Network Working Group
Internet-Draft
Intended status: Informational
Expires: February 12, 2012

W. Kumari
Google
J. Halpern
Ericsson

August 11, 2011

Virtual Machine mobility in L3 Networks.
draft-wkumari-dcops-l3-vmmobility-00

Abstract

This document outlines how Virtual Machine mobility can be accomplished in datacenter networks that are based on L3 technologies. It is not really intended to solve (or fully define) the problem, but rather to outline it at a very high level to determine if standardization within the IETF makes sense.

Status of this Memo

Copyright Notice

Table of Contents

1. Author Notes

[RFC Editor -- Please remove this section before publication!]

1. Fix terminology section!
2. Rejigger Introduction into Intro and Background!
3. Do we need to extend the mapping service to include (Customer_ID)? This will allow the use of overlapping addresses by customers, but *does* limit the encapsulating technologies.
4. Currently I'm envisioning this as IP only. It would be fairly trivial to make the query in be for the MAC address instead of the IP. This does lead to some interesting issues, like what do we do with broadcast, such as ARP? Have the mapping server reply with all of the destinations and then have the source replicate the packet?!

2. Introduction

There are many ways to design and build a datacenter network (and the definition of what exactly a datacenter network is very vague!), but in general they can be separated into two main classes, Layer-2 based and Layer-3 based.

A Layer-2 based datacenter is one in which the majority of the traffic is bridged (or switched) in a large, flat Layer-2 domain or number of Layer-2 domains. VLANs are often employed to provide customer isolation.

A Layer-3 based datacenter is one in which much of the communication between hosts is switched. In this architecture there are a large number of separate Layer-3 domains (for example, one subnet per rack) and communication between hosts is usually routed. Communication between hosts in the same subnet is (obviously) bridged / switched. While customer isolation can be provided though careful layout and access control lists, in general this architecture is better suited to a single (or small number) of users, such as a single organization.

This delineation is obviously a huge simplification as the design and build out of a datacenter has many dimensions and most real-world datacenters have properties of both Layer-2 and Layer-3.

Virtual Machines are fast gaining popularity as they allow a datacenter operator to more fully leverage their hardware resources and, in essence provide statistical multiplexing of compute resources. By selling multiple VMs on a single physical machine they can maximise their investment, quickly allocate resources to

customers and potentially move VMs to other hosts when needed.

One of the factors driving the design of datacenters is the desire to provide Virtual Machine Mobility. This allows an operator to move the guest machine state from one machine to another, including all of the network state, including keeping TCP connections alive. This allows a datacenter operator to dynamically move guest machines around to better allocate resources and take devices offline for maintain without negatively impacting customers. VM Mobility can even be used to move running machine around to provide better latency - for example an instance can be moved from the East Coast of the USA to Australia and back on a daily basis to "follow the sun".

In many cases VM Mobility requires that the source and destination host machines are on the same layer-2 networks, which has lead to the formation of large Layer-2 networks containing thousands (or tens of thousands) of machines. This has led to some scaling concerns, such as those being addressed in the ARMD Working Group. Some operators are more comfortable running Layer-3 networks (and, to be honest think that big Layer-2 networks are bad JuJu.)

This document outlines how VM Mobility can be designed to work in a datacenter (or across datacenters) that are broken up into multiple Layer-3 domains.

2.1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Terminology

There is a whole industry build around these technologies, and as with many new industries each vendor has their own, unique nomenclature for the various parts. This is the terminology that we are using within this document -- it may not line up with what others call something, but "A rose by any other name..."

Guest network A virtual network connecting guest instances owned by a customer. This is also referred to as a Guest LAN or Customer Network.

Host Machine A machine that "hosts" guest (virtual) machines and runs a Hypervisor. This is usually a powerful server, using software and hardware to provide isolation between the guest machines, often referred to as a Hypervisor. The host machine emulates all of the functions of a "normal" machine so that, ideally, the guest OS is unaware that it is not running on

dedicated hardware.

Gateway A device that provides access to external networks. It provides services to

Guest Machine A "Virtual Machine" that run on a Host Machine.

Hypervisor A somewhat loose them that encompasses the hardware and software that provides isolation between guest machines and emulates all of the functions of bare metal server. This usually includes such things as a virtual Network Interface Card (NIC), a virtual CPU (usually assisted by specialized hardware in the host machine's CPU), virtual memory, etc.

Mapping Service A service providing a mapping between guest machines and host machines on which those guests are running. This mapping service also provides mappings to Gateways that provide connectivity to devices outside the customer networks.

Virtual Machine A synonym for Guest Machine

Virtual Switch A vitualized bridge created by the Hypervisor, bridging the virtual NICs in the virtual machines and providing access to the physical network.

