Module 7
Failure Recovery and Fault Tolerance

Basic concept

In any **distributed system**, three kinds of problems can occur.

1) Faults

2)Errors(System enters into an unexpected state)

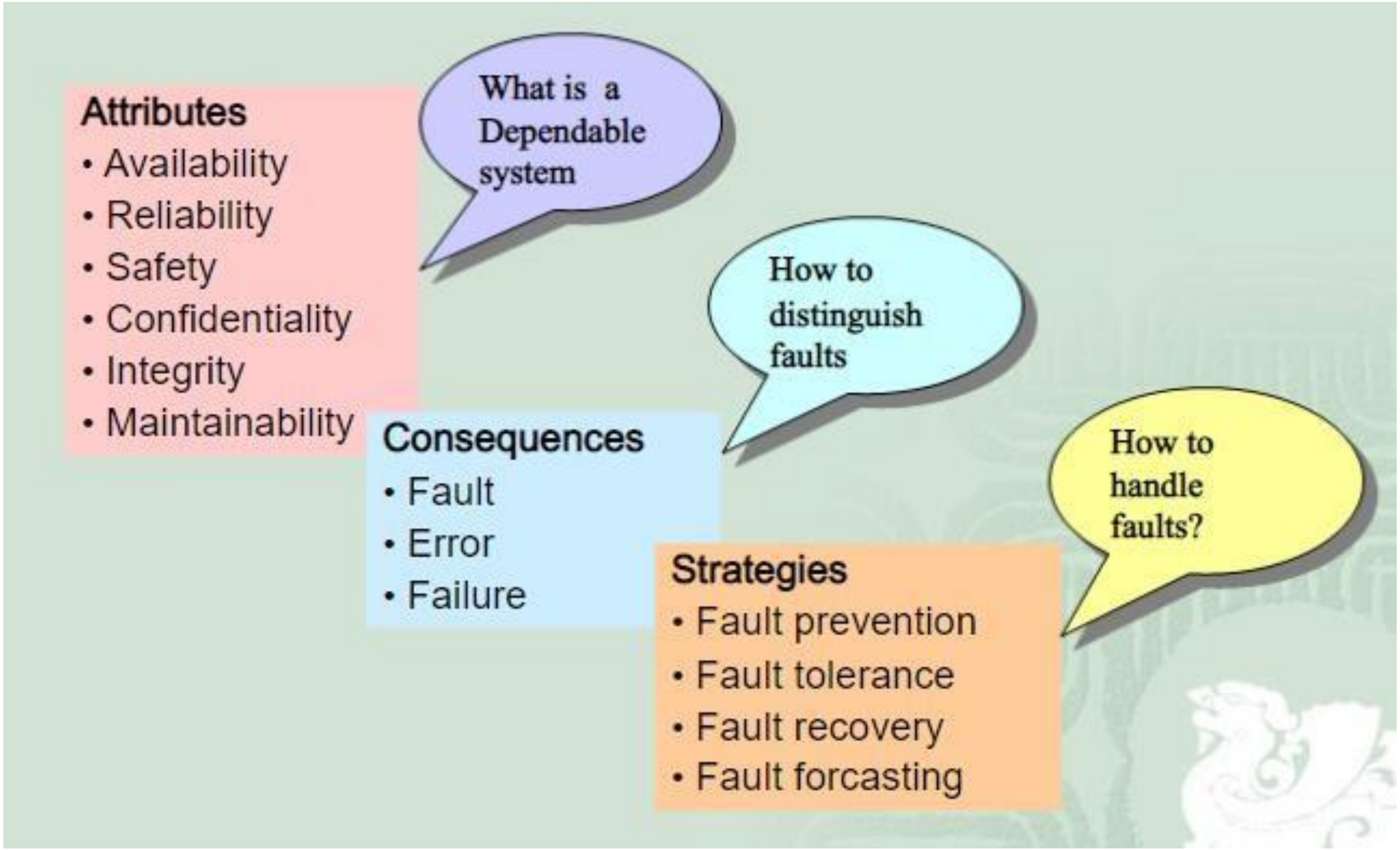
3)Failures

All these are inter related. ...

