسبة لـ Round Trip Time 0 الخاص بالمضيف TCP

إذا هذه هي أغلب القيم الافتراضية التي يعمل بها المحاكي ولكن كيف يمكن لنا أن نغير هذه القيم وكيف يمكن لنا أن نستفيد من قيمتها في عملية رياضية معينة مثلا لحساب عدد الحزم التي وصلت إلى Sink وغير ذلك ، المهم الآن كيف ذلك؟

يتم ذلك باستخدامها مباشرة أو بتعريف متغير مثلاً ثم ارجاع القيمة له كما يلي :

set interv [$app set interval_]

هذه في حالة ارجاع أو الحصول على القيمة الافتراضية التي يعمل بها المحاكي أما في حالة اسناد قيمة جديدة ونريد من المحاكي أن يستخدمها بدل القيمة الافتراضية نقوم بالآتي:

$app set interval_ 0.02

$app set packetSize_ 1000

وهي الطريقة التي يتم بها اسناد قيمة جديدة بدل القيمة الافتراضية.

سأوضحها في مثال عملي كامل :

```
set  ns [Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
proc  finish { } {
        global ns nf
        close $nf
        $ns flush-trace
        exec nam out.nam
        exit
}
set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0 $n1 5mb 2ms DropTail
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
```

```
set udpsink [new Agent/Null]
$ns attach-agent $n1 $udpsink
set app [new Application/Traffic/CBR]
$app attach-agent $udp
$app set interval_ 0.02
$app set packetSize_ 1000
$ns at 0.0 "$app start"
$ns at 3.0 "$app stop"
$ns at 4 "finish"
$ns run
```

في المثال أعلاه المتغيرات ملونة باللون البني والأزرق.

حسناً بعد هذا سأنتقل بك إلى مستوى أكثر تفصيلاً وسأقول لك الآتي:

يوجد نوعان من المتغيرات المستخدمة مع أي كلاس في المحاكي في لغة TCL تحديداً وهما:

- Class Variable.
- Instance Variable.

ما هما هذين النوعين؟

ذكرت قبل قليل صيغة المتغيرات في ملف ns-default.tcl وهي تذكيراً لك:

Agent/TCP set rtt_ 0

وذكرت أيضاً صيغة التعديل فيها وهي على حسب الصيغة المذكورة:

$agent set rtt_ 1

المتغير الموجود في الصيغة الأولى يسمى Class Variable والمتغير في الصيغة الثانية يسمى Instance Variable .

أكثر من ذلك ذكرت لك أنه يمكنك تغيير القيمة الافتراضية عن طريق الصيغة الثانية في الكود ووضحتها في المثال السابق ،أيضاً يمكنك تغيير القيمة الافتراضية عن طريق الصيغة الأولى ولكن ما هو الفرق ؟؟؟

الفرق هو أن استخدام الصيغة الأولى تقوم بتعديل على مستوى كل الكلاسات الموجودة في برنامجك بينما استخدام الصيغة الثانية تقوم بالتعديل على مستوى الكلاس فقط.

مثلاً : تخيل معي هذا السيناريو!

برنامج يحتوي على 5 عقد ويستخدم بروتوكول TCP في العقدة الأولى والثانية والثالثة والرابعة ويقوم بالارسال إلى العقدة الخامسة.

لتغيير قيمة حجم الحزمة في بروتوكول TCP في العقدة الأولى يتم استخدام Instance Variable ونقوم بالآتي:

```
$tcpnode1 set packetSize_ 2000
```

لتغيير قيمة حجم الحزم لكل المضيفات من نوع TCP يتم استخدام Class Variable:

```
Agent/TCP set packetSize_ 1500
```

الخلاصة تغيير في قيمة Class Variable يقوم بتغيير القيمة الافتراضية لكل المضيفات التي يتم انشاؤها في البرنامج.

تطبيق المراقبة على الشبكة Monitoring:

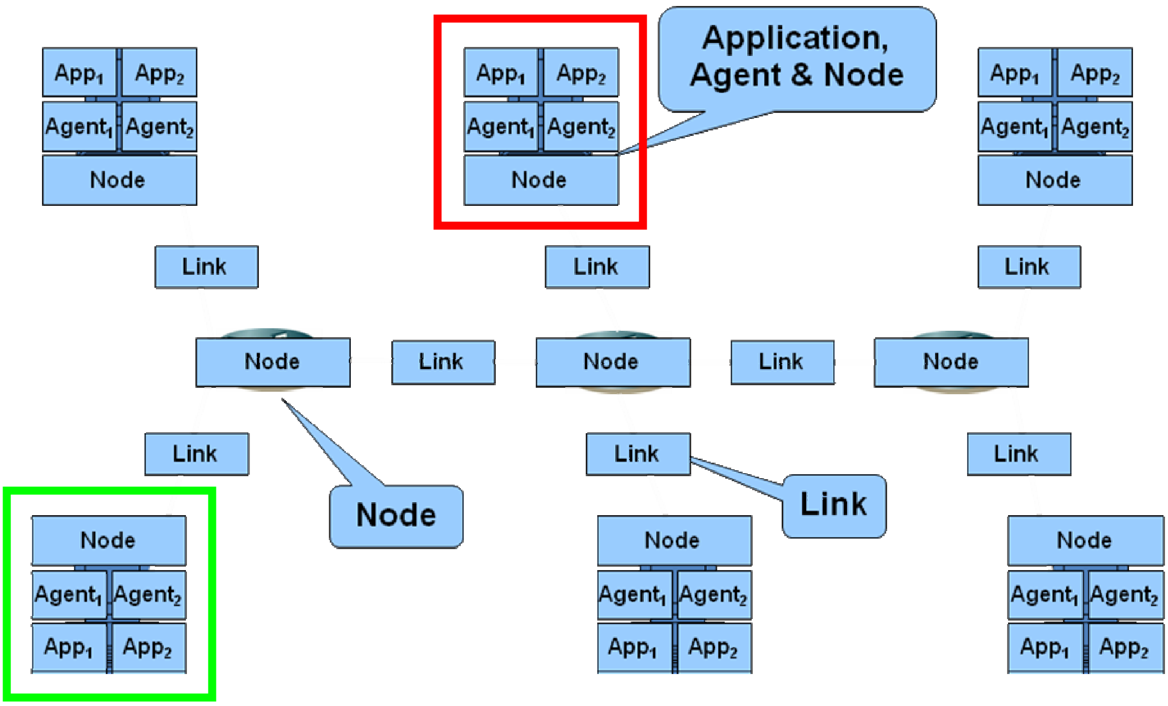
لا يخلو برنامج من برامج المحاكاة من جمع معلومات واستخراج نتائج واحصائيات لمقارنتها مع نظام أو module مقترح ، لذلك يوفر المحاكي مجموعة كبيرة من المتغيرات التي تقوم بالمساعدة في جمع معلومات من الشبكة ومن هذه الطرق المراقبة Monitoring ويوجد طريقتين الأولى تكون على مستوى الصف Queue والثانية على مستوى المضيف Agent.

- المراقبة على مستوى الصف Queue:

أي حزمة في المحاكي بأنواعها routing, data, acknowledge packet ...etc تمر أولاً بالصف فإذا كان الرابط link أو Media غير محجوز يتم حذف الحزمة من الصف أولاً وفي هذه الحالة بالنسبة للصف تكون الحزمة في حدث المغادرة departure من الصف الموجود في العقدة الحالية بعد ذلك يتم ارسالها عبر الرابط إلى المحطة التالية حسب جدول routing table.

- المراقبة على مستوى الحزم Agent:

هيكل تركيب الشبكة يبدأ من مستوى عقدة node ثم مضيف agent ثم تطبيق application كما في الشكل التالي:



في الشكل السابق اذا افترضنا أن العقدة التي باللون الأحمر تريد الارسال إلى العقدة التي باللون الأخضر نسمى العقدة المرسلة المصدر source والعقدة المستقبلة بالمصب sink ، لاحظ أيضاً أن العقدة المرسلة

يمكن أن يكون بها اكثر من مصدر وهذا يعني أنها يمكن أن ترسل وتستقبل في نفس الزمن، الذي أريد أن أصل إليه من كل هذا الشرح أنه يمكن مراقبة المصب sink لحساب عدد الحزم أو كمية bytes التي استقبلها لاستخراج مجموعة من الاحصائيات التي يمكن الاستفادة منها مثلاً في حساب Bandwidth والانتاجية Throughput وغيرها.

نأتي الآن إلى معرفة كيفية تطبيق هذا الكلام بصورة عملية وللقيام بذلك يجب أولاً أن نتكلم قليلاً عن المتغيرات التي سنأخذ القيم منها.

يقوم المحاكي بتخزين المتغيرات الخاصة بالبروتوكولات والنماذج المطبقة في الملف الموجود في المسار ns2.31/tcl/lib/ns-default.tcl والصيغة المكتوبة تم ذكرها سابقاً ، المهم يوجد كلاسين مهمين جداً ويستعملان كثيراً في جمع عدد من الاحصائيات والمعلومات عن الشبكة وهما QueueMonitor and LossMonitor وهما يحتويان على عدد من المتغيرات التي تقوم بتخزين معلومات عن الشبكة والتي موضحة بالصورة التالية:

الكلاس الأول : QueueMonitor

```
QueueMonitor set size_ 0
QueueMonitor set pkts_ 0
QueueMonitor set parrivals_ 0
QueueMonitor set barrivals_ 0
QueueMonitor set pdepartures_ 0
QueueMonitor set bdepartures_ 0
QueueMonitor set pdrops_ 0
QueueMonitor set pmarks_ 0
QueueMonitor set bdrops_ 0

QueueMonitor set qs_pkts_ 0
QueueMonitor set qs_bytes_ 0
QueueMonitor set qs_drops_ 0

QueueMonitor set first_pkt_ 0
QueueMonitor set last_pkt_ 0
```

الكلاس الثاني: LossMonitor

```
Agent/LossMonitor set nlost_ 0
Agent/LossMonitor set npkts_ 0
Agent/LossMonitor set bytes_ 0
Agent/LossMonitor set lastPktTime_ 0
Agent/LossMonitor set expected_ 0
```

في الرسمة الأولى كلاس QueueMonitor والذي يوجد به عدد من المتغيرات نأخذ منها :

size_ : size of queue in bytes.

حجم الحزم الموجودة في الصف بوحدة byte.

pks_ : Number of packets in queue.

