Distributed Algorithms

Lecturer:
Vladimir Zakharov

LECTURE COURSE PROGRAM

This course is

- not about network technologies, their implementations and applications,

LECTURE COURSE PROGRAM

This course is

- not about network technologies, their implementations and applications,
- not about parallel algorithms for the solution of computing problems,

LECTURE COURSE PROGRAM

This course is

not about network technologies, their implementations and applications,

not about parallel algorithms for the solution of computing problems,

but

- about mathematical (formal) models, and
- mathematical techniques

used for the development of algorithms which provide certain specific functionality of distributed systems — collections of autonomous computing processes interacting via shared memory or message passing.

Lecture slides are available

at the www page

of mathematical cybernetics dep.

mk.cs.msu.ru

LECTURE COURSE PROGRAM

Part 1. Formal models of distributed systems

Lecture 1. Distributed systems and their generic features. Distributed systems architecture. ISO Open System Interaction standard. Algorithmic problems of distributed computing systems. Specific features of distributed algorithms.

LECTURE COURSE PROGRAM

Part 1. Formal models of distributed systems

Lecture 1. Distributed systems and their generic features. Distributed systems architecture. ISO Open System Interaction standard. Algorithmic problems of distributed computing systems. Specific features of distributed algorithms.

Lecture 2. Formal models of distributed systems. Transition systems. Synchronous and asynchronous message passing. Fairness properties. Dependent and independent events. Causal order of events. Equivalence of execution. Distributed computations. Logical clocks.

LECTURE COURSE PROGRAM

Part 2. Communication Protocols

Lecture 3. Communication Protocols. Errors in message transmission. Symmetric sliding window protocol: protocol design. Mathematical techniques for proving correctness of distributed algorithms. Correctness of sliding window protocol. Implementation of sliding window protocol. Alternating Bit Protocol.

LECTURE COURSE PROGRAM

Part 2. Communication Protocols

Lecture 3. Communication Protocols. Errors in message transmission. Symmetric sliding window protocol: protocol design. Mathematical techniques for proving correctness of distributed algorithms. Correctness of sliding window protocol. Implementation of sliding window protocol. Alternating Bit Protocol.

Lecture 4. A timer-based communication protocol. Protocol design. Correctness of timer-based protocol. Implementation of timer-based protocol.

LECTURE COURSE PROGRAM

Part 3. Routing algorithms

Lecture 5. Routing problem. Destination-based routing. The all-pairs shortest-path problem. Floyd–Warshall algorithm. Toueg's shortest-path algorithm. Merlin–Segall algorithm. Chandy–Misra algorithm.

LECTURE COURSE PROGRAM

Part 3. Routing algorithms

Lecture 5. Routing problem. Destination-based routing. The all-pairs shortest-path problem. Floyd–Warshall algorithm. Toueg's shortest-path algorithm. Merlin–Segall algorithm. Chandy–Misra algorithm.

Lecture 6. Netchange algorithm. Design of the algorithm. Correctness of Netchange algorithm. Routing with compact routing tables: interval, prefix, hierarchical routing

LECTURE COURSE PROGRAM

Part 4. Wave and Election Algorithms

Lecture 7. Wave algorithms: definition, basic properties, applications. Ring algorithm. Tree algorithm. Echo algorithm. Phase algorithm. Finn's algorithm.

LECTURE COURSE PROGRAM

Part 4. Wave and Election Algorithms

Lecture 7. Wave algorithms: definition, basic properties, applications. Ring algorithm. Tree algorithm. Echo algorithm. Phase algorithm. Finn's algorithm.

Lecture 8. Traversal algorithms. Tarry's traversal algorithm. Depth first search algorithm. Awerbuch's algorithm. Sidon's algorithm.

LECTURE COURSE PROGRAM

Part 4. Wave and Election Algorithms

Lecture 7. Wave algorithms: definition, basic properties, applications. Ring algorithm. Tree algorithm. Echo algorithm. Phase algorithm. Finn's algorithm.

Lecture 8. Traversal algorithms. Tarry's traversal algorithm. Depth first search algorithm. Awerbuch's algorithm. Sidon's algorithm.

Lecture 9. Leader election. Elections and waves. Leader election on the rings. Chang–Roberts election algorithm. Peterson/Dolev–Klawe–Rodeh algorithm.

LECTURE COURSE PROGRAM

Part 4. Wave and Election Algorithms

Lecture 7. Wave algorithms: definition, basic properties, applications. Ring algorithm. Tree algorithm. Echo algorithm. Phase algorithm. Finn's algorithm.

Lecture 8. Traversal algorithms. Tarry's traversal algorithm. Depth first search algorithm. Awerbuch's algorithm. Sidon's algorithm.

Lecture 9. Leader election. Elections and waves. Leader election on the rings. Chang–Roberts election algorithm. Peterson/Dolev–Klawe–Rodeh algorithm.

Lecture 10. Leader election in the arbitrary networks. Gallager–Humblet–Spira algorithm. Korach–Kutten–Moran algorithm.

LECTURE COURSE PROGRAM

Part 5. Termination Detection and Fault Tolerance.

Lecture 11. Termination and deadlock detection. Dijkstra–Scholten algorithm. Shavit–Francez algorithm. Wave-based solutions. Safra's algorithm. Credit-recovery algorithm.

LECTURE COURSE PROGRAM

Part 5. Termination Detection and Fault Tolerance.

Lecture 11. Termination and deadlock detection. Dijkstra–Scholten algorithm. Shavit–Francez algorithm. Wave-based solutions. Safra's algorithm. Credit-recovery algorithm.

Lecture 12. Fault tolerance of distributed systems. The impossibility of building robust asynchronous systems. Synchronous robust decision-making algorithms.

LITERATURE

1. **Lynch N.** **Distributed Algorithms**. Morgan Kaufmann (March 1, 1996), 906 pp.

LITERATURE

1. **Lynch N.** **Distributed Algorithms**. Morgan Kaufmann (March 1, 1996), 906 pp.
2. **Fokkink W.** **Distributed Algorithms: Intuitive Approach**. The MIT Press (December 6, 2013), 231 pp.

LITERATURE

1. **Lynch N.** **Distributed Algorithms**. Morgan Kaufmann (March 1, 1996), 906 pp.
2. **Fokkink W.** **Distributed Algorithms: Intuitive Approach**. The MIT Press (December 6, 2013), 231 pp.
3. **Tel G.** **Introduction to Distributed Algorithms**. Cambridge University Press (2000), 596 pp.

LITERATURE

1. **Lynch N.** **Distributed Algorithms**. Morgan Kaufmann (March 1, 1996), 906 pp.
2. **Fokkink W.** **Distributed Algorithms: Intuitive Approach**. The MIT Press (December 6, 2013), 231 pp.
3. **Tel G.** **Introduction to Distributed Algorithms**. Cambridge University Press (2000), 596 pp.
4. **Santoro N.** **Design and Analysis of Distributed Algorithms**. Willey-Interscience (2007), 589 pp.

Lecture 1.

Distributed systems and their generic properties.
Architecture of distributed systems.
Algorithmic problems of distributed computing systems.
Specific features of distributed algorithms.

Distributed Systems

A **distributed system** is any collection of **interacting autonomous** information processing agents (computers, processors, processes).

To be qualified as "autonomous", the nodes must at least be equipped with their own private control; thus, a parallel computer of the single-instruction, multiple-data (SIMD) model does not qualify as a distributed system.

To be qualified as "interacting", the components of a distributed system must be able to exchange information.

Distributed Systems

A **distributed system** is any collection of **interacting autonomous** information processing agents (computers, processors, processes).

To be qualified as "autonomous", the nodes must at least be equipped with their own private control; thus, a parallel computer of the single-instruction, multiple-data (SIMD) model does not qualify as a distributed system.

To be qualified as "interacting", the components of a distributed system must be able to exchange information.

Interacting computers, processors, and programs are called the **nodes** of a distributed system.

Distributed Systems

A **distributed system** is any collection of **interacting autonomous** information processing agents (computers, processors, processes).

To be qualified as "autonomous", the nodes must at least be equipped with their own private control; thus, a parallel computer of the single-instruction, multiple-data (SIMD) model does not qualify as a distributed system.

To be qualified as "interacting", the components of a distributed system must be able to exchange information.

Interacting computers, processors, and programs are called the **nodes** of a distributed system.

An environment which provides an interaction between the nodes of a distributed system is called a **communication subsystem**.

Distributed Systems

A **distributed system** is any collection of **interacting autonomous** information processing agents (computers, processors, processes).

To be qualified as "autonomous", the nodes must at least be equipped with their own private control; thus, a parallel computer of the single-instruction, multiple-data (SIMD) model does not qualify as a distributed system.

To be qualified as "interacting", the components of a distributed system must be able to exchange information.

Interacting computers, processors, and programs are called the **nodes** of a distributed system.

An environment which provides an interaction between the nodes of a distributed system is called a **communication subsystem**.

A **distributed algorithm** is an algorithm performed by a distributed system.

Distributed Systems

Motivation for the development and use of distributed systems.

1. Information exchange.

Distributed Systems

Motivation for the development and use of distributed systems.

1. Information exchange.
2. Resource sharing.

Distributed Systems

Motivation for the development and use of distributed systems.

1. Information exchange.
2. Resource sharing.
3. Increasing reliability through replication.

Distributed Systems

Motivation for the development and use of distributed systems.

1. Information exchange.
2. Resource sharing.
3. Increasing reliability through replication.
4. Increasing performance through parallelization.

Distributed Systems

Motivation for the development and use of distributed systems.

1. Information exchange.
2. Resource sharing.
3. Increasing reliability through replication.
4. Increasing performance through parallelization.
5. Simplification of design through specialization.

Distributed Systems

Motivation for the development and use of distributed systems.

1. Information exchange.
2. Resource sharing.
3. Increasing reliability through replication.
4. Increasing performance through parallelization.
5. Simplification of design through specialization.

What other application of distributed systems do you know?

Distributed systems

Examples of distributed systems:

- wide-area communicating network,

Distributed systems

Examples of distributed systems:

- wide-area communicating network,
- local-area network,

Distributed systems

Examples of distributed systems:

- wide-area communicating network,
- local-area network,
- multi-processor computing system,

Distributed systems

Examples of distributed systems:

- wide-area communicating network,
- local-area network,
- multi-processor computing system,
- multi-thread program,

Distributed systems

Examples of distributed systems:

- wide-area communicating network,
- local-area network,
- multi-processor computing system,
- multi-thread program,
- cloud computing system.

Distributed systems

Examples of distributed systems:

- wide-area communicating network,
- local-area network,
- multi-processor computing system,
- multi-thread program,
- cloud computing system.

What other examples of distributed systems do you know?

Computer Networks

Computer Networks is a collection of computers supplied with a communication mechanisms by means of which the computers can exchange information. This exchange takes place by sending and receiving **messages** (data packets).

Depending on the distance between the computers and their ownership, computer networks are divided into **wide-area network** (global networks) and **local-area networks** (local networks).

Global Networks

A wide-area network usually connects computers owned by different organizations (industries, universities, etc.).

The physical distance between the nodes is typically several kilometers or more.

Each node of such a network is a complete computer installation, including all the peripherals and a considerable amount of application software.

The main object of a wide-area network is the exchange of information between users at the various nodes.

Local Networks

A local-area network usually connects computers owned by a single organization.

The physical distance between the nodes is typically a few hundred metres or less.

A node of such a network in most cases is a workstation, a file server, or a printer server, i.e., a relatively small station dedicated to a specific function within the organization.

The main objects of a local-area network are usually information exchange and resource sharing.

Local Networks

A local-area network usually connects computers owned by a single organization.

The physical distance between the nodes is typically a few hundred metres or less.

A node of such a network in most cases is a workstation, a file server, or a printer server, i.e., a relatively small station dedicated to a specific function within the organization.

The main objects of a local-area network are usually information exchange and resource sharing.

But is there nowadays any difference between wide-area and local networks?

Computer Networks

The main differences between global and local networks

1. Reliability parameters.

Computer Networks

The main differences between global and local networks

1. Reliability parameters.
2. Communication time.

Computer Networks

The main differences between global and local networks

1. Reliability parameters.
2. Communication time.
3. Hardware and software homogeneity.

Computer Networks

The main differences between global and local networks

1. Reliability parameters.
2. Communication time.
3. Hardware and software homogeneity.
4. Security requirements.

Wide-area networks

Wide-area networks are always organized as **point-to-point** networks. This means that communication between a pair of nodes takes place by a mechanism particular to these two nodes.

