Distributed Systems

June 4th 2019

A distributed system is a collection of entities, each of which is autonomous, programmable, asynchronous, and failure-prone, and which communicate through an unreliable medium.

While we should design for failure, we should still try to make the system as reliable as possible. 7 properties we should achieve for are:

1) Fault-tolerant: Recover from component failures w/o incorrect actions
2) Highly Available: Can restore operations, resuming services even when some components fail.
3) Recoverable: Failed components can restart & rejoin.
4) Consistent: Can co-ordinate actions, to act like a non-distributed system.
5) Scalable: Can operate correctly when scaled up.
6) Predictable: Responsive and doesn't take too long
7) Secure:

To design for failure, the following assumptions cannot be made:

The 8 Fallacies

- The network is reliable: The most common type of error are network failures. Thus, the programmer must write network error-handling.

  Examples include: stalling/waiting indefinitely for an answer packet which consume resources, not retrying stalled operations when the network becomes available again, etc.

June 4th 2019

- Latency is zero: This can cause packet loss, and if unbounded traffic is implemented, then a lot of bandwidth is wasted, increasing dropped packets.
- Bandwidth is infinite: May result in bottlenecks if bandwidth limits are ignored.
- The network is secure: Complacency may result in compromise
- Topology doesn't change: Changes in topology can have latency and bandwidth changes.
- There is one administrator: Multiple administrators may introduce conflicting policies of which senders must be aware of to reach their desired paths.
- Transport cost is zero: "Hidden" costs of building & maintaining a network are non-negligible and should be noted in budgets to avoid shortfalls.
- The network is homogeneous: Assumption of a homogeneous network can lead to networking issues, like the other fallacies.

The most important takeaway is to ensure that as many failure scenarios are taken care of as humanly thinkable.

Design Principles

Failure scenarios must be tested for, and explicitly listed. Both servers & clients must be able to deal w/ unresponsive senders/receivers.

Attempt to minimize the amount of data sent over the network to reduce bandwidth.