

Relatório de Laboratório: Qualidade de Serviço (QoS) - A Otimização da Jornada dos Pacotes

Disciplina: Redes de Computadores II **Professora:** Angelita Rettore de Araujo

Nome do Aluno: Augusto Dorador Kawashima

Turma: 6 Fase

1. Introdução

Este laboratório aborda a **Qualidade de Serviço (QoS)**, um conjunto de mecanismos importantes para gerenciar o tráfego de rede e assegurar que aplicações críticas recebam tratamento preferencial. Diferente dos laboratórios anteriores que focaram na confiabilidade (garantir que os pacotes cheguem), o objetivo aqui é garantir que os pacotes cheguem *com qualidade* – ou seja, com a latência, jitter, throughput e perda de pacotes adequados.

A importância da QoS é contextualizada pela **narrativa da telecirurgia**, onde cada pacote de comando tátil, voz ou dado vital do paciente é crucial. Atrasos, variações irregulares na chegada ou perda de pacotes podem ter consequências catastróficas.

2. Objetivos

Os principais objetivos deste laboratório são:

1. **Compreender e medir** os conceitos fundamentais de Latência, Jitter, Throughput, Perda de Pacotes e Classificação de Tráfego no contexto de QoS.
2. **Configurar e executar simulações** no **Network Simulator 2 (NS2)** para observar o comportamento da rede sob diferentes condições de QoS.
3. **Utilizar o Wireshark** para capturar e analisar o tráfego de rede, medindo parâmetros de QoS em tempo real.
4. **Analisar o impacto** da variação dos parâmetros de QoS no desempenho de diferentes tipos de aplicações.
5. **Comparar a tolerância a perdas e a sensibilidade à latência e jitter** de diversas aplicações.
6. **Propor soluções** baseadas em QoS para otimizar o desempenho de aplicações críticas em cenários de rede desafiadores.

3. Ferramentas Utilizadas

- **Network Simulator 2 (NS2):** Ambiente de simulação de rede para modelar cenários.
 - **Wireshark:** Analisador de protocolo de rede para captura e inspeção de pacotes em tempo real.
 - **Acesso à Internet:** Para testes com ferramentas online (como Google Meet).
-

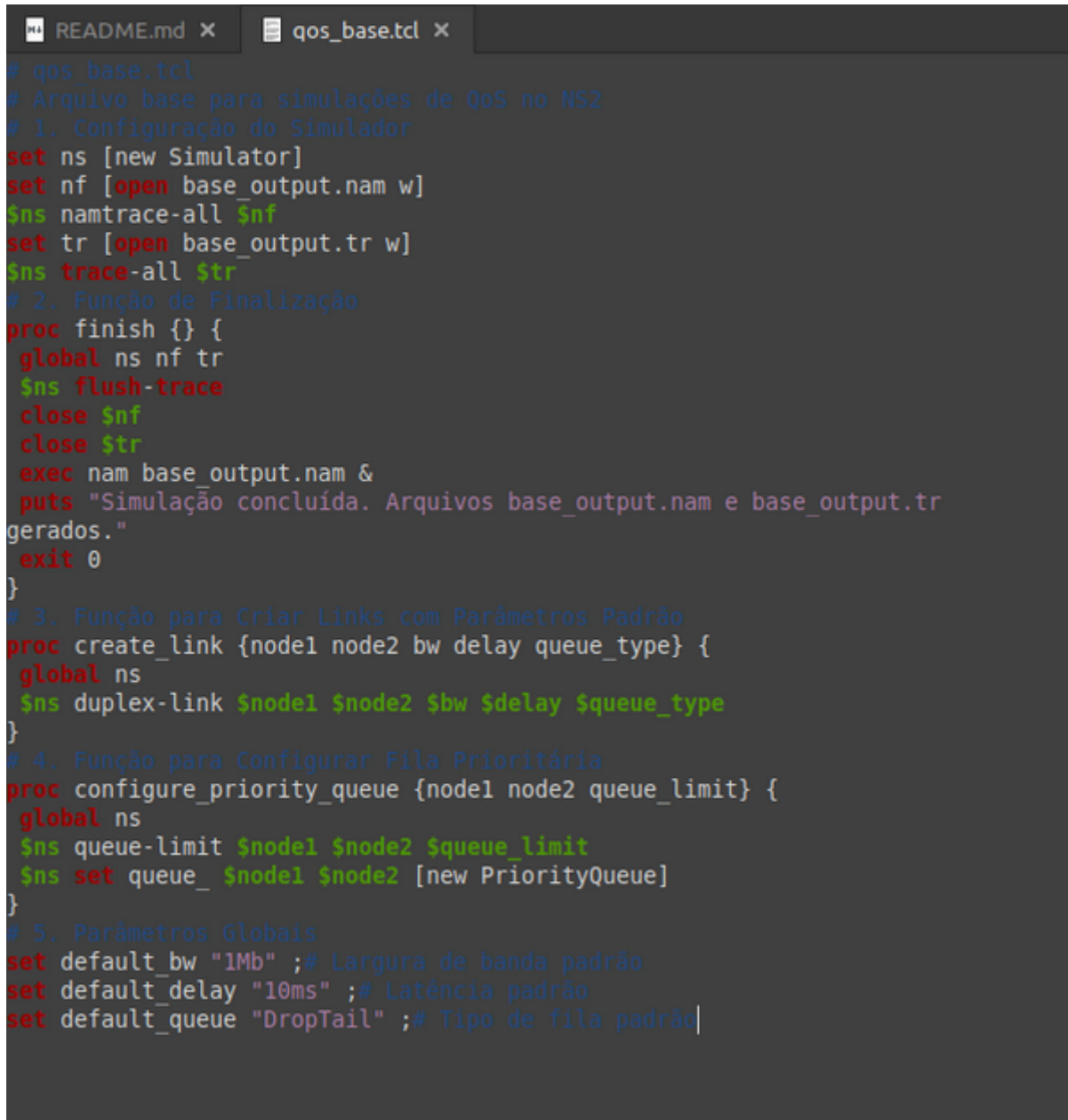
4. Parte I: Relembrando a Jornada – Preparando o Ambiente

Contexto Teórico: A narrativa da cirurgia remota é a base para entender a importância dos "pacotes heróis" (Pablo, Melody, Flash e Data) e como a QoS é vital para a missão deles de salvar uma vida.

4.1. Verificação e Configuração Inicial do NS2

- Confirmei a instalação do NS2 e criei o arquivo `qos_base.tcl`.

