Distributed Algorithms

Seminar 3
Termination Detection and Garbage collection
Termination Detection

◆ Dijkstra-Scholten Algorithm
◆ Shavit-Francez Algorithm
◆ Weight Throwing Algorithm
◆ Rana’s Algorithm
◆ Safra’s Algorithm
Termination Detection

• Dijkstra-Scholten Algorithm
  – Form a tree rooted at initiator

• Consider 3 processes: p, q, r
  – If p sends message to q, q becomes active, joins p

p sends message to q

q becomes active and joins p in tree
IDEA: TO MAKE A TREE OF PROCESS STATES

ROOT/INITIATOR
RULES for forming T

• T contains only one Initiator
• At sender
  – P sends m1 to q; set CCp=1
  – CCp : Number of children at p
• At receiver
  – If q not part of T, thus joins T
  – If q already part of T, sends control message to P; Cpp=Cpp - 1
Quit Tree T

• Initiator: Passive and $CC_{\text{initiator}} = 0$
• Non-initiator
  – *when?* Passive & Cci = 0
  – *How?* Sends control message m’ to parent j
  – At parent, CCj = CCj – 1
Ex. 6.1

• How much time does the Dijkstra-Scholten algorithm need at most to call announce after the basic algorithm has terminated?
  – Hint: depth of the tree
Shavit-Francez Algorithm

• Generalized Dijkstra-Scholten algorithm for decentralized basic algorithms
• Not one tree, many trees
RULES for forming T

• T contains only initiator’s
• At sender
  – P sends m1 to q; set CCp=1
  – CCp : Number of children at p
• At receiver
  – If q not part of T, thus joins T
  – If q already part of a tree, sends control message to P; Cpp=Cpp - 1
Quit Tree \( T \)

- **Initiator**: Passive and \( CC_{\text{initiator}} = 0 \)
- **Non-initiator**
  - *when?* Passive & \( CC_i = 0 \)
  - *How?* Sends control message \( m' \) to parent \( j \)
  - At parent, \( CC_j = CC_j - 1 \)
Algorithm completes?

• Initiator with CCi=0 starts wave with id “i”
• Processes not part of “any” tree participate in Wave
• When wave completes(all processes participate), algo completes
Ex 6.2

• Give computation of Shavit-Francez algorithm with 2 initiators, in which one of the initiators becomes active again after it becomes passive, and both initiators concurrently call announce.
6.2: Give computation of Shavit-Francez algorithm with 2 initiators, in which one of the initiators becomes active again after it becomes passive, and both initiators concurrently call announce

- p and q send basic messages to each other (CCp = CCq = 1)
- p becomes passive
- q receives p’s message and becomes passive, and sends ack to p
- When p receives q’s **ack**, it becomes active (CCp=0)
- P becomes passive again, sends **ack** to q
- After p and q receive each other **ack**, both start wave, all processes participate (CCq=0)
- Both waves complete, and both call announce
Ex 6.3

• Consider weight-throwing termination detection, where in case of underflow at p, it gives itself extra weight, and informs initiator. Give an example to show that if p would not wait for ack from initiator, then initiator could prematurely detect termination.
Ex 6.3

• Consider weight-throwing termination detection, where in case of underflow at $p$, it gives itself extra weight, and informs initiator. Give an example to show that if $p$ would not wait for ack from initiator, then initiator could prematurely detect termination.

• Initially $p$ has weight 2 and is the initiator
Ex 6.3

• Consider weight-throwing termination detection, where in case of underflow at p, it gives itself extra weight, and informs initiator. Give an example to show that if p would not wait for **ack** from initiator, then initiator could prematurely detect termination. (Consider situation below)

• p and r are passive, q is active with weight 1
Ex 6.3

• Consider weight-throwing termination detection, where in case of underflow at p, it gives itself extra weight, and informs initiator. Give an example to show that if p would not wait for ack from initiator, then initiator could prematurely detect termination. (Consider situation below)

