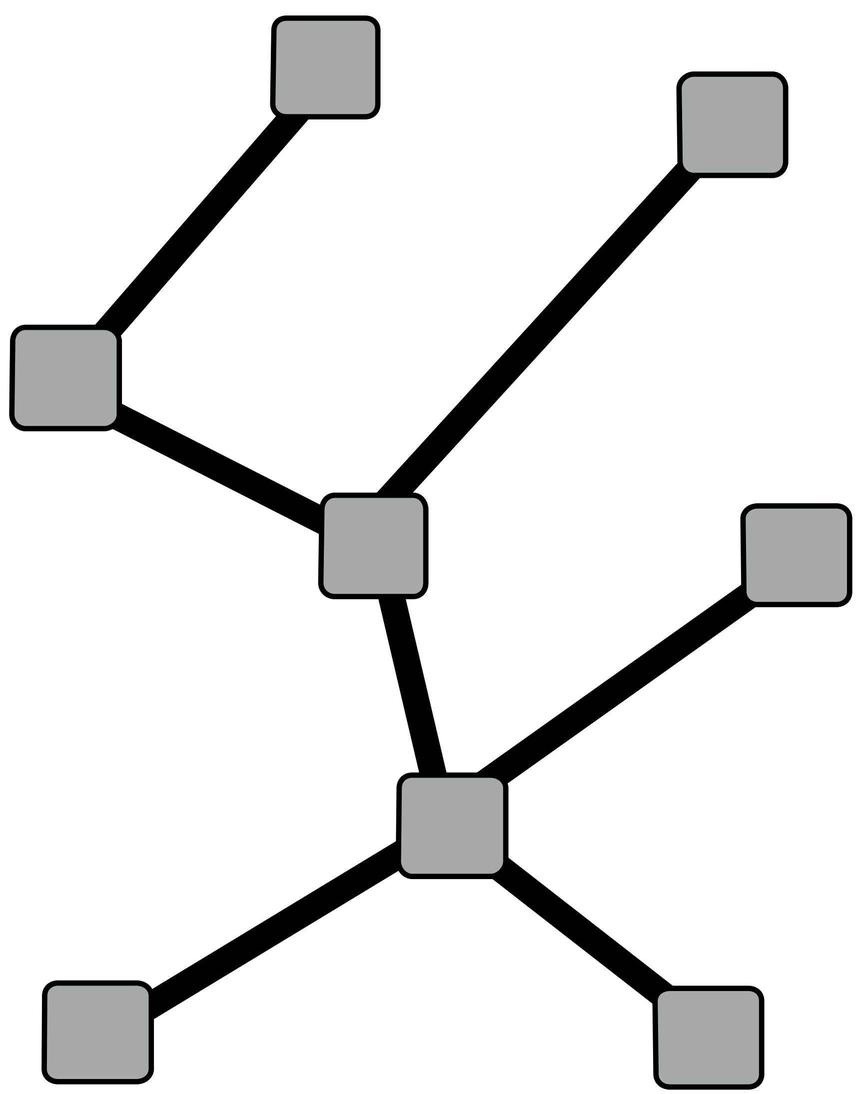
A Synchronous View of Loosely Time-Triggered Architectures

Guillaume Baudart
Albert Benveniste
Timothy Bourke

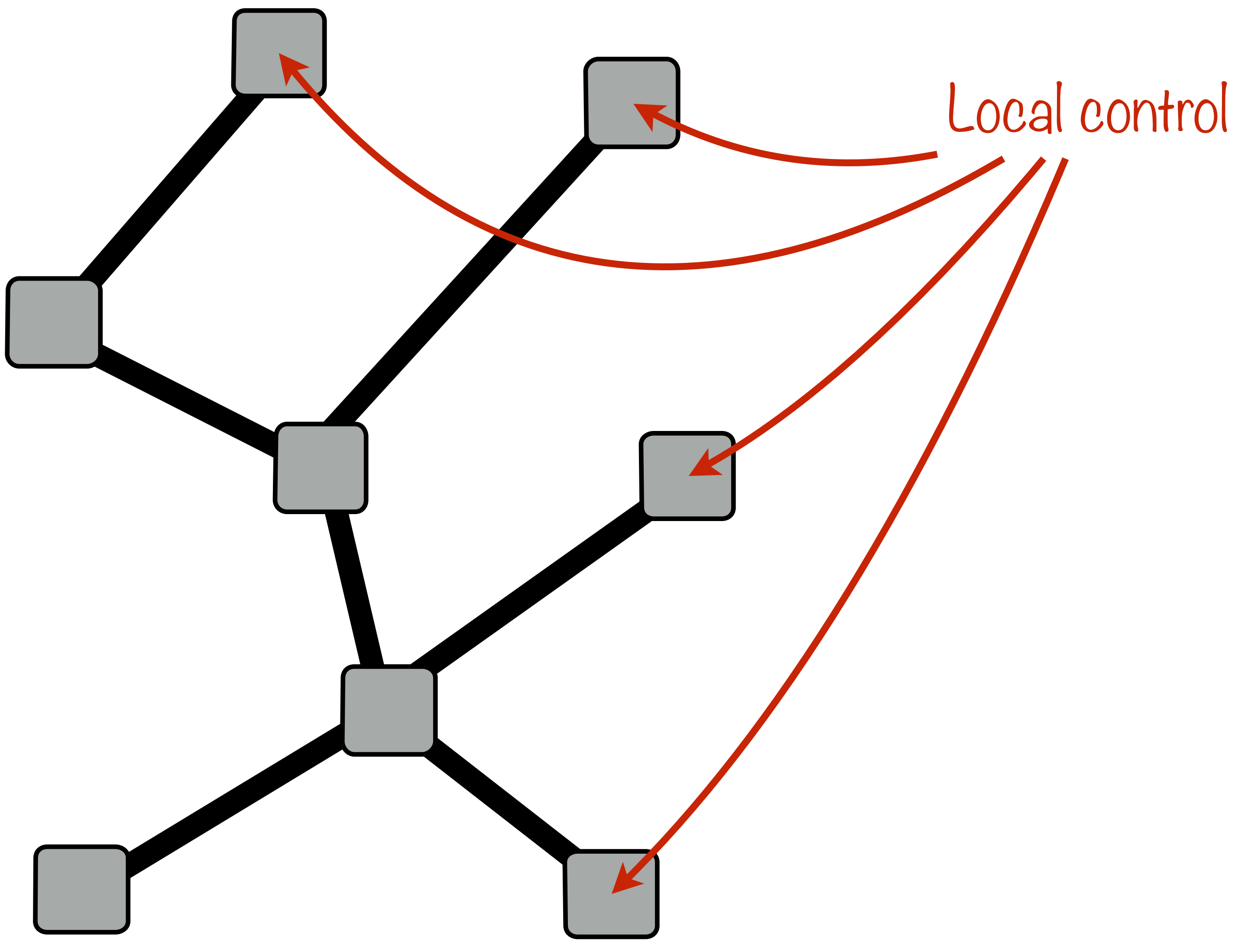
Séminaire 68NQRT'15

Rennes, 30-04-2015

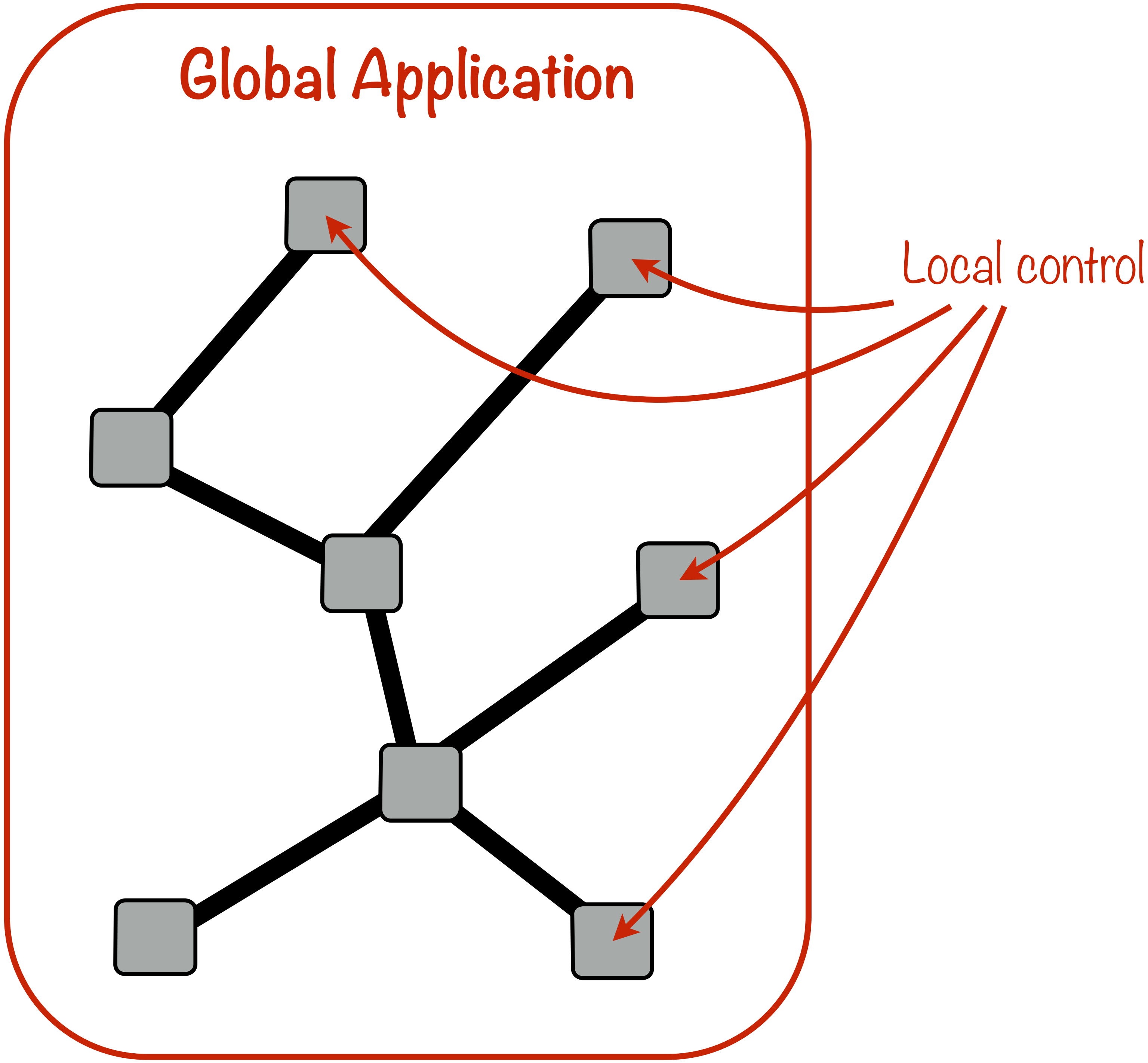
Overview



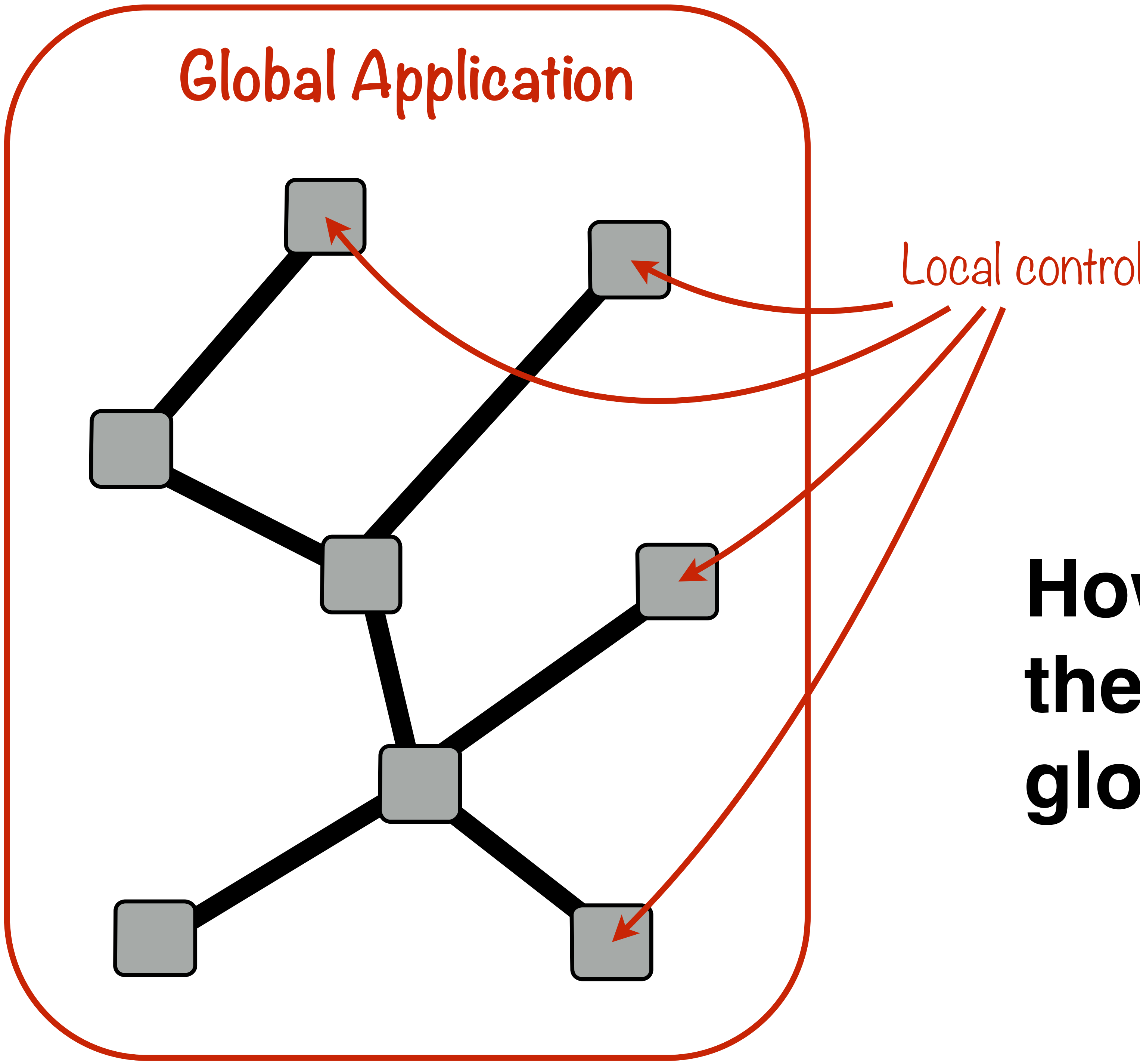
Overview



Overview

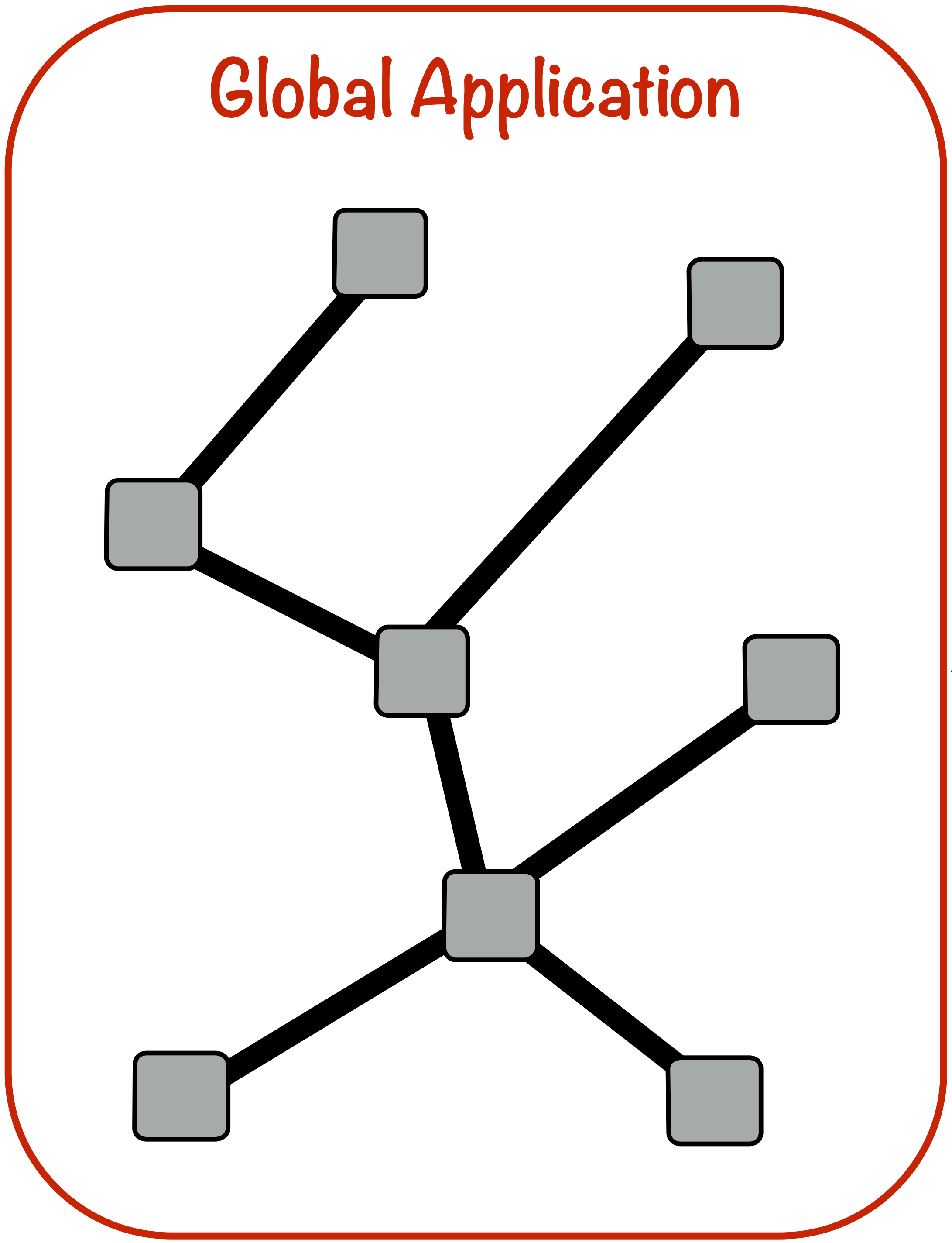


Overview



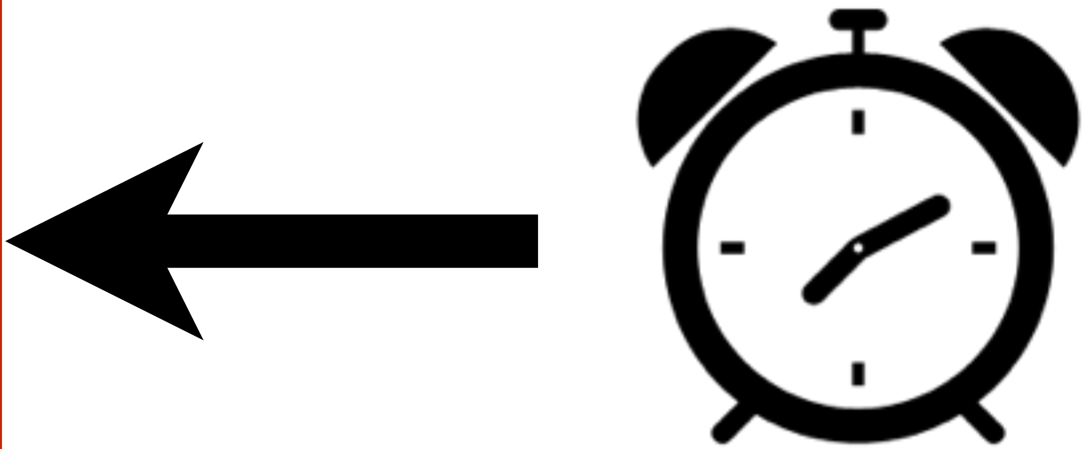
How to preserve the semantics of the global application?

Overview



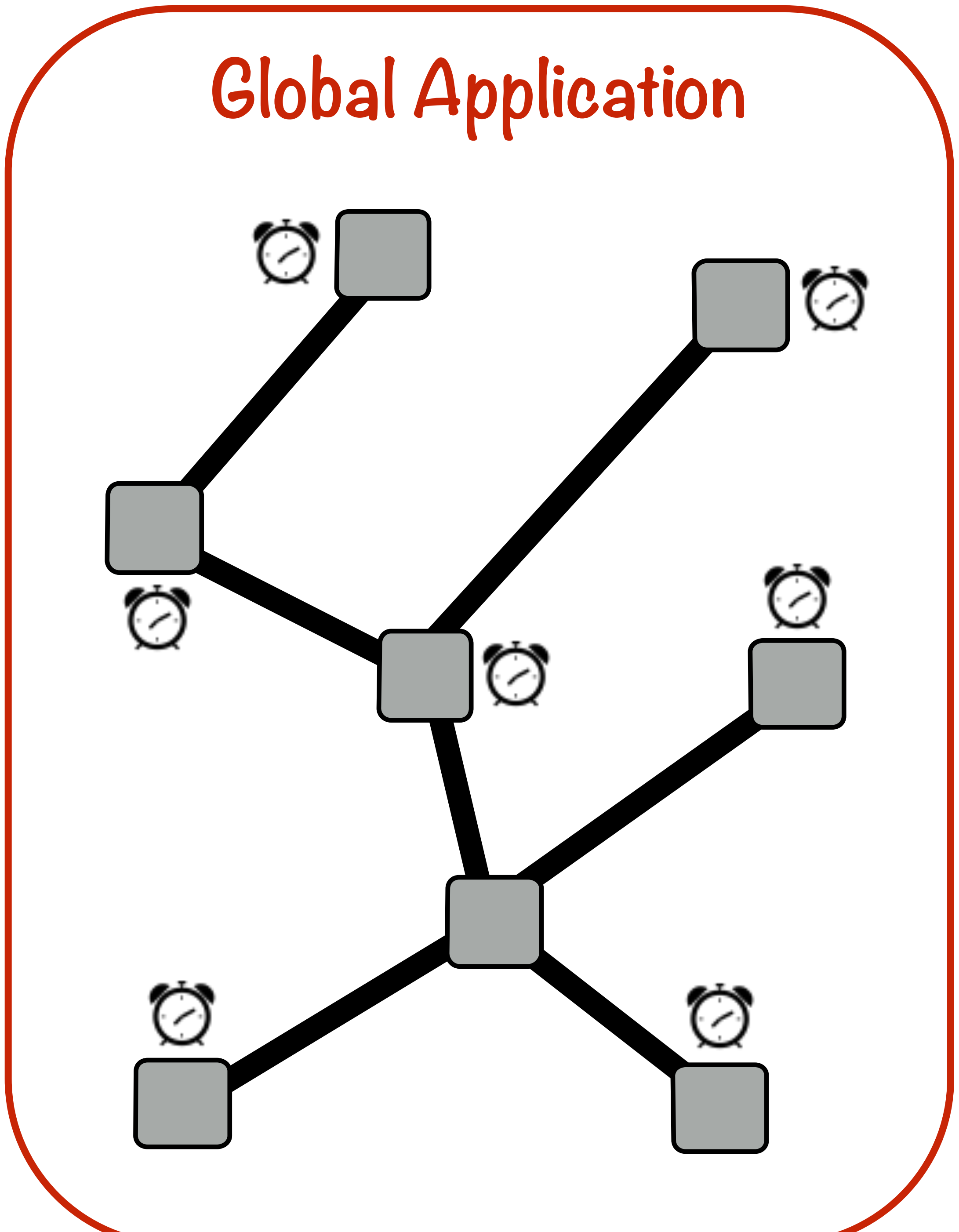
Clock synchronization

eg. TTA [Kopetz, Bauer 2003]



Introduces control dependencies between nodes.

Overview



Unsynchronized nodes + Middleware = LTTA

[Tripakis et al. 2008]
[Caspi, Benveniste 2008]

Less constraining protocols.
Limited control dependencies.

Outline

1. What is an LTTA?
    1. Quasi-Periodic Architecture
    2. Synchronous Applications

2. General Framework

3. The two protocols
    1. Back-Pressure LTTA
    2. Time-Based LTTA

4. What About Clock Synchronisation?

Outline

1. What is an LTTA?
    1. Quasi-Periodic Architecture
    2. Synchronous Applications
2. General Framework
3. The two protocols
    1. Back-Pressure LTTA
    2. Time-Based LTTA
4. What About Clock Synchronisation?

