ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Engineering and
Networks Laboratory

Modeling Structured Event Streams in System Level Performance Analysis

Simon Perathoner, Tobias Rein, Lothar Thiele, Kai Lampka, Jonas Rox

LCTES 2010, Stockholm, Sweden

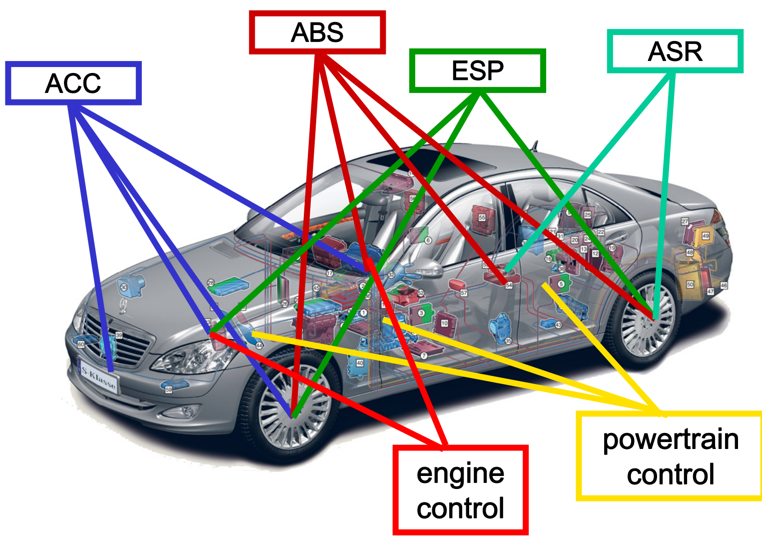
April 13th, 2010

Outline

- Motivation
- Modular Performance Analysis with Real-Time Calculus
- Approach 1: FIFO Scheduling
- Approach 2: Event Count Curves
- Comparison / Case Study
- Conclusion

Motivation

Distributed Stream-based Embedded Systems



Design and analysis are complex due to:

- Concurrency
- Interference on shared resources
- Non-determinism in timing

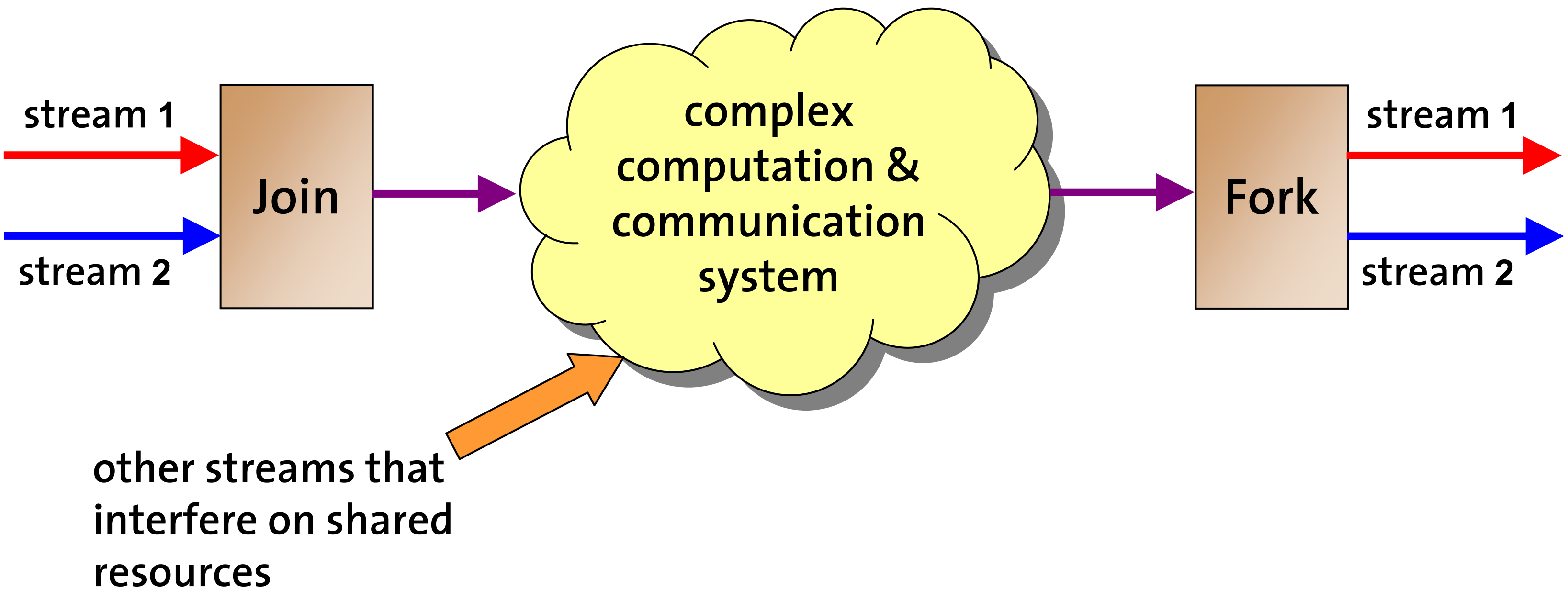
Requirements for Performance Analysis:

- Coverage of system behaviors
- Accuracy of results
- Computational efficiency

Analytic methods for Modular Performance Analysis:
- MPA-RTC (ETHZ)
- SymTA/S (TUBS)

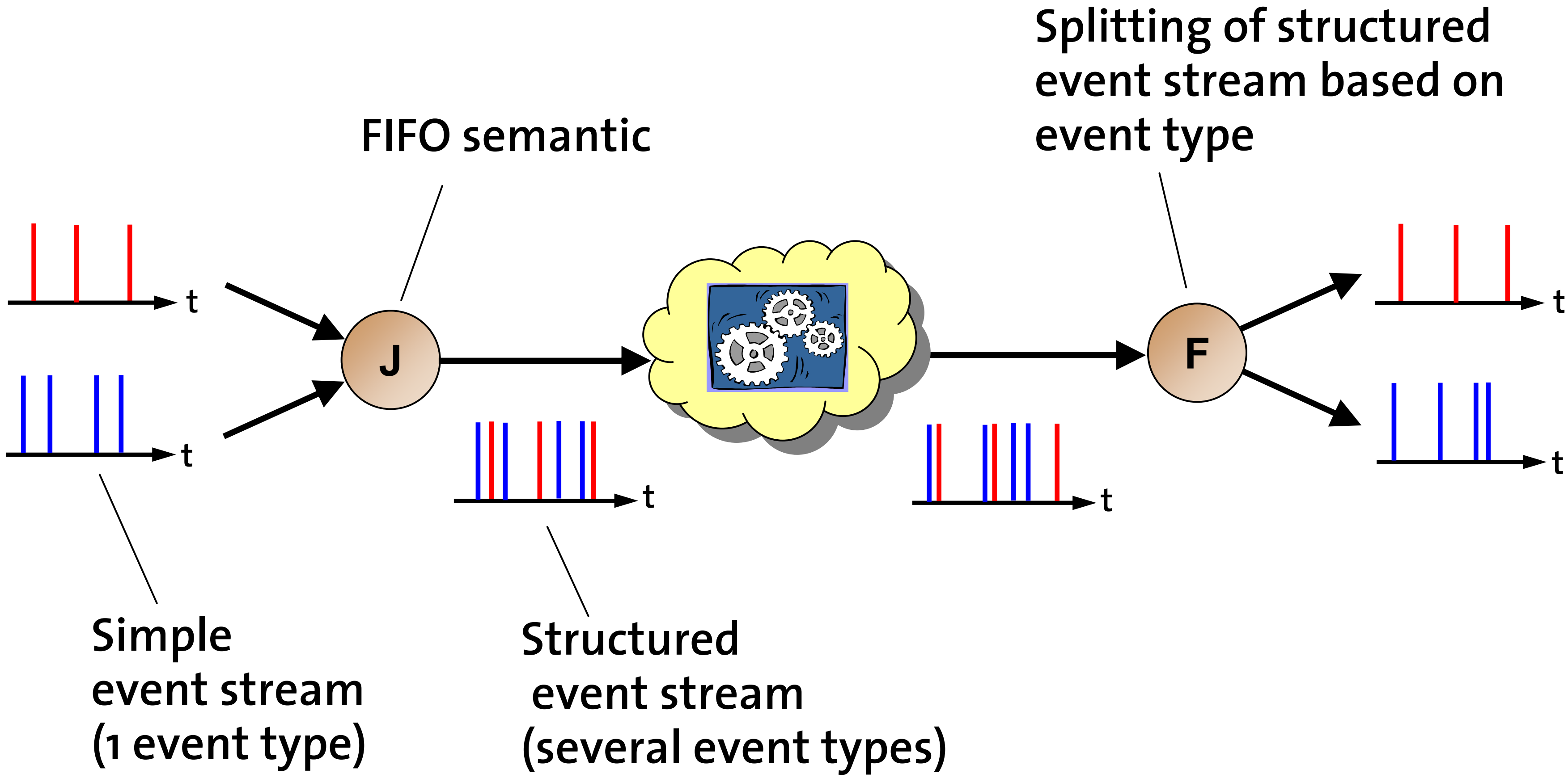
Motivation

Relevant design pattern: Join and Fork of event or data streams



Goal: Extend analytic methods for performance analysis such that join/fork operations on streams can be captured accurately

Definitions

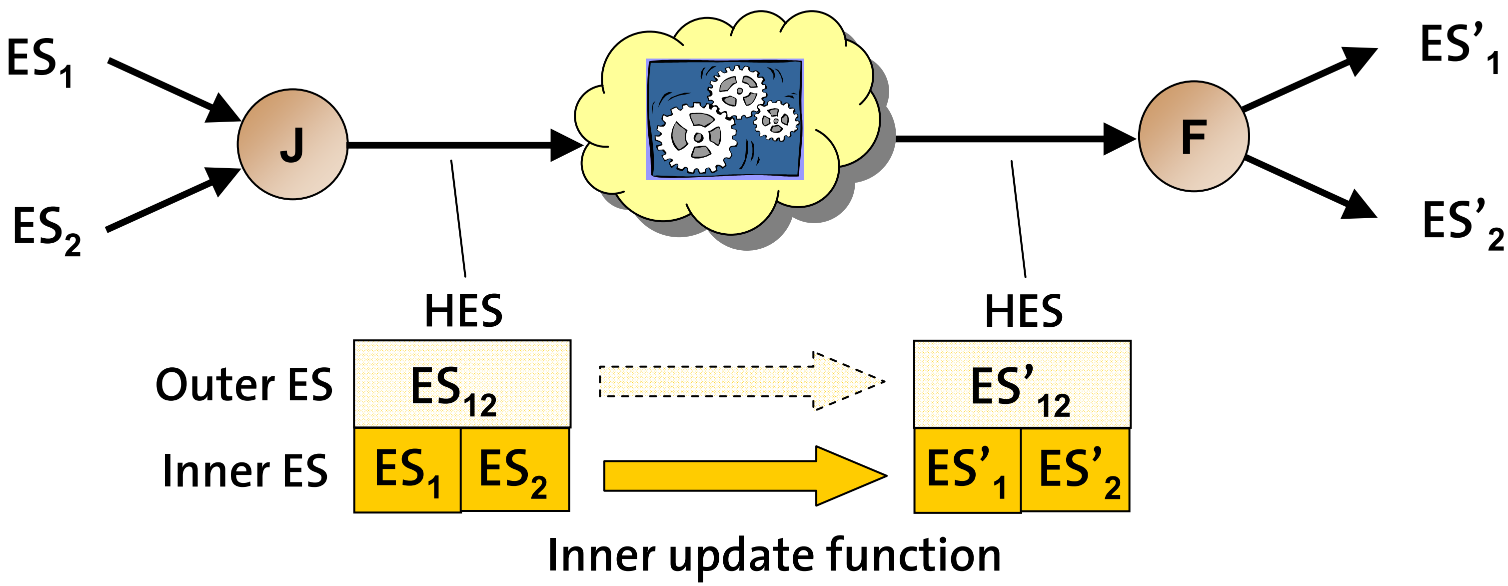


Event type = provenance

Related Work

Hierarchical Event Model (TU Braunschweig)

