# Writing a Protocol Specification

RES, Lecture 4

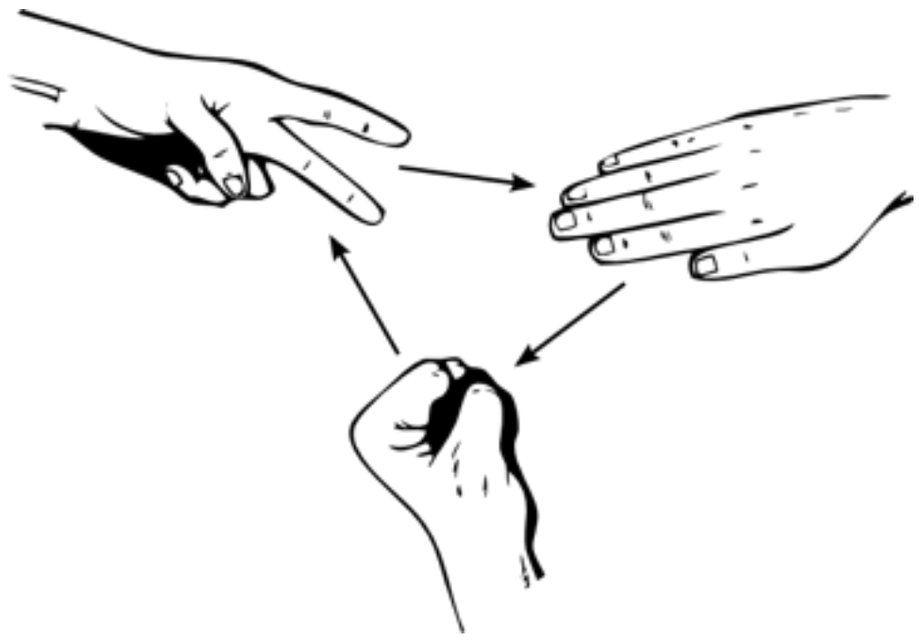
Olivier Liechti

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

# What is an Application-Level Protocol?

*An application-level protocol is* ***a set of rules****, which define how the* ***components*** *of a* ***distributed system interact*** *with each other so that a* ***service*** *can be fulfilled for the benefit of one or more* ***users****.*



# Questions to Address in a Protocol...

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- What are the different **components** in the distributed system.
- What are their **roles** and what **functions** do they provide?
- How do components **find each other**, before they can start an interaction?
  - Do the clients need to know the **address** of available of servers (through out-of-band agreement)
  - or does the protocol support **dynamic discovery** of peers?
- Once they have found each other, how do components **start to interact** with each other?
  - Are some components listening for connection requests?
  - Are some components initiating connection requests?

# Questions to Address in a Protocol...

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- What is the **format**, in other words what is the **syntax**, of the different types of **messages** exchanged by the components?
- What are the **actions** that should be **triggered** after the reception of different types of messages? In other words, what is the **semantics** of the messages?
- Is the interaction between components **stateful** or **stateless**?
  - Do the messages exchanged between two peers belong to some kind of **connection** or **session**, with an associated state?
  - Or, in the contrary, are the messages exchanged by two components **independent** from each other?

# Questions to Address in a Protocol...

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- For **stateful protocols**, what are the **different states** in which a session can be? What are the possible transitions between these states?
  - What types of messages can be exchanged in these different states?
  - Is there some **data** associated with each of the states?
- How **secure** are the interactions between the different components in the system? Are there things that the components need to do in order to ensure some level of security?

