PEDRO MANTOVANI ANTUNES

Iniciação Tecnológica / 2013 – 2014

Implementação e avaliação de protocolo de transporte multicaminho

Eduardo Parente Ribeiro / Departamento de Engenharia Elétrica

Seleção Inteligente de Rotas por Sistemas Finais em Redes IP / 2005016733

Curitiba

2014

PEDRO MANTOVANI ANTUNES

Iniciação Tecnológica / 2013 – 2014

Implementação e avaliação de protocolo de transporte multicaminho

Relatório apresentado à Coordenadoria de Iniciação Científica e Integração Acadêmica da Universidade Federal do Paraná por ocasião da conclusão das atividades de Iniciação Científica – Edital 2013-2014

Eduardo Parente Ribeiro / Departamento de Engenharia Elétrica

Seleção Inteligente de Rotas por Sistemas Finais em Redes IP / 2005016733

Curitiba

2014

RESUMO

Com o avanço da *Internet* e a evolução de diversos modos de comunicação sem fio, aumentou-se o interesse pela comunicação multicaminho, que pode potencialmente melhorar a qualidade e a experiência do usuário final com a *Internet.* Neste âmbito, busca-se avaliar a qualidade da transmissão em certos cenários, principalmente na comunicação *VoIP (Voice Over IP)* e transferência de vídeos, que utilizam o protocolo *UDP (User Datagram Protocolo)* e, alternativamente, o *SCTP (Stream Control Transmission Protocol).* Este último protocolo, não possui padronizado seleção inteligente de caminho, ainda que tenha suporte ao recurso *multihoming*. O objetivo deste trabalho é avaliar vários algoritmos de seleção de caminho para futura implementação no protocolo *SCTP.* Um problema conhecido no SCTP é a instabilidade do algoritmo de seleção de caminho em um sistema com múltiplos agentes comunicativos, provocando altas latências nos enlaces. A simulação de múltiplas transmissões multi-caminho torna possível a avaliação das alternativas de seleção inteligente de caminho, como a comparação de *SRTT (Smooth Round Trip Time)*, perdas de pacotes e tendência de mudança de latência. Foi notado que a alteração de certos parâmetros em um mesmo algoritmo de seleção pode ocasionar melhoras ou pioras na estabilidade do sistema, ou seja, na capacidade de ele se acomodar em um caminho. Foi destacado como um parâmetro fundamental a histerese de troca, que conforme ajustada, pode representar melhoras significativas para a latência e consequentemente, a experiência final do usuário em um cenário de *VoIP*.

Palavras-chave: SCTP, *multihoming*, *reactive delay-centric*, *predictive delay-centric.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE EQUAÇÕES

1. OBJETIVOS

Na *Internet*, um dos cenários mais difíceis de obter uma boa qualidade de transmissão é a comunicação *VoIP* e *streaming* de vídeos. Isto acontece porque nestes casos a latência, o *jitter* e a perda de pacotes tem mais influência sobre a experiência final do usuário. O objetivo deste trabalho está em diminuir o máximo possível a latência (e consequentemente o *jitter*) de múltiplas aplicações disputando a mesma banda disponível em uma topologia multi-abrigada. Como será visto neste trabalho, o alto atraso nos pacotes deve-se à trocas excessivas de caminho, criando uma instabilidade. Por consequência, pode-se definir como objetivo deste trabalho o aumento da estabilidade do sistema, usufruindo de recursos disponíveis na camada 4 do modelo OSI.

2. INTRODUÇÃO

Quando inventado, o protocolo SCTP procurou suprir algumas ausências deixadas pelos protocolos TCP (*Transmission Control Protocol*) e UDP, como o recurso *multihoming*. Mesmo tendo este recurso como nativo, o SCTP padronizado pela IETF *(Internet Enginnering Task Force*) (RFC 4960) não utiliza os sistemas multi-abrigados de forma eficiente, uma vez que as trocas de caminhos só ocorrem quando há uma falha total de comunicação (perda de pacotes) no caminho primário. Desde então, diversos algoritmos vem sendo desenvolvidos pela comunidade científica de forma a usufruir de forma mais vantajosa o recurso *multihoming*, usando-o para diminuir latências de forma transparente ao usuário final.

Atualmente, o algoritmo mais difundido para a seleção de caminhos é através da comparação de SRTT's (também chamado de *reactive delay-centric*). Este é um método prático e eficiente que melhora o desempenho significantemente caso um dos caminhos esteja muito degradado. Entretanto, últimos trabalhos mostram que quando há concorrência de aplicações (daqui em diante denominadas de agentes) sob o efeito deste algoritmo, pode surgir uma instabilidade no sistema que ocasiona elevadas latências. Existem diversos métodos de combater essa instabilidade, alterando (ou adicionando) parâmetros no algoritmo de seleção de caminho.

3. REVISÃO BIBLIOGRÁFICA

3.1. Protocolo SCTP

O protocolo SCTP foi publicado pela IETF no início dos anos 2000 com o intuito de substituir futuramente o protocolo TCP. Este último tem se tornado muito limitador para as aplicações modernas, e muitos tem utilizado protocolos proprietários sobre o UDP a fim de obter uma transferência de dados mais confiável (STEWART *et al.*, 2000). O SCTP foi criado para fornecer diversas funções que são essenciais para o transporte de dados VoIP e ainda podem melhorar o desempenho e confiabilidade de outros tipos de aplicações (ONG e YOAKUN, 2002).

Entre os novos recursos disponíveis no SCTP, os mais importantes para a linha de comunicação sobre IP seriam o *multihoming* e o *multistreaming*. O suporte nativo ao *multihoming* provê à aplicação o uso de todas as interfaces físicas disponíveis para redundância de caminhos, diminuindo as interrupções no serviço de transmissão de dados (STEWART e XIE, 2001). O recurso *multistreaming* permite a divisão do fluxo de dados principal em diversos subfluxos, garantindo controle de ordem de pacotes individualmente para cada subfluxo. Desta forma, perdas de pacotes em apenas um dos subfluxos não bloqueiam o envio dos outros (STEWART e XIE, 2001).

3.2. Seleção inteligente de caminho

O sistema *multihoming* atualmente suportado pela IETF propõe trocas de caminho apenas quando o caminho primário apresenta muitas perdas de pacotes consecutivas. Isto não é aceitável para comunicações VoIP ou *streaming* de vídeos, uma vez que a detecção de falha de caminho demora no mínimo 15 segundos para ocorrer (KELLY *et al, 2004*). O primeiro algoritmo proposto para corrigir este problema foi o *reactive delay-centric*, desenvolvido por KELLY *et al* (*2004*). Este método escolhe o caminho primário baseado nos valores atuais do SRTT. Assim, quando a latência do caminho primário é maior que a de um caminho secundário, a troca de caminho é realizada, provendo melhor qualidade de serviço para o usuário final.

3.3. *Predictive delay-centric*

Um dos outros métodos de seleção de caminho criados para a melhora de qualidade para o usuário foi o *predictive delay-centric*. Este algoritmo usa o método MACD (*Moving Average Convergence / Divergence)* para a seleção de caminho. O MACD é muito utilizado no mercado financeiro para previsões de valores de ações. No caso da *Internet* pode ser usado para prever latências, calculando dois valores de SRTT, um de curto prazo e outro de longo prazo. Este método apresentou melhoras de qualidade em diversos testes de *streaming* de vídeos realizados (TORRES, 2014).

3.4. Concorrência de Agentes

A concorrência de múltiplas aplicações pelos mesmos enlaces provou-se problemática para os algoritmos de seleção de caminho (GAVRILOFF, 2009). Isto acontece porque os SRTTs dos múltiplos agentes são muito parecidos, e todas as aplicações tendem a seguir por um mesmo caminho. Isto pode ocasionar uma instabilidade, fazendo com que todos os agentes comuniquem-se pelo mesmo caminho em um mesmo instante de tempo. Esta condição agrava-se quando a banda disputada pelos agentes é muito próxima da banda total disponível. Diversos parâmetros podem ser alterados no protocolo SCTP ou no algoritmo de seleção de caminho para diminuir essa instabilidade, sendo que este é o objeto de estudo deste trabalho.

4. MATERIAIS E MÉTODOS

4.1. Computadores e Topologia de Rede

Como o princípio deste trabalho é sempre disponibilizar os recursos para *softwares* livres, todas as simulações e testes foram executados em sistema operacional Linux, sendo que os primeiros testes foram realizados na distribuição Ubuntu e posteriormente na

distribuição Debian. Contudo, os resultados deste trabalho são válidos para qualquer distribuição Linux até o *kernel* 3.11.0-12-generic.

A topologia de rede exigida foi muito simples; dois computadores com duas placas de rede *Ethernet* cada. Uma das placas de cada computador foi ligada diretamente à outra, através de um cabo *crossover*. A outra placa de cada máquina está ligada na *intranet* do bloco de Engenharia Elétrica da Universidade Federal do Paraná, e o enlace passa por *switches* de baixa latência. Desta forma, cada enlace torna-se independente, e um não consegue comunicar-se com o outro.