J. Rox & R. Ernst, *Modeling Event Stream Hierarchies with Hierarchical Event Models,* DATE 2008



Limitations: - Method is not transparent to existing component models

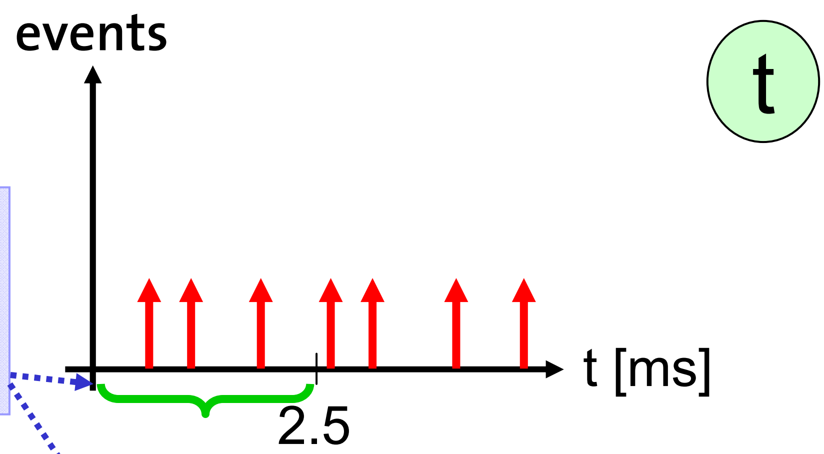
- Deep processing of HES required by component models

Outline

- Motivation
- Modular Performance Analysis with Real-Time Calculus
- Approach 1: FIFO Scheduling
- Approach 2: Event Count Curves
- Comparison / Case Study
- Conclusion

