Table of Contents

Challenges

Heterogeneity

Openness

Security

Scalability

Ability to handle growth of number of users

- cost of resources: should scale linearly with number of users
- performance loss: dependent on underlying distributed algorithms
  - $O(n^2)$ would not be scalable
  - $O(\log n)$ is scalable
- resources running out: e.g. IPv4 addresses; use of particular data types
- performance bottlenecks: resolved with decentralised architectures/algorithms

Failure Handling

- **detection**: some failures can be detected; others are hard to be certain of, e.g. remote server failure
  - e.g. checksums
- **masking**: hide/ameliorate effects of failure
  - e.g. message retransmission after timeout
  - e.g. Zoom may mask errors by interpolating frames
- **tolerating**:
- **recovery**:
- **redundancy**:

Concurrency

- multiple clients accessing a shared resource

- sequential access can handle it but slows down system
- starting up a system also poses concurrency issues: comment that if all of Google was switched off it couldn't be restarted

Transparency

- system doing something at a lower level that isn't seen by the user
- e.g. on a mobile phone, while moving around, the hardware will change between cells and frequencies without user being aware of it
- sometimes you don't want transparency: some failures need to involve the user e.g. if the Ethernet cable is unplugged
- **access transparency**: e.g. distributed file system. API you use to access files locally should be the same as the API you use to access remotely. That way applications don't need to change depending on whether you're accessing locally/remotely.

- **location transparency** is more about server side-changes, while **mobile transparency** is more about client-side changes

QoS

World Wide Web