Quasi-Periodic Architecture

- A set of "quasi-periodic" processes with local clocks and nominal period T^n (jitter ε)

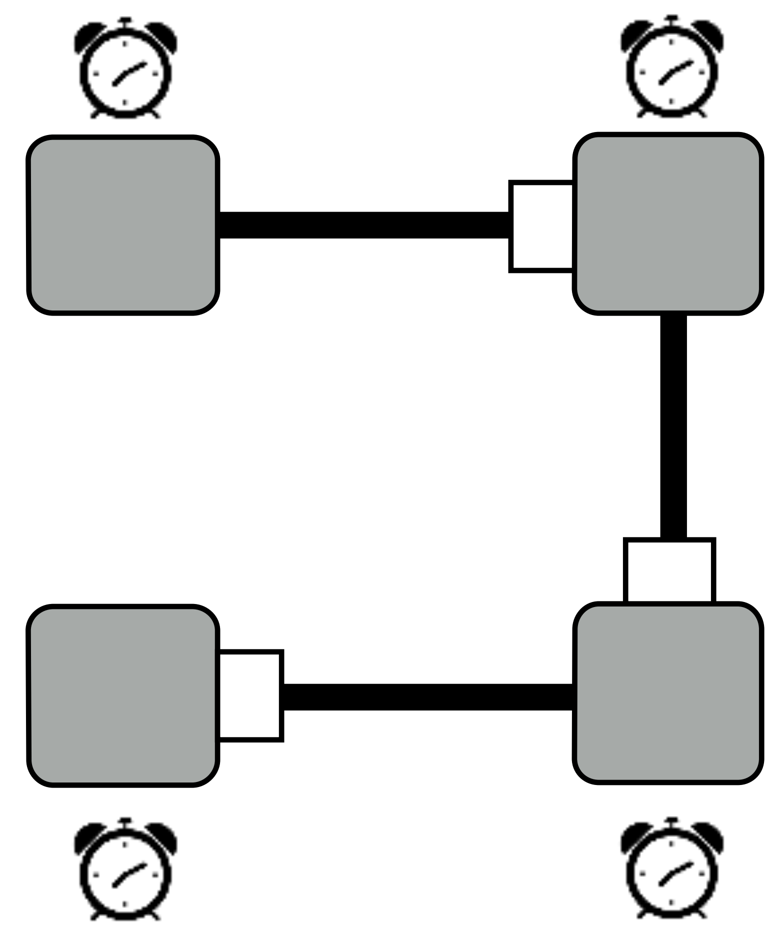
$$0 < T_{\min} \leq T^n \leq T_{\max} \quad \text{or}$$
$$T^n - \varepsilon \leq \kappa_i - \kappa_{i-1} \leq T^n + \varepsilon$$

$(\kappa_i)_{i \in \mathbb{N}}$ clock activations

- Buffered communication without message inversion or loss

- Bounded communication delay

$$\tau_{\min} \leq \tau \leq \tau_{\max}$$



Quasi-Periodic Architecture

- A set of "quasi-periodic" processes with local clocks and nominal period T^n (jitter ε)

$$0 < T_{\min} \leq T^n \leq T_{\max} \quad \text{or}$$

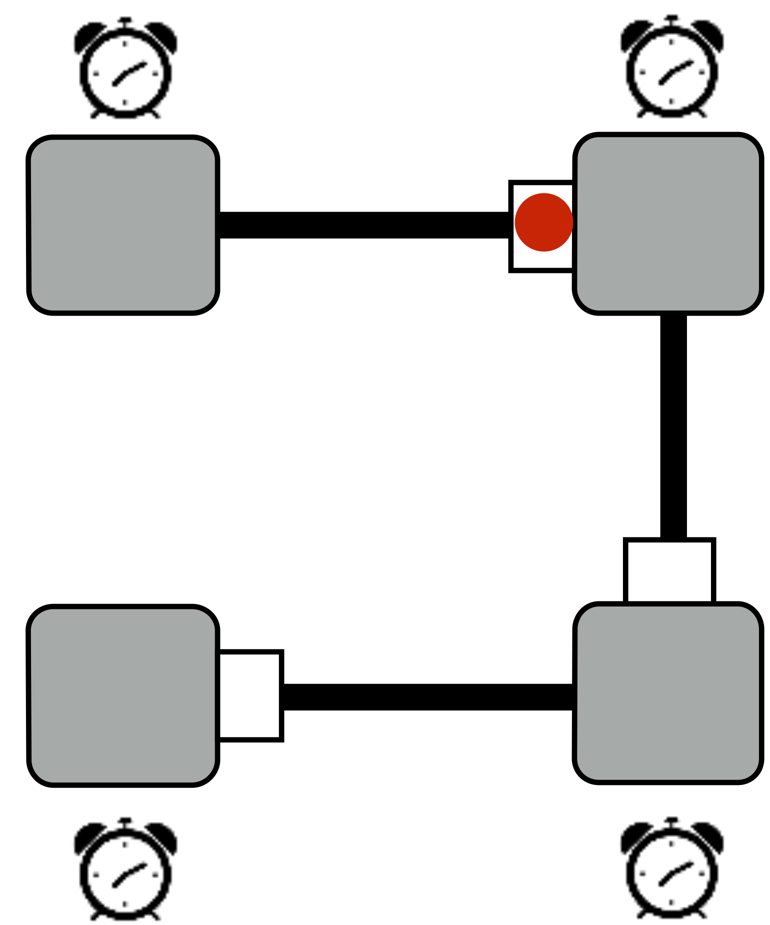
$$T^n - \varepsilon \leq \kappa_i - \kappa_{i-1} \leq T^n + \varepsilon$$

$(\kappa_i)_{i \in \mathbb{N}}$ clock activations

- Buffered communication without message inversion or loss

- Bounded communication delay

$$\tau_{\min} \leq \tau \leq \tau_{\max}$$



Quasi-Periodic Architecture

- A set of “quasi-periodic” processes with local clocks and nominal period T^n (jitter ε)

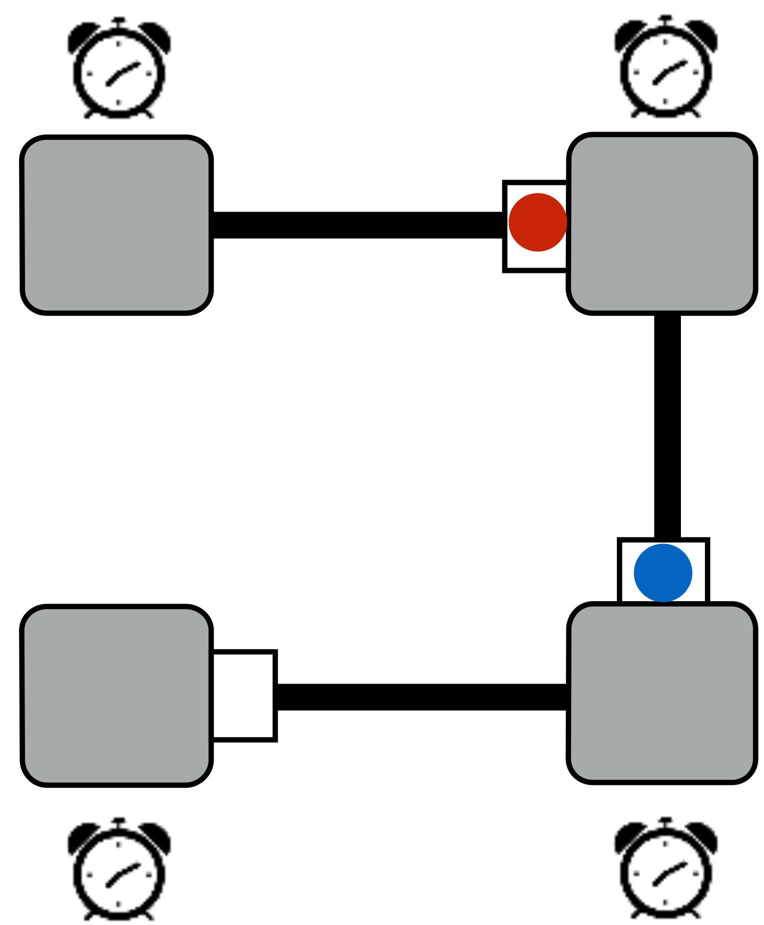
$$0 < T_{\min} \leq T^n \leq T_{\max} \quad \text{or}$$
$$T^n - \varepsilon \leq \kappa_i - \kappa_{i-1} \leq T^n + \varepsilon$$

$(\kappa_i)_{i \in \mathbb{N}}$ clock activations

- Buffered communication without message inversion or loss

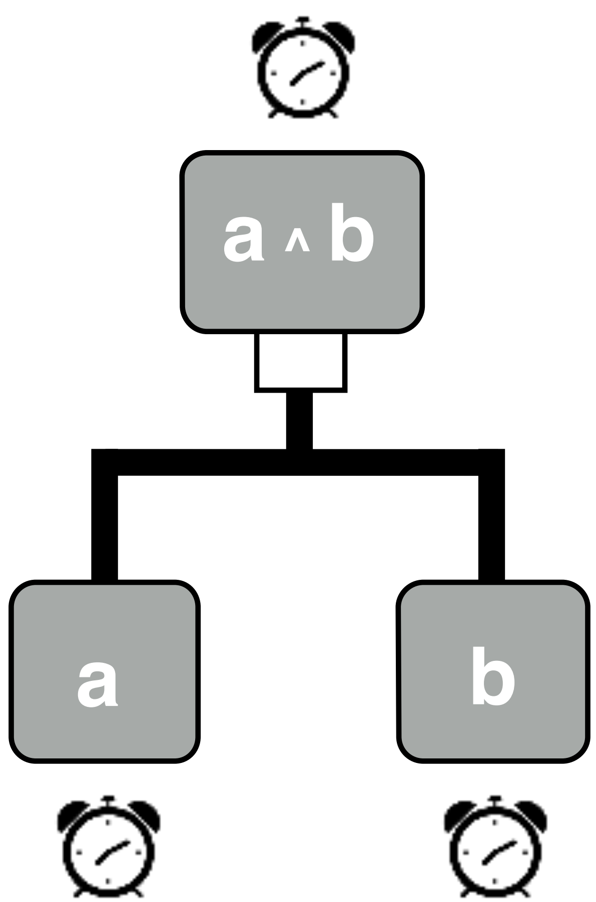
- Bounded communication delay

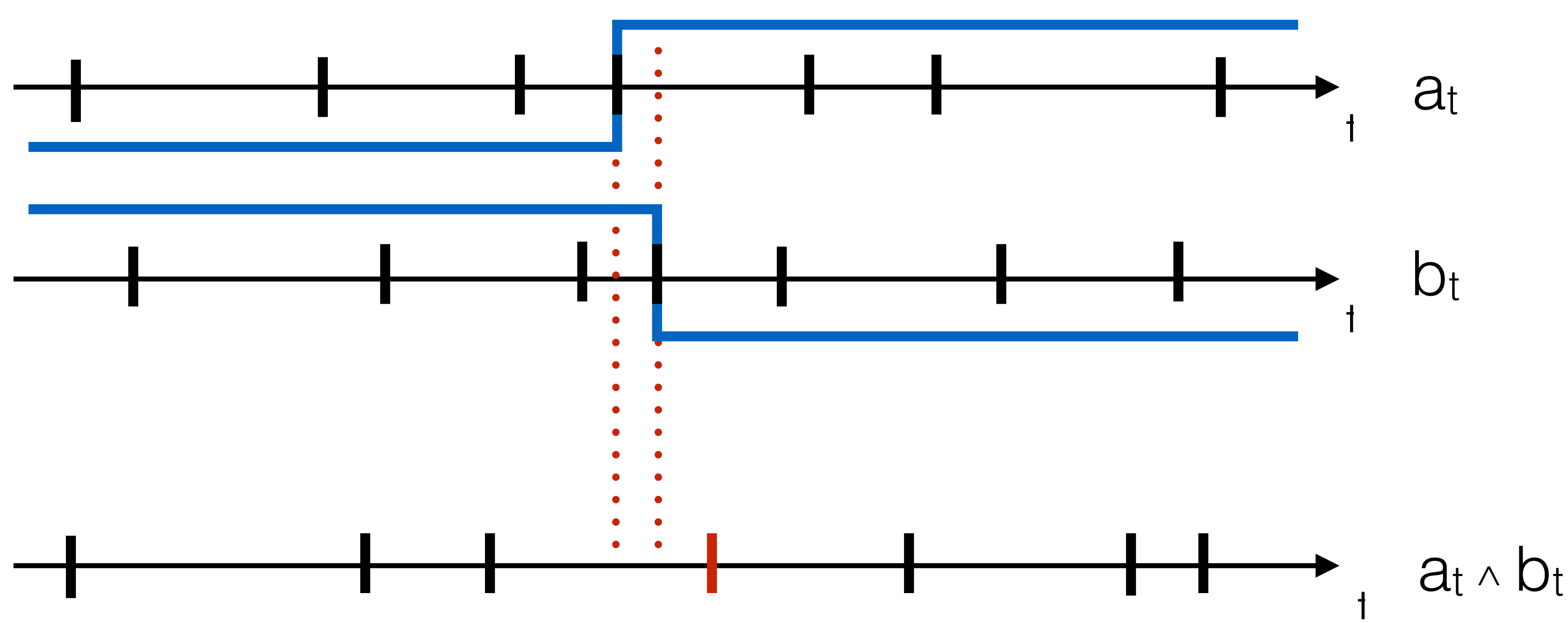
$$\tau_{\min} \leq \tau \leq \tau_{\max}$$



Problems

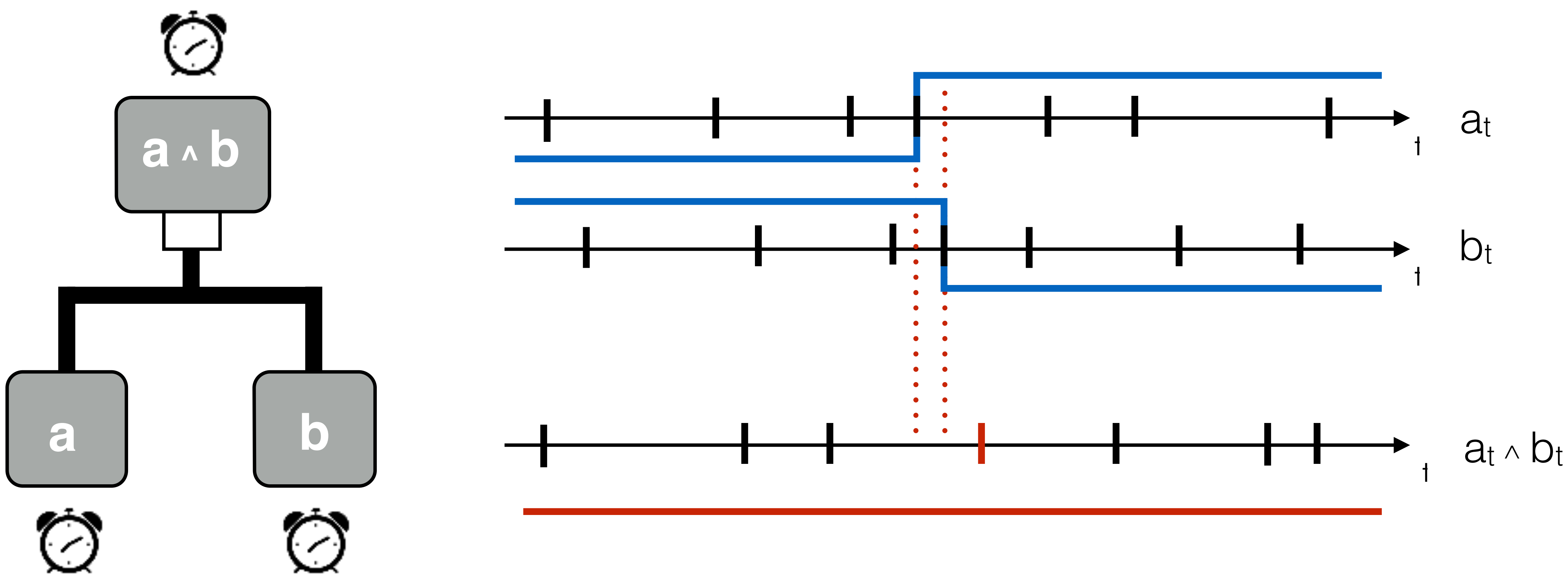
- **Overwriting:** Loss of values
- **Oversampling:** Duplication of values
- **Combination of signals**





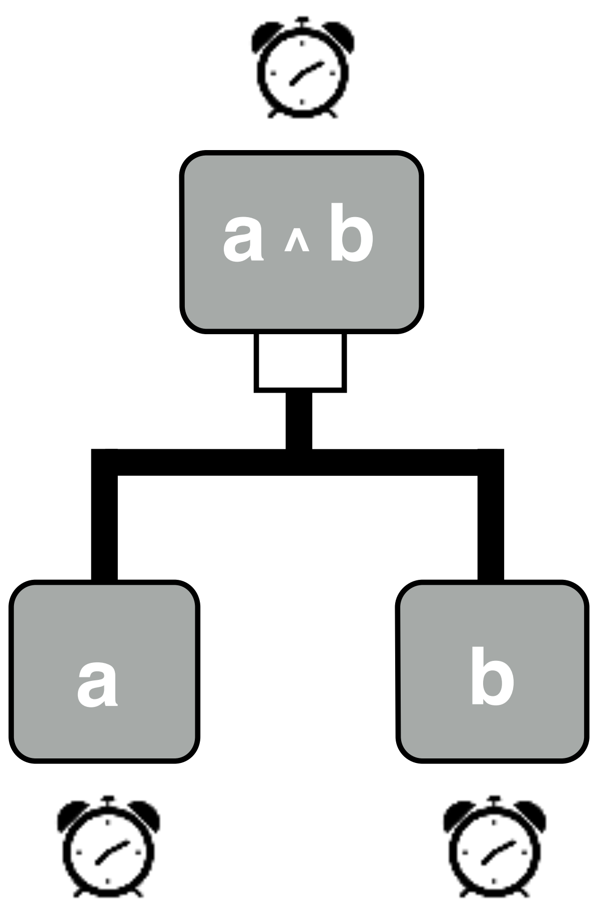
Problems

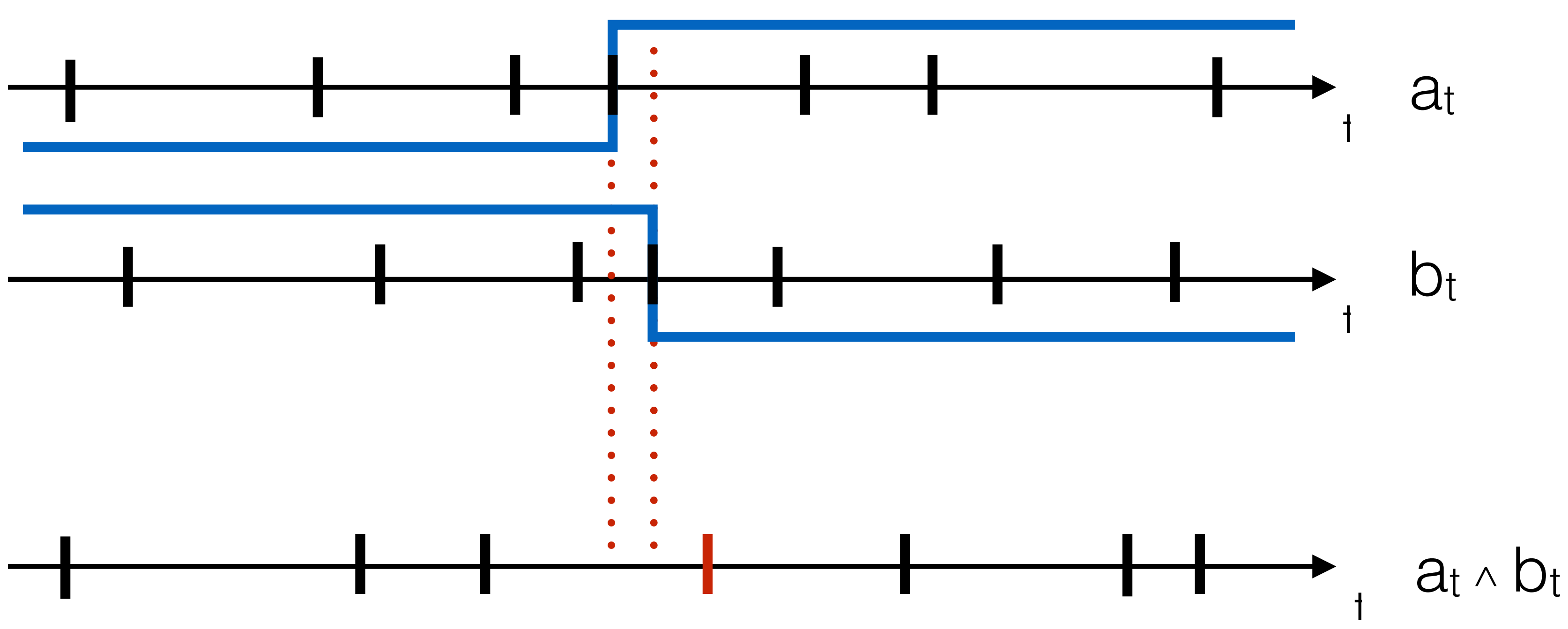
- **Overwriting:** Loss of values
- **Oversampling:** Duplication of values
- **Combination of signals**



Problems

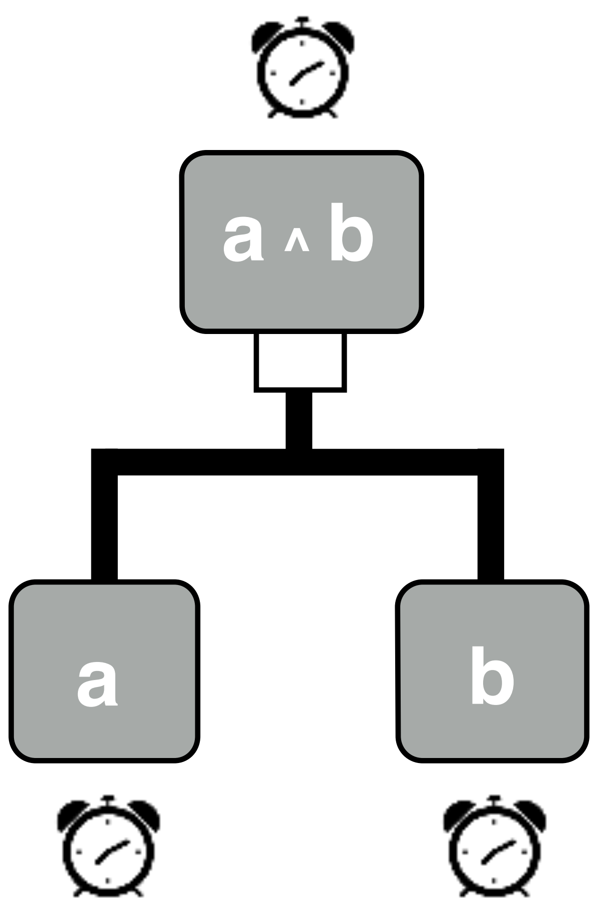
- **Overwriting:** Loss of values
- **Oversampling:** Duplication of values
- **Combination of signals**

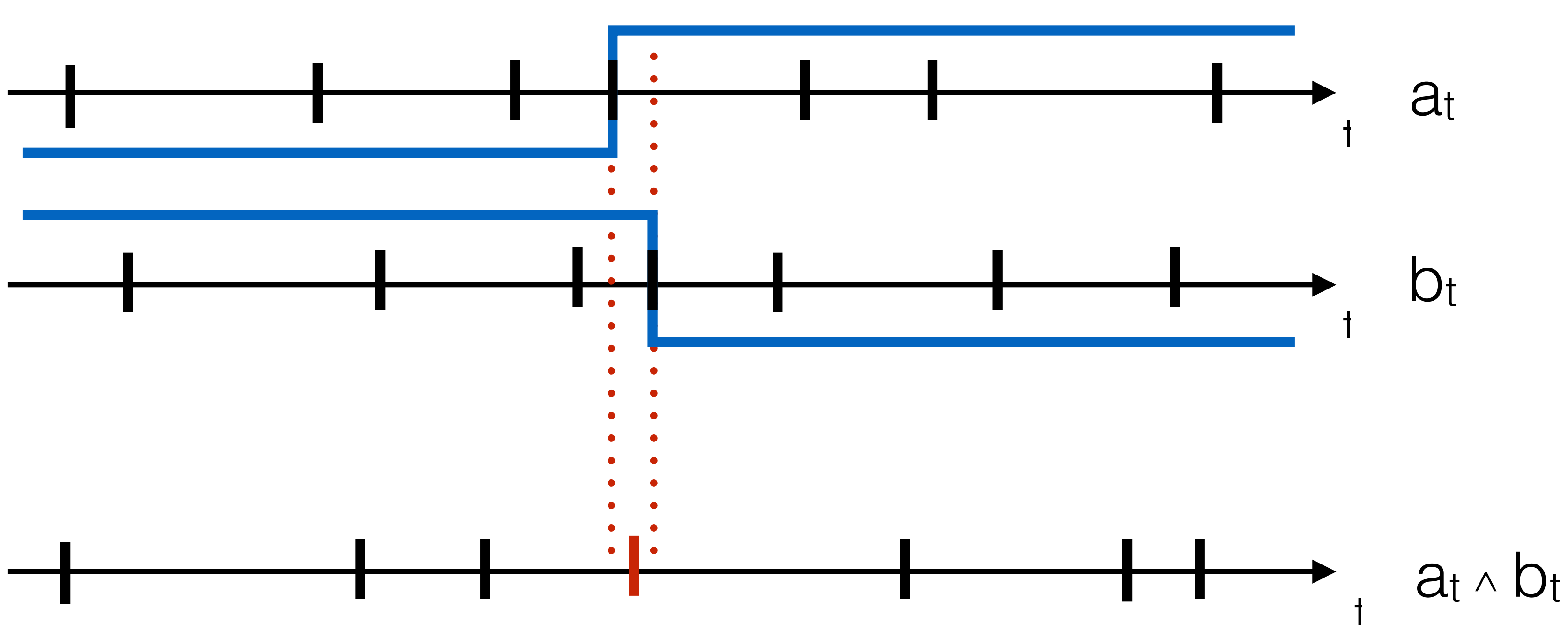




Problems

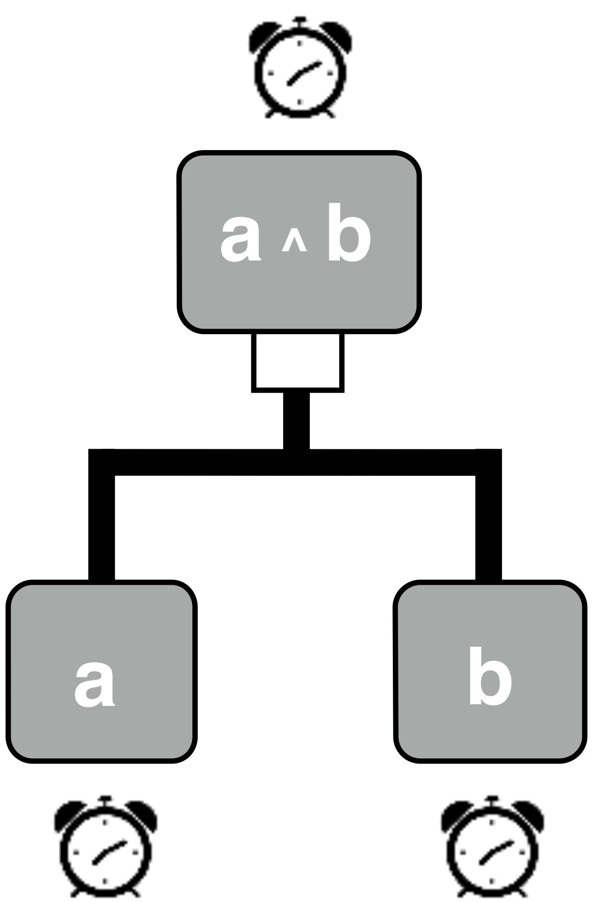
- **Overwriting:** Loss of values
- **Oversampling:** Duplication of values
- **Combination of signals**