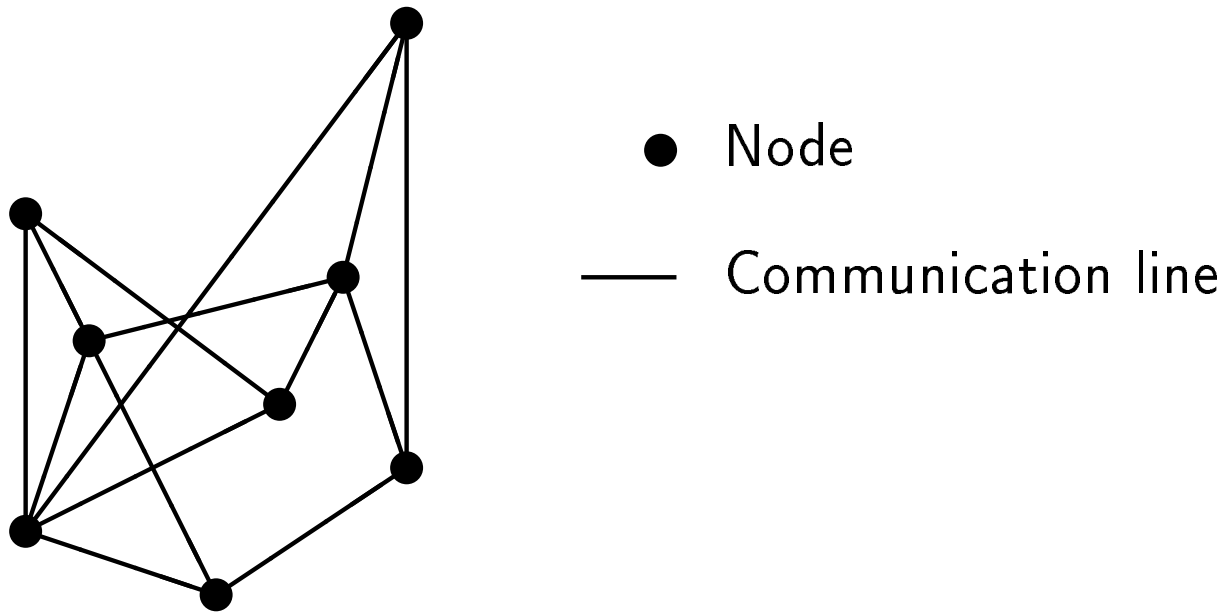


Рис.: An example of a point-to-point network.

Wide-area networks

Algorithmic problems

1. Reliability of point-to-point data exchange.

Wide-area networks

Algorithmic problems

1. Reliability of point-to-point data exchange.
2. Selection of communication paths.

Wide-area networks

Algorithmic problems

1. Reliability of point-to-point data exchange.
2. Selection of communication paths.
3. Congestion control.

Wide-area networks

Algorithmic problems

1. Reliability of point-to-point data exchange.
2. Selection of communication paths.
3. Congestion control.
4. Reliability of long-range data exchange.

Wide-area networks

Algorithmic problems

1. Reliability of point-to-point data exchange.
2. Selection of communication paths.
3. Congestion control.
4. Reliability of long-range data exchange.
5. Deadlock prevention.

Wide-area networks

Algorithmic problems

1. Reliability of point-to-point data exchange.
2. Selection of communication paths.
3. Congestion control.
4. Reliability of long-range data exchange.
5. Deadlock prevention.
6. Security.

Wide-area networks

Algorithmic problems

1. Reliability of point-to-point data exchange.
2. Selection of communication paths.
3. Congestion control.
4. Reliability of long-range data exchange.
5. Deadlock prevention.
6. Security.

What other algorithmic problems for wide-area networks do you know?

Local-area Networks

The communication between nodes of a local network takes place via a single bus-like mechanism to which all nodes are connected.

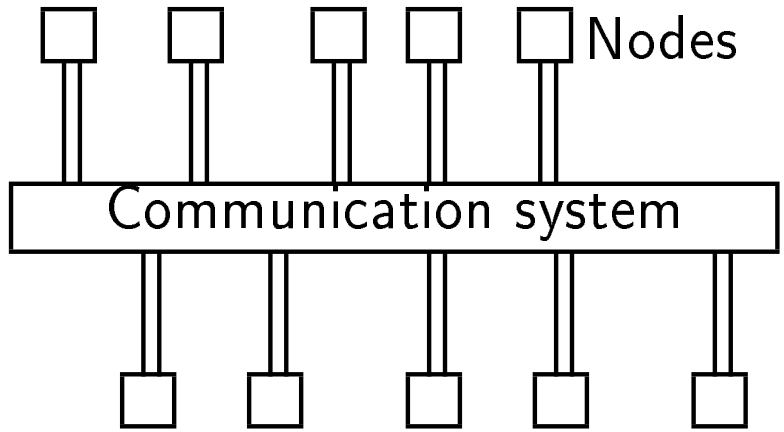


Рис.: A bus-connected network.

Local Networks

Algorithmic problems for local networks

1. Broadcasting and synchronization.

Local Networks

Algorithmic problems for local networks

1. Broadcasting and synchronization.
2. Leader election.

Local Networks

Algorithmic problems for local networks

1. Broadcasting and synchronization.
2. Leader election.
3. Termination detection.

Local Networks

Algorithmic problems for local networks

1. Broadcasting and synchronization.
2. Leader election.
3. Termination detection.
4. Resource allocation.

Local Networks

Algorithmic problems for local networks

1. Broadcasting and synchronization.
2. Leader election.
3. Termination detection.
4. Resource allocation.
5. Mutual exclusion.

Local Networks

Algorithmic problems for local networks

1. Broadcasting and synchronization.
2. Leader election.
3. Termination detection.
4. Resource allocation.
5. Mutual exclusion.
6. Deadlock detection and resolution.

Local Networks

Algorithmic problems for local networks

1. Broadcasting and synchronization.
2. Leader election.
3. Termination detection.
4. Resource allocation.
5. Mutual exclusion.
6. Deadlock detection and resolution.

What other algorithmic problems for local networks do you know?

Multiprocessor Computers

A **multiprocessor computer** consists of several processors installed on a small scale area.

This type of computer system is distinguished from local-area networks by the following criteria.

1. Its processors are homogeneous, i.e., they are identical as hardware.

2. The physical scale of the machine is very small, typically of the order of one meter or less.

3. The processors are intended to be used together in one computation (either to increase the speed or to increase the reliability).

If the main design objective of the multiprocessor computer is to improve the speed of computation, it is often called a parallel computer .
If its main design objective is to increase the reliability, it is often called a replicated system .

Multiprocessor Computers

The communication between the processors can take place either via a bus or via point-to-point connections.

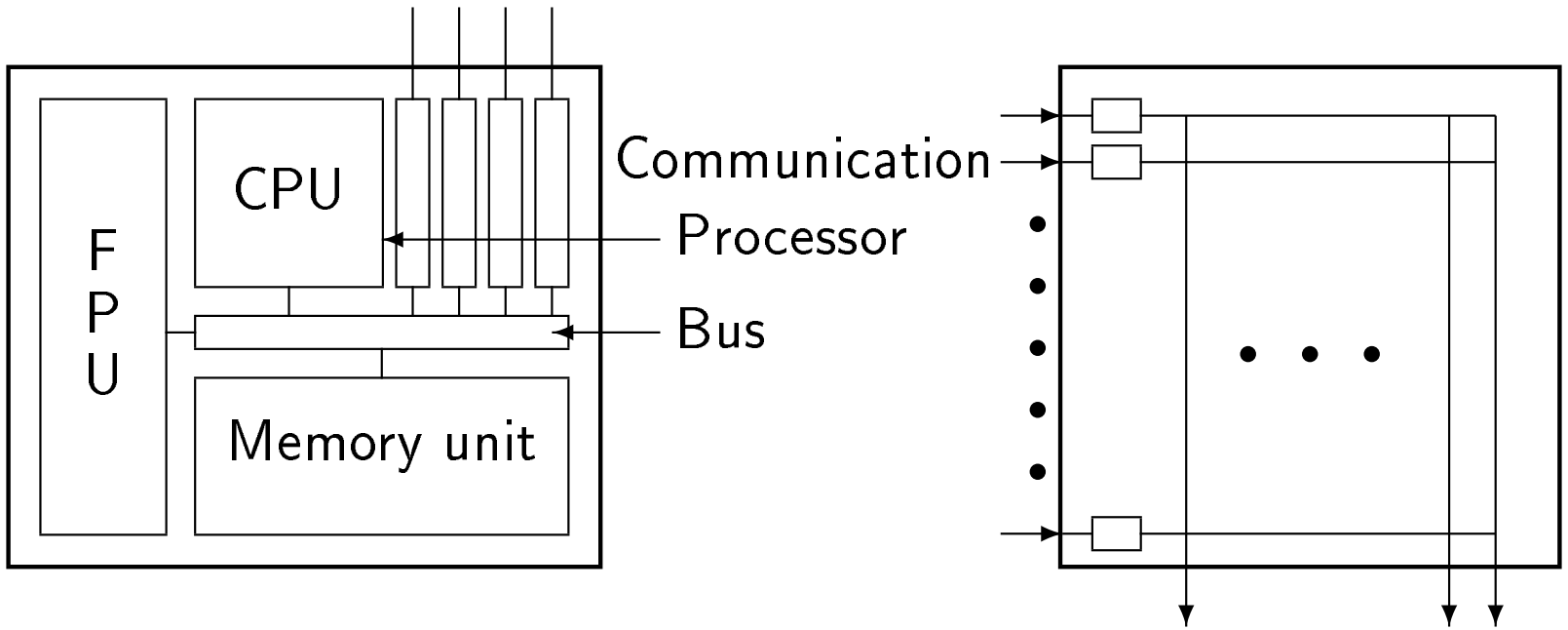


Рис.: A Transputer and a routing chip.

Multiprocessor computers

Algorithmic problems for multiprocessor computers

1. Implementation of a message-passing system.

Multiprocessor computers

Algorithmic problems for multiprocessor computers

1. Implementation of a message-passing system.
2. Implementation of a virtual shared memory.

Multiprocessor computers

Algorithmic problems for multiprocessor computers

1. Implementation of a message-passing system.
2. Implementation of a virtual shared memory.
3. Load balancing.

Multiprocessor computers

Algorithmic problems for multiprocessor computers

1. Implementation of a message-passing system.
2. Implementation of a virtual shared memory.
3. Load balancing.
4. Robustness against undetectable failures.

Multiprocessor computers

Algorithmic problems for multiprocessor computers

1. Implementation of a message-passing system.
2. Implementation of a virtual shared memory.
3. Load balancing.
4. Robustness against undetectable failures.

What other algorithmic problems for multiprocessor do you know?

Interacting Processes

The design of complicated software systems may often be simplified by organizing the software as a collection of (sequential) processes, each with a well-defined, simple task.

Interacting Processes

The design of complicated software systems may often be simplified by organizing the software as a collection of (sequential) processes, each with a well-defined, simple task.

A design as a collection of cooperating processes causes the application to be logically distributed, but it is quite possible to execute the processes on the same computer, in which case it is not physically distributed.

The operating system must control the concurrent execution of the processes and provide the means for communication and synchronization between the processes.

Interacting Processes

Algorithmic problems for interacting processes

1. Atomicity of memory operations.

Interacting Processes

Algorithmic problems for interacting processes

1. Atomicity of memory operations.
2. Synchronization of processes.

Interacting Processes

Algorithmic problems for interacting processes

1. Atomicity of memory operations.
2. Synchronization of processes.
3. Garbage collection.

Interacting Processes

Algorithmic problems for interacting processes

1. Atomicity of memory operations.
2. Synchronization of processes.
3. Garbage collection.
4. Data exchange.

Interacting Processes

Algorithmic problems for interacting processes

1. Atomicity of memory operations.
2. Synchronization of processes.
3. Garbage collection.
4. Data exchange.

What other algorithmic problems for interacting processes do you know?

