Designing and Embedding Reliable Virtual Infrastructures

Wai-Leong Yeow
wlyeow@ieee.org

Institute for Infocomm Research
1 Fusionopolis Way, Singapore 138632

Cédric Westphal
cwestphal@docomolabs-usa.com

DOCOMO USA Labs
3240 Hillview Ave, Palo Alto, CA 94304, USA

Ulaş C. Kozat
kozat@docomolabs-usa.com

DOCOMO USA Labs
3240 Hillview Ave, Palo Alto, CA 94304, USA

ABSTRACT

In a virtualized infrastructure where physical resources are shared, a single physical server failure will terminate several virtual servers and crippling the virtual infrastructures which contained those virtual servers. In the worst case, more failures may cascade from overloading the remaining servers. To guarantee some level of reliability, each virtual infrastructure, at instantiation, should be augmented with backup virtual nodes and links that have sufficient capacities. This ensures that, when physical failures occur, sufficient computing resources are available and the virtual network topology is preserved. However, in doing so, the utilization of the physical infrastructure may be greatly reduced. This can be circumvented if backup resources are pooled and shared across multiple virtual infrastructures, and intelligently embedded in the physical infrastructure. These techniques can reduce the physical footprint of virtual backups while guaranteeing reliability.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems—*Reliability, availability and serviceability*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network communications*

General Terms

Algorithms, Reliability

Keywords

Infrastructure Virtualization

1. INTRODUCTION

With infrastructure rapidly becoming virtualized, shared and dynamically changing, it is essential to have strong reliability support from the physical infrastructure, since a single physical server or link failure affects several shared virtualized entities. Providing reliability is often linked with over-provisioning both computational and network capacities, and employing load balancing for additional robustness. Such highly reliable systems are good for applications where large discontinuity may be tolerable, e.g. restart of network flows while re-routing over link or node failures, or partial job restarts at node failures. A higher level of fault tolerance is required at applications where some failures have a substantial impact on the current *state* of the system. For instance, virtual networks with servers which perform admission control, scheduling, load balancing, bandwidth broking, AAA or other NOC operations that maintain snapshots of the network state, cannot tolerate total failures. In master-slave/ worker architectures, e.g. MapReduce, failures at the master nodes waste resources at the slaves/workers.

Through synchronization [4, 10] and migration techniques [9, 22] on virtual machines and routers, we postulate that *fault tolerance can be introduced at the virtualization layer*. This has several benefits. Different levels of reliability can be customized and provisioned over the same physical infrastructure. There is no need for specialized, fault tolerant servers. Instead, redundant (backup) virtual servers can be created dynamically, and resources are pooled together, increasing the primary capacity. Both will lead to a better overall utilization of the physical infrastructure.

In this paper, we propose an Opportunistic Redundancy Pooling (ORP) mechanism to leverage the properties of the virtualized infrastructure and achieve a $n : k$ redundancy architecture, where k redundant resources can be backups for *any* of the n primary resources, and *share* the backups across multiple virtual infrastructures (VInfs).

For a quick motivating example, consider two VInfs with n_1 and n_2 computing nodes. They would require k_1 and k_2 redundancy to be guaranteed reliability of r_1 and r_2, respectively. Sharing the backups with ORP will achieve a redundancy of $k' = \max(k_1, k_2)$ with the same levels of reliability, reducing the resources that are provisioned for fault tolerance by at most 50%.

In addition, there is joint node and link redundancy such that a redundant node can take over a failed node with guaranteed connectivity and bandwidth. ORP ensures VInfs do not connect to more redundant nodes than necessary in order to keep the number of redundant links low.

The other contribution of this paper is a method to statically allocate physical resources (compute capacity and bandwidth) to the primary and redundant VInfs simultaneously, taking into account the output of the ORP mechanism. It attempts to reduce resources allocated for redundancy by utilizing existing redundant nodes, and overlapping band-

widths of the redundant virtual links as much as possible.

Our paper focuses on the problem of resource allocation for virtual infrastructure embedding with reliability guarantee. Practical issues such as system health monitoring, protocol design, recovery procedures, and timing issues are out of the scope of this paper.

The organization of this paper is as follows. In the next section, we briefly describe the background, notations and define reliability in Section 2. Then, we describe a virtual architecture that can provide fault tolerance and estimate the benefits of sharing redundancies in Section 3. We see how the link topology is preserved under failures in Section 4, and how resources can be efficiently allocated in the physical infrastructure in Section 5. Finally, we evaluate and validate the ideas through simulation in Section 6, present related work in Section 7. Section 8 concludes this paper.

2. PROBLEM STATEMENT

We consider a resource allocation problem in a virtualized infrastructure, such as a data center, where the virtualized resources can be leased with reliability guarantees. The physical infrastructure is modeled as an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of physical nodes and $\mathcal{E}$ is the set of physical links. Each node $\mu \in \mathcal{N}$ has an available computational capacity of Γ_μ. Each undirected link $(\mu, \nu) \in \mathcal{E}, \mu, \nu \in \mathcal{N}$ has an available bandwidth of $\Lambda_{\mu\nu}$.

Each resource lease request is modeled as an undirected graph $G = (N, E)$. N is a set of compute nodes and E is a set of edges. We call this a virtual infrastructure (VInf). γ_u is the computation capacity requirement for each node $u \in N$, and bandwidth requirements between nodes are λ_{uv}, $(u, v) \in E$ and $u, v \in N$.

Reliability is guaranteed on the set of critical nodes $C \subseteq N$ of a VInf G through redundant virtual nodes in the physical infrastructure $\mathcal{G}$. A backup (redundant) node b must be able to assume full execution of a failed critical node c. Hence, the backup node must have sufficient resources in terms of computation $\gamma_b \geq \gamma_c$ and bandwidth to neighbors of c: $\lambda_{bu} \geq \lambda_{cu}, \forall u \in N, (c, u) \in E$.

The problem is to allocate least resources for a VInf G on a physical infrastructure $\mathcal{G}$, including redundancy such that a reliability guarantee of at least r is achieved.

