MANUAL DE PRACTICAS DE LABORATORIO DE SISTEMAS DE TELECOMUNICACIONES. FIAD UABC.

ANTECEDENTES

El simulador utilizado para la realización de este laboratorio es el ns-2 (Network Simulator). En su página Web http://www.isi.edu/nsnam/ns/ se puede encontrar toda la información de interés acerca de ns-2. La interfaz hacia el usuario de ns-2 está basada en el lenguaje tcl/Tk.

Aquí http://www.beedub.com/book/3rd/Tclintro.pdf puedes encontrar una introducción "clásica" a Tcl/Tk (utilízalo únicamente como referencia... no es necesario que lo leas a priori...)

Se aconseja, especialmente, tener presente la siguiente información:

- El manual "oficial" de ns-2. http://www.isi.edu/nsnam/ns/ns-documentation.html
- Tutoriales de ns-2:
  - Tutorial de Marc Greis http://www.isi.edu/nsnam/ns/tutorial/index.html
  - NS by Example http://es.scribd.com/doc/6588147/NS-by-Example

**Practica 1. Instalar NS2 en tu computadora.**

Hay instrucciones precisas de cómo instalar el NS2, en su página oficial.

Existen varias opciones, decide cual es la que mejor te acomoda a tu experiencia con sistemas operativos.

- Software:
  - Versión de ns para cygwin (entorno Linux para MS Windows). Esta es la opción preferible si quieres instalarte ns en un sistema Windows. Para ello, antes tendrás que haber instalado cygwin (instalación muy sencilla). Cygwin es un entorno similar a Linux que se puede ejecutar en un sistema Windows. Su instalación es muy sencilla y se realiza mediante un selector de paquetes de instalación en el que no has de olvidar incluir al entorno gráfico X (conjunto de paquetes de nombre X11 en el instalador).
  - Otra opción es instalar un Sistema operativo linux, por ejemplo la distribución de Ubuntu, e instalar sobre linux el ns2.
  - Instalar una maquina virtual, y en ella instalar tu distribución de linux.

Verifica que ns2 esta bien compilado y que puedes correrlo desde la consola de linux, asi como el NAM, y XGRAPH.

**Practica 2. Conceptos de TCL, OTCL, y como iniciar NS y NAM.**

Una vez instalado el simulador, revisa los conceptos de OTCL y TCL, a través de efectuar el Tutorial de TCL, del MIT. http://www.isi.edu/nsnam/otcl/doc/tutorial.html

Debes escribir un script donde pongas en práctica los elementos de programación orientada a objetos presentados en este tutorial. Por ejemplo obtén tu propia bagel, a través de herencia..

Finalmente corre los scripts que presenta el libro NS for Begginers en su apartado 1.2.

http://www-sop.inria.fr/members/Eitan.Altman/COURS-NS/n3.pdf

Estos ejercicios reafirman el uso de OTcl y Tcl usando ns2 como interfaz.

Modifica en algo estos ejemplos para que se observe tu dominio sobre el tema.

**Practica 3. Familiarización con ns-2**

Esta práctica se realizara en 4 sesiones de laboratorio.

Para adquirir los conocimientos básicos necesarios para manejar ns-2, se propone en este ejercicio realizar algunos de los apartados del tutorial de Marc Greis. En concreto, los apartados de mayor interés son los: IV, V, VI, VIII y IX. En esos apartados, además de poder comprobar qué aspecto tiene un script ns-2, se podrá apreciar la importancia de dos herramientas auxiliares a ns-2: nam (Network Animator) y xgraph (generación de gráficos). De especial atención es el apartado VIII, que muestra como procesar datos de salida para mostrar parámetros de una simulación de manera gráfica.

Escribe todos los scripts que se presentan en esos apartados, y modifícalos para obtener tus propios resultados, variando, parámetros de la simulación, como numero de nodos, tipos de trafico, protocolos de colas, ancho de banda de enlaces, retardos, etc. Debes anexar en el reporte, tus modificaciones a los scripts, y tus resultados de manera grafica, anexando imágenes del NAM, y graficas que obtengas de procesar tus datos.

Practica 4. Como analizar los archivos de salida generados por NS2.

Esta práctica se realizara en 4 sesiones de laboratorio.

