

DAQ Data Flow Simulations

Experience from ATLAS and ALICE

DAQ@LHC

13 April 2016

**Matías Bonaventura^{1,2}, Daniel Foguelman², Rodrigo D. Castro²,
Mikel Eukeni Pozo Astigarraga¹, Iosif Legrand^{1,3}**

¹CERN

²Computer Science Department, University of Buenos Aires, Argentina

³California Institute of Technology

Outline

- ❖ **Main Objectives for the simulation activities**
- ❖ **Simulation methodologies and tools for describing complex DAQ systems**
 - ❖ **The DEVS formalism and PowerDEVS tool used by ATLAS**
 - ❖ **OMNeT++ & The MONARC simulation system used by ALICE**
- ❖ **ATLAS examples**
- ❖ **ALICE examples**
- ❖ **Using simulation as a design and optimization tool**

The main goals for the Simulation Activities

- To perform realistic simulation and modelling of large scale distributed computing systems
- To offer a dynamic and flexible simulation environment to be used as a design tool capable to evaluate different technologies, network topologies and to help in optimizing the flow algorithms.
- To provide a design framework to evaluate the performance of a range of possible computer and networking systems, as measured by their ability to provide the experiments with the requested data rate for the on-line system, and to optimise the cost

Design Considerations for the Modelling System

- The Simulation framework should not intend to be a detailed simulator for all basic components such as servers, switches or operating systems.
- Instead, based on realistic **mathematical models** and the measured **statistical distribution** for the systems parameters for all the basic components, it aims to correctly describe the performance and scalability of large scale data intensive computing systems with complex interactions and real-time constrains.

The Main Types of Simulations

- **Continuous time**
- **Discrete time**
- **Mixed mode**

Discrete Time Simulations

- **Predefined fix periodic time window**
- **Discrete Event Based → queue for Events**

Modelling application programs:

- **State Machine with transition rules**
- **Threads like having a program counter and states**

Simulation Tools:

DES: **PowerDEVS, OMNeT++, NS2/3, Matlab/Simulink..**

Actors : **Ptolemy, Scala, MONARC simulation ...**

Possible Options to Simulate Large Computing Systems that Handle a Lot of Data

We need to simulate a large number of processes that transfer huge amounts of data with real time constraints

- To evaluate different algorithms for data flows, control algorithms, error recovery, etc.
- Evaluate the scalability of the system and estimate the limits

Options for simulating interacting programs:

- State machine
- Petri net (place/transition net or P/T net)
- Simulation threads (actors)

How to simulate the traffic for very large amounts of data:

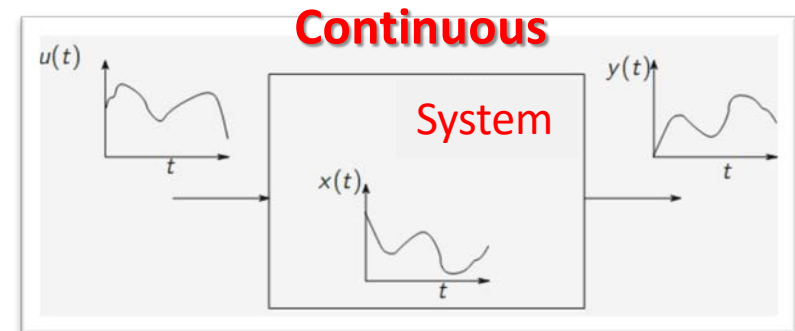
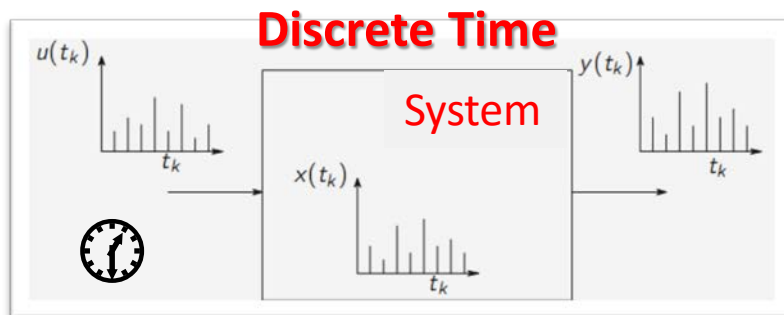
- Packet / frame level simulation is unaffordable for ~Tb/s and ~hours of running
- Continuous flow as long as nothing is changing in the system

The “Challenges ” to Simulate Large, Complex Computing Systems with Real Time Constrains

- It is necessary to abstract from the real system all components and their time dependent interaction.
- **THE MODEL** has to be equivalent to the simulated system in all important respects.
- For large, complex systems it is really important to distinguish between **components** and **interactions** that
 - need detailed implementation in the simulation framework
 - or, can be modeled based on their statistical properties
- The simulation time should remain relatively small if we intent to use the simulation framework for system design and optimization.

ATLAS: DEVS for Hybrid Systems

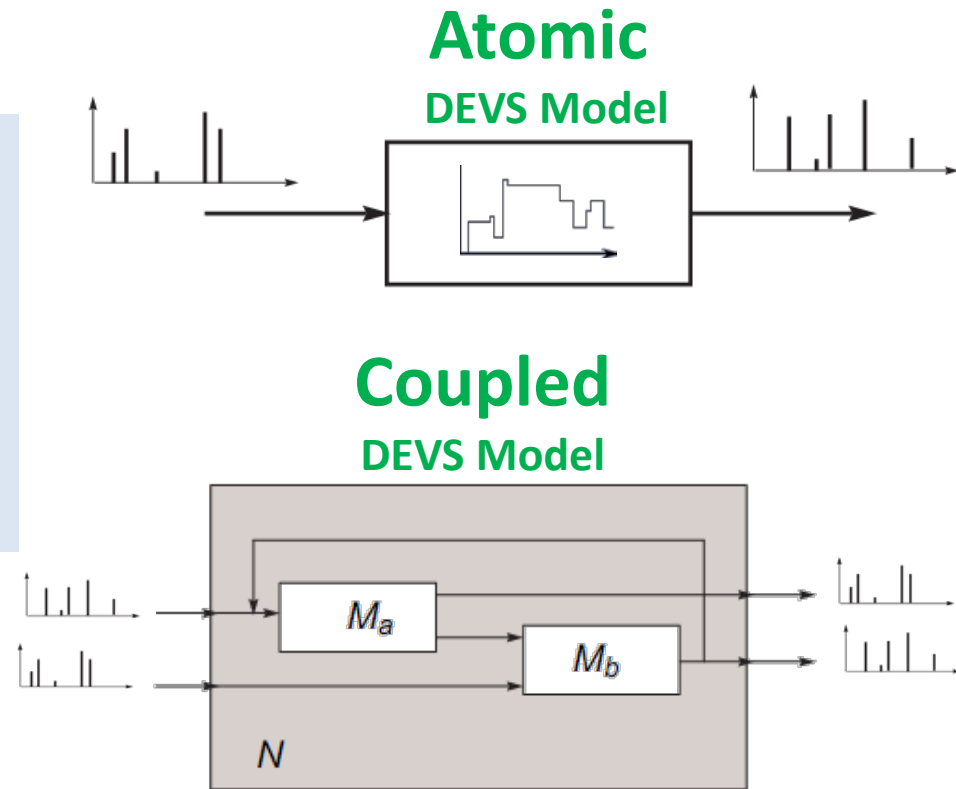
- Discrete **EV**ent **S**ystems specification (*Bernard Zeigler, '76, '90, 2000*)
- Based on principles of the General Systems Theory
- **DEVS** allows to:
 - **Represent exactly** any type of **Discrete System (clocked, un-clocked)**
 - **Approximate** Continuous Systems at any desired degree of accuracy
 - Mix different representations **within a same model**
 - **choose freeley the level of abstraction !**



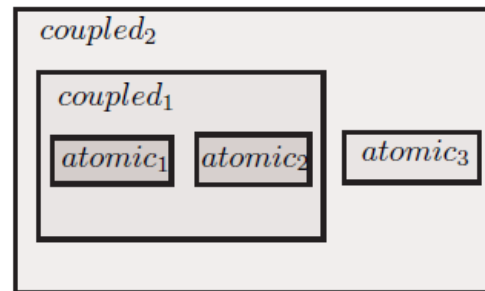
DEVS Models

A DEVS Atomic Model:

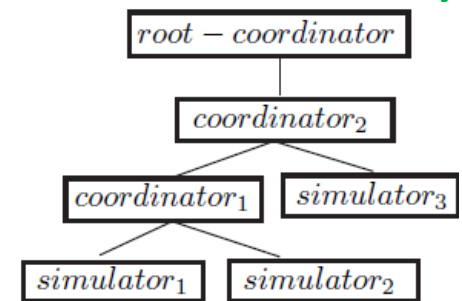
- **Processes** a sequence of **Input Events**
- According to its **Inputs** and **Internal States**
- **Produces** a sequence of **Output Events**
- On a **continuous time base**
- Allows representing **any system** undergoing a **finite number of changes** within **finite time intervals**.
- Can be **coupled modularly and hierarchically** to build **complex systems**.
- **DEVS Models** are **strictly separated** from the underlying **DEVS simulation algorithms**
- **No Global Event List !**



