# Introducción al Simulador NS-2

Modelado y Análisis de Redes de Telecomunicaciones

Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República

# Agenda



# Agenda



# Generalidades

Simulador de eventos discretos orientado al estudio de redes.

Provee el soporte para simulación de:

1. protocolos de transporte (TCP y UDP)
2. aplicaciones y fuentes de tráfico (FTP, CBR, Telnet, etc)
3. políticas de manejo de colas (Drop Tail, RED, etc)
4. algoritmos de ruteo (Dijkstra)
5. redes cableadas o wireless
6. etc...

Además permite:

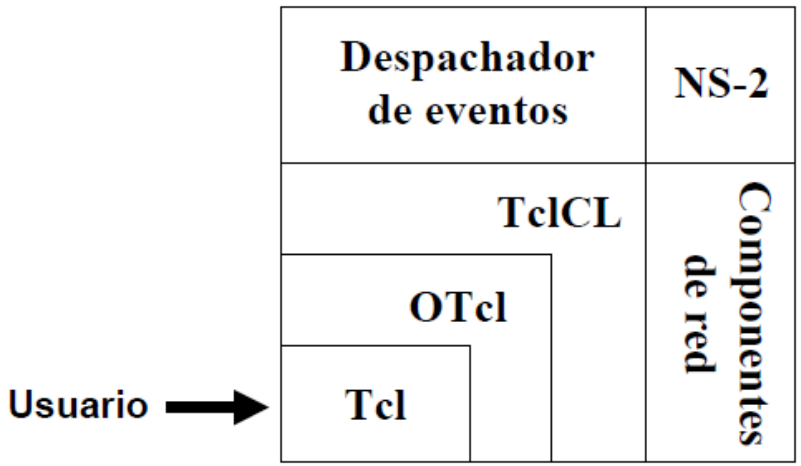
1. almacenar y visualizar trazas
2. crear y modificar módulos

# Estructura de Software

- Desarrollado en **dos** lenguajes: C++ y OTcl
- C++ para el "core" del NS: es el lenguaje compilado con el que se implementa la mayor parte de los modelos de los objetos de red utilizados durante las simulaciones
- OTcl para el control.
  - OTcl es una extensión del lenguaje de programación Tcl (Tool Command Language) orientada a objetos
  - Tcl es un lenguaje de scripting (conjunto de comandos) interpretado

El NS-2 básicamente es un "intérprete" de scripts OTcl.

# Arquitectura



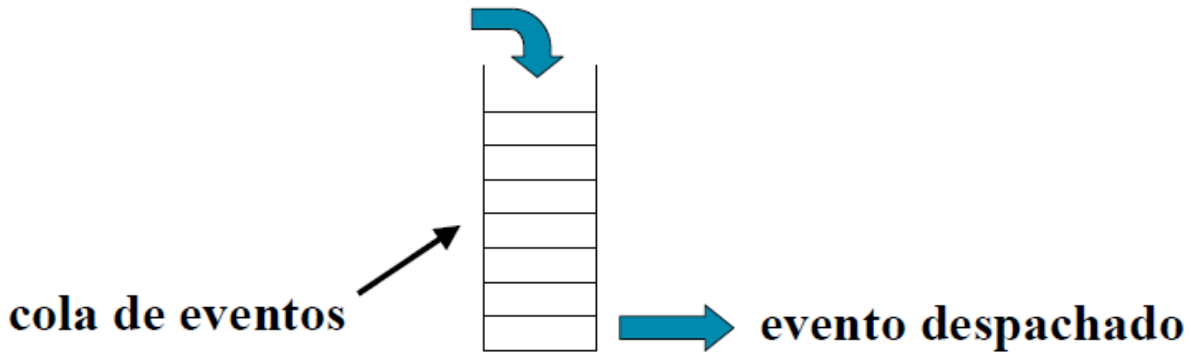

## Componentes de Red

nodos, enlaces, agentes, paquetes, etc.

