PEDRO MANTOVANI ANTUNES

Iniciação Tecnológica / 2014 – 2015

Avaliação de protocolo de transporte multicaminho na presença de múltiplos agentes

Eduardo Parente Ribeiro / Departamento de Engenharia Elétrica

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

Curitiba

2015

PEDRO MANTOVANI ANTUNES

Iniciação Tecnológica / 2014 – 2015

Avaliação de protocolo de transporte multicaminho na presença de múltiplos agentes

Relatório apresentado à Coordenadoria de Iniciação Científica e Integração Acadêmica da Universidade Federal do Paraná por ocasião da conclusão parcial das atividades de Iniciação Científica – Edital 2014-2015

Eduardo Parente Ribeiro / Departamento de Engenharia Elétrica

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

Curitiba

2015

RESUMO

O avanço da telecomunicação torna o acesso à *Internet* cada vez mais facilitado por múltiplos meios de comunicação diferentes, como o Wifi, 3G, LTE e WiMax, por exemplo. Muitos equipamentos hoje já possuem *hardware* suficiente para acessar mais de um meio de comunicação simultaneamente, o que pode trazer vantagens para usuário final. Contudo, os protocolos de transporte mais populares, TCP e UDP, não fazem proveito deste recurso, chamado *multihoming*. O protocolo SCTP possui suporte ao *multihoming*, porém ainda não o utiliza muito eficiente, já que se aproveita apenas da redundância de acesso. Estudos mostram outras utilizações para o *multihoming*, como transferência de arquivos por múltiplos caminhos, entrega da transmissão sem perdas e trocas de caminhos para os menos congestionados. O principal algoritmo utilizado para esta última utilização do *multihoming* é o *delay-centric*, que realiza a transmissão pelo caminho de menor latência, avaliado pelos SRTTs. Últimos trabalhos mostraram certa instabilidade quando múltiplas fontes de transmissão utilizam este algoritmo simultaneamente, levando ao aumento generalizado do atraso dos pacotes. Este trabalho propõe alterações e incrementos no algoritmo de *delay-centric* para corrigir o problema da instabilidade e diminuir a latência dos pacotes. Juntamente com o *delay-centric* convencional, foi testado um algoritmo similar, o *delay-centric* preditivo, que utiliza a tendência de aumento da latência para a decisão de caminho. Este último algoritmo provou ser mais eficiente do que o *delay-centric* convencional para diminuir a instabilidade. A implementação do tempo de guarda, um algoritmo complementar, também provou melhorar a estabilidade do sistema, melhorando a experiência do usuário final em um cenário multimídia.

Palavras-chave: SCTP, *multihome*, *delay-centric*, *VoIP.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE EQUAÇÕES

1. INTRODUÇÃO

O protocolo SCTP (*Stream Control Transmission Protocol*) quando criado procurou suprir ausências deixadas pelos protocolos TCP (*Transmission Control Protocol*) e UDP (*User Datagram Protocol*), como o suporte ao *multihoming*. Este recurso oferece diversas possibilidades de implementações, que podem potencialmente melhorar a experiência do usuário final com a *Internet*. 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 ele só funciona como mecanismo de redundância, caso a transmissão pelo caminho primário falhe (perda de pacotes). Desde então, diversas soluções estão sendo propostas a fim de usufruir melhor o recurso *multihoming*.

Uma das soluções propostas é voltada para a transmissão de dados em tempo real na *Internet*, como chamadas VoIP (*Voice Over IP)*, streaming de vídeos, entre outros. Neste tipo de cenário*,* a mais importante característica para uma boa qualidade de serviço (*QoS*) é um baixo atraso na transmissão dos pacotes. O algoritmo proposto para sistemas multi-abrigados leva em conta este atraso na transmissão de todos os caminhos disponíveis, e escolhe o caminho com menor atraso para a transmissão. Este é um método prático e eficiente que melhora o desempenho significantemente caso um dos caminhos esteja com alta latência. Entretanto, últimos trabalhos mostram que quando diversas aplicações estão disputando banda sob o efeito deste algoritmo, instabilidades no mecanismo de troca de caminhos podem surgir, ocasionando elevados atrasos no envio dos pacotes.

2. REVISÃO BIBLIOGRÁFICA

2.1. Recurso *multihoming*

O interesse pelo *multihoming* na *Internet* é relativamente recente. Na época da publicação do protocolo TCP, por exemplo, não se imaginava qualquer tipo de *hardware* de sistema final popular que pudesse possuir mais de um meio de acesso com a *Internet*. O SCTP foi o primeiro protocolo a possuir o *multihoming* nativamente, porém outros métodos

também são possíveis, como a extensão *Multipath* TCP (FORD *et al*, 2013), *Multihoming* com *Mobile* IP na camada de rede (ÅHLUND *et al*, 2005) e a troca na camada de aplicação em cima do protocolo UDP (CUNNINGHAM *et al*, 2004).

2.2. Estimativa da latência

A estimativa de latência fim-a-fim é realizada tanto no SCTP quanto no TCP para calcular o *timeout* de retransmissão (RTO) de cada pacote (STEWART *et al*, 2007; POSTEL, 1981). Para adquirir essa estimativa, os protocolos medem o tempo de ida e volta dos pacotes (RTT) toda vez que um pacote ACK é recebido. Como o valor do RTT é altamente variável, o protocolo calcula o SRTT (*Smoothed Round Trip Time*), definido na Equação 1:

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

Equação 1: Fórmula para cálculo do SRTT

Onde o valor de α é 0.125, conforme recomendado pelas RFC 4960 e RFC 6298 (PAXSON, 2011). O primeiro valor do SRTT no início de uma transmissão é igual o valor do primeiro RTT calculado.

