Análisis de Protocolos con Net Simulator 2 (ns-2)

Ing. José Luis Muñoz

¿Porqué usar simuladores de red?

- En el año 2000 se inicio el proyecto ARIES* (Advanced Research on Internet E-Servers) como parte de las actividades de investigación del Open Systems Lab en Ericsson Corporate Unit of Research.
- Objetivo del proyecto: Buscar tecnologías y diseñar prototipos que demuestren la factibilidad de usar Linux y el software para crear servidores de servicios Internet en términos de disponibilidad garantizada, tiempo de respuesta y escalabilidad.

(*) Network Simulator 2: a Simulation Tool for LinuxBy Ibrahim Haddad and David Gordon Linux Journal Octubre 2002

¿Porqué usar simuladores de red?

- El proyecto fue exitoso y continuó en en el 2001 enfocándose en mejorar las capacidades de clustering del Linux ser el sistema operativo de elección en servidores de Internet móvil.
  - Mejoras en balanceo de carga, distribución de tráfico y seguridad además de soporte IPv6.
- Inquietud surgida: ¿Cuál es el impacto de soportar IPv6 sobre otros protocolos usados por distintas aplicaciones en los clusters Linux?
  - p.e. efectos de IPv6 en SCTP, SCTP sobre HTTP, FTP, etc.

¿Porqué usar simuladores de red?

- El proyecto ARIES, no disponía del tiempo y recursos para instalar un laboratorio de pruebas con múltiples nodos y aplicaciones que usaran SCTP sobre IPv6.
- Solución: NS-2

¿Porqué usar simuladores de red?

- Los simuladores de red permiten evaluar y probar nuevos servicios y protocolos, lo cual tomaría mucho tiempo y dinero si se fuera directamente a una implementación
  - Por ejemplo redes con calidad de servicio y envío multicast requieren entornos grandes y complejos
- Con la simulación, se pueden hacer pruebas a gran escala y lo suficientemente exactas en sus resultados para validar o desechar algunas opciones de diseño en los protocolos.

NS-2

- El Network Simulator 2 (NS2), fue desarrollado por el Information Sciences Institute de la University of Southern California.
- Es el simulador más usado en el mundo académico y en los centros de investigación de las principales empresas de IT del mundo.

NS-2

- Simula un amplio rango de tecnologías de redes
  - Implementa protocolos tales como TCP y UPD
  - Genera comportamientos de tráfico FTP, Telnet, Web, CBR y VBR
  - Es capaz de simular mecanismos de gestión de colas en routers Drop Tail, RED y CBQ
  - Soporta diversos algoritmos de enrutamiento (p.e. Dijkstra)
  - NS implementa multicasting y algunas de las capas de enlace MAC para simular LANs.

NS-2

- Es un simulador de eventos Discreto
- Opera a nivel de paquete
  - Desde el nivel de enlace en adelante
- Permite simular redes cableadas e inalámbricas
- Ultimo release ns-2.31, hace 10 días

Plataformas

- Casi todas las variantes de Unix
  - √ FreeBSD o *BSD
  - √ Linux
  - √ Sun Solaris
- Windows
  - – Con cygwin
  - – O port de Atul Adya http://research.microsoft.com/~adya/

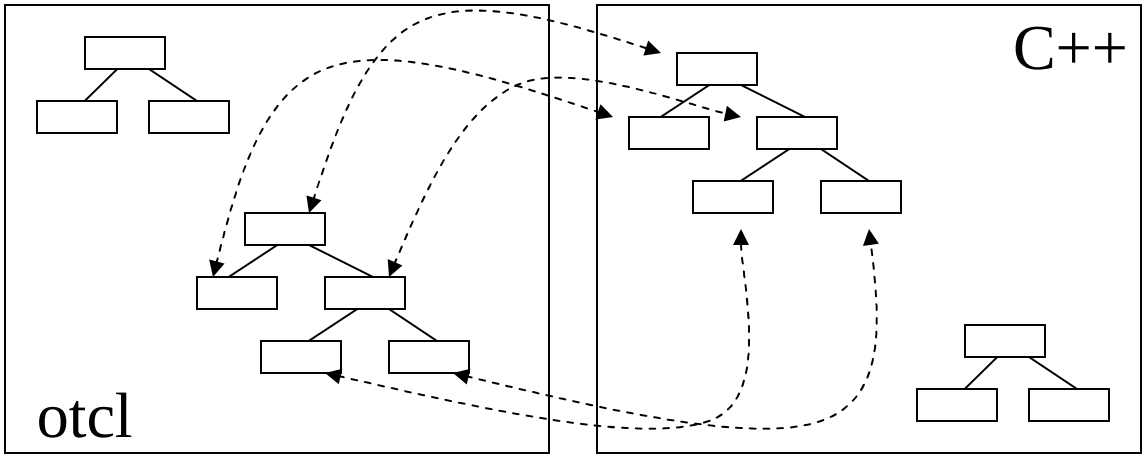
Arquitectura

- Orientada a objetos
  - Reusable
  - Fácil mantenimiento
  - Forzosa planificación previa
  - Performance

Arquitectura

- Requiere el conocimiento de dos lenguajes: C++ y oTCL.
  - C++ a nivel de datos
    - Actúa paquete por paquete
  - oTCL a nivel de control
    - En forma periódica o activada por eventos
- Es un compromiso entre facilidad de composición y velocidad
- Curva de aprendizaje no tan rápida
  - La depuración es otra historia...

