UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE TECNOLOGIA

Simulação e Emulação de Tráfego Multimídia em Redes IP

Fernando Luiz Pinotti

Limeira
2011

08/2011

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE TECNOLOGIA

Simulação e Emulação de Tráfego Multimídia em Redes IP

Fernando Luiz Pinotti

Orientador: Prof. Dr. Varese Salvador Timóteo

Dissertação apresentada à Faculdade de Tecnologia da Universidade Estadual de Campinas, como parte dos requisitos para a obtenção do grau de Mestre em Tecnologia e Inovação.

Limeira
2011

FICHA CATALOGRÁFICA ELABORADA POR SILVANA MOREIRA DA SILVA SOARES – CRB-8/3965

BIBLIOTECA UNIFICADA FT/CTL

UNICAMP

Pinotti, Fernando Luiz, 1987-

P656s Simulação e emulação de tráfego multimídia em redes IP / Fernando Luiz Pinotti. – Limeira, SP : [s.n.], 2011.

Orientador: Varese Salvador Timóteo.

Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Tecnologia.

1. Comunicações multimídia. 2. Modelagem de tráfego. 3. Redes de computação - Protocolos. I. Timóteo, Varese Salvador. II. Universidade Estadual de Campinas, Faculdade de Tecnologia. III. Título. .

Título em inglês: Simulation and emulation of multimedia traffic over IP network

Palavras-chave em inglês (Keywords):

1- Multimedia communication

2- Traffic models

3- Communication protocols

Área de concentração: Tecnologia e Inovação

Titulação: Mestre em Tecnologia

Banca examinadora: Varese Salvador Timóteo, Edson Luiz Ursini, Gélio Mendes Ferreira

Data da Defesa: 01-09-2011

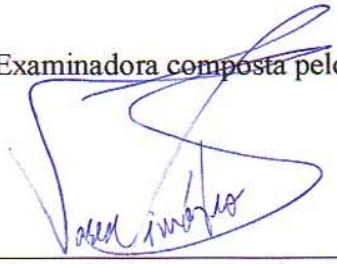
Programa de Pós-Graduação em Tecnologia

DISSERTAÇÃO DE MESTRADO ACADÊMICO

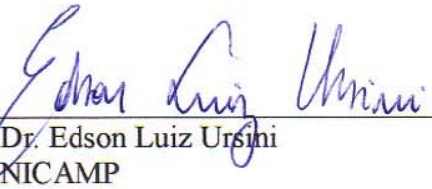
Simulação e emulação de tráfego multimídia em redes IP

Autor: Fernando Luiz Pinotti

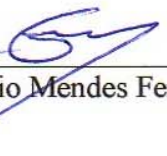
A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:



Prof. Dr. Varese Salvador Timóteo, Presidente
FT/UNICAMP



Prof. Dr. Edson Luiz Ursini
FT/UNICAMP



Prof. Dr. Gélio Mendes Ferreira
UFABC

Dedicatória

Ao meu querido pai, que independente de onde, estaremos sempre juntos.

Agradecimentos

Primeiramente gostaria de agradecer ao meu orientador, Professor Doutor Varese Salvador Timóteo, que seu apoio foi de enorme importância nas decisões e realizações dos percursos desta dissertação, e, principalmente o seu apoio no momento mais difícil que a vida nos faz passar.

Ao Professor Doutor Edson Luiz Ursini pelo seu vasto conhecimento e experiencia, principalmente na utilização do ARENA.

Um agradecimento especial a toda minha família, por ter me dado a oportunidade de chegar até aqui.

Aos meus amigos de faculdade e colegial pelo apoio e motivação.

Ao Programa Santander, por ter me dado a oportunidade financeira de ter o privilegio de estudar na Universidad Politécnica de Madrid.

À CAPES, pela bolsa de estudo, por proporcionar condições para que essa dissertação fosse realizada, e a UNICAMP, que sem ela nada seria possível.

RESUMO

Esta dissertação apresenta dois estudos, o primeiro trata de uma substituição do *socket* pela pilha TCP/IP uIP de um gerador de tráfego multimídia sobre IP com o objetivo de manipularmos os campos dos cabeçalhos IP (Ipv4 *Type of Service,* e Ipv6 *Traffic Class)*, que serão utilizados para a identificação dos serviços multimídia. O gerador gera diferentes tráfegos multimídia simultâneos seguindo funções de distribuições conhecidas utilizando o conceito de *thread*. O segundo estudo trata de modelos de simulações utilizando o simulador de eventos discretos ARENA. Propomos três diferentes modelos de simulação que simulam ambientes multimídia, onde serviços *stream* e elástico são requisitados simultaneamente pelos usuários. No modelo 1 foram realizados simulações propondo um estudo de demanda de requisições dos usuários. Onde, três cenários foram estudados. O primeiro, quando ocorre um aumento repentino de usuários, com isso aumentando o intervalo de requisições dos serviços. O segundo cenário, quando a duração dos serviços aumentam ocorrendo bloqueios de novos serviços por falta de recursos. E o terceiro cenário é com relação ao Controle de Admissão de Chamada (CAC) do sistema. Os modelos 2 e 3 são ambientes VPN, onde é estudado o *Sojourn*, tempo que o pacote leva para chegar ao *host* de destino. A principal diferença entre os dois modelos é que o modelo 2 não apresenta a implementação de atributos como *jitter* e latência. O *jitter* e latência influenciam no tempo que leva para o quadro ser entregue ao seu destino, podendo causar diversos problemas nos serviços *stream*, como degradação da qualidade do serviço ou bloqueio do serviço. O estudo de modelos de simulações é de grande importância para a validação dos resultados obtidos na emulação do gerador de tráfegos multimídia, tendo em conta que na simulação os pacotes não são transmitidos através de um meio físico, e no emulador são transmitidos de um *host* a outro utilizando Ethernet.

Palavras-chaves: Modelos de tráfego, Comunicação Multimídia, Protocolos de Comunicação.

ABSTRACT

This dissertation presents two studies,the first is a replacement of a socket for a TCP/IP stack uIP of a multimedia traffic generator over IP in order to manipulate the fields of IP headers (IPv4 Type of Service and IPv6 Traffic Class), which will be used for the multimedia services identification. The multimedia traffic generator generates simultaneous multimedia services following different well-known distributions functions using the concept of thread. A second study of simulation models is propose using the discrete event simulator ARENA, we propose three different simulation model witch simulates multimedia environments, where stream and elastics services are required simultaneous by users. The model 1 were performed simulations proposing a request demand study from the users. Where three scenarios were studied. The first when occurs a sudden increase of users, thereby increasing the range of requests of services. The second scenario when the length of services increases, occurring blockages of new services due the lack of resources, for example, trunks in a telephony exchange. The third scenario is related to Call Admission Control (CAC) system. Models 2 and 3 are VPN environments, where is studied the sojourn, time that the packet takes to reach the destination. The main differences between the two models is that the model 2 does not preset the attributes implementation jitter and latency. The jitter and latency affects the time it takes the frame to be delivered to its destination, may cause various problems in streaming services, such as quality of service degradation or blocking the service. The study of simulation models are of great importance for the emulation's results validations of the multimedia traffic generator, taking into account that the simulation packets are not transmitted over a physical structure, and the emulation are transmitted from one host to another using Ethernet LAN.

Key-words: Traffic Models , Multimedia Communication, Communication Protocols

LISTA DE FIGURAS

Figura 1: Perfil típico de um tráfego de blocos.
Figura 2: Perfil típico de um tráfego de transação.
Figura 3: Perfil típico de um tráfego *stream*.
Figura 4: Relação entre tráfego elástico e tráfego *stream*.
Figura 5: Reservas dos serviços.
Figura 6: Representação esquemática da Máquina de Estado do Gerador.
Figura 7: Segmento TCP, [24].
Figura 8: Datagrama UDP, [24].
Figura 9: Laço principal de checagem de pacotes.
Figura 10: Diagrama de blocos dos processos de transmissão de dados ao receber de um novo pacote.
Figura 11: Diagrama de blocos dos processos de envio de um novo pacote.
Figura 12: Diagrama de fluxo com as principais características do uIP.
Figura 13: Tráfego entre as interfaces. TUN/TAP e eth0 através da ponte br0.
Figura 14: *Workspace* do ARENA. Modelo de simulação 1.
Figura 15: Quantidade de videoconferências em função do intervalo médio entre requisições. A linha preta representa o número total de videoconferências, a azul as concluídas, e a vermelha as perdas. A duração média da videoconferência foi fixada em 900 segundos e CAC igual a 2.
Figura 16: Quantidade de vídeos *on demand* em função do intervalo médio entre requisições. A linha preta representa o número total de vídeos *on demand*, a azul as concluídas, e a vermelha as perdas. A duração média do vídeo *on demand* foi fixada em 7200 segundos com desvio padrão de 900 segundos, utilizando função de distribuição normal e CAC igual a 2.
Figura 17: Trecho destacado na Fig. 16. Comportamento do vídeo *on demand* entre o intervalo 3600 a 21600 segundos.
Figura 18: Quantidade de vídeo *clips* em função do intervalo médio entre requisições. A coluna preta representa o número total de vídeo *clips*, a azul as concluídas, e a vermelha as perdas. A duração média do vídeo *clips* foi fixada em 300 segundos e desvio padrão de 180 segundos, utilizando a distribuição normal e CAC igual a 4.
Figura 19: Quantidade de videoconferências em função da duração média da videoconferência. A linha preta representa o número total de videoconferências, a azul as concluídas, e vermelha as perdas. O intervalo médio entre requisições da videoconferência foi fixado em 3600 segundos e CAC igual a 2.
Figura 20: Quantidade de vídeos *on demand* em função da duração média do vídeo *on demand*. A linha preta representa o número total de vídeos *on demand*, a azul as concluídas, e a vermelha as perdas. O intervalo entre requisições do serviço foi fixado em 14400 segundos e CAC igual a 2.
Figura 21: Quantidade de vídeo *clips* em função da duração média do vídeo *clips*. A linha preta representa o número total de vídeo *clips*, a azul as concluídas, e a vermelha as perdas. O intervalo entre requisições do serviço foi fixado em 900 segundos, e CAC igual a 4.
Figura 22: Quantidade de videoconferências em função ao número máximo de videoconferência simultâneos. A linha preta representa o número total de videoconferência, a azul as concluídas, e a vermelha as perdas. A duração média e intervalo entre requisições dos serviços são 30000 e 3600 segundos respectivamente. Um valor exagerado de duração de serviço foi utilizado para podermos visualizar o efeito das mudanças no número máximo de serviços simultâneos.
Figura 23: Quantidade de vídeos *on demand* em função ao número máximo de vídeo *on demand* simultâneos. A linha preta representa o número total de vídeos *on demand*, a azul as concluídas, e a vermelha as perdas. A duração média e intervalo entre requisições dos serviços são 7200 e 14400 segundos respectivamente.

Figura 24: Quantidade de vídeo *clips* em função do número máximo de vídeo *clips* simultâneos. A linha preta representa o número total de vídeo *clips*, a azul as concluídas, e a vermelha as perdas. A duração média e intervalo entre requisições dos serviços são 300 e 900 segundos respectivamente.
Figura 25: *Workspace* do ARENA. Modelo de simulação 2.
Figura 26: Tempo de *Sojourn* do serviço de Dados, em segundos, em função a variação do volume de tráfego VoIP, em Erlang. A linha preta o serviço de Dados está com prioridade e a linha azul sem, variando de 5 em 5 Erlang o volume de tráfego do serviço VoIP.
Figura 27: *Trace Router* do site www.uol.com.br..
Figura 28: *Trace Router* do site www.b1.rs.
Figura 29: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos A entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição exponencial..
Figura 30: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos B entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição exponencial.
Figura 31: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos A entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição hiper-exponencial.
Figura 32: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos B entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição hiper-exponencial.
Figura 33: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos A entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição Pareto.
Figura 34: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos B entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição Pareto.
Figura 35: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos A entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição Pareto Truncada.
Figura 36: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos B entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição Pareto Truncada.
Figura 37: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos A entre a distribuição exponencial e hiper-exponencial. Os valores de *jitter*, latência e perda são os da Tab. 17.
Figura 38: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos B entre a distribuição exponencial e hiper-exponencial. Valores de *jitter*, latência e perda da Tab. 17.
Figura 39: Tela principal do ARENA.
Figura 40: Parâmetros do bloco *ARRIVE.*
Figura 41: Parâmetros do bloco *ENTER.*
Figura 42: Parâmetros do bloco *CHOOSE.*
Figura 43: Configurando *Conditions*.
Figura 44: Parâmetros do bloco *ASSIGN.*
Figura 45: Configurando Associações.
Figura 46: Parâmetros do bloco *PROCESS.*
Figura 47: Parâmetros *queue* do bloco *PROCESS.*

Figura 48: Parâmetros do bloco *DEPART.*
Figura 49: Parâmetros do bloco *LEAVE.*
Figura 50: Parâmetros do bloco *Variables.*
Figura 51: Parâmetros do bloco Simulate.

LISTA DE TABELAS

Tabela 1: Principais serviços multimídia e suas respectivas categorias de tráfego.
Tabela 2: Vazão típica das aplicações.
Tabela 3: Prioridades para tráfego multimídia.
Tabela 4: Características dos serviços gerados na simulação do Arena.
Tabela 5: Características dos serviços gerados na emulação do Gerador de Tráfego Multimídia.
Tabela 6: Valor dos parâmetros iniciais utilizados para a simulação.
Tabela 7: Parâmetros de intervalo, duração e tráfego dos serviços.
Tabela 8: Saídas do serviço de Dados com prioridade e sem prioridade. Sem prioridade (**SemP) t**odos os serviços estão utilizando a mesma prioridade. Com prioridade (**ComP)** o serviço de Dados possui uma prioridade inferior aos demais.
Tabela 9: Número máximo de serviços simultâneo. O serviço câmera IP possui um asterisco porque é um serviço contínuo, não envolvendo parâmetros como CAC e prioridade. E o serviço de Dados, é influenciado pelo volume do tráfego *stream*.
Tabela 10: Resultados das 10 replicações e a media de cada serviço. Todos os serviços estão utilizando a mesma prioridade. As Saídas *Tipo*_Ta representam o tempo de *sojourn* dos serviços, dado em segundos. A saída *obs Tipo*_Ta são a quantidade total de serviços gerados.
Tabela 11: Resultados das 10 replicações e a media de cada serviço. O serviço de Dados está com uma prioridade inferior aos demais serviços. As Saídas *Tipo*_Ta representam o tempo de *sojourn* dos serviços, dado em segundos. A saída *obs Tipo*_Ta são a quantidade total de serviço gerado.
Tabela 12: Comparação do *sojourn* nos casos A,B e C do serviço de Dados, distribuição exponencial. Utilizando 70 Erlang.
Tabela 13: Comparação do *sojourn* nos casos A,B e C do serviço de Dados, distribuição hiper-exponencial. Utilizando 70 Erlang.
Tabela 14: Comparação do *sojourn* nos casos A,B e C do serviço de Dados, distribuição Pareto. Utilizando 70 Erlang.
Tabela 15: Valores dos atributos *Jitter* e Atraso em segundos e Perda em porcentagem, dos seis serviços do modelo.
Tabela 16: Valores dos atributos *Jitter* e Atraso em segundos e Perda em porcentagem, dos seis serviços do modelo.
Tabela 17: Valores dos atributos *Jitter* e Atraso em segundos e Perda em porcentagem, dos seis serviços do modelo.
Tabela 18: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.
Tabela 19: Tempo *Sojourn* do serviço de Dados, P1 sem prioridade, P2 com prioridade.
Tabela 20: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.
Tabela 21: Tempo *Sojourn* do serviço de Dados, P1sem prioridade, P2 com prioridade.
Tabela 22: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.
Tabela 23: Tempo *Sojourn* do serviço de Dados, P1 sem prioridade, P2 com prioridade.
Tabela 24: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.
Tabela 25: Tempo *Sojourn* do serviço de Dados, P1 sem prioridade, P2 com prioridade.
Tabela 26: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.
Tabela 27: Tempo *Sojourn* do serviço de Dados, P1 sem prioridade, P2 com prioridade.
Tabela 28: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.
Tabela 29: Tempo *Sojourn* do serviço de Dados, P1 sem prioridade, P2 com prioridade.
Tabela 30: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.

Tabela 31: Tempo *Sojourn* do serviço de Dados, P1 sem prioridade, P2 com prioridade.
Tabela 32: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.
Tabela 33: Tempo *Sojourn* do serviço de Dados, P1 sem prioridade, P2 com prioridade.
Tabela 34: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.
Tabela 35: Tempo *Sojourn* do serviço de Dados, P1 sem prioridade, P2 com prioridade.
Tabela 36: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.
Tabela 37: Tempo *Sojourn* do serviço de Dados, P1 sem prioridade, P2 com prioridade.
Tabela 38: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.
Tabela 39: Tempo *Sojourn* do serviço de Dados, P1 sem prioridade, P2 com prioridade.
Tabela 40: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.
Tabela 41: Tempo *Sojourn* do serviço de Dados, P1 sem prioridade, P2 com prioridade.

SUMÁRIO

1. Introdução ….....1
2. Modelos de fontes de tráfego.....3
3. Gerador de tráfego Multimídia ….....17
4. Pilha TCP/IP.....23
5. Modelos de Simulação.....33
6. Conclusões.....64

1. Introdução

Os sistemas emergentes de comunicação de banda larga estão possibilitando cada vez mais a introdução de diversas aplicações multimídia com alto grau de complexidade que exigem uma taxa de dados elevada. Aplicações em tempo real de vídeo, videoconferência e áudio são bons exemplos. Equipamentos como o CRS-3 (*Carrier Routing System*) da Cisco, vem sendo um contribuidor forte nas redes de alta velocidade, proporcionando uma capacidade de 322 Tbps, três vezes maior do que a sua versão anterior, CRS-1, possibilitando suprir tais aplicações em grande escala.

No passado, redes multimídia como IPTV, tiveram limitações pela baixa penetração da banda larga e pelo custo relativamente elevado de instalação de cabeamento capaz de transportar as informações de forma viável na casa do assinantes. Nos anos que se seguiram, no entanto, o serviço de IPTV residencial cresceu a um ritmo acelerado, estima-se disponibilidade de mais de 400 milhões de residencias em 2010 por todo o globo e as pesquisas mostram um aumento na migração de usuários cada vez maior [1].

O número de assinantes globais de IPTV deve crescer de 28 milhões em 2009 para 83 milhões em 2013 [2]. Europa e Ásia são os principais territórios em termos da grande quantidade de assinantes. Em termos de serviços, a Europa e América do Norte geram uma maior parcela de rendimento global, devido à ARPUs (*Average Revenue Per User*) muito baixa na China e Índia (ARPU é o valor médio do rendimento de uma empresa de coleta de cada usuário por mês. Tais como serviços extras, tais como planos de dados, mensagens e conteúdo para *download*). As receitas do mercado global de IPTV estão previstas para crescer de € 12 bilhões em 2009 para € 38 bilhões em 2013 [2].

Muitos fornecedores mundiais de telecomunicações estão explorando o IPTV como uma nova oportunidade de receita de seus mercados existentes e como uma medida defensiva contra a invasão de mais serviços convencionais de TV a cabo. Além disso, há um número crescente de instalações IPTV dentro de escolas, universidades, empresas e instituições locais. No setor industrial, IPTV é uma realidade e empresas de telecomunicações ao redor do mundo continuam a aumentar suas implementações comerciais de serviços de IPTV.

Grandes fabricantes de equipamentos de telecomunicações como a Alcatel e a Siemens já estabeleceram parceria e realizam acordos com grandes operadoras de telecomunicações. Diversas empresas como a Alcatel, AT&T, BT Vision, DT e muitas outras estão trabalhando com a Microsoft e, juntos, estão alimentando grandes implementações como o Mediaroom, serviço de IPTV [3].

A Siemens comprou uma pequena companhia chamada Myrio em 2006 que fornece serviços de IPTV. A empresa conta com mais de 75 operadoras regionais nos EUA, bem como algumas operadoras no exterior, incluindo como clientes a Royal KPN holandesa Telecom, a Belgacom na Europa e Datanetwork Advanced Communications, na Tailândia, [4]. A Ericsson foi uma das ultimas grandes empresas a entrar nessa batalha suprindo com produtos de rede e softwares IPTV.

Juntamente com o IPTV surgem novas tecnologias e serviços que permitem que as empresas de telecomunicações implementem serviços avançados como a alta qualidade de canais *multicast* IPTV, HDTV baseada em IP, e *Whole Home Media Networking* (WHMN). Outra vantagem da evolução dos serviços de IPTV inclui a personalização e acesso imediato a uma vasta variedade de conteúdos em demanda digital.

Em [5] um modelo para tráfego multimídia é apresentado, neste caso vídeo *stream* utilizam MPEG-4. No qual o tráfego é dissociado gerado por uma aplicação multimídia em um número de componentes básicos e resulta um modelo de tráfego para cada componente. Os componentes são integrados utilizando um modelo de fonte, no qual um *Graphical User Interface* (GUI) é utilizado para facilitar o processo de modelagem do trafego.

Nenhum modelo de tráfego comumente utilizado é capaz de capturar o comportamento fractal de um tráfego Ethernet [6]. Com o proposito de modelar a natureza fractal do tráfego Ethernet, os autores [6] desenvolveram métodos para aumentar a performance da sua autossimilaridade utilizando processos estocásticos Brownian.

Em[7], os autores modelam tráfegos MPEG-4 em nível de *Group of Pictures* (GoP) utilizando modelos autorregressivos. A característica de um tráfego de vídeo *streaming* também depende da compressão adotada. O padrão MPEG4 e H.263 [8] são candidatos potenciais de *codec* para prover multimídia sobre redes sem fio e com fio. Possíveis [9] funções de distribuição para serviços de videoconferência utilizando compressão MPEG-4 mostram que a compressão do tipo Pearson V possui uma precisão mais apurada.

Uma analise estatística de amostra empírica de vídeo VBR é obtido aplicando compressões entre os quadros de vídeo de um filme [10]. Modelos de *wavelet* para trafego de vídeo VBR proporcionam tráfegos simultâneos de vídeo de longo e curto alcance, com a capacidade de reduzir a dependência temporal de forma significativa [11].

Simulações de priorização de filas de pacotes baseado na garantia de Qualidade de Serviço (QoS) em portas de entrada de *switch* de alto desempenho possibilita assegurar QoS de tráfegos multimídia [12].

Vale apana ressaltar que neste trabalho veremos modelos analíticos, simulações e emulações de tráfego multimídia *stream* e elástico, seguindo diferentes funções de distribuições.

1.1 Simulação x Emulação

Os estudos de simulação auxiliam os projetistas na concepção de algorítmos de controle de tráfego a Qualidade de Serviço e também aperfeiçoam a relação custo/desempenho da rede [13]. No desenvolvimento de um programa de simulação para rede de comunicações, há uma necessidade de modelar as demandas dos usuários aos recursos de rede, caracterizar os recursos exigidos para atender a essas demandas e estimar o desempenho baseado em dados de saída gerados pela simulação [14].

A simulação pode ser definida como: “técnica em que se utiliza de um simulador, considerando-se o simulador como um objeto ou representação parcial ou total de uma tarefa a ser replicada” [15]. Essa definição traz dois importantes aspectos necessários à simulação: o primeiro diz respeito ao tratamento baseado em tarefas, no qual se enfatiza o que deve e como deve ser feito para que se atinja o objetivo proposto, enquanto o segundo é a relação com o simulador propriamente dito, que são as formas de interação com o simulador.

Na área de computação, um emulador é um software que reproduz as funções de um determinado ambiente, a fim de permitir a execução de outros softwares sobre ele. Pode ser pela transcrição de instruções de um processador alvo para o processador no qual ele está rodando, ou pela interpretação de chamadas para simular o comportamento de um hardware específico. O emulador também é responsável pela simulação dos circuitos integrados ou chips do sistema de hardware em um software. Basicamente, um emulador expõe as funções de um sistema para reproduzir seu comportamento, permitindo que um software criado para uma plataforma funcione em outra.

Os ambientes de redes computacionais geralmente são formados por camadas. A quantidade e o papel de cada camada pode variar de acordo com o ambiente. Alguns ambientes são puramente físicos como os terminais de *mainframe* e, portanto, na sua emulação cabe apenas o tratamento dos dados enviados do terminal ou para ele, e reproduzir a interação com o usuário. Outros ambientes podem não possuir um *firmware* (algumas vezes chamado de bios), sendo que os programas que serão executados conhecem todo o hardware e sua emulação seria basicamente a interpretação das chamadas ao

hardware para reproduzir seu comportamento. Alguns ambientes possuem *firmware* mas não possuem sistema operacional. Nesse casos é necessário emular também o *firmware* ou obter um com o fabricante.

Devemos distinguir o conceito de emulação do conceito de simulação, tendo em conta que o segundo não corresponde à codificação precisa do comportamento de um sistema, mas sim à dedução de modelos abstratos de funcionamento. A diferença entre os dois conceitos situa-se entre a produção de analogia e a produção de síntese [16].

2. Modelagem de fontes de tráfego

A tecnologia de transmissão em redes multimídia tem proporcionado uma integração de vídeo, voz e dados. Os serviços podem ser divididos em dois tipos distintos: *stream*, como áudio e vídeo e, os que não possuem necessidade de serem em tempo real, denominados de elásticos, que são os serviços de dados.

Discutiremos duas categorias de aplicações de acordo com as restrições temporais. No contexto do protocolo IP a aplicação em tempo real frequentemente são associadas com o protocolo de transporte UDP, e aplicativos que não requerem tempo real associadas com o protocolo TCP. Em aplicações em tempo real, a sequência dos pacotes são definidos pela fonte. Os pacotes chegam ao seus destinos respeitando a sequencia adotada pela fonte do transmissor. Caso isso não ocorra cabe ao receptor ordenar novamente os pacotes. Algumas variações nos valores de *delay* e *jiiter* são permitidas sem que haja grandes perdas e se os limites são respeitados.

Em aplicações que não são em tempo real, as fontes não possuem uma taxa de pacotes definida. As atividades de transferência de dados toleram variação de *delay* e *jitter* com perdas. Com isso não temos uma qualidade de serviço em termos de comparação a aplicações em tempo real (*delay*, *jitter*). É necessário suprir uma taxa de bits nominal. Esse tipo de aplicação baseando-se em TCP gera tráfego "Elástico" (tráfego elástico e *stream* é definido no final do capitulo 2.1) com as seguintes características:

- A taxa da fonte de tráfego varia de um pacote a outro pacote de acordo com a janela de congestionamento do TCP.
- A taxa de bits da fonte de tráfego muda quando a fonte detecta a perda de um pacote e faz a sua retransmissão.
- A taxa de bits da fonte de tráfego depende do tempo da ida e da volta RTT. RTT é o tempo necessário para o pacote chegar no seu destino mais o tempo necessário para o ACK retornar para a fonte.

A maior diferença entre estas duas categorias é o comportamento do aplicativo sobre as condições da rede. A taxa de bits dos aplicativos UDP são independentes da rede e determinados pela fonte, enquanto que a taxa de bits dos aplicativos TCP variam de uma rede para outra. Isso ocorre porque normalmente o protocolo UDP utiliza valores de *codecs* fixos, exigindo que haja uma taxa de banda disponível no minimo igual ou superior do *codec* que o serviço utiliza. Caso essa condição não seja satisfeita ocorrerá degradação da qualidade do serviço, mal funcionamento, ou dependendo da situação não poderá haver o serviço.

No Capitulo 3 veremos com mais detalhes os protocolos de transporte TCP/IP e UDP.

2.1. Modelagem de componentes de tráfego multimídia

Aplicação multimídia consiste em uma integração de dois ou mais tipos de mídia. Um ambiente simples como uma projeção de uma aula pode conter aplicações de áudio, vídeo e dados, sendo que cada aplicação utiliza um tipo de tratamento de tráfego. Mesmo se duas aplicações tem os mesmos tipos de mídia, a quantidade de dados gerados e as estatísticas do processo de geração de dados podem ser diferentes. Por consequência, em lugar de deduzir um modelo de tráfego para cada aplicação, é mais significativo identificar padrões de tráfego comuns a um grande número de aplicações multimídia [5].

O primeiro passo na modelagem de tráfego multimídia são os algoritmos necessários para gerar padrões de tráfego de cada componente, tais como Bloco, Transação, e *Streaming*. Os algoritmos apenas fornecem os procedimentos para a geração de tráfego. Para gerar um padrão de tráfego especifico de uma aplicação multimídia, os parâmetros do modelo devem ser especificados.

Tráfego de Bloco

Para definir uma fonte de tráfego de blocos precisamos especificar a função de distribuição e os parâmetros correspondentes para cada uma das seguintes variáveis:

- Número de Bloco por sessão.
- Tamanho do Bloco.
- Tempo de chegada dos Blocos.

O intervalo possibilita o uso de funções de distribuição continuas mostradas na Tab. 1. Afim de especificar o número de blocos por sessão, uma variação discreta é necessária. Isso pode ser feito usando a distribuição de Poisson ou geométrica.

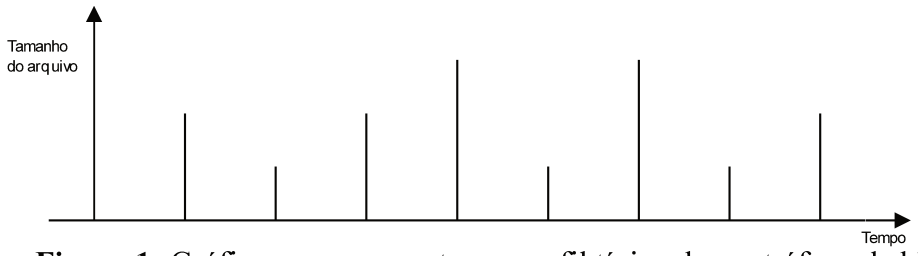


Figura 1: Gráfico que representa um perfil típico de um tráfego de blocos.

Tráfego de Transação

O tráfego de transição consiste em alterações “ON” e “OFF”. Durante o período “ON” pacotes são gerados continuamente. As seguintes variáveis são necessárias para definir uma sessão:

- Duração da Sessão.
- Duração ON.
- Duração OFF.
- Tamanho do pacote.
- Intervalo de chegada dos pacotes durante o período ON.

Para cada variável, uma função de distribuição pode ser usada e os parâmetros correspondentes para as distribuições devem ser especificadas.

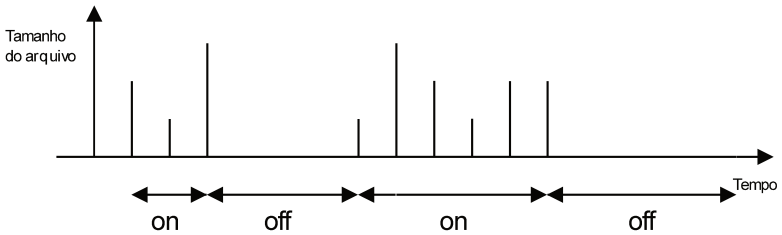


Figura 2: Gráfico que representa um perfil típico de um tráfego de transação.

Tráfego *Stream*

A fonte do tráfego *stream* emite unidades de dados em períodos sem silencio significativo, uma representação esquemática pode ser visto na Fig 3. Além disso, é provável que se mantenha rajadas em diferentes escalas de tempo (devido os chamados efeito de auto-similaridade) [6]. Modelos de auto-regressivos [7] produzem tráfego com uma exponencial deteriorando a função de autocorrelação. A característica de um tráfego de vídeo *stream* também depende da compressão adotada. O padrão MPEG-4 e H.263 [8] são candidatos potenciais de *codec* para prover multimídia em redes *wireless*, por exemplo.

- Uma fonte de taxa constante de bits, especificando a taxa de bits.
- Uma fonte de taxa de bits variável com o modelo auto-regressivo [17].
- Uma fonte de taxa de bits variável utilizando o modelo F-ARIMA [10].
- Uma fonte de taxa de bits variável com o modelo Wavelet [11].

