MÓDULO 4:

FILTRAGEM DE PARÂMETROS E EVENTOS EM TEMPO REAL

Documento TR-01
Versão 1.0 de abril de 2024
Projeto da disciplina STR
www.feelt.ufu.br

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Ádrian Marcello Mendes Dias
11911ECP009
Aluno de Engenharia de Computação, UFU

1. DATA VERSÃO ORIGINAL	2. DATA ÚLTIMA ATUALIZAÇÃO	3. DATA COBERTA
13-04-2024	23-04-2024	ABR/24
4. TÍTULO DESTE DOCUMENTO FILTRAGEM DE PARÂMETROS E EVENTOS EM TEMPO REAL		**5a. PROCESSO SEI DO P&D** **5b. NÚMERO PROJETO P&D**
6. AUTOR(ES) ÁDRIAN MARCELLO MENDES DIAS		**5c. ETAPA DO PROJETO** TODAS **5d. TIPO DE PRODUTO** DOCUMENTAÇÃO TÉCNICA DE SOFTWARE DE DISCIPLINA STR
7. ENDEREÇO AV. JOÃO NAVES DE ÁVILA, 2121, BLOCO 3N – UBERLÂNDIA – MG		**8. NÚMERO DO DOCUMENTO** **TR-01**
9. DISTRIBUIÇÃO DESTE DOCUMENTO DISTRIBUIÇÃO ABERTA A TODOS OS INTERESSADOS.		
10. NOTAS COMPLEMENTARES -		
11. RESUMO ESTE DOCUMENTO DESCREVE A MODELAGEM DOS ELEMENTOS DA APLICAÇÃO MU QUE GERA PACOTES DE DADOS PARA OUTROS ELEMENTOS DE SOFTWARE RETRATADO POR OUTROS REPORTES TÉCNICOS. TODOS MÓDULOS CONSTITUEM UM SISTEMA SUPERVISÓRIO PARA O SETOR ELÉTRICO.		
12. PALAVRAS-CHAVE P&D; IOT; SISTEMA EM TEMPO REAL, MERGE UNIT, SISTEMA SUPERVISÓRIO, MEDIÇÃO EM SUBESTAÇÕES DE ENERGIA.		
13. CLASSIFICAÇÃO SEGURANÇA: **ABERTA**	**14. NÚMERO DE PÁGINAS** -	**15. NOME DO RESPONSÁVEL PRINCIPAL E CONTATO** ÁDRIAN MARCELLO MENDES DIAS e-mail: adrian.marcello@ufu.br

HISTÓRICO DE VERSÕES DESTE TR

Tabela 1 – *Histórico de versões deste reporte técnico.*

Versão	Data	Modificações
0.1	abril/2024	• Descrição inicial da solução
1.0	abril/2024	• Visão geral da solução • Requerimentos básicos • Modelagem de pacotes e fluxo de pacotes • Interfaces básicas

SUMÁRIO

RESUMO GERAL

Este relatório técnico aborda os elementos que constituem o software de filtragem de parâmetros em tempo real, um módulo do sistema de monitoramento de plantas elétricas proposto pelo professor. Em resumo, o módulo tem a função de monitorar pacotes transmitidos na rede pelo módulo 1, indicando ao usuário a ocorrência de eventos e enviando um pacote de relatório para o módulo 3 (módulo de monitoramento). O usuário deve inserir regras, que fornecerão os parâmetros que o módulo precisa para notificar um evento, como qual corrente. Durante esse processo, o módulo identifica e contabiliza o número de eventos ocorridos por dispositivo de medição, também chamados de MU (*measurement unit)*. Essa documentação descreve majoritariamente o módulo 4 do sistema de monitoramento de plantas elétricas, não tratando com profundidade outros componentes do sistema.

1 – Introdução: visão geral da solução

1.1 – Propósito e escopo

O Módulo de Filtragem de Parâmetros e Eventos em Tempo Real é um componente do Sistema de monitoramento de plantas elétricas proposto pelo professor. Na figura abaixo é possível ter uma visão geral desta solução;

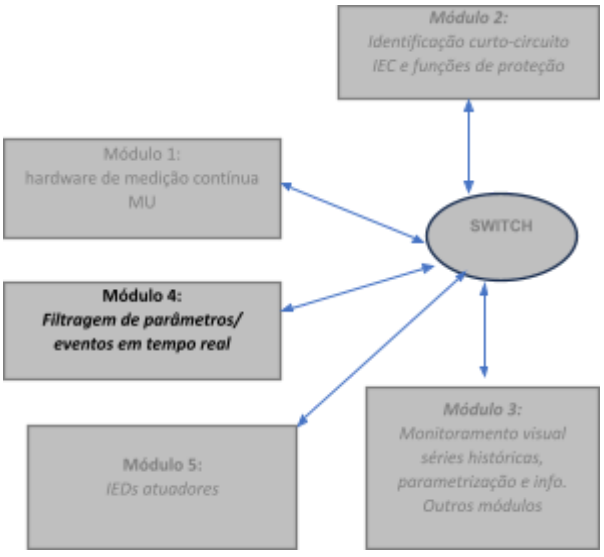

Figura 1.1.1: Visão geral do sistema de monitoramento de plantas elétricas.

