Distributed Algorithms

Consistent Snapshots

Univ.-Prof. Dr.-Ing. habil. Gero Mühl

Architecture of Application Systems
Faculty for Computer Science and Electrical Engineering
University of Rostock

Overview

> Snapshot problem

> Consistency criterion for snapshots

> Snapshot algorithms
>> Chandy and Lamport
>> Lai and Yang

Motivation

- Determine “current” *snapshot* of the global state (local states + messages) *without* stopping the system
- Consistent snapshots important
  - Checkpoints of a distributed database
  - Current load of a distributed system
  - Does a deadlock exist?
  - Has the algorithm terminated?
  - Can an object be collected?

  Stable predicates

- How can a “consistent” snapshot be determined?

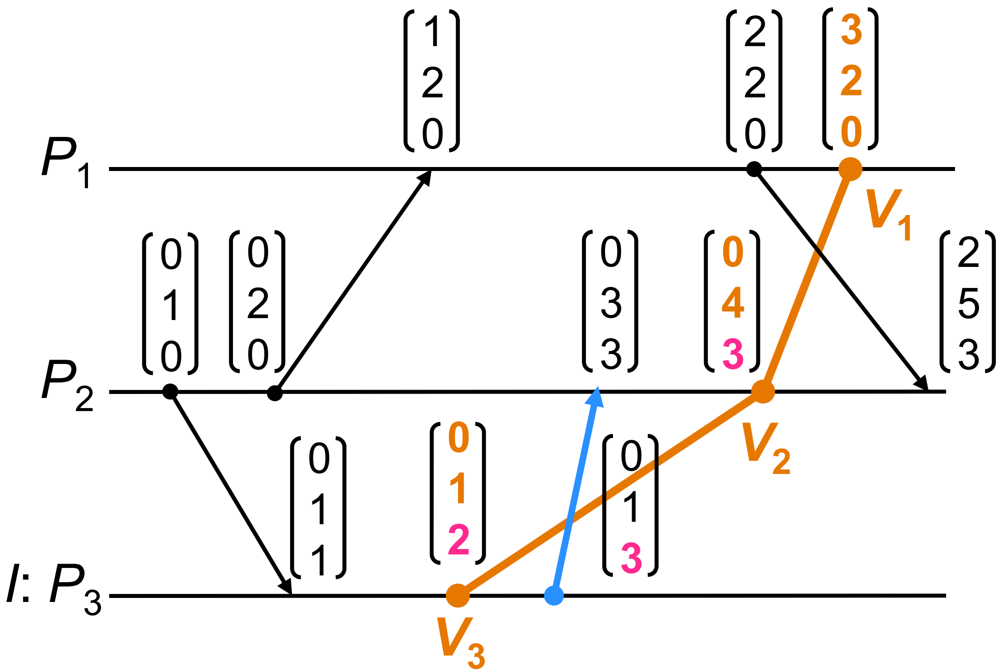
Snapshot Problem

- It is impossible to save the state of all processes at exactly the same time
- Saved state
  - will generally be out of date
  - has most likely never “really” been like that and
  - is probably inconsistent, because of messages from the future
- Requirement: snapshots should be consistent
  - saved state should not be influenced by messages from the future, i.e., sent after the state has been saved at the sender
- Consistent snapshots can be used to detect stable predicates
  - If a stable predicate holds for a consistent snapshot, it also holds for sure in reality

Consistency Criterion for Cuts

- Cut is a matrix comprising the processes' vector timestamps
  - $X = \{V_1, \ldots, V_n)$
- Vector of matrix row maximums
  - $t_x = (\max(V_1[1], \ldots, V_n[1])), \ldots, \max(V_1[n], \ldots, \mathrm{V}_n[n]))$
- Vector of matrix diagonal elements
  - $d_x = (V_1[1], \ldots, V_n[n])$
- X is consistent if and only if $t_x = d_x$
  - Each element of the matrix diagonal must be equal to the maximum of the respective row
  - If $t_x[\,i\,] > d_x[\,i\,]$ applies, some P_j received a message from P_i that was sent after the state was saved at P_i, but before it was saved at P_j
  - $t_x[\,i\,] < d_x[\,i\,]$ cannot occur

Consistency Criterion for Cuts



$$
\begin{array}{ccc} V_1 & V_2 & V_3 \\ \end{array}
\begin{pmatrix} 3 & 0 & 0 \\ 2 & 4 & 1 \\ 0 & 3 & 2 \end{pmatrix}
$$

$$
\begin{aligned}
t_x &= (3, 4, 3)^T \\
&\neq \\
d_x &= (3, 4, 2)^T
\end{aligned}
$$

