Fakultät Informatik/Mathematik

Lehrstuhl für Rechnernetze/Kommunikationssysteme

HOCHSCHULE FÜR
TECHNIK UND WIRTSCHAFT
DRESDEN
UNIVERSITY OF APPLIED SCIENCES

Prüfungsvorleistung

RTSP-STREAMING MIT FEC SCHUTZ

INTERNET-TECHNOLOGIEN 2

Falk-Jonatan Strube

Betreuender Hochschullehrer

Prof. Dr.-Ing. Jörg Vogt

Eingereicht am: 11. Januar 2018

INHALTSVERZEICHNIS

1 ÜBERBLICK

Beim FEC-Fehlerschutz beim RTSP-Stream wird zusätzlich zu den Datenpaketen in gewissen Abständen (abhängig von der Gruppengröße) ein FEC-Pakete versendet. Dieses Paket enthält die Daten aus den potentiell zu korrigierenden Paket mit XOR verknüpft. Ein verlorenes Paket aus der Gruppe kann somit wiederhergestellt werden: Die vorhandenen Pakete der Gruppe müssen mit dem FEC-Paket erneut mit XOR-Verknüpft werden.

2 IMPLEMENTIERUNG FECMANAGER

In dieser Implementation wurde das so realisiert, dass sowohl der `Client` als auch der `server` einen `FECmanager` haben. Dieser hat folgende Aufgabenbereiche:

- Buffern der Datenpakete
- Erstellen von FEC-Paketen (für den Server)
- Korrigieren von Datenpaketen (für den Client)

2.1 BUFFER

Der Buffer hat eine fest Größe. In ihn werden die Daten entsprechend der Sequenznummer (per Modulo-Rechnung) zugewiesen. Wichtig ist, dass der Client nicht länger warten und Buffern darf, als der Buffer groß ist, weil sonst Einträge überschrieben werden, bevor sie eine Chance haben ausgegeben zu werden.

2.2 ERSTELLEN VON FEC-PAKETEN

Die FEC-Pakete werden erstellt, indem jeweils die letzten k Datenpakete mit XOR-Verknüpft werden. Das FEC-Paket mit der Sequenznummer i deckt somit die Gruppe der Datenpakete von $i - k$ bis i ab. Das FEC-Paket wird nicht gebuffert.

2.3 KORRIGIEREN VON DATENPAKETEN

Das FEC-Paket kann ein fehlendes Datenpaket der Gruppe korrigieren (Gruppengröße und -reichweite siehe Unterabschnitt 2.2). Der Funktion des `FECmanager`s muss das FEC-Paket übergeben werden, da es nicht gebuffert wird. Sodann wird falls möglich ein entsprechend fehlender Buffereintrag korrigiert.

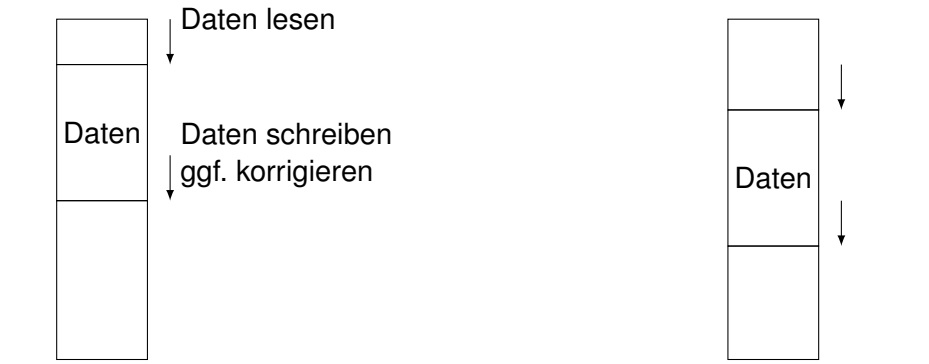
3 IMPLEMENTIERUNG SERVER

Beim Server wurde die Taktung des Timers verdoppelt. Jeden zweiten Takt wird ein Datenpaket versendet. Jeder anderer wird, ein FEC-Paket erstellt und versendet, falls es Zeit dafür ist (abhängig von der Gruppengröße alle $2k$ Takte).