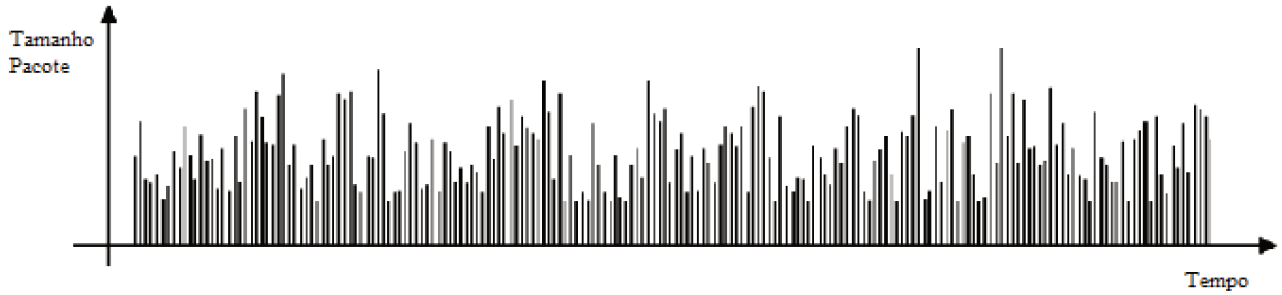


Figura 3: Gráfico que representa um perfil típico de um tráfego *stream*.

Identificação da Categoria de um tráfego Multimídia

Para modelar uma aplicação multimídia, num emulador ou simulador utilizando um ou mais dos componentes básicos descritos acima, primeiramente deve-se fazer a sua identificação. Aplicações multimídia normalmente geram traços de tráfego, onde os componentes podem ser identificados a partir de suas fontes. Mesmo se dois componentes são originários da mesma fonte, os pacotes geralmente podem ser identificados por marcações adequadas. No *header* do TCP/IP, essas marcações são identificadas no campo *Traffic Class* ou *ToS* do pacote enviado ou recebido. Sempre que chega ou se transmite um novo pacote é especificado nesse campo qual o tipo de serviço utilizado. No Ipv4 esse campo é chamado de *ToS* que é composto de 8 bits e no Ipv6 de *Traffic Class* que também é composto de 8 bits. Na Tab. 1 iremos destacar os principais tipos de aplicações e suas respectivos modelos.

Serviços	Bloco	Transação	Streaming
Orientação de Localização	x	x	
Pesquisa de Vôo		x	
Empresas – Negócios			
Entrada de pedido		x	
Mensagem	x	x	
Palestras em sala	x	x	x
Mensagens de imagem	x		
Mensagem multimídia	x	x	x
Vídeo Mensagem	x	x	x
PDA Virtual	x	x	
Entretenimento móvel			
Filmes pequenos ou curtos			x
Musicas			x
Áudio Streaming			x
Jogos moveis		x	x
Telefone com vídeo		x	x
Chat móvel		x	
Comercio móvel	x	x	x
Maquinas de venda automática		x	
Controle remoto		x	
Medição remota		x	

Tabela 1: Principais serviços multimídia e suas respectivas categorias de tráfego.

Abordagem do Tráfego em Fluxo

Na abordagem proposta neste trabalho de validação do modelo de tráfego há uma diferença. Os tipos de tráfego normalmente são descritos como bloco, transação, e *stream*. E utilizam o conceito de pacotes. Por outro lado estamos utilizando o conceito de fluxo. O fluxo é basicamente definido para o caso tráfego elástico, em que considera-se que a sessão TCP é encerrada após um certo tempo sem a ocorrência de pacotes para a origem ou destino. No tráfego *stream* há um fluxo natural que permanece ativo enquanto durar a aplicação.

Para efeito da aplicação do conceito de fluxo os tipos o tráfego são divididos em *stream* e elástico. Nossa definição de *stream*/elástico é em relação ao QoS. Basicamente o *stream* tolera perda, mas não tolera atraso, enquanto o elástico tolera atraso mas não tolera perda.

O *stream* é modelado como trafego de transação para aplicações interativas que requer serviços em tempo real. Seu perfil é um fluxo de dados com altas taxas de utilização quando ativo, e com alta prioridade no canal pois não tolera atrasos. Tráfegos com esse perfil usarão o conceito de *stream*, por exemplo vídeo e voz. O protocolo que melhor se adéqua a esse perfil é o UDP, tendo como principais características um sistema de transmissão simples comparado ao TCP, sem sinalização de tráfego para fornecer confiabilidade, ordem de chegada dos pacotes, ou perca de pacotes, dessa forma garantindo atrasos reduzidos.

O elástico por outro lado possui baixa prioridade no canal. É modelado como fontes ON-OFF em relação ao volume de dados a serem transmitidos, e utilizado para aplicações não interativas. A característica de elasticidade de tráfego é devida basicamente ao comportamento dinâmico de protocolos do tipo TCP, que garante a integridade da informação realizando retransmissão caso algum pacote não seja entregue.

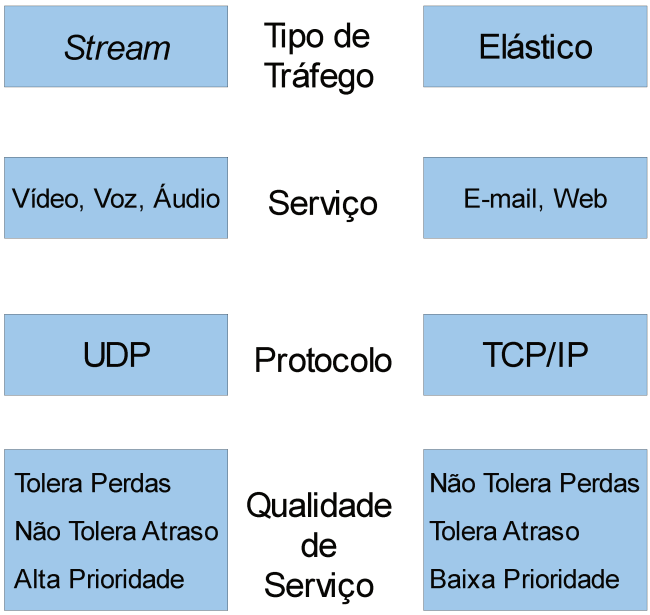


Figura 4: Desenho esquemático comparando características de tráfego elástico e tráfego *stream*.

2.2 Funções de Distribuição

As distribuições de probabilidade mais utilizadas, e mesmo em desempenho de tráfego e problemas de confiabilidade, são as distribuições exponencial e normal. O mesmo procedimento pode ser seguido para vários outros tipos de distribuição de probabilidade, contínuas ou discretas.

O comportamento de cada serviço pode diferir de uma rede para outra apresentado diferentes distribuições. Abaixo são descritas algumas das distribuições. Será feita uma breve descrição das distribuições e situações nas quais são aconselháveis sua utilização.

Distribuição Exponencial

A distribuição exponencial é utilizada quando se deseja analisar o espaço ou intervalo de acontecimento de um evento, utilizada para eventos contínuos. A probabilidade para um certo tempo e espaço entre eventos sucessivos, ocorrendo em um processo de Poisson. É comumente utilizado para intervalos entre chegadas. A função densidade de probabilidade (PDF) da distribuição exponencial é mostrada abaixo:

$$f(x;\lambda)=\lambda e^{-\lambda x}, \qquad (1)$$

onde $x \geq 0$ e $\lambda > 0$ são os parâmetros da distribuição.

Distribuição de Weibull

A distribuição de Weibull, nomeada pelo seu criador Waloddi Weibull, é uma distribuição de probabilidade contínua . A distribuição de Weibull utiliza-se a para modelar o tempo entre falhas de equipamentos. Em redes, um exemplo é a sua utilização no modelamento de transações. Pode representar o período que se situa um parâmetro (*on* ou *off*). A função densidade de probabilidade de Weibull de variável aleatória x é expressa por:

$$f(x;\lambda,k)=\frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}, \qquad (2)$$

onde $x \geq 0$ e $k > 0$, e k é o parâmetro de tamanho, e $\lambda > 0$ é o parâmetro de escala da distribuição. Sua função de distribuição cumulativa é uma extensão da função exponencial . A distribuição Weibull é relacionada a uma série de outras distribuições de probabilidade, em particular, pode descrever uma interpolação entre a distribuição exponencial ($k = 1$) e a distribuição Rayleigh

($k = 2$).

Distribuição de Pareto

A distribuição de Pareto foi introduzida primeiramente pelo economista italiano Vilfredo Pareto em 1897. Desde então muitos pesquisadores a utilizaram em diversas áreas. É uma distribuição de probabilidade que modela fatos dos campos social, científica, industrial e muitos outros tipos de fenômenos observáveis. Fora do campo da economia, é às vezes referida como a distribuição de Bradford. A função densidade de probabilidade da distribuição Pareto é:

$$f(x,\alpha)=\alpha\frac{x_m^{\alpha}}{x^{(\alpha+1)}}, \qquad (3)$$

onde $x > x_m > 0$ e x_m é o valor minimo do x. A família de distribuições de Pareto são parametrizados por, x_m e α, onde α é o índice de Pareto.

Dentre as variáveis que assumem comportamento de distribuição Pareto, e que estão ligadas à análise de tráfego em redes podem ser citados:

- Número de conexões em uma rajada de uma seção FTP.
- Tamanho das rajadas em bytes.
- Tamanho do arquivos.
- Tempo de CPU consumido por processos.
- Período ocioso (enquanto não há tráfego de dados).
- Tamanho de uma conexão WWW.

Distribuição Lognormal

Uma variável pode ser modelada como lognormal se puder ser considerada como o produto multiplicativo de muitos pequenos fatores independentes. Exemplos típicos é o retorno a longo prazo de um investimento em uma ação, pode-se considerar como o produto dos retornos diários. Esta distribuição é também muito utilizada para estimativa de tempo de reparo de um processo, ou seja, o tempo entre ocorrer uma falha e o tempo para que ela seja corrigida. A função densidade de probabilidade da distribuição Lognormal é dado por

$$f(x;\mu,\sigma)=\frac{1}{x\sigma\sqrt{2\pi}}\,\mathrm{e}^{\frac{-\ln(x-\mu)^2}{2\sigma^2}}, \qquad (4)$$

onde μ e σ são a media e o desvio padrão respectivamente e x > 0.

Distribuição de Poisson

A distribuição de Poisson é utilizada para variáveis aleatórias discretas. Ela determina a quantidade de ocorrências em um determinado intervalo de tempo. Os eventos devem ocorrer em um certo intervalo de tempo ou espaço. É utilizada em chamadas telefônicas por unidade de tempo, acidentes por unidade de tempo, chegada de clientes por unidade de tempo e etc. Sua função densidade de probabilidade é dada como:

$$p(x;\lambda,t)=\frac{(\lambda t)^x\,\mathrm{e}^{-\lambda t}}{x!} \quad para\ x=0,1,2,... \qquad (5)$$

onde λ é a taxa média do processo e t é o intervalo de tempo/espaço.

A distribuição possui as seguintes características:

- O tempo de chegada é exponencial, que não possui memória.
- A taxa média do processo, λ, deve permanecer constante durante o período de tempo e espaço considerados.
- Quanto menor o segmento de tempo e espaço, menor a probabilidade de ocorrer mais de um evento naquele segmento. A probabilidade de ocorrência de 2 ou mais eventos tende a zero quando o tamanho do segmento tende a zero.

Distribuição Hiper-exponencial

A distribuição Hiper-exponencial é uma distribuição contínua na qual é dada como:

$$fx(x)=\sum_{n=1}^{n} fy_i(x)\times p_i, \qquad (6)$$

onde y_i é uma distribuição exponencial. É considerada uma distribuição hiper-exponencial desde que seu coeficiente seja maior do que o coeficiente da distribuição exponencial.

Um exemplo no contexto de telefonia onde se um usuário possui um modem e um telefone, a probabilidade p_i do usuário utilizar o telefone ao mesmo tempo que o modem é dado por pela distribuição hiper-exponencial.

2.3 Qualidade de Serviço (QoS)

Na area de redes de comunicações o termo Qualidade de Serviço (QoS) refere-se a mecanismos de reserva de recursos. O QoS é uma ferramenta que possui diferentes prioridades as diferentes aplicações, para garantir um determinado nível de desempenho para um fluxo de dados. Diferentes parâmetros podem ser usados para garantir o QoS, como, taxa de bits, atraso, *jitter*, taxa de erro, e etc. Mecanismos que garantam a qualidade de serviço são importantes se a capacidade da rede é insuficiente, especialmente para aplicações em tempo real, como vídeo *stream*, já que estas aplicações exigem uma taxa de bits fixa e são sensíveis a atrasos. A seguir veremos alguns desses mecanismos.

Vazão

Em termos práticos as aplicações geram vazões que devem ser atendidas pela rede. Dessa forma o canal deve ter uma largura de banda mínima para o funcionamento adequado para cada tipo de aplicações. A tabela a seguir ilustra a vazão tipica de algumas aplicações.

Aplicações	Vazão Tipica
Voz	10 Kbps a 100 Kbps
Aplicações Web	10 Kbps a 500 Kbps
Video Streaming	100 Kbps a 1 Mbps
Video MPEG	1 Mbps a 10 Mbps

Tabela 2: Vazão típica das aplicações.

Latência

A latência da rede pode ser definida como o somatório dos atrasos impostos pela rede e equipamentos utilizados no sistema de comunicação. Como o atraso de propagação, fila, velocidade de transmissão e processamento nos equipamentos.

Devido à latência, o tempo que leva para um pacote chegar ao seu destino pode aumentar. Isso porque existe congestionamentos durante o percurso de entrega do pacote. Como por exemplo longas filas, ou rotas com um maior número de nós.

Delay

O delay ocorre quando os pacotes de serviços, como de voz, lavam mais tempo do que o esperado para chegar ao seu destino. Isso causa degradação na qualidade da chamada. Entretanto se for tratado adequadamente os seus efeitos são minimizados, como a utilização de *buffers*.

Jitter

O *jitter* pode ser entendido como a variação no tempo e na sequencia de entrega das informações, ocorrendo uma distorção no processamento da informação na recepção. Para evita-lo deve-se utilizar mecanismos específicos de compensação e controle que dependem da aplicação em questão. Genericamente, uma das soluções mais comuns para o problema consiste na utilização de *buffers*.

As aplicações não interativas não são muito afetadas pelo jitter, porém os usuários de aplicativos que envolvem mídia interativas (áudio e vídeo) como Voz Sobre IP (VoIP), e videoconferência podem ser muito afetados pelo jitter. Isso porque as aplicações interativas utilizam *codecs*, que, para um funcionamento correto, para garantir uma taxa de fluxo de dados constante. Os pacotes que carregam mídias interativas são gerados em suas origens a taxas constantes mas, ao atravessarem a rede, sofrem retardos diferentes e chegam ao destino não sincronizados.

Em principio, o problema dos pacotes fora de ordem poderiam ser resolvidos com o auxilio de um protocolo de transporte como o TCP, que verifica a sequencia das mensagens e faz as devidas correções. Entretanto, na pratica tem-se que a grande maioria das aplicações multimídia optam por utilizar o UDP ao invés do TCP pela maior simplicidade e menos *overhead*. Nestes casos, o problema de sequencia deve ser resolvido por protocolos de mais alto nivel normalmente incorporados a aplicação como, por exemplo, o RTP (*Real Time Transfer Protocol*).

A definição ideal do tamanho de um *buffer* para prevenir o *jitter* é muito importante para a qualidade da transmissão de mídias interativo. Se o *buffer* é pequeno demais pode não conseguir abrigar uma longa rajada de pacotes devido a retenções na rede ou em um servidor de mídia, ocasionando perda por transbordo. Também por ser de tamanho insuficiente, o *buffer* não será capaz de armazenar um número de pacotes necessário para anular o efeito do *jitter* caso isto aconteça, o *buffer* será esvaziado e a reprodução do vídeo ou áudio sofrerá paralisações dando origem aos atrasos. Se o *buffer* é grande demais acentua-se o problema de interrupção entre sessões interativas, ou seja, interrupções entre as falas dos locutores ou entre as respostas dos participantes de uma videoconferência.

O protocolo RTP [RFC 1889] leva em conta o princípio básico de que um *buffer* dinâmico aumenta seu tamanho quando “enxerga” um aumento no *jitter* da rede. Nele, o campo *inter-arrival jitter* prevê uma medida de curto prazo para o congestionamento da rede. A finalidade dessa medida é indicar um congestionamento na rede antes que ocorra uma perda de pacote. O protocolo RTP utiliza diversos algoritmos para estimar uma possível mudança no tamanho do *buffer,* através da verificação de todos os campos *inter-arrival jitter* de um determinado receptor ao longo do tempo ou de múltiplos receptores de uma mesma rede.

2.4 Técnicas alternativas de QoS

IntServ e RSVP

A técnica IntServ está atualmente sendo definida pelo IETF e corresponde a um conjunto de recomendações (RFC 2211, 2212, 2215) visando a implantação de uma Infraestrutura robusta para a internet que possa suportar o transporte de áudio, vídeo e dados em tempo real [25].

Na arquitetura IntServ, dois aspectos operacionais são considerados para que a aplicação possa realizar a reserva dos recursos que serão utilizados antes de iniciar o envio dos pacotes pela rede. Eles são:

- Como as aplicações solicitam sua necessidade de QoS à rede.
- E como os dispositivos intermediários, como roteadores, devem proceder para garantir o QoS solicitado.

A solicitação é realizada através do protocolo de sinalização RSVP. Nele a aplicação informa quais recursos serão necessários. Os dispositivos intermediários da rede serão os responsáveis para garantir a reserva solicitada. Uma vez aceita a reserva, os fluxos de dados correspondentes a aplicação são identificados e roteados.

DiffServ

O DiffServ usa mecanismos de marcação de pacotes, verificação local da classe dos pacotes, e gerenciamento de recursos para suportar diferentes níveis de serviços em uma rede IP. Seguindo [25], sua terminologia é a seguinte:

- *Per-hop behavior* (PHB): Todos os pacotes devem sofrer a mesma analise durante um salto, enfileiramento e descarte, caso os recursos não satisfaçam o serviço.
- *Differentiated services code point* (DSCP): É um valor no cabeçalho IP que indica que o pacote em questão sofrerá a analise PHB.

Os DSCPs no cabeçalho do pacote indicam como os pacotes devem ser tratados nos saltos entre os roteadores intermediários até que o pacote chegue ao seu destino. O DSCP esta presente no cabeçalho do protocolo IPv4 no campo *Type of Service* (ToS) e no protocolo IPv6 no campo *Traffic Class*. Seis bits, dos 8 bits de um byte do ToS, são alocados para o DSCP, e os outros dois bits são alocados para a *Explicit Congestion Notification* (ECN). O comportamento dos saltos do DiffServ podem ser definidos da seguinte forma:

- Melhor esforço.
- Baixo atraso, latência, e *jitter*.

O Diffserv foi desenvolvido pensando em ser mais simples e escalável do que o IntServ/RSVP, porque não requer sinalização ou fluxo na rede. Todas as aplicações podem utiliza-lo sem precisar realizar nenhum tipo de mudança nos roteadores. O meio de transporte utilizado para o DiffeServ pode ser tanto IP quanto ATM, Frame Relay, MPLS, ou uma combinação poque esses exemplos de protocolos de transporte estão em um nível acima da camada de rede. Assim sendo, os roteadores analisam apenas a camada de rede para que possam descobrir qual é o *host* de destino dos pacotes para encaminha-los. Diferentes manipulações de pacotes e mapeamento são possíveis por exemplo, o indicador da classe de serviço pode representar prioridades de congestionamento de fluxo onde pacotes com baixa prioridade são descartados primeiro.

Politica de controle de admissão de chamadas.

De acordo com as necessidades práticas e especificações do 3GPP, as prioridades para tráfego multimídia são descritos na Tab. 5 [23].

Classe	Definição 3GPP	Prioridade	Classe de tráfego
1	Conversação	Alta	Voz
2	*Streaming*	Média	Vídeo
3	Interatividade	Baixa	Internet
4	*Background*	Mais Baixa	E-mail

Tabela 3: Prioridades para tráfego multimídia.

Os tráfegos de voz e vídeo são sensíveis ao atraso e os pacotes de voz não podem ser retransmitidos quando ocorre uma falha. No caso do vídeo, pode-se perder alguns bits menos significativos quando o recursos são insuficientes. Normalmente, os pacotes podem tolerar algum atraso, especialmente os de dados que são do tipo elástico, podendo tolerar mais atrasos do que os do tipo *stream*. No caso de *stream*, se o sistema estiver sobrecarregado é recomendado o bloqueio do pacote ao invés de colocá-lo em um *buffer*, pois a sessão pode ser refeita novamente após ser bloqueada.

Como indicado na Tab. 5, as classes dos tráfegos descrevem diferentes tipos de comportamento, com isso, teremos diferentes QoS que significam prioridades diferentes. Por exemplo, a probabilidade de bloqueio é um critério importante para QoS para voz, mas não chega a ser tao importante como para dados, pois ele pode suportar atrasos. As classes com prioridades mais altas têm privilégio sobre as classes com prioridades mais baixas no processo de CAC. A QoS das classes de prioridades mais altas devem ser processadas antes que as outras, certos privilégios devem ser reservados, como a retirada de uma parte dos recursos destinados às classes mais baixas quando os recursos do sistema estiver inadequado. Suprindo a taxa de tráfego necessária para seu funcionamento correto. Assim, a QoS da classe de tráfego poderá ser garantida.

Basicamente, a politica do CAC é baseada na estimativa da capacidade do recursos reservados e não reservados. A política de reserva é mostrada na Fig. 9. A capacidade total é considerada como sendo igual a 1, então de acordo com requisitos de QoS e as definição de prioridades, a capacidade máxima pode ser usada por voz, vídeo, internet e de e-mail são: 1, (1–c), (1–b), (1–a), respectivamente, onde a, b, c são dinamicamente ajustados com base na demanda do sistema e na medição da carga de tráfego, e $a \geq b \geq c$. Portanto, os limites da capacidade não reservada para voz, vídeo, internet e de e-mail são de 0, c%, b%, e a%, respectivamente. Quando chega uma chamada de voz, a capacidade da banda não reservada é estimada seja superior ao limiar ou não. Se assim for, a chamada é admitida, caso contrário, pode privar uma parte do recurso alocado para as classes de menor prioridade de tráfego, como vídeo, internet ou de e-mail, que tenham sido admitidos. O processo de privação é o seguinte: os tráfegos de e-mail são postos de lado, e colocados em um *buffer*. Então, faz-se a estiva da capacidade disponível novamente. Se for mais do que o limite, o pedido de voz poderia ser admitido, caso contrário, o tráfego de internet que está no canal é bloqueado, e da capacidade disponível faz-se a estimativa novamente. Se a capacidade disponível ainda é inferior ao limiar, a qualidade de transmissão do tráfego de vídeo será degradada para liberar uma parte dos recursos alocados. O pedido de voz será admitido se o recurso residual é adequado, caso contrário, será rejeitado.

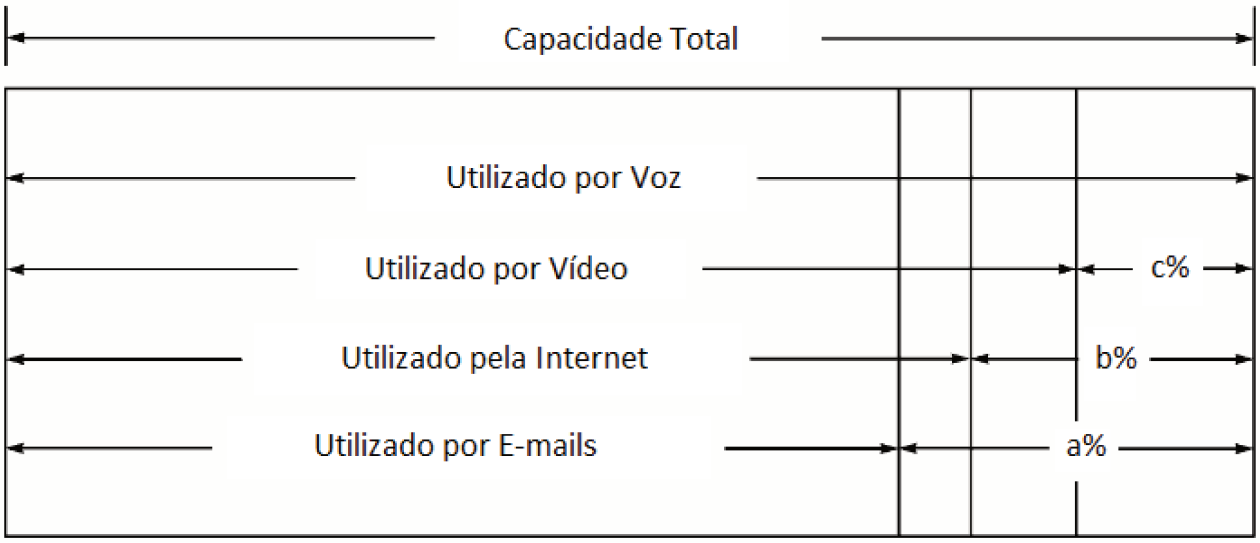


Figura 5: Reservas dos serviços.

2.5 Modelo Analítico

Baseado na referência [28], o dimensionamento do tráfego é realizado por dois modelos independentes, um para o tráfego elástico e um outro para o *stream*. Assim a capacidade total do canal é dividida em dois, uma parte é dedicada ao tráfego elástico e a outra ao tráfego *stream*. Por exemplo, C = C*s* + C*e* = 4 Mbit/s, onde C é a capacidade total do canal, C*s* é a porção *stream*, e C*e* a porção elástica. O modelo anaítico aqui apresentado é utilizado nos modelos de simulações 2 e 3.

2.5.1 Dimensionamento do Tráfego Elástico

Como o serviço de Dados é o único tráfego elástico presente no nosso modelo, assumimos que o valor da banda C*e* igual a 128 kbit/s é suficiente. Em um caso ideal, o canal que transmite o protocolo TCP, tráfego elástico, pode ser representado como um sistema Processor Sharing (PS) no qual todos os fluxos ativos compartilham os recursos disponíveis igualmente [29]. Além disso, o canal pode ser modelado como uma fila M/G/R PS, onde o processo de chegada é Markoviano (M) ou exponencial e a distribuição de duração do serviço é general (G). No nosso caso utilizamos a distribuição exponencial com R=2, que equivale a uma banda C*e* = 64x2 = 128 kbit/s (assumindo a menor quantidade de banda disponível é 64 kbit/s). Para estimar o tempo de espera para o tratamento deste tráfego, é utilizado a distribuição de espera Erlang C, dada por

$$E_2(R,\rho) = \frac{\frac{\rho^R}{R!} \times \frac{R}{R-\rho}}{\sum_{i=0}^{R-1} \frac{\rho^i}{i!} + \frac{\rho^R}{R!} \times \frac{R}{R-\rho}}, \qquad (7)$$

onde ρ denomina o tráfego gerado pelo serviço de Dados, no qual é igual a 10x0.035 = 0.35 Erl (Tab. 7) que fornece uma probabilidade esperada de E_2 (2, 0.35) = 0.05213.

Para o modelo M/G/R PS, o tempo médio para transferir um arquivo de tamanho *x* é dado por

$$E[T(x)] = \frac{x}{h_e}\left(1 + \frac{E_2(R,\rho)}{R - \rho}\right) = 0.17\text{s}, \quad (8)$$

onde x = 84 kbytes (Tab. 7) e supondo que a taxa de transferência de uma conexão não é limitada, a taxa de pico é a largura de banda do canal, h_e = 4 Mbit/s porque a exigência é de que 100 kbytes seja transferidos em 15 segundos, o que implica em 84kbytes seja transferidos em 12,6 segundos, portanto o tempo de permanência é muito baixo comparado com o desempenho. Como visto, os requerimentos para um QoS foi satisfeito com Ce = 128 kbit/s.

2.5.2 Dimensionamento do Tráfego *Stream*

A capacidade total do canal é de 4 Mbit/s, sendo que 128 kbit/s foram alocados para tráfegos elásticos, então, haverá uma capacidade disponível de 3872 kbit/s para o tráfego *stream*. Para cada solicitação de serviço i, a constante da taxa de bit c_i é reservada, e após sua requisição, ela é liberada imediatamente. A solução geral na forma de produto pode ser escrito como

$$p(n_1, n_2, ..., n_N) = \prod_{i=1}^{N} \frac{\rho_i^{n_i}}{n_i!}\frac{1}{G}, \quad (9)$$

onde G é a constante de normalização. A probabilidade de bloqueio do serviço i para suportar o tráfego ρ_i é denotado por b_i, e pode ser calculado pelo método aproximado de Labourdette [30] dado por

$$b_i \approx \frac{1 - \alpha^{c_i}}{1 - \frac{C_s}{M}} E\left(\frac{M}{d}, \frac{C_s}{d}\right), \quad (10)$$

onde Cs é o canal reservado para o tráfego *stream* e

$$M = \sum_{i=1}^{m} c_i \rho_i, \quad (11)$$

onde M é o tráfego total oferecido, m é o número de tipos de elementos de rede, p_i e c_i são, respectivamente, a intensidade do serviço i e banda efetiva. Todos os serviços *stream* tem uma largura de banda efetiva de 32 kbit/s, que é o valor do *codec* utilizado

$$\sum_{i=1}^{m} \frac{c_i \rho_i}{C_s} \alpha^{C_i} - 1 = 0, \quad (12)$$

onde α: parâmetro que satisfaz a equação,

$$d = \frac{\log \frac{C_S}{M}}{\log \alpha}, \quad (13)$$

onde d: largura de banda do sistema equivalente,

$$E_1\left(\frac{C_S}{d}, \frac{M}{d}\right) = \frac{\left(C_S/d\right)^{\left(M/d\right)}}{\sum_{i=0}^{\left(M/d\right)} \frac{\left(C_S/d\right)^i}{i!}}, \quad (14)$$

denota a formula de bloqueio de Erlang B para um número fracionário de troncos (servidores), exigindo um procedimento numérica específico para a sua resolução.

Sendo *Cs* = 3872 kbit/s, *m*=5, com valores de ρ_i da Tab. 7, obtemos α= 1.00000710266, *M*= 3084800 bit/s, *d*= 32000 bits (são todos iguais os *codec* dos serviços) obtemos

$$E\left(\frac{C_s}{d}, \frac{M}{d}\right) = 2.10^{-3}. \qquad (15)$$

Portanto, os valores de bloqueio $b_1 \approx 2\text{x}10^{-3}$ (todos os serviços *stream*), satisfazendo os requisitos de QoS estipulados.

3. Gerador de Tráfego Multimídia

Um gerador de tráfego multimídia é um gerador de tráfego associado a serviços multimídia, como áudio, vídeo, voz, e dados. Para se gerar um padrão de tráfego, MPEG-4 por exemplo, de modo semelhante aos vistos em um cenário real, utilizam-se funções de distribuições que melhor representam o seu comportamento. Dessa forma, para que um gerador de tráfego seja o mais fiel possível a tráfegos reais, é necessário a utilização de diferentes tipos de distribuições para cada tipo de serviço.

O gerador multimídia originalmente foi feito utilizando-se um *Berkeley sockets* (mais conhecido como BSD socket API), e foi desenvolvido por Agugliari e Oliveira [20]. O *socket* foi utilizado, em princípio, para facilitar a implementação, pois em um primeiro momento foi importante a verificação dos perfis de tráfego. A incorporação da pilha TCP/IP ao gerador é essencial para se ter controle dos cabeçalhos do pacote IP que são responsáveis por determinar o tipo de serviço. No Capitulo 4 discutiremos a pilha TCP/IP que substituirá o *socket*.

