13. JANUAR 2015

ENTWURF

VERTEILTE SYSTEME PRAKTIKUM 4

STEFFEN GIERSCH & MARIA LÜDEMANN
BAI 5 VERTEILTE SYSTEME BEI HERRN PROFESSOR KLAUCK
HAW Hamburg

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
03.01.2015	3 Stunden	Steffen & Maria
05.01.2015	6 Stunden	Steffen & Maria
06.01.2015	7,5 Stunden	Steffen & Maria
10.01.2015	6 Stunden	Steffen & Maria
11.01.2015	5 Stunden	Steffen & Maria
12.01.2015	5 Stunden	Steffen

Gesamt: 40 Stunden

Quellen: Beim Entwurf haben wir uns am Entwurf der Gruppe von Max Zender und Sven Freiberg orientiert. Wir haben uns mit der Gruppe unterhalten und in diesem Diskurs sind gemeinsame Meinungen unterstützt worden. Somit lehnen sich unsere Gedanken an die dieser Gruppe an.

Aktueller Stand:

- Entwurfsdokument ist in Arbeit
- Implementation abgeschlossen

1 INHALTSVERZEICHNIS

2 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.

3 KOMPONENTEN

3.1 STATION – PR_MASTER(PUBLIC RELATION MASTER)

Der PR_Master 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

3.1.1 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.

3.1.2 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.

3.1.3 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.

3.2 Zeit Synchronisation Time_Master

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

Er erhält außerdem vom Reciever die Zeiten aus den Nachrichten 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.

3.3 Datenquelle – data_source

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

3.4 Datensenke – data_sink

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

4 Nachrichten

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

4.1 PR_Master

4.1.1 Sender

Nachricht: **{new_timer, TimeToWait, ReservedSendIntervall}**

Absender: Slot_Manager

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

ReservedSendIntervall -> Zeitintervall in dem gesendet werden muss

Nachricht: **{payload, Data}**

Absender: Data_source

Parameter: data -> 24 Bytes Nutzdaten

4.1.2 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_detected}**

Beschreibung: Eine Kollision wurde entdeckt, Reservierung verworfen

4.1.3 Slot_Manager

Nachricht: **{get_reservable_slot, PID}**

Parameter: PID -> die Prozess Id des Senders

Absender: Sender

Beschreibung: Ist die Frage nach einem reservierbarem Slot

Antwort: **{reservable_slot, Slot}**

Parameter: Slot -> Slot der reserviert wurde

Nachricht: **{slot_missed}**

Absender: Sender

Beschreibung: Wurde der Slot verpasst gibt der Sender dem Slot_Manager darüber bescheid

4.2 Time_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

4.3 DATA_SINK

Nachricht: **{data, Data}**

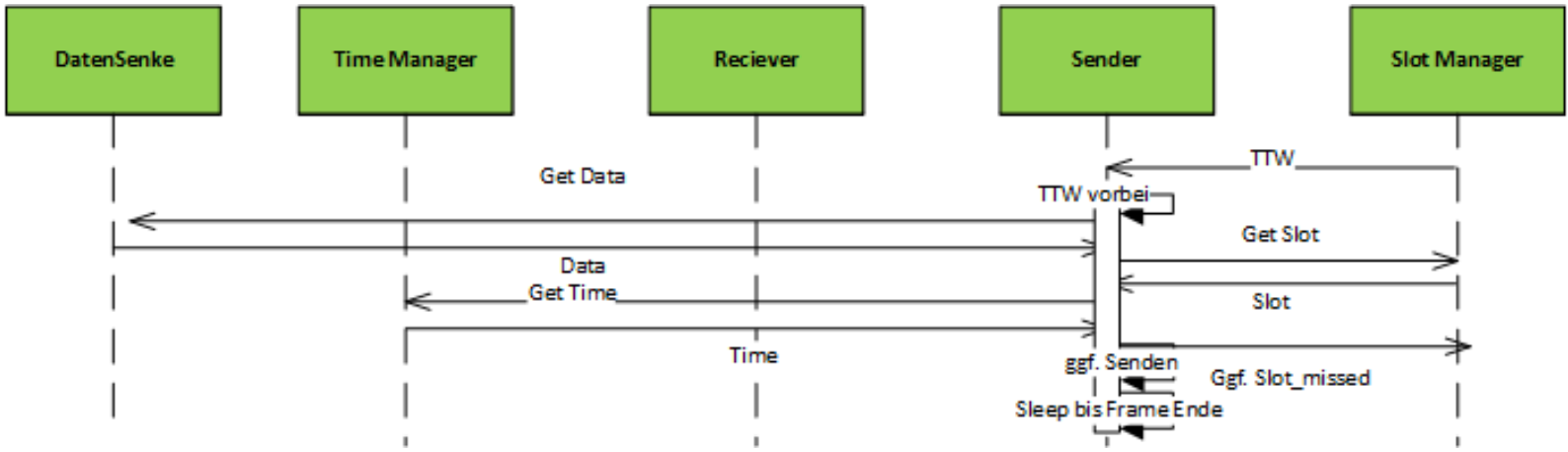
Parameter: Data -> 24 Bytes Nutzdaten die weggeschrieben werden

