Team: 1, The Kiet Dang, Sofian Wüsthoff

Aufgabenaufteilung:

1. The Kiet Dang,
   Vorbereitung: Klassendiagramm, Zeitsynchronisation, IP-Multicast
   Implementierung:
   + Datei aus Datenquelle Empfangen
   + Zeitsynchronisation
   + IP-Multicast Empfangen
2. Sofian Wüsthoff,
   Vorbereitung: STDMA, Kollision, Requirements
   Implementierung:
   + Slots verteilen
   + Kollision behandeln
   + IP-Multicast Senden

Quellenangaben:

1: Vorlesung Verteile Systeme

2:http://marinegyaan.com/what-is-stdma-sotdma-technology/stdma/

3: https://www.baeldung.com/java-broadcast-multicast

4: java doc

5. stackoverflow

Bearbeitungszeitraum:

20.11.2019 5 Stunden

25.11.2019 4 Stunden

30.11.2019 8 Stunden

6.12.2019 5 Stunden

8.12.2019 3 Stunden

Aktueller Stand: Der Entwurf ist ausführlich und fertig. Wir könnten die Datei aus der Datenquelle übernehmen und in unsere Nachrichten einpflegen. Dazu funktioniert auch schon das Kommunizieren zwischen den Stationen im IP-Multicast Netz. Wir versuchen gerade die Zeitsynchronisation zu implementieren, sowie Kollisionen zu behandeln.

Änderungen des Entwurfs:

- Ergänzen des Zeit Diagramms zwischen Klasse A und Klasse B
- Detaillierte Beschreibung des Verfahrens, der Kollisionsvermeidung
- Requirements überarbeitet

Aufgabe 3 – 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. Betrieb soll mit bis zu 25 Slots laufen
4. Vergabe der Slots über STDMA – Verfahren
5. Kollisionen in der Auswahl der Slots vermeiden
6. Zeiten basieren auf UTC
7. Uhren von Klasse A synchronisieren sich untereinander
8. Uhren von Klasse B synchronisieren sich mit denen von Klasse A

Systementwurf

Jede Station hat die Möglichkeit, durch IP-Multicast, Daten zu senden und zu empfangen. Senden 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 der „messageBuffer“.

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

Die Zeit Synchronisation 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 ist als 10 ms des Zeitpunkts der Nachricht, wird der Unterschied zwischen den Zeitpunkten als Offset synchronisiert.

Der Sender-Thread wird am Beginn prüfen, was ist 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 auf an Gruppe teilnimmt, um Nachrichten zu schicken.

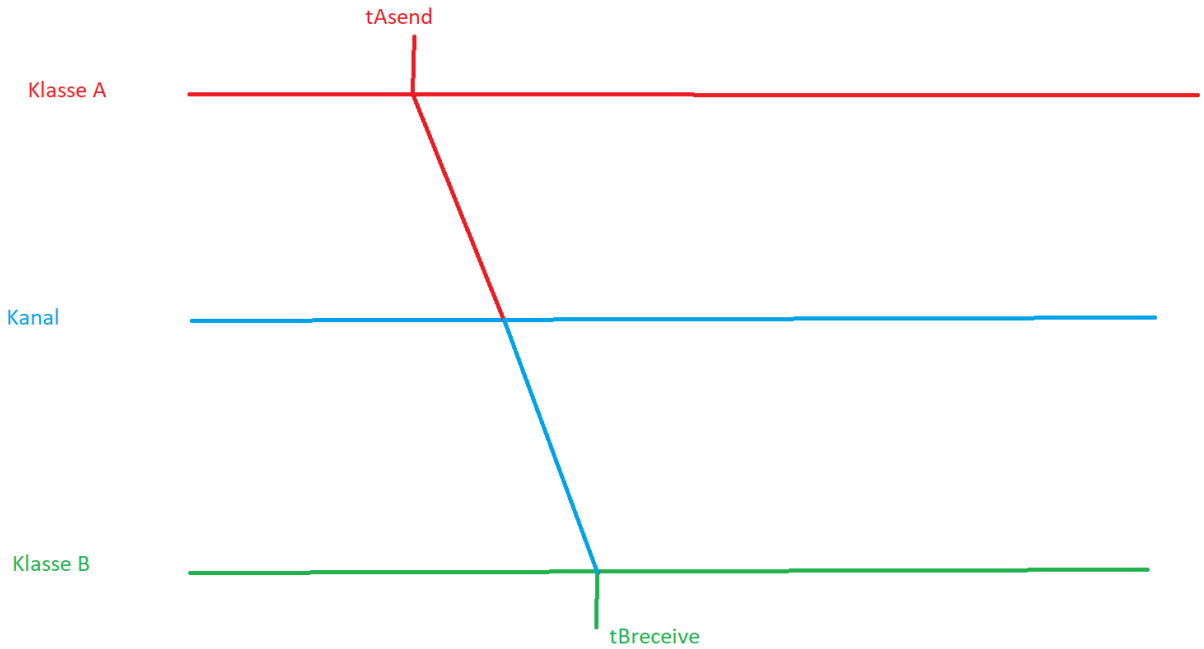
In der Klasse A muss nur geprüfet werden, ob es die erste Station in Gruppe ist. Wenn sie ist die erste Station der Gruppe ist, dann nimmt sie einen zufälligen Slot auf dem Frame, um Nachrichten zu schicken.

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 welchen Slot die Nachricht im nächsten Frame gesendet wird. Es wird durch den Nachrichtenpuffer des jetzigen Frames iteriert und von jeder Nachricht wird dann herausgelesen, an welchen Slot die Station als nächstes Senden wird. Dadurch werden dann die herausgelesen vermieden und somit eine Kollision.

1 - UML KLassendiagramm

2 – *Zeit Diagramme*



tDiff = tAsend - tBreceive
if
tDiff >= 10ms (toleranz Zeit)
then
tB = tB + tDiff