Performance Issues in the Design of a VPN Resistant to Traffic Analysis

Claudio Ferretti, Alberto Leporati, Riccardo Melen
Dipartimento di Informatica, Sistemistica e Comunicazione
Università di Milano Bicocca
Milano, Italy
{ferretti, leporati, melen}@disco.unimib.it

Abstract— **This paper describes the architecture of BlankNet, a secure Virtual Private Network, which is capable of protecting its users against most kinds of attacks to traffic secrecy, including traffic analysis attacks. A key aspect of the BlankNet architecture is the definition of the allowed packet communication patterns: we call this pattern a *virtual topology*. Among the various possible solutions, the binary cube topology is found to have favorable delay characteristics in light and moderate network traffic scenarios, while under heavy loading a completely connected topology turns out to be the most convenient one.**

Secure VPN; traffic analysis; hypercube; network performance

I. INTRODUCTION

In the field of Computer Security the development of a defense technique always stems from the analysis and understanding of the attack model which must be neutralized.

It is customary to classify security attacks into active and passive ones: the former category includes impersonation, forging/modifying content and denial of service, while the most studied form of passive attack is the interception and analysis of content. The latter can be counteracted by means of well known techniques, based on cryptography: the effectiveness of these solutions is very high.

Another kind of passive attack is traffic analysis. A lot of information can be gathered by an attacker which observes *timing*, *source*, *destination* and *size* of messages, even without any knowledge of their content. Neutralizing this kind of attack requires complex and expensive countermeasures, which are employed only in scenarios requiring the highest degrees of security.

This paper describes a part of the BlankNet project, being currently carried out at the University of Milano Bicocca. BlankNet defines the architecture of a secure VPN, which is capable of protecting its users against most kinds of attacks to traffic secrecy: in particular not only it does protect the content of messages by means of cryptographic techniques, but it is also designed to defeat traffic analysis attacks. The BlankNet VPN can be built upon any kind of network, ranging from an enterprise LAN to the Internet.

Much of the inspiration for the BlankNet architecture has come from our analysis of the characteristics of the Tor project [1]. This work is aimed at developing highly performing solutions solving the same kind of problems.

The subject of VPN topologies has been analyzed thoroughly in a context rather different from the present one, namely in the field of peer-to-peer networking: Chord is a very interesting example of these applications [2].

There is a huge literature on interconnection network for multiprocessors, where the binary cube and many other topologies have been studied [3][4]; this paper is indebted also to the research carried out on high speed space-division switching [5], particularly for what concerns some aspects of the performance analysis in Section V.

The rest of the paper is organized as follows: Section II contains a basic description of the BlankNet architecture; Section III sets up the key assumptions which allow a meaningful evaluation to be made; Section IV identifies a topology which is better than the ring and the complete connection in terms of packet delay in a lightly loaded network; Section V develops an analytical model which allows to extend the comparison of topologies to high loading situations; Section VI concludes the paper mentioning the issues to be faced when applying the results obtained to a concrete implementation scenario.

II. THE BLANKNET ARCHITECTURE

The security objectives of BlankNet can be obtained in a rather straightforward fashion, as described below.

All the stations transmit fixed size *packets* at fixed time intervals, towards destinations which are defined according to some predetermined rule. The payload of each packet may be a random bit sequence or an encrypted *message*, directed toward a final destination which may be different from the node which receives the packet.

Each node decodes the packets and processes the message headers, while the message payloads are protected by a second layer of coding, which ensures end-to-end secrecy; if a node is not the final destination of a message, encodes it again with a different key and forwards it according to a routing algorithm.

The encode/decode operations performed at each hop prevent an external observer from correlating the messages sent and received by a node and tracking the message paths: Figure 1 depicts the range of encoding operations.

The node architecture is represented in Figure 2, where a detail needs to be pointed out: while the routing buffer RQ is managed as a normal FIFO queue, the transmission buffer TQ must be seen as a set of FIFO queues, one for each possible packet destination.