A estimativa de latência dos caminhos secundários no SCTP também é feito pelo SRTT. O protocolo envia pacotes de pulsação nos caminhos secundários, denominados *Heartbeats*, a fim de verificar a disponibilidade e o SRTT dos caminhos. Contudo, o intervalo padrão de envio entre *Heartbeats* é de 30s, o que acaba não dando a devida estimativa do SRTT necessária para aplicações de tempo real. Para resolver este problema, diversos autores empregam o valor do intervalo de HB como sendo 1s (KELLY *et al*, 2004; GAVRILOFF, 2009; TORRES, 2014).

2.3. Algoritmo *delay-centric*

O sistema *multihoming* atualmente suportado pelo SCTP 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

delay-centric foi o algoritmo baseado em atraso proposto como alternativa ao atual esquema de trocas de caminho (KELLY *et al, 2004*). Este mecanismo avalia o SRTT de todos os caminhos disponíveis e troca a transmissão para o caminho de menor SRTT. Uma histerese também pode ser adicionada, a fim de evitar a troca desnecessária de caminhos quando os SRTTs forem muito próximos. A histerese proposta é de 10ms.

2.4. Tempo de guarda

O tempo de guarda é um algoritmo auxiliar que pode ser implementado juntamente com qualquer outro algoritmo de seleção, e pode ajudar a evitar instabilidades quando há concorrência de aplicações. Quando uma troca é sinalizada, o tempo de guarda faz com que o algoritmo aguarde um tempo antes de efetivamente realizar a troca de caminho (GAVRILOFF, 2009). A adição de uma aleatoriedade no tempo de guarda melhora a distribuição do instante de decisão de troca ao longo do tempo. O valor de 1 a 4s, sorteado aleatoriamente, é utilizado como tempo de guarda neste trabalho.

Também foi proposta uma redução no intervalo de envio dos *Heartbeats* durante o tempo de guarda para 20ms, a fim de melhorar a estimativa do SRTT do caminho secundário no momento crítico.

2.5. *Delay-centric* preditivo

Uma alternativa proposta ao delay-centric tradicional consiste na consideração da tendência do SRTT como critério para realizar a troca de caminho (TORRES, 2014). O mecanismo é apelidado de preditivo pois tenta prever o aumento ou a diminuição do SRTT através do método MACD (*Moving Average Convergence / Divergence*). Para tal, são calculados dois SRTTs por caminho. Uma versão de curto prazo, com $\alpha = 0.667$, chamado de $SRTT_S$, e uma versão de longo prazo, com $\alpha = 0.154$, chamado de $SRTT_L$ (TORRES, 2014). Estas duas versões de SRTT são calculadas porque quando $SRTT_S$ torna-se maior que $SRTT_L$, existe uma tendência de aumento da latência geral do caminho. Desta forma, a troca de caminho apenas ocorre quando:

- A latência do caminho primário tende a aumentar, ou seja, $SRTT_S > SRTT_L$.

- $SRTT_S$ do caminho primário é maior que um limiar, definido como 150ms.
- $SRTT_S$ do caminho primário é maior que o $SRTT_S$ do caminho alternativo.

Este método apresentou melhoras de qualidade em diversos testes de *streaming* de vídeo realizados, com dados coletados de *Wifi*, 3G e *Ethernet* (TORRES, 2014).

Aqui apresentamos uma versão modificada deste algoritmo, para que o cálculo da tendência ocorra tanto no caminho primário quanto no secundário. Desta forma, caso a tendência de aumentar a latência do caminho secundário seja maior que a do caminho primário, a troca de caminho não irá ocorrer. Isso se opõe ao primeiro item do algoritmo original, onde apenas a tendência do caminho primário é levada em conta. Assim, este item é substituído por:

- A tendência de aumentar a latência é maior no caminho primário do que no secundário, ou seja, $(SRTT_S - SRTT_L) > (srtt_S - srtt_L)$. Onde SRTT em letras maiúsculas é o SRTT do caminho primário, e srtt em letras minúsculas, do caminho secundário.

2.6. Concorrência de Aplicações

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 comunicativos 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. Alguns parâmetros no algoritmo de seleção de caminho podem ser alterados de modo a diminuir esta instabilidade.

3. MATERIAIS E MÉTODOS

3.1. Topologia de Rede e Computadores

A topologia de rede consistiu em dois computadores com duas interfaces de rede *Ethernet*. Cada uma das placas de rede de cada computador foi ligada diretamente à outra, através de cabos *crossover*. Os enlaces foram configurados para não conseguirem comunicar entre si. Os resultados aqui foram então obtidos para apenas dois caminhos disponíveis. A Figura 1 ilustra a topologia utilizada.

