University of Castilla–La Mancha

A publication of

Department of Computing Systems

OpaSim: an OPA Simulator for High-Performance Interconnections

by

J. Cano, G. T. Fernández, F.J. Andújar,
F.J. Alfaro, J.L. Sánchez, G. Mora

Technical Report #**DIAB-18-12-1** December 2018

(*) This work has been jointly supported by the MINECO and European Commission (FEDER funds) under the project TIN2015-66972-C5-2-R, Junta de Comunidades de Castilla-La Mancha under the project SBPLY/17/180501/000498, and Intel Coorporation under the agreement CONV180068.

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS
ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE CASTILLA–LA MANCHA
CAMPUS UNIVERSITARIO s/n
02071, ALBACETE, SPAIN
Phone +34.967.599200, Fax +34.967.599224

OpaSim: an OPA Simulator for High-Performance Interconnections

Javier Cano, Guillermo T. Fernández, Francisco J. Alfaro, José L. Sánchez
Computing Systems Department
Faculty of Computer Science Engineering
University of Castilla-La Mancha
02071 – Albacete, Spain
{Javier.Cano, Guillermo.Fernandez, Fco.Alfaro, Jose.SGarcia}@uclm.es

Francisco J. Andújar-Muñoz
Computing Systems Department
Faculty of Computer Science Engineering
University of Valladolid
47011 – Valladolid, Spain
fandujarm@infor.uva.es

Gaspar Mora-Porta
Intel Coorporation
Intel Labs
Santa Clara, United States
gaspar.mora.porta@intel.com

Contents

Abstract

In the agreement between Intel and the researching group RAAP of the University of Castilla-La Mancha titled *"Advanced Routing, Congestion Control and Quality-of-Service Mechanisms in HPC Fabrics such as Intel© OmniPath©"*, the researching group committed to a study of mechanisms regarding the Quality of Service in this architecture. For that purpose, the group has developed a simulator, called *OPASim*, that serves as the basis of the research concerning the implementation of those mechanisms. This simulator, created from the public information existing regarding the Intel© OmniPath© Architecture (*OPA*), contains a series of speculations regarding the deeper implementation of the elements composing it, so that it can fulfill the requirements elicited in the public documents. For this purpose, OPASim has been developed as a modular, fast, efficient and flexible simulator capable of defining a wide variety of different topologies and configurations with a minimum effort, just by passing the necessary parameters in the configuration file. OPASim has allowed the group to pursue the investigation of Quality of Service implementation from a rather interesting angle. In this document, the different topologies and strategies followed will be explained, as well as some conclusions extracted from such data regarding the real use of OmniPath© in a physical device.

1 Introduction

The expected growth in the demand for supercomputing services will require a processing performance of 1 Exaflop in a few years. Increasing the supercomputer performance can be achieved by: a) increasing the number of cores per node without increasing its individual performance; b) increasing the number of nodes without increasing the number of cores per node; c) combining both previous options.

The interconnection network is a key element in these systems because it may become a bottleneck, degrading the performance of the entire system. Interconnection network performance depends on several factors (e.g. topology, routing, layout, etc.), which must be carefully studied by system designers prior to any implementation specially in brand new interconnection technologies such as OmniPath©.

The Intel© OmniPath© Architecture (Intel© OPA) represents a new product line aimed for high-performance interconnection networks. OPA is designed for the integration of fabric components with CPU and memory components to enable the low latency, high bandwidth and dense systems required for the Exascale generation of datacenters.

The introduction of Intel© OmniPath © Architecture product line marks one of the most significant new interconnects for high-performance computing since the introduction of InfiniBand [4, 6]. However, OPA has not been studied as much as other interconnection network technologies such as InfiniBand. One of the reasons why OPA has not been studied may be the fact that is still a very new technology. Another possibility may be the lack of tools that allow the researchers to study OPA-based systems.

Simulation is the most common technique to analyze any interconnection network architecture in a flexible, cheap and reproducible way. Regarding OPA, as far as we know, there is not any tool, simulator or simulation model publicly available. This fact slows and in some cases, hinders research tasks. In this technical report we present a simulation model and a discrete-event simulator based on the Intel© OmniPath© Architecture. We have based our work on all public documents about the architecture. However, some assumptions have been taken because some internal details remain unpublished.