 ISBN:978-1-61208-113-7

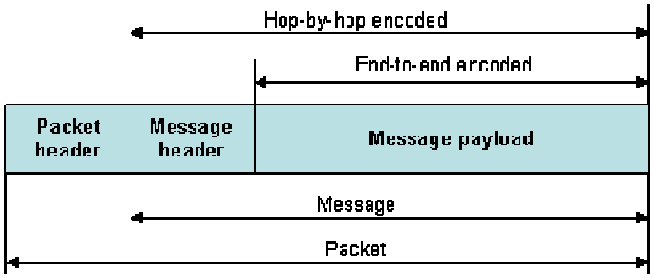

Figure 1. BlankNet message encoding

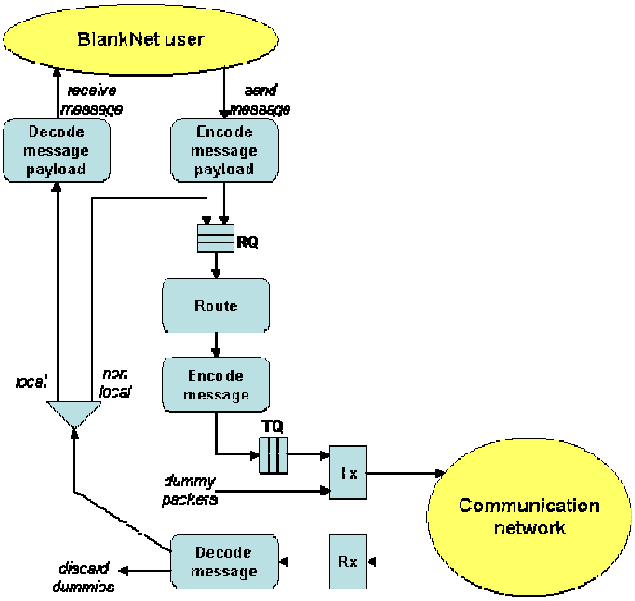

Figure 2. BlankNet node architecture

A key aspect of the BlankNet architecture is the definition of the allowed packet communication patterns: as it has been said before, the packets (which carry messages or random payloads) are sent to a predetermined set of nodes, at predetermined transmission times: we call this pattern a *virtual topology*. It is possible to implement a BlankNet using different virtual topologies, having different characteristics and performance.

In order to fix ideas, let us consider the unidirectional ring, depicted in Figure 3a, which is probably the simplest solution to implement. In this case, every node always transmits its packets to the same destination (node 1 to node 2, node 2 to node 3 etc). A message from node 1 to node 4 will be relayed by nodes 2 and 3 before reaching its destination: the traversal of intermediate hops is a penalty to be paid for obscuring the traffic patterns to external observers.

A different virtual topology is the complete connection depicted in Figure 3b. Here every node transmits its packets directly to the other nodes, but the transmissions must take turns (for instance node 1 sends its packets to the other nodes following the cyclic order: 2,3,4,5,2,3,4,5,2,…): in this case the performance penalty required for traffic pattern obfuscation is the time that messages wait for the transmission slot to the proper destination.

A bit of thinking suggests that these two topologies may have similar performance: in a network with N nodes with a ring topology, a message must cross N/2 (virtual) links in the average before reaching its destination, but it does not suffer from contention for the transmission time slot, while with a completely connected topology the message must wait, in the average, N/2 transmission slots before being sent (directly) to its destination.

It is however intuitive that, if we consider the delay experienced in a loaded network, with queues building up at each node, the complete connection should have a significant advantage with respect to the ring, because it does not employ network resources (transmission slots) for the intermediate hops.

These observations lead to two general questions, which are analyzed in this paper:

A) Can we find virtual topologies which are more efficient than the ring or the complete connection?

B) How do we model the performance of the various topologies in order to make a meaningful comparison in various operating conditions?

An answer to question A is given in Section IV, while Section V develops an analytical performance model.

III. BASIC SYSTEM PARAMETERS AND OPERATING MODES

As a first step towards the objective evaluation of the various virtual topologies, it is necessary to define precisely the application scenario where the comparisons are made. This definition involves the determination of some key parameters and operating modes of the VPN.