Failure Occurs when a server does not receive incoming requests from client or fails to send messages in response to clients request.

Basic Concept

- **Fault tolerance** refers to the ability of a **system** (computer, network, cloud cluster, etc.) to continue operating without interruption when one or more of its components fail.
- **Fault-tolerant systems** use backup components that automatically take the place of failed components, ensuring no loss of service.



Attributes
- Availability
- Reliability
- Safety
- Confidentiality
- Integrity
- Maintainability

What is a Dependable system

Consequences
- Fault
- Error
- Failure

How to distinguish faults

Strategies
- Fault prevention
- Fault tolerance
- Fault recovery
- Fault forcasting

How to handle faults?

System attributes:

Availability – system always ready for use, or probability that system is ready or available at a given time

Reliability – property that a system can run without failure, for a given time

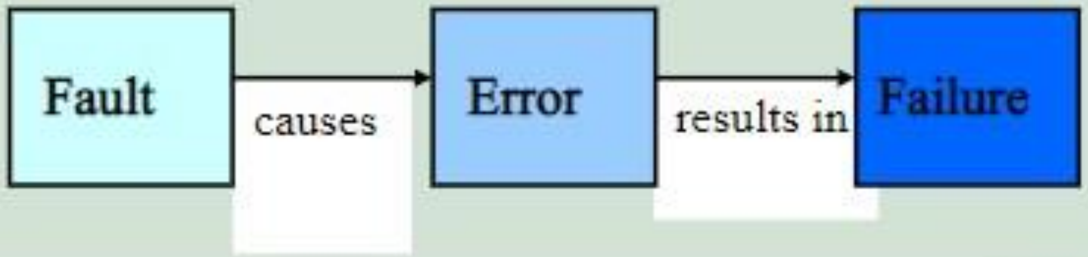
Safety – indicates the safety issues in the case the system fails

Maintainability – refers to the ease of repair to a failed system

Failure in a distributed system = when a service cannot be fully provided

System failure may be partial

A single failure may affect other parts of a system (failure escalation)



Fault – is a defect within the system

Error – is observed by a deviation from the expected behaviour of the system

Failure occurs when the system can no longer perform as required (does not meet spec)

Fault Tolerance – is ability of system to provide a service, even in the presence of errors

Hard or Permanent – repeatable error, e.g. failed component, power fail, fire, flood, design error (usually software), sabotage

