A study on substrate network synchrony demands to support hybrid synchrony virtual networks

Rasha Ghassan Hasan[1], **Odorico Machado Mendizabal**[1], **Rômulo Reis De Oliveira**[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, odoricomendizabal@furg.br,

romulo.reis@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. By offering a flexible and economic alternative for the deployment of customized networks, a wide set of applications becomes eligible to run on top of such infrastructures. 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 fulfil specific synchrony requirements and better support a considerable class of distributed systems. Considering HSVNs, one of the main challenges is to minimize the need for synchronous components in the substrate network, due to their high cost. In this paper, we propose a mathematical model that aims at defining an economic synchrony configuration of the substrate network, subject to the virtual networks demands, and we analyse the physical synchronous resources regarding their properties and topology.*

1. Introduction

Network virtualization has been proposed as a flexible and economic approach to build multiple virtual networks over a shared substrate [Chowdhury et al. 2012]. As can be seen in the literature, the diversity of applications pose different challenges for the VNs embedding process, *i.e.* how resources of a physical network (a.k.a. Substrate Network – SN) are used to support VNs. For example, topology [Zhu and Ammar 2006], security [Bays et al. 2012], resilience [Yu et al. 2011], and synchrony [Hasan et al. 2013] requirements.

On a recent work [Hasan et al. 2013], we proposed a new architecture based on VNs for distributed applications that require hybrid synchrony. Those distributed applications may demand a subset of resources to behave synchronously. Thus, the SN has to support such subsets of resources that ensure time bounds on communication and processing. The resulting architecture that combines *Hybrid Synchrony (HS)* with *Virtual Networks (VN)* we name *HSVN*. The SN is composed by two classes of nodes: (i) *synchronous nodes*, which allow timely CPU execution, achieved through implementing periodical real-time tasks, and (ii) *asynchronous nodes*, that have no timely guarantees. Analogously, two classes of the SN physical links are available: (i) *synchronous links*, which

ensure time-bounded messages transmission delay, achieved through implementing QoS communication guarantees, and (ii) *asynchronous links* that have no timely guarantees.

From the system developer standpoint, delegating the resource allocation to the VN mapping process is very advantageous. As well as CPU capacity or bandwidth, the synchrony level demanded by applications is also considered by the mapping process in searching for an economic sharing of resources. From the network virtualization standpoint, the main advantage of HSVN, compared to classical VNs architecture, is the coexistence of synchronous and asynchronous components. By providing this setup, distributed applications running on top of HSVNs can rely on the hybrid synchronous assumption [Cristian and Fetzer 1999, Veríssimo 2006]. Therefore, stronger properties provided by synchronous components can be enjoyed by the application as a whole, even though only some (possibly the majority) of HSVN's components are asynchronous. As discussed in Section 2, important classes of applications can benefit from hybrid architectures.

Although virtualization drops considerably the cost with synchronous components, the HSVN embedding process presented in [Hasan et al. 2013] does not take into account the amount of synchrony demanded by HSVNs. Rather, a fixed number of synchronous components in the SN need to be booked in advance. Since the synchronous components building cost is remarkably higher than the asynchronous ones, a key challenge is to efficiently handle HSVN's synchronous requirements.

In this paper, we consider the synchrony demands on the HSVNs as an input for the embedding process. With this information, the embedding process determines a sufficient number of synchronous components, and their location on the SN, so that the VNs requirements are satisfied. The mapping model aims at minimizing the synchronous resources made available by the SN. With such a model, it is possible to define exactly which physical resources have to be synchronous, and then configure them to preserve timely guarantees (*e.g.* by scheduling real-time tasks for nodes, and applying QoS policies for links). Differently from results presented in [Hasan et al. 2013], the embedding process proposed in this work avoids reservation of unused synchronous resources at the SN. Moreover, this evaluation shows levels of synchrony that will be demanded to a SN.

The rest of this paper is organised as follows. Section 2 motivates the use of hybrid synchronous models and virtual networks, Section 3 presents our proposed mathematical model, that aims at configuring the SN in terms of synchrony, and maps the HSVN on it. Section 4 details about the performance evaluation and gives important insights about the topology and features of physical resources chosen to be synchronous on the SN. Finally, Section 5 describes related work, and Section 6 concludes this paper.

2. Hybrid Synchronous Models and Virtual Networks

The development of distributed systems is strongly dependent on the assumptions about the environment. By understanding how system components and the surrounded environment behaves, system developers are capable to design algorithms that are correct under certain assumptions.

Generally, stronger assumptions about system's components and the environment allow the development of simpler solutions. The downside is that strong assumptions do not represent with sufficient accuracy a wide set of real environments. Conversely,

weaker assumptions require complex solutions and, in some cases, some problems are even impossible to solve under weaker assumptions. An advantage, though, is that weaker assumptions usually describe realistic environments and solutions proved to be correct under weaker assumptions are also correct under stronger assumptions.

