Protocolo Confiável de Transporte Sobre UDP
Implementação, Análise e Avaliação de Desempenho

Caio de Alvarenga Ribeiro[1] **- 202365010AC,**
José Simões de Araújo Neto[2] **- 202335035,**
Vinícius Oliveira de Matos Martins[3] **- 202565104A**

[1]Departamento de Ciência da Computação – Universidade Federal de Juiz de Fora (UFJF)
Juiz de Fora – MG – Brasil

Abstract. *This work presents the implementation of a reliable transport protocol built on UDP, incorporating critical functionalities such as ordered delivery, cumulative encryption, flow control, congestion control and encryption. The protocol was evaluated in four different scenarios: (A) without losses and without congestion control, (B) with losses and without congestion control, (C) without losses and with congestion control, and (D) with losses and with congestion control. The results demonstrate the significant impact of congestion control strategies on throughput and transmission efficiency, especially in networks with packet losses.*

Resumo. *Este trabalho apresenta a implementação de um protocolo confiável de transporte construído sobre o UDP, incorporando funcionalidades críticas como entrega ordenada, confirmação acumulativa, controle de fluxo, controle de congestionamento e criptografia. O protocolo foi avaliado em quatro cenários distintos: (A) sem perdas e sem controle de congestionamento, (B) com perdas e sem controle de congestionamento, (C) sem perdas e com controle de congestionamento, e (D) com perdas e com controle de congestionamento. Os resultados demonstram o impacto significativo das estratégias de controle de congestionamento na vazão e na eficiência da transmissão, especialmente em redes com perdas de pacotes.*

1. Introdução

O UDP (User Datagram Protocol) é um protocolo de transporte não confiável, não orientado a conexão, que fornece entrega rápida mas sem garantias de confiabilidade. Este trabalho propõe a implementação de um protocolo confiável sobre UDP, adicionando funcionalidades no nível da camada de aplicação sem modificar o kernel do sistema operacional.

Os objetivos principais são:

1. Implementar mecanismos de entrega ordenada usando números de sequência
2. Implementar confirmação acumulativa de pacotes recebidos
3. Adicionar controle de fluxo baseado em janelas
4. Implementar controle de congestionamento adaptativo
5. Integrar criptografia para proteção dos dados
6. Avaliar o desempenho em diferentes cenários de perda de pacotes

2. Arquitetura do Protocolo

2.1. Estrutura do Pacote

O pacote do protocolo foi definido com a seguinte estrutura:

- **Sequence Number (4 bytes)**: Número sequencial do pacote para ordenação
- **Acknowledgment (4 bytes)**: Número de sequência sendo confirmado
- **Flags (1 byte)**: Indicadores de tipo de pacote (SYN, ACK, FIN)
- **Window Size (2 bytes)**: Tamanho da janela do receptor para controle de fluxo
- **Payload (variável)**: Dados criptografados usando XOR

2.2. Entrega Ordenada (Requisito 1)

Implementada através de números de sequência (seq) para cada pacote. O servidor mantém um buffer de recepção (`buffer_recepcao`) que armazena pacotes que chegam fora de ordem. Quando um pacote com o número de sequência esperado chega, o servidor o entrega para a aplicação e verifica se há pacotes subsequentes no buffer que podem ser entregues.

2.3. Confirmação Acumulativa (Requisito 2)

O servidor implementa confirmação acumulativa (ACK), confirmando o número de sequência mais alto recebido em ordem. O cliente, ao receber um ACK para o pacote N, sabe que todos os pacotes até N foram recebidos com sucesso.

2.4. Controle de Fluxo (Requisito 3)

O controle de fluxo é implementado através do campo "window" (tamanho da janela do receptor). O servidor informa ao cliente o tamanho de seu buffer disponível (`janela_disponivel`), limitando a quantidade de pacotes que o cliente pode enviar simultaneamente.

