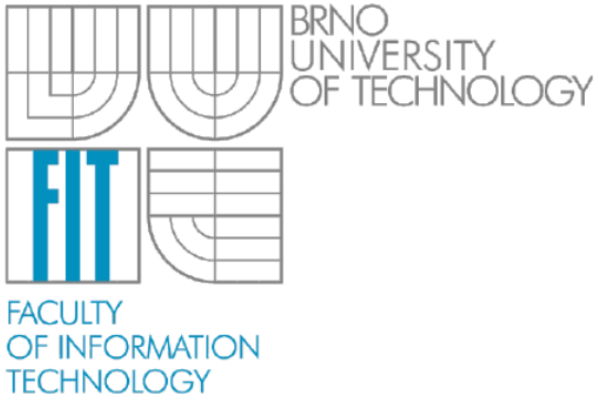
TID

Processing of network traffic in low-power devices with FPGA

Jan Viktorin

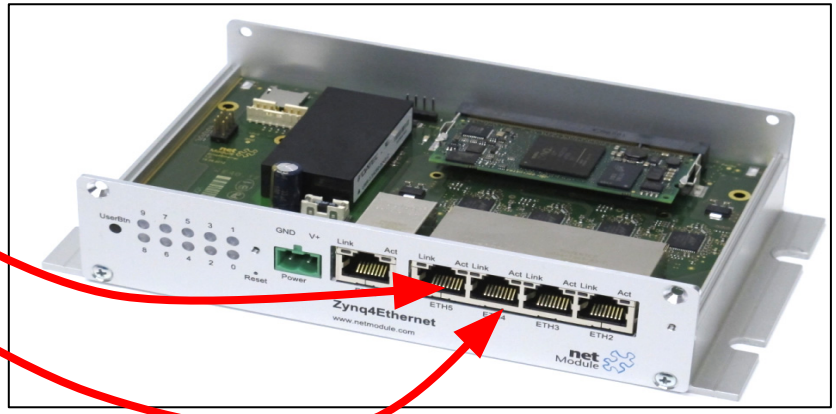
December, 2014



Introduction & Motivation

Motivation: Embedded probe

Internet



Packet filtering

Deep packet inspection

Embedded solution working at **wirespeed** 1-10 Gbps is a challenge. FPGA is needed for acceleration.

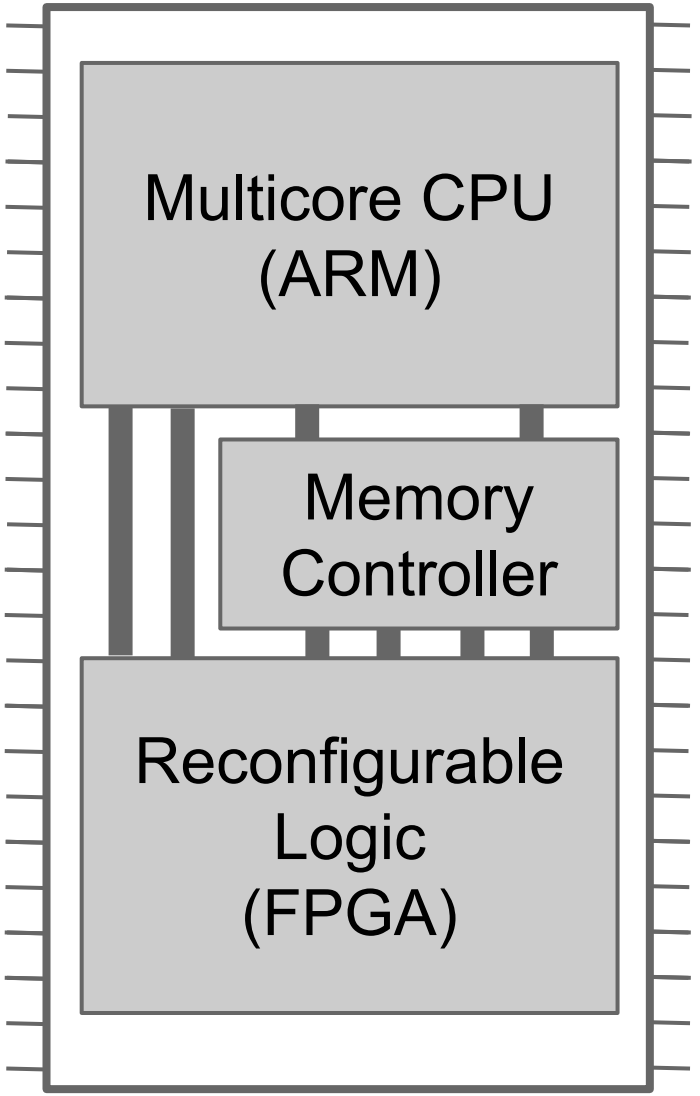
Low-power is a basic requirement today.