In practical terms, components of a given infrastructure can have different properties in various aspects such as synchrony, security, or performance. For instance, while some infrastructure channels would be secure (*i.e.*, all information passing through these channels are encrypted), others would just send a stream of unmodified data to the destination. Therefore, it is important that a system architect chooses the most suitable components of the underlying infrastructure to fulfil system's requirements, and then develops applications in accordance with infrastructure's guarantees. If a distributed application requires confidentiality of the data being transferred, then secure channels would be preferred. Otherwise, by using insecure channels, the confidentiality of data would have to explicitly be provided by the application before sending the data.

In this paper the synchrony aspect is addressed, *i.e.*, processes evolve synchronously through rounds of computation delimited by a global clock, or such timely guarantees are relaxed or even non-existent. In an asynchronous system, no assumption about process execution speed and/or message delivery delays is made. Conversely, in a synchronous system, relative processing speed and the message delays are bounded [Schneider 1993]. Therefore, different assumptions about process execution speeds and message delivery delays would require specific design decisions.

Assuming that the underlying infrastructures behaves asynchronously showed to be realistic to a wide range of applications. Furthermore, it is the weakest model in terms of synchrony. That means an algorithm that works in an asynchronous model also works in other models with stronger synchrony assumptions. The opposite is not valid, *i.e.*, an algorithm that works in a synchronous model is prone to incorrect behaviour if timing constraints are violated. Although it is very attractive, in the asynchronous model it is impossible to distinguish a crashed process from an arbitrarily slow process [Chandra and Toueg 1996]. As a consequence, many key problems of fault-tolerant computing are not solvable under the asynchronous assumption. For example, the FLP impossibility result proved that consensus cannot be solved deterministically in asynchronous systems where, at least, one process may crash [Fischer et al. 1985].

By asserting that a system is synchronous, system developers can rely on the timely behaviour of the components. This, in turn, enables one to employ simpler algorithms than those required to solve the same problem in an asynchronous system [Schneider 1993]. For instance, processes can perfectly distinguish faulty from slow processes. Then, the FLP impossibility result is not applicable to the synchronous model. However, building synchronous systems requires infrastructures composed exclusively by timely components, which could be very expensive or even infeasible.

Hybrid models assume intermediate levels of synchrony. Cristian and Fetzer proposed the timed-asynchronous model [Cristian and Fetzer 1999], where the system alternates between synchronous and asynchronous behaviour. In that model the degree of synchrony varies over time. In [Veríssimo 2006], Veríssimo presented the wormhole model, that also exploits the space dimension to provide hybrid synchrony. This means

that timely guarantees of system components may be different (*i.e.*, some components are more predictable than others). For instance, one part of a system would behave synchronously, while other part would be fully asynchronous. Due to the hybrid behaviour, both models are stronger than asynchronous model and weaker than synchronous one.

Hybrid models become a good alternative to the development of distributed systems with timeliness requirements. By enforcing small parts of the system to behave synchronously while other parts are asynchronous, stronger properties provided by synchronous parts can be enjoyed by the system as a whole. For this reason, hybrid systems overcome limitations of the homogeneous systems (fully synchronous/asynchronous).

In [Hasan et al. 2013], we introduced virtualized infrastructure based on the hybrid synchronous model. The Hybrid Synchrony Virtual Networks (HSVN) combines the advantages of hybrid synchronous model with the advantages of virtualization. Sharing of physical resources among VNs becomes an attractive solution to reduce costs with infrastructure, especially the synchronous resources which are more expensive than the asynchronous ones. Therefore, even a small set of synchronous components could be enough to support synchrony requirements from different VNs.

In [Hasan et al. 2013], we illustrated a perfect failure detector $\mathcal{P}$ being implemented in a HSVN. An interesting aspect of implementing $\mathcal{P}$ in a hybrid synchronous system is that, application workload is totally independent of the failure detector modules. Thus, application processes can communicate through asynchronous channels and still enjoy stronger properties provided by the failure detector service. Failure detectors have attracted interest in the development of reliable distributed systems, since consensus and related problems (*e.g.* atomic broadcast [Chandra and Toueg 1996]) can also be solved with it. Further, the failure detection approach can be adapted to solve other relevant problems, such as predicate detection [Gartner and Kloppenburg 2000] and election [Matsui et al. 2000].

Notice that other classes of applications would also benefit from HSVN. In [Mostéfaoui et al. 2006] Mostefaoui *et al.* present an eventual leader election protocol. The proposed protocol benefits from the best of both worlds (synchronous and asynchronous), such that it converges as soon as some synchrony assumption is satisfied. In fact, general round-based protocols for synchronous systems would be adapted to execute in hybrid synchronous models. The design of such hybrid protocols would require a suitable programming model. In this sense, Gorender *et al.* [Gorender et al. 2007] proposed an adaptive programming model for distributed computing, which provides upper-layer applications with process state information according to the current system synchrony. The underlying system model is hybrid, composed by a synchronous part and an asynchronous part.

