JGroups

Um Framework para comunicação em Grupo

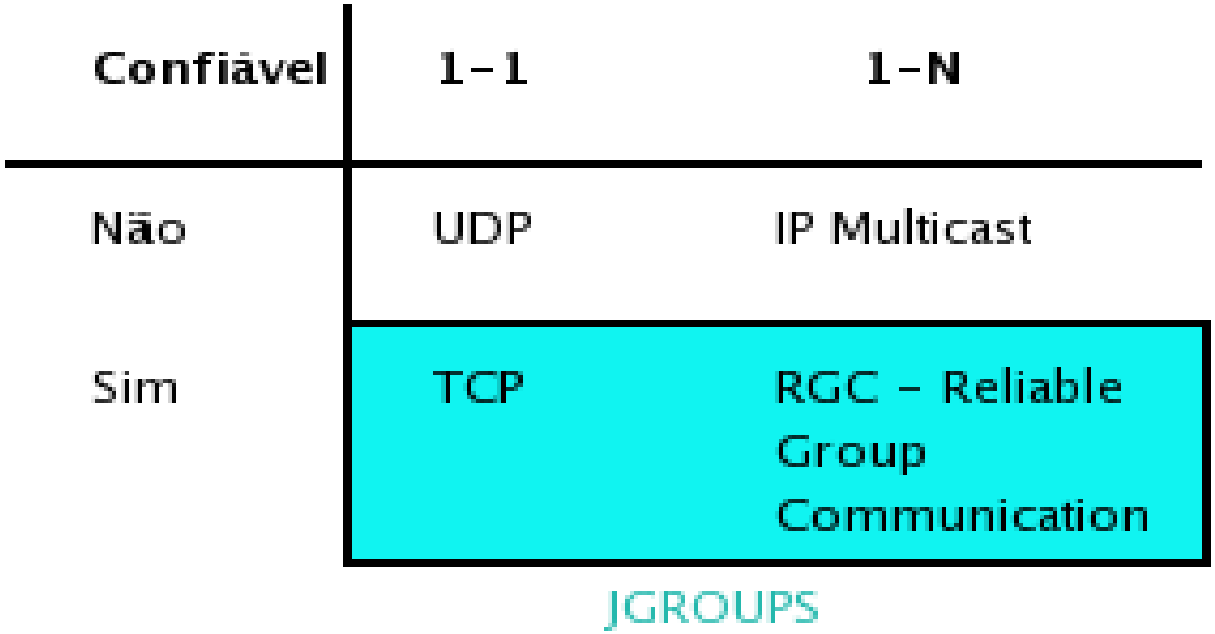
Ivan F. Martinez
05/2006

Agenda

- Introdução
- Aplicação
- Exemplo de utilização
- Arquitetura
  - Protocolos
  - API
- JGroups-ME
- Aplicações que utilizam JGroups

Introdução

- Framework para Comunicação Confiável em Grupo (RGC)
- Comunicação 1-N tão simples como 1-1



Os Problemas

- Vários processos em equipamentos diferentes
- Tolerância a falhas
- Balanceamento de carga
- Dificuldades de comunicação
  - Fragmentação em rede
  - Perda de pacotes
  - Ordenação
- Implementação complexa
- Gerenciamento dos membros do grupo

