JGroups: A Toolkit for Reliable Multicasting

Bela Ban (JBoss Inc.)
Scott Marlow (Novell Inc.)

Agenda

- Overview
  - ✓ Purpose
  - ✓ API
  - ✓ Architecture
  - ✓ Protocols
- Performance

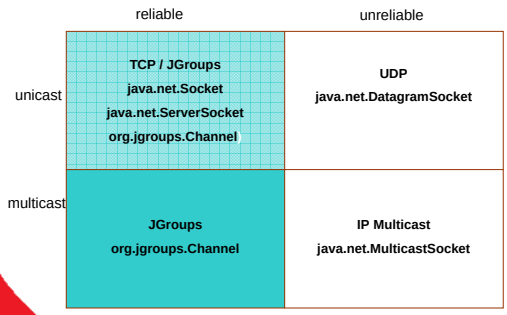
What is unreliable ?

- Messages get
  - ✓ dropped
    - too big (UDP has a size limit)
    - buffer overflow: receiver, switch, NIC, IP buffer
  - ✓ reordered
    - sending m1 --> m2, receiving m2 --> m1
- IP multicast doesn't know membership
  - ✓ node joins, leaves, crashes
- Fast sender overwhelms slower receivers (OOMs)

So what does JGroups buy me ?

- Library for reliable multicasting
  - ✓ Goal: to provide TCP's reliability for multiple receivers
- Provides
  - ✓ Fragmentation & retransmission
  - ✓ Flow control
  - ✓ Ordering
  - ✓ Cluster membership plus changes
- LAN or WAN based
  - ✓ IP multicasting transport default for LAN
  - ✓ TCP transport default for WAN

Comparison to java.net.*



API: Channel

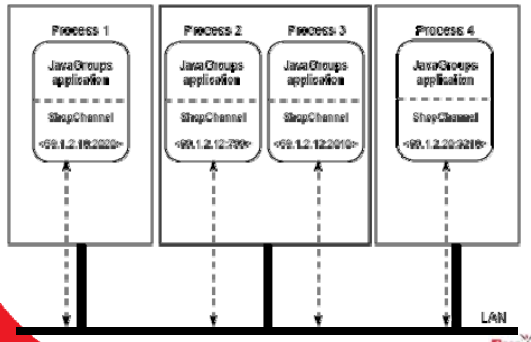
- Create a channel with a set of properties
- Join a cluster X
- Send a message to all nodes of X
- Send a message to a single node
- Receive messages
- Retrieve membership
- Notification on join, leave, crash of nodes
- Disconnect from cluster X
- Close the channel

API

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

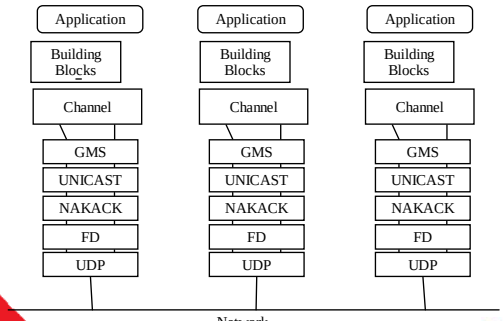
7

Group topology



8

Architecture



9

Overview of available protocols

10

Available protocols I

- Transport
  - ✓ UDP, TCP, TCP_NIO, LOOPBACK
- Discovery
  - ✓ PING, TCPPING, TCPGOSSIP, MPING
- Group membership (GMS)
- Reliable delivery & FIFO
  - ✓ NAKACK, UNICAST
- Failure detection
  - ✓ FD, FD_SOCK, VERIFY_SUSPECT

11

Available protocols II

- Security
  - ✓ ENCRYPT, AUTH
- Fragmentation (FRAG/FRAG2)
- State transfer (STATE_TRANSFER)
- Ordering
  - ✓ FIFO, TOTAL_TOKEN, SEQUENCER
- Merging:
  - ✓ MERGE(2), MERGEFAST
- Message garbage collection
  - ✓ STABLE