Even though synchronous components can be shared by multiple HSVNs, the synchronous components are considerably more expensive than the asynchronous ones. Thus, an efficient mechanism for HSVNs mapping that prevents allocation of synchronous nodes unnecessarily is needed.

3. A New Embedding Model for HSVN

In this section we propose an allocation model for HSVN in the shape of a mixed integer program (MIP) [Sierksma 2002]. Differently from our previous work [Hasan et al. 2013],

this model decides the physical resources that need to be synchronous, in an economic manner, and maps the VNs nodes and links based on it. The model constraints (eq. 2-9) are the same in [Hasan et al. 2013] because they impose soundness conditions to the HSVN model that cannot be dropped, while the objective function and variables are different. For instance, certain parameters in the previous paper became open variables in the current work.

Variables definition- The substrate network is represented by an undirected graph $G(N, L)$, composed of a set of physical nodes N connected through a set of physical links L. Each virtual network VN^k belonging to the set of virtual networks VN will be presented by an undirected graph $G^k(N^k, L^k)$, where N^k is given by $N^k = N^k_s \cup N^k_a$, where N^k_s and N^k_a contain the synchronous and asynchronous VN nodes respectively. Similarly, $L^k = L^k_s \cup L^k_a$. A binary function $sync(i^k)$ expresses the VN nodes synchrony demand: $sync(i^k) = 1$ if $i^k \in N^k_s$, otherwise $sync(i^k) = 0$ (***i.e.***, $i^k \in N^k_a$). Similarly, $sync(i^k, j^k)$ expresses the VN links synchrony demand. Besides synchrony, two other attributes are considered for the SN and VNs 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 output variables for our mathematical model, they are four binary functions: 1) $sync(i)$ indicates the synchrony status of the SN node, 2) $sync(i, j)$ indicates the synchrony of the SN link, 3) $\sigma(i^k, i)$ expresses whether node $i \in N$ maps node $i^k \in N^k$, and 4) $\rho(i^k, j^k, i, j)$ expresses whether the physical link $(i, j) \in L$ is part of the path that maps the virtual link $(i^k, j^k) \in L^k$.

The mathematical model- The objective function (O.F.) aims at minimizing the synchronous resources (nodes and links) on the SN, beside minimizing the BW consumption.

Objective: minimize

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

Subject to

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

$$\sum_{\forall VN^k \in VN} \sum_{\forall (i^k, j^k) \in L^k} \rho(i^k, j^k, i, j) \cdot bw(i^k, j^k) \leq bw(i, j) \tag{2}$$

for every $i \in N$

$$\sum_{\forall VN^k \in VN} \sum_{\forall i^k \in N^k} \sigma(i^k, i) \cdot cpu(i^k) \leq cpu(i) \tag{3}$$

- *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 $VN^k \in VN$, $i^k \in N^k$, $i \in N$

$$sync(i^k) \cdot \sigma(i^k, i) \leq sync(i) \tag{7}$$

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

$$sync(i^k, j^k) \cdot \rho(i^k, j^k, i, j) \leq sync(i, j) \tag{8}$$

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

$$sync(i^k, j^k) \cdot \rho(i^k, j^k, i, j) \leq sync(i) \cdot sync(j) \tag{9}$$

The capacity constraint (2) assures that the total bandwidth of the virtual links, mapped on paths that include a certain physical link, does not exceed the bandwidth capacity of this physical link. Similarly, constraint (3) represents the equivalent restriction regarding nodes CPU. The node mapping constraint (4) assures that each virtual node is mapped on a physical one. 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, since the unavailability of a SN node will impact, at most, one node on a given VN. For any virtual link (a, b), the links mapping constraint (6), adopted by [Bays et al. 2012] and [Infuhr and Raidl 2011], assures the creation of a valid physical path. Because the right side of the equation will be 1 and -1 for a and b respectively, so, a will have an outgoing arc and b an ingoing one. For all other nodes on the SN, the right side of the equation will be zero, thus the concatenation of arcs will form a valid path. The nodes synchrony constraint (7) assures that synchronous virtual nodes are mapped only on synchronous SN nodes, whereas asynchronous virtual nodes are allowed to be mapped on synchronous or asynchronous SN nodes. This is acceptable because, the synchronous SN nodes supply what the asynchronous ones do, but the reverse is not valid. Similarly, the links synchrony constraint is presented in Equation 8. Finally, Equation 9 guarantees that the intermediate physical nodes on the synchronous physical path should be also synchronous. This is because these nodes play role in the routing process, thus impacting the source-destination delay.

After solving the mathematical model, each node and link on the SN is defined to be synchronous or asynchronous, each virtual node is mapped to one physical node, and each virtual link is mapped to one physical path at maximum, where a physical path can be a unique physical link or a concatenation of physical links.

4. Performance Evaluation

This section evaluates the performance of our model for allocation of HSVN. The aspects considered during the analysis of our model are: i) economy in the configuration of the SN synchronous resources; ii) privilege of physical resources chosen to be synchronous, and iii) the topology of the subnetwork composed of synchronous resources on the SN.