Exemplos de Aplicações

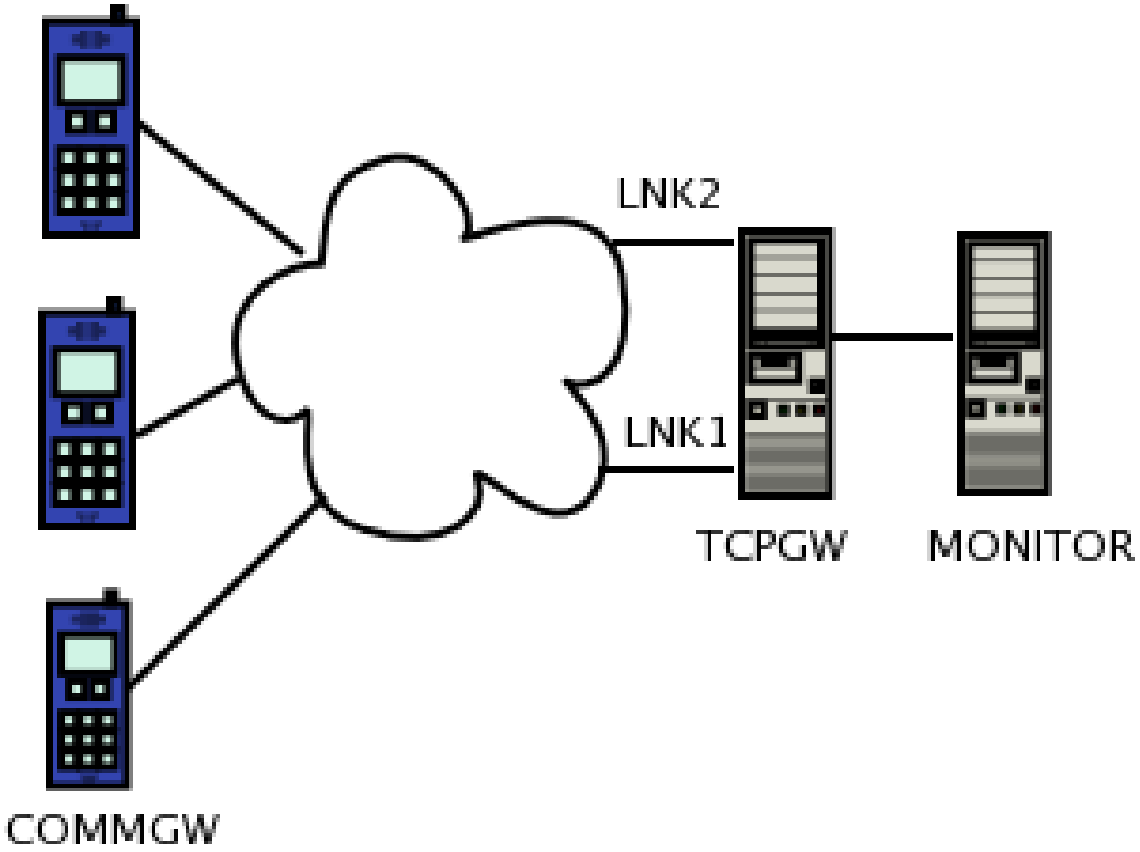
- Tradicional CHAT
- Cache Distribuído
- Computação em Grade
- Comunicação Inter-Processos
- RPC
- Distribuição de informação, contexto
- Redundância
- Notificação/Avisos Push

Histórico

- 1998-10/1999
  - Inicio do JavaGroups - Cornell (Horus/Ensemble)
- 2001-11
  - Versão 1.0
- 2003-09
  - Mudança de nome JGroups
  - Versão 2.2
- 2003-12
  - 1ª demonstração do JGroups-ME
- 2005-12
  - Versão 2.2.9.1
- 2006-05
  - 2.3 Candidate Release 1

Estudo de Caso

- Sistema de telemetria integrado via Celular

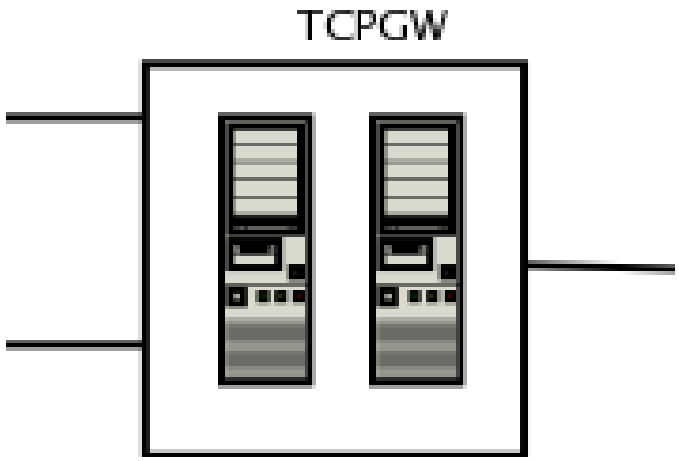


TCPGW

- Recebe conexões vindas dos terminais celulares e funciona como um “CHAT” para que o processo de monitoramento tenha acesso ao terminal remoto

- Alteração com aproximadamente 20 linhas de código permitiu a utilização de várias maquinas com mesmo processo

