

Performance nas Aplicações em Rede

Claudio Miranda – SUMMA
TECHNOLOGIES

claudio@claudius.com.br



COMDEX'2003

- ▶ Em dispositivos móveis, a performance da aplicação é um fator importante. A comunicação de dados é um requisito necessário para o sucesso de aplicações em um ambiente cada vez mais conectado.

- ▶ Claudio Miranda, consultor sênior da SUMMA TECHNOLOGIES, participante ativo da comunidade Java brasileira através do SouJava, palestrante em outros eventos de Java no Brasil e JavaOne. Colunista da revista JavaMagazine. Como arquiteto e consultor participa em projetos na utilização da plataforma Java/J2EE/J2ME.

Ao fim desta palestra você saberá onde é possível otimizar o código e o ambiente para melhorar o tempo de resposta de suas aplicações MIDP e entender um pouco mais do protocolo HTTP.

Para um ótimo aproveitamento deste tutorial é importante:

- ▶ Experiência na linguagem Java
- ▶ Conhecimentos de CLDC/MIDP
- ▶ Será coberto apenas o protocolo HTTP

- ▶ Considerações
- ▶ Conectividade em MIDP
- ▶ Threads
- ▶ XML
- ▶ Proxy Server
- ▶ Recomendações do fornecedor

- ▶ **Considerações**
- ▶ Conectividade em MIDP
- ▶ Threads
- ▶ XML
- ▶ Proxy Server
- ▶ Recomendações do fornecedor

- ▶ Rede da Operadora
 - Latência ?
 - Velocidade ?
 - Gateway:
 - Restrições (portas, multiplas conexões)
 - Suporta HTTP 1.1 (Keep-alive, etc.)
- ▶ Expectativa dos Usuários
- ▶ Entender da aplicação
 - Fluxo de mensagens
 - Quantidade de dados em cada interação
- ▶ Conexões Batch

