ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Network Protocol Design and Evaluation

01 - Introduction

Stefan Rührup

University of Freiburg
Computer Networks and Telematics

Summer 2009

CoNe
Freiburg

IIF
INSTITUT FÜR
INFORMATIK
FREIBURG

Organization

- **Schedule**
  - Lecture: Wednesday 9-11 and Friday 11-12
  - Exercise: Friday 12-13

- **Requirements**
  - no formal requirements
  - basic knowledge of network protocols, programming and software engineering

Contents

- **Protocol design principles**
- **Modeling and specification**
- **Simulation of network protocols**
- **Performance evaluation**
- **Formal analysis**

Material

- **Slides will be available as PDF on the course website**
  http://cone.informatik.uni-freiburg.de/teaching/lecture/protocol-design-s09/

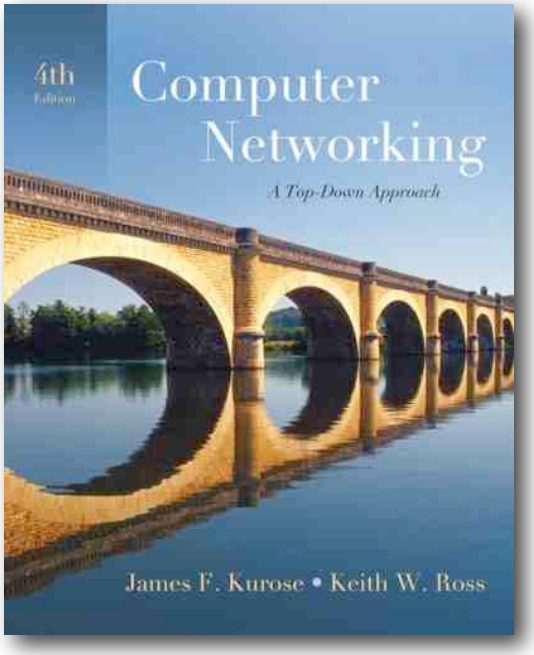
- **More material** (book chapters, research papers) related to special topics will be announced in the lecture

- **Books**
  There are lots of books on computer networking and protocols. Most of them focus on the architectures and algorithms, not so much on design aspects.

Books ...

- James F. Kurose, Keith W. Ross:
  Computer Networking -
  A Top-Down Approach,
  4/e, 2007, ISBN 0-321-49770-8

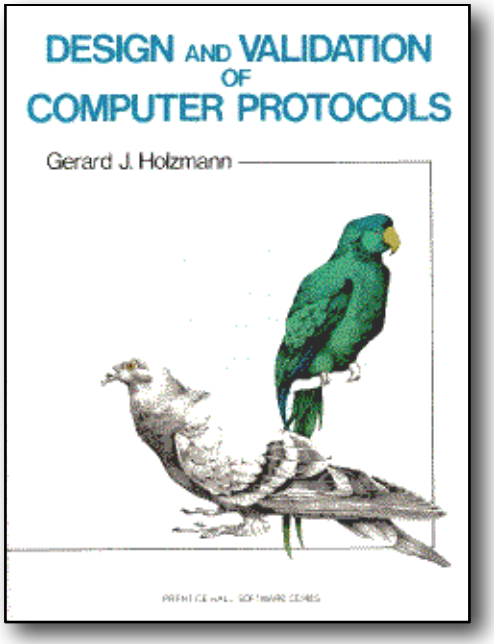
- Overview of computer networks, Internet protocols, from link layer to application layer.



Books ...

- Gerard J. Holzmann:
  Design and Validation of Computer Protocols, Prentice Hall, 1991,
  ISBN 0-13-539834-7

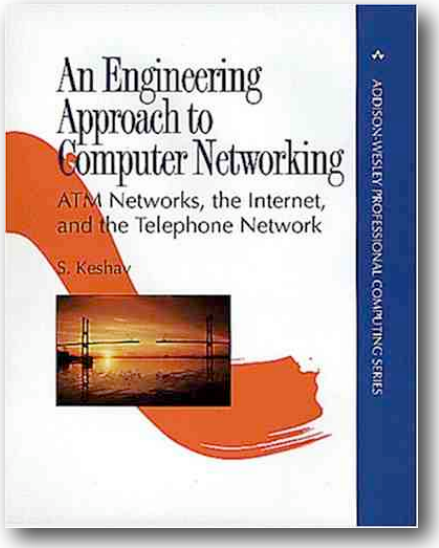
- Design principles of communication protocols, formal specification and validation



Books ...

- S. Keshav:
  An Engineering Approach to Computer Networking,
  Addison Wesley, 1997,
  ISBN 0201634422

- Overview of communication protocols, explains design decisions



Related lectures and courses

- **Systeme II**
  (highly recommended for this lecture)
- **Communication Systems**
- **Special lectures:** Ad hoc networks, Wireless Sensor Networks, P2P Networks
- **Lab course/Bachelor Projects:**
  Wireless Sensor Networks

In this introduction...

- **What is a protocol?**
- **Protocol design and what it is good for**
- **Examples for protocol design**
- **Basic elements of a protocol**

Motivation

- **Statement: Network protocols are standardized, they are tested and ready to use.**

- **Why Protocol Design?**
  - New demands by advances in communication technology
  - More distributed, net-based, and mobile applications
  - Customization, cross-layer optimization etc.

