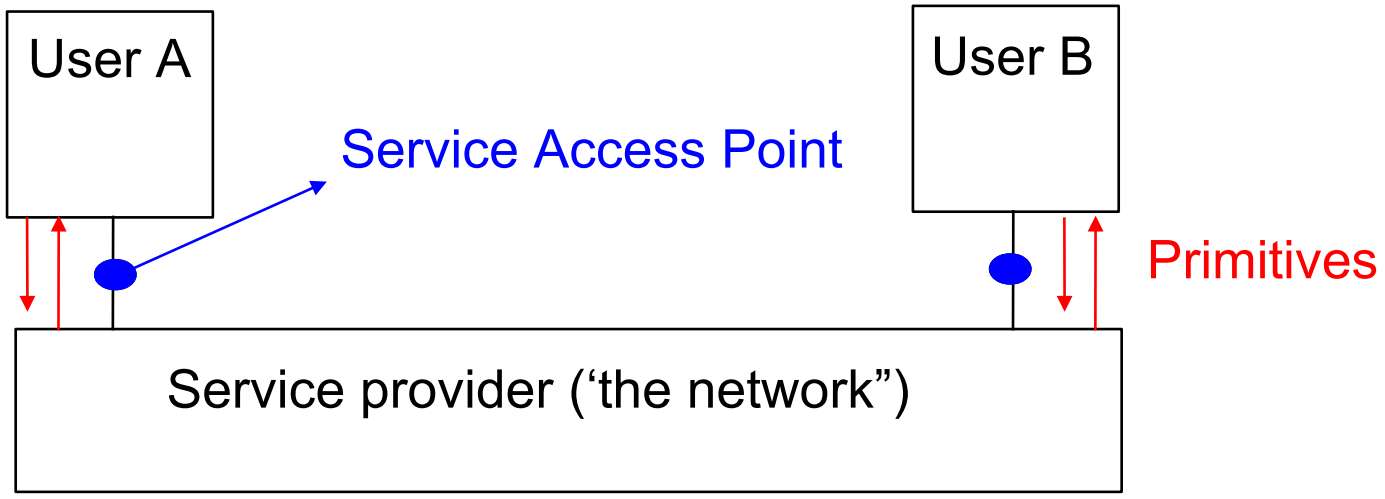
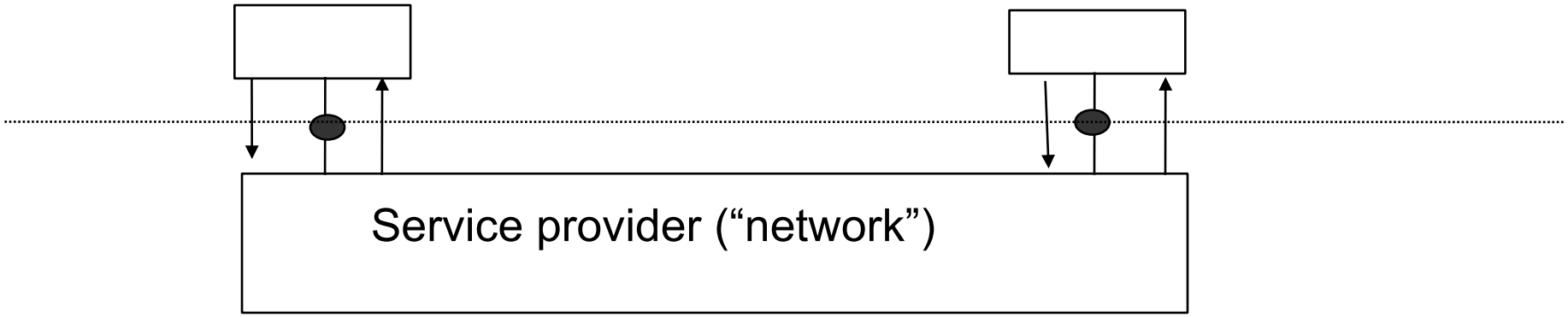
# Basic concepts