4.2. Validação da Restrição de Banda e Estrutura de Filas

Para não ser necessário trabalhar com envios a altíssimas velocidades, procurou-se restringir a banda de envio com o *software tc* (*traffic control)*, recurso nativo do Linux. Como não se conhecia o funcionamento do *software*, foi necessário fazer testes de validação para checar se ele era operado conforme o esperado. O primeiro teste consistiu em enviar pacotes CBR (*Constant Bit Rate)* em diversas taxas diferentes e conferir se o atraso médio e a perda de pacotes estavam próximos ao calculado.

O comportamento do atraso médio deve ser mínimo enquanto o enlace não é saturado, e máximo quando este é saturado. O valor máximo do atraso médio varia conforme ajustamos o tamanho da fila no tc. Por outro lado, a porcentagem de perda de pacotes é 0% até a saturação e depois vai aumentando conforme a Equação 1.

$$\%\ Perdas = 1 - \frac{Limite\ Real\ do\ Enlace}{Taxa\ Enviada}$$

Equação 1: Cálculo da Porcentagem de Perdas de Pacote

Nas medições de saturação e perda de pacote, a taxa de saturação é um pouco abaixo da teórica, pois no *software* utilizado para as medidas (iperf), os cabeçalhos não são contabilizados. Desta forma, em um enlace de 100 Mbps, a saturação ocorre perto de 95,7 Mbps. Os Gráficos 1 e 2 são os valores obtidos de atraso médio e perda de pacotes para diferentes restrições de banda no tbf, 1Mbps e 100kbps.

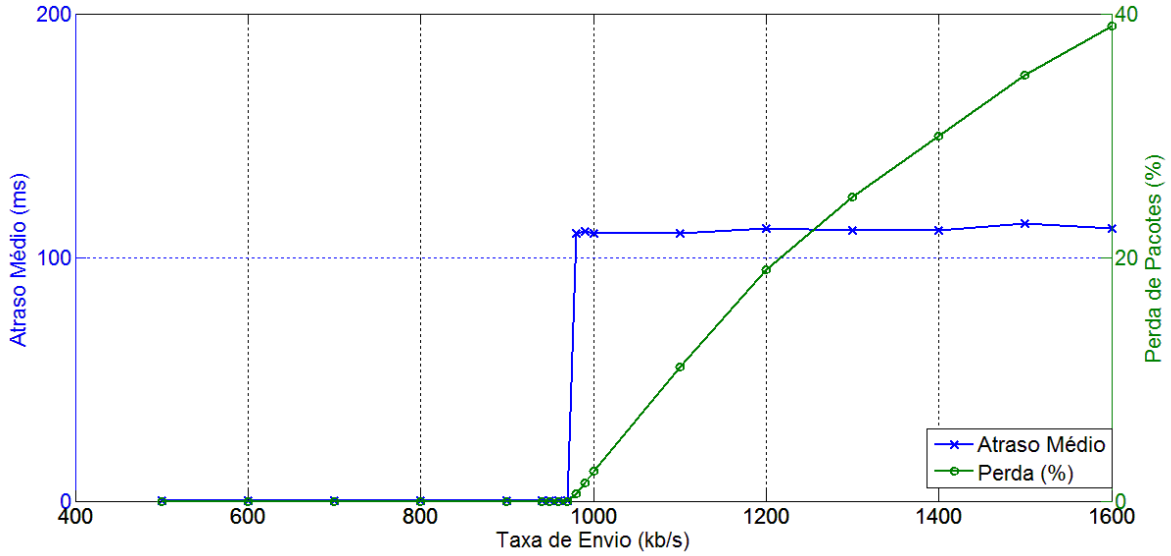

Gráfico 1: Atraso Médio e Perda de Pacotes em função da Taxa de Envio (1Mbps)

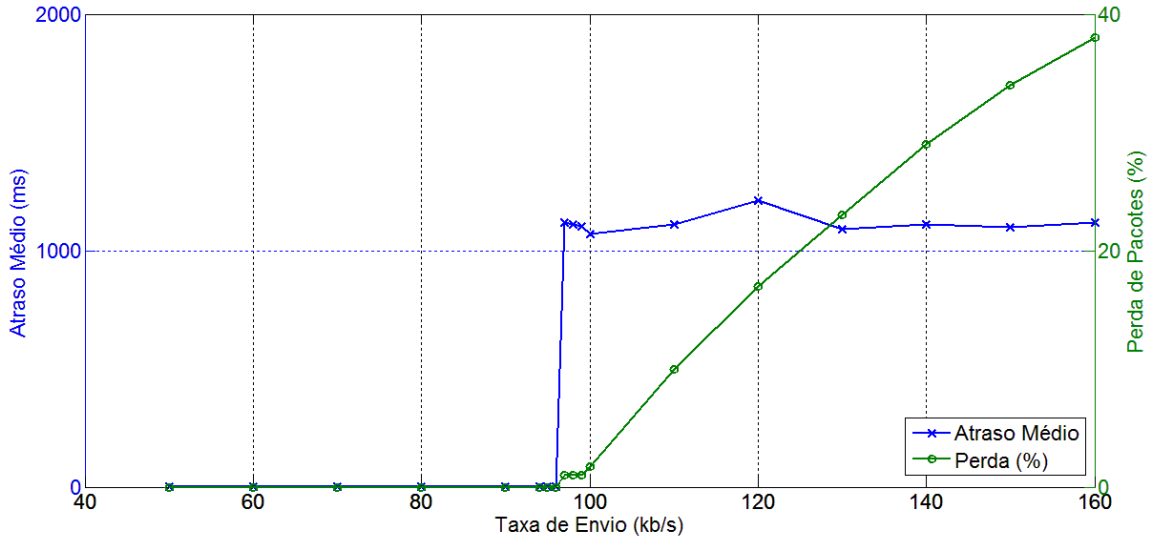

Gráfico 2: Atraso Médio e Perda de Pacotes em função da Taxa de Envio (100kbps)

Assim, percebe-se que independente da largura de banda limitada, o comportamento foi sempre o mesmo, com o atraso subindo somente após a saturação da banda. A Tabela 1 compara as perdas de pacotes medidas e calculadas para o caso em 100Mbps.

Taxa de Envio	Calculado	Medido
90 Mbps	0.00%	0.0%
100 Mbps	4.30%	4.7%
110 Mbps	13.00%	14.0%
120 Mbps	20.25%	20.0%
130 Mbps	26.38%	27.0%

140 Mbps	31.64%	32.0%
150 Mbps	36.20%	36.0%
160 Mbps	40.19%	41.0%
170 Mbps	43.71%	44.0%
180 Mbps	46.83%	47.0%
190 Mbps	49.63%	50.0%
200 Mbps	52.10%	53.0%
210 Mbps	54.40%	54.0%
220 Mbps	56.50%	57.0%

Tabela 1: Diferenças entre as Perdas de Pacotes medida e calculada

O último teste de validação refere-se ao tamanho da fila máximo suportado pelo tbf. O Gráfico 3 compara os valores calculados e medidos dos atrasos com o enlace saturado para diferentes valores de fila. O objetivo é verificar até qual tamanho de fila é possível trabalhar sem haver distorção de valores. Como o tamanho de pacote trabalhado é de 1470 Bytes, observamos na figura que o tamanho de fila máximo é perto de 105,8 kB ou 846,7 kb.

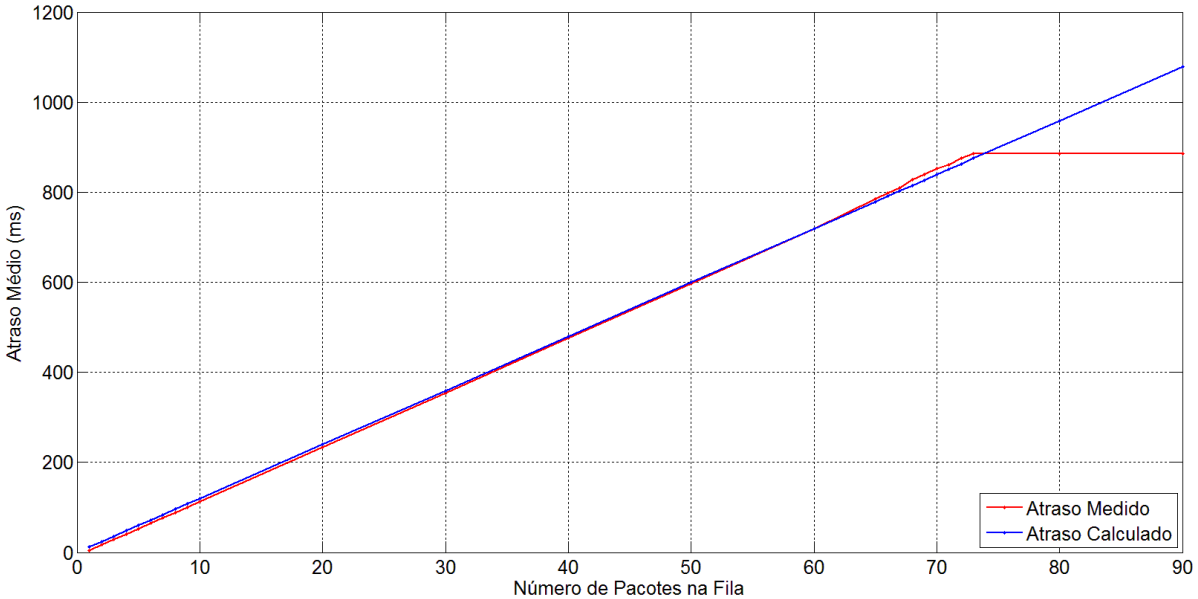

Gráfico 3: Atraso Médio em Função do Número de Pacotes de 1470 Bytes na Fila

Verificada a funcionalidade do *tc*, todos os testes foram realizados com a seguinte configuração no *tc*:

tc qdisc add dev eth0 root tbf rate 500kbit latency 1.9s burst 1300B

tc qdisc add dev eth0 root tbf rate 500kbit latency 1.9s burst 1300B

A banda foi limitada para 500kbps, mas poderia ser outro valor qualquer, desde que os agentes enviassem a banda correspondente para a mesma utilização de banda total. O valor de *burst* não influencia no resultado para os testes aqui feitos, já que estes estão em um regime CBR. O valor de *latency* é outra forma de representar o valor máximo do *buffer* de saída, sendo que quanto maior o valor de *latency* maior a fila (ou *buffer*) de saída.

