



DIPLOMARBEIT

Objektorientierte Entwicklung eines GUI-basierten Tools für die Simulation ereignisbasierter verteilter Systeme

Durchgeführt an der

Fachhochschule Aachen
Fachbereich Elektrotechnik und Informationstechnik

Eupener Str. 70
D-52066 Aachen

mit Erstprüfer und Betreuer Prof. Dr.-Ing. Martin Oßmann
und Zweitprüfer Prof. Dr. rer. nat. Heinrich Fassbender

durch

Cand. Inform. (FH) Paul C. Bütow

Matr.Nr.: 266617
Matthiashofstr. 15
D-52064 Aachen

Aachen, den 22. Juni 2008

Danksagungen

Ohne die Hilfe bestimmter Personen wäre die Anfertigung dieser Diplomarbeit viel schwieriger gewesen. Daher möchte ich mich bei den Folgenden bedanken:

- Martin Oßmann für die Betreuung der Diplomarbeit und der Bereitstellung des für mich sehr interessanten Themas
- Andre Herbst, für das Testen des Simulators; durch seine Hilfe wurden viele Mängel und Bugs aufgedeckt
- Mein Bruder Florian Bütow, für Tipps und Tricks rund um Java, für die Bereitstellung eines Buches sowie für das Testen des Simulators
- Meine Eltern Jörn und Leslie Bütow, die mir das Studium ermöglichten und stets für alle Dinge ein offenes Ohr hatten sowie für das Sponsoring eines weiteren Buches
- Die Open Source Gemeinde; diese Diplomarbeit wurde, einschließlich dieses Dokuments, mit 100% Open Source Software erstellt (mit Ausnahme der BIOS-Version des PCs)

Inhaltsverzeichnis

1. Einleitung	7
1.1. Was ist ein Verteiltes System?	7
1.2. Motivation	7
2. Grundbegriffe	8
2.1. Client/Server Modell	8
2.2. Prozesse und deren Rollen	9
2.3. Nachrichten	9
2.4. Lokale und globale Uhren	9
2.5. Ereignisse	10
2.6. Protokolle	10
3. Der Simulator	12
3.1. Die grafische Benutzerschnittstelle	12
3.1.1. Die Menüleiste	13
3.1.2. Die Toolbar	13
3.1.3. Die Visualisierung	14
3.1.4. Die Sidebar	16
3.1.5. Loggfenster	18
3.1.6. Expertenmodus	18
3.2. Ereignisse	18
3.2.1. Prozessabsturz	18
3.2.2. Prozesswiederbelebung	18
3.2.3. Aktivierung und Deaktivierung von Protokollen	18
3.2.4. Weitere Protokollereignisse	18
3.3. Protokolle	18
3.3.1. Beispiel (Dummy) Protokoll	18
3.3.2. Das Ping-Pong Protokoll	18

Inhaltsverzeichnis

3.3.3. Das Broadcast-Sturm Protokoll	18
3.3.4. Das Protokoll zur internen Synchronisierung in einem syn- chronen System	18
3.3.5. Christians Methode zur externen Synchronisierung	18
3.3.6. Berkeley Algorithmus zur internen Synchronisation	18
3.3.7. Das Ein-Phasen Commit Protokoll	18
3.3.8. Das Zwei-Phasen Commit Protokoll	18
3.3.9. Der ungenügende (Basic) Multicast	18
3.3.10. Der zuverlässige (Reliable) Multicast	18
3.4. Zeitformate	18
3.4.1. "Normale Zeit"	18
3.4.2. Die Logische Uhr von Lamport	18
3.4.3. Die Vektor-Zeitstempel	18
3.5. Einstellungen	18
3.5.1. Simulationseinstellungen	18
3.5.2. Prozesseinstellungen	18
3.5.3. Protokolleinstellungen	18
4. Die Implementierung	19
4.1. Gliederung der Pakete	20
4.2. Editoren	20
4.3. Ereignisse	20
4.3.1. Interne Ereignisse	20
4.4. Protokolle	20
4.4.1. Protokoll-API	20
4.5. Serialisierung von Simulationen	20
4.5.1. Rückwärtskompatibel	20
4.6. Programmierrichtlinien	20
4.7. Entwicklungsumgebung	20
5. Ausblick	21
A. Schemata	22
B. Akronyms	23
C. Literaturverzeichnis	24

Abbildungsverzeichnis

2.1. Client/Server Modell	8
2.2. Client/Server Protokolle	10
3.1. Standard GUI	12
3.2. "Datei"-Menü	13
3.3. Die Menüzeile inklusive Toolbar	14
3.4. Visualisierung einer noch nicht gestarteten Simulation	15

Tabellenverzeichnis

Kapitel 1.