2.1 Reliability

We define reliability as the probability that critical nodes of a VInf remain in operation, over all possible node failures. This is not to be confused with availability, which is defined as a ratio of uptime to the sum of uptime and downtime [20]. By guaranteeing a reliability of r, we are ensuring that there are sufficient redundant physical resources available in times of failure, with probability r. For a VInf with n critical nodes and k backup nodes, we want to ensure that the probability of a k-node failure out of the $n+k$ virtual nodes is less than $1 - r$. This covers cases where some critical and backup nodes fail simultaneously.

2.2 How many backups?

The number of redundant nodes depend on the physical mapping, and the failure models of both the physical nodes and the virtual infrastructure. We impose two physical mapping constraints: (i) each virtual node is only mapped to one physical node, and (ii) the mapped physical nodes are placed apart to avoid correlated failures among the physical machines, e.g. on different racks with different power supplies. This way, the failure rate of a virtual node u is directly derived from the physical node μ it is mapped on. Guaranteeing reliability can then be focused on the failure model of the virtual infrastructure. In general, the reliability of the overall virtual infrastructure (including k redundant nodes) can then be computed as

$$r(k) = \sum_{y=0}^{k} \sum_{x=0}^{\min(n,y)} \binom{k}{y-x} p^{y-x} (1-p)^{k-(y-x)} f(x) \quad (1)$$

for some failure probability distribution $f(x)$ in which x is the number of critical nodes that failed. The binomial term is due to independent failures of the redundant nodes, and the assumption that physical nodes hosting them are homogeneous with a failure rate p.

Virtual node failures due to any cascading effect (e.g. load-based, tree-based, or degree-based) can be fully captured by the $f(x)$ term. Computing $f(x)$ for these models is detailed more in the full version of this paper [24]. Once $f(x)$ is obtained, a numerical method for searching k can then be used to ensure guarantee a certain level of reliability r.

For ease of exposition, we focus in this paper on independent node failures. Then $r(k)$ reduces to

$$r(k) = \sum_{x=0}^{k} \binom{n+k}{x} p^{x} (1-p)^{n+k-x} \quad (2)$$

Thus the integer ceiling of its inverse corresponds to the minimum number of redundant nodes k.

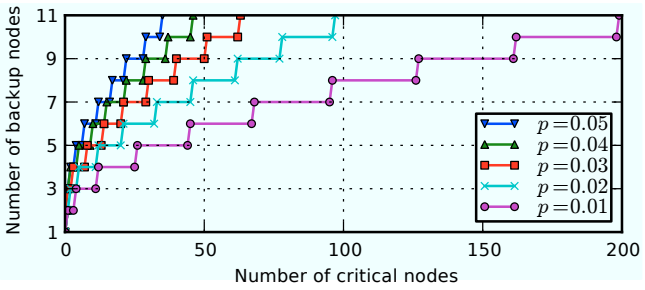


Figure 1: Minimum number of backups needed for a reliability guarantee of 99.999%.

3. REDUNDANCY POOLING

Since redundant nodes are idle, those that are provisioned for one VInf can be reused with another VInf, provided the reliability guarantees are still met. To simplify discussion, we now assume the failure rates of all critical and physical nodes are independent and uniform. Fig. 1 shows the number of backup nodes required as the number of critical nodes increase for a reliability guarantee of 99.999% over various failure probabilities $p = 0.01, \ldots, 0.05$. The range of physical failure values were chosen due to a recent Intel study from different locations and different types of cooling [2].

The curves' sub-linear property can be used to reduce redundancy: if two or more VInfs pool their backup nodes, the total number of backup nodes required is reduced. For example, for the case of $p = 0.01$, two VInfs of 100 critical nodes each will require a total of 16 backup nodes ($k = 8$ each). This number can be reduced to 11 when both VInfs

(with combined $n = 200$) pool their backup nodes together, saving redundant resources by 31.25%.

However, it can be shown [24] that for large n, k is asymptotically linear, so combining large VInf yields little gain.

For this reason, we introduce Opportunistic Redundancy Pooling (ORP). This is a method to pool backup nodes such that there is no additional overhead on bandwidth (and synchronization, in the case of hot standbys). Another advantage with this method is that VInfs with different reliability guarantees can be pooled together. It makes use of the discrete steps of the curves as shown in Fig. 1. For example, in the case where a VInf with 30 critical nodes and $p = 0.03$ needs 8 backup nodes, the reliability is 99.9998544522%, due to the discrete steps on the curve. The excess 0.0008544522% reliability can be "lent" to other VInfs that require no more than 8 backup nodes. Conversely, it can also be viewed that the residual excess from a few VInfs are pooled to reduce the number of backups a VInf needs, as shown in Fig. 2.

Suppose there are $m+1$ VInfs which require $k_0, k_1, \ldots, k_m$ backup nodes for reliability guarantee of $r_0, r_1, \ldots, r_m$, respectively, and $k_0 \geq \sum_{i=1}^{m} k_i$. VInf-0 pools its backup nodes with VInf-i, $i > 0$, and each backup node backs up VInf-0 and at most one other VInf. We assume that the recovery protocol when activating backup nodes prioritize VInf-i over VInf-0, and VInf-i uses no more than k_i backup nodes after recovery. Then, the reliabilities of VInf-i for $i > 0$ are unchanged, and the pooling is admissible only if the reliability of VInf-0 r_0' after pooling backups is no less than r_0.

