

Verteilte Systeme

Hausarbeit

1

1

Aufgabenstellung

- ◆ Für die Hausarbeit ist ein **kausaler Multicast** gemäß dem gegebenen CBcast-Algorithmus in der Programmiersprache Erlang/OTP zu realisieren.
- ◆ Bei der Implementierung sind die vorgegebenen Schnittstellen einzuhalten, so dass die einzelnen zugehörigen Programmteile mit Software anderer EntwicklerInnen austauschbar ist.
- ◆ Für den implementierten kausalen Multicast ist eine kleine, prototypische Anwendung zu implementieren, z.B. eine Replikation einer Datei.

2

2

Aufgabenstellung

- ◆ Für die schriftliche Ausarbeitung ist die Gliederung gemäß Vorgabe einzuhalten.
- ◆ Bei dem Korrektheitsbeweis ist die Korrektheit bzgl. der Aufgabenstellung nachzuweisen, d.h. durch konkrete, detaillierte Bezüge zum Programmcode ist nachzuweisen, dass der vorgegebene Algorithmus korrekt implementiert wurde.
- ◆ Bei der Komplexitätsanalyse ist durch konkreten, detaillierten Bezug zum Programmcode eine Analyse der realisierten Komplexität vorzunehmen. Ggf. ist nachzuweisen, dass die vorgegebene Komplexitätsklasse des vorgegebenen Algorithmus eingehalten wird.

3

3

Aufgabenstellung

Zur Abgabe gehören

- ◆ Schriftliche Ausarbeitung mit den Kapiteln/Abschnitten
 - Theorie
 - Entwurf
 - Realisierung
 - Analyse mit den Abschnitten
 - Korrektheitsbeweis und
 - Komplexitätsanalyse
 - Fazit
- ◆ Programmcode bestehend aus den Dateien
 - cbCast.erl
 - vectorC.erl
 - towerClock.erl
 - towerCBC.erl

Die Abgabe zählt als nicht erfolgt, wenn ein Teil dieser Abgabe fehlt oder in unzulänglicher Qualität (zB nicht lauffähiger Code oder inhaltlich nicht adäquaten Kapitel/Abschnitten) abgegeben wurde.

4

4

Ausarbeitung

Folgende Gliederung der schriftlichen Ausarbeitung ist einzuhalten:

1. „**Theorie**“: z.B. Historie des/der Algorithmen, der Algorithmus selbst, evtl. Komplexitätsklasse, Schwächen/Stärken etc.
2. „**Entwurf**“: Aufbereitung der Theorie für eine technische Realisierung, detaillierte inhaltliche Analyse der Theorie, Einarbeitung der Anforderungen. Bestimmung der konkreten (abstrakten) Datenstrukturen und konkreten Algorithmen.
3. „**Realisierung**“: technische Umsetzung. Wie wurden Datenstrukturen und Abläufe umgesetzt, ggf. welche Änderungen mussten vorgenommen werden. Wie wurden die Anforderungen sichergestellt. Direkter Bezug zum Entwurf muss vorhanden sein.
4. „**Analyse**“ Implementierung: Abschnitte zu Korrektheitsbeweis des Codes (Direkte Bezüge zwischen Code und Entwurf müssen vorhanden sein) und Komplexitätsanalyse (Direkter Bezug zum Entwurf und/oder Code müssen vorhanden sein). Des Weiteren sind z.B. Laufzeitmessungen oder Laufzeitexperimente (Verhalten bei unterschiedlichen Parametern) möglich.
5. „**Fazit**“: Rückblickend den Weg von Theorie zu Praxis betrachten und eine Bewertung der Anwendung.

5

5

Entwurf: hier

- ◆ Der Entwurf enthält eine detaillierte Beschreibung der benötigten Daten in den jeweiligen Schritten der Lösungs-Algorithmen (**Lösungs-Datenstrukturen**). Es wird detailliert beschrieben, welche Daten in welcher Form in den jeweiligen Schritten benötigt werden. Die Beschreibung orientiert sich an den inhaltlichen/funktionalen Anforderungen. Die Datenstrukturen sind explizit vor den Algorithmen zu beschreiben.
- ◆ Der Entwurf enthält die detaillierte Beschreibung der inhaltlichen/funktionalen Aufgaben/Abläufe des Problems (**Lösungs-Algorithmen**) und ggf. der verwendeten Datenstrukturen. Jeder Schritt ist im Detail durchdacht, d.h. es gibt bzgl. der inhaltlichen/funktionalen Aufgabe keine offenen Fragen mehr! Darstellungsvorgabe: Diagramme gemäß Vorgabe auf nächster Folie plus ggf. Erklärungen/Erläuterungen.
- ◆ Der Entwurf enthält die Beschreibung aller wichtigen Schnittstellen sowie Vorgaben aus der Aufgabenstellung und beschreibt damit auch im Detail die Architektur des Systems (**Lösungs-Architektur**).
- ◆ Der Entwurf ist **vollständig in seiner Beschreibung**, d.h. für die technische Umsetzung (Implementierung) ist der Entwurf als einziges Dokument (auch für andere Teams) ausreichend
- ◆ Der Entwurf **beschreibt nicht die technische Umsetzung**, also die Art und Weise, wie die beschriebenen, Problem orientierten Algorithmen in eine Programmiersprache effizient und korrekt umgesetzt werden. Daher ist der Entwurf unabhängig von der gewählten Programmiersprache (außer den ggf. gegebenen technischen Vorgaben)

6

6

Entwurf: hier

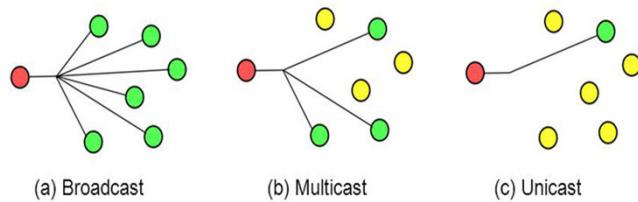
- ◆ Für die lokalen Algorithmen: Vorgabe für Ablaufdiagramme DIN 6601 (Sinnbilder und ihre Anwendung): z.B. <https://docplayer.org/77642934-Informationsverarbeitung-sinnbilder-und-ihre-anwendung.html> mit z.B. draw.io (<https://app.diagrams.net/>)
- ◆ Für die verteilten Algorithmen: Vorgabe für Sequenzdiagramme nach UML: z.B. GitMind (<https://gitmind.com/de/>), Lucidchart (<https://www.lucidchart.com/pages/>) oder StarUML (<https://staruml.io/>)
- ◆ In den Diagrammen sind Zeichen (z.B. Zahlen/Schlüsselwörter etc.) zu verwenden, um so aus dem Code heraus auf diese Stellen im Entwurf verweisen zu können.

7

7

CBCAST-Algorithmus

- ◆ Seien p_1, p_2, \dots, p_n Gruppenmitglieder
- ◆ Sei VT_j eine Vektoruhr des Gruppenmitglieds p_j
- ◆ $VT_j[i]$ stellt die Anzahl der von p_i gesendeten Multicast-Nachrichten dar, die potentiell kausal zu der letzten an p_j gelieferten Nachricht geführt haben



8

8

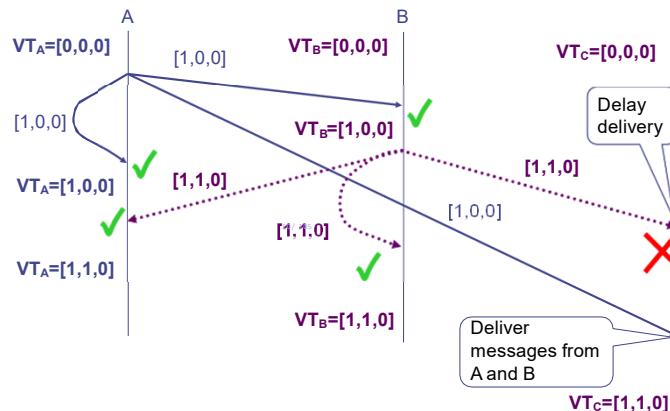
CBCAST-Algorithmus

- ◆ Initialisiere alle VT_k mit einem Nullvektor
- ◆ Wenn p_j eine Nachricht per multicast versendet
 - inkrementiere $VT_j[j]$ um eins
 - Füge aktuelles VT_j als Vektorzeitstempel vt zur Nachricht hinzu
- ◆ Nachricht ist bei p_i ausgelieferbar, wenn (sei vt der Zeitstempel der Nachricht)
 - Nachricht muss die nächste in der von p_j erwarteten Reihenfolge sein:
 $VT_i[j] + 1 = vt[j]$ (d.h. $VT_i[j] < vt[j]$)
 - Alle kausal vorhergehenden Nachrichten, die p_j zugestellt wurden, wurden auch p_i zugestellt: $VT_i[k] \geq vt[k]$ (für $k \neq j$)
- ◆ Wenn die Nachricht bei p_i zugestellt/ausgeliefert wird, synchronisiere den lokalen Zeitstempel VT_i von p_i mit dem in der Nachricht empfangenen Zeitstempel vt (jeweils das Maximum).

