MEMOCODE 2024

Efficient Coordination for Distributed Discrete-Event Systems

Arizona State University

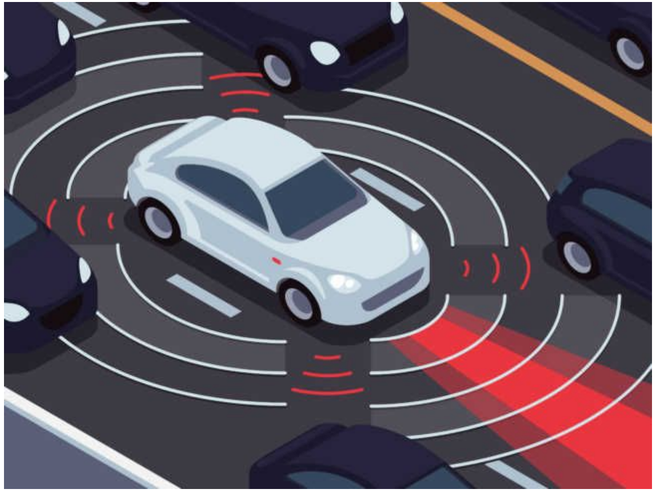
Byeonggil Jun[1], Edward A. Lee[2], Marten Lohstroh[2], and Hokeun Kim[1]

UC Berkeley

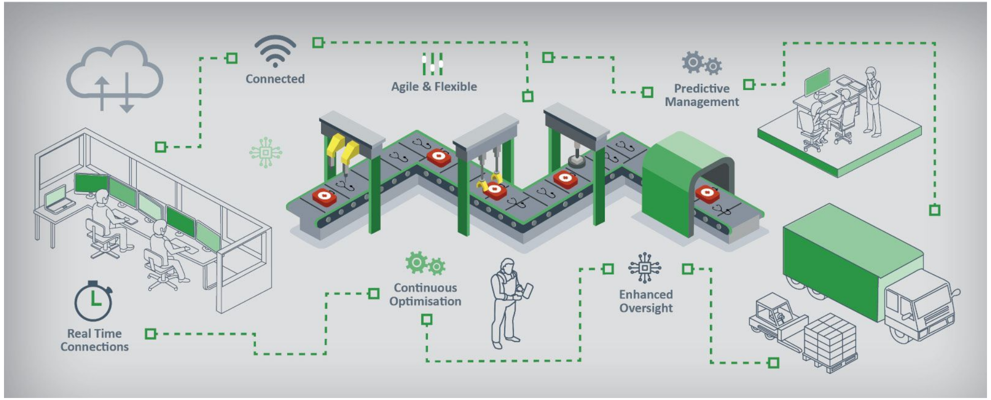
1: Arizona State University
2: University of California, Berkeley

Introduction

- Determinism often matters in distributed cyber-physical systems
- HLA (high-level architecture) is one way to ensure determinism in distributed systems
  - But, HLA incurs a huge network overhead



https://www.iiot-world.com/artificial-intelligence-ml/autonomous-vehicles/challenges-in-training-algorithms-for-autonomous-cars/



https://slcontrols.com/solutions/smart-factory/

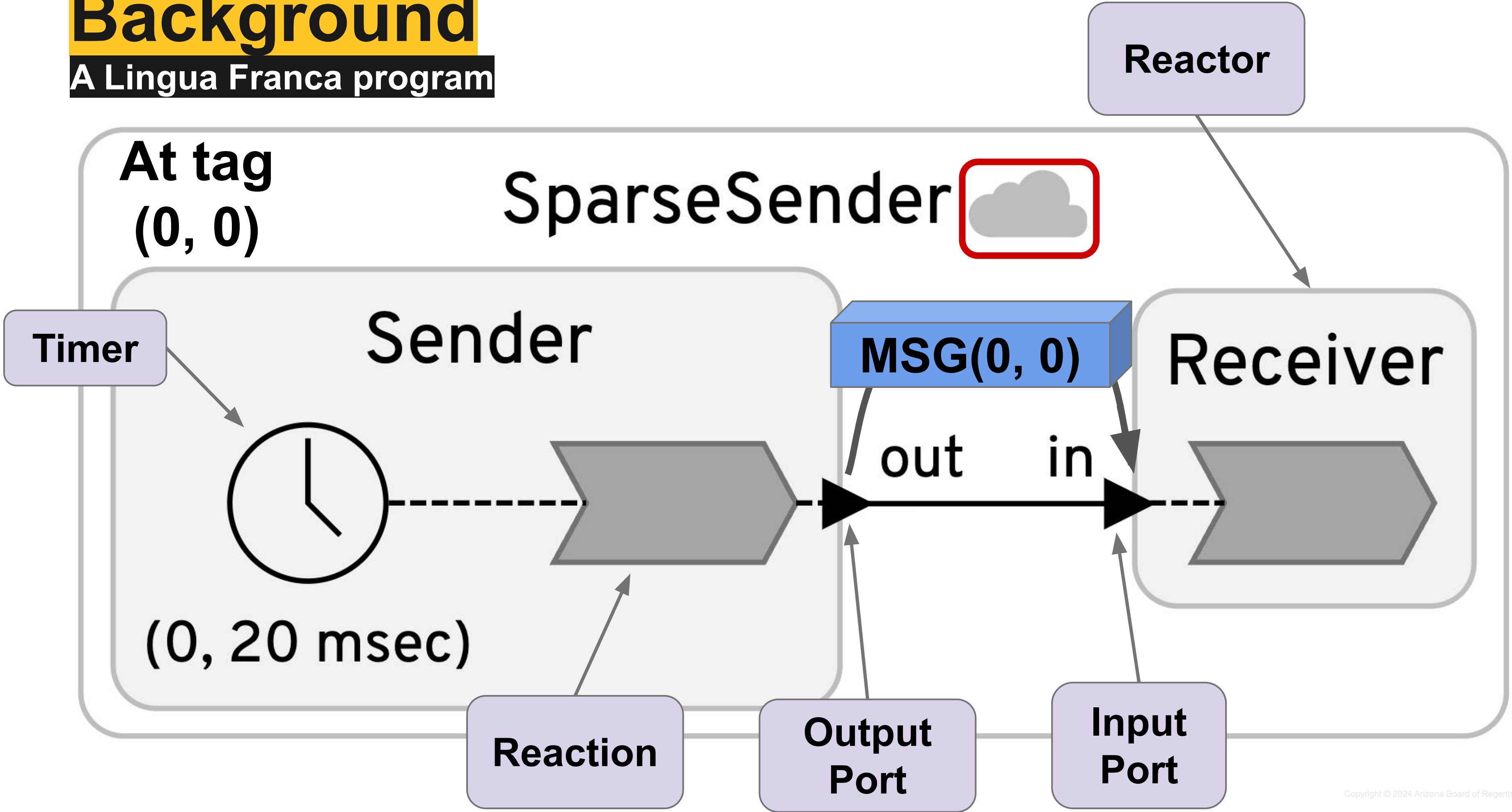
Related Work

- High Level Architecture, IEEE Standards 2010[1]
    - Runtime Infrastructure and Federates
- Rudie et.al., “Minimal communication in a distributed discrete-event system,” IEEE TACON 2003[2]
- Wang et.al., “Optimistic Synchronization in HLA-Based Distributed Simulation,” ACM PADS 2004[3]
- COSSIM, ACM TACO 2020[4]