• q increases weight = 2, sends information message to p, then sends basic message to r with weight = 1
Ex 6.3

• Consider weight-throwing termination detection, where in case of underflow at p, it gives itself extra weight, and informs initiator. Give an example to show that if p would not wait for **ack** from initiator, then initiator could prematurely detect termination. (Consider situation below)

• q increases weight = 2, sends information message to p, then sends basic message to r with weight = 1 (**info still not reached p**)
Ex 6.3

- Consider weight-throwing termination detection, where in case of underflow at p, it gives itself extra weight, and informs initiator. Give an example to show that if p would not wait for ack from initiator, then initiator could prematurely detect termination. (Consider situation below)

- q becomes passive, returns weight to p, before info message arrives

- p is passive, weight = 2

- Calls announce!
Exercise 6.4a. Shavit-Francez

• Decentralized basic algo on undirected ring of size 3. Processes $p,q,r$, where $p$ and $q$ are initiators.

• $p$ sends message to $q,r$, becomes passive

• While $q$ sends message to $r$

• When $q$ receives $p$’s message, becomes passive

• After receiving msg from first $p$ and then $q$, $r$ sends msg to both $p$ and $q$ and becomes passive

• After reception of message from $r$, $p$ and $q$ send messages to each other, and after reception of these messages become passive.
Exercise 6.4a. Shavit-Francez

• \( p \) & \( q \) are roots

\[ \begin{align*}
CC_p &= 2 \\
CC_q &= 0 \\
CC_r &= 0
\end{align*} \]
Exercise 6.4a. Shavit-Francez

- p & q are roots
Exercise 6.4a. Shavit-Francez

- $p$ & $q$ are initiators

![Diagram showing nodes and connections with control messages]

- $CC_p = 2-1$
- $CC_q = 1$
- $CC_r$
- Message 3: $m_3$
Exercise 6.4a. Shavit-Francez

- $p$ & $q$ are roots

$CC_p = 1$

$CC_q = 1-1$

Control message

$CC_r$
Exercise 6.4a. Shavit-Francez

• $p \& q$ are roots

$CC_p = 1$

$CC_q = 0$

$CC_r$

Q starts wave

$p \& r$ refuse to participate
Exercise 6.4a. Shavit-Francez

- P root

\[ \text{CC}_p = 1 \]

\[ \text{CC}_q = 0 \]

\[ \text{CC}_r = 2 \]

q is no longer root node
Exercise 6.4a. Shavit-Francez

• P root

\[
\begin{align*}
CCp &= 1 \\
CCq &= 0 \\
CCr &= 2^{-1}
\end{align*}
\]

Control message

P

q

r
Exercise 6.4a. Shavit-Francez

- P-root

CCp = 1

Q makes r parent

CCq = 0

CCr = 2 - 1
Exercise 6.4a. Shavit-Francez

- **P-root**

\[ \text{CC}_p = 1+1 \]
\[ \text{CC}_q = 0 \]
\[ \text{CC}_r = 2-1 \]
Exercise 6.4a. Shavit-Francez

• P-root

\[ CC_p = 2-1 \]

\[ CC_q = 0 \]

\[ CC_r = 2-1 \]
Exercise 6.4a. Shavit-Francez

• P-root

\[ \text{CCp} = 1 \]

\[ \text{CCq} = 0 + 1 \]

\[ \text{CCr} = 2 - 1 \]
Exercise 6.4a. Shavit-Francez

- P-root

\[ \begin{align*}
CC_p &= 1 \\
CC_q &= 1-0 \\
CC_r &= 1
\end{align*} \]
Exercise 6.4a. Shavit-Francez

• **P-root**

![Diagram]

- **CCp = 1**
- **CCr = 1**
- **CCq = 0**
- Leaving the tree
Exercise 6.4a. Shavit-Francez

• P-root

CCp = 1

Leaving the tree

CCq = 0

CCr = 0
Exercise 6.4a. Shavit-Francez

- P-root – starts wave and completes

\[ \text{CC}_p = 0 \]
\[ \text{CC}_q = 0 \]
\[ \text{CC}_r = 0 \]
Rana’s Algorithm

