Technische Universität Berlin

Distributed Algorithms 2015/16

Termination

Reinhardt Karnapke | Communication and Operating Systems Group

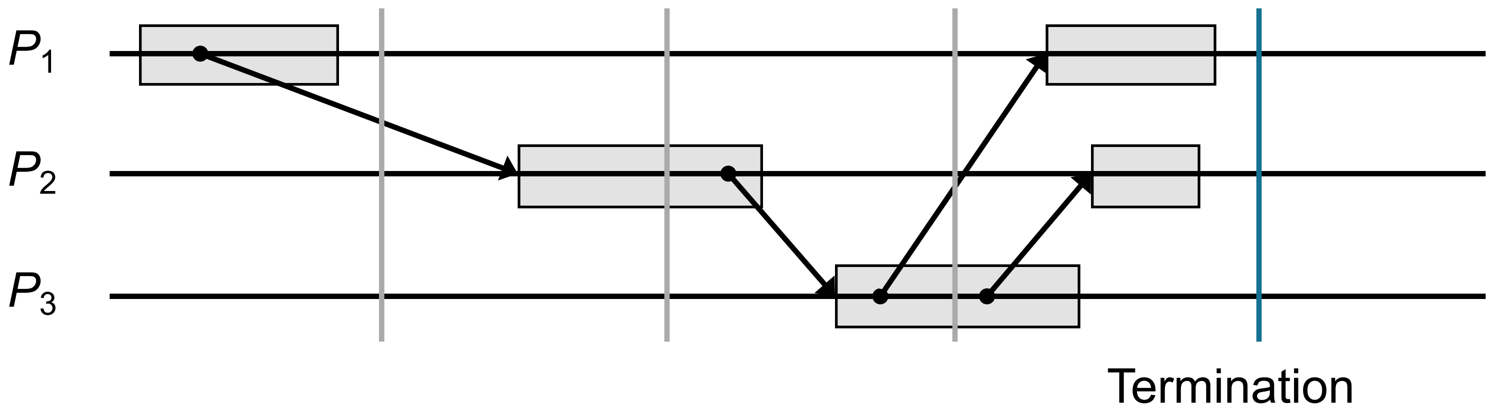
Overview

- Termination in different system models

- Algorithms for termination detection
  - Double Counting Algorithm
  - Time Zone Algorithm
  - Vector Algorithm
  - Credit Algorithm

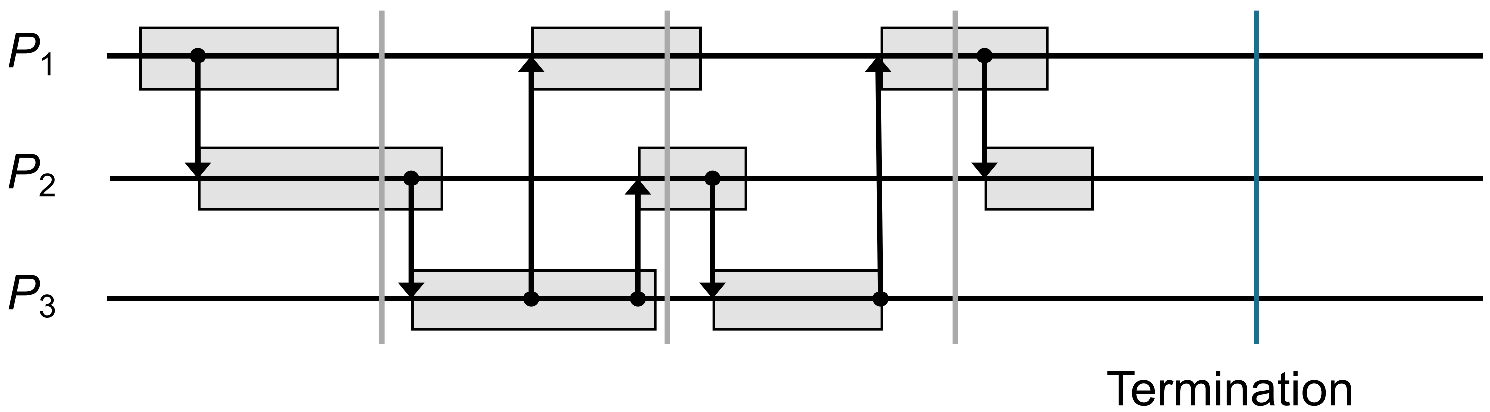
Asynchronous Model

- Processes are *active* or *passive*
- Only active processes can send basic messages
- An active process can become passive at any time
- A passive process can only be reactivated by a basic message
- Termination detection
  - Determine whether all processes are passive and no messages are on the way at a certain point in time



