Análise de estabilidade e imparcialidade em um novo algoritmo para transmissão multicast em camadas

Valter Roesler[1], Gaspare Giuliano Bruno[2] e José Valdeni de Lima[2]

roesler@exatas.unisinos.br, {gaspare, valdeni}@inf.ufrgs.br

[1]UNISINOS – Universidade do Vale do Rio dos Sinos, Centro de Ciências Exatas e Tecnológicas. Av. Unisinos, 950, CEP 93022-000. São Leopoldo – RS – Brasil

[2]UFRGS – Universidade Federal do Rio Grande do Sul – Instituto de Informática. Caixa Postal 15.064, CEP 91501-970. Porto Alegre – RS – Brasil

Abstract

This paper focuses on stability and fairness issues of a new congestion control algorithm to be used in receiver adaptation for layered multicast transmissions. Its main benefit is to allow multimedia transmissions in heterogeneous environments such as the Internet, providing access to clients with different bandwidths, adapting automatically to the network's capacity.

The proposed algorithm is called ALM (Adaptive Layered Multicast), and the adaptation occurs on the receiver side, ensuring a great simplicity to the system, as the only function of the transmitter is to send data in different layers (multicast groups) using packet pairs [1], without any need of confirmation or response from the receivers.

The internal elements of the network, such as routers, do not need any modification (like running a special piece of software or using QoS), and the protocol runs over today's Internet. To validate the method in terms of stability and fairness with concurrency of various ALM sessions, the ns2 [2] software was used (with modifications where required). Various simulations were undertaken to prove the algorithm's functionality using tail drop and RED discarding policies on the router queues, and the evaluation environment was almost the same as [3] and [4], in order to make a comparison with other works in this area.

The idea behind the ALM's algorithm is as simple as that: "it gives more bandwidth to those who have less". That means that two concurrent sessions without losses will tend to reach equilibrium, because the session with less bandwidth will have a higher increasing rate in relation to the other one.

***Keywords**: congestion control algorithm, multimedia transmission, layering multicast, adaptability, multimedia distributed systems.*

Resumo

Este trabalho tem por objetivo analisar questões de estabilidade e imparcialidade em um novo algoritmo de controle de congestionamento. Este algoritmo é usado para adaptação no receptor em transmissões multimídia utilizando multicast em camadas. Seu maior benefício é permitir transmissões multimídia em ambientes heterogêneos como a Internet, permitindo acesso a clientes localizados em enlaces com diferentes larguras de banda, pois os mesmo se adaptam automaticamente à capacidade da rede naquele momento.

O algoritmo proposto foi chamado ALM (Adaptive Layered Multicast), e a adaptação ocorre no lado do receptor, assegurando uma grande simplicidade ao sistema, visto que a única função do transmissor é enviar os dados em diferentes camadas[1] usando pares de pacotes [1], sem qualquer necessidade de confirmação ou resposta dos receptores.

Os elementos internos da rede, tais como os roteadores, não necessitam quaisquer modificações (como por exemplo a execução de um determinado código ou utilização de QoS), e o protocolo funciona sobre a Internet atual. Para validar o método em termos de estabilidade e imparcialidade com várias sessões ALM concorrendo pela banda, se utilizou o software ns2 [2], com modificações no código onde requerido. Várias simulações foram feitas para provar a funcionalidade do algoritmo utilizando políticas de descarte RED e droptail *nas filas dos roteadores. O ambiente de avaliação utilizado foi parecido com o utilizado por Gopalakrishnan em [3] e [4], a fim de fazer uma comparação com outros trabalhos na mesma área.*

A idéia base do algoritmo do ALM é simples, e seu lema é: "aumentar mais a banda para os tráfegos que têm menos". Isso significa que duas sessões concorrentes que não estejam sofrendo perdas na rede vão tender a um ponto de equilíbrio, pois a sessão que está utilizando menos banda vai ter uma taxa de crescimento maior em relação à outra.

***Palavras-chave**: algoritmos para controle de congestionamento, transmissão multimídia, multicast em camadas, adaptabilidade, sistemas distribuídos multimídia.*

1 Introdução

Um dos grandes desafios atuais da Internet é permitir acesso universal às transmissões multimídia, mesmo para receptores localizados em redes com diferentes larguras de banda e com vários tipos de resolução de tela, ou mesmo características diversas de processamento.

O uso do multicast permite que uma única transmissão seja entregue a um grande número de receptores através da rede, entretanto, a heterogeneidade[2] da Internet torna o uso do multicast um problema de difícil solução, exigindo que a comunicação utilize algum método de permitir acesso tanto aos receptores localizados em enlaces rápidos quanto aos localizados em enlaces lentos.

[1] Cada camada representa, na prática, um grupo multicast diferente.

[2] Nesse caso heterogeneidade se refere a topologias físicas e níveis de congestionamento diferentes, fazendo com que existam tanto receptores localizados em enlaces rápidos como lentos.

Uma solução para o problema da heterogeneidade da rede é a utilização de transmissões multicast em camadas, na qual o transmissor codifica o fluxo da mídia em várias camadas e transmite cada camada utilizando um grupo multicast diferente. Cada camada adiciona qualidade à anterior, e o receptor se inscreve em tantas camadas quanto permitirem as condições da rede e da sua máquina. Este mecanismo foi definido por McCanne em 1996 [5], e é definido neste trabalho como "camadas complementares", pois cada camada complementa a qualidade da anterior. Cabe observar que a causa do congestionamento pode não estar localizada no enlace do receptor, entretanto, quando esse receptor e todos os outros que estão sentindo o congestionamento deixam de assinar determinado grupo multicast (normalmente da camada mais alta), o fluxo de dados diminui e o congestionamento tende a diminuir.

Outro tipo de transmissão em camadas é definida aqui como "camadas opcionais", que consistem de camadas adequadas a um determinado conjunto de receptores, mas não a outros, como por exemplo legendas em duas linguagens diferentes [6].

Em geral, a adaptação para transmissões multicast em camadas pode acontecer nos seguintes pontos [7]:

- **Lado do transmissor**: é necessária uma certa interação entre o transmissor e os receptores, a fim de que o transmissor seja informado das condições da rede, causando tráfego na rede e afetando a escalabilidade.
- **Núcleo da rede**: necessita de uma parte do algoritmo ou um tipo especial de fila rodando no núcleo da rede (como os roteadores), a funcionalidade fica dependente de mudanças na infraestrutura da rede e/ou implementações especiais pelos fabricantes de roteadores, tornando difícil sua utilização nos dias de hoje
- **Lado do receptor**: é o método utilizado no algoritmo proposto neste trabalho (ALM). O objetivo foi manter o sistema simples, assim, nenhuma interação regular é necessária entre transmissor e receptores, e o algoritmo não necessita quaisquer alterações nos roteadores.

