

BP 1	Verteilte Terminierung: Überblick
8	Verteilte Terminierung
8.1	Die Fragestellung Das Märchen von der verteilten Terminierung Ergebnisorientierte Terminierung Kommunikationsorientierte Terminierung
8.2	Analyse des Problems
8.3	Zählverfahren
8.4	Zeitzoneverfahren
8.5	Die Vektormethode
8.6	Die Kreditmethode
13.12.01	Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg \ ist ohne Genehmigung des Autors unzulässig
	8.1-1

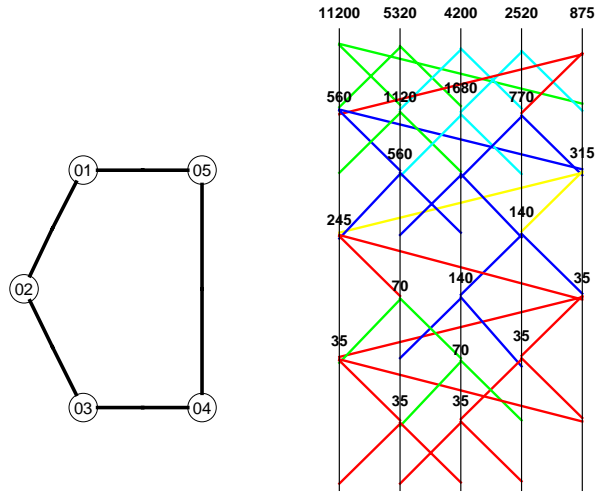
BP 1	Verteilte Terminierung: Überblick
8.1	Die Fragestellung Das Märchen von der verteilten Terminierung
13.12.01	Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg \ ist ohne Genehmigung des Autors unzulässig
	8.1-2

BP 1	Verteilte Terminierung: Überblick
<input type="checkbox"/>	Berechnungen sollen nach folgendem Muster erfolgen: 1. Prozesse befinden sich stets in einem der Zustände <i>aktiv</i> oder <i>passiv</i> . 2. Nur aktive Prozesse können Basisnachrichten aussenden. 3. Aktive Prozesse können jederzeit passiv werden. 4. Passive Prozesse können nur durch Empfang von Basisnachrichten in den aktiven Zustand wechseln.
<input type="checkbox"/>	Interpretationen für <i>passiv</i>: Typische Interpretationen für <i>passiv</i> 1. Das Programm, das ein Prozeß ausführt, ist beendet. 2. Der Prozeß befindet sich in einer Endlosschleife, die mit Sicherheit nicht mehr zum Aussenden von Basisnachrichten führt. 3. Der Prozeß hat seine letzte Basisnachricht versandt, rechnet aber lokal noch einige Zeit weiter. 4. Der Prozeß wartet blockiert auf den Empfang einer Basisnachricht.
13.12.01	Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg \ ist ohne Genehmigung des Autors unzulässig
	8.1-3

BP 1	Verteilte Terminierung: Überblick
<input type="checkbox"/>	Eine Berechnung heißt <i>terminiert</i>, wenn alle Prozesse passiv sind und keine Basisnachricht unterwegs ist.
<input type="checkbox"/>	Beispiel: Berechnung des größten gemeinsamen Teilers • Jede der Zahlen, zu denen der größte gemeinsame Teiler berechnet werden soll, ist einem Prozeß als sein Anfangswert zugeordnet. • Start: Sende an alle Nachbarn den eigenen Anfangswert • Empfang einer Nachricht: Falls der empfangene Wert kleiner als der eigene ist, wird folgendermaßen verfahren: - Ist der empfangene Wert ein Teiler des eigenen, so wird der empfangene Wert an alle Nachbarn verteilt, - ansonsten wird der Rest, der bei der Division des eigenen Wertes durch den empfangenen bleibt, an alle Nachbarn verteilt.
13.12.01	Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg \ ist ohne Genehmigung des Autors unzulässig
	8.1-4

