INFO7015 - Tópicos em Redes de Computadores - Trabalho Prático 2

Profa. Dra. Michele Nogueira

Aluna: Carolina Moraes

Obs: “Potência” deverá ser tratada como “valor” no trabalho.

Exercício de aquecimento A [10%]: Varie o tamanho da janela fixa editando controller.cc para ver o que acontece. Faça um gráfico 2D de taxa de transferência versus um atraso no sinal com percentil de 95 à medida que você altera esse valor. Qual é o melhor tamanho de janela única que você pode encontrar para maximizar a potência (taxa de transferência/atraso)? Quão repetíveis são as medições feitas com o mesmo tamanho de janela em várias execuções?

R: Eu testei vários tamanhos de janelas diferentes, no fim acabei ficando com os valores sendo incrementados de 5 em 5. Conforme mostrado na TABELA 1. A coluna valor foi calculada como Taxa de transferência / Atraso, e o melhor valor é mostrado em destaque.

TABELA 1 - TAXA DE TRANSFERÊNCIA/ATRASO X TAMANHO DA JANELA FIXA

Tamanho da Janela	Taxa de Transferência (Mbits/s)	Atraso Sinal Percentil de 95 (ms)	Valor
5	1,05	109	9,63
10	1,93	155	12,45
15	**2,66**	**212**	**12,55**
20	3,26	277	11,77
25	3,73	343	10,87
30	4,07	401	10,15
35	4,32	453	9,54
40	4,51	504	8,95
45	4,65	557	8,35
50	4,76	607	7,84
55	4,85	652	7,44
60	4,91	711	6,91
65	4,94	763	6,47
70	4,96	808	6,14
75	4,98	855	5,82
80	4,99	896	5,57
85	5	935	5,35
90	5	972	5,14

Os valores das colunas Taxa de Transferência e Atraso Sinal Percentil de 95 serão mostrados em um gráfico 2D como Taxa de transferência x Atraso de Sinal Percentil 95, segue GRÁFICO 1.

