Timely hybrid synchronous virtual networks

Rasha Hasan[1], **Fernando Luís Dotti**[1]

[1]Faculdade de Informática – Pontifícia Universidade Católica do Rio Grande do Sul
Av. Ipiranga, 6681 Prédio 32 sala 505, Porto Alegre, Brazil

rasha.hasan@acad.pucrs.br, fernando.dotti@pucrs.br

Abstract. *Network virtualization has been proposed in the last years, and it received special attention from both networking and distributed systems communities. However, specific applications' requirements, such as topology, security, and resilience, pose different challenges to the network embedding problem. Among these requirements lays the one of synchrony, that some applications demand time bounds for processing and communication. In this sense, Hybrid Synchrony Virtual Networks (HSVN) has been proposed to fulfill specific synchrony requirements and better support a considerable class of distributed systems. In our previous works, we gave attention to "space" HSVNs that was addressed to support hybrid synchrony systems in space. In this work, analogously, we discuss hybrid synchrony virtual networks for the "time" dimension. We regard the time HSVNs abstractions and techniques as being a refinement of the space HSVNs, since it further defines repeating time windows of synchrony while allowing the resource to behave asynchronously for a period of time. The timely hybrid model leads to the possibility of further sparing synchronous resources, if compared to the space model, as will be presented in this paper.*

1. Introduction

In the last three decades of research in Distributed Systems (DSs), one core aspect discussed is the one of synchrony. With an asynchronous system, we make no assumptions about process execution speeds and/or message delivery delays; with a synchronous system, we do make assumptions about these parameters [Schneider 1993]. In particular, in a synchronous system, the relative speeds of processes, as well the delays associated with communication channels are assumed to be bounded. The research on DSs has longly touched problems that base on the synchrony property of the system environment, for example the consensus problem [Dwork et al. 1988] where synchrony ensures progress of several distributed algorithms; another example is failure detection [Gartner 2001]. Many applications benefit from consensus and failure detection as building blocks in the distributed algorithms, for example Apache Cassandra [Hewitt 2011].

Synchrony in DSs impacts directly the complexity and functionality of fault-tolerant algorithms. Although a synchronous infrastructure contributes towards the development of simpler and reliable systems; yet such an infrastructure is too expensive and sometimes even not feasible to implement. On the other hand; a fully asynchronous infrastructure is more realistic, but some problems showed to be unsolvable in such an environment, such as the impossibility result by Fischer, Lynch and Paterson [Fischer et al. 1985]. The limitations in both fully synchronous or fully asynchronous systems have led researchers to the development of partial synchronous distributed systems, which we call in our work, as well, by Hybrid synchronous Distributed Systems. Hybrid models assume intermediate levels of synchrony. In [Veríssimo 2006], Veríssimo presented the wormhole model, that considers the *space* dimension for hybrid synchrony, where one part of a system would behave synchronously, while another part would be fully asynchronous. Cristian and Fetzer proposed the timed-asynchronous model [Cristian and Fetzer 1999], where the system alternates between synchronous and asynchronous behavior over *time*. Due to the hybrid behavior, both models are stronger than asynchronous model and weaker than synchronous one.

During our research path [Hasan et al. 2013, Hasan et al. 2014, De Oliveira et al. 2015], we argued that Virtual Networks (VNs) and a suitable VN embedding process offer suitable environment for running distributed applications with partial synchrony. This has led to the abstraction of new type of virtual networks that we name *The Hybrid Synchrony Virtual Networks (HSVNs)*. They are virtual networks that have subsets of nodes and links that obey time bounds for processing and communication. Our previous contributions treated the *Space_HSVNs* considering physical resources hybrid in space, i.e. resources behave either synchronously or asynchronously during all the time. The *Space_HSVNs* are addressed to the DSs of hybrid synchrony in *space*, which we call *space hybrid synchronous systems*.

The timed-asynchronous model assumes that the system alternate between synchronous and asynchronous behavior. More specifically, according to [Dwork et al. 1988] partially synchronous systems can alternate between synchronous and asynchronous behavior, being hybrid in *time*. For each execution, there is a time after which the upper bound δ is respected by the system. This time is called *Global Stabilization Time (GST)*. Since the upper bound cannot hold forever, it is accepted that it holds just for a limited time Δ_s. In practical terms, Δ_s is the time needed for consensus to make progress or to be reached. We call these *timely hybrid synchronous systems*.

Actually, both models (i.e., hybrid models in *space* and in *time*) are not completely excludent. If a resource has no ability to behave synchronously, then it offers no time guarantees at all. However if a resource is able to behave synchronously, the space model defines that it is always synchronous while the time model defines that it behaves eventually synchronous as described above. From the point of view of resources utilization, we can regard the time model as being a refinement of the space model, since it further defines repeated windows of synchrony while allowing the resources to behave asynchronously for a period of time. It is possible to support timely hybrid synchronous systems with Space_HSVNs, adopting the assumption and embedding model proposed in [Hasan et al. 2013, Hasan et al. 2014, De Oliveira et al. 2015], but this choice would result in wasting synchronous resources because of reserving them for virtual demands that would behave synchronously only eventually (i.e., only during time windows). With the goal of sparing synchronous resources, we propose new type of HSVNs, that is the Time_HSVNs suitable for the timely hybrid synchronous nature of certain DSs.