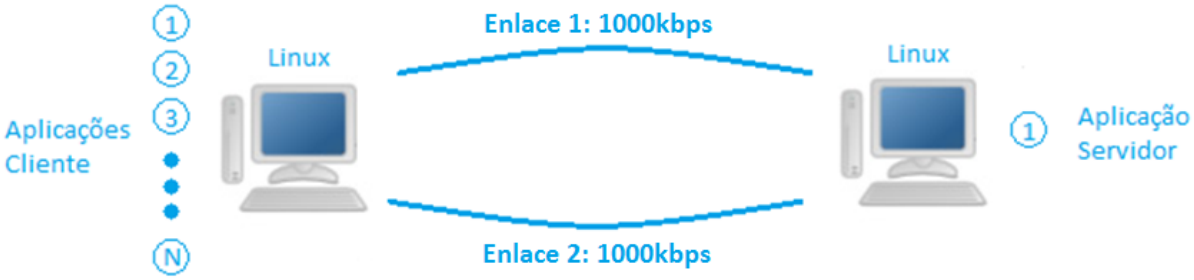

Figura 1: Topologia utilizada nos experimentos

Ambos os computadores operam com o sistema operacional Linux, distribuição Debian, *kernel* 3.2.63-2. Todos os programas de teste foram escritos em C e compilados com o gcc versão 4.7.2-5. Além disso, para trabalhar com menores taxas de envio, a banda de envio de cada interface de rede foi limitada a 1000kbps com o controle de tráfego do *kernel* do Linux (tc). A fila de saída foi regulada ao máximo valor, de forma a simular a latência gerada pela fila de diversos roteadores entre um computador e outro.

3.2. 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.

A transmissão de dados consistiu em 6 fontes UDP comunicando através de um regime CBR (*Constant Bit* Rate) com pacotes de 250B com cabeçalhos incluídos. Todos os pacotes são retornados quando chegam ao servidor através de pacotes ACK de 52B de tamanho incluindo os cabeçalhos. Cada uma das fontes também envia *heartbeats* no caminho secundário. Para evitar o sincronismo entre as fontes, cada uma das transmissões é iniciada aleatoriamente durante o primeiro segundo do experimento.

3.3. Análise dos dados

Com o objetivo de se obter uma maior precisão estatística, cada cenário foi repetido 200 vezes, sendo que cada rodada consistia no envio de 3000 pacotes por aplicação. Através desta repetição, foi possível obter o atraso médio dos pacotes, seu desvio padrão e os intervalos de confiança. Este valor de 200 repetições para cada experimento foi obtido empiricamente. Para chegar a tal valor, foi realizado o experimento de *delay*-centric com 2000 repetições. A fim de obter uma estimativa, foi considerada a média das 2000 repetições como sendo a média do experimento. Foi desenhado então o gráfico do erro percentual da média em função do número de simulações realizadas. Três pontos do gráfico completo foram analisados: ρ=0.90, ρ=0.93 e ρ=0.96. A definição de ρ é explicada adiante na Equação 2.

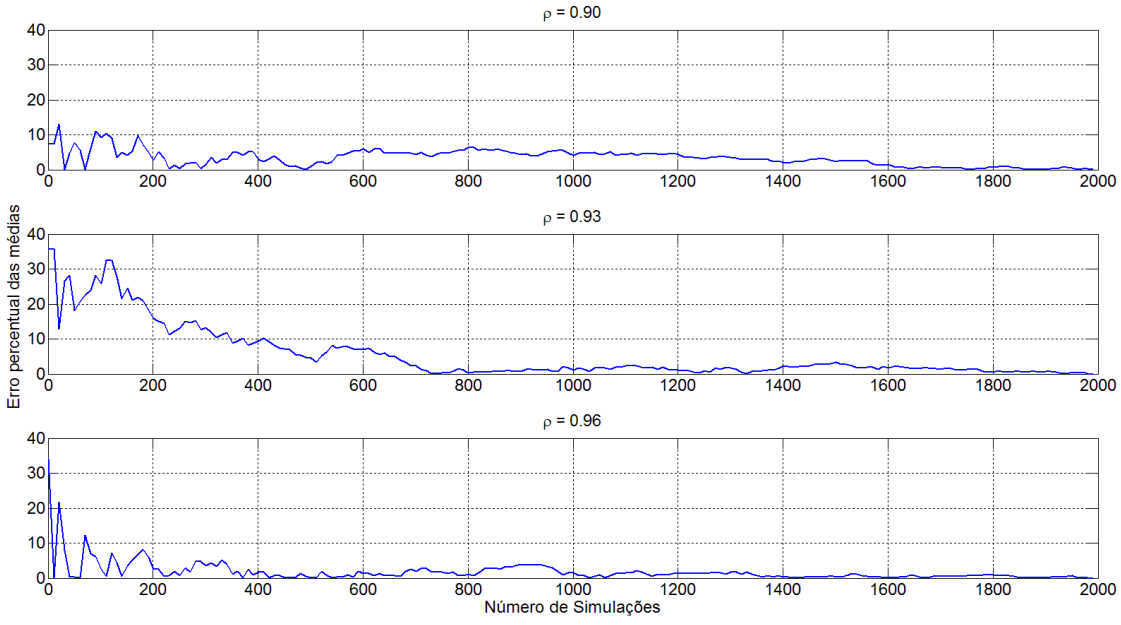

Figura 2: Erro percentual da média em função do número de simulações

A razão para se escolher o valor de 200 repetições para todos os cenários foi que este número não apresentava erros percentuais tão acentuados quanto números inferiores de simulações, e também apresentou um tempo de simulação razoável. Caso fosse escolhido fazer 2000 simulações para todos os cenários, o tempo de simulação seria 10 vezes maior, e os resultados demorariam muito mais para serem obtidos.

