Programming With Time

Patricia Derler, National Instruments

WSTS, April 2017

State of the Art

Challenges

- Safety-critical real-time systems
- Cyber-Physical systems control physical processes in tight feedback loops
- Internet of Things need to reason across many distributed nodes
- ... and many more

State of the Art

Challenges

- Safety-critical real-time systems
- Cyber-Physical systems control physical processes in tight feedback loops
- Internet of Things need to reason across many distributed nodes
- ... and many more

Progress in

- Clock synchronization protocols, e.g. Network Time Protocol (NTP), Precision Time Protocol (PTP), ...
- Time aware/sensitive networks, TSN

State of the Art

Challenges

- Safety-critical real-time systems
- Cyber-Physical systems control physical processes in tight feedback loops
- Internet of Things need to reason across many distributed nodes
- ... and many more

Progress in

- Clock synchronization protocols, e.g. Network Time Protocol (NTP), Precision Time Protocol (PTP), ...
- Time aware/sensitive networks, TSN

Now is the time to focus on how to use time in the design and development of CPS, IoT, ...

NATIONAL INSTRUMENTS

Cyber-Physical Systems

Multiple computers, comprising of sensors and actuators, connected on a network that act and react on events to meet timing constraints.

Printing Press, Bosch-Rexroth

Source: http://offsetpressman.blogspot.com/2011/03/how-flying-paster-works.html

Large Hadron Collider, Cern

Power generation and distribution

Mars Rover

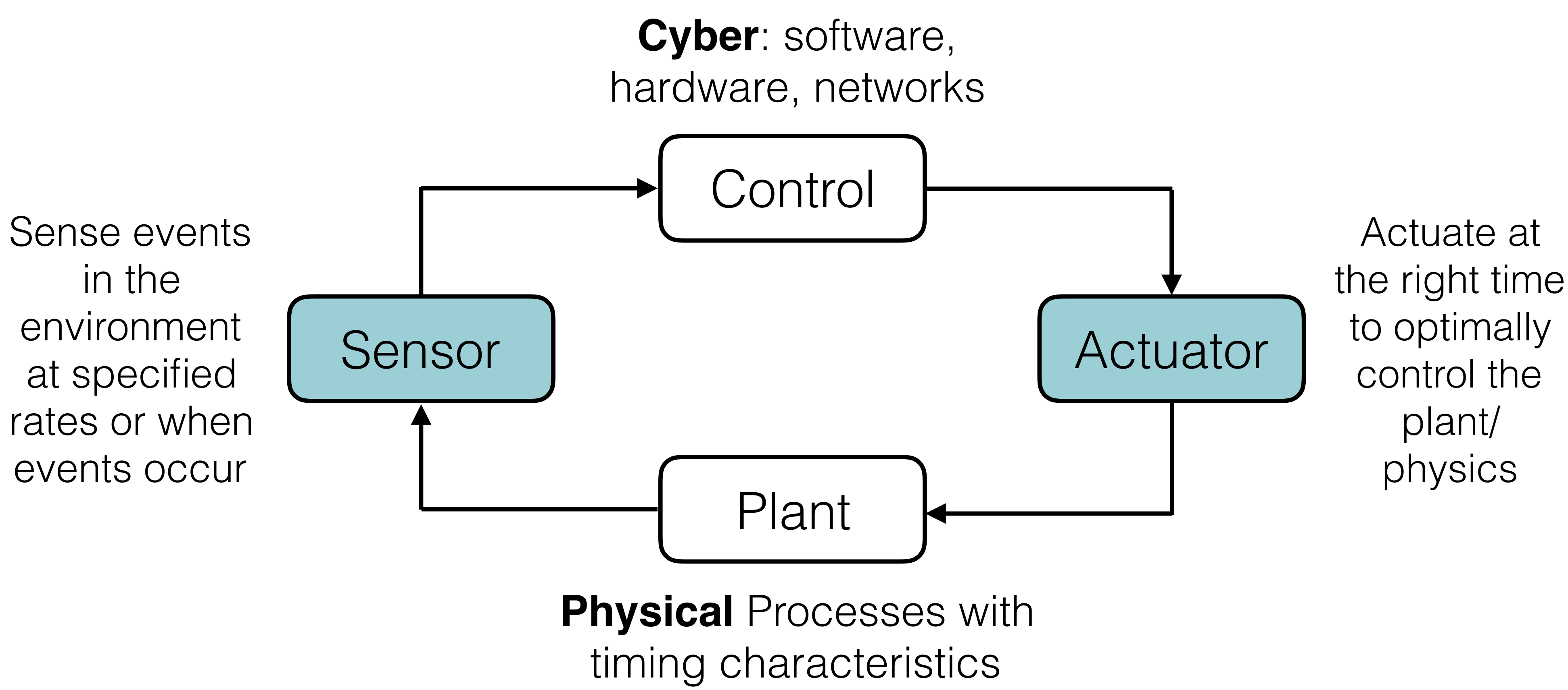
Automotive Industry

- Medical devices and systems
- Traffic control
- Automotive systems
- Process control
- Energy conservation
- Environmental control instrumentation
- Critical infrastructure control (electric power, water resources)
- Communications systems
- (Military) Defense systems
- Manufacturing
- Avionics
- Building Automation

NATIONAL INSTRUMENTS

Cyber-Physical Systems

Multiple computers, comprising of sensors and actuators, connected on a network that act and react on events to meet timing constraints.



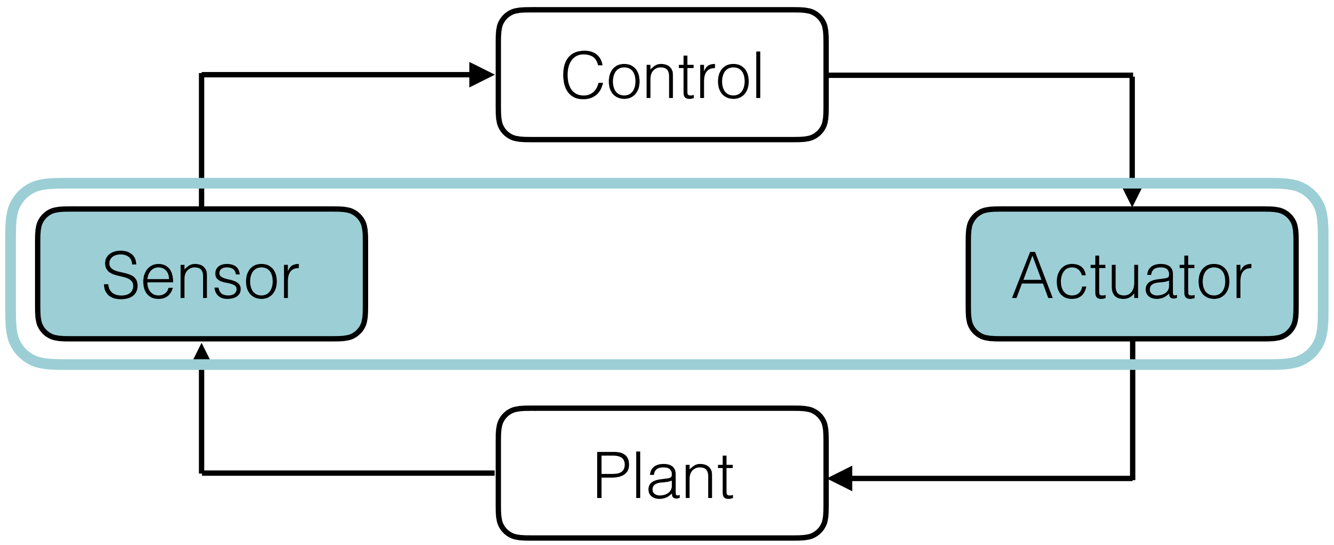
NATIONAL INSTRUMENTS

Cyber-Physical Systems

Multiple computers, comprising of sensors and actuators, connected on a network that act and react on events to meet timing constraints.

Cyber: software, hardware, networks

Sense events in the environment at specified rates or when events occur



Actuate at the right time to optimally control the plant/ physics

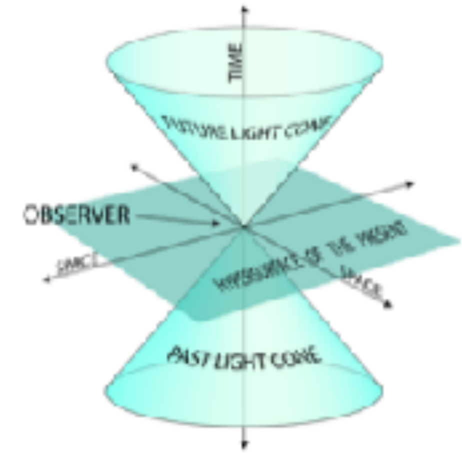
Physical Processes with timing characteristics

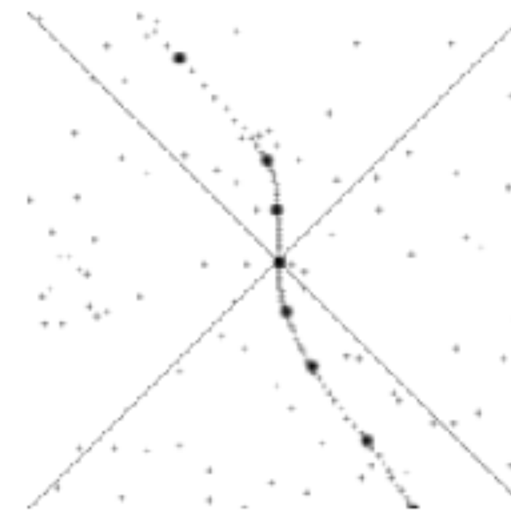
Timing Requirements specify when the cyber needs to interact with the physical

- Latency
- Simultaneity
- Chronological
- Frequency
- Phase
- Sporadic
- Burst

Challenges in Programming with Time

- Time representation
- Precision
- Phase alignment
- Jitter
- Hardware clock
- Distributed systems
- Clock edge, clock domain, clock rate
- Multiple timescales, relation to global/TAI time
- Clock synchronization
- Execution time, WCET
- Response time, WCRT
- Communication time
- Timing tolerances





Challenges in Programming with Time

- Time representation
- Precision
- Phase alignment
- Jitter
- Hardware clock
- Distributed systems
- Clock edge, clock domain, clock rate
- Multiple timescales, relation to global/TAI time
- Clock synchronization
- Execution time, WCET
- Response time, WCRT
- Communication time
- Timing tolerances

All these concerns make programming with time difficult. We need the right abstractions.

Time in the Software Lifecycle

Requirements definition

- Specify **timing** requirements, capture them in natural language/spreadsheets
- e.g. It should take exactly 100ms between sensing x and actuating y, with an acceptable tolerance of 2ms

Design

- Model the system with **timing** requirements in mind

Implementation

- Implement the system with **timing** requirements in mind

Testing

- Does the implementation satisfy the **timing** requirements?

Traditional Development

Design:
Functional model

Platform independent,
no timing information

↓

Implementation:
Software implemented on specific hardware, tweaked and tuned to achieve correct timing behavior

Platform dependent,
timing depends on hardware:
execution time, communication time, scheduling overhead, network latency, jitter

Traditional Development

Design:
Functional model

Platform independent,
no timing information

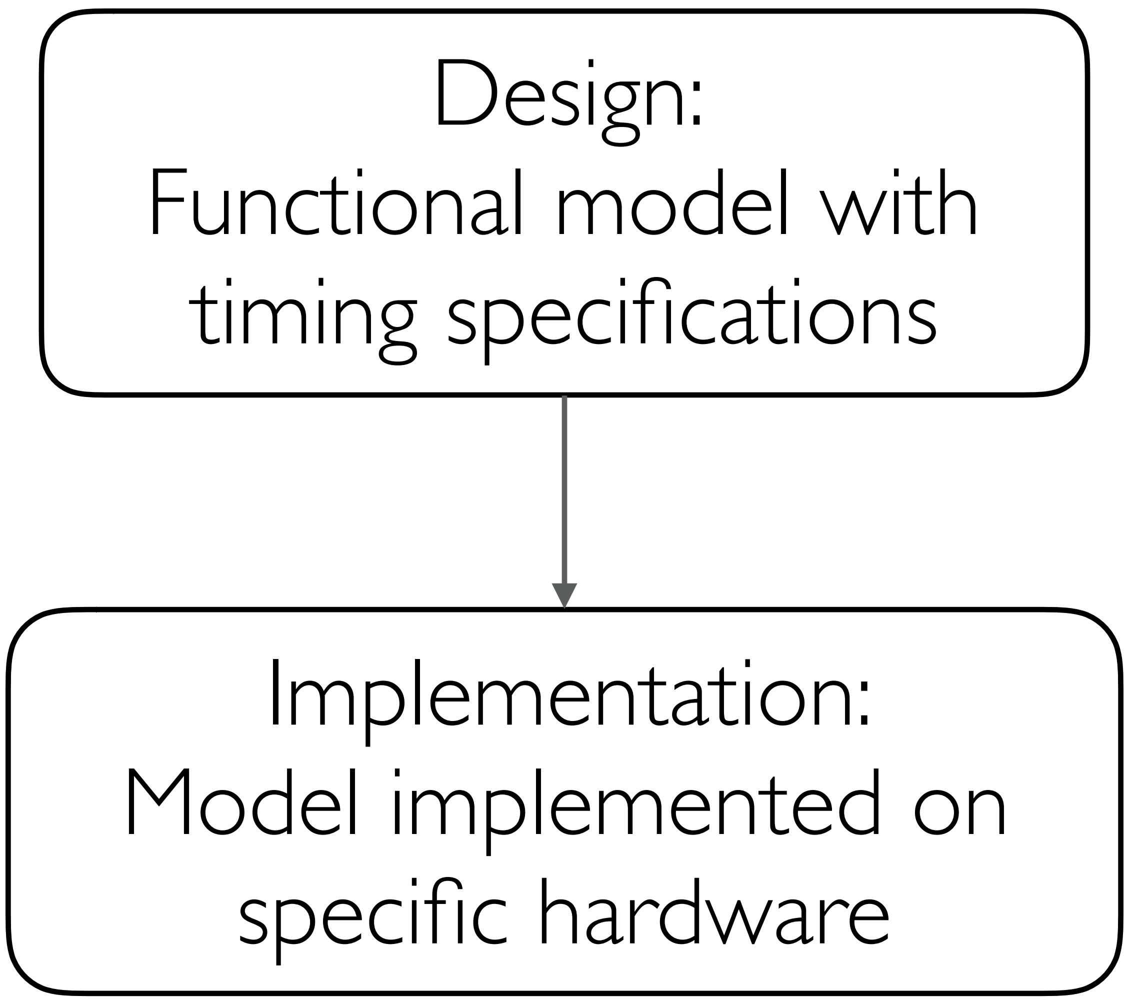
↓

Implementation:
Software implemented on specific hardware, tweaked and tuned to achieve correct timing behavior

Platform dependent,
timing depends on hardware:
execution time, communication time, scheduling overhead, network latency, jitter