## Despachador de Eventos

maneja los eventos en orden temporal

# 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.

# Agenda



# Pasos para escribir una simulación

- Crear despachador
- Crear registros de eventos
- Crear los Componentes de Red
- Agendar los eventos
- Correr el simulador

# Crear el despachador

**set ns [new Simulator]**

- Esto crea una instancia de la clase Simulator que será el encargado de manejar los eventos.
- Los métodos de dicha clase permiten armar la topología y configurar todo lo referente a la simulación

# Programar eventos y correr el simulador

La sintaxis de programación de eventos es la siguiente:

**$ns at <tiempo> <evento>**

- $ns es la variable "despachador" de la clase Simulator que se creó en el paso anterior
- <evento> puede ser cualquier comando ns/tcl válido

**$ns run**

La línea anterior es la que corre el simulador, es la última línea del script

# 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.nam w]** - para visualización

- También se pueden hacer trazas específicas y no de toda la simulación.
  Ver en detalle el **ejemplo3.tcl**

# Tcl: Manejo de Variables y operaciones

Uso *set* para crear y asignarle un valor a una variable

**set a 100**

El signo de $ hace que el intérprete sustituya la variable por su valor

**puts $a**

Para realizar operaciones aritmétricas se utiliza el comando *expr*

**expr 2*$a**

Cadena contenida dentro de corchetes

La cadena se evalúa y se sustituye por el resultado obtenido.

**set b [expr 2*$a]**

**puts $b**

salida: *200*

Cadena contenida entre comillas

**set b "expr 2*$a"**

**puts $b**

salida: *expr 2*$a*

obs: Para ejecutarla se requiere utilizar el comando *eval*

# Tcl: Estructuras de control

## for

```
for {set i 1} {$i <= 3} {incr i}{
puts $i
}
```

# Tcl: Estructuras de control

## for

```
for {set i 1} {$i <= 3} {incr i}{
puts $i
}
```

## while

```
set i 1
while {$i <= 3} {
puts $i
incr i
}
```

# Tcl: Estructuras de control

## for

```
for {set i 1} {$i <= 3} {incr i}{
puts $i
}
```

## while

```
set i 1
while {$i <= 3} {
puts $i
incr i
}
```

## if

```
if {$i < 0} {
puts "Negativo"
}else{
puts "Positivo o cero"
}
```

# Otros comandos Tcl útiles

## Crear variables aleatorias

```
set gen [new RNG]
for {set j 1} {$j < $run} {incr j} {
$gen next-substream}
set r [new RandomVariable/Uniform]
$r set min_ 0.0
$r set max_ 10.0
$r use-rgn $gen
```

## Procedimientos

Un procedimiento se define de la siguiente manera:

```
proc nombreProc {arg1 agr2 ...} { ...
...
}
```

# Agenda



# Ejemplo 1: Hola Mundo

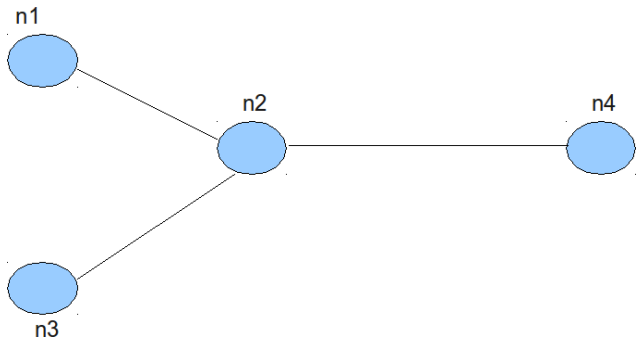
holaMundo.tcl

```
set ns [new Simulator]
$ns at 1 “puts \“Hola Mundo!!!\” ”
$ns at 1.5 exit
$ns run
```

Para correr el programa ejecutar:

ns *holaMundo.tcl*

# Ejemplo 2: Red Cableada




Características a simular:

- Tráfico CBR/UDP desde $n1$ a $n4$
- Tráfico FTP/TCP desde $n3$ a $n4$
- Enlace $n1$-$n2$ 1Mbps, 2ms, DropTail
- Enlace $n3$-$n2$ 2Mbps, 2ms, DropTail
- Enlace $n2$-$n4$ 2Mbps, 3ms, DropTail

Visualizar el script **ejemplo2.tcl**

# Ejemplo 2 cont

**Definimos el despachador:**

set ns [new Simulator]

**Creamos los archivos de salida:**

set nam [open *salida.nam* w]
$ns namtrace-all $nam
set tf [open *salida.out* w]
$ns trace-all $tf

# Ejemplo 2 cont

**Creo la topología:**

- Creo los nodos:
  set n1 [$ns node]
  set n2 [$ns node]
  set n3 [$ns node]
  set n4 [$ns node]
- Creo los enlaces:

$ns duplex-link $n1 $n2 <anchoBanda> <retardo> <tipoCola>

$ns duplex-link $n1 $n2 1Mb 2ms DropTail
$ns duplex-link $n3 $n2 2Mb 2ms DropTail
$ns duplex-link $n2 $n4 2Mb 3ms DropTail

# Ejemplo 2 cont

**Definición de fuentes de tráfico: FTP/TCP:**

```
set tcp [new Agent/TCP]
set tcpsink [new Agent/TCPSink]
$ns attach-agent $n3 $tcp
$ns attach-agent $n4 $tcpsink
$ns connect $tcp $tcpsink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
```

# Ejemplo 2 cont

**Definición de fuentes de tráfico: CBR/UDP:**

```
set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n1 $udp
$ns attach-agent $n4 $null
$ns connect $udp $null
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
```

**Seteo las características del tráfico CBR:**

```
$cbr set packetSize_ 500 # tamaño de paquete
$cbr set interval_ 0.005 # tiempo entre paquetes (seg)
```

# Ejemplo 2 cont

**Defino procedimiento:**

fin

```
proc fin {} {
global ns tf nam
puts "done!"
$ns flush-trace
close $tf
close $nam
exec nam salida.nam &
exit 0
}
```

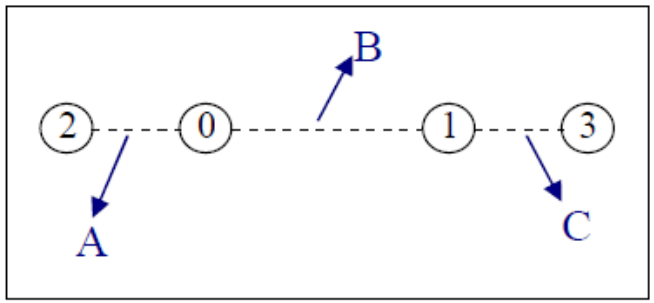
**Agendamos los eventos y corremos el simulador:**

```
$ns at 0.5 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$cbr stop"
$ns at 4.5 "$ftp stop"
$ns at 5.0 "fin"

$ns run
```

# Ejemplo 3

## Características simuladas




- Topología constituía por 4 nodos (Ver fig)
- nodo 2 es una fuente FTP
- nodo 3 es el receptor
- Se generan trazas particulares:
  - ventana de la fuente tcp en el nodo 2
  - datos asociados a la cola de los enlaces

# Agenda



# Agenda



# Antecedentes

El estándar IEEE 802.11 se creó para definir un tronco común de desarrollo de redes de área local inalámbricas (WLAN)

- 1997 Aparece IEEE 802.11, hoy legacy
- 1999 802.11a, Capa fsica de 54Mbps (5GHz)
- 1999 802.11b, Capa fsica de 11Mbps (2.4GHz)
- 2003 802.11g, Capa fsica de 54Mbps (2.4GHz)
- 2005 802.11e, QoS, service differentiation
- 2007 802.11-2007 norma que integra los ammendements ya aprobados
- 2009 802.11n, Capa fsica de 300Mpbs
- ...

