JBoss The Professional Open Source Company

Professional Open Source™

JGroups: a library for reliable multicasting

Bela Ban
bela@jboss.com
http://www.jgroups.org
http://www.jboss.com/products/jgroups

Agenda

Professional Open Source™

Overview
- Architecture

API

Available protocols
- Transports, discovery, reliable retransmission, fragmentation, state transfer, failure detection, group membership

What is unreliable ?

Professional Open Source™

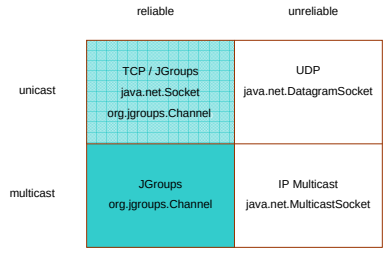
Messages get
- dropped
  - too big (UDP has a size limit), no fragmentation
  - buffer overflow at the receiver, switch
    - NIC, IP network buffer
- reordered

We don't know who is in a group (IP multicast)
- we don't know when a new node joins, leaves, or crashes

Fast sender might overwhelm slower receiver(s)
- flow control

Overview

Professional Open Source™



So what Is JGroups ?

Professional Open Source™

Library for reliable multicasting

Provides
- Fragmentation
- Message retransmission
- Flow control
- Ordering
- Group membership, membership change notification

LAN or WAN based
- IP multicasting transport default for LAN
- TCP transport default for WAN

API

Professional Open Source™

Channel: conceptually similar to java.net.MulticastSocket
- plus group membership, reliability

Operations:
- Create a channel with a set of properties
- Connect to a group named "X". Everyone that connects to "X" will see each other
- Send a message to all members of X
- Send a message to a single member
- Receive a message
- Retrieve membership
- Be notified when members join, leave (including crashes)
- Disconnect from the group
- Close the channel

Message

Professional Open Source™

src, dest: Address
- IpAddress: IP address plus port
- src: filled in when sending (by JGroups)
- dest: null == send to all members of the group

buffer: byte[]

headers: HashMap of headers
- each protocol can add/remove its own headers
- example: sequence number for reliable retransmission

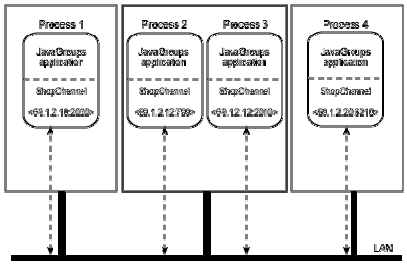
API

Professional Open Source™

```
JChannel channel=new JChannel("file://home/bela/config.xml");
channel.connect("demo-group");
System.out.println("members are: " +
channel.getView().getMembers());
Message msg=new Message(null, null, "Hello world");
channel.send(msg);
Message m=(Message)channel.receive(0);
System.out.println("received msg from " + m.getSrc() + ": " +
m.getObject());
ch.disconnect();
ch.close();
```


Group topology

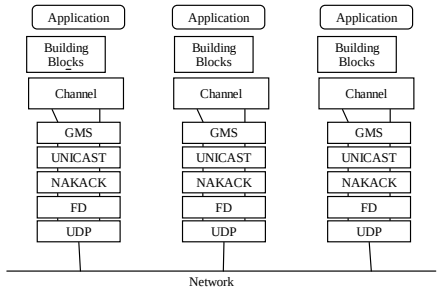
Professional Open Source™



JBoss The Professional Open Source Company

Professional Open Source™

Architecture of JGroups



3/8/2005

Stats

Professional Open Source™

JGroups has ~ 110KLOC
- 37KLOC protocols
- 34KLOC kernel
- 13 KLOC building blocks
- 26KLOC (unit) tests

~ 70 protocols shipped with JGroups

Set of well-tested stacks (in XML files)
- JGroups/conf

Available protocols I

Professional Open Source™

Transport
- UDP, TCP, TCP_NIO, TUNNEL, JMS, LOOPBACK

Discovery
- PING, TCPPING, TCPGOSSIP, UDPPING

Group membership

Reliable delivery & FIFO
- NAKACK, SMACK, UNICAST

Failure detection
- FD, FD_SOCK, FD_PID, FD_SIMPLE, FD_PROB, VERIFY_SUSPECT

Security
- ENCRYPT, SSL ConnectionTable (n/a)

Fragmentation (FRAG)

State transfer (STATE_TRANSFER)

Available protocols II

Professional Open Source™

- Ordering
  - FIFO, CAUSAL, TOTAL, TOTAL_TOKEN
- Virtual Synchrony
  - FLUSH, QUEUE, VIEW_ENFORCER
- Probabilistic Broadcast
  - PBCAST
- Merging:
  - MERGE(2), MERGEFAST
- Distributed message garbage collection
  - STABLE
- Debugging
  - PERF, TRACE, PRINTOBJS, SIZE, BSH
- Simulation
  - SHUFFLE, DELAY, DISCARD, DEADLOCK, LOSS, PARTITIONER

Available protocols III

Professional Open Source™

- Dynamic configuration
  - AUTOCONF
- Flow control
  - FLOW_CONTROL, FC
- Misc
  - Message bundling, COMPRESS

