Trabalho Prático Nº2 – Monitorização Distribuída de Redes

Duração: 8 aulas

Introdução

As redes de computadores são fundamentais para a comunicação entre sistemas em ambientes modernos, sendo crucial assegurar a sua operação eficiente e contínua. A monitorização das redes permite identificar e solucionar problemas antes que causem grandes transtornos, garantindo a continuidade dos negócios e a satisfação dos utilizadores.

Este trabalho tem como objetivo o desenvolvimento de um sistema de gestão de redes (*Network Monitoring System* – NMS) capaz de fornecer informação detalhada sobre o estado dos links e dos dispositivos numa rede, bem como gerar alertas caso sejam detetadas anomalias. Este sistema deverá ser desenvolvido como uma aplicação distribuída com base num modelo cliente-servidor onde uma aplicação cliente (**NMS_Agent**) é responsável por recolher diferentes métricas de interesse e de as reportar a um servidor centralizado (NMS_Server), conforme ilustra a Figura 1. Dois protocolos aplicacionais deverão ser desenvolvidos para permitir uma eficiente troca de mensagens entre o NMS_Agent e o NSM_Server, nomeadamente:

- **NetTask** (utilizando **UDP**) para a comunicação de tarefas e a coleta contínua de métricas.
- **AlertFlow** (utilizando **TCP**) para notificação de alterações críticas no estado dos dispositivos de rede.

Por fim, este trabalho visa familiarizar os alunos com o desenvolvimento de protocolos aplicacionais e a utilização de **sockets** para comunicação em rede, explorando a resiliência e robustez de soluções distribuídas. O NMS proposto é levemente inspirado no protocolo "NetFlow". No entanto, é expectável que cada grupo proponha uma solução própria e inovadora.

