22. Dezember 2014

Steffen Giersch & Maria Lüdemann

BAI 5 Verteilte Systeme bei Herrn Professor Klauck

HAW Hamburg

Entwurf

Verteilte Systeme Praktikum 4

**Team:** 2 Steffen Giersch & Maria Lüdemann

**Aufgabenaufteilung:**

|  |  |
| --- | --- |
| Aufgabe | Teammitglied |
| Entwurfsplanung | Steffen Giersch |
| Entwurfsplanung | Maria Lüdemann |
| Implementation | Steffen Giersch |
| Implementation | Maria Lüdemann |

**Bearbeitungszeitraum:**

|  |  |  |
| --- | --- | --- |
| Datum | Dauer | Teammitglied |
| 15.12.2014 | 2 Stunden | Steffen & Maria |
| 22.12.2014 | 5,5 Stunden | Steffen & Maria |
| 21.11.2014 | 2 Stunden | Maria |
| 25.11.2014 | 6,5 Stunden | Steffen & Maria |
| 26.11.2014 | 7 Stunden | Steffen |
| 27.11.2014 | 2 Stunden | Steffen & Maria |
| 28.11.2014 | 1 Stunde | Maria |
| 29.11.2014 | 4,5 Stunden | Steffen & Maria |

**Gesamt: 29 Stunden**

**Quellen**: Beim Entwurf haben wir uns am Entwurf der Gruppe von Max Zender und Sven Freiberg orientiert.

**Aktueller Stand:**

* + - Entwurfsdokument ist in Arbeit
    - Implementation noch nicht begonnen

# Inhaltsverzeichnis

[1 Inhaltsverzeichnis 2](#_Toc407020781)

[2 Funktionalität 3](#_Toc407020782)

[3 Komponenten 3](#_Toc407020783)

[3.1 Station - station 3](#_Toc407020784)

[3.2 Zeit Synchronisation time\_Sync\_Manager 3](#_Toc407020785)

[3.3 Datenquelle – data\_source 3](#_Toc407020786)

[3.4 Datensenke – data\_sink 3](#_Toc407020787)

[4 Nachrichten 3](#_Toc407020788)

# Funktionalität

Es soll ein System implementiert werden dass ein Zeitmultiplexverfahren(TDMA) simuliert. Also die Möglichkeiten bietet mit mehreren Sendestationen über nur eine einzige Leitung Nachrichten zu senden. Dabei haben wir uns entschieden das System in Erlang zu implementieren da Erlang eine sehr gute Grundlage für Zeitkritische Systeme bietet.

# Komponenten

## Station – station

Die Station ist das eigentliche Herzstück des Systems, es beinhaltet alle Relevanten Komponenten für die jeweiligen Sende Stationen und kümmert sich um das Senden und Empfangen von Nachrichten, das Nachrichten zusammen bauen und die Stationsinterne Zeit. Wobei es zwei Arten von Stationen geben soll. Die Stationen des Typs A mit hinreichend genauer Zeit die ihre Zeiten mit denen der anderen Stationen vom Typ A synchronisieren und denen vom Typ B die eine nicht hinreichend genaue Zeit haben. Diese Stationen synchronisieren ihre Zeit mit denen der Stationen vom Typ A. Diese Komponente kümmert sich auch um die Konfiguration und das Starten der beinhalteten Module

Beinhaltet:

* Sender
* Reciever
* Slot\_manager

### Sender

Der Sender kümmert sich um das Versenden der Nachrichten. Er kümmert sich nicht darum wann er senden muss, also wann sein Slot innerhalb des Frames ist sondern wird vom Stationseigenen slot\_manager angewiesen wenn sein Sendeslot kurz bevor steht. Er prüft allerdings, ob er bei Benachrichtigung vom slot\_manager noch in der richtigen Zeit liegt und noch senden darf. Es soll so implementiert werden, dass der Sender überprüft ob er seinen Sendeslot verpasst hat, dies kann durch ungenaue Zeiten passieren, wenn er es verpasst hat dann sendet er in diesem Slot nicht.

### Reciever

Der Reciever erhält und verwertet die Nachrichten die über das System eingehen. Dabei stellt er auch Kollisionen fest und entscheidet wie damit zu verfahren ist. Dies macht er indem er für jeden Slot die Nachrichten sammelt und am Ende eines Slots überprüft ob mehr als eine Nachricht empfangen worden ist. Ist tatsächlich mehr als eine Nachricht eingegangen, wird davon ausgegangen dass es sich somit um eine Kollision handelt. Wenn nicht, liegt auch keine Kollision vor und es kann ganz normal fortgefahren werden.

**Verhalten bei Kollision:**

Wenn mehr als eine fremde Nachrichte entdeckt wird (nicht aus der eigenen Station),werden sie verworfen.

Wenn mehrere Nachrichten entdeckt werden von denen mindestens eine, eine eigene ist, werden alle Nachrichten verworfen und eine Nachricht gesendet um die geplante Slot Reservierung vom Slot\_Manager zurück setzten zu lassen. Dies wird getan weil mit der eigenen Nachricht ein neuer Slot hätte reserviert werden sollen, dies durch die Kollision allerdings nichtig wird. So muss beim nächsten Senden ein zufälliger Slot genutzt werden.

