Model-based High-level Integration of Heterogeneous Components

Jawher Jerray[1], Rabéa Ameur-Boulifa[1] and Ludovic Apvrille[1]

[1] LTCI, Télécom Paris, Institut Polytechnique de Paris, Sophia-Antipolis, France.

ICSOFT Conference
Wednesday 12th July, 2023

Context

Embedded systems

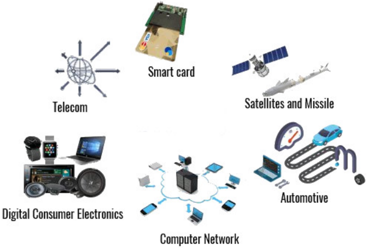
- Made up of hardware and software.
- Perform specific tasks.

Context

Embedded systems

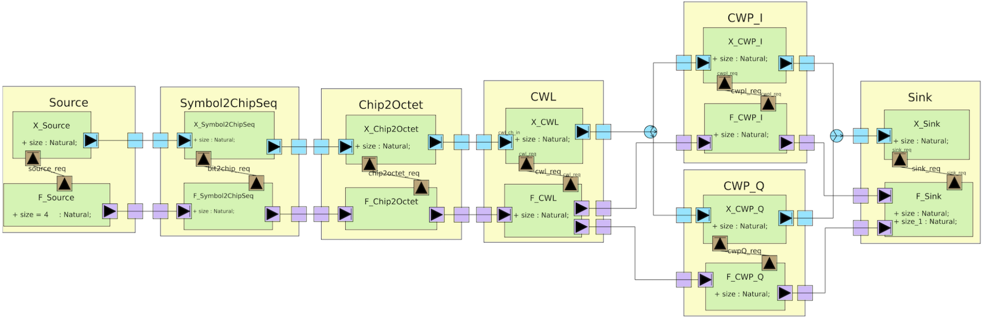
- Made up of hardware and software.
- Perform specific tasks.

- formally verifying the well functioning of embedded systems.
- modeling and designing complex embedded systems.



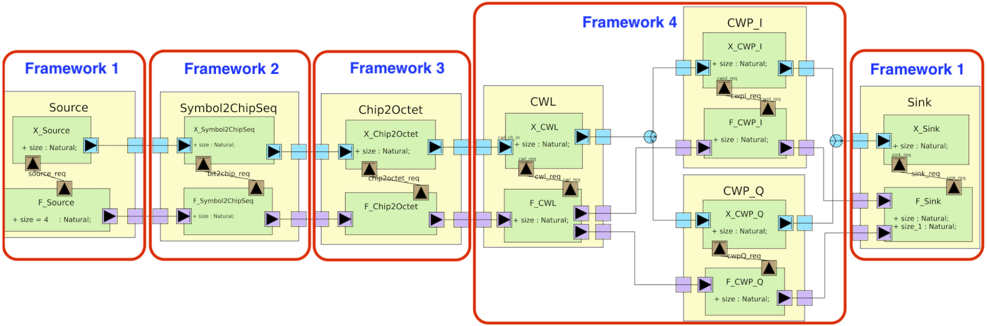
Problematic

- An embedded system can contain many complex components.



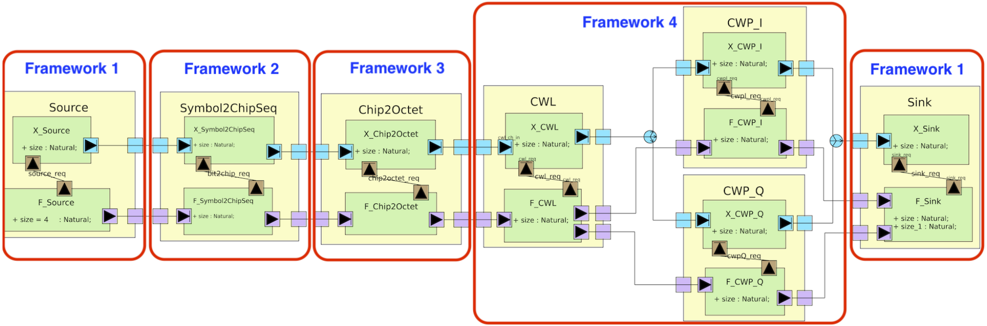
Problematic

- An embedded system can contain many complex components.
- Several frameworks might be used to model and simulate those components.



Problematic

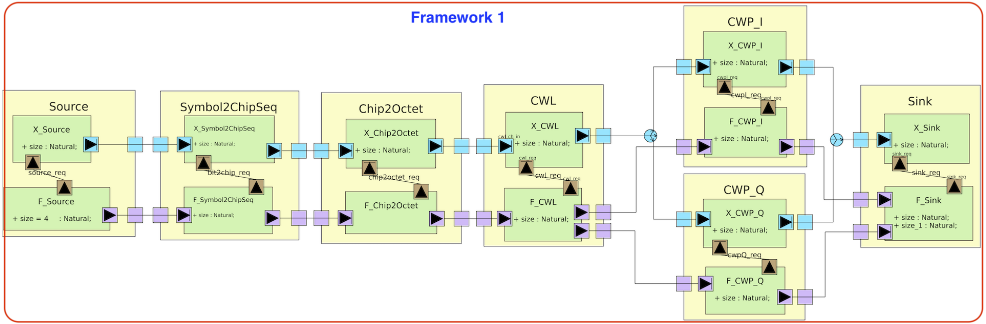
- An embedded system can contain many complex components.
- Several frameworks might be used to model and simulate those components.



- Question: How to maintain the communication between those heterogeneous components (Different semantics between frameworks)?

Problematic

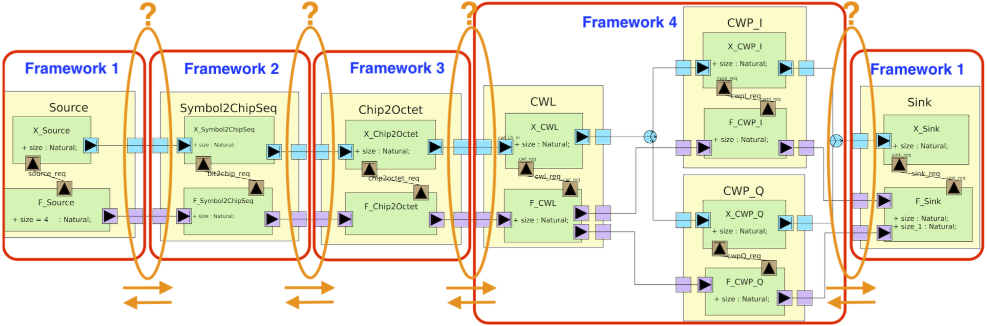
- An embedded system can contain many complex components.
- Several frameworks might be used to model and simulate those components.



- Possible solution: Translate everything in a single framework?

Problematic

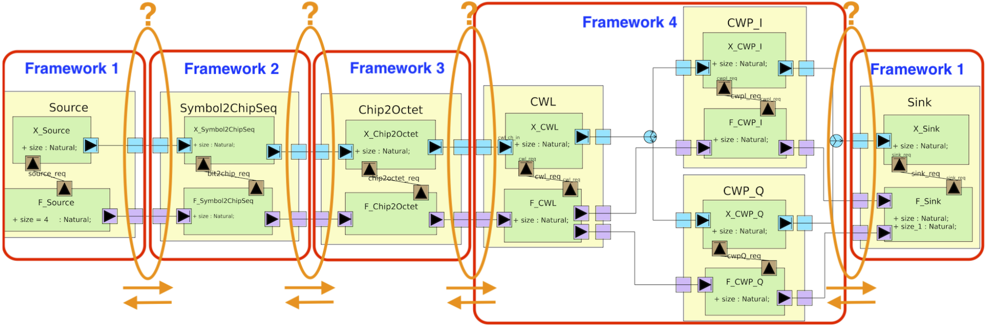
- An embedded system can contain many complex components.
- Several frameworks might be used to model and simulate those components.



- Other possible solution: Co-simulation for specific frameworks as represented in previous works.

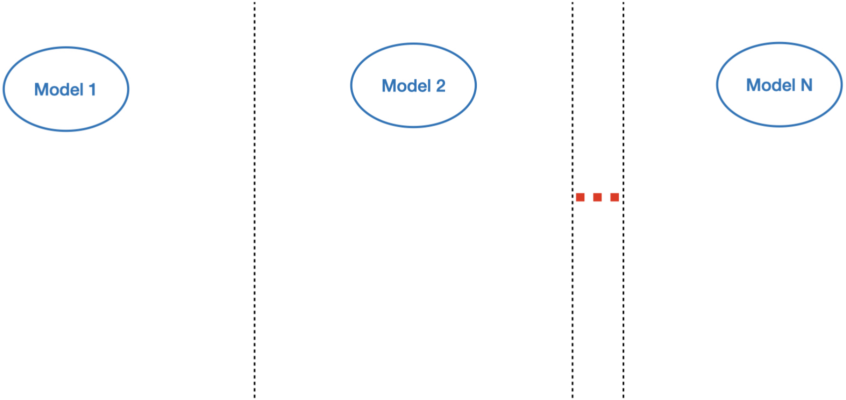
Problematic

- An embedded system can contain many complex components.
- Several frameworks might be used to model and simulate those components.