عدد الحزم الموجودة حالياً في الصف.

parrivals_ : Number of packets arrive in queue.

عدد الحزم التي وصلت حالياً إلى الصف.

barrivals_ : The size of packets arrive to queue in bytes.

حجم الحزم التي وصلت حالياً إلى الصف بوحدة byte.

pdepartures_ : Number of packets depart from queue.

عدد الحزم التي غادرت حالياً من الصف.

bdepartures_ : The size of packets depart from queue in bytes.

حجم الحزم التي غادرت حالياً من الصف بوحدة byte.

في الرسمة الثانية كلاس LossMonitor الذي يوجد به عدد من المتغيرات أيضاً نأخذ منها على سبيل المثال:

nlost_ : Number of lost packet.

عدد الحزم المفقودة.

npkts_ : Number of packets.

عدد الحزم التي استقبلت.

bytes_ : The size of packets have been received in bytes.

حجم الحزم التي استقبلت بوحدة byte.

أما عن كيفية الحصول على قيمها واستخدامها في البرنامج فهي كالتالي:

```
set ns [new Simulator]

#Open the output file
set f0 [open out0.tr w]

#Create 5 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Connect the nodes
$ns duplex-link $n0 $n2 1Mb 2ms DropTail
$ns duplex-link $n1 $n2 1Mb 2ms DropTail
$ns duplex-link $n2 $n3 1Mb 2ms DropTail
```

```
#Define a 'finish' procedure
proc finish {} {
      global f0
      #Close the output files
      close $f0
      exit 0
}

#Create a UDP agent and attach it to the node
set source [new Agent/UDP]
$ns attach-agent $n0 $source

#Create an Expoo traffic agent and set its configuration parameters
set traffic [new Application/Traffic/CBR]
$traffic set packetSize_ 1000
$traffic set rate_ 100k

# Attach traffic source to the traffic generator
$traffic attach-agent $source

#Create traffic sink and attach them to the node n3
set sink [new Agent/NULL]

$ns attach-agent $n3 $sink

#Connect the source with the sink
$ns connect $source $sink

# set up the queue-monitor

#monitor-queue node0 node1 File {sampleInterval}

set qmon [$ns monitor-queue $n2 $n3 $f0 0.5]

# sampleInterval is optional and defaults to 0.1 seconds


#Define a procedure which periodically records the number of packets
proc recordpkt {} {
      global qmon f0
      #Get an instance of the simulator
      set ns [Simulator instance]

      #Set the time after which the procedure should be called again
      set time 0.5

      #How many bytes have been received by queue?
      set pktbytes [$qmon set size_]

      #Get the current time
      set now [$ns now]

      #put no of packet in file
      puts $f0 "$now $pktbytes"
```

```
        #Re-schedule the procedure
        $ns at [expr $now+$time] "recordpkt"
}
#Start logging the number of packets
$ns at 0.0 "recordpkt"

#Start the traffic source
$ns at 10.0 "$traffic start"

#Stop the traffic source
$ns at 50.0 "$traffic stop"

#Call the finish procedure after 60 seconds simulation time
$ns at 60.0 "finish"

#Run the simulation
$ns run
```

قمنا بفتح ملف للكتابة ثم قمنا بتفعيل عملية المراقبة على الرابط ما بين العقدة الأولى والثانية (n2,n3) والدالة تقوم باستقبال العقدة الأولى والثانية ثم الملف ومقدار هذا المقدار يتم استخدامه لتخزين وحساب النتائج كل 0.5 ثانية لذلك يسمى sample Interval معدل اختبار العينة.

يوجد أكثر من كلاس تأخذ خصائص الكلاس QueueMonitor وتختلف في طريقة جمع البيانات فيمكن جمع معلومات ليس فقط بين طرفي link أو رابط معين بل يمكن أن تكون بالنسبة لمسار تدفق (flow) البيانات بين مرسل ومستقبل والتي سنتكلم عنها في مواضيع متقدمة في الأجزاء القادمة من هذه السلسلة.

أما الآن ننتقل إلى الكلاس الثاني وهو LossMonitor ونقوم بشرحه بمثال عملي كامل وسنوضح خطوات استخدام الكلاس باللون الأحمر:

```
set ns [new Simulator]

#Open the output file
set f0 [open out0.tr w]
set f1 [open out1.tr w]
set f2 [open out2.tr w]

#Create 5 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

#Connect the nodes
$ns duplex-link $n0 $n3 1Mb 100ms DropTail
$ns duplex-link $n1 $n3 1Mb 100ms DropTail
$ns duplex-link $n2 $n3 1Mb 100ms DropTail
$ns duplex-link $n3 $n4 1Mb 100ms DropTail

#Define a 'finish' procedure
proc finish {} {
     global f0 f1 f2

     #Close the output files
     close $f0
     close $f1
     close $f2

      exit 0
}

#Define a procedure that attaches a UDP agent to a previously created
node
#'node' and attaches an Expoo traffic generator to the agent with the
#characteristic values 'size' for packet size 'burst' for burst time,
#'idle' for idle time and 'rate' for burst peak rate. The procedure
connects
#the source with the previously defined traffic sink 'sink' and
returns the
#source object.
proc attach-expoo-traffic { node sink size burst idle rate } {

     #Get an instance of the simulator
     set ns [Simulator instance]

     #Create a UDP agent and attach it to the node
     set source [new Agent/UDP]
     $ns attach-agent $node $source
```

```
     #Create an Expoo traffic agent and set its configuration
parameters
     set traffic [new Application/Traffic/Exponential]

     $traffic set packetSize_ $size
     $traffic set burst_time_ $burst
     $traffic set idle_time_ $idle
     $traffic set rate_ $rate

      # Attach traffic source to the traffic generator
      $traffic attach-agent $source

     #Connect the source and the sink
     $ns connect $source $sink
     return $traffic
}


#Define a procedure which periodically records the bandwidth received
by the
#three traffic sinks sink0/1/2 and writes it to the three files
f0/1/2.
proc record {} {
      global sink0 sink1 sink2 f0 f1 f2

     #Get an instance of the simulator
     set ns [Simulator instance]

     #Set the time after which the procedure should be called again
      set time 0.5

     #How many bytes have been received by the traffic sinks?
      set bw0 [$sink0 set bytes_]
      set bw1 [$sink1 set bytes_]
      set bw2 [$sink2 set bytes_]

     #Get the current time
      set now [$ns now]

     #Calculate the bandwidth (in MBit/s) and write it to the files
      puts $f0 "$now [expr $bw0/$time*8/1000000]"
      puts $f1 "$now [expr $bw1/$time*8/1000000]"
      puts $f2 "$now [expr $bw2/$time*8/1000000]"

     #Reset the bytes_ values on the traffic sinks
      $sink0 set bytes_ 0
      $sink1 set bytes_ 0
      $sink2 set bytes_ 0

     #Re-schedule the procedure
      $ns at [expr $now+$time] "record"
}
```

```
#Create three traffic sinks and attach them to the node n4
set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]

$ns attach-agent $n4 $sink0
$ns attach-agent $n4 $sink1
$ns attach-agent $n4 $sink2

#Create three traffic sources
set source0 [attach-expoo-traffic $n0 $sink0 200 2s 1s 100k]
set source1 [attach-expoo-traffic $n1 $sink1 200 2s 1s 200k]
set source2 [attach-expoo-traffic $n2 $sink2 200 2s 1s 300k]

#Start logging the received bandwidth
$ns at 0.0 "record"

#Start the traffic sources
$ns at 10.0 "$source0 start"
$ns at 10.0 "$source1 start"
$ns at 10.0 "$source2 start"

#Stop the traffic sources
$ns at 50.0 "$source0 stop"
$ns at 50.0 "$source1 stop"
$ns at 50.0 "$source2 stop"

#Call the finish procedure after 60 seconds simulation time
$ns at 60.0 "finish"

#Run the simulation
$ns run
```

هذا الكود يقوم بتسجيل وقراءة Bandwidth في الشبكة ويقوم بتخزينه في ملفات واستخدم في ذلك كلاس LossMonitor ، ففي البداية يقوم بفتح الملفات ثم اسناد الكلاس كمصب بالنسبة للـ Agent بالنسبة لتطبيق من نوع UDP فهذا الكلاس لا يعمل إلا مع UDP وأخيراً قراءة عدد bytes المستقبلة وتسجيلها في الملف.

الباب الثالث

تحليل الشبكة

Analysis of Network

تحليل الشبكة Analysis of Network

بعد أن تعرفنا على طريقتين من طرق الحصول على معلومات من الشبكة يأتي الآن دور التحليل لمعرفة صحة النتائج.

تحليل الشبكة في محاكي الشبكات NS2 يتم بتعريف ملف لكتابة أحداث الشبكة به يسمى هذا الملف بملف التحليل Trace File ، من هذا الملف نستطيع تحليله بأكثر من طريقة ، فمن الطرق أن نقوم باستخدام واحدة من الأدوات الموجودة في نظام اللينيكس وهي grep وهي تقوم بالبحث عن نمط معين في الملف وتقوم بطباعة السطر الذي تمت مطابقة النمط به.

يتم حساب الأسطر مثلاً الخاصة ببروتوكول TCP باستخدام هذه الأداة سيكون الناتج جميع الأسطر التي تحتوي على كلمة TCP والتي هي واحدة من الحقول الموجودة في ملف Trace File بعد ذلك يتم استخدام النتائج في حساب مثلا عدد الحزم التي أرسلت من نوع TCP وهكذا.

أيضاً من الطرق التي يمكن استخدامها استخدام أداة الفلترة Awk وهي عبارة عن لغة مستقلة بذاتها لها صيغة معينة في كتابة الكود الخاص بها مثلها مثل لغة Perl التي يمكن ايضاً استخدامها في عملية التحليل.

كذلك من الطرق استخدام برامج تقوم باستقبال ملف Trace File وتقوم بتحليله واستخراج النتائج على شكل رسوم بيانية ، ومن أشهر هذه البرامج Tracegraph الذي يقوم بتحليل الملف واستخراج عدد كبير جداً من التقارير على شكل إحصائيات ورسوم بيانية ثنائية وثلاثية الأبعاد.

والآن سنتحدث عن:

- كيفية إنشاء ملف التحليل.
- ملف التحليل.
- استخدام الأداة grep في التحليل.
- استخدام لغة awk في التحليل.