BP 1 Verteilte Terminierung: Überblick

Nachrichtenspiel



13.12.01

Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

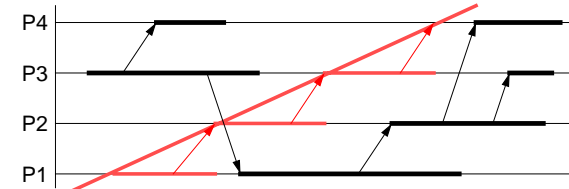
8.1-5

BP 1 Verteilte Terminierung: Analyse des Problems

8.2 Analyse des Problems

Kommunikationsmodelle

◆ Asynchrones Modell (Berechnungen und Übertragungen benötigen Zeit)



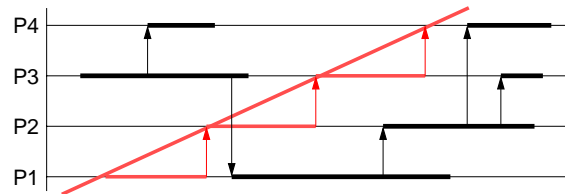
13.12.01

Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

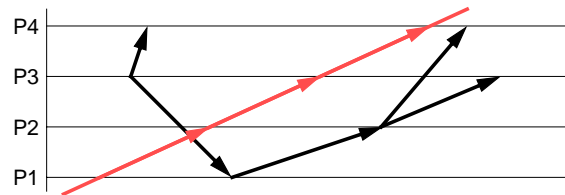
8.2-6

BP 1 Verteilte Terminierung: Analyse des Problems

◆ Synchrones Modell (Übertragung zeitlos)



◆ Atommodell (Bearbeitung zeitlos)



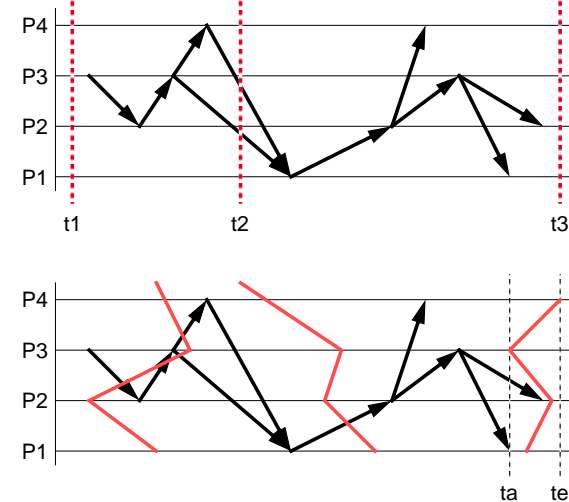
13.12.01

Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

8.2-7

BP 1 Verteilte Terminierung: Analyse des Problems

□ Zeitdiagramm einer Berechnung im Atommodell



13.12.01

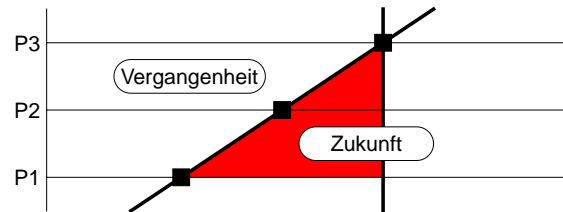
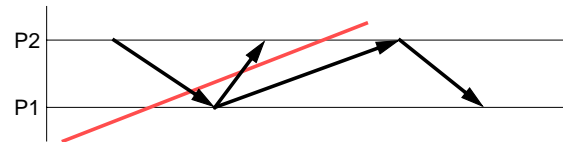
Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