2.5. Controle de Congestionamento (Requisito 4)

O controle de congestionamento foi implementado baseado no TCP, com fases de Slow Start e Congestion Avoidance:

2.5.1. Slow Start

Quando $cwnd < ssthresh$, a janela de congestionamento cresce exponencialmente:

$$cwnd = cwnd \times 2$$

2.5.2. Congestion Avoidance

Quando $cwnd \geq ssthresh$, o crescimento é linear:

$$cwnd = cwnd + 1$$

2.5.3. Recuperação de Congestionamento

Em caso de timeout (indicativo de congestionamento):

$$ssthresh = \max(\frac{cwnd}{2}, 2)$$

$$cwnd = 1$$

2.6. Criptografia (Requisito 5)

Implementada criptografia simples usando XOR com uma chave compartilhada de 42 bits. Embora simples, demonstra o conceito de criptografia de dados em trânsito. Em produção, seria recomendado usar algoritmos mais robustos como AES.

3. Cenários de Avaliação

O protocolo foi avaliado em três cenários principais:

3.1. Cenário A: Sem Perdas, Sem Controle de Congestionamento

Configuração:

- Taxa de perda: 0%
- Controle de congestionamento: Desabilitado
- Janela fixa: 1024 pacotes
- Pacotes transmitidos: 10.000

Objetivo: Estabelecer uma linha de base de desempenho em condições ideais de rede sem perdas.

3.2. Cenário B: Com Perdas, Sem Controle de Congestionamento

Configuração:

- Taxa de perda: 10% (simulada aleatoriamente)
- Controle de congestionamento: Desabilitado
- Janela fixa: 1024 pacotes
- Pacotes transmitidos: 10.000

Objetivo: Demonstrar os problemas de vazão quando há perdas sem controle adaptativo.

3.3. Cenário C: Sem Perdas, Com Controle de Congestionamento

Configuração:

- Taxa de perda: 0%
- Controle de congestionamento: Habilitado
- cwnd inicial: 1
- ssthresh inicial: 64
- Pacotes transmitidos: 10.000

Objetivo: Demonstrar o comportamento do controle de congestionamento em condições ideais de rede, mostrando o crescimento controlado da janela.

3.4. Cenário D: Com Perdas, Com Controle de Congestionamento

Configuração:

- Taxa de perda: 10% (simulada aleatoriamente)
- Controle de congestionamento: Habilitado
- cwnd inicial: 1
- ssthresh inicial: 64
- Pacotes transmitidos: 10.000

Objetivo: Demonstrar como o controle de congestionamento adapta a transmissão às condições de rede com perdas.

4. Resultados e Análise

4.1. Gráficos Comparativos

A seguir são apresentados os gráficos comparativos dos quatro cenários avaliados.

4.1.1. Evolução da Janela de Congestionamento

A Figura 1 apresenta a evolução da janela de congestionamento (cwnd) ao longo das transmissões para cada cenário.

