



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. Diplom-Inform. (FH) Paul C. Bütow

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

Aachen, den 23. 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	14
3.1.3. Die Visualisierung	15
3.1.4. Die Sidebar	17
3.1.5. Das Loggfenster	19
3.1.6. Der Expertenmodus	20
3.2. Ereignisse	24
3.2.1. Prozessabsturz	24
3.2.2. Prozesswiederbelebung	24
3.2.3. Aktivierung und Deaktivierung von Protokollen	24
3.2.4. Weitere Protokollereignisse	24
3.3. Protokolle	24
3.3.1. Beispiel (Dummy) Protokoll	24
3.3.2. Das Ping-Pong Protokoll	24

Inhaltsverzeichnis

3.3.3. Das Broadcast-Sturm Protokoll	24
3.3.4. Das Protokoll zur internen Synchronisierung in einem syn- chronen System	24
3.3.5. Christians Methode zur externen Synchronisierung	24
3.3.6. Berkeley Algorithmus zur internen Synchronisation	24
3.3.7. Das Ein-Phasen Commit Protokoll	24
3.3.8. Das Zwei-Phasen Commit Protokoll	24
3.3.9. Der ungenügende (Basic) Multicast	24
3.3.10. Der zuverlässige (Reliable) Multicast	24
3.4. Zeitformate	24
3.4.1. "Normale Zeit"	24
3.4.2. Die Logische Uhr von Lamport	24
3.4.3. Die Vektor-Zeitstempel	24
3.5. Einstellungen	24
3.5.1. Simulationseinstellungen	24
3.5.2. Prozesseinstellungen	24
3.5.3. Protokolleinstellungen	24
4. Die Implementierung	25
4.1. Gliederung der Pakete	26
4.2. Editoren	26
4.3. Ereignisse	26
4.3.1. Interne Ereignisse	26
4.4. Protokolle	26
4.4.1. Protokoll-API	26
4.5. Serialisierung von Simulationen	26
4.5.1. Rückwärtskompatibel	26
4.6. Programmierrichtlinien	26
4.7. Entwicklungsumgebung	26
5. Ausblick	27
A. Schemata	28
B. Akronyms	29
C. Literaturverzeichnis	30

Abbildungsverzeichnis

2.1. Client/Server Modell	8
2.2. Client/Server Protokolle	10
3.1. Der Simulator nach dem ersten Starten	12
3.2. "Datei"-Menü	13
3.3. Standard GUI	14
3.4. Die Menüzeile inklusive Toolbar	14
3.5. Visualisierung einer noch nicht gestarteten Simulation	15
3.6. Rechtsklick auf einen Prozessbalken	16
3.7. Die Sidebar	17
3.8. Die Ereignisauswahl via Sidebar	18
3.9. Der Ereigniseditor mit 3 programmierten Ereignissen	19
3.10. Das Loggfenster	19

Tabellenverzeichnis

3.1. Farbliche Differenzierung von Prozessen und Nachrichten	17
--	----

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 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.

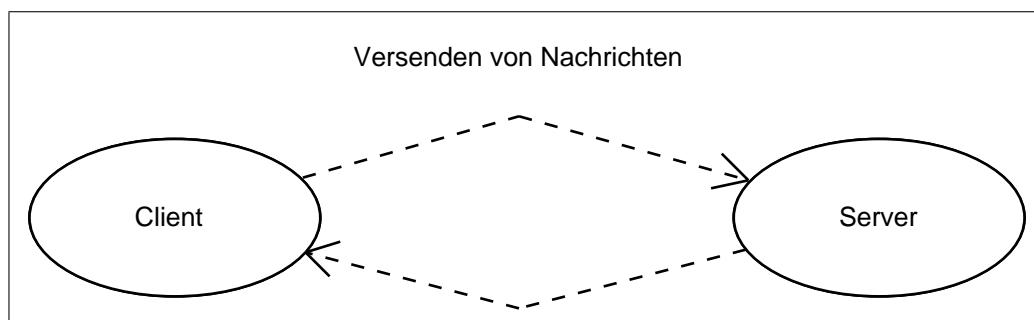


Abbildung 2.1.: Client/Server Modell

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 ab. Was unter einem Protokoll zu verstehen ist, wird später behandelt. 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

