Network Headers Optimization for Distributed Programming

João Henrique de Souza Pereira[1], Liria M. Sato[2]
[1]Dept. de Eng. de Sistemas Eletrônicos (PSI)
[2]Dept. de Eng. de Computação e Sistemas Digitais (PCS)
USP – University of São Paulo
São Paulo-SP, Brazil
joaohs@usp.br, liria.sato@poli.usp.br

Pedro Frosi Rosa[3], Sergio Takeo Kofuji[1]
[3]Faculdade de Computação (FACOM)
UFU – Federal University of Uberlândia
Uberlândia-MG, Brazil
frosi@facom.ufu.br, kofuji@pad.lsi.usp.br

***Abstract*—This paper investigates the possibility of network cost reduction for distributed programming applications and presents a proposal for networks, with level 2 connectivity of the TCP/IP architecture. This proposal reduces the network traffic by minimizing the overhead of packet headers. To prove this enhancement possibility, it is presented the network analysis performed for the LAM MPI, sending a vector of integers and returning its sum. By the analysis of this test, it was confirmed that the current proposal allows, in this case, a reduction of 15.08% over the total network traffic (in bytes).**

Keywords-Computer Networks; Distributed Programming; MPI; Optimization

I. Introduction

The distributed communication allows the use of computational resources of distinct hosts through the separation of memory, processor and clock, that each machine has individually [1].

There are several technologies used to perform the distributed communication and when they use the TCP/IP architecture, they make use of the lower layers (physical and data link) and intermediate (network and transport) of this architecture. The use of these layers, associated with control and data transfer between the hosts can result, in certain cases, in a high network resource demand and consequently in a high computational cost [2].

Considering the importance of the distributed communication for computational systems and for the society itself, this paper presents an alternative for network communication cost reduction, specifically for distributed programming, when the TCP/IP architecture is used with connectivity in layer 2.

This purpose leads to show the proposed enhancements and verify the impact of this alternative in distributed programming. It is relevant to clarify that it is not the purpose of this paper to expand the discussions to all network communication fields, because of the extension of this study area. Thus, the discussion presented here is limited to the distributed programming field.

Therefore, this paper was organized as follows. Section 2 shows some related works in network optimization for distributed programming in TCP/IP architecture. Section 3 presents The network optimization proposal for distributed programming in TCP/IP architecture with connectivity in layer 2. Section 4 presents the results and performance analysis evaluation and the last section shows the conclusion and suggestions for future works in this research area.

II. Related Works Concerning Network Optimization in TCP/IP Architecture

The TCP/IP architecture is largely used in the network area and follows the OSI Reference Model, from ISO. However, TCP/IP does not use the section and presentation layers, from OSI Reference Model. The main protocols of this architecture came up three decades ago, as well as the IP, TCP and UDP, published in the IETF in RFC 760, 761 and 768, in 1980 [3]-[5].

As described in [6] and [7], besides the increasing computational evolution, there has not been significant developments in layers 3 and 4 in the TCP/IP architecture, since its main protocols specification. Also, the new demands of applications were met through new specifications/adaptations in protocols, without structural development of intermediate layers, which generated gaps in attending the applications requirements.

Some others studies show that the developments in layers 3 and 4 in Internet architecture were blocked due to the expansion of installed base, enlarged because of military and academic interests and commercial use [8]. Therefore, the developments in layers 3 and 4 have become more complex through the years, and the new requirements demanded by the applications have been attended by strategies which do not alter the original structure of these layers.

For example, in the beginning of the 80's the networks did not need voice and video transmission over computer networks, as nowadays. It was also not necessary to support sensor networks, their real time needs and inter/intra cluster communication and computational grids. Concerning the demanded requirements by the applications, that have changed at the last decades, here are some of them:

QoS;
Security;
Mobility;
Throughput;
Real time;
Network Management.

Requirements like these, among others, have had changes and evolutions throughout the years which not always were followed by developments and optimizations in the distributed systems. To contribute in this area, the next section presents a proposal for attending the applications needs in distributes programming, focusing in cost reduction of network communication.