Models hierarchy

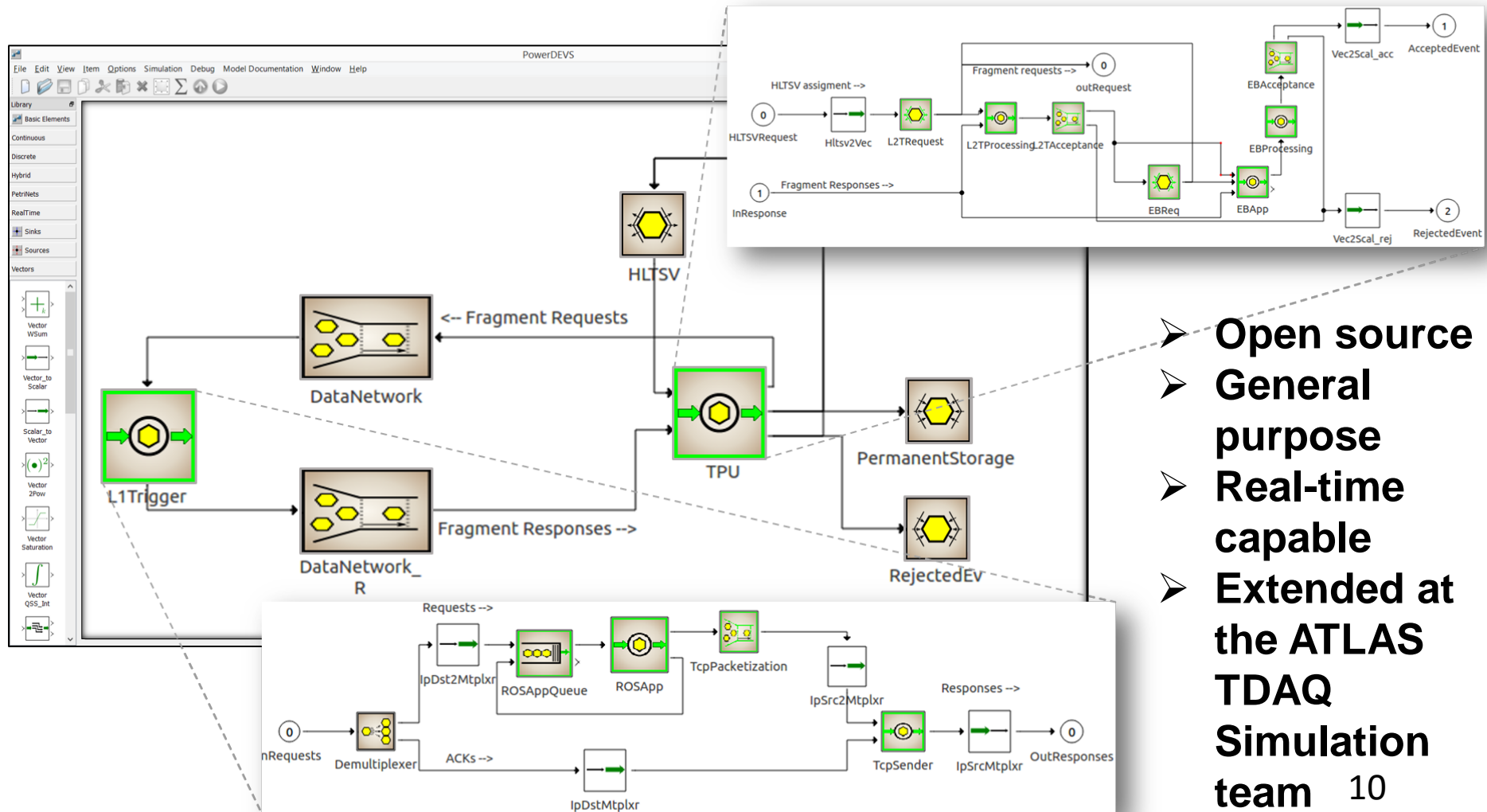


Simulators hierarchy



Tool: PowerDEVS

- Visual GUI for defining DEVS models coupling (block oriented)
- C++ for defining each DEVS model's dynamic functions



- Open source
- General purpose
- Real-time capable
- Extended at the ATLAS TDAQ Simulation team 10

Process Oriented Simulation

The MONARC Simulation Model

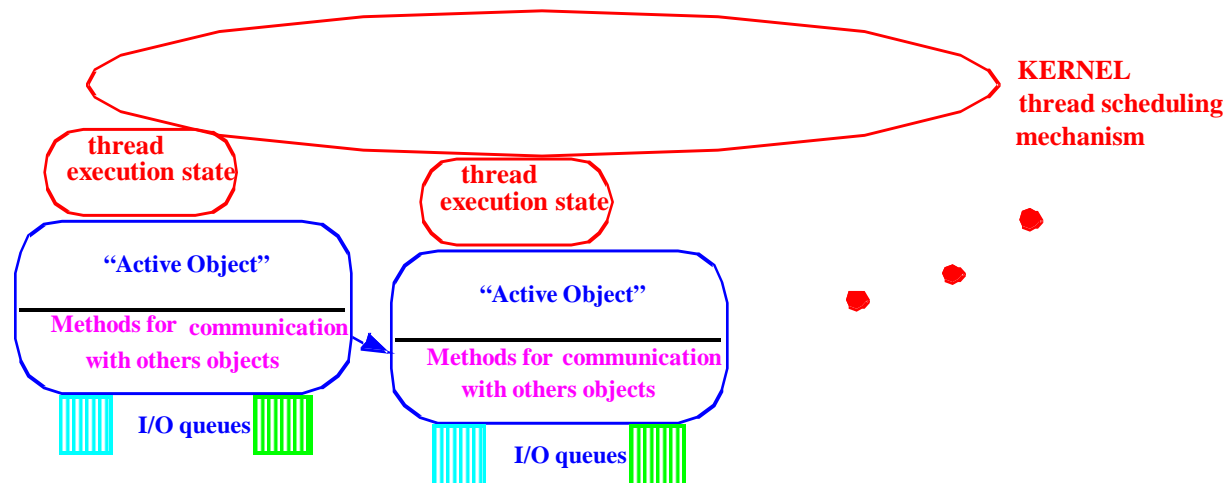
Process oriented DES Based on “ACTIVE OBJECTS” or “ACTORS”

Thread: execution of a piece of code that occurs independently of and possibly concurrently with another one

Execution state: set of state information needed to allow concurrent execution

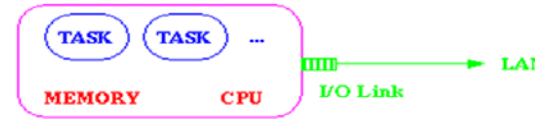
Mutual exclusion: mechanism that allows an action to be performed on an object without interruption

Asynchronous interaction: signals / semaphores for interrupts



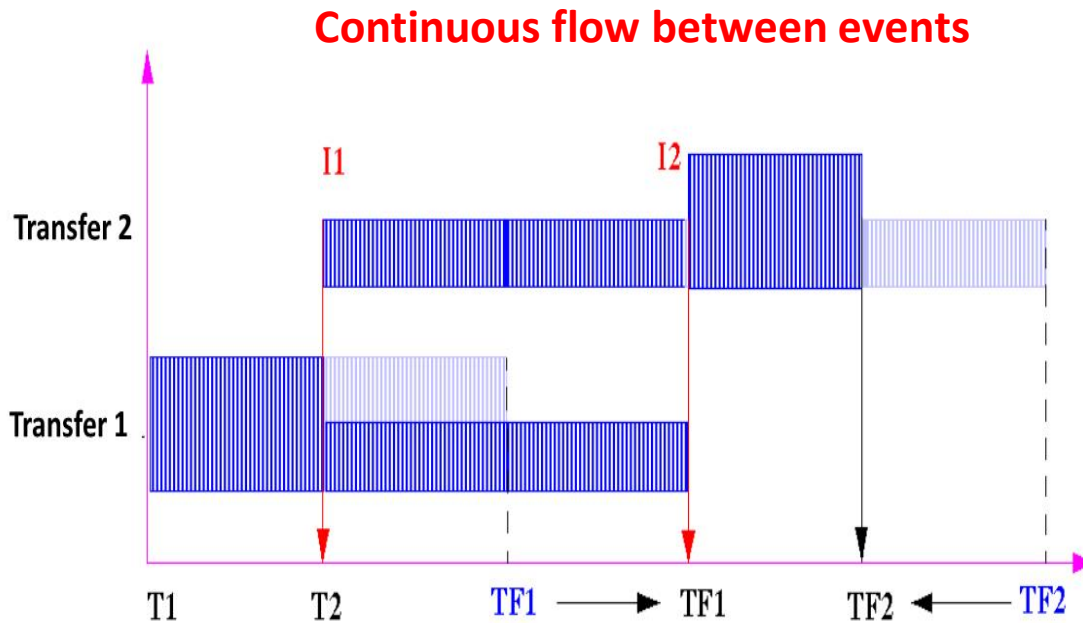
Process Oriented Simulation : Data Transfer and Multitasking Processing Models

Concurrent running tasks (or data transfer jobs) share resources (CPU, memory, I/O links)



“Interrupt” driven scheme:

For each new task or when one task is finished, an interrupt is generated and all “processing times” are recomputed.



It provides:

An efficient mechanism to simulate multitask processing and **sustained flows of events**

Handling of concurrent jobs with different priorities.

An easy way to apply different load balancing schemes.

Monarc Model

Input Parameters for the Simulation Program

- **Correctly identify and describe the time response functions for all active components in the system. This should be done based on realistic measurements.**
- **The simulation frame allows one to introduce any time dependent response function for the interacting components.**

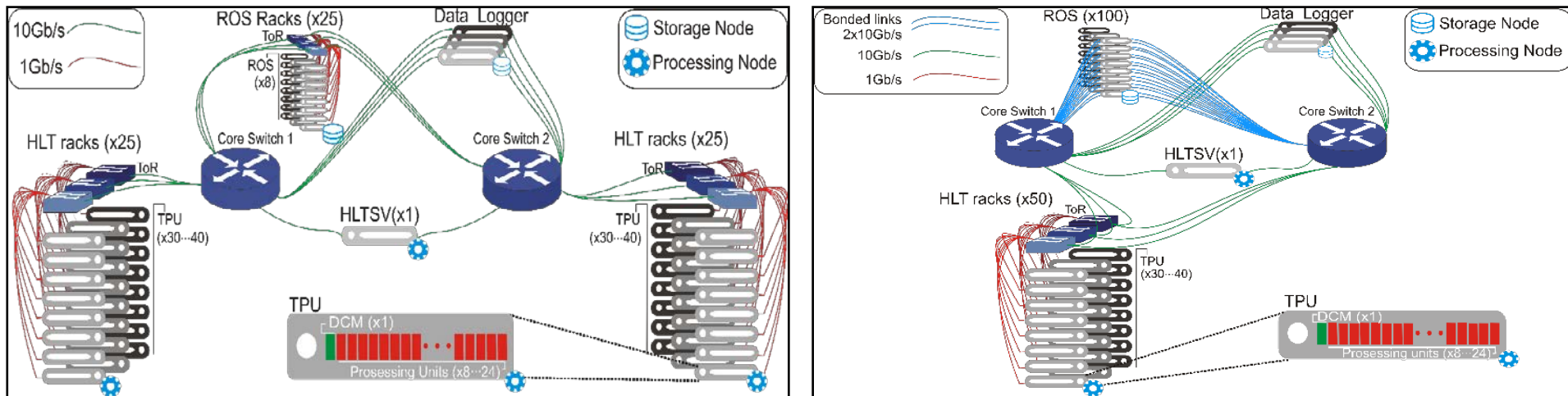
$$S (T_i) = F (S(T_{i-k}) , \{SysP\}, \{ReqP\})$$

