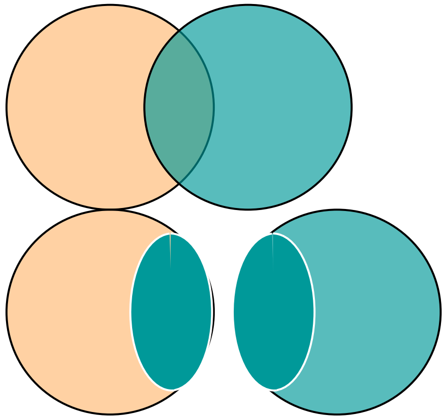
COMP90050:Advanced Database Systems

Professor Rao Kotagiri

Lecture Set2

Fault Tolerance



P(A) = probability of an event A is happening in **certain period.**

P(A and B) = probability both events A and B happening in that period = P(A)*P(B) assuming A and B are independent events.

P(A or B) = P(A) + P(B) - P(A and B)

Assuming A and B are independent.

= P(A) + P(B) - P(A) * P(B)

= P(A) + P(B) if P(A) and P(B) are very small.

Mean time to event MT(A) = 1/P(A)

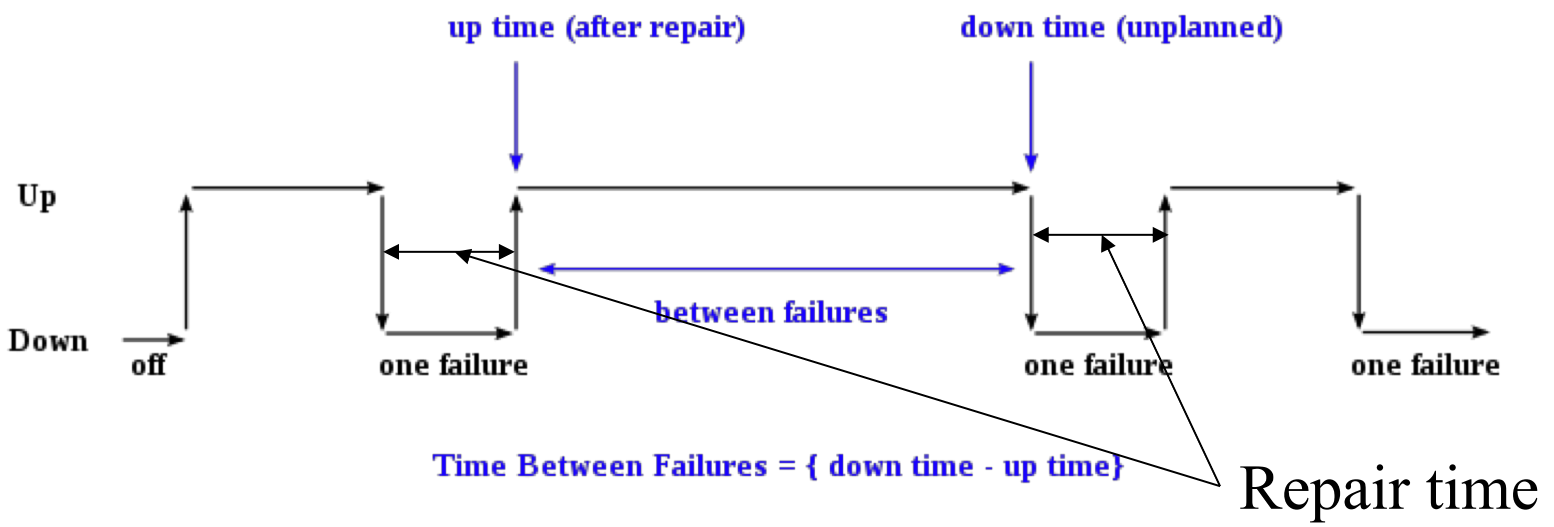
Fault Tolerance ...

If events A, B, have mean time MT(A), MT(B), then the mean time to the first event is

1/(P(A) + P(B) – P(A,B)), assuming P(A) and P(B) are very small we can assume P(A,B) $\cong$ 0.

= 1/(P(A) + P(B)).

Fault Tolerance ...



Module availability : measures the ratio of service accomplishment to elapsed time.

$$= \frac{\text{Mean time to failure}}{\text{Mean time to failure + mean time to repair}}$$

Fault Tolerance ...

If all n different events have same mean time m the Mean time to the first one of the events = m/n

$$if\ p\ is\ the\ probabily\ of\ an\ event\ in\ given\ time,$$
$$the\ mean\ time\ m = \frac{1}{p}$$

$$if\ there\ are\ n\ events\ proabaility\ of\ one\ of\ these\ events$$
$$= n * p\ assuming\ p\ is\ very\ small$$

$$There\ fore\ mean\ time\ to\ one\ of\ these\ events = \frac{1}{n * p}$$
$$= \left(\frac{1}{\mathrm{p}}\right) * \left(\frac{1}{\mathrm{n}}\right) = \frac{\mathrm{m}}{\mathrm{n}}$$

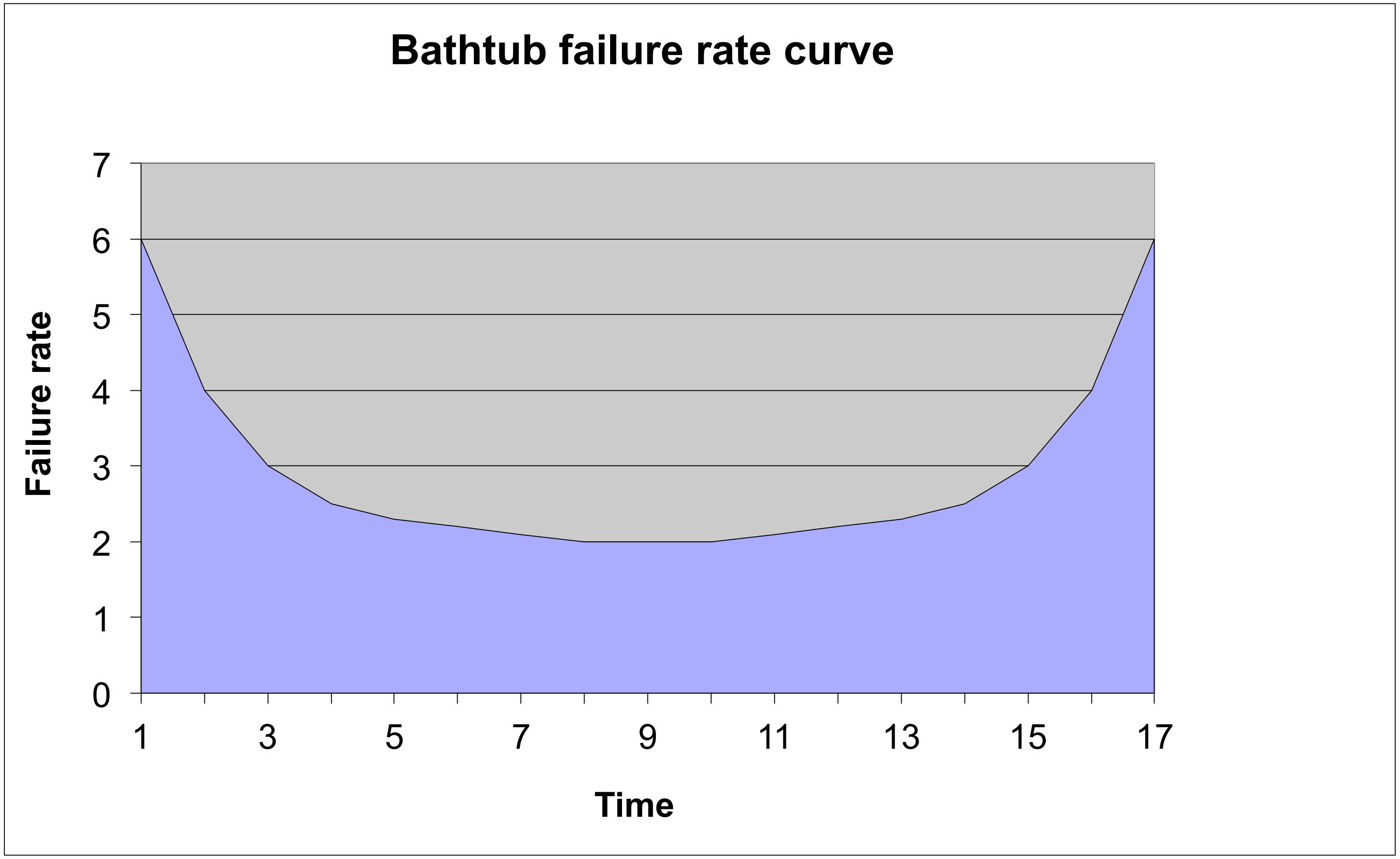
Fault Tolerance ...

