Hybrid Synchrony Virtual Networks: Definition and Embedding

Rasha Hasan, Odorico Machado Mendizabal, Fernando Luís Dotti
Faculdade de Informática
Pontifícia Universidade Católica do Rio Grande do Sul
Porto Alegre, Brazil
Email: rasha.hasan@acad.pucrs.br, odoricomendizabal@furg.br, fernando.dotti@pucrs.br

***Abstract*—Virtual Networks (VNs) have attracted considerable attention in the last years since they offer a flexible and economic approach to deploy customer suited networks and run their applications. Such applications have different requirements, such as topology, security, resilience, and thus pose different challenges to the network embedding problem. In the last three decades of research in distributed systems, one core aspect discussed is the one of synchrony, since it impacts directly the complexity and functionality of fault-tolerant algorithms. In this paper, we argue that VNs and a suitable VN embedding process offer both abstractions and techniques to discuss and address the support of applications with hybrid synchrony demands, thus contributing in core aspects of reliable distributed systems. This work introduces the general idea of Hybrid Synchrony Virtual Networks (HSVNs) and presents a mathematical model that formalises the embedding of HSVNs into a physical network. Our results show that the model proposed is able to, correctly and efficiently, allocate resources on the SN, in an optimal manner.**

Keywords*—*Virtual Network; Distributed Systems; Synchrony

I. Introduction

Virtual Networks (VNs) have attracted considerable attention in the last years, both as an experimental environment to evaluate new protocols, as well as a technology to be integrated in the current network architectures. As can be seen in the literature, the diversity of applications pose different requirements on their supporting VNs, e.g., topology, security, and resilience requirements. In this context, network embedding, a key aspect that defines how resources of a physical network (also called Substrate Network - SN) are used to support VNs, assumes several variants according to the kinds of applications and respective VNs demands.

In the last three decades of research in Distributed Systems (DSs), also triggered by the impossibility result by Fischer, Lynch and Paterson in the 80's [1], one core aspect discussed is the one of synchrony. It is known that the development of DSs depends on the guarantees provided by the underlying infrastructure. If, on the one hand, infrastructures with synchronous guarantees contribute towards development of simpler and reliable systems, on the other hand providing such guarantees may be very expensive or even infeasible. Thus, the assumption of asynchronous environments was commonly adopted because both it is considered more realistic and any solution for the asynchronous case can be generalized to the synchronous case. Since dealing with the uncertainty inherent to asynchronous models requires complex algorithms, and due to evolution in networking technologies, more recently the assumption of partial synchrony has been considered in the literature [2], [3].

In this paper, we propose and argue that VNs and the VN embedding process offer both abstractions and techniques to support applications with Hybrid Synchrony (HS) demands (or partial synchrony). To the authors' best knowledge, this is undiscussed in the VN field and, as mentioned above, of paramount importance to host a prominent class of DSs. More specifically, the contributions of this paper are: (i) we introduce the need and the idea of VNs with hybrid synchrony requirements, characterise the kind of support needed from SNs to cope with these VN requirements, and thus formalise the main abstractions to discuss about hybrid synchrony both at VN and at SN level; (ii) we provide an example of an important distributed application, namely a failure detector, that benefits from hybrid synchronous infrastructures; (iii) we discuss and formalise the network embedding problem for VNs with hybrid synchrony requirements through a mathematical model; (iv) we evaluate the performance of our model in terms of mapping cost, physical resources load, embedding time, and measure the efficiency of our approach to spare synchronous resources.

The paper is organized as follows: Section II discusses related work. In Section III, we motivate the importance of hybrid synchrony to support DSs. In Section IV, we propose and formalize the notion of VNs with hybrid synchrony (HS), which we call HSVNs, together with the embedding model. Section V includes performance evaluation, and Section VI illustrates an example for the HSVN mapping. Finally, in Section VII , we conclude the paper.

II. 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.* [4] propose a relaxed version of mixed integer program, 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.* [5] propose a security-aware mapping model where three levels of security are discussed. Yu *et al.* [6] propose an algorithm that combines VN mapping with substrate link backup to improve VNs resilience. Unlike the previous works, Hsu *et al.* [7] map virtual links through path splitting technique, in addition, path migration is used to maximize the number of coexisting VNs. Botero *et al.* [8] 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.* [9], 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.* [10] 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.

Inführ *et al.* [11] 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.