III. Headers Optimization Proposal for Distributed Programming

The increase in the computational capacity of network elements allowed the expansion of applications with more complexity and distinct requirements, such as applications with support to distributed processing. Even though this evolution has been expressive for network elements, it was not followed by developments in the protocols in data link, network, and transport layers of the TCP/IP architecture which kept its structure mostly unchanged [9]-[10].

The network and transport layers of this architecture are used by the systems which make use of the distributed programming to meet the communication needs of the applications. For the purpose of this work to increase the network performance in the distributed programming area, the application needs necessary to support are:

1. Host addressing;
2. Process addressing;
3. Packet size control;
4. Delivery guarantee;
5. Interoperability with TCP/IP.

A possible solution that guarantees the fulfillment of these needs and that brings a better network cost can be an alternative to the optimization of distributed programming. Our proposal is reduce the existing redundancies in link, network, and transport layers of the TCP/IP architecture and bring an alternative for the layer model in this architecture. To have success in this alternative, is suggested a change in the layer structure for distributed programming, according to Figure 1, in the networks with connectivity in layer 2.

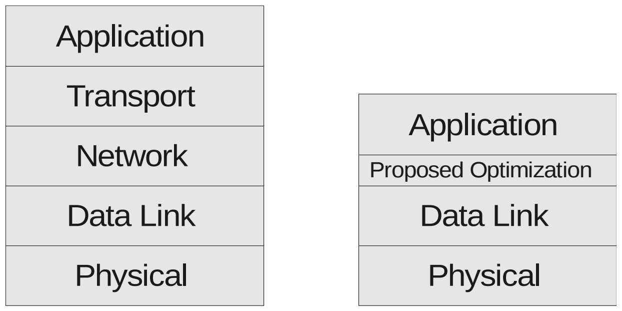

Figure 1. Layers Proposal

The structural change in the current architecture can cause impact in compilation of applications and in their communication at the Distributed Operational Systems area. It is not the purpose of this work to discuss the impacts in compilation of applications, but to propose an alternative which minimizes the network cost from the structure optimization of communication protocols used in this architecture.

To propose an alternative which minimizes the network cost from structure optimization of communication protocols used in TCP/IP architecture, the next sections present a proposal to meet the five necessary requirements (listed before) for the network communication in distributed programming.

A. *Considerations over host addressing*

The host addressing in data networks is performed by logic and physical addresses. However, according to Hegering [11] for a local network that has connectivity in layer 2, the use of these two addresses by host can be redundant in some cases. In networks with connectivity in layer 2, the logical address can be substituted by the physical one in some cases, for example, for hosts with fixed addressing.

To keep the compatibility with current standards, the applications and their users can make use of logical address, and let the Distributed Operational System solve it by the use of ARP (Address Resolution Protocol) [12]-[14]. Once the physical address is localized by the Reply of ARP (Eth Type 0806 and ARP Opcode 0002) in the return of the Request of logical address (Eth Type 0806 and ARP Opcode 0001), there is no more need to address the packets with the use of layer 3 in a network with connectivity in layer 2.

Therefore, the packets can be delivered directly by the physical address, without the IP address, saving 20 bytes, which is the minimum IP heading [3] [15].

Even though the exclusion of layer 3 does not impact in the connections in layer 2, this does not solve the need of multiple procedures of addressing, which for TCP/IP architecture is done by the use of ports in layer 4, both for UDP protocol as well as for TCP.

B. Considerations over the addressing of processes

To optimize the use of Transport layer, it is necessary an alternative to address the processes. This requirement can be met without using UDP or TCP, in the layer 4 of TCP/IP. The deliver proposal is to send 1 byte of control in the beginning of the payload of the link layer. From this byte, the first two bits inform how many bytes will be used for the addressing of multiple processes, according to rule 2^N, where N is the decimal value of these two bits.

