Designing a Protocol

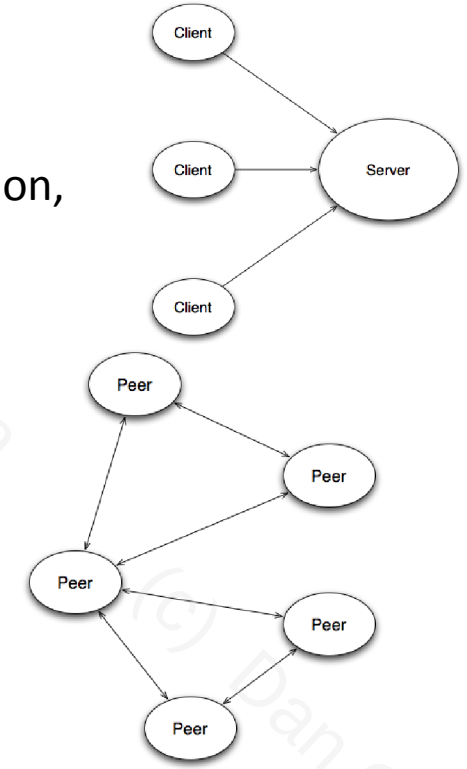
1. Choose the patterns of communication and data transmission
2. Establish the design goals
3. Choose the message format “philosophy”
4. Design the message structure: format, fields, types of messages, etc.
5. Design the communication rules (sequences)

Steps 4 and 5 go together

Patterns of communication

- Client - server
  one party initiates the communication, the other responds

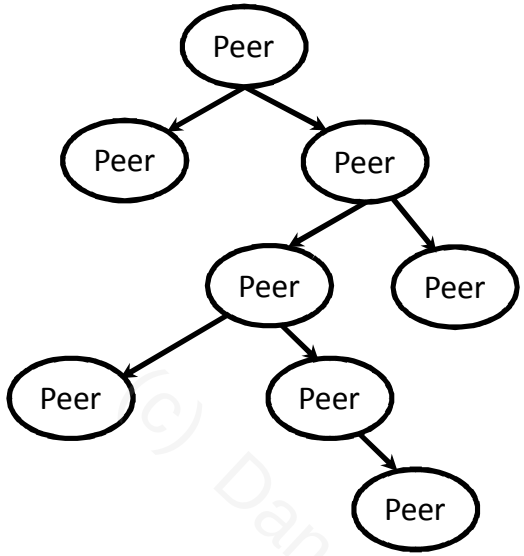
- Peer-to-peer
  any party may initiate the communication



See Robin Sharp, *Principles of protocol design*, Springer, 2008

Patterns of communication

- Hierarchical communication many parties, organized in a hierarchy, and communicate only via the branches of the tree



See Robin Sharp, *Principles of protocol design*, Springer, 2008

Patterns of transmissions

- One-to-one
  only two parties involved in communication at a time
- Multicast
  one or more parties may transmit data to multiple parties at a time
- Broadcast
  one or more parties may transmit data to all parties at the same time

Each or all of these patterns may be used at different stages in the communication

Design Goals

- Define the framework for communication
  - Should the communication be fast?
  - Do we need reliable exchanges? (E.g. confirmations and such)
  - How important is the authentication of parties?
  - Is the transferred data confidential? What degree of authorization is needed?
  - How many types of parties are involved? Can they all communicate to each other?
  - Are there bandwidth or connection availability limitations?
  - Do we need to maintain communication channels? Are connectionless models more suitable, instead?
  - Do we need complex error handling?
  - ...

