Sistema de múltiplas conexões

Execução: Individual
Data de entrega: a ser definida

INTRODUÇÃO

O caminho da indústria 5.0 está começando a ser traçado, marcando mais um passo da revolução industrial. Novas abordagens voltadas à sustentabilidade e os humanos começam a surgir a fim de suportar diferentes

aplicações emergentes. Neste contexto, o desenvolvimento de equipamentos colaborativos, inteligentes, eficientes e confiáveis é fundamental para alcançar soluções de fabricações com desperdício zero, defeito zero e customização em massa. Ademais, como não pode ser diferente, as redes de computadores são possibilitadores importantes nesta revolução, uma vez que a comunicação e colaboração necessitam de uma infraestrutura de comunicação para que as decisões autônomas entre equipamentos sejam realizadas.

O gerente da indústria siderúrgica se empolgou com o rumo da Indústria 5.0 e decidiu "revolucionar" o processo de algumas linhas de produção da sua siderúrgica. Inicialmente, o gerente está com alguns equipamentos autônomos prontos para serem instalados e entrarem em funcionamento. Contudo, estes equipamentos precisam "conversar" uns com os outros para que tomem decisões eficientes. Para isso, surge a necessidade de uma infraestrutura de comunicação capaz de prover conexões simultâneas entre equipamentos, como ilustra a Figura 1.

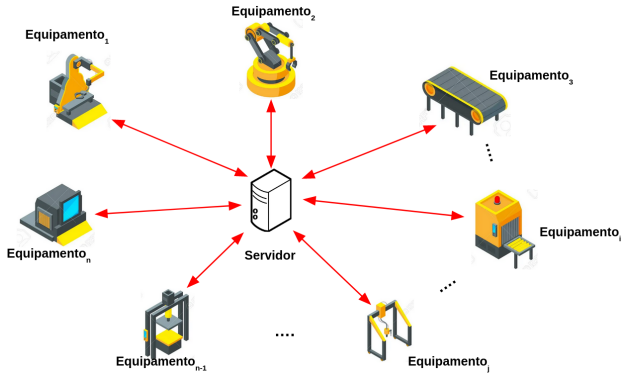

Figura 1 - Exemplo comunicação centralizada entre equipamentos

Neste trabalho prático, você será responsável por desenvolver um sistema que coordene múltiplas conexões simultâneas entre os equipamentos e permita a comunicação entre eles. Neste sistema, você deve desenvolver um **servidor**, responsável por coordenar as múltiplas conexões à medida em que os equipamentos entram e saem da rede, e os **equipamentos (os clientes)**, que solicitam informações de outros equipamentos por intermédio do servidor. Toda conexão deve utilizar a interface de *sockets* na linguagem C.

Você desenvolverá 2 programas para um sistema simples de troca de mensagens de texto utilizando apenas as funcionalidades da biblioteca de *sockets* POSIX e a comunicação via protocolo TCP. O código do servidor deverá utilizar múltiplas *threads* para a manutenção das múltiplas conexões. As próximas seções detalham o que cada entidade (servidor ou equipamento) deve fazer.

Os objetivos gerais deste trabalho são:

1. Implementar servidor utilizando a interface de *sockets* na linguagem C ou Python;
2. Implementar equipamento utilizando a interface de *sockets* na linguagem C ou Python;
3. Escrever o relatório;

PROTOCOLO

O protocolo de aplicação deverá funcionar sobre TCP. Isso implica que as mensagens serão entregues sobre um canal de bytes com garantias de entrega em ordem, mas é sua responsabilidade determinar onde começa e termina cada mensagem.

Especificação das Mensagens

Esta seção especifica as mensagens utilizadas na comunicação de controle e dados da rede, bem como as mensagens de erro e confirmação. Nas tabelas abaixo, as células em "–" correspondem aos campos

que não precisam ser definidos nas mensagens. A **coluna Tipo** não precisa ser adicionada à mensagem, para facilitar a implementação utilize a identificação da mensagem (Id Msg).

