Distributed Algorithms

Introduction

Univ.-Prof. Dr.-Ing. habil. Gero Mühl

Architecture of Application Systems (AVA)
Faculty for Informatics and Electrical Engineering
University of Rostock

Default for the lecture

Channel Characteristics

from the processes‘ view

- Reliability
  - Reliable — Every message sent arrives once and without changes
  - Unreliable — Errors may occur
    - Loss — Sent message is *not* received
    - Duplication — Sent message is received several times
    - Corruption — Sent message is received corrupted
    - Adding — A message that was *not* sent is received
- Order
  - Unordered — Messages can overtake each other
  - FIFO — Messages cannot overtake each other

Default for the lecture

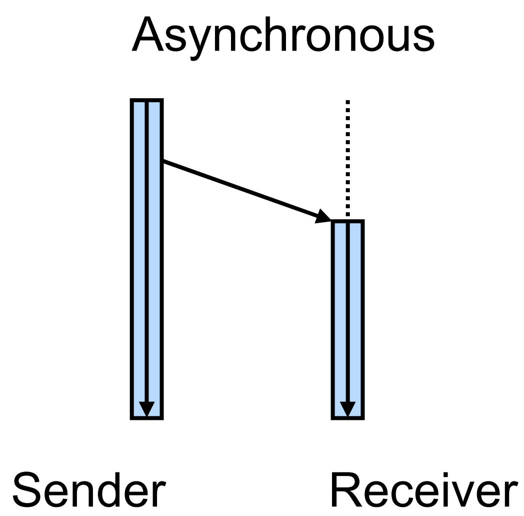
Channel Characteristics

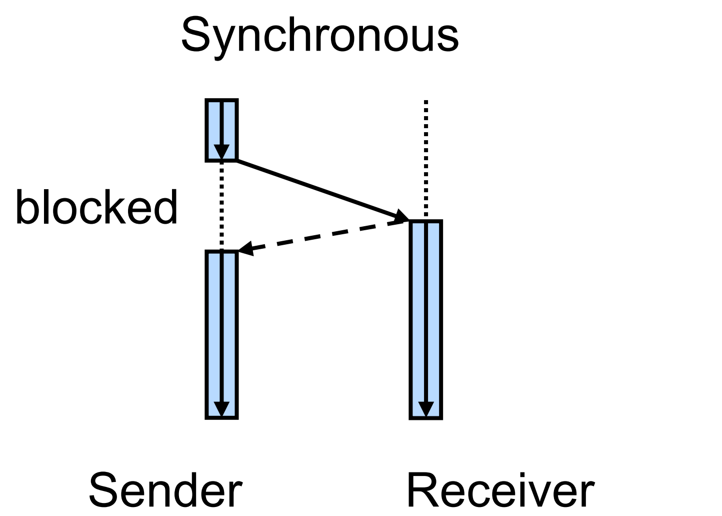
- Capacity
  - Unrestricted — An arbitrary number of messages can be within the channel is possible
  - Restricted — A maximum of *n* messages can be in the channel at the same time
    - When an overflow occurs, either messages are rejected or sender is blocked, until there is enough space
- Direction
  - Bidirectional — Send and receive possible in both directions
  - Unidirectional — Send and receive only possible in one direction

Default for the lecture

Channel Characteristics

> Synchronous vs. Asynchronous
>> Asynchronous — Sender of a message is *not* blocked
>> Synchronous — Sender of a message is blocked, until the receiver got the message





Distributed Algorithm

- Is executed on the nodes of the system as processes
- We consider usually only one process per node
- Thus, we often use both terms synonymously
- Different parts of the algorithm can run on different nodes
- The nodes communicate by exchanging messages over the communication channels

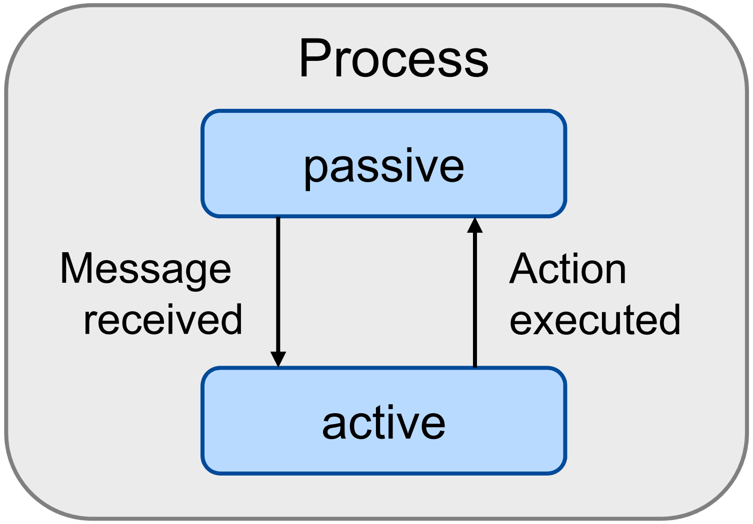
State of a Distributed Algorithm

- Each node has a local state, which consists of the local variables of the algorithm
- The state of a channel consists of the messages in it