Sys Type	Unavailability minutes/year	Availability	Class
Unmanaged	52 560 (~37 days)	90%	1
Managed	5 256 (3.7 days)	99%	2
Well managed	526 (~9 Hours)	99.9%	3
Fault tolerant	53 (~1 hour)	99.99%	4
Ultra availability	0.05 (3 Secs)	99.99999%	7

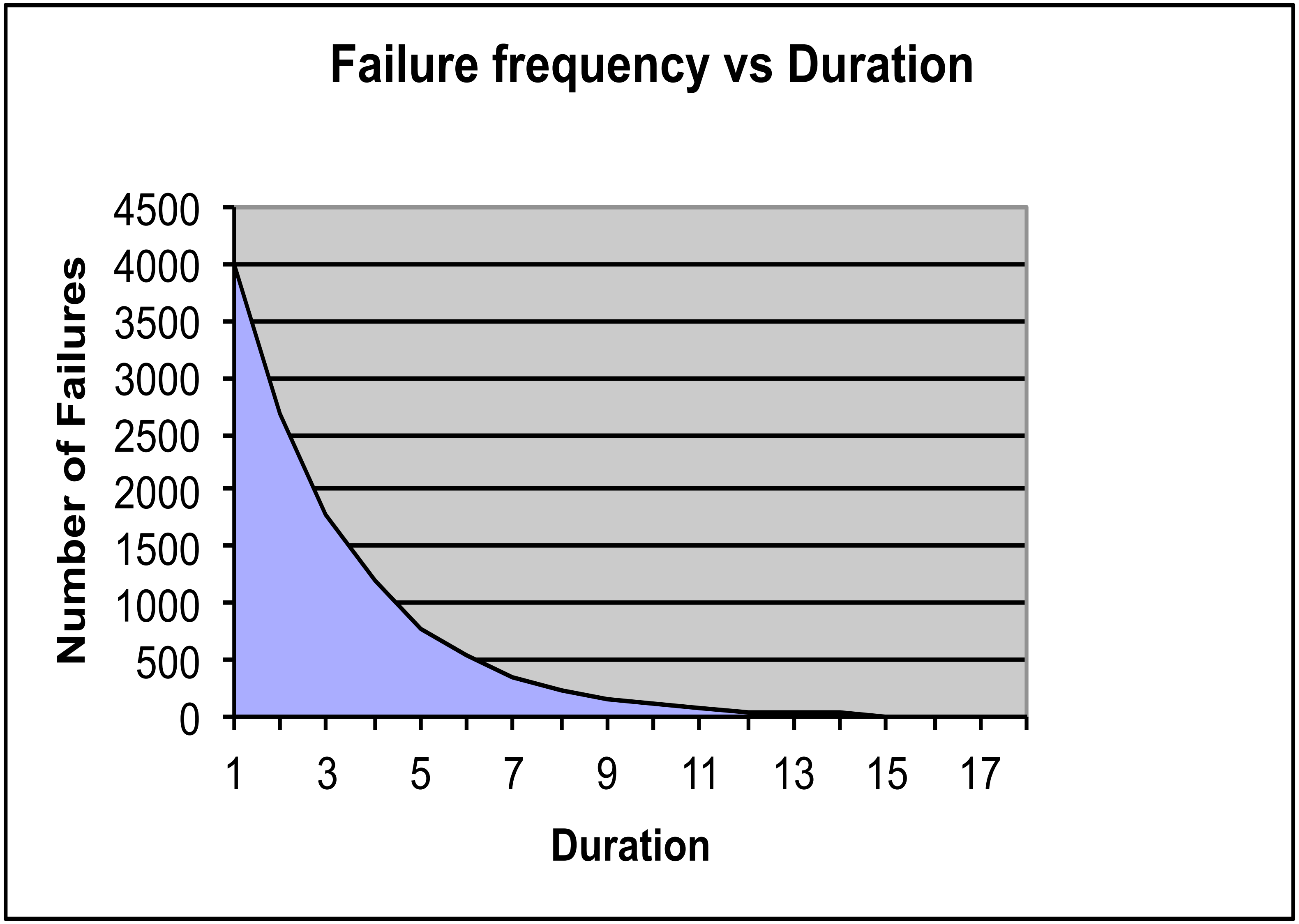
Fault Types

- Environmental : failures such as cooling, power, weather, data communication lines, fires, earthquakes, tsunami, wars, sabotage
- Operational: system administration, system configuration and system operation procedures
- Maintenance: procedures for regular maintenance, replacement of hardware on regular basis
- Hardware: devices, cooling
- Software: programs
- Process: strikes, management decisions for shutdowns
- Civil Wars

Failure Rates



Failure Rates



Disk failure rates

Type of Error	MTTF	Recovery	Consequence
soft data	1 h	Retry	none
seek	6 h	Retry	none
Maskable hard data	3 d	ECC (Error Correcting Coding)	remap
Unmaskable hard data	1 y	none	remap
repair	5 y	repair	data unavailable

Category of faults (Japanese Study 1985)

Category of Fault	Mean Time To Failure (MTTF)
Vendor (Hardware, Software, Maintenance)	6 months
Application	8 months
Telecommunication Lines	2 years
Operations	2 years
Environment	2 years
Any of the above faults	10 weeks

Fault Tolerance

- Failvote - use two are more modules and compare their outputs. Stops if there are no majority outputs agreeing. It fails twice as often with duplication but gives clean failure semantics
  - With triplication the system has MTTF of 5/ 6 of single module = 1/3+1/2 = 0.83 (Let M is MTTF of single module. MTTF any one the modules in triplicate system is M/3 and then MTTF one of two working modules is M/2. Giving overall MMTF of M*(1/3+1/2) = 0.83M)
- Failfast (voting)- this scheme is similar to fail voting except the system senses which modules are available and then uses the majority of the available modules.
  - A 10-plex module system continues to operate until the failure of 9 modules where as Failvote stops when 5 modules fail.
  - Failfast voting system has better reliability than failvote voting (since it stops when there is no majority agreement).