Neste cenário, o módulo 4 atua dentro da subestação de energia, interagindo diretamente com outros módulos da rede.

O propósito deste software é permitir que o usuário insira regras personalizadas que serão usadas pelo módulo para monitorar em tempo real os dados de medição provindos do módulo 1, contabilizando um evento sempre que uma regra é satisfeita e, por fim, emitindo um relatório com identificação de número de eventos por dispositivo. Esse relatório será enviado para rede e será utilizado pelo módulo 3 para visualização e monitoramento do sistema.

Sendo assim, o escopo deste módulo abrange:

- Recebimento de pacotes de rede;
- Criação dinâmica de regras para identificação de evento;
- Processamento de pacotes recebidos;
- Emissão de relatório de eventos.

Como o módulo 1 consiste de várias de vários IED's (*inteligent electronic device*) que realizam medição, então este módulo recebe e processa pacotes com medidas de vários dispositivos diferentes.

A figura abaixo apresenta uma visualização simples da relação lógica direta entre o módulo 4, tema deste relatório, e os módulos 1 e 3: As medições sendo realizadas serão enviadas pelo módulo 1, recebidas e processadas no módulo 4, que enviará um relatório a ser capturado pelo módulo 3.

Figura 1.1.2: *Visualização simples da interação com outros módulos*

Importante notar que a arquitetura de comunicação não envolve comunicação direta, o pacote enviado pelo módulo 1 é distribuído em modo *broadcast* na rede, assim como o pacote relatório enviado pelo módulo 4.

1.2 – Produto: perspectivas e funções

A principal função deste módulo é identificar a ocorrência de eventos nas medidas realizadas pelo módulo 1, visando ter capacidade de receber e processar um número abundante de dados em quantidade de tempo satisfatório para o monitoramento da subestação. Os pacotes de medição oriundos do módulo 1 são processados com base em regras inseridas pelo usuário. Em seguida, um relatório de eventos é disponibilizado para o usuário e um pacote com as mesmas informações é enviado à rede, contendo a quantidade de eventos ocorridos no total e a quantidade de eventos ocorridos em cada um dos dispositivos que emitiu suas medidas na rede.

Para estes fins, é possível dividir as principais funções do módulo:

- Receber pacotes UDP oriundos do módulo 1;
- Criar regras parametrizáveis pelo usuário;
- Utilizar regras para analisar pacotes recebidos;
- Contabilizar quantidade de eventos ocorridos com base nas regras;
- Disponibilizar relatório de eventos para o usuário;
- Enviar pacote de relatório para a rede.

Essas funções podem ser resumidas em blocos da seguinte maneira:

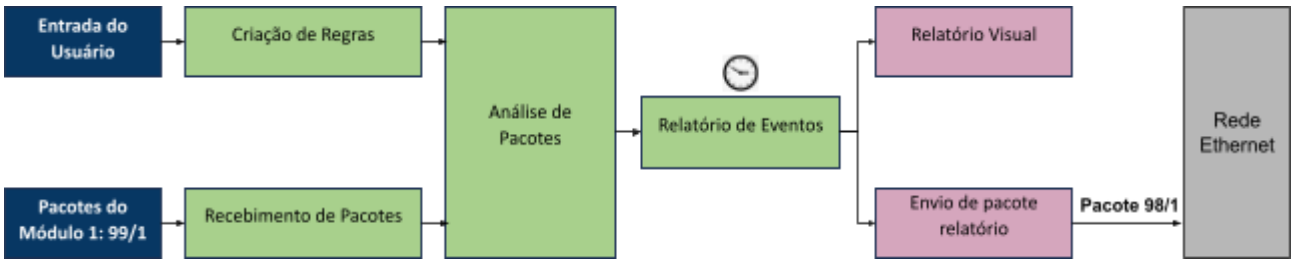

Figura 1.2.1: *Principais funções do sistema em verde. Entradas em azul. Saídas em rosa*

- **Recebimento de Pacotes:** Essa função é executada paralelamente durante todo o funcionamento do módulo, recebendo os pacotes UDP vindos da rede, transformando-os em objetos e colocando-os em *buffers*, onde serão buscados para análise;
- **Criação de Regras:** Essa função ocorre conforme as entradas do usuário. O usuário preenche campos de formulário com a corrente parâmetro a ser utilizada na regra, o operador de comparação e o valor numérico a ser comparado. Essas informações são então armazenadas em forma de objeto;

- **Análise de Pacotes:** Essa função é executada por cada objeto de regra individualmente e paralelamente. Seu dever é buscar os pacotes armazenados em seu *buffer* e analisá-los para descobrir se a regra é satisfeita, caso seja, essa função contabiliza que houve um evento naquele IED;
- **Relatório de Eventos:** É uma função periódica que a cada 0,5 segundos colhe as informações de eventos armazenados e gera dois produtos: um relatório em texto na interface visual do software, e um pacote JSON que é enviado para a rede.

