SAL: Scaling Data Centers Using Smart Address Learning

Alexander Shpiner[1], Isaac Keslassy[1], Carmi Arad[2], Tal Mizrahi[1,2], and Yoram Revah[2]

[1]Technion, {shalex@tx, isaac@ee}.technion.ac.il

[2]Marvell Israel, {carmi, talmi, yoramr}@marvell.com

ABSTRACT

Multi-tenant data centers provide a cost-effective many-server infrastructure for hosting large-scale applications. These data centers can run multiple virtual machines (VMs) for each tenant, and potentially place any of these VMs on any of the servers. Therefore, for inter-VM communication, they also need to provide a VM resolution method that can quickly determine the server location of any VM. Unfortunately, existing methods suffer from a scalability bottleneck in the network load of the address resolution messages and/or in the size of the resolution tables.

In this paper, we propose Smart Address Learning (SAL), a novel approach that expands the scalability of both the network load and the resolution table sizes, making it implementable on faster memory devices. The key property of the approach is to selectively learn the addresses in the resolution tables, by using the fact that the VMs of different tenants do not communicate. We further compare the various resolution methods and analyze the tradeoff between network load and table sizes. We also evaluate our results using real-life trace simulations. Our analysis shows that SAL can reduce both the network load and the resolution table sizes by several orders of magnitude.

1. INTRODUCTION

Multi-tenant data centers provide an increasingly popular solution for hosting large-scale service applications [20, 27]. Their appeal comes from their scalability, since they are increasingly cost-effective as they get larger [15]. To ensure scalability, data center providers run multiple virtual machines (VMs) per data center, and can allocate the VMs of a client application to multiple servers, thus also achieving load balancing, fault tolerance and power saving. For efficient implementation of these features, the network has to support unbounded VM placement and migration such that any VM is able to be assigned to any server. In particular, it must provide resolution of the VM location for inter-VM communication: when a new connection is created between two VMs, the initiating VM needs to retrieve the location of the other VM. The services for the physical location resolution of the logical entities have to be supplied by the data center network infrastructure, e.g. by network probing, by the forwarding tables, or by some level of indirection relying on a central database.

Unfortunately, existing location resolution methods often suffer from scalability issues, especially with the resolution network load and the forwarding table size. This is because the network load of the resolution request broadcast messages increases with the number of VMs [11], while it should be kept low in order to leave bandwidth for the application data communication. Moreover, the forwarding-table entries needed for the ever-increasing number of VMs would not fit anymore the on-chip memory that is needed to allow fast access and update times [14, 21]. These issues get especially acute as data centers grow, and may become critical in future multi-tenant data centers, which are expected to include millions of VMs [15, 20].

Several architectures have been proposed to break this scalability bottleneck by using overlay networks [6, 10–12, 18, 19, 23, 25, 26, 29]. These architectures rely on dividing the data center network into segments of broadcast domains, and use network devices called *edge bridges* to connect between the segments and the network core. The edge bridges are responsible for the address resolution of the VM connections between the segments. Thus, the problem is mitigated for intra-segment VM communication. However, these solutions still do not address the scalability problem of VM communications between different segments. In fact, [9] states that the overlay network may still suffer from a bottleneck at the gateway nodes in processing resolving target stations physical address (MAC or IP) and the overlay edge address within the data center.

In this paper, we propose a new address resolution approach called Smart Address Learning (SAL). SAL enables scaling the data center while keeping both the resolution table sizes and the network load low. To do so, we use the fact that VMs of different tenants do not communicate directly. Thus, the edge-bridge resolution tables only need to learn addresses of the VMs that belong to the tenants they serve. For instance, if an edge bridge serves a local network with VMs of a tenant

i, it only needs to follow the location of the other VMs of tenant i, and can ignore the resolution information of VMs of any tenant $j \neq i$. This selective learning makes the table usage more efficient and increases its hit rate. In addition, SAL decreases the network load, because the VM location updates are only sent to the tables that serve the same tenant, instead of being flooded.

The SAL approach can be easily combined in current data center architectures with any network core routing protocol and it is distributed, scalable and fault-tolerant. We introduce two versions of our approach: *pull* and *push*, which differ by the trigger of address learning.

We further provide an analytical model for the evaluation of the table sizes and the network load under SAL and other resolution methods. In addition, we compare SAL against alternative methods using simulations based on synthetic as well as real-life VM creation, placement and tenancy traces.

To our knowledge, this paper is the first to introduce a model for comparing address resolution methods in data centers, as well as the first to evaluate them using real-life trace simulations.