Network Architecture

Networks are always organized as a collection of modules, each performing a very specific function and relying on services offered by other modules.

Network architecture supports a strict **hierarchy** between these modules, because each module exclusively uses the services offered by the previous module.

The modules are called layers or levels

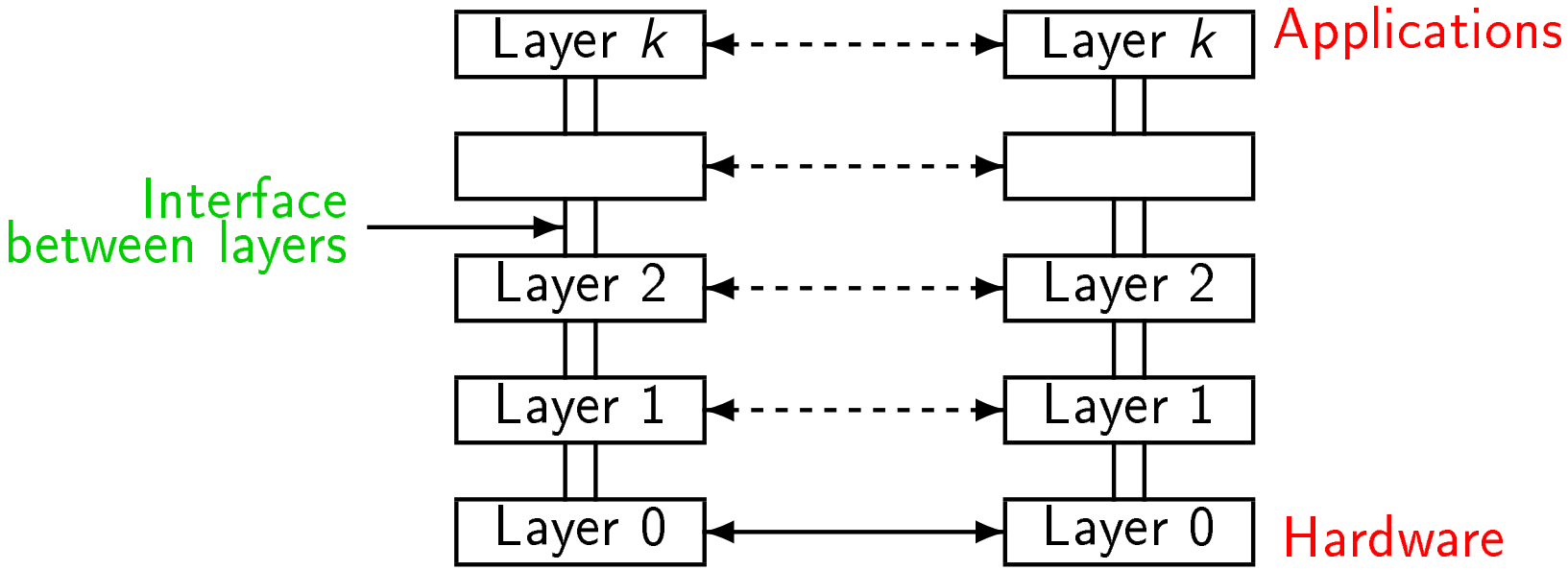


Рис.: A layered network architecture.

Network Architecture

Network architecture is the collection of layers and the accompanying definitions of all interfaces and protocols.

As a network may contain nodes produced by various manufacturers, it is important that products of different companies are compatible.
Therefore standard network architectures have been developed.

Some standards have received an "official" status:

- Reference Model **Open-Systems Interconnection** , **adopted by the International Standards Organization, ISO.**
- **OSI model for local networks: adopted by the Institute of Electrical and Electronic Engineers, IEEE.**
- **Family of TCP/IP protocols.**

Reference model OSI

The lowest layer 0 of the hierarchy is the hardware layer.
The 0/1 interface describes the procedures by which layer 1 can transmit raw information via the connecting wires, and a definition of the level itself specifies what types of wire are used, how many volts represents a one or a zero, etc.

Reference model OSI

The lowest layer 0 of the hierarchy is the hardware layer.
The 0/1 interface describes the procedures by which layer 1 can transmit raw information via the connecting wires, and a definition of the level itself specifies what types of wire are used, how many volts represents a one or a zero, etc.

The OSI reference model consists of seven layers:

Reference model OSI

The lowest layer 0 of the hierarchy is the hardware layer.
The 0/1 interface describes the procedures by which layer 1 can transmit raw information via the connecting wires, and a definition of the level itself specifies what types of wire are used, how many volts represents a one or a zero, etc.

The OSI reference model consists of seven layers:
Define all layers of the OSI reference model.

Reference model OSI

The lowest layer 0 of the hierarchy is the hardware layer.
The 0/1 interface describes the procedures by which layer 1 can transmit raw information via the connecting wires, and a definition of the level itself specifies what types of wire are used, how many volts represents a one or a zero, etc.

The OSI reference model consists of seven layers:

1. **physical layer**,
2. **data-link layer**,
3. **network layer**,
4. **transport layer**,
5. **session layer**,
6. **presentation layer**,
7. **application layer**.

Reference model OSI

1. Physical layer

Reference model OSI

1. Physical layer

is intended to transmit sequences of bits over a communication channel. The design of the layer itself is purely a matter for electrical engineers, while the 1/2 interface specifies the procedures by which the next layer calls the services of the physical layer. The service of the physical layer is not reliable; the bitstream may be scrambled during its transmission.

Reference model OSI

1. Physical layer

is intended to transmit sequences of bits over a communication channel. The design of the layer itself is purely a matter for electrical engineers, while the 1/2 interface specifies the procedures by which the next layer calls the services of the physical layer. The service of the physical layer is not reliable; the bitstream may be scrambled during its transmission.

2. Data-link layer

Reference model OSI

1. Physical layer

is intended to transmit sequences of bits over a communication channel. The design of the layer itself is purely a matter for electrical engineers, while the 1/2 interface specifies the procedures by which the next layer calls the services of the physical layer. The service of the physical layer is not reliable; the bitstream may be scrambled during its transmission.

2. Data-link layer

is intended to mask the unreliability of the physical layer. To this end the layer divides the bitstream into frames . The receiver of a frame can check whether it was received correctly and then inform the sender about correctly or incorrectly received, or lost frames.

Reference model OSI

3. Network layer

Reference model OSI

3. Network layer

is used to provide a communication between all pairs of nodes by selecting the routes through the network and to control the traffic load of each node and channel. The network layer contains algorithms to compute and update the routing tables. The service offered by the network layer is not reliable: messages (packets) sent from one node to the other may be lost, or duplicated, or shuffled. The layer must guarantee a bounded packet lifetime.

Reference model OSI

3. Network layer

is used to provide a communication between all pairs of nodes by selecting the routes through the network and to control the traffic load of each node and channel. The network layer contains algorithms to compute and update the routing tables. The service offered by the network layer is not reliable: messages (packets) sent from one node to the other may be lost, or duplicated, or shuffled. The layer must guarantee a bounded packet lifetime.

4. Transport layer

Reference model OSI

3. Network layer

is used to provide a communication between all pairs of nodes by selecting the routes through the network and to control the traffic load of each node and channel. The network layer contains algorithms to compute and update the routing tables. The service offered by the network layer is not reliable: messages (packets) sent from one node to the other may be lost, or duplicated, or shuffled. The layer must guarantee a bounded packet lifetime.

4. Transport layer

is intended to mask the unreliability of the network layer, i.e., to provide reliable **end-to-end** communication between any two nodes. The algorithms used for transmission control in the transport layer use similar techniques to the algorithms in the data-link layer: sequence numbers, feedback via acknowledgements, and retransmissions.

Reference model OSI

5. Session layer

Reference model OSI

5. Session layer

is aimed at providing facilities for maintaining connections between processes at different nodes: to open and close connections, to exchange data, to recover a communication if a node crashes during a session, to avoid collisions if critical operations are to be performed at both ends simultaneously.

Reference model OSI

5. Session layer

is aimed at providing facilities for maintaining connections between processes at different nodes: to open and close connections, to exchange data, to recover a communication if a node crashes during a session, to avoid collisions if critical operations are to be performed at both ends simultaneously.

6. Presentation layer

Reference model OSI

5. Session layer

is aimed at providing facilities for maintaining connections between processes at different nodes: to open and close connections, to exchange data, to recover a communication if a node crashes during a session, to avoid collisions if critical operations are to be performed at both ends simultaneously.

6. Presentation layer

is used to perform data conversion when the representation of information in one node differs from the representation in another node, to perform data compression and decompression , encryption , decryption , and integrity checking .

Reference model OSI

5. Session layer

is aimed at providing facilities for maintaining connections between processes at different nodes: to open and close connections, to exchange data, to recover a communication if a node crashes during a session, to avoid collisions if critical operations are to be performed at both ends simultaneously.

6. Presentation layer

is used to perform data conversion when the representation of information in one node differs from the representation in another node, to perform data compression and decompression , encryption , decryption , and integrity checking .

7. Application layer

Reference model OSI

5. Session layer

is aimed at providing facilities for maintaining connections between processes at different nodes: to open and close connections, to exchange data, to recover a communication if a node crashes during a session, to avoid collisions if critical operations are to be performed at both ends simultaneously.

6. Presentation layer

is used to perform data conversion when the representation of information in one node differs from the representation in another node, to perform data compression and decompression , encryption , decryption , and integrity checking .

7. Application layer

is used for applications.

OSI model for local networks

If the network organization relies on a common bus shared by all nodes then

- the network layer is almost empty,
- the design of the transport layer is very much simplified by the limited amount of non-determinism introduced by the bus,
- but the data-link layer is complicated by the fact that the same physical medium is accessed by a potentially large number of nodes.

OSI model for local networks

If the network organization relies on a common bus shared by all nodes then

- the network layer is almost empty,
- the design of the transport layer is very much simplified by the limited amount of non-determinism introduced by the bus,
- but the data-link layer is complicated by the fact that the same physical medium is accessed by a potentially large number of nodes.

OSI model for local networks has 3 main layers:

1. **physical layer**,
2. **medium access control layer**,
3. **logical link control layer**.

OSI model for local networks

2.1. Medium access control layer

is intended to resolve conflicts that arise between nodes that want to use the shared communication medium.

OSI model for local networks

2.1. Medium access control layer

is intended to resolve conflicts that arise between nodes that want to use the shared communication medium.

2.2. Logical link control layer

is comparable to the data-link layer in the OSI model: it is used to control the exchange of data between nodes using techniques similar to those used in the OSI protocols, namely sequence numbers and acknowledgements.

Language Support

A language for programming distributed applications must provide the means to express

Parallelism

is usually expressed by defining several processes , each being a sequential entity with its own state space. A language may either offer the possibility of statically defining a collection of processes or allow the dynamic creation and termination of processes.

Language Support

A language for programming distributed applications must provide the means to express

Parallelism

is usually expressed by defining several processes , each being a sequential entity with its own state space. A language may either offer the possibility of statically defining a collection of processes or allow the dynamic creation and termination of processes.

Communication

is achieved either by message passing (which provides both communication and synchronization) and/or by shared memory (which provides communication only).

Language Support

A language for programming distributed applications must provide the means to express

Parallelism

is usually expressed by defining several processes , each being a sequential entity with its own state space. A language may either offer the possibility of statically defining a collection of processes or allow the dynamic creation and termination of processes.

Communication

is achieved either by message passing (which provides both communication and synchronization) and/or by shared memory (which provides communication only).

Non-determinism

is based on guarded commands . A guarded command is a list of statements, each preceded by a boolean expression (its guard). The process may continue its execution with any of the statements for which the corresponding guard evaluates to true.

Distinguished features of distributed algorithms

1. Lack of knowledge of global state.

Nodes in a distributed system have access only to their own states and not to the global state of the entire system. Consequently, it is not possible to make a control decision based upon the global state.

The state of the communication subsystem is never directly observed by the nodes. This information can only be deduced indirectly by comparing information about messages sent and received by the nodes.

Distinguished features of distributed algorithms

1. Lack of knowledge of global state.

Nodes in a distributed system have access only to their own states and not to the global state of the entire system. Consequently, it is not possible to make a control decision based upon the global state.

The state of the communication subsystem is never directly observed by the nodes. This information can only be deduced indirectly by comparing information about messages sent and received by the nodes.