The study presented in this paper is distinct from the aforementioned works in the following aspects: *(1)* we consider the delay constraints (or time bounds) on both links and nodes, not only links, since in the considered class of DSs some links should have time guarantees in delivering the messages, and some nodes should be performing real-time tasks; *(2)* a physical path is considered synchronous not only when its links are synchronous, rather the path's intermediate nodes should be all synchronous as well, since they play role in the routing process, impacting the source-destination delay; *(3)* the mapping model we propose aims at optimizing the usage of the synchronous resources whose building cost is high comparatively. For example, some VNs slices adopted in [11] had no delay requirements, yet the SN considered had no distinction in kind of resources, which results in an unneeded cost, and *(4)* unlike other works, the SN we consider is hybrid in its components synchrony. Some nodes and links have time bounds and others do not, which is suitable for DSs applications that have hybrid synchronous requirements.

III. Work motivation: Synchrony Models in distributed systems

The design of DSs is strongly dependent on the assumptions about the environment where they execute. For instance, different assumptions about process execution speeds and message delivery delays would require specific design decisions. Thus, an important aspect to consider is the synchrony level offered by the underlying infrastructure. 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 [12].

Assuming that, underlying infrastructures behaving asynchronously showed to be realistic to a wide range of applications. Although they are very attractive, key problems of fault-tolerant computing are not solvable under the asynchronous assumption. For example, Fischer, Lynch and Paterson have proven that consensus cannot be solved deterministically in asynchronous systems where at least one process may crash [1].

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 [12]. For instance, processes can perfectly distinguish faulty from slow processes. 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 [2], where the system alternates between synchronous and asynchronous behaviour. In that model, the degree of synchronism varies over time. In [3], Veríssimo presented the wormhole model, that exploits the space dimension to provide hybrid synchrony. This means that timely guarantees of system components may be different. For instance, one part of a system would behave synchronously, while other part would be fully asynchronous.

Once behaviours caused by faults and arbitrary delays are expected in the conventional infrastructures, hybrid models become a good option to improve the development of fault-tolerant applications. 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.

Example: Failure detector - Failure detectors have attracted interest in the development of reliable DSs, since consensus and related problems (e.g., atomic broadcast [13]) can be solved with it. The failure detection approach can also be adapted to solve other relevant problems, such as predicate detection [14] and election [15].

Failure detectors are used to detect faulty processes in a group of processes, and they are defined in terms of abstract properties, namely *accuracy* and *completeness*. A failure detector that satisfies *strong accuracy* and *strong completeness* properties is a *perfect failure detector* ($\mathcal{P}$) [13]. It means it never makes mistakes (suspect erroneously) and, eventually detects every crash.

A failure detector $\mathcal{P}$ can be built on top of synchronous environments. The problem is that implementing $\mathcal{P}$ in fully synchronous environments depends on the existence of an underlying infrastructure with timely guarantees (sometimes infeasible) while implementing it in asynchronous systems is even impossible.

Macêdo *et al.* [16] propose an implementation of a failure detector $\mathcal{P}$ that runs on hybrid synchronous environments. They assume the underlying system has synchronous processes, some channels behave synchronously and others asynchronously.

Basically, each module fd_i periodically asks to processes p_j if they are alive. Upon receiving a message "are you alive", every correct process replies to the sender with a "I'm alive" message. Upon receiving the replying message, fd_i knows the process p_j is up. However, if a timeout expires, it means that no answer from p_j was received in the last τ time units. If the channel connecting processes fd_i to fd_j is synchronous, then it is known that the process p_j has failed. Process p_j is added to the faulty list in p_i, and a notification informing the detection is sent to all other processes. Otherwise, if the channel is asynchronous, there is no way to detect if the process p_j has failed or the reply message is delayed.

We illustrate a failure detector $\mathcal{P}$ running in a hybrid synchronous environment in Figure 1. It shows a hypothetical topology for an application composed by six processes. All processes are hosted in synchronous nodes, and they communicate with each other through payload channels (pa_i). Further, a failure detector module fd_i is attached to each process P_i. Connection between failure detectors modules in a synchronous partition is done by synchronous channels (solid lines in the figure). Connection between fd modules in different partitions can be asynchronous (dotted lines). In order to improve legibility, payload channels pa_i, CPU and bandwidth constraints were omitted in the figure. In this example, the payload channels should be represented by a complete graph connecting every pair of processes.