- The state of a distributed algorithm at a certain time consists of
  - the state of the nodes at that time and
  - the state of the channels at that time

Processing Model

- Initial action is executed at initiators when starting the algorithm
- Each process waits passively for a message to arrive
- If a message arrives, a respective atomic action is executed
- Messages arriving in the meantime are buffered
- An action can
  - change the local state of the process and
  - send messages to other processes



P_1:
{Init}
    SEND(Ping) TO P_2;

P_2:
{RECEIVE (Ping) from P_1}
    SEND(Pong) TO P_1;

Synchronous Model

- Duration of actions and delay of messages are both bounded and the bounds are known
  - Example: Actions need no time, message delay is exactly one time unit.
  - In this case, a distributed algorithm can run in synchronized rounds easing its analysis.
- In the synchronous model, decisions can be made due to the course of time
  - For example, if a message is not received within the maximal delay, one can conclude securely that an error occurred (node crash, message loss etc.)

Asynchronous Model

- Actions and messages can last arbitrarily long or the bounds are unknown.
- From the course of time no reliable information can be concluded
  - For example, if a message is not received before a timeout occurs, this can have several reasons
    1. Message delay is longer than usual
    2. Message got lost
    3. Sending process has sent the message later than usual
    4. Sending process has crashed before it could send the message

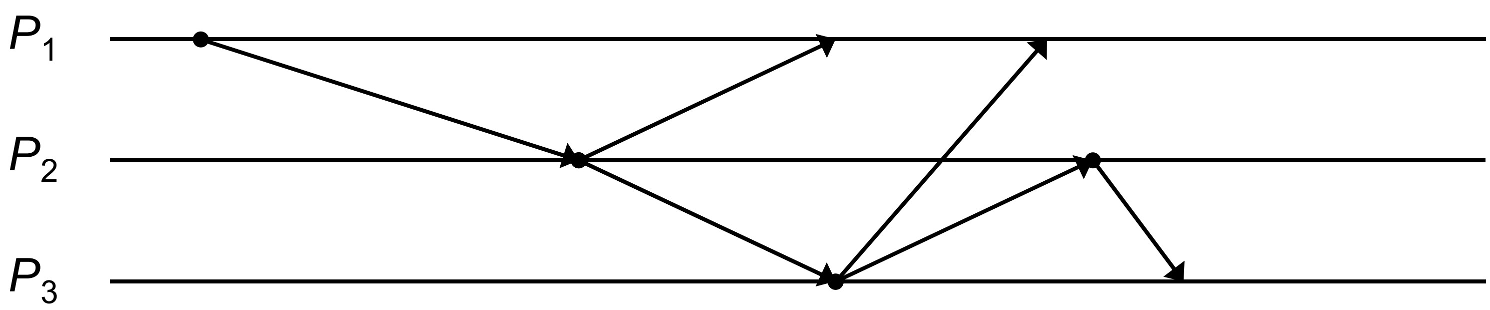
Synchronous vs. Asynchronous Model

- Algorithms for problems in *synchronous* systems
  - Often simple algorithms exist
  - However, they are mostly not directly applicable, because they make unrealistic assumptions
- Algorithms for problems in *asynchronous* systems
  - For some problems, no algorithm exists
  - If an algorithm exist, it is often complex and inefficient
  - But, if there is an algorithm, it is usually well applicable