*And more questions, depending on your context...*

# How Should I Write a Specification for My Own Protocol???

HTTP 1.0
*stateless, syntax*

POP3
*simple, stateful*

SMTP

FTP
*multiple channels*

IRC
*several RFCs*

IMAP

TFTP
*reliable UDP*

SIP
*component types*

# Techniques & Tools

# Textual Descriptions

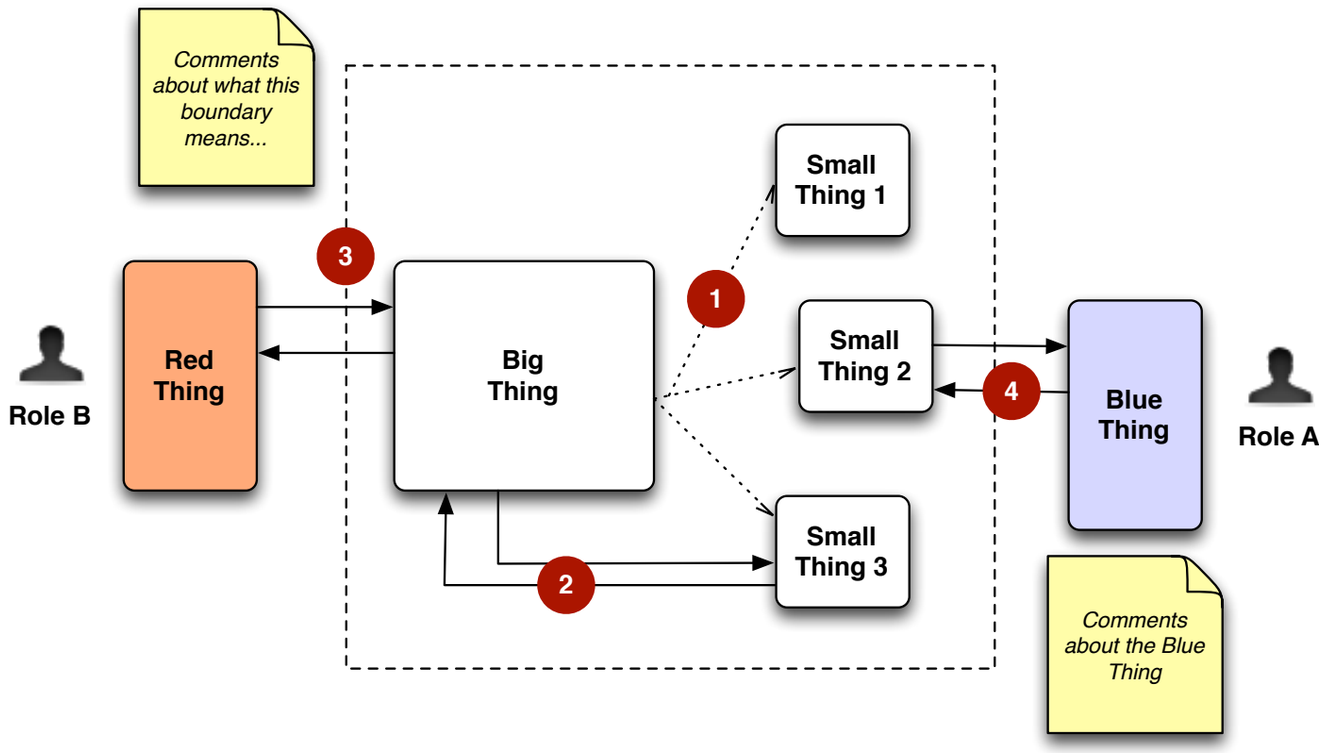
heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