- **Response functions are based on “the previous states” of the component, a set of system related parameters (SysP) and parameters for a specific request (ReqP).**
- **This allows to describe correctly **Highly Nonlinear Processes****

ATLAS: Evaluation of candidate network topologies

- Evolution of topologies: from LS1 to Run2
- 200 ROS, 2 Core Switches, 50 ToR switches, ~2000 TPU servers, >30000 applications

ATLAS DAQ network – Simulation of different topologies



LS1 intermediate topology

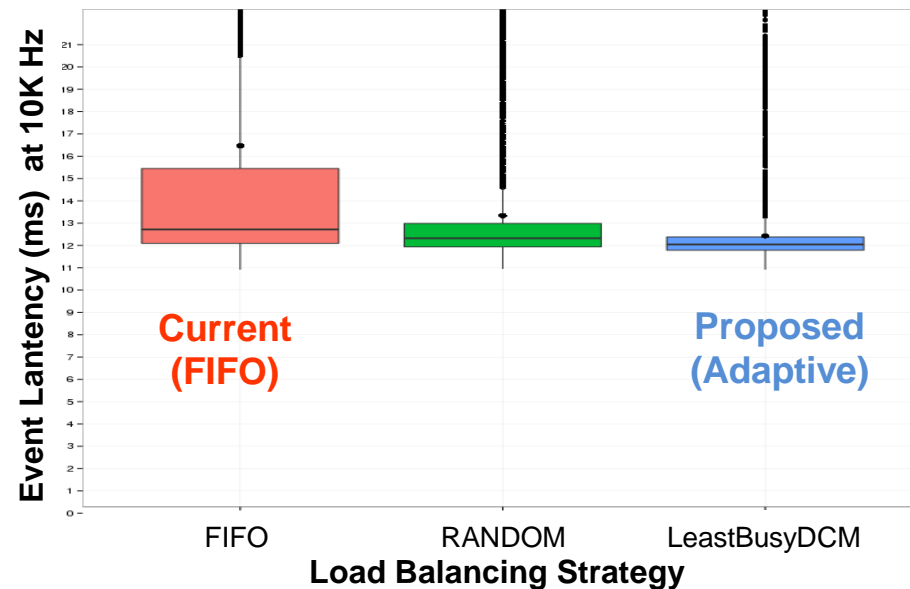
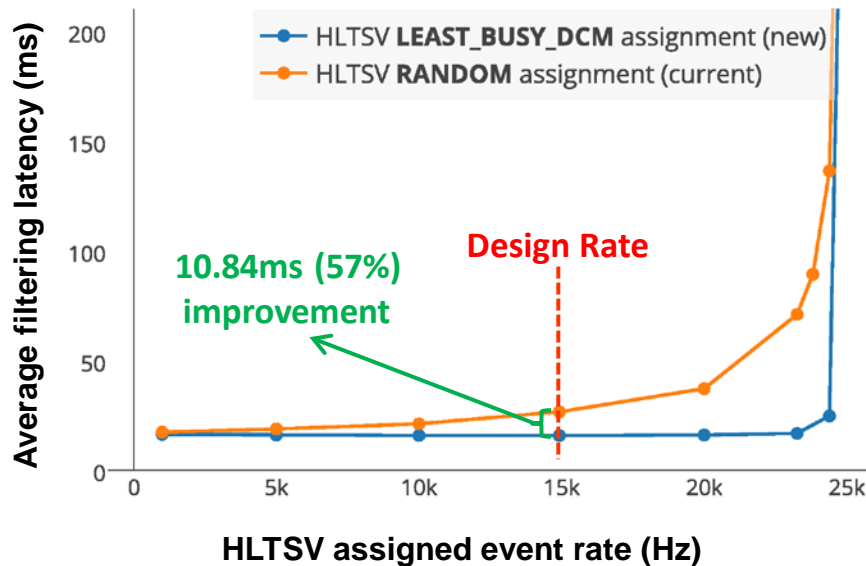


Run2 final topology

Evaluation of candidate network control algorithms

- Quickly evaluate performance implications of different options
- **Once a model is available, data exploration can trigger new ideas**
- E.g.: New enhanced Load Balancing to optimize DAQ Farm usage and Event Latency

Simulation to evaluate different HLTSV load balancing strategies



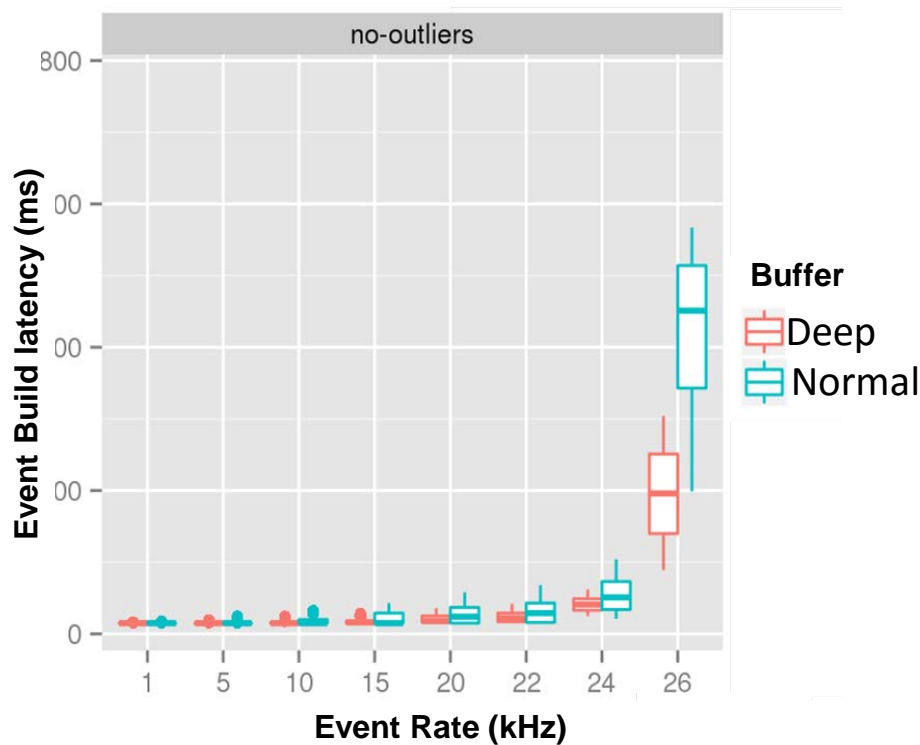
- **Simulation showed** that using an **adaptive load balancing** can **improve average Event Build Latency** as much as **50%**
- Afterwards **validated** on a TBED real infrastructure.

Evaluation of candidate network design decisions

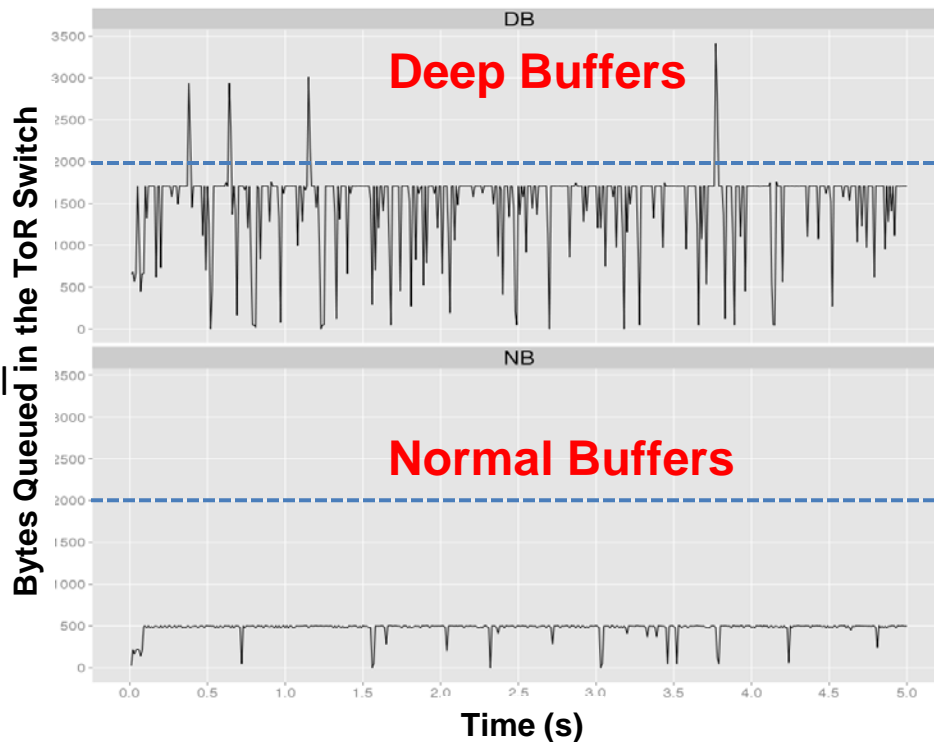
- Early evaluation of different hardware options for the ToR switches
 - Deep Buffers: 10MB shared for all ports
 - Normal buffers: 700Kb per port (39 ports per switch)