Absender: Reciever

5 DIAGRAMME

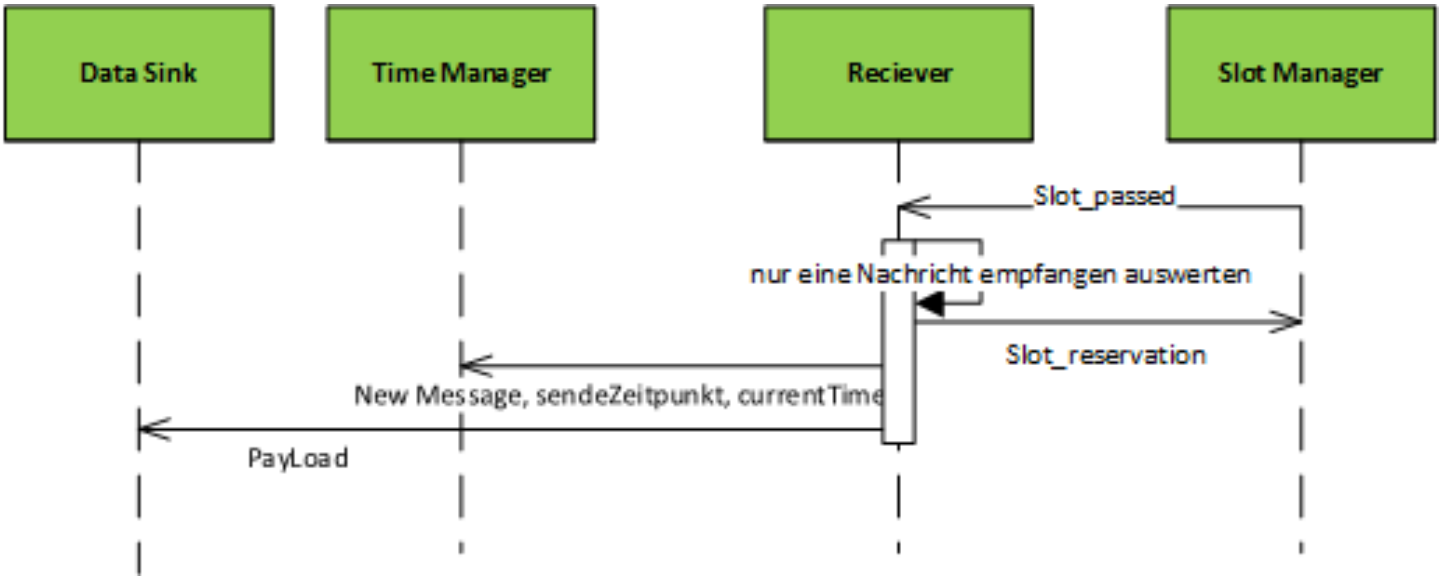
5.1 SENDER VOR DEM SLOT

Zur Verdeutlichung des Abblaufs soll das folgende Abblaufdiagramm dienen. Es stellt die Kommunikation dar die um den Sender herum entsteht kurz bevor eine Nachricht „gefunkt“ werden soll.