TCPGW

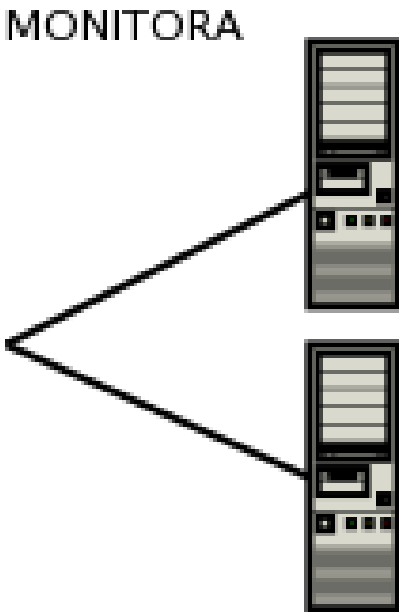


MONITOR

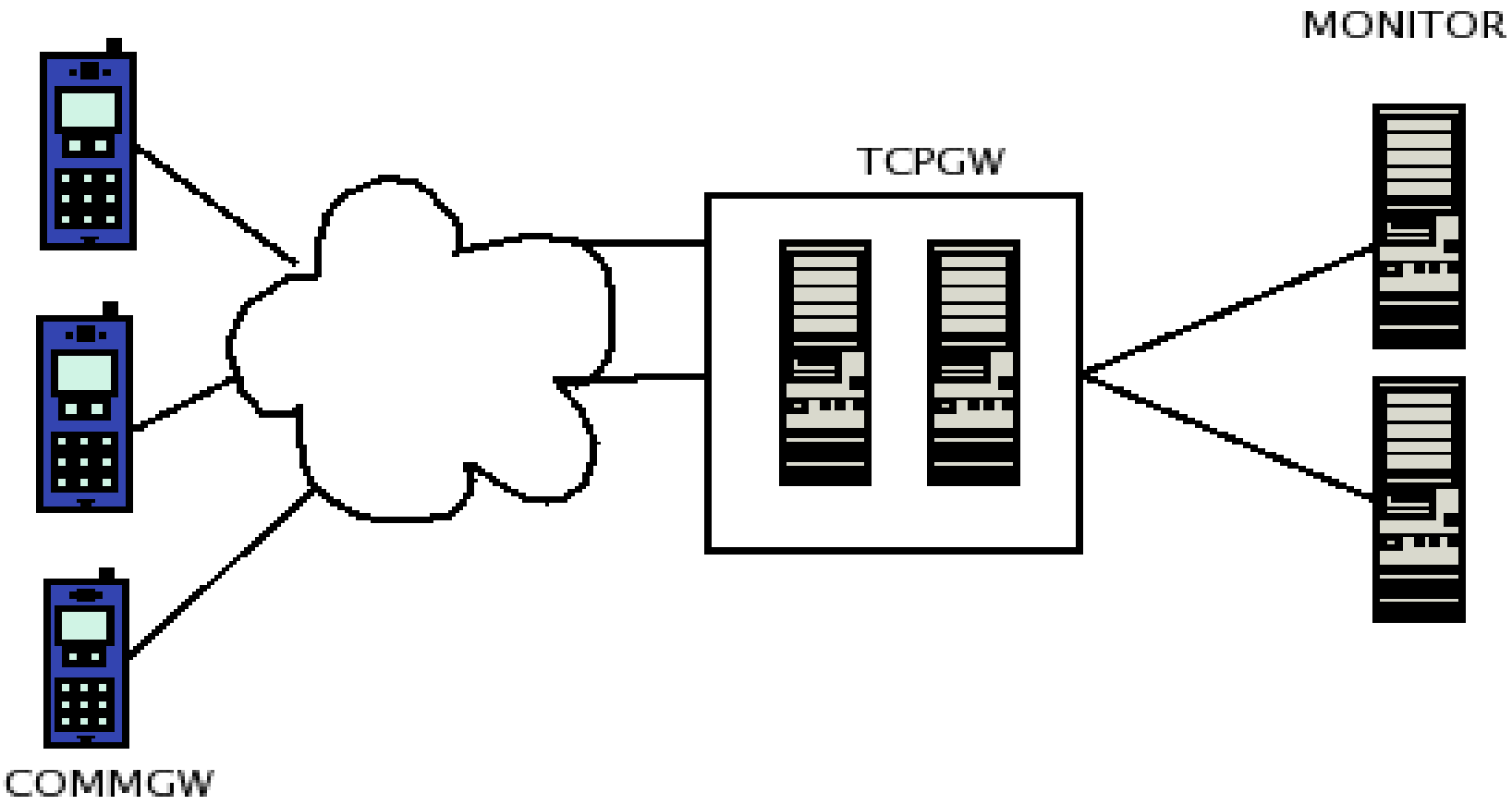
- Processo que a partir dos períodos de aquisição registrados no banco de dados acessa os terminais remotos para buscar informação