HTTP Server



Sem suporte
a TCP/IP



Rede TCP/IP



Suporte a
TCP/IP
Nativo

Rede TCP/IP

Rede sem suporte a IP:
TL/PDC ou WSP



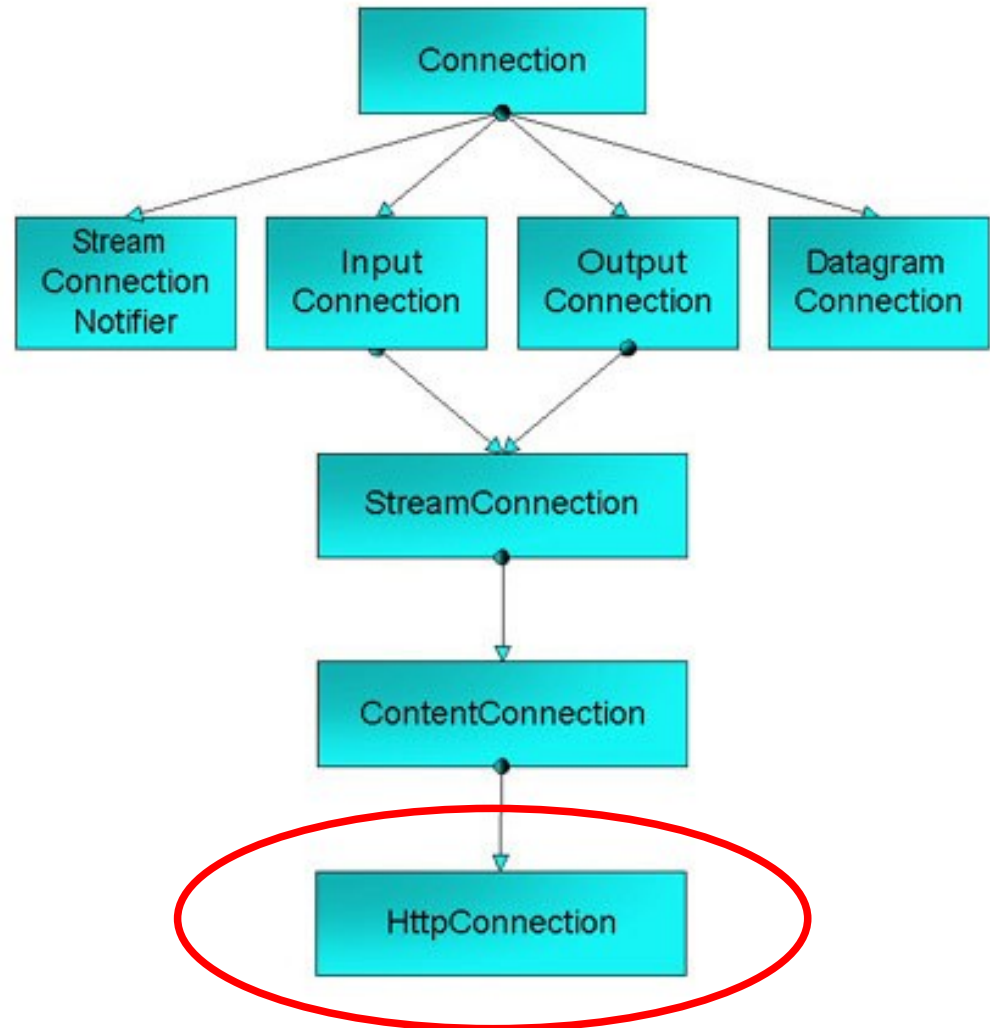
Gateway da Operadora

- ▶ Considerações
- ▶ **Conectividade em MIDP**
- ▶ Threads
- ▶ XML
- ▶ Proxy Server
- ▶ Recomendações do fornecedor

▶ **G**eneric
Connection
Framework

- ▶ HTTP 1.1 é requerido
- ▶ Outros protocolos são opcionais

`javax.microedition.io`



```
url = "http://www.claudius.com.br/comdex/t1.txt";  
conn = (HttpConnection) Connector.open(url);  
conn.setRequestMethod(HttpConnection.GET);  
in = conn.openInputStream();  
StringBuffer buffer = new StringBuffer();  
int b = 0;  
while ((b = in.read()) != -1) {  
    buffer.append((char) b);  
}  
String resultado = buffer.toString();
```

Leitura byte a byte

```
url = "http://www.claudius.com.br/comdex/t1.txt";  
conn = (HttpConnection) Connector.open(url);  
conn.setRequestMethod(HttpConnection.GET);  
in = conn.openInputStream();  
StringBuffer buffer = new StringBuffer();  
int b = 0;  
while (b = in.read()) != -1) {  
    buffer.append((char) b);  
}  
String resultado = buffer.toString();
```

Conectividade em MIDP

Leitura em uma operação apenas

```
url = "http://www.claudius.com.br/comdex/t1.txt";  
conn = (HttpConnection) Connector.open(url);  
conn.setRequestMethod(HttpConnection.GET);  
int length = (int) conn.getLength();  
DataInputStream in = conn.openDataInputStream();  
byte[] bArr = new byte[length];  
in.readFully(bArr);  
String resultado = new String(bArr);
```

Demonstração



Conectividade em MIDP

- ▶ Sempre que puder realize a leitura em uma operação só (`DataStream.readFully`). Use leitura byte a byte quando não é possível determinar o `content-length` (HTTP 1.0)
- ▶ Não use `InputStream.available()`
- ▶ Sempre realize medições com `System.currentTimeMillis()`

Cabeçalhos HTTP

- ▶ **Connection:** Controla se as conexões HTTP devem ser reutilizadas ou fechadas
- ▶ **Content-length:** o tamanho em bytes da quantidade de dados a ser lidos/enviados ao servidor HTTP.

- ▶ Considerações
- ▶ Conectividade em MIDP
- ▶ **Threads**
- ▶ XML
- ▶ Proxy Server
- ▶ Recomendações do fornecedor

- ▶ Aplicação com menor possibilidade de “congelamento”.
- ▶ Tarefas assíncronas
- ▶ Verifique a limitação do aparelho
- ▶ Seja cuidadoso

