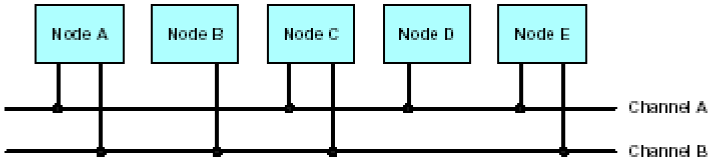
Overview of FlexRay scheduling issues

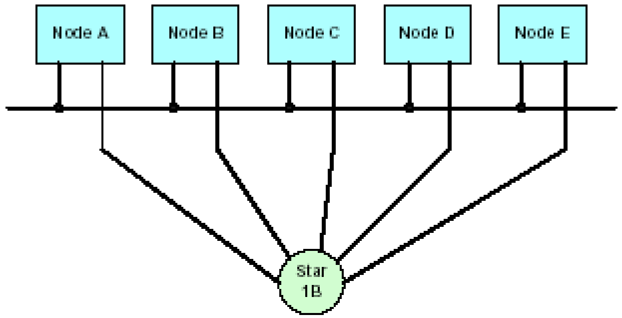
Marco Di Natale, Wei Zheng

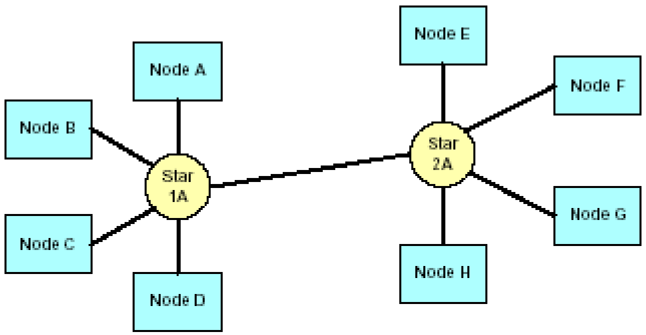
Credits: Tom Forest –slides concepts & suggestions

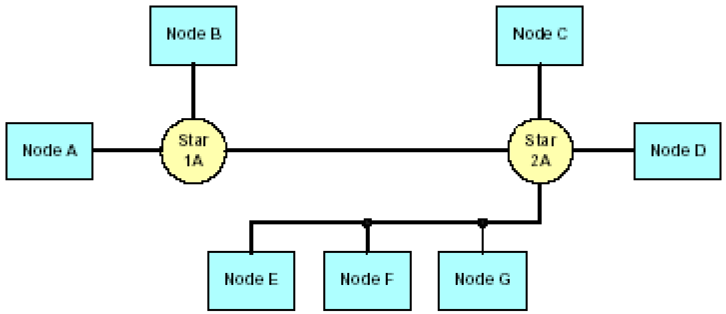
FlexRay: Network topology

- FlexRay supports different interconnection topologies
  - single-channel system
  - dual-channel system
  - dual-channel system with mixed connectivity, where some nodes are connected to both channels while other nodes are connected to only one



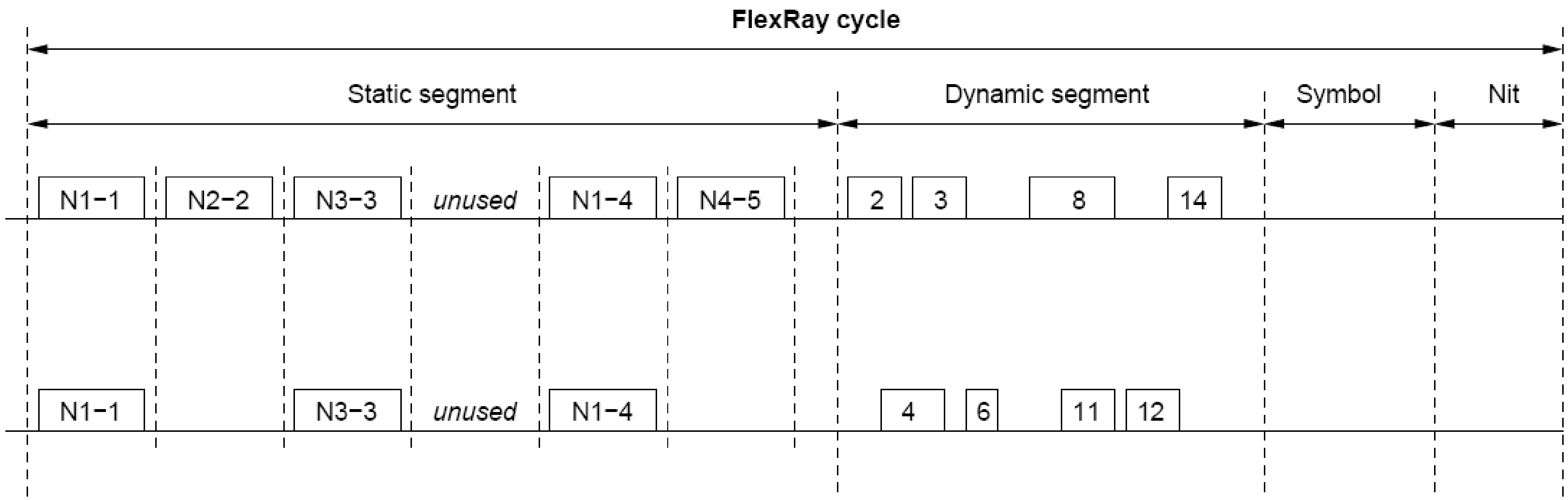






FlexRay: Schedule

- Communication on the bus is arranged according to a cyclic forever-repeating structure, with four segments
  - Static: transmission time is allocated statically. Composed of fixed-length slots
  - Dynamic (optional): time is allocated dynamically, transmission times may vary
  - Symbol Window (optional)
  - Network Idle Time
- In dual channel systems, cycles and segments start at the same time on both channels, but the schedule may be different