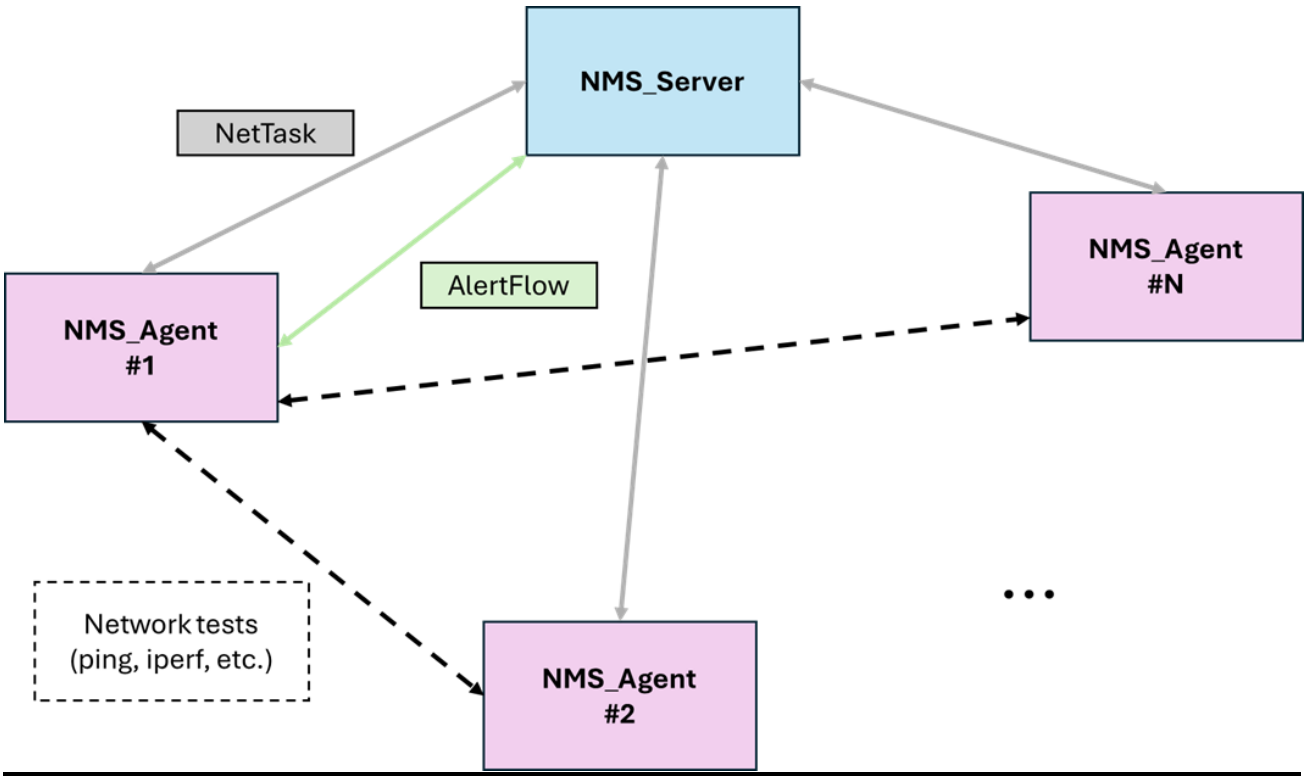

Figura 1 - Visão geral

Objetivos

- Coletar **métricas de rede** e apresentar resultados úteis para o **gestor de rede**.
- Desenvolver uma solução resiliente e eficiente para a **monitorização contínua** e **notificação de eventos críticos** na rede.
- Desenvolver **protocolos aplicacionais** usando **UDP** e **TCP** como camadas de transporte.
- Implementar a comunicação cliente-servidor utilizando **sockets TCP e UDP**.
- Compreender e implementar mecanismos de **controlo de fluxo (opcional)**, **números de sequência** e **retransmissão** em cima do protocolo UDP.

Descrição do Trabalho

O trabalho é dividido em duas componentes principais: **NMS_Server** e **NMS_Agent**.

NMS_Server

O servidor, Figura 2, será responsável por coordenar as atividades dos NMS_Agents registados e estará dividido nos seguintes requisitos funcionais:

1. **Interpretação de Tarefas**: Processar tarefas descritas num ficheiro JSON, que serão enviadas aos NMS_Agents. A tarefa especifica que métricas devem ser coletadas e com que frequência. Também identifica quais dados devem ser monitorizados.
2. **Apresentação de Métricas**: Oferecer uma interface para consultar as métricas recolhidas pelos NMS_Agents.

3. **Comunicação UDP (NetTask)**: Comunicar com os NMS_Agents através de **UDP**, enviando tarefas e recebendo as métricas coletadas.
4. **Comunicação TCP (AlertFlow)**: Receber notificações de alterações críticas nos dados monitorizados.
5. **Armazenamento de Dados**: Armazenar todas as métricas e informações recebidas dos NMS_Agents.