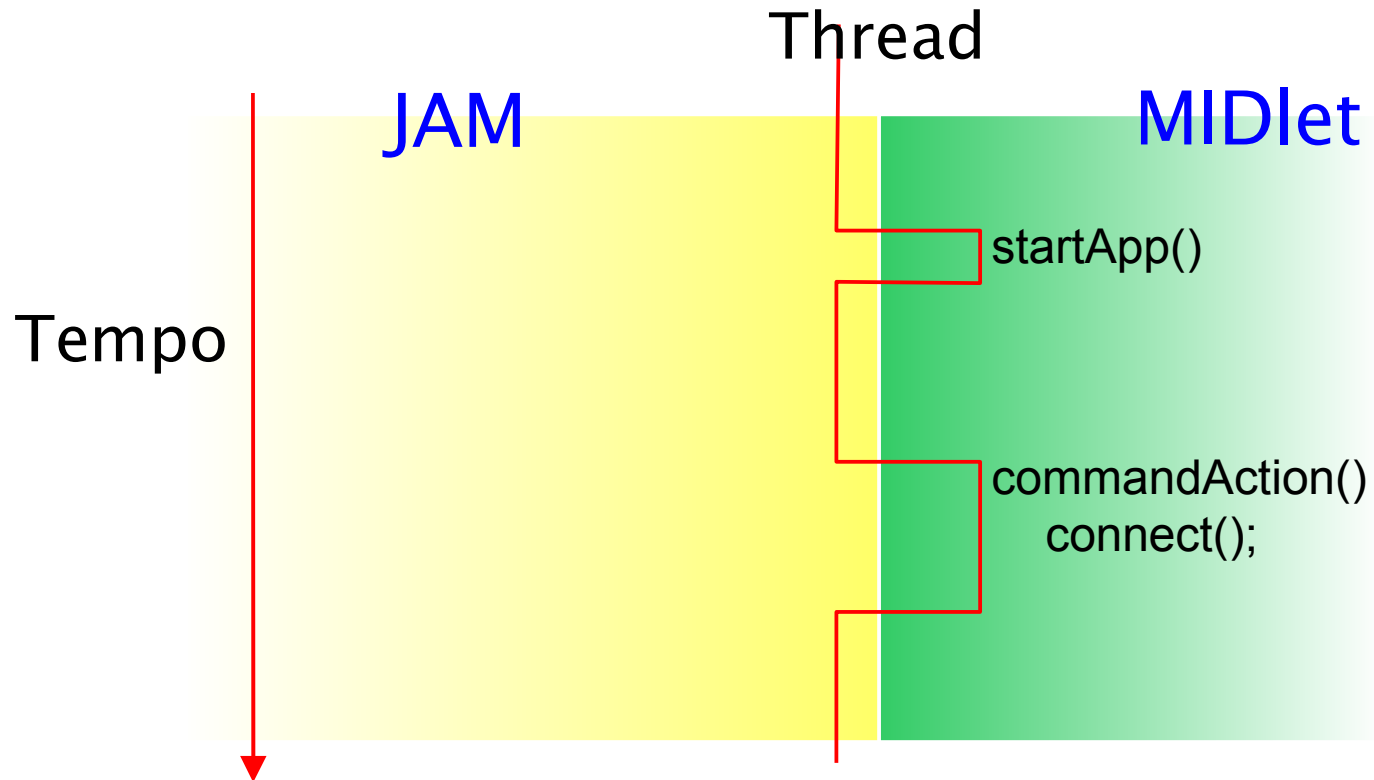
```

public void commandAction(Command c, Displayable d) {
    connectHttp();
}

void connectHttp() {
    ...
    url = "http://www.claudius.com.br/comdex/t1.txt";
    conn = (HttpConnection) Connector.open(url);
    conn.setRequestMethod(HttpConnection.GET);
    in = conn.openInputStream();
    ...
}

```

Ocupação da mesma thread de controle do MIDlet



```
Public void commandAction(Command c, Displayable d) {  
    connectHttp();  
}
```

```
void connectHttp() {  
    Thread t = new Thread(new Conexao());  
    t.start();  
}
```

