Team: 1, The Kiet Dang, Sofian Wüsthoff

Aufgabenaufteilung:

1. The Kiet Dang,
   Vorbereitung: Klasse Diagramm, Zeitsynchronisation, IP Multicast
   Implementierung:
   + Datei aus Datenquelle Empfängen
   + Zeitsynchronisation
   + IP Multicast Empfängen
2. Sofian Wüsthoff,
   Vorbereitung: STDMA, Kollision, Requirements
   Implementierung:
   + Slot verteilen
   + Kollision behandeln
   + IP Multicast Senden

Quellenangaben:

1: Verteilte Systeme Vorlesungsfolien

2: Broadcasting and Multicasting in Java
https://www.baeldung.com/java-broadcast-multicast

3: javadoc

4: Stackoverflow

5: Understanding STDMA
https://link.springer.com/article/10.1186/s13638-016-0680-7

Bearbeitungszeitraum:

20.11.2019 5 Stunden

25.11.2019 4 Stunden

30.11.2019 8 Stunden

Aktueller Stand: Der Entwurf ist ausführlich und fertig. Wir könnten die Datei aus der Datenquelle übernehmen und auf unsere Nachrichten packen. Dazu haben wir auch schon das Kommunizieren zwischen den Stationen im IP-Multicast Net6z. Wir versuchen gerade die Zeitsynchronisation zu implementieren sowie die Kollision zu behandeln.

Änderungen des Entwurfs: noch keine Änderungen

Senden und Empfangen mit Zeitmultiplexverfahren – Systementwurf

Aufgabenstellung

Erstellen einer Sende-/Empfangsstation, welche Zeitmultiplexverfahren nutzt, um Daten zu Senden und Empfangen. Dabei wird nur ein Kanal benötigt, welcher von mehreren Stationen zeitlich gestaffelt zum Senden benutzt werden kann und sonst jeder Zeit empfangsbereit sind. Die Stationen sind in Form von n Slots eines Frames, welches die Zeitachse repräsentiert, unterteilt. Für die Anwendung soll des weiteren IP-Multicast verwendet werden.

Requirements

1. Ein Kanal über den gesendet und empfangen wird
2. Kommunikation über IP-Multicast
3. Kanal besteht aus gleichgroßen Frames
4. Frames dauern eine Sekunde
5. Frames haben 25 Slots
6. Betrieb soll mit bis zu 25 Slots laufen
7. Vergabe der Slots über STDMA – Verfahren
8. Kollisionen in der Auswahl der Slots vermeiden
9. Klasse A mit hinreichend genau betrachteter Uhr
10. Klasse B mit nicht hinreichend genau betrachteter Uhr
11. Zeiten basieren auf UTC
12. Uhren von Klasse A synchronisieren sich untereinander
13. Uhren von Klasse B synchronisieren sich mit denen von Klasse A
14. Pakete mit Länge von 34 Bytes
15. Paketaufbau: Byte 0 : Stationsklasse, Byte 1 -24: Nutzdaten, Byte 25: Nummer des Slots, in dem die Station im nächsten Frame senden wird, Byte 26 – 33: Zeitpunkt, zu dem das Paket gesendet wurde in Millisekunden

Systementwurf

Jede Station hat die Möglichkeit, durch IP-Multicast, Daten zu senden und zu empfangen. Sende -und Empfangen-Methoden von Stationen werden parallel durch einen Empfänger-Thread und einen Sender-Thread ausgeführt.

Die Stationen sollen auch zwei Puffer haben. Der erste Puffer, der „inputBuffer“, wo der dritte Thread die Datei aus der Datenquelle entgegennimmt und puffert. Der andere Puffer, welcher zuständig für einkommende Nachrichten anderer Stationen ist, ist der „messageBuffer“.

Der Empfänger-Thread reagiert jeder Zeit auf eintretende Nachrichten, während der Sender-Thread genau einmal pro Frame senden soll.