- **Important part of the design of distributed systems**

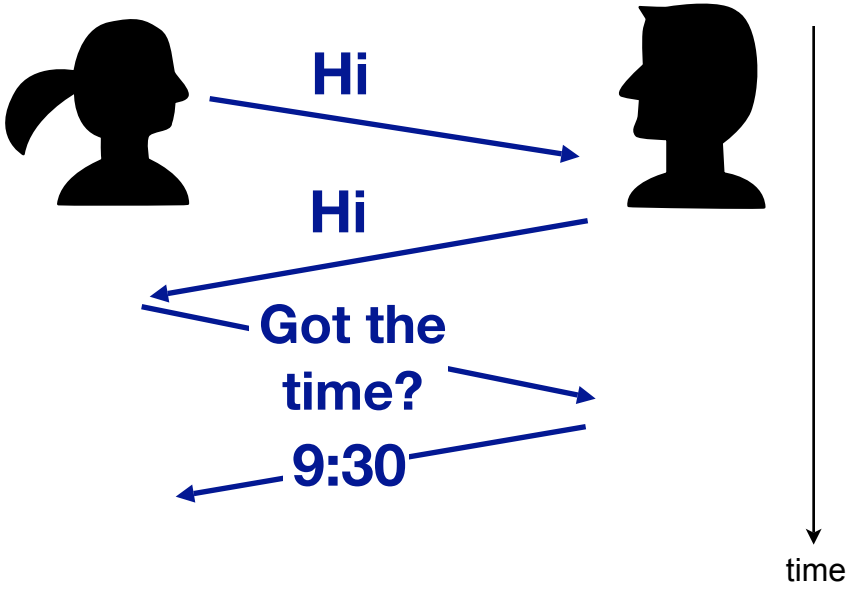
Motivation (2)

- **...and why protocol evaluation?**

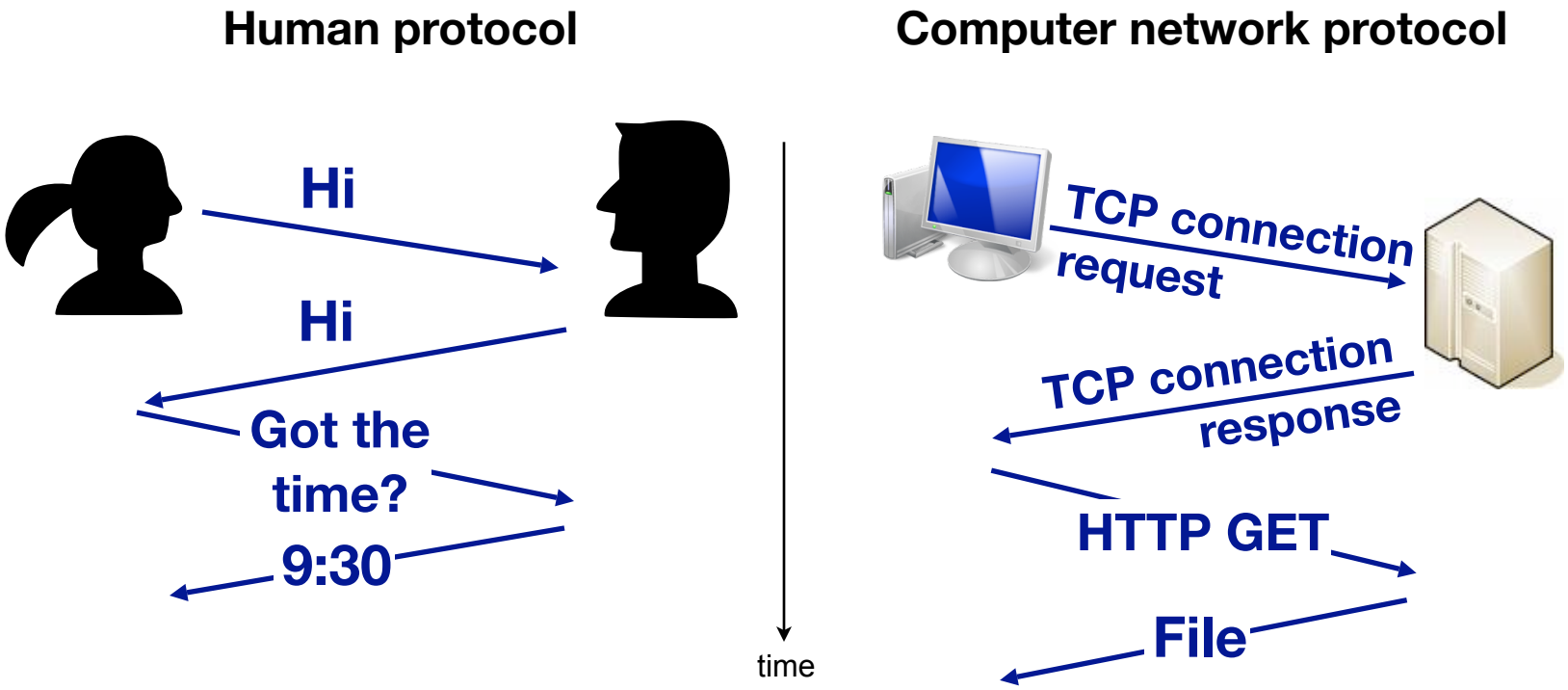
- **Testing and (problem) analysis**
  - for new protocols
  - if standard protocols are used in new contexts (e.g. Internet protocols in wireless networks)
  - if protocols of different layers are combined
  - if protocols share resources

What is a protocol?

- **Human protocol:**
  - "I have a question"
  - "What's the time?"
- **… specific msgs sent**
- **… specific actions taken when msgs received, or other events**



What is a protocol? (2)



What is a protocol? (3)

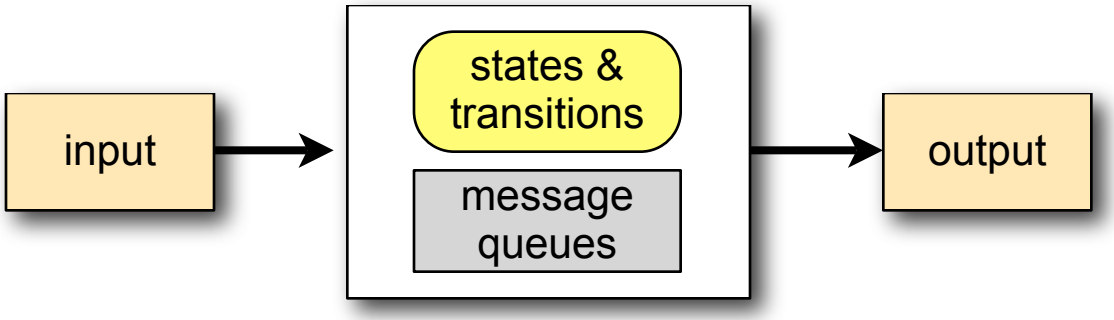
- **High level definition** (Holzmann 1991):
  Agreement on information exchange in distributed networking
- **A protocol defines...**
  - format for valid messages (*syntax*)
  - rules for data exchange (*grammar*)
  - a vocabulary of messages and their meaning (*semantics*)

  ... similar to a language

  (a refined definition will follow later)

What is a protocol? (4)

- **Low-level abstraction: A state machine**
- The **protocol state** defines which actions are performed and how to respond to **events**
- A communicating state machine consists of
  - set of states
  - state transitions
  - message queues



What is protocol design?