O gerador de tráfego multimídia é composto por dois subsistemas: o gerador, que gera os pacotes de acordo com o perfil desejado para cada serviço; e o receptor, que está presente em outro elemento da rede e recebe os pacotes e armazena as informações sobre o tráfego. Para as simulações, foi utilizada inicialmente apenas a distribuição exponencial, tanto para intervalo de chegada quanto para tamanho/duração dos serviços, por ser a mais utilizada quando se deseja analisar o espaço ou intervalo de acontecimento de um evento.

O gerador de tráfego multimídia consiste dos seguintes principais módulos:

Seletor de Serviço: Contém as informações de cada serviço. Armazena os dados responsáveis pela criação de um novo serviço em um tempo determinado, além garanti o início do funcionamento da máquina de estado. Cada serviço tem um comportamento pré-definido e um momento certo para que seja iniciado. Isso é verificado através de uma distribuição de probabilidade que represente o comportamento da rede. No entanto, para que um serviço seja iniciado, deverá antes ser verificado o controle de admissão de cada serviço, que é requisito da rede simulada, ou seja, o número máximo de serviços simultâneos suportados pela rede.

Perfilador de Tráfego: Contém as informações sobre o tipo de tráfego que vai ser gerado para cada serviço. Esse módulo por sua vez irá influenciar na geração dos pacotes, indicando a periodicidade e o tamanho com que eles devem ser criados. Tendo identificado as características de um novo serviço, é necessário, através do perfil de tráfego, identificar a duração do serviço (no caso de voz, áudio e vídeo), ou o tamanho dos pacotes (no caso de dados), cujos parâmetros variam a cada novo serviço. Eles também devem seguir uma distribuição de probabilidade.

Gerador de Pacote: Gera os pacotes para cada serviço. Os pacotes deverão ser identificados no Gerador e no Receptor, pois contem as informações de cada serviço. Como os dados gerados são aleatórios, e a fim de se obter dados para montagem da estatística final dos pacotes, ao invés de enviar pacotes sem significado, são enviados no conteúdo os dados relacionados a seguir:

- Tipo do serviço (voz, vídeo, áudio ou dados).
- Identificação do serviço (para numeração dos serviços de acordo com cada tipo).
- Número do pacote (para contagem final dos pacotes de cada serviço e verificação de perdas).
- Tempo de geração do serviço (momento em que o serviço foi gerado).
- Tempo de duração total do Serviço Voz, Vídeo e Áudio ou tamanho total do serviço para Dados.
- Tempo de geração do pacote e tempo do pacote entrar na fila.
- Tempo que leva para o *Socket* enviar o pacote.
- Bits de preenchimento (para completar o tamanho do pacote, se necessário).

Após cada pacote ser gerado contendo as informações listadas anteriormente, ele é enviado para a rede via *socket*. Ele foi utilizado, em princípio, para facilitar a implementação, pois neste primeiro momento é importante verificar os perfis de tráfego. Para trabalhos futuros, a substituição do *socket* pela pilha TCP/IP é essencial, proporcionando controle do cabeçalho do protocolo IP, como os seguintes *headers: Type of Service, Tcflow,* e *Traffic Class*, que serão utilizados para separar os pacotes de acordo com o serviço.

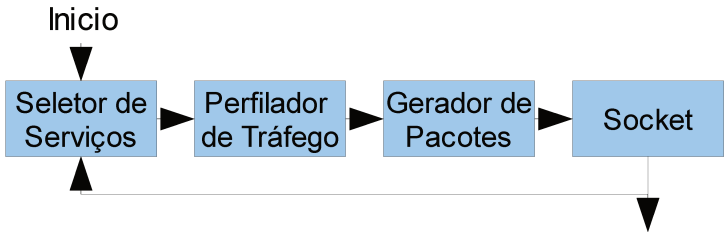


Figura 6: Representação esquemática da Máquina de Estado do Gerador.

Para a execução em paralelo da geração dos quatro serviços e do tratamento e envio dos pacotes foi utilizado o conceito de Thread. Ela está presente nos seguintes pontos:

- Ciclo principal Serviço Tráfego Pacote.
- Tratamento de geração individual de pacotes, após ser definido o perfil do tráfego.

Para que sua implementação fosse comum a vários sistemas operacionais foram utilizadas a linguagem C++ e o padrão POSIX Threads, através da biblioteca pthread.h. O padrão define uma API para criar e manipular Threads em sistemas baseados em UNIX, como GNU/Linux, Mac OS X e SOLARIS. Sua utilização também é possível no sistema operacional Windows, através da API pthread-w32 que traduz as instruções de Thread para que possam ser interpretadas nas plataformas 32-bits do Windows.

O modelo arquitetural de Threads segue o padrão conhecido em Java, que pode ser através da criação de uma nova classe Thread ou através da implementação de Runnable. Este conceito foi retirado de [21] que contém uma implementação padrão de classes para POSIX Threads.

3.1 Descrição dos Protocolos de Transporte

Protocolo TCP (*Transmission Control Protocol*)

O protocolo TCP, formalmente definido na RFC 793, tem como principal característica oferecer um fluxo confiável de dados fim a fim. Uma rede pode não ser confiável devido a vários fatores como topologia, largura de banda, retardo, tamanhos de pacote entre outros, que são completamente diferentes umas das outras. Ele foi projetado para se adaptar a diferentes aspectos de rede e ser robusto diante dos fatores de falha. O cabeçalho TCP contém 20 bytes que são detalhados a seguir:

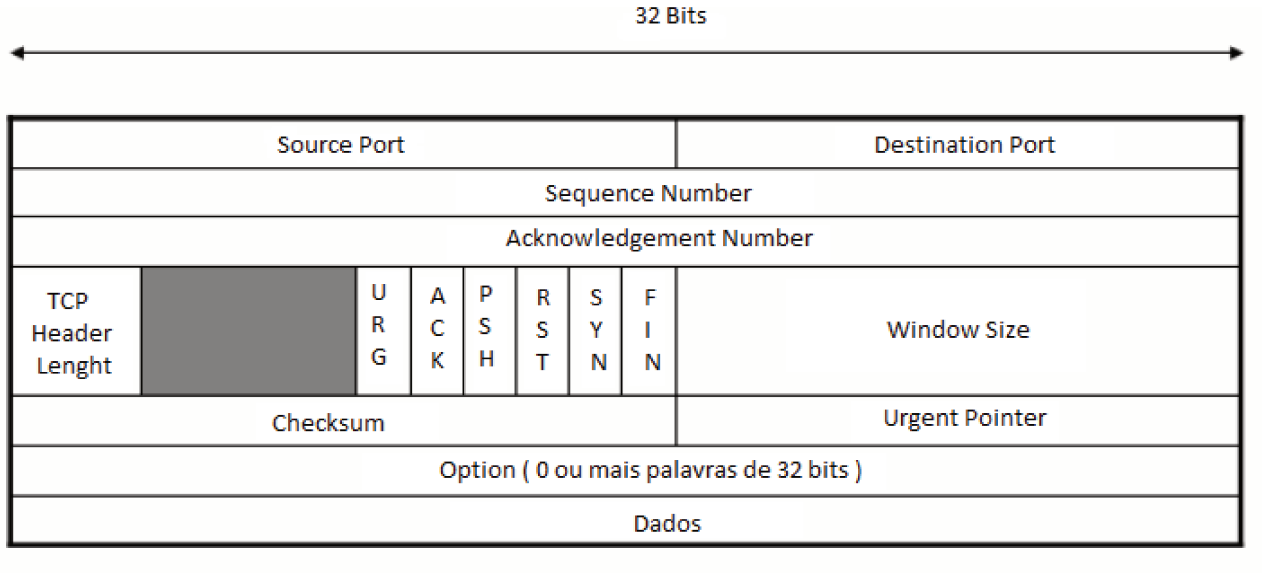


Figura 7: Segmento TCP [24].

- Source / Destination port: Número IP e porta TCP endereçam um destino único ou um destino para a entrega de um segmento. O IP determina o *host* e a porta determina a aplicação que tem aberto a porta. Um *socket* deve ser utilizado para múltiplas e simultâneas conexões.

- Sequence/ Acknowledgement number: Enumera cada byte para fragmentação e reconstrução.

- TCP header length: Número com palavra de 32 bits do cabeçalho TCP

- Flags: URG – Sinaliza dado urgente do *Urgent Pointer*
ACK – Sinaliza que número de *Akcnowledgements* é válido
PSH – Esse tipo de dado deve ser enviado imediatamente sem *buffer* adicional por razões de eficiência
RST – Limpa a conexão
SYN – Usado no estabelecimento de conexão
FIN – Libera canal de comunicação

- *Window size*: Para controle de fluxo ponto a ponto. Contém o número de octetos que o receptor está pronto para aceitar. Uma janela TCP é a quantidade de dados não confirmados pelo receptor que um remetente pode enviar através de uma conexão antes de receber uma confirmação de recebimento do destinatário.
- *Checksum*: Informação usada para checar o cabeçalho, dados, e pseudo cabeçalho para confiabilidade. O receptor adiciona todos os 16 bits recebidos incluindo o *checksum*, e o resultado deve ser 0 se os campos do *checksum* estiverem de acordo.

- *Options*: Uma opção é o máximo tamanho do *payload* do TCP que o *host* irá aceitar. O padrão é 556 bytes. É um fator de aumento de escala da janela de recepção para redes de alta velocidade. Ele é utilizado na negociação do MSS (*Maximum Segment Size*).

Protocolo UDP (*User Datagram Protocol*)

O UDP, ao contrário do TCP, oferece um meio para transmissão sem que haja a necessidade de conexão. Ele é descrito na RFC 768 e não realiza : Controle de Fluxo, Controle de Erros, Retransmissão após recepção de um segmento incorreto. O segmento UDP consiste de um pequeno cabeçalho de 8 bytes, como pode ser visto na figura a seguir.

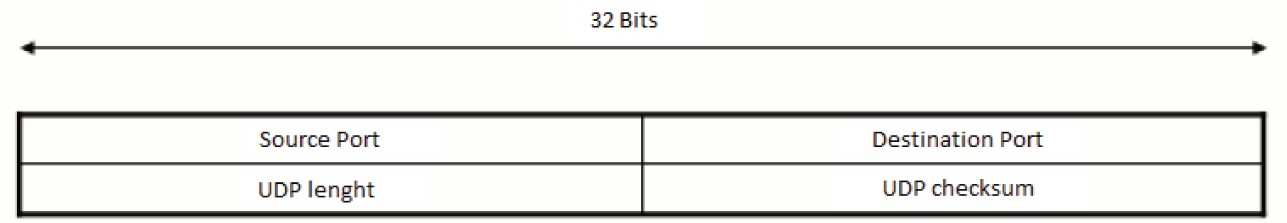


Figura 8: Datagrama UDP [24].

- *Source Port*: Usada quando uma resposta deve ser devolvida a origem

- *Destination Port*: Indica qual a porta de comunicação do destino

- *UDP Length*: Inclui o cabeçalho de 8 bytes e os dados, indicando o tamanho do segmento

- *UDP Checskum*: Campo opcional para controle de erro.

Ele apenas fornece uma interface para o protocolo IP com o recurso adicional de demultiplexação de vários processos que utilizam as portas. É útil para processos cliente-servidor, onde o cliente envia uma pequena solicitação ao servidor e espera uma resposta. Caso ocorra um *timeout,* o cliente tenta novamente porém sem existir inúmeras trocas de mensagem como no TCP. O UDP é uma escolha adequada para:

- Fluxos de dados em tempo real que suporta perda limitada de pacotes, como vídeos ou voz.
- Aplicações sensíveis a atrasos na rede, mas pouco sensíveis a perdas.

O protocolo TCP envolve retransmissões e espera de dados, trazendo como consequência uma alta latência.

3.2 Resultados Numéricos

As simulações foram inicialmente feitas utilizando-se o gerador de tráfego e o software de simulação de eventos discretos de propósito geral ARENA [18, 23]. O gerador de tráfego permite um nível maior de detalhes de transmissão e mais próximo da realidade (como taxa de um *codec*, tamanho do pacote, transmissão pelo canal e problemas que podem ocorrer durante uma transmissão como queda de conexão), e o software ARENA permite uma simulação com abstração dos pacotes, abstraindo detalhes de transmissão e preocupando-se mais com as características de geração e disponibilidade de cada serviço.

O gerador de tráfego permite conviver com um ambiente mais próximo do real. No entanto, pelo fato dele realizar uma emulação em tempo real, se sua duração for de 10 horas, ele terá que ser executado durante 10 horas.

Já o simulador por eventos discretos, pelo fato de trabalhar com as características mais importantes dos eventos que ocorrem na rede, pode fazer a mesma rodada de 10 horas de simulação em apenas alguns minutos. Na verdade a simulação sera a mais simples possível, de modo que consiga refletir os principais problemas de contenção das filas que certamente se formarão na rede devido a perdas/atrasos.

Os resultados das simulações do ARENA e do Gerador de tráfego foram bem próximos. Em relação à rejeição de serviços *stream* devido ao controle de admissão, também foram obtidos dados coerentes se compararmos com as simulações realizadas. Apenas em um serviço de vídeo ocorreu falha nas simulações do gerador de tráfego, contra zero do simulador ARENA. Essa perda provavelmente ocorreu devido à geração aleatória, que iniciou um serviço antes dos outros serem finalizados. As outras perdas também ficaram em zero, indicando que o sistema está bem dimensionado para atender ao tráfego feito na simulação.

Inicialmente as simulações do ARENA e a emulação do Gerador de Tráfego Multimídia foram realizadas utilizando uma distribuição exponencial com os mesmos parâmetros. O fato de se utilizar apenas as distribuições exponencial inicialmente foi para facilitar a sua calibração. Na referência [37] também há detalhes sobre a emulação, simulação e resultados obtidos.

- Largura de banda total disponível: 4 Mbps

- Tempo total: 10 horas

- VoIP
  - Ligações com duração média de 180 s (3 min)
  - Tempo médio entre requisições de serviço de 36 s (100 chamadas por hora em média)
  - Máximo de 70 chamadas simultâneas (controle de admissão)
  - CODEC de 16 kbps

- Áudio
  - Sessões com duração média de 500 s (~8 min)
  - Tempo médio entre requisições de serviço de 360 s (10 chamadas por hora em média)
  - Máximo de 10 sessões simultâneas (controle de admissão)
  - CODEC de 20 kbps

- Vídeoconferência
  - Sessões com duração média de 900 s (15 min)
  - Tempo médio entre requisições de serviço de 3600 s (1 chamada por hora em média)
  - Máximo de 2 sessões simultâneas (controle de admissão)
  - CODEC de 32 kbps

Os resultados obtidos no ARENA e no Gerador de Tráfego Multimídia estão apresentados nas Tab. 4 e 5. Podemos observar que os valores são bem compatíveis. Note que o ARENA simula 10 horas de serviço em bem menos tempo (aproximadamente 10 minutos), enquanto o gerador executa o tempo total (10 horas).

Na emulação foi utilizado dois computadores Apple, um como transmissor e o outro como receptor. O transmissor é um iMac Intel Core 2 Duo 2.2Ghz, e o receptor um Mac mini com um PowerPC G4 de 1 Ghz. Na simulação foi utilizado um notebook HP AMD Turion X2 64 2.0 Ghz. Aparentemente, as diferenças observadas devem-se a fatores aleatórios intrínsecos ao modelo estatístico. Dependem também do intervalo de geração e das características do emulador (o que não ocorre no caso da simulação).

Serviço	Média (s)	Mínimo (s)	Máximo (s)	Quantidade
VoIP	169,35	0,46	1179	1023
Áudio	496,75	13,9	2845	110
Vídeoconferência	1071	51,66	4361	12

Tabela 4: Características dos serviços gerados na simulação do Arena.

Serviço	Mínimo (s)	Máximo (s)	Quantidade
VoIP	18	656	1009
Áudio	93	2736	87
Vídeoconferência	49	3287	19

Tabela 5: Características dos serviços gerados na emulação do Gerador de Tráfego Multimídia.

4. Pilha TCP/IP

Neste capitulo é descrito o comportamento da pilha uIP (TCP/IP) que, a princípio, irá substituir o *socket* do gerador descrito no capitulo 3. O uIP foi desenvolvido inicialmente por Adam Dunkels do grupo *Networked Embedded Systems* no *Swedish Institute of Computer Science* [22]. Além dele, o uIP teve contribuições de diversos desenvolvedores de software de todas as partes do mundo. Nós a escolhemos porque é uma pilha baseada em Unix e de código aberto, possibilitando a sua adaptação de forma a alcançar o nosso proposito, que é a substituição do *socket* pela pilha TCP/IP, afim de termos controle do cabeçalho IP .

A implementação do uIP [20] foi feita para possuir apenas os requisitos mínimos necessários para o funcionamento adequado da pilha TCP/IP, proporcionando uma interface de rede simples que contem os protocolos IP, ICMP,UDP e TCP.

A pilha uIP foi criada para a possibilidade de realizar comunicação utilizando o protocolo TCP/IP em até mesmo microcontroladores de 8-bits. O tamanho do código é de apenas alguns kilobytes e a RAM utilizada pode ser configura da para ser menor que algumas centenas de bytes.

4.1 O uIP

A pilha uIP pode ser executada tanto como um sistema multitarefas, ou como monotarefa. Nos dois casos, o laço principal realiza duas ações repetidamente:

- Checa se algum pacote chegou.
- Checa se o tempo de saída periódico foi excedido.

Quando um pacote chega, a função de entrada *uip_input()* é chamada pelo laço principal. Nela são feitos diversas verificações, como a do *header* para descobrir qual tipo de serviço e canal pertence aquele pacote para então notificar a devia aplicação, verifica se a largura do pacote é adequada ao tamanho do *buffer*, e configura diversas constantes e ponteiros para que a próximo laço os identifique. Além disso é feito a verificação de erros nos pacotes utilizando o *checksum.*

O *timer* periódico de saída é utilizado em mecanismos do TCP, tais como o *delayed acknowledgment*, retransmissão e o tempo de *round-trip* (ida e volta). Quando o laço principal infere que o *timer* periódico deve disparar, deve-se chamar a função de manipulação do *timer* periódico, *uip_periodic()*. Com isso, a pilha TCP/IP pode realizar uma retransmissão do pacote quando ocorre um atraso que excede a temporização de resposta de algum evento realizado.

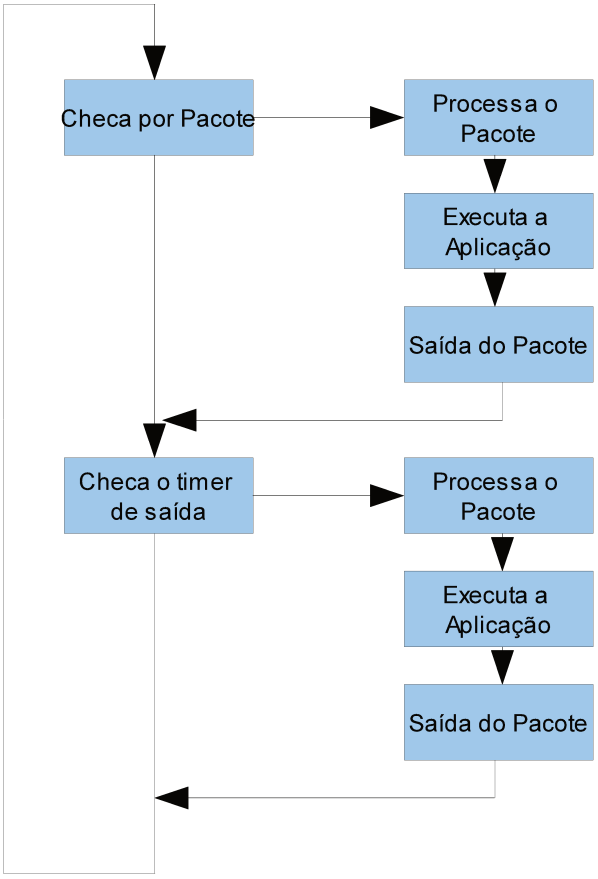


Figura 9: Laço principal de checagem de pacotes.

Controle de Memória

A pilha uIP não usa alocação de memória dinâmica de forma explícita. Em vez disso, usa um *buffer* global para guardar os pacotes, e possui uma tabela fixa para guardar os estados das conexões. Esse *buffer* é grande o suficiente para conter um pacote com o seu tamanho máximo. Quando um pacote chega, o *driver* do dispositivo o guarda no *buffer*. Se o *buffer* contem dados, o TCP/IP irá notificar a aplicação correspondente imediatamente para desocupar o *buffer* o mais rápido possível. Dessa forma, o *buffer* estará sempre livre para os próximos pacotes de dados que chegarão. Com isso, a aplicação terá que processar rapidamente os dados que estão no *buffer* ou colocá-los em um *buffer* secundário para ser processado futuramente. Os pacotes no *buffer* não serão apagados ou sobrepostos pelos novos pacotes recebidos antes que a aplicação os tenha processado. Assim, os pacotes são postos em uma fila pelo dispositivo de rede caso os dados no buffer ainda não tenham sido processados. Se a fila estiver cheia, os pacotes de chegada serão perdidos. Isso porque o uIP utiliza uma janela de entrada muito pequena, significando que haverá apenas um segmento TCP por conexão.

No uIP, o mesmo *buffer* global que é usado para os pacotes recebidos é também usado para os pacotes que serão transmitidos. Se a aplicação transmite dados de tamanhos aleatórios, pode-se utilizar parte do *buffer* global que não esta sendo utilizado como um *buffer* temporário. Para enviar os dados a

pilha, o aplicativo passa a apontar para os dados, bem como para a variável *len* que indica o seu comprimento. O cabeçalho TCP/IP é escrito dentro do *buffer* global. Quando estão prontos para serem enviados o *driver* os envia juntamente com os dados da aplicação para a rede. Os dados não ficam em fila para retransmissão. Em vez disso, a aplicação irá ter que reproduzir o dado se a retransmissão for necessária.

A capacidade máxima de uso de memória depende fortemente das aplicações do dispositivo em que as implementações são executadas. A configuração de memória determina a quantidade de tráfego que o sistema deve ser capaz de lidar e os montantes máximos de conexões simultâneas. Um dispositivo que envia e-mails grandes, enquanto ao mesmo tempo, executa um servidor web com páginas web de tamanhos aleatórios e vários clientes simultâneos, por exemplo, exigirá mais memória RAM e capacidade de processamento do que um simples servidor Telnet.

Application Program Interface (API)

O API define como o programa de interface da aplicação interage com a pilha TCP/IP. O API mais utilizado para pilhas TCP/IP é o *socket* BSD, que é utilizado em sistemas Unix, e tem influenciado muito o Winsock API da Microsoft. Como o *socket* API usa a sintaxe *stop-and-wait*, requer um suporte de multitarefa do sistema operacional. O uIP provê dois APIs para se trabalhar: *protosocket*, um *socket* BSD como o API sem o *overhead* do *multi-threading*, e um API de mais baixo nível que o *protosocket*, baseado em eventos.

O *protosocket* provê uma programação sequencial de interface para facilitar a programação do uIP. Adicionando um pequeno aumento na memória, 1000 bytes no código e 26 bytes extras por conexão TCP e apenas funciona para conexões TCP. Eles utilizam *protothread* para fornecer controle de fluxo sequenciais. Isso faz com que a memória fique mais leve, mas também significa que o *protosocket* herda as limitações funcionais do *protothread*. O *protosocket* possui funções de transmissão sem que haja a necessidade de retransmissões ou *acknowledgment*, bem como as funções de leitura de dados sem ter que lidar com os dados sendo divididos em mais de um segmento TCP.

API Baseado em Eventos

O API baseado em eventos do uIP usa uma interface baseada em eventos, onde a aplicação é chamada em resposta de certos eventos, como o recebimento de um pacote. Uma aplicação acima do uIP é implementada como uma função em C que é chamada pelo uIP em resposta a determinados eventos. O uIP chama a aplicação quando um dado é recebido, quando um dado foi entregue ao outro ponto da conexão com sucesso, quando uma nova conexão foi configurada, ou quando um dado foi retransmitido. A aplicação é também periodicamente consultada para verificar se deseja enviar novos dados. A aplicação prove apenas uma função *callback* e responsável por lidar com os diferentes mapeamentos de serviços da rede para diferentes portas e conexões. A aplicação atua nos dados de chegada e nas requisições de conexões, assim que a pilha TCP/IP receber um pacote.

O uIP é diferente das outras pilhas TCP/IP porque requer o auxílio da aplicação para a realização de retransmissão. Em outras pilhas TCP/IP o dado transmitido permanece na memória até que se saiba que foi transmitido com sucesso. Se o dado precisa ser retransmitido, a própria pilha cuida dessa retransmissão sem precisar notificar a aplicação. Com essa abordagem, é necessário que os dados permaneçam na memória esperando pelo *acknowledgment* mesmo se a aplicação esteja pronta para reconstruir o dado se a retransmissão tiver que ser feita.

No uIP utiliza-se a ideia de que a aplicação sempre está pronta para a recuperação do dado transmitido caso haja uma perda, com isso, é a aplicação a responsável por recriar o pacote para fazer a retransmissão. O uIP não mantém o caminho do conteúdo do pacote depois de ter sido transmitido pelo driver, e requer que a aplicação faça um papel ativo na parte de retransmissão. Quando o uIP decide que um segmento deve ser retransmitido, ele chama a aplicação com um *flag* indicando uma retransmissão. O aplicativo checa o *flag* e produz o mesmo dado que foi transmitido anteriormente. Do ponto de vista da aplicação, a realização de uma retransmissão não difere de como o dado original foi transmitido. Portanto, a aplicação pode ser escrita de tal forma que o mesmo código é usado tanto para o envio de dados como para retransmissão de dados. Além disso, é importante notar que, mesmo que a operação de retransmissão seja realizada envolvendo a aplicação, é responsabilidade da pilha saber quando uma retransmissão deve ser feita. Assim, a complexidade da aplicação não aumenta necessariamente porque realiza uma parte ativa fazendo a retransmissão.

Ponteiro da Conexão

Para uma melhor compreensão de como funciona o uIP, iremos detalhar alguns processos de maior importância. Quando a aplicação é chamada pelo uIP, uma variável global *uip_conn* é configurada para apontar para a estrutura de conexão. Os campos da estrutura *uip_conn* podem ser utilizados para distinguir entre diferentes serviços, ou para checar para qual endereço IP a conexão é requisitada. Um uso típico seria inspecionar o valor *lport* do *uip_conn* (*lport* é o campo que contem a porta local do TCP) para decidir qual serviço a conexão deveria prover. Por exemplo, um aplicativo pode decidir a agir como um servidor HTTP se o valor do *lport* do *uip_conn* é igual a 80 e agir como um servidor Telnet se o valor é 23.

Chegada de Dados

Quando novos dados chegam, eles são armazenados no *buffer*. Dessa forma, os conteúdos antigos serão substituídos e os novos serão atribuídos à aplicação correspondente através do ponteiro *uip_appdata*. O *uip_appdata* é um ponteiro que aponta para os dados da aplicação que está no *buffer*. Ela é chamada quando recebe ou envia dados.

Além disso, um valor diferente de zero será passado à função de teste *uip_newdata,* indicando que novos dados chegaram. O tamanho do dado é obtido através da função *uip_datalen* e mantido na variável *uip_len,* para poder informar a aplicação o seu tamanho.

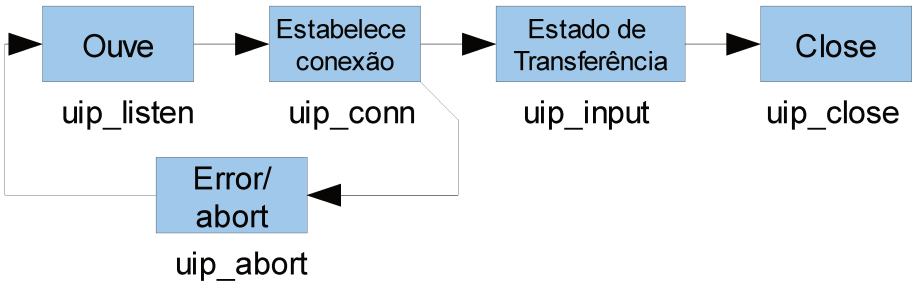


Figura 10: Diagrama de blocos dos processos de transmissão de dados ao receber de um novo pacote.

Basicamente, o processo de transmissão de dados ao receber um novo pacote é dado pelas funções da Fig.10. A função *uip_listen* realiza um loop de checagem constante para verificar se novos dados chegaram. Caso algum dado chegue, a função *uip_conn* estabelecerá a conexão entre os dois pontos. Se ocorrer algum problema na conexão, o *uip_abort* realiza o processo de abortar a conexão.

Para se processar um dado de entrada, a função *uip_input* é chamada quando a camada de enlace (L2) receber novos dados da rede. Os pacotes devem ser mantidos no *uip_buf* e o tamanho do pacote deve ser mantido na variável *uip_len*. Quando a função retorna, pode haver um pacote de saída novo colocado no *buffer uip_buf*. Se assim for, a variável *uip_len* é definida como o comprimento do pacote. Se nenhum pacote deve ser enviado, a variável *uip_len* é definida como zero.

Transmitindo Dados

Quando os dados são transmitidos, o uIP ajusta o tamanho dos dados a serem transmitidos pela aplicação de acordo com o espaço no *buffer* disponível e a janela TCP atual anunciada pelo receptor. A quantidade de espaço no *buffer* é derminada pela configuração da memória. Assim, é possível que todos os dados enviados a partir do pedido não cheguem ao receptor. Para evitar perdas o aplicativo pode usar a função *uip_mss* para verificar a quantidade de dados que realmente deve ser enviado pela pilha, sem que haja esse tipo de perda.

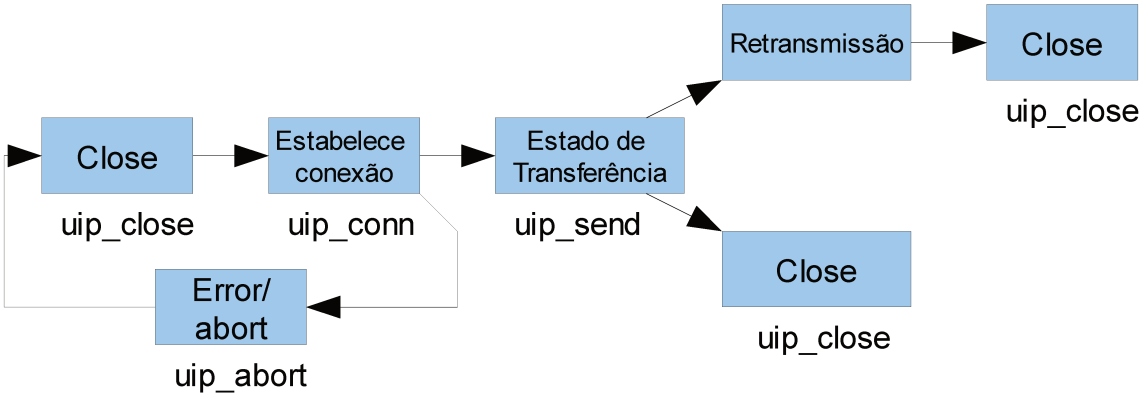


Figura 11: Diagrama de blocos dos processos de envio de um novo pacote.

A aplicação transmite dados usando a função do uIP *uip_send*. Essa função usa dois argumentos: um ponteiro para o dado a ser enviado e o tamanho do dado. Se o aplicativo precisa de espaço na memória para a manipulação do dado atual a ser enviado, o *uip_buf* pode ser usado para este propósito. Para isso, é utilizado o ponteiro *uip_sappdata,* que irá apontar a posição de memória no *uip_buf.* O aplicativo pode enviar apenas uma parte dos dados por vez em uma conexão, e não é possível chamar o *uip_send* mais de uma vez por aplicativo. Portanto, somente os dados da última chamada serão enviados.