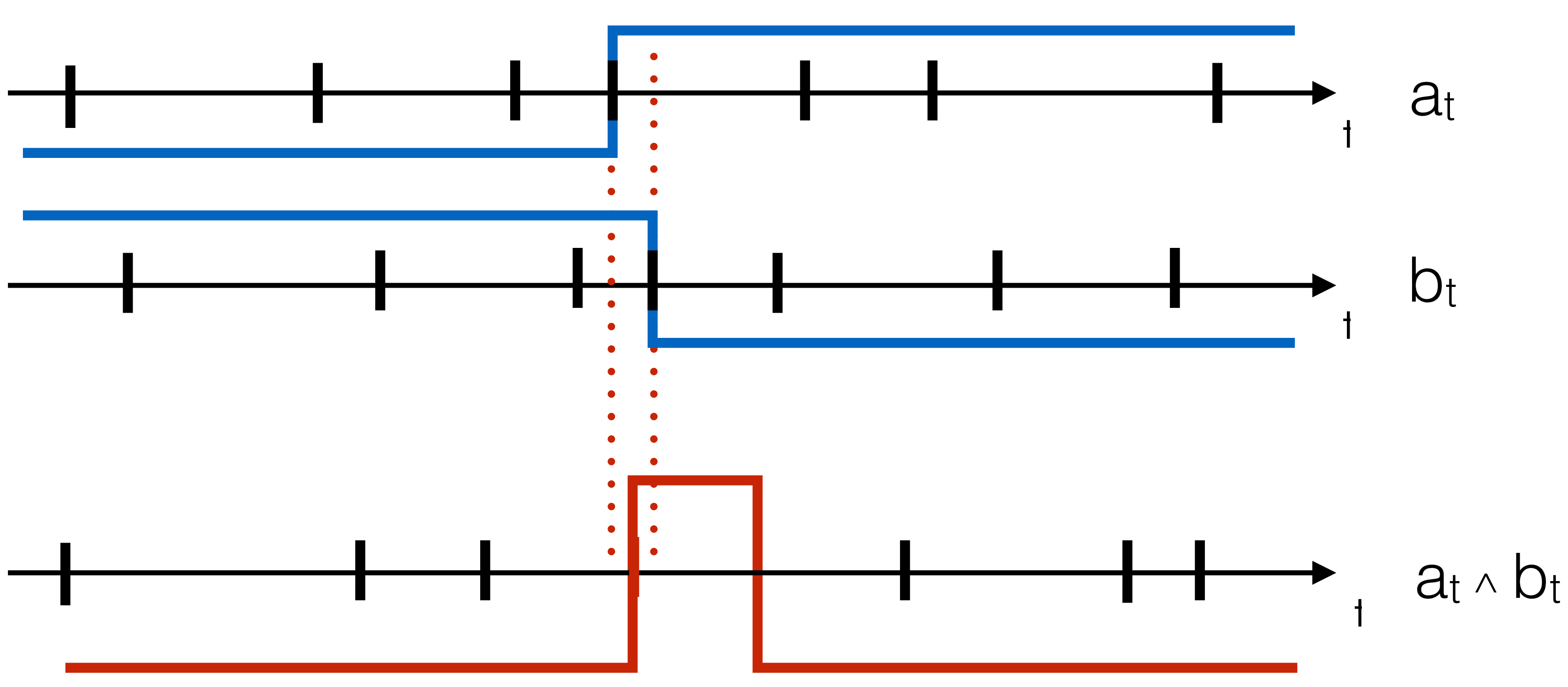




Problems

- **Overwriting:** Loss of values
- **Oversampling:** Duplication of values
- **Combination of signals**





Synchronous Applications

Network of Communicating Mealy Machines

- Initial state S_{init}, Inputs I, Output O
- Transition function $F \;:\; \mathcal{S} \times \mathcal{V}^I \to \mathcal{S} \times \mathcal{V}^O$

Semantics

Synchronous $\quad [\![m]\!]^S \;:\; (\mathcal{V}^{n_i})^\infty \to (\mathcal{V}^{n_o})^\infty$

Kahn $\quad [\![m]\!]^K \;:\; (\mathcal{V}^\infty)^{n_i} \to (\mathcal{V}^\infty)^{n_o}$

Synchronous Applications

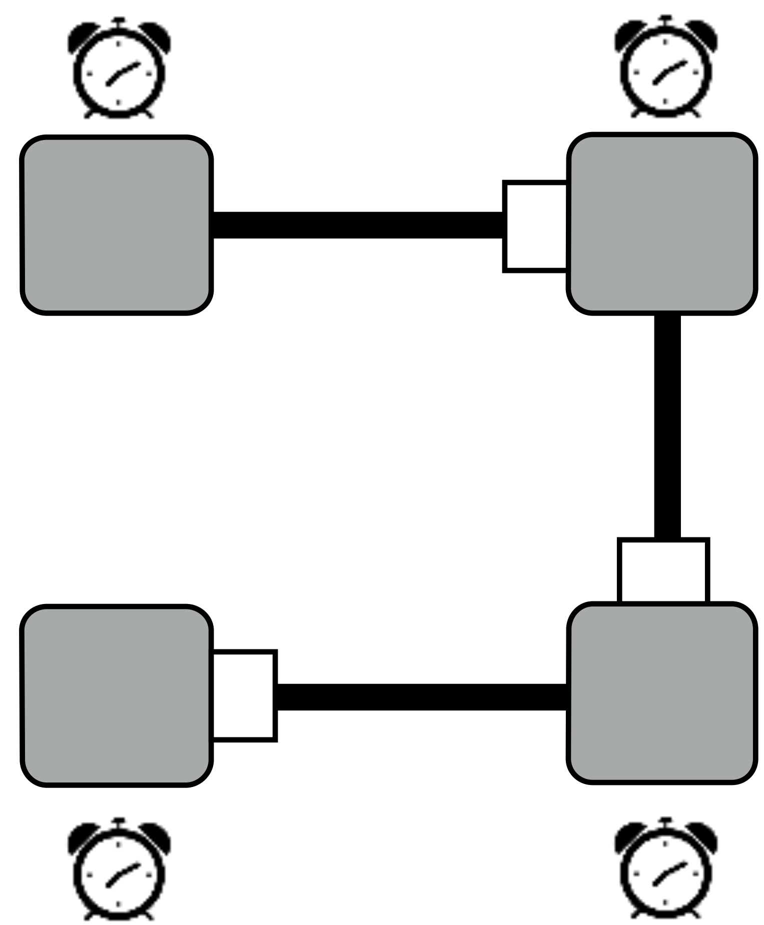
Network of Communicating Mealy Machines

- **‘Moore-style’ Composition:** machines communicate through a unit delay
- **No instantaneous dependency** between nodes (avoid ‘microschedule’ and ‘causality’ issues)
- A composition is also a Mealy machine

Basically, classic synchronous programs ‘mono-clock’.

What are LTTAs?

- **Base:** A quasi-periodic architecture
- **Goal:** Safely deploy a synchronous application
- **Idea:** Add a layer of middleware



Different Approaches

- **Discrete abstractions,** e.g., quasi-synchrony [Caspi 2000]
  - Allows verifications
  - State explosion, incomplete model

- **Petri nets,** [Benveniste et al. 2010]
  - Unify LTTA protocols
  - Complex model, no implementation

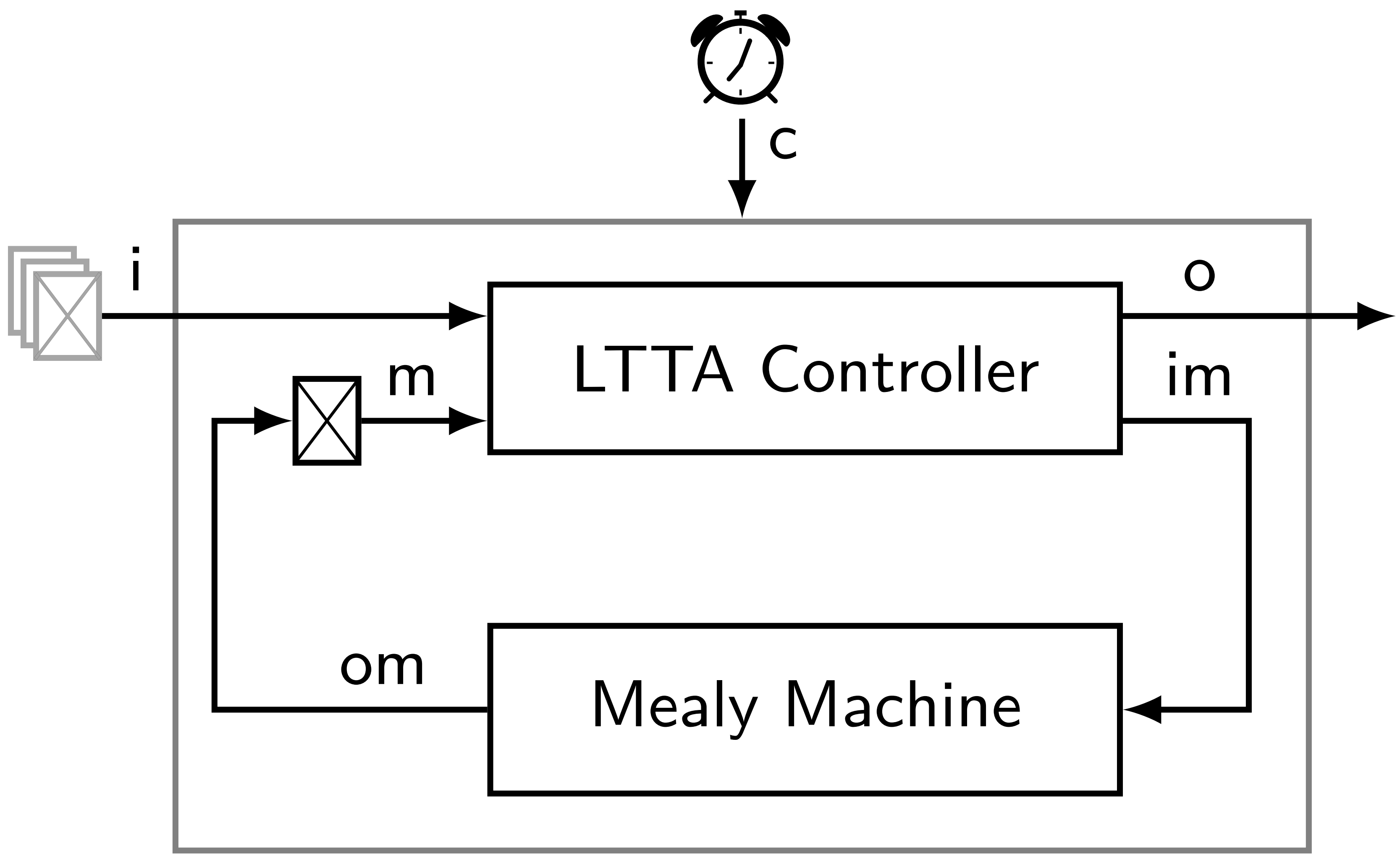
- **Zélus:**
  - A single language for discrete- and real-time,
  - Implementation, simulation using numerical solvers
  - But no verification at this point

Outline

1. What is an LTTA?
    1. Quasi-Periodic Architecture
    2. Synchronous Applications
2. General Framework
3. The two protocols
    1. Back-Pressure LTTA
    2. Time-Based LTTA
4. What About Clock Synchronisation?