- Abstract model of the network behaviour
  - Network is considered as a black box
  - Users interact with the network by using primitives that are exchanged through a service access point (SAP)

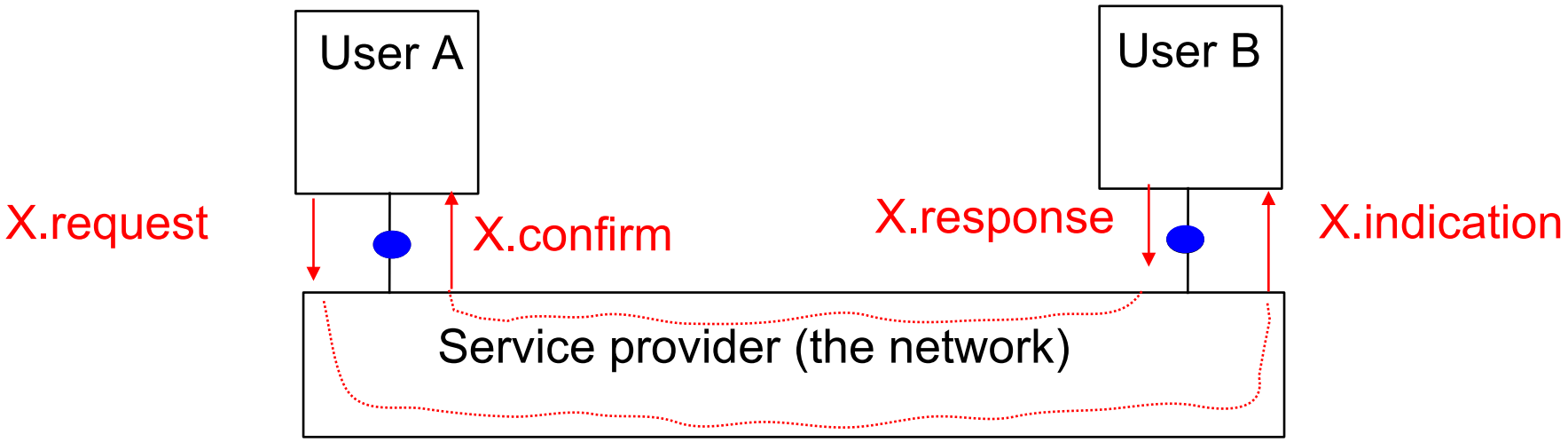


# Types of primitives




- Primitive
  - Abstract representation of the interaction between one user and its network provider
  - Can contain parameters such as :
    - source
    - destination
    - message (SDU or Service Data Unit)

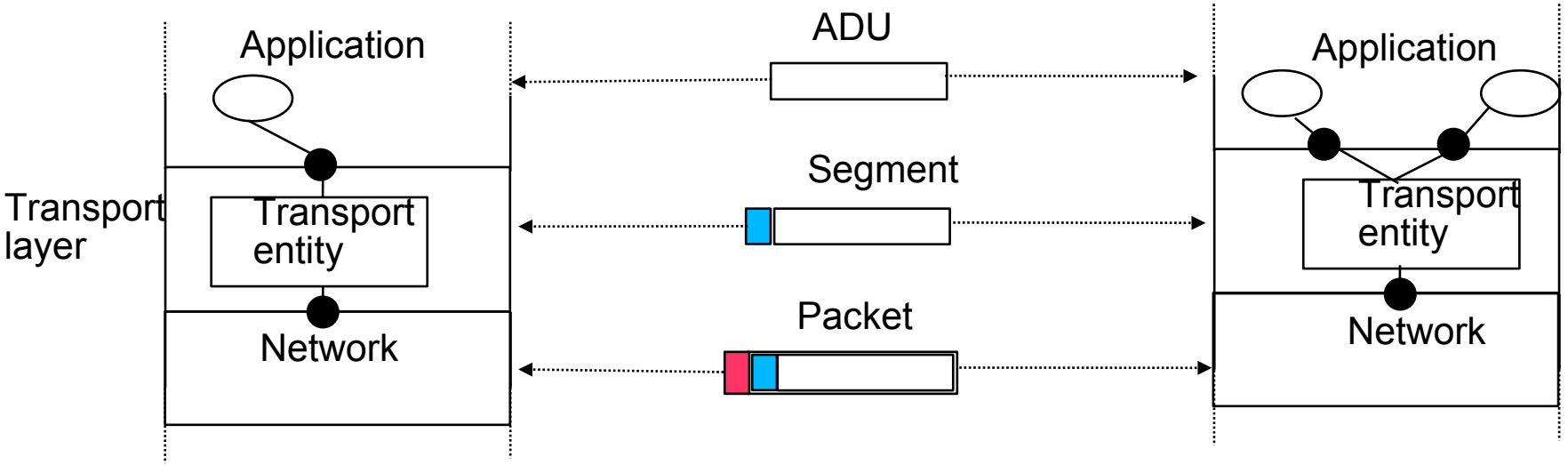
# Types of primitives (2)