---

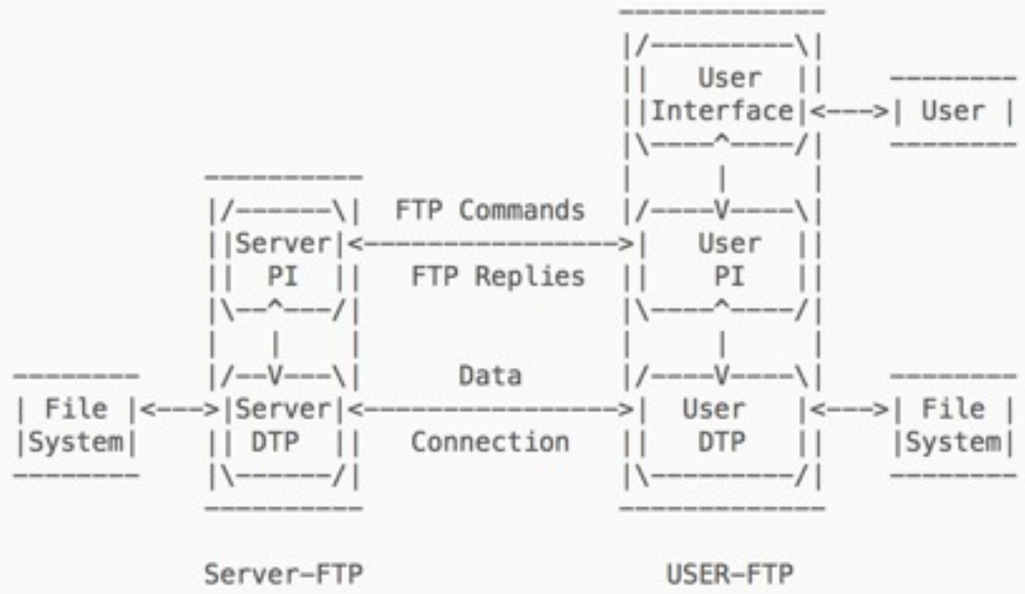
```
Network Working Group                                          S. Christey
Request for Comments: 2795                           MonkeySeeDoo, Inc.
Category: Informational                                      1 April 2000



The Infinite Monkey Protocol Suite (IMPS)


Status of this Memo

   This memo provides information for the Internet community.  It does
   not specify an Internet standard of any kind.  Distribution of this
   memo is unlimited.

Copyright Notice

   Copyright (C) The Internet Society (2000).  All Rights Reserved.

Abstract

   This memo describes a protocol suite which supports an infinite
   number of monkeys that sit at an infinite number of typewriters in
   order to determine when they have either produced the entire works of
   William Shakespeare or a good television show.  The suite includes
   communications and control protocols for monkeys and the
   organizations that interact with them.
```