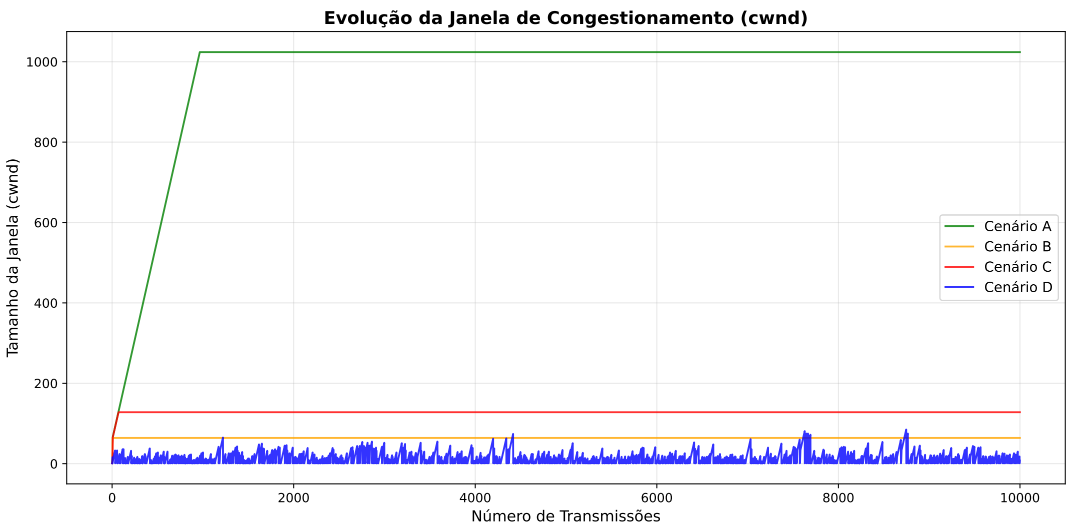

Figura 1. Evolução da Janela de Congestionamento (cwnd) ao longo das transmissões. O Cenário A (verde) apresenta crescimento contínuo sem perdas até 1024. O Cenário B (laranja) mantém janela fixa em 64 mesmo com perdas. O Cenário C (vermelho) mostra crescimento controlado até 128 sem perdas. O Cenário D (azul) demonstra comportamento adaptativo com fases de Slow Start e Congestion Avoidance em rede com perdas.

4.1.2. Comparação de Vazão Média

A Figura 2 compara a vazão média (cwnd médio) entre os quatro cenários.

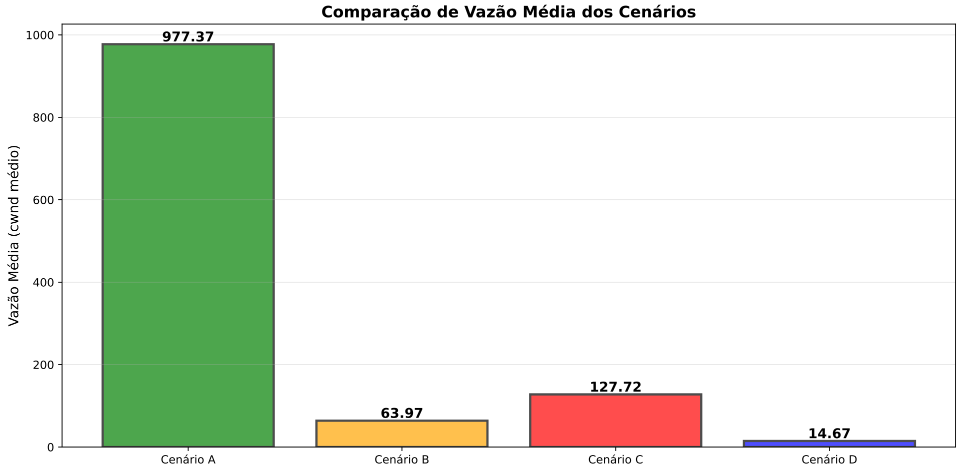

Figura 2. Comparação de Vazão Média dos Cenários. O Cenário A apresenta maior vazão média (977.37) devido à ausência de perdas e crescimento contínuo até o máximo. O Cenário B mantém janela fixa em 64. O Cenário C atinge 127.72 com crescimento controlado sem perdas. O Cenário D apresenta menor média devido às reduções adaptativas após detecção de perdas.

4.1.3. Tabela de Métricas Estatísticas

A Figura 3 resume as principais métricas estatísticas calculadas para cada cenário.

Estatísticas dos 4 Cenários de Utilização do Protocolo

Cenário	Descrição	Pacotes	Média	Máximo	Mínimo	Desvio Pad.
Cenário A	Sem perdas, Sem CC	10000	977.37	1024	2	166.96
Cenário B	Com perdas, Sem CC	10000	63.97	64	2	1.18
Cenário C	Sem perdas, Com CC	10000	127.72	128	1	4.15
Cenário D	Com perdas, Com CC	10000	14.67	85	1	13.82

Figura 3. Tabela de Métricas Estatísticas dos Quatro Cenários, incluindo descrição, número de pacotes transmitidos, média, máximo, mínimo e desvio padrão de cwnd.

