Fehleranalyse

Aufgabe 3 – Team 6

Mert Signic, Michael Müller

Verteilte Systeme

Prof. Dr. Klauck & Dipl.-Ing Schulz
Sommersemester 2018

Gliederung:

1. Was waren die Fehler (Fehlerwirkung)?
2. Fehler 1:
    a. Fehlerklassifizierung (Kurz Übersicht)
    b. Wo entstanden die Fehler (Lokalisierung)?
    c. Wie entstanden die Fehler (Fehlerquellen & -ursachen)?
    d. Wieso fielen die Fehler nicht auf?
    e. Wie wurden sie behoben (Lösungsstrategie)?
3. Fehler 2 (analog zu Fehler 1)
4. Fehler 3 (analog zu Fehler 1)
5. Generelles Vorgehen beim Testen
6. Quellen

(Prolog) Was war die Aufgabe?

Die Anwendung die wir Realisieren soll das Zeitmultiplex verfahren umsetzen.
Dabei sollen die Stationen über UDP Multicast (auch Kanal genannt) miteinander kommunizieren. Wobei auf Kollisionen geachtet wird.
Die Vergabe der Sende-Slots wird bei jeder Station lokal individuell bestimmt, indem man Nachrichten aus dem Kanal auswertet. Zudem werden die Nachrichten von Typ A Stationen verwendet, um die Uhren der eigenen Station zu synchronisieren.

1. Was waren die Fehler?

Das Programm führte zu Kollisionen die nicht hätten sein dürfen.

(Fehler 1 & 2) Auffällig war, dass schon reservierte Slots noch einmal reserviert wurden. Dies darf nicht passieren. Jeder Slot darf nur einmal reserviert werden!
So kam es dazu, dass diese Regel nicht beachtet wurde und (meist 2) Stationen im selben Slot gesendet haben, was dann zu Kollisionen führte die nicht hätten stattfinden dürfen.
Hier hat also die Slotreservierung nicht funktioniert wie sie sollte.

```
  ========= Frame: 1527237919  ==================
1527237919030 (slot  1): received 'team 06-11' B next slot:  2 TX: 1527237919029 (1 / 10)
1527237919070 (slot  2): received 'team 06-05' B next slot: 14 TX: 1527237919070 (0 / 10)
1527237919154 (slot  4): received 'team 06-16' B next slot: 12 TX: 1527237919153 (1 / 10)
1527237919229 (slot  6): received 'team 06-03' B next slot:  7 TX: 1527237919228 (1 / 10)
1527237919272 (slot  7): received 'team 06-15' B next slot: 24 TX: 1527237919271 (1 / 10)
1527237919348 (slot  9): received 'team 06-07' B next slot: 10 TX: 1527237919346 (2 / 10)
1527237919393 (slot 10): received 'team 06-06' B next slot: 10 TX: 1527237919393 (0 / 10)  <-- Slot bereits angekündigt
```

Abb. 1, Visualisierung von Fehler 1

```
1527237920550 (slot 14): received 'team 06-05' B next slot: 24 TX: 1527237920550 (0 / 10)
1527237920590 (slot 15): received 'team 06-02' B next slot:  2 TX: 1527237920589 (1 / 10)
1527237920675 (slot 17): received 'team 06-12' B next slot:  5 TX: 1527237920674 (1 / 10)
1527237920715 (slot 18): received 'team 06-17' B next slot:  3 TX: 1527237920714 (1 / 10)
1527237920760 (slot 20): received 'team 06-04' B next slot: 24 TX: 1527237920759 (1 / 10)
```

