HS 2009 Prof. Dr. Roger Wattenhofer, Thomas Locher, Remo Meier, Benjamin Sigg

Distributed Systems
Theory exercise 1

Assigned: November 6, 2009
Discussion: November 13, 2009

1 Shared memory vs. message passing

1.1 Comparison

Shared memory allows multiple processes to read and write data from the same location. *Message passing* is another way for processes to communicate: each process can send messages to other processes.

Make a comparison between shared memory and message passing: where are they different and where are the similar? You might consider different models of message passing, for example with or without message loss.

1.2 Examples

Consider the actions described below, in which model (shared memory or message passing) can you describe them best? Why and how?

- Communication via postcards
- Two people speaking in a room
- Instant messages via Skype (data remains on client if partner is offline)
- Many people speaking in a room

2 Writing to multiple registers at the same time

A *n-register* allows up to n registers to be written at the same time. Processes may still only read one value at a time. Let $n = 6$, give a protocol which solves consensus for 3 processes. You may assume the registers are initialized with -1 and processes have a unique id.

Hints: You don't need more than 6 registers (or one 6-register). You don't need to write into *all* registers, you can write into a subset (e.g. you can atomically write into 3 registers). Compare pairs of processes, find out which process is the fastest.

3 Analyzing a protocol

A lousy programmer wanted to solve consensus for 2 processes and came up with a sophisticated protocol. Does the protocol really solve consensus? Why?

```
initialize(){
  // s is shared
  s = '?';

  // i is also shared
  i = 0;

  // the input, an array of length 2
  input[] = [random({0,1}), random({0,1})];
}

// making the decision
decide(){
  // the id of this process, 0 or 1
  id = this.getThreadId();

  decisionMade = false;
  decision = input[ id ];

  while( decisionMade == false ){
    value = s; // read s
    if( value == '?' ){
      s = input[ id ]; // write s
      decision = input[ id ];
    }
    else if( value != decision ){
      decisionMade = true;
      decision = value;
    }
    else{
      if( i.fetchAndInc() == 1 ){
        decision = input[ 1-id ];
      }
      decisionMade = true;
    }
  }
}
```