Simulation to compare different hardware options

Event Build latency at different HLTSV rates

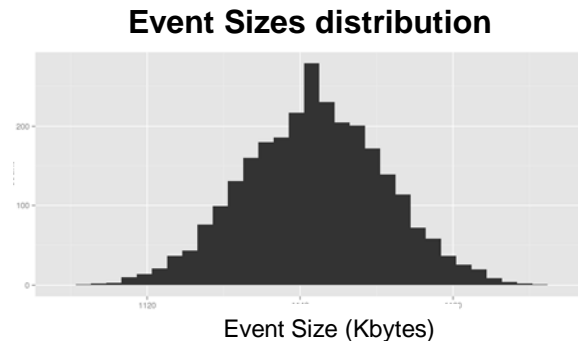
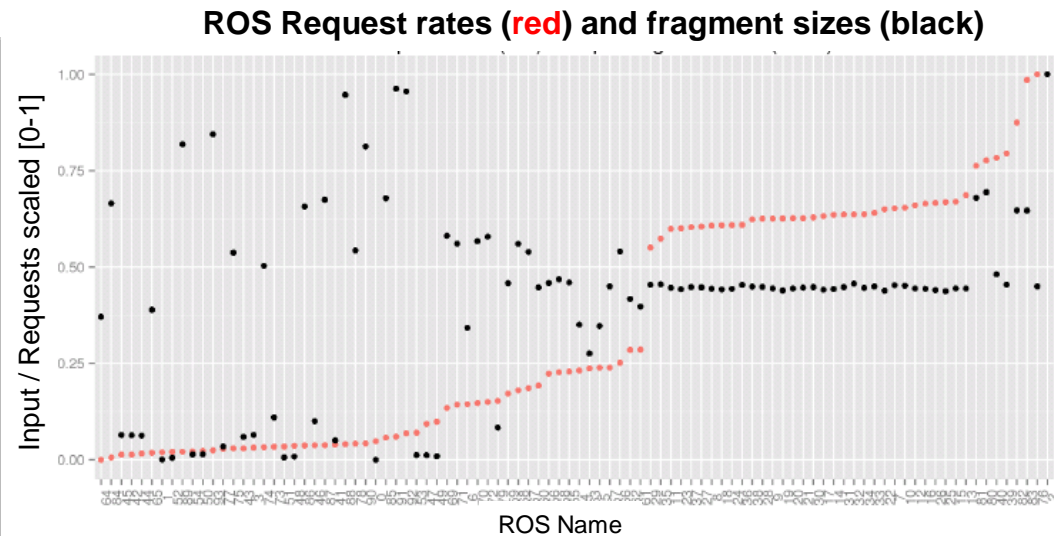
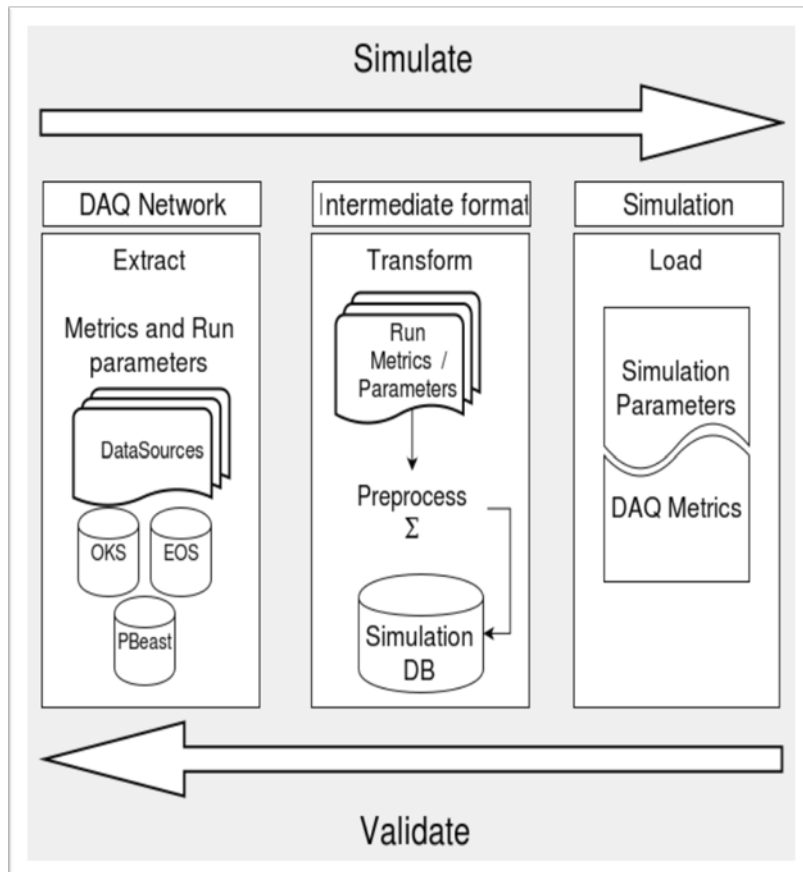


Evolution of buffered data in ToR Switches



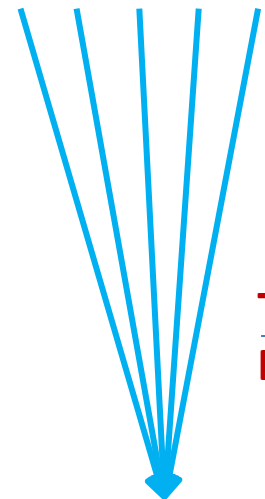
Automate the Simulation and Validation cycle

- Enhance system reproducibility capabilities
- Simulations **parametrized with realistic measured distributions**
- **Tools** to extract measurements (PBeast, IS, etc.) and configure simulation



ALICE O2: SEPN Topology using Infiniband or 100 Gbps Ethernet for Time Frame Collection

Concurrent
Fan In



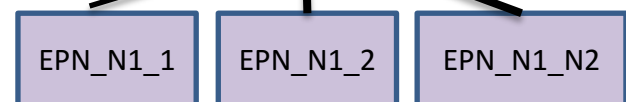
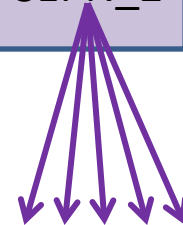
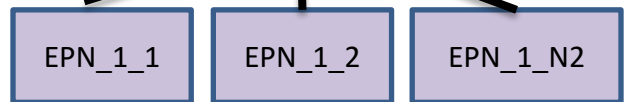
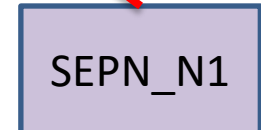
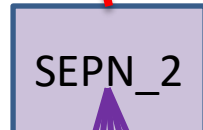
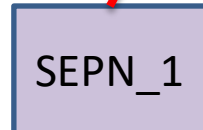
CIn

Tens of Tbps
Bisection Traffic

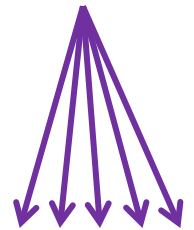


Reduce the
number of high
speed links
connected to the
switching fabric

COut

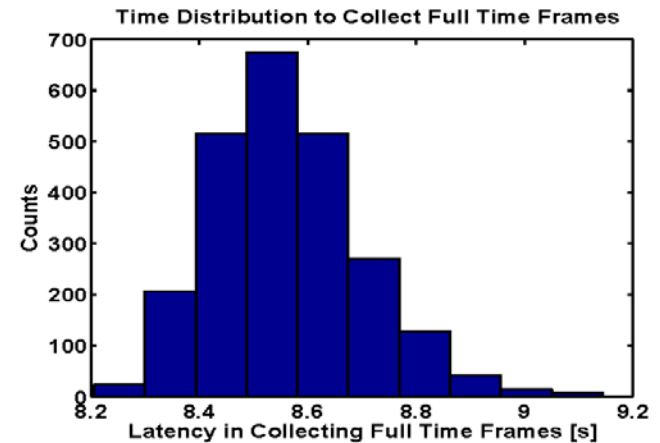
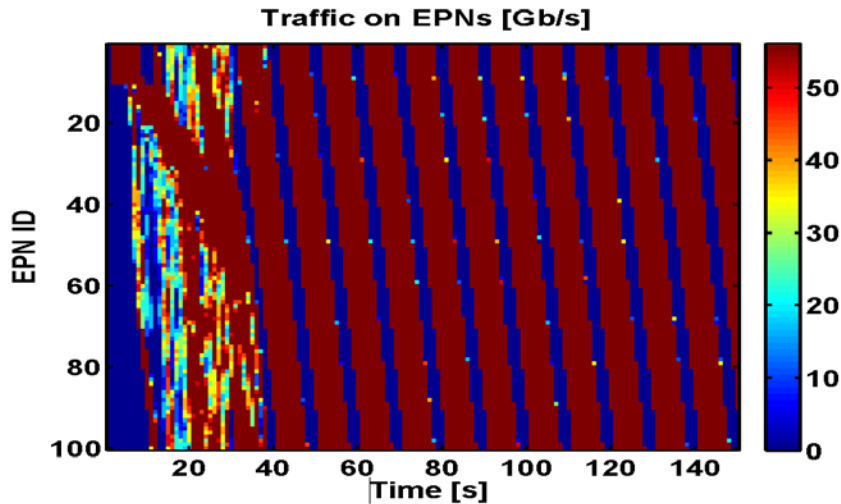
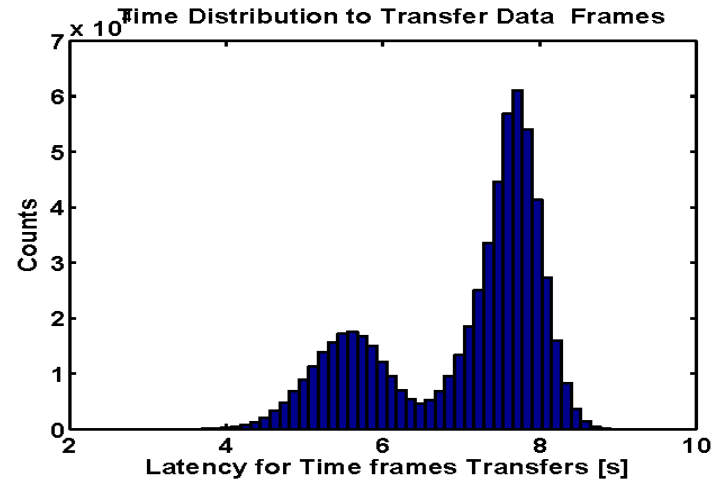
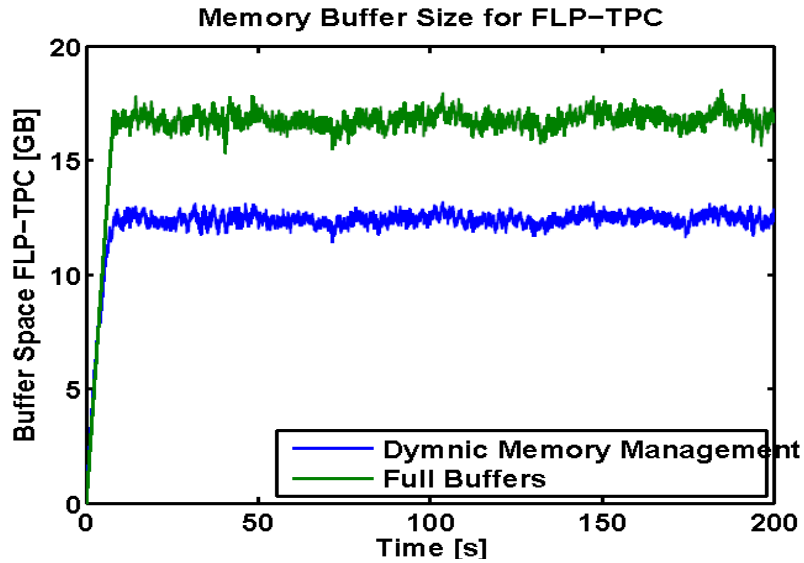