Design Goals

- A communication protocol should be:
  - simple
    ~ don't make easy tasks hard to do
    ~ don't provide two ways for doing the same thing
  - scalable
    ~ estimate the number of clients per server (or peers communicating)
    ~ design the protocol so that it balances the responsibilities (e.g. shifts the communication balance to the clients, to free the servers which are already full of responsibilities)
  - efficient
    ~ minimize the command overhead
    ~ minimize the data traffic
  - extensible
    ~ make room for further extensions
    ~ don't overdo it, though

Message formats

- Two approaches:

  - Text-oriented protocols
  - Protocols using binary messages

Text-Oriented

- All messages are readable character strings
- Advantages
  - human readable, easy to understand and monitor
  - flexible, easy to extend (if properly designed)
  - easy to test, even with ‘‘standard’’ clients (telnet?)
- Disadvantages
  - human readable, easy to read by unauthorized persons (without encryption)
  - may become complex, harder to parse in code
  - may make the messages unjustifiably large

Binary messages

- Messages are blocks of structured binary data
- Advantages
  - Better ways of structuring the data
  - Suitable for large or complex data transfers
  - Messages are as small as possible
- Disadvantages
  - Hard to read, debug or test
  - Need to consider the data representation conventions on hosts and network (e.g. the “endianness”: little-endian vs. big-endian)

Designing the Message

- A very important aspect in protocol design
- Influences all the characteristics of the communication: scalability, efficiency, simplicity, extensibility
- The design involves two aspects:
  a) types of messages
  b) message structure

Types of Messages

- One message type for each distinct aspect of the communication
- Three categories of messages:
  - commands
  - data transfer
  - control

  Each category may include several message types

Command Messages

- Define the stages of the dialogue between the parties
- Address various communication aspects:
  - communication initiation or ending
  - describe the communication stage (e.g. authentication, status request, data transfer)
  - status changes (e.g. requests for switching to the data transfer mode)
  - resource changes (e.g. requests for new communication channels)
  - ...

Data transfer

- Messages that carry data over the network
- They are usually sent as a responses to specific commands
- Data is usually fragmented in multiple messages
- Besides the actual data, may describe:
  - the type of the binary data format
  - clues for the layout of the structured data (when the structure is flexible/dynamic)
  - data size, offset or sequence information
  - type of the data block: last / intermediary

Control Messages

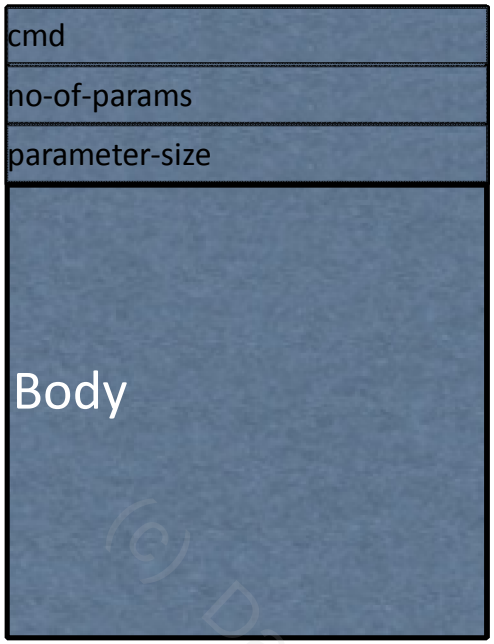
- Control the dialogue between the parties
- Address various communication aspects:
  - coordination (e.g. receipt confirmation, retry requests)
  - cancellation or interruption
  - availability checks
  - ...

Message Structure

- Header: contains structured fields describing the actual data in the message:
  - message type
  - command
  - body size
  - recipient information
  - sequence information
  - retransmission count
  - etc.
- Body: the actual data to be transmitted:
  - the command parameters
  - the data payload

Message Structure

- The header structure must be well-known by the receiving party
- The header may contain clues helping the recipient to understand the rest of the message and the details regarding the data in the body (e.g. size, data format or encryption, etc.)
- Headers usually have a fixed size, while the body size may be variable (within limits)

