FlexRay

Bernd Finkbeiner, Rüdiger Ehlers, Markus N. Rabe, Sebastian Hahn,
Michael Gerke

Reactive Systems Group
Saarland University
Germany

12.06.2012

If you hit the brakes ...

... and they don't work:

Brakes should work!

Drive-by-Wire

FlexRay

Drive-by-Wire

FlexRay

FlexRay Bus Protocol

FlexRay™

FlexRay

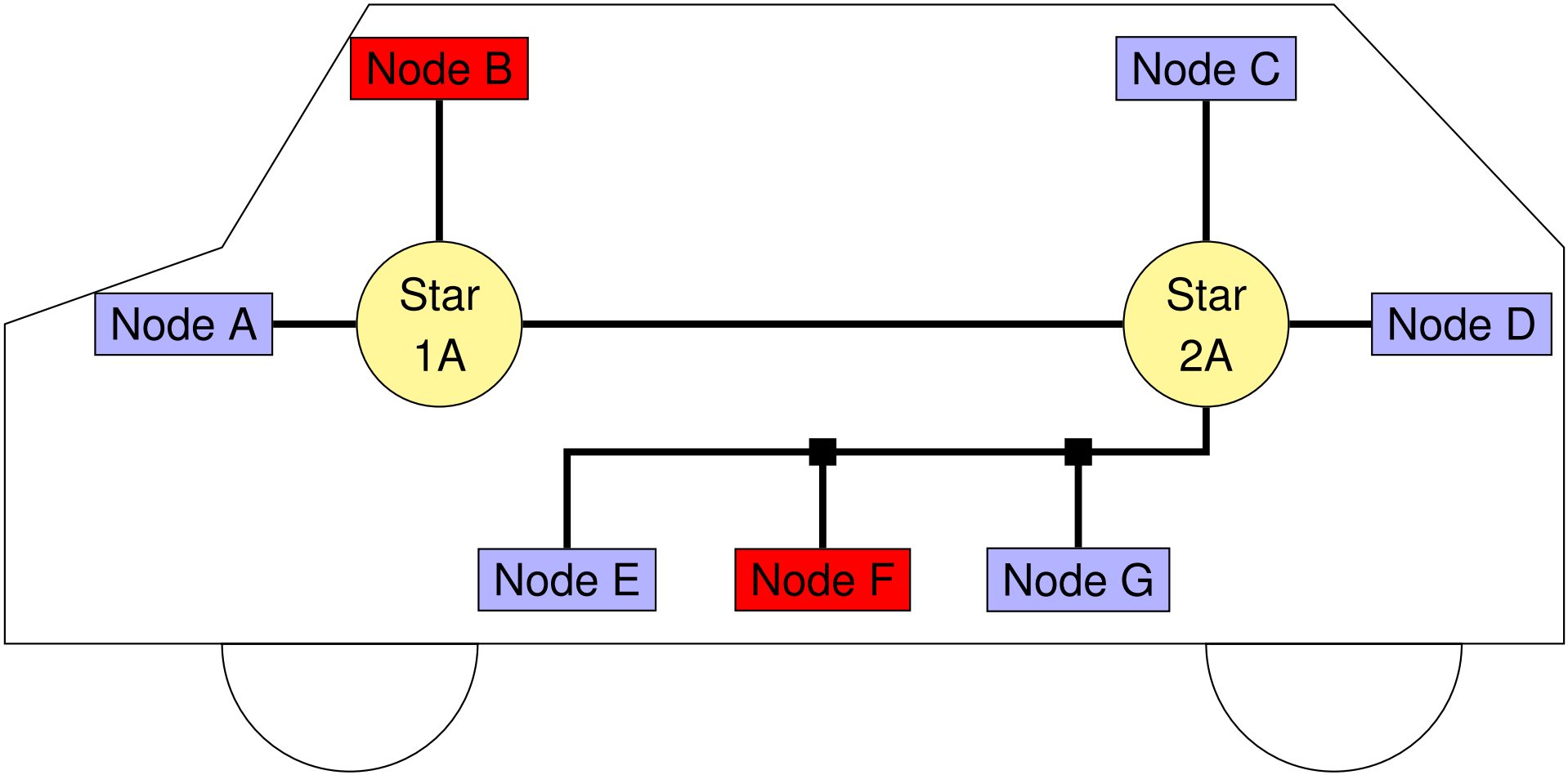
- communication protocol for distributed components in cars
- used in BMW X 5 and BMW's 7 series for X-by-wire
- developed by: BMW, Bosch, Daimler, Freescale, General Motors, NXP Semiconductors, Volkswagen, et al.

FlexRay Goals

Goals of the FlexRay design:

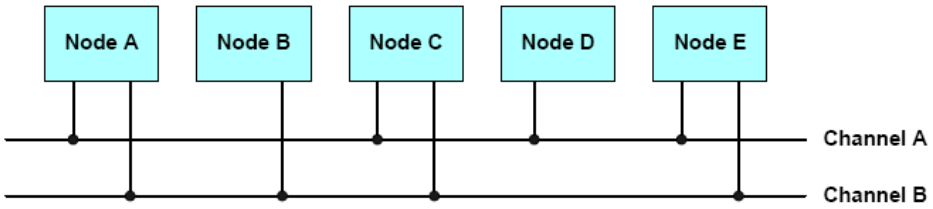
- High data rates (up to 10 megabits/second)
- Time- and event triggered communication
- Fault tolerance and redundancy

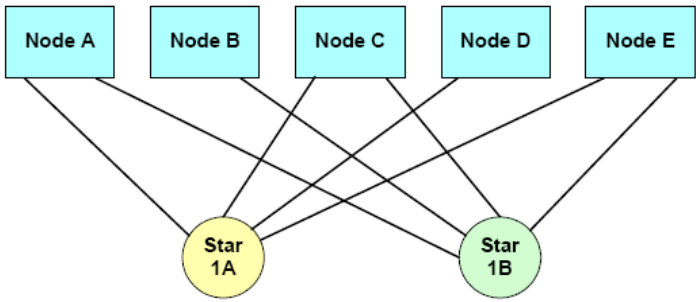
FlexRay Network



FlexRay Network Topologies

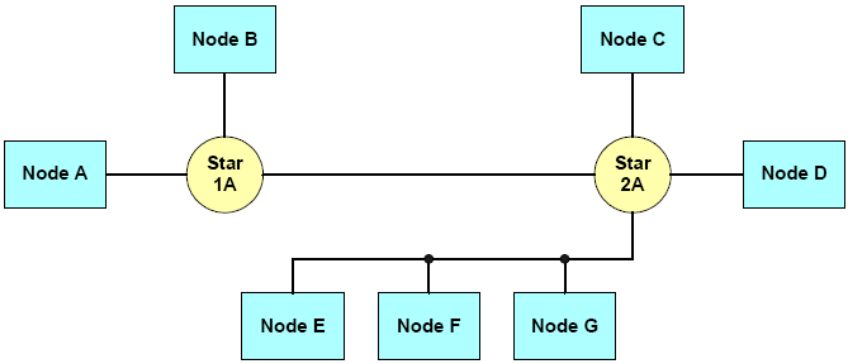
Dual channel bus





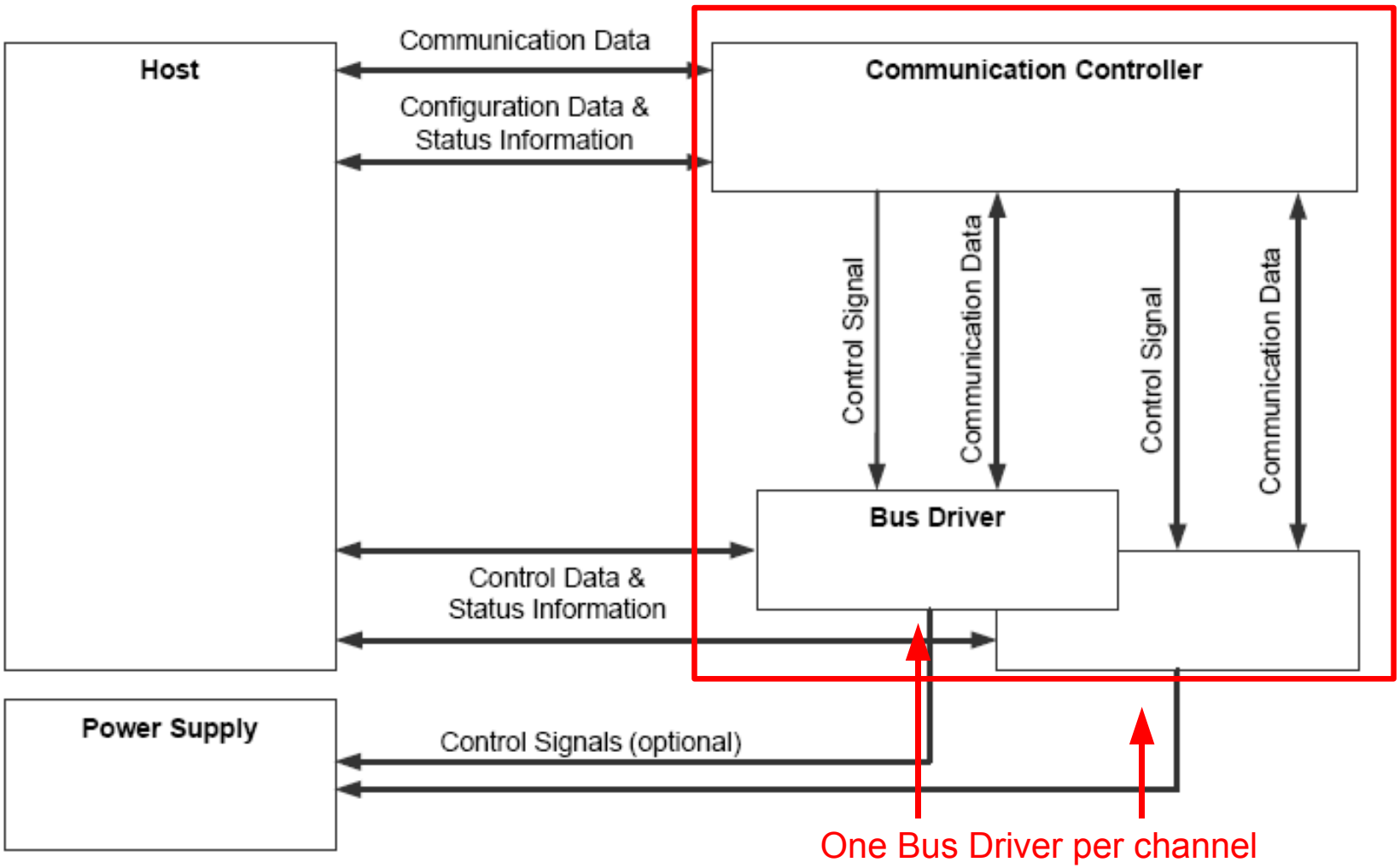
Dual channel star architectures

Mixed architectures

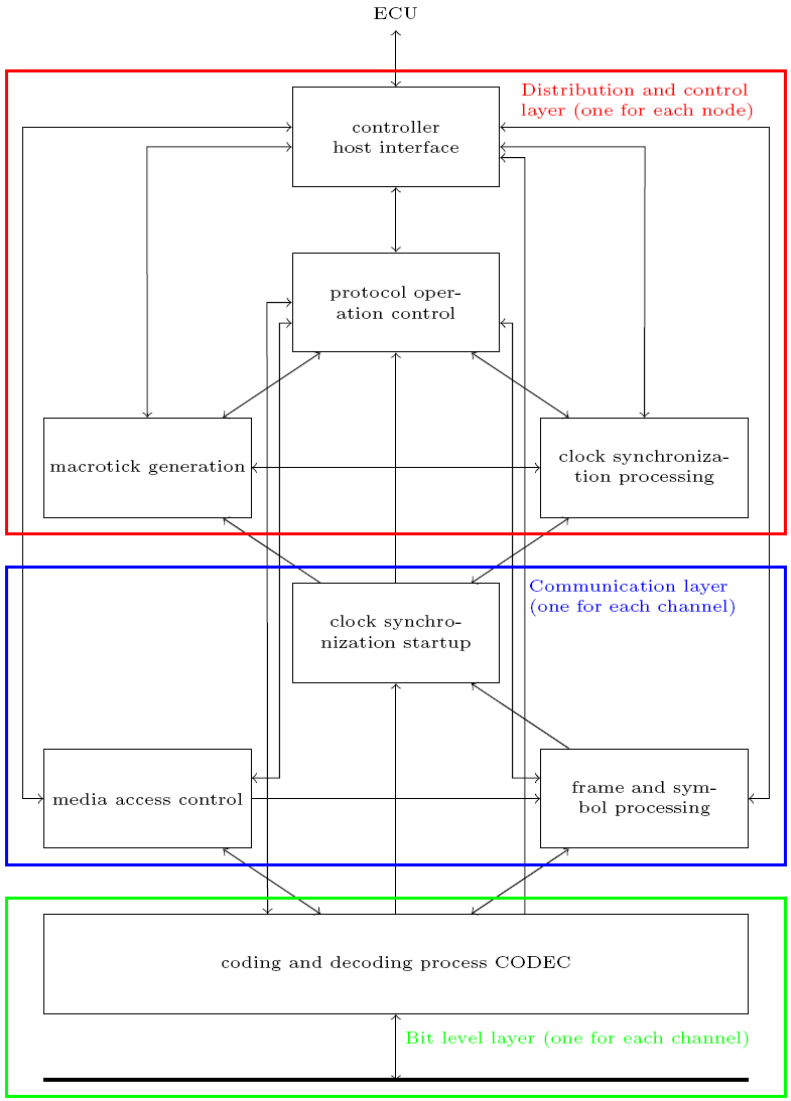


→ up to 2 channels supported

FlexRay Node



Architecture of a Bus Controller



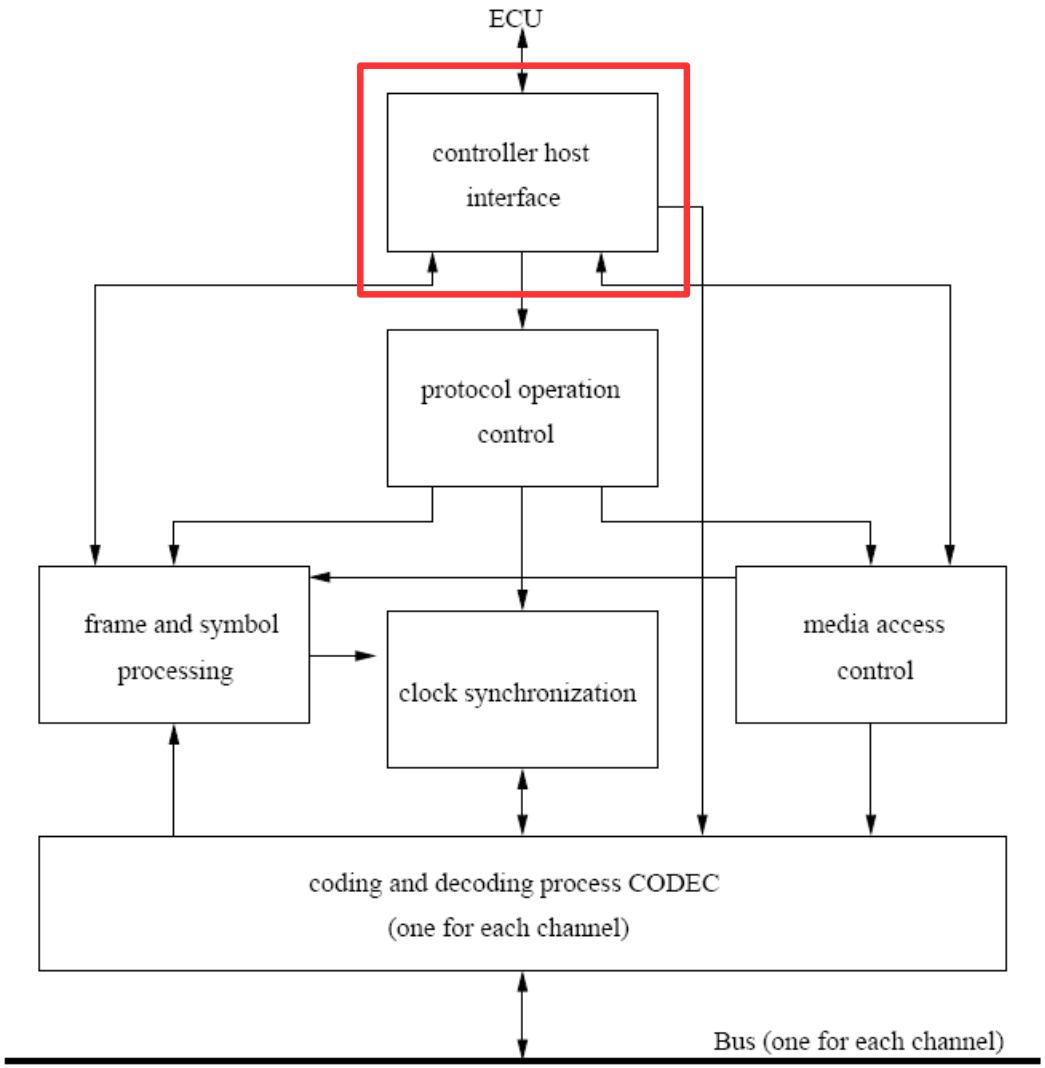
Roughly divided into 3 layers:

Distribution and control layer (1 for each node)

Communication layer (1 for each channel)

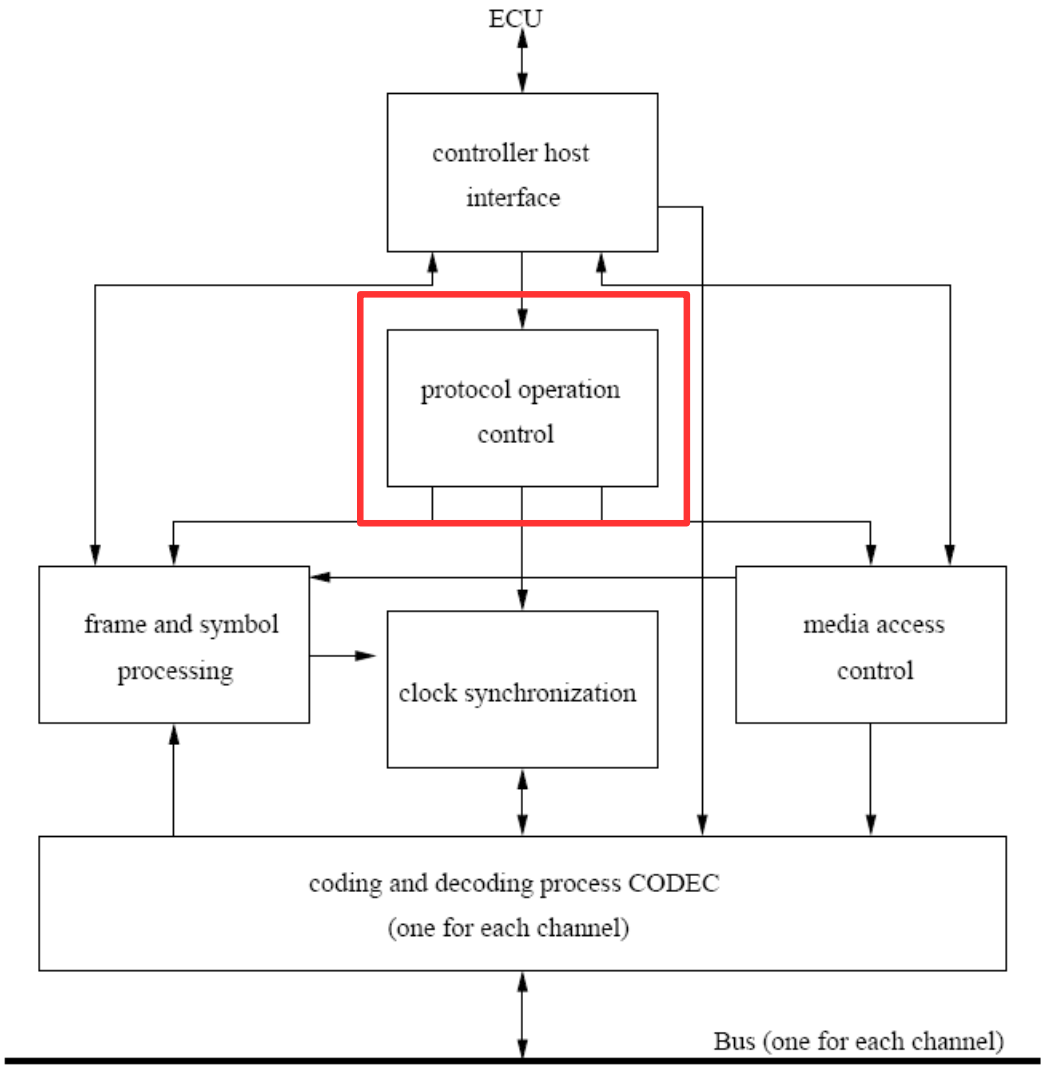
Bitlevel layer (1 for each channel)

Controller Host Interface



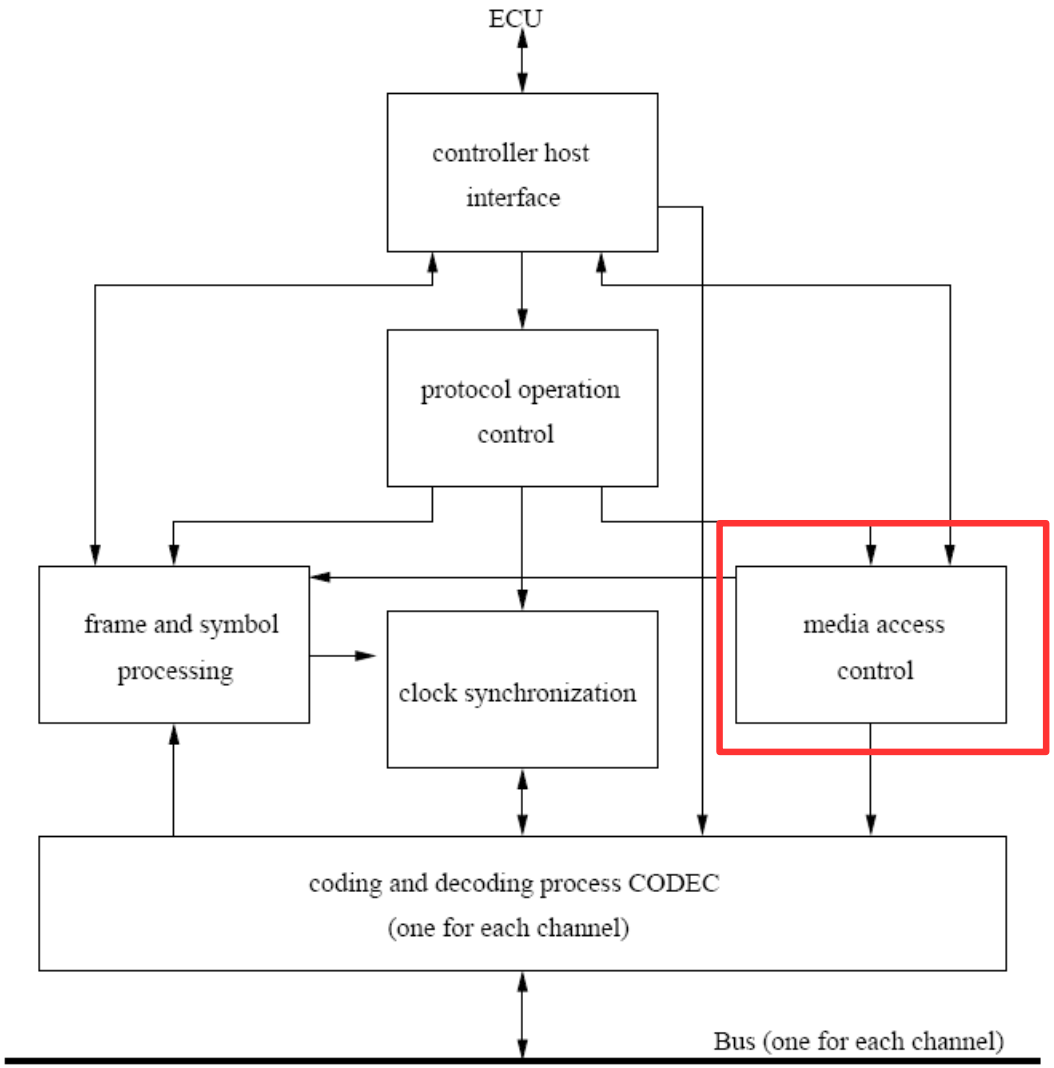
- well defined interface that controls all communication between host and bus driver
- offers host control over configuration (depending on phase)
- offers aggregated status reports to the host
- forwards the hosts commands to the relevant subprocesses
- buffers incomming and outgoing communication

Protocol Operation Control



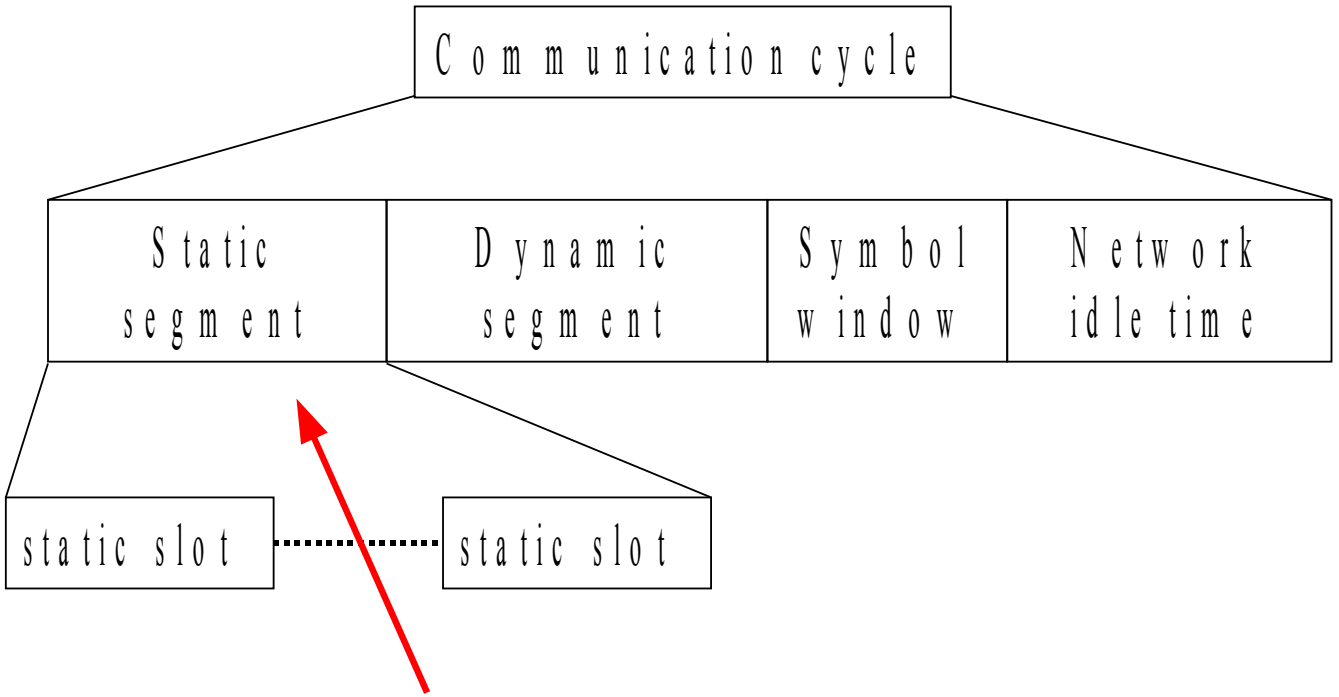
- controls status of the other subprocesses
- controls overall protocol behavior

Media Access Control



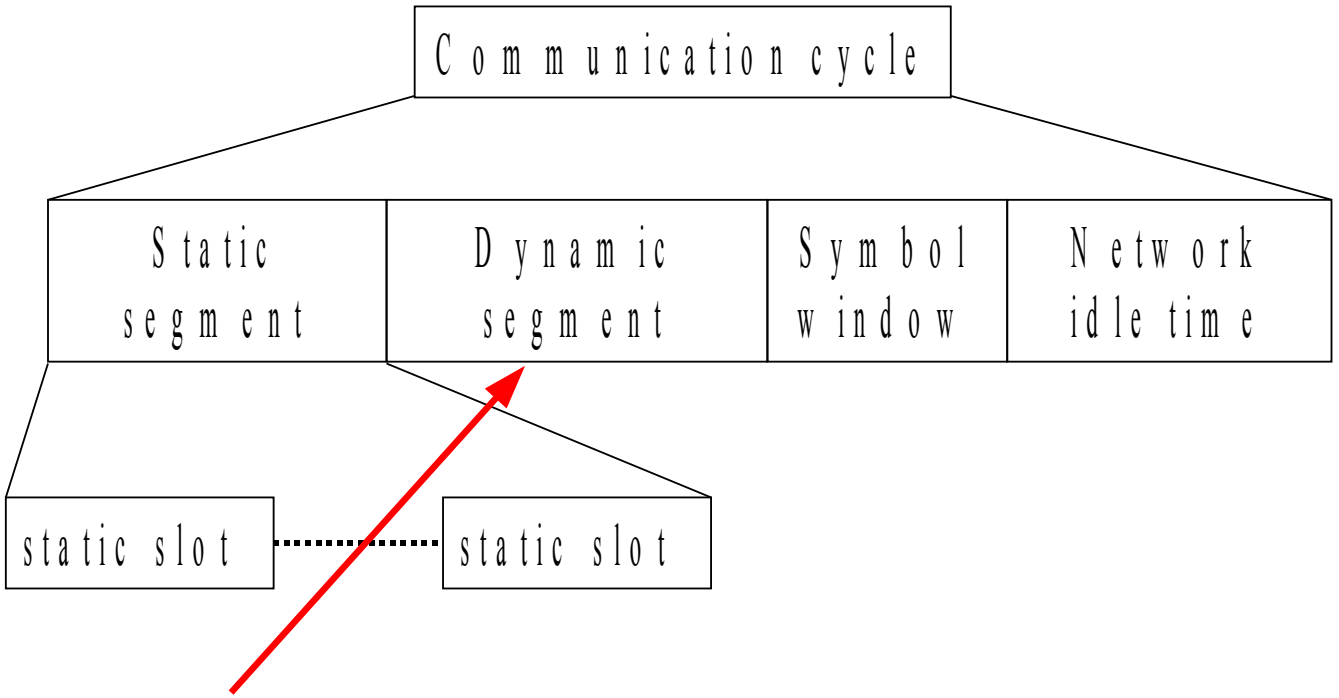
- controls access to the bus
- uses a Time Division Multiple Access (TDMA) scheme
- guarantees compliance with schedule
- assembles frame header

Time Division Multiple Access



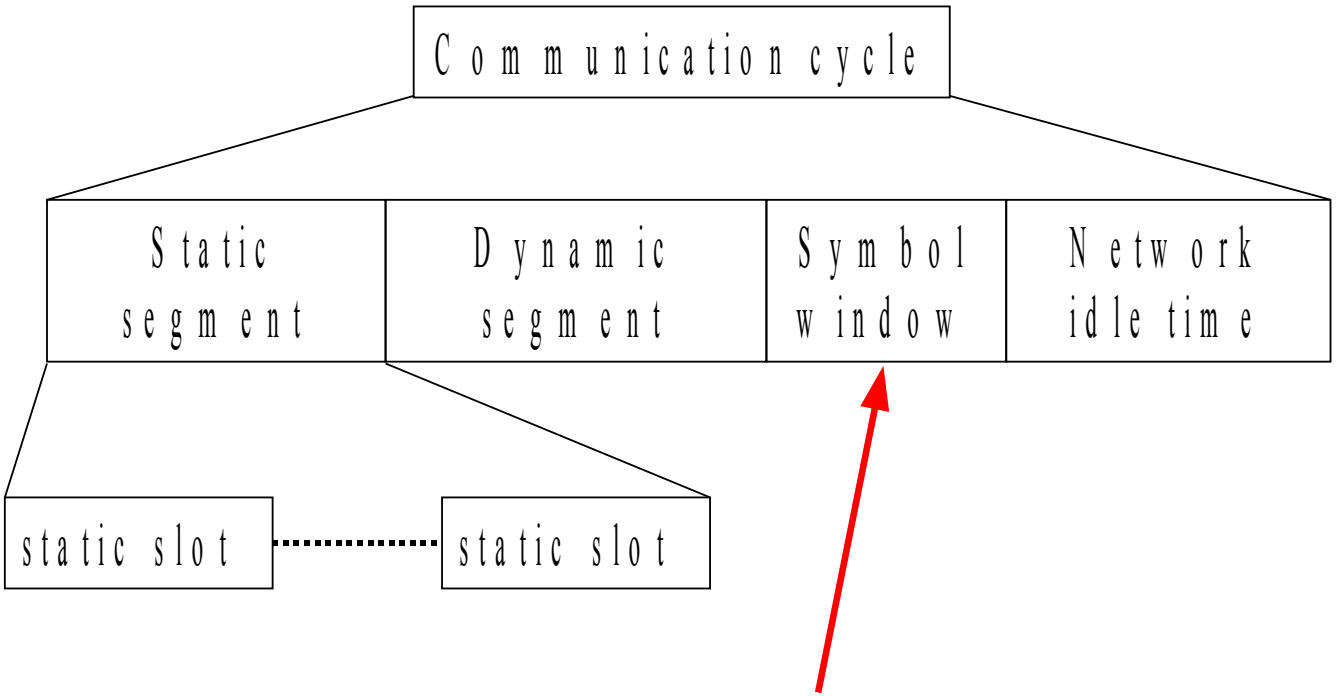
Used for normal communication, has a fixed schedule.