Distributed
Fan Out

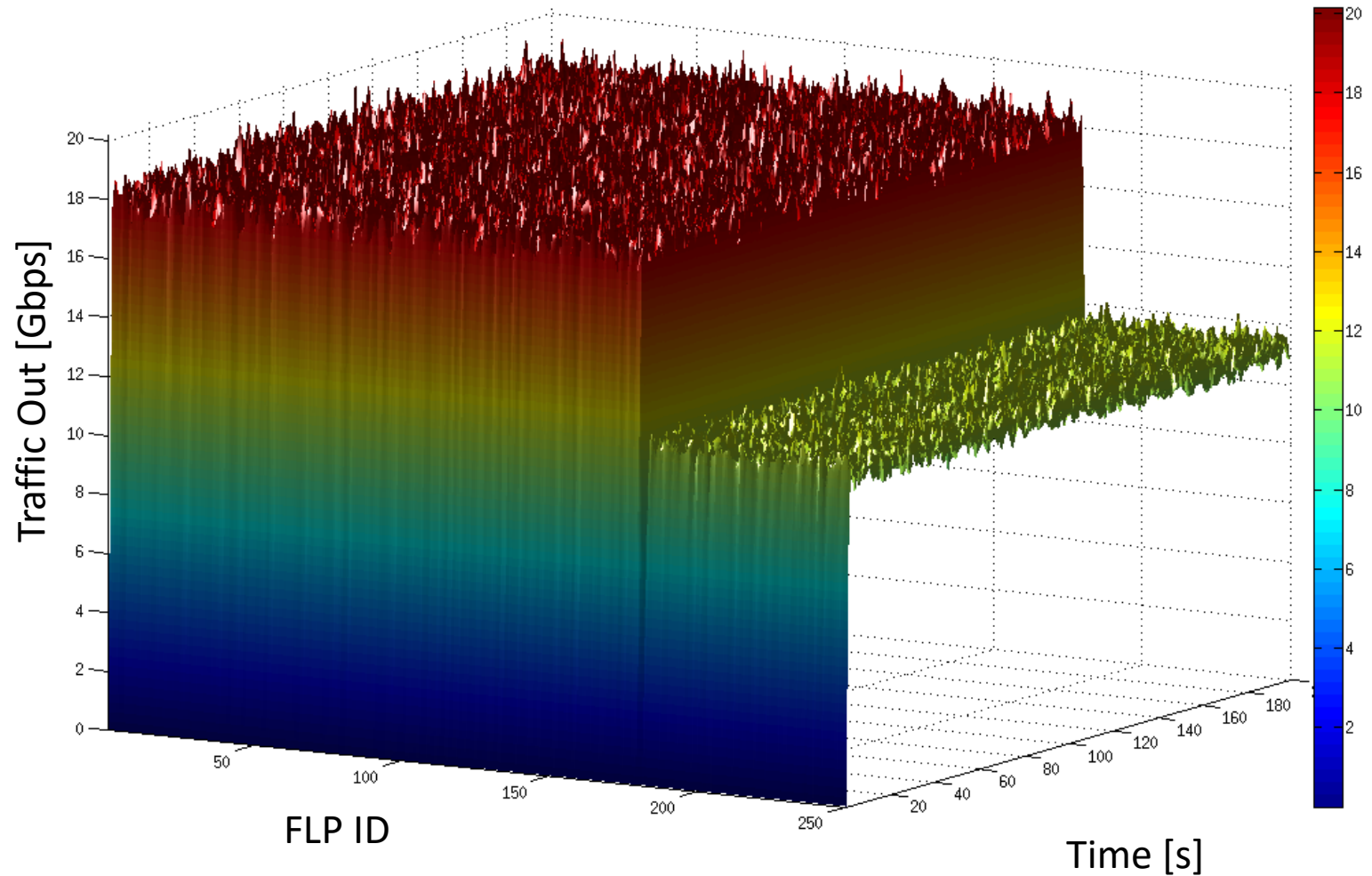


~ 1500 EPNs

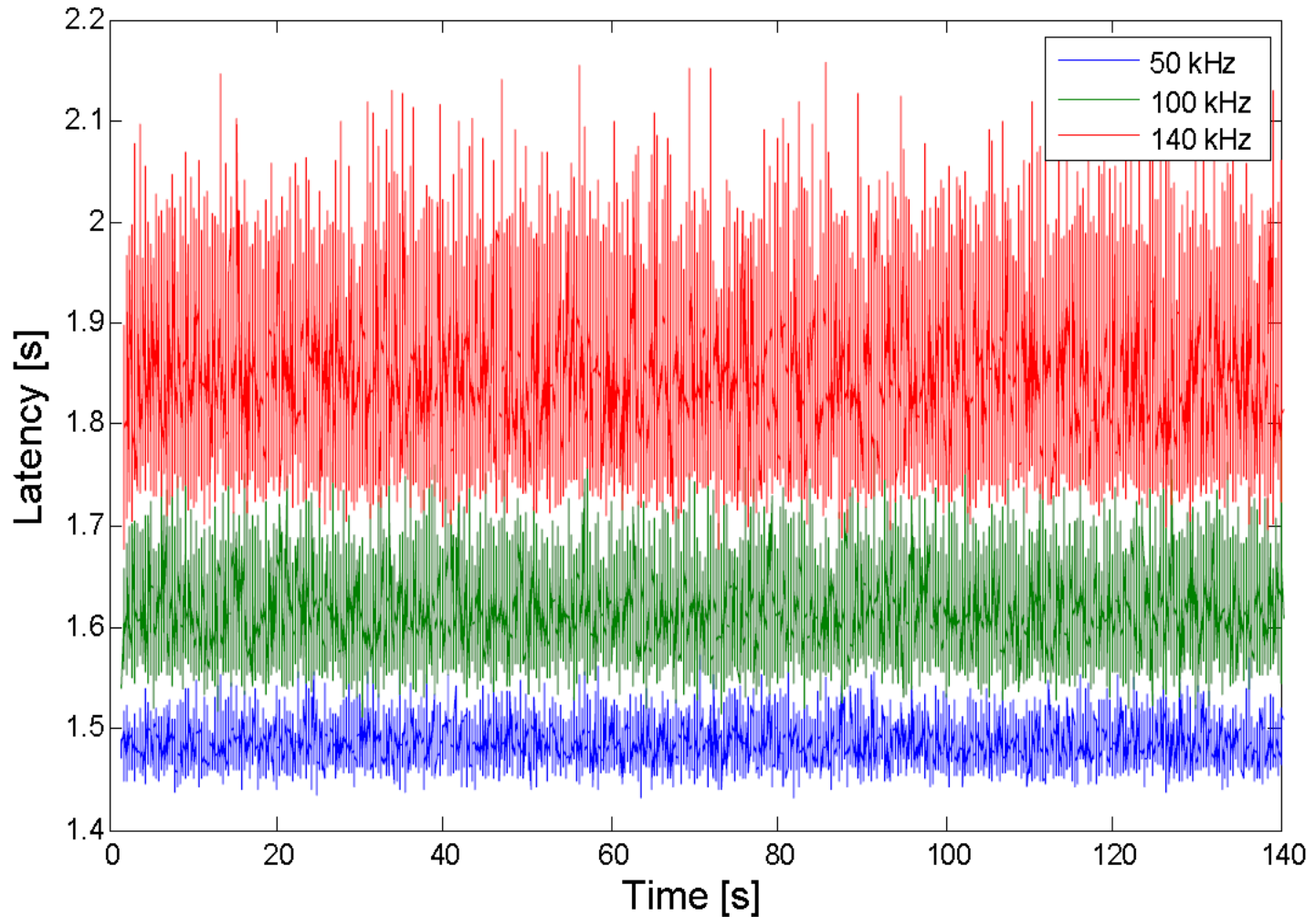
ALICE O2 System: A Few Simulation Results



