Protocol Design

T-110.4100 Computer Networks
16.10.2006

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

Table of Contents

Goals and Requirements

- Need to exchange information between two or more hosts => 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
  - Usable vs. secure

Design and Specification

- Extending an existing protocol vs. creating from scratch
- Three aspects:
  - Host processing: protocol states, transitions, timers, etc
  - Serialized network data and formatting
  - Implementation complexity, performance, etc
- Typical session design: handshake, connection maintenance and teardown
- KISS = Keep It Simple Stupid!

Protocol Correctness

- Verify that the protocol "works"
  - Implement it!
  - Simulation or emulation
  - Mathematical analysis
  - Security analysis
- Ready for deployment?
  - More difficult to "fix" already deployed protocols and implementations

Layering

- On which layer should the protocol operate?
  - Not always clear: e.g. 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 layering vs. layer violations

Addressing and Naming

- Human readable
  - Hostnames, URIs, email-addresses
- Machine readable
  - IP addresses
  - MAC addresses
  - Cryptographic names
    - Public keys (gpg) or fingerprints (ssh)

Network Environment

- Access Media
  - wired vs. wireless media
- Single-hop vs. Multihop networks
- LAN, WAN
- IPv4 and IPv6 networks
- Multihoming and multiaccess
- Multipath
- Mobile host vs. fixed host
- Infrastructure requirements (third host)

Some Protocol Properties

- Reliability
- Duplicate handling
- Congestion control
- Error detection (checksums, CRC)
- Error correction (Reed-Solomon)
- Zero configuration vs. managed
- Multiplexing
- Mobility
- Multihoming
- Security
- Privacy

About State Machines

- Stateless operation
- Stateful operation
  - State transitions
  - Symmetric (mirrored) state machine
  - Asymmetric state machine (receiver and sender state)
  - Hard state
    - state transitions explicitly confirmed
    - state does not expire
  - Soft state
    - needs to be refreshed, otherwise falls back to default state

Protocol Communications

- Unicast, anycast, broadcast, multicast
- Point-to-point vs. end-to-end
- Client/server vs. p2p
- Separate control and data channel
- Internet routing vs. overlay routing
- Strict packet ordering using seq numbers
- Acknowledgments and Automatic Repeat reQuest: wait-ack, nak, go-back-n, sack
  - Window size

Protocol Encoding 1/2

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

Protocol Encoding 2/2

- Binary formats
  - e.g. IPv4, IPv6, TCP
  - Integers in Big-Endian format
  - Padding
  - Saves bandwith when compared to text enc
  - XDR, ASN.1, BER, TLV, etc
- Typically binary formats are visualized in “box notation” for engineers in protocol specifications

Robustness 1/3

- Retransmissions (e.g. WLAN) and timeouts
- Application and host restarts
- Simultaneous handshakes
- DoS and DDos protection
- Timeouts
- Failover mechanisms
- Synchronous vs. asynchronous communication

Robustness 2/3

- Incompatible protocols should reject communication with each other!
  - For example v1 and v2 protocol
- Critical and optional protocol options and negotiation
- Be conservative in sending and liberal in receiving (for interoperability)
  - Specification is a guideline: interoperability between real-world implementations more important in practice

Robustness 3/3

- Design for change and modularity
- Avoid layer violations
  - However: cross-layer interaction
- Design it as simple as you can, but not simpler
- Completeness, consistence and clarity

Security 1/5

- Better to embed in the design from day one
  - 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
  - localhost 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

Security 3/5

- Security countermeasures (with varying levels of protection):
  - Access control lists, passwords, authentication
  - hashchains, HMACs, signatures
  - symmetric cryptography
- Attacks against availability: resource depletion / exhaustion (DoS/DDoS), countermeasures:
  - Rate limitation
  - Computation puzzles
    - Require connection initiator to do some work

Security 4/5

- Reuse existing mechanisms: SSL vs. IPsec
  - IPsec does not require changes in the application
  - How does the application know that the connection is secured?
- Opportunistic security vs. infrastructure
  - Leap of faith/time or huge deployment cost?
- Usability <> security
  - Security increases complexity
  - Avoid manual configuration
- Privacy adds complexity

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!
- Open Design vs. Security by Obscurity
  - Four eye balls is more than two

Scalability

- Backwards and forwards compatibility
- State explosion
- Computational overhead and complexity
  - Small devices with poor CPU and batteries
- Load balancing
- Decentralization
- Caching
- Adaptability
- Efficiency: e.g. MTU and fragmentation

Deployment Obstacles

- Middlebox traversal
  - Does the protocol go through NATs, routers. proxies and firewalls? On what probability?
- NAT traversal
  - NATs make protocol engineering difficult; each protocol has to take care of NATs redundantly
  - Deployed NAT devices work differently!
  - New transport protocols get dropped
  - Server and p2p don't work
  - Referrals don't work
  - Counter-measures: UDP encapsulation, hole punching, STUN, TURN relay, ALGs, MIDCOM

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!