Soft Fault

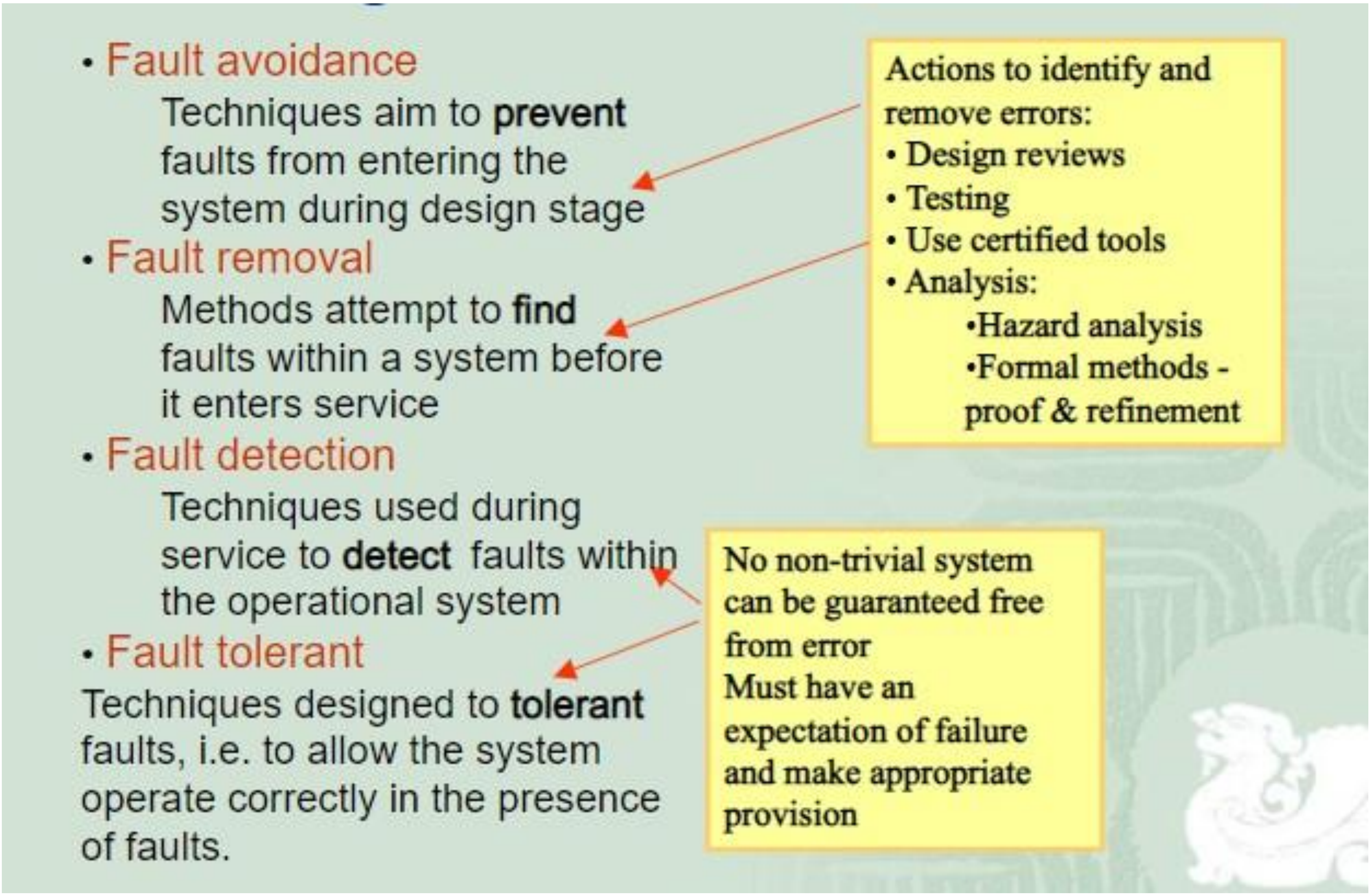
Transient – occurs once or seldom, often due to unstable environment (e.g. bird flies past microwave transmitter)

Intermittent – occurs randomly, but where factors influencing fault are not clearly identified, e.g. unstable component

Operator error – human error

Type of failure	Description
Crash failure	A server halts, but is working correctly until it halts
Amnesia crash	Lost all history, must be reboot
Pause crash	Still remember state before crash, can be recovered
Halting crash	Hardware failure, must be replaced or re-installed
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	The 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

Strategies to handle faults



- Fault avoidance

  Techniques aim to **prevent** faults from entering the system during design stage
- Fault removal

  Methods attempt to **find** faults within a system before it enters service
- Fault detection

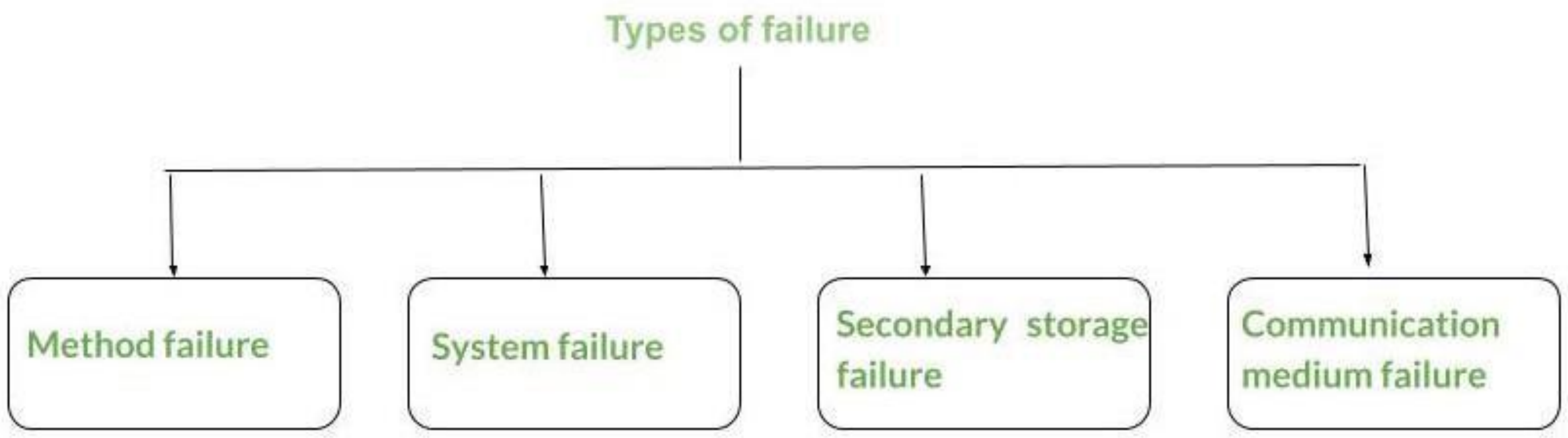
  Techniques used during service to **detect** faults within the operational system
- Fault tolerant

