TP1.2.2 – DHCP – preliminary draft 2

This document captures the motivation and detailed design of the workspace service's DHCP support released in version TP1.2.2.

See http://workspace.globus.org/vm/TP1.2.2/ for more information.

We assume the reader has a basic familiarity with the workspace service TP1.2 "resource pool model" and its networking mechanisms.

I. Requirements

Previously, to deliver IP/DNS/hostname information to a VM, the service used an ad hoc name/value pair mechanism. It sent the information by writing to a well known file in the VM before it launched, and if writing that file failed, sent it over kernel parameters. In both cases the information was parsed and acted on by an init script in the VM. (kernel parameters were not the default because that buffer is normally a limited size and would not accommodate information for many NICs)

A. Grid control of addresses

It is important to continue sending static information to specific VMs: IP management of new VM deployments at a site, while ultimately arbitrated and managed by the site, happens in the grid context.

- The network configuration request (IP pool or specific IP) by remote clients is managed by grid context authorization policies. While it is possible these policies could be expressed in a local representation and a translation would need to occur between local and grid contexts, this would be far more cumbersome than the familiar *grid-mapfile* example of such a translation. Authorization decisions about workspace network configuration take two general forms:
    - Straight identity and attribute based authorization using established technologies.
    - Authorization decisions can also take into account any information in the workspace request in combination with identity or attribute credentials: for example, image size, number of NICs in the total request (including multiple workspaces at once, virtual clusters), and bandwidth SLAs. These attributes of a request may have significant effects on which addresses to allow the workspace to have.
- If a specific IP is requested, this may require grid orchestration, for example in the context of workspace live migration (where the VM is not shut down and hence needs the same address at the new site) or working with VPNs/virtual networks (where the namespace inside the virtual network will usually be governed by a grid context central authority for that network, the so called "Grid DHCP").
- Local clients deploying via the workspace service currently require grid credentials or require a local agent to act on behalf of the user, using a service credential. Clients deploying VMs by some other means, for example manually, will need to have their own IP management method. While this can work alongside the workspace service, the service is not involved in any of the decisions or monitoring of those network configurations.

B. Deployment issues

Sending the information via DHCP is a natural step, it requires no OS modifications in all major operating systems and thus makes workspace images more standard and less cumbersome to create. However, to achieve this convenience there are some challenges to overcome:

i. No site-DHCP integration

DHCP servers are common for clusters, even if the nodes are assigned addresses statically (what is known as a long term or infinite DHCP lease which can be based on MAC address). It is one way of centralizing address management. Sites supporting workspaces can not just rely on a site's locally deployed DHCP server:

- Monitoring: the workspace service needs to know on a NIC by NIC basis what information was assigned, to report it in the grid context for use by applications

- Decision making: as stated above with respect to authorization decisions, the workspace service also needs to decide (potentially resolving complicated situations) which exact addresses get assigned on a NIC by NIC basis.
- (If every decision were explicitly sent to the DHCP server and the server just executed our wishes for each specfic MAC->IP mapping, monitoring of the server's decisions would not be necessary, we would just trust that the server gave the correct IP to the correct MAC address)
- The site DHCP may not even be on the same LAN, for example the workspace could be joining a virtual network and the address would only make sense in that context (and further, the address might even conflict with local addresses).

To relieve these restrictions would require a programmatic (and in many cases remote) interface between the local DHCP server and the workspace service:

- Writing and supporting alterations to even just the popular permutations of deployments would be a lot of work.
- Site administrators would resist such an invasive requirement
    - It would require deploying unsupported, altered versions of their time tested DHCP servers.
    - It would require the workspace service had rights to alter the DHCP server and thus a security breach (or bug) could affect the entire network.
    DHCP is a core network service and this resistance is a very fair position to take.

ii. Non-interference, "invisibility" of solution

Because we should not integrate with a site's current DHCP server (and in case they don't even have one), a DHCP solution for VMs deployed in the hypervisor pool brings two requirements:

- No request from workspaces should ever touch the site's actual network to avoid policy conflicts:
    - The site may have an open policy regarding DHCP requests, assuming all physical entities on the network are connected to the network by the administrators only: no user can initiate DHCP requests because ports 67 and 68 are privileged. While it can be arranged for guest workspaces to be stood up without giving root access for the remote client, the DHCP request from the VM will always be issued in the standard way, by a privileged process using ports 67 and 68, and could thus still conflict with this assumption. In this case of an open policy, the DHCP rules would need to be modified anytime a block of MAC or IP addresses is allocated for grid use via workspaces.
    - The site may have detailed policies regarding DHCP requests (for example, groups or static assignments keyed by the MAC address of the

requester). In this case of a detailed policy, the DHCP rules would still need to be modified anytime a block of MAC or IP addresses is allocated for grid use via workspaces (and it may be harder to think about and orchestrate).

- o The site may not have a DHCP server, in which case DHCP requests will add unwanted noise to the network.
- o If there is no DHCP server, other users may have root on the network and they will be able to listen on the DHCP port and respond to workspace requests [[TF: is this a valid concern… ?]]

- Any request generated from a VM on the hypervisor node that is not being managed by the workspace service must make it to the site's network to allow for site controlled VMs to use the site DHCP server if that is their deployment model for VMs not managed by the workspace service.

These two requirements can be summed up by the directive to *be invisible to any current network deployment*, no matter how simple or complicated it is.

II. Design overview

To meet the requirements of network configuration delivery via DHCP while remaining "invisible" to any current network deployment, we will require that the administrator install a DHCP daemon on every resource pool node and adjust the resource pool node's iptables configuration to give us the behaviors outlined in the Requirements section above.

We've chosen a specific DHCP server to support for TP1.2.2. The software we produce will be factored as well as possible to accommodate different DHCP server implementations in the future if that is required.

Important: note that we do not allow this DHCP server to choose the information in its DHCP reply from a range of addresses. The particular address to use is chosen by the GT4 workspace service on the head node and sent to workspace-control. We are using the DHCP protocol to deliver this information into the VM by assigning one-to-one mappings from MAC to IP address. We are not allowing the DHCP server to make any choice, it is only being used as a standard way to get the information into the VM as it is booting.

A. ISC DHCP

We've chosen the Internet Systems Consortium's DHCP server:
- http://www.isc.org/products/DHCP

This is an industry standard DHCP server and freely available for use and distribution via a BSD-like license:
- http://www.isc.org/sw/dhcp/dhcp-copyright.php

i. Availability

It is supported on almost every relevant architecture and OS distribution (quote from the ISC website):

- **NetBSD**: Full functionality
- **FreeBSD**: Full functionality
- **Linux**: Full functionality with Linux 2.0.33 and later kernels.
- **Mac OS X**: Full functionality.
- **BSDI BSD/OS**: Full functionality.
- **DEC Alpha OSF/1**: Full functionality.
- **SunOS 4.1.4**: Full functionality.
- **Ultrix**: Full functionality.
- **Solaris 2.5, 2.6 and 7 on sparc**: Full functionality.

- **HPUX**: Partial functionality: only systems with a single network interface are supported.
- **NextStep**: Full functionality, provided that the Berkeley Packet Filter option is installed.
- **Rhapsody**: Full functionality.

Installation to these platforms requires little more than a one-line command to most popular package management systems. Since this is such a standard package and incorporated directly into these OS' package management systems, we have chosen to *not bundle it with the workspace-control program* that is also installed on each resource pool node. That way updates to the server can be managed out of band.

ii. Policy management

The ISC DHCP server, like almost all freely available DHCP servers, is loaded with DHCP lease policies via a configuration file (unfortunately not a SQL backend).

The server needs to be restarted to reload its policies. (Because DHCP clients rebroadcast their requests when receiving no answer, this small window of downtime will not affect operations.)

When the *workspace-control* program receives a request to provide networking for a VM it is starting (or when it receives a request to remove a VM), it will call out to a separate program *dhcp-config* via *sudo* that will make the policy adjustment to the ISC DHCP configuration file. That same program will also take the appropriate action to cause it to restart.