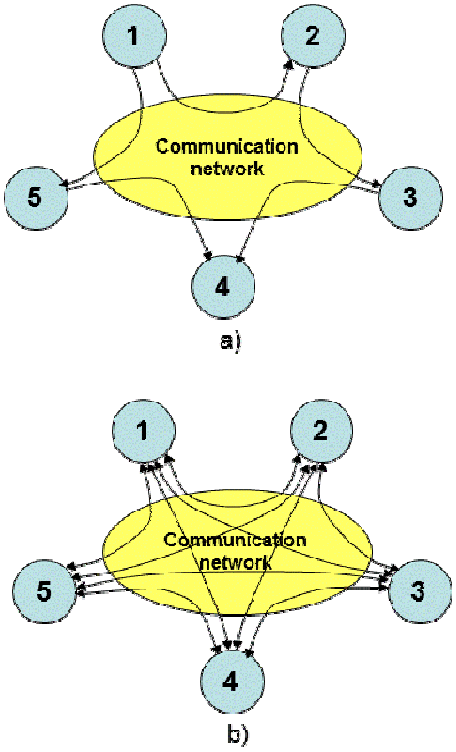

Figure 3. (a) unidirectional ring and (b) completely-connected virtual topologies

A. Transmission delays

In the first section of the paper we compared implicitly the time taken to cross N/2 virtual links to the time spent in waiting for the N/2-th transmission slot: however these two times do not need to be comparable. We shall use the following definitions:

- T_{slot} is the time between two successive packet transmissions performed by the Tx unit in Figure 2;
- T_{link} is the time needed to cross a virtual link, i.e. the time between the beginning of the transmission of a packet by Tx and the completion of its reception by the Rx block at the next node.

Note that, except for very particular implementations, only a fraction of the capacity of the access link towards the Communication Network is dedicated to the BlankNet VPN, while other applications can share the link: therefore, typically, T_{slot} is much larger than the transmission time of a packet. We shall assume that the transmission of BlankNet packets on the link has precedence over any other application, that is Tx preempts any other transmission process on the physical network interface.

T_{link} varies considerably depending on the underlying Communication Network: it can easily span two/three orders of magnitude going from a LAN to the Internet.

In order to characterize the various possible environments we define the parameter $r=T_{link} / T_{slot}$. In the following we shall consider only two cases, which in our view are the most meaningful:

- r close to 1, which may occur in an enterprise LAN environment: for instance, if we dedicate 10% of a 100 Mbps Ethernet link to the BlankNet, and transmit 10 kbit packets, T_{slot} would be 1 ms, which is comparable to T_{link} for an enterprise LAN;
- r around 100, which is a likely value in a public network environment: for instance, T_{link} may be 100 ms, and if we dedicate 10% of a 10 Mbps ADSL access link to the BlankNet, still transmitting 10 kbit packets, T_{slot} would be 1 ms.

Therefore our evaluations are carried out for r=1, r=10 and r=100.

B. Timing and synchronization

We do not make specific assumptions about the time taken by the various functional blocks in Figure 2, except that they are "fast enough". More precisely, for the purpose of our analysis, we suppose that, once a message is received, it can be transmitted in the next time slot (provided that no contention occurs at the TQ buffer). In other terms, T_{slot} is long enough to account for the local processing at each node.

Moreover, we make the simplifying hypotheses that r is an integer number, and that T_{slot} is the same for all the BlankNet nodes. Although these are clearly not very realistic, they do not have an impact on the relative performance of the various virtual topologies.

A further issue regards the relative synchronization of the various nodes in a BlankNet. The relevance of this problem can be shown by considering the complete connection virtual topology of Figure 3b. If all the nodes are completely independent, it is possible that all the other nodes transmit a packet to node 1 in the same time slot: this causes congestion at the input of node 1, adding a further delay to the message reception. This delay can be a large or negligible factor, with respect to the BlankNet performance, depending on the ratio of the VPN speed to the physical network speed.

If there is a global synchronization of transmission times similar problems can be effectively solved. However, for most part of following analysis, we shall assume that no global synchronization exists, but the congestion problem is negligible due to the high speed of the communication network and the limited degree of the majority of virtual topologies to be considered. In specific situations, we shall underline the advantages brought by synchronization.

C. Routing

The unidirectional ring is the only topology where there are no alternative routes to a destination; for instance, also in a completely connected topology it is possible to send a message to a different node, and then transmit it from there to the final node: this option could even be advantageous if the packet has a long time to wait for the scheduled transmission time through the direct link.

