Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Referat

Daniel Schruhl

Entwurf und Implementierung einer STDMA Station

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Daniel Schruhl

Entwurf und Implementierung einer STDMA Station

Referat eingereicht im Rahmen der Referat eingereicht im Rahmen der Vorlesung Verteilte Systeme

im Studiengang Angewandte Informatik (AI)
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. C. Klauck

Eingereicht am: 31. Mai 2017

Daniel Schruhl

Thema der Arbeit
Entwurf und Implementierung einer STDMA Station

Stichworte
STDMA, Verteilte Systeme, Clojure

Kurzzusammenfassung
STDMA, Distributed Systems, Clojure

Daniel Schruhl

Title of the paper
Design and implementation of an STDMA station

Keywords

Abstract

Inhaltsverzeichnis

Listings

1 Einführung und Ziele

In verteilten Systemen werden oft Nachrichten unter verschiedenen Teilnehmern eines Netzes ausgetauscht. Dabei sind diese Teilnehmer oft nur durch ein Medium miteinander verbunden. Dieses eine Medium muss dann mehrere Teilnehmer unterstützen können.

Um dieses Problem zu lösen soll ein Softwareprodukt erstellt werden, das eine Station darstellt, die auf einem Medium Nachrichten empfangen und senden kann.

Dabei sollen mehrere Stationen auf einem Medium Nachrichten verschicken und senden. Das geschieht, indem das Medium in feste Frames aufgeteilt wird und jeder Frame eine feste Anzahl an Slots hat, in dem aus allen teilnehmenden Stationen immer nur eine in ihrem zugeteilten Slot Senden darf. Das Medium wird hierbei im Betracht der Zeit aufgeteilt. Diese Aufteilung (Multiplexing) wird auch time-division multiple access (TDMA) genannt. In diesem Fall sollen die Stationen selber untereinander ihre Slots zum Senden verwalten und untereinander gleichmäßig aufteilen. Das wird auch Self-organized time-division multiple access (STDMA) genannt.

1.1 Randbedingungen

Das zu verwendende Medium ist ein Socket, der per IP Multicast Nachrichtenpakete (Datagrams) an mehrere Klienten austeilen kann. Die TTL der Multicast-Pakete ist auf 1 zu setzen, um unnötige Netzlasten und Netzstörungen zu vermeiden.

Die Frames haben eine Länge von einer Sekunde. Ein Frame hat 25 Slots. Diese sind von 1 bis 25 nummeriert. Die Frames sind in Sekunden seit dem 1.1.1970, 00:00 Uhr nummeriert.

Jede Station darf nur einmal pro Frame senden. Das muss in der Mitte ihres zugeteilten Slots passieren.

Die Vergabe der Slots soll ohne zentrale Vergabeinstanz geschehen. Das geschieht, indem jedes Nachrichtenpaket ein Datenfeld enthält, in das die sendende Station die Nummer ihres Slots

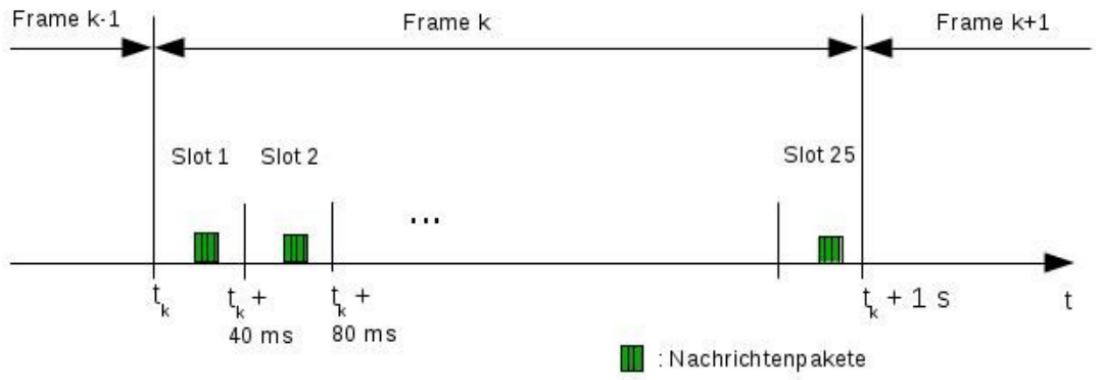

Abbildung 1.1: Aufteilung des Mediums in Frames und Slots, Quelle: Klauck (2017)

einträgt, die sie für den nächsten Frame zum Senden verwendet. Das bedeutet also, dass durch das Lesen der Nachrichtenpakete in einem Frame bestimmt werden kann, welche Slots im nächsten Frame frei sind.