Fault Tolerance ...

- E.g. Consider a system with modules each with MTTF of 10 years including soft faults.
- With a paired (Duplex) system with Failvote MTTF = 10/2 = 5 years.
- With triplex system is 10/3 for the first failure + 10/2 for the 2nd failure = 8.3 years.
- If triplex system can mask off all transient faults and if the MTTF of hard faults is100 years, the system MTTF will be 100*(1/3 + 1/2) = 83.7 years.

Fault Tolerance ...

Failvote needs majority to agree to accept an action (Read/write for e.g.)

- In Triplication system we have 3 devices we need at least 2 agree to proceed with the action.
- devices not working we have no majority decision to make and system stops. Scenario A: All the devices working all the three agree -- accept the action(3 out 3 agree)
- Scenario B: two of the devices working and those two agree -- accept the action (2 out of three agree)
- Scenario C: two of the devices working and those two disagree -- do not accept the action as we do not have 2 out of 3 agreeing.

- If we start with 10 devices. We can keep going as long as 6 of them working and agreeing on the decision.
- The moment 5th one fails the system stops as there no 6 devices agreeing.

Fault Tolerance ...

In Failfast, we are only concerned of majority among the working ones. We are assuming that we can tell which ones are working. Hence we can continue to operate until 2 working ones and if both agree 2 out 2 and we can proceed with the action. But if they differ we do accept the action.

- 0 devices are faulty, we have 10 working and we need at least 6 to agree
- 1 device is faulty, we have 9 working and we need at least 5 to agree
- 2 devices are faulty, we have 8 working and we need at least 5 to agree
- 3 devices are faulty, we have 7 working and we need at least 4 to agree
- 4 devices are faulty, we have 6 working and we need at least 4 to agree
- 5 devices are faulty, we have 5 working and we need at least 3 to agree
- 6 devices are faulty, we have 4 working and we need at least 3 to agree
- 7 devices are faulty, we have 3 working and we need at least 2 to agree
- 8 devices are faulty, we have 2 working and we need both to agree
- 9 devices are faulty, we have 1working and we have to stop as we have nothing to compare with!

Fault Tolerance ...

N-plex repair- in this configuration the faulty equipment is repaired with an average time of MTTR (mean time to repair) as soon as a fault is detected. (Some times MTTR just time needed to replace)

Typical Values for recent disks:

MTTR = Few hours (assuming we stock spare disks) to 1 Day

MTTF = 750000 hours (~ 86 years)

Probability a particular module is not available

$= MTTR/(MTTF+MTTR)$

$\cong MTTR/MTTF \quad \text{if } MTTF >> MTTR$

Probability of (n-1) modules unavailable = $p_{n-1} = \left(\frac{MTTR}{MTTF}\right)^{(n-1)}$

Probability a module N fails P_f = 1/MTTF

Fault Tolerance ...

Probability that the system fails with the particular ith module failing last =

$$P_f * P_{n-1} = \left(\frac{1}{MTTF}\right)\left(\frac{MTTR}{MTTF}\right)^{(n-1)}$$

Probability that N-plex system fails = $\left(\frac{n}{MTTF}\right)\left(\frac{MTTR}{MTTF}\right)^{(n-1)}$

MTTF of N-plex system = $\left(\frac{MTTF}{n}\right)\left(\frac{MTTF}{MTTR}\right)^{(n-1)}$

Fault Tolerance ...

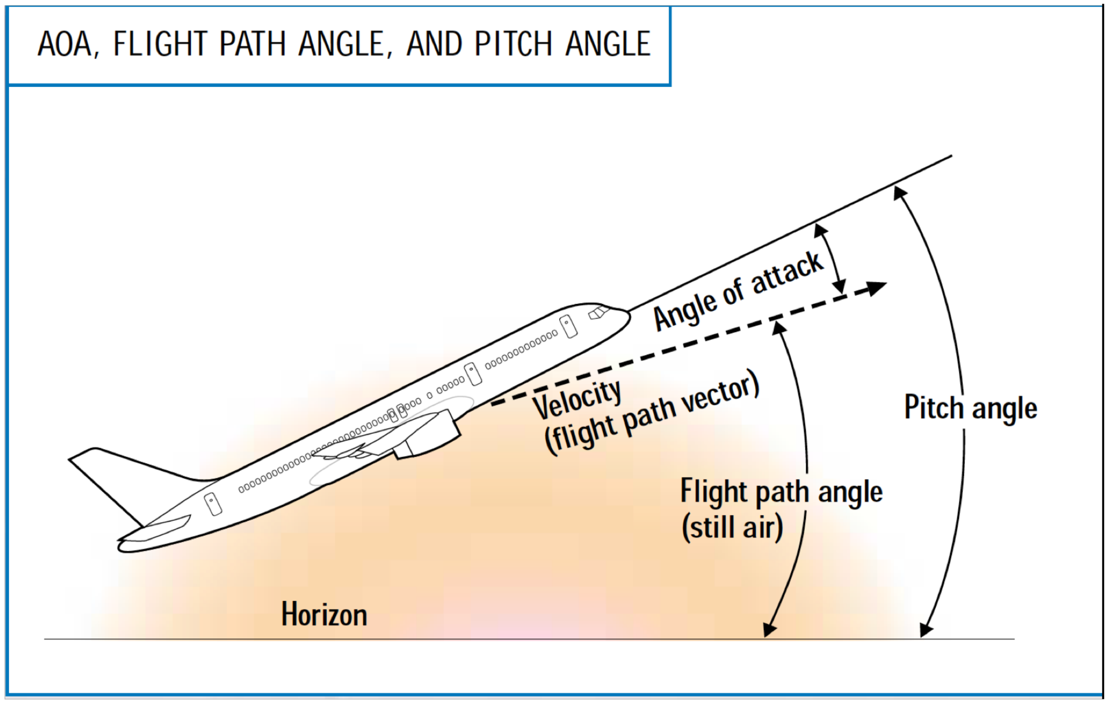
E.g. MTTF 750000 hours and MTTR = 48-hours

MTTF for a duplex system = 668,878 years

MTTF for a triplex system = 6, 967,483,590 years

MTTF (days)	MTTR (days)	Number Devices	MTTF of the n-Plex System (days)	MTTF (years)
730	10	2	26645	73
730	10	3	1296723	3553
730	10	4	70995603	194509

Boeing 737-8 MAX



There is senor to measure the attack angle. If it is too high the aircraft will lose lift and fall down like a brick. The rumor is Boeing 737-8 MAX did not build redundancy for measuring attack angle + software might have error that forced the plane to reduce the angle by pushing nose down and causing crash thinking the attack angle is too large!

Fault Tolerance ...