[1] IEEE, “IEEE standard for modeling and simulation (M&S) high level architecture (HLA)– framework and rules,” IEEE Std 1516-2010 (Revi- sion of IEEE Std 1516-2000) - Redline, pp. 1–38, 2010.
[2] K. Rudie, S. Lafortune, and F. Lin, ‘Minimal communication in a distributed discrete-event system’, IEEE Transactions on Automatic Control, vol. 48, no. 6, pp. 957–975, 2003.
[3] Wang X, Turner SJ, Low MYH, Gan BP. Optimistic Synchronization in HLA-Based Distributed Simulation. SIMULATION. 2005;81(4):279-291. doi:10.1177/0037549705054931
[4] Nikolaos Tampouratzis et.al. 2020. A Novel, Highly Integrated Simulator for Parallel and Distributed Systems. ACM Trans. Archit. Code Optim. 17, 1, Article 2 (March 2020), 28 pages. https://doi-org.ezproxy1.lib.asu.edu/10.1145/3378934

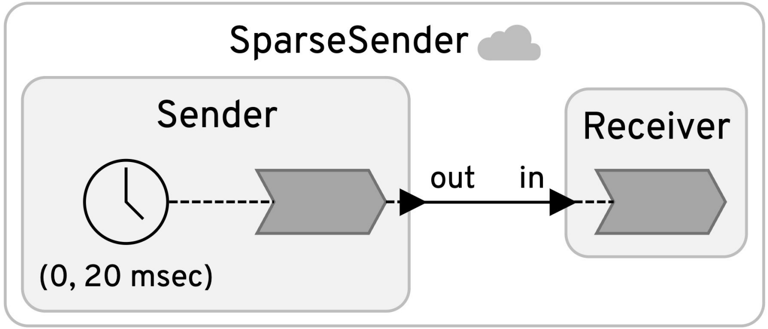
Background

A Lingua Franca program



Background

Centralized Coordination

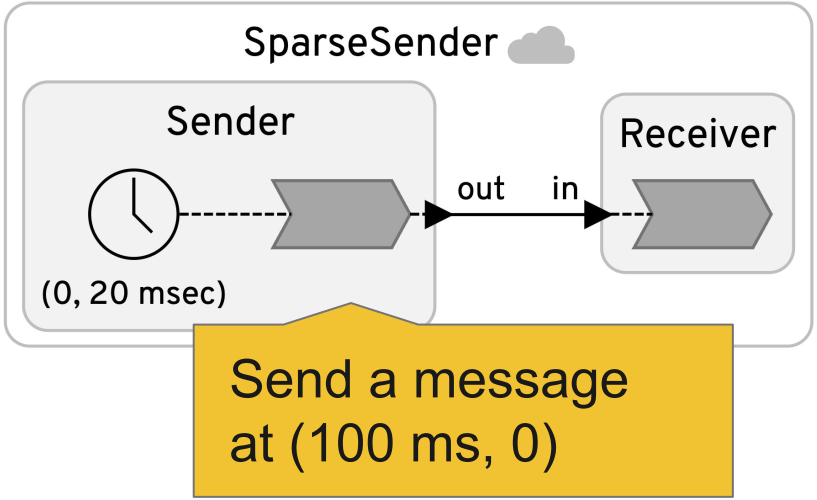


- Tagged Message (MSG)
  - A message with a tag
- Latest Tag Complete (LTC)
  - To notify that a federate has finished a tag
- Next Event Tag (NET)
  - To report the tag of the earliest unprocessed event
- Tag Advance Grant (TAG)
  - To grant a federate to advance its tag to G(TAG), the payload tag

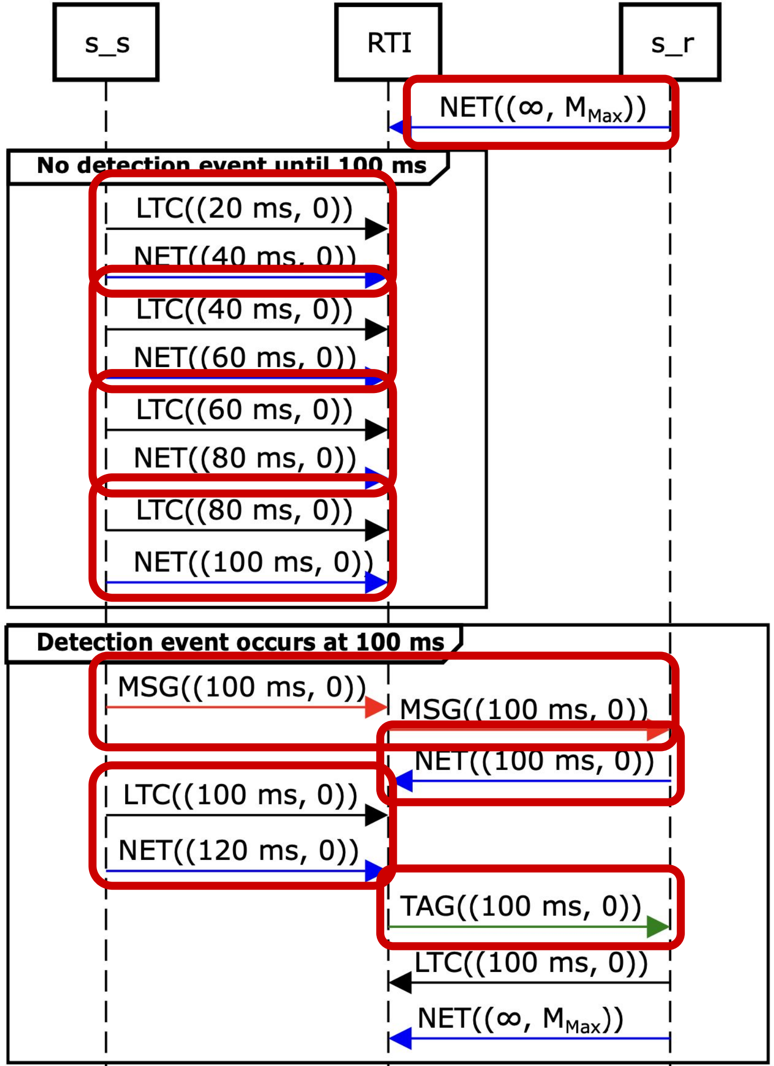
Name	Payload	Description	Direction
MSG_{ij}	Tag & Message	Tagged Message	j to i via RTI
LTC_j	Tag	Latest Tag Complete	j to RTI
NET_j	Tag	Next Event Tag	j to RTI
TAG_i	Tag	Tag Advance Grant	RTI to i

Motivating Example

Sparse Sender



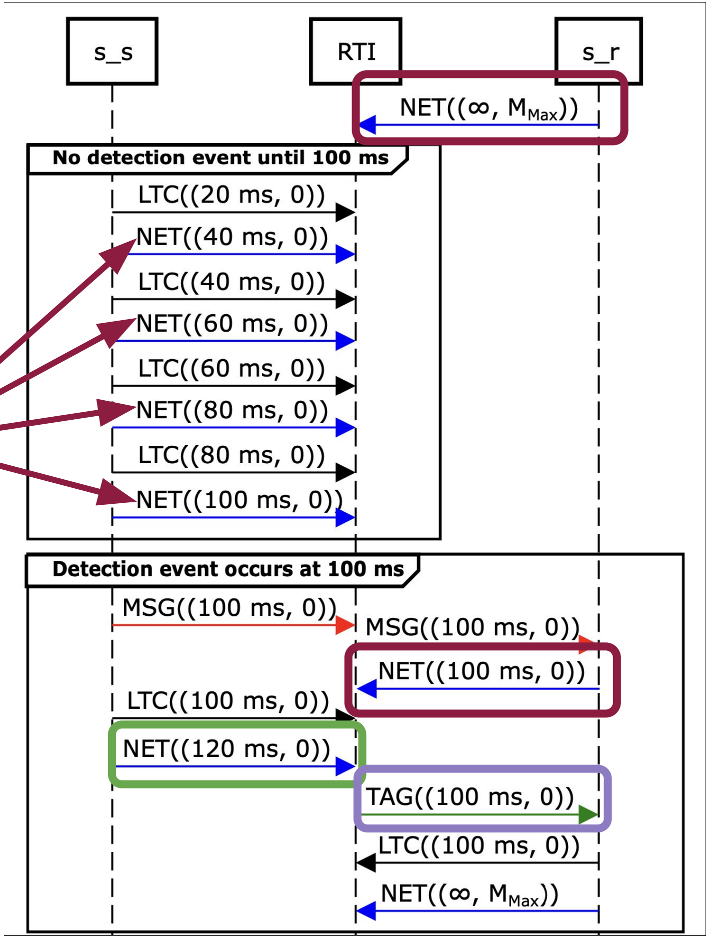
Name	Payload	Description	Direction
MSG_{ij}	Tag & Message	Tagged Message	j to i via RTI
LTC_j	Tag	Latest Tag Complete	j to RTI
NET_j	Tag	Next Event Tag	j to RTI
TAG_i	Tag	Tag Advance Grant	RTI to i

