Technische Hochschule Ingolstadt

Projekt
TCP

Wintersemester 16/17

Projektdoku

Informationen zum Einrichten der Example-Umgebung

von

Benjamin Dinkelmeyer, Andreas Lanzl und Sonnia Tsague

Inhaltsverzeichnis

1 Allgemeine Infos

In diesem Dokument wird beschrieben, wie die Umgebung zu installieren ist (siehe 2), zeigt wie man eine Simulation erstellt (3) und listet einige ausgeführte Beispiele auf (4), die mit der IKR-SimLib erstellt worden sind. Zudem ist eine Befehlskette aufgelistet, die dem Benutzer die Ausführung erleichtern soll. Abschießend sind noch eine Hand voll 'known Bugs' in 6 gelistet. Mit der IKR-Simlib (zu finden unter `www.ikr.uni-stuttgart.de/IKRSimLib`) kann Netzwerkverkehr simuliert und grafisch dargestellt werden. Die Datei *TutorialSimLib* liefert noch viele weitere Informationen. Zudem sind dort auch einige Ausführungsbeispiele zu finden.
Achtung: Für die Ausführung des Simulationen ist ein Debian-Linux-System zu verwenden. Zuzüglich werden Adminrechte benötigt

2 Installation der Umgebung

Über das Installationsskript *QemuSimlibEnvInstall.sh* (im Git-Repository im ZIP-File *SimLibEnvironment*zu finden) ist die Installation der Umgebung ein Kinderspiel. Danke Andi :)

1. Download des ZIP-Files
2. Die ZIP-Datei auf einem Ubuntu/Debian System entpacken
3. In einer Shell zum Verzeichnis springen (z.B. `cd ~/Schreibtisch/SimLibEnv`)
4. Das Skript ausführen: `./create-simlib-qemu-simulation.sh <Beliebiger Ordnername>`
   z.B. kann der Befehl so lauten: `./create-simlib-qemu-simulation.sh myFancyIKRFolder`
   Das Skript lädt nun alle erforderlichen IKR-SimLib-Dateien und konfiguriert diese und erzeugt noch einige nützliche Skripte. Die SimLib-Installationsdateien, welche für den Anwender eher unrelevant sind, liegen im Ordner *install*. Die wichtigen Dateien sind in dem Ordner zu finden, dessen Namen man bei der Ausführung des Skripts mitgegeben hat (hier: *myFancyIKRFolder*). Der Inhalt des Ordners ist in Abbildung 1 zu sehen.
5. Wenn es keine Fehler gab, kann fröhlich drauf los simuliert werden #smiley - hierfür siehe nächstes Kapitel Simulation

3 Simulation

In der Abbildung 1 sind einige wichtigen Dateien erkennbar:

sim.par Ist die Parameterdatei, die die Simulation konfiguriert. Das heißt, diese Datei wird verändert um andere Ergebnisse zu erhalten. Über `vim sim.par` kann man sich die Konfiguration ansehen. Alternative zum vim ist natürlich ein grafischer Texteditor ihrer Wahl.

run.sh Ist ein Skript, das die Simulation startet

plot.py Ist ein Skript, das die Ergebnisse grafisch (via Python) darstellt.

view.sh Ist ein Skript, das die Ergebnisse grafisch (via Xmgrace; heißt: nicht so schön) darstellt.

Um die **erste Simulation** zu starten wechselt man über `cd <OrdnerName>` in den Ordner mit den wichtigen Skripten. Hier führt man den Befehl `./run.sh` aus und lehnt sich zurück. Es wird nun anhand der sim.par die Kommunikation simuliert. Das Ergebnis ist in Abbildung 2 zu sehen.

Man erkennt, dass einige neue Dateien hinzugekommen sind. Die Wichtigsten sind im Folgenden erklärt:

*.pcap sind die Dateien, die den Netzwerkverkehr im Wireshark nachvollziehbar machen. Die Files können im Wireshark geöffnet werden. Hierzu den in die Konsole *wireshark* eingeben und eines der .pcap-Files öffnen.