Software Defined Monitoring

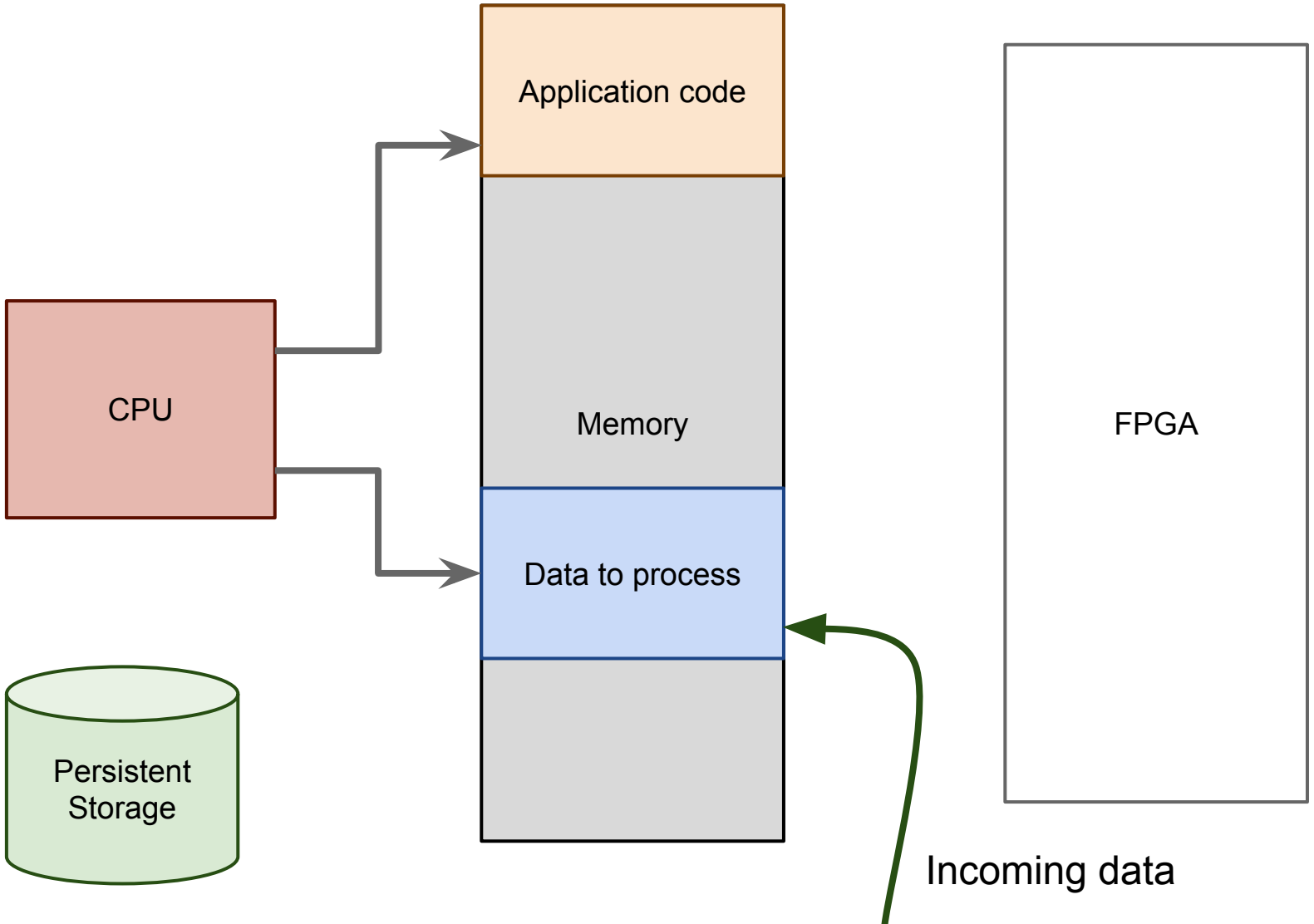
Collecting of netflow statistics

Protection of end-user networks

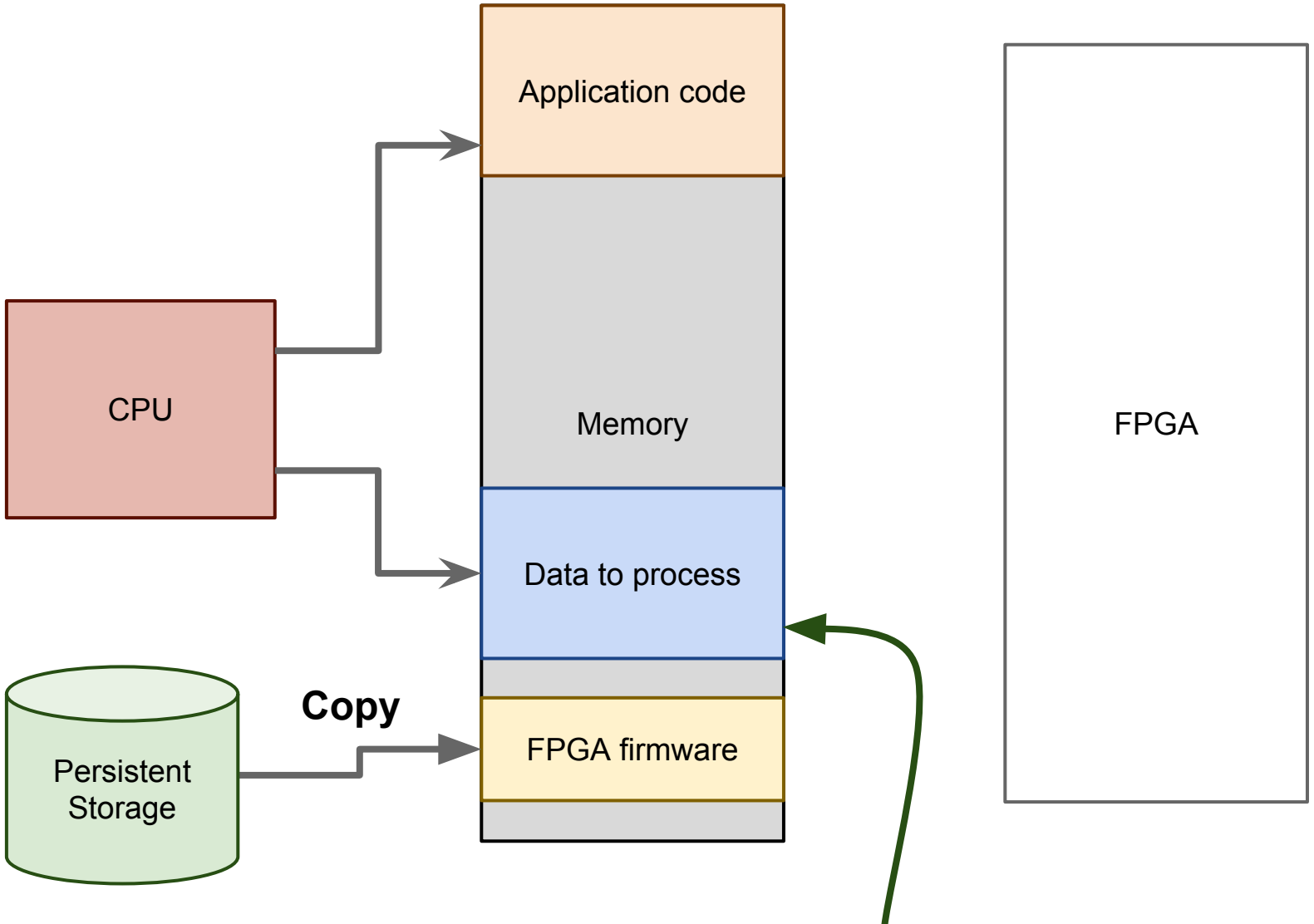
CPU with FPGA



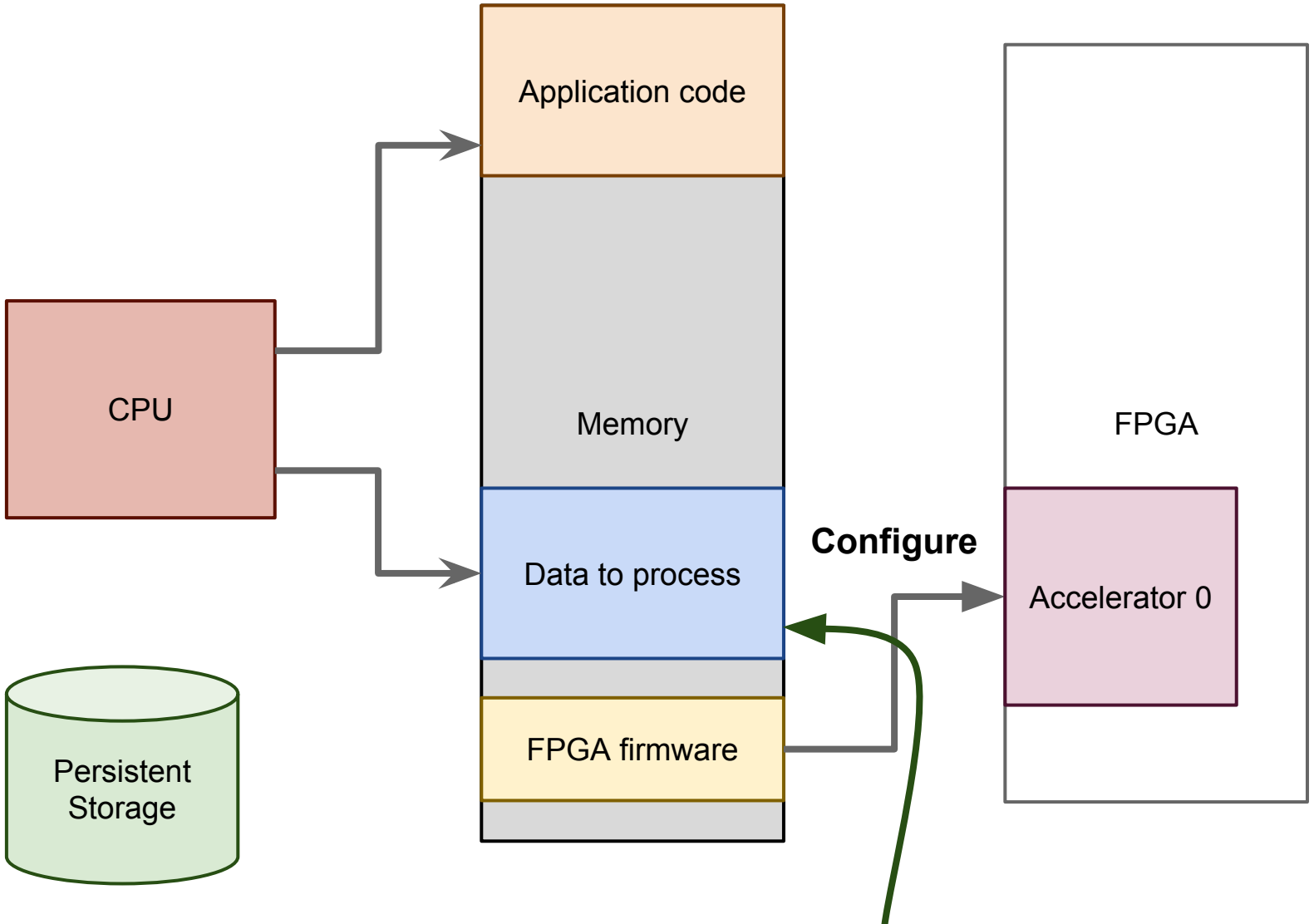
Task offload to FPGA (1)