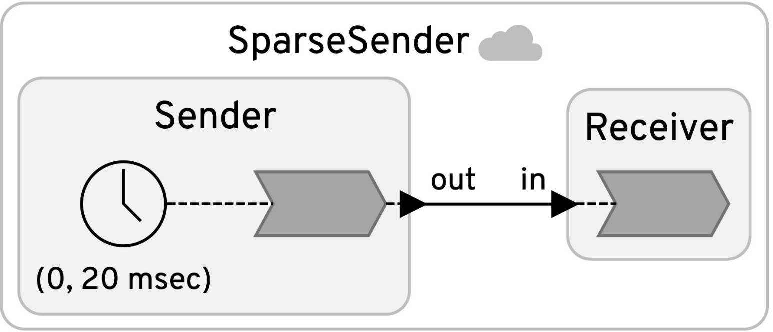


Motivating Example

- Purposes of NET signals
  - Acquire TAG signals
  - Let the RTI grant TAG signals to other federates

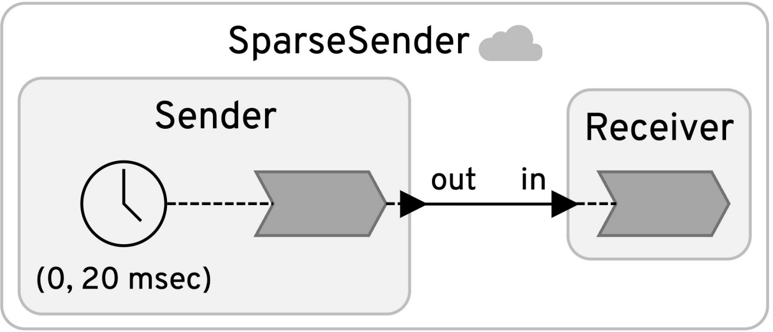
Unnecessary!!

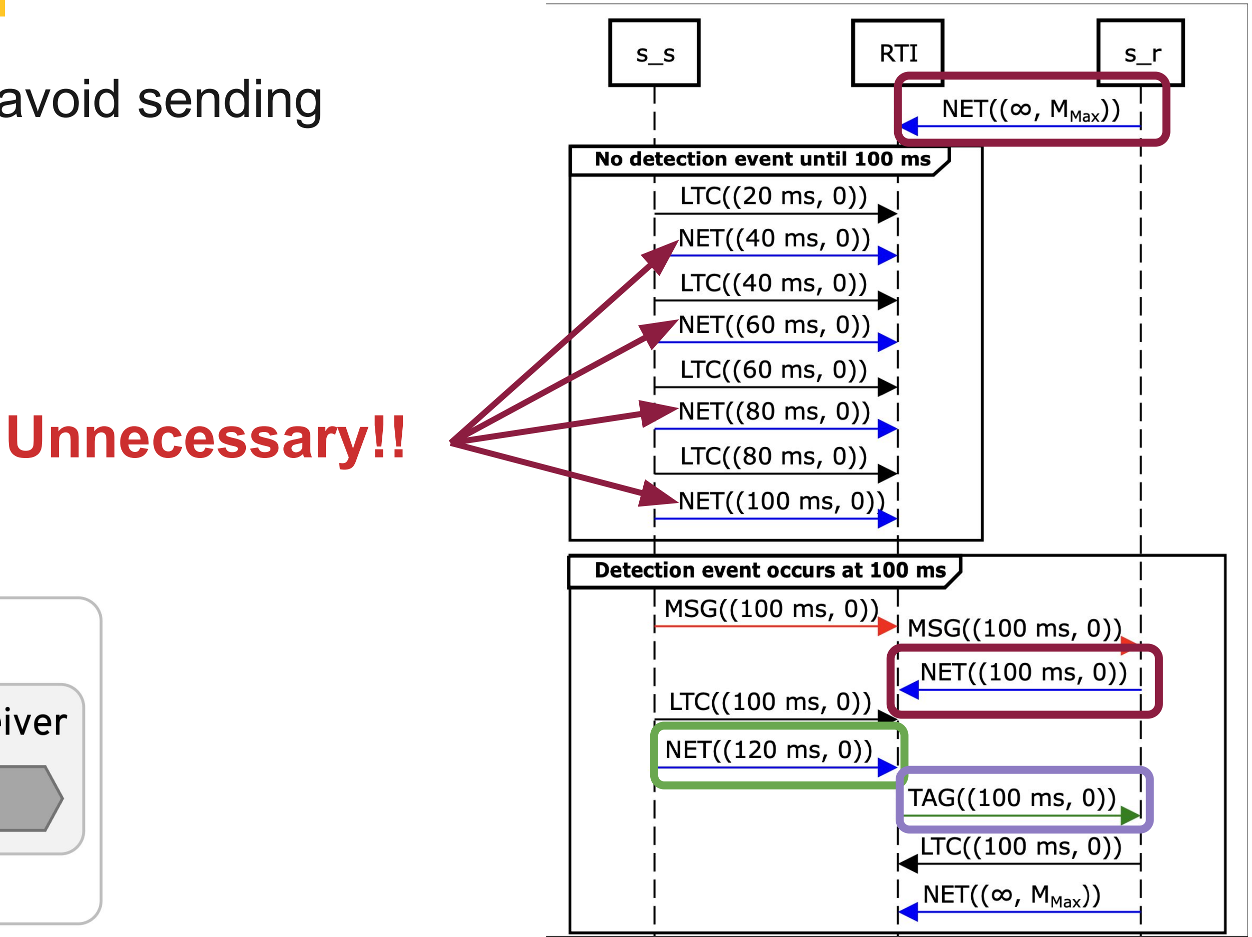




Research goal

- How can a federate avoid sending unnecessary NETs?