Com o uso da adaptação no lado do receptor, o sistema fica mais simples como um todo, gerando menos tráfego na rede. Além disso, as filas utilizadas podem ser FIFO com política de descarte RED (*Random Early Detection*) [8] ou *droptail*, podendo ser utilizado na Internet atual. Evidentemente que o algoritmo se beneficiaria do uso do QoS, mas a idéia é obter o máximo possível utilizando uma rede *best-effort*. Os benefícios do uso de QoS deverão ser analisados em um artigo futuro.

Os eventos adequados para a utilização do algoritmo devem ser grandes o suficiente para justificar o aumento da complexidade na infraestrutura da rede e tráfego de controle gerado pelas camadas na transmissão multicast, mas ainda devem ser pequenos o suficiente ou muito esparsos geograficamente para serem distribuídos com um custo efetivo através de satélite ou cabo. Exemplos de eventos adequados podem ser a transmissão de uma grande conferência internacional ou de um pequeno canal de TV.

Este artigo tem como principal foco a análise da estabilidade e imparcialidade entre diferentes sessões ALM compartilhando um gargalo através de um canal privativo, e as próximas sessões estão organizadas da seguinte forma: a sessão 2 define os conceitos de estabilidade e imparcialidade utilizados neste artigo, bem como as alternativas a serem utilizadas pela aplicação a fim de se adaptar a certas instabilidades. A sessão 3 mostra o ambiente de avaliação considerado para as simulações, enquanto a sessão 4 mostra os detalhes do algoritmo usado no lado do receptor. Na sessão 5, várias simulações que foram usadas para validar o sistema podem ser vistas, bem como os gráficos e resultados. A sessão 6 discute alguns trabalhos relacionados ao assunto e a sessão 7 apresenta algumas conclusões.

2 Definições de estabilidade e imparcialidade

Uma boa estabilidade pode ser definida como a habilidade do receptor manter a transmissão sem muitas variações na qualidade da imagem ou mudanças no seu tamanho, pois isso pode perturbar o usuário. Nesse artigo, é considerado um distúrbio na estabilidade quando acontecem variações na qualidade percebida pelo usuário devido à *joins* ou *leaves* efetuados pelo algoritmo rodando no receptor. Essa variação pode acontecer devido a mudanças no tráfego (como o início de uma nova sessão concorrente) ou a uma característica do algoritmo.

A aplicação pode minimizar até certo ponto a instabilidade recebida, adaptando-se à característica do algoritmo nas seguintes maneiras:

- **Instabilidade devido ao *join***: suponha que o receptor está decodificando 4 camadas de uma transmissão de vídeo. Se ele faz *join* para a camada 5, o receptor deve esperar um período de estabilização (5 segundos, por exemplo), antes de modificar a qualidade que está sendo mostrada ao usuário. Esta precaução pode impedir mudanças temporárias e potencialmente irritantes na qualidade percebida pelo usuário no caso de um *join* sem sucesso (que gerou perdas, forçando o receptor a efetuar o *leave* logo em seguida).
- **Adaptação progressiva**: caso o *join* do item anterior durar mais do que 5 segundos, é sinal que o sistema se adaptou bem à nova carga na rede e essa melhoria na qualidade deve ser passada ao usuário. Em vez de fazer isso rapidamente, a aplicação deve fazer essa adaptação de qualidade lentamente, de forma a não causar desconforto visual ao usuário.

A situação fica mais complexa quando acontecem instabilidades temporárias que fazem o algoritmo deixar uma determinada camada, voltando a ela poucos instantes mais tarde. No caso de *leave*, o receptor pára de receber as informações (pacotes com sinal codificado) que estava utilizando para gerar o vídeo com aquela qualidade, portanto, não há formas de manter o mesmo nível de qualidade para o usuário, fazendo com que o mesmo perceba uma diminuição / variação de qualidade.

Além da estabilidade, este artigo trata da imparcialidade[3] entre as diferentes sessões sendo transmitidas. Cada receptor decide independentemente o número de camadas que vai se inscrever, e no caso de diferentes sessões compartilhando o mesmo gargalo, a independência entre os receptores pode causar injustiças, ou a divisão desigual de largura de banda entre eles.

O objetivo do algoritmo do ALM é fornecer uma alocação de banda imparcial entre todas as sessões compartilhando o mesmo gargalo, ou seja, todas as sessões devem transferir idealmente a mesma soma de bytes durante um certo período de tempo, considerando as mesmas condições para todos os receptores.

3 Ambiente de avaliação

O objetivo deste artigo é mostrar os resultados do algoritmo para questões de estabilidade e imparcialidade entre diferentes sessões ALM interagindo entre si através de um gargalo comum, assim, foi considerada uma rede privativa virtual por onde passam várias sessões ALM concorrentes. Outro artigo trata da adaptação do algoritmo competindo com tráfego TCP e UDP [9].

Alguns parâmetros utilizados neste artigo para as simulações foram os seguintes:

[3] Imparcialidade neste artigo se refere à divisão igualitária de banda entre os diversos receptores.

- **Tamanho do pacote**: 250 bytes: nas simulações foi utilizado o mesmo tamanho de pacote para todas camadas;
- **Tamanho da fila**: 30 pacotes no *droptail*; 20 pacotes no RED, com *thresh* = 5 e *maxthresh* = 15;
- **Camadas exponenciais**: 6 camadas: 32 kbit/s, 64 kbit/s, 128 kbit/s, 256 kbit/s, 512 kbit/s e 1024 kbit/s.
- **Tempo de simulação**: 700s.

A topologia mostrada na figura 1 foi usada para todas as simulações e seu maior objetivo é analisar o comportamento de sessões ALM concorrentes compartilhando um mesmo enlace de menor velocidade (gargalo). Os nós *T0* a *T15* representam os transmissores ALM (cada um envia um conjunto de camadas diferente); nós *R0* a *R15* representam os receptores ALM, e cada um se conecta a um transmissor específico, assim, o receptor R0 faz *join* nas camadas enviadas pelo nó *T0*, o receptor *R1* faz *join* nas camadas enviadas pelo nó *T1*, e assim por diante.

Os nós *Rot1* e *Rot2* representam os roteadores conectados através do enlace de menor velocidade, que tem uma largura de banda de *n*500kbit/s*, onde *n* é o número de transmissores no momento. Isso foi feito para acomodar 4 camadas para cada sessão (que demandam um total de 480 kbit/s = 32+64+128+256).

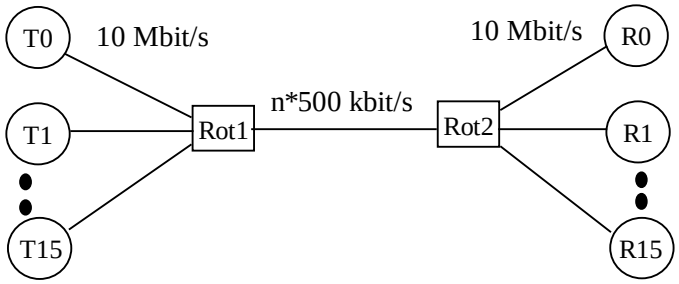

Figura 1. Topologia usada nas simulações.