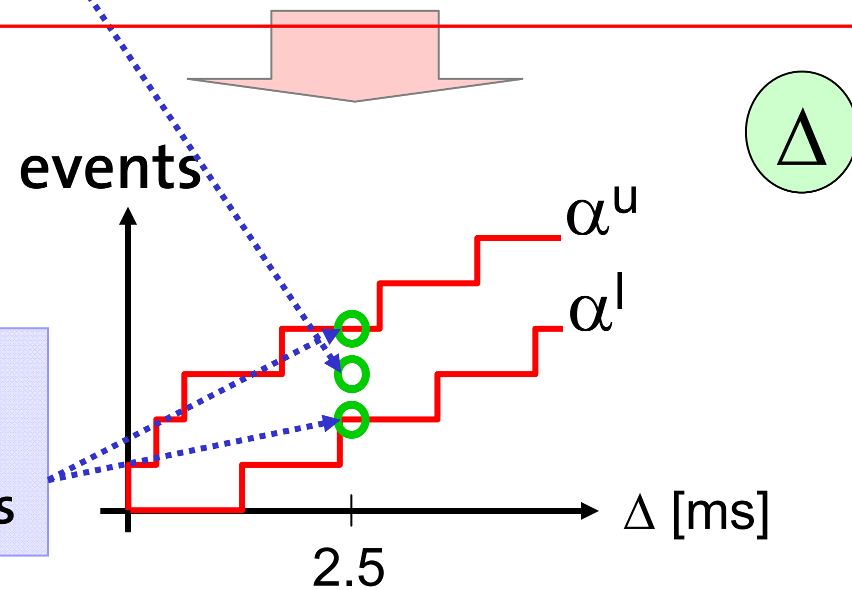
Arrival Curves

Event Stream



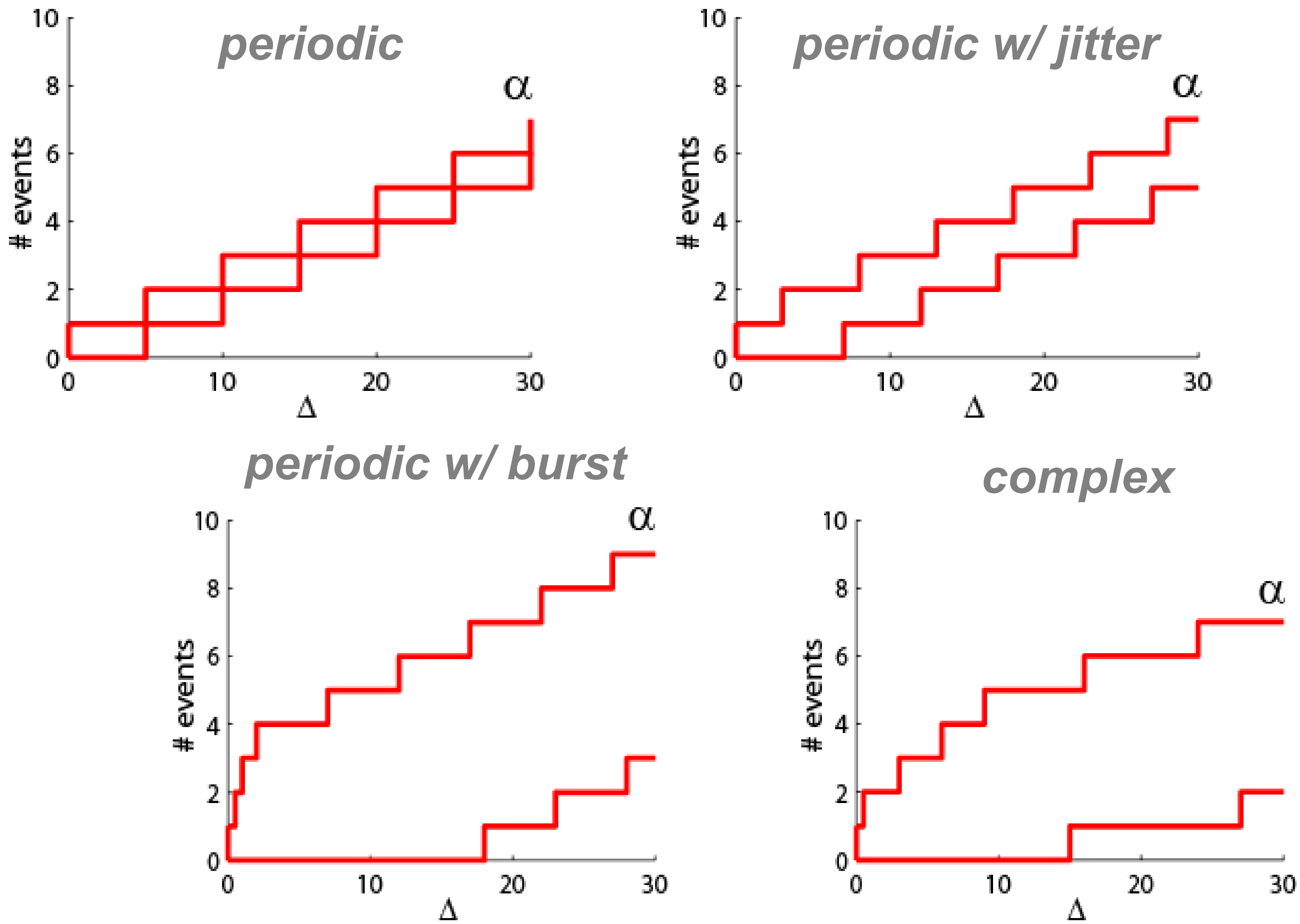
number of events in
in t=[0 .. 2.5] ms

Arrival Curves $[\alpha^l, \alpha^u]$



maximum/minimum
arriving events in *any*
interval of length 2.5 ms

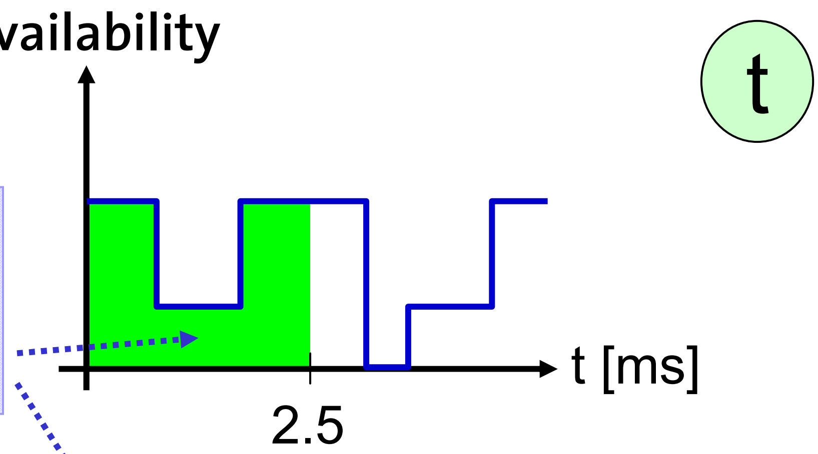
Examples of Arrival Curves



Service Curves

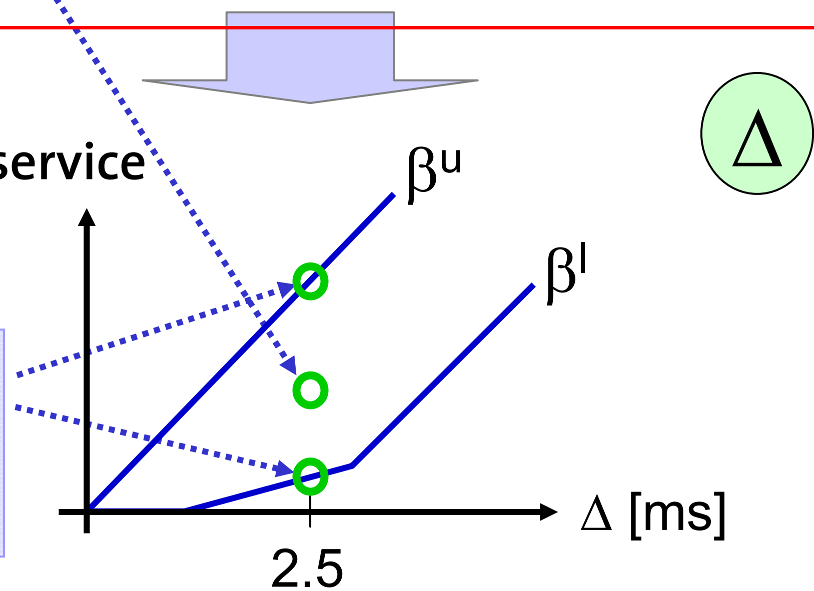
Resource Availability

available service
in t=[0 .. 2.5] ms



Service Curves $[\beta^l, \beta^u]$

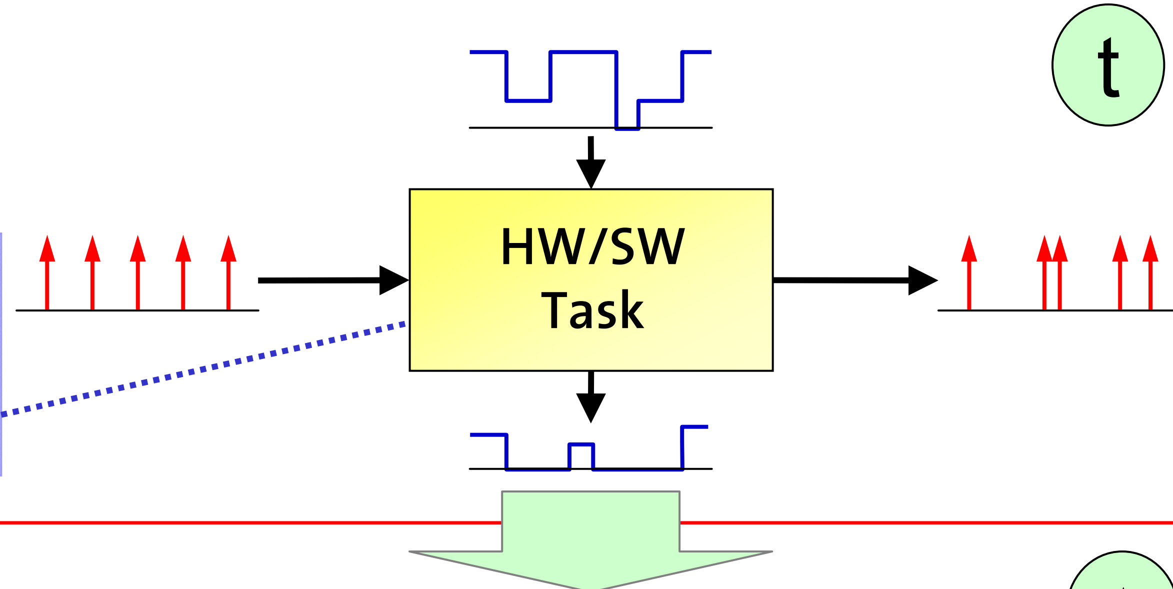
maximum/minimum
available service in *any*
interval of length 2.5 ms



Processing Model (HW/SW)

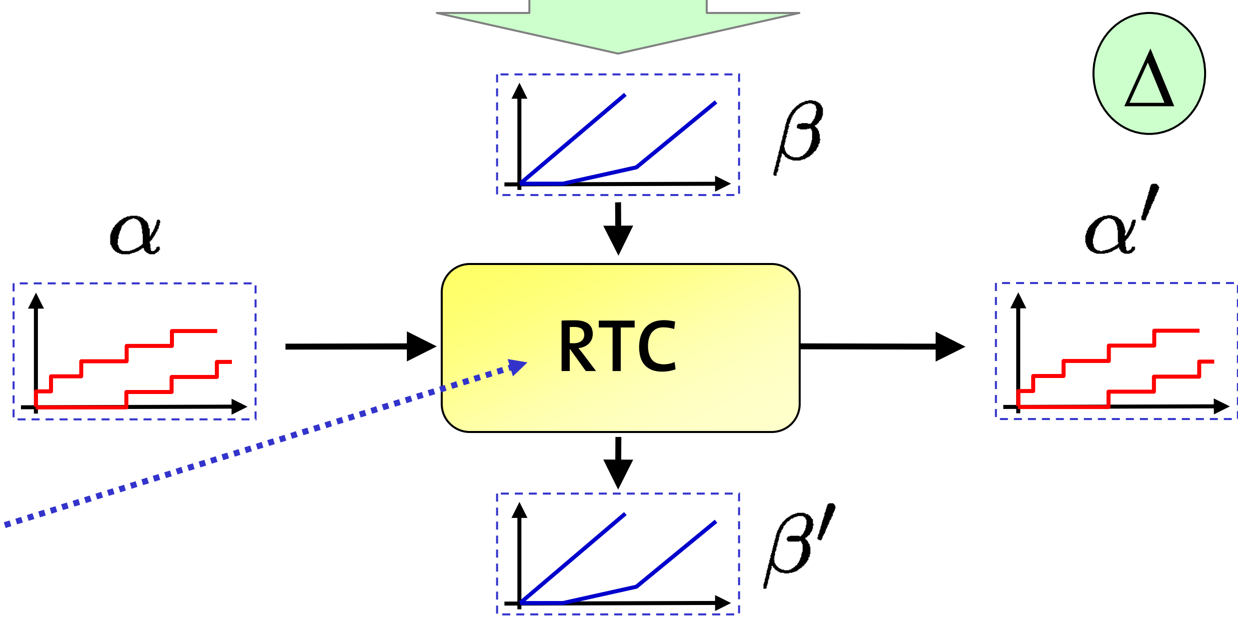
HW/SW Components

Processing semantics and functionality of hardware or software tasks

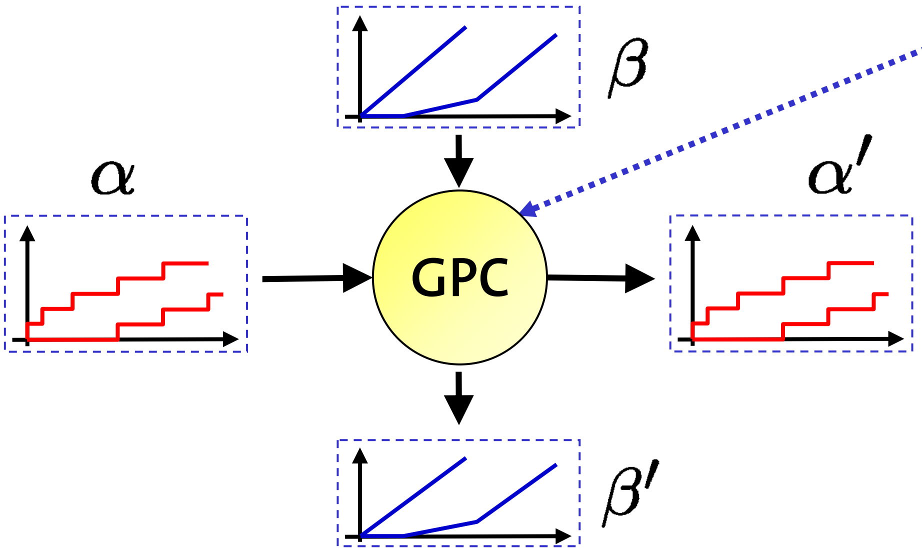


Abstract Components

$$\alpha'(\Delta) = f_\alpha(\alpha, \beta)$$
$$\beta'(\Delta) = f_\beta(\alpha, \beta)$$



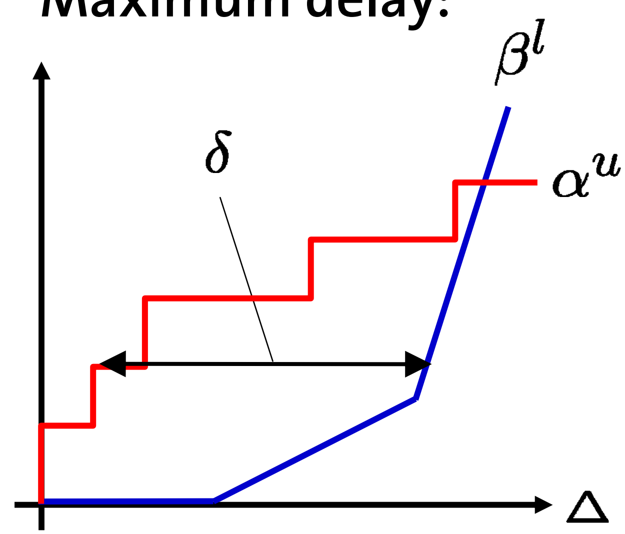
Example: Greedy Processing Component (GPC)



$$\begin{aligned}
\alpha'^u &= GPC_{\alpha'^u}(\alpha^u, \beta^u, \beta^l) \\
\alpha'^l &= GPC_{\alpha'^l}(\alpha^l, \beta^u, \beta^l) \\
\beta'^u &= GPC_{\beta'^u}(\beta^u, \alpha^l) \\
\beta'^l &= GPC_{\beta'^l}(\beta^l, \alpha^u)
\end{aligned}$$

- FIFO input buffer
- Greedy event processing

Maximum delay:



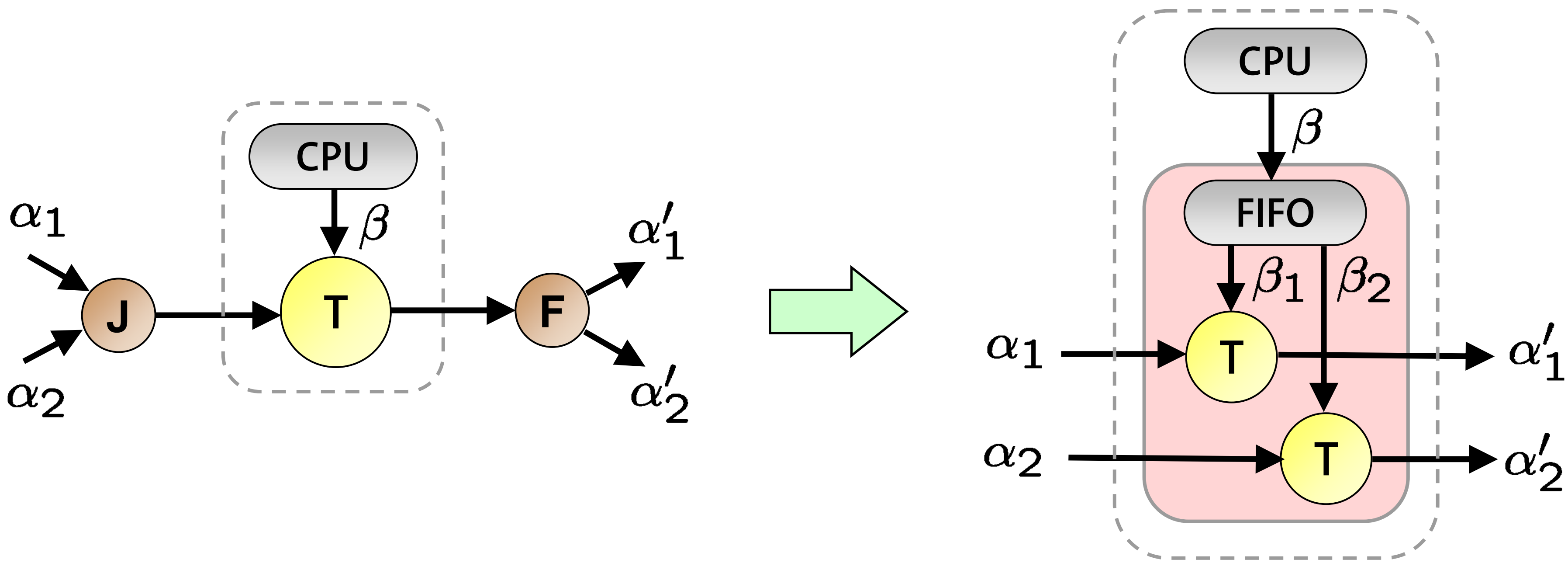
Outline

- Motivation
- Modular Performance Analysis with Real-Time Calculus
- Approach 1: FIFO Scheduling
- Approach 2: Event Count Curves
- Comparison / Case Study
- Conclusion