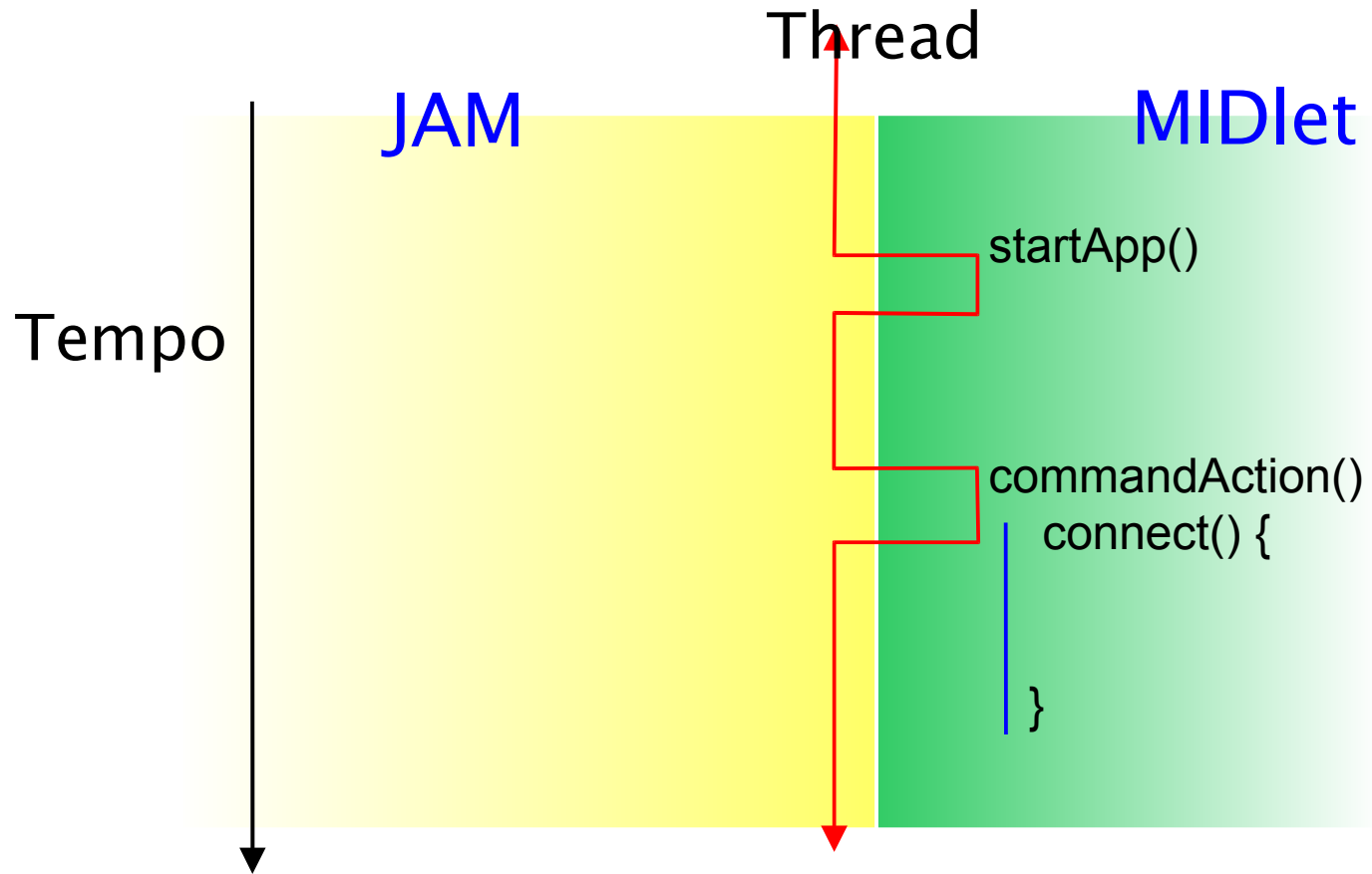
- ▶ A Interface de Usuário é liberada
- ▶ É necessário mostrar ao usuário o processamento em andamento

```

public class Conexao implements Runnable {
    public void run() {
        HttpURLConnection conn = null;
        DataInputStream in = null;
        try {
            conn = (HttpURLConnection) Connector.open(url);
            conn.setRequestMethod(HttpURLConnection.GET);

            in = conn.openDataInputStream();
            int length = (int) conn.getLength();
            byte[] bytesLidos = new byte[length];
            in.readFully(bytesLidos);
            resultado = new String(bytesLidos);
        } finally {
            IOUtil.close(in);
            IOUtil.close(conn);
        }
    }
}

```





Demonstração



- ▶ Crie Threads para ter operações de conexões assíncronas
- ▶ Ao criar Threads tome cuidado para não sobrecarregar a JVM
- ▶ Use WorkerThreads (Schedulable Threads)
 - controla a quantidade de Threads