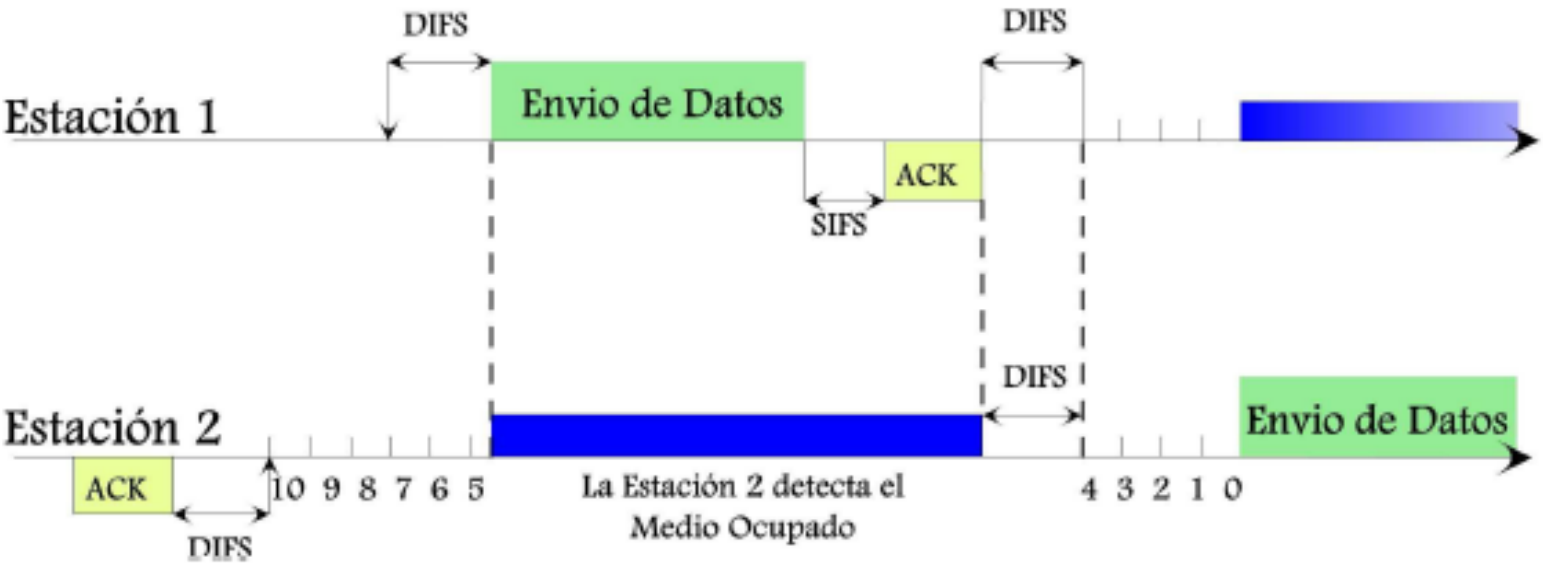
# La sub capa MAC

- Esta capa debe encargarse del acceso al medio compartido
- Se basa en CSMA/CA con ventana de backoff.
- Propone como opcional un algoritmo MACA (RTS/CTS) para lidiar con los problemas de estación oculta

El modo predominante es el denominado Distributed Coordination Function (DCF)

obs: A continuación se explica brevemente el funcionamiento de DCF.

# DCF - Modo Básico de acceso al medio



# Modelo de Bianchi para 802.11 DCF

G. Bianchi (ver ref) propone un modelo para analizar el desempeño de 802.11:
Hipótesis principales:

- Demanda de saturación (siempre hay paquetes).
- Canal ideal.
- Número fijo de estaciones ($N$).
- Independencia entre usuarios.
- Probabilidad de colisión constante e independiente del estado de backoff

# Descripción del modelo

Se separa el problema en dos partes:

1. Identificar cuál es la probabilidad de acceso al medio ($\tau$) y la probabilidad de colisión ($p$) en función de los parámetros del algoritmo DCF de backoff y $N$ (cant. usuarios en la celda).

# Descripción del modelo

Se separa el problema en dos partes:

1. Identificar cuál es la probabilidad de acceso al medio ($\tau$) y la probabilidad de colisión ($p$) en función de los parámetros del algoritmo DCF de backoff y $N$ (cant. usuarios en la celda).
2. Una vez obtenida la probabilidad de acceso $\tau$ , se calcula el throughput máximo a obtener a partir de:
   - $\tau$ probabilidad de acceso.
   - $N$ usuarios en la celda.
   - $Ptr$ probabilidad de transmisión.
   - $Ps$ probabilidad de transmisión exitosa.
   - $P$ tamaño medio de paquete.
   - Parámetros de tiempo de la capa física.

# Descripción del modelo cont.

## Parámetros

- $m$ no. máximo de backoff stages.
- $W_{min}$ ventana mínima de backoff
- $W_{max}$ ventana máxima de backoff, $W_{max} = 2^m * W_{min}$