The main contributions of this paper are: *(i)* define the assumptions and abstractions needed to characterize both Substrate Networks (SNs) and Virtual Networks (VNs) suitable for Time_HSVNs, *(ii)* and develop an embedding model for Time_HSVNs that answers the timely synchronous nature of the system and aware of sparing synchronous resources which are relatively expensive. The rest of this paper will be organized as the following: Section 2 summarizes the related works; Section 3 details the abstractions and techniques we propose for the Time_HSVNs together with the embedding model; in Section 4 we evaluate the embedding model; and in Section 5 we conclude the work.

2. Related Work

The time-variant nature of networks has attracted considerable attention in the literature. Xie et al. [Xie et al. 2012] observed that during the networking intensive phases of applications, collision and competition occurs for the network resources, resulting in making the applications running time unpredictable. The uncertainty in execution time further translates into unpredictable cost, as tenants need to pay for the reserved virtual machines for the entire duration of their jobs. Xie et al. [Xie et al. 2012] propose the design of the first network abstraction (to the authors' best knowledge), TICV (Time Interleaved Virtual Clusters), that captures the time-varying nature of cloud applications, and they propose a systematic profiling-based methodology for making the abstraction practical and readily usable in today's data center networks. The network abstractions that [Xie et al. 2012] consider are similar to those proposed in [Guo et al. 2010, Ballani et al. 2011], but the last two works overlook the real time variant nature of resource requirements and simply assume that the customer will specify them somehow.

Zhang et al. conducted a series of research that considers the bandwidth (BW) variant nature of VNs during the process of resources provisioning, [Zhang et al. 2011, Zhang et al. 2012, Zhang et al. 2014]. The authors modeled the time-variant nature of the VNs demands as the combination of a *basic* sub-requirement, which exists all through the VNs lifetime, and a *variable* sub-requirement, which exists with a probability. For the basic sub-flows, fixed bandwidth is allocated (traditional BW sharing). But for the variable sub-flows, the authors consider specific design of the SN; they assume that the time is partitioned into frames of equal length, and each frame is further divided into slots of equal length. The authors develop two first-fit algorithms that map the variable sub-flows to the time slots on the SN. The first algorithm does not consider the inter-flows collision per slot, whereas the second algorithm is aware of it. The inter-flows collision per slot is calculated based on the probability of occurrence of the variable sub-flows.

In the topic of VNs mapping, we find that our work is near to the works concerned with delay constraints. For example, Zhang et al. [Zhang et al. 2010] propose a heuristic algorithm for mapping virtual multi-cast service-oriented networks subject to delay. Another work [Infuhr and Raidl 2011] addressed the VNs mapping problem with delay, routing and location constraints. Nevertheless, these works consider a homogeneous physical infrastructure and cannot be adopted for hybrid synchronous DSs that demand hybrid physical infrastructure. Moreover, these works do not consider applications with time-variant delay, both for processing speeds and messages communication.

Our work considers the synchrony time-variant nature of virtual networks, addressed for timely synchronous distributed systems. The problem raised in this paper (i.e., time_HSVNs embedding) could be solved using the models proposed in our previous works [Hasan et al. 2013, Hasan et al. 2014, De Oliveira et al. 2015] by overlooking the synchrony timely-variant nature of VNs, but this would result is reserving synchronous resources permanently for demands that require synchrony only during time windows. In this paper, we argue that, adopting suitable abstractions and techniques, together with a suitable embedding model, increases the resources usage efficiency.

3. Time HSVNs: abstractions and techniques

Formerly in this paper, we referred to DSs that demand synchrony eventually during the system life, in other words, they demand synchrony during periods of time to allow the progress of these systems. We assume that this kind of systems are supported by virtual networks that reflect the timely synchrony nature. We name this type of VNs the *Time Hybrid Synchronous Virtual Networks* abbreviated to *Time-HSVNs*. In this section we detail about *(i)* the Time-HSVNs characterization; *ii)* the SN suitable design to answer the Time_HSVNs demands; and *(iii)* the Time-HSVNs embedding over the adopted SN.

3.1. Time-HSVNs characterization

Time-HSVNs carry the common features of typical VNs [Chowdhury and Boutaba 2010], but in addition, they need to be further characterized to allow them to reflect the timely synchrony nature. We consider that the synchrony demands of each VN has a cyclic pattern with the cycle T time units. During T, each virtual node and link demands synchrony once, for a certain period of time. The time windows when the virtual element is provided synchrony is named *the synchronous rounds*, and the time windows when it is not provided synchrony is named *the asynchronous rounds*. We assume that the client is able to define the synchronous rounds he needs within T, and he is able to express it to the virtual network provider. The client needs to be provided synchrony, eventually within T, during the specified time duration he expresses, without caring for *when* it will be provided within T. The VNs cyclic pattern makes them reflect the nature of timely synchronous DSs, which repeatedly demand eventual synchrony during the system life.