1.3 – Restrições do produto e considerações

O produto descrito nesta documentação foi desenvolvido para atuar em condições específicas. Naturalmente, existem algumas restrições e limitações que serão informadas nesta seção.

Tabela 1.3.1: *Restrições e limitações previstas para o sistema.*

Nº	Restrição/limitação	Descrição/detalhamento
1	Opções Pré definidas de Parâmetros nas Regras Criadas	Este software foi criado especificamente para escutar pacotes gerados pelo módulo 1, logo as regras de detecção de eventos só podem ser criadas com os parâmetros que estão dentre as opções de variáveis produzidas pelo módulo 1
2	Limitação de recebimento da Rede em casos extremos	Em caso de curto na corrente, o módulo 1 envia uma grande quantidade de pacotes. Por utilizar comunicação UDP, essa grande quantidade de pacotes vindas de uma IED em curto pode “afogar” o recebimento de outros pacotes, atrasando a detecção de eventos em outras IED’s
3	Presença do sistema na mesma rede	Como os pacotes do módulo 1 são enviados para a rede no modo *broadcast*, é necessário que a máquina executando o sistema esteja na mesma rede, para o devido recebimento dos pacotes. Esse requisito também é necessário para que os pacotes relatórios sejam devidamente recebidos no módulo 3
4	Processamento de Grande Quantidades de Pacotes	Em casos de recebimento de um volume alarmante de dados, como por exemplo na detecção de um curto pelo módulo 1, a grande quantidade de pacotes pode afetar o desempenho do processo de *bufferização* dos pacotes

2 – Requisitos

2.1 – Cenários de uso

O sistema possui apenas um cenário de uso.

a) **Cenário padrão – operação em condições normais**: No cenário de funcionamento padrão, o sistema recebe pacotes da rede, analisa a ocorrência de eventos e emite

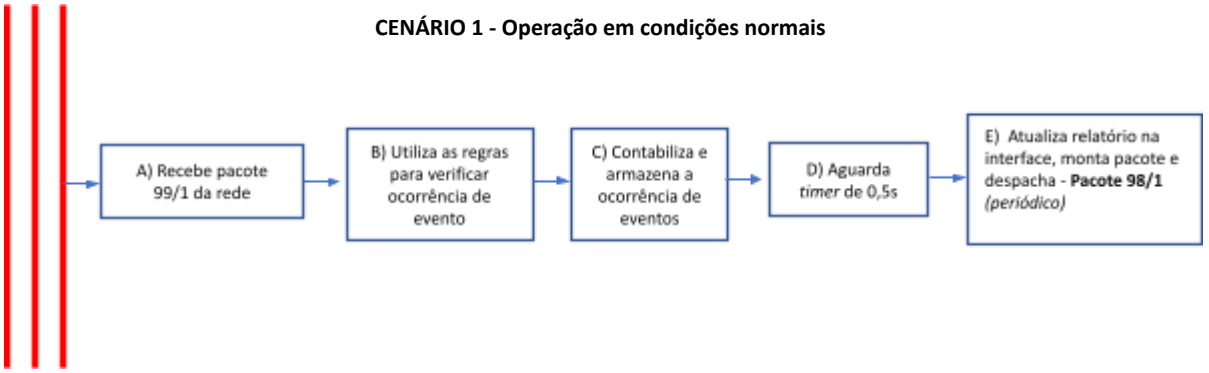

Figura 2.1.1: *Cenário padrão*

2.2 – Requisitos e validação

Com base na proposta do módulo 4, este software deve cumprir alguns requisitos, listados na tabela a seguir:

Tabela 2.2.1: *Mapa de requerimentos.*

Classe/ Componente	Nº req.	Requisito	Origem requisito	Prio	Tipo validação
1 - Robustez	1.1	Suportar o recebimento de um grande número de pacotes	Suportar alto número de amostragem vindas do módulo 1	1	Testar cenário padrão, utilizando mockup 1b com diversas unidades geradoras, incluindo casos de curto circuito. .
	1.2	Processar em tempo real os pacotes recebidos, atualizando o relatório em curto período de tempo		1	
2 - Funcional	2.1	Permitir que o usuário crie eventos dinâmicos	Especificação do módulo	1	Teste de inserção e remoção de regras dinamicamente antes e durante o recebimento de pacotes.
	2.2	Contabilizar a quantidade de eventos total e por IED individualmente		1	Teste do cenário padrão. Utilizando mockup 1b para envio de pacotes IED. Múltiplas Unidades Geradoras são utilizadas e múltiplas regras são inseridas com o fim de avaliar o processo de identificação de eventos
	2.3	Fazer o processo de identificação com base no recebimento dos dados do módulo 1.		1	
	2.4	Emitir um relatório em formato de pacote JSON para o módulo 3 a cada 0,5 seg.		1	Utilizar script externo para monitorar pacotes enviados pelo módulo

2.3 – Versionamento

Os recursos do software são distribuídos em versões conforme estimado pela tabela na sequência.