Define $z^{\text{VInf}}(k, y)$ as the probability that a total of y nodes fail in a VInf with k backup nodes, i.e.,

$$z^{\text{VInf}}(k, y) = \sum_{x=0}^{\min(n,y)} \binom{k}{y-x} p^{y-x}(1-p)^{k-(y-x)} f^{\text{VInf}}(x), \tag{3}$$

where n is the number of nodes of that VInf. The reliability of VInf-0 after pooling is then

$$r_0' = 1 - \sum_{x=0}^{k'} Pr\left(\begin{array}{c} x \text{ of } k' \text{ backups are down, or} \\ \text{used by VInf-1, } \ldots, \text{ VInf-}m \end{array} \right) \times Pr\left(\begin{array}{c} \text{more than } k_0 - x \text{ nodes fail} \\ \text{from VInf-0 with } k_0 - k' \text{ backups} \end{array} \right). \tag{4}$$

The first term is the probability mass function (pmf) of the sum of m independent VInfs with k_i backup nodes each. The pmf of each independent event $q^{\text{VInf-}i}(x)$ is

$$q^{\text{VInf-}i}(x) = \begin{cases} z^{\text{VInf-}i}(k_i, x) & , 0 \leq x < k_i \\ 1 - \sum_{y=0}^{k_i - 1} z^{\text{VInf-}i}(k_i, y) & , x = k_i. \end{cases} \tag{5}$$

Convolving all m pmfs give the first term of (4) to be

$$Q(x) = \mathcal{F}^{-1}\left(\prod_{i=1}^{m} \mathcal{F}\left(q^{\text{VInf-}i}(x)\right) \right), \tag{6}$$

where $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$ is the Discrete Fourier Transform (DFT) and its inverse, respectively, of minimum length k'. It is, however, more convenient to keep the length to be at least k_0 so that more VInfs can be pooled in future without having to recompute m DFTs again. Then, (4) reduces to

$$r_0' = 1 - \sum_{x=0}^{k'} Q(x) \left(1 - \sum_{y=0}^{k_0 - x} z^{\text{VInf-0}}(k_0 - k', y) \right). \tag{7}$$

The time complexity to decide whether VInfs $1, \ldots, m$ can be pooled with VInf-0 is bounded by the m DFTs, which evaluates to $O(mk \log k)$.

Refer to Fig. 3 for a graphical explanation with two VInfs. VInf-0 shares some of its backup nodes with one other VInf under ORP. We show two cases of VInf-1, one with reliability requirement of 99.9% and another with 99.9999%. VInf-0's reliability requirement is kept at 99.999%. For simplicity, failure rates of all critical nodes are set to be independent and uniform at $p = 0.01$.

In the top plot, the number of backup nodes k_0 for VInf-0 increases in a step-wise fashion similar to Fig. 1 as the number of critical nodes increase. The step-wise increase in k_0 creates opportunities for VInf-1 to reuse some of VInf-0's backup nodes as there is much excess in VInf-0's reliability prior to pooling (see the shaded area in the middle plot). The lower plot shows the maximum number of critical nodes VInf-1 while reusing VInf-0's backup nodes, and the respective number of backup nodes reused are shown in the top plot. Since VInf-1 is essentially utilizing VInf-0's excess reliability, the peaks and valleys of the curve in the lower plot follows that of the middle plot. It can be observed, too, that the size of VInf-1 is significant as compared to that of VInf-0, and the number of backups conserved is up to 50%.

The advantage of ORP can be summarized as follows.

No degradation for large n. The pooling scheme makes use of the excess reliability arise from discrete steps in k. Hence, there will always be gaps that can be filled with VInfs that need smaller k.

Pooling over different r. This scheme allows for VInfs of arbitrary reliability requirements to be pooled together.

Flexibility in adding VInfs. A new VInf-$m+1$ can always be added into VInf-0's pool of backup nodes so long as VInf-0's new reliability computed from (7) is still above the required r_0. All previous m DFTs can be stored to speed up this admission control procedure by a factor of $O(m)$.

Flexibility in removing VInfs. VInf-i can always be removed from the pool since VInfs other than VInf-0 are unaffected, and VInf-0 will have its effective reliability increased. Conversely, if VInf-0 is removed, the other VInfs simply reclaim the respective backups as their own, which can be pooled with new incoming VInfs.

There are other ways of extending ORP. We do not study these cases due to three major reasons: (i) there is a compromise of flexibility in dynamically adding and removing VInfs, (ii) the gains may be marginal as compared to the initial sharing, and (iii) the time complexity to re-evaluate of the reliabilities of all pooled VInfs may be high.

Pooling VInfs this way is opportunistic, since we do not predict the statistics of future incoming VInfs. In general, however, VInf-0 should be the one with the largest number

This means either the VInf needs lower reliability guarantee, has smaller number of critical nodes, has a skewed $f(x)$ that gives smaller k, or all of the above.

Having priority enables us to focus on getting the new r_0' to be greater than the guarantee r_0, which leads to a simple computation of r_0'. As described later, the method of computing also supports an incremental evaluation of r_0' when more VInfs are admitted into the system.

For performance reasons, the length could be kept at $2^{\lceil \log_2 k_0 \rceil}$ to make use of Fast Fourier Transform algorithms.

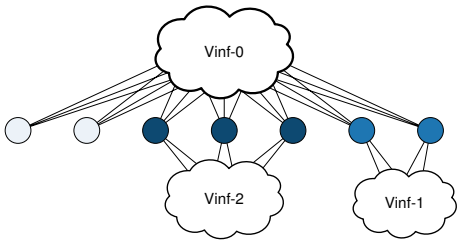


Figure 2: Pooling backup nodes. This can be as: either (i) VInf-0 "lends" some of its backup nodes to other VInfs, or (ii) VInfs 1 and 2 collectively "lend" their backup nodes to VInf-0.

