TSR

This exam consists of 20 multiple choice questions. In every case only one answer is correct. You should answer in a separate sheet. If correctly answered, they contribute 0.5 points to the exam grade. If incorrectly answered, the contribution is negative: -0.167. So, think carefully your answers.

THEORY

1. This is not one of the "main reasons" for using distributed systems:

a	To improve the efficiency of their applications, splitting the problem to be solved in multiple pieces, and running multiple agents in different computers to solve it.
b	To achieve failure transparency.
c	To allow resource sharing, especially when those devices are expensive devices that may be remotely accessed.
d	To make application deployment as easy as possible.

2. In cloud systems, hardware virtualisation technology is a basic mechanism for this service model:

a	SLA.
b	SaaS.
c	IaaS.
d	Message-based middleware.

3. In the SaaS model in cloud systems, this statement is true:

a	Its services may be locally accessed, without any network intervention, using hardware virtualisation.
b	Its client-server interactions must be based on an asynchronous message-oriented middleware.
c	It provides distributed software services to its customers, in some cases under a pay-as-you-go model.
d	Service users decide how the software must be deployed.

4. Unit 2 recommends the asynchronous programming paradigm because that paradigm…

a	…makes application deployment trivial.
b	…is event-based and ensures the atomic execution of each action.
c	…provides failure transparency.
d	…is based on reverse proxies and, because of this, is highly scalable.

5. Regarding the aspects of synchrony described in Unit 2, it is true that:

a	Logical clocks ensure process synchronisation.
b	Synchronous message order is based on setting bounds on message propagation time.
c	Synchronous processes progress in steps. At each step, every process completes an action.
d	Synchronous communication requires that channels keep sent messages until the receiver is ready for accepting them.

6. These sentences relate middleware and standards. What is false?

a	The usage of standards facilitates that middleware and agent implementations from different companies interoperate.
b	The usage of standards provides a high-level interface in middleware layers. With this, programming tasks become easier.
c	Middleware APIs may be non-standard. ZeroMQ is an example of this kind.
d	Middleware layers cannot comply with any standard, since standards are only defined for internal elements of the operating system kernel.

7. Which of these communication-related elements can be considered an example of middleware?

a	The IP protocol.
b	A distributed naming service.
c	TCP.
d	The APACHE server.

8. In the context of middleware examples, which are the problems of distributed object systems when they are compared with messaging systems?

a	Their potential high coupling that could lead to a blocking behaviour when shared resources are concurrently used by many agents.
b	They are not location transparent.
c	They provide a low level of abstraction, complicating the resulting programs.
d	Their behaviour is highly asynchronous, being almost impossible to debug.

TSR

SEMINARS

9. Considering this program:

```
var fs=require('fs');
if (process.argv.length<5) {
  console.error('More file names are needed!!');
  process.exit();
}
var files = process.argv.slice(2);
var i=-1;
do {
  i++;
  fs.readFile(files[i], 'utf-8', function(err,data) {
      if (err) console.log(err);
      else console.log('File '+files[i]+': '+data.length+' bytes.');
  })
} while (i<files.length);
console.log('We have processed '+files.length+' files.');
```

This sentence is true if we assume that no error aborts this program execution and sufficient file names have been given as arguments:

a	Due to the readFile() callback asynchrony, this program is unable to show in each iteration the intended file name and size.
b	It prints the name and length for each one of the files passed as command-line arguments.
c	It prints "We have processed 0 files" as its first message.
d	It discards some file names given as arguments to this program, after the "node program-name" elements.

10. Regarding the program shown in the previous question...

a	It needs multiple turns for completing its execution, since each file being read requires a turn for its callback.
b	The "i" increase (i.e., i++) is incorrectly placed. It must be inside the callback body.
c	This program shows an error and terminates if fewer than five file names have been passed as arguments.
d	It prints the same length in all iterations. We need a closure for preventing this faulty behaviour from happening.

11. Regarding the mutual exclusion algorithms seen in Seminar 2, it is true that...

a	The central server algorithm correctly manages the situations in which that central server fails.
b	The virtual unidirectional ring algorithm does not lose the token if the current process in the critical section fails.
c	The multicast algorithm with logical clocks uses fewer messages than the multicast algorithm based on quorums.
d	The multicast algorithm with logical clocks complies with all 3 mutual exclusion correctness conditions.

12. **Considering this program and knowing that it does not generate any error...**

```
var ev = require('events');
var emitter = new ev.EventEmitter;
var num1 = 0;
var num2 = 0;
function myEmit(arg) { emitter.emit(arg,arg) }
function listener(arg) {
   var num=(arg=="e1"?++num1:++num2);
   console.log("Event "+arg+" has happened " + num + " times.");
   if (arg=="e1") setTimeout( function() {myEmit("e2")}, 3000 );
}

emitter.on("e1", listener);
emitter.on("e2", listener);
setTimeout( function() {myEmit("e1")}, 2000 );
```

The following sentence is true:

a	Event “e1” happens only once, 2 seconds after this program is started.
b	Event “e2” never happens.
c	Event “e2” happens periodically and its period is three seconds.
d	Event “e1” happens periodically and its period is two seconds.

