On Protocol Design

T-110.4100 Computer Networks
27.3.2012

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

Scope of the Lecture

- General overview of protocol design
  - Characteristics of successful protocols
  - Different protocol design models
  - Security
  - Standardization
- What this lecture is not about
  - Programming of protocols
  - Web-protocol development

Related Courses at Aalto

- S-38.3159 Protocol Design
- T-106.4300 Web Software Development
- T-75.1110 XML-kuvauskielten perusteet
- T-109.4300 Network Services Business Models
- T-109.5410 Technology Management in the Telecommunications Industry
- T-110.5241 Network Security

Related Courses at Aalto

- S-38.3159 Protocol Design
- T-106.4300 Web Software Development
- T-75.1110 XML-kuvauskielten perusteet
- T-109.4300 Network Services Business Models
- T-109.5410 Technology Management in the Telecommunications Industry
- T-110.5241 Network Security

Introduction to Protocol Design

- Need to exchange information between two or more entities (devices or software)
  - Need a common language (protocol)
- Protocol specifications define on-wire formats
  - Sometimes include implementor “hints”
- Can't have the cake and eat it all
  - Reliable vs. fast
  - Extensible vs. simple
  - Trade-offs often necessary
- Non-technical constraints
  - Money, time and human resources

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
  - Separate policy from mechanism

Initial Success Factors

- Early adopters get the benefits (RFC5218)
  - No extra infrastructure needed
  - Changes required only at the client side
- Meets a real (business) need
- Incrementally deployable
  - No flag day!
- Open code, specs, maintenance and patent free, good technical design
  - Surprisingly, these have secondary value

Wild Success Factors

- “Wild success” means that protocol is deployed beyond original plans
- Factors for success
  - Extensibility
  - Scalability
  - Security threats sufficiently migrated
- See RFC5218 (What Makes for a Successful Protocol)

Technical Design Criteria

- Extensibility
- Reliability
- Scalability
- Stateless servers
- Ordered delivery
- Congestion control
- Error correction
- Error recovery
- Zero configuration
- Incr. deployable
- Rest vs. RPC
- Energy efficiency
- Security
- Privacy
- Availability
- Anonymity

Scalability

- Is the protocol designed to endure a drastic increase in the number of users?
  - Bottlenecks: infrastructure and servers
- Computational overhead and complexity
  - Small devices with limited CPU and batteries
  - Wireless networking consumes energy!
- Decentralization (distributable protocols)
  - Load balancing (server redundancy)
- Caching for optimized performance

Protocol Evolution

- Mandatory vs. optional protocol parameters
  - Optional for backwards compatibility
- Extension compatibility
  - Do all of the N extensions work together?
- Backwards incompatible extensions introduced
  - Bump protocol version from v1 to v2
  - Versioning should be included from day one!
- Be conservative in sending and liberal in receiving

Interoperability

- Interoperability tests verify compatibility of two different implementations
  - Protocol specifications
- Multiple implementations from different vendors or organizations
  - Are the implementations compatible?
  - Is the specification strict enough?

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)

Design Models

- Architectural models
  - Centralized vs. distributed service
  - Client-server vs. peer-to-peer
  - Cloud computing
  - REST vs. RPC
- 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. messaging oriented

Representational State Transfer

- REST Constraints
  - Separation of concerns (client-server)
  - Stateless server
  - Cacheability
  - Support for intermediaries (proxies)
  - Uniform interface
- Benefits
  - Scalability
  - Simple interfaces

REST vs. RPC

- Remote Procedure Call (RPC)
  - Non-uniform APIs
  - Complex operations
- RESTful web is usually a better choice than SOAP
  - Easier to develop and to scale up
  - Web is resource oriented by its nature
- See RESTful Web Services (Richardson)

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)