MinimalDumbbel.L4Conection0OfStack*_tcpinfo.csv Enthält die Simulationsergebnisse in tabellarischer Form. Dieses kann ebenfalls über z.B. `vim MinimalDumbbel.L4Conection0OfStack0_tcpinfo.csv` angezeigt werden. Der Stern steht hier für den jeweiligen Client (0 bis n-1).

MinimalDumbbell.stdout Logfile für die Standardausgabe

MinimalDumbbell.stderr Logfile für die Standardfehlerausgabe

Abschließend kann man über
`python plot.py MinimalDumbbell.L4Connection0OfStack0_tcpinfo.csv` (python Darstellung)

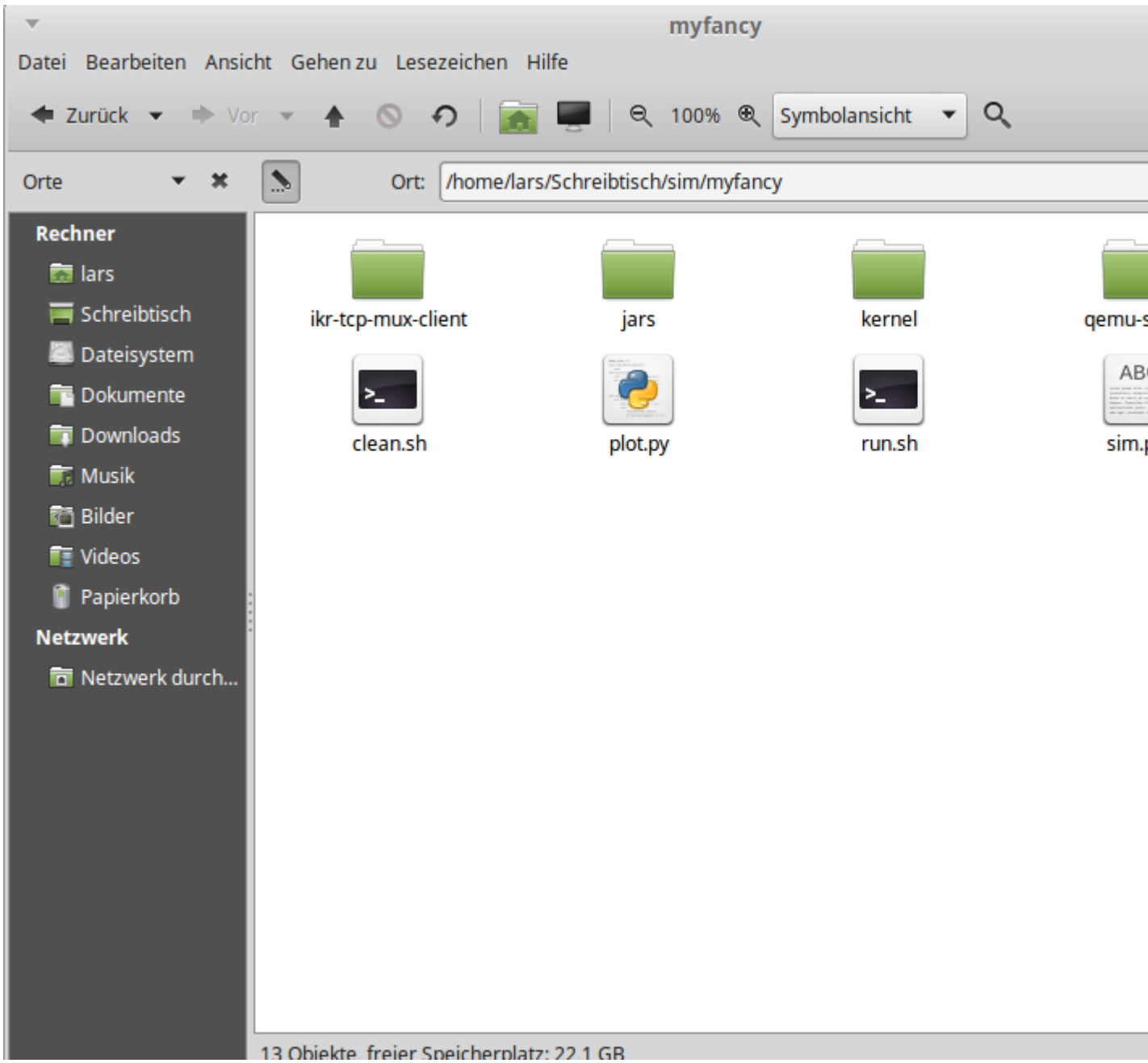


Abbildung 1: Der Ordnerinhalt nach Ausführung des Installers

oder
`./view.sh MinimalDumbbell.L4Connection0OfStack0_tcpinfo.csv` (xmgrace Darstellung)