4.3 Cliente e Servidor UDP

Para obter-se maior facilidade em manipular os parâmetros do SCTP, foram desenvolvidos em linguagem C um cliente e um servidor UDP que simulem o exato comportamento do SCTP *multihoming*, comunicando-se pelos dois caminhos. O cliente UDP comunica-se através de um regime CBR com pacotes de 250B, sendo que a banda enviada por ele é estipulada conforme a utilização de banda (ρ) requerida. O algoritmo de seleção de caminho foi feito na camada de aplicação, porém o seu comportamento é muito similar ao algoritmo na camada de transporte. Os envios de *Heartbeat* nos primeiros testes eram feitos a cada 5 pacotes de dados. Como esta seria uma taxa de envio muito elevada para *Heartbeat's,* foi implementado o envio temporal nos testes seguintes. O momento em que este novo método de envio de HB's foi implementado está especificado nos resultados dos testes. O servidor UDP apenas recebe os pacotes e envia um pacote de confirmação de recebimento no menor tempo possível. Estes pacotes também servem como referência para o cálculo do SRTT. A fim de aumentar a aleatoriedade e dispersão dos pacotes, também foi definido um intervalo aleatório de 0 a 10s entre a inicialização dos 6 clientes UDP.

5. RESULTADOS

5.1. Verificação da instabilidade

Para demonstrar o efeito da instabilidade e seu comportamento, foram desenhados dois gráficos mostrando o caminho ativo de cada agente durante o tempo de execução, um deles mostrando um caso de estabilidade, onde os agentes se distribuem igualmente pelos caminhos disponíveis e se acomodam nestes, e um caso de instabilidade, onde todos os

agentes comunicam-se ao mesmo tempo pelo mesmo caminho, e trocam de caminho quase simultaneamente, gerando altas latências.

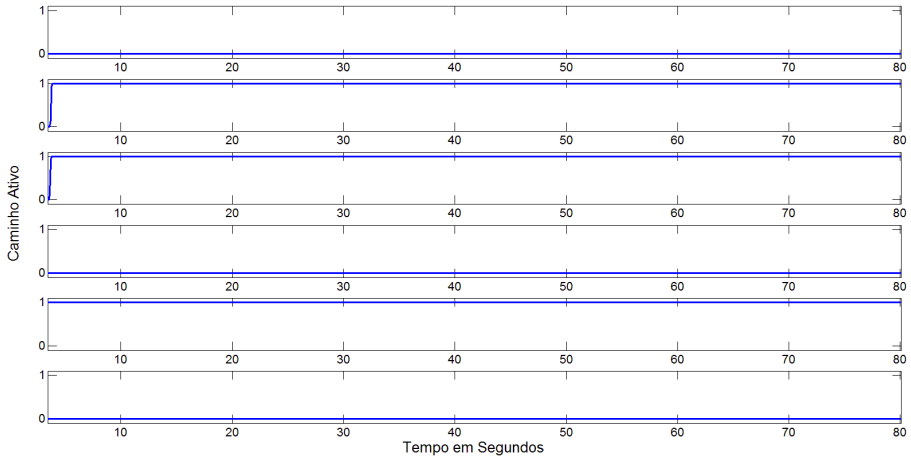

Gráfico 4: Caso de estabilidade entre agentes

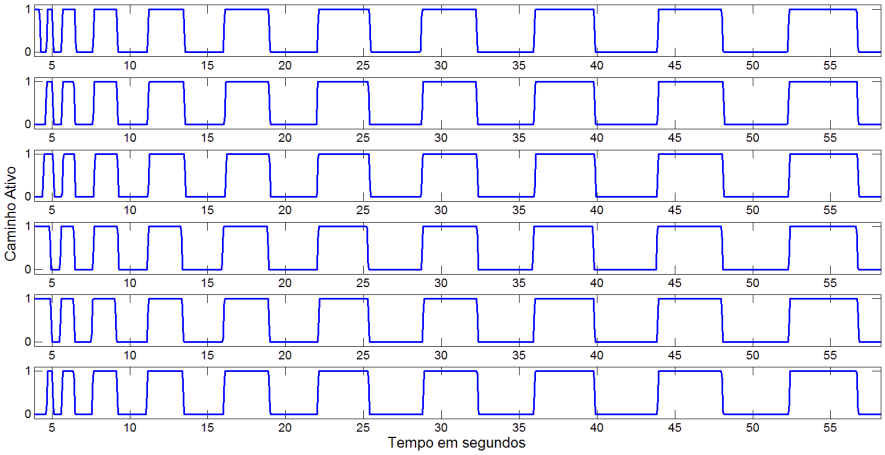

Gráfico 5: Caso de instabilidade entre agentes

5.2. Troca de caminho por *delay-centric*

Foram realizadas simulações para diversas utilizações de banda diferentes (ρ), por esse ser um fator importantíssimo na condição de estabilidade do sistema. Foram utilizados valores de ρ acima de 0,72 somente, pois para valores inferiores o sistema estabiliza em quase 100% dos casos. Todos os testes foram realizados 10 vezes, a fim de obter uma tendência para cada faixa de utilização de banda.

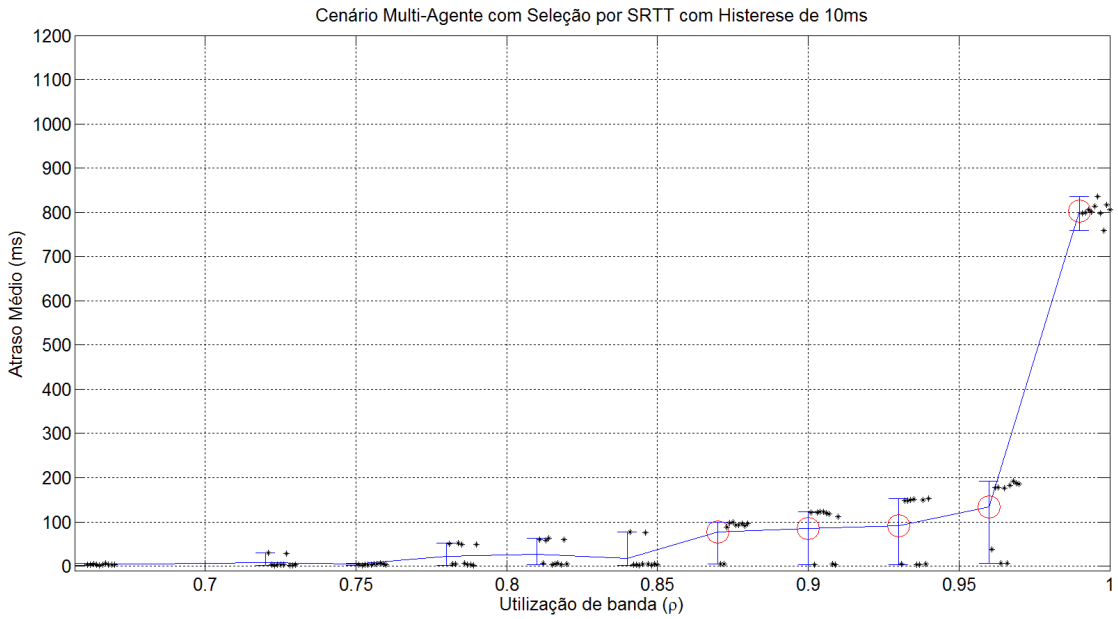

Gráfico 6: Atraso Médio em função de ρ (Histerese 10ms)

Observa-se para cada valor de ρ um comportamento bimodal, ou seja, ou o atraso é mínimo (valores menor que 1ms) ou é máximo (valor que varia conforme o valor de ρ). Os círculos vermelhos aparecem quando o número de vezes em que o atraso foi máximo aparece mais que o número de vezes com o atraso mínimo. O primeiro parâmetro a ser alterado foi o da histerese, testando-se então com 30ms e 60ms.

