Fast Forwarding with Network Processors

Laurent Lefèvre, E. Lemoine, Cong-Duc Pham, B. Tourancheau

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Fast Forwarding with Network Processors

L. Lefèvre — E. Lemoine — C. Pham — B. Tourancheau

N° 4710

Janvier 2003

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4710--FR+ENG

Fast Forwarding with Network Processors

L. Lefèvre* , E. Lemoine*** , C. Pham* , B. Tourancheau**

Thème 1 — Réseaux et systèmes
Projet INRIA/RESO

Rapport de recherche n° 4710 — Janvier 2003 — 14 pages

Abstract: Forwarding is a mechanism found in many network operations. Although a regular workstation is able to perform forwarding operations it still suffers from poor performances when compared to dedicated hardware machines. In this paper we study the possibility of using Network Processors (NPs) to improve the capability of regular workstations to forward data. We present a simple model and an experimental study demonstrating that even though NPs are less powerful than Host Processors (HPs) they can forward data more efficiently than HPs in some specific cases.

Key-words: Network processors, workstation, forwarding

* INRIA/RESO
** Sun Labs Europe

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 —Télécopie +33 4 76 61 52 52

Forwarding efficace sur processeur réseau

Résumé : Faire suivre des données (*forwarding*) est une opération fréquente dans les réseaux actuels. Même si une station de travail conventionnelle est capable d'effectuer cette opération de forwarding, les performances obtenues par ces dernières sont loin de celles des machines spécialisées. Dans ce rapport nous étudions la possibilité d'utiliser des processeurs réseau au niveau des cartes d'interface réseau afin d'augmenter le niveau de performance des stations de travail pour le forwarding de données réseau. Nous présentons un modèl simple et une étude expérimentale démontrant que même si les processeurs réseau actuels sont moins performants que les processeurs généraux des stations de travail ils sont capables, dans certaines conditions, d'opérer le forwarding plus rapidement que les processeurs généraux.

Mots-clés : Processeurs réseau, station de travail, *forwarding*

1 Introduction

The forwarding operation consists of receiving data from the network, performing some computation (eventually mangling the data) and sending the data back to the network. It is a very common network operation found in many mechanisms: routing, address translating (NAT), protocol translating. Specific hardware machines can achieve forwarding at very high speed, nevertheless, because of cost consideration, making use of off-the-shelf components —workstation PCs for example— remains attractive. In addition, one other motivation behind PC-based forwarding machine is flexibility. However, using workstation PCs instead of dedicated hardware machines raises performance issues: despite of their increasing power current general-purpose processors are not able to face today's network throughput. Elevating the forwarding performances of PCs to those of dedicated machines remains a great challenge.

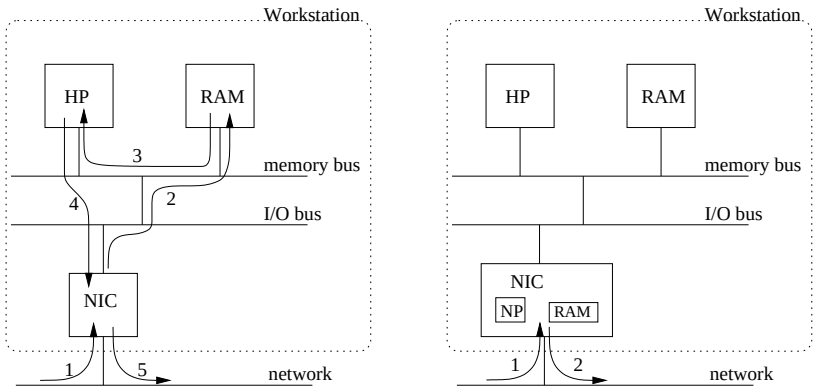
A workstation forwards data as follows (figure 1(a)): The Network Interface Card (NIC) pulls the data in from the network, copies it in host memory (through the Input/Output bus) and interrupts the host processor to notify it of new incoming data. If the data is destined to another node in the network the Host Processor (HP) will perform the forwarding algorithm (routing algorithm for example) and transfer the data back to the NIC (through the I/O bus again). These operations cost a lot in terms of performance.

One alternative is to forward data directly from the NIC, without passing by the HP (figure 1(b)). This solution has three advantages:

- the data-path is shortened because the forwarded data is no longer being transfered through the I/O bus;
- costly context switchings due to interrupts are avoided;
- if the network is full-duplex then it is possible to start re-sending the message before it has been entirely received in the NIC's memory (cut-through technique similar to the Frame Relay strategy on long distance networking).