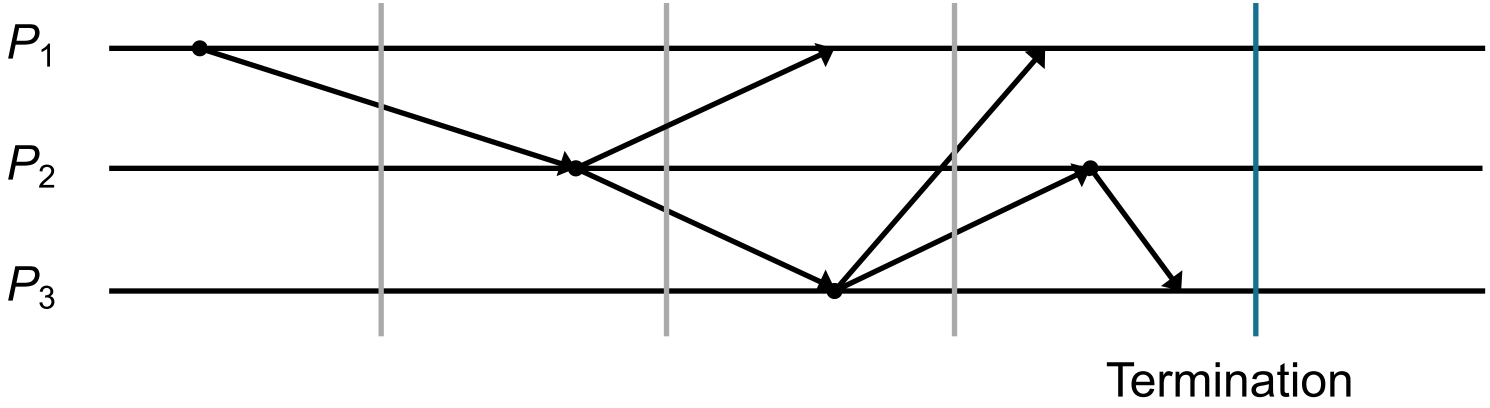
Process Model

- Difference to the asynchronous model
  - Messages have no delay
  - This can be imitated by synchronous communication
    - Sender is blocked until the message is received
  - ⇒ Perpendicular arrows in space-time-diagram
- Termination detection
  - determine, whether all processes are *passive* at a *certain* point in time



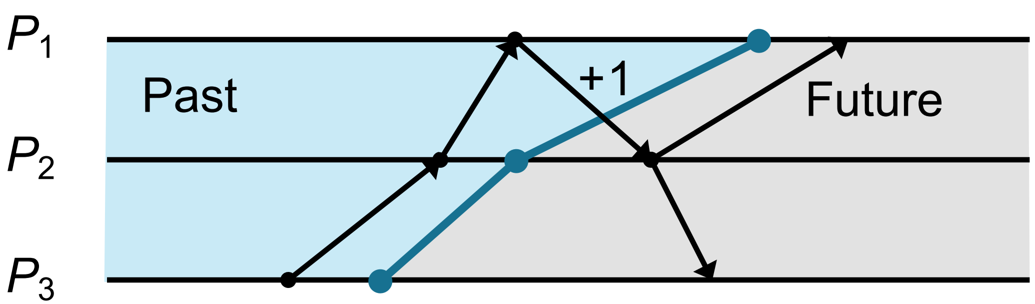
Atom Model

- Difference to the asynchronous model
  - Actions are atomic and need no time
- If a process receives a message, its local state changes accordingly to the respective action and it can send out messages instantly
- Termination detection
  - Determining whether no messages are on the way at a *certain* point in time



Simple Counting Algorithm

- An observer visits each node one after the other and separately sums up the basic messages sent and received
- Dissimilar sums indicate that a message was sent but has not arrived yet!
- Thus: If both sums are identical, no message is on its way and the basic algorithm is terminated, isn‘t it?