13. **Considering the program shown in the previous question...**

a	The first event “e2” happens three seconds after the program is started.
b	Since both events use the same listener, they both print messages with exactly the same contents when they happen.
c	The first event “e2” happens two seconds after the first event “e1” has happened.
d	None of its events happens twice or more times.

14. **The ØMQ REQ-REP communication pattern is considered synchronous because...**

a	Both sockets are connected or bound to the same URL.
b	Both sockets are bidirectional.
c	The REP socket uses a synchronous method for handling the received messages.
d	Once a message A is sent, both sockets cannot transmit their next sent message until they have received A’s reply (REQ) or a new request (REP).

15. Considering these two node.js programs...

```
// server.js
var net = require('net');
var server = net.createServer(
 function(c) {//'connection' listener
  console.log('server connected');
  c.on('end', function() {
   console.log('server disconnected');
  });
  c.on('data', function(data) {
   console.log('Request: ' +data);
   c.write(data+ 'World!');
  });
});
server.listen(9000);
```

```
// client.js
var net = require('net'); var i=0;
var client = net.connect({port:
  9000}, function() {
    client.write('Hello ');
  });
client.on('data', function(data) {
  console.log('Reply: '+data);
  i++; if (i==1) client.end();
});
client.on('end', function() {
  console.log('client ' +
    'disconnected');
});
```

The following sentence is true:

a	The server terminates after sending its first reply to the first client.
b	The client never terminates.
c	This server may manage multiple connections.
d	This client cannot connect to this server.

16. Leader election algorithms (from Seminar 2)...

a	…require the execution of a mutual exclusion algorithm, since their leader identity is placed in a shared resource and can only be updated by a critical section owner.
b	…require consensus among all participating processes: all they must choose the same leader.
c	…do not need unique process identities.
d	...must respect causal order.

17. We want to implement a leader election algorithm using NodeJS and ØMQ, using the first algorithm explained in Seminar 2: the virtual ring algorithm. In order to implement this service, choose the best option:

a	Each process uses a REQ socket for sending messages to its successor in the ring and a REP socket for receiving messages from its predecessor.
b	Each process uses a ROUTER socket for sending messages to its successor in the ring and a DEALER socket for receiving messages from its predecessor.
c	Each process uses a SUB socket for sending messages to its successor in the ring and a PUB socket for receiving messages from its predecessor.
d	Each process uses a PUSH socket for sending messages to its successor in the ring and a PULL socket for receiving messages from its predecessor.

18. We want to implement a leader election algorithm in NodeJS and ØMQ, using the second algorithm (bully) explained in Seminar 2. In order to implement the "election" (to ask better candidates about their liveness) and "reply" (to answer those "election" messages, confirming its liveness) messages of that algorithm, a feasible alternative for each participating process could be, assuming N processes:

a	A single connected REQ socket for sending the "election" messages to the other N-1 processes and receiving their "reply" answers and a single bound REP socket for managing the other communication side.
b	A single DEALER socket for sending "election" and "reply" messages to the other N-1 processes. That same socket is also used for receiving the incoming messages.
c	N-1 PUSH sockets for sending both "election" and "reply", when needed. A single SUB socket for receiving and handling those messages.
d	A single bound PULL socket for receiving messages. N-1 PUSH sockets connected to the PULL of the other agents, for sending both "election" and "reply", when needed.

19. Which is the ØMQ socket type that uses multiple outgoing queues?

a	PUB sockets, for managing their broadcasts.
b	PUSH sockets, for managing multiple asynchronous unicast send() operations.
c	REQ sockets, in case of being connected to multiple REP sockets.
d	ROUTER sockets, using one outgoing queue per connection.

20. Considering these programs...

```
//client.js
var zmq=require('zmq');
var rq=zmq.socket('req');
rq.connect('tcp://127.0.0.1:8888');
rq.connect('tcp://127.0.0.1:8889');
for (var i=1; i<=100; i++) {
  rq.send(''+i);
  console.log("Sending %d",i);
}
rq.on('message',function(req,rep){
  console.log("%s: %s",req,rep);
});
```

```
// server.js
var zmq = require('zmq');
var rp = zmq.socket('rep');
var port = process.argv[2] || 8888;
rp.bindSync('tcp://127.0.0.1:'+port);
rp.on('message', function(msg) {
  var j = parseInt(msg);
  rp.send([msg,(j*3).toString()]);
});
```

…and assuming that we have started one client and two servers using these commands:
$ node client & node server 8888 & node server 8889 &
The following sentence is true:

a	One server receives all requests with even values for "i" while the other receives all requests with odd values for "i".
b	Every server receives, manages and replies all 100 requests. Thus, the client receives and prints 200 replies.
c	The first few requests are lost since the client has been started before the first server was started.
d	If one of the servers fails in the middle of the execution, the client and the other server are able to manage without blocking all their remaining requests and replies.