Time Division Multiple Access



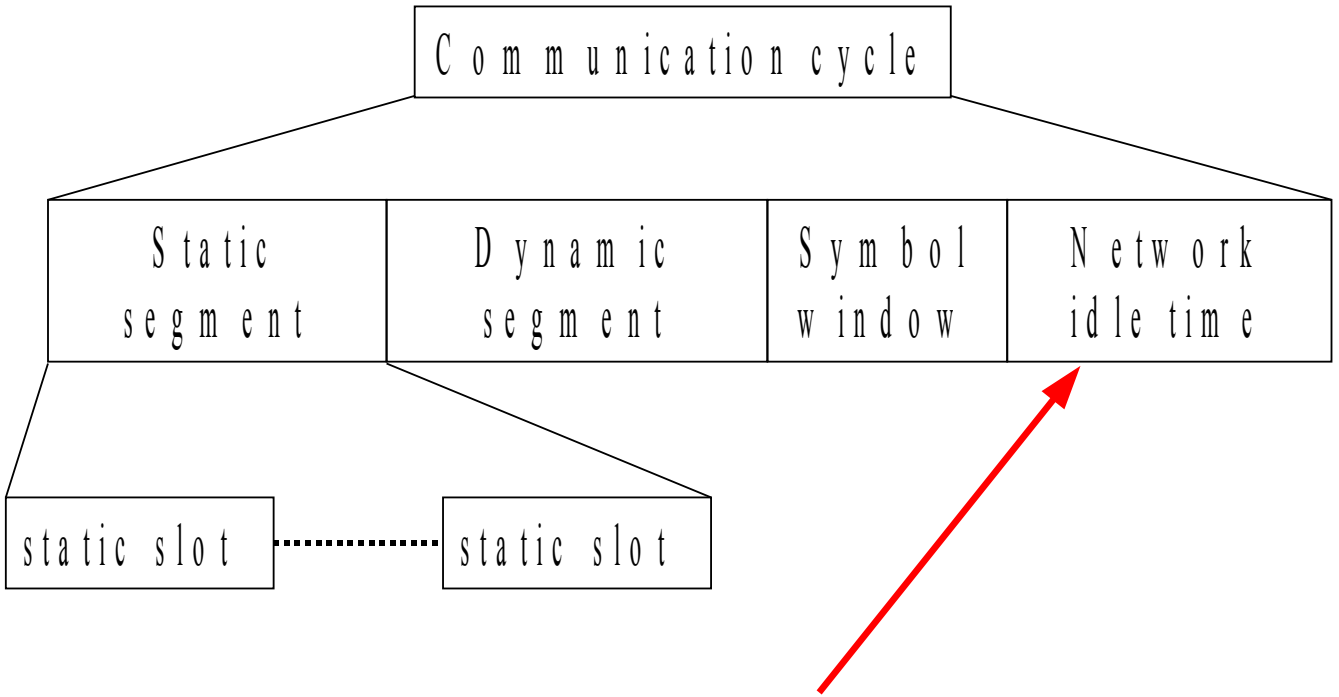
Used for irregular communication, bus arbitration is needed.

Time Division Multiple Access



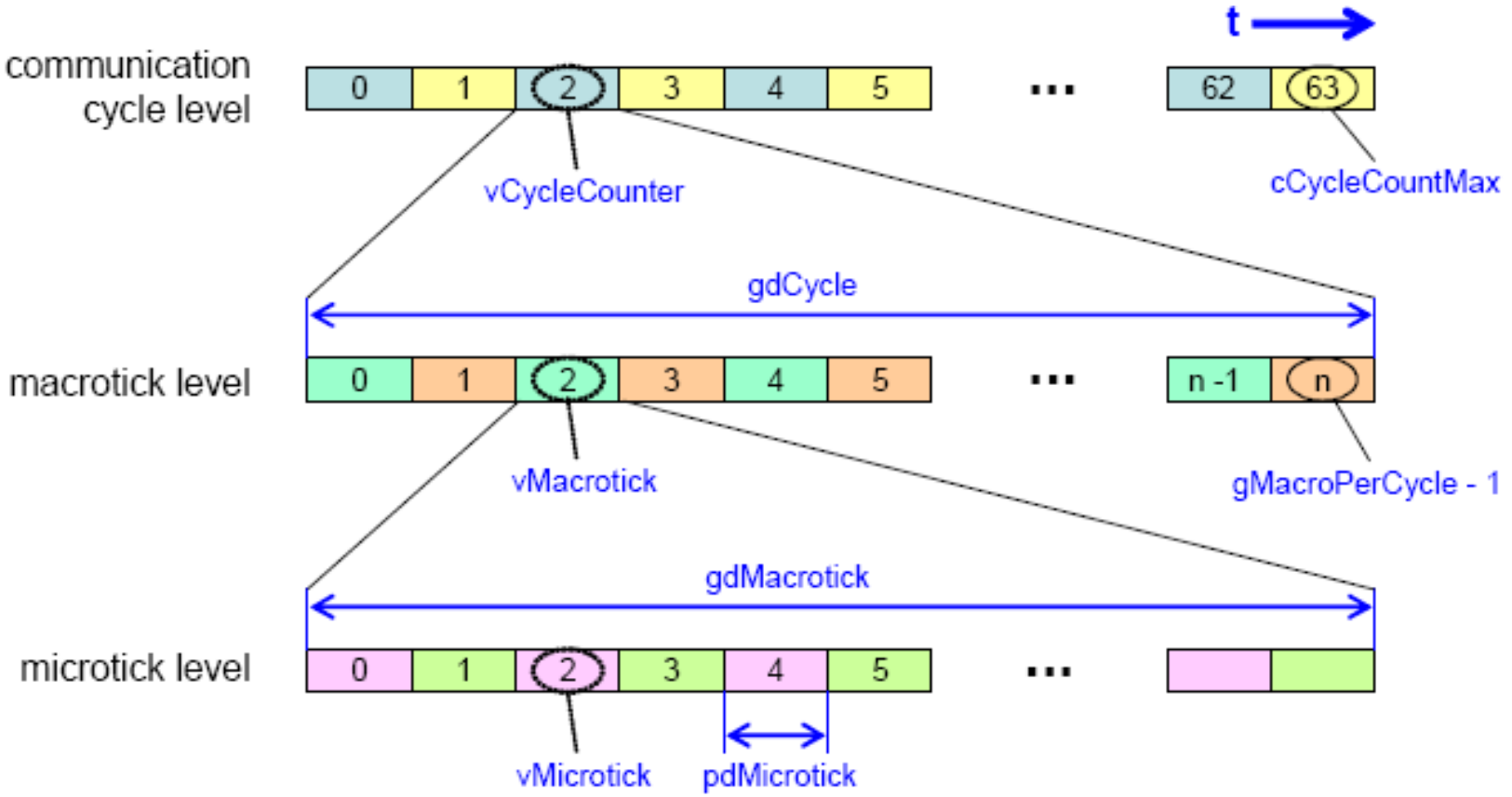
Used for symbol transmission, no bus arbitration.

Time Division Multiple Access

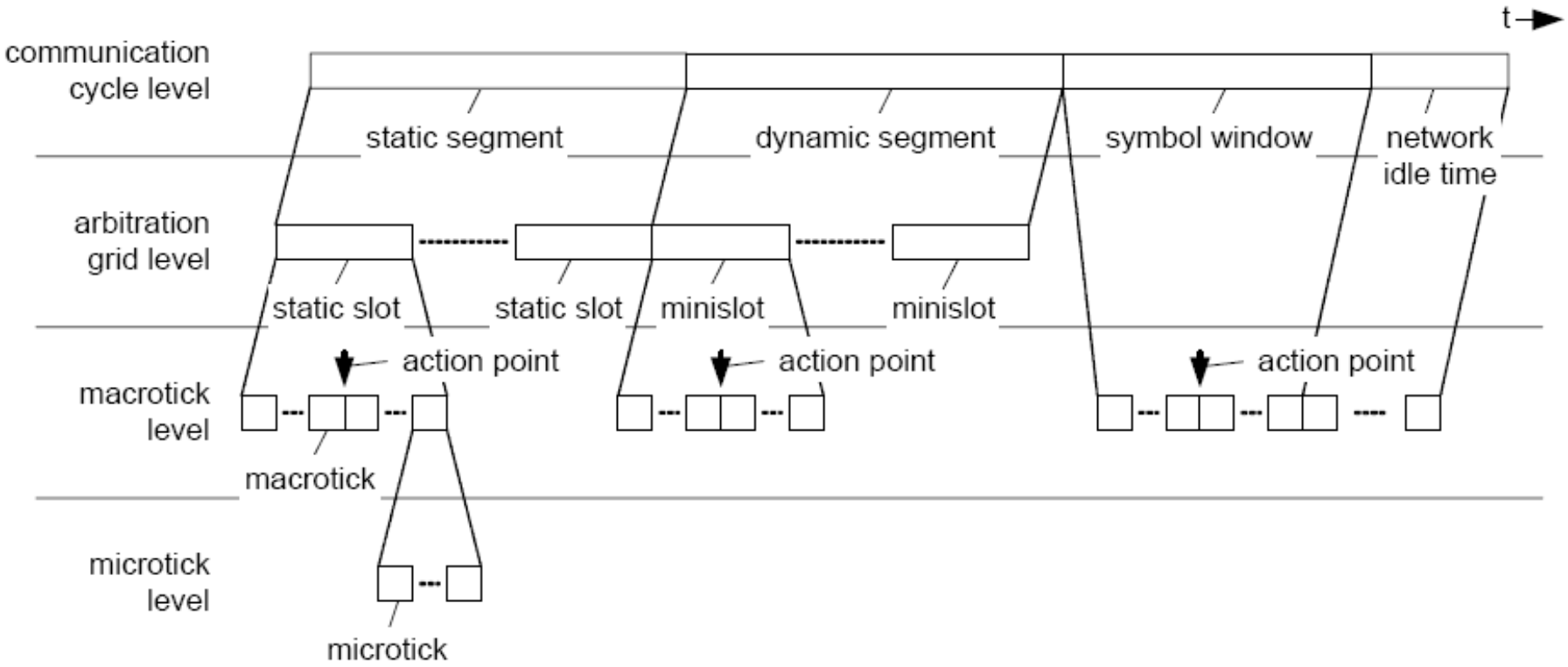


Used for clock adjustments derived from the synchronization, no communication in this segment.

Timing of Cycles



Timing Hierarchy

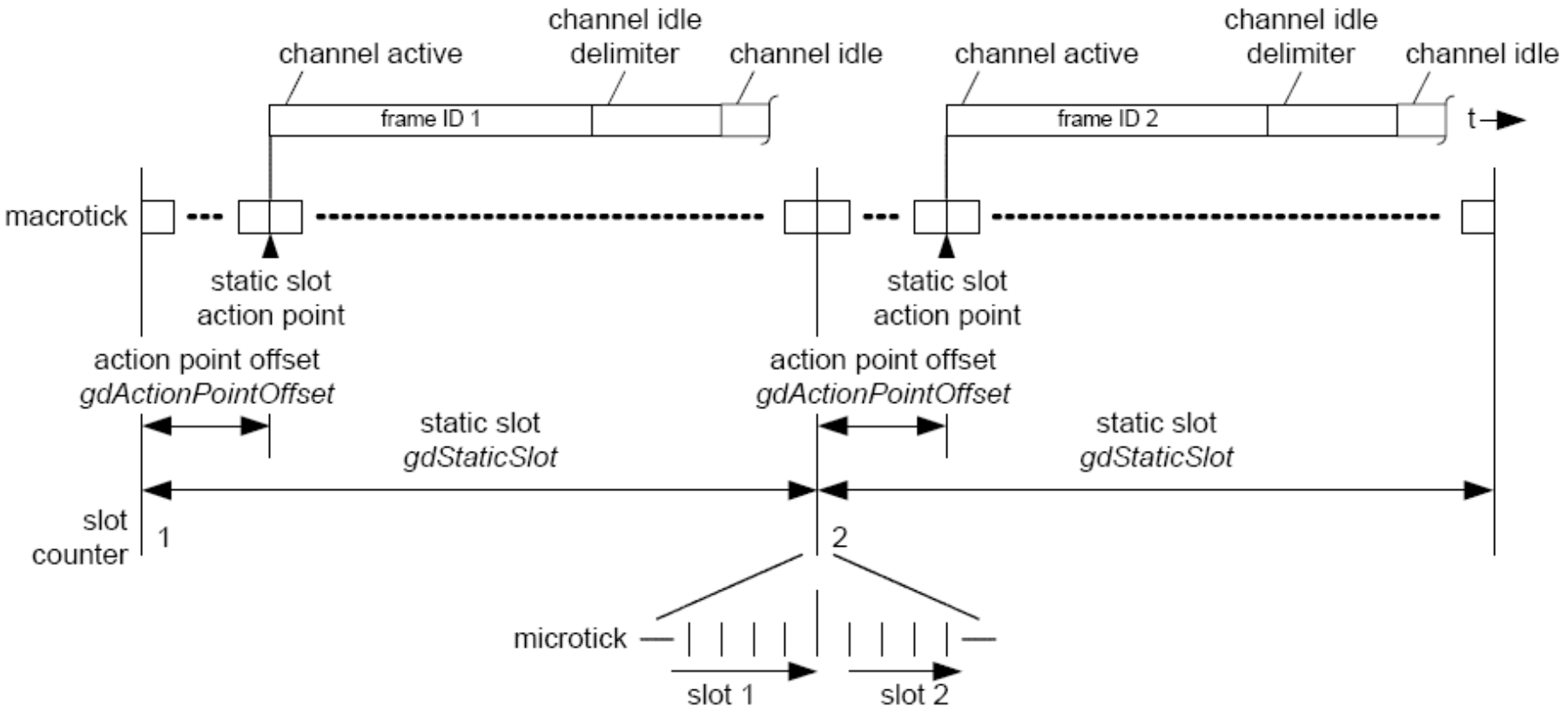


Media access control (MAC)

Slot usage:

- short idle time
- transfer of the frame
- long idle time

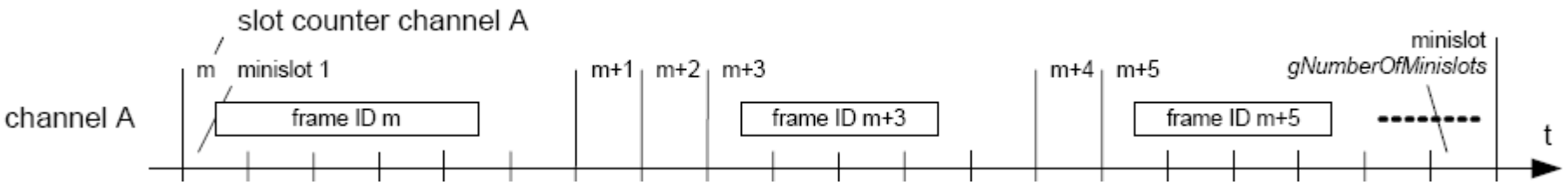
Static Segment



- fixed length slots
- each ECU sends in its own slot

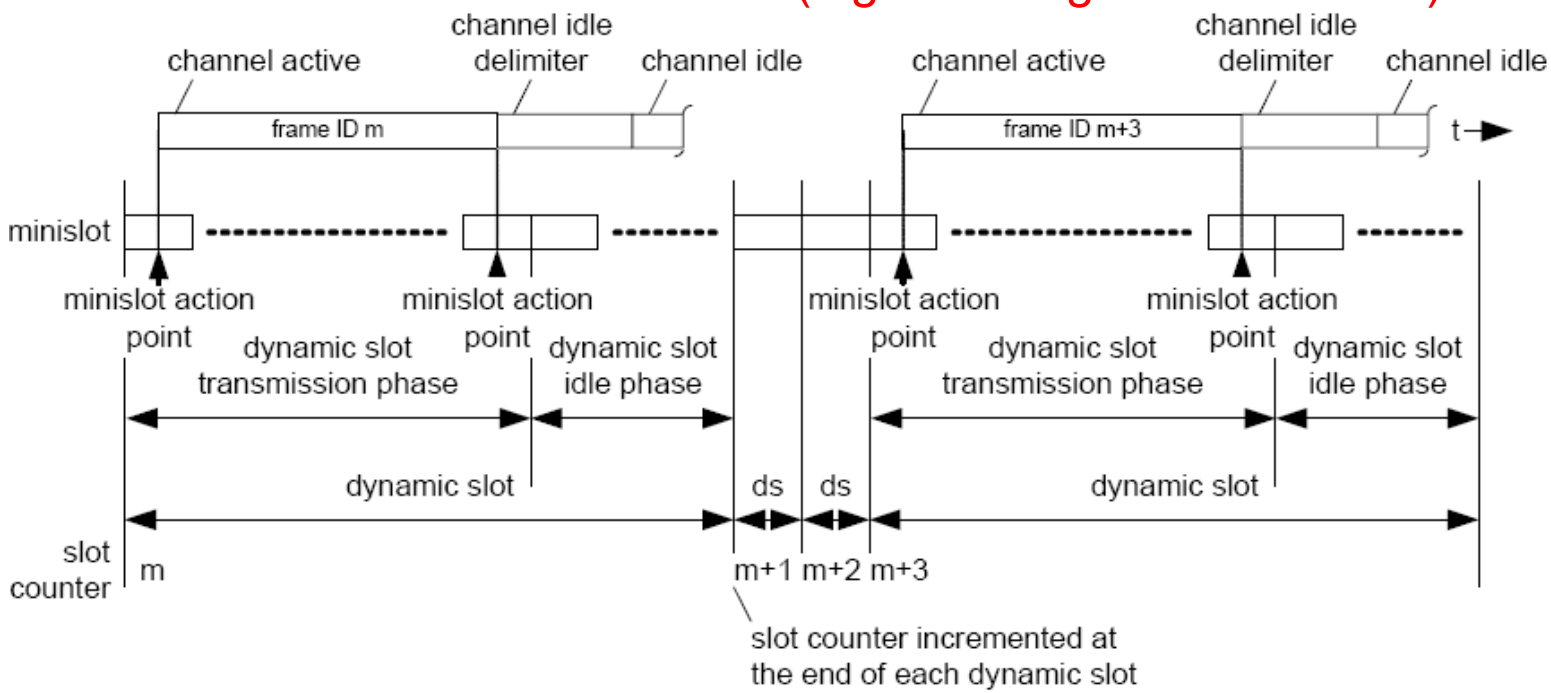
→ fixed communication rate, reliable but unflexible

Dynamic Segment

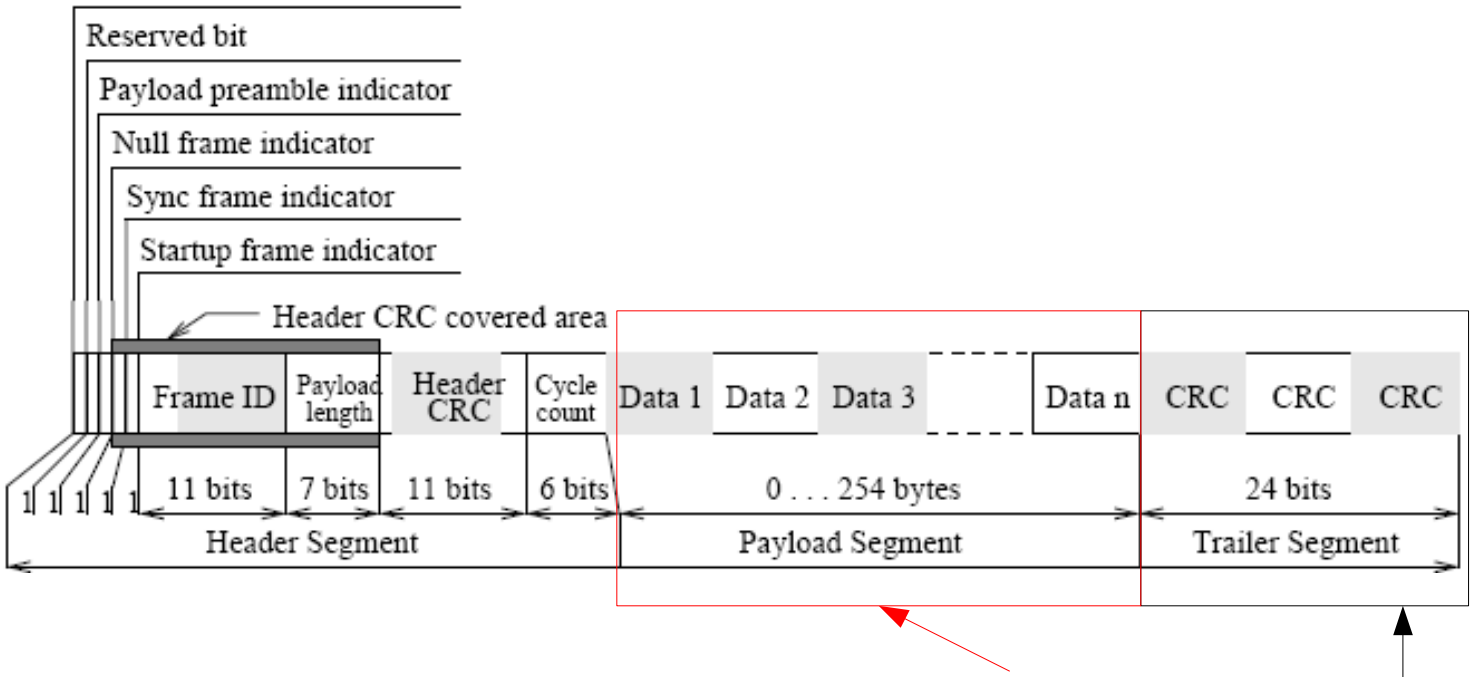


- small minislots as a basis
- dynamic slots, one for each ECU (longer if communication occurs)
- maximum number of minislots → ID is priority

→ more flexible but less reliable (high IDs might be excluded)

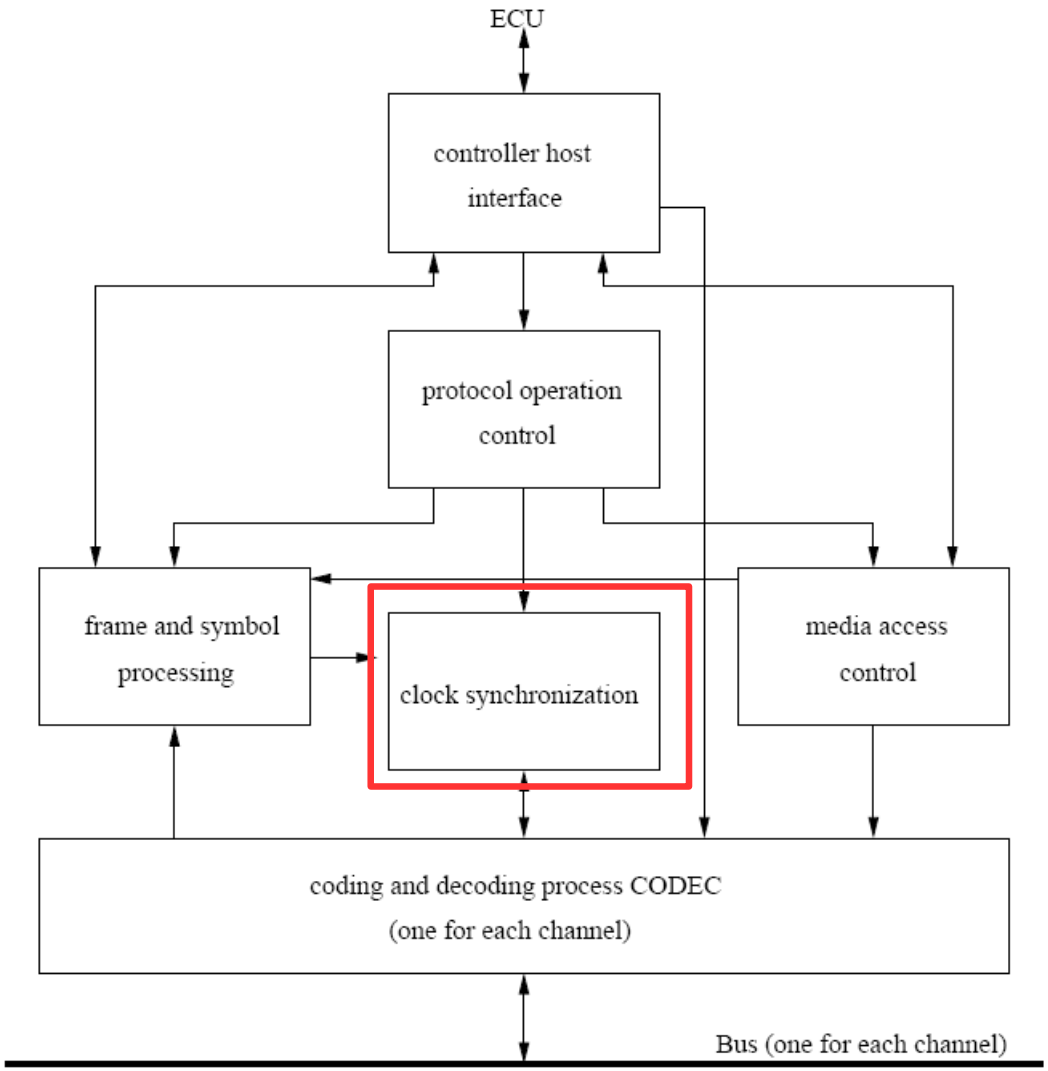


FlexRay Message Frame



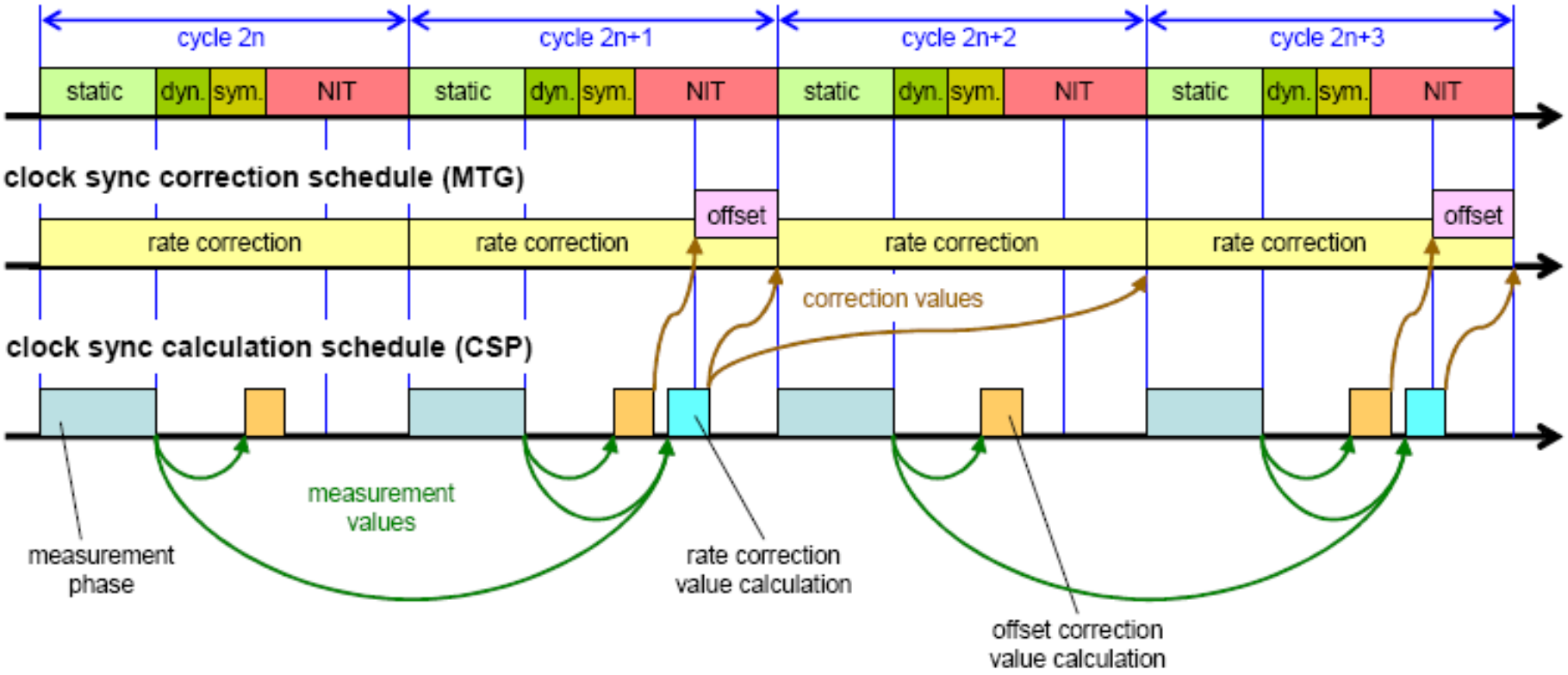
- Message in payload section: up to 254 bytes
- CRC used to notice transmission errors