9

9

CBCAST-Algorithmus



10

10

Kausaler Multicast hier

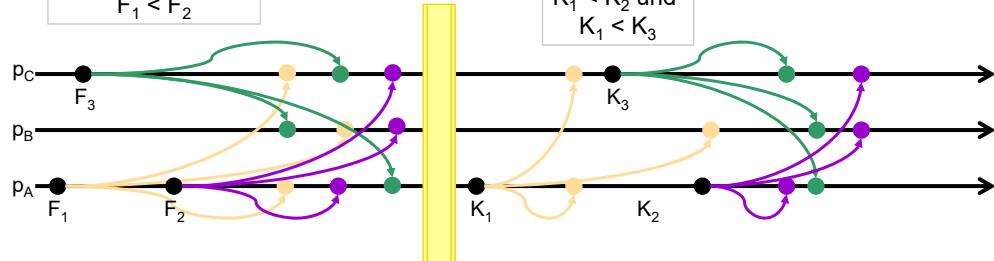
- ♦ Comm-Module (Gruppenmitglieder) müssen sich beim Multicast-Tower (towerCBC) registrieren. Dazu senden sie die Adresse ihres receivers (Dispatchers):
 $\{<\text{PID}>, \{\text{register}, <\text{PIDReceiver}>\}\}$
- ♦ Anwender kommunizieren mit dem Comm-Modul
 $(\text{received}, \text{read}, \text{send}, \text{init}, \text{stop})$.
- ♦ Das Comm-Modul benötigt eine Holdbackqueue (HBQ) und eine Deliveryqueue (DLQ).
- ♦ Der kausale Multicast wird in der HBQ sicher gestellt. Dort wird entschieden, ob die Nachricht ausgeliefert werden darf.
- ♦ Die DLQ stellt die Multicastreihenfolge dar. In der DLQ werden die Nachrichten sortiert nach Vorgabe der HBQ.
- ♦ Die lokale Vektoruhr wird bei der Auslieferung mit dem Zeitstempel der Nachricht synchronisiert.
- ♦ Ein `received/read` des Anwenders ist letztlich ein `received/read` an die DLQ. Die DLQ liefert die nächste auslieferbare Nachricht.
- ♦ Achtung: es darf kein total geordneter Multicast entstehen. Unterster zuverlässiger Multicast (towerCBC) ist ungeordnet!
- ♦ Die Nachrichten eines Anwenders (`send`) werden im eigenen Comm-Modul direkt in die DLQ eingestellt.

11

11

Kausaler Multicast hier

lokal Kausal und Nebenläufig
 $F_1 < F_2$



Sende-Ereignisse eines Prozesses sind hier kausal abhängig, daher hier eine FiFo-Reihenfolge $F_1 < F_2$!

Kausal
 $K_1 < K_2$ und
 $K_1 < K_3$

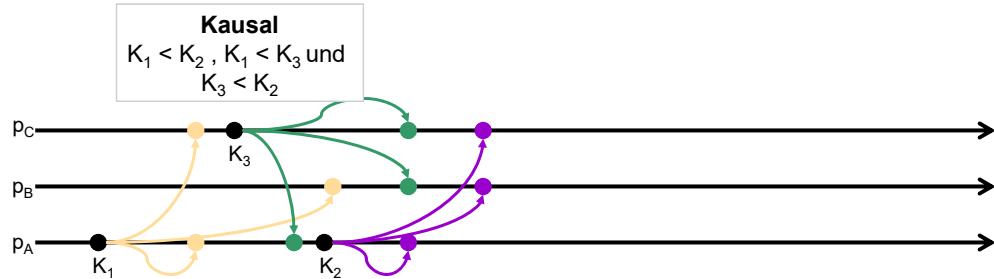
Sende-Ereignisse eines Prozesses sind hier kausal abhängig von Empfangs-Ereignissen dieses Prozesses, daher $K_1 < K_2$ und $K_1 < K_3$!

Hinweis: der zur Verfügung gestellte zuverlässige ungeordnete Multicast stellt nicht den kausalen Multicast sicher. Im Gegenteil: er liefert die Multicast-Nachrichten zuverlässig, aber möglichst ungünstig für den kausalen Multicast aus. Zu Testzwecken.

12

12

Kausaler Multicast hier



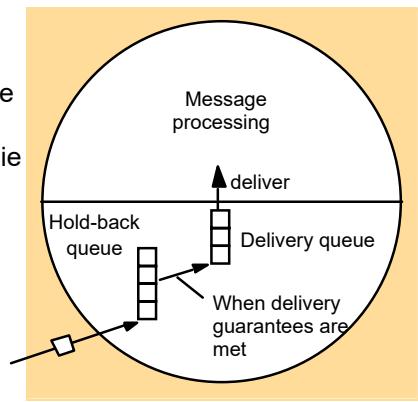
Sende-Ereignisse eines Prozesses
sind hier kausal abhängig von
Empfangs-Ereignissen dieses
Prozesses, daher $K_1 < K_2$, $K_1 < K_3$
und $K_3 < K_2$!

13

13

Implementierung

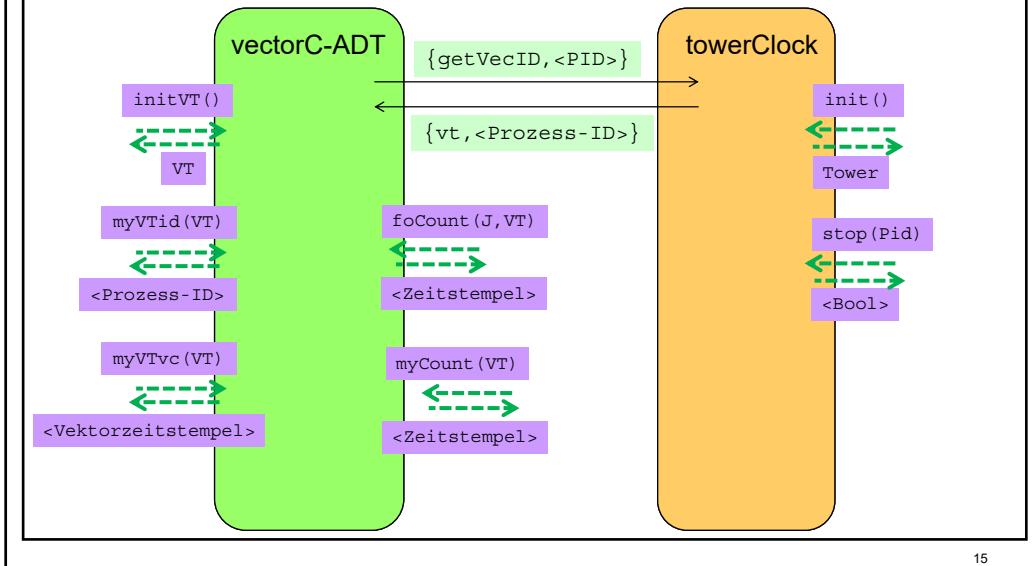
- Jede Kommunikationseinheit hat eine lokale Warteschlange für eingehende Nachrichten, geordnet nach der Vektoruhr (afterreqVTJ), die **Hold-Back Queue**. Hierin enthaltene Elemente dürfen nicht ausgeliefert werden. Sie stellt den kausalen Multicast sicher.
- Außerdem hat jede Kommunikationseinheit eine **Delivery-Queue**. Hierin enthaltene Elemente werden entsprechend ihrer Reihenfolge (Einstellungsreihenfolge durch Hold-Back Queue) an die Anwendungen ausgeliefert.
- Erst wenn die Nachricht ausgeliefert wird, ist sie bei der Anwendung bekannt und die lokale Vektoruhr kann mit dem Vektorzeitstempel der Nachricht synchronisiert werden.