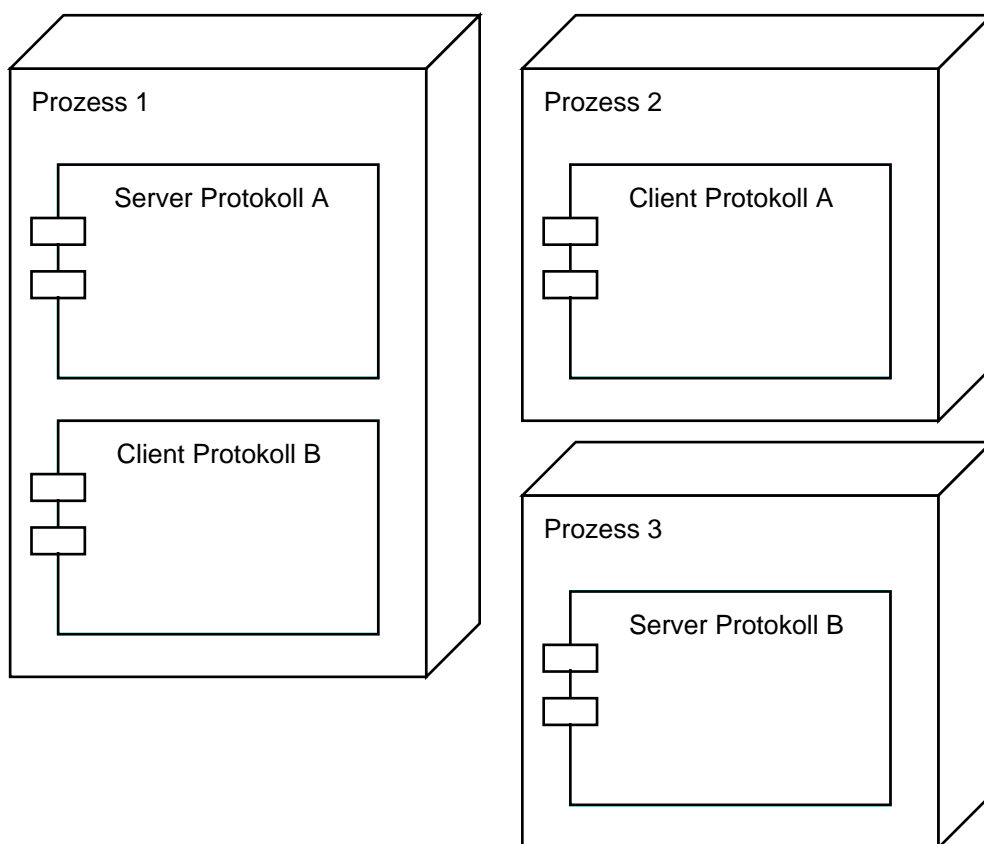


Abbildung 2.2.: Client/Server Protokolle

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

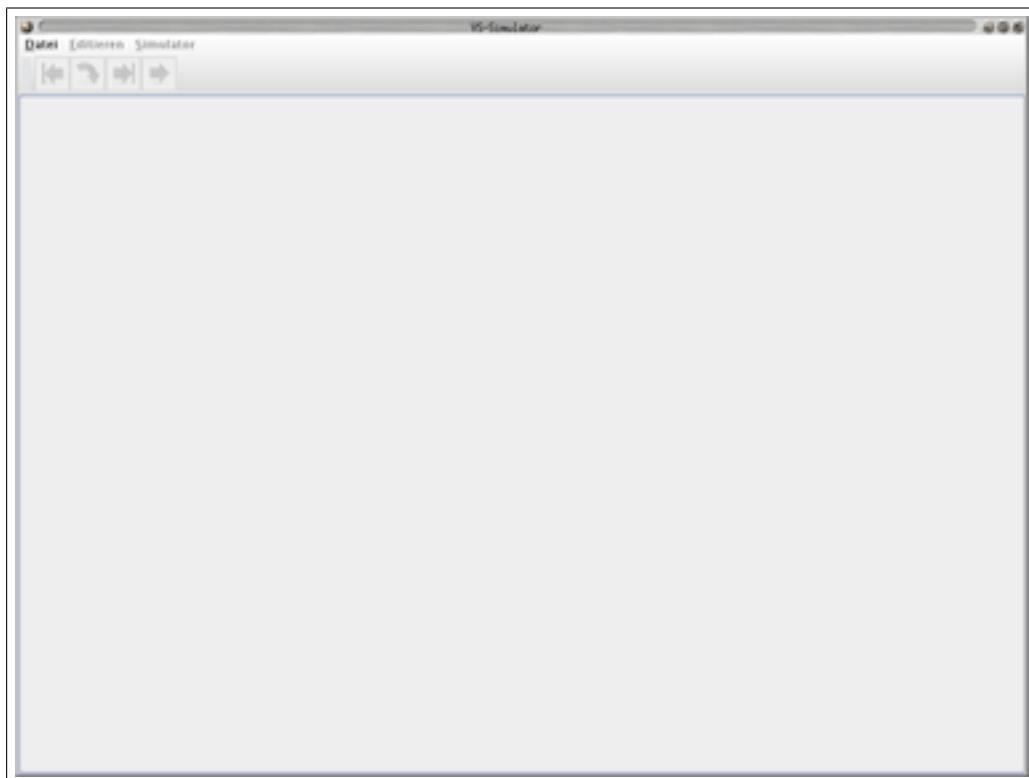


Abbildung 3.1.: Der Simulator nach dem ersten Starten

Der Simulator präsentiert sich nach dem ersten Starten wie in Abbildung 3.1. Für die Erstellung einer neuen Simulation wird im Menü "Datei" (Abbildung 3.2) der Punkt "Neue Simulation" ausgewählt, wo anschließend das Einstellungsfenster für die neu zu erstellende

Simulation erscheint. Auf die einzelnen Optionen wird später genauer eingegangen und es werden nun nur die Standardeinstellungen übernommen. Die GUI mit einer frischen Simulation sieht dann wie in Abbildung 3.3 aus.



Abbildung 3.2.: "Datei"-Menü

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.

Über das "Editieren"-Menü gelangt man zu den Simulationseinstellungen. Es werden auch alle beteiligten Prozesse zum Editieren aufgelistet. Das "Simulator"-Menü bietet die selben Optionen wie die Toolbar, welche im nächsten Teilkapitel beschrieben wird.

Einige Menüs sind erst erreichbar, wenn im aktuellen Fenster bereits eine Simulation erstellt oder geladen wurde.