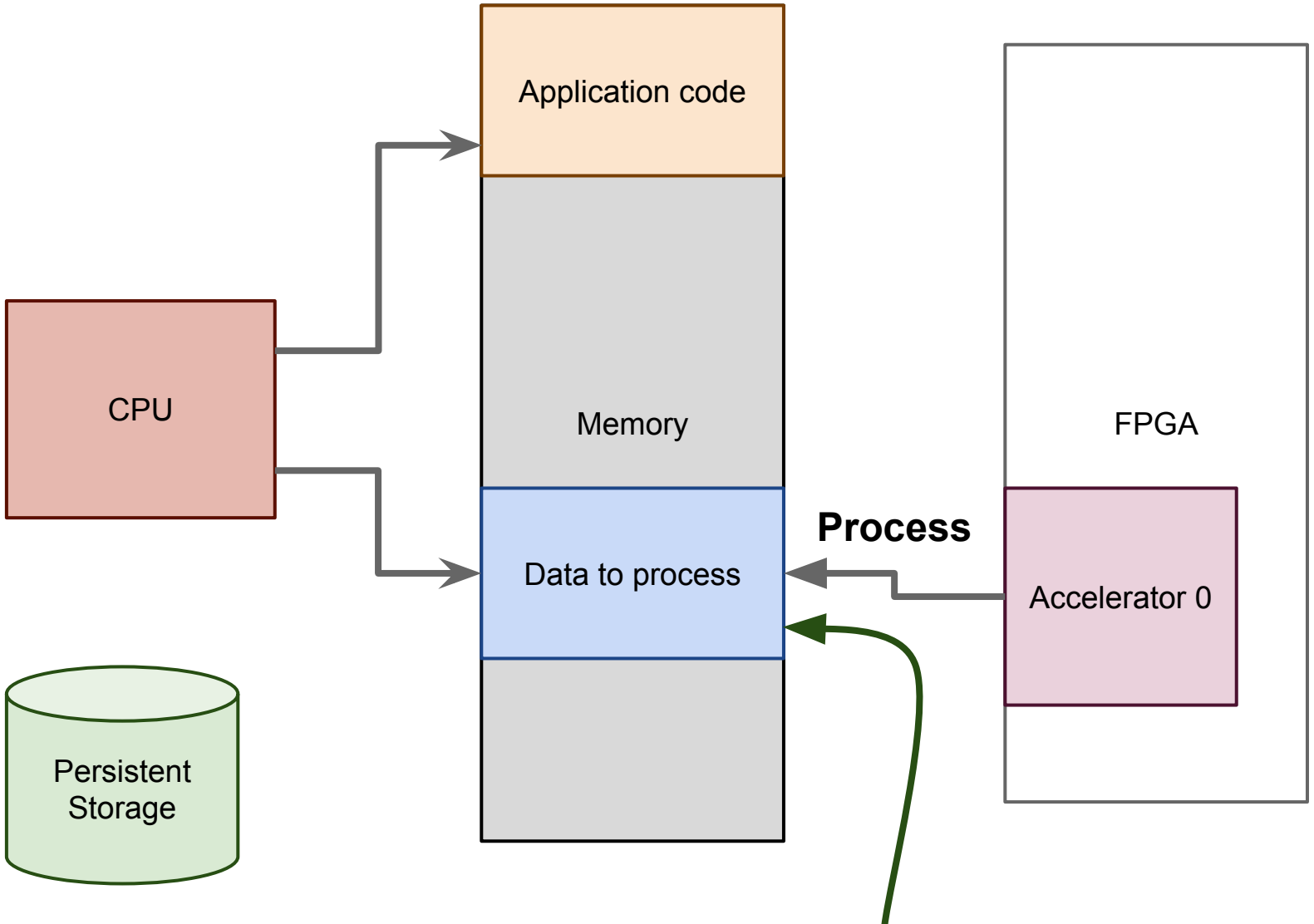
Task offload to FPGA (2)



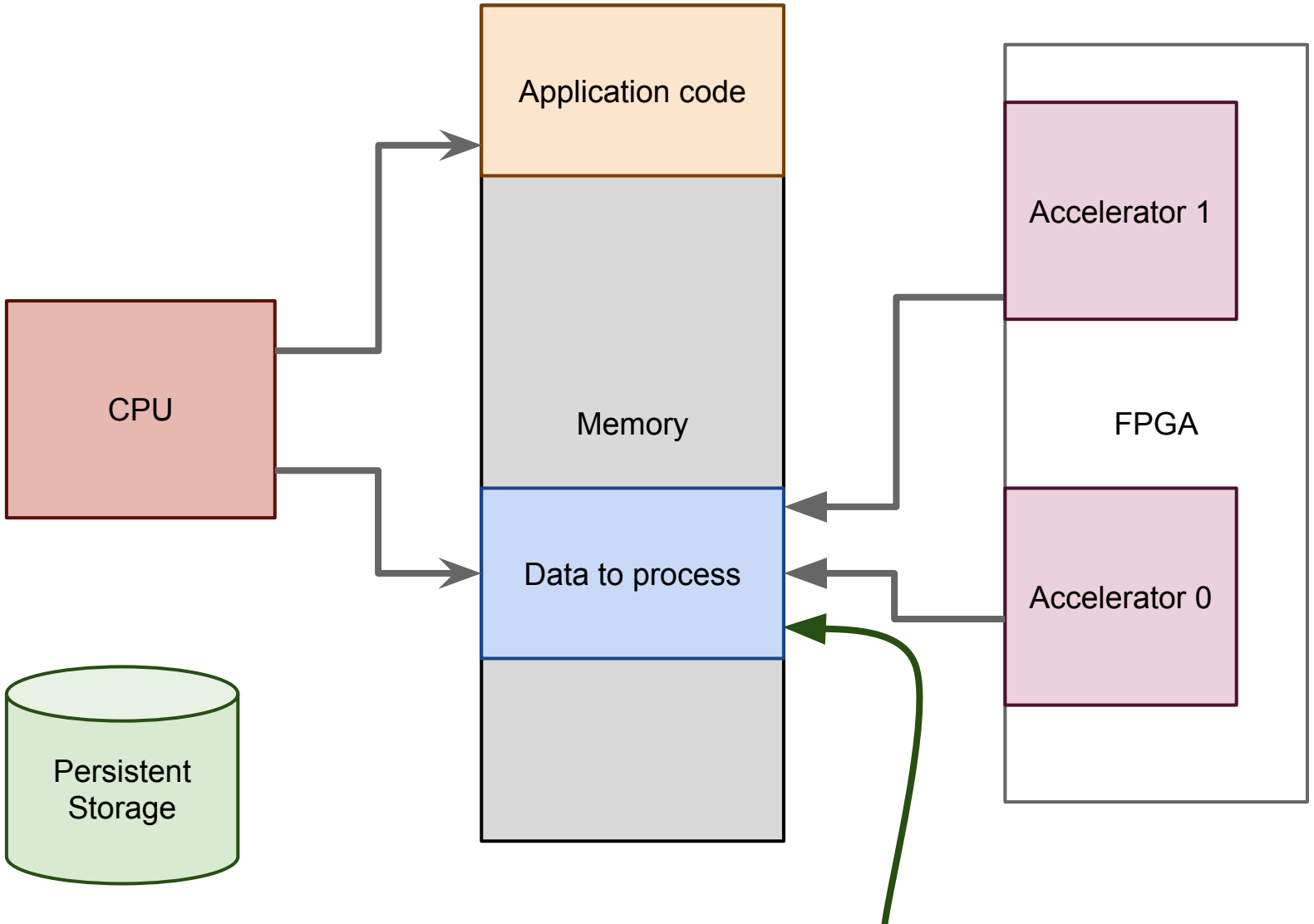
Task offload to FPGA (3)