# Component Diagrams

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud



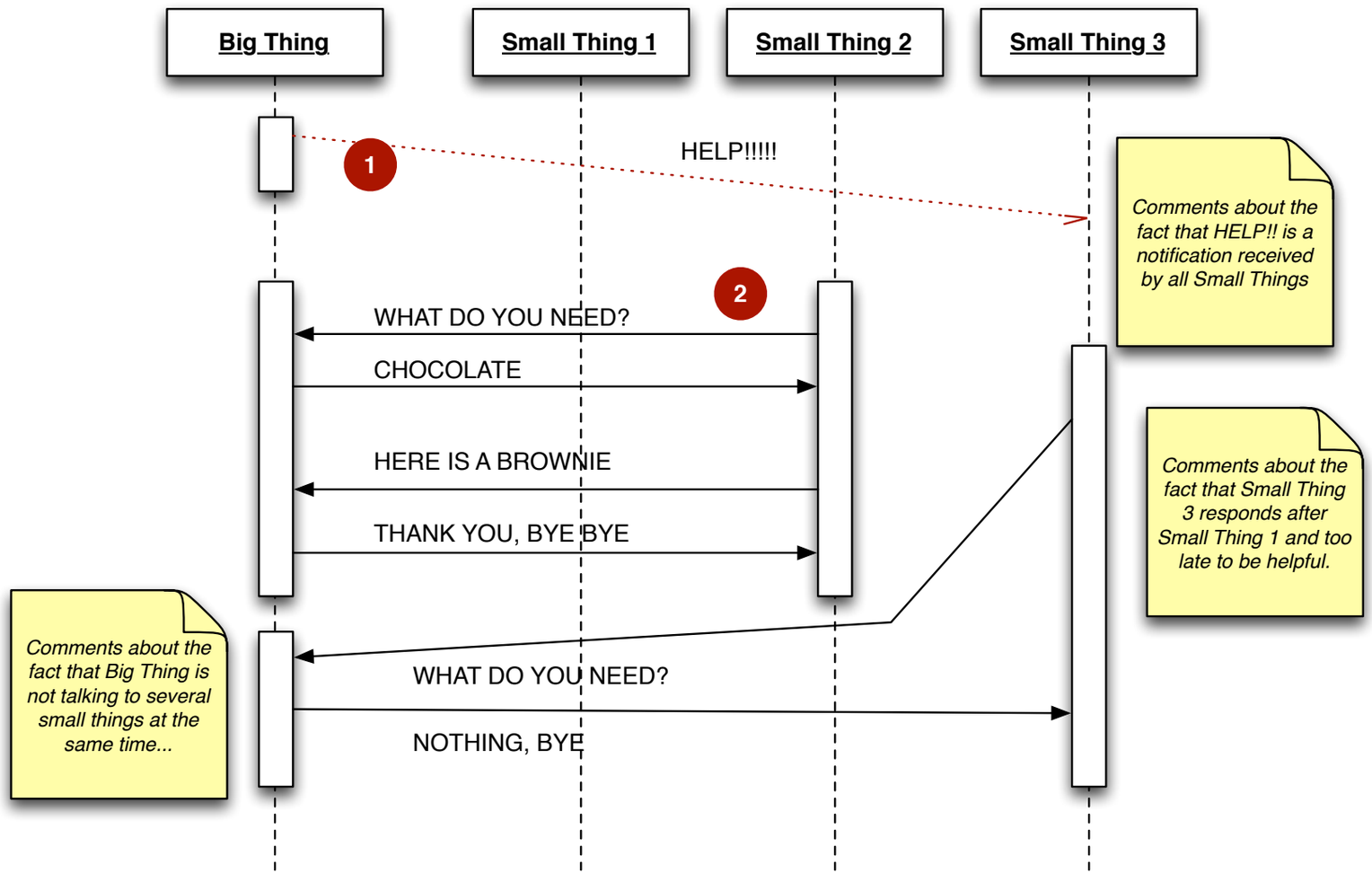

With the above definitions in mind, the following model (shown in Figure 1) may be diagrammed for an FTP service.




NOTES: 1. The data connection may be used in either direction.
2. The data connection need not exist all of the time.