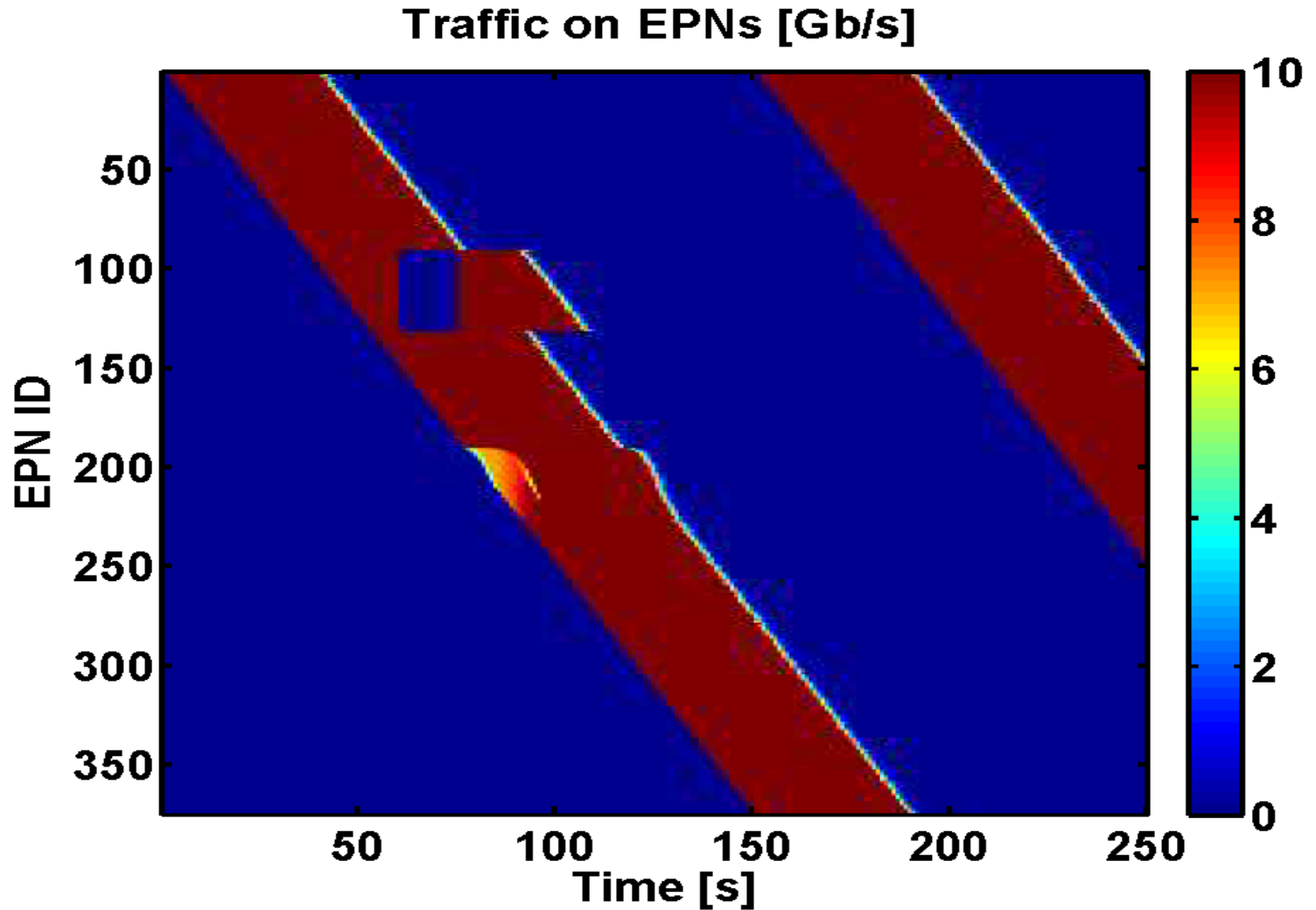
ALICE O2: Traffic OUT from FLP units



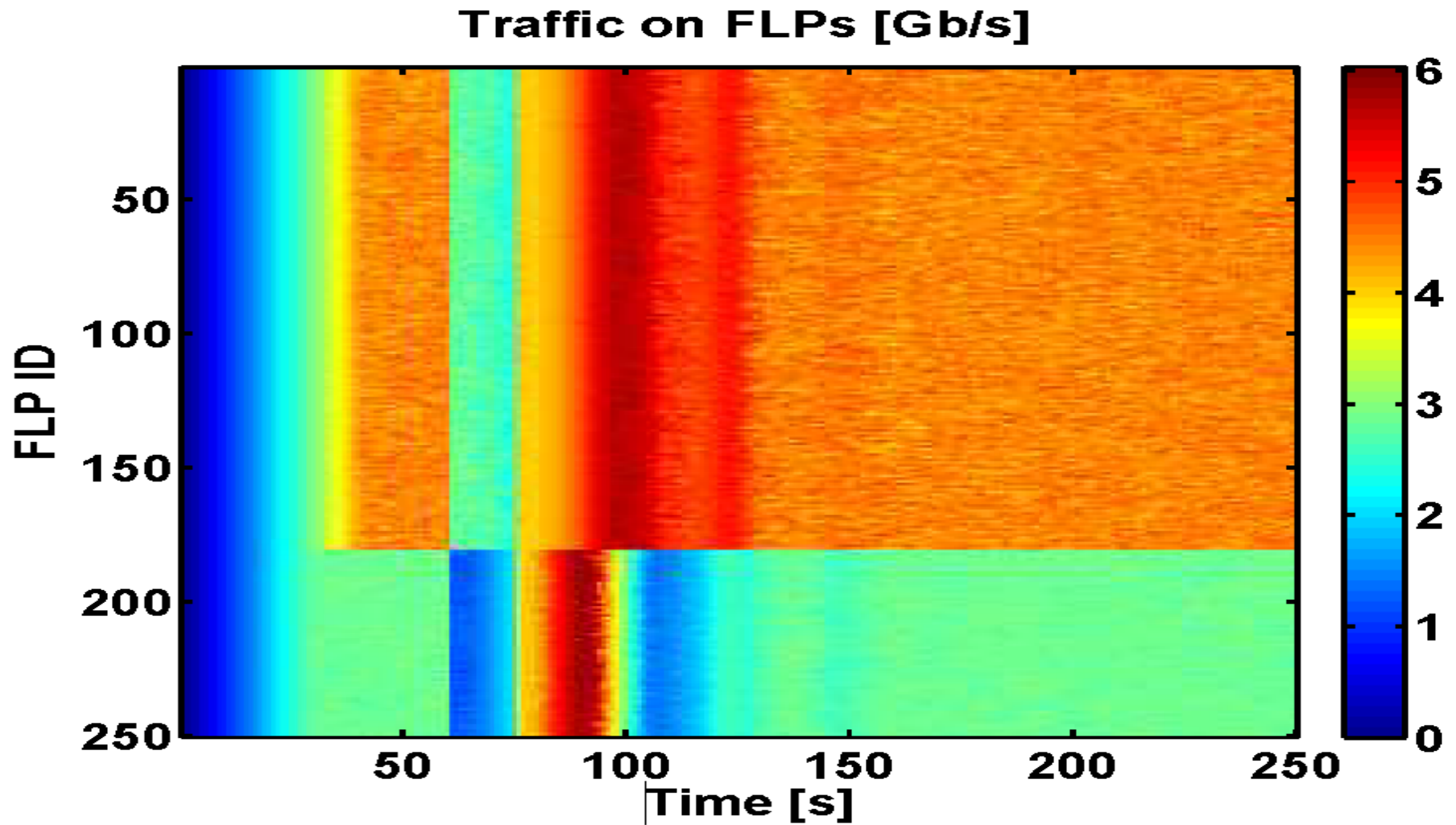
Super ENP case Latency at different input rate



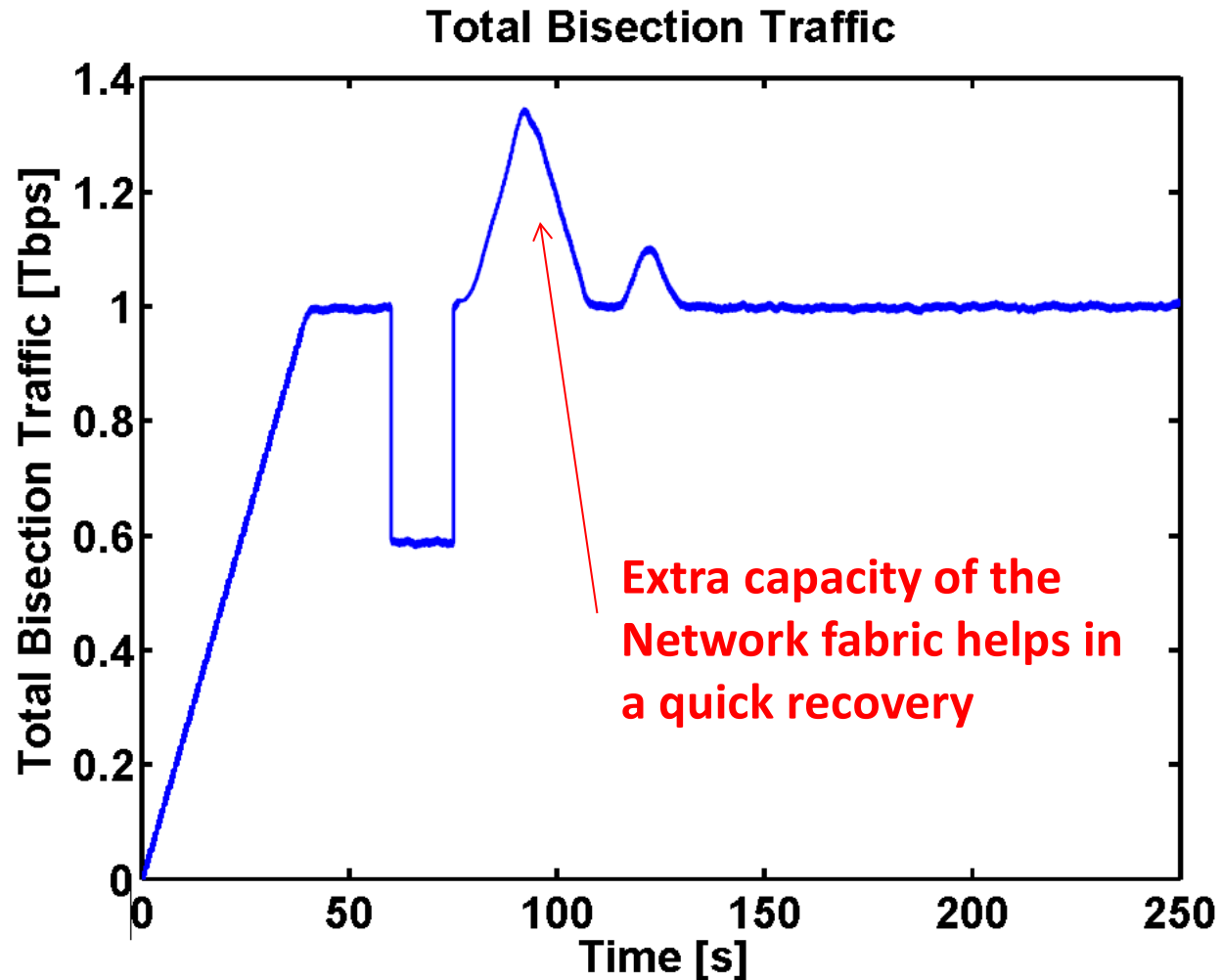
Simulate an Error on EPNs connection ~ 5s drop that affects ~ 10% of the EPN nodes