2. Lack of a global time-frame.

The temporal order relation induced on the events constituting the execution of a distributed algorithm is not total; for some pairs of events there may be a reason for deciding that one occurs before the other, but for other pairs it is the case that neither of the events occurs before the other

Generic features of distributed algorithms

3. Non-determinism.

the execution of a distributed system is usually non-deterministic, due to possible differences in execution speed of the system components.

Generic features of distributed algorithms

3. Non-determinism.

the execution of a distributed system is usually non-deterministic, due to possible differences in execution speed of the system components.

Lack of knowledge of the global state, lack of a global time-frame, and non-determinism makes the design of distributed algorithms an intricate craft, because the three aspects interfere in several ways.

An Example: Single-message Communication

Reliable information exchange via non-reliable communication channel.

Two processes *a* (producer) and *b* (consumer) are connected by a data network, which transmits messages from one process to the other. A message may be passed in the network an arbitrarily long time after it is sent, but may also be lost altogether in the network.

An Example: Single-message Communication

Reliable information exchange via non-reliable communication channel.

Two processes a (producer) and b (consumer) are connected by a data network, which transmits messages from one process to the other. A message may be passed in the network an arbitrarily long time after it is sent, but may also be lost altogether in the network.

The reliability of the communication is increased by the use of network control procedures (NCPs), via which a and b access the network. The process a initiates the communication by giving an information unit m to the procedure A .

An Example: Single-message Communication

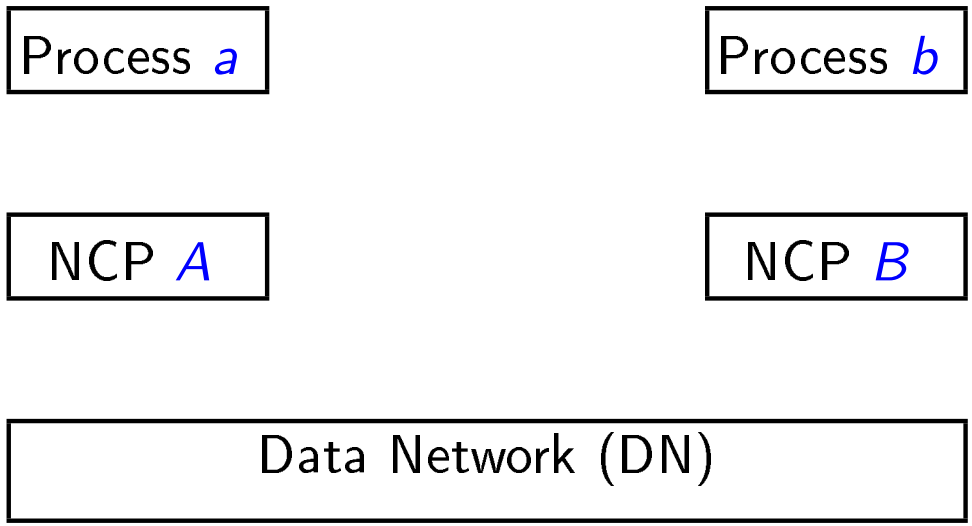
Reliable information exchange via non-reliable communication channel.

Two processes a (producer) and b (consumer) are connected by a data network, which transmits messages from one process to the other. A message may be passed in the network an arbitrarily long time after it is sent, but may also be lost altogether in the network.

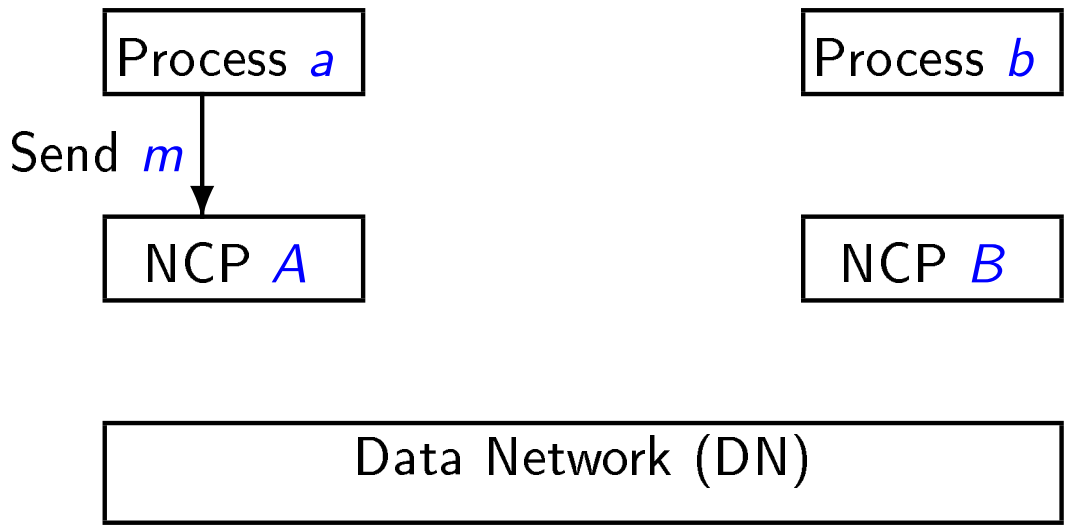
The reliability of the communication is increased by the use of network control procedures (NCPs), via which a and b access the network. The process a initiates the communication by giving an information unit m to the procedure A .

The interaction between the NCPs (via the data network, DN) must ensure that the information m is delivered to the process b (by NCP B) after which a is notified of the delivery (by NCP A).

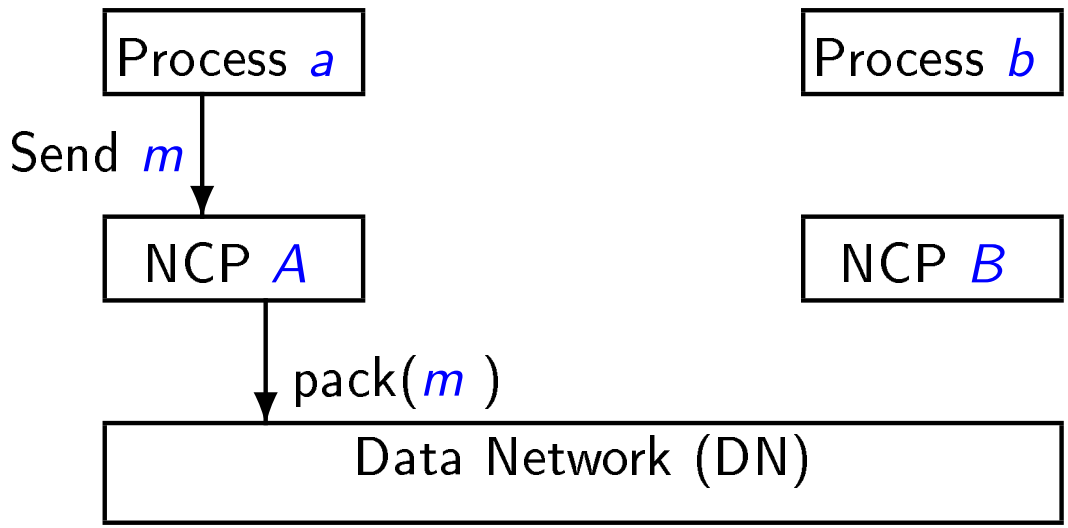
Reliable data exchange via non-reliable channel



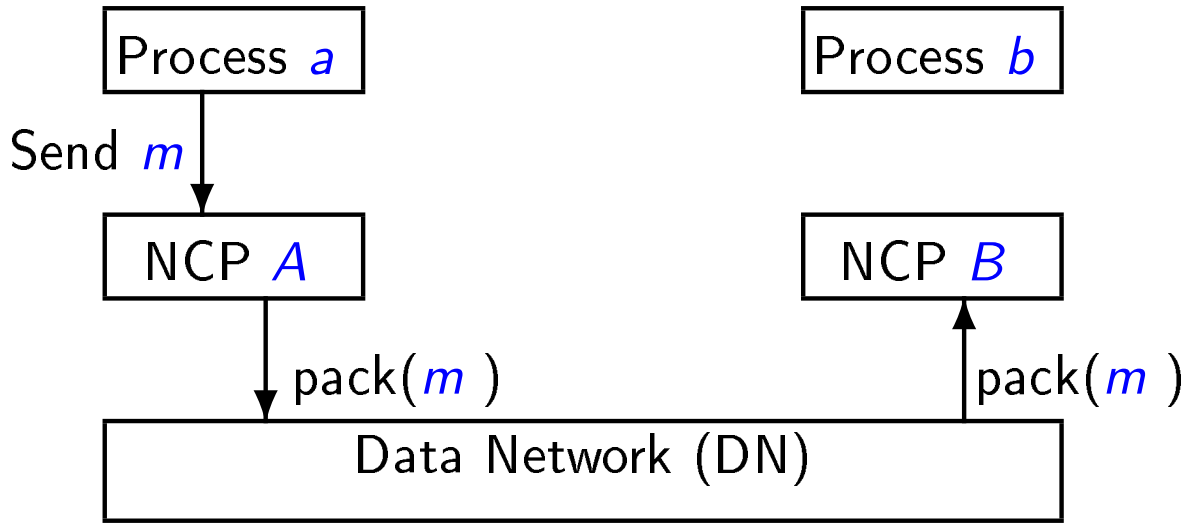
Reliable data exchange via non-reliable channel



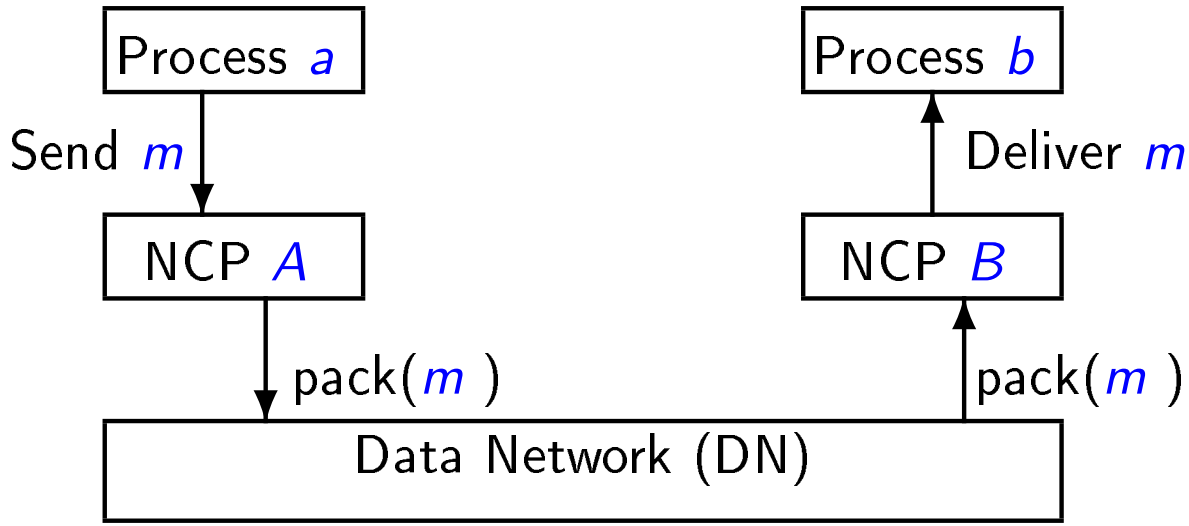
Reliable data exchange via non-reliable channel