Tabela 2.3.1: *Tabela de recursos do sistema e versão.*

Versão	Recurso
0.1 *(abr/24)*	● **Funcionamento inicial** ○ Criação de Regras através da classe Rule; ○ Classe PacoteIED para instanciar pacotes recebidos da rede; ○ Recebimento e *bufferização* de pacotes, com funções de iniciar e parar; ○ Algoritmo de identificação de eventos; ○ Classe IED para contabilização de eventos; ○ *Timer* para atualização do relatório.
0.2 *(abr/24)*	● **Expansão das funções básicas** ○ Modificações na interface visual; ○ Classe Relatorio para armazenar contabilização de eventos; ○ Envio de pacotes de relatório na rede;
1.0 *(abr/24)*	● **Primeira Versão Totalmente Funcional** ○ Inserção e remoção dinâmica de regras durante recebimento de pacotes;
1.1 *(abr/24)*	● **Pequenos Ajustes e Correções** ○ Pequeno ajuste na comunicação UDP.

2.4.2 – Máquina de estados

Baseado no cenário padrão e requerimentos construídos, tem-se a seguinte proposição para a máquina de estados do módulo de filtragem de parâmetros e eventos em tempo real.

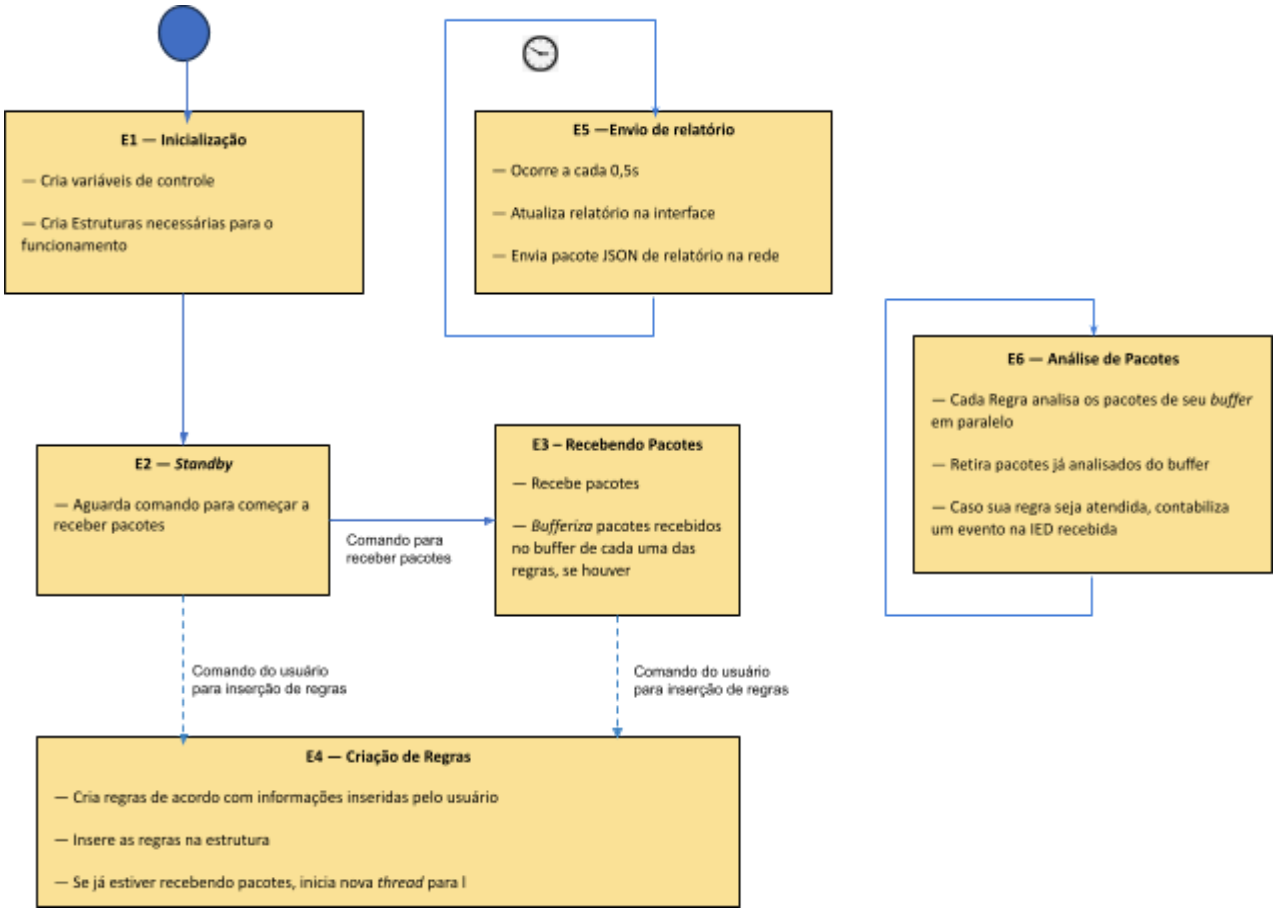

Figura 2.4.2.1: *Máquina de estados do software*

Importante notar que no mínimo 2 desses estados são operações em paralelo, o ‘Recebendo Pacotes’ e o ‘Envio de Relatório’, além de mais uma atividade de ‘Análise de pacotes’ em paralelo para cada regra inserida pelo usuário.