Transport

Professional Open Source™

- Task
  - Send messages from above to all members in the group, or to a single member
  - Receive messages from NW, pass up stack
- UDP: multicast and multiple UDP unicast
- TCP: mcast done by multiple TCP unicasts
- TUNNEL: send to external router, e.g. through firewall

Discovery

Professional Open Source™

- Task
  - Initial discovery of members
  - Used by GMS to determine coordinator to send JOIN request to
- Each member returns its own addr, plus the addr of the coordinator
  - Typical response ({A,A}, {B,A}, {C,A})
- Wait for n milliseconds or m responses
- Responses determine coordinator (A)
- Client then sends join request to A

Discovery - UDP

Professional Open Source™

- Multicast discovery request
- Each member responds with a unicast UDP datagram (local-addr, coord-addr), back to the sender

Discovery - TCPGOSSIP

Professional Open Source™

- Can be used by both UDP and TCP
- External GossipServer(s)
  - org.jgroups.stack.GossipServer
  - Maintains table of <group, members>
  - Each member registers (groupname, own addr)
  - Lease based - members have to periodically renew registration
  - Multiple GossipServers possible
- To obtain initial membership for a given group, TCPGOSSIP contacts the GossipServer(s)
- Membership info does not need to be accurate - only goal is to determine coord to send JOIN request to

Discovery - TCPPING

Professional Open Source™

Give a set of well known members
- <TCPPING timeout="3000" initial_hosts="host1[7800],host2[7800]" port_range="3" num_initial_members="3"/>
- We're sending discovery requests to host1 and host2 at port 7800
- We return as soon as
  - we have received responses from 3 members or
  - 3 seconds have elapsed

For discovery, those members are pinged

If at least 1 responds, we can find the coordinator

Does not require additional process

Group Membership

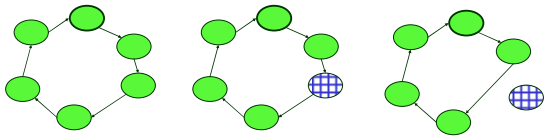
Professional Open Source™

Task
- Maintain a list of members
- Notify members when a new member joins, or an existing member leaves (or crashes)
- Each member has the same *ordered* list
- List can be retrieved by Channel.getView()
- First (= oldest) member is *coordinator*
- If coord crashes, 2nd oldest takes over

Failure detection

Professional Open Source™

Task
- Detect if a member has crashed and send SUSPECT event up the stack (to be handled by GMS)
- Logical ring over membership
  - Each member pings its neighbor to the right



Reliable delivery & FIFO

Professional Open Source™

Lossless and FIFO delivery for multicast and unicast messages
- Multicast: NAK and ACK
- Unicast: ACK

Missing messages (gaps) are retransmitted
- Sender resends or
- Receiver requests retransmission

Protocol stack configuration

Professional Open Source™

```
<config>
   <UDP mcast_send_buf_size="10000000" mcast_port="45566" ucast_recv_buf_size="10000000"
mcast_addr="228.8.8.8" loopback="false" mcast_recv_buf_size="10000000"
max_bundle_size="64000" max_bundle_timeout="30" use_incoming_packet_handler="false"
use_outgoing_packet_handler="true" ucast_send_buf_size="10000000" ip_ttl="32"
enable_bundling="true"/>
   <PING timeout="2000" down_thread="false" num_initial_members="3"/>
   <MERGE2 max_interval="10000" down_thread="false" min_interval="5000"/>
   <FD_SOCK down_thread="false"/>
   <VERIFY_SUSPECT timeout="1500" down_thread="false"/>
   <pbcast.NAKACK max_xmit_size="60000" down_thread="false" use_mcast_xmit="true"
gc_lag="50" retransmit_timeout="300,600,1200,2400,4800"/>
   <UNICAST timeout="300,600,1200,2400,3600" down_thread="false"/>
   <pbcast.STABLE stability_delay="1000" desired_avg_gossip="5000" down_thread="false"
max_bytes="250000"/>
   <pbcast.GMS print_local_addr="true" join_timeout="3000" down_thread="false"
join_retry_timeout="2000" shun="true"/>
   <FC max_credits="1000000" down_thread="false" min_threshold="0.10"/>
   <FRAG frag_size="60000" down_thread="false" up_thread="true"/>
   <COMPRESS down_thread="false" min_size="500" compression_level="3" up_thread="true"/>
   <pbcast.STATE_TRANSFER down_thread="false" up_thread="false"/>
</config>
```


Advantages of protocol stacks

Professional Open Source™

Each property is implemented by 1 protocol
- Fragmentation, retransmission, ordering

Protocols are assembled into a stack

Stack has exactly the properties needed by the appl / required by the network

Can't get this with java.net.Socket, always comes with full TCP/IP

Small scope: a protocol does just one job, but does it well

Protocol stacks are fashionable:
- Servlet 2.3 filters
- Interceptors (Corba, JBoss)
- AOP: separation of concerns, e.g. fragmentation should not be an application concern (JBossAop, AspectJ etc)

Application remains the same, stack can be changed

"JGroups is Aspect-Oriented Messaging Middleware"

Links

Professional Open Source™

http://www.jgroups.org
http://www.jboss.com/products/jgroups
Wiki: http://www.jboss.org/wiki/Wiki.jsp?page=JGroups

JBoss The Professional Open Source Company

Professional Open Source™

Questions ?