14

14

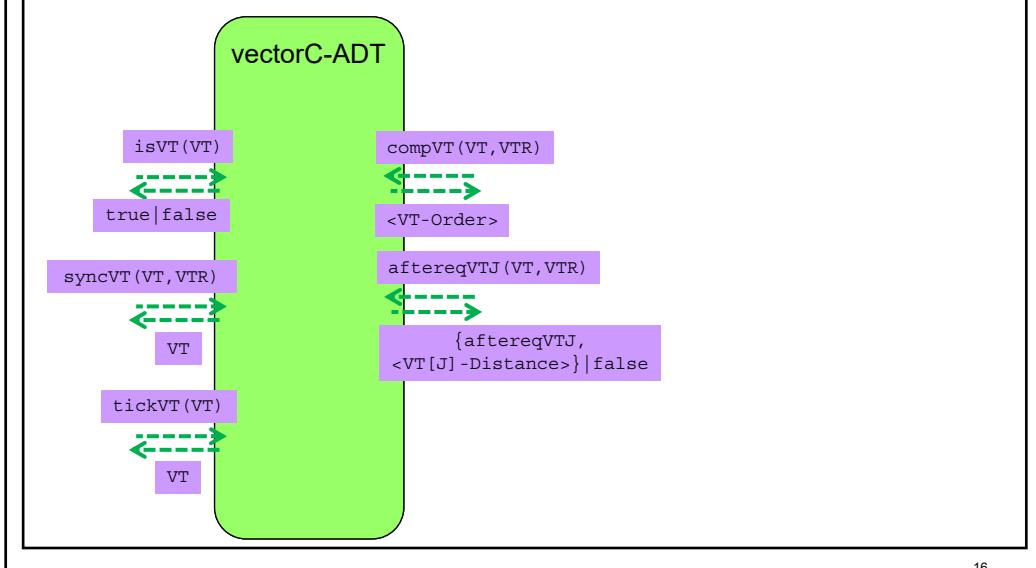
Vektoruhr ADT



15

15

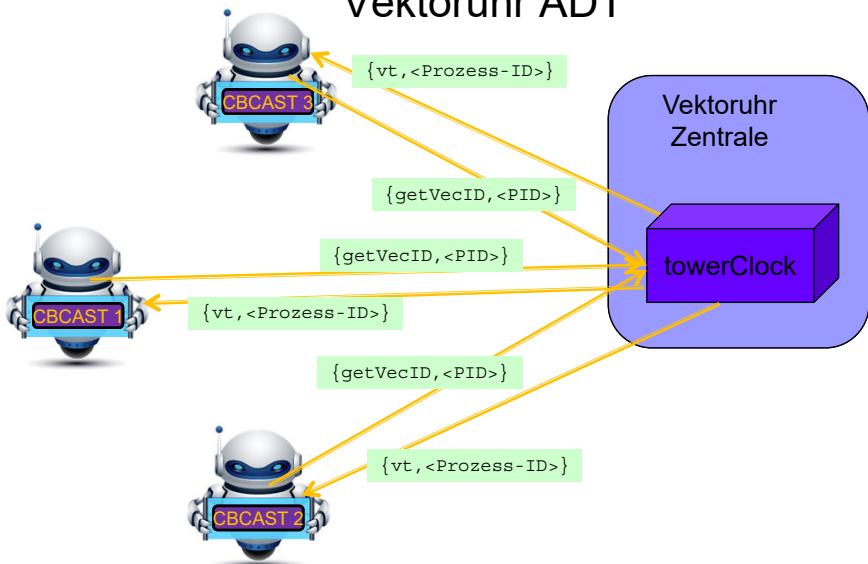
Vektoruhr ADT



16

16

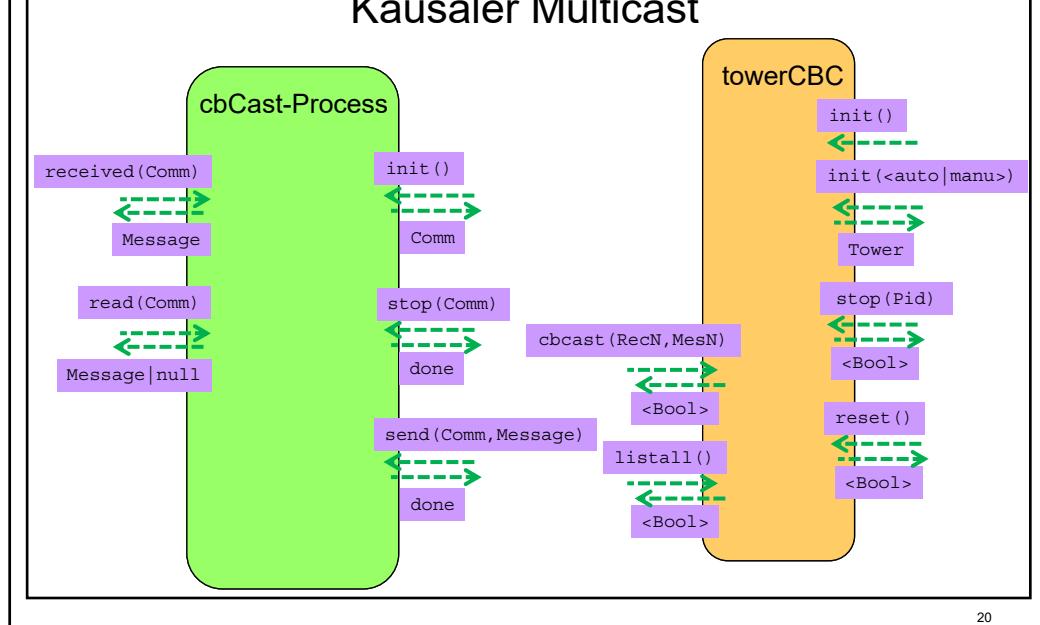
Vektoruhr ADT



17

17

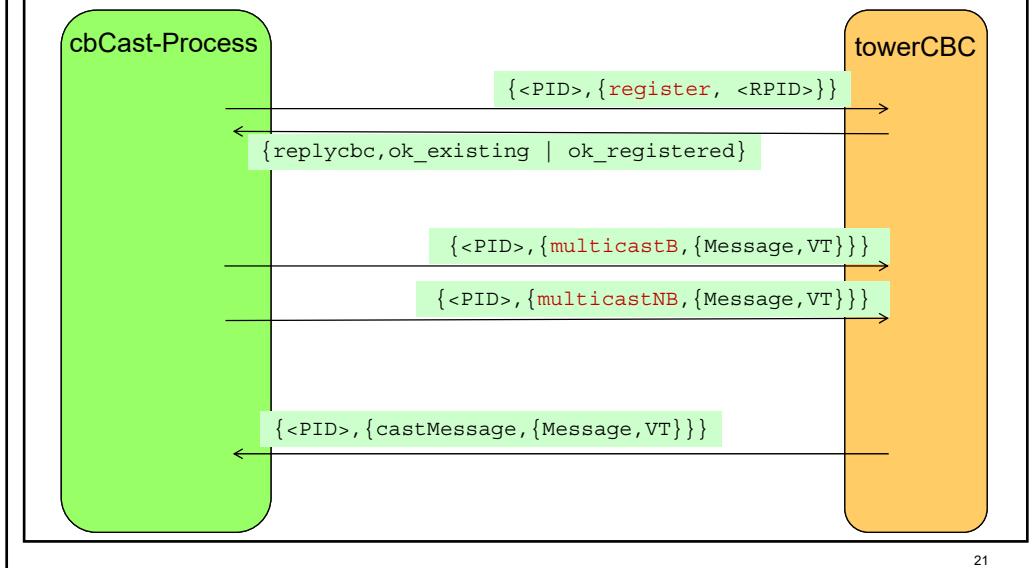
Kausaler Multicast



20

20

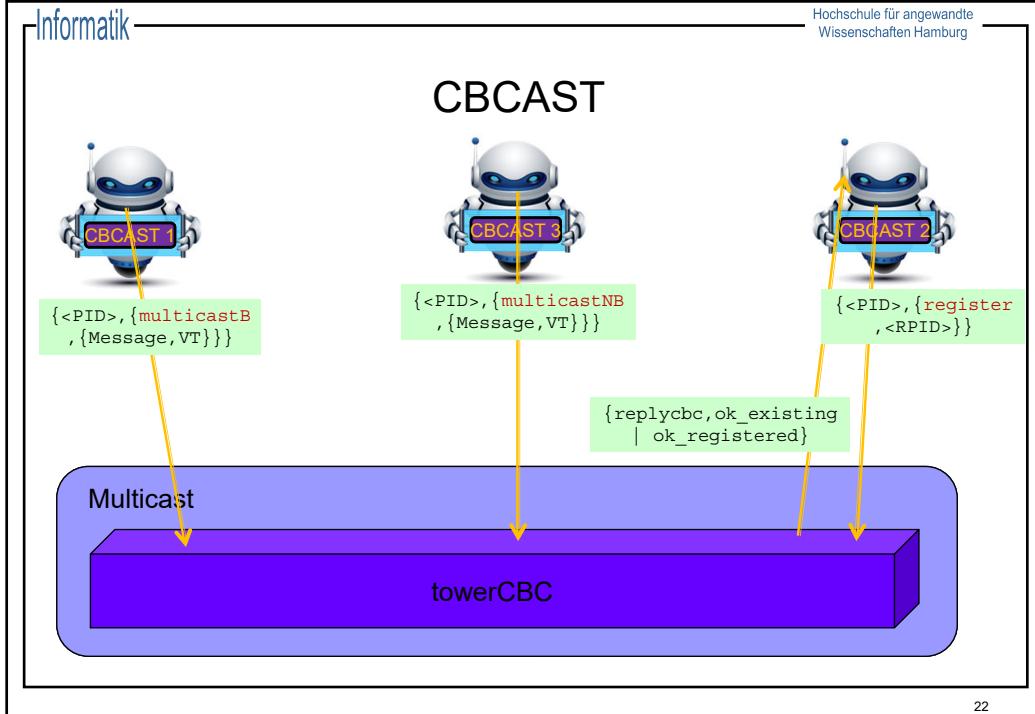
Kausaler Multicast



21

21

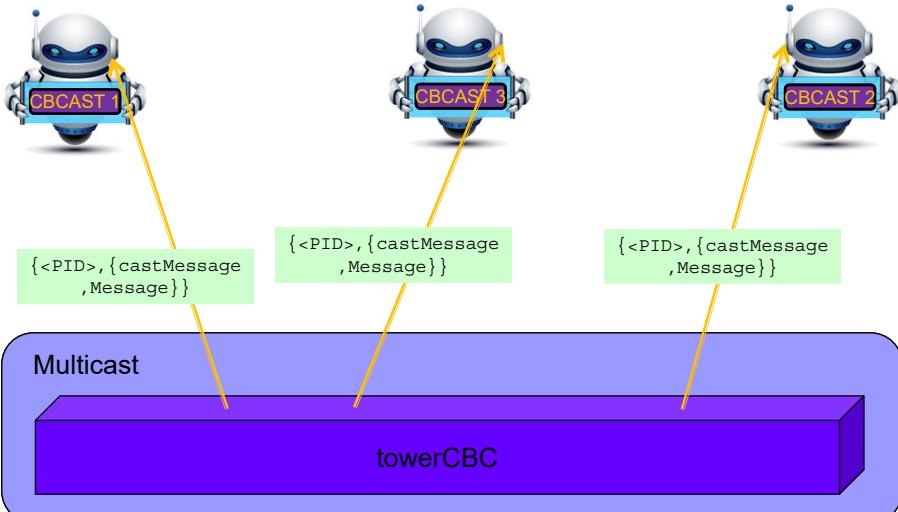
CBCAST



22

22

CBCAST



23

23

Praxistest

```
(towerCBC@Qigong-KLC1> testCBC:testADT().
testCBC.cfg erfolgreich in "<0.87.0>" gelesen.
testCBC: towerClock "vtKLCclockC" von Server "'towerClock@Qigong-KLC'" per ping eingebunden
X_2 {3,[0,0,2]}
          Soll: X_2 {[0,0,2],3}
Y_4 {5,[0,0,0,0,4]}
          Soll: Y_4 {[0,0,0,0,4],5}
Z_6 {6,[0,0,0,0,0,6]}
          Soll: Z_6 {[0,0,0,0,0,6],6}
XY {3,[0,0,2,0,4]}
          Soll: XY {[0,0,2,0,4],3}
ZY {6,[0,0,0,0,4,6]}
          Soll: ZY {[0,0,0,0,4,6],6}
X_2 {3,[0,0,2]}
          Soll: X_2 {[0,0,2],3}
Z_6 {6,[0,0,0,0,0,6]}
          Soll: Z_6 {[0,0,0,0,0,6],6}
XY concurrentVT ZY, XY afterVT Y, Y beforeVT ZY, Y equalVT Y
          Soll: XY concurrentVT ZY, XY afterVT Y, Y beforeVT ZY, Y equalVT Y
XY {aftereqVTJ,-6} ZY, XY {aftereqVTJ,0} Y, Y {aftereqVTJ,-6} ZY, Y {aftereqVTJ,0}
          Soll: XY {aftereqVTJ,-6} ZY, XY {aftereqVTJ,0} Y, Y {aftereqVTJ,-6} ZY, Y {aftereqVTJ,0} Y
ZY {aftereqVTJ,-2} XY, Y {aftereqVTJ,-2} XY, ZY {aftereqVTJ,0} Y
          Soll: ZY {aftereqVTJ,-2} XY, Y {aftereqVTJ,-2} XY, ZY {aftereqVTJ,0} Y
```

