IK2213
Network Services and Internet-based Applications

Assignment 1: WebMail

Raúl Jiménez <rauljc@kth.se>

General info

- This is not a course focused on programming
- Everything is on the course web
- Two-people teams
- Development server
  - `ssh username@studdev.ssvl.kth.se`

General info

- Deadlines and make-up
- Grading process
- Grades table
- Forum

Schedule

- Today
  - Introduction
- 4 Apr 13.00 - 15.00
  - Supervision
- 10 Apr 8.00 AM
  - Submission
- 2 weeks after feedback
  - Make-up

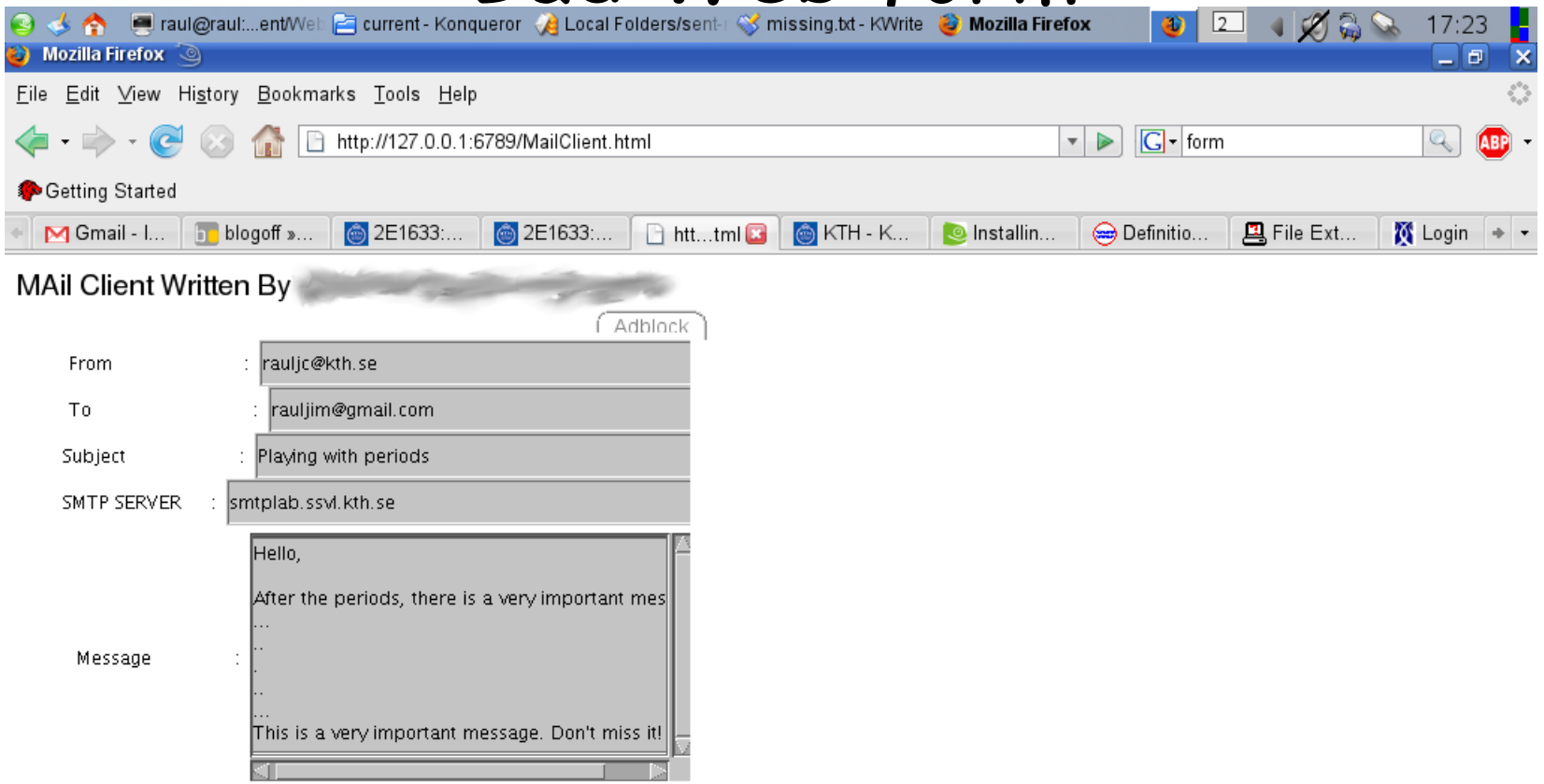
WebMail

- Web service for sending e-mails
- Small catch
  - This is intended for a small-scale system
    - Can't afford to use a large, existing web server, such as Apache
  - You will write the web server yourself
- You will learn about
  - Socket programming, parsing, HTTP, SMTP, DNS, MIME, ...

Web Mail Server

- Web form for sending mail
- Could look something like this:

Bad Web form


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://127.0.0.1:6789/MailClient.html