• *De-centralized*(many initiators) basic algorithm on undirected network
• Every **basic** message is acknowledged
• Waves carry logical clock value
Rana’s algorithm

• Every process has logical clock values
• Does not require full-blown logical clock (only control and acks are considered)
• Quiet process start wave
  – Passive
  – **ALL** basic messages are acknowledged
Ex. 6.5

• Suppose Rana’s algorithm is adapted as follows: only quite processes that have been quiet from some logical time $< t$ (instead of $\leq t$) onward can take part in wave tagged with timestamp $t$. Give an example of finite computation for which termination would not be detected.
Ex. 6.5

• Suppose Rana’s algorithm is adapted as follows: only quite processes that have been quiet from some logical time \( t < t \) (instead of \( t \leq t \)) onward can take part in wave tagged with timestamp \( t \). Give an example of finite computation for which termination would not be detected.

• Hint: Initially \( p \) and \( q \) active, then become passive, start waves tagged with 0, refuse to participate in each others waves \( T < 0 \)

\[
\begin{align*}
T = 0 & \quad T = 0 \\
p & \quad q
\end{align*}
\]
Safra’s algorithm

• Every process maintains counter
• Send message counter++
• Receive message, color:black, counter--
• “White” token is started by initiator
• Token carries sum of all counters
• **Only** Passive node- forwards white token
• Black(passive) forwards black token, changes itself to white
Ex 6.6

• Give example of Safra’s algorithm where coloring sending process black instead of receiving ones is incorrect
6.6) Give example of Safra algorithm where coloring sending process black instead of receiving ones is incorrect

- S sends a msg to q, increments counter to 1, s becomes black

Cs=1
6.6) Give example of Safra algorithm where coloring sending process black instead of receiving ones is incorrect

- P is passive and sends token
6.6) Give example of Safra algorithm where coloring sending process black instead of receiving ones is incorrect

- Token is black, sum!=0, no termination!
6.6) Give example of Safra algorithm where coloring sending process black instead of receiving ones is incorrect

- Roundtrip2, message from s not reached q, Token reached r, white and sum=0
6.6) Give example of Safra algorithm where coloring sending process black instead of receiving ones is incorrect

- q receives message from s, and q sends msg back to s, thus q becomes black, s recvs message before token

\[ Cs = 1 - 1 = 0 \]
\[ Cp = 0 \]
\[ Cq = -1 \]
\[ Cr = 0 \]
6.6) Give example of Safra algorithm where coloring sending process black instead of receiving ones is incorrect

• Token arrives at s
6.6) Give example of Safra algorithm where coloring sending process black instead of receiving ones is incorrect

• White token reaches passive p, with sum=0, ANNOUNCE!!!! But q is still active…
Garbage Collection: Indirect reference counting

1. Object at Root

2. References as parts of tree

3. Every node has counters; Cci=x

Ccj=y
RULES for forming T (from Dijkstra-Scholten)

• T contains only one Initiator

• At sender
  – P sends m1 to q; set CCp=1
  – CCp : Number of children at p

• At receiver
  – If q not part of T, thus joins T
  – If q already part of T, sends control message to P; Cpp=Cpp - 1
Ex 7.2

- Do indirect (a) reference counting and (b) weighted-reference counting
- Processes p, q, r
- p owns object O. Initially there is one O-pointer
- p sends q and r message containing O-reference
- p deletes O-pointer
- After getting message from p, q and r create O-reference
- q and r send message to each other both containing duplicated O-reference and delete their O-reference
- After arrival of these messages, q and r create O-reference again
- q and r both delete O-reference
7.2. Indirect Ref Counting

• P sends messages to q and r
• CCp=2

CCq=0

CCr=0
7.2 Indirect Ref Counting

- P deletes O-pointer
- But cannot mark O as garbage
7.2. Indirect Ref Counting

- P deletes O-pointer
- But cannot mark O as garbage
- q and r create o-reference