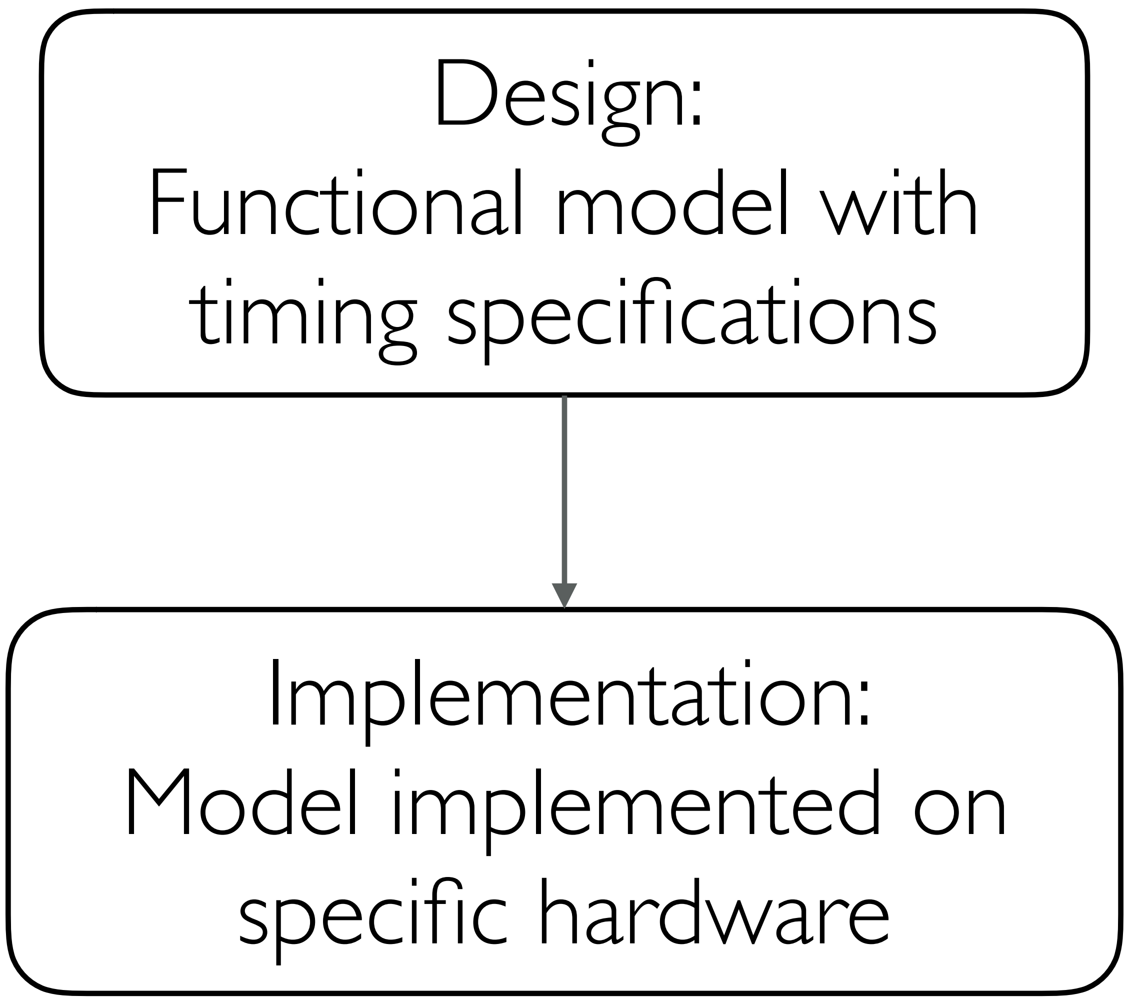
Brittle Designs

New Paradigm



Platform independent functional and timing application requirements

New Paradigm



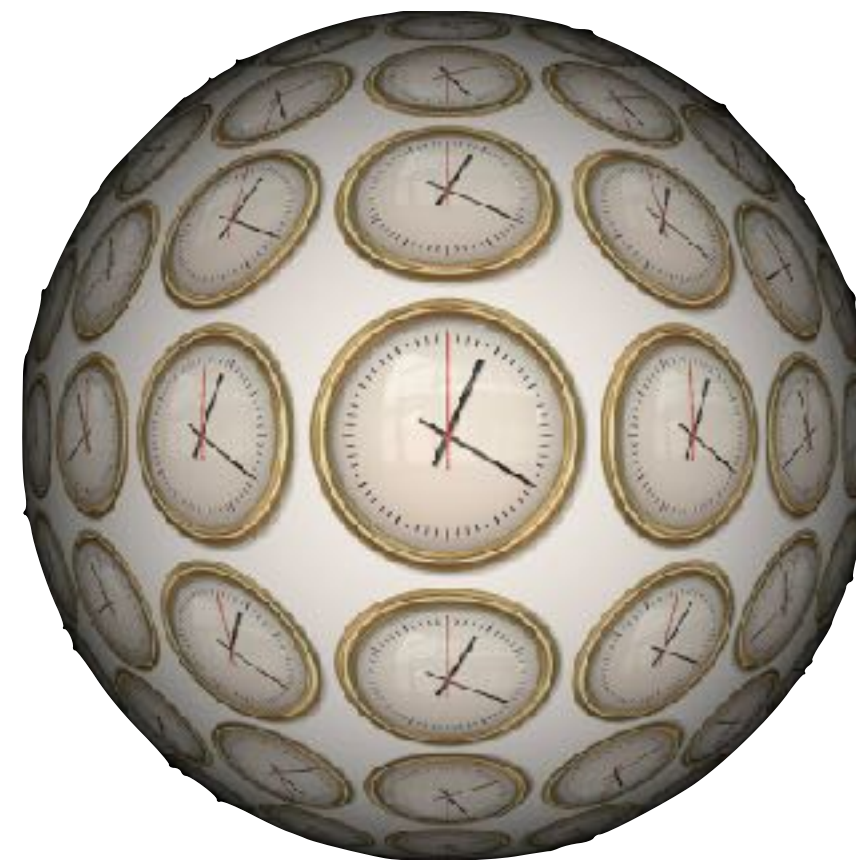
Platform independent functional and timing application requirements

A correct implementation must satisfy both, the functional and the timing specifications

Enabling a New Paradigm

Correct-by-Construction Design

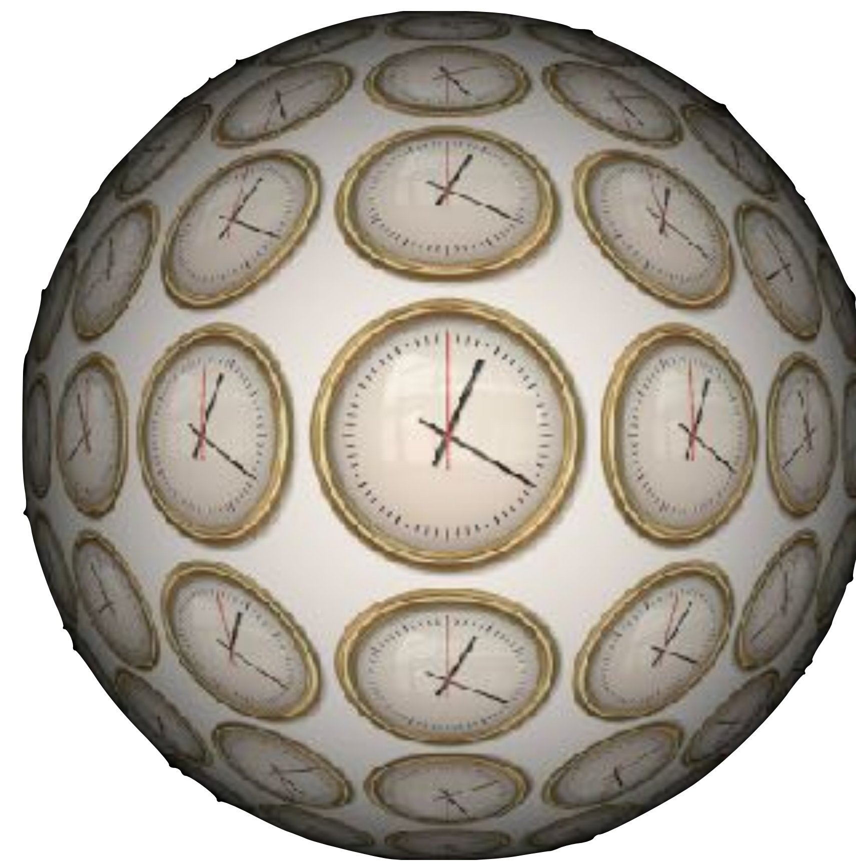
- Model system requirements in an abstract, **mathematical model**
- **Analyze** the model for correctness
- Verified tool chain to **generate** the implementation (automatically)



Enabling a New Paradigm

Correct-by-Construction Design

- Model system requirements in an abstract, **mathematical model**
- **Analyze** the model for correctness
- Verified tool chain to **generate** the implementation (automatically)



Global notion of time

- At design time, assume a global notion of time
- Abstract away details of imperfect clocks
- Made possible by modern clock synchronization techniques

ABSTRACTIONS FOR PROGRAMMING WITH TIME

Capturing Timing Requirements

Traditionally: Natural Language

- In form of text documents or spreadsheets
- Ambiguous, cannot be interpreted by computer

Capturing Timing Requirements

Traditionally: Natural Language

- In form of text documents or spreadsheets
- Ambiguous, cannot be interpreted by computer

Formal, mathematical unambiguous description

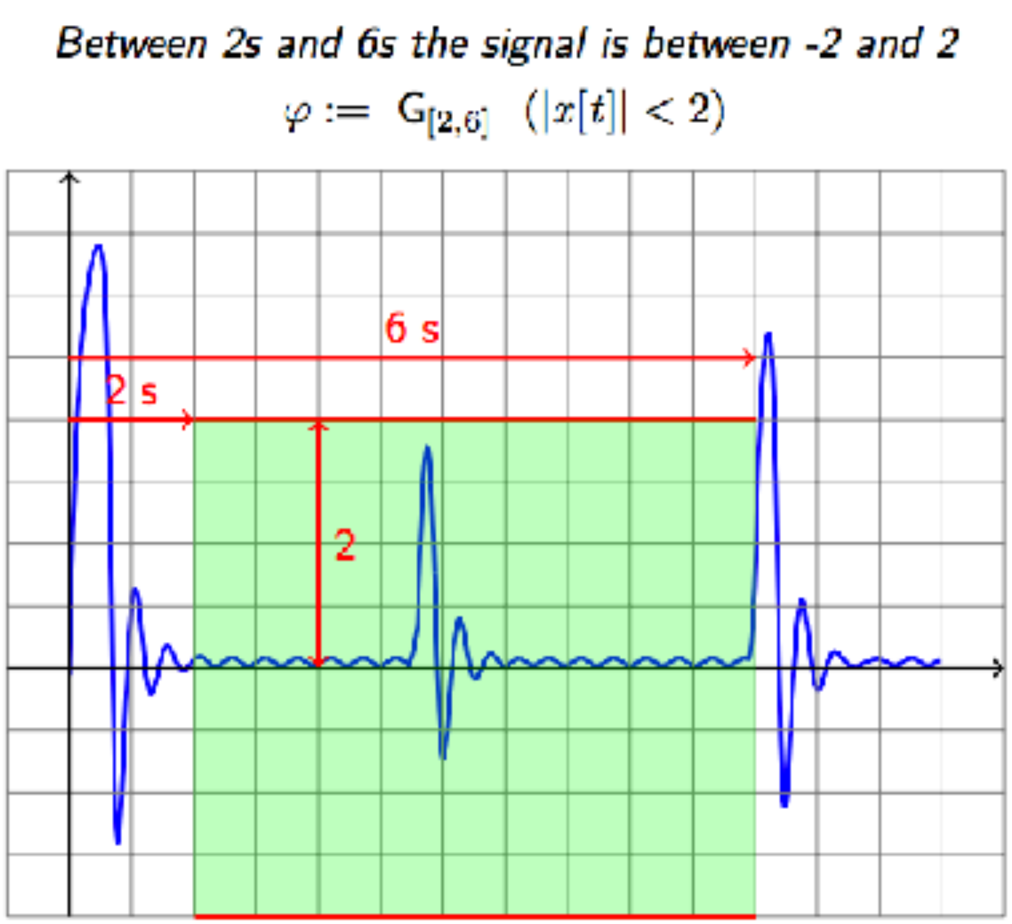
- **Temporal logic** to formally specify patterns that timed behaviors of systems should (not) satisfy
- LTL, CTL, TCTL, MTL, TILCO-X, STL, ...
- Signal Temporal Logic (STL)[1]: properties related to the order of discrete events and the temporal distance between them

[1]Alexandre Donzé, On Signal Temporal Logic, UC Berkeley, Lecture EECS294-98 Spring, 2014

Capturing Timing Requirements

Traditionally: Natural Language

- In form of text documents or spreadsheets
- Ambiguous, cannot be interpreted by computer



Formal, mathematical unambiguous description

- **Temporal logic** to formally specify patterns that timed behaviors of systems should (not) satisfy
- LTL, CTL, TCTL, MTL, TILCO-X, STL, ...
- Signal Temporal Logic (STL)[1]: properties related to the order of discrete events and the temporal distance between them

[1]Alexandre Donzé, On Signal Temporal Logic, UC Berkeley, Lecture EECS294-98 Spring, 2014

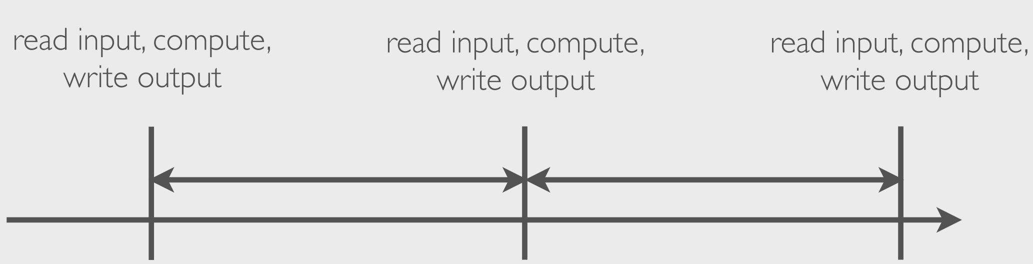
Timing as Part of the Model

- Software is split into **tasks**
- Timing of operations on the task is defined:
  - When are inputs to the task read?
  - When are outputs from the task written?

Timing as Part of the Model

- Software is split into **tasks**
- Timing of operations on the task is defined:
  - When are inputs to the task read?
  - When are outputs from the task written?

e.g. Zero Execution Time abstraction



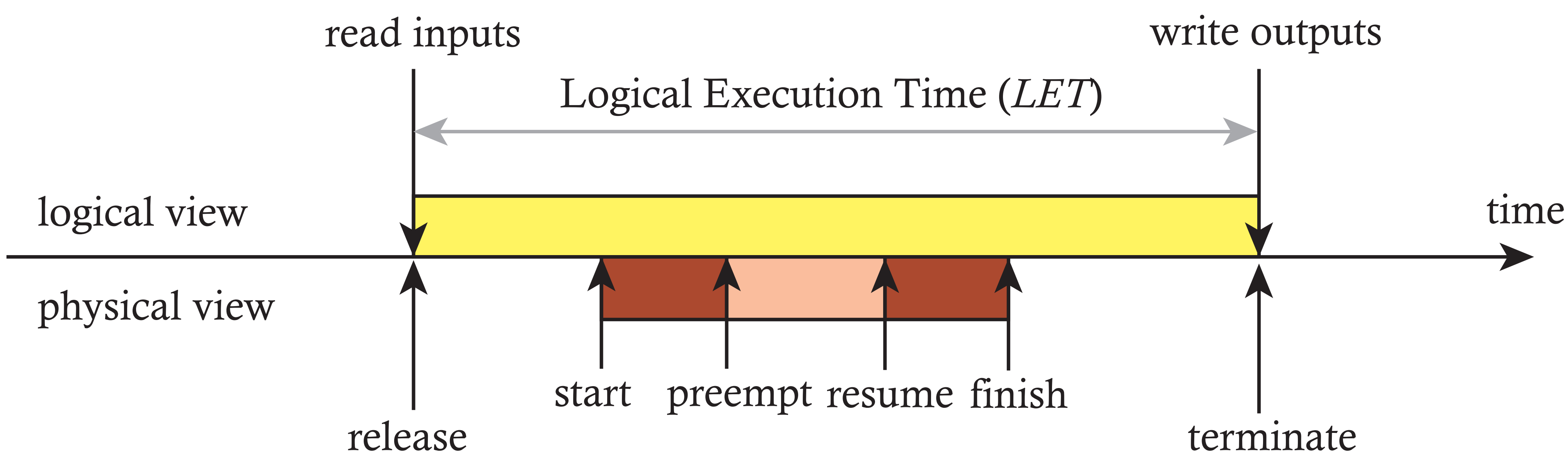
Key abstraction in Synchronous Programming[1, 2, 3]

[1]F. Boussinot and R. De Simone. The ESTEREL language. Proceedings of the IEEE, 79(9), 1991.

[2]N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous data flow programming language LUSTRE. Proceedings of the IEEE, 79(9), 1991.

[3]P. Le Guernic, T. Gauthier, M. Le Borgne, and C. Le Maire. Programming real-time applications with SIGNAL. Proceedings of the IEEE, 79(9), 1991.

Logical Execution Time



Implemented in Giotto[1] and similar models of computation (HTL[2], TDL[3], FTOS[4], …)

[1]Henzinger, Thomas A., Benjamin Horowitz, and Christoph Meyer Kirsch. "Giotto: A time-triggered language for embedded programming." International Workshop on Embedded Software. Springer Berlin Heidelberg, 2001.

[2]T. A. Henzinger, C. M. Kirsch, E. R. B. Marques and A. Sokolova, "Distributed, Modular HTL," 2009 30th IEEE Real-Time Systems Symposium, Washington, DC, 2009, pp. 171-180.

[3]A. Naderlinger, J. Pletzer, W. Pree and J. Templ, "Model-Driven Development of FlexRay-Based Systems with the Timing Definition Language (TDL)," Software Engineering for Automotive Systems, 2007. ICSE Workshops SEAS '07. Fourth International Workshop on, Minneapolis, MN, 2007, pp. 6-6.

[4]C. Buckl, D. Sojer and A. Knoll, "FTOS: Model-driven development of fault-tolerant automation systems," 2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010), Bilbao, 2010, pp. 1-8.