8.2-8

BP 1 Verteilte Terminierung: Zählverfahren

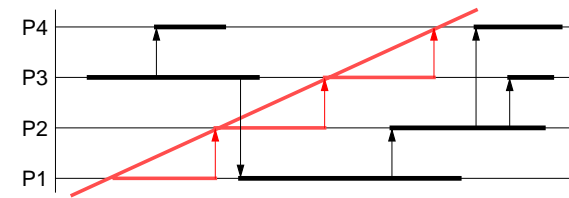
8.3 Zählverfahren

- Einfaches Zählen genügt nicht

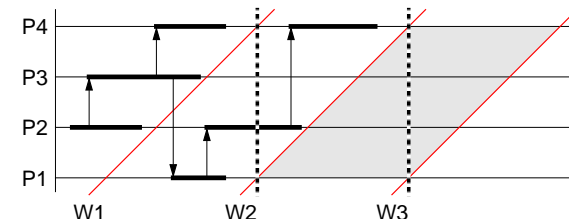


BP 1 Verteilte Terminierung: Zählverfahren

- Auch im synchronen Modell tritt das Phänomen der Nachricht aus der Zukunft auf



◆ Doppelzählung



BP 1 Verteilte Terminierung: Zählverfahren

- Lösungsideen zur "Reparatur" der Zählverfahren

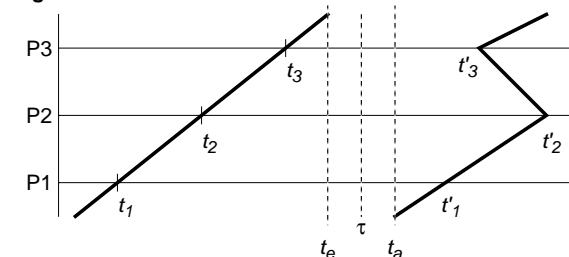
- Einfrieren der Basiskommunikation im gefährlichen Dreieck
- Erkennen von inkonsistenten Zeitschnitten durch geeignete logische Zeitstempel
- Differenzierteres Zählen durch Identifikation der Nachrichten
- Nachträgliches Überprüfen

- Beurteilungskriterien

- Einfachheit, Verifizierbarkeit, Korrektheit
- Zugrundeliegendes Modell
- Topologieabhängigkeit
- Globales Wissen
- Dynamik
- Symmetrie
- Fehlertoleranz
- Implementierbarkeit
- Effizienz

BP 1 Verteilte Terminierung: Doppelzählverfahren

- Bezeichnungen

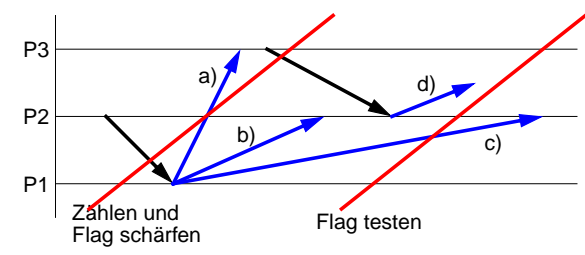


- t_i Zeitpunkt, zu dem die erste Welle Prozeß i erreicht.
- t'_i Zeitpunkt, zu dem die zweite Welle Prozeß i erreicht.
- $s_i(t)$ Zahl der Basisnachrichten, die Prozeß i bis Zeitpunkt t versandt hat.
- $r_i(t)$ Zahl der Basisnachrichten, die Prozeß i bis Zeitpunkt t empfangen hat.
- t_e Ende der ersten Welle

BP 1	Verteilte Terminierung: Doppelzählverfahren
	<p>t_a Beginn der zweiten Welle</p>
13.12.01	<div> <div>Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig</div> <div>8.3-13</div> </div>

BP 1	Verteilte Terminierung: Doppelzählverfahren
	<p>Weiter sei: $S^* = \sum_i s_i(t_i)$, $S'^* = \sum_i s_i(t'_i)$,</p> <p>$R^* = \sum_i r_i(t_i)$ und $R'^* = \sum_i r_i(t'_i)$.</p> <p>S8.1 Satz</p> <p>Wird festgestellt, daß die Zahl der gesendeten und empfangenen Basisnachrichten entlang einer Welle identisch ist und stimmen die Werte für zwei sich nicht überlappende Wellen überein, dann ist die zugrundeliegende Berechnung terminiert.</p> <p>Beweis: Es gelten folgende Beziehungen</p> <ol style="list-style-type: none"> (1) $t \leq t' \Rightarrow (s_i(t) \leq s_i(t') \wedge r_i(t) \leq r_i(t'))$ (2) $t \leq t' \Rightarrow (S(t) \leq S(t') \wedge R(t) \leq R(t'))$ (3) $R^* \leq R(t_e)$ (4) $S'^* \geq S(t_a)$ (5) $\forall t (R(t) \leq S(t))$
13.12.01	<div> <div>Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig</div> <div>8.3-14</div> </div>

