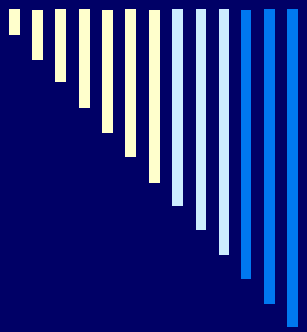
A Packet Scheduling Algorithm for IPSec Multi-Accelerator Based Systems

Alberto Ferrante and
Vincenzo Piuri

DTI
University of Milan

Fabien Castanier

AST
ST Microelectronics



Outline

1. IPSec;
2. The scheduling algorithm;
3. Model for simulations and results;
4. Architectural enhancements.

IPSec

- IPSec:
  - is a suite of protocols;
  - adds security at network (IP) level;
  - makes extensive use of cryptographic functions
    - it is resource consuming;
  - is included as security mechanism in IPv6.

1
2
3
4
5

Goals

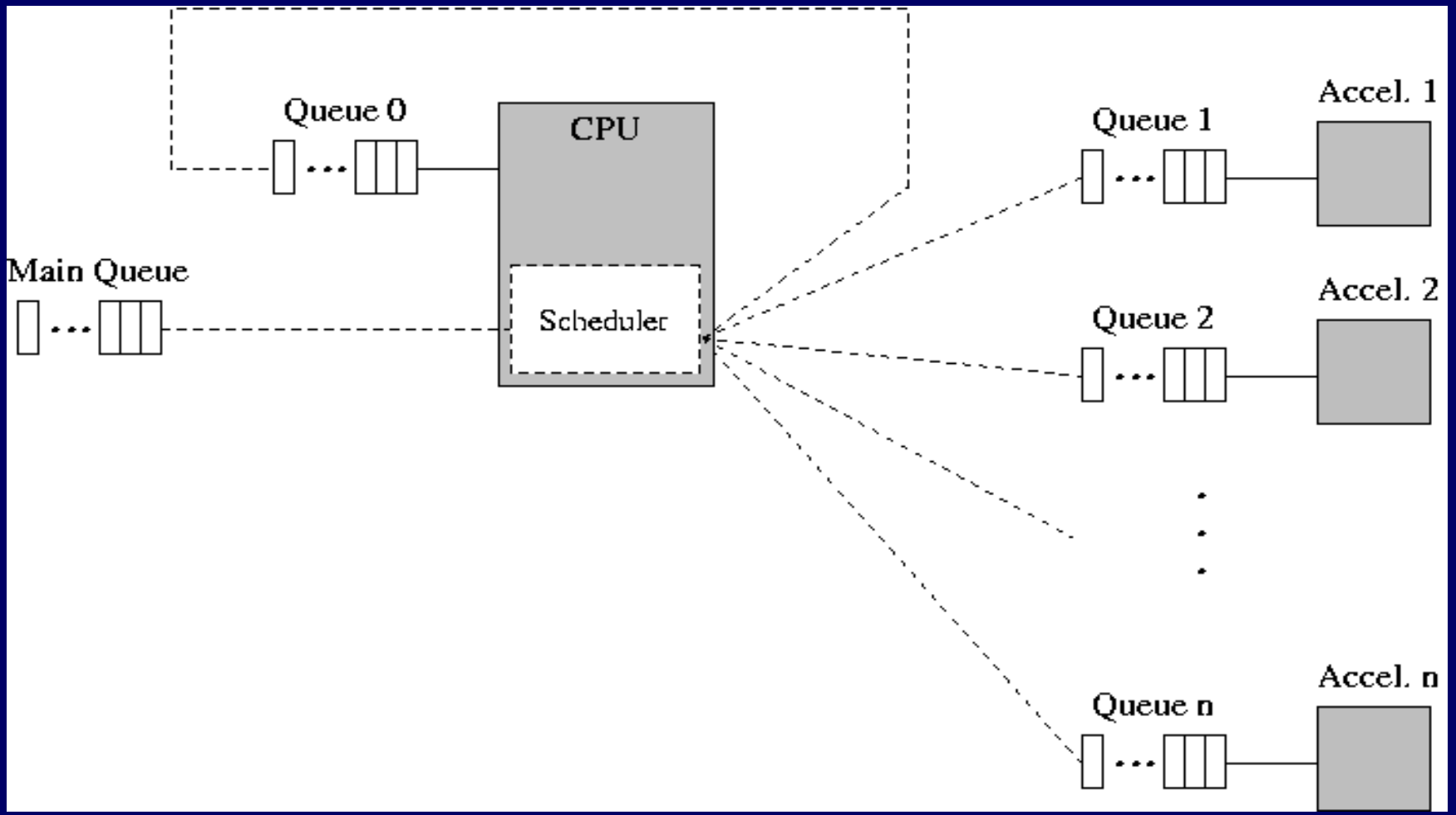
- Obtain a scheduling algorithm being able to:
  - Schedule packet processing between *N* crytpo-accelerators;
  - Schedule packets also to a software implementation of the cryptographic algorithms;
- Minimize latency obtaining high throughput.

