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 three different scenarios: (A) without losses and without congestion control, (B) with losses and without congestion control, and (C) 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 três cenários distintos: (A) sem perdas e sem controle de congestionamento, (B) com perdas e sem controle de congestionamento, e (C) 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: 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.

4. Resultados e Análise

4.1. Gráficos Comparativos

A seguir são apresentados os gráficos comparativos dos três 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.

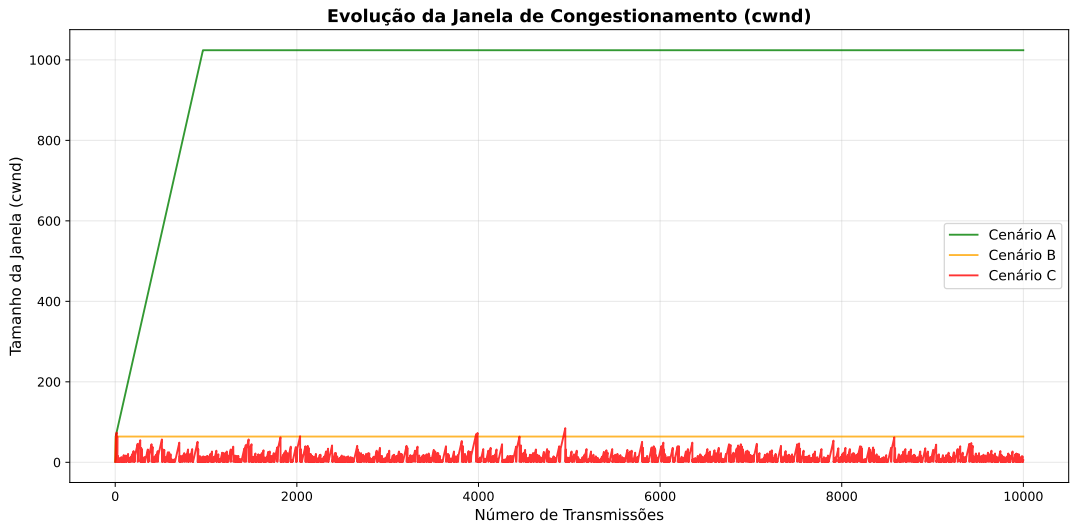

Figura 1. Evolução da Janela de Congestionamento (cwnd) ao longo das transmissões. O Cenário A (verde) apresenta crescimento contínuo sem perdas. O Cenário B (laranja) mantém janela alta mesmo com perdas. O Cenário C (vermelho) demonstra comportamento adaptativo com fases de Slow Start e Congestion Avoidance.

4.1.2. Comparação de Vazão Média

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

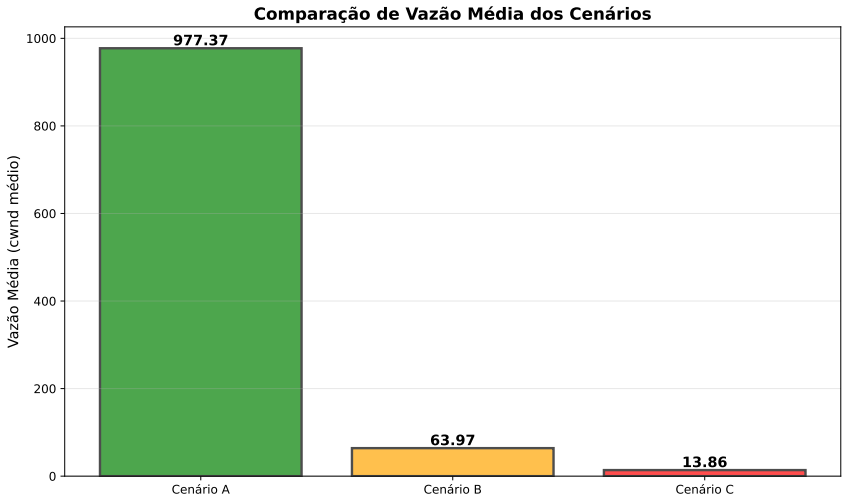

Figura 2. Comparação de Vazão Média dos Cenários. O Cenário A apresenta maior vazão média (977.37) devido à ausência de perdas e crescimento contínuo até o máximo. O Cenário B mantém janela fixa em 64. O Cenário C apresenta menor média (13.86) devido às reduções adaptativas após detecção de perdas.

4.1.3. Tabela de Métricas Estatísticas

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

Estatísticas dos Cenários

Cenário	Pacotes	Média	Máximo	Mínimo	Desvio Pad.	Mediana
Cenário A	10000	977.37	1024	2	166.96	1024.00
Cenário B	10000	63.97	64	2	1.18	64.00
Cenário C	10000	13.86	85	1	12.63	10.00

Figura 3. Tabela de Métricas Estatísticas dos Três Cenários, incluindo número de pacotes transmitidos, média, máximo, mínimo, desvio padrão e mediana 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 Efetivo

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

1. **Evolução dinâmica de cwnd**: A janela oscila entre 1 e 85, demonstrando ciclos de Slow Start e recuperação
2. **Média reduzida**: 13.86 pacotes, 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**: 12.63, indicando variabilidade controlada

A estratégia adaptativa do Cenário C 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 Três Cenários

Métrica	Cenário A	Cenário B	Cenário C
Pacotes Transmitidos	10.000	10.000	10.000
Média de cwnd	977.37	63.97	13.86
Máximo de cwnd	1024	64	85
Mínimo de cwnd	2	2	1
Desvio Padrão	166.96	1.18	12.63
Mediana de cwnd	1024	64	10

Observações da Tabela 1:

1. **Média de cwnd**: O Cenário A apresenta média muito alta (977.37), pois a janela cresce continuamente até o máximo. O Cenário B mantém média estável em 63.97 (janela fixa). O Cenário C tem menor média (13.86) devido às reduções adaptativas após detectar perdas.
2. **Desvio Padrão**: O Cenário A apresenta desvio padrão alto (166.96) devido ao crescimento inicial. O Cenário B tem desvio mínimo (1.18), indicando comportamento constante. O Cenário C tem desvio moderado (12.63), refletindo as oscilações do controle de congestionamento.
3. **Máximo de cwnd**: O Cenário A atinge o máximo de 1024. O Cenário B fica limitado a 64. O Cenário C atinge até 85 antes de detectar perdas e reduzir.
4. **Mínimo de cwnd**: O Cenário C desce até 1 (Reset em Slow Start após timeout), enquanto A e B não descem abaixo de 2, confirmando a estratégia reativa do Cenário C.

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: Três 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 vazão reduzida - Com controle de congestionamento (Cenário C), o protocolo se adapta eficientemente

O **diferencial crítico** é que o Cenário C 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.