Ein Nachrichtenpaket hat folgende Struktur:

- 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 Bytes

Stationen haben dabei eine Unterteilung in Klasse A und B. Diese Klassen werden für eine Synchronisierung der Zeit verwendet. Generell wird die UTC Zeit verwendet. Die Uhr einer Station kann einen anfänglichen Offset haben. Die Stationen synchronisieren dann untereinander im Laufe der Teilnahme am Netz ihre Uhren untereinander, so dass der Offset sich verändern kann.

Klasse A Uhren gelten als hinreichend genau und als Referenz zur Synchronisation. Klasse B und Klasse A Stationen können dann anhand anderer empfangen Nachrichtenpakete von Klasse A Stationen ihren Offset verändern. Dabei wird die Sendezeit und die Klasse innerhalb eines Nachrichtenpaketes verwendet.

Bei Kollisionen sollen die betroffenen Nachrichtenpakete als nicht empfangen betrachtet werden. Ein kollisionsfreier Betrieb mit maximal 25 Stationen muss nach endlicher Zeit erreicht werden und darf nicht wieder verlassen werden.

Das Produkt soll über die Kommandozeile gestartet werden können. Dabei müssen folgende Parameter mit übergeben werden können:

- Interfacename des Kommunikationsendpunktes
- Adresse der Multicast Gruppe, Klasse D IP-Adresse
- Port des Sockets
- Stationsklasse, A oder B
- Anfänglicher UTC Offset

Als Datenquelle für die Nutzdaten soll das STDIN des Produktes verwendet werden, um der Station neue Nutzdaten zukommen zu lassen.

1.2 Kontextabgrezung

Das Produkt muss auch in einer ssh-Sitzung auf einem anderen Rechner gestartet werden können. Es muss auf Rechnern mit Linux Betriebssystem lauffähig sein.

2 Gesamtsystem

Das Produkt ist in Clojure (v1.9) umgesetzt worden. Clojure ist eine funktionale Programmiersprache, die auf der JVM läuft. Dabei ist eine interoperabilität mit Java möglich. Das erleichtert den Umgang mit Sockets, da der Umgang mit diesen in Java gut dokumentiert und verwendbar ist. Außerdem hat Clojure im Vergleich mit Erlang einige andere Vorteile wie lazily evaluated Sequences oder eine bessere Testumgebung.

Funktionale Programmiersprachen haben den Vorteil in verteilten Systemen, das die Ergebnisse von Funktionen bei gleichen Parametern immer gleich sind. Diese Immutability von Daten hat den Vorteil, das nebenläufige Prozesse dadurch keine Datensynchronisation machen müssen. Dadurch wird eine erhöhte Threadsicherheit erreicht.

2.1 Architekturüberblick

2.2 Konfigurationsparameter

2.3 Benutzungsschnittstellen

3 Subsysteme und Komponenten

3.1 Connector

3.1.1 Aufgabe und Verantwortung

3.1.2 Schnittstelle

3.1.3 Entwurfsentscheidungen

3.1.4 Konfigurationsparameter

3.2 Payload-Source

3.2.1 Aufgabe und Verantwortung

3.2.2 Schnittstelle

3.2.3 Entwurfsentscheidungen

3.2.4 Konfigurationsparameter

3.3 Message-Writer

3.3.1 Aufgabe und Verantwortung

3.3.2 Schnittstelle

3.3.3 Entwurfsentscheidungen

3.3.4 Konfigurationsparameter

3.4 Station

3.4.1 Aufgabe und Verantwortung

3.4.2 Schnittstelle

3.4.3 Entwurfsentscheidungen

3.4.4 Konfigurationsparameter

Literaturverzeichnis

[Klauck 2017] Klauck, Prof. Dr. C.: VSP Aufgabe 3. (2017)

Erklärung zur schriftlichen Ausarbeitung des Referates

Hiermit erkläre ich, dass ich diese schriftliche Ausarbeitung meines Referates selbstständig und ohne fremde Hilfe verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe sowie die aus fremden Quellen (dazu zählen auch Internetquellen) direkt oder indirekt übernommenen Gedanken oder Wortlaute als solche kenntlich gemacht habe. Zudem erkläre ich, dass der zugehörige Programmcode von mir selbständig implementiert wurde ohne diesen oder Teile davon von Dritten im Wortlaut oder dem Sinn nach übernommen zu haben. Die Arbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher nicht veröffentlicht.

Hamburg, 31. Mai 2017 Daniel Schruhl