The complementary bits to these two will be used for the addressing of processes. For example, for the value "00" there are 6 complementary bits which allow 64 addresses, as follows:

value 00: 2^0 = 1byte	2^6 = 64 addresses
value 01: 2^1 = 2bytes	2^14 = 16.384 addresses
value 10: 2^2 = 4bytes	2^30 = 1G addresses
value 11: 2^3 = 8bytes	2^62 = 4E addresses

The limit for the number of ports for the TCP and UDP protocols is 2^16 (64K) [6] – less than this proposal which supports 2^62. For the current distributed applications, nowadays a higher quantity of addresses is not so necessary, but in the future it can be. For example, it can be when it will be necessary communication among hosts with huge quantity of processors and that support huge quantity of processes in HPC (High Performance Computing).

The TCP has 20 bytes of heading and to address the processes, is proposed a variable quantity between 1 to 8 bytes according to the quantity of addresses of used processes. Therefore, the quantity reduction of bytes sent in relation to the TCP heading will be:

value 00: 2^0 = 1byte	1/20 = 95,0%
value 01: 2^1 = 2bytes	2/20 = 90,0%
value 10: 2^2 = 4bytes	4/20 = 80,0%
value 11: 2^3 = 8bytes	8/20 = 60,0%

The UDP heading has 8 bytes and in this proposal the proposed sent byte reduction in relation to the heading of this transport protocol will be:

value 00: 2^0 = 1byte	1/8 = 87,5%
value 01: 2^1 = 2bytes	2/8 = 75,0%
value 10: 2^2 = 4bytes	4/8 = 50,0%
value 11: 2^3 = 8bytes	8/8 = 0,0%

Therefore, the quantity of addressing of processes is raised with the reduction of band consumption, except for the value "11" compared to the UDP, which presents 0% of reduction.

For the current architecture of distributed communication, with the maximum number of ports equal to 2^16, the number "10" would be sufficient, once it allows the equivalent to address up to 1 Giga ports (processes). For this value, there is a reduction of network cost of 80% in the transport layer, compared to the TCP use, and 50% to the UDP.

Removing the IP, TCP and UDP protocols at the network and transport layers it is necessary to control the packet size, since this information is sent, in the TCP/IP architecture, by the layers that were removed (layers 3 and 4). On an attempt to guarantee this need to control packet size, is presented hereafter a solution proposal.

C. Considerations over packet size control

The quantity of bytes of the packet is informed in the IP Datagram in the field "Total Length", of 16 bits and in the transport layer this information is held in the field "Length" of the TCP and the UDP, also of 16 bits. In the TCP/IP architecture, the use of the "Total Length" in the IP and the "Length" in the TCP and UDP it is redundant, once they have a close relationship, defined by:

$$Length = Total\ Length - IHL \times 4$$

To inform the packet size, without layers 3 and 4, is proposed to send one byte, where the first two bits indicate how many bytes will inform the packet size by the rule 2^N. Thus, this proposal to control packet size is homogeneous with the previous proposal of addressing processes and follows the same logic of construction, where the maximum data packet size will be:

value 00: 2^0 = 1byte	(2^6)-1 = 63 bytes
value 01: 2^1 = 2bytes	(2^14)-1 = 16.383 bytes
value 10: 2^2 = 4bytes	(2^30)-1 = 1G-1 bytes
value 11: 2^3 = 8bytes	(2^62)-1 = 4E-1 bytes

For the IP, TCP, and UDP that use 16 bits to inform the packet size, the maximum size is 64k bytes, being inferior to the capacity of this proposal, which allows increasing the packet sizes. This can be useful in certain cases, in the distributed communication when it is necessary sending a great quantity of data. For example, in the classification and target analysis and spacial cube edges using UWB (Ultra-Wideband). In this situation, it would not be necessary to send multiple packets, which reduces the network overhead generated by sending the heading in each packet.

It is important to highlight that for certain networks there is fragmentation of the packets due to the MTU (Maximum Transmission Unit) what can limit this development possibility. In Ethernet networks, for example, it is common the use of MTU equal to 1500 bytes and there is also the Jumbo Frame with 9000 bytes.

