The FlexRay Protocol

Peter Böhm

27.9.05

Overview

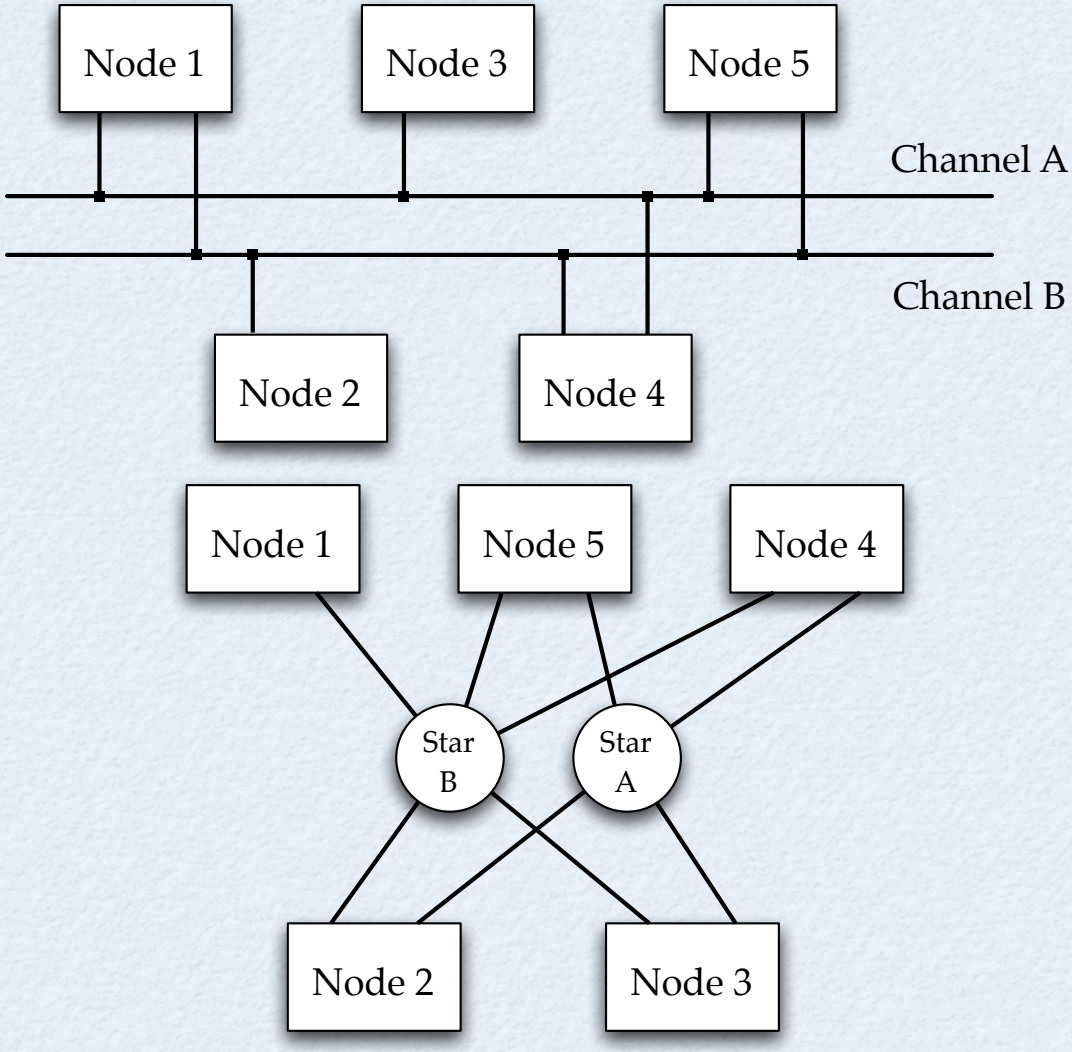
1. Introduction
2. Network Topology
3. Nodes
4. Communication Controller
5. Schedule
6. Message Processing
7. Clock Synchronization
8. Wake-up/Start-up
9. Summary

1. Introduction

- FlexRay: Communication in distributed systems within automotive context
- developed by the FlexRay consortium (BMW, DaimlerChrysler, Motorola, Philips) founded in 1999
- since 1999 many well-known companies joined (e.g. Bosch, GM, VW, Mazda, etc.)
- aim: flexible, fault-tolerant communication protocol

2. Network Topology

2 typical network topologies:

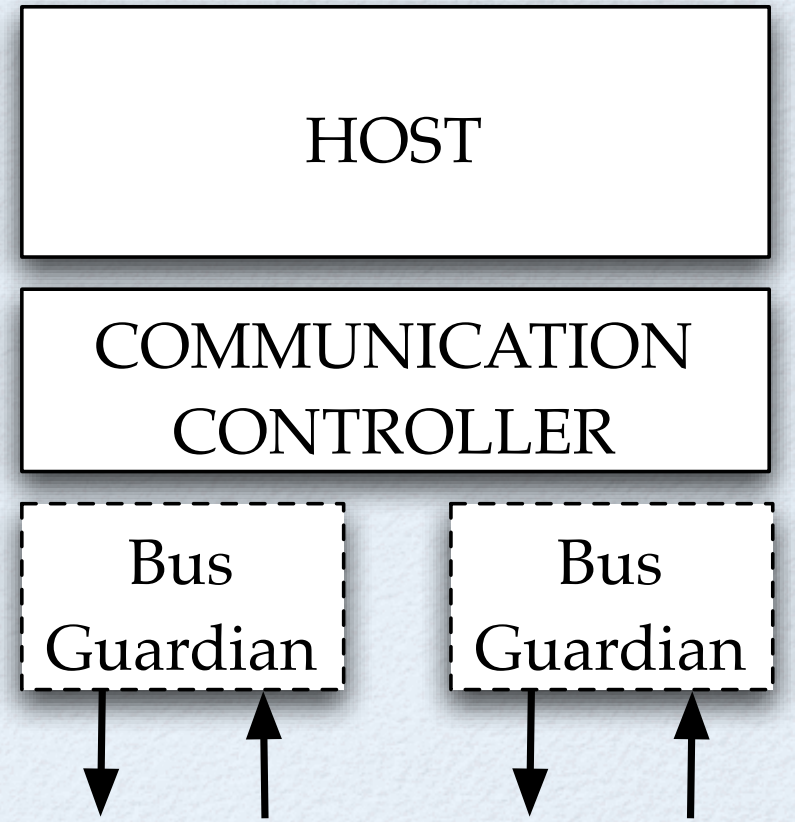


- star, bus topology or combination
- max. 2 channels
- optional bus guardians

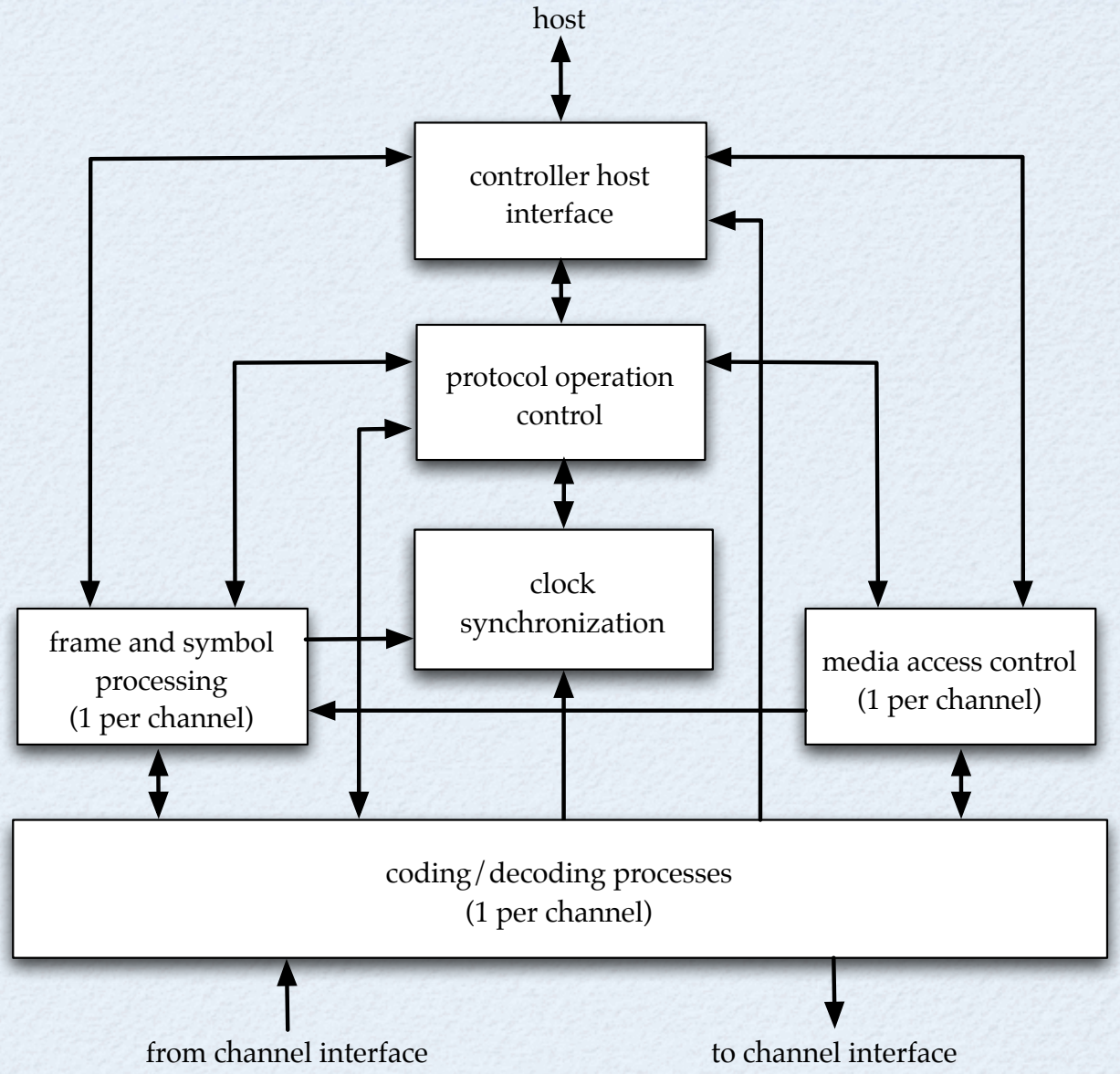
➡ various, flexible network topologies

3. Nodes

- main interest:
  communication controller (CC)
- CC's task:
  - interface to host
  - message processing
    - transmission
    - reception
  - clock synchronization



4. Communication Controller



4. Communication Controller

- Controller Host Interface:
  - interface between host and controller
    - control command interface
    - message interface
  - handles configuration and status data
  - message buffers for reception and transmission
- Protocol Operation Control
  - purpose: react to host commands and protocol conditions
  - change operation modes of core processes
- Clock Synchronization