3.2. SN design

We assume the existence of certain mechanisms that guarantee building physical network elements (nodes and links) that behave synchronously. These mechanisms can be related to the type of physical materials used, or to the procedures followed for configuring them, such as admission control and Quality of Service policies. The exact mechanisms for building synchronous resources is out of the scope of our work, but we assume their existence. In our previous works with Space_HSVNs [Hasan et al. 2013, Hasan et al. 2014, De Oliveira et al. 2015], we distinguished between two types of resources; synchronous and asynchronous, where both types maintain their synchrony status during the system life. In the current step of our work, Time_HSVNs demand a refinement of the Space_HSVNs abstractions and techniques to suit better the new view of synchrony (i.e., periodic eventual synchrony).

In [Zhang et al. 2011], Zhang et al. propose a bandwidth sharing technique that allocates bandwidth (BW) in accordance with VNs traffic fluctuation as detailed in the related works. The authors consider specific design of the SN as the following: the time is partitioned into frames of equal length, and each frame is further divided into slots of equal length. The authors develop an algorithm that maps the variable sub-flows to the time slots on the SN in a way that the sub-flows maped to the same slot do not violate the BW capacity, neither exceed a collision threshold allowed, where the sub-flows collision is calculated based on the probability of their occurrence. The proposed methodology allows an opportunistic bandwidth sharing between the sub-flows.

In our work, we inspire a suitable SN design from the work of Zhang et al. [Zhang et al. 2011] after adapting it in what matches our problem: *i)* in our work the virtual flows are of fixed BW demand during time (not opportunistic demands), thus, we disconsider collision probability; *ii)* the HSVNs demand synchrony once during T, see 3.1, so, we need only one time window during T that applies Zhang et al. technique. We name this time window *synchronous frame*. The synchronous frame is further partitioned into time slots of equal size, we name them *synchronous slots*; *iii)* the virtual demands mapped to a synchronous slot should not violate the physical BW capacity to eliminate competition and assure synchrony. The length of the synchronous frame and the number of time slots within a frame is related to the VNs number and demands. We assume that each virtual node and link do not demand synchrony slots that exceed the number of slots per synchronous frame.

3.3. Time_HSVNs Embedding

The virtualization architecture we adopt is the one proposed by Schaffrath et al. [Schaffrath et al. 2009]. We assume that the virtual network provider (VNP) has complete information about: *i)* the SN topology and its attributes (nodes Central Processing Unit (CPU), links bandwidth (BW), and synchronous slots number and length), and *ii)* the virtual networks topology and demands (nodes CPU, links BW, synchrony demands). The VNP receives the synchrony demands in term of time period, and translates it into number of synchronous slots of the SN slots. We deal with the case of off-line VNs embedding. The time_HSVNs will be provided synchrony during the synchronous frame. Out of the synchronous frame, additional asynchronous demands can be mapped and competition can occur. This does not pose any problem for demands that do not expect synchrony.

The Time_HSVNs embedding problem can be stated as the following: *How to map the virtual synchronous slots to the physical synchronous slots, with the objective of minimizing the mapping cost represented by the used BW?* The approach we followed for solving the Time_HSVNs was to benefit from our previous works on Space_HSVNs [Hasan et al. 2013, Hasan et al. 2014, De Oliveira et al. 2015], by refining the proposed model for Space_HSVNs to a new version that expresses the synchronous slots. Further, we enhanced the achieved solution to allow more VNs to be mapped on the same SN. So, the Time_HSVNs mapping would go through two phases as the following:

1. the macro mapping phase: This phase maps the virtual elements (nodes and links) to the physical resources that can support them. This phase leads to a mapping solution, that considers minimizing the physical bandwidth consumption. At the end of this phase, each virtual node and link will be mapped to a physical node and path that can support them. The macro mapping phase model is achieved by refining the Space_HSVNs mapping model to express synchronous slots.
2. the micro mapping phase: This phase increases the efficiency of the solution achieved in the macro mapping phase, by allowing embedding possible future VNs demands on the same given SN. this phase is performed individually for each physical node and link used in the macro mapping phase. The micro mapping phase maps the virtual synchronous demands to the physical synchronous

slots. For solving the micro mapping phase, we adopted an off-the-shelf problem from the literature due to its similarity, that is the Cutting Stock problem (CSP).

3.4. The Macro Mapping Phase

The inputs of this phase are: *(i)* the SN topology and attributes, and *(ii)* the VNs topologies and demands. And the output of this phase will be assigning each virtual node to one physical node and each virtual link to a physical path, where a path can be composed of one link or more. At this phase, the problem turns to be: *how to map the VNs on top of the SN with the least physical bandwidth consumption possible.* We formulate the macro mapping problem in the shape of a Integer Linear Program (ILP).