كيفية إنشاء ملف التحليل:

لانشاء ملف التحليل يجب أولاً أن نقوم بفتح ملف للكتابه به كالآتي:

```
set tf  [open out.tr]
```

يمكن عن طريق دالة trace-all تسجيل جميع أحداث الشبكة في كل الشبكة ويمكن أيضاً تحديد رابط معين عن طريق دالة trace-queue والتي تحدد رابط معين في الشبكة وتتم هذه الخطوة كالأتي:

```
$ns trace-all $tf
```

Or

```
$ns trace-queue $n0 $n2 $tf
```

في الجزء الأول من السلسلة تحدثنا عن الدالة namtrace-all لاحظ هي نفس الدالة فقط لكن تستخدم للكتابة في ملف العرض nam file الذي سيعرض ببرنامج Nam ويمكن ايضاً استخدام الدالة:

```
$ns namtrace-queue $n2 $n4 $nf
```

لعرض الأحداث ما بين العقدتين الثانية والرابعة في برنامج Nam.

بقي أن نقوم بإغلاق الملفات التي قمنا بفتحها لحفظ النتائج بها وذلك في دالة finish وتكون كالتالي:

```
proc finish {} {
        global ns nf tf
        $ns flush-trace
        close $nf
        close $tf
}
```

ـ ملف التحليل Trace File:

يعتبر ملف التحليل هو الملف الذي يقوم برنامج المحاكي NS2 بتخزين جميع الأحداث المتعلقة بمحاكاة الشبكة به في صيغة معينه عبارة عن مجموعة من الأعمدة.

صيغة ملف التحليل :

يوجد أكثر من صيغة لهذا الملف فهناك صيغة أخرى خاصة بالشبكات اللاسلكية ونحن هنا فقط سنتحدث عن الشبكات السلكية وصيغتها كالآتي:

<event>	<time>	<from>	<to>	<pkt>	<size>	-- <fid>	<src.port>	<dst.port>	<seq>	<pktID>
+	1.0	1	2	cbr	210	---- 0	1.0	3.1	0	0
-	1.2	1	2	cbr	210	---- 0	1.0	3.1	0	0
r	2.0234	1	2	cbr	210	---- 0	1.0	3.1	0	0
d	3.04	2	3	cbr	210	---- 0	2.0	3.1	0	1

تحتوي الصيغة كما هي موضحة بالمثال على :

<event> : وهو نوع الحدث ويأخذ أربعة قيم + بمعنى دخول الحزمة إلى الصف Queue (enqueue) والذي تحدثنا سابقاً بأن الحزمة تدخل أولاً إلى الصف ثم يتم ارسالها ، - بمعنى خروج الحزمة من الصف (dequeue) ، r بمعنى وصول الحزمة إلى الوجهة التي تقصدها أو الهدف (receive) ، d بمعنى هذه الحزمة تم حذفها إما بسبب الزحمة أو انقطاع رابط من الشبكة وغيره (drop).

<time> : زمن المحاكاة بمعنى أن الحدث حصل في هذا الزمن.

<from> : بمعنى أن هذا الحدث صدر عن الحزمة المرسلة رقم كذا يعني العقدة الأولى في المثال.

<to> : بمعنى أن هذا الحزمة متجهه إلى العقدة رقم كذا.

<pkt> : نوع الحزمة ففي هذا المثال CBR والتي يتم اسنادها مع بروتوكول UDP، فإذا كانت حزمة خاصة ببروتوكول TCP سيكتب بدلها tcp.

<size> : حجم الحزمة.

---- : تسمى هذه الخانات flags يتم استخدامها مع الشبكات اللاسلكية.

<fid> : flow ID كلمة flow تعني مسار تدفق البيانات من مصدر معين إلى هدف معين ، يعني مثلاً العقدة رقم 1 تريد الارسال إلى العقدة رقم 5 فمن الطبيعي أن تمر هذه الحزمة بعدد من العقد مثل 2,3,4 فنسمي مسار هذه الحزمة من لحظة خروجها من العقدة رقم 1 إلى العقدة رقم 5 بمسار تدفق البيانات flow ويتم استخدام هذا الرقم أي flowID في عملية تلوين الحزم المتجهة في هذا المسار بلون معين يتم تعريفه في البرنامج وتظهر نتائجه في برنامج Nam ، أيضاً يتم استخدامه في تمييز مسار البيانات لمعرفة مسار الحزمة كاملاً والذي سيستخدم في عملية المراقبة لجمع معلومات من الشبكة في حالة مراقبة مسار بيانات معين في الشبكة.

<src.port><dst.port> : لكي نوضح معنى هذين الحقلين يجب أولاً أن نوضح معنى كلمة port وما المقصود بها في المحاكي وفي ملف التحليل:

أنظر إلى هذه الرسمة جيدا والتي تعبر عن شكل العقدة داخلياً:

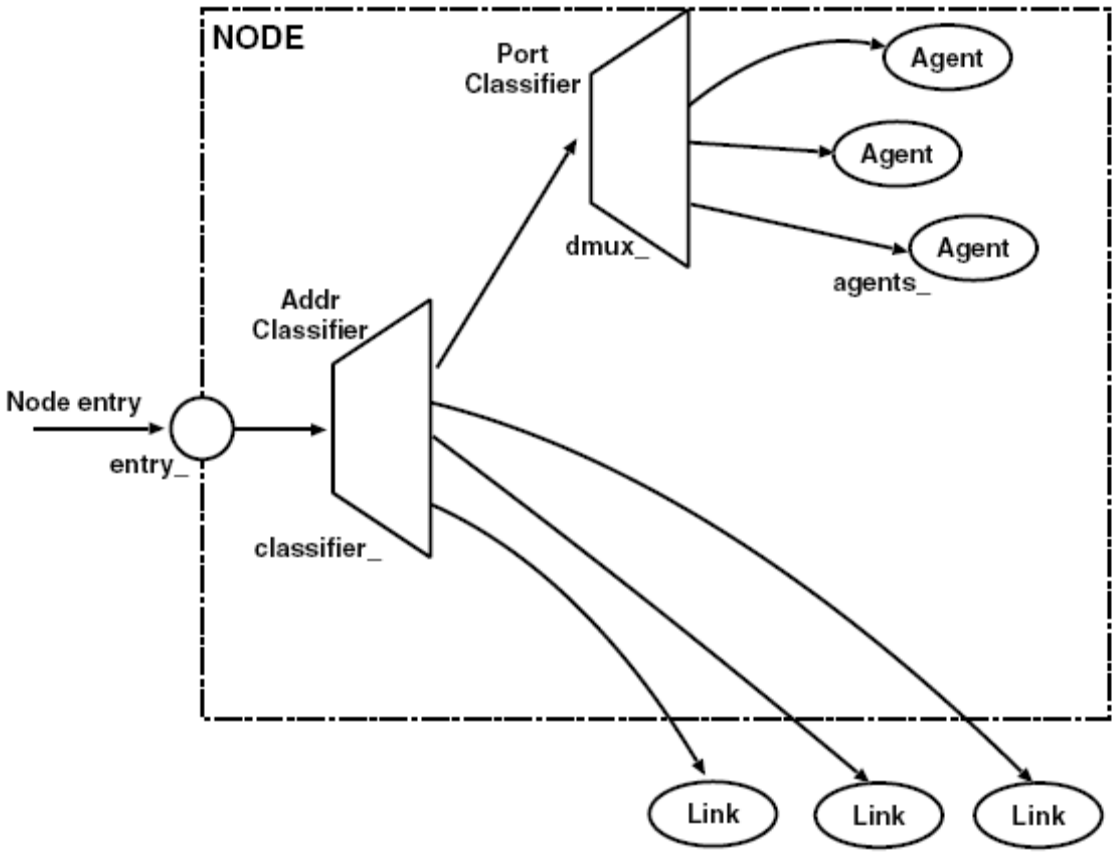


Figure 4.1: Structure of an Unicast Node

لاحظ معي في الرسمة اعلاه يوجد classifier بداخلة يوجد addr_classifier مرتبط به port Classifier ولاحظ أيضاً Agents مرتبطة بـ dmux_ في port classifier ويوجد أكثر من agent ، هذا الشكل هو الهيكل لعقدة تقوم بارسال Unicast.

طيب؟؟ ما معنى هذه الرسمة وما المقصود بهذه الأسماء؟؟؟

لنبدأ أولا من مدخل العقدة وهنا يسمى entry_ وتعتبر مدخل للعقدة منها يتم توجيه الحزمة إلى المصنف classifier والقيام ببعض العمليات الضرورية.

ما هو المصنف classifier ؟

المصنف هو عبارة عن كلاس يحتوى على عدة أنواع من المصنفات مثل:

address classifier, multicast classifier, multipath classifier,…etc

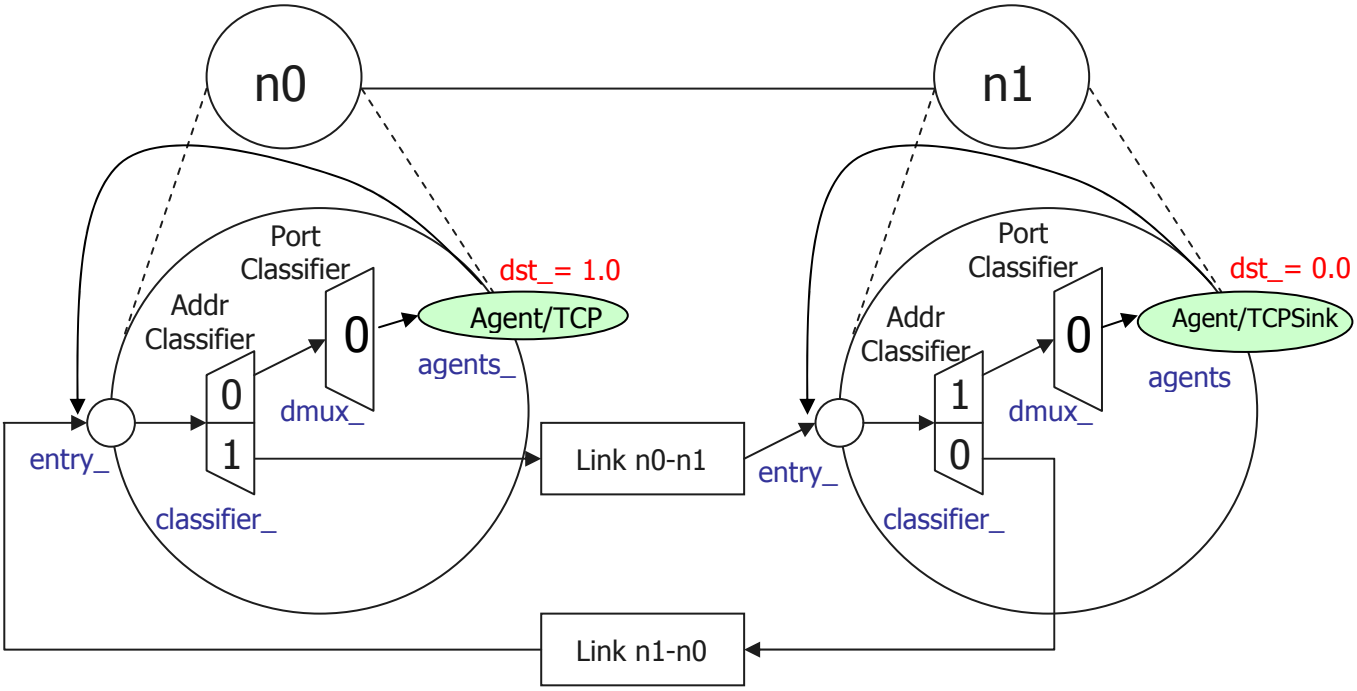
والتي يستخدم كل منها في تطبيقات معينة وفي مثالنا لأننا نقوم بارسال سلكي وبين جهازين فقط نستخدم address classifier.