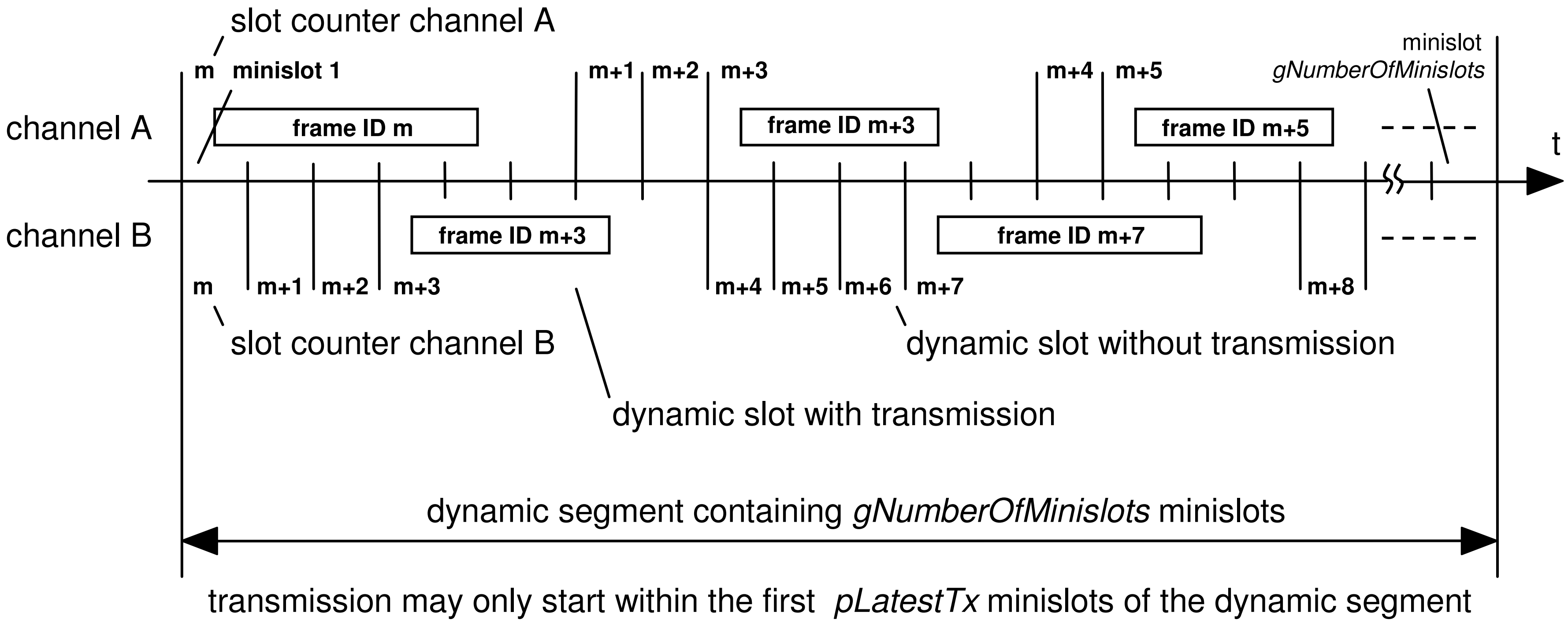
Static Segment Access

- All static slots/frames have the same duration
  - In dual channel systems, the static slots have the same size (and are therefore synchronized).
- TDMA access
  - Each static slot is assigned to one node
  - If a node owns the current slot
    - If a frame is ready the node transmits the frame in the slot
    - If no frame is ready, or a frame is not scheduled to transmit in the given, node sends a special null frame (*a frame -regular or null- is always sent*)

Dynamic Segment Access

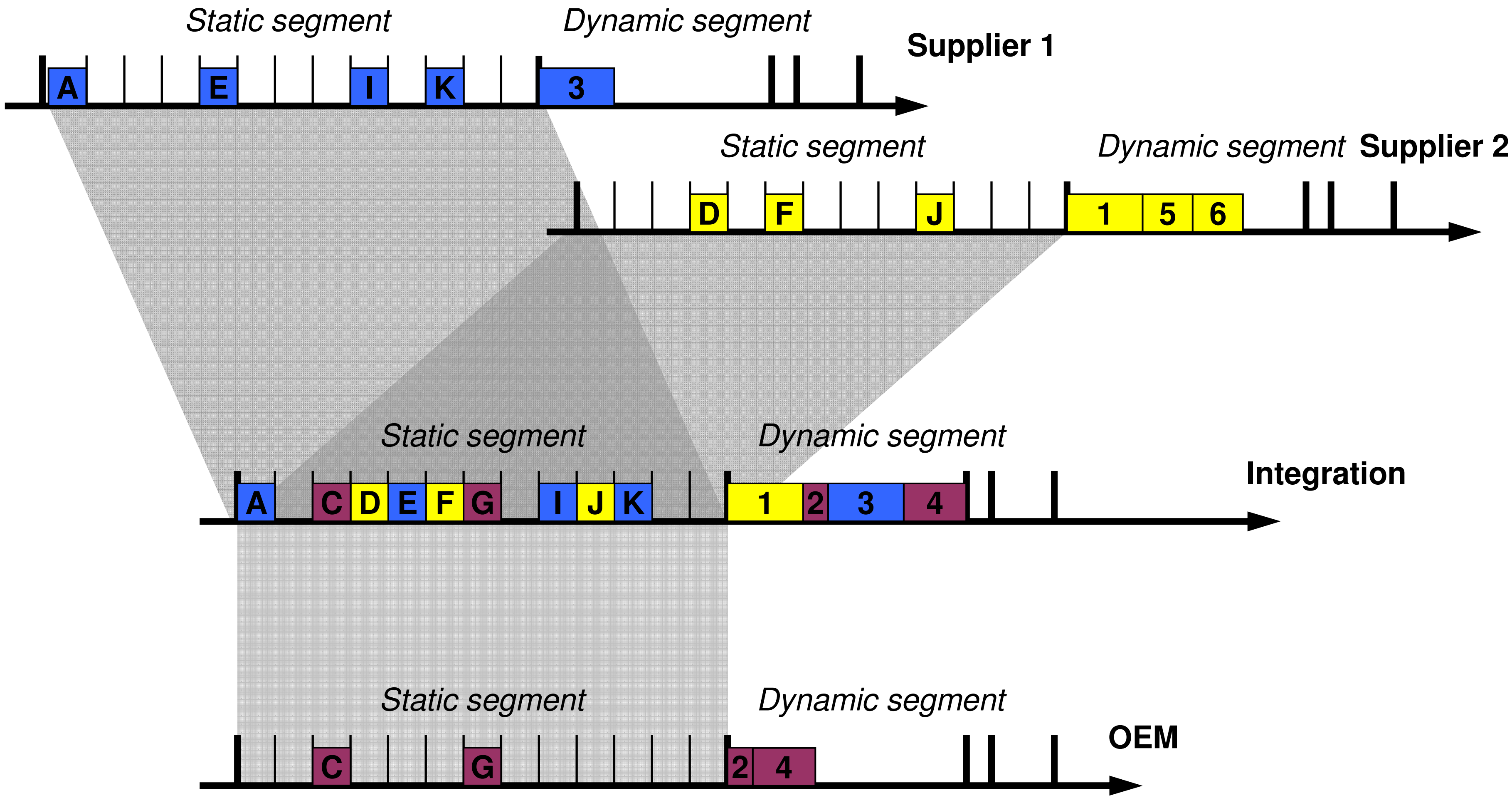
- Network transmission rights are assigned using minislotting (virtual token)
  - The time is divided in minislots
  - Each minislot is assigned an index, starting from 1
  - Outgoing dynamic messages (of variable size) are associated with an index
  - If no message matches the minislot index, there is no transmission and after the duration of the minislot the index is incremented
  - If there is a message matching the minislot index, the message is transmitted. All the minislots occurring during the message transmission retain the same index. The slot is incremented only after the message ends transmission and one minislot goes by without activity
  - In dual channel systems, minislots are aligned, but message transmissions are not
  - The dynamic segment ends after the last minislot, with a safety margin (pLatestTx) to ensure that dynamic message transmissions do not overlap with the following cycle
    - Not all messages may have been transmitted

