|  |  |
| --- | --- |
| **PR Rechnernetze und Netzwerkprogrammierung 621.801/621.802** | **WS 2019 / 2020** |
| **Institut für Informationstechnologie (ITEC)** | **Rainer/Timmerer** |
| **Übungsblatt 05** |  |

**Ü 5.1 Post Office Protocol 3**

|  |  |
| --- | --- |
|  | Machen Sie sich mit dem POP3-Protokoll vertraut. Verwenden Sie dabei das RFC 1939 und untersuchen Sie wie man sich bei einem Mailserver anmeldet und wie man Nachrichten abruft. Beantworten Sie folgende Fragen:   * Wie ist das POP3-Protokoll grundsätzlich aufgebaut?   Ein Server startet den Pop3 Server auf dem 110 Port  Wenn ein Client sich darauf verbinden möchte, sendet der Server eine „Begrüßung“  Der Client und der Server tauschen danach Kommandos und Responses aus.  Kommandos sind „Case-Insensitive“, Keywords sind 3 oder 4 Buchstaben lang und alle Kommandos werden mit CRLF beendet.  Responses sind oft Multiline, wenn das der Fall ist endet jede Line mit einem CRLF und die letzte mit dem „Termination Octet“ 046 und ein CR LF  Eine Pop3 Session durchläuft verschiedene Stadien  Verbinden  „Authorization“ (Username Pwd Abfrage)  „Transaction“ (Austausch)  „Update“ (nach dem „QUIT“ Kommand gibt der POP3 Server die Ressourcen wieder frei und sagt „Tschüss“)   * Ist POP3 ein sicheres Protokoll? Argumentieren Sie warum bzw. warum nicht.    Nein, es stellt keine Verschlüsselung für den Zeitraum der Übertragung zur Verfügung   * Was ist der Unterschied zwischen *single-line* und *multi-line response*?   Single-Line = eine Zeile wird übertragen.  Multi-Line = mehrere Zeilen werden Übertragen. Nachricht wird mit „CRLF.CRLF“ erst beendet.  Vergleichen Sie POP3 mit IMAP! Beschreiben Sie die wesentlichen Unterschiede. |
| **Ü 5.2** | **Java Sockets: POP3-Client** |
|  | Implementieren Sie einen einfachen POP3-Client in Java. Ihr Client soll:  1. sich bei einem POP3-Server **anmelden**, 2. eine **Liste aller Nachrichten** anfordern und  3. eine **Nachricht** vom Server **abrufen**.  Achten Sie auf einen ordnungsgemäßen Verbindungsabbau.  **Hinweise:**   * Verwenden Sie java.net.Socket für die Netzwerkkommunikation und die String.toByte()-Methode um Strings einfach in InputStreams zu schreiben. * Testen Sie Ihre Implementierung mit einem Emailserver, der unverschlüsselte Anmeldung unterstützt. Alternativ können Sie <https://www.stunnel.org/>verwenden, um eine sichere Verbindung mit einem beliebigen POP3-Server Ihrer Wahl herzustellen. |
| **Ü 5.3** | **Java Sockets: POP3-Server** |
|  | Implementieren Sie einen POP3-Server mit minimaler Funktionalität, mit dem ein Email-Client Ihrer Wahl zusammenarbeitet. Nutzen Sie dabei die über den Moodle-Kurs bereitgestellte Datenbank-Klasse *SampleDataBase.java*, die Ihnen eine Grundmenge an Nachrichten zur Verfügung stellt. Ziel ist es, dass die Nachrichten aus der Datenbankklasse in der Inbox eines Email-Clients angezeigt werden. Achten Sie bei Ihrer Implementierung darauf, dass sich mehrere Benutzer gleichzeitig zum Server verbinden können. Verwenden Sie dazu für jeden Benutzer einen eigenen Thread.  **Hinweise:**   * Verzichten Sie auf User-Management und akzeptieren Sie jede beliebige Kombination aus Login-Name und Passwort. * Implementieren Sie, im Falle von Mozilla-Thunderbird, die Kommandos CAPA, USER, PASS, STAT, TOP, RETR, LIST, DELE und QUIT. * Konsultieren Sie für Ihre Implementierung die RFCs 1939 und 2449.   Testen Sie Ihre Implementierungen aus Ü 5.2 und Ü 5.3 mit tatsächlichen Client- und Serverimplementierungen und verfassen Sie einen Erfahrungsbericht inkl. Screenshots. |

Seite 1