25

25

Praxistest

```

XY is Vector? true bla is Vector? false, myVId(ZY) = 6, myCount(Y) = 4
Soll: XY is Vector? true bla is Vector? false, myVId(ZY) = 6, myCount(Y) = 4
myVTvc(ZY) = [0,0,0,0,4,6], foCount(5,Y) = 4
Soll: myVTvc(ZY) = [0,0,0,0,4,6], foCount(5,Y) = 4
X2_4 {3,[0,0,4,0,4]}
Soll: X2_4 {[0,0,4,0,4],3}
Y2_8 {5,[0,0,0,0,8]}
Soll: Y2_8 {[0,0,0,0,8],5}
Z2_12 {6,[0,0,0,0,4,12]}
Soll: Z2_12 {[0,0,0,0,4,12],6}
XY2 {3,[0,0,4,0,8]}
Soll: XY2 {[0,0,4,0,8],3}
ZY2 {6,[0,0,0,0,8,12]}
Soll: ZY2 {[0,0,0,0,8,12],6}
X_2 {3,[0,0,2]}
Soll: {3,[0,0,2]} bei Vertrauen: {3,[0,0,4]}
XY2 concurrentVT ZY2, XY2 afterVT Y2, Y2 beforeVT ZY2, Y2 equalVT Y2
Soll: XY2 concurrentVT ZY2, XY2 afterVT Y2, Y2 beforeVT ZY2, Y2 equalVT Y2
XY2 {aftereqVTJ,-12} ZY2, XY2 {aftereqVTJ,0} Y2, Y2 {aftereqVTJ,-12} ZY2, Y2 {aftereqVTJ,0} Y2
Soll: XY2 {aftereqVTJ,-12} ZY2, XY2 {aftereqVTJ,0} Y2, Y2 {aftereqVTJ,-12} ZY2, Y2 {aftereqVTJ,0} Y2
ZY2 {aftereqVTJ,-4} XY2, Y2 {aftereqVTJ,-4} XY2, ZY2 {aftereqVTJ,0} Y2
Soll: ZY2 {aftereqVTJ,-4} XY2, Y2 {aftereqVTJ,-4} XY2, ZY2 {aftereqVTJ,0} Y2
ok
(towerCBC@Qigong-KLC)2>

```

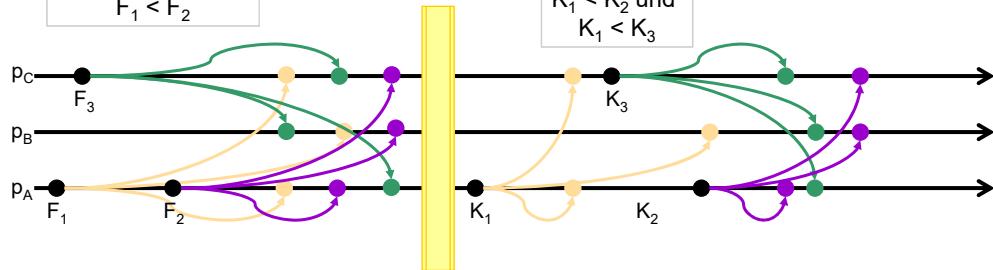
26

26

Kausaler Multicast hier

lokal Kausal und Nebenläufig
 $F_1 < F_2$

Kausal 1
 $K_1 < K_2 \text{ und } K_1 < K_3$



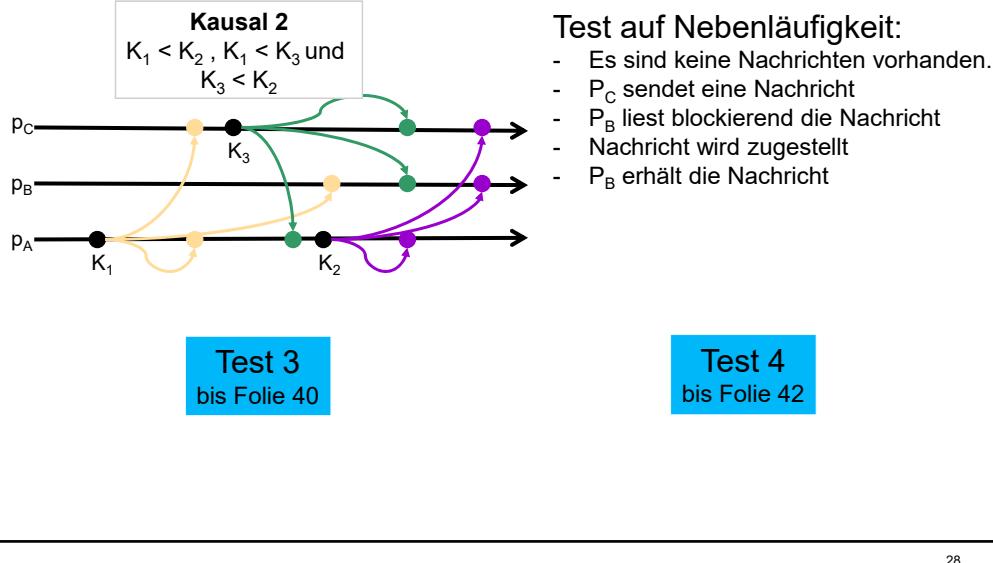
Test 1
bis Folie 29

Test 2
bis Folie 33

27

27

Kausaler Multicast hier



28

28

Praxistest

```
(towerCBC@Qigong-KLC)1> testCBC:test().
testCBC->'towerCBC@Qigong-KLC' Startzeit: 17.01 08:56:17,616|testCBC.cfg erfolgreich in <0.87.0> gelesen.
testCBC: towerClock vtKLCClockC von Server 'towerClock@Qigong-KLC' per ping eingebunden
testCBC: botA commbot von Server 'botA@Qigong-KLC' per ping eingebunden
testCBC: botB commbot von Server 'botB@Qigong-KLC' per ping eingebunden
testCBC: botC commbot von Server 'botC@Qigong-KLC' per ping eingebunden
towerCBC Startzeit: 17.01 08:56:17,909| Typ: manu mit PID <0.106.0> registriert mit Namen towerKLCCbc.
Register um 17.01 08:56:18,861| von <9576.109.0> (<9576.109.0>): Neu registriert.
towerCBC 'towerCBC@Qigong-KLC' von Server towerKLCCbc per ping eingebunden
testCBC: Kommunikationsbot commbot mit PID "<0.114.0>" auf node 'botA@Qigong-KLC' registriert
cbCast-botA@Qigong-KLC-KLC Startzeit: 17.01 08:56:18,840| mit PID <0.114.0>
Beim TowerCBC registriert:ok_registered
Register um 17.01 08:56:19,746| von <9577.109.0> (<9577.109.0>): Neu registriert.
towerCBC 'towerCBC@Qigong-KLC' von Server towerKLCCbc per ping eingebunden
testCBC: Kommunikationsbot commbot mit PID "<0.114.0>" auf node 'botB@Qigong-KLC' registriert
cbCast-botB@Qigong-KLC-KLC Startzeit: 17.01 08:56:19,737| mit PID <0.114.0>
Beim TowerCBC registriert:ok_registered
Register um 17.01 08:56:20,647| von <9578.109.0> (<9578.109.0>): Neu registriert.
towerCBC 'towerCBC@Qigong-KLC' von Server towerKLCCbc per ping eingebunden
testCBC: Kommunikationsbot commbot mit PID "<0.114.0>" auf node 'botC@Qigong-KLC' registriert
cbCast-botC@Qigong-KLC-KLC Startzeit: 17.01 08:56:20,639| mit PID <0.114.0>
Beim TowerCBC registriert:ok_registered
testCBC: TowerCB (<0.106.0>) und TowerClock (<9583.92.0>) entfernt gestartet
testCBC: BotA (<9576.109.0>), BotB (<9577.109.0>) und BotC (<9578.109.0>) entfernt gestartet
multicast manuell Nachricht 1|{"beige",<9576.120.0>} von <9576.114.0> erhalten.
DLQ>>> Nachricht "beige" von Prozess 1 mit Zeitstempel [1] in DLQ eingefügt.
multicast manuell Nachricht 2|{"gruen",<9578.120.0>} von <9578.114.0> erhalten.
DLQ>>> Nachricht "gruen" von Prozess 3 mit Zeitstempel [0,0,1] in DLQ eingefügt.
```

29

29

Praxistest

```

multicast manuell Nachricht 3>{"lila",<9576.122.0>} von <9576.114.0> erhalten.
DLQ>>> Nachricht "lila" von Prozess 1 mit Zeitstempel [2] in DLQ eingefügt.
Nachrichten 1: beige, 2: gruen, 3: lila an Multicast gesendet.
multicast manuell an 1 - <9576.114.0> mit Nachricht Nummer 3
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
multicast manuell an 2 - <9577.114.0> mit Nachricht Nummer 2
HBQ>>> Nachricht "gruen" von Prozess 3 mit Zeitstempel [0,0,1] in HBQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[0,0]}).
DLQ>>> Nachricht "gruen" von Prozess 3 mit Zeitstempel [0,0,1] in DLQ eingefügt.
HBQ>>> Vergleich {2,[0,0]} mit {3,[0,0,1]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
multicast manuell an 3 - <9578.114.0> mit Nachricht Nummer 1
HBQ>>> Nachricht "beige" von Prozess 1 mit Zeitstempel [1] in HBQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[0,0,1]}).
DLQ>>> Nachricht "beige" von Prozess 1 mit Zeitstempel [1] in DLQ eingefügt.
HBQ>>> Vergleich {3,[0,0,1]} mit {1,[1]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
multicast manuell an 3 - <9578.114.0> mit Nachricht Nummer 2
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[0,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
Nachrichten per Multicast zugestellt: A lila; B gruen; C beige,gruen,lila.