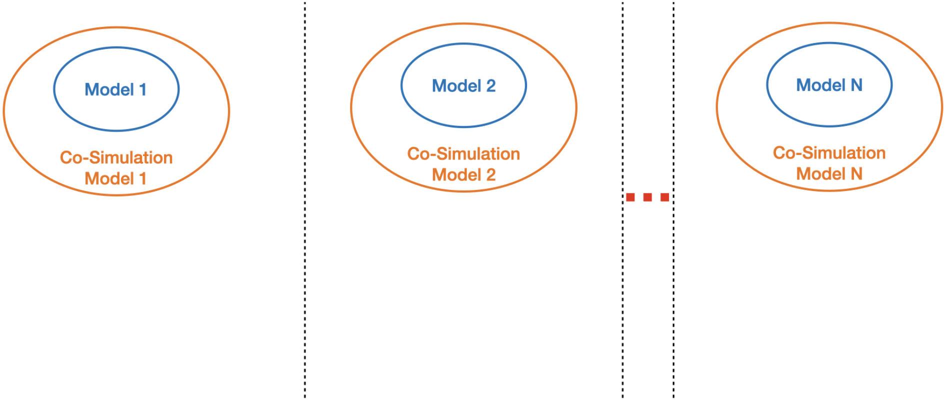


- Our solution: a generic simulation glue that allows to join heterogeneous components together using a distributed event streaming platform.

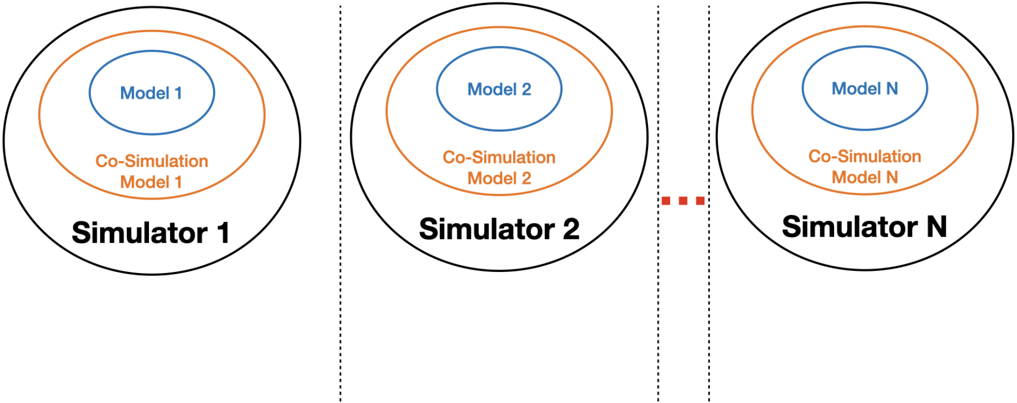
Proposed approach



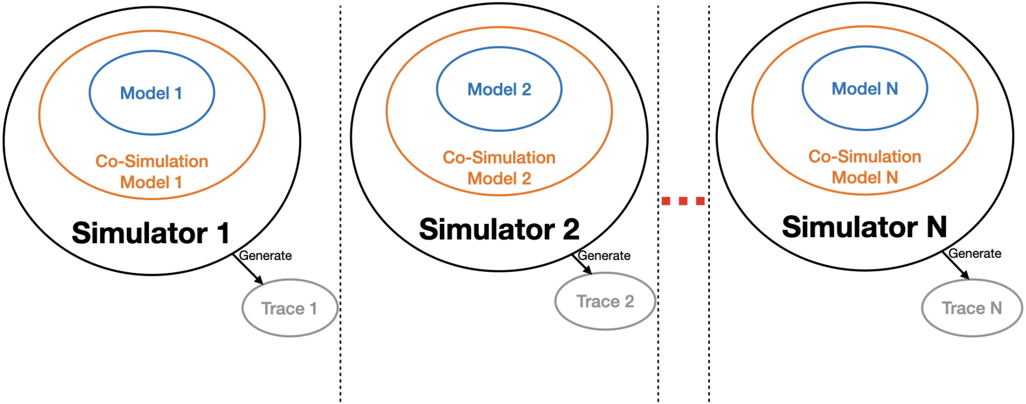
Proposed approach



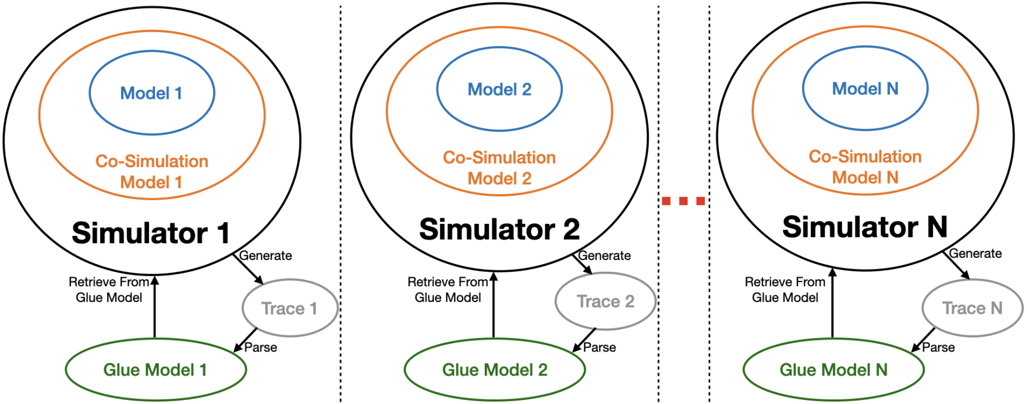
Proposed approach



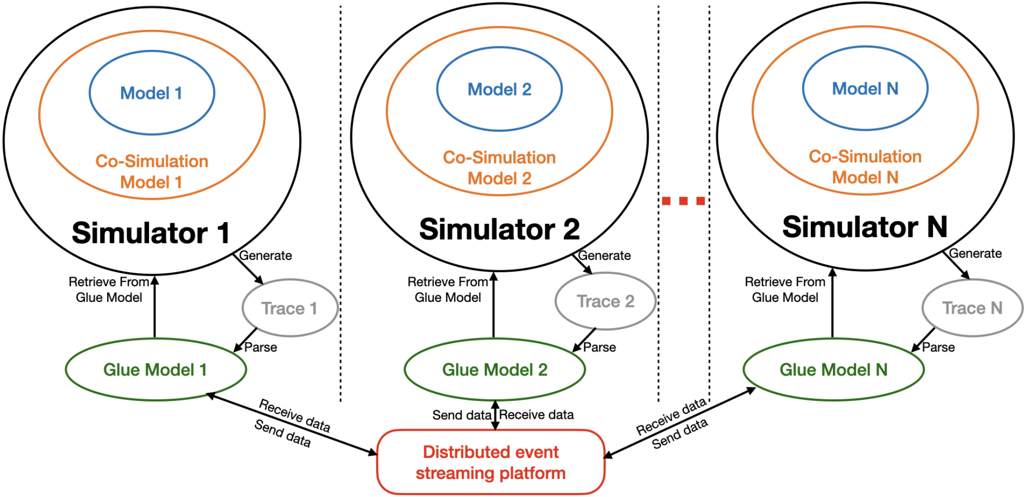
Proposed approach



Proposed approach

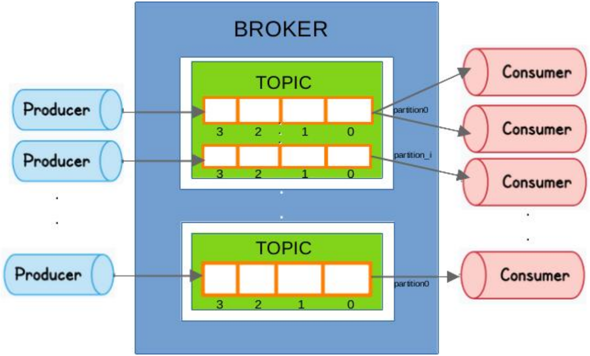


Proposed approach



Kafka Brocker

- Kafka is a distributed event streaming platform that aims to send and receive messages between entities.
- It can be used to guarantee the communication between two models designed and executed with different simulation techniques.