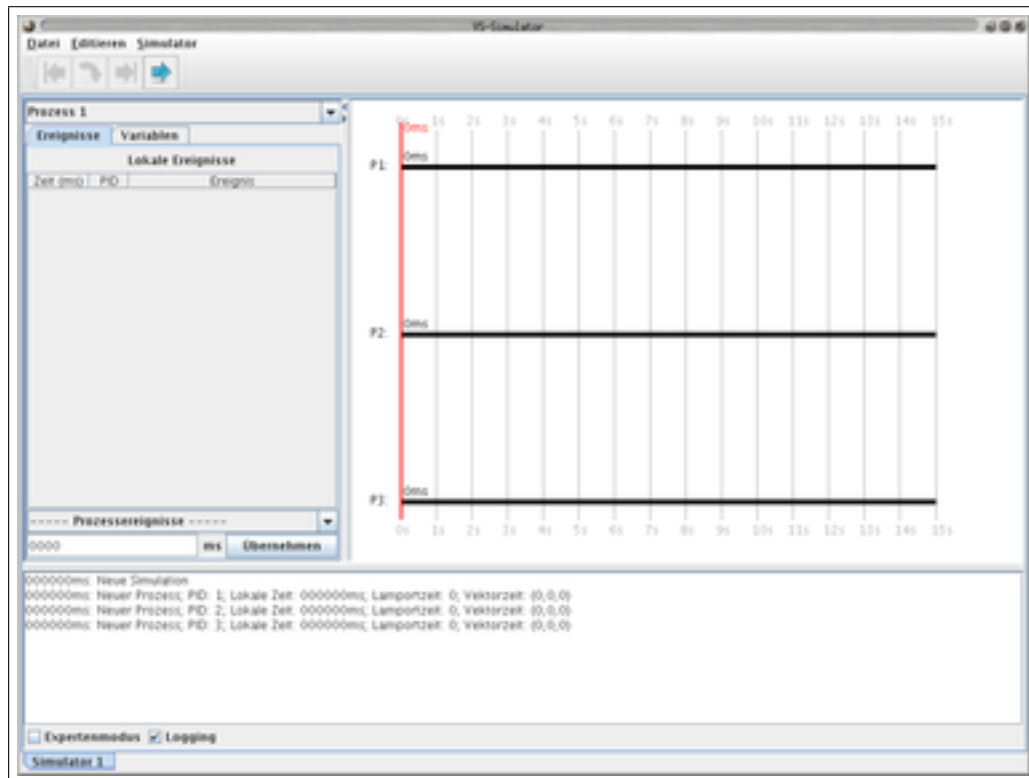


Abbildung 3.3.: Standard GUI

3.1.2. Die Toolbar

Oben links befindet sich die Toolbar (Abbildung 3.4) des Simulators. Die Toolbar enthält die Funktionen, die vom Benutzer am häufigsten verwendet werden.



Abbildung 3.4.: Die Menüleiste inklusive Toolbar

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.

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

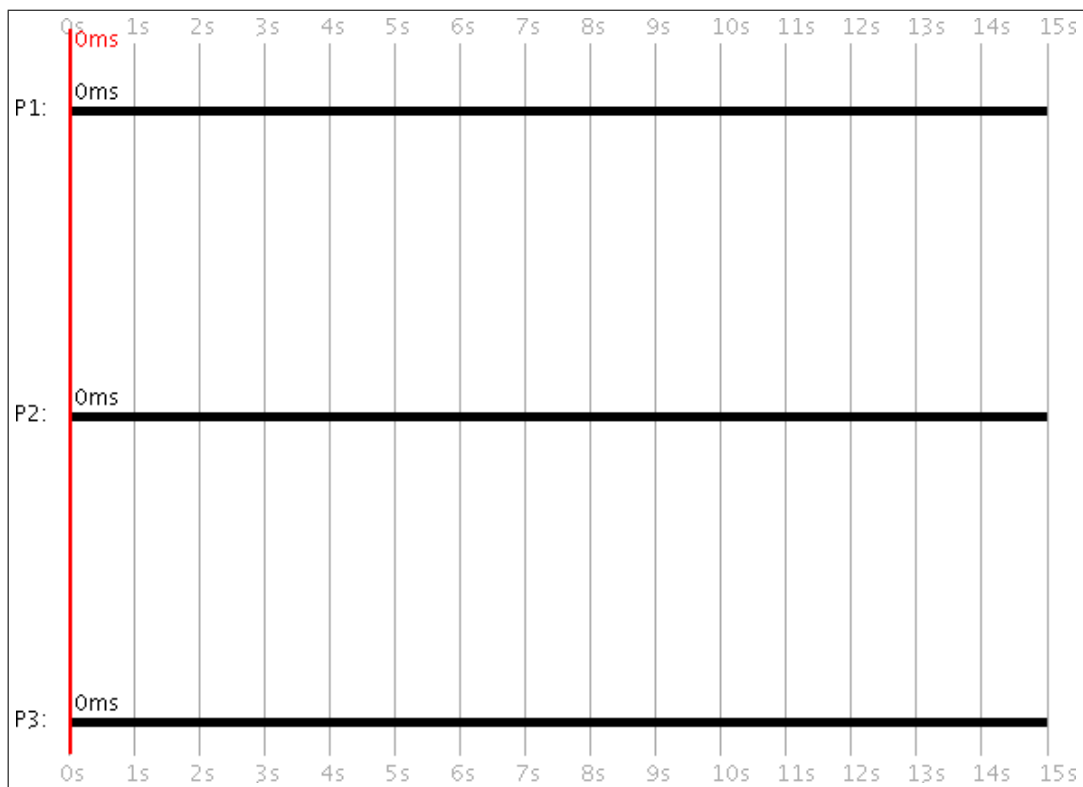


Abbildung 3.5.: Visualisierung einer noch nicht gestarteten Simulation

Mittig rechts (Abbildung 3.3) 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 3.5 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.

