



**Universidade do Minho**  
Escola de Engenharia  
Licenciatura em Engenharia Informática

## **Comunicações por Computador**

### **Trabalho Prático 2**

Ano Letivo de 2024/2025

## **Network Monitoring System**

Flávia Alexandra da Silva Araújo (A96587)  
Joshua David Amaral Moreira (A105684)  
Miguel Torres Carvalho (A95485)

28 de novembro de 2024

CC

# Resumo

No presente relatório, é apresentada a solução desenvolvida para a Unidade Curricular de **Comunicações por Computador**, como parte do projeto final do semestre - **Network Monitoring System**. Este trabalho teve como objetivo projetar um sistema capaz de monitorizar e analisar o tráfego de rede entre um servidor central e vários agentes distribuídos, permitindo a recolha de métricas de desempenho e a execução de tarefas de monitorização.

O sistema foi desenvolvido utilizando dois protocolos aplicacionais principais: **AlertFlow**, destinado à gestão de alertas, e **NetTask**, que assegura a comunicação robusta e confiável entre o servidor e os agentes. Foram implementadas funcionalidades de tolerância a falhas e adaptação a condições adversas, como alta latência e perda de pacotes, garantindo a integridade dos dados transmitidos.

O trabalho foi concluído com a implementação e validação das principais funcionalidades do sistema, demonstrando a sua eficácia em ambientes de teste representativos. Este relatório documenta detalhadamente o *design*, a implementação e os resultados obtidos.

**Palavras-Chave:** *Network Monitoring System, AlertFlow, NetTask, Comunicações por Computador.*

# Índice

<b>1</b>	<b>Arquitetura da Solução</b>	<b>1</b>
<b>2</b>	<b>Especificações dos Protocolos Aplicacionais</b>	<b>2</b>
2.1	<i>AlertFlow</i>	2
2.1.1	Formato de Mensagem	2
2.1.2	Diagrama de Sequência	2
2.2	<i>NetTask</i>	3
2.2.1	Formato de Cabeçalho e Descrição de Campos	3
2.2.2	Descrição de Funcionalidades	4
2.2.3	Diagramas de Sequência	6
<b>3</b>	<b>Implementação</b>	<b>7</b>
3.1	Estrutura do Projeto?	7
3.2	Parâmetros dos Executáveis	7
3.3	Ficheiro de Configuração	7
3.4	Bibliotecas Utilizadas	7
3.5	Detalhes Técnicos? Maybe	7
<b>4</b>	<b>Testes e Resultados</b>	<b>8</b>
<b>5</b>	<b>Conclusões e Trabalho Futuro</b>	<b>9</b>
	<b>Bibliografia</b>	<b>10</b>
	<b>Anexos</b>	<b>11</b>
	[I] Exemplo de Anexo	11

# Índice de Figuras

2.1	Formato do cabeçalho do protocolo <i>NetTask</i> . . . . .	3
-----	--	---

# Índice de Tabelas

## Índice de *Snippets*

# 1 Arquitetura da Solução

Dúvida: o diagrama da arquitetura da solução deve ser mais geral ou mais específico à solução (python) desenvolvida?

## **2 Especificações dos Protocolos Aplicacionais**

### **2.1 *AlertFlow***

#### **2.1.1 Formato de Mensagem**

#### **2.1.2 Diagrama de Sequência**

- Normal



## 2.2 *NetTask*

O protocolo *NetTask* é essencial para a funcionalidade harmonizada do **Network Monitoring System**, sendo este usado para a maioria das comunicações entre o servidor e os agentes, tais como, a primeira conexão de um agente ao servidor, o envio de tarefas pelo servidor, o envio de resultados de tarefas pelos agentes, e a terminação de conexões nos dois sentidos.

Deste modo, o protocolo *NetTask* foi desenvolvido para ser robusto e adaptável a condições adversas de rede, garantindo a entrega fiável e integral de mensagens, sobretudo em rotas deterioradas, com perdas ou duplicação de pacotes, latências elevadas e taxas de débito variáveis.