Modeling with LETs

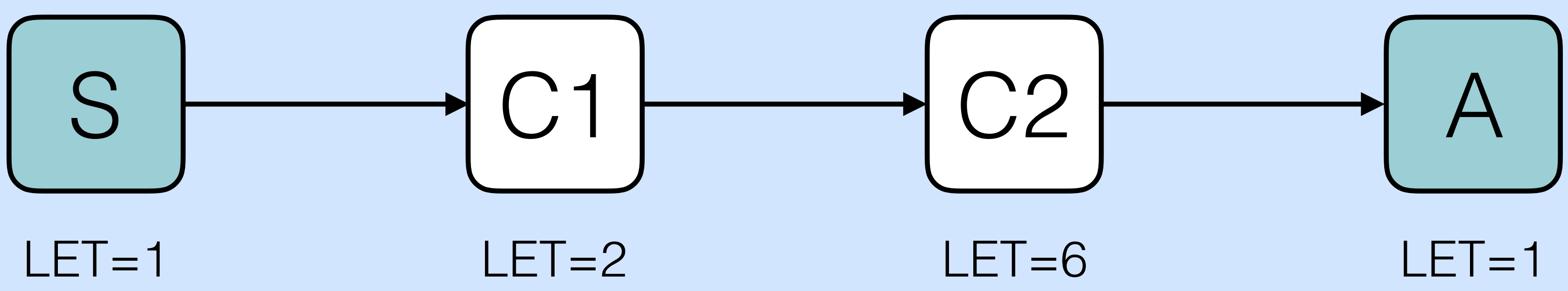
Specify logical execution time for every task in a way that satisfies the timing requirement

Example: Latency between sensing input and actuation should be 10ms

Modeling with LETs

Specify logical execution time for every task in a way that satisfies the timing requirement

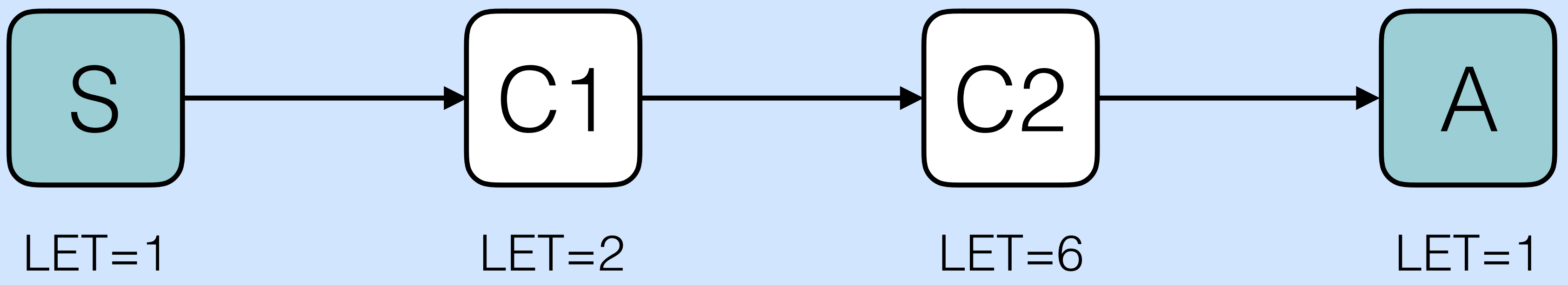
Example: Latency between sensing input and actuation should be 10ms



Modeling with LETs

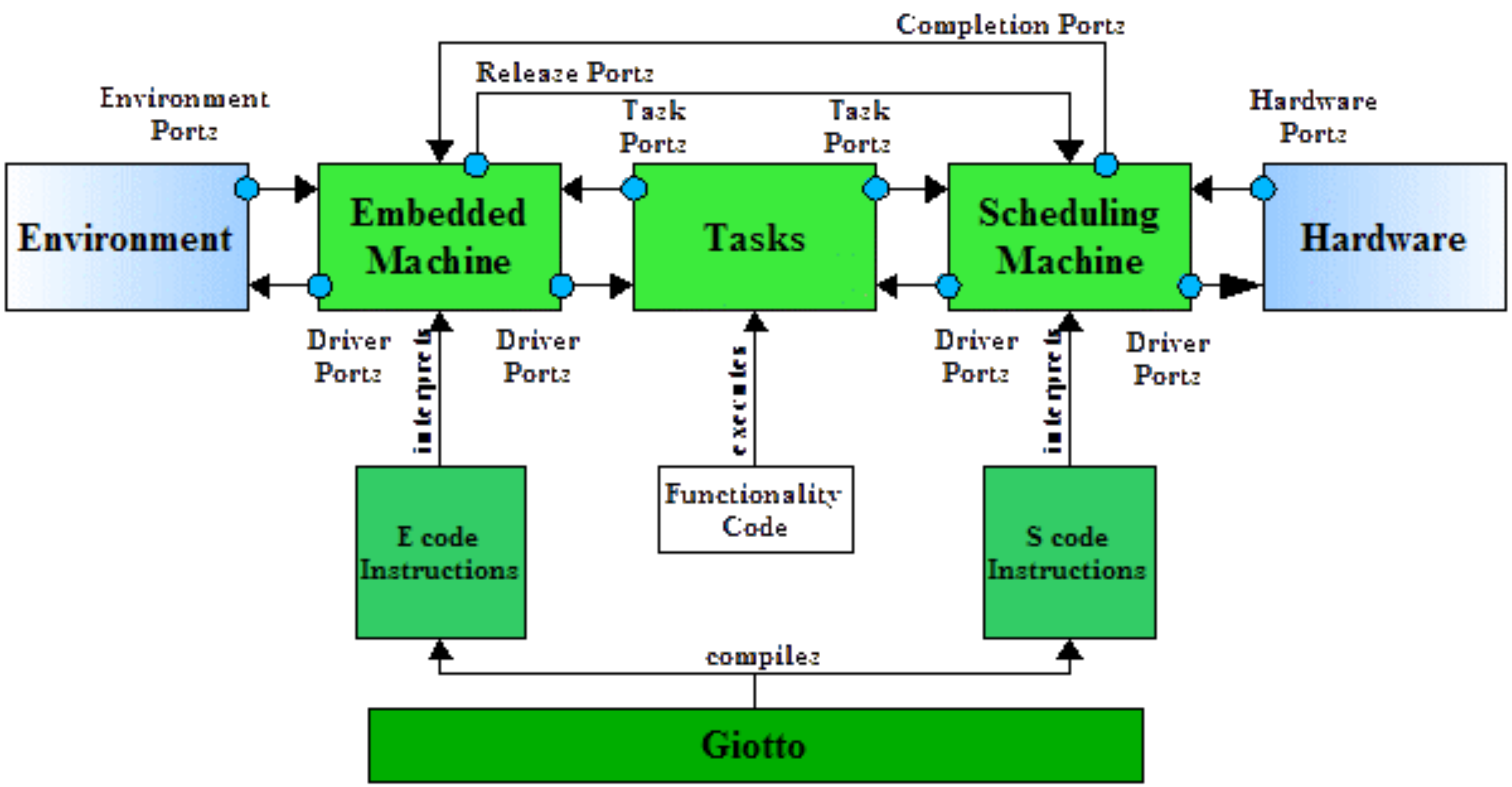
Specify logical execution time for every task in a way that satisfies the timing requirement

Example: Latency between sensing input and actuation should be 10ms



Sum of all LETs from S to A = 10

GIOTTO : Modeling with LETs



[1]Henzinger, Thomas A., Benjamin Horowitz, and Christoph Meyer Kirsch. "Giotto: A time-triggered language for embedded programming." International Workshop on Embedded Software. Springer Berlin Heidelberg, 2001.

Ptides

Programming Temporally Integrated Distributed Embedded Systems

- Extends discrete event model of computation with logical time, and physical time
- Relates physical time and logical time only where necessary - at IO side effects
- Timing of inputs from environment is event-triggered, timing of outputs to environment is well defined with respect to input timing
- Explicit delay nodes describe IO latency

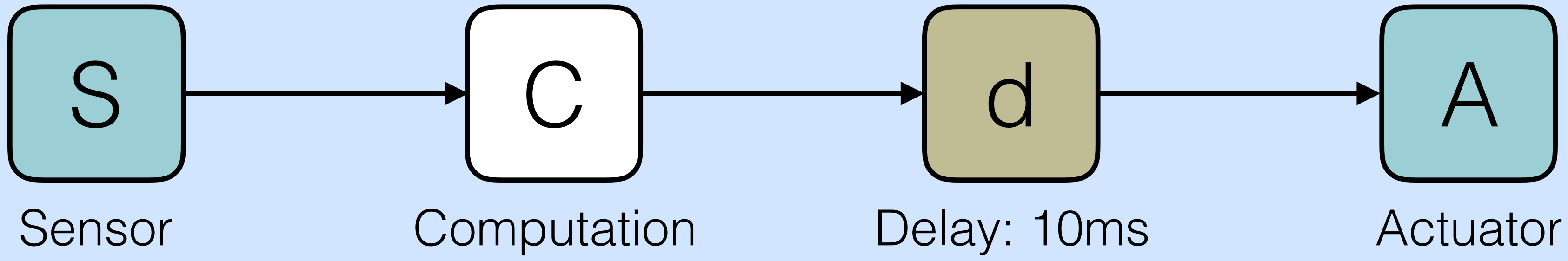
Yang Zhao, Jie Liu and Edward A. Lee. A Programming Model for Time-Synchronized Distributed Real-Time Systems. In Proceedings of the IEEE Real Time and Embedded Technology and Applications Symposium (RTAS), 2007.
Patricia Derler, Edward A. Lee, Michael Zimmer, Logically Synchronous Models of Distributed Systems with Explicit Timing Specifications, In Proceedings of 20th IMEKO TC4 International Symposium and 18th International Workshop on ADC Modelling and Testing Research on Electric and Electronic Measurement for the Economic Upturn, Benevento, Italy, September 15-17, 2014.

Ptides

Programming Temporally Integrated Distributed Embedded Systems

- Extends discrete event model of computation with logical time, and physical time
- Relates physical time and logical time only where necessary - at IO side effects
- Timing of inputs from environment is event-triggered, timing of outputs to environment is well defined with respect to input timing
- Explicit delay nodes describe IO latency

Example: Latency between sensing input and actuation should be 10ms



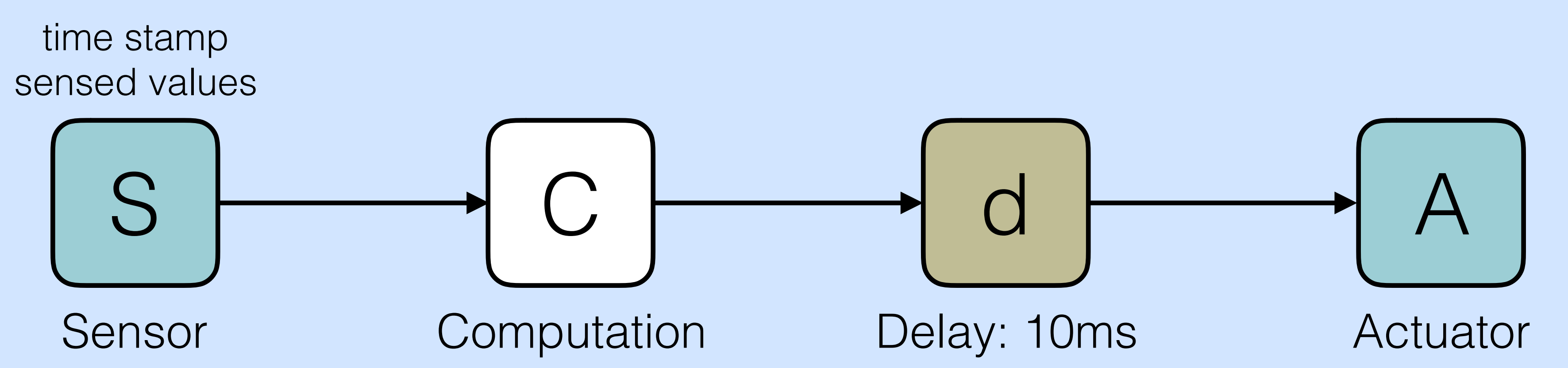
Yang Zhao, Jie Liu and Edward A. Lee. A Programming Model for Time-Synchronized Distributed Real-Time Systems. In Proceedings of the IEEE Real Time and Embedded Technology and Applications Symposium (RTAS), 2007.
Patricia Derler, Edward A. Lee, Michael Zimmer, Logically Synchronous Models of Distributed Systems with Explicit Timing Specifications, In Proceedings of 20th IMEKO TC4 International Symposium and 18th International Workshop on ADC Modelling and Testing Research on Electric and Electronic Measurement for the Economic Upturn, Benevento, Italy, September 15-17, 2014.

Ptides

Programming Temporally Integrated Distributed Embedded Systems

- Extends discrete event model of computation with logical time, and physical time
- Relates physical time and logical time only where necessary - at IO side effects
- Timing of inputs from environment is event-triggered, timing of outputs to environment is well defined with respect to input timing
- Explicit delay nodes describe IO latency

Example: Latency between sensing input and actuation should be 10ms



Yang Zhao, Jie Liu and Edward A. Lee. A Programming Model for Time-Synchronized Distributed Real-Time Systems. In Proceedings of the IEEE Real Time and Embedded Technology and Applications Symposium (RTAS), 2007.

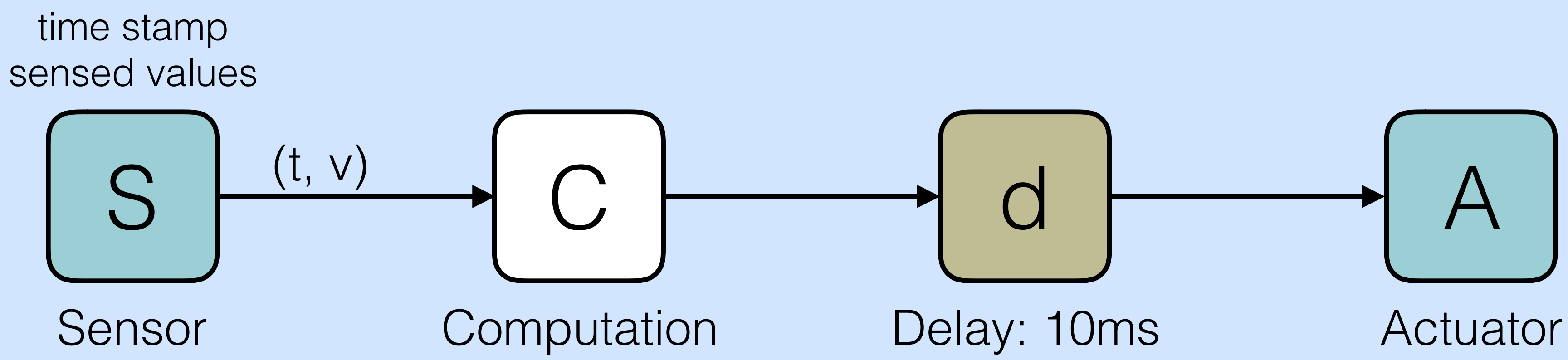
Patricia Derler, Edward A. Lee, Michael Zimmer, Logically Synchronous Models of Distributed Systems with Explicit Timing Specifications, In Proceedings of 20th IMEKO TC4 International Symposium and 18th International Workshop on ADC Modelling and Testing Research on Electric and Electronic Measurement for the Economic Upturn, Benevento, Italy, September 15-17, 2014.

Ptides

Programming Temporally Integrated Distributed Embedded Systems

- Extends discrete event model of computation with logical time, and physical time
- Relates physical time and logical time only where necessary - at IO side effects
- Timing of inputs from environment is event-triggered, timing of outputs to environment is well defined with respect to input timing
- Explicit delay nodes describe IO latency

Example: Latency between sensing input and actuation should be 10ms



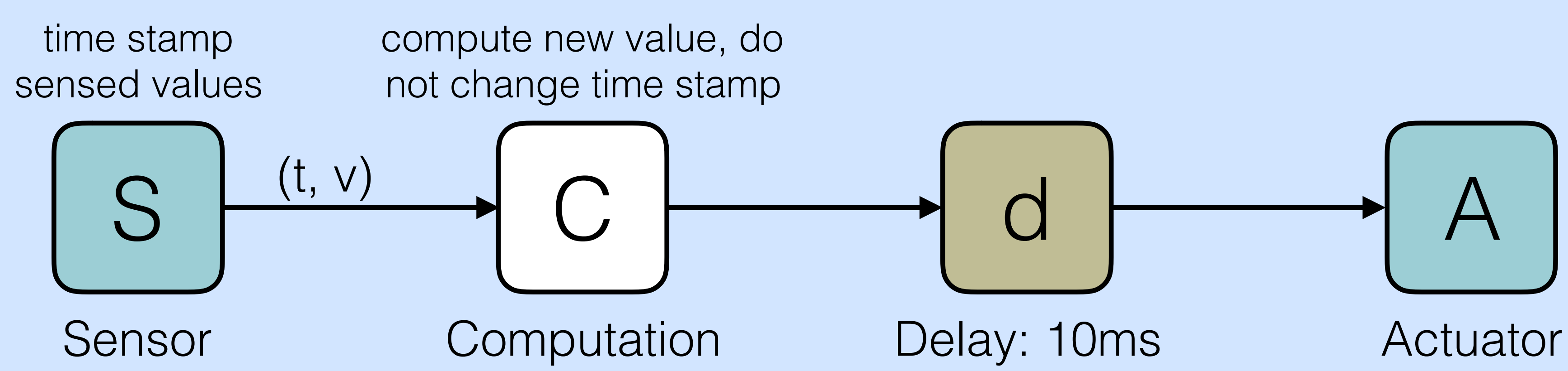
Yang Zhao, Jie Liu and Edward A. Lee. A Programming Model for Time-Synchronized Distributed Real-Time Systems. In Proceedings of the IEEE Real Time and Embedded Technology and Applications Symposium (RTAS), 2007.
Patricia Derler, Edward A. Lee, Michael Zimmer, Logically Synchronous Models of Distributed Systems with Explicit Timing Specifications, In Proceedings of 20th IMEKO TC4 International Symposium and 18th International Workshop on ADC Modelling and Testing Research on Electric and Electronic Measurement for the Economic Upturn, Benevento, Italy, September 15-17, 2014.