# Sequence Diagrams

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud



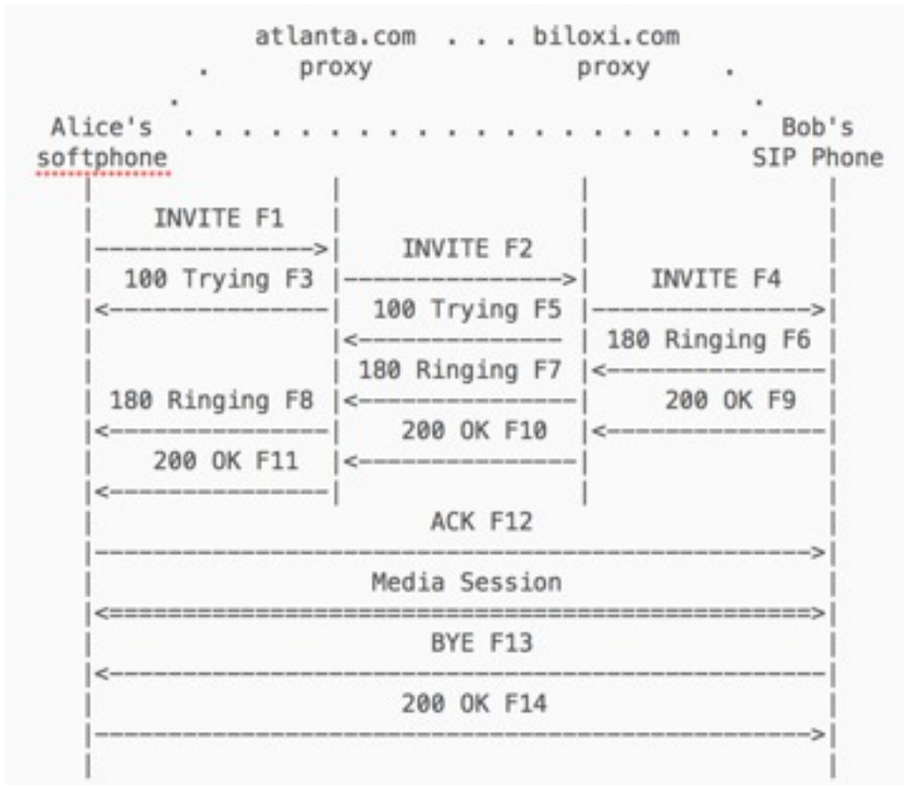
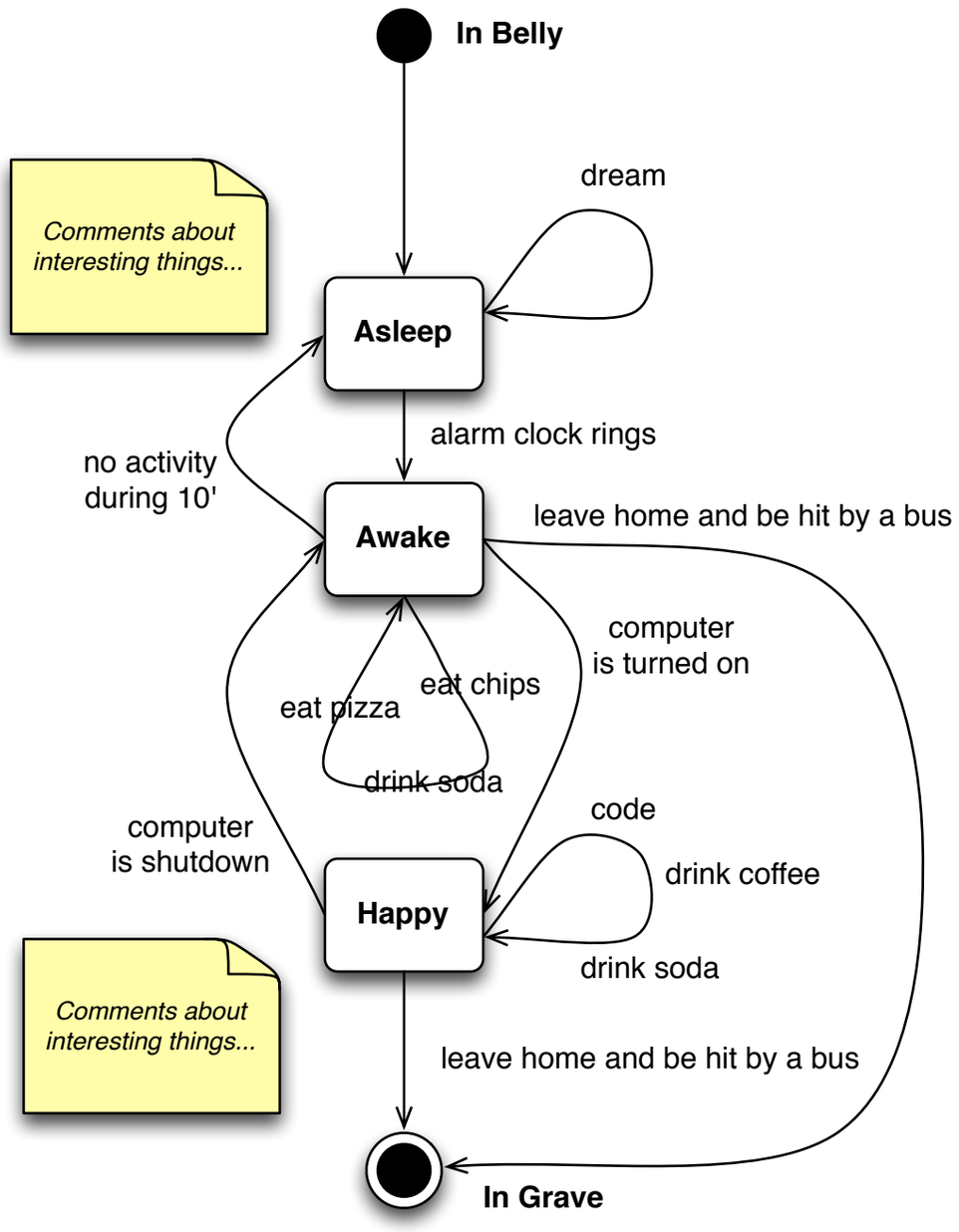



