Script generated by TTT

Title: Distributed_Applications (17.06.2013)

Date: Mon Jun 17 09:11:02 CEST 2013

Duration: 46:16 min

Pages: 17

JGroups is a reliable group communication toolkit written in Java. It is based on IP multicast and extends it with

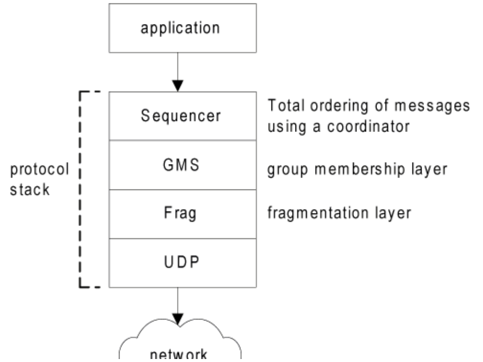
- reliability, especially ordering of messages and atomicity.
- management of group membership.

Programming Interface of JGroups

groups are identified via channels.

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

a channel is connected to a protocol stack specifying its properties.



Introduction

Group communication facilities the interaction between groups of processes.

JGroups

JGroups is a reliable group communication toolkit written in Java. It is based on IP multicast and extends it with

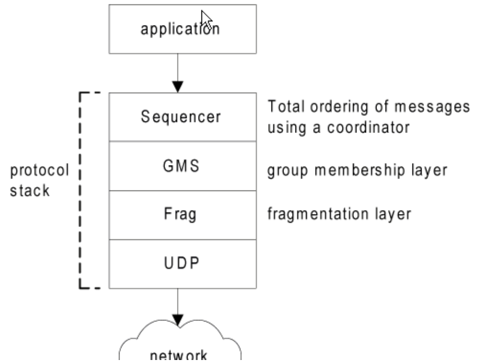
- reliability, especially ordering of messages and atomicity.
- management of group membership.

Programming Interface of JGroups

groups are identified via channels.

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

a channel is connected to a protocol stack specifying its properties.



management of group membership.

Programming Interface of JGroups

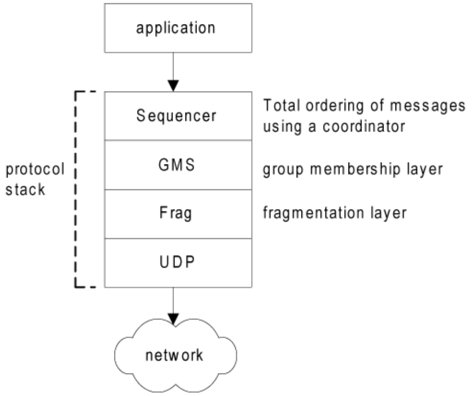
channels behave like sockets

groups are identified via channels.

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

asynchronous message sending receiving

a channel is connected to a protocol stack specifying its properties.



[Code Example](#)


```
String props = "UDP:Frag:GMS:causal";
Message send_msg;
Object recv_msg;
Channel channel = new JChannel(props);
channel.connect("MyGroup");
send_msg = new Message(null, null, "hello World");
channel.send(send_msg);
recv_msg = (Message) channel.receive(0);
System.out.println("Received " + recv_msg);
channel.disconnect();
channel.close();
```


JGroups is a reliable group communication toolkit written in Java. It is based on IP multicast and extends it with

reliability, especially ordering of messages and atomicity.

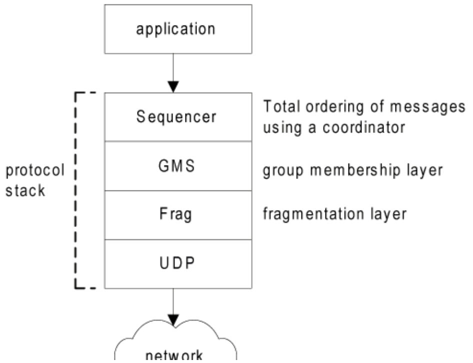
management of group membership.

Programming Interface of JGroups

groups are identified via channels.

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

a channel is connected to a protocol stack specifying its properties.



problem of distributed processes to agree on a value; processes communicate by message passing.