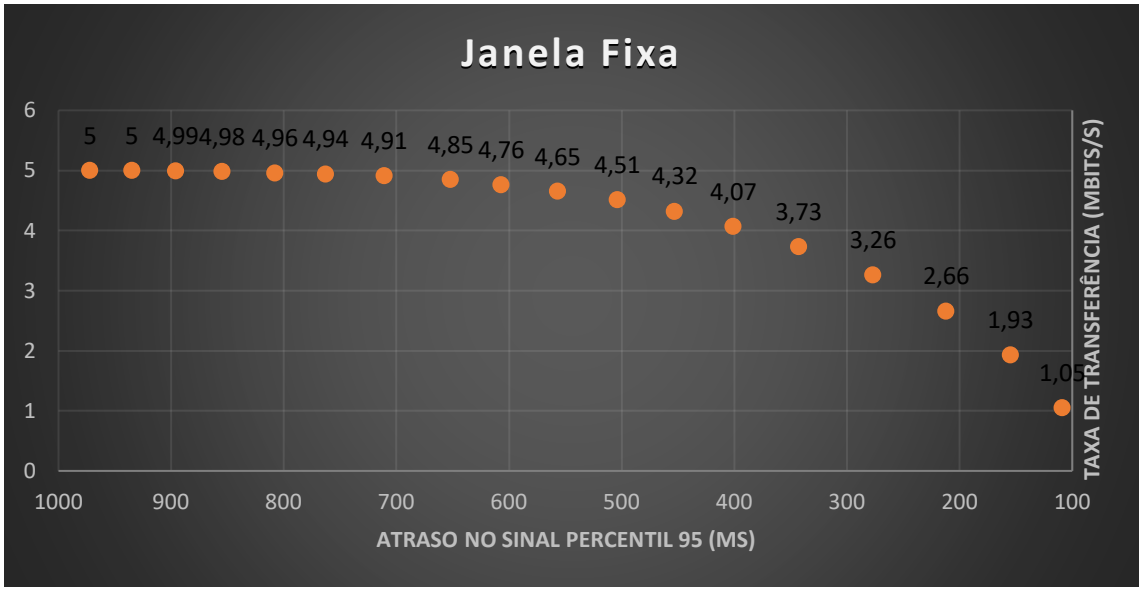

GRÁFICO 1

Para responder as questões do exercício A, descobri que a melhor pontuação para a coluna valor foi em um tamanho de janela fixo de aproximadamente 15 datagramas. As medições feitas com o mesmo tamanho de janela em várias execuções foram muito repetíveis no ambiente de testes disponibilizado para o trabalho (mahimahi). A Taxa de Transferência e o Atraso variaram muito pouco, acabei não detalhando devido a pouca variação.

Exercício de aquecimento B [40%]: Implemente um esquema AIMD simples, semelhante à fase de prevenção de congestionamento do TCP. Quão bem isso funciona? Quais constantes você escolheu?

R:Para este Exercício, implementei um protocolo AIMD simples, que incrementa a *cwnd* por *alfa/cwnd* em cada *ack* e diminui a janela por beta em um evento de perda, por exemplo, *cwnd =cwnd/beta*. Isso imita o comportamento do AIMD do TCP na prevenção de congestionamentos. Assim, para sinalizar quando uma diminuição multiplicativa deve ocorrer, usei um valor de tempo limite fixo como um sinal. Defini o tempo limite para 80 ms, que é aproximadamente o atraso de sinal de percentil de 95 nos algoritmos superiores. Também defini o tamanho inicial da janela de congestionamento para 10 datagramas, com base nos resultados experimentais do exercício A e para aplicar aprimorei no tamanho da janela, conforme proposto na RFC 6928.

Testei alguns valores diferentes para *alfa* e *beta* e registrei o desempenho deles na TABELA 2. Descobri que os valores típicos de alfa = 1 e beta = 2 têm desempenho pior do que muitos dos tamanhos de janela fixos que testei no Exercício A. Em particular, o AIMD aumentou a latência à medida que o algoritmo tenta usar lentamente toda a Taxa de transferência disponível. Em seguida, ajustei as constantes para ver como seria no *cwnd,* afetou o crescimento da pontuação da coluna valor da tabela.

TABELA 2 - TAXA DE TRANSFERÊNCIA E ATRASO X CONSTANTES AIMD

Alpha	Beta	Taxa de Transferência (Mbits/s	Atraso Sinal Percentil de 95 (ms)	Valor
1	2	4,72	847	5,57
1	3	4,68	765	6,12
1	4	4,65	734	6,34
1	5	4,64	724	6,41
1	10	4,59	705	6,51
0,1	1,1	3,8	377	10,08
0,1	1,5	2,72	142	19,15
0,1	2	2,43	186	13,06
0,5	2	4,41	570	7,74
0,2	4	3,23	267	12,10
2	2	4,88	1190	4,10
4	2	4,96	1655	3,00
4	10	4,95	1533	3,23

Em geral, descobri que um aumento lento aditivo (com alfa <1), combinado com uma diminuição multiplicativa passiva (beta <2), proporcionou uma melhor pontuação para a coluna valor do que outras configurações que testei. No entanto, mesmo neste melhor caso, apenas cerca de metade da taxa de transferência disponível foi utilizada. Mesmo os melhores valores constantes que encontrei (mostrados em laranja na TABELA 2) não são ideais.