Ptides

Programming Temporally Integrated Distributed Embedded Systems

- Extends discrete event model of computation with logical time, and physical time
- Relates physical time and logical time only where necessary - at IO side effects
- Timing of inputs from environment is event-triggered, timing of outputs to environment is well defined with respect to input timing
- Explicit delay nodes describe IO latency

Example: Latency between sensing input and actuation should be 10ms



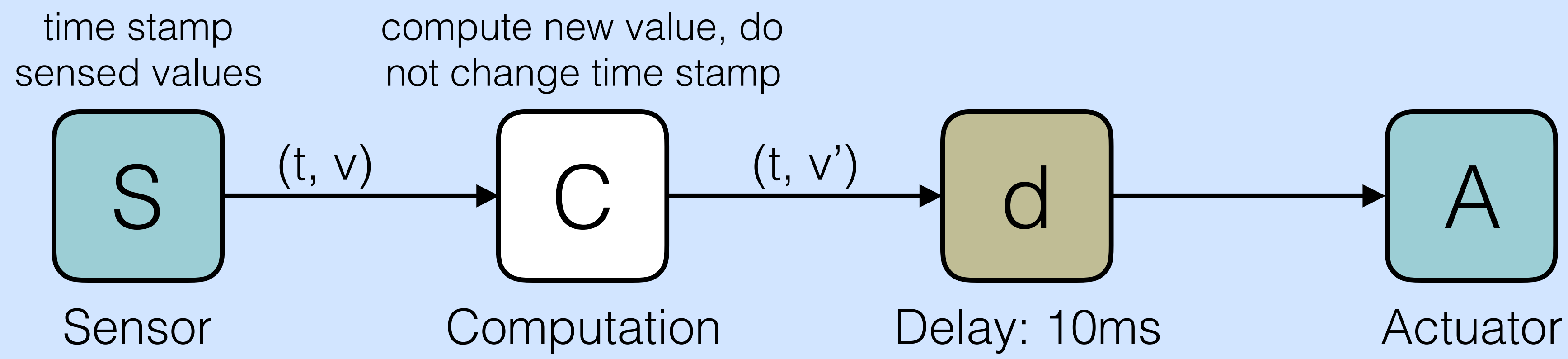
Yang Zhao, Jie Liu and Edward A. Lee. A Programming Model for Time-Synchronized Distributed Real-Time Systems. In Proceedings of the IEEE Real Time and Embedded Technology and Applications Symposium (RTAS), 2007.
Patricia Derler, Edward A. Lee, Michael Zimmer, Logically Synchronous Models of Distributed Systems with Explicit Timing Specifications, In Proceedings of 20th IMEKO TC4 International Symposium and 18th International Workshop on ADC Modelling and Testing Research on Electric and Electronic Measurement for the Economic Upturn, Benevento, Italy, September 15-17, 2014.

Ptides

Programming Temporally Integrated Distributed Embedded Systems

- Extends discrete event model of computation with logical time, and physical time
- Relates physical time and logical time only where necessary - at IO side effects
- Timing of inputs from environment is event-triggered, timing of outputs to environment is well defined with respect to input timing
- Explicit delay nodes describe IO latency

Example: Latency between sensing input and actuation should be 10ms



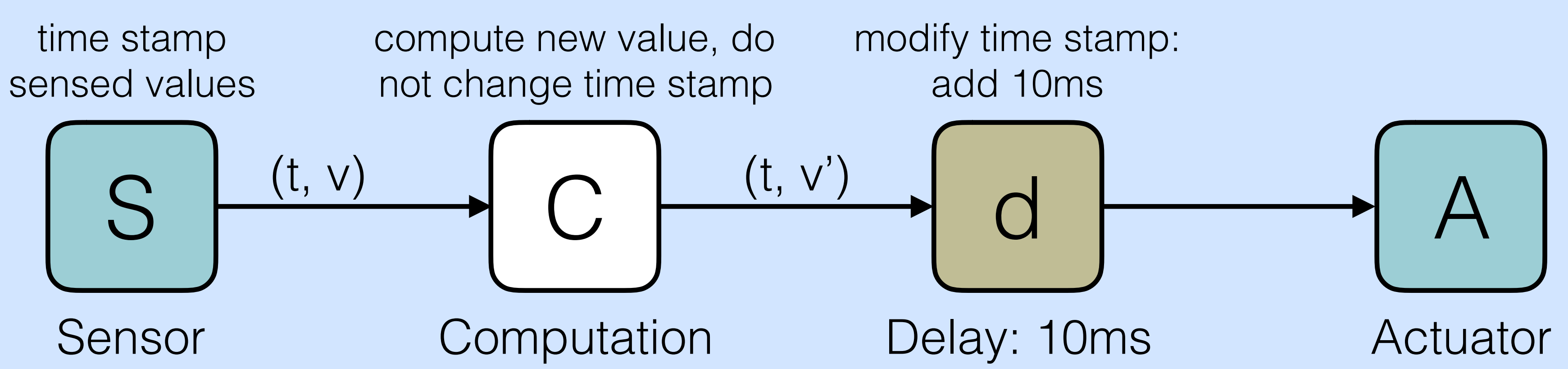
Yang Zhao, Jie Liu and Edward A. Lee. A Programming Model for Time-Synchronized Distributed Real-Time Systems. In Proceedings of the IEEE Real Time and Embedded Technology and Applications Symposium (RTAS), 2007.
Patricia Derler, Edward A. Lee, Michael Zimmer, Logically Synchronous Models of Distributed Systems with Explicit Timing Specifications, In Proceedings of 20th IMEKO TC4 International Symposium and 18th International Workshop on ADC Modelling and Testing Research on Electric and Electronic Measurement for the Economic Upturn, Benevento, Italy, September 15-17, 2014.

Ptides

Programming Temporally Integrated Distributed Embedded Systems

- Extends discrete event model of computation with logical time, and physical time
- Relates physical time and logical time only where necessary - at IO side effects
- Timing of inputs from environment is event-triggered, timing of outputs to environment is well defined with respect to input timing
- Explicit delay nodes describe IO latency

Example: Latency between sensing input and actuation should be 10ms



Yang Zhao, Jie Liu and Edward A. Lee. A Programming Model for Time-Synchronized Distributed Real-Time Systems. In Proceedings of the IEEE Real Time and Embedded Technology and Applications Symposium (RTAS), 2007.

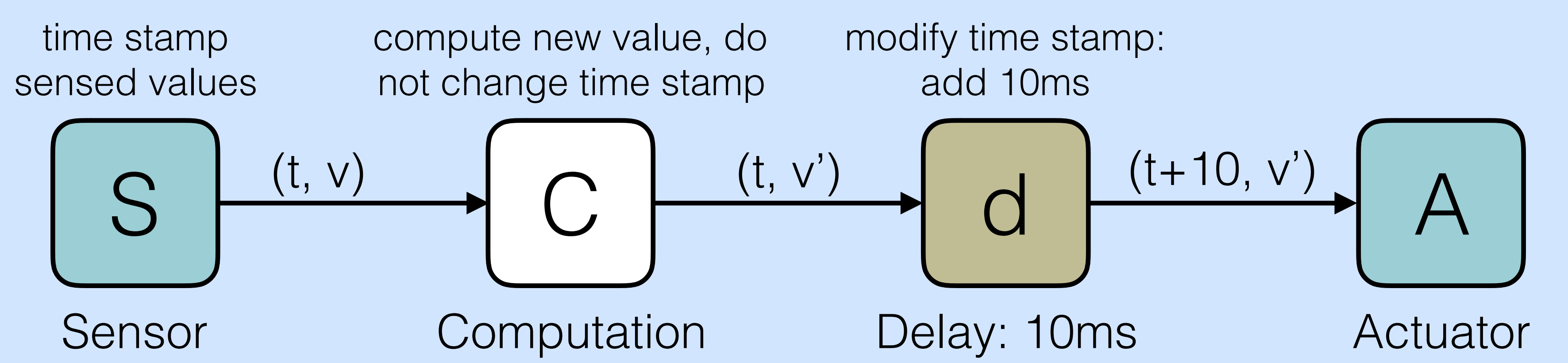
Patricia Derler, Edward A. Lee, Michael Zimmer, Logically Synchronous Models of Distributed Systems with Explicit Timing Specifications, In Proceedings of 20th IMEKO TC4 International Symposium and 18th International Workshop on ADC Modelling and Testing Research on Electric and Electronic Measurement for the Economic Upturn, Benevento, Italy, September 15-17, 2014.

Ptides

Programming Temporally Integrated Distributed Embedded Systems

- Extends discrete event model of computation with logical time, and physical time
- Relates physical time and logical time only where necessary - at IO side effects
- Timing of inputs from environment is event-triggered, timing of outputs to environment is well defined with respect to input timing
- Explicit delay nodes describe IO latency

Example: Latency between sensing input and actuation should be 10ms



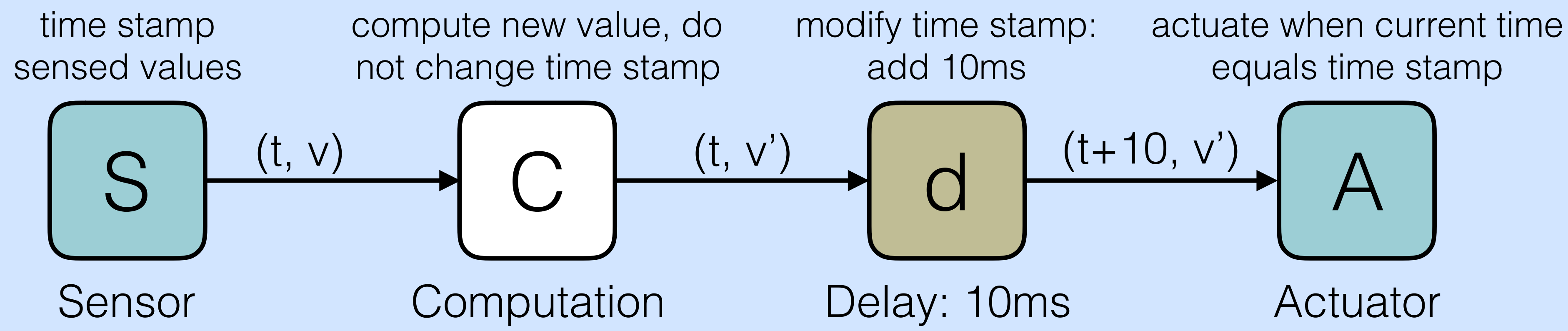
Yang Zhao, Jie Liu and Edward A. Lee. A Programming Model for Time-Synchronized Distributed Real-Time Systems. In Proceedings of the IEEE Real Time and Embedded Technology and Applications Symposium (RTAS), 2007.
Patricia Derler, Edward A. Lee, Michael Zimmer, Logically Synchronous Models of Distributed Systems with Explicit Timing Specifications, In Proceedings of 20th IMEKO TC4 International Symposium and 18th International Workshop on ADC Modelling and Testing Research on Electric and Electronic Measurement for the Economic Upturn, Benevento, Italy, September 15-17, 2014.

Ptides

Programming Temporally Integrated Distributed Embedded Systems

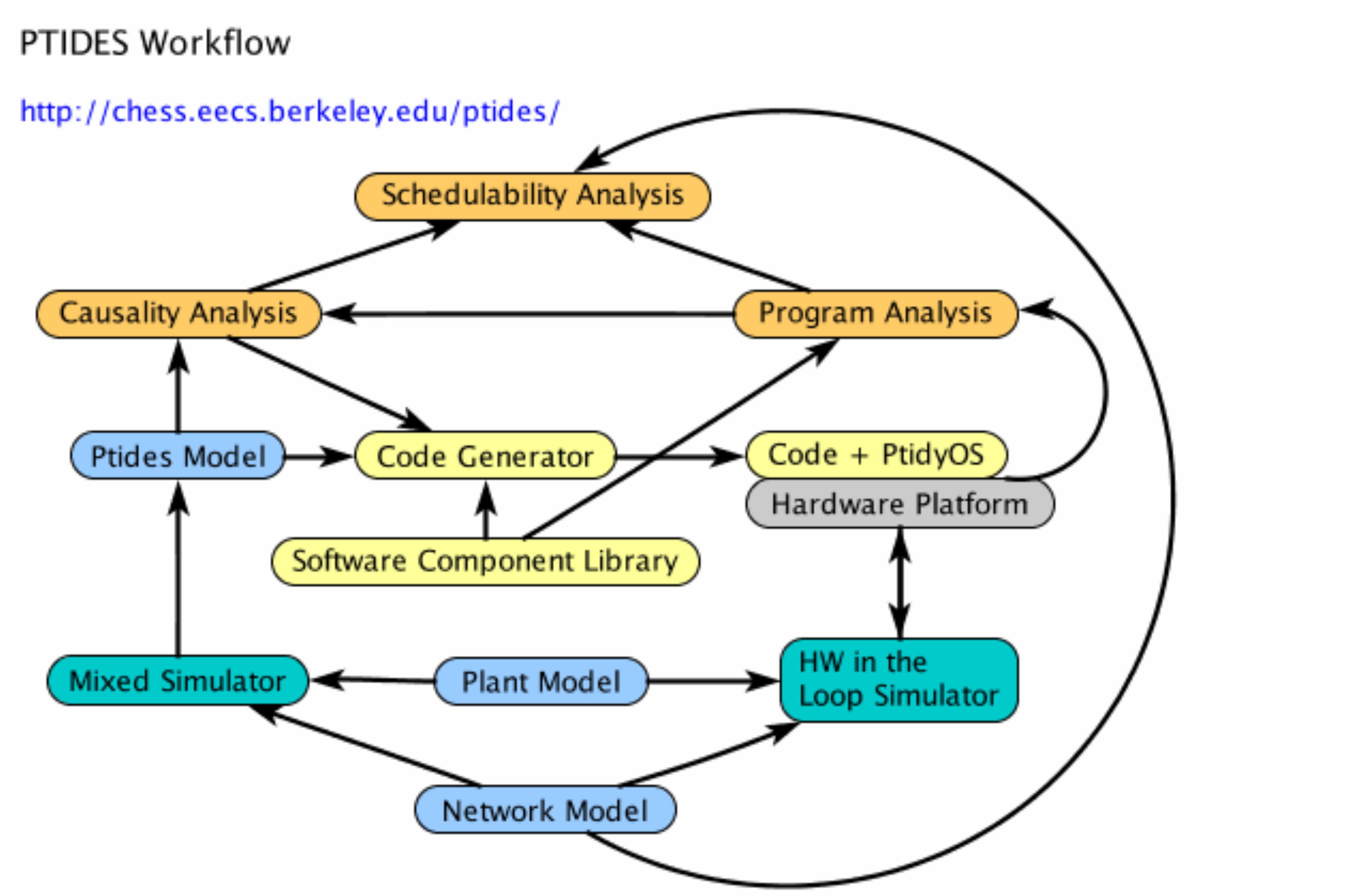
- Extends discrete event model of computation with logical time, and physical time
- Relates physical time and logical time only where necessary - at IO side effects
- Timing of inputs from environment is event-triggered, timing of outputs to environment is well defined with respect to input timing
- Explicit delay nodes describe IO latency

Example: Latency between sensing input and actuation should be 10ms



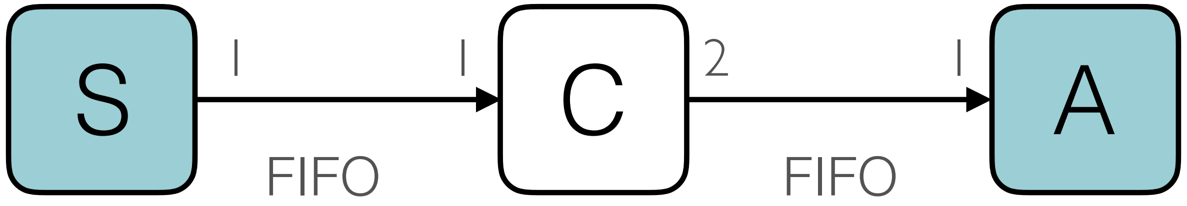
Yang Zhao, Jie Liu and Edward A. Lee. A Programming Model for Time-Synchronized Distributed Real-Time Systems. In Proceedings of the IEEE Real Time and Embedded Technology and Applications Symposium (RTAS), 2007.
Patricia Derler, Edward A. Lee, Michael Zimmer, Logically Synchronous Models of Distributed Systems with Explicit Timing Specifications, In Proceedings of 20th IMEKO TC4 International Symposium and 18th International Workshop on ADC Modelling and Testing Research on Electric and Electronic Measurement for the Economic Upturn, Benevento, Italy, September 15-17, 2014.