Variables definition - The SN is represented by an undirected graph $G(N, L)$, composed of a set of physical nodes N and links L. Analogously, each virtual network VN^k belonging to the set of virtual networks VN will be presented by an undirected graph $G^k(N^k, L^k)$. The number of synchronous slots provided by the physical node i and physical link (i, j) are $sync(i)$ and $sync(i, j)$. Analogously, $sync(i^k)$ and $sync(i^k, j^k)$ are the number of synchronous slots demanded by the virtual node i^k and link(i^k, j^k). Besides synchrony, two other attributes are considered for the SN and VN elements: nodes CPU, and links bandwidth (BW). The syntax for those attributes on the SN and VN respectively are: $cpu(i)$, $bw(i, j)$, $cpu(i^k)$, and $bw(i^k, j^k)$. Finally, we define the model output variables, they are: a binary function $\sigma(i^k, i)$ that expresses whether node $i \in N$ maps node $i^k \in N^k$, and a binary function $\rho(i^k, j^k, i, j)$ that expresses whether link (i, k) is part of the physical path that maps the virtual link (i^k, j^k). After solving the macro mapping model, each physical node i is mapping a set of virtual nodes $N^k(i)$, and each physical link (i, j) on the SN is mapping a set of virtual links $L^k(i, j)$.

The Macro mathematical model - It is formulated in the shape of an ILP as bellow:

Mapping objective- The Objective Function (1), we consider is inspired from [Hasan et al. 2013], which is to minimize the total bandwidth used.

Objective: minimize

$$\textstyle\sum_{\forall VN^k \in VN} \sum_{\forall (i^k, j^k) \in L^k} \rho(i^k, j^k, i, j) \cdot sync(i^k, j^k) \cdot bw(i^k, j^k); \tag{1}$$

Mapping constraints-
- *Capacity constraints:*
for every $(i, j) \in L$ and every $(i^k, j^k) \in L^k$

$$\rho(i^k, j^k, i, j) \cdot bw(i^k, j^k) \leq bw(i, j) \tag{2}$$

for every $i \in N$ and every $i^k \in N^k$

$$\sigma(i^k, i) \cdot cpu(i^k) \leq cpu(i) \tag{3}$$

- *Nodes mapping constraints:*
for every $VN^k \in VN$, $i^k \in N^k$

$$\sum_{\forall i \in N} \sigma(i^k, i) = 1 \tag{4}$$

for every $VN^k \in VN$, $i \in N$

$$\sum_{\forall i^k \in N^k} \sigma(i^k, i) \leq 1 \tag{5}$$

- *Links mapping constraint:*
for every $VN^k \in VN$, $(i^k, j^k) \in L^k$, $i \in N$

$$\sum_{\forall j \in N} \rho(i^k, j^k, i, j) - \sum_{\forall j \in N} \rho(i^k, j^k, j, i) = \sigma(i^k, i) - \sigma(j^k, i) \tag{6}$$

- *Nodes synchrony constraints:*
for every $i \in N$

$$\sum_{\forall VN^k \in VN} \sum_{\forall (i^k) \in N^k} \sigma(i^k, i) \cdot sync(i^k) \leq sync(i) \tag{7}$$

- *Links synchrony constraints:*
for every $(i, j) \in L$

$$\sum_{\forall VN^k \in VN} \sum_{\forall (i^k, j^k) \in L^k} \rho(i^k, j^k, i, j) \cdot sync(i^k, j^k) \leq sync(i, j) \tag{8}$$

The capacity constraint (2) assures that the bandwidth of every virtual link does not exceed the bandwidth of the physical link mapping it. Similarly, constraint (3) is regarding nodes CPU. The node mapping constraint (4) assures that each virtual node is mapped once on a physical node. Constraint (5) assures that virtual nodes belonging to the same VN are not mapped on the same physical node. This is to achieve load balancing besides improving the reliability. This procedure minimizes the number of virtual nodes prone to failure by a physical node failure. For any virtual link (a, b), the links mapping constraint (6), adopted from [Zhang et al. 2010], assures the creation of a valid physical path. When mapping a set of virtual nodes $N^k(i)$ on one physical node i; the nodes synchrony constraint (7) assures that the number of virtual slots mapped on i do not exceed the number of synchrony slots provided by i. This constraint considers the worst case, when each virtual slot requires a complete synchrony slot alone without sharing. Similarly, constraint (8) represents the equivalent restriction regarding links synchrony.

3.5. The Micro mapping phase

This phase increases the efficiency of the solution achieved in the macro phase, by allowing embedding possible future VNs demands on the same SN. This is achieved by scheduling the synchronous demands efficiently within the synchronous frame. By viewing the problem at this stage as an optimization problem, the problem turns to be: *how to schedule the virtual demands within a synchronous frame minimizing the number of synchronous slots used.* Revising the literature, we found a very similar problem that is *the Cutting Stock Problem (CSP)* [Haessler and Sweeney 1991], which is one of the NP-hard problems cited by KARP [Karp 1972], and from the cutting stock problem we inspired the solution of the timely HSVNs micro mapping problem.

In operations research, the cutting-stock problem is the problem of cutting standard-sized pieces of stock material (e.g., paper rolls or sheet metal) into pieces of specified sizes while minimizing material wasted. Translating the CSP elements into the micro mapping problem: the *specified sized pieces* are the synchronous slots demanded by the VNs; the *standard-sized pieces of stock material* is the synchronous frame on SN; *a pattern* is the set of demands accepted within a stock in the CSP, and it will be the set of virtual slots accepted within the physical slot in the micro mapping problem; and the objective of the CSP in to *minimizing the stock waste* and in our problem it would be to minimize the number of synchronous slots used within the synchronous frame.