Einleitung

1.1. Was ist ein Verteiltes System?

1.2. Motivation

[Bri03] Zitat.

Kapitel 2.

Grundbegriffe

Für das Verständnis wie die Simulation von verteilten Systemen funktioniert, werden hier einige Grundbegriffe erläutert. Eine Vertiefung findet erst in den nachfolgenden Kapiteln statt.

2.1. Client/Server Modell

Der Simulator basiert auf dem Client/Server Prinzip. Bei jeder sinnvollen Simulation gibt es mindestens einen teilnehmenden Client und einen Server, die miteinander über Nachrichten (s.u.) kommunizieren (Abbildung 2.1). Bei komplexen Simulationen können auch mehrere Clients und/oder Server mitwirken. In der Regel empfangen Server nur Nachrichten, die von Clients verschickt wurden und vice versa.

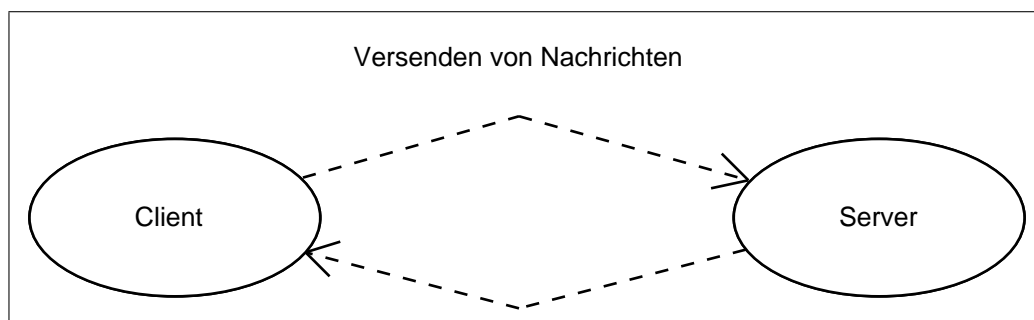


Abbildung 2.1.: Client/Server Modell

2.2. Prozesse und deren Rollen

Ein verteiltes System wird anhand von Prozessen simuliert. Jeder Prozess nimmt hierbei eine oder mehrere Rollen ein. Beispielsweise kann ein Prozess die Rolle eines Clients einnehmen und ein weiterer Prozess die Rolle eines Servers. Ein Prozess kann auch Client und Server gleichzeitig sein. Es ist auch möglich, dass ein Prozess die Rollen mehrerer Server und Clients auf einmal einnimmt. Ob das sinnvoll ist hängt vom Szenario ab. Um einen Prozess zu kennzeichnen besitzt jeder Prozess eine **eindeutige** Prozess-Identifikationsnummer (PID).

2.3. Nachrichten

Damit das Client/Server Modell angewandt werden kann, müssen Nachrichten verschickt werden können. Eine Nachricht kann von einem Client- oder Serverprozess verschickt werden und kann beliebig viele Empfänger haben. Der Inhalt einer Nachricht hängt vom verwendeten Protokoll (s.u.) ab. Um eine Nachricht zu kennzeichnen besitzt jede Nachricht eine **eindeutige** Nachrichten-Identifikationsnummer (NID).

2.4. Lokale und globale Uhren

In einer Simulation gibt es **genau eine** globale Uhr. Sie stellt die aktuelle und **immer korrekte** Zeit dar. Eine globale Uhr geht nie falsch.

Zudem besitzt jeder beteiligter Prozess eine eigene lokale Uhr. Sie stellt die aktuelle, jedoch nicht zwangsmäßig global-korrekte, Zeit des jeweiligen Prozesses dar. Wenn die Prozesszeit nicht korrekt ist (nicht der globalen Zeit gleicht), dann wurde die Prozessuhr entweder im Laufe einer Simulation neugestellt oder sie besitzt eine Uhrabweichung. Eine Uhrabweichung gibt an, um wieviel eine Uhr falsch geht. Wenn eine lokale Uhr nicht neugesetzt wird und auch keine Uhrabweichung hat, dann gibt sie stets die korrekte globale Zeit wieder.