ما هو dmux_؟

يوجد Port Classifier في الأعلى وتحت الرسمة يوجد كلمة dmux_ هذه الكلمة اختصار لـ DeMultiplexer وهو بمعنى المجمع وهو معروف في الالكترونيات والذي يستقبل كمية من المدخلات ويقوم بتنشيط طرف واحد فقط للخرج على حسب اشارة معينة، هنا تم استخدام نفس المفهوم لأن الحزمة تكون متجهة لمضيف واحد فقط وقادمة من مصدر معين ويوجد في العقدة أكثر من مضيف كل مضيف موجود في port محدد فلكي تتجه الحزمة إلى المضيف الصحيح يتم تنشيط فقط المضيف الصحيح على حسب عنوان port الموجود في الحزمة فإذا كان 1 يتم تنشيط port 1 وإذا كان 5 ينشط port 5 وهكذا.

هذا الشكل الذي أمامك هو عقدة بطرفها تصل الحزم ويتم ارسالها إلى روابط link أو استقبالها في نفس العقدة إذا كانت هي الهدف .

يبدو الكلام غامض قليلاً حسناً انظر الرسمة كمثال يوضح أكثر:



تخيل معي هذا السيناريو:

تنطلق الحزمة من المضيف (Agent/TCP) في العقدة الأولى n0 فتذهب إلى نقطة entry_ والتي تقوم بامرار الحزمة إلى المصنف Classifier والذي ينظر إلى الجزء العلوي من العنوان فقط الخاص

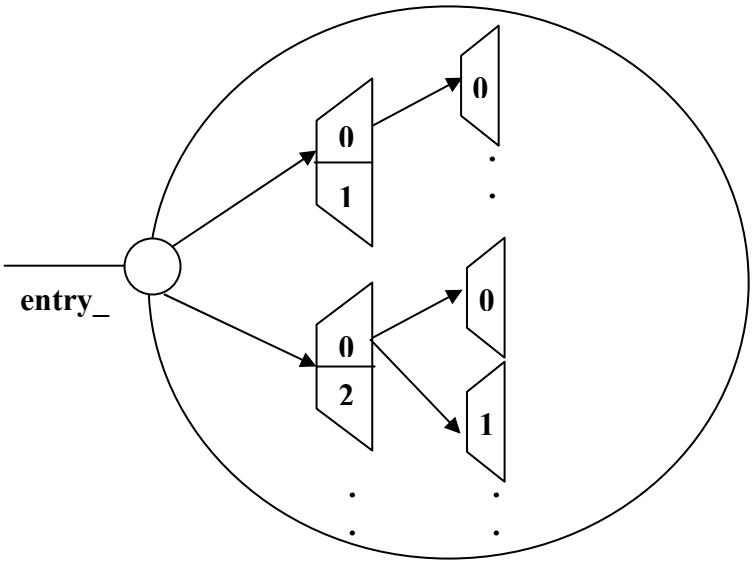
بالهدف dst=1.0 فيأخذ 1 ويقوم بامرار الحزمة في المسار رقم 1 فتذهب downstream (المقصود بها أن الحزمة صادرة من العقدة ومتجهة إلى عقدة اخرى وعكسها upstream ويعني ان الحزمة متجهة نحوها مثل acknowledge فهي تعود إلى المصدر كـ upstream) إلى العقدة التالية عبر الرابط فتصل إلى نقطة _entry في العقدة الثانية فتذهب إلى المصنف فينظر إلى الجزء العلوي (highest bit) وهو 1 فيقوم بامرارها عبر المسار 1 والذي يدخل إلى المضيف فيذهب أولاً إلى port classifier الذي ينظر إلى الجزء الآخر من العنوان وهو 0 فيقوم باسناده إلى المضيف الموجود في port 0 ويقوم المصب بالعملية العكسية باستخدام dst_=0.0 ويقوم بارسال acknowledge لتأكيد وصول الحزمة في حالة استخدام بروتوكول TCP باتجاه upstream.

فهذا هو معنى src.port and dst.port المذكورة في ملف التحليل.

نقطة مهمة: قد تتساءل كيف يعرف هذه الأرقام وعلى أي أساس يتم وضعها ؟

يتم وضع هذه الأرقام على حسب الرابط حيث يتم وضع رقم العقدة في الجزء العلوي من addr_calssifier ورقم العقدة الثانية في الطرف الآخر من الرابط في الجزء السفلي فمثلاً اذا كان بين العقدة n0,n1 يكون مثل الشكل السابق وإذا كان الرابط ما بين العقدتين n2,n4 يكون الجزء العلوي من addr_classifier 2 والجزء السفلي 3 ومثله في الطرف الآخر فتكون 3 ثم 2 .

فمثلاً إذا كان لدينا عقدة n1 مربوطة بعقدتين n1,n2 سيكون وضع العقدة n0 بالشكل التالي:



فلذلك يمكن أن يكون في العقدة الواحدة أكثر من مضيف agent وكل مضيف موجود في port معين وهكذا فبهذه الطريقة يمكن ربط العقدة الواحدة بأكثر من عقدة لبناء شبكة وكذلك يمكن ارسال حزمة إلى أكثر من مضيف في عقد أخرى في نفس الوقت وهو ما يسمى broadcast.

نقطة أخرى مهمة :اذا كان كل سطر من أسطر ملف التحليل يعبر عن حدث فكيف نعرف أن هذه الحزمة ارسلت بواسطة العقدة رقم 2 ومتجهة إلى العقدة رقم 5 ، كيف ؟؟

+ 1.0 1 2 cbr 210 ---- 0 1.0 3.1 0 0

لاحظ src.port and dst.port وهي التي تجيب عن السؤال وهي تمثل مصدر الحزمة ومن أي ميناء port خرجت وإلى ميناء ستذهب في الهدف dst.port.

<seq><pktID> : يتم استخدام sequence number and packet id في تعريف الحزمة لتمييزها عن باقي الحزم ، فيتم استخدام packet id لتمييز الحزمة عن باقي الحزم وsequence number الذي يتم استخدامه في عملية الارسال للتأكد من أنه لن يتم ارسال نفس الحزمة أكثر من مرة يتم ايضاً ذكر هذا الرقم في مجال سرية المعلومات والتي تكون في إعادة الارسال لحزمة ما والتي تعتبر هجوم على الشبكة يسمى reply attack فهذا المحاكي يقوم بمحاكاة ولا يتقيد بشكل شبكة معينة بل يقوم بوضع معلومات عامة يمكن استخدامها في أكثر من مجال.

استخدام الأداة grep في التحليل:

يوفر نظام linux مجموعة من الأدوات والأوامر التي يمكن استخدامها في مجالات كثيرة ومن هذه الأوامر أمر grep الذي وظيفته الأساسية البحث في الملفات عن نمط معين فإذا وجد النمط في كلمة من كلمات السطر يتم طباعة السطر بأكمله كناتج تنفيذ.

حسناً كيف نستخدمها ؟؟

بما أن ملف التحليل يحتوي على معلومات كثيرة فمن الممكن أن أقوم باستخراج أسطر معينه حسب نمط معين كناتج يمكن كتابته كتقرير والذي يمكن رسمه بيانياً في ما بعد.

صيغة هذا الأمر:

grep pattern file

لنأخذ أمثلة باستخدام هذا الأمر والذي سنقوم باستخراج عدد من التقارير بواسطته:

- حساب عدد الحزم المرسلة في الشبكة:

لحساب ذلك نكتب الأمر التالي :

grep + out.tr or grep ^+ out.tr

حيث سيقوم بطباعة جميع الأسطر التي تحتوي على علامة +، نحن نعرف أن + هي الحزم التي أرسلت من العقدة ودخلت إلى الصف.

نأتي الآن لحساب عدد هذه الأسطر هل سنقوم بحساب عددها يدوياً ! كلا يوفر نظام اللينيكس أيضاً أمراً أخراً يسمى wc(word count) والذي يقوم بحساب عدد الأسطر والكلمات والأحرف في ملف ما.

كذلك يوجد ميزة رائعة جداً في أوامر اللينيكس وهي ما تسمى pipe والتي معناها أنه يمكن استخدام أكثر من أمر في أمر واحد بحيث يكون ناتج كل أمر مدخل إلى الأمر الذي يليه والعلامة موجودة في لوحة المفاتيح يمين حرف الطاء أو يمين علامة يساوي.

إذا سيكون الأمر لحساب عدد الأسطر وباستخدام علامة pipe كالتالي:

grep ^+ out.tr | wc -l

الخيار l– معناه line فلحساب عدد الأسطر نكتب l– ولحساب عدد الكلمات نكتب w– ولحساب عدد الأحرف نكتب الخيار c– ولحسابهم جميعا نكتب فقط wc فسيقوم الأمر بطباعتهم جميعاً.