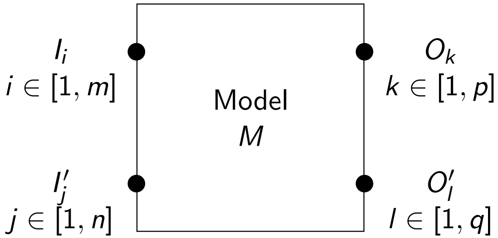


Kafka server

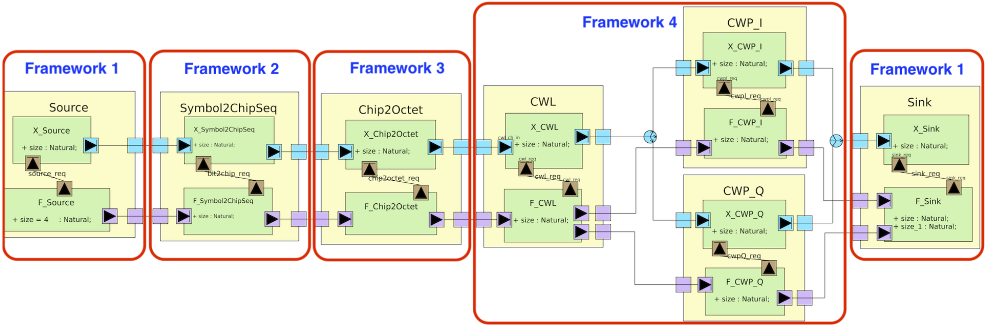
Model

- A model M contains 4 types of ports:
  - Internal input ports I_i
  - Internal output ports O_k
  - External input ports I'_j
  - External output ports O'_l

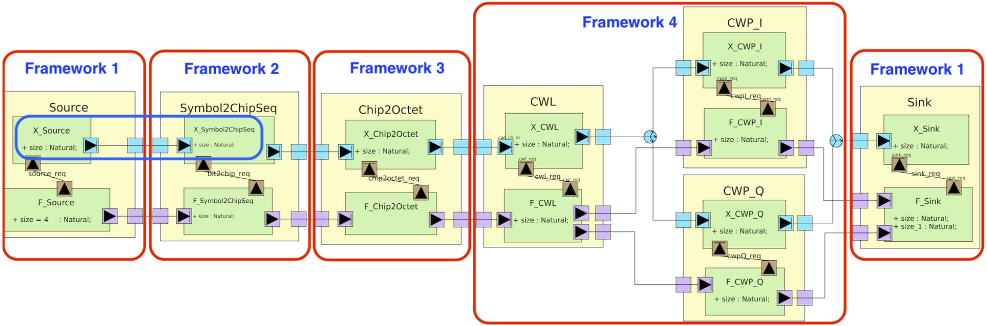
$\}$ are used to exchange data with other models via Kafka.



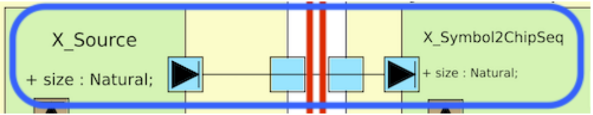
Example of communication via Kafka: BRBW Channel



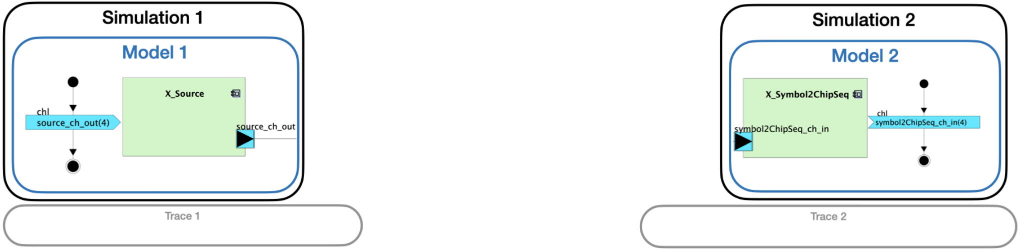
Example of communication via Kafka: BRBW Channel



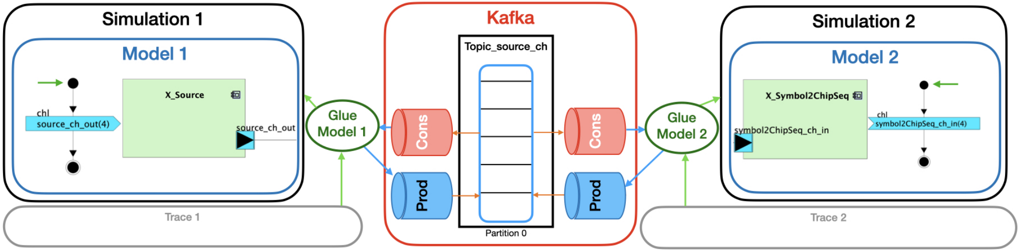
Example of communication via Kafka: BRBW Channel



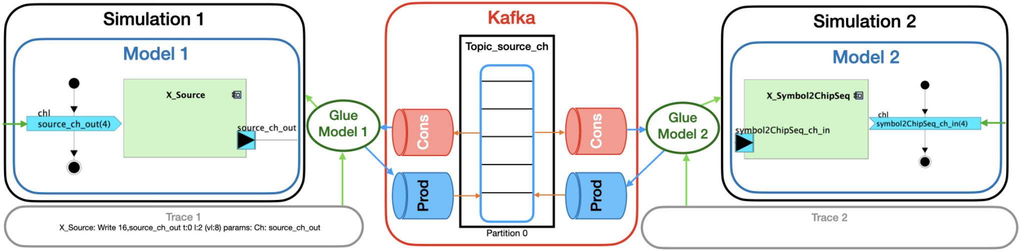
Example of communication via Kafka: BRBW Channel



Example of communication via Kafka: BRBW Channel

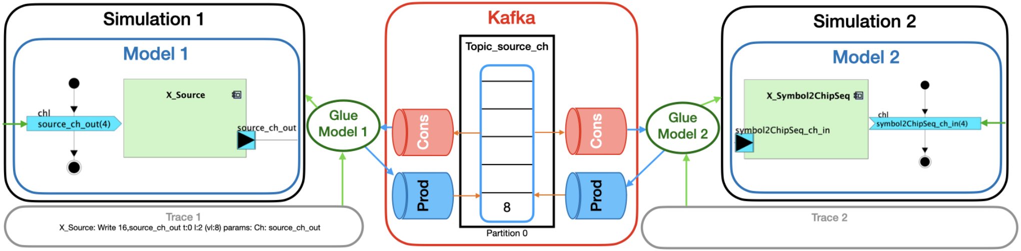


Example of communication via Kafka: BRBW Channel



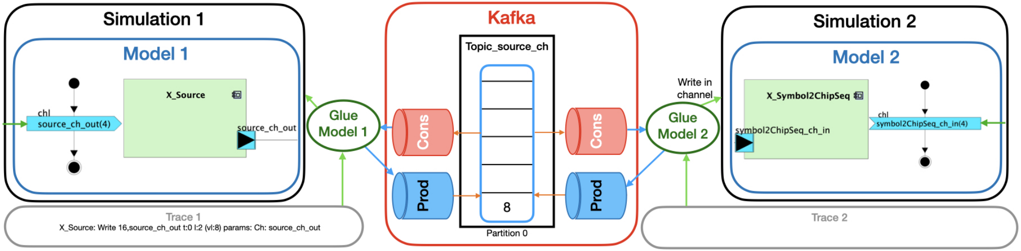
```
X_Source: Write 16,source_ch_out t:0 l:2 (vl:8) params: Ch: source_ch_out
```

Example of communication via Kafka: BRBW Channel



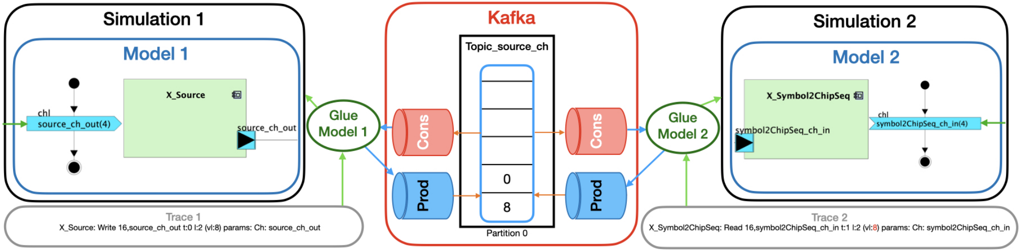
X_Source: Write 16,source_ch_out t:0 l:2 (vl:8) params: Ch: source_ch_out

Example of communication via Kafka: BRBW Channel



```
X_Source: Write 16,source_ch_out t:0 l:2 (vl:8) params: Ch: source_ch_out
```

