Improvement of the Computer Network Transmission Band Usage by Packing Agent Technology

Tomasz Bartczak[1], Stanisław Paszczyński[2], Member, IEEE and Bogdan M. Wilamowski[3], Fellow Member, IEEE

[1,2] Tomasz Bartczak and Stanisław Paszczyński, Department of Dispersed Systems, University of Computer Technology and Management, Rzeszów, Poland, e-mail: (tbartczak, spaszczynski)@wsiz.rzeszow.pl

[3] Bogdan M. Wilamowski, Department of Electrical & Computer Engineering, Auburn University, Alabama, USA, e-mail: wilam@ieee.org

Abstract— **In this paper the algorithms which could improve transmission band for computer network has been shown. The analysis of file size distribution has shown that the typical website has more than 40% of files with sizes smaller than 1kB. Additionally, these small files are more frequently used because 80% of all references by client browser to website resources are for these small files. The paper presents an idea of a mobile agent, who could improve transmission band in computer network, especially for HTTP files. The server agent based on the resource list, required by the client compresses the files into MTU (Maximum Transfer Unit) packs has been designed. As a result, a number of transmitted packets (frames) can be reduced several times.**

Index Terms—**transmission band usage, packing algorithm, data compression.**

I. Introduction

Today, thanks to modern technology, information exchange is really fast. The number of people using Internet and Web has increased significantly. Also, the quality and Internet access is much better. Users expect fast, safe and reliable data flow. However, one of the problems not only for Internet users, is the limited availability of a transmission band in a network. As the result of such situations the access to the transmission band is limited, and this leads to lower users' productivity and satisfaction (i.e. downloading takes much more time than could be acceptable). It is known that most users give up waiting for information after about 8 seconds by giving the computer a next command.

The network performance is being constantly improved and this results with better usage of the transmission. There are several solutions to improve usage of a transmission band:
advanced compression of data,
dividing data flow to several parallel flows,
increasing size of TCP (Transmission Control Protocol) packets,
using protocols based on improved UDP (User Datagram Protocol).

In some applications the above solutions increase throughput (even 10 times) of data that is being used in the network. Unfortunately, these solutions are costly and often additional hardware is required.

There are other ways to improve usage of transmission band. One of them [8] proposes a three-step process to reduce the size of hypertext documents for reducing the network traffic on the Internet:
Shrinking HTML documents by removing data that will not alter the appearance and the function,
Encoding HTML documents by representing most frequent used tags and words with single bytes,
Compressing HTML documents by using standard compression utilities.

Another way to improve usage of transmission band is [6] specific extensions to the HTTP protocol for delta encoding and data compression. Compression of HTTP request and response messages [5] is used to optimize the final link between a mobile client and a stationary base station. Other compressions are also used [3, 4] to improve web structure notion.

TCP manually and automatic tuning techniques [9, 10, 11, 12, 13] by adapting the transmission protocol to transmission data are another way to improve usage of transmission band.

For every computer network protocol [1, 2], there is a specific maximum quantity of information (MTU) that can be transmitted in one frame. MTU - Maximum Transfer Unit is the maximum number of bytes that can be transferred in a single packet between two computers in the network. The MTU is associated with a complete path. The MTU of a path is the minimum of the MTU of the individual links in the path. For Ethernet, the MTU size is 1500 bytes. When PPPoE is used, the MTU size is reduced by eight bytes because of the additional overhead. Unfortunately, the packets do not use all of their capacities. For example, when much smaller file than MTU is being sent in a single packet then many bytes in the frame are not being used. Transmission protocols often send them in data packets with padding. For example, telnet typically sends one character in one data packet, when entered by client keyboard and the associated overhead is significantly larger than actual information. If a lot of files of small sizes are being sent then fraction of user data in total flow is significantly reduced. This effect is similar to usage of hard disc space especially when small files are saved and information where data is saved takes more of disk space than actual data.

II. Methods to increase the transmission throughput

Search engines are one of the most used processes on the Internet. The website, in order to be fully displayed, must receive all the files. Before the webpage is fully displayed sometimes as many as 50 different kind and size files (tab. 1, 2) are required [7].

Tab. 1. Distribution of files sizes at main pages of selected website.