Techniques designed to **tolerant** faults, i.e. to allow the system operate correctly in the presence of faults.

Actions to identify and remove errors:
- Design reviews
- Testing
- Use certified tools
- Analysis:
  - Hazard analysis
  - Formal methods - proof & refinement

No non-trivial system can be guaranteed free from error
Must have an expectation of failure and make appropriate provision

Type of Failures



Method failure

- In this type of failure, the distributed system is generally halted and unable to perform the execution.
- Sometimes it leads to ending up the execution resulting in an associate incorrect outcome.
- Method failure causes the system state to deviate from specifications, and also method might fail to progress.

Method failure

- **Behavior –**
  It may be understood as if incorrect computation like Protection violation, deadlocks, timeout, user input, etc is performed then the method stops its execution.
- **Recovery –**
  Method failure can be prevented by aborting the method or restarting it from its prior state.

System Failure

- In system failure, the processor associated with the distributed system fails to perform the execution. This is caused by computer code errors and hardware issues. Hardware issues may involve CPU/memory/bus failure. This is assumed that whenever the system stops its execution due to some fault then the interior state is lost.

System Failure

- **Behavior –**
  It is concerned with physical and logical units of the processor. The system may freeze, reboot and also it does not perform any functioning leading it to go in an idle state.
- **Recovery –**
  This can be cured by rebooting the system as soon as possible and configuring the failure point and wrong state.

Secondary Storage device failure

- A storage device failure is claimed to have occurred once the keep information can’t be accessed. This failure is sometimes caused by parity error, head crash, or dirt particles settled on the medium.

Secondary Storage device failure

- **Behavior –**
  Stored information can’t be accessed.
- **Recovery/Design strategies –**
  Reconstruct content from the archive and the log of activities and style reflected disk system. A system failure will additionally be classified as follows.
  - Associate cognitive state failure
  - A partial cognitive state failure
  - a disruption failure
  - A halting failure