2.4.3 – Interfaces de usuário

Para fins de caracterização do sistema, a figura na sequência ilustra a interface desta aplicação indicando alguns de seus recursos previstos.

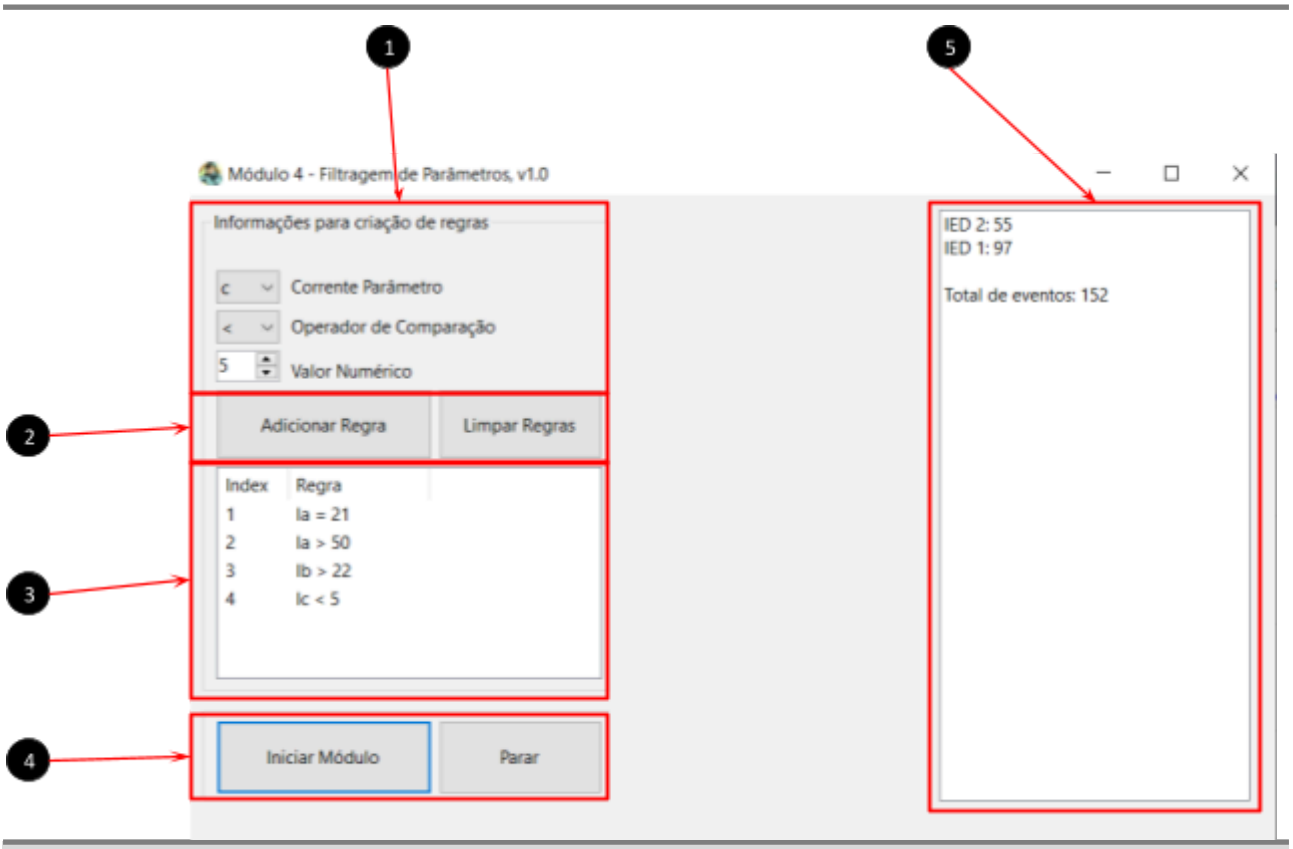

Figura 1.x: *Interface principal da aplicação*

Na sequência, uma breve descrição destes principais elementos:

❶ Campos para inserir parâmetros da regra. Corrente a ser analisada, operador de comparação a ser utilizado e valor numérico a ser comparado.

❷ Botões para adicionar regra nova e remover todas as regras adicionadas.

❸ Lista de regras já adicionadas, com suas informações.

❹ Botões para iniciar e parar o recebimento de pacotes, respectivamente.

❺ Relatório em texto dos eventos ocorridos. Dispões as mesmas informações que são enviadas no pacote JSON.