The Micro mathematical model - We express it bellow for links but it goes similarly for nodes. Consider one physical link (i, j) with a synchronous frame of $sync(i, j)$ slots, that maps a set of virtual links $L^k(i, j)$, where every virtual link has two attributes: the number of synchronous slots demanded $sync(i^k, j^k)$ and the capacity of BW demanded $bw(i^k, j^k)$. Within each physical slot, the virtual synchronous slots that can be mapped to it form a pattern X_j [Haessler and Sweeney 1991]. The patterns are formed based on the BW of the virtual demands compared with the physical link BW. These patterns are the input to the micro model. The micro model aims at minimizing the number of synchronous slots used within the synchronous frame which is achieved by minimizing the total number of patterns (Equation (9)). Assuming a_{ij} is the number of times order i appears in pattern j [Haessler and Sweeney 1991]; constraint (10) assures providing every virtual link with a number of synchronous slots that is at minimum equal to the number of synchronous slots demanded $sync(i^k, j^k)$. The output of the micro mapping model will be telling which are the used patterns, and how many of each pattern is needed.

Objective: minimize

$$\sum\nolimits_{\forall j \in PATTERNS}(X_j) \tag{9}$$

Cutting constraint
for every $i \in L^k(i, j)$

$$\sum_{\forall j \in PATTERNS} (a_{ij} \cdot X_j) \geq sync(i^k, j^k) \tag{10}$$

After performing the micro mapping phase, the SN is updated (used BW and CPU is substracted from the SN nodes and links capacity), and the macro mapping phase can run again, allowing more VNs to be mapped on the same SN. The combination on the macro phase, the micro phase, and the SN updating we name *an optimization cycle* (Opt_cyc). Figure 1 illustrates two optimization cycles for mapping groups of virtual links on one physical link. The physical link updating happens either by reducing its capacity, or reducing the number of synchronous slots it supports. Either way, there will be a waste in the physical bandwidth, we refer to the bandwidth waste resulted by reducing the physical capacity W_h, and by reducing the synchronous slots W_v. The updating approach chosen is the one with the least waste (the smaller value between W_h and W_v is marked with a star * in the figure). This updating methodology we follow is because the macro mapping phase (at the beginning of each optimization cycle) considers physical slots of equal capacity and fully empty.

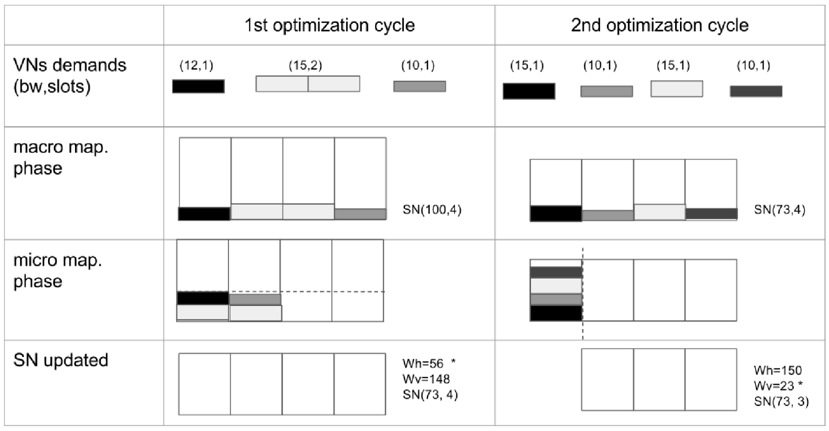

Figure 1. An illustrative scheme for the HSVNs optimization cycles

4. Performance Evaluation

The Time_HSVNs abstractions we considered together with the Time_HSVNs embedding model are supposed to lead to further sparing of synchronous resources, if compared to the space model. We run preliminary experiments that allow investigating the performance of the proposed embedding approach of Time_HSVNs. The aspects considered during the analysis of our model are: *(i)* the embedding cost; *(ii)* the physical resources load; *(iii)* the optimization time; *(iv)* the topology of the physical subnetwork composed of the used resources; and *(v)* the micro mapping phase efficiency.

Experiments were designed as a full factorial [Jain 1991], exploring all possible combinations between the networks parameters. Such choice of experiments was done by other works like [Bays et al. 2012]. Similar to [Yu et al. 2008, Bays et al. 2012], physical and virtual networks were randomly generated. For this we used BRITE tool (Boston university Representative Internet Topology gEnerator) [Medina et al.] with Waxman model [Waxman 1988]. We implemented the mathematical model with ZIMPL language (Zuse Institute Mathematical Programming Language) [Koch 2004], both for the macro and micro mapping phases, and we used CPLEX [IBM] to solve the Integer Program (IP), running on a computer Intel HM75, Core i3-3217U 1.80 GHz (Giga Hertz), cash 3 MB (Mega Byte), Random Access Memory (RAM) of 2 GB (Giga Byte), DDR3 and operating system Xubuntu 14.04.