A avaliação da qualidade dos algoritmos foi feita através do atraso médio dos pacotes, que foi colocada em um gráfico em função da utilização de banda ρ. A definição de ρ foi estendida para o cenário multicaminho conforme a Equação 2.

$$\boldsymbol{\rho = \frac{B}{C}}$$

Equação 2: Utilização de banda

Na equação, B é o tráfego total sendo enviado em kbps. C é a capacidade agregada dos caminhos em kbps. Como cada enlace foi limitado em 1000kbps, C é 2000kbps para a topologia deste trabalho. Nos experimentos, ρ foi variado de 0.66 a 0.99. Valores inferiores a 0.66 foram desconsiderados porque abaixo desta utilização de banda quase não existem casos de instabilidade e o atraso dos pacotes é quase 0ms.

3.4. Cenários analisados

Foram analisados quatro cenários diferentes neste trabalho. Estes quatro trabalhos realizam uma comparação entre os algoritmos de *delay-centric* e *delay-centric* preditivo, com e sem tempo de guarda. Especificamente, os cenários são:

- *Delay-centric* tradicional, com histerese = 10ms.
- *Delay-centric* tradicional, com histerese = 10ms e tempo de guarda = 1 a 4s.
- *Delay-centric* preditivo, limiar de troca = 70ms.
- *Delay-centric* preditivo, limiar de troca = 70ms e tempo de guarda = 1 a 4s.

O algoritmo *delay-centric* foi realizado como no trabalho de Kelly (2004), com a adição de uma histerese de troca de 10ms, para evitar trocas sucessivas. Por outro lado, o algoritmo de *delay-centric* preditivo foi realizado com as alterações explicadas no item 2.5. Além disso, o limiar de troca foi configurado em 70ms. Isto foi feito porque na topologia usada, todos os atrasos de pacotes são gerados no caminho de ida, pela banda ser limitada

apenas no computador de envio. Isto não reflete o cenário real da *Internet*, onde os atrasos são gerados tanto no caminho de ida quanto no de volta.

O intervalo de *Heartbeat* em todos os mecanismos foi configurado aleatoriamente entre 0.5 e 1.5s, sendo este intervalo sorteado após cada *Heartbeat* enviado. A única exceção é quando o algoritmo está no tempo de guarda, onde o intervalo de *Heartbeat* é alterado para 20ms.

Todos os mecanismos foram simulados também com a presença de tráfego de fundo, a fim de deixar os experimentos mais parecidos com o cenário real da *Internet*. O tráfego de fundo consistiu de pacotes UDP de tamanho fixo de 250 bytes com tempo de envio aleatório, distribuído de forma exponencial. Apenas a proporção de 40% de tráfego de fundo e 60% de tráfego principal foi apresentada nos resultados.

4. RESULTADOS

4.1. *Delay-centric*

O experimento com o *delay-centric* inicialmente serviu como base para verificar o problema da instabilidade das fontes comunicativas, e para determinar o comportamento deste algoritmo na distribuição das fontes entre os caminhos. Em um cenário com baixa utilização de banda, com ρ menor que 0.75, o *delay-centric* conseguiu distribuir as fontes entre os caminhos de forma eficiente, mantendo o atraso baixo para todos os agentes. Conforme a utilização de banda foi aumentando, o mecanismo de troca de caminho começou a apresentar instabilidades e não conseguiu distribuir as fontes igualmente entre os caminhos, gerando altos atrasos de pacotes. Este comportamento fez com que o histograma da distribuição do atraso médio dos experimentos distribuísse em duas modas. A moda superior, representando os casos de instabilidade, apresenta atrasos elevados na transmissão dos pacotes. A moda inferior, que representa os casos de estabilidade, tem o atraso nos pacotes muito perto de 0ms. Este histograma bimodal significa que a instabilidade ou estabilidade persiste do início ao fim da execução das aplicações, ou seja,

caso o algoritmo entre em um cenário de instabilidade, ele não sairá dele até o final da execução das aplicações que geram esta instabilidade.

A Figura 3 ilustra o cenário de *delay-centric*, mostrando as modas superior e inferior, assim como o atraso médio total dos pacotes de todo o experimento. Para cada moda, também são representados os intervalos de confiança de 95%.