Observer visits every node and reports:

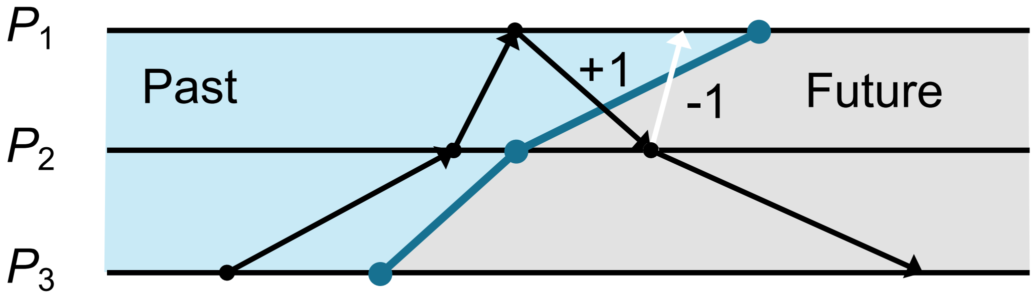
- 3 messages sent
- 2 messages received

⇒ sent - received= 1

⇒ no termination

Simple Counting Algorithm

- Unfortunately, this algorithm does not work because the condition that the counters are equal is only necessary but not sufficient for termination
- Example: Observer reports 3 messages received and 3 messages sent → phantom termination
  - This is due to the message that was sent in the „future“, but received in the „past“
  - It balances the summation difference



Observer visits each node and reports:
- 3 messages received
- 3 messages sent

⇒ sent - received= 0
⇒ Termination (false)

Solution Ideas

- Freeze the communication in the dangerous area
  ⇒ strong decrease of the concurrency of the system

- Subsequent check ⇒ Double Counting Algorithm

- Detecting or avoiding of inconsistent time cuts through logical time stamps ⇒ Time Zone Algorithm

- Differentiated Counting ⇒ Vector Algorithm

Freezing the System

- First wave freezes the system
  - No process accepts a message anymore
  - Messages arriving in the meantime are buffered and regarded as „still on the way“
  - ⇒ No messages are sent in the frozen system

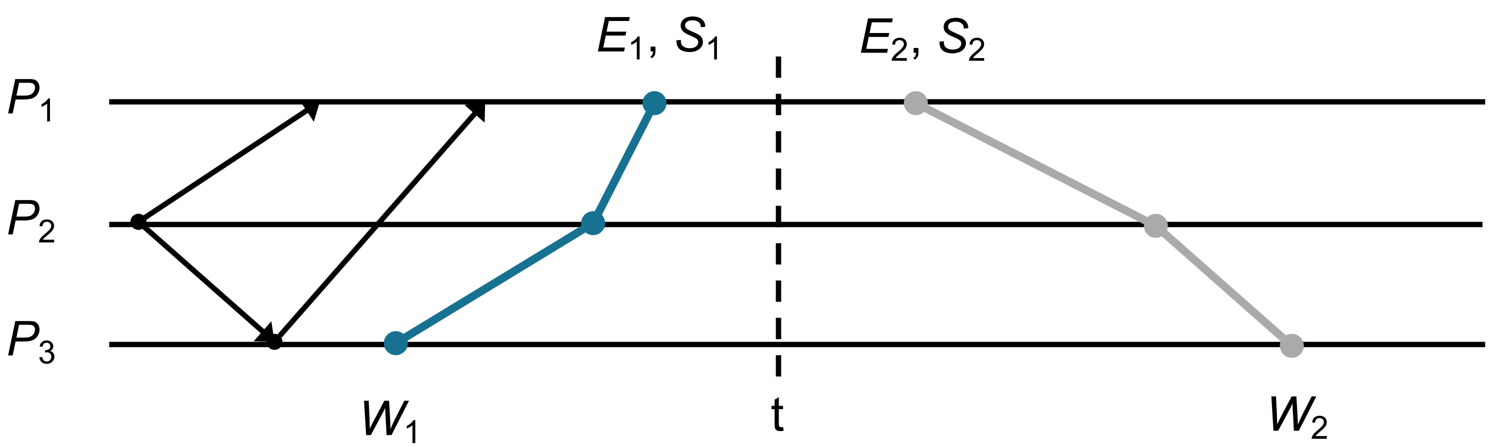
- Second wave sums up the messages sent and received
  - If both sums are equal, the system is terminated