Abb. 2, Visualisierung von Fehler 2

(Fehler 3) Hinzu kommt in seltenen Fällen, dass richtig reserviert wurde, jedoch der Slot so verpasst wurde, dass man im Slot davor / danach sendete. Was wiederum zu falschen Kollisionen führte.
Hier hat also die Prüfung der Sendezeit beim Senden nicht richtig funktioniert oder die innere Uhr der Station falsch lief.

```
1527237916960 (slot 25): received 'team 06-04' B next slot: 24 TX: 1527237916959 (1 / 10)
  ========= Frame: 1527237917  ==================
1527237917030 (slot  1): received 'team 06-11' B next slot:  4 TX: 1527237917029 (1 / 10)  <-- Slot 4 angekündigt
1527237917071 (slot  2): received 'team 06-19' B next slot: 15 TX: 1527237917070 (1 / 10)
1527237917113 (slot  3): received 'team 06-08' B next slot: 24 TX: 1527237917111 (2 / 10)
1527237917190 (slot  5): received 'team 06-05' B next slot:  5 TX: 1527237917190 (0 / 10)
1527237917233 (slot  6): received 'team 06-17' B next slot: 22 TX: 1527237917232 (1 / 10)
1527237917309 (slot  8): received 'team 06-03' B next slot: 20 TX: 1527237917308 (1 / 10)
1527237917352 (slot  9): received 'team 06-12' B next slot:  9 TX: 1527237917350 (2 / 10)
1527237917388 (slot 10): received 'team 06-09' B next slot: 17 TX: 1527237917386 (2 / 10)
1527237917433 (slot 11): received 'team 06-14' B next slot: 13 TX: 1527237917432 (1 / 10)
1527237917475 (slot 12): received 'team 06-20' B next slot:  2 TX: 1527237917474 (1 / 10)
1527237917512 (slot 13): received 'team 06-06' B next slot: 16 TX: 1527237917512 (0 / 10)
1527237917550 (slot 14): received 'team 06-02' B next slot:  6 TX: 1527237917548 (2 / 10)
1527237917629 (slot 16): received 'team 06-07' B next slot:  8 TX: 1527237917627 (2 / 10)
1527237917673 (slot 17): received 'team 06-16' B next slot: 11 TX: 1527237917672 (1 / 10)
1527237917711 (slot 18): received 'team 06-10' B next slot: 12 TX: 1527237917710 (1 / 10)
1527237917755 (slot 19): received 'team 06-18' B next slot:  7 TX: 1527237917754 (1 / 10)
1527237917860 (slot 22): received 'team 88-01' A next slot: 23 TX: 1527237917860 (0 / 10)
1527237917916 (slot 23): received 'team 06-01' B next slot: 24 TX: 1527237917915 (1 / 10)  <-- Slot bereits angekündigt
1527237917960 (slot 25): received 'team 06-04' B next slot:  3 TX: 1527237917959 (1 / 10)
  ========= Frame: 1527237918  ==================
1527237918077 (slot  2): received 'team 06-20' B next slot: 24 TX: 1527237918075 (2 / 10)
1527237918120 (slot  4): received 'team 06-04' B next slot: 20 TX: 1527237918119 (1 / 10)
1527237918150 (slot  4): received 'team 06-11' B next slot:  1 TX: 1527237918149 (1 / 10)
   >>>> collides with previous packet. (Frame: 1527237918)
```

Abb. 3, Visualisierung von Fehler 3

2. Fehler 1

a. Fehlerklassifizierung (Kurz Übersicht)

Fehlerkriterien	Fehlerklassifizierung
Art	Logischer Fehler
Ursache	Falsche Implementation des Entwurfs
Zeitpunkt der Erstellung	Bei der Implementation des Entwurfs
Zeitpunkt des Auftretens	In der Ausführung (wenn die Station in der Sendephase ist und einen Slot für den nächsten Frame reserviert)
Auswirkung	Eine Nachricht die einen Slot im nächsten Frame reserviert wird nicht verarbeitet und mit Zufallspech wählt die eigene Station den selben Slot was zu einer nicht legalen Kollision führen wird.
Aufwand / Dauer zur Behebung	Kurz, da nur der Moment der Anfrage zeitlich nach hinten geschoben werden muss
Status	Behoben

b. Wo entstand der Fehler?

Der Entwurf ist noch richtig.

In der Sendephase wird erst kurz vor Senden der Nachricht der Slot für den nächsten Frame vom SlotFinder angefordert. So ist garantiert, dass alle bisher empfangenen Nachrichten (und deren reservierten Slots) berücksichtigt wurden.

Die Implementation war nicht fehlerfrei.

Es fiel auf, dass es auffällig oft Kollision von Stationen gab die im Frame davor direkt nacheinander sendeten. Zum Beispiel (Abbildung 1) sendete Station 07 in Slot 9, dass diese im nächsten Frame Slot 10 reserviert. Station 06 in Slot 10, wollte auch die 10. Alle anderen Stationen wurden richtig berücksichtigt. So lag es nahe, dass die Nachricht von Station 07 in Station 06 ggf. nicht berücksichtigt wurde, wenn es darum ging einen eigenen Slot im nächsten Frame zu reservieren.

