Redes de computadoras, 3 de diciembre de 2024

Estimación del temporizador de retransmisión (*RTO, retransmission timeout*) en TCP

Martínez Buenrostro Jorge Rafael

correo, molap96@gmail.com
Universidad Autónoma Metropolitana
Unidad Iztapalapa, México

Introduccion

El protoclo de transmisión (TCP) estima el proceso del RTT para predecir el tiempo de espera (timeout) de la fuente, a fin de ajustar el temporizador de retransmisión. El emisor TCP mide el RTT desde el momento que se envía un segmento hasta recibir el acuse de recibo (ACK) correspondiente

Procedimiento

Para poder comenzar con esta práctica selecione cuatro trazas al azar de los sets proporcionados, dichas trazas son:

SetA - hop02

SetD - hop04

SetG - hop21

SetF - hop29

El siguiente paso es crear un script en AWK para generar las trazas del proceso RTT (*sampleRTT*), su estimación (*estimatedRTT*) y el valor del temporizador (*TimeoutInterval*). La base para este script son las secciones 2.1 a 2.3 del **RFC 6298**. En la figura siguiente se puede ver el código del script

```
BEGIN {
    # Inicializacion de parametros
    alpha = 1/8   # Suavizado para SRTT
    beta = 1/4    # Suavizado para RTTVAR
    K = 4         # Factor para RTTVAR
    G = 1         # Valor minimo para RTTVAR
    RTO = 1       # Valor inicial de RTO
    firstRTT = 1  # Bandera para la primera medicion RTT
    sample_count = 0 # Contador de muestras procesadas
    start_sample = 200 # Comenzar a partir de la muestra 200
    max_samples = 30   # Tomar solo 30 muestras
}

{
    # Incrementar el contador de muestras
    sample_count++

    # Solo procesar muestras a partir de la muestra 200
    if (sample_count >= start_sample && sample_count < start_sample +
        max_samples) {
        # RTT_value es el unico valor por linea
        RTT = $1

        # Primer RTT
        if (firstRTT == 1) {
            SRTT = RTT
            RTTVAR = RTT / 2
            RTO = SRTT + (K * RTTVAR > G ? K * RTTVAR : G)
            firstRTT = 0
        } else {
            # RTT subsecuentes
            RTTVAR = (1 - beta) * RTTVAR + beta * (SRTT > RTT ? SRTT - RTT :
                RTT - SRTT)
            SRTT = (1 - alpha) * SRTT + alpha * RTT
            RTO = SRTT + (K * RTTVAR > G ? K * RTTVAR : G)
        }

        # Escribir en los archivos de salida
        print RTT >> "sampleRTT"
        print SRTT >> "estimatedRTT"
        print RTO >> "timeoutInterval"
    }

    # Si ya se procesaron 30 muestras, terminamos el script
    if (sample_count >= start_sample + max_samples) {
        exit
    }
}

END {
    print "Proceso completado. Los archivos de salida son: sampleRTT,
        EstimatedRTT, TimeoutInterval."
}
```

Figura 1: Script para extraer los datos requeridos

Una vez creado lo ejecutamos para cada una de las trazas seleccionadas, a continuación se muestra la forma de ejecución

```
awk -f rfc6298.awk hop02.txt
```

Figura 2: Ejecución del script para la traza *hop02.txt*

Como se puede ver en la *Figura 1* al ejecutar el script se generan tres trazas: **sampleRTT**, **estimatedRTT** y **timeoutInterval**. El siguiente paso es crear las instrucciones en *Octave* para poder visualizar en una sola gráfica: la traza original y las trazas generadas por el script. A continuación se muestran las instrucciones propuestas, a reserva del nombre de los ejes y el titulo que cambiará al graficar cada una de las trazas. Además veremos las gráficas creadas por defecto.

```
load estimatedRTT;
load timeoutInterval;
plot(sampleRTT, '--*', 'Color', 'b', 'LineWidth', 0.5, 'MarkerSize', 8);
hold on;
plot(estimatedRTT, '--*', 'Color', 'k', 'LineWidth', 0.5, 'MarkerSize', 8);
plot(timeoutInterval, '--*', 'Color', 'r', 'LineWidth', 0.5, 'MarkerSize', 8)
    ;
legend('sampleRTT','estimatedRTT','timeoutInterval');
grid on;
xlabel('Eje x');
ylabel('Eje y');
title('Erro incurrido');
print -dpng "traza.png";
```

Figura 3: Ejecución del script para la traza *hop02.txt*