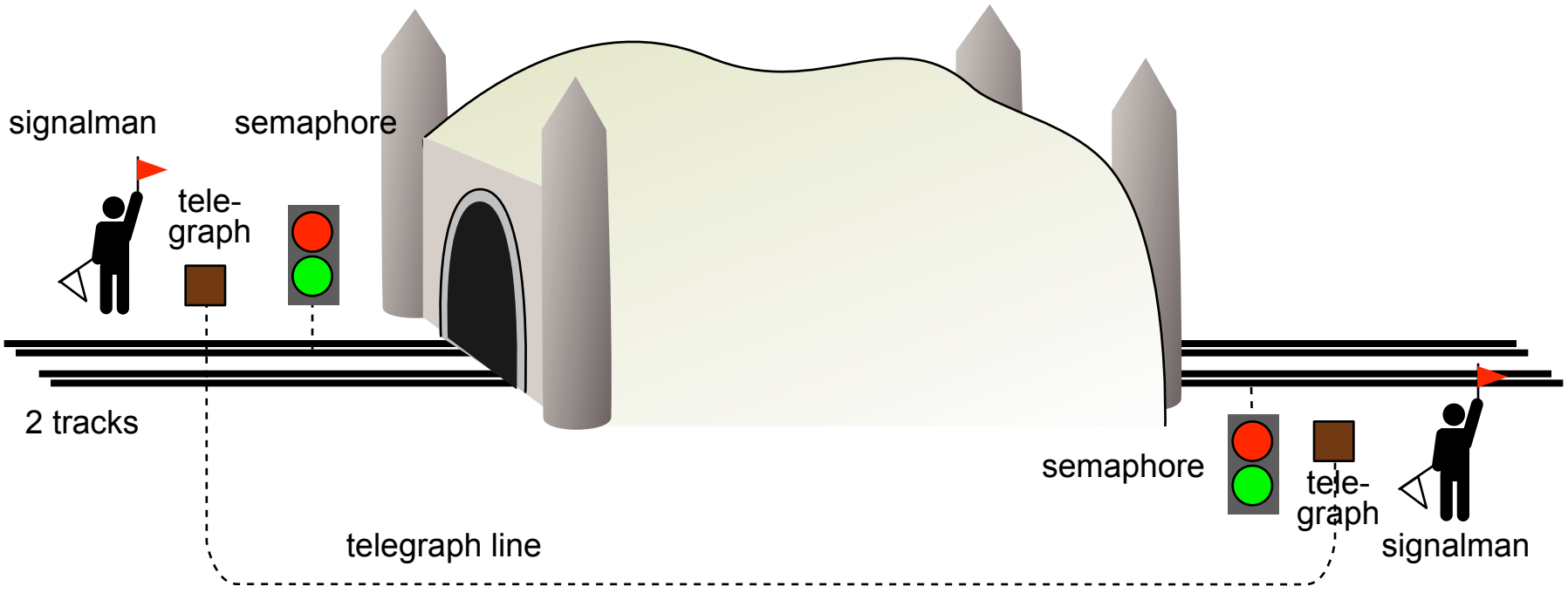
- **It's more than just implementation**
- **A development process:**
  - Requirements
  - Specification and validation
  - Implementation
  - Test and evaluation

- **...a challenging task, even for simple protocols**

Bad protocol design or human failure?

- The Clayton Tunnel accident: rail crash in Clayton, West Sussex, UK in 1861 due to a false signal
- Example for a failure of a simple communication protocol
- 21 dead, 176 injured