2.5. Ereignisse

Eine Simulation besteht aus der Hintereinanderausführung von endlich vielen Ereignissen. Beispielsweise kann es ein Ereignis geben, welches einen Prozess eine Nachricht verschicken läßt oder den Prozess selbst abstürzen läßt. Jedes Ereignis tritt zu einem bestimmten Zeitpunkt ein. Wenn es zeitgleiche Ereignisse gibt, so werden sie ebenso hintereinander ausgeführt, behalten aber in der Simulation die selben Ausführzeiten.

2.6. Protokolle

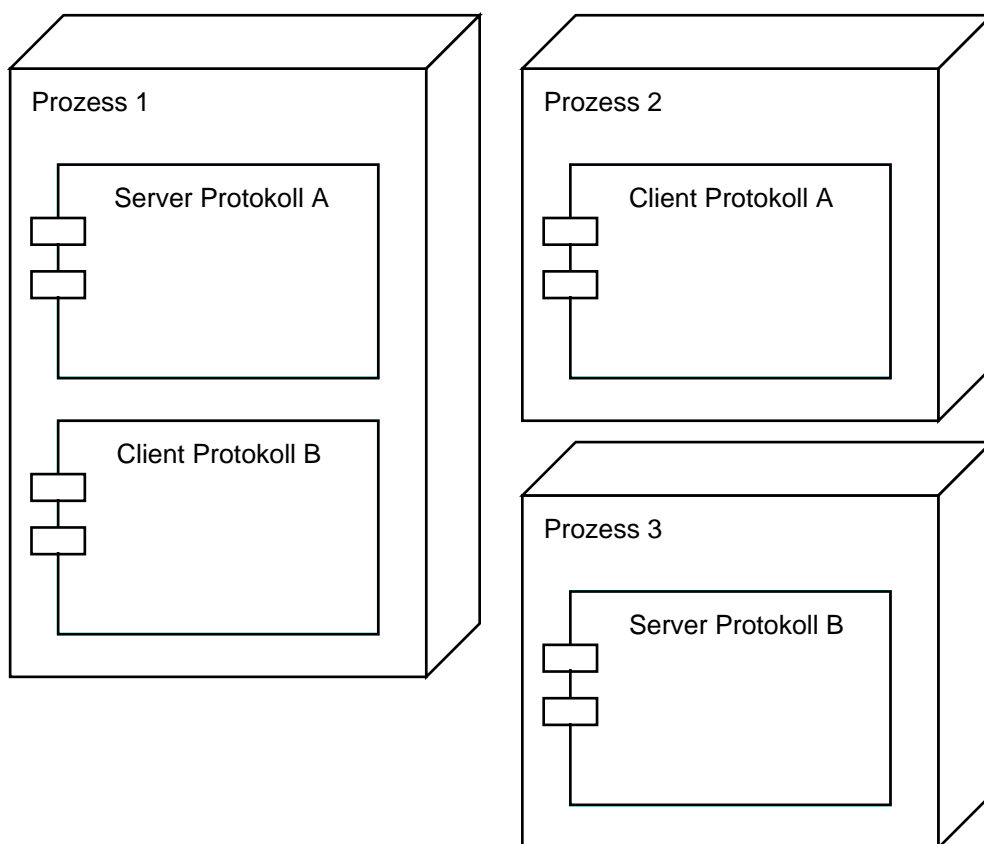


Abbildung 2.2.: Client/Server Protokolle

Eine Simulation besteht aus der Anwendung von Protokollen. Es wurde bereits erwähnt, dass ein Prozess die Rollen von Servern und/oder Clients annehmen kann. Bei jeder Server- und Clientrolle muss zusätzlich das dazugehörige Protokoll spezifiziert werden. Ein Protokoll definiert, wie ein Client und ein Server Nachrichten verschickt und wie bei Ankunft einer

Nachricht reagiert wird. Ein Prozess verarbeitet eine empfangene Nachricht nur, wenn er das jeweilige Protokoll versteht.

In Abbildung 2.2 sind 3 Prozesse dargestellt. Prozess 1 unterstützt serverseitig das Protokoll "A" und clientseitig das Protokoll "B". Prozess 2 unterstützt clientseitig das Protokoll "A" und Prozess 3 serverseitig das Protokoll "B". D.h., Prozess 1 kann mit Prozess 2 via Protokoll "A" und mit Prozess 3 via Protokoll "B" kommunizieren. Die Prozesse 2 und 3 sind zueinander inkompatibel und können voneinander erhaltene Nachrichten nicht verarbeiten.

In der Regel können Clients nicht mit Clients und Server nicht mit Server kommunizieren. Je nach verwendetem Protokoll kann dies jedoch variieren. Alle vom Simulator verfügbaren Protokolle werden später behandelt.