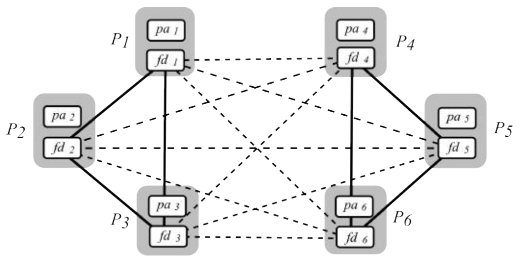


Fig. 1: Application topology with failure detector $\mathcal{P}$

Although not all failure detectors are in the same synchronous partition, the $\mathcal{P}$ implementation allows every application process to benefit from a perfect detection. Even in cases in which not all fd_i modules belong to a synchronous partition, it is possible to take advantage of the existing synchrony, provided that some subgraphs are synchronous. In such cases, assumptions from weaker failure detectors (e.g., $\Diamond\mathcal{P}$, $\Diamond\mathcal{S}$ [13]) would be ensured and still useful for the applications.

Another interesting aspect of the hybrid synchronous system is that application workload is totally independent of the failure detector modules. Application processes can communicate through asynchronous channels and still enjoy stronger properties provided by the failure detector service.

IV. Our proposal: Hybrid Synchrony Virtual Networks

Our proposal lays in offering VNs to support DSs applications with partial synchrony. The rational behind it is mainly twofold. First; the synchronous elements in DSs are considerably more expensive than the asynchronous ones, since they require fundamental handling mechanisms. Resource sharing, provided by the nature of VNs, allows simultaneous use for this class of expensive resources. Secondly, VNs allow flexible resource allocation mechanisms by the Infrastructure Service Provider (ISP). This provides modern DSs applications with scalability which is an important aspect in the field.

The aforementioned reasons lead to the abstraction of new type of VNs: the Hybrid Synchrony Virtual Networks (HSVN). They are virtual networks that have a subset of nodes and links that obey time bounds for processing and communication. This abstraction put us to meet two main aspects associated: (i) the SN design, since VNs inherit properties that only exist in the underlying infrastructure, and (ii) suitable efficient embedding process for the HSVN.

Although HSVN can run on fully synchronous SN, this decision would have to pay the excess in an unneeded cost, since even asynchronous virtual nodes and links will be mapped on synchronous physical ones. We argue that hybrid synchronous SN, combined with a suitable embedding, is capable to answer the timely requirements in an economic manner. Hybrid synchronous SNs have two classes of nodes: (i) *synchronous nodes* with functioning time guarantees, achieved through the implementation of periodical real-time tasks, and (ii) *asynchronous nodes* that have no timely guarantees. Analogously, two classes of physical links are available: (i) *synchronous links* that have time-bounded messages transmission delay, achieved through the implementation of Quality of Service (QoS) policies and admission control, and (ii) *asynchronous links* that have no timely guarantee.

A. The HSVN embedding model

We propose a HSVN embedding mathematical model in the shape of a Mixed Integer Program (MIP). Our model answers hybrid synchrony requirements and, at the same time, optimizes the synchronous resources usage besides the BW and CPU. The HSVN exploits the space dimension to provide hybrid synchrony [3].

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. N is given by $N_s \cup N_a$, where N_s and N_a contain all the synchronous and asynchronous SN nodes, respectively. Similarly, L is given by $L_s \cup L_a$. 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 $N^k = N^k_s \cup N^k_a$ and $L^k = L^k_s \cup L^k_a$. We consider that there is a cost $c(i, j)$ for one unit of traffic going through the physical link $(i, j) \in L$. Analogously, $c(i)$ is the cost for processing one unit of traffic in node $i \in N$. $c(i, j)$ and $c(i)$ are of higher value if the link and node were synchronous. A binary function $sync(i)$ expresses the SN nodes synchrony: $sync(i) = 1$ if $i \in N_s$, otherwise $sync(i) = 0$ (i.e., $i \in N_a$). Similarly, $sync(i, j)$ expresses the SN links synchrony. Functions $sync(i^k)$ and $sync(i^k, j^k)$ indicate the virtual nodes and links synchrony respectively ($i^k \in N^k$ and $(i^k, j^k) \in L^k$). 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 output variables for our mathematical model: 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 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 embedding model- The Objective Function (O.F.) we consider is inspired from [10], which is the total resources used (e.g., BW and CPU). We modify the O.F. with the goal of minimizing the use of synchronous resources besides the BW and CPU. For this purpose, $c(i)$ and $c(i,j)$ are inserted in (1).