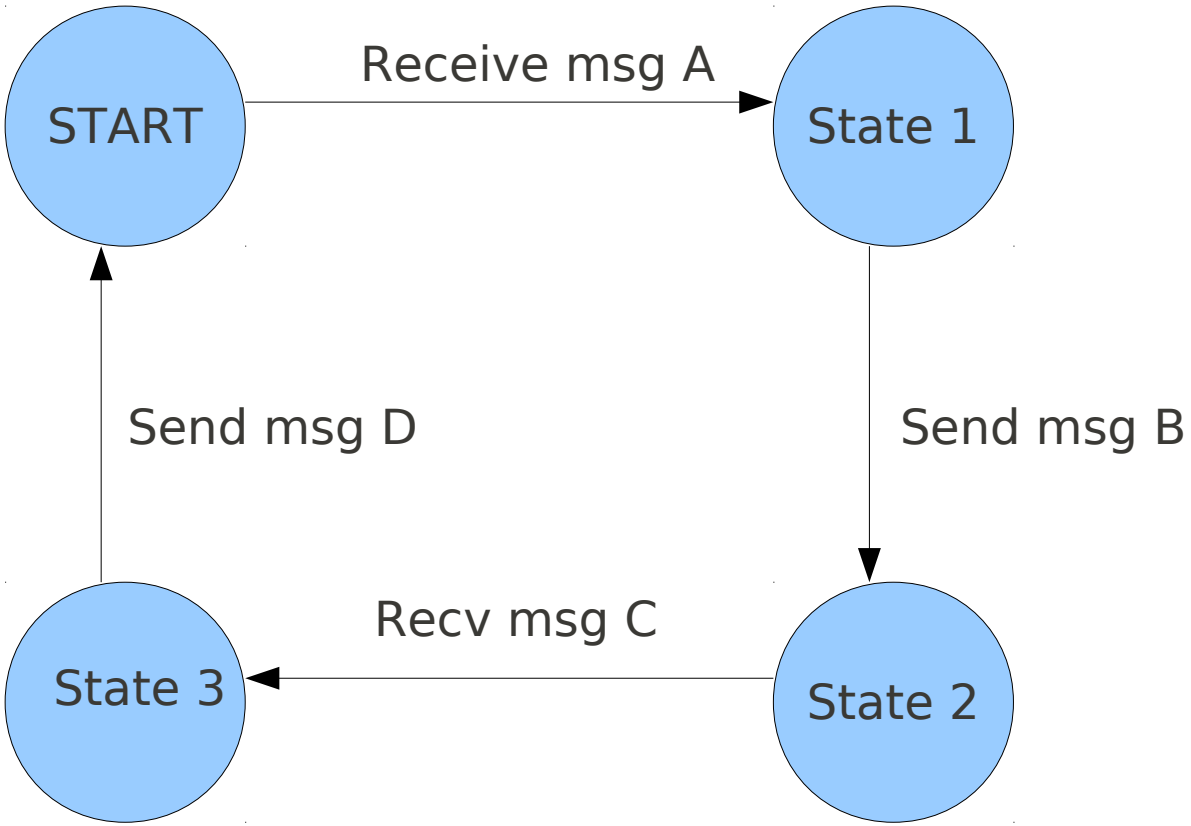
Addressing and Naming

- Human-readable names
  - Hostnames, FQDN, URIs
  - Subject to internationalization issues
- Machine-readable names
  - ISP or device manufacturer assigned (IP address, MAC addresses)
  - Self-assigned addresses (ad-hoc networks)
  - Cryptographic names (PGP, SSH, CGA, HIP)
  - Do not embed IP addresses in application-layer protocols (see RFC5887)
- Translation of names (DNS, directories, etc)