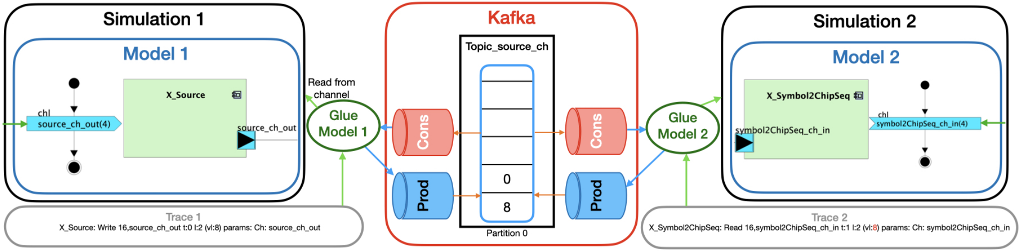
Example of communication via Kafka: BRBW Channel



```
X_Source: Write 16,source_ch_out t:0 l:2 (vl:8) params: Ch: source_ch_out
```

```
X_Symbol2ChipSeq: Read 16,symbol2ChipSeq_ch_in t:1 l:2 (vl:8) params: Ch: symbol2ChipSeq_ch_in
```

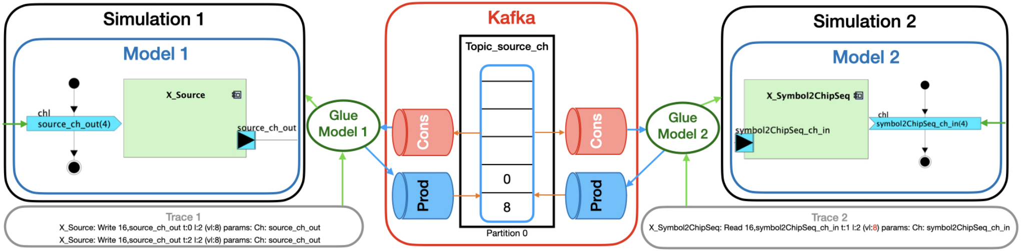
Example of communication via Kafka: BRBW Channel



X_Source: Write 16,source_ch_out t:0 l:2 (vl:8) params: Ch: source_ch_out

X_Symbol2ChipSeq: Read 16,symbol2ChipSeq_ch_in t:1 l:2 (vl:8) params: Ch: symbol2ChipSeq_ch_in

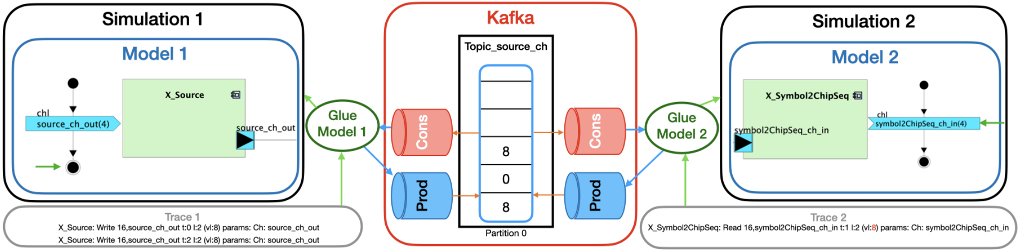
Example of communication via Kafka: BRBW Channel



```
X_Source: Write 16,source_ch_out t:0 l:2 (vl:8) params: Ch: source_ch_out
X_Source: Write 16,source_ch_out t:2 l:2 (vl:8) params: Ch: source_ch_out
```

```
X_Symbol2ChipSeq: Read 16,symbol2ChipSeq_ch_in t:1 l:2 (vl:8) params: Ch: symbol2ChipSeq_ch_in
```

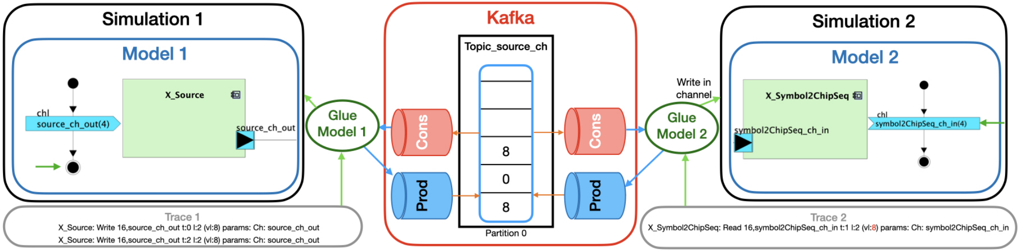
Example of communication via Kafka: BRBW Channel



```
X_Source: Write 16,source_ch_out t:0 l:2 (vl:8) params: Ch: source_ch_out
X_Source: Write 16,source_ch_out t:2 l:2 (vl:8) params: Ch: source_ch_out
```

```
X_Symbol2ChipSeq: Read 16,symbol2ChipSeq_ch_in t:1 l:2 (vl:8) params: Ch: symbol2ChipSeq_ch_in
```

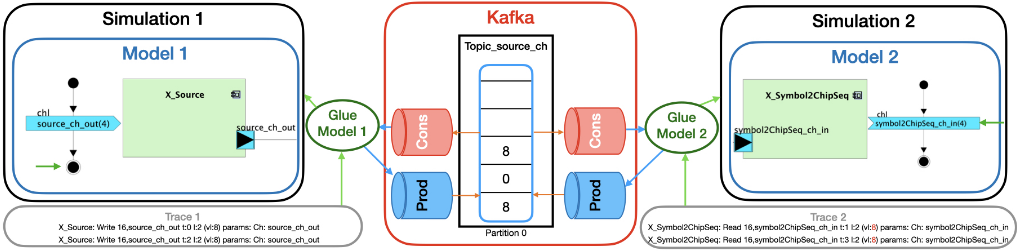
Example of communication via Kafka: BRBW Channel



```
X_Source: Write 16,source_ch_out t:0 l:2 (vl:8) params: Ch: source_ch_out
X_Source: Write 16,source_ch_out t:2 l:2 (vl:8) params: Ch: source_ch_out
```

```
X_Symbol2ChipSeq: Read 16,symbol2ChipSeq_ch_in t:1 l:2 (vl:8) params: Ch: symbol2ChipSeq_ch_in
```

Example of communication via Kafka: BRBW Channel



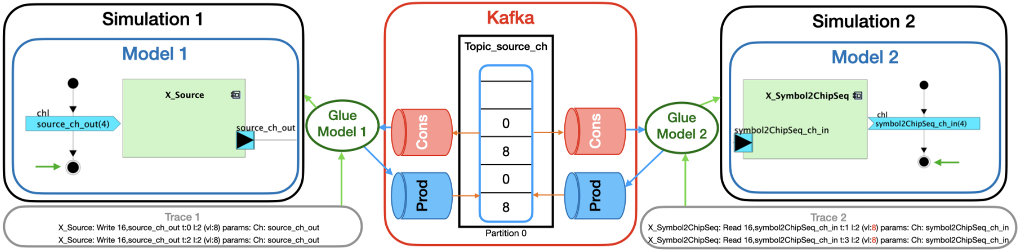
Trace 1

```
X_Source: Write 16,source_ch_out t:0 l:2 (vl:8) params: Ch: source_ch_out
X_Source: Write 16,source_ch_out t:2 l:2 (vl:8) params: Ch: source_ch_out
```

Trace 2

```
X_Symbol2ChipSeq: Read 16,symbol2ChipSeq_ch_in t:1 l:2 (vl:8) params: Ch: symbol2ChipSeq_ch_in
X_Symbol2ChipSeq: Read 16,symbol2ChipSeq_ch_in t:3 l:2 (vl:8) params: Ch: symbol2ChipSeq_ch_in
```

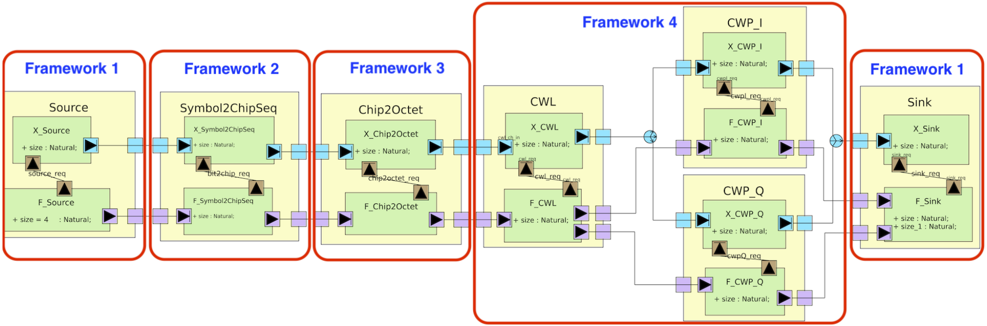
Example of communication via Kafka: BRBW Channel



