On Protocol Design

T-110.4100 Computer Networks
13.10.2010

Miika Komu <miika@iki.fi>
Data Communications Software
CSE / Aalto University

Table of Contents

- Goals & requirements
- Design & specs
- Protocol properties
- Failure tolerance
- Scalability
- Compatibility
- Interoperability
- N/w Environments
- Protocol models
- Layering
- Addressing & Naming
- State & Transitions
- Packet Flow Diagrams
- Protocol Encoding
- Security
- Correctness
- Deployment
- Standardization

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 can conflict with each other:
  - Reliable vs. fast
  - Extensible vs. simple
- Do not overlook economics: money, time and people set the limits for goals and requirements

Protocol Success Factors

- Scalability: from 100 users to 1 million
- Flexibility: application to new use cases
- Incremental deployment
- Does it meet a real (user) need?
- Cost savings
- Zero configuration
- Market (un)certainty
  - Uncertain: modularity & flexibility
  - Certain: fixed & efficient

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
  - Deployment: wireless networks, mobile devices, sensors, firewalls, NATs, etc
- Remember:
  - Design it as simple as you can, but not simpler..
  - Reuse/extend existing design or protocol if possible

Design Criteria for Protocols

- Extensibility
- Reliability
- Scalability
- Availability
- Ordered delivery
- Congestion control
- Error correction
- Error recovery
- Stateless
- Zero configuration
- User centric
- Mobile networking
- Energy efficiency
- Security
- Privacy
- Anonymity

Fault Tolerance

- Failures types
  - Network malfunction
  - Software or device crashes and reboots
- How to achieve?
  - Retransmission, redundancy
  - Session resumption
  - Proper protocol and software error handling
  - Test engineering

Scalability

- Can the protocol endure a drastic increase in the 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 (server redundancy)
- Caching for optimized performance
- Testing with network simulators (e.g. NS3)

Protocol Compatibility

- Protocol specifications define on-wire formats
  - Sometimes include implementation issues
- 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 of two different implementations
- Multiple implementations from different vendors or organizations
  - Are the implementations compatible?
  - Is the specification strict enough?
- Be conservative in sending and liberal in receiving
    - Backwards & forwards compatibility

Network Environments

- Single-hop vs. multi-hop
- Access Media (wired vs. wireless)
- LAN, WAN
- Trusted vs. untrusted networks
- NATted/IPv4 vs. IPv6 networks
- Infrastructure: name servers, middleboxes
- Device mobility, network mobility
- Multihoming, multiaccess, multipath
- Delay tolerant networking (e.g. email)

Protocol Models

- Architectural models
  - Centralized vs. distributed service
  - Client-server vs. peer-to-peer
  - Cloud computing
- 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
  - Byte transfer vs. publish-subscribe

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)

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 detection)
  - 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 algorithms into the protocol!
  - Crypto algorithms are safe only until a flaw is found
  - Key sizes get deprecated due to faster machines
- 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?
- Network Address Translators (NATs)
  - Naming of hosts becomes more difficult
  - NATs make protocol engineering difficult
  - By default, NATs block new incoming connections
  - Penetration by manual pinholing, ICE or Teredo
  - NATs support only TCP and UDP (and maybe IPsec)
  - Old NAT devices have different NAT algorithms

Standardization

- Why?
  - Even wizards make errors; more reviewers, less errors
  - Customer demands?
  - Drawback: standardization takes time
- Few standards organizations
  - W3C: Web standardization
  - IETF: Applications, routing, transport, IPv4/IPv6, security
  - IEEE: Electricity (ethernet, wlan), POSIX, ..
  - ITU-T, ETSI, 3GPP: Cellular technology