

INF01151 – SISTEMAS OPERACIONAIS II N
SEMESTRE 2020/1

TRABALHO PRÁTICO PARTE 1: THREADS, SINCRONIZAÇÃO E COMUNICAÇÃO

ESPECIFICAÇÃO DO TRABALHO

A proposta de projeto é implementar um serviço de troca de mensagens instantâneas, para permitir a comunicação síncrona entre vários usuários em grupos. A proposta deve ser desenvolvida em duas etapas. A primeira etapa compreenderá funcionalidades que dependerão de tópicos como threads, processos, comunicação e sincronização para serem implementadas. A aplicação deverá executar em ambientes Unix (Linux), mesmo que tenha sido desenvolvida em outras plataformas. O projeto deverá ser implementado em C/C++, usando a API Transmission Control Protocol (TCP) sockets do Unix.

FUNCIONALIDADES BÁSICAS

O projeto compreenderá um app para usuários (cliente) e um servidor. Ao iniciar o app em seu dispositivo, o usuário deve especificar um grupo no qual deseja ingressar. Os grupos devem ser criados automaticamente pelo servidor no ingresso do primeiro usuário. O servidor deve gerenciar os grupos e os usuários pertencentes a cada grupo, garantindo que todas as mensagens enviadas pelos usuários sejam entregues a todos os usuários participantes do grupo.

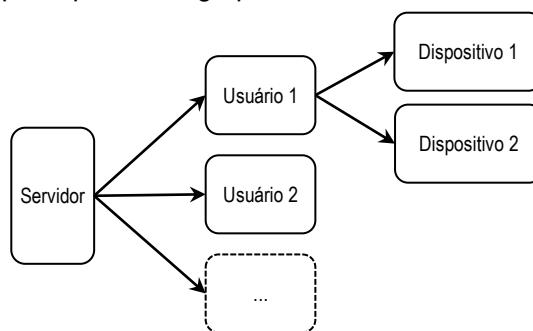


Figura 1. Visão geral da organização e comunicação entre as entidades do sistema.

O serviço deve fornecer suporte às seguintes funcionalidades básicas:

- **Identificação dos usuários e grupos:** Os usuários e grupos serão identificados por uma string entre 4 e 20 caracteres. Os identificadores deverão iniciar por uma letra, e conter apenas letras, números e ponto (.).
- **Múltiplos usuários:** O servidor deve permitir vários acessos simultâneos ao serviço, ou seja, deve atender vários usuários ao mesmo tempo.
- **Múltiplas sessões:** Um usuário deve poder utilizar o serviço através de até **dois** dispositivos distintos simultaneamente¹. Caso o usuário tente acessar o serviço a partir de um terceiro

¹ Ao abrir a app em um novo dispositivo, o usuário poderá ingressar no mesmo grupo, ou em outro grupo. No primeiro caso, o nome do usuário aparecerá apenas uma vez na lista de usuários do grupo, e as mensagens enviadas por ambos os dispositivos aparecerão como sendo enviadas pelo mesmo usuário. Da mesma forma, as mensagens recebidas no grupo devem aparecer em ambos os dispositivos. Isso inclui as mensagens enviadas pelo próprio usuário (imagine uma pessoa usando ao mesmo tempo o whatsapp no celular e na web).

dispositivo, deverá aparecer uma mensagem de erro informando que o número máximo de conexões simultâneas para um mesmo usuário foi atingido. Ao iniciar uma sessão, o usuário deve receber as últimas N^2 mensagens enviadas no grupo que está ingressando.

- **Consistência nas estruturas de armazenamento e troca de mensagens:** As estruturas de armazenamento de dados no servidor devem ser mantidas em um estado consistente. As mensagens enviadas por um usuário em um grupo devem ser distribuídas a todos os usuários participantes do mesmo. Todos os usuários do grupo devem possuir a mesma visão histórica da conversa que ocorre no grupo; em outras palavras, as mensagens enviadas no grupo devem aparecer na mesma ordem para todos os usuários.
- **Persistência de dados no servidor:** Os grupos e o histórico das últimas N mensagens trocadas nos grupos devem ser restabelecidas na reinicialização do servidor.

O SISTEMA

O projeto está dividido em duas etapas, sendo que a segunda etapa irá contemplar funcionalidades complementares às enumeradas na primeira etapa. Portanto, considere uma implementação modular e com possibilidade de extensão, e o encapsulamento das funções de comunicação do cliente e do servidor em módulos isolados.