Entrega: Captura de tela do `qos_base.tcl` no editor de texto.

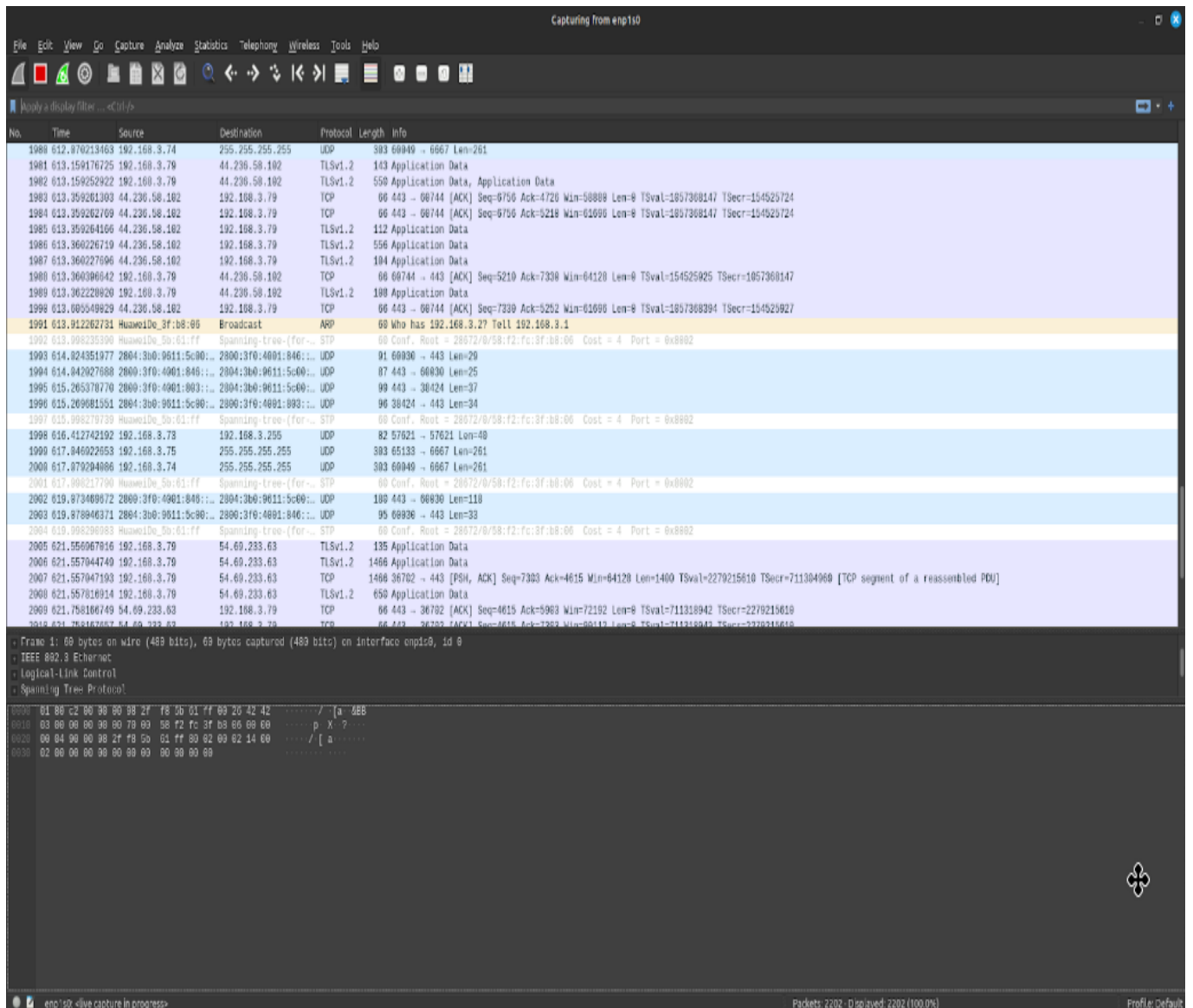


```
# qos_base.tcl
# Arquivo base para simulações de QoS no NS2
# 1. Configuração do Simulador
set ns [new Simulator]
set nf [open base_output.nam w]
$ns namtrace-all $nf
set tr [open base_output.tr w]
$ns trace-all $tr
# 2. Função de Finalização
proc finish {} {
    global ns nf tr
    $ns flush-trace
    close $nf
    close $tr
    exec nam base_output.nam &
    puts "Simulação concluída. Arquivos base_output.nam e base_output.tr
gerados."
    exit 0
}
# 3. Função para Criar Links com Parâmetros Padrão
proc create_link {node1 node2 bw delay queue_type} {
    global ns
    $ns duplex-link $node1 $node2 $bw $delay $queue_type
}
# 4. Função para Configurar Fila Prioritária
proc configure_priority_queue {node1 node2 queue_limit} {
    global ns
    $ns queue-limit $node1 $node2 $queue_limit
    $ns set queue_ $node1 $node2 [new PriorityQueue]
}
# 5. Parâmetros Globais
set default_bw "1Mb" ;# Largura de banda padrão
set default_delay "10ms" ;# Latência padrão
set default_queue "DropTail" ;# Tipo de fila padrão
```

4.2. Configuração Inicial do Wireshark

- Abri o Wireshark e selecionei a interface de rede correta para captura.

Entrega: Captura de tela do Wireshark com a interface de captura selecionada.