4 IMPLEMENTIERUNG CLIENT

Der Client lässt die einkommenden Datenpakete erst für einige Sekunden buffern. Damit besteht genug Zeit, dass auch bei großer Gruppengröße die entsprechenden Pakete korrigiert werden können. Sobald die Bufferzeit verstrichen ist wird gleichzeitig weiter gebuffert und abgespielt

(siehe Abbildung 1). Abgespielte Buffereinträge werden gelöscht. Die Taktung des Timers ist, ähnlich wie beim Server, verdoppelt, sodass die Daten- und FEC-Pakete empfangen werden können. Die FEC-Pakete werden sofort angewandt (falls möglich und nötig).



(a) Buffer zum Zeitpunkt t_1 (b) Buffer zum Zeitpunkt t_2

Abbildung 1: Buffer im Client (Daten „wandern“)

5 OPTIMAL GRUPPENGRÖSSE

Es soll die optimale Gruppengröße für 10% Paketverlust gewählt werden.

Das Problem ist, dass bei größerer Gruppengröße die Wahrscheinlichkeit größer ist, dass mehr als ein Paket in einer Gruppe verloren geht. Das kann (mit $p = 1$) nicht mehr korrigiert werden. Also ist zu überlegen, wie groß die Gruppengröße gewählt werden kann, damit sie überhaupt noch Korrekturen vornehmen kann. Weiterhin ist zu beachten, dass die FEC-Pakete auch verloren gehen können.

Es wurden einmalige Messungen für verschieden Gruppengrößen vorgenommen (bei 500 Einzelbildern und 20 Sekunden):

Gruppengröße	Verloren	Korrigiert		FPS
		abs.	%	
20	56	3	5%	22.4
18	49	9	18%	23
16	47	7	15%	23
14	47	14	30%	23.4
12	48	12	25%	23.2
10	51	15	29%	23.2
8	48	23	48%	23.75
6	41	24	59%	24.15
4	42	27	64%	24.25
2	53	44	83%	24.55

Tabelle 1: Messungen der FPS nach FEC-Gruppengröße

Da die Ursprüngliche Framerate mit 25 FPS sehr gering ist, führt jede Verringerung der FPS-Anzahl zu einer deutlich verminderten „subjektiv zufriedenstellenden Bildqualität“. Dem

entsprechend wäre ein Wert zwischen 6 und 8 empfehlenswert, damit die Framerate möglichst nicht noch unter 24 FPS sinkt.

6 IMPLEMENTATIONSFEHLER

Durch einen noch nicht gefundenen Fehler in der Implementation kann es dazu kommen, dass korrigierte ein Bildfehler (am unteren Rand) haben.

Bei verschiedenen einfachen Tests werden keine Fehler beim Erstellen eines FEC-Pakets oder dem Korrigieren eines Datenpakets gefunden (siehe Listing 1: Die Daten in Zeile 13 und 17 entsprechen den Originaldaten (zuzüglich folgender Nullen). Das FEC-Paket ist korrekt.).

```
1  Input:
2  1 - Hex:                    [1, 2] / Bin: 0001 0010
3  2 - Hex:                [5, 7, 10] / Bin: 0101 0111 1010
4  3 - Hex:                    [6] / Bin: 0110
5  4 - Hex:   [10, 10, 10, 10, 15, 0] / Bin: 1010 1010 1010 1010 1111 0000
6  5 - Hex:                   [14] / Bin: 1110
7
8  FEC:
9      Hex:     [6, 15, 0, 10, 15, 0] / Bin: 0110 1111 0000 1010 1111 0000
10
11 Remove 3.
12       FEC-Paket 5 corrected frame 3.
13 Corrected 3: [6, 0, 0, 0, 0, 0]
14
15 Remove 2.
16       FEC-Paket 5 corrected frame 2.
17 Corrected 2: [5, 7, 10, 0, 0, 0]
```

Listing 1: Ausgabe des FEC Tests

Bei näherer Inspektion des Buffers zeigt sich, dass bei einem korrigiertem Bild anscheinend einige Bits am Ende fehlen. Dadurch kommt der Bildfehler zustande. Wie er verursacht wird, müsste durch weiter Analysen erörtert werden.