Foi considerada uma largura de banda de 10 Mbit/s para cada transmissor e receptor, deixando-os com largura de banda de sobra para qualquer número de camadas. O atraso dos enlaces foi fixo em 10ms.

Os experimentos utilizaram tráfego CBR (*Constant Bit Rate*). Os fluxos CBR foram obtidos utilizando uma taxa de transmissão constante para cada camada, mas o escalonamento dos pacotes tinha uma incerteza de 1s, ou seja, havia uma variável randômica entre –0,5 e +0,5s provocando uma pequena rajada nas transmissões. A fórmula para escalonar os novos pacotes era: *t += interval_ * Random::uniform(-0,5, 0,5)*. Isso significa que o novo pacote será escalonado no próximo intervalo mais um tempo randômico variando entre –0,5 e +0,5s.

Uma análise do algoritmo utilizando tráfego VBR (*Variable Bit Rate*), mais precisamente Pareto e Exponencial, foi submetido em outro artigo [14], e os resultados foram bem animadores. Não haveria espaço nesse artigo para colocar os resultados para todos tipos de tráfego testados, portanto, a análise se fixa no modelo CBR descrito no parágrafo anterior.

4 Descrição do algoritmo: ALM (*Adaptive Layered Multicast*)

A fim de alcançar rapidamente a sua fatia de banda e se manter estável após isso, o algoritmo foi dividido em duas fases: *estado inicial* e *regime permanente*. Essas fases serão analisadas a seguir, bem como questões gerais sobre o algoritmo, como o *intervalo de execução*.

O receptor se inscreve em tantas camadas quanto a variável *bwshare* permite, e pode aumentar ou diminuir mais de uma camada por vez se necessário. O valor máximo de *bwshare* que o receptor pode se inscrever é limitado pela **capacidade máxima fim a fim da rede**, obtida através da utilização de pares de pacotes.

O uso de pares de pacotes é explicado com detalhes por Carter [1], e se baseia na transmissão de dois pacotes em seqüência pelo transmissor. O receptor, por sua vez, utiliza os pares de pacotes recebidos para inferir sua banda máxima até o transmissor, e essa inferência se dá através do espaçamento entre os pacotes e do tamanho dos mesmos. O objetivo desse método é fazer com que o receptor não se inscreva em camadas acima da capacidade física da rede. Por exemplo, caso o receptor tenha um canal de 64 kbit/s, de nada adianta se inscrever numa camada que geraria um tráfego acima desse valor, mesmo estando sozinho na rede.

Entretanto, esse método é valioso quando existir apenas uma sessão na rede, provocando uma adaptação e estabilidade perfeita nos receptores. Com mais sessões simultâneas a banda é compartilhada e a nunca chega no máximo, pois o gargalo é dividido entre os receptores das diferentes sessões. Em [14], os testes foram feitos sem o uso do método de pares de pacotes.

4.1 Visão geral do algoritmo

Durante a fase "*estado inicial*", o algoritmo tenta alcançar rapidamente a sua fatia de banda entre os fluxos concorrentes. Para isso ele executa o seguinte código:

```
Se "sem perdas"
  aumenta rapidamente bwshare (até o máximo)
Senão
  divide bwshare por  $factor
  vai para a fase "regime permanente"
```

Assim, o algoritmo vai incrementar rapidamente sua largura de banda disponível até o máximo descoberto pelos pares de pacotes ou até a detecção de perdas – ou informação de congestionamento, se o sistema estiver utilizando ECN (*Explicit Congestion Notification*). Nesse momento, é considerado que a largura de banda passou do ponto ótimo (pois ocorreram perdas), então, o algoritmo divide sua largura de banda por um determinado fator (2, por exemplo) e vai para a segunda fase (*regime permanente*).

Durante a fase de *regime permanente*, o lema do algoritmo para incremento de banda é: "dar mais largura de banda para quem tem menos". Isso significa que, se há dois fluxos competindo dividindo o mesmo ponto de congestionamento (gargalo), o que tiver menos largura de banda vai incrementar mais rapidamente em relação ao outro, e ambos vão tender para um ponto de equilíbrio.

No caso de ocorrerem perdas, deve ser diminuída a banda permitida aos receptores, e o lema fica: "tirar mais largura de banda de quem tem mais". Isso significa que o fluxo com maior largura de banda vai diminuir mais rapidamente em relação ao outro, e ambos também vão tender a um ponto de equilíbrio.

Para a implementação das definições efetuadas acima para a fase de regime permanente, se utiliza uma variável que determina a fatia de banda (*bwshare*) permitida para o receptor num determinado momento. Essa variável muda a cada *intervalo de execução*, de acordo com o seguinte código:

```
Se "sem perdas"
incrementa  lentamente bwshare (até o máximo)
```

Senão
decrementa mais rapidamente bwshare (5%, por exemplo)

As taxas de incremento e decremento são proporcionais à largura de banda atual usada pelo receptor, assim, por exemplo, se um receptor tem 500 kbit/s e outro tem 100 kbit/s, caso ambos sofram perdas de pacotes, eles vão decrementar *bwshare* em 5%. O primeiro vai subtrair 25 kbit/s e o segundo 5 kbit/s, reduzindo a diferença entre eles (de 400 kbit/s para 380 kbit/s). O mesmo acontece quando se está incrementando a largura de banda, mas em menor escala, ou seja, mais lentamente.

4.2 Principais parâmetros

Os principais parâmetros do algoritmo são o *intervalo de execução*, a *taxa de incremento* e a *taxa de decremento* (principalmente na fase de *regime permanente*). Eles são discutidos a seguir.

O *intervalo de execução* é o tempo entre duas execuções consecutivas do algoritmo. Mudando esse parâmetro modifica a velocidade de adaptação do receptor e a estabilidade do sistema. Usando um número pequeno (por exemplo, 100ms), vai fazer o sistema alcançar a sua fatia justa no compartilhamento mais rapidamente, mas não será tão estável como seria com um *intervalo de execução* maior, como, por exemplo, 1s.

A figura 2 mostra graficamente as razões para isso. Na figura, há 5 camadas exponenciais (32 kbit/s, 64 kbit/s, 128 kbit/s, 256 kbit/s, 512 kbit/s) – o eixo dos Y representa o total de banda usada pelo receptor, que é a soma das larguras de banda de cada camada na qual o receptor está inscrito.