Dynamic Segment Example



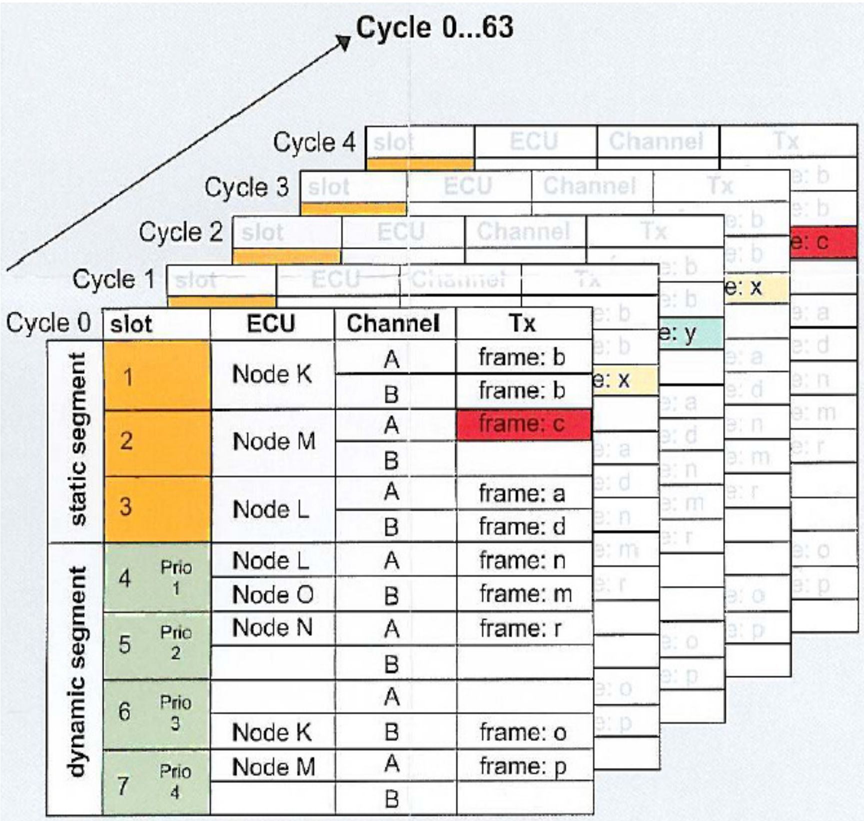
Scheduling and composition

- Composing static and dynamic segment schedules



Application examples - schedule

- **<u>Cycle multiplexing</u>**
- The schedule table for each cycle can be different



Cycle multiplexing can be used to increase the number of frames that can be transmitted in a schedule

- If processes (applications) need a slower Tx period than defined by the cycle time

Frames can be transmitted with a period that is (a power of two) multiple of the cycle time

In the same slot different frames can be transmitted in different cycles

- Static segment:
  Same node sends in same slot in every cycle
- Dynamic segment:
  Different nodes can send in same slot ("Slot Multiplexing")

Static vs Dynamic Segment

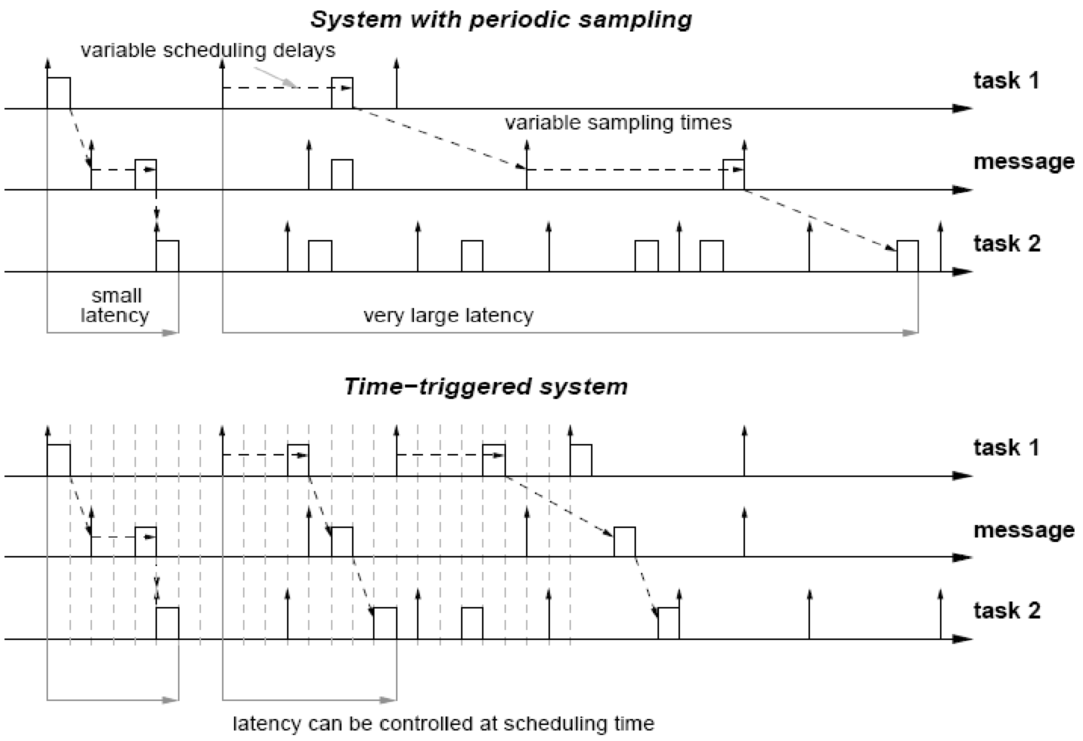
- Allows more efficient use of bandwidth for information that is not sent every cycle
- Messages can be sent on event
- A message “not sent” only takes up the duration of a minislot, not the entire duration of transmission
- Bandwidth benefit obviously a function of the ratio of minislot duration to duration of an occupied slot
- event triggered communication is priority-based: messages with lower slot numbers take precedence over messages with higher slot numbers
- Can be thought of as an arbitration procedure
- Arbitration is somewhat different from CAN: time is consumed by unused slots, it is possible that a message ready for transmission will not be sent even though the network was idle long enough to have allowed its transmission

FlexRay: application

- The definition of a FlexRay communication schedule depends on many factors
  - Use of Static/Dynamic/Both segments
  - Communication and synchronization model: loose or tight synchronization of task and message scheduling
  - Need for standardization and reuse (planning for reuse and extensibility)
  - Protocol constraints and technology constraints
  - Other issues
    - Integration of event-triggered and time-triggered systems
  - Other optimizations ….
    - Avoiding jitter
    - Minimizing latencies
    - Providing time determinism

FlexRay: application

- The definition of a FlexRay communication schedule depends on many factors
  - Communication and synchronization model: loose or tight synchronization of task and message scheduling

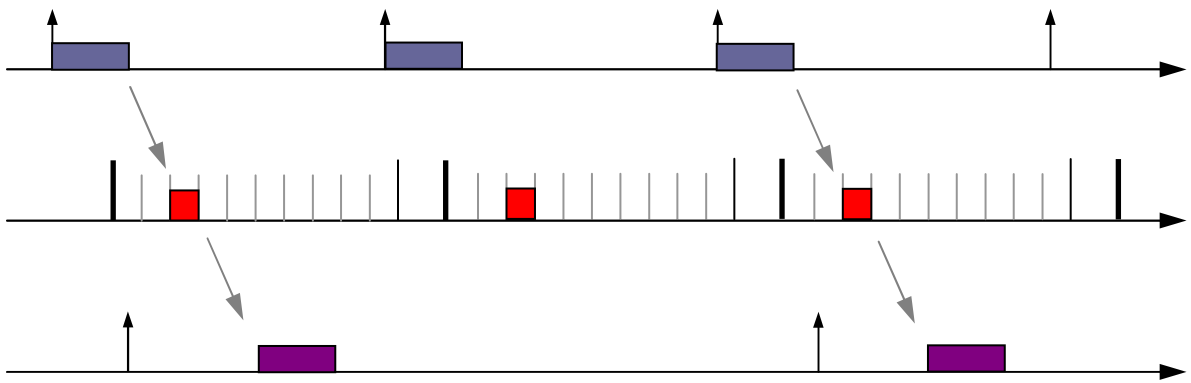


From fully event-based asynchronous to fully-synchronized.