```

30

30

Praxistest

```

HBQ>>> Nachricht "lila" von Prozess 1 mit Zeitstempel [2] in HBQ eingefügt.
multicast manuell an 3 - <9578.114.0> mit Nachricht Nummer 3
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[0,0,1]}).
HBQ>>> Vergleich {3,[0,0,1]} mit {1,[2]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -2. Nachricht bleibt in HBQ
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[0,0]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[0,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Vergleich {3,[0,0,1]} mit {1,[2]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -2. Nachricht bleibt in HBQ
DLQ>>> Anfrage nach nicht leer erhalten: true
    >>>Empfange Nachrichten: A "beige" (beige); B "gruen" (gruen); C "gruen" (gruen);
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[0,0]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "beige" von Prozess 1 mit Zeitstempel [1] aus DLQ gelöscht (blockierend).
DLQ>>> Nachricht "gruen" von Prozess 3 mit Zeitstempel [0,0,1] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: true
DLQ>>> Anfrage nach nicht leer erhalten: true
DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[0,0,1]}).

```

31

31

Praxistest

```

HBQ>>> Vergleich {3,[0,0,1]} mit {1,[2]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -2. Nachricht bleibt in HBQ
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "gruen" von Prozess 3 mit Zeitstempel [0,0,1] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: true
multicast manuell an 1 - <9576.114.0> mit Nachricht Nummer 1
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
Nachrichten per Multicast zugestellt: A beige; B lila.
HBQ>>> Nachricht "lila" von Prozess 1 mit Zeitstempel [2] in HBQ eingefügt.
multicast manuell an 2 - <9577.114.0> mit Nachricht Nummer 3
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[0,0,1]}).
HBQ>>> Vergleich {2,[0,0,1]} mit {1,[2]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -2. Nachricht bleibt in HBQ
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[0,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[0,0,1]}).
HBQ>>> Vergleich {2,[0,0,1]} mit {1,[2]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -2. Nachricht bleibt in HBQ
DLQ>>> Anfrage nach nicht leer erhalten: true
    >>>Empfange Nachrichten: A "lila" (lila); B null (null); C "beige" (beige);
HBQ>>> Vergleich {3,[0,0,1]} mit {1,[2]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -2. Nachricht bleibt in HBQ
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[2]}).

```

32

32

Praxistest

```

HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> null Nachricht aus DLQ gelesen (nicht blockierend).
DLQ>>> Nachricht "lila" von Prozess 1 mit Zeitstempel [2] aus DLQ gelöscht (blockierend).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[0,0,1]}).
DLQ>>> Anfrage nach nicht leer erhalten: false
DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Vergleich {3,[0,0,1]} mit {1,[2]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -2. Nachricht bleibt in HBQ
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "beige" von Prozess 1 mit Zeitstempel [1] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: false
multicast manuell an 1 - <9576.114.0> mit Nachricht Nummer 2
HBQ>>> Nachricht "gruen" von Prozess 3 mit Zeitstempel [0,0,1] in HBQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[2]}).
DLQ>>> Nachricht "gruen" von Prozess 3 mit Zeitstempel [0,0,1] in DLQ eingefügt.
HBQ>>> Vergleich {1,[2]} mit {3,[0,0,1]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
Nachrichten per Multicast zugestellt: A gruen; B beige.
multicast manuell an 2 - <9577.114.0> mit Nachricht Nummer 1
HBQ>>> Nachricht "beige" von Prozess 1 mit Zeitstempel [1] in HBQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[0,0,1]}).
DLQ>>> Nachricht "beige" von Prozess 1 mit Zeitstempel [1] in DLQ eingefügt.
HBQ>>> Vergleich {2,[0,0,1]} mit {1,[1]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben

```

33

33

Praxistest

```

HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[0,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Vergleich {2,[0,0,1]} mit {1,[2]}.
HBQ>>> Ergebnis aftereqV1J mit Distanz -2. Nachricht bleibt in HBQ
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[1,0,1]}).
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "lila" von Prozess 1 mit Zeitstempel [2] in DLQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[2]}).
    >>>Empfange Nachrichten: A "gruen" (gruen); B "beige" (beige)|"lila" (lila); C "lila" (lila);
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Vergleich {3,[1,0,1]} mit {1,[2]}.
HBQ>>> Ergebnis aftereqV1J mit Distanz -1. Nachricht wird in DLQ verschoben
DLQ>>> Nachricht "gruen" von Prozess 3 mit Zeitstempel [0,0,1] aus DLQ gelöscht (blockierend).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[0,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: false
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Vergleich {2,[0,0,1]} mit {1,[2]}.
HBQ>>> Ergebnis aftereqV1J mit Distanz -2. Nachricht bleibt in HBQ
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[1,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "beige" von Prozess 1 mit Zeitstempel [1] aus DLQ gelöscht (blockierend).

```

34

34

Praxistest

```

DLQ>>> Nachricht "lila" von Prozess 1 mit Zeitstempel [2] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: false
DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[1,0,1]}).
DLQ>>> Nachricht "lila" von Prozess 1 mit Zeitstempel [2] in DLQ eingefügt.
HBQ>>> Vergleich {2,[1,0,1]} mit {1,[2]}.
HBQ>>> Ergebnis aftereqV1J mit Distanz -1. Nachricht wird in DLQ verschoben
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[1,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "lila" von Prozess 1 mit Zeitstempel [2] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: false
Test für lokal Kausal und Nebenläufig beendet.

DLQ>>> Nachricht "beigeK1" von Prozess 1 mit Zeitstempel [3,0,1] in DLQ eingefügt.
Nachrichte 4: beigeK1 an Multicast gesendet.
multicast manuell Nachricht 4|{"beigeK1",<9576.114.0>} von <9576.114.0>erhalten.
multicast manuell an 1 - <9576.114.0> mit Nachricht Nummer 4
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[3,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
Nachrichten per Multicast zugestellt: A beigeK1; C beigeK1.
HBQ>>> Nachricht "beigeK1" von Prozess 1 mit Zeitstempel [3,0,1] in HBQ eingefügt.
multicast manuell an 3 - <9578.114.0> mit Nachricht Nummer 4
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[2,0,1]}).
DLQ>>> Nachricht "beigeK1" von Prozess 1 mit Zeitstempel [3,0,1] in DLQ eingefügt.
HBQ>>> Vergleich {3,[2,0,1]} mit {1,[3,0,1]}.
HBQ>>> Ergebnis aftereqV1J mit Distanz -1. Nachricht wird in DLQ verschoben

```

35

35

Praxistest

```

HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[3,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[2,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[2,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
    >>>Empfange Nachrichten: A "beigeK1" (beigeK1); B null (null); C "beigeK1" (beigeK1);
DLQ>>> Anfrage nach nicht leer erhalten: true
DLQ>>> null Nachricht aus DLQ gelesen (nicht blockierend).
multicast manuell Nachricht 5{"gruenK1",<9578.173.0>} von <9578.114.0>erhalten.
DLQ>>> Anfrage nach nicht leer erhalten: true
Nachrichten 5: gruenK1 an Multicast gesendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[3,0,1]}).
DLQ>>> Anfrage nach nicht leer erhalten: false
multicast manuell Nachricht 6{"lilaK1",<9576.168.0>} von <9576.114.0>erhalten.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[2,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
Nachrichten 6: lilaK1 an Multicast gesendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "beigeK1" von Prozess 1 mit Zeitstempel [3,0,1] aus DLQ gelöscht (blockierend).
DLQ>>> Nachricht "beigeK1" von Prozess 1 mit Zeitstempel [3,0,1] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: false
DLQ>>> Anfrage nach nicht leer erhalten: false
DLQ>>> Nachricht "gruenK1" von Prozess 3 mit Zeitstempel [3,0,2] in DLQ eingefügt.
DLQ>>> Nachricht "lilaK1" von Prozess 1 mit Zeitstempel [4,0,1] in DLQ eingefügt.
multicast manuell an 1 - <9576.114.0> mit Nachricht Nummer 6
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[4,0,1]}).

```

36

36

Praxistest

```

HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
multicast manuell an 2 - <9577.114.0> mit Nachricht Nummer 6
HBQ>>> Nachricht "lilaK1" von Prozess 1 mit Zeitstempel [4,0,1] in HBQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[2,0,1]}).
HBQ>>> Vergleich {2,[2,0,1]} mit {1,[4,0,1]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -2. Nachricht bleibt in HBQ
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: false
multicast manuell an 3 - <9578.114.0> mit Nachricht Nummer 6
HBQ>>> Nachricht "lilaK1" von Prozess 1 mit Zeitstempel [4,0,1] in HBQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[3,0,2]}).
DLQ>>> Nachricht "lilaK1" von Prozess 1 mit Zeitstempel [4,0,1] in DLQ eingefügt.
HBQ>>> Vergleich {3,[3,0,2]} mit {1,[4,0,1]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
Nachrichten per Multicast zugestellt: A lilaK1; B lilaK1; C lilaK1.
multicast manuell an 1 - <9576.114.0> mit Nachricht Nummer 5
HBQ>>> Nachricht "gruenK1" von Prozess 3 mit Zeitstempel [3,0,2] in HBQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[4,0,1]}).
DLQ>>> Nachricht "gruenK1" von Prozess 3 mit Zeitstempel [3,0,2] in DLQ eingefügt.
HBQ>>> Vergleich {1,[4,0,1]} mit {3,[3,0,2]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
multicast manuell an 2 - <9577.114.0> mit Nachricht Nummer 5
HBQ>>> Nachricht "gruenK1" von Prozess 3 mit Zeitstempel [3,0,2] in HBQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[2,0,1]}).

```

37

37

Praxistest

```

HBQ>>> Vergleich {2,[2,0,1]} mit {1,[4,0,1]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -2. Nachricht bleibt in HBQ
HBQ>>> Vergleich {2,[2,0,1]} mit {3,[3,0,2]}.
HBQ>>> Ergebnis nicht aftereqVTJ.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: false
multicast manuell an 2 - <9577.114.0> mit Nachricht Nummer 4
HBQ>>> Nachricht "beigeK1" von Prozess 1 mit Zeitstempel [3,0,1] in HBQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[2,0,1]}).
DLQ>>> Nachricht "beigeK1" von Prozess 1 mit Zeitstempel [3,0,1] in DLQ eingefügt.
HBQ>>> Vergleich {2,[2,0,1]} mit {1,[3,0,1]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben
HBQ>>> Vergleich {2,[2,0,1]} mit {3,[3,0,2]}.
HBQ>>> Ergebnis nicht aftereqVTJ.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
Nachrichten per Multicast zugestellt: A gruenK1; B gruenK1,beigeK1; C gruenK1.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[3,0,2]}).
multicast manuell an 3 - <9578.114.0> mit Nachricht Nummer 5
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[4,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[2,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Vergleich {2,[2,0,1]} mit {1,[4,0,1]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -2. Nachricht bleibt in HBQ
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[4,0,1]}).
HBQ>>> Vergleich {2,[2,0,1]} mit {3,[3,0,2]}.