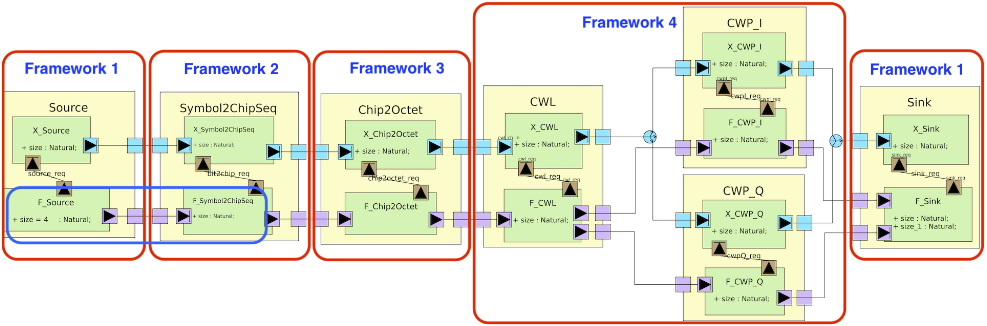
```
X_Source: Write 16,source_ch_out t:0 l:2 (vl:8) params: Ch: source_ch_out
X_Source: Write 16,source_ch_out t:2 l:2 (vl:8) params: Ch: source_ch_out
```

```
X_Symbol2ChipSeq: Read 16,symbol2ChipSeq_ch_in t:1 l:2 (vl:8) params: Ch: symbol2ChipSeq_ch_in
X_Symbol2ChipSeq: Read 16,symbol2ChipSeq_ch_in t:3 l:2 (vl:8) params: Ch: symbol2ChipSeq_ch_in
```

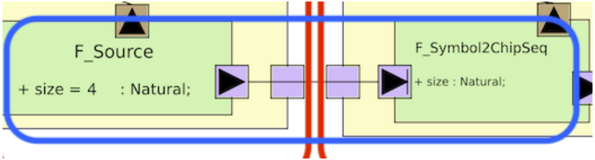
Example of communication via Kafka: BRBW Event



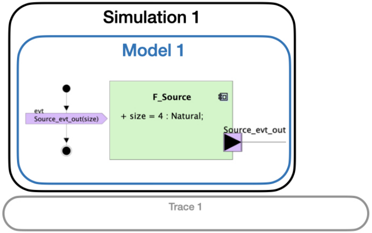
Example of communication via Kafka: BRBW Event

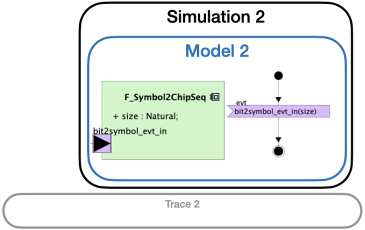


Example of communication via Kafka: BRBW Event

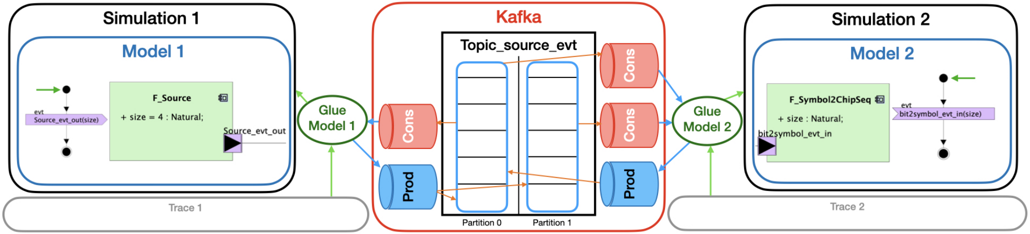


Example of communication via Kafka: BRBW Event

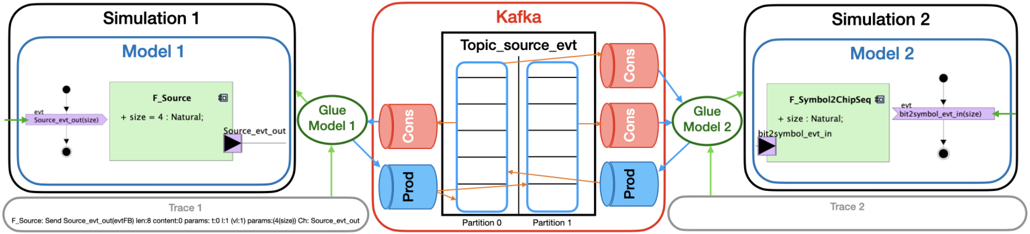




Example of communication via Kafka: BRBW Event

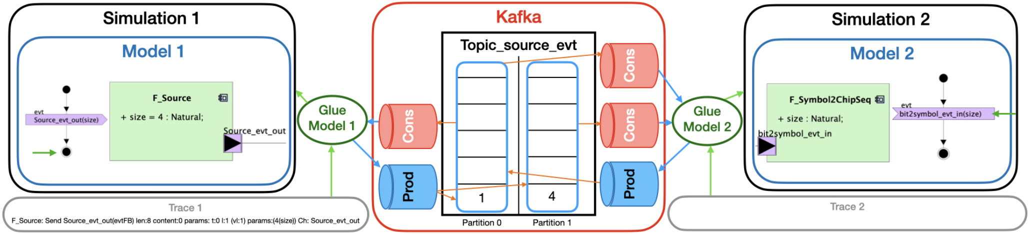


Example of communication via Kafka: BRBW Event



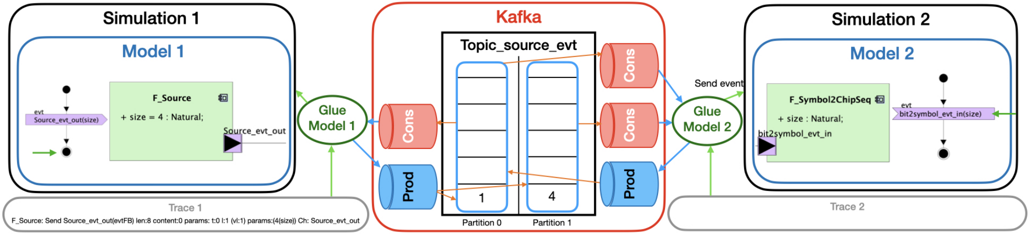
```
F_Source: Send Source_evt_out(evtFB) len:8 content:0 params: t:0 l:1 (vl:1) params:(4(size)) Ch: Source_evt_out
```

Example of communication via Kafka: BRBW Event



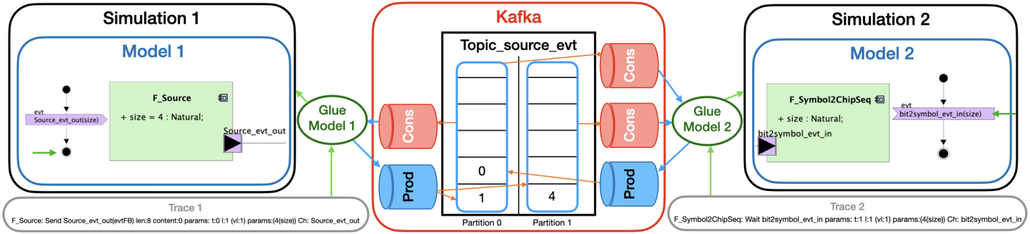
```
F_Source: Send Source_evt_out(evtFB) len:8 content:0 params: t:0 l:1 (vl:1) params:(4(size)) Ch: Source_evt_out
```

Example of communication via Kafka: BRBW Event



```
F_Source: Send Source_evt_out(evtFB) len:8 content:0 params: t:0 l:1 (vl:1) params:(4(size)) Ch: Source_evt_out
```

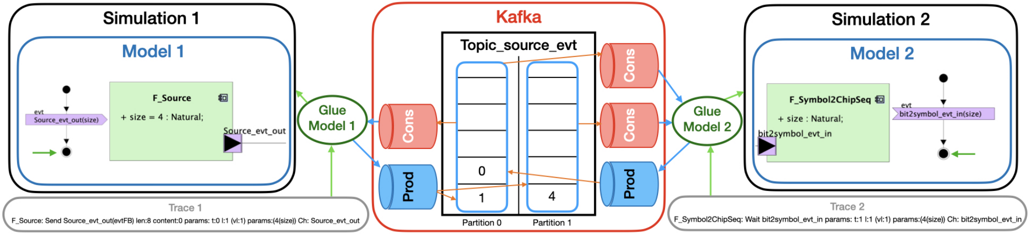
Example of communication via Kafka: BRBW Event



```
F_Source: Send Source_evt_out(evtFB) len:8 content:0 params: t:0 l:1 (vl:1) params:(4(size)) Ch: Source_evt_out
```

```
F_Symbol2ChipSeq: Wait bit2symbol_evt_in params: t:1 l:1 (vl:1) params:(4(size)) Ch: bit2symbol_evt_in
```

