CSE/EE 461 Lecture 26
Course Wrapup

Tom Anderson
tom@cs.washington.edu

Security Lessons

- Hard to resecure a machine after penetration
  - how do you know you've removed all the backdoors?
- Hard to detect if machine has been penetrated
  - Western Digital example
- Any system with bugs is vulnerable
  - and all systems have bugs: fingerd, ping of death, Code Red, nimda)

Soapbox

- Information = property
  - is it ok to break into a computer system if you don't intend to steal anything -- just to look around?

Course Topics

- Internet architecture
  - how a web request works, from click to display
    - DNS lookup, connection setup, request/response to server, IP routing, media access, wire signalling, …
  - end to end principle
- Link layer
  - Signal transmission
  - Checksums and CRC's
  - Media access (Ethernet)

Course Topics

- Routing (IP)
  - forwarding and addressing mechanics
  - link state and distance vector routing (OSPF)
  - interdomain routing (BGP)
  - server load balancing and NATs
- Transport (TCP)
  - ARQ and sliding window
  - Connection setup/teardown and flow control
  - Remote procedure call
  - Congestion control: RTT estimation and window size

Course Topics

- Services
  - DNS lookup, caching and replication
  - distributed cache coherence
- Multicast
  - forwarding, routing, retransmission, congestion control
- Real-time
  - scheduling and buffer management
  - resource reservations
- Security
  - encryption and why that's not enough

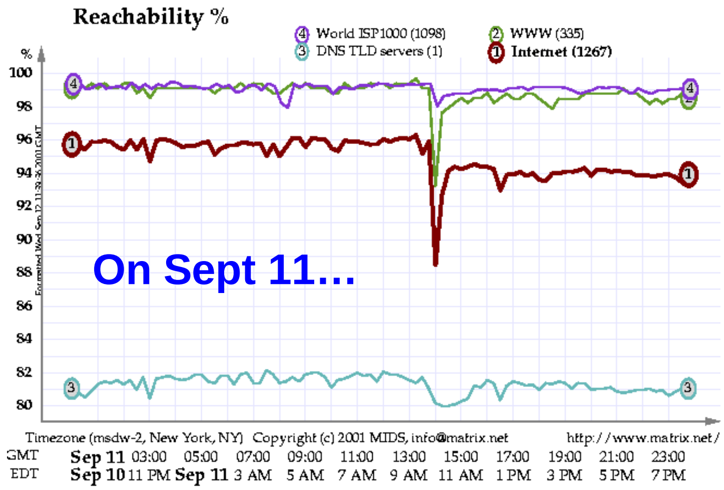
Internet Design Principles

- End to end principle
  - Expect failures to occur at every step, so end hosts must be ultimately responsible for error recovery
  - example: TCP checksum, sliding window
- Soft state
  - if possible, state should be recoverable after a failure
  - example: link state routing messages are resent periodically, whether needed or not
- Design for scalability
  - using backoff: Ethernet, TCP congestion control
  - using hierarchy: IP addresses, DNS, routing (BGP)
  - using neighbors: IGMP, multicast retransmissions

The Future: Reliability

- Internet has ~ 98-99% uptime
  - measured end to end: can two hosts communicate?
  - telephone network: 99.99% uptime
  - air traffic control: 99.999% uptime
- How do we build more reliable systems?
  - Internet effective at masking router/link failures
    - "fail stop" errors: system crashes and reboots
  - Not as good at more arbitrary failures
    - Operational mistakes, programming errors, malicious attacks

How robust is the Internet to fail-stop problems?

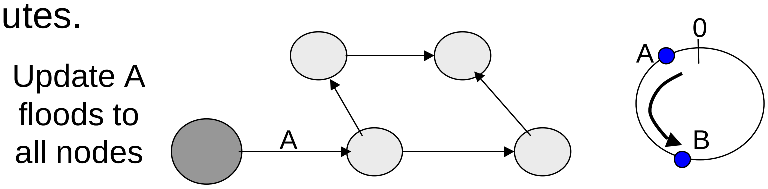


What about arbitrary failures?

- Lots of examples where more arbitrary failures have caused large problems
  - misconfigured routers at Virginia ISP (AS7007) advertised zero cost routes to everywhere (April 97)
  - caused nearby AS's to send all their traffic to that AS
  - disrupted connectivity for hours
  - Another example (RFC 2525, 1999): 18 TCP bugs known to be lurking out there
- Thesis: Need a new protocol design methodology to prevent these kinds of problems

ARPANET Link-state Flooding

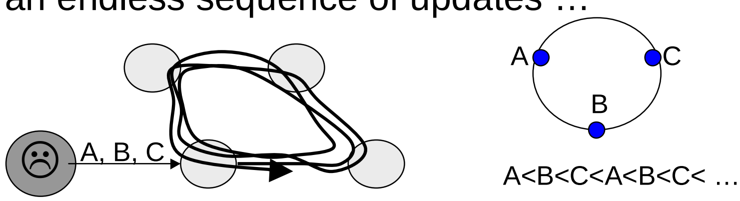
- In link state routing, routers exchange updates with their neighbors. These are flooded so they reach everyone. Then they are used to calculate routes.



- Sequence numbers are used to order updates. ARPANET used modulo arithmetic to decide which update is new.