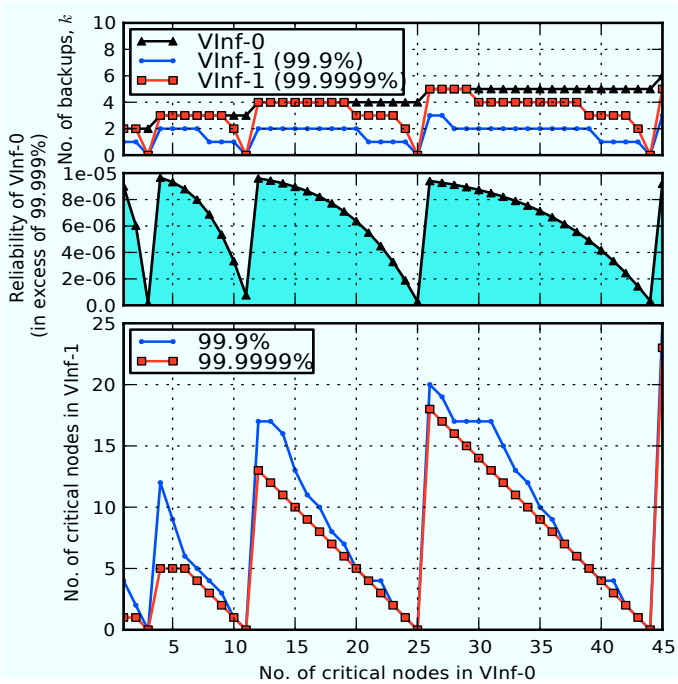


Figure 3: VInf-0 shares its k_0 backup nodes with VInf-1 ($p = 0.01$). The lower plot shows the maximum number of critical nodes VInf-1 can have while reusing VInf-0's backup nodes for two cases of reliability guarantee, while maintaining VInf-0's reliability above 99.999%. The upper plot shows the corresponding number of VInf-0's backup nodes used by VInf-1, and the middle plot shows the excess reliability from VInf-0 that has been reused for VInf-1.

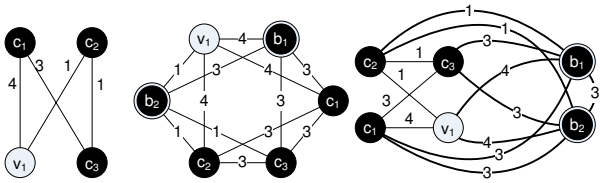
of backup nodes as this allows for more degrees of freedom in choosing other VInf-i to pool backup nodes with.

4. PROVIDING RELIABLE LINKS

The virtual infrastructure $G = (V, E)$ has to be preserved when backup nodes resume execution of failed critical nodes. This translates to ensuring that every backup node has guaranteed bandwidth to all neighbors of all critical nodes.

4.1 Minimum Redundant Links

It is possible to minimize the total number of links while providing redundancy for a VInf. Harary and Hayes [14] studied the problem of constructing a new graph $G' = (N \cup B, E')$ with minimum links (i.e., $|E'|$) such that upon the removal of any $k = |B|$ nodes (i.e., k node failures), the resultant graph always contain the original VInf G.



(a) VInf with three critical nodes. (b) Least number of links for VInf with two backups. (c) Addition of redundant (bold) links L to VInf for two backups.

Figure 4: (b) and (c) show two different ways of preserving VInf (a) to account for two redundant nodes. While the former utilizes 1 less link and 1 less bandwidth unit, it requires some nodes to be rearranged in the recovery phase.

However, this poses a limitation since the result only guarantees graph isomorphism and not equality. In other words, there may be a need to physically swap remaining VMs while recovering from some failure in order to return to the original infrastructure G. Recovery may then be delayed, or require more resources for such swapping operations.

We illustrate this using the example in Fig. 4. The new graph G' in Fig. 4b is obtained using Theorem 2 in [14]. If nodes b_1 and c_3 fail, the only way to recover is to have c_3 to be in the position of the current c_2 and backup node b_2 assuming the role of c_2. The problem here is that node c_2 is not a backup for node c_3 in the first place! Hence, the recovery procedure is lengthened to two steps: (i) recover node c_3 at backup node b_2, and then (ii) swap nodes c_2 and c_3. This problem will always arise no matter where the backup nodes are placed in G'.

Furthermore, deriving optimal graphs G' with minimal links for any general graph G has exponential complexity. To the best of the authors' knowledge, optimal solutions are found only on regular graphs such as lines, square-grids, circles, and trees [14, 1, 12].

4.2 Redundant Links without Swapping

To overcome the aforementioned limitations, we choose to use the following set of redundant links, at the expense of incurring more redundant resources. Formally, the set of redundant links L that are added to G is a union of links

$$L^1 = \{(b, u) \,|\, \exists (c, u) \in E, \forall b \in B, u \in N, c \in C\} \quad (8)$$

$$L^2 = \{(a, b) \,|\, \forall a, b \in B\}, \quad (9)$$

where B and C are the sets of backup and critical nodes, respectively. The first set L^1 connects all backup nodes to all neighbors of all critical nodes, and the second set L^2 interconnects all backup nodes since two critical nodes may be neighbors of each other and may fail simultaneously. The latter set can be omitted if there are no links between any critical nodes. See Fig. 4c for an example.

Adding L to G is much more straightforward and does not suffer from the aforementioned swapping / rearrangement

Theorem 2 works only on unweighted graphs. We subsequently obtained the minimum link weights through exhaustive iteration.

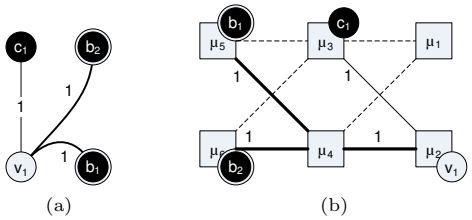


Figure 5: On the left is a VInf with redundant links to two backup nodes. Suppose that the virtual nodes (circles) are deployed at the physical nodes (squares) on the right. Unutilized links are dotted lines and redundant links are in bold. Only 1 unit of bandwidth needs to be reserved on link (μ_2, μ_4).

problem. More importantly, when pooling backups across VInfs, redundant links L can be added without affecting other existing VInfs which use the same backup nodes.

The number of redundant links in L may seem large ($O(nk+k^2)$), but the amount of physical bandwidth reserved can be reduced while embedding them. This is because not all links will be in use at the same time. A simple example in Fig. 5 can illustrate this. The small VInf in Fig. 5a consists of two nodes and a link between them with 1 bandwidth unit. One of the nodes is critical and is backed up by two redundant nodes. Suppose that due to limited available compute capacities, the physical deployment of the virtual nodes is that of Fig. 5b. If the redundant links are embedded verbatim into the physical infrastructure, the link (μ_2, μ_4) would require 2 units. However, it is only necessary to reserve 1 unit on this link, since at most 1 backup node will be in use at any time. Minimizing the physical footprint of these redundant links during physical embedding give rise to a tight coupling between the virtual nodes and links.