FIFO Scheduling

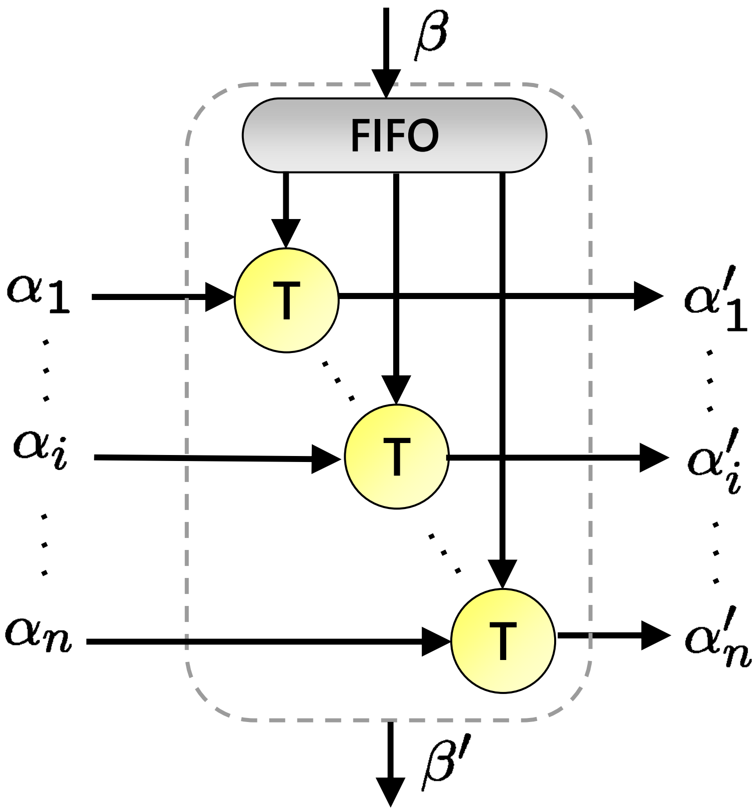
- Keep sub-streams separated in the model
- Adapt existing component models

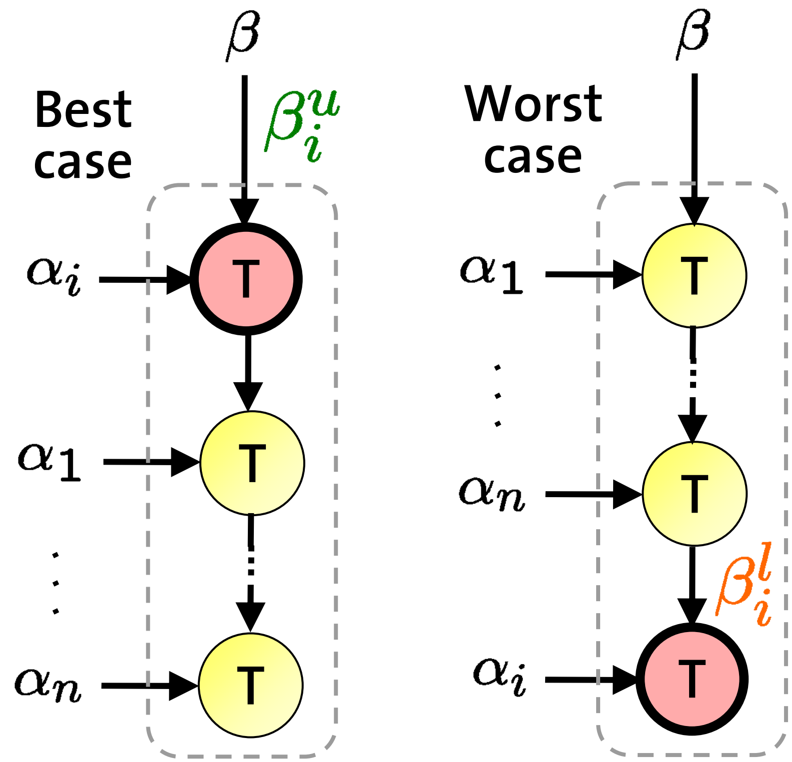
Abstract FIFO component
for the MPA framework



Modeling approach justified by FIFO semantics of Join operator

FIFO Component in MPA





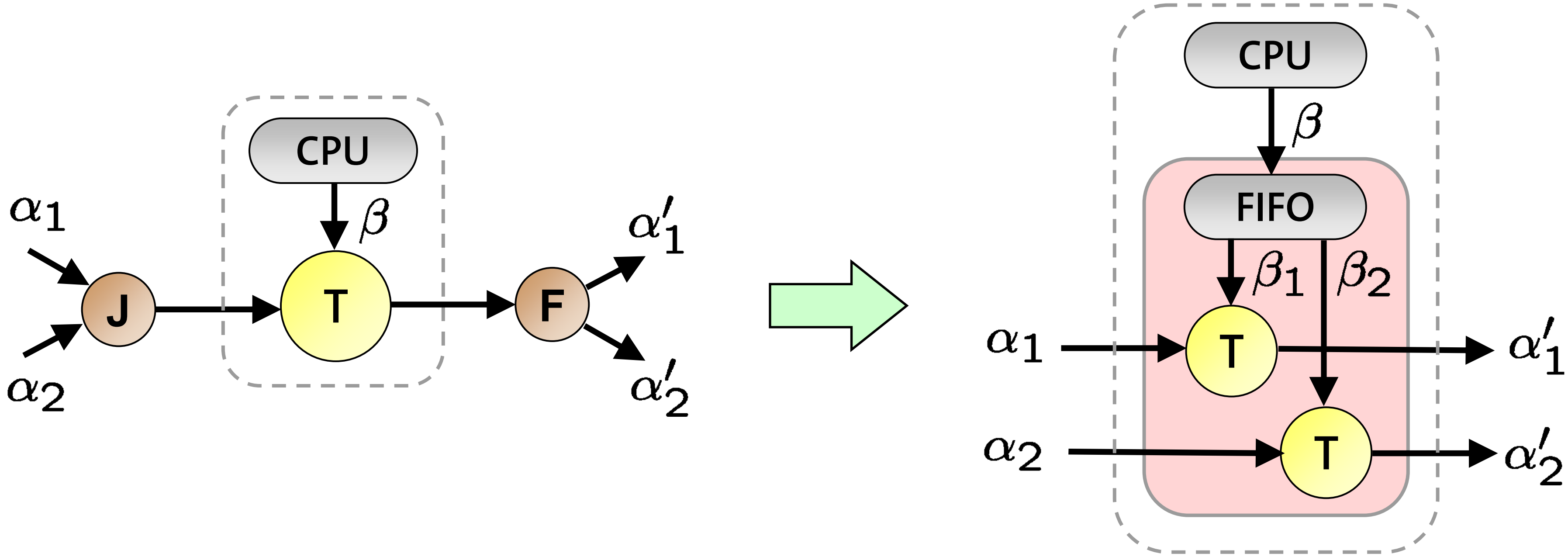
$$\beta'^u = GPC_{\beta'^u}(\beta^u, \sum_i \alpha_i^l)$$

$$\beta'^l = GPC_{\beta'^l}(\beta^l, \sum_i \alpha_i^u)$$

$$\alpha_i'^u = GPC_{\alpha'^u}(\alpha_i^u, \beta_i^u, \beta_i^l)$$

$$\alpha_i'^l = GPC_{\alpha'^l}(\alpha_i^l, \beta_i^u, \beta_i^l)$$

FIFO Scheduling



Abstract processing component handles structure of input stream explicitly

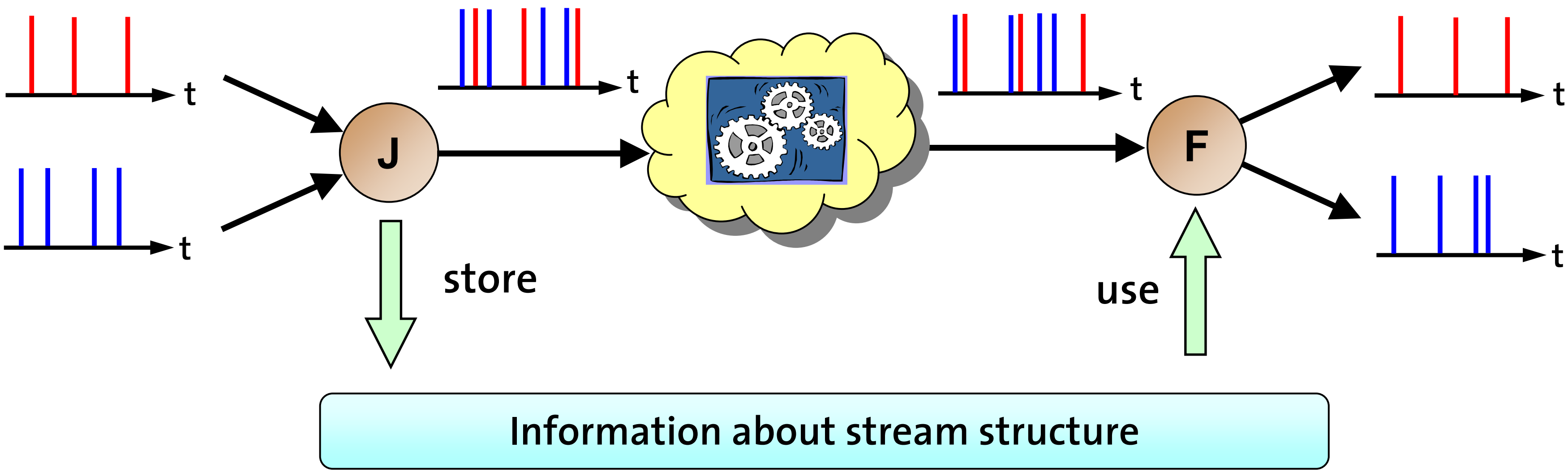
$\Rightarrow$ **Method is not transparent to existing component models!**

Outline

- Motivation
- Modular Performance Analysis with Real-Time Calculus
- Approach 1: FIFO Scheduling
- Approach 2: Event Count Curves
- Comparison / Case Study
- Conclusion