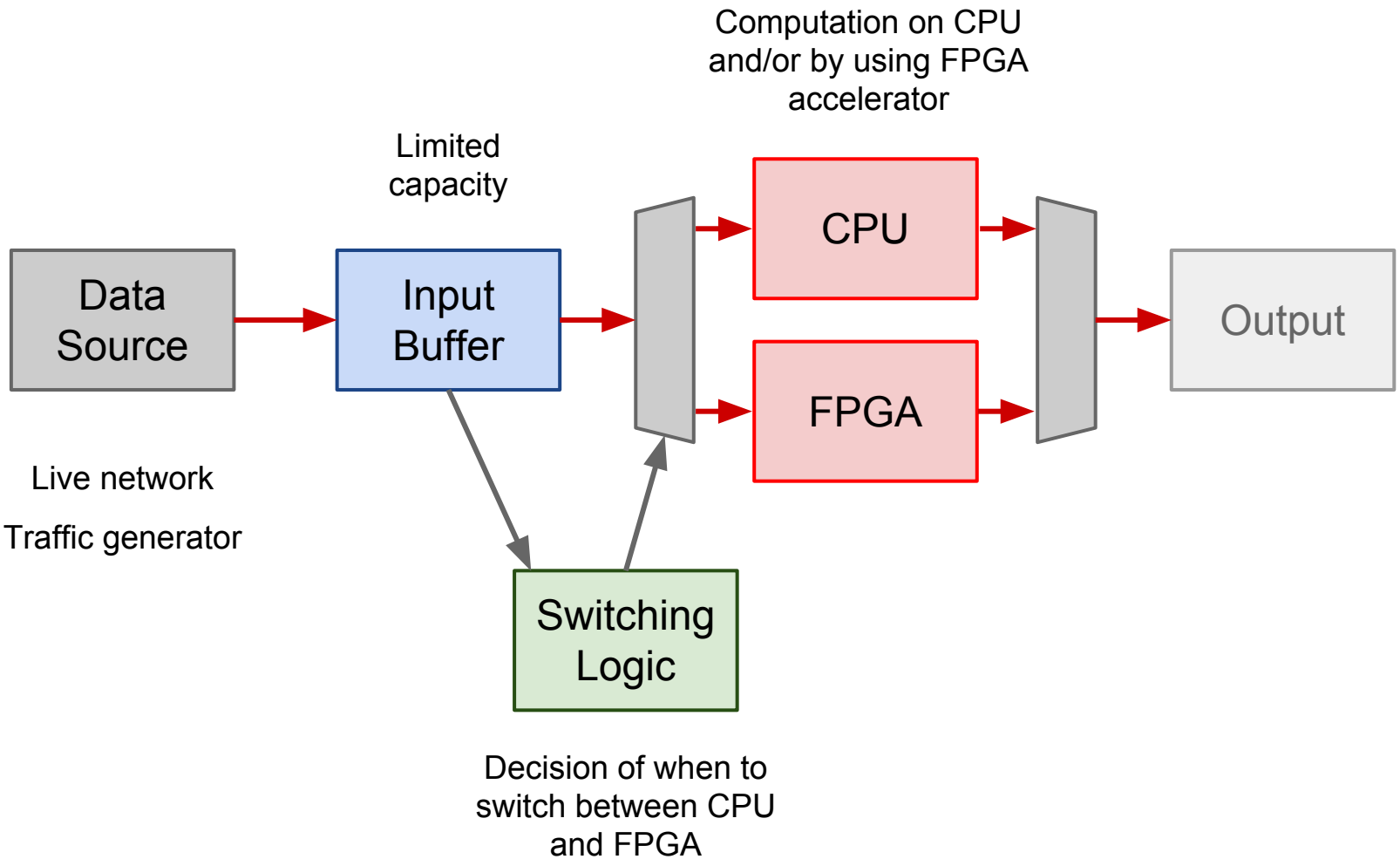
Task offload to FPGA (4)



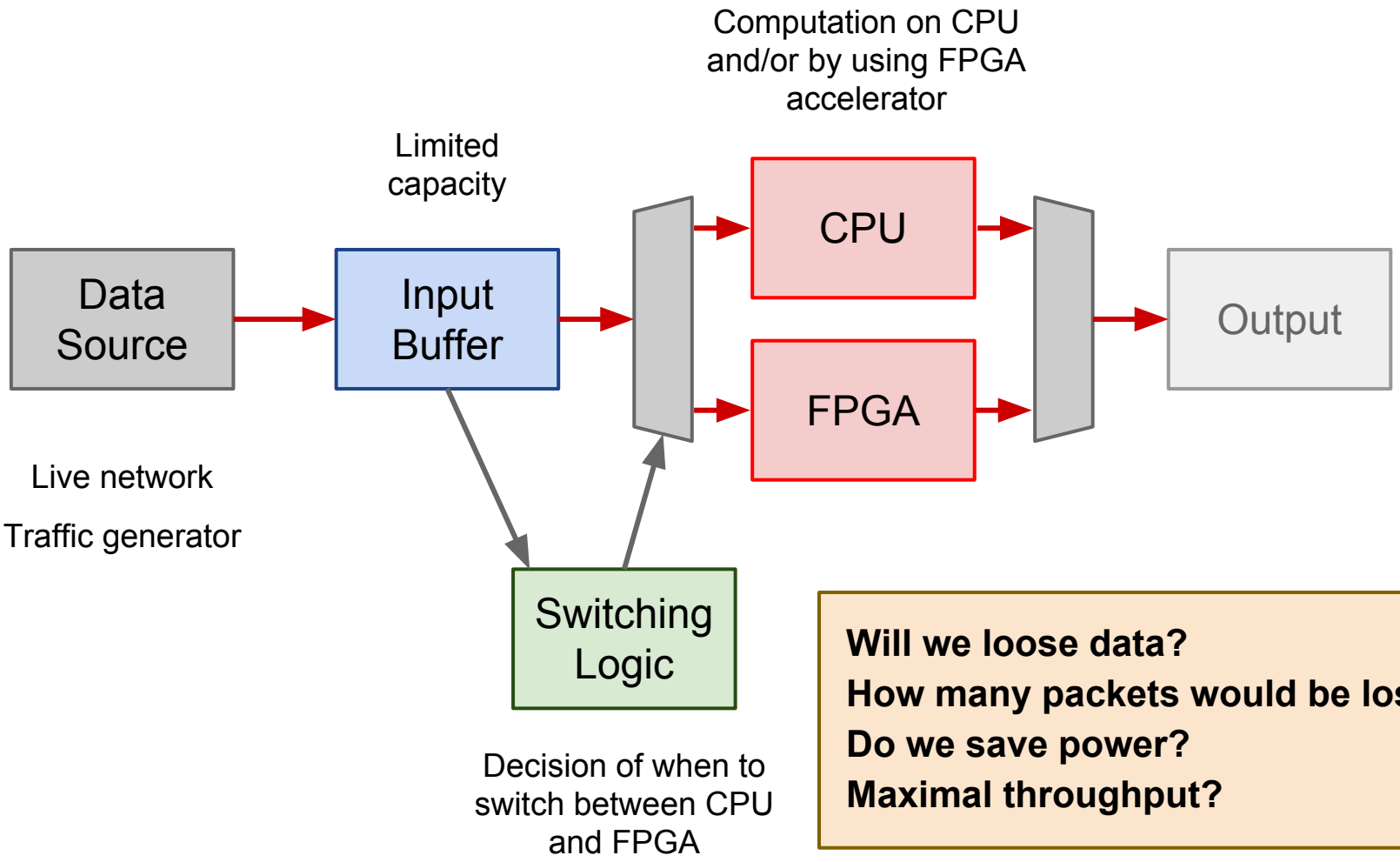
Task offload to FPGA (5)