BP 1	Verteilte Terminierung: Doppelzählverfahren
	$R^* = S'^* \Rightarrow R(t_e) \geq S(t_a) \quad \text{wegen (3) u. (4)}$ $\Rightarrow R(t_e) \geq S(t_e) \quad \text{wegen } t_e \leq t_a \text{ u. (2)}$ $\Rightarrow R(t_e) = S(t_e) \quad \text{wegen (5)}$ <p>Also ist zum Zeitpunkt t_e das System terminiert, da bezüglich des Schnittpunkts t_e das kritische Dreieck leer ist.</p>
13.12.01	<div> <div>Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig</div> <div>8.3-15</div> </div>

BP 1	Verteilte Terminierung: Skeptischer Algorithmus
	<p>□ Skeptischer Algorithmus</p> 
13.12.01	<div> <div>Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig</div> <div>8.3-16</div> </div>

BP 1 Verteilte Terminierung: Wahlbasierte Algorithmen

Wahlbasierte Algorithmen

Bei Empfang einer Basisnachricht:

```
s--; com = true;
```

Beim Senden einer Basisnachricht:

```
s++;
```

Start des Algorithmus durch Prozeß s:

```
n = 0; predecessor = 0; accu = 0; com = false;
clock.t++; clock.i = s; /* clock enthält als logische Zeit
                        den Wert eines lokalen Zählers
                        und die Prozeßnummer.
                        */
send(allNeighbors, EXPLORER(clock));
```

13.12.01

Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

8.3-17

BP 1 Verteilte Terminierung: Wahlbasierte Algorithmen

Bei Empfang einer Nachricht EXPLORER(t) von Prozeß p:

```
if (t > clock) {
    clock = t;
    if (predecessor == 0 && n != numberOfNeighbors) testFailed();
    predecessor = p;
    if (numberOfNeighbors == 1) {
        send(predecessor, ECHO(t, s, false));
    } else {
        n = 1; accu = 0; com = false;
        send(allNeighborsWithoutPredecessor, EXPLORER(clock));
    }
} else if (t == clock){
    checkCompleted();
}
```

13.12.01

Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

8.3-18

BP 1 Verteilte Terminierung: Wahlbasierte Algorithmen

Beim Empfang von ECHO(t, sum, failed):

```
if (t == clock) {
    accu += sum;
    com |= failed;
    checkCompleted();
}

checkCompleted()
{
    n++;
    if (n == numberOfNeighbors) {
        if (predecessor != 0)
            send(predecessor, ECHO(t, accu + s, com));
        else if (accu + s == 0 && !com) systemTerminated();
        else testFailed();
    }
}
```