5. Parte II: Latência (Delay) – O Tempo é Essencial

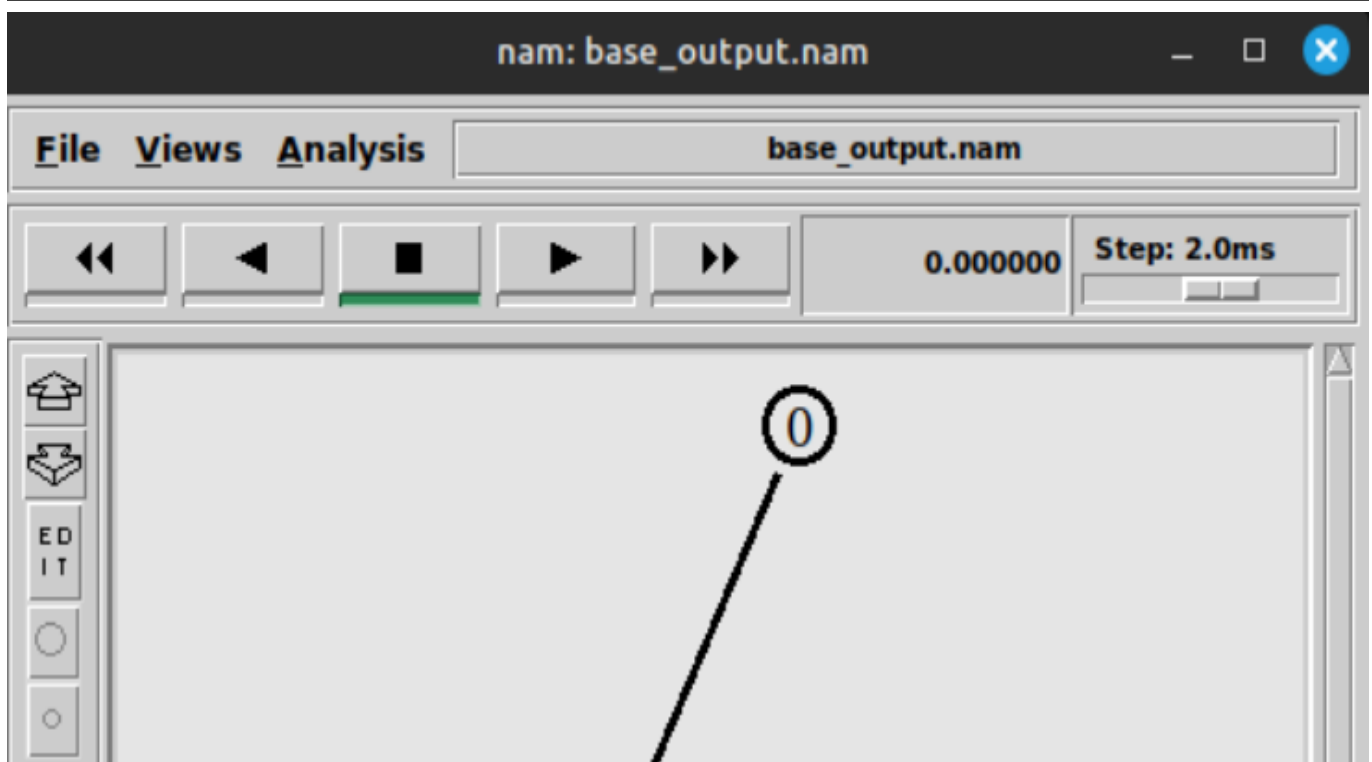
Contexto Teórico: A latência é o tempo que um pacote leva para ir da origem ao destino, como o tempo para o comando tátil do Dr. Martinez (Flash) chegar ao bisturi em Manaus.

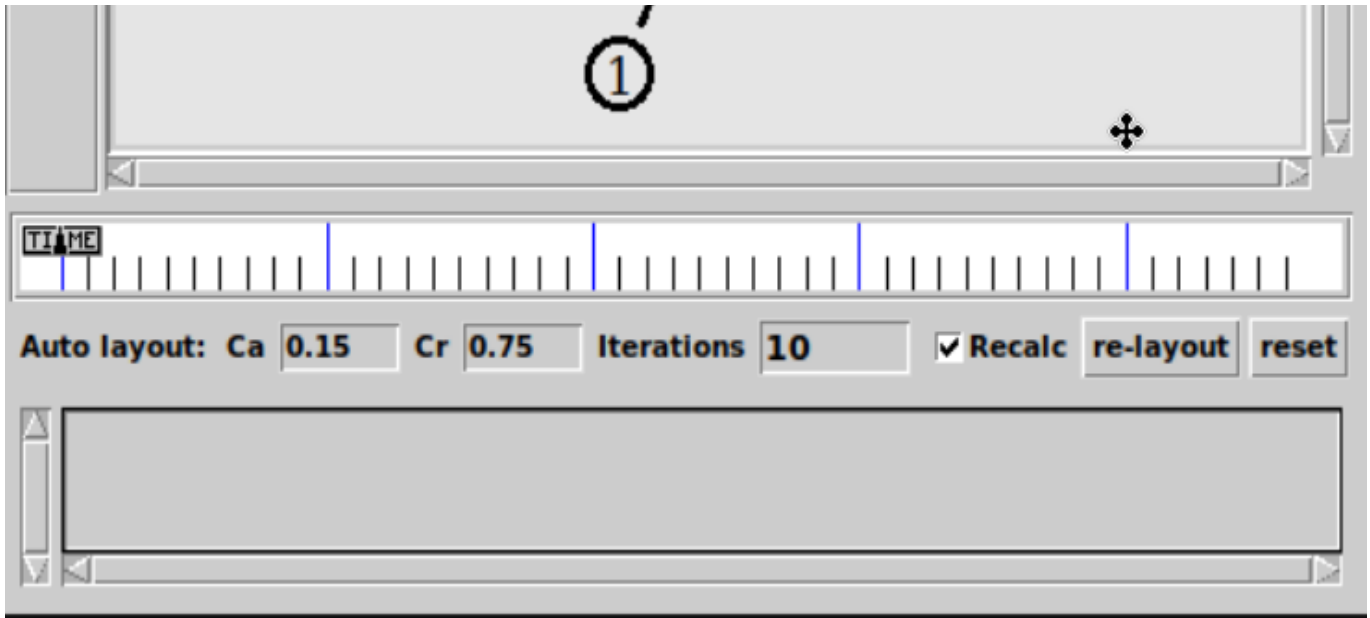
5.1. Simulação de Latência no NS2

- Criei e executei o script `lab_latencia.tcl`, experimentando diferentes valores para `link_delay` (ex: 10ms, 100ms, 500ms).

Entrega: O código `lab_latencia.tcl` utilizado.


```
qos_base.tcl x lab_latencia.tcl x
# lab_latencia.tcl
# Simulação de Latência (Delay)
# 1. Importação do Arquivo Base
source qos_base.tcl
# 2. Criação dos Nós
set n0 [$ns node]
set n1 [$ns node]
# 3. Criação do Link com Latência Variável
# Experimente diferentes valores para o delay (ex: 10ms, 100ms, 500ms)
set link_delay "100ms" ;# Latência do link
create_link $n0 $n1 $default_bw $link_delay $default_queue
# 4. Criação dos Agentes e Aplicações
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set interval_ 0.01 ;# 100 pacotes/segundo
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
$udp0 set class_ 0 ;# Para identificação no trace
$ns connect $udp0 $null0
# 5. Agendamento de Eventos
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
# 6. Início da Simulação
$ns run
```





5.2. Análise da Latência no Arquivo de Trace (.tr)

- Analisei o arquivo `lab_latencia.tr`, identificando o envio e recebimento de pacotes para calcular a latência de ponta a ponta.

