UERJ – Universidade do Estado do Rio de Janeiro
PEL / FEN e CComp / IME

Prof. Alexandre Sztajnberg 2014/2

Lista 1 – *Sockets*

Neste projeto vamos fazer alguns testes com *sockets* UDP e TCP, usando a linguagem C, C++ ou Java. Temos um cliente, de onde partem as mensagens para um servidor, que retorna a mensagem. Queremos fazer medidas de desempenho. A maioria dos cálculos e *logs* são feitos no cliente.

Investigar, em trabalhos relacionados com avaliação de desempenho usando medidas reais de comunicação através de socket, parâmetros importantes a serem considerados:

- quantas medidas são necessárias para um intervalo de confiança de 98%
- tempo ou % de mensagens de *warmup*
- cálculo de média, mediana, desvio padrão
- ver como *plotar* isso em uma planilha (tem que dominar isso!!!)
- investigar e discutir como podemos medir a variação de atraso (*jitter*)
-

PS.: podemos fazer isso coletivamente.

Para todos os testes, temos que medir:

- o RTT (*round trip time*), ou seja o tempo de ida-e-volta de uma mensagem (porque não podemos medir o tempo só de ida? Ou melhor como fazê-lo?);
- a variação de atraso no servidor e no cliente;
- para todas estas medidas temos que ter informações como média, mediana, DVP, etc. e o intervalo de confiança. Estes elementos tem que ficar bem claros para vocês. Por exemplo, porque 1000 medidas e não 100? 1000 são suficientes? Por que?

O cliente envia uma mensagem par ao servidor, aguarda a resposta e faz os cálculos e logs necessários. As mensagens devem ter o tamanho 1 (se for possível), 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048 ... 32768 bytes. Verificar se o que está sendo transmitido é efetivamente o número de bytes desejado (ou seja, se um buffer grande for pré-alocado e depois preenchido com 8 bytes, o que está sendo transmitido é o buffer grande – e não 8 bytes), com o WireShark.

Experimental 1. Introduzir atrasos para simular:

- Redes reais de 10, 100 e 1000 Mbps
- Compartilhamento de canal com aplicações "matadoras" (fluxo de vídeo ou áudio, por exemplo)

Possíveis fontes:
http://code.google.com/p/xjperf/
http://wanem.sourceforge.net/
http://www.trafficsqueezer.org/
http://www.linuxfoundation.org/collaborate/workgroups/networking/netem
http://info.iet.unipi.it/~luigi/dummynet/

Experimental 2. Vamos fazer o cliente funcionar, também, como um gerador de rampa, para cada tamanho de mensagem. Ou seja, o cliente começa a fazer requisições em uma taxa mais lenta, até uma taxa mais rápida, e depois desce novamente. Para isso, não será possível simplesmente fazer um loop envia-recebe-calcula-envia_novamente. É preciso controlar a taxa de requisições por segundo (ou milisegundo). A alternativa para um teste parecido seria aumentarmos o número de clientes, de forma a aumentar a taxa de requisições em uma faixa de tempo. Combinar os dois também seria interessante. Vamos discutir isso em sala de aula.

O "servidor" não faz quase nada, apenas devolve uma resposta para o cliente com o mesmo número de bytes (o objetivo é medir a transmissão). Mas, tem que fazer a contabilidade da variação do atraso.

Para facilitar o teste em máquinas diferentes, o número IP e a porta do "servidor" deve poder ser passada como parâmetro.

Coletar as informações (média, desvio padrão), gerar uma saída na forma abaixo, e colocar numa planilha (Excell, por exemplo) para podermos verificar e comparar as curvas de medidas.

UDP/10	1	4	8	16	32	64	128...
MED X	X	X	X	X	X	X	...
DVP Y	Y	Y	Y	Y	Y	Y	...

Testes (podemos negociar):

Felipe. cliente TCP, servidor TCP – estabelecendo uma conexão a cada mensagem, 100Mbps

Galvani. 1 cliente TCP, servidor TCP – estabelecendo a conexão apenas no início; 100Mbps

Jailton. 2 clientes TCP, servidor TCP concorrente - estabelecendo a conexão apenas no início, 100Mbps

Ricardo. 1 cliente UDP, servidor UDP, 100Mbps

Rodrigo. 1 cliente TCP, servidor TCP, 10 Mbps

Thais. 1 cliente TCP, servidor TCP, 1000 Mbps