5. RESOURCE ALLOCATION: A MIXED INTEGER PROGRAMMING PROBLEM

We use a joint node and link resource allocation approach to address the tight coupling between the virtual nodes and links of a VInf. In [8], a multi-commodity flow (MCF) problem is formulated to jointly allocate nodes and links of a VInf to physical infrastructure. We adapt the MCF problem with additional constraints to solve for the minimum bandwidth used on redundant links.

The MCF problem is a network flow problem where the objective is to assign flows between sources and destinations in a network. The virtual links of a VInf can be seen as flows between virtual nodes. To determine the actual locations of the virtual nodes, the physical network is appended with virtual nodes and "mapping" links connecting every virtual node to their possible physical locations, i.e., links (u, μ) for all virtual nodes u and physical nodes μ are appended to the set of physical links $\mathcal{E}$. The first physical node in which a flow passes through will be the location of the virtual node of that flow. Additional constraints are added to the MCF to ensure that a virtual node has only one physical location.

5.1 Allocation Constraints

Denote by $\rho_{u\mu}$ a binary variable that represents the mapping between a physical node and a virtual node, i.e., $\rho_{u\mu} =$ 1 if a virtual node u is mapped onto physical node μ, 0 otherwise. Compute capacity constraints on the physical nodes are captured as follows

$$\rho_{u\mu}\gamma_u \leq \Gamma_\mu, \quad \forall u \in N \cup B, \forall \mu \in \mathcal{N}'. \tag{10}$$

The set of backup nodes B may be omitted from the above if backup nodes are reused from a redundancy pool, and the compute capacity reserved is already more than the maximum of that of the new critical nodes, i.e, $\max \gamma_c$. Otherwise, the above constraints may be included and the RHS is the deficit compute capacity.

Bandwidth constraints and link mappings are derived from the MCF problem. A virtual link (u, v) between two virtual nodes u and v can be seen as a flow between source and destination under the MCF problem. Due to the inclusion of redundant links, we define four types of flows:

Virtual links E: flows between two virtual nodes $u, v \in N$. The amount of bandwidth used on a link (i, j) is denoted by $\ell_E^{uv}[ij]$.

L^1 flows: flows between a backup node $a \in B$ and a neighbor $v \in N$ of some critical node. The amount of bandwidth on L^1 flows depend on which critical node c the backup node a recovers, and how much bandwidth can be "overlapped" across different failure scenarios. As such, we denote by $\ell_{L^1}^{acv}[ij]$ the amount of bandwidth used on a link (i, j) when such a recovery occurs. This allows us to model the overlaps between redundant flows.

Aggregate flows: flows on a link between redundant nodes B and the neighbor $v \in N$ of some critical node. This reflects the actual amount of bandwidth reserved after overlaps on link (i, j). We denote this by $\ell_o^v[ij]$.

L^2 flows: flows between two backup nodes $a, b \in B$. The amount of bandwidth used on a link (i, j) is denoted by $\ell_{L^2}^{ab}[ij]$. Unlike L^1 flows, we do not model any possible overlapping of these redundant links with L^1. This is to ensure the L^2 flows can be easily reused when sharing with other VInfs.

The flows E, L^1 and L^2 follow the conservation of flow equations. Due to space limitations, we omit these flow conservation constraints and proceed to describe the new constraints with respect to redundant and mapping links.

The actual amount of bandwidth reserved on a physical link (μ, ν) after considering overlaps of L^1 flows can be captured by the following constraint:

$$\sum_{\substack{a \in B, c \in C', \\ (c,v) \in E}} \ell_{L^1}^{acv}[\mu\nu] \leq \ell_o^v[\mu\nu], \quad \forall (\mu, \nu) \in \mathcal{E}, \forall (a, v) \in L^1, \quad \forall C' \in C, |C'| \leq k. \tag{11}$$

The subset of critical nodes C' represent a possible failure scenario where at most k critical nodes fail. The RHS captures the maximum bandwidth used in those cases. Unfortunately, the caveat here is that this leads to an exponential expansion of constraints when k goes large. The impact of overlapping redundant links, however, is significant as can be observed in our evaluation [24].

The last set of constraints defines the link capacity on physical links (μ, ν) and mapping links (u, μ).

$$\sum_{(u,v) \in E} \left[\ell_E^{uv}[\mu\nu] + \ell_E^{uv}[\nu\mu]\right] + \sum_{v \in N} \left[\ell_o^v[\mu\nu] + \ell_o^v[\nu\mu]\right] + \sum_{a,b \in B} \left[\ell_{L^2}^{ab}[\mu\nu] + \ell_{L^2}^{ab}[\nu\mu]\right] \leq \Lambda_{\mu\nu}, \ \forall (\mu, \nu) \in \mathcal{E}, \tag{12}$$

$$\sum_{(u,v)\in E}\left[\ell_E^{uv}[u\mu]+\ell_E^{uv}[\mu u]\right]+\sum_{v\in N}\left[\ell_o^{v}[u\mu]+\ell_o^{v}[\mu u]\right]$$
$$+\sum_{a,b\in B}\left[\ell_{L^2}^{ab}[u\mu]+\ell_{L^2}^{ab}[\mu u]\right]\le\Lambda\rho_{u\mu},\quad\forall(u,\mu)\in N\times\mathcal{N}. \tag{13}$$

The first constraint accounts for all flows on a physical link (μ,ν) in *both* directions, and the total should be less than the physical remaining bandwidth $\Lambda_{\mu\nu}$. For the second constraint, the LHS is similar in that it accounts for all flows on the mapping link (u,μ), and Λ is an arbitrary large constant. This way, the mapping variable $\rho_{u\mu}$ will be set to 1 if there are non-zero flows on that link in either direction.