![Diagram showing the pointers and counts]

- $CC_p = 2$
- $CC_q = 0$
- $CC_r = 0$
7.2. Indirect Ref Counting

• q and r send messages to each other
• Delete original o-reference
• Do not notify p, as counter!=0

CCp = 2
CCq = 1
CCr = 1
7.2. Indirect Ref Counting

- $q$ and $r$ restore reference after messages
- Send ctrl messages, $CC_q$, $CC_r=0$

![Diagram](image-url)
7.2. Indirect Ref Counting

- q and r send delete o-reference
- Send decrement message to p

```latex
\begin{align*}
\text{CCp} &= 2 \\
\text{CCq} &= 0 \\
\text{CCr} &= 0
\end{align*}
```
7.2. Indirect Ref Counting

• CCp=0, no pointers to O, p detects O is garbage

\[ CCp = 0 \]
\[ CCq = 0 \]
\[ CCr = 0 \]
Weighted Reference Counting

Object creation, partial weight = total weight
Partial weight distributed to references
If no reference held yet, creates reference with weight from message
If reference already exists, adds the weight of reference to its own weight
Weighted Reference Counting

Delete reference, subtract the weight from Total weight of the object owner
Terminates when Partial weight = Total weight
Ex 7.2

• (b) weighted-reference counting
• Processes p, q, r
• p owns object O. Initially there is one O-pointer
• p sends q and r message containing O-reference
• p deletes O-pointer
• After getting message from p, q and r create O-reference
• q and r send message to each other both containing duplicated O-reference and delete their O-reference
• After arrival of these messages, q and r create O-reference again
• q and r both delete O-reference
Exercise 7.6

• Show that using technique from section 7.2, indirect reference counting gives rise to Dijkstra-Scholten termination detection
  – Initialization
  – Basic Messages
  – Passive Processes
  – Termination Detection
Exercise 7.6

• Indirect reference counting
  – Each process \( p \) hosts a root object \( O_p \)
  – There is artificial non-root object \( Z \)
  – *Only* initiator \( p \) holds reference from \( O_{po} \) to \( Z \)

• Resulting Termination Detection algo
  – Duplication of references to \( Z \) used to build tree rooted in \( Z \)
  – Initially only reference from \( O_{po} \) is child of \( Z \) and \( Cc_{po} = 0 \) (No: of children)
  – Process \( p \) receives ref to \( Z \) for first time, sent process becomes parent and \( Cc_p = 0 \)
Exercise 7.6

• Basic Messages
  – Construction
    • Every message carries duplication of Z reference
  – Indirect Ref Counting
    • When process recv reference but already holds ref to this obj, sends back decrement
    • When process received decrement, dec counter by 1

  – Termination detection algo
    • When process p sends a message, CCp=CCp+1
    • Message is rec by process q
      – Not in tree, make p parent, CCq=0
      – Already in tree, sends decrement message to p, CCp=CCp-1
Exercise 7.6

• Passive Processes
  – Construction
    • When process becomes passive, delete Z reference
  – Indirect Ref Counting Algo
    • When duplicated(or created) reference has been deleted, and counter=0, sent a decrement to parent in tree
  – Termination detection algo
    • When a non-initiator q is passive and CCq=0, informs parent P that no longer a child, CCp=CCp-1
Exercise 7.6

• Termination Detection
  – Construction
    • Basic algo terminated when Z is garbage
  – Indirect Ref Counting
    • When counter of Z becomes 0, can be reclaimed
  – Termination Detection Algo
    • Po(Initiator) is passive and CCp=0, calls announce
Exercise 7.7

• Show using technique in 7.2 that weighted reference counting gives rise to variation of weight-throwing termination detection, in which initiator cannot reuse weight that was send to it.
Exercise 7.7

• Initialization
• Basic messages
• Passive processes
• Termination Detection
Questions

• Mail a.nottamkandath@vu.nl