- Third wave unfreezes the system again

- Obvious disadvantage of the algorithm is a massively decreased concurrency

Double Counting Algorithm

- An observer twice visits all nodes one after each other and determines the respective sums of the messages received and sent
- If all four sums are equal, the basic algorithm terminated: $E_1 = S_1 = E_2 = S_2$



Double Counting Algorithm – Characteristics

- If termination was not detected, use second wave of the previous round as the first wave of the new round
- Number of the control rounds is a priori not limited by the number of basic messages
  - There may be a very slow basic message; while it is on the way arbitrarily many control rounds may be started
  - The following variant removes this characteristic:
    a process containing a basic message, without sending a new one, starts a new round
- The double counting Algorithm is *re-entrant*
  - The local states of the visited processes are not changed
  - Thus, several concurrent initiators can test for possible termination at the same time

Control Topologies

The waves for the termination detection can be realized differently

Sequential Waves

- E.g., through the construction of a logical ring and two subsequent sequential ring circuits of a token which sums up the counter reading separately for both circulations
- Time and message complexity $O(n)$

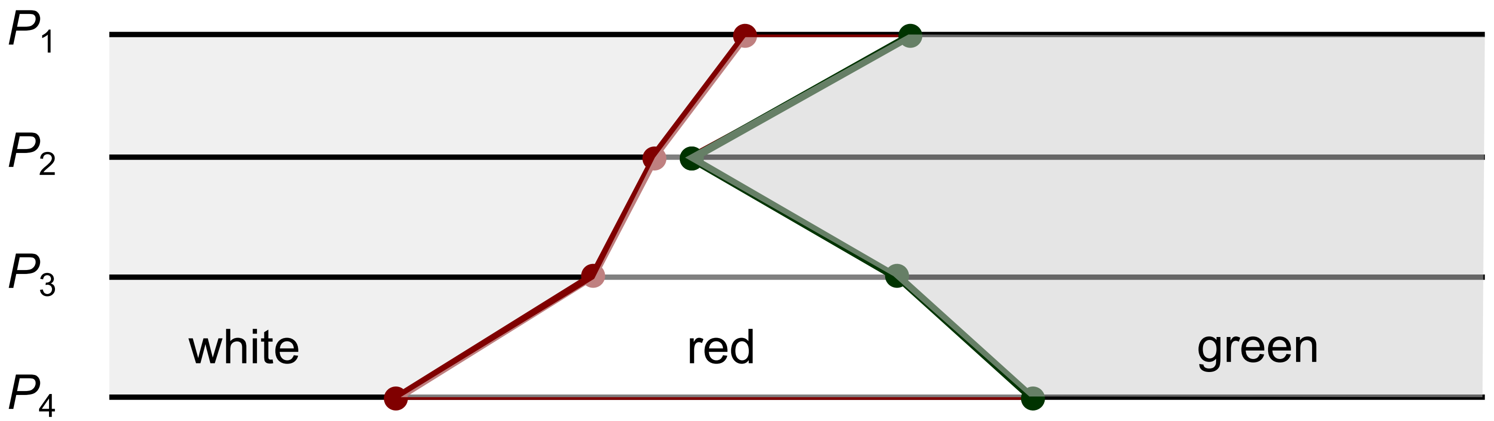
Parallel Waves

- E.g., through the construction of a span tree and two subsequent accumulations of the counter readings, *each* from the leafs to the initiator
- Achieves a better time complexity through parallel messages
- With well-balances trees time complexity is $O(\log n)$

Using the Echo-Algorithm

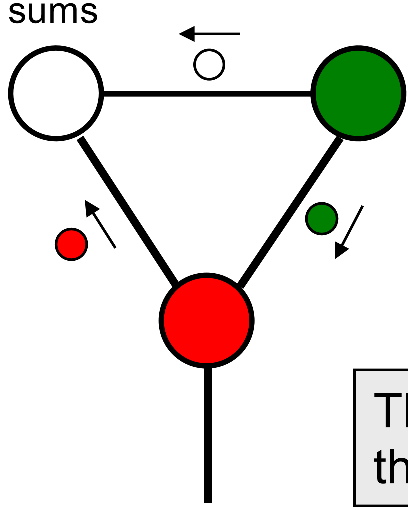
- Usage of the forth wave (nodes become red) and the back wave (nodes become green) of a single run of the echo-algorithm for the accumulation of the counter readings

- That works because a green node cannot have a white neighbor; thus, a green message cannot reach a white node



