Chapter 5: Availability

What is Availability?

- **Availability** refers to a property of software that it is there and ready to carry out its task when you need it to be.
- **Availability** refers to the ability of a system to mask or repair faults such that the cumulative service outage period does not exceed a required value over a specified time interval.
- **Availability** is about minimizing service outage time by mitigating faults

Availability

- Availability v.s. reliability or dependability
  - Availability encompasses what is normally called reliability.
  - Availability encompasses other consideration such as service outage due to period maintenance
- Availability is closely related to
  - security, e..g, denial-of-service
  - performance
  - …

TABLE 5.1 System Availability Requirements

Availability	Downtime/90 Days	Downtime/Year
99.0%	21 hours, 36 minutes	3 days, 15.6 hours
99.9%	2 hours, 10 minutes	8 hours, 0 minutes, 46 seconds
99.99%	12 minutes, 58 seconds	52 minutes, 34 seconds
99.999%	1 minute, 18 seconds	5 minutes, 15 seconds
99.9999%	8 seconds	32 seconds

Availability General Scenario

Portion of Scenario	Possible Values
Source	Internal/external: people, hardware, software, physical infrastructure, physical environment
Stimulus	Fault: omission, crash, incorrect timing, incorrect response
Artifact	System's processors, communication channels, persistent storage, processes
Environment	Normal operation, startup, shutdown, repair mode, degraded operation, overloaded operation
Response	Prevent the fault from becoming a failure Detect the fault: • log the fault • notify appropriate entities (people or systems) Recover from the fault • disable source of events causing the fault • be temporarily unavailable while repair is being effected • fix or mask the fault/failure or contain the damage it causes • operate in a degraded mode while repair is being effected
Response Measure	Time or time interval when the system must be available Availability percentage (e.g. 99.999%) Time to detect the fault Time to repair the fault Time or time interval in which system can be in degraded mode Proportion (e.g., 99%) or rate (e.g., up to 100 per second) of a certain class of faults that the system prevents, or handles without failing

Sample Concrete Availability Scenario

- The heartbeat monitor detects that the server is nonresponsive during normal operations. The system informs the operator and continues to operate with no downtime.
  - **Stimulus**: non-responsiveness
  - **Response**: inform the operator
  - **Response measure**: no downtime, or 100 availability percentages
  - **Environment**: normal operation
  - **Artifact**: heartbeat monitor
  - **Stimulus source**: server