Examples

all correct computers controlling a spaceship should decide to proceed with landing, or all of them should decide to abort (after each has proposed one action or the other)

in an electronic money transfer transaction, all involved processes must consistently agree on whether to perform the transaction (debit and credit), or not

desirable: reaching consensus even in the presence of faults

assumption: communication is reliable, but processes may fail

[Consensus Problem](#)

[Consensus in synchronous Networks](#)

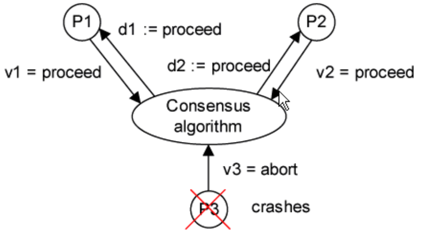
Consensus Problem

agreement on the value of a ***decision variable*** amongst all correct processes

p_i is in state undecided and proposes a single value v_i , drawn from a set of values.

next, processes communicate with each other to exchange values.

in doing so, p_i sets decision variable d_i and enters the decided state after which the value of d_i remains unchanged



[Properties]
[Algorithm]
[The Byzantine Generals Problem]
[Interactive Consistency Problem]
[Relationship between these Problems]

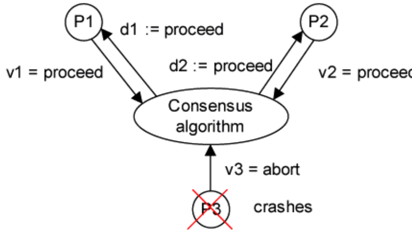
Consensus Problem

agreement on the value of a ***decision variable*** amongst all correct processes

p_i is in state undecided and proposes a single value v_i , drawn from a set of values.

next, processes communicate with each other to exchange values.

in doing so, p_i sets decision variable d_i and enters the decided state after which the value of d_i remains unchanged



[Properties]
[Algorithm]
[The Byzantine Generals Problem]
[Interactive Consistency Problem]
[Relationship between these Problems]

Generated by Targeteam

Properties

The following conditions should hold for every execution of the algorithm:

termination : eventually, each correct process sets its decision variable

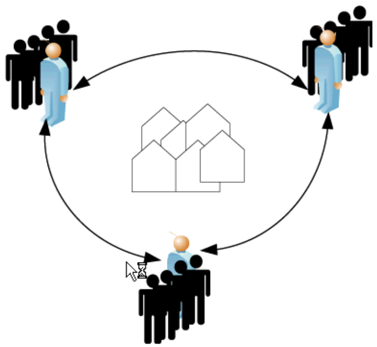
agreement : the decision variable of all correct processes is the same in the decided state.

integrity : if the correct processes all proposed the same value, then any correct process has chosen that value in the decided state.

The Byzantine Generals Problem

three or more generals are to agree to attack or to retreat.

- one general, the commander issues order
  - others (lieutenants to the commander) have to decide to attack or retreat
- one of the generals may be treacherous
  - if commander is treacherous, it proposes attacking to one general and retreating to the other
  - if lieutenants are treacherous, they tell one of their peers that commander ordered to attack, and others that commander ordered to retreat

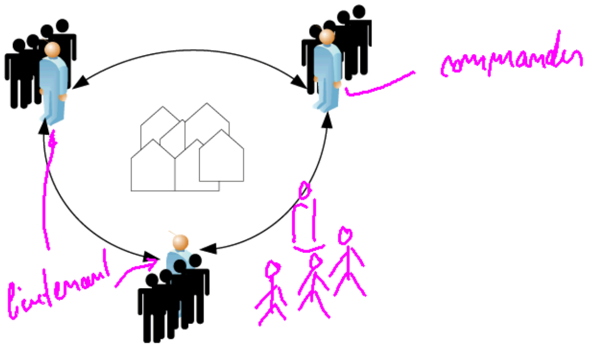


difference to consensus problem: one process supplies a value that others have to agree on

properties:

The Byzantine Generals Problem

if lieutenants are treacherous, they tell one of their peers that commander ordered to attack, and others that commander ordered to retreat



difference to consensus problem: one process supplies a value that others have to agree on

properties:

- termination: eventually each correct process sets its decision variable.
- agreement: the decision value of all correct processes is the same.
- integrity: if the commander is correct, then all processes decide on the value that the commander proposes.