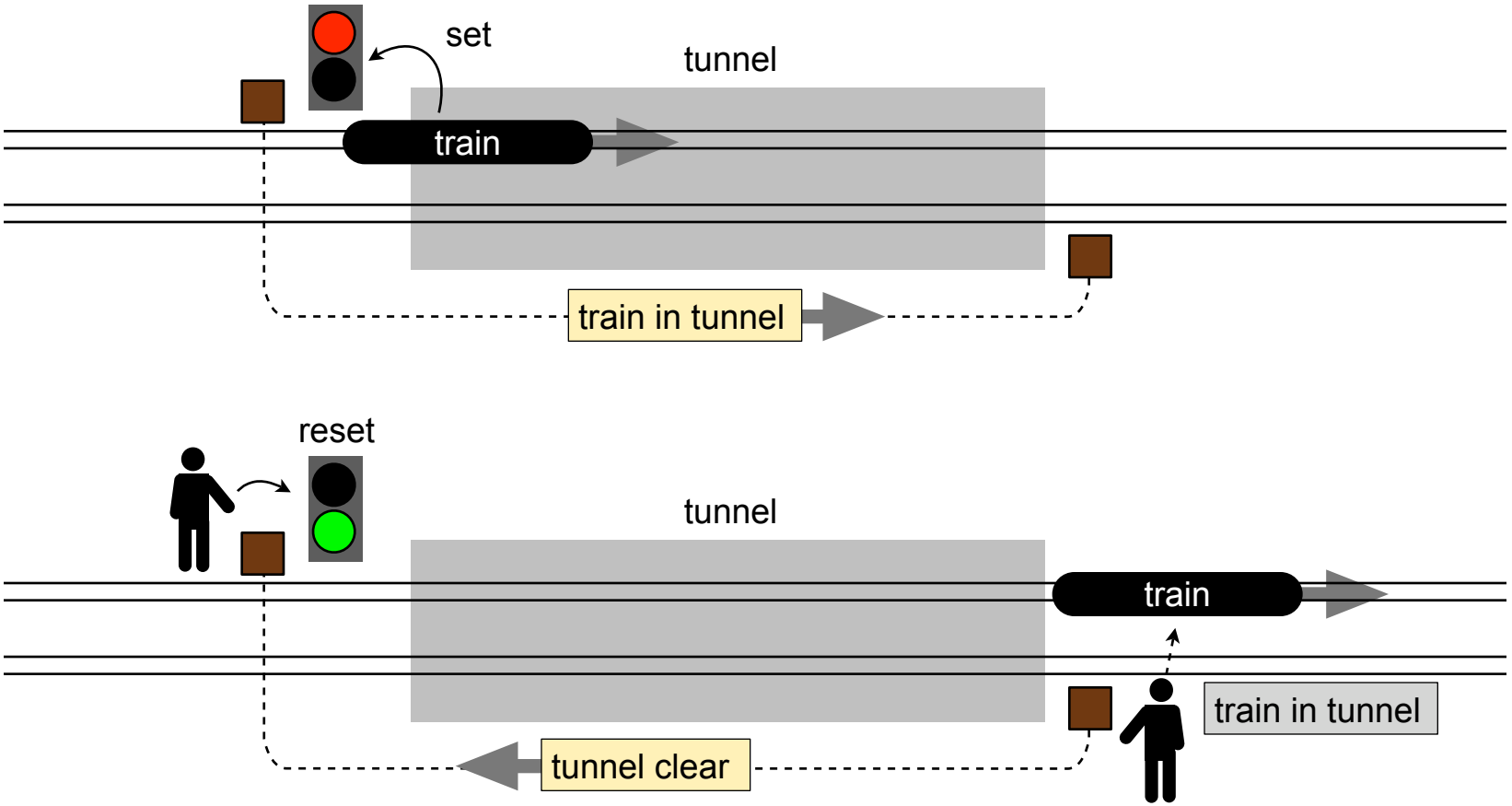
The setting



3 codes/messages: train in tunnel | tunnel clear | train in tunnel?

- A semaphore blocks automatically once a train passes.
- It is reset by a signalman, if the other signalman reports that the train left the tunnel

The protocol

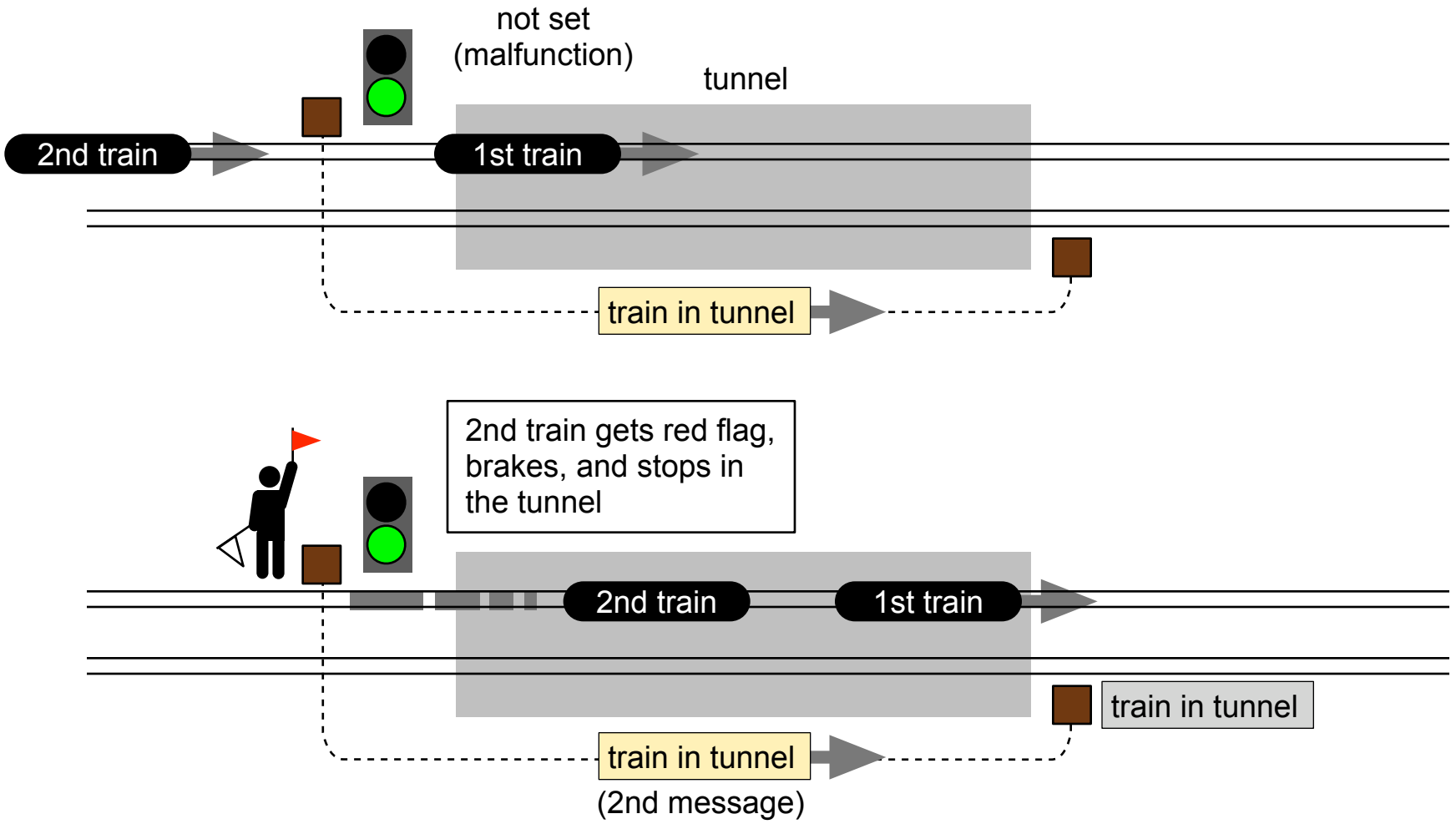


The protocol (2)