Mit einem Linksklick auf einen Prozessbalken ist es ebenso möglich einen Prozess zu editieren. Dies muss also nicht zwingend über das "Simulator"-Menü geschehen. Ein Rechtsklick hingegen öffnet ein Popup-Fenster mit weiteren Auswahlmöglichkeiten (Abbildung 3.6). Ein Prozess kann über das Popup-Menü nur dann abstürzen oder wiederbelebt werden, wenn die Simulation aktuell läuft.

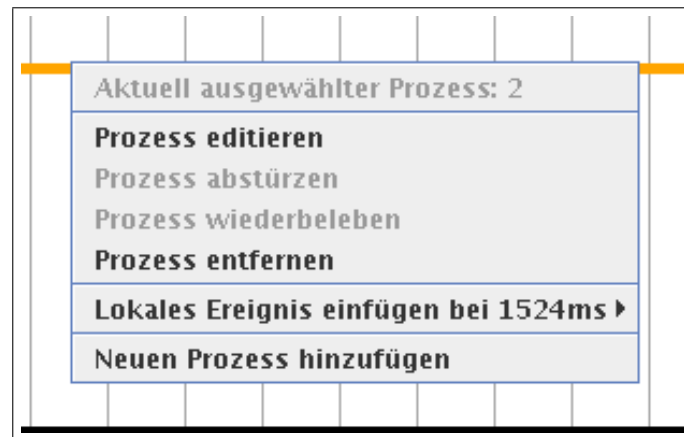


Abbildung 3.6.: Rechtsklick auf einen Prozessbalken

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.

Farbliche Differenzierung

Für Visualisierung einer Simulation spielen Farben eine große Rolle. Standardmäßig werden die Prozesse (Prozessbalken) und Nachrichten mit den Farben wie in Tabelle 3.1 aufgelistet dargestellt. Dies sind lediglich die Standardfarben, welche man über die Einstellungen neudefinieren kann.

Prozessfarbe	Bedeutung
Schwarz	Simulation läuft derzeit nicht (z.B. noch nicht gestartet, abgelaufen oder pausiert)
Orange	Die Maus befindet sich über den Prozessbalken
Rot	Der Prozess ist abgestürzt
Nachrichtfarbe	Bedeutung
Grün	Die Nachricht ist noch unterwegs hat das Ziel noch nicht erreicht
Blau	Die Nachricht hat das Ziel erfolgreich erreicht
Rot	Die Nachricht ging verloren (entweder weil der Zielprozess abgestürzt ist oder weil sie unterwegs verloren ging)

Tabelle 3.1.: Farbliche Differenzierung von Prozessen und Nachrichten

3.1.4. Die Sidebar

Mithilfe der Sidebar (mittig links, Abbildung 3.3) lassen sich Ereignisse bestimmter Prozesse verwalten. Ganz oben (Abbildung 3.7) ist der zu verwaltende Prozess selektiert (hier mit der PID 1). In dieser Prozessauswahl gibt es auch die Möglichkeit "Alle Prozesse" auszuwählen, womit die Ereignisse aller Prozesse gleichzeitig verwaltet werden können. Unter "Lokale Ereignisse" versteht man die Ereignisse, die auftreten, wenn die zum jeweiligen Prozess lokale Prozesszeit eingetreten ist. Die darunterliegende Ereignistabelle listet alle programmierten Ereignisse (hier noch keine vorhanden) mitsamt Eintrittszeit sowie PID auf.

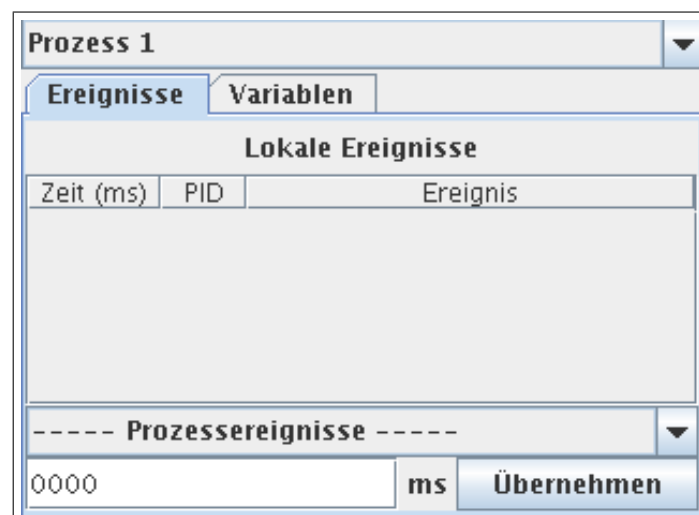


Abbildung 3.7.: Die Sidebar

Für die Erstellung eines neuen Ereignisses kann man entweder mit Rechtsklick auf einen