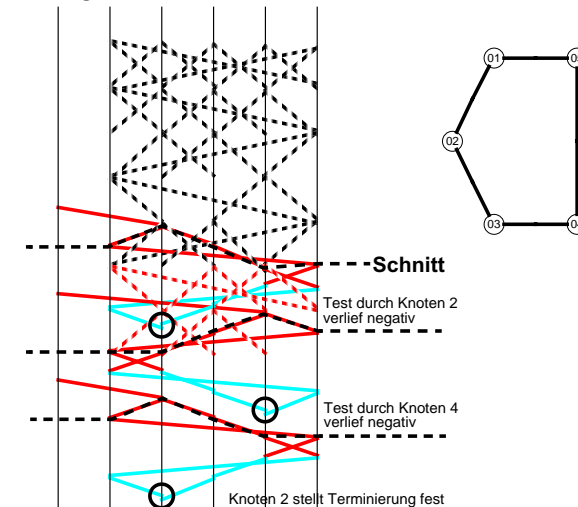
13.12.01

Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

8.3-19

BP 1 Verteilte Terminierung: Wahlbasierte Algorithmen

Beispiel: Größter gemeinsamer Teiler

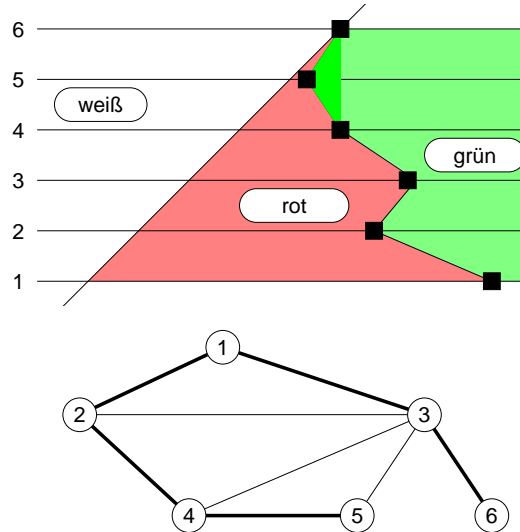


13.12.01

Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

8.3-20

BP 1 Verteilte Terminierung: Wahlbasierte Algorithmen



BP 1 Verteilte Terminierung: Wahlbasierte Algorithmen

1. Wenn von einem roten Prozeß eine Basisnachricht empfangen wird, wird das lokale Kommunikationsflag gesetzt.
2. Wenn eine Basisnachricht von einem roten Prozeß versandt wird, wird schließlich das Kommunikationsflag des Initiators gesetzt.
3. Ein weißer Prozeß kann keine von einem grünen versandte Basisnachricht empfangen.
4. Wenn am Ende des Algorithmus das Flag des Initiators nicht gesetzt wurde, sind alle Nachrichten, die von einem weißen Prozeß empfangen wurden, auch von einem weißen Prozeß versandt worden.
5. Eine Basisnachricht von einem weißen Prozeß, die von einem weißen empfangen wurde, ändert den akkumulierten Nachrichtenzähler nicht.
6. Wenn am Ende des Algorithmus das System nicht terminiert ist, aber das Flag des Initiators nicht gesetzt wurde, dann existiert eine von einem weißen Prozeß ausgesandte Basisnachricht, die von einem grünen Prozeß empfangen wurde oder noch von einem solchen Prozeß empfangen werden wird.
7. Wenn eine von einem weißen Prozeß ausgesandte Basisnachricht von einem grünen empfangen wird und das Flag nicht gesetzt ist, dann ist der Wert des akkumulierten Zählers größer als 0.

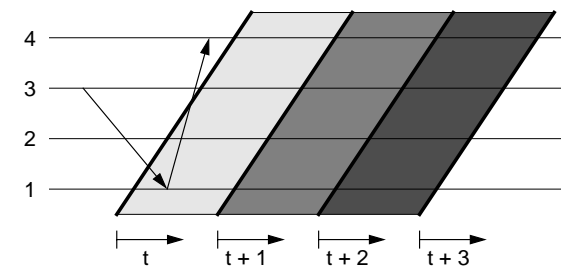
BP 1 Verteilte Terminierung: Wahlbasierte Algorithmen

S8.2 Satz

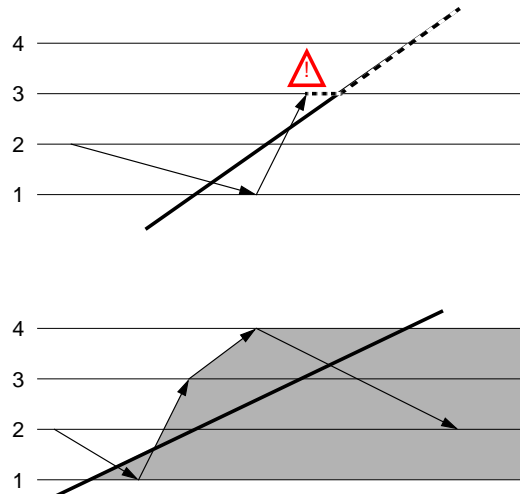
Wenn die Basisberechnung am Ende des Terminierungsalgorithmus noch nicht terminiert ist, dann ist das Flag gesetzt oder der akkumulierte Zähler ist größer als 0.