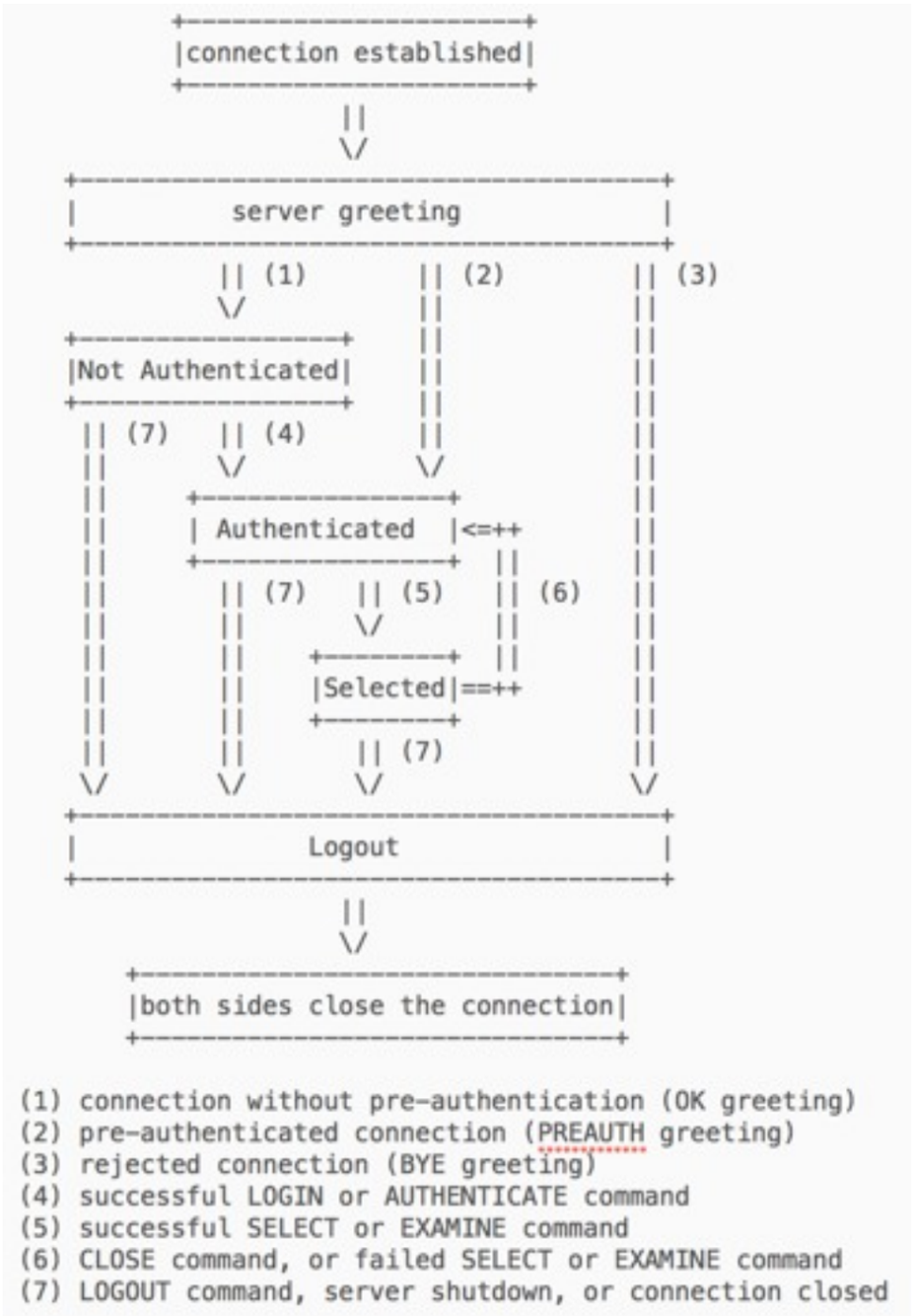

Figure 1: SIP session setup example with SIP trapezoid

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```

# State Machine Diagrams

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud



# Grammars (1)

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Network Working Group
Request for Comments: 5234
STD: 68
Obsoletes: 4234
Category: Standards Track

D. Crocker, Ed.
Brandenburg InternetWorking
P. Overell
THUS plc.
January 2008

Augmented BNF for Syntax Specifications: ABNF

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

Internet technical specifications often need to define a formal syntax. Over the years, a modified version of Backus-Naur Form (BNF), called Augmented BNF (ABNF), has been popular among many Internet specifications. The current specification documents ABNF. It balances compactness and simplicity with reasonable representational power. The differences between standard BNF and ABNF involve naming rules, repetition, alternatives, order-independence, and value ranges. This specification also supplies additional rule definitions and encoding for a core lexical analyzer of the type common to several Internet specifications.

