Lab 9: Application-level Network Protocols and Sockets

SMTP

The Simple Mail Transfer Protocol (SMTP) is an internet standard communication protocol for electronic mail transmission. Mail servers and other message transfer agents use SMTP to send and receive mail messages. User-level email clients typically use SMTP only for sending messages to a mail server for relaying (source: Wikipedia).

Sending Email using Socket

Theoretically, we could always send emails using socket, once we know the mail server address and the port number. All we have to do is to open a socket that connects to the mail server and get an `OutputStream` of the socket.

```
Socket s = new Socket("mail.server.com", 25); // 25 is SMTP
PrintWriter out = new PrintWriter(s.getOutputStream(), StandardCharsets.UTF-8)
```

Next, we need to send the mail in correct SMTP format. The SMTP specification states that the email sent to an SMTP server must follow the below syntax (lines must be terminated with `\r` followed by `\n`). This means that we have to "manually" create the correct message format in our code, which is highly error-prone.

```
HELO sending host
MAIL FROM: sender e-mail address
RCPT TO: recipient e-mail address
DATA
Subject: subject
(blank line)
mail message (any number of lines)
.
QUIT
```

A simple demo code using socket to send email could be as follows. **Replace `yourmail@sustech.edu.cn` to your own sustech mail address and replace `your email password` to your own password**. Then execute the program and check your sustech mailbox.

```
Socket socket = new Socket("smtp.sustech.edu.cn", 25);
BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

out.println("HELO localhost");
// 220 means the SMTP server is ready to proceed with the next command.
System.out.println("Response: " + in.readLine());
// 250 means everything went well and your message was delivered to the recipient
server.
```

```
System.out.println("Response: " + in.readLine());


out.println("auth login");
// 334 means input Base64-encoded mail address
System.out.println("Response: " + in.readLine());
out.println(Base64.getEncoder().encodeToString("yourmail@sustech.edu.cn".getBytes(
)));
// 334 means input Base64-encoded password
System.out.println("Response: " + in.readLine());
out.println(Base64.getEncoder().encodeToString("your email password".getBytes()));
// 235: Authentication succeeds
System.out.println("Response: " + in.readLine());

out.println("MAIL FROM: yourmail@sustech.edu.cn");
// 250 means everything went well and your message was delivered to the recipient
server.
System.out.println("Response: " + in.readLine());
// send to this address
out.println("RCPT To: yourmail@sustech.edu.cn");
System.out.println("Response: " + in.readLine());

// mail information
out.println("DATA");
out.println("From: \"Alice\" <alice@qq.com>");
out.println("To: \"Bob\" <bob@qq.com>");
out.println("Date: Tue, 15 Jan 2008 16:11:11-0500");
out.println("Subject: Just wanted to say hi");
out.println();
out.println("This is the message body.");
out.println(".");
out.println("QUIT");

String str;
// 354: start mail input
// 250 OK
while((str = in.readLine())!=null){
System.out.println(str);
}

// clean up
out.close();
in.close();
```

In the past, you might be able to send emails this way. However, due to spam floods these days, modern mail servers typically have built-in authentication checks. Implementing the corresponding SMTP specification and authentication schemes manually would be very tedious.

Nowadays, it would be more efficient and reliable to use ready-made libraries for this purpose, for example, the `JavaMail` API, which handles low-level tedious details and let users focus on the mail content.