- حساب عدد الحزم المرسلة في الشبكة فقط من نوع tcp :

في المثال السابق قمنا بحساب عدد الحزم المرسلة فيمكن من التنيجة السابقة حساب فقط الأسطر التي نوعها tcp ويتم ذلك كالآتي:

grep ^+ out.tr | grep tcp | wc –l

or grep ^+ out.tr | grep tcp | wc –l > output.tr

هنا من الأمر الأول سينتج كل الحزم المرسلة وفي الأمر الثاني سينتج الحزم المرسلة فقط من نوع tcp وفي الأمر الأخير سيتم حساب عدد الأسطر الناتجة وطباعتها على الشاشة أو طباعة المخرجات في ملف output.tr عن طريق علامة التوجيه redirect.

لاحظ معي أن هذه الأداة تقوم فقط بالعمل على مستوى الصفوف ولا تقوم بالعمل على مستوى الأعمدة أي البحث عامودياً في الملف فلذلك عدد التقارير التي يمكن الحصول عليها من هذه الأداة محصورة جداً فيمكن على سبيل المثال حساب تقارير أخرى مثل :

- عدد الحزم المفقودة من نوع cbr. (بحساب عدد grep ^d out.tr : d)
- عدد الحزم التي استقبلت في الشبكة . (بحساب عدد grep ^r out.tr : r)
- عدد الحزم المتوجهة إلى العقدة 2

وغيرها.

استخدام لغة awk في التحليل:

لغة awk هي عبارة عن لغة برمجة نوعها script language وطريقة عملها تختلف قليلاً عن لغات البرمجة الأخرى التي تندرج تحت هذا النوع ويغلب استخدام هذه اللغة في علمية الفلترة للملفات، وتتميز بأنها يمكنها العمل على مستوى الملفات عامودياً والتي تفتقر إليها الأداة grep .

فلن نقوم بشرح هذه الأداة لأنها كما ذكرت لغة برمجة قائمة بذاتها.

صيغة البرنامج في اللغة :

```
BEGIN { …}
{
….
}
END { …}
```

حيث أنه ما بين أقواس BEGIN يتم تعريف المتغيرات والعدادات واعطائها القيم الابتدائية ، وما بين أقواس END يتم كتابة شكل المخرجات من طباعة بشكل معين وغيره، أما الأقواس الموجود في المنتصف فيتم تنفيذها **بعدد أسطر الملف**.

في ملف التحليل يحتوي على آلاف الأسطر فإذا قمنا بكتابة برنامج عادي مثلاً بلغة الجافا أو لغة السي وقمنا بقراءة السطر ثم حسابه حسب شرط معين فإن البرنامج لمئة سطر ممكن لكن لآلاف الأسطر قد يقف البرنامج من التنفيذ أو قد يأخذ وقتاً طويلاً جداً، لكن في هذه اللغة هي لغة سريعة وتستعمل مع آلاف الأسطر لأنها بالأساس صممت لذلك فهي تقوم بتنفيذ الأوامر الموجودة في الأقواس الموجودة ما بين قوسي BEGIN and END بعدد أسطر الملف مما سيوفر الكثير من الوقت ويعطي نتائج بسرعة.

في التحليل يوجد هناك تقارير لا يمكن حسابها باستخدام الأداة grep لانها لا تستطيع العمل على مستوى الملف عامودياً فمثلاً الانتاجية والتي نحتاج فيها لمعرفة الزمن مع حجم الحزم التي استقبلت كذلك propagation delay والذي يمكن حسابه عن طريق الأحداث والزمن .

لتنفيذ البرنامج باستخدام لغة awk نقوم بكتابة التالي:

```
awk -f avgStat.awk out.tr
```

قمنا بامرار ملف اللغة عن طريق الخيار f– للأمر awk ثم المدخلات هي ملف التحليل.

لن أتطرق لكيفية كتابة هذه التقارير باستخدام هذه اللغة فهي متوفرة على الانترنت وتحتوي على تقارير أكثر من التي ذكرتها فيمكن البحث عنها وتنزيلها بدلاً من تعلم اللغة ثم محاولة كتابتها من جديد.

الملف المذكور avgStat.rar يمكن البحث عنه في الانترنت فيوجد به التقارير:

- Avg Throughput.
- Avg Delay.
- Avg Jitter.
- Instant Throughput.
- Instant Delay.

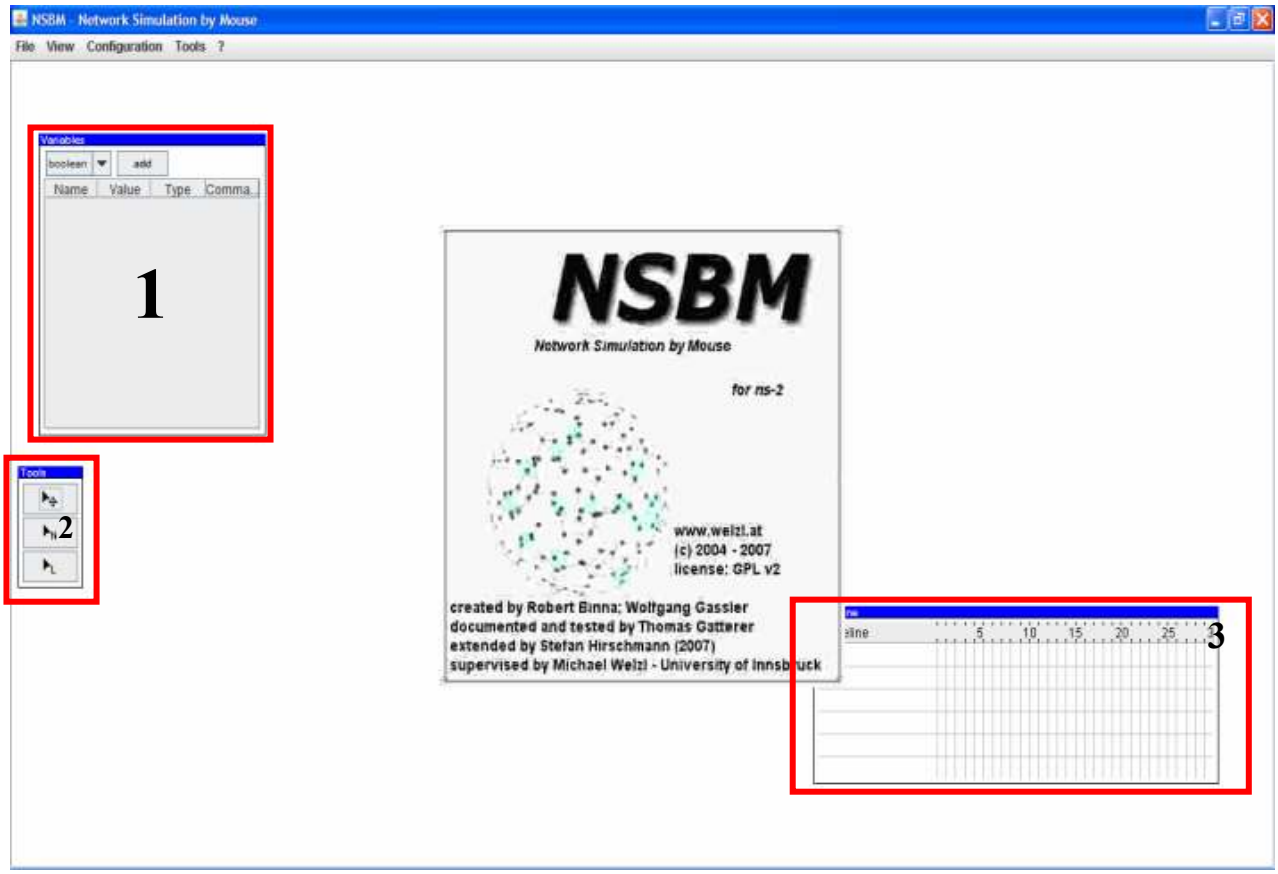
ويوجد ملف readme.txt يحتوي على أمثلة وكيفية استخدام هذه الملفات مع الأمثلة.

واجهة البرنامج تحت المجهر:

يعتبر محاكي الشبكات NS2 من المحاكيات التي انتشرت انتشاراً واسعاً حول العالم وكانت المشكلة للطلبة الذين يرغبون في تعلمه هو صعوبة الاستخدام بالاضافة لضعف الواجهة الرسومية التي من خلالها يمكن انشاء الشبكة فقمت ببحث شخصي في هذا الموضوع وقمت بكتابة نتائج هذا البحث بين أيديكم والتي تركز على استخدام المحاكي في التعليم:

اجريت العديد من الدراسات حول استخدام المحاكي في التعليم ولكن هل يمكن بشكله هذا أن يساعد الطلبة على فهم مادة الشبكات باستخدام NS2 ، يوجد عدد من المشاريع التي اهتمت بهذا المجال فمن هذه المشاريع **NSBM(Network Simulation By Mouse)** الذي بدأ في عام 2004-2007 والهدف منه القدرة على استخدام وعمل برامج باستخدام المحاكي NS2 بدون كتابة سطر برمجي واحد والاكتفاء فقط بزر الماوس وتمت دراسة معملية على الطلاب ويوجد ورقة علمية تم نشرها باسم المشروع تشرح النتائج التي تم الوصل لها ، ولتشغيله ستحتاج فقط لتركيب jdk1.6 الخاص بلغة الجافا لتشغيل ملفات jar الذي هو امتداد البرنامج nsbm.jar.

شكل البرنامج :



يمكن هذا البرنامج من انشاء شبكة عن طريق استخدام الماوس بالضغط في المساحة البيضاء ورسم الشبكة واختيار نوع المؤشر ما بين رابط وعقدة ومؤشر يمكنكم تحميل البرنامج وتجربته فهو من البرامج المساعدة على انشاء الشبكة بالإضاف لذلك والميزة المهمة جداً يقوم بتوليد الكود المقابل لما تنشئه ويمكنك حفظ الملف بكل سهولة.

