Design Guidelines for Robust Internet Protocols

Rahul Singhai
KReSIT, IIT Bombay

Reference Paper : Design Guidelines for Robust Internet Protocols by Anderson, Shenker, Stoica & Wetherall. ACM SIGCOMM 2002.

Internet Robustness

- Robustness was one of the Internet's original design goals
- Adopted failure-oriented design style:
  - Hosts responsible for error recovery
  - Critical state refreshed periodically
  - Failure assumed to be the common case
- Internet has withstood some major outages with minimal service interruption
  - Hurricanes, Tunnel Fires, Earth-quakes, terrorist attacks, etc.

Traditional Failure Models

- Most Internet protocols designed with assumption that a node participates in fail-stop manner :
  - Node fails completely & is detected by other nodes
- But these protocols are usually vulnerable to participating nodes misbehaving:
  - Malicious nodes
    - Denial-of-service, spoofing, etc.
  - Misconfigured nodes
  - Bug in software

Semantic vs. Syntactic Failures

- Syntactic failures:
  - Node responds with ill-formed message
- Semantic failure:
  - Node responds with well-formed message, that is semantically incorrect

Examples

- CodeRed & Nimda worms triggered routing problems due to BGP instabilities
- Incorrect Sequence number in earlier link state protocols
- Routing Misconfiguration
  - Router advertises short route to some prefix
  - All other routers believe
- Congestion Control
  - Hosts can ignore TCP CC and send too fast
  - Receiver can lie to sender about ECN (Explicit Congestion Notification) bits

Traditional Techniques

- Cryptographic authentication:
  - Verifies who is talking, but not what they say
- Formal verification:
  - Verifies that correct protocol operation leads to the desired result
- Fault-tolerance via consensus: (Byzantine techniques)
  - Requires that several nodes have enough information to do the required computation
  - In network routing, for instance, only the nodes at the end of a link know about its existence

Design Guidelines for Robust Protocols

- Already Existing Basic Guidelines
  - How & where to keep state
    - Critical state should be refreshed periodically
  - How & where to initiate recovery
    - End-point error recovery
- Design Defensively
  - Incoming information might be incorrect
  - Nodes might be malicious or broken

Guideline # 1: Value Conceptual Simplicity

- Interfaces becoming complex over time (backward compatibility)
- Complex designs harder to correctly implement by all parties
- Use clean & simple interfaces, functional semantics
- Example : BGP route oscillation
  - Multi-Exit Discriminators are used to specify ingress preferences
  - Route reflector tells best route information to all BGP routers
  - Using MED attributes for selecting best route can cause persistent route oscillations, because they are not comparable, when learned from different ASs

Guideline # 2: Minimize Your Dependencies

- If nodes can be untrustworthy, then reduce dependence on information from them
- Example: TCP Congestion Control
  - Sender trusts receiver for congestion information
  - TCP fast recovery uses DUPACK to retransmit a missing packet
  - Problem: A misbehaving receiver can send infinite stream of DUPACKs
  - Fix: Use selective ACKs. Receiver explicitly lists which packets are acknowledged (either negatively or positively)

Guideline # 3: Verify When Possible

- Actively test node's responses
- Compare data to information provided
- Example: Explicit Congestion Notification
  - Routers marks ECN when congested
  - Receivers returns marks via ACKs
  - Problem:
    - Receiver may fail to mark
    - Mark removed by firewalls normalizing IP header fields or VPN Boxes striping off outer IP header
  - Fix:
    - Use nonce. Receivers echo nonce sum in ACKs

Guideline # 4: Protect Your Resources

- Resource Exhaustion by unsolicited requests from unauthenticated parties
- Example: TCP Connection Setup
  - Connection state maintained at least until three-way handshake completes or time-out happens
  - SYN flood denial-of-service attacks
  - Fix:
    - Send back state with SYN-ACK & keep on initiating client
    - SYN cookies used to verify states sent back at time of handshake

Guideline # 5: Limit Scope of Vulnerability

- Errors cannot always be caught or prevented
- Design protocols to limit possible damage or cascading of effects of errors
- Examples:
  - BGP Route flapping:
    - Route announcements & withdrawals were propagated throughout Internet
    - Every location saw all route flaps in entire Internet
    - Route flap damping limits extent to which failures propagate
    - It holds down routes that are repeatedly change status
  - BGP Error Processing
    - One vendor’s BGP implementation ignores but propagates incorrect route announcements, while another vendor’s routers terminate & restarts any BGP session propagating obvious incorrect announcements

Guideline # 6: Expose Errors

- Don’t let errors trigger other bigger problems
  - Search & fix errors before they cascade
- Example: TCP Checksum failure
  - A bug in sender host or router corrupts data
  - Receivers ignored packet
  - Sender retransmits

Questions?

Guideline #6 Example

- ISP filters:
  - Prevents ISP from being used as transit by other customers
  - Filters customer advertised routes based on their prefixes
  - Link failure to a multi-homed customer may result in ISP providing transit to other ISP
  - Fix: Filter on AS-PATH rather than prefixes