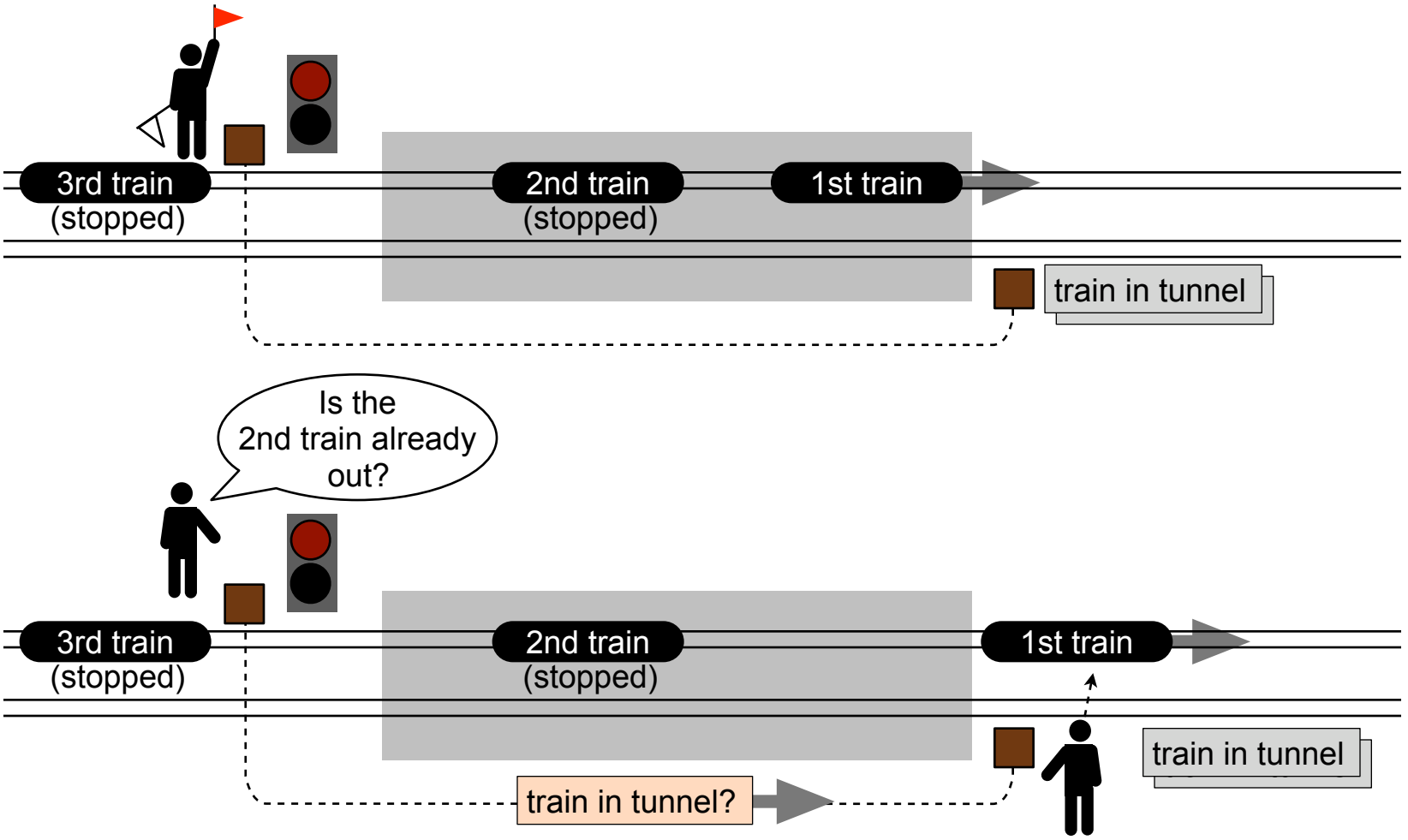
- The protocol should ensure that only one train per track is in the tunnel
- In case of a semaphore malfunction the signalmen are notified and use their flags
- The third message (“train in tunnel?”) is optional and can be used to request a message again

Is this protocol reliable?

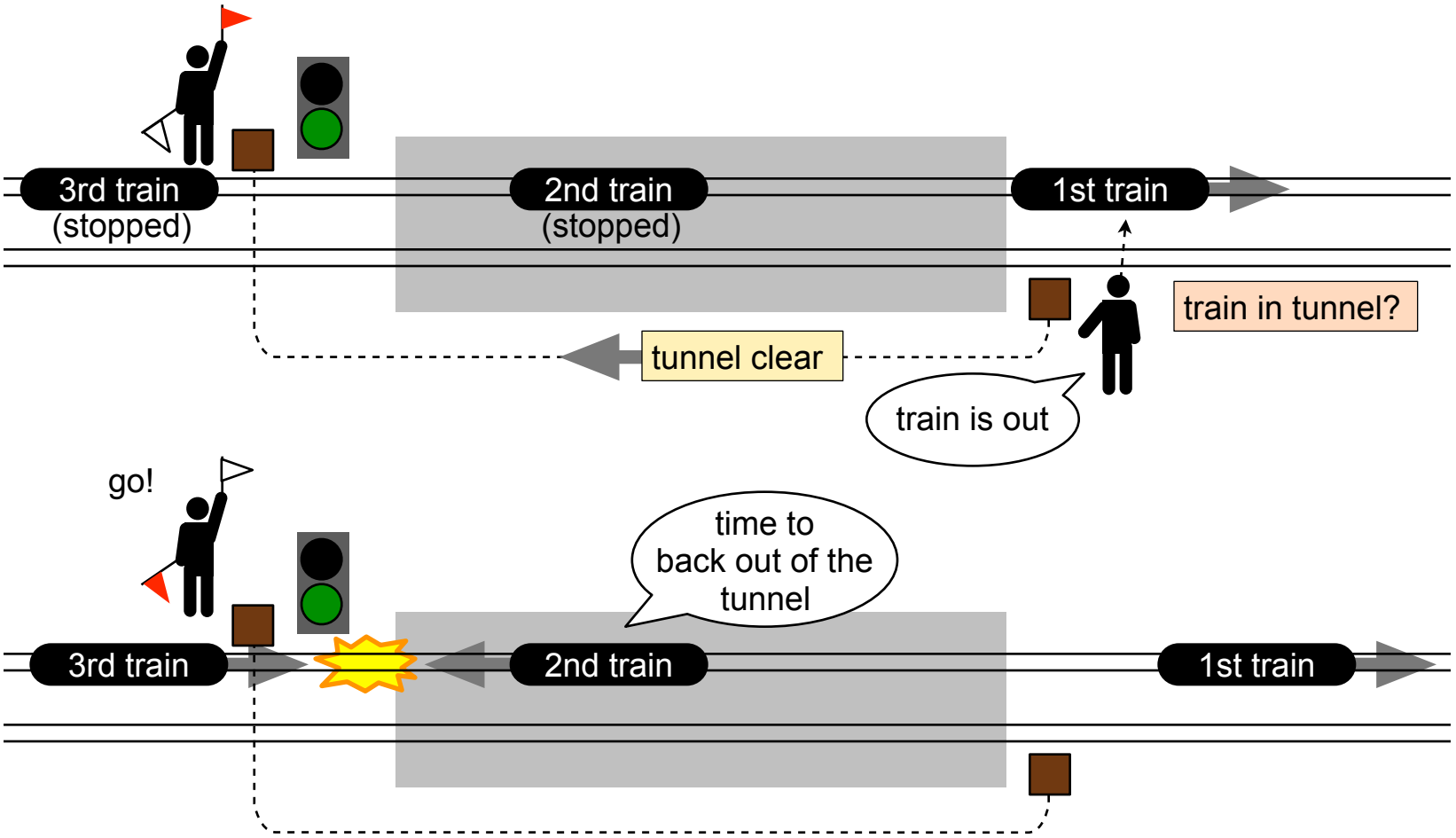
The accident (1)