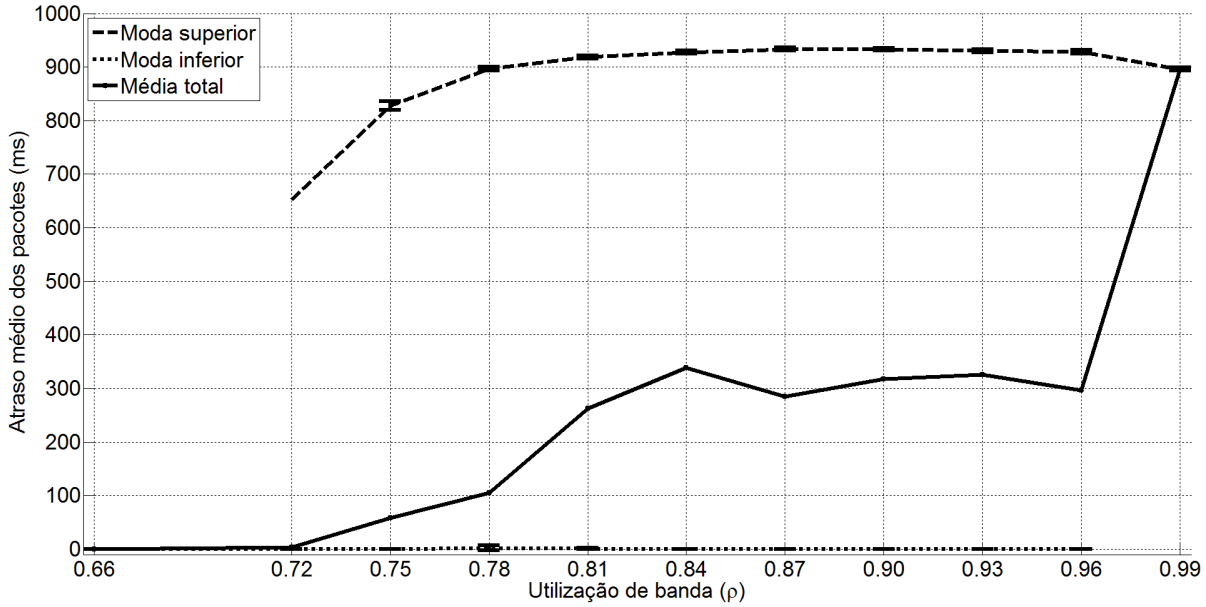

Figura 3: Mecanismo de *delay-centric*

A partir do gráfico, nota-se que o atraso médio dos pacotes passa de 100ms quando ρ é maior ou igual a 0.78. Este valor de 100ms é crítico pois a partir deste valor, a qualidade da transmissão de dados em tempo real começa a ser prejudicada. Já não é possível garantir o QoS de uma ligação VoIP, por exemplo.

O experimento com o *delay-centric* inicialmente serviu como base para verificar o problema da instabilidade presente no algoritmo, uma vez que em um cenário ideal, sem instabilidades, o atraso permaneceria perto de 0ms até ρ = 1 por se tratar de um regime CBR.

4.2. *Delay-centric* com tempo de guarda

O próximo experimento realizado foi empregado o mecanismo *delay-centric* como no item 4.1., porém com a adição do tempo de guarda, conforme explicado no item 3.4. A representação deste experimento está na Figura 4.

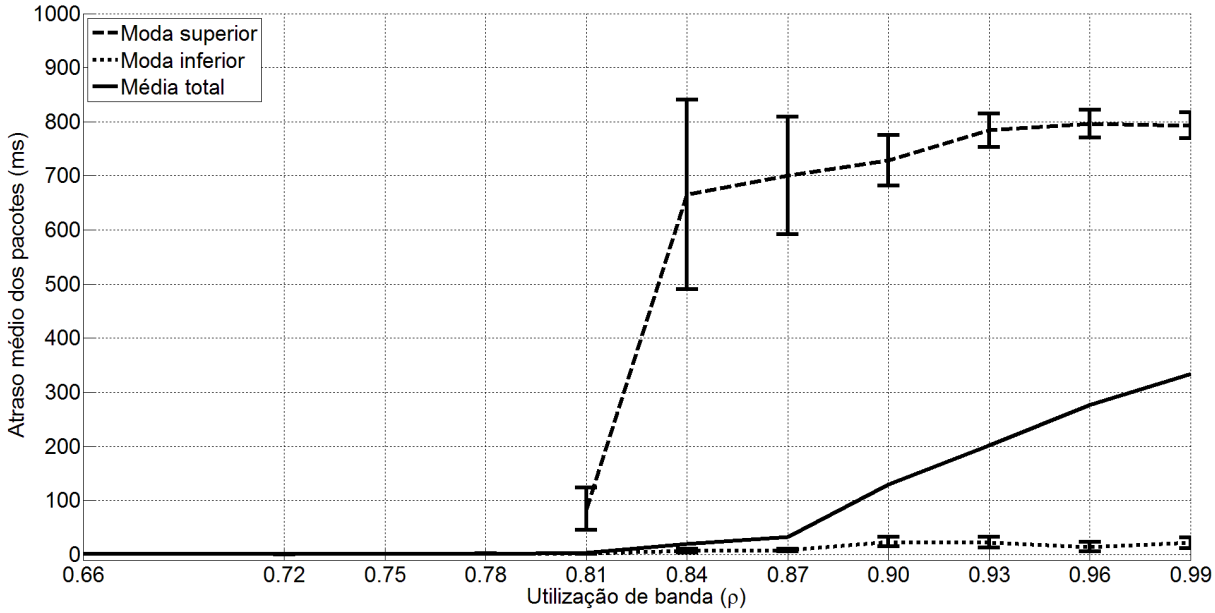

Figura 4: Mecanismo de *delay-centric* com tempo de guarda

Como é observado, a implementação do tempo de guarda reduziu muito os casos de instabilidade e conseguiu abaixar muito o atraso médio dos pacotes em relação ao *delay-centric* puro. Isto acontece devido à redução do intervalo de envio dos *Heartbeats*, o que eleva a precisão do SRTT do caminho secundário em um momento crítico. Como o valor médio de 1 segundo pode ser muito alto para o intervalo entre *Heartbeats* em um cenário tão dinâmico quanto este, a diminuição para 20ms durante o tempo de guarda, aliada com a aleatoriedade deste, pode ter sido a principal causa da melhora da estabilidade do sistema.