3 – Modelagem

3.1 – Blocos de elementos principais

Na sequência é mostrado um conjunto de diagramas de blocos para exemplificar a arquitetura do sistema. Cada bloco cinza é uma função, ou método do sistema. Pelas setas contíguas é possível visualizar o fluxo de dados, já as setas tracejadas representam as entradas do usuário.

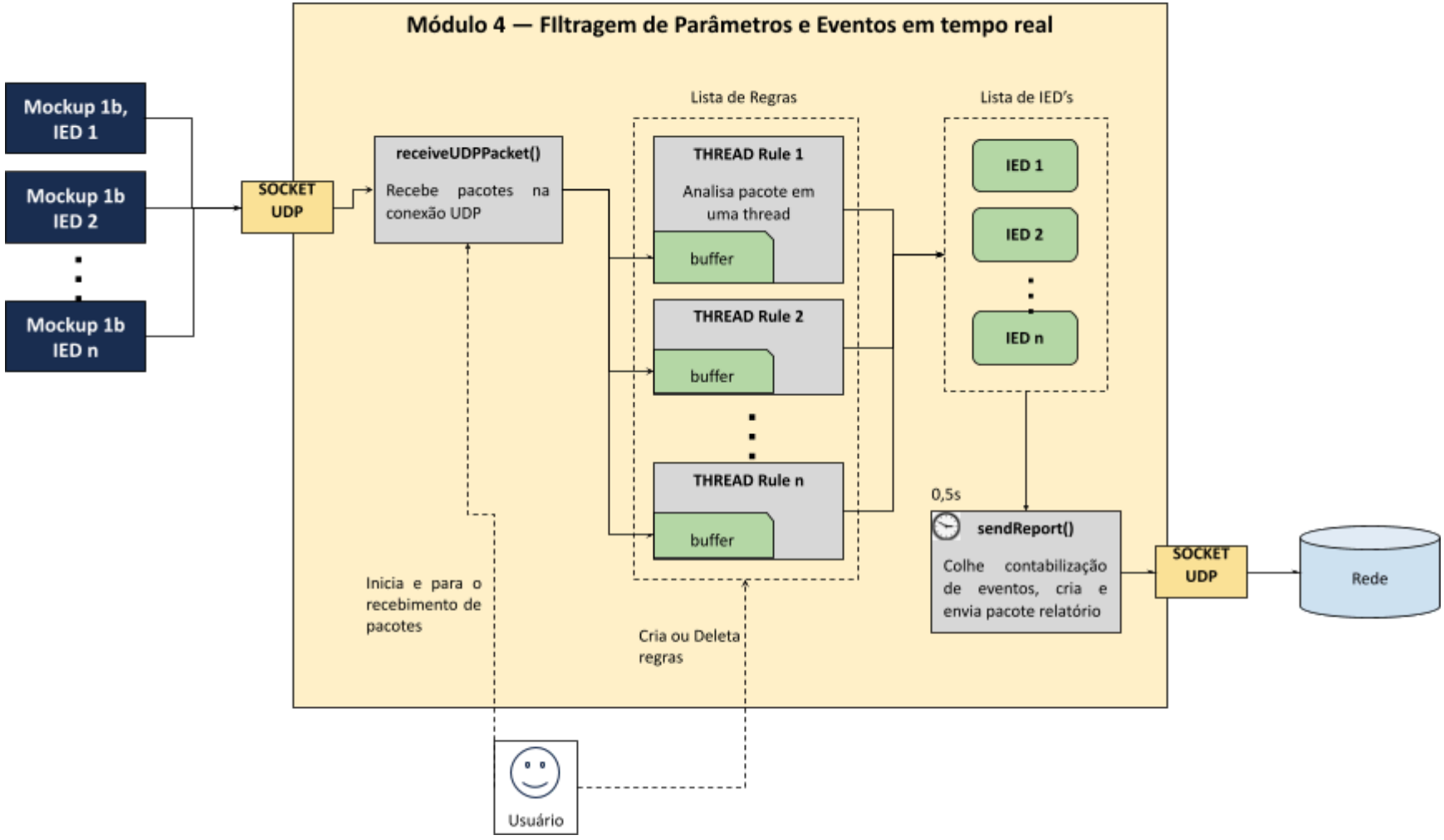

Figura 3.1.1: Diagrama de Arquitetura da Solução

3.2 – Fluxo geral de mensagens

A figura na sequência ilustra resumidamente as mensagens que são trocadas entre este módulo e a rede.