Usage of the Echo-Algorithm

- Remark: with constructed spanning trees, the usage of the forth wave and the back wave of a single instance does not work correctly!
- Assume, a spanning tree is constructed on a topology that is not a tree
- Then, there is at least one edge that is not part of the spanning tree
- Over this edge, a basic message can get from a green node to a white node balancing the difference of the sums

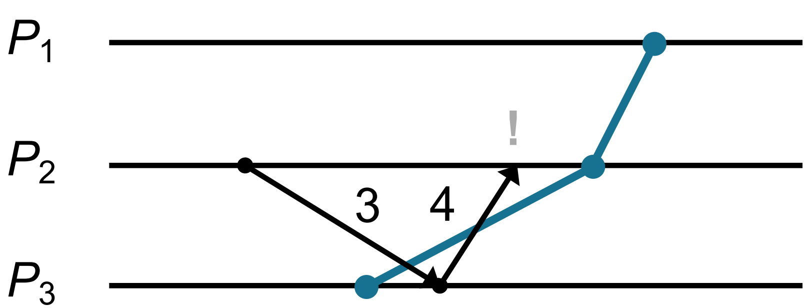


The basic message arrives before the explorer.

Time Zone Algorithm

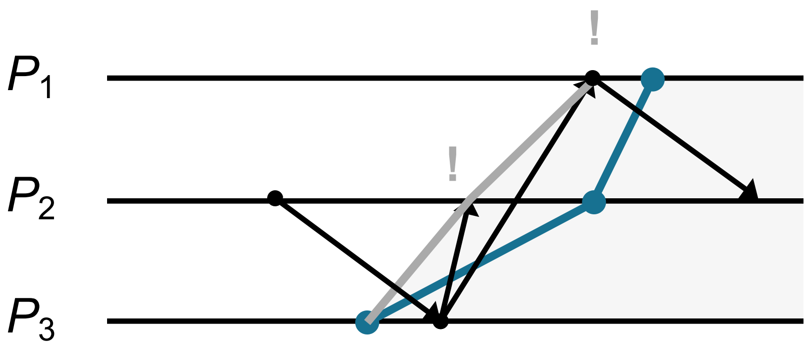
- Again, an observer visits all nodes one after the other and builds a send and receive sum, respectively
- But now, the visit of the observer increments a time zone counter on the visited node
- Current value of time zone counter is attached to every basic message from the sending node
- Thus, messages from the future can be recognized

  → They set a flag evaluated by the following wave and then reset
- Execute waves until both sums are equal and the flag is *not* set



Moving Forward the Intersection Line

- When the first message from the future is received on a node, the counters are saved
- If the observer passes by later, the saved counters are handed to it instead of the current counters
- Thus, the intersection line is moved forward
- Messages from the future then reside completely in the future



Time Zone Algorithm

- Recognizes inconsistent intersections
- Again, unlimited number of control rounds