Arquitectura

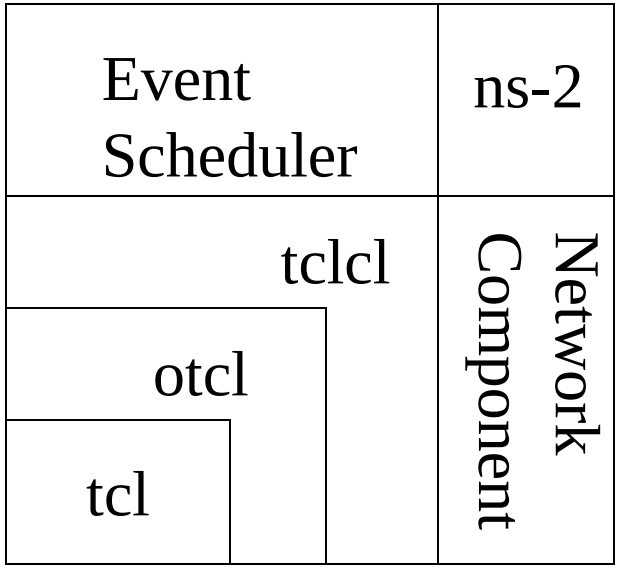


Dualidad oTCL y C++*

(*) http://nile.wpi.edu/NS/overview.html

Arquitectura

- oTCL es básicamente ***tcl* con extensiones de objeto***
  - Permite definir clases y crear objetos de esas clases.



- otcl: soporte orientado a objetos
- tclcl: enlace entre C++ y otcl
- Scheduler de eventos discretos
- Componentes de red de datos

(*) Tcl: tool command language

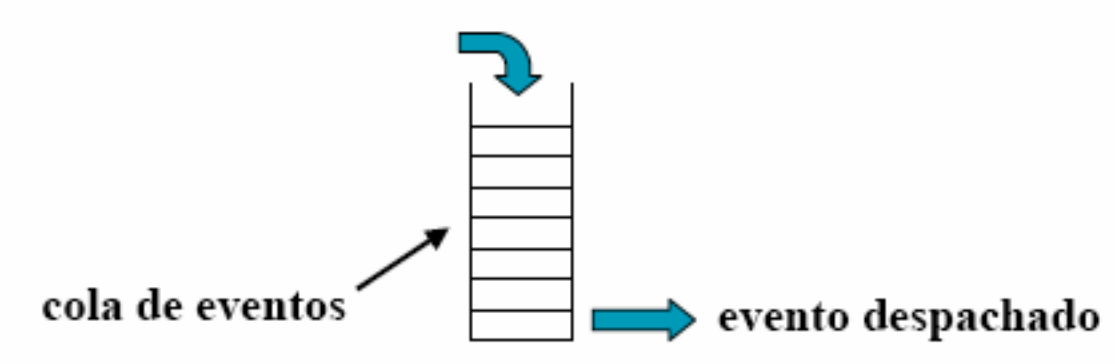
Arquitectura

Perspectiva del usuario

Script oTcl
Programa de simulación

Resultados de la simulación
- Análisis
- NAM

Despachador de eventos



- El despachador de tareas mantiene una cola de eventos ordenados por el tiempo en que deben ser ejecutados.
- Al llegar un nuevo evento el despachador se encarga de atenderlo, invocando los distintos componentes de red involucrados

Script en modo interactivo

```
linux21% ns
% set ns [new Simulator]

% $ns at 1 “puts \“Hello World!\””
1
% $ns at 1.5 “exit”
2
% $ns run
Hello World!
linux21%
```

Script en modo batch

```
#simple.tcl

    set ns [new Simulator]

    $ns at 1 “puts \“Hello World!\””

    $ns at 1.5 “exit”

    $ns run

linux21% ns simple.tcl

Hello World!

linux21%
```

Tcl básico

```
set a 43
set b 27
proc test { a b } {
   set c [expr $a + $b]
   set d [expr [expr $a - $b] * $c]
   for {set k 0} {$k < 10} {incr k} {
     if {$k < 5} {
       puts “k < 5, pow = [expr pow($d, $k)]”
     } else {
       puts “k >= 5, mod = [expr $d % $k]”
     }
   }
}
test 43 27
```

Otcl básico

```
Class Mama
Mama instproc saludar {} {
   $self instvar edad_
   puts “$edad_ años de edad
   mom: ¿Como estas?”
}

Class Hijo -superclass Mama
Hijo instproc saludar {} {
   $self instvar age_
   puts “$edad_ años de edad
   kid: ¿Que tal?”
}
```

```
set mama [new
  Mama]
$mama set edad_ 45
set hijo [new
  Hijo]
$hijo set age_ 15


$mama saludar
$hijo saludar
```

Componentes de red

- Nodos
- Enlaces
- Agentes (implementan protocolos de distintas capas)
- Aplicaciones
- Paquetes
- etc. …

Definición de una simulación ns

- Creación del despachador
- Creación de registros de eventos
- Creación de la topología de la red
- Programar secuencia de eventos (scheduler)
- Correr el simulador

Creación del despachador de tareas

- Primero se crea una instancia de la clase **Simulator**
  - **set pruebaNS [new Simulator]**

- Esto crea el Despachador de Tareas. Los métodos de la clase Simulator permiten después armar la topología y configurar la simulación.