Ptides Workflow



Dataflow with Timing

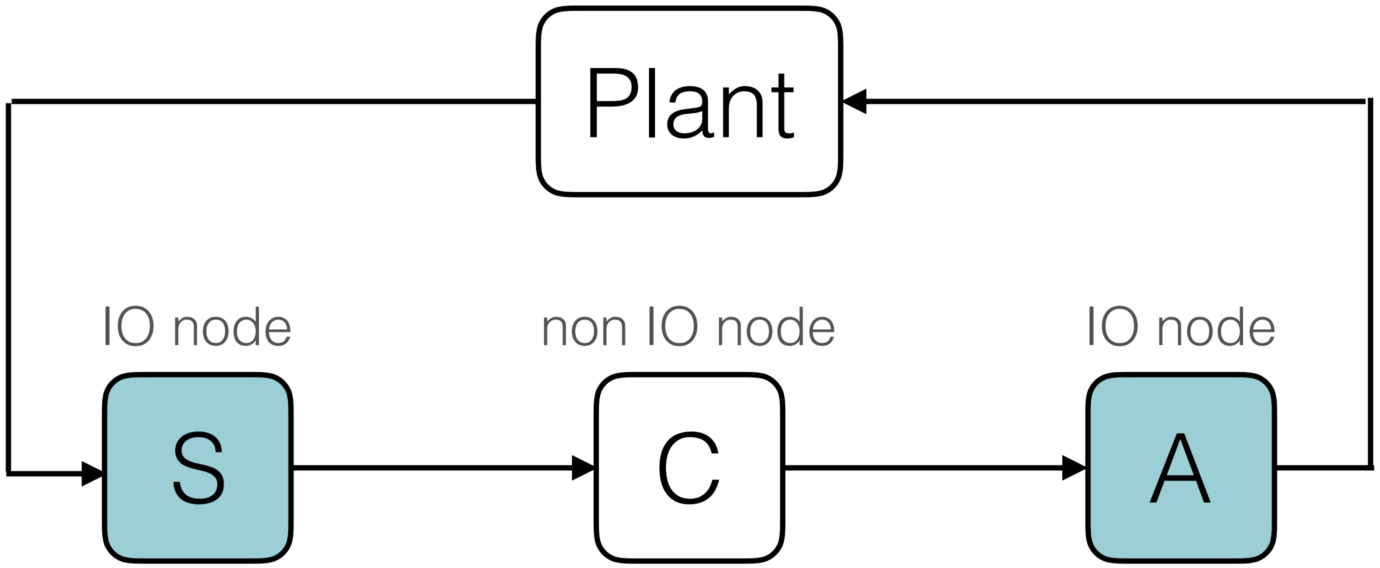
Timing Specifications on IO nodes in Synchronous Dataflow (SDF)[1]



Synchronous Dataflow (SDF):
nodes consume and produce fixed amount of tokens, communicate via FIFO channels, can have initial tokens/ delays on channels

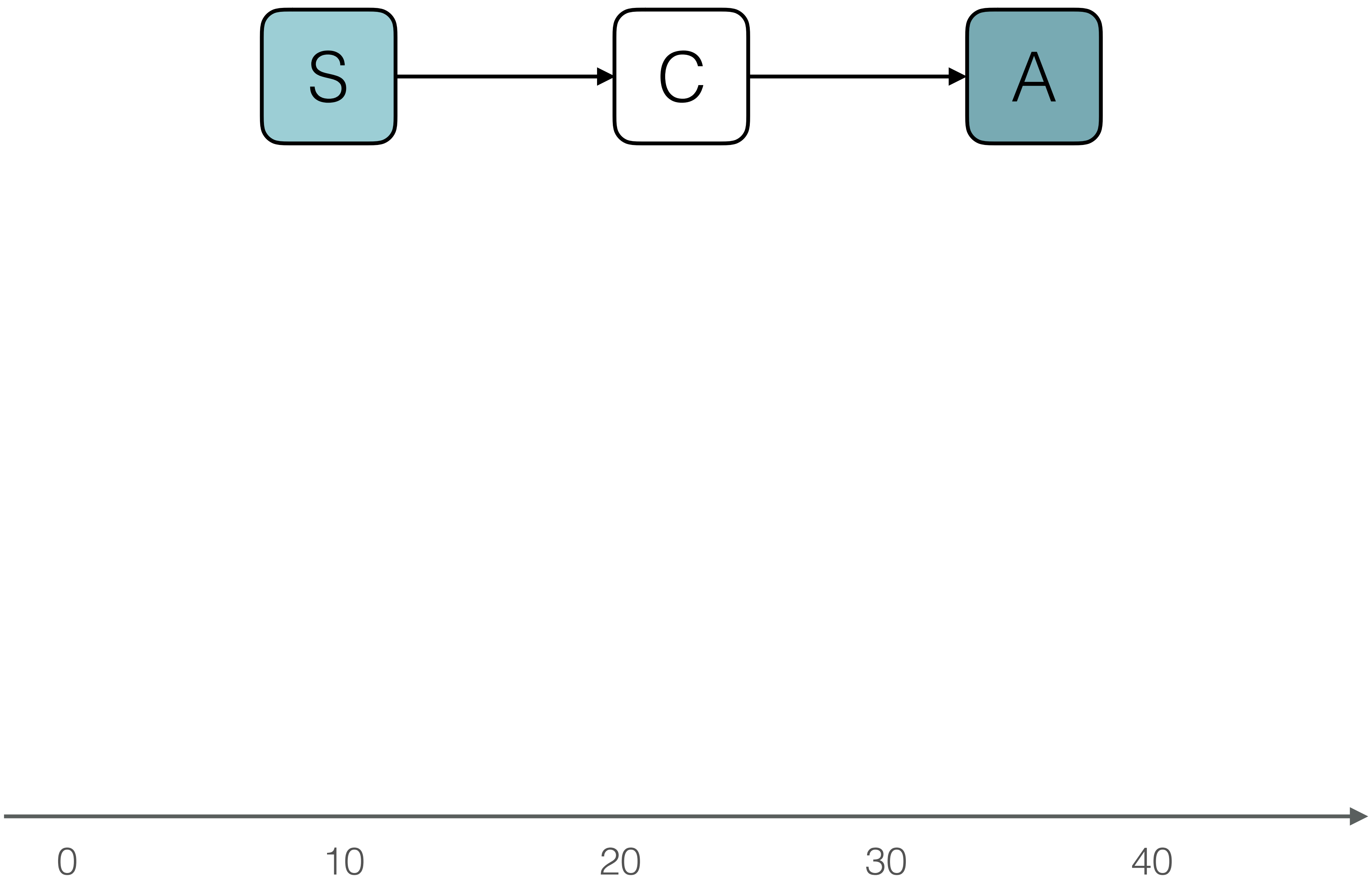
[1]Patricia Derler, Kaushik Ravindran, and Rhishikesh Limaye, Specification of Precise Timing in Dataflow Models, Memocode 2016

SDF with Timing

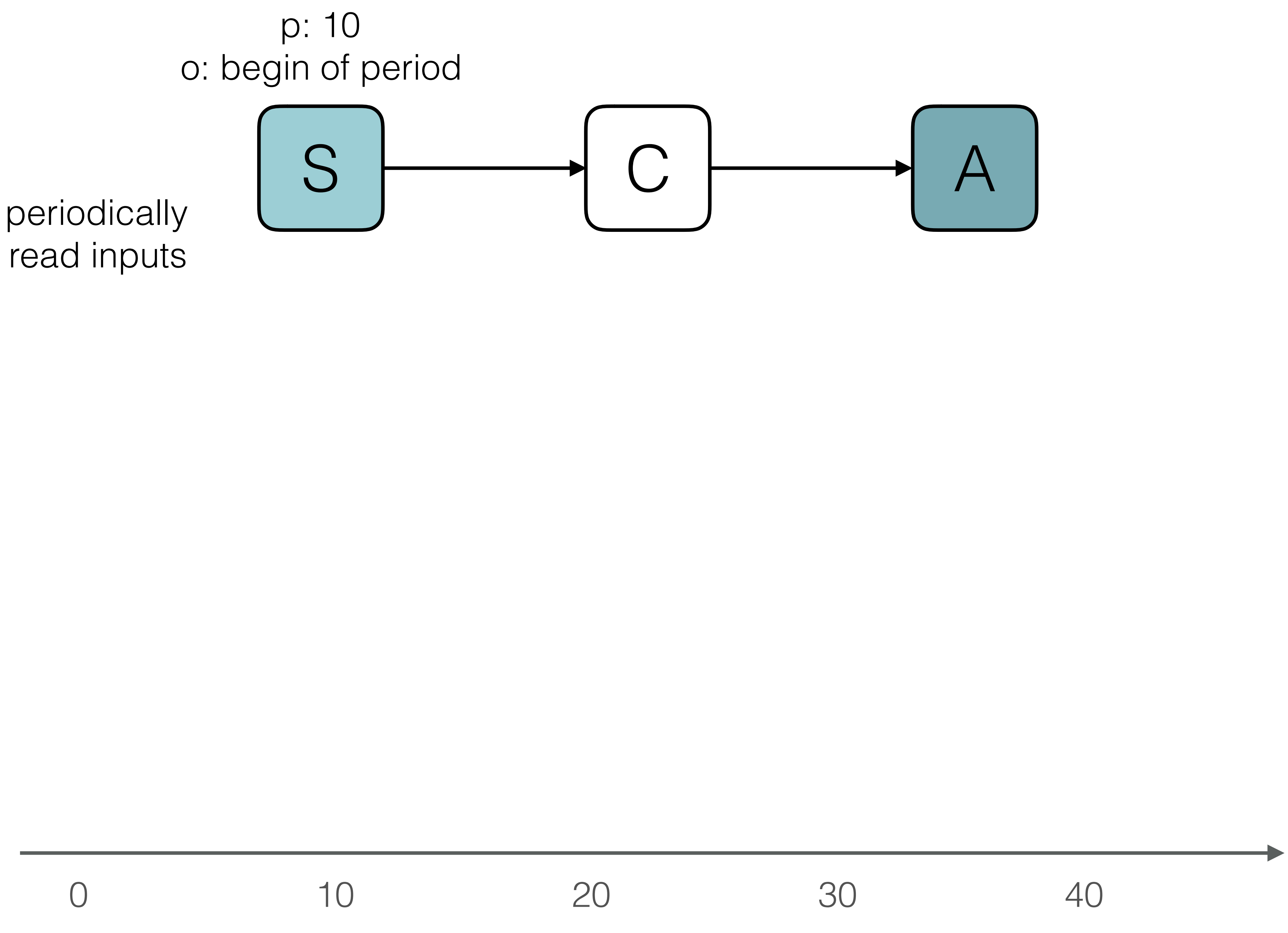


- IO node: the exact time of the interaction with the physics is called side effect. Side effects need timing specifications
- non IO nodes: do not have side effects, do not need timing specifications