Prozessbalken (Abbildung 3.6) klicken, oder unterhalb der Ereignistabelle ein Ereignis auswählen (Abbildung 3.8), im Textfeld die Zeit eintragen und auf “Übernehmen” klicken. Beispielsweise wurden auf Abbildung 3.9 drei Ereignisse hinzugefügt: Absturz nach 123ms, Wiederbelebung nach 321ms und erneuerter Absturz nach 3000ms des Prozesses mit der ID 1.

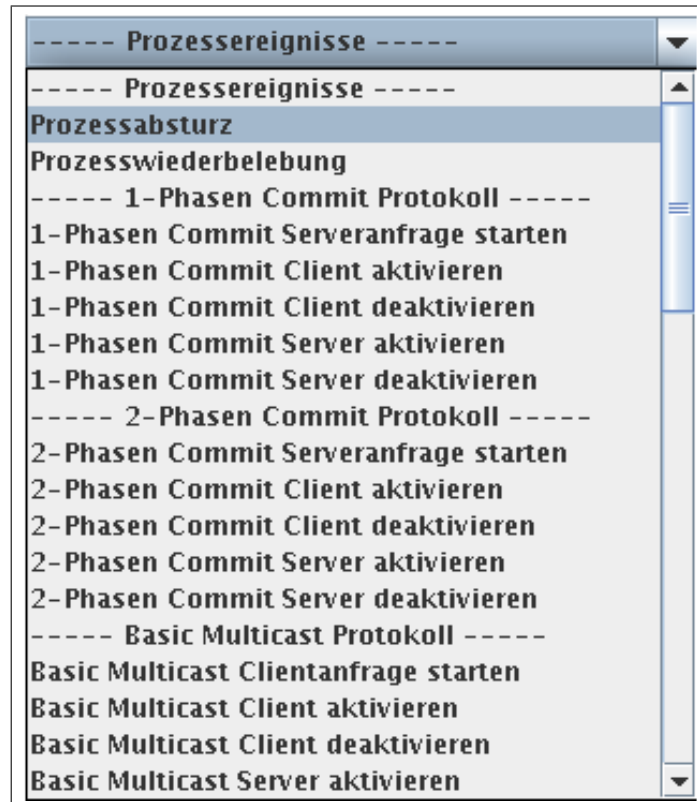


Abbildung 3.8.: Die Ereignisauswahl via Sidebar

Mit Rechtsklick lassen sich alle selektierten Ereignisse entweder kopieren oder löschen. Die Spalten für die Zeit und der PID lassen sich nachträglich editieren. Somit besteht eine komfortable Möglichkeit bereits programmierte Ereignisse auf eine andere Zeit zu verschieben oder einem anderen Prozess zuzuweisen.

Neben dem “Ereignis”-Tab gibt es in der Sidebar einen Tab mit dem Namen “Variablen”. Dort kann man alle Variablen des aktuell ausgewählten Prozesses editieren. Mehr dazu aber später.