Clock Synchronization



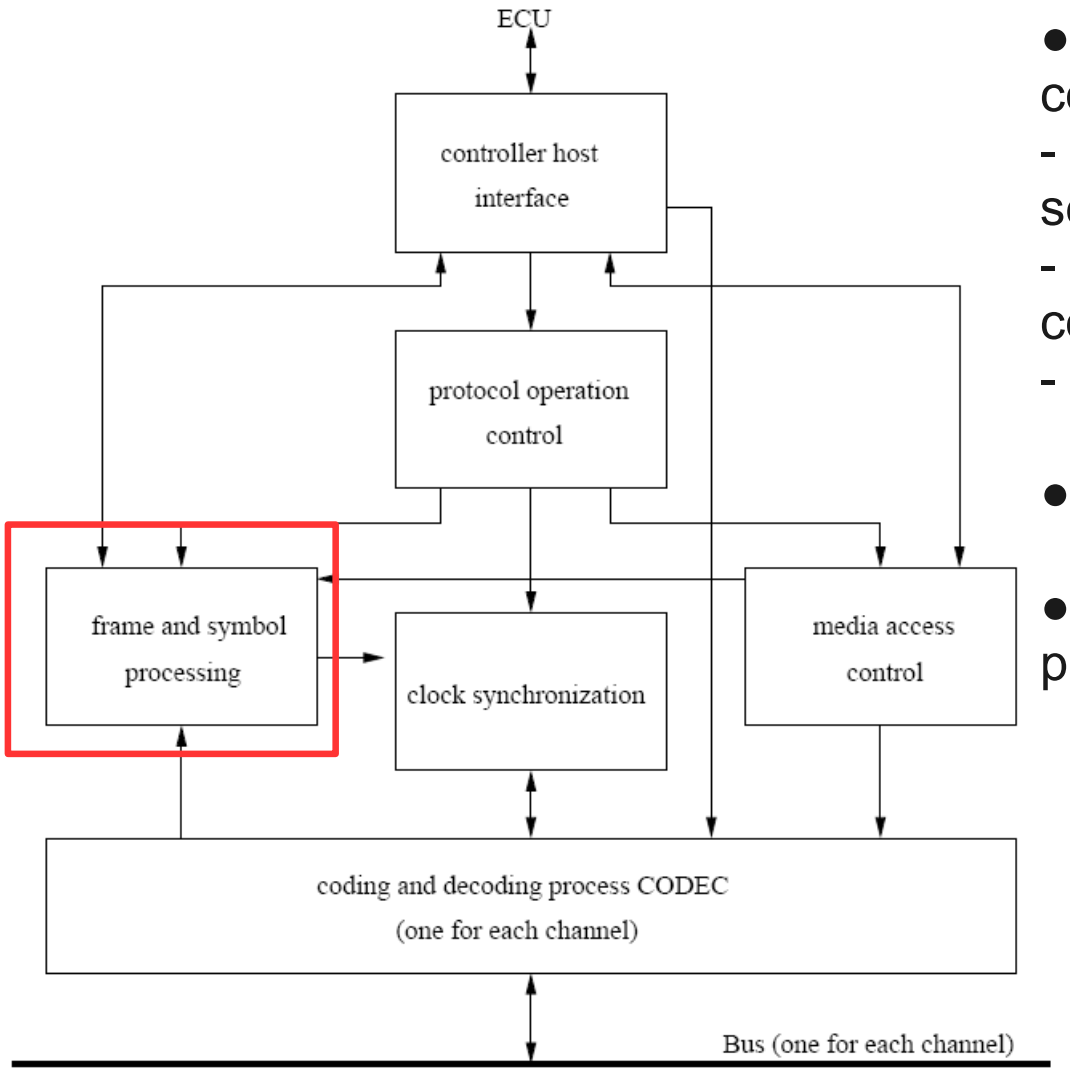
- tries to establish a shared view of time (to a certain extend)
- uses sync frames to get an impression of the time assumptions of other nodes
- adjusts the length of communication cycles

Clock Synchronization



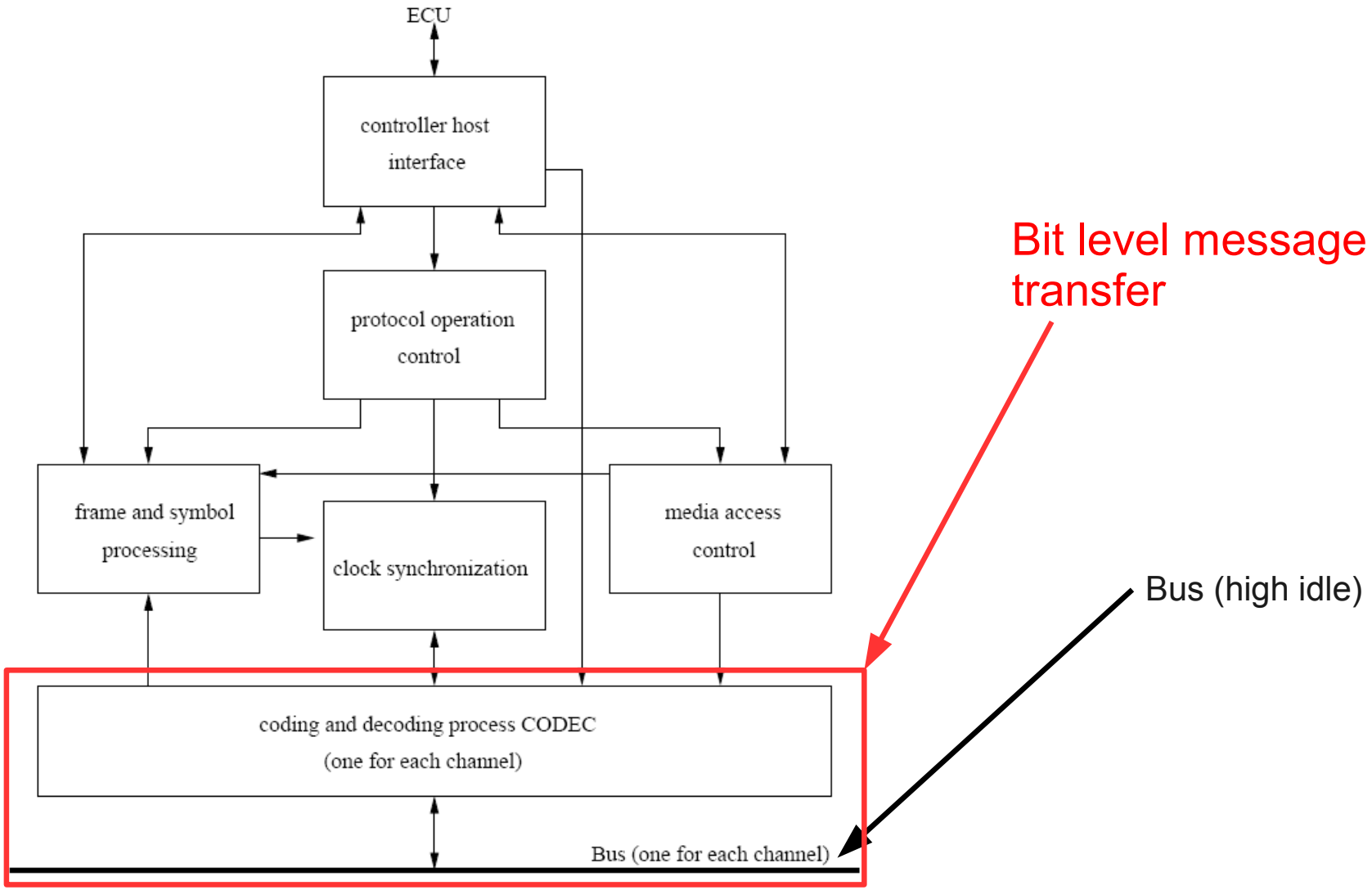
- rate correction adjusts the numbers of microticks per macrotick
- offset is an adjustment to the number of macroticks in the current cycle

Frame and Symbol Processing

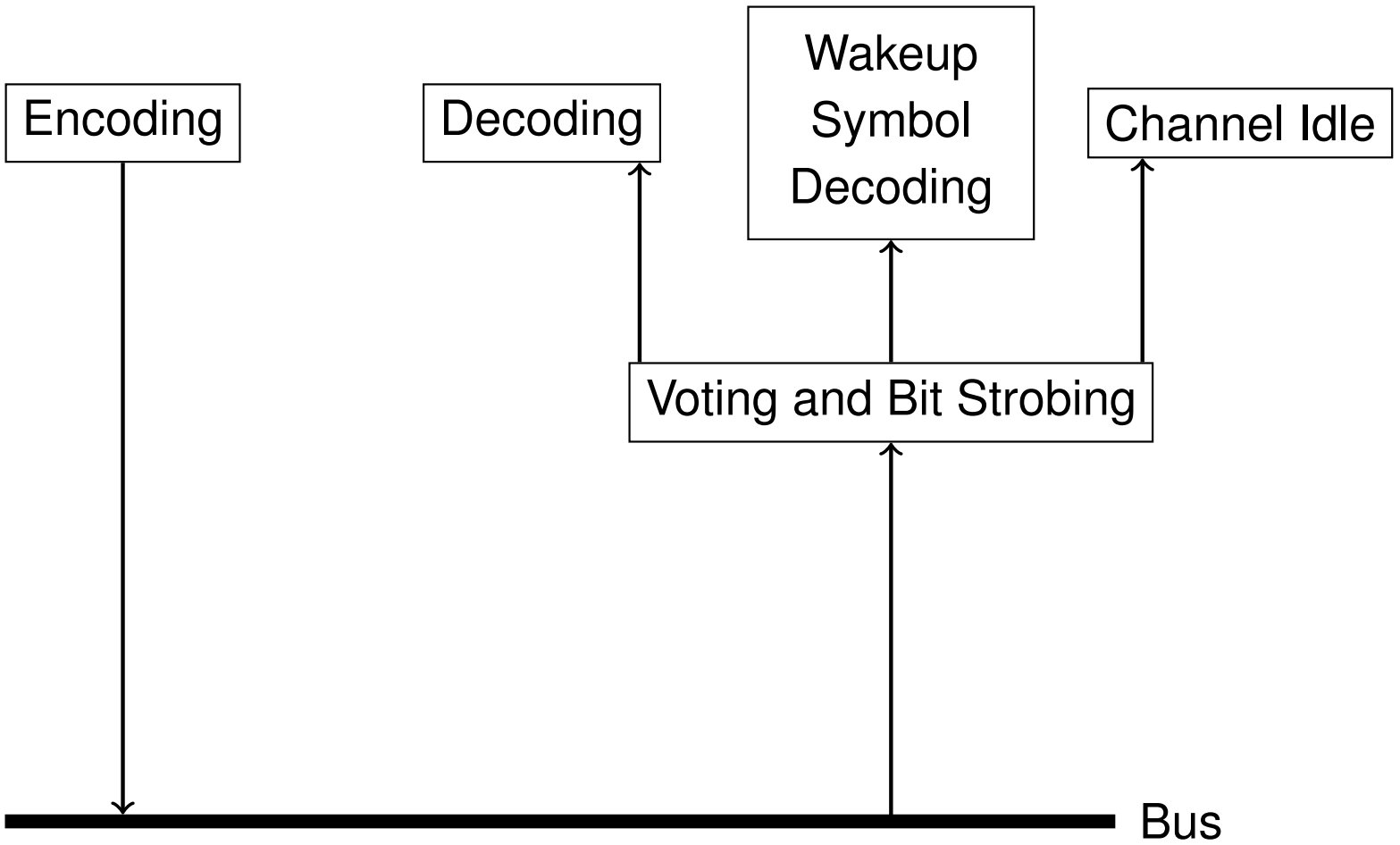


- controls integrity of communication:
  - Checks for appropriate segment
  - Checks length of communication elements
  - Checks for correct header
- Reports errors
- Dismantles frame and passes on the payload

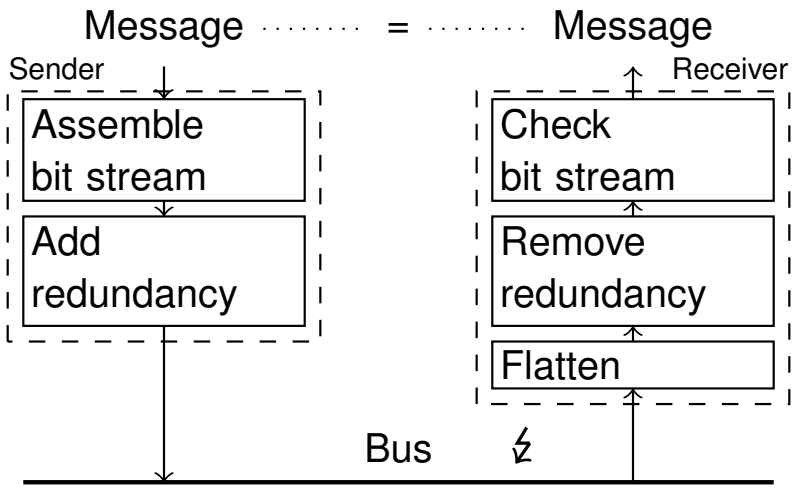
The FlexRay Controller



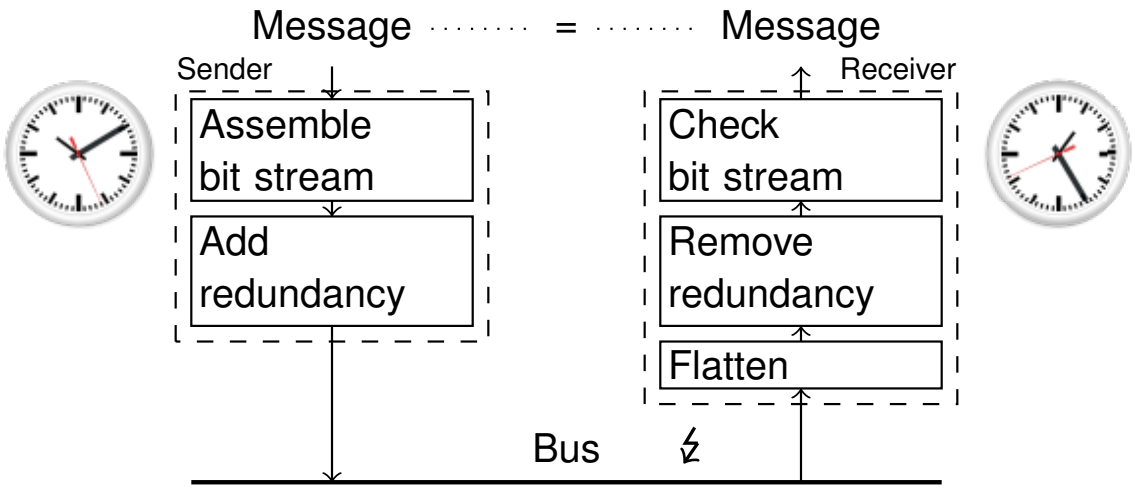
Coding and Decoding



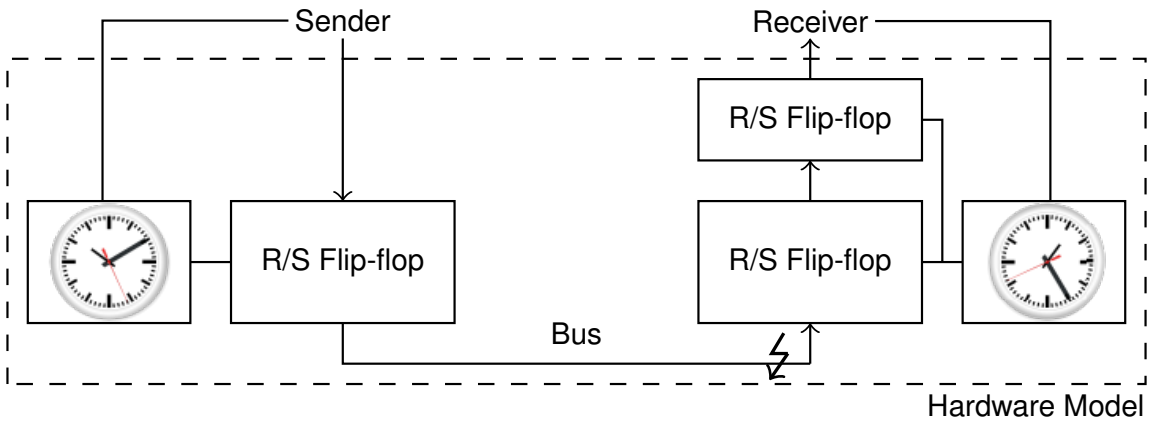
FlexRay Physical Layer Protocol



FlexRay Physical Layer Protocol



Error Sources in Communication Scenario



Sources of error:

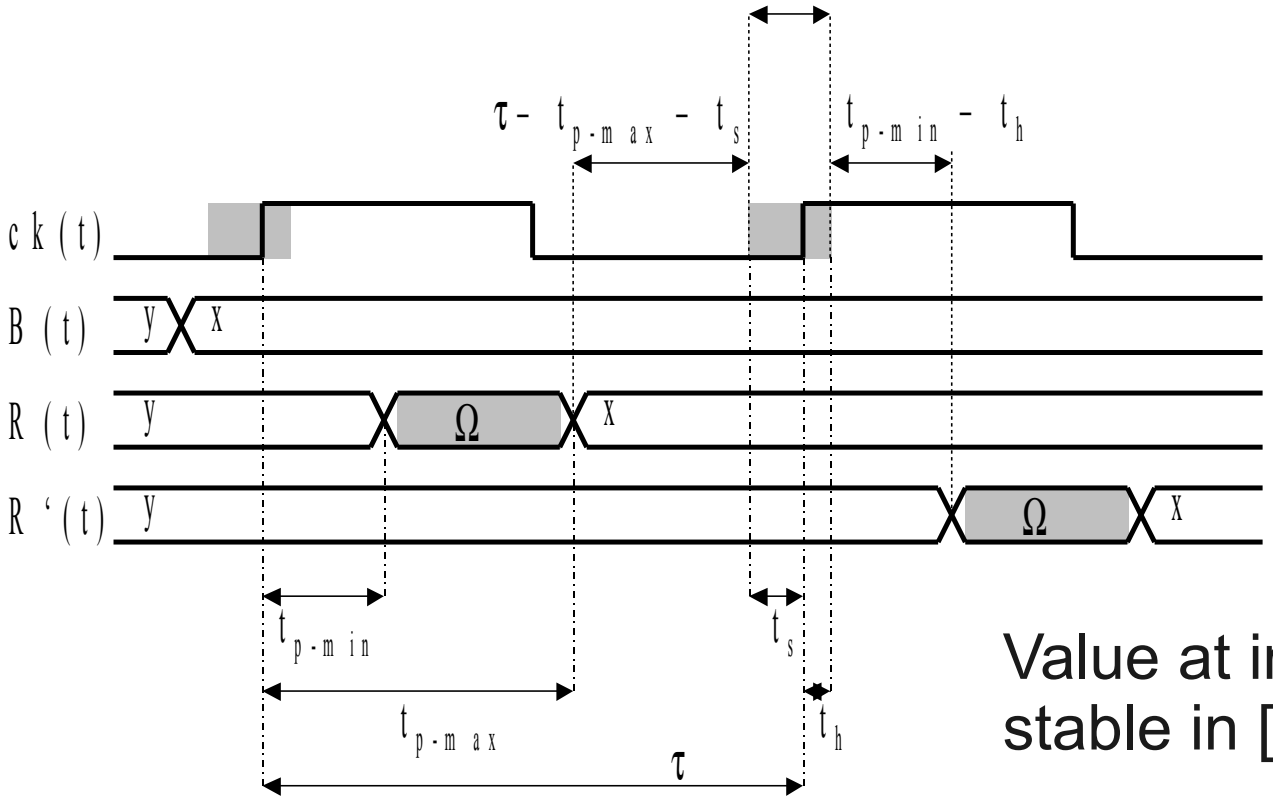
jitter:

- clock drift
- variance in delay
- unstable value in sampling interval (setup/hold)

glitches:

- bits flipped on the bus

Register Semantics



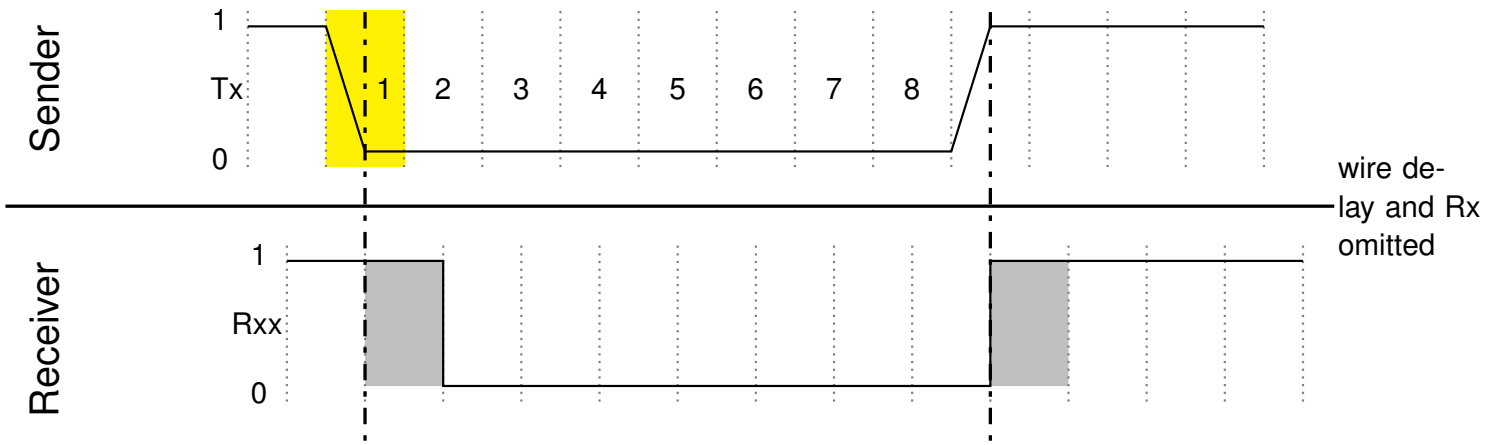
Value at input has to be stable in [ck-t_s,ck+t_h]

τ =cycle time

$t_{p\text{-}min/max}$ =delay of voltage change and signal propagation

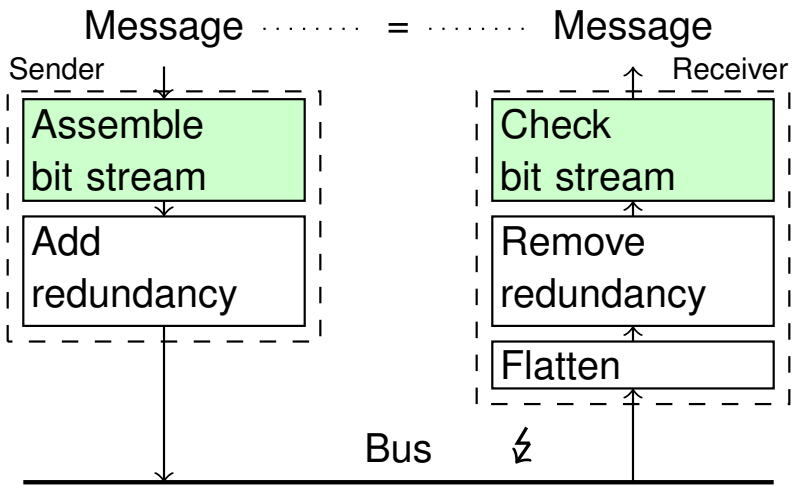
$t_{s/h}$ =hold and setup times of the register

Jitter Effects



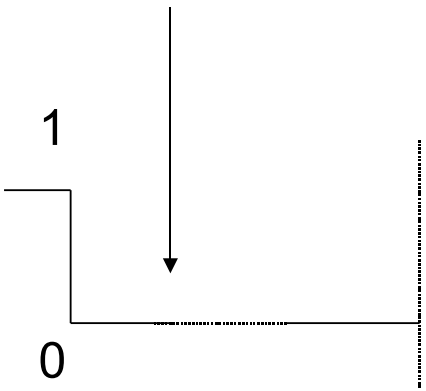
- clock drift $\rightarrow$ asynchronous communication
- signal changes are not instantaneous
- unstable value during setup / hold $\rightarrow$ error