System to be simulated



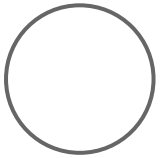
Behaviour?



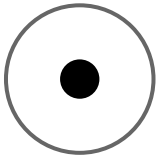
Will we loose data?
How many packets would be lost?
Do we save power?
Maximal throughput?

Petri Nets

Petri Nets

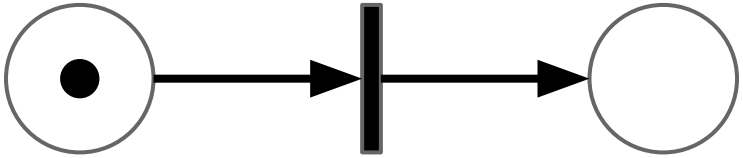


Empty place

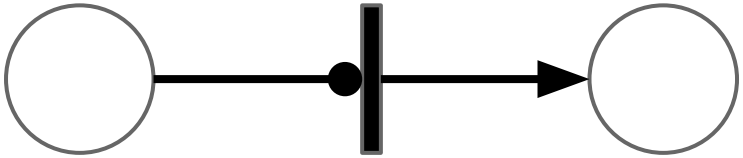


Place with *token* (or *marker*)

Transition (or event)



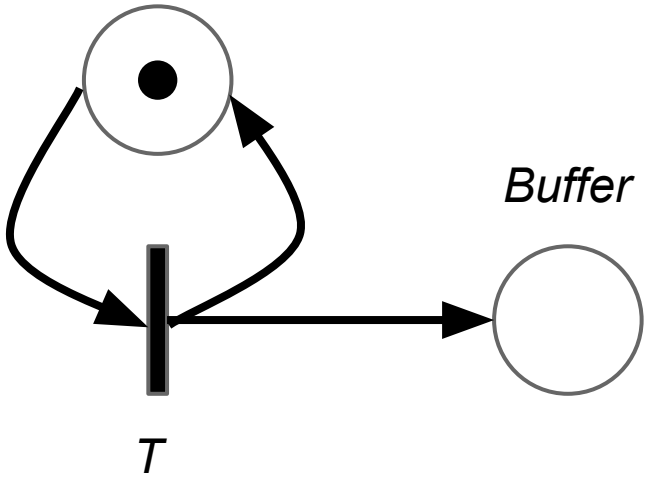
Transition that is ready to *fire*



Transition that is ready to *fire*
by the *inhibitor arc*

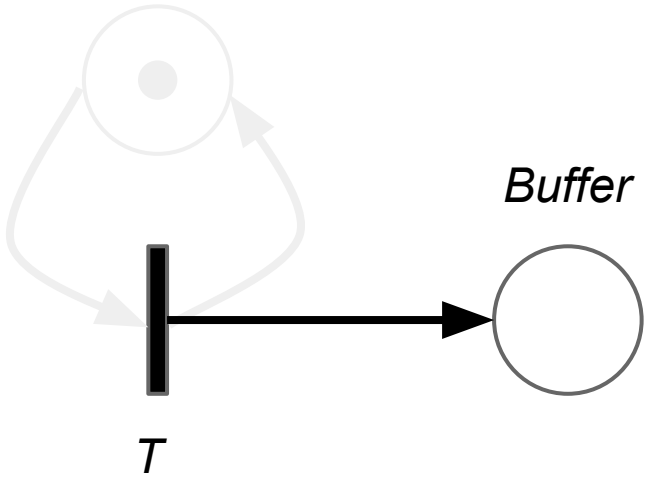
Petri Nets with inhibitor arcs are *Turing complete*

Example: Data generator



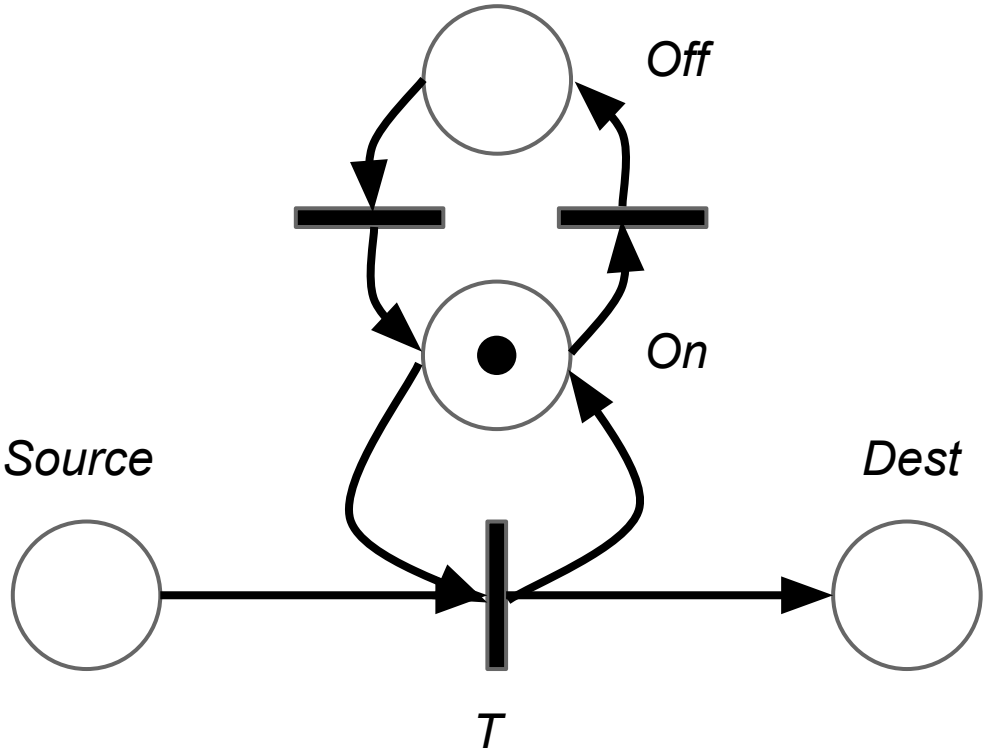
Generates a new token into the *Buffer* every *T* time units.