Das *plot.py*-Python-Skript ist so implementiert, dass nur das Conjestion Window angezeigt wird. Hierfür wird die .csv-Datei mit dem einfachen plot.py aufgerufen.

Wenn man im *sim.par*-File die letzte Zeile (MinimalDumbbell.L4Connection*.LoggedTcpInfoValues...) nicht kommentiert, werden mehr Messwerte (Treshhold usw.) festgehalten und bei der graphischen Visualisierung farblich getrennt dargestellt.

Um mehrere Ergebnisse gleichzeitig anzuzeigen, kann man das eines der beiden python-Skripte mit dem *-Operator aufrufen: z.B. `python plot.py MinimalDumbbell.L4Connection0OfStack*_tcpinfo.csv`.

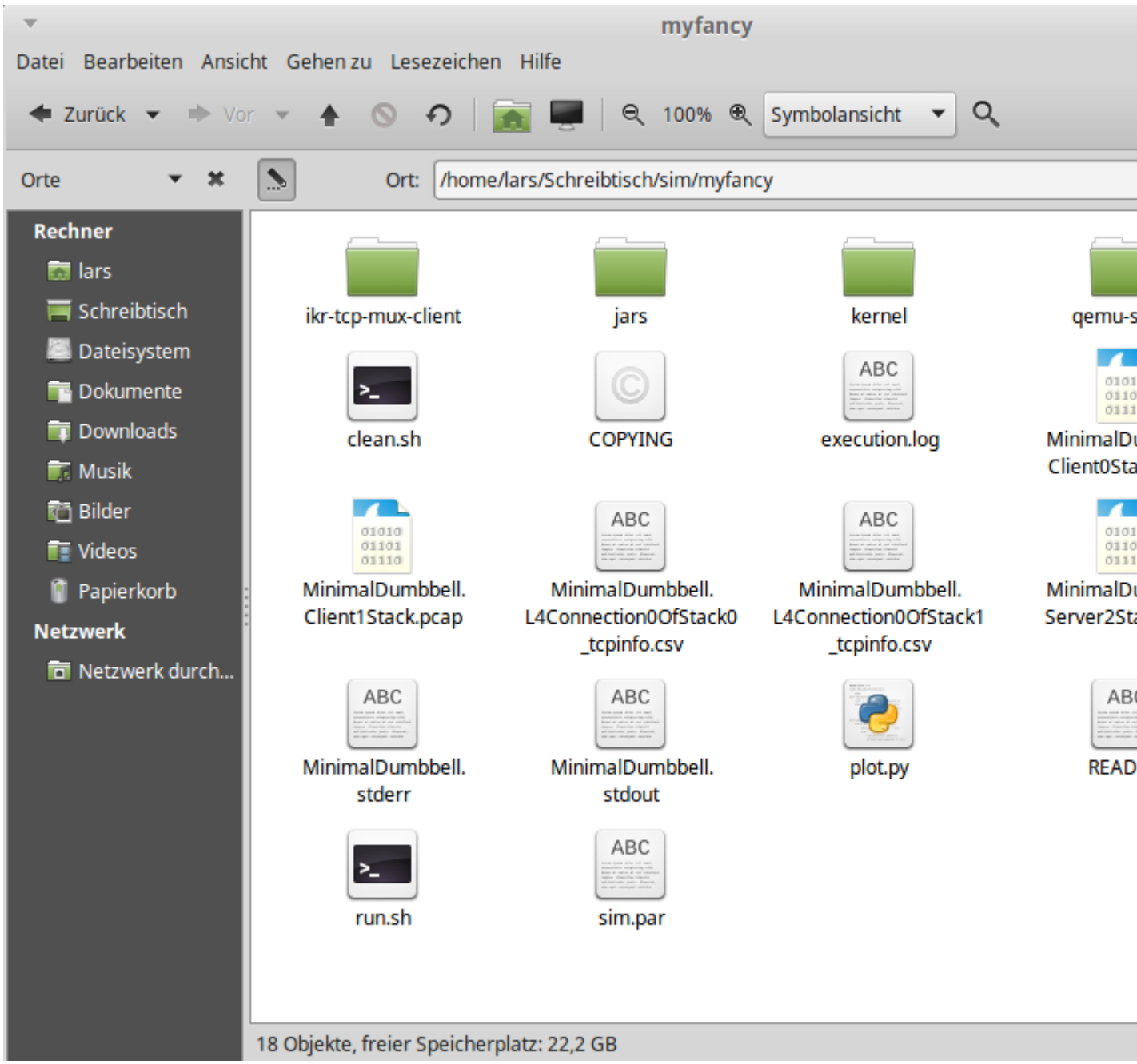
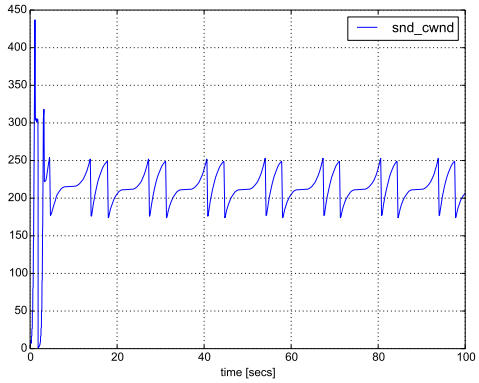


