CSE 601: Distributed Systems

Toukir Ahammed

Fault Tolerance

Introduction to Fault Tolerance

- An important goal in distributed systems design is to construct the system in such a way that it can automatically recover from partial failures without seriously affecting the overall performance.
- In particular, whenever a failure occurs, the distributed system should continue to operate in an acceptable way while repairs are being made, that is, it should tolerate faults and continue to operate to some extent even in their presence.

Definition of Fault Tolerance

- **Fault tolerance**, meaning that a system can provide its services even in the presence of faults. In other words, the system can tolerate faults and continue to operate normally.
- Fault tolerance is the ability of a system to continue performing its intended functions in presence of faults.
- In a broad sense, fault tolerance is associated with reliability, with successful operation, and with the absence of breakdowns.
- A fault-tolerant system should be able to handle faults in individual hardware or software components, power failures, or other kinds of unexpected problems and still meet its specification.

Definition of Fault Tolerance

- Fault tolerance is necessary because it is practically impossible to build a perfect system.
- The fundamental problem is that, as the complexity of a system grows, its reliability drastically decreases, unless compensatory measures are taken.
- For example, if the reliability of individual components is 99.99%, then the reliability of a system consisting of 100 non-redundant components is 99.01%, whereas the reliability of a system consisting of 10,000 non-redundant components is just 36.79%.
- Such a low reliability is unacceptable in most applications. If a 99% reliability is required for a 10,000-component system, individual components with a reliability of at least 99.999% should be used, implying a sharp increase in cost.

Fault Tolerance and Redundancy

- There are various approaches to achieving fault tolerance. Common to all these approaches is a certain amount of redundancy.
- For our purposes, *redundancy* is the provision of functional capabilities that would be unnecessary in a fault-free environment.
- This can be a replicated hardware component, an additional check bit attached to a string of digital data, or a few lines of program code verifying the correctness of the program's results.
- Two kinds of redundancy are possible: space redundancy and time redundancy.

Failure masking by redundancy

- If a system is to be fault tolerant, the best it can do is to try to hide the occurrence of failures from other processes. The key technique for masking faults is to use redundancy.
- Three kinds are possible:

1. Information redundancy,
2. Time redundancy, and
3. Physical redundancy

Failure masking by redundancy

- With **information redundancy**, extra bits are added to allow recovery from garbled bits. For example, a Hamming code can be added to transmitted data to recover from noise on the transmission line.
- Physical redundancy is a well-known technique for providing fault tolerance. It is used in biology (mammals have two eyes, two ears, two lungs, etc.), aircraft (747s have four engines but can fly on three), and sports (multiple referees in case one misses an event).
- **Physical redundancy** provides additional components, functions, or data items that are unnecessary for fault-free operation.

Fault Tolerance and Redundancy

- With physical redundancy, extra equipment or processes are added to make it possible for the system as a whole to tolerate the loss or malfunctioning of some components. Physical redundancy can thus be done either in hardware or in software. For example, extra processes can be added to the system so that if a small number of them crash, the system can still function correctly. In other words, by replicating processes, a high degree of fault tolerance may be achieved.
- With **time redundancy**, an action is performed, and then, if need be, it is performed again. The computation or data transmission is repeated and the result is compared to a stored copy of the previous result. Transactions use this approach. If a transaction aborts, it can be redone with no harm.

Dependable Systems

- Being fault tolerant is strongly related to what are called **dependable systems**.
- The ultimate goal of fault tolerance is the development of a dependable system.
- In broad terms, *dependability* is the ability of a system to deliver its intended level of service to its users.
- This term covers a number of useful requirements which are expected from a distributed system.

Dependability Attributes

- Major dependability attributes are
  1. *Reliability*
  2. *Availability*
  3. *Safety*
  4. *Maintainability*
- Other possible attributes include
  - *testability*,
  - *performability*
  - *security*.
- Depending on the application, one or more of these attributes may be needed to appropriately evaluate a system behavior.

Dependability Attributes

- For example, in an Automatic Teller Machine (ATM), the proportion of time in which the system is able to deliver its intended level of service (system availability) is an important measure.
- For a cardiac patient with a pacemaker, continuous functioning of the device is a matter of life and death. Thus, the ability of the system to deliver its service without interruption (system reliability) is crucial.
- In a nuclear power plant control system, the ability of the system to perform its functions correctly or to discontinue its function in a safe manner (system safety) is of uppermost importance.

Reliability

- **Reliability** refers to the property that a system can run continuously without failure.
- *Reliability R(t)* of a system at time *t* is the probability that the system operates without a failure in the interval [0, *t*], given that the system was performing correctly at time 0.
- Reliability is a measure of the continuous delivery of correct service. High reliability is required in situations when a system is expected to operate without interruptions.
- For example, a spacecraft mission control system is expected to provide uninterrupted service. A flaw in the system is likely to cause the destruction of the spacecraft.