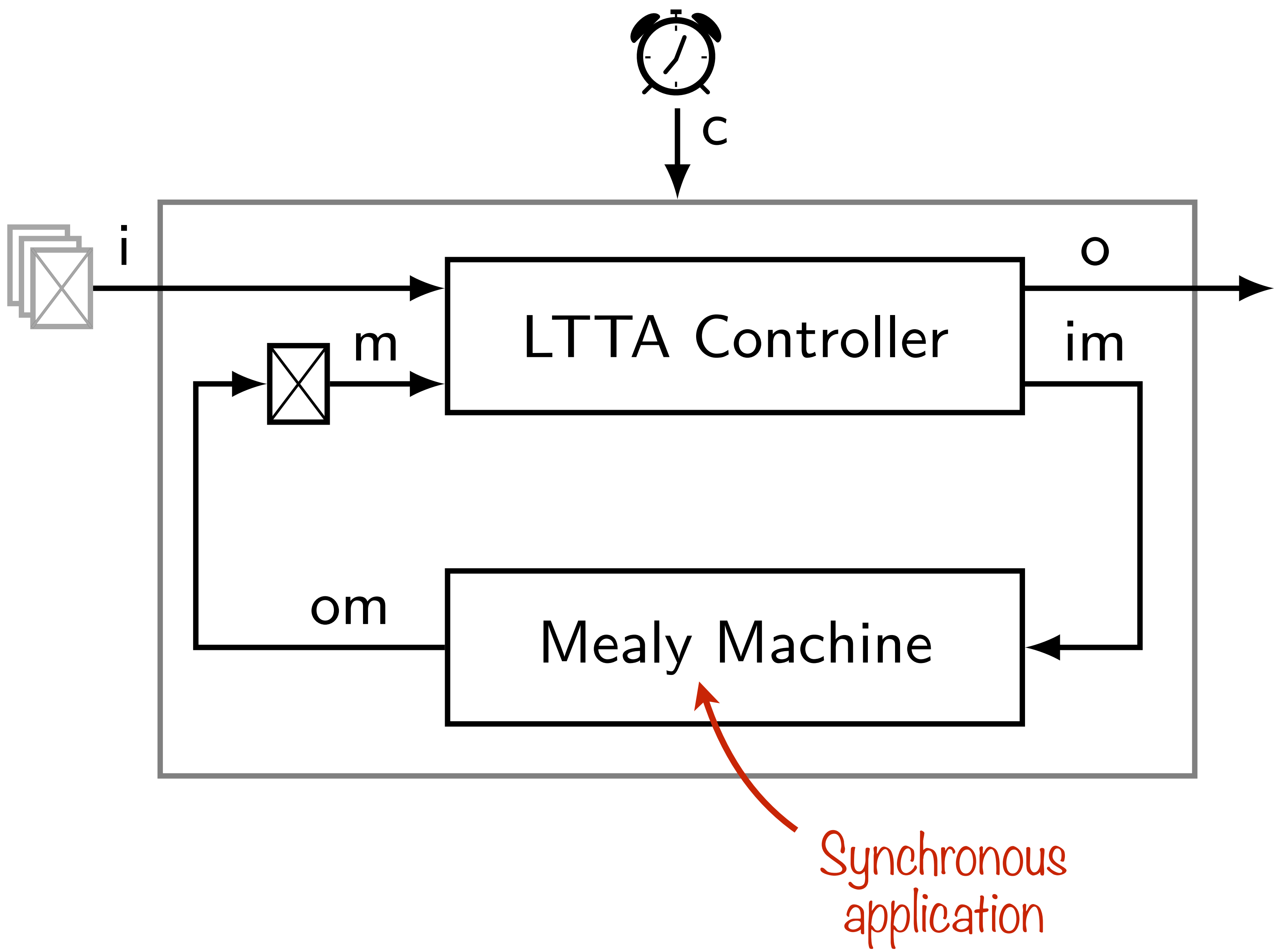
General Framework

Modelling nodes



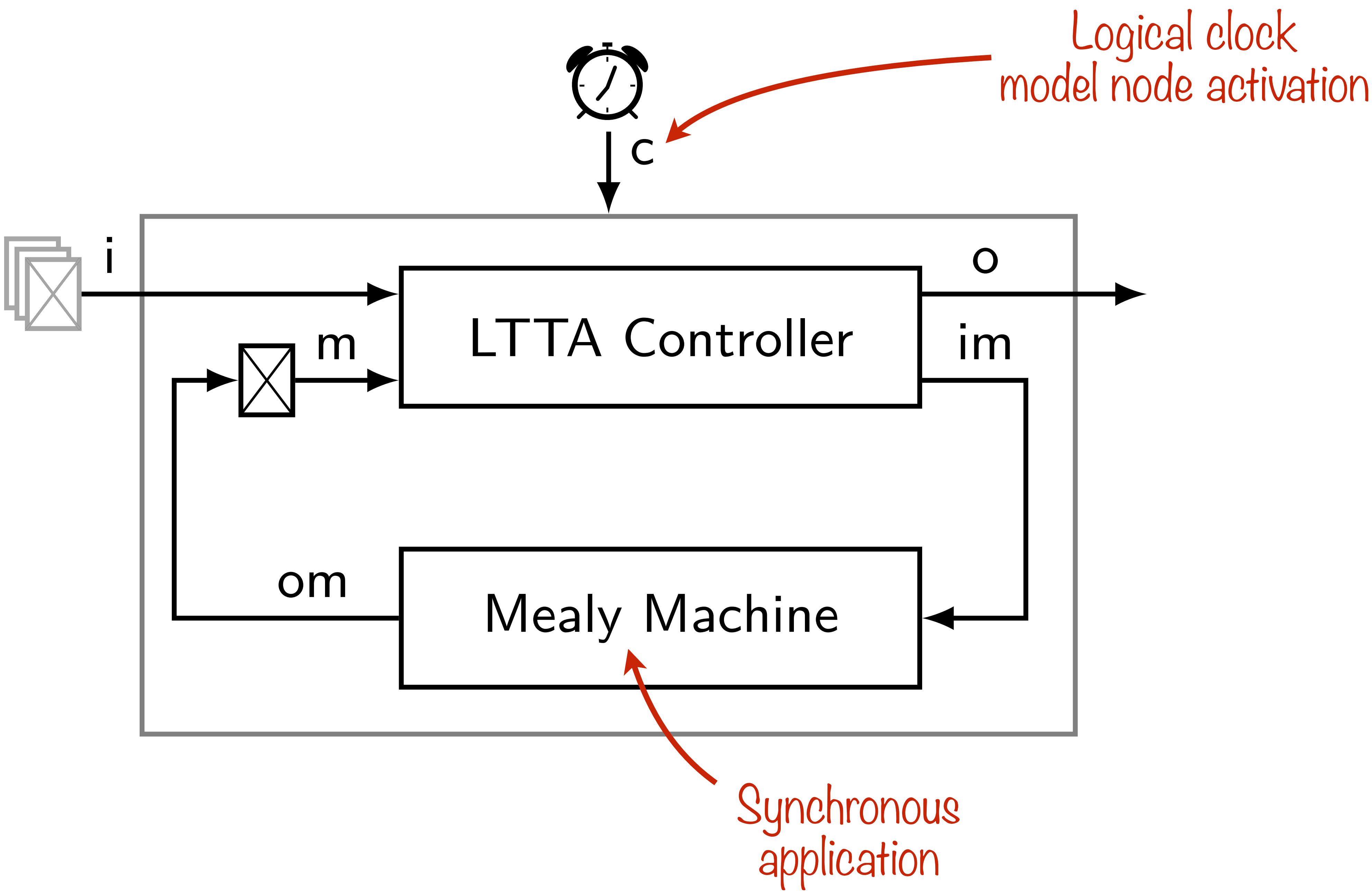
General Framework

Modelling nodes



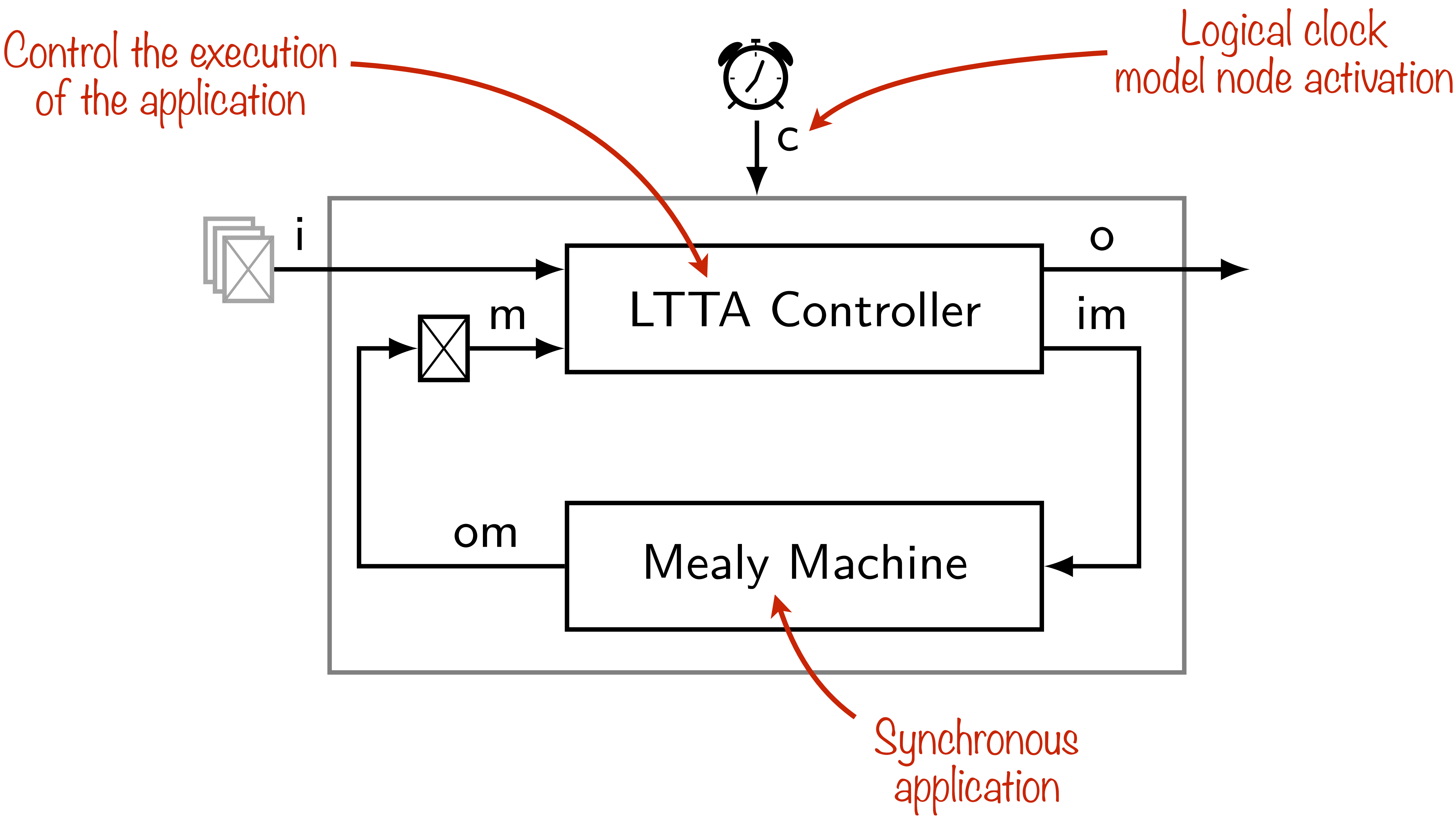
General Framework

Modelling nodes



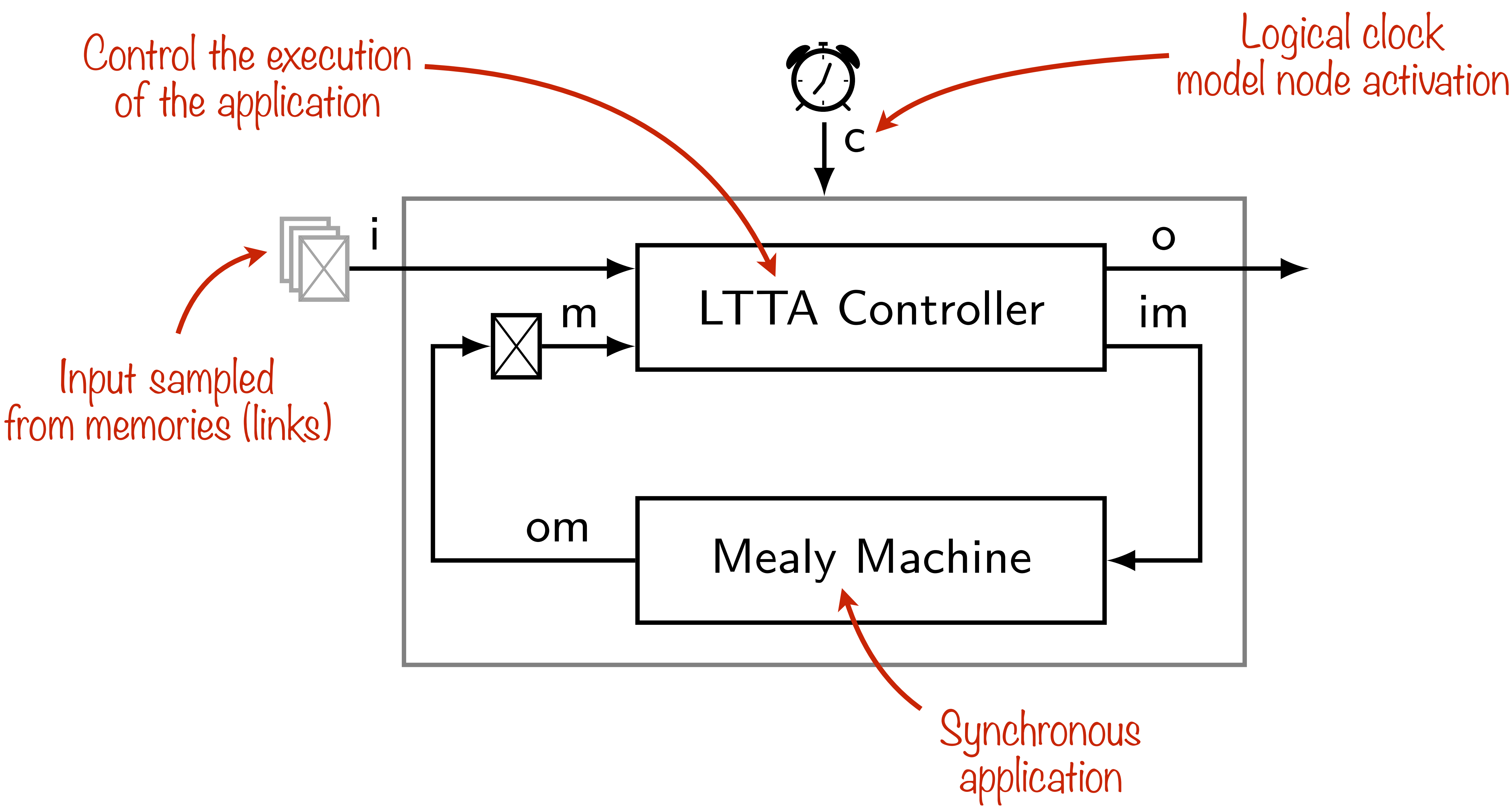
General Framework

Modelling nodes



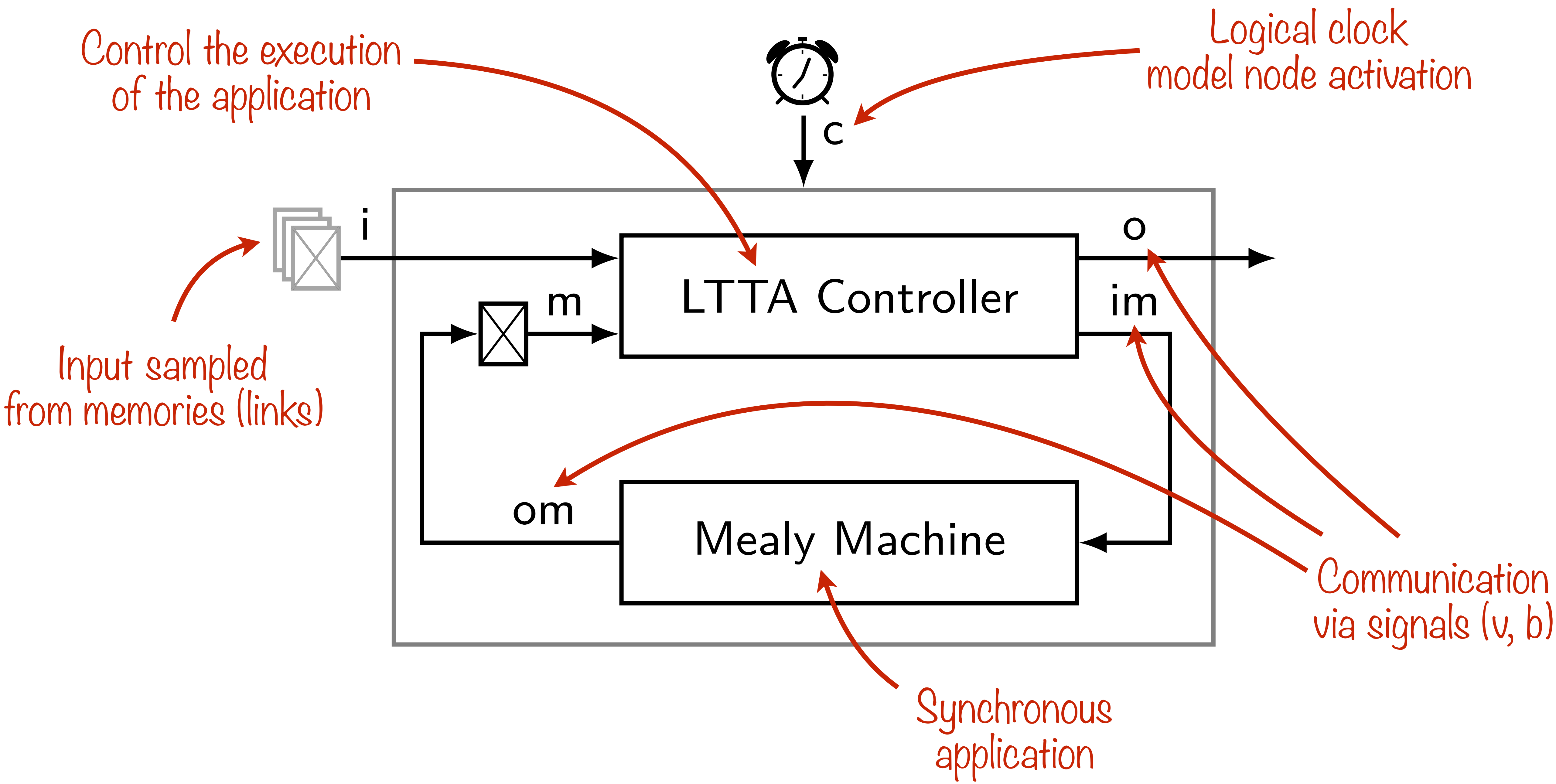
General Framework

Modelling nodes



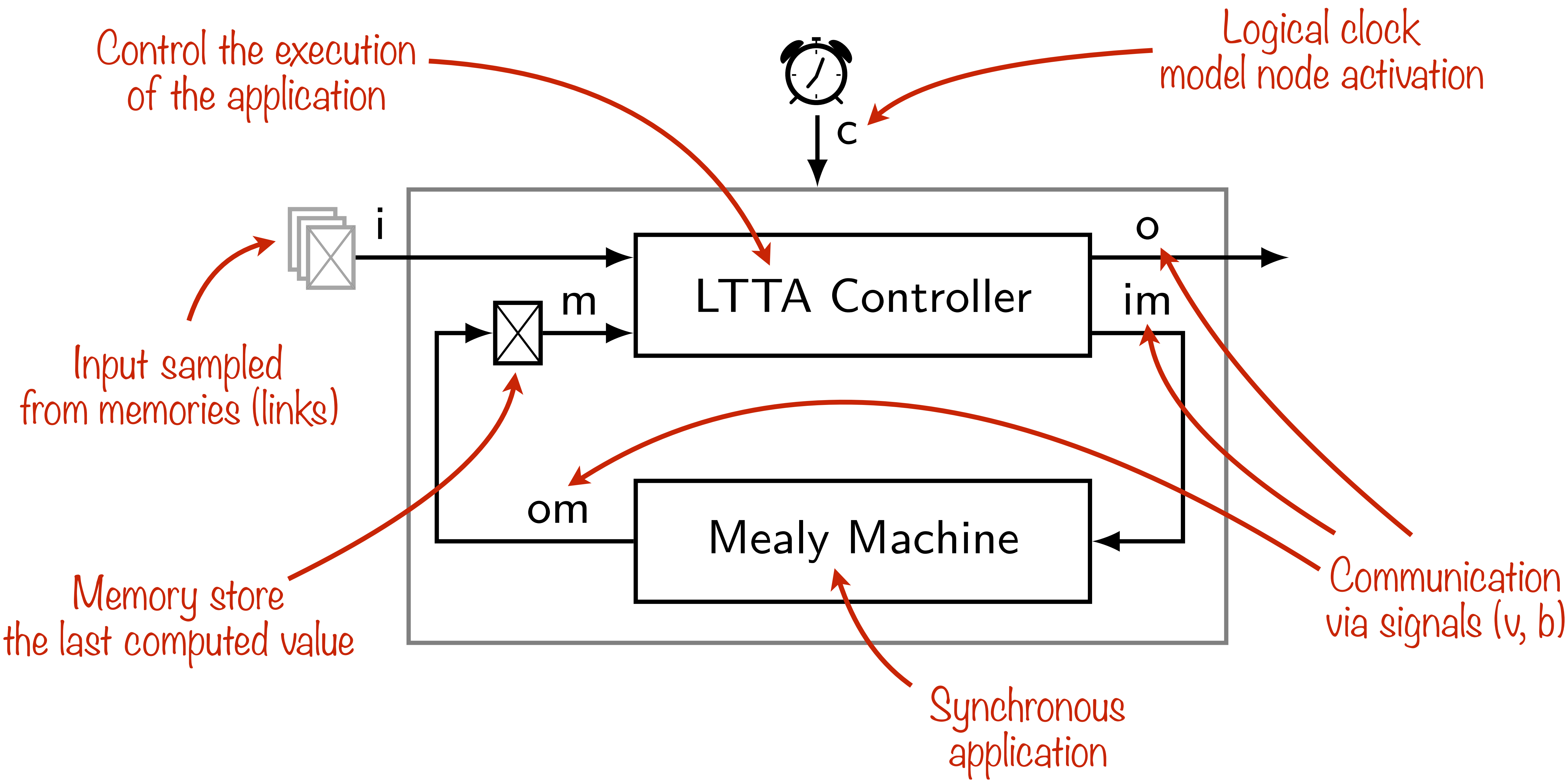
General Framework

Modelling nodes



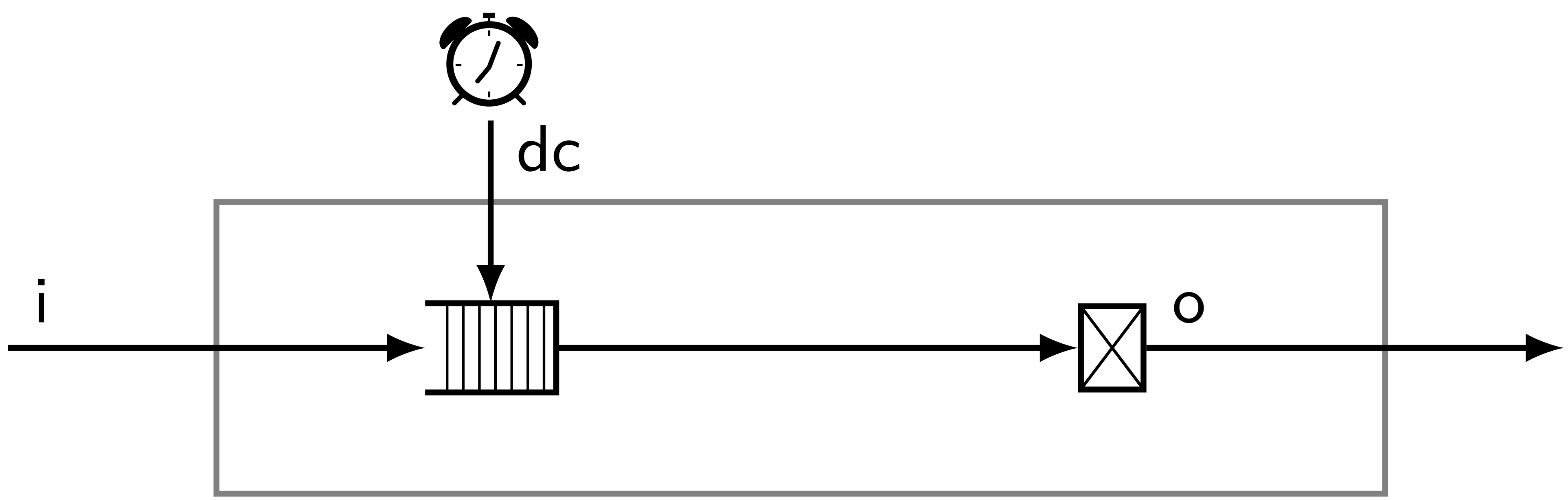
General Framework

Modelling nodes



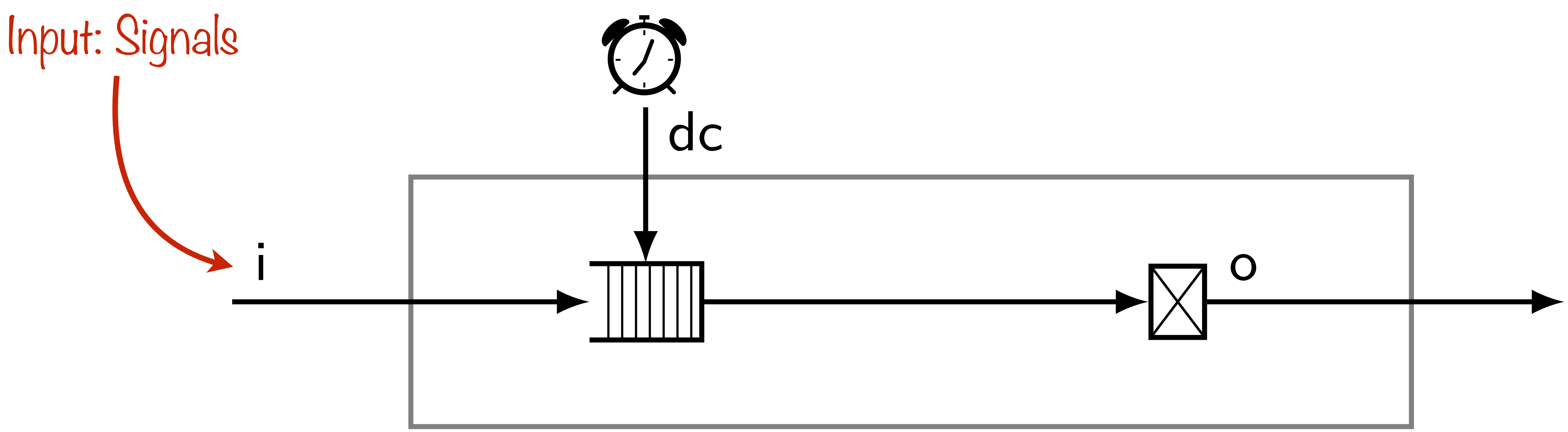
General Framework

Modelling Links



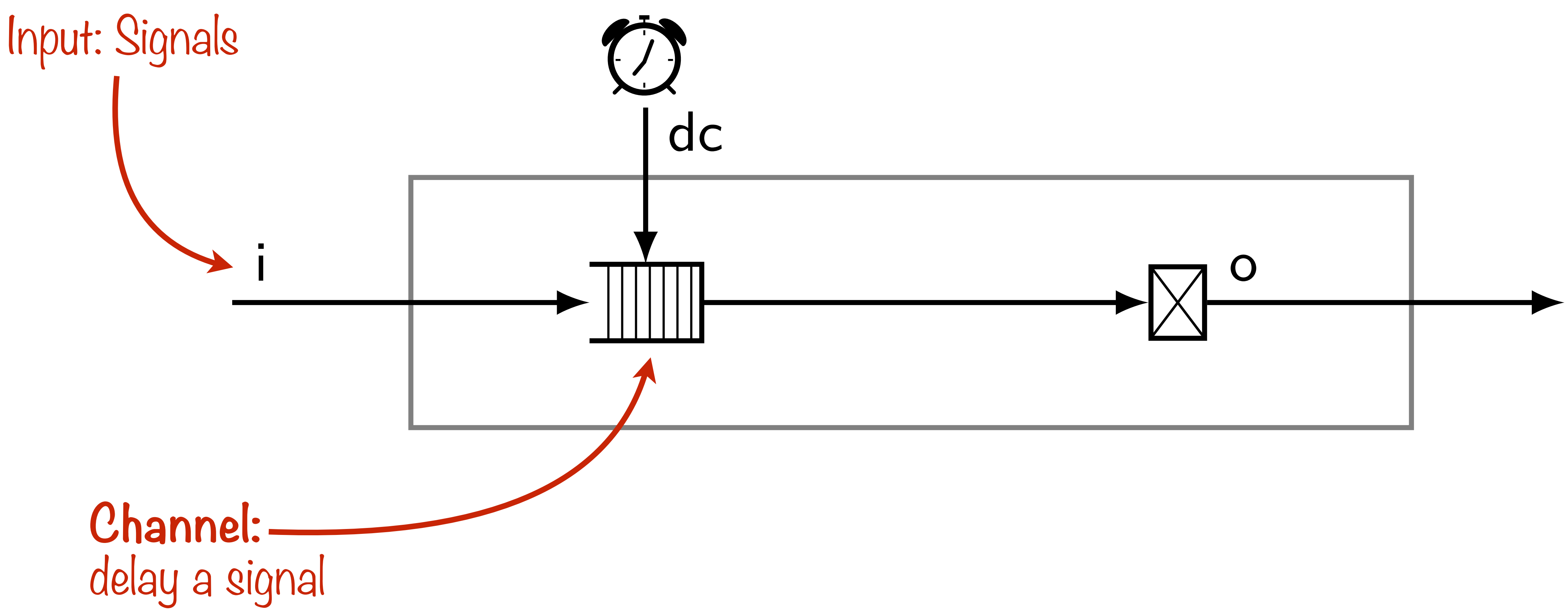
General Framework

Modelling Links



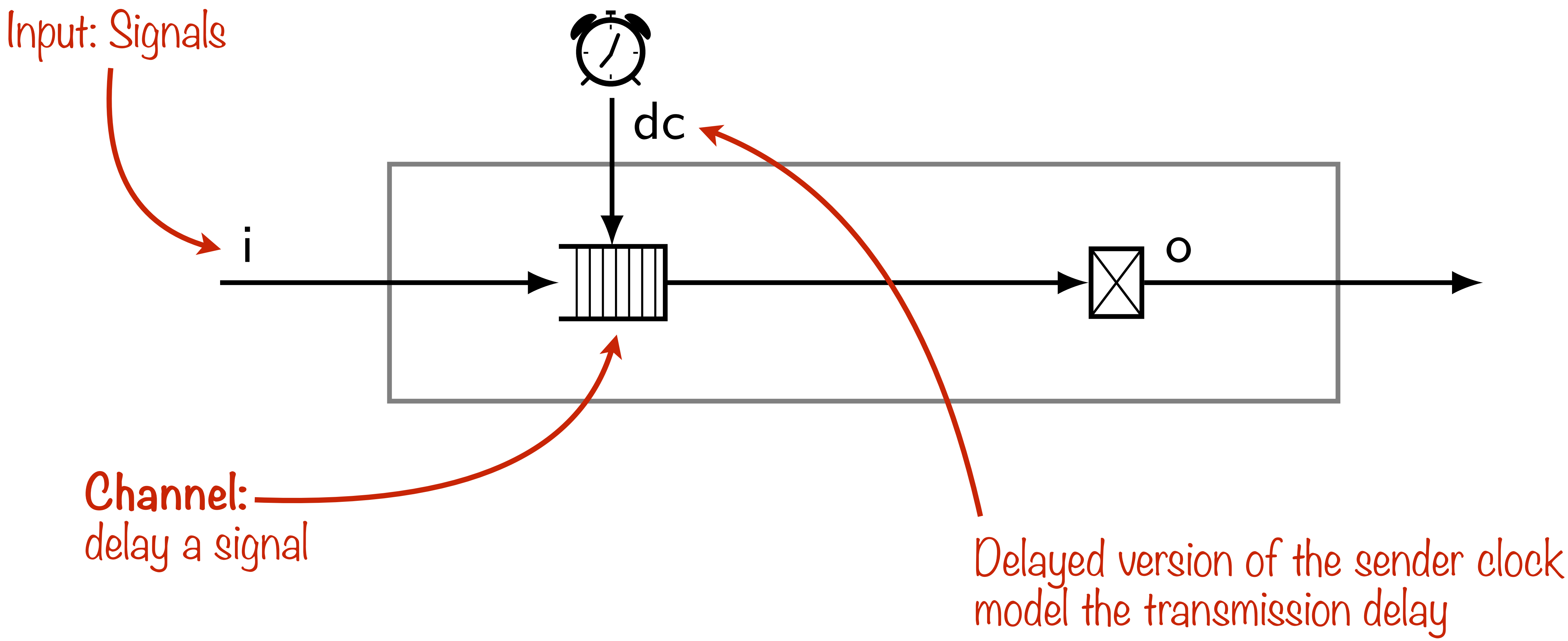
General Framework

Modelling Links



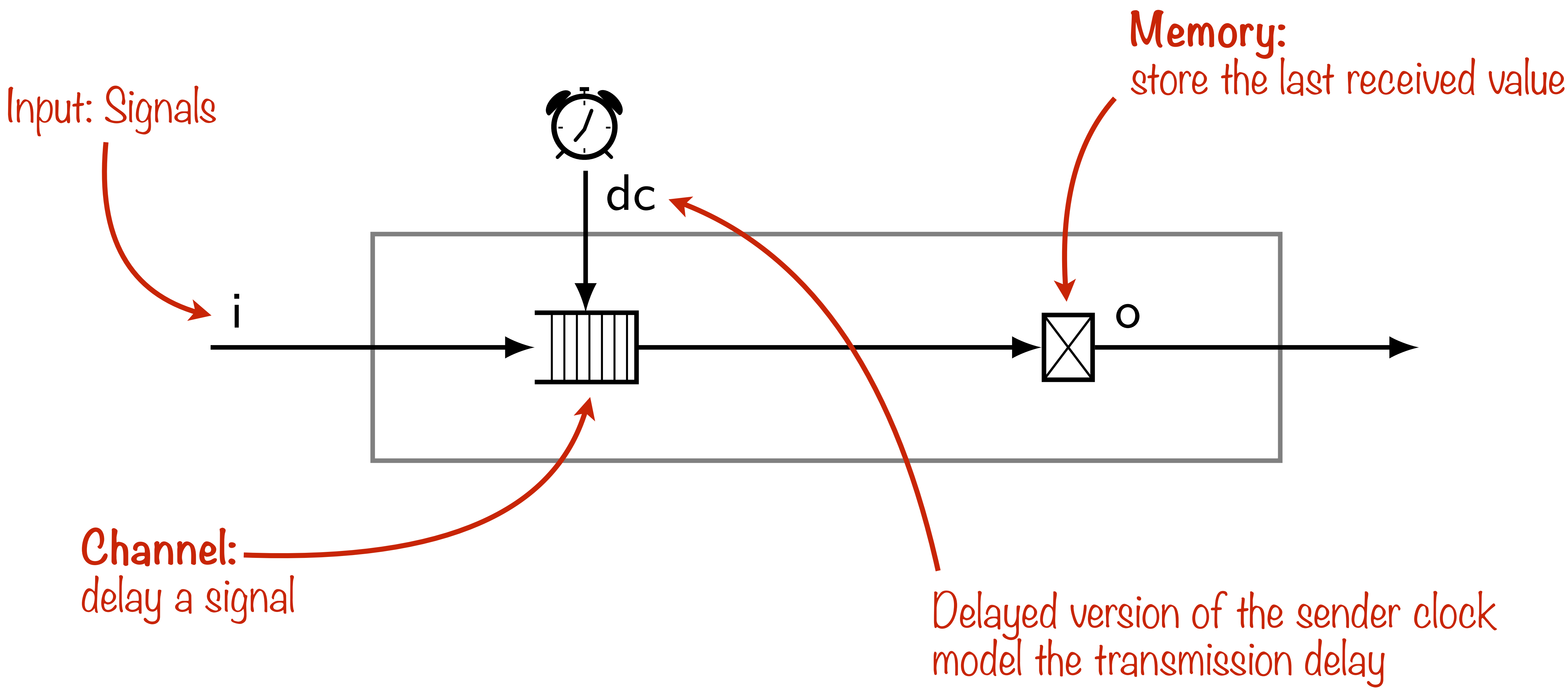
General Framework

Modelling Links



General Framework

Modelling Links

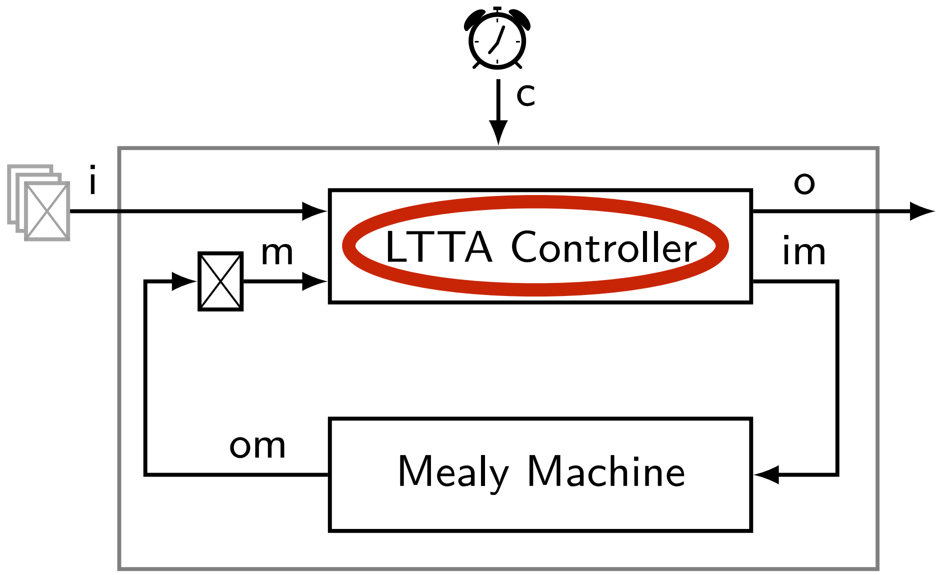


What's next?

Design controllers that ensure a synchronous execution of embedded machines

- **Back-Pressure LTTA**
  [Tripakis et al. 2008]

- **Time-Based LTTA**
  [Caspi, Benveniste 2008]



Outline

1. What is an LTTA?
    1. Quasi-Periodic Architecture
    2. Synchronous Applications

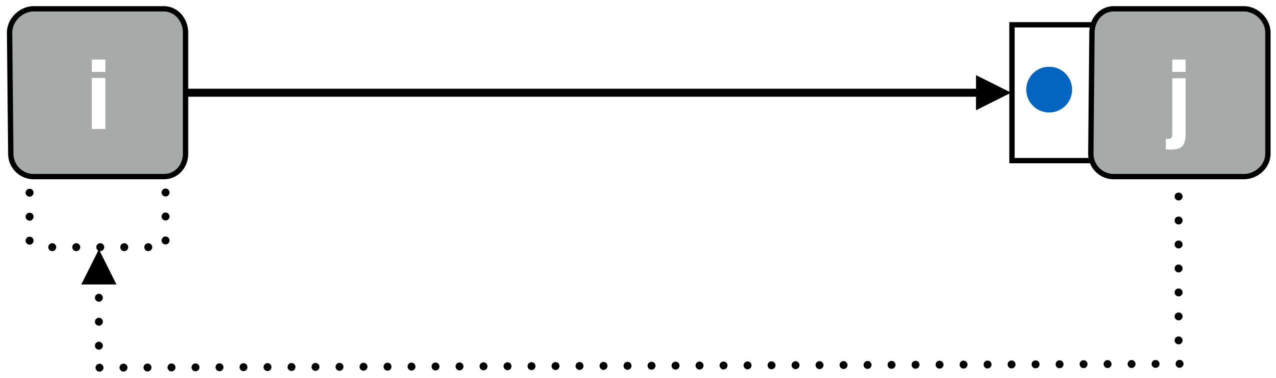
2. General Framework

3. The two protocols
    1. Back-Pressure LTTA
    2. Time-Based LTTA

4. What About Clock Synchronisation?

Back-Pressure Kahn Network

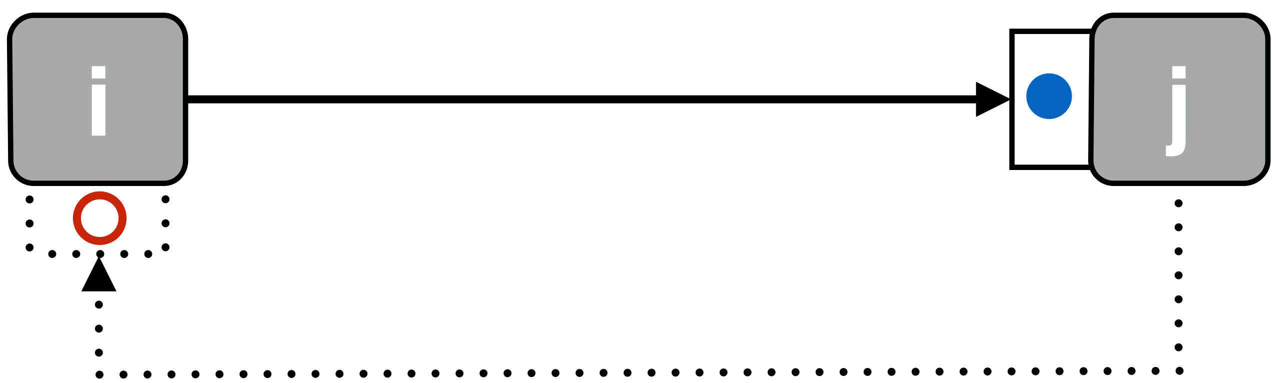
Buffer of Size 1



- Reading from a buffer is acknowledged to the writer
- Nodes alternate between **exec** and **send**

Back-Pressure Kahn Network

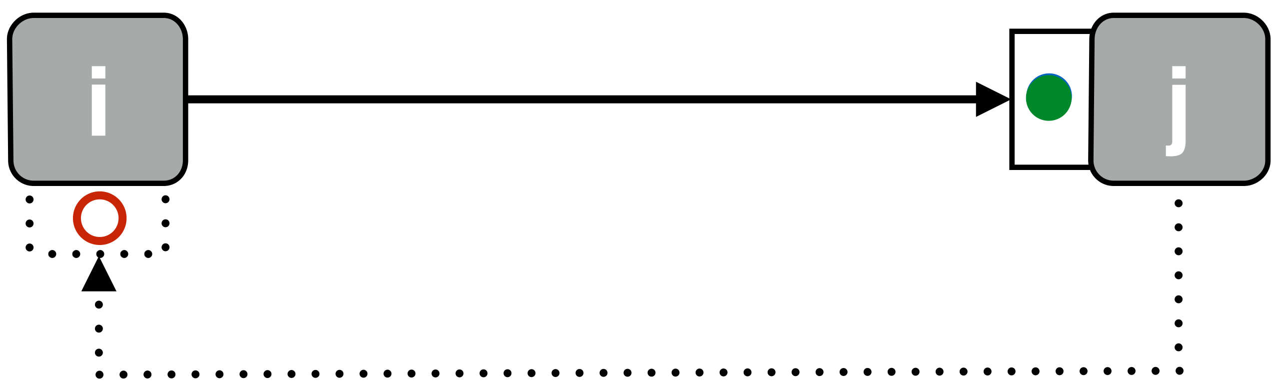
Buffer of Size 1



- Reading from a buffer is acknowledged to the writer
- Nodes alternate between **exec** and **send**