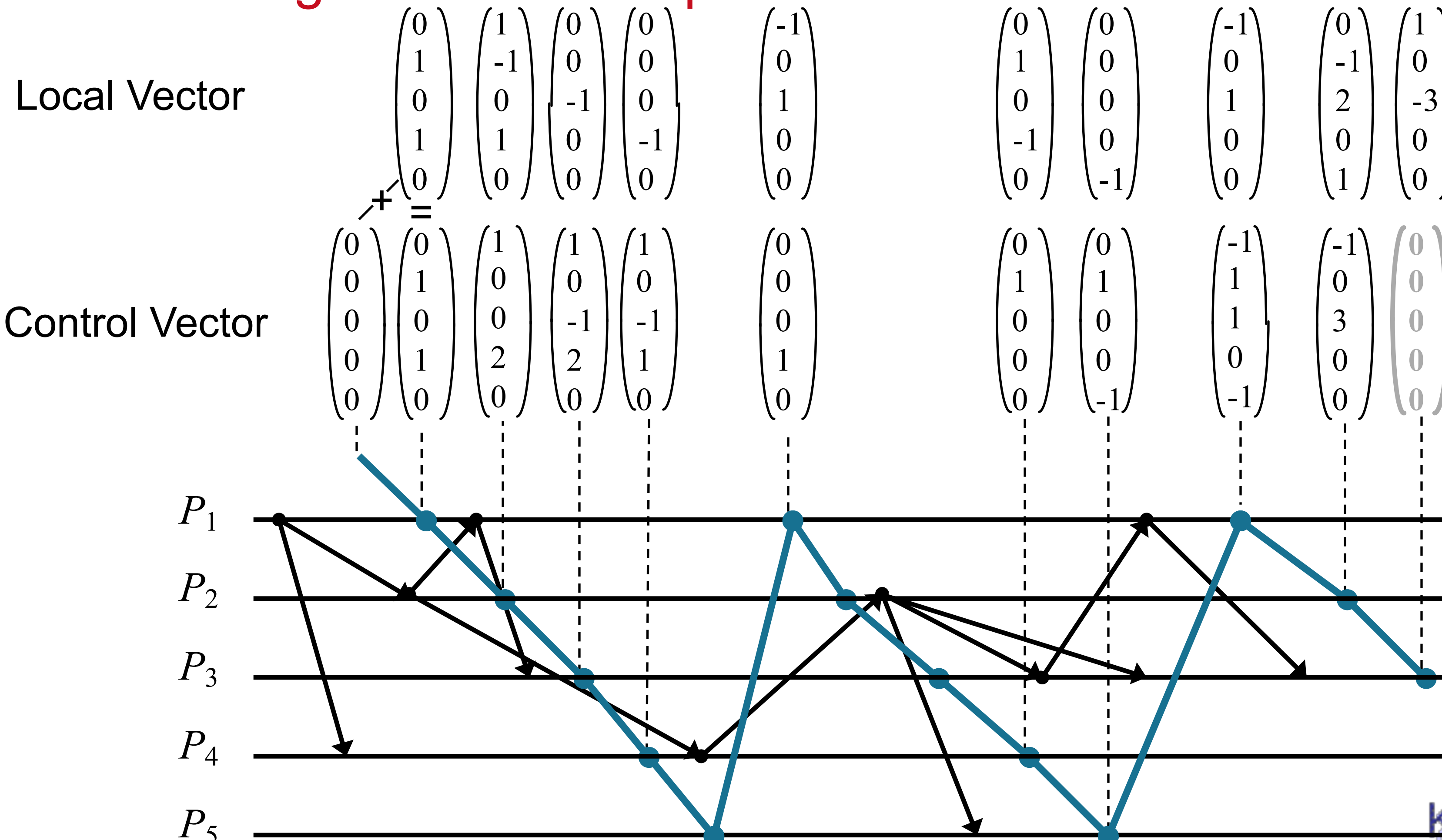
- Disadvantages
  - Basic messages are affected
  - Not re-entrant because the local state of nodes is changed
  - Waves of several initiators can disturb each other

- Double counting algorithm is more elegant and more universal

Vector Algorithm

- Each process P_A has a local vector V_A with length n that is initialized to the zero vector
- If P_A sends a basic message to P_B, $V_A[B]$ is increased by 1
- If P_B receives a basic message, $V_B[B]$ is decreased by 1
- A control vector C visits all processes subsequently.
- On a visit, the local vector V is added to C and V itself is set back to the zero vector
- Termination is detected, as soon as the control vector becomes a zero vector.
- Values in the vector can not lead to false-positive for termination

Vector Algorithm – Example Trace



Vector Algorithm

Improvement

- If the component of the next node to be visited is 0, skip that node

⇒ Potentially useless visits of such a node are avoided

Advantages

- Independent from the net topology
- Low number of control messages
- Basic messages remain untouched

Disadvantages

- Length of the control message (vector) → $O(n)$
- Algorithm is *not* re-entrant

Credit Algorithm

- Is not based on a wave algorithm, but on a global system invariant
- Primary process starts the distributed calculation with credit 1
- If a process sends a message, the message receives half of the current credit of the process
- If an active process receives a message, its credit increases by the credit of the message
- If a process becomes passive, it sends its current credit to the primary process