Abbildung 2: Der Ordnerinhalt nach Ausführung von run.sh

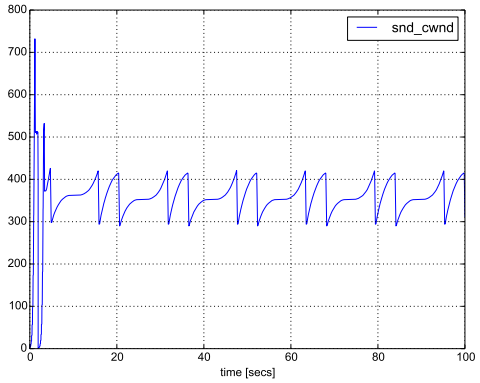
Dies macht natürlich nur Sinn, wenn mehrere Clients oder mehrere Server in der Simulation beteiligt waren.

Für die nächste Simulation muss man 'nur' die Datei *sim.par* anpassen, run.sh und das Pythonskript oder das Shellskript für die grafische Visualisierung ausführen.

Sollte bei einem Plot auffallen, dass der SlowStart einen extrem hohen Wert erreicht hat, so sind alle weiteren Ergebnisse sehr klein und nahe der x-Achse angesiedelt. Um dies zu umgehen, kann über **scale** eine maximale Höhe für den Plot angegeben werden.
z.B. `python plot.py MinimalDumbbell.L4Connection0OfStack*_tcpinfo.csv scale 1000`.



(a) TransmissionRate von 15 $\frac{Mbit}{s}$



(b) TransmissionRate von 25 $\frac{Mbit}{s}$

Abbildung 3: Transmissionrate

Im Tutorial $tutorial_{simuLib} - IRK$ sind bereits einige Beispiele ausgeführt und die jeweiligen sim.par-Einstellungen hinterlegt. Diese kann kopiert werden, um das selbe Ergebnis zu simulieren. Die fertigen sim.par Dateien liegen im Ordner *param-templates*.

4 Ausführungsbeispiele

Eine schnelle sichtbare Änderung erhält man, wenn man für *DumbbellModel.DefaultCongestionControl = reno* konfiguriert. In Abbildung 3 wurde die Transmissionrate jeweils einmal auf 15 $\frac{Mbit}{s}$ bzw. 25 $\frac{Mbit}{s}$ gedrosselt/erhöht. Wie zu erkennen ist, ist die ConjuestionWindow bei einer höheren Transmissionrate erhöht.