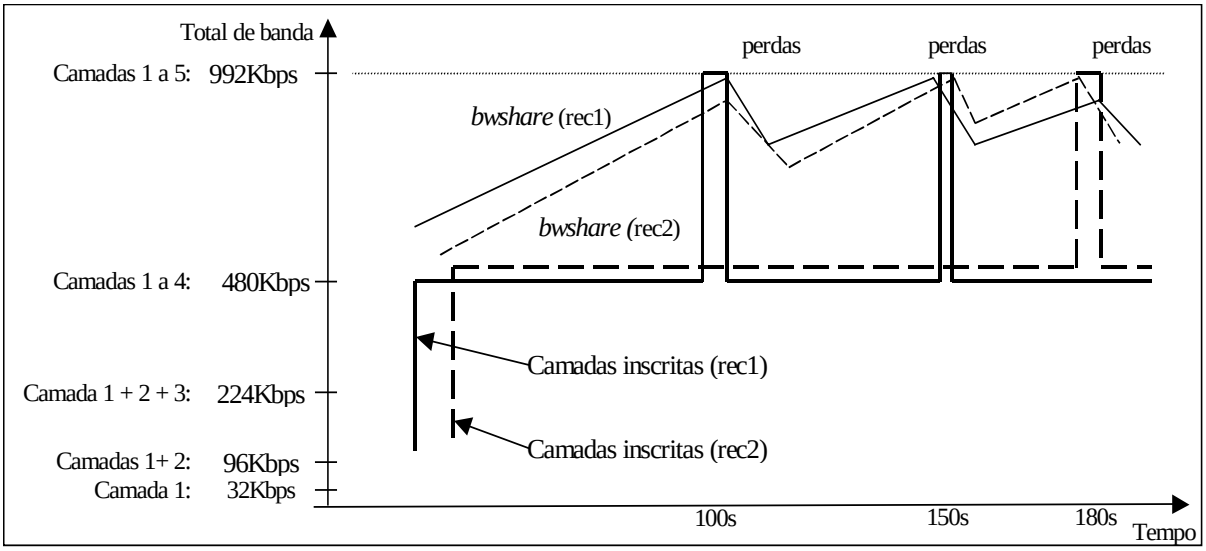

Figura 2. Visão geral do algoritmo em termos de decisão para subir ou descer camadas.

As linhas mais grossas na figura indicam a quantidade de banda que o receptor 1 (representado pela linha contínua) e o receptor 2 (representado pela linha pontilhada) estão utilizando. Essa quantidade é relacionada diretamente com a quantidade de camadas que cada receptor se inscreveu. Pode-se ver que os receptores estão estáveis quando inscritos nas primeiras quatro camadas, e as tentativas de se inscrever na camada de número 5 geraram perdas.

A largura de banda permitida aos receptores incrementa ou decrementa a cada *intervalo de execução*. Isso é representado na figura através das linhas finas, sendo que a linha contínua

representa o receptor 1 e a linha pontilhada o receptor 2. Essas linhas representam os resultados obtidos para a variável *bwshare* em cada *intervalo de execução.*

Caso a largura de banda permita (linhas finas na figura atingindo uma largura de banda suficiente para o receptor se inscrever em uma nova camada), o receptor vai se inscrever em outra camada. Caso essa tentativa gere perdas (ver representação na figura), o receptor vai deixar uma camada.

Como pode ser inferido, se o intervalo de execução é maior, o sistema vai levar mais tempo para causar instabilidade na rede, mas a velocidade de adaptação vai aumentar.

As simulações mostradas neste artigo usaram um *intervalo de execução* de 1s, que foi satisfatório para os objetivos de estabilidade propostos.

Outro parâmetro importante é a *taxa de incremento*. Na fase de *regime permanente*, a taxa de incremento é fixa em $10^9/bwshare$. Este parâmetro está relacionado com a agressividade do protocolo. Como pode ser visto, a *taxa de incremento* é inversamente proporcional ao valor de *bwshare*, ou seja, à medida que a largura de banda do receptor fica maior, ele vai incrementar mais lentamente sua banda. Assim, a tendência é que fluxos diferentes tendam a um ponto de equilíbrio.

O mesmo acontece com a *taxa de decremento*. Na fase de *regime permanente*, a *taxa de decremento* é fixa em 5% da largura de banda atual, assim, à medida que a largura de banda fica maior, o percentual representa um valor maior, causando ao algoritmo um decréscimo maior em relação aos outros fluxos com menor largura de banda.

4.3 Resumo do algoritmo

Resumindo, o algoritmo tem as seguintes características: Inicialmente é executada a sub-rotina *init*, que configura as variáveis e inicia a execução do código a cada IE (Intervalo de Execução). Assim, cada Intervalo de Execução, o seguinte algoritmo é executado:

- Começa na fase *estado inicial*, que incrementa rapidamente a largura de banda permitida ao receptor, tentando alcançar sua fatia justa junto aos tráfegos concorrentes. Nessa fase, com o uso de filas com política de descarte tal como RED ou *droptail*, seria melhor para o receptor se os roteadores descartassem primeiro os pacotes dos outros tráfegos já estabilizados, assim, o receptor poderia alcançar uma fatia de banda razoável. Isso é o mais provável, já que o tráfego dos outros fluxos é maior, possuindo maior número de pacotes passando pelo roteador.
- Quando o receptor sente perda de pacotes, ele passa para a fase de *regime permanente*, onde ele fica até o fim da sessão.
- Na fase de *regime permanente*, as sessões com menos banda incrementam sua fatia de tráfego mais rapidamente que sessões com mais banda, tendendo a um ponto de equilíbrio.
- Na fase de regime permanente, as sessões com mais banda decrementam sua fatia de tráfego mais rapidamente que sessões com mais banda, tendendo a um ponto de equilíbrio.
- O intervalo de execução é um parâmetro relacionado com a estabilidade e velocidade de adaptação do algoritmo. À medida que o intervalo de execução se torna maior, a estabilidade aumenta e a velocidade de adaptação diminui. Este parâmetro deve ser escolhido cuidadosamente dependendo do tipo de tráfego concorrente. Com TCP, este parâmetro deve permanecer baixo (abaixo de 200ms).