BP 1 Verteilte Terminierung: Zeitzoneverfahren

8.4 Zeitzoneverfahren



BP 1 Verteilte Terminierung: Zeitzonenverfahren



13.12.01 Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

8.4-25

BP 1 Verteilte Terminierung: Terminierung (Zeitzone-Algorithmus)

Symmetrischer, ringbasierter Zeitzonealgorithmus

```
int CLOCK = 0; // aktuelle, lokale Zeit
int COUNT = 0; // Zahl gesendeter minus Zahl empfangener
                // Nachrichten
int TMAX = 0; // Höchster Zeitstempel der empfangenen
                // Basisnachrichten
```

(a) Senden einer Basisnachricht an P_i

1. `COUNT = COUNT + 1;`
2. `send(CLOCK, ..., P_i);`

(b) Empfang einer Basisnachricht (TSTAMP, ...) durch P_j

3. `COUNT = COUNT - 1;`
4. `TMAX = max(TSTAMP, TMAX);`
5. `Bearbeiten der Nachricht;`

13.12.01 Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

8.4-26

BP 1 Verteilte Terminierung: Terminierung (Zeitzone-Algorithmus)

(c) Empfang einer Kontrollnachricht (TIME, ACCU, INVALID, INIT) durch P_j

```
6.  CLOCK = max(TIME, CLOCK);
7.  if (INIT == j)
8.    if (ACCU == 0 & !INVALID)
        terminated();
9.    else tryAgain();
10. else send(TIME, ACCU + COUNT, INVALID | TMAX >= TIME, INIT,
            P_(j + 1) % n + 1);
```

(d) Bei Start einer Kontrollrunde durch P_j

```
12. CLOCK := CLOCK + 1;
14. send(CLOCK, COUNT, false, j, P_(j + 1) % n + 1);
```

13.12.01 Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

8.4-27

BP 1 Verteilte Terminierung: Terminierung (Zeitzone-Algorithmus)

Zeitzonealgorithmus für sternförmige Architektur mit endlichem Epochenzähler und Sternmittelpunkt P_0

(a) Senden einer Basisnachricht an P_i

1. `COUNT = COUNT + 1;`
2. `send(TIME_ZONE, ..., P_i);`

(b) Empfang einer Basisnachricht <TSTAMP, ...>

3. `COUNT = COUNT - 1;`
4. `TIME_WARP = TIME_WARP / TSTAMP == (TIME_ZONE + 1) % k;`
5. `Bearbeiten der Nachricht;`

(c) Empfang einer Kontrollnachricht von P_0

6. `send(COUNT, TIME_WARP, P_0);`
7. `TIME_WARP = false;`
8. `TIME_ZONE = (TIME_ZONE + 1) % k;`

13.12.01 Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg
ist ohne Genehmigung des Autors unzulässig

8.4-28

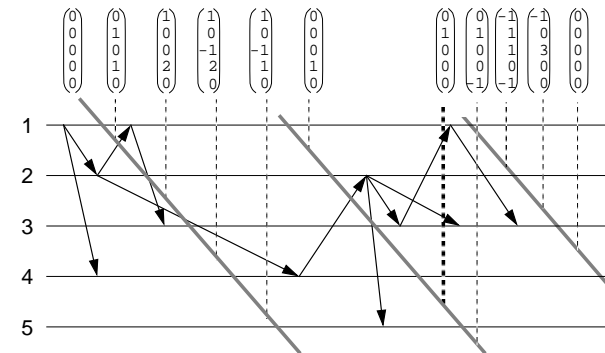
BP 1 Verteilte Terminierung: Terminierung (Zeitzone-Algorithmus)

8.5 Vektorverfahren

Jeder Prozeß P_j führt einen lokalen Vektor V der Länge n (= Gesamtzahl der Prozesse). V ist mit dem Nullvektor initialisiert. Wenn P_j eine Basisnachricht an P_i verschickt, wird $V[i]$ um 1 erhöht, wenn P_j eine Basisnachricht erhält, wird $V[j]$ um 1 erniedrigt.

Ein Kontrollvektor C umläuft einen (virtuellen) Ring. Wenn er bei einem Prozeß vorbeikommt, wird der lokale Vektor hinzuaddiert und V selbst wieder auf den Nullvektor gesetzt.