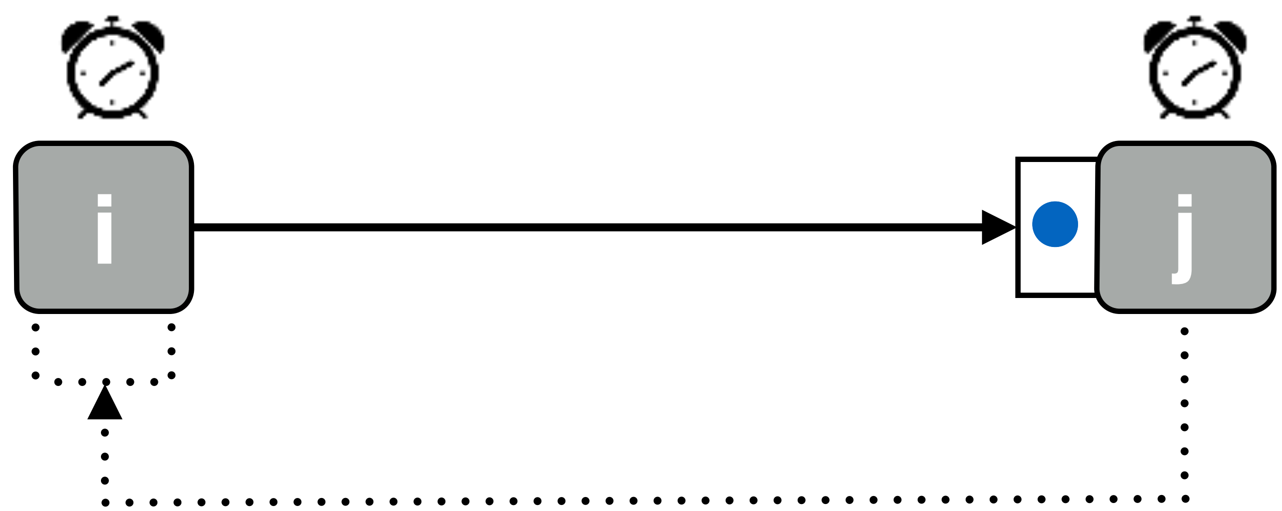
Back-Pressure Kahn Network

Buffer of Size 1



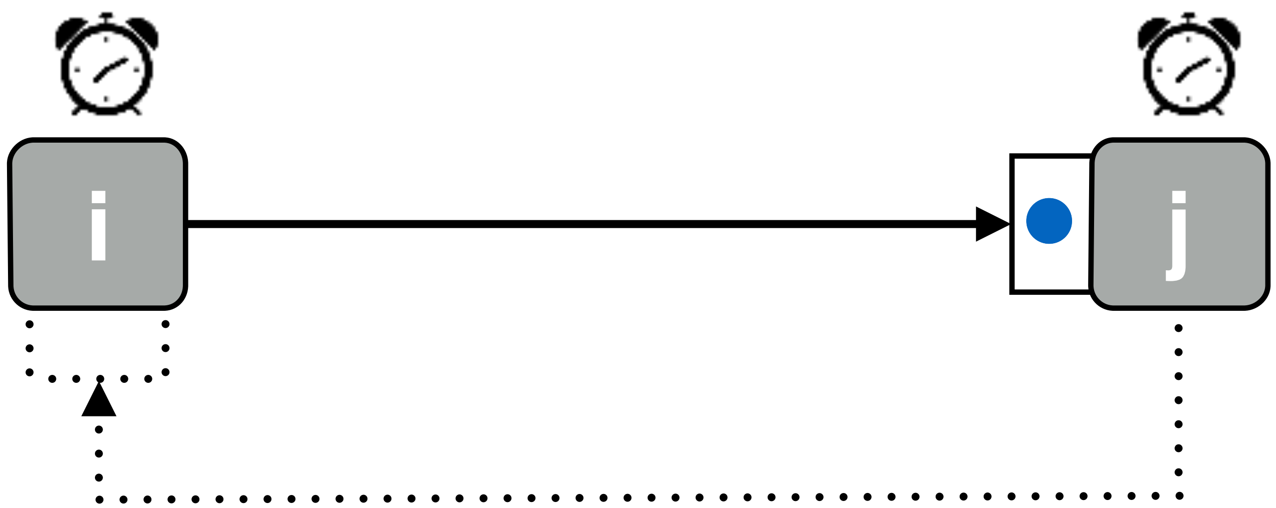
- Reading from a buffer is acknowledged to the writer
- Nodes alternate between **exec** and **send**

Back-Pressure LTTA



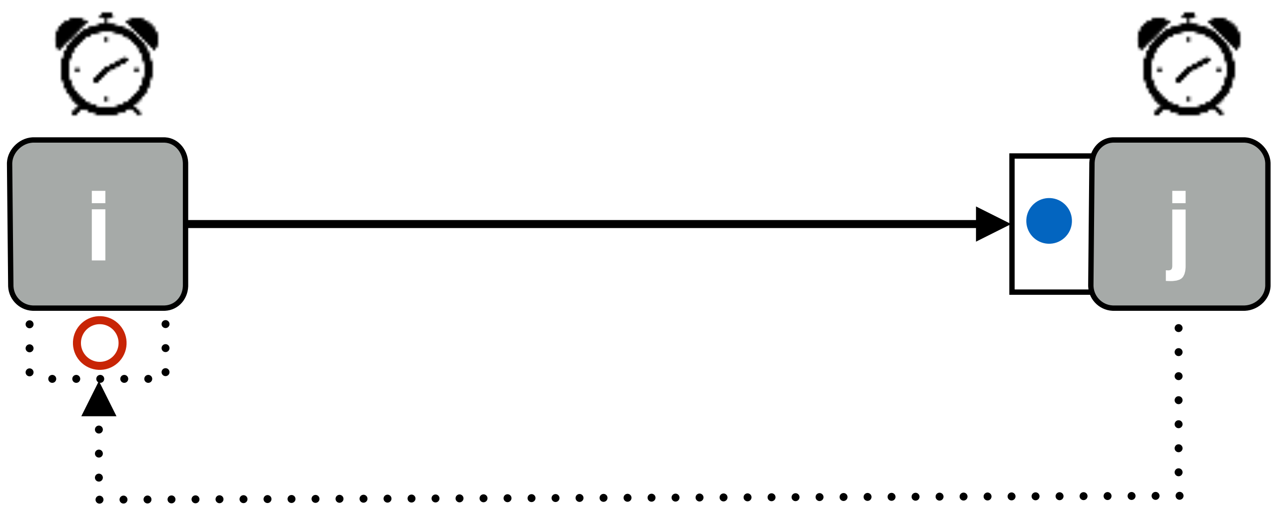
- **Difference:** nodes are triggered by their local clock
- **Idea:** adding skipping mechanism

Back-Pressure LTTA



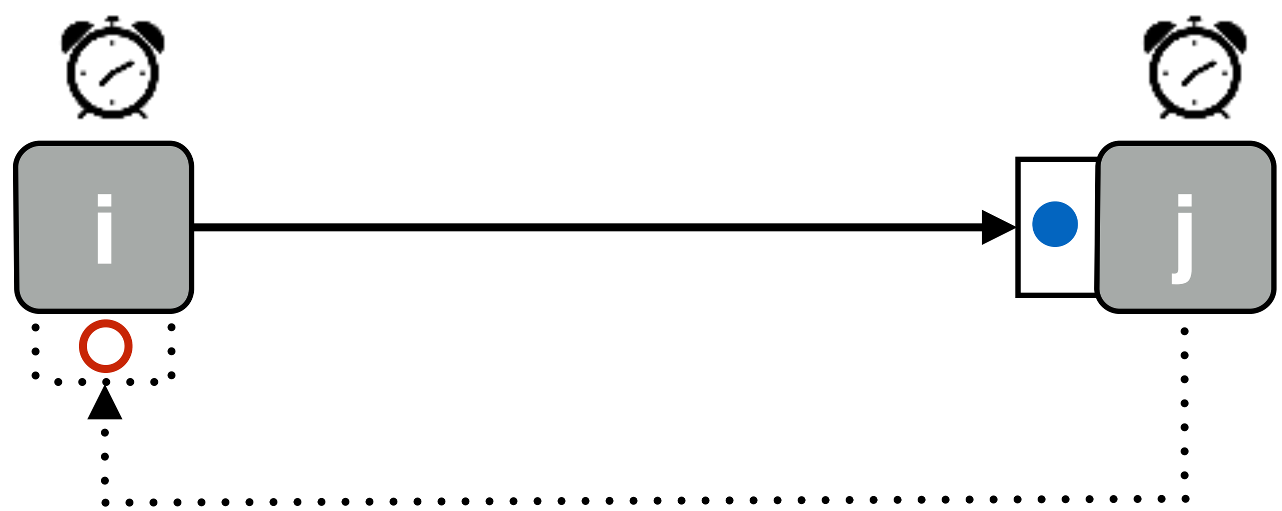
- **Difference:** nodes are triggered by their local clock
- **Idea:** adding skipping mechanism

Back-Pressure LTTA



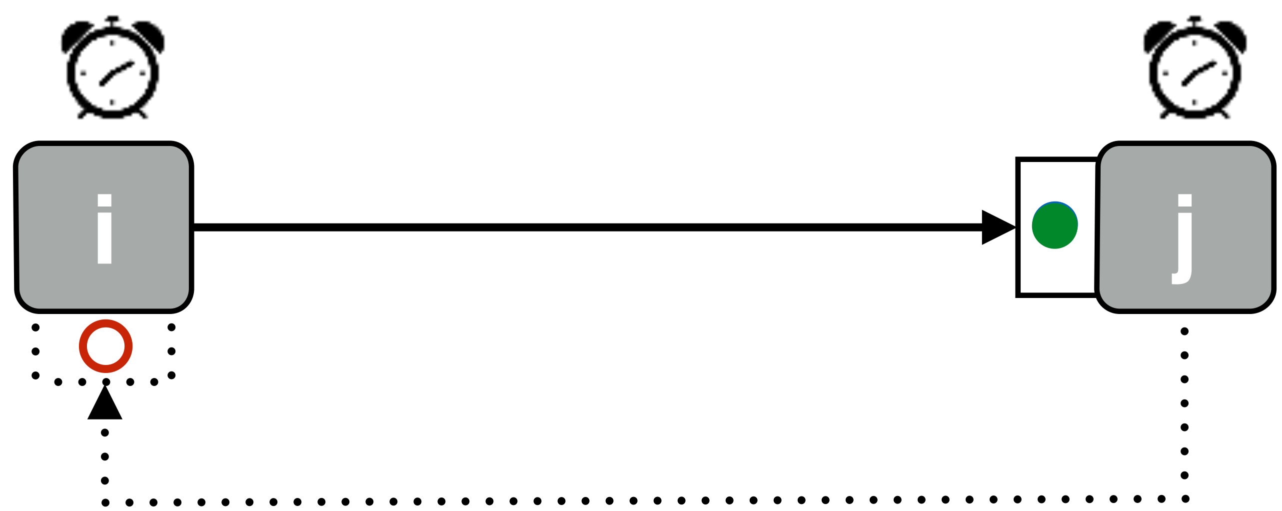
- **Difference:** nodes are triggered by their local clock
- **Idea:** adding skipping mechanism

Back-Pressure LTTA



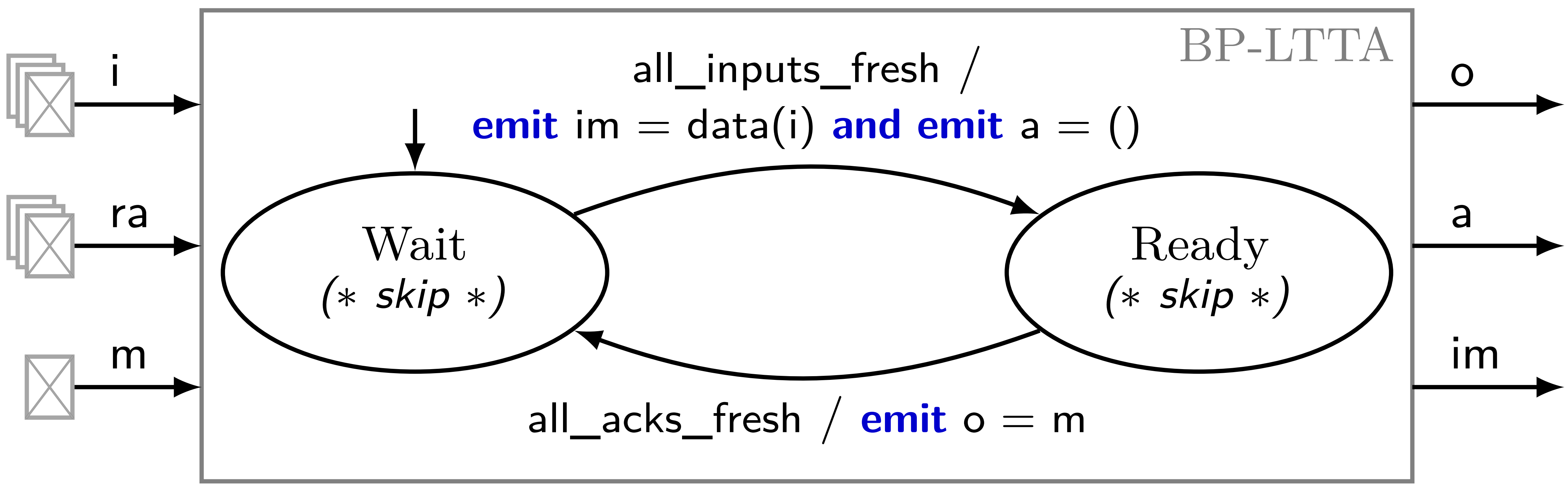
- **Difference:** nodes are triggered by their local clock
- **Idea:** adding skipping mechanism

Back-Pressure LTTA



- **Difference:** nodes are triggered by their local clock
- **Idea:** adding skipping mechanism

Back-Pressure LTTA



Back-Pressure LTTA

- **Theorem 1:**
  Composition of the controller and the embedded machine is always well-defined (no cycle)

- **Theorem 2:**
  Back-pressure LTTA preserves the Kahn semantics of the embedded application
  (forget the skips)

- **Theorem 3:**
  The worst case throughput is: $1/\lambda_{\mathrm{BP}} = 2(T_{\max} + \tau_{\max})$

Time-Based LTTA

- **Problem:** Back-pressure multiplies the number of messages and memories, and *blocks if a node crashes*

- **Idea:** Replace back-pressure by waiting, using timing characteristics of the architecture

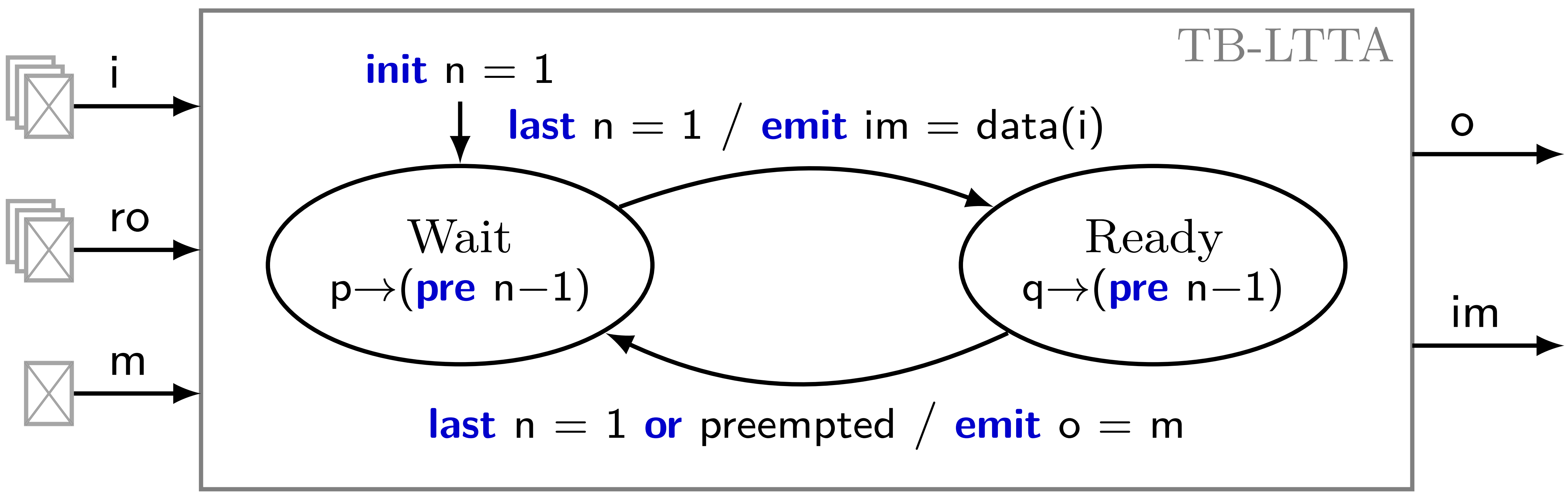
- **First solution:** [Caspi, Benveniste 2008]
Slow down the nodes to mimic a synchronous architecture, global synchronisation

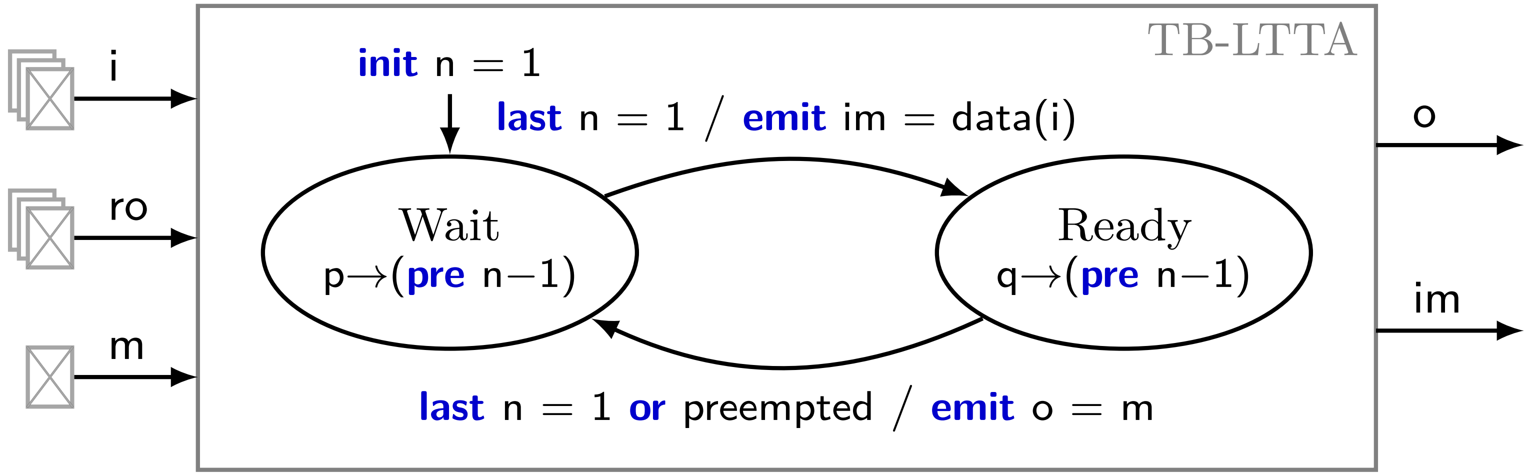
- **Our proposal:** Relax broadcast assumption,
localise synchronisations

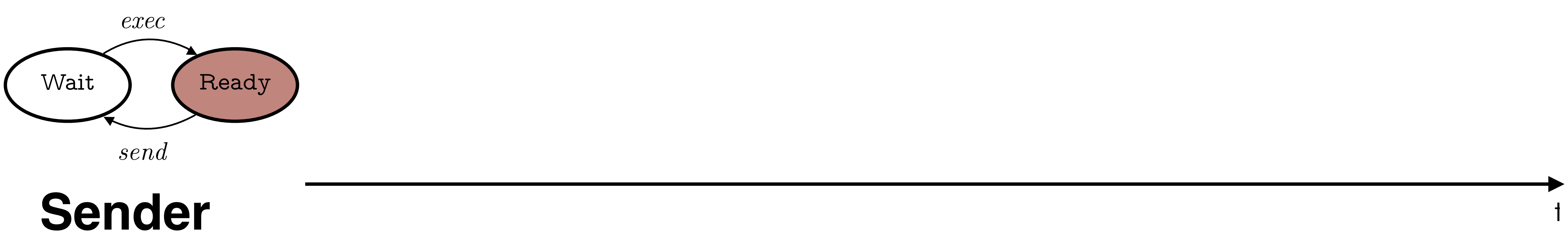
Time-Based LTTA

- Nodes alternate between **exec** and **send**
- Sender sees publication of all receivers
- **Idea:** At some point, a node can be sure that:
  - the last sent data has been read
  - a fresh value is available in the memory

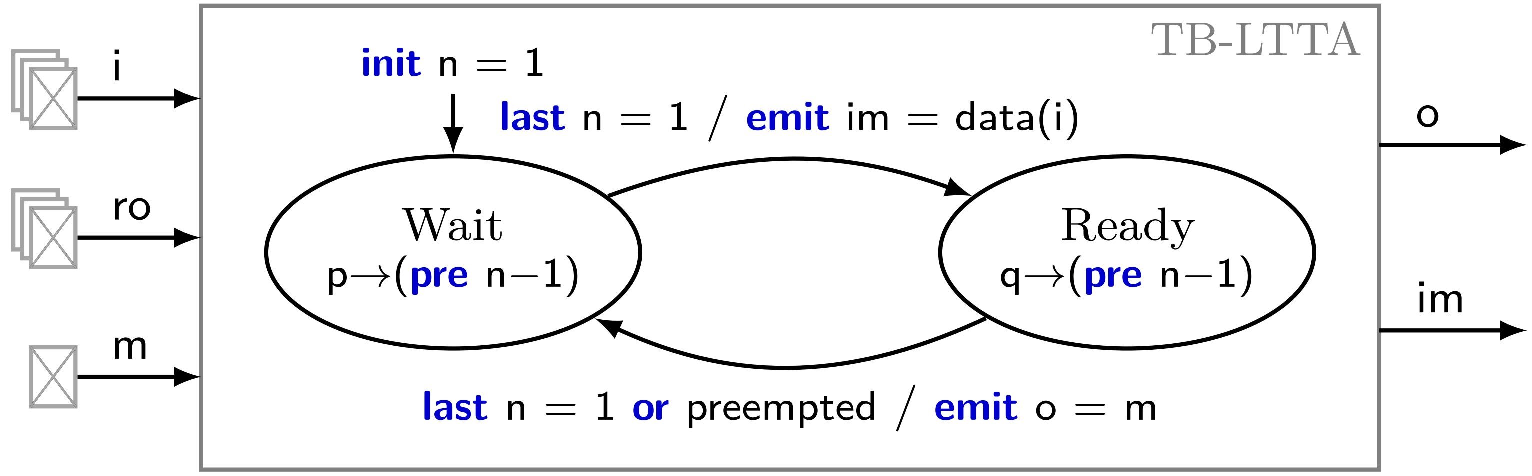
Time-Based LTTA

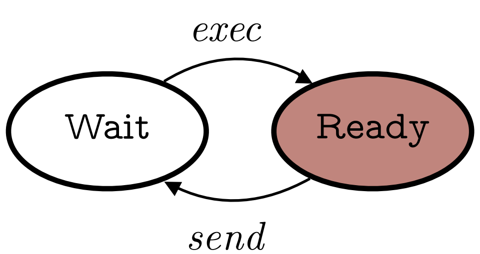



TB-LTTA
i
ro
m
init n = 1
last n = 1 / emit im = data(i)
Wait
p→(pre n−1)
Ready
q→(pre n−1)
last n = 1 or preempted / emit o = m
o
im