Zu beachten hier ist, dass der Sender, der Reciever und der Slot_Manager zusammen gefasst die PR_Master Komponente ergeben hier aber genauso dargestellt werden wie Time Manager und die Daten Senke die jeweils eigene Komponenten sind. Am Anfang eines Frames schickt der Slot_Manager dem Sender seine TTW(Time To Wait) also die Zeit die er innerhalb des Frames warten muss bevor sein Slot beginnt. Wenn diese Zeit abgelaufen ist wacht der Sender auf und beginnt alle Daten zu sammeln die er für die Nachricht braucht. Als erstes holt er sich die Daten von der Daten Senke und fragt einen freien Slot für das nächste Frame beim Slot Manager ab. Wenn er diese Daten hat, baut er die Nachricht zusammen und fragt die aktuelle Zeit beim Time Manager ab um zu überprüfen dass er seinen Slot noch nicht verpasst hat. Dann wird der Zeit Stempel an die Nachricht gehängt und versendet sofern der Slot nicht verpasst wurde. Wurde der Slot verpasst wird das Paket verworfen und der Sender legt sich bis zum Ende des Frames schlafen bis der Slot Manager ihn wieder durch eine neue TTW weckt. Sofern der Sender im ersten Slot senden soll, wird eine TTW von 0 gesendet. Es muss dann davon ausgegangen werden, dass er in der Lage ist das Paket innerhalb des Slots zusammen zu bauen und noch zu senden. Wenn nicht verpasst er seinen Slot und sendet nicht.

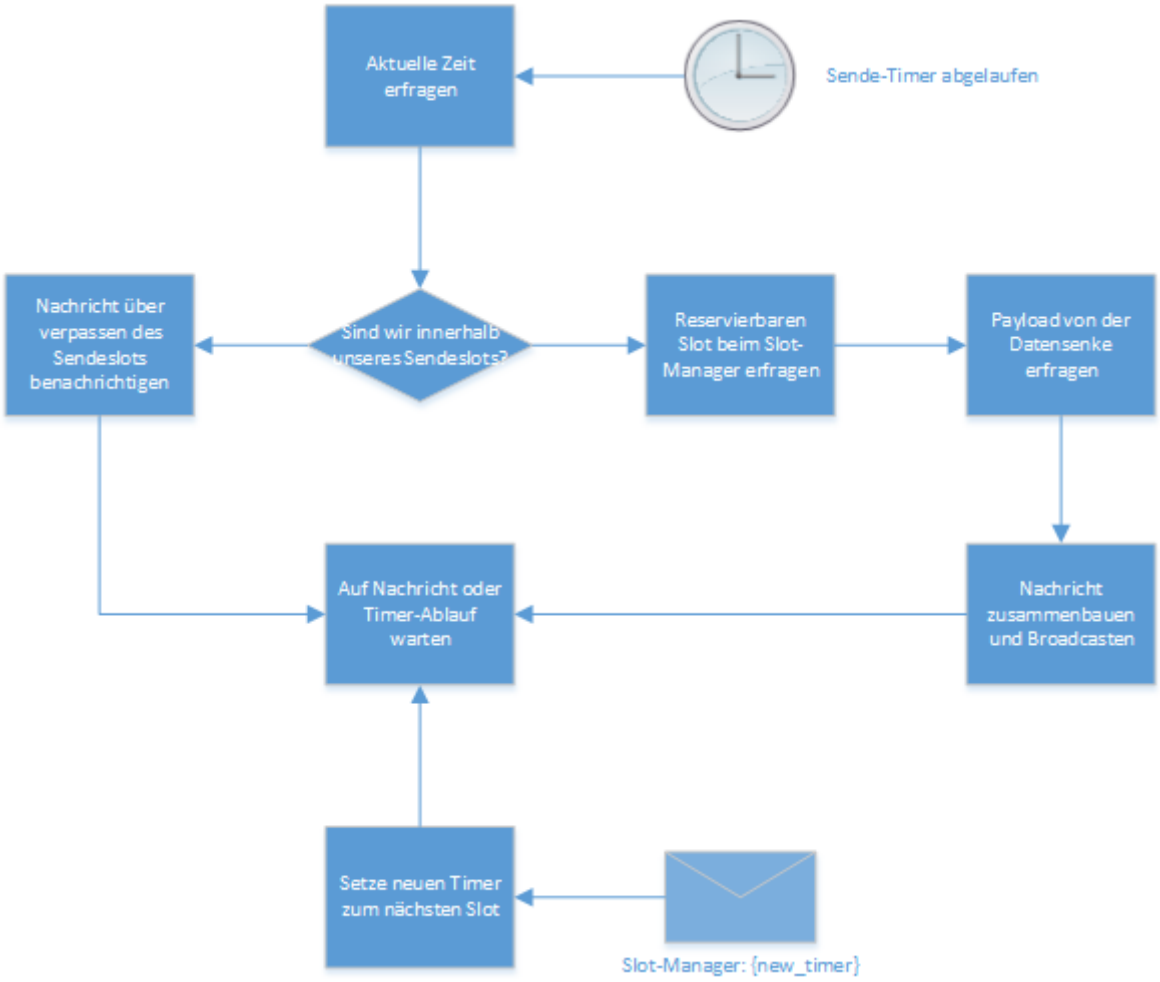
5.2 Reciever Slot Ende eine Nachricht erhalten