Availability

- Relatively few systems are designed to operate continuously without interruption and without maintenance of any kind. In many cases, we are interested not only in the probability of failure, but also in the number of failures and, in particular, in the time required to make repairs. For such applications, the attribute which we would like to maximize is the fraction of time that the system is in the operational state, expressed by availability.
- **Availability** is defined as the property that a system is ready to be used immediately.
- *Availability A(t)* of a system at time *t* is the probability that the system is functioning correctly at the instant of time *t*.

Availability

A highly available system is one that will most likely be working at a given instant in time.

Availability	**(%) Downtime**
90	36.5 days/year
99	3.65 days/year
99.9	8.76 h/year
99.99	52 min/year
99.999	5min/year
99.9999	31 s/year

Availability Examples

- A telephone subscriber expects to complete a call without interruptions. However, a downtime of a few minutes a year is typically considered acceptable.
- Surveys show that the average expectation of an online shopper for a web page to load is 2 s. This means that e-commerce web sites should be available all the time and should respond quickly even when a large number of shoppers access them simultaneously.
- Another example is the electrical power control system. Customers expect power to be available 24 h a day, every day, in any weather condition. A prolonged power failure may lead to health hazards and financial loss.

Availability vs Reliability

- In contrast to availability, reliability is defined in terms of a time interval instead of an instant in time.
- A highly-reliable system is one that will most likely continue to work without interruption during a relatively long period of time.
- This is a subtle but important difference when compared to availability. If a system goes down for one millisecond every hour, it has an availability of over 99.9999 percent, but is still highly unreliable.
- Similarly, a system that never crashes but is shut down for two weeks every August has high reliability but only 96 percent availability.

Safety

- **Safety** refers to the situation that when a system temporarily fails to operate correctly, nothing catastrophic happens.
- For example, many process control systems, such as those used for controlling nuclear power plants or sending people into space, are required to provide a high degree of safety. If such control systems temporarily fail for only a very brief moment, the effects could be disastrous.
- *Safety S(t)* of a system at time *t* is the probability that the system either performs its function correctly or discontinues its operation in a fail-safe manner in the interval [0, *t*], given that the system was operating correctly at time 0.

Safety

- Safety can be considered as an extension of reliability, namely reliability with respect to failures that may create safety hazards.
- From the reliability point of view, all failures are equal. For safety considerations, failures are partitioned into *fail-safe* and *fail-unsafe* ones.
- As an example, consider an alarm system. The alarm may either fail to function correctly even though a danger exists, or it may give a false alarm when no danger is present. The former is classified as a fail-unsafe failure. The latter is considered a fail-safe one.

Maintainability

- **Maintainability** refers to how easy a failed system can be repaired.
- A highly maintainable system may also show a high degree of availability, especially if failures can be detected and repaired automatically.
- However, automatically recovering from failures is easier said than done.

Dependability Impairments

- Dependability impairments are usually defined in terms of faults, errors, or failures.
- A common feature of the three terms is that they give us a message that something went wrong.
- The difference is that, in the case of a fault, the problem occurred on the physical level; in the case of an error, the problem occurred on the computational level; in the case of a failure, the problem occurred on a system level.

Faults, Errors, and Failures

- A *fault* is a physical defect, imperfection, or flaw that occurs in some hardware or software component. Examples are a short circuit between two adjacent interconnects, a broken pin, or a software bug.
- An *error* is a deviation from correctness or accuracy in computation, which occurs as a result of a fault. Errors are usually associated with incorrect values in the system state. For example, a circuit or a program computed an incorrect value, or incorrect information was received while transmitting data.
- A *failure* is a nonperformance of some action which is due or expected. A system is said to have a failure if the service it delivers to the user deviates from compliance with the system specification for a specified period of time.

Faults, Errors, and Failures

- A system is said to **fail** when it cannot meet its promises.
- In particular, if a distributed system is designed to provide its users with a number of services, the system has failed when one or more of those services cannot be (completely) provided.
- An **error** is a part of a system's state that may lead to a failure.
- For example, when transmitting packets across a network, it is to be expected that some packets have been damaged when they arrive at the receiver. Damaged in this context means that the receiver may incorrectly sense a bit value (e.g., reading a 1 instead of a 0), or may even be unable to detect that something has arrived.
- The cause of an error is called a **fault**. Clearly, finding out what caused an error is important. For example, a wrong or bad transmission medium may easily cause packets to be damaged

Faults, Errors, and Failures

- Faults are reasons for errors and errors are reasons for failures.
- For example, consider a power plant in which a computer-controlled system is responsible for monitoring various plant temperatures, pressures, and other physical characteristics. The sensor reporting the speed at which the main turbine is spinning breaks. This fault causes the system to send more steam to the turbine than is required (error), over-speeding the turbine, and resulting in the mechanical safety system shutting down the turbine to prevent it being damaged. The system is no longer generating power (system failure, fail-safe).
- Not every fault causes an error and not every error causes a failure.