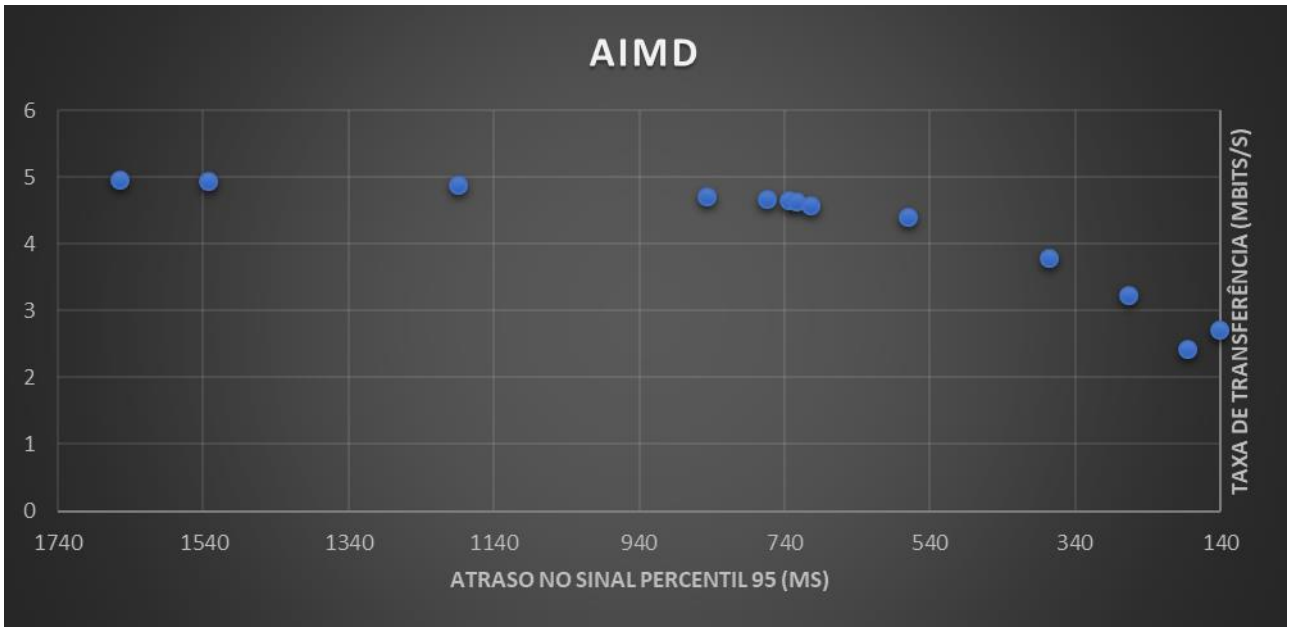

GRÁFICO 2

Os valores disponíveis na TABELA 2 são mostrados no GRÁFICO 2. Comparado com o GRÁFICO 1 da janela fixa, observa-se que dá para alcançar uma Taxa de transferência ligeiramente melhor e atraso melhor, mas em geral têm atrasos maiores do que os tamanhos mais baixos das janelas fixas.

Exercício D [50%]: Tente diferentes abordagens e trabalhe para maximizar a potência (compare com os resultados dos demais colegas da sala e tente melhorar). Em seu relatório, explique sua abordagem, incluindo as decisões importantes que você teve que tomar e como você as fez. Inclua gráficos ilustrativos com os resultados.

R:Para o desenvolvimento desta etapa, consultei algoritmos disponíveis que cito nas referências. Construí a ideia de usar um aumento e diminuição aditivos, conforme alguns algoritmos de controle de congestionamento. Foi utilizado o feedback dos datagramas reconhecidos para definir o limite do RTT. Neste algoritmo, foi usado um aumento aditivo (controlado por alfa) e um decréscimo aditivo (controlado por beta). Em todos os momentos, o algoritmo está aumentando ou diminuindo, com base em uma média móvel exponencialmente ponderada dos RTTs vistos, e o RTT atual de um datagrama específico. Em um nível alto, quando os RTTs vistos são estáveis, ou dentro de um intervalo especificado do RTT médio, a janela de congestionamento é adicionalmente aumentada. No entanto, quando um RTT é relatado que excede essa permissão em torno do RTT médio, ele sinaliza congestionamento e o algoritmo diminui a janela de congestionamento para evitar o inchaço das filas de gargalo. Quando a rede está congestionada, a média móvel exponencialmente ponderada também é redefinida para o tempo estimado de propagação de ida e volta, que serve como uma linha de base do RTT esperado em uma rede não congestionada.