Programar eventos y ejecución de la simulación

- La sintaxis de programación de eventos es:
  - $pruebaNS at <tiempo> <evento>

  Donde <evento> es cualquier comando ns/tcl válido

  Para iniciar el trabajo del despachador se corre el simulador
  - $pruebaNS run

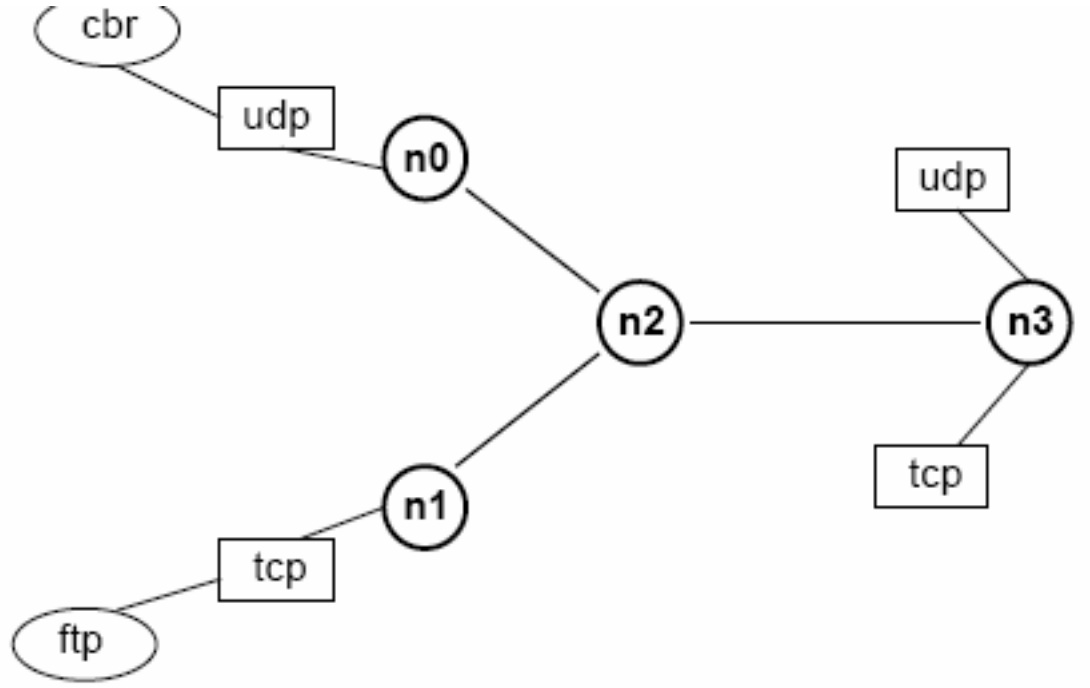
Ejemplo 1: Hola Mundo

- holaMundo.tcl
  - **set ns [new Simulator]**
  - **$ns at 1 “puts \ “Hola Mundo!!!\””**
  - **$ns at 1.5 “exit”**
  - **$ns run**

  >> ns holaMundo.tcl

  Hola Mundo!!!

Creación de la red



Creación de la red

- Nodos

  set n0 [$ns node]

  set n1 [$ns node]

- Enlaces y políticas de manejo de colas

  $ns duplex-link $n0 $n1 <ancho de banda> <retardo> <tipo de cola>

  - Tipo de cola: DropTail, RED, CBQ, etc.

Creación de una conexión TCP

- TCP
  - set tcp [new Agent/TCP]
  - set tcpsink [new Agent/TCPSink]
  - $ns attach-agent $n0 $tcp
  - $ns attach-agent $n1 $tcpsink
  - $ns connect $tcp $tcpsink

Generación de tráfico sobre TCP

- FTP
  - set ftp [new Application/FTP]
  - $ftp attach-agent $tcp
- Telnet
  - set telnet [new Application/Telenet]
  - $telnet attach-agent $tcp

Creación de un agente UDP

- UDP
  - set udp [new Agent/UDP]
  - set null [new Agent/Null]
  - $ns attach-agent $n0 $udp
  - $ns connect $udp $null

Crear tráfico sobre UDP

- CBR
  - set src [new Application/Traffic/CBR]
- Exponencial o Pareto on-off
  - set src [new Application/Traffic/Exponential]
  - set src [new Application/Traffic/Pareto]

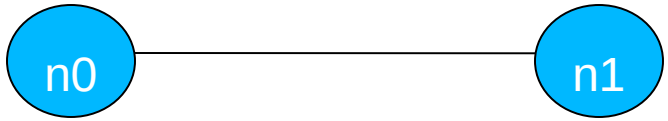
Registro de eventos

- Es posible registrar todos los eventos que suceden en la simulación, generando un archivo de texto con toda la información:
  - $ns trace-all [open salida.out w]
  - $ns namtrace-all [open salida.namw]
- Este comando registra todos los paquetes que pasan por todos los enlaces, generando la siguiente salida:

```
+ 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
- 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
r 1.00234 0 2 cbr 210 ------- 0 0.0 3.1 0 0
```

Registro de eventos

- Tambien se pueden registrar eventos en enlaces particulares;