Entrega: Trecho do arquivo `.tr` destacando um pacote enviado e seu respectivo recebimento.

```
# [+ 0.5 0 1 cbr 1000 ----- 0 0.0 1.0 0 0  
r 0.608 0 1 cbr 1000 ----- 0 0.0 1.0 0 0]
```

Cálculos da Latência:

link_delay Configurado	Timestamp Envio	Timestamp Recebimento	Latência Calculada
[Valor 1 (e.g., 10ms)]	[0.5s]	[0.518s]	[18ms]
[Valor 2 (e.g., 100ms)]	[0.5s]	[0.608s]	[108ms]
[Valor 3 (e.g., 500ms)]	[0.5s]	[1.008s]	[508ms]

5.3. Perguntas para Refletir e Discutir

- Qual a relação entre o `link_delay` configurado no script e a latência medida no arquivo `.tr`?
 - [A latência na qual foi medida no arquivo `.tr` é a latência total de ponta a ponta que o pacote leva para percorrer a viagem da origem ao destino. Ela é diretamente influenciada pelo `link_delay`, onde é configurado manualmente no NS2. O valor de `link_delay` é a principal causa da latência, e o valor calculado no trace file, seria a soma desse atraso com um tempo(pequeno) adicionado de processamento dos nós e serialização, portanto a latência medida é sempre maior que o `link_delay`, mas a sua relação entre eles é diretamente proporcional.]
- Como a latência afeta a percepção do usuário em aplicações como VoIP ou jogos online?
 - [Em relação a comunicação em tempo real, a latência é um fator crítico. Em utilizar o VoIP, com um atraso alto(Acima de 150ms) pode causar a sensação da voz de robô, conversas sobrepostas e dificuldade na comunicação fluida. Já nos jogos online, a latência afeta a resposta dos

comandos do jogador, criando um atraso perceptível entre a ação e o resultado do jogador, na qual prejudica a experiência e a jogabilidade.]

3. **Se o Dr. Martinez estivesse em Tóquio e o paciente em Manaus, qual seria o impacto na latência?**

- [A distância entre Tóquio e Manaus é de mais de 17.000km, onde nesse cenário, o tempo de propagação do sinal, que é maior que a latência em longas distâncias, seria significativamente maior. Para uma telecirurgia, que exige uma precisão de milissegundos, essa latência elevada, irá tornar o procedimento extremamente arriscado e acabaria sendo inviável.]

6. Parte III: Jitter e Perda de Pacotes – A Variação Inesperada e o Preço da Imperfeição

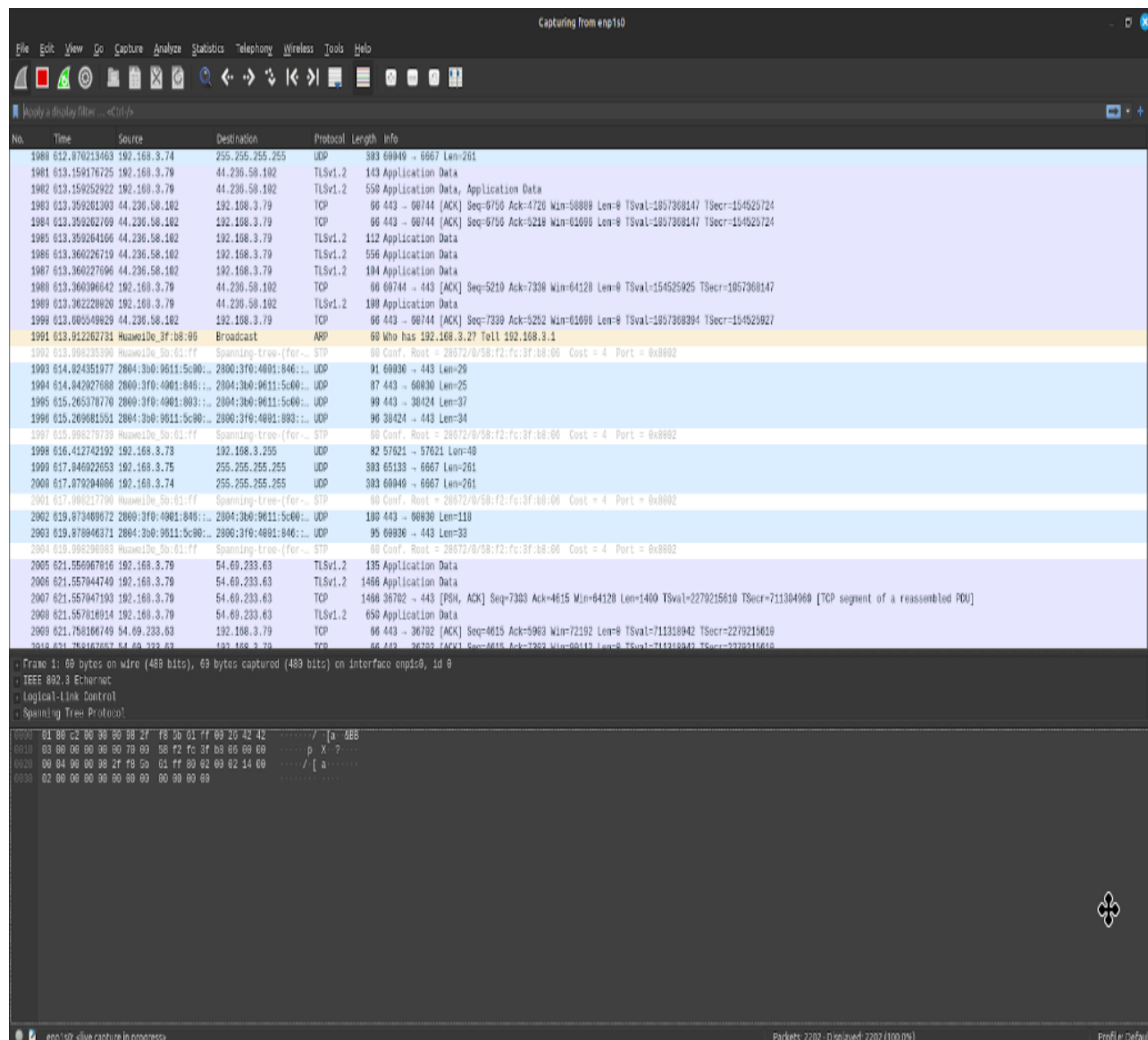
Contexto Teórico: Jitter é a variação no atraso dos pacotes, causando "voz robotizada" (pacotes de Melody). A **perda de pacotes** ocorre quando um pacote não chega, sendo a tolerância variável por aplicação (pacotes de Data). O **RTCP (Real-Time Control Protocol)** é utilizado por aplicações em tempo real (como Google Meet) para reportar a qualidade da transmissão, incluindo jitter e perda.