Atom Model

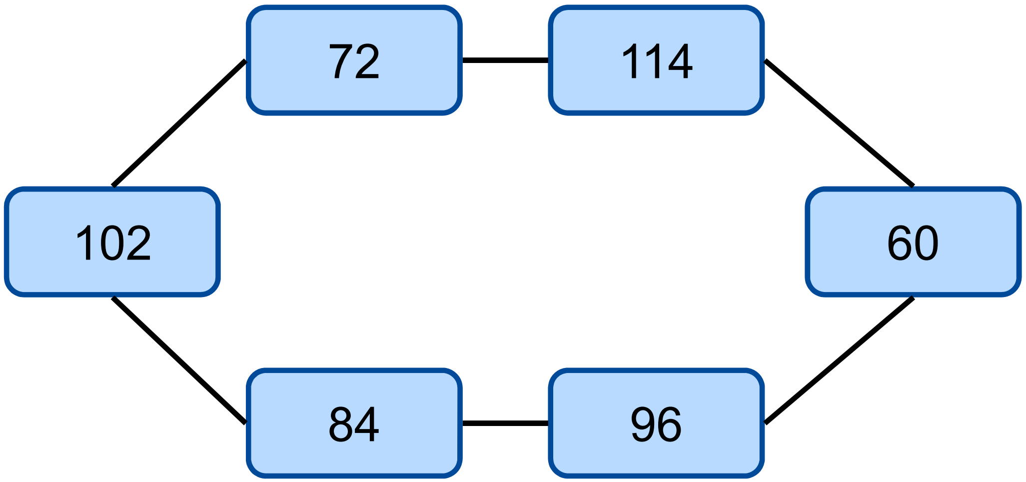
- The atom model is a partially synchronized model
- The difference compared to the asynchronous model is that actions are timeless
- If a process receives a message, its local state changes correspondingly to the action and it can send messages

- Time Space Diagram: Graphical representation of (local events and) interactions of all processes



Example: Distributed Maximum-Algorithm

- A group of philosophers with different ages is sitting around a table and wants to determine the oldest among them
- Every philosopher can only communicate with his two neighbors



Example: Distributed Maximum-Algorithm

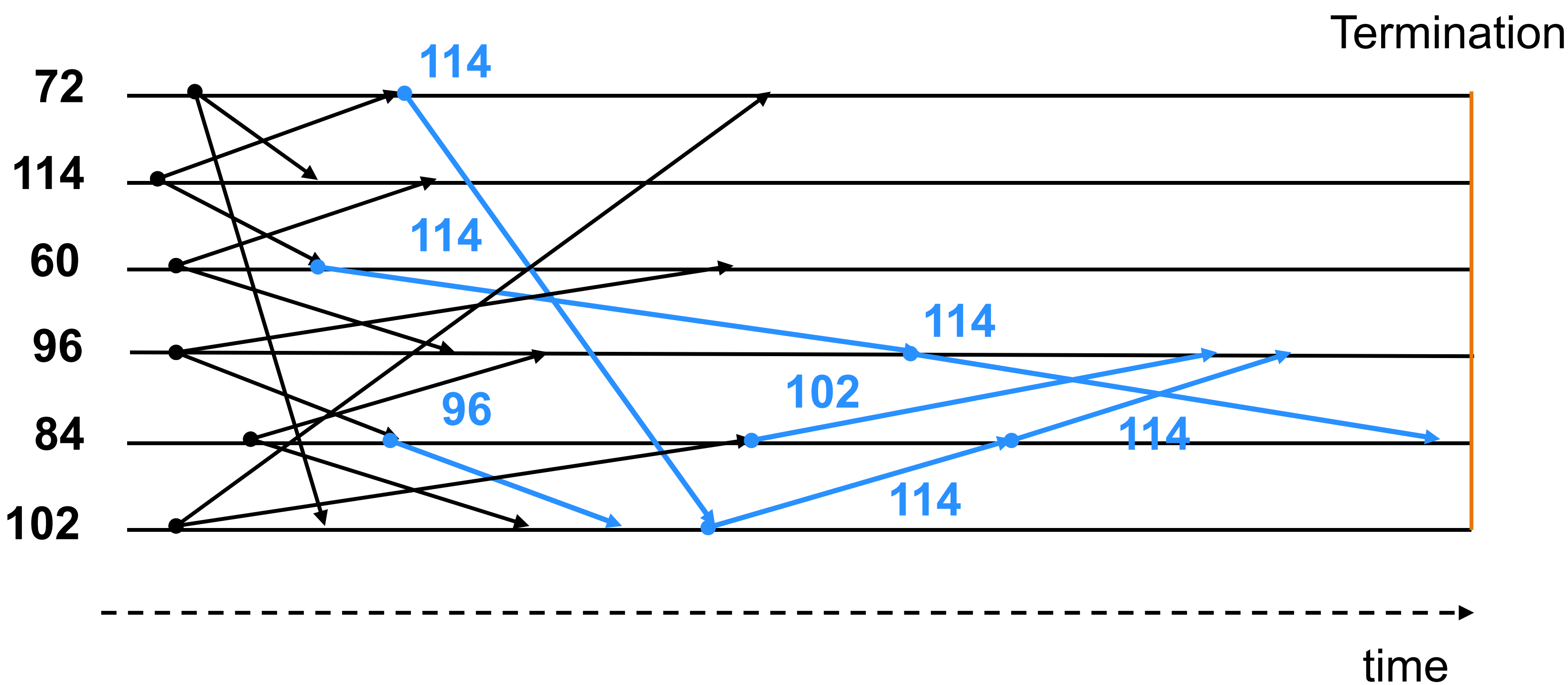
```
{INIT}:// is executed by every philosopher
   myMax := <own age>;
   SEND(<left neighbor>, myMax);
   SEND(<right neighbor>, myMax);

{RECEIVE (<left neighbor>, neighborMax)}
   IF (neighborMax > myMax) THEN
      myMax := neighborMax;
      SEND(<right neighbor>, myMax);
   ENDIF

{RECEIVE (<right neighbor, neighborMax)}
   IF (neighborMax > myMax) THEN
      myMax := neighborMax;
      SEND(<left neighbor>, myMax);
   ENDIF
```