  - `$ns trace-queue $n0 $n1`
  - `$ns namtrace-queue $n0 $n1`

Ejemplo 2 - simulación

- Instanciamos la clase Simulator (crear el Despachador)

  set ns [new Simulator]

- Creamos un archivo de salida para el namregistramos en el todos los eventos

  set nf [open out.namw]

  $ns namtrace-all $nf

Ejemplo 2 - simulación

- Definimos un procedimiento fin que ejecutaremos al final de la simulación para cerrar el archivo de salida y ejecutarlo con el nam.

```
proc fin { } {
  global ns nf
  $ns flush-trace
  close $nf
  exec namout.nam &
  exit 0
}
```

Ejemplo 2 - simulación

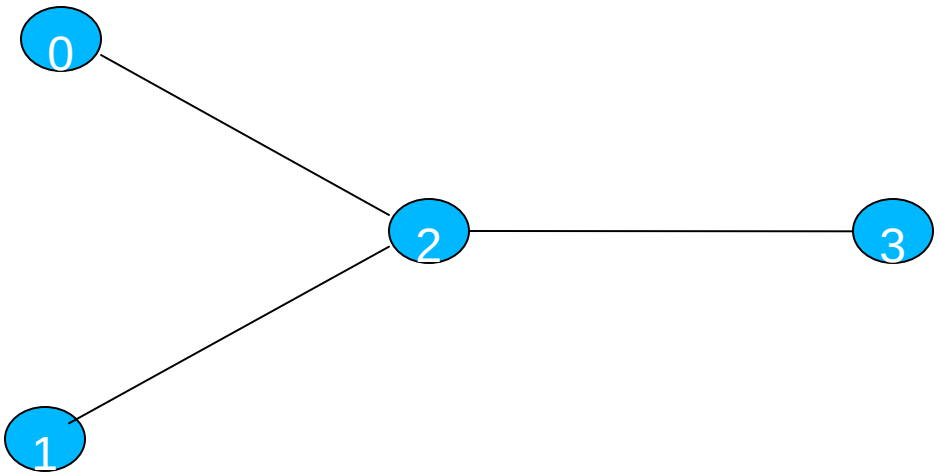
- Creamos la topología de red

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms DropTail
```

Ejemplo 2 - simulación

- Se puede ordenar los nodos para la visualización en NAM