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

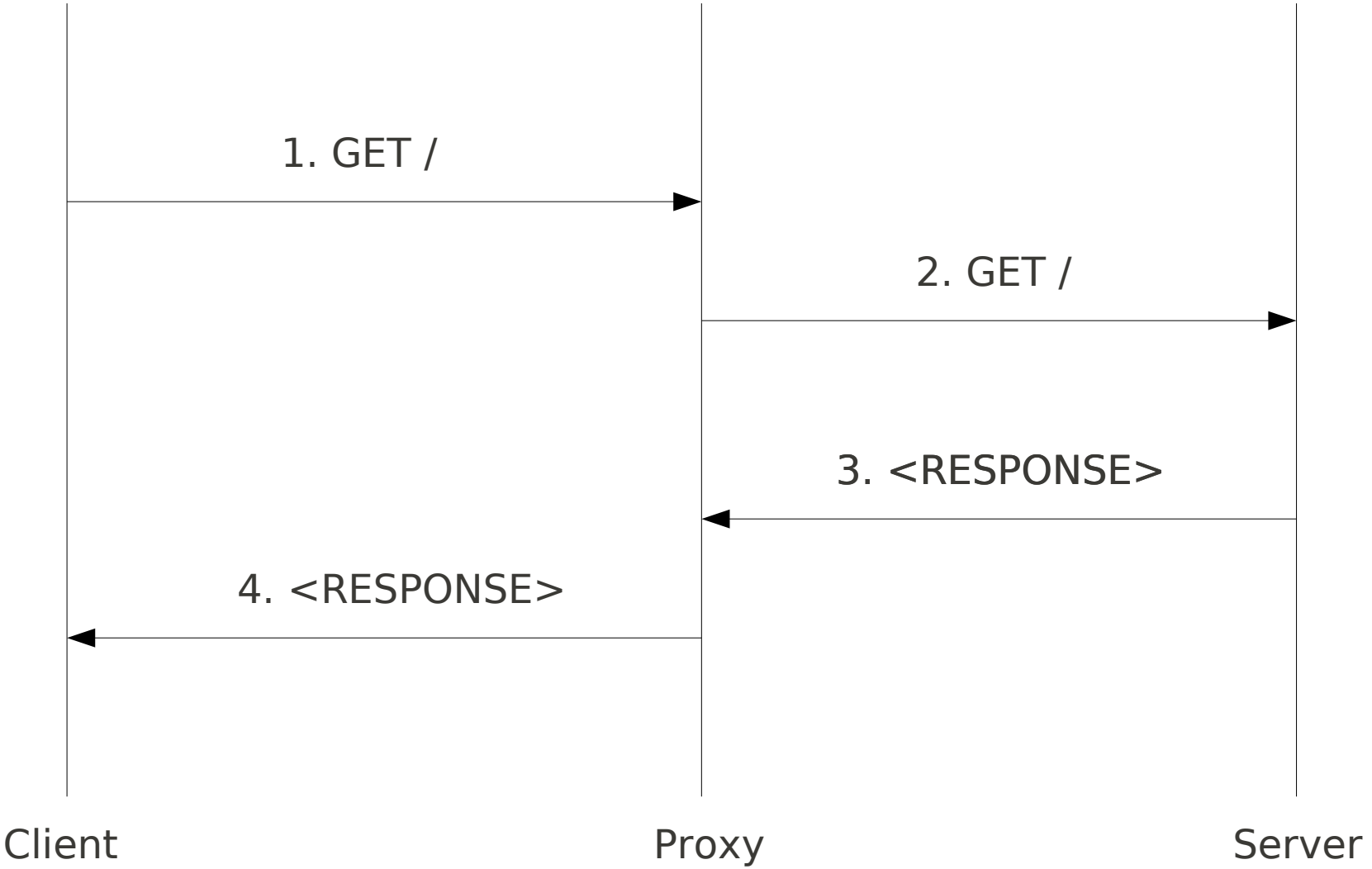
Example State Machine



Packet Flow Diagrams

- Illustrate the protocol to the reader of the protocol specification
  - Use case, not a full specification
- Two or more communicating entities
- Illustrates also the flow of time

