

Application Layer, Transport Layer, TCP/IP_Socket_Programming

Udstyr:

2 stk. virtuelle Ubuntu Linux-maskiner kørende i 1 eller 2 laptops
2 stk. USB-Ethernet converters (kan anvendes til kommunikation mellem 2 laptops)
MonoDevelop udviklingsplatform med C-compiler, C++-compiler, C#-compiler,
eller
Python interpreter (findes i I4IKN's Ubuntu-distribution)

TCP-Client/Server:

1. Skriv en iterativ TCP-server med support for en client ad gangen, som kan modtage en tekststreng fra en client. Serveren skal køre i en virtuel Linux-maskine. Tekststrengen skal indeholde et filnavn + en eventuel sti-angivelse. Tilsammen skal informationen i tekststrengen udpege en fil af en vilkårlig type/størrelse beliggende i serveren, som en tilsluttet client ønsker at hente fra serveren. Hvis filen ikke findes skal serveren returnere en fejlmelding til client'en. Hvis filen findes skal den overføres fra server til client i segmenter på 1000 bytes ad gangen – indtil filen er overført fuldstændigt. Serverens portnummer skal være 9000. Server-applikationen skal kunne startes fra en terminal med kommandoen:

`#!/file_server` (for C/C++ applikationers vedkommende)

`#!/file_server.exe` (for C# applikationers vedkommende)

`#python file_server.py` (for Python applikationers vedkommende)

2. Skriv en TCP-client kørende i en anden virtuel Linux-maskine, som kan sende en tekststreng, indtastet af operatøren. Tekststrengen skal indeholde et filnavn + en eventuel sti-angivelse til en fil i TCP-serveren, som er beskrevet i punkt 1. Client'en skal modtage den ønskede fil fejlfrit fra serveren – eller udskrive en fejlmelding hvis filen ikke findes i serveren. Client-applikationen skal kunne startes fra en terminal med kommandoen:

`#!/file_client <file_server's ip-adr.> <[sti] + filnavn>` (for C/C++ applikationers vedkommende)

`#!/file_client.exe <file_server's ip-adr.> <[sti] + filnavn>` (for C# applikationers vedkommende)

`#python file_client.py <file_server's ip-adr.> <[sti] + filnavn>` (for Python applikationers vedkommende)

3. Som kvalitetskontrol for client/server systemet skal den overførte fil kunne sammenlignes med den oprindelige fil vha. terminal-kommandoen:
`cmp <afsendt fil> <modtaget fil>`
eller
`diff -s <afsendt fil> <modtaget fil>`
<afsendt fil> er overført til client vha. af email, ftp eller anden pålidelig, ikke proprietær overføringsmetode.
Der må ikke være forskel mellem filerne, hverken mht. til størrelse eller mht. indhold.

Gennemførelse:

1. Programudvikling og udfærdigelse af journal skal foregå i grupper på 2-4 studerende.

Dokumentation:

1. Beskriv udviklingsforløbet, funktionaliteten og resultatet for øvelse 8 i en journal i størrelsesordenen 5 A4 sider i en .PDF-fil . Aflever desuden jeres source code i en .zip-fil.
Filerne (PDF-journal zip-fil m. source code) afleveres på Blackboard sammen med øvelse 9, når øvelse 9, som omhandler UDP, er udført.

Dvs. øvelse 8 (TCP-socket-programmering) og øvelse 9 (UDP socket-programmering) udgør tilsammen én obligatorisk aflevering.

Journalen for øvelse 8 skal demonstrere forståelsen for udviklingen af TCP-client/server.

Demonstration:

1. Grupperne demonstrerer over for underviser, at TCP-client/server-opstillingen fungerer efter hensigten.

Generelle hints:

Slides: Chapter_2_(Application Layer part 5) – Socket Programming)

C hints:

Filen *Exercise_8_c.zip*, som ligger på BlackBoard
Og et eksempel på et nyttigt link:
http://www.linuxhowtos.org/C_C++/socket.htm

C++ hints:

Filen *Exercise_8_c++.zip*, som ligger på BlackBoard
Og et eksempel på et nyttigt link:
http://www.linuxhowtos.org/C_C++/socket.htm

C# hints:

Filen *Exercise_8_c#.zip*, som ligger på BlackBoard
Og et eksempel på et nyttigt link:
<http://csharp.net-informations.com/communications/csharp-socket-programming.htm>

Python hints:

Filen *Exercise_8_py.zip*, som ligger på BlackBoard
Og et eksempel på et nyttigt link:
<https://docs.python.org/2/howto/sockets.html>