Várias observações motivaram o desenho deste algoritmo. Primeiro, entendi com o Exercício A e B que, em geral, um algoritmo que reage ao atraso em vez de perda ou timeout do datagrama é capaz de responder ao congestionamento muito mais rapidamente e é capaz de manter as filas relativamente pequenas. Em segundo lugar, a partir do entendimento dos algoritmos utilizados num dos artigos, percebi que o algoritmo desenvolvido deve ser capaz de evitar ficar preso em um estado estável indesejável. No caso, vi que ter dois estados simples (aumentar ou diminuir) era uma maneira intuitiva de garantir que o algoritmo fosse responsivo. Essa foi apenas uma escolha que fiz para simplificar, outros algoritmos como o BBR (visto nos artigos lidos no início da disciplina) usam estados mais complexos (por exemplo, ProbeBW) para garantir que a taxa de transferência disponível seja rapidamente utilizada à medida que ela muda continuamente. Com esse design geral do algoritmo desenvolvido para este trabalho, passei a testar o desempenho de vários parâmetros: (1) alfa, que define a quantidade de aumento de aditivo a ser aplicada ao cwnd; (2) beta, que define a quantidade de redução de aditivo para cwnd; (3) rtt _allowance, que define quanto uma medição de RTT pode exceder o RTT médio enquanto ainda está sendo considerada estável; (4) ewma_weight, que defini o peso aplicado à medição atual do RTT no EWMA do RTT; e timeout, que definiu o RTO em milissegundos antes de outro datagrama ser enviado. Alguns dos resultados da exploração são mostrados na TABELA 3.

TABELA 3

alfa	beta	rtt_allowance	ewma_weight	timeout	taxa de transferência (Mbits/s)	atraso sinal percentil 95 (ms)	valor
1	2	1,4	0,15	1000	4,94	2064	2,39
1	4	1,4	0,15	1000	4,85	1902	2,55
1	5	1,4	0,15	1000	4,85	1896	2,56
1	10	1,4	0,15	1000	4,67	1788	2,61
2	5	1,4	0,15	1000	4,98	2807	1,77
2	5	1,4	0,01	1000	3,25	120	27,08
2	5	1,4	0	1000	2,92	114	25,61
5	2	1,4	0,01	1000	4,71	1853	2,54
2	5	1,1	0,01	1000	0,42	111	3,78
2	5	1,5	0,01	1000	4	643	6,22
2	5	1,4	0,01	100	3,29	105	31,33
2	5	1,4	0,01	50	3,24	83	39,04
2	5	1,4	0,01	25	3,47	86	40,35
2	5	1,4	0,01	15	3,55	98	36,22

A partir dos experimentos, e observando o comportamento dos gráficos, entendi que a escolha inicial do EWMA_weight (0,15) permitiu que a média se movesse muito rapidamente, permitindo assim que o algoritmo enchesse os buffers. Em vez disso, a redução desse peso de forma que a média permaneceu relativamente próxima da estimativa inicial da propagação de atraso, e ainda reduzindo significativamente o atraso de sinal de percentil 95. No entanto, também descobri que eliminar totalmente o peso poderia afetar negativamente a coluna valor. Uma abordagem alternativa seria eliminar a média do EWMA e usar alguma variância estatística para determinar quais RTTs considerariam estáveis, conforme visto em aula, mas essa alternativa não foi explorada, pois nunca utilizei antes. Em seguida, descobri que era importante no algoritmo AIAD, garantir que a redução aditiva ocorresse a uma taxa mais rápida do que o aumento aditivo. Intuitivamente, isso permite que o algoritmo feche mais rapidamente a janela quando o congestionamento é detectado. Se fosse usado uma redução ainda mais agressiva (por exemplo, exponencial), a taxa de transferência seria desperdiçada, pois apenas algumas diminuições poderiam quase fechar completamente a janela.