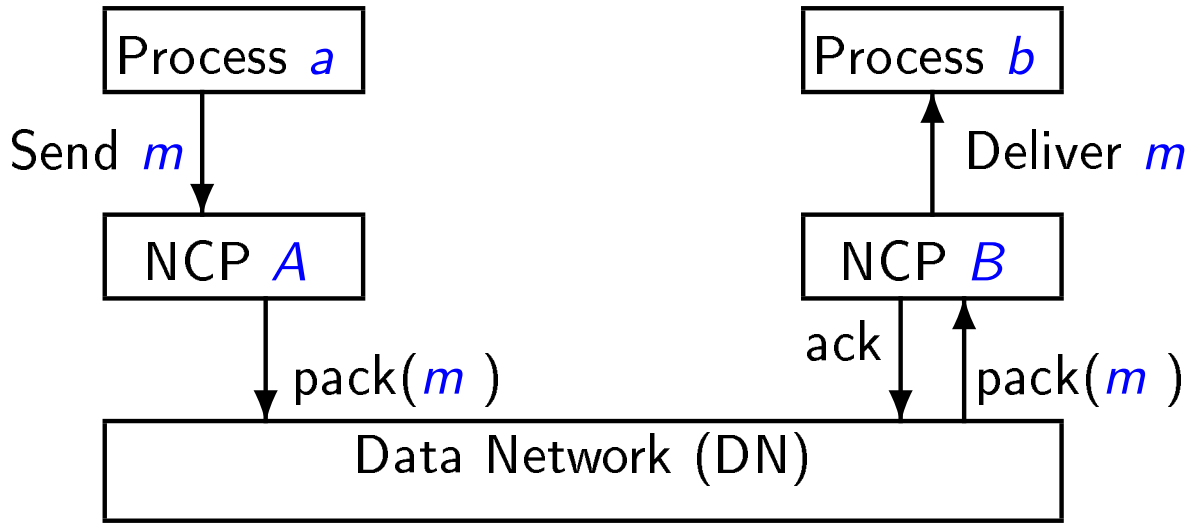
Reliable data exchange via non-reliable channel



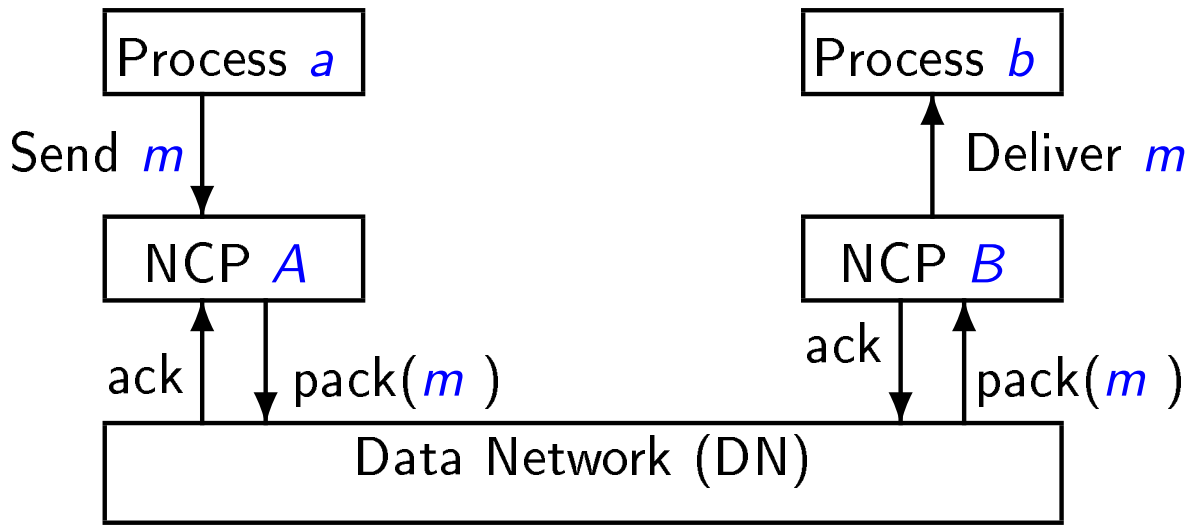
Reliable data exchange via non-reliable channel



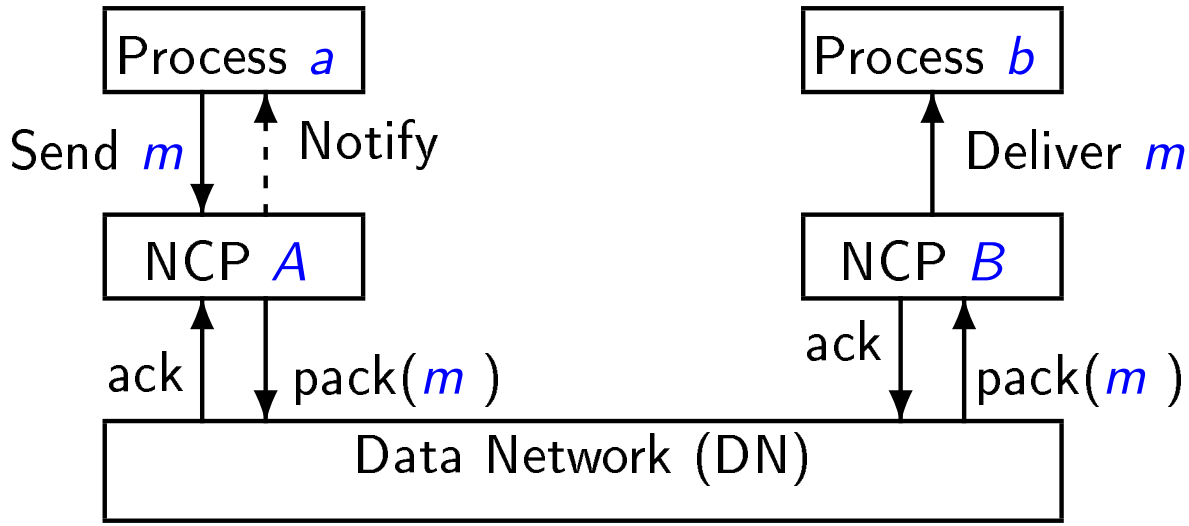
Reliable data exchange via non-reliable channel



Reliable data exchange via non-reliable channel



Reliable data exchange via non-reliable channel



Reliable data exchange via non-reliable channel

The network's unreliability forces NCPs A and B to engage in a conversation consisting of several messages.

Reliable data exchange via non-reliable channel

The network's unreliability forces NCPs A and B to engage in a conversation consisting of several messages.

Conversation Rules.

- NCP maintain state information about this conversation, but this information is discarded after the message exchange is complete.

Reliable data exchange via non-reliable channel

The network's unreliability forces NCPs A and B to engage in a conversation consisting of several messages.

Conversation Rules.

- NCP maintain state information about this conversation, but this information is discarded after the message exchange is complete.
- The initialization of status information is called opening .

Reliable data exchange via non-reliable channel

The network's unreliability forces NCPs A and B to engage in a conversation consisting of several messages.

Conversation Rules.

- NCP maintain state information about this conversation, but this information is discarded after the message exchange is complete.
- The initialization of status information is called opening .
- The discarding of status information is called closing ; after closing the conversation, an NCP is in exactly the same state as before opening it; this is called the closed state.

Reliable data exchange via non-reliable channel

The network's unreliability forces NCPs A and B to engage in a conversation consisting of several messages.

Conversation Rules.

- NCP maintain state information about this conversation, but this information is discarded after the message exchange is complete.
- The initialization of status information is called opening .
- The discarding of status information is called closing ; after closing the conversation, an NCP is in exactly the same state as before opening it; this is called the closed state.
- Information unit m is said to be lost if a was notified of its receipt by b , but the unit was never actually delivered to b .

Reliable data exchange via non-reliable channel

The network's unreliability forces NCPs A and B to engage in a conversation consisting of several messages.

Conversation Rules.

- NCP maintain state information about this conversation, but this information is discarded after the message exchange is complete.
- The initialization of status information is called opening .
- The discarding of status information is called closing ; after closing the conversation, an NCP is in exactly the same state as before opening it; this is called the closed state.
- Information unit m is said to be lost if a was notified of its receipt by b , but the unit was never actually delivered to b .
- Unit m is said to be duplicated if it was delivered twice.

Reliable data exchange via non-reliable channel

Reliable data exchange is impossible.

Assume that after initialization of a communication by a , NCPs A and B start a conversation, during which NCP B is supposed to deliver m to b after the receipt of a message M from NCP A .

Reliable data exchange via non-reliable channel

Reliable data exchange is impossible.

Assume that after initialization of a communication by a , NCPs A and B start a conversation, during which NCP B is supposed to deliver m to b after the receipt of a message M from NCP A . Consider the case where NCP B crashes and is restarted in the closed state after NCP A has sent the message M .

Reliable data exchange via non-reliable channel

Reliable data exchange is impossible.

Assume that after initialization of a communication by a , NCPs A and B start a conversation, during which NCP B is supposed to deliver m to b after the receipt of a message M from NCP A .
Consider the case where NCP B crashes and is restarted in the closed state after NCP A has sent the message M .
In this situation, neither NCP A , nor NCP B can tell whether m was already delivered when NCP B crashed; NCP A because it cannot observe the events in NCP B (lack of knowledge of the global state) and NCP B because it has crashed and was restarted in the closed state.

Reliable data exchange via non-reliable channel

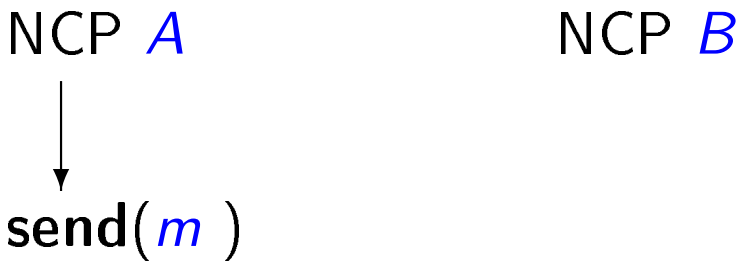
Reliable data exchange is impossible.

Assume that after initialization of a communication by a , NCPs A and B start a conversation, during which NCP B is supposed to deliver m to b after the receipt of a message M from NCP A .
Consider the case where NCP B crashes and is restarted in the closed state after NCP A has sent the message M .
In this situation, neither NCP A , nor NCP B can tell whether m was already delivered when NCP B crashed; NCP A because it cannot observe the events in NCP B (lack of knowledge of the global state) and NCP B because it has crashed and was restarted in the closed state.
Regardless of how the NCPs continue their conversation, an error may be introduced.
If NCP A sends the message to NCP B again and NCP B delivers the message, a duplication may arise.
If notification to a is given without delivery, a loss may arise. □

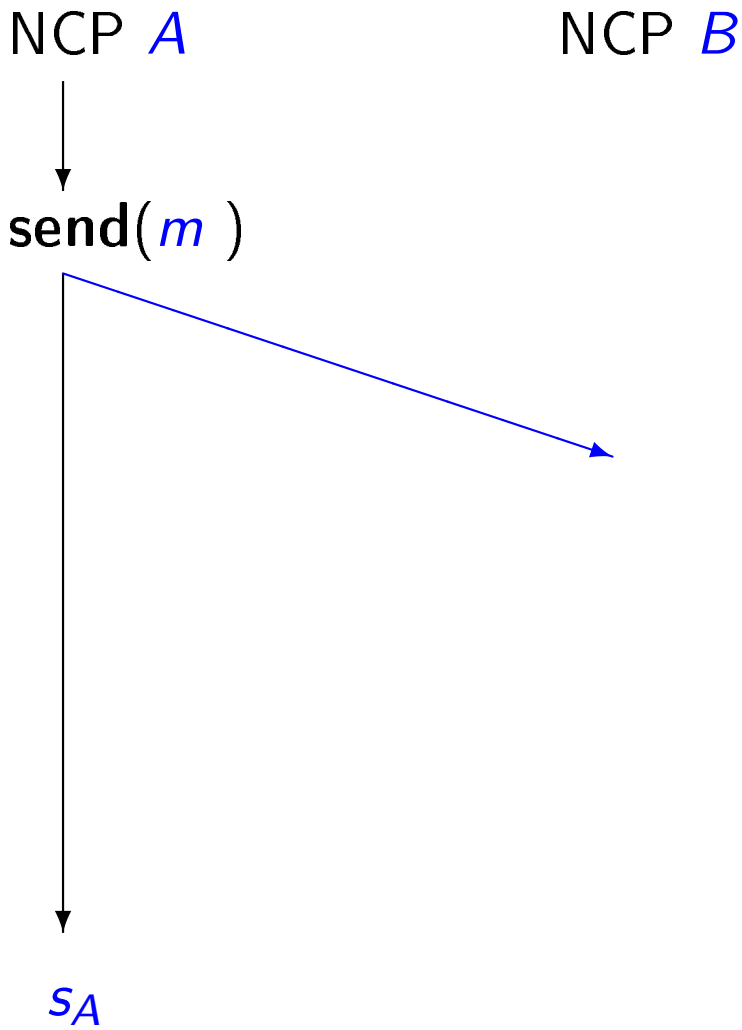
Reliable data exchange via non-reliable channel