Protocol Architecture



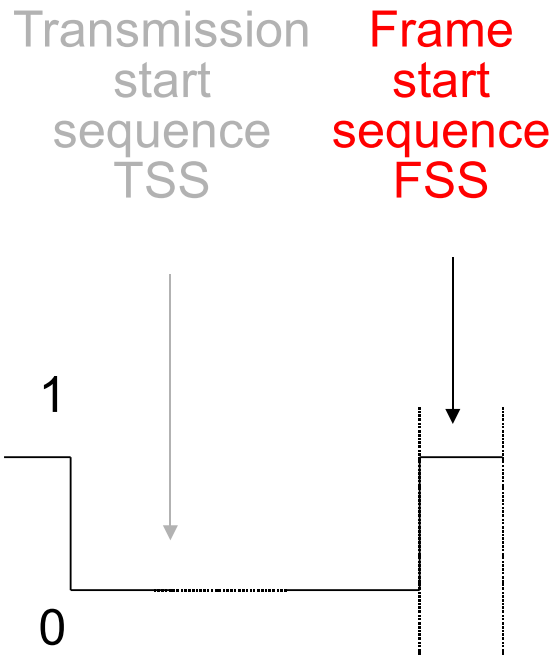
Frames

Transmission start sequence TSS



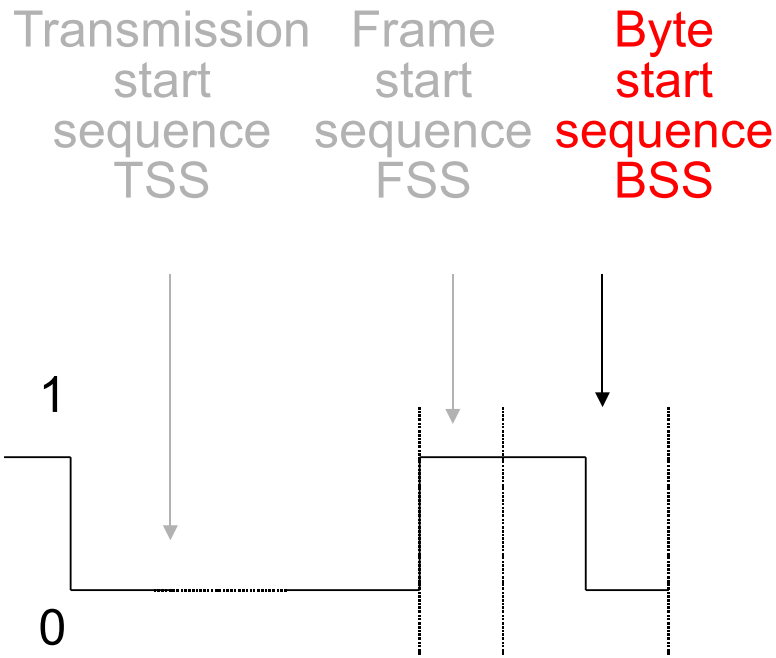
Frame coding in static segment

Frames



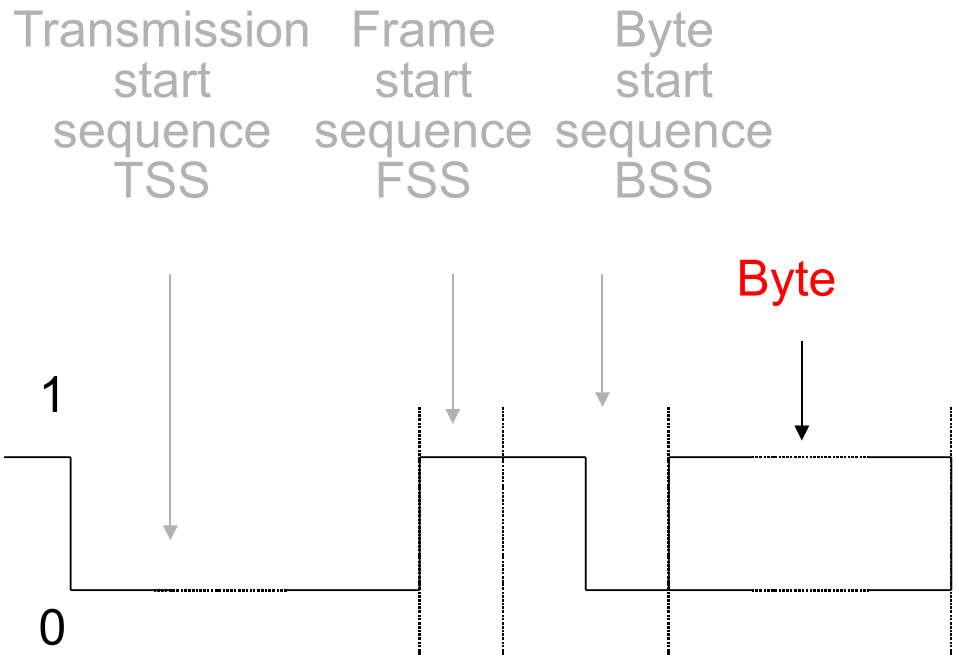
Frame coding in static segment

Frames



Frame coding in static segment

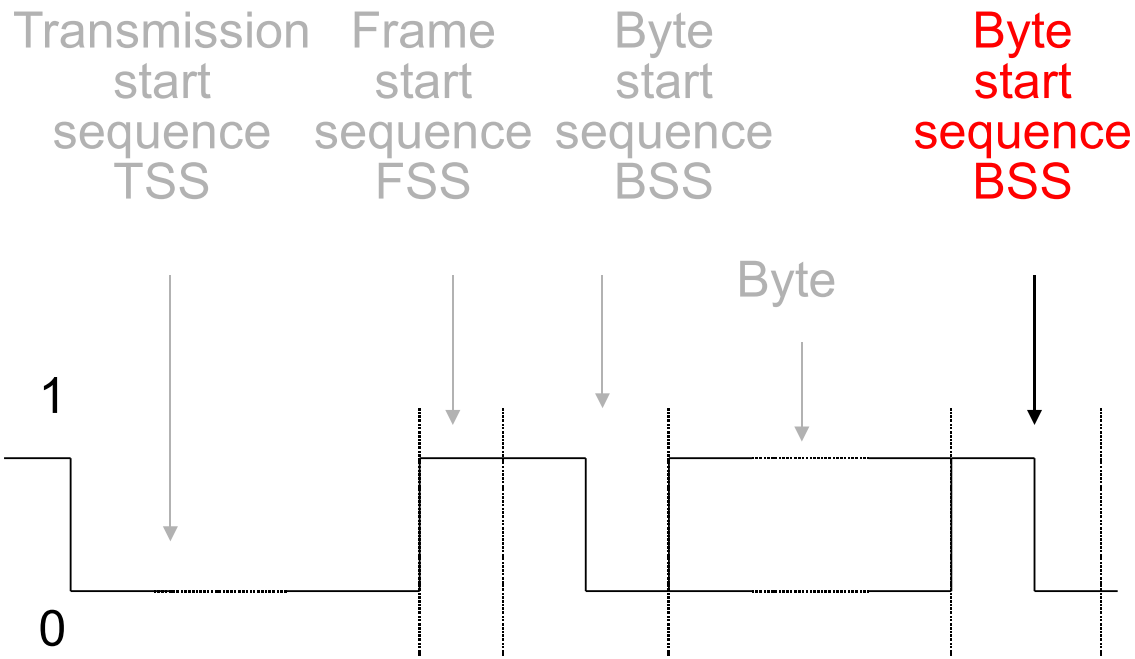
Frames



Frame coding in static segment

Encoding (ENC): Bit Stream Format

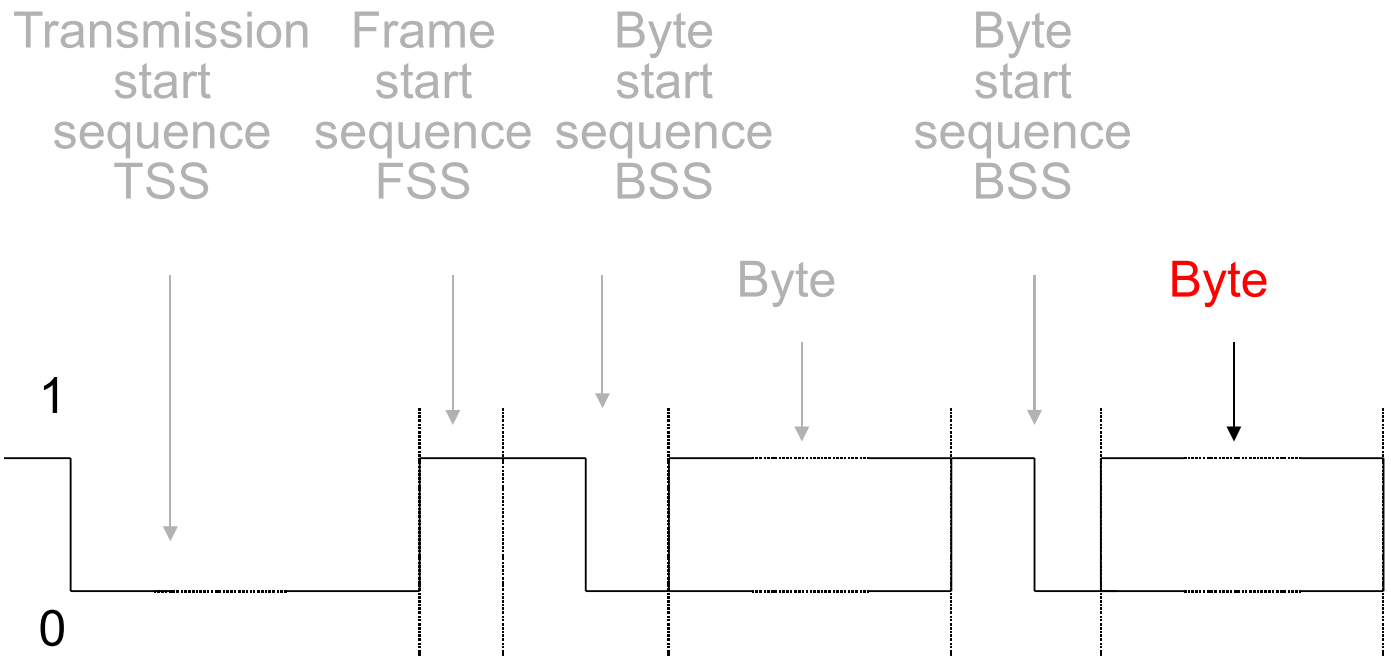
Frames



Frame coding in static segment

Encoding (ENC): Bit Stream Format

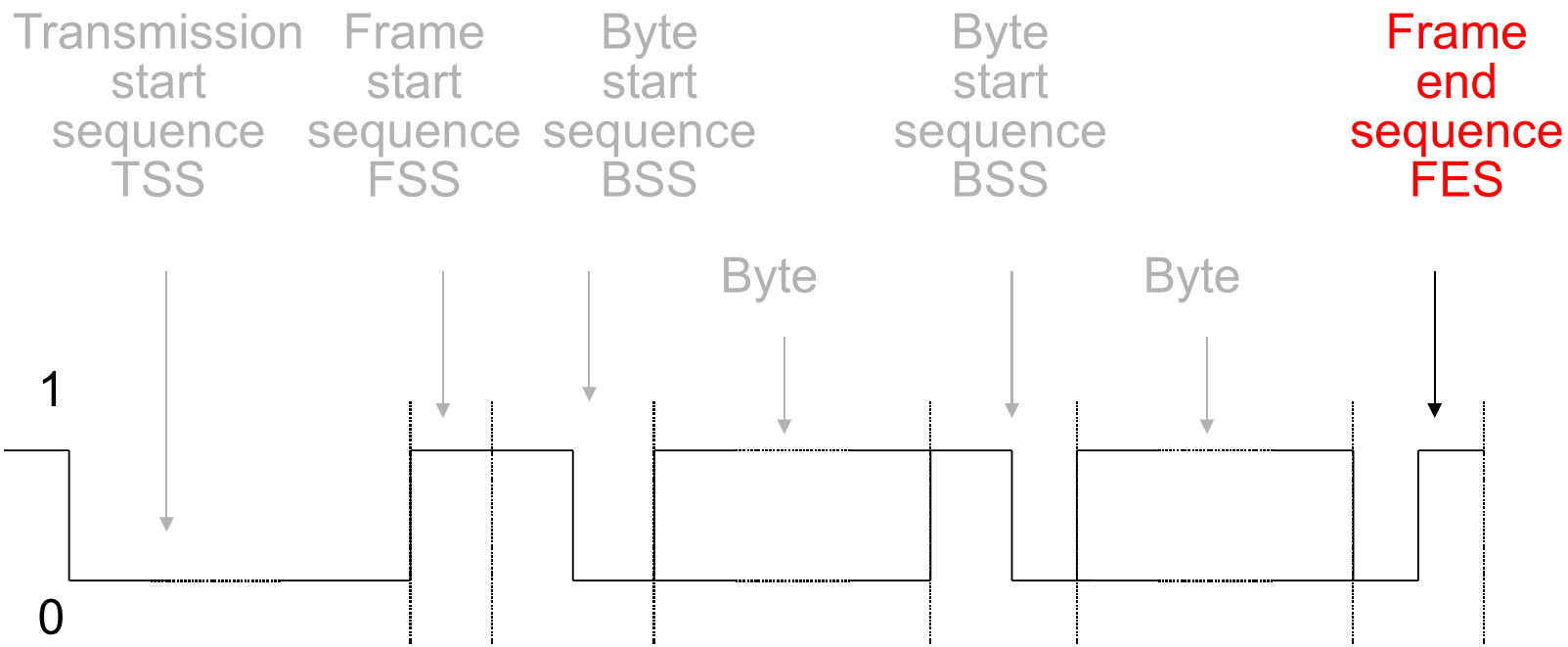
Frames



Frame coding in static segment

Encoding (ENC): Bit Stream Format

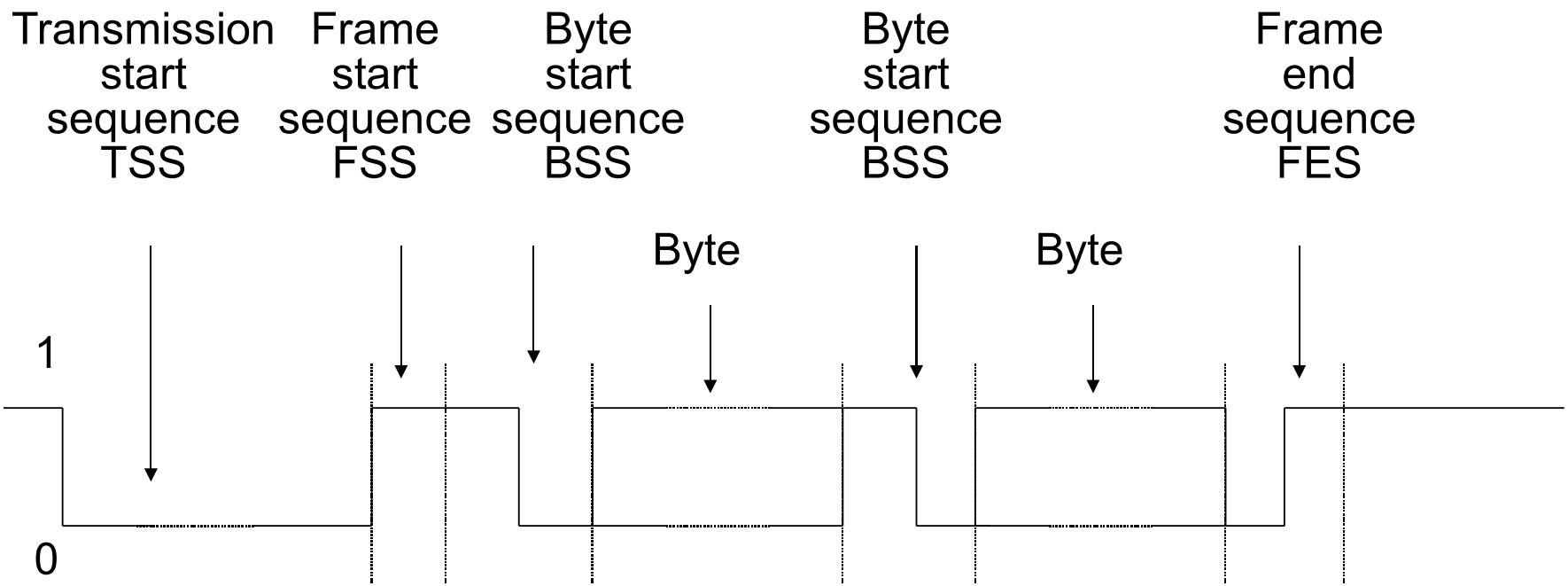
Frames



Frame coding in static segment

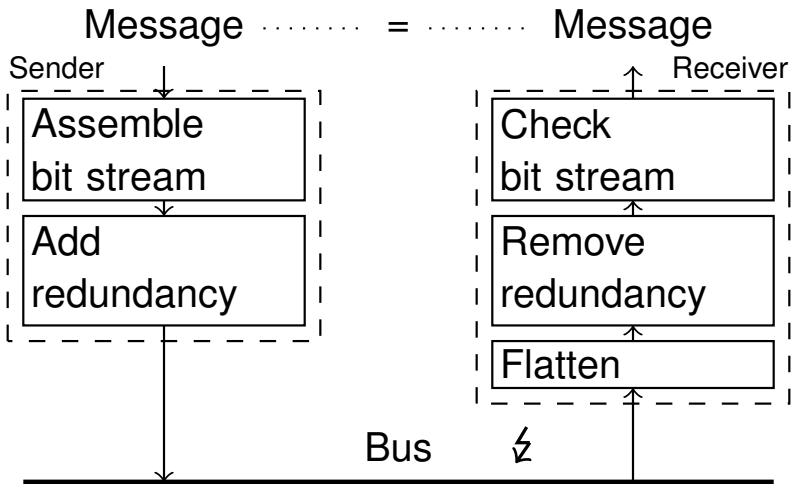
Encoding (ENC): Bit Stream Format

Frames



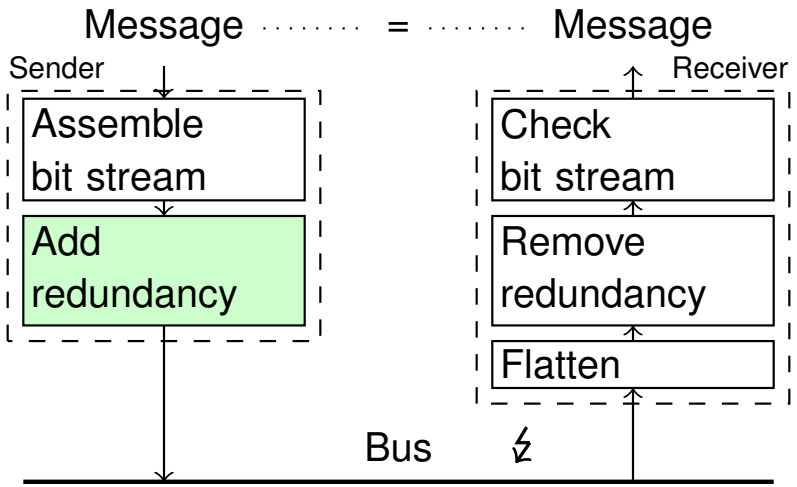
Frame coding in static segment

Protocol Architecture



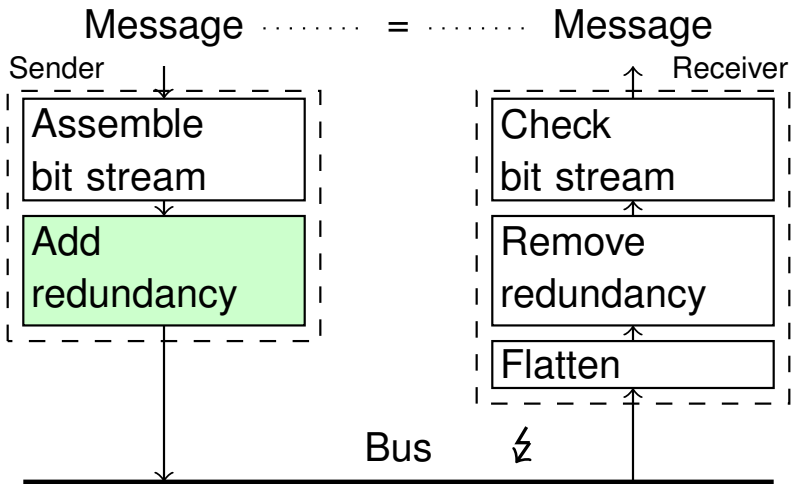
- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)
- Remove redundancy: pick 5^{th} voted value from each bit cell

Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)
- Remove redundancy: pick 5^{th} voted value from each bit cell

Protocol Architecture



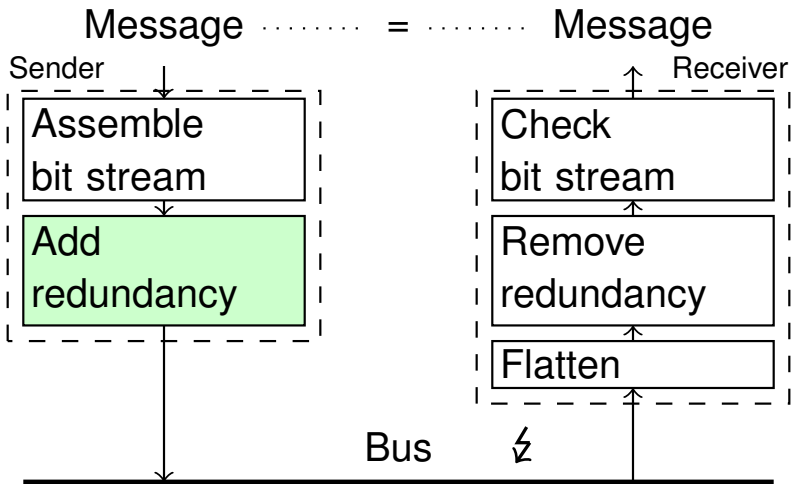
- Add redundancy: send each bit for 8 cycles (bit cell)

  Stream: 1 =Bus: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

- Flatten: take majority of last 5 samples (voting window size 5)

- Remove redundancy: pick 5^{th} voted value from each bit cell

Protocol Architecture



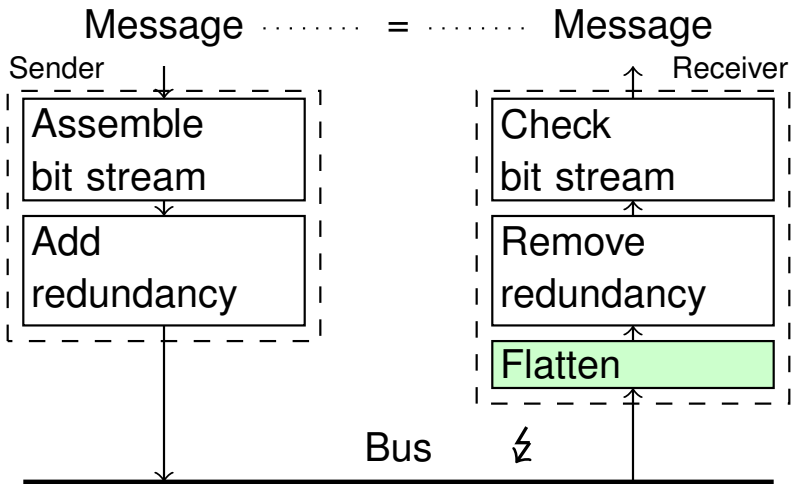
- Add redundancy: send each bit for 8 cycles (bit cell)

  Stream: 10 =Bus:

1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

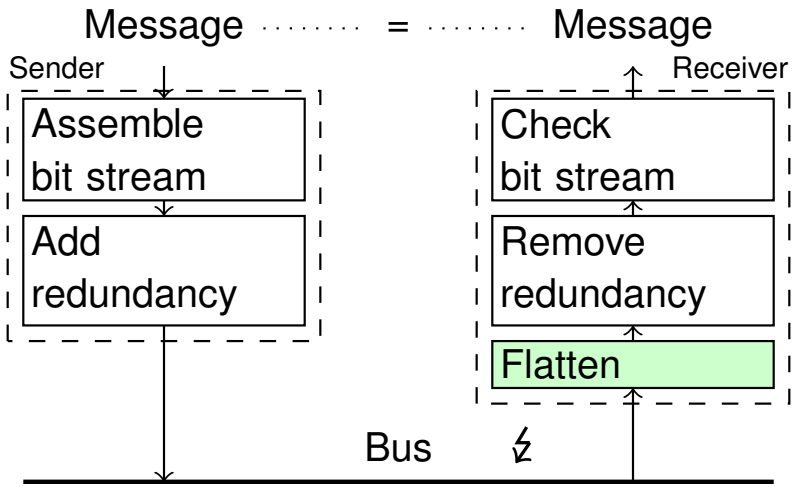
- Flatten: take majority of last 5 samples (voting window size 5)
- Remove redundancy: pick 5^{th} voted value from each bit cell

Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)
- Remove redundancy: pick 5^{th} voted value from each bit cell

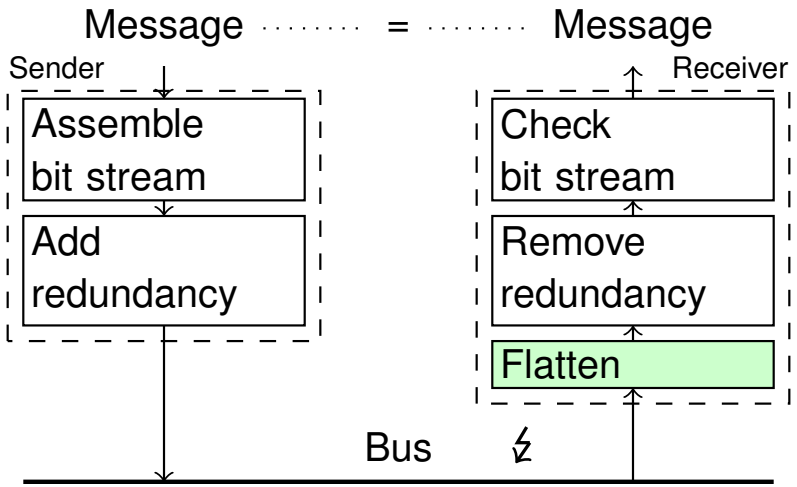
Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)