The rest of this technical report is organized as follows: Section 2 gives an in depth overview of the simulation model that we have developed, and some internal details of the simulator's implementation. The latencies assumed in the model are explained in Section 2.2, while evaluation results are presented in Section 3. Finally, Section 4 shows the conclusions and future work of this technical report.

2 OPA simulator details

We have developed a discrete-event based network simulator that generates the movement of packets from source nodes to destination nodes using an OPA-based interconnection network. The main elements such as network interfaces, OmniPath routers and links, have been simulated. The tool is capable of modeling the behavior of OPA routers. Furthermore, the simulation tool offers the possibility of modifying a large range of parameters such as queue sizes, topology, routing, packet sizes, etc.

The main goal is to obtain a simulation tool as flexible as possible, following the basic behavior of OPA, aimed at performing comparative studies. The tool is already capable of running simulations using a wide variety of synthetic traffic types such as uniform, bit-reversal, bit-complement, etc., and MPI file-traced applications using the VEF framework [1]. Performance and scalability of the network will be evaluated using several metrics: throughput, end-to-end latency, network latency, etc.

In the future, the goal is to expand this simulator to be able to provide the different traffic flows with Quality of Service (QoS) following some specific criteria that should be previously defined by the Fabric Manager (FM). Those OPA elements whose behavior can be configured following different criteria will be modeled. Also, we will model and evaluate different solutions for those OPA elements whose behavior is not fixed in the public OPA documents.

2.1 Simulation model

Our first step in the simulation tool development has been to define a design approach based on the public information available in [2, 3] and to design a specific simulation tool suitable for OPA. Specifically, we have developed *OPASim*, a special-purpose simulation tool based on several of our previous simulation tools. OPASim is an event-driven, flexible, open-source, efficient and fast simulation tool that models the specific details of the OPA router with enough granularity and accuracy. We assume that each link transmits one flit per cycle. Hence, the bandwidth is defined based on the needed number of clock cycles and the flit size. Figure 1 details the 48-port router model that we have considered and implemented into the tool OPASim. For different number of ports the structure is similar.