By using a separate program with a defined command-line syntax, we:
- ensure that the implementation of the policy adjustment is encapsulated
- allow the action to be accomplished via sudo, ensuring that only the authorized user (in Xen's case, the same user running workspace-control in domain 0) can change the lease policies

The following diagram depicts workspace-control receiving an address assignment from the workspace service. Before starting the VM it adjusts the local DHCP server policy (depicted by the "conf" object in the figure) and restarts the DHCP server so that it will reload its policy. Thus, when the VM boots and requests an address, the specific address will be returned (the DHCP server bases its decision on the MAC address of the requester and we configure it with a one-to-one mapping for MAC->IP (as well as hostname, dns, and default gateway)).
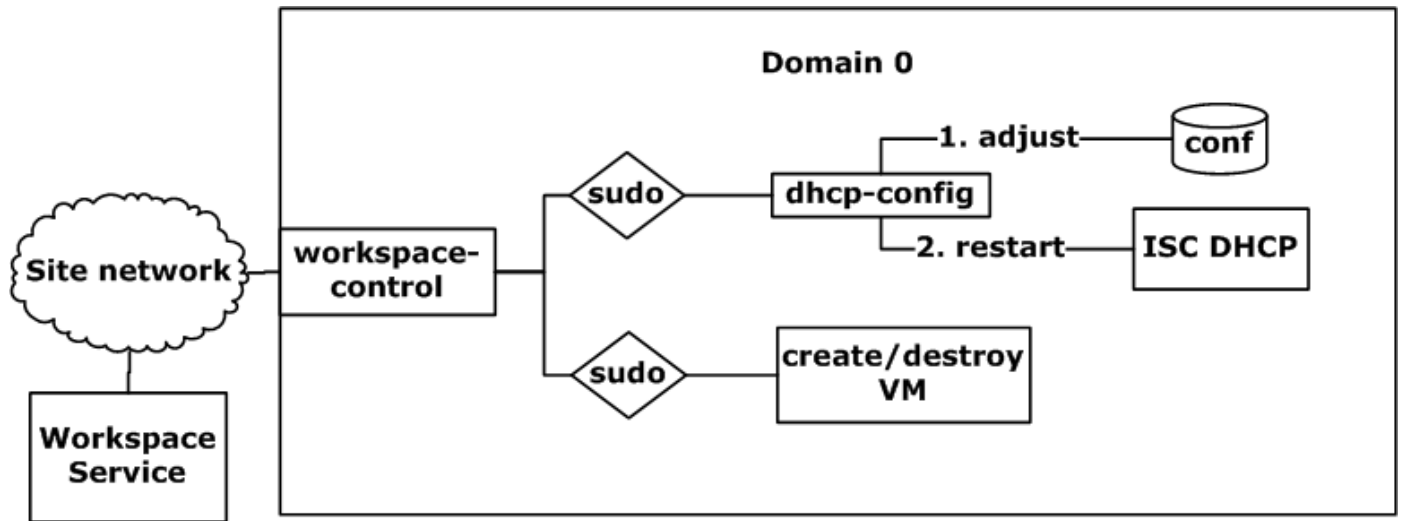
Figure 1. DHCP policy updates

The default *dhcp-config* will be provided with the workspace-control installation.