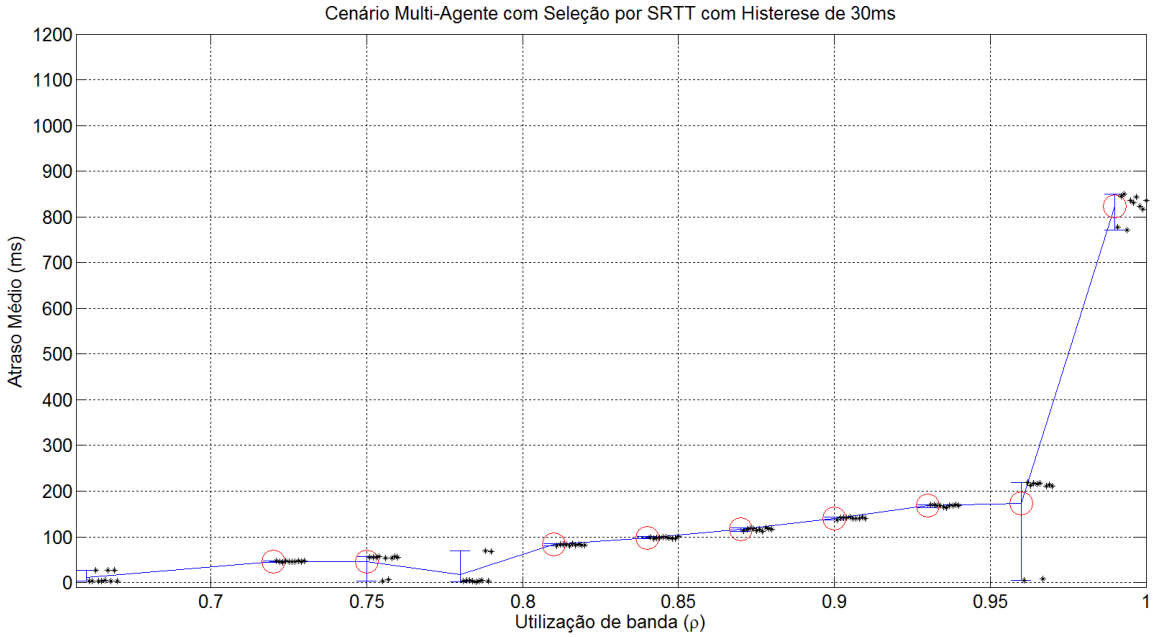

Gráfico 7: Atraso Médio em função de ρ (Histerese 30ms)

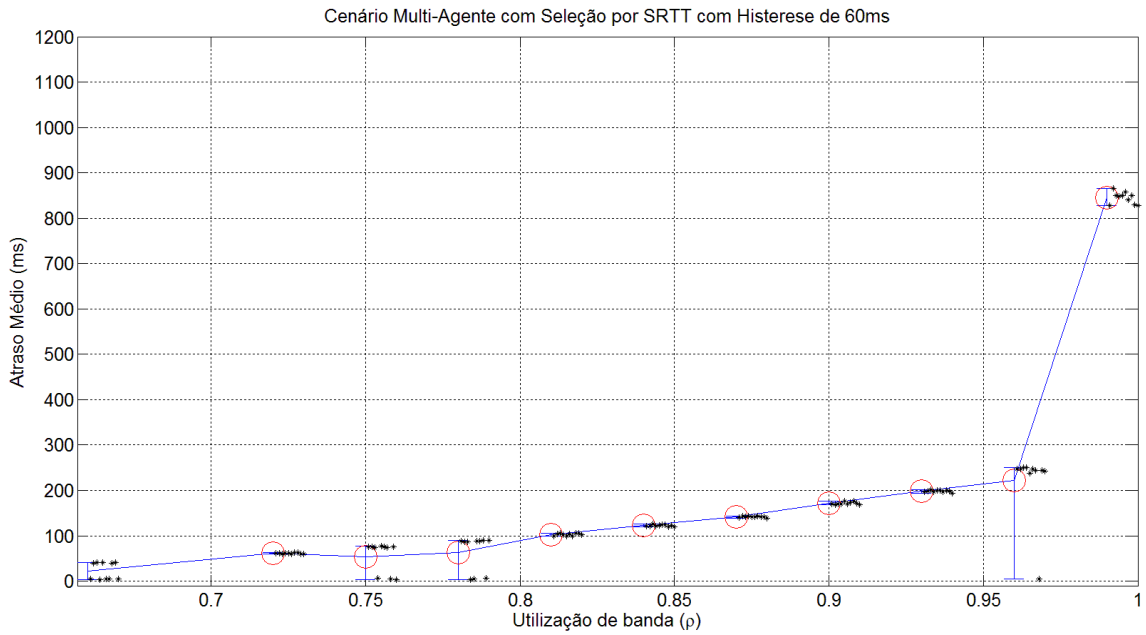

Gráfico 8: Atraso Médio em função de ρ (Histerese 60ms)

Comparando-se os gráficos, o cenário com histerese de 10ms foi o que se mostrou mais eficiente, com maior estabilização e menor atraso médio.

Do Gráfico 8 para o Gráfico 9, houve uma mudança na forma do envio de HB's. Anteriormente, era enviado um Heartbeat a cada 5 pacotes de dados e o *Heartbeat* era levado em conta no cálculo da banda a ser enviada. A partir do Gráfico 9, o *Heartbeat* passou a ser temporal, com intervalo médio entre eles de 1s na maioria dos casos.

Esta mudança na forma de envio de HB's é responsável pelo aumento da variabilidade do atraso médio no último caso. Isto acontece pelo fato de que os *Heartbeats* neste teste foram muito menos frequentes do que nos testes anteriores, tornando o algoritmo menos suscetível às mudanças de caminho, portanto demorando a detectar a eficiência dos caminhos, aumentando o atraso médio.

Apesar de esta medida aumentar o atraso máximo das simulações, ela foi necessária, pois representa um comportamento mais factível, sendo que os envios de HB não são desenvolvidos para utilizarem uma taxa de envio muito alta.

No próximo teste, o intervalo de *Heartbeat* foi configurado inicialmente para um valor aleatório uniforme, entre 0.5s e 1.5s. Este valor foi sorteado apenas uma vez na inicialização do agente, portanto todos os *Heartbeats* são espaçados deste valor. O Gráfico 9 representa o experimento. Como os resultados anteriores foram mais vantajosos com a histerese de 10ms e pacotes de 250 bytes, estes parâmetros foram repetidos.

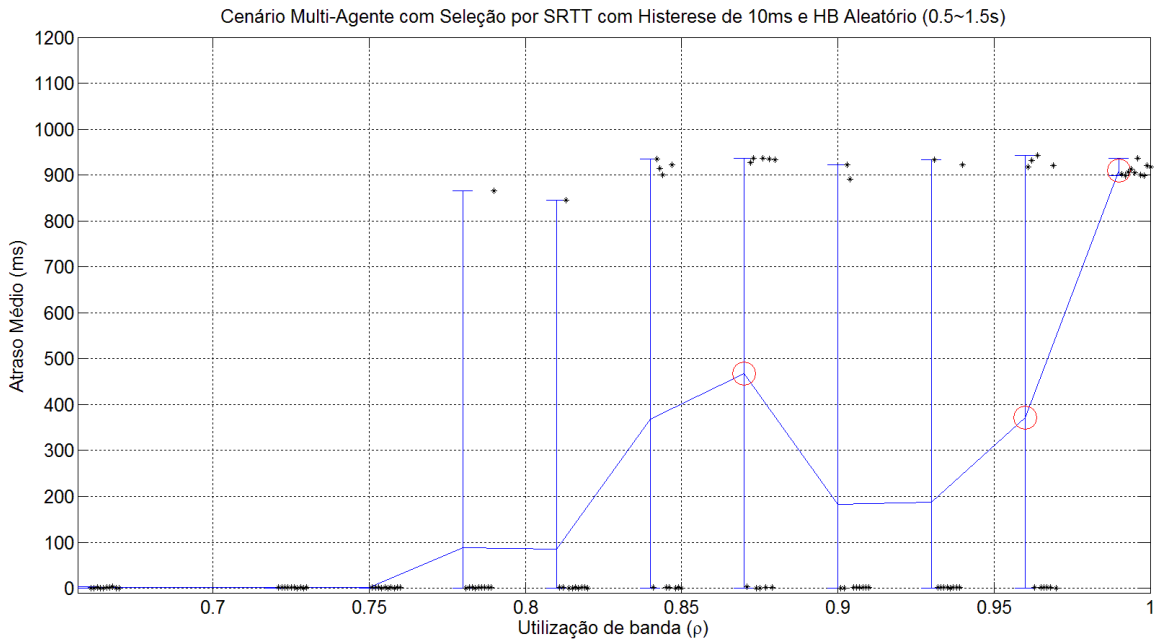

Gráfico 9: Atraso Médio em função de ρ com HB Aleatório

Os parâmetros alterados e testados nas simulações seguintes foram:

- Alteração do HB Aleatório inicial para aleatório a cada pacote.
- Alteração do intervalo de aleatoriedade do HB para 2.5s a 3.5s.
- Alteração do intervalo de aleatoriedade do HB para 0.5s a 3.5s.
- Implementação do algoritmo de tempo de guarda, conforme proposto por Gavriloff (2009), com aleatoriedade entre 0.5s e 1.5s.
- Implementação do algoritmo de tempo de guarda fixo em 1s.