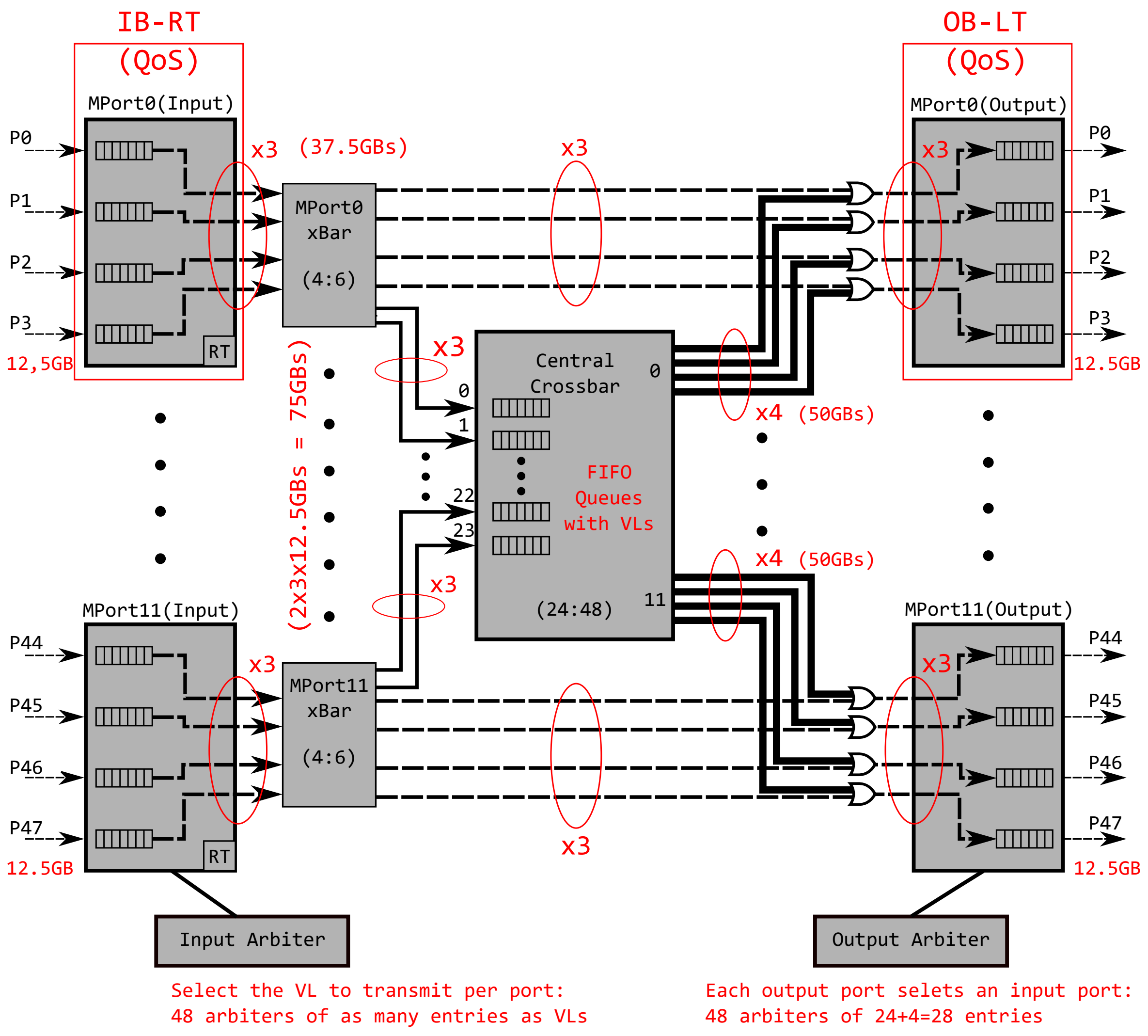

Figure 1: Diagram of the modeled OPA router of 48 ports. For clarity, MPorts are unfolded in Input and Output buffers.

Inside of an OPA router we can configure an important range of bandwidths. We have defined the input/output bandwidth (12.5 GBps) as a reference, assuming that a x3 port presents a speed-up of 3 and so it can deliver 3 flits/cycle. The number of input and output ports is represented as INPORTS:OUTPORTS in the crossbar elements (MPort xBars and Central Crossbar). For instance, in Figure 1, the MPort0 xBar has 4 input ports and 6 output ports (4:6), and the Central Crossbar is configured with 24 input ports and 48 output ports (24:48). The model assumed in this figure includes the following elements:

- Input buffers: They store the flits from the input ports. There is one input buffer per input port.
- Routing unit: There is one routing unit per input buffer.
- MPort xbar: This crossbar has 4 input ports, one per input buffer; and 6 output ports: 4 ports to output buffers and 2 ports for the Central Crossbar. Note that the 75 Gbps link to the Central Crossbar is represented in this model as 2 links. Each link presents a speed-up of x3 (it may deliver 3 flits/cycle) resulting on $2\ Links \times 3 \times (12,5\ GBps) = 75\ GBps$.
- Output buffers: They store the flits of the output ports. There is one output buffer per output port.
- Input arbiter: Given an input buffer, it selects the virtual line (VL) that participates in the next allocator phase. The more VLs, the bigger the arbiter is.
- Output arbiter: Given an output buffer, it chooses which input port is going to transmit flits. A flit can arrive at this output buffer coming from an input buffer or from the Central Crossbar.

Regarding the events used in the simulation model, OPASim defines the typical events for a pipelined switch architecture:

- IB (Input Buffering): Each input buffer can receive 1 flit/cycle. The flit arrives at an input port and is stored in the corresponding queue, depending on the VL. If that flit is the header flit, it is set as *RT-ready*, which indicates that the flit does not have the information about which path through the router should take to reach its destination, and it needs to be routed by the routing unit. If that flit is not the header flit, then it is set as *X-ready*, and is stored on the input buffer waiting to be moved to the appropriate output or Central Crossbar buffer. This grant depends on if the destination port is in the same MPort or it needs to cross the Central Crossbar. If the destination of a given flit is not in the same MPort (because that destination is not directly connected to the MPort), then it will need to cross the Central Crossbar. For example, let's suppose that in an OPA router with 48 ports and 4 ports per MPort (like the model depicted in Figure 1), a flit needs to travel from the input port 0 to the output port 5. The input port is on the MPort 0 (which contains input ports from 0 to 3) and the output port is on the MPort 1 (which contains output ports from 4 to 7), hence, the flit must cross the Central Crossbar in order to achieve MPort 1.