In the following we shall limit the routing options by means of these two rules:

1. a message is always transmitted to a node that is closer to destination in terms of (virtual) links to be crossed;
2. in case more than one node can be selected according to rule 1, the choice is made at random.

Rule 1 simplifies the routing operation and the analysis, and makes routing load-independent; rule 2 distributes traffic load evenly among all the possible routes.

IV. AN ALTERNATIVE VIRTUAL TOPOLOGY

In this section we will compare the different topologies in terms of the average time required to a message to reach its destination, if it does not find any other message ahead in the TQ buffers (light-loading). We define this metric as $D(0)=D_s+D_l$, where D_s is the time spent in the nodes waiting for the proper transmission slot, and D_l is the total time taken to cross all the links to destination. In the following we shall take T_{slot} as the time unit, therefore we have $D(0)=s+rd$, where s is the average total number of transmission slots spent waiting for transmission and d is the average number of links crossed. Note that, using terminology of graph theory, d is the average distance between two nodes of the topology graph.

It is easy to find out that for the ring we have $D(0)=rN/2$ and for the complete connection we have $D(0)=(N-2)/2+r$; because the complete connection is equal to the ring for $r=1$ and better in all the other cases, we shall not consider the ring in the following.

A. The binary cube

The binary cube topology comprises $N=2^k$ nodes, each one connected to k other nodes by bidirectional links. More precisely, every node can be identified by a bit string such as $b_{k-1}\ldots b_1 b_0$ and is directly connected to the k nodes $\underline{b}_{k-1}\ldots b_1 b_0$

 ISBN:978-1-61208-113-7

,…, $b_{k-1}…\underline{b_1}b_0$, $b_{k-1}…b_1\underline{b_0}$. Figure 4 depicts a binary cube with *N=16* (*k=4*).

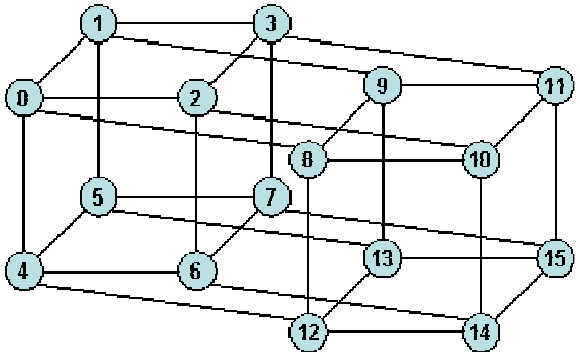

Figure 4. A four-dimensional binary cube

Starting from any node in a cube, we reach *k* new nodes crossing one link and $\binom{k}{i}$ new nodes after crossing the *i*-th link. Based on this consideration, it is simple to calculate the average distance in a binary cube, which is $d = \frac{k}{2}\frac{N}{N-1}$; moreover, because the nodes are supposed to work asynchronously, and we use rule 2 of the preceding section to decide on the next hop (instead of, for instance, sending the packet at the first time slot compatible with rule 1), we have $s = \frac{k-1}{2}d$. Therefore, for the binary cube, we obtain:

$$D(0) = \left(\frac{k-1}{2} + r\right)\frac{k}{2}\frac{N}{N-1} \qquad (1)$$

As we shall see in the next paragraph, a smarter routing rule and the synchronization of the nodes can create a significant performance advantage.

Table I reports the values of *D(0)* for the binary cube and the complete connection (CC) for various VPN sizes and *r* values.

TABLE I.

N	*r=1*		*r=10*		*r=100*	
	CC	*cube*	*CC*	*cube*	*CC*	*cube*
128	64	14,1	73	45,9	163	363,3
256	128	18,1	137	54,2	227	415,6
512	256	22,5	265	63,1	355	468,9
1024	512	27,5	521	72,6	611	523,0
2048	1024	33,0	1033	82,5	1123	577,8
4096	2048	39,0	2057	93,0	2147	633,2
8192	4096	45,5	4105	104,0	4195	689,1

It can be easily seen that the cube outperforms the complete connection in most use cases, the exceptions being for the combination of large *r* and small number of nodes (in grey on the table).