Scheduler - Assumptions

The scheduling algorithm relies heavily on two facts:

- Processing time of each packet
  is known in advance;
- Each packet can be processed independently from the others.

Scheduling Algorithm (1)



Scheduling Algorithm (2)

- Each received packet is processed by the scheduler that:
  - selects a set of suitable processors;
  - computes the *finishing time* for each of the processors;
  - allocates the packet to the processor with lowest finishing time.

1
2
3
4
5

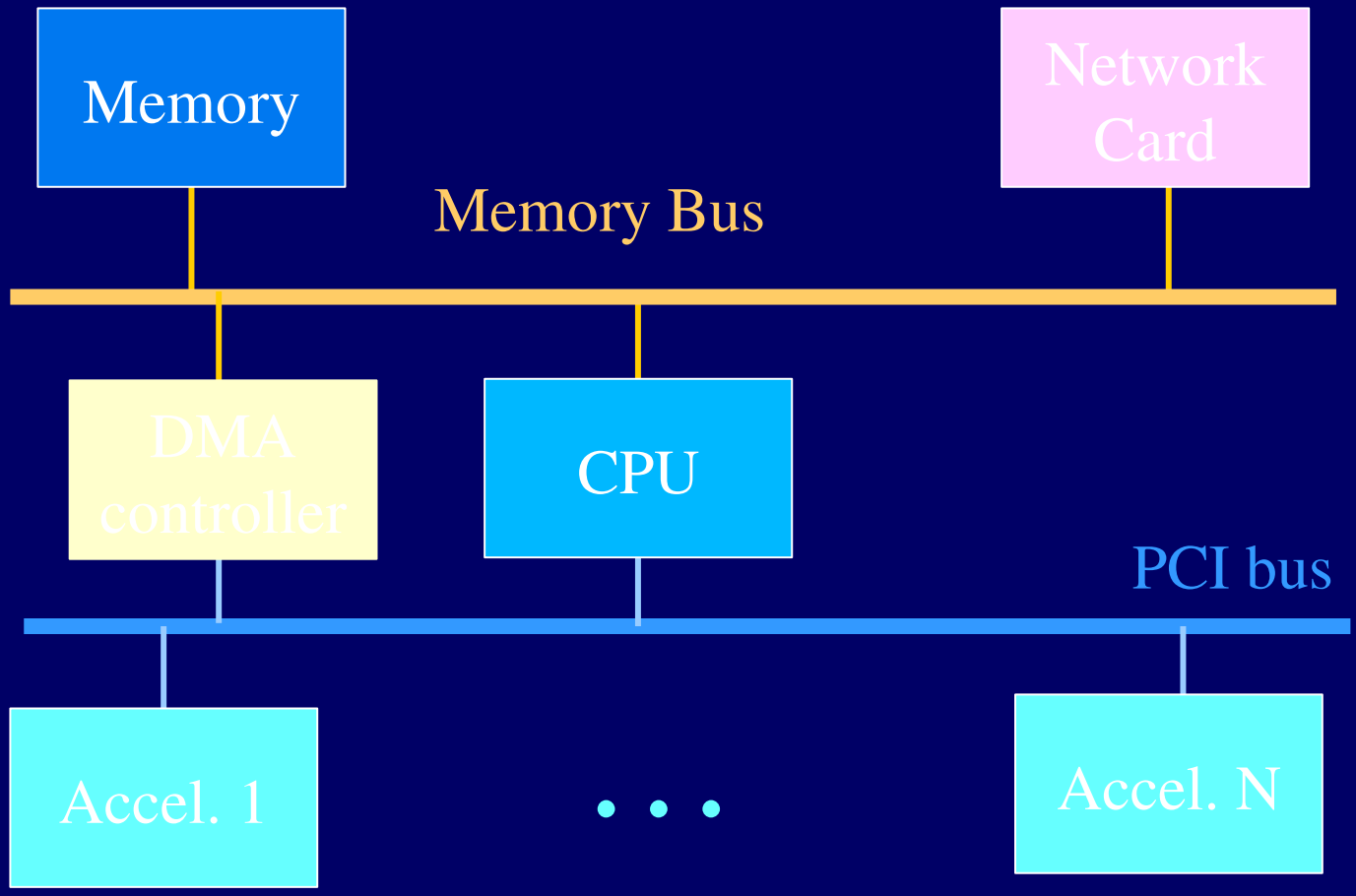
Scheduling Algorithm (3)