- Aproximadamente 15 linhas de código permitiram a inicialização do processo em várias máquinas, e somente o coordenador faz a aquisição dos dados.

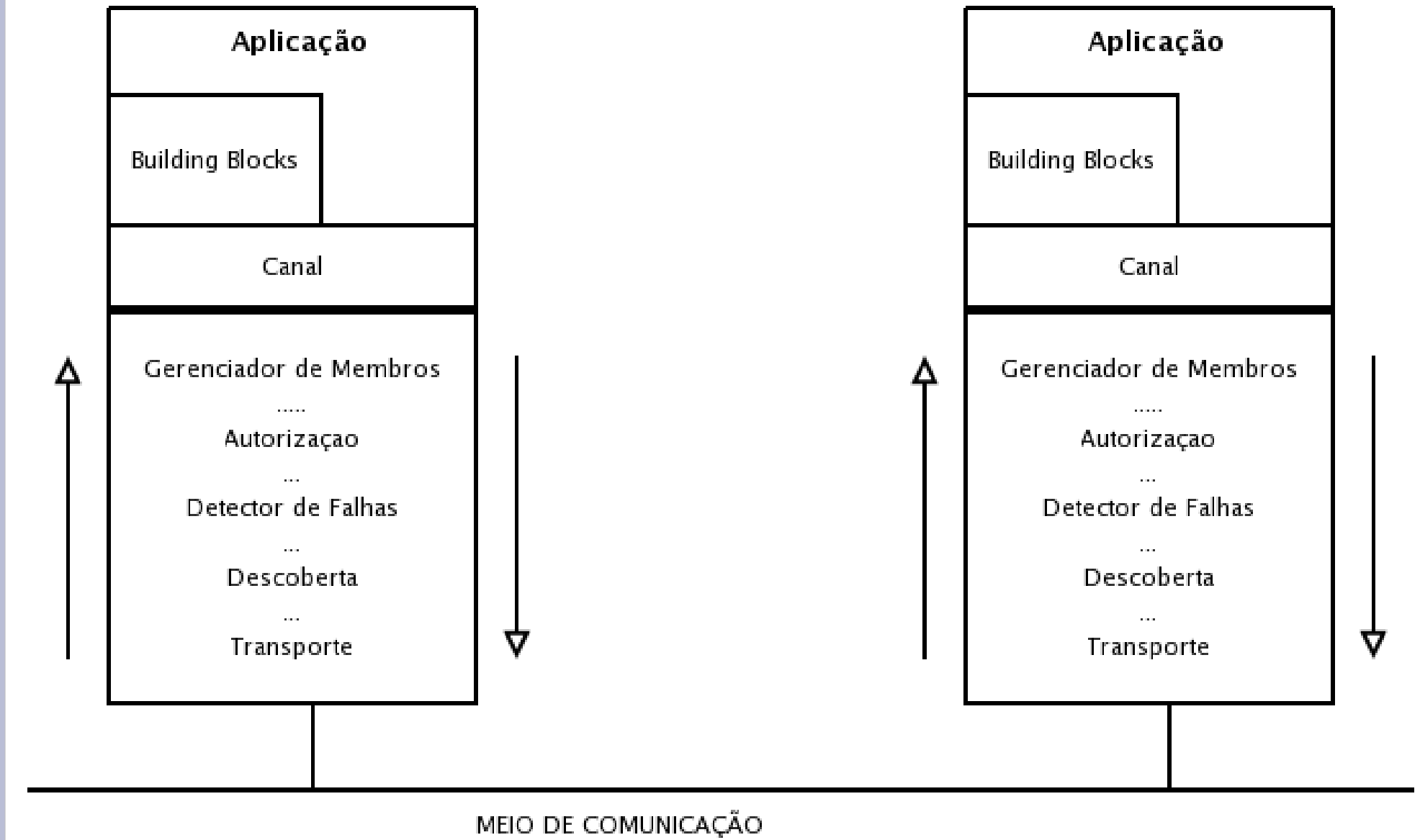
MONITOR



Conjunto final



Arquitetura



Protocolos Básicos

- Transporte
  - JMS, LOOPBACK, TCP, TCP_NIO, TUNNEL, UDP
- Descoberta
  - GOSSIP, MPING, PING, TCPPING
- Detecção de Falhas
  - FD, FD_SIMPLE, FD_SOCK,VERIFY_SUSPECT
- Confiabilidade
  - FRAG, NAKACK, SMACK, UNICAST
- Membros
  - GMS, VIEW_SYNC

Protocolos Adicionais

- Reagrupamento
  - MERGE, MERGE2, MERGE3, MERGEFAST
- Ordenação
  - CAUSAL, FIFO, SEQUENCER, TOTAL, TOTAL_TOKEN
- Sincronização
  - FLUSH, QUEUE, STABLE, VIEW_ENFORCER
- Controle de Fluxo
  - FC, FLOW_CONTROL, FLOWCONTROL

Protocolos Adicionais

- Autenticação:
  - AUTH (cvs/JGroups 2.3)
    - PLAIN/MD5/Shared Key
- Privacidade
  - ENCRYPT
- Outros
  - AUTOCONF
  - COMPRESS
  - PIGGYBACK
  - STATE_TRANSFER

Protocolos para o Desenvolvedor

- Depuração
  - BSH, PERF, PERF_TP, PRINTOBJS, PRINTMETHODS, SIZE, STATS, TRACE
- Simulação
  - DELAY, DEADLOCK, DISCARD, LOSS, PARTITIONER, SHUFFLE

Multiplexer

- JGroups 2.3
- Permite pilha compartilhada
- Múltiplos canais lógicos utilizando uma pilha comum de protocolos.
- Ainda não está totalmente especificado.
- Economia de Memória/Threads
- Interessante para sistemas com diversas aplicações rodando na mesma JVM (Ex. JBoss)

Protocol - API

```
package org.jgroups.stack;

public abstract class Protocol {
