Leader Election in a Synchronous Ring

Das Problem

- synchrones Ring-Netzwerk
- Prozesse sollen einen Anführer (Leader) bestimmen
- eine der urspruenglichen Anwendungen: verlorenes Token im Token Ring
- n Knoten
- wir rechnen modulo
- nur lokale Kenntnisse (kein Prozess kennt seinen Index oder die seiner Nachbarn, er weiß nur, dass es einen linken und rechten Nachbarn gibt)

Problemvarianten

- Sollen Non-Leaders auch feststellen, dass sie nicht Leader sind oder reicht es, wenn der Leader weiß, dass er Leader ist?
- Unidirektionaler oder Bidirektionaler Ring?
- n bekannt?
- UIDs (unique process identifiers) oder nicht?

Unmöglichkeitsresultat für identische Prozesse

Theorem

A ein System aus $n > 1$ Prozessen, angeordnet als bidirektionaler Ring. Wenn alle Prozesse in A identisch sind, löst A nicht das Leader Election-Problem.

Proof.

- Angenommen, *A* ist so ein System.
- O.B.d.A. hat jeder Prozess nur einen Startzustand (ansonsten lösche alle bis auf einen – für jeden Prozess gleichermaßen).
- Dann gibt es nur eine Execution.
- Für alle Runden *r* gilt: Alle Prozesse sind nach Runde *r* in identischen Zuständen. (Induktion)
- Also ermitteln sich entweder alle oder niemand als Leader.
- Widerspruch!

Lösung

- Symmetrie brechen
- ohne unterschiedlichen Code für die Prozesse zu verwenden...
- UIDs (unique process identifiers)

LCR-Algorithmus

- Le Lann, Chang, Roberts
- nur unidirektionale Kommunikation
- n unbekannt
- nur Vergeichsoperationen auf UIDs

... slide ...

Korrektheitsbeweis

- i_{max}: Index des Prozesses mit maximaler UID
- u_{max}: maximale UID
- wir zeigen:
  - nach n Runden erkennt sich Prozess i_{max} als *leader*
  - kein anderer Prozess erkennt sich jemals als *leader*

Nach n Runden erkennt sich Prozess i_{max} als *leader*

- Invariante: Nach n Runden gilt $status_{i_{max}} = leader$.
- Für Induktion ist Aussage für kleinere n nötig:
- Invariante: Für $0 \leq r \leq n-1$ gilt, dass nach r Runden $send_{i_{max}+r} = u_{max}$.

Kein anderer Prozess erkennt sich jemals als *leader*

- Invariante: Alle Prozesse außer i_{max} haben immer $status = unknown$.
- Für Induktion, verstärke wieder die Aussage:
- Invariante: Für alle r, i, j gilt: Nach r Runden, falls $i \neq i_{max}$ und $j \in [i_{max}, i)$, dann $send_j \neq u_i$.
  - $r = 0 \Rightarrow send_j = u_j$. UIDs sind eindeutig.
  - $r \rightarrow r + 1$: Angenommen, i und $j \in [i_{max}, i)$ existieren mit $send_j = u_i$ in Runde $r + 1$. Dann war in Runde r $send_{j-1} = u_i$.
    - Falls $j - 1 \in [i_{max}, i)$, ist $send_{j-1} = u_i$ (in Runde r) ein Widerspruch zur Induktionsvoraussetzung.
    - Falls (in Runde $r + 1$) $j = i_{max}$, dann $send_{i_{max}} = u_i$, also nach Code $u_i > u_{i_{max}}$, Widerspruch.

Anhalten/Non-Leader Ausgaben

- Nicht-Leader erfahren nicht
  - wann der Algorithmus beendet ist
  - dass sie nicht der Leader sind
  - wer der Leader ist
- Lösung durch Erweiterung des LCR: Zusätzliche Nachricht *report*, die vom neu gewählten Leader initiiert wird

Komplexität des LCR

- time complexity: n Runden
- communication complexity: $O(n^2)$ Nachrichten

HS-Algorithmus

- Hirschberg, Sinclair
- hat geringere communication complexity: $O(n \log n)$, damit geringere Netzwerklast
- n unbekannt
- nur Vergleichsoperationen auf UIDs
- *bi*direktionale Kommunikation

... two slides ...

Komplexität des HS

- time complexity: $O(n)$ Runden
- communication complexity: $O(n \log n)$ Nachrichten
  - geringere Komplexität, weil ein Großteil der Nachrichten schon ab frühen Phasen nicht mehr gesendet wird
  - Übung