Im SlotFinder gibt es zwar einen weiteren Fehler (Fehler 2) jedoch ist dieser lös gelöst von Fehler 1. Und unter Ausschluss von Fehler 2 hat auch (wie man anhand der Stations Logs erkennen konnte) der SlotFinder so gearbeitet wie das erwartet wurde. Weswegen wir wussten, der Fehler liegt nicht im SlotFinder.

c. Wie entstand der Fehler?

Der Fehler lag dann im Detail daran, dass der Slot zu früh angefordert wurde. Zu dem Zeitpunkt der Slot Anforderung sendete eine andere Station (im Slot davor) zeitgleich die Nachricht, die dann eben nicht berücksichtigt wurde. Bis diese nämlich voll ausgewertet wurde hat die eigene Station längst einen Slot ausgewählt (der mit ein wenig Pech der ist den die Station davor gerade reserviert hatte).

In der nachfolgenden Abbildung kann man genau das sehen.
In Zeile 338 wird eine Nachricht empfangen und an die Pids (Also an die Uhr und den SlotFinder) gesendet. Zeitgleich wird nach einem noch freien Slot gefragt und zurückgegeben (Zeile 340). Leider ist die vorhin angekommene Nachricht noch nicht voll ausgewertet, dies passiert erst nach der Anforderung in Zeile 340.
In Diesem Fall ging es noch gut, da die eingehende Nachricht im nächsten Frame Slot 17 für sich beansprucht (vgl. Zeile 334 mit Zeile 341). Dies hätte jedoch mit ein wenig Pech schiefgehen können und unsere Station hätte auch die 17 statt die 4 ausgewählt.

```
334  "18.05 20:30:46,760|" -- Slot Taken: [8,20,6,13,14,14,12,22,19,8,21].
335  "18.05 20:30:46,760|" -- Clock TransportDelay is 0.
336  "18.05 20:30:46,761|" -- Slot Possible: [1,2,3,4,5,7,9,10,11,15,16,17,18,23,24,24,25].
337                    [...]
338  "18.05 20:30:46,794|" - Recv Got Message.
339  "18.05 20:30:46,800|" - Recv Now Sending to the pids.
340  "18.05 20:30:46,800|" - Send Got NextSlotnumber 4.
341  "18.05 20:30:46,801|" -- Slot Taken: [8,20,6,13,14,14,12,22,19,8,21,17].
342  "18.05 20:30:46,801|" -- Clock TransportDelay is 0.
343  "18.05 20:30:46,801|" -- Slot Possible: [1,2,3,4,5,7,9,10,11,15,16,18,23,24,24,25].
```

Abb. 4, aus Run 13 ‚team 06-02.log‘ , auch zu erkennen: Fehler 2

d. Wieso fiel der Fehler nicht eher auf?

Zwischen Abgabe und Nachreichung wurde ein Problem gefixt.

Und zwar bekommt der SlotFinder die kollisionsfreie Nachricht (geschaltet in einer Liste) und merkt sich diese (fügt sie hinten an die bereits erhaltenen Nachrichten an), um sie in find_slot_in_next_frame auszuwerten.
Irrtümlich wurde allerdings im Aufruf von find_slot_in_next_frame (Zeile 29) jedoch nicht NewMessages sondern Messages (also die vor dem aktuellen Slot empfangenen Nachrichten) übergeben. So wurde die letzte empfangene Nachricht nicht direkt ausgewertet.

```
27      {newmessages, ReceivedMessages} ->
28              NewMessages = ReceivedMessages ++ Messages,
29              NewSelectedSlot = find_slot_in_next_frame(NewMessages, LogFile),
30              loop(CorePid, StationName, NewSelectedSlot, NewMessages, LogFile);
```

Abb. 5, der nun richtige 1. Parameter der find_slot_in_next_frame Funktion

Diesen Fehler haben wir bemerkt. Und dementsprechend die richtige Liste übergeben. Damit dachten wir wäre das Problem der fehleranfälligen Slot Findung gelöst. Und sendeten die Nachreichung ab.