	File size distribution in percent for main pages of selected websites			
Files sizes [B]	onet.pl	interia.pl	wsiz.rzeszow.pl	sejm.gov.pl
<100	16	13	8	4
100 - 500	16	17	40	0
501 - 1000	3	0	12	11
1001 - 1500	22	0	4	44
1501 - 3000	22	25	4	15
3001 - 4500	6	8	0	15
4501 - 6000	6	17	0	4
6001 - 10000	6	17	28	4
Above 10000	3	4	4	4
Total size [B]	154758	148752	187458	124635
Total number of files	32	24	25	27

Tab. 2. Distribution of files sizes at selected distance-learning courses. IT ESS2 is for *HP IT Essentials II: Network Operating Systems* course on cisco.netacad.net and CCNA is for *Cisco Certified Network Associate* course on cisco.netacad.net.

	File size distribution in percent for selected distance learning courses	
Files sizes [B]	IT ESS2	CCNA
<100	1	8
100 - 500	2	14
501 - 1000	24	19
1001 - 1500	15	6
1501 - 3000	7	1
3001 - 4500	21	3
4501 - 6000	4	4
6001 - 10000	7	21
Above 10000	19	24
Total number of files	2847	11380

The files are transmitted from the website using communication protocols that shape them to specific size and split into packet with appropriate packet headers. Besides transferring of the useful data (i.e. websites files), protocols generate an additional traffic of data exchange depending on transmission protocol (frames for openings and closings sessions, acknowledgement frames, etc.)
Traffic analyses show that user data transferred in the network occupy only a small fraction of transmission band. One can define the coefficient E of user data transmission efficiency for given protocol as:

$$E \equiv \frac{D}{D_T} \quad (1)$$

where D is data volume to be transferred, and D_T is volume of total transferred information. E satisfies relation $E \leq 1$ and $E=1$ for ideal case (ideal protocol). Typically E has value even smaller then 0.2 for LAN [5].
There are several ways to improve channel performance. One of them is optimization of the transmission algorithm by reduction of the quantity of transmitted information fragments. When the quantity of protocol frames is being limited, the fraction user data in total flow is being increased. In the case of small files it's possible to concatenate some of the files into one, which still can be sent in one packet. This situation could be compared to car traffic, when drivers decide to use public transportation instead of their own cars.
Based on these ideas two algorithms, which could improve channel performance, are proposed. Block diagrams for these algorithms are shown in Fig. 1, Fig. 2, Fig. 4 and Fig. 5.

```
Enter the list of files and their size.
Sort the files by size.
Set the status of all files to "free".
Create empty packet.
Set the option of a measured file to the grater file on the list.
If size packet+file+tag < MTU then concatenate file to packet, add tag and set file status to "occupied".
If there is next file with "free" status then set the option of a measured file to the next file of the list with status "free".
if there is file with "free" status then go to 6, else STOP.
```

Fig. 1. Pseudo-code of the file packing algorithm 1.

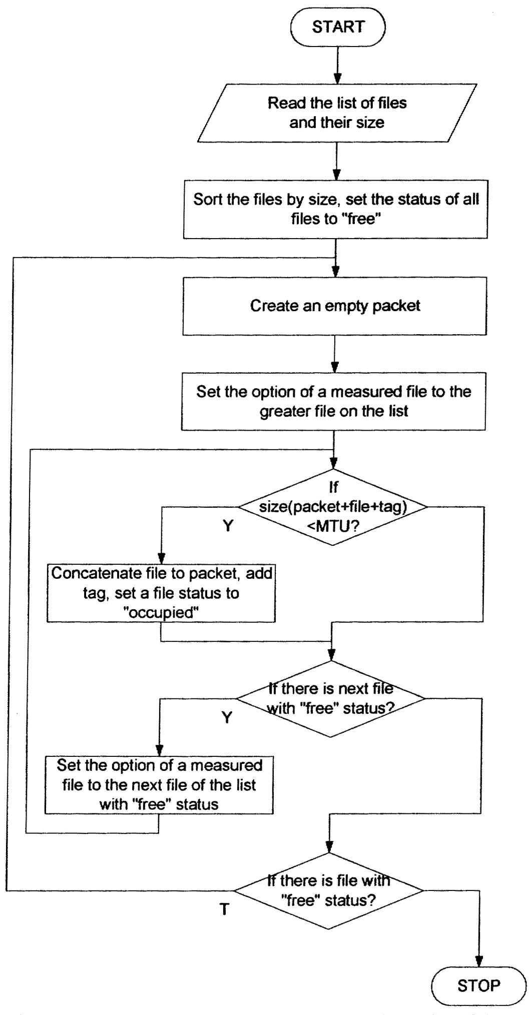
START
Read the list of files and their size
Sort the files by size, set the status of all files to "free"
Create an empty packet
Set the option of a measured file to the greater file on the list
If size(packet+file+tag) <MTU?
Y
Concatenate file to packet, add tag, set a file status to "occupied"
If there is next file with "free" status?
Y
Set the option of a measured file to the next file of the list with "free" status
If there is file with "free" status?
T
STOP