## Variables

- $i$ estado de backoff
- $W$ ventana de backoff, $W = 2^i * W_{min}$

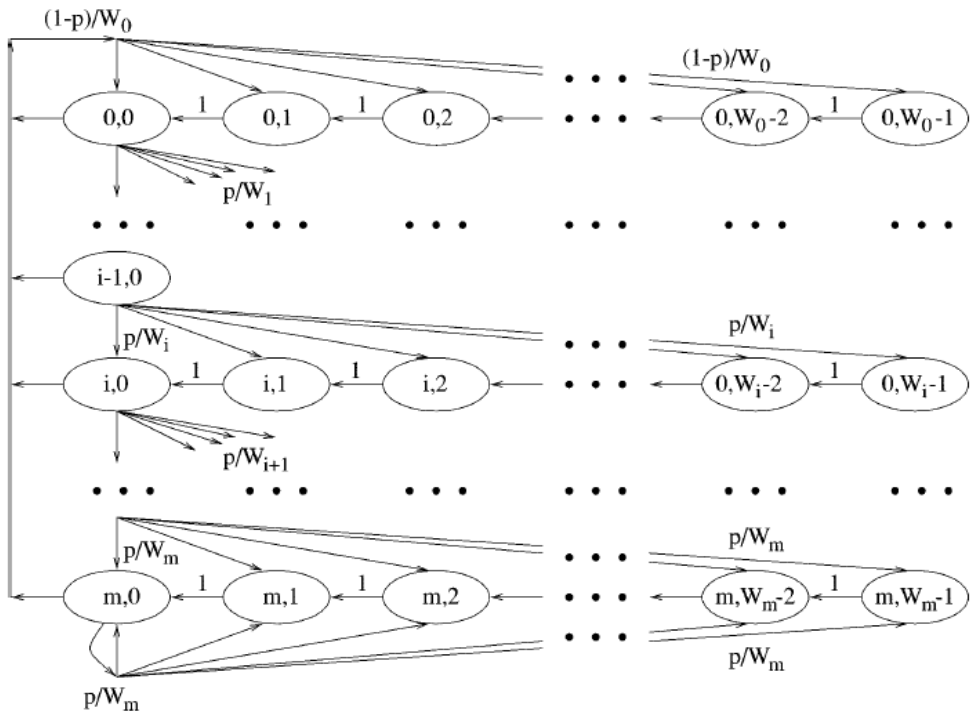
Cada estación corre de manera independiente el siguiente algoritmo:

1. sortea $k$ entre 0 y $W$
2. backoff: desde $k$, en cada time slot $k = k - 1$
3. cuando $k = 0$ intenta transmitir
   - Si no logra transmitir $i = i + 1$ y vuelve al paso 1
   - Si logra transmitir $i = 0$

# Modelado: Cadena de Markov

Se define un proceso markoviano bidimensional, donde:

- $b(t)$ proceso estocástico que representa el contador de backoff (k)
- $s(t)$ proceso estocástico que representa el estado de backoff (i)



# Resolviendo el modelo se obtiene:

Probabilidad estacionaria de transmisión

$$\tau = \frac{2(1-p)}{W_{min}(1-(2p)^m)(1-p) + (1-2p)(W_{min}(2p)^m - 1)} \quad (1)$$

Probabilidad de colisión

$$p = 1 - (1-\tau)^{n-1} \quad (2)$$

Probabilidad de que una estación transmita en cierto time slot

$$P_{tr} = 1 - (1-\tau)^n \quad (3)$$

Probabilidad de que una transmisión resulte exitosa

$$P_s = n\tau(1-\tau)^{n-1}/P_{tr} \quad (4)$$

# Throughput de la celda (S)

Cociente entre el tiempo de transmisión (exitosa) sobre el total de tiempo que se ocupa el canal

$$S = \frac{P_{tr} P_s E[P]}{(1 - P_{tr})\sigma + P_{tr} P_s T_s + P_{tr}(1 - P_s) T_c} \quad (5)$$

Donde:

- $T_s$ tiempo que permanece ocupado el canal debido a una transmisión exitosa
- $T_c$ tiempo que permanece ocupado el canal debido a una colisión
- $\sigma$ es la duración de un time slot.