Today's ASIC-based NICs cannot forward data without passing by the HP, they are designed to always interrupt the HP regardless of the type of the incoming data. Some recent NICs (e.g. [Cor00], [Ram00]) embed a so-called Network Processor (NP) which is responsible of controlling the input and output channels. These NPs are fully programmable thus allowing, according to some specific need, third-party to modify the NIC's behavior.

Techniques for improving workstation-based forwarding equipments already exist (see [YC00] and [WHT98]), they mainly take advantage of the large amount of memory present on the Network Interface Card. Our objective is to also take advantage of the processing power present on the NIC and then move protocols (or part of them) onto the NIC. Prior to the attempt of off-loading complex functions onto the NIC it is necessary to determine more precisely what can effectively be off-loaded. This paper focuses on the forwarding operation and tries to determine if the use of network processors for forwarding data (*NP forwarding*) can lead to better performances than classical forwarding schemes using the host processors (*HP forwarding*). Given that today's NPs are much slower than HPs we try here to evaluate



(a) *HP Forwarding* (b) *NP Forwarding*

Fig. 1. *The different steps in the forwarding operation.*

how much computation the NP can handle while keeping NP forwarding performances above those of HP forwarding. Towards this end we have built a simple model of the forwarding operation and validated this model with a software prototype based on Myrinet/LANai NIC ([BCF95]).

This paper is outlined as follows: section 2 presents a simple model of the forwarding operation and gives some results. Section 3 describes our prototype implementation with the experiences we have conducted and gives some results corroborating the results of the model. Section 4 presents some related works and section 5 concludes and announces future works.

2 Forwarding model

This section presents a simple forwarding model to figure out if NP forwarding can lead to better performances than HP forwarding.

2.1 Notations

Let T_N and T_H be the times elapsed in the forwarding node for a ℓ-byte message for, respectively, NP forwarding and HP forwarding. T_N and T_H are the two parameters we want to compare in this study. The other parameters introduced by the model are presented in table 1. In order to pipeline all the stages of a message transfer, most of communication protocols cut out big messages in packets. $nb(\ell)$ and $sz(\ell)$ characterize this cutting. In the

Parameter	Description
$T_{rcv}(n)$	time to receive n bytes from network to NIC
$T_{snd}(n)$	time to send n bytes from NIC to network
$T_{pci}(n)$	time to transfer n bytes on the I/O (PCI) bus
T_{algoH}	algorithm's execution time on the HP
T_{algoN}	algorithm's execution time on the NP
$nb(\ell)$	number of packets in a ℓ-byte message
$sz(\ell)$	packet size (in bytes) for a ℓ-byte message

Table 1. *Parameters of the forwarding model*

following we assume that the message size is a multiple of the packet size, therefore we have $\ell = nb(\ell) \times sz(\ell)$. For non cut messages, $sz(\ell) = \ell$ and $nb(\ell) = 1$.

2.2 Model

The HP forwarding time for a ℓ-byte message is given by:

$$\begin{aligned} T_H = {} & T_{rcv}(\ell) \quad \text{overlapped with} \quad T_{pci}(\ell) \\ & + T_{algoH} \\ & + T_{rcv}(\ell) \quad \text{overlapped with} \quad T_{snd}(\ell) \end{aligned}$$

The term "*overlapped* with" indicates that the transfer times between the NIC and the network, and between the NIC and the Host overlap (transfers are pipelined). The following expression gives T_H as a function of the pipeline characteristics ($nb(\ell)$ and $sz(\ell)$):

$$\begin{aligned} T_H = {} & T_{rcv}(sz(\ell)) \\ & + (nb(\ell) - 1) \times \max\left(T_{rcv}(sz(\ell)), T_{pci}(sz(\ell))\right) \\ & + T_{pci}(sz(\ell)) \\ & + T_{algoH} \\ & + T_{pci}(sz(\ell)) \\ & + (nb(\ell) - 1) \times \max\left(T_{pci}(sz(\ell)), T_{snd}(sz(\ell))\right) \\ & + T_{snd}(sz(\ell)). \end{aligned}$$