Objective: minimize

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

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

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

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, and only once, on a physical node. Without this constraint, and since the O.F. aims at minimizing cost, then the optimizer might choose not to map any node, which is against the goal. 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. This procedure minimizes the number of virtual nodes prone to failure by a physical node failure. This is an important aspect in fault tolerant distributed systems, where a maximum number of faulty processes is accepted to allow the system to tolerate the fault, i.e., not to let the fault impact the system output. For any virtual link (a,b), the links mapping constraint (6), adopted in [5] and [11], 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, meaning 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 (8). Note that the allocation of synchronous physical resources for asynchronous virtual demands is done only if there are no other possible options (physical asynchronous resources got exhausted). This is achieved via minimizing the O.F. Finally, constraint (9) guarantees that when 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 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.

V. Performance evaluation

We evaluate the performance of our model through: 1) mapping cost, 2) physical resources load, 3) optimizing the usage of synchronous resources and, 4) embedding time.

A. Workloads and tools

Like some other works [17] [5], the physical and virtual networks were randomly generated. For this we used BRITE [18] tool (Boston university Representative Internet Topology gEnerator) with Waxman [19] model. We implemented the model with ZIMPL language [20] (Zuse Institute Mathematical Programming Language) and used CPLEX Optimization Studio [21] to solve the MIP, running on a computer with a CPU of 4 cores and 1.60 GHz, and 2 GB of main memory. We ran twelve experiments divided into three groups, A, B and C, with VNs total size of 10, 20, and 30 nodes respectively. Table I describes the parameters for each experiment.

TABLE I: Experimentsparameters

Group:VN size	A:10 routers,B:20 routers,C:30 routers			
Scenario	**1**	**2**	**3**	**4**
SN size	25 nodes			
SN BW	uniformly distributed: 1Gbps-3Gbps			
SN CPU	nodes fully free initially			
VNs BW	uniformly distributed: 100Mbps-1Gbps			
VN CPU	10, 15, 25 % of SN nodes CPU			
SN sync.	30%			100%
VNs sync.	0%.	30%	60%	x%

In all the experiments, the SN size was fixed in 25 nodes. Initially, all CPUs are free, and links BW is uniformly distributed between 1-3 Gbps. In scenarios 1, 2, and 3 of each group, the SN was set up with 30% of synchronous resources, whereas in scenario 4 of each group the SN was fully synchronous. This scenario will be the base for cost comparison since it simulates the case where all the SN nodes and links have time bounds.

The VNs were generated with 3, 4, or 5 nodes each, the virtual nodes have 10%, 15%, or 25% of the SN nodes CPU. The VNs links BW was uniformly distributed between 100 Mbps and 1 Gbps. The VNs synchrony varies within each group: 0% in scenario 1, 30% in scenario 2 and 60% in scenario 3. Note that the VNs synchrony in the fourth scenario of each group was referred to as x% because in this scenario the mapping cost will be independent of the VNs synchrony requests since the SN resources have no differentiation in synchrony (the SN is fully synchronous).

B. Results

The first parameter evaluated is the mapping cost, represented by our model objective function This parameter is a combination of CPU and BW used. Figure 2 depicts the mapping cost for each of the twelve experiments performed. We note down three main observations: *(i)* within each group, the mapping cost increases gradually with the increment of the VNs synchrony requests. For example, the mapping cost increased 173% when the VNs synchrony demands increased from 0% in B1 to 30% in B2, and increased more 76% in B3 with 60% VNs synchrony demands. This is explained by the increase in physical synchronous resources (nodes and links) usage, which are more expensive. *(ii)* by comparing the counterparts experiments of the three groups, e.g., A2, B2, and C2 (all with 30% VNs synchronous demands), we notice that the mapping cost increases. This is due to the incremental VN size, which tends naturally to reserve more physical resources. *(iii)* comparing the three first experiments within each group with the fourth one, we can say that our model can host hybrid VNs in an economic way. That is, the SN can be used in an optimized way to allocate these demands. For instance, experiment C3 depicts the mapping of a hybrid VN with 60% of synchrony demands on a hybrid SN with 30% of synchronous resources. Whereas mapping the same VNs demand on a fully synchronous SN, experiment C4, is subject to an extra 94% un-needed cost. So, hybrid VNs do not need fully synchronous SN, rather a hybrid SN with suitable mapping is enough to allocate the needed demands, and spares resources for future ones.