Mensagens de Controle					
Tipo	**Id Msg**	**Id origem**	**Id Destino**	**Payload**	**Descrição**
REQ_ADD	01	–	–	–	Mensagem de requisição de entrada de equipamento na rede
REQ_REM	02	$IdEQ_i$	–	–	Mensagem de requisição de saída de equipamento na rede, onde $IdEQ_i$ corresponde a identificação do equipamento solicitante.
RES_ADD	03	–	–	$IdEQ_i$	Mensagem de resposta com identificação $IdEQ_i$ do equipamento EQ_i
RES_LIST	04	–	–	$IdEQ_j$, $IdEQ_k$, …	Mensagem com lista de identificação de equipamentos. Onde $IdEQ_j$, $IdEQ_k$, … correspondem aos identificadores dos equipamentos transmitidos na mensagem.

Mensagens de Dados					
Tipo	**Id Msg**	**Id origem**	**Id Destino**	**Payload**	**Descrição**
REQ_INF	05	$IdEQ_i$	$IdEQ_j$	–	Mensagem de requisição de informação de um equipamento EQ_i de origem para um equipamento EQ_j de destino. Os campos $IdEQ_i$ e $IdEQ_j$ correspondem as identificações dos equipamentos de origem e destino, respectivamente.
RES_INF	06	$IdEQ_j$	$IdEQ_i$	Dados	Mensagem de resposta de informação de um equipamento EQ_j de origem para um equipamento EQ_i de destino. Os campos $IdEQ_j$ e $IdEQ_i$ correspondem as identificações dos equipamentos de origem e destino, respectivamente. O campo PAYLOAD corresponde ao valor do dado.

Mensagens de Erro ou Confirmação					
Tipo	**Id Msg**	**Id origem**	**Id Destino**	**Payload**	**Descrição**
ERROR	07	–	$IdEQ_j$	Code	Mensagem de erro transmitida do Servidor para equipamento $IdEQ_j$. O campo payload informa o código de erro. Abaixo descrição de cada código: 01 : "*Equipment not found*". 02 : "*Source equipment not found*"

					03 : "*Target equipment not found*" 04 : "*Equipment limit exceeded*"
OK	08	–	$IdEQ_j$	Code	Mensagem de confirmação transmitida do Servidor para equipamento $IdEQ_j$. O campo payload informa o código de confirmação. Abaixo decrição de cada código: 01 : "Successful removal".

Fluxo das Mensagens de Controle

Esta seção descreve o fluxo de mensagens de controle transmitidas entre os equipamentos e o servidor a fim de coordenar a comunicação dos equipamentos na rede. Além das decisões e impressões em tela realizadas pelos equipamentos e o servidor.

Abertura de conexão com Servidor

1. Um equipamento **EQ_i** solicita para o Servidor a abertura de conexão a fim de obter seu identificador na rede. A solicitação é realizada pelo envio da mensagem **REQ_ADD**.
2. Servidor recebe requisição de **EQ_i** e verifica se quantidade máxima de conexões foi alcançado.
    2.1. Em caso positivo, Servidor responde mensagem de error **ERROR(04)** para equipamento **EQ_i**.
        2.1.1. Equipamento **EQ_i** recebe mensagem **ERROR(04)** e imprime em tela a descrição do código de erro correspondente (vide *Especificação das Mensagens*).
    2.2. Em caso negativo, Servidor define um identificador **$IdEQ_i$** para **EQ_i** , registra **$IdEQ_i$** em sua base de dados, imprime em tela a mensagem "Equipment $IdEQ_i$ added" e envia para todos equipamentos atualmente conectados na rede (transmissão em *broadcast*) a identificação do novo equipamento por meio a mensagem **RES_ADD($IdEQ_i$)**.
        2.2.1. Equipamentos atualmente conectados recebem **RES_ADD($IdEQ_i$)**, registram **$IdEQ_i$** em suas bases de dados e imprimem em tela a mensagem "Equipment $IdEQ_i$ added". O equipamento **EQ_i** , ao receber **RES_ADD($IdEQ_i$)**, registra sua nova identificação e imprime em tela a mensagem "New ID: $IdEQ_i$".
        2.2.2. Servidor envia lista de equipamentos atualmente conectados para **EQ_i** (*transmissão unicast*) por meio da mensagem **RES_LIST($IdEQ_j$, $IdEQ_k$, …)**.
        2.2.3. Equipamento **EQ_i** recebe **RES_LIST($IdEQ_j$, $IdEQ_k$, …)** e registra novos equipamentos em sua base de dados.