Example: Data generator



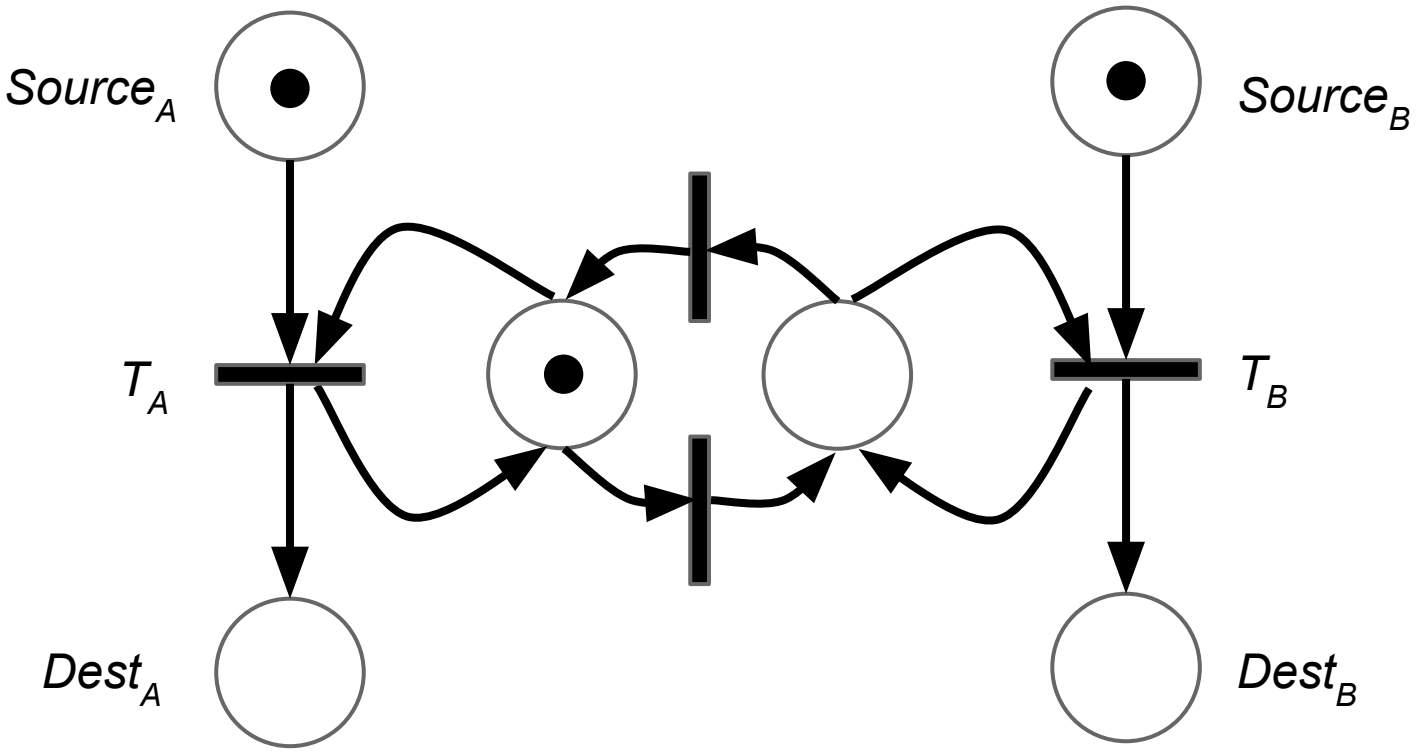
Generates a new token into the *Buffer* every *T* time units.

Example: Processing unit (CPU)



Processing unit takes 1 token from *Source* and after *T* time units it produces a result, i. e. 1 token, into *Dest*. The unit may stop for some reason and resume again.

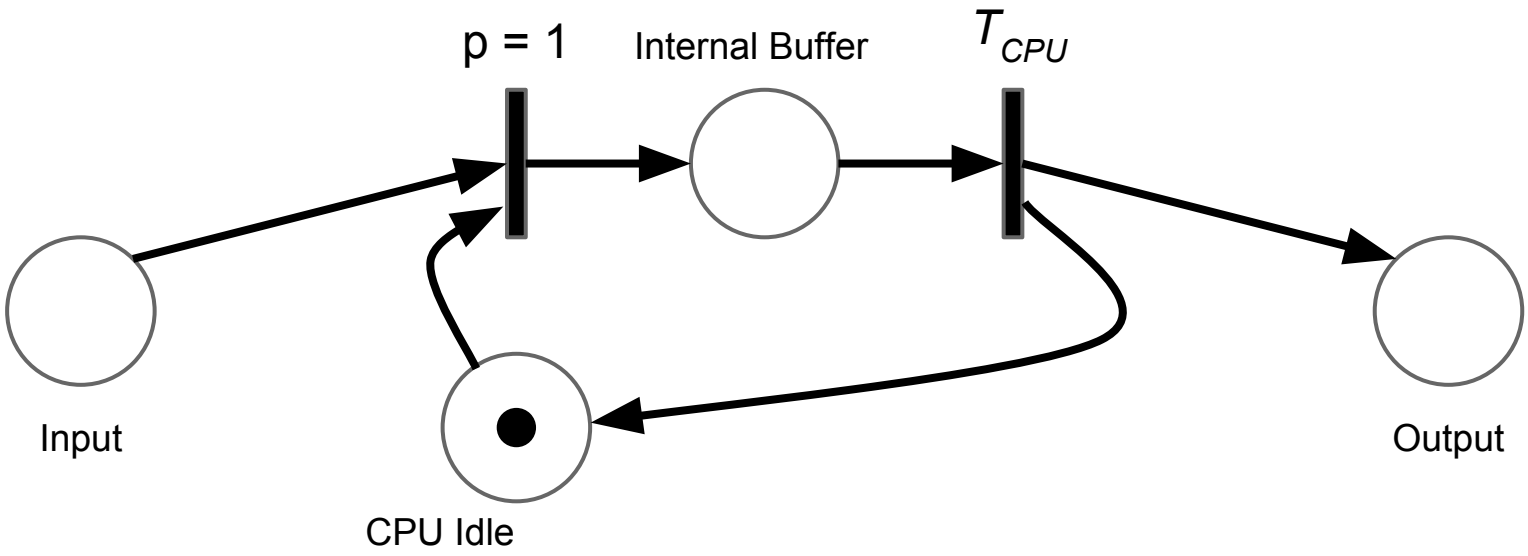
Example: Sharing CPU



A single CPU is shared between two processings A and B at random.