In all the following experiments, the substrate network size was fixed to 15 nodes. Initially all CPUs (Central Processing Unit) of SN nodes are free, and links Band Width (BW) is uniformly distributed between 1-3 Gbps (Giga bit per second). We ran twelve experiments divided into three groups, A, B and C, with VNs total size of 10, 20, and 30 nodes in each group respectively. The VNs were generated with 3, 4, or 5 nodes each, and CPU demands 10%, 15%, or 25% of the SN nodes CPU capacity, and BW demands uniformly distributed between 100 Mbps (Mega bit per second) and 1 Gbps. VN nodes demand one synchronous slot per T. In scenarios 1, 2, 3 and 4 of each group, the virtual links synchronous slots demanded varies between 1, 2 , 3 , and 4 slots. The SN provides periodically, each $T = 20$ seconds, a synchronous frame of 4 seconds length, divided into four equal time slots. Table 1 details the experiments parameters.

The first parameter evaluated is the mapping cost represented by the used BW,

Table 1. Experiments parameters

Expe.	A1	A2	A3	A4	B1	B2	B3	B4	C1	C2	C3	C4
VN size	10 nodes				20 nodes				30 nodes			
Virtual links sync. slots	1	2	3	4	1	2	3	4	1	2	3	4
Virtual nodes sync.	1 slot per T											
each VN size	3,4,5 nodes											
VNs BW	uniformly distributed: 100Mbps-1Gbps											
VNs CPU	10,15,25 % of SN nodes CPU											
SN size	15 nodes											
SN BW	uniformly distributed: 1 Gbps-3 Gbps											
SN CPU	nodes fully free initially											

which is the model objective function. We evaluate our results by comparing them with the BW used in case the experiments in Table 1 were mapped using the space model and the SN previously addressed in [Hasan et al. 2013], see Figure 2. With the time model, the used BW indicated is the one consumed within the synchrony frame. To allow just comparison, we calculate the BW used in the space model during time window equal to the frame length. We note down the following main observations:*(i)* within one experiments group, the used BW is proportional to the number of synchronous slots demanded by the VNs. For example, in scenario A2 the BW used is 13.624 Mbps/frame, and it is 20.436 in scenario A3, the proportion between the two figures is the proportion of the number of synchronous slots demanded in each scenario 2/3. *(ii)* by comparing the counterparts experiments of the three groups (e.g., A2, B2, and C2) we notice that the BW used increases. This is due to the VNs size increment, which tends naturally to reserve more resources. *(iii)* within each experiment's group, the BW used with the space model is equal to the maximum BW used within the group (i.e., experiment 4 of each group). *(iv)* the time model is more efficient as it spares more resources. For example, mapping the VNs in scenario B3 with the time model spares 33,33% of the resources needed when mapping the same set of VNs with the space model. And the spared ratio increases when the synchronous demands within T decreases, e.g. in B2, the time model spares 100% of the resources, and in B1 spares 300%.

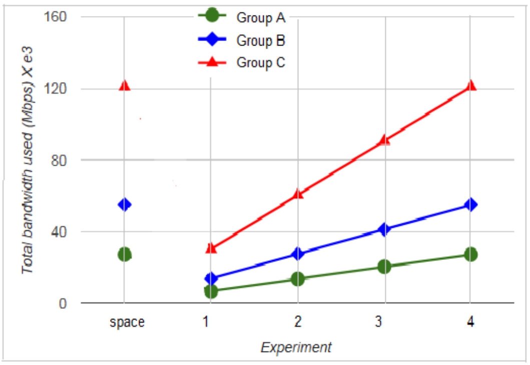

Figure 2. Used bandwidth

The second parameter evaluated is the physical resources load. This study is useful when done on a scenario that can possibly load the SN. We chose scenario C2 (VNs of big size). Figure 3 depicts the Cumulative Distributed Function (CDF) for physical nodes and links in experiment C2. We note that 80% of the SN nodes had a load that varies between 10% and 60%, only 13.33% had a load between 60% and 80%, and no nodes were highly loaded more than 80%. And regarding the physical links, we note that, 41.37% of the SN links had a load that ranges between 10% and 60%, and only 14% of the physical links had a load that exceeded 60%, and no links were fully loaded. So, the SN resources seem to have load distribution which is good, since concentrating the load in certain elements will result in congestion, leading to block mapping certain VNs in the future. This is achieved because, the proposed model does not push the mapping process to exhaust the used physical resources before allocating new ones, rather, all resources are given the same chance to be chosen, as long as they allow mapping on the shortest path.

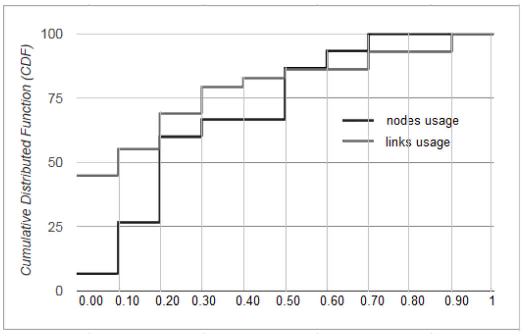

Figure 3. CDF for resource usage in experiment C2

The third parameter evaluated is the mapping time. The optimization process reached its end with scenarios A and B. Whereas in scenarios C, the optimization was terminated with optimization gap less than 2%. We took this decision when the optimization progressed slowly without much gain. For example, in scenario C3, it took 30 minutes to reach a solution with 4.46% gap, then another 33 minutes to reach another solution with 1.86% gap. In realistic scenarios, the client might prefer a semi-optimal solution in a short computational time, than an optimal solution after long time. From Table 2, we notice that, *(i)* most of the scenarios demanded optimization time that is less than 20 minutes, which is an acceptable computational time; *(ii)* we notice that the time model demands more time than the space model. And we notice that the difference between both increases with the increment of the problem size (i.e., VNs size and synchronous slots demanded) which increases the number of variables that need to be solved by the optimization process.