شرح الرسمة :

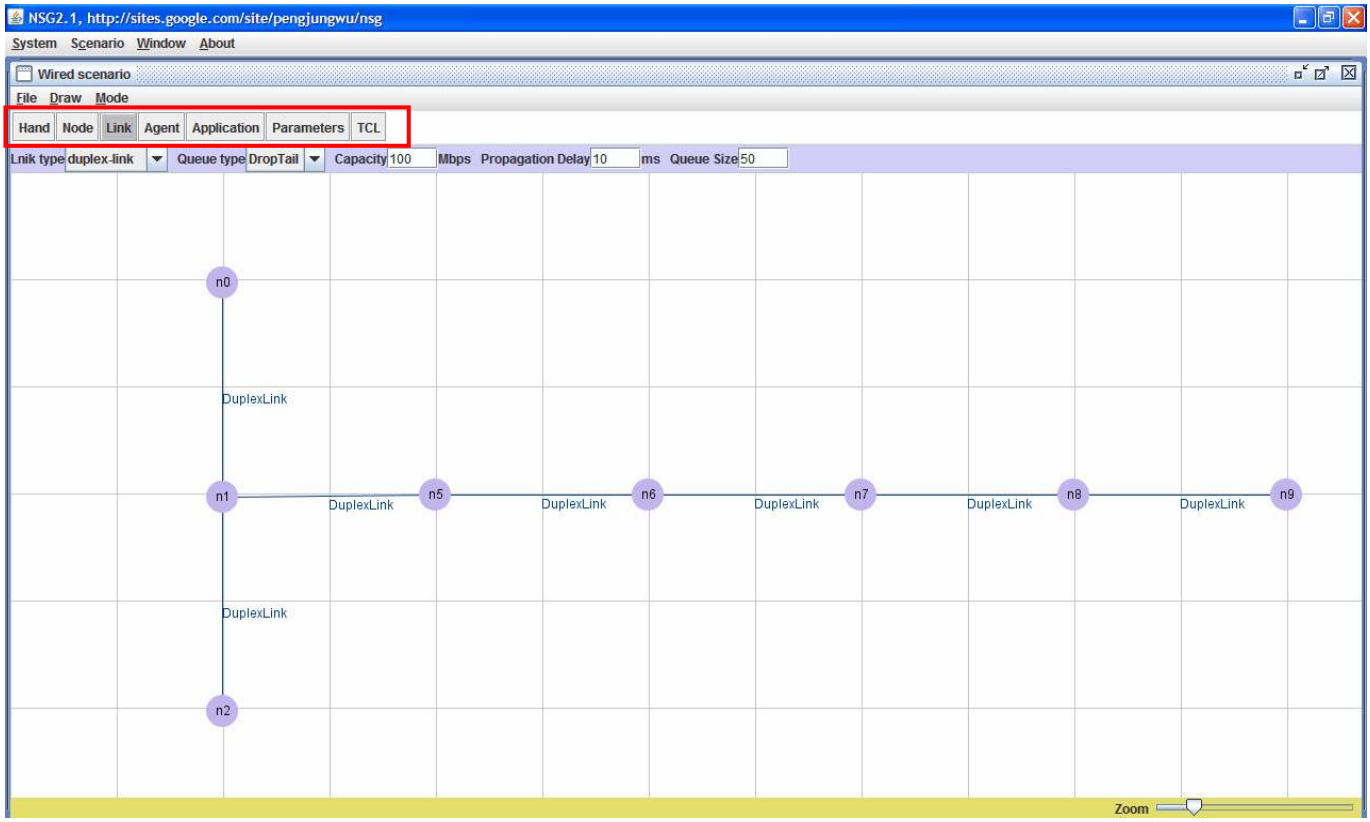
1- يمكن هذا الجزء من انشاء متغيرات يمكن استخدامها في عملية التهئية.

2- هذا الجزء يحتوي على ثلاثة أزرار الأول يمثل سهم وبأسفله علامة + والغرض منها تحريك العقد لتنظيمها في المساحة البيضاء، الزر الثاني سهم واسفله حرف N واختصار لعقدة Node ويمكن من رسم العقدة عن طريق الضغط في المساحة البيضاء بعد تحديده، الزر الثالث سهم أسفله حرف L وهو اختصار Link ويمكن من إنشاء رابط ما بين عقدتين عن طريق تحديد العقدة الأولى بالضغط عليها والضغط على العقدة الثانية بالضغط عليها أيضاً فينشأ الرابط بينهما.

3- تمثل منطقة خط زمن المحاكاة والذي عن طريقه يتم جدولة الأحداث أي التطبيقات بعد انشاءها في الشبكة بالسحب على امتداد الخط لتحديد بداية ارسال التطبيق ومدته.

المشروع الثاني هو **NSG(NS2 Scenario Generator)** وهو مشروع تم في سنة 2008 والهدف منه أيضاً تسهيل التعامل مع المحاكي لكن الفارق الذي تميز به أنه يقسم المشروع على طبقات layers طبقة العرض وطبقة العقدة وطبقة الرابط وطبقة المضيف وطبقة التطبيق وعند اختيار أي من هذه الطبقات فإن التحكم سيكون خاص بهذه الطبقة يعني مثلاً إذا قمنا باختيار طبقة العقدة فيمكن عن طريقها انشاء عقد ولا يمكن في طبقة اخرى كذلك من المزايا الرائعة أنه يقوم بتوليد سيناريو فيوفر إذا كنا في طبقة العقدة انشاء عقد بشكل أفقي أو رأسي أو على شكل شبكة grid أو عشوائي وتحديد عدد العقد والمسافة بينها .

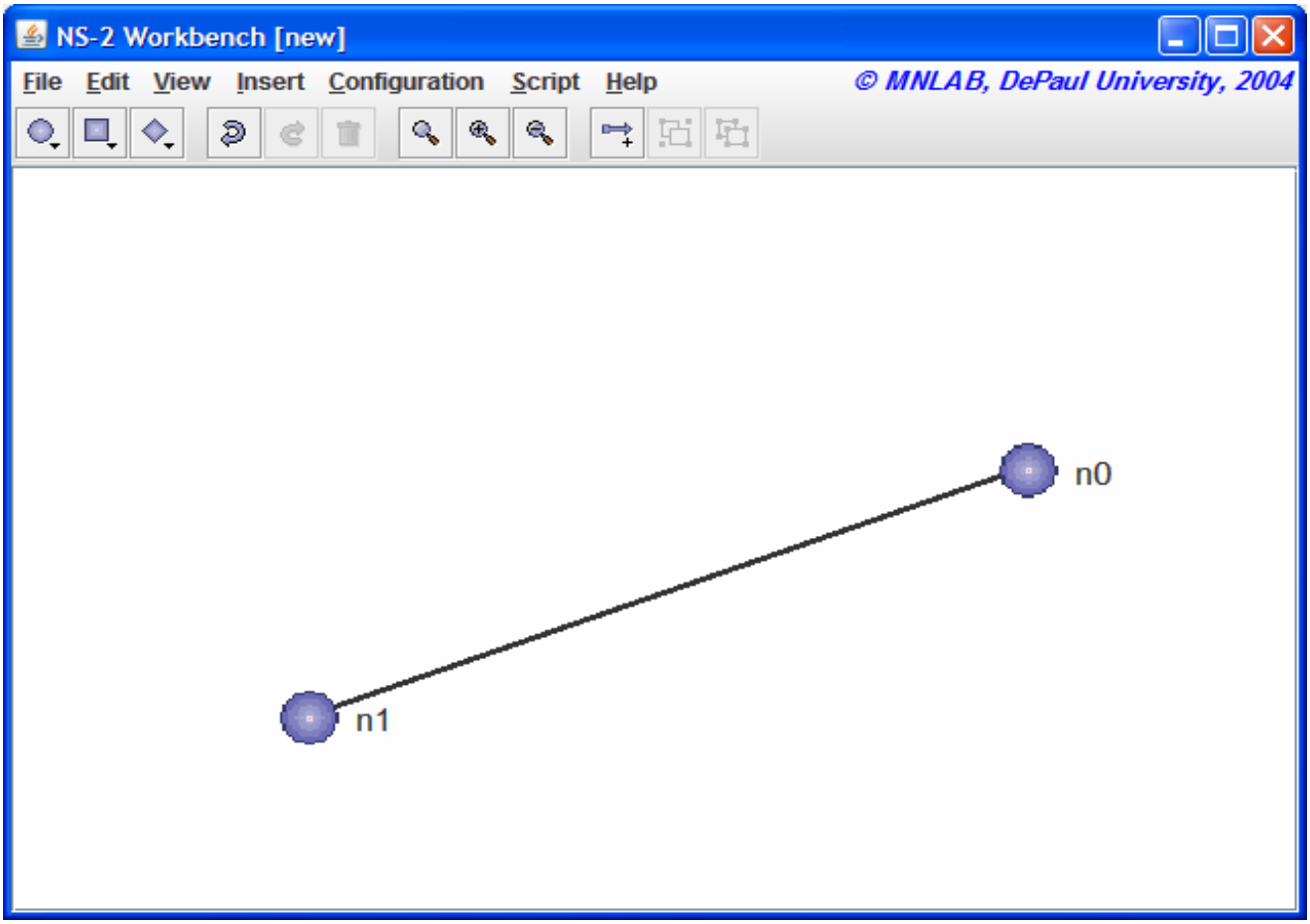
شكل البرنامج:



يتم تقسيم منطقة العمل إلى طبقات طبقة العرض وطبقة العقدة وطبقة الرابط وطبقة المضيف وطبقة التطبيق وطبقة المتغيرات وطبقة الكود والتي منها يتم انشاء الكود حيث يتم تغيير التهئية على حسب الطبقة ويتم الرسم عن طريق الضغط بزر الماوس في المنطقة البيضاء المجدولة.

المشروع الثالث هو مشروع قبل المشاريع السابقة بدأ من سنة 2004 وتم تحديثه وتعديله وآخر نسخة تم عملها هي نسخة beta تمت هذه السنة 2009 واسم المشروع **NSBench(NS-2 Workbench)** والغرض منه أيضاً تسهيل كتابة برامج NS2 لكن المميز جداً فيه مع أنه لا يصل إلى درجة المشروعين السابقين من ناحية الشكل والاستخدام لكن يتفوق عليهما بكثرة المكونات والنماذج التي يغطيها ويدعمها كما يدعم التنفيذ مباشرة عن طريقه مثله مثل مشروع NSBM كما يوفر ميزة جديدة وهي امكانية تجميع العقد مع بعضها البعض والتعامل معها كمجموعة واحدة.

وشكل البرنامج:



يتم الرسم بنفس طريقة البرنامجين السابقين فيمكن تعلم ذلك بسهولة فلا يوجد فرق كبير بينهم.