```
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
```



Ejemplo 2 - simulación

Creamos un agente UDP en el nodo n0 y un receptor en n3

set udp [new Agent/UDP]

$ns attach-agent $n0 $udp

set null [new Agent/Null]

$ns attach-agent $n3 $udp

$ns connect $udp $null

Creamos una fuente CBR en el agente UDP creado

set cbr [new Application/Traffic/CBR]

$cbr set packetSize_ 500 # tamaño de paquete

$cbr set interval_ 0.005 # tiempo entre paquetes (seg)

$cbr attach-agent $udp

Ejemplo 2 - simulación

Creamos un agente TCP en el nodo n1 y un receptor en n3

set tcp [new Agent/TCP]

$ns attach-agent $n1 $tcp

set sink [new Agent /TCPSink]

$ns attach-agent $n3 $sink

$ns connect $tcp $sink

Creamos una aplicación FTP en el agente TCP creado

set ftp [new Application/FTP]

$ftp attach-agent $tcp

Ejemplo 2 - simulación

Programar la ocurrencia de eventos y ejecutamos la simulación

$ns at 0.5 “$cbr start”

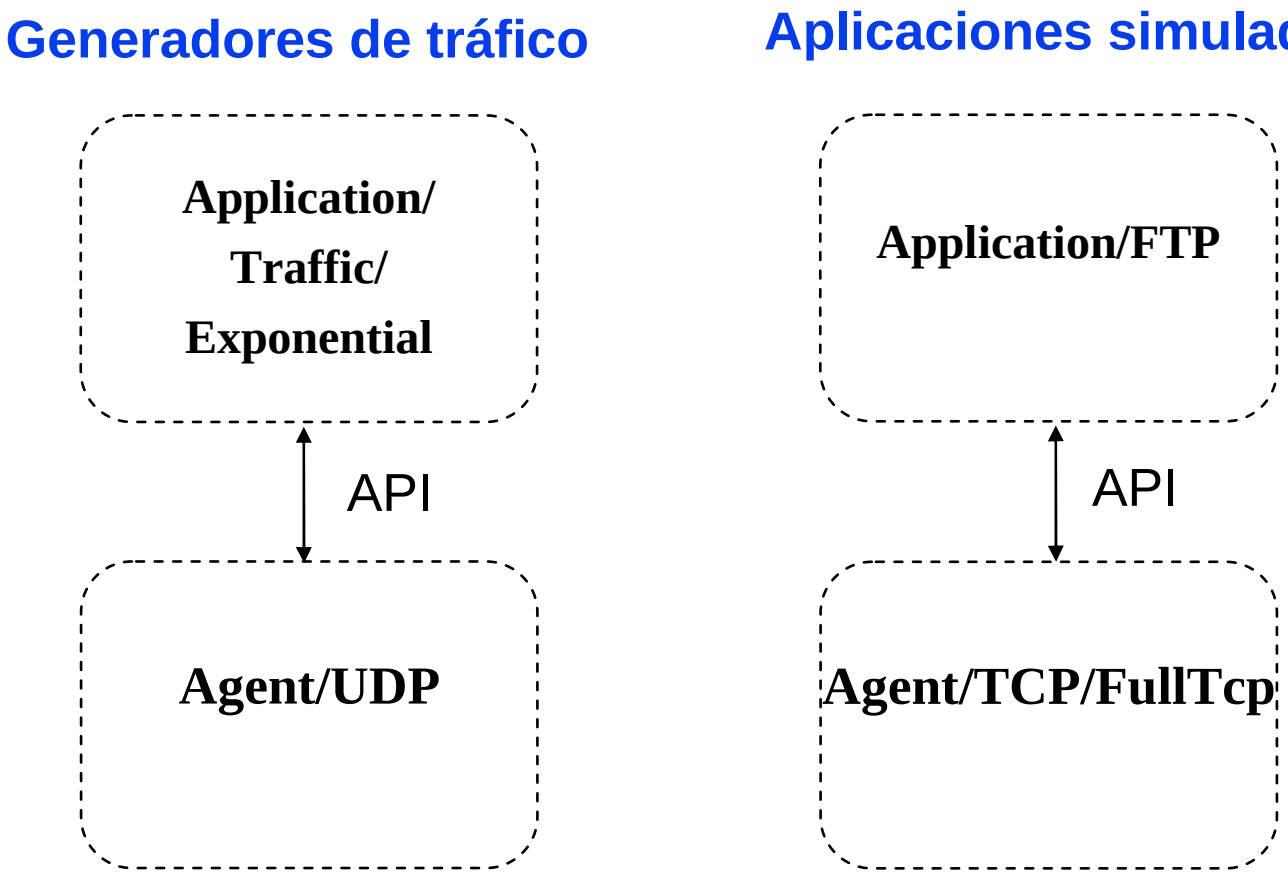
$ns at 1.0 “$ftp start”

$ns at 4.0 “$cbr stop”

$ns at 5.0 “fin”

$ns run

Composición de una Aplicación



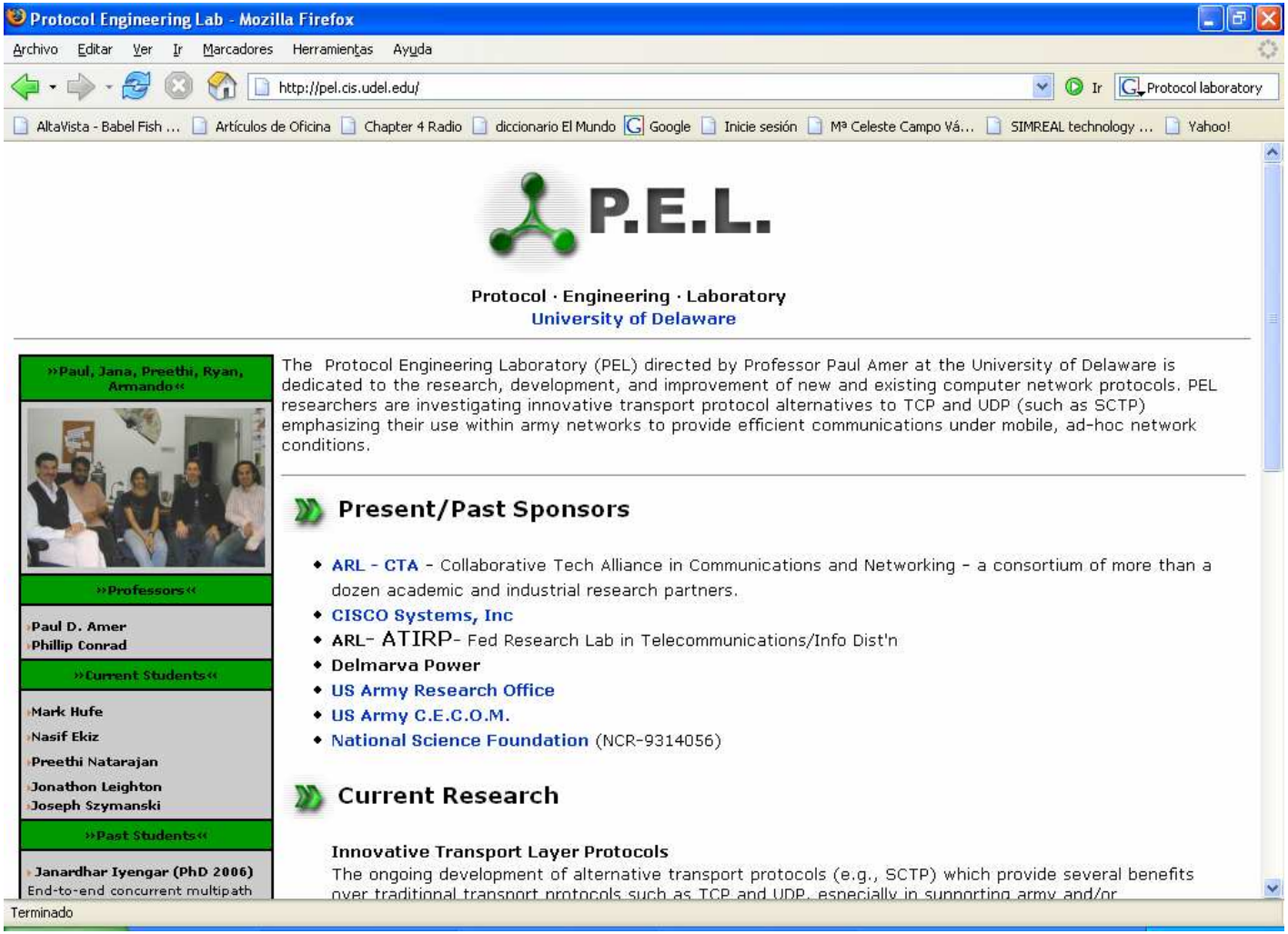
El diagrama ilustra como los agentes y aplicaciones se conectan y comunican entre sí via el API de ns

Conexión de agentes de transporte a los nodos

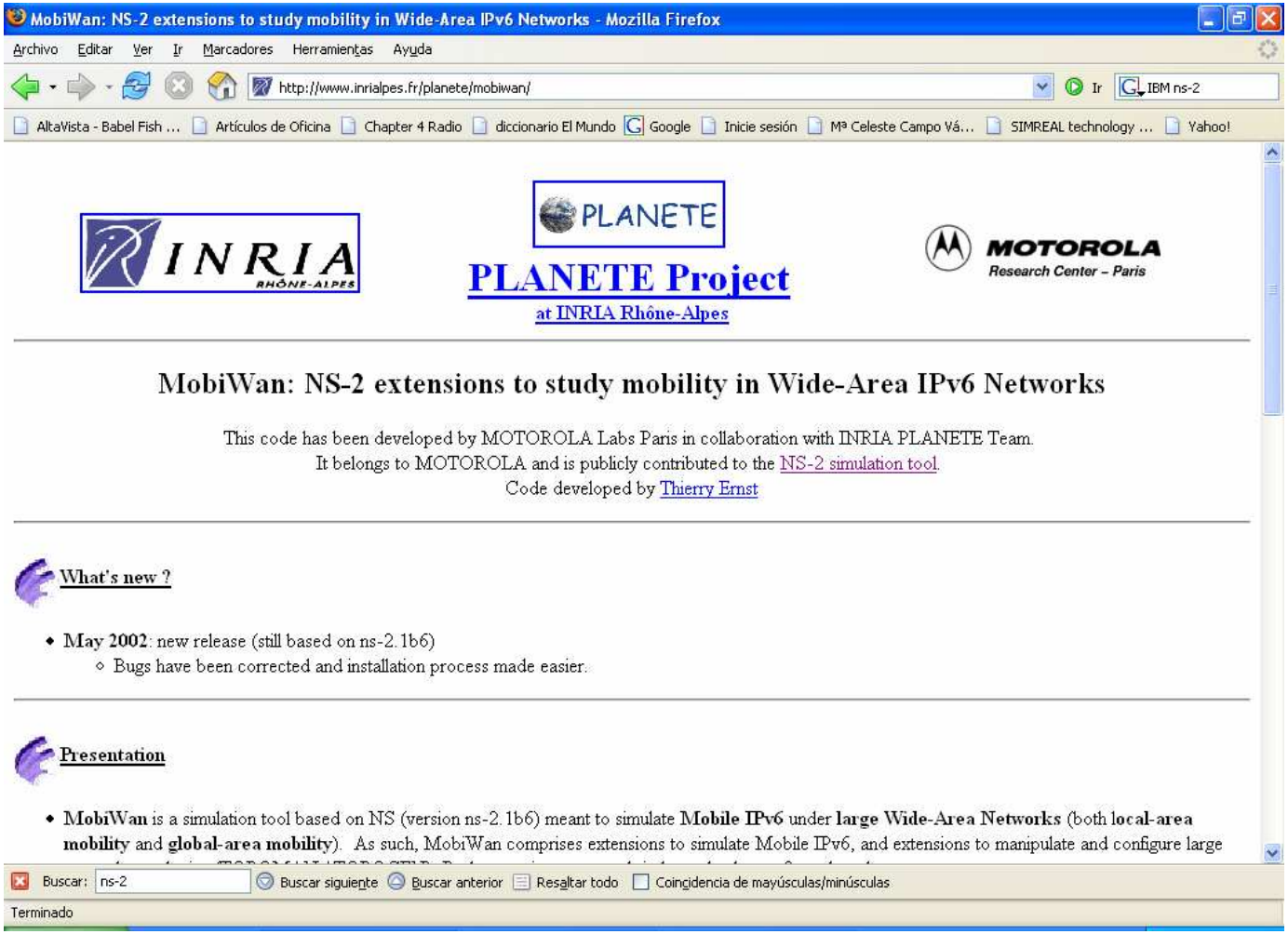
```
set src [new Agent/TCP/FullTcp]
set sink [new Agent/TCP/FullTcp]
$ns_ attach-agent $node_(s1) $src
$ns_ attach-agent $node_(k1) $sink
$ns_ connect $src $sink