12

Available protocols III

- Flow control
  - ✓ FC
- Misc
  - ✓ Message bundling, COMPRESS
- Debugging
  - ✓ PERF, TRACE, PRINTOBJS, SIZE, BSH, STATS
- Simulation
  - ✓ SHUFFLE, DELAY, DISCARD, DEADLOCK, LOSS, PARTITIONER

13

Performance

- Cluster of M nodes
- N senders (0 < N <= M)
- Every sender sends X messages of Y bytes
- Senders send X, receive N * X msgs, compute message rate and throughput
- Lab
  - ✓ 4-8 nodes, Dell PE 1850, 2x 3Ghz procs, 4G ram, Linux 2.6
  - ✓ 1GB switch, 1.5K MTU (no jumbo frames)

14

UDP and TCP-NIO

- 2 nodes, 2 senders, 1M 1K msgs
  - ✓ UDP: 58684, TCP-NIO: 54793 msgs/sec
- 3 nodes, 3 senders, 1M 1K msgs
  - ✓ UDP: 49096, TCP-NIO: 46671 msgs/sec
- 4 nodes, 4 senders, 1M 1K msgs
  - ✓ UDP: 37192, TCP-NIO: 42776 msgs/sec
- 6 nodes, 6 senders, 1M 1K msgs
  - ✓ UDP: 31123, TCP-NIO: 40129 msgs/sec

15

Performance

- Replication gets more costly with increasing cluster size
  - ✓ Numbers still good (no jumbo frames)
    - Jumbo frames: more data with same rate
  - ✓ Buddy Replication - don‘t replicate to everybody
  - ✓ Performance drops slightly less in TCP-NIO than UDP
    - Switch dependent: some switches are hostile to broad- and multi-casts
    - Flow control more efficient in TCP
      - ✓ New version fixes this (JGRP-2)

16

TCP-NIO I

- Java New I/O support
  - ✓ Traditional Java socket i/o requires thread per server socket
  - ✓ Supports multiplexed, non-blocking i/o
  - ✓ Difficult to master but is worth the effort
  - ✓ Common NIO programming mistakes
    - Treating write operations as blocking
    - Causing “selector.register“ to block
  - ✓ Asynchronous support is planned for Java 7 release (JSR 203)

17

TCP-NIO II

- Scales up
  - ✓ Many socket connections per thread
  - ✓ Implementation avoids shared memory
- Highly configurable
  - ✓ reader_threads
  - ✓ writer_threads
  - ✓ processor_threads
  - ✓ processor_queueSize
  - ✓ Process requests in reader thread option

18

New stuff since 2.2.7

- 2.2.8
  - ✓ Fast message marshalling
    - 60% better compaction, 40% speedup
  - ✓ MPING
  - ✓ Concurrent startup
- 2.2.9 / 2.2.9.1
  - ✓ JMX support
  - ✓ Receiver
    - Push style for message reception
  - ✓ TCP-NIO

New stuff since 2.2.7

- 2.3
  - ✓ AUTH
    - Prevents rogue nodes from joining cluster
  - ✓ SEQUENCER
    - Total order
  - ✓ Multiplexer
    - Multiple apps on the same channel
    - More efficient use of resources
    - Faster startup time for JBossAS
  - ✓ Partial state transfer

Planned

- 2.4
  - ✓ Streaming state transfer
  - ✓ FLUSH (virtual synchrony)
  - ✓ Threadless stack and OOB messages
    - using customizable threadpools (util.cc)
  - ✓ Logical addresses
- 2.5 and beyond
  - ✓ TOTAL-TOKEN
  - ✓ UNIFORM
  - ✓ Use of Apache Portable Runtime (APR)
  - ✓ Memory based flow control

Links

- www.jboss.com/products/jgroups
- http://www.jgroups.org/javagroupsnew/docs/Perftest.html