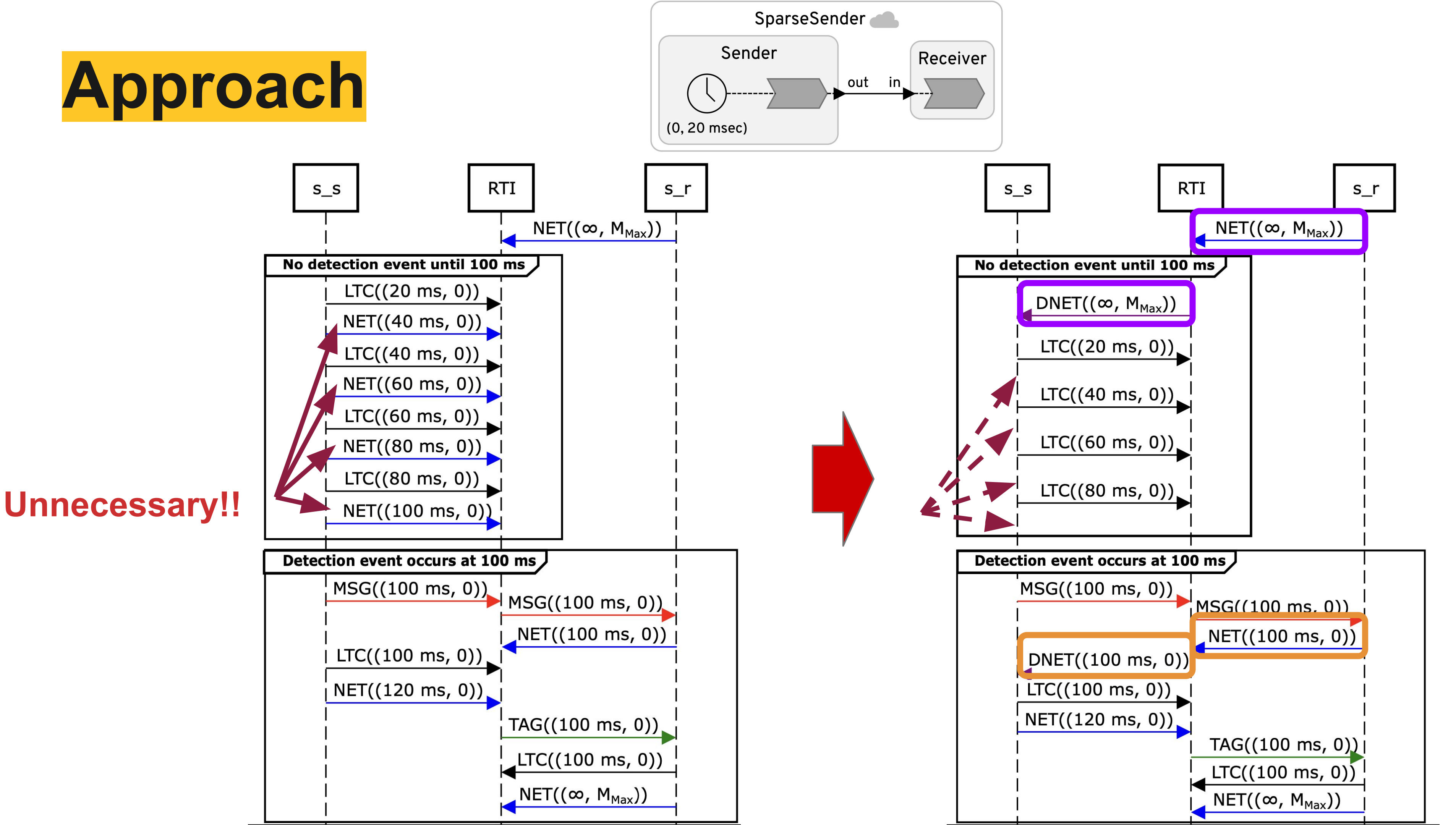




Approach

- **Our Approach to eliminate unnecessary NET signals**
  - **Downstream Next Event Tag (DNET)**
  - For each federate, the RTI computes the latest unnecessary NET
  - A federate does not need to send any NETs with a tag _g_ less than or equal to G(DNET), payload of DNET

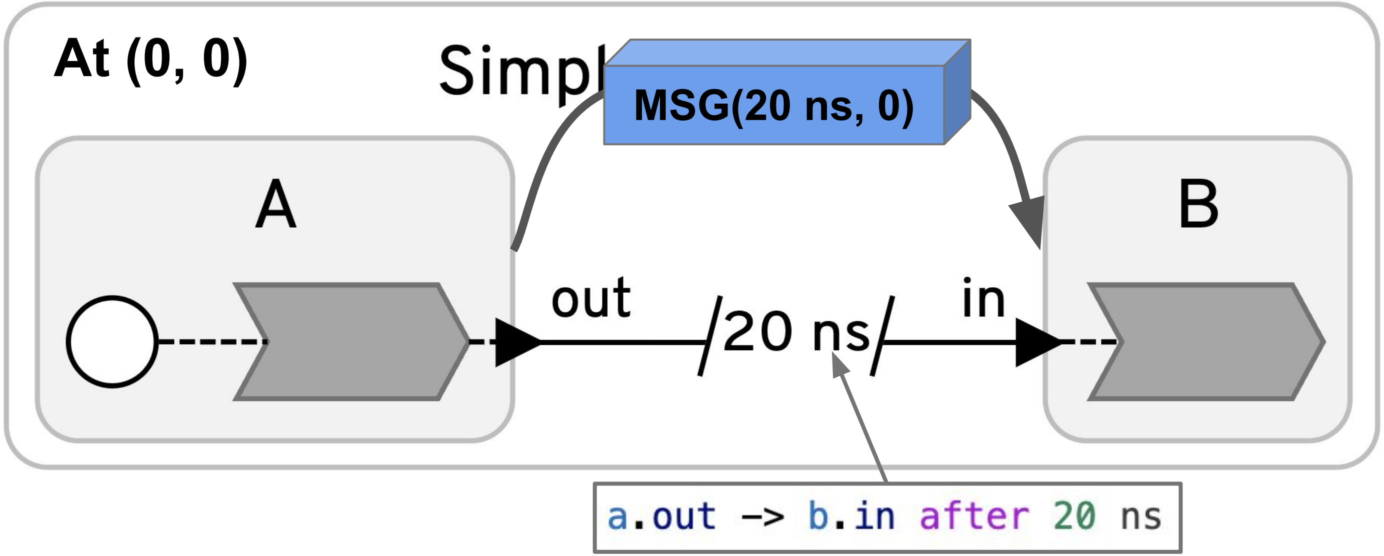
Approach



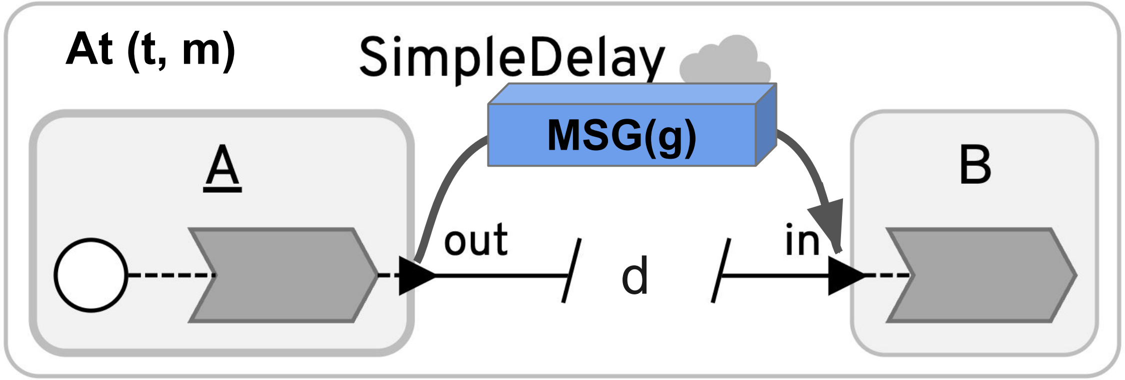
SparseSender
Sender
(0, 20 msec)
out
in
Receiver
s_s
RTI
s_r
NET((∞, MMax))
No detection event until 100 ms
LTC((20 ms, 0))
NET((40 ms, 0))
LTC((40 ms, 0))
NET((60 ms, 0))
LTC((60 ms, 0))
NET((80 ms, 0))
LTC((80 ms, 0))
NET((100 ms, 0))
Detection event occurs at 100 ms
MSG((100 ms, 0))
MSG((100 ms, 0))
NET((100 ms, 0))
LTC((100 ms, 0))
NET((120 ms, 0))
TAG((100 ms, 0))
LTC((100 ms, 0))
NET((∞, MMax))
Unnecessary!!
s_s
RTI
s_r
NET((∞, MMax))
No detection event until 100 ms
DNET((∞, MMax))
LTC((20 ms, 0))
LTC((40 ms, 0))
LTC((60 ms, 0))
LTC((80 ms, 0))
Detection event occurs at 100 ms
MSG((100 ms, 0))
MSG((100 ms, 0))
NET((100 ms, 0))
DNET((100 ms, 0))
LTC((100 ms, 0))
NET((120 ms, 0))
TAG((100 ms, 0))
LTC((100 ms, 0))
NET((∞, MMax))