- *finishing time = waiting time + processing time*
- Two parameters are added
  to the formula considered for the CPU
  to allow tuning its load:
  - α_0 is a multiplicative constant;
  - β_0 is an additive constant.

Scheduler in Practice

- Each time a packet goes in or out from a processor's queue, the waiting time is updated:
  - a sum or a subtraction is only needed;
- In each scheduling operation, at most *N+1* comparisons are needed.

Reference Architecture



Data transfers to accelerators are done in DMA mode

Model For Simulations (1)

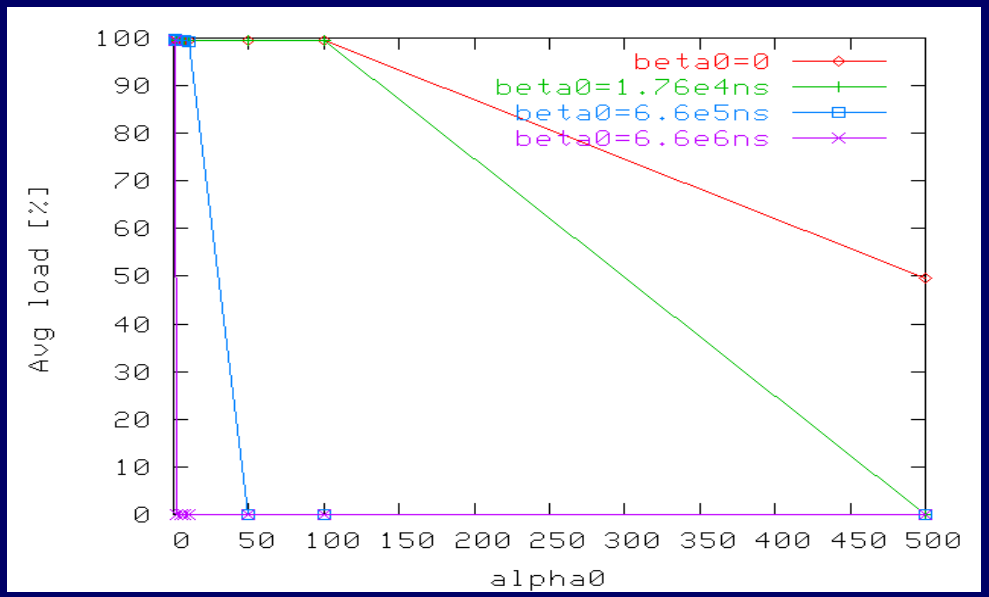
- It models the main parts of the system;
  - In accelerators AES encryption
    is only considered;
  - The only form of synchronization considered
    is bus contention:
    - Accesses to memory are faster,
    - Model not done to really measure performance;

1 2 3 4 5

Model For Simulations (2)

- It has been implemented in SystemC;
- Simulation inputs are taken from files provided on ITA website:
  - 1mln of packets were considered in each simulation