Cut is not consistent due to blue message

Algorithm by Chandy and Lamport, 1985

- Requires reliable FIFO-channels
- Uses flooding as basic wave procedure
- Uses the flushing principle for communication channels
  - A flush message “pushes” the basic messages that are still in transit out of the FIFO-channels
  - If a node has received a flush message over a channel, it knows that it will receive no more basic messages over that channel

Algorithm by Chandy and Lamport

- Each process *P* receives exactly one flush message from each of its neighbors

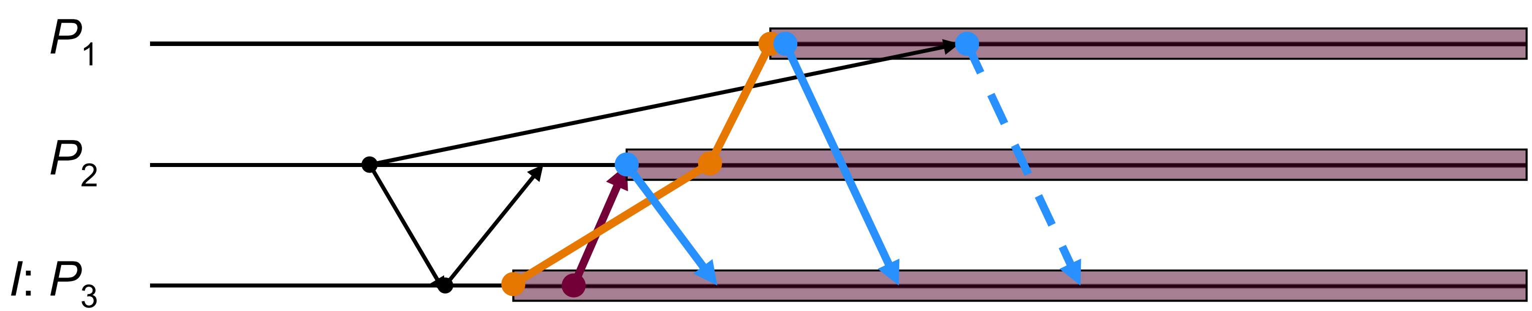
- Case 1: *P* receives for the first time a flush message
  - Let *Q* be the process, *P* received the flush message from
  - *P* saves its state and notes the channel <*Q*, *P*> as empty
  - *P* sends a flush message to all its neighbors

- Case 2: *P* receives further flush messages
  - Let *R* (≠ *Q*) be the process, *P* received that flush message from
  - *P* notes for the channel <*R*, *P*> the sequence of basic messages it received from *R* since the receipt of the first flush message

- Snapshot consists of all local states and messages saved

Algorithm by Lai and Yang, 1987

- Initially, all nodes are black and send black messages
- Initiator of snapshot algorithm gets red and saves local state
- Red nodes only send red messages
- Nodes get red if they are visited or received a red message
- When a node becomes red, it saves its local state and sends it to the initiator
- Red nodes send a copy of all black messages they receive to the initiator → termination detection?



Algorithm by Lai and Yang, 1987

- Snapshot is complete
  - if initiator received the local states of all nodes *and*
  - a copy of each black message that was still in transit
- How does the initiator know that it has all black messages?
- Solution: deficit counter similar to counting algorithm
  - Each node counts the messages sent and received
  - Counter reading is saved along with snapshot
  - Difference of both counters indicates number of black messages in transit

Exemplary Exam Questions

1. What is a consistent snapshot?
2. Explain the consistency criterion for consistent snapshots!
3. Describe the snapshot algorithm of Lai and Yang as well as the snapshot algorithm of Chandy and Lamport!

Literature

1. G. Coulouris, J. Dollimore, and T. Kindberg. Distributed Systems: Concepts and Design. Addison-Wesley, 4th edition, 2005. Chapter 11.5 + 11.6
2. A. S. Tanenbaum and M. van Steen. Distributed Systems: Principles and Paradigms. Prentice Hall, 2002. Chapter 5.3
3. N. Lynch. Distributed Algorithms. Morgan Kaufmann, 1996. Chapter 19
4. **F. Mattern. Verteilte Basisalgorithmen. Springer-Verlag, 1989. Kapitel 3: Das Schnappschussproblem**
5. G. Tel. Introduction to Distributed Algorithms. Cambridge University Press, 2nd edition, 2000. Chapter 10
6. K. M. Chandy and L. Lamport. Distributed Snapshots: Determining Global States of Distributed Systems. ACM Transactions on Computer Systems, 3(1):63--75, February 1985.
7. T. H. Lai and T. H. Yang. On distributed snapshots. Information Processing Letters, 25(3):153--158, 1987.

Thank you for your kind attention!

Univ.-Prof. Dr.-Ing. habil. Gero Mühl

`gero.muehl@uni-rostock.de`
`http://wwwava.informatik.uni-rostock.de`