B. Synchronized operation of the binary cube

We can achieve a significant improvement of the cube performance if we synchronize globally the transmissions on the entire BlankNet. Let us assume that, in a specific time slot, all the transmissions are along virtual links corresponding to the same dimension of the cube, that is to say packets are sent towards destination nodes that differ from source nodes in the same bit position: Figure 5 depicts the four transmission phases (each lasting one T_{slot}) in the four-dimensional cube of Figure 4.

In order to understand the advantage of this global synchronization, let us consider at first the case of *r=1*. If *r=1*, it is guaranteed that a message, transmitted along a cube dimension, arrives at the beginning of the time slot corresponding to the next dimension in the transmission order. Therefore it is either transmitted immediately or waits for a time slot corresponding to a dimension it would not have crossed anyway.

As an example, assume the transmission order of Figure 5, and consider the path of a message entering at node 0 at the beginning of phase 2 and destined to node 11: transmission to node 2, wait one phase at node 2, transmission to node 10, transmission to node 11. This turns out to be the worst case, and requires *D(0)=k* phases, which compares quite favourably to the average values for the asynchronous cube.

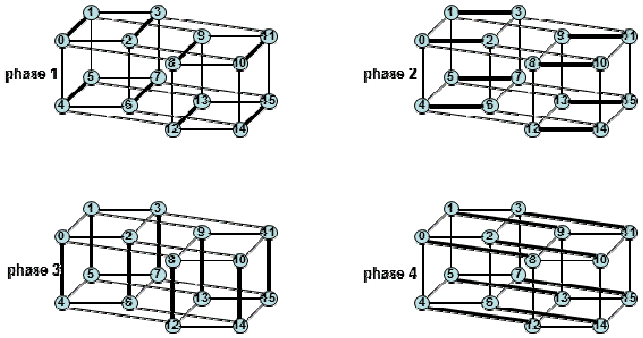

Figure 5. The four phases of synchronized transmission in a four-dimensional binary cube

Along these lines we can also derive an approximation for *D(0)* if *r>1*: it turns out that the same favourable synchronization occurs if *k* and *r* are mutually prime. In this case, a message from node *i* to node *j* would wait at most $s=k-d_{ij}$ slots (d_{ij} being the number of links to be crossed from *i* to *j*), therefore the delay formula for the synchronous binary cube can be upper bounded:

$$D(0) < k + (r-1)\frac{k}{2}\frac{N}{N-1} \qquad (2)$$

Of course the condition that *k* and *r* are mutually prime requires that r is deterministic, that is to say that the jitter introduced by the communication network is negligible with respect to T_{slot}. Because of this we assume that the synchronous behaviour is attainable only for small values of *r*. Table II compares the performance of the asynchronous cube and the above bound for the synchronous cube when

r=1, 3 and *5* (we excluded the cases where *r* and *k* are not mutually prime).

TABLE II.

N	*r=1*		*r=3*		*r=5*	
	asyn	*syn*	*asyn*	*syn*	*asyn*	*syn*
128	14,1	7	21,2	14,1	28,2	21,1
256	18,1	8	26,1	16,0	34,1	24,1
512	22,5	9	31,6	---	40,6	27,0
1024	27,5	10	37,5	20,0	47,5	---
2048	33,0	11	44,0	22,0	55,0	33,0
4096	39,0	12	51,0	---	63,0	36,0
8192	45,5	13	58,5	26,0	71,5	39,0

V. AN ANALYTICAL MODEL

In order to take in account the effect of traffic, we use a simple queuing model of the node.

The hypotheses made on the operations of the BlankNet node lead us to analyze the behavior of the TQ buffer, which can be modeled as a set of *k* independent queues, each one corresponding to a transmission phase (or, equivalently, to a virtual link).

Because the purpose of our evaluations is the comparison of virtual topology, we can assume a fairly simple traffic model: a message is introduced in the network by each BlankNet user with probability λ at each T_{slot} (it does not make sense to suppose a faster transmission rate by the user because the maximum transmission speed at each node is one message per T_{slot}). Messages are directed with equal probability to all possible destinations in the VPN: as a consequence, because of the isotropy of the topologies that we are analyzing, all the virtual links are equally loaded.