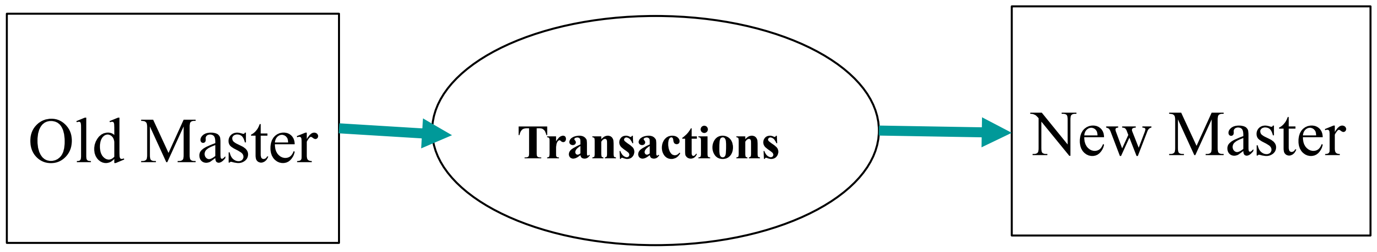
Old master - new master Technique

This method is very similar to shadow paging (we will discuss this later in detail).

– record all updates (transactions) to be performed in a separate file (stable store)

– at night (usually) produce a separate new (next day) master using the old (previous day’s) master and the batched updates (transactions).

Old Master and New Master



Fault Tolerance ...

This model of updates gives us fault tolerance because we can always produce new master file for the next day as long as we have old master file and the transactions to be performed.

The problem is -- this is not on line processing and the failure of a transaction is not notified to the client until the next day.

Software reliability

- The main distinction between software reliability and hardware reliability:
  - Hardware reliability requires tolerating component failures.
  - Software reliability requires tolerating design and coding faults.
  - The distinction between Hardware and Software is becoming less as most hardware units have substantial amount of software components. These systems are generally called embedded systems.
- N-version programming
  - use n programs which are run in parallel, taking majority vote for each answer
  - the advantage is that the diversity of design and coding can mask many failures.

Transactions

- Each program is written as an ACID state transaction with consistency checks.
- Restarting an application with out proper recovery (repair) can make the systems very unreliable.
- Even a transaction processing system cannot tolerate application errors.
- Bohrbugs (after the Physicist Niels Bohr) -- these are deterministic bugs – these are relatively easy to handle
- Heisenbugs (after the Physicist Werner Heisenberg) -- transient (non deterministic) software errors that only appear occasionally, these are usually related to load conditions and timing (race conditions).

- Dense faults - these are the faults that a system can tolerate up to N faults with in a period. If there are more than N faults the system may be interrupted.

- Byzantine faults - these are the faults which are not part of the model and the design is not catered to tolerate such faults. E.g. Designs that do not consider environmental issues like say earth quakes.

How to improve software reliability

Process Pairing

Periodic Transfer of data: One process called Primary does all the work until it fails. The second process called backup takes over the primary and continues the computation. In order to do this Primary need to tell on a regular basis that it is alive and also transmit its state to the secondary.

Checkpoint-restart: The primary records its state on a duplexed storage module. At takeover the secondary starts reading the state of the primary from the duplexed storage and resumes the application.

How to improve software reliability ...

- Checkpoint messages: The primary sends its state changes as messages to the backup. At takeover the backup gets its current state from the most recent checkpoint message.
- Persistent: backup restarts in the null state and lets Transaction mechanism to clean up all uncommitted transactions. This is the approach taken by the most Database Systems.

How to improve software reliability ...

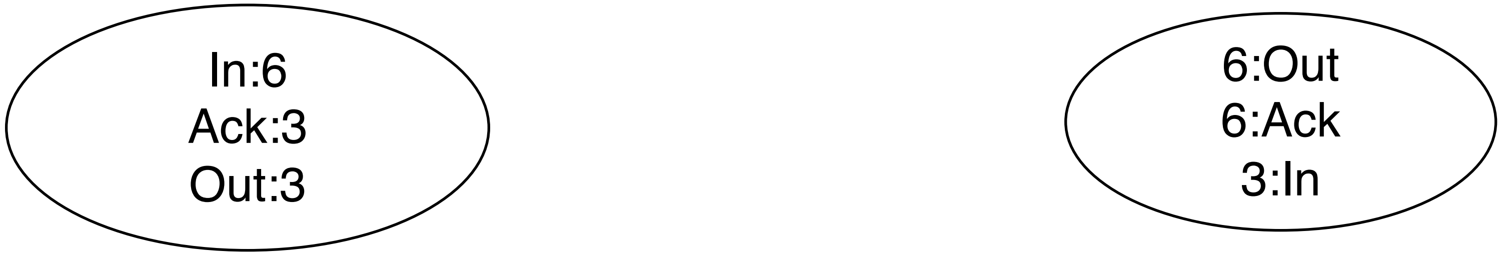
- Highly available storage
  - write to several storage modules.
  - have some kind of checksum to make sure that the data read is correct with a very high probability.
  - Disk mirroring is an example of this.
  - Shadowing is another mirroring technique which allows atomic write operations.
- Highly available Processes
  - process pairing
  - transaction based restart
  - checkpoint restart

How to improve communication reliability ...

Out = #messages sent

In= #messages received

Ack = #acknowledgements

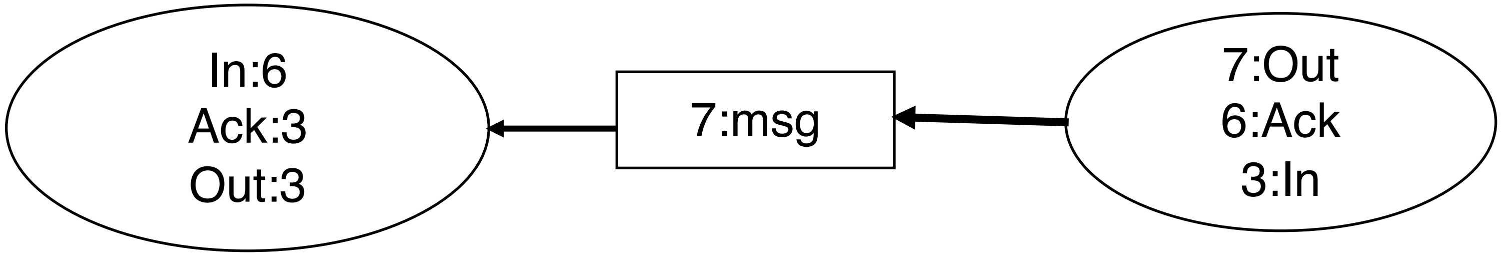


How to improve communication reliability …

Out = #messages sent

In= #messages received

Ack = #acknowledgements



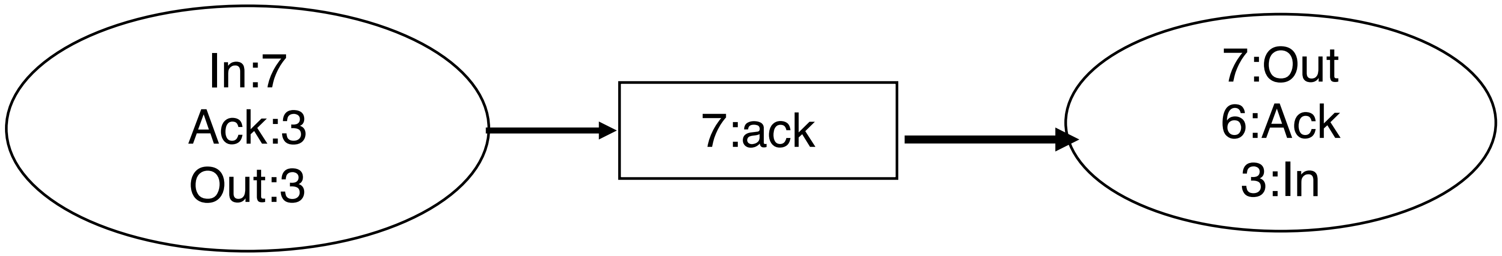
How to improve communication reliability ...