We assume that the times to receive from the network to the NIC and to send from the NIC to the network are equal; we note this time T_{net}. Thus, for a $sz(\ell)$-byte packet we have $T_{net}(sz(\ell)) = T_{rcv}(sz(\ell)) = T_{snd}(sz(\ell))$. We also assume that the I/O bus is over-dimensioned compared to the network; we have then $\max\left(T_{rcv}(sz(\ell)), T_{pci}(sz(\ell))\right) = T_{rcv}(sz(\ell))$ and $\max\left(T_{pci}(sz(\ell)), T_{snd}(sz(\ell))\right) = T_{snd}(sz(\ell))$. This assumption is true for recent PCI bus technology and will remain true for next I/O interconnect technologies (PCI

X and Infiniband for instance). By considering these assumptions the HP forwarding time for a ℓ-byte message becomes:

$$T_H = 2\,nb(\ell) \times T_{net}(sz(\ell)) + 2\,T_{pci}(sz(\ell)) + T_{algoH}.$$

For NP forwarding two cases must be considered. The first case is when the NIC receives the entire message in its memory before sending it back to the network (*store-and-forward* case). For example, this case occurs when the checksum of the message must be calculated and inserted in the message header. The second case is when the beginning of the message can be sent back to the network before the message has been entirely received in the NIC memory (*cut-through* case). For the store-and-forward case, T_N is given by:

$$T_N = 2\,nb(\ell) \times T_{net}(sz(\ell)) + T_{algoN}.$$

For the cut-through case, it is given by:

$$T_N = (nb(\ell) + 1) \times T_{net}(sz(\ell)) + T_{algoN}.$$

2.3 Results

$(T_H - T_N)$ is the quantity that allows us to determine whether it is beneficial to do NP forwarding or not. If $(T_H - T_N)$ is positive then NP forwarding performs better than HP forwarding, otherwise HP forwarding performs better.

Store-and-forward case

$$T_H - T_N = T_{algoH} + 2\,T_{pci}(sz(\ell)) - T_{algoN}.$$

It turns out that the store-and-forward case is not very favorable to the NP forwarding strategy: the packet size, $sz(\ell)$, being small, the quantity $T_{pci}(sz(\ell))$ is also small. Therefore few headroom remains for the execution of the algorithm. Let δ the quantity $T_{algoN} - T_{algoH}$. We can deduce the δ for which we have $T_H = T_N$, we note it δ^*.

$$T_H - T_N = 0 \qquad \Longleftrightarrow \qquad \delta^* = 2\,T_{pci}(sz(\ell)).$$

If the algorithm is such that $T_{algoN} - T_{algoH} > \delta^*$ then HP forwarding performs better than NP forwarding, otherwise the latter performs better.

Cut-through

$$\begin{aligned} T_H - T_N = {} & T_{algoH} \\ & + 2\,T_{pci}(sz(\ell)) \\ & + (nb(\ell) - 1) \times T_{net}(sz(\ell)) \\ & - T_{algoC}. \end{aligned}$$

Parameter	Value
ℓ	8192 Bytes
$sz(\ell)$	2048 Bytes
$nb(\ell)$	4
$T_{pci}(sz(\ell))$	$11\mu s$
$T_{net}(sz(\ell))$	$15\mu s$

Table 2. *Value of the input parameters for 8192-byte messages.*

Parameter	Value
δ^* (store-and-forward)	$22\mu s$
δ^* (cut-through)	$67\mu s$

Table 3. *Results for 8192-byte messages.*

For the cut-through technique δ^*, introduced previously, is given by:

$$T_H - T_N = 0 \qquad \Longleftrightarrow \qquad \delta^* = 2\,T_{pci}(sz(\ell)) + (nb(\ell) - 1)\,T_{net}(sz(\ell)).$$

δ^* in the cut-through case is greater than previously for the store-and-forward case. This result shows that the cut-through case is much more favorable to NP forwarding than the store-and-forward case.

2.4 Example

We present here the results of our simple model for messages of 8192 bytes; we will later (section 3.4) compare these results to those of the experimental study. Table 2 summarizes the values used for the input parameters. $sz(\ell)$ and $nb(\ell)$ depends on the communication system; the values of $sz(\ell)$ and $nb(\ell)$ given in table 2 are those of the BIP communication system ([PT98]), we will say more about BIP in section 3. $T_{pci}(sz(\ell))$ and $T_{net}(sz(\ell))$ have been measured on a Pentium III PC (600Mhz) with a LANai9.2 (200Mhz) Myrinet NIC and a 64bits/66Mhz PCI bus.