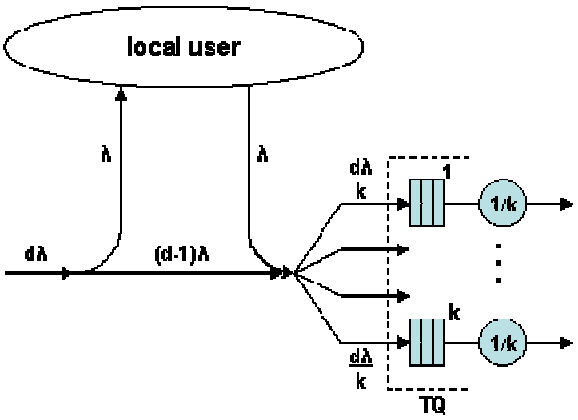

Figure 6. Distribution of the load on the virtual links in case of uniform traffic destination and isotropic topology

The load on each virtual link can be determined by a simple reasoning: every message introduced in the network crosses, in the average, *d* links before reaching its destination; therefore every physical link carries *dλ* messages per time unit (T_{slot}) in the average; because this load is equally divided into *k* virtual links, each having *1/k*-th of the total capacity, the load on every virtual link is *dλ* as well. Figure 6 depicts the assumptions above.

In the case of the binary cube, we approximate the behaviour of each virtual queue with a simple M/D/1 model, which gives the following expression for *W*, the average waiting time in each queue:

$$W = \frac{kd\lambda}{2(1-d\lambda)} \qquad (2)$$

It is immediately seen that the VPN reaches saturation when $\lambda=1/d$. At this point we can add a load-dependent factor to the overall time required to transfer a message to its destination, obtaining $D(\lambda)=D_q(\lambda)+D_s+D_t$, where $D_q(\lambda)=dW$. Therefore, for the (asynchronous) binary cube $D(\lambda)$ has the following expression:

$$D(\lambda) = \left(\frac{\lambda k^2}{2 \cdot \left(2\frac{N-1}{N} - \lambda k \right)} + \frac{k-1}{2} + r \right) \frac{k}{2} \frac{N}{N-1} \qquad (4)$$

For the complete connection we do not have transit traffic, and the queuing model that we use for each virtual queue is a more precise binomial/D/1: the queue has a service cycle lasting *N-1* time units, and an input process characterized by the following probability p_i of *i* arrivals (messages sent by the user) in a service cycle:

$$p_i = \binom{N-1}{i} \left[\frac{\lambda}{N-1} \right]^i \left[1 - \frac{\lambda}{N-1} \right]^{N-1-i} \qquad (5)$$

The waiting time for this model is known to be [6]:

$$W = \frac{N-2}{N-1} \cdot \frac{(N-1)\lambda}{2 \cdot (1-\lambda)} \qquad (6)$$

Hence the $D(\lambda)$ formula is:

$$D(\lambda) = \frac{N-2}{2} \left(1 + \frac{\lambda}{(1-\lambda)} \right) + r \qquad (7)$$

Figure 7 and Figure 8 compare the $D(\lambda)$ functions for the cube and the complete connection for a small (*N=256*) and a large (*N=4096*) VPN, with *r=1*.