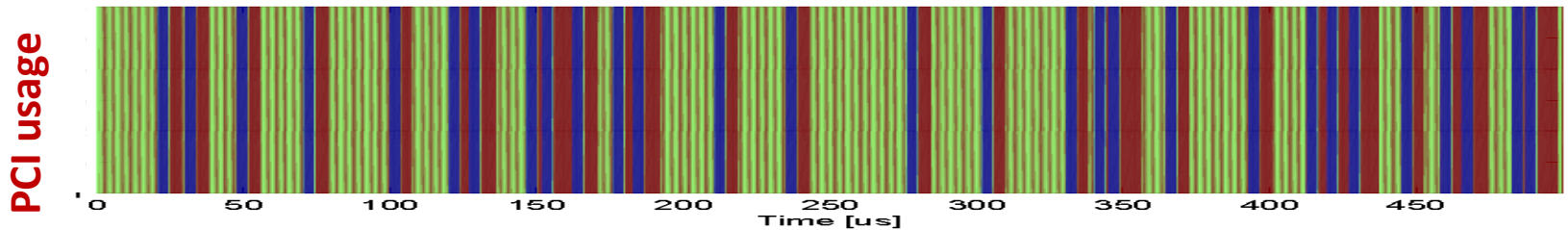
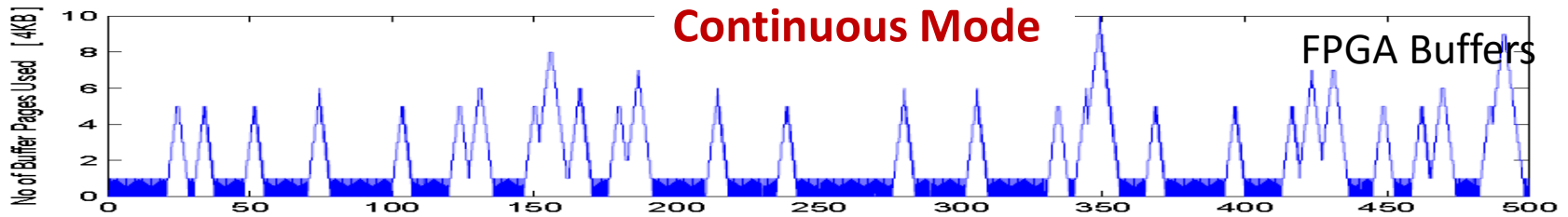
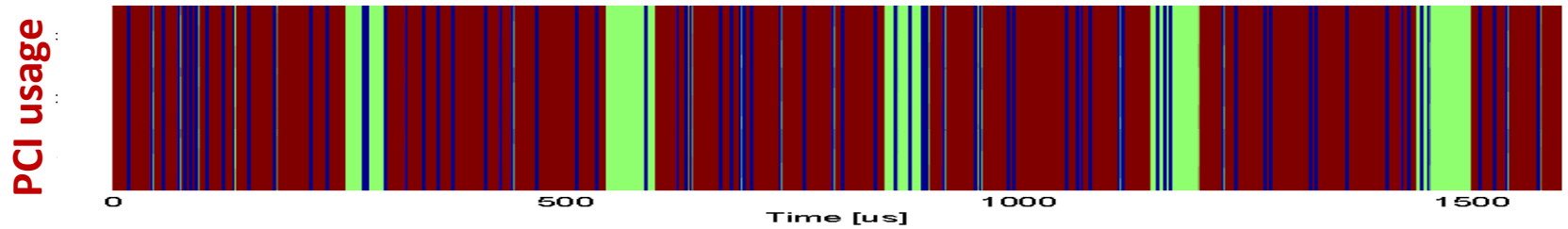
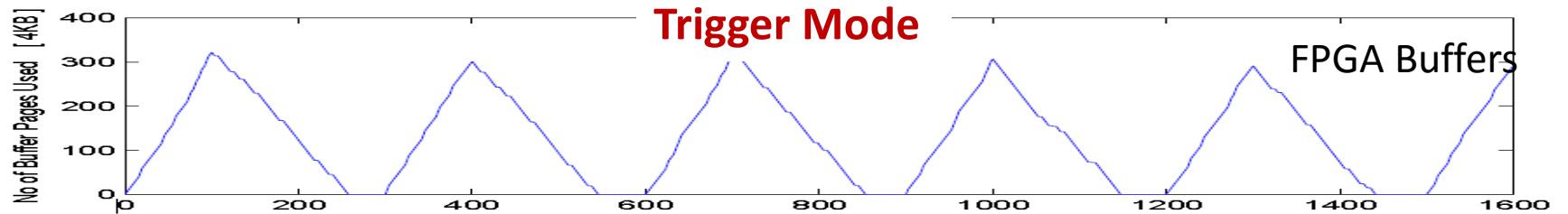
Simulate an Error on EPNs connection



Simulate an Error on EPNs connection



ALICE: FLP Internal Data Flow

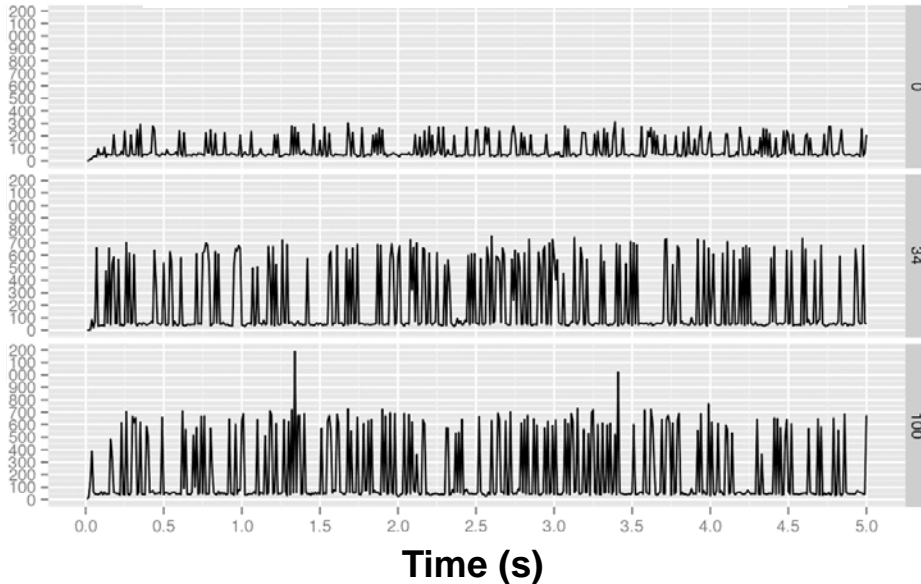


Simulation can help understanding the behavior of complex systems

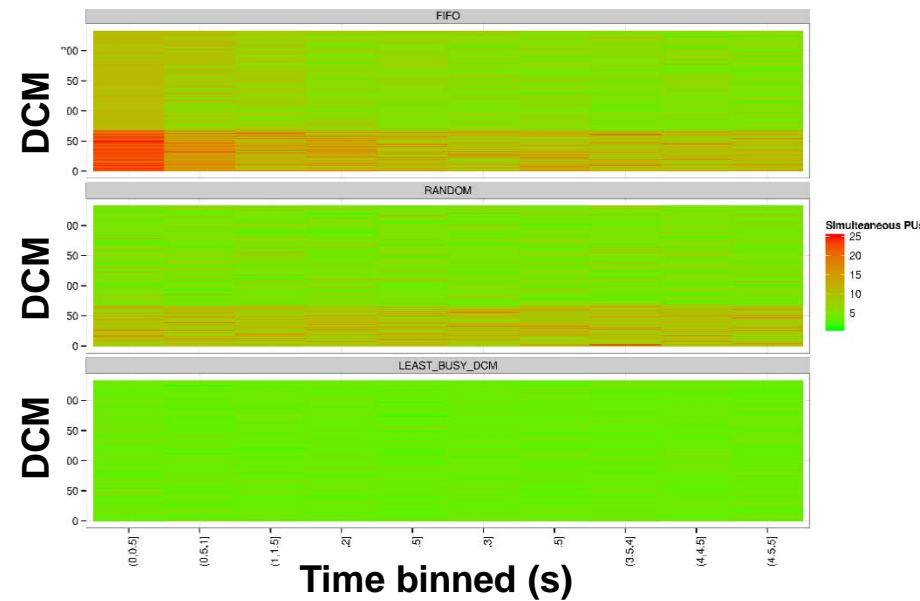
- Focus can be made on different components, scenarios, time scales, etc.
- Simulation permits better visualization and analysis (data gathering is easy)

Kbytes queued in the core switch

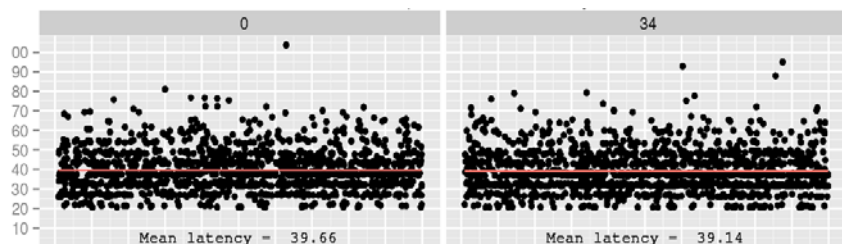
Core Switches Queue sizes evolution



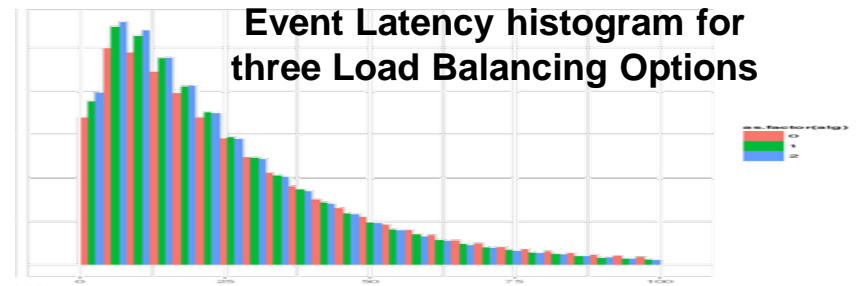
Farm usage visualization – Load per TPU



Individual Event Build Latency Evolution



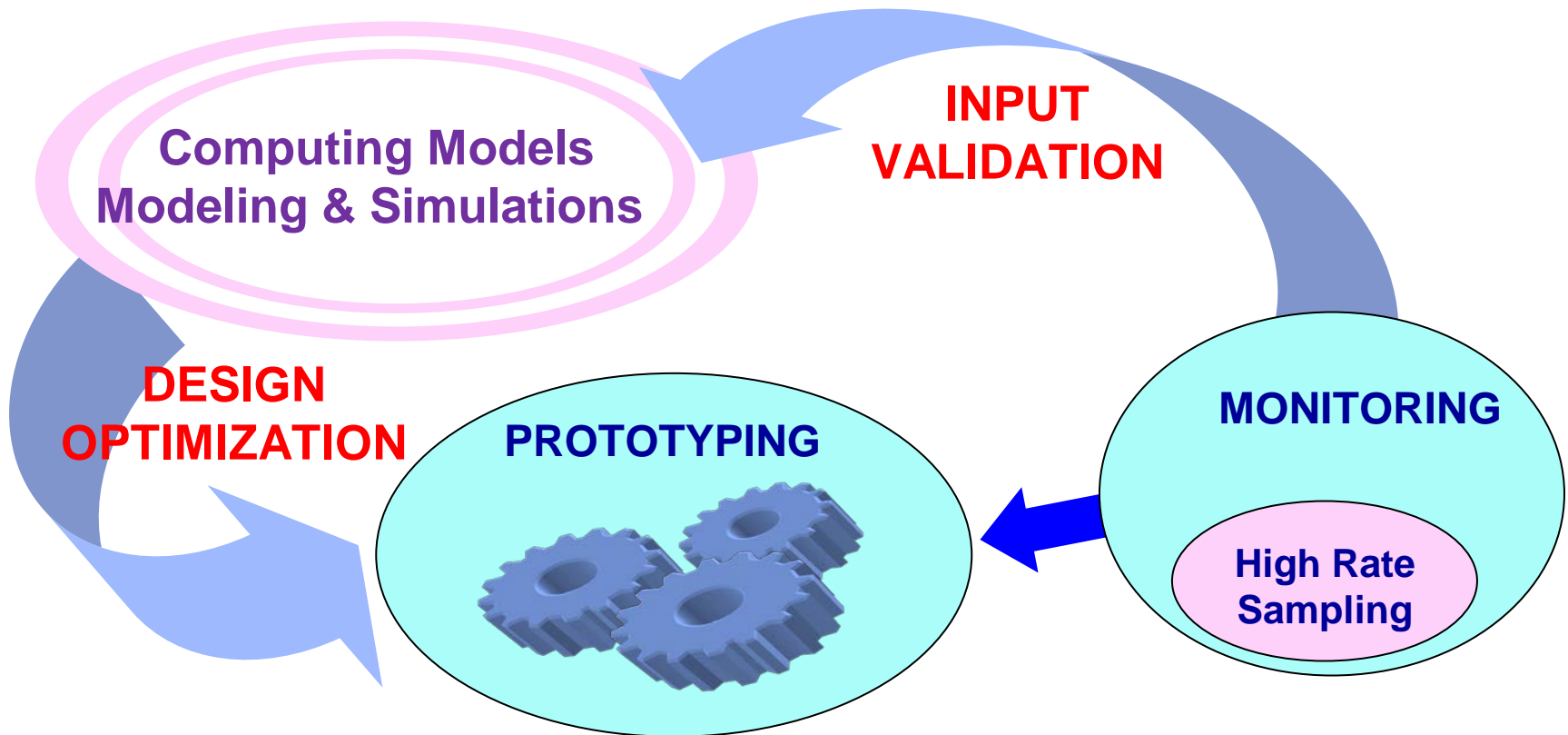
Event Latency histogram for three Load Balancing Options