O tempo de guarda é um complemento ao algoritmo de seleção por SRTT. Ao avaliar a latência menor de um caminho secundário, ele aguarda um tempo específico. Depois de passado esse tempo, os SRTT's são verificados novamente, e se o caminho secundário continuar com menor latência, só aí a troca de caminhos é realizada.

Como todos os testes tiveram o mesmo comportamento bimodal com atrasos máximos próximos de 1000 ms, a comparação entre eles foi apresentada na Tabela 2 e 3, onde o número mostrado representa o número de vezes que a simulação se estabilizou, entre um total de 10 testes.

	Histerese 60ms	Histerese 30ms	Histerese 10ms	HB Inicial 0.5~1.5s	HB 0.5~1.5s
ρ = 0.72	0	0	8	10	10
0.75	3	1	10	10	10
0.78	3	8	6	9	9
0.81	0	0	6	9	7
0.84	0	0	8	6	6
0.87	0	0	2	5	7
0.90	0	0	3	8	5
0.93	0	0	4	8	8
0.96	1	2	2	6	7
0.99	0	0	0	0	0

Tabela 2: Comparação entre os parâmetros

	HB 2.5~3.5s	HB 0.5~3.5s	HB 0.5~1.5sTG 0.5~1.5s	HB 0.5~1.5s TG 1s	HB 0.5~1.5s S/ Histerese
ρ = 0.72	10	10	9	5	10
0.75	10	10	7	3	10
0.78	10	10	4	8	9
0.81	7	6	7	7	7
0.84	5	6	1	4	6
0.87	5	6	3	5	7
0.90	7	7	5	3	5
0.93	7	5	5	2	8
0.96	6	7	2	3	7
0.99	0	0	0	0	0

Tabela 3: Continuação da Tabela 2

Analisando as duas tabelas, os parâmetros que obtiveram melhores desempenhos foram o HB Aleatório Inicial entre 0.5s e 1.5s com Histerese de 10ms, o HB Aleatório a cada pacote entre 0.5s e 1.5s com Histerese de 10ms, e o HB Aleatório a cada pacote entre 0.5s e 1.5s sem Histerese. Como o resultado desses 3 testes foram muito parecidos, foi adotado o mais conservador como o de melhor desempenho: HB Aleatório a cada pacote com Histerese de 10ms.

A seguir, foi realizado o experimento com o número de agentes variando entre 6, 12, 24, 48 e 96, a fim de mostrar alguma correlação entre estabilidade e número de agentes. Para conservar os mesmos valores de ρ, manteve-se como constante os valores de limitação de banda a 500 kbps, e variou-se a banda enviada por cada agente.

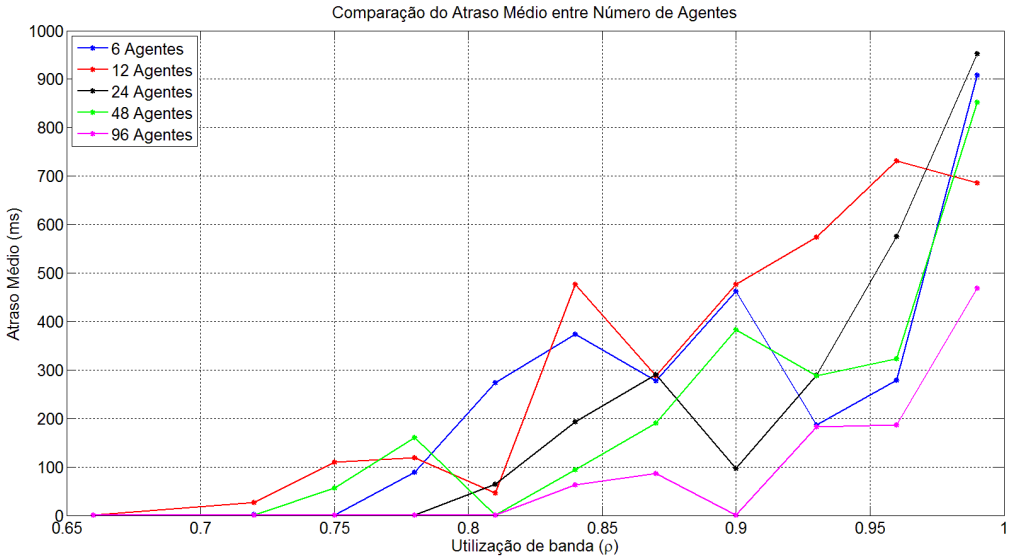

Gráfico 10: Comparação entre o Atraso Médio de Múltiplos Agentes

Gráfico 10 mostra o atraso médio para cada número de agentes em função de ρ. Pela aleatoriedade dos dados mostrados, percebe-se que na maioria dos casos não há muita correlação entre o número de agentes e a estabilidade do sistema. Porém, no último caso, com 96 agentes simultâneos, é possível perceber uma queda considerável no atraso, indicando que os agentes conseguem se acomodar com mais facilidade pelos caminhos conforme aumentamos o número de agentes. Isto pode ser representado como a granularidade do sistema, como a granularidade de grãos de areia, por exemplo. Quanto menor a granularidade (ou maior número de agentes), mais fácil os grãos de areia se distribuem em um lugar. Entretanto, no sistema em testes, isto não vale em 100% dos casos, e as melhoras reais só começam a aparecer de verdade para um número muito alto de agentes, que raramente é alcançado em um caso prático.

Como os resultados de quase todos os testes não apresentaram um comportamento monótono conforme o valor da utilização de banda, procurou-se aumentar o número de simulações. Por se tratar de um comportamento bimodal, os resultados poderiam estar sendo obtidos através de um baixo intervalo de confiança. Como não se sabe a distribuição estatística dos experimentos, não é possível definir a média e o desvio padrão para calcular com qual intervalo de confiança os testes estão sendo feitos. A alternativa foi realizar um número de testes maior, e verificar se a média dos resultados obtidos para 100 testes, por exemplo, é muito diferente para 10 testes.

O Gráfico 11 nos mostra que, independente do número de simulações, a média oscila muito perto dos valores já obtidos para 10 simulações, ou seja, o intervalo de

confiança está razoável da forma que estava sendo feito anteriormente. É visível que para 100 testes, a curva parece mais suavizada, porém os resultados não divergem muito.

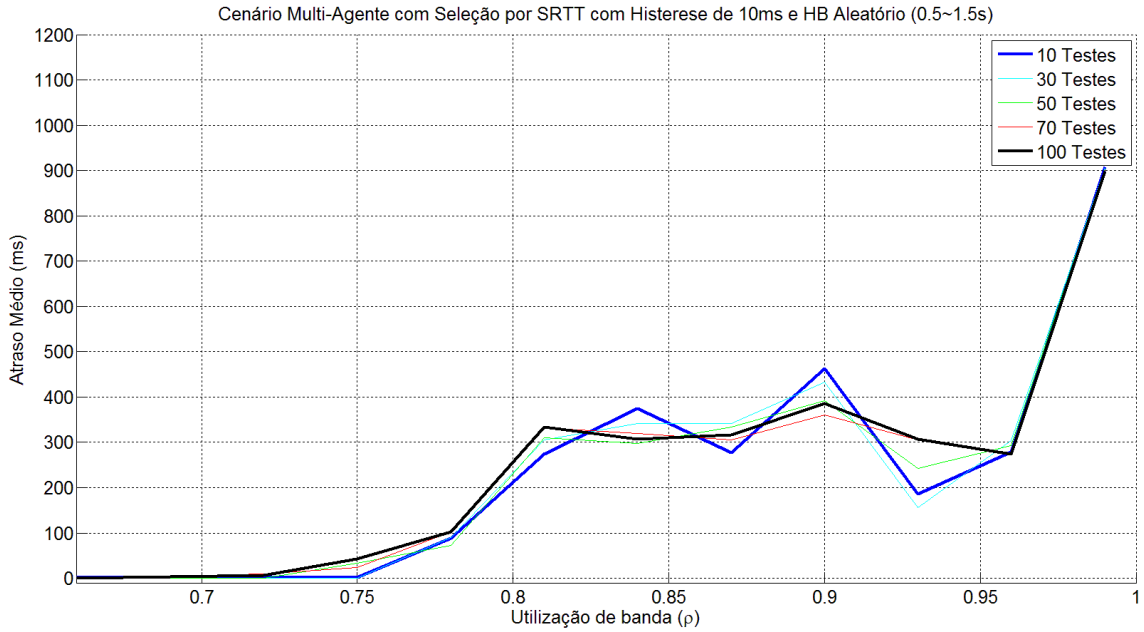

Gráfico 11: Gráfico para diferentes Intervalos de Confiança

5.3. Troca de caminho por *predictive delay-centric*

Mais um método de seleção de caminho recentemente criado e testado é o *predictive delay-centric*, que utiliza-se do indicador MACD (*Moving Average Convergence / Divergence*). Este indicador é amplamente utilizado na área de compra e venda de ações, por retratar uma possível tendência dos valores. No caso da comunicação multi-abrigada, o valor representa a latência ou o atraso do caminho.