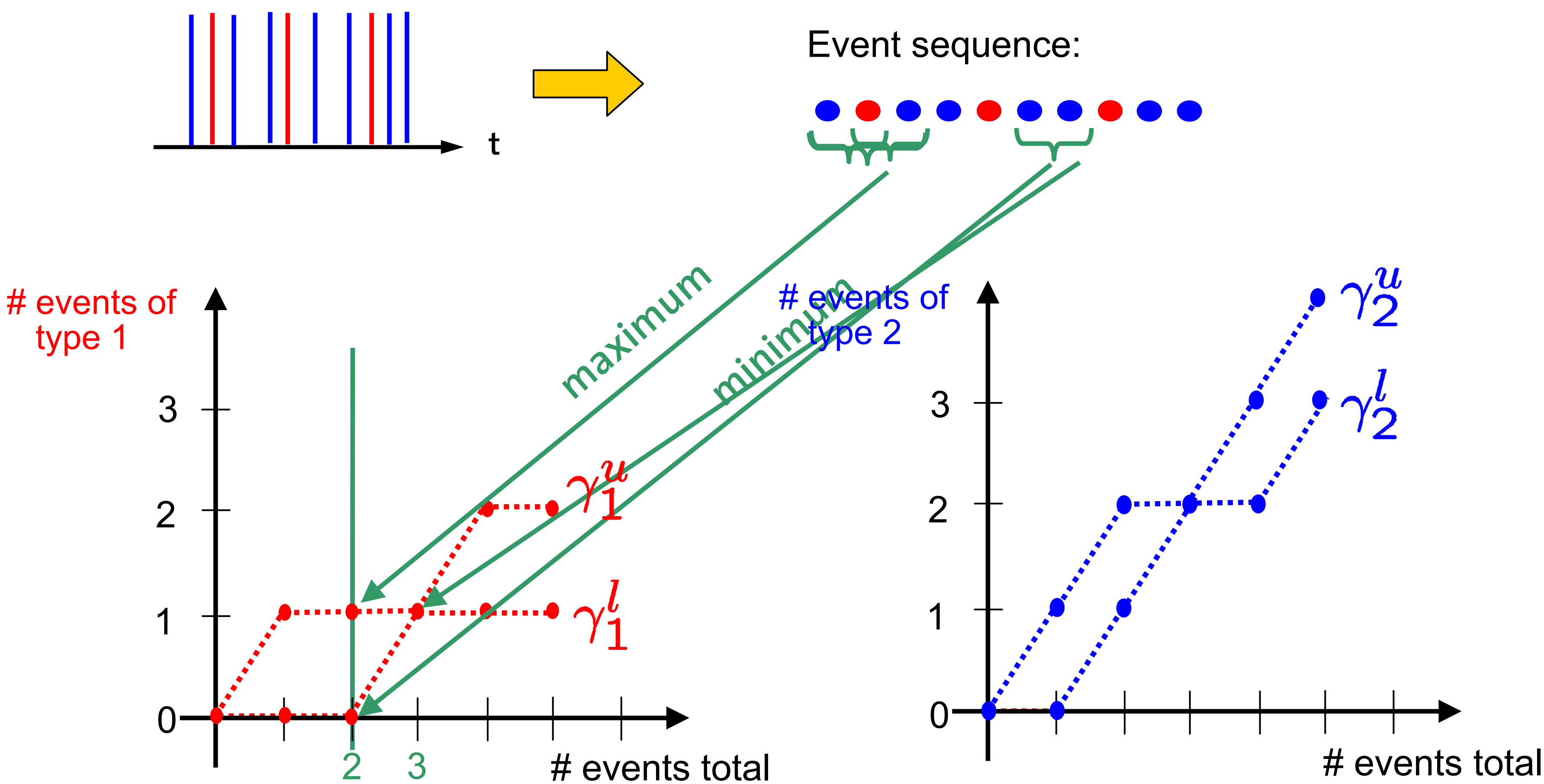
Event Count Curves: Basic Idea

The processing components in MPA preserve the order of the events in a stream

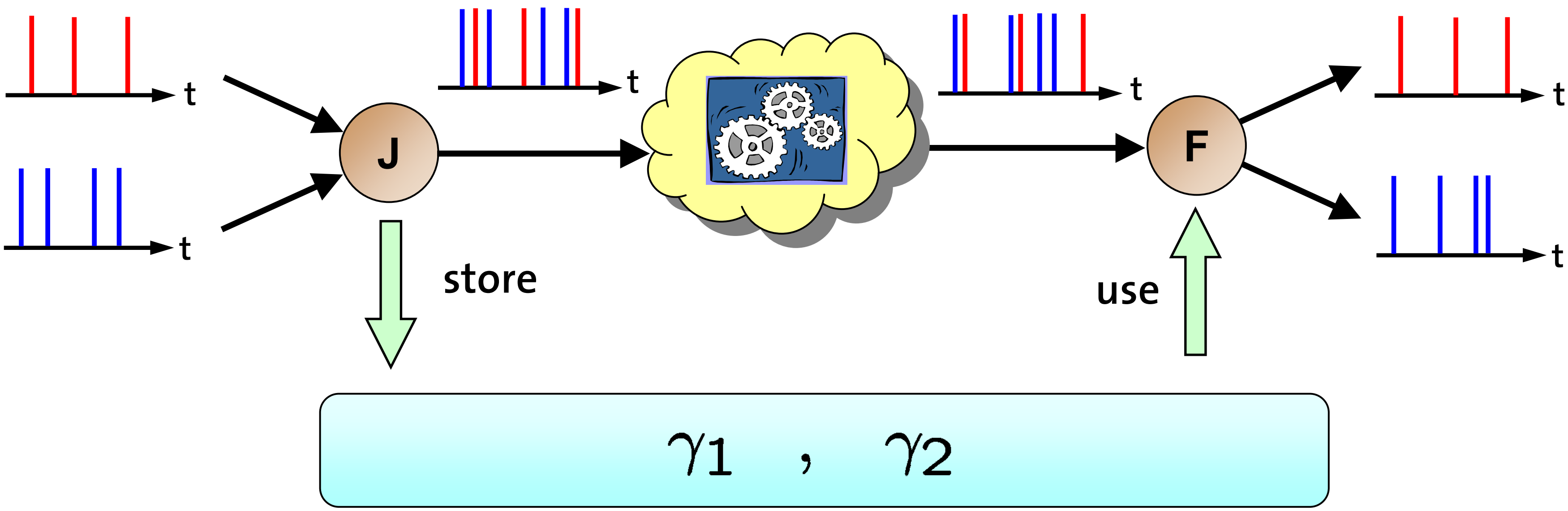


Event Count Curves: Definition

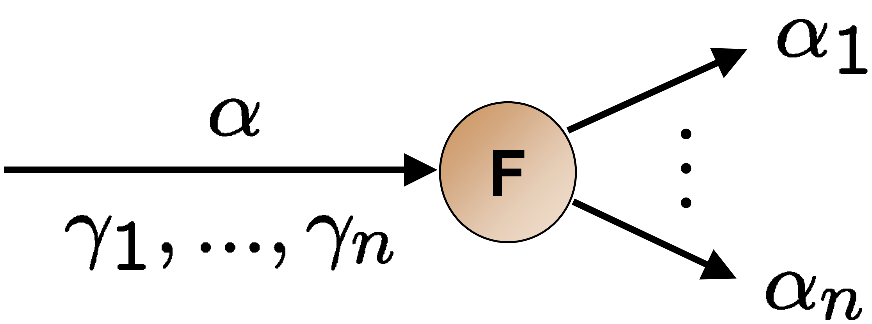
For a fixed number of events in the structured stream the lower (upper) ECC γ_i^l (γ_i^u) bounds the min (max) number of events of type i.



Event Count Curves

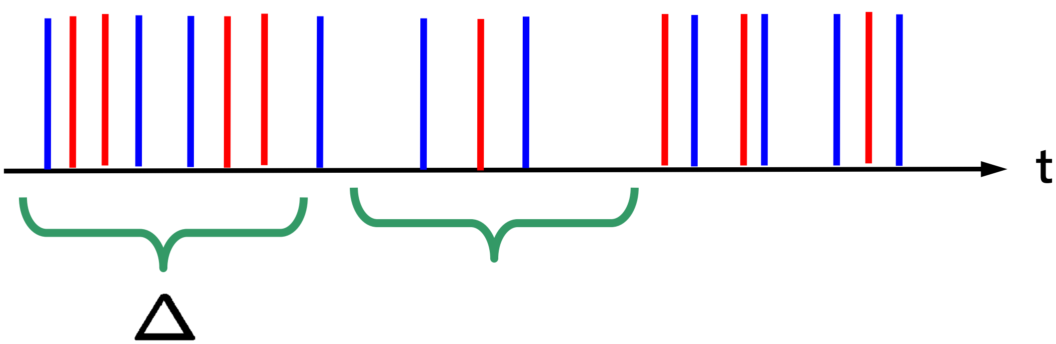


Fork (Simple Event Streams)



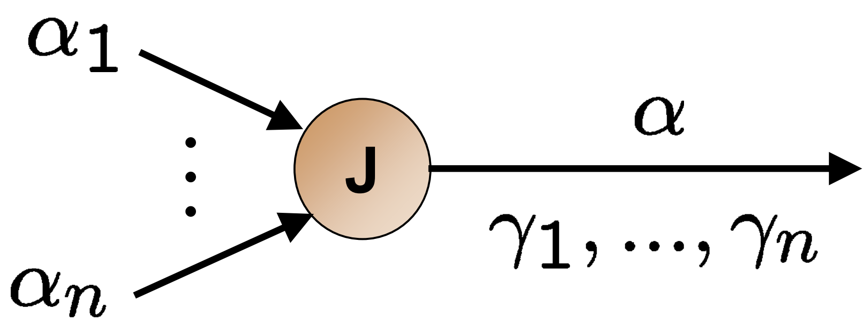
$$\alpha_i^l(\Delta) = \gamma_i^l(\alpha^l(\Delta))$$

$$\alpha_i^u(\Delta) = \gamma_i^u(\alpha^u(\Delta))$$



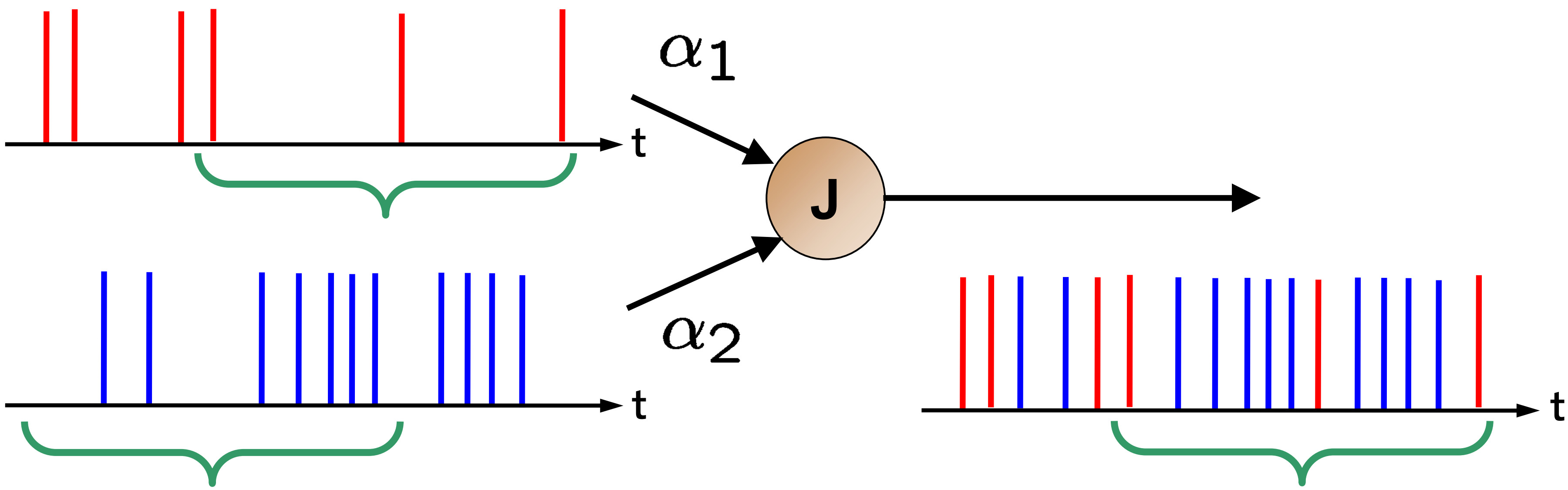
$$\gamma_1^l(\alpha^l(\Delta)) = 1$$

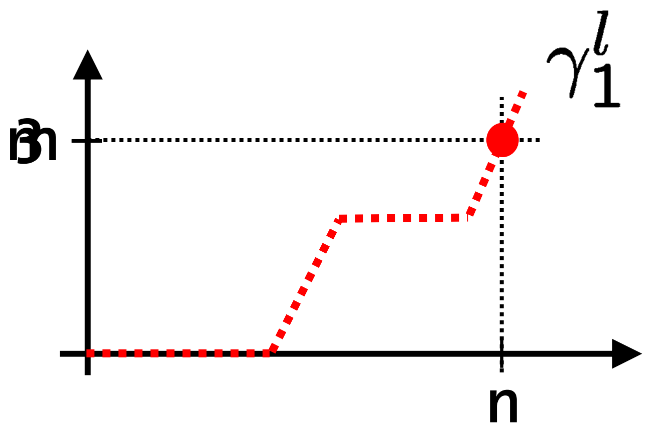
Join (Simple Event Streams)



$$\alpha^l(\Delta) = \sum_i \alpha_i^l \quad ; \quad \alpha^u(\Delta) = \sum_i \alpha_i^u$$