Example of communication via Kafka: BRBW Event



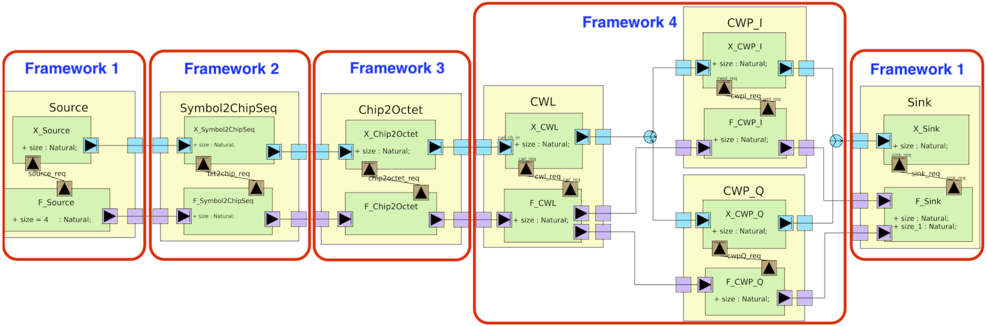
Trace 1

```
F_Source: Send Source_evt_out(evtFB) len:8 content:0 params: t:0 l:1 (vl:1) params:(4(size)) Ch: Source_evt_out
```

Trace 2

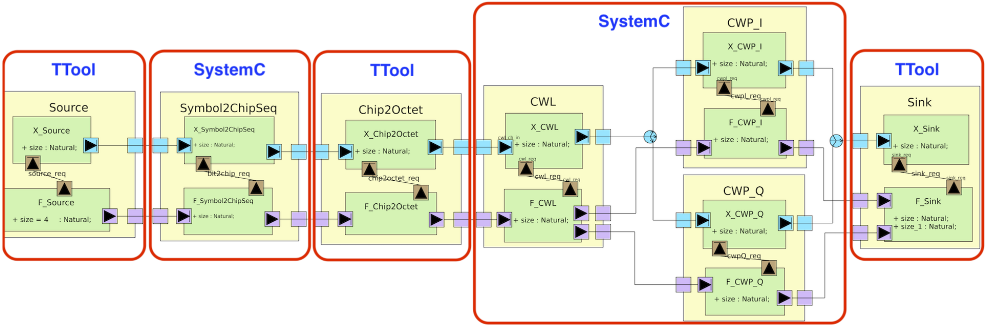
```
F_Symbol2ChipSeq: Wait bit2symbol_evt_in params: t:1 l:1 (vl:1) params:(4(size)) Ch: bit2symbol_evt_in
```

ZigBee transmitter example



ZigBee transmitter example

ZigBee transmitter example



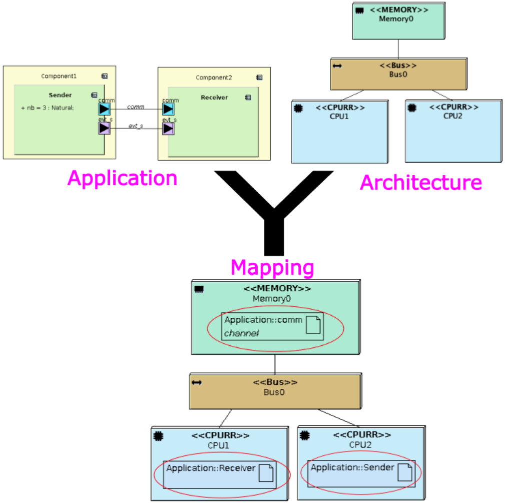
ZigBee transmitter example

TTool/DIPLODOCUS

TTool/DIPLODOCUS

TTool is a tool for partitioning embedded systems and simulating models.

- Application model: components, tasks, ports, . . .
- Architecture model: CPU, buses, memories, . . .
- Mapping model: assigns tasks and channels onto the architecture.



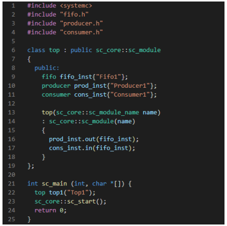
SystemC

SystemC

SystemC is a C++ library that extends the language to design systems on a chip (SoC) and perform verification.

- SystemC introduces several data types which support hardware modeling.
- SystemC model is composed of modules which communicate via ports.

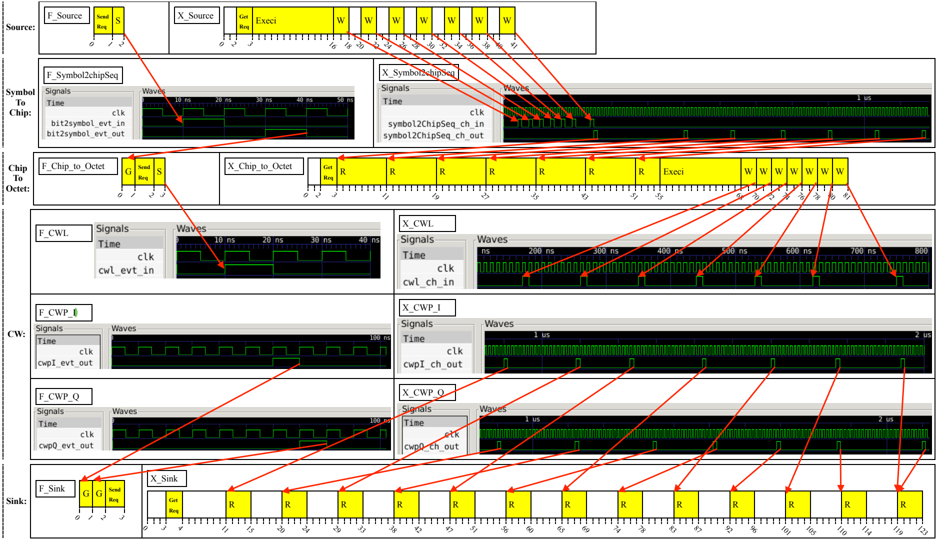
SystemC



```
#include <systemc>
#include "fifo.h"
#include "producer.h"
#include "consumer.h"

class top : public sc_core::sc_module
{
  public:
    fifo fifo_inst{"Fifo1"};
    producer prod_inst{"Producer1"};
    consumer cons_inst{"Consumer1"};

    top(sc_core::sc_module_name name)
    : sc_core::sc_module(name)
    {
      prod_inst.out(fifo_inst);
      cons_inst.in(fifo_inst);
    }
};

int sc_main (int, char *[]) {
  top top1("Top1");
  sc_core::sc_start();
  return 0;
}
```

SystemC model

ZigBee transmitter co-simulation results



Conclusion and Future Work

Conclusion

- We presented a technique that allows to integrate heterogeneous components.
- We ensured the communication between components modeled by TTool and SystemC which have different semantics using a distributed event streaming platform.

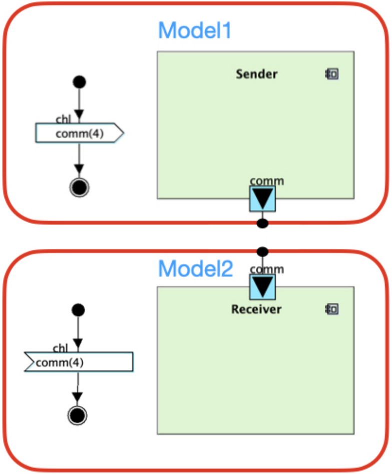
Future Work

- We envisage to use Socket instead of Kafka to get better latency.
- Build a global simulation trace for a model with heterogeneous components.

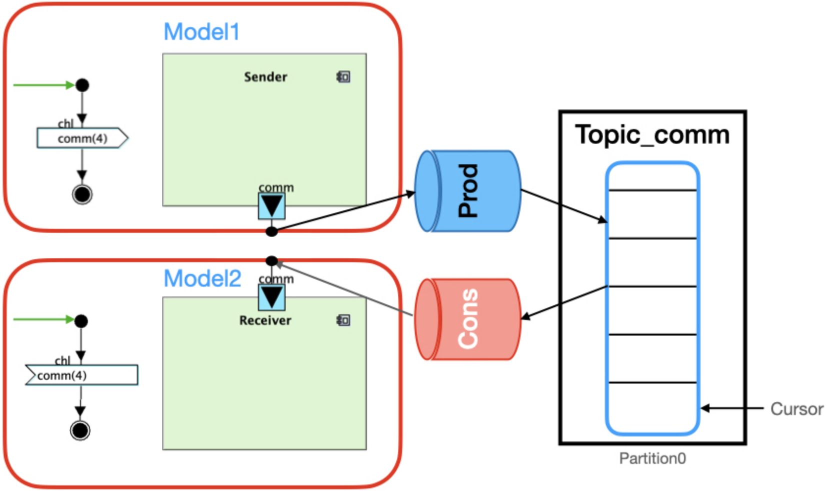
Thank you for your attention!

Questions?