Like in related works (e.g., [Yu et al. 2008, Bays et al. 2012]), physical and virtual networks were randomly generated. For this we used BRITE [Medina] tool with Waxman [Waxman 1988] model. We implemented the mathematical model with ZIMPL language [Koch 2004] and used CPLEX [IBM] to solve the MIP, running on a computer with 6 cores Intel Xeon processor, 2x2.66 GHz, 32GB of RAM memory, and MAC operating system. Most of the experiments took a considerable time to reach the optimal solution, the reason is that the MIP we propose has four output variables, which leads to considerable set of variables based on the problem size under analysis. Besides, two of those variables (the mapping variables) are based on the value of the other two variables (the synchrony variables) found by the solver, this makes the optimization process long and consumes exponentially the machine memory. For the VNs with small size, the solver reached optimal solution, whereas for bigger size (e.g. group C), the solver took too long (in the order of 24h), or the process was stopped due to fully memory allocation. We decided to stop the solver after finding a solution, even if the solution is not optimal. This might match realistic scenarios, in which the client might prefer a semi-optimal solution in an acceptable computational time, rather than an optimal solution in too long computational time. Thus, during the discussion of results the reader should consider that an optimal solution would perform even better in terms of synchronous resource sharing.

In all the following experiments, the SN size was fixed in 25 nodes. Initially all CPUs of SN nodes are free, and links BW is uniformly distributed between 1-3 Gbps. We start reporting twelve experiments divided into three groups, A, B and C, with VNs total size of 10, 20, and 30 nodes respectively. We refer to these scenarios together as set X. 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 and 1 Gbps. In scenarios 1, 2, 3 and 4 of each group, the VNs synchrony demand varies between 0%, 30%, 60%, and 100%. The parameters for each experiment in the set X are described in Table 1. These parameters match the ones in our previous work [Hasan et al. 2013], since one of the goals in this paper is to compare our current to the previous mapping model. We will summarize our previous work before the comparison.

Table 1. Experiments parameters in set X

Expe.	A1	A2	A3	A4	B1	B2	B3	B4	C1	C2	C3	C4
VN size	10 nodes				20 nodes				30 nodes			
VNs sync.	0%.	30%	60%	100%	0%	30%	60%	100%	0%	30%	60%	100%
SN size	25 nodes											
SN BW	uniformly distributed: 1 Gbps-3 Gbps											
SN CPU	nodes fully free initially											
each VN size	3,4,5 nodes											
VNs BW	uniformly distributed: 100Mbps-1Gbps											
VNs CPU	10,15,25 % of SN nodes CPU											

In brief, in our previous work [Hasan et al. 2013], we argued that for mapping virtual networks with hybrid synchrony demands, it is a waste of resources to use a fully synchronous SN, like the SN used in [Zhang et al. 2010, Infuhr and Raidl 2011]. Rather,

using a hybrid synchronous SN, together with an economic mapping process, reduces the cost. The mapping model proposed was aware of sparing the physical synchronous resources, and using them only when needed. In [Hasan et al. 2013], we used the same SN of 25 nodes, with the same attributes illustrated in Table 1. Respectively, 33% and 34% of the physical nodes and links were fixed to be synchronous on the SN, then we started mapping VNs with increasing size on the configured SN (scenarios in set X). We showed that this approach let the SN, described above, adequate to handle fully synchronous VNs, with total size up to 30 nodes (*scenario C4*).

In the current paper, we have a new approach for mapping HSVNs, with the goal of minimizing the mapping cost even more. We do not reserve, in advance, the synchronous resources on the SN. Rather, the SN synchrony is defined as an output of the MIP. Thus, the physical resources chosen to be synchronous are subject to the VNs demands, and they change if the VNs demands change (*i.e.,* physical synchronous resources are different in each scenario in set X). The model proposed minimizes the number of physical synchronous resources to the limit enough for a given set of VNs. Figure 1 compares SN synchronous resources allocation in our previous and current work, both for scenarios in set X. We notice a clear reduction in the amount of SN synchronous resources needed to map the virtual components. For example, in scenario A4, 16% of synchronous nodes and 6% of synchronous links were enough to map the demand. In comparison with our previous work, this means economizing 52% and 83% of SN synchronous nodes and links, respectively. Obviously, higher gains are observed in scenarios A1, B1, and C1, where no synchronous resources are required. Since in the previous approach some SN components must be synchronous independently of VN demands, the improvement on synchronous resources reservation is of 100%. The reduction rate in the SN synchronous resources for the complete set of scenarios can be found in Table 2.