set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$udp0 set packetSize_ 536 ; #(max=1000)
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
$ns connect $udp0 $null0
```

Ejemplos de aplicación



Ejemplos de aplicación



Ejemplos de aplicación

Information Sciences Institute - Research - Mozilla Firefox

Archivo Editar Ver Ir Marcadores Herramientas Ayuda

http://www.isi.edu/research/#Networking

AltaVista - Babel Fish ... Artículos de Oficina Chapter 4 Radio diccionario El Mundo Google Inicie sesión Mª Celeste Campo Vá... SIMREAL technology ... Yahoo!

Networking

One of the birthplaces of the Internet, ISI continues to research new ways to link computational resources--and safeguard --> --existing ones.

ISI researchers developed the Domain Name System, as well as many of the basic protocols underlying email, telnet, and ftp. ISI researchers also developed and licensed many of the core multicast technologies that support teleconferencing and distance education.

ISI has developed key elements of the Next Generation Internet, such as ultrahigh capacity local networks, and ISI scientists are among the leaders in the development of Internet Protocol version 6. Other key research interests include the X-Bone system for automating deployment and management of overlay networks, routing policy and management systems, mapping and fault detection for large networks, and active networking for dynamic network control and signaling. Applying architectural design principles learned in the development of the Internet, researchers are exploring ways to scale networks that are pervasive, unattended, and widely embedded throughout the physical environment. ISI also manages or participates in several national and regional operational networks such as Los Nettos, CAIRN, TRAIL, CalRen-2, and the Active Nets Backbone as collaborative research test beds. It coordinates routing, interconnects, and multiple test bed experiments for the Next Generation Internet program's SuperNet, a coast-to-coast network providing individual researchers with multigigabit-per-second bandwidth.