Table 2. Embedding time (in minutes)

Group.	space	exp.1	exp.2	exp.3	exp. 4
A	0.13	0.07	0.08	0.09	0.19
B	0.75	1.46	1.16	5.31	18.13
C	8	15.09	46.55	65.95	38.89

The fourth parameter studied is the topology of the physical subnetwork composed of the physical used elements (i.e., nodes and links). Figure 4 illustrates this topology for scenarios A1, A2, and A3. The topologies under study are the ones in red. We notice that, even though all these scenarios with the same VNs size (10 nodes), yet the model tends towards reserving more physical elements with the increment of the synchronous slots demanded by the VNs. For example, in scenario A1 the physical subnetwork under study is composed of 6 nodes, whereas in scenario A3 it is composed of 10 nodes. Previously, we noted down that the model tends towards distributing the BW load on the physical resources, Figure 3. Now we add that the model tends also towards distributing the synchrony load as well. So, when the VNs increase their synchrony demands (i.e., number of synchrony slots), the model tends towards reserving new elements. This behavior avoids congesting the synchronous frames of the used elements, allowing mapping new arriving VNs. Because the time HSVNs will be blocked or by exhausting the SN CPU and BW, or by exhausting the SN synchronous slots.

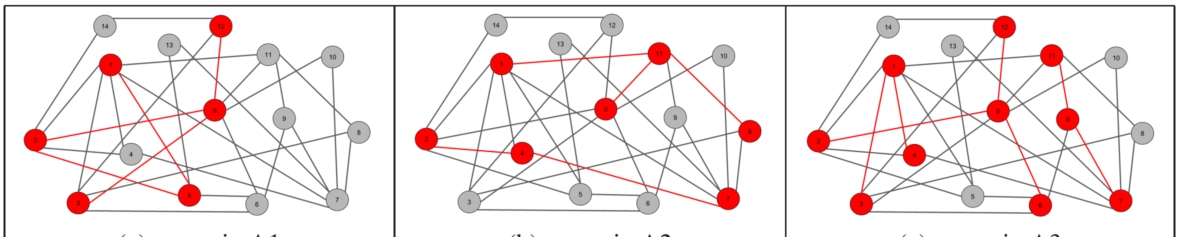
(a) scenario A1 (b) scenario A2 (c) scenario A3

Figure 4. Topology divergence of used physical resources

The fifth parameter studied is the micro mapping model efficiency represented by the number of VNs accepted. We run this study on the case of one physical link and an endless queue of virtual links attended in order. We consider five experiment groups with different load range, Table 3. We run three experiments per group, with different synchrony demands. In each experiment several opt_cyc are performed till the physical link is exhausted and no demands are accepted. Our main observations: *i)* the efficiency decreases when the virtual links load increases. For example, the efficiency in group K was 9, 13, and 16 whereas in group L it was 6, 8, and 12. *ii)* the efficiency increases when the maximum number of synchronous slots demanded decreases. For example, in group K, when the maximum number of synchronous slots demanded decreased from 3 to 1, the model efficiency increased from 9 to 16; *iii)* the micro model efficiency is the same in group N and O, the reason is that, both groups are with high virtual links load, this does not allow slots sharing between virtual demands, and the mapping solution achieved in the macro mapping phase cannot be optimized further with the micro mapping phase.

Table 3. Number of mapped virtual links with different load and synchrony demands

Scenario	**VNs load/SN capacity**	$sync(i^k, j^k)$**=1,2,3 slots**	$sync(i^k, j^k)$**=1,2 slots**	$sync(i^k, j^k)$**=1 slot**
K	(0-20] %	9	13	16
L	(20-40] %	6	8	12
M	(40-60] %	5	6	8
N	(60-80] %	1	2	4
O	(80-100] %	1	2	4

5. Conclusion

In our previous works, we gave attention to *space* HSVNs that was addressed to support hybrid synchrony systems in space. In this work, analogously, we discuss hybrid synchrony virtual networks for the *time* dimension. The main contributions of this paper are: *i)* characterize the time_HSVNs to reflect the synchrony time-variant nature; *ii)* adopt a suitable design for the SN to support the demands; and *iii)* propose a resources allocation model. Simulation results show that the proposed embedding framework answers the synchrony time-variant demands efficiently (spares synchronous resources), distributes the load over the physical resources (nodes and links), and has an acceptable computation time for reaching the embedding solution. In addition, topological study of the subnetworks composed of the used resources on the SN showed that, the embedding model is aware of the synchronous demands variation, and the resulting subnetwork scatters more on top of the SN when the synchronous demands increase. Further study of the micro-mapping phase showed that its efficiency is a function of the VNs load and synchrony.