A figura abaixo apresenta uma sugestão de como a equipe pode implementar os módulos do sistema. Os **módulos de comunicação** são responsáveis por operações de troca de dados entre o servidor e os apps dos usuários³. O gerenciador de comunicação deve trocar periodicamente mensagens de *keep alive* para identificar usuários que desconectam de forma não graciosa (por ex., quando a app ou o dispositivo do usuário travam). O timeout para detectar que um usuário desconectou de forma não graciosa deve ser de até 1 minuto. Os **módulos de gerenciamento** de mensagens são responsáveis por gerenciar a troca de mensagens entre os usuários nos grupos, incluindo os metadados (usuário autor da mensagem e hora de envio). Para isso, o servidor deve manter um arquivo com as últimas mensagens trocadas em cada grupo, incluindo os seus metadados.

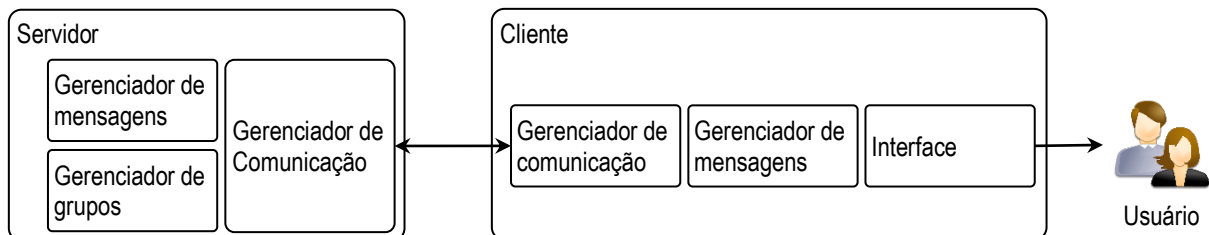


Figura 2. Sugestão de organização dos módulos do sistema.

A sincronização está relacionada à dinâmica de troca de mensagens no grupo. Quando um usuário envia uma mensagem ao grupo, a mesma somente deve aparecer na tela do usuário quando a mesma for recebida com sucesso pelo servidor. Esse comportamento deverá garantir o requisito de que todos os usuários vejam as mesmas mensagens enviadas no grupo na mesma ordem.

INTERFACE DO USUÁRIO

Um cliente deve poder estabelecer uma sessão com o servidor via linha de comando utilizando:

```
$ ./app <username> <groupname> <server_ip_address> <port>
```

onde <username> representa o identificador do usuário, <groupname> representa o identificador do grupo, <server_ip_address> e <port> representam o endereço IP do servidor e a porta, respectivamente.

² O parâmetro N deve ser informado na inicialização do servidor, por linha de comando ou via arquivo de configuração.

³ Utilize múltiplas threads/processos para que o módulo de comunicação do servidor possa suportar usuários simultâneos.

Após iniciar uma sessão, uma mensagem de sistema deve ser enviada no grupo informando que o usuário está online. Essa mensagem deve ser recebida pelo próprio usuário, com uma indicação personalizada, como no exemplo abaixo.