  - 3 parts: macrotick generation, clock synchronization and clock synchronization startup
  - macrotick: smallest synchronized time unit

4. Communication Controller

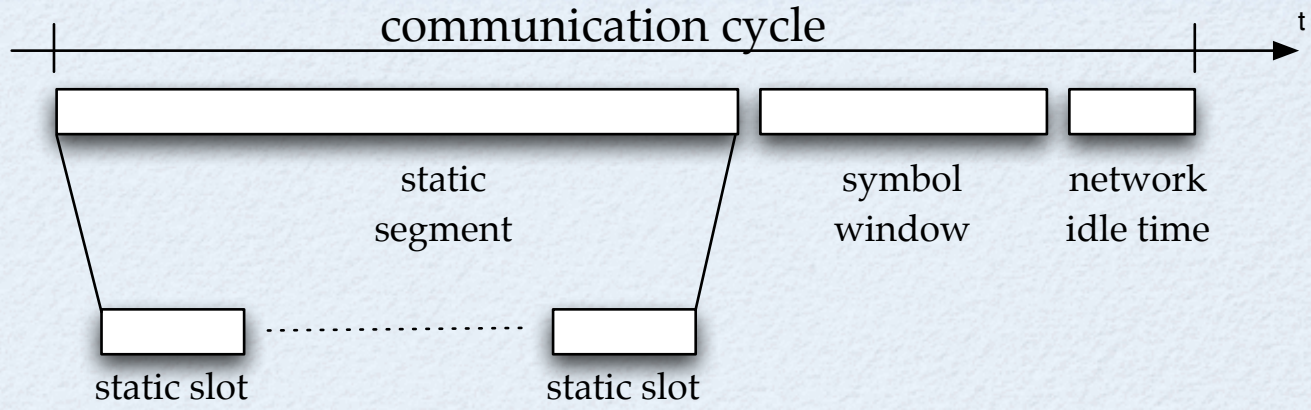
- Media Access Control (Transmission)
  - schedules the bus write accesses
  - assembles message header
- Frame and Symbol Processing (Reception)
  - handles received messages
  - performs timing and error checks; e.g. syntax tests, etc.
- Coding/Decoding Processes (Read/Write)
  - encodes frames for transmission, i.e. each bit 8 times on bus
  - decodes received frames
  - appends CRC for transmission
  - CRC check on received frames

5. Schedule

- time-triggered
  - time-devision multiple access (TDMA)
  - fixed time intervals for bus writing
  - fixed assignment: node → intervals