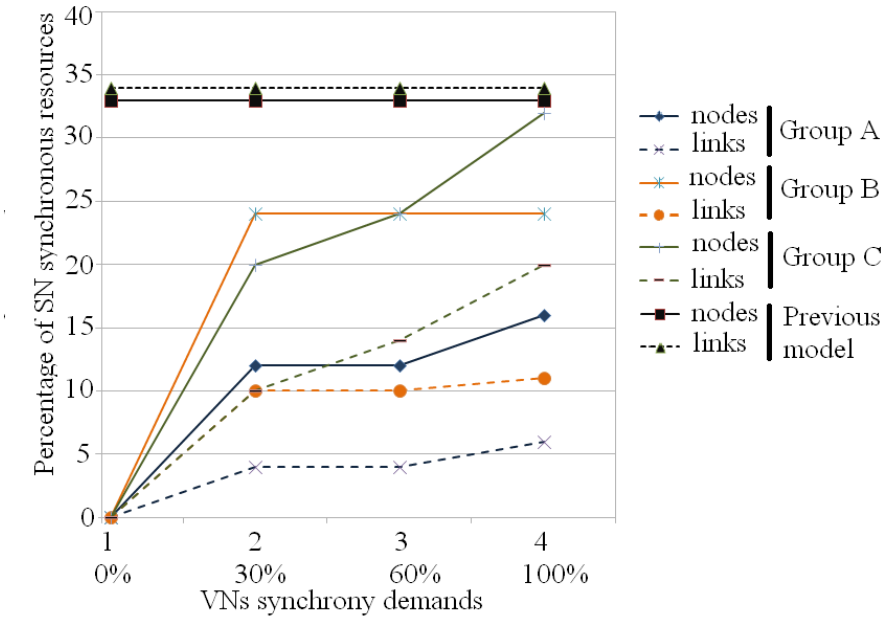


Figure 1. Percentage of sync. resources in SN for scenarios in the set X

Table 2. Economy in SN sync resources between the current model and our previous one [Hasan et al. 2013] - performed for scenarios set X

SN sync resources	economy in SN sync nodes (%) : economy in SN sync links (%)			
Scenario	1	2	3	4
Group A	100 : 100	64 : 88	64 : 88	52 : 83
Group B	100 : 100	27 : 71	27 : 71	27 : 68
Group C	100 : 100	39 : 71	27 : 59	3 : 41

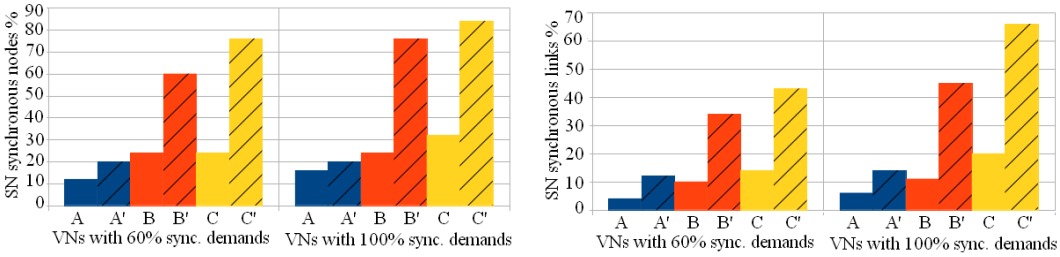
In Figure 1, we note some observations: *i)* for each scenario, synchrony demands on links and nodes grow compatibly (see solid and hashed lines for nodes and links in the graph). *ii)* in scenario C4, the synchronous nodes defined with our new model reached 32%, very near to the fixed portion in the previous work (33% synchronous nodes), which attests the correctness of the previous work, where the fixed synchronous resources could map the VNs in scenario C4. *iii)* in some scenarios, even when VNs synchrony demands increases, the number of SN synchronous resources does not necessarily increases (*e.g.* scenarios B2, B3, and B4). This can be explained by resources sharing provided by VNs mapping. Several virtual components can be mapped on the same physical resources, as long as this does not violate the constraints of CPU and BW capacities for nodes and links, constraints 2 and 3, respectively.

One important feature of the mapping model proposed in this paper is that it lets observe the physical resources chosen to be synchronous and the topology of the subnetworks composed of physical synchronous resources. Next, we discuss these aspects. To perform this kind of evaluation accordingly, we increased the problem size under analysis. We increase: *a)* the size of each virtual network, *b)* the CPU demands, and *c)* the total size of VNs (adding Group D'). The new set of scenarios are named X' (with prim), and their parameters are shown in Table 3. Figures 2(a) and 2(b) depict a comparison in the SN synchronous resources needed in the counterparts scenarios in the X and X' sets, at similar VNs synchrony demands, *i.e.* 60% and 100%. The figures illustrate that, the increment in the problem size has pushed the model to define larger set of synchronous physical nodes and links.