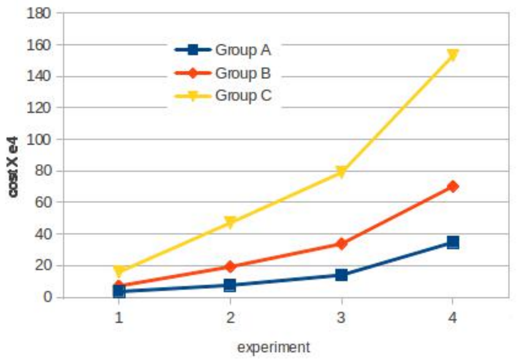


Fig. 2: Mapping cost

The second parameter to evaluate is the SN resources load. We present the load evaluation for scenario C3 only, because, within the twelve experiments performed, it is the one with the maximum VNs size and higher synchrony percentage as hybrid substrate. Figure 3 shows the cumulative distribution function (CDF) of the SN nodes CPU and links BW usage. We note that only 4% of the SN nodes reached 65% of use, and about 3% of the SN links have BW consumption that exceeds 60%. So, the SN resources seem to have fair load, which is an important factor since avoiding to fully charge nodes and links tends increase the possibility of mapping future demands within the same SN.

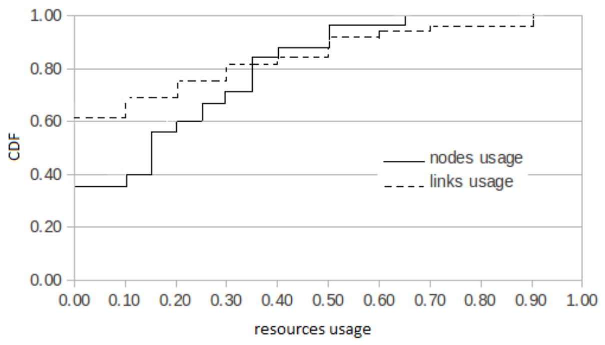


Fig. 3: CDF for resource usage in experiment C3

Next, to check closely the model ability in sparing the synchronous resources, we devised a scenario where all asynchronous SN resources get used. Consequently, synchronous SN resources are used for mapping asynchronous VNs demands. This case is allowed only when SN is running out of asynchronous physical resources. To investigate this point, we modified experiment B2 in the set to experiment B2', where the SN in B2' has 25% asynchronous resources and the VNs CPU demands increased to 60% of the SN nodes CPU. Figure 4 is a scheme of resource mapping in B2'. The horizontal axis is the SN resources, divided into synchronous in the positive portion of the axis, and asynchronous in the negative one. The same for the VNs synchronous and asynchronous demands on the vertical axis. Note that, the altitude holds no information, it is just the positioning (positive or negative). This division results in four quarters, we number them counter clockwise. Optimally, the synchronous demands are to be mapped on synchronous SN resources, and asynchronous on asynchronous. This leads to points allocated only in the first and third quarters. No points are supposed to appear in the second quarter, since it is meaningless to map synchronous virtual demands on asynchronous physical resources. The fourth quarter is supposed to have the minimum number of allocations, which is an indication of optimizing the use of synchronous resources (i.e., few mappings of asynchronous demands on synchronous physical resources). In other works, that consider fully synchronous SN, all the allocations will appear in the right half of the plane (quarter 1 and 4), which is the expensive part. In our work we insert the possibility of allocations existing in the left half of the plane, which reduces the cost potential.

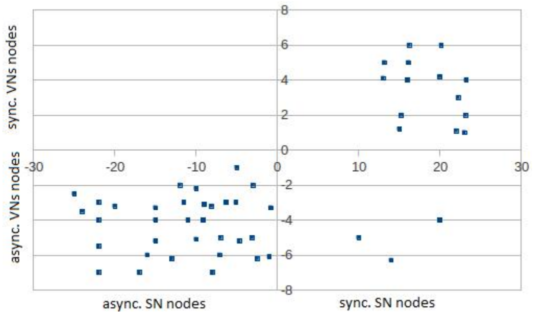


Fig. 4: Scheme of resource mapping in experiment B2'

Finally, We evaluate the optimization time for each experiment performed, see Table II. We notice that most of the values were less than 10 minutes which is a reasonable computational time. For some experiments the value was high, which might be a compromise for obtaining an optimal solution.