Table 3 gives the results of the model for the store-and-forward and cut-through cases. The two values are calculated using the expressions of δ^* derived previously. For the store-and-forward technique, the simple model says that if δ (the difference between the execution time of the forwarding algorithm when executed on the NP and the execution time of the forwarding algorithm when executed on the HP) is greater than $22\mu s$ then HP forwarding gives better performance than NP forwarding. Under this threshold NP forwarding performs better. For the cut-through technique, the model says that NP forwarding leads to better performance for a δ up to $67\mu s$. These numerical results confirm the fact that the cut-through technique leads to much better performances than the store-and-forward technique.

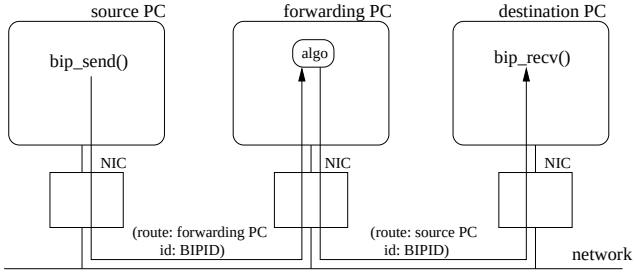


Fig. 2. *Data path for HP forwarding.*

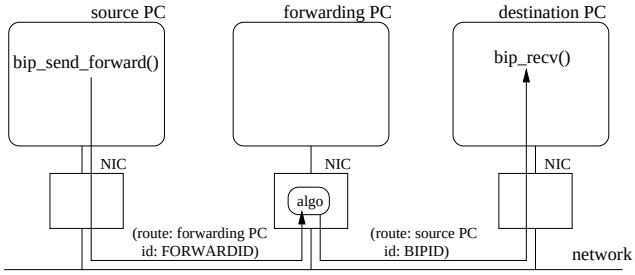


Fig. 3. *Data path for NP forwarding.*

3 Experimental study

This section presents the experimental results when comparing NP forwarding and HP forwarding implementations. It begins by presenting the testbed configuration and by describing the conducted experiences and the software prototype we have realized. It finally compare the experimental results to those provided by the simple model.

3.1 Testbed configuration

The testbed consists of a Myrinet-based network and three Pentium III (600Mhz) PCs with LANai9 Myrinet NIC (200Mhz/2Mbytes). HP forwarding has been implemented in a user-level program that makes direct calls to send and receive functions of the BIP communication library [PT98]. Briefly, BIP is a user-level communication library for Myrinet network that provides to the user most of the Myrinet network's raw performance. BIP is composed of a linux driver, an embedded code (executed by the LANai NP located on the Myrinet NIC) and a user-level library. The API provides very basic message passing primitives (`bip_send`, `bip_recv`) and is usually not intended to be used by the end-user but rather by a middleware

such as MPI [For95]. There are multiple reasons for using BIP. First, its source code is freely available. Second, since BIP is interrupt free and since its associated overhead is almost negligible, comparing NP forwarding performances and HP forwarding performances using BIP really gives insight in the amount of time that is spent in data transfers over the I/O bus. Therefore the obtained results gives us a lower bound of the possible benefits of off-loading the forwarding operation onto the Network Interface Card.

3.2 Description of the experiences

The experiences have been realized on the three PCs of the testbed: a source PC, a forwarding PC and a destination PC. Messages are sent from the source PC, passed by the forwarding PC and received by the destination PC. After receiving all the messages the destination PC sends a message back to the source PC to indicate the completion of the test (see figures 2 and 3).

Each message is processed —through the forwarding algorithm— at the forwarding PC before being sent out to the network. All experiences consist of comparing NP forwarding and HP forwarding. With HP forwarding, the entire message must be received before starting to send it back to the network since the PCI bus is half-duplex (hence, this case corresponds to a store-and-forward operation). With NP forwarding, this constraint does no longer exist: the Myrinet network is full-duplex so the NIC is capable of sending and receiving simultaneously. Since messages are cut into packets, it is possible to start re-sending them to the network before they have been entirely received (cut-through technique). As the cut-through technique is not always applicable we will also look at the NP forwarding store-and-forward technique (messages are entirely received before being sent back).

