VSP4 Entwurf

Verteilte Systeme Praktikum 4 - Gruppe 1 - Team 10

Matthias Nitsche und Swaneet Sahoo

Inhaltsverzeichnis

Dokumentationskopf

Team: 10, Matthias Nitsche (MN), Swaneet Sahoo (SS)

Aufgabenaufteilung:

Entwurf (MN, SS)

Implementation (MN, SS)

Bearbeitungszeitraum:

13.06.2015: Entwurf 5h

17.06.2015: Entwurf 6.5h

18.06.2015: Implementation 5h

20.06.2015: Implementation 7.5h

23.06.2015: Implementation 9h

24.06.2015: Testing 2h

Total: 5 + 6.5 + 5 + 7.5 + 9 + 2 = 35h

Aktueller Stand:

Entwurf fertig

Änderungen im Entwurf:

Sender

- Das Abholen eines potentiell reservierbaren Slots vom SlotBroker(getNextFrameSlotNr) in Abb.2 wurde zu Ende des jeweiligen Frames verlagert.
- NextSendNr existert nicht mehr.
- Die beim Senden einer Message reservierte Nummer ist nicht mehr NextSendNr, sondern eine freie Nummer die kurz zuvor vom Broker abgefragt wird.

Slot Broker

- CurrentSlots wird am Anfang jedes Frames nicht mehr auf dem vorherigen Wert von NextSlots gesetzt. Beide werden nun pro Frame auf [1..25] resetted. CurrentSlots repräsnetiert die Slots im aktuellen Frame, in dennen physikalisch keine Pakete versandt wurden. NextSlots repräsentiert die bereits für das nächste Frame reservierten Slots.
- getNextFrameSlotNr kann nun jederzeit abgefragt werden.

Überblick

Aufgabenstellung

In der folgenden Aufgabe wird ein Time Division Multiple Access (TDMA) Verfahren vorgestellt und spezifiziert. Insbesondere ist zu beachten das dies ein zeitkritisches Verfahren ist welches über einen eigenen Uhrensynchronisationmechanismus verfügt. Im Folgenden werden wichtige Komponenten und Abläufe beschrieben, wie ein solches System implementiert werden könnte.

Self-organized Time Division Multiple Access (STDMA)

TDMA ist ein Verfahren das Kanäle in zeitliche Frames unterteilt, die wiederum in Slots eingeteilt werden. Die Idee ist es, das Akteure im System Zeitslots reservieren können bei der sie als einzige das Recht der Übertragung haben und Kollisionen zu minimieren. Anders als bei TDMA, wo diese Kommunikation über eine zentrale Instanz läuft wird bei STDMA, ein Akteur im System, zu einem Sender und Empfänger von Daten und muss sich mit den anderen Akteuren Synchron halten.

Komponenten

Clock

Die Clock ist unsere Uhr, die Komponenten übergreifend, alle Funktionen anbietet, um eine einheitliche Zeit in UTC zu gewährleisten. Sie hilft um von primitiven Operationen zu abstrahieren und den Code DRY zu halten. Zusätzlich wird ein Offset Anfangs bestimmt den die Uhr bekommt, mit dem wir die zeitlichen Schwankungen zwischen unterschiedlichen Stationen persistieren. “Es werden die Zeitstempel der Typ A Stationen (abzüglich/zuzüglich der Slotzeit im Frame) per arithmetischem Mittel ausgewertet und bestimmt die eigene Abweichung (eigene Uhr: Sytemzeit +/- Abweichung).” ([3] Letzte Folie “Tipps”)

Slot Broker

Der Slot Broker übernimmt jede Verantwortung für die korrekte Reservierung von Slots. Es muss möglich sein zu fragen ob Slots schon reserviert wurden (Kollisionserkennung). Auf Anfrage durch z.B. getNextSlotNr() wird eine zufällige nicht reservierte Slot Nummer gewählt und an den Anfrager versendet. Der Slot Broker muss sowohl eine Übersicht über den aktuellen Frame k und dessen Slots, als auch über den nächsten Frame k+1 haben um eine korrekte Reservierung durchführen zu können.

Receiver

Der Receiver erhält Pakete von Sendern per IP-Multicast und wertet diese aus. Ferner stößt der Receiver die Uhrensynchronisation an wenn die Pakete von einer Station A kommen. Der Inhalt der Pakete wird letztendlich in die Datensenke weitergeleitet. Der Receiver kümmert sich darum die Art der Pakete (z.B. von Station A oder B) und die Timestamps zu ermitteln. Jedes eintreffende Paket A wird verwendet um die Uhr korrekt zu aktuallisieren. Der Receiver konsultiert allerdings zuallererst den Slot Broker um die Verarbeitbarkeit dieser Nachricht zu gewährleisten.