**Verhalten ohne Kollision:**

Der Slot\_Manager bekommt eine Nachricht durch die eine Reservierung gesetzt wird. Die Zeit aus der empfangenen Nachricht wird an die Time\_Master Komponente weiter gegeben sofern sie aus einer Station vom Typ A kommt und die Data\_Sink bekommt den Payload der Nachricht übermittelt.

### Slot\_manager

Der Slot\_Manager kümmert sich um die Definition, Handhabung und Vergabe der Slots innerhalb des Frames. Er ist die einzige Komponente mit einem Überblick über die einzelnen Slots und kümmert sich darum, wann ein Slot beginnt und endet sowie wann ein Frame beginnt und endet, um die Reservierung der Slots und welche Slots noch frei sind. Dementsprechend hält er sich eine Liste der freien Slots.

Der Slot\_Manager setzt immer wieder von neuem einen Slot\_Timer, immer wenn dieser abgelaufen ist wird der Time\_Master nach der aktuellen Zeit gefragt und der Reciever darüber benachrichtigt, dass ein Slot beendet wurde und setzt dann den Timer neu.

Immer wenn ein Frame vorbei ist wird der Time\_Master darüber benachrichtigt, danach erfragt er vom Time\_Master wie nach jedem Slot die aktuelle Zeit.

Für den Fall dass ein Slot reserviert wurde wird dem Sender dieser Slot übergeben. Wurde kein Slot reserviert wird ein beliebiger freier Slot aus dem letzten Frame genommen, dann aber im nächsten Frame, übergeben dabei wird darauf geachtet, dass wirklich ein zufälliger Slot gewählt wird und nicht etwa ein fest gescripteter z.B der erste freie genommen wird. Um zu vermeiden dass man gescriptete Kollisionen erzeugt.

Danach wird die Liste der freien Slots zurückgesetzt.

## Zeit Synchronisation time\_Time\_Master

Der Time\_Manager wird immer von Komponenten nach der aktuellen Zeit gefragt. Und gibt diese auf Anfrage zurück.

Er erhält außerdem vom Reciever die Zeiten aus den Nachrchten der Stationen vom Typ A.

Die dritte Nachricht die der Time\_Master erhält ist die Information darüber, dass das Frame abgelaufen ist. Beim erhalten dieser Nachricht, wird das arithmetische Mittel aus den erhaltenen Zeiten ermittelt und mit der Zeit verrechnet.

## Datenquelle – data\_source

Die Datenquelle liest durchgehend auf dem InputStream die 24 Bytes Nutzdaten ein und versendet die jeweils aktuellsten 24 Bytes auf Anfrage.

## Datensenke – data\_sink

Die Datensenke nimmt die 24 Bytes Nutzdaten vom Reciever entgegen und schreibt diese in eine Log-Datei weg.

# Nachrichten

Hier sollen einmal alle Nachrichten aufgezählt und definiert werden die eine Komponente erhalten kann.

## Station

### Sender

**Nachricht: {new\_timer, TimeToWait}**

**Absender:** Slot\_Manager

**Parameter:** TimeToWait -> Zeit die bis zum Slot gewartet werden muss

**Nachricht: {pay\_load, Data}**

**Absender:** Data\_source

**Parameter:** data -> 24 Bytes Nutzdaten

### Reciever

**Nachricht**: **{slot\_passed}**

**Absender:** Slot\_Manager

**Beschreibung**: Information darüber wenn der Slot abgelaufen ist

**Antwort: {no\_messages}**

**Beschreibung:** Keine Nachrichten erhalten

**Antwort: {slot\_reservation, Slot}**

**Parameter:** Slot -> Slot der reserviert wurde

**Beschreibung:** Slot erfolgreich reserviert

**Antwort: {collision\_detectet}**

**Beschreibung:** Eine Kollision wurde entdeckt, Reservierung verworfen

### Slot\_Manager

**Nachricht**: **{get\_reservable\_slot}**

**Absender:** Sender

**Beschreibung:** Ist die Frage nach einem reservierbarem Slot

**Antwort: {reservable\_slot, Slot}**

**Parameter:** Slot -> Slot der reserviert wurde

## Tim**e\_Master**

**Nachricht**: **{new\_message,StationTime, RecievedTime }**

**Parameter:** StationTime-> Zeit der Station vom Typ A

RecievedTime -> Zeitpunkt an dem die Nachricht eingegangen ist

**Absender:** Reciever

**Nachricht**: **{get\_current\_time, PID}**

**Parameter:** PID -> Prozess ID der abfragenden Komponente

**Absender:** Reciever, Slot\_Manager, Sender

**Antwort: {current\_time, Time}**

**Parameter:** Time -> aktuelle Zeit

**Nachricht**: **{sync}**

**Absender:** Slot\_Manager

**Beschreibung:** Triggert, dass der Time\_Master sich selbst, anhand den erhaltenen Zeiten aus dem letzten Frame mit den anderen Stationen synchronisiert

## Data\_Sink

**Nachricht**: **{data, Data}**

**Parameter:** Data -> 24 Bytes Nutzdaten die weggeschrieben werde

**Absender:** Reciever