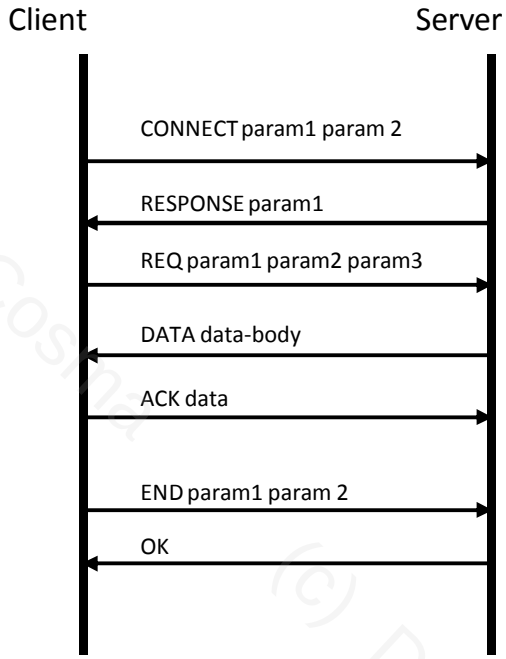


Communication Rules

- Along with the messages, this is the other essential part of the protocol
- Describe the sequences of commands, data and control messages, at each and all the stages in the communication, for all parties in the system
- Should be clearly and thoroughly specified, through detailed descriptions of each communication scenario (for each possible case of peer interaction)

Communication Rules

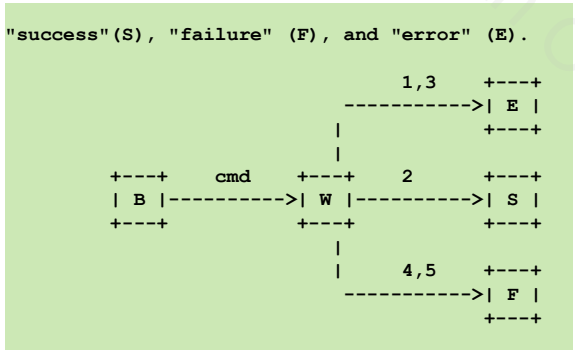
Sequence diagrams are more than useful...

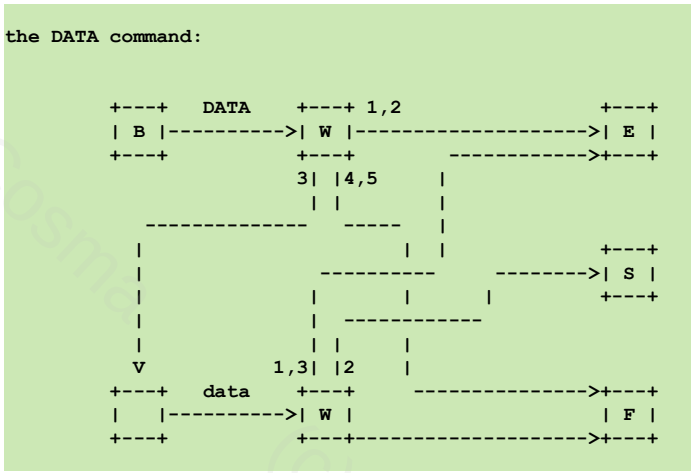


An imaginary diagram for a non-existent protocol

Communication Rules

...but *state diagrams*
are almost mandatory





State diagrams from the SMTP specification (1982)
[source: RFC 821]

Documenting the Design

- The protocol specification must be available for all interested parties, as a specific document
- The specification must be
  - clear, easy to understand
  - comprehensive (complete)
  - non-ambiguous
  - maintainable (for versioning and such)

  By only having the specification, parties must be able to thoroughly implement the software components involved in communication

Specification Content

- Introduction
  - purpose of the protocol, domain, environment, prerequisites
- The communication model
  - parties involved, relations, roles, general description of the dialogue flow between components, etc.
- Communication steps or procedures
  - description of each stage, procedure or aspect of communication
- Message description
  - syntax and semantics for all types of messages (commands, headers, codes, etc.)
- Sequence of commands and replies
  - the detailed description of the communication rules, including state diagrams, sequence diagrams, and comprehensive explanations for the procedures