Fechamento de Conexão com Servidor

1. Um equipamento **EQ_i** solicita para o Servidor o fechamento da conexão por meio da mensagem **REQ_REM($IdEQ_i$)**.
2. Servidor recebe **REQ_REM($IdEQ_i$)** e verifica se **$IdEQ_i$** existe na base de dados
    2.1. Em caso negativo, o servidor responde mensagem de erro **ERROR(01)** para equipamento **EQ_i**.
        2.1.1. Equipamento **EQ_i** recebe mensagem **ERROR(01)** e imprime em tela a descrição do código de erro correspondente (vide *Especificação das Mensagens*).
    2.2. Em caso positivo, o servidor remove **EQ_i** da base de dados, responde mensagem **OK(01)** para **EQ_i** , desconecta **EQ_i** , imprime em tela a mensagem "Equipment $IdEQ_i$ removed" e envia mensagem **REQ_REM($IdEQ_i$)** para todos equipamentos atualmente conectados na rede (transmissão em *broadcast*).

2.2.1. Equipamento **EQ_i** recebe mensagem **OK(01)** e imprime na tela descrição da mensagem, fecha a conexão e encerra a execução.
2.2.2. Equipamentos atualmente conectados recebem **REQ_REM(IdEQ$_i$)**, removem **IdEQ$_i$** de suas bases de dados e imprimem em tela "Equipment IdEQ$_i$ removed".

Fluxo das Mensagens de Dados

Esta seção descreve fluxo de mensagens de dados trocadas entre os equipamentos e o servidor a fim dos equipamentos trocarem informações uns com os outros na rede. Além das decisões e impressões em tela realizadas pelos equipamentos e servidor

Equipamento EQ$_i$ solicita informação para equipamento EQ$_j$ na rede

1. Um equipamento **EQ_i** solicita para o Servidor informação de equipamento **EQ_j** por meio da mensagem **REQ_INF(IdEQ$_i$, IdEQ$_j$)**.
2. Servidor recebe **REQ_INF(IdEQ$_i$, IdEQ$_j$)** e verifica se **IdEQ$_i$** existe em sua base de dados.
    2.1. Em caso negativo, Servidor imprime em tela "*Equipment IdEQ$_i$ not found*" e responde mensagem de erro **ERROR(02)** para equipamento **EQ_i**.
        2.1.1. Equipamento **EQ_i** recebe **ERROR(02)** e imprime na tela a descrição da mensagem.
    2.2. Em caso positivo, o servidor verifica se o **EQ_j** existe em sua base de dados.
        2.2.1. Em caso negativo, Servidor imprime em tela "*Equipment IdEQ$_j$ not found*" e responde mensagem de erro **ERROR(03)** para equipamento **EQ_i**.
            2.2.1.1. Equipamento **EQ_i** recebe **ERROR(03)** e imprime na tela a descrição da mensagem.
        2.2.2. Em caso positivo, Servidor repassa **REQ_INF(IdEQ$_i$, IdEQ$_j$)** para **EQ_j**
            2.2.2.1. Equipamento **EQ_j** recebe mensagem **REQ_INF(IdEQ$_i$, IdEQ$_j$)** , imprime em tela "requested information", gera informação (**valor aleatório***) e responde para Servidor as informações por meio da mensagem **RES_INF(IdEQ$_j$, IdEQ$_i$, PAYLOAD)**.
            2.2.2.2. Servidor recebe **RES_INF(IdEQ$_j$, IdEQ$_i$, PAYLOAD)** e verifica se **IdEQ$_j$** existem na base de dados.
                2.2.2.2.1. Em caso negativo, o servidor imprime em tela "*Equipment IdEQ$_j$ not found*" e responde mensagem de erro **ERROR(02)** para equipamento **EQ_j**.
                    2.2.2.2.1.1. Equipamento **EQ_j** recebe **ERROR(03)** e imprime na tela a descrição da mensagem.
                2.2.2.2.2. Em caso positivo, Servidor verifica se **EQ_i** existe na base de dados.
                    2.2.2.2.2.1. Em caso negativo, Servidor imprime em tela "*Equipment IdEQ$_i$ not found*" e responde mensagem de erro **ERROR(03)** para equipamento **EQ_i**.
                        2.2.2.2.2.1.1. Equipamento **EQ_i** recebe **ERROR(03)** e imprime na tela a descrição da mensagem.
                    2.2.2.2.2.2. Em caso positivo, o servidor repassa **RES_INF(IdEQ$_j$, IdEQ$_i$, PAYLOAD)** para **IdEQ$_i$**.
                        2.2.2.2.2.2.1. **IdEQ$_i$** recebe **RES_INF(IdEQ$_j$, IdEQ$_i$, PAYLOAD)** e imprime em tela "Value from IdEQ$_j$: PAYLOAD".