Para combater tais adversidades, o protocolo aplicacional *NetTask* responsabiliza-se pelas funcionalidades que serão exploradas no subcapítulo Descrição de Funcionalidades.

Nos subcapítulos seguintes, serão detalhadas as especificações do protocolo, nomeadamente o formato do cabeçalho, descrição de funcionalidades e diagramas de sequência que ilustram o comportamento do protocolo em situações normais e adversas.

### 2.2.1 Formato de Cabeçalho e Descrição de Campos

NMS NetTask Version (1 byte)	Sequence Number (2 bytes)	Flags (5 bits)	Type (3 bits)
Window Size (2 bytes)		Checksum (2 bytes)	
Message ID (2 bytes)			
Agent Identifier (32 bytes)			
Data			

Figura 2.1: Formato do cabeçalho do protocolo *NetTask*

- **NMS *NetTask* Version** (1 byte): versão do protocolo, para a compatibilidade de versões entre o servidor e os agentes;
- **Sequence Number** (2 bytes): número de sequência da mensagem, para a ordenação de pacotes, deteção de pacotes duplicados e identificação de *acknowledgements*;

- **Flags** (5 bits): *flags* de controlo:
  - ACK** (1º bit): *Acknowledgement*, utilizado para confirmar a receção de pacotes;
  - RET** (2º bit): *Retransmission*, indica que o pacote é uma retransmissão;
  - URG** (3º bit): *Urgent*, indica que a mensagem é urgente;
  - WP** (4º bit): *Window Probe*, utilizado para controlo de fluxo;
  - MF** (5º bit): *More Fragments*, para (des)fragmentação de pacotes.
- **Type** (3 bits): tipo da mensagem:
  - **0 - Undefined** : mensagem indefinida, utilizada para testes ou quando nenhum tipo de mensagem é aplicável, por exemplo no envio de *window probes*;
  - **1 - First Connection** : primeira conexão de um agente ao servidor;
  - **2 - Send Task** : envio de tarefas pelo servidor;
  - **3 - Send Metrics** : envio de resultados de tarefas pelos agentes;
  - **4 - EOC (End of Connection)** : terminação de conexões nos dois sentidos;
  - **\* - Reserved** : reservado para futuras extensões. (códigos de 5 a 7);
- **Window Size** (2 bytes): indica o tamanho da janela de receção, para controlo de fluxo;
- **Checksum** (2 bytes): soma de verificação da mensagem, para deteção de erros;
- **Message Identifier** (2 bytes): identificador da mensagem, utilizado para a desfragmentação e ordenação de pacotes;
- **Agent Identifier** (32 bytes): identificador do agente, podendo este ser do recetor ou do emissor da mensagem;
- **Data/Payload** (n bytes): carga útil da mensagem com tamanho variável, contendo a informação a ser transmitida nas mensagens do tipo *Send Task* e *Send Metrics*.

## 2.2.2 Descrição de Funcionalidades

### Fragmentação de Pacotes

A fragmentação de pacotes é efetuada quando a carga útil da mensagem excede o tamanho máximo permitido, sendo esta dividida em fragmentos de forma a não exceder

este limite. Cada fragmento é encapsulado num pacote *NetTask* com a *flag MF* ativado, indicando que existem mais fragmentos a serem enviados. Para permitir a identificação e ordenação dos fragmentos, é atribuído um *Message Identifier* único a cada mensagem fragmentada.

### **Retransmissão de Pacotes Perdidos**

A retransmissão de pacotes perdidos é efetuada quando o emissor não recebe um *acknowledgement* de um pacote enviado, após um determinado intervalo de tempo. O emissor reenvia o pacote com a *flag RET* ativada, indicando que se trata de uma retransmissão. O recetor, ao receber um pacote com esta *flag*, descarta o pacote se este já tiver sido recebido anteriormente.

### **Deteção de Erros**

A deteção de erros é efetuada através da soma de verificação da mensagem (*Checksum*), calculada com base no cabeçalho e na carga útil da mensagem. O recetor, ao receber um pacote, calcula a soma de verificação e compara-a com a recebida. Se a soma de verificação calculada for diferente da recebida, o pacote é descartado.

