Fault-tolerant Computing

Frans Kaashoek
Where are we in 6.033?

• Modularity to control complexity
  • Names are the glue to compose modules
• Strong form of modularity: client/server
  • Limit propagation of errors
• Implementations of client/server:
  • In a single computer using virtualization
  • In a network using protocols
• Compose clients and services using names
  • DNS
How to respond to failures?

• Failures are contained; they don’t propagate
  • Benevolent failures
• Can we do better?
  • Keep computing despite failures?
  • Defend against malicious failures (attacks)?
• Rest of semester: handle these “failures”
  • Fault-tolerant computing
  • Computer security
Fault-tolerant computing

• General introduction: today
  • Replication/Redundancy
• The hard case: transactions
  • updating permanent data in the presence of concurrent actions and failures
• Replication revisited: consistency
A fatal exception 0E has occurred at 0028:C00068F8 in PPT.EXE<01> + 000059F8. The current application will be terminated.

* Press any key to terminate the application.
* Press CTRL+ALT+DEL to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue
Availability in practice

- Carrier airlines (2002 FAA fact book)
  - 41 accidents, 6.7M departures
  - 99.9993% availability
- 911 Phone service (1993 NRIC report)
  - 29 minutes per line per year
  - 99.994%
- Standard phone service (various sources)
  - 53+ minutes per line per year
  - 99.99+%
- End-to-end Internet Availability
  - 95% - 99.6%
Disk failure conditional probability distribution

Infant mortality

Burn out

Stable failure period

1 / (reported MTTF)

Expected operating lifetime

Bathtub curve
Fail-fast disk

failfast_get (data, sn) {
    get (s, sn);
    if (checksum(s.data) = s.cksum) {
        data ← s.data;
        return OK;
    } else {
        return BAD;
    }
}
Careful disk

careful_get (data, sn) {
    r ← 0;
    while (r < 10) {
        r ← failfast_get (data, sn);
        if (r = OK) return OK;
        r++;
    }
    return BAD;
}
Durable disk (RAID 1)

durable_get (data, sn) {
    r ← disk1.careful_get (data, sn);
    if (r = OK) return OK;
    r ← disk2.careful_get (data, sn);
    signal(repair disk1);
    return r;
}