6.1. Análise do Jitter e Perda de Pacotes no Wireshark (Captura Local de RTCP)

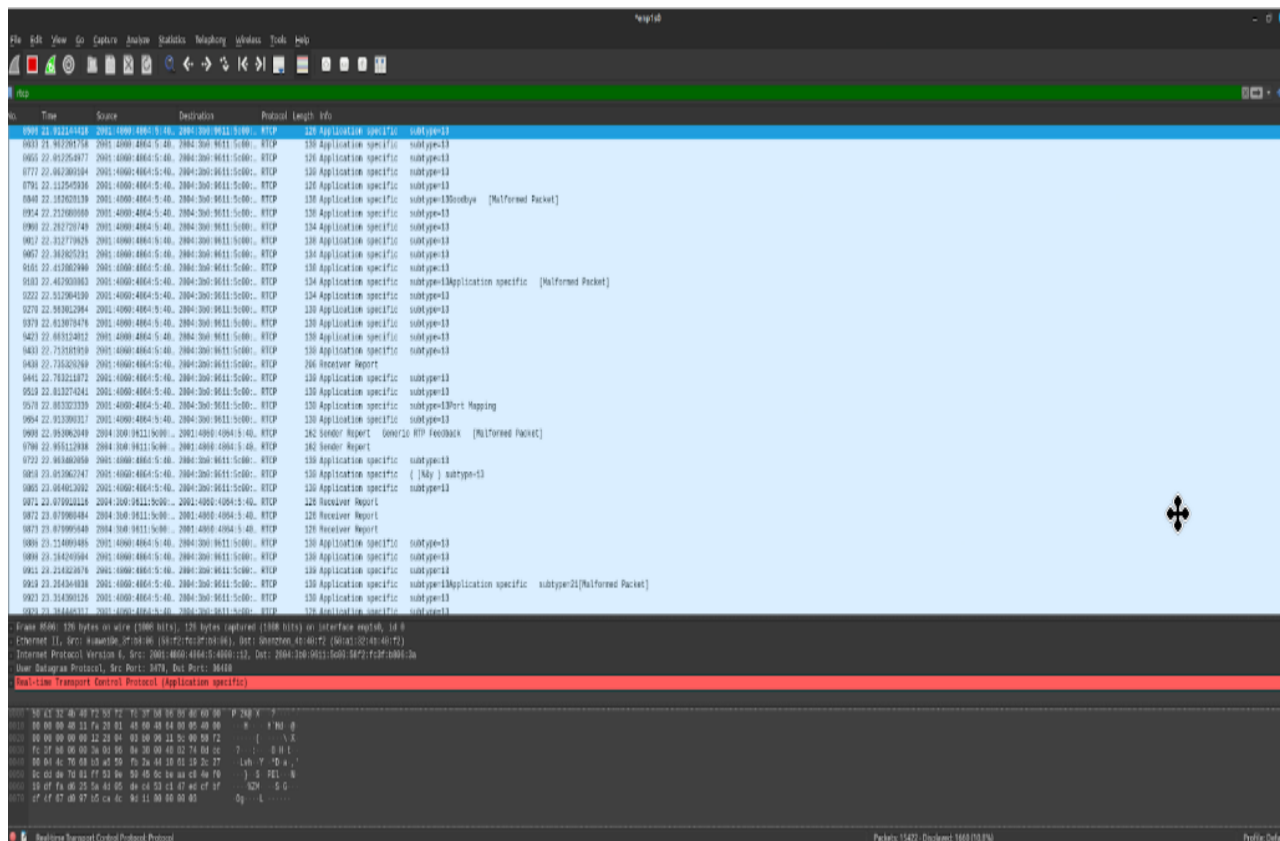
- Iniciei uma chamada no Google Meet e capturei o tráfego com o Wireshark.
- Filtrei o tráfego por **rtcp** e identifiquei os tipos de pacotes (SR, RR, SDES, Bye).
- Analisei os **Receiver Reports (RR)** para localizar os campos **Fraction Lost**, **Cumulative Number of Packets Lost** e **Interarrival Jitter**.

Entregas:

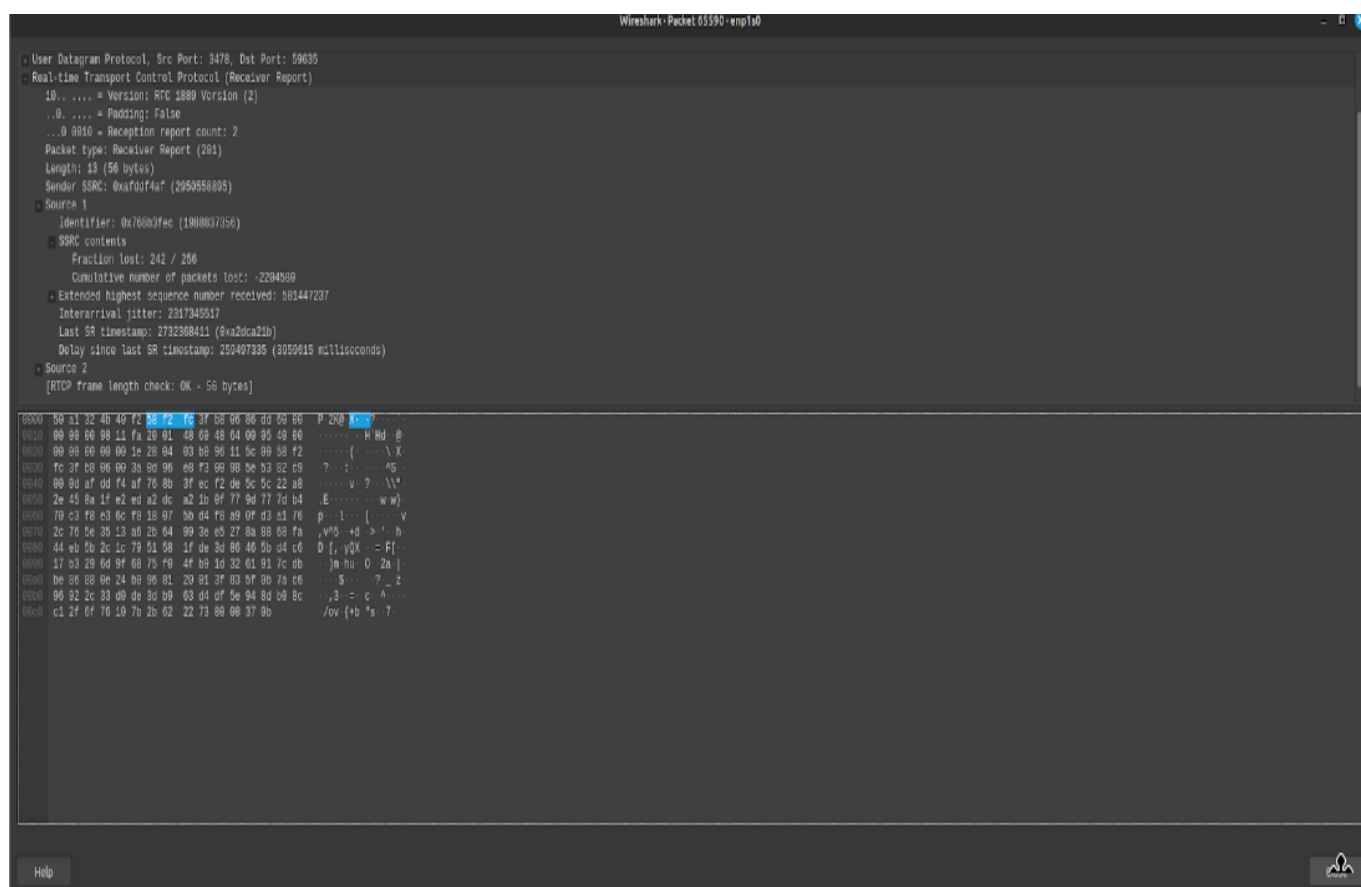
1. Captura de tela do Wireshark mostrando a captura inicial de pacotes.