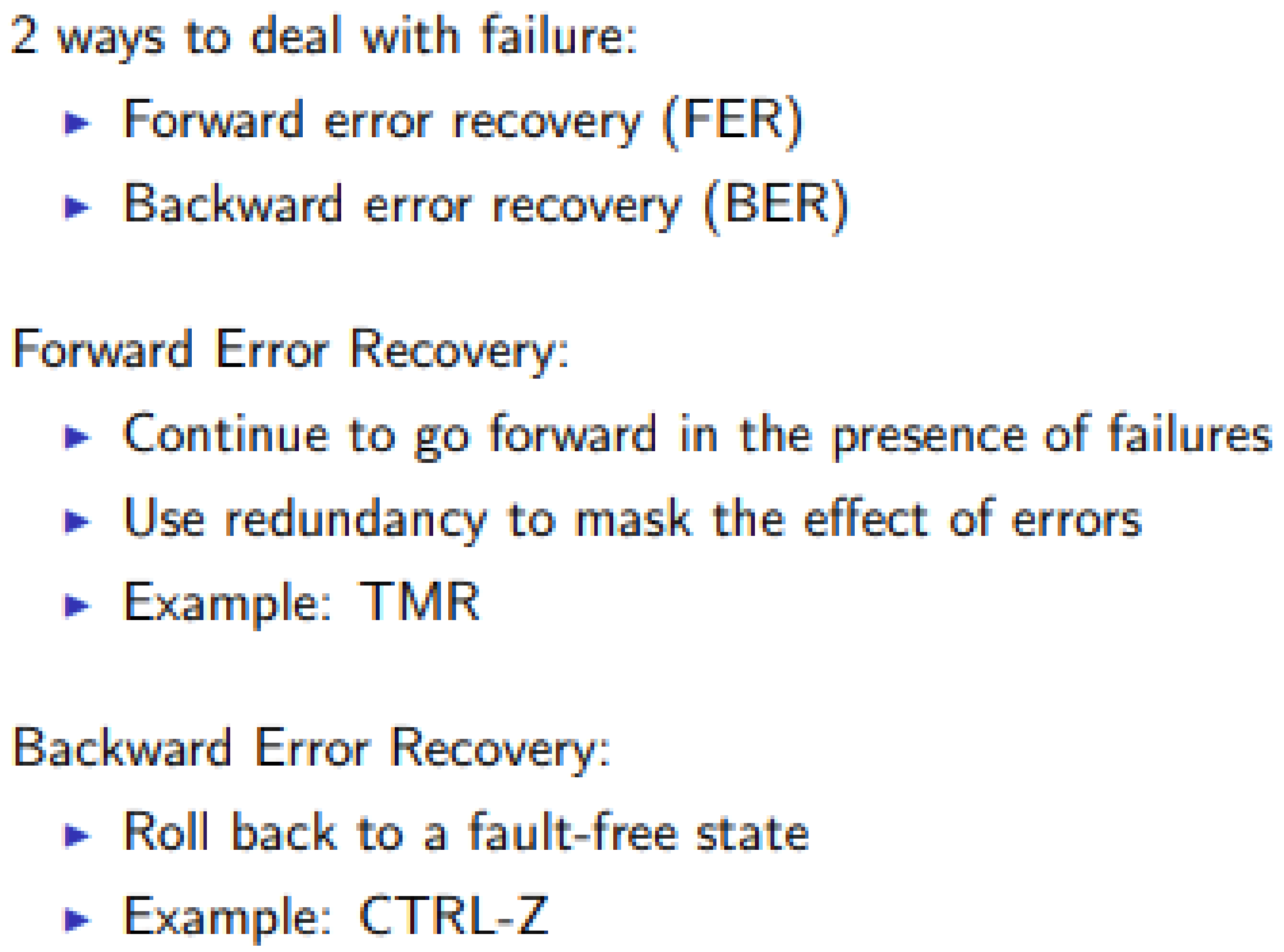
Communication Medium Failure

- A communication medium failure happens once a web site cannot communicate with another operational site within the network. it’s typically caused by the failure of the shift nodes and/or the links of the human activity system.

Communication Medium Failure

- **Behavior –**
  A web site cannot communicate with another operational site.
- **Errors/Faults –**
  Failure of shift nodes or communication links.
- **Recovery/Design strategies –**
  Reroute, error-resistant communication protocols.

Methods to deal with failure



2 ways to deal with failure:

- Forward error recovery (FER)
- Backward error recovery (BER)

Forward Error Recovery:

- Continue to go forward in the presence of failures
- Use redundancy to mask the effect of errors
- Example: TMR

Backward Error Recovery:

- Roll back to a fault-free state
- Example: CTRL-Z

Forward Error Recovery

Uses sufficient physical or information redundancy to not only detect errors but also correct them.