For the distributed programming, sending the packets can be done by the use of UDP or TCP. For the communication with UDP there is no guarantee for the packet delivery, however this guarantee takes place in the TCP. The considerations over this guarantee are approached in the next sub-section.

D. Considerations over packet delivery guarantee

In the TCP/IP architecture not all packets need a delivery guarantee by the transport layer, since some applications use their own mechanisms of control and make use of the UDP for network cost reduction [16]. With the technological evolution, in the last decades and the gradual reduction of network elements with less quality, which increased the collisions (hub, for example), the distributed programming had therefore an advance in quality of communication.

Thus, is suggested that the data delivery guarantee, between hosts, be performed by the link layer, from the use of network elements with enough quality for that and using the principles discussed by Tanenbaum and Kurose [17] [18], where the layer 2 is responsible for detecting and correcting the errors in sending data between hosts.

The proposals to address hosts/processes, control the packet size and guarantee the delivery in networks with connectivity in layer 2 can be used in such a way to guarantee the inter-operability with current architectures, which will be discussed in the next sub-section.

E. Considerations over inter-operability with current architectures

In order not to cause impact in current communication structures, this proposal needs to be implemented in the most transparent way possible for the applications. This is possible beginning with the change of modules of Distributed Operational Systems, which are responsible for implementing the protocol stack of the TPC/IP architecture.

In this protocol stack modification, it is proposed to use a new Ether type, substituting the 0x800 (2048 decimal). The new value will be used to identify the packets with this new structure. To use this new Ether type, it is suggested to register in IANA the Ether type 0x809 (2057 decimal), which is available for use according to the list (partial) bellow, depicted from IANA:

Ether type		Exp. Ether.		Description	References
2048	0800	513	1001	Internet IPv4	[IANA]
2049	0801	-	-	X.75 Internet	[XEROX]
2050	0802	-	-	NBS Internet	[XEROX]
2051	0803	-	-	ECMA Internet	[XEROX]
2052	0804	-	-	Chaosnet	[XEROX]
2053	0805	-	-	X.25 Level 3	[XEROX]
2054	0806	-	-	ARP	[IANA]
2055	0807	-	-	XNS Compatability	[XEROX]
2056	0808	-	-	Frame Relay ARP	[RFC1701]
2076	081C	-	-	Symbolics Private	[DCP1]
2184	0888-088A	-	-	Xyplex	[XEROX]

Fonte: http://www.iana.org/assignments/ethernet-numbers

The applications constructed by the definitions of this proposal inform the Distributed Operational System, in the “bind” routine, that they are prepared to receive the communication flow. Thus, the Distributed Operational System send the data directly to them, without the need of going through the traditional TCP/IP stack.

The Distributed Operational System will also be in charge of translating the data from the traditional applications to the new ones when the packets are sent with the current TCP/IP structure. In this situation, the Distributed Operational System will send the received data to the new applications with the change in the traditional ports to the new addressing way proposed here, which will guarantee the inter-operability with the tradicional applications.

When the Distributed Operational System identifies the packet received with the Ether type 0x809, a link layer automatically delivers the data for the application with the address correlation of the received process for the UDP/IP port with the same value. Thus, to inter-operate with the traditional applications, the process address information must be limited to the value 65,535, limit for the number of UDP and TCP ports.

The next section presents the performance evaluation that this proposal enables through a real test analysis of a distributed programming with MPI use.

IV. Performance Evaluation

To measure accurately the enhancement possibility shown here, it is necessary to verify the impact of this alternative in the network communication of distributed programming. For so, the next subsections analyze a real test case with the MPI use and the network optimization.

A. Network cost measurement of MPI distributed programming

In the tests performed, 2 hosts were used (1 master and 1 slave) with the Linux Operational System version 2.6.27.5-41 fc9.i686 and LAM implementation for MPI, version 7.1.4 / MPI 2, from the Indiana University. For capturing the packets the Wireshark version 1.0.3 was used.

For sending and receiving the data were used respectively the MPI Send and MPI Recv.