A seguir pode-se ver uma parte resumida da implementação no NS. Algumas partes de proteções foram omitidas para focar no núcleo do código.

```
ALM instproc init {levels chk_estimate n_id} {
    ...
   #  configura as variáveis para ALM
   ...
   # chama a sub-rotina a ser executada cada Intervalo de Execução
   set subscription_ 0
   $self add-layer
   $self AlmIE
}

#####################################################
# AlmIE: código a ser executado cada Intervalo de Execução
#####################################################
ALM instproc AlmIE {} {
# escalonador para repetir sub-rotina cada Intervalo de Execução
  set ns_time [$ns_ now]
  $ns_ at [expr $ns_time + $ALM_IE] "$self AlmIE"

# Cada chegada de par de pacote chama uma subrotina que calcula a banda
# máxima descoberta pelo par de pacotes. A proteção a seguir é para não
# executar o código antes da chegada do primeiro par de pacotes.
  if {$bwmax==0} { ;# protecao no inicio dos tempos
    return
  }

  set bwatual [lindex $rates_cum [expr $subscription_-1]] ;# banda total consumida
  set bwacima [lindex $rates_cum $subscription_] ;# banda necessária para subir camada

  switch [lindex $cngopt $cngmode] {
    "SlowStart" { ;# começa em SlowStart
      if {$numloss==0} {
        set cngwndmax [expr $bwmax]
       # aumenta rapidamente bwshare (quase dobra banda cada IE)
        set cngwnd [expr $cngwnd + $cngwnd / $rss]
       # proteção para não passar da banda máxima descoberta pelos pares de pacotes
        if {$cngwnd > $cngwndmax} {
          set cngwnd $cngwndmax
        }
      } else { ;# ocorreram perdas – hora de passar para “CngAvoid”
        set cngwnd [expr $cngwnd/2] ;# reduz pela metade e passa para cngavoid.
        set cngmode 1
      }
# fase de regime permanente
    "CngAvoid" {
      if {$numloss==0} { ;# sem perdas
        # quanto maior a banda menos aumenta. fincr é referência(valor fixo) – estudar melhor
        set cngwnd [expr $cngwnd + ($fincr/$cngwnd)]
      } else {
        set cngwnd [expr $cngwnd - ($cngwnd * 0.05)] ;# desce 5% cada perda.
      }
     # para estabilização. Quando desce camada espera um IE
    ...
    }
  set novabw $cngwnd
```

```
# Trata se deve subir camada
  if {$novabw > $bwacima} {
    # cria um numero aleatório de IEs de espera para subir:
    # isso evita que varias sessões subam ao mesmo tempo
    ...
    set r [$self ALM_add-layer $novabw] ;# sobe camada
    if {$r == -1} { ;# proteção caso não tenha conseguido subir camada
      set cngwnd $cngwndant
    }

# Trata se deve descer camada
  if {$subscription_ <= 1} { ;# não tem mais o que descer
    return
  }

  while {$novabw < $bwatual} { ;# desce camada
    $layer_($subscription_) leave-group
    incr subscription_ -1
    if {$subscription_ <= 1} { return } ;# não tem mais o que descer

    set bwatual [lindex $rates_cum [expr $subscription_-1]]
    # sempre que desce camada deixa a banda na metade entre as camadas.
    set bwacima [lindex $rates_cum [expr $subscription_]]
    set cngwnd [expr (($bwatual+$bwacima) / 2)] ;# banda no meio das camadas
  }
}
```

5 Resultados das simulações

O objetivo desta sessão é mostrar os resultados obtidos através das simulações, utilizando o ambiente de avaliação descrito anteriormente. O número de transmissores variou de 1 a 16, procurando analisar o impacto dessa variação na estabilidade. O gargalo da topologia de rede simulada foi de *n*500 kbit/s*, onde *n* é o número de transmissores.

Havia exatamente um receptor por transmissor, já que o objetivo foi analisar a estabilidade e não a adaptação em ambientes heterogêneos, pois isso já foi feito em [9]. Todos os experimentos foram repetidos para política de descarte nas filas dos roteadores RED e *droptail*, entretanto, alguns não constam neste artigo, pois os resultados foram muito similares. Quando os resultados forem relevantes, eles serão analisados.

A fim de fazer os gráficos mais fáceis de ler e interpretar, se utilizou a mesma idéia descrita por Gopalakrishnan em [3], ou seja, a camada a que pertence o fluxo é desenhada no gráfico com um deslocamento de 0,06 vezes o número da sessão. Por exemplo, o receptor ALM3 inscrito na camada 4 seria desenhado em 4,18 ao invés de 4,0.

A figura 3 mostra o gráfico para um único fluxo ALM passando através de um gargalo de 500 kbit/s. Pode ser visto que a estabilidade obtida é total, e isso acontece devido ao limite imposto ao receptor quanto à máxima largura de banda disponível no sistema – descoberto através do uso dos pares de pacotes.

Em termos de número de bytes transferidos, mediu-se que o receptor obteve 41,3105 Mbytes em 700s. Esse valor será utilizado como referência mais tarde no artigo.

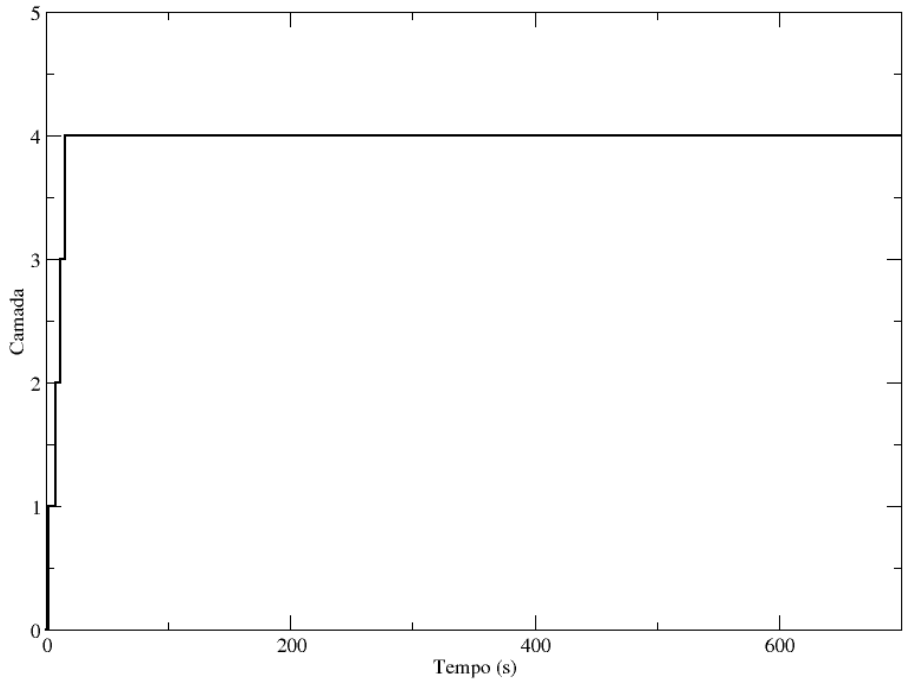

Figura 3. Um único fluxo ALM através de 500 kbit/s. Comportamento igual para RED e *droptail*.

A figura 4 mostra dois fluxos ALM competindo por 1Mbit/s. Como pode ser visto, com RED o sistema se adaptou perfeitamente e com *droptail* ocorreram dois *join*s (nos instantes 268s e 518s) que geraram perdas e tiveram a duração de 1 segundo. Esse tipo de instabilidade pode ser retirado pela aplicação, como explicado anteriormente, assim, o usuário não iria perceber qualquer alteração de banda.

Em termos de imparcialidade, os receptores obtiveram 41,2935 Mbytes e 41,29025 Mbytes usando filas do tipo RED, e 41,33925 Mbytes e 41,33075 Mbytes usando filas *droptail*. Esse resultado foi considerado muito bom para os objetivos propostos pelo algoritmo.