Opportunity: reducing latencies and jitter

FlexRay: application

- The definition of a FlexRay communication schedule depends on many factors
  - Communication and synchronization model: loose or tight synchronization of task and message scheduling

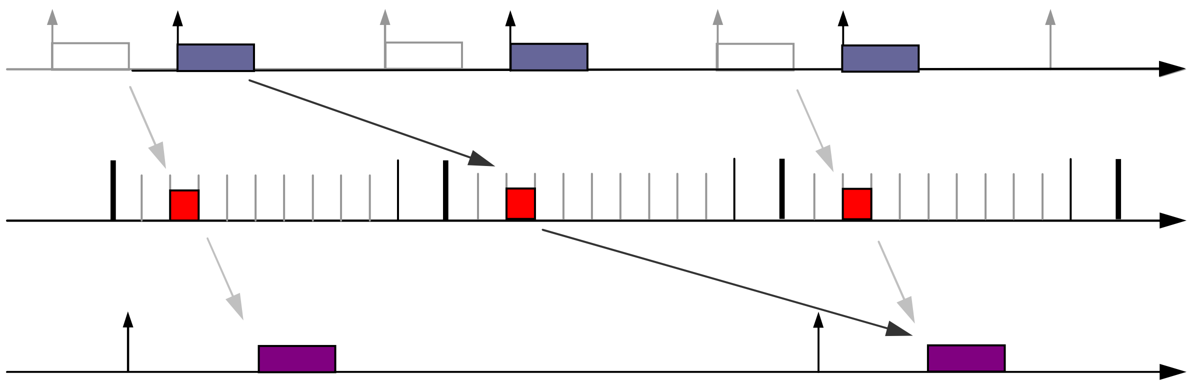


Possible intermediate solution, task and message domains are not synchronized

Simplifies implementation, but loses most of the advantages of FlexRay at the boundary between subsystems

FlexRay: application

- The definition of a FlexRay communication schedule depends on many factors
  - Communication and synchronization model: loose or tight synchronization of task and message scheduling

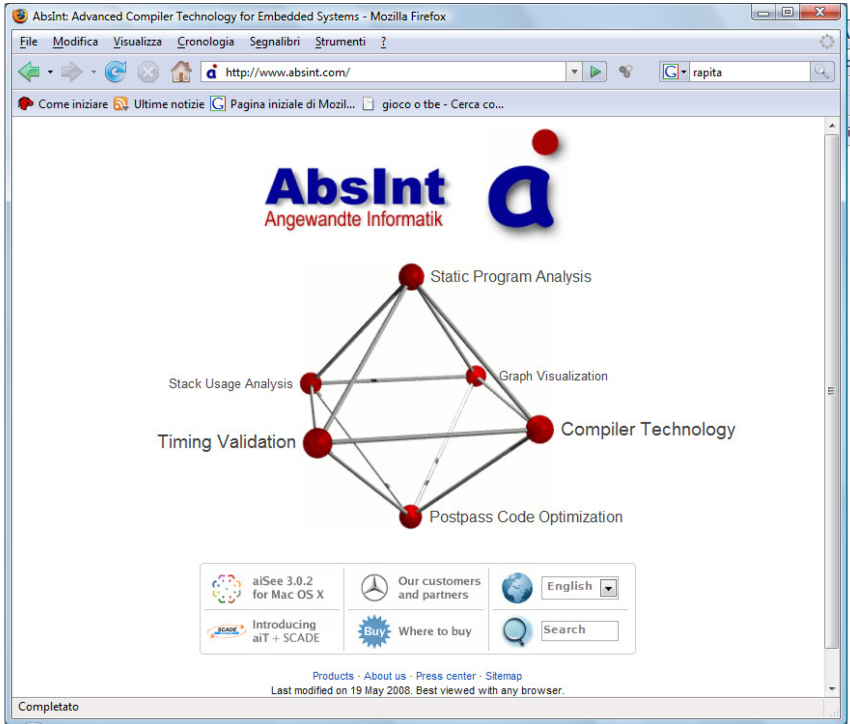


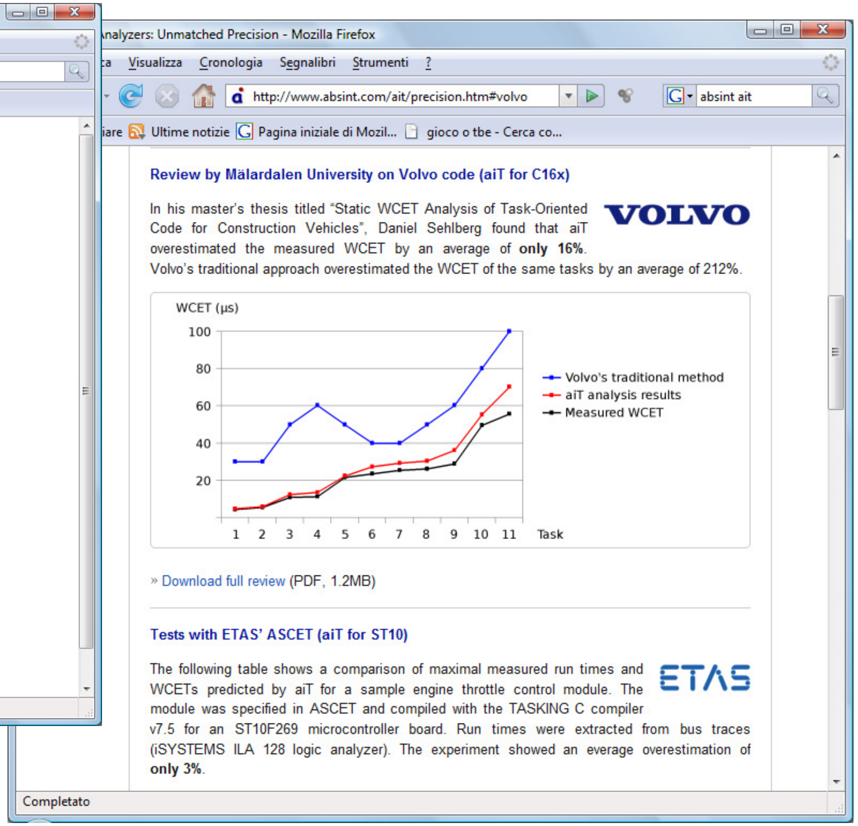
Possible intermediate solution, task and message domains are not synchronized