form
Getting Started
MAil Client Written By
Adblock
From : rauljc@kth.se
To : rauljim@gmail.com
Subject : Playing with periods
SMTP SERVER : smtplab.ssvl.kth.se
Message :
Hello,
After the periods, there is a very important mes
...
..
.
..
...
This is a very important message. Don't miss it!
Applet MailClient started

WebMail

- You will write:
  - HTTP server (web)
  - SMTP client (mail)
  - Application which use the two of them

Layers

- Application USERS
- Library DEVELOPERS
- Protocol YOU

WEBmail

Application

Library

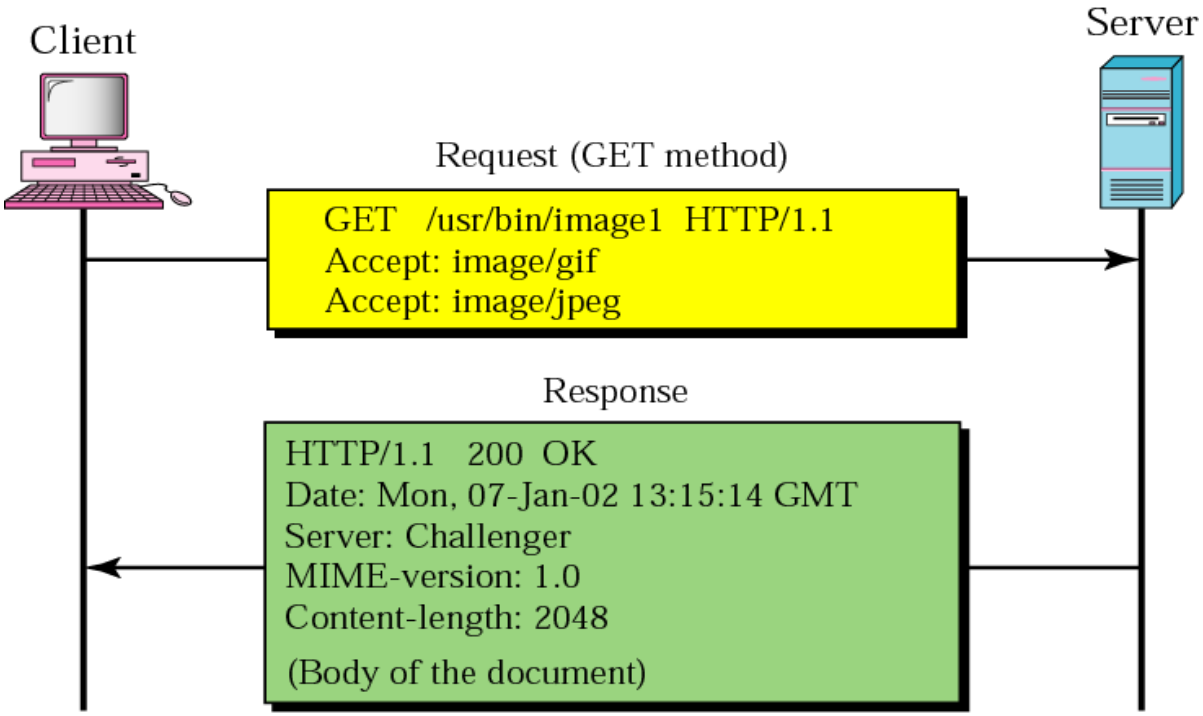
```
#! /usr/bin/env python
import string
import sys
from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer

class MyHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        f = open('/home/raul/public_html/index.html')
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()
        self.wfile.write(f.read())
        f.close()
        return

server = HTTPServer(('',int(sys.argv[1])), MyHandler)
server.serve_forever()
```