Out = #messages sent

In= #messages received

Ack = #acknowledgements

Reliable message passing



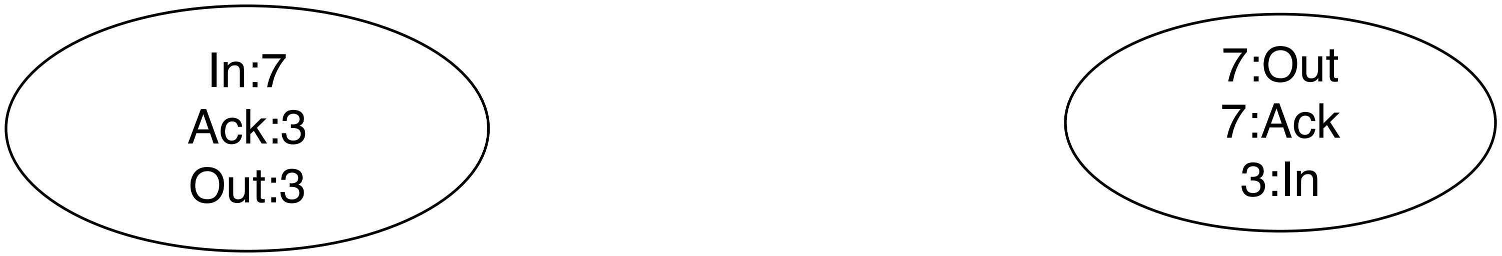
How to improve communication reliability …