### **Ordenação de Pacotes**

A ordenação de pacotes é efetuada com base no *Sequence Number* da mensagem, permitindo a reordenação de pacotes que chegam fora de ordem. O recetor mantém um registo dos pacotes recebidos e descarta pacotes duplicados, reordenando os pacotes recebidos de forma a garantir a entrega integral e ordenada das mensagens.

### **Deteção e Manuseamento de Pacotes Duplicados**

A deteção e manuseamento de pacotes duplicados, tal como na ordenação de pacotes, recorre ao *Sequence Number* da mensagem. O recetor mantém um registo dos pacotes recebidos para deteção de pacotes duplicados, descartando pacotes que já tenham sido recebidos anteriormente, e enviando um *acknowledgement* ao emissor para confirmar a receção do pacote para evitar retransmissões desnecessárias.

### **Controlo de Fluxo**

O controlo de fluxo é efetuado através do tamanho da janela de receção (*Window Size*), indicando ao emissor o número de pacotes que o recetor pode receber sem congestionar a ligação. O emissor ajusta o tamanho da janela de envio com base no tamanho da janela de receção do recetor, evitando a sobrecarga da ligação e a perda de pacotes. O recetor, ao receber um pacote, envia um *acknowledgement* com o tamanho da janela de receção atualizado, permitindo ao emissor ajustar o tamanho da janela de envio.

### **Compatibilidade de Versões**

A compatibilidade de versões é garantida através do campo *NMS NetTask Version* do ca-

beçalho da mensagem, permitindo a identificação da versão do protocolo entre o servidor e os agentes. Se a versão do protocolo do emissor for diferente da versão do recetor, a mensagem é descartada, evitando possíveis incompatibilidades e erros.

### **2.2.3 Diagramas de Sequência**

- First Connection
- End of Connection
- Retransmission
- Desfragmentação e Ordenação de pacotes
- Controlo de Fluxo
- Deteção de Erros
- Deteção e Manuseamento de Pacotes Duplicados

## 3 Implementação

### 3.1 Estrutura do Projeto?

### 3.2 Parâmetros dos Executáveis

### 3.3 Ficheiro de Configuração

### 3.4 Bibliotecas Utilizadas

Dúvida: deve-se referir todas as bibliotecas utilizadas no desenvolvimento?  
por exemplo: *socket*, *os*, *sys*, *threading*, etc Ou apenas as bibliotecas mais relevantes para a solução? *ping3*, *iperf*, etc

### 3.5 Detalhes Técnicos? Maybe

## 4 Testes e Resultados

Para além da testagem manual contínua ao longo do desenvolvimento, foram desenvolvidos testes unitários para as classes e funções mais críticas do sistema. Estes testes foram desenvolvidos com recurso à biblioteca *unittest* do Python.

# 5 Conclusões e Trabalho Futuro

## 5.1 Conclusões

A realização deste projeto permitiu a aplicação prática dos conhecimentos adquiridos durante o semestre na Unidade Curricular de **Comunicações por Computador**, nomeadamente no que diz respeito à conceção e implementação de protocolos aplicacionais para a comunicação entre sistemas.

A implementação do **Network Monitoring System** permitiu a integração de vários conceitos e técnicas de comunicação, como a fragmentação de pacotes, retransmissão de pacotes perdidos, controlo de fluxo, deteção de erros e ordenação de pacotes, garantindo a fiabilidade e robustez do sistema.

A utilização de protocolos aplicacionais personalizados, como o *AlertFlow* e o *NetTask*, permitiu a implementação de funcionalidades específicas para a monitorização e análise de tráfego de rede, adaptadas às necessidades do sistema.

Por conseguinte, o sistema desenvolvido demonstrou ser eficaz na monitorização e análise de tráfego de rede, permitindo a recolha de métricas de desempenho e a execução de tarefas de monitorização de forma eficiente e fiável.

## 5.2 Trabalho Futuro

TODO

# Referências

- [1] Connolly, T., & Begg, C. (2015). Database Systems: Example (6th ed.). Pearson Education. London, UK.



# Anexos

[I] Exemplo de Anexo