- X.request
  - request from a user to a service provider
- X.indication
  - primitive generated by the network provider to a user (often related to an earlier and remote X.request primitive)
- X.response
  - primitive used to answer to an earlier X.indication primitive
- X.confirm
  - primitive generated by the network provider to a user (related to a remote X.response primitive)

# The transport layer (2)

- Internal organisation
  - The transport layer uses the service provide by the network layer
  - Two transport layer entities exchanges segments



# Contents

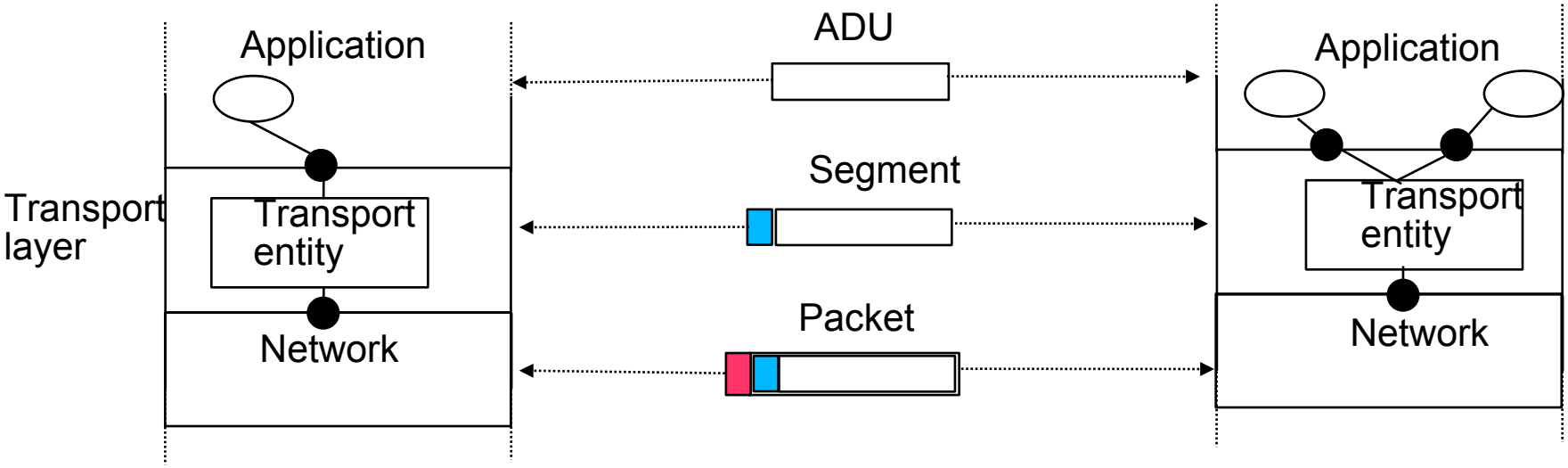
- A short history of networking
- Basics Reminder
- Case study : Building a reliable transport layer