- RT (RouTing): If the header flit of the packet is tagged as *RT-ready*, the event performs the routing function in order to determine which output port that packet has to take to reach its destination node. After that, the header flit is tagged as *VA-SA-ready* and the input buffer which stores this flit is eligible for the VA-SA stage. Note that this event is only applied to header flits, non-header flits always follow the header flit path through the router. The routing function is configurable by the user and must be according to the configured topology.

- VA-SA (Virtual Allocator and Switch Allocator): A two-stage allocator is considered:

  - Virtual Allocator: Each input arbiter, with at least one *VA-SA-ready* header flit on it, chooses a VL that will be allowed to deliver a packet. The arbiter in our first approach is a round-robin arbiter. However, we are developing more sophisticated arbiters able to provide applications with QoS. Note that, because the Central Crossbar ports have VLs as well, this allocator stage is also performed on the input ports of the Central Crossbar.
  - Switch Allocator: Each output arbiter chooses an input buffer with a VL assigned from the previous stage. The selected input buffers will be allowed to move a packet to an output buffer or to the Central Crossbar buffer, depending on whether the requested output port is on the same MPort, as we explained before. Buffers allowed to transmit tag the top header flit as *X-ready.* A central buffer has to arbitrate between the 4 input buffers which are connected to its MPort. An output buffer has to arbitrate between the 24 Central Crossbar buffers and its 4 MPort buffers.

- X (Xbar): Once the allocation is performed, the input and Central Crossbar buffers that were selected on the VA-SA stage transmit those flits tagged as *X-ready* to the appropriate output buffer or Central Crossbar buffer. If a packet is moved from an input buffer to the Central Crossbar, the header flit is tagged again as *VA-SA-ready* in order to perform a VA-SA stage from Central Crossbar buffers to output buffers. Flits that reach the requested output buffer, are tagged as *OB-ready.* The bandwidth depends on the input/output pair. The movement from MPorts to Central Crossbar buffers can deliver 3 flits/cycle and the movement from Central Crossbar to output buffers can deliver 4 flits/cycle.

- OB (Output Buffering): Each output buffer with *OB-ready* packets chooses which VL will send flits to the neighbor router. The VL selection process is performed by a round-robin scheduler. The scheduler assigns a VL to each output buffer with *OB-ready* packets and enough credits to transmit at least one packet. If an output buffer transmits packet's last flit (i.e., the tail flit of a packet), then the output buffer releases the VL assigned by the scheduler. The strategy of releasing the assigned VL forces the scheduler to select again another VL, giving the opportunity to other VLs of being selected. Each output port can send 1 flit/cycle. At this point, QoS and packet preemption can be applied. In our first approach, the VLs are selected in a round-robin way and packet preemption is not implemented yet.

At the input, output and central buffers, VLs are dynamically managed, which means that we do not have an independent buffer per VL. Instead, the buffer storage space is divided according to the traffic requirements. We ensure a minimum and a maximum space per VL. This strategy also provides much more flexibility than to have a fixed and reserved space for each VL in buffers [7]. This flexibility is specially useful for future QoS purposes.

2.2 Stage latencies

According to the available information [2, 3], in total a packet needs 100 ns to cross the router. We have analyzed each possible source of delay in our OPA simulation model and distributed the latency among the steps that a packet must follow through the pipelined router architecture. Figure 2 shows our latency distribution among the different actions performed across simulation events that we consider a potential source of latency.