Kapitel 3.

Der Simulator

3.1. Die grafische Benutzerschnittstelle

Die grafische Benutzerschnittstelle (GUI; Graphical User Interface) präsentiert sich standardmäßig wie in Abbildung 3.1.

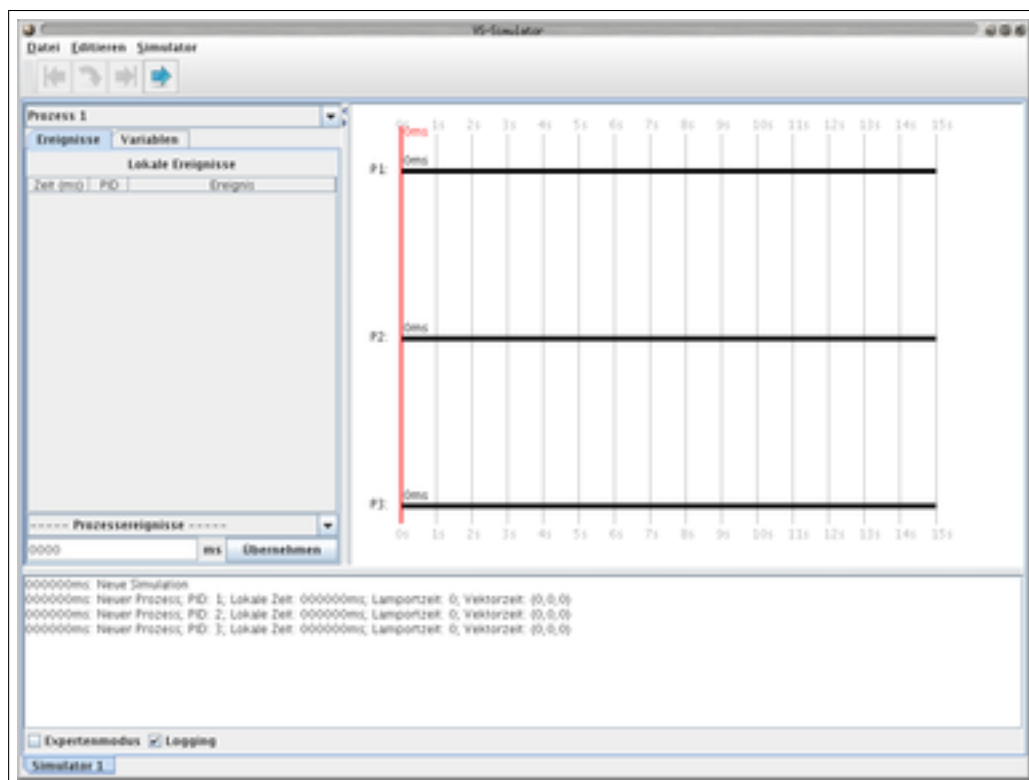


Abbildung 3.1.: Standard GUI

Im folgenden werden in die einzelnen Komponenten des GUIs eingegangen.

3.1.1. Die Menüzeile

Im Datei-Menü (Abbildung 3.2) lassen sich neue Simulationen erstellen oder die aktuell geöffnete Simulation schliessen. Neue Simulationen öffnen sich standardmäßig in einem neuen Tab. Es können allerdings auch neue Simulationsfenster, die wiederum eigene Tabs besitzen, geöffnet oder geschlossen werden. In jedem Tab befindet sich eine von den Anderen vollständig unabhängige Simulation. Es können somit beliebig viele Simulationen parallel ausgeführt werden. Die Menüeinträge “Öffnen”, “Speichern” und “Speichern unter” dienen für das Laden und Speichern von Simulationen.