Worker Threads

```
public synchronized void run() {  
    while (!running) {  
        try { wait(); }  
        catch (InterruptedException ie) {}  
        if (running) {  
            connect();  
        }  
    }  
}
```

```
public synchronized void go() {  
    notify();  
}
```

- ▶ Considerações
- ▶ Conectividade em MIDP
- ▶ Threads
- ▶ XML
- ▶ Proxy Server
- ▶ Recomendações do fornecedor

- 
- ▶ Existem 24 dicas para otimizar o processamento em XML

- ▶ Use um ProxyServlet
- ▶ NÃO USE XML em MIDP

- ▶ Considerações
- ▶ Conectividade em MIDP
- ▶ Threads
- ▶ XML
- ▶ **Proxy Server**
- ▶ Recomendações do fornecedor

- ▶ Integração com serviços existentes em servidores
- ▶ Diminuição do tráfego de dados
- ▶ Menor custo
- ▶ Maior manutenabilidade
- ▶ Alta Disponibilidade
- ▶ Facilidade de atualização
- ▶ Otimização de protocolo e recursos

Proxy Servlet

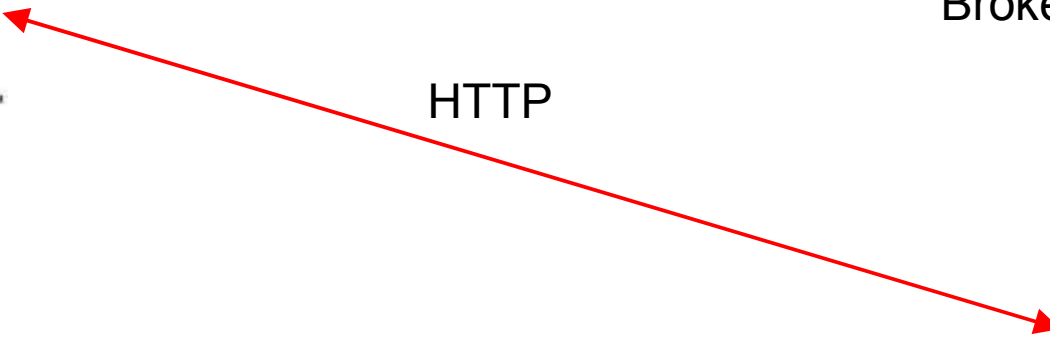


HTTP



Broker de Dados/Integração

HTTP



Broker de Dados/Integração

Proxy Servlet



Muito processamento e parsing de dados no telefone, muita atividade de GC

HTTP

Muito processamento e parsing de dados no telefone

HTTP



Broker de Dados/Integração



Broker de Dados/Integração

PROBLEMAS!

Proxy Servlet



HTTP

Parser de dados
Customização para
disponibilização de
informações



Servidor para aplicações
wireless



Broker de Dados/Integração



Broker de Dados/Integração

- ▶ Considerações
- ▶ Conectividade em MIDP
- ▶ Threads
- ▶ XML
- ▶ Proxy Server
- ▶ **Recomendações do fornecedor**

- ▶ Veja sempre as recomendações do fornecedor
- ▶ Limitações do telefone: Tamanho do JAR, HEAP, Tamanho das colunas do JAD, Protocolos disponiveis no GCF
- ▶ Use com cuidado a API do fornecedor
- ▶ Verifique atualizações de firmware e bugs
- ▶ Frequente os foruns e listas de e-mails

- ▶ Use threads para conectividade
- ▶ Libere recursos assim que possível
- ▶ Reuse conexões
- ▶ Use telas de esperas para operações demoradas
- ▶ Use um ProxyServer
- ▶ Não use XML
- ▶ Leia sempre de uma vez só usando o Content-length e `DataInputStream.readFully()`
- ▶ E



▶ KISS

▶ YAGNI

▶ DTSTTCPW

- ▶ KISS = Keep It Simple, Stupid!
- ▶ YAGNI = You Aren't Gonna Need It
- ▶ DTSTTCPW = Do The Simplest Thing That Could Possibly Work



Perguntas e Respostas



Courtesy Nokia



Muito Obrigado!

Claudio Miranda – SUMMA
TECHNOLOGIES

claudio@claudius.com.br