Simplifies implementation, but loses most of the advantages of FlexRay at the boundary between subsystems

FlexRay: application

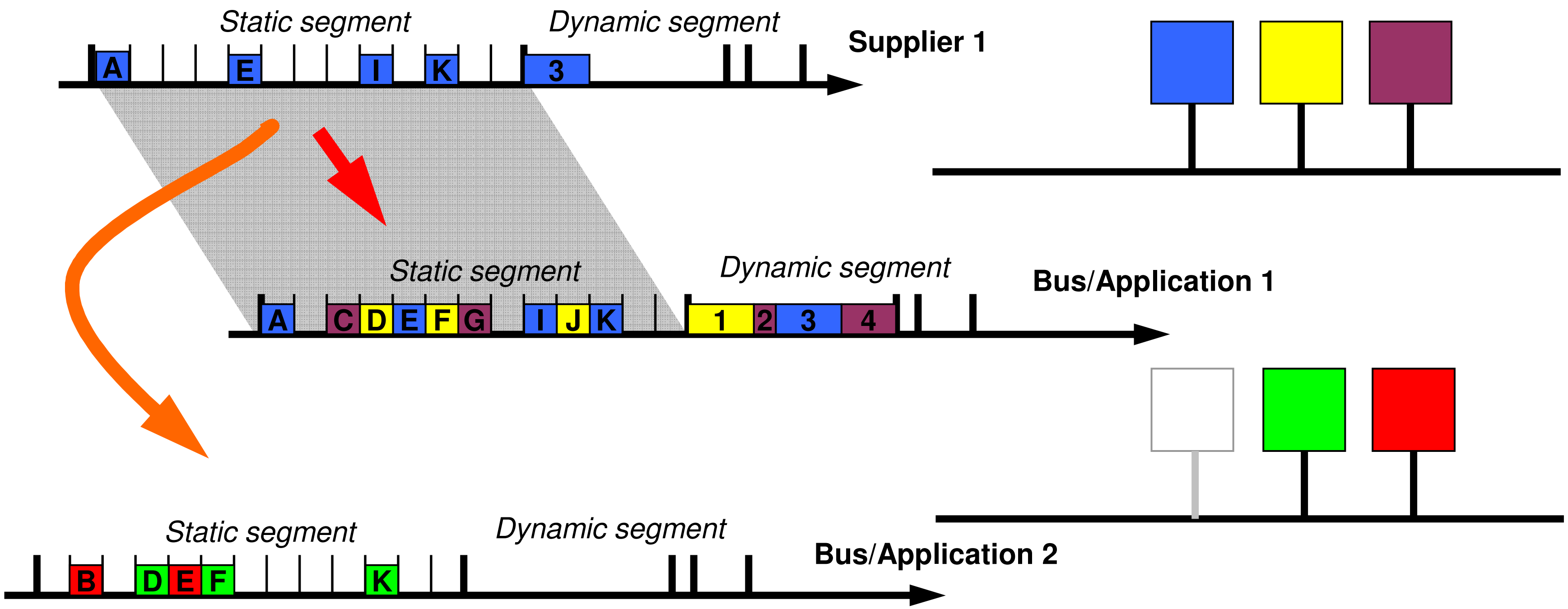
- Tight synchronization of task and message scheduling needs a design-time characterization of the worst case execution time of tasks
  - By profiling and testing with high coverage
  - By static analysis
  - An example: AiT tool from AbsInt or Rapita tools





FlexRay: application

- The definition of a FlexRay communication schedule depends on many factors
  - Need for standardization and reuse



Portability across applications/buses requires standardization of cycle and slot sizes and planning for extensions

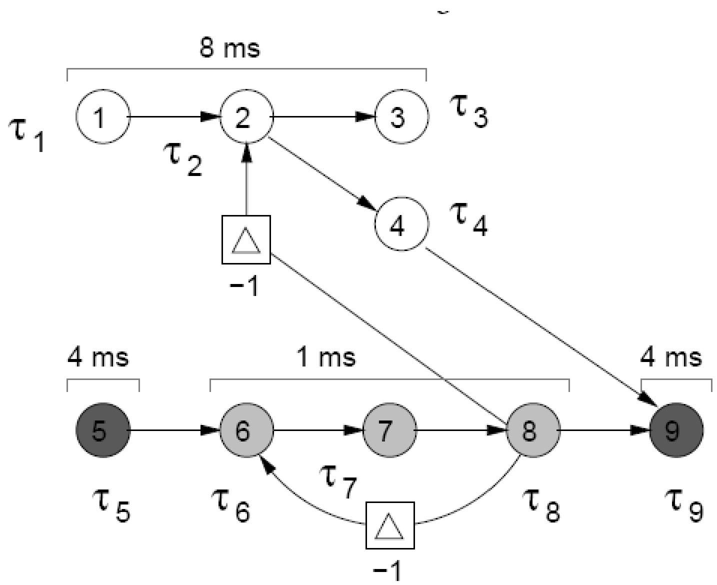
FlexRay: application

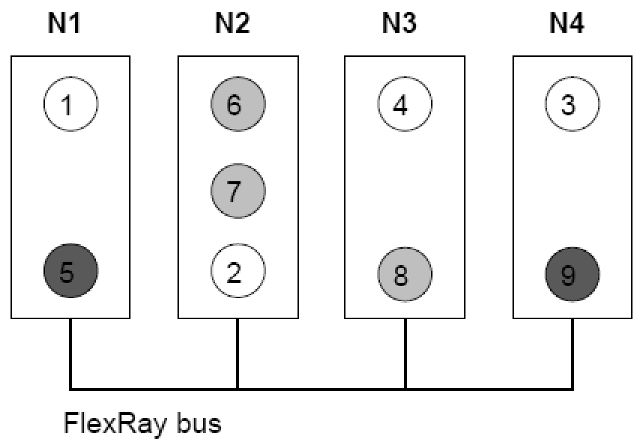
- The definition of a FlexRay communication schedule depends on many factors
  - Protocol constraints and technology constraints

- Each slot is statically allocated to a node or unused. If a node uses in one FlexRay cycle, other nodes cannot use the same index in following cycles.
- Max communication cycle is 16 ms
- Other limitations on the max number of slots
- Typically, there is a tradeoff between slot size (flexibility) and efficiency
- Adapter registers allocation (as performed by the tools)

FlexRay: An MILP scheduler

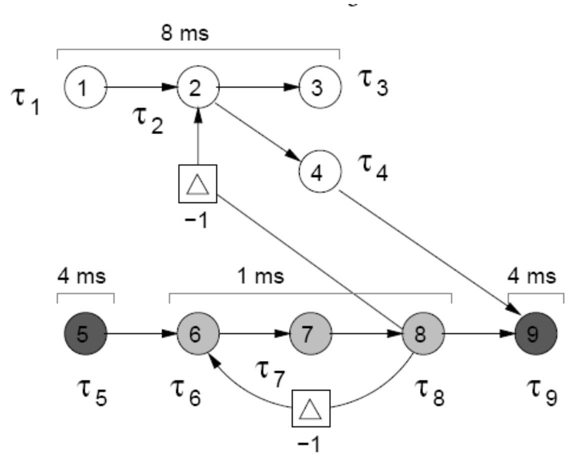
- A FlexRay communication schedule may be defined starting from an application specification.
- The application may be described listing tasks, periods and the signals they are exchanging.

