Thema: Uhrensynchronisation
Aus ATA Welsh: 6.3 Clock Synchronization
Von Jan Schröder

Szenario:

- n Prozessoren in einem Netzwerk
- Uhren driften nicht!

Ziel:

Synchronisation

Überblick:

1. modellieren von hardware clocks
2. Definitionen und Begriffe
3. Uhrensynchronisation mit zwei Prozessoren – zwei einfache Algorithmen
4. Uhrensynchronisation mit n Prozessoren – Aussagen über obere und untere Schranken

1. Modellieren von hardware clocks

Erweitere p_i um die hardware clock HC_i:

- HC_i ist in jeder Transition von p_i verfügbar
- P_i kann HC_i nicht ändern
- HC_i driftet nicht → $HC_i(t) = t + c_i$ für eine konstante c_i

2. Definitionen und Begriffe

Definition 6.2 A view with clock values for processor p_i consist of:

- initial state of p_i
- a sequence of events (computation and deliver) that occur at p_i
- a hardware clock value assigned to each event

requirement:

the hardware clock values must increase

Definition 6.3 A timed view with clock values of a processor p_i consists of:

- view with clock values

and additional:

- real time assignment to each event
- consistency with hardware clock:
  $HC_i(t) = t + c_i$ for a constant c_i

Definition Merge:

- set of n timed views η_i for each p_i ($0 \leq i \leq n$)

merge as follows:

1. combine the initial states of every η to obtain the initial configuration
2. interleave the events in the timed views consistently with the real times.

   Deliver events at time t before computation events at t
3. apply in order to obtain timed execution (if the new timed views are consistent: any send events must occur before the deliver events)

denote result:

$\text{merge}(\eta_0, \ldots, \eta_{n-1})$

Definition 6.4 Shifting an execution:

- α timed execution with hardware clocks
- $\eta_0, \ldots, \eta_{n-1}$ the timed views extracted from α
- x a vector of n real numbers

shift(α,x) := merge($\eta_0, \ldots, \eta_{n-1}$) after adding x_i to real time events in η_0

Lemma 6.14

- α timed execution with hardware clocks HC_i $(0 \le i \le n-1)$
- x vector of n real numbers
- HC_i' the hardware clocks in shift(α,x) $(0 \le i \le n-1)$

Following holds:

a) $HC_i' = HC_i - x_i$

b) Messages from p_i to p_j have delay $\delta - x_i + x_j$ (δ the message delay in α and $0 \le i,j \le n-1$)

Adjusted clocks

- every p_i has a state component adj_i that it can manipulate
- $AC_i(t) = HC_i(t) + adj_i(t)$

Definition 6.5 Achieving ε-Synchronized Clocks:

- admissible timed Execution
- algorithm terminated after t_f

following has to hold:

$|AC_i(t) – AC_j(t)| \leq \varepsilon$ for all processors pi and pj and all $t \geq t_f$

figure 6.10

3. Uhrensynchronisation mit 2 Prozessoren

Im Folgenden seinen d und u Variablen:

- d,u nicht negativ
- $d \geq u$
- Nachrichten haben delay in [d-u , d] (in jeder admissible timed execution)
- u heißt die Unsicherheit (uncertainty) im delay

4. <u>Uhrensynchronisation mit n Prozessoren – Aussagen über obere und untere Schranken</u>

Alg 6.2

Theorem 6.15 (An Upper Bound):
Algorithm 6.2 achieves u(1 – 1/n)-synchronization for n processors

Theorem 6.17 (A Lower Bound):

- A Algorithm
- A achieves ε-synchronized clocks

following holds:

$$\varepsilon \geq u(1 - 1/n)$$