In the test case was sent a vector of 100 positions, for summing up in the slave. In this procedure, 56 packet between 2 hosts were sent. All sent packet had 20 bytes of IP header (IHL minimum size), which means that there was no information in the optional field and the communication took place in 30mseg.

The Figure 2 shows the packet size sent, where it is also verified the correlation between the total size informed in the IP heading and the size informed by the UDP/TCP, discussed before. The Y scale was intentionally limited in 300 for better viewing. That was necessary because the packets from 9 to 12 had their sizes over the average of the others, due to their use to send the packets from the master to the slave. In fact, these 4 packets are fragments of a sole packet, with the total size informed by the UDP equal to 4,508 bytes.

For the packets from 9 to 11 the IP total size was 1500 bytes, which is the MTU value in the used test environment. Each one of the 3 packets had 1480 bytes of data and 20 bytes of IP heading. Packet number 12 had 20 bytes of IP heading plus 8 bytes of the UDP heading.

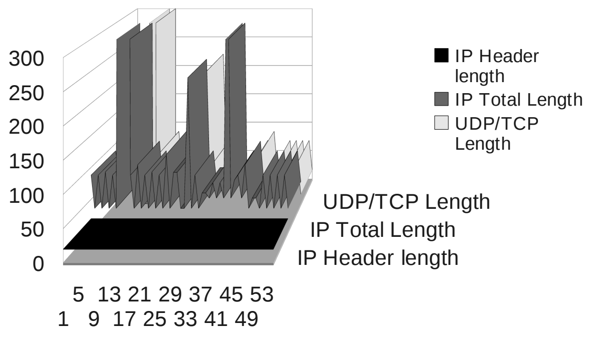

Figure 2. Packet Size between Master and Slave

Exactly 50% of the packets sent were originated by the master and 50% by the slave, considering that 76.8% (43) used the UDP as transport protocol and the rest 23.2% (13) were transmitted with TCP. From the packets sent by the master, 82.1% (23) used the UDP and 17.9% (5) the TCP, whereas the slave sent 71.4% (20) packets with the UDP and 28.6% (8) with TCP.

Figure 3 shows the distribution chart of TCP and UDP use between the master and the slave. It is possible to notice that the TCP packets have delivery confirmation, therefore, for each TCP packet sent there is another one, for confirmation.

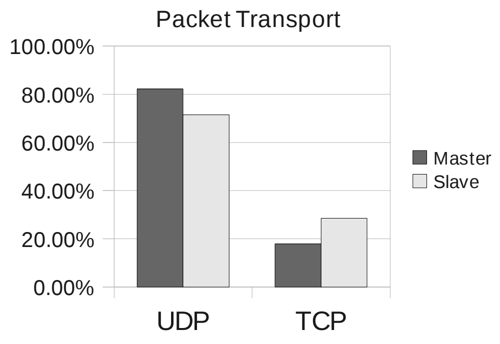

Figure 3. Packet Transport between Master and Slave

The difference between the quantity sent between the master and the slave is explained by the TCP/IP connection/disconnection procedure in 3 ways, where the slave started and finished the connection. In the disconnection, the slave sent the confirmation of the last TCP packet received, being with 3 packets sent more than the master.

B. *Result of the proposed network cost optimization*

To show the impact of this proposal in the network cost reduction, is showed the re-construction of the first packet which was sent from the master to the slave. Also is presented the analysis of the total optimization for all packets of the test performed with the LAM MPI.