Do cálculo do SRTT, temos que:

$$\boldsymbol{SRTT_n = (1 - \alpha) SRTT_{n-1} + \alpha RTT_i}$$

Equação 2: Fórmula para cálculo do SRTT

A fim de calcular a tendência do atraso do caminho, são calculados dois SRTT's, um de longo prazo e outro de curto prazo, aqui denominados de $SRTT_S$ (*short*) e $SRTT_L$ (*long*). O $SRTT_S$ utiliza um valor de α igual a 0.667, e o $SRTT_L$ um valor para α de 0.154, conforme sugerido por Torres (2014).

Com os valores calculados, a tendência de aumento da latência acontece caso a média de curto prazo ultrapasse a média de longo prazo. Segundo o algoritmo desenvolvido por Torres (2014), a troca de caminho acontece quando as seguintes condições se tornam verdadeiras:

- O $SRTT_S$ do caminho secundário seja menor que o do caminho primário.
- O $SRTT_S$ do caminho primário seja maior que o $SRTT_L$ do caminho primário (tendência de aumento da latência).
- O $SRTT_S$ do caminho primário ultrapasse um limiar, definido inicialmente como 150ms.

Seguindo este algoritmo, foi realizado novamente o teste multi-agente, com os resultados exibidos pelo Gráfico 12. Novamente, o HB foi feito de forma temporal e aleatória entre 0.5s e 1.5s. Vale notar que por se trabalhar agora com um limiar de troca muito grande e por trabalhar com tendências, o comportamento bimodal muitas vezes deixou de aparecer.

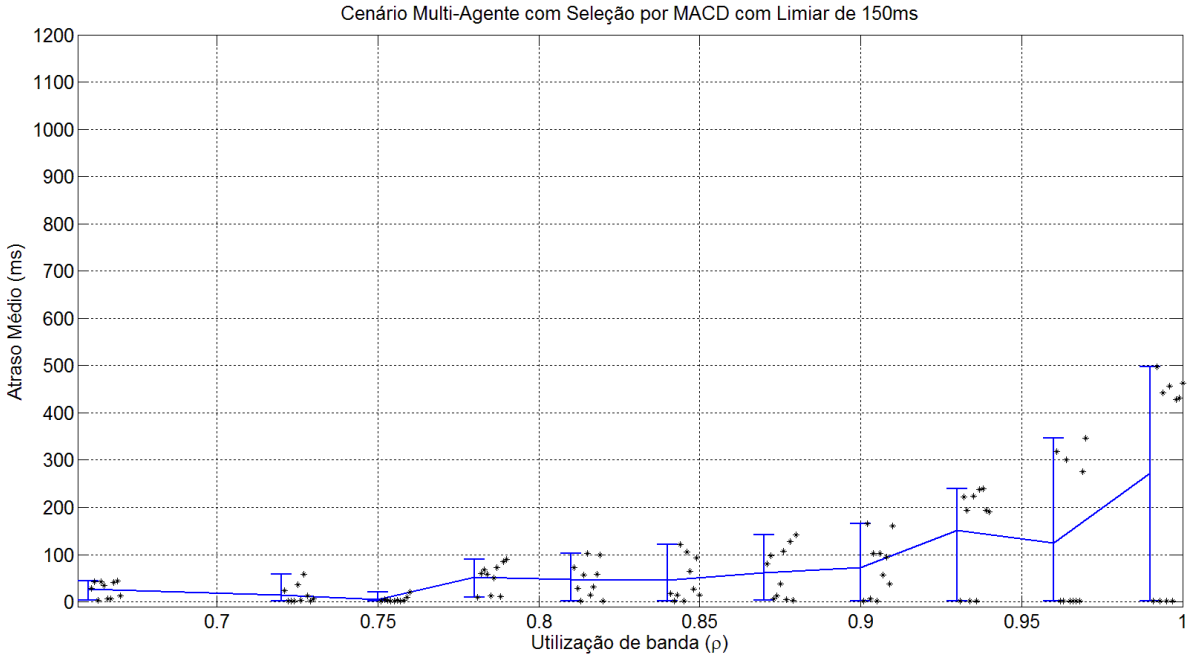

Gráfico 12: *Predictive delay-centric* com limiar de 150ms

De qualquer forma, é visível a diminuição do atraso médio em relação à outros testes, mesmo que muitas vezes ele não se mantenha abaixo dos milissegundos. O último teste realizado foi feito sem o limiar de troca, de forma a avaliar o impacto deste no atraso médio. O resultado está apresentado no Gráfico 13.

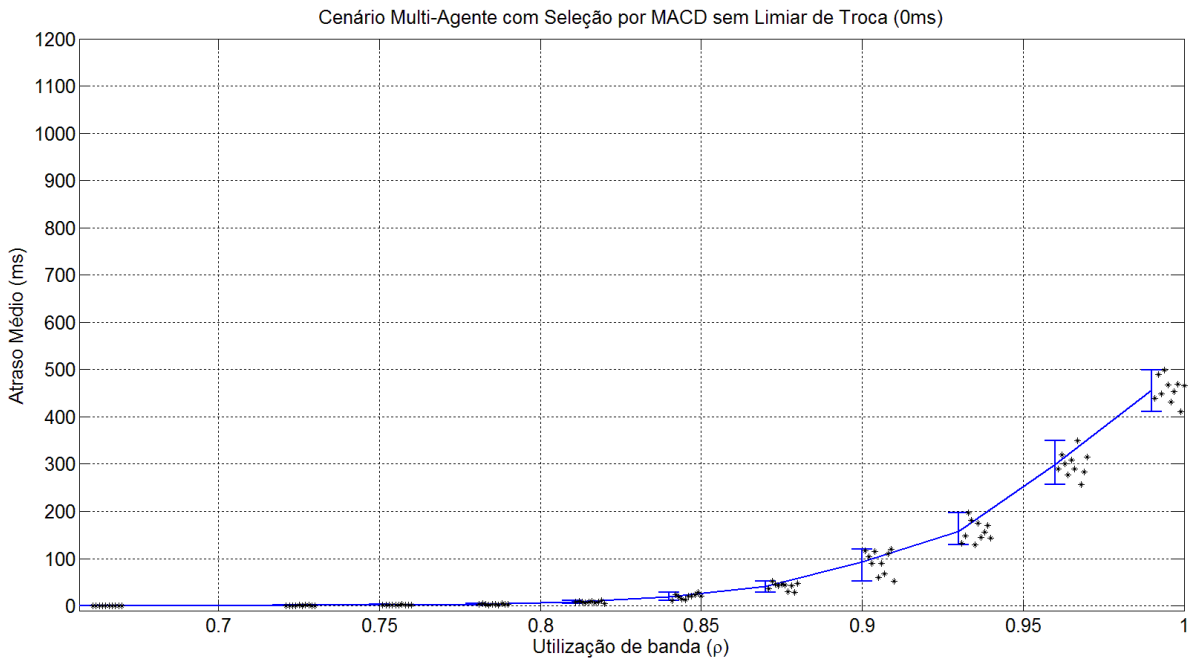

Gráfico 13: *Predictive delay-centric* sem limiar de troca

Neste último caso, percebe-se a ausência total do comportamento bimodal, sendo que agora todos os pontos concentram-se em torno de uma média. Isto não é desejável, já que desta forma o sistema nunca irá obter latências baixas para um ρ muito perto de 1.

5.4 Diferença entre tendências

Um espaço em branco deixado pelo algoritmo do *predictive delay-centric* foi o fato de que ele analisa somente a tendência do caminho primário. Isto pode ocasionar uma troca indesejada de caminho, se o caminho primário tiver uma tendência de aumento de latência, mas o caminho secundário tiver uma tendência de aumento ainda maior. Desta forma, foi desenvolvido aqui um aprimoramento do algoritmo do *predictive delay-centric.* Agora, são calculadas as tendências do caminho primário e secundário, e a troca só acontece quando as seguintes condições são atingidas:

- O $SRTT_S$ do caminho secundário seja menor que o do caminho primário.
- A tendência de aumento da latência seja maior no caminho secundário do que no caminho primário. (Diferenças entre tendências)
- O $SRTT_S$ do caminho primário ultrapasse um limiar pré-definido.

Foram realizadas simulações com o limiar de troca de 1ms, 10ms, 40ms, 70ms e 150ms.