Reliable message passing

Out = #messages sent

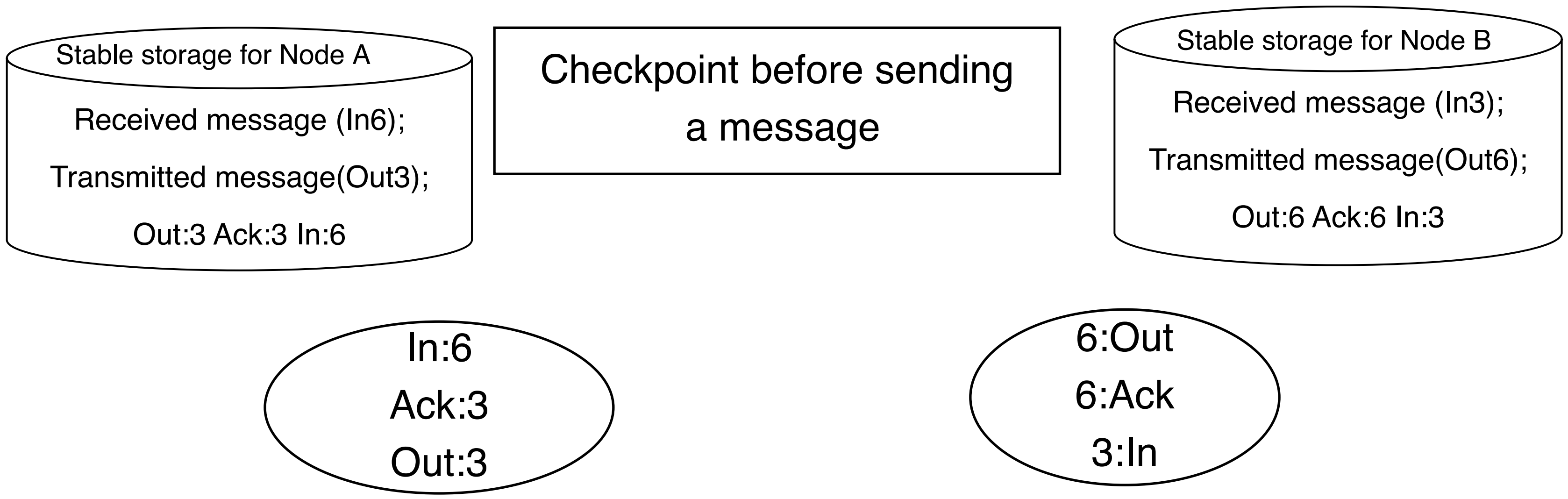
In= #messages received

Ack = #acknowledgements

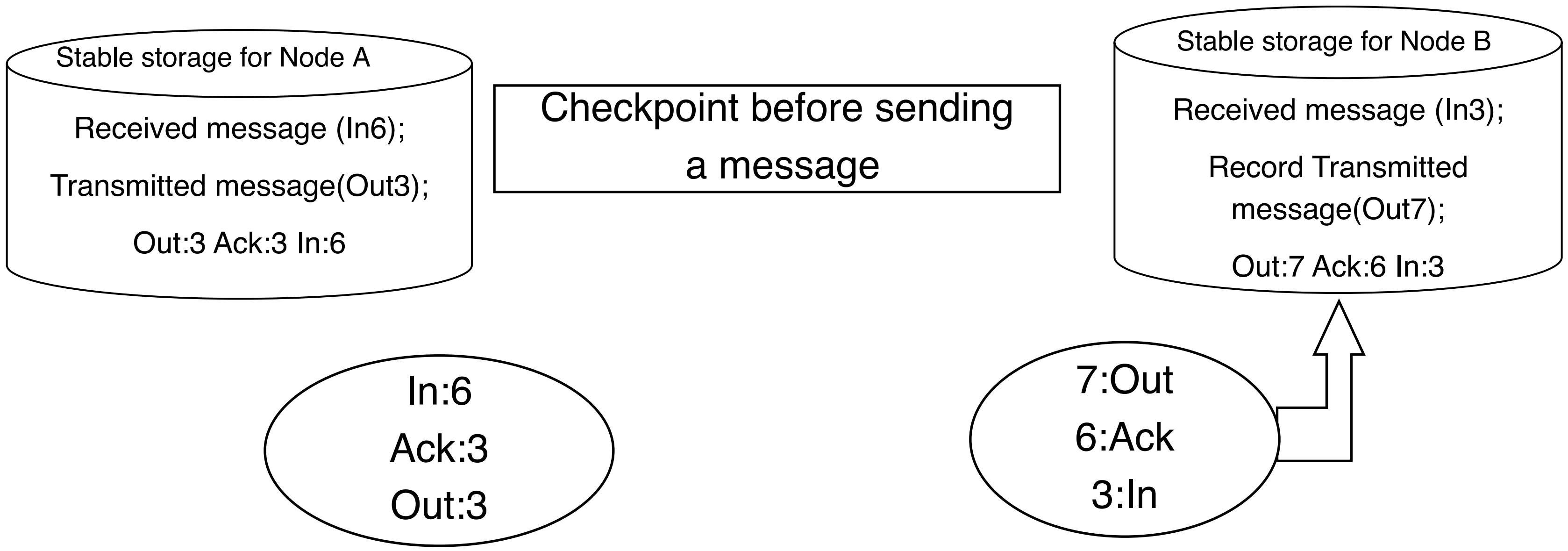


How to improve communication reliability …

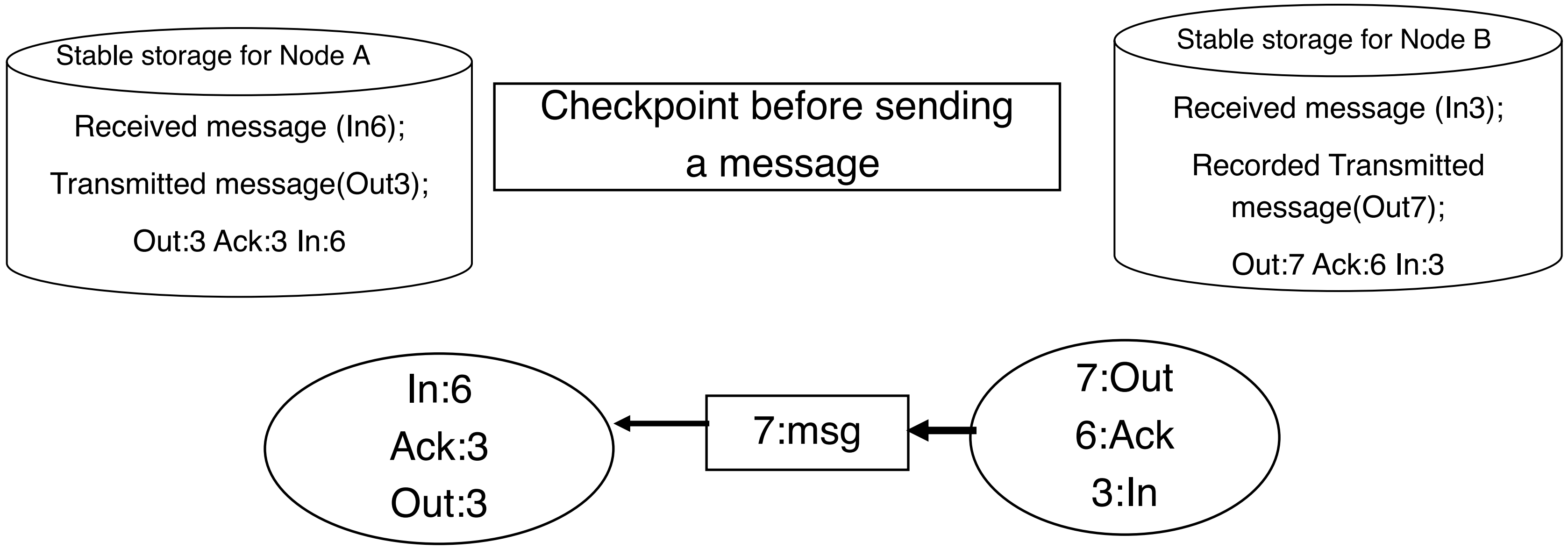
Checkpoint before sending a message



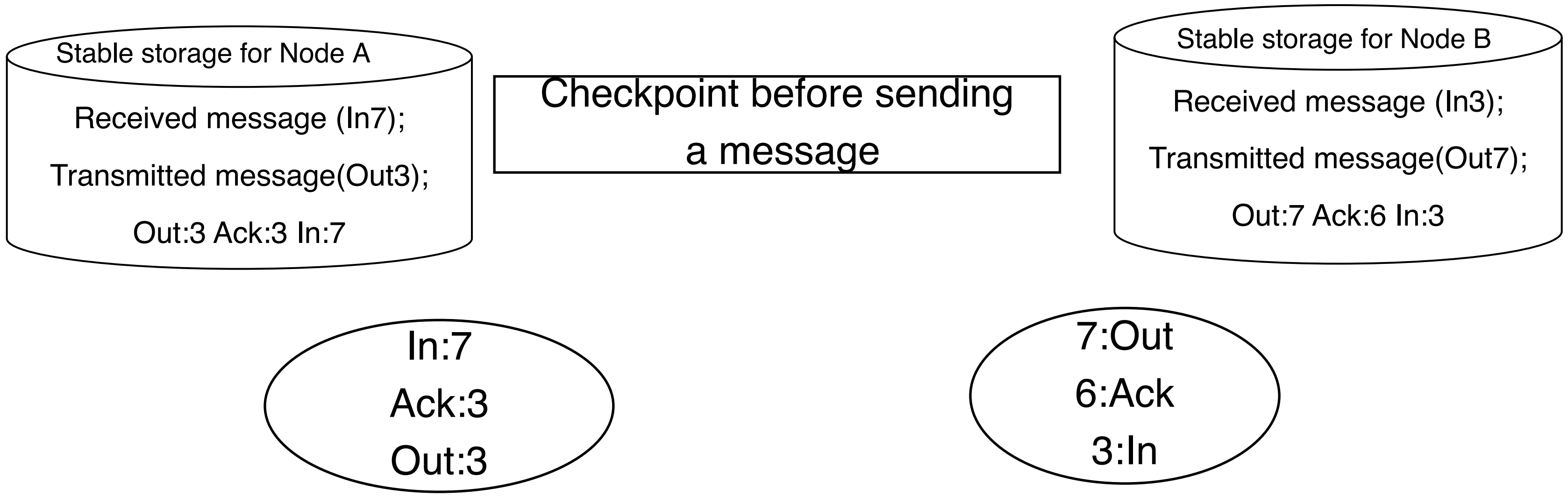
How to improve communication reliability …


Stable storage for Node A
Received message (In6);
Transmitted message(Out3);
Out:3 Ack:3 In:6
Checkpoint before sending a message
Stable storage for Node B
Received message (In3);
Record Transmitted message(Out7);
Out:7 Ack:6 In:3
In:6
Ack:3
Out:3
7:Out
6:Ack
3:In

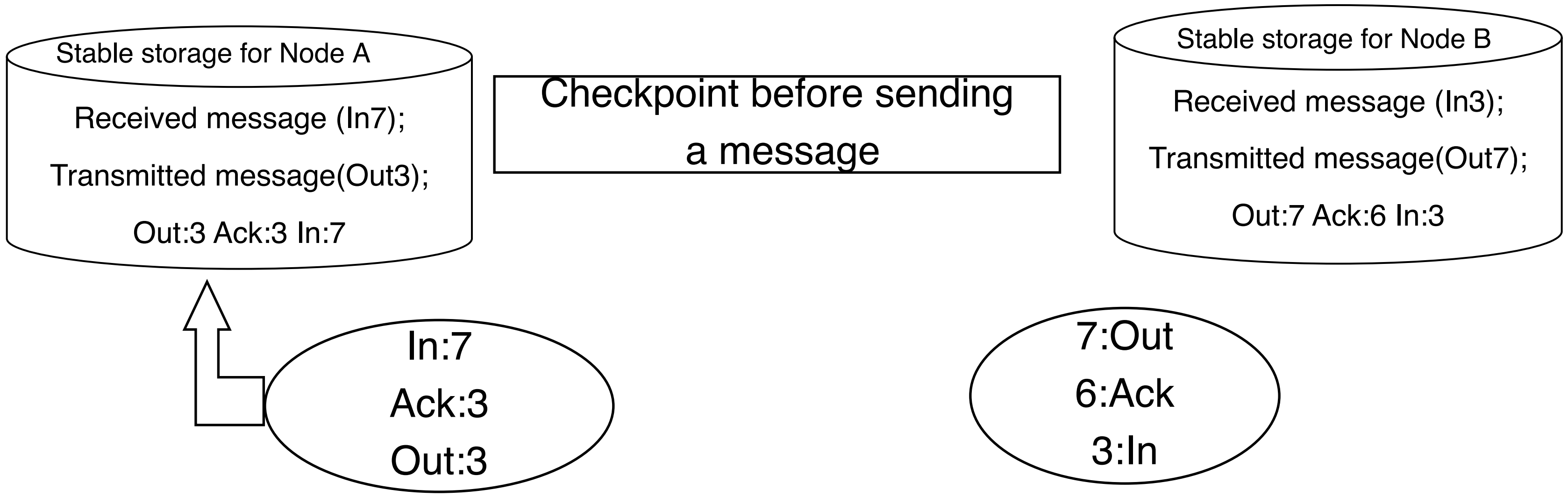
How to improve communication reliability ...


