# On Protocol Design

T-110.4100 Computer Networks
20.10.2009

Miika Komu <miika@iki.fi>
Helsinki Institute for Information Technology

# 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

# Design Criteria for Protocols

- Reliability
- Scalability
- Packet ordering
- Congestion control
- Error correction
- Error recovery
- Adaptability
- Availability
- Zero conf
- Mobile systems
- Energy efficiency
- Security
- Privacy
- Anonymity

# Fault Tolerance

- Failures
  - Network malfunction
  - Software or device crashes or reboots
- Retransmissions (e.g. in WLAN)
  - Timeouts, acknowledgments and window size
- How to realize
  - Proper protocol and software error handling
  - Distributed services

# Scalability

- Can the protocol endure drastic increase in number of users?
- State explosion
  - Especially at middleboxes (e.g. routers)
- Computational overhead and complexity
  - Small devices with limited CPU and batteries
- Decentralization (distributed protocols)
  - Load balancing
- Caching for optimized performance
- Network simulators (e.g. NS3)

# Compatibility

- Protocol specifications minimally define on-wire formats
- Backwards incompatible extensions introduced
  - Bump protocol version from v1 to v2
- Mandatory and optional protocol parameters
  - Optional parameters for backwards compatibility
- Extension compatibility
  - Do all of the N extensions work together?

# Interoperability

- Interoperability tests verify compatibility
- Multiple implementations from different vendors or organizations
  - Are the implementations compatible?
  - Is the specification strict enough?
- Be conservative in sending and liberal in receiving
    - Backward & forward compatibility

# Network Environment

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

# Protocol Models

- Architectural models
  - Centralized vs. distributed service
  - Client-server vs. peer-to-peer
    - Cloud computing
  - Publish-subscribe
- 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
  - Asynchronous vs. synchronous

# Layering

- Abstract and isolate different protocol functionality on different layers of the stack
  - A layer should be replaceable with another
- 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)
- Sometimes redundant functionality
  - TLS vs. IPsec

# 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 tear down
- 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

# Packet Flow Diagrams

- Illustrate the protocol to the reader of the protocol specification
- Examples of packet flows between two or more hosts
- Illustrates also the flow of time

# Protocol Encoding 1/2

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

# Protocol Encoding 2/2

- Binary formats
  - Integers in big-endian format
  - Padding for alignment
  - Bandwidth efficient
  - Example protocols: IPv4, IPv6, TCP
  - Example formats: XDR, ASN.1, BER, TLV
- 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
  - Prevent, detect, contain

# Security 2/5

- Internal vs. external threat
  - Attacker within company or outside
  - Local software (e.g. trojan) vs. remote attack
- Active (modify packets) and passive (read packets) attacks
- Man-in-the-middle
- Blind attack
- 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!
- Countermeasures against attacks for availability (resource depletion, exhaustion,DoS/DDoS):
  - 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 user 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 vulnerable 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 software
  - Future compatibility

# Deployment Obstacles

- Middlebox traversal
  - Does the protocol go through NATs, routers, proxies and firewalls?
- NAT traversal
  - NATs make protocol engineering difficult
  - Old NAT devices all work differently
  - New transport protocols get dropped
    - Referrals don't work
  - Manual hole punching for NAT boxes to offer services
  - Counter-measures: relays, TCP/UDP encapsulation, automatic hole punching with 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
  - Two interoperable implementations are required for RFC
- IETF also includes research groups for experimental track
- IPR: best effort notification about patents
  - Watch out for submarines!