Applications for this bandwidth include the "Digital Amphitheater," which will allow new dimensions in meetings, education and even artistic performance. ISI also develops simulation, design, and testing tools such as the Virtual Internet Testbed and the Multicast Internet Testbed for projects that go beyond the scale and testing capabilities of today's networks. ISI and partners have demonstrated real-time transmission of full bandwidth high definition TV over the Internet, along with a new way of transmitting high definition TV using native Internet protocols, and also GRIP, a technique to enable PCs to communicate securely at gigabit/sec speeds.

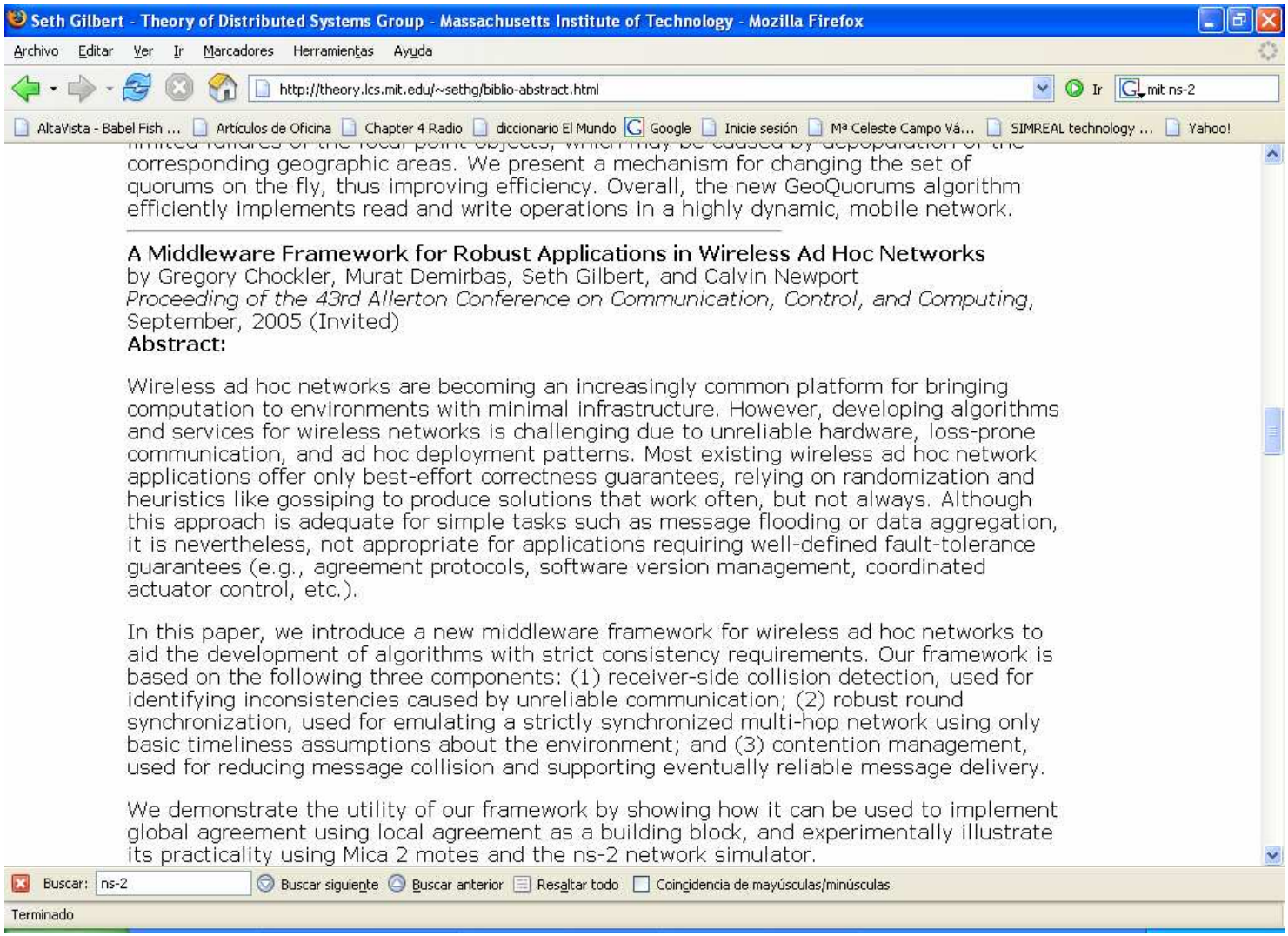
PCEN

The Postel Center for Experimental Networking (PCEN) was established at ISI in 1999 to commemorate Jon Postel's contributions to the development of the Internet. PCEN provides facilities, funding, and support for distinguished visiting scholars and graduate fellows to perform applied research and develop tools of general utility in the spirit of its namesake's work. Its endowment has been generously funded by Cisco Systems, Centergate, and Sun Microsystems, as well as by a number of private individuals.

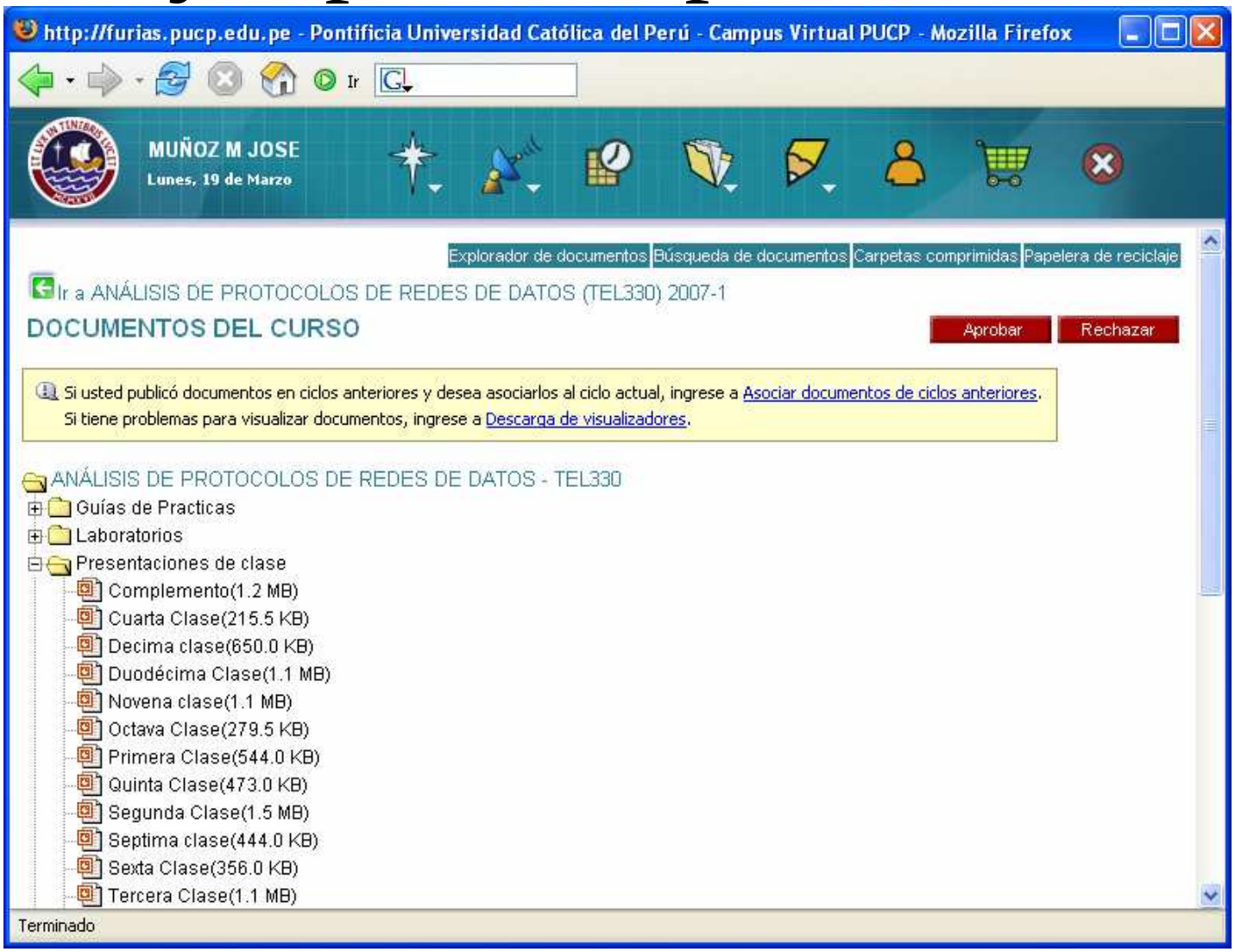
Work at PCEN is driven by the interests of its visiting scholars and graduate research

Terminado

Ejemplos de aplicación



Ejemplos de aplicación



Conclusiones

- Los simuladores permiten validar posibles soluciones de conectividad, sin llegar a implementaciones que son en muchos casos costosas o dificiles
- Ns-2 es el simulador de red más difundido en el mundo académico
- Es un software que requiere dos cosas: conocer el problema de conectividad a resolver y aprender oTCL/C++
- En la PUCP es usado para cursos de comunicaciones de datos

Referencias

- http://nile.wpi.edu/NS/
- http://www.ee.surrey.ac.uk/Personal/L.Wood/ns/
- http://crysol.inf-cr.uclm.es/node/224
- http://nslab.ee.ntu.edu.tw/courses/net-simtest-spring-07/