Protocol

Wireshark is your friend!

Built-in Web Server

- One way to do this would be to use an existing Web server and dynamic pages
  - Apache + PHP, for example
- But we want this to be a small and simple application
- You should implement the Web server functionality yourself
- Does not need to be complete
  - sufficient functionality for WebMail, plus error handling



HTTP Server Socket

- TCP receive socket
  - Configurable port number
- HTTP events
  - Process an incoming request for the HTTP server
    - Reply with a mail form
  - Process a filled in mail form
    - Send mail to SMTP server
    - Report status to the user

Think Through the Scenarios

- What happens if it is a completely different GET request?
- Do you support GET and/or POST methods?
  - Not a requirement to do both
  - Why did you choose one of them?
- Are the fields filled in correctly?
- Do not make assumptions about what the user does
  - He/she may decide not to send the email after all
    - Will you tie up resources waiting for the filled in form?
  - Is the SMTP server name valid?
  - …

webMAIL

Application

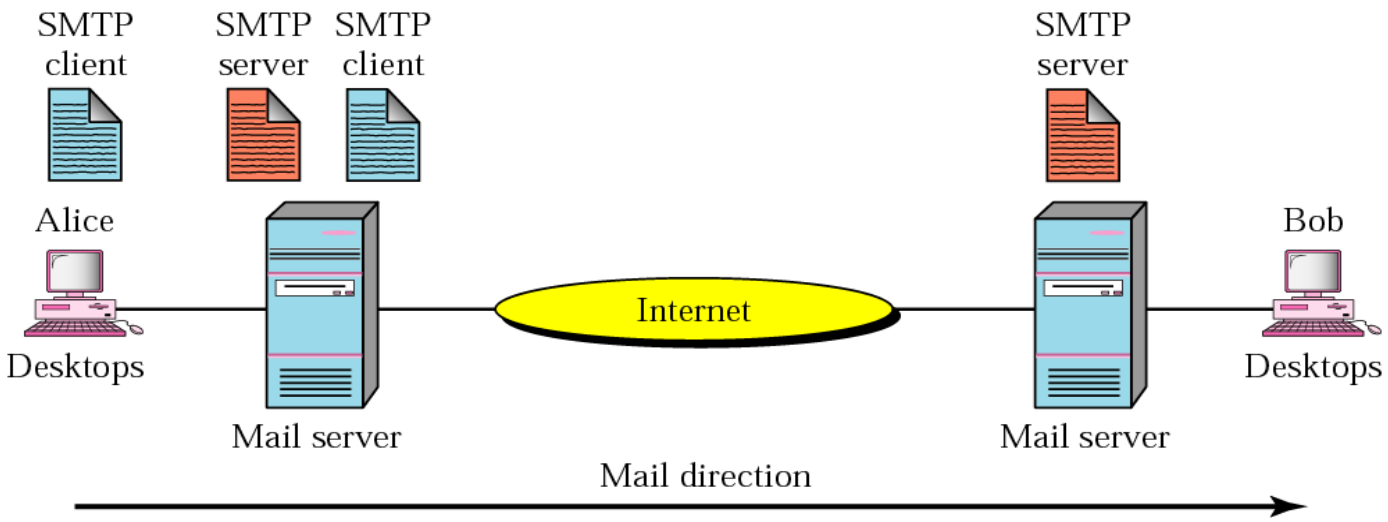
Library

```
def sendMail(to, username, password):
        _from = 'noreply@ik2213.ssvl.kth.se'
        msg = 'Subject: IK2213\n\n' +\
           [...]
            'host: studdev.ssvl.kth.se\n' +\
            'username: ' + username + '\n' +\
            'password: ' + password + '\n'

          mailer = smtplib.SMTP('ik2213.ssvl.kth.se')
          mailer.sendmail(_from, to, msg)
          mailer.quit()
```