TABLE II: Embedding time (in minutes)

Group.	exp.1	exp.2	exp.3
A	0.85	0.40	0.31
B	37.55	1.64	10.26
C	58.34	27.12	4.47

VI. GRAPH-BASED EXAMPLE FOR HSVN MAPPING

In this section, we present a simple graph-based example for mapping HSVNs. The goal is to see, in practice, some aspects considered by our approach.

We consider three VNs with different synchrony demands, together with a hybrid SN. Both the VNs and the SN are shown in Figure 5. VN^1 represents a virtual infrastructure for an application equipped with a failure detector, similar to that presented in Section III, but with four nodes. The links connecting payload channels are omitted in the figure to improve legibility. VN^1 has hybrid requirements regarding synchrony, whereas VN^2 and VN^3 represent fully synchronous and asynchronous applications, respectively. By solving the MIP for the example under analysis, every virtual node was mapped on one physical node, and each virtual link was mapped on one physical path, where a physical path can be one physical link, or a concatenation of several physical links. Figure 5 shows the optimal solution found for nodes and links mapping.

In the light of this example, we point at some aspects previously detailed in the paper: *i)* our mapping approach considers both hybrid VNs (e.g., VN^1) and homogeneous VNs (e.g., VN^2 and VN^3), *ii)* the proposed model aims at sparing the synchronous resources, e.g., an asynchronous virtual link (fd_2^1, fd_4^1), was mapped on a path of three asynchronous links {(b,h),(h,a),(a,c)}, connecting b to c, to avoid mapping it on a synchronous shorter path of one link {(b,c)}, connecting the same two nodes, *iii)* the HSVN allows resource sharing, which is important, especially for sharing the synchronous resources, e.g., the synchronous nodes c, and d could be used both for mapping VN^1 and VN^2. The same with the synchronous link (d, c), finally, *iv)* regarding mapping cost, we ran the optimizer for the adopted example in two cases, first, with a hybrid SN as illustrated in Figure 5, secondly, with a fully synchronous SN. The O.F. values obtained were 5400 and 11400 respectively in both cases. This shows clearly that, the use of hybrid SN, together with a suitable mapping process, minimizes the cost considerably.

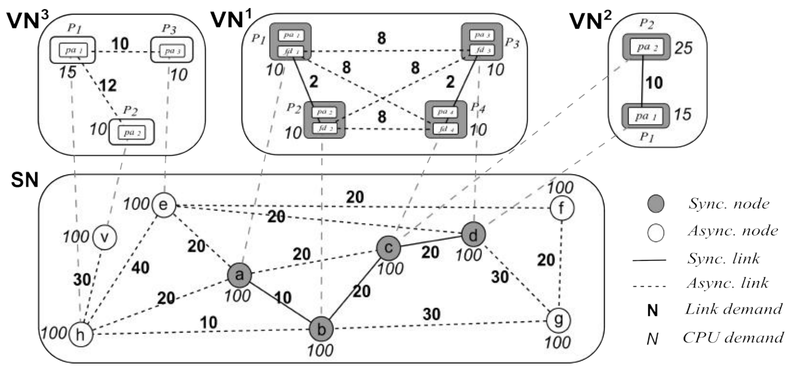
VII. DISCUSSION

In this paper, we have proposed the concept of Hybrid Synchrony Virtual Networks, i.e., virtual networks that have a subset of nodes and links that obey time bounds for processing and communication. The rationale behind it is, at one hand, that there is an important class of systems, namely fault-tolerant distributed systems, that can benefit from the hybrid synchrony. On the other hand, the embedding of several virtual networks in a substrate network allows resource sharing, which is important since synchronous resources are expensive. The proposed embedding mathematical model adopts these aspects.

Our results show that our model can host hybrid synchrony VNs in an economical way, which is achieved mainly through: 1) the usage of a hybrid synchronous SN instead of a fully synchronous one, and 2) synchronous resources are spared, in other words, mapping asynchronous virtual demands on top of synchronous physical resources is considered the last resource invested only before rejecting the demand. Moreover, the model reflects a reasonable load distribution on the underlying nodes and links, and acceptable optimization time.