Example Flow Diagram



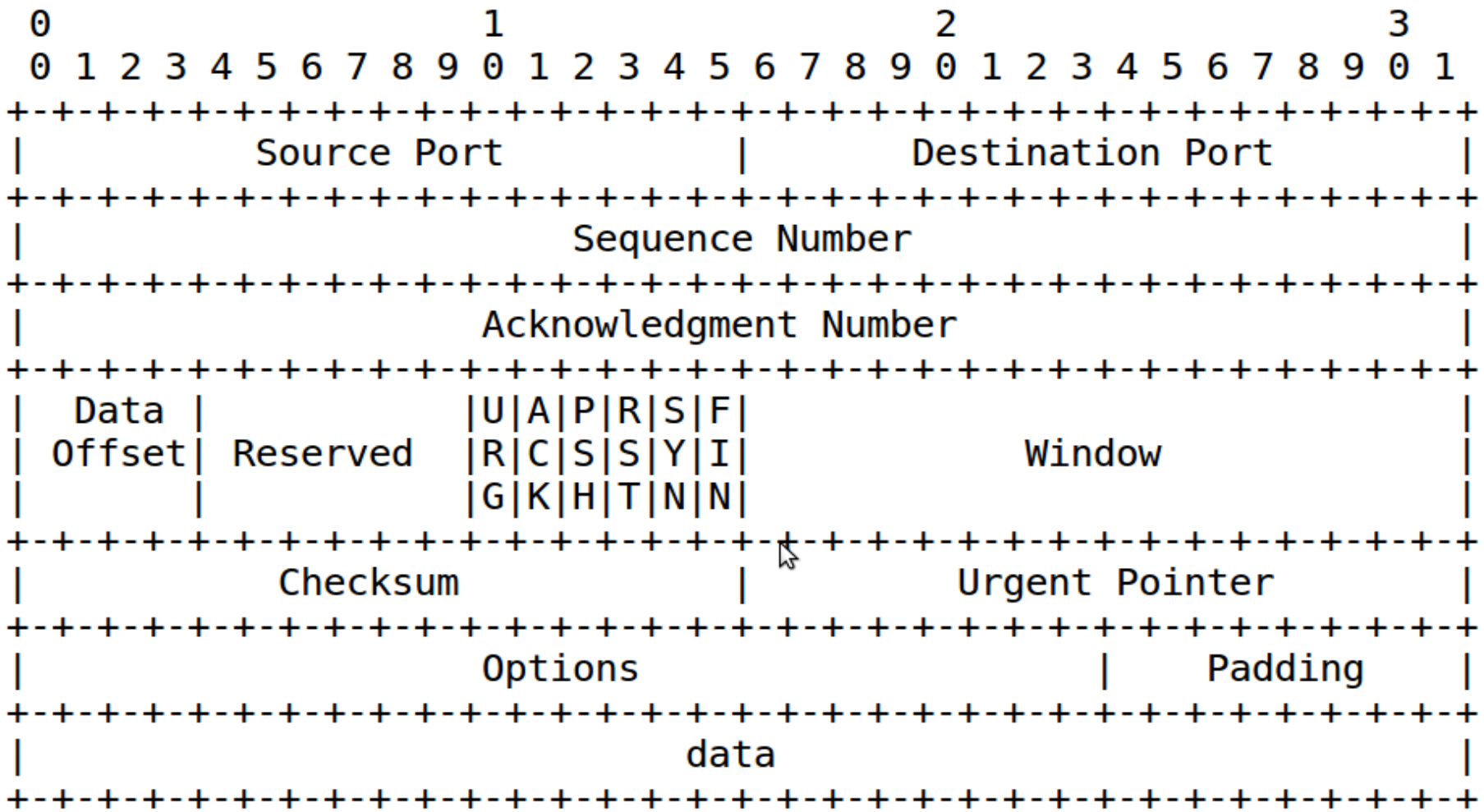
Protocol Encoding 1/2

- Serialization (marshalling) to wire format
- Related terms: PDU, datagram, framing, fragmentation, MTU
- Text encoding (app-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, UDP
  - Example formats: XDR, ASN.1, BER, TLV
- Typically binary formats are visualized in “box notation” for engineers in protocol specifications

Box Notation Example (RFC793)



TCP Header Format

Note that one tick mark represents one bit position.

Security 1/4

- 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 2/4

- 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
  - Replicated resources (e.g. cloud networking)

Security 3/4

- Opportunistic security vs. infrastructure
  - Opportunistic security is used e.g. in SSH
  - 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 4/4

- Do not hard-code cryptographic algorithms into the protocol!
  - Algorithms are safe only until a flaw is found
  - For example, MD5 is depracated
  - Also key lengths deprecate due to Moore's laws
- Better to embed in the design from day one
    - Security difficult to add after deployment
    - Privacy even more difficult to add afterwards
- The overall strength of the system is as strong as its weakest link
- Reuse existing protocols (e.g. TLS), do not invent new!

Protocol Correctness

- Verify that the protocol works
  - Peer review
  - Implement your own specification
    - Implementation insight
    - Performance analysis
  - Mathematical (crypto) analysis
  - Scalability analysis with simulators
- Ready for deployment?
  - More difficult to fix already deployed software
  - Future compatibility

Deployment Obstacles

- Firewalls
  - Firewalls drop packets with IPv4 options
  - TCP options are much better
  - End-host firewalls (virus scanners, Selinux)
- Network Address Translators (NATs)
  - Engineering of end-to-end (or P2P) protocols difficult
  - By default, NATs block new incoming connections
  - Overlapping namespaces (error prone)
  - Easier naming with split-horizon DNS
  - Unreliable penetration with uPnP, 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 wants to avoid vendor lock-in?
  - Security
  - Drawback: standardization takes time
- Few standards organizations
  - W3C: Web standardization
  - IETF: Applications, routing, transport, IPv4/IPv6, security
  - IEEE: Data-link layer (ethernet, wlan), POSIX, ..
  - ITU-T, ETSI, 3GPP: Cellular technology

Related Literature 1/2

- Theory
  - Principles of Protocol Design, Sharp, 1995
  - Reasoning about Knowledge, Fagin et al, 1995
- Protocol Engineering
  - RFC3117: On the Design of Application Protocols, Rose, 2001
  - RFC6250: Evolution of the IP Model, Thaler, 2011
  - RESTful Web Services, Richardson et al, 2007, O'Reilly

Related Literature 2/2

- Service Adoption and Deployment
  - Network Services: Investment Guide, Gaynor, 2003
  - RFC5218: What Makes for a Successful Protocol, Thaler et al, 2008
- Security:
  - RFC3631: Security Mechanisms for the Internet, Bellovin et al, 2003