The total elapsed time is measured at the source PC: the source counts the number of real time clock cycles (by reading into the RTC register) between the sending of 10000 messages and the reception of the final acknowledgment message. The latency is equal to the total elapsed time (converted in microseconds) divided by 10001 (to obtain the latency for one message). The bandwidth is obtained by dividing the message size by the latency.

3.3 Prototype implementation

To implement NP forwarding we have modified two components of BIP: the user-level library and the embedded code. A new primitive `bip_send_forward` has been added in the API. Like `bip_send`, this primitive is called with three arguments: the destination node's logical number, the data-buffer's memory address (the actual message) and the size of the data. A message sent with `bip_send_forward` differs from a message sent with `bip_send` in two ways: it transitates by the forwarding node —which logical number is fixed— and it holds a particular identifier (`FORWARDID`). The NIC embedded code has been modified so that it can differentiate classical BIP messages from messages holding the `FORWARDID` identifier. When a message sent with `bip_send_forward` arrives at the forwarding node, it is recognized by the NIC and handled accordingly. After processing, the NIC sends the message back to the network towards the final destination.

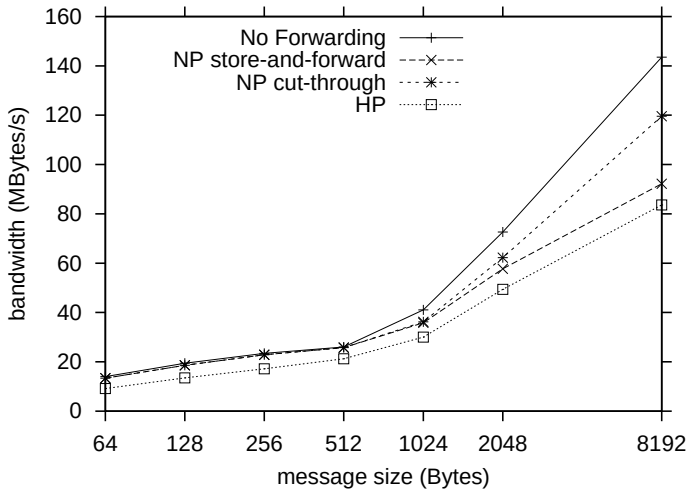


Fig. 4. *Bandwidth vs message size for the Basic Forwarding Algorithm.*

In the following we present experimental results for NP forwarding (for both store-and-forward and cut-through techniques) and for HP forwarding and make comparisons between the two.

3.4 Results and test-case

We consider two types of forwarding algorithm: the first one, that we name *Basic Forwarding Algorithm*, does not perform any computation except modifying the route and the identifier in each message header (which is mandatory to route each message towards the final destination). The second, that we name *Compute-Intensive Forwarding Algorithm*, consists in not only modifying the route and the identifier in each message header but also reading, performing some computation and writing some bytes in each message.

Basic Forwarding Algorithm Figure 4 shows that for the store-and-forward technique, NP forwarding gives better performances than HP forwarding even though the two bandwidth curves stick together. The reason is that data transfers between the network and the NIC, and between the NIC and the host are pipelined for the HP forwarding case. Since the I/O bus (PCI) is over-dimensioned in our tested, PCI transfer times become almost negligible. When using the cut-through technique, NP forwarding leads to much better performances than HP forwarding. The cut-through curve is very close to the no-forwarding curve which corresponds to the case where messages are sent from the source to the destination without going through the forwarding node. When using this cut-through technique, forwarding can

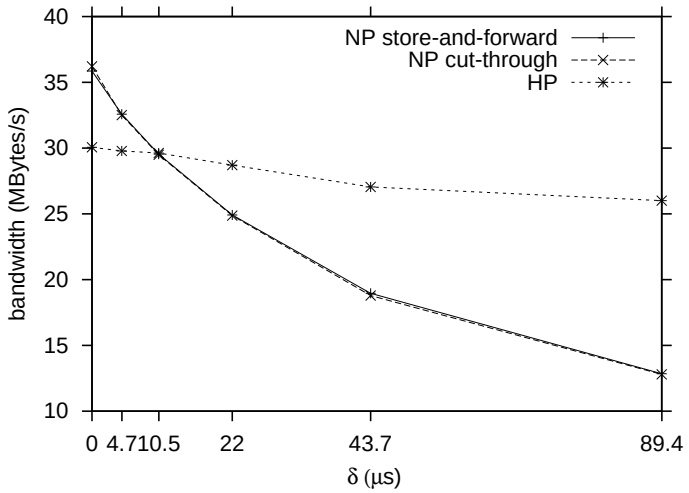


Fig. 5. *Bandwidth vs difference between the execution time of the forwarding algorithm when executed on the NP and the execution time of the forwarding algorithm when executed on the HP (δ) for 1024-byte messages.*

be achieved almost at the wire speed. These observed results corroborate the model's results shown in section 2.