Wir hätten vor dem Absenden noch einen Lauf machen sollen, dann wäre uns Fehler 1 noch aufgefallen. Jedoch hatten wir Samstagnachmittag bis abends kein Zugriff auf einem Linux Rechner. Weshalb wir es nicht testen konnten.

Zudem wurden nur die einzelnen Unterfunktionen per eunit getestet. Das große Ganze jedoch dann immer erst mit Ausführung einer Station bzw. mehrerer Stationen.
Da man ja im Vorhinein nie weiß welche Funktion (egal wie groß) falsch sein könnte, müsste man also von vorne herein alle Funktionen auf alle Arten testen. Für die größten Testcases (wie der hier) war aber leider keine Zeit, da die Zeit schon so kaum reichte. Denn auch die Bachelorarbeit / Lohnarbeit brauchte eine gewisse zeitliche Zuwendung.

Deswegen verfolgten wir den Ansatz, die größten Funktionen so zu testen, in dem wir die Log Dateien nach Ausführung überprüften. Optimal ist das nicht. Gerne hätten wir es mit mehr Zeit alles in den eunit Tests abgefangen.

e. Wie wurde er behoben?

Die Slot Anforderung wurde nun in sender.erl zeitlich nach hinten verschoben, direkt vor dem Senden der Nachricht wird nun erst ein Slot angefordert. Somit ist garantiert, dass alle bis dahin empfangenen Nachrichten des Frames berücksichtigt werden, was nun den Slot direkt vor dem eigenen miteinschließt. (Wie im Entwurf spezifiziert)

3. Fehler 2

a. Fehlerklassifizierung (Kurz Übersicht)

Fehlerkriterien	Fehlerklassifizierung
Art	Logischer Fehler
Ursache	Falsche Implementation des Entwurfs
Zeitpunkt der Erstellung	Bei der Implementation des Entwurfs
Zeitpunkt des Auftretens	In der Ausführung (wenn die Station in der Sendephase ist und einen Slot für den nächsten Frame reserviert)
Auswirkung	Ein bereits reservierter Slot 24 wird noch einmal reserviert, was zu einer nicht legalen Kollision führen wird.
Aufwand / Dauer zur Behebung	Sehr Kurz, da nur die zweite 24 aus der Konstanten Liste in SlotFinder.erl gelöscht werden muss.
Status	Behoben

b. Wo entstand der Fehler?

Der Entwurf ist noch richtig.

Es gibt die Slotnummern 1 bis einschließlich 25.

Die Implementation war nicht fehlerfrei.

Bei Testläufen hatten wir uns ausgeben lassen welcher Slot nun schon reserviert ist und welche noch nicht reserviert wurden. Dabei viel auf, dass obwohl die 24 schon reserviert war sie dennoch noch in der Liste der nicht reservierten enthalten war. Beim genauerem Hinsehen konnten wir feststellen wie sich zu Beginn eines neuen Frames stets zwei Mal die Zahl 24 in der Liste der nicht reservierten Slots befand.

Da bei Framebeginn die nicht reservierte Liste initial mit der Konstanten Liste in SlotFinder besetzt wurde, musste also in der Konstante schon etwas falsch sein.

Zusehen in Abbildung 2, obwohl die 24 schon im weit entfernten Slot 14 reserviert wurde (von Station 06) wird sie dennoch noch einmal in Slot 20 (von Station 04) reserviert.

Die große zeitliche Entfernung wird hier hervorgehoben, um klar zu machen, dass Fehler 2 kein Aspekt von Fehler 1 ist, sondern ein selbstständiger Fehler mit eigener Fehlerursache und -wirkung.

c. Wie entstand der Fehler?

Im SlotFinder gibt es eine Liste (als Konstante) mit allen möglichen Slots.
Diese Liste sollte die Zahlen 1 bis einschließlich 25 je einmal halten.
Jedoch kam die 24 zwei Mal vor.

Siehe Abbildung 4 (Zeilen 336 und 343).

d. Wieso fiel der Fehler nicht eher auf?