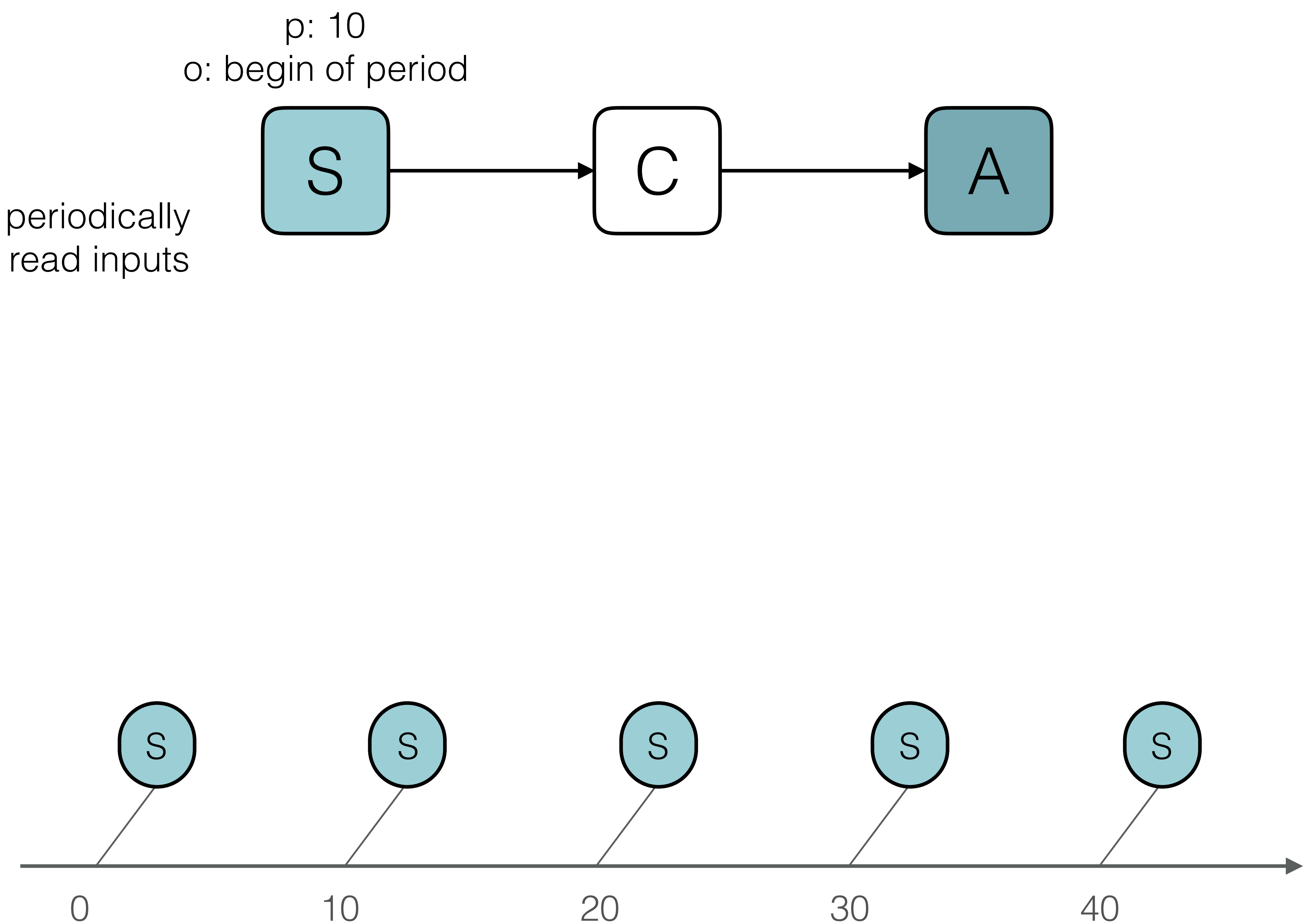
SDF with Timing



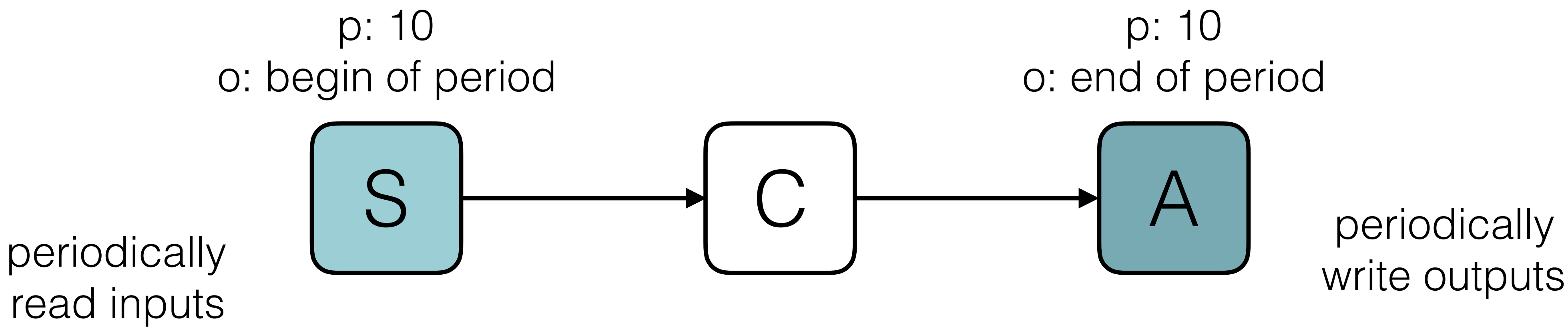
SDF with Timing

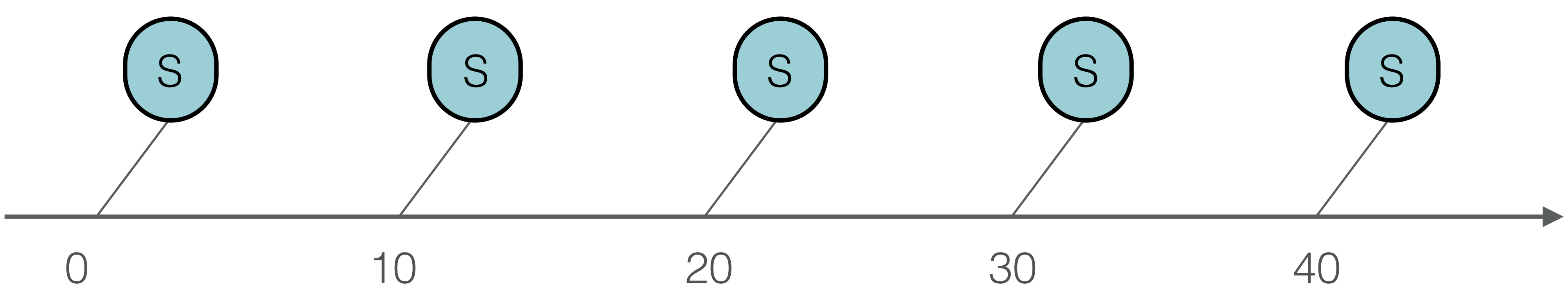


SDF with Timing

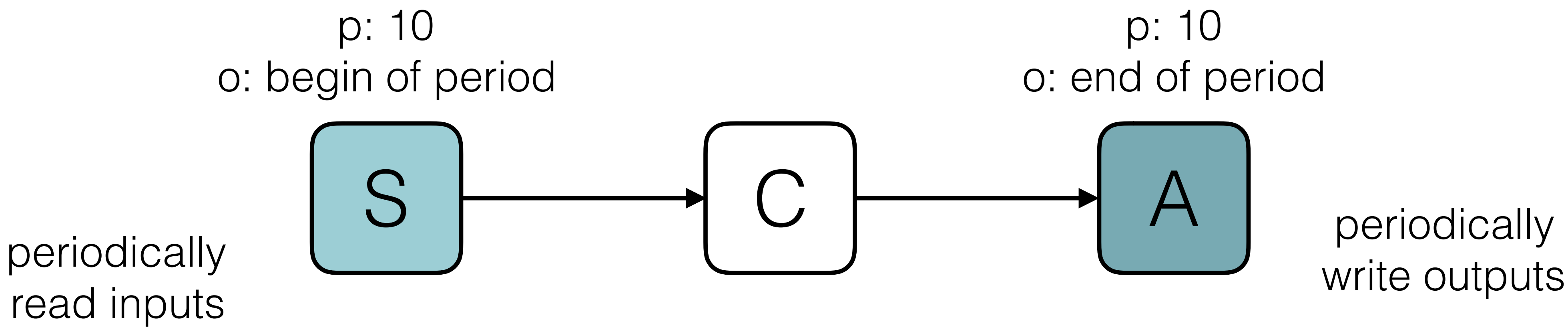


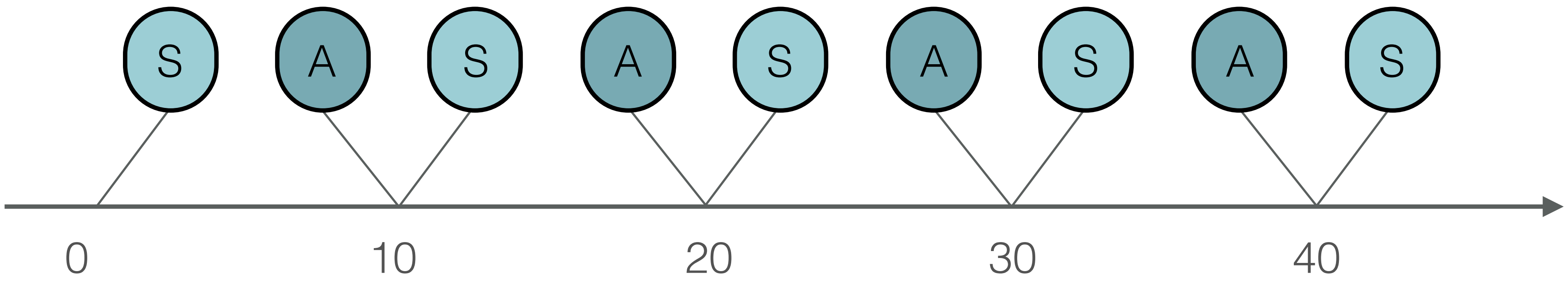
SDF with Timing





SDF with Timing



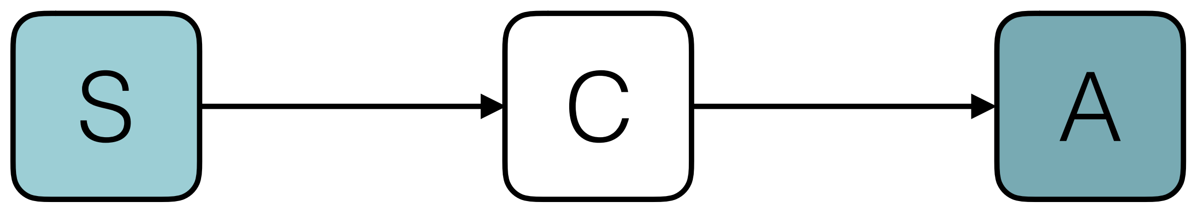


SDF with Timing

p: 10
o: begin of period

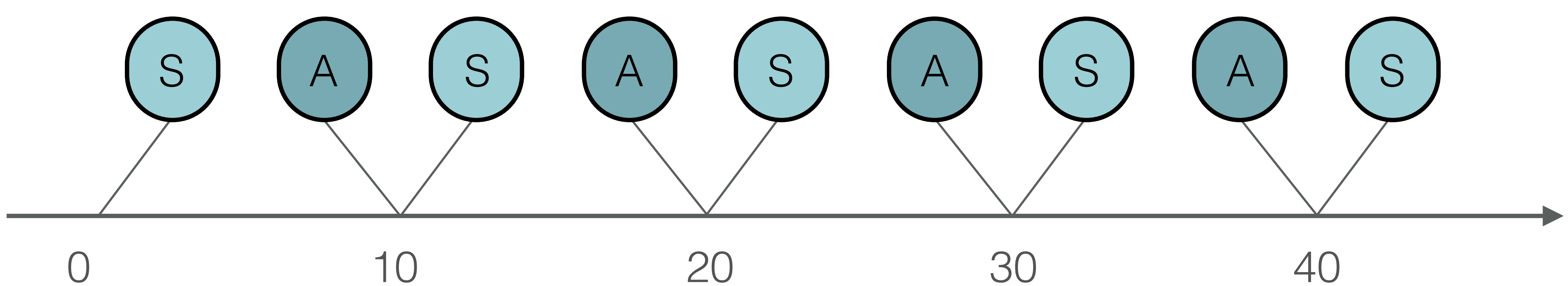
p: 10
o: end of period

periodically
read inputs



periodically
write outputs

consume inputs from incoming FIFO and produce tokens on outgoing FIFO fast enough such that A always has a new value to actuate

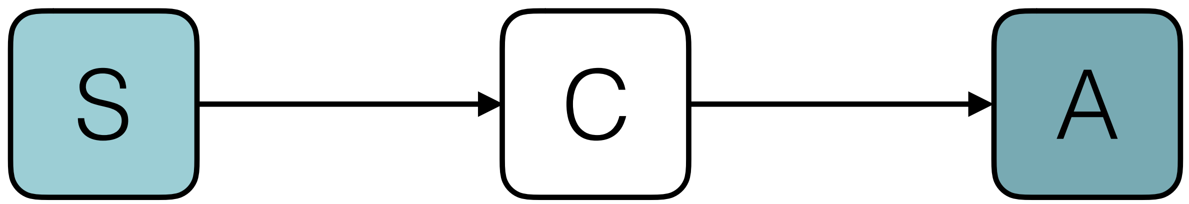


SDF with Timing

p: 10
o: begin of period

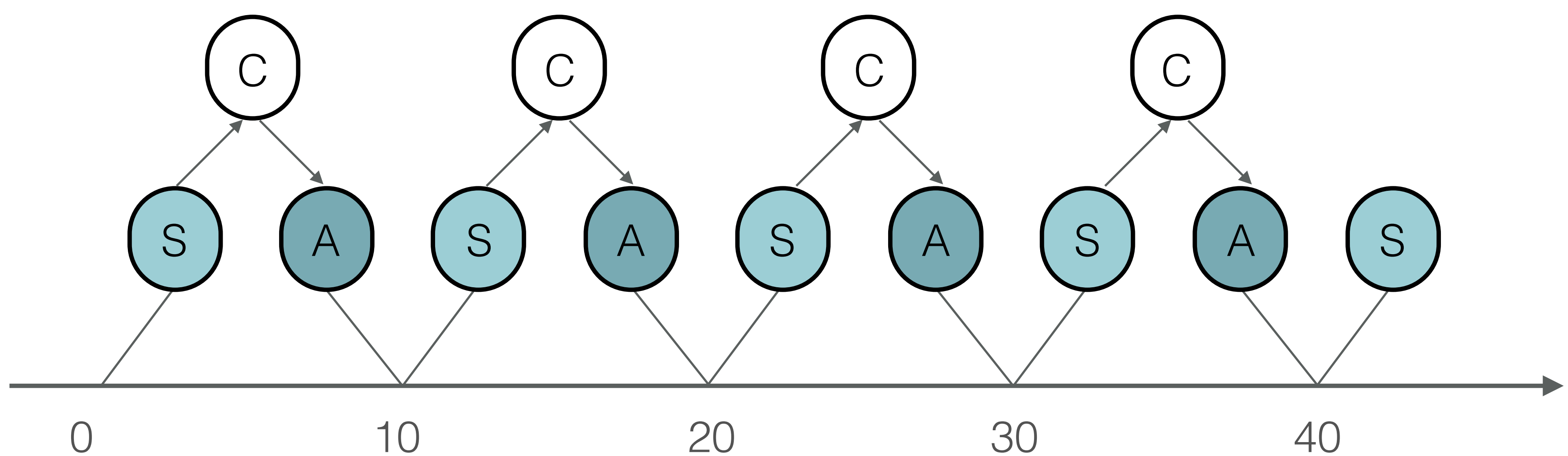
p: 10
o: end of period

periodically read inputs



periodically write outputs

consume inputs from incoming FIFO and produce tokens on outgoing FIFO fast enough such that A always has a new value to actuate

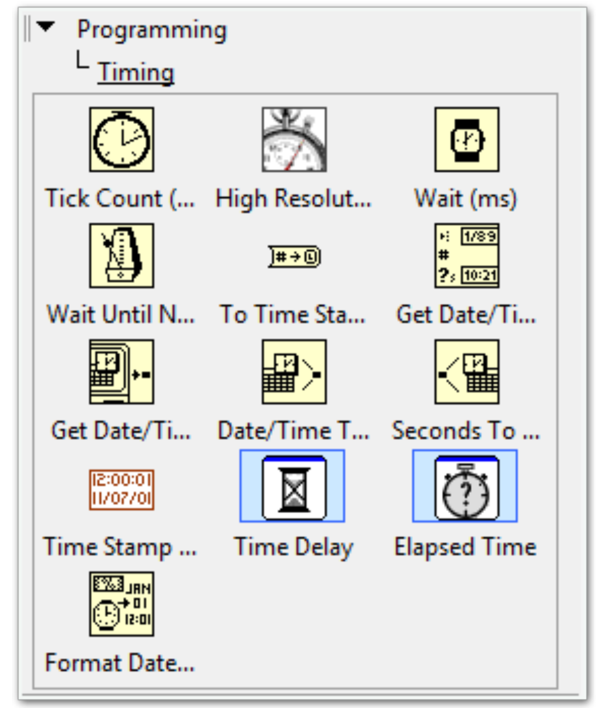


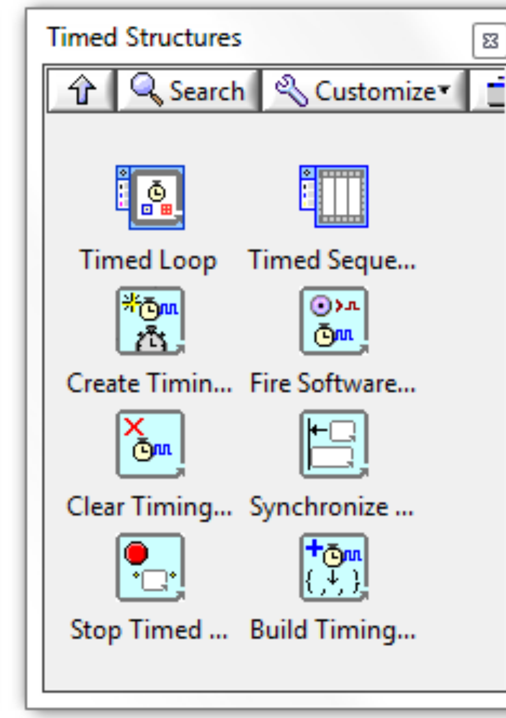
A LOT OF RESEARCH, BUT …

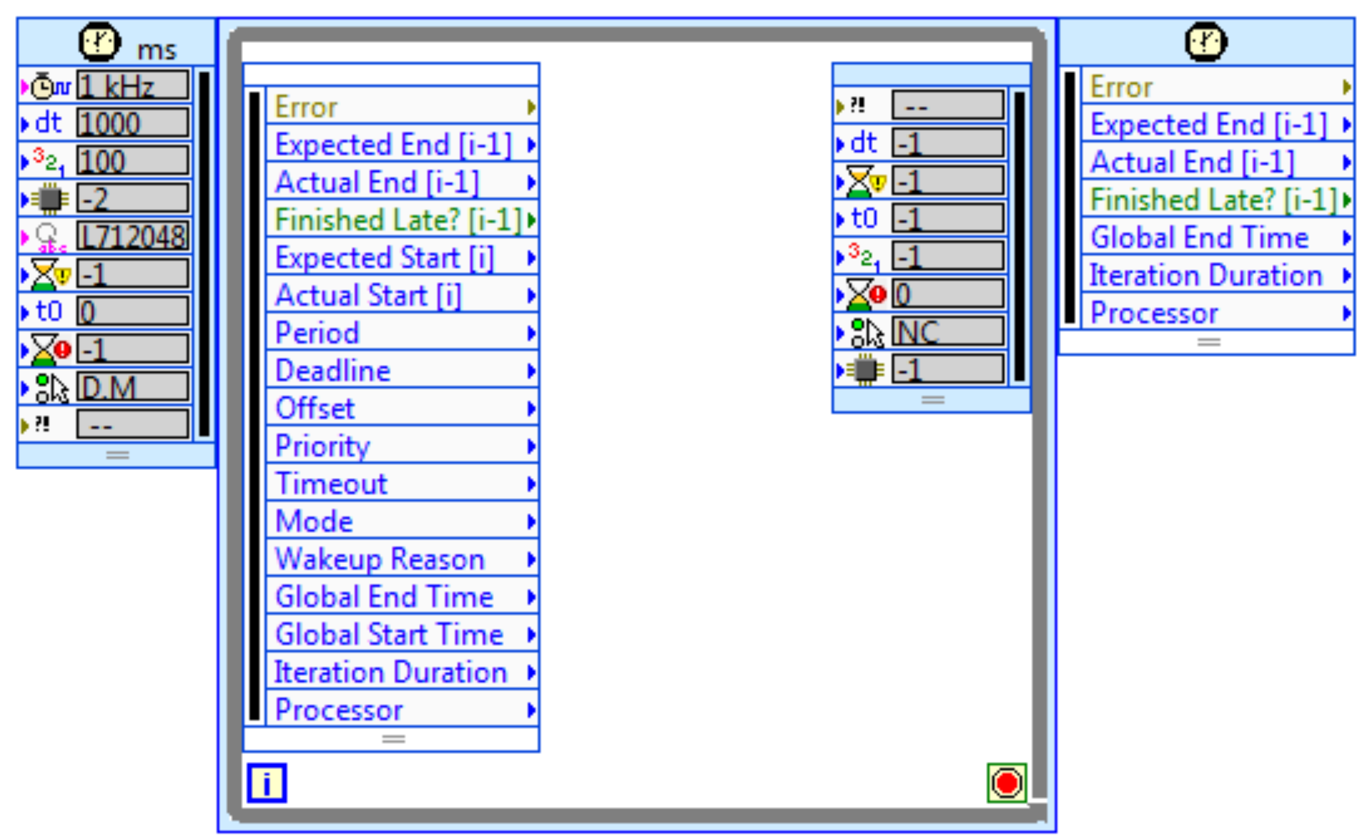
Already available today

Timing Specifications in LabVIEW G

- Time Sources: 1kHz, 1MHz, software triggered, ...
- Time Structures
  - Timed Loop
  - Timed Sequence
  - Single Cycle Time Loop (SCTL): for FPGA, executes all functions inside within one tick of the selected FPGA clock
- Programming with Time: Wait until time or tick, Measure elapsed time, Get current time, ...







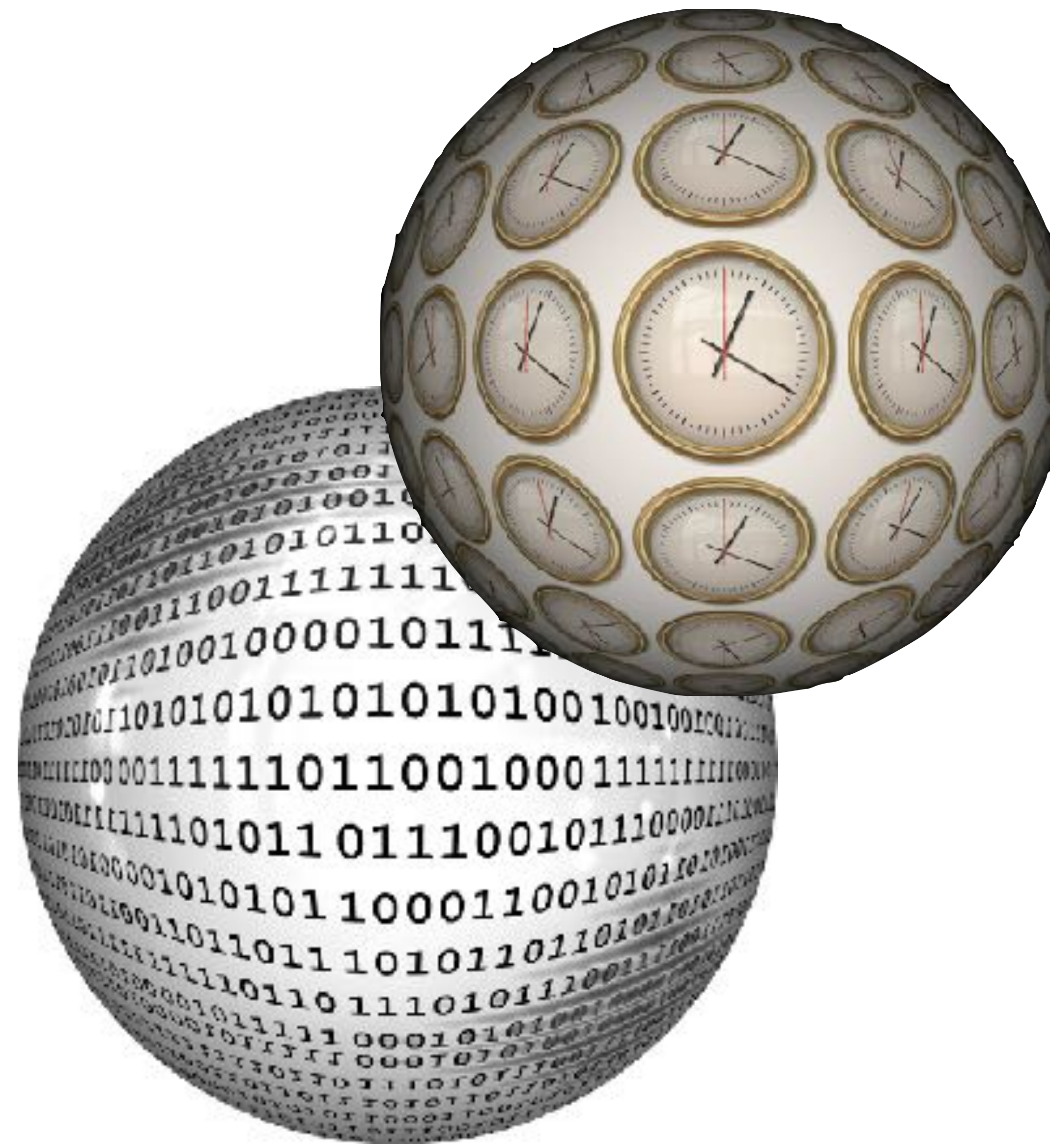
NATIONAL INSTRUMENTS

Summary

Thanks to advances in clock synchronization and time sensitive networks, we can now focus on how to **program with time**.