Faults Classification

- **Transient faults** occur once and then disappear. If the operation is repeated, the fault goes away. A bird flying through the beam of a microwave transmitter may cause lost bits on some network. If the transmission times out and is retried, it will probably work the second time.
- An **intermittent fault** occurs, then vanishes of its own accord, then reappears, and so on. A loose contact on a connector will often cause an intermittent fault. Intermittent faults cause a great deal of aggravation because they are difficult to diagnose. Typically, when the fault doctor shows up, the system works fine.
- A **permanent fault** is one that continues to exist until the faulty component is replaced. Burnt-out chips, software bugs, and disk head crashes are examples of permanent faults.

Failure Models

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 a 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 any time

Dependability Means

- *Dependability means* are the methods and techniques enabling the development of a dependable system.
- Fault tolerance is one of these methods that is used in combination with other methods to attain dependability, such as fault prevention, fault removal, and fault forecasting.

Fault Tolerance

- Fault tolerance targets the development of systems which function correctly in the presence of faults. Fault tolerance is achieved by using some kind of redundancy. The redundancy allows a fault either to be *masked*, or *detected*, with subsequent location, containment, and recovery.
- Redundancy is necessary, but not sufficient for fault tolerance. For example, two duplicated components connected in parallel do not make a system fault-tolerant, unless some form of monitoring is provided, which analyzes the results and selects the correct one.

Fault Tolerance

- *Fault masking* is the process of insuring that only correct values get passed to the system output in spite of the presence of a fault.
  - For example, a memory protected by an error-correcting code corrects the faulty bits before the system uses the data.
  - Another example of fault masking is triple modular redundancy with the majority voting.
- *Fault detection* is the process of determining that a fault has occurred within a system.
  - *Acceptance tests* is a fault detecting mechanism that can be used for systems having no replicated components.
  - *Comparison* is an alternative technique for detecting faults, used for systems with duplicated components. The output results of two components are compared. A disagreement in the results indicates a fault.

Fault Tolerance

- *Fault location* is the process of determining where a fault has occurred.
  - A failed acceptance test cannot generally be used to locate a fault. It can only tell that something has gone wrong. Similarly, when a disagreement occurs during the comparison of two modules, it is not possible to tell which of the two has failed.
- *Fault containment* is the process of isolating a fault and preventing the propagation of its effect throughout the system.
  - Once a faulty component has been identified, a system *recovers* by reconfiguring itself to isolate the faulty component from the rest of the system and regain operational status. This might be accomplished by having the faulty component replaced by a redundant backup component. Alternatively, the system could switch the faulty component off and continue operation with a degraded capability. This is known as *graceful degradation*.

Fault Prevention

- *Fault prevention* is a set of techniques attempting to prevent the introduction or occurrence of faults in the system in the first place. Fault prevention is achieved by quality control techniques during the specification, implementation, and fabrication stages of the design process.
- For hardware, this includes design reviews, component screening, and testing.
- For software, this includes structural programming, modularization, and formal verification techniques.

Fault Removal

- *Fault removal* is a set of techniques targeting the reduction of the number of faults which are present in the system.
- During the development phase, fault removal involves three steps: *Verification* is the process of checking whether the system meets a set of given conditions. If it does not, the other two steps follow: the fault that prevents the conditions from being fulfilled is *diagnosed* and the necessary *corrections* are performed.
- During the operational life of the system consists *preventive maintenance* (parts are replaced, or adjustments are made before failure occurs) and *corrective maintenance* (performed after the failure has occurred in order to return the system to service as soon as possible).

Fault Forecasting

- *Fault forecasting* is a set of techniques aiming to estimate how many faults are present in the system, possible future occurrences of faults, and the consequences of faults.
- Fault forecasting is done by performing an evaluation of the system behavior with respect to fault occurrences or activation.
- The evaluation can be *qualitative*, which aims to rank the failure modes or event combinations that lead to system failure, or *quantitative*, which aims to evaluate in terms of probabilities the extent to which some attributes of dependability are satisfied.

Exercises

- Compute the downtime per year for availability = 90, 75, and 50%.
- A telephone system has less than 3min per year downtime. What is its availability?