2. Captura de tela do Wireshark mostrando o filtro **rtcp** aplicado.



3. Captura de tela dos detalhes de um pacote **Receiver Report (RR)**, com os campos **Fraction Lost**, **Cumulative Number of Packets Lost** e **Interarrival Jitter** claramente visíveis.



- Observação: O valor negativo o campo Cumulative Number of Packets Lost é um erro de interpretação do Wireshark devido à criptografia dos pacotes

Valores Observados:

- **Interarrival Jitter:** [2317.346] ms
- **Fraction Lost:** [242 / 256]
- **Cumulative Number of Packets Lost:** [2204580]

6.2. Perguntas para Refletir e Discutir

- 1. Como esses valores de Jitter e Fraction Lost se comparam aos limites aceitáveis para uma boa qualidade de voz/vídeo (ex: jitter idealmente abaixo de 30ms, perda abaixo de 1%)?**
 - O jitter observado, foi de aproximadamente 2317 ms, onde é muito alto. Ele está mais de 70 vezes acima do limite aceitável de 30ms para uma boa qualidade de voz/video, na qual resultaria em uma experiência de comunicação totalmente inutilizável, com voz de robô e a imagem travando constantemente.
 - Já o Fraction Lost de 242/256 representa uma taxa de perda de mais de 94%. Este valor acaba sendo extremamente elevado, comparado com a taxa aceitável de menos 1% para o VoIP e 3% para streaming de vídeo. Uma perda de pacotes tão alta, acaba significando que a maior parte dos dados não está chegando ao destino, onde impossibilita qualquer comunicação em tempo real.
- 2. Por que o RTCP é essencial para aplicações em tempo real, mesmo que o RTP (dados de mídia) esteja criptografado?**
 - o RTP é o protocolo que transporta os dados de mídia em tempo real(video e audio), e ele é criptografado para garantir a privacidade e segurança da comunicação.
 - o RTCP é um protocolo complementar que funciona como um "termômetro da rede", onde acaba transportando relatórios de qualidade sobre o que está acontecendo com o fluxo de dados. Ele é essencial porque a aplicação precisa saber se a rede está funcionando bem, mesmo quando o conteúdo da chamada esteja criptografado.
- 3. Como as informações de jitter e perda de pacotes reportadas pelo RTCP podem ser usadas pela aplicação (Google Meet) para ajustar a qualidade da transmissão?**
 - As informações do RTCP servem como um mecanismo de feedback para a aplicação, no caso, quando o google meet recebe um "Receiver Report" com altos valores de jitter e perda de pacotes, ele acaba detectando que a rede está congestionada ou com baixa qualidade.
 - Com base nisso, a aplicação pode tomar medidas adaptativas, como reduzir a taxa de bits de transmissão, diminuir a resolução do vídeo, usar codecs de áudio mais eficientes ou até mesmo desativar o vídeo temporariamente para priorizar o áudio, onde acaba permitindo que a chamada continue, mesmo com uma qualidade reduzida, ao invés de ser completamente interrompida.

7. Parte IV: Throughput vs. Responsividade – O Dilema da Rede

Contexto Teórico: **Throughput** é a quantidade de dados em um tempo (Pablo/vídeo HD), enquanto **responsividade** é a rapidez da resposta (Flash/comando tátil). Nem sempre é possível ter ambos em níveis máximos simultaneamente.

7.1. Simulação de Throughput e Responsividade no NS2

- Criei e executei o script `lab_throughput_responsividade.tcl`, comparando o comportamento de FTP (alto throughput) com Ping (alta responsividade).