Compute-Intensive Forwarding Algorithm We use two message sizes, 1024 bytes (figure 5) and 8192 bytes (figure 6), for the experimental study. The bandwidth curves in both figures are plotted versus δ, δ being the difference between the execution time of the forwarding algorithm when executed on the NP and the execution time of the forwarding algorithm when executed on the HP.

Figure 5 shows, for 1024-byte message, that if δ is greater than about $10\mu s$ then HP forwarding performs better than NP forwarding. This is true both for the store-and-forward and the cut-through cases. For 1024-byte messages, the cut-through technique does not lead to better performances than the store-and-forward technique.

For 8192-byte messages (figure 6), the cut-through technique takes effect and thus gives better performance results than the store-and-forward technique. With the latter, if δ is greater than about $19\mu s$ then HP forwarding performs better than NP forwarding. However, with the cut-through technique, NP forwarding leads to better performances than HP forwarding for a δ up to $65\mu s$. It is to be noted that these results are very close to those of our forwarding model (see section 2.4).

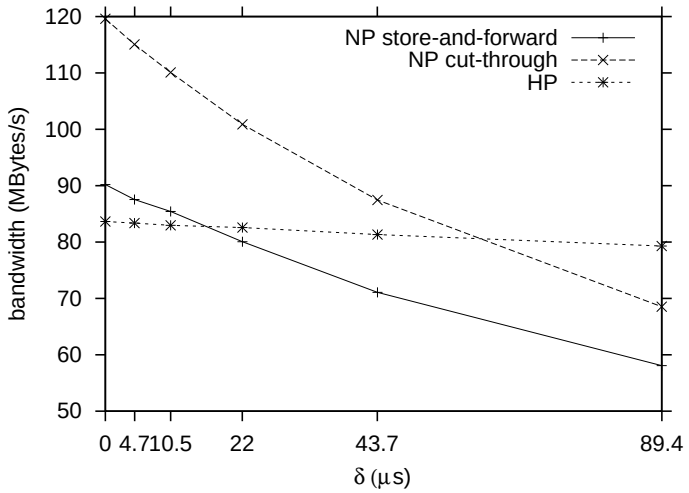


Fig. 6. *Bandwidth vs difference between the execution time of the forwarding algorithm when executed on the NP and the execution time of the forwarding algorithm when executed on the HP (δ) for 8192-byte messages.*

4 Related work

To our knowledge evaluating and modeling the possible performance gains that can be obtained using Network Processors for simple networking operations have not been studied in the literature yet. The related research works described below experiment Network Processors for specific tasks.

K. Yocum and al. have developed Payload Caching [YC00], a technique that utilizes the memory of the NIC for caching data This work is related to ours in the sense that it aims to enhance workstation-based forwarding equipments. However, Payload Caching uses a technique completely different from ours: in Payload Caching, the NIC caches a portion of incoming messages (the payload) before transferring them up to the host's memory. Then, if the HP decides to forward the data without modifying it, and if the data is still present in the NIC's memory then it is sent off directly from the NIC. An exchange between the host's memory and the NIC's memory is thus avoided. Payload Caching tries to benefit from the great quantity of memory embedded on the NIC whereas our work aims at studying the possible performance benefits when making use of the computing power present on the NIC.

S. Walton and al. attempt in [WHT98] to enhance the performances of workstation-based routers. The idea behind this work is very close to the one behind Payload Caching: the PCI bus in the PC architecture rapidly becomes the bottleneck so shortening the data-path is crucial. In this work data goes directly from the input NIC to the output NIC without

staging by the host's memory. Here again the computing power present on the NIC is not fully exploited.

T. Spalink and al. are building a software-based router using network processors [SKPG01]. The router is partly implemented on a NIC which is based on the Intel IXP1200 network processor [Cor00]. This NIC is fairly complex, it comprises six 100Mb/s Ethernet ports and two 1Gb/s Ethernet ports and is thus especially designed for routing. Unlike this work we want to focus on less specific hardware.

