Byzantine Agreement - Disposition

Lukas Peter Jørgensen, 201206057, DA4

26. marts 2014

1 Fault tolerance

Dependability

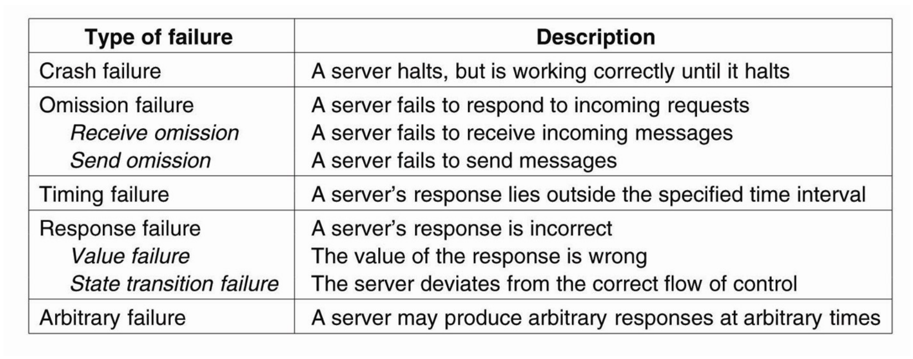
Fault tolerant is strongly related dependable systems.
Dependability is a term that covers a number of useful requirements for distributed systems including:

1. Availability
2. Reliability
3. Safety
4. Maintainability

Availability is defined as whether the system is ready to be used immediately.
Reliability is defined as whether a system can run continuously without failure.
Safety is defined as whether something catastrophic happens if the system temporarily fails to operate correctly.
Maintainability is defined as how easy it is to repair a failed system.
The cause of an error is called a fault, a fault might be something simple as overheated-hardware or something more complex as bad weather.

Fault types

Transient	occurs once and then dissappears (e.g. a bird flying through a microwave transmitter)
Intermittent	occurs, then vanishes for no reason, then reappears etc. Could be a loose contact for example.
Permanent	occurs and doesn't go away, could be software bug, overloaded hardware etc.



Type of failure	Description
Crash failure	A server halts, but is working correctly until it halts
Omission failure	A server fails to respond to incoming requests
Receive omission	A server fails to receive incoming messages
Send omission	A server fails to send messages
Timing failure	A server's response lies outside the specified time interval
Response failure	A server's response is incorrect
Value failure	The value of the response is wrong
State transition failure	The server deviates from the correct flow of control
Arbitrary failure	A server may produce arbitrary responses at arbitrary times

Byzantine failures

Also known as arbitary failures, the server might be spitting out arbitrary output, or worse yet it might do it intentionally.

2 Redundancy masking

If a system is to be fault tolerant, the best way is to try to hide the occurrence of failures from other processes.
Redundancy is the key technique to achieve this, there are 3 kinds of redundancies:

- Information redundancy
- Time redundancy
- Physical redundancy

Information redundancy

With information redundancy, you can avoid some issues by providing extra information, e.g. by sending some extra bits to allow recovery from garbled bits.

Time redundancy

With time redundancy, an action is performed and then if need be it is performed again.

Physical redundancy

Add more machines, so you can allow more of them to crash without comprimising the system.

3 Two-army problem

Agreement not possible.

4 Byzantine generals problem

$2k + 1$ correctly functioning processes must be present.
They only have to reach consensus on the nonfaulty values.