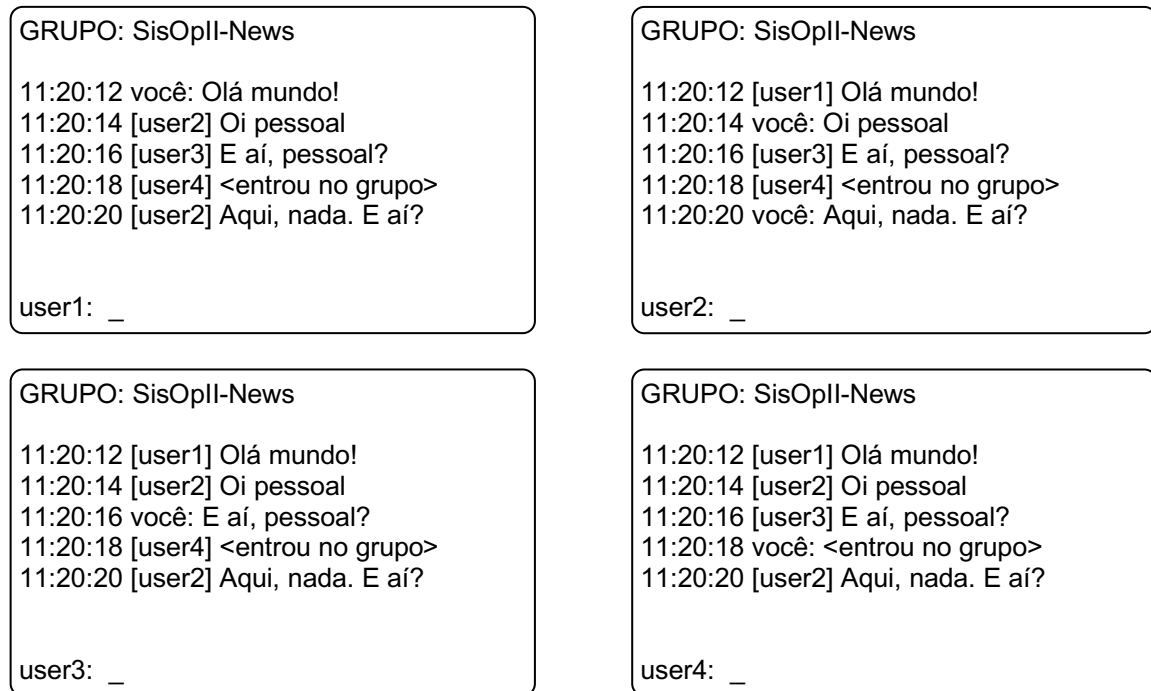


Figura 3. “Prints” das telas de quatro usuários participando em um mesmo grupo (SisOpII-News). Os “prints” não necessariamente representam a forma como a interface do app deve ser estruturada; a definição do layout da interface fica a critério da equipe. No entanto, espera-se que a interface seja intuitiva e apresente o nome do usuário e o grupo onde o usuário se encontra.

Ao apertar CTRL+C (interrupção) ou CTRL+D (fim de arquivo), o app deverá encerrar, sinalizando ao servidor que o usuário está desconectando do serviço. Essa informação também deve ser enviada a todos os usuários, indicando que o usuário saiu do grupo.

Sugere-se que a interface possua uma thread para escrever as mensagens na tela, e outra thread para ler as mensagens digitada pelo usuário.

FORMATO DE ESTRUTURAS

A equipe tem liberdade para definir o tamanho e formato das mensagens que serão usadas para troca de dados entre o app do cliente e o servidor. Sugere-se a especificação de uma estrutura para definir as mensagens trocadas entre cliente/servidor. Abaixo é apresentada uma sugestão de como implementar a estrutura para a troca de comandos e mensagens entre o app e o servidor.

```
typedef struct __packet{
    uint16_t type;           //Tipo do pacote (p.ex. DATA | CMD)
    uint16_t seqn;          //Número de sequência
    uint16_t length;        //Comprimento do payload
    uint16_t timestamp;     //Timestamp do dado
    const char* _payload;    //Dados da mensagem
} packet;
```

DESCRIÇÃO DO RELATÓRIO

A equipe deverá produzir um relatório fornecendo os seguintes dados:

- Descrição do ambiente de testes: versão do sistema operacional e distribuição, configuração da máquina (processador(es) e memória) e compiladores utilizados (versões).
- Explicação e respectivas justificativas a respeito de:
 - (A) Como foi implementada a concorrência no servidor para atender múltiplos clientes;
 - (B) Em quais áreas do código foi necessário garantir sincronização no acesso a dados;
 - (C) Descrição das principais estruturas e funções que a equipe implementou;
 - (D) Explicar o uso das diferentes primitivas de comunicação;
- Inclusão no relatório de uma descrição dos problemas que a equipe encontrou durante a implementação e como estes foram resolvidos (ou não).

A **nota será atribuída baseando-se nos seguintes critérios**: (1) qualidade do relatório produzido conforme os itens acima, (2) correta implementação das funcionalidades requisitadas e (3) qualidade do programa em si (incluindo uma interface limpa e amigável, documentação do código, funcionalidades adicionais implementadas, etc).

DATAS E MÉTODO DE AVALIAÇÃO

O trabalho deve ser feito em grupos de **3 OU 4 INTEGRANTES**. Não esquecer de identificar claramente os componentes do grupo no relatório.

Faz parte do pacote de entrega os arquivos fonte e o relatório em um arquivo ZIP. O trabalho deverá ser entregue até às **08:30 do dia 7 de outubro**. A entrega deverá ser via moodle (link para submissão no Tópico 08). As demonstrações ocorrerão no mesmo dia, no horário da aula.

Após a data de entrega, o trabalho deverá ser entregue via e-mail para alberto@inf.ufrgs.br (subject do e-mail deve ser "INF01151: Trabalho Parte 1"). Neste caso, será descontado 02 (dois) pontos por semana de atraso. O atraso máximo permitido é de duas semanas após a data prevista para entrega. Isto é, nenhum trabalho será aceito após o dia 21 de outubro.