B. ebtables adjustments

i. interface binding

A normal Xen node with one NIC has this network interface and bridging configuration:
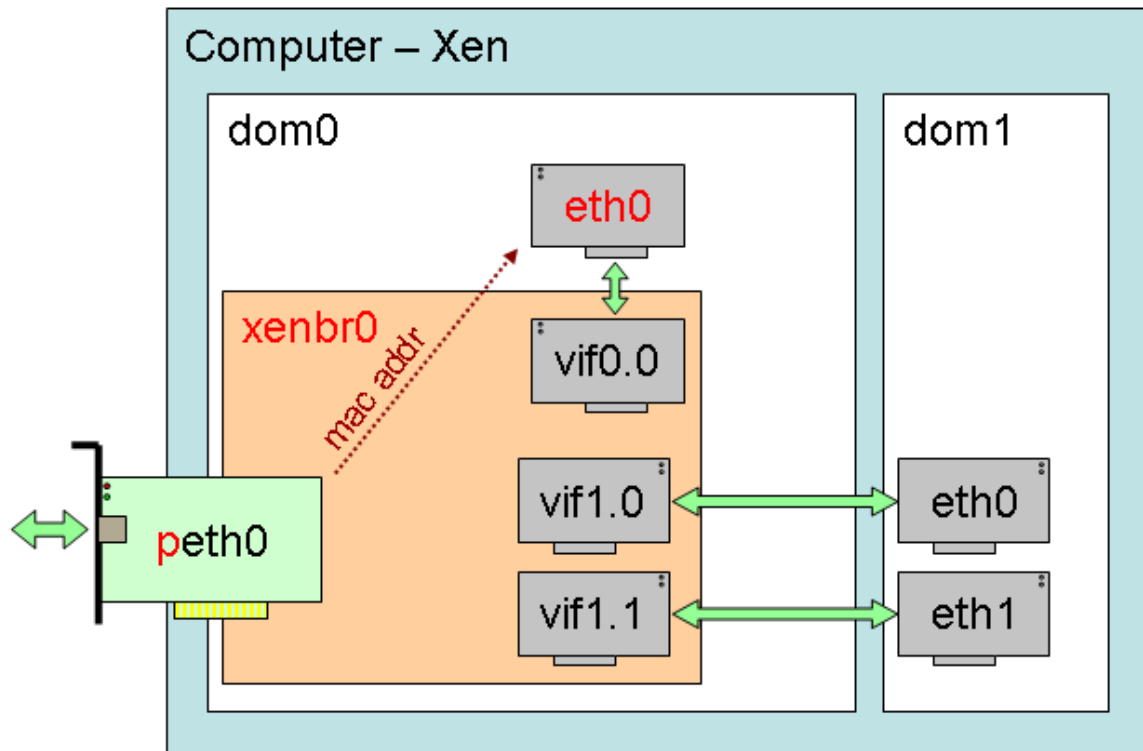

Figure 2. Single NIC Xen networking setup
(figure is courtesy of the XenWiki)

eth0 is the interface that appears to dom0 userspace.

All NICs in this diagram (including *eth0* in dom0) have a virtual interface *vif*.** that is bridged to the actual physical NIC *peth0*.

The ISC DHCP server will be bound to dom0's *eth0* in this example.

ii. ebtables adjustments

Xen is integrated heavily with Linux for dom0 operations and has some hooks for adding and removing *iptables* rules upon VM creation/destruction. Broadcast packets coming from a guest virtual interface (we will use *vif1.0* in this example) are by default copied to

the whole LAN. To avoid this situation, we investigated using iptables but found that *ebtables* offers the most solid solution.

e*btables* is a standard Linux package and the kernel support for it is in a default Xen installation. As with ISC DHCP, installing the userspace tools (*/sbin/ebtables*) is a one line command to the package management system.

To avoid copying the DHCP request to the LAN (or to any other local VMs), we intercept the packet with ebtables before the bridging decision. Any DHCP request is directed to the correct dom0 interface (see note below on advanced configurations). Further, because it is already adding *ebtables* rules, it adds two simple rules for spoofing protection, making it impossible for a NIC to use a different MAC or IP address than assigned.

The inspection flows like so:

1. Is the packet coming from a workspace virtual interface?
2. If not, proceed without further processing.
3. If so, is the MAC address incorrect? Drop the packet.
4. Is this is a DHCP packet?
5. If so, allow it to be bridged only to the appropriate interface for the bridge that it is on. (see note below on advanced configurations). No other interface on the bridge will see the broadcast packet.
6. If not a DHCP packet, it must have the correct source IP address, otherwise the packet is dropped.