Example of communication via Kafka: NBRNBW Channel

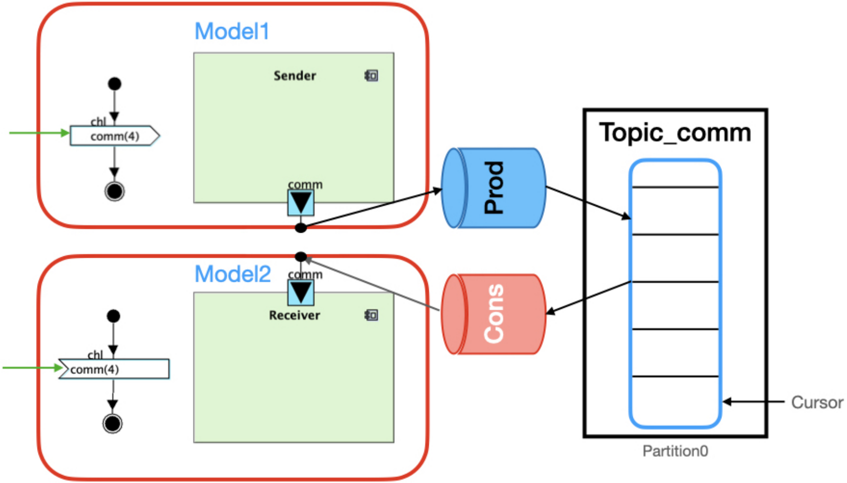


Example of communication via Kafka: NBRNBW Channel



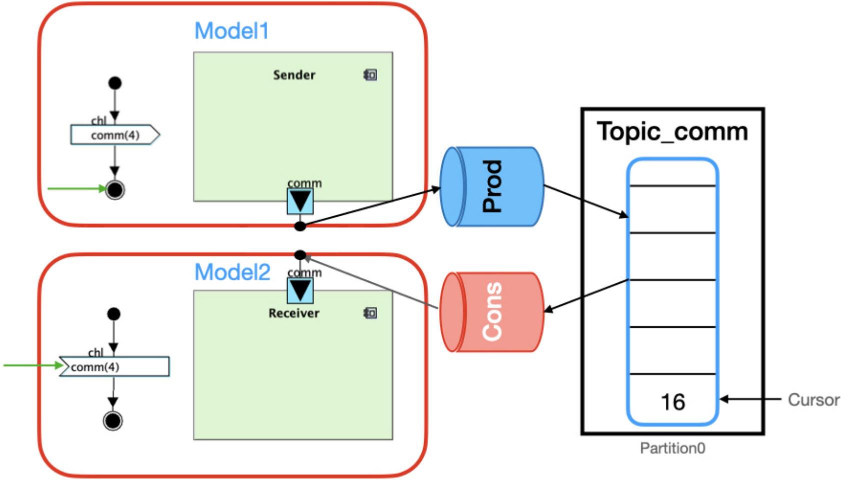
Example of communication via Kafka: NBRNBW Channel

Application__Sender: Write 16,Application__comm t:0 l:4 (vl:16) params: Ch: Application__comm



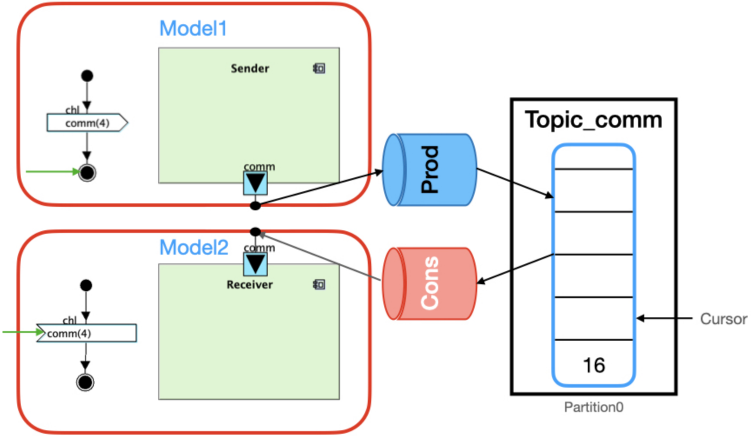
Example of communication via Kafka: NBRNBW Channel

Application__Sender: Write 16,Application__comm t:0 l:4 (vl:16) params: Ch: Application__comm



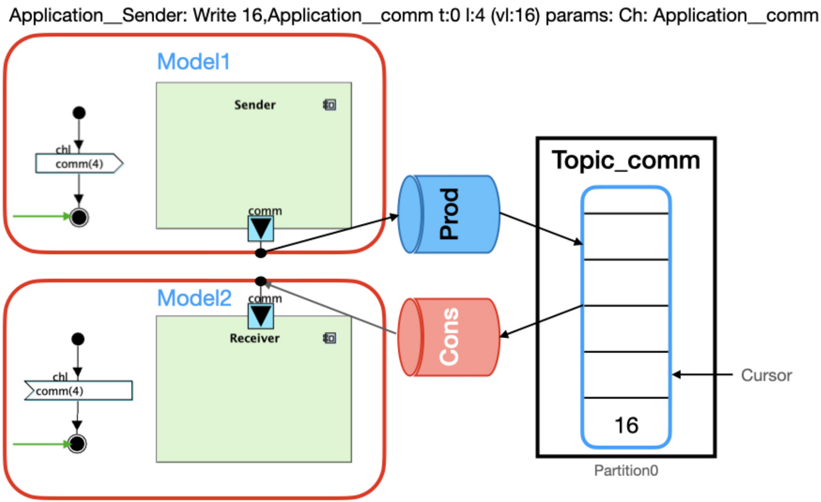
Example of communication via Kafka: NBRNBW Channel

Application__Sender: Write 16,Application__comm t:0 l:4 (vl:16) params: Ch: Application__comm



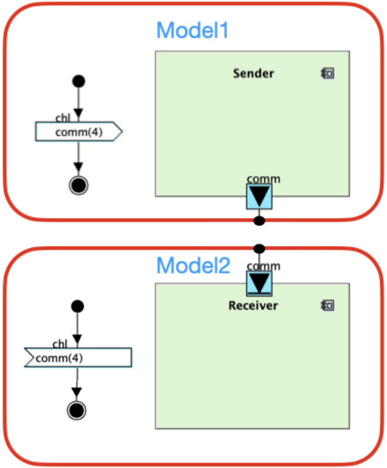
wic comm 16

Example of communication via Kafka: NBRNBW Channel

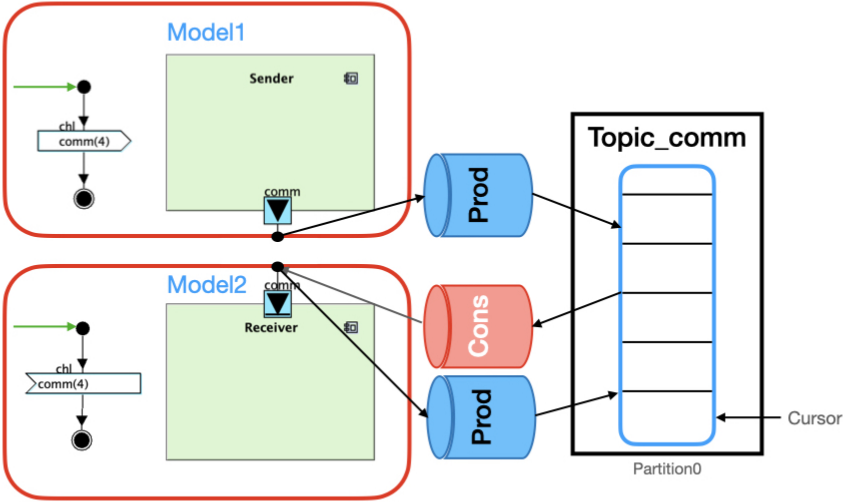


Application__Sender: Write 16,Application__comm t:0 l:4 (vl:16) params: Ch: Application__comm