For the mapping of the physical and virtual nodes, straightforward constraints are expressed. We refer to [24] for the details. We only mention that additional mapping restrictions can be included in these constraints, such as specifying the preference (or exclusion) of specific physical nodes to allocate a given virtual node, or separation constraints enforcing that separate virtual nodes do not share the same physical node.

5.2 Objective Function and Approximation

We seek to minimize the amount of resources used for a VInf. The objective function of the adapted MCF is then

$$\min\sum_{\mu\in\mathcal{N}}\alpha_\mu\sum_{u\in N\cup B}\rho_{u\mu}\gamma_\mu+\sum_{(\mu,\nu)\in\mathcal{E}}\beta_{\mu\nu}\times\Bigg[\sum_{v\in N}\left[\ell_o^{v}[\mu\nu]+\ell_o^{v}[\nu\mu]\right]$$
$$+\sum_{a,b\in B}\left[\ell_{L^2}^{ab}[\mu\nu]+\ell_{L^2}^{ab}[\mu\nu]\right]+\sum_{(u,v)\in E}\left[\ell_E^{uv}[\mu\nu]+\ell_E^{uv}[\nu\mu]\right]\Bigg] \tag{14}$$

where α_μ and $\beta_{\mu\nu}$ are node and link weights, respectively. To achieve load balancing across time, the weights can be set as $\frac{1}{\Gamma_\mu+\epsilon}$ and $\frac{1}{\Lambda_{\mu\nu}+\epsilon}$, respectively.

The variables of this linear program are the non-zero real-valued flows ℓ and the boolean mapping variables ρ. The presence of boolean variables turns the linear program into a NP-Hard problem. An alternative is to relax the boolean variables to real-valued variables, obtain an approximate virtual node embedding by picking a map with the largest $\rho_{u\mu}$, and re-run the same linear program with the virtual nodes assigned to obtain the link assignments [8].

6. EVALUATION

In this section, we evaluate the performance of the system when allocating resources with and without redundancy pooling and redundant bandwidth reduction, labeled **share** and **noshare** respectively. In particular, we focus on the resource usage of the physical infrastructure and the admission rates of VInf requests. We further compare that to a system where VInfs do not have reliability requirement, i.e. zero redundancy (labeled **nonr**), as a baseline to gauge the additional amount of resources consumed for reliability.

Our simulation setup is as follows. The physical infrastructure consists of 40 compute nodes with capacity uniformly distributed between 50 and 100 units. These nodes are randomly connected with a probability of 0.4, and the bandwidth on each physical link is uniformly distributed between 50 and 100 units. VInf requests arrive randomly over a timespan of 800 time slots and the inter-arrival time is assumed to follow the Geometric distribution at a rate of 0.75 per time slot. The resource lease times of each VInf follows the same distribution but at a rate of 0.01 per time slot. A high request rate and long lease times ensures that the physical infrastructure is operating at high utilization. Each VInf consists of nodes between 2 to 10, with a compute capacity demand of 5 to 20 per node. Up to 90% of these nodes are critical and all failures are independent with probability 0.01. Connectivity between any two nodes in the VInf is random with probability 0.4, and the minimum bandwidth on any virtual link is 10 units. We scale the reliability guarantee of each VInf from 99.5% to 99.995% while the maximum bandwidth of any virtual link is limited to 30 units. Another set of results that scales the maximum bandwidth is presented in [24]. A custom discrete event simulator written in Python is used to run this setup on the Amazon EC2 platform, and the relaxed mixed integer programs are solved using the open-source CBC solver [5].

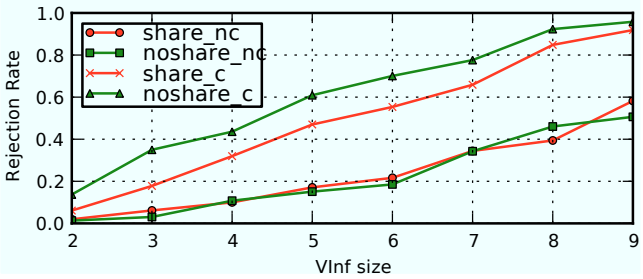


Figure 6: Rejection rates of VInfs according to the sizes. Suffix c refers to VInfs with critical nodes ($r = 99.99\%$) and maximum bandwidth 35. nc refers to those without.

Fig. 7 shows the mean performance of the three cases **share**, **noshare** and **nonr** across 10 simulation runs. **noshare** has the least acceptance rate and VInf occupancy, and more backup nodes per VInf than **share**, which is able to pool redundancies and efficiently reuse backups.

CPU usage per VInf is slightly higher in **noshare** than **share**. The redundant nodes in **share** consume less resource (suffix _redC) than that in **noshare** despite admitting more VInfs. This gap widens with increased reliability guarantee, showing the effect of redundancy pooling.

The bandwidth usage per VInf is actually smaller for **share** than **noshare** although larger bandwidth is dedicated for redundancy (suffix _redB) in **share** than **noshare**. In the latter case, the redundant links use less bandwidth due to lower acceptance rates for VInfs with critical nodes rather than being more efficient with resources as illustrated in Fig. 6. This shows that **noshare** is highly inefficient; expansion of redundant backups and links without pooling have led to larger granularity in VInf resource requests. **share** conserves more bandwidth and is able to admit larger sized VInfs.

In summary, increasing redundant nodes and expanding a VInf with backup links leads to VInfs with larger granularity. If the physical infrastructure admits these expanded VInfs verbatim as in the case of **noshare**, much inefficiencies can occur. VInfs that have more nodes, bandwidth, or higher reliability requirement (or all of them) get expanded much larger than **share**, leading to more rejections and losses in revenue. Smaller VInfs, especially those with no critical nodes, are more readily admitted and it is almost impossible for larger VInfs to be admitted. Although more CPU and bandwidth are used in the **noshare** case, there is substantially less VInfs than **share** (as much as 24%) present

in the physical infrastructure. This is so even where more bandwidth is dedicated for redundant links in **share** as more VInfs with more critical nodes get admitted. In comparison to the case where no reliability is guaranteed (**nonr**), the number of VInfs that can be admitted dropped by at most 20% and the largest drop in acceptance rate goes from 65% to 51% when compared to **share**. When compared to **noshare**, the figures are 38%, and from 65% to 41% respectively. Hence, the resources required for provisioning reliability is quite significant.