Quando testei a provisão de RTT, que define quanto um RTT medido pode exceder a média enquanto ainda está sendo considerado "estável", inicialmente foi selecionado um valor de 1,4 devido as observações de um experimento com um tamanho de janela fixo de 1 datagrama. Em particular, o RTT mínimo era de 42ms e que os RTTs de pacotes individuais podiam variar significativamente de cerca de 40ms a cerca de 60ms. Outras experiências com uma pequena permissão mostram que a pequena margem é muito limitante e prejudica o desempenho da taxa de transferência. Consequentemente, uma permissão de cerca de 1,4 (ou seja, um valor RTT medido que é 1,4 vezes maior que a média pode ser considerada estável). Isso pareceu funcionar bem devido à natureza do rastreamento da rede celular, que resultou em grandes picos no RTT quando a Taxa de transferência do link caiu. Semelhante à média do RTT, o objetivo disso é apenas definir o que é um RTT “estável”, e uma abordagem alternativa poderia estar usando estatísticas sobre os RTTs vistos, ao invés de uma provisão fixa como foi implementado aqui. Descobri que o ajuste dos quatro primeiros parâmetros resultou em um número para a coluna valor um pouco menor que 30, reduzir o valor do timeout melhorou significativamente os números da coluna valor. Possivelmente, isso se deva ao mal-entendido inicial de “atraso de sinal”. Anteriormente, foi assumido que estava tentando minimizar o atraso de datagramas individuais. No entanto, o “atraso de sinal” também é negativamente afetado pelo tempo morto, quando nenhum datagrama está sendo enviado. Diminuindo o tempo limite, garanti

que há um mínimo de datagramas sendo enviados, a uma taxa que seja baixa o suficiente para não estourar buffers, mas com frequência suficiente para garantir que o sinal de atraso não seja afetado por segundos de tempo morto. No geral, o algoritmo foi mais sensível a alterações no RTT allowance, EWMA weighting e timeout. A escolha de alfa e beta teve menos efeito, desde que ambos os valores fossem relativamente pequenos e beta fosse maior que alfa.

Como ponto de comparação, demonstrei no GRÁFICO 3 as execuções de cada um dos exercícios propostos no para o trabalho 2. Percebi que o uso da abordagem avançada de AIAD baseada em atraso permitiu reduzir significativamente o atraso de sinal de percentil de 95, mantendo uma taxa de transferência razoável do que nossas outras tentativas, resultando nas melhores pontuações de potência.