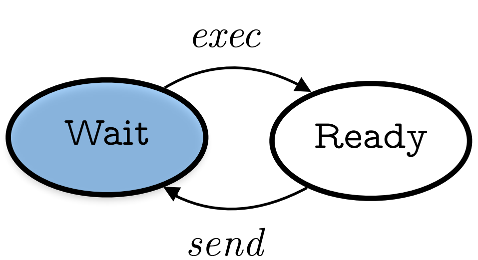

exec
Wait
Ready
send
Sender
t


exec
Wait
Ready
send
Receiver
t

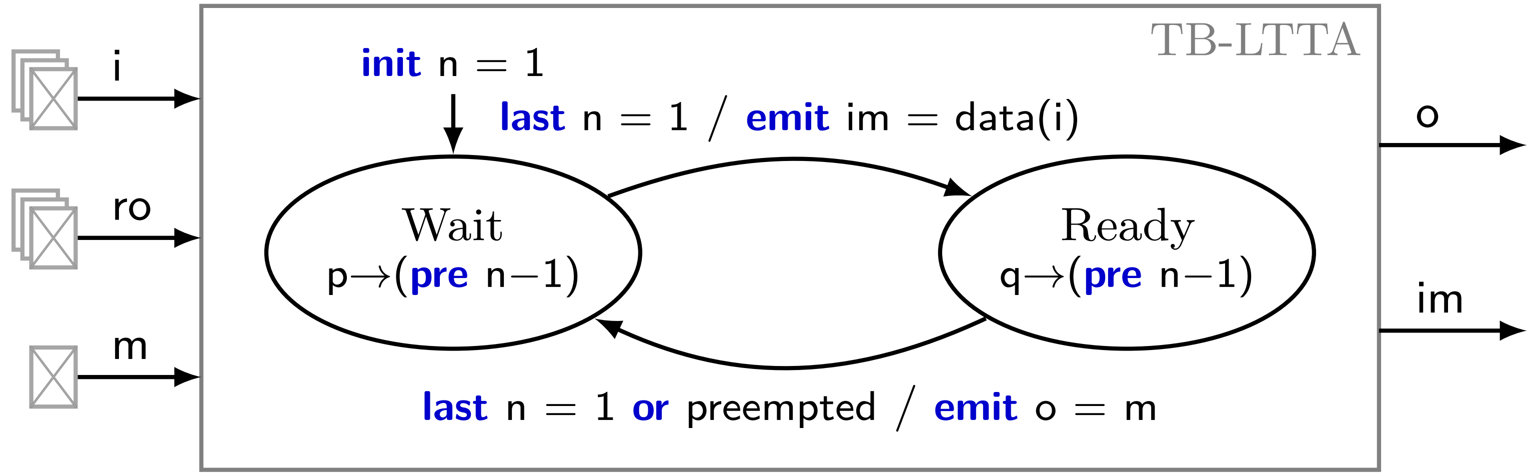




Sender

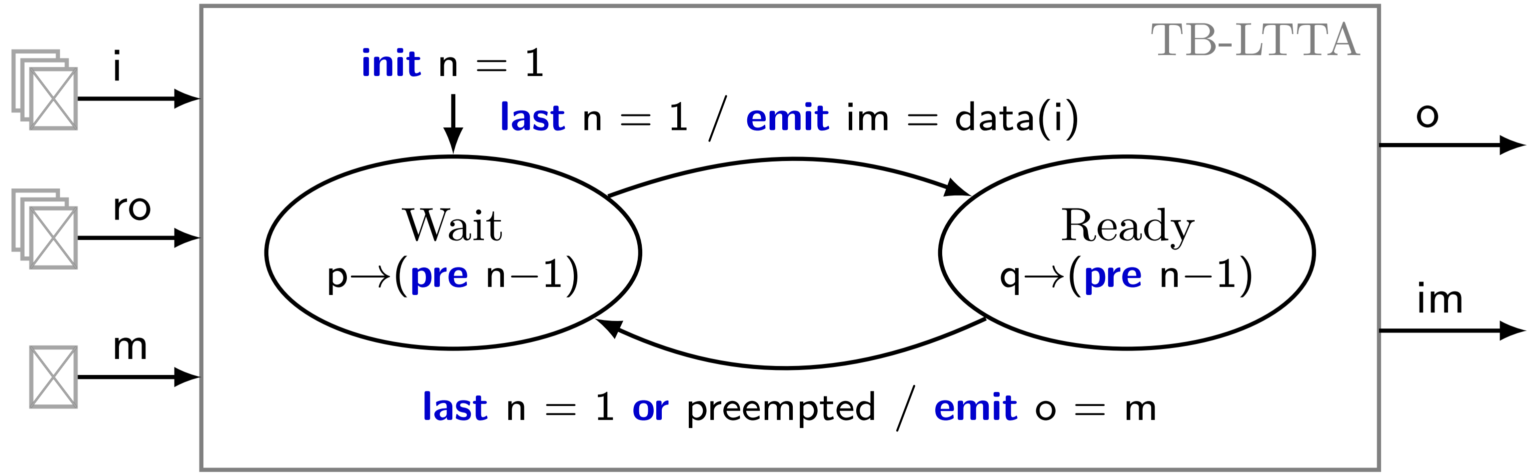


Receiver



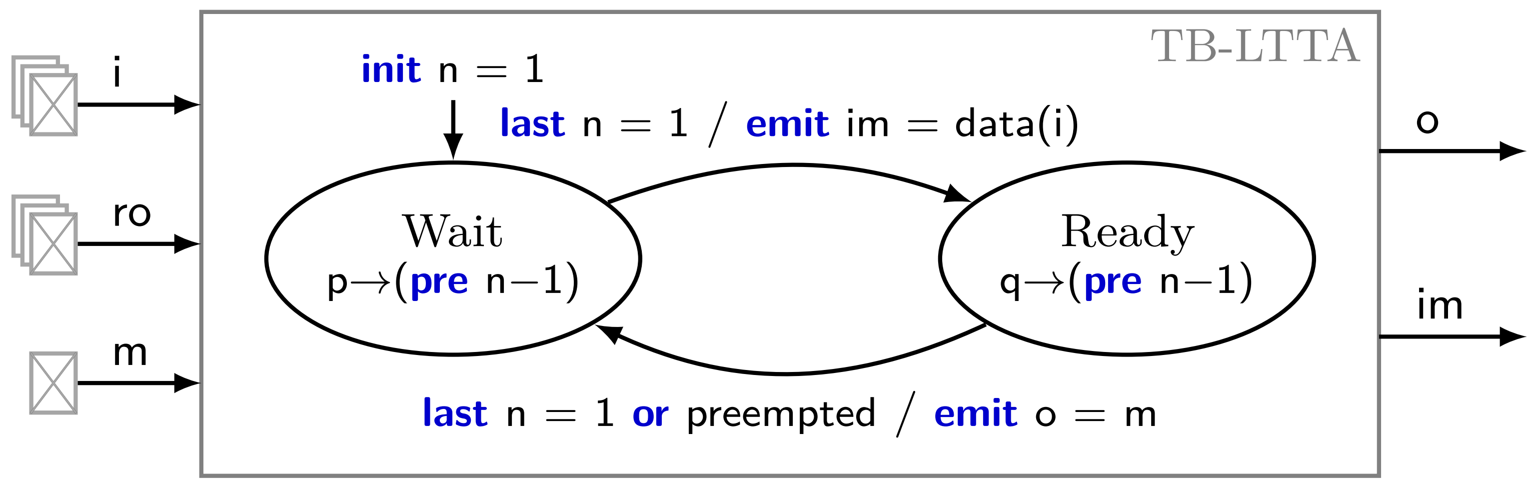
Sender

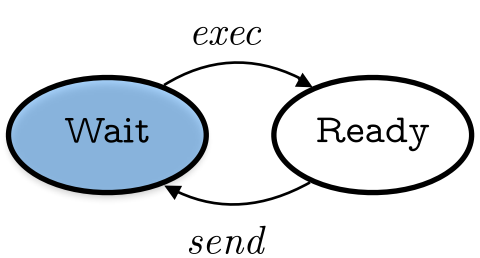
Receiver



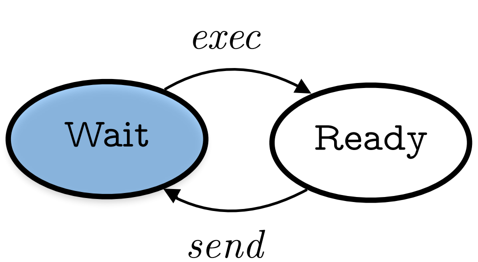
Sender

Receiver

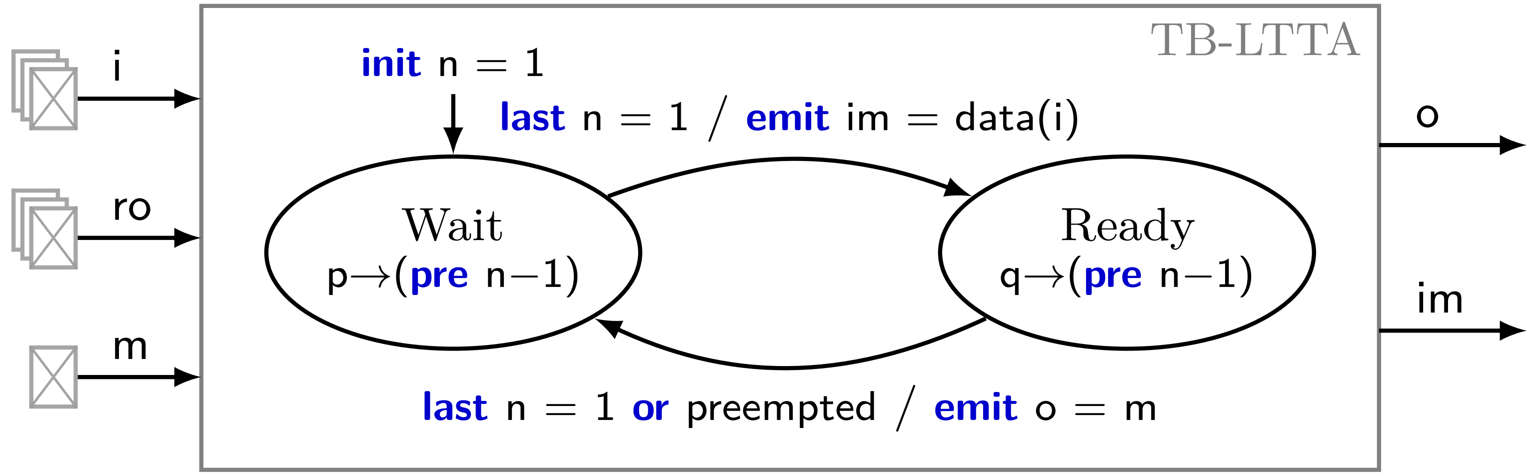


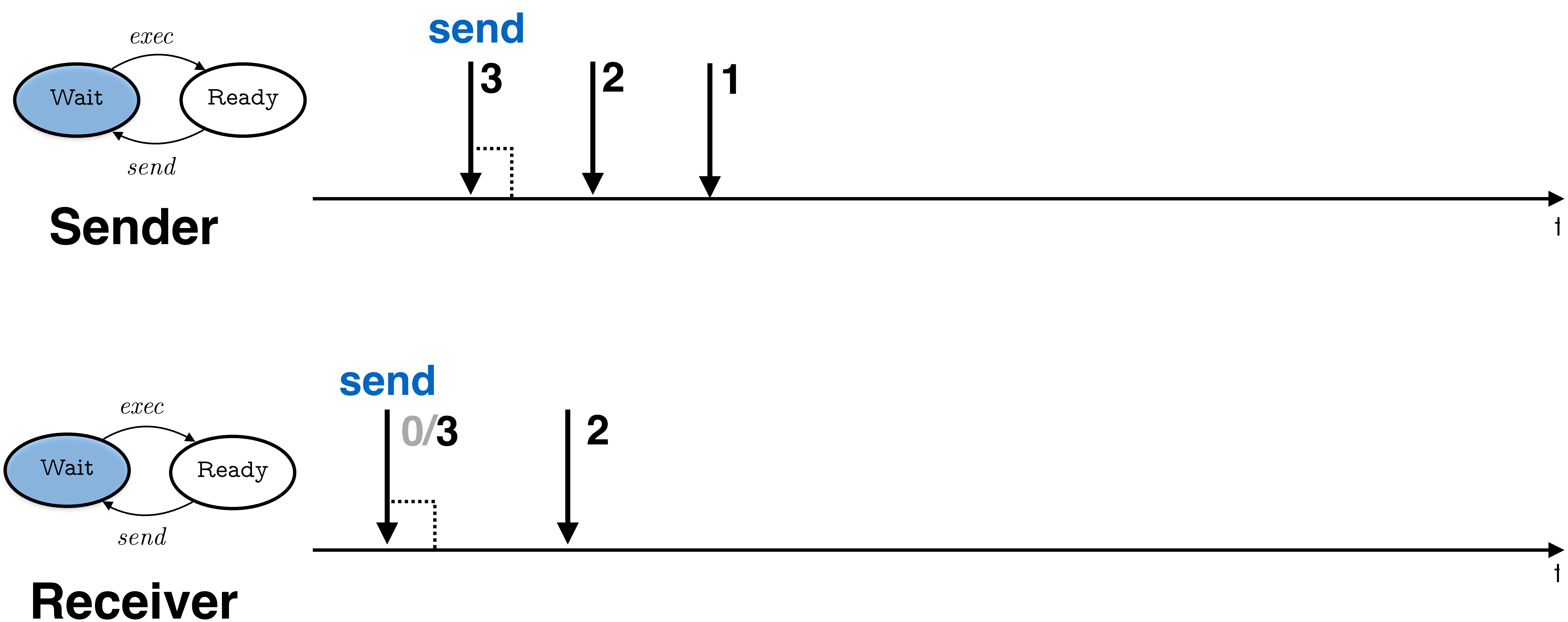


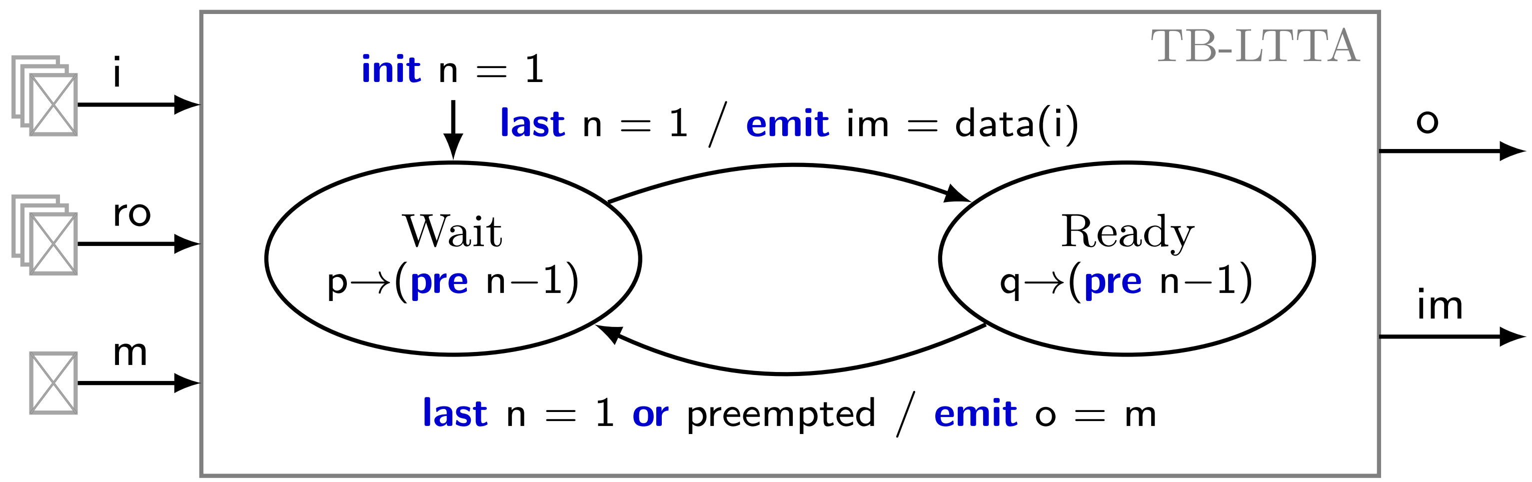
Sender

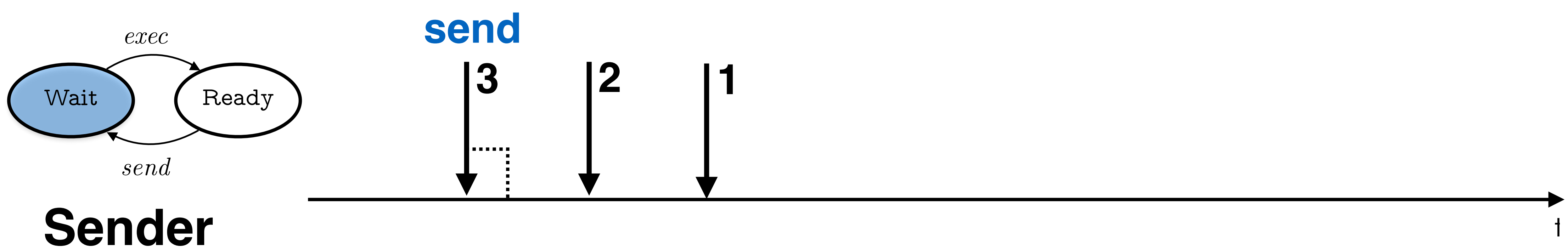


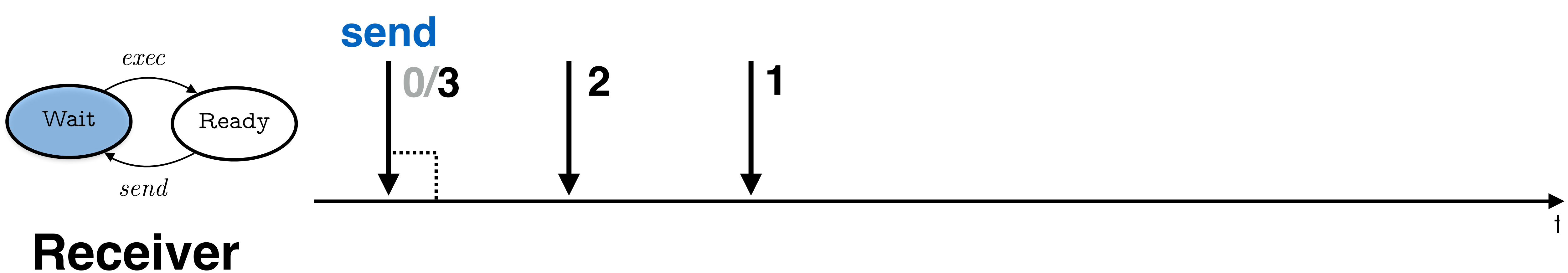
Receiver

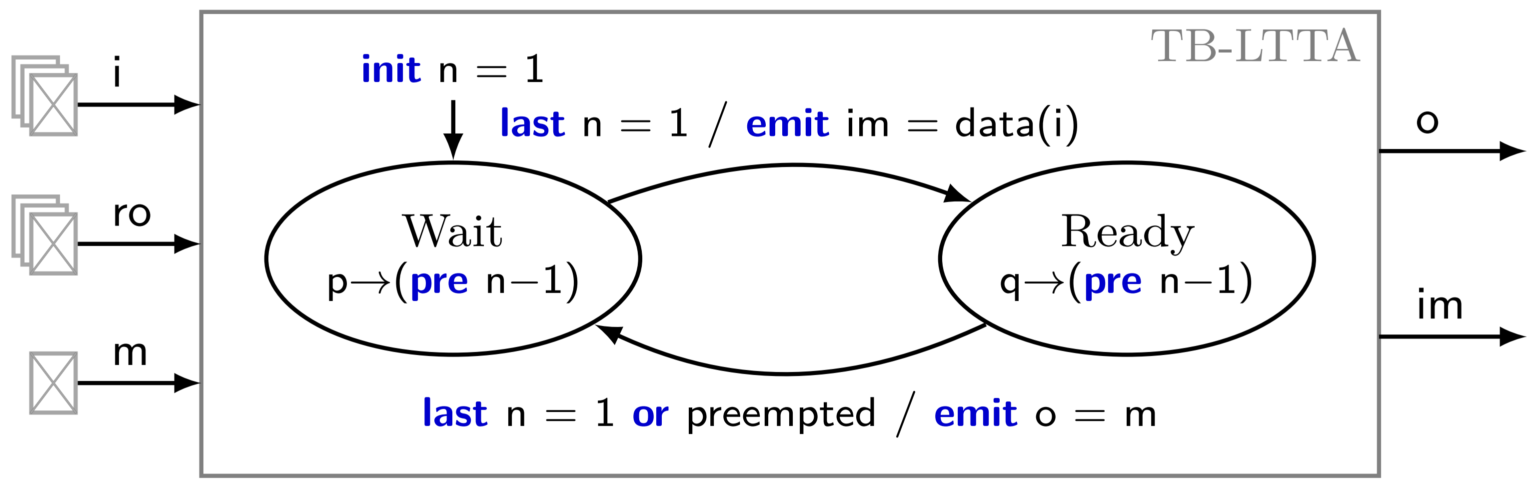

TB-LTTA
i
ro
m
o
im
init n = 1
last n = 1 / emit im = data(i)
Wait
p→(pre n−1)
Ready
q→(pre n−1)
last n = 1 or preempted / emit o = m

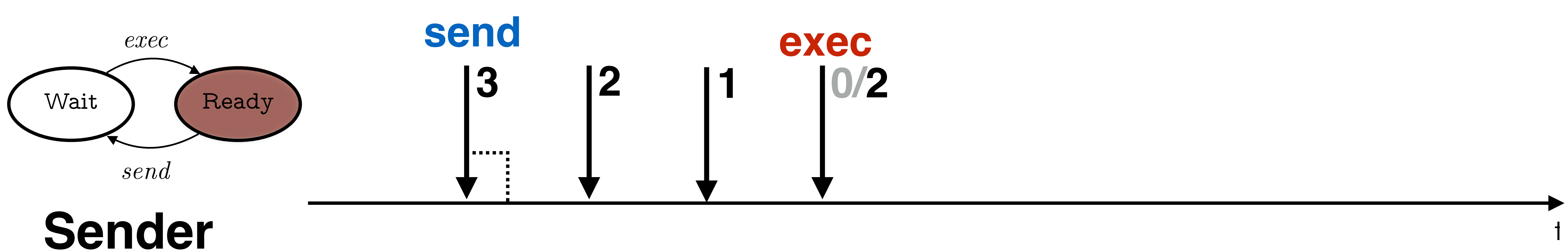

exec
Wait
Ready
send
Sender
send
3
2
1
t
exec
Wait
Ready
send
Receiver
send
0/3
2
t

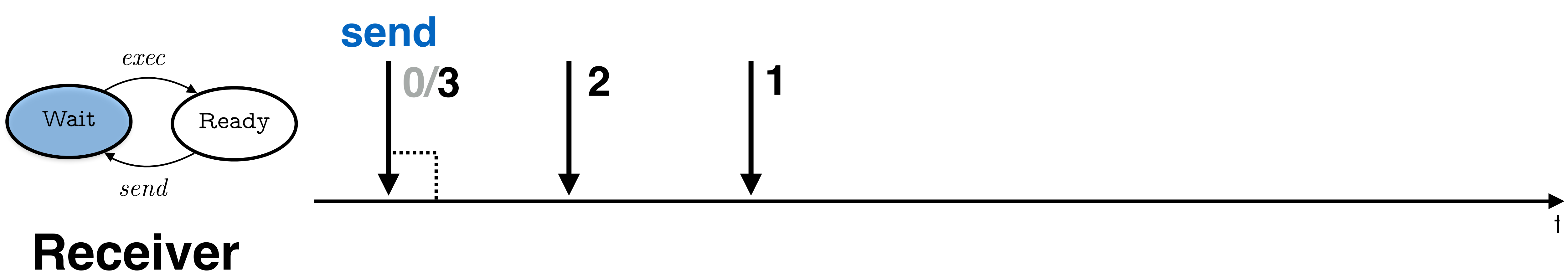

TB-LTTA
i
ro
m
o
im
init n = 1
last n = 1 / emit im = data(i)
Wait
p→(pre n−1)
Ready
q→(pre n−1)
last n = 1 or preempted / emit o = m

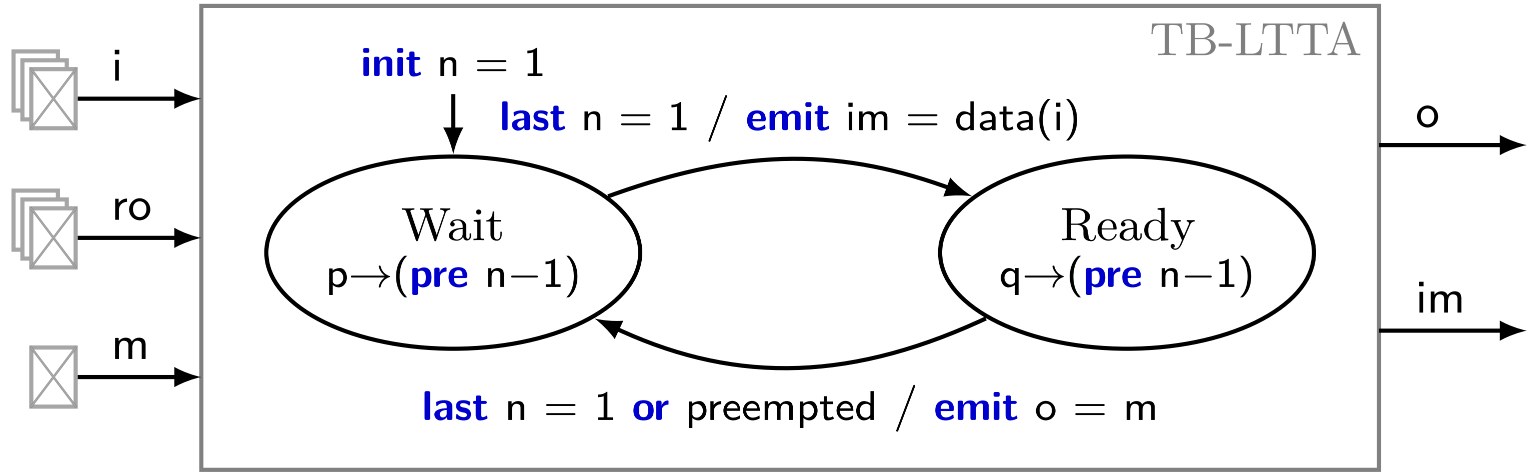

exec
Wait
Ready
send
Sender
send
3
2
1
t

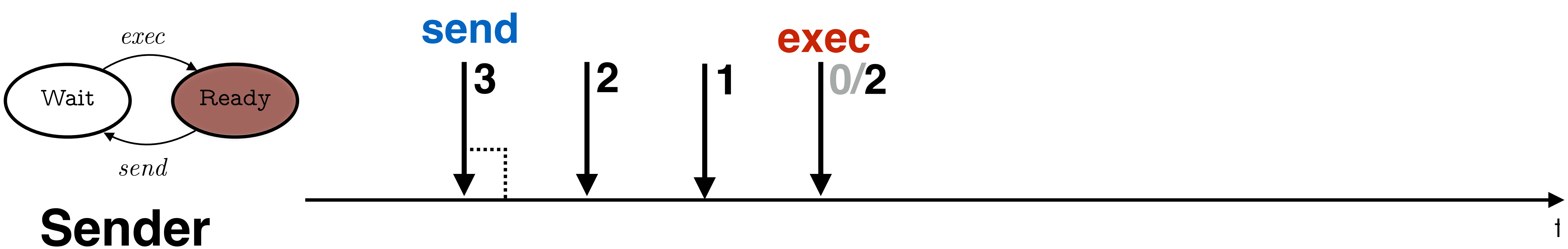

exec
Wait
Ready
send
Receiver
send
0/3
2
1
t

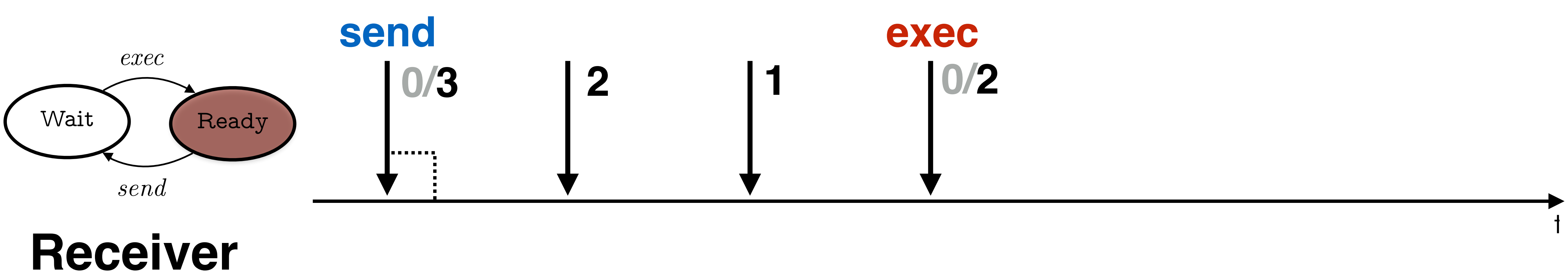

TB-LTTA
i
ro
m
o
im
init n = 1
last n = 1 / emit im = data(i)
Wait
p→(pre n−1)
Ready
q→(pre n−1)
last n = 1 or preempted / emit o = m