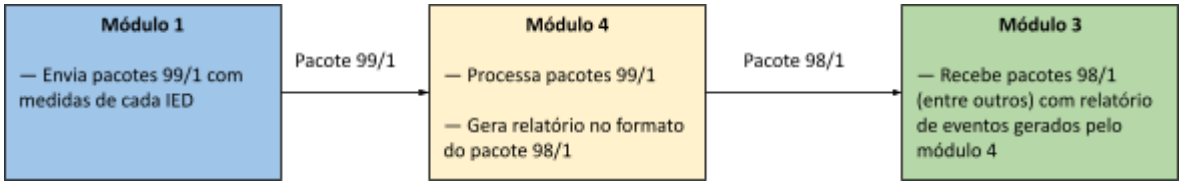

Figura 3.2.1: Diagrama ilustrando a troca de mensagens entre este sistema e outros módulos na rede

É possível observar que o módulo 4 captura mensagens com dados de medição vindas do módulo 4 para que possa executar suas funções de identificação de eventos. Em paralelo a isso, a cada 0,5 segundos este módulo irá utilizar as informações armazenadas para montar um pacote de relatório, contendo o número de eventos ocorridos no total e o número de eventos ocorridos por IED, enviando na rede para que o módulo 3 possa receber e usá-lo em seu monitoramento. Essas duas interações são as únicas troca de mensagens que o módulo executa.

3.3 – Modelagem detalhada dos recursos

3.4.1 – Recebimento de mensagens

Essa é a parte do sistema responsável por receber os pacotes 99/1 da rede e distribuí-los para o *buffer* de cada uma das regras cadastradas pelo usuário. Mais informações a respeito destes *buffers* podem ser encontradas na seção 3.4.2. O sistema só começa a receber pacotes a partir do momento que o usuário clica no botão “Iniciar Módulo”.

Na figura abaixo é possível visualizar um diagrama do funcionamento.

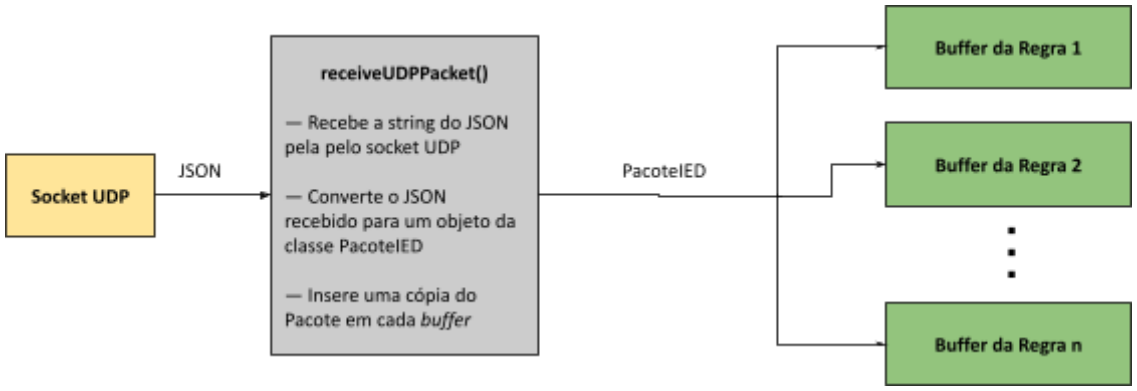

Figura 3.4.1.1: Diagrama ilustrando a troca de mensagens entre este sistema e outros módulos na rede

A função receiveUDPPacket() é a função responsável por receber os pacotes JSON da rede, converter para um objeto da classe PacoteIED e colocar uma cópia desse objeto no *buffer* de cada regra, onde eles aguardarão para serem analisados. Elá é executada em uma *thread* que começa no momento que o usuário comanda.

Além disso, essa função utiliza de uma variável de controle que indica se o módulo deve receber pacotes ou não. Quando a variável é colocada no valor falso, a função sai de seu loop, para de receber pacotes e eventualmente sua acaba, encerrando também a *thread*.

O algoritmo da função se encontra em pseudo-código abaixo:

Tabela 3.4.1.1: *Algoritmo equivalente da função receiveUDPPacket()*

Função	Algoritmo
receiveUDPPacket	1) Estabelece conexão UDP 2) Muda variável de controle para indicar que está recebendo pacotes 3) Cria vetor de *bytes* para receber pacote 4) Enquanto variável de controle indicar que está recebendo: a) Recebe *bytes* da conexão UDP b) converte para *string* c) faz um "truncamento" da string para evitar bytes indesejados após o fim do json d) converte string do json para objeto PacoteIED e) para cada regra R adicionada no módulo: i) insere PacoteIED no buffer da regra R

O formato do pacote de dados recebido é descrito abaixo, e também representa a classe PacoteIED:

Tabela 3.4.1.2: *Formato do pacote de dados 99/1*

Campo	Valores	Significado
URI	**99/1**	Identificador do pacote
Ia	int	Valor medido da corrente A
Ib	int	Valor medido da corrente B
Ic	int	Valor medido da corrente C
numPacote	int	Número desse pacote
idDispositivo	string	Id do IED que mediu e gerou esse pacote

3.4.2 – Análise de Pacotes

A análise de pacotes é realizada pela classe Rule. Cada regra inserida pelo usuário na interface, cria uma instância da classe Rule que executará a análise dos pacotes para identificar eventos. Essas instâncias então são inseridas numa lista, o que indica que essa regra está ativa e estará processando pacotes. Logo, no caso de um comando para limpar as regras, essa lista é esvaziada.