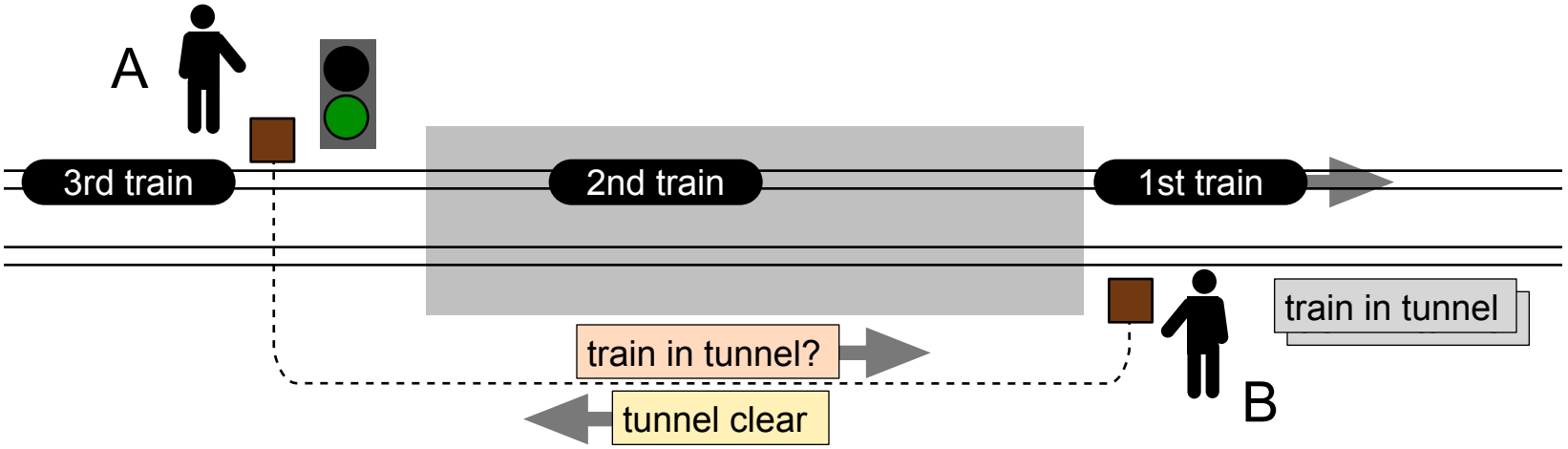
The accident (2)



The accident (3)



Who is to blame?



- The driver of the second train? ...but he followed the signal.
- Signalman A, who misinterpreted the “tunnel clear” message?
  ... but how could he know which train was meant?
- Signalman B, who did not react on two “train in tunnel” msgs.?
  ... but this was **not specified**!

Conclusion

- The signalmen did not have the appropriate set of messages
- An unexpected case occurred that could not be handled by the protocol.
- The protocol could not recover from an error, it was *incomplete* in this sense

Lessons learned

- Consider that errors might occur (expect the unexpected)
- Allow to handle unexpected errors
- Check whether unnecessary or invalid assumptions are made

The problem of protocol design

- **How to find the appropriate rules** for communication that are minimal, logically consistent, complete, and efficient?

  ... difficult.

- **First,** we begin with a thorough specification of the rules, the messages and assumptions about the environment, ... all **elements of the protocol**

Five elements of a protocol

Elements of a protocol specification (Holzmann 1997):

1. The **service** to be provided by the protocol
2. The **assumptions about the environment** in which the protocol is executed
3. The **vocabulary** of messages used to implement the protocol
4. The **encoding** (format) of each message in the vocabulary
5. The **procedure rules** guarding the consistency of message exchanges

Another example

- Lynch's protocol: A protocol for full-duplex alternating data transmission protocol for a half-duplex telephone line
- presented by W.C. Lynch as "*a reasonable looking but inadequate scheme published by one of the major computer manufacturers"*
- an example for the difficulty to design protocols

[W.C. Lynch: "Computer Systems: Reliable full-duplex file transmission over half-duplex telephone line", Comm. ACM, 1968]
see also [Holzmann 1991]

Lynch's protocol - specification

- **Service**: The protocol should enable reliable full-duplex message transmission (simultaneous transmissions in both directions). A message transmission is answered by a positive or negative acknowledgement.