NCP *A* NCP *B*

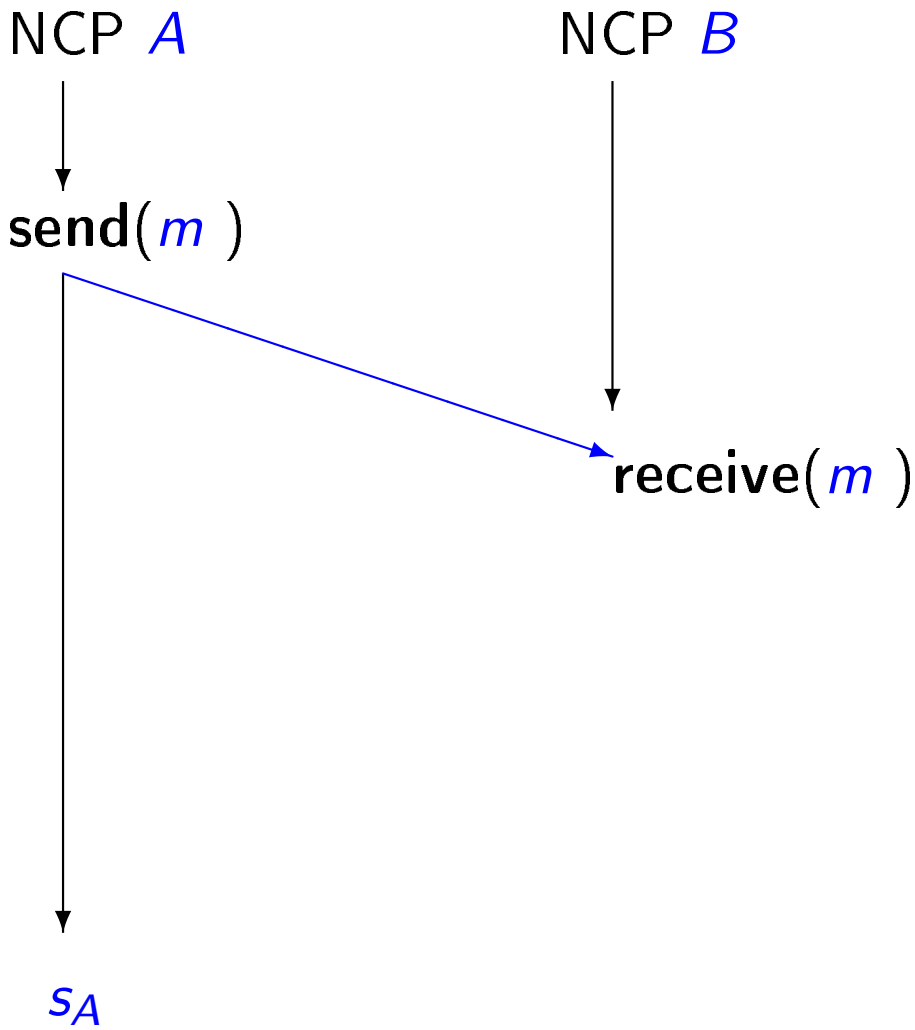
Reliable data exchange via non-reliable channel



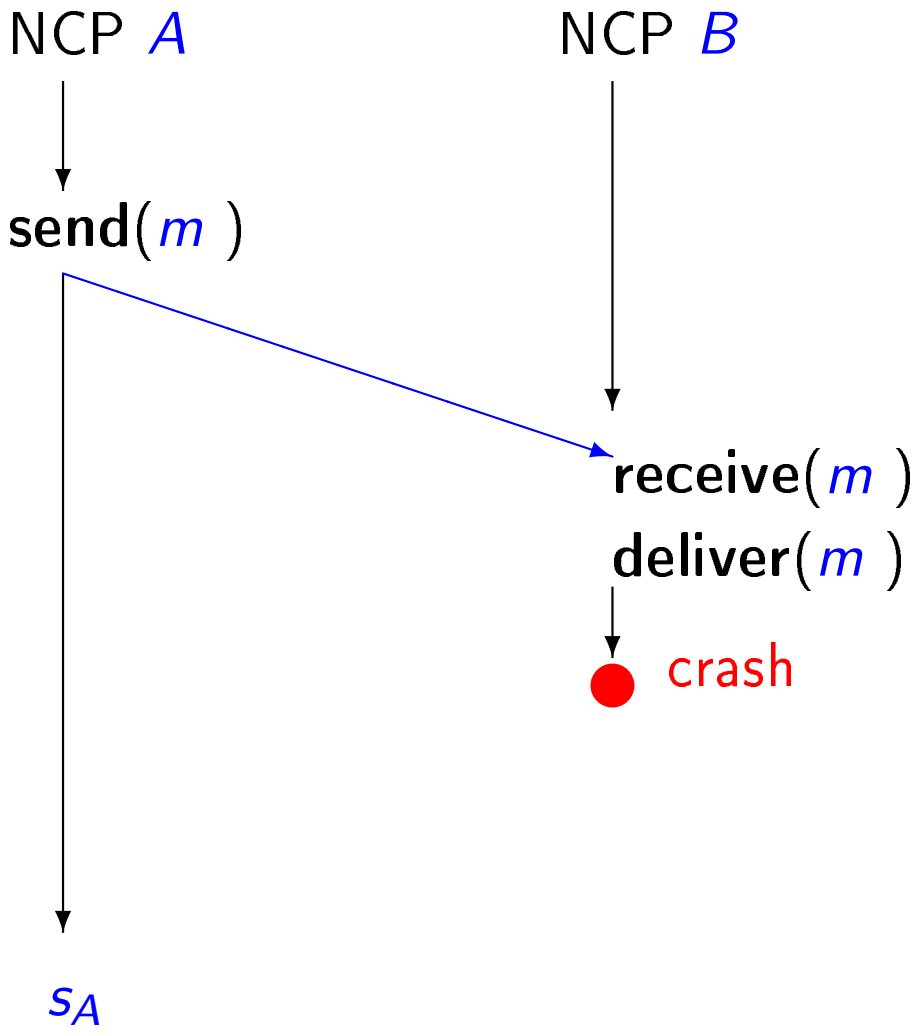
Reliable data exchange via non-reliable channel



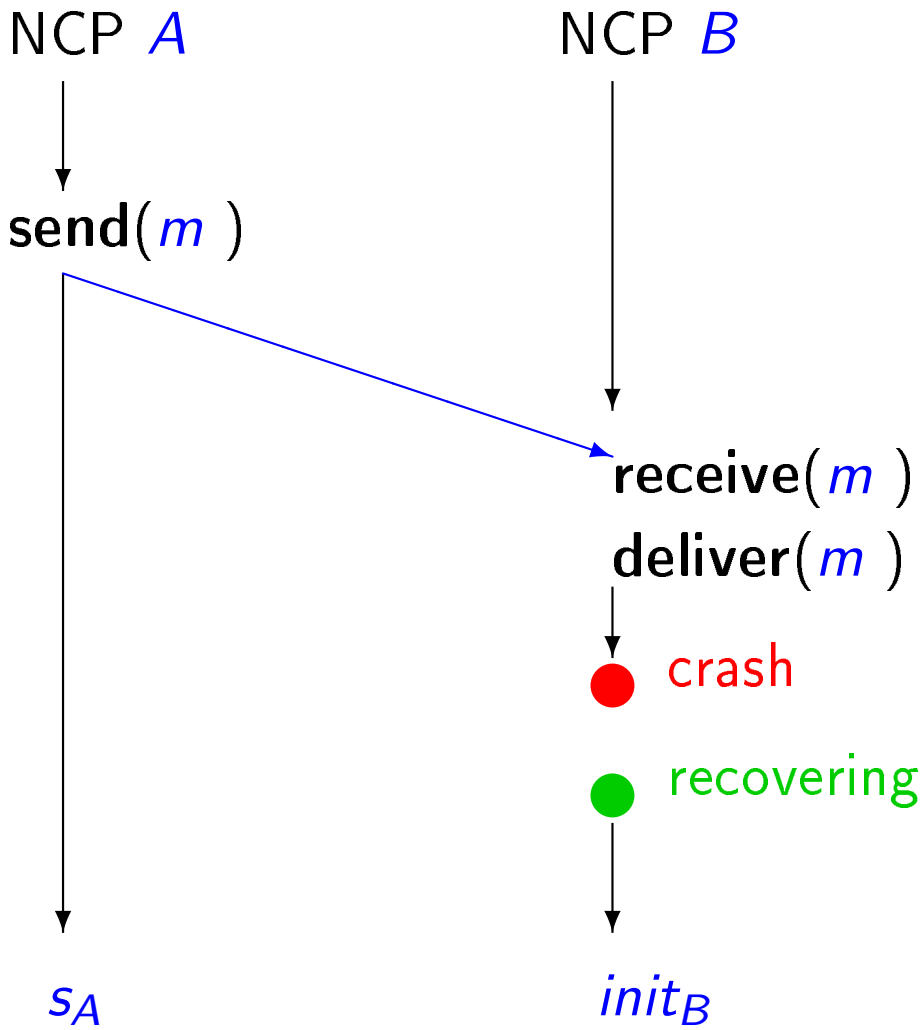
Reliable data exchange via non-reliable channel



Reliable data exchange via non-reliable channel



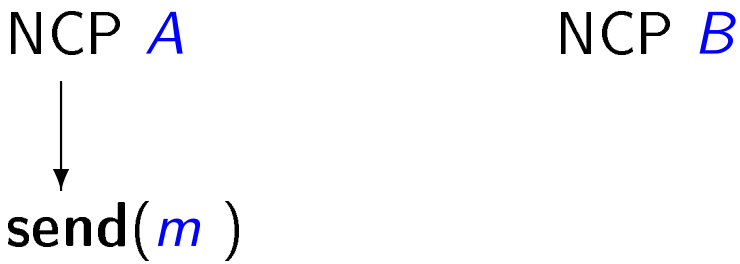
Reliable data exchange via non-reliable channel



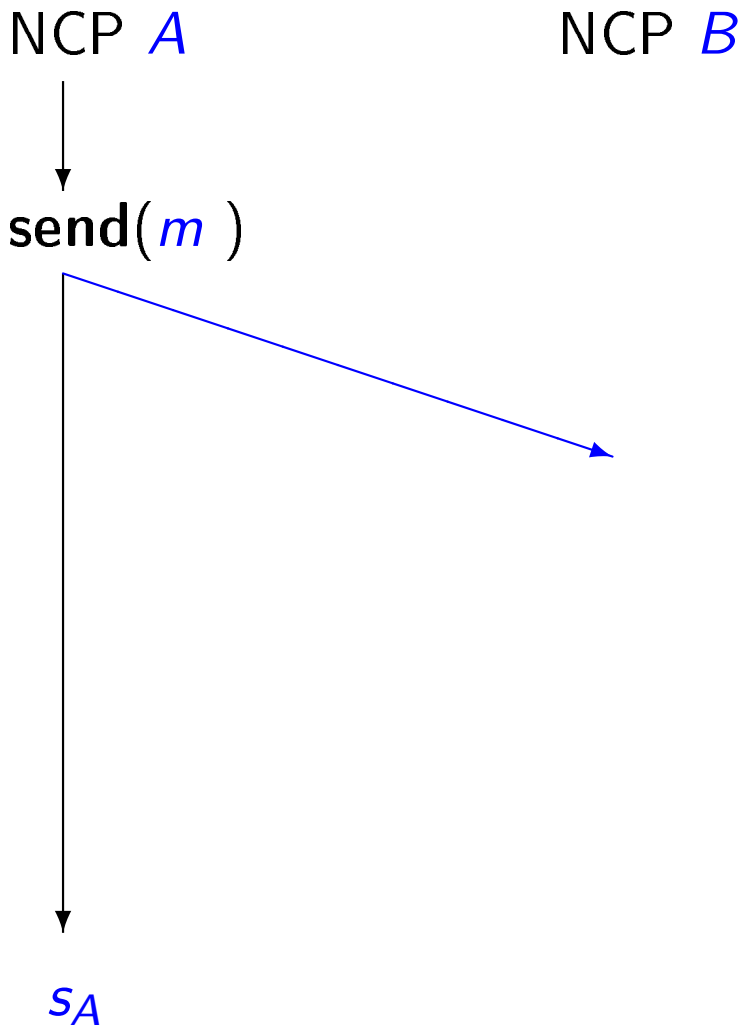
Reliable data exchange via non-reliable channel