Realiza las simulaciones que se presentan en el libro Ns for beginners http://www-sop.inria.fr/members/Eitan.Altman/COURS-NS/n3.pdf, en los capitulos 3 y 4.

Posteriormente realiza los ejercicios del Ns by example http://perform.wpi.edu/NS/, unicamente el apartado de post simulation.

De manera similar a la practica 3, modifica los scripts para que obtengas tus propios resultados y conclusiones, y se muestre evidencia de que entiendes como funcionan los scripts que procesan los archivos de salida de estas practicas.

Practica 5. Utilización de ns-2 para estudiar el comportamiento de protocolos

Esta práctica se desarrollara en 2 sesiones de laboratorio.

En este ejercicio se va a intentar ilustrar algunos de los mecanismos ya conocidos de TCP usando ns-2.

*Apartado 1: Slow Start*

Escribe un script que establezca una simulación que permita evaluar el desempeño de este mecanismo de TCP. Debe mostrarse cómo se puede parametrizar el modelo de simulación. Los parámetros de la simulación que se pueden modificar:

a) El tamaño de ventana de recepción de uno de los nodos del modelo de simulación.

b) El tamaño del *buffer* de paquetes en el nodo r1.

c) La velocidad binaria del enlace entre los nodos s1 y r1.

d) La velocidad binaria del enlace entre los nodos r1 y r2.

e) El retardo de propagación del enlace entre los nodos r1 y r2.

f) Un indicador (YES/NO) de si el mecanismo "slow start" está activado o no.

g) Un indicador (YES/NO) de si el mecanismo "ack retardados" está activado o no. "ack retardados" en ms.

Mediante nam y ajustando la velocidad de reproducción de la simulación, intenta ver la relación entre las variables TCP monitorizadas: `cwnd_, ack_ y t_seqno_`. ¿Entiendes el funcionamiento del mecanismo "slow start"?

¿Qué efectos en la tasa binaria efectiva de la conexión entre s1 y s2 tiene aumentar/disminuir la velocidad binaria entre los nodos `node_(r1)` y `node_(r2)` del script?

Para contestar a esta pregunta puedes recurrir a simulaciones de barrido de parámetros. Si asumimos que conseguir en un mismo tiempo de simulación un mayor número de segmentos TCP enviados es un indicador de una mayor tasa binaria efectiva, podrías intentar "barrer" varios valores de la velocidad binaria efectiva intentando obtener una gráfica X-Y que represente el último número de secuencia TCP enviado en el eje Y y los valores de la velocidad binaria en el eje X.

- Usando el mismo paquete de simulación, intenta reproducir los resultados mostrados en la figura 3 de [1]. Para ello, crea una simulación individual con valores para los parámetros:
  - Window size (pkts): 32
  - Queue size in r1 (pkts): 30
  - s1-r1 bitrate (Mbps): 10
  - r1-r2 bitrate (Mbps): 0.23
  - r1-r2 delay (ms): 300
  - slowstart (YES/NO) NO
  - delayed ack (YES/NO) NO

- delayed ack timeout (cualquier valor numérico)

Puedes visualizar los resultados de la misma manera que se indicó en el punto anterior.

- ¿Cuál es, aproximadamente, la tasa binaria efectiva conseguida en los 10 segundos de simulación?
    - ¿Es mejor o peor que en el caso en que sí está activado “slow start”?
- Vamos a intentar resolver la congestión pagando más dinero por un enlace entre los dos nodos intermedios más “rápido”. Cambia la velocidad entre los nodos r1 y r2 a 1Mbps. Además, vamos a intentar conseguir un envío continuo aumentando la ventana de transmisión de tal manera que aproximadamente (VENTANA-1)*tiempo_transmisión = 2*retardo_ida_y_vuelta. Por ello cambia el parámetro Window size a 145.
    - ¿Ha mejorado mucho la tasa binaria efectiva con respecto a la situación anterior? ¿Por qué?
- Aumenta ahora el tamaño de las colas de los nodos a 500. ¿Qué sucede? ¿Cómo es ahora la tasa binaria efectiva?
    - ¿Qué conclusiones obtienes ahora sobre qué parámetros de una red hay que modificar para solventar situaciones de congestión?