Fig. 2. Block diagram of the file packing algorithm 1.

If a prepared portion of information is much bigger than MTU, then using data compression would be efficient. However, for a small portion of data (files up to a couple hundred bytes) data compression is not worthy and what is even worse is that in these cases sizes of compressed files would be most likely greater then uncompressed ones (Fig. 3).

The compression ration depends on the type of the files. For example text files, numerical data, or HTML files can be efficiently compressed while files like GIF, PNG, JPEG, or PDF would be difficult to compress. Therefore, based on a file size itself it is not possible to estimate size of a file after compression. It's much easier to concatenate files into packets, so the size of a file is similar (but less or equal with header added) to the size of MTU. Compression can be also added to the process as shown in the algorithm on Fig 4 and Fig 5.

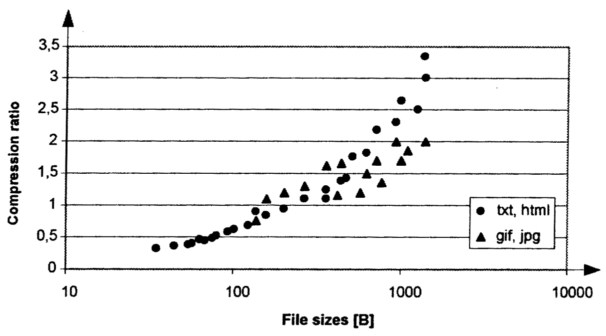

Fig. 3. Compression ratio as file sizes function (zip compressor) for some set of typical content website.

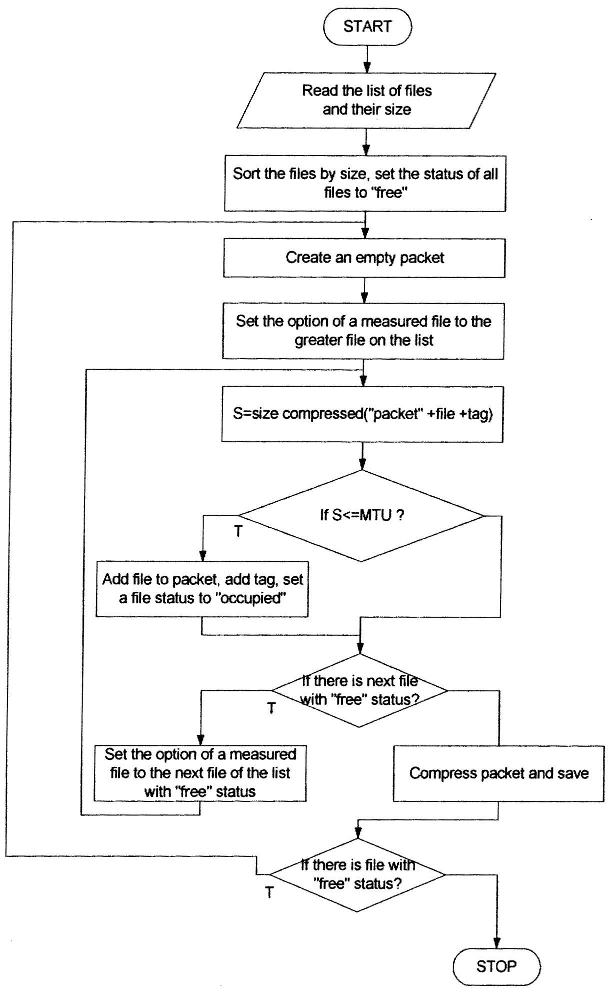
START
Read the list of files and their size
Sort the files by size, set the status of all files to "free"
Create an empty packet
Set the option of a measured file to the greater file on the list
S=size compressed("packet" +file +tag)
If S<=MTU ?
T
Add file to packet, add tag, set a file status to "occupied"
If there is next file with "free" status?
T
Set the option of a measured file to the next file of the list with "free" status
Compress packet and save
If there is file with "free" status?
T
STOP