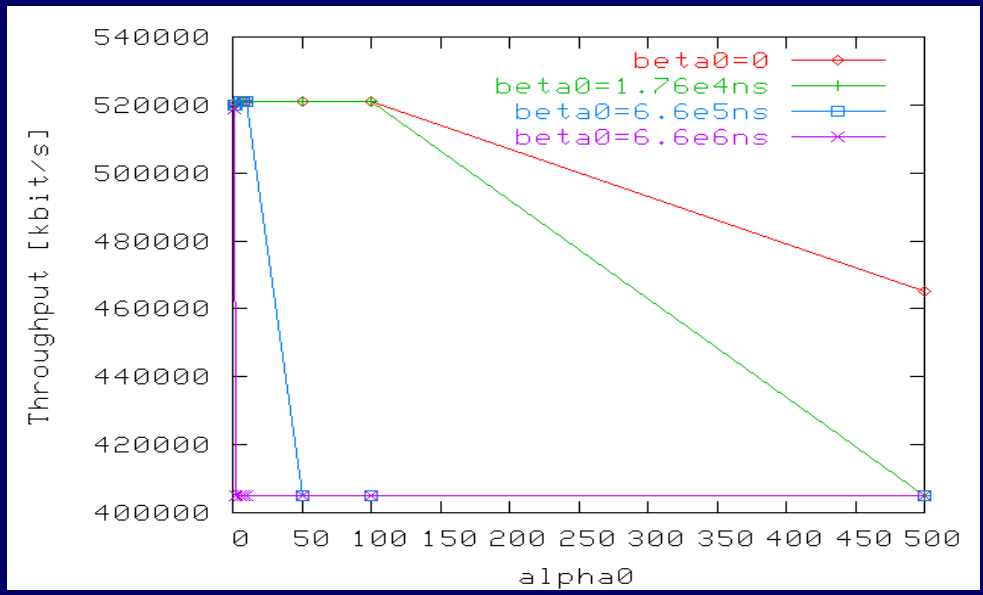
Results - CPU Load and Throughput



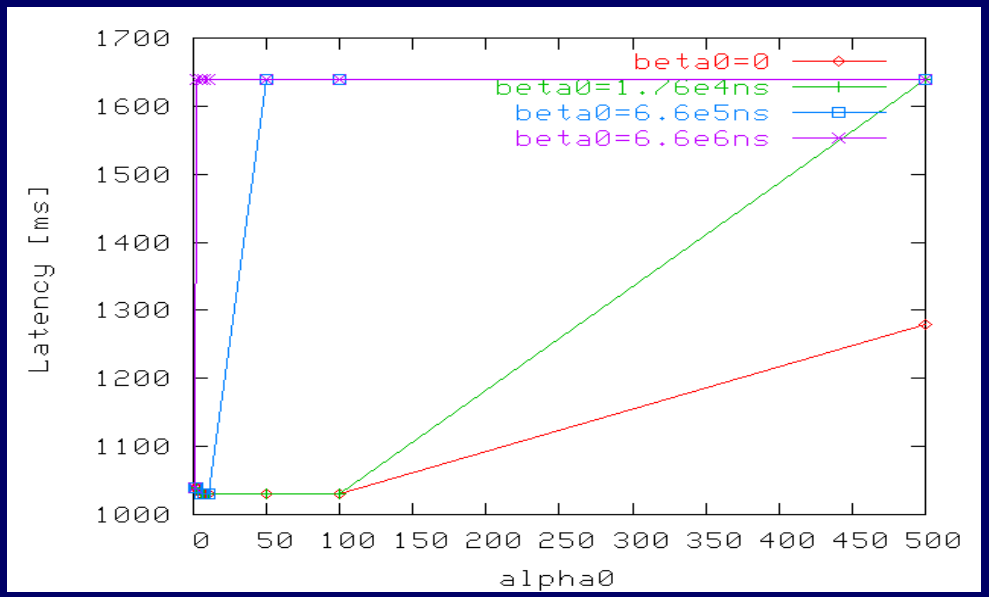
CPU load

Required bandwidth: 1Gbit/s;
Number of accelerators: 2.