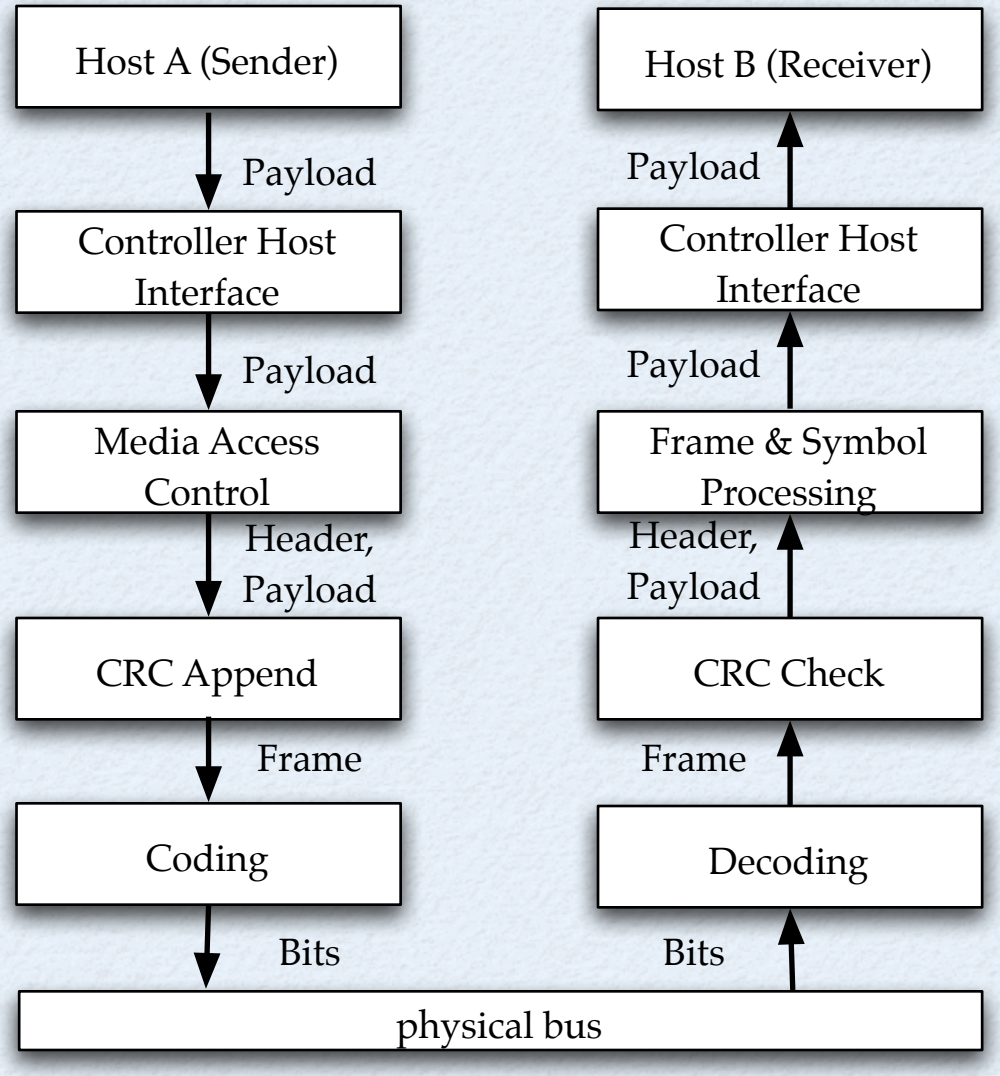
    ➡ static, deterministic schedule
  - nodes: only list with own transmission times
  - different approach: event-triggered
- fundamental element: communication cycle
  - periodically, recurring time unit
  - whole schedule executed once

5. Schedule: Communication Cycle



- static slot:
  - 1 message per static slot
  - all same length, i.e. same amount of macroticks
  - TDMA part of schedule
  - unique, fixed assignment to a node
- symbol window:
  - special messages, called symbols
  - wake-up symbol
- network idle time:
  - needed for clock synchronization

6. Message Processing



7. Clock Synchronization

- problem:
  - ➡ physical clocks deviate
  - ➡ TDMA-schedule: consistent view of time required to ensure communication
- synchronization of local clock against a fictive global clock
- fictive global clock derived from some node's view of time
- FlexRay clock synchronization provides:
  - ability to use the most accurate clocks for synchronization
  - fault-tolerance

8. Wake-up/Start-up: Error Model

- 3 level error model
- active
  - normal operation
  - no error state
- passive
  - an error occurred (e.g. clock synchronization failed)
  - node does not transmit and just listens the bus
  - trying to reintegrate
- halt
  - entered on host request or a fatal error detection
  - node completely stops operation

8. Wake-up/Start-up

- wake-up/start-up strategy needed after:
  1. power-on
  2. entering passive mode
- power-on: nodes start with non-synchronized clocks
  - some nodes serve at masters
  - others adopt their view of time
- passive mode: (e.g. due to clock synchronization failure)
  - node need to reintegrate itself
  - performs clock synchronization until its view of time is corrected

9. Summary

- very flexible network topology
  - ➡ scalable fault-tolerance
- time-triggered schedule with no common knowledge
- fault-tolerant message transmission with error checks
- fault-tolerant clock synchronization
- passive mode
  - ➡ self-diagnostic error mechanism with possible reintegration

➡ flexible as well as fault-tolerant

communication protocol