  E.g.: ... | 1 | 1 | 1 | 1 | 1 | $= 1$
- Remove redundancy: pick 5^{th} voted value from each bit cell

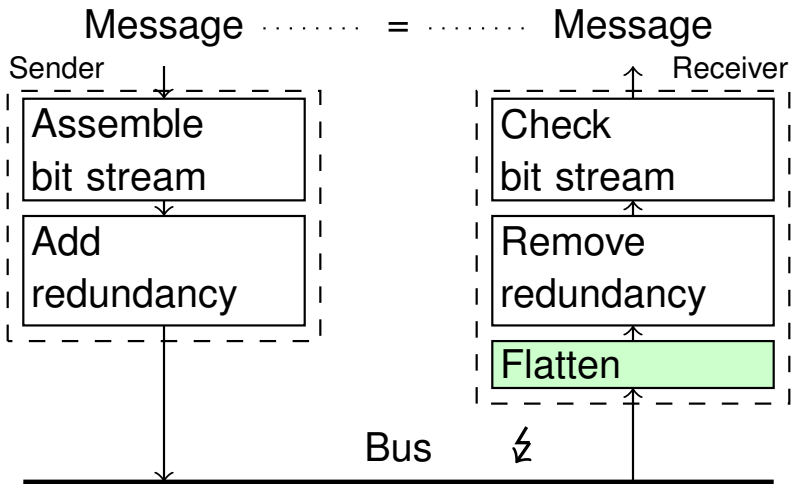
Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)

  E.g.: ... | 1 | 1 | 1 | 1 | 1 | 0 | $= 1$
- Remove redundancy: pick 5th voted value from each bit cell

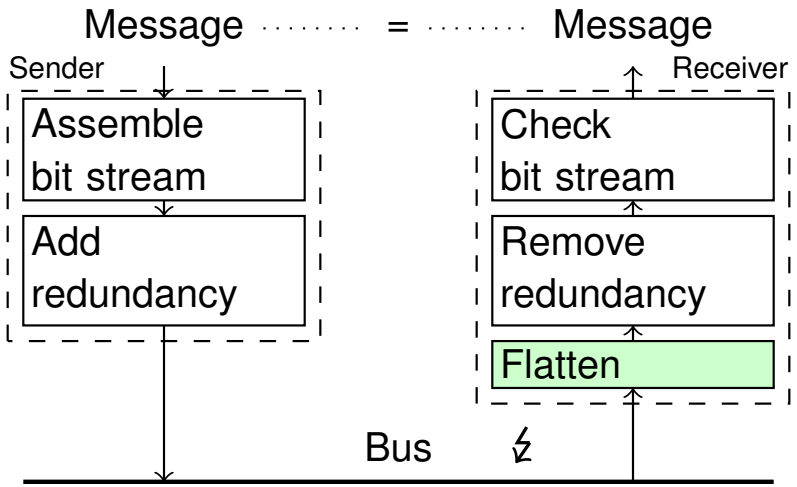
Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)

  E.g.: ... | 1 | 1 | 1 | 1 | 1 | 0 | 0 | $= 1$
- Remove redundancy: pick 5th voted value from each bit cell

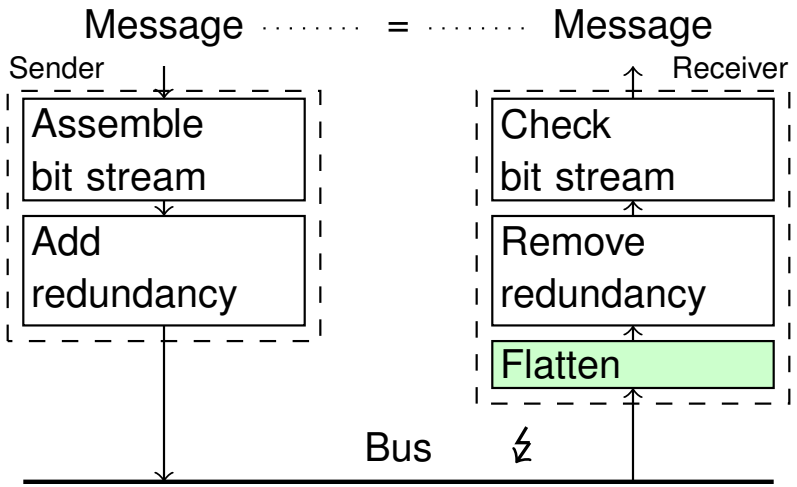
Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)

  E.g.: ... | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | $= 0$
- Remove redundancy: pick 5^{th} voted value from each bit cell

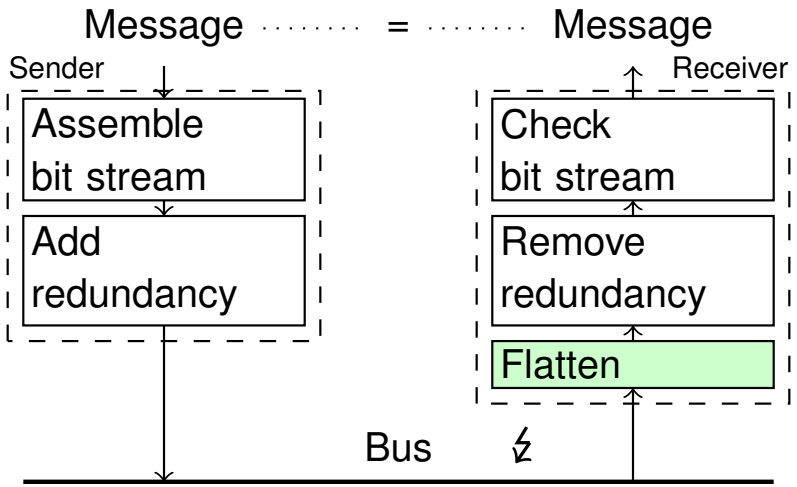
Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)

  E.g.: ... | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | $= 0$
- Remove redundancy: pick 5th voted value from each bit cell

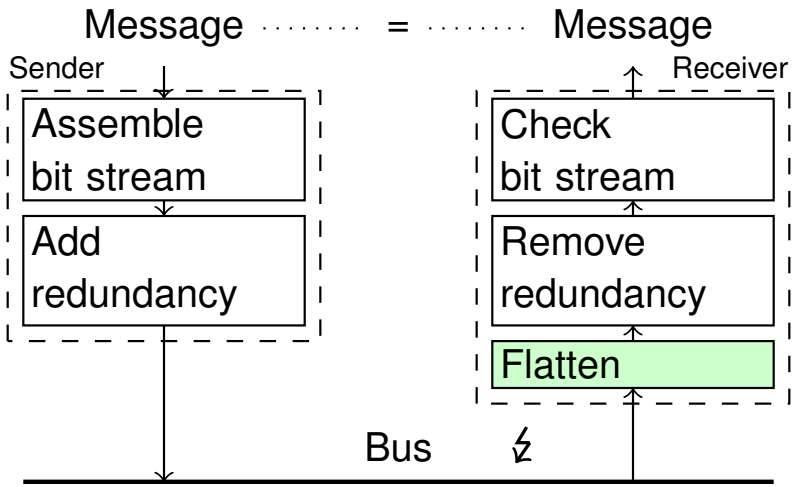
Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)

  E.g.: ... | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | $= 0$
- Remove redundancy: pick 5^{th} voted value from each bit cell

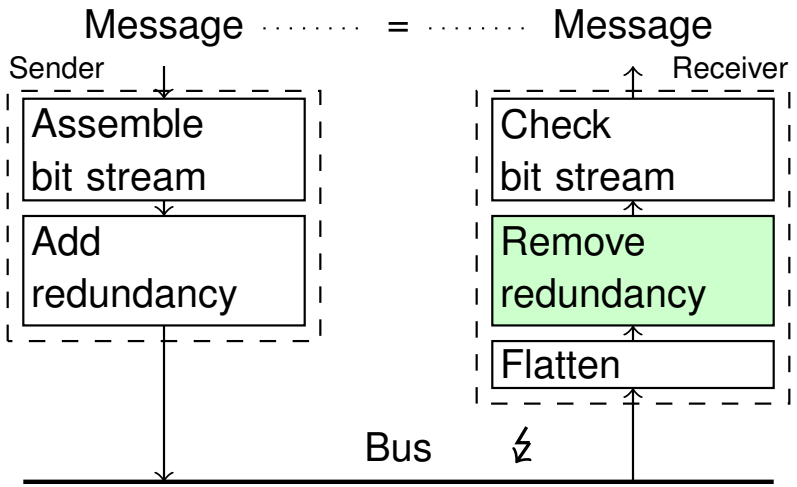
Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)

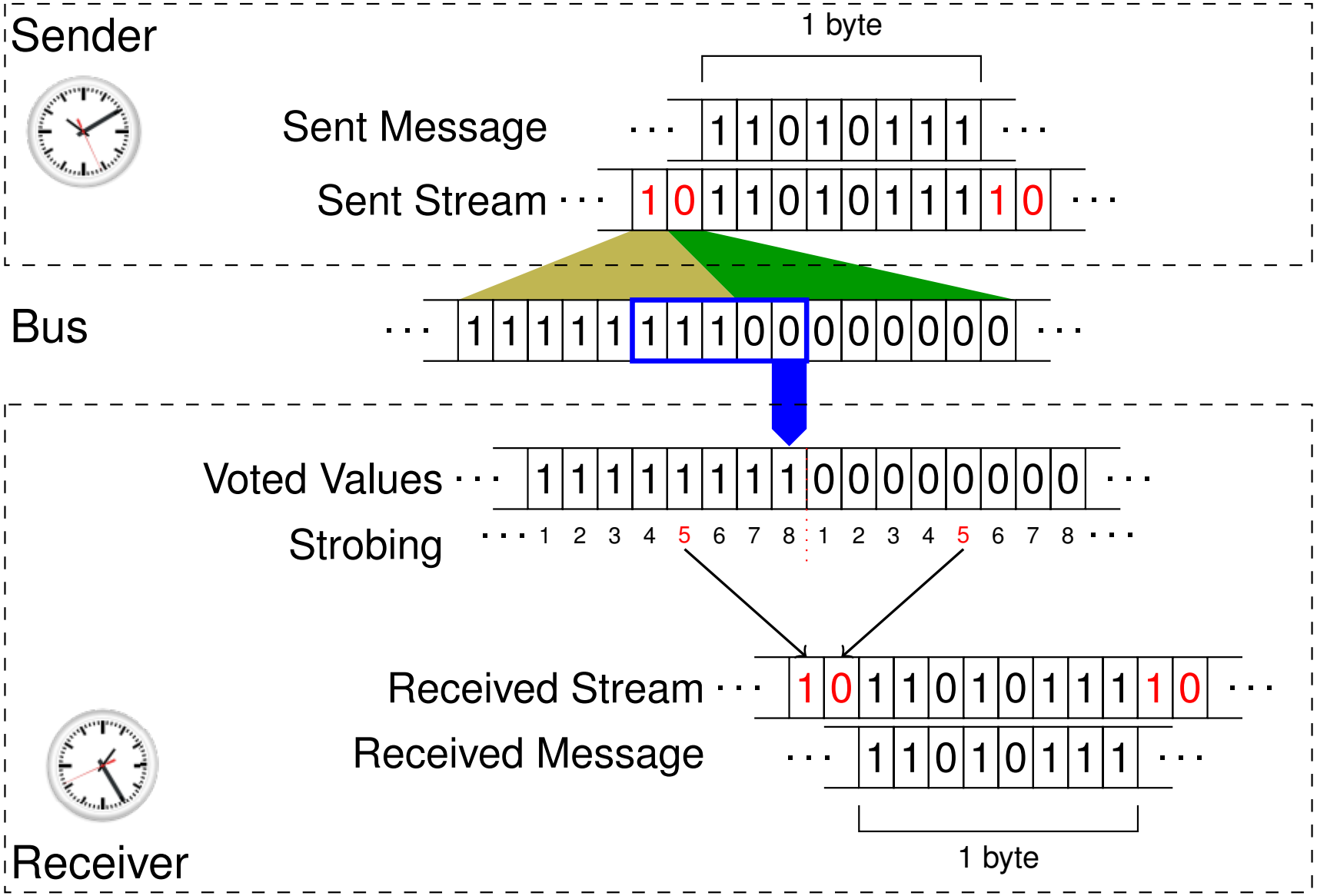
  E.g.: ... | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | $= 0$
- Remove redundancy: pick 5^{th} voted value from each bit cell

Protocol Architecture



- Add redundancy: send each bit for 8 cycles (bit cell)
- Flatten: take majority of last 5 samples (voting window size 5)
- Remove redundancy: pick 5^{th} voted value from each bit cell

Example: FlexRay Physical Layer Protocol



Encoding (ENC)

Symbols

Collision avoidance and media access test symbol (CAS/MTS): $TSS+0^{30}$

Wakeup symbol (WUS): $0^{15\text{-}60}+idle^{45\text{-}180}$

sent in a wakeup pattern (WUP): $WUS^{2\text{-}63}$

Wakeup Symbol Decoding (WUSDEC)

- Waits for a WUP like pattern:

  $0^x+1^y+0^x$

  produced by two consecutive WUS

- Reports to protocol operation control(POC)

Channel Idle (IDET)

Whenever 1^{11} is received,
a channel idle recognition point (CHIRP)
is reported.

Decoding (DEC)

- Handles reception of CAS/MTS and frames
- Checks for errors

Frames:

- reassembles the message
- checks format
- checks CRC (frame “fingerprint”)

CAS/MTS:

- checks length of LOW signal

Glitch Position Counts

Position of glitch more important than number of glitches

- easy to destroy message using just 3 meanly placed glitches
- message survives hundreds of nicely placed glitches
- environment independent of protocol $\rightarrow$ glitches independent of protocol
- *pattern*: relative position of glitches to each other
- pattern of glitches matters

Glitch Position Counts

Position of glitch more important than number of glitches

- easy to destroy message using just 3 meanly placed glitches
- message survives hundreds of nicely placed glitches
- environment independent of protocol $\rightarrow$ glitches independent of protocol
- *pattern*: relative position of glitches to each other
- pattern of glitches matters

Guaranteed Glitch Tolerance

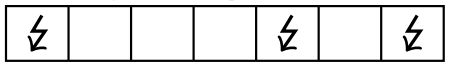
The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Guaranteed Glitch Tolerance

The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
  E.g.: ...  ...
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Guaranteed Glitch Tolerance

The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
  E.g.: ...  ...
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Guaranteed Glitch Tolerance

The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
  E.g.: ...  ...
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Guaranteed Glitch Tolerance

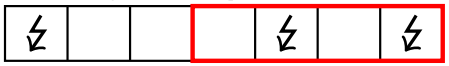
The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
  E.g.: ...  ...
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Guaranteed Glitch Tolerance

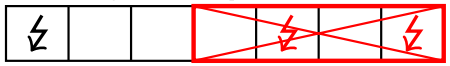
The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
  E.g.: ...  ...
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Guaranteed Glitch Tolerance

The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
  E.g.: ...  ...
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Guaranteed Glitch Tolerance

The protocol tolerates

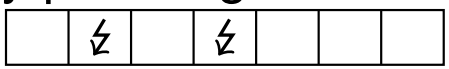
- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)
- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)

Note: one message $\approx$ 21.000 samples

Guaranteed Glitch Tolerance

The protocol tolerates

- 1 glitch in every sequence of 4 consecutive samples (1 out of 4)

- 2 arbitrarily placed glitches in 88 consecutive samples (at most 2)
  E.g.: ...  ...

Note: one message $\approx$ 21.000 samples

The FlexRay Communication Protocol

- specified for a dependable automotive network

Basic characteristics:

- synchronous and asynchronous frame transfer
- guaranteed frame latency during synchronous transfer
- prioritization of frames during asynchronous transfer
- multi-master clock synchronization
- error detection and signaling
- error correction on the physical layer