Main Features Necessary to Simulate Complex DAQ Systems

- **Hierarchical Simulation** model capable to describe independently each sub-system and to include different levels of details on each of these components.
- Model correctly all the **hardware components** and the **management software** used to control the data flow . It should be easy to “plug –in” different control software algorithms.
- The **statistical distribution** for the physical parameters (data buffers, computing time ..) and the response time for complex sub-systems is essential. It is necessary to have **good monitoring** information on prototype systems.

Simulation and Modeling are important for the design of an efficient, cost-effective system



- Simulation and modelling should be part of the system design as a continuous process to validate and optimize the overall on-line computing model.
- **Computing system simulation should be something very similar with Monte Carlo simulations for the physics part.**

Summary

Appropriate simulation frameworks can really help in the design of DAQ computing systems, management software and optimization for the algorithms that control the data flow.

- Compare different architectures
 - Estimate how systems will scale with increased data volume and rates
 - Risk analysis & Cost Optimization
- **Common problems and the methodologies used are quite similar. We believe that collaborative projects between the LHC experiments in simulation and modeling for the DAQ systems can help the development of such simulation frameworks.**

Thank you !

Questions ?

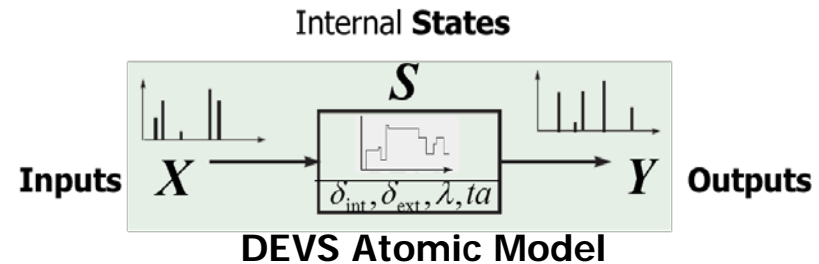
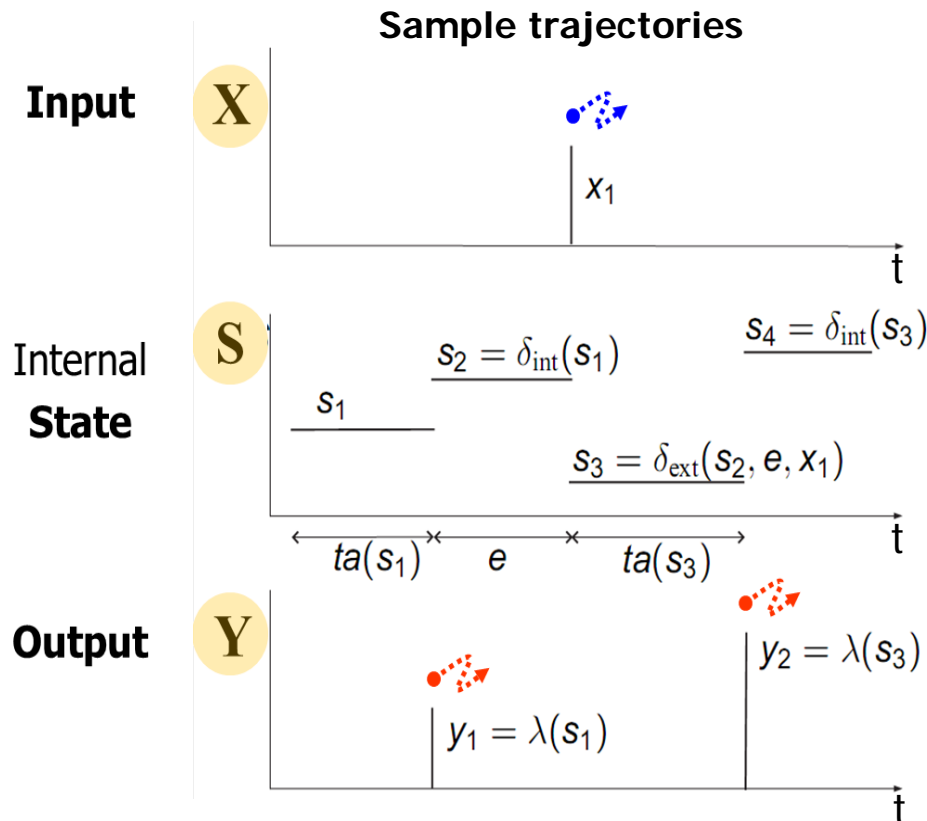
Backup Slides


DEVS Model Definition


A DEVS model is defined by the following mathematical structure:

$$M_D = (\underbrace{X, Y, S}_{\text{Sets}}, \underbrace{\delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta}_{\text{Dynamical Functions}})$$

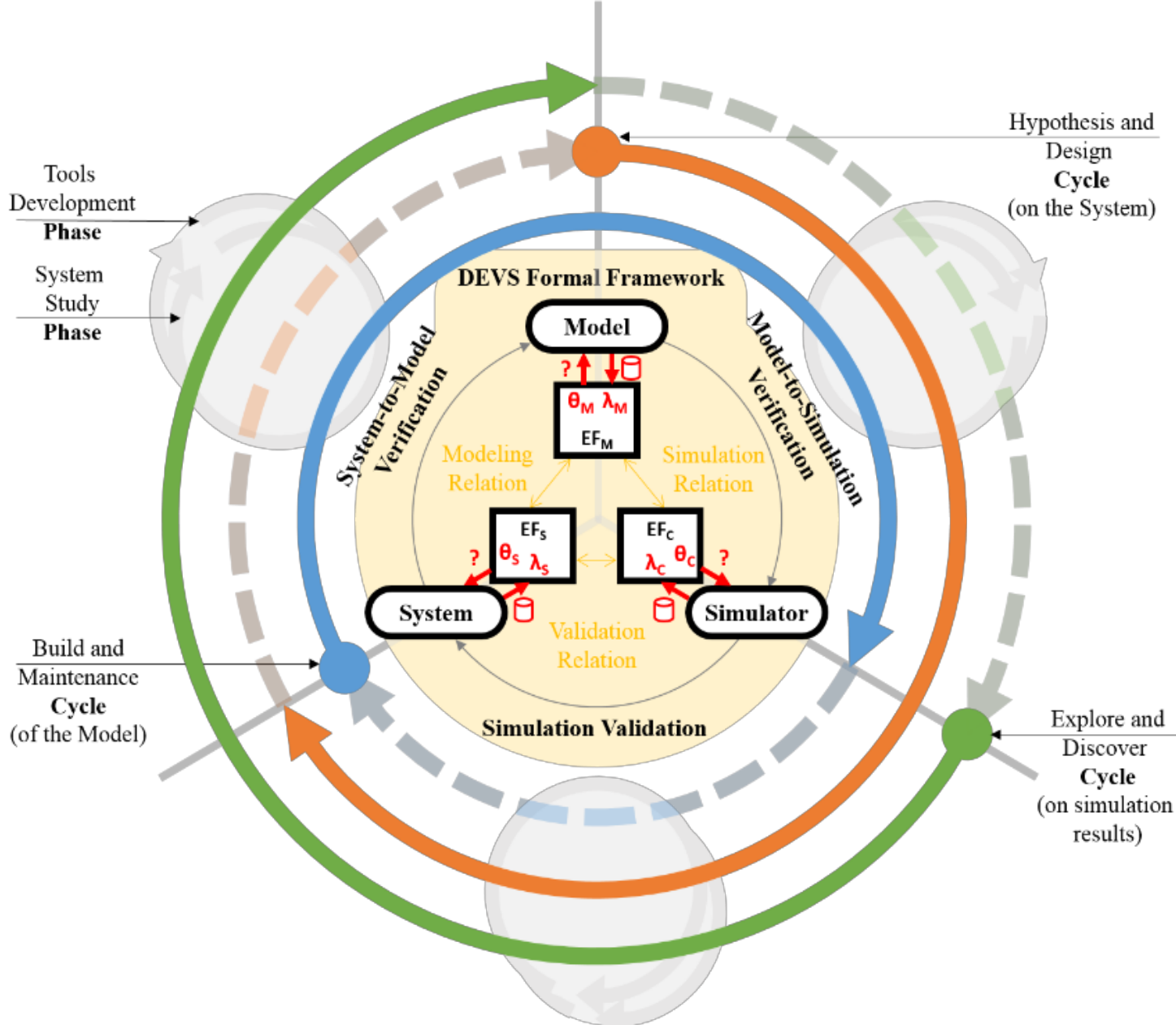
Implemented by the modeler in any programming language of choice



 **Internal Transition**
(Independent of the *external system*)

 **External transition**
(Dependent on the *external system*)

Methodology: M&S-driven engineering



Cycles:

- **Design**
- **Build**
- **Explore**

Foster quick deliverables and early feedback. Focus on most important aspects first.

Phases:

Allow for **separated** yet **encompassed** development of **models and tools**