Protocol

```
#! /usr/bin/env python

import socket

HOST = 'ik2213.ssvl.kth.se'
PORT = 25

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
s.send('HELO tslab.ssvl.kth.se')
s.send('MAIL FROM: <alice@kth.se>')
s.send('RCPT TO: <rauljc@kth.se>')
s.send('DATA')
s.send('Hello\n.\n')
s.close()
```

SMTP Server IP Address



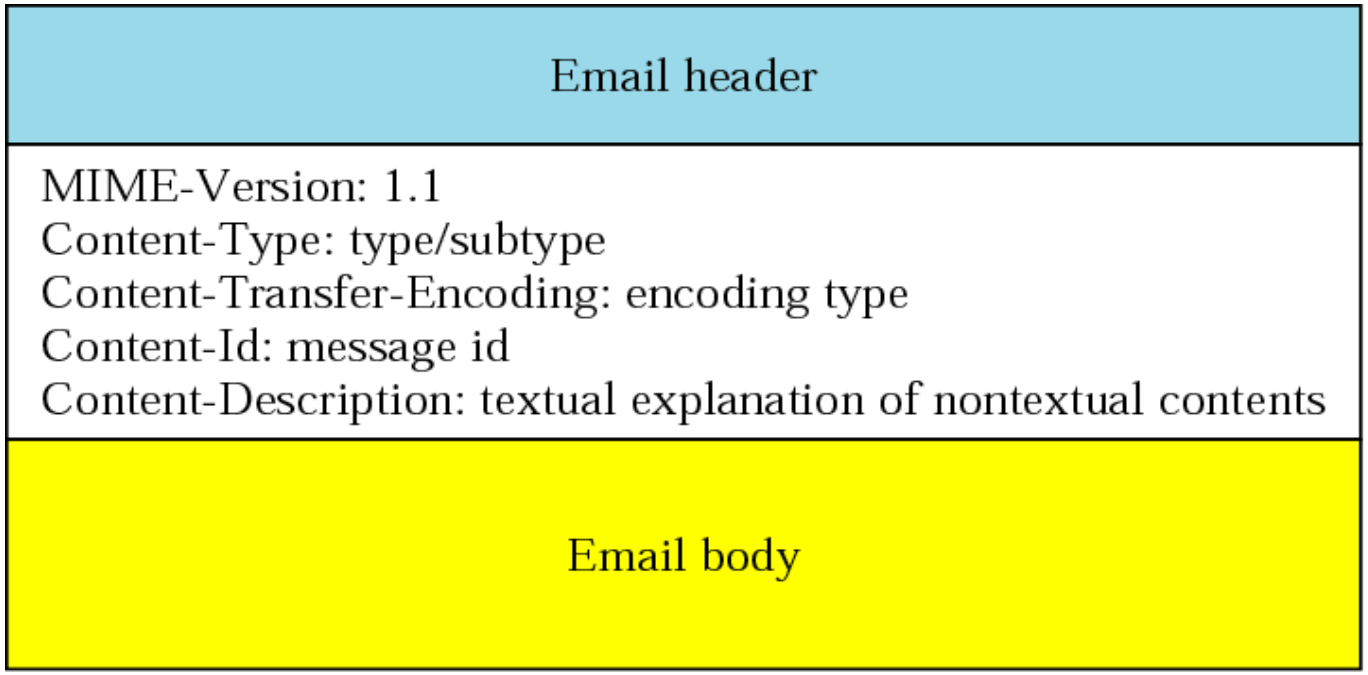
- How to obtain IP address of the mail server?
  - Configure an (outgoing) mail server
    - Grade BASIC
  - Do an "MX" DNS lookup to get the mail server for the recepient's domain (blue.com)
    - Grade MEDIUM and ADVANCED
    - Beware: this can sometimes be tricky to test
      - Some ISPs block outgoing traffic to port 25
      - OK at KTH, though…