Die Liste an sich wurde nie selbst getestet.
In den Funktionen in denen die Liste gebraucht wird wurde sie nicht per „?Konstantenname“ genutzt sondern per Parameter mitgegeben. So wurde beim Testen immer eine eigene Liste mit möglichen Slotnummern mitgegeben statt die interne Konstante zu verwenden.
Zudem kommt, dass wir Sets nicht verwenden wollten, da der Lernerfolg größer ist wenn man keine fortgeschrittenen ADTs von erlang direkt verwendet. Und wir nun eine Liste wie ein Set behandelten aber nie logisch sicherstellten, dass es auch wirklich wie ein Set arbeitet (vor allem die Eigenschaft, dass jedes Element nur einmal vorkommt).

e. Wie wurde er behoben?

Die zweite 24 wurde aus der Liste gelöscht.

```
12  -define(DEFAULT_POSSIBLE_SLOTS, [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25]).
```

Abb. 6, die nun gefixte Liste ohne die zweite 24 in SlotFinder.erl

4. Fehler 3

a. Fehlerklassifizierung (Kurz Übersicht)

Fehlerkriterien	Fehlerklassifizierung
Art	Technischer Fehler
Ursache	Frame Länge oft länger als die spezifizierten 1000 MS, da v.a. die Timer meist etwas länger als die festgelegte Zeit brauchen
Zeitpunkt der Erstellung	Bei der Implementation des Entwurfs
Zeitpunkt des Auftretens	In der Ausführung, wenn die Station mit einem Frame fertig ist und den nächsten beginnt
Auswirkung	Die Framestartzeit wird immer weiter nach hinten geschoben weshalb die Stationszeit nach einiger Laufzeit völlig daneben ist.
Aufwand / Dauer zur Behebung	Lang, da die Architektur umgestellt werden muss. Von Sequentiellen Abarbeiten der Frames hinzu Prüfung der Framestartzeit und Signalen (das eine neuer Frame begonnen hat).
Status	Behoben

b. Wo entstand der Fehler?

Auch hier passt es im Entwurf noch.

Die Frames sind exakt 1000 MS, die Slots genau 40 MS lang. Es sollte nicht gesendet werden, wenn die Mitte des Sendeslots verpasst wird.

Hierbei muss man beachten, dass man zwischenzeitlich nach einem Entwurf implementierte der vor sah eben diese Sequentielle Logik zu implementieren. Dieses Vorgehen war also damals korrekt, mit dem jetzigen Entwurf wäre die Abweichung natürlich Falsch gewesen.
Mit der Korrektur in Entwurf und danach im Code sind beide stets konsistent.

Jedoch auch in der Implementation Fehler:

Wie man in Abbildung 3 sieht, führt diese Zeitverschiebung zu erheblichen Problemen. Zunächst reserviert Station 04 Slot 24 für den nächsten Frame (Zeile 38). Im nächsten Frame sendet die Station jedoch im nächsten Frame in Slot 25 (Zeile 58). Es war Glück, dass es hier noch nicht zu einer Kollision kam. Im selben Zug reserviert sie Slot 3, was eine korrekte (und die letzte mögliche) Wahl ist. Sende dann jedoch wiederum in Slot 4. Also war hier etwas falsch mit der Überprüfung der Sendezeit oder der internen Uhr.

In den Stations spezifischen Logs viel uns nicht auf, dass die Station an sich falsch sendet. Die Sendezeit wurde richtig errechnet. Und es wurde auch richtig geprüft ob die Sendezeit eingehalten wird oder nicht. Deswegen haben wir untersucht auf welcher Basis diese Sendezeit errechnet wird und stellten fest, dass die Frame beginne immer später wurden. So als bräuchten die Frames eben nicht exakt 1000 MS sondern länger. Und diese Zeit jenseits der 1000 MS akkumulierte sich und verschob den nächsten Framebeginn immer wieder ein klein wenig nach hinten. Den Fehler vermuteten wir deshalb in der core.erl Datei im frame_loop. Und wie die Frames aufeinander aufbauen.

c. Wie entstand der Fehler?

Auch hier war die Implementation der Fehler. Dieses Mal technischer Natur, da „timer“ in Erlang nie 100% exakt arbeitet und nur eine Mindestwartezeit statt die Gesamtwartezeit garantiert.
Unser Programm hat als Kernstück core.erl. Dieses Modul besitzt die frame_loop Funktion. Diese repräsentiert einen Frame und sollte exakt 1000 MS lang sein. Ob ein Frame auf die Sekunde genau anfängt wurde nie kontrolliert, sondern es wurde fest damit gerechnet, dass die Framelänge genau eingehalten wird.