Sender

Der Sender hat die Aufgabe, Pakete aus der Datenquelle korrekt per IP-Multicast an alle beteiligten Receiver zu versenden. Er muss sich, ähnlich wie der Slot Broker, merken, welches sein Slot für den aktuellen Frame k und k+1 ist. Er ist außerdem verantwortlich dafür sich seine aktuellen Slot Nummern (rechtzeitig!) vom Slot Broker zu besorgen.

Initialisierungphase

Für die Intialisierung des Systems wird ein bereitgestelltes Script “startStations.sh” verwendet. Dieses muss für jeden neuen Akteur gestartet und konfiguriert werden. Die Anzahl der A und B Stationen kann konfiguriert werden sodass im Grunde zwei Aufrufe von “startStations.sh” ausreichen. Danach ist das System bereit und geht in die Arbeitsphase über.

Arbeitsphase

In der Arbeitsphase ist STDMA und die im Hintergrund laufende Uhrensynchronisation im vollen Gange. Im Folgenden wird der Ablauf und die Kommunkation zwischen den Komponenten während der Uhrensynchronisation und dem Empfangen, Verarbeiten und Versenden von Nachrichten erläutert. Die Kommunikation mit der Clock-Komponente wird nur in der Uhrensynchronisation explizit dargestellt.

Uhrensynchronisation

Die Uhrensynchronisation läuft immer im Hintergrund. Die Abweichung vom UTC ist im Clock gespeichert. Pro Frame wird der arithmetische Mittel aller Abweichungen der eigenen Empfangszeit *now* von den Sendezeiten *ts* von Nachrichten des Stationsstyps A verwendet um die eigene Abweichung *offset* anzupassen. Bei der Berechnung wird derzeit die Übertragungsdauer von Nachrichten nicht berücksichtigt. Der Receiver schickt vor der Uhrensynchronisation die Nachrichten zunächst an den Slot Broker der anhand der Timestamps und der Slotarithmetik entscheidet, ob die Nachricht überhaupt verarbeitet werden darf (falls sie z.B. mit einer vorherigen Nachricht kollidiert).