Retransmissão

As retransmissões são realizadas pelo *timer* periódico. Todas as vezes que o *timer* periódico é executado, o tempo da retransmissão para cada conexão é decrementado. Se o tempo chegar a zero, a retransmissão será feita. Como o uIP não mantém o conteúdo do pacote depois destes terem sido enviados pelo *driver*, o uIP requer que a aplicação faça um papel ativo na realização da retransmissão,

recriando os mesmos dados que foram enviados anteriormente para a retransmissão.

Quando o uIP decide que o segmento deverá ser retransmitido, a função *uip_rexmit* é configurada com um *flag*, indicando que uma retransmissão é necessária. A aplicação verificará o *flag* da função *uip_rexmit* que indica qual pacote deve ser recriado e em seguida reproduzirá o mesmo dado que fora mandado anteriormente.

Fechando a Conexão

A aplicação fecha uma conexão chamando a função *uip_close* durante uma sessão. Isso fará com que a conexão seja fechada corretamente comunicando ao *host* remoto. Caso ocorra um erro fatal, a aplicação poderá abortar a conexão, e o faz chamando a função *uip_abort*. Se a conexão for encerrada pelo *host* remoto, o teste de função *uip_closed* é verdadeiro. A aplicação pode então encerrar a conexão e liberar os recursos para novas conexões.

Reportando Erros

Há dois erros fatais que podem ocorrer com uma conexão: ou a conexão foi abortada pelo *host* remoto, ou os últimos dados da conexão foram retransmitidos muitas vezes e com isso foi abortada. O uIP reportará chamando a função da aplicação. Dessa forma a aplicação poderá testar qual foi a condição de erro utilizando as funções *uip_aborted* e *uip_timedout* para reportar o erro.

Polling

Quando a conexão está ociosa, o uIP chama a aplicação todas as vezes que o *timer* periódico disparar. A aplicação usa a função de teste *uip_poll* para checar se esta sendo 'ouvido' pelo uIP. O evento tem duas finalidades. A primeira é para deixar a aplicação sabendo periodicamente que uma conexão esta ociosa, que permite a aplicação fechar conexões que ficaram ociosas por muito tempo. O outro propósito é o de deixar a aplicação enviar novos dados que forem processados. A aplicação pode apenas mandar dados quando chamado pelo uIP e, portanto, esse evento é o único procedimento de enviar dados em uma conexão ociosa.

Checagem de Portas

Para se detectar requisições de conexões em determinadas portas o uIP utiliza a função *uip_listen.* Essa função possui uma lista de portas referindo-se a cada aplicação podendo rejeitar uma conexão se a porta é desconhecida. A aplicação checa o campo *lport* da estrutura *uip_conn* para descobrir a qual porta essa nova conexão está associada. Quando um pedido de conexão verifica uma dessas portas, o uIP cria uma nova conexão e chama sua respectiva aplicação. Dessa forma o estado do ponteiro de teste *uip_connected* é verdadeiro pois uma nova conexão foi criada.

Estabelecendo a Conexão

Uma nova conexão pode ser feita a partir da função *uip_connect*. Essa função aloca uma nova conexão e define um *flag* de estado, no qual se abrirá uma conexão TCP para o endereço IP e porta especificados, na próxima vez que a conexão for consultada pela uIP. A função *uip_connect* retorna um ponteiro para a estrutura *uip_conn* para a nova conexão. Se não houver espaço livres, a função retorna

NULL. A função *uip_ipaddr* pode ser usada para encapsular um endereço IP, para um vetor de dois elementos de 16 bits usado pela uIP, para representar os endereços IP.

Controle de Fluxo

A função *uip_process* é uma função robusta, e complexa, responsável pela verificação e orientação da pilha. Nela se processa informações como: chegada e envio de dados, comunicação com o aplicativo, checksuns, estabelecimento de conexão, *timer*, etc. O propósito da função não ter sido dividida em diversas outras menores é que o tamanho do código aumentaria, implicando em sobrecarregar as passagens de parâmetros e o fato de que a otimização não seria tão eficiente.

A pilha TCP/IP analisa o cabeçalho do pacote, e chama a sua respectiva aplicação. Esse pacote é armazenado no *uip_buf* até que a aplicação o leia. Caso seja necessário armazenar esses bytes, é a aplicação que se encabe desse processo, pois uma vez lido, será apagado do *uip_buf* para novos dados que estão chegando ou que estão esperando para ser transmitidos. Caso os dados estejam fora da sequência correta, a pilha não encaminhará os dados a aplicação.

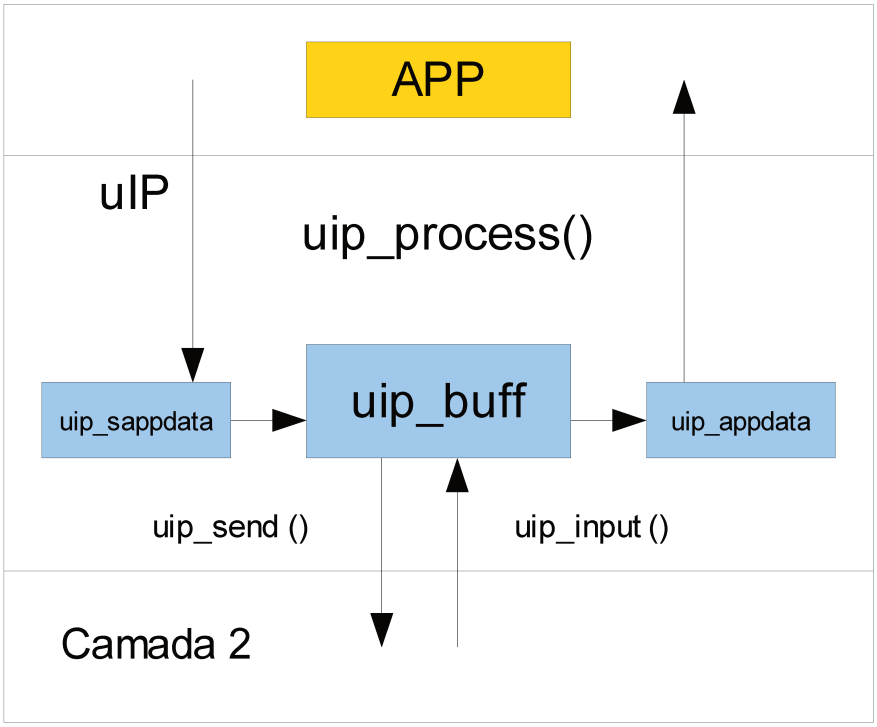


Figura 12: Diagrama de fluxo com as principais características do uIP.

Se a aplicação deseja enviar um dado, ela deve colocar os dados dentro do *uip_buf*. Para isso, o ponteiro *uip_sappdata* deverá apontar para o primeiro byte que esteja disponível no *buffer* para o seu armazenamento. A pilha TCP/IP calcula o seu respectivo *checksun*, completa os campos do *header* necessário e finalmente envia o pacote.

4.2 Interface TUN/TAP

A interface TUN/TAP foi criada por Maxim Krasnyansky e Maksim Yevmenkin [36], com o proposito de realizar uma interface de rede virtual no *kernel* que recebe e transmite pacotes. O TUN/TAP realiza, basicamente, as mesmas funções da camada LLC. Podendo ser um dispositivo simples *Point-to-Point* ou um dispositivo *Ethernet*. Do ponto de vista do *kernel*, a interface TUN/TAP funciona basicamente igual uma interface de rede normal, mas ao invés de receber e enviar pacotes de um dispositivo físico ele se comunica com um dispositivo logico.

Redes virtuais podem ser usadas de diversas maneiras tornando-as muito flexíveis. Geralmente, o TUN/TAP é utilizado em dois cenários. O primeiro é o VPN. Nesse cenário, o *kernel* envia os pacotes para o dispositivo TUN ou TAP. Em seguida o software VPN irá encriptar e encaminhar os pacotes para o outro lado do tunelamento VPN, onde serão decriptados e entregues ao seu destino. O segundo cenário é o caso mais tradicional, no qual funcionam como virtualizador/emulador de pacotes. Neste caso, o sistema operacional virtualizado se comunica com um dispositivo de rede virtual (normalmente um adaptador *Ethernet* virtual). O software de virtualização então cria um dispositivo TAP e os interconecta, criando uma ponte de comunicação.

Basicamente o TUN é um dispositivo de rede virtual *Point-to-Point* desenvolvido para suportar tunelamento utilizando quadros IP. O TAP é um dispositivo de rede virtual *Ethernet,* desenvolvido para suportar tunelamento *Ethernet* utilizando quadros *Ethernet*.

O TUN/TAP é composto por extensões do *kernel*, um provendo interface TUN e a outra TAP. Eles criam um conjunto de dispositivos /dev/tunX e /dev/tapX, respectivamente, onde o X é o identificador da interfaces virtuais. Quando a primeira interface é criada, ela é denominada de /dev/tap0 ou /dev/tun0 dependendo do dispositivo e, pode-se atribuir um endereço a ela da mesma forma que é feito em um interface de rede normal. Após a configuração da interface, os pacotes que o *kernel* enviou através da interface (determinados pela tabela de roteamento) podem ser verificados por sua vez no dispositivo de rede (utilizando um *tcpdump* por exemplo).

Essa forma de interface virtual vem se popularizando cada vez mais. Diversas plataformas, como FreeBSD, Linux, Max OS X, Solaris, Windows a adaptaram. E diversos redes privadas virtuais o utilizam, como OpenVPN, OpenSSH, Hamachi e NeoRouter.

A pilha TCP/IP uIP não possui um *driver* que se comunica com a camada física, pois a ideia do desenvolvedor, Adam Dunkels [22], foi a de desenvolver uma pilha TCP/IP que poderia ser usada em diversos tipos de microcontroladores por exigir pouco recurso de processamento e memória para seu funcionamento. Logo, o TUN/TAP foi usado para esse proposito, afim de substituir virtualmente o *driver.*

Para a utilização da interface TUN/TAP do Unix primeiramente é preciso configura-lo para que possa identificar o uIP e criar o tunelamento entre o TUN/TAP e o próprio uIP. Uma das configurações que são feitas no uIP é o endereço IP do destino que irá estabelecer comunicação. Nessa função de configuração, é preciso identificar o endereço IP da interface TUN/TAP, para que o uIP possa estabelecer uma conexão com o dispositivo virtual. Isso é feito configurando o respectivo endereço da interface TUN/TAP na função *uip_ipaddr* do uIP.

As próximas etapas de configuração das interfaces são realizadas no próprio *shell* do Unix. A criação da interface TUN/TAP e a configuração do seu identificador e do endereço IP. O endereço como já citamos deve ser o mesmo do uIP.

Para que o tráfego da interface TUN/TAP possa chegar a interface de rede eth0, o Unix exige que uma terceira interface de controle seja criada para realizar o papel de uma ponte entre o TUN/TAP e a interface de rede eth0.

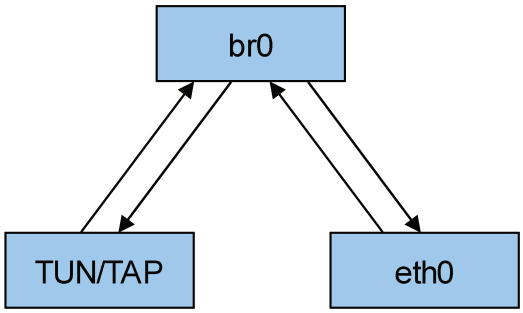


Figura 13: Tráfego entre as interfaces. TUN/TAP e eth0 através da ponte br0.

Consequentemente a interface br0 também terá um identificador e um endereço IP, para que a interface TUN/TAP e eth0 possam transmitir os pacotes de dados. A interface eth0 está vinculado com o adaptador de rede do computador, com isso, se incumbirá de transmitir os dados para o meio físico.

Mudanças feitas no código

Foram feitas mudanças no código com o intuito de identificação dos serviços multimídia utilizando campos do *header* IPv4 e IPv6. O protocolo IPv4 e IPv6 possuem por padrão campos de identificação. Esses campos identificam qual o tipo de segmento que esta dentro do pacote IP. Dispositivos intermediários e *hosts* podem verificar esses campos para fazerem uma rotina de prioridade de nível mais baixo, na camada de rede, aumentando o desempenho entregando o pacote com uma menor latência.

Para a identificação dos serviços utilizamos os campos *Type of Service* do IPv4 e *Traffic Class* e *Tcflow* do IPv6. Inicialmente esses campos já existiam no uIP, mas eram preenchidos com zero. Bits de redundância eram inseridos para completar o tamanho total do *header* a ser transmitido. Então retiramos os bits de redundância e inserimos identificadores para serviços de voz, vídeo, áudio e dados.

Além da identificação implementamos uma função para gerar a estatística dos serviços. Assim conseguimos saber a quantidade que foi gerada de cada serviços e quantos pacotes associados a cada serviço foram transmitidos. E através da comparação com a estatística registrada no receptor conseguirmos calcular a quantidade de serviços perdidos, serviços completados, e retransmissões de cada serviço.

Problemas encontrados

Após a configuração dos endereços IP das interfaces (TUN/TAP, br0, eth0), portas, e a criação da ponte, foi possível realizar testes para verificar sua confiabilidade. O teste mais comum para este tipo de situação é “pingar” as interfaces. O *ping* basicamente é uma mensagem enviada de um *host* para outro *host*, afim de identificar se ele recebe e se transmite de volta um reconhecimento (*ack)*.

Para a realização deste teste utilizamos uma ferramenta (*tcpdump*) que possibilitou checarmos todos os bits que chegam e saem das interfaces. Dessa forma conseguimos identificar com precisão se o quadro que saiu de uma das interfaces é o mesmo que chegou na outra.

O resultado do teste constatou que a ponte estava se comunicando corretamente. Todas as interfaces se comunicaram sem perda de pacotes. E todo o tráfego que chegava em uma delas, por exemplo na eth0, era automaticamente transmitido para a ponte, que por sua vez a enviava para o TUN/TAP.

O próximo passo seria enviarmos com sucesso algum quadro da pilha uIP para o TUN/TAP. O uIP, juntamente com a pilha TCP/IP, possui exemplos de aplicação para que possamos testar a pilha e, também, para podermos aprender o seu funcionamento. Essas aplicações de exemplo vão de um simples Hello-World a um WebClient. Por fim, tentamos enviar para a interface um quadro com todas as aplicações e nenhuma delas teve sucesso. Em principio, nenhuma delas compilavam, em diversos caso haviam linhas de código pela metade, ou funções faltando parâmetros. Em cada aplicação testada foi preciso desenvolver novamente a aplicação para consertar os erros que haviam nos exemplos.

No total foram quatro aplicações testadas, Hello-World, SMTP, WebClient, e WebServer. Em todas elas somente após muita correção e implementação a pilha compilou. Mas ao executa-las nenhum bit chegava ao TUN/TAP. Mais uma vez após varias verificações constatamos problemas na função que transmite os dados, a função *uip_send()*.

A depuração mostrou que o uIP possui a opção de transmitir os quadros para a interface TUN/TAP, mas ela não foi desenvolvida. Dessa forma, a pilha exige o desenvolvimento de um *driver* para o seu funcionamento. Um *driver* que utilize a interface TUN/TAP ou que se comunique diretamente com o meio físico. Para ambos os casos, o mais seguro é desenvolver um *driver* que se comunique diretamente com o meio físico para evitar futuros problemas com a interface TUN/TAP.

Outro ponto é o fato do uIP ter sido desenvolvido para ser implementado em microcontroladores de baixa capacidade, onde sua arquitetura foi projetada para atender a um fluxo de dados mais restrito, tendo em vista a utilização de apenas um *buffer* global de armazenamento de pacotes para todas as funções que a camada de rede deve suportar. Isso pode acabar tornando-se, para a nossa abordagem, um problema de velocidade de processamento no envio e recepção dos pacotes, já que o volume de tráfego que utilizamos é elevado.

Com o surgimento dessas inconveniências houve um aumento na quantidade de atividades a serem feitas. Assim um estudo mais amplo com as simulações do ARENA foi proposto.

5. Modelos de Simulação

Simulações, como a maioria dos métodos de análise, envolvem sistemas e modelos que os representam [19]. Neste capítulo daremos alguns exemplos dos tipos de modelos de simulação e sua descrição, para que, em seguida, analisemos os modelos que propomos.

A função de um modelo é descrever o funcionamento da realidade através de um pequeno número de variáveis que permita a sua representação. Existem diferentes tipos de modelos de simulação. Os modelos comumente utilizados são os modelos físicos, lógicos ou matemáticos. A seguir veremos alguns exemplos [38]:

- **Modelo Físico**: Neste tipo de modelo é criada uma maior afinidade com a realidade, visto que a simulação é experimentada como real. Como por exemplo, simulações de salas de controle com o proposito de treinar trabalhadores em situações de planejamento/acidente nuclear. Simulação de vôo real, para treinar pilotos em casos de panes, como queda de energia, problemas com o trem de pouso, e etc.
- **Modelo Lógico**: Modelos lógicos são aproximações e suposições, tanto estrutural quando quantitativo, em como o sistema trabalha ou como deve trabalhar. O modelo lógico é normalmente representado em um programa de computador, podendo ser utilizado para diversos fins, como estudar o comportamento de um determinado cenário.
- **Modelo Matemático**: São utilizados para estudar situações extremas, dificilmente observadas na realidade.

Uma segunda forma de classificação de módulos é a relação entre os elementos envolvidos, podendo ser classificada das seguintes formas:

- **Estáticos ou Dinâmicos**: denominam-se como modelos estáticos os que visam representar o estado de um sistema em um instante ou que em suas formulações não se leva em conta a variável tempo, enquanto os modelos dinâmicos são formulados para representarem as alterações de estado do sistema ao longo da contagem do tempo de simulação.
- **Determinístico ou Estocástico**: são modelos determinísticos os que em suas formulações não fazem uso de variáveis aleatórias, enquanto os estocásticos podem empregar uma ou mais variáveis aleatórias.
- **Discretos ou Contínuos**: são modelos discretos aqueles em que o avanço da contagem de tempo na simulação se dá na forma de incrementos cujos valores podem ser definidos em função da ocorrência dos eventos ou pela determinação de um valor fixo, nesses casos só sendo possível determinar os valores das variações de estado do sistema nos instantes de atualização da contagem de tempo; enquanto para os modelos contínuos o avanço da contagem de tempo na simulação dá-se de forma contínua, o que possibilita determinar os valores das variáveis de estado a qualquer instante.

Propomos três diferentes modelos de simulação para ambientes multimídia, onde serviços *stream* e elástico são requisitados ao mesmo tempo pelos usuários. No modelo 1 foram realizados simulações propondo um estudo de demanda de requisições dos usuários, onde três cenários foram estudados. O primeiro é quando ocorre um aumento repentino de usuários, com isso diminuindo o intervalo de requisições dos serviços. O segundo cenário, quando a duração dos serviços aumenta, ocorrendo bloqueios de novos serviços por falta de recursos, por exemplo, troncos em um central

telefônica. E o terceiro cenário é com relação ao Controle de Admissão de Chamada (CAC) do sistema. Esses tipos de situações são amplamente estudadas em casos de mudanças do comportamento de tarifação das empresas de telefonias móvel e fixa, onde a cobrança não é mais feita por minutos utilizados, mas sim por chamadas feitas, aumentando significativamente a duração das chamadas. Esse tipo de estudo é crucial por parte das empresas para não acarretar em uma baixa Qualidade de Serviço e bloqueios de chamadas elevados infringindo as regras da ANATEL.

O modelo 2 e 3 são ambientes VPN, onde é estudado o *Sojourn*, tempo que o pacote leva para chegar ao *host* de destino, de um serviço elástico. Veremos cada um com mais detalhes neste capitulo. A principal diferença entre os dois modelos é que o modelo 2 não apresenta a implementação de atributos como *jitter* e latência, enquanto que o modelo 3 os possui.

Para realizar as simulações foi utilizado a versão de "estudante" do simulador ARENA 12.0. A principal diferença entre da versão de "estudante" e a completa é que ela possui um limite de entidades simultâneas. Por causa disso algumas simulações que gostaríamos de ter executado não foi possível, porque ultrapassavam as 150 entidades que a versão de "estudante" permite. Para as simulações foi utilizado o hardware:

- Notebook Acer Aspire 5530G
- AMD Turion X2 Ultra 64 bits, 2.2Ghz
- Memória Ram de 2 Gb
- Windows 7

O tempo de simulação varia de 1 hora a 10 horas para cada simulação, dependendo dos parâmetros utilizados. No total foram executados 190 simulações para os três modelos propostos.

Em cada modelo utilizamos diferentes valores de replicações. Quanto maior o número de replicações mais preciso o resultados são. Por outro lado o seu aumento resulta em tempos maiores para o termino da simulação, inviabilizando simular um número grande de diferentes cenários e parâmetros. Em cada modelo utilizamos valores diferentes de replicações. Seus valores são citados na descrição de cada modelo.

O campo *warm-up* define o tempo necessário para alcançar condições de estado estacionário. Nos modelos de simulação 1, 2 e 3 utilizamos valores igual a 1000 segundos.

5.1 Modelo de Simulação 1

O primeiro modelo de simulação é o estudo de um ambiente *stream*, onde temos três serviços : Videoconferência, Vídeo-Clip, e Vídeo *On Demand*. Cada serviço possui padrões diferentes de comportamento, como duração do serviço, intervalo entre as requisições dos serviços, CAC, e funções de distribuições. Como já discutido anteriormente, com esse estudo analisaremos a quantidade de serviços completados, rejeitados e a quantidade total gerada dependendo da demanda atribuída ao sistema. É importante citarmos a importância de um estudo de QoS para que haja uma margem de tolerância caso ocorra um pico acima do normal e também para que não ocorra nenhum erro ao se projetar uma rede. Para isso, utilizaremos três cenários diferentes. Na referência [37] encontram-se as simulações que veremos neste modelo. Eles são:

- Cenário A: Variação do intervalo de requisição dos serviços.
- Cenário B: Variação da duração dos serviços.
- Cenário C: Variação do controle de admissão de chamadas.

O modelo utilizado para os três cenários foi o mesmo, e será detalhado a seguir.

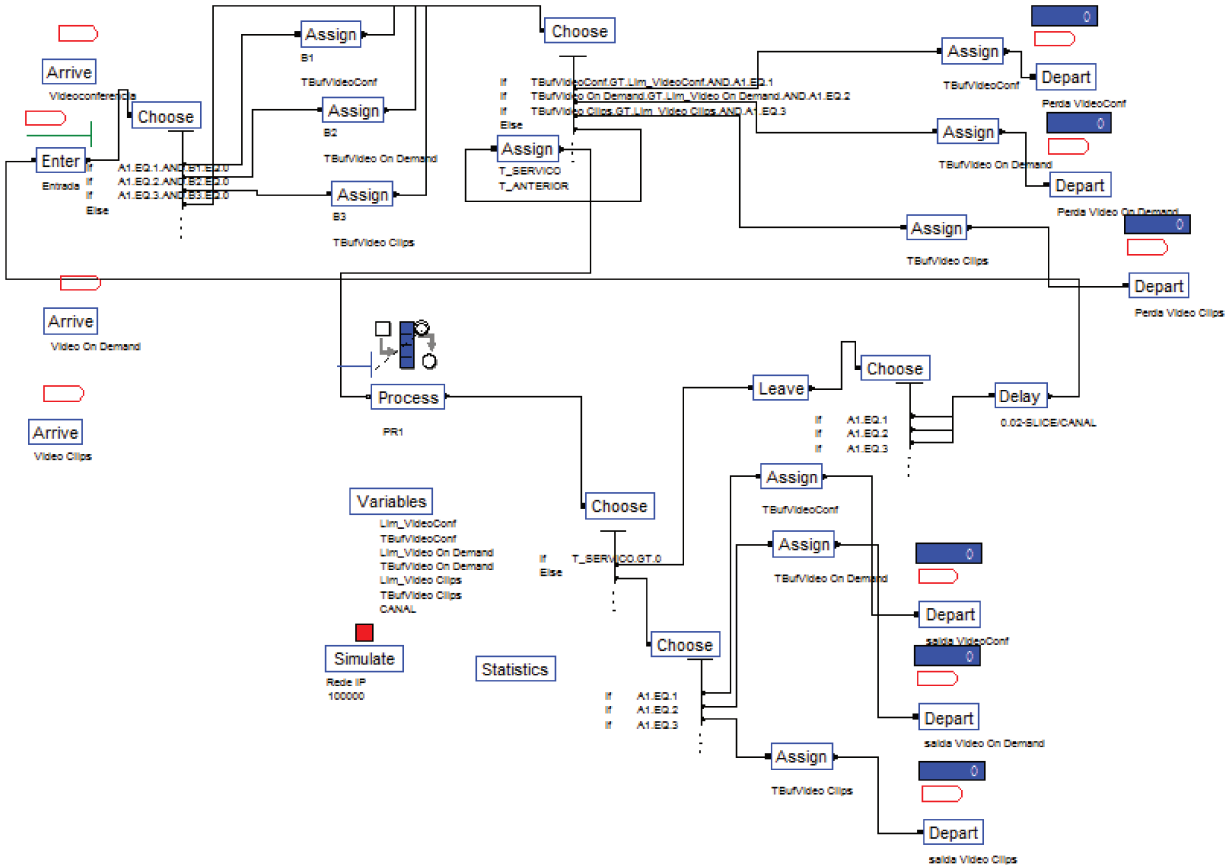


Figura 14: *Workspace* do ARENA. Modelo de simulação 1.

5.1.1 O modelo

O modelo possui três blocos *Arrive*, que são os responsáveis pela geração dos serviços, Videoconferência, Vídeo-*Clip*, e Vídeo *On Demand.* Cada bloco possui valores que representam seus serviços e funções de distribuição. Para cada serviço existem duas saídas, uma para os serviços rejeitados e outra para os completados. Levando a um total de seis saídas.

Após a geração do serviços, o bloco *Choose* verificará qual serviço que foi gerado e encaminhará ao respectivo bloco *Assign* do serviço. O bloco *Assign* irá então incrementar a variável *TbuffVideoConf* por exemplo, indicando que um serviço de Videoconferência está sendo solicitado e ocupará mais um espaço no sistema. Em seguida um outro bloco *Choose*, verifica as variáveis *TbuffVideoConf* e a *Lim_VideoConf*, a variável *Lim_VideoConf* é a responsável pelo Controle de Admissão de Chamada (CAC) do serviço em particular. Caso o valor do *TbuffVideoConf* seja maior que o aceitável pelo CAC, o serviço será bloqueado e encaminhado para o *Assign* responsável pela saída dos serviços rejeitados, e o respectivo contador será incrementado. Por outro lado, caso seja aceito, o bloco *Assign* irá decrementar uma parte do tempo total da duração do serviço, duração que foi determinado pela função de distribuição do bloco *Arrive* que criou o serviço. Após decrementar a duração do serviço, chegamos no bloco *Process*, que é o responsável pelo processamento. Nele teremos

influências do *codec* do serviço, capacidade de processamento, e regras de filas. Todo esse processo simulará a latência do servidor ou *host*. Por final, um último bloco *Choose,* analisa o tempo de duração restante para completar o serviço. Caso a duração não tenha terminado é feito um *loopback* repassando todos os processos novamente até que a duração do serviço tenha acabado, indicando o encerramento do serviço.

As variáveis globais são inseridas no bloco *Variables.* São variáveis de controle, como o CAC de cada serviço, e o tamanho do canal. Mais detalhes sobre os blocos se encontram no apêndice A.

5.1.2 Configurações das simulações

Para a realização das simulações foi utilizado diferentes funções de distribuições. Para o serviço videoconferência utilizamos a distribuição exponencial para gerar a duração do serviço e o intervalo entre requisições dos serviços. No vídeo *clip* e vídeo *on demand* foram utilizados a distribuição normal no intervalo entre requisições do serviços, e exponencial para a duração do serviço. O canal foi configurado a 384 kbps, e utilizamos replicações igual a 1 e 5. Na Tab. 6 estão os parâmetros iniciais de cada serviço.

- Videoconferência: Foi utilizado a função de distribuição exponencial no intervalo entre requisições do serviço e duração do serviço.
- Vídeo *On Demand*: Foi utilizado função de distribuição exponencial no intervalo entre requisições do serviço. E função de distribuição normal na duração do serviço.
- Vídeo *Clip*: Foi utilizado função de distribuição exponencial no intervalo entre requisições do serviço. E função de distribuição normal na duração do serviço.

Parâmetros	Vídeo *clip*	Vídeo *on-demand*	Videoconferência
Duração do quadro ou tempo entre pacotes (ms)	20	20	20
Tamanho máximo do pacote(bytes)	92	92	92
Taxa *Codec* (kbps)	32	32	32
Tamanho do pacote (bytes)	80	80	80
Duração média do serviço (segundos)	300	7200	900
Desvio padrão (segundos)	180	900	*
Intervalo médio de chegada dos serviços	900	14400	3600
Máximo de serviços simultâneos	4	2	2

Tabela 6: Valor dos parâmetros iniciais utilizados para a simulação.

No asterisco, o serviço de videoconferência esta seguindo um distribuição exponencial, dessa forma não possui um desvio padrão.

O ARENA não possui um campo especifico para inserir o *codec* do serviço, logo temos que atribuir para cada serviço o tamanho do quadro. O tamanho do quadro é utilizado quando o serviço é processado no bloco *process*, para que possa calcular o tempo que o serviço deve permanecer no bloco até que todo o tempo respectivo ao *codec* tenha se esgotado, significando que a taxa do *codec* foi satisfeita. A seguir veremos o calculo para os três serviços simulados. Como os três serviços utilizam a

mesma taxa de *codec* o calculo será o mesmo:

$$Taxa\,do\,Codec\,(\frac{bit}{s})\times Duração\,do\,quadro\,(s)=Slice\,(bit)\,. \tag{16}$$

Para um *codec* de 32 kbps temos:

$$32\,kbps\rightarrow 32000\,\frac{bit}{s}\times(20\times 10^{-3}\,s)=640\,bits\,. \tag{17}$$

No bloco *process* temos:

$$\frac{Tamanho\,do\,quadro}{Taxa\,do\,canal}=\frac{Bit}{Bit/s}=segundos\,. \tag{18}$$

Para simularmos o estudo proposto foi realizado a variação de um parâmetro por vez, com o objetivo de causar um colapso no sistema. Dessa forma conseguimos traçar gráficos com a quantidade de serviços completados e rejeitados em função do intervalo entre requisições, duração e CAC dos serviços. E através dele, notar os pontos que o *QoS* deixa de ser satisfatório, onde o número de usuário esta muito acima do que o sistema suporta, não conseguindo reservar recurso para a demanda.

Os parâmetros da Tab. 6 servem como uma referencia para a simulação. Ao se encerrar o estudo de um cenário, voltamos com os valores contidos na Tab. 6 para começarmos o estudo de um outro cenário.

5.1.3 Resultados

Cenário A

No cenário A realizamos simulações variando o intervalo de requisição de chegada dos serviços. Como temos três serviços, foram simulados três casos. Quando ocorre a variação do intervalo de chegada do serviço de videoconferência, quando ocorre do vídeo *clip* e quando ocorre do vídeo *on demand*. Os casos foram simulados focando um serviço por vez, para que possamos ver o comportamento de cada serviço separado. Existem inúmeros casos e cenários possíveis a serem simulados, mas como a nossa ideia não é focarmos em apenas um modelo de simulação, tivemos que fazer uma seleção dos que mais agregam informações.