NCP *A* NCP *B*

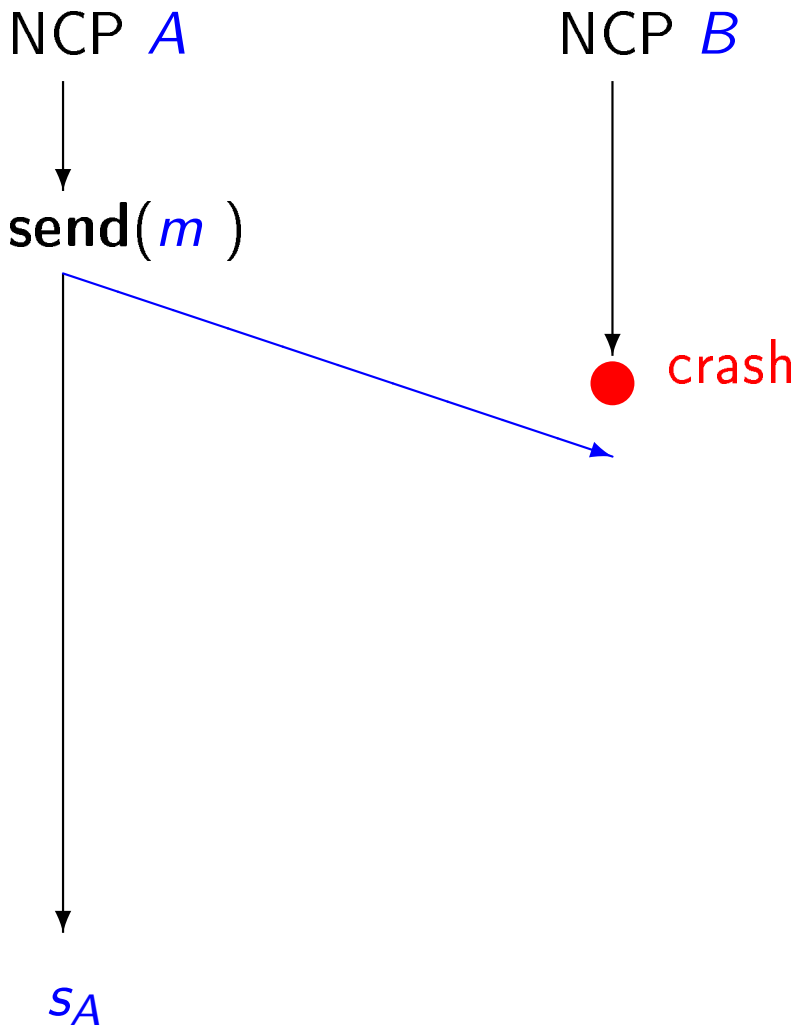
Reliable data exchange via non-reliable channel



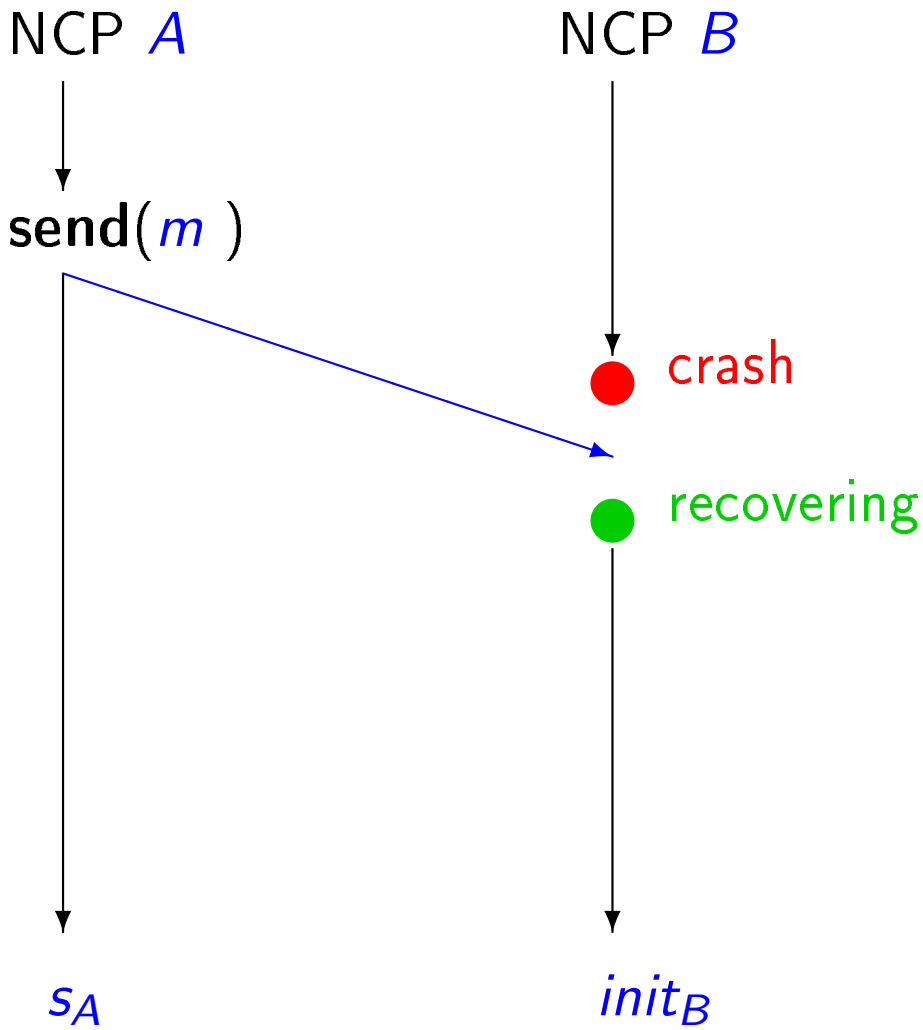
Reliable data exchange via non-reliable channel



Reliable data exchange via non-reliable channel



Reliable data exchange via non-reliable channel



Reliable data exchange via non-reliable channel

A one-message conversation.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* m, *close*

This protocol introduces a loss whenever the network refuses to deliver a message, but there is no possibility of introducing duplications.

Reliable data exchange via non-reliable channel

A two-message conversation.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* m, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify*, *close*

Reliable data exchange via non-reliable channel

A two-message conversation.

However, this protocol is not reliable either: message duplication is possible when the confirmation is delayed.

Reliable data exchange via non-reliable channel

A two-message conversation.

However, this protocol is not reliable either: message duplication is possible when the confirmation is delayed.

1. *NCP A :* $send\ \langle \mathbf{data}, m \rangle$

Reliable data exchange via non-reliable channel

A two-message conversation.

However, this protocol is not reliable either: message duplication is possible when the confirmation is delayed.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* m, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*

Reliable data exchange via non-reliable channel

A two-message conversation.

However, this protocol is not reliable either: message duplication is possible when the confirmation is delayed.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* m, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *DN* $\langle \textbf{\textit{ack}} \rangle$ *lost in the network*

Reliable data exchange via non-reliable channel

A two-message conversation.

However, this protocol is not reliable either: message duplication is possible when the confirmation is delayed.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* m, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *DN* $\langle \textbf{\textit{ack}} \rangle$ *lost in the network*
4. *NCP A* : *timeout, send* $\langle \textbf{\textit{data}}, m \rangle$

Reliable data exchange via non-reliable channel

A two-message conversation.

However, this protocol is not reliable either: message duplication is possible when the confirmation is delayed.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* m, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *DN* $\langle \textbf{\textit{ack}} \rangle$ *lost in the network*
4. *NCP A* : *timeout, send* $\langle \textbf{\textit{data}}, m \rangle$
5. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* m, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*

Reliable data exchange via non-reliable channel

A two-message conversation.

However, this protocol is not reliable either: message duplication is possible when the confirmation is delayed.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* m, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *DN* $\langle \textbf{\textit{ack}} \rangle$ *lost in the network*
4. *NCP A* : *timeout, send* $\langle \textbf{\textit{data}}, m \rangle$
5. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* m, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
6. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify, close*

Reliable data exchange via non-reliable channel

A two-message conversation.

In addition, when sending multiple messages, a message may be lost

Reliable data exchange via non-reliable channel

A two-message conversation.

In addition, when sending multiple messages, a message may be lost

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$

Reliable data exchange via non-reliable channel

A two-message conversation.

In addition, when sending multiple messages, a message may be lost

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*

Reliable data exchange via non-reliable channel

A two-message conversation.

In addition, when sending multiple messages, a message may be lost

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *NCP A* : *timeout, send* $\langle \textbf{\textit{data}}, m_1 \rangle$

Reliable data exchange via non-reliable channel

A two-message conversation.

In addition, when sending multiple messages, a message may be lost

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *NCP A* : *timeout, send* $\langle \textbf{\textit{data}}, m_1 \rangle$
4. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*

Reliable data exchange via non-reliable channel

A two-message conversation.

In addition, when sending multiple messages, a message may be lost

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *NCP A* : *timeout, send* $\langle \textbf{\textit{data}}, m_1 \rangle$
4. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
5. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$ *(шаг 1), notify, close*

Reliable data exchange via non-reliable channel

A two-message conversation.

In addition, when sending multiple messages, a message may be lost

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *NCP A* : *timeout, send* $\langle \textbf{\textit{data}}, m_1 \rangle$
4. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
5. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$ *(шаг 1), notify, close*
6. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_2 \rangle$

Reliable data exchange via non-reliable channel

A two-message conversation.

In addition, when sending multiple messages, a message may be lost

1. *NCP A* : *send* $\langle \textbf{data}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{data}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{ack} \rangle$, *close*
3. *NCP A* : *timeout, send* $\langle \textbf{data}, m_1 \rangle$
4. *NCP B* : *receive* $\langle \textbf{data}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{ack} \rangle$, *close*
5. *NCP A* : *receive* $\langle \textbf{ack} \rangle$ *(шаг 1), notify, close*
6. *NCP A* : *send* $\langle \textbf{data}, m_2 \rangle$
7. *DN* $\langle \textbf{data}, m_2 \rangle$ *потеряно в сети*

Reliable data exchange via non-reliable channel

A two-message conversation.

In addition, when sending multiple messages, a message may be lost

1. *NCP A* : *send* $\langle \textbf{data}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{data}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{ack} \rangle$, *close*
3. *NCP A* : *timeout, send* $\langle \textbf{data}, m_1 \rangle$
4. *NCP B* : *receive* $\langle \textbf{data}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{ack} \rangle$, *close*
5. *NCP A* : *receive* $\langle \textbf{ack} \rangle$ *(шаг 1), notify, close*
6. *NCP A* : *send* $\langle \textbf{data}, m_2 \rangle$
7. *DN* $\langle \textbf{data}, m_2 \rangle$ *потеряно в сети*
8. *NCP A* : *receive* $\langle \textbf{ack} \rangle$ *(шаг 2), notify, close*

Reliable data exchange via non-reliable channel

A two-message conversation.

In addition, when sending multiple messages, a message may be lost

1. *NCP A* : send $\langle \textbf{data}, m_1 \rangle$
2. *NCP B* : receive $\langle \textbf{data}, m_1 \rangle$, deliver m_1, send $\langle \textbf{ack} \rangle$, close
3. *NCP A* : timeout, send $\langle \textbf{data}, m_1 \rangle$
4. *NCP B* : receive $\langle \textbf{data}, m_1 \rangle$, deliver m_1, send $\langle \textbf{ack} \rangle$, close
5. *NCP A* : receive $\langle \textbf{ack} \rangle$ (шаг 1), notify, close
6. *NCP A* : send $\langle \textbf{data}, m_2 \rangle$
7. *DN* $\langle \textbf{data}, m_2 \rangle$ потеряно в сети
8. *NCP A* : receive $\langle \textbf{ack} \rangle$ (шаг 2), notify, close

The slow delivery is not detected due to the lack of a global time.

Reliable data exchange via non-reliable channel

A three-message conversation.