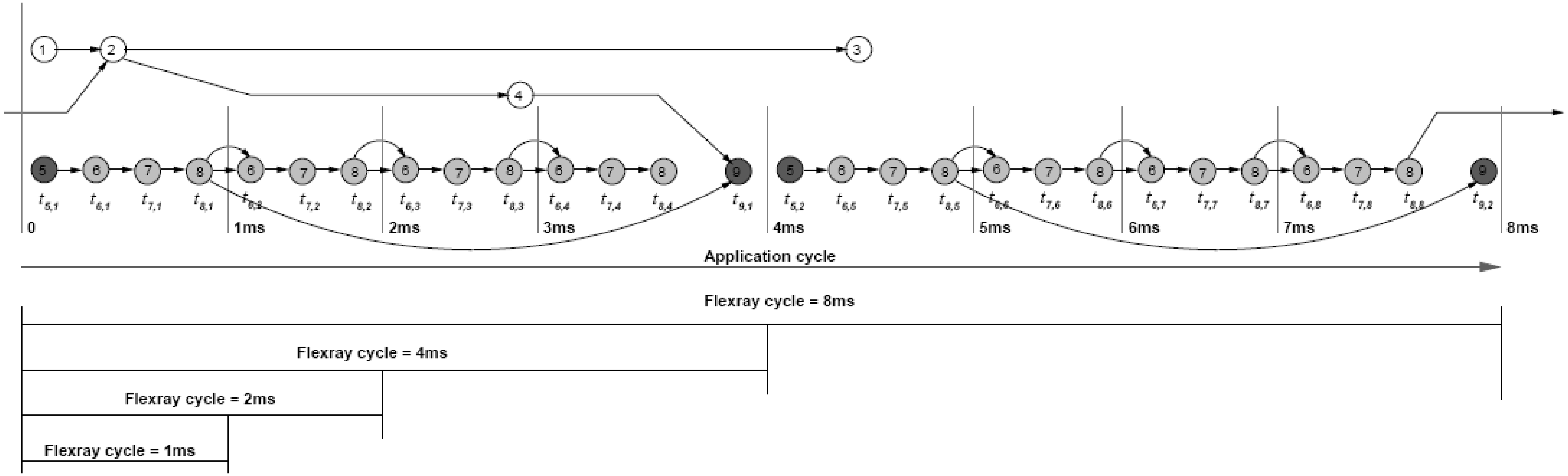




FlexRay: An MILP scheduler

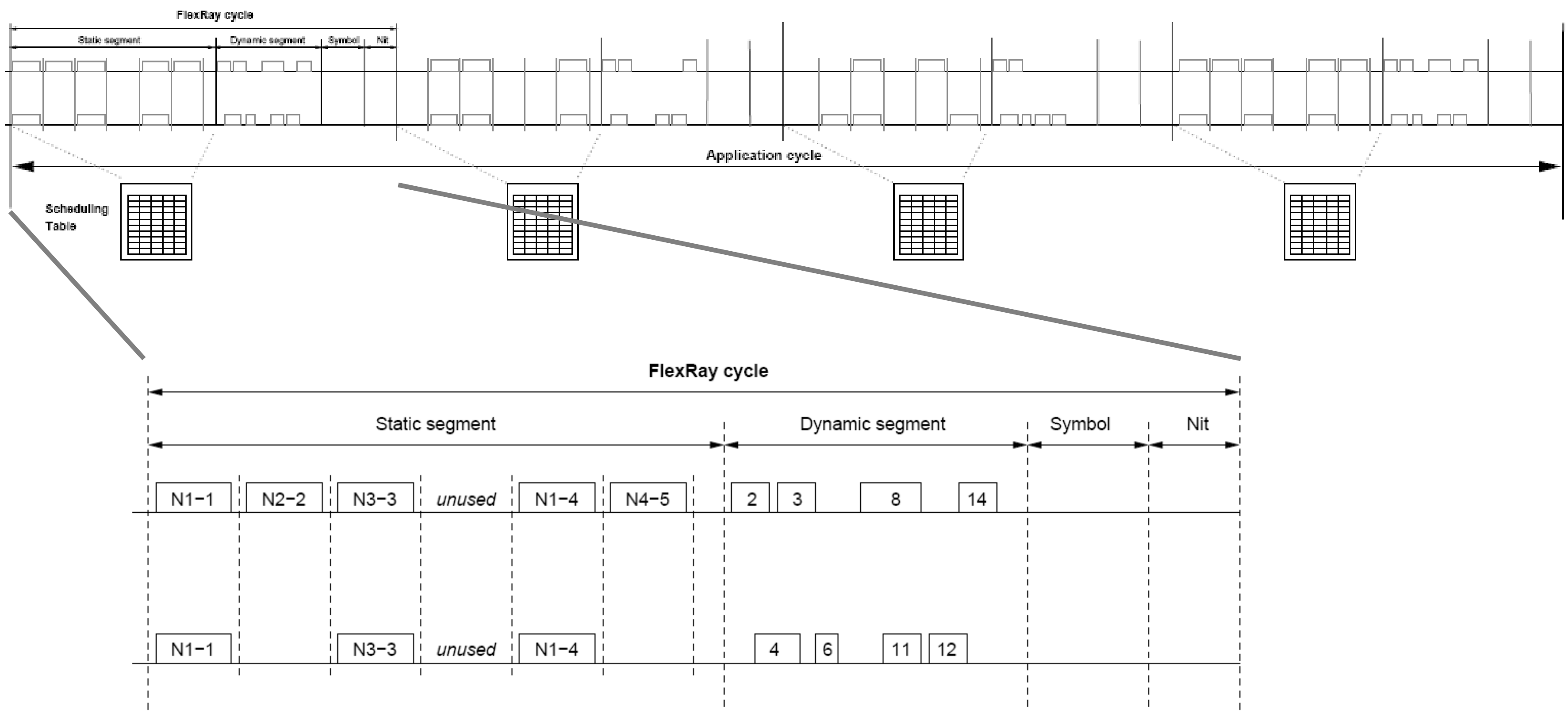
- The application model is unrolled by listing all the task and signal instances in the Application cycle (e.g. the hyperperiod or LCM of the task periods)





FlexRay: An MILP scheduler

- The communication cycle or FlexRay cycle, is then defined as a submultiple (power of 2) of the application cycle