Throughput



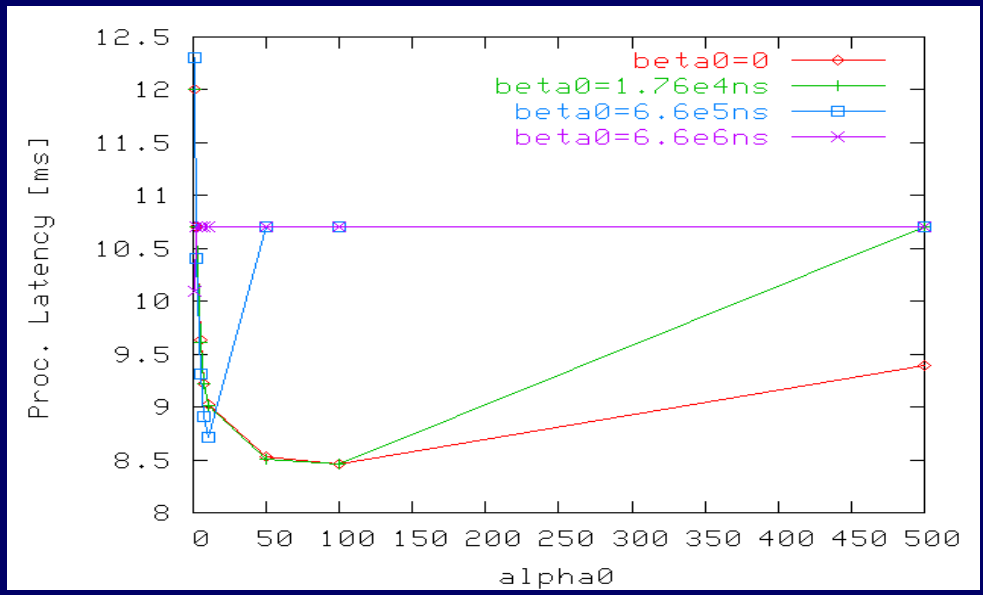
Results – Global and Processing Latency



Global Latency

Required bandwidth: 1Gbit/s;
Number of accelerators: 2.

Processing Latency



Simulations at 200Mbit/s

- All the traffic can be easily processed by one accelerator;
- The scheduling algorithm gives no benefices.

1
2
3
4
5

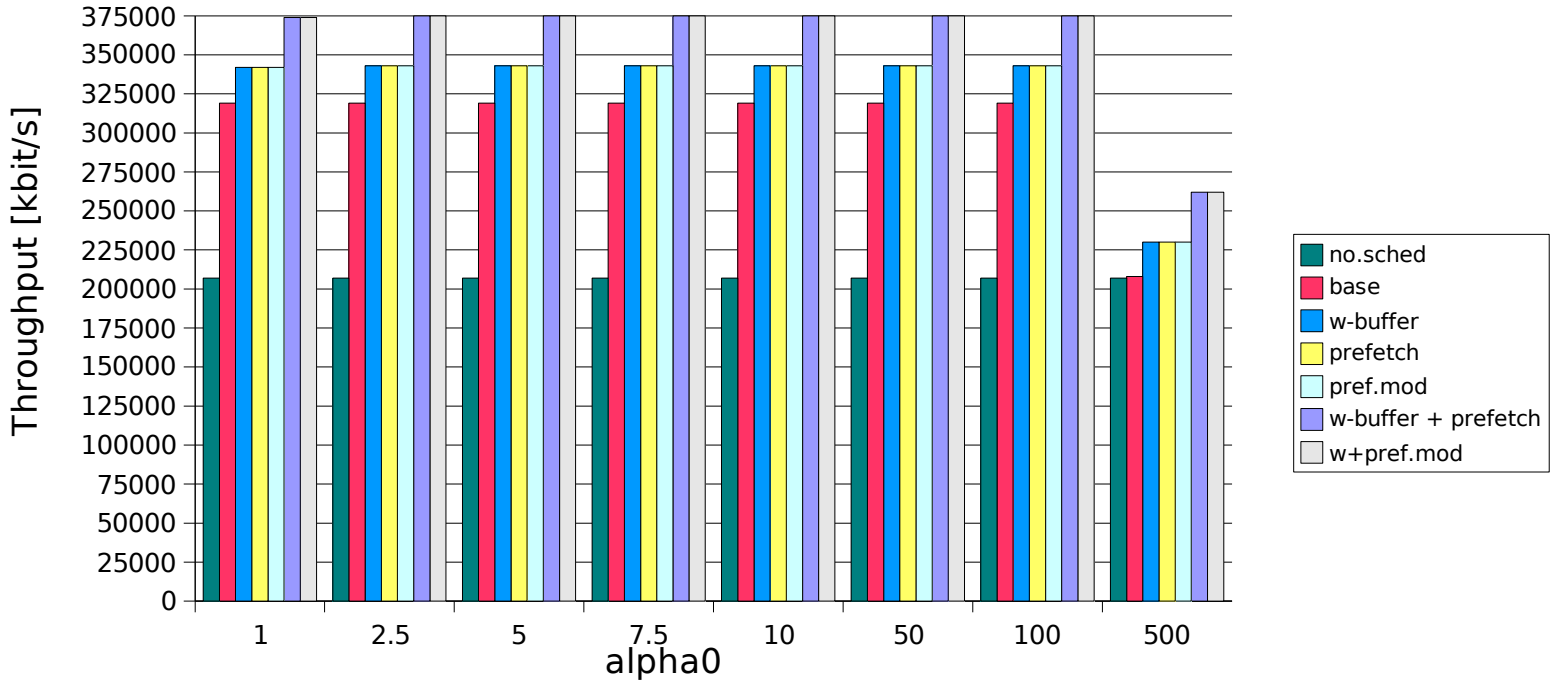
Architectural Enhancements

- Decouples input from processing;
- Prefetched data are always used.