- **Environment**:
  - Two users sharing one communication channel
  - Messages can be corrupted, but the are never lost. Corrupted messages are detected and turned into error messages

Lynch's protocol - specification

- **Vocabulary of messages**:

  Messages plus acknowledgement and error flag

  Vocabulary = {ACK, NACK, ERR}

- **Message format**:
  - data (char)
  - ack flag (binary)

    + error flag (binary)

Lynch's protocol - specification

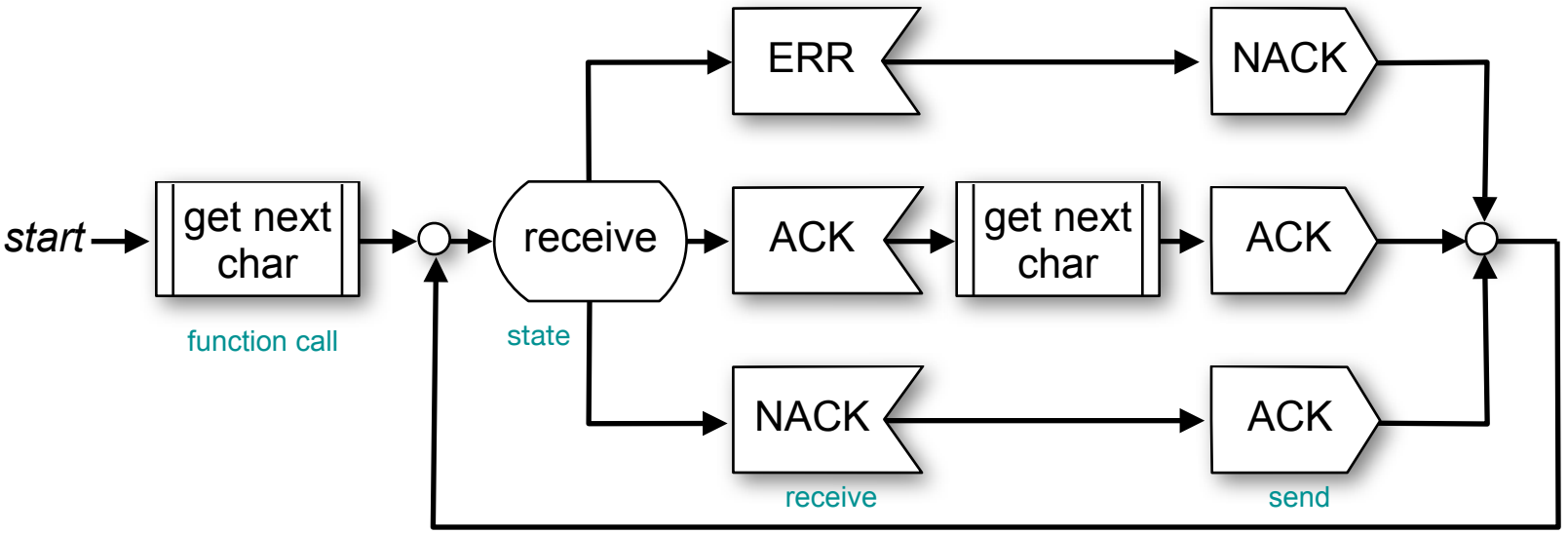
- **Procedure Rules** (informal):

  "1. If the previous reception was error-free, the acknowledge bit of the next transmission is one; if the reception was in error the bit is zero.

  2. If the acknowledge bit of the previous reception was zero, or the previous reception was in error, retransmit the old message; otherwise fetch a new message for transmission."

Lynch's protocol - specification

- **Procedure Rules** (formal):



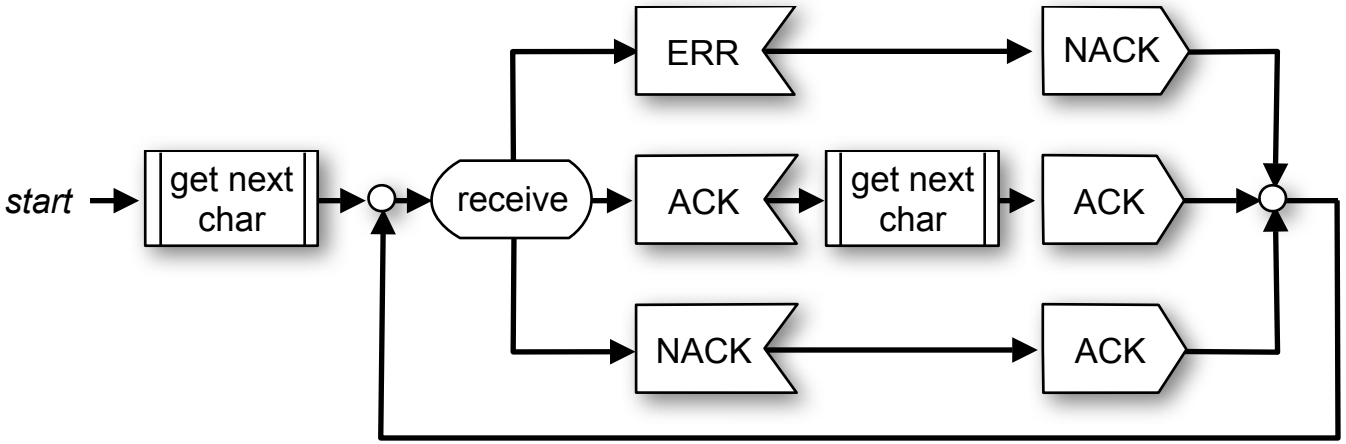
[Holzmann 1991]

Lynch's protocol - example

Terminal A			Terminal B		
msg	error	ack	msg	error	ack
a	OK	ACK			
			x	OK	ACK
b	ERR	(ACK)			
			x	OK	**NACK**
b	OK	ACK			
			y	OK	ACK