- The following presentation of the algorithm assumes the asynchronous model

Credit Algorithm

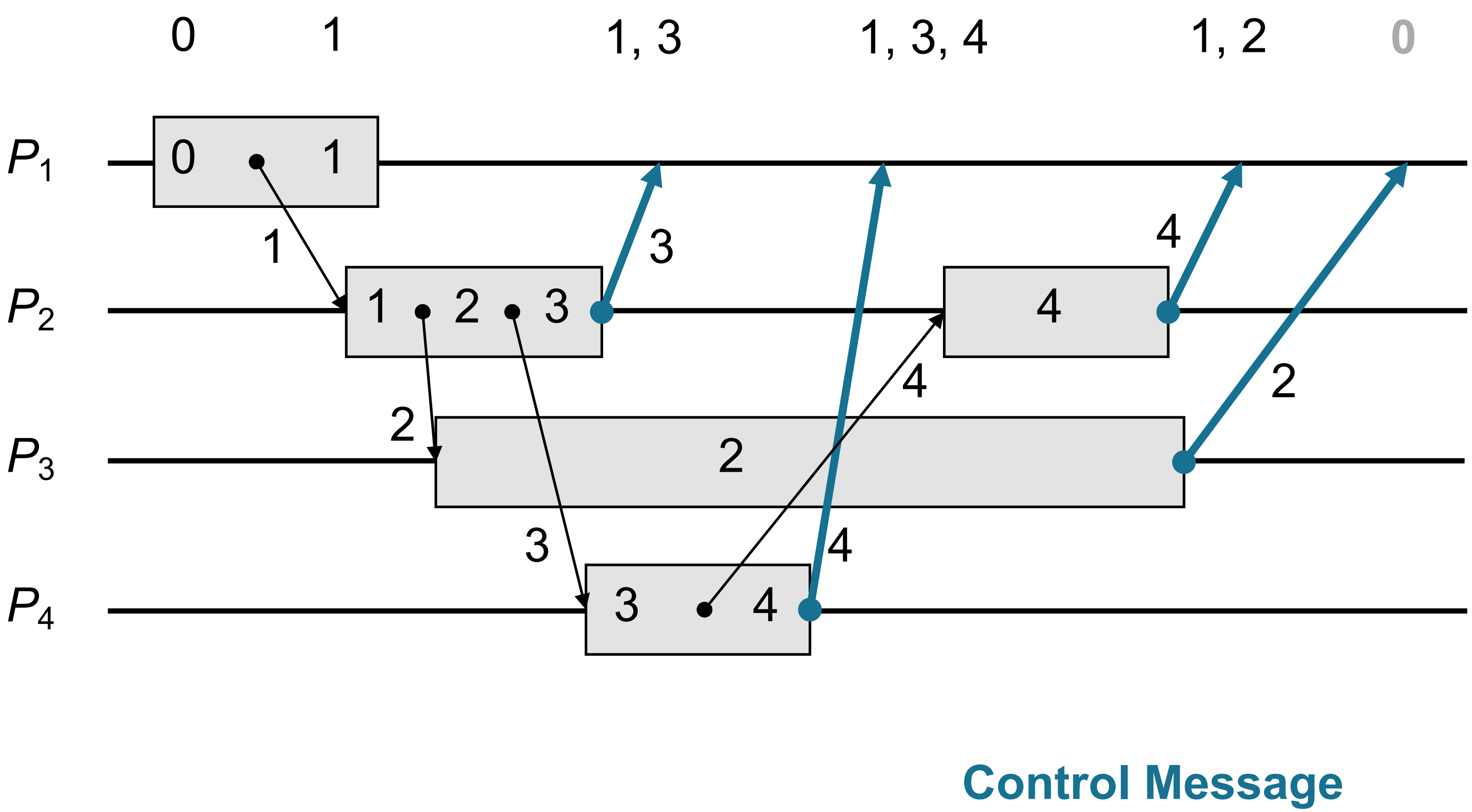
- Main Invariant
  - The credit sum is always 1
- Further Characteristics
  - Basic messages always carry a credit > 0 with them
  - Active processes always have a credit > 0
- Termination
  - If the primary process has credit 1 again

Representation of the Credit Portions

- Floating point number inconvenient
- Better: storage of the negative dual logarithm of the credit portion $c = -\, ld\ 2^{-d}$
  - $k = 1/1 \rightarrow c = 0$
  - $k = 1/2 \rightarrow c = 1$
  - $k = 1/4 \rightarrow c = 2$
  - …
- Halving of the credit $c := c + 1$

- Bit vector for the storage of the credit portions
- Recombination of credit portions through binary addition

Example Trace of the Credit Algorithm



Literature

1. F. Mattern. Verteilte Basisalgorithmen. Springer-Verlag, 1989. Kapitel 4: Verteilte Terminierung

2. F. Mattern. Algorithms for distributed termination detection. Distributed Computing, 2(3):161–-175, 1987.