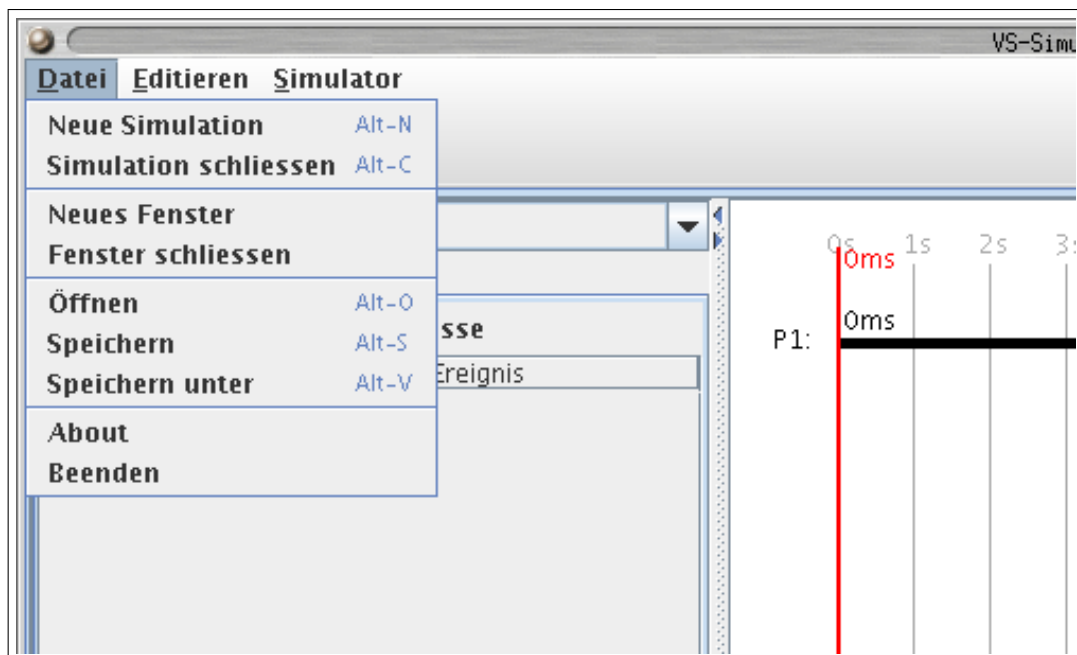


Abbildung 3.2.: “Datei”-Menü

Über das “Editieren”-Menü gelangt man zu den Simulationseinstellungen. Es werden auch alle beteiligten Prozesse zum Editieren aufgelistet. Und das Menü “Simulator” bietet die selben Optionen wie die Toolbar (s.u.).

3.1.2. Die Toolbar

Oben links befindet sich die Toolbar (Abbildung 3.3) des Simulators. Die Toolbar enthält die Funktionen, die vom Benutzer am schnellsten und häufigsten verwendet werden.

Die Toolbar bietet vier verschiedene Aktionen an:

- Starten der Simulation, kann nur betätigt werden, wenn die Simulation derzeit nicht läuft.
- Pausieren der Simulation, kann nur betätigt werden, wenn die Simulation derzeit läuft.
- Wiederholen der Simulation, kann nicht betätigt werden, wenn die Simulation noch nicht gestartet wurde.
- Zurücksetzen der Simulation, kann nur betätigt werden, wenn die Simulation pausiert wurde oder wenn die Simulation abgelaufen ist.



Abbildung 3.3.: Die Menüleiste inklusive Toolbar

Die Toolbar lässt sich auch nach Belieben repositionieren (z.B. links, rechts oder unten des Simulatorfensters). Hierfür muss per "Drag-n-Drop" die "raue Fläche" zur Zielposition gezogen werden.

3.1.3. Die Visualisierung

Mittig rechts (Abbildung 3.1) befindet sich die grafische Repräsentation der Simulation. Die X-Achse repräsentiert die Zeit in Millisekunden. Die aktuelle Simulation endet nach genau 15 Sekunden. In Abbildung 3.4 sind 3 Prozesse (mit den PIDs 1, 2 und 3) dargestellt, die jeweils einen eigenen horizontalen schwarzen Balken besitzen. Auf diesen Prozessbalken kann man die jeweilige lokale Prozesszeit ablesen. Die vertikale rote Linie stellt die globale Zeit dar.

Die Prozessbalken dienen auch für Start- und Zielpunkte von Nachrichten. Wenn beispielsweise Prozess 1 eine Nachricht zum Prozess 2 verschickt, so wird eine Linie vom einen Prozessbalken zum Anderen gezeichnet. Nachrichten, die ein Prozess an sich selbst schickt, werden nicht visualisiert.

Generell kann die Anzahl der Prozesse nach belieben variieren. Die Dauer der Simulation beträgt mindestens 5 -und maximal 120 Sekunden. Die Simulation endet erst, wenn die globale Zeit 15 Sekunden erreicht hat, und nicht, wenn eine lokale Prozesszeit die 15 Sekunden erreicht.