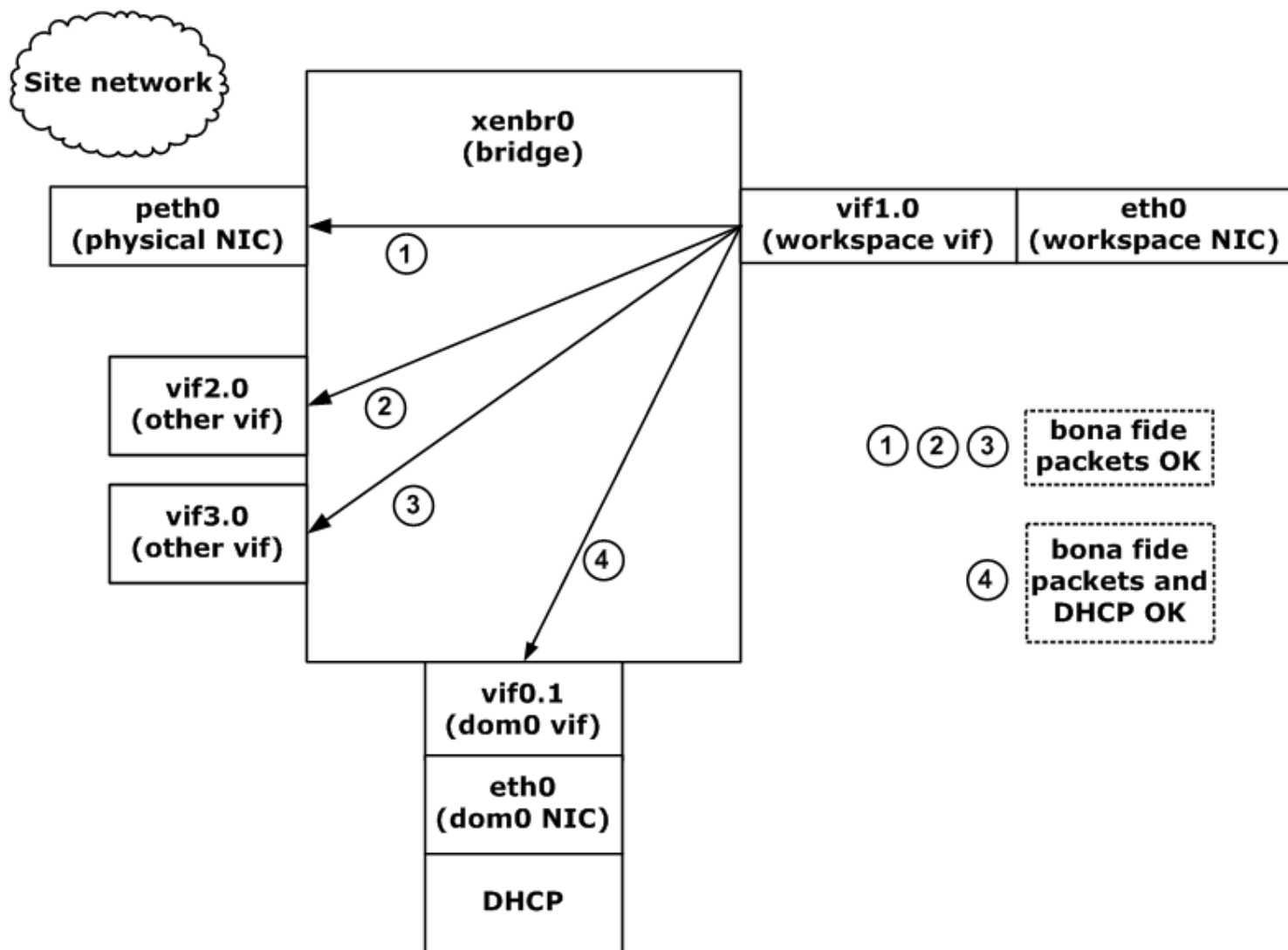The resulting situation is shown in the figure 3:

Figure 3. ebtables networking setup

Note that the workspace can by default still send packets to any interface on the bridge or LAN, it just cannot send DHCP requests to anywhere except *vif0.1* in the example and it cannot send packets without the assigned MAC and IP for source addresses.

iii. Policy management

A new script *ebtables-config* is included with *workspace-control*. This script is called via *dhcp-config* after *dhcp-config* invoked via *sudo*.