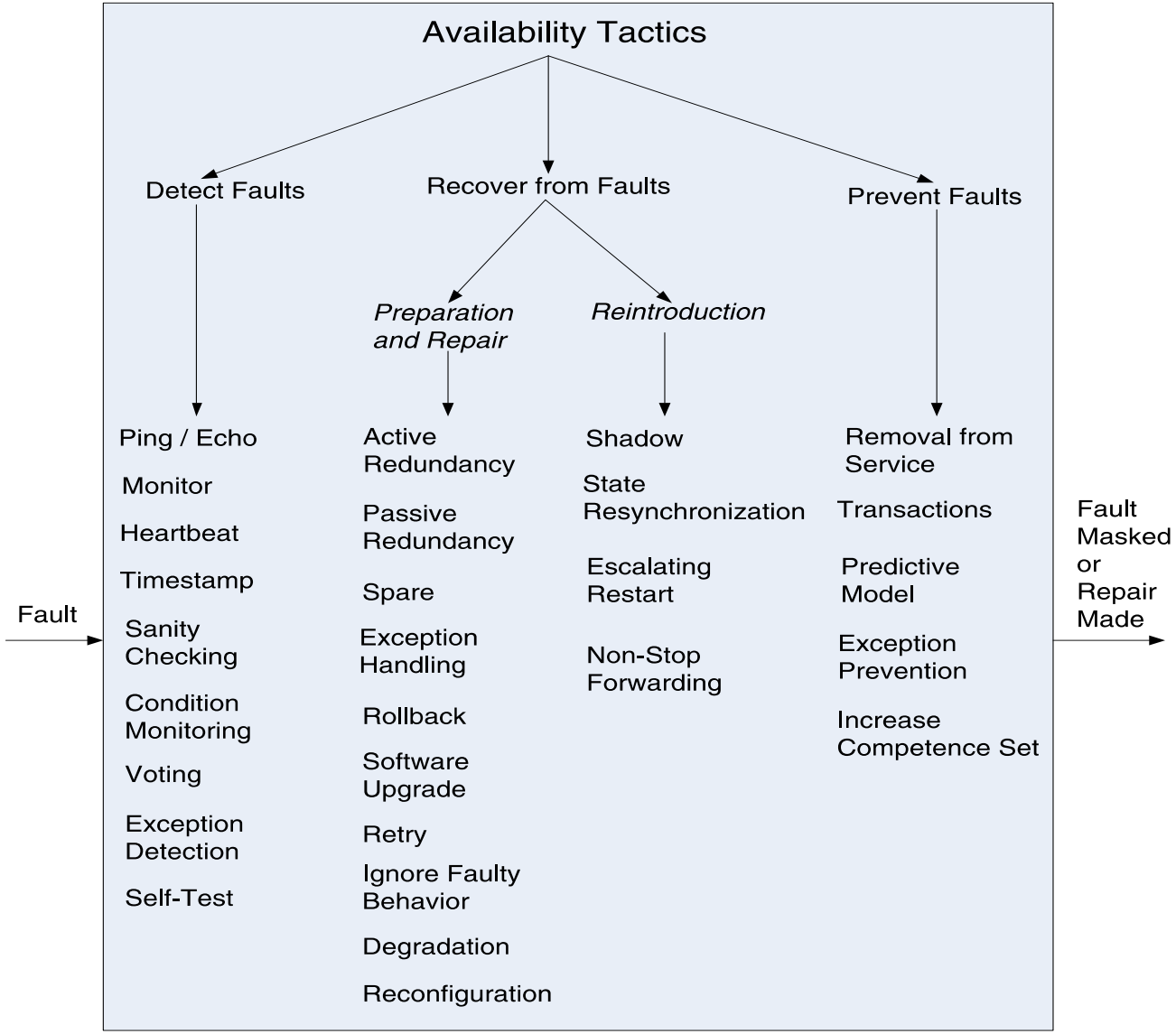
Goal of Availability Tactics

- Fault (故障) v.s. failure (失效) ?
- A **failure** occurs when the system (unexpectedly) no longer delivers a service consistent with its specification
  - this failure is observable by the system's actors.
- A **fault** (or combination of faults) has the potential to cause a failure.

Goal of Availability Tactics

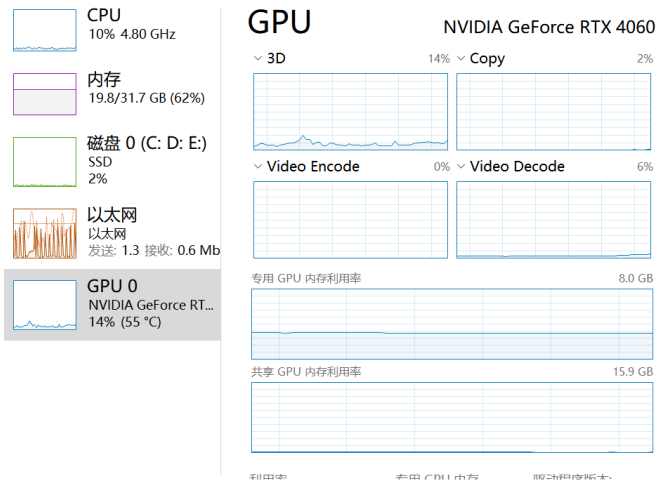
- **Availability tactics** enable a system to endure faults so that services remain compliant with their specifications.
- **The tactics** keep faults from becoming failures or at least bound the effects of the fault and make repair possible.

Availability Tactics



Detect Faults

- **Ping/echo:** used to determine reachability and the round-trip delay through the associated network path. (by a system monitor)
- **Monitor**: a component used to monitor the state of health of other parts of the system



Detect Faults: Heartbeat

- **Heartbeat**: a periodic message exchange between a system monitor and a process being monitored.
  - The process periodically resets the *watchdog* timer in its monitor,
  - Piggybacking the heartbeat messages on to other control messages reduces the overhead
- Difference between ping and heartbeat?
  - Who initiates the health check?

Detect Faults

- **Timestamp**: used to detect incorrect sequences of events, primarily in distributed message-passing systems.
- **Condition Monitoring**: checking conditions in a process or device, or validating assumptions made during the design.
  - For example, **checksum** in data storage and transmission

Checksum

Data: Let's say we have the following sequence of bytes represented in hexadecimal:

```
0xA8, 0x50, 0x3F, 0x2D
```

Step 1: Convert Hexadecimal to Decimal

First, convert each byte from hexadecimal to decimal:

- `0xA8` = 168
- `0x50` = 80
- `0x3F` = 63
- `0x2D` = 45

Step 2: Calculate the Sum

Next, sum these decimal values:

$$\text{Sum} = 168 + 80 + 63 + 45 = 356$$

Step 3: Calculate the Checksum

To obtain the checksum, you can take the sum modulo 256 (since we are dealing with 8-bit unsigned values):

$$\text{Checksum} = 356 \mod 256 = 100$$

Step 4: Convert Checksum Back to Hexadecimal

Convert the checksum back to hexadecimal:

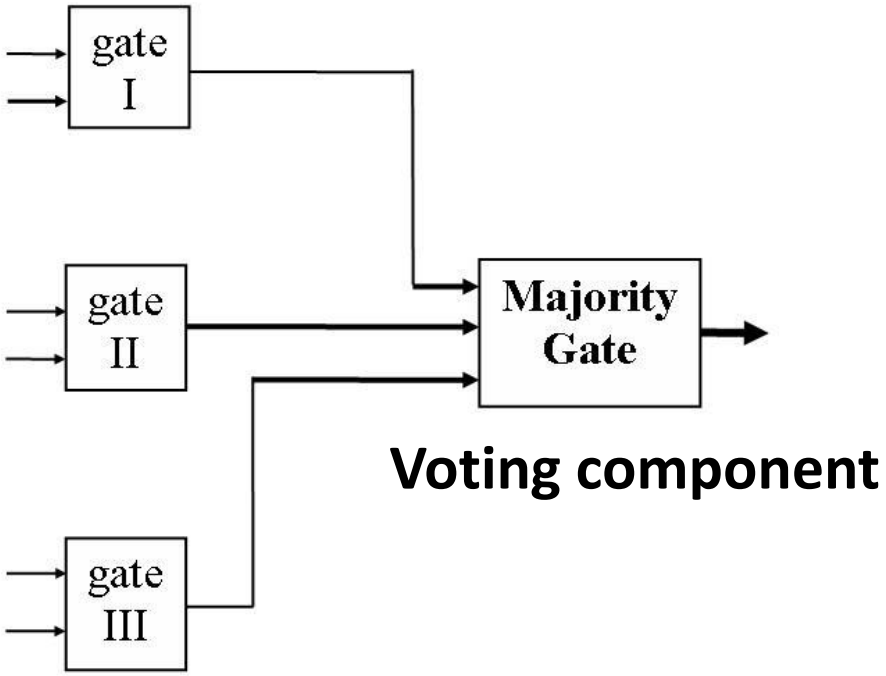
$$100 = 0x64$$

Final Result

So, the checksum for the data `0xA8, 0x50, 0x3F, 0x2D` is **0x64**.

Detect Faults

- **Voting**: the common realization of this tactic is Triple Modular Redundancy (TMR)



Recover from Faults (Preparation & Repair)

- **Active Redundancy (hot spare):** all nodes in a *protection group* process identical inputs in parallel, allowing redundant spares to maintain *synchronous state* with the active nodes.
- **Spare (cold spare):** redundant spares of a protection group remain out of service until a fail-over occurs, at which point a power-on-reset procedure is initiated on the redundant spare prior to its being placed in service.

Recover from Faults (Preparation & Repair)

- **Passive Redundancy (warm spare):** only the active members of the protection group process input traffic;
- one of their duties is to provide the redundant spare(s) with periodic state updates.

Recover from Faults (Preparation & Repair)

- **Rollback**: revert to a previous known good state, referred to as the "rollback line".
- This tactic is combined with redundancy tactics
- After a rollback has occurred, a standby version of the failed component becomes active
- Rollback depends on a copy of a previous state (a checkpoint)
- Checkpoint can be stored in a fixed location and needs to be updated regularly

For example, when a mobile phone system is upgraded to a new version and some serious bugs or compatibility issues are found, the system can choose to roll back to the previous stable version.

Recover from Faults (Preparation & Repair)

- **Retry**: where a failure is transient and retrying the operation may lead to success.
  - For example, network re-transmission
  - There should be a limit on the number of retries that are attempted before a permanent failure is declared.
- **Ignore Faulty Behavior**: ignoring messages sent from a source when it is determined that those messages are spurious.
  - E.g., ignore the messages from a denial of service attacker

Recover from Faults (Preparation & Repair)

- **Degradation**: maintains the most critical system functions in the presence of component failures, dropping less critical functions.
- **Reconfiguration**: reassigning responsibilities to the resources left functioning, while maintaining as much functionality as possible.

Prevent Faults

- **Removal From Service**: temporarily placing a system component in an out-of-service state for the purpose of mitigating potential system failures
  - Temporarily remove the server with degraded performance from the service queue
  - Restart your computer
- **Predictive Model**: take corrective action when conditions are detected that are predictive of likely future faults.
  - e.g. Temperature is too high…

Prevent Faults

- **Increase Competence Set:** designing a component to handle more cases—faults—as part of its normal operation
- **Transactions:** "two-phase commit" (a.k.a. 2PC) protocol. This tactic prevents race conditions caused by two processes attempting to update the same data item.
  - If you transfer money from account A to account B, the transaction steps include:
    - Deduct money from account A
    - Deposit the corresponding amount of money into account B
  - Through the transaction mechanism, these two steps will either succeed or fail, avoiding the situation where the deduction succeeds but the credit fails

Summary

- Availability refers to the ability of the system to be available for use when a fault occurs.
- The fault must be recognized (or prevented) and then the system must respond.
- The response will depend on the criticality of the application and the type of fault
  - can range from “ignore it” to “keep on going as if it didn’t occur.”

Summary

- Tactics for availability are categorized into detect faults, recover from faults and prevent faults.
- Detection tactics depend on detecting signs of life from various components.
- Recovery tactics are retrying an operation or maintaining redundant data or computations.
- Prevention tactics depend on removing elements from service or limiting the scope of faults.