        ...
        public void down(Event evt) { passDown(evt); }
        public void up(Event evt) { passUp(evt); }
        public void passUp(Event evt);
        public void passDown(Event evt)
        ...
        // all protocol names have to be unique !
        public abstract String getName();

}
```

Event - API

```
package org.jgroups;

public class Event {

    public Event(int type);
    public Event(int type, Object arg);

    public int getType();
    public void setType(int type);
    public Object getArg();
    public void setArg(Object arg);
}
```

Exemplo de Configuração em Texto

```
UDP(mcast_addr=224.27.27.27;mcast_port=54324;ip_tt
 l=32)
:PING(timeout=3000;num_initial_members=6)
:FD(timeout=5000)
:VERIFY_SUSPECT(timeout=1500)
:pbcast.NAKACK(gc_lag=10;retransmit_timeout=3000)
:pbcast.STABLE(desired_avg_gossip=20000;max_bytes=
 1000000)
:UNICAST(timeout=5000)
:FRAG
:pbcast.GMS(join_timeout=5000;join_retry_timeout=2
 000;shun=false;print_local_addr=false)
```

Exemplo de configuração em XML

```
<config>
<UDP mcast_addr="224.27.27.27" mcast_port="54324"
 ip_ttl="32" />
   <PING timeout="3000" num_initial_members="6" />
   <FD timeout="5000" />
   <VERIFY_SUSPECT timeout="1500" />
 <pbcast.NAKACK gc_lag="10"
 retransmit_timeout="3000" />
<pbcast.STABLE desired_avg_gossip="20000"
 max_bytes="1000000" />
 <UNICAST timeout="5000" />
 <FRAG />
 <pbcast.GMS join_timeout="5000"
 join_retry_timeout="2000" shun="false"
 print_local_addr="false" />
</config>
```