The first packet sent had 102 bytes, distributed as follows:

1) Ethernet Protocol (14 bytes):

```
00 1a 4d a3 34 11 00 1b 24 f6 d8 14 08 00
```

2) IP (20 bytes):

```
45 00 00 58 00 00 40 00 40 11 12 6d 0a 0a 0a 0a 0a 0a 0a 0b
```

3) UDP (8 bytes):

```
82 a3 bb 29 00 44 55 92
```

4) Data (60 bytes):

```
00 00 00 00 00 00 02 8f 00 00 00 01 40 00 00 0e 00 00 00 00
00 00 00 00 00 00 01 00 00 00 00 00 ff ff d9 1c 00 00 00 01
00 00 26 e4 00 00 00 05 00 00 00 05 00 00 00 05 bf 96 40 98
```

In this proposal there are changes in the protocols according to the structures described in section 2. For the Ethernet protocol there is a change of the Ether Type from 0800 to 0809, according to 2.5. To inform the destination port of this packet (bb 29 = 47913) according to 2.2, it is used the value “10” with 4 bytes to designate the port number. Therefore, there are the byte “80 00 bb 29” by the binary construction.

```
1000 0000  0000 0000  1011 1011  0010 1001
```

For the packet size control there is the value “00” which allows inform the data field with up to 63 bytes (2^6-1). Thus, for this packet there is 1 byte (3C) of information:

```
0011  1100
```

There is no change in the data field and the packet reconstruction is performed according to the sequence bellow:

1) Ethernet Protocol (14 bytes):

```
00 1a 4d a3 34 11 00 1b 24 f6 d8 14 08 09
```

2) Optimization Proposed (5 bytes):

```
80 00 bb 29 3C
```

3) Data (60 bytes):

```
00 00 00 00 00 00 02 8f 00 00 00 01 40 00 00 0e 00 00 00 00
00 00 00 00 00 00 01 00 00 00 00 00 ff ff d9 1c 00 00 00 01
00 00 26 e4 00 00 00 05 00 00 00 05 00 00 00 05 bf 96 40 98
```

For this packet, there is an enhancement in network cost from 102 to 79 bytes, which represents a reduction of 22.5% in size. Concerning the layers 3 and 4, this proposal reduces 82.1% of its overhead, since the 28 bytes of the heading (20 from IP + 8 from the UDP) become only 5. For the TCP use, the network cost reduction would be greater since this protocol has the heading with a size greater than the UDP and performs the packet confirmation.

The percentage reduction of the packet network cost, shown before, has many variations according to the packets size. For the test performed where the master sends a vector of 100 positions for the sum up in a slave, the reduction of network cost, per packet, is shown in Figure 4.

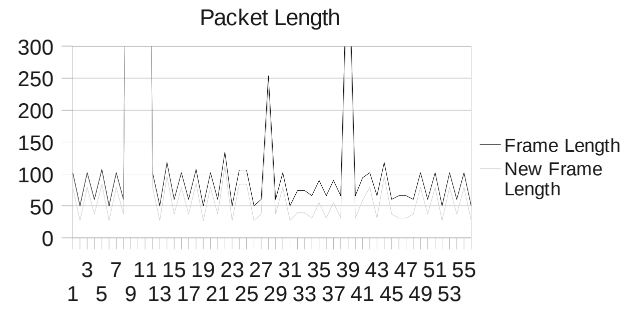

Figure 4. Network Optimization per Packet

Also, in this figure the Y scale was intentionally limited in 300 for better viewing, because of the sizes of the packets from 9 to 12.

In this test, the 56 packets had a total of 9,500 bytes. With the new structure, there is a reduction to 8,067 bytes, which represents an enhancement of 15.08% for the quantity of bytes sent. In relation to the IP, UDP and TCP headers, this proposal reduces the overhead from 1,724 to 291 bytes, with a total of 83.12% of optimization.

V. Conclusion

The proposal presented in this work enables the network cost reduction for the distributed programming in systems with connectivity in layer 2. This optimization is possible from attending the needs in communication for hosts/processes addressing, packet size control, delivery guarantee and inter-operability with the current architectures. With the fulfillment of these needs, this work contributes to the performance development of the distributed programming applications.

For the test performed with the LAM MPI, there was a reduction of 15.08% in the quantity of bytes between the master and the slave for the sum of a vector of integers. In scenarios that demand high processing this percentage optimization can be even higher, because the option to sent bigger packets.