Join (Simple Event Streams)

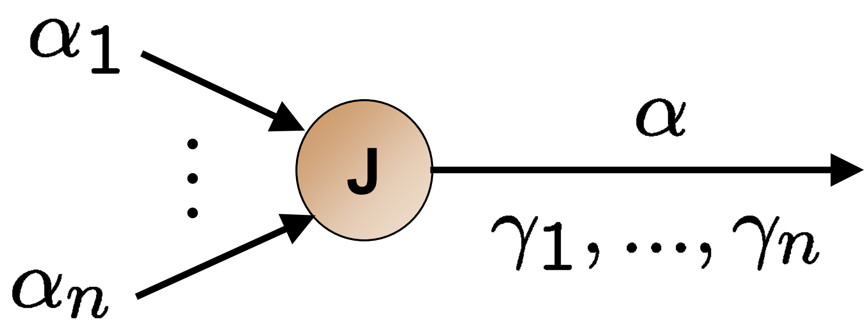




$$n = 3 + \alpha_2^u(\alpha_1^{-l}(3)) = 12$$

$$\gamma_1^l(12) = 3$$

Join (Simple Event Streams)

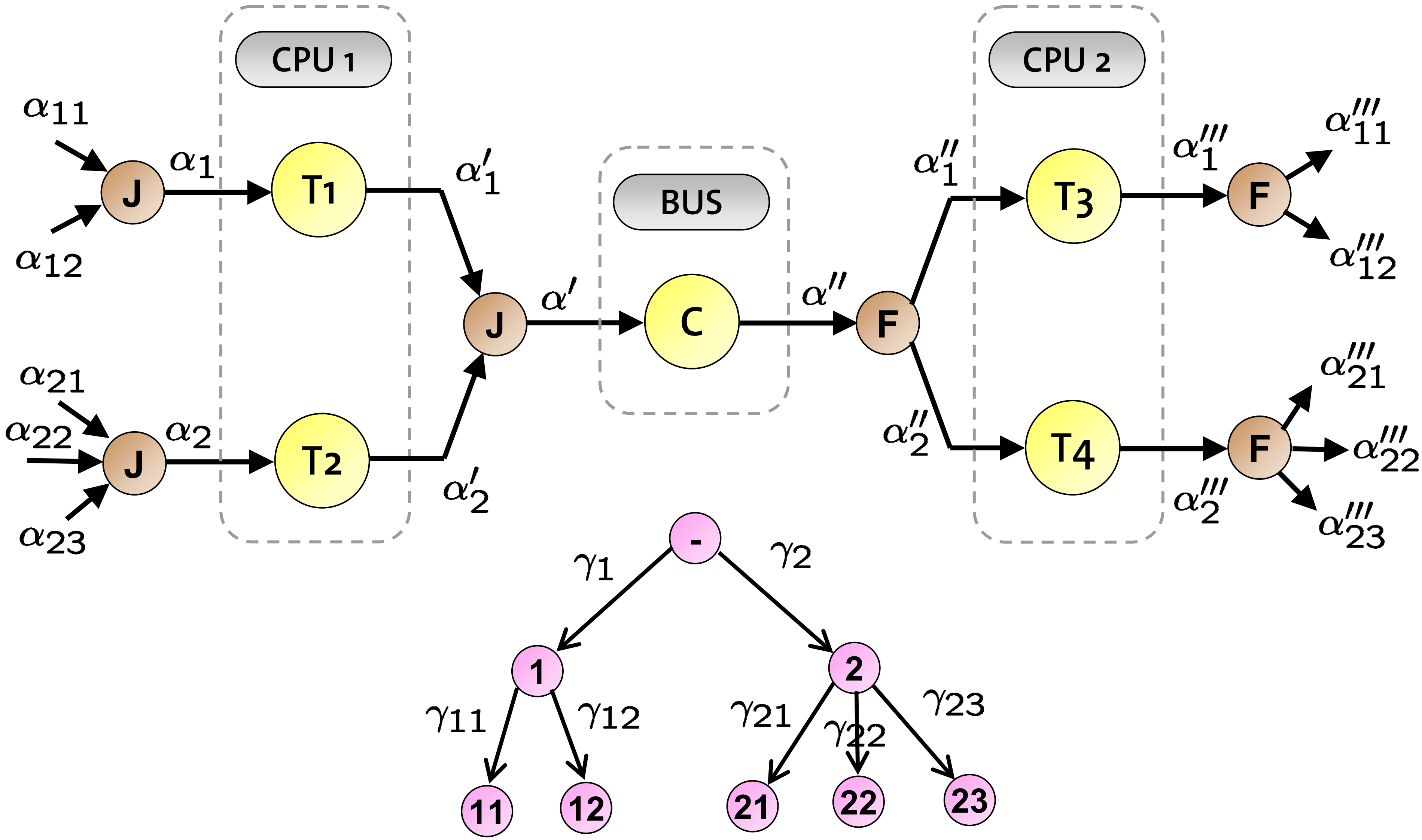


$$\alpha^l(\Delta) = \sum_i \alpha_i^l \quad ; \quad \alpha^u(\Delta) = \sum_i \alpha_i^u$$

$$\gamma_i^l(n) = \epsilon_i^{-u}(n) \qquad \epsilon_i^u(n_i) = n_i + \sum_{j \neq i} \alpha_j^u(\alpha_i^{-l}(n_i))$$

$$\gamma_i^u(n) = \epsilon_i^{-l}(n) \qquad \epsilon_i^l(n_i) = n_i + \sum_{j \neq i} \alpha_j^l(\alpha_i^{-u}(n_i))$$

Hierarchical application of ECCs

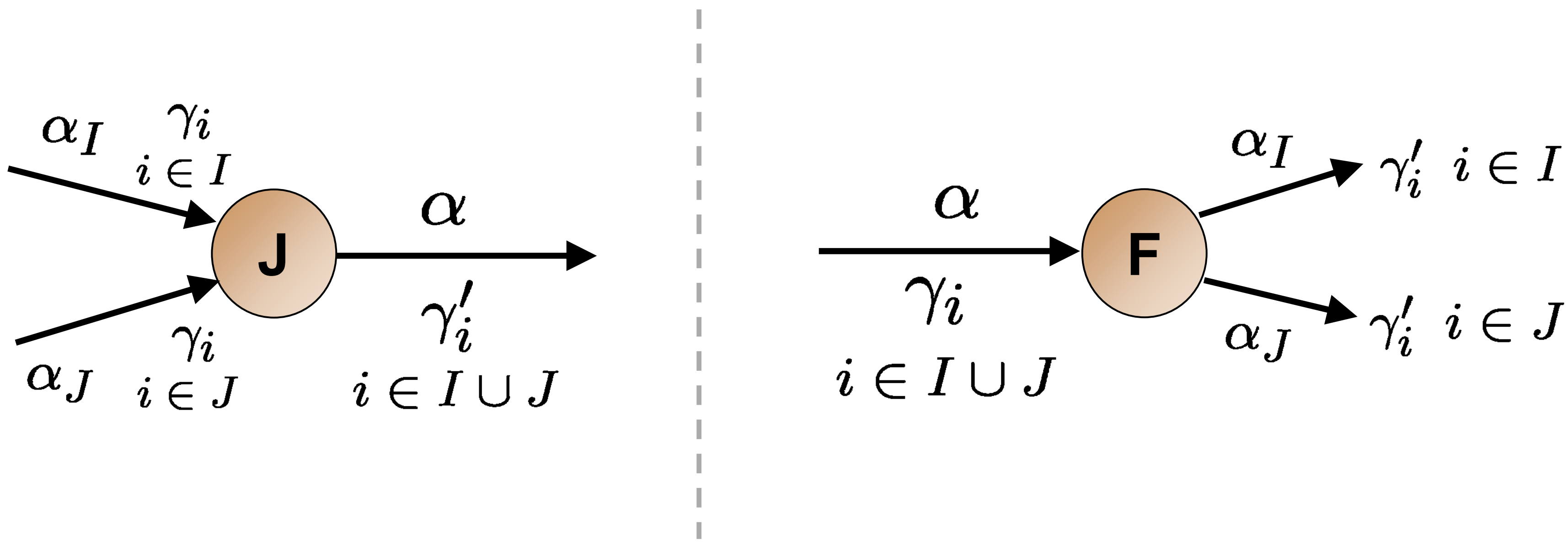


Arbitrary Join and Fork of Structured Streams

Limitation of hierarchical ECC organization:

Structured streams can be decomposed only in the way they have been composed!