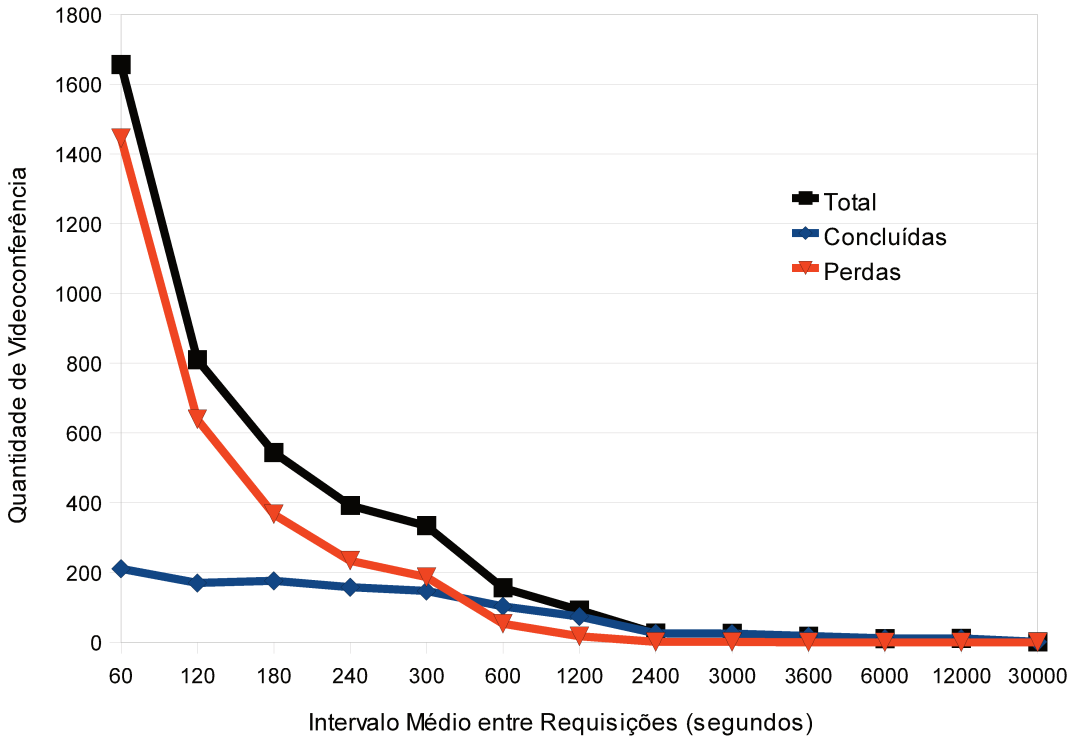


Figura 15: Quantidade de videoconferências em função do intervalo médio entre requisições. A linha preta representa o número total de videoconferências, a azul as concluídas, e a vermelha as perdas. A duração média da videoconferência foi fixada em 900 segundos e CAC igual a 2.

Como podemos ver o comportamento das linhas preta e vermelha da Fig. 15 estão seguindo uma distribuição exponencial como esperado. Nos intervalos de requisições mais frequentes o sistema não consegue garantir recursos para todos os usuários. Isso se deve ao CAC e a duração média do serviço. Como a duração foi configurada em 900 segundos e o sistema suporta apenas 2 chamadas simultâneas ocorre uma bloqueio de chamas de até 87,5% para o pior caso, 60 segundos. Apenas a partir de 2400 segundos os bloqueios ficam aceitáveis, com menos de 2% de bloqueio, que é um valor padrão aceitável.

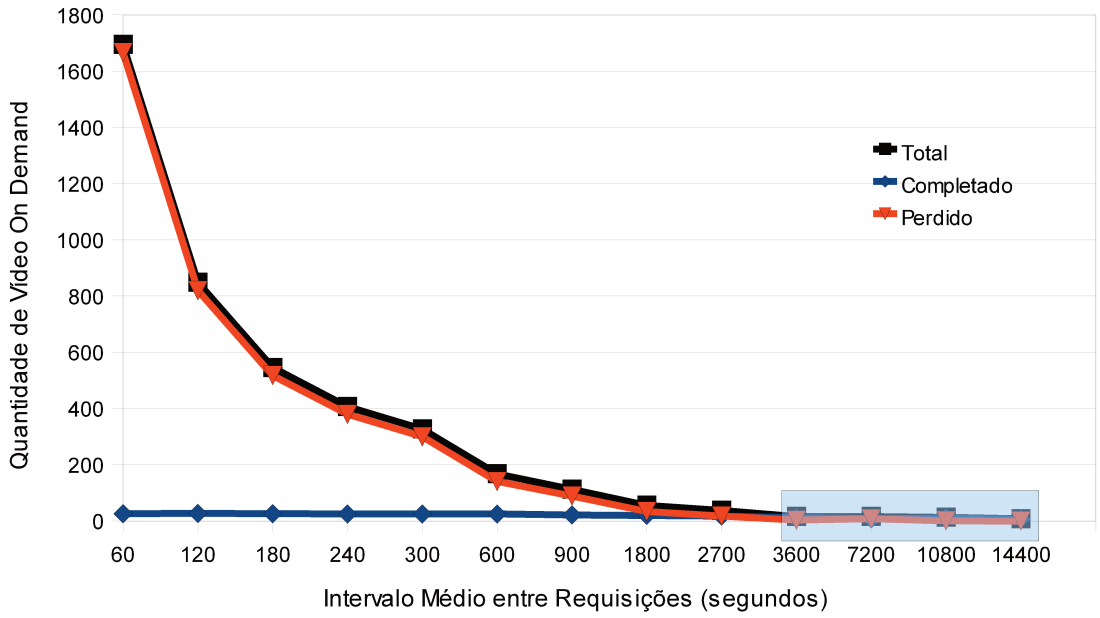


Figura 16: Quantidade de vídeos *on demand* em função do intervalo médio entre requisições. A linha preta representa o número total de vídeos *on demand*, a azul as concluídas, e a vermelha as perdas. A duração média do vídeo *on demand* foi fixada em 7200 segundos com desvio padrão de 900 segundos, utilizando função de distribuição normal e CAC igual a 2.

No caso do vídeo *on demand* o número de serviços completados foram mínimos, devido a duração elevada do serviço, chegando a um bloqueio de 98,3% para 60 segundos. O destaque na Fig. 16, mostra os intervalos entre requisições de serviço que começam a ser aceitáveis. Na Fig. 17 podermos analisar-lo melhor, ampliando a região.

Foram feitas simulações com valores maiores do que 3600 segundos para podermos chegar ao valor em que a taxa de bloqueio seja inferior a 1%. Podemos notar que mesmo aumentando o intervalo entre requisições do serviço pode ocorrer casos em que o bloqueio aumente comparado com um intervalo menor, como é o caso do 18000 segundos. Isso ocorre porque um novo serviço foi gerado e o sistema já contem 2 serviços em execução, devido a durações elevadas que um serviço de vídeo *on demand* pode durar.

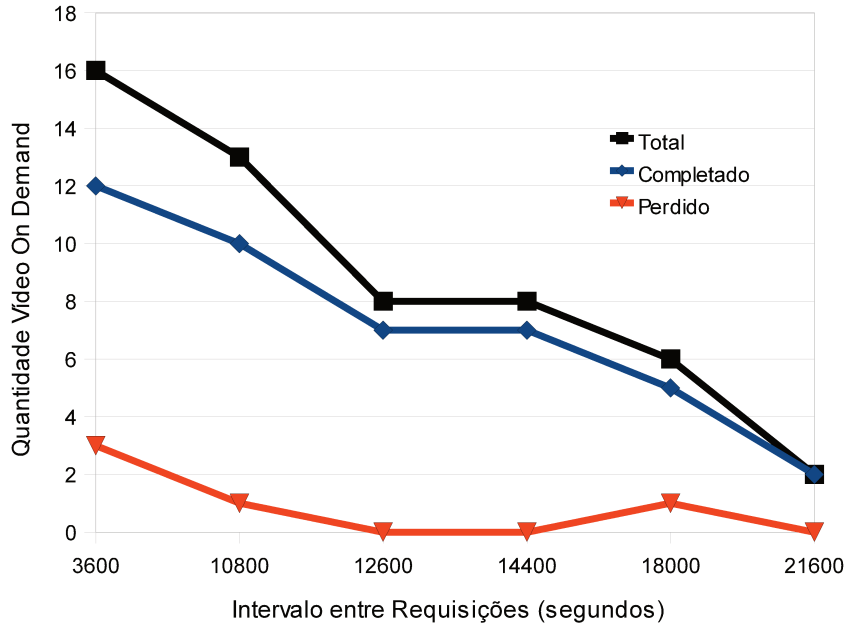


Figura 17: Trecho destacado na Fig. 16. Comportamento do vídeo *on demand* entre o intervalo 3600 a 21600 segundos.

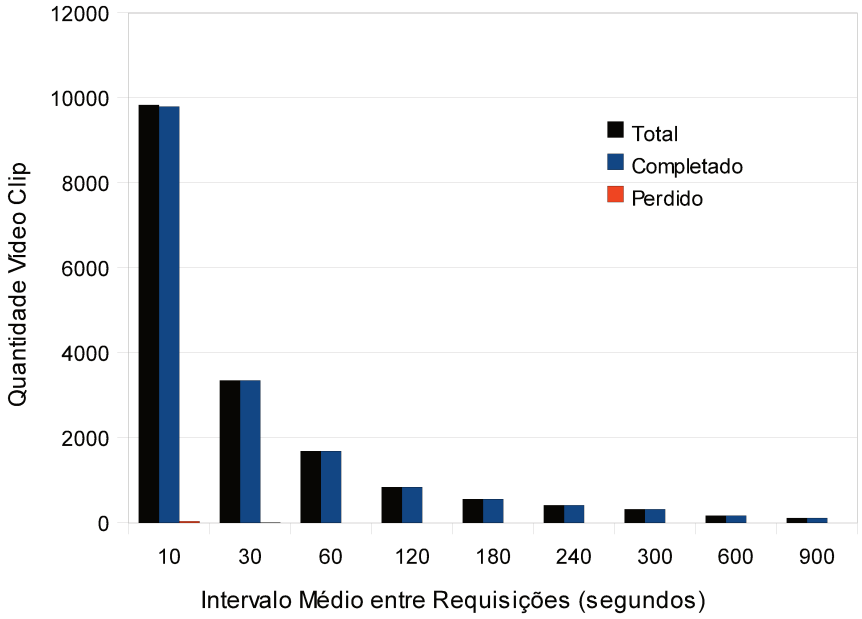


Figura 18: Quantidade de vídeo *clips* em função do intervalo médio entre requisições. A coluna preta representa o número total de vídeo *clips*, a azul as concluídas, e a vermelha as perdas. A duração média do vídeo *clips* foi fixada em 300 segundos e desvio padrão de 180 segundos, utilizando a distribuição normal e CAC igual a 4.

Por possuir um CAC maior que dos demais serviços, e uma duração de serviço pequena, o vídeo *clips* foi o único serviço que conseguiu chegar a padrão aceitáveis de bloqueio. Sendo quase todos os intervalos entre requisições de serviço sem bloqueios. Apenas houve bloqueio em intervalos muito baixos, como 10 segundos e 30 segundos. A taxa de serviços completados foi de 99,6% para 10 segundos, e 99,91% para 30 segundos.

Cenário B

No cenário B realizamos simulações variando a duração dos serviços. Temos três casos, cada um referente a um serviço.

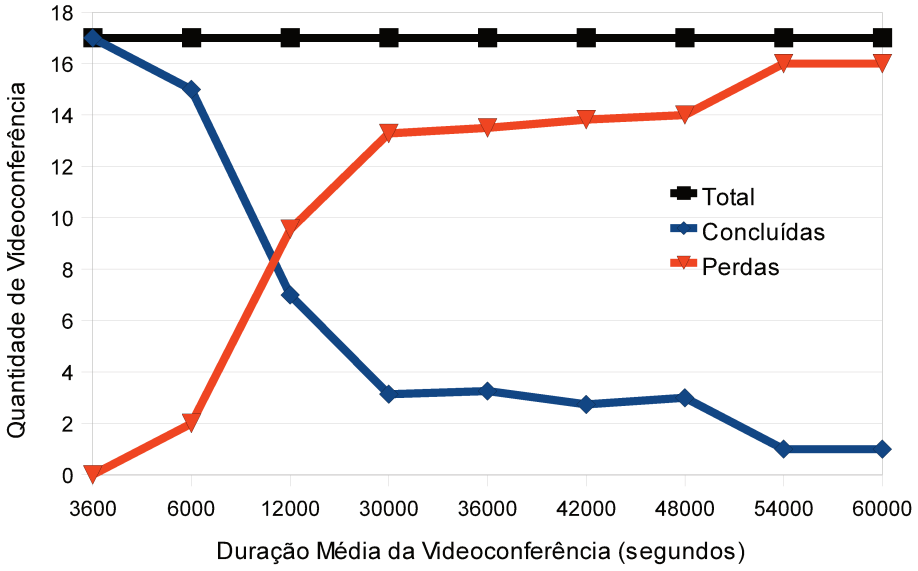


Figura 19: Quantidade de videoconferências em função da duração média da videoconferência. A linha preta representa o número total de videoconferências, a azul as concluídas, e vermelha as perdas. O intervalo médio entre requisições da videoconferência foi fixado em 3600 segundos e CAC igual a 2.

As durações menores que 3600 segundos não estão presentes na Fig. 19 porque ocorreu uma repetição dos resultados entre 60 a 3600 segundos, total de 17 serviços, 17 completados e 0 perdido. Isso se deve principalmente porque o intervalo entre requisições dos serviços é de 3600 segundos. Logo, a duração dos serviços são pequenas demais para o intervalo dos serviços, deixando o sistema ocioso para novas requisições de serviço.

Na Fig. 20, podemos notar que para a duração de 7200 segundos o número total de serviços não é igual a soma dos serviços completados e perdidos. Frequentemente isso ocorre devido ao tempo total da simulação que se encerra antes que o tempo do serviço tenha sido completado. Podemos considerar poe exemplo esse caso como um serviço corrompido, onde um *link* se rompe durante uma transferência ou o *host* é desligado.

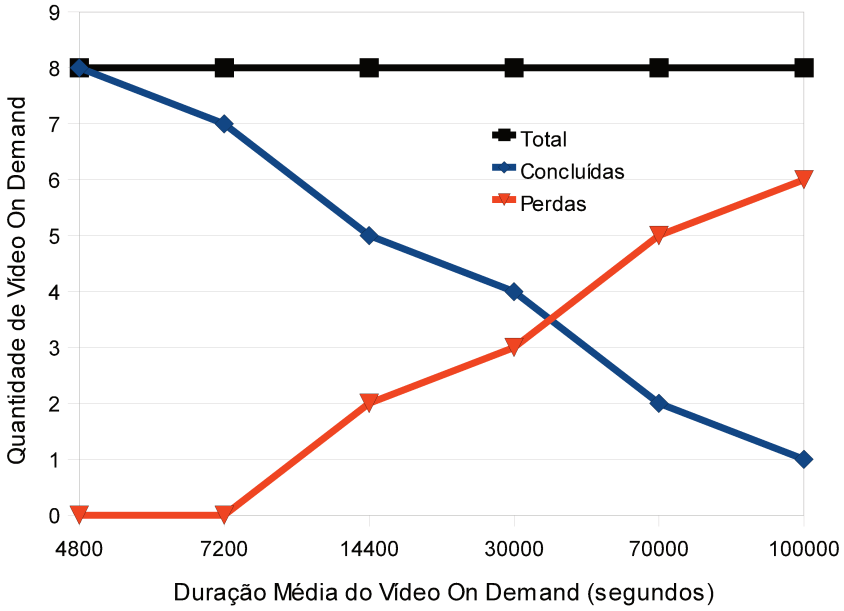


Figura 20: Quantidade de vídeos *on demand* em função da duração média do vídeo *on demand*. A linha preta representa o número total de vídeos *on demand*, a azul as concluídas, e a vermelha as perdas. O intervalo entre requisições do serviço foi fixado em 14400 segundos e CAC igual a 2.

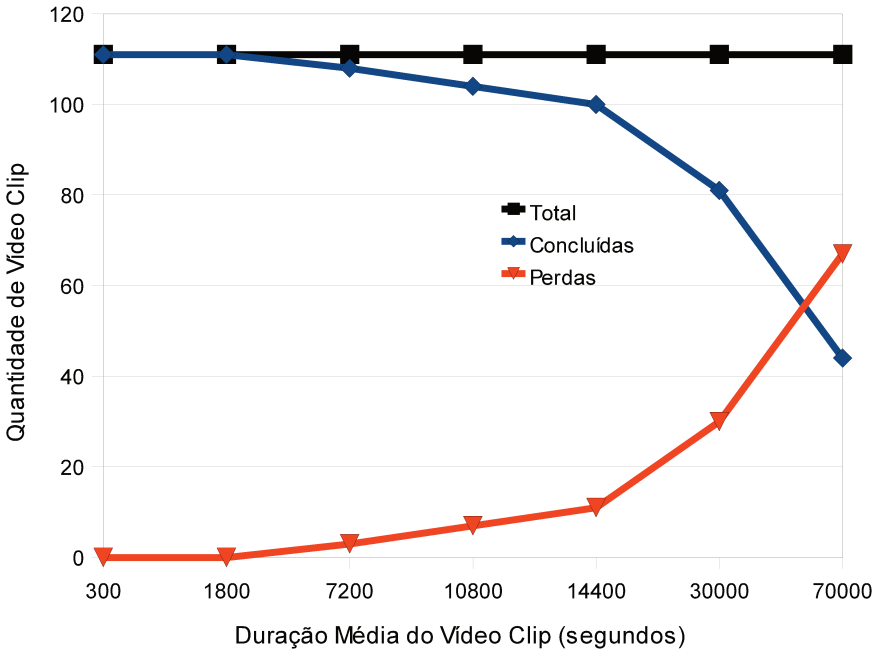


Figura 21: Quantidade de vídeo clips em função da duração média do vídeo clips. A linha preta representa o número total de vídeo clips, a azul as concluídas, e a vermelha as perdas. O intervalo entre requisições do serviço foi fixado em 900 segundos, e CAC igual a 4.

Cenário C

No cenário C foi realizado simulações variando o controle de admissão de chamadas. Esse cenário reflete uma ação dos administradores da rede, com o intuito de melhorar a estrutura física da rede para que possa suportar a nova demanda de recursos dos usuários. Como a instalação de um número maior de troncos em um central de comutação.

Através das simulações deste cenário podemos analisar a quantidade de CAC necessário para suportar os serviços de forma que não haja um número de perdas elevado.

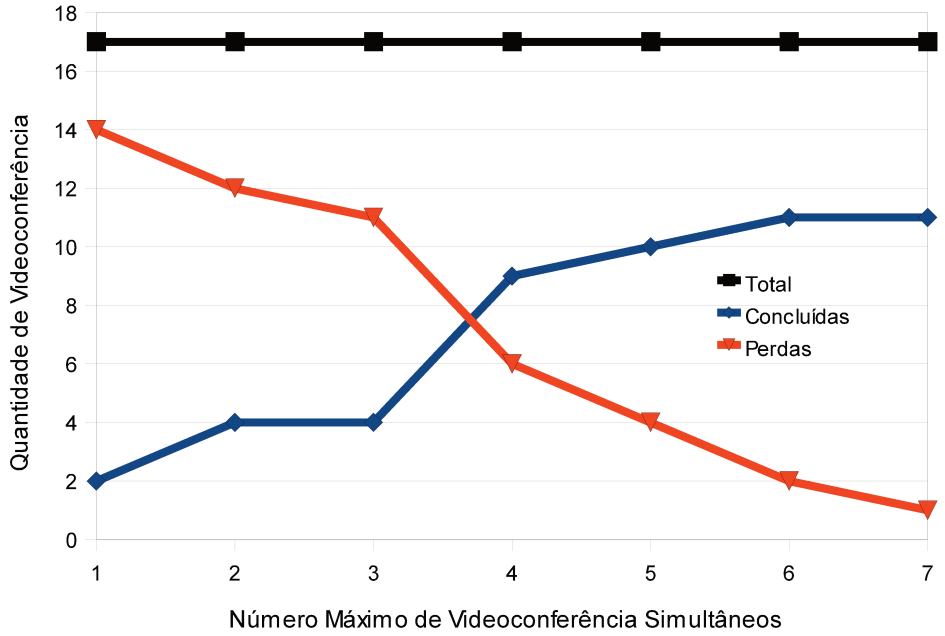


Figura 22: Quantidade de videoconferências em função ao número máximo de videoconferência simultâneos. A linha preta representa o número total de videoconferência, a azul as concluídas, e a vermelha as perdas. A duração média e intervalo entre requisições dos serviços são 30000 e 3600 segundos respectivamente. Um valor exagerado de duração de serviço foi utilizado para podermos visualizar o efeito das mudanças no número máximo de serviços simultâneos.

Mantendo a média do intervalo entre requisições e duração dos serviços fixos, podemos variar o número máximo de serviços simultâneos. Podemos ver a supressão dos serviços rejeitados e o restabelecimento dos serviços concluídos restabelecida conforme aumentamos o número máximo de serviços simultâneos.

O intervalo médio entre requisições de serviços da Fig. 23 é o dobro da duração do serviço, resultando em 100% de serviços concluídos quando temos dois ou mais serviços simultâneos. Devemos lembrar que uma rede está sempre em constante crescimento, sendo positivo considerar margens de seguranças para que picos ou um aumento repentino na demanda não comprometa a rede.

No caso do serviço de vídeo clips, mostrado na Fig. 24 segue as mesmas funções de distribuição da Fig. 23, sua diferença são os valores do intervalo médio entre requisições e duração do serviço. Neste caso, podemos notar que a rede se encontra ociosa em intervalos longos, porque em nenhum momento é gerado dois serviços de vídeo *clip* ao mesmo tempo.

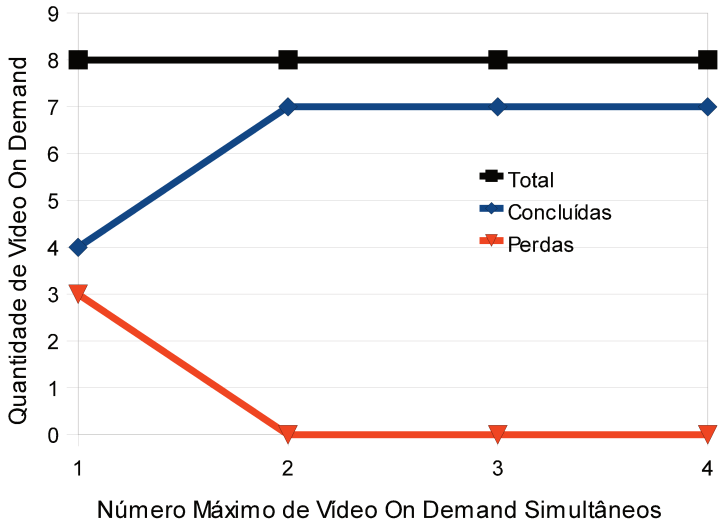


Figura 23: Quantidade de vídeos *on demand* em função ao número máximo de vídeo *on demand* simultâneos. A linha preta representa o número total de vídeos *on demand*, a azul as concluídas, e a vermelha as perdas. A duração média e intervalo entre requisições dos serviços são 7200 e 14400 segundos respectivamente.

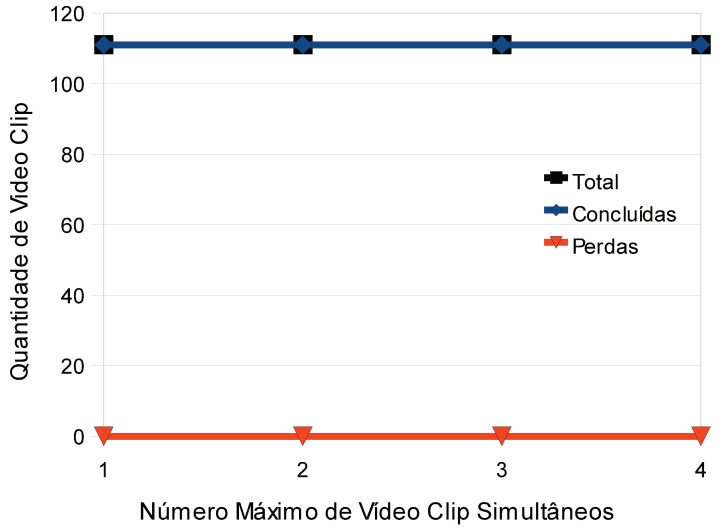


Figura 24: Quantidade de vídeo *clips* em função do número máximo de vídeo *clips* simultâneos. A linha preta representa o número total de vídeo *clips*, a azul as concluídas, e a vermelha as perdas. A duração média e intervalo entre requisições dos serviços são 300 e 900 segundos respectivamente.

5.2 Modelo de Simulação 2

O modelo supõe o dimensionamento de uma rede com 6 tipos diferentes de serviços, sendo 5 do tipo *stream* e apenas um do tipo elástico (chamado de Dados), como segue a tabela a seguir.

Serviço	Intervalo entre Requisições	Duração (s)	Tráfego (Erl)
Videoconferência	1 chamada/h	900	0.25
Vídeo *On Demand*	0.25 chamada/h	7200	0.50
Dados	1 arquivo/300s	(*)	35
Vídeo *Clips*	4 chamadas/h	300	0.33
VoIP (PABX)	100 chamadas/h	180	5
Câmera IP	1 quadro/s	1	1

Tabela 7: Parâmetros de intervalo, duração e tráfego dos serviços.

No asterisco, o serviço de Dados são gerados com uma média de tamanho igual a 84 kbytes, próximo ao de [26], onde a transmissão depende da velocidade da rede. E também, a média do tamanho do serviço gerado pelo serviço de Dados é determinado dividindo 84000x8 por 64 kbit/s correspondente à granularidade mínima do canal de voz sem compressão, resultando em 10.5s. Note que 0.035 Erl = (1/300).(84000.8/64000). Cada serviço individual VoIP é um PABX que corresponde a 5 Erl.

Assumindo que os serviços gerados pelo VoIP e videoconferência possuem características semelhantes ao serviço de voz por comutação de circuitos, podemos ajustar a duração desses serviços pela distribuição de probabilidade exponencial [27]. A partir de pressupostos semelhantes, nós adotamos que o intervalo entre as requisições dos serviços seguem a distribuição exponencial.

Os serviços vídeo *on demand* e vídeo *clip* também seguem a distribuição exponencial para os intervalos entre requisições. Entretanto, a distribuição de duração do serviço tem características Gaussinas. No caso do vídeo *on demand*, a média de duração é de 7200 segundos com um desvio padrão de 900 segundos. E no caso vídeo clips a media de duração é de 300 segundos com um desvio padrão de 100 segundos. O modelo assume que o serviço Câmera IP requer uma banda de 32 kbit/s (*codec* MPEG-4), para transmitir um quadro por segundo em uma única direção.

O sistema será simulado com dois tipos de cenários: sem prioridade e com prioridade. No caso sem prioridade todos os serviços terão a mesma prioridade. Para o caso com prioridade todos os serviços *stream* terão alta prioridade, e o serviço elástico (Dados) terá baixa prioridade. Dessa forma, enquanto os serviços *stream* são designados para ter um bloqueio máximo de 2%, o de Dados, com baixa prioridade, não poderá ter *delays* maiores que 15 segundos na entrega de um arquivo de 100 kbytes (equivalentemente a 84 kytes em 12.6 segundos).

5.2.4 Detalhamento do Modelo

O modelo simulado pode ser visto como uma estacão de trabalho ou um servidor que prove serviços a usuários ou entidades. No caso do tráfego *stream*, o conceito de duração do serviço é acompanhado pelo conceito de largura de banda efetiva, que significa quanto da largura de banda é

alocada para o serviço durante o seu tempo ativo.

A duração do serviço no caso elástico depende do tamanho do canal atribuído a ele e do atributo TAMANHO. O atributo TAMANHO é uma variável de valor a partir de funções de distribuição que contem a quantidade de bits do serviço de Dados.

O serviço é processado no servidor (bloco *Process*) atendendo uma fila de pacote de acordo com o protocolo de enfileiramento FIFO. Na fila de um único servidor há o tipo de serviço M/G/1 FIFO ou prioridade *Head-of-Line* (FIFO em cada fila individual).

Para o tratamento de cada serviço individual (uma fatia ou pacote), o processo continuará até que tenha esgotado o seu tamanho (convertido em tempo dividindo pelo canal) estipulado, ou até que todo o serviço tenha sido transmitido pelo canal.

Esta maneira de converter o tamanho do serviço no tempo de duração é indicado por sua representação de ser independente de fatores como a velocidade com que um servidor envia a informação para a rede, a velocidade de transmissão do canal, etc. Vale ressaltar que a duração do serviço é a soma do tempo de transmissão de todas as fatias que compõem os tempos de espera para a transmissão de suas fatias sucessivas.

O tamanho de cada fatia (ou pacote) é uma característica de cada serviço e, neste modelo de simulação, por simplicidade, supõe-se que o comprimento médio de serviço do tráfego elástico corresponde a cerca de 100 cortes, resultando em uma fatia do tamanho de 7000 bits.

Por outro lado, o tráfego *stream* utiliza um *codec* de 32 kbit/s, sendo que o intervalo entre os pacotes consecutivos não podem ultrapassar 20 milissegundos, o tamanho da fatia será de 32 [kbit/s].20 [ms] = 640 bits.

5.2.5 Resultados do modelo analítico

Os resultados do modelo serão apresentados nesta seção. O tempo valido das simulações foi de 15000 segundos, e um total de 10 replicações para cada simulação. Para obter o resultado final de cada simulação temos que fazer a média aritmética das 10 replicações. A quantidade de replicações é uma ferramenta importante para a convergência dos resultados.

É importante destacar que por estarmos utilizando a versão de estudante do software ARENA, algumas simulações não foram possíveis de executar por serem demasiadas robustas, e ultrapassarem o limite permitido de 150 entidades. Dessa forma conseguimos simular até 85 Erlang para este modelo de simulação.

O volume do tráfego influencia os requisitos de desempenho. Outro fator extremamente importante é a influencia do uso de prioridades para os serviços, que influencia o desempenho do sistema. Como o modelo analítico não considera a prioridade, é essencial usar o modelo de simulação para notarmos a sua influencia.

As simulações propostas visam a análise do *sojourn* (tempo de processamento + tempo de serviço) do tráfego elástico (Dados) quando ocorre um aumento significativo no volume do tráfego *stream* (VoIP). Realizamos as simulações da seguinte forma:

- A) Com prioridade, variando o serviço VoIP de 40 Erlang até 85 Erlang, sem CAC.
- B) Sem prioridade, variando o serviço VoIP de 40 Erlang até 85 Erlang, sem CAC.
- C) Repetimos novamente A e B, mas com CAC, e serviço VoIP fixo em 70 Erlang.

Além das simulações com prioridade e sem prioridade utilizando a função de distribuição exponencial, iremos realizar os mesmos processos com duas outras distribuição. Elas são, Hiper-exponencial, com o objetivo de simular um fluxo de rajadas no serviço elástico, e Pareto. Dessa forma teremos dois padrões diferentes, com e sem fluxos em rajadas. Para isso, a quantidade de bits gerados pelo atributo TAMANHO irá variar de acordo com as novas distribuições gerando o fluxo em rajada.