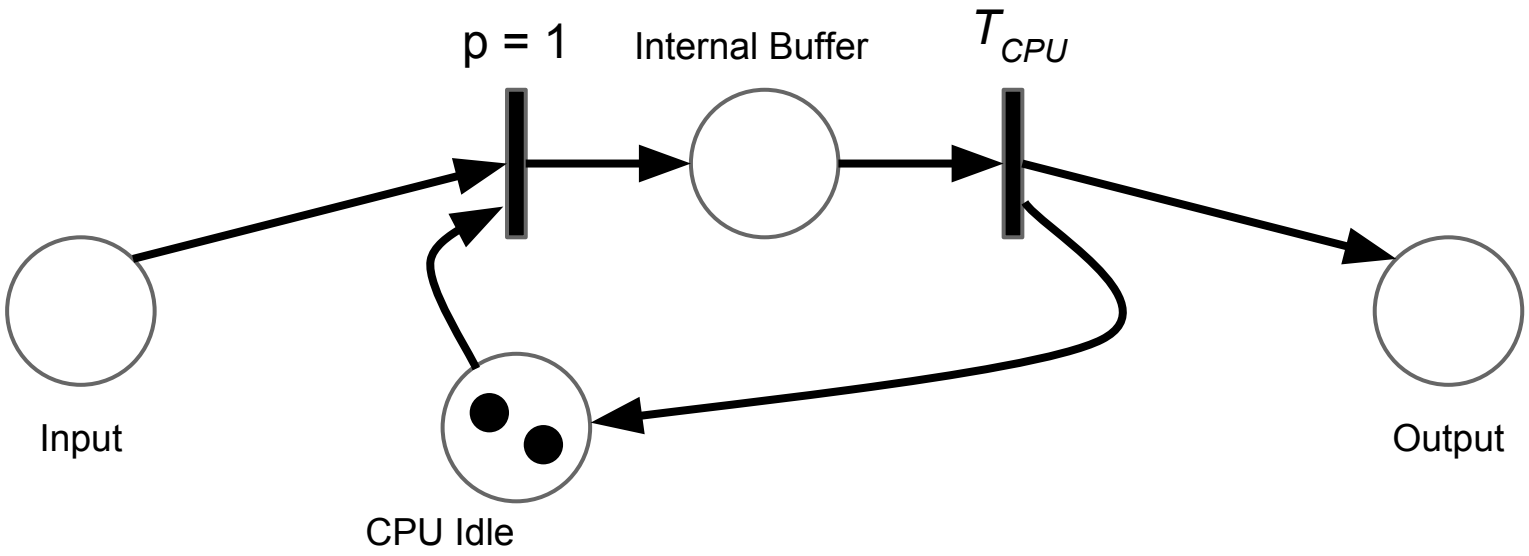
FPGA offload simulation

Processing packets on CPU



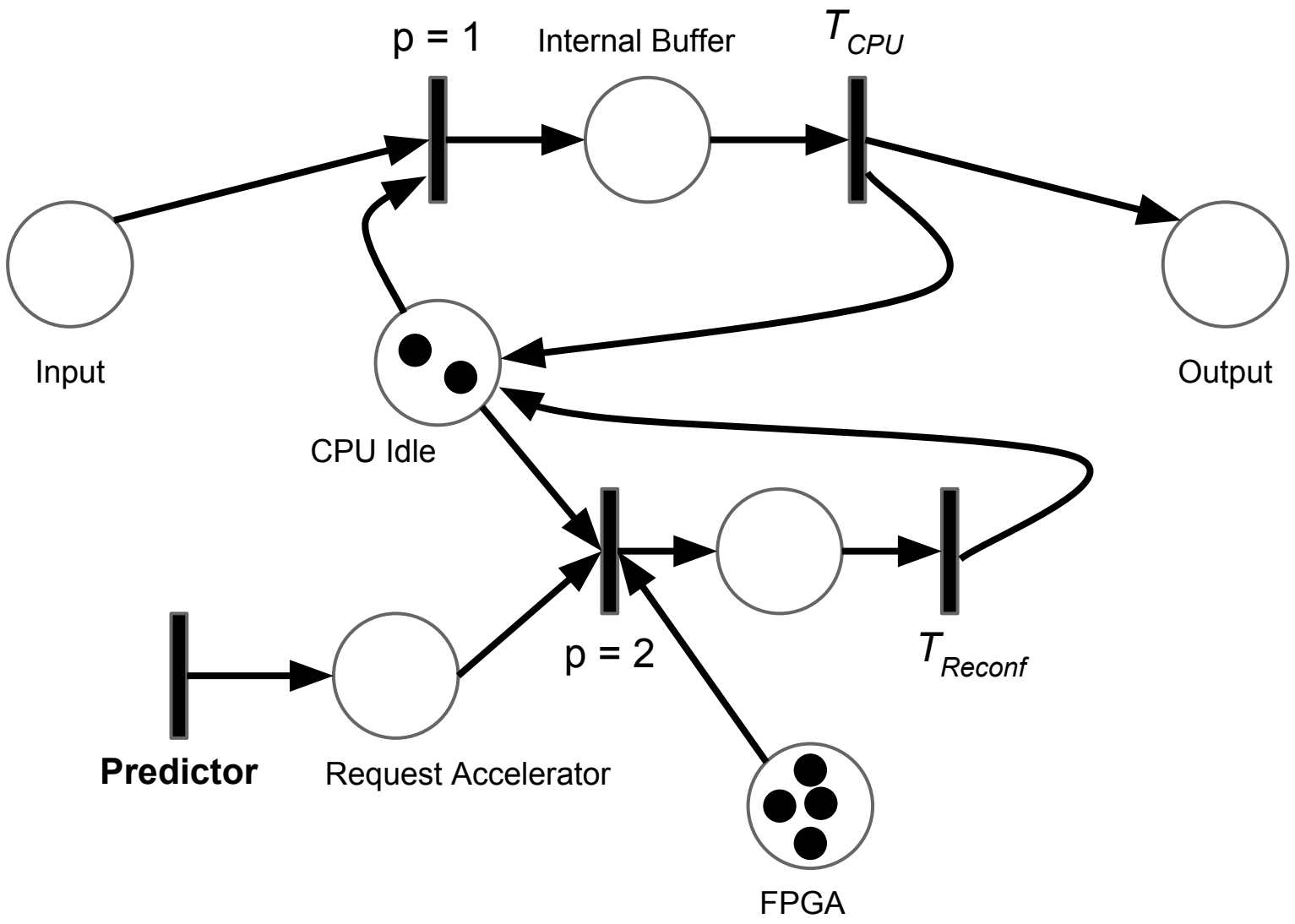
If the CPU is idle and there is **no other ready transition with higher priority** the system can process data from input. The processing takes T_{CPU} time units.