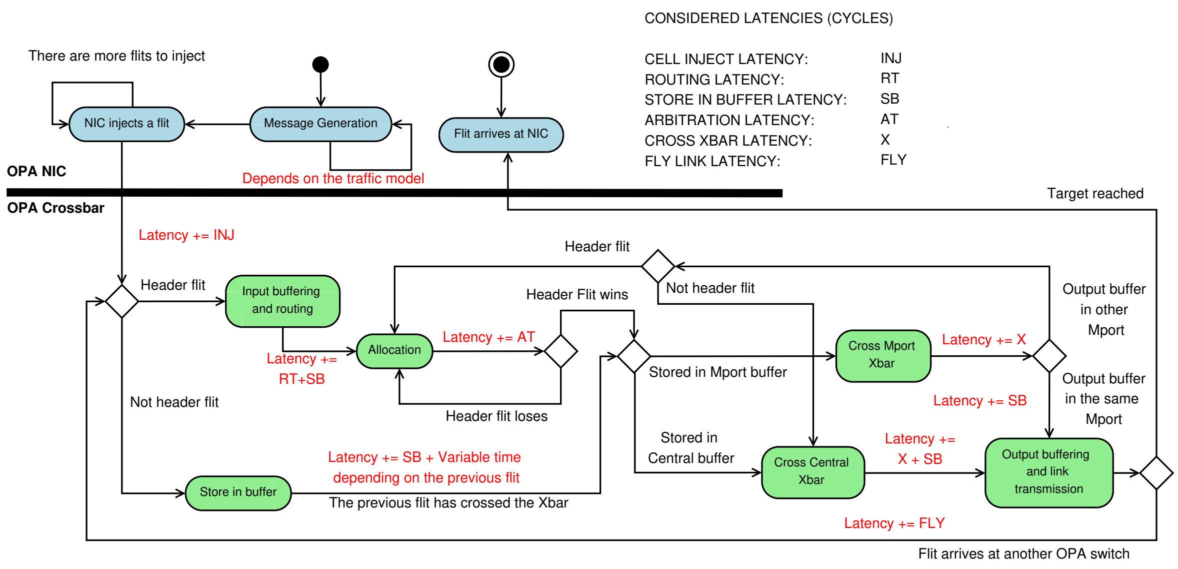

Figure 2: Latency distribution among the different simulation events.

We consider the total latency of a packet header flit from its injection to its transmission to the next router as follows:

- The source NIC injects the header flit, represented by the "NIC injects a flit" state in Figure 2 at the top left section of the figure.
- The flit adds INJ cycles to the total latency of this packet.
- After that, the header flit triggers the "Input Buffering and Routing" event. The input buffering latency is SB cycles, and the routing latency is RT cycles. At this point, the accumulated latency of the header flit is $INJ + RT + SB$ cycles.

- Then, that flit is processed by the “Allocation” event, triggered by the “Routing” event. The “Allocation” event adds latency to the header flit, since it needs to select the VL and the MPort crossbar. This is the arbitration latency AT. The total flit latency is now equal to $INJ + RT + SB + AT$ cycles.

- The next events triggered are “Cross MPort Xbar” and “Cross Central Xbar”, where the flit can be moved directly to an output buffer or first to a Central Crossbar buffer and, later, to the output port required by the header flit. If the flit travels to an output buffer, then the flit adds X cycles to the final latency. However, if the flit has to cross a Central Crossbar buffer, the flit adds $X + AT + X + SB$ cycles. This is because the header flit will have to cross its MPort, then the “Allocation” event will happen again, it will be sent from the Central Crossbar buffer to the output buffer and finally, it will be stored incoming the output buffer. In Section 2.1 (VA-SA and X events descriptions), more details about this process can be found. Hence, the flit will add $X+AT+X+SB$, cycles. Then the total latency at this point is $INJ+RT+SB+AT+X+AT+X+SB$ cycles.

- Note that there is no latency associated with the Central Crossbar buffers storage, as we consider those to be “perfect” buffers. This means that buffers do not require any additional time to store data.

- Finally, the header flit is sent to the neighbor router. Then, the latency required to go through the link is added (i.e., FLY). Therefore, the final latency is $INJ + RT + SB + AT + X + AT + X + SB + FLY$ cycles in the worst scenario without contention, which means that the flit does not have to wait to use the router resources.

It is important to note that we have taken into account the following assumptions:

- The output buffering latency includes serializer and transmission latencies.
- The input buffering latency also includes the deserializer latency.