# Transport layer protocols

- How can we provide a reliable service in the transport layer
  - Hypotheses : always start simple !
    - The application sends small SDUs
    - The network layer provides a perfect service
      - There are no transmission errors inside the packets
      - No packet is lost
      - There is no packet reordering
      - There are no duplications of packets
    - Data transmission is unidirectional

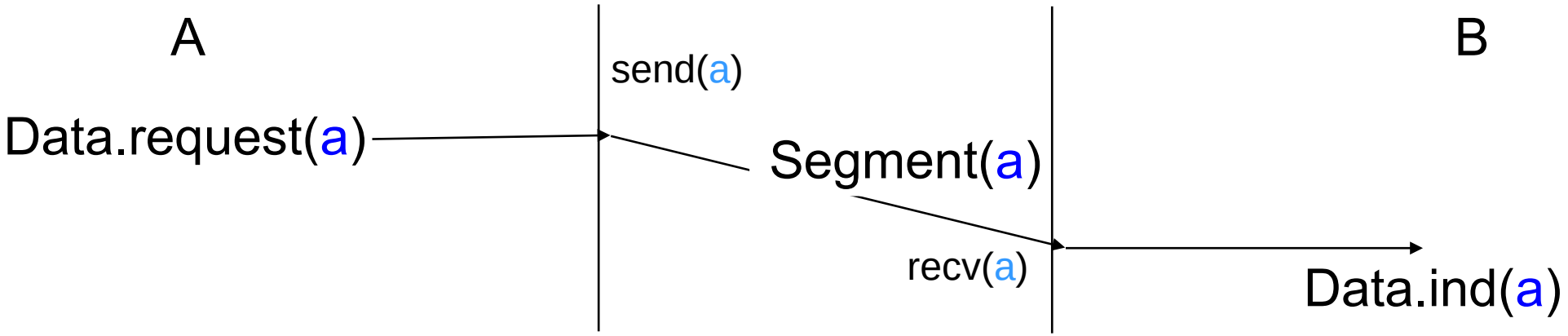
# Transport layer protocols (2)

- Reference environment




- Notations
  - `data.req` and `data.ind` primitives for application/transport interactions
  - `recv()` and `send()` for interactions between transport entity and network layer

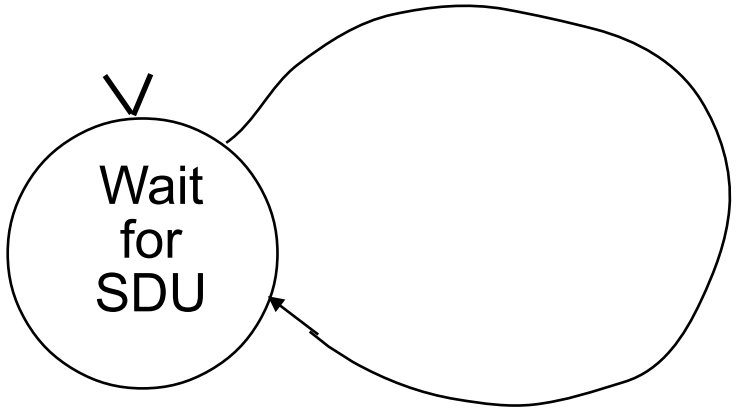
# Protocol 1 : Basics




- Principle
  - Upon reception of `data.request(SDU)`, the transport entity sends a segment containing this SDU through the network layer (`send(SDU)`)
  - Upon reception of the contents of one packet from the network layer (`recv(SDU)`), transport entity delivers the SDU found in the packet to its user by using `data.ind(SDU)`

# Protocol 1 as a FSM

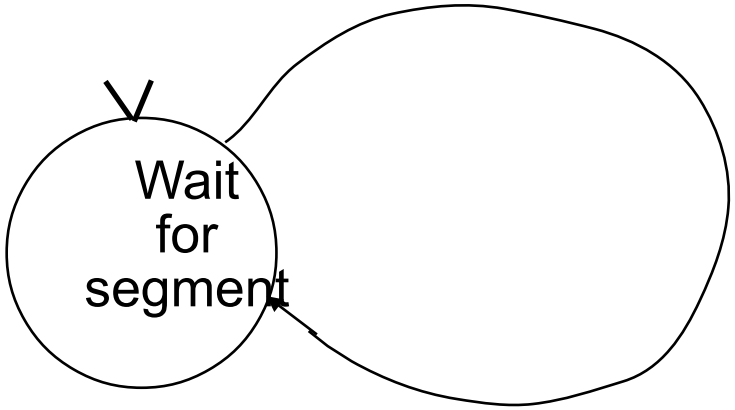
- Sender




Data.req(SDU)

send(SDU)