Example of communication via Kafka: BRNBW Channel

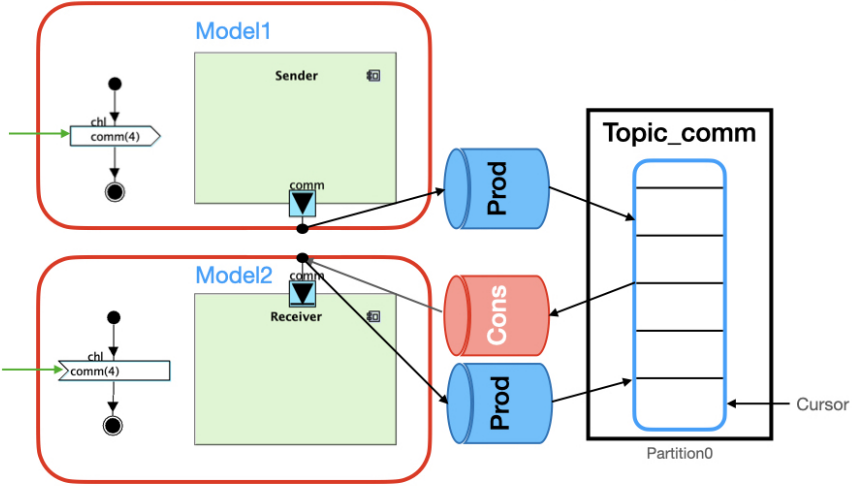


Example of communication via Kafka: BRNBW Channel



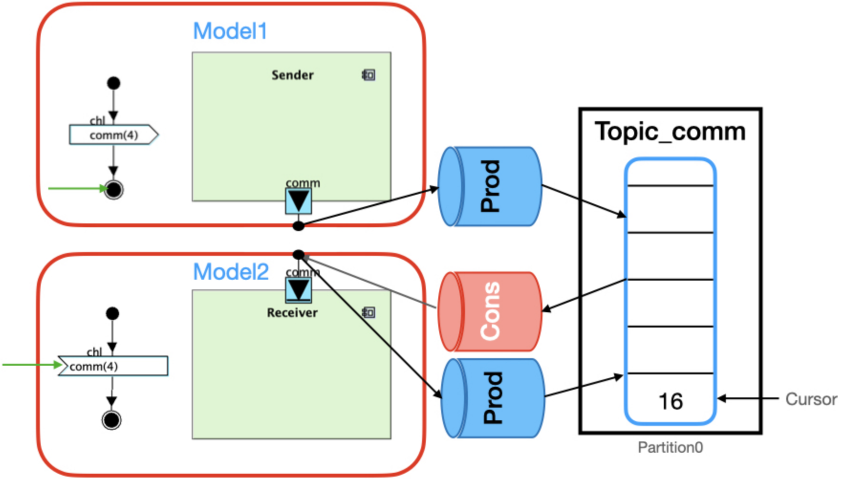
Example of communication via Kafka: BRNBW Channel

Application__Sender: Write 16,Application__comm t:0 l:4 (vl:16) params: Ch: Application__comm



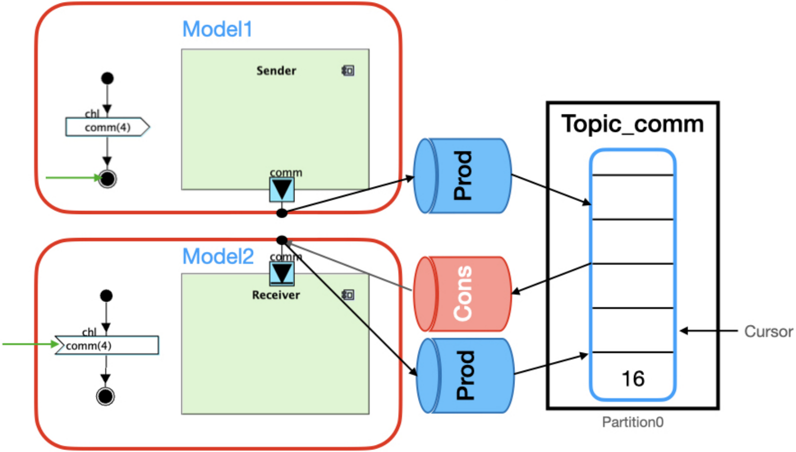
Example of communication via Kafka: BRNBW Channel

Application__Sender: Write 16,Application__comm t:0 l:4 (vl:16) params: Ch: Application__comm



Example of communication via Kafka: BRNBW Channel

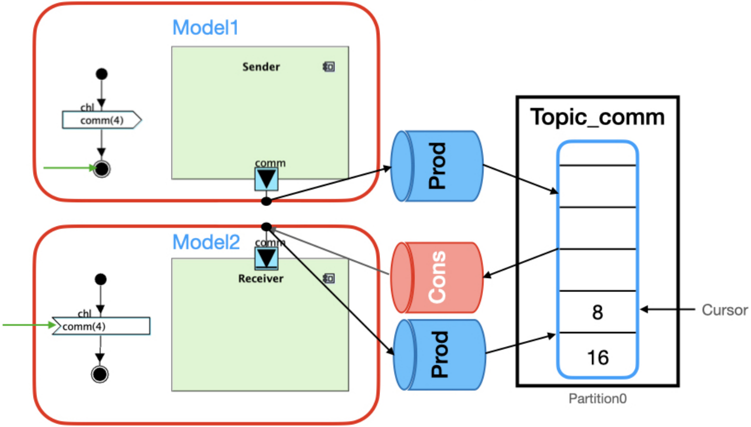
Application__Sender: Write 16,Application__comm t:0 l:4 (vl:16) params: Ch: Application__comm



wic comm 16

Example of communication via Kafka: BRNBW Channel

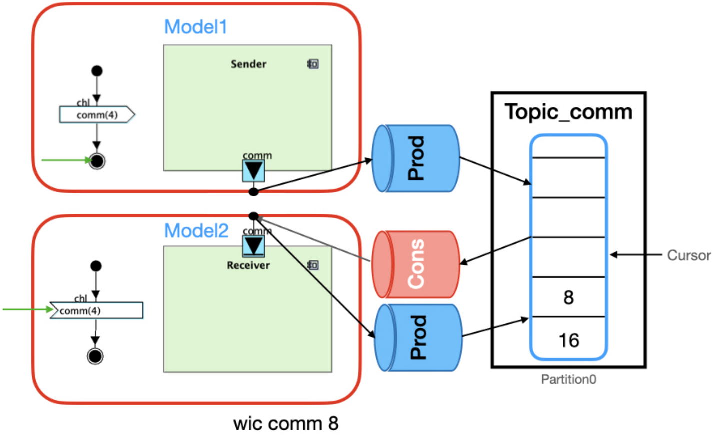
Application__Sender: Write 16,Application__comm t:0 l:4 (vl:16) params: Ch: Application__comm



Application__Receiver: Read 16,Application__comm t:1 l:2 (vl:8) params: Ch: Application__comm

Example of communication via Kafka: BRNBW Channel

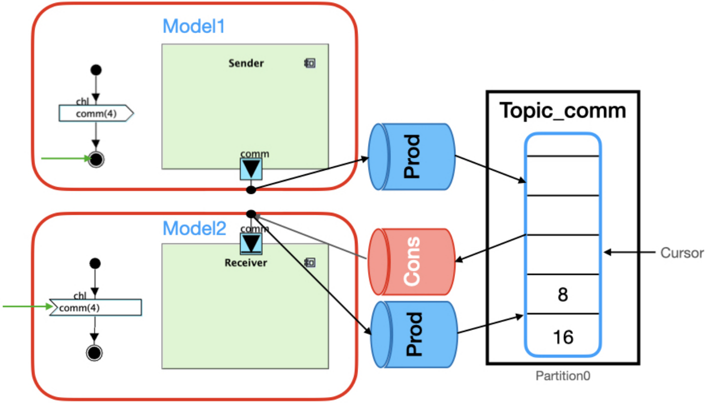
Application__Sender: Write 16,Application__comm t:0 l:4 (vl:16) params: Ch: Application__comm



Application__Receiver: Read 16,Application__comm t:1 l:2 (vl:8) params: Ch: Application__comm

Example of communication via Kafka: BRNBW Channel

Application__Sender: Write 16,Application__comm t:0 l:4 (vl:16) params: Ch: Application__comm

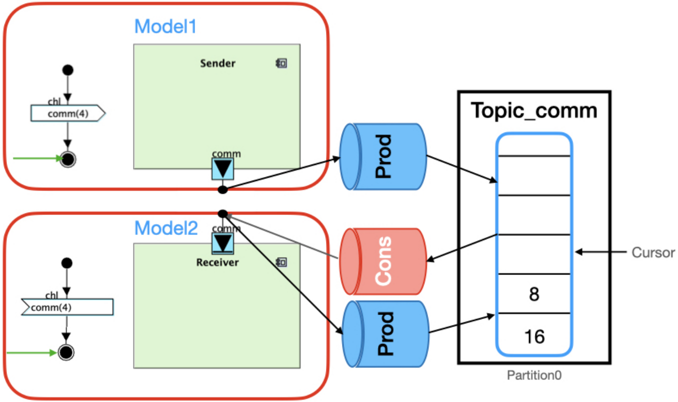


Application__Receiver: Read 16,Application__comm t:1 l:2 (vl:8) params: Ch: Application__comm

Application__Receiver: Read 16,Application__comm t:2 l:2 (vl:8) params: Ch: Application__comm

Example of communication via Kafka: BRNBW Channel

Application__Sender: Write 16,Application__comm t:0 l:4 (vl:16) params: Ch: Application__comm



Application__Receiver: Read 16,Application__comm t:1 l:2 (vl:8) params: Ch: Application__comm

Application__Receiver: Read 16,Application__comm t:2 l:2 (vl:8) params: Ch: Application__comm