Our analytical model and simulation results show that SAL can reduce the network load for a given resolution table size by up to four orders of magnitude. It also yields a lower update rate and a higher hit rate in the resolution table, thus potentially enabling implementation of fast on-chip resolution tables even for large multi-tenant data centers.

2. RELATED WORK

Commodity techniques of location resolution in small networks cannot be directly applied to the large-scale data center networks. One such technique, ARP over Ethernet layer-2 infrastructure, limits the scalability of the network due to the high load of the broadcast messages and the large forwarding tables [24, 31]. For instance, [11] states that address resolution traffic constitutes more than 88% of the whole broadcast traffic in the data center networks, and that less than $32,000$ hosts in the same broadcast domain can saturate 100 Mb/s network links with their peak load ARP traffic. Moreover, the broadcast domain is recommended to be limited to several hundreds of nodes. As the network grows, the broadcast messages significantly increase the network load, and the forwarding database tables grow as well, due to a larger number of addresses to learn.

Another commodity technique, the hierarchical IP-based layer-3 addressing, mitigates the advantages of VM migration by limiting it to a specific subnet, because the VM needs to maintain its IP address during its runtime, which can be difficult to do while crossing subnets.

In recent years, several overlay network architectures have been proposed to break this limitations in the data centers [6, 10–12, 18, 19, 23, 25, 26, 29]. In these architectures, the VM packets are encapsulated in (or rewritten with) the overlay network headers. The overlay network header is used to route the packet through the network core, which can be implemented using various routing protocols such us commodity Ethernet, hierarchical IP routing, TRILL, MPLS etc. The encapsulation point, denoted edge bridge, can be for instance the server hypervisor or the top-of-the-rack switch.

Table 1 summarizes the differences between these approaches, and compares them with our suggested SAL approach. As mentioned, these methods still lack of scalability in either network load or table sizes when the number of VMs increases. In addition, the table compares additional desired properties, including distribution, fault-tolerance, and compatibility with the commodity techniques. The methods can be roughly divided into three categories:

Central database — The central database approach is used in VL2 [12] and Portland [26]. The distributed hash table on the aggregation switches, as used in SEATTLE [18], also relates to this category. In this approach each VM location is listed in a unique central consistent database. The edge bridge resolves the location by sending a unicast request message to the consistent directory. Note that the edge bridges also hold a cache table that lists the recently-used resolution entries. The usage of a central address resolution database has several drawbacks. These methods may have scalability problems in large data centers due to high resolution updates and unbalanced requests rate. For instance, [26] states that for maintaining the resolution requests, approximately 70 processing cores are needed, which is beyond the capacity of a single commodity machine. VL2 [12] replicates the database to multiple cached servers. However, this raises consistency and concurrent-replication issues, as well as potential scalability problems with when the update rate is high. Moreover, it requires maintaining additional servers for backing up the data. It is also vulnerable to malicious attacks, which lead to service unavailability if the fabric manager fails to perform address resolution [5]. In addition, SEATTLE [18] presents potential fault-tolerance weakness, because the mapping DBs/switches are not backed up, and in a case of DB/switch failure, all the associated mapping information is lost.

We next examine two distributed approaches:

Pull — The distributed Pull approach does not rely on a consistent database, but on broadcasting resolution request messages over the network and learning the resolution from the reply. The address resolution is pulled on-demand, meaning, at the time the resolution is required at the edge bridge. This approach is used in EtherProxy [11], Diverter [10], SARP [25] and several

Table 1: Comparison of the Resolution Methods.

	Location of the consistent information	Cache inconsistency or miss cost	Total number of entries	Potential hot spot
Central DB [12, 26]	Central DB	Request-reply message to DB	Minimal	*Severe*
DHT-based DB [18]	Distributed Hash Table (DHT)	Request message to DHT and redirect	Minimal	*Moderate*
Pull [10, 11]	None	*Resolution request broadcast (high frequency)*	As low as needed	None
Push [23]	Edge bridge (server or TOR switch)	Resolution request broadcast (less likely to happen)	*Maximal*	None
SAL + Pull	None	*Resolution request broadcast (medium frequency)*	As low as needed	None
SAL + Push	Edge bridge (server or TOR switch)	Resolution request broadcast (less likely to happen)	Medium	None

other architectures. Unfortunately, this broadcasting may evolve into a vast flooding of the data center network core, and therefore cause a prohibitive network load. Note that here as well, the edge bridges may hold a cache table that lists the recently used resolution entries, and attempt to store entries for the active connections. However, these entries may be inconsistent.