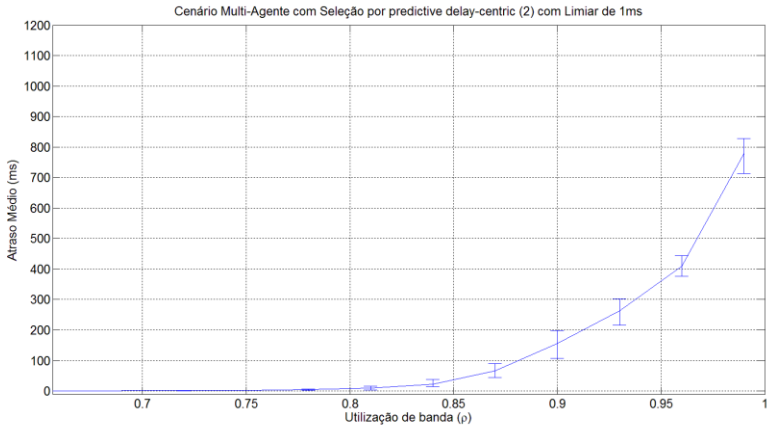

Gráfico 14: Diferença entre tendências com limiar de troca de 1ms

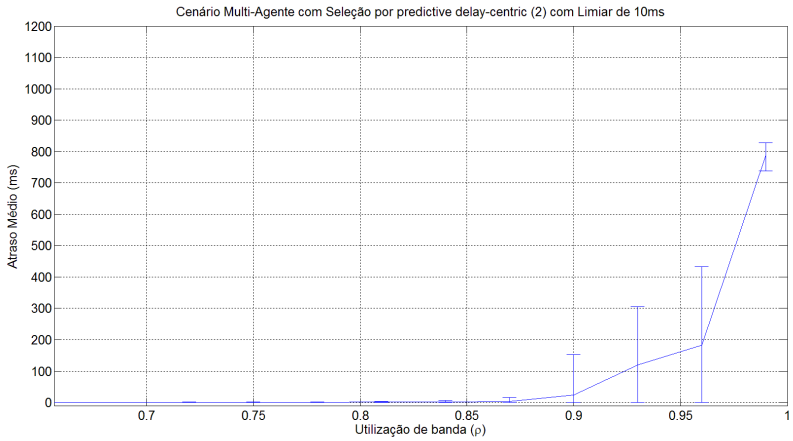

Gráfico 15: Diferença entre tendências com limiar de troca de 10ms

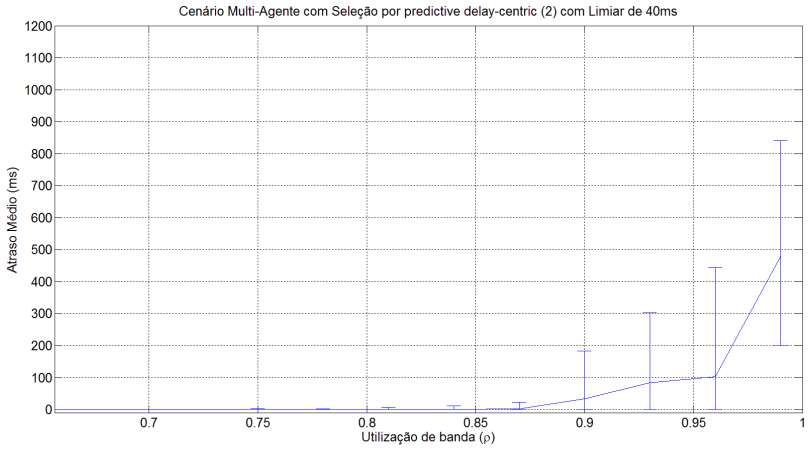

Gráfico 16: Diferença entre tendências com limiar de troca de 40ms

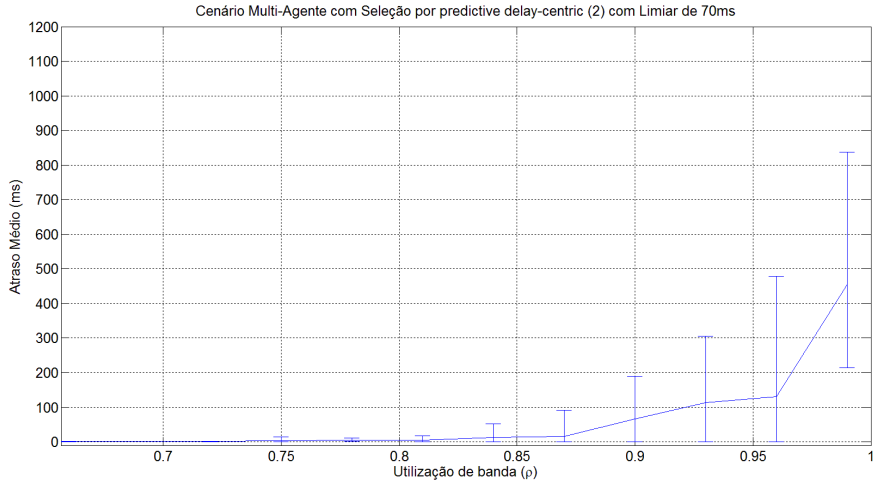

Gráfico 17: Diferença entre tendências com limiar de troca de 70ms

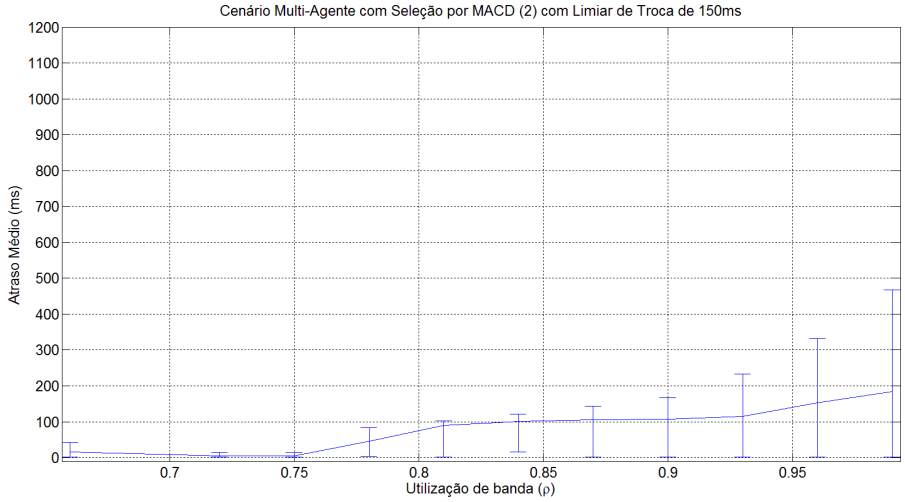

Gráfico 18: Diferença entre tendências com limiar de troca de 150ms

Conforme apresentado pelos Gráficos, o resultado da estabilidade altera-se muito conforme o limiar de troca estipulado. Se este for muito baixo, o sistema não se estabiliza em utilizações de banda elevadas e o comportamento bimodal desaparece, fazendo com que todos os pontos concentrem-se em torno uma média. Para um limiar de troca muito elevado, o sistema consegue estabilizar mais vezes e diminuir o atraso médio para utilizações de banda muito elevadas. Contudo, ele começa a apresentar situações de instabilidade para utilizações de banda inferiores que antes eram totalmente estabilizadas. Assim, procurou-se um limiar de troca em que o sistema obtivesse melhor desempenho tanto em um ρ baixo quanto elevado. Este valor foi obtido empiricamente através dos Gráficos, e chegou-se a um patamar de 40ms como sendo o limiar mais perto do ideal.

6. DISCUSSÃO

Ao longo deste trabalho, foram realizados diversos testes a fim de analisar cada parâmetro individualmente, com o intuito de eliminar o problema da instabilidade em 100% dos casos. Contudo, nada do que foi feito conseguir atingir esse objetivo por completo. Não obstante, foi possível observar as melhorias de acordo com os parâmetros selecionados.

O problema da instabilidade multi-agentes, notado inicialmente por Gavriloff (2009), foi confirmado. Entretanto, a solução do algoritmo de tempo de guarda apresentada pelo mesmo em seu trabalho, não apresentou resultados favoráveis nos testes aqui feitos.

O algoritmo de troca *predictive delay-centric* proposto por Torres (2014) apresentou aparentes melhoras no problema da instabilidade, e pode ser uma das alternativas para no mínimo reduzir o efeito da instabilidade na maior parte dos casos. Foi realizada também uma alteração no algoritmo do *predictive delay-centric*, fazendo com que este compare as tendências dos caminhos primários e secundários a fim de realizar a troca de caminho baseado nestas informações. Este método diminuiu latências consideravelmente no cenário multi-agente, melhorando a estabilidade dos sistemas. Testando diversos valores de limiar de troca diferentes, foi obtido um valor de referência para o caso multi-agente, sendo este de 40ms. O valor proposto anteriormente por Torres (2014) de 150ms foi testado, porém não apresentou resultados muito eficientes para o cenário de multi-agentes.

7. CONCLUSÕES

No cenário atual da *Internet*, mesmo que já existam aparelhos comunicando-se por mais de um ISP (*Internet Service Provider*), é muito comum um deles ser muito mais eficiente do que outro, especialmente no caso dos *smartphones*, onde o WiFi geralmente é melhor que as redes móveis. Nestes casos, o algoritmo de *delay-centric* para a troca de caminho pode ser suficiente, porque as trocas ocorreriam apenas em picos de latência.

O fato é que quando existem diversas aplicações disputando as bandas disponíveis e as instabilidades começam a ocorrer, o *delay-centric* deve estar bem ajustado para minimizar estes efeitos. Como várias tentativas de corrigir por completo esta instabilidade não obtiveram sucesso, é natural que se comecem as buscas por outros métodos de troca

de caminhos que não ocorram esse tipo de instabilidade. O algoritmo por *predictive delay-centric* já apresentou ser uma alternativa muito interessante para substituir o *delay-centric* simples, conforme testes apresentados por Torres (2014). Nos testes multi-agentes também apresentou potencial para diminuir a instabilidade e latência final.

Nos testes por comparação de SRTT, a configuração que apresentou melhores respostas à estabilidade foi: Histerese de troca com valor de 10ms, *Heartbeat* com intervalo aleatório a cada pacote com valores entre 0.5s e 1.5s, seguindo uma distribuição uniforme. A configuração padrão do *predictive delay-centric* proposta por Torres (2014) apresentou bons resultados no cenário multi-agente: Limiar de troca de 150ms, com *Heartbeat* aleatório entre 0.5s e 1.5s.