BP 1 Verteilte Terminierung: Vektorverfahren



BP 1 Verteilte Terminierung: Terminierung (Vektorverfahren)

♦ Verhalten von Prozeß P_j

(a) Senden einer Basisnachricht an P_i

1. $COUNT[i] = COUNT[i] + 1;$

(b) Nach lokalen Aktivitäten bei Empfang einer Basisnachricht

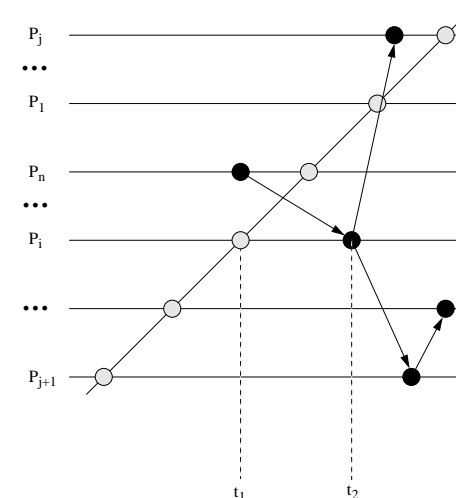
2. $COUNT[j] = COUNT[j] - 1;$

(c) Empfang einer Kontrollnachricht accumulate(ACCU)

```
3. COUNT = COUNT + ACCU;
4. if (COUNT == {0, 0, ..., 0}) systemTerminated();
5. else {
    send(accumulate(COUNT), P(j + 1) % n);
6.   COUNT = {0, 0, ..., 0};
7. };
```

BP 1 Verteilte Terminierung: Terminierung (Vektorverfahren)

Nachweis der Richtigkeit des Verfahrens



BP 1	Verteilte Terminierung: Kreditverfahren
8.6	Kreditverfahren <ul style="list-style-type: none"> ◆ Forderungen an die Kreditanteile: <ol style="list-style-type: none"> 1. Zu jedem Zeitpunkt besitzt die Summe aller Kreditanteile, die von Prozessen oder Nachrichten gehalten werden, den Wert 1. 2. Ein aktiver Prozeß hält einen Kreditanteil größer 0. 3. Eine Basisnachricht, die unterwegs ist, hält einen Kreditanteil größer 0. ◆ Mögliche Umsetzung der Idee <p>(R1) Wenn ein Prozeß passiv wird, übersendet er seinen Kreditanteil an den Urprozeß.</p> <p>(R2) Wenn eine Basisnachricht mit Kreditanteil C bei einem aktiven Prozeß ankommt, wird C an den Urprozeß weitergesandt.</p> <p>(R3) Wenn eine Basisnachricht mit Kreditanteil C bei einem passiven Prozeß ankommt, wird C dem dadurch aktivierten Prozeß übergeben.</p> <p>(R4) Wenn ein aktiver Prozeß mit Kreditanteil C eine Basisnachricht versendet, behält der Prozeß C/2, die andere Hälfte erhält die Nachricht.</p>
13.12.01	<small>Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg \ ist ohne Genehmigung des Autors unzulässig</small> <div>8.6-33</div>

BP 1	Verteilte Terminierung: Kreditverfahren
	<ul style="list-style-type: none"> ◆ Darstellung von C: CREDIT = - Id C ◆ Beim Versenden einer Basisnachricht: <pre>CREDIT++; send(CREDIT, ...);</pre> ◆ Beim Wechsel in den passiven Zustand: <pre>send(CREDIT, Urprozess, ...); active = false; /* Als Darstellung für CREDIT = 0 */</pre> ◆ Bei Empfang einer Basisnachricht (CR, ...): <pre>if (active) send(CR, Urprozess, ...); else { active = true; CREDIT = CR; }</pre>
13.12.01	<small>Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg \ ist ohne Genehmigung des Autors unzulässig</small> <div>8.6-34</div>

BP 1	Verteilte Terminierung: Kreditverfahren
	<ul style="list-style-type: none"> ◆ Urprozeß empfängt eine Basisnachricht (CR, ...): <p>DEBTS Menge von Werten d_i derart, daß</p> $CREDIT_{Urprozess} = 1 - \sum_i 2^{-d_i}$ <pre>K = CR; while (K ∉ DEBTS) { DEBTS = DEBTS ∪ {K}; K--; } DEBTS = DEBTS - {K}; if (DEBTS == ∅) terminated();</pre>
13.12.01	<small>Universität Erlangen-Nürnberg, Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme), F. Hofmann Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg \ ist ohne Genehmigung des Autors unzulässig</small> <div>8.6-35</div>