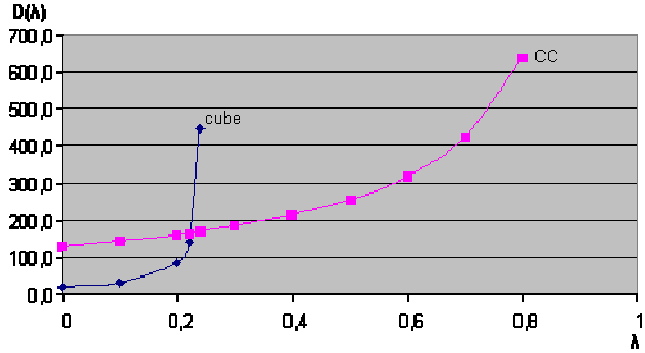

Figure 7. Delay function for *N=256, r=1*

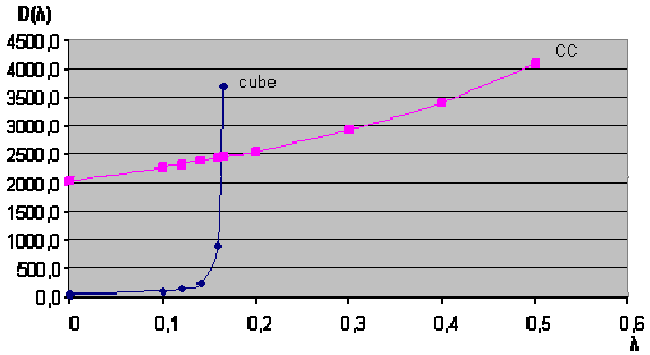

Figure 8. Delay function for *N=4096, r=1*

Figure 9 compares the two solutions with *r=100* for a large network (we know that for a small network the complete connection is superior even in light loading).

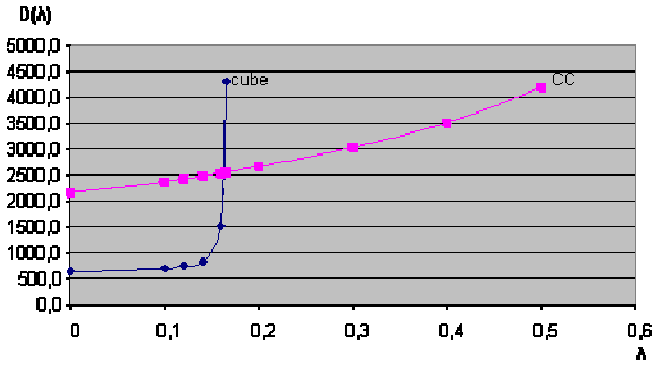

Figure 9. Delay function for *N=4096, r=100*

It is easy to see that, in all cases, the relative advantage of the two topologies changes drastically with the VPN loading, due to the effect of internal congestion which affects all topologies with *d>1*.

VI. Conclusion and Future Work

Our analysis has shown the existence of virtual topologies which perform much better than the ring and the complete connection as the basis of a BlankNet VPN.

However this advantage is present only for light-to-moderate loading conditions, after which we have to turn back to topologies with a very small average distance and to the complete connection in particular.

This work does not intend to suggest the use of the binary cube in practical situations, because it admits only configurations where *N* is a power of 2, a very unusual situation in a realistic application. However the characteristics of the binary cube suggest that several other candidate topologies exist, and in particular that we should work with graphs whose average distance grows logarithmically with the number of nodes (such as, for instance, the shuffle-exchange and its derivatives): these topologies, which can grow smoothly, shall be the objective of our further research.

The limitations of all these solutions under high loading suggest that we either implement a congestion control mechanism which prevents the BlankNet from working too close to the critical threshold $\lambda=1/d$, or that we develop a reconfiguration mechanism which allows the dynamic change of the virtual topology; while we are working on the first option, from a preliminary analysis the second one seems to introduce huge complexities without obtaining a significant payoff.

References

[1] www.torproject.org, checked on 06.01.2011

[2] Dabek, T. et al.: "Building Peer-to-Peer Systems with Chord, a Distributed Lookup Service," *Proc. 8th Workshop on Hot Topics in Operating Systems*, Elmau, Germany, May 2001.

[3] Bhuyan, L.N.: Qing Yang; Agrawal, D.P.; "Performance of multiprocessor interconnection networks," *Computer* , vol.22, no.2, pp.25-37, Feb 1989.

[4] Hayes, J.P.; Mudge, T.: "Hypercube supercomputers," *Proceedings of the IEEE* , vol.77, no.12, pp.1829-1841, Dec 1989.

[5] Karol, M.; Hluchyj, M.; Morgan, S.: "Input Versus Output Queueing on a Space-Division Packet Switch," *Communications, IEEE Transactions on* , vol.35, no.12, pp. 1347- 1356, Dec 1987.

[6] Meisling, T.: "Discrete-Time Queuing Theory," *Operations Research*, Vol. 6, no. 1, pp. 96-105, Jan. - Feb., 1958.

 ISBN:978-1-61208-113-7