Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Verteilte Systeme
Aufgabe 3

Konstantin Dolberg
Maximilian Schmidt

Fakultät Technik und Informatik
Department Informatik

Inhaltsverzeichnis

1 Kommunikation

Für die Kommunikation wird das UDP Protokoll verwendet, da es nicht zu einem Verbindungsaufbau zwischen den einzelnen Host kommt, sondern über Multicast kommuniziert wird.

1.1 Nachrichtenformat

Für die Nachrichtenpakete ist einheitlich nachfolgendes Format zu verwenden:
Byte 0: Stationsklasse ('A' oder 'B')
Byte 1 – 24: Nutzdaten. (Darin Byte 1 – 10: Name der sendenden Station.)
Byte 25: Nummer des Slots, in dem die Station im nächsten Frame senden wird.
Byte 26 – 33: Zeitpunkt, zu dem dieses Paket gesendet wurde. Einheit: Millisekunden seit dem 1.1.1970 als 8-Byte Integer, Big Endian.

Gesamtlänge: 34 Byte

1.2 Multicast Adresse und Port

Multicast-Adresse und Port müssen bei Initialisierung als Parameter mitgegeben werden oder als Default-Wert mit diesen Werten gesetzt werden:
Adresse: 225.10.1.2
Empfangsport: 15007

1.3 Frames und Slots

Es wird versucht jeweils von jedem Sender in jedem Frame einen Slot zu verwenden um zu Kommunizieren. Ein Frame geht dabei genau eine Sekunde und startet immer zur vollen Sekunde. Ein Frame besteht aus 25 Slots, die also 40 ms lang sind.

2 Datenquelle

Die Datenquelle ist die Erzeugungskomponente der Nutzdaten. Sie erzeugt ein 24Byte Nutzdatenpakete(Byte 0-9: Stationsname, Rest: Daten). Die Datenquelle ist nicht zu implementieren, sondern herunterzuladen.

3 Sender

Der Sender empfängt Daten aus dem Sendbuffer und sendet sie per UDP-Multicast an die Empfänger.

3.1 Ablauf

Der Sender holt sich aus dem SendBuffer die Nutzdaten und sendet eine Anfrage an den Empfänger nach einer Slot Nummer sowie einer Zeitangabe(Systemzeit mit Offset[Long]). Aus der Zeitangabe muss ein Offset gebildet werden und mit diesem der Sendezeitpunkt berechnet werden. Die Slot Nummer muss in einem Bereich von 1-25 liegen, um gültig zu sein. Bei einer -1 ist keine Slot Nummer mehr frei. Danach erfolgt eine Kollisionsabfrage beim Empfänger und bei negativem Ergebnis wird gesendet, falls der Slot nicht schon vorbei ist. Bei einem positiven Ergebnis wird ein Logeintrag veranlasst und sich zusätzlich eine neue Slot Nummer für den nächsten Frame besorgt. Wenn der Wert -1 empfangen wird, ist der früher reservierte Slot gültig.

4 DatasourceBuffer

Der DatasourceBuffer ist dafür zuständig, um die 24-Byte Nutzdaten vorrätig zu halten. Er erhält die jeweiligen Nutzdaten von der Datenquelle und liefert sie nach Abfrage vom Sender an diesen aus. Versuche von STDIN zu lesen sind bisher fehlgeschlagen.

5 Empfänger

Der Empfänger lauscht auf dem Nachrichtenkanal, um die Kommunikation zu verfolgen und dem Sender die Informationen zur Verfügung zu stellen.

5.1 Initialisierung

Bei der Initialisierung beobachtet der Empfänger zunächst einen kompletten Frame. Dabei wird herausgefunden welcher Slot im nächsten Frame frei ist, außerdem wird die Uhr synchronisiert. Mit diesen Informationen wird dann der Senderprozess gestartet.
Die Datensenke und der Sendbuffer werden sofort mit dem Empfänger initialisiert.

5.2 Schnittstelle

- int getFreeSlotNextFrame(): Gibt einen bisher noch nicht belegten Slot im nächsten Frame
- long getTime(): Gibt die aktuelle Uhrzeit
- boolean isCollision(): Überprüft ob ein Senden zu eine Kollision führen würde. True wenn dem so ist
- getSlotForCollusion(): Wird vom Sender am Ende eines Frames aufgerufen. Wenn der Sender eine Kollision erzeugt hat, gibt diese Funktion einen noch nicht reservierten Frame zurück, ansonsten wird -1 zurückgegeben. Das bedeutet für den Sender, dass der Frame den er beim Senden über getFreeSlotNextFrame() bekommen hat, der gültige Frame ist.

5.3 Uhrensynchronisation

Der Empfänger bekommt bei seiner Initialisierung mitgeteilt, welcher Typ (A oder B) er ist. Je nachdem wird dann die Uhr mit jeder eintreffenden Nachricht synchronisiert (siehe Zeitsynchronisation).

5.4 Slotverarbeitung

Die Slotverarbeitung wird immer am Ende eines Slots gemacht, da erst dann sichergestellt werden kann ob Kollisionen aufgetreten sind oder nicht. Sind keine Kollisionen aufgetreten, werden die Informationen verarbeitet, ansonsten verworfen.

5.5 Kollisionserkennung

Eine Kollision ist aufgetreten, sobald innerhalb eines Slots zwei oder mehr Nachrichten eingetroffen sind. Im eigenen Senderslot reicht bereits eine eingegangen Nachricht um eine Kollision vorauszusehen. Dann muss der Sender benachrichtigt werden, keine Nachricht zu senden (siehe Sender Laufzeitsicht).

6 Datensenke

Die Datensenke ist für die Entgegennahme der Nutzdaten und das Logging zuständig.

6.1 Initialisierung

Die Datensenke wird mit dem Start des Programms initialisiert und ist für keine weitere Komponenteninitialisierung zuständig.

6.2 Datenausgabe

Das Logging und Ausgabe der Nutzdaten findet über den Stdout Kanal statt. Folgende Ereignisse sind zu Loggen:

- Erkannte Kollisionen
    - Zusätzlich vermerk wenn selbst beteiligt
- Anzahl der Frames in denen gesendet wurde
- Anzahl der Frames in denen nicht gesendet wurde
- Eingegangene und Ausgegangene Nutzdaten