* O valor aleatório deve ser um número decimal aleatório entre 0 e 10 com 2 casas decimais (e.g. 2.34, 5.87 ou 10.00)

IMPLEMENTAÇÃO

Pequenos detalhes devem ser observados no desenvolvimento de cada programa que fará parte do sistema. É importante observar que o protocolo é simples e único (o cliente sempre tem que enviar a mensagem codificada para o servidor e vice-versa, de modo que o correto entendimento da mensagem deve ser feito por todos os programas).

Como mencionado anteriormente, a implementação do protocolo da aplicação utilizará TCP. Haverá apenas um socket em cada equipamento (cliente), independente de quantos outros programas se comunicarem com aquele processo. O programador deve usar as funções *send* e *recv* para enviar e receber mensagens. No caso do servidor, ele deve manter um socket para receber novas conexões (sobre o qual ele executará *accept*) e um socket para cada cliente conectado.

O tipo de endereço utilizado neste trabalho prático deve ser IPv4. O **servidor** deve tratar até 15 conexões simultâneas. Ademais, o servidor é responsável por definir identificações únicas para cada equipamento na rede. Um **equipamento (clientes)** inicia sem identificação, após a solicitação de entrada na rede ele recebe sua identificação. Também, o equipamento deve receber mensagens do teclado. Tanto servidor quanto equipamento devem imprimir as mensagens recebidas na tela.

Comandos

O equipamento deve receber alguns comandos pela entrada padrão (teclado) a fim de interagir com o sistema. Tais comando são descritos a seguir:

- **close connection** : Comando requisita fechamento de conexão, o que dispara o fluxo de mensagem de controle “Fechamento de Conexão com Servidor”.
- **list equipment** : Comando lista os equipamentos na base da dados do equipamento por id. (e.g. “**IdEQ$_i$ IdEQ$_j$ IdEQ$_k$ ….**”)
- **request information from IdEQ$_j$:** Comando requisita informação de equipamento **IdEQ$_j$** o que dispara o fluxo de mensagem de dados “Equipamento EQ$_i$ solicita informação para equipamento EQ$_j$ na rede”

Execução

Seu servidor deve receber um número de porta na linha de comando especificando em qual porta ele vai receber conexões (Sugestão: utilize a porta 51511 para efeitos de padronização do trabalho). Seu equipamento (cliente) deve receber, **estritamente nessa ordem**, o endereço IP e a porta do servidor para estabelecimento da conexão. Para realizar multiplas conexões de equipamentos com o servidor basta executar multiplas vezes o código do equipamento.

A seguir, um exemplo de execução de dois equipamentos conectados com um servidor em três terminais distintos:

```
Terminal 1: ./server 51511
Terminal 2: ./equipment 127.0.0.1 51511
Terminal 3: ./equipment 127.0.0.1 51511
```

AVALIAÇÃO

O trabalho deve ser realizado individualmente e **deve ser implementado na linguagem de programação C** utilizando somente a biblioteca padrão (interface POSIX de sockets de redes). Deve ser possível executar seu programa no sistema operacional **Linux** e **não deve utilizar bibliotecas Windows, como o winsock**. Seu programa deve interoperar com qualquer outro programa implementando o mesmo protocolo (você pode

testar com as implementações dos seus colegas). Procure escrever seu código de maneira clara, com comentários pontuais e bem indentados. Isto facilita a correção dos monitores e tem impacto positivo na avaliação.

Entrega

Cada aluno deve entregar documentação em PDF de até 6 páginas, sem capa, utilizando fonte tamanho 10, e figuras de tamanho adequado ao tamanho da fonte. **Ele deve conter uma descrição da arquitetura adotada para o servidor, os refinamentos das ações identificadas no mesmo, as estruturas de dados utilizadas, as decisões de implementação não documentadas nesta especificação**. Como sugestão, considere incluir as seguintes seções no relatório: introdução, arquitetura, servidor, equipamento, discussão e conclusão. O relatório deve ser entregue em formato PDF. A documentação corresponde a 20% dos pontos do trabalho, mas só será considerada para as funcionalidades implementadas corretamente.