4. Overview

By providing a "shim" layer within the network stack provided by the Hypervisor (or Guest machine) we can create a virtual L2 network connecting the machines belonging to a customer, even if these machines are in different L3 networks (subnets).

When an application on a virtual machine sends a packet to a receiver on another virtual machine, the operating system on the sending VM needs to resolve the hardware address of the destination IP address (using ARP in IPv4 or Neighbor Discovery / Neighbor Solicitation in IPv6). To do this, it generates an ARP / NS packet and broadcasts / multicasts this. As with all traffic sent by the VM, this is handed to a virtual network card, which is simulated by the hypervisor (yes, some VM technologies provide direct access to hardware, this will be further discussed later). The hypervisor examines the packet to provide access control (and similar) and then discards or munges or sends the packet on the physical network. So far this describes the current operation of VM networking.

In order to provide Layer-2 connectivity between a set of virtual machines that run on host machines in different IP subnets (for example, in a Layer-3 based datacenter (or even owned and operated by different providers) we simply build an overlay network connecting the host machines.

When the VM passes the ARP / NS packet to the virtual NIC, the hypervisor intercepts the packet, records which VM generated the

request and extracts the IP address to be resolved. It then queries a mapping server with the guest VM identifier and requested address to determine the IP address of the host machine that hosts the requested destination VM, the VM identifier on that host, and the virtual MAC assigned to that virtual machine. Once the source guest VM receives this information it caches it, and either encapsulates the original ARP / NS in an encapsulation / tunneling mechanism (similar to GRE) or simply synthesized an response and hands that back to the source VM.

Presumably the source VM initialed resolution of the destination VM because it wanted to send traffic to it, so shortly after the source has resolved the destination it will try and send an data packet to it. Once this data packet reaches the hypervisor on the source host machine, the hypervisor simply encapsulates the packet in a tunneling protocol and ships it over the IP network to the destination. When the packet reaches the destination host machine, the packet is decapsulated, the VM ID is extracted and the packet is passed up to the destination VM. (TODO (WK): We need a tunneling mechanism that has a place to put the VM ID -- find one, extend one or simply define a new one). In many ways much of this is similar to LISP...

As the ability to resolve (and so send traffic to) a given machine requires getting the information from a mapping server, communication between hosts can be easily granted and revoked by the mapping server. It is expected that the mapping server will know which VMs are owned by each customer and will, by default, allow access between only those VMs (and a gateway, see below), but if the operator so chooses it can (but probably shouldn't!) allow access between VMs owned by different customers, etc. In addition, because the mapping server uses both the IP address and VM ID to look up the destination information (and the traffic between VMs is encapsulated), overlapping customer space is seamlessly handled (other than in the pathological case where operators allow the customers to interconnect at L2!).

Obviously just having a bunch of customer machines communication just amongst themselves isn't very useful - the customer will want to reach them externally, they will be serving traffic to / from the Internet, etc. This functionality is provided by gateway machines - these machines decapsualte traffic that is destined to locations outside the virtual network and encapsulate traffic bound for ht destination network, etc.

By encapsulating the packet (for example in a GRE packet) the Hypervisor can provide a virtual, transparent network to the receiver. In order to obtain the necessary information to encapsulate the packet (for example, the IP address of the machine

hosting the receiving VM) the sending Hypervisor queries the Mapping Service. This service maps the tuple of (Customer_ID, Destination Address) to the host machine hosting the instance.

For example, if guest machine GA, owned by customer CA on host machine HX wishes to send a packet to guest machine GB (also owned by customer CA) on host machine HY it would generate an ARP request (or, in IPv6 land, a neighbor solicitation) for GB. The Hypervisor process on HX would intercept the ARP, and query the Mapping Service for (CA, GB) which would reply with the address of HY (Hypervisor also cache this information) The Hypervisor on HX would then encapsulate the ARP request packet in a GRE packet, setting the destination to be HY and sending the packet. When the Hypervisor process on HY receives the packet it would decapsulate the packet and hand it to the guest instance GB. This process is transparent to GA and GB - as far as they are concerned, they are both connected to a single network. While the above might sound like a heavyweight operation, the hypervisor is (in general) already examining all packets to provide a virtualized switch, performing access control functions and similar - performing the mapping functionality and encapsulation / decapsulation is not expected to be expensive.

The Mapping Service contains information about all of the guest machines, which customer they are associated with, and routes to external networks. If a guest machine sends a packet that is destined to an external network (such as a host on the Internet), the mapping server returns the address of a Gateway.

5. IANA Considerations

No action required.

6. Security Considerations

7. Privacy

There

8. Acknowledgements

I would like to thank Google for 20% time.

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Authors' Addresses

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: warren@kumari.net

Joel M. Halpern
Ericsson

Email: joel.halpern@ericsson.com