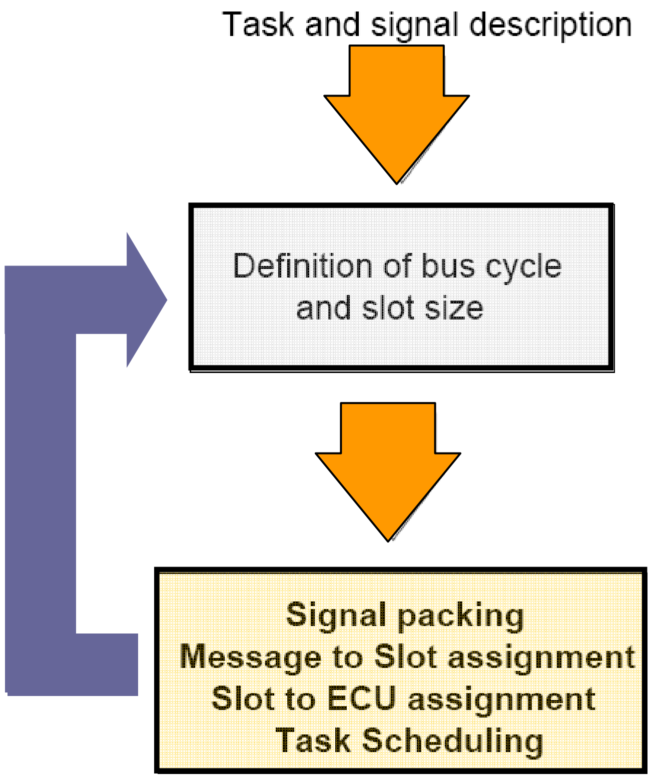


An MILP scheduling solution

- Using MILP for scheduling computation and optimization
  - Task and message scheduling are synchronized
  - Tasks may be scheduled by priority (OSEK) or time-triggered (OSEKTime)
  - Scheduler checks feasibility conditions
    - Total size of signals transmitted in a slot <= frame size
    - Precedence constraints between tasks and messages
    - Period constraints
  - but also allows other types of constraints
    - Maximum jitter
    - Worst case end-to-end latency
  - Allows modeling realistic implementations
    - Scheduling delays/scheduling jitter
    - Copy time from application to peripheral adapter
  - Allows optimization with respect to metric functions
    - Latency/jitter minimization
    - Possibly extensibility ?

An MILP scheduling solution

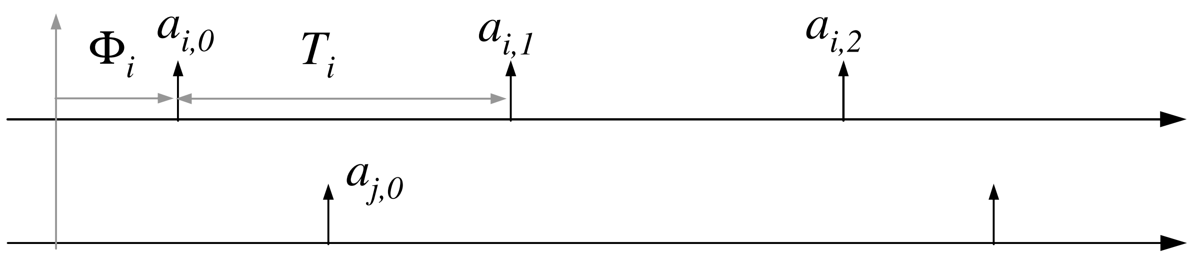
- Input: task and signal description
- A two-stage solution.
  - In the first stage the FlexRay cycle and the slot size are selected
  - In the second stage the scheduling is completed, peromring the signal to slot mapping, the slot to node assignment and possibly the scheduling of the tasks
    - OSEK (priority) based or
    - ESKTime (time-triggered) based
- The steps can be performed iteratively



Activation, release and deadline constraints

- Task activations are periodic with known period
- Activation phases Φ_i are optimization variables

$$a_{i,k} - a_{i,k-1} = T_i$$
$$a_{i,o} = \Phi_i$$
$$0 \le \Phi_i \le T_i$$



- Tasks must complete before the deadlines.
- f_i are optimization variables, d_i are system parameters

$$f_i \le d_i$$

Task scheduling

- OSEKTime scheduling
  - Time-triggered, task start times are defined in the scheduling table, once started they can possibly preempt other tasks that are in execution (or be preempted)

$$a_i \leq s_i$$

- OSEK scheduling
  - Priority-based, at activation each task is ready and is executed by a scheduler based on its priority

$$0 \leq a_{i,k} - A_{i,k} \leq J_i$$

Start times and preemption

- (OSEKTime only) Order of start times and preemption

$$y_{i,j} = \begin{cases} 0, & \text{if start time of } t_i \text{ precedes start time of } t_j \\ 1, & \text{otherwise} \end{cases}$$

"big M" formulation

$$s_i < s_j + y_{i,j} \times M$$
$$s_j < s_i + (1 - y_{i,j}) \times M$$

Enforce the meaning of y

$$p_{i,j} = \begin{cases} 0, & \text{if task } t_i \text{ is not preempted by task } t_j \\ 1, & \text{otherwise} \end{cases}$$

$$p_{i,j} + p_{j,i} \leq 1$$

Tasks cannot mutually preempt

$$p_{i,j} \leq 1 - y_{i,j}$$

A task can preempt only if it starts later

$$f_i \leq s_j + y_{i,j} \times M + p_{i,j} \times M$$
$$f_j < f_i + y_{i,j} \times M + (1 - p_{i,j}) \times M$$

Based on start times order and on preemption definition order finish times

Task finish times

- OSEKTime

$$f_i = s_i + C_i + \sum_{(i,j)\in\theta} p_{i,j} \times C_j$$

Finish time, accounting for preemption

- OSEK

$$r_i = C_i + \sum_{j\in hp(i)} \lceil \frac{r_i + J_j}{T_j} \rceil C_j$$

Response time computation for preemptive, priority based scheduling

$$r_i = C_i + \sum_{j\in hp(i)} (\frac{r_i + J_j}{T_j} + \alpha) C_j$$

… and its linear approximation

$$f_i \geq A_i + r_i$$

FlexRay protocol rules

$$s^s_{j,k} = l_{comm} \times j + l_{slot} \times k.$$

Beginning of slot k in cycle j

$$A_{i,j,k} = \begin{cases} 1, & \text{if signal } m_i \text{ is mapped to com cycle j, slot k} \\ 0, & \text{otherwise} \end{cases}$$

$$s^s_{j,k} \le s_i + (1 - A_{i,j,k}) \times M$$
$$s_i \le s^s_{j,k} + (1 - A_{i,j,k}) \times M$$
$$f_i \le s^s_{j,k} + l_{slot} + (1 - A_{i,j,k}) \times M$$
$$s^s_{j,k} \le f_i - l_{slot} + (1 - A_{i,j,k}) \times M$$