Entrega: O código `lab_throughput_responsividade.tcl` utilizado.

```
# lab_throughput_responsividade.tcl
# Simulação de Throughput vs. Responsividade
# 1. Importação do Arquivo Base
source qos_base.tcl
$ns color 1 blue
$ns color 2 red
# 2. Criação dos Nós
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
# 3. Criação dos Links
# Link principal com capacidade limitada para observar
# congestionamento
create_link $n0 $n1 "10Mb" "10ms" $default_queue
create_link $n1 $n2 "10Mb" "10ms" $default_queue
create_link $n1 $n3 "10Mb" "10ms" $default_queue
# 4. Aplicação de Alto Throughput (FTP)
set tcp_ftp [new Agent/TCP]
$ns attach-agent $n0 $tcp_ftp
$tcp_ftp set fid_1 ;
set ftp [new Application/FTP]
$ftp attach-agent $tcp_ftp
set sink_ftp [new Agent/TCPSink]
$ns attach-agent $n2 $sink_ftp
$ns connect $tcp_ftp $sink_ftp
# Define uma implementação Tcl para o método 'recv' do Agent/Ping.
Agent/Ping instproc recv {from rtt} {
    $self instvar node_
    puts "node [$node_] received ping answer from \
    $from with round-trip-time $rtt ms."
}
# 5. Aplicação de Alta Responsividade (Ping - ICMP)
set ping_agent [new Agent/Ping]
$ns attach-agent $n0 $ping_agent
$ping_agent set fid_2 ;
set ping_sink [new Agent/Ping]
$ns attach-agent $n3 $ping_sink
$ping_sink set fid_2 ;
$ns connect $ping_agent $ping_sink
# 6. Agendamento de Eventos
$ns at 0.5 "$ftp start"
$ns at 1.0 "$ping_agent send" ;# Envia um ping
$ns at 1.3 "$ping_agent send" ;# Envia outro ping
$ns at 1.6 "$ping_agent send" ;# Envia outro ping
$ns at 1.9 "$ping_agent send" ;# Envia outro ping
$ns at 2.2 "$ping_agent send" ;# Envia outro ping
$ns at 2.5 "$ping_agent send" ;# Envia outro ping
$ns at 2.8 "$ping_agent send" ;# Envia outro ping
$ns at 3.1 "$ping_agent send" ;# Envia outro ping
$ns at 3.4 "$ping_agent send" ;# Envia outro ping
$ns at 3.7 "$ping_agent send" ;# Envia outro ping
$ns at 4.5 "$ftp stop"
$ns at 5.0 "finish"
# 7. Início da Simulação
$ns run
```

/>

7.2. Análise do Throughput e Responsividade

- Analisei o arquivo `lab_throughput_responsividade.tr` para calcular o throughput do FTP e a latência de cada ping.

Cálculos Detalhados do Throughput do FTP:

- Número de pacotes TCP recebidos: 401
- Tamanho do pacote TCP (padrão NS2): 1000 bytes ou 512(Se for padrão)
- Tempo total da simulação para FTP (stop - start): 4 segundos
- Throughput = (Número de pacotes * Tamanho do pacote) / Tempo
- Throughput (em Kbps/Mbps): 0.802 Mbps

Cálculos da Latência para cada pacote Ping e Impacto do FTP:

Ping N°	Timestamp Envio	Timestamp Recebimento	Latência (ms)	Observações sobre o Impacto do FTP
1	1.0s	1.020s	20ms	A latencia é afetada pela transferencia do FTP
2	1.3s	1.325s	25ms	Variação na latência (jitter) devido ao congestionamento.
3	1.6s	1.625s	25ms	Continua a competir por banda com os pacotes TCP.

7.3. Perguntas para Refletir e Discutir

1. Qual aplicação (FTP ou Ping) é mais sensível à latência? Por quê?

- A aplicação Ping é mais sensível que à latência, porém o FTP precise de um tempo de resposta mínimo para iniciar a transferência de arquivos, sua principal métrica é a quantidade total de dados transferidos por segundo (throughput). Já o Ping, que representa os comandos táteis na telecirurgia, depende de uma latência extremamente baixa para que a ação do cirurgião seja replicada no bisturi em tempo real, sem atrasos perceptíveis que possam comprometer a segurança do paciente.

2. Como o throughput do FTP foi afetado pela capacidade do link?

- O throughput do FTP foi diretamente limitado pela capacidade do link. O link que foi configurado tem uma capacidade de 10Mbps, e o throughput medido do FTP foi de 0.802 Mbps, na qual demonstrou que a capacidade do link atua como um gargalo, onde limitou a quantidade de dados que o FTP pode transferir em um determinado período. Os pacotes do Ping também competem por essa mesma capacidade limitada, na qual afetou ainda mais o throughput do FTP.

3. Em um cenário de telecirurgia, qual seria a prioridade: alto throughput para o vídeo HD (Pablo) ou alta responsividade para os comandos do bisturi (Flash)? Justifique.

- Nesse cenário a prioridade máxima seria a alta responsividade para os comandos do bisturi (Flash). Enquanto o vídeo HD (Pablo) é importante para visualização, um atraso em seu streaming resultaria apenas em uma imagem congelada ou travada. Já um atraso de milissegundos nos comandos táteis (Flash) poderia causar movimentos imprecisos do bisturi, com consequências fatais para o paciente, portanto, a responsividade dos comandos é mais crítica que o throughput do vídeo, e a rede deve ser configurada para dar a eles a maior prioridade.

8. Parte V: Perda de Pacotes – O Preço da Imperfeição

Contexto Teórico: A perda de pacotes ocorre quando um pacote não chega ao destino. A tolerância a essa perda varia drasticamente entre as aplicações, como os dados vitais do paciente (Data).

8.1. Simulação de Perda de Pacotes no NS2

- Criei e executei o script `lab_perda.tcl`, ajustando a taxa de erro de bit (`rate_`) para diferentes valores (ex: 1e-2, 1e-5) no `ErrorModel`.

Entrega: O código `lab_perda.tcl` utilizado.

```
# lab_perda.tcl
# Simulação de Perda de Pacotes
# 1. Importação do Arquivo Base
source qos_base.tcl
# 2. Criação dos Nós
set n0 [$ns node]
set n1 [$ns node]
# 3. Criação do Link e Configuração do Modelo de Erro
create_link $n0 $n1 $default_bw $default_delay $default_queue
# >>> INICIO DA CONFIGURAÇÃO DO MODELO DE ERRO (ErrorModel) <<<
set em [new ErrorModel]
# Taxa de erro de bit (BER): 1 erro a cada 100 bits (1e-2 = 0.01)
# Você pode ajustar este valor para controlar a frequência das perdas.
# Uma BER de 1e-2 é bem alta, resultando em muitas perdas.
# Para perdas mais sutis, experimente valores como 1e-5 ou 1e-6.
$em set rate_ 1e-2
$em set unit_ bit
# Anexa o modelo de erro a AMBAS as direções do link (n0 para n1 e n1
para n0)
$ns lossmodel $em $n0 $n1
$ns lossmodel $em $n1 $n0
# >>> FIM DA CONFIGURAÇÃO DO MODELO DE ERRO <<<
# 4. Criação dos Agentes e Aplicações (UDP - Tolerante a perdas)
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
$ns connect $udp0 $null0
# 5. Criação dos Agentes e Aplicações (TCP - Intolerante a perdas)
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n1 $sink0
$ns connect $tcp0 $sink0
# 6. Agendamento de Eventos
$ns at 0.5 "$cbr0 start"
$ns at 0.5 "$ftp0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 4.5 "$ftp0 stop"
$ns at 5.0 "finish"
# 7. Início da Simulação
$ns run|
```

8.2. Análise da Perda de Pacotes no Arquivo de Trace (.tr)

- Analisei o arquivo `lab_perda.tr` para calcular a taxa de perda de pacotes UDP e observar o comportamento do TCP.