4.3. *Delay-centric* preditivo

Apesar de o *delay-centric* preditivo nunca ter sido testado em um cenário com múltiplas fontes de transmissão, esperava-se que este mecanismo apresentasse um comportamento melhor que o *delay-centric* tradicional, já que as tendências que aparecem no preditivo podem indicar o estado transiente dos outros caminhos. A Figura 5 ilustra o cenário do *delay-centric* preditivo.

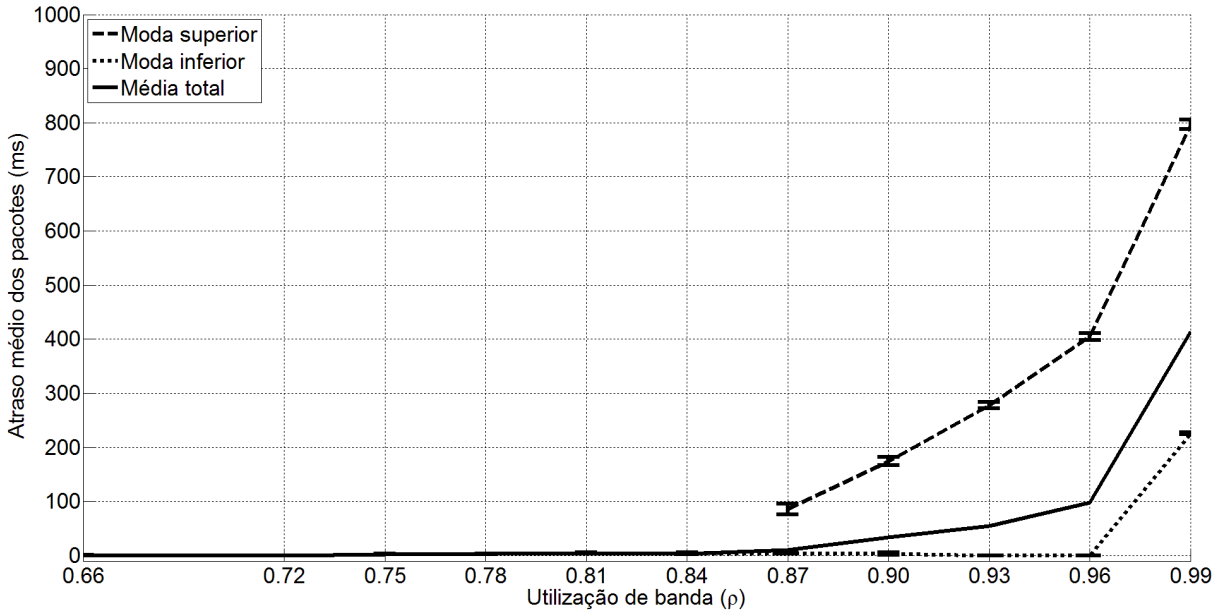

Figura 5: Mecanismo de *delay-centric* preditivo

O *delay-centric* preditivo apresentou menores atrasos de pacotes para todas as faixas de utilização de banda testadas, comparando-se ao *delay-centric* tradicional. Isso pode comprovar a tendência natural do algoritmo em atingir a estabilidade. Mesmo nos casos de instabilidade, percebe-se que o atraso médio na moda superior é menor que no *delay-centric.*

Comparando ao item 4.2., que possui tempo de guarda, o algoritmo MACD só desempenhou pior no último valor de utilização de banda testado, 0.99. Em todos os outros valores, o preditivo obteve um atraso menor ou próximo.

4.4. *Delay-centric* preditivo com tempo de guarda

Assim como o tempo de guarda ajudou o mecanismo de *delay-centric* a diminuir os níveis de latência e instabilidade, o mesmo era esperado para o *delay-centric* preditivo. O gráfico obtido deste cenário está representado na Figura 6.