Decouples output from processing.

1
2
3
4
5

Results - Throughput

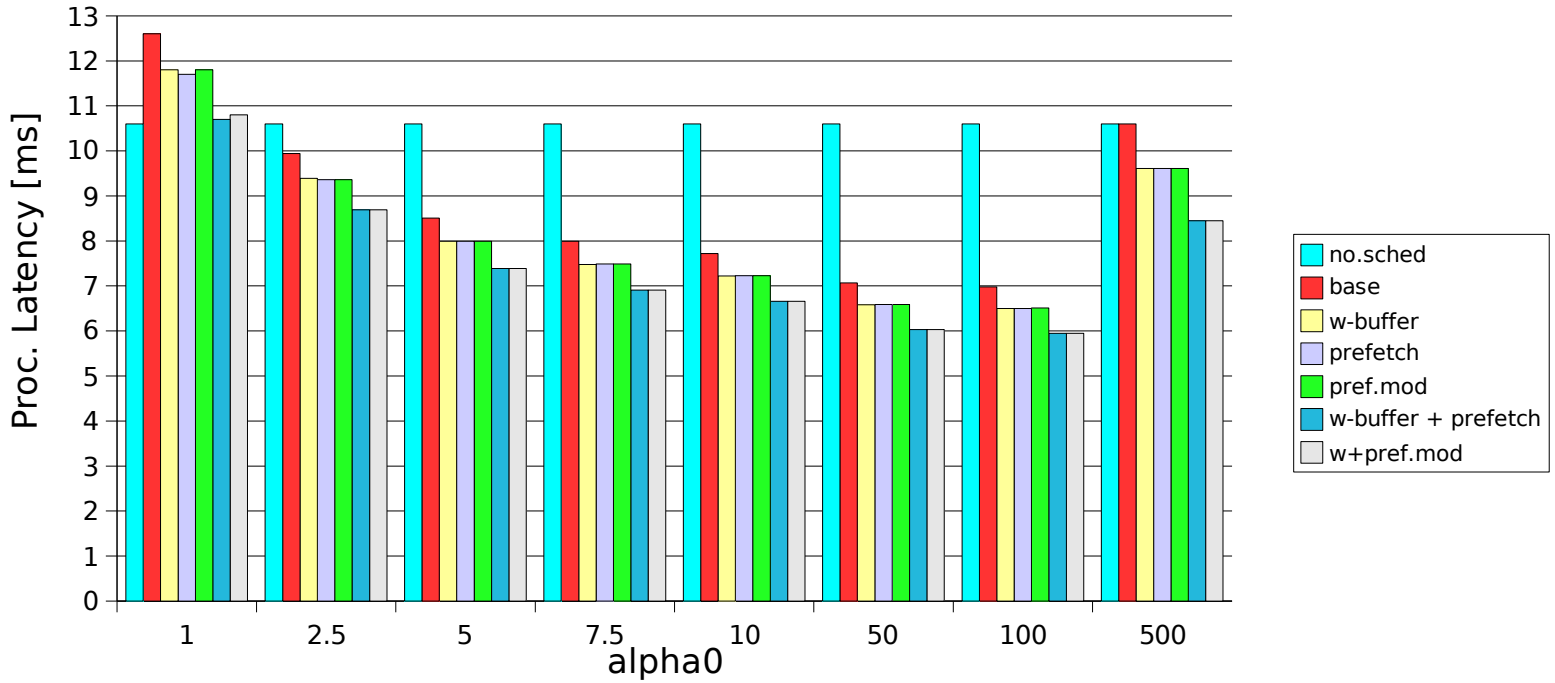


Required bandwidth: 1Gbit/s

Number of accelerators: 1

β_0: $1.76*10^4$

Results – Processing latency



Required bandwidth: 1Gbit/s

Number of accelerators: 1

β_0: $1.76*10^4$

Conclusions (1)

- We have obtained an algorithm that is able to distribute IPSec packet processing over multiple processors;
- We have shown that the algorithm works as desired.

1
2
3
4
5

Conclusions (2)

- The scheduling algorithm
  is only useful when:
  - More than one accelerator is present:
    - Having multiple accelerators
      may allow for scalability at “low” price;
  - The system is overloaded:
    - The CPU can help processing
      short peaks over the supported bandwidth.

1
2
3
4
5

Future Work

- Compare the algorithm with selected general-purpose scheduling algorithms;
- Add QoS support to the scheduling algorithm;
- Optimize the processing of “small packets”.

1 2 3 4 5