Stable storage for Node A
Received message (In6);
Transmitted message(Out3);
Out:3 Ack:3 In:6
Checkpoint before sending a message
Stable storage for Node B
Received message (In3);
Recorded Transmitted message(Out7);
Out:7 Ack:6 In:3
In:6
Ack:3
Out:3
7:msg
7:Out
6:Ack
3:In

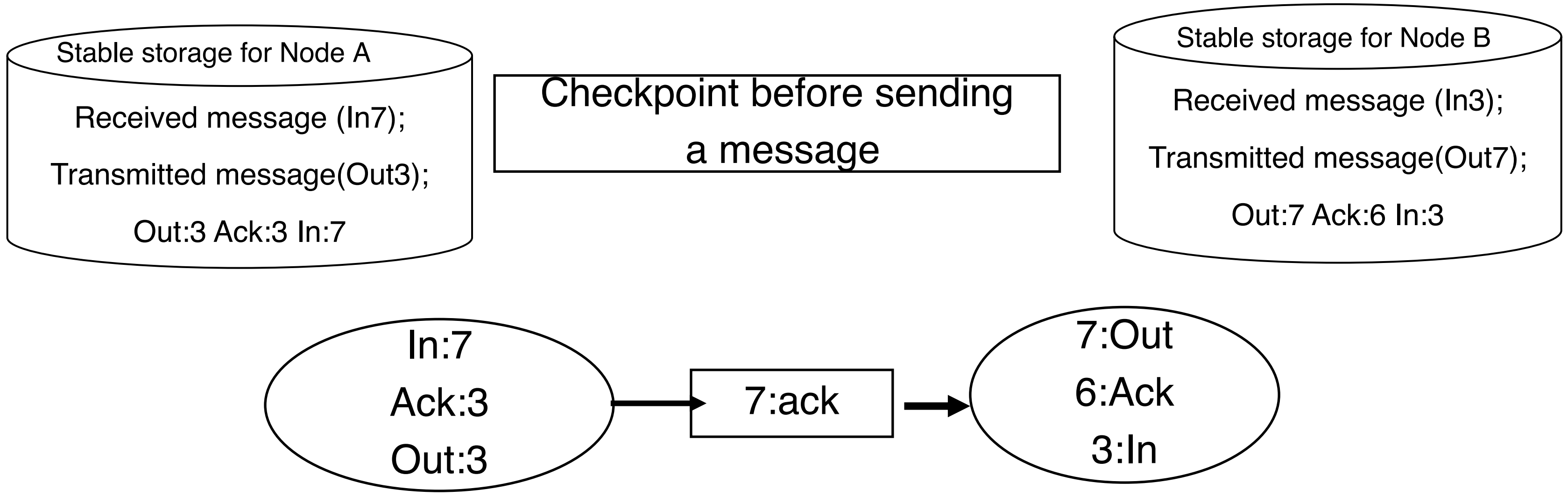
How to improve communication reliability ...


Stable storage for Node A
Received message (In7);
Transmitted message(Out3);
Out:3 Ack:3 In:7
Checkpoint before sending a message
Stable storage for Node B
Received message (In3);
Transmitted message(Out7);
Out:7 Ack:6 In:3
In:7
Ack:3
Out:3
7:Out
6:Ack
3:In

How to improve communication reliability ...

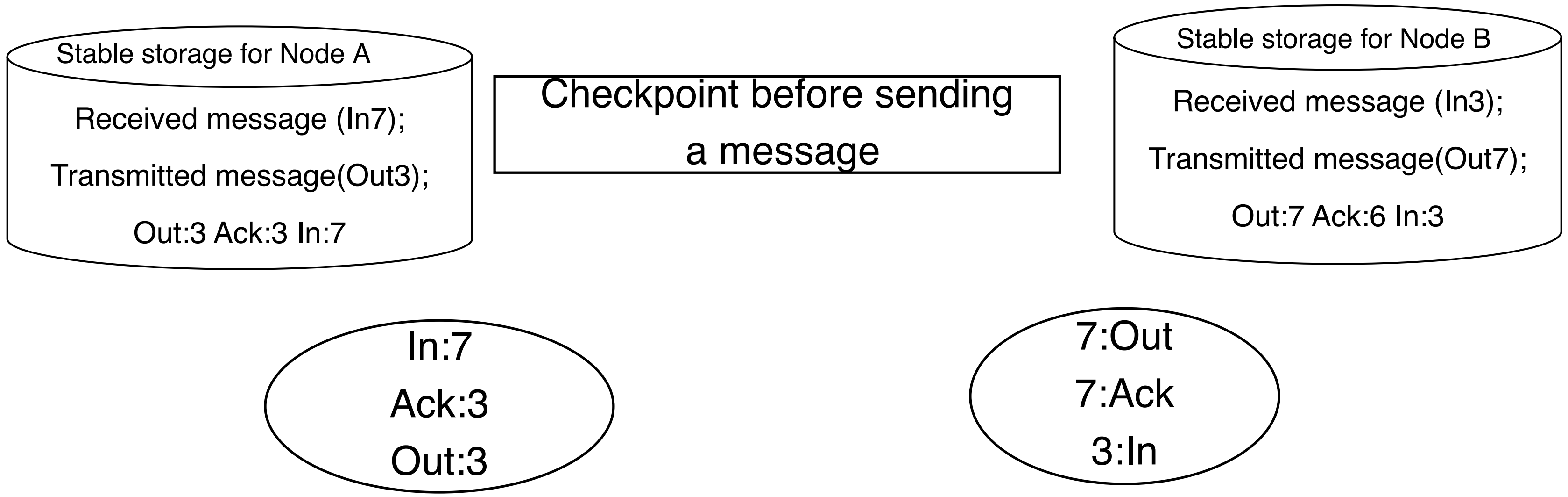

Stable storage for Node A
Received message (In7);
Transmitted message(Out3);
Out:3 Ack:3 In:7
Checkpoint before sending a message
Stable storage for Node B
Received message (In3);
Transmitted message(Out7);
Out:7 Ack:6 In:3
In:7
Ack:3
Out:3
7:Out
6:Ack
3:In

How to improve communication reliability ...