Atomic Action

Example: Distributed Maximum-Algorithm

> Due to different message latencies, traces can vary

> Exemplary trace



Example: Distributed Maximum-Algorithm

- Are there several possible traces for one input?
- Does every trace lead to the correct result?
- How many messages are needed at least and when does this case occur?
- How many messages are needed at most or on average? What does that depend on?
- How should the philosophers know to start the algorithm?
- Is it sufficient if an arbitrary philosopher starts the algorithm?

Example: Distributed Maximum-Algorithm

> Does the algorithm always terminate?
> How long does it take at least or at most or on average, until the algorithm terminates?
> How does a single philosopher recognize termination?
> What happens if several philosophers have the same age?

Characteristics of Distributed Algorithms

- Deterministic vs. non-deterministic
  - Deterministic
    - The same initial state always results in the same trace
  - Non-deterministic
    - Different traces are possible for the same initial state
- Distributed algorithms are usually non-deterministic
  - This is due to the unpredictable message delay and the varying processing speed

Characteristics of Distributed Algorithms

- Determined vs. not determined
  - Determined
    - The same initial state always give the same result
    - Deterministic algorithms are always determined
    - Non-deterministic algorithms can be determined (e.g. Quicksort with randomly chosen Pivot-element)
  - Not determined
    - Different results are possible for the same initial state
- There are both determined and not determined distributed algorithms

Characteristics of Distributed Algorithms

- Terminating algorithm
  - Algorithm terminates for each (valid) initial state after a limited number of steps
- Partially correct algorithm
  - If the algorithm terminates, it always deliver a correct result
- Totally correct algorithm
  - It is both terminating and partially correct
- Distributed algorithms shall usually be totally correct
  - But not for every problem to be solved there is a totally correct algorithm

Complexity of Distributed Algorithms

- Message complexity
- Variable time complexity
  - Actions are timeless
  - Messages need at most one time unit
- Unit time complexity
  - Actions are timeless
  - Messages need exactly one time unit
- Not always *variable time complexity* ≤ *unit time complexity*
  - The reason is that the variable time complexity allows traces which cannot occur with unit time complexity
  - These traces may be those that cause the algorithm to terminate only after some additional steps

Exemplary Exam Questions

Definitions and Basics

1. What is a distributed system and what are the main differences to a parallel computer?
2. Specify what potentially objectives are pursued with the use of distributed systems!
3. Explain the essential characteristics of a distributed system and the potentially resulting problems!
4. Why it is usually easier to implement algorithms for centralized systems than for distributed systems?
5. Give some examples for conceptual problems in distributed systems!

Exemplary Exam Questions

Basic Models

1. How can a distributed system be modeled as an abstract graph?
2. Name some basic network topologies and explain their properties!
3. What are the characteristics of a small-world network?
4. What is a scale-free network?
5. Describe some of the properties of communication channels!
6. What is a distributed algorithm and how is the state of a distributed algorithm defined?

Exemplary Exam Questions

Basic Models

7. What are the differences between the synchronous and the asynchronous system model?
8. Describe the atom model!
9. Explain the following characteristics of an algorithm:
   determined / not determined,
   deterministic / non-deterministic,
   partially correct, terminating, totally correct!
10. How do the variable time complexity and the unit time complexity differ?

Literature

1. S. Strogatz. Exploring complex networks. Nature, 410:268--276, 2001.
2. D. Watts and S. Strogatz. Collective Dynamics of 'Small-World' Networks. Nature, 393:440--442, 1998.
3. Akkoyunlu, E. A., Ekanadham, K., and Huber, R. V. Some constraints and tradeoffs in the design of network communications. In Proceedings of the Fifth ACM Symposium on Operating Systems Principles. ACM, New York, NY, 67-74, 1975.

Thank you for your kind attention!

Univ.-Prof. Dr.-Ing. habil. Gero Mühl

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