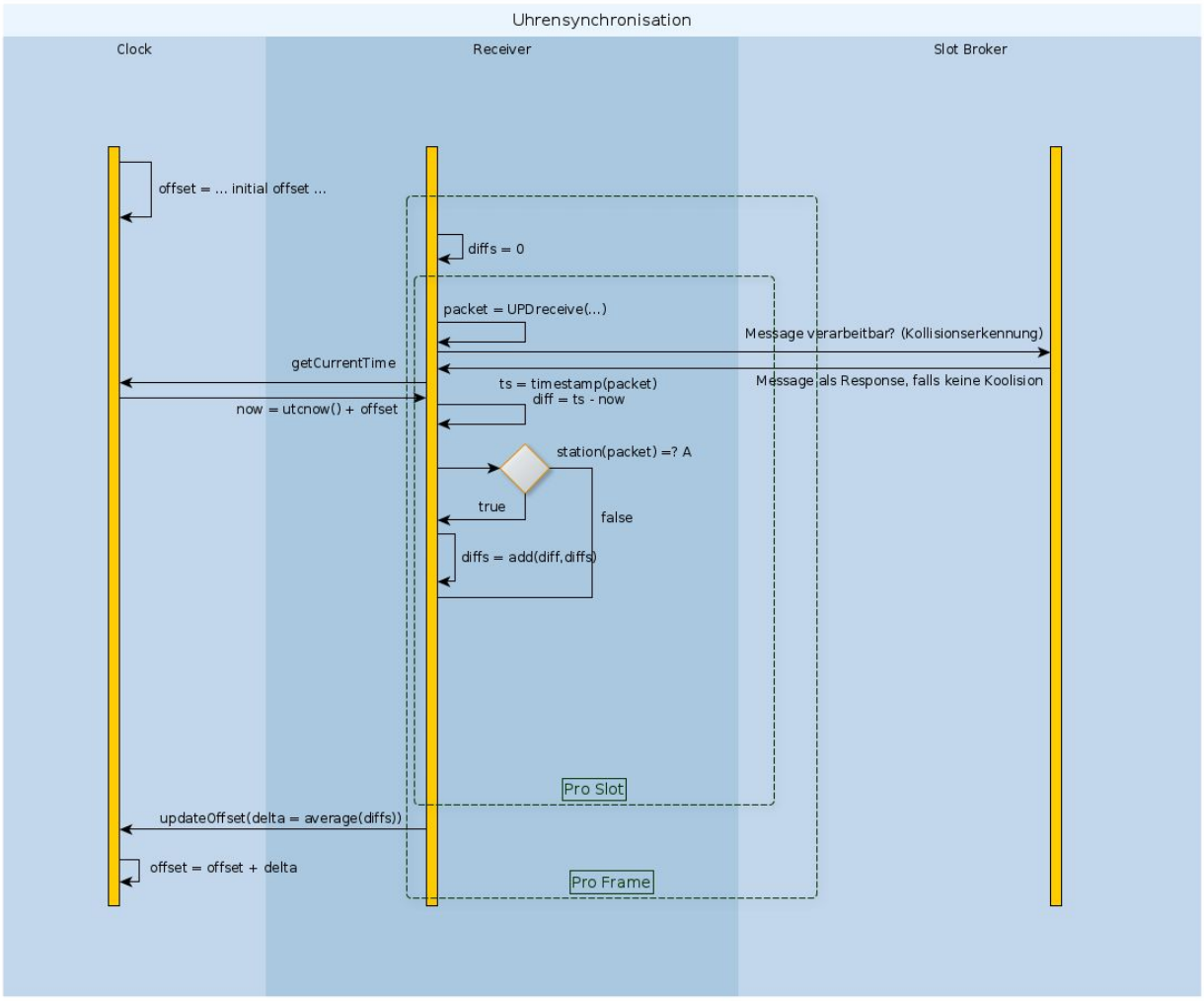

Abbildung 1: Uhrensynchronisation

Sender

Im Folgenden wird die Kommunikation zwischen dem Sender und dem Slot Broker beschrieben.

