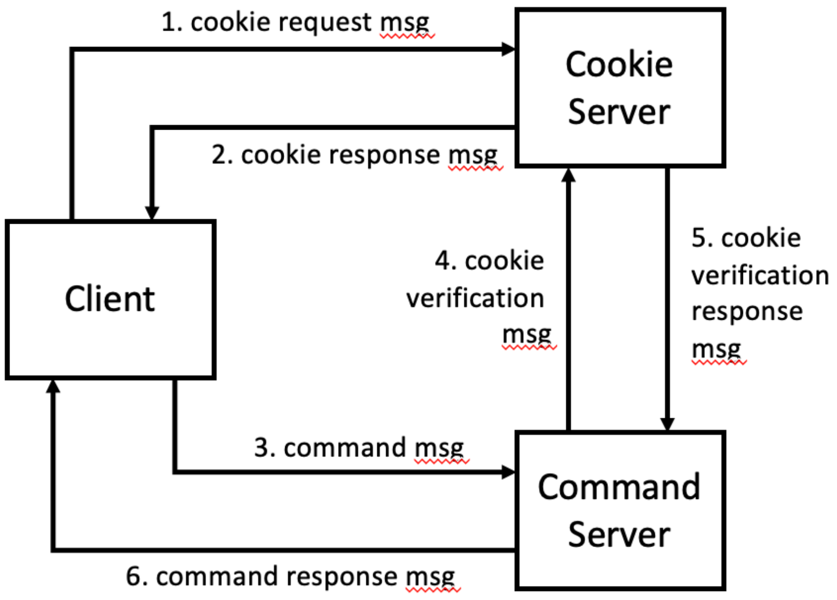
Internetworking – Hands-on

Prof. Dr. Marcus Schöller

Access Protocol Specification

1. Overview

This document specifies version 1.1 of the Command Protocol (CP). CP emulates an application layer protocol by utilizing the PHY protocol.



Before sending any commands to the server, a client has to get itself a cookie. The cookie can be requested from a cookie server. The cookie is embedded in subsequent command messages sent to the command server.
A command server first has to validate the cookie of the incoming message. To do so, the command server communicates with the cookie server. Only commands with a valid cookie are processed by the command server.
Message 3 – 6 shall be protected from message corruption. The sender of such a message calculates a checksum and appends it to the message. The receiver verifies the checksum and only accepts message that are not corrupted. CP shall use CRC (cyclic redundancy check) as its checksum function.

2. Protocol behavior

2.1. Requesting cookies

A client requests a cookie from the cookie server by sending a cookie request message. The client waits for a cookie response message to be returned. When no response is received within two seconds the request is considered lost and should be resent to the server. If three consecutive cookie request messages time out, an exception has to be returned to the client app.

There shall be at most one active cookie for any client at any point in time. A client is uniquely identified by the clients IP address and UDP port number. If a client requests a cookie any previously issued cookies shall be invalidated. A server can limit the total amount of active cookies. In case more cookies are requested, the server will indicate a failure in the response message.

The cookie lifetime shall be 60 seconds. The server has to invalidate any cookie that has exceeded its lifetime and shall not accept any commands for this cookie anymore. The client shall request a new cookie before issuing any new commands.

The cookie response shall indicate whether the server has accepted the cookie request message or occurrence of a failure. If the server has accepted the request, it shall return a cookie within the cookie response message to the client. The client has to provide this cookie in all future command messages. If the response indicates a failure, an exception shall be returned to the client app. The client app shall not be able to send any commands to the command sever in case of a failure. The client can request a cookie again after a five seconds timeout.

2.2. Issuing commands

Once the client has an active cookie, it can send arbitrary many command messages to the command server as long as the cookie is valid. Any command message shall contain the cookie. The command sends a response indicating success or failure of processing the command message.

2.3. Cookie Verification

A command server needs to verify that the cookie presented in a command message is valid and belongs to the client sending the command. This is achieved by sending a cookie verification message to the cookie server. The cookie server responds with a cookie verification response message indicating the result of the verification process.

3. Message Format

All messages are encoded as ASCII text strings. Messages consist of a number of *fields* separated by whitespaces.