Finally, we assume that each port has a bandwidth of 100 Gbps and the flit size is 64 bits. This means that the operational frequency is 1.6 GHz. Given that, the latency is 160 cycles in the worst case. Table 1 shows the distribution of latencies in cycles per each step considered in the router.

2.3 Default OPA simulation model configuration

OPASim allows us to define multiple simulation parameters using an input configuration file. Some parameters were already defined for other simulation tools we developed previously and they have been inherited by this simulator. However, other parameters are close related to the OPA router and they have been specifically defined for this model. The complete list of input parameters is the following:

Router Step Considered	Cycles
Routing (RT)	32
Storing in input buffer (SB)	50
Storing in output buffer (SB)	50
Arbitration (AT)	16
Crossbar MPort (X)	2
Crossbar Central (X)	2
Transmission (FLY)	8

Table 1: Latencies considered in the OPA router.

- NUMBER_OF_PORTS: It defines how many ports (input/output) the router presents. This number must be divisible by 4, because each MPort unit has 4 ports. Besides, the total number of ports must be greater than or equal to 8 so that at least two MPort units are defined. If nothing is specified for this parameter, the default value is 48 input/output ports.

- PERCENTAGE_OF_LONG_MESSAGES: Percentage of long messages injected using synthetic traffic. It must be an integer from ojete

- PERCENTAGE_OF_SHORT_MESSAGES: Percentage of short messages injected using synthetic traffic. It must be an integer from 0 to 100.

- PACKET_SIZE: This parameter sets the packet size used in the simulations measured in flits. If nothing is specified for this parameter, the default value is 16 flits.

- NUMBER_OF_VC: This parameter defines the number of virtual channels (VC) used in each link. If nothing is specified for this parameter, the default value is 8.

- QUEUE_SIZE: This parameter establishes the size of the queues used both for input and output buffers, measured in flits. If nothing is specified for this parameter, the default value is 256 flits.

- BYTES_PER_FLIT: This parameter defines the number of bytes of information stored into each flit. If nothing is specified for this parameter, the default value is 8 bytes.

- FLITS_PER_CREDIT: This value fixes the maximum number of flits that each credit consumes. If nothing is specified for this parameter, the default value is 4 flits per credit.

- EXCLUSIVE_VC_CREDITS: This parameter defines how many credits are reserved for each VL. If nothing is specified for this parameter, the default value is 16 credits reserved for each VL.

- MAX_VC_CREDITS: This value establishes the number of credits that a VL can consume. If nothing is specified for this parameter, the default value is 48 credits per VL.
- SHORT_MESSAGE_SIZE: This parameter sets the size of the short messages, measured in flits.
- LONG_MESSAGE_SIZE: This parameter sets the size of the long messages, measured in flits.
- PERCENTAGE_SHORT_MESSAGES: This parameter sets the percentage of short messages injected using synthetic traffic. It is an integer from 0 to 100. Note that there is not a PERCENTAGE_LONG_MESSAGES parameter, but it is computed automatically as (100 - PERCENTAGE_SHORT_MESSAGES)

Note that this default values have been chosen according to the first generation of OmniPath routers. We have developed this simulator to be as flexible as possible in order to be able to simulate OmniPath scenarios from future generations.

3 Evaluation Results

In this section, we present some preliminary results obtained with the OPASim simulation tool described in Section 2. We have analyzed the performance of a single router with 48 nodes connected, using uniform synthetic traffic with variable injection rate. For these simulations, we have run 30 different experiments varying the seed of the random numbers generation, and the value shown in the figures is the average of the obtained results.

Note that the goal of this preliminary evaluation is not to provide applications with QoS (which is our final target), but just to check if the initial model that has been implemented is correct and offers the expected behavior under different scenarios.

Figure 3 shows the throughput of that configuration. The maximum throughput is 0.72 flits/cycle/NIC with an injection rate of 1 flit/cycle/NIC. This throughput does not reach 100%, which is the theoretical maximum, as usual in this kind of studies both simulated and in the real product. This is because of multiple factors, such as HoL blocking, internal contention, etc.

Figure 4 shows the obtained network latency. The packet latency increases in an exponential way until the congestion point is reached at 0.8 flits/cycle/NIC. After this point, latency stabilizes.