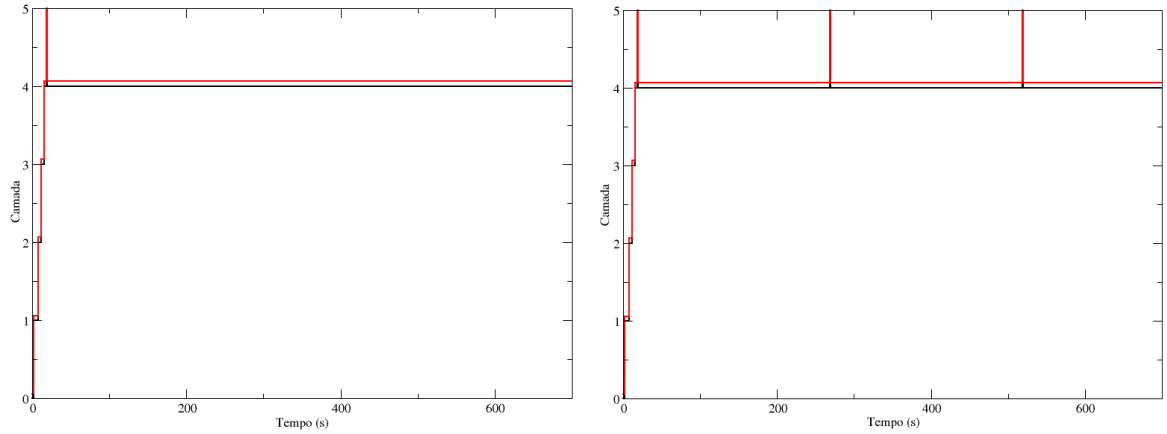

Figura 4. Dois fluxos ALM. 1Mbit/s no gargalo. RED e *droptail*.

A figura 5 mostra quatro fluxos ALM competindo por 2Mbit/s. Como pode ser visto, o sistema mostrou alguma instabilidade com o quarto fluxo ALM em filas do tipo RED. Isso acontece pois a política de descarte RED não deixa a fila atingir uma situação de congestionamento, descartando os pacotes antes que a fila fique cheia. Na simulação, o quarto

fluxo foi o escolhido para ser descartado, e seu usuário sofreu a conseqüência, que foram quatro instabilidades em 700 segundos, ou uma instabilidade a cada três minutos, aproximadamente.

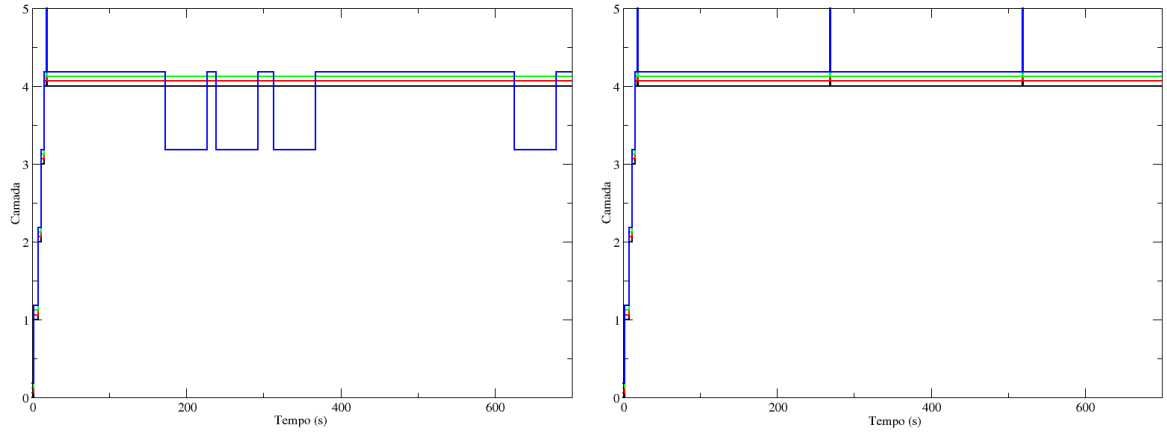

Figura 5. Quatro fluxos ALM. 2Mbit/s no gargalo. RED e *droptail*.

Em termos de imparcialidade, os receptores receberam 41,21175, 41,28775, 41,3175 e 34,3035 Mbytes usando filas do tipo RED. O último fluxo recebeu menos largura de banda, o que está de acordo com o gráfico. Usando filas do tipo *droptail*, os receptores obtiveram um total de 41,3335, 41,32575, 41,30475 e 41,2823 Mbytes respectivamente, o que foi mais imparcial que utilizando RED, como pode ser comprovado pelo gráfico na figura 5.

Investigando um pouco mais profundamente a instabilidade que aconteceu com RED, foi visto que o tamanho da fila estava configurado em 20 pacotes, onde a variável *thresh* = 5 e *maxthresh* = 15. Como o tamanho de pacote usado foi relativamente pequeno (250 bytes), nós tentamos aumentar esse tamanho (para 500 bytes) e ver as conseqüências. O comportamento esperado era um aumento na estabilidade, pois a fila pode comportar mais bytes. Os resultados comprovaram a teoria, como pode ser visto na figura 6, onde o segundo fluxo foi afetado, mas com apenas uma instabilidade em 700 segundos (o que é mais baixa frente às 4 instabilidades vistas na figura anterior).

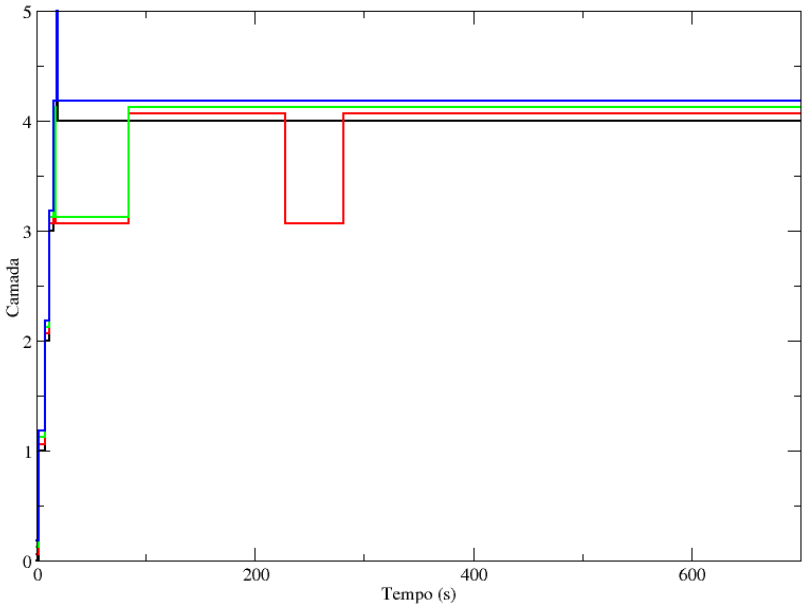

Figura 6.Quatro fluxos ALM – RED com tamanho de pacote de 500 bytes.

Deste ponto em diante, os resultados de imparcialidade serão suprimidos, pois eles são praticamente os mesmos vistos acima e podem ser inferidos através dos gráficos.

A figura 7 mostra oito fluxos ALM competindo por 4Mbit/s no gargalo. Como pode ser visto, o sistema mostrou novamente alguma instabilidade para três fluxos utilizando política de descarte do tipo RED. Usando *droptail*, a imparcialidade foi completa, e o usuário não toma consciência da concorrência. A estabilidade com RED pode ser melhorada através do incremento do tamanho de pacote para 500 bytes, como visto anteriormente.