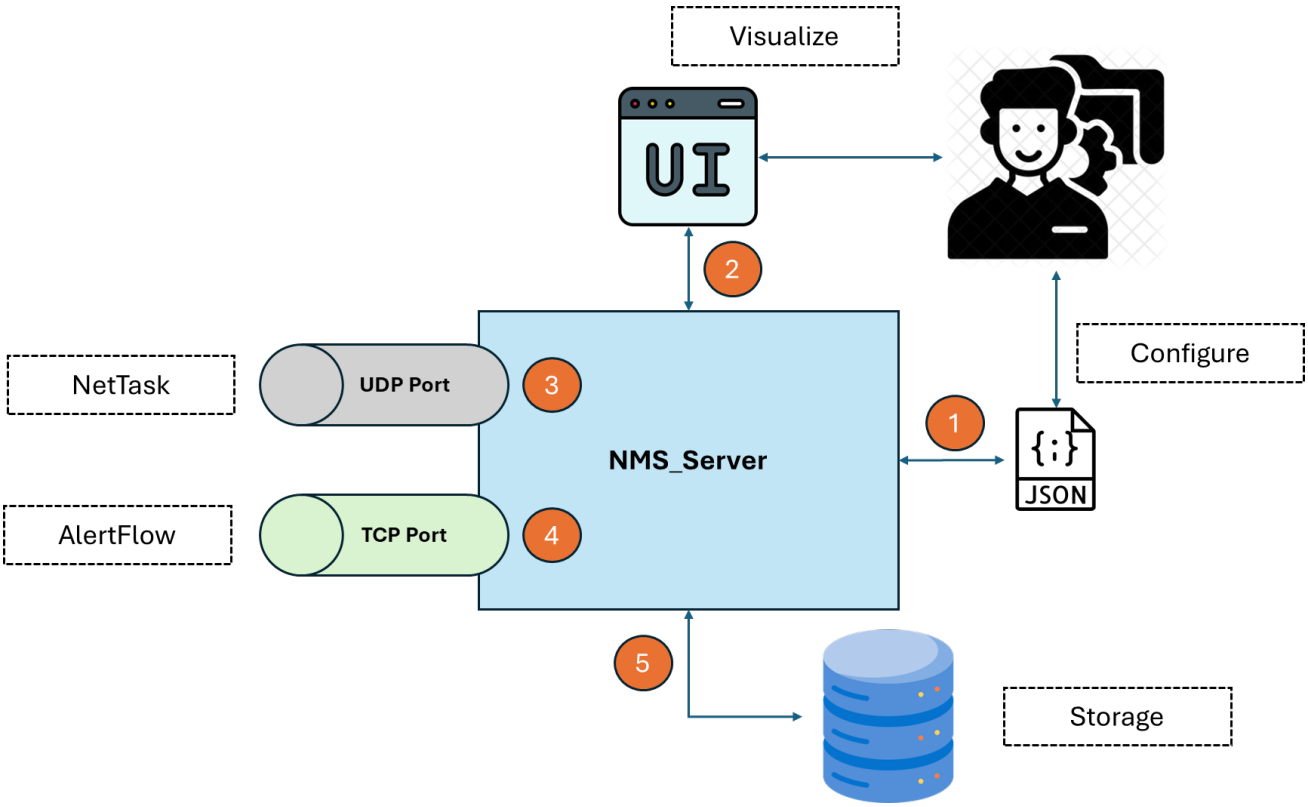

Figura 2 - NMS Server

NMS_Agent

O agente, Figura 3, será executado em dispositivos de rede (routers no emulador CORE) e será composto por três requisitos funcionais:

6. **Comunicação UDP (NetTask)**: Registar o NMS_Agent no NMS_Server, receber a tarefa, executar os testes de monitorização e reportar os resultados periodicamente.
7. **Comunicação TCP (AlertFlow)**: Enviar notificações ao NMS_Server sempre que houver alterações significativas nas métricas monitorizadas (por exemplo, falhas de interface ou variações críticas no desempenho).
8. **Execução de Primitivas de Sistema**: Executar comandos do sistema como:
    - ping para testar latência.
    - iperf para testes de largura de banda (cliente ou servidor).
    - Comandos de monitorização de interfaces de rede (usando o comando ip).

O NMS_Agent deve identificar todas as suas comunicações com o NMS_Server através de um **ID único**.