In Abbildung 4 wurden drei Clients und ein Server in der Standard sim.par konfiguriert. Man kann gut erkennen, dass zu jedem Zeitpunkt die Summe aller ConjestionWindows ungefähr gleich bleibt.

5 Befehlskette

Die Befehlskette ist in der Regel folgende:

sim.par editieren `vim sim.par`

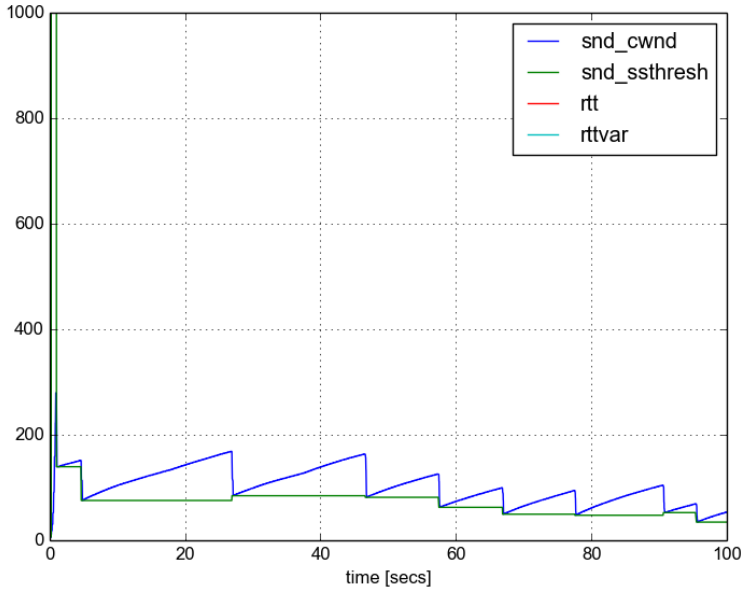
run.sh ausführen `./run.sh`

Ergebnisse visualisieren `python plot.py MinimalDumbbell.L4Connection0OfStack*_tcpinfo.csv`

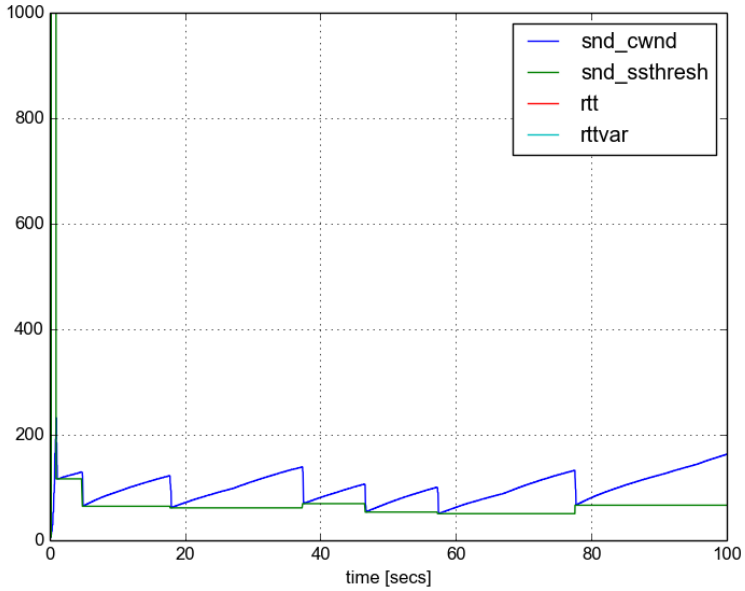
Ergebnisse anzeigen `vim MinimalDumbbell.L4Connection0OfStack0_tcpinfo.csv`

6 Known Issues (ohne Gewähr)

- In der Datei sim.par muss zwischen jedem '=' ein Leerzeichen sein, sonst kommt es zu Fehlern!



(a) Verlauf der Simulation von **Client 1**



(b) Verlauf der Simulation von **Client 2**