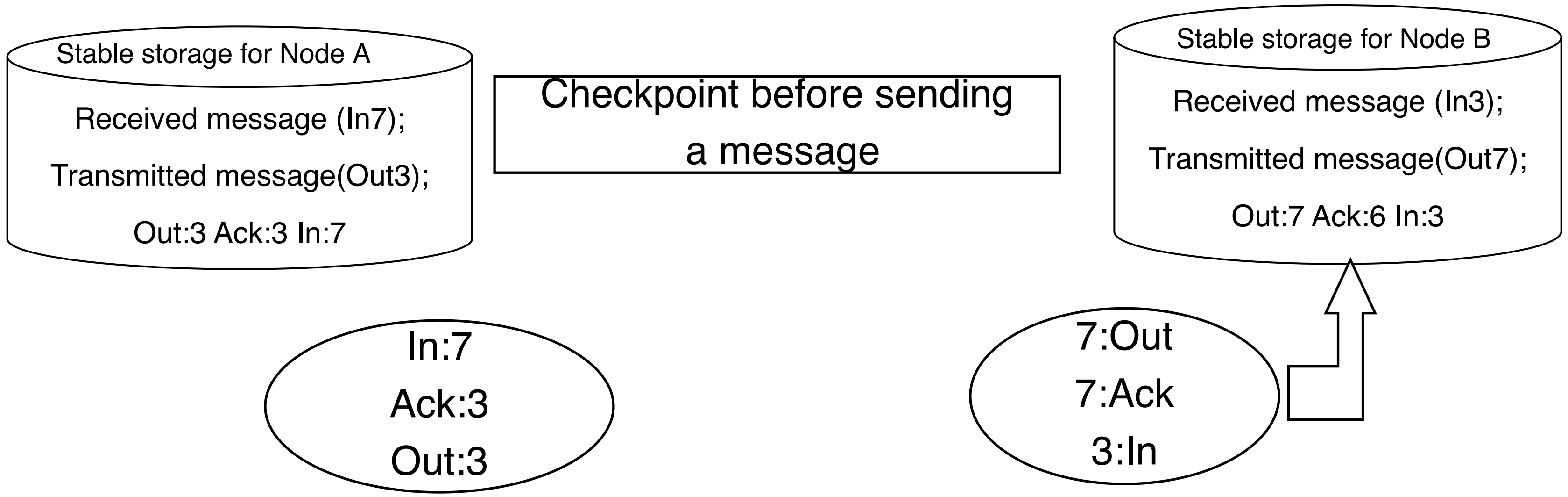


Stable storage for Node A
Received message (In7);
Transmitted message(Out3);
Out:3 Ack:3 In:7
Checkpoint before sending a message
Stable storage for Node B
Received message (In3);
Transmitted message(Out7);
Out:7 Ack:6 In:3
In:7
Ack:3
Out:3
7:ack
7:Out
6:Ack
3:In

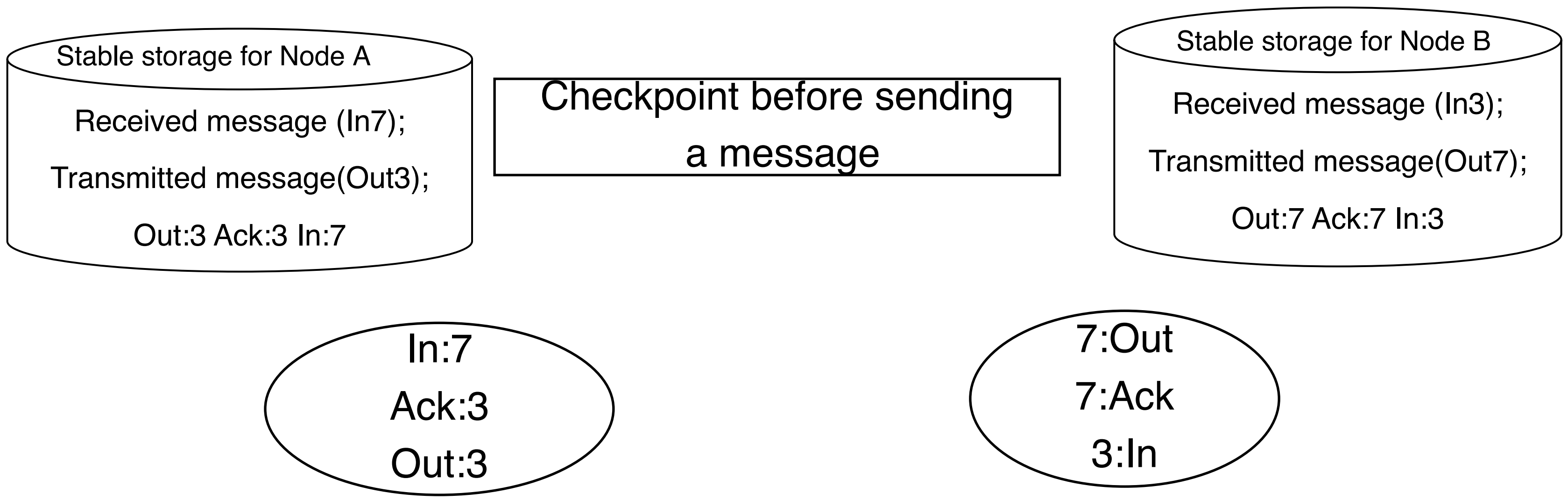
How to improve communication reliability …


Stable storage for Node A
Received message (In7);
Transmitted message(Out3);
Out:3 Ack:3 In:7
Checkpoint before sending a message
Stable storage for Node B
Received message (In3);
Transmitted message(Out7);
Out:7 Ack:6 In:3
In:7
Ack:3
Out:3
7:Out
7:Ack
3:In

How to improve communication reliability ...


Stable storage for Node A
Received message (In7);
Transmitted message(Out3);
Out:3 Ack:3 In:7
Checkpoint before sending a message
Stable storage for Node B
Received message (In3);
Transmitted message(Out7);
Out:7 Ack:6 In:3
In:7
Ack:3
Out:3
7:Out
7:Ack
3:In

How to improve communication reliability ...


Stable storage for Node A
Received message (In7);
Transmitted message(Out3);
Out:3 Ack:3 In:7
Checkpoint before sending a message
Stable storage for Node B
Received message (In3);
Transmitted message(Out7);
Out:7 Ack:7 In:3
In:7
Ack:3
Out:3
7:Out
7:Ack
3:In

System Expectations

- System Delusion (Wrong beliefs)
  - Transaction system can guarantee correct behaviour with respect to reality.
  - Application need not reflect reality.
- Workflow systems are an extension of transaction systems whose behaviour can be expressed with rules/procedures to reflect the business process of a particular organization.
  - E.g. A building permit should not be granted unless it is approved by various authorities such as EPI (Environment Protection Authority), National Trust, etc.
- There is a nice example about inventory in Gray's book.

Summary

Improving system’s reliability and hence its performance involves

• Improving Hardware reliability that is CPUs, Memory and Storage Units. This can be done by employing lot of redundancy such as N-plex systems.

• Software reliability can be improved by employing process-pairing or transaction based recovery.

• Communication Systems reliability not only need hardware redundancy but also need to guaranty transmission and reception of messages and this can be done using stable storage and using the notion of retransmissions and repeated acknowledgements until messages are delivered and acknowledged.