Será utilizado um sistema para detecção de código repetido, portanto não é admitido cola de trabalhos. **Será adotada a média harmônica entre as notas da documentação e da execução, o que implica que a nota final será 0 se uma das partes não for entregue**.

Cada aluno deve entregar, além da documentação, o **código fonte em C** e um **Makefile** para compilação do programa. Instruções para submissão e compatibilidade com o sistema de correção semi-automática:

- O Makefile deve compilar o “equipment” e o “server”.
- Seu código deve ser compilado pelo comando “make” sem a necessidade de parâmetros adicionais.
- A entrega deve ser feita no formato ZIP, seguindo a nomenclatura: TP2_MATRICULA.zip
- O nome dos arquivos deve ser padronizado:
    - server.c
    - equipment.c
    - common.c, common.h (se houver)

Prazo de entrega

Os trabalhos poderão ser entregues até às 23:59 (vinte e três e cinquenta e nove) do dia especificado para a entrega. O horário de entrega deve respeitar o relógio do sistema Moodle, ou seja, a partir de 00:00 do dia seguinte à entrega no relógio do Moodle, os trabalhos **não poderão ser entregues**. Logo não serão considerados trabalhos entregues fora do prazo definido.

Dicas e Cuidados

- O guia de programação em rede do Beej (http://beej.us/guide/bgnet/) tem bons exemplos de como organizar um servidor
- Procure escrever seu código de maneira clara, com comentários pontuais e bem identado.
- Não se esqueça de conferir se seu código não possui erros de compilação ou de execução.
- Implemente o trabalho por partes. Por exemplo, implemente o tratamento das múltiplas conexões, depois crie os formatos das mensagens e, por fim, trate as mensagens no servidor ou cliente.

Exemplos de execução

Esta seção apresenta alguns exemplos de execuções do sistema. A fim de facilitar a explicação, as tabelas a seguir detalham o passo a passo dos comandos de entrada (**em negrito**) e as informações que devem ser impressas em tela em cada instante de tempo. A Tabela 1 apresenta um cenário de conexão de dois equipamentos (Terminal 1 e 2) com o servidor (Terminal 0).

Tempo	Terminal 0	Terminal 1	Terminal 2
t_1	**./server 51511**		
t_2		**./equipment 127.0.0.1 51511**	
t_3	Equipment 01 added		
t_4		New ID: 01	
t_5			**./equipment 127.0.0.1 51511**
t_6	Equipment 02 added		
t_7		Equipment 02 added	New ID: 02

Tabela 1 - Cenário exemplo de abertura de conexão

A Tabela 2 apresenta um cenário que solicita equipamentos conectados e fechamento de conexão, assumindo que equipamentos 01 (Terminal 1), 02 (Terminal 2) e 03 (Terminal 3) estão atualmente conectados com servidor (Terminal 0).

Tempo	Terminal 0	Terminal 1	Terminal 2	Terminal 3
t_1		**list equipment**		
t_2		02 03		
t_3			**close connection**	
t_4	Equipment 02 removed			
t_5			Successful removal	
t_6		Equipment 02 removed		Equipment 02 removed
t_7				**list equipment**
t_8				01

Tabela 2 - Cenário exemplo de fechamento de conexão e listar equipamentos

A Tabela 3 apresenta um cenário de solicitação de informação para equipamento existente e inexistente, assumindo que equipamentos 01 (Terminal 1), 02 (Terminal 2) e 03 (Terminal 3) estão atualmente conectados com servidor (Terminal 0) e equipamento 04 não está conectado.

Tempo	Terminal 0	Terminal 1	Terminal 2	Terminal 3
t_1		**request information from 03**		
t_2				requested information
t_3		Value from 03: 2.56		

t_4			**request information from 04**	
t_5	Equipment 04 not found			
t_6			Target equipment not found	

Tabela 3 - Cenário exemplo de solicitação de informação para equipamento existente e inexistente

A Tabela 4 apresenta um cenário em que existem 15 equipamentos conectados simultaneamente com o servidor (Terminal 0) e um novo equipamento (Terminal 1) solicita abertura de conexão.

Tempo	Terminal 0	Terminal 1
t_1		**./equipment 127.0.0.1 51511**
t_2		Equipment limit exceeded

Tabela 4 - Cenário exemplo de limite de equipamentos excedido