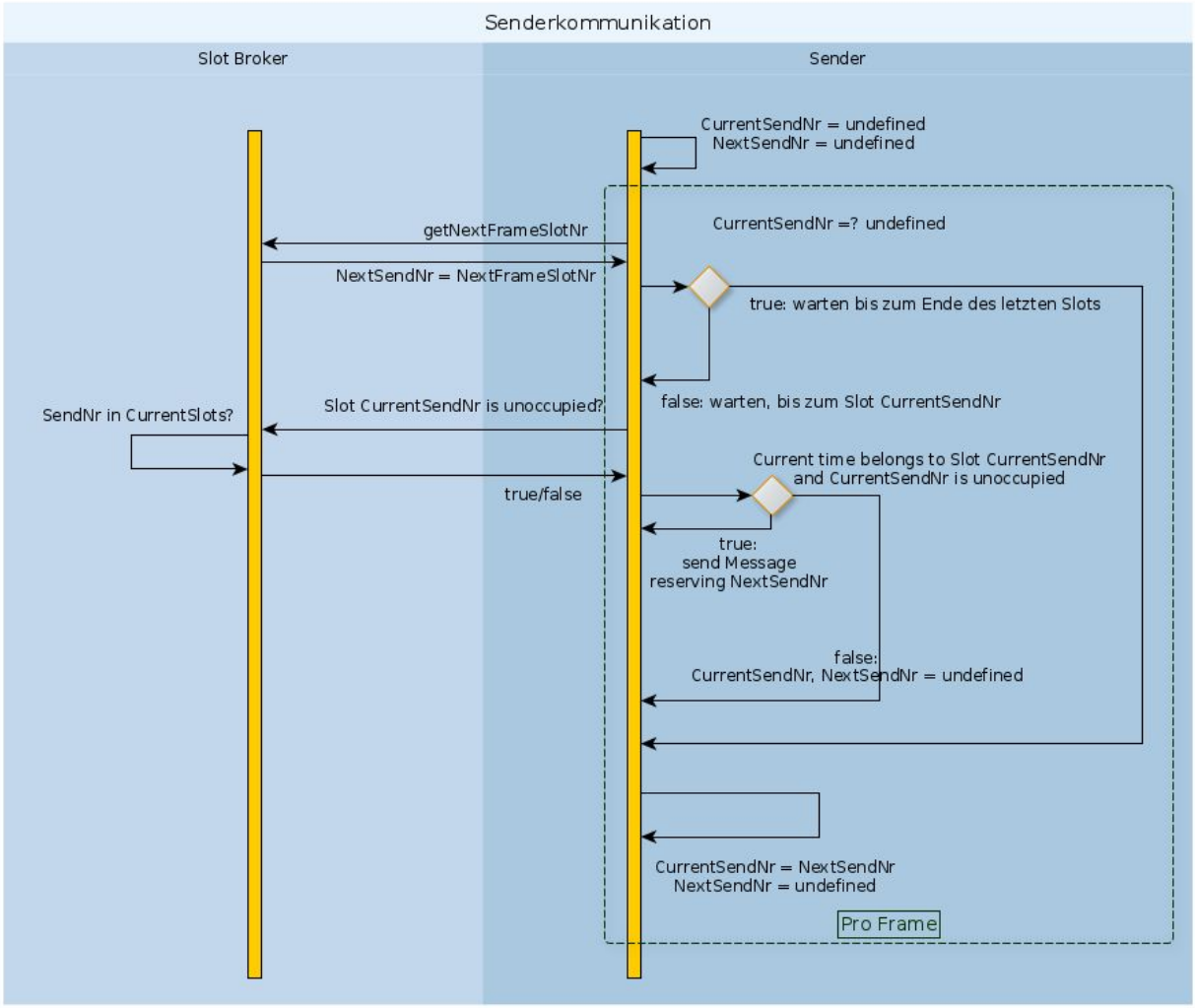

Abbildung 2: Senderkommunikation

Beschreibungen zu dem Diagramm:

- **CurrentSendNr**: Der vorher reservierte/ausgewählte Slot in dem im aktuellen Frame gesendet werden soll. Ist zu Beginn nicht definiert und wird bei jedem neuen Frame auf den Wert von NextSendNr gesetzt.
- **NextSendNr**: Die in dem nächsten Frame zu reservierende Nummer. Diese Nummer wird in die gesendete Nachricht reingeschrieben.
- In der ersten Iteration sind beide Variablen unbelegt. Es wird eine *NextSendNr* vom Slot Broker angefragt und der Frame wird abgewartet. Der Sender wird versuchen in der nächsten Iteration auf der *NextSendNr* zu senden.

- Bei der Abfrage “Slot *CurrentSendNr* unoccupied?” wird abgefragt, ob in dem zu sendenden Slot bereits eine Nachricht empfangen wurde. War dies so, dann wird im Slot Broker dieser Slot nicht mehr in den *CurrentSlots* frei sein. Dann wird keine Nachricht mehr gesendet (Kollisionsvermeidung). Falls noch nichts gesendet worden ist und wir noch zeitlich im auch korrekten Slot sind, wird die Nachricht gesendet.
- Sonst werden die Variablen zurückgesetzt und der Sender verhält sich so wie ab der ersten Iteration.

Slot Broker

Der Slot Broker hat wie in den Komponenten beschrieben die Aufgabe die Verteilung und Reservierung von Slots zu machen.