Table of Contents

# Grammars (2)

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
4. HTTP Message

4.1 Message Types

   HTTP messages consist of requests from client to server and responses
   from server to client.

      HTTP-message   = Simple-Request           ; HTTP/0.9 messages
                     | Simple-Response
                     | Full-Request            ; HTTP/1.0 messages
                     | Full-Response

   Full-Request and Full-Response use the generic message format of RFC
   822 [7] for transferring entities. Both messages may include optional
   header fields (also known as "headers") and an entity body. The
   entity body is separated from the headers by a null line (i.e., a
   line with nothing preceding the CRLF).

      Full-Request   = Request-Line             ; Section 5.1
                       *( General-Header        ; Section 4.3
                        | Request-Header        ; Section 5.2
                        | Entity-Header )       ; Section 7.1
                       CRLF
                       [ Entity-Body ]          ; Section 7.2

      Full-Response  = Status-Line              ; Section 6.1
                       *( General-Header        ; Section 4.3
                        | Response-Header       ; Section 6.2
```

```
5.1  Request-Line

   The Request-Line begins with a method token, followed by the
   Request-URI and the protocol version, and ending with CRLF. The
   elements are separated by SP characters. No CR or LF are allowed
   except in the final CRLF sequence.

      Request-Line = Method SP Request-URI SP HTTP-Version CRLF
```

```
5.1.1 Method

   The Method token indicates the method to be performed on the resource
   identified by the Request-URI. The method is case-sensitive.

      Method         = "GET"                    ; Section 8.1
                     | "HEAD"                   ; Section 8.2
                     | "POST"                   ; Section 8.3
                     | extension-method

      extension-method = token
```

# Scenarios and Examples

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
10. Example POP3 Session

      S: <wait for connection on TCP port 110>
      C: <open connection>
      S:    +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
      C:    APOP mrose c4c9334bac560ecc979e58001b3e22fb
      S:    +OK mrose's maildrop has 2 messages (320 octets)
      C:    STAT
      S:    +OK 2 320
      C:    LIST
      S:    +OK 2 messages (320 octets)
      S:    1 120
      S:    2 200
      S:    .
      C:    RETR 1
      S:    +OK 120 octets
      S:    <the POP3 server sends message 1>
      S:    .
      C:    DELE 1
      S:    +OK message 1 deleted
      C:    RETR 2
      S:    +OK 200 octets
      S:    <the POP3 server sends message 2>
      S:    .
      C:    DELE 2
      S:    +OK message 2 deleted
      C:    QUIT
      S:    +OK dewey POP3 server signing off (maildrop empty)
      C:  <close connection>
      S:  <wait for next connection>
```

# Proposed Template

# Template Overview

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

1. Introduction
2. Terminology
3. Protocol Overview
    1. System Architecture
    2. System Components
    3. Interactions between Components
4. Protocol Details
    1. Transport Protocols and Connections
    2. State Management
    3. Message Types, Syntax and Semantics
    4. Miscellaneous Considerations
    5. Security Considerations
5. Examples
6. References

# The Lab

# Schedule & Evaluation

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- **Today**
  - **Individually**
    - Read and study the “story” and sketch out the system architecture (10’)
    - Write a list of questions that need to be treated in the protocol (10’)
  - **In groups of 4 students**
    - Share your ideas, do a synthesis and document it in a few slides (20’)
    - Decide how you would split the specification work in two sub-groups (5’)

# Schedule & Evaluation

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- **Before Thursday (work in pairs of students)**
  - Read the lecture material
  - **Fork** the https://github.com/wasadigi/Smart-Calculator repository. **Clone** your fork on your local machine.
  - Go through the POP3, HTTP, SMTP and IRC specifications
- **On Thursday**
  - **Design** the protocol by addressing the identified questions
  - **Document** the design in a specification document (you can use the template)
  - Make sure that you **commit and push** your work to your Github fork.

# Schedule & Evaluation

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- **On Monday, March 31st**
  - Starting early morning, I will look at the content of your forks on Github
  - I will select between 2 and 4 groups, who will present their work during the lecture
  - The best group will get an extra bonus grade