Interactive Consistency Problem

Each process suggests a single value.

goal : all correct processes agree on a vector of values ("decision vector"); each component correspond to one processes' agreed value

example: agreement about each processes' local state.

properties:

- termination: eventually each correct process sets its decision vector.
- agreement: the decision vector of all correct processes is the same.
- integrity: if p_i is correct, then all correct processes decide on v_i as the i-th component of their vector.

Relationship between these Problems

Assume that the previous problems could be solved, yielding the following decision variables

Consensus : C_i (v_1 ,.., v_n) returns the decision value of process p_i

Byzantine Generals : BG_i (k, v) returns the decision value of process p_i where p_k is the commander which proposes the value v

Interactive Consistency : IC_i (v_1 ,.., v_n)[k] returns the k-th value in the decision vector of process p_i where v_1 ,.., v_n are the values that the processes proposed

Possibilities to derive solutions out of the solutions to other problems

solution to ***IC from BG***

run BG n times, once with each p_i acting as commander

IC_i (v_1 ,.., v_n)[k] = BG_i (k, v_k) with (i, k = 1, .., n)

solution to ***C from IC***

run IC to produce a vector of values at each process

apply an appropriate function on the vector's values to derive a single value

C_i (v_1 ,.., v_n) = majority(IC_i (v_1 ,.., v_n)[1],.., IC_i (v_1 ,.., v_n)[n])

solution to ***BG from C***

commander p_k sends its proposed value v to itself and each of the remaining processes

all processes run C with the values v_1 ,.., v_n that they receive

Relationship between these Problems

Byzantine Generals : BG_i (k, v) returns the decision value of process p_i where p_k is the commander which proposes the value v

Interactive Consistency : IC_i (v_1 ,.., v_n)[k] returns the k-th value in the decision vector of process p_i where v_1 ,.., v_n are the values that the processes proposed

Possibilities to derive solutions out of the solutions to other problems

solution to ***IC from BG***

run BG n times, once with each p_i acting as commander

IC_i (v_1 ,.., v_n)[k] = BG_i (k, v_k) with (i, k = 1, .., n)

solution to ***C from IC***

run IC to produce a vector of values at each process

apply an appropriate function on the vector's values to derive a single value

C_i (v_1 ,.., v_n) = majority(IC_i (v_1 ,.., v_n)[1],.., IC_i (v_1 ,.., v_n)[n])

solution to ***BG from C***

commander p_k sends its proposed value v to itself and each of the remaining processes

all processes run C with the values v_1 ,.., v_n that they receive

derive BG_i (k, v) = C_i (v_1 ,.., v_n) with i = 1, .., n

termination, agreement and integrity preserved in each case.

Assumption : no more than f of the n processes crash (f < n).

The algorithm proceeds in f+1 rounds in order to reach consensus.

the processes B-multicast values between them.

at the end of f+1 rounds, all surviving processes are in a position to agree.

algorithm for process $p_i \in$ concensus group g

```
On initialization
    values_i (1) := {v_i }; values_i (0) := {};

in round r (1 ≤ r ≤ f+1)
    B-multicast(g, values_i (r)-values_i (r-1));
       //send only values that have not been sent
    values_i (r+1) := values_i (r)
    while (in round r) {
       On B-deliver(v_j ) from some p_j
          values_i (r+1) := values_i (r+1) ∪ v_j
    }

After (f+1) rounds
    assign d_i = minimum (values_i (f+1))
```


Issues

The following section discusses several important basic issues of distributed applications.

Data representation in heterogeneous environments.

Discussion of an execution model for distributed applications.

What is the appropriate error handling?

What are the characteristics of distributed transactions?

What are the basic aspects of group communication (e.g. algorithms used by ISIS) ?

How are messages propagated and delivered within a process group in order to maintain a consistent state?