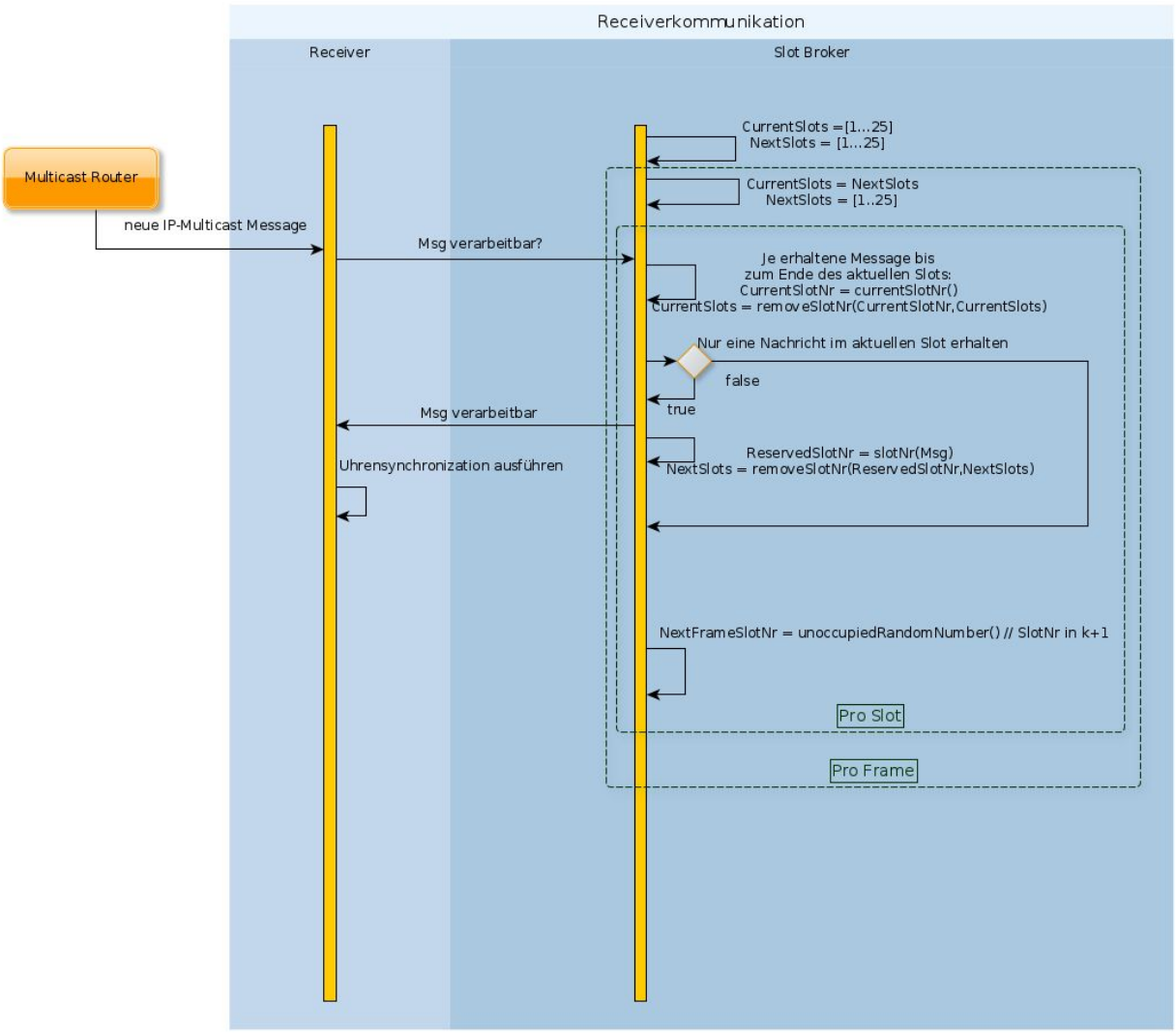

Abbildung 3: Slot-Broker-Kommunikation

Beschreibung zu Abbildung 3:

- **CurrentSlots**: Die vorher noch nicht reservierten/ausgewählten Slots, die jetzt, im aktuellen Frame noch frei sind. Initial wird CurrentSlots auf eine Liste mit Zahlen von [1..25] gelegt. Sobald das System initialisiert wurde wird CurrentSlots immer auf das alte NextSlots gesetzt.
- **NextSlots**: Die Slots die im nächsten Frame noch nicht benutzt werden. Diese Nummern können daher noch vergeben werden, wird wie CurrentSlots initial auf eine Liste von 1 bis 25 gesetzt.
- **NextFrameSlotNr**: Eine zufällige Nummer von 1 bis 25 die für einen Sender reserviert werden kann.

Sobald der Receiver per IP-Multicast ein Paket erhält, validiert der Slot Broker ob die gegebene SlotNr noch nicht besetzt ist. Falls nicht, kann das Paket weiter verarbeitet werden und der Slot im aktuellen Frame als versendet gesehen werden.

Alle Nachrichten in dem aktuellen Slot werden vom Receiver an den Slot Broker weitergeleitet. Die aktuelle CurrentSlotNr wird dann aus den CurrentSlots entfernt, so dass eine neue Nachricht die Eintrift, zum Ignorieren aller Nachrichten in diesem Slot führt. Falls mehr als eine Nachricht im aktuellen Slot gekommen ist (was einer Kollision gleich kommt) reservieren wir die Slots für die nächste Runde nicht und ignorieren alles was gesendet wurde. Falls jedoch genau ein Paket ankommt im aktuellen Slot können wir die SlotNr für die nächste Runde reservieren. In diesem Fall wird an den Receiver die Rückmeldung gegeben das die Message valide ist.
Der Slot Broker generiert für den Sender in jedem Slot eine neue NextFrameSlotNr.

Literatur

- *[1] Aufgabenstellung von Prof. Klauck und Herr Schulz http://users.informatik.haw-hamburg.de/~scotty/pub/Verteilte-Systeme/AI5-VSP/Aufgabe4/VSP_Aufgabe4.pdf*
- *[2] Werkzeuge und Datensenke von Prof. Klauck und Herr Schulz http://users.informatik.haw-hamburg.de/~scotty/pub/Verteilte-Systeme/AI5-VSP/Aufgabe4/*
- *[3] Aufgabenpräsentation von Prof. Klauck http://users.informatik.haw-hamburg.de/~klauck/VerteilteSysteme/Aufgaben.pdf*