Challenges

- Logical Delay
  - To indicate logical time elapsing through a connection



Challenges

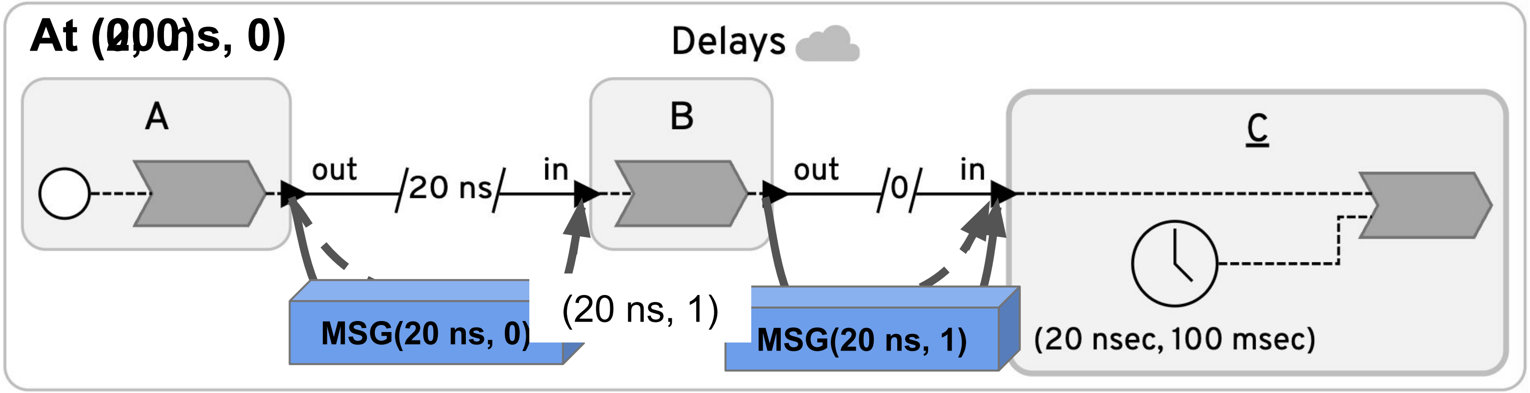


$$g = \begin{cases} (t, m), & \text{if } d < 0 \\ (t, m+1), & \text{if } d = 0 \\ (t+d, 0) & \text{if } d > 0 \end{cases}$$

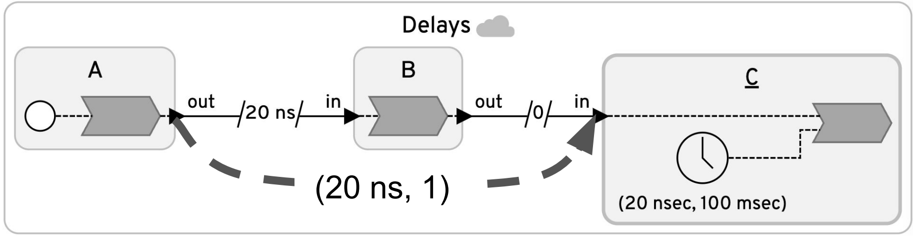
(No Delay)

Challenges

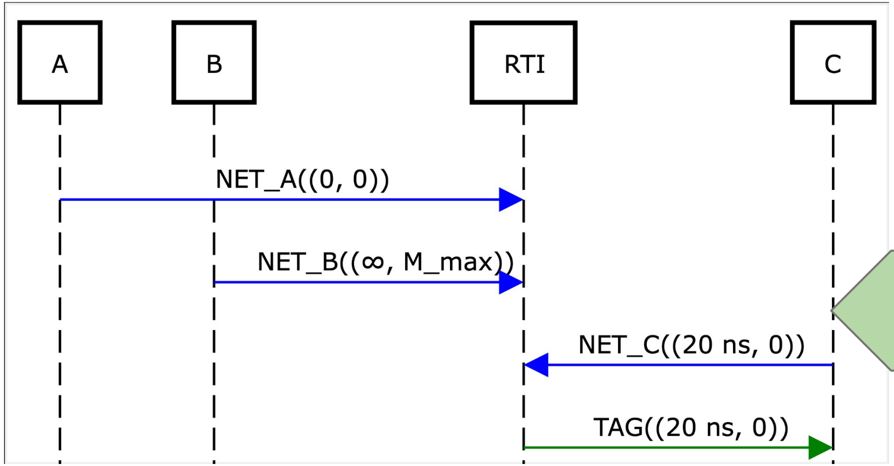
- Minimum tag increment over all connections
  - Ex) D_{CA} = (20 ns, 1) and D_{BC} = (0, 1)



Challenges

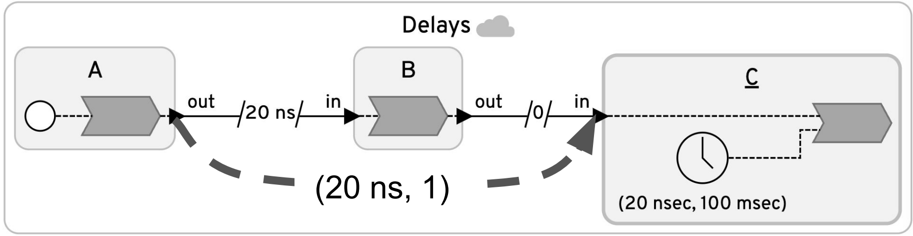


- Earliest Incoming Message Tag (EIMT)
  - A's event at (0, 0) may cause a message to C with tag (20 ns, 1)
  - C will not receive any message with a tag < (20 ns, 1)