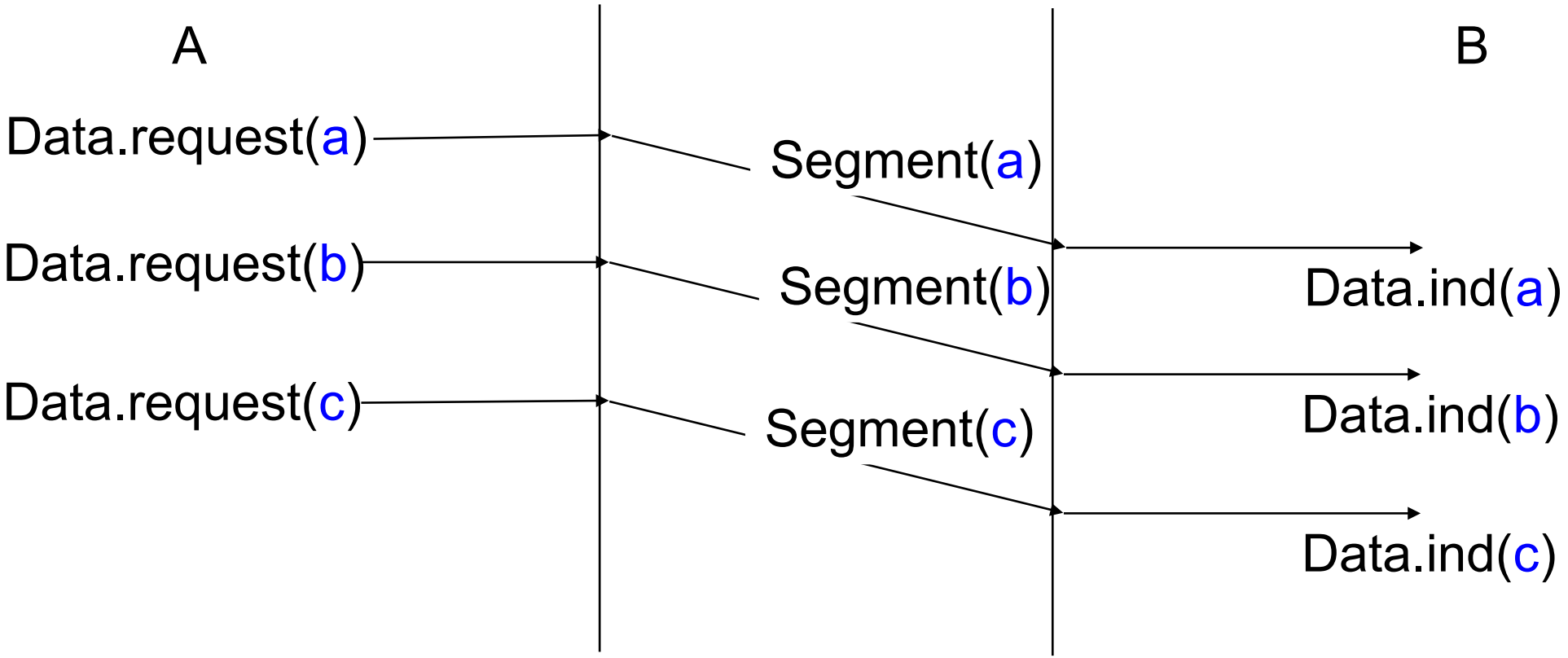
- Receiver




recvd(SDU)

Data.ind(SDU)

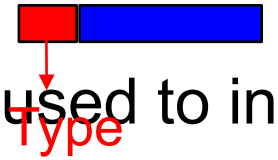
# Protocol 1 : Example




- Issue
  - What happens if the receiver is much slower than the sender ?
    - e.g. receiver can process one segment per second while sender is producing 10 segments per second ?