Problem – an endless flood

- One night the ARPANET stopped working. A corrupt router had injected messages that led to an endless sequence of updates …

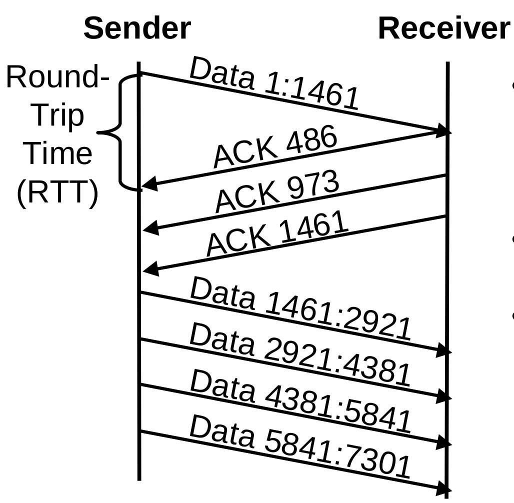


- This was hard to fix – purge entire network of bad data

Solution: reset, don't wrap #s

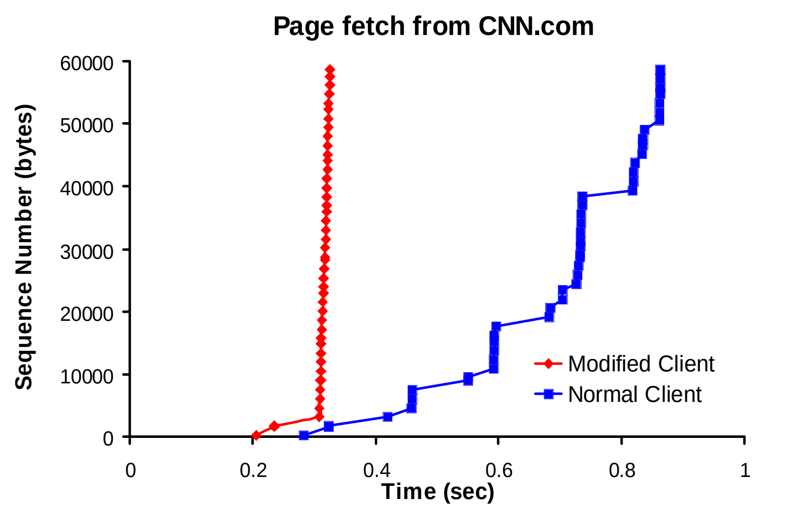
- Sequence numbers taken from a large, linear space
- Now repeated updates in any order cannot be interpreted as new and cause an endless cycle
  - New work requires fresh messages to be injected by routers
- We use aging to purge an update with maximum sequence number, should that arise.

TCP Congestion Control



- Rule: grow window by one full-sized packet for each valid ACK received
- Send M ACKs for one pkt
- Growth factor proportional to M (x^M in slow-start!)

TCP Daytona Performance

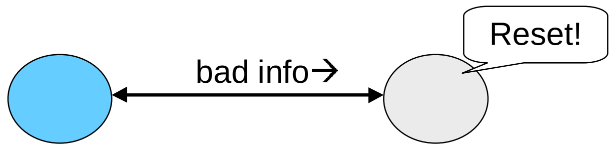


Solution: Require Proof

- Solution against ack splitting
  - check that entire packet is ack'ed before opening window
- More generally
  - client can spoof fast recovery by sending large # of duplicate acks (after halving cwnd, each dupack increases cwnd by 1)
  - client can ack before actually receiving packet
- Solution: add random bit to packet; receiver must echo back to sender to prove receipt

BGP Error Handling

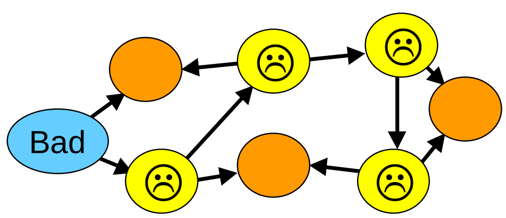
- In BGP routing, peers exchange announcements over a TCP connection and use them to select forwarding paths



- If bad information is received by a peer, which of course shouldn't happen, it resets the connection and retries.

Problem – errors can be magnified

- Some routers pass on bad info rather than reset (yellow)
- Bad info propagates much further than otherwise
- Many "correct" routers see the bad info and reset (orange)



- This caused a widespread outage in October 2001

Solution: weed out individual errors

- Add error checking at a finer granularity
  - Individual routes rather than whole peering sessions
- Correct behavior is then to drop individual errors
- Bad behavior, which passes errors, doesn't hurt as much

- BGP spec being revised in NANOG and IETF.

Broader Question

- How do we design protocols so that errors don't happen and/or if they do, they don't have widespread effect?
  - end to end principle & soft state help with fail-stop failures, but not with implementation/operator error
  - neither do encryption, more complete specs, …
- *Defensive* protocol design
  - expect protocol and implementation errors, and design system to be robust in face of problems

Defensive Protocol Design

- Minimize dependencies
  - clean simple interfaces with as little interdependence as possible
- Verify information
  - add redundancy so that nodes can check information provided by other nodes
- Protect resources
  - e.g., against DoS attacks
- Contain faults
  - so problems don't propagate
- Expose errors
  - end to end failure recovery hides problems, reduces likelihood problems will be fixed