7. RELATED WORK

Network virtualization is a promising technology to reduce the operating costs and management complexity of networks, and it is receiving an increasing amount of research interest [7]. Reliability is bound to become a more prominent issue as infrastructure providers move toward virtualizing their networks over cheaper commodity hardware [3].

Analysis on the reliability of overlay networks in terms of connectivity in the overlays has been developed [17]. Unfortunately, it is not applicable to our problem as we are concerned with critical virtual nodes and embedding them as an entire infrastructure with reliability guarantees.

Fault tolerance is provided in data centers [19, 13] through excessive redundant nodes and links organized in a special way. These works provide reliability but do not customize reliability guarantees to embedded VInfs.

While we are not aware of works studying the allocation of reliable virtual networks, [23] considered the use of "shadow VNet", namely a parallel virtualized slice, to study the reliability of a network. However, such slice is not used as a back-up, but as a monitoring tool, and as a way to debug the network in the case of failure. [22] considered the use of virtualized router as a management primitive that can be used to migrate routers for maximal reliability.

Meanwhile some works address node fault tolerance at the virtualization level. Bressoud [4] was the first few to introduce fault tolerance at the hypervisor. Two virtual slices residing on the same physical node can be made to operate in sync through the hypervisor on the same physical node. Others [10, 9] have made progress for the virtual slices to be duplicated and migrated over a network. Various duplication techniques and migration protocols were proposed for different types of applications (web servers, game servers, and benchmarking applications) [9]. Remus [10] and Kemari [21] are two other systems that allows for state synchronization between two virtual nodes for full, dedicated redundancy. However, these works focus on the practical issues, and do not address the resource allocation issue.

VNsnap [15] is another method to take static snapshots of an entire virtual infrastructure to some reliable storage, in order to recover from failures. This can be stored as reliably in a distributed manner as replicas [6], or as parities [11, 25]. VNsnap does not address synchronization, nor guarantee sufficient resources for recovery from snapshots.

Fundamentally, there are methods to construct topologies for redundant nodes that address both nodes and links reliability [1, 12]. Based on some input graph, additional links are introduced such that the least number is needed. However, a node failure, in this case, may involve migrations among the remaining nodes to preserve the original topology. This may not be suitable in a scenario where migrations may disrupt other running parts of the network.

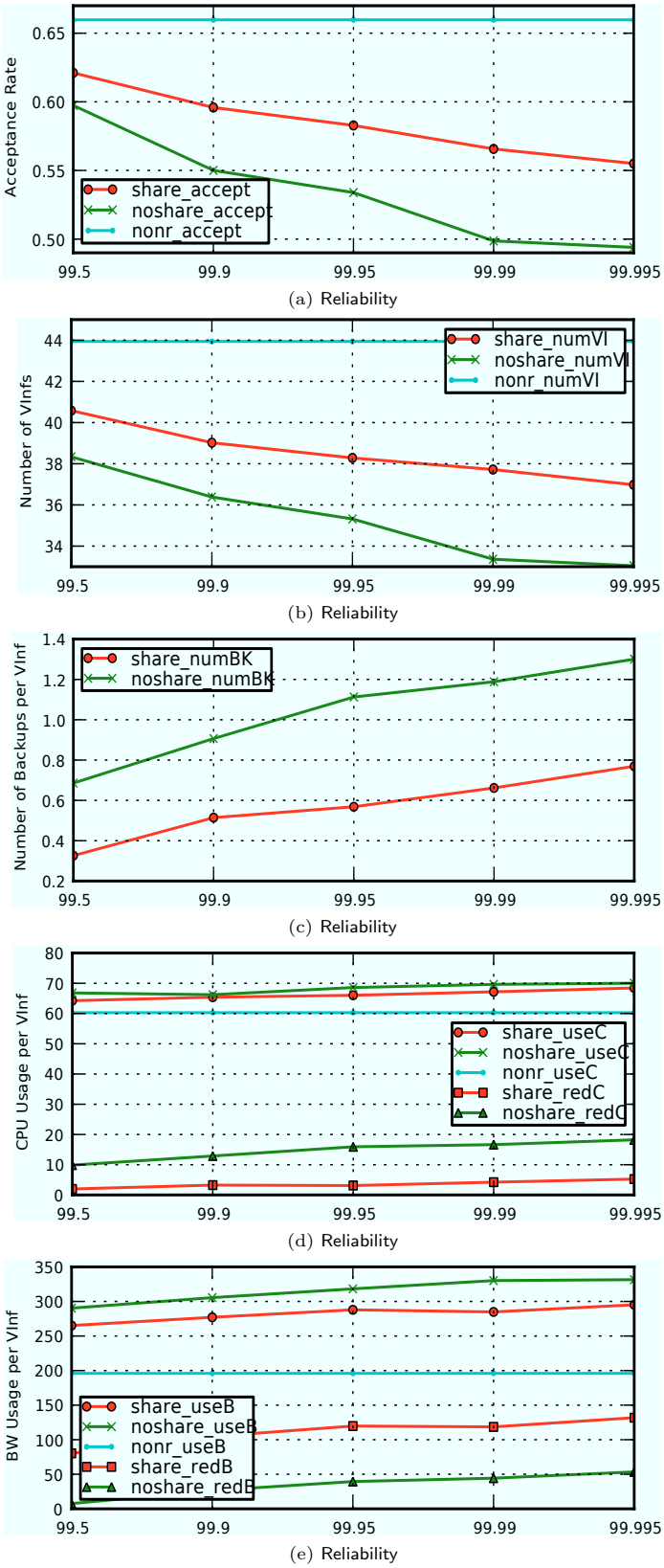


Figure 7: Figures above compare the performance of the system when backup nodes and links are shared against unshared. This is studied under varying reliability guarantees. nonr is the baseline case where VInfs are admitted without any redundancy.