For future works, is suggested the implementation of this proposal and its expansion in tests and analysis of performance for sending data with gradual size, starting from small (1 byte) up to large ones (Giga bytes). Another suggestion is to study over the possibility of reduction of sent packet quantity, for example, the removal of confirmation packets from TCP and the use of link layers with MTU higher than 1500 to minimize the packet fragmentation.

It is also pertinent the study a better mechanism to place the ports, for different applications to communicate between themselves and between the masters and the slaves using distributed programming. For example, using the Horizontal Addressing by Entity Title and the Domain Title Service (DTS), described in [19].

References

[1] *Silberschatz, A., Galvin, P. B., Gagne, G. (2005) "Operating Systems Concepts". 7th ed., Wiley.*

[2] *Tanenbaum, Andrew S., Steen, Maarten Van. (2007) "Sistemas Distribuídos", 2ª ed, São Paulo, Prentice Hall Brasil.*

[3] *J. Postel, "DoD standard Internet Protocol", DARPA Information Processing Techniques Office, USC/Information Sciences Institute, RFC 760, 1980.*

[4] *J. Postel, "DoD Standard Transmission Control Protocol", DARPA Information Processing Techniques Office, USC/Information Sciences Institute, RFC 761, 1980.*

[5] *J. Postel, "DoD Standard User Datagram Protocol", DARPA Information Processing Techniques Office, USC/Information Sciences Institute, RFC 768, 1980.*

[6] *J. H. S. Pereira, S. T. Kofuji and P. F. Rosa, "Distributed Systems Ontology", New Technologies, Mobility and Security Conference – NTMS, IEEE Xplore, Cairo, 2009.*

[7] *J. H. S. Pereira, S. T. Kofuji and P. F. Rosa, "Horizontal Address Ontology in Internet Architecture", New Technologies, Mobility and Security Conference – NTMS, IEEE Xplore, Cairo, 2009.*

[8] *Comer, Douglas (1995) "Internetworking with TCP/IP Volume I – Principles, Protocols and Architecture", 3ª ed, New Jersey, Prentice Hall.*

[9] *E. S. Santos, F. S. F. Pereira, J. H. S. Pereira, P. F. Rosa e S. T. Kofuji, "Optimization Proposal for Communication Structure in Local Networks", International Conference on Networking and Services – ICNS, IEEE Xplore, Mexico, 2010.*

[10] *F. S. F. Pereira, E. S. Santos, J. H. S. Pereira, P. F. Rosa e S. T. Kofuji, "FINLAN Packet Delivery Proposal in a Next Generation Internet", International Conference on Networking and Services – ICNS, IEEE Xplore, Mexico, 2010.*

[11] *Hegering, Heinz-Gerd, Läpple, Alfred, (1993) "Ethernet - Building a Communications Infrastructure", Addison-Wesley, Munich.*

[12] *Arkko, J., Pignataro, C. (2009) "IANA Allocation Guidelines for the Address Resolution Protocol (ARP")", RFC 5494.*

[13] *Cheshire, S. (2008) "IPv4 Address Conflict Detection", RFC 5227.*

[14] *Plummer, D. (1982) "Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48bit Ethernet Address for Transmission on Ethernet Hardware", RFC 826.*

[15] *Halsall, F. (1992) "Data Communications, Computer Networks and Open Systems", 3ª ed, UK, Addison Wesley.*

[16] *Comer, Douglas, Stevens, David L. (1999) "Internetworking with TCP/IP Volume II – Design, Implementation and Internals", 3ª ed, New Jersey, Prentice Hall.*

[17] *Kurose, J. F. e Ross, K. W. (2005) "Redes de computadores e a Internet – Uma Nova Abordagem", 3ª ed, São Paulo, Addison Wesley.*

[18] *Tanenbaum, Andrew S. (2007) "Sistemas Operacionais Modernos", 2ª ed, Rio de Janeiro, Prentice Hall Brasil.*

[19] *J. H. S. Pereira, P. F. Rosa e S. T. Kofuji, "Horizontal Addressing by Title in a Next Generation Internet", International Conference on Networking and Services – ICNS, IEEE Xplore, Mexico, 2010.*