To build complex applications, we need the right levels of **abstraction**. Instead of abstracting away time, we should provide the right API to program with time.

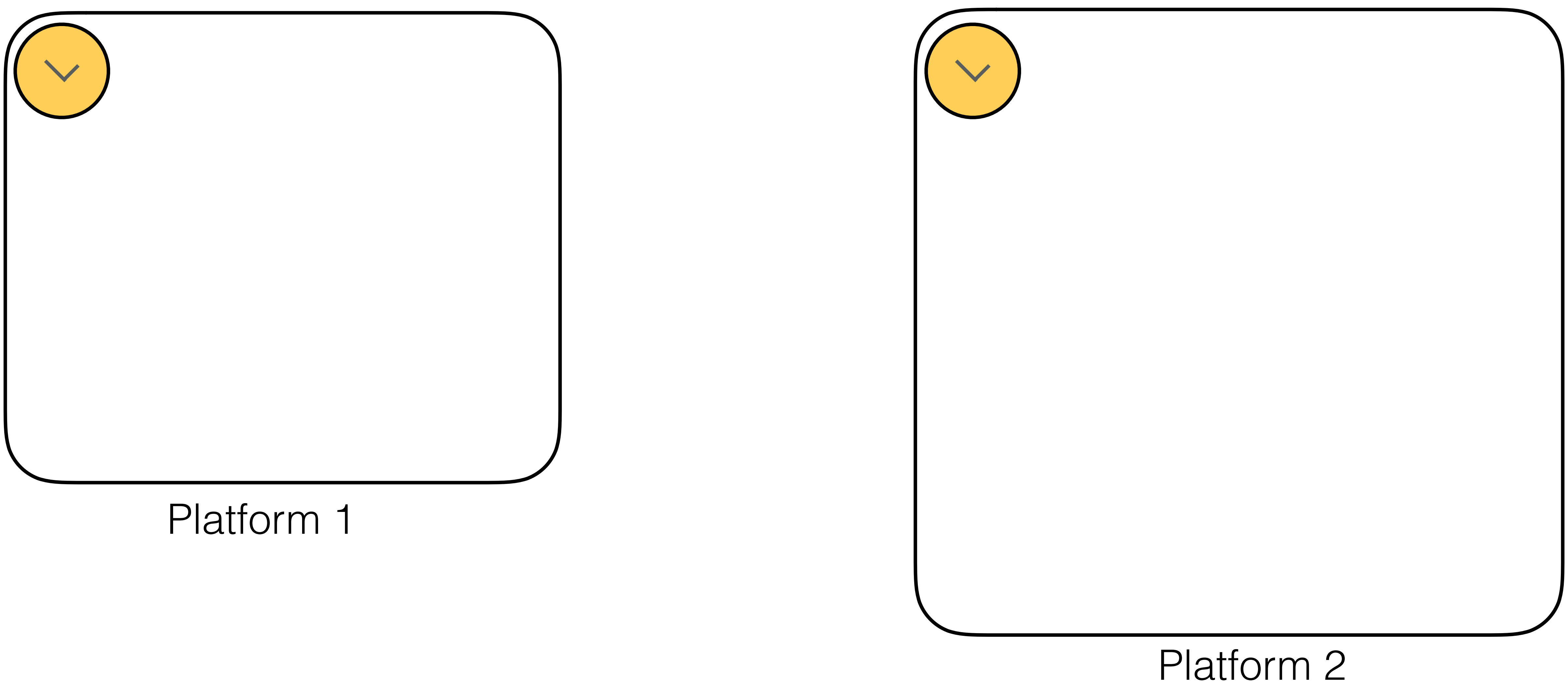
Academia has provided valuable programming models. Now it is time to pick these up in **industry**.



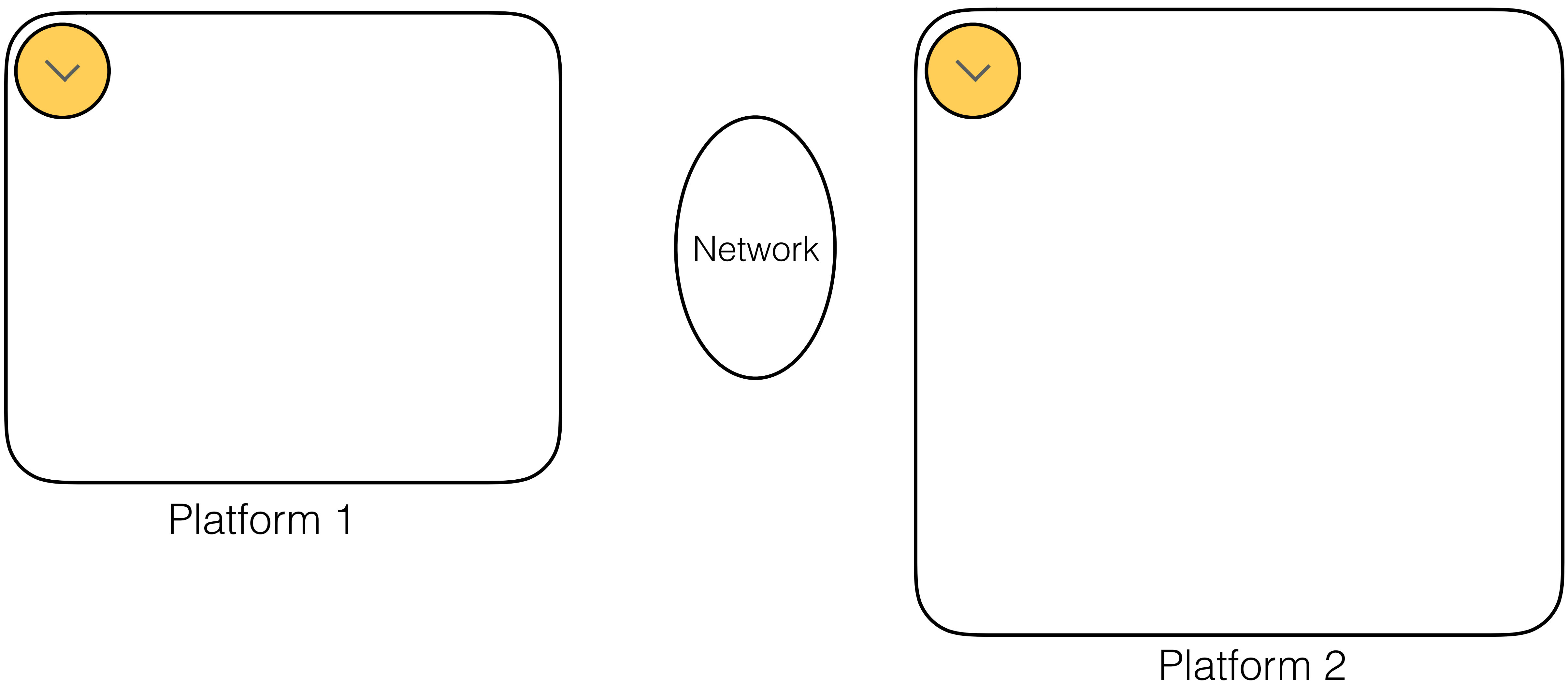
THANK YOU!