Our problem involves virtual network embedding [8, 18] with added node and link redundancy for reliability. Our model employs the use of path-splitting [26], which allows a link to be split over multiple routes such that the aggregate flow across those routes equal to the demand between the two nodes. This gives more resilience to link failures and allows for graceful degradation. A related work that does not use path-splitting for embedding reliability is [16].

8. CONCLUSION

We considered the problem of efficiently allocating resources in a virtualized physical infrastructure for Virtual Infrastructure (VInfs) with reliability guarantees, which is guaranteed through redundant nodes and links. The resource allocation approach disregards whether the redundant nodes are passive or hot standbys so long as ample resources are available when recovering from failures.

Since a physical infrastructure hosts multiple VInfs, it is more resource efficient to share redundant nodes between VInfs. We introduced a pooling mechanism ORP to share these redundancies for both independent and cascading types of failures. The physical footprint of redundant links can be reduced as well, by considering the maximum over all failure scenarios while allocating resources with a linear program adapted from the Multi-Commodity Flow problem. Both mechanisms have significant impact in conserving resources and improving VInf acceptance rates.

9. REFERENCES

[1] M. Ajtai, N. Alon, J. Bruck, R. Cypher, C. Ho, M. Naor, and E. Szemeredi. Fault tolerant graphs, perfect hash functions and disjoint paths. *Symposium on Foundations of Computer Science*, 0:693–702, 1992.

[2] D. Atwood and J. G. Miner. Reducing Data Center Cost with an Air Economizer. `http://www.intel.com/it/pdf/Reducing_Data_Center_Cost_with_an_Air_Economizer.pdf`, Aug. 2008.

[3] S. Bhatia, M. Motiwala, W. Muhlbauer, Y. Mundada, V. Valancius, A. Bavier, N. Feamster, L. Peterson, and J. Rexford. Trellis: a platform for building flexible, fast virtual networks on commodity hardware. In *ACM CONEXT '08*, pages 1–6, 2008.

[4] T. C. Bressoud and F. B. Schneider. Hypervisor-based fault tolerance. *ACM Trans. Comput. Syst.*, 14(1):80–107, 1996.

[5] CBC: Coin-or Branch and Cut. `https://projects.coin-or.org/Cbc`.

[6] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A Distributed Storage System for Structured Data. In *USENIX OSDI '06*, Nov. 2006.

[7] N. M. M. K. Chowdhury and R. Boutaba. Network virtualization: State of the art and research challenges. *IEEE Communication Magazine*, 47(7):20–26, July 2009.

[8] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba. Virtual Network Embedding with Coordinated Node and Link Mapping. In *IEEE INFOCOM '09*, Apr. 2009.

[9] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *USENIX NSDI'05*, May 2005.

[10] B. Cully, G. Lefebvre, D. M. M. Feeleyand, and N. Hutchinson. Remus: High availability via asynchronous virtual machine replication. In *USENIX NSDI '08*, Apr. 2008.

[11] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Decentralized erasure codes for distributed networked storage. *IEEE Trans. Inf. Theory*, 52(6):2809–2816, June 2006.

[12] S. Dutt and N. R. Mahapatra. Node-covering, error-correcting codes and multiprocessors with very high average fault tolerance. *IEEE Trans. Comput.*, 46(9):997–1015, 1997.

[13] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. Bcube: A high performance, server-centric network architecture for modular data centers. In *SIGCOMM*, Aug. 2009.

[14] F. Harary and J. P. Hayes. Node fault tolerance in graphs. *Networks*, 27(1):19–23, 1996.

[15] A. Kangarlou, P. Eugster, and D. Xu. VNsnap: Taking Snapshots of Virtual Networked Environments with Minimal Downtime. In *IEEE/IFIP DSN '09*, June 2009.

[16] G. Koslovski, W.-L. Yeow, C. Westphal, T. T. Huu, J. Montagnat, and P. Vicat-Blanc. Reliability support in virtual infrastructures. In *IEEE CloudCom*, Dec. 2010.

[17] K. Lee, H.-W. Lee, and E. Modiano. Reliability in Layered Networks with Random Link Failures. In *IEEE INFOCOM '10*, Mar. 2010.

[18] J. Lischka and H. Karl. A Virtual Network Mapping Algorithm based on Subgraph Isomorphism Detection. In *VISA '09*, Aug. 2009.

[19] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. Portland: A scalable fault-tolerant layer 2 data center network fabric. In *SIGCOMM*, Aug. 2009.

[20] P. D. T. O'Connor, D. Newton, and R. Bromley. *Practical reliability engineering*. John Wiley and Sons, fourth edition, 2002.

[21] Y. Tamura, K. Sato, S. Kihara, and S. Moriai. Kemari: VM Synchronization for Fault Tolerance. In *USENIX '08 Poster Session*, June 2008.

[22] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford. Virtual routers on the move: live router migration as a network-management primitive. In *SIGCOMM*, Aug. 2008.

[23] A. Wundsam, A. Mehmood, A. Feldmann, and O. Maennel. Network troubleshooting with shadow vnets. In *SIGCOMM Posters & Demos*, Aug. 2009.

[24] W.-L. Yeow, C. Westphal, and U. C. Kozat. Designing and Embedding Reliable Virtual Infrastructures. Technical report, Docomo USA Labs, Mar. 2010. `arXiv:1005.5367 [cs.NI]`.

[25] W.-L. Yeow, C. Westphal, and U. C. Kozat. Highly Available Virtual Machines with Network Coding. In *IEEE INFOCOM '11 Mini-conference*, Apr. 2011.

[26] M. Yu, Y. Yi, J. Rexford, and M. Chiang. Rethinking virtual network embedding: substrate support for path splitting and migration. *SIGCOMM Comput. Commun. Rev.*, 38(2):17–29, 2008.