# Agenda



# Biblioteca: dei80211mr

- Se incluyó a partir del ns-2.33 (se recomienda instalar Ubuntu 10.10 y ns-2.35)
- Implementa 802.11 b y g
- Implementa las capas PHY (fsica) y MAC (Acceso al Medio)
  - ARF
  - PER (Packet error Rate)
  - SINR (Signal to InterferenceplusNoise Ratio)
- Para usarla:
  - load <directorio>/ns-allinone-2.35/lib/libdei80211mr.so;

# Características

- Los agentes de tráfico se definen igual que en wired (UDP, TCP, CBR, FTP, etc, etc)
- Pueden existir nodos móviles
- Se debe definir un área en la que operarán los nodos (Topography)
- Se debe determinar las características del medio, ej: ruido
- Hay que determinar las posiciones de cada nodo
- Se debe crear un god (General Operations Director-información de la topología)

Ejemplos de simulaciones en:

<directorio>/ns-allinone-2.35/dei80211mr-1.1.4/samples/

# Puntos a destacar

**Definición de las características de los nodos wireless**

```
set val(chan) Channel/WirelessChannel ;# channel type
set val(prop) Propagation/TwoRayGround ;# radio-propagation model
set val(ant) Antenna/OmniAntenna ;# Antenna type
set val(ll) LL ;# Link layer type
set val(ifq) Queue/DropTail/PriQueue ;# Interface queue type
set val(ifqlen) 50 ;# max packet in ifq
set val(netif) Phy/WirelessPhy ;# network interface type
set val(mac) Mac/802_11 ;# MAC type
set val(rp) DSDV ;# ad-hoc routing protocol
```

# Puntos a destacar

## Configuración de los nodos

```
$ns_ node-config -adhocRouting
$val(rp)
-llType $val(ll) \
-macType $val(mac) \
-ifqType $val(ifq) \
-ifqLen $val(ifqlen) \
-antType $val(ant) \
-propType $val(prop) \
-phyType $val(netif) \
-topoInstance $topo \
-channel $chan \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF
```

# Puntos a destacar

## Configuración de los nodos

```
$ns_ node-config -adhocRouting
$val(rp)
-llType $val(ll) \
-macType $val(mac) \
-ifqType $val(ifq) \
-ifqLen $val(ifqlen) \
-antType $val(ant) \
-propType $val(prop) \
-phyType $val(netif) \
-topoInstance $topo \
-channel $chan \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF
```

Los nodos que sean creados luego de la definición anterior tendrán esas características

Con “ON” y “OFF” se elige que loguear y que no.

# Puntos a destacar

**Ejemplo de como crear un nodo**

```
set AP [$ns_ node ]
$AP random-motion 0
```

# Puntos a destacar

**Ejemplo de como crear un nodo**

set AP [$ns_ node ]
$AP random-motion 0

Defino la posición inicial:
$AP set X_ 5.0
$AP set Y_ 2.0
$AP set Z_ 0.0

# Agenda



# Traza normal

salida.out

```
+ 0.5 0 1 cbr 500 ------- 0 0.0 3.1 0 0
- 0.5 0 1 cbr 500 ------- 0 0.0 3.1 0 0
r 0.506 0 1 cbr 500 ------- 0 0.0 3.1 0 0
+ 0.506 1 3 cbr 500 ------- 0 0.0 3.1 0 0
- 0.506 1 3 cbr 500 ------- 0 0.0 3.1 0 0
+ 0.508929 0 1 cbr 500 ------- 0 0.0 3.1 1 1
- 0.508929 0 1 cbr 500 ------- 0 0.0 3.1 1 1
r 0.511 1 3 cbr 500 ------- 0 0.0 3.1 0 0
r 0.514929 0 1 cbr 500 ------- 0 0.0 3.1 1 1
+ 0.514929 1 3 cbr 500 ------- 0 0.0 3.1 1 1
- 0.514929 1 3 cbr 500 ------- 0 0.0 3.1 1 1
+ 0.517857 0 1 cbr 500 ------- 0 0.0 3.1 2 2
- 0.517857 0 1 cbr 500 ------- 0 0.0 3.1 2 2
r 0.519929 1 3 cbr 500 ------- 0 0.0 3.1 1 1
r 0.523857 0 1 cbr 500 ------- 0 0.0 3.1 2 2
+ 0.523857 1 3 cbr 500 ------- 0 0.0 3.1 2 2
- 0.523857 1 3 cbr 500 ------- 0 0.0 3.1 2 2
+ 0.526786 0 1 cbr 500 ------- 0 0.0 3.1 3 3
- 0.526786 0 1 cbr 500 ------- 0 0.0 3.1 3 3
r 0.528857 1 3 cbr 500 ------- 0 0.0 3.1 2 2
r 0.532786 0 1 cbr 500 ------- 0 0.0 3.1 3 3
+ 0.532786 1 3 cbr 500 ------- 0 0.0 3.1 3 3
- 0.532786 1 3 cbr 500 ------- 0 0.0 3.1 3 3
+ 0.535714 0 1 cbr 500 ------- 0 0.0 3.1 4 4
- 0.535714 0 1 cbr 500 ------- 0 0.0 3.1 4 4
r 0.537786 1 3 cbr 500 ------- 0 0.0 3.1 3 3
r 0.541714 0 1 cbr 500 ------- 0 0.0 3.1 4 4
```