Table 3. Experiments parameters in set X'

Expe.	**A'1**	**A'2**	**A'3**	**A'4**	**A'5**	**A'6**	**B'1**	**B'2**	**B'3**	**B'4**	**B'5**	**B'6**
VN size	10 nodes						20 nodes					
Expe.	**C'1**	**C'2**	**C'3**	**C'4**	**C'5**	**C'6**	**D'1**	**D'2**	**D'3**	**D'4**	**D'5**	**D'6**
VN size	30 nodes						40 nodes					
VNs sync.	0%.	20%	40%	60%	80%	100%	0%.	20%	40%	60%	80%	100%
SN size	25 nodes											
SN BW	uniformly distributed: 1 Gbps-3 Gbps											
SN CPU	nodes fully free initially											
each VN size	fixed to 5 nodes											
VNs BW	uniformly distributed: 100Mbps-1Gbps											
VNs CPU	10,20,30,40,50% of SN nodes CPU											

The SN described in Table 3 has nodes that vary in their connectivity degree between [2-8]. See Figure 3, its vertical axis indicates the connectivity degree, and its horizontal axis starts with the SN (as base of comparison, as we will see), and continues with each scenario performed in set X'. On the leftmost side of Figure 3, we can see the number of nodes available on the SN at each connectivity degree in the range [2-8]. For example, on the figure we can read that the SN has 1 node with connectivity 8, 1 node with connectivity 7, 3 nodes with connectivity 6, and so on. The rest of the figure illus-



(a) SN sync nodes %

(b) SN sync links %

Figure 2. Percentage of SN sync. resources: Comparing groups X and X'

trates the total number of physical synchronous nodes for each scenario in set X', at each connectivity degree [2-8]. For example, in scenario B6, we note that the model has allocated 19 synchronous nodes, in a way that 1 was with connectivity 8, 1 with connectivity 7, 3 with connectivity 6, and so on. In general, we notice that the model tends toward choosing the physical nodes with high connectivity degree to be synchronous. For example, in scenario C5, the model chose all the physical nodes with connectivity [6-8] on SN to be synchronous (compare the numbers at scenario C5 with the base numbers at SN). A possible interpretation for this behaviour is that, nodes with high connectivity degree allow multi-use of the same node, since on one hand, it is connected to a high number of neighbour nodes which fulfils topology constraints, and on the other hand, nodes with high connectivity have high bandwidth sum (*i.e.* the sum of BW capacity of all the physical links connected to it) which fulfils BW constraint. Once the model aims at minimizing the number of the synchronous resources, then such nodes choice is chased.

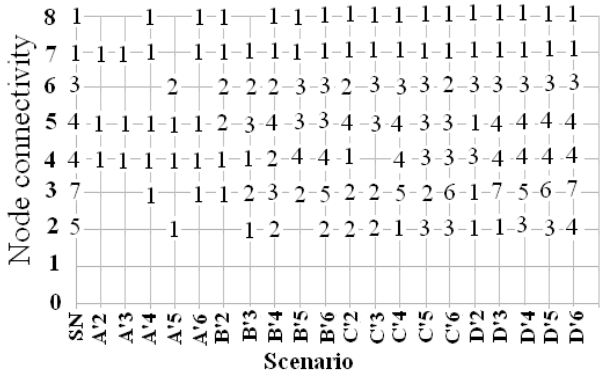


Figure 3. Frequency of synchronous nodes vs. nodes connectivity

Next, we evaluate the topology of synchronous resources in each scenario. We noticed that the topology of the synchronous subnetwork starts by a ring topology for small problem size, and tends towards becoming mesh topology with the increment in the problem size. This observation confirms the previous one regarding the synchronous nodes chosen, being the ones with high connectivity degree. Figure 4 depicts the topology of the synchronous resources in scenario A'3, A'4, A'6, and B'4 as an example. Synchronous nodes are plotted as red circles, and synchronous links as hashed lines. We notice that the synchronous resources gather on one subnetwork, no synchronous islands are observed. Such a gathering allows multi-use of same synchronous nodes and links to answer given

VNs, which fits with the model goal in minimizing the number of synchronous resources.