Our work can be generalized in three directions: *(i)* although we have dedicated enough efforts to illustrate perfect failure detectors, a wider set of applications benefit from hybrid synchrony. For instance, general purpose applications would communicate mainly through asynchronous channels and still rely on timely execution triggers. Thus, certain actions would be executed in a timely fashion (e.g., checkpointing [22], election [15], or any round-based agreement); *(ii)* the hybrid SN we are proposing, combined with our embedding model, can host not only hybrid synchrony applications, but also homogeneous ones (fully synchronous or fully asynchronous); *(iii)* while in this step of our work we are concerned with synchrony, we envisage that similar models may, in the future, be used to denote other kinds of specific functionalities expected from the resources. such as subsets of nodes and links with special security or resilience features.

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



VN^1 ***links***	***SN path***
(fd_1, fd_2)	$\{(a, b)\}$
(fd_1, fd_3)	$\{(a, e), (e, d)\}$
(fd_1, fd_4)	$\{(a, c)\}$
(fd_2, fd_3)	$\{(b, g), (g, d)\}$
(fd_2, fd_4)	$\{(b, h), (h, a), (a, c)\}$
(fd_3, fd_4)	$\{(d, c)\}$

VN^2 ***links***	***SN path***
(pa_1, pa_2)	$\{(d, c)\}$

VN^3 ***links***	***SN path***
(pa_1, pa_2)	$\{(h, v)\}$
(pa_1, pa_3)	$\{(h, e)\}$

Fig. 5: Virtual networks mapping

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.

Acknowledgement

This work has been supported by FAPERGS-NPRV (Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul - Nucleo de Pesquisa em Redes Virtuais), project PRONEM 11/2038-1.

References

[1] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[2] F. Cristian and C. Fetzer, "The timed asynchronous distributed system model," *IEEE Trans. on Parallel and Distributed Systems*, vol. 10, no. 6, pp. 642–657, 1999.

[3] P. E. Veríssimo, "Travelling through wormholes: a new look at distributed systems models," *ACM SIGACT News*, vol. 37, no. 1, pp. 66–81, 2006.

[4] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. on Networking*, vol. 20, no. 1, pp. 206–219, 2012.

[5] L. R. Bays, R. R. Oliveira, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Security-aware optimal resource allocation for virtual network embedding," in *Proc. of CNSM*, 2012.

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

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

[8] J. F. Botero, X. Hesselbach, A. Fischer, and H. De Meer, "Optimal mapping of virtual networks with hidden hops," *Telecommunication System*, vol. 52, no. 3, pp. 1–10, 2013.

[9] A. Belbekkouche, M. M. Hasan, and A. Karmouch, "Resource discovery and allocation in network virtualization," *IEEE Communication Surveys and Tutorials*, vol. 14, no. 4, pp. 1114–1128, 2012.

[10] M. Zhang, C. Wu, M. Jiang, and Q. Yang, "Mapping multicast service-oriented virtual networks with delay and delay variation constraints," in *IEEE GLOBECOM*. IEEE Communication Society, 2010.

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

[12] F. B. Schneider, "Distributed systems (2nd ed.)," S. Mullender, Ed. ACM Press/Addison-Wesley Publishing Co., 1993, ch. What good are models and what models are good?

[13] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *Journal of the ACM*, vol. 43, no. 2, pp. 225–267, 1996.

[14] F. C. Gartner and S. Kloppenburg, "Consistent detection of global predicates under a weak fault assumption," in *The 19th IEEE Symposium on Reliable Distributed Systems*. IEEE, 2000.

[15] H. Matsui, M. Inoue, T. Masuzawa, and H. Fujiwara, "Fault-tolerant and self-stabilizing protocols using an unreliable failure detector," *IEICE Trans. on Information and Systems*, vol. 83, no. 10, pp. 1831–1840, 2000.

[16] R. de Araujo Macedo and S. Gorender, "Perfect failure detection in the partitioned synchronous distributed system model," in *Proc. of International Conf. on Availability, Reliability and Security (ARES)*, 2009.

[17] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM-SIGCOMM*, vol. 38, no. 2, pp. 17–29, 2008.

[18] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: Boston university representative internet topology generator." [Online]. Available: http://www.cs.bu.edu/brite

[19] B. M. Waxman, "Routing of multipoint connections," *Selected Areas in Communications*, vol. 6, no. 9, pp. 1622–1617, 1988.

[20] T. Koch, "Rapid mathematical programming," Ph.D. dissertation, Tichnische Universität Berlin, 2004.

[21] IBM, "Cplex." [Online]. Available: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer

[22] E. N. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson, "A survey of rollback-recovery protocols in message-passing systems," *ACM Computing Surveys (CSUR)*, vol. 34, no. 3, pp. 375–408, 2002.