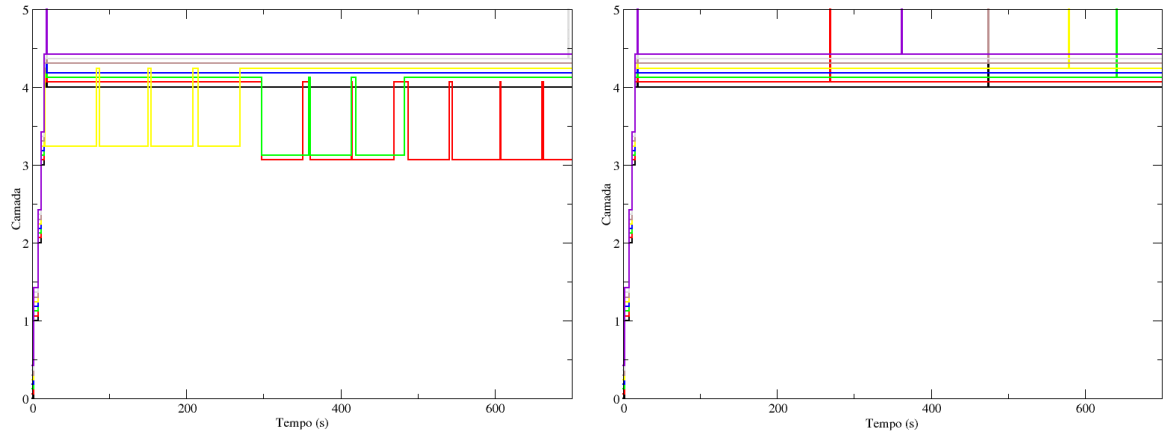

Figura 7. Oito fluxos ALM. 4Mbit/s no gargalo. RED e *droptail*.

A figura 8 mostra dezesseis fluxos ALM competindo por 8 Mbit/s. Como pode ser visto, o sistema mostrou novamente alguma instabilidade utilizando a política de descarte do tipo RED. Essa instabilidade pode ser minimizada aumentando o tamanho do pacote para 500 bytes, como visto anteriormente. O gráfico é confuso, mas analisando o arquivo "*dat*" gerado (usado para criar o gráfico), foi descoberto que 6 fluxos permaneceram completamente estáveis, o fluxo com maior variação teve 8 variações, e na média houve 2,5 variações por fluxo em 700 segundos.

Com política de descarte de filas do tipo *droptail*, os resultados foram bastante satisfatórios, e o usuário não perceberia quaisquer mudanças de qualidade na imagem recebida (com exceção das causadas por perda de pacotes).

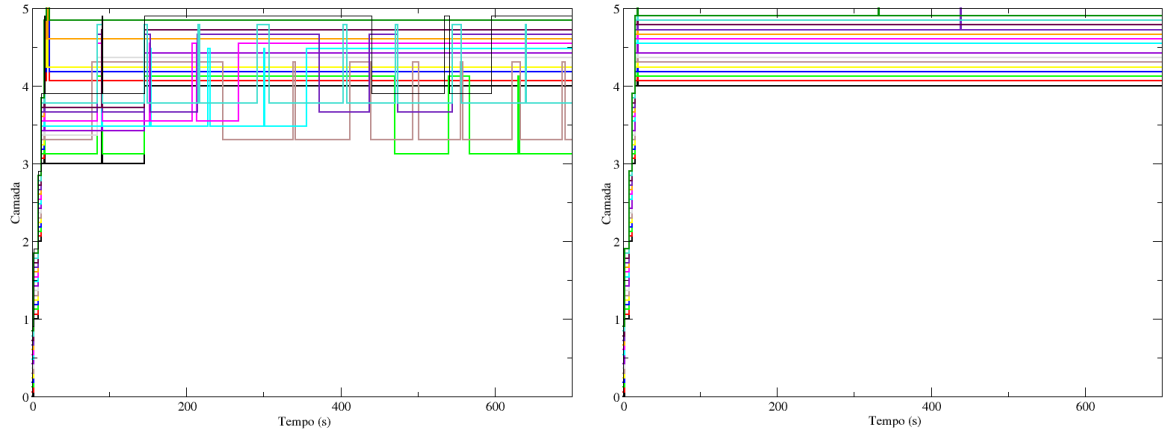

Figura 8. Dezesseis fluxos ALM. 8Mbit/s no gargalo. RED e *droptail*.

Algumas conclusões gerais sobre esses resultados podem ser vistas no item 7 "Conclusões".

6 Trabalhos relacionados

O objetivo desta sessão é comparar alguns trabalhos existentes que utilizam transmissão multicast em camadas com o ALM, a fim de deixar mais claro a contribuição fornecida por este algoritmo.

O protocolo proposto por McCanne, denominado RLM (*Receiver-driven Layered Multicast*) [5], descreve uma abordagem para transmissão de multicast em camadas com o objetivo de enviar sinais multimídia em ambientes heterogêneos. Nessa abordagem, a responsabilidade de adaptação ao número correto de camadas é do receptor (tal qual o ALM). Quando um novo receptor entra no sistema, ele se inscreve no grupo multicast mais básico, e tenta se associar com outros grupos, monitorando a perda de pacotes (que indicam congestionamento). No caso de congestionamento, o receptor se adapta, se desinscrevendo da camada mais alta. De tempos em tempos, o receptor tenta subir um nível (através de um *join* na camada superior), a fim de melhorar a recepção. No caso de detecção de perda de pacotes, ele se desinscreve dessa camada, voltando à camada que estava anteriormente. O intervalo entre tentativas de *join* aumenta à medida que o tempo passa, a fim de não gerar congestionamentos repetidamente em uma rede.

Uma das diferenças entre o RLM e o ALM é que o RLM faz tentativas de *join* regularmente para qualquer tipo de rede, independente da topologia. Isso causa instabilidade e gera tráfego desnecessário. No algoritmo do ALM, é utilizada uma abordagem de pares de pacotes a fim de inferir a largura de banda máxima fim a fim da rede, assim, se a topologia não permite o incremento de camada, o algoritmo vai permanecer estabilizado, como pode ser visto na figura 3.

Outra diferença é que o RLM diminui a camada mais alta em caso de detecção de perdas, causando um efeito de injustiça caso o algoritmo esteja rodando na Internet com tráfego concorrente TCP. Isso acontece pois o TCP é agressivo, gerando perdas e causando um contínuo decréscimo nas camadas do RLM, até o mínimo permitido, e fazendo com que o TCP fique com toda a banda disponível. Com o ALM, isso não acontece, pois os parâmetros podem ser ajustados para serem imparciais com tráfego TCP concorrente.

Algumas patologias do RLM foram descobertas por Legout [10], e uma delas é que a velocidade de adaptação do RLM é muito lenta. O protocolo ALM consegue se adaptar muito mais rapidamente devido à fase de “estado inicial”, vista na sessão 4.