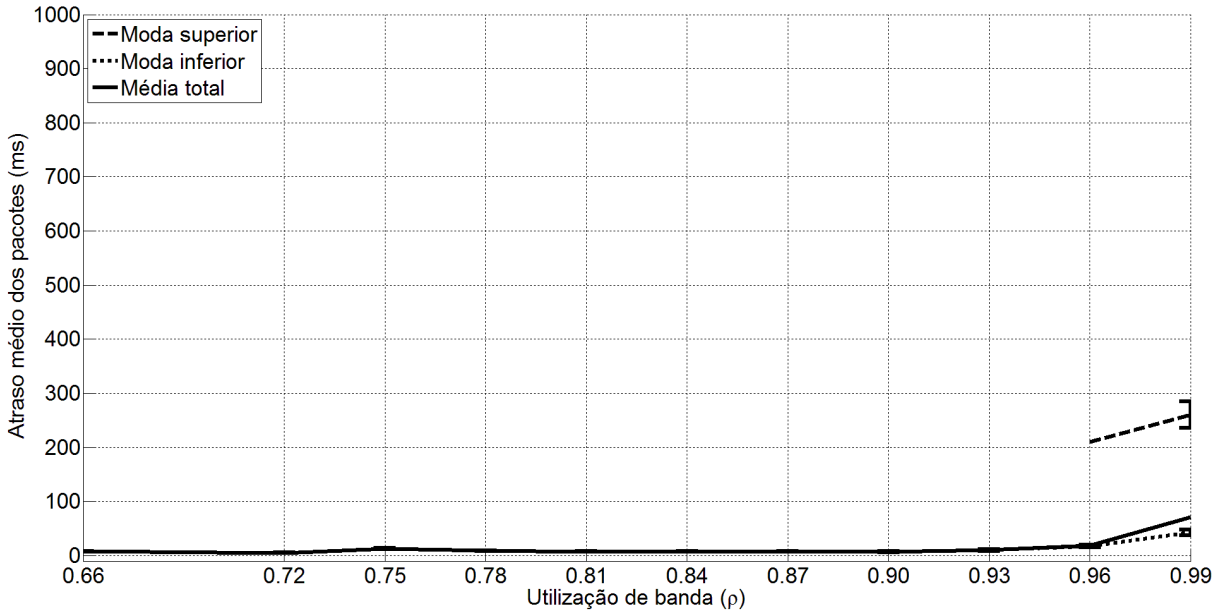

Figura 6: Mecanismo de *delay-centric* preditivo com tempo de guarda

De fato, os atrasos médios analisados para este cenário foram muito baixos, mesmo nas maiores utilizações de banda. Os casos de instabilidade só começaram a surgir em ρ maior de 0.96, onde a instabilidade já se torna quase inevitável, mostrando a eficiência do algoritmo. A aliança entre os mecanismos do *delay-centric* preditivo, tempo de guarda e redução do intervalo de *Heartbeats* durante o tempo de guarda provou ser uma combinação excelente, e mesmo nos piores casos, o atraso médio dos pacotes não foi alto o suficiente para prejudicar muito o QoS de uma ligação VoIP, por exemplo.

4.5. Cenários com tráfego de fundo

Os experimentos executados anteriormente sem o tráfego de fundo já apresentam respostas muito importantes, como o comportamento de cada mecanismo de troca em cenários de alta utilização de banda, mostrando a tendência de cada um a sofrer instabilidades. Entretanto, o cenário real da *Internet* é diferente. Diversos fluxos imprevisíveis irão gerar filas nos roteadores de forma aleatória, e a latência de cada caminho será diretamente influenciada por isso.

Para simular tais condições, um programa gerador de tráfego foi inicializado junto com as fontes de transmissão. Este gerador de tráfego cria pacotes de um tamanho fixo e os envia espaçados de forma aleatória e exponencialmente distribuída. A proporção aqui analisada foi de 40% de tráfego de fundo e 60% de tráfego principal.

O gráfico continuou sendo analisado da mesma forma, atraso médio dos pacotes em função da utilização de banda. A única diferença é que o tráfego de fundo induziu os algoritmos a se distribuírem em torno de uma moda somente, ao invés de duas. A Figura 7 apresenta o comportamento de todos os mecanismos de troca analisados neste cenário.

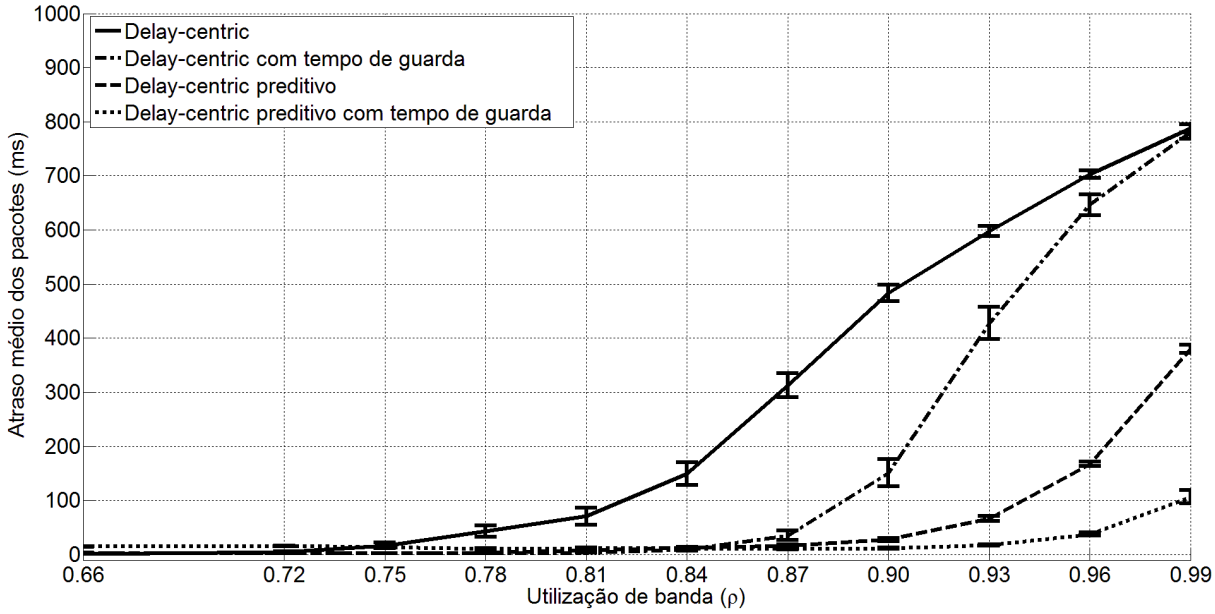

Figura 7: Cenário com tráfego de fundo

Da mesma forma que no cenário sem tráfego de fundo, o *delay-centric* preditivo com tempo de guarda foi o mecanismo que apresentou melhor desempenho em relação aos demais. Para todos os algoritmos, a inserção de um tráfego de fundo aumentou o atraso médio dos pacotes, porém não de forma acentuada. A única exceção a isto foi o *delay-centric* tradicional, em que o atraso médio abaixo entre ρ = 0.75 e 0.87.

