Fault-tolerant Computing

6.033 Spring 2007
April 4, 2007

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

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
  - 99.99% availability
- Standard phone service (various sources)
  - 53+ minutes per line per year
  - 99.99% availability
- End-to-end Internet Availability
  - 95% - 99.6%
### Careful disk

```c
void 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)

```c
void 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;
}
```