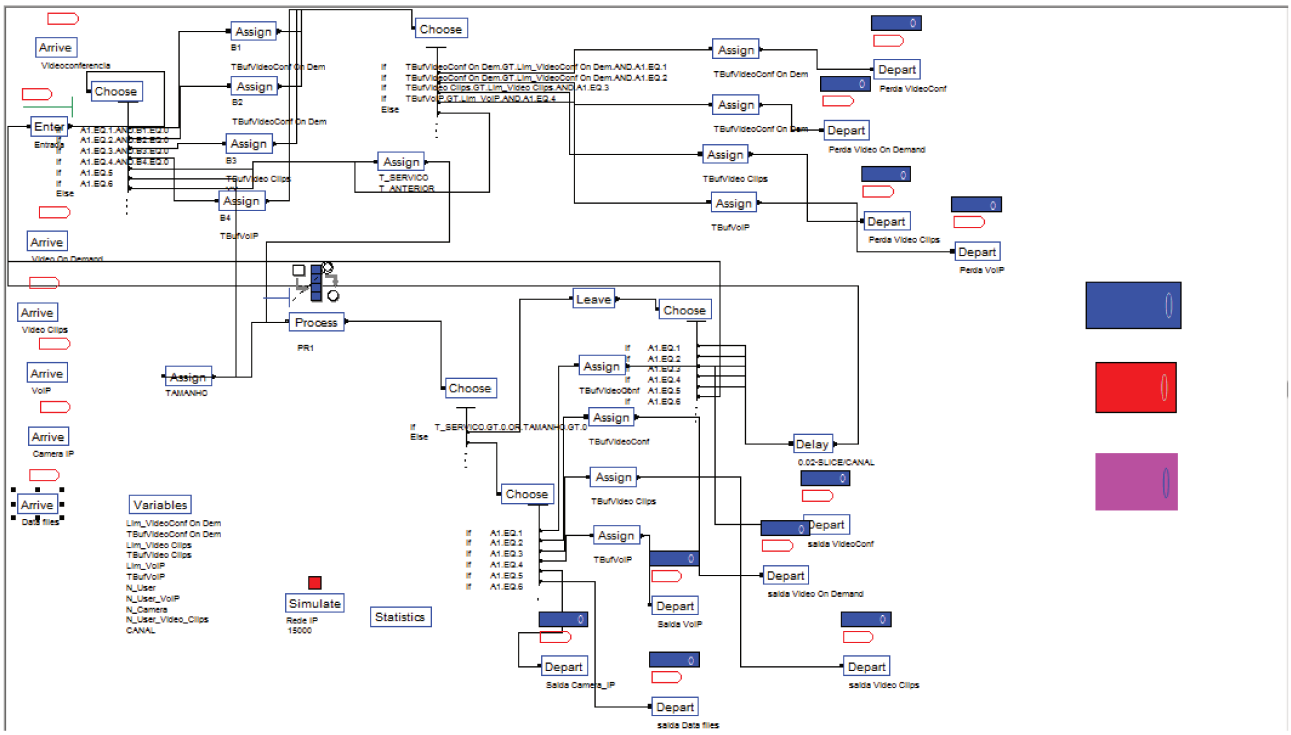


Figura 25: *Workspace* do ARENA. Modelo de simulação 2.

Como ilustra a Fig. 25, os blocos do modelo permanecem com algumas modificações, novos blocos foram inseridos representando os seis serviços, blocos de controle e de saída. Por outro lado a estrutura lógica do modelo permanece a mesma, com apenas a adição de novos atributos de controle.

5.2.5.1Utilizando Distribuição Exponencial ao Atributo TAMANHO do serviço de Dados

Caso A e B

	40	45	50	55	60	65	70	75	80	85
SemP	0,301	0,306	0,335	0,366	0,381	0,416	0,433	0,469	0,506	0,533
ComP	0,317	0,329	0,371	0,423	0,458	0,518	0,573	0,687	0,955	1,358

Tabela 8: Saídas do serviço de Dados com prioridade e sem prioridade. Variando de 5 em 5 Erlang o volume de tráfego do serviço VoIP. Sem prioridade (**SemP) t**odos os serviços estão utilizando a mesma prioridade. Com prioridade (**ComP)** o serviço de Dados possui uma prioridade inferior aos demais.

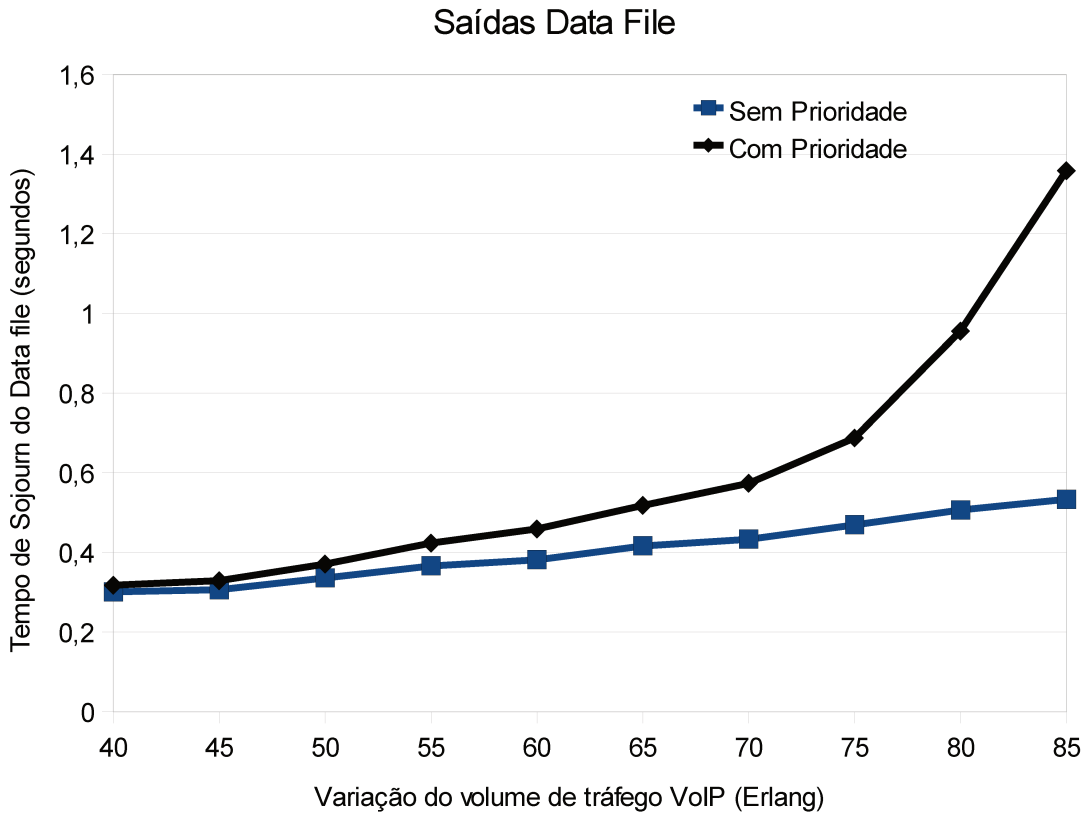


Figura 26: Tempo de *Sojourn* do serviço de Dados, dado em segundos em função a variação do volume de tráfego VoIP em Erlang. Na linha preta o serviço de Dados está com prioridade e na linha azul sem, variando de 5 em 5 Erlang o volume de tráfego do serviço VoIP.

Como vimos, o perfil de um tráfego *stream* é diferente de um tráfego elástico. Em um ambiente com QoS a priorização do tráfego da rede é fundamental. A Fig. 26 mostra o *sojourn* o tempo que leva para um quadro do serviço de Dados chegar ao seu destino levando em consideração que o serviço de Dados é um serviço elástico, portanto utiliza o protocolo de transporte TCP.

A linha preta, com prioridade, indica que os serviços *stream* possuem uma prioridade maior com relação ao elástico, dessa forma acarretando em um aumento do *throughtput* do serviço de Dados.

Por outro lado, a linha em azul, sem prioridade, indica que todos os serviços possuem a mesma prioridade para utilização dos recursos. Como podemos notar a taxa do *sojourn* diminui significativamente mesmo com 85 Erlang quando não há prioridade.

Caso C

Da mesma forma que as demais simulações, a seguir encontra-se as 10 replicações para o caso C, 70 Erlang VoIP. O valor do CAC sem prioridade e com prioridade são os mesmos, eles são:

Serviço	CAC
Vídeo *On demand*	14
Videoconferência	14
Vídeo Clip	8
VoIP	85
Câmera IP	(*)
Data file (Dados)	(**)

Tabela 9: Número máximo de serviços simultâneo. O serviço câmera IP possui um asterisco porque é um serviço contínuo, não envolvendo parâmetros como CAC e prioridade. E o serviço de Dados, é influenciado pelo volume do tráfego *stream*.

No asterisco, o serviço Câmera IP não possui um CAC porque o serviço é contínuo. Já no caso Data file (Dados) o serviço é elástico, dessa forma sempre que haver recursos suficientes será gerado um novo serviço.

Sem Prioridade	1	2	3	4	5	6	7	8	9	10	Média
Saída Vídeo On Demand_Ta	7625,6	7398,9	7463,6	7861,2	8128,2	7154,4	7867,7	7711,6	7042,6	7359,1	7561,29
obs Vídeo On Demand_Ta	4	1	4	3	4	3	2	2	4	4	3,1
Saída Camera_IP_Ta	1,0022	1,0021	1,0022	1,0021	1,0022	1,0021	1,0022	1,0021	1,0022	1,0021	1,00215
obs Camera_IP_Ta	140000	140000	140000	140000	140000	140000	140000	140000	140000	140000	140000
Saída Data files_Ta	0,39507	0,38753	0,36759	0,4312	0,40247	0,36959	0,39683	0,47525	0,3798	0,43598	0,404131
obs Data files_Ta	157	175	172	143	138	134	153	151	129	162	151,4
Saída Videoconf_Ta	881,17	1993,6	1222,2	1210,1	1091,5	912,18	985,66	953,02	1215,6	778,71	1124,374
obs Videoconf_Ta	9	12	8	9	9	11	11	10	10	7	9,6
Saída VoIP_Ta	179,46	179,28	178,11	181,09	176,56	176,96	181,07	180,59	179,47	180,75	179,334
obs VoIP_Ta	5416	5375	5453	5379	5343	5423	5406	5374	5369	5440	5397,8
Saída Video clips_Ta	305,4	294,85	313,7	306,33	299,93	289,92	293,42	290,21	295,7	304,13	299,359
obs Video clips_Ta	168	143	163	139	149	174	149	157	142	161	154,5

Tabela 10: Resultados das 10 replicações e a media de cada serviço. Todos os serviços estão utilizando a mesma prioridade. As Saídas *Tipo*_Ta representam o tempo de *sojourn* dos serviços, dado em segundos. A saída *obs Tipo*_Ta são a quantidade total de serviço gerado.

Como podemos notar, ao comparar as Tabs. 10 e 11 vemos que apenas os valores do *sojourn* do serviço de Dados mudam. Isso se deve principalmente à dois fatores. Porque os serviços estão usando os mesmos intervalos entre requisições de serviços, duração de serviço e, funções de distribuições nas duas Tabela s. Outro fator é o tamanho do canal, no qual a banda passante é suficiente para que os serviços não se alterem ao diminuir a prioridade do serviço de Dados. Por outro lado o serviço de Dados se altera, porque ocorre uma mudança em sua prioridade, dando preferencia de tratamento a um tráfego *stream*.

Podemos notar também que a quantidade total de serviços de Dados é o mesmo nas duas Tabela s (Tabela 10 e 11), mudando apenas o *sojourn*. Isso pode ter ocorrido porque a diferença do *sojourn*, com prioridade e sem prioridade, foi pequena, não sendo suficientemente grande para poder ter gerado mais serviços no intervalo de tempo que durou a simulação.

Com Prioridade	1	2	3	4	5	6	7	8	9	10	Média
Saída Vídeo On Demand_Ta	7625,6	7398,9	7463,6	7861,2	8128,2	7154,4	7867,7	7711,6	7042,6	7359,1	7561,29
obs Vídeo On Demand_Ta	4	1	4	3	4	3	2	2	4	4	3,1
Saída Camera_IP_Ta	1,0022	1,0021	1,0022	1,0021	1,0022	1,0021	1,0022	1,0021	1,0022	1,0021	1,00215
obs Camera_IP_Ta	140000	140000	140000	140000	140000	140000	140000	140000	140000	140000	140000
Saída Data files_Ta	0,48998	0,47287	0,4492	0,53579	0,49041	0,45352	0,49322	0,57805	0,47358	0,53792	0,497454
obs Data files_Ta	157	175	172	143	138	134	153	151	129	162	151,4
Saída Videoconf_Ta	881,16	1993,6	1222,2	1210,1	1091,5	912,18	985,67	953,02	1215,6	778,71	1124,374
obs Videoconf_Ta	9	12	8	9	9	11	11	10	10	7	9,6
Saída VoIP_Ta	179,43	179,29	178,11	182,09	176,56	176,96	181,07	180,59	179,47	180,75	179,432
obs VoIP_Ta	5416	5376	5453	5379	5342	5423	5406	5374	5369	5440	5397,8
Saída Video clips_Ta	305,4	294,85	313,7	306,33	299,93	289,92	293,43	290,21	295,7	304,12	299,259
obs Video clips_Ta	168	143	163	139	149	174	149	157	142	161	154,5

Tabela 11: Resultados das 10 replicações e a media de cada serviço. O serviço de Dados está com uma prioridade inferior aos demais serviços. As Saídas *Tipo*_Ta representam o tempo de *sojourn* dos serviços, dado sem segundos. A saída *obs Tipo*_Ta são a quantidade total de serviço gerado. Ao compararmos o *sojourn* dos casos A, B e C chegamos na seguinte tabela.

Casos	*Sojourn*
Caso A, com prioridade, sem CAC	0,573
Caso B, sem prioridade, sem CAC	0,433
Caso C, sem prioridade, com CAC	0,404
Caso C, com prioridade, com CAC	0,497

Tabela 12: Comparação do *sojourn* caso A,B e C do serviço de Dados, distribuição exponencial. Utilizando 70 Erlang.

Como esperado, em todos os casos o tempo de *sojourn* do serviço de Dados diminui ao aplicarmos CAC no tráfego *stream*. Já com a priorização do tráfego, o *sojourn* do tráfego elástico aumenta, porque a prioridade utilizada favorece tráfegos *stream*.

5.2.5.2 Utilizando Distribuição Hiper-exponencial ao Atributo TAMANHO do serviço de Dados

Como já mencionado, mudamos a função de distribuição exponencial para hiper-exponencial do serviço de Dados afim de simular um tráfego elástico com rajadas. Dessa forma os tamanhos dos quadros terão um comprimento diferente, variando os tempos necessário de processamento mais o de transferência (*sojourn*). Para um fator de comparação, realizamos as simulações seguindo os mesmos casos A, B e C, para 70 Erlang.

Casos	*Sojourn*
Caso A, com prioridade, sem CAC	0,609
Caso B, sem prioridade, sem CAC	0,447
Caso C, sem prioridade, com CAC	0,414
Caso C, com prioridade, com CAC	0,515

Tabela 13: Comparação do *sojourn* caso A,B e C do serviço de Dados, distribuição hiper-exponencial. Utilizando 70 Erlang.

Os resultados para todos os casos foram maiores, comparado aos da Tab. 12, por utilizarmos fluxo em rajada. A expressões que utilizamos para gerar a quantidade de bits que o quadro possui é dada por

$$f(x)=\lambda_1 e^{-\lambda_1 \times x_1}+\lambda_2 e^{-\lambda_2 \times x_2}, \quad (19)$$

onde $\lambda_1=1/(32000\times 8)$, $x_1=32000\times 8$, $\lambda_2=1/(205330\times 8)$, $x_2=205330\times 8$. Dessa forma conseguirmos simular um fluxo em rajadas com uma média próxima aos valores que a distribuição exponencial gera.

5.2.5.3 Utilizando Distribuição Pareto ao Atributo TAMANHO do serviço de Dados

Para simularmos a distribuição Pareto utilizamos sua expressão no formato exponencial [31],

$$Y=\log(\frac{X}{x_m}), \quad (20)$$

onde x_m é o valor mínimo da distribuição. Se Y segue uma exponencial de média $1/\alpha$,

$$x_m . e^{Y}, \quad (21)$$

segue uma distribuição de pareto de média $E[x]=84000\times 8$ (valor da média do serviço de Dados). Fazendo $x_m = 1$, temos $\alpha=E[x]/(E[x]-1)=84000\times 8/(8400\times 8-1)=1{,}0000014880975$.

Veja que mudam os valores das distribuições dos tamanhos dos serviços de Dados mas não muda a média (que deve continuar a ser 84000x8 bits). Na verdade, a distribuição de Pareto, que é de cauda longa, nesse caso não vai provocar rajadas, mas vai provocar variações mais acentuadas nos tamanhos serviços de Dados. Essas variações no tamanho também acabam provocando rajadas. A distribuição de cauda longa provocaria rajadas se estivesse no processo de chegadas (no entanto, o processo de serviço pode provocar maior variabilidade no tempo de atendimento e no tamanho da fila).

Casos	*Sojourn*
Caso A, com prioridade, sem CAC	0,001873
Caso B, sem prioridade, sem CAC	0,001816
Caso C, sem prioridade, com CAC	0,001809
Caso C, com prioridade, com CAC	0,001860

Tabela 14: Comparação do *sojourn* caso A,B e C do serviço de Dados, distribuição Pareto. Utilizando 70 Erlang.

5.3 Modelo de Simulação 3

O modelo de simulação 3 é uma continuação do modelo 2. Tanto os blocos como seus valores são os mesmos apresentados no modelo 2. Dessa forma o modelo analítico permanece o mesmo. A diferença entre os dois modelos é que o modelo 3 simula um ambiente VPN.

Uma *Virtual Private Network* (VPN) é uma forma segura de se conectar a uma rede privada através de um local remoto, utilizando a internet ou uma rede pública insegura para transportar os pacotes utilizando criptografia. O VPN é frequentemente utilizado por usuários remotos ou empresas que querem compartilhar informações e recursos de um mesmo ambiente de rede.

Como o VPN foi desenvolvido para acesso a redes remotas, a distancia entre os dois nós podem acarretar em múltiplos saltos entre roteadores, e meios físicos diferentes, até chegar ao seu destino. Esses fatores vão influenciar o QoS do sistema diretamente. Problemas de propagação, como *jitter*, latência e perda, que em uma rede local privada não chega a prejudicar a comunicação, por serem valores pequenos, agora, em um ambiente VPN podem aumentar muito o tempo de transferência de um arquivo ou até mesmo impedir a utilização de serviços.

Podemos realizar um teste simples utilizando a ferramenta "*Trace Router*" do Windows para identificar o tempo de propagação gasto entre dois pontos distintos no globo e o tempo de propagação dentro de uma rede local. Essa ferramenta mostra todos os saltos e, tempo de propagação total necessário para que o pacote chegue ao seu destino. Para demonstração utilizaremos dois sites, um no Brasil, e outro na Rússia.

A figura a seguir mostra os 12 saltos necessários até que o pacote chegue ao seu destino, o servidor da UOL. O primeiro salto é o roteador interno da LAN, portanto é uma rede interna privada. Podemos notar a diferença de tempo que leva o primeiro salto com os demais, sendo os demais saltos foram realizados na WAN. Por meio deste exemplo simples podemos notar a influencia do *jitter*, latência e perda de pacotes ao compararmos as 3 colunas que representam o tempo gasto em cada salto.

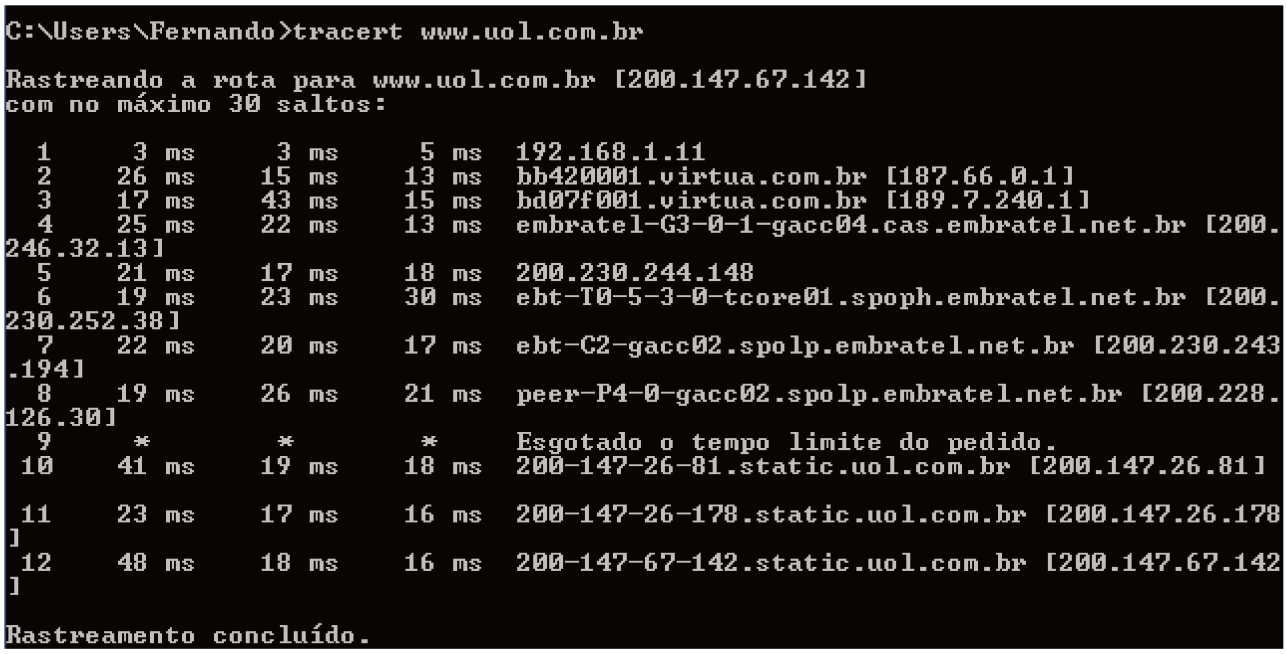


Figura 27: *Trace Router* do site www.uol.com.br.

O tempo total de propagação para o site da UOL foi de 264 ms. Um tempo normal utilizando o protocolo TCP/IP. Já o site russo, www.b1.rs, levou 1805 ms, sendo preciso passar por 13 saltos, utilizando TCP/IP. Um tempo como esse para tráfegos do tipo elástico, como a aplicação HTTP é aceitável. Mas se isso ocorrer a um tráfego *stream*, podemos ter problemas de degradação do serviço ou a impossibilidade de estabelecimento de conexão por falta de condições mínimas exigidas pelo serviço.

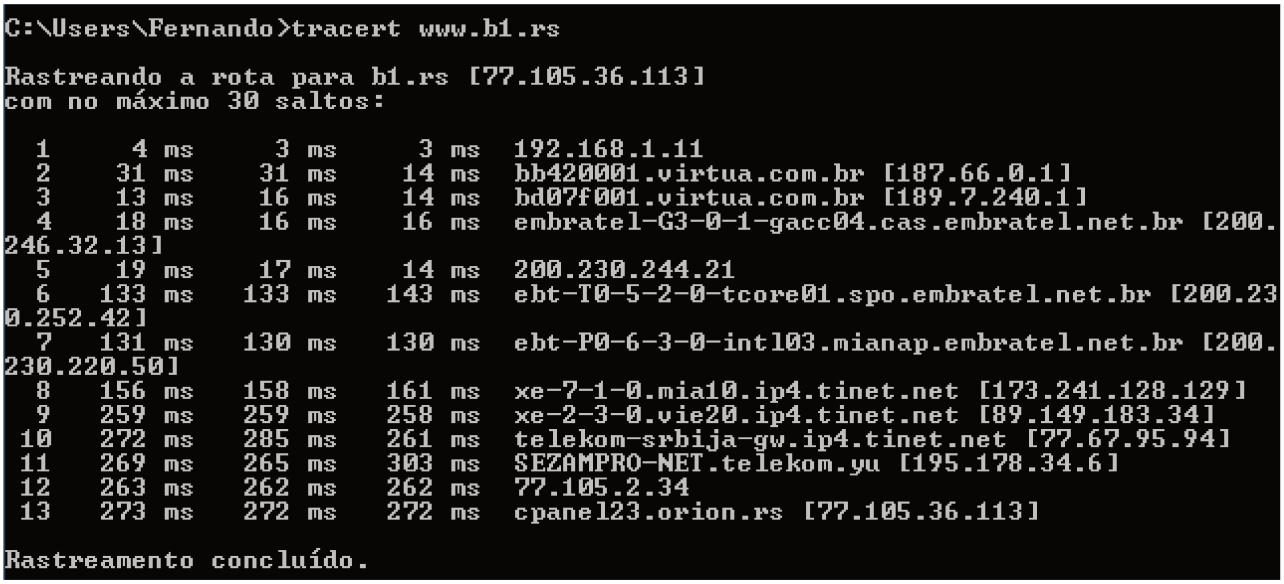


Figura 28: *Trace Router* do site www.b1.rs.

5.3.1 Resultados

Para implementar o modelo de simulação a fim de simular um ambiente VPN utilizamos três atributos novos nos seis serviços já conhecidos. Eles são, *Jitter*, Perda, e Atraso, sendo o Atraso o atributo responsável a simular a latência. Como o software ARENA utilizado é a versão estudante, neste modelo de simulação tivemos a inconveniência do limite de entidade, não permitindo simulações acima de 70 Erlang.

Conforme ilustrado no exemplo do "*Trace Router*" o *jitter* mais a latência resultam no tempo total de atraso do pacote. Dessa forma, conseguimos simulá-los adicionando um tempo extra que levaria o pacote para chegar ao seu destino. O atributo perda representa uma porcentagem de perda que pode ocorrer a cada pacote transmitido no VPN.

Os valores do *jitter*, latência e perda foram sugeridos a partir de diversas referencias [32, 33, 34 35]. As Tabs. 15, 16 e 17 a seguir se encontram os valores de cada atributo por serviço. Propusémos três diferentes valores para o serviço de Dados mantendo os valores dos outros serviços.

	Videoconferência	Vídeo *On Demand*	Vídeo *Clip*	VoIP	Câmera IP	Data file (Dados)
Jitter	7×10^{-3}	7×10^{-3}	7×10^{-3}	100×10^{-3}	7×10^{-3}	8×10^{-3}
Atraso	0,03	0,1	0,1	400×10^{-3}	0,03	60×10^{-3}
Perda	10^{-5}	10^{-5}	10^{-5}	0,05	10^{-5}	10^{-9}

Tabela 15: Valores dos atributos *Jitter* e Atraso em segundos, e Perda em porcentagem, dos seis serviços do modelo.

	Videoconferência	Vídeo *On Demand*	Vídeo *Clip*	VoIP	Câmera IP	Data file (Dados)
Jitter	7×10^{-3}	7×10^{-3}	7×10^{-3}	100×10^{-3}	7×10^{-3}	8×10^{-3}
Atraso	0,03	0,1	0,1	400×10^{-3}	0,03	60×10^{-3}
Perda	10^{-5}	10^{-5}	10^{-5}	0,05	10^{-5}	0,01

Tabela 16: Valores dos atributos *Jitter* e Atraso em segundos, e Perda em porcentagem, dos seis serviços do modelo.

	Videoconferência	Vídeo *On Demand*	Vídeo *Clip*	VoIP	Câmera IP	Data file (Dados)
Jitter	7×10^{-3}	7×10^{-3}	7×10^{-3}	100×10^{-3}	7×10^{-3}	16×10^{-3}
Atraso	0,03	0,1	0,1	400×10^{-3}	0,03	60×10^{-3}
Perda	10^{-5}	10^{-5}	10^{-5}	0,05	10^{-5}	10^{-9}

Tabela 17: Valores dos atributos *Jitter* e Atraso em segundos, e Perda em porcentagem, dos seis serviços do modelo.

Além da mudança dos atributos, realizaremos, como no modelo 2, simulações para os casos A,B e C, para as funções de distribuição exponencial, hiper-exponencial e pareto. Na representação a seguir entenderemos melhor todos os casos propostos.

- Função de Distribuição Exponencial
  - Utilizaremos os valores dos atributos da Tab. 15 para realizar os casos:
    - Caso A, com prioridade, sem CAC
    - Caso B, sem prioridade, sem CAC
    - Caso C, sem prioridade, com CAC
    - Caso C, com prioridade, com CAC
  - Utilizaremos os valores dos atributos da Tab. 16 para realizar os casos:
    - Caso A, com prioridade, sem CAC
    - Caso B, sem prioridade, sem CAC
    - Caso C, sem prioridade, com CAC
    - Caso C, com prioridade, com CAC
  - Utilizaremos os valores dos atributos da Tab. 17 para realizar os casos:
    - Caso A, com prioridade, sem CAC
    - Caso B, sem prioridade, sem CAC
    - Caso C, sem prioridade, com CAC
    - Caso C, com prioridade, com CAC

- Função de Distribuição Hiper-exponencial
  - Utilizaremos os valores dos atributos da Tab. 15 para realizar os casos:
    - Caso A, com prioridade, sem CAC
    - Caso B, sem prioridade, sem CAC
    - Caso C, sem prioridade, com CAC
    - Caso C, com prioridade, com CAC
  - Utilizaremos os valores dos atributos da Tab. 16 para realizar os casos:
    - Caso A, com prioridade, sem CAC
    - Caso B, sem prioridade, sem CAC
    - Caso C, sem prioridade, com CAC
    - Caso C, com prioridade, com CAC
  - Utilizaremos os valores dos atributos da Tab. 17 para realizar os casos:
    - Caso A, com prioridade, sem CAC
    - Caso B, sem prioridade, sem CAC
    - Caso C, sem prioridade, com CAC
    - Caso C, com prioridade, com CAC

- Função de Distribuição Pareto
  - Utilizaremos os valores dos atributos da Tab. 15 para realizar os casos:
    - Caso A, com prioridade, sem CAC
    - Caso B, sem prioridade, sem CAC
    - Caso C, sem prioridade, com CAC
    - Caso C, com prioridade, com CAC

- Utilizaremos os valores dos atributos da Tab. 16 para realizar os casos:
  - Caso A, com prioridade, sem CAC
  - Caso B, sem prioridade, sem CAC
  - Caso C, sem prioridade, com CAC
  - Caso C, com prioridade, com CAC
- Utilizaremos os valores dos atributos da Tab. 17 para realizar os casos:
  - Caso A, com prioridade, sem CAC
  - Caso B, sem prioridade, sem CAC
  - Caso C, sem prioridade, com CAC
  - Caso C, com prioridade, com CAC

- Função de Distribuição Pareto Truncada
  - Utilizaremos os valores dos atributos da Tab. 15 para realizar os casos:
    - Caso A, com prioridade, sem CAC
    - Caso B, sem prioridade, sem CAC
    - Caso C, sem prioridade, com CAC
    - Caso C, com prioridade, com CAC
  - Utilizaremos os valores dos atributos da Tab. 16 para realizar os casos:
    - Caso A, com prioridade, sem CAC
    - Caso B, sem prioridade, sem CAC
    - Caso C, sem prioridade, com CAC
    - Caso C, com prioridade, com CAC
  - Utilizaremos os valores dos atributos da Tab. 17 para realizar os casos:
    - Caso A, com prioridade, sem CAC
    - Caso B, sem prioridade, sem CAC
    - Caso C, sem prioridade, com CAC
    - Caso C, com prioridade, com CAC

A função de distribuição de Pareto truncada contêm um fator de correção, que segundo a referência [39] dependendo do número de dígitos do processador da máquina utilizado nas simulações deve-se utilizar um determinado fator de correção. A geração da distribuição Pareto está correta, mas pelas características de cauda longa, ela acaba sendo influenciada pelos dígitos de precisão da máquina e acaba na forma de distribuição truncada. De [39], o valor do fator multiplicativo para 64 bits (máquina utilizada para as simulações) resultou em torno de 15150.

$$E[x] = \frac{\alpha}{1-\alpha}.FATOR \quad \text{e} \quad FATOR = x_{min}\left[\frac{1-w^{\frac{\alpha}{1-\alpha}}}{1-w}\right] = 15150 \tag{22}$$

Nos casos A e B temos um volume de tráfego no serviço VoIP (40, 55 e 70 Erlang), e para o caso C fixo em 70 Erlang. Foram simuladas 5 replicações por simulação e um período de 16000 segundos por replicação. Com 5 replicações conseguimos uma precisão do resultados em torno de 7% para tráfegos elásticos e 3% para tráfegos *stream*.

- **Função de Distribuição Exponencial**
    - Resultados obtidos utilizando os valores da Tab. 15.

	40	55	70
Caso A	5,37572	5,3992	5,44496
Caso B	5,42214	5,39442	5,8207

Tabela 18: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.

	70
Caso C P1	5,46388
Caso C P2	5,45328

Tabela 19: Tempo *Sojourn* do serviço de Dados, P1 = sem prioridade, P2 = com prioridade.