Push — The distributed Push approach relies on sending address resolution updates with each location change. The edge bridges learn the VM addresses at each location update, and manages resolution tables at the edge bridges. Thus, it avoids request broadcasting, but requires larger resolution tables. Keeping the location information consistent and close to the VM allows for a faster start-up time of the new connections and a lower network load. For instance, this approach is used in Netlord NL-ARP address learning approach [23]. Netlord replicates the resolution database on every server, and therefore uses a maximal possible number of entries. The consistency exists due to update pushing, i.e. the edge bridge sends an update message upon every change of the VM status that it is responsible of, similarly to the gracious ARP mechanism. Unfortunately, in order to be efficient, this approach requires large tables. Note that if the table capacity is large enough and the update messages always arrive within a negligible time, the push architectures tables are always consistent. Usually, the Push approach is combined with the Pull approach for resolving cases with resolution table inconsistency due to table overflow or resolution packet losses.

In summary, both current centralized and distributed address resolution approaches in the data center have limited scalability when the number of VMs increases.

As mentioned before our suggested approach is based on selective learning of addresses from the incoming resolution request messages. A similar idea is used in the selective ARP learning [8], where an ARP table is configured to learn a pre-configured specific set of IP addresses. However, the selective ARP learning approach uses only a passive filtering, without dynamic adaption to VM re-placement and to tenancy.

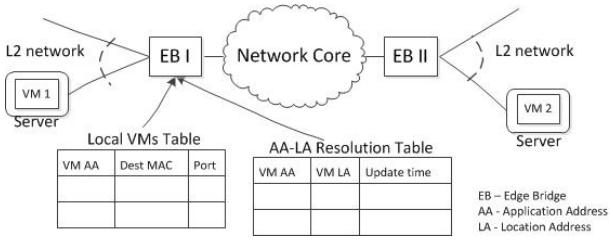


Figure 1: Network Model. The edge bridge (EB) connects the VMs in its L2 network to the other VMs through the data center network core. The EB implements overlay network encapsulation. It uses two tables for the address resolution. The first is a consistent table that lists all the local VMs under the EB, while the second lists the address resolution of the VMs outside the network under the EB. In the paper, we focus on the scalability of the second table.

3. NETWORK MODEL AND ASSUMPTIONS

We begin by defining the network model, as illustrated in Figure 1. The model is fairly standard and follows recent literature [11, 12, 25, 26].

We use the terms *application address* (AA) and *location address* (LA) to define both the addresses in the user VM address space and in the physical data center address space, respectively [12]. Note that both those

address spaces can be assigned in Ethernet layer 2, IP layer 3 or any other proprietary protocol of the data center provider. For example, in VL2 [12] both AA and LA are IP addresses, while in Portland [26] both AA and LA are MAC addresses. The VM AA is combined from a pair of identifiers: the tenant ID within the data center and the VM ID within the tenant, and thus allows easy association of the VM to a tenant. By the term *resolution* we further refer to the translation of the AA address of a VM into its LA address.

We denote as an *edge bridge* (EB) the encapsulation point where the inter-VM data packets are encapsulated in (or rewritten with) the overlay data center network header. In general, the encapsulation point can be either the ToR switch, the aggregation switch, or the server hypervisor. We assume for simplicity that the communication inside the local network under the EB is L2-protocol based, but other methods would hold as well. One advantage of this approach is that the VMs in a network under the EB are interconnected over an L2 network, and do not necessarily need to send internal messages through the EB. Furthermore, broadcast ARP-request messages that are injected by a VM are stopped at the EB and do not propagate to the core network. For the address resolution requests for VMs outside the L2 network, the EB replies using an ARP-reply message with its own MAC address. This common approach is also used by many other overlay network architectures [11, 12, 25, 26].

Our model supports any common network core protocol and topology. The routing between the EBs can be implemented using standard IP routing with ECMP, MPLS or TRILL tunnels, layer-2 Ethernet with VLANs [22], or any other protocol, as long as each EB can communicate with each other EB.

Each edge bridge stores an *LA-to-AA resolution table* and a local *forwarding table*. The LA-to-AA resolution table is used to resolve the destination AA for a given LA. The next section introduces SAL, a novel learning scheme for the resolution entries. In addition, the local forwarding table lists the AAs of all the VMs under the edge bridge layer-2 network together with their layer-2 MAC addresses and the output port towards them. We assume that the placement controller of the data center keeps the forwarding table consistent.

The time-out mechanism is popular in the conventional ARP tables, because the tables are stored in a shared memory space and the timeout mechanism avoids their overflow on the account of other system processes. Hence, the ARP tables intend to store the entries for the active entries only. However, in our model, the EBs use dedicated fast memory to store the resolution entries in order to allow fast access times and high bandwidth. So, there is no cost of storing inconsistent entries. On the other hand, repeatedly acquiring information for the resolution entries that were removed by the time-out mechanism cost in additional network load. Therefore, the resolution tables in our model avoid using the time-out mechanism for the entries. The old, last recently used, inconsistent entries are overwritten, when a new resolution information is required to be written to a full memory.