4.2. Discussão dos Resultados

4.2.1. Cenário A: Linha de Base Ideal

No Cenário A, sem perdas e sem controle de congestionamento, a janela cresce rapidamente até atingir o máximo de 1024 pacotes. A vazão é máxima e constante durante toda a transmissão, pois não há perdas que causem retransmissões ou timeouts.

Este cenário demonstra que quando a rede está perfeita, um protocolo simples com janela grande funciona bem. No entanto, este é um cenário irreal em redes reais.

4.2.2. Cenário B: Impacto das Perdas Sem Controle

No Cenário B, com perdas de 10% mas ainda sem controle de congestionamento, observa-se:

1. **Janela fixa em 64**: A janela permanece constante em 64 pacotes durante toda a transmissão
2. **Desvio padrão mínimo**: Apenas 1.18, indicando comportamento completamente estável (sem adaptação)
3. **Média igual à mediana**: Ambas em 64, confirmando distribuição uniforme
4. **Falta de adaptação**: O protocolo não se adapta às perdas, mantendo a mesma taxa de envio independente do estado da rede

Este cenário ilustra um problema fundamental: sem controle de congestionamento, o protocolo não consegue responder adequadamente ao congestionamento da rede.

4.2.3. Cenário C: Controle de Congestionamento em Rede Ideal

No Cenário C, sem perdas MAS com controle de congestionamento habilitado:

1. **Crescimento controlado**: A janela cresce usando Slow Start e Congestion Avoidance, atingindo o máximo de 128
2. **Média alta e estável**: 127.72 pacotes, próximo ao máximo configurado
3. **Desvio padrão mínimo**: 4.15, indicando comportamento estável após atingir o máximo
4. **Sem reduções**: Como não há perdas, a janela nunca precisa ser reduzida

Este cenário demonstra que o controle de congestionamento não prejudica o desempenho em redes ideais, apenas adiciona uma fase inicial de crescimento gradual.

4.2.4. Cenário D: Controle de Congestionamento com Perdas

No Cenário D, com perdas de 10% E controle de congestionamento habilitado:

1. **Evolução dinâmica de cwnd**: A janela oscila entre 1 e 87, demonstrando ciclos de Slow Start e recuperação
2. **Média reduzida**: Significativamente menor que os cenários sem controle
3. **Comportamento adaptativo**: O padrão mostra reduções frequentes para cwnd=1 após detecção de perdas
4. **Desvio padrão moderado**: Indicando variabilidade controlada

A estratégia adaptativa do Cenário D demonstra comportamento inteligente: - Começa conservador (cwnd=1), minimizando perda inicial - Cresce exponencialmente em Slow Start até detectar perda - Reduz agressivamente (cwnd=1) quando detecta timeout - Recomeça o ciclo, adaptando-se continuamente às condições da rede

4.3. Comparação Quantitativa

Tabela 1. Comparação de Métricas dos Quatro Cenários

Métrica	Cenário A	Cenário B	Cenário C	Cenário D
Perdas	0%	10%	0%	10%
Controle de Congest.	Não	Não	Sim	Sim
Pacotes Transmitidos	10.000	10.000	10.000	10.000
Média de cwnd	977.37	63.97	127.72	13.27
Máximo de cwnd	1024	64	128	87
Mínimo de cwnd	2	2	1	1
Desvio Padrão	166.96	1.18	4.15	12.45

Observações da Tabela 1:

1. **Cenário A vs C**: Ambos sem perdas, mas A sem CC atinge 1024 enquanto C com CC fica limitado a 128. A diferença mostra o overhead do controle de congestionamento em redes ideais.
2. **Cenário B vs D**: Ambos com perdas de 10%, mas D com CC adapta-se às perdas (média 13.27) enquanto B mantém janela fixa (média 63.97). Sem CC, pacotes são desperdiçados.
3. **Desvio Padrão**: O Cenário D tem maior variabilidade (12.45) devido às oscilações adaptativas. O Cenário C tem baixa variabilidade (4.15) pois cresce e estabiliza.
4. **Eficiência**: O Cenário D, embora com menor média, é mais eficiente em redes com perdas pois não desperdiça recursos retransmitindo pacotes desnecessariamente.