Para contabilizar os eventos, a classe IED armazena a quantidade de eventos detectados de cada dispositivo, identificados pela propriedade "idDispositivo" dos pacotes que chegam. Para gerar o relatório, todos os IED's conhecidos são mantidos em uma lista, isso significa que sempre que a leitura de um novo dispositivo chega ao módulo, uma nova instância da classe IED é criada e colocada na lista.

Quando o usuário comanda o início do módulo, além de criar a *thread* da função receiveUDPPacket, o sistema percorre a lista de regras e cria uma *thread* para cada uma, executando o método analyzePackets, e altera o valor de uma variável de controle que diz se regra está analisando pacotes. Além deste, as regras também possuem o método checkEvent, que verifica se um pacote atendeu os parâmetros definidos pela regra.

Abaixo está um resumo do algoritmo desses dois métodos principais da classe Rule.

Tabela 3.4.2.1: *Algoritmos equivalente para métodos da classe Rule*

Método	Algoritmo
analyzePackets	1. Enquanto variável *running* for verdadeira: a. se buffer estiver vazio, execute um Thread.sleep de 20ms b. se não: i. retira pacote do buffer ii. se função *checkevent* indica que teve evento: 1. se id do IED existe na lista de IEDs, contabiliza um evento neste IED 2. se id do IED não existe, cria um novo objeto IED com um evento já contado e adiciona-o na lista de IEDs
checkEvents	1. caso parametro da regra seja corrente a: a. salva propriedade Ia do pacote 2. caso parametro da regra seja corrente b: a. salva propriedade Ib do pacote 3. caso parametro da regra seja corrente c: a. salva propriedade Ic do pacote 4. caso operação da regra seja '>': a. retorna se parâmetro salvo é maior que valor da regra 5. caso operação da regra seja '=': a. retorna se parâmetro salvo é igual ao valor da regra 6. caso operação da regra seja '<': a. retorna se parâmetro salvo é menor que valor da regra

3.4.3 – Emissão de Relatório

A emissão de relatório é feita a cada 0,5s: um *timer* cria uma *thread* para executar a função sendReport. Essa função é responsável por percorrer a lista de IED's, contar quantos eventos teve em cada uma para obter um valor total, depois usar todas essas informações para gerar um relatório. Esses dados alimentam um componente da interface visual ao mesmo tempo que preenchem uma instância da classe Relatorio. A classe Relatorio armazena a quantidade de eventos total e por IED, depois é convertida diretamente para uma string JSON, para então poder ser enviada pela rede.

Um diagrama dessa relação pode ser visto em seguida.

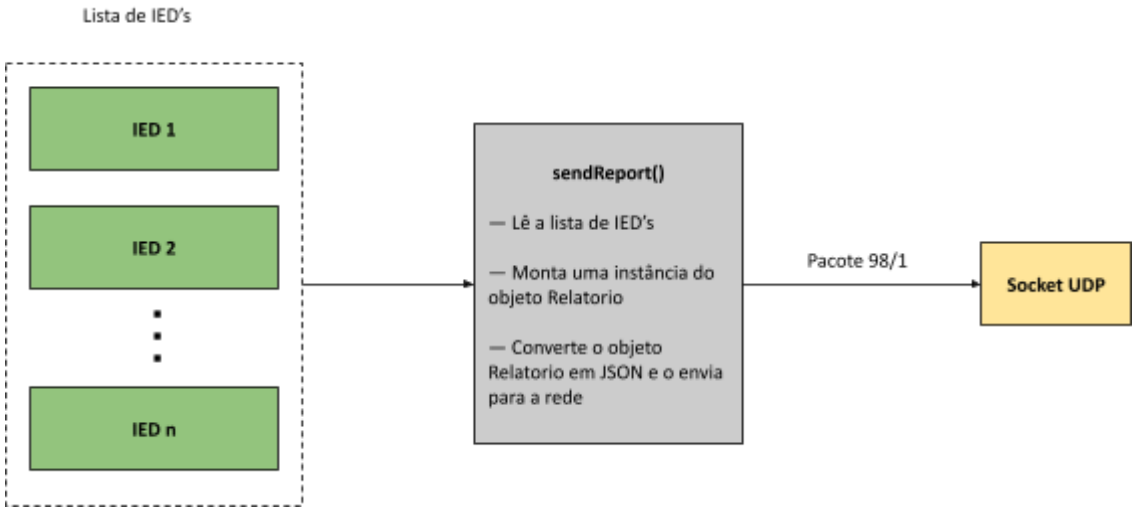

Figura 3.4.3.1: Diagrama ilustrando o envio de relatório pela função sendReport

Em acordo com outros módulos desenvolvidos, o pacote de relatório foi chamado de pacote "98/1"

Tabela 3.4.3.1: *Formato do pacote de dados 98/1*

Campo	Valores	Significado		
URI	**98/1**	Identificador do pacote		
totalEvents	int	Número total de eventos ocorridos		
events	array	**Campo**	**Valores**	**Significado**
		id	int	Id da IED
		qtdEventos	int	quantidade de eventos ocorridos nessa IED