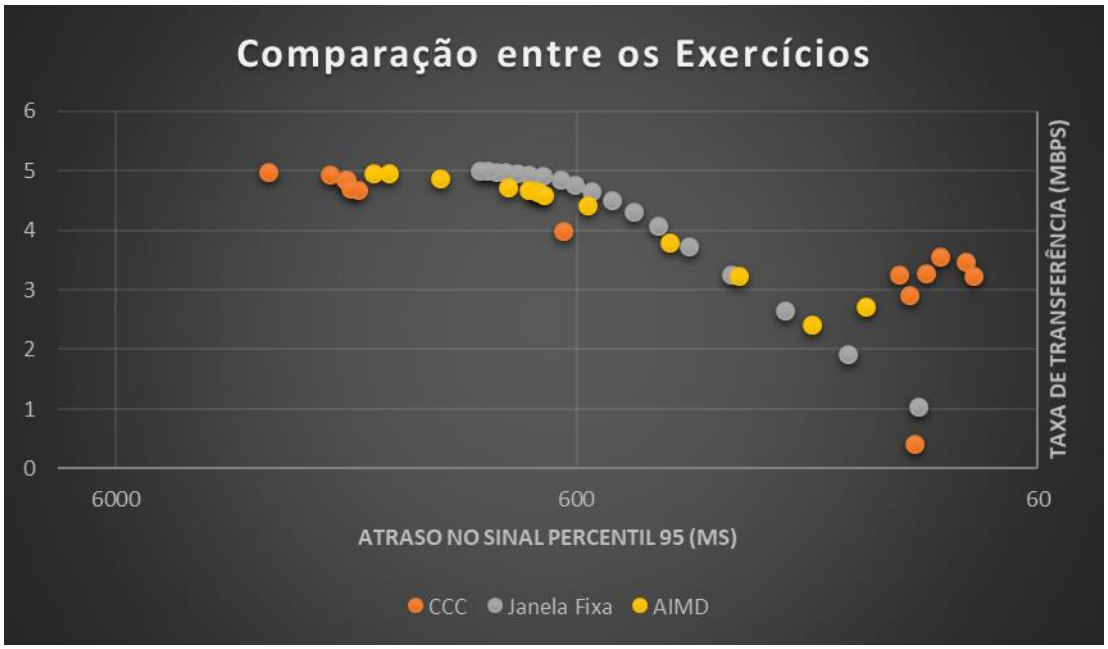

GRÁFICO 3

Na FIGURA 1, dá para observar que o algoritmo é conservador, pois frequentemente não utiliza totalmente a largura de banda disponível. Além disso, está simplesmente fazendo um aumento aditivo, observa-se que o algoritmo também pode ser um pouco lento para crescer e utilizar toda a largura de banda disponível. Por exemplo, em cerca de 70 segundos no trace, a largura de banda disponível aumenta, mas o algoritmo aumenta adicionalmente sua janela de congestionamento e perde um pouco da taxa de transferência disponível. Uma ideia que pode ter abordado isso seria implementar um estado de sondagem curto como no BBR, onde o BBR pode convergir exponencialmente rápido para o aumento recente da *bottleneck bandwidth*.