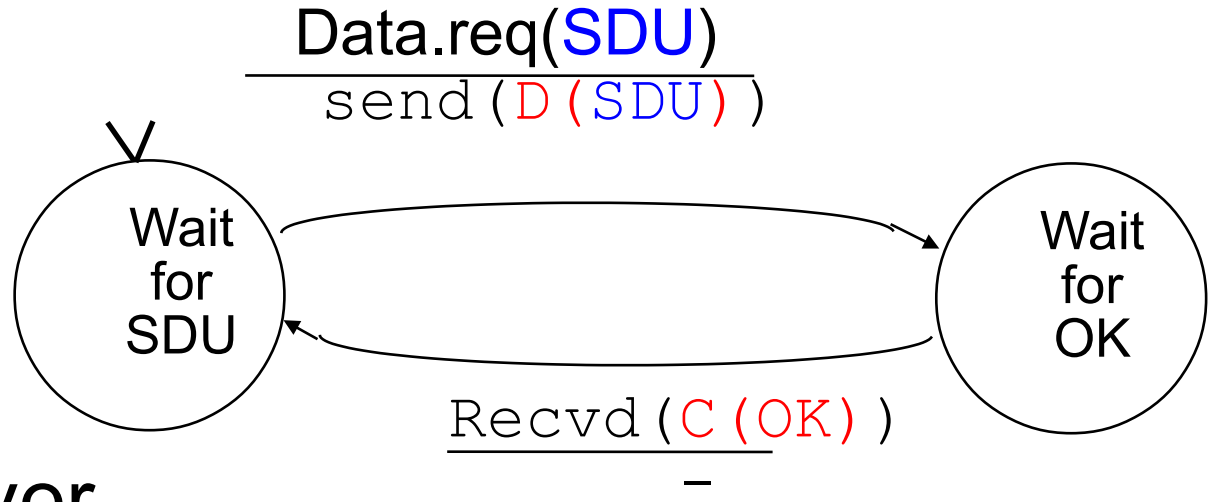
# Protocol 2

- Principle
  - Use a control segment (OK) that is sent by the receiver after having processed the received segment
  - creates a feedback loop between sender and receiver
- Consequences
  - Two types of segments
    - Data segment containing on SDU
      - Notation : D(SDU)
    - Control segment
      - Notation : C(OK)
  - Segment format
    - At least one bit in the segment header is used to indicate the type of segment