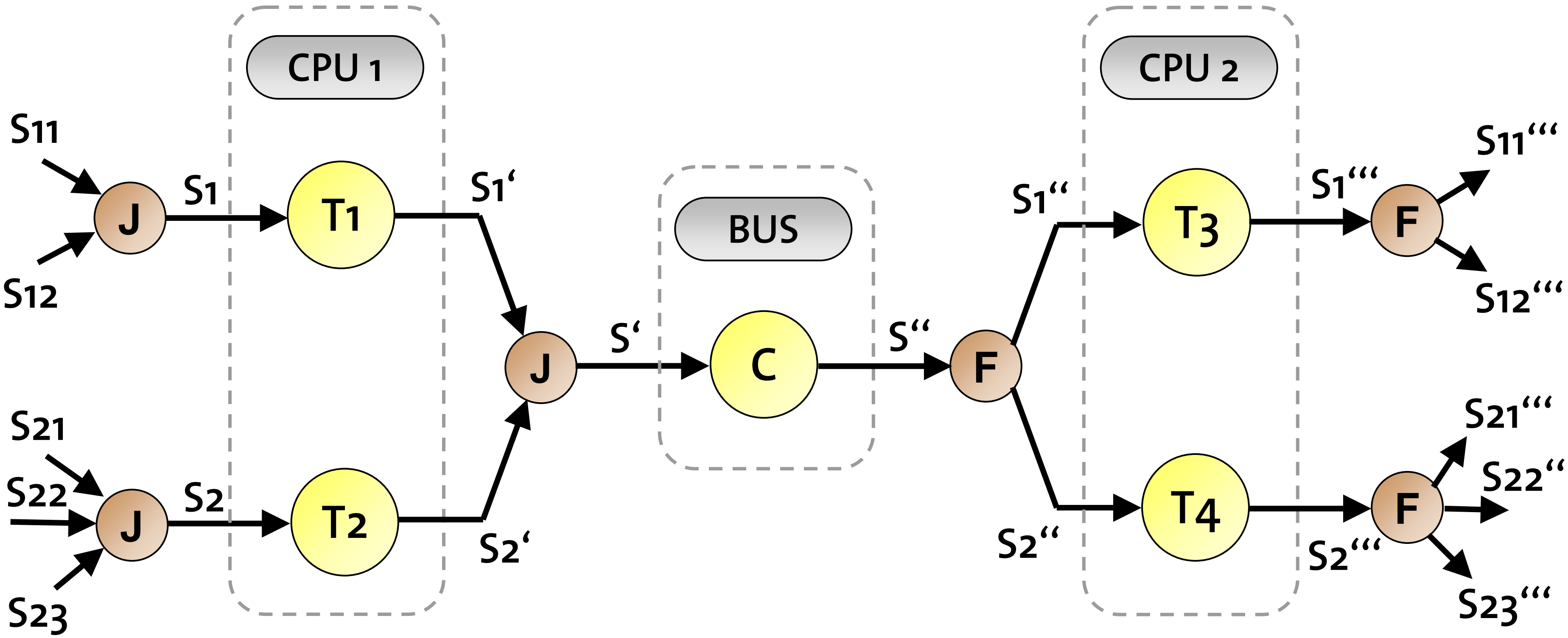
⇒ More general join/fork operators are desirable



Outline

- Motivation
- Modular Performance Analysis with Real-Time Calculus
- Approach 1: FIFO Scheduling
- Approach 2: Event Count Curves
- Comparison / Case Study
- Conclusion

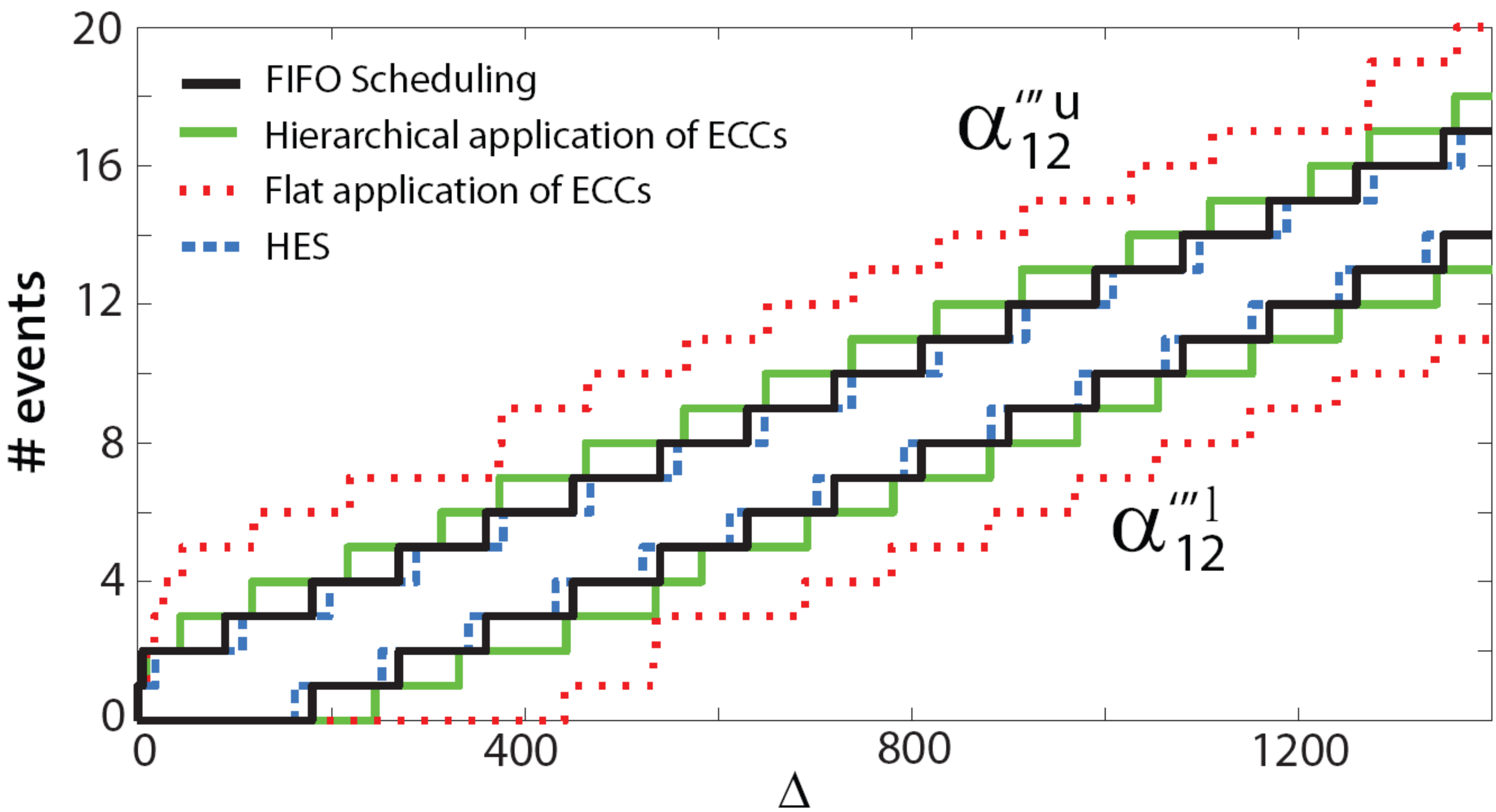
Example



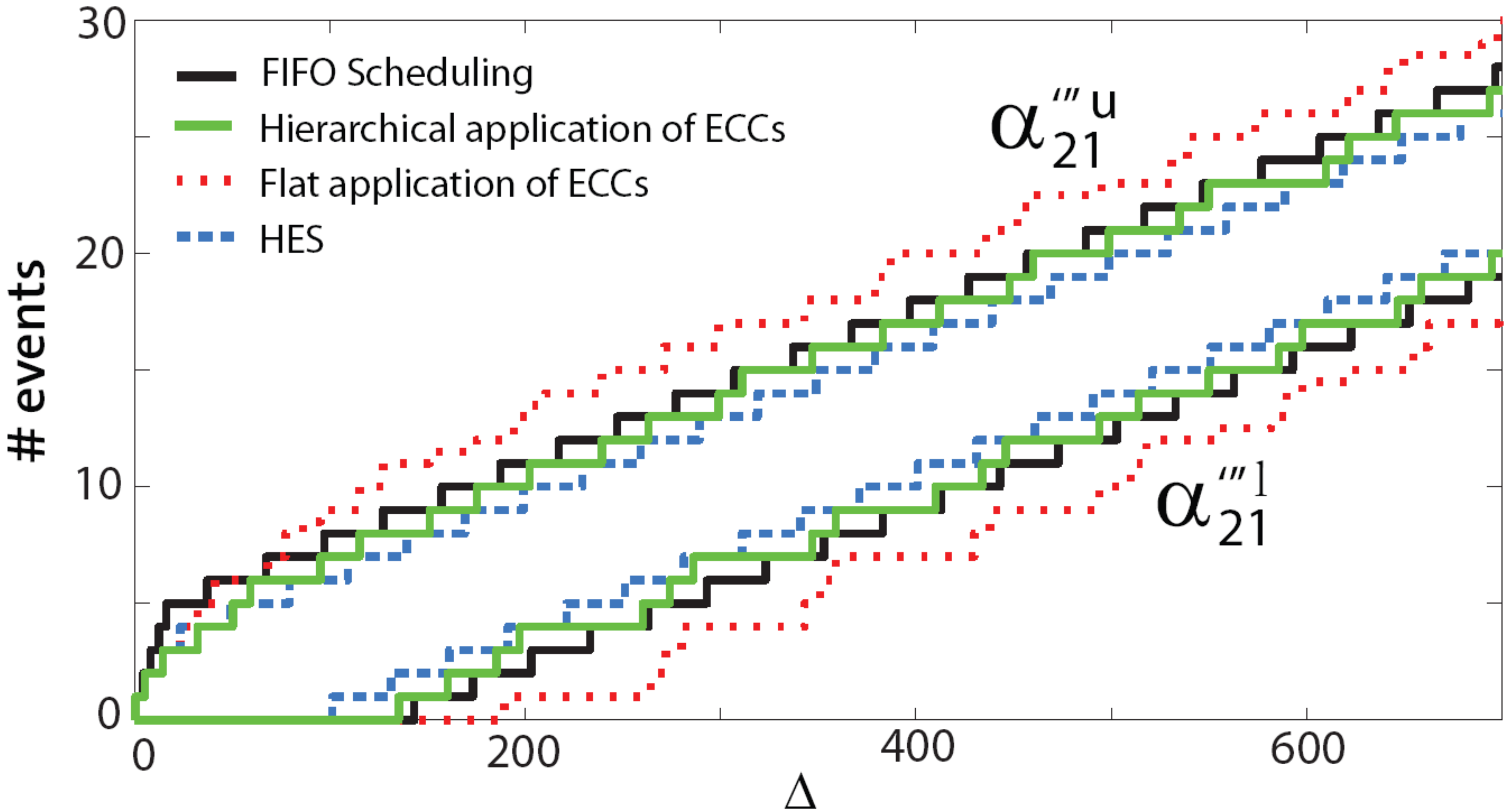
Stream	Period	Jitter
S11	100	30
S12	90	15
S21	30	0
S22	80	20
S23	75	5

Task	BCET	WCET
T1	2	3
T2	3	4
C	9	12
T3	2	3
T4	4	5

Analysis Results

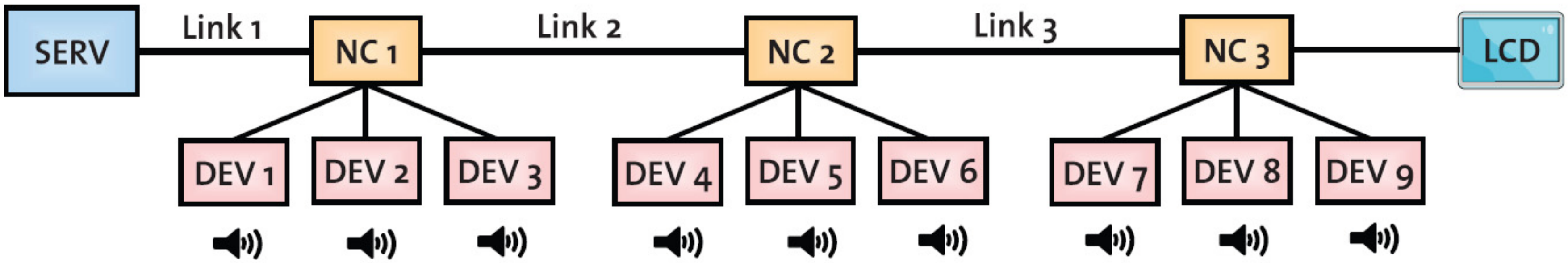


Analysis Results



Case Study: Specification

Distributed information/entertainment application

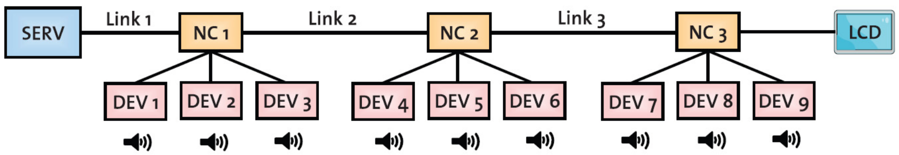


2 classes of network traffic:

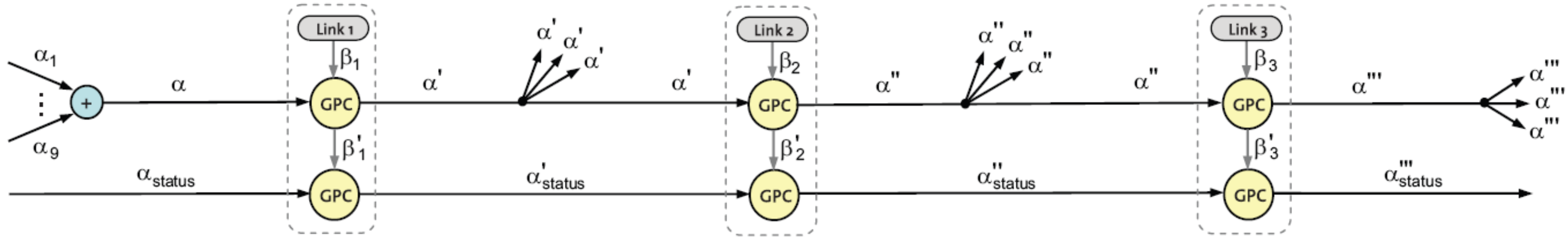
- On-demand audio streaming (high prior.)
- Real-time status information (low prior.)

	Size	Period	Jitter	Deadline
Audio Frames	1'518 Bytes	30 ms	5 ms	100 ms
Status Frames	106'500 Bytes	5 s	0 s	1.5 s

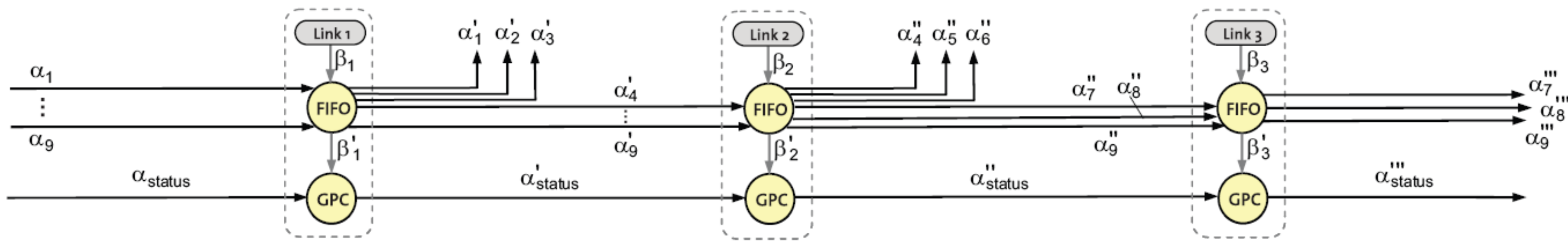
Case Study: Models (1)



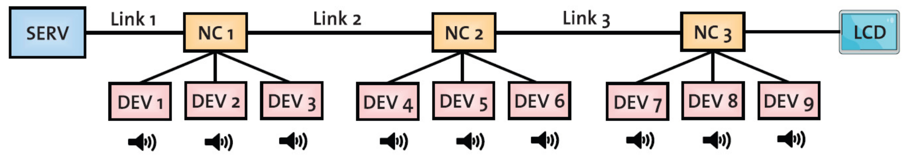
Classic MPA



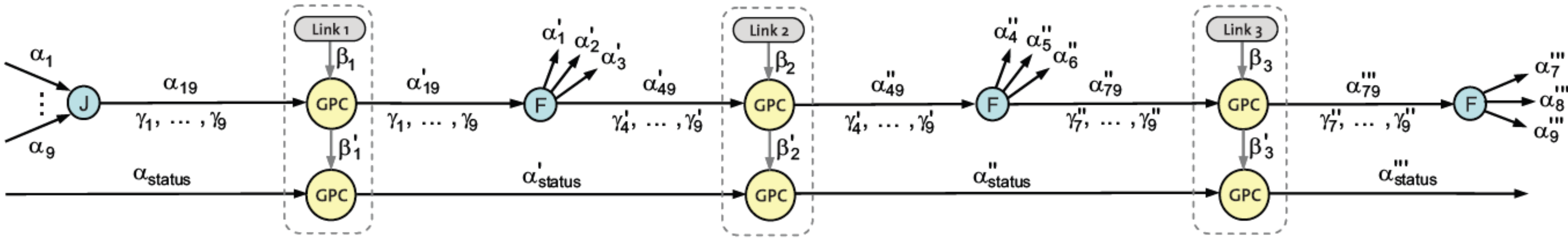
FIFO components



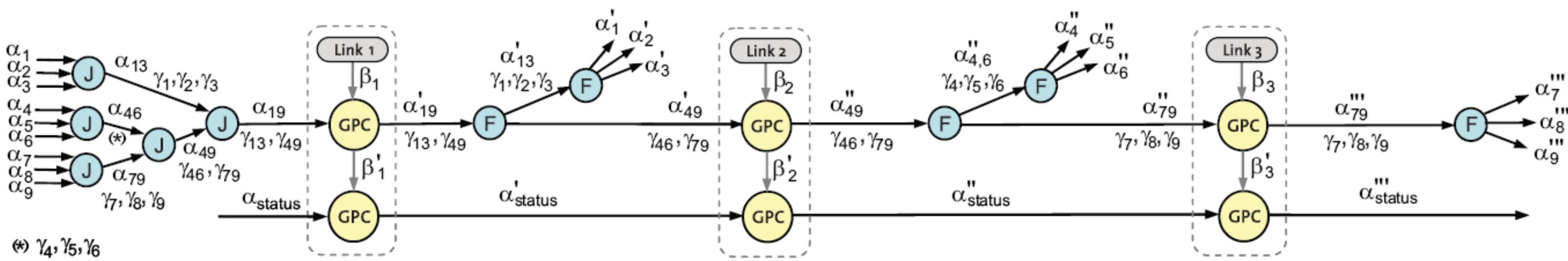
Case Study: Models (2)



ECC (flat)

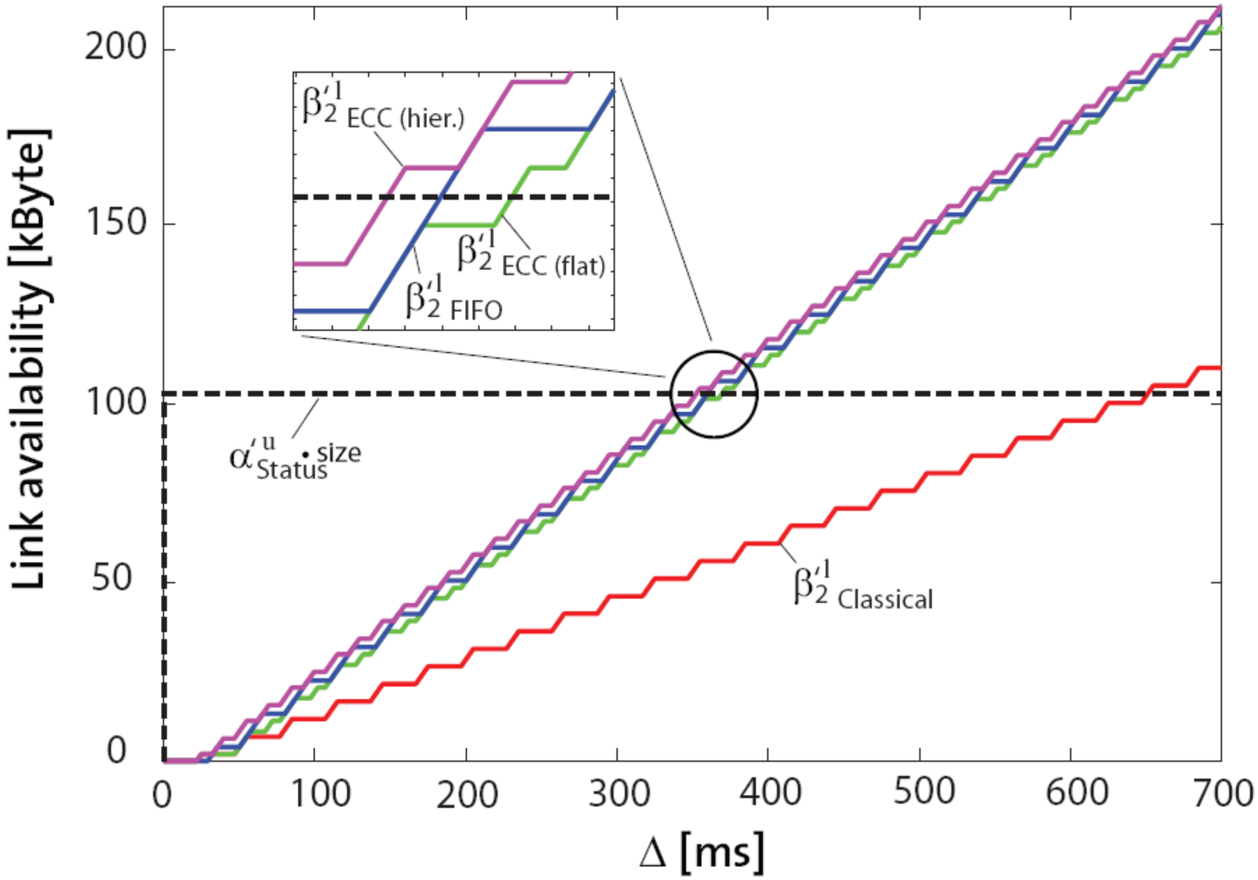


ECC (hierarchic)



Case Study: Results

	Classic	FIFO	ECC (flat)	ECC (hierar.)
Max. delay	1.954 s	1.255 s	1.316 s	1.248 s



Outline

- Motivation
- Modular Performance Analysis with Real-Time Calculus
- Approach 1: FIFO Scheduling
- Approach 2: Event Count Curves
- Comparison / Case Study
- Conclusion

Conclusion

- Two new methods for modeling and analysis of joined event streams in modular performance analysis

- The concept of Event Count Curves is orthogonal to existing event stream models (PJD, arrival curves) and transparent to all analysis components $\Rightarrow$ Compositionality of methods is not affected

- ECC operators for structured streams allow *arbitrary* decomposition of streams

- Comparisons and Case Study highlight the utility of the proposed methods

Thank you!

Simon Perathoner

perathoner@tik.ee.ethz.ch