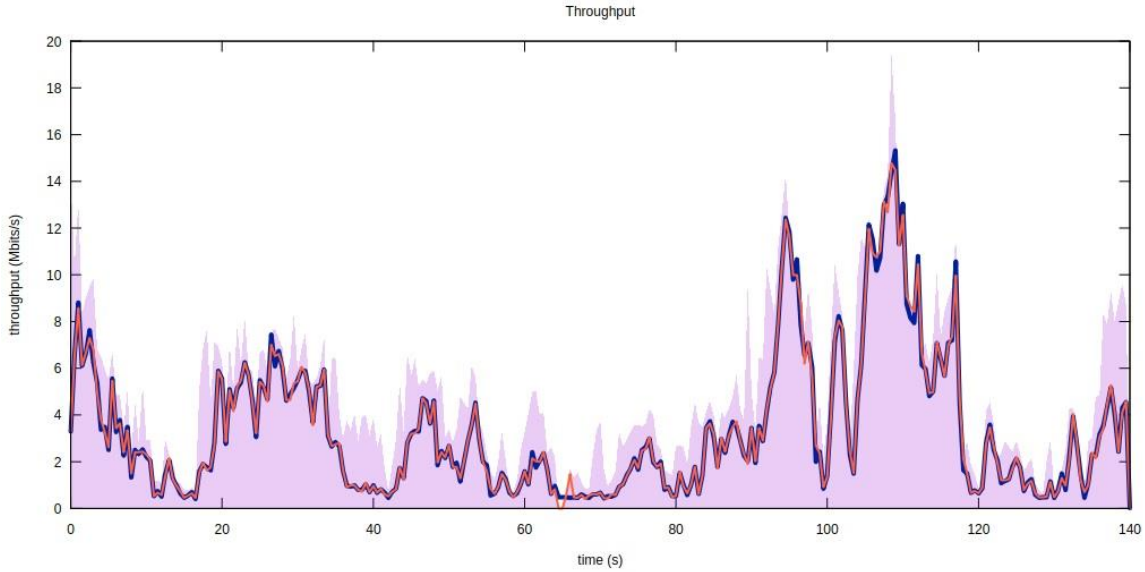

FIGURA 1: CCC RODANDO COM ALFA = 2, BETA = 5, RTT_ALLOWANCE = 1,4, EWMA_WEIGHT = 0,01 E TIMEOUT = 25

Em contraste, a FIGURA 2 mostra o comportamento do CCC com alfa = 5 e beta = 2. Note que ajustando o algoritmo de forma que o aumento aditivo ocorra mais rápido que a diminuição aditiva, o algoritmo é muito mais agressivo e mais rápido para preencher a largura de banda disponível. No entanto, isso resulta em um atraso de sinal de percentil 95 muito maior e uma pontuação para a variável valor de cerca de 23. Usei esses gráficos de taxa de transferência disponíveis na página do ranking que o pessoal da turma localizou, para obter uma noção visual do comportamento do algoritmo em várias configurações de parâmetros e saber como ajustar os parâmetros para melhores os valores em geral.

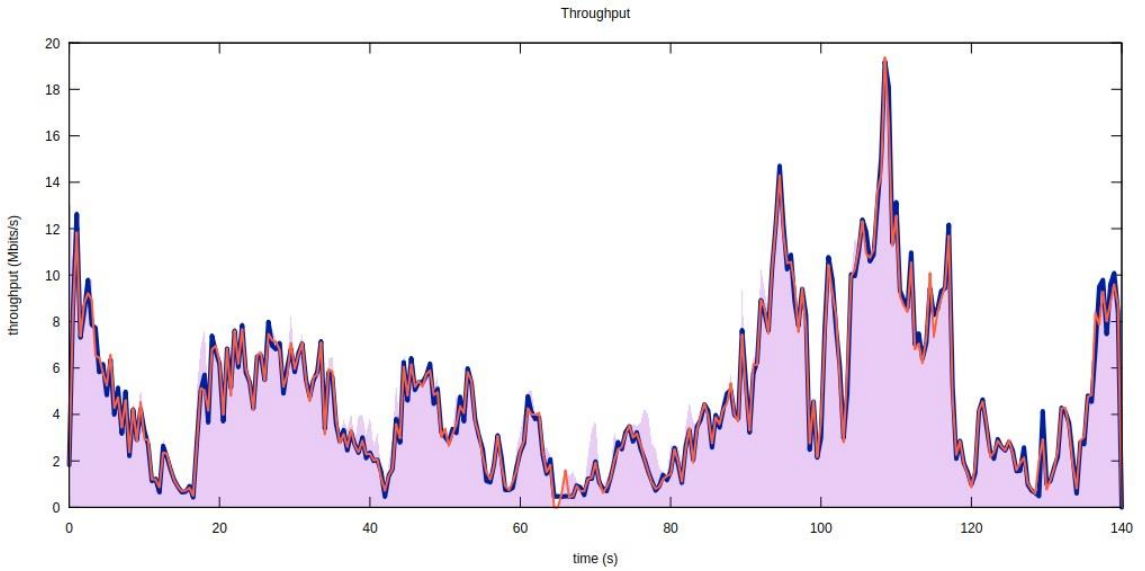

FIGURA 2: CCC RODANDO COM ALFA = 5, BETA = 2, RTT_ALLOWANCE = 1,4, EWMA_WEIGHT = 0,01 E TIMEOUT = 25

```
Average capacity: 5.04 Mbits/s
Average throughput: 3.34 Mbits/s (66.2% utilization)
95th percentile per-packet queueing delay: 46 ms
95th percentile signal delay: 84 ms
```

FIGURA 3

A FIGURA 3 mostra uma das execuções do exercício, onde a “potência” resultou em 39,88.

Exercício E [0%]: escolha um nome legal para o seu esquema!

R: CCC (Controle de Congestionamento by Carol)

Referências:

https://web.mit.edu/copa/

https://github.com/venkatarun95/genericCC

http://alfalfa.mit.edu/#code

https://github.com/keithw/sprout

e demais artigos postados na página da disciplina.