The following notational conventions are used in describing the messages:

- String literals are rendered in typewriter text.
- Whitespace is denoted by ⟨WS⟩.
- Fields of a message other than literals are indicated by ⟨*fieldname*⟩.
- The notation ⟨*msg*⟩ denotes a string of ASCII printable characters **including** whitespaces.
- `[⟨Fields⟩]` in square brackets are optional.

3.1. Cookie Request Message

A client requests a cookie by sending a cookie request message to the server. The **cookie request** message has the following format:

```
cp⟨WS⟩cookie_request
```

3.2. Cookie Response Message

The **cookie response** message has one of the two following formats:

1. If the server accepts the cookie request, it shall respond with a cookie response message:
   `cp⟨WS⟩cookie_response⟨WS⟩ACK⟨WS⟩⟨cookie⟩`
   On receipt of this message, the client is allowed to send command messages to the server.
2. If the server does not accept the cookie request, it shall respond with a cookie response message indicating failure. The server should indicate the type of failure using suitable error codes.
   `cp⟨WS⟩cookie_response⟨WS⟩NAK[⟨WS⟩⟨error⟩]`
   On receipt of this message, the client can restart the cookie request process by sending another cookie request message or abort operation. Requesting a cookie after receipt of a failure message shall be delayed by five seconds. The client shall not send any command message without receiving a cookie first.

3.3. Command Messages

The **command** message has the following format:

```
cp⟨WS⟩command⟨WS⟩⟨id⟩⟨WS⟩⟨cookie⟩⟨WS⟩⟨length⟩⟨WS⟩⟨command⟩
          [⟨WS⟩⟨message⟩]⟨WS⟩⟨checksum⟩
```

The command message is sent from a client to the server from which the client previously requested a cookie. The fields in the message are defined as follows:

- **id**: Every command is uniquely identified by a command id. The client can start with an arbitrary non-negative id. The id of any message needs to be greater than the id of the previous message. The maximum id is 65535.
- **cookie**: The cookie is the cookie value received from the server.
- **command**: The command can be either “status” or “print”.
- **length**: The length of the command field plus the optional message field if present.
- **message**: The message depends on the command the client sends. The message field is omitted for status commands. In case of a print command, the text to print to the server screen is contained in the message field.
- **checksum**: The checksum is calculated over all message fields except the `cp` field.

3.4. Command response message

The **command reponse** message has the following format:

```
cp⟨WS⟩command_response⟨WS⟩⟨id⟩⟨WS⟩⟨success⟩⟨WS⟩⟨length⟩
          [⟨WS⟩⟨message⟩]⟨WS⟩⟨cecksum⟩
```

The command response message is sent from the server to the client in response to an incoming command message. The fields in the message are defined as follows:

- **id**: The server uses the same id the client used in the command message to match response message to command message. The maximum id is 65535.
- **success**: This can be either “ok” or “error”.
- **length**: The length of the message field. The length is 0 if no message is included.
- **message**: The message depends on the which command the client has issued. For a print command the message field is omitted. For a status command the message field contains at least two lines in JSON formatting for the number of successfully processed command messages and the time-to-live value of the current cookie.
- **checksum**: The checksum is calculated over all message fields except the `cp` field.

3.5. Cookie verification message

The **cookie verification request** message has the following format:
`cp⟨WS⟩cookie_verification_request⟨WS⟩⟨cookie⟩⟨WS⟩⟨client⟩⟨WS⟩⟨cecksum⟩`

- **cookie**: The cookie value to be verified.
- **client**: The client IP address and UDP port number in JSON format.
  Example: ip: 192.168.100.50 udp: 6000
- **checksum**: The checksum is calculated over all message fields except the `cp` field.

3.6. Cookie verification response message

The **cookie verification response** message has one of the two following formats:

1. If the cookie is valid:
   `cp⟨WS⟩coookie_verification_response⟨WS⟩ok⟨WS⟩⟨cookie⟩⟨WS⟩⟨cecksum⟩`
   - **checksum**: The checksum is calculated over all message fields except the `cp` field.

2. If the cookie is invalid:
   `cp⟨WS⟩coookie_verification_response⟨WS⟩error⟨WS⟩⟨cookie⟩⟨WS⟩⟨length⟩⟨WS⟩⟨message⟩⟨WS⟩⟨cecksum⟩`
   - **length**: The length of the message.
   - **checksum**: The checksum is calculated over all message fields except the `cp` field.