    - Resultados obtidos utilizando os valores da Tab. 16.

	40	55	70
Caso A	5,47774	6,00104	5,79242
Caso B	5,85068	5,41588	5,63746

Tabela 20: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.

	70
Caso C P1	5,3832
Caso C P2	5,84266

Tabela 21: Tempo *Sojourn* do serviço de Dados, P1 = sem prioridade, P2 = com prioridade.

    - Resultados obtidos utilizando os valores da Tab. 17.

	40	55	70
Caso A	5,45896	5,5932	5,26142
Caso B	5,96788	5,41106	5,61714

Tabela 22: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.

	70
Caso C P1	5,5377
Caso C P2	5,38454

Tabela 23: Tempo *Sojourn* do serviço de Dados, P1 = sem prioridade, P2 = com prioridade.

- **Função de Distribuição Hiper-exponencial**
    - Resultados obtidos utilizando os valores da Tab. 15.

	40	55	70
Caso A	10,29024	10,249	10,10156
Caso B	11,0954	10,55894	10,03174

Tabela 24: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.

	70
Caso C P1	10,86248
Caso C P2	9,396886

Tabela 25: Tempo *Sojourn* do serviço de Dados, P1 = sem prioridade, P2 = com prioridade.

- Resultados obtidos utilizando os valores da Tab. 16.

	40	55	70
Caso A	10,4243	10,59846	10,46824
Caso B	9,9131	10,39784	10,32982

Tabela 26: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.

	70
Caso C P1	10,60298
Caso C P2	10,51146

Tabela 27: Tempo *Sojourn* do serviço de Dados, P1 = sem prioridade, P2 = com prioridade.

- Resultados obtidos utilizando os valores da Tab. 17.

	40	55	70
Caso A	10,13156	10,49194	10,10788
Caso B	10,7448	9,891	10,57862

Tabela 28: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.

	70
Caso C P1	10,0855
Caso C P2	10,18852

Tabela 29: Tempo *Sojourn* do serviço de Dados, P1 = sem prioridade, P2 = com prioridade.

- **Função de Distribuição Pareto**

- Resultados obtidos utilizando os valores da Tab. 15.

	40	55	70
Caso A	0,001782	0,001782	0,00178
Caso B	0,001778	0,00178	0,001776

Tabela 30: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.

	70
Caso C P1	0,001764
Caso C P2	0,001764

Tabela 31: Tempo *Sojourn* do serviço de Dados, P1 = sem prioridade, P2 = com prioridade.

- Resultados obtidos utilizando os valores da Tab. 16.

	40	55	70
Caso A	0,002276	0,00252	0,002142
Caso B	0,002272	0,002516	0,002056

Tabela 32: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.

	70
Caso C P1	0,002598
Caso C P2	0,002598

Tabela 33: Tempo *Sojourn* do serviço de Dados, P1 = sem prioridade, P2 = com prioridade.

- Resultados obtidos utilizando os valores da Tab. 17.

	40	55	70
Caso A	0,001782	0,001782	0,00178
Caso B	0,001778	0,00178	0,001776

Tabela 34: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.

	70
Caso C P1	0,001764
Caso C P2	0,001764

Tabela 35: Tempo *Sojourn* do serviço de Dados, P1 = sem prioridade, P2 = com prioridade.

- **Função de Distribuição Pareto Truncada**

  - Resultados obtidos utilizando os valores da Tab. 15.

	40	55	70
Caso A	0,81436	1,05592	1,44327
Caso B	0,879763	1,068883	0,896023

Tabela 36: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.

	70
C P1	1,186677
C P2	0,843783

Tabela 37: Tempo *Sojourn* do serviço de Dados, P1 = sem prioridade, P2 = com prioridade.

  - Resultados obtidos utilizando os valores da Tab. 16.

	40	55	70
Caso A	0,801303	1,89029	2,458423
Caso B	1,732033	2,251197	1,178111

Tabela 38: Tempo *Sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.

	70
C P1	1,11076
C P2	1,16293

Tabela 39: Tempo *Sojourn* do serviço de Dados, P1 = sem prioridade, P2 = com prioridade.

- Resultados obtidos utilizando os valores da Tab. 17.

	40	55	70
Caso A	0,735877	0,788977	0,87084
Caso B	0,776757	0,923763	0,745057

Tabela 40: Tempo *sojourn* do serviço de Dados, caso A com prioridade, caso B sem prioridade.

	70
C P1	0,653647
C P2	0,607877

Tabela 41: Tempo *Sojourn* do serviço de Dados, P1 = sem prioridade, P2 = com prioridade.

5.3.2 Comparação dos Resultados

Faremos duas comparações, entre os resultados dos casos de uma mesma distribuição, e em seguida entre as distribuições.

- **Distribuição Exponencial**

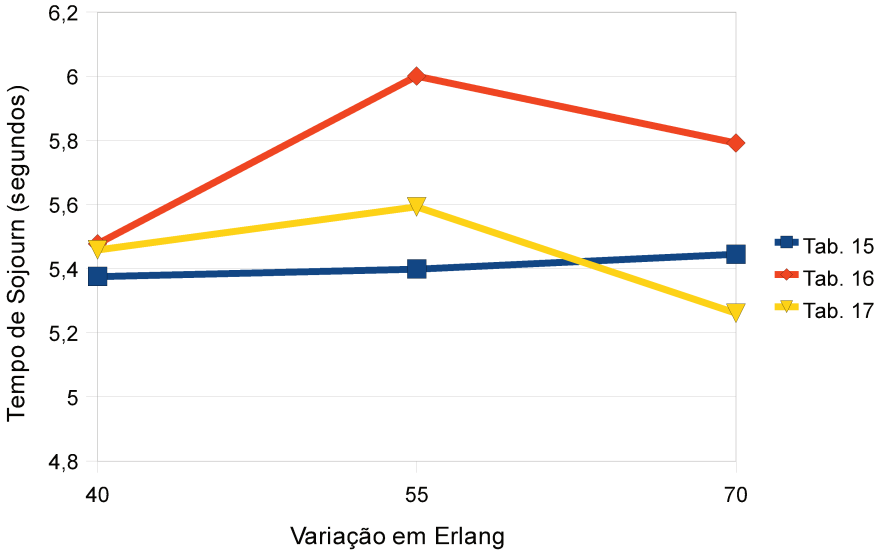


Figura 29: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos A entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição exponencial.

No caso exponencial o comportamento dos resultados com prioridade (caso A), Fig. 29, e sem prioridade (caso B), Fig. 30, tiveram uma mudança brusca de comportamento, principalmente, nas linhas amarela e vermelha, mas elas estão seguindo uma mesma tendencia de comportamento. A média do *sojourn* do caso A e B permaneceram próximas mesmo mudando suas prioridades.

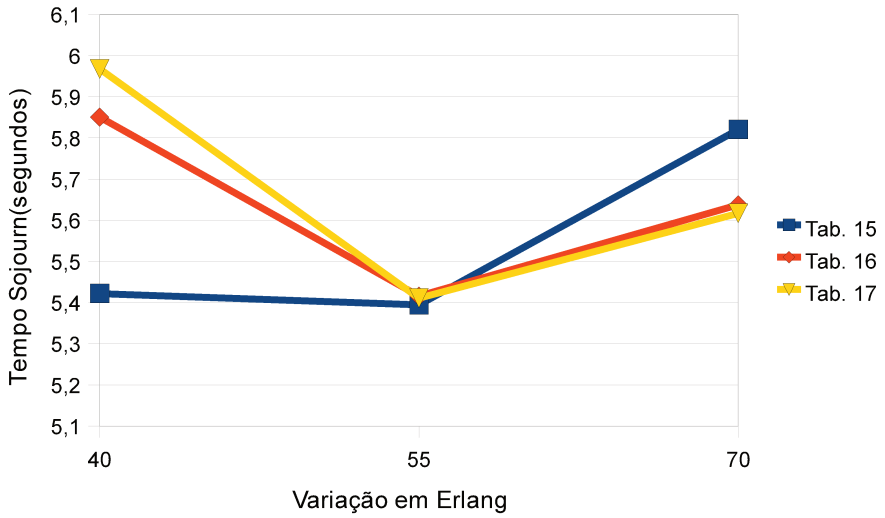


Figura 30: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos B entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição exponencial.

- **Distribuição Hiper-Exponencial**

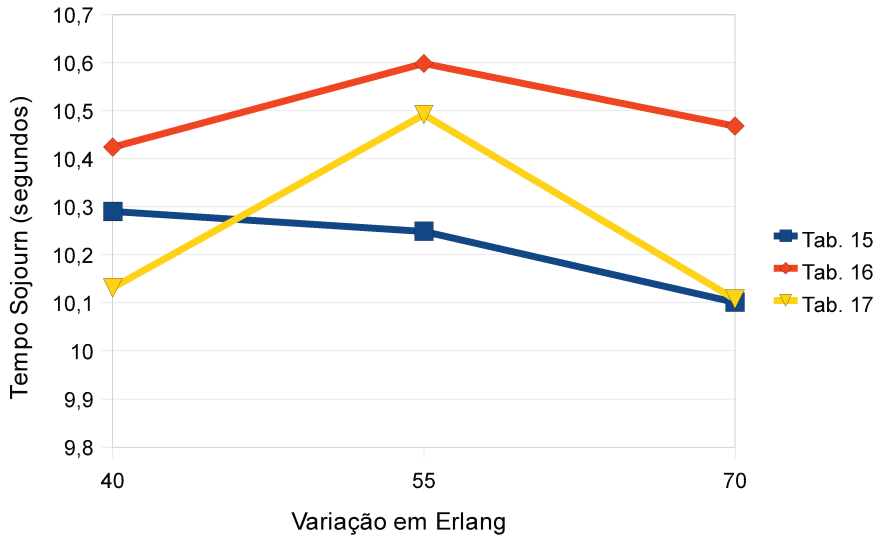


Figura 31: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos A entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição hiper-exponencial.

Com a adição dos atributos *jitter* e latência tivemos resultados com comportamentos diferentes dos vistos no modelo de simulação 2. Como mostra a Fig. 31 e 32, nem sempre ao aumentarmos o volume do tráfego VoIP (40, 55 e 70 Erlang) temos um aumento do tempo *sojourn* do serviço de Dados.

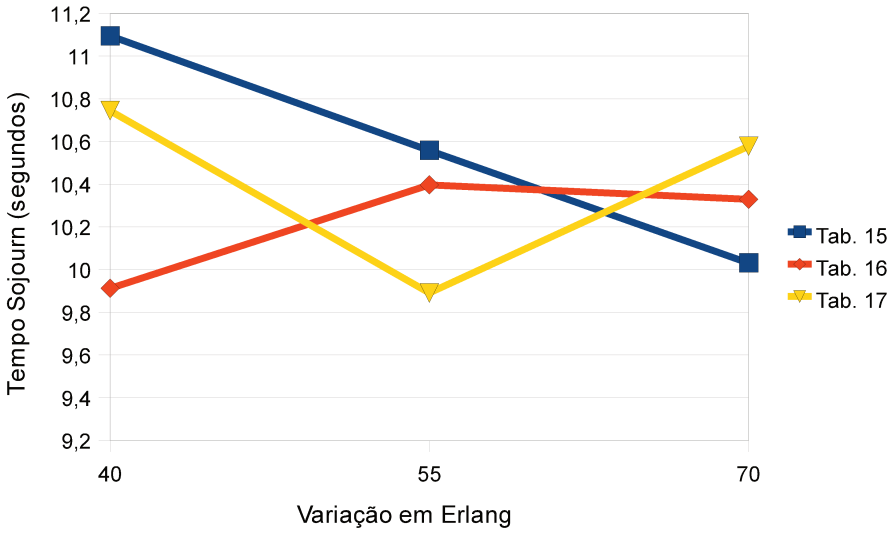


Figura 32: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos B entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição hiper-exponencial.

- **Distribuição Pareto**

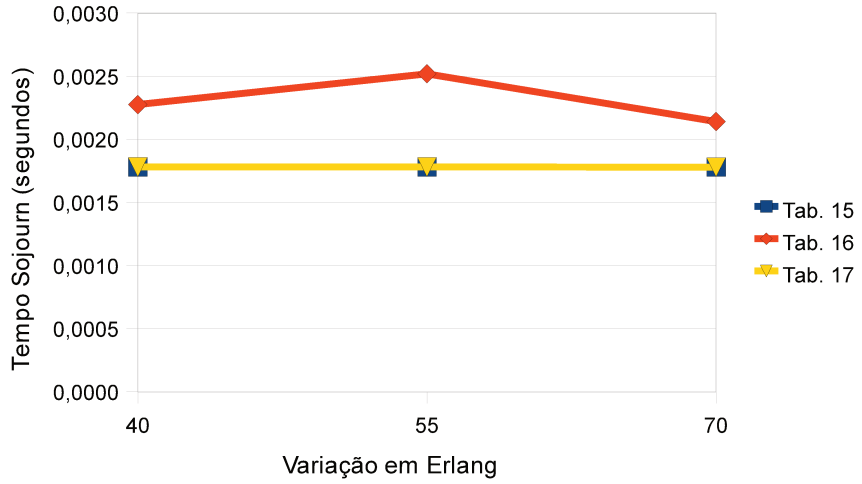


Figura 33: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos A entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição Pareto.

A distribuição de Pareto foi a que mais se aproximou de um padrão de comportamento ao simularmos caso A e B. Podemos nota-lo nas Fig. 33 e 34.

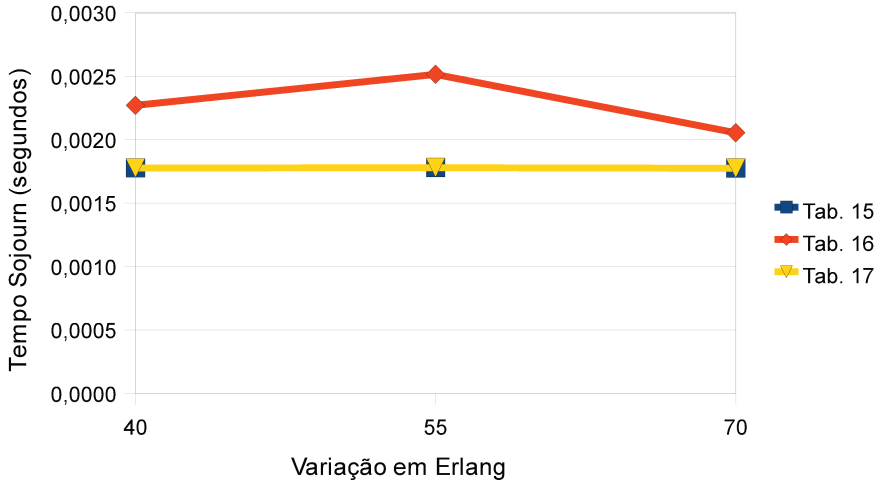


Figura 34: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos B entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição Pareto.

- **Distribuição Pareto Truncada**

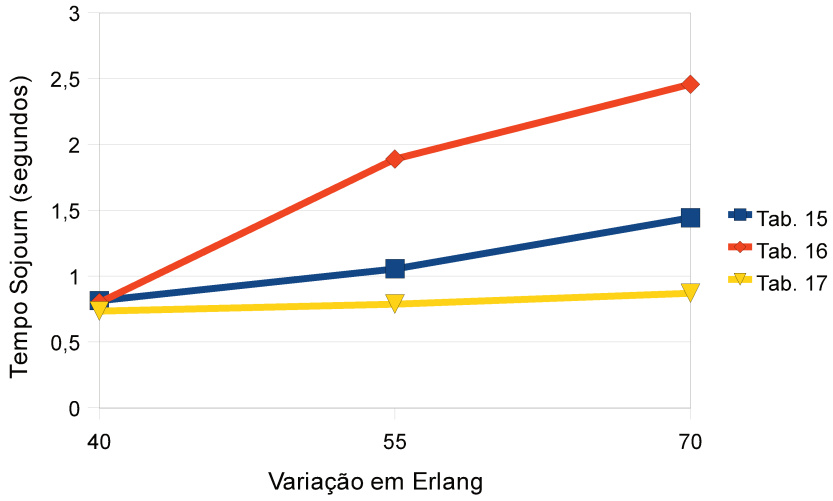


Figura 35: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos A entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição Pareto Truncada.

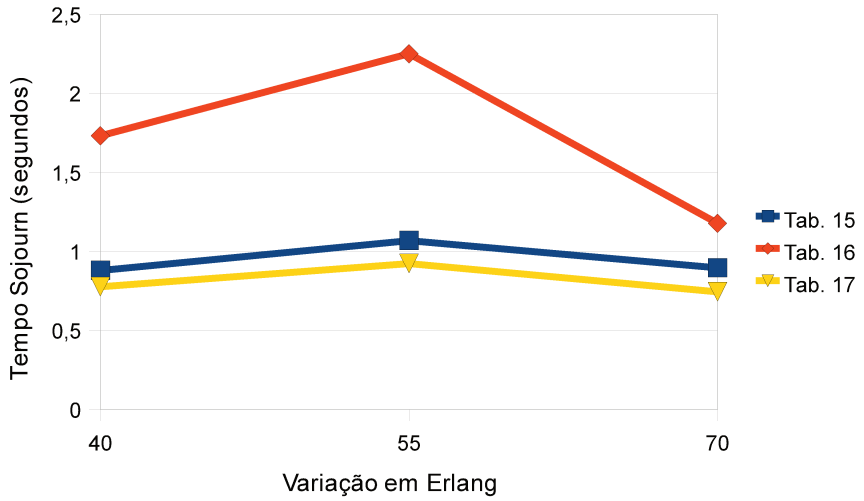


Figura 36: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos B entre os diferentes valores de *jitter*, latência e perda (Tab. 15, 16 e 17) do serviço de Dados utilizando distribuição Pareto Truncada.

- **Comparação entre funções de distribuição**

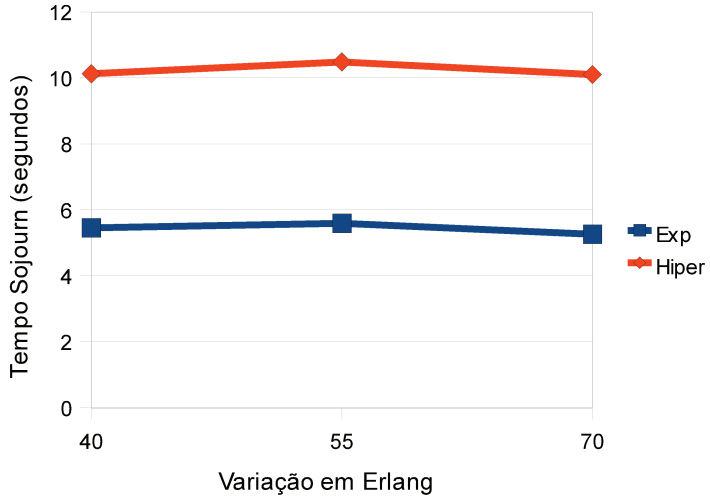


Figura 37: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos A entre a distribuição exponencial e hiper-exponencial. Valores de *jitter*, latência e perda da Tab. 17.

Veja que mudam os valores do tempo de *sojourn* dos serviços de Dados, na Fig. 35 e 36, ao cambiarmos as funções de distribuições, mas não mudam as tendencias de comportamento das linhas. Na Fig. 35, nas duas distribuições ao termos 55 Erlang ocorre um pequeno aumento no tempo *sojourn*. O efeito contrario ocorre na Fig. 36, diminuindo o tempo de *sojourn*.

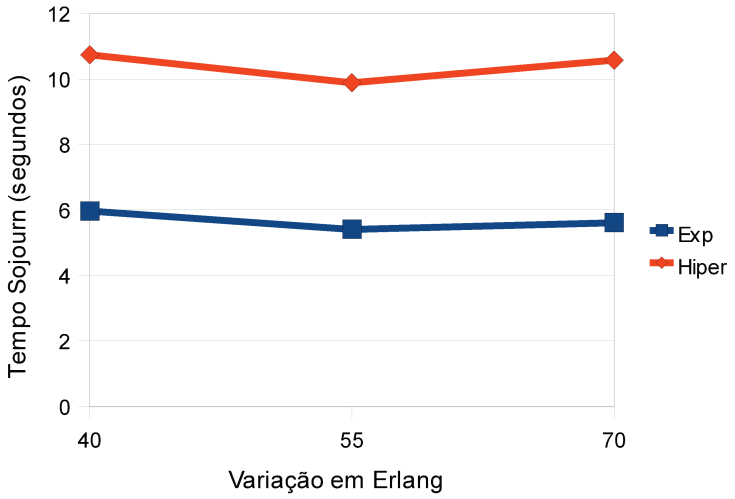


Figura 38: Tempo de *sojourn* do serviço de Dados, em segundos, em função a variação do volume do tráfego em Erlang. Comparação dos casos B entre a distribuição exponencial e hiper-exponencial. Valores de *jitter*, latência e perda da Tab. 17.

6. Conclusões

Nesse trabalho foi discutido um conjunto de modelo analítico, simulações e emulações equacionando a questão do dimensionamento do enlace. Quando se trata de um enlace onde se tem o controle de suas condições de tráfego (ou se pode fazer medidas), o modelo analítico e simulações satisfazem às necessidades. Quando é necessário estimar outros parâmetros, como em um VPN, o gerador de tráfego/emulador tem um papel fundamental porque possui a influencia do canal resultando em resultados próximos de um ambiente real.

Muitos desafios surgiram durante o desenvolvimento das atividades desta dissertação, como o gerador de tráfegos multimídia e os modelos de simulação. As emulações feitas utilizando *socket* mostrou-se fieis aos resultados obtidos utilizando as mesmas configurações no ARENA. A quantidade de serviços gerados no emulador são quase sempre menores ao do ARENA pelo fato do arena ser simulado localmente, não transmitindo os pacotes através de um meio físico.

Para um aprimoramento no gerador foi sugerido a implementação da pilha TCP/IP uIP para termos controle do *header* IP. No *header* IP existe campos de identificação do serviço que o pacote pertence, e sem esta manipulação não conseguimos fazer um tratamento de identificação na camada de rede, tendo que ser feito na camada de transporte utilizando a identificação de cada serviços em suas respectivas portas. Essa implementação é fundamental para aumentar a agilidade do tráfego dos pacotes nos roteadores que tratam o tráfego com priorização, melhorando o QoS da rede evitando filas. Foram feitas implementações na pilha TCP/IP uIP com o objetivo de aprimorar a pilha e adapta-la para a integração com o emulador de tráfegos multimídia. Infelizmente após as implementações feitas no uIP estarem completas, descobrimos a incapacidade de utilização do uIP, por não possuir um *driver.* Sem um *driver* a pilha não consegue se comunicar com a interface da placa de rede, impossibilitando que os pacotes sejam transmitidos através do meio físico. A implementação de um *driver* para a pilha foi descartado pelo tempo que levaria para o seu desenvolvimento e testes de confiabilidade. Por outro lado, a utilização da interface TUN/TAP mostrou-se insegura, mesmo o uIP possuindo uma função para o seu uso, o TUN/TAP não transmitiu nenhum bit das aplicações de exemplo que o uIP possui.

Foram propostos três diferentes modelos de simulação que simulam ambientes multimídia, onde serviços *stream* e elástico são requisitados ao mesmo tempo pelos usuários. A função de um modelo é descrever o funcionamento da realidade através de um pequeno número de variáveis que permita a sua conversão. O modelo 1 possui três blocos *Arrive*, que são os responsáveis pela geração dos serviços *stream*, videoconferência, vídeo-c*lip*, e vídeo o*n demand.* Cada bloco possui valores que representam seus serviços e funções de distribuição. Para cada serviço há duas saídas, serviços rejeitados e serviços completados. O modelo 2 e 3 possuem seis serviços, videoconferência, vídeo-c*lips*, vídeo o*n demand,* VoIP*,* câmera IP, e *data file* (serviço de Dados), sendo apenas o serviço de Dados do tipo elástico.

No modelo 1 foram realizados simulações propondo um estudo de demanda dos usuários, onde três cenários foram estudados. Esses tipos de situações são amplamente estudadas em casos de mudança do comportamento na tarifação das empresas de telefonias moveis e fixa, onde a cobrança não é mais feita por minutos utilizados, mas sim por chamadas feitas aumentando significativamente a duração das chamadas. Os estudos de demanda são cruciais por parte das empresas para não acarretar em um baixo QoS (infringindo as regras da ANATEL) e bloqueios de chamadas elevado. Dos resultados obtidos nas Figs. 22 e 23 podemos notar que mesmo havendo um aumento brusco no tempo médio da duração das chamadas ou, no intervalo médio entre as requisições é possível ter uma margem aceitável de QoS em ambientes em que ocorram picos proporcionando recursos um pouco acima da média.

No modelo 2 ficou claro a importância da priorização dos serviços, porque o perfil de um tráfego *stream* é diferente de um tráfego elástico. Em um ambiente com QoS a priorização do tráfego da rede é fundamental. Na simulação da Fig. 26, podemos notar com clareza a diferença do *sojourn* quando se utiliza priorização dos serviços, o tempo que o pacote elástico demorou para chegar ao seu destino, com um volume de tráfego de 85 Erlang, foi 2,54 vezes maior quando a prioridade do serviço de Dados é inferior aos demais, aumentando o tempo de entrega dos pacotes elástico.

No modelo 3 os parâmetros do *jitter* e da latência mudaram significativamente o comportamento do *sojourn* visto no modelo 2. Em todos os cenários simulados no modelo 3 houve um aumento significativo no tempo de *sojourn*. Além disso, mesmo aumentando o volume do tráfego *stream*, o tempo de *sojourn* do tráfego elástico em muitos casos não aumentou. Isso se deve pela influencia do *jitter* e da latência em ambientes VPN. Nos exemplos utilizando o *Trace Router* foi possível notar a influência do *jitter* e da latência em um ambiente real, em que a variação do tempo que leva para o pacote chegar ao seu destino passando pelos mesmos meios físicos nem sempre são as mesmas. Como as simulações e o *Trace Router* mostraram, estamos sempre vulneráveis aos problemas de *jitter* e de latência, resultando em tempos de *sojourn* inesperados.

Em trabalhos futuros o estudo da substituição do socket da pilha TCP/IP é fundamental. A pilha uIP estudada nesse trabalho mostrou que apesar de ser uma pilha compacta e possuir o perfil que desejamos, como ser código aberto em C e desenvolvida para Linux, ela possui alguns problemas que precisam ser contornados para que seja possível utiliza-la, como o *driver* e a agilidade do *buffer*. Simulações utilizando a versão completa do software ARENA proporcionará futuros estudos com volumes de tráfegos diferentes e com várias replicações dos modelos propostos. Muitos casos importantes não foi possível simular devido as opções restritas que a versão de estudante do ARENA proporciona.

Referências

[1] Gartner Inc. Carina Forsling. *Worldwide Consumer Broadband Penetration Sees Rapid Growth but Current Price Strategy Alone is Not Sustainable for Telecom Carriers Says Gartner.* http://www.gartner.com/it/page.jsp?id=501276

[2] Internetional Television Expert Group. ITEG.org. http://www.international-television.org/tv_market_data/global-iptv-forecast-2009-2013.html

[3] Mediaroom, Projeto envolvendo diversas empresas, como Microsoft, AT&T, BT Vision, DT e muito outras. http://www.microsoft.com/Mediaroom/default.aspx

[4] Nokia Siemens Network. Soluções para IPTV. http://www.nokiasiemensnetworks.com/portfolio/solutions/iptv-is-your-opportunity-to-enhance-revenue

[5] A . Golaup e H. Aghvami, *A multimedia traffic modeling framework for simulation based performance evaluetion studies*, Computer Networks 50 2071 (2006).

[6] W.E. Leland, M.S. Taqqu, W. Willinger, D.V. Wilson, *On the self-similar nature of Ethernet traffic (extended version)*, IEEE/ACM Trans. Networking 2 (1) (1994) 1–14

[7] A. Golaup, A.H. Aghvami, *Modeling of MPEG4 traffic at GoP level using autoregressive processes*, IEEE Veh. Technol. Conference Fall 2002, Vancouver, British Columbia, Canada, 2002, pp. 854–858.

[8] F.H.P. Fitzek, M. Reisslein, *MPEG-4 and H.263 video traces for network performance evaluation*, Report no. TKN-00-06, Technical University Berlin, Berlin, October 2000.

[9] A . Lazaris, P. Koutsakis e M. Paterakis, *A new model for video traffic originating from multiplexed MPEG-4 videoconference streams*, Performance Evaluation 65 51 (2008).

[10] M.W. Garrett, W. Willinger, Analysis, M*odeling and generation of self-similar VBR video traffic*, in: Proceedings of ACM SIGCOMM94, pp. 269–280.

[11] S. Ma, C. Ji., *Modeling video traffic in the wavelet domain*, in: Proceedings of INFOCOM98, San Francisco, CA, vol. 1, 1998, pp. 201–208

[12] S. A . AlQahtani, *A Simulation-Based Comparison of Multimedia Traffic Prioritization Schemes for Higjt-Performance Input-Queued Packet Switches*, J. Computer Sci. 2 (4) (2006)

[13] P.E wirth, *The role od teletraffic modeling in the new communication paradigms*, *IEEE comum*. Mag. (1997) 86-92.

[14] V.S. Frost, B. Melamed, *Traffic modeling for telecommunications network, IEEE comum*. Mag.

(1994) 70-81.

[15] Ziv A, Wolpe PR, Small SD, *Glick S. Simulation-based medical education: an ethical imperative*. Acad Med 2003; 78(8) :783- 8.

[16] MANOVICH, Lev, Image Future, Outubro de 2003, editado em Abril de 2004, revisto em 2006

[17] A. Golaup, A.H. Aghvami, Modeling of MPEG4 traffic at GoP level using autoregressive processes, IEEE Veh. Technol. Conference Fall 2002, Vancouver, British Columbia, Canada, 2002, pp. 854–858.

[18] Arena FAQ, http://www.arenasimulation.com/Tools_Resources_Arena_FAQs.aspx#

[19] W. David Kelton, Randall P. Sadowski, David T. Sturrock, W. Kelton, Randall Sadowski, David Sturrock. *Simulation with Arena.* 3rd edition, 2003.

[20] Leandro Agugliari, Tito Ricardo Bianchin Oliveira. Desenvolvimento de um gerador de tráfego multimídia. TGI apresentado à Faculdade de Tecnologia – FT – UNICAMP.

[21] CARVER, Richard H.; TAI, Kuo-Chung. *Modern Multithreading: Implementing, Testing and Debugging Multithreaded Java and C++/Pthreads/Win32 Programs*. Editora Wiley. Outubro 2005.

[22] Adam Dunkels. The uIP Embedded TCP/IP Stack. The uIP 1.0 Reference Manual. Uip-refman.

[23] Li Frank Yong, Stol Norvald. *A priority-oriented call admission control paradigm with QoS re-negotiation for multimedia services in UMTS*. IEEE 53rd Vehicular Technology Conference, Greece, 2001, vol.3, 2021–2025.

[24] Fonte bibliográfica da Figura/Tabela : Andrew Stuart Tanenbaum, *Computer Networks.*

[25] Qualidade de Serviço (QoS) em Redes IP Princípios Básicos, Parâmetros e Mecanismos, Prof. Hugo Santana, Universidade Santa Cecília – Unisanta.

[26] S. M. C. TOME, E. L. URSINI, N. MINCOV, Dimensionamento de Rede IP Integrada Corporativa/Operativa de Subestações de Energia, I Congresso Tecnológico Infobrasil, Fortaleza, CE, 2008.

[27] R.L. Sharma, *Network Design Using EcoNets*, International Thomson Computer Press, Boston, 1997.

[28] M.B. Trindade, M.S. Medrano, A.C. Lavelha, Planejamento de Enlaces IP Multi-serviço, considerando Requisitos de QoS, Congresso Nacional de Tecnologia da Informação e Comunicação – SUCESU 2003, Salvador, 2003

[29] A. Riedl, T. Bauschert and A. Probst, *Dimensioning of IP access networks with elastic traffic*, NETWORKS 2000, Proceedings, Toronto, 2000.