Immer wieder war sie jedoch länger als 1000 MS.
Weshalb der Frame für eine Station immer später begann (Weil die Loops sequentiell ausgeführt wurden, also aufeinander aufbauten). Weshalb die Station dachte sie sende in Slot 3, war global jedoch gesehen schon in Slot 4.

Diese Zeitverschiebung kann man ganz gut in der nachfolgenden Abbildung sehen. Man sieht wie die Framestartzeit nur selten auf die Sekunde genau los geht, sondern meist später. Man sieht wie gegen Ende die Startzeit immer später wird. Bis sie schon mehr als einen halben Slot versetzt anfängt. Am Offset kann dies nicht liegen. Da die Framestartzeit ja aus aktueller Systemuhrzeit (vsutil:getUTC()) und offset (Addition) berechnet wird und auch dann immer auf die Sekunde genau sein soll.

```
"18.05 13:26:28,001|" Core New Frame Started at 1526642788001 with Slot 0 --------.
"18.05 13:26:29,006|" Core New Frame Started at 1526642789006 with Slot 0 --------.
"18.05 13:26:30,007|" Core New Frame Started at 1526642790007 with Slot 24 --------.
"18.05 13:26:31,007|" Core New Frame Started at 1526642791000 with Slot 0 --------.
"18.05 13:26:32,007|" Core New Frame Started at 1526642792000 with Slot 15 --------.
"18.05 13:26:33,006|" Core New Frame Started at 1526642793000 with Slot 0 --------.
"18.05 13:26:34,006|" Core New Frame Started at 1526642793998 with Slot 4 --------.
"18.05 13:26:35,006|" Core New Frame Started at 1526642794998 with Slot 0 --------.
"18.05 13:26:36,006|" Core New Frame Started at 1526642796002 with Slot 24 --------.
"18.05 13:26:37,006|" Core New Frame Started at 1526642796999 with Slot 0 --------.
"18.05 13:26:38,006|" Core New Frame Started at 1526642798002 with Slot 10 --------.
"18.05 13:26:39,006|" Core New Frame Started at 1526642799008 with Slot 0 --------.
"18.05 13:26:40,006|" Core New Frame Started at 1526642800009 with Slot 20 --------.
"18.05 13:26:41,012|" Core New Frame Started at 1526642801017 with Slot 4 --------.
"18.05 13:26:42,012|" Core New Frame Started at 1526642802019 with Slot 0 --------.
"18.05 13:26:43,012|" Core New Frame Started at 1526642803024 with Slot 3 --------.
"18.05 13:26:44,012|" Core New Frame Started at 1526642804011 with Slot 21 --------.
"18.05 13:26:45,012|" Core New Frame Started at 1526642805013 with Slot 5 --------.
"18.05 13:26:46,013|" Core New Frame Started at 1526642806017 with Slot 12 --------.
"18.05 13:26:47,013|" Core New Frame Started at 1526642807022 with Slot 3 --------.
"18.05 13:26:48,013|" Core New Frame Started at 1526642808026 with Slot 7 --------.
```

Abb. 7, aus Run 12, ‚team 06-01.log‘

d. Wieso fiel der Fehler nicht eher auf?

Der Fehler fiel bei genauem Blick auf die Logs auf, doch hatten wir erst spät eine Idee wie wir ihn lösen können. Und kümmerten uns zunächst um die Fehler 1 & 2 für die wir bereits eine gute Spur hatten.

e. Wie wurde er behoben?

Nun haben wir einen Timer in der UTCClock der nach 1000 MS sich selber ein Signal gibt. Dieser Timer wird angepasst wenn der Offset angepasst wurde (also dementsprechende ‚Restzeit im aktuellen Frame'). Der Timer wird beim starten der Uhr erstellt und jedes Mal resettet wenn eine neuer Frame beginnt.
Läuft der Timer ab, schickt der Timer eine bestimmte Nachricht an die UTCClock, die wiederum eine bestimmte Nachricht an den Core schickt. Für den Core das Zeichen, dass ein neuer Frame eben begonnen hat.
Der Timer ist initial auf 1000 MS gestellt, da etwas Zeit vergeht bis die Nachricht (dass ein neuer Frame angefangen hat) beim Core angekommen ist und der Timer selbst manchmal zusätzliche Zeit benötigt, könnte es jedoch unter Umständen dazu kommen, das die Nachricht im Core 1 MS zu spät ankommt.