Ampliando os estudos sobre o *predictive delay-centric*, o novo método testado calculando a diferença entre as tendências da latência apresentou resultados ainda melhores que os obtidos sem este cálculo. Para este caso, o ajuste de parâmetros que apresentou melhores resultados foi com o limiar de troca em 40ms e o *Heartbeat* aleatório entre 0.5s e 1.5s.

Para trabalhos futuros, uma das linhas de trabalho seria ampliar os testes sobre o *predictive delay-centric*, ajustando parâmetros e melhorando ainda mais o seu desempenho. Uma das ideias seria adicionar uma histerese à comparação da tendência ($SRTT_S$ e $SRTT_L$). Por mais que neste trabalho o algoritmo por *delay-centric* não obteve nenhuma perspectiva de melhora, também é possível continuar o trabalho por esta linha, acrescentando outros tipos de método de prevenção de instabilidade, como por exemplo, forçar temporariamente um caminho principal aleatório, verificando se em tal estado o sistema melhore seu atraso médio. Outros métodos de troca de caminho também devem ser avaliados, algoritmos que levem em conta a perda de pacotes e o jitter, por exemplo.

8. REFERÊNCIAS BIBLIOGRÁFICAS

STEWART, R.; XIE, Q.; MORNEAULT, K.; SHARP, C.; SCHWARZBAUER, H.; TAYLOR, T.; RYTINA I.; KALLA, M.; ZHANG, L.; PAXSON, V. **Stream Control Transmission Protocol**. IETF RFC 2960, Outubro de 2000. Disponível em: https://www.ietf.org/rfc/rfc2960.txt Acesso em: 24 jul. 2014.

ONG, L.; YOAKUM, J. **An Introduction to the Stream Control Transmission Protocol (SCTP)**. IETF RFC 3286, Maio de 2002. Disponível em: https://www.ietf.org/rfc/rfc3286.txt Acesso em: 24 jul. 2014.

STEWART, R. **Stream Control Transmission Protocol**. IETF RFC 4960, Setembro de 2007. Disponível em: https://www.ietf.org/rfc/rfc4960.txt Acesso em: 24 jul. 2014.

STEWART, R.; TUEXEN, M.; POON, K.; LEI, P.; YASEVICH, V. **Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)**. IETF RFC 6458, Dezembro de 2011. Disponível em: https://www.ietf.org/rfc/rfc6458.txt Acesso em: 24 jul. 2014.

KELLY, A.; MUNTEAN, G.; PERRY, P.; MURPHY, J. **Delay-centric in SCTP over WLAN**. Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE Vol. 49 (63), No. 4, Maio de 2004. Disponível em: http://elm.eeng.dcu.ie/~pel/papers/TACCS2004_49_4_2.pdf Acesso em: 24 jul. 2014.

HALL, Brian. **Beej's Guide to Network Programming Using Internet Sockets**. Julho de 2012. Disponível em: http://www.beej.us/guide/bgnet/output/print/bgnet_A4.pdf Acesso em: 24 jul. 2014.

LEUNG, V.; PARENTE RIBEIRO, E.; WAGNER, A.; IYENGAR, J. **Multihomed Communication with SCTP (Stream Control Transmission Protocol)**. CRC *Press,* Dezembro de 2012.

STEWART, R.; XIE, Q. **Stream Control Transmission Protocol (SCTP): A Reference Guide**. Addison-Wesley Longman Publishing Co., Inc., Novembro de 2001.

GAVRILOFF, I. **Análise de aspectos envolvidos no mecanismo de seleção de caminho baseado em atraso para sistemas multiabrigados utilizando SCTP**. Março de 2009. Disponível em: http://dspace.c3sl.ufpr.br/dspace/bitstream/handle/1884/18581/Dissertacao_Igor_Gavriloff.pdf?sequenc.. Acesso em: 24 jul. 2014

TORRES, A. **Método para melhoria da qualidade na transmissão de vídeos sobre o protocolo SCTP**. Abril de 2014.

SIEMON, D. **Queueing in the Linux Network Stack**. LINUX Journal, Julho de 2013. Disponível em: http://www.coverfire.com/articles/queueing-in-the-linux-network-stack/ Acesso em: 24 jul. 2014

CARDOZO, T.; VIEIRA, A.; ZIVIANI, A.; SILVA, A. **Análise Sistemática do Fenômeno Bufferbloat**. Anais do XIX Workshop de Gerência e Operação de Redes e Serviços-WGRS, 2014. Disponível em: http://sbrc2014.ufsc.br/anais/files/wgrs/ST3-2.pdf Acesso em: 24 jul. 2014

9. RELATÓRIO DE ATIVIDADES COMPLEMENTARES

Nome do bolsista: Pedro Mantovani Antunes

Plano de Trabalho do bolsista: **Implementação e avaliação de protocolo de transporte multicaminho**

Projeto de Pesquisa: Seleção Inteligente de Rotas por Sistemas Finais em Redes IP

Banpesq/Thales: 2005016733

Orientador: Prof. Eduardo Parente Ribeiro

Segue anexado à página seguinte o Histórico Escolar.

UNIVERSIDADE FEDERAL DO PARANÁ

Histórico Escolar Simplificado

Data: 24/07/2014
Hora: 13:53
IP: 10.10.5.30

Aluno: PEDRO MANTOVANI ANTUNES
Matrícula: GRR20133893

Curso: 102A - Curso de Engenharia Elétrica - Sistemas Eletrônicos Embarcados - Noturno
Período: 3

Situação: Sem evasão
Versão: 2009

Código	Disciplina	C.H.	Cred	Situação	Média / Conceito	Período
TE206	Análise Vetorial na Engenharia Elétrica	60	4	Aprovado	96,00	2013 / 1o. Semestre
TE200	Engenharia e Sociedade I	30	2	Aprovado	100,00	2013 / 1o. Semestre
TE205	Fundamentos de Sistemas Eletromecânicos	60	3	Aprovado	86,00	2013 / 1o. Semestre
TE203	Fundamentos Matemáticos para a Engenharia Elétrica I	60	4	Aprovado	94,00	2013 / 1o. Semestre
TE201	Laboratório Matemático para Engenharia Elétrica I	30	1	Aprovado	95,00	2013 / 1o. Semestre
TE207	Técnicas de Programação em Engenharia Elétrica I	60	3	Aprovado	92,00	2013 / 1o. Semestre
TE209	Circuitos Lógicos	60	4	Aprovado	91,00	2013 / 2o. Semestre
TE219	Comunicação e Expressão para Engenheiros	30	2	Aprovado	92,00	2013 / 2o. Semestre
TE204	Fundamentos Matemáticos para a Engenharia Elétrica II	60	4	Aprovado	92,00	2013 / 2o. Semestre
TE210	Fundamentos para Análise de Circuitos Elétricos	60	4	Aprovado	86,00	2013 / 2o. Semestre
TE202	Laboratório Matemático para Engenharia Elétrica II	60	1	Aprovado	84,00	2013 / 2o. Semestre
TE208	Técnicas de Programação em Engenharia Elétrica II	60	3	Aprovado	100,00	2013 / 2o. Semestre
TE211	Análise de Circuitos Elétricos I	60	4	Aprovado	88,00	2014 / 1o. Semestre
TE218	Análise de Sinais	60	3	Aprovado	93,00	2014 / 1o. Semestre
TE214	Fundamentos da Eletrônica	30	2	Matrícula	*****	2014 / 1o. Semestre
TE223	Introdução à Eletroquímica	30	2	Aprovado	100,00	2014 / 1o. Semestre
TE213	Introdução à Expressão Gráfica na Engenharia Elétrica	30	2	Aprovado	87,00	2014 / 1o. Semestre
TE229	Introdução aos Processos Estocásticos em Engenharia Elétrica	60	4	Matrícula	*****	2014 / 1o. Semestre
TE215	Laboratório de Eletrônica I	30	1	Matrícula	*****	2014 / 1o. Semestre
	Total Créditos/Carga Horária Vencidos:	810	46			

Autenticação: 2292.07A6.3686.1FDD.219B.459A.037D.B443
Consulte em: http://www.portaldoaluno.ufpr.br/usuario/publico/autenticacao/form.action

10. APRECIAÇÃO DO ORIENTADOR

O desempenho do bolsista no projeto foi excelente. O aluno teve boa dedicação e empenho nas atividades e demonstrou boas habilidades nas tarefas de programação e pesquisa. O desempenho acadêmico também foi considerado excelente. O relatório final está aprovado.

Curitiba, 05 de agosto de 2014.

Prof. Eduardo Parente Ribeiro
Departamento de Engenharia Elétrica – UFPR
Fone: 3361-3227
Email: edu@eletrica.ufpr.br

11. DATA E ASSINATURA DO BOLSISTA E ORIENTADOR

_____
Pedro Mantovani Antunes – Bolsista CNPq
Universidade Federal do Paraná

_____
Prof. Dr. Eduardo Parente Ribeiro – Orientador
Universidade Federal do Paraná

Curitiba
24 de Julho de 2014