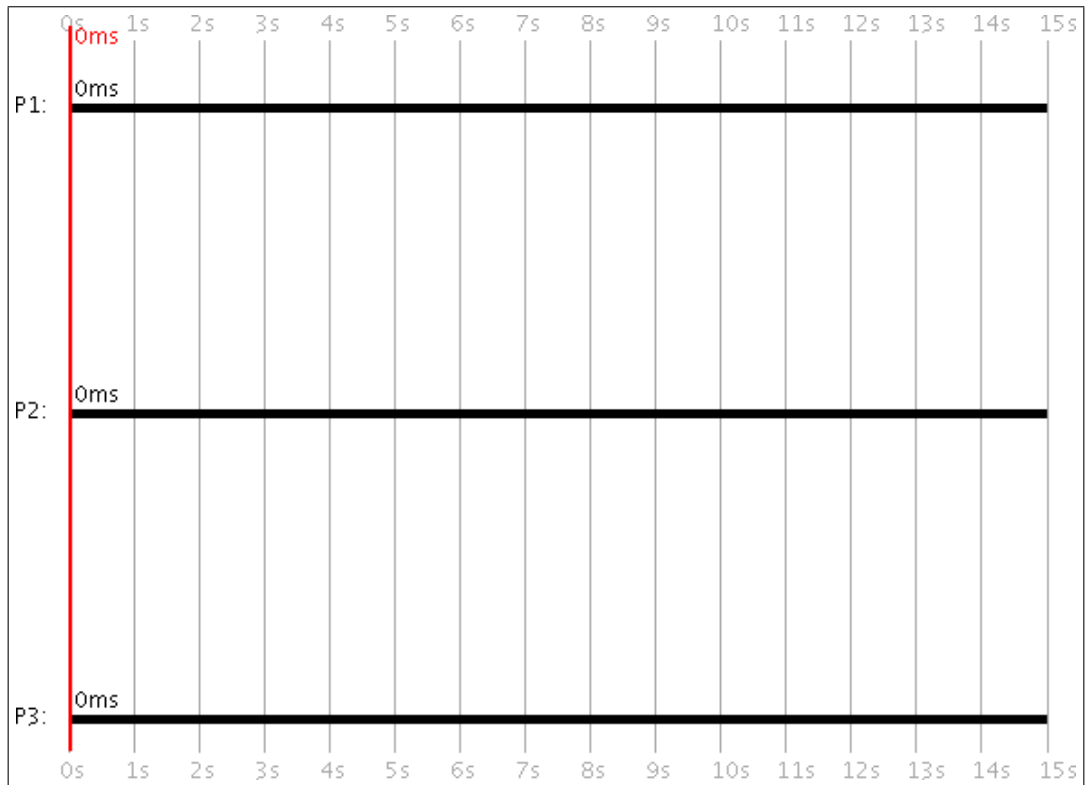


Abbildung 3.4.: Visualisierung einer noch nicht gestarteten Simulation

3.1.4. Die Sidebar

[illegible]

Ereignisse

Variablen

3.1.5. Loggfenster

3.1.6. Expertenmodus

Lamport- und Vektorzeit

Loggfilter

Lokale und globale Ereignisse

3.2. Ereignisse

3.2.1. Prozessabsturz

3.2.2. Prozesswiederbelebung

3.2.3. Aktivierung und Deaktivierung von Protokollen

3.2.4. Weitere Protokollereignisse

3.3. Protokolle

3.3.1. Beispiel (Dummy) Protokoll

3.3.2. Das Ping-Pong Protokoll

3.3.3. Das Broadcast-Sturm Protokoll

**3.3.4. Das Protokoll zur internen Synchronisierung in einem
synchronen System**

3.3.5. Christians Methode zur externen Synchronisierung

3.3.6. Berkeley Algorithmus zur internen Synchronisation

3.3.7. Das Ein-Phasen Commit Protokoll

Kapitel 4.

Die Implementierung

4.1. Gliederung der Pakete

4.2. Editoren

4.3. Ereignisse

4.3.1. Interne Ereignisse

4.4. Protokolle

4.4.1. Protokoll-API

4.5. Serialisierung von Simulationen

4.5.1. Rückwärtskompatibel

4.6. Programmierrichtlinien

4.7. Entwicklungsumgebung

Kapitel 5.

Ausblick

Anhang A.

Schemata

Anhang B.

Akronyms

GUI Graphical User Interface

NID Nachrichten-Identifikationsnummer

PID Prozess-Identifikationsnummer

VS Verteiltes System

Anhang C.

Literaturverzeichnis

[Bri03] Brian.H.Wilcox. Robotic vehicle system engineering. Internet, April 2003.
<http://team.caltech.edu/members/RoboticVehicleSystemEngineering3.pdf>.