Fig. 4. Block diagram of the file packing algorithm with compression (algorithm 2).

```
Enter the list of files and their size.
Sort the files by size.
Set the status of all files to "free".
Create empty packet.
Set the option of a measured file to the grater file on the list.
Set S = size compressed (packet+file+tag).
If S < MTU then add file to packet, add tag and set file status to "occupied".
If there is next file with "free" status then set the option of a measured file to the next file of the list with status "free", else compress packet and save.
if there is file with "free" status then go to 6, else STOP.
```

Fig. 5. Pseudo-code of the file packing algorithm with compression (algorithm 2).

Both algorithms reduce the required number transmitted packets k to n_1 or n_2. We have:

$$D_T = \sum_{i=1}^{k} R_i + kH + A_k \text{ , } D_T^{n_1} = \sum_{i=1}^{n_1} R_{pi} + n_1 H + A_{n_1} \text{ ,}$$

$$D_T^{n_2} = \sum_{i=1}^{n_2} R_{pi} + n_2 H + A_{n_2} \qquad (2)$$

where R_i - size of i-th file, R_{pi} - size of i-th packet with the files, k - number of files to be sent, H - sum of header and footer sizes in every packet, A - size of additional data transferred beside data packets (i.e. acknowledgment, collision packets), n_1 - number of packets sent using algorithm 1, n_2 -number of packets sent using algorithm 2.
There exists relation

$$k > n_1 \text{ ,} \qquad (3)$$

because several files are added in one packet and transmitted by one frame in algorithm 1. Thus transmitted frames number is decreased and the number of headers and tails is decreased. Size of additional data transferred beside data packet is decrease, which we can write as:

$$A_k > A_{n1} \qquad (4)$$

and using (2)

$$\sum_{i=1}^{n_1} R_{pi} - \sum_{i=1}^{k} R_i = k \cdot g \qquad (5)$$

where g is tag size, which is usually very small (typically is less then 10 B per file packed). Tag also includes file name for identification. So, user data size is a little bit higher when packet are composed but a profit on reduction of total number of transferred packets is significant (i.e. we observe in FTP protocol that for one frame with data transmitted there was about 1 kB additional information transmitted too. It is significantly bigger than sum of tags and file names in packet).

Because

$$k \cdot g << A_k - A_{n1} \qquad (6)$$

then using (3)-(6) the following relation can be written

$$D_k > D_{n1} \text{ .} \qquad (7)$$

From definition (1) and relation (7)

$$E_k < E_{n1} \qquad (8)$$

This means that transmission band could be better used by utilization of algorithm 1.
For algorithm 2 there is relation

$$n_1 > n_2 \qquad (9)$$

because files are additionally compressed and packed into packets with sizes very close but not greater than MTU size The average number of files in one packet prepared by algorithm 2 is bigger than this prepared by algorithm 1. Thus, the number of frames, headers, tails and additional data transferred is also smaller, which can be described by relation

$$A_{n1} > A_{n2} \text{ .} \qquad (10)$$

Additionally

$$\sum_{i=1}^{n_1} R_{pi} > \sum_{i=1}^{n_2} R_{pi} \qquad (11)$$

because of a reduction of number of packets in comparison with the number obtained by means of algorithm 1.
From expression (9)-(11) the following relation can be obtained

$$D_{n1} > D_{n2} \text{ .} \qquad (12)$$

From definition (1) and expression (12)

$$E_{n1} < E_{n2} \text{ .} \qquad (13)$$

This means, that algorithm 2 is better for transmission band usage than algorithm 1.
These types of algorithms take much more time for packets preparation, but it could be shown that it is still a small fraction of total transmission time (time for packets preparation + their transmission time) for contemporary servers with high speed microprocessors. The agent could prepare the packets in advance taking into account statistics of site node visits thus additionally reducing total transmission time. (The computational complexity of the algorithms will be estimated soon)