At this phase of our work, we consider that the synchrony pattern of each virtual node and link is independent of each other, in future works, we intend to consider cases when VNs elements synchrony is mutually dependent, to reflect better the nature of the time hybrid synchronous DSs.

Acknowledgement

This work was supported in part by CAPES PVE - Grant No. 88881.062190/2014-01.

References

Ballani, H., Karagiannis, T., and Rowstron, A. I. T. (2011). Towards predictable datacenter networks. In *Proc. of the 11th ACM SIGCOMM conference*, pages 242–253.

Bays, L. R., Oliveira, R. R., Buriol, L. S., Barcellos, M. P., and Gaspary, L. P. (2012). Security-aware optimal resource allocation for virtual network embedding. In *the 8th International Conference on Network and Service Management (CNSM)*.

Chowdhury, N. M. and Boutaba, R. (2010). A survey of network virtualization. *Computer Networks*, 54(5):862–876.

Cristian, F. and Fetzer, C. (1999). The timed asynchronous distributed system model. *IEEE Trans. on Parallel and Distributed Systems*, 10(6):642–657.

De Oliveira, R. R., Dotti, F. B., and Hasan, R. (2015). Heuristicas para mapeamento deredes virtuais de sincronia hibrida. In *33o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 291–304.

Dwork, C., Lynch, N., and Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382.

Gartner, F. C. (2001). A gentle introduction to failure detectors and related subjects. Technical report, Darmstadt University of Technology, Department of Computer Science.

Guo, C., Lu, G., Wang, H. J., Yang, S., and Kong, C. (2010). Secondnet: a data center network virtualization architecture with bandwidth guarantees. In *ACM CoNEXT*.

Haessler, R. W. and Sweeney, P. E. (1991). Cutting stock problems and solution procedures. *European Journal of Operational Research*, 54:141–150.

Hasan, R., Mendizabal, O. M., and Dotti, F. L. (2013). Hybrid synchrony virtual networks: Definition and embedding. In *The Thirteenth International Conf. on Networks*.

Hasan, R., Mendizabal, O. M., Oliveira, R. R. D., and Dotti, F. L. (2014). A study on substrate network synchrony demands to support hybrid synchrony virtual networks. In *32o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*.

Hewitt, E. (2011). *Cassandra: The Definitive Guide*. OReilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

IBM. Cplex. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer.

Infuhr, J. and Raidl, G. R. (2011). Introducing the virtual network mapping problem with delay, routing and location constraints. In *Proc. of 5th International Networking Optimization Conference (INOC)*.

Jain, R. (1991). *Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling*, chapter 16. Wiley Computer Publishing.

Karp, R. M. (1972). *Complexity of Computer Computations*, chapter Reducibility Among Combinatorial Problems, pages 85–103. Springer.

Koch, T. (2004). *Rapid mathematical programming*. PhD thesis, Tichnische Universität Berlin.

Medina, A., Lakhina, A., Matta, I., and Byers, J. Brite: Boston university representative internet topology generator. In *Available online: http://www.cs.bu.edu/brite*.

Schaffrath, G., Werle, C., Papadimitriou, P., Feldmann, A., Bless, R., Greenhalgh, A., Wundsam, A., Kind, M., and Maennel, O. (2009). Network virtualization architecture: Proposal and initial prototype. In *Proc. of 1st ACM workshop on virtualized infrastructre systems and architectures, ser. VISA'09*, pages 63–72, New York, NY, USA.

Schneider, F. B. (1993). *Distributed systems (2nd Ed.)*, chapter 2 (What Good are Models and What Models are Good?). Wesley Publishing Co.

Veríssimo, P. E. (2006). Travelling through wormholes: a new look at distributed systems models. *ACM SIGACT News*, 37(1):66–81.

Waxman, B. M. (1988). Routing of multipoint connections. *Selected Areas in Communications*, 6(9).

Xie, D., Ding, N., and Hu, Y.C. andKompella, R. (2012). The only constant is change: Incorporating time-varying network reservations in data centers. In *SIGCOMM 12*.

Yu, M., Yi, Y., Rexford, J., and Chiang, M. (2008). Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM-SIGCOMM*, 38(2).

Zhang, M., Wu, C., Jiang, M., and Yang, Q. (2010). Mapping multicast service-oriented virtual networks with delay and delay variation constraints. In *in Proc. to IEEE Global Telecommunication Conference (GLOBECOM)*, Miami, FL.

Zhang, S., Qian, Z., Tang, B., Wu, J., and Lu, S. (2011). Opportunistic bandwidth sharing for virtual network mapping. In *Global Telecommunications Conference*, pages 1–5.

Zhang, S., Qian, Z., Wu, J., and Lu, S. (2012). An opportunistic resource sharing and topology-aware mapping framework for virtual networks. In *Proc. to IEEE INFOCOM conference*, pages 2408–2416.

Zhang, S., Qian, Z., Wu, J., Lu, S., and Epstein, L. (2014). Virtuail network embedding with opportunistic resource sharing. *IEEE Journal of Transactions on Parallel and Distributed Systems*, 25(3):816–827.