      Puedes intentar ilustrar esas conclusiones con simulaciones de barridos de parámetros (por ejemplo, para visualizar cómo mejora/empeora la tasa binaria efectiva con cambios en el tamaño de las colas de los nodos, el retardo, etc.).

*Apartado 2: Slow Start/Congestion Avoidance/Fast Retransmit y Fast Recovery*

- Slow Start, como has visto en el apartado anterior, intenta “llenar” poco a poco un camino extremo-a-extremo de segmentos TCP. Para ello emplea la ventana de congestión `cwnd` que va aumentando exponencialmente, si no hay congestión, a medida que se reciben los asentimientos.

- El algoritmo “Congestion Avoidance” intenta que, bajo determinadas circunstancias, el crecimiento de `cwnd` no sea exponencial para tratar de evitar potenciales situaciones de congestión (en otras palabras, se “frena” un poco el envío de nuevos segmentos de datos). La idea es la siguiente: “Congestion Avoidance” fija un umbral ssthresh de tal manera que:

    1. si cwnd < ssthresh entonces estamos en situación de “slow start” y cwnd se incrementa en uno con cada ack recibido.

    2. Si cwnd > ssthresh entonces estamos en situación de “congestion avoidance” y cwnd aumenta linealmente. En este último caso, con cada ack: cwnd = cwnd + 1/cwnd. (NOTA: el valor de cwnd indica un número de segmentos, no de bytes) ¿Esto es un aumento lineal? En realidad con esta expresión se consigue un aumento lineal CADA RTT (Round Trip Time, el tiempo que tardan los datos en ir y volver de un extremo a otro de la red). Imagínate que cwnd sea 4. Entonces se pueden enviar 4 segmentos de datos SEGUIDOS. Si no hay congestión, se recibirán los correspondiente 4 acks SEGUIDOS y, si estamos en situación de “congestion avoidance” entonces cwnd pasará a ser aproximadamente 5. (NOTA: esto se produce asumiendo que cada segmento TCP de datos genera, en recepción, un ack... piensa qué ocurrirá si tenemos activados mecanismos como el de acks retardados...).

- En TCP se asume que, si los temporizadores de retransmisión están bien dimensionados, el tener que retransmitir un segmento indica que hay congestión (en la redes de hoy en día es difícil que se descarte un segmento por errores en la transmisión y es mucho más probable que los descartes se deban a situaciones de “llenado” de buffers en nodos intermedios). Del mismo modo, recibir acks “duplicados” (más de un ack asintiendo los mismos datos) indica que el receptor está recibiendo segmentos “fuera de secuencia”. Como la probabilidad de que los segmentos se desordenen en tránsito es pequeña, se podría asumir que el problema no es que haya desorden sino que es que algún segmento se ha perdido (probablemente por descarte debido a congestión). Dicho de otro modo, la recepción de acks “duplicados” también se puede interpretar como indicación de que hay que retransmitir un segmento descartado por congestión.

  Dicho esto, ¿Cómo se inicializan cwnd y ssthresh y cómo cambian cuando hay situaciones de congestión? Ello depende de si tenemos activados los mecanismos de “Fast Retransmit y Fast Recovery”:

    - Sin “Fast Retransmit y Fast Recovery

        - cwnd se inicializa a 1. ssthresh generalmente se inicializa a 65535 (o a un valor grande).

        - Si hay que retransmitir un segmento (por vencimiento de temporizador o por recepción de acks duplicados):

        - ssthresh = max(2, 0.5*min(cwnd, ventana de recepción)).
        - cwnd se reinicializa a 1 (siempre se comienza con una situación de “slow start”)

- Con “Fast Retransmit y Fast Recovery” (aquí se describe cómo funciona ns-2.
    - cwnd se inicializa a 1. ssthresh generalmente se inicializa a 65535 (o a un valor grande).
    - Si hay que retransmitir un segmento por vencimiento de temporizador:
        - ssthresh = max(2, 0.5*min(cwnd, ventana de recepción)).
        - cwnd se reinicializa a 1 (lo mismo que en la situación anterior)
    - Si se reciben más de 3 acks duplicados:
        - ssthresh = max(2, 0.5*min(cwnd, ventana de recepción)).
        - cwnd = ssthresh (se pasa directamente a la situación de “congestion avoidance”).