```

38

38

Praxistest

```

HBQ>>> Ergebnis nicht aftereqVTJ.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[3,0,2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
    >>>Empfange Nachrichten: A "lilaK1" (gruenK1); B "beigeK1" (beigeK1)|"lilaK1" (lilaK1); C "gruenK1" (gruenK1);
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "lilaK1" von Prozess 1 mit Zeitstempel [4,0,1] aus DLQ gelöscht (blockierend).
    >>>Empfange Nachrichten: A "gruenK1" (lilaK1); B "gruenK1" (gruenK1); C "lilaK1" (lilaK1);
DLQ>>> Anfrage nach nicht leer erhalten: true
DLQ>>> Anfrage nach nicht leer erhalten: true
DLQ>>> Anfrage nach nicht leer erhalten: true
Test Kausal 1 beendet.

HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[2,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[3,0,2]}).
    >>>Empfange Nachrichten: A null (null); B null (null); C null (null);
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[4,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Vergleich {2,[2,0,1]} mit {1,[4,0,1]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -2. Nachricht bleibt in HBQ
multicast manuell Nachricht 7|{"beigeK2",<9576.191.0>} von <9576.114.0>erhalten.
DLQ>>> Nachricht "gruenK1" von Prozess 3 mit Zeitstempel [3,0,2] aus DLQ gelöscht (blockierend).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Vergleich {2,[2,0,1]} mit {3,[3,0,2]}.
HBQ>>> Ergebnis nicht aftereqVTJ.
Nachrichte 7: beigeK2 an Multicast gesendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[4,0,1]}).

```

39

39

Praxistest

```

HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[3,0,2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "gruenK1" von Prozess 3 mit Zeitstempel [3,0,2] aus DLQ gelöscht (blockierend).
DLQ>>> Nachricht "beigeK1" von Prozess 1 mit Zeitstempel [3,0,1] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: true
DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[4,0,2]}).
DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[3,0,2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[3,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "lilaK1" von Prozess 1 mit Zeitstempel [4,0,1] aus DLQ gelöscht (blockierend).
DLQ>>> Nachricht "lilaK1" von Prozess 1 mit Zeitstempel [4,0,1] in DLQ eingefügt.
DLQ>>> null Nachricht aus DLQ gelesen (nicht blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[4,0,2]}).
HBQ>>> Vergleich {2,[3,0,1]} mit {1,[4,0,1]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben
DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "beigeK2" von Prozess 1 mit Zeitstempel [5,0,2] in DLQ eingefügt.
DLQ>>> Nachricht "gruenK1" von Prozess 3 mit Zeitstempel [3,0,2] in DLQ eingefügt.
DLQ>>> null Nachricht aus DLQ gelesen (nicht blockierend).
HBQ>>> Vergleich {2,[3,0,1]} mit {3,[3,0,2]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben
DLQ>>> Anfrage nach nicht leer erhalten: false

```

40

40

Praxistest

```

HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[3,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "lilaK1" von Prozess 1 mit Zeitstempel [4,0,1] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[4,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[4,0,1]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "gruenK1" von Prozess 3 mit Zeitstempel [3,0,2] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[4,0,2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> null Nachricht aus DLQ gelesen (nicht blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: false
multicast manuell an 1 - <9576.114.0> mit Nachricht Nummer 7
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[5,0,2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
Nachrichten per Multicast zugestellt: A beigeK2; C beigeK2.
HBQ>>> Nachricht "beigeK2" von Prozess 1 mit Zeitstempel [5,0,2] in HBQ eingefügt.
multicast manuell an 3 - <9578.114.0> mit Nachricht Nummer 7
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[4,0,2]}).
DLQ>>> Nachricht "beigeK2" von Prozess 1 mit Zeitstempel [5,0,2] in DLQ eingefügt.
HBQ>>> Vergleich {3,[4,0,2]} mit {1,[5,0,2]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.

```

41

41

Praxistest

```

DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[5,0,2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[4,0,2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[4,0,2]}).
    >>>Empfange Nachrichten: A "beigeK2" (beigeK2); B null (null); C "beigeK2" (beigeK2);
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
multicast manuell Nachricht 8>{"gruenK2",<9578.218.0>} von <9578.114.0>erhalten.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
Nachrichten 8: gruenK2 an Multicast gesendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[4,0,2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[5,0,2]}).
DLQ>>> null Nachricht aus DLQ gelesen (nicht blockierend).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: false
DLQ>>> Nachricht "beigeK2" von Prozess 1 mit Zeitstempel [5,0,2] aus DLQ gelöscht (blockierend).
DLQ>>> Nachricht "beigeK2" von Prozess 1 mit Zeitstempel [5,0,2] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: false
DLQ>>> Anfrage nach nicht leer erhalten: false
DLQ>>> Nachricht "gruenK2" von Prozess 3 mit Zeitstempel [5,0,3] in DLQ eingefügt.
Nachrichten per Multicast zugestellt: A gruenK2.
multicast manuell an 1 - <9576.114.0> mit Nachricht Nummer 8
HBQ>>> Nachricht "gruenK2" von Prozess 3 mit Zeitstempel [5,0,3] in HBQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[5,0,2]}).
DLQ>>> Nachricht "gruenK2" von Prozess 3 mit Zeitstempel [5,0,3] in DLQ eingefügt.
HBQ>>> Vergleich {1,[5,0,2]} mit {3,[5,0,3]}.

```

42

42

Praxistest

```

HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[5,0,2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[4,0,2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[5,0,3]}).
    >>>Empfange Nachrichten: A "gruenK2" (gruenK2); B null (null); C "gruenK2" (gruenK2);
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
multicast manuell Nachricht 9>{"lilaK2",<9576.225.0>} von <9576.114.0>erhalten.
DLQ>>> null Nachricht aus DLQ gelesen (nicht blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[5,0,2]}).
Nachrichten 9: lilaK2 an Multicast gesendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[5,0,3]}).
DLQ>>> Anfrage nach leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "gruenK2" von Prozess 3 mit Zeitstempel [5,0,3] aus DLQ gelöscht (blockierend).
DLQ>>> Nachricht "gruenK2" von Prozess 3 mit Zeitstempel [5,0,3] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: false
DLQ>>> Anfrage nach nicht leer erhalten: false
DLQ>>> Nachricht "lilaK2" von Prozess 1 mit Zeitstempel [6,0,3] in DLQ eingefügt.
multicast manuell an 1 - <9576.114.0> mit Nachricht Nummer 9
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[6,0,3]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true

```

43

43

Praxistest

```

multicast manuell an 2 - <9577.114.0> mit Nachricht Nummer 9
HBQ>>> Nachricht "lilaK2" von Prozess 1 mit Zeitstempel [6,0,3] in HBQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[4,0,2]}).
HBQ>>> Vergleich {2,[4,0,2]} mit {1,[6,0,3]}.
HBQ>>> Ergebnis nicht aftereqVTJ.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: false
multicast manuell an 2 - <9577.114.0> mit Nachricht Nummer 8
HBQ>>> Nachricht "gruenK2" von Prozess 3 mit Zeitstempel [5,0,3] in HBQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[4,0,2]}).
HBQ>>> Vergleich {2,[4,0,2]} mit {1,[6,0,3]}.
HBQ>>> Ergebnis nicht aftereqVTJ.
HBQ>>> Vergleich {2,[4,0,2]} mit {3,[5,0,3]}.
HBQ>>> Ergebnis nicht aftereqVTJ.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: false
multicast manuell an 2 - <9577.114.0> mit Nachricht Nummer 7
HBQ>>> Nachricht "beigeK2" von Prozess 1 mit Zeitstempel [5,0,2] in HBQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[4,0,2]}).
DLQ>>> Nachricht "beigeK2" von Prozess 1 mit Zeitstempel [5,0,2] in DLQ eingefügt.
HBQ>>> Vergleich {2,[4,0,2]} mit {1,[5,0,2]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben
HBQ>>> Vergleich {2,[4,0,2]} mit {3,[5,0,3]}.
HBQ>>> Ergebnis nicht aftereqVTJ.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
multicast manuell an 3 - <9578.114.0> mit Nachricht Nummer 9
HBQ>>> Nachricht "lilaK2" von Prozess 1 mit Zeitstempel [6,0,3] in HBQ eingefügt.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[5,0,3]}).