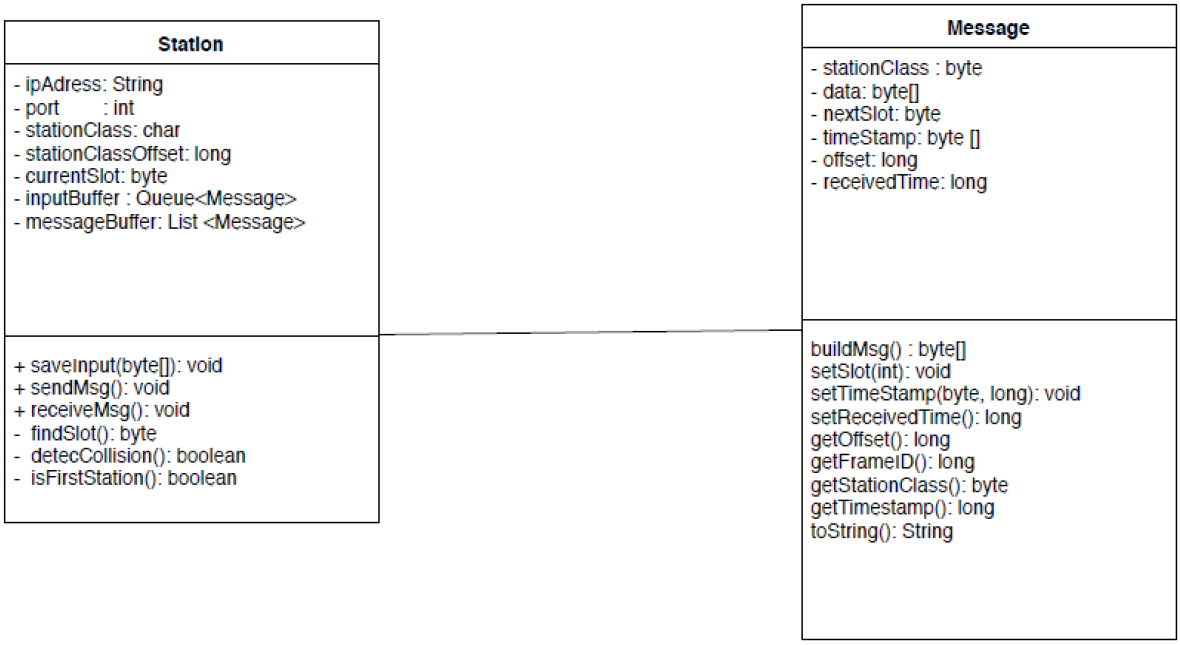
Die Zeitsynchronisation wird im Empfänger-Thread geschehen. Wenn eine empfangende Station von der Klasse B ist, vergleichen wir den aktuellen Zeitpunkt dieser Station mit dem Sendezeitpunkt der Nachricht einer Station der Klasse A. Wenn die aktuelle Zeit der Station der Klasse B später als 10 ms des Zeitpunkts des Sendens der Nachricht ist, wird der Unterschied zwischen den Zeitpunkten als Offset zur Synchronisierung genutzt.

Der Sender-Thread wird am Beginn prüfen, was die Klasse der Station ist.
Wenn eine Station zur Klasse B gehört und in der Multicast Gruppe keine Station zur Klasse A gehört, muss die Station der Klasse B so lange warten, bis eine Station der Klasse A an der Gruppe teilnimmt, um Nachrichten zu schicken.

In der Klasse A muss nur geprüft werden, ob sie die erste Station der Gruppe ist. Wenn sie die erste Station der Gruppe ist, dann nimmt sie einen zufälligen Slot auf dem Frame, um Nachrichten zu schicken.

Die Vergabe der Slots wird über das Self-organized time-division multiple access (STDMA) – Verfahren erfolgen. Dabei sucht sich jede Station eigenständig einen freien Slot zum Senden, während alle anderen Slots stetig zum Empfangen genutzt werden.

Damit aber eine Kollision vermieden werden kann, wird im Paket angegeben, von welchem Slot die Nachricht im nächsten Frame gesendet wird. Dadurch muss die sendende Station beim Durchgehen der Liste der Slots prüfen, ob dieser der vorher angegebene Slot ist, von dem gesendet wird. Sobald eine Kollision erkannt wurde, wird versucht die letzte Nachricht erneut zu schicken und die Stationen, die kollidieren, müssen die Slots neu berechnen.



1 - UML Klassendiagramm