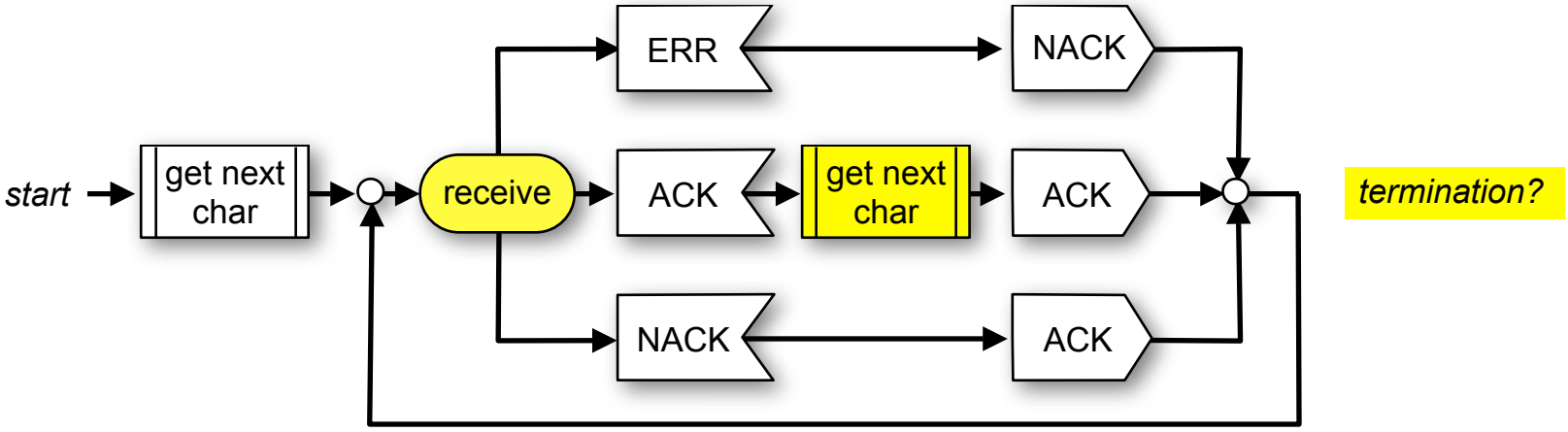
retransmit

error received



Specification problems

- **Design flaws:**
  - Transmission in one direction does not continue if get_next_char does not return data
  - Initiation and termination not specified

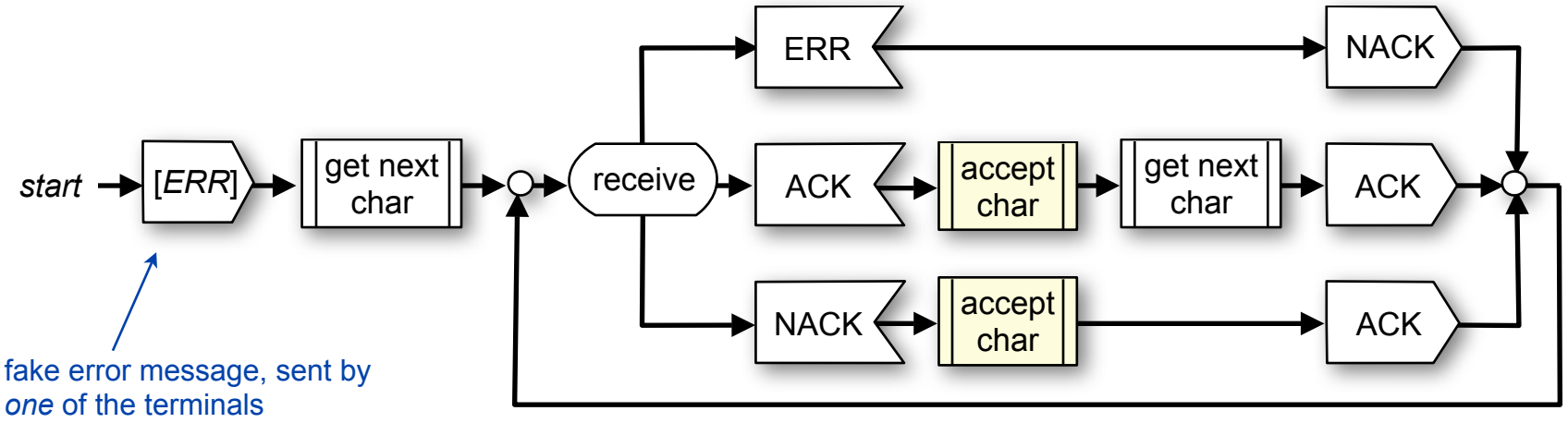


Specification problems

- **An attempt to repair:**
  - One-way transmission: Use filling messages, such that *get_next_char* does not block
  - Initiation by sending a fake error message, such that *receive* does not block
  - How to terminate after exchanging filler messages?

Specification problems

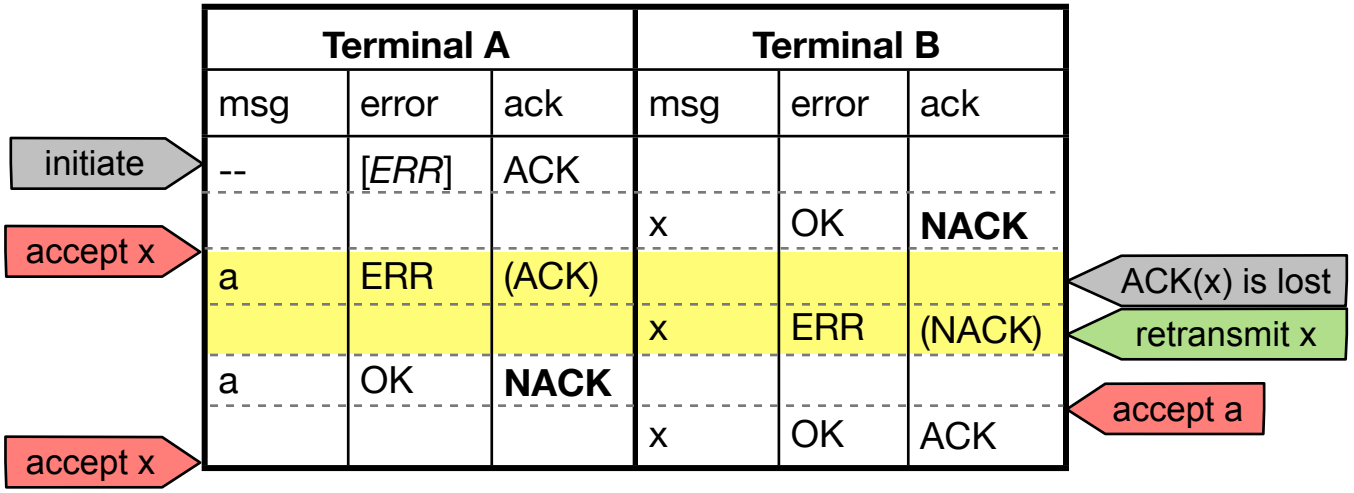
- **More problems:** Question when to accept a message is left open. If a terminal accepts all error-free messages then duplicates cannot be detected.



Specification problems

- **Example for the duplication problem:**

  A's sequence: a, b, c; B's sequence: x, y, z



Lessons learned

- Design flaws are not always obvious
- Some errors occur very rarely.
  This makes it very difficult to detect a design flaw.
- Flaws are usually difficult to fix.
- Even simple protocols require a good design.