Note that these results could be vastly improved with a better allocator. As we mentioned above, the allocator developed in the first version is a very simple one. We are developing and testing the three-phase iSLip allocator [5], which is much more efficient, and which we expect that will drastically improve the performance.

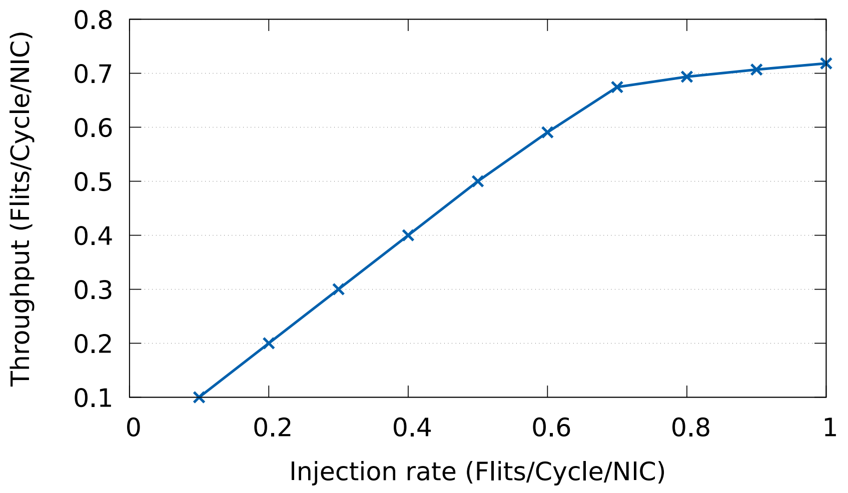

Figure 3: Network throughput for a single OPA router with 48 nodes connected using uniform synthetic traffic with contention.

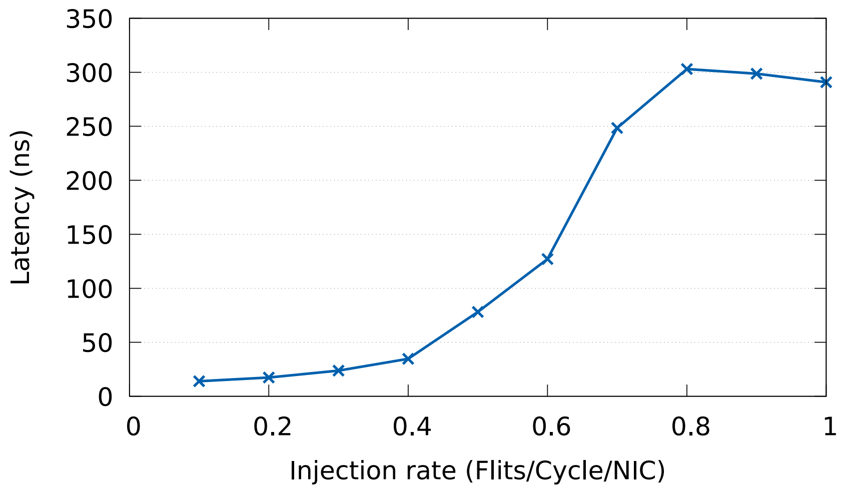

Figure 4: Network latency for a single OPA router with 48 nodes connected using uniform synthetic traffic.

Finally, we have also evaluated the performance under no contention. In this scenario, a NIC with ID x will send packets to another NIC with ID y, being $y = (x+1)\%NumTotalNICs$, and $NumTotalNICs$ the total amount of NICs in the simulation scenario. In that way, a node is sending traffic only to the consecutive node in ID order. As in this example $NumTotalNICs = 48$, the target node for NIC x in this experiment will be selected following $y = (x+1)\%48$. Using this traffic pattern, messages arrive at their destination without having to compete with other messages for the router resources. This happens because each source NIC always chooses a single destination and the allocator can always connect each input port with an output port or Central Crossbar, in the first attempt. This means that packets ready for the next event do not have to wait in order to get the allocator grant. Figure 5 shows the throughput of this simulation model, which verifies that in this scenario the obtained throughput matches the expected theoretical one. This means that the simulated router model is working as expected.