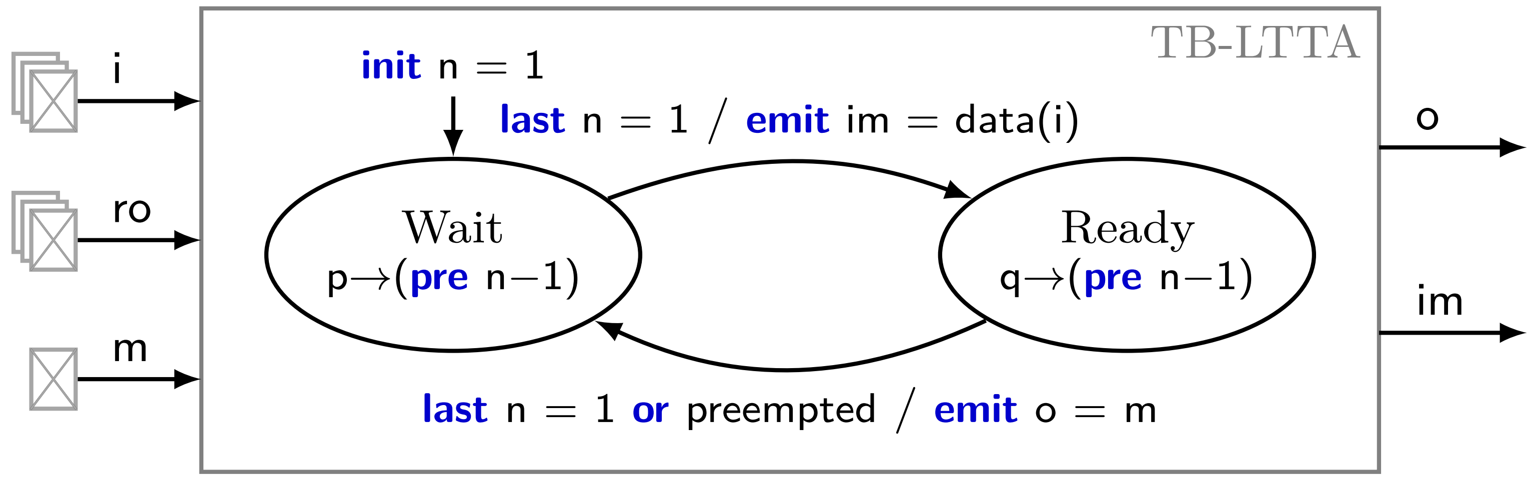

exec
Wait
Ready
send
Sender
send
3
2
1
exec
0/2
t

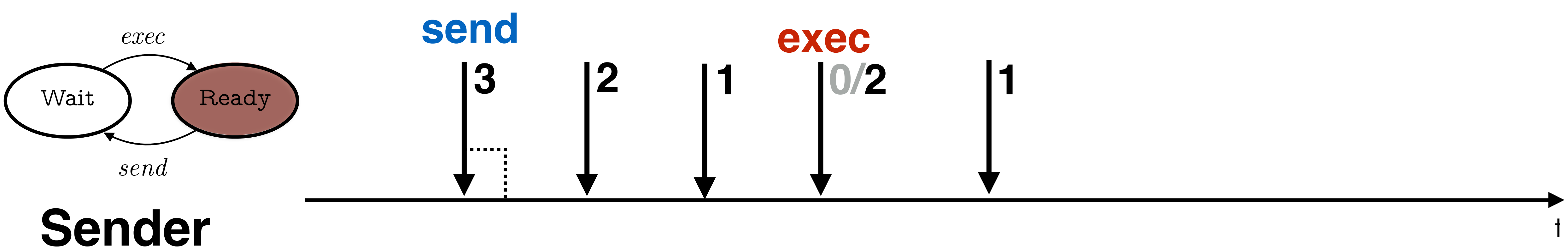

exec
Wait
Ready
send
Receiver
send
0/3
2
1
t


TB-LTTA
i
ro
m
o
im
init n = 1
last n = 1 / emit im = data(i)
Wait
p→(pre n−1)
Ready
q→(pre n−1)
last n = 1 or preempted / emit o = m

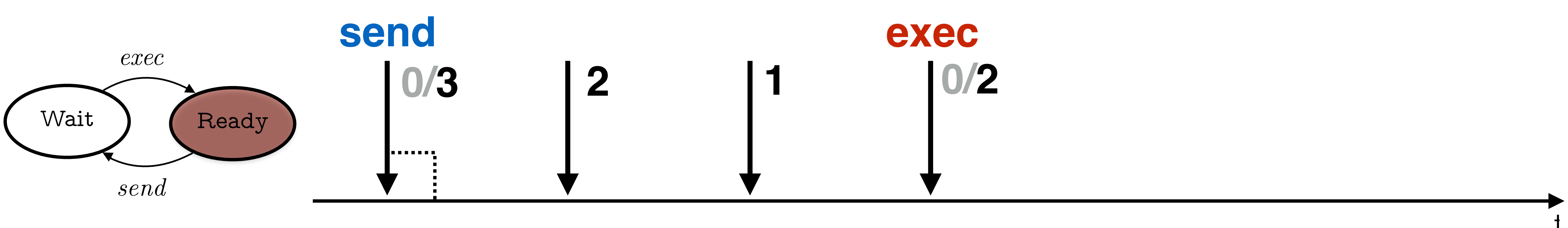

exec
Wait
Ready
send
Sender
send
3
2
1
exec
0/2
t


exec
Wait
Ready
send
Receiver
send
0/3
2
1
exec
0/2
t

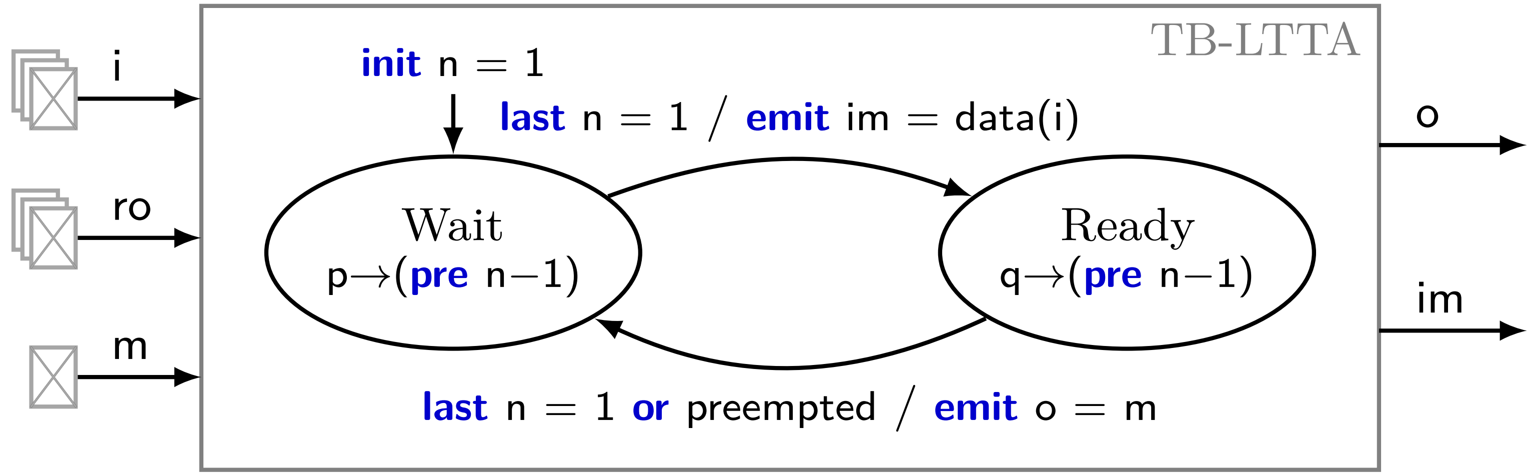


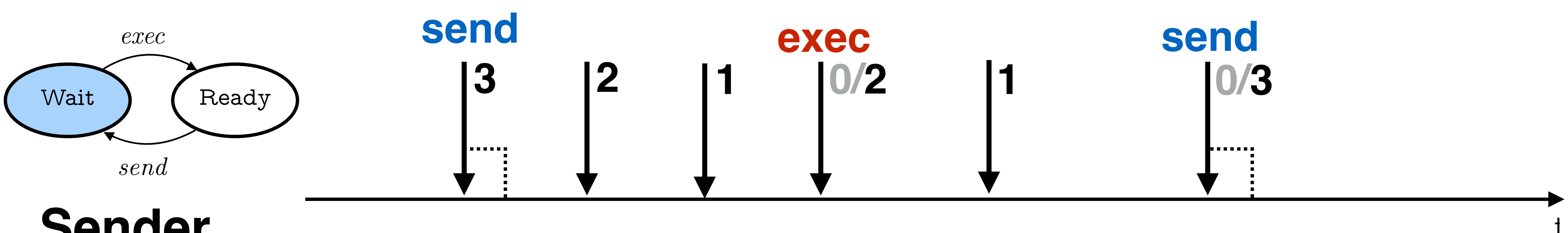


Sender

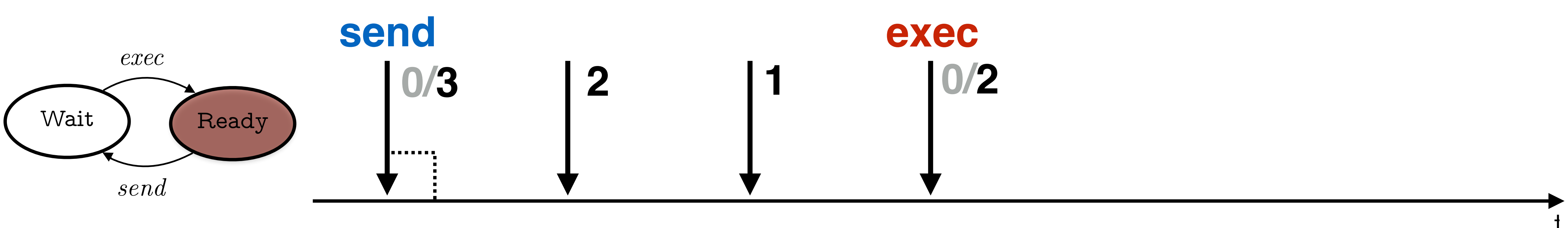


Receiver

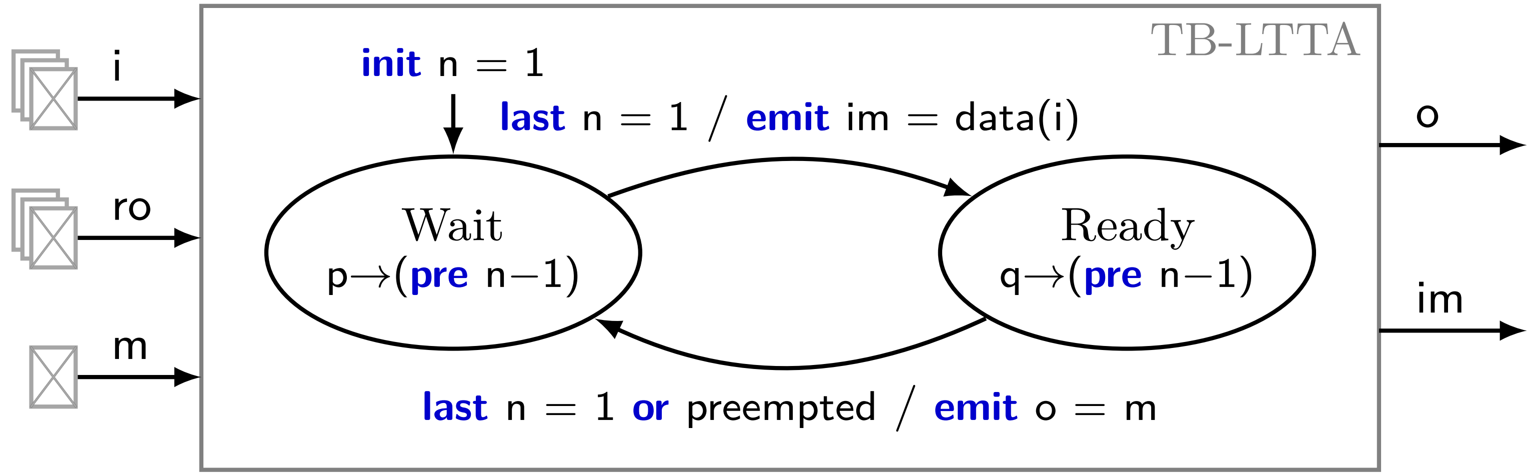




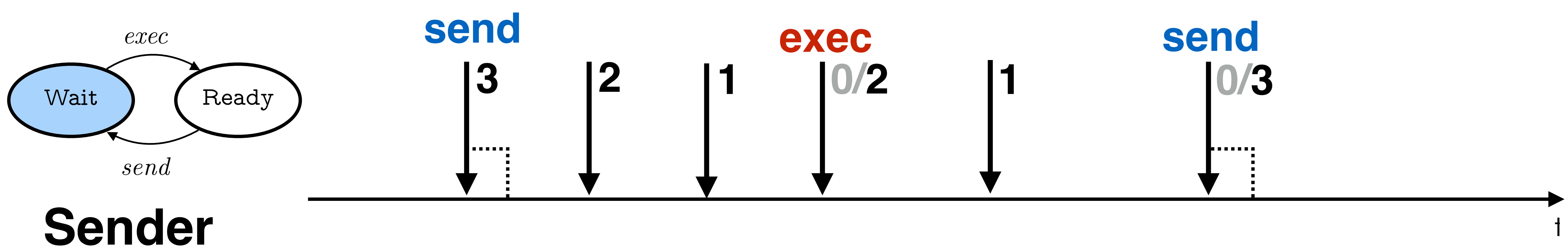
Sender



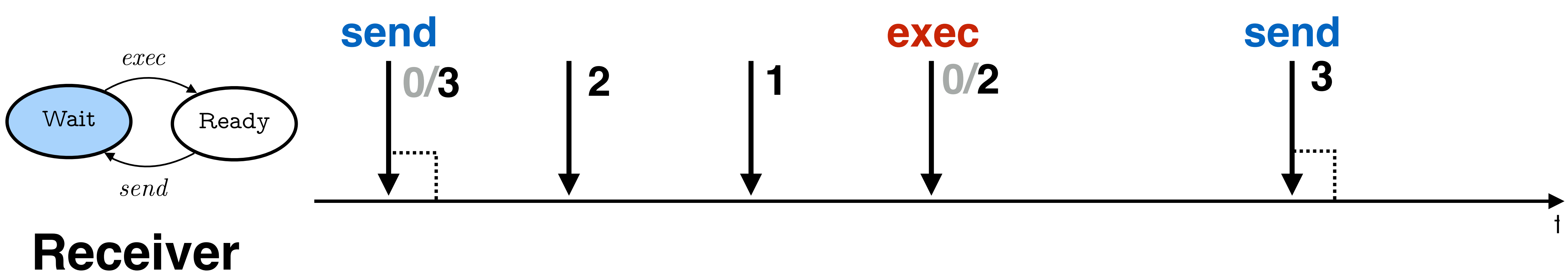
Receiver



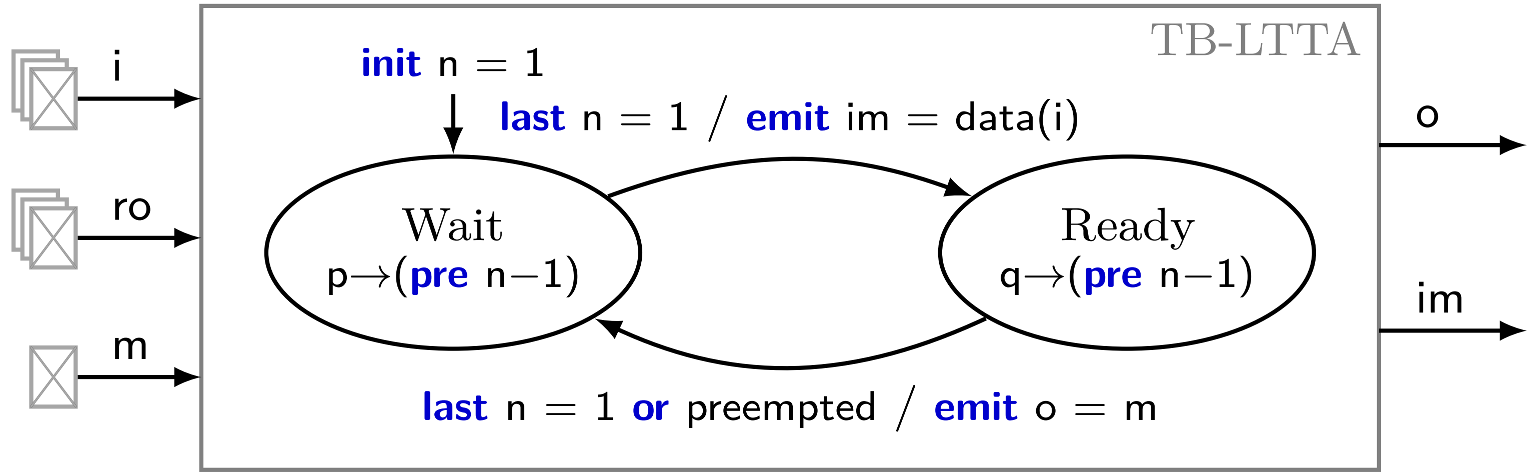
TB-LTTA
i
ro
m
init n = 1
last n = 1 / emit im = data(i)
Wait
p→(pre n−1)
Ready
q→(pre n−1)
last n = 1 or preempted / emit o = m
o
im



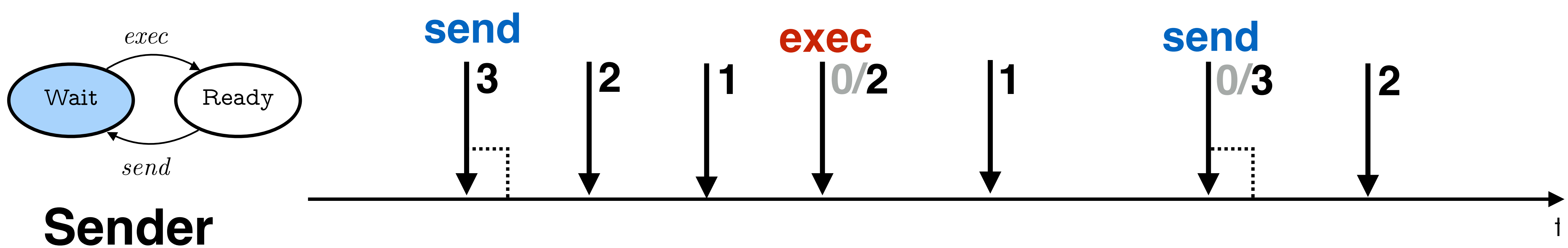
exec
Wait
Ready
send
Sender
send
3
2
1
exec
0/2
1
send
0/3
t



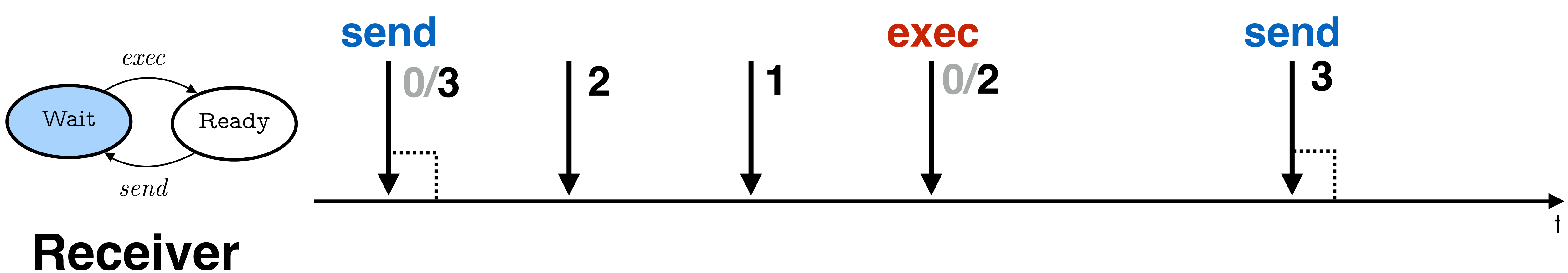
exec
Wait
Ready
send
Receiver
send
0/3
2
1
exec
0/2
send
3
t



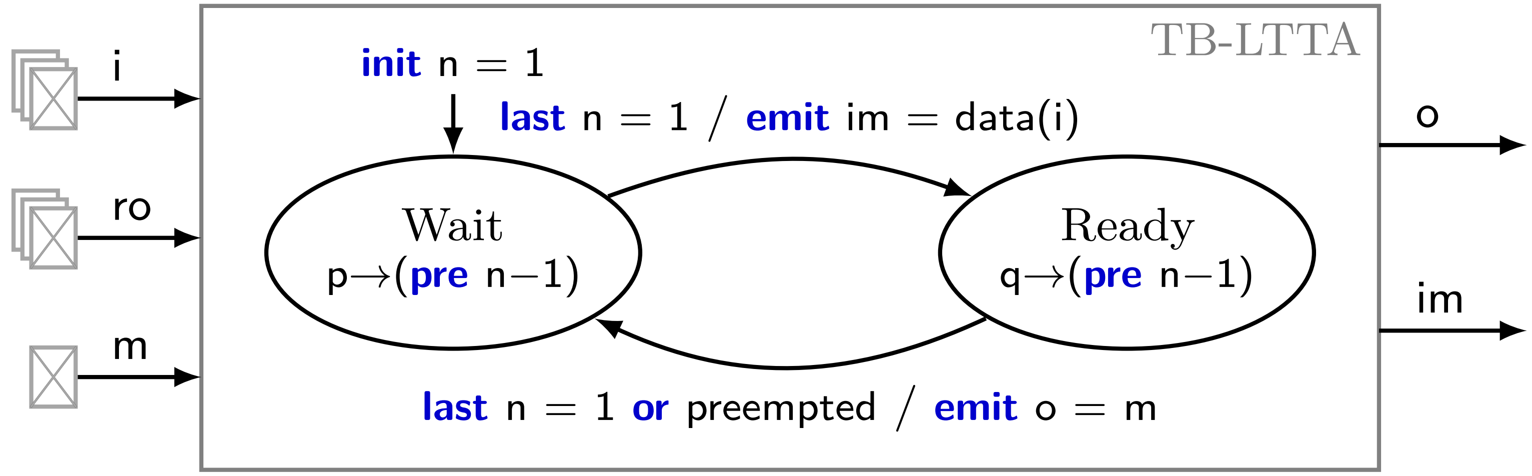
TB-LTTA
i
ro
m
init n = 1
last n = 1 / emit im = data(i)
Wait
p→(pre n−1)
Ready
q→(pre n−1)
last n = 1 or preempted / emit o = m
o
im

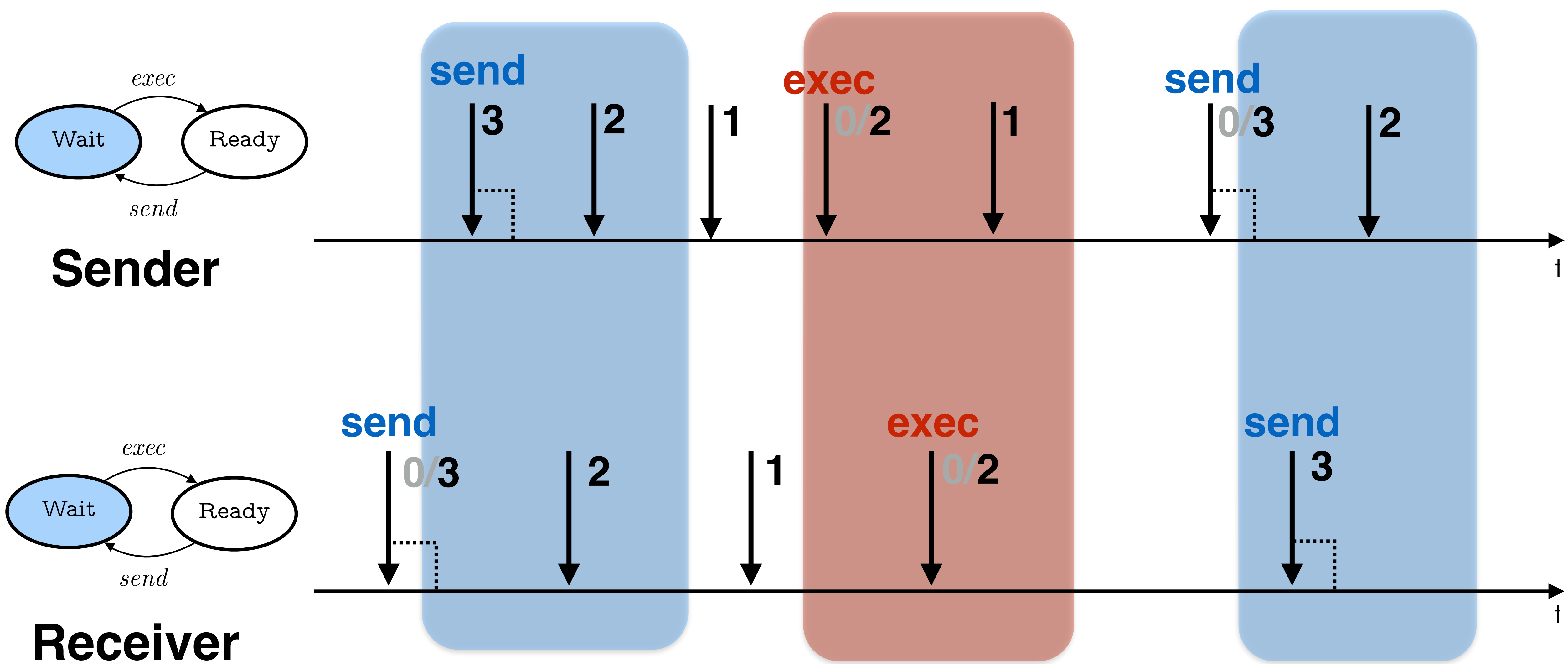


exec
Wait
Ready
send
Sender
send
3
2
1
exec
0/2
1
send
0/3
2
t



exec
Wait
Ready
send
Receiver
send
0/3
2
1
exec
0/2
send
3
t


TB-LTTA
i
ro
m
o
im
init n = 1
last n = 1 / emit im = data(i)
Wait
p→(pre n−1)
Ready
q→(pre n−1)
last n = 1 or preempted / emit o = m


exec
Wait
Ready
send
Sender
exec
Wait
Ready
send
Receiver
send
3
2
1
exec
0/2
1
send
0/3
2
t
send
0/3
2
1
exec
0/2
send
3
t

Time-Based LTTA

- **Theorem 1:**
  Composition of the controller and the embedded machine is always well-defined (no cycle)

- **Theorem 2:**
  Time-based LTTA preserves the Kahn semantics of the embedded application

- **Theorem 3:**
  The worst case throughput is: $1/\lambda_{\mathrm{TB}} = (p+q)T_{\max}$

The Time-Based Protocol

Theorem 2: The following initial counter values ensure the preservation of the Kahn semantics

$$
\begin{aligned}
p &> \frac{2\tau_{max} + T_{max}}{T_{min}} \\
q &> \frac{\tau_{max} - \tau_{min} + (p+1)T_{max}}{T_{min}} - p
\end{aligned}
$$

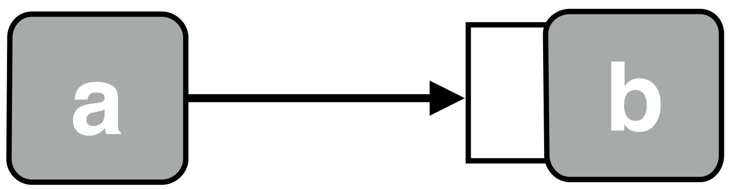
The Time-Based Protocol

Proof sketch

- Worst case reasoning
- Tuning constants p and q (counter initial values)
- Ensure that the receiver always reads the proper data