يمكنكم تحميل هذه البرامج والعمل عليها سواء في بيئة نظام التشغيل ويندوز أو لينيكس فقط يجب تنزيل الجافا الاصدارة السادسة jdk1.6 .

هذه المشاريع ساعدت كثيراً في تسهيل كتابة وانشاء البرامج باستخدام محاكي الشبكات NS2 فيمكن للطالب العمل في البداية بهذه البرامج إلى أن يحترف وبعد ذلك يتحول إلى مستوى الكود أي كتابة البرامج باستخدام الكود ، تجربتي الشخصية بدأت بتدريس معمل الشبكات لمدة سنتين ففي السنة الأولى قمت بتدريس الطلبة عن طريق الكود مباشرة حيث كنت اشرح لهم تفاصيل الكود فوجدت صعوبة بالغة

في توصيل الفهم المطلوب لهم وكان الغرض الأساسي من المعمل هو تدريس المفاهيم التي يتم أخذها في الجزء النظري في المعمل لكن كانت النتيجة عكس ذلك حيث كان تركيز الطلبة على فهم الكود وطريقة كتابة البرنامج ولكن في السنة الثانية قمت بالتدريس باستخدام برنامج NSG فوجدت فارقاً كبيراً جداً حيث اصبح تركيز الطلبة على ما يدرس لهم وليس على طريقة استخدام البرنامج فالبرنامج سهل جداً ولا يوجد صعوبة في تعلمه ، حتى انني قمت بتصحيح اختبار النظري للمادة فوجدت اجابات لأسئلة يعتبر الجزء العملي جزء من الاجابة حيث كانت الاجابة الخاصة بالجزء العملي صحيحة كلها بينما الأخطاء في الجانب النظري للمادة باعتبار أن الطالب يقوم بقراءة الجزء النظري ما قبل الامتحان بيوم أو يومين ،فكانت تجربة رائعة فأنصح ممن يدرسون في الجامعات أن يستخدموا هذه البرامج فهي مفيدة جداً وجميلة جداً وتساعد الطلبة على الاستيعاب أكثر وفهم المادة المقررة لهم بيسر وسهولة .

آخر بحثي كان في برنامج OPNET التجاري الذي يوجد نسخة تعليمية منه للطلبة وهي مجانية ولكن تفتقد للعديد من المميزات الموجودة في النسخة الكاملة والتي سعرها مرتفع جداً ، المهم وجدت دراسة عملت سنة 2005 كانت بعنوان التدريس باستخدام برنامج OPNET IT Guru بواسطة Dr. Vasil Y. Hnatyshin الشاهد من الموضوع أن هذا الدكتور وضح مميزات وعيوب البرنامج وأنه يزيد العبئ على المدرس وصعب الفهم بالنسبة للطلبة فلذلك تم التحول إلى NS2 وتم استخدامه في التعليم ووجد مميزات كثيرة أهمها امكانية التعديل حتى لو كانت المراجع غير متوفرة فيوجد الكثير من التمارين والدراسات التي الآن موجودة على الانترنت فيمكن الاستعانة به لكن فقط يفتقد إلى سهولة تعلمه للمبتدئين وكما اسردت فيوجد أكثر من مشروع طبق من ناحية التعليم ، كذلك نقطة أخيرة يكفي فقط أن تكون أغلب مشاريع منظمة IEEE تطبق وتنشر باستخدام NS2 فأرجو أن ينمو التعليم في الدول العربية وتستخدم الوسائل الحديثة في التعليم لكي تنشط البحوث والدراسات في الوطن العربي.

طرق تنصيب المحاكي في أنظمة التشغيل:

يمكن العمل على محاكي الشبكات على نظامي التشغيل لينيكس وويندوز بالطرق التالية:

- العمل على نظام لينيكس مثل Centos, RedHat, Ubuntu, Fedora, …etc وذلك عن طريق تحميل البرنامج من الموقع الرسمي للبرنامج http://www.isi.edu/nsnam/ns/, وبعد ذلك فك الضغط سواء على طريق الوحدة الطرفية Terminal عن طريق الذهاب أولاً لمسار البرنامج الذي حمل فيه إذا كان في الجذر الرئيسي root أو إذا كان في مسار آخر مثل سطح المكتب فيكن الذهاب إليه عن طريق الأمر:

cd /Desktop

وفك الضغط عن طريق الأمر

tar -zxvf ns-allinone-2.31.gz.tar

or

tar -xvf ns-allinone-2.31.tar.tar

بعد ذلك امر التنصيب

./install

بعد ذلك يتم تنصيب البرنامج كاملاً وبعد الانتهاء يقوم بعرض رسالة توضح أنه يجب عليك أن تقوم بالامر validate./. فهو غير مهم وإلى هذه الخطوة يكون المحاكي قد نصب تنصيباً كاملاً في جهازك.

المشاكل التي قد تحدث في أنظمة التشغيل لينيكس سببها هو نقص في الحزم التي يعتمد عليها المحاكي في عملية التنصيب ولحلها يجب تنصيب أدوات التطوير Development Tools الخاصة بالنظام مع تفعيل كل الخيارات وكذلك الحزم الخاصة بالواجهة فإذا كان النظام يدعم GNOME فيتم تنصيب حزم الواجهة الخاصة به زائد أدوات التطوير الخاصة به أيضاً ، بعد عمل ذلك لن يكون هناك أي مشاكل وإذا كانت هناك مشاكل مثل نظام فيدورا فالمشكلة عدم وجود حزمة tk والتي تدعم الواجهة الرسومية ضمن الحزم الخاصة بالواجهة الرسومية بالنظام فيمكن تحميل الحزم التي تدعم tk لنظام فيدورا وبعدها لن يكون هناك مشاكل باذن الله.

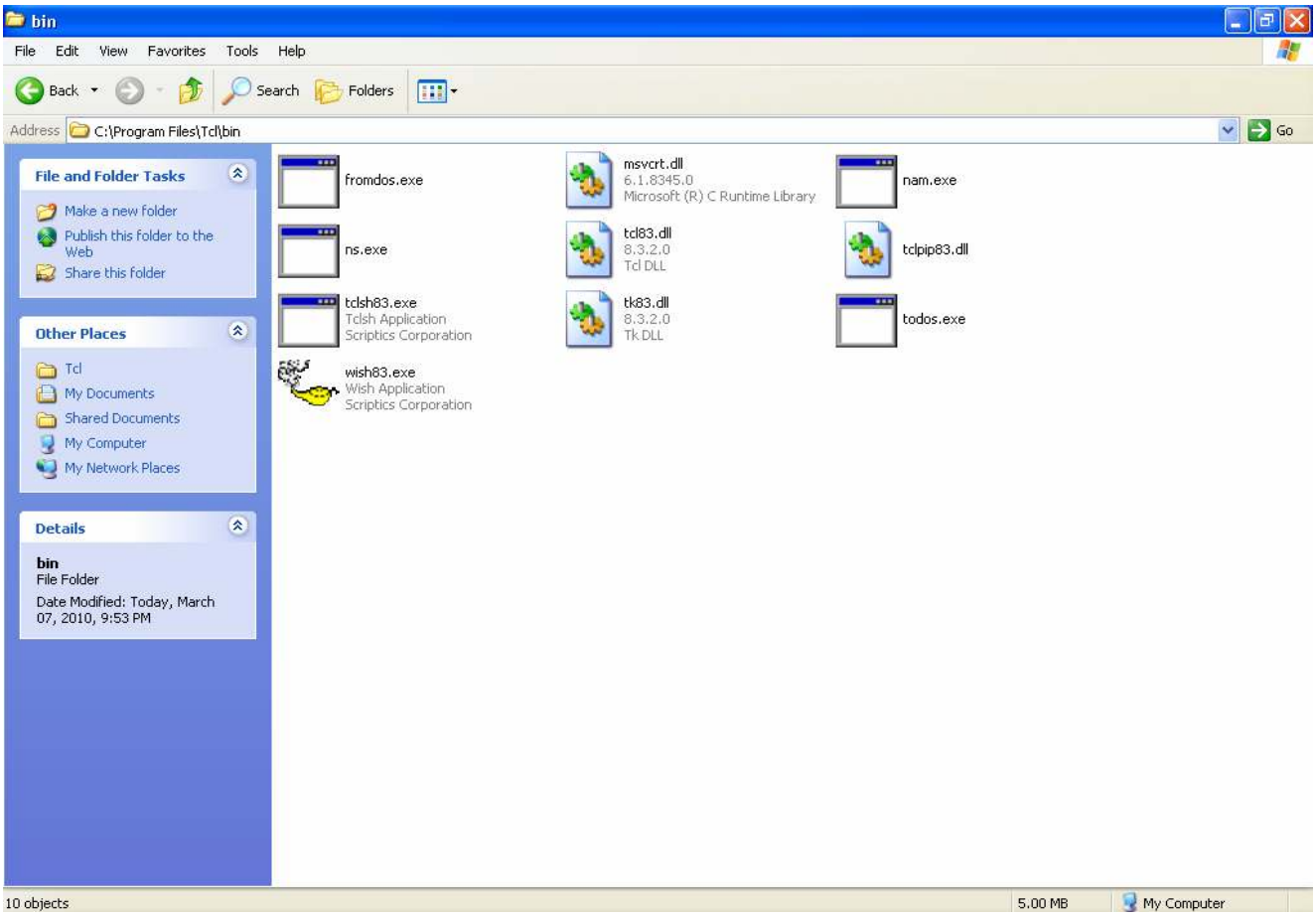
- الطريقة الثانية تنصيب المحاكي في نظام ويندوز والذي يمكن تشغيل المحاكي به بأكثر من طريقة:

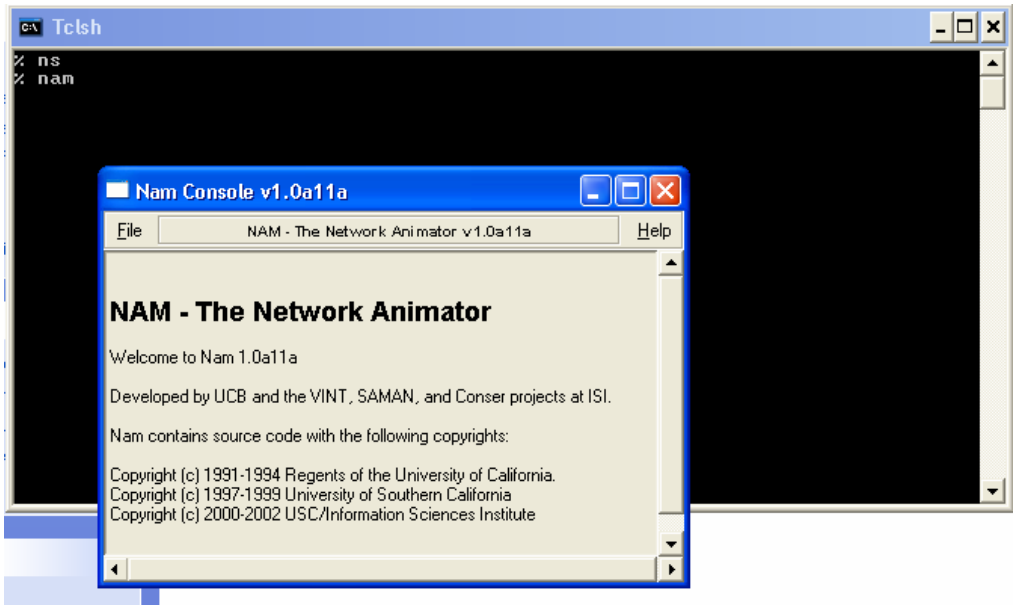
  - استخدام برنامج CYGWIN الذي يمثل emulator بالنسبة لنظام لينيكس في شكل برنامج Dos الخاص بنظام ويندوز ، ويوجد العديد من المراجع التي تشرح كيفية تنصيبه وهي موجودة في الموقع الرسمي للمحاكي زائد موقع cygwin الذي يمكن تحميل البرنامج منه.
  - استخدام برامج virtual machine تقوم بتشغيل اسطوانات من النوع iso ويمكن عن طريقه تنصيب نظام التشغيل لينيكس في نظام ويندوز وتشغيله عن طريق البرنامج فمن أشهر هذه البرامج Vmware and VirtualBox واللذان يوفران العديد من المميزات ، ما عليك سوى تحميل اصدارة لينيكس بامتداد iso ومن ثم تشغيلها عن طريق هذه البرامج.
  - استخدام Live CD وهي تقنية معروفة و تعمل عن طريق الاسطوانة CD وبعد اخراج الاسطوانة كأن شيئاً لم يكن ويتم الاستعانة بالهارديسك لتخزين ما يتم عمله من مستندات وأكواد وغيره، يوجد NS2 Live CD قم بالبحث بهذا الاسم وستجد اصدارة لينيكس عبارة عن Live CD بها برنامج NS2.

  وظهرت أيضاً في الآونة الأخيرة تقنية Live USB وهي عبارة عن أنظمة تشغيل تعمل من خلال فلاش USB ولا أقول أن هذه التقنية موجودة فقط مع نظام التشغيل لينيكس بل هي موجودة أيضاً في نظام التشغيل ويندوز فيوجد windows live cd لكنها تذكر مع اللينيكس لأنها لا توفر سوى أشياء بسيطة جدا لا تقارن مع التي توفرها أنظمة اللينيكس فمثلاً يوجد أصدارة اسمها puppy linux وهي عبارة عن Live USB وشكلها نفس شكل نظام Vista من ناحية Graphics ولا يوجد في نظام ويندوز Live USB حتى الآن.

- الطريقة الأخيرة وهي عن طريق تشغيل المحاكي كملف تنفيذي يتم تحميل الملفات التالية:
  - ns.exe
  - nam.exe
  - tcl830.exe
  - tfd170a.zip
  - tcl83.zip
  - حيث يحتوي ملف tfd170a.zip على ملفين قم بفك ضغطهما وقم بوضعهما في في bin في مسار TCL بعد تنصيبه C:\Program Files\Tcl\bin وقم بلصق هذين الملفين داخل هذا المجلد زائد ملفي ns.exe and nam.exe فيصبح bin يحتوي على الملفات ns.exe, nam.exe, todos.exe and fromdos.exe ثم قم بنسخ ملف المكتبة tcl83.dll إلى C:\WINDOWS\system32 وبعد ذلك قم بتشغيل أي مثال عن طريق ns.exe عن طريق فتح بواسطة لملف example.tcl.

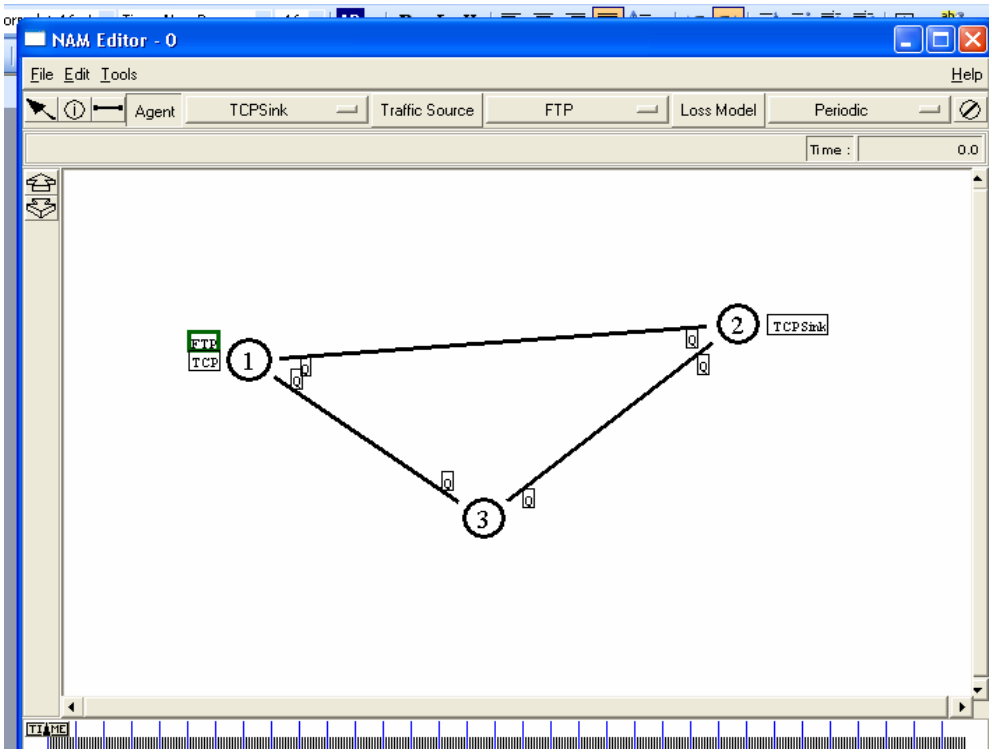
هذا هو شكل المجلد بعد نسخ الملفات:





وهذا هو شكل البرنامج بعد تنفيذه .

وهنا فقط يمكنك تنفيذ أمثلة لكن اذا كنت تريد التعديل على البرنامج فلا يمكنك عمل ذلك مع هذه الطريقة.



أخيراً :

في نهاية هذا الجزء تطرقنا للعديد من المواضيع والي كان أغلبها نظري وهي تعتبر مهمة جداً فهي تساعد على فهم المحاكي أكثر وأكثر وكان الهدف من هذا الجزء الاجابة على معظم الأسئلة التي قد تدور في ذهن أي مبتدئ في تعلم هذا المحاكي.

أريد أن ألفت نظر المهتمين بأمر هذا المحاكي أن يفكروا أكثر من ناحية التعليم وأن يتم التوجه إلى كتابة بعض التجارب التي تهدف إلى تعلم المحاكي وممارسته وتكون بمثابة منهج يتبع في الجامعات في التدريس لكي يتم التوصل إلى منهج يمثل مجموعة من التجارب المعملية التي تمثل كورس الشبكات وتشرح مبادئ ومفاهيم الشبكات بطريقة سهلة وعملية تساعد على فهم أفضل للطلبة.

قريباً.....

الجزء الثالث إن شاء الله سيكون حول المفاهيم التي تساعد على عملية extending of ns والذي سيحتوي على العديد من الأمثلة والمشاريع المحلولة والتطرق لأغلب المشاكل الناتجة من عملية التعديل والذي سيشمل اضافة مكون جديد بالنسبة للمجالات التالية:

Wireless, routing, mobile, agents, application, Queue Management Algorithms and protocols.

كما يستعرض مجموعة من الدراسات case studies في أكثر من مجال مع كيفية التطبيق واستخراج الرسوم البيانية graphs وكيفية الحصول على المعلومات اللازمة لعمل الرسم البياني.

والذي يعتبر أهم جزء بالنسبة لأكثر المهتمين في هذا المجال والذي يمثل 95% من الأسباب التي دعت الطلبة للتعرف على المحاكي الذي هو بمثابة منفذ لدراسة سواء كانت دراسة بكالريوس أو ماجستير أو دكتوراه، انشاء الله سيكون الجزء بمثابة دليل عملي لعملية الاضافة والتوسيع.

وهو سيكون أصعب جزء ويحتاج لدعم لكي يتم نشره في دور النشر وطباعته باللغة العربية والانجليزية وسيتم تسمية دليلك في تعديل وإضافة وتوسيع محاكي الشبكات NS2 وسيكون جزء من السلسلة ولمن اراد الدعم المراسلة عبر الايميل amjedns2@gmail.com

المراجع:

Slides:

- Teaching with OPNET Software :Dr. Vasil Y. Hnatyshin ,Department of Computer Science ,Rowan University,2005.
- Networks and Communication Course - NS-2 Network Simulator: University of Alexandria ,Faculty of Engineering-Computer and Systems Engineering Department .
- Network Simulator "ns" by Chadi Barakat , INRIA Sophia Antipolis ,France, PLANETE research group
- Ns Tutorial , 2006
- NS Simulator for beginners, Jeonghoon Park, Chapter 2. ns Simulator Preliminaries, 2007-04-14
- A introduction to Ns2: Luo Tao 27. Mrz 2009
- *ns-2* Introduction : mort (Richard Mortier) , Microsoft Research Cambridge.

Books:

- An Introduction to NS-2: Giovanni Perbellini, Verona, 12/09/2005
- NS Simulator for beginners.
- Introduction to Network Simulator NS2 , Teerawat Issariyakul • Ekram Hossain,2009
- The *ns* Manual, January 6,2009
- Ns by example, WPI.
- Tutorial for the Network Simulator "ns".

استغفر الله

استغفر الله

استغفر الله

اللهم صلي وسلم على رسول الله

صلى الله عليه وسلم