5. Requisitos Atendidos

5.1. Requisito 1: Entrega Ordenada

Status: Implementado

- Localização: `servidor.py`, linhas 23-29
- Descrição: Usa buffer de recepção para ordenar pacotes que chegam fora de ordem

5.2. Requisito 2: Confirmação Acumulativa

Status: Implementado

- Localização: `servidor.py`, linhas 38-45
- Descrição: ACK confirma o número de sequência mais alto recebido em ordem

5.3. Requisito 3: Controle de Fluxo

Status: Implementado

- Localização: `cliente.py`, linhas 65 e `servidor.py`, linha 40
- Descrição: Janela do receptor comunicada ao cliente via campo window

5.4. Requisito 4: Controle de Congestionamento

Status: Implementado

- Localização: `cliente.py`, linhas 60-88
- Descrição: Implementação de Slow Start, Congestion Avoidance e recuperação
- Variáveis: cwnd (congestion window) e ssthresh (slow start threshold)

5.5. Requisito 5: Criptografia

Status: Implementado

- Localização: `protocolo.py`, linhas 5-10 e 12-17
- Descrição: Criptografia XOR com chave compartilhada
- Handshake: `cliente.py`, linhas 32-45

5.6. Requisito 6: Avaliação do Protocolo

Status: Implementado

- 10.000+ pacotes: Transmitidos em todos os cenários
- Perdas arbitrárias: Simuladas com taxa aleatória no servidor
- Múltiplos cenários: Quatro cenários avaliados e documentados

6. Conclusões

Este trabalho demonstrou com sucesso a implementação de um protocolo confiável de transporte sobre UDP, integrando todos os requisitos especificados:

1. A **entrega ordenada** foi garantida através de números de sequência e buffering
2. A **confirmação acumulativa** simplificou a lógica de reconhecimento
3. O **controle de fluxo** baseado em janela evita sobrecarga do receptor
4. O **controle de congestionamento** adapta dinamicamente à rede
5. A **criptografia** protege os dados em trânsito

Os resultados experimentais mostram que: - Em redes perfeitas (Cenário A), um protocolo simples funciona bem - Em redes com perdas (Cenário B), sem adaptação resulta em desperdício de recursos - Com controle de congestionamento sem perdas (Cenário C), o crescimento é controlado e estável - Com controle de congestionamento e perdas (Cenário D), o protocolo se adapta eficientemente

O **diferencial crítico** é que o Cenário D demonstra comportamento inteligente: começa conservador, cresce agressivamente quando possível, e recua quando detecta problemas. Esta adaptação é a base de protocolos de sucesso como TCP e QUIC.

7. Trabalhos Futuros

Possíveis melhorias e extensões:

- Implementar **ACK seletivo** em vez de apenas acumulativo
- Usar **criptografia robusta** (AES, ChaCha20) em produção
- Implementar **fast retransmit** com detecção de ACKs duplicados
- Adicionar **estimação de RTT** e timeout dinâmico
- Implementar **multiplicative increase, multiplicative decrease** (MIMD)
- Avaliar com **latências variáveis** e em redes reais

Referências

[1] KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a Internet: Uma Abordagem Top-Down**. 7. ed. [S.l.]: Pearson, 2017. (Capítulos 1-4).

[2] TANENBAUM, A. S. **Computer Networks**. 5. ed. [S.l.]: Pearson, 2010.

[3] CEDERJ. **Slides, listas de exercícios e vídeos do curso de Redes de Computadores**. Fundação CECIERJ, 2024.