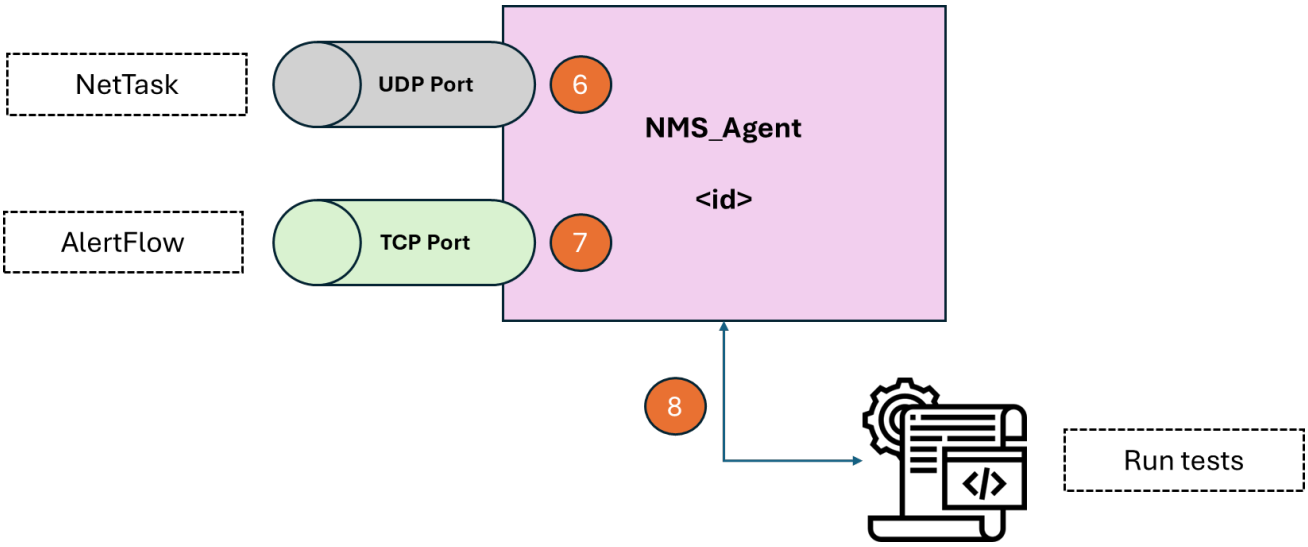

Figura 3 – NMS Agent

Protocolo NetTask

O **NetTask** é um protocolo aplicacional que usa a camada de transporte **UDP** e deve incorporar características normalmente associadas ao **TCP**, tais como:

- **Números de sequência e acknowledgment (ACK)**: Para garantir a entrega e a correta ordenação das mensagens.
- **Controlo de fluxo (opt)**: Para evitar a sobrecarga na comunicação.
- **Retransmissão**: Em caso de perda de pacotes na rede.

Os alunos devem implementar estes mecanismos para assegurar uma comunicação eficiente e resiliente, mesmo quando a rede está sujeita a falhas ou degradação.

Protocolo AlertFlow

O protocolo **AlertFlow** será responsável pela comunicação de alterações críticas nas métricas monitorizadas, utilizando a camada de transporte **TCP** para garantir a entrega confiável das mensagens.

Sempre que uma mudança significativa ocorrer nos valores das métricas (como falhas numa interface de rede ou variações nos parâmetros de latência), o NMS_Agent deve enviar uma notificação ao NMS_Server via TCP, indicando a alteração.

Tarefa

O NMS_Server recebe um ficheiro JSON que descreve as métricas a serem coletadas para cada dispositivo, bem como os limites que, ao serem ultrapassados, desencadeiam a comunicação via AlertFlow. É fundamental que

o NMS_Server interprete corretamente o arquivo, de modo a atribuir a tarefa adequada a cada NMS_Agent, garantindo assim uma monitorização eficiente e a resposta imediata a eventuais problemas.

A estrutura abaixo serve como exemplo ilustrativo de ficheiro de configuração:

Task:

- **task_id**: Um identificador único para a tarefa (por exemplo, "task-202").
- **frequency**: A frequência em que os dados devem ser coletados (por exemplo, 20 segundos).
- **devices**: Uma lista de dispositivos (routers) que serão monitorizados. Cada dispositivo possui uma série de propriedades detalhadas.