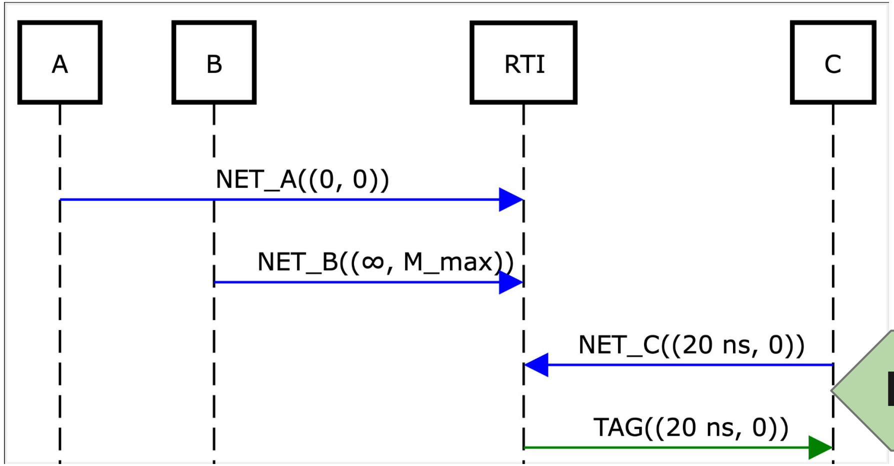


$EIMT_C = (20 \text{ ns}, 1)$

Challenges

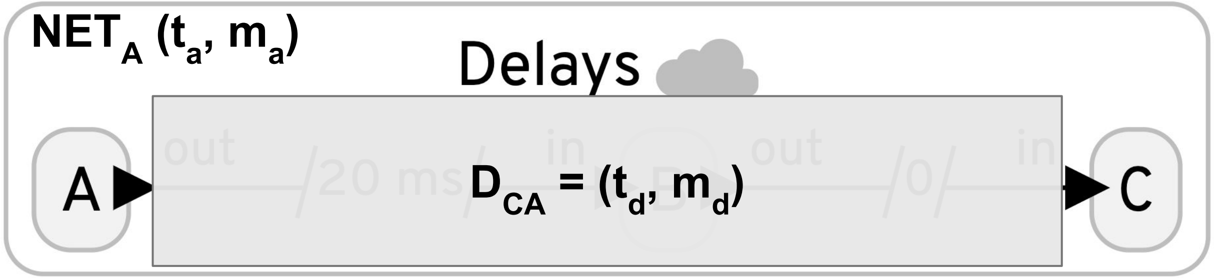


- Earliest Incoming Message Tag
  - The RTI uses EIMT for computing TAG signals



$NET_C = (20\ ns, 0) < EIMT_C = (20\ ns, 1)$

Challenges

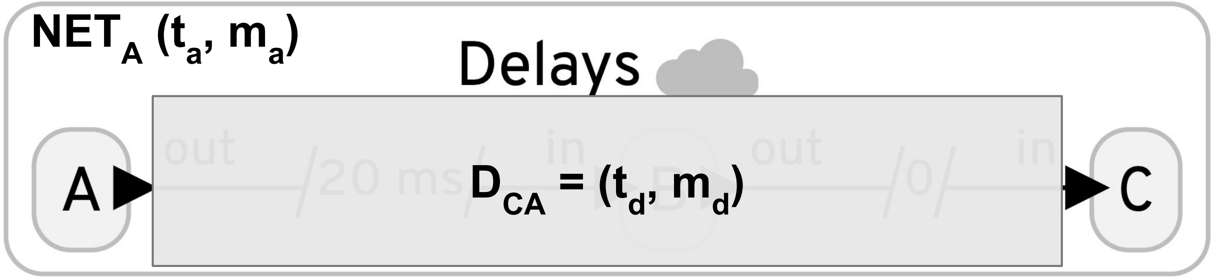


- Compute $EIMT_C$ = **(t, m)** when A's next event tag is **(t_a, m_a)** and other federates do not have any events

$$(t, m) = A((t_a, m_a), (t_d, m_d))$$

$$= \begin{cases} (t_A, m_A + m_D), & \text{if } t_D = 0 \\ (t_A + t_D, m_D), & \text{if } t_D > 0 \end{cases}$$

Challenges

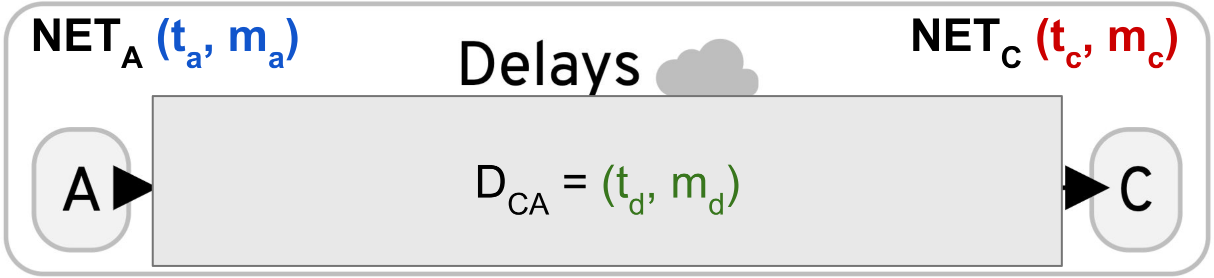


- Compute $EIMT_C$ = **(t, m)** when A's next event tag is (t_a, m_a) and other federates do not have any events

$$(t, m) = A((t_a, m_a), (t_d, m_d))$$

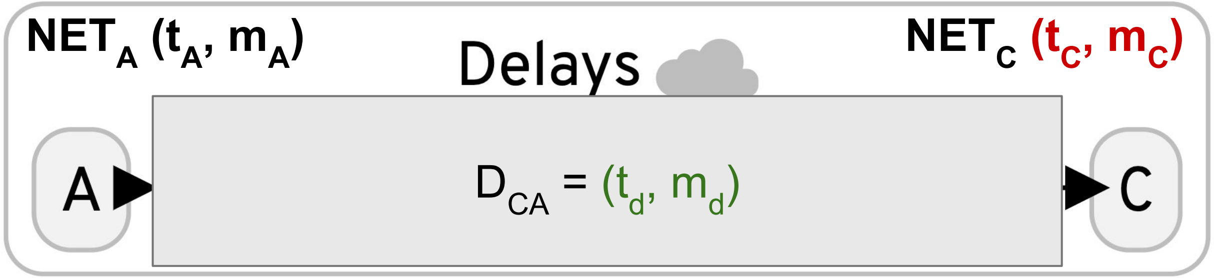
$$= \begin{cases} (t_a, m_a + m_d), & \text{if } t_d = 0 \wedge m_a + m_d < M_{\max} \\ (t_a, M_{\max}), & \text{if } t_d = 0 \wedge m_a + m_d \geq M_{\max} \\ (t_a + t_d, m_d), & \text{if } t_d > 0 \wedge t_a + t_d < \infty \end{cases}$$

DNET & Delays



- $A((t_a, m_a), (t_d, m_d)) > (t_c, m_c)$ to send TAG_C (t_c, m_c)
- NET_A (t_a, m_a) is unnecessary if $A((t_a, m_a), (t_d, m_d)) <= (t_c, m_c)$
- Define a function S to find the **latest** tag that satisfies
  - $A((t, m), (t_d, m_d)) <= (t_c, m_c)$ where $(t, m) = S((t_c, m_c), (t_d, m_d))$
  - S acts like tag subtraction (X + D = C if X = C - D)

DNET & Delays



- Define a function S to find the **latest** tag that satisfies
  - **$A((t, m), (t_d, m_d)) <= (t_c, m_c)$** where $(t, m) = S((t_c, m_c), (t_d, m_d))$
  - S acts like tag subtraction (X = C - D -> X + D = C)

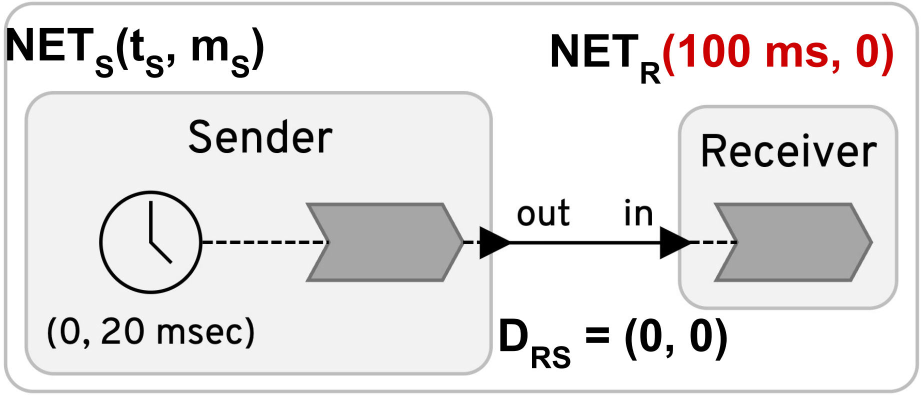
$$(t, m) = S((t_C, m_C), (t_d, m_d))$$

$$= \begin{cases} (t_c - t_d, m_c - m_d), & \text{if } t_c \geq t_d = 0 \wedge m_c \geq m_d \\ (t_c - t_d, M_{\max}), & \text{if } t_c \geq t_d > 0 \wedge m_c \geq m_d \\ (t_c - t_d - 1, M_{\max}), & \text{if } t_c > t_d > 0 \wedge m_c < m_d \\ (-\infty,\ 0), & \text{if } t_c = -\infty \vee (t_c, m_c) < (t_d, m_d) \\ (\infty,\ M_{\max}) & \text{if } t_c = \infty \end{cases}$$

