On Protocol Design

T-110.4100 Computer Networks
13.10.2008
Miika Komu <miika@iki.fi>
Helsinki Institute for Information Tech.

Table of Contents

Goals and Requirements

- Need to exchange information between two or more devices → need for a protocol
  - The usage scenarios are mapped to protocol engineering goals and requirements
- Can't have everything: goals usually conflict with each other, need to prioritize
  - Reliable vs. fast
  - Versatile vs. simple
- Do not overlook economics: money, time and people set the limits for goals and requirements

Design and Specification

- Three technical aspects:
  - Host processing: protocol states, transitions, retransmissions, ordering of packets
  - What goes on wire: serialization, formatting, framing and fragmentation, messages, round trips
  - Reality: implementation complexity, performance
- KISS = Keep It Simple Stupid!
- Design it as simple as you can, but not simpler
- Reuse/extend existing design or protocol if possible

Requirements for Protocols

- Reliability
- Error correction
- Packet ordering
- Congestion control
- Availability
- Error detection
- One-to-one vs. one-to-many
- Zero conf vs. manual
- Mobility
- Multihoming
- Security
- Privacy
- Middlebox aware
- Energy efficiency

Failure Tolerance

- Retransmissions (e.g. in WLAN)
  - Timeouts, acknowledgments and window size
- Failover mechanisms
  - Network malfunction
  - Implementation crash and reboot
  - Host reboot
- Are all corner cases covered?
  - Protocol error handling
  - Simultaneous connection initiation

Scalability

- State explosion (at middleboxes)
- Computational overhead and complexity
  - Small devices with limited CPU and batteries
- Decentralization (distributed protocols)
  - Load balancing vs. fault tolerance
- Caching for performance
- Adaptability

Interoperability

- Multiple implementations from different vendors or organizations
  - Are the implementations compatible?
  - Is the specification strict enough?
- Be conservative in sending and liberal in receiving
- Specification is a guideline: interoperability between real-world implementations more important in practice

Compatibility

- Incompatible protocols should reject communications with each other
  - For example v1 and v2 protocol
- Mandatory and optional protocol parameter
  - Optional parameters: future compatibility
- Extension compatibility
  - Do all of the N extensions work together?
- Backwards and forwards compatibility

Network Environment

- Single-hop vs. Multi-hop
- Access Media
  - Wired vs. wireless media
- LAN, WAN
- NATted/IPv4 vs. IPv6 networks
- Multihoming, multiaccess, multipath
- Mobility: host mobility, network mobility
- Infrastructure: name servers, middleboxes
- Interplanetary networks

Protocol Models

- Architectural models
  - Client-server vs. p2p
  - Centralized vs. distributed
  - Cloud computing
  - Publisher vs. subscriber
- Communication models
  - Unicast, anycast, broadcast, multicast
  - Point-to-point vs. end-to-end
  - End-to-end vs. end-to-middle
  - Internet routing vs. overlay routing

Layering

- On which layer should the protocol operate?
  - TLS vs. IPsec
- Application layer: more intelligent decisions, easier to implement, easier to deploy
  - Application frameworks and middleware
- Lower layers: generic purpose “service” to application layer => software reuse
- Strict vs. loose layering (cross-layer interaction)

Addressing and Naming

- Human readable
  - Hostnames, FQDN, URIs
  - Subject to internationalization issues
- Machine readable
  - Operator or device manufacturer assigned (IP address, MAC addresses)
  - Self-assigned addresses (ad-hoc networks)
  - Cryptographic names (PGP, ssh, HIP)

States and Transitions

- State machine models different phases of communication
  - Example: handshake, communications, connection maintenance and teardown
- Stateless operation: operates based on packet contents
- Stateful operation: packet contents + “history”
  - State transitions
  - Symmetric (mirrored) state machine
  - Asymmetric state machine
  - Hard state: state transitions explicitly confirmed and state does not expire
  - Soft state: needs to refreshed, otherwise expires

Flow Diagrams

- Examples of packet flows
- Illustrate the protocol to the reader of the protocol specification
- Usually contain at least two hosts
- Illustrates also the flow of time

Protocol Encoding 1/2

- Serialization (marshalling) to wire format
- PDU, framing, segmentation, MTU
- Text encoding (appl. layer protocols)
  - xml, html, sip
  - easier to debug for humans
  - lines usually separated by newlines
  - character set (internationalization) issues
  - inefficient (compression could be used)

Protocol Encoding 2/2

- Binary formats
  - Integers in Big-Endian format
  - Padding
  - Bandwidth efficient
  - IPv4, Ipv6, TCP, XDR, ASN.1, BER, TLV, etc
- Typically binary formats are visualized in “box notation” for engineers in protocol specifications

Security 1/5

- Better to embed in the design from day one
  - Security difficult to add afterwards to deployed protocols
  - Privacy even more difficult to add afterwards
  - We don't need security – think again!
- Attack pattern
  - Scan, intrude, exploit, abuse, cover tracks
- Protection pattern
  - Detect, prevent, contain

Security 2/5

- Internal vs. external threat
  - Attacker within company or outside
  - Local software (e.g. trojan) vs. remote attack
- Active (write packets) and passive (read packets) attacks
- Man-in-the-middle
- Blind attack
- Link-local attacks vs. remote attacks
- Reflection, amplification, flooding
- DoS vs. DDos attack

Security 3/5

- Security countermeasures:
  - Access control lists, passwords, hashes
  - Public-key signatures and certificates
  - Cryptography
  - Open design vs. security by obscurity
  - Don't forget about user education!
- Attacks against availability: resource depletion / exhaustion (DoS/DDoS), countermeasures:
  - Rate limitation
  - Intermediaries (firewalls, network intrusion detect.)
  - Capthas, computational puzzles

Security 4/5

- Opportunistic security vs. infrastructure
  - Leap of faith/time or huge deployment cost?
- Reuse existing mechanisms: SSL vs. IPsec
  - IPsec does not require changes in the application
  - How does the application know that the connection is secured?
- Find the balance between usability and security
  - Security increases complexity
  - Avoid manual configuration and prompting

Security 5/5

- Do not hard-code crypto algos to the protocol! Use suites and negotiation because algos become vurnerable due to faster machines (Moore's law)
- Murphy's law: everything that can go wrong, will go wrong
  - Hackers will find and abuse holes in the design and implementations
  - The overall strength of the system is as strong as its weakest link!

Protocol Correctness

- Verify that the protocol “works”
  - Implement your own specification!
  - Review from other people
  - Simulation or emulation
  - Mathematical analysis
  - Security analysis
  - Scalability/performance analysis
- Ready for deployment?
  - More difficult to “fix” already deployed protocols and implementations
  - Future compatibility

Deployment Obstacles

- Middlebox traversal
  - Does the protocol go through NATs, routers, proxies and firewalls?
- NAT traversal
  - NATs make protocol engineering difficult
  - Legacy NAT devices all work differently
  - New transport protocols get dropped
  - Tweaking “holes” to NAT boxes
  - Referrals don't work
  - Counter-measures: TCP/UDP encapsulation, hole punching, ICE/STUN or Teredo

IETF Standardization

- Why? More reviewers => better security, compatibility, deployment, scalability
  - Even wizards make errors
  - Why not? Standardization takes time
- Open participation, no membership fee
- Process pattern: BoF -> WG -> drafts -> RFC -> close WG
- Rough consensus and running code
  - To get an RFC, two interoperable implementations are required
- IETF also includes research groups for experimental designs
- IPR: best effort notification about patents
  - Watch out for submarines!