- FER is inseparable from error detection
- Error detection is on the critical path

Examples:

- Triple modular redundancy
- Error correcting codes

Backward Error Recovery

Restores the application to a previous error-free state when a failure is detected

- Separate error detection mechanisms are required
  - Error detection is off the critical path

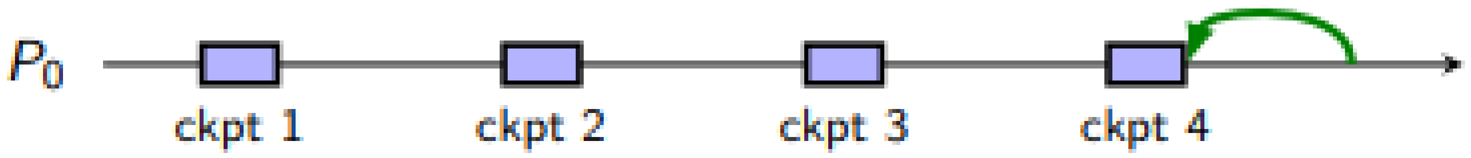
- Information about the state of the application need to be saved during failure free execution
- Require less computing resources

BER techniques:

- Checkpointing
- Logging

Checkpointing

Periodically save the state of the application to avoid losing everything in the event of a crash:



Note that saving the state of the application may take some non negligible amount of time.

Logging

Idea

- Log all changes to the state (at least the non-deterministic ones)
- When a failure occurs, use logged data to recover the failed state

Comments

- Solves the output commit problem
  - Better suited than checkpointing when interactions with the outside world are frequent
- Can be combined with checkpointing to reduce the size of the logs

APPROACHS

1.The operation –based approach

2. state-based approach

The operation based approach

methods:

updating-in-place

do

undo

redo

display   based on log records

write-ahead-log protocol

update an object only after the undo is recorded

before committing the updates, redo and undo logs recorded.

State-based approach

shadow pages:

1. a special case of the state-based recovery approach is the technique based on shadow pages.
2. Under this technique only a part of the saved to facilitate recovery
3. whenever a process wants to modify on object ,the page containing the object is duplicated and is maintained on stable storage.
4. The other unmodified copy is known as the shadow page

Commit protocols

- In **distributed** data base and transaction **systems** a **distributed commit protocol** is required to ensure that the effects of a **distributed** transaction are atomic, that is, either all the effects of the transaction persist or none persist, whether or not failures occur.

One phase commit protocol

- A **one-phase commit protocol** involves a coordinator periodically communicating with the servers that are carrying out each individual **transaction** in order to inform them whether to **commit** the **transaction** or abort it.

Two phase commit protocol

- The **two phase commit protocol** is a **distributed** algorithm which lets all sites in a **distributed system** agree to **commit** a **transaction**. The **protocol** results in either all nodes committing the **transaction** or aborting, even in the case of site failures and message losses

Two phase commit protocol

•one process acts as *coordinator*
•other processes are called *cohorts*
•stable storage is available at each site
•each site uses write-ahead log protocol, to achieve local fault recovery and rollback
•Phase-I. At the coordinator
• 1.the coordinator sends a COMMIT-REQUEST message to every cohort requesting the cohorts to commit
•2.the coordinator waits for replies from all the cohorts.
•At cohorts:
• on receving the CO,,IT-REQUEST message,a cohorts takes the following actions
• if the transaction excuting at the cohort is successful,it writs UNDO and REDO log on the stable storage and sends an AGREED message to the ccordinator.
•Otherwise ,it sends an ABORT message to the coordinator

Two phase commit protocol

Phase II At the coordinator

1.If all the cohorts reply AGREED and the coordinator also agrees,then the coordinator writes a COMMIT record into the log.Then it sends a COMMIT message to all the cohorts,otherwise ,the coordinators sends an ABORT message to all the cohorts.

2.The coordinator then waits for acknowledgements from each cohort.

3.If an acknowledgement is not recived from any cohort within a timeout period,the coordinator resends the COMMIT/ABORT mmesage to the cohort.

4.If all the acknowledgements are recievd,the coordinator writes a COMPLETE record to the log (to indicate the completion of the transaction)

At cohorts:

1.On receiving a COMMIT message,a cohort releases all the resources and locks held by it for excuting the transaction,and sends an acknowledgement.

2.On receiving an ABORT message,a cohort undoes the transaction using the UNDO log record,releases all the resources and locks held by it for performing the transaction ,and sends an acknowledgement.

limitations

- Biggest drawback is that it is a blocking protocol
- Whenever the coordinator fails, cohort sites will have to wait for its recovery
- This is undesirable as these sites may be holding locks on the resource.