Finally, in the multi-tenant environment, the VMs are divided into groups of tenants. The VMs of a tenant are assumed to communicate only between themselves, and possibly with hosts outside of the data center, but not with VMs of other tenants. This is logical, since they belong to different applications. It also makes sense for security isolation. Therefore, VMs typically only communicate with a small number of other VMs [7, 16, 18]. We will leverage this assumption in the paper to reduce the amount of information that needs to be stored in the resolution tables. For simplicity, we only focus on internal VM-to-VM communication in this paper, and neglect the communications to hosts outside of the data center.

4. SMART ADDRESS LEARNING (SAL)

4.1 SAL Overview

This section presents our suggested Smart Address Learning (SAL) approach. SAL implements a *distributed resolution database*, in which the resolution tables are stored on the edge bridges (EBs).

In SAL, the EB resolution tables only store the addresses of the VMs that belong to the tenants of the VMs hosted in the EB network. For example, consider Figure 2. The servers under EB I host VMs of tenants A and B only. Therefore, the resolution table of EB I only stores the addresses of VMs of the tenants A and B. Likewise, the servers under EB III host VMs of tenant B only, hence the resolution table of EB III would only store VM addresses for tenant B.

More specifically, any EB that broadcasts an address resolution request message will include the AA and LA of its requesting VM. Upon receiving the message, the other EBs will selectively learn this AA-to-LA mapping in their resolution table *if and only if* their network contains another VM of the same tenant as the requesting VM. Therefore, EBs without VMs of this tenant can disregard this message, and as a result their resolution can typically be smaller than without this selective learning.

4.2 Resolution Table Update

This section presents how our suggested SAL algorithm updates the EB resolution tables following a VM *location update*, i.e. following a VM creation, destruction or migration. We consider two variants of the update method: *pull* and *push*.

In the *pull* version, the location information is pulled

by the EB when this information is required by the encapsulation process, and is not available in its resolution table. On the other hand, in the *push* variant, the location updates are immediately propagated to other forwarding databases on selected EBs.

Intuitively, the *pull* version is preferable when the location update rate is high relative to the address resolution request rate, and when pushing the updates through broadcasting is costly. We further analyze the tradeoffs involved in the next sections.

4.2.1 Pull Update (On-Demand Update)

In the *pull* variant, the location information is pulled to the EB resolution table at the time of resolution request if the information is unavailable in the table. The update is done by broadcasting an address resolution request message to all the other EBs. This message also contains the AA and LA of the source VM that requests the connection. In SAL, the smart learning ensures that other EBs that receive this message only insert this LA address in their tables if they host VM of the same tenant.

Figure 2 illustrates the *pull* variant. It shows how EB I requests information on VM A.4 by broadcasting a request message, thus *pulling* information from the network. It further emphasizes how only EBs with VMs from tenant will add information on VM A.1, while other EBs such as EB III will not. This is the core selection principle behind the SAL algorithm.

Note that if a VM is migrated during an active connection, its resolution update can be pushed immediately in order to avoid a communication disruption by the migration process. In addition, due to the inconsistent information in the resolution tables, it may happen that an EB receives a data message that is destined to the VM that was previously hosted in its network, but already migrated from it. Then the EB answers the source EB with an error message, and the source EB will re-initiate the full address resolutions process. Incidentally, an optional alternative implementation for the EB is to redirect the packets to the EB of the updated VM location, and then ask it to inform back the source EB of the new location.

For simplicity, we assume that each use and update of an entry in the table refreshes its update timestamp. When the table fills up, the oldest entry is cleared from the table.

4.2.2 Push Update (On-Change Update)

In the *push* variant, the updates are pushed to the resolution tables. In our suggested SAL algorithm, in order to reduce network load, the location update broadcast is replaced with a selective multicast to selected EBs only.

Specifically, upon VM location change, the update is propagated (pushed) immediately through a multicast message to the EBs that host VMs of the same tenant of the VM. The multicast destination EBs are known to the sending EB, because it holds the address resolution of all the tenant VMs in its address resolution table. When an update needs to be sent, the EB selects from the forwarding database the location addresses of all the VMs of the tenant whose VM is updated. An easy and fast selection can be achieved by assigning application addresses (AAs) that contain the tenant ID in the specific bits, or even better, by logically organizing the table as a tree with a single node per AA, pointing to the different VMs.