5. CONCLUSÕES

Com a modernização dos hardwares, o recurso *multihoming* está cada vez mais popular entre os sistemas finais de *Internet*, como *notebooks* e celulares. Com esta popularização, é natural que os protocolos de comunicação tentem desfrutar deste recurso de forma eficiente.

Em transmissões multimídia em tempo real, como chamadas VoIP e *streaming* de vídeos, a condição mais desejada é uma comunicação sem atrasos. O recurso *multihoming* pode auxiliar este tipo de transmissão selecionando o caminho com menor atraso de

pacotes, através de um mecanismo chamado *delay-centric*. Diversas pesquisas mostram as vantagens da implementação do *delay-centric* como uma das formas de utilizar o *multihoming* em protocolos que já possuem suporte a este recurso, como o SCTP.

Entretanto, últimos trabalhos mostram um problema de instabilidade com o algoritmo de *delay-centric* quando múltiplas fontes de comunicação utilizam este mecanismo simultaneamente. Neste trabalho comprovamos este problema de instabilidade, que inevitavelmente eleva os atrasos dos pacotes. Foi provada que uma das causas de instabilidade é a baixa amostragem do SRTT nos caminhos secundários. Isto foi corrigido com a implementação de um mecanismo complementar, chamado tempo de guarda, onde as fontes aguardam um tempo aleatório para confirmar as trocas de caminho. Durante o tempo de guarda, foi alterado o intervalo de envio de *Heartbeats* para 20ms, o que elevou a estimativa de latência do caminho secundário, consequentemente eliminando as instabilidades do sistema e abaixando os atrasos dos pacotes.

Um algoritmo alternativo ao *delay-centric* também foi testado, chamado *delay-centric* preditivo. Este mecanismo analisa as tendências de aumento e diminuição dos atrasos e escolhe o melhor caminho baseado nisto. Foi demonstrado que este método de seleção de caminho possui melhor desempenho na eliminação de instabilidades do que o *delay-centric* tradicional. Aliado com o mecanismo de tempo de guarda e diminuição do intervalo entre *Heartbeats*, o *delay-centric* preditivo conseguiu eliminar as instabilidades em quase 100% das simulações.

Todos os mecanismos de troca também foram testados com a inserção de um tráfego exponencial e aleatório ao fundo, que gerava atrasos transientes nos caminhos. Assim como nos testes sem este tráfego de fundo, o *delay-centric* preditivo com tempo de guarda desempenhou melhor e conseguiu manter os atrasos baixos para todas as fontes de comunicação, mesmo nas utilizações de banda mais elevadas.

6. REFERÊNCIAS BIBLIOGRÁFICAS

STEWART, R.; RAMALHO, M.; XIE, Q.; TUEXEN M.; CONRAD P. **Stream Control Transmission Protocol (SCTP)**. RFC 4960, 2007.

FORD, A.; RAICIU C.; HANDLEY, M.; BONAVENTURE, O. **TCP Extensions for Multipath Operation with Multiple Addresses**. RFC 6824, 2013.

ÅHLUND, C.; ZASLAVSKY, A. **Multihoming with Mobile IP**. Lecture Notes in Computer Science Volume 2720, p 235-243, 2003.

CUNNINGHAM, G.; MURPHY, L.; MURPHY, S.; PERRY, P. **Seamless Handover of Streamed Video over UDP between Wireless LANs**. IEEE conference, p 1-6, 2004.

POSTEL, J. **Transmission Control Protocol**. RFC 793, 1981.

PAXSON, V.; ALLMAN, M.; CHU, J.; SARGENT, M. **Computing TCP's Retransmission Timer**. RFC 6298, 2011.

KELLY, A.; MUNTEAN, G.; PERRY, P.; MURPHY, J. **Delay-centric Handover in SCTP over WLAN**. Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE, vol. 49, no. 63, 2004.

GAVRILOFF, I. **Análise de aspectos envolvidos no mecanismo de seleção de caminho baseado em atraso para sistemas multiabrigados utilizando SCTP**. Dissertação de Mestrado – UFPR, 2009.

TORRES, A. **Método para melhoria da qualidade na transmissão de vídeos sobre o protocolo SCTP**. Dissertação de Mestrado – UFPR, 2014.

ONG, L.; YOAKUM, J. **An Introduction to the Stream Control Transmission Protocol (SCTP)**. RFC 3286, 2002.

STEWART, R.; TUEXEN, M.; POON, K.; LEI, P.; YASEVICH, V. **Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)**. RFC 6458, 2011.

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

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

SIEMON, D. **Queueing in the Linux Network Stack**. LINUX Journal, 2013. Disponível em: http://www.coverfire.com/articles/queueing-in-the-linux-network-stack/ Acesso em: 03 mar. 2015.

THE LINUX FOUNDATION. **Netem**. 2009. Disponível em: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem Acesso em: 03 mar. 2015.