Making use of Network Processors is of interest in other areas. Works presented in [WW00] and [WT01] attempt to enhance MPI implementations by moving some MPI functions onto the NICs. In [COF99] is presented a global memory system that benefits from Network Processors.

5 Concluding remarks

In this paper we have presented a model and an experimental study of the forwarding operation. Our model allows to compare the performances of the forwarding operation when performed on the Network Interface Card with those of the forwarding operation when performed on the Host Processor, it can thus predict when NP forwarding leads to better performances. The experimental results obtained with our software prototype corroborate the results obtained with the model. We have introduced two different NP forwarding techniques: store-and-forward and cut-through. The cut-through technique consists in pipelining the send (from NIC to network) and receive (from network to NIC) operations. This technique applies when one only needs to read the message or when one needs to write in the message header regardless of the rest of the message (which may have not been received yet). If the message header must be modified regarding the rest of the message (e.g. checksum) then only the store-and-forward technique applies. This technique consists in receiving the entire message before starting re-sending it to the network. The cut-through technique gives much better performances than the store-and-forward technique, especially for large messages. For 1024-byte messages, both techniques perform better than forwarding data at the host level when the difference between the execution time of the forwarding algorithm when executed on the NP and the execution time of the forwarding algorithm when executed on the NP (δ) is lower than or equal to $10\mu s$. Beyond this threshold HP forwarding leads to better performance results. For 8192-byte messages, with the cut-through technique, if δ is lower than or equal to $65\mu s$ then it is worth performing the forwarding operation at the NIC level. For a simple forwarding algorithm NP forwarding with the cut-through technique gives 45% of improvement.

When NP forwarding is used the Host Processor does not participate in the forwarding operation at all, therefore, applications running on the HP can benefit from more CPU resources. The possible performance gain obtained by offloading some host machine's networking tasks onto the NP is not easy to measure; not only are saved CPU cycles but also context switches and possibly cache and TLB misses. Until now, we have not studied precisely this offload effect. We plan to study that in real-life contexts and the next step is to

move some protocol functions (e.g. IP firewalling, active routers functions) onto the NIC and study the repercussions this has on the applications running on the Host Processor.

References

[BCF95] N.J. Boden, D. Cohen, and R.E. Felderman. Myrinet - A Gigabit-per-Second Local-Area Network. In *IEEE Micro*, pages 29–36, February 1995.

[COF99] Y. Coady, J.S. Ong, and M.J. Feeley. Using Embedded Network Processors to Implement Global Memory Management in a Workstation Cluster. In *The Eighth IEEE International Symposium on High Performance Distributed Computing*, Redondo Beach, California, August 1999.

[Cor00] Intel Corporation. IXP1200 Network Processor Datasheet, September 2000. `http://www.intel.com/design/network/products/npfamily/ixp1200.htm`.

[For95] Message Passing Interface Forum. MPI: A message passing standard, June 1995. Available from `http://www.mpi-forum.org/docs/docs.html`.

[PT98] L. Prylli and B. Tourancheau. BIP: A new Protocol Designed for High Performance Networking on Myrinet. In *1st Workshop on Personal Computer based Networks Of Workstations (PC-NOW '98)*, volume 1388 of *Lect. Notes in Comp. Science*, pages 472–485, Springer-Verlag, April 1998.

[Ram00] Ramix. PMC697 Ethernet NIC, 2000. `http://www.ramix.com/products/index.php`.

[SKPG01] T. Spalink, S. Karlin, L. Peterson, and Y. Gottlieb. Building a Robust Software-Based Router Using Network Processors. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, October 2001.

[WHT98] S. Walton, A. Hutton, and J. Touch. Efficient High-Speed Data Paths for IP Forwarding using Host Based Routers. In *Proceedings of the 9th IEEE Workshop on Local and Metropolitan Area Networks*, pages 46–52, November 1998.

[WT01] R. Westrelin and B. Tourancheau. Support for MPI at the network interface level. In *8th European (PVM/MPI) Users Group Meeting*, Santori (Thera) Island, Greece, September 2001. Springer - Verlag. to appear.

[WW00] A. Wijeyeratnam and A. Wagner. MPI-NP II: A Network Processor Based Message Manager for MPI. In *International Conference on Communication in Computing*, Las Vegas, June 2000.

[YC00] K. Yocum and J. Chase. Payload Caching: High-Speed Data Forwarding for Network Intermediaries. In *2001 USENIX Technical Conference*, December 2000.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399