$\square$

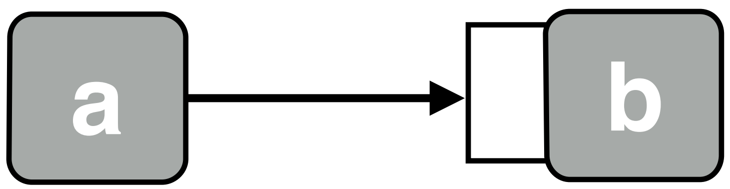
The Time-Based Protocol



$$
\begin{aligned}
p &> \frac{2\tau_{\max}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} \\
q &> \frac{\tau_{\max} - \tau_{\min}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} + p\left(\frac{T_{\max}}{T_{\min}} - 1\right)
\end{aligned}
$$

Property 1: $W^{a}_{k-1} \prec E^{b}_{k}$

The Time-Based Protocol



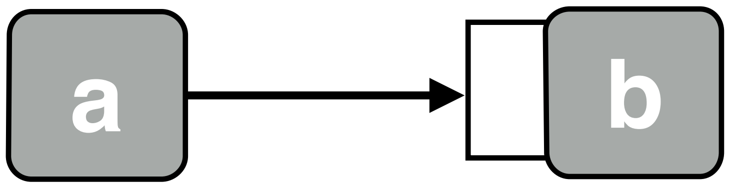
$$p > \frac{2\tau_{\max}}{T_{\min}} + \frac{T_{\max}}{T_{\min}}$$

$$q > \frac{\tau_{\max} - \tau_{\min}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} + p\left(\frac{T_{\max}}{T_{\min}} - 1\right)$$

Property 1: $W^a_{k-1} \prec E^b_k$

W^b_{k-1}

The Time-Based Protocol

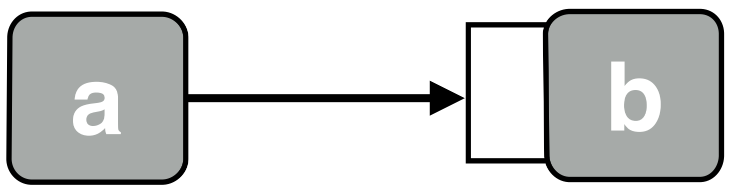


$$
\begin{aligned}
p &> \frac{2\tau_{\max}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} \\
q &> \frac{\tau_{\max} - \tau_{\min}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} + p\left(\frac{T_{\max}}{T_{\min}} - 1\right)
\end{aligned}
$$

Property 1: $W^a_{k-1} \prec E^b_k$

$$W^b_{k-1} \xrightarrow{T_{\max} + \tau_{\max}} W^a_{k-1}$$

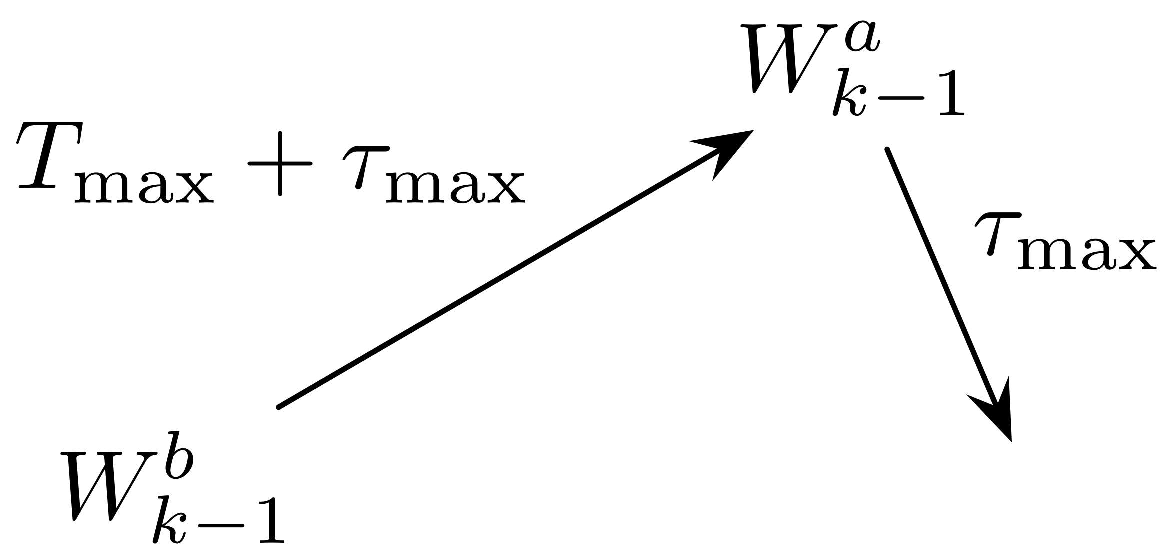
The Time-Based Protocol



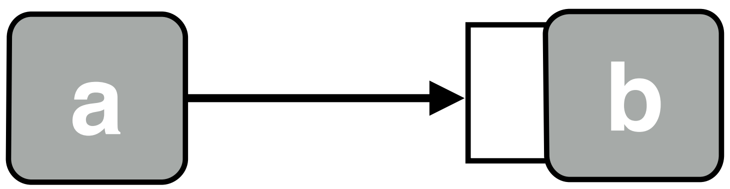
$$p \;>\; \frac{2\tau_{\max}}{T_{\min}} + \frac{T_{\max}}{T_{\min}}$$

$$q \;>\; \frac{\tau_{\max} - \tau_{\min}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} + p\left(\frac{T_{\max}}{T_{\min}} - 1\right)$$

Property 1: $W^a_{k-1} \prec E^b_k$



The Time-Based Protocol

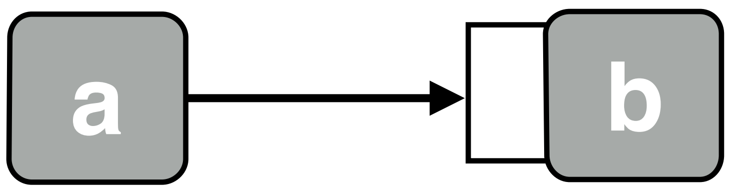


$$
\begin{aligned}
p &> \frac{2\tau_{\max}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} \\
q &> \frac{\tau_{\max} - \tau_{\min}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} + p\left(\frac{T_{\max}}{T_{\min}} - 1\right)
\end{aligned}
$$

Property 1: $W^a_{k-1} \prec E^b_k$

$$
\begin{tikzcd}
 & W^a_{k-1} \arrow[dr, "\tau_{\max}"] & \\
W^b_{k-1} \arrow[ur, "T_{\max} + \tau_{\max}"] \arrow[rr, "pT_{\min}"'] & & E^b_k
\end{tikzcd}
$$

The Time-Based Protocol

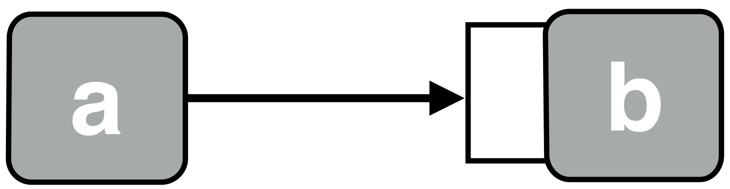


$$p > \frac{2\tau_{\max}}{T_{\min}} + \frac{T_{\max}}{T_{\min}}$$
$$q > \frac{\tau_{\max} - \tau_{\min}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} + p\left(\frac{T_{\max}}{T_{\min}} - 1\right)$$

Property 1: $W^a_{k-1} \prec E^b_k$

Property 2: $E^b_k \prec W^a_k$

The Time-Based Protocol



$$p > \frac{2\tau_{\max}}{T_{\min}} + \frac{T_{\max}}{T_{\min}}$$

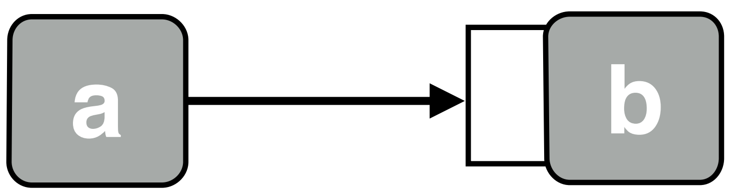
$$q > \frac{\tau_{\max} - \tau_{\min}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} + p\left(\frac{T_{\max}}{T_{\min}} - 1\right)$$

Property 1: $W^a_{k-1} \prec E^b_k$

Property 2: $E^b_k \prec W^a_k$

W^a_{k-1}

The Time-Based Protocol



$$p > \frac{2\tau_{\max}}{T_{\min}} + \frac{T_{\max}}{T_{\min}}$$

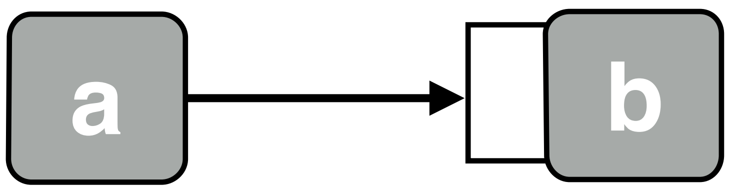
$$q > \frac{\tau_{\max} - \tau_{\min}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} + p\left(\frac{T_{\max}}{T_{\min}} - 1\right)$$

Property 1: $W_{k-1}^a \prec E_k^b$

Property 2: $E_k^b \prec W_k^a$

$$W_{k-1}^a \xrightarrow{pT_{\min}} E_k^a$$

The Time-Based Protocol



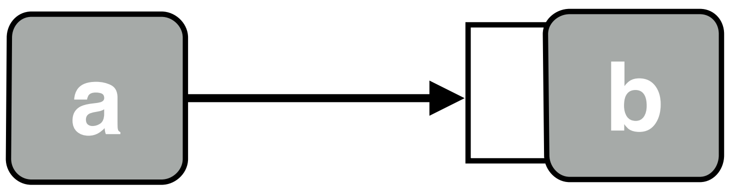
$$p > \frac{2\tau_{\max}}{T_{\min}} + \frac{T_{\max}}{T_{\min}}$$
$$q > \frac{\tau_{\max} - \tau_{\min}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} + p\left(\frac{T_{\max}}{T_{\min}} - 1\right)$$

Property 1: $W_{k-1}^a \prec E_k^b$

Property 2: $E_k^b \prec W_k^a$

$$W_{k-1}^a \xrightarrow{pT_{\min}} E_k^a \xrightarrow{qT_{\min}} W_k^a$$

The Time-Based Protocol



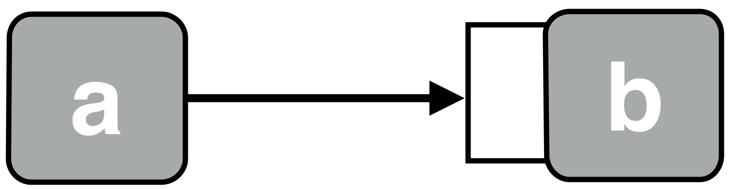
$$
\begin{aligned}
p &> \frac{2\tau_{\max}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} \\
q &> \frac{\tau_{\max} - \tau_{\min}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} + p\left(\frac{T_{\max}}{T_{\min}} - 1\right)
\end{aligned}
$$

Property 1: $W^a_{k-1} \prec E^b_k$

Property 2: $E^b_k \prec W^a_k$

$$
\begin{tikzcd}
W^a_{k-1} \arrow[r, "pT_{\min}"] & E^a_k \arrow[r, "qT_{\min}"] & W^a_k \arrow[dr, "\tau_{\min}"] & \\
 & & & {}
\end{tikzcd}
$$

The Time-Based Protocol



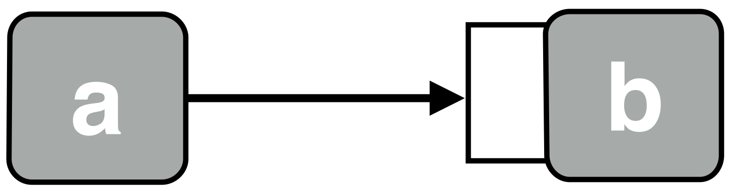
$$
\begin{aligned}
p &> \frac{2\tau_{\max}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} \\
q &> \frac{\tau_{\max} - \tau_{\min}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} + p\left(\frac{T_{\max}}{T_{\min}} - 1\right)
\end{aligned}
$$

Property 1: $W^a_{k-1} \prec E^b_k$

Property 2: $E^b_k \prec W^a_k$

$$
\begin{tikzcd}
W^a_{k-1} \arrow[r, "pT_{\min}"] \arrow[dr, "T_{\max}+\tau_{\max}"'] & E^a_k \arrow[r, "qT_{\min}"] & W^a_k \arrow[dr, "\tau_{\min}"] & \\
& W^b_{k-1} & & {}
\end{tikzcd}
$$

The Time-Based Protocol



$$
\begin{aligned}
p &> \frac{2\tau_{\max}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} \\
q &> \frac{\tau_{\max} - \tau_{\min}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} + p\left(\frac{T_{\max}}{T_{\min}} - 1\right)
\end{aligned}
$$

Property 1: $W^a_{k-1} \prec E^b_k$

Property 2: $E^b_k \prec W^a_k$

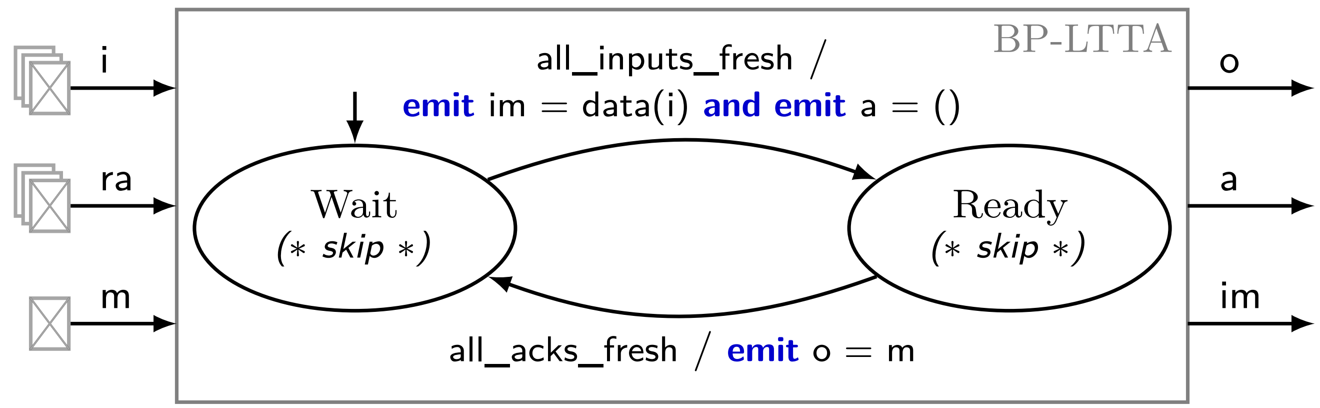
$$
\begin{tikzcd}
W^a_{k-1} \arrow[r, "pT_{\min}"] \arrow[dr, "T_{\max} + \tau_{\max}"'] & E^a_k \arrow[r, "qT_{\min}"] & W^a_k \arrow[dr, "\tau_{\min}"] & \\
& W^b_{k-1} \arrow[rr, "pT_{\max}"'] & & E^b_k
\end{tikzcd}
$$

The Time-Based Protocol

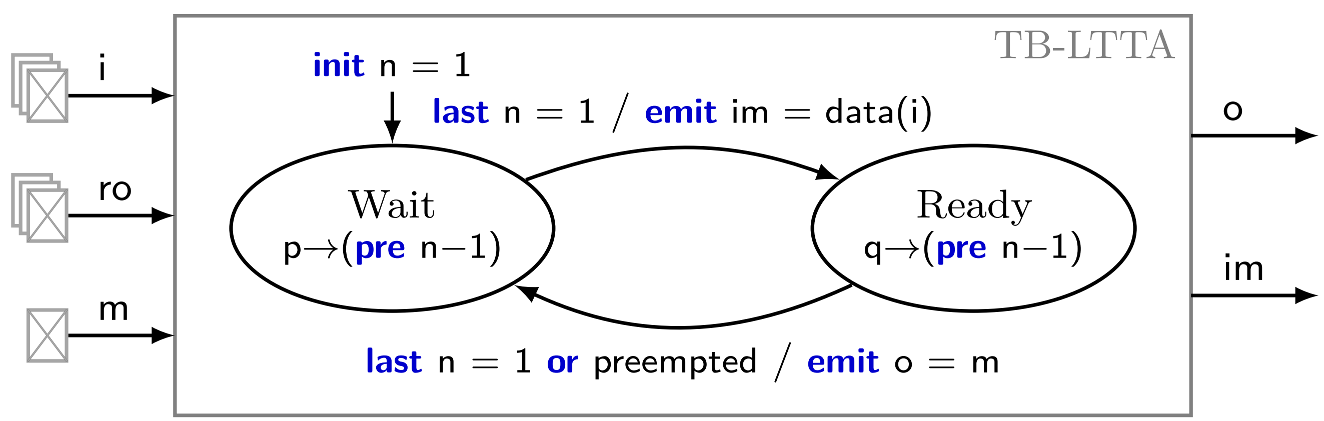
Corollary: The protocol ensures alternation between **exec** and **send** phases for each pair of communicating nodes

Broadcast communication: ensure clean alternation throughout the entire architecture (idem for back-pressure LTTA)

Comparison



Back-pressure



Time-based

Comparison

	Time-Based	Back-Pressure
Flexibility	Require architecture specifications	Very flexible
Robustness	Can run in a degraded mode	Stuck if a node crash
Fault Tolerance	Can be programmed in the application	Implemented in the Middleware
Communication	Any	Any
Pipelining	Limited	Optimal

Comparison

	Time-Based	Back-Pressure
Flexibility	Require architecture specifications	Very flexible
Robustness	Can run in a degraded mode	Stuck if a node crash
Fault Tolerance	Can be programmed in the application	Implemented in the Middleware
Communication	Any	Any
Pipelining	Limited	Optimal

Outline

1. What is an LTTA?
   1. Quasi-Periodic Architecture
   2. Synchronous Applications

2. General Framework

3. The two protocols
   1. Back-Pressure LTTA
   2. Time-Based LTTA

4. **What About Clock Synchronisation?**

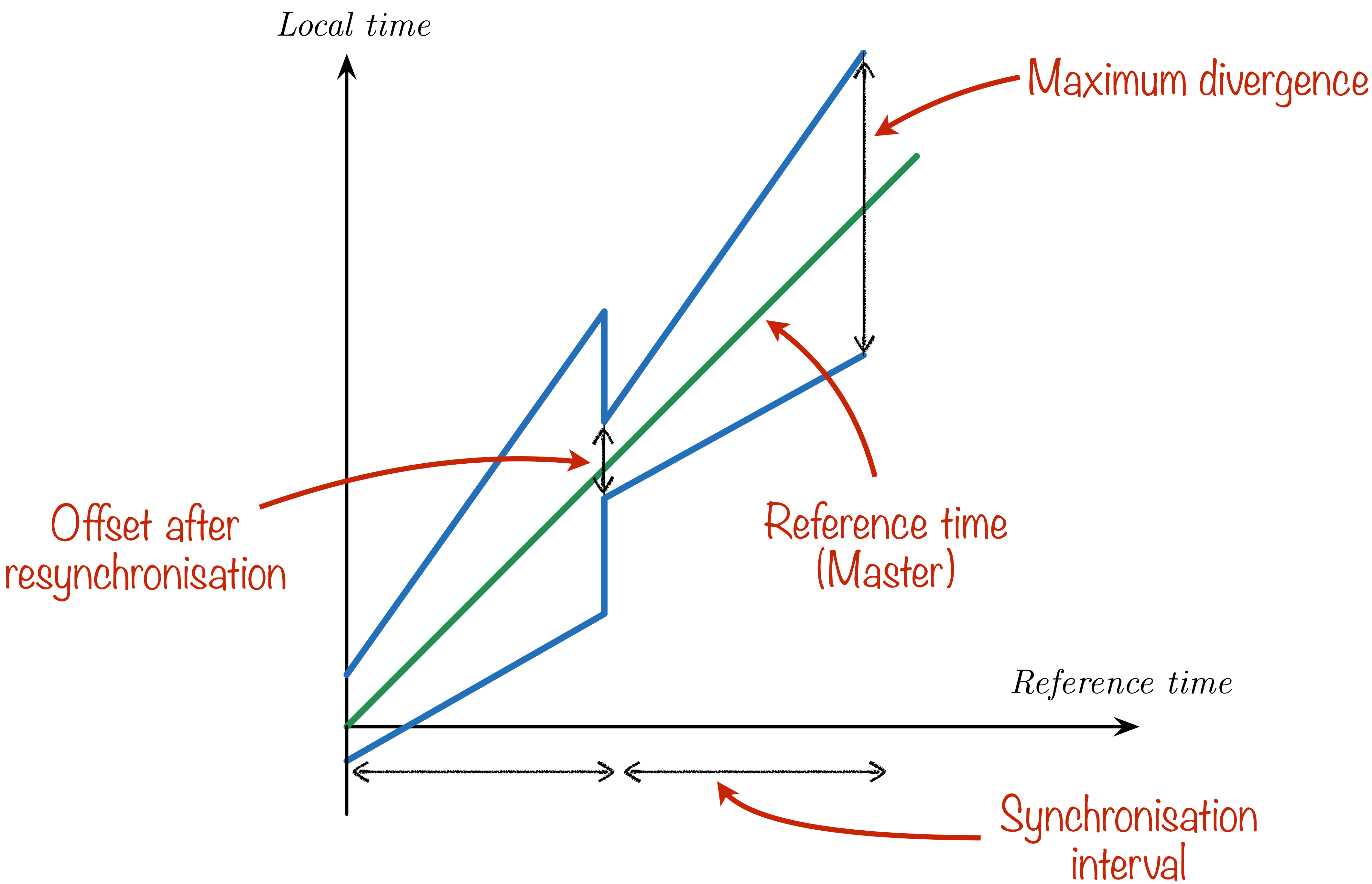
Global Clock Protocol

Central Master Synchronisation

- **Goal:** Implement clock synchronisation on a quasi-periodic architecture
- We use the most efficient protocol for comparison purposes
- One node is used as a time reference for all other nodes: the **central master clock**

Global Clock Protocol

The Big Picture



Global Clock Protocol

Central Master Synchronisation

Synchronisation interval	R
Offset after resynchronisation	$\Phi = \tau_{\max} + T_{\max} - \tau_{\min}$
Drift rate	$\rho = \frac{T_{\max}}{T_n} - 1 = \frac{T_{\max} - T_{\min}}{T_{\max} + T_{\min}}$
Maximum divergence	$\Gamma = 2\rho R$
Precision	$\Pi = \Phi + \Gamma$

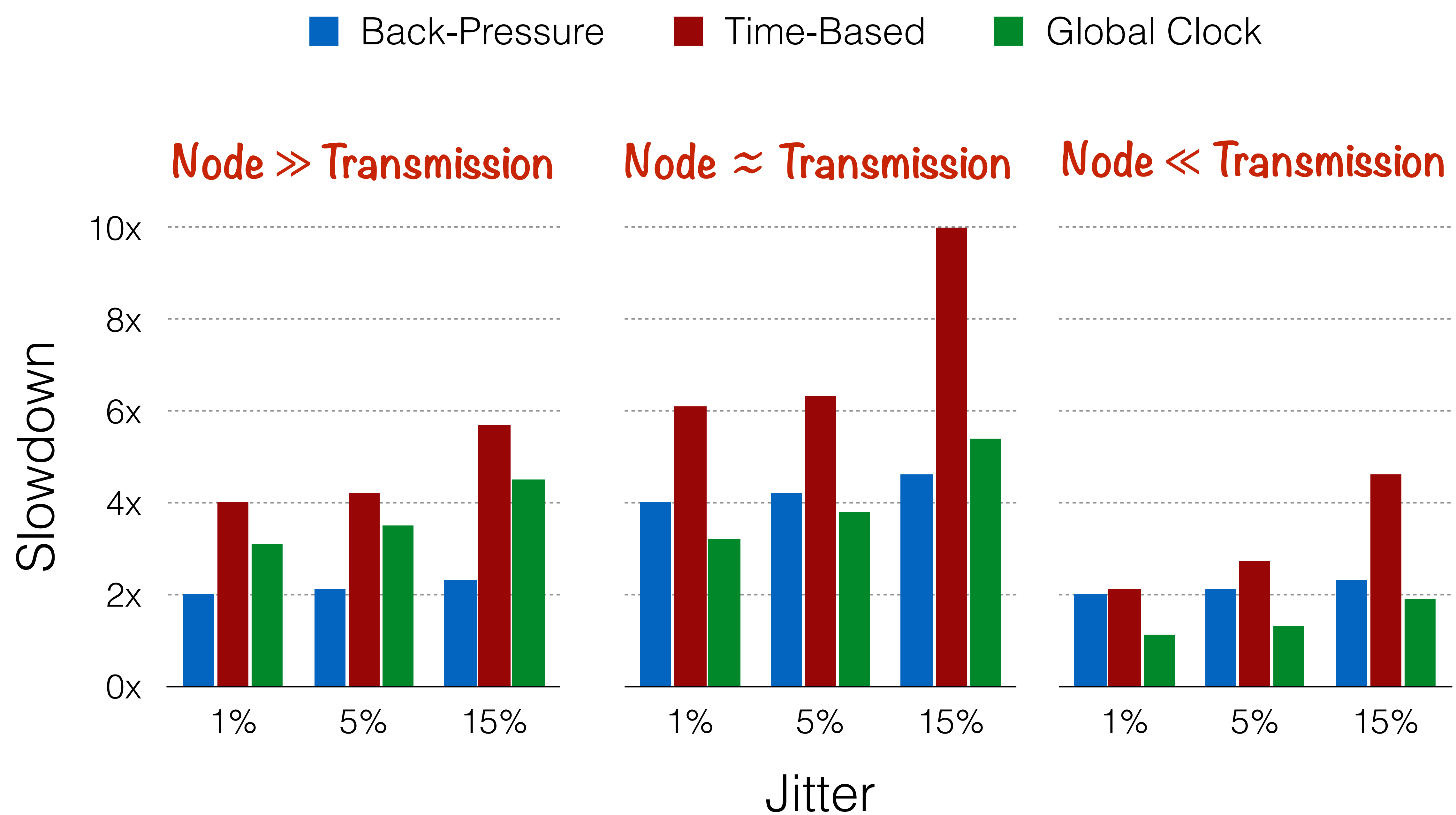
Global Clock Protocol

Lock-Step Execution

- Given the precision of the synchronisation one can build a global notion of time or **macroticks**
- Activating nodes on each macrotick imposes a synchronous execution.
- But we need to wait for the transmission delay between execution steps.

Comparative Evaluation

Compared to synchronous execution*



* The smaller, the better.

Conclusion

- Our new model simplifies and clarifies those of previous papers
- A new proposition for the time-based protocol that does not require broadcast communication and does allow pipelining
- Model and Simulation of the protocols in Zélus
  Discrete model + link with continuous time
- Comparison with clock synchronisation deployed on the same architecture