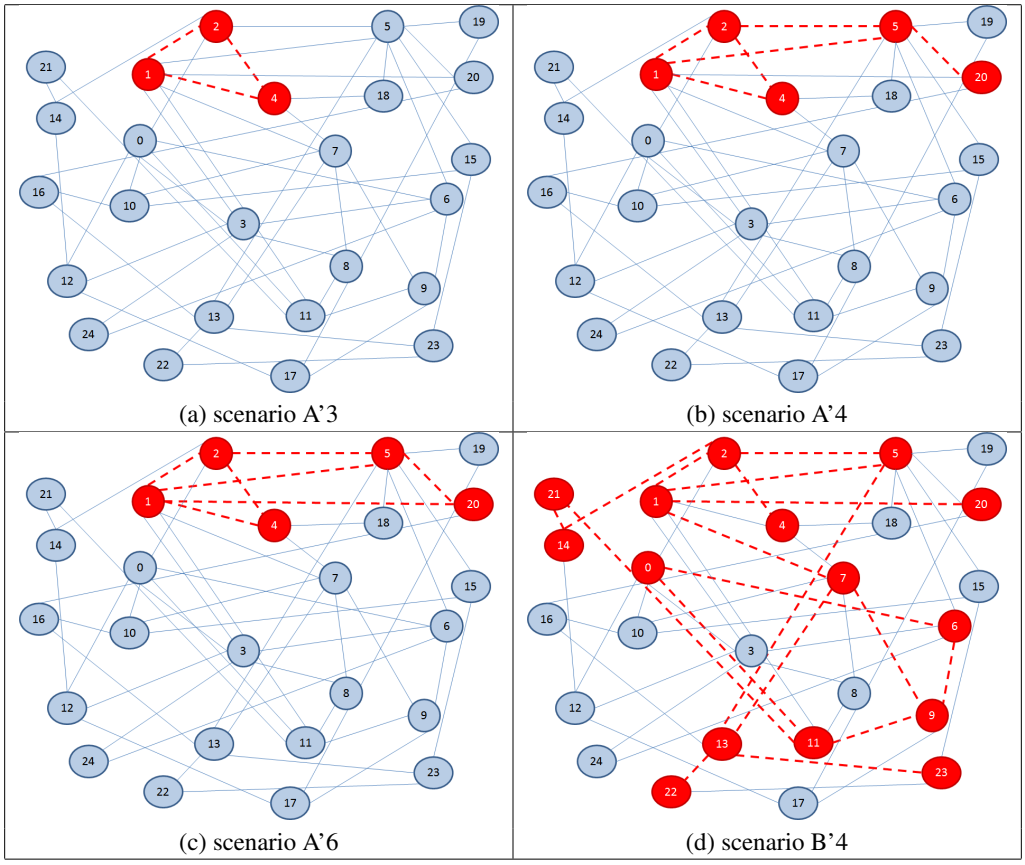


Figure 4. Topology divergence of synchronous resources

5. Related Work

Revising the literature, we found several works treating the VNs mapping problem through two main approaches: optimization models and heuristics. Chawdhury *et al.* propose a MIP model, where the objective function is a weighted sum of node and link mapping, with the goal of increasing the acceptance ratio and decrease cost. Bay *et al.* [Bays et al. 2012] propose a security-aware mapping model where three levels of security are discussed. In [Yu et al. 2011], authors propose an algorithm that combines VN mapping with substrate link backup to improve VNs resilience. Unlike the previous works, Hsu *et al.* [Hsu et al. 2012] map virtual links through path splitting technique, in addition, path migration is used to maximize the number of coexisting VNs. Botero *et al.* [Botero et al. 2013] were the first to propose a heuristic algorithm that considers the CPU of the physical paths intermediate nodes. For wider collection of VNs mapping, see [Belbekkouche et al. 2012] for survey.

In the tropic of VNs mapping, we find that our work is nearer to those who were concerned with delay constraints. For example, Zhang et al. [Zhang et al. 2010] propose a heuristic algorithm for mapping virtual multicast service-oriented networks subject to delay and delay variation. They consider SNs composed of links with maximum delay.

Their work benefits real-time and interactive applications, where packets are supposed to be received at the destination within specific time bounds, and the delay difference of packets reception at multiple destinations should be minimal.

Another work [Infuhr and Raidl 2011] addressed the VNs mapping problem with delay constraints besides routing and location constraints. The SN considered is composed of links with maximum delay, and nodes that have maximum routing capacity and location constraints. Four different categories were used to represent cases in which VNs have different sets of requirements regarding BW, delay, and nodes CPU: *(1) web slice* for low BW requirements, short delays, and no specific CPU requirements, *(2) stream slice* for medium to high BW requirements, no delay bounds, and 3 processing units per routed bandwidth, *(3) P2P slice* for medium BW and CPU requirements and no delay bounds, and *(4) VoIP slice* for medium BW and delay requirements, and high CPU requirements.

In our previous work [Hasan et al. 2013], we proposed the abstraction of Hybrid Synchrony Virtual Networks (HSVN), *i.e.,* virtual networks that have a subset of nodes and links that obey time bounds for processing and communication. We argued that important classes of distributed systems may benefit from hybrid synchrony, and presented an example of perfect failure detector implemented on top of a HSVN. The embedding of several virtual networks in a substrate network allows resource sharing, which is important since synchronous resources are expensive. Economizing the cost, which we aimed at, was achieved through: 1) the usage of a hybrid synchronous SN instead of a fully synchronous one, and 2) sparing the use of synchronous resources, in other words, mapping asynchronous virtual demands on top of synchronous physical resources is considered the last resource invested only before rejecting the demand. Comparing our previous work with others, we achieved less mapping cost for VNs type that has different synchrony demands. For example, some VNs slices adopted in [Infuhr and Raidl 2011] had no delay requirements, yet the SN considered had no distinction in kind of resources, which results in an unneeded cost.