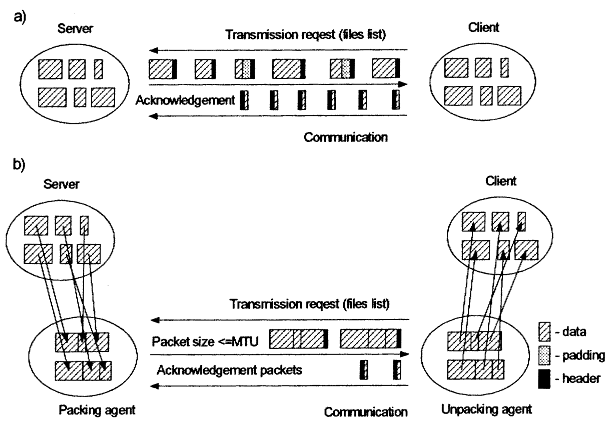

Fig. 6. An idea of reduction a number of packets sent in TCP/IP protocol by use the packing agent. a) without packing, b) with packing agent.

With the implementation of algorithms shown above, the mobile agent technique [14, 15, 16] can be used (Fig. 6). Agent placed on side of a sender would be responsible for preparing packets for transmission and agent placed on a receiver side would be responsible for the recovering information to its original form.

The mobile agent technique can be very effective in the case when there is no resident packing agent on the server. Agents could realize also other tasks e.g. searching and/or filtering information before packing. The size of such agent can be relatively very small (in our research the packing agent code size was equal to 55kB) thus the cost of its transmission and required system resources - neglected.

Mobile agents could be very convenient when fitting to frequently changed transmission protocol would be required. To obtain mobile version of the agent our code has to be significantly reworked and equipped with "mobility" procedures as well as appropriate "credential" for used ontology.

III. Experimental Results

In the experiment, simple two nodes network with client – server configuration has been used. Information was sent in three different ways: traditional, and then with proposed two algorithms. The obtained results were compared.

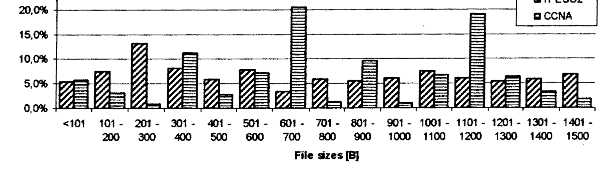

Fig. 7. Distribution in percent of file sizes (smaller then Ethernet protocol MTU).

Traffic was measured by TCPdump, which is the linux tool (see www.tcdump.org tool for details). 1460 files were used with the total size of 785919 bytes. The average size was 538 bytes, with a distribution as on Fig. 7. It took 70 seconds to send unmodified information using FTP (File Transfer Protocol) and 21935 packets were required to transmit this information.

When the algorithm 1 was used the 1460 files were packed to 565 packets, at a total size of 825339 bytes (average 1460 bytes). The number of files was reduced by to 38.7% (packets with data). In the case of algorithm 1 only 8498 packets were required and 28 seconds were needed to transmit the information. The number of sent packets had been reduced to 38.7%. These two numbers are identical, because the reduction of files and packets sent are proportional. In this experiment it is linear function, because collisions are eliminated. The transfer time was reduced to 40%.

Tab. 3. Coefficient of user data transmission efficiency and transfer time with algorithm 1 and algorithm 2.

Transmission method	E – coefficient of user data transmission efficiency	Informatio n transfer time [s]	Information total transfer time with preparing and unpacking packets [s]
standard	3,21	70	70
with algorithm 1	2,15	28	41
with algorithm 2	1,97	20	37

When the algorithm 2 was used the number of files was reduced to 394 data packets, at a total size of 579276 bytes (average 1472 bytes). Only 5930 packets were required and only 20 seconds were needed to transmit the information. The number of files as well as number of packets sent was reduced to 27%. The transfer time was reduced to 28.6%.

Above results show an optimal usage of size MTU packet in a reduced amount of time, although the information sent was 5% bigger than the first one (because the added tags identified files in packets). Also, the number of sent packets had been reduced.

The experiment contained two parts: preparing the packets and sending the information. The time required to preprocess the information depends on the speed of the used computer. In the experiment Pentium IV processor with 1,3 GHz clock was used. The algorithms were implemented in C++ language. In this case, the time to prepare the packets with algorithm 1 was 13 seconds and for second algorithm 17 seconds were required. In this stage of investigations we didn't optimize the speed of computer programs. We expect that after eliminating intermediate disk operations in these programs they will run at least 10 times faster. Anyway, in this experiment the total transmission times were reduced to 58.6% and to 52.9% respectively.