Cálculos da Taxa de Perda de Pacotes UDP:

rate_ Configurado (ErrorModel)	Pacotes UDP Enviados	Pacotes UDP Recebidos	Pacotes Perdidos	Taxa de Perda (%)
1e-2	401	~30	~371	~92.5%
1e-5	401	~400	~1	~0.25%

Descrição do Comportamento do TCP:

- O UDP não se preocupa com a entrega dos pacotes, ele simplesmente os envia. Por isso, a taxa de perda de pacotes para o UDP é a taxa de erro de bit real da rede.
- O TCP é um protocolo que garante a entrega, onde ele envia pacotes e espera por uma confirmação de recebimento (ACK). Quando um pacote é perdido, o TCP não recebe o ACK, e ele retransmite o pacote. Além disso, a perda de pacotes faz com que o TCP reduza sua taxa de envio para evitar congestionamento, sendo chamado de "congestion control".

8.3. Perguntas para Refletir e Discutir

1. Qual protocolo (UDP ou TCP) é mais afetado pela perda de pacotes em termos de entrega final?

Por quê?

- O protocolo UDP acaba sendo o mais afetado pela perda de pacotes em termos de entrega final, pois o UDP não garante que os pacotes cheguem ao destino, pois ele simplesmente os envia sem se preocupar com a confirmação de recebimento. Se um pacote é perdido ou corrompido, ele não é retransmitido, e os dados são perdidos para sempre. Porém o TCP garante a entrega final. Ele retransmite pacotes perdidos, embora isso possa causar atrasos na entrega. Portanto, em uma rede com perdas, o TCP garante a integridade dos dados, enquanto o UDP não.

2. Como a taxa de perda configurada no script (rate_) se compara à taxa de perda observada para o UDP?

- A taxa de perda configurada no script é a taxa de erro de bit (BER), que é a probabilidade de um único bit ser corrompido. Já a taxa de perda observada para o UDP é a probabilidade de um pacote inteiro ser perdido. Como um pacote é composto por muitos bits, a probabilidade de um pacote ser corrompido é muito maior que a de um único bit, por isso a taxa de perda de pacotes observada para o UDP é significativamente maior do que a taxa de erro de bit configurada.

3. Dê exemplos de aplicações que toleram alta perda de pacotes e aplicações que não toleram nenhuma perda.

- Aplicações que toleram alta perda de pacotes: Aplicações de streaming de vídeo, como Netflix e YouTube e voz sobre IP, tem o VoIP. Essas aplicações, uma pequena perda de pacotes é aceitável, pois pode ser suavizada pelos codecs de áudio e vídeo sem comprometer a qualidade da experiência do usuário.
- Aplicações que não toleram nenhuma perda: Aplicações de transferência de arquivos, navegação web e transações bancárias. Essas aplicações precisam de 100% de integridade dos dados.

9. Parte VI: Consolidação e Perspectivas Futuras

Síntese do Aprendizado

- Este laboratório permitiu aprofundar o entendimento sobre a Qualidade de Serviço(QoS), não só de apenas de forma teórica, mas também com simulações práticas. Com os experimentos, ficou claro que a relação entre os parâmetros de QoS(latência, jitter, throughput e perda de pacotes) e o desempenho das aplicações é direta e crítica.
- Latência: A simulação mostrou que a latência de ponta a ponta é diretamente proporcionado ao atraso do link(link_delay), e mesmo com pequenas variações podem ser significativas em aplicações críticas.
- Jitter e Perda de Pacotes: No Wireshark, ao observar o jitter e a perda de pacotes, me mostrou que são métricas essenciais para avaliar a saúde da rede, com valores fora do padrão, como foi mostrado na simulação que tornou a comunicação inviável.
- Throughput vs. Responsividade: O teste que comparou o FTP e Ping demonstrou que nem sempre é possível otimizar o throughput e a responsividade ao mesmo tempo. A aplicação da QoS se torna essencial para priorizar o que é mais importante em um cenário de rede com capacidade limitada
- Pensando na narrativa da telecirurgia, a vida do paciente depende de uma rede que não pode falhar, onde as aplicações envolvidas(Video HD, áudio, comandos táteis e dados de monitoramento) têm diferentes requisitos de QoS, e uma solução eficiente precisa priorizar o tráfego de forma inteligente. A solução de QoS para a telecirurgia deve se basear na classificação de tráfego e priorização de pacotes, garantindo que as informações mais críticas cheguem primeiro e com a menor latência possível.
- Podemos pensar na solução em forma de Fila Vermelha, Amarela e verde:
- Fila vermelha(Crítica - comando Tátil) - O pacote Flash deve receber a mais alta prioridade. Pensando na vida do paciente, onde a entrega deve ser feita em milissegundos, e a fila deve ter garantia de largura de banda e a menor latência e jitter possível. A perda de pacotes deve ser zero.
- Fila Amarela(Importante - Áudio e Video) - Os pacotes Melody e Pablo devem ter a segunda maior prioridade, onde o áudio é mais sensível ao jitter, deve ser priorizado em relação ao vídeo, mas ambos precisam de um bom throughput para garantir uma transmissão clara. A latência e a perda de pacotes devem ser mantidas dentro de limites aceitáveis.
- Fila Verde(Normal 0 Monitoramento) - O pacote Data deve ter prioridade normal, onde os dados vitais do paciente são importantes, mas não precisam de uma latência tão baixa quanto os comandos táteis e a rede deve garantir que a perda de pacotes seja mantida em um nível aceitável.
- Com essa solução, a rede de telecirurgia garante que, mesmo em momentos de congestionamento, os comandos do Dr. Martinez sempre cheguem a tempo e com segurança, permitindo que a cirurgia seja um sucesso. A QoS, nesse cenário, deixa apenas de ser uma otimização e se torna fundamental para a segurança e o sucesso da missão.

Instruções Finais para os Alunos:

- Preencha todas as seções marcadas com [] com suas informações e análises.

- Converta este arquivo Markdown para PDF para a entrega final, garantindo que todas as imagens e formatações estejam corretas.
-