In the current paper, we achieved a better cost economy for the HSVN mapping, by proposing a model that is able to define the physical synchronous resources needed to map given HSVNs. This eliminates the existence of any unused synchronous resources on the SN. Moreover, we issued a study on the SN that maps the HSVNs, in the sense of resources type chosen to be synchronous, and the topology divergence of the physical subnetwork composed of synchronous resources, when the HSVN size increases. To the best of the authors knowledge, this type of study has not yet been presented in the literature.

6. Conclusion

This paper revisits our former work [Hasan et al. 2013] addressing Hybrid Synchrony Virtual Networks (HSVN), (*i.e.,* virtual networks with subsets of nodes and links that require time bounds for processing and communication). Since the cost of synchronous resources is higher than asynchronous ones, in [Hasan et al. 2013] we proposed a mapping model that aims at carefully using the synchronous resources of a SN that supports hybrid synchrony. In this paper we proposed an enhanced version of the HSVN mapping model that allows configuring the substrate network synchrony in an economic manner.

By considering the HSVN synchrony requirements, the new mapping approach al-

lows to set certain nodes and links on a given SN to be synchronous, aware of minimizing the amount of synchronous physical resources to map the demands. A performance evaluation of our approach indicates a substantial economy of synchronous resources compared to previous results [Hasan et al. 2013]. In addition, by exploring a set of scenarios with different synchrony requirements, the analysis conducted in this work gives some important insights about the privilege of physical resources chosen to be synchronous and their topology on the SN. We observed that the model tends towards choosing the nodes with high connectivity degree to be synchronous. A possible interpretation for this is that, such nodes choice allows their multi-use when mapping, since each node is connected to high number of neighbouring nodes and is provided with high bandwidth sum. Such a choice minimizes the number of synchronous resources on the SN, which is the aim of the mapping model. Moreover, we noticed that the synchronous resources configured on the SN tend towards gathering in a mesh topology. Which agrees with the first observation about the nodes type chosen.

Our future work goes in the direction of online mapping for the HSVNs, when the SN resources, or/and the VNs demands are time variant. The mapping approach proposed in this paper, allows only static resource allocation, and thus, does not answer the online mapping. For this reason, we are developing a heuristic algorithm, which is supposed to allow resource allocation for the HSVNs in a dynamic manner.

References

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 *Proc. of CNSM*.

Belbekkouche, A., Hasan, M. M., and Karmouch, A. (2012). Resource discovery and allocation in network virtualization. *IEEE Communication Surveys and Tutorials*, 14(4).

Botero, J. F., Hesselbach, X., Fischer, A., and De Meer, H. (2013). Optimal mapping of virtual networks with hidden hops. *Telecommunication System*, 52(3).

Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2).

Chowdhury, M., Rahman, M. R., and Boutaba, R. (2012). Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Trans. on Networking*, 20(1).

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

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

Gartner, F. C. and Kloppenburg, S. (2000). Consistent detection of global predicates under a weak fault assumption. In *The 19th IEEE Symposium on Reliable Distributed Systems*. IEEE.

Gorender, S., de Araujo Macedo, R. J., and Raynal, M. (2007). An adaptive programming model for fault-tolerant distributed computing. *Dependable and Secure Computing, IEEE Transactions on*, 4(1):18–31.

Hasan, R., Mendizabal, O. M., and Dotti, F. L. (2013). Hybrid synchrony virtual networks: Definition and embedding. In *The Thirteenth International Conference on Networks (ICN)*, pages 104–110.

Hsu, W. H., Shieh, Y. P., Wang, C. H., and Yeh, S. C. (2012). Virtual network mapping through path splitting and migration. In *Proc. of The 26th International Conference on Advanced Information Networking and Applications Workshops*.

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)*.

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

Matsui, H., Inoue, M., Masuzawa, T., and Fujiwara, H. (2000). Fault-tolerant and self-stabilizing protocols using an unreliable failure detector. *IEICE Trans. on Information and Systems*, 83(10).

Medina, A.; Lakhina, A. M. I. B. J. Brite: Boston university representative internet topology generator. http://www.cs.bu.edu/brite.

Mostéfaoui, A., Mourgaya, E., Raynal, M., and Travers, C. (2006). A time-free assumption to implement eventual leadership. *Parallel Processing Letters*, 16(02):189–207.

Schneider, F. B. (1993). Distributed systems (2nd ed.). chapter What good are models and what models are good? ACM Press/Addison-Wesley Publishing Co.

Sierksma, G. (2002). *Linear and Integer Programming: Theory and Practice, Second Edition*. Marcel Dekker publisher.

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

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

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):17–29.

Yu, Y., zhi, C. S., Xin, L., and Yan, W. (2011). Rmap: An algorithm of virtual networks resilience mapping. In *Proc. of the 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*.

Zhang, M., Wu, C., Jiang, M., and Yang, Q. (2010). Mapping multicast service-oriented virtual networks with delay and delay variation constraints. In *IEEE GLOBECOM*. IEEE Communication Society.

Zhu, Y. and Ammar, M. (2006). Algorithms for assigning substrate network resources to virtual network components. In *IEEE INFOCOM*, Barcelona, Spain.