Devices:

- Cada dispositivo tem um **device_id** (por exemplo, "r1", "r2", etc.).
- **device_metrics**: Informações sobre o uso de recursos do dispositivo:
    - **cpu_usage**: Indica se a monitorização do CPU está ativada.
    - **ram_usage**: Indica se a monitorização da RAM está ativada.
    - **interface_stats**: Uma lista simplificada de nomes das interfaces de rede (por exemplo, ["eth0", "eth1", "eth2"]).
- **link_metrics**: Métricas específicas para a comunicação entre dispositivos:
    - **bandwidth**: Configurações para medir a largura de banda, incluindo o uso de uma ferramenta (iperf), se é um cliente ou servidor, o endereço do servidor, duração do teste, tipo de transporte e frequência.
        - **jitter**: Configurações para medir o jitter (variação na latência), usando também o iperf com opções semelhantes.
        - **packet_loss**: Configurações para medir a perda de pacotes, igualmente utilizando o iperf.
        - **latency**: Configurações para medir a latência usando ping, especificando o destino, contagem de pacotes e frequência.
    - **alertflow_conditions**: Condições para disparo de alertas baseadas em métricas:
        - **cpu_usage**: Limite de uso da CPU (Ex.: 80%).
        - **ram_usage**: Limite de uso da RAM (Ex.: 90%).
        - **interface_stats**: Limite de pacotes por segundo para as interfaces (Ex.: 2000 pps).
        - **packet_loss**: Limite de perda de pacotes (Ex.: 5%).
        - **jitter**: Limite para jitter (Ex.: 100 ms).

Exemplos de comunicação entre NMS_Agents eNMS_Server

Registo do NMS_Agent (NetTask)

- **NMS_Agent:** Envia um pedido de registo ao NMS_Server.
- **NMS_Server:** Recebe e confirma o registo do NMS_Agent.

Solicitação de Tarefas (NetTask)

- **NMS_Server:** Envia uma tarefa ao NMS_Agent para coletar métricas.

- **NMS_Agent:** Recebe a tarefa e confirma a execução.

Coleta de Métricas (NetTask)

- **NMS_Agent:** Coleta as métricas periodicamente e as envia ao NMS_Server.
- **NMS_Server:** Confirma o recebimento das métricas.

Notificação de Alerta (AlertFlow)

- **NMS_Agent:** Detecta uma alteração crítica e envia uma notificação ao NMS_Server.
- **NMS_Server:** Recebe a notificação de alerta.

Cenário de teste

Deve-se usar como plataforma de teste o emulador CORE e a topologia CC-Topo-2024.imn. Para o cenário de teste mínimo é necessário arrancar o servidor NMS_Server e três ou mais NMS_Agents em diferentes localizações. As localizações devem ser escolhidas de modo a incluir ligações com perdas. O primeiro passo é executar o NMS_Server, em seguida, iniciam-se várias instâncias do NMS_Agent.

A demonstração deverá conter o registo do NMS_Agent, a interface para o gestor da rede no NMS_Server, a possibilidade de carregar um ficheiro de configuração JSON e a observação dos testes configurados, envio dos relatórios periódicos via NetTask e a ocorrência de um evento que dê início a uma notificação via AlertFlow.

Requisitos Técnicos

- O projeto deve ser implementado na linguagem selecionada pelo grupo de trabalho.
- O emulador de redes **CORE 7.5** será utilizado para simular o ambiente de rede.
- O ficheiro de tarefas deve ser fornecido no formato **JSON**.
- Cada NMS_Agent deve possuir um **ID único** que identifique todas as suas comunicações.
- O sistema deve ser capaz de suportar múltiplos NMS_Agents a reportar métricas em simultâneo para um único NMS_Server.

Planeamento semanal sugerido

Semana 1: Estudo e Design Inicial

- Estudo detalhado do enunciado e requisitos técnicos.
- Design dos protocolos NetTask (UDP) e AlertFlow (TCP), incluindo formatos de mensagem e mecanismos de controlo.
- Decisão da linguagem de programação a ser utilizada para o desenvolvimento.

Semana 2: Configuração do Ambiente e Desenvolvimento Inicial

- Configuração do emulador CORE 7.5 e do ambiente de desenvolvimento.
- Desenvolvimento da comunicação básica via UDP e TCP entre NMS_Agent e NMS_Server.
- Implementação inicial do módulo de interpretação de tarefas no NMS_Server (capaz de ler e processar arquivos JSON).