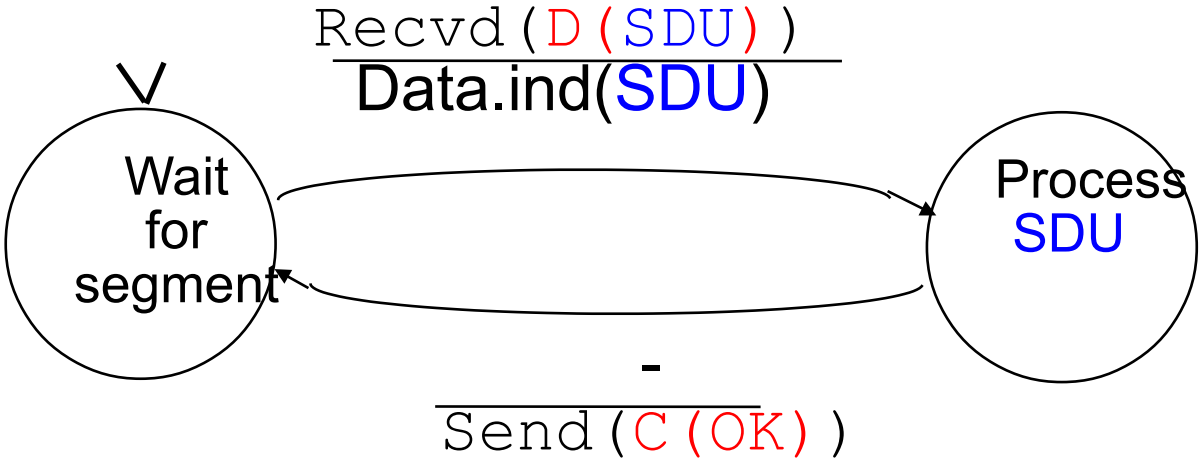


# Protocol 2 (cont.)

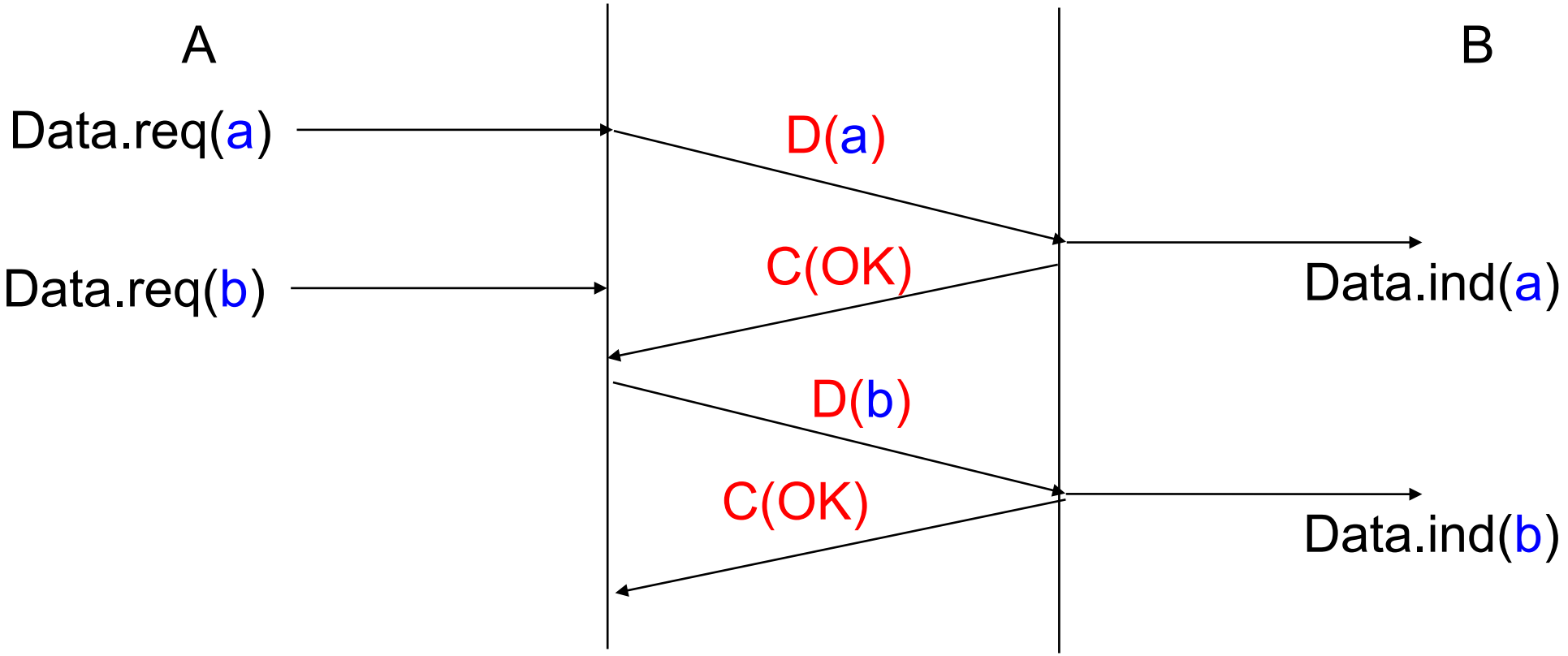
- Sender




- Receiver



# Protocol 2 : Example




Lock-step : - The sender is locked until it receives next OK from the receiver.
- Only one message in flight at any time.