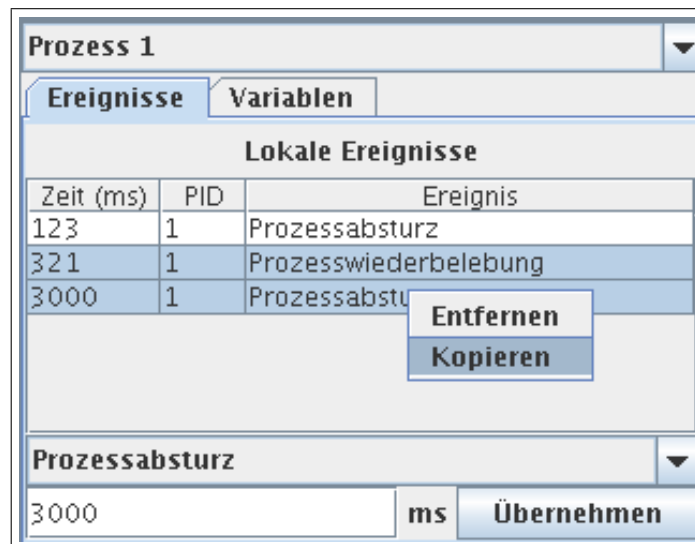


Abbildung 3.9.: Der Ereigniseditor mit 3 programmierten Ereignissen

3.1.5. Das Loggfenster

Das Loggfenster (Abbildung 3.3, unten) protokolliert in chronologischer Reihenfolge alle eingetroffenen Ereignisse. Auf Abbildung 3.10 sieht man das Loggfenster nach Erstellung unserer Simulation, an welcher 3 Prozesse beteiligt sind. Am Anfang eines Loggeintrages wird stets die globale Zeit in Millisekunden protokolliert. Bei jedem Prozess wird ebenso seine lokale Zeit sowie die Lamport- und die Vektor-Zeitstempel aufgeführt. Letztere werden später genauer behandelt.

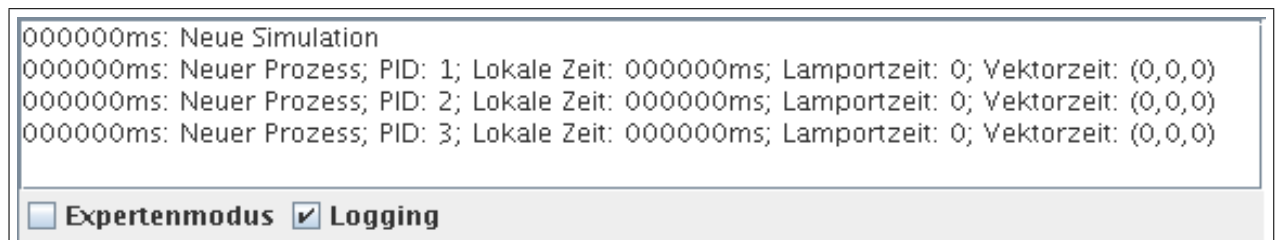


Abbildung 3.10.: Das Loggfenster

Mit dem Deaktivieren der Checkbox "Logging" läßt sich das direkte Loggen von Nachrichten temporär ausstellen. Ohne aktivierter Checkbox erscheinen keine neuen Nachrichten mehr im Loggfenster. Nach Reaktivieren der Checkbox werden alle ausgelassenen Nachrichten nachträglich in das Fenster geschrieben. Eine Deaktivierung des Loggings kann zu verbessertem Leistungsverhalten des Simulators führen (z.B. kein Ruckeln; ist vom verwendeten Computer, auf dem der Simulator läuft, abhängig).

Über die Checkbox “Expertenmodus” wird der Expertenmodus aktiviert beziehungsweise deaktiviert.

3.1.6. Der Expertenmodus

[illegible]

[illegible]

jf sldkfjsdlfkj lsdkfjds lfjds flsjfsljds flsdjf sldkfjsdlfkj lsdkfjds lfjds flsjfsljds flsdjf sldkfjsdlfkj
lsdkfjds lfjds flsjfsljds flsdjf sldkfjsdlfkj lsdkfjds lfjds flsjfsljds flsdjf sldkfjsdlfkj lsdkfjds lfjds
flsjfsljds flsdjf sldkfjsdlfkj lsdkfjds lfjds flsjfsljds flsdjf sldkfjsdlfkj lsdkfjds lfjds flsjfsljds flsd-
jf sldkfjsdlfkj lsdkfjds lfjds flsjfsljds flsdjf sldkfjsdlfkj lsdkfjds lfjds flsjfsljds flsdjf sldkfjsdlfkj
lsdkfjds lfjds flsjfsljds flsdjf sldkfjsdlfkj

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

3.3.8. Das Zwei-Phasen Commit Protokoll

3.3.9. Der ungenügende (Basic) Multicast²⁴

3.3.10. Der zuverlässige (Reliable) Multicast

3.4. Zeitformate

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>.