Semana 3: Desenvolvimento do Módulo de Parsing do NMS_Server

- Desenvolvimento do módulo de parsing no NMS_Server para interpretar e atribuir corretamente as tarefas baseadas no arquivo JSON.
- Testes iniciais para garantir que as tarefas sejam corretamente distribuídas aos NMS_Agents.

Semana 4: Implementação do Protocolo NetTask

- Desenvolvimento do módulo de comunicação UDP no NMS_Agent.
- Implementação da lógica para registar NMS_Agents no NMS_Server.
- Envio e receção de tarefas via NetTask, com início da coleta de métricas.

Semana 5: Coleta de Métricas

- Implementação das primitivas de sistema (ping, iperf) no NMS_Agent.
- Desenvolvimento do módulo de coleta de métricas e envio periódico ao NMS_Server.
- Testes iniciais de comunicação e coleta de dados.

Semana 6: Implementação do Protocolo AlertFlow

- Desenvolvimento do módulo de comunicação TCP para notificações de alerta no NMS_Agent.
- Implementação do tratamento de condições críticas no NMS_Agent (alertas disparados quando os limites são ultrapassados).
- Configuração e testes do módulo de comunicação TCP no NMS_Server.

Semana 7: Desenvolvimento do Módulo de Apresentação de Métricas

- Implementação do módulo de apresentação de métricas no NMS_Server para o gestor de redes.
- Desenvolvimento de uma interface para visualizar as métricas recolhidas de forma clara e estruturada.
- Integração do módulo de apresentação com o módulo de armazenamento de dados e testes de usabilidade.

Semana 8: Finalização

- Validação da qualidade dos testes realizados, incluindo verificação de resiliência e desempenho.
- Escrita do relatório técnico detalhado, cobrindo a implementação, decisões de design, e os testes.
- Revisão e ajustes finais no código e no relatório.
- Submissão do trabalho completo.

Avaliação

A avaliação será baseada nos seguintes critérios:

- **Viabilidade da implementação dos protocolos aplicacionais** (NetTask e AlertFlow).
- **Resiliência** e capacidade de lidar com falhas de rede no protocolo NetTask.
- **Implementação correta** dos mecanismos de controle de fluxo e números de sequência para o protocolo NetTask.
- **Funcionalidade completa** do sistema de monitorização, incluindo a execução correta das primitivas de sistema (ping, iperf, etc.).

- **Qualidade das métricas** apresentadas ao gestor de rede.
- **Estrutura e organização** do código, incluindo modularidade e clareza.
- **Documentação** do projeto e explicação detalhada das soluções implementadas.
- **Demonstração.**

Entregáveis

Os alunos devem submeter:

- O código-fonte completo da solução (NMS_Agent e NMS_Server).
- Um relatório técnico descrevendo a implementação, as decisões de design e os testes realizados.

O relatório deve ser escrito em formato de artigo com um máximo de 10 páginas (recomenda-se o uso do formato LNCS - Lecture Notes in Computer Science, instruções para autores em http://www.springer.com/computer/lncs?SGWID=0-164-6-793341-0). Deve descrever o essencial do desenho e implementação com a seguinte estrutura recomendada:

- Introdução
- Arquitetura da solução
- Especificação do(s) protocolo(s) propostos
    a. formato das mensagens protocolares
    b. diagrama de sequência data troca de mensagens
- Implementação
    a. detalhes, parâmetros, bibliotecas de funções, etc.
- Testes e resultados
- Conclusões e trabalho futuro

Prazos

- **Dia de entrega**: semana de 02 a 06 de dezembro de 2024 (data do turno).
- **Semana de Demonstração:** semana de 06 a 10 de janeiro de 2025.

Conclusão

Este trabalho prático desafia os alunos a desenvolver uma solução distribuída de monitorização de redes, utilizando técnicas avançadas de programação de sockets e criação de protocolos aplicacionais. Ao desenvolver o **NetTask** e o **AlertFlow**, os alunos serão expostos a conceitos essenciais de comunicações por computador, lidando com a resiliência e confiabilidade de redes.