DNET & Delays

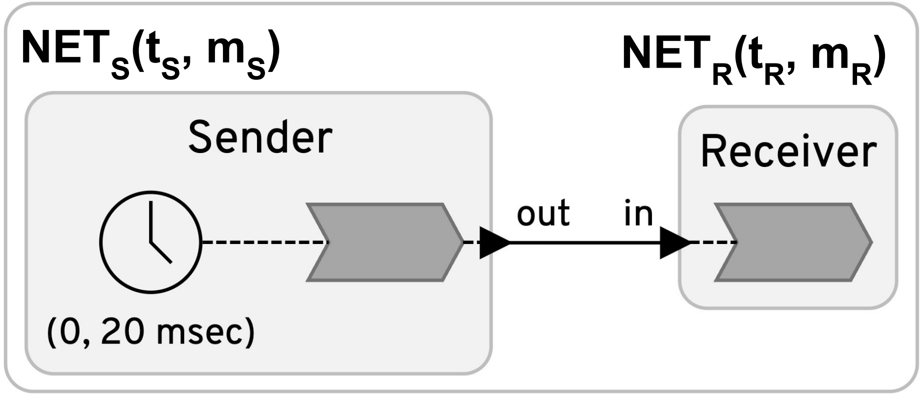
$S((t_C, m_C), (t_d, m_d))$
$=(t_c - t_d, m_c - m_d)$ if $t_c \geq t_d = 0 \wedge m_c \geq m_d$

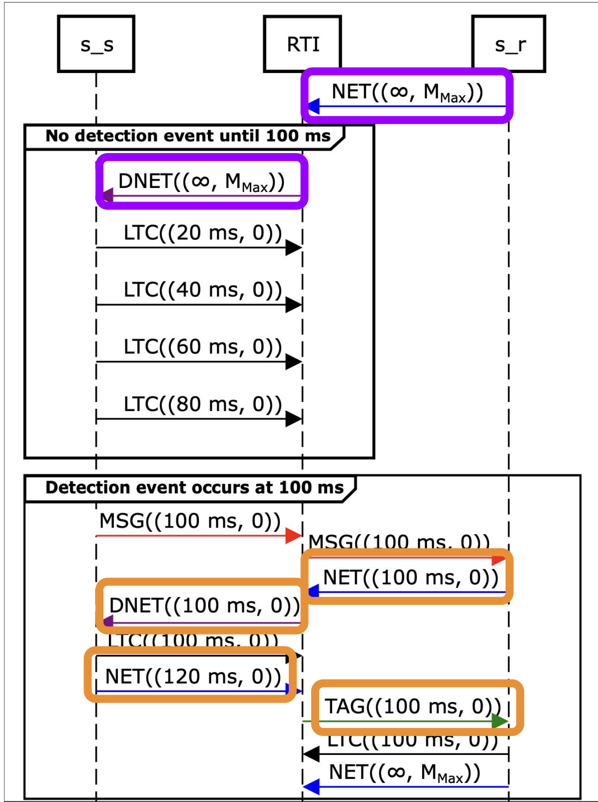
- G(DNET) is the **latest** tag that a federate doesn't need to send NET
- R has an event at **(100 ms, 0)**, what NET_S **(t_S, m_S)** is **unnecessary** to grant TAG_R **(100 ms, 0)** to R?
    - **(t_S, m_S) <= S(G(NET_R), D_{RS}) = S((100 ms, 0), (0, 0)) =(100 ms, 0)**



DNET & Delays

- Sender skips sending NET with tags < G($DNET_S$), where G($DNET_S$) is **S(G(NET_R), D_{RS})**





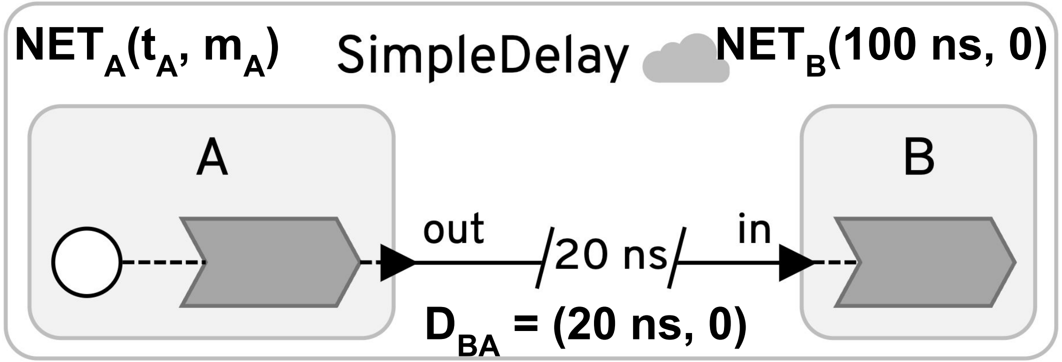
DNET & Delays

$$S((t_C, m_C), (t_d, m_d)) = (t_c - t_d, M_{\max}), \quad \text{if } t_c \geq t_d > 0 \wedge m_c \geq m_d$$

- $G(DNET_A)$ = **$S((100\text{ ns}, 0), (20\text{ ns}, 0))$** = **$(80\text{ ns}, M_{max})$**
  - If $(t_A, m_A) = G(DNET)$ = **$(80\text{ ns}, M_{max})$**

    $EIMT_B$ = **$A((80\text{ ns}, M_{max}), (20\text{ ns}, 0)) = (100\text{ ns}, 0)$** $\le G(NET_B)$
  - If (t_A, m_A) = **$(81\text{ ns}, 0)$**

    $EIMT_B$ = **$A((81\text{ ns}, 0), (20\text{ ns}, 0)) = (101\text{ ns}, 0) > G(NET_B)$**



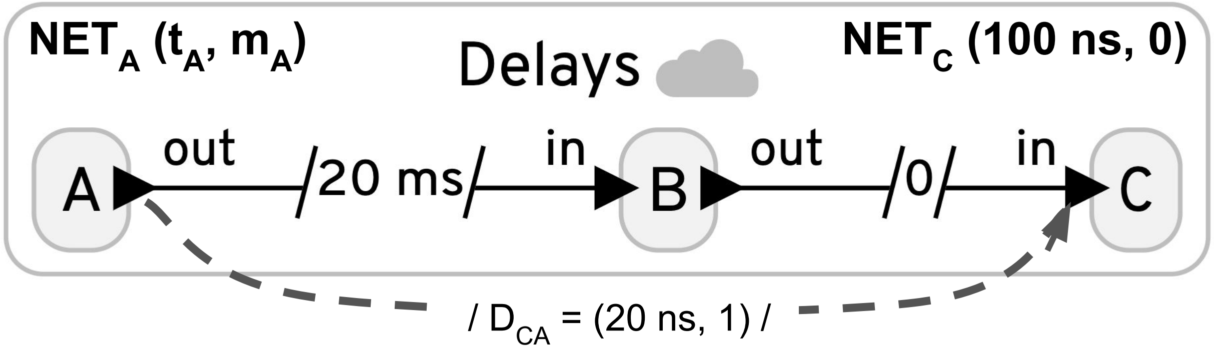
DNET & Delays

$$S((t_C, m_C), (t_d, m_d)) = (t_c - t_d - 1, M_{\max}), \text{ if } t_c > t_d > 0 \land m_c < m_d$$