```

44

44

Praxistest

```

DLQ>>> Nachricht "lilaK2" von Prozess 1 mit Zeitstempel [6,0,3] in DLQ eingefügt.
HBQ>>> Vergleich {3,[5,0,3]} mit {1,[6,0,3]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
Nachrichten per Multicast zugestellt: A lilaK2; B lilaK2,gruenK2,beigeK2; C lilaK2,gruenK2.
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[5,0,3]}).
multicast manuell an 3 - <9578.114.0> mit Nachricht Nummer 8
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[6,0,3]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[4,0,2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Vergleich {2,[4,0,2]} mit {1,[6,0,3]}.
HBQ>>> Ergebnis nicht aftereqVTJ.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[6,0,3]}).
HBQ>>> Vergleich {2,[4,0,2]} mit {3,[5,0,3]}.
HBQ>>> Ergebnis nicht aftereqVTJ.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "lilaK2" von Prozess 1 mit Zeitstempel [6,0,3] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[5,0,3]}).
DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[4,0,2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
    >>>Empfange Nachrichten: A "lilaK2" (lilaK2); B "beigeK2" (beigeK2)| "gruenK2" (gruenK2)| "lilaK2" (lilaK2);

```

45

45

Praxistest

```

HBQ>>> Prüfung auf auslieferbare Nachrichten ({1,[6,0,3]}).
HBQ>>> Vergleich {2,[4,0,2]} mit {1,[6,0,3]}.
HBQ>>> Ergebnis nicht aftereqVTJ.
DLQ>>> Anfrage nach nicht leer erhalten: true
Test Kausal 2 beendet.

HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Vergleich {2,[4,0,2]} mit {3,[5,0,3]}.
HBQ>>> Ergebnis nicht aftereqVTJ.
    >>>Empfange Nachrichten: A null (null); B null (null); C null (null);
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[5,0,3]}).
DLQ>>> null Nachricht aus DLQ gelesen (nicht blockierend).
multicast manuell Nachricht 10>{"krimskrams",<9578.246.0>} von <9578.114.0>erhalten.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: false
DLQ>>> Nachricht "lilaK2" von Prozess 1 mit Zeitstempel [6,0,3] aus DLQ gelöscht (blockierend).
DLQ>>> Nachricht "beigeK2" von Prozess 1 mit Zeitstempel [5,0,2] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: false
DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[5,0,2]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten ({3,[6,0,3]}).
HBQ>>> Vergleich {2,[5,0,2]} mit {1,[6,0,3]}.
HBQ>>> Ergebnis nicht aftereqVTJ.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "gruenK2" von Prozess 3 mit Zeitstempel [5,0,3] in DLQ eingefügt.
DLQ>>> null Nachricht aus DLQ gelesen (nicht blockierend).
HBQ>>> Vergleich {2,[5,0,2]} mit {3,[5,0,3]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben

```

46

46

Praxistest

```

DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "krimskrams" von Prozess 3 mit Zeitstempel [6,0,4] in DLQ eingefügt.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[5,0,2]}).
HBQ>>> Vergleich {2,[5,0,2]} mit {1,[6,0,3]}.
HBQ>>> Ergebnis nicht aftereqVTJ.
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "gruenK2" von Prozess 3 mit Zeitstempel [5,0,3] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[5,0,3]}).
DLQ>>> Nachricht "lilaK2" von Prozess 1 mit Zeitstempel [6,0,3] in DLQ eingefügt.
HBQ>>> Vergleich {2,[5,0,3]} mit {1,[6,0,3]}.
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: true
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[5,0,3]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Nachricht "lilaK2" von Prozess 1 mit Zeitstempel [6,0,3] aus DLQ gelöscht (blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[6,0,3]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> null Nachricht aus DLQ gelesen (nicht blockierend).
DLQ>>> Anfrage nach nicht leer erhalten: false
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[6,0,3]}).
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.
DLQ>>> Anfrage nach nicht leer erhalten: false
test auf Nebenläufigkeit. Auf 'botB@Qigong-KLC' folgenden Aufruf durchgeführt: cbCast:received({commbot,'botB@Qigong-KLC'}).
HBQ>>> Nachricht "krimskrams" von Prozess 3 mit Zeitstempel [6,0,4] in HBQ eingefügt.

```

47

47

Praxistest

```
Nachricht per Multicast zugestellt: B krimskrams.  
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[6,0,3]}).  
  
>>>     >>>     >>> Nebenläufigkeit erfolgreich: "krimskrams"  
  
multicast manuell an 2 - <9577.114.0> mit Nachricht Nummer 10  
DLQ>>> Nachricht "krimskrams" von Prozess 3 mit Zeitstempel [6,0,4] in DLQ eingefügt.  
HBQ>>> Vergleich {2,[6,0,3]} mit {3,[6,0,4]}.  
HBQ>>> Ergebnis aftereqVTJ mit Distanz -1. Nachricht wird in DLQ verschoben  
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.  
DLQ>>> Anfrage nach nicht leer erhalten: true  
HBQ>>> Prüfung auf auslieferbare Nachrichten ({2,[6,0,3]}).  
HBQ>>> Prüfung auf auslieferbare Nachrichten beendet.  
DLQ>>> Nachricht "krimskrams" von Prozess 3 mit Zeitstempel [6,0,4] aus DLQ gelöscht (blockierend).  
DLQ>>> Anfrage nach nicht leer erhalten: false  
Test Nebenläufigkeit beendet.  
kill um 17.01 08:57:44,946| von <0.87.0>.  
true  
(towerCBC@Qigong-KLC)2>
```

48

Aufgabe Hausarbeit

In dieser Hausarbeit ist eine einfache **Infrastruktur für einen kausalen Multicast mittels Vektoruhr** inklusiver einer prototypischen kleinen Anwendung zu implementieren. Die zu verwendende Programmiersprache ist [Erlang OTP](#). Für den Multicast werden Kommunikationseinheiten verwendet, die eine Kommunikationsgruppe bilden. Alle Nachrichten innerhalb der Gruppe werden per kausalem Multicast versendet. Dazu wird eine Vektoruhr verwendet. In den Kommunikationseinheiten wird jeweils eine Holdbackqueue und eine Deliveryqueue benötigt. Der kausale Multicast wird in der Holdbackqueue sicher gestellt. Dort wird entschieden, ob die Nachricht ausgeliefert werden darf. Die Deliveryqueue stellt die Multicastreihenfolge dar.

Die Verwendung von komplexeren Datenstrukturen von Erlang OTP ist untersagt: Etwa die Verwendung von z.B. `dict` oder `sets` ist nicht zulässig. Lediglich die Basis-Strukturen Liste (`lists`) und Tupel (`tuple`) dürfen eingesetzt werden. Algorithmen auf diesen Basisstrukturen müssen aber selbst implementiert werden.

Lesen Sie sich die Aufgabenstellung sorgfältig durch, damit nicht auf Grund einer Nachlässigkeit die erfolgreiche Bearbeitung gefährdet ist!

Aufgabenstellung

Basis ist der Algorithmus für den kausalen Multicast:

Seien p_1, p_2, \dots, p_n Gruppenmitglieder

Sei VT_j eine Vektoruhr des Gruppenmitglieds p_j

$VT_i[j]$ stellt die Anzahl der von p_i gesendeten Multicast-Nachrichten dar, die potentiell kausal zu der letzten an p_j gelieferten Nachricht geführt haben

1. Initialisiere alle VT_i mit einem Nullvektor
2. Wenn p_j eine Nachricht per multicast versendet
 - inkrementiere $VT_j[j]$ um eins
 - Füge aktuelles VT_j als Vektorzeitstempel vt zur Nachricht hinzu
3. Nachricht ist bei p_i auslieferbar, wenn (sei vt der Zeitstempel der Nachricht)
 - Nachricht muss die nächste in der von p_j erwarteten Reihenfolge sein: $VT_i[j]+1 = vt[j]$ (,d.h. $VT_i[j] < vt[j]$)
 - Alle kausal vorhergehenden Nachrichten, die p_j zugestellt wurden, wurden auch p_i zugestellt: $VT_i[k] \geq vt[k]$ (für $k \neq j$)
4. Wenn die Nachricht bei p_i zugestellt/ausgeliefert wird, synchronisiere den lokalen Zeitstempel VT_i von p_i mit dem in der Nachricht empfangenen Zeitstempel vt (jeweils das Maximum)

Achtung: Zu beachten ist, dass der Vektorzeitstempel vt aus der Nachricht und der lokale Vektorzeitstempel VT aus der allgemeinen Sicht der Vektoruhr nebenläufig sind! Daraus resultiert die besondere Anforderung an die ADT: `aftereqVTJ`. Bis auf die Position J soll VT nach oder gleich vt sein ($VT_i[k] \geq vt[k]$ (für $k \neq j$)). An der Position J gilt $VT_i[j] < vt[j]$.

Das System besteht im Wesentlichen aus den drei Komponenten Kommunikationseinheit, Multicastsender (towerCBC) und der Vektoruhr-ADT zusammen mit ihrer Vektoruhrzentrale (towerClock).

Die Kommunikationseinheit: In der Kommunikationseinheit werden die benötigten Funktionen für die Kommunikation zur Verfügung gestellt. Damit hier ein Austausch gewährleistet wird, ist die Datei als cbCast.erl zu bezeichnen. Folgende Festlegungen sind einzuhalten:

- Die **Nachricht** eines Anwenders ist eine Zeichenkette.
- Nachfolgende **Schnittstellen** sind anzubieten:

1. init(): erstellt einen Prozess für die Kommunikationseinheit. Rückgabe ist ihre PID.
2. stop(Comm): terminiert die Kommunikationseinheit mit PID comm. Rückgabe ist done.
3. send(Comm,Message): sendet eine Nachricht Message über die Kommunikationseinheit Comm als kausaler Multicast an alle anderen in der Gruppe.
4. received(Comm): empfängt blockierend eine Nachricht von der Kommunikationseinheit Comm. Die Rückgabe ist eine Zeichenkette.
5. read(Comm): empfängt nicht blockierend eine Nachricht von der Kommunikationseinheit Comm. Wenn keine Nachricht vorhanden ist, wird null zurück gegeben, sonst eine Zeichenkette.
6. {<PID>, {castMessage, {<Message>, <VT>}}}: als Nachricht. Empfängt die Nachricht <Message> mit Vektorzeitstempel <VT>. <PID> wird nicht benötigt.