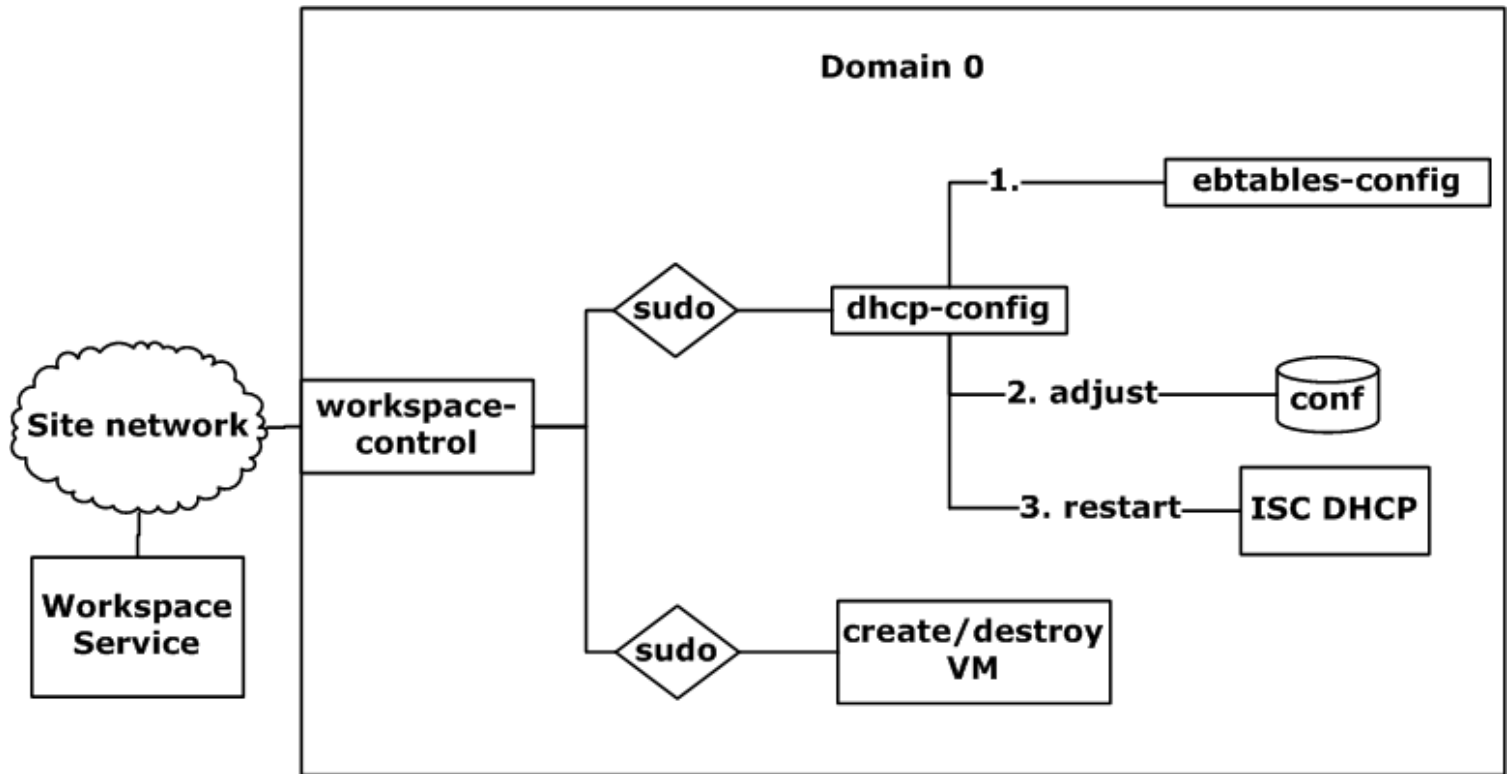The situation in Figure 1 above is therefore extended to look like so:

Figure 4. ebtables policy adjustment

In the future we may add an advanced configuration to allow for generic *ebtables* or *iptables* adjustments that are not tied to DHCP configuration. For example, we include the anti-spoofing rules currently but those and many other network rules could be added regardless of the workspace's use of DHCP. For example, we are considering designs for connectivity rules (such as "workspace A can only send packets to workspace B,C,D, and the internet).