Building Blocks

- Padrões pré-definidos para utilização
- PullPushAdapter
- DistributedHashTable/DistributedTree
- ReplicatedHashTable/ReplicatedTree
- MessageDispatcher(Comunicação Sincrona)
- RequestCorrelator
- RpcDispatcher
- GroupRequest

Canais

- Classe : org.jgroups.Channel/JChannel
- Endereçamento comum
- Nome comum
- Permite que os membros conectados no canal se comuniquem
- 1-1
- 1-N
- Modo Push ou Pull

JChannel - API

```
package org.jgroups;

public class Jchannel {

      public JChannel(String properties) throws;

      public void connect(String channel_name) throws;
      public void disconnect();
      public void close();

      public void send(Message msg) throws;
      public void send(Address dst, Address src, Serializable obj)
                                          throws;
      public Object receive(long timeout) throws;
      public View getView();

      public Address getLocalAddress();
      public void setOpt(int option, Object value);
}
```

Utilizando JChannel

```
JChannel channel = new JChannel(config);

channel.setOpt(Channel.LOCAL, Boolean.FALSE);

channel.setOpt(Channel.AUTO_RECONNECT,
  Boolean.TRUE);

channel.connect("canal1");
```

Mensagens

- Classe – org.jgroups.Message
- Qualquer objeto Seriável

Message - API

```
package org.jgroups;

public class Message {

    public Message(Address dest, Address src,
                   Serializable obj);

    public Object getObject();
    public Address getDest();
    public Address getSrc();
    public void putHeader(String key, Header hdr);
    public Header getHeader(String key);
}
```

Exemplo Envio/Recebimento de Mensagem

```
channel.send(null, null, "Hello World");


try {
        Object o = channel.receive(1000);
} catch (TimeoutException e) {}


MessageListener ml = new MessageListener() {
                public void receive(Message arg0) {
                    System.out.println(arg0);
                }
                public byte[] getState() { return null; }
                public void setState(byte[] arg0) { };
};
PullPushAdapter adapter = new PullPushAdapter(channel, ml);
```

Outras Classes

- org.jgroups.Address
  - Identificador de Endereço
- org.jgroups.View
  - Indica os membros do grupo
  - Ordenados, o 1º é Coordenador

Licença

- LGPL
- Pode ser utilizado em qualquer projeto mesmo não Open Source
- Mudanças, caso sejam liberadas devem ser LGPL

JGroups-ME

- Implementação reduzida de para J2ME
- Versão atual baseada em JGroups 2.2.8
- CLDC 1.1
- CDC (Connected Device Configuration)/ PP (Personal Profile) 1.0

Aplicações que utilizam JGroups

- Jakarta – Java Caching System
  - http://jakarta.apache.org/jcs/
- JBoss Cache
  - http://www.jboss.com/products/jbosscache
- JBoss Clustering
- Jetty – Replicação de Sessão
  - http://jetty.mortbay.org/jetty/index.html
- Sequoia – Evolução de C-JDBC 2.0
  - http://sequoia.continuent.org/
- Tomcat – Replicação de Sessão
  - http://tomcat.apache.org/
- ActiveMQ – JMS 1.1 Provider
  - http://docs.codehaus.org/display/ACTIVEMQ/Home

Frameworks de Abstração para Comunicação em Grupo

- Hedera
  - http://hedera.continuent.org/HomePage
- JGCS
  - http://jgcs.sourceforge.net/

Outros Frameworks de Comunicação em Grupo

- Appia
  - http://appia.continuent.org
- Tribe
  - http://tribe.objectweb.org/
- ActiveCluster
  - http://activecluster.codehaus.org/
- The Spread Toolkit
  - http://www.spread.org/
- NeEM
  - http://neem.sourceforge.net/
- JazzEnsemble (Wireless)
  - http://dsl.cs.technion.ac.il/projects/JazzEnsemble

Referências

- http://www.jgroups.org
- http://www.jboss.org/products/jgroups
- http://www.jgroups-me.org/
- Ensemble
  - http://dsl.cs.technion.ac.il/projects/Ensemble/
- Building Secure and Reliable Network Applications
  - Kenneth P. Birman

Contato

Ivan Francolin Martinez
ivanfm@ime.usp.br

Orientador:
Prof. Dr. Francisco Carlos da Rocha Reverbel
reverbel@ime.usp.br