BACKUP SLIDES

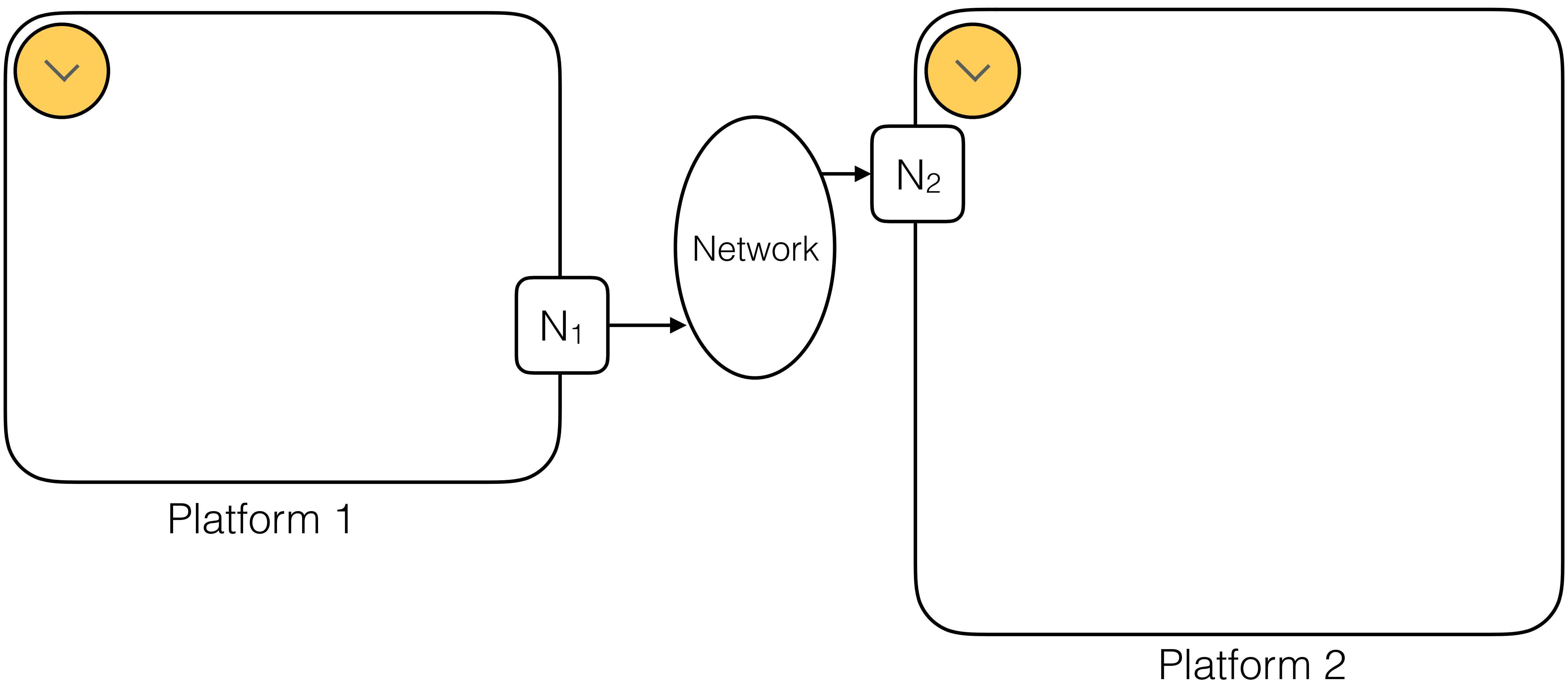
A Ptides Example



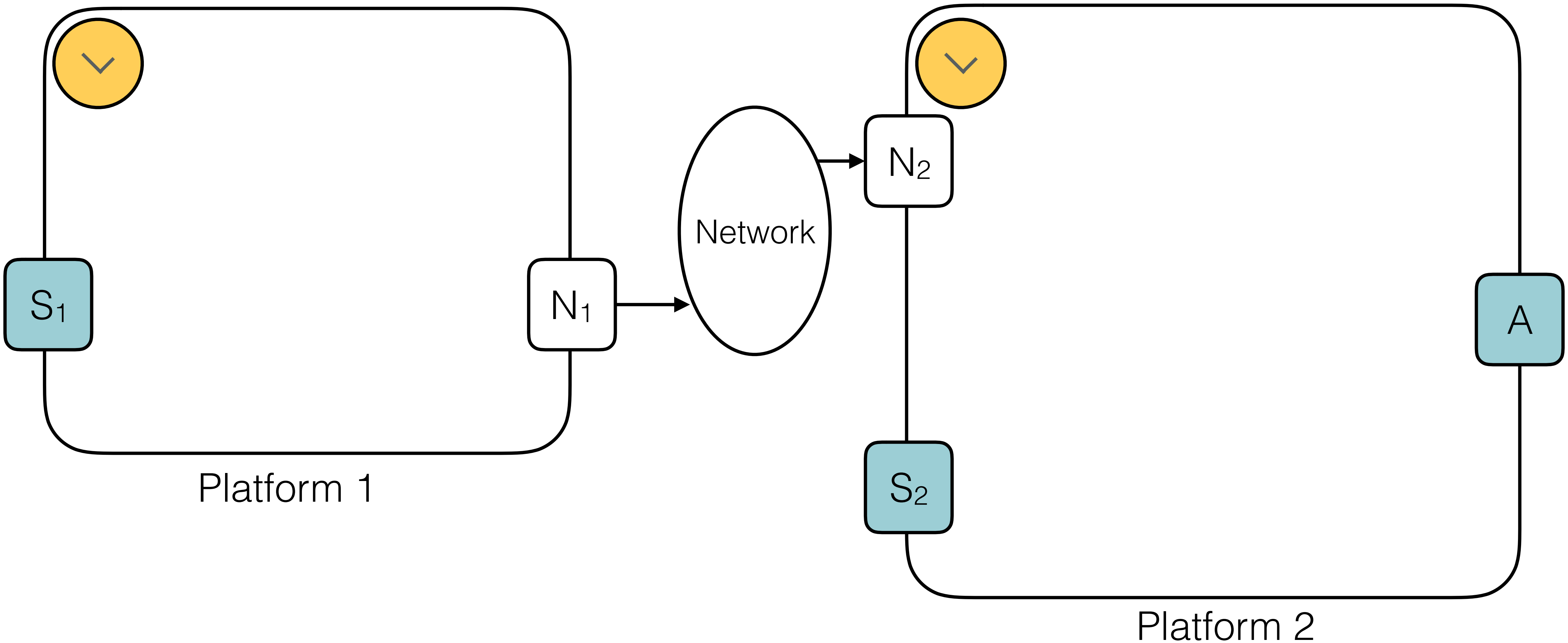
A Ptides Example



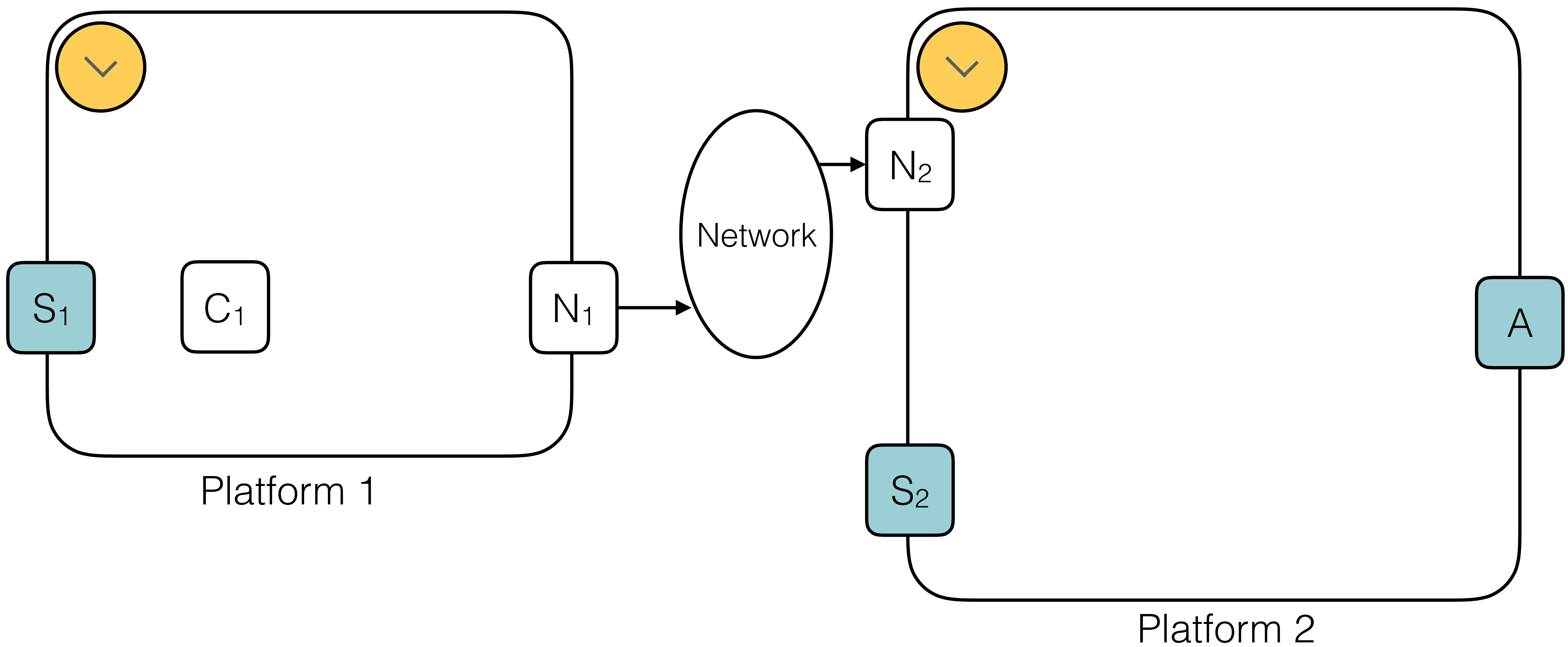
A Ptides Example



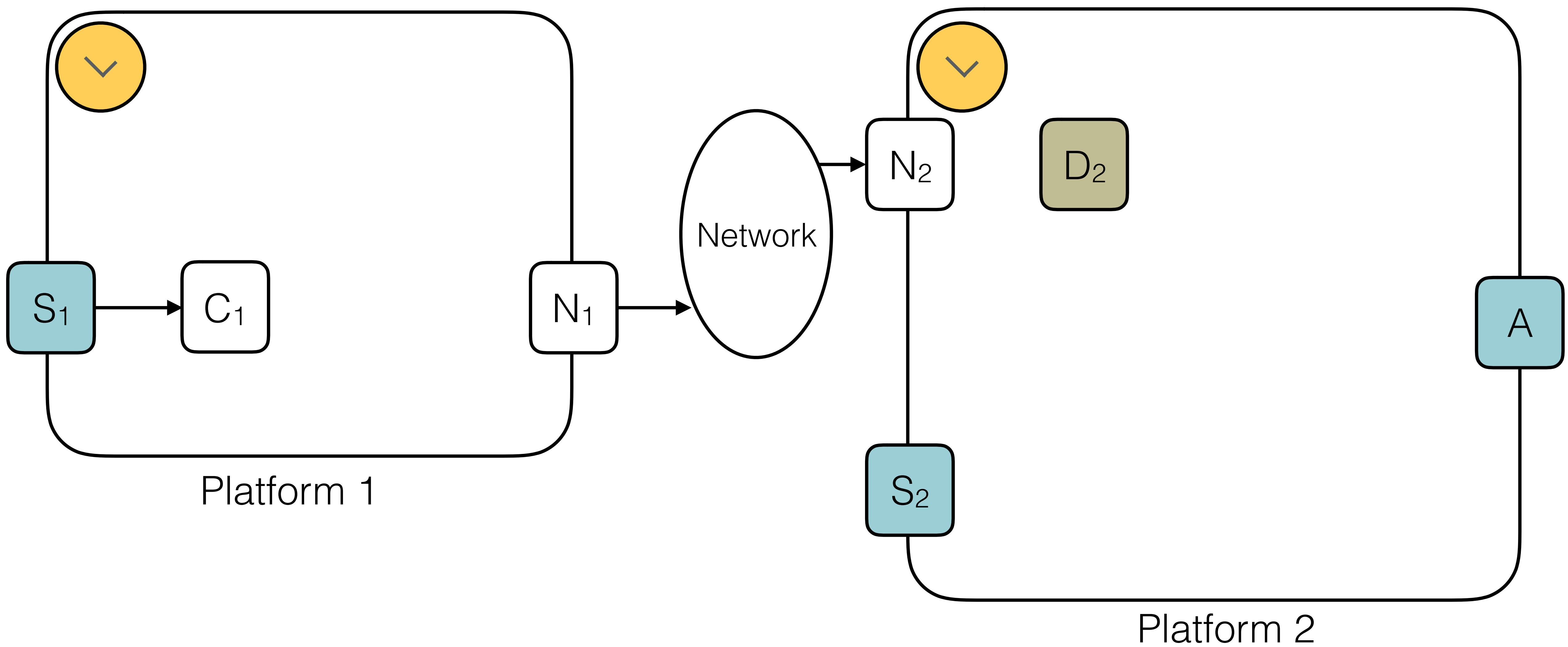
A Ptides Example



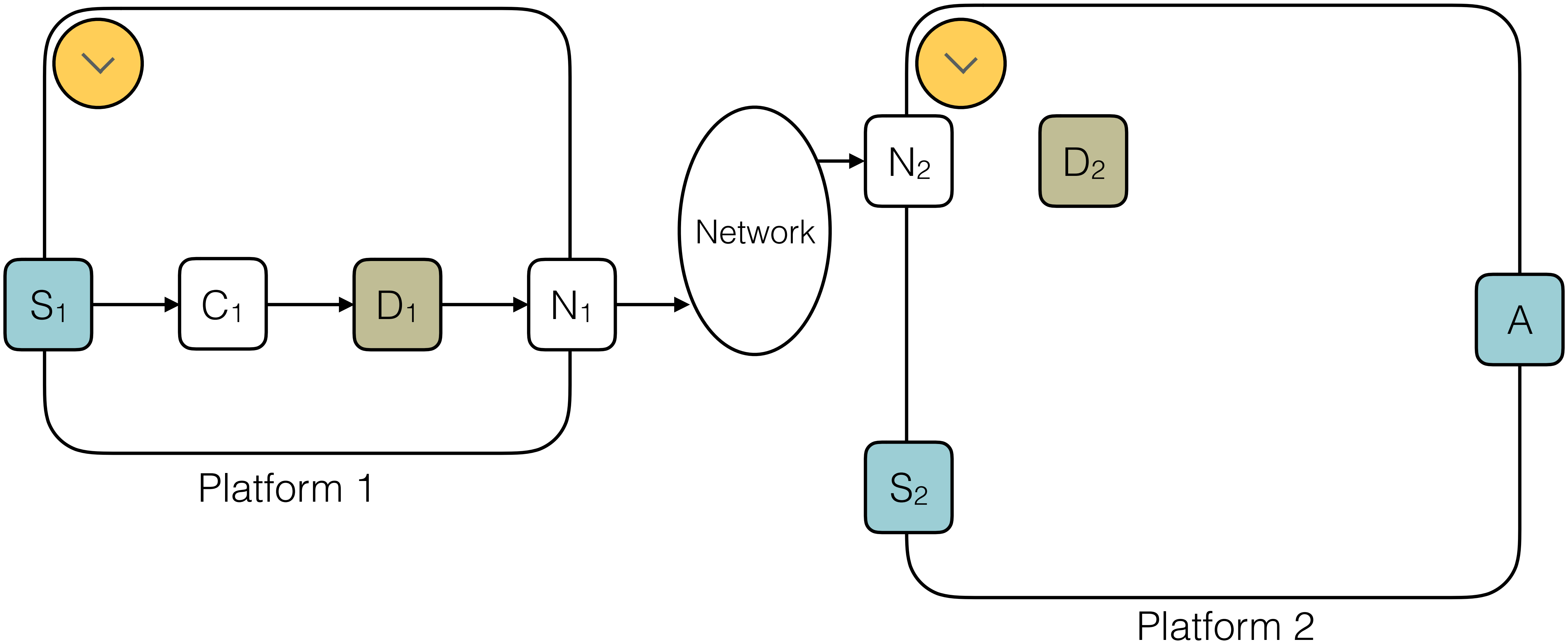
A Ptides Example



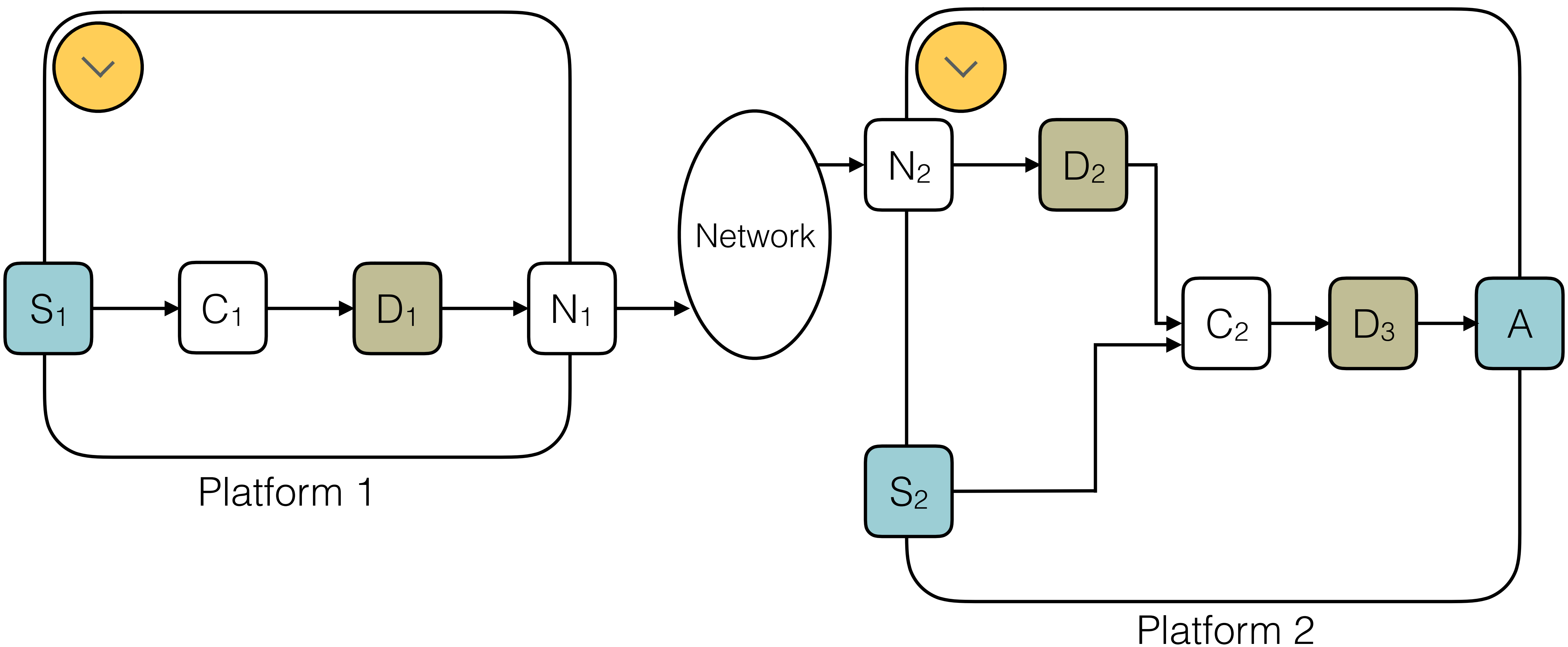
A Ptides Example



A Ptides Example

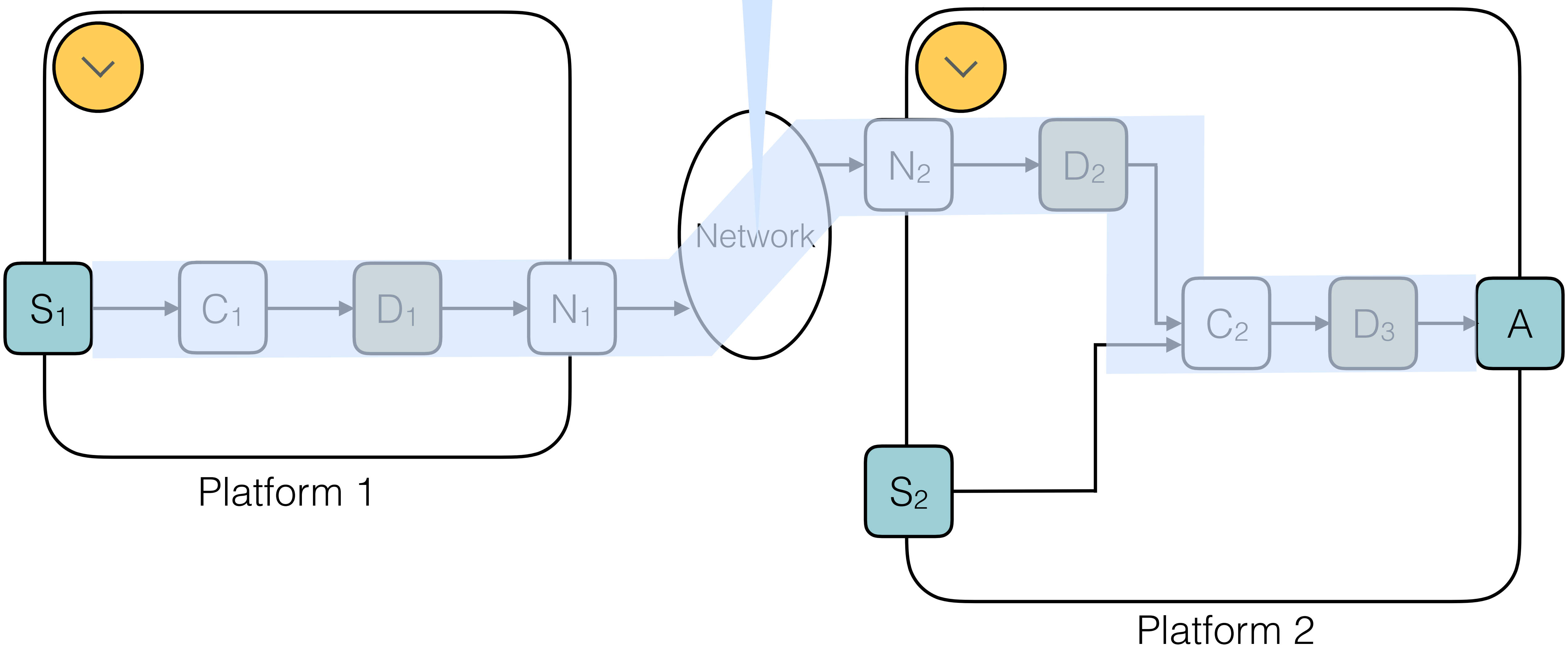


A Ptides Example



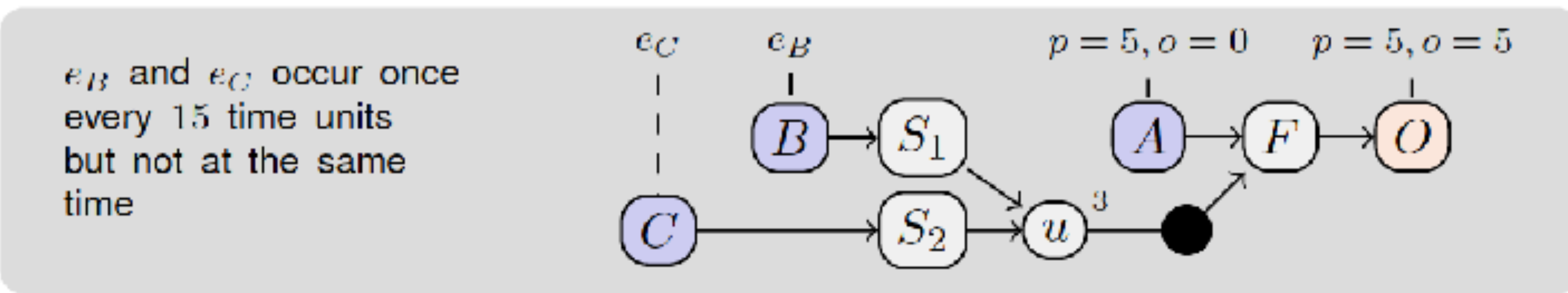
A Ptides Example

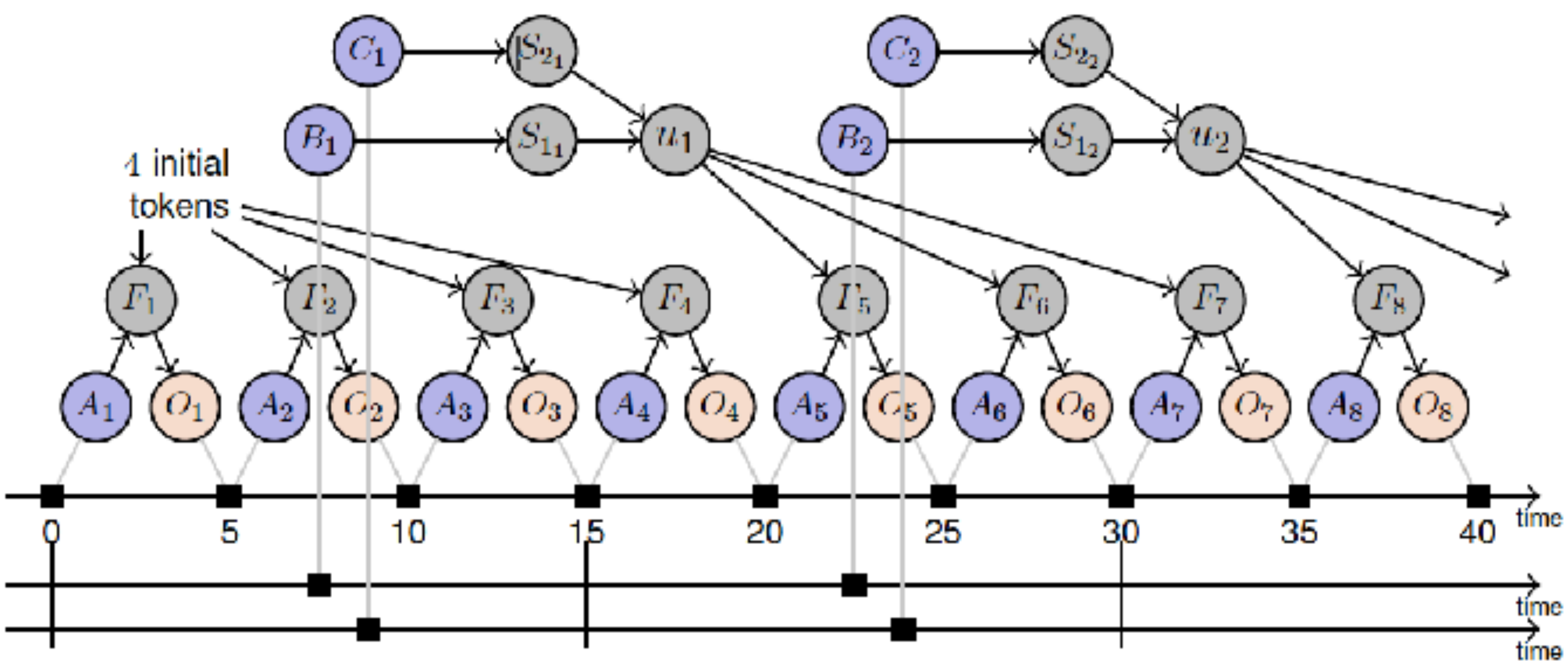
Latency between S_1 and A = D_1+ D_2+ D_3
All execution and network transmission must finish within that time



NATIONAL INSTRUMENTS

SDF with Timing





- Total execution time for A, F and O must be < 5
- First 4 outputs by O use initial values
- An input on B (or C) influences O within 10 to 35 time units
- Total execution time along path B, S1, u, F and O cannot exceed 10 time units
- Pattern of node executions repeats every 15 time units after an initial phase that lasts 20 time units