An alternative implementation is to send the location update message from a data center controller that decides on the placement of the VMs. This controller manages the VM placement and thus has a consistent VM location information.

Figure 3 depicts the *push* process. It shows how EB I *pushes* information on newly-created VM A.5 of tenant A, by selectively multicasting an update message only to the relevant EBs that contain VMs from the same tenant A. Thus, the network load is typically less than in a full broadcast message.

Special treatment is required in the following two cases. First, when a VM is assigned to an EB network where no other VM of the same tenant exists, the EB needs to retrieve the location information of all other VMs of the tenant. This can be done by broadcasting a request message to all other EBs, or with the assistance of the data center placement controller.

In a second special case, the last VM of a tenant in an EB is removed due to deletion or migration to other EBs. In this case, the EB can remove all the location entries of all other VMs of this tenant in other EBs. This can be done easily by the EB itself, by checking the number of remaining VMs of the tenant in its resolution table after removing a VM.

In order to preserve consistency of the updates, a timestamping mechanism is used. An update message holds a timestamp of the update time. Before the table update, the EB validates that the received message timestamp is newer than the last time the entry was updated. Each table entry update refreshes its recently used timestamp. When the table fills up, the oldest entry is cleared from the table.

Inconsistency of the information in the tables may still occur with the push variant. It can happen if the message arrival fails, or if the table is filled up. To overcome this inconsistency, the pull update mechanism is still preserved in the push variant. If an entry that was previously removed is required for a resolution, a resolution request is broadcast to all the other EBs.

5. ANALYTICAL MODEL

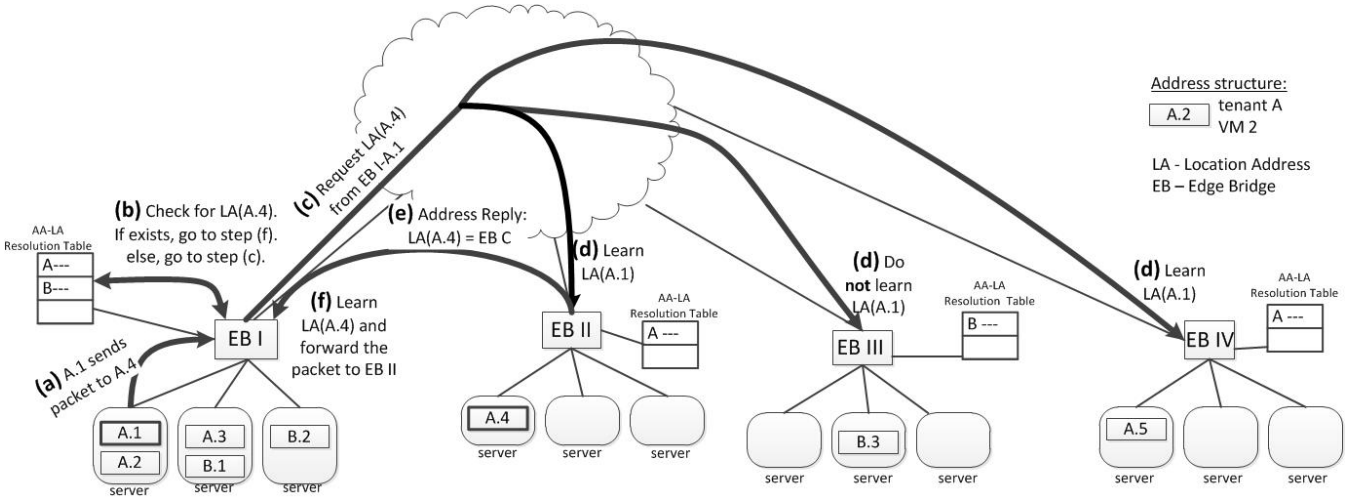


Figure 2: ***Pull*** **variant of SAL. VM** $A.1$ **initializes a connection with VM** $A.4$ **and the resolution process starts. (a) A.1 transmits a data packet destined to A.4. Both VMs belong to tenant A. The packet arrives to EB I, which needs to encapsulate it with the LA of A.4. (b) The LA of A.4 is checked in the resolution table. If it is absent, (c) the EB creates an address resolution message and broadcasts it to other EBs in the network. The address resolution message contains the LA information of the source A.1. (d) Upon reception of this address resolution message, each EB that serves VMs of tenant A learns or updates the LA of A.1. Other EBs do not learn the address in their tables. (e) In addition, EB II that serves the destination A.4, replies by unicast message to EB I with the LA of A.4. (f) Finally, EB I inserts the LA of A.4 in the resolution table and forwards the data packet from A.1 to A.4.**