- $G(DNET_C)$ = **S((100 ns, 0), (20 ns, 1))** = **(79 ns, M_{max})**
  - If (t_A, m_A) = G(DNET) = **(79 ns, M_{max})**

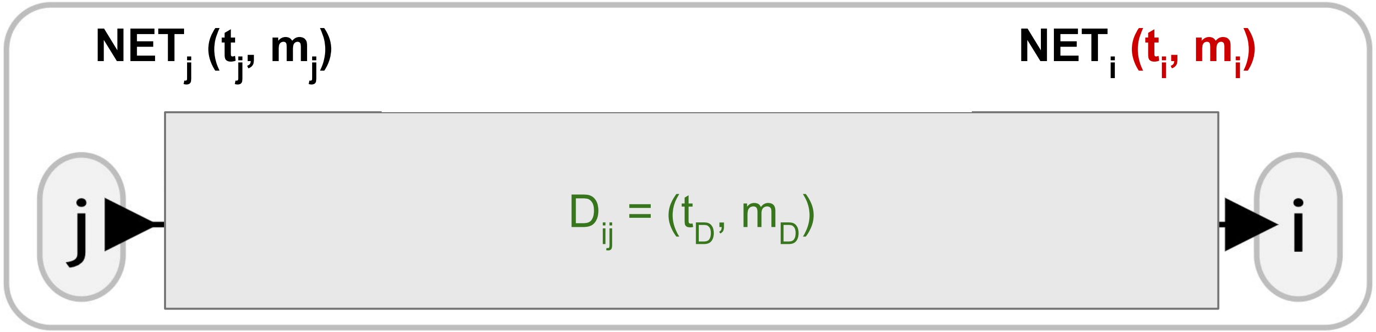
    $EIMT_C$ = **A((79 ns, M_{max}), (20 ns, 1)) = (99 ns, 1)** <= $G(NET_C)$
  - If (t_A, m_A) = **(80 ns, 0)**

    $EIMT_C$ = **A((80 ns, 0), (20 ns, 1)) = (100 ns, 1)** > $G(NET_C)$

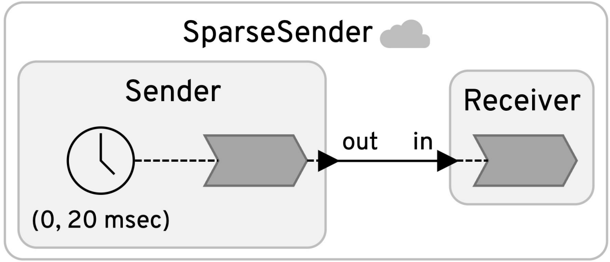


DNET Computation

- How to compute $DNET_j$?
  - Look up every downstream federate
  - For each downstream federate i, find **S((t$_i$, m$_i$), (t$_D$, m$_D$))**, the latest tag g satisfying **A(g, (t$_D$, m$_D$)) <= (t$_i$, m$_i$)**
  - Determine $G(DNET_j)$ as the minimum **S((t$_i$, m$_i$), (t$_D$, m$_D$))**



Evaluation

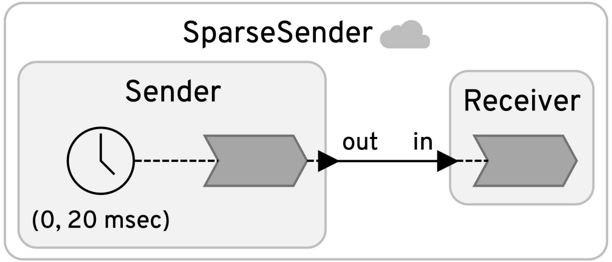


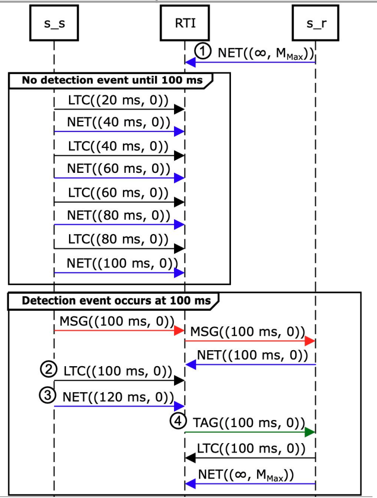
- Sender produces outputs (MSG_{RS}) sparsely
  - We assume Sender sends messages every 5 seconds
  - Total execution time is 500 seconds
- Counting the number of NET signals while varying the period of the timer

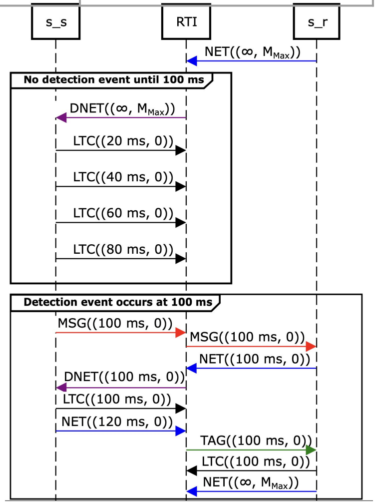
Evaluation

- Number of NET Signals

Timer period	5 ms	10 ms	20 ms	50 ms	100 ms
Without DNET (Baseline)	100,161	50,191	25,193	10,195	5,195
With DNET	677	385	301	288	297

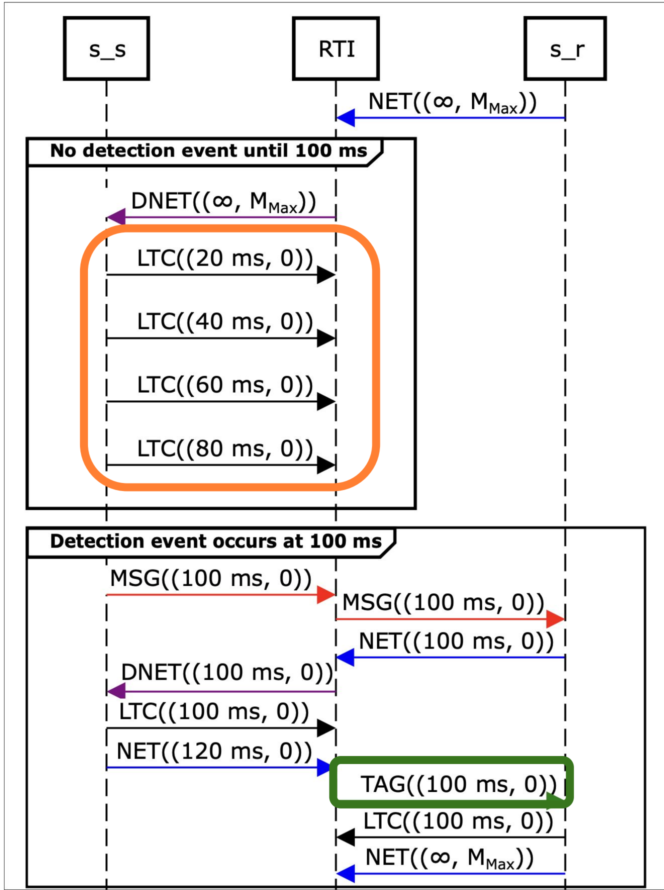






Work-In-Progress

- Some LTC and TAG signals are also unnecessary
- These can affect a program's feasibility



Future Work

- Our solution's effectiveness varies with the sender's sparsity or programs' structure
  - When a sender sends messages every time, DNET is not needed
  - If a federate has too many upstream federates and have lots of events, DNET may flood
- Dynamic control of DNET is needed to maximize its benefit
  - Set a threshold of events without producing any messages

Conclusion

- Our solution effectively reduces the network overhead of HLA-based discrete event systems
- This is beneficial to systems that require precise timing control where network communication cost is high
- Our future work further optimizes the network overhead of these kind of systems

Thank you!