Outro trabalho relacionado é o RLC (*Receiver-driven Layered Congestion Control*), de Vicisano [11], cujo protocolo teve por objetivo melhorar o algoritmo do RLM limitando a fase de *join* para alguns pontos de sincronização, onde o transmissor envia o dobro dos pacotes em todas camadas. Os pontos de sincronização são usados para testar a qualidade do enlace, indicando ao receptor (através de perdas) se ele pode ou não se inscrever em uma nova camada. O problema é que aparentemente não foi considerada a elasticidade das filas dos roteadores, que absorvem em grande parte a rajada enviada. Isso faz com que o algoritmo não funcione adequadamente [10].

Existem outros trabalhos relacionados com adaptação no lado do receptor, tal como o RSLP (*Receiver-Selectable Loss Priorities*) de Gopalakrishnan [4] e o PLM (*Packet-pair receiver-driven layered multicast*) de Legout [12]. Ambos são bastante interessantes, mas necessitam modificações nos roteadores intermediários, tais como o uso de filas WF2Q (*worst-case fair weighted fair queuing*) [13] ou descarte por prioridades.

O protocolo ALM também se beneficiaria do uso de uma fila com descarte por prioridades, mas o objetivo deste artigo foi mostrar seu comportamento na Internet dos dias de hoje, sem qualquer tipo de QoS.

7 Conclusões

O objetivo deste artigo foi mostrar questões de estabilidade e imparcialidade em um novo algoritmo para transmissão de multicast em camadas através de linhas privativas (com concorrência de seu próprio tráfego) usando filas FIFO nos roteadores, com política de descarte RED e *droptail*.

O algoritmo foi visto de forma detalhada, e ficou claro que ele possui três parâmetros principais: *intervalo de execução*, *taxa de incremento* e *taxa de decremento*. Modificando esses parâmetros modifica sua agressividade, podendo torná-lo imparcial inclusive com tráfego TCP, mas diminuindo a estabilidade. Esses três parâmetros em conjunto são responsáveis pela estabilidade do algoritmo, bem como sua velocidade de adaptação.

Os resultados mostraram que o algoritmo é totalmente estável sozinho (um único fluxo), e isso é facilmente explicável, pois o sistema nunca vai passar da máxima largura de banda fim a fim (pois a banda máxima devido à topologia é descoberta através da utilização dos pares de pacotes). Entretanto, com mais transmissores competindo pela banda, o algoritmo mostrou uma certa instabilidade, principalmente com política de descarte do tipo RED. Isso acontece pois esse tipo de fila descarta mais pacotes em relação ao *droptail*. Usando *droptail*, entretanto, os resultados ficaram acima das expectativas iniciais.

Foi visto que o tamanho do pacote escolhido para as simulações foi relativamente pequeno em relação ao tamanho da fila, e as simulações com RED poderiam ser melhoradas através da utilização de um tamanho de pacote maior (ou uma fila maior). Isso é importante, pois mostra claramente que incrementando o tamanho do pacote ou o tamanho da fila também incrementa a estabilidade com RED – mas certamente aumenta também a latência fim a fim.

Nós ainda estamos explorando o número de pacotes descartados em cada camada (taxa de perdas de cada transmissão de vídeo). Isso deve dar algumas idéias a respeito da viabilidade de transmitir sinais multimídia em camadas sem o uso de QoS (vai depender da habilidade do decodificador em se recobrar de perdas). As perdas sem QoS serão comparadas com o comportamento do ALM com a utilização de QoS.

Outro ponto a ser estudado com maior profundidade é a influência das instabilidades na aplicação, ou seja, o quanto de instabilidade pode ser absorvido pela aplicação mantendo uma apresentação agradável ao usuário.

A escolha dos principais parâmetros também é um fator a ser considerado, pois todos os receptores devem utilizar os mesmos parâmetros, caso contrário a divisão de tráfego não será justa entre os diferentes fluxos existentes.

8 Referências

[1] CARTER, Robert. CROVELLA, Mark. “Measuring Bottleneck link speed in packet switched networks”. March 1996.

[2] VINT project – Virtual InterNetwork Testbed. In http://www.isi.edu/nsnam/vint (08/06/2001).

[3] GOPALAKRISHNAN, R. GRIFFIOEN, J. HJALMTYSSON, G. SREENAN, C. “Stability and Fairness Issues in Layered Multicast”. In: Network and Operating Systems Support for Digital Audio and Video, NOSSDAV '99. Proceedings… June 1999.

[4] GOPALAKRISHNAN, R. et al. "A simple loss differentiation approach to layered multicast". In: IEEE INFOCOM, 2000. Proceedings… Tel Aviv, ISRAEL, March 2000.

[5] McCANNE, S., JACOBSON, V., VETTERLI, M. “Receiver driven layered multicast”. In: ACM SIGCOMM 96. Proceedings… Stanford, CA, pp 117-130, August 1996.

[6] ROESLER, V. BRUNO, G. LIMA, V. "ALM - Adaptive Layering Multicast". In: SIMPOSIO BRASILEIRO DE MULTIMIDIA, SBMIDIA, 2001. Proceedings... Florianopolis: Universidade Federal de Santa Catarina, Brazil. October 2001.

[7] WU, D., HOU, T., ZHANG, Y. “Transporting Real-Time Video over the Internet: Challenges and Approaches”. In: Proceedings of the IEEE, V. 88, N. 12. December 2000.

[8] FLOYD, Sally. JACOBSON, Van. “Random Early Detection Gateways for Congestion Avoidance”. IEEE/ACM Transactions on Networking. 1993.

[9] ROESLER, V. LIMA, V. "A receiver-driven layered multicast congestion control algorithm for multimedia transmissions over droptail and RED queues". Submitted to: Networking 2002, Pisa, Italy. May 2002.

[10] LEGOUT, A. BIERSACK, W. "Pathological Behaviors for RLM and RLC". In: Network and Operating Systems Support for Digital Audio and Video, NOSSDAV’2000. Proceedings… Chapel Hill, North Carolina, pp. 164-172, June 2000.

[11] VICISANO, L., RIZZO, L., CROWCROFT, J. “*TCP-like congestion control for layered multicast data transfer*”. In: IEEE INFOCOM, 1998. Proceedings… March 1998.

[12] LEGOUT, A. BIERSACK, W. "*PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes*". In: ACM SIGMETRICS 2000, XVII, Santa Clara, California, June 2000.

[13] BENNETT, J. ZHANG, H. “*WF2Q: Worst-case Fair Weighted Fair Queueing*”. In: INFOCOM '96. Proceedings… San Francisco, California, March 1996.

[14] ROESLER, V. BRUNO, G. LIMA, V. “*Stability issues using CBR, Exponential and Pareto traffic in a new algorithm for layered multicast transmissions*”. Submitted to: Network and Operating Systems Support for Digital Audio and Video, NOSSDAV’2002, Miami, USA, May, 2002.