We expect also additional significant reduction of packet sent by FTP after using RAM disk instead hard disk (even 5 times) due to a big difference in their access (read/write) times an also after appropriate "tuning" of TCP protocol. For FTP and used computer system the hard disk is too slow so it causes a re-fragment of some packets by FTP before transfer and thus unnecessary growth of the total number of packets sent.

We observed similar as above shown influence of elaborated algorithms on transmission efficiency for other sets of files for $k > 500$.

Time needed to retrieve information on the receiving side (client side) can be neglected in comparison with time required for packet preparation. For the algorithm 1 (typically it was less than 10ms) it is only time required for localization file in unpacking packet and for algorithm 2 additional times for single uncompressing process of the data part of packet (typically was less than 100 ms).

IV. Conclusion

The proposed approach seems to be effective and better usage of a transmission band is possible. Implementation through the agent technique is cheap and easy to use. Presented algorithms can be improved by using more advanced files packing algorithms and using optimal compression algorithms for different file types (intelligent compressor). Also research concern $E(k)$ dependence for $k < 500$ will be realized.

V. References

[1] RFC 2616, Fielding R. at al., Hypertext Transfer Protocol -- HTTP/1.1, June 1999, http://www.rfc-editor.org/

[2] RFC 959, Postel J. at al., File Transfer Protocol, October 1985, http://www.rfc-editor.org/

[3] M. Adler and M. Mitzenmacher, "Towards compressing web graphs", *In Proc. of the IEEE DCC*, March 2001, pp. 203-212.

[4] T. Suel, J. Yuan, "Compressing the Graph Structure of the Web", *Proc. of the IEEE DCC,* March 2001, pp. 213-222.

[5] R. Krashinsky, "Efficient Web Browsing for Mobile Clients using HTTP Compression", Technical Report MIT-LCS-TR-882, MIT Lab for Computer Science, Jan. 2003.

[6] J. C. Mogul, F. Douglis, A. Feldmann, B. Krishnamurthy, "Potential benefits of delta encoding and data compression for HTTP", *Proc. of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication*, 1997.

[7] J. Charzinski, "HTTP/TCP connection and flow characteristics", *Performance Evaluation* 42 (2000), pp. 149–162.

[8] B. Choi, N. Bharade, Network traffic reduction by hypertext compression, *Proc. of the International Conference on Internet Computing*, 2002, pp. 877-882.

[9] H. Chen, P. Mohapatra, "CATP: A Context-Aware Transportation Protocol for HTTP", *Proc. of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03)*, May 2003, pp.922-927.

[10] H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. Wium Lie, Ch. Lilley, "Network Performance Effects of HTTP/1.1, CSS1, and PNG, Applications, Technologies, Architectures, and Protocols for Computer Communication", *Proc. of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication*, 1997.

[11] E. Weigle, W. Fang, "A Comparison of TCP Automatic Tuning Techniques for Distributed Computing", *Proc. Of the 11th IEEE International Symposium on High Performance Distributed Computing HPDC,* November 2002.

[12] Y. Miyake, T. Hasegawa, T. Hasegawa, T. Kato, "Acceleration of TCP Throughput over Satellite-Based Internet Access Using TCP Gateway", *Proc. of the Fifth IEEE Symposium on Computers and Communications (ISCC)*, July 2000, pp. 245-253.

[13] B. Tierney, "TCP Tuning Guide for Distributed Applications on Wide-Area Networks" *In USENIX & SAGE Login*, http://www-didc.lbl.gov/tcp-wan.html, February 2001.

[14] R. Pinheiro, A. Poylisher, H. Caldwell, "Mobile agents for aggregation of network management data" *In Proc of the First International Symposium on Agent Systems and Applications and Third International Symposium on Mobile Agents (ASA/MA99)*, October 1999, pp. 130-140.

[15] M. Wooldridge, N. R. Jennings, "Pitfalls of Agent-Oriented Development", In K. P. Sycara and M. Wooldridge, editors: Agents '98: *Proc. of the Second International Conference on Autonomous Agents,* May 1998.

[16] L. Wilson, D. Burroughs, A. Kumar, J. Sucharitaves, "A framework for linking distributed simulations using software agents", *In Proc. of the IEEE, Special issue on agents in modeling and simulation: exploiting the metaphor*, February 2001, vol. 89, pp. 174-185.