Wenn nach einem Slot nur eine Nachricht beim Reciever eingegangen ist bildet die obige Abbildung ab wie die Kommunikation zwischen den einzelnen Komponenten aussieht. Kommt die Nachricht vom Slot Manager, dass ein Slot um ist schaut der Reciever nach wie viele Nachrichten eingegangen sind. War es nur eine einzige sendet er dem Slot_Manager dass es eine Slot Reservierung gab und sendet dem Time Manager die Nachricht, dass es eine neue Nachricht gab und wenn es eine A Station war wird die Zeit aus der Nachricht mitgesendet sowie der Zeitstempel zu dem die Nachricht im Reciever angekommen ist. Den Zeitstempel bekommt jede Nachricht in dem Moment wo der Reciever sie empfängt. Dann fragt er den Time Manager nach der aktuellen Zeit und merkt sich diese bis zu dem Punkt, dass der Slot rum ist und ggf. die Nachricht an den Time Manager gesendet wird. Außerdem bekommt die Daten Senke den Pay Load der Nachricht zu gesendet.

5.3 Sender Flussdiagramm

Das unten dargestellte Diagramm verdeutlicht die Kommunikation des Senders noch einmal als Flussdiagramm um auch die getroffenen Entscheidungen etwas klarer zu zeigen.



5.4 Slot Manager Flussdiagramm

Siehe Anhang 1

Das Flussdiagramm stellt dar welche Nachrichten der Slot Manager bekommt und versendet und welche Entscheidungen er dabei trifft.

6 Änderungen am Entwurf

Bei der Implementierung des Entwurfs ist uns aufgefallen, dass es nicht reicht einen Receiver Prozess zu haben. Es ist notwendig die Nachrichten die auf Erlang Ebene gesendet und empfangen werden von der eigentlichen UDP Schicht zu trennen die den „Funk“ imitiert. Somit haben wir den Receiver dann zwei geteilt einmal in eine receiver.erl die sich um die Verbindung der einzelnen Prozesse und die Interprozesskommunikation kümmert und einen udp_receiver.erl der sich um die UDP Verbindung kümmert.

Die Nachricht an den Sender vom SlotManager die ihm an Anfang eines Slots sagt wie lange er warten muss sollte auch übermitteln bis wann und ab wann gesendet worden sein muss damit der Sender überprüfen kann ob er seinen Slot verpasst hat. Außerdem soll er nicht bis direkt zu seinem Slot warten, sondern ein wenig in ihn hinein, da das Senden und Empfangen der erforderlichen Anfragen und Daten im Regelfall schnell genug von statten geht um in der Regel den Sendeslot nicht zu verpassen aber verhindert dadurch, dass zu früh gesendet werden könnte. Uns war es wichtig, darauf zu achten, dass lieber nicht gesendet wird und damit Kollisionen eher vermieden werden.

Wir haben uns dazu entschieden Slots nicht direkt am Anfang des Slots zu erfassen sondern bei der Berechnung wenige ms darauf zu rechnen um im Falle von Ungenauigkeiten nicht die Gefahr besteht, dass wir vor dem Slot landen sondern einfach dichter an den dessen Anfang rutschen.