[30] J.F.P. Labourdette e G.W. Hart, *Blocking Probabilities in Multitraffic Loss Systems: Insensitivity, Asymptotic Behavior and Approximations, IEEE Transactions on Communications, v. 40, n. 8, ago. 1992*

[31] http://en.wikipedia.org/wiki/Pareto_distribution

[32] Acatel, Futurecom de 2001.

[33] Kurose: Rede de Computadores e a Internet, James F. Kurose e Keith W. Ross, terceira edição.

[34] Jorge Moreira de Souza, QoS para VoIP I.

[35] Prof. Dr. Valter Roesler, UFRGS, Análises e Medida de Desempenho em Redes Locais.

[36] http://vtun.sourceforge.net/tun/

[37] Fernando L. Pinotti, Tito Oliveira, Varese Timóteo, Edson Ursini, An IP-based multimedia traffic generator, Proceeding IWT 2011, ISSN 1806-7662.

[38] Dalila Fonseca, Deolinda Pacheco, Fernando Marques, Ricardo Soares, Porto Editora, Aplicações informáticas A, 2006, Porto.

[39] *Accelerated Simulation of Power-Law Traffic in Packet Networks*, Ho I Ma, *Department of Electronic Engineering Queen Mary University of London, September 2003.*

APÊNDICE A- ARENA, PROGRAMA DE SIMULAÇÃO

O ARENA 12.0 (versão educacional) utilizado neste trabalho é um software de simulação por eventos discretos. Ele é composto de uma interface gráfica onde os principais componentes para gerar as simulações, são os módulos configura dos de acordo com a funcionalidade dos mesmos.

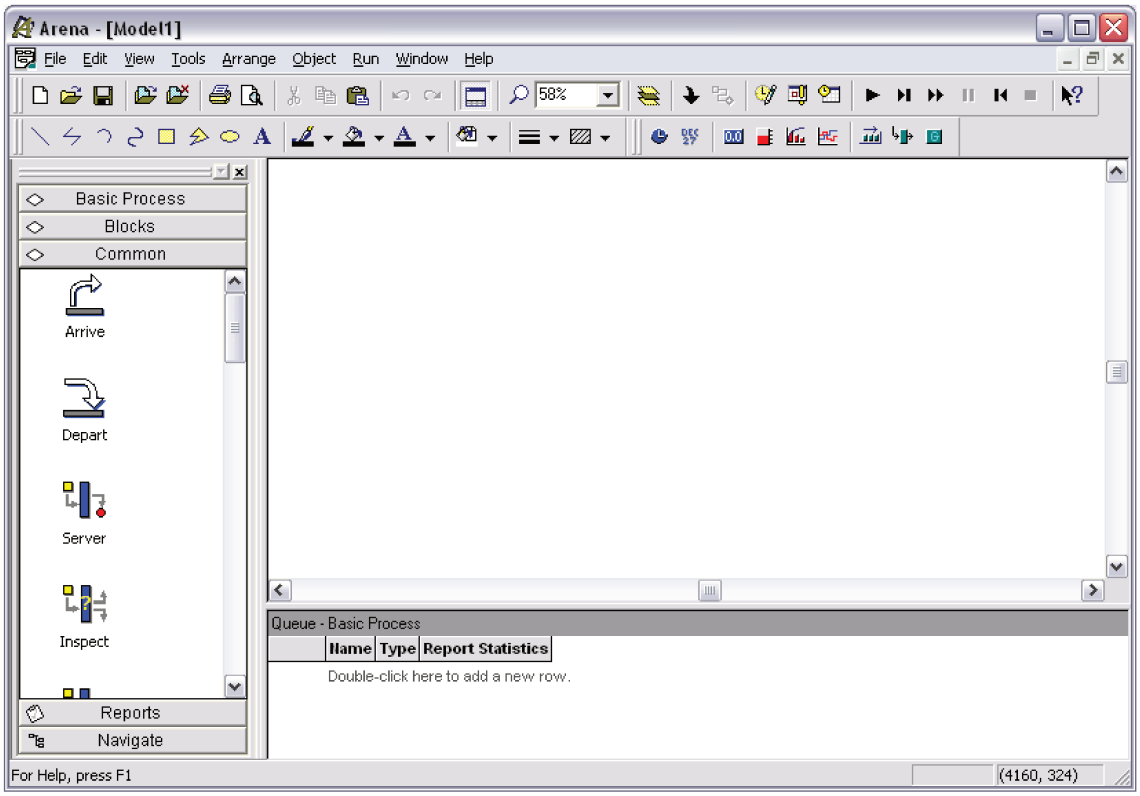


Figura 39: Tela principal do ARENA.

Ao final de cada simulação o ARENA gera um relatório da simulação, observando o comportamento das variáveis definidas em um arquivo de extensão.out.

O ARENA 12.0 dispõem de módulos simples, que realizam apenas uma função, e módulos compostos, que são um conjunto de funções em um mesmo módulo, afim de simplificar as simulações que requerem um número elevado de entidades. Em nossos modelos foram usados os dois tipos de módulos.

DESCRIÇÃO DOS BLOCOS UTILIZADOS

As informações aqui presentes foram obtidas através do *Help* do ARENA.

Bloco *ARRIVE*

O módulo *Arrive*, presente no painel *Common*, é usado para gerar as entidades que chegam a um modelo. A partir dele, são criadas as entidades, individualmente ou em lotes. No nosso modelo utilizamos seis blocos *Arrive*, cada um deles representando a geração de um serviço único. Eles são cinco serviço do tipo *stream*, Videoconferência, Vídeo Clip, Câmera IP, V*ideo on Demand*, *VoIP,* e um serviço do tipo elástico, *Data files* (Dados).

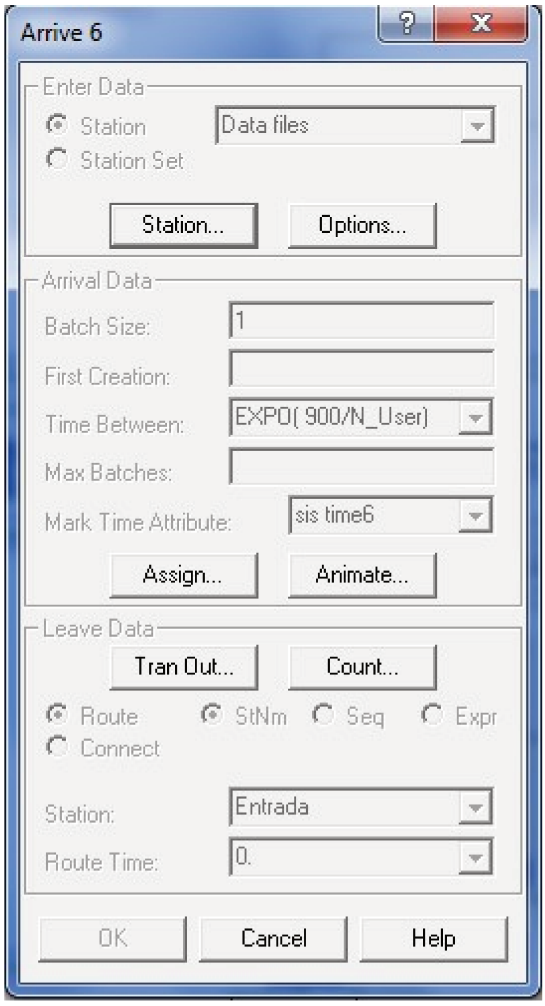


Figura 40: Parâmetros do bloco *ARRIVE.*

Enter Data

Station: Este campo define o nome da estação associada a este módulo.

Arrival Data

Batch Size: Este campo define o número de entidades em cada lote.

First Creation: Instante, a partir do qual, a primeira entidade pode ser criada.

Time Between: Este campo define o tempo entre cada criação de entidade (ou lote de entidades). Podem ser utilizadas funções de distribuição para definir este valor, como destaca a figura .

Mark Time attribute: Especifica o nome do atributo da entidade, utilizado para determinar o tempo de sistema.

Leave Data

Station: Este campo especifica a estação para a qual a entidade é transferida.

Route Time: Este campo define o tempo que uma entidade gasta para ir de uma estação a outra.

Bloco *ENTER*

Este módulo define uma estação (ou um conjunto de estações) correspondendo a um local físico ou lógico onde ocorrem operações. Se o módulo *Enter* definir um conjunto de estações, ele define os locais de múltiplos processamentos. Uma entidade pode se mover de um módulo anterior para um módulo *Enter* de três maneiras: transferida para a estação (ou conjunto de estações) associada com o módulo, através de uma conexão gráfica ou por um redirecionamento via um campo *Next Label*. Quando uma entidade chega a um módulo *Enter*, pode ocorrer um tempo de descarregamento e qualquer máquina usada para transportar a entidade é liberada.

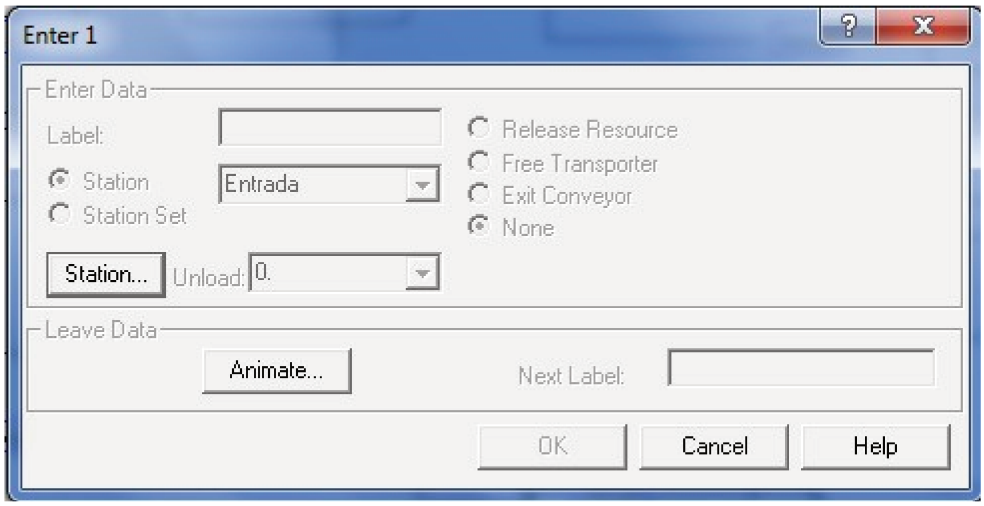


Figura 41: Parâmetros do bloco *ENTER.*

Label: Este campo define o nome do *label* que será associado com este módulo.

Station Type: Determina se uma única estação ou um conjunto de estações é usado para identificar o ponto de entrada para este módulo. Se o *Station Set* for o selecionado, significa que este módulo está definindo a entrada em um submodelo com várias estações.

Station: Define o nome da estação associada a este módulo.

Unload: Este campo define o tempo que as entidades levam para serem descarregadas dos transportadores.

Transfer Type: Indica o modo como as entidades liberam seus meios de transporte.

Bloco *CHOOSE*

O módulo *choose* fornece opções de escolha para as entidades baseadas no condicional *if*, juntamente com as regras *else* e *always*. Os destinos das opções são definidos por conectores ou por *labels*.

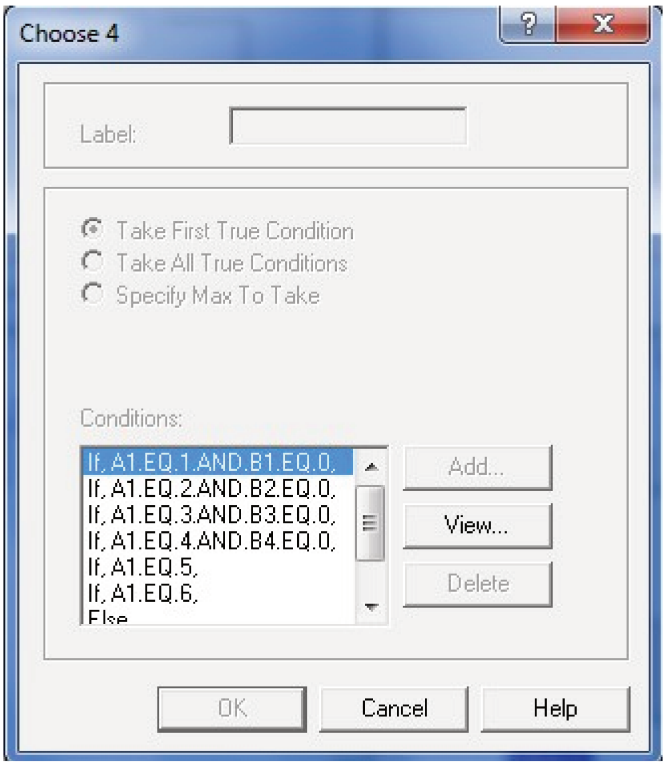


Figura 42: Parâmetros do bloco *CHOOSE.*

Condition

Selection Option: Estas opções determinam o número de entidades que sairá do módulo *choose*, baseadas nas opções que serão avaliadas como verdadeiras. Selecionando-se a *Take First True Condition*, a entidade sairá do módulo *Chance* baseado na primeira condição *if* avaliada como verdadeira. Nenhuma cópia da entidade original será criada. *Take all true conditions* fará com que a entidade saia da primeira condição *if* avaliada como verdadeira e cópias das entidades que chegam sejam criadas para cada uma das condições *always* e *if true* restantes. *Specify Max to take* permite que um número máximo e definido de opções sejam avaliadas pelas entidades.

Condition: Este grupo é usado para definir todas as possíveis opções que uma entidade pode escolher.

Para adicionar uma condição basta clicar em *ADD*, e configurar os parâmetros abaixo:

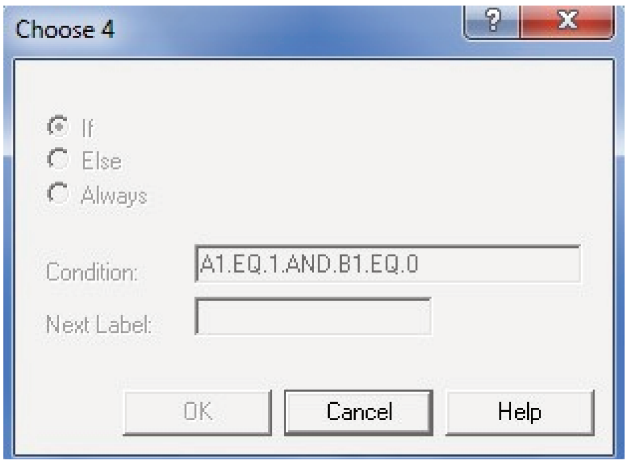


Figura 43: Configurando *Conditions*.

O tipo pode ser condicional (*if*) ou determinístico (*else/always*). O *if* requer a definição de uma expressão condicional. Somente uma opção *else* deve ser usada por módulo.

Condition: Este campo descreve a condição correspondente para cada condição *if*. Pode se usar sinais lógicos ou de comparação, como: *or, and*, <, >,etc.

Next Label: Este campo define o próximo módulo para onde a entidade se deslocará se a opção especificada for selecionada.

Bloco ASSIGN

Este módulo permite a associação de um valor a uma variável definida pelo usuário, fluxos contínuos ou níveis, atributo ou entidades, variável de status de modelo, ou um estado de recurso. Associações múltiplas podem ser feitas em um único módulo *Assign*. Quando uma entidade chega a um módulo *Assign*, o valor ou estado associado é avaliado e é associado a uma variável ou recurso específico. Se um atributo ou figura é especificado, o atributo ou figura da entidade que chega é associado ao novo valor.

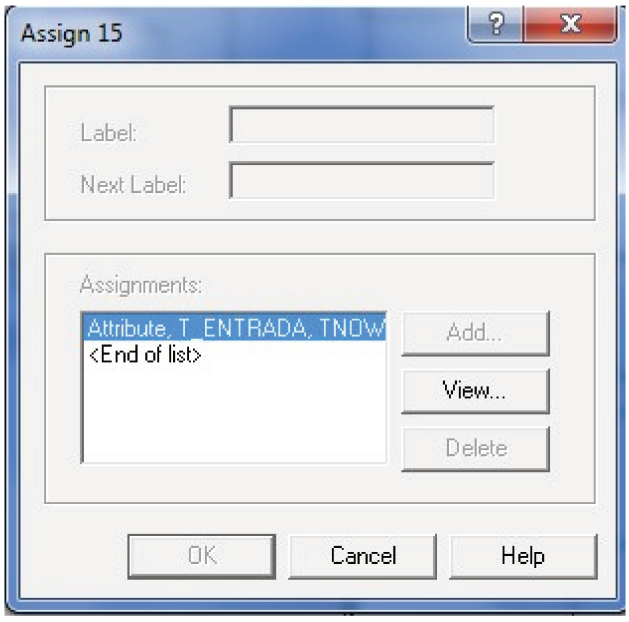


Figura 44: Parâmetros do bloco *ASSIGN*.

Clicando em *Edit* é possível editar as associações

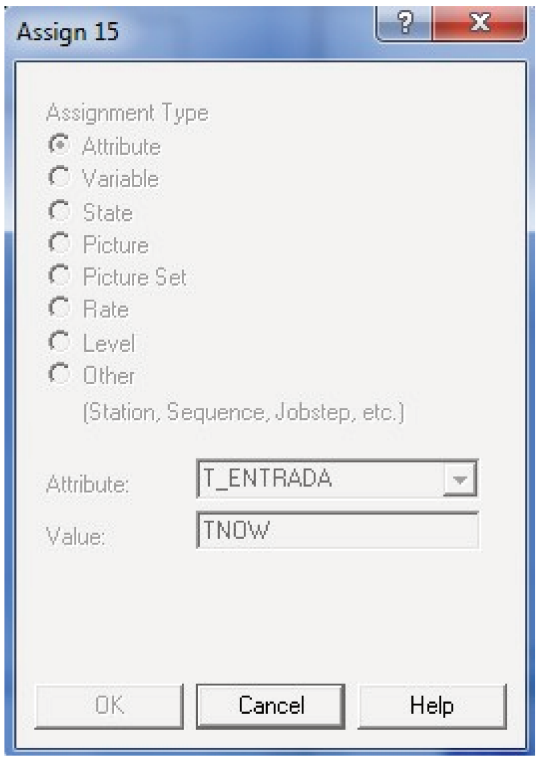


Figura 45: Configurando Associações.

Assigment Type: Indica o tipo de variável que será associada.

Bloco *PROCESS*

O módulo *Process* é usado para definir uma etapa de processamento. Quando uma entidade chega a um módulo *Process*, ela espera até que um servidor esteja disponível. Este servidor pode ser um recurso ou um transportador. E enquanto o recurso estiver sendo utilizado, as outras entidades que chegam ao módulo *Process* têm que esperar.

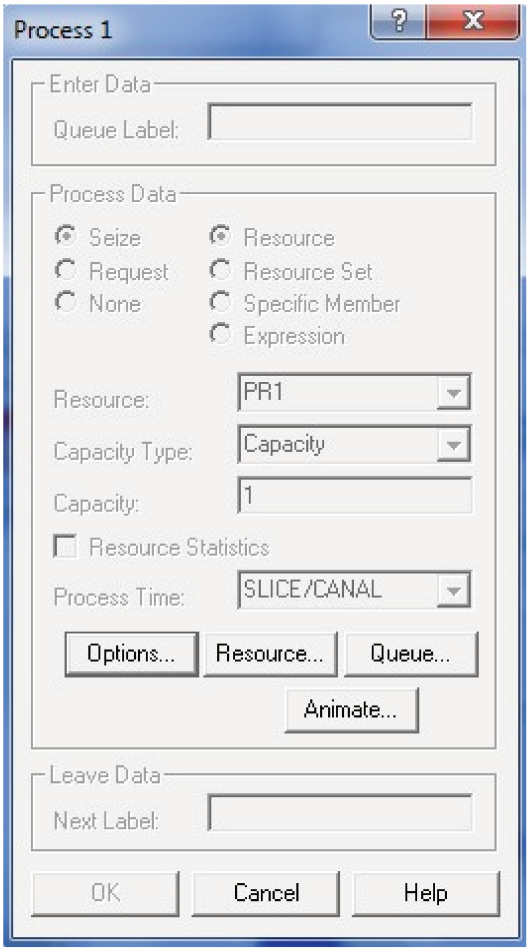


Figura 46: Parâmetros do bloco *PROCESS*.

Queue Label: Define o nome da *Queue Label* associada com o módulo.

Server Action: Define se um recurso (*seize*) ou um transportador (*request*) é requerido pela entidade.

Resource Name: É visível quando o *Server Action for Seize* permite indicar o recurso a ser solicitado. Pode-se optar por um recurso único, um conjunto de recursos, um membro específico de um conjunto ou o recurso pode ser selecionado por uma expressão.

Resource: Indica o nome do recurso a ser capturado.

Capacity Type: Define as características da capacidade de recurso.

Capacity: Define a capacidade do recurso.

Resource Statistics: Esta opção indica se as estatísticas do recurso serão ou não coletadas.

Process Time: Este campo define o tempo de processamento do recurso quando este é ocupado por uma entidade.

Next Label: Define o nome do próximo módulo que a entidade se moverá. Se for feito um link para o próximo módulo, o campo será removido.

No ícone *Queue* podemos configurar as características da fila, como mostra a figura a seguir.

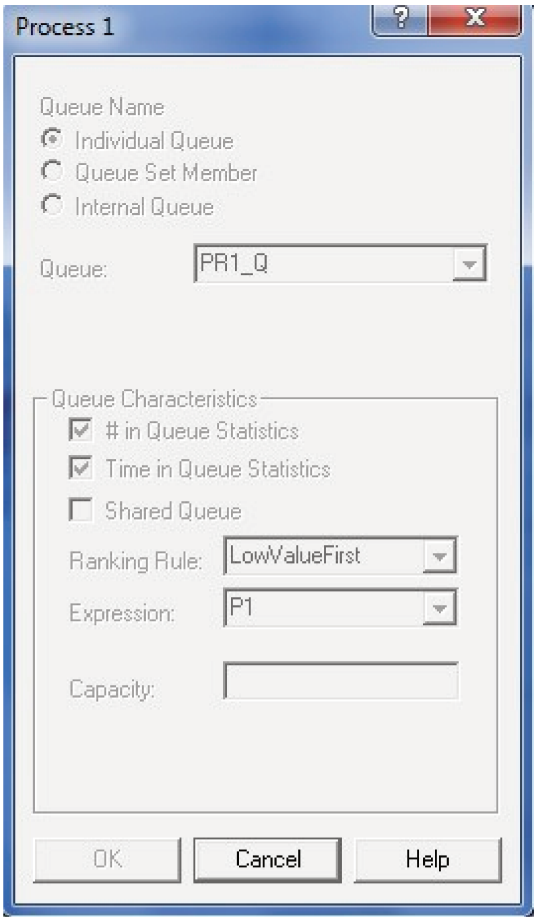


Figura 47: Parâmetros *queue* do bloco *PROCESS.*

Ranking Rule: Define a característica da fila, podendo ser um FIFO, .. . No nosso caso a fila esta

seguindo uma característica de prioridade, LowValorFrist.

Expression: Define qual variável se implica a regra adotada pelo *Ranking Rule.*

Bloco *DEPART*

O módulo *Depart* é usado para coletar as estatísticas do sistema e remover as entidades já processadas do modelo.

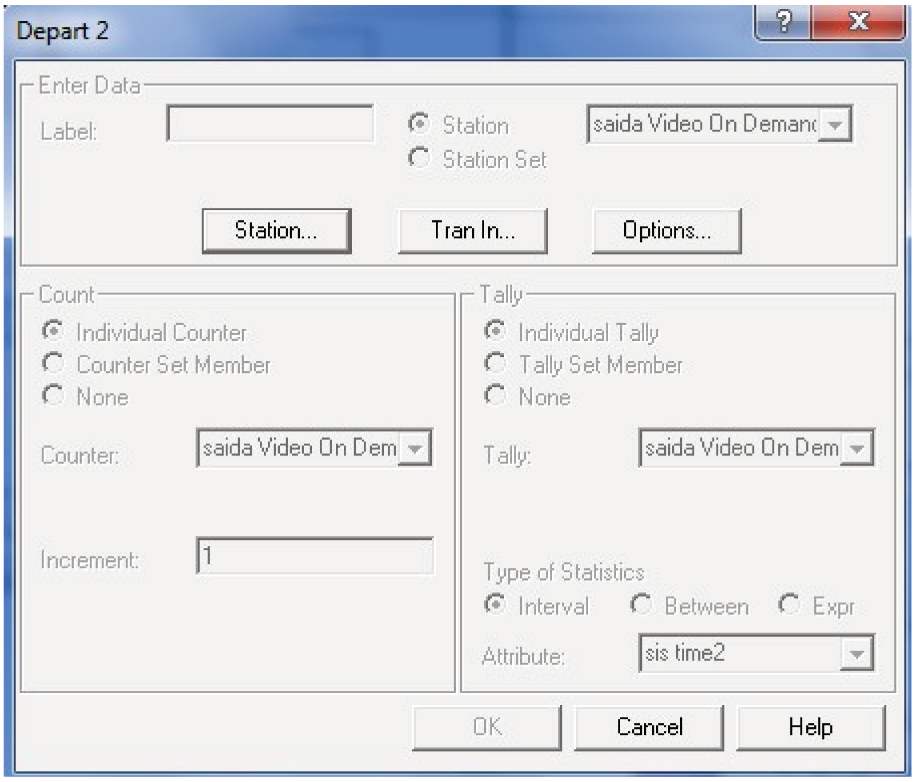


Figura 48: Parâmetros do bloco *DEPART.*

Enter Data

Station: Este campo define o nome da estação que está associada a este módulo.

Count

Counter: Este campo define o nome do contador que será incrementado ou decrementado.

Increment: Este campo define o valor do incremento no contador.

Tally: Este campo é visível quando o *Tally name* escolhido é o *Individual Tally*, e define o nome do *Tally* em que os dados estatísticos são coletados.

Attribute: Este campo é visível se a opção *Interval* é escolhida em *Type of Statistics* e define o nome usado para determinar as estatísticas de intervalo.

Bloco *LEAVE*

O módulo *Leave* é utilizado para transferir uma entidade para uma estação ou um módulo.

Uma conexão gráfica pode ser utilizada para transferir uma entidade para um outro módulo e uma entidade pode ser redirecionada para um módulo através da referência *Label* no campo *Next Level*.

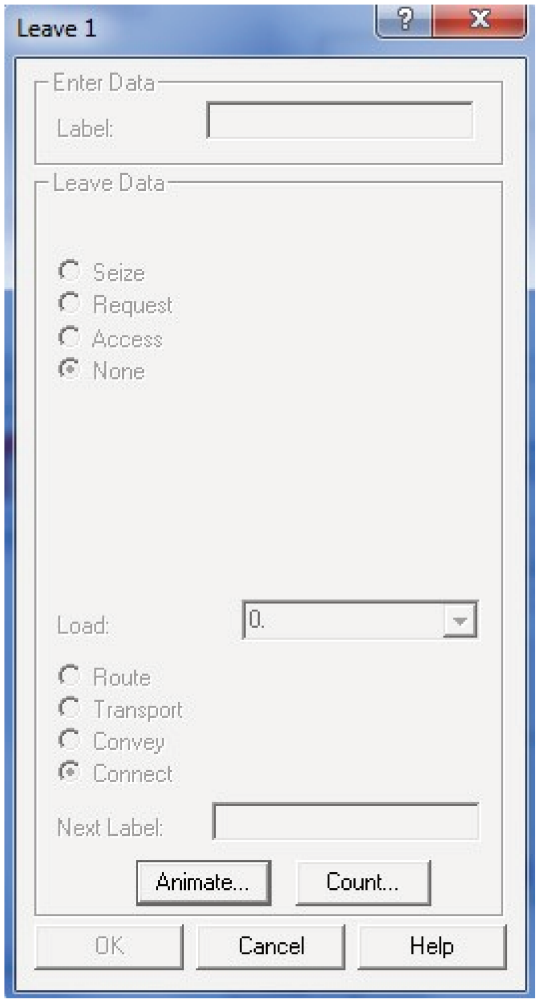


Figura 49: Parâmetros do bloco *LEAVE.*

Label: Este campo define o nome do *label* que será associado a este módulo.

From Station: Este campo especifica a estação de onde a entidade está sendo transferida.

To station: Este campo indica o nome da estação em que as entidades se dirigirão.

Route time: Este campo define o tempo que a entidade leva para se dirigir à outra estação.

Bloco *Variables*

O módulo *Variables* define as variáveis globais que serão utilizadas nos demais módulos do modelo de simulação.

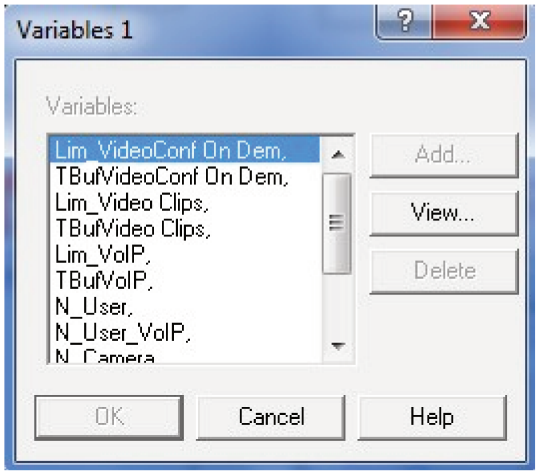


Figura 50: Parâmetros do bloco *Variables.*

Variables: Este campo define o nome das variáveis globais.

Add: Novas variáveis serão adicionadas, neste mesmo campo se define os seus valores.

Bloco *Simulate*

No módulo *Simulate* podemos inserir informações sobre o modelo de simulação. Como o título, data de quando foi feito, e o nome do analista. Além disso neste módulo são inseridas as informações de replicação da simulação.

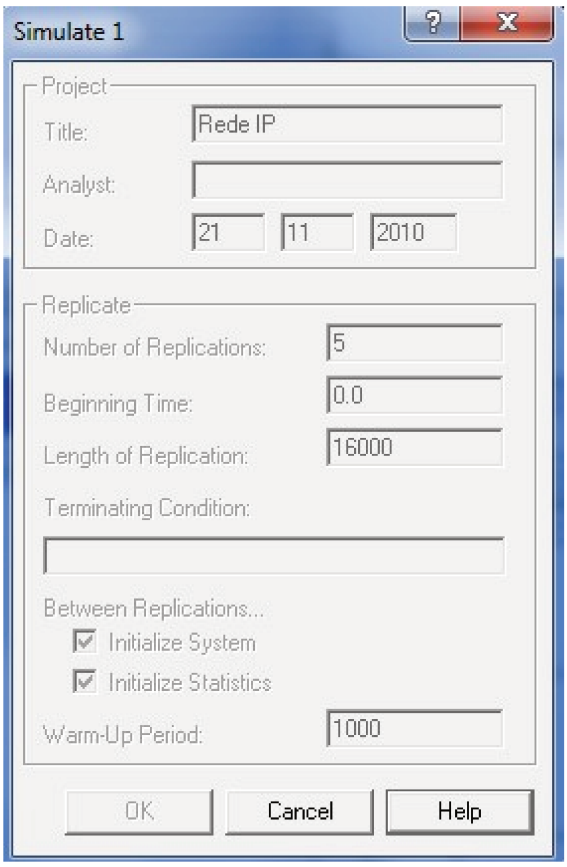


Figura 51: Parâmetros do bloco *Simulate.*

Number of Replications: Este campo define o número inteiro da replicação da simulação a ser executado.

Lenght of Replication: Este campo define o tamanho da replicação a ser simulada. Caso o campo esteja em branco o seu valor será infinito.

Warm-Up Period: Este campo define o tempo desejado para atingir o estado estacionário.