Dieses Verhalten kann man in Abbildung 7 gut beobachten. Dort kann man nun auch sehen, dass die Änderung direkt geholfen hat den Framestart wesentlich genauer zu machen.

```
"28.05 12:15:56,002|" Core New Frame Started at 1527502556001 with Slot 0 --------.
"28.05 12:15:57,001|" Core New Frame Started at 1527502557001 with Slot 6 --------.
"28.05 12:15:58,001|" Core New Frame Started at 1527502558000 with Slot 10 --------.
"28.05 12:15:59,000|" Core New Frame Started at 1527502559000 with Slot 22 --------.
"28.05 12:16:00,001|" Core New Frame Started at 1527502560001 with Slot 5 --------.
"28.05 12:16:01,002|" Core New Frame Started at 1527502561000 with Slot 19 --------.
"28.05 12:16:02,001|" Core New Frame Started at 1527502562001 with Slot 10 --------.
"28.05 12:16:03,003|" Core New Frame Started at 1527502563000 with Slot 15 --------.
"28.05 12:16:04,001|" Core New Frame Started at 1527502564001 with Slot 8 --------.
"28.05 12:16:05,001|" Core New Frame Started at 1527502565000 with Slot 1 --------.
"28.05 12:16:06,001|" Core New Frame Started at 1527502566001 with Slot 23 --------.
"28.05 12:16:07,002|" Core New Frame Started at 1527502567001 with Slot 18 --------.
"28.05 12:16:08,000|" Core New Frame Started at 1527502568000 with Slot 2 --------.
"28.05 12:16:09,001|" Core New Frame Started at 1527502569001 with Slot 10 --------.
"28.05 12:16:10,001|" Core New Frame Started at 1527502570000 with Slot 4 --------.
"28.05 12:16:11,002|" Core New Frame Started at 1527502571000 with Slot 15 --------.
```

Abb.8, die verbesserte Version, aus Run 12,‘team 06-01.log‘

5. Generelles Vorgehen beim Testen

Bisher haben wir Fehler durch das Testen im Labor behoben. Vieles war unteranderem durch unsere geschriebenen Tests abgedeckt.

Generell gingen wir folgendermaßen vor:
Code und Tests wurden, wie in Clean Code von Robert C. Martin beschreiben, Schritt für Schritt zeitlich nah erstellt.
Danach haben wir unseren Code kompiliert und überprüft, ob der Compiler bereits Fehler warf. Waren hier keine Syntax-Fehler zu finden, wurde die Anwendung mit den Eunit Tests getestet.

Meist fingen wir an die Anwendung mit nur einer Station zu starten, um zu prüfen, ob es einen Laufzeit-Fehler gibt. Gab es diese nicht, wurden die Logs überprüft, ob die Daten in den Logs stimmen. War auch hier kein Fehler zu entdecken, haben wir die Anzahl der Stationen erhöht und die Anwendung erneut gestartet. Hierbei war stets der Sniffer aktiv. Hier dann natürlich auch wieder eine Logische Überprüfung der Logs.

Nachdem Unstimmigkeiten entdeckt wurden, die durch Logik-Fehler auftraten, wurden diese dann lokalisiert. Dabei haben wir gemeinsam Gebrainstormt wie und wann dieser bestimmte Fehler auftreten kann, und wo im Code dieser Fehler auftritt.
Gab es zu diesem Punkt noch keine Ideen, so wurde mit Hilfe der Sequenzdiagramme versucht den Fehler zu lokalisieren.
Zuletzt wurde gemeinsam nach einer Lösung gesucht und gemeinsam die Lösung implementiert.

Weshalb die ursprüngliche Aufteilung der Komponenten stetig aufweichte und wir beide nun den gesamten Code kennen.

6. Quellen

Abbildungen 1, 2 und 3 entstanden aus Ihrer gesendeten Log Datei (vom 25.05.18).
Code Abbildungen sind aus dem aktuellen Code.
Die Logs zu den Log Abbildungen sind in unserer Owncloud:
https://owncloud.informatik.haw-hamburg.de/index.php/f/26095377