If a signal is mapped into a slot, its start and finish times are the same as the start and finish times of the slot

$$\sum_{j<n_{comm}, k<n_{slot}} A_{i,j,k} = 1$$

A signal can be assigned to one slot only

$$\sum_{i \in \phi} C_i \times A_{i,j,k} \le l_{slot}$$

The total size of the signals mapped into one slot cannot exceed the slot size

Slot ownership

$$A_{e_i,j} = \begin{cases} 1, & \text{if slot j is owned by ECU } e_i \\ 0, & \text{otherwise} \end{cases}$$

$$A_{i,j,k} \leq A_{e_i,k}$$

If a signal is mapped into a slot, the ECU of its source task must own the slot

$$\sum_{e_p \in \varepsilon} A_{e_p,k} \leq 1$$

Each slot can be owned by at most one ECU

$$A_{e_p,k} \leq \sum_{i \in \phi, j < n_c omm} A_{i,j,k}$$

All the slots with the same index must be mapped to the same ECU

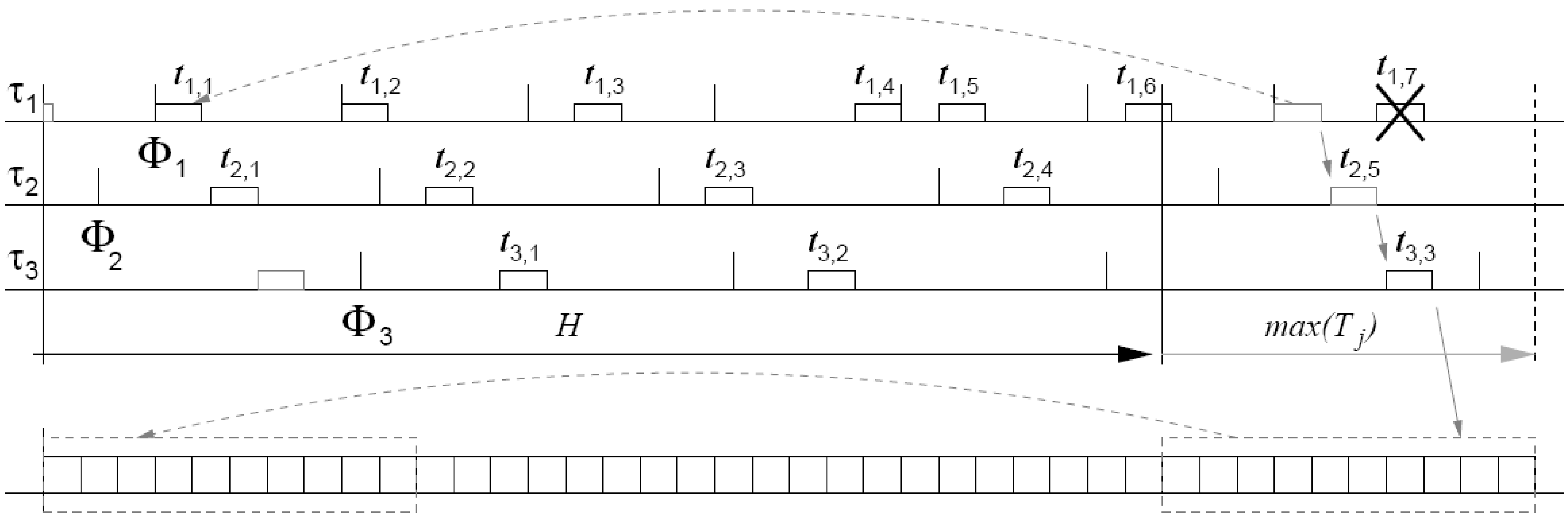
Data dependencies

$$f_i \leq s_j - \gamma_j$$

Each task must finish before the start time of the signal it produces. Each signal must arrive before the start of the task that consumes it (with a margin accounting for the copy time)

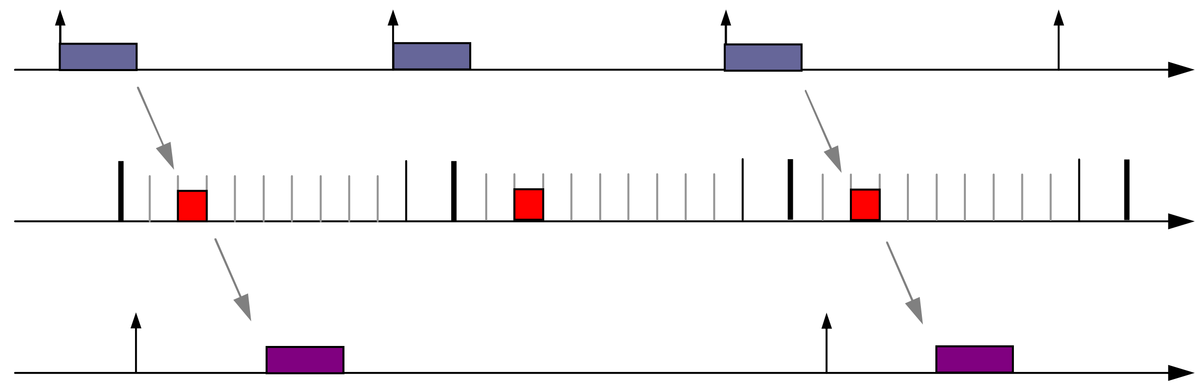
Wraparound conditions

- The schedule must extend up to the application cycle plus the maximum possible initial phase (initial period)
- However, the head and the tail of the schedule must be kept consistent.



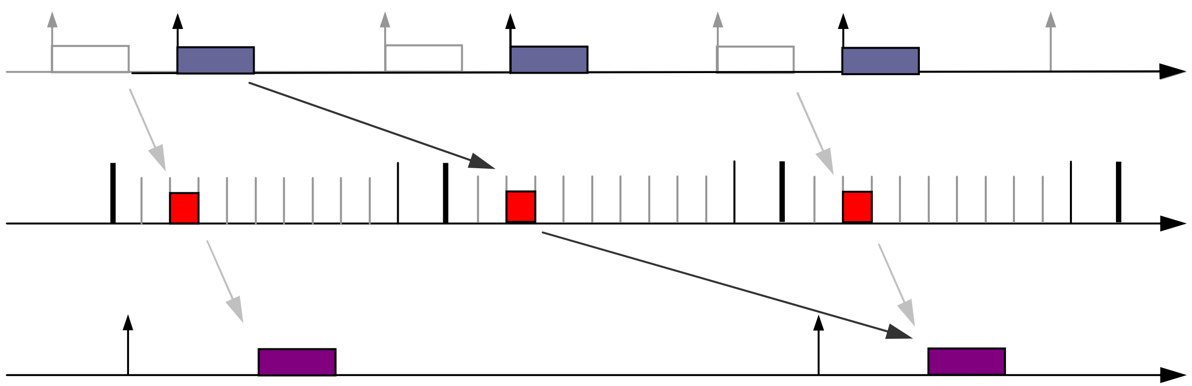
FlexRay: application

- Communication and synchronization model: loose or tight synchronization of task and message scheduling



task and message domains are synchronized

Predictable and tight latencies



task and message domains are not synchronized

Simplifies implementation, but loses most of the advantages of FlexRay at the boundary between subsystems

Use of Dynamic vs Static Segment

- Allows more efficient use of bandwidth for information that is not sent every cycle
- Messages can be sent on event
- A message “not sent” only takes up the duration of a minislot, not the entire duration of transmission
- Bandwidth benefit obviously a function of the ratio of minislot duration to duration of an occupied slot
- ***Need models for analysis and synthesis (guide the design)***

FlexRay: application

- Tight synchronization of task and message scheduling needs a design-time characterization of the worst case execution time of tasks
  - By profiling and testing with high coverage
  - By static analysis
  - An example: AiT tool from AbsInt or Rapita tools