Die Vektoruhr-ADT: In der ADT werden die benötigten Funktionen für die Vektoruhr zur Verfügung gestellt. Damit hier ein Austausch der ADT gewährleistet wird, ist die Datei als vectorC.erl zu bezeichnen. Die Vektoruhr wird bei den Anwendern durch den Vektorzeitstempel VT realisiert. Die Anwendung ist dafür verantwortlich, den Algorithmus der Vektoruhr umzusetzen. Nachfolgende **Schnittstellen** sind dazu anzubieten:

1. initVT(): erstellt einen initialen Vektorzeitstempel. Dazu nimmt sie gemäß Spezifikation in [towerClock.cfg](#) Kontakt mit der Zentrale towerClock auf, um eine Identität zu erhalten. Rückgabe ist ein initialer Vektorzeitstempel. Dieser stellt die Vektoruhr dar. Achtung: die towerClock.cfg dient nur der ADT vectorC.erl. Der Tower selbst liest diese Datei nicht ein.
2. myVTid(VT): gibt die ProzessID zurück, also <Pnum> des Vektorzeitstempels, als ganze Zahl.
3. myVTCv(VT): gibt den Vektor zurück, also den Vektorzeitstempel, als Liste aus ganzen Zahlen inklusive 0.
4. myCount(VT): gibt den eigenen Zeitstempel als ganze Zahl zurück bzw. den zugehörigen Ereigniszähler, .
5. foCount(j, VT): gibt den Zeitstempel der Position j als ganze Zahl zurück bzw. den zugehörigen Ereigniszähler.
6. isVT(VT): prüft, ob VT ein Vektorzeitstempel ist. Rückgabe ist true oder false.
7. syncVT (VT1, VT2): synchronisiert zwei Vektorzeitstempel (jeweils das Maximum, VT1 wird als eigener Zeitstempel angesehen). Rückgabe ist der neue Vektorzeitstempel.
8. tickVT(VT): zählt den Ereigniszähler des zugehörigen Prozesses um 1 nach oben. Rückgabe ist der neue Vektorzeitstempel.
9. compVT(VT1, VT2): vergleicht Vektorzeitstempel. Rückgabe: afterVT, beforeVT, equalVT oder concurrentVT.
10. aftereqVTj(VT, VTR): vergleicht im Sinne des kausalen Multicast die Vektorzeitstempel, ob VT aftereq VTR ist, ohne Beachtung von Position j. Die Distanz wird berechnet für die Identität j des Vektorzeitstempels VTR, dem Zeitstempel der Nachricht. Rückgabe: {aftereqVTj, <Distanz an der Stelle j>} oder false. Dabei wird die Distanz berechnet, durch VT[j] - VTR[j], wobei j die Identität von VTR ist. Wenn die Distanz -1 ist, dann kann die Nachricht mit Zeitstempel VTR ausgeliefert werden.

Vektoruhr Zentrale / Tower: Der Tower verwaltet die Prozessnummern und gehört zur Vektoruhr-

ADT dazu. Damit hier ein Austausch der ADT gewährleistet wird, ist die Datei als `towerClock.erl` zu bezeichnen. Jede Kommunikationseinheit erhält über `initVT()` beim Tower seine Prozessnummer `<Pnum>`. Nachfolgende **Schnittstellen** sind anzubieten:

1. `{getVecID,<PID>}`: als Nachricht, wobei `<PID>` die Antwortadresse ist. Der Tower sendet an `<PID>` mittels der Nachricht `{vt,<Prozess-ID>}` die eindeutige ID `<Prozess-ID>` (positive ganze Zahl) dieser Vektoruhr.
2. `init()`: startet die Vektoruhr Zentrale gemäß Spezifikation in [`towerClock.cfg`](#). In dieser Datei stehen die Node und der lokale Name der Vektoruhr Zentrale. Rückgabewert ist die PID des Tower.
3. `stop(<PID>)`: wobei `<PID>` die Kontaktadresse des Tower oder seine PID ist. Diese Funktion beendet den Tower. Rückgabewert bei Erfolg ist `true`.

Ungeordneter Multicast: Für die Kommunikationseinheiten wird zum Test ein zuverlässiger ungeordneter Multicast [`towerCBC.beam`](#) zur Verfügung gestellt. Die Spezifikation steht in [`towerCBC.cfg`](#). Ein eigener zuverlässiger ungeordneter Multicast ist jedoch zu implementieren. Damit hier ein Austausch der ADT gewährleistet wird, ist die Datei als `towerCBC.erl` zu bezeichnen. Achtung: die `towerCBC.cfg` dient nur der Kommunikationseinheit `cbCast.erl`. Der Tower selbst liest diese Datei nicht ein. Er sendet im Auftrag der Kommunikationsseinheiten die Nachrichten an die Multicast-Gruppe. Zum Test bietet er Schnittstellen an, um die Nachrichten manuell an die Kommunikationseinheiten zu verteilen. Beachten Sie: es darf kein geordneter Multicast entstehen. Orientieren Sie sich hier bitte an dem zur Verfügung gestelltem `towerCBC!` Nachfolgende **Schnittstellen** werden angeboten:

1. `init()|init(auto|manu)`: startet den Multicast im automatischen Modus (`init()|init(auto)`) oder dem manuellen Modus (`init(manu)`). Rückgabewert ist die PID des Multicast. Achtung: um den manuellen Modus nutzen zu können, muss die Kommunikationseinheit `cbCast.erl` mit `multicastNB` senden!
2. `stop(<PID>)`: wobei `<PID>` die Kontaktadresse des Multicast ist. Diese Funktion beendet den Multicast. Rückgabewert ist bei Erfolg `true`.
3. `reset(<PID>)`: wobei `<PID>` die Kontaktadresse des Multicast ist. Diese Funktion setzt den Multicast wieder in den initialen Zustand. Rückgabewert ist bei Erfolg `true`.
4. `listall()`: muss auf der Node des Multicast ausgeführt werden und listet alle registrierten Kommunikationseinheiten in der zugehörigen Log-Datei auf. Rückgabewert ist bei Erfolg `true`, sonst `false`.
5. `cbcast(<Receiver>,<MessageNumber>)`: muss auf der Node des Multicast ausgeführt werden und sendet die als `<MessageNumber>-te` beim Multicast eingetroffene Nachricht an den Empfänger `<Receiver>`. Dient im manuellen Zustand der manuellen Durchführung des Multicast. Rückgabewert ist bei Erfolg `true`, sonst `false`. Achtung: um den manuellen Modus nutzen zu können, muss die Kommunikationseinheit `cbCast.erl` mit `multicastNB` senden!
6. `{<PID>,{register,<RPID>}}`: als Nachricht. Registriert die Kommunikationseinheit `<RPID>` beim Multicast. `<PID>` erhält als Antwort `{replycbc,ok_existing}` wenn er bereits registriert wurde oder `{replycbc,ok_registered}` wenn er neu registriert wurde. Der Multicast wird an `<RPID>` gesendet.
7. `{<PID>,{multicastB,{<Message>,<VT>}}}`: als Nachricht. Sendet die Nachricht `<Message>` mit Vektorzeitstempel `<VT>` als ungeordneten, blockierenden Multicast an alle Gruppenmitglieder. Blockierend bedeutet, dass in der Zeit kein anderer Multicast versendet wird. `<PID>` ist eine PID und wird lediglich im Log vermerkt.
8. `{<PID>,{multicastNB,{<Message>,<VT>}}}`: als Nachricht. Sendet die Nachricht `<Message>` mit Vektorzeitstempel `<VT>` als ungeordneten, nicht blockierenden Multicast an alle Gruppenmitglieder. Nicht blockierend bedeutet, dass in der Zeit andere Multicast versendet

werden können. <PID> ist eine PID und wird lediglich im Log vermerkt.

Hinweise zum kausalen Multicast

- Comm-Module (Gruppenmitglieder) müssen sich beim Multicast-Tower (towerCBC) registrieren. Dazu senden sie die Adresse ihres receivers (Dispatchers): {<PID>, {register,<PIDReceiver>}}
- Anwender kommunizieren mit dem Comm-Modul (received, read, send, init, stop).
- Das Comm-Modul benötigt eine Holdbackqueue (HBQ) und eine Deliveryqueue (DLQ).
- Der kausale Multicast wird in der HBQ sicher gestellt. Dort wird entschieden, ob die Nachricht ausgeliefert werden darf.
- Die DLQ stellt die Multicastreihenfolge dar. In der DLQ werden die Nachrichten sortiert nach Vorgabe der HBQ.
- Die lokale Vektoruhr wird bei Auslieferung der Nachricht (received, read) mit dem Zeitstempel der Nachricht synchronisiert.
- Ein received/read des Anwenders ist letztlich ein received/read an die DLQ. Die DLQ liefert die nächste auslieferbare Nachricht.
- Achtung: es darf kein total geordneter Multicast entstehen. Unterster zuverlässiger Multicast (towerCBC) ist ungeordnet!
- Die Nachrichten eines Anwenders (send) werden im eigenen Comm-Modul direkt in die DLQ eingestellt.
- Der zur Verfügung gestellte zuverlässige ungeordnete Multicast stellt beim manuellen Test nicht den kausalen Multicast sicher. Im Gegenteil: er liefert die Multicast Nachrichten zuverlässig, aber möglichst ungünstig für den kausalen Multicast aus.

Anwendung: für das erstellte System ist ein **Anwendungsscenario zu beschreiben** und zu implementieren.

Hinweise

- Sofern in der Zusammenarbeit Ihrer Software mit den zur Verfügung gestellten Einheiten es zu Fehlermeldungen kommt, die die bereit gestellten Einheiten betreffen, senden Sie bitte diese Meldungen inklusive der kompletten Anwendung an mich zur Rücksprache.
- Die Austauschbarkeit folgender Einheiten muss sichergetellt sein: Die Kommunikationseinheit in cbCast.erl, die Vektoruhr-ADT in vectorC.erl sowie der Tower der Vektoruhr-ADT in towerClock.erl. Dies bedeutet insbesondere, dass keine weiteren Dateien benötigt werden dürfen, um die Funktionalität zu erhalten.
- In den Dateien [util.erl](#) und [vsutil.erl](#) gibt es nützliche Funktionen. Diese dürfen nicht modifiziert werden. Diese stehen dem System auch jederzeit zur Verfügung und wirken sich somit nicht auf die Austauschbarkeit der beschriebenen Einheiten aus, d.h. die austauschbaren Einheiten können diese Module nutzen.
- Als Beispiel und zum Testen können Sie das folgende System verwenden: [Testsystem](#). Hinweis: die vectorC ist hier als RPC realisiert, um Fehler durch den nicht zulässigen Zugriff in die ADT mittels Tests aufzudecken.
- Halten Sie das System einfach und strukturiert! Wenn die Kernfunktionalität erstellt wurde, kann das System immer noch beliebig erweitert/verbessert werden.

Abgabe

Die Abgabe ist auf den Folien zur Aufgabenstellung genau spezifiziert.

Gratis Counter by GOWEB