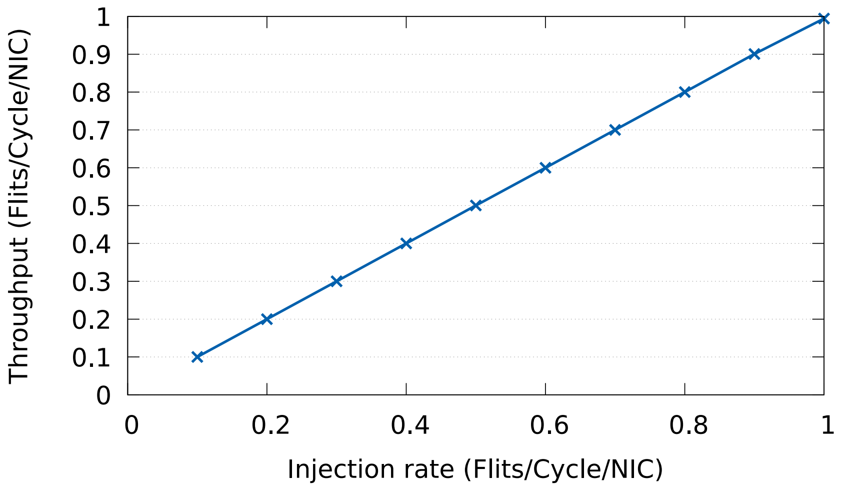

Figure 5: Network throughput for a single OPA router with 48 nodes connected and without contention, using synthetic traffic.

4 Conclusions and Future Work.

This technical report includes all the developed work during the first phase of the agreement with Intel Coorporation©. During this time, the OPA public information has been studied in order to understand the mechanisms that OPA offers so as to provide applications with QoS. At the same time, we have developed an OPA simulation tool that we have called *OPASim*. OPASim has been developed to model the main characteristics of an OPA router. This tool has been thoroughly tested, in order to confirm that its behavior corresponds to the theoretical behavior of OPA devices. Obtained results confirm that the simulator presents the expected behavior.

Once this phase is complete, as future work, we plan to continue advancing in the OPASim development based on two directions. On the one hand, we will include in the simulator the different features that OPA supplies to provide applications with QoS. These mechanisms should be tested, together with other variants of them, so as to compare different alternatives. On the other hand, we will select specific applications demanding QoS provision. Traces of these applications must be collected to feed OPASim for a later study of the behavior of the modeled QoS mechanisms.

References

[1] Francisco J Andújar, Juan A Villar, José L Sánchez, Francisco J Alfaro, and Jesus Escudero-Sahuquillo. Vef traces: a framework for modelling mpi traffic

in interconnection network simulators. In *Cluster Computing (CLUSTER), 2015 IEEE International Conference on*, pages 841–848. IEEE, 2015.

[2] Mark S Birrittella, Mark Debbage, Ram Huggahalli, James Kunz, Tom Lovett, Todd Rimmer, Keith D Underwood, and Robert C Zak. Intel® omni-path architecture: Enabling scalable, high performance fabrics. In *High-Performance Interconnects (HOTI), 2015 IEEE 23rd Annual Symposium on*, pages 1–9. IEEE, 2015.

[3] Mark S Birrittella, Mark Debbage, Ram Huggahalli, James Kunz, Tom Lovett, Todd Rimmer, Keith D Underwood, and Robert C Zak. Enabling scalable high-performance systems with the intel omni-path architecture. *IEEE Micro*, 36(4):38–47, 2016.

[4] Michael Feldman and Addison Snell. A new high performance fabric for hpc. *Intersect360 Research white paper, May*, 2016.

[5] Nick McKeown. The islip scheduling algorithm for input-queued switches. *IEEE/ACM Transactions on Networking (TON)*, 7(2):188–201, 1999.

[6] Gregory F Pfister. An introduction to the infiniband architecture. *High Performance Mass Storage and Parallel I/O*, 42:617–632, 2001.

[7] Yuval Tamir and Gregory L Frazier. Dynamically-allocated multi-queue buffers for vlsi communication switches. *IEEE Transactions on Computers*, 41(6):725–737, 1992.