SMTP Mail

- Plain text
- Try "telnet *mailserver* 25" to experiment with sending mails
- MIME encoding of non-text data
  - Multipurpose Internet Mail Extension
- For grade MEDIUM, you should add support for Swedish characters in Subject and Body

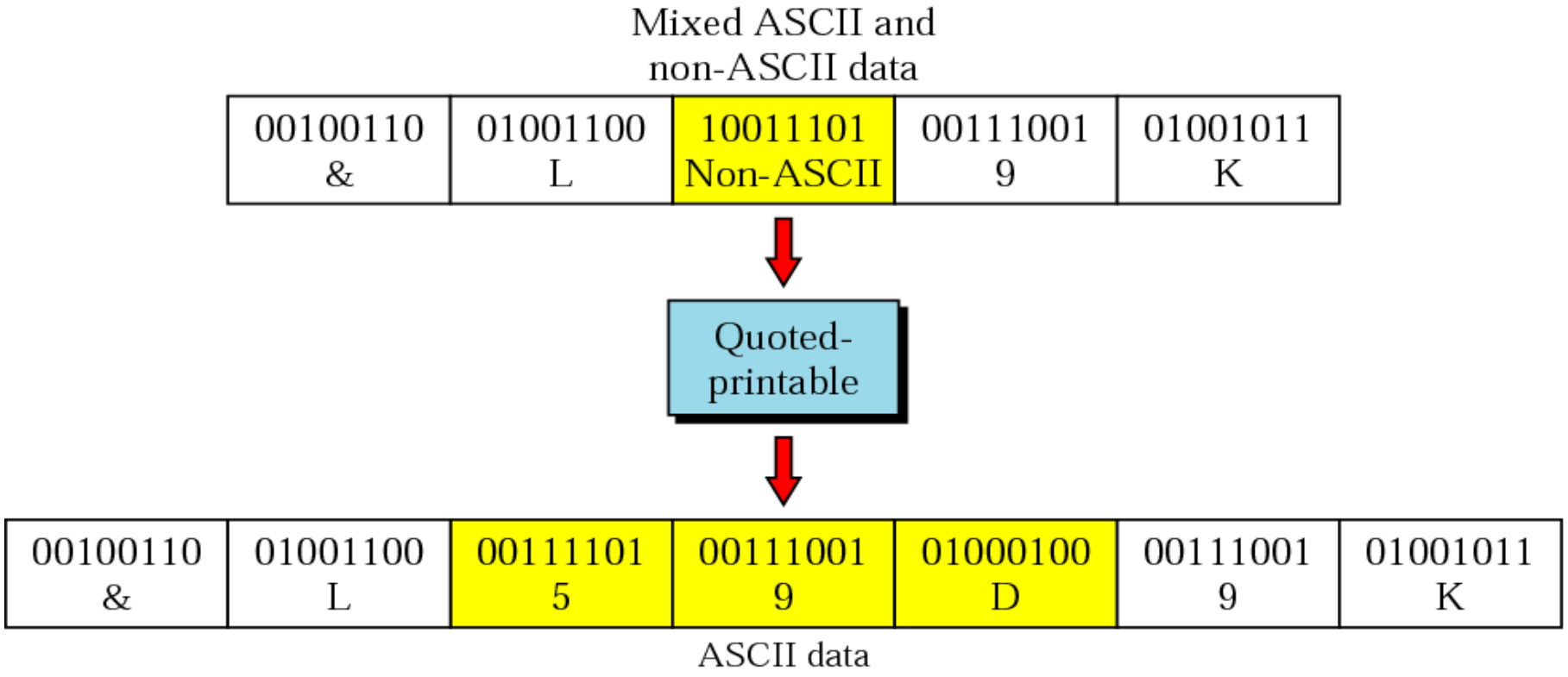
```
HELO host
MAIL FROM: <alice@kth.se>
RCPT TO: <rauljc@kth.se>
DATA
From: Alice
To: Raul
Date: 2004-02-19 08:01
Subject: Important mail

Please read carefully!
.
QUIT
```