Formato: event time FromNode ToNodo PktType PktSize Flags Fid SrcAdd DesAdd SeqNum PktId

# Traza de una simulación wireless

```
s 1.931451000 _0_ MAC  --- 0 ACK 14 [0 1 0 0]
r 1.931501000 _1_ MAC  --- 0 ACK 14 [0 1 0 0]
s 1.931751000 _0_ MAC  --- 2088 tcp 1074 [3c 1 0 800] ------- [0:0 1:0 32 0] [1053 0] 0 0
r 1.931937000 _1_ MAC  --- 2088 tcp 1040 [3c 1 0 800] ------- [0:0 1:0 32 0] [1053 0] 1 0
s 1.931947000 _1_ MAC  --- 0 ACK 14 [0 0 1 0]
r 1.931997000 _0_ MAC  --- 0 ACK 14 [0 0 1 0]
s 1.932127000 _1_ MAC  --- 2126 ack 74 [3c 0 1 800] ------- [1:0 0:0 32 0] [1053 0] 0 0
r 1.932165000 _0_ MAC  --- 2126 ack 40 [3c 0 1 800] ------- [1:0 0:0 32 0] [1053 0] 1 0
s 1.932175000 _0_ MAC  --- 0 ACK 14 [0 1 0 0]
r 1.932225000 _1_ MAC  --- 0 ACK 14 [0 1 0 0]
s 1.932375000 _0_ MAC  --- 2089 tcp 1074 [3c 1 0 800] ------- [0:0 1:0 32 0] [1054 0] 0 0
r 1.932561000 _1_ MAC  --- 2089 tcp 1040 [3c 1 0 800] ------- [0:0 1:0 32 0] [1054 0] 1 0
s 1.932571000 _1_ MAC  --- 0 ACK 14 [0 0 1 0]
r 1.932621000 _0_ MAC  --- 0 ACK 14 [0 0 1 0]
s 1.932791000 _1_ MAC  --- 2128 ack 74 [3c 0 1 800] ------- [1:0 0:0 32 0] [1054 0] 0 0
r 1.932829000 _0_ MAC  --- 2128 ack 40 [3c 0 1 800] ------- [1:0 0:0 32 0] [1054 0] 1 0
s 1.932839000 _0_ MAC  --- 0 ACK 14 [0 1 0 0]
r 1.932889000 _1_ MAC  --- 0 ACK 14 [0 1 0 0]
s 1.933019000 _0_ MAC  --- 2091 tcp 1074 [3c 1 0 800] ------- [0:0 1:0 32 0] [1055 0] 0 0
r 1.933205000 _1_ MAC  --- 2091 tcp 1040 [3c 1 0 800] ------- [0:0 1:0 32 0] [1055 0] 1 0
s 1.933215000 _1_ MAC  --- 0 ACK 14 [0 0 1 0]
r 1.933265000 _0_ MAC  --- 0 ACK 14 [0 0 1 0]
s 1.933695000 _1_ MAC  --- 2130 ack 74 [3c 0 1 800] ------- [1:0 0:0 32 0] [1055 0] 0 0
r 1.933733000 _0_ MAC  --- 2130 ack 40 [3c 0 1 800] ------- [1:0 0:0 32 0] [1055 0] 1 0
```

Formato: event time node level — pktnr type pktsize [MAC info] ...

# Referencias


- http://nsnam.isi.edu/nsnam/index.php/NS-2_Trace_Formats
- http://www.isi.edu/nsnam/ns/
- http://grouper.ieee.org/groups/802/11/
- G. Bianchi, Performance analysis of the IEEE 802.11 distributed coordination function, in IEEE Journal on Selected Areas in Communications, Vol 18, No. 3, pp 535-547, March 2000.


NS-2 es distribuido gratuitamente por la Universidad de Berkeley
Ver otras referencias en página del curso