Reliable data exchange via non-reliable channel

A three-message conversation.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* m, *send* $\langle \textbf{\textit{ack}} \rangle$
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify*, *send* $\langle \textbf{\textit{close}} \rangle$, *close*
4. *NCP B* : *receive* $\langle \textbf{\textit{close}} \rangle$, *close*

Reliable data exchange via non-reliable channel

A three-message conversation.

1. *NCP A :* *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B :* *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* m, *send* $\langle \textbf{\textit{ack}} \rangle$
3. *NCP A :* *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify, send* $\langle \textbf{\textit{close}} \rangle$, *close*
4. *NCP B :* *receive* $\langle \textbf{\textit{close}} \rangle$, *close*

In this case, previous scenarios leading to errors are no longer possible.

Reliable data exchange via non-reliable channel

A three-message conversation.

However, this protocol is not reliable either: message loss is possible.

Reliable data exchange via non-reliable channel

A three-message conversation.

However, this protocol is not reliable either: message loss is possible.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$

Reliable data exchange via non-reliable channel

A three-message conversation.

However, this protocol is not reliable either: message loss is possible.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$

Reliable data exchange via non-reliable channel

A three-message conversation.

However, this protocol is not reliable either: message loss is possible.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify*, *send* $\langle \textbf{\textit{close}} \rangle$, *close*

Reliable data exchange via non-reliable channel

A three-message conversation.

However, this protocol is not reliable either: message loss is possible.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify, send* $\langle \textbf{\textit{close}} \rangle$, *close*
4. *DN* $\langle \textbf{\textit{close}} \rangle$ *lost in the network*

Reliable data exchange via non-reliable channel

A three-message conversation.

However, this protocol is not reliable either: message loss is possible.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify, send* $\langle \textbf{\textit{close}} \rangle$, *close*
4. *DN* $\langle \textbf{\textit{close}} \rangle$ *lost in the network*
5. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_2 \rangle$

Reliable data exchange via non-reliable channel

A three-message conversation.

However, this protocol is not reliable either: message loss is possible.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify, send* $\langle \textbf{\textit{close}} \rangle$, *close*
4. *DN* $\langle \textbf{\textit{close}} \rangle$ *lost in the network*
5. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_2 \rangle$
6. *DN* $\langle \textbf{\textit{data}}, m_2 \rangle$ *lost in the network*

Reliable data exchange via non-reliable channel

A three-message conversation.

However, this protocol is not reliable either: message loss is possible.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify, send* $\langle \textbf{\textit{close}} \rangle$, *close*
4. *DN* $\langle \textbf{\textit{close}} \rangle$ *lost in the network*
5. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_2 \rangle$
6. *DN* $\langle \textbf{\textit{data}}, m_2 \rangle$ *lost in the network*
7. *NCP B* : *timeout, retransmit* $\langle \textbf{\textit{ack}} \rangle$ *(for the step 2)*

Reliable data exchange via non-reliable channel

A three-message conversation.

However, this protocol is not reliable either: message loss is possible.

1.	*NCP A* :	*send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2.	*NCP B* :	*receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$
3.	*NCP A* :	*receive* $\langle \textbf{\textit{ack}} \rangle$, *notify, send* $\langle \textbf{\textit{close}} \rangle$, *close*
4.	*DN*	$\langle \textbf{\textit{close}} \rangle$ *lost in the network*
5.	*NCP A* :	*send* $\langle \textbf{\textit{data}}, m_2 \rangle$
6.	*DN*	$\langle \textbf{\textit{data}}, m_2 \rangle$ *lost in the network*
7.	*NCP B* :	*timeout, retransmit* $\langle \textbf{\textit{ack}} \rangle$ *(for the step 2)*
8.	*NCP A* :	*receive* $\langle \textbf{\textit{ack}} \rangle$, *notify, send* $\langle \textbf{\textit{close}} \rangle$, *close*

Reliable data exchange via non-reliable channel

A three-message conversation.

However, this protocol is not reliable either: message loss is possible.

1.	*NCP A* :	*send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2.	*NCP B* :	*receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* m_1, *send* $\langle \textbf{\textit{ack}} \rangle$
3.	*NCP A* :	*receive* $\langle \textbf{\textit{ack}} \rangle$, *notify, send* $\langle \textbf{\textit{close}} \rangle$, *close*
4.	*DN*	$\langle \textbf{\textit{close}} \rangle$ *lost in the network*
5.	*NCP A* :	*send* $\langle \textbf{\textit{data}}, m_2 \rangle$
6.	*DN*	$\langle \textbf{\textit{data}}, m_2 \rangle$ *lost in the network*
7.	*NCP B* :	*timeout, retransmit* $\langle \textbf{\textit{ack}} \rangle$ *(for the step 2)*
8.	*NCP A* :	*receive* $\langle \textbf{\textit{ack}} \rangle$, *notify, send* $\langle \textbf{\textit{close}} \rangle$, *close*
9.	*NCP B* :	*receive* $\langle \textbf{\textit{close}} \rangle$, *close*

Reliable data exchange via non-reliable channel

A three-message conversation (improved variant).

1. *NCP A* : send $\langle \textbf{data}, m, x \rangle$
2. *NCP B* : receive $\langle \textbf{data}, m, x \rangle$, send $\langle \textbf{ack}, x, y \rangle$
3. *NCP A* : receive $\langle \textbf{ack}, x, y \rangle$, notify, send $\langle \textbf{close}, x, y \rangle$, close
4. *NCP B* : receive $\langle \textbf{close}, x, y \rangle$, deliver m, close

Reliable data exchange via non-reliable channel

A three-message conversation (improved variant).

However, this protocol is not reliable either: message duplication is possible.

1. *NCP A* : $send\ \langle \mathbf{data}, m, x \rangle$

Reliable data exchange via non-reliable channel

A three-message conversation (improved variant).

However, this protocol is not reliable either: message duplication is possible.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m, x \rangle$
2. *NCP A* : *timeout, retransmit* $\langle \textbf{\textit{data}}, m, x \rangle$

Reliable data exchange via non-reliable channel

A three-message conversation (improved variant).

However, this protocol is not reliable either: message duplication is possible.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m, x \rangle$
2. *NCP A* : *timeout, retransmit* $\langle \textbf{\textit{data}}, m, x \rangle$
3. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m, x \rangle$ *(for the step 2), send* $\langle \textbf{\textit{ack}}, x, y$

Reliable data exchange via non-reliable channel

A three-message conversation (improved variant).

However, this protocol is not reliable either: message duplication is possible.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m, x \rangle$
2. *NCP A* : *timeout, retransmit* $\langle \textbf{\textit{data}}, m, x \rangle$
3. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m, x \rangle$ *(for the step 2), send* $\langle \textbf{\textit{ack}}, x, y$
4. *NCP A* : *receive* $\langle \textbf{\textit{ack}}, x, y_1 \rangle$*, notify, send* $\langle \textbf{\textit{close}}, x, y_1 \rangle$*, close*

Reliable data exchange via non-reliable channel

A three-message conversation (improved variant).

However, this protocol is not reliable either: message duplication is possible.

1. *NCP A* : *send* $\langle \textbf{data}, m, x \rangle$
2. *NCP A* : *timeout, retransmit* $\langle \textbf{data}, m, x \rangle$
3. *NCP B* : *receive* $\langle \textbf{data}, m, x \rangle$ *(for the step 2), send* $\langle \textbf{ack}, x, y$
4. *NCP A* : *receive* $\langle \textbf{ack}, x, y_1 \rangle$, *notify, send* $\langle \textbf{close}, x, y_1 \rangle$, *close*
5. *NCP B* : *receive* $\langle \textbf{close}, x, y_1 \rangle$, *deliver m, close*

Reliable data exchange via non-reliable channel

A three-message conversation (improved variant).

However, this protocol is not reliable either: message duplication is possible.

1. *NCP A* : send $\langle \textbf{data}, m, x \rangle$
2. *NCP A* : timeout, retransmit $\langle \textbf{data}, m, x \rangle$
3. *NCP B* : receive $\langle \textbf{data}, m, x \rangle$ (for the step 2), send $\langle \textbf{ack}, x, y$
4. *NCP A* : receive $\langle \textbf{ack}, x, y_1 \rangle$, notify, send $\langle \textbf{close}, x, y_1 \rangle$, close
5. *NCP B* : receive $\langle \textbf{close}, x, y_1 \rangle$, deliver m, close
6. *NCP B* : receive $\langle \textbf{data}, m, x \rangle$ (for the step 1), send $\langle \textbf{ack}, x, y$

Reliable data exchange via non-reliable channel

A three-message conversation (improved variant).

However, this protocol is not reliable either: message duplication is possible.

1. *NCP A* : send $\langle \textbf{data}, m, x \rangle$
2. *NCP A* : timeout, retransmit $\langle \textbf{data}, m, x \rangle$
3. *NCP B* : receive $\langle \textbf{data}, m, x \rangle$ (for the step 2), send $\langle \textbf{ack}, x, y$
4. *NCP A* : receive $\langle \textbf{ack}, x, y_1 \rangle$, notify, send $\langle \textbf{close}, x, y_1 \rangle$, close
5. *NCP B* : receive $\langle \textbf{close}, x, y_1 \rangle$, deliver m, close
6. *NCP B* : receive $\langle \textbf{data}, m, x \rangle$ (for the step 1), send $\langle \textbf{ack}, x, y$
7. *NCP A* : receive $\langle \textbf{ack}, x, y_2 \rangle$, reply $\langle \textbf{nocon}, x, y_2 \rangle$

Reliable data exchange via non-reliable channel

A three-message conversation (improved variant).

However, this protocol is not reliable either: message duplication is possible.

1. *NCP A* : send $\langle \mathbf{data}, m, x \rangle$
2. *NCP A* : timeout, retransmit $\langle \mathbf{data}, m, x \rangle$
3. *NCP B* : receive $\langle \mathbf{data}, m, x \rangle$ (for the step 2), send $\langle \mathbf{ack}, x, y$
4. *NCP A* : receive $\langle \mathbf{ack}, x, y_1 \rangle$, notify, send $\langle \mathbf{close}, x, y_1 \rangle$, close
5. *NCP B* : receive $\langle \mathbf{close}, x, y_1 \rangle$, deliver m, close
6. *NCP B* : receive $\langle \mathbf{data}, m, x \rangle$ (for the step 1), send $\langle \mathbf{ack}, x, y$
7. *NCP A* : receive $\langle \mathbf{ack}, x, y_2 \rangle$, reply $\langle \mathbf{nocon}, x, y_2 \rangle$
8. *NCP B* : receive $\langle \mathbf{nocon}, x, y_2 \rangle$ in response to $\langle \mathbf{ack}, x, y_2 \rangle$, deliver m, close

Reliable data exchange via non-reliable channel

A four-message conversation.

1. NCP A : send $\langle \mathbf{data}, m, x \rangle$
2. NCP B : receive $\langle \mathbf{data}, m, x \rangle$, send $\langle \mathbf{open}, m, x, y \rangle$
3. NCP A : receive $\langle \mathbf{open}, m, x, y \rangle$, send $\langle \mathbf{agree}, m, x, y \rangle$
4. NCP B : receive $\langle \mathbf{agree}, m, x, y \rangle$, deliver m, send $\langle \mathbf{ack}, x, y \rangle$, close
5. NCP A : receive $\langle \mathbf{ack}, x, y \rangle$, notify, close

Research Field

Task 1.

Show that in a 4-message protocol, duplication or loss of messages is possible due to the fact that NCP A has to reckon with the possibility of NCP B failure.
In this case, duplication or loss of messages occurs even if NCP B does not actually fail.

Research Field

Task 2.

Build a two-message protocol that never allows message loss (although it can duplicate messages).

Research Field

Task 2.

Build a two-message protocol that never allows message loss (although it can duplicate messages).

Hint: Use the message identification numbers that were introduced in the improved version of the three-message dialog.

Research Field

Task 2.

Build a two-message protocol that never allows message loss (although it can duplicate messages).

Hint: Use the message identification numbers that were introduced in the improved version of the three-message dialog.

PROVE that the constructed protocol does not really lose a single message.

Research Field

Task 3.

Build a five-message protocol that prevents information loss and allows duplication only when one of the NCP actually crashes.

END OF LECTURE 1.