MIME



MIME header

Quoted-printable



Future Delivery

- For grade ADVANCED
  - Add a field where the user can specify a (future) time when the mail should be sent
  - When the mail has been sent, send a confirmation message to the sender's email address
    - With error message, if sending failed
  - Add a "status" Web page
    - For administrative purpose
    - Display status of pending mails
      - Sender and receiver addresses
      - Subject
      - Time when mail was submitted to WebMail
      - Time when mail will be sent

Requirements Summary

- Grade BASIC: Basic WebMail functionality
  - Outgoing SMTP server specified by the user
- Grade MEDIUM: MIME and MX lookup
  - Support for character encoding of Swedish characters in subject and message body
  - Look up outgoing SMTP server from email address
- Grade ADVANCED: Future delivery
  - User specified sending time
  - Notification to sender
  - Status page

Simplifications

- **Formatting of From: and To: fields**
  - One email address only
  - Basic email address format
    - <fred@bedrock.com> instead of "Fred" <fred@bedrock.com>
- **Character encoding**
  - Only "quoted-printable"
  - Assume ISO 8859-15 as character map
- **Error handling**
  - Rudimentary error handling
    - Detect errors
    - Print a reasonably informative error message
    - Don't crash!

Programming Environment

- This assignment is not restricted to a particular programming language. You can choose:
  - C, C++, Java
- Your solution must work on the **development server**
- You get no code from us…
- And you should not use external libraries or packages without permission
  - But for this assignment, expect "no" for an answer…

SMTP Server

- We have set up an SMTP server for you
  - ik2213.ssvl.kth.se
  - Use this server for testing
  - This is the server we will test your solutions against
  - If you try other servers, it may or may not work
    - SMTP servers are restrictive about relaying mail
    - SPAM prevention techniques, such as Greylisting, may temporarily reject mails

Testing

- Test the examples given in the **forum** plus your own test suite
- Wireshark is your best friend
  - (used to be Ethereal)
  - Network sniffer and protocol analyser
  - http://www.wireshark.org/
  - Don't sniff other traffic than your own!
    - It violates school rules, is against good network citizenship, and is pointless for the assignment!
- Tcpdump
  - Network sniffer and (rudimentary) protocol analyser

What to Hand In

- Submit on the course web a file whose name is the surname of the author(s)
- Contents
  - Readme file (text file)
  - Report (2-3 pages; PDF or ODF format)
  - Source code

README

- Clear, step-by-step, set of instructions:
  - How to compile
  - How to configure (e.g. HTTP port)
  - How to run
  - How to use (if any non-trivial feature)

Written Report

- You should describe **in your own words** what you have done
- The **target audience** is at the level of your fellow students, but with limited knowledge in networking and communications
- Possible components
  - Report title and names of group members
  - Introduction
    - What is this about?
    - Background and goal
  - Problem
    - Problem formulation, difficulties?
  - Solution
    - A clear explanation of what you did, but **no source code**!
  - Discussion and conclusions
    - Possible application areas
    - Possible extensions and modifications

When to Hand It In

- You **must** submit your solution by April 10th 8.00 AM
  - A serious attempt to solve the problem
  - You cannot submit after the deadline
- You will receive feedback
- Based on the feedback, you are allowed to fix your solution and resubmit a make-up

Links and Tips

- HTML tutorials on the Web
  - For example, http://webmonkey.wired.com/webmonkey/
- SMTP
  - "telnet *mailserver* 25"
- Wikipedia is an free, open encyclopedia, with good coverage of computing and communication
  - You can find many protocols described there
  - http://www.wikipedia.org/
- Java
- Socket programming