Processing packets on 2 CPUs

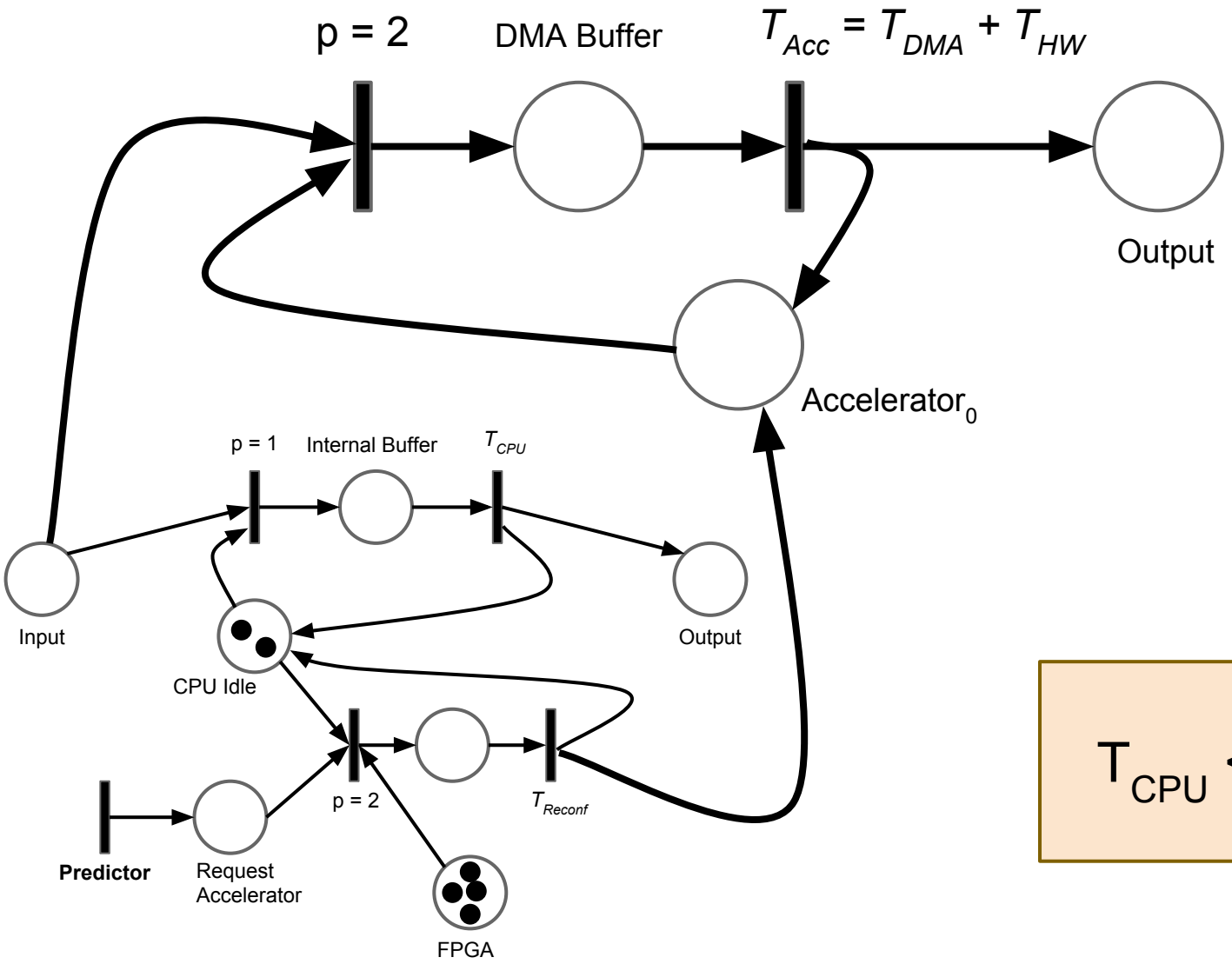


If a CPU is idle and there is **no other ready transition with higher priority** the system can process data from input. The processing takes T_{CPU} time units.

Prediction of reconfigration



HW Acceleration



$T_{CPU} << T_{ACC}$

State & Future work

Simulation model

- Current model is only restricted to predicting **overload** but **not underload**
- It is necessary to use high-level petri nets because the complexity of the model grows fast
- Improve model to simplify extensions - more complex pipeline

Simulation runtime

- Implement simulation runtime based on the designed Petri Net

Done

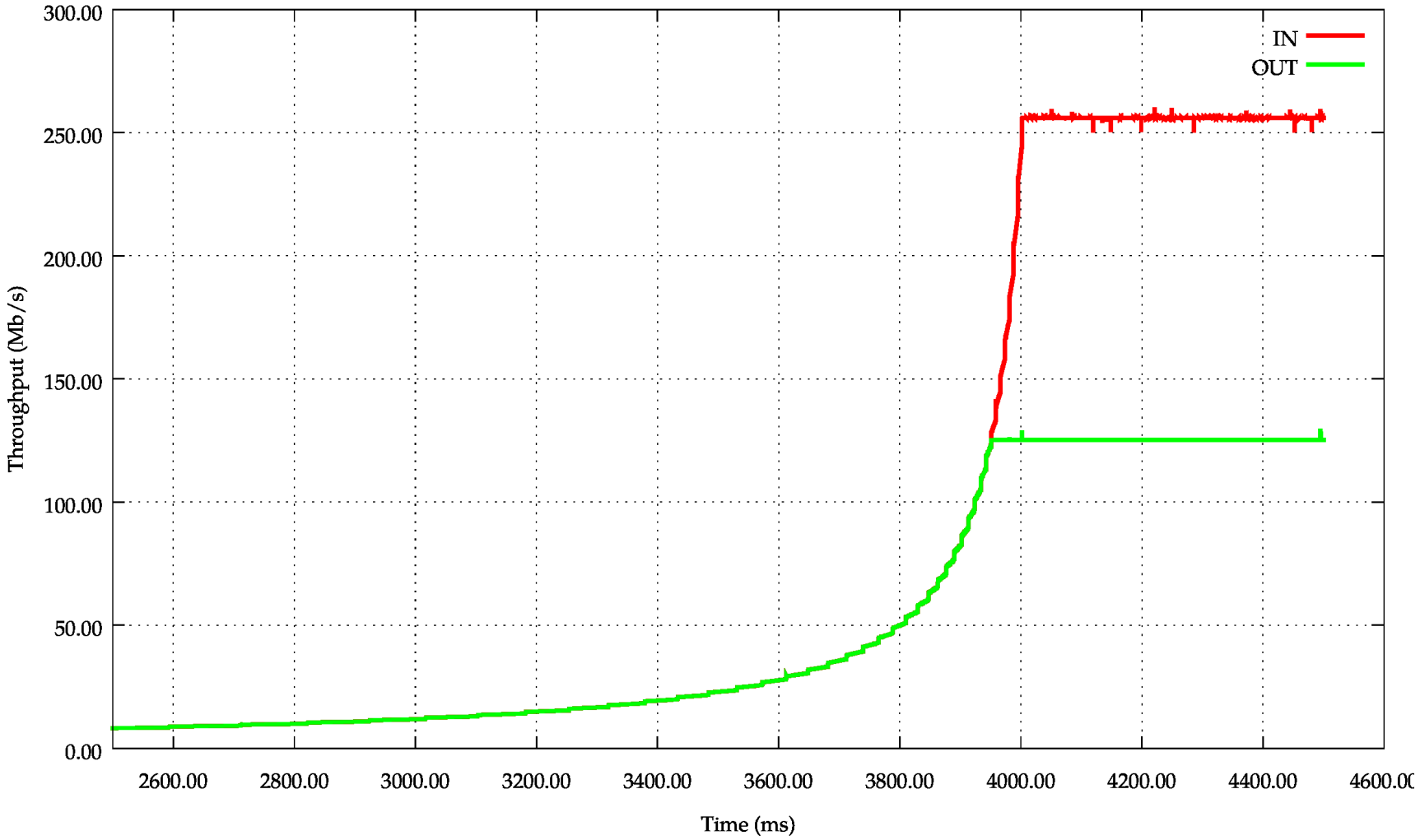
- Find appropriate parameters of the simulated system (memory delays, CPU load, accelerator's throughput, …)

Mostly done

- Design a good traffic predictor(s) and simulate its behaviour using the simulation system

Possible goal of my dissertation

Early experiments



Early experiments

- Generator of fixed sized packets at fixed bitrate to test throughput of the system
- Generator of fixed sized packets with increasing bitrate
- Simple predictor that requests reconfiguration based on maximal possible speed of link

References

DAVID, René; ALLA, Hassane. *On Hybrid Petri Nets*. Discrete Event Dynamic Systems: Theory and Applications. Volume 11. Issue 1-2. Netherlands, January 2001. ISSN 0924-6703, 1573-7594.

ALLAM, Mohamed; ALLA, Hassane. *Modeling and Simulation of an Electronic Component Manufacturing System Using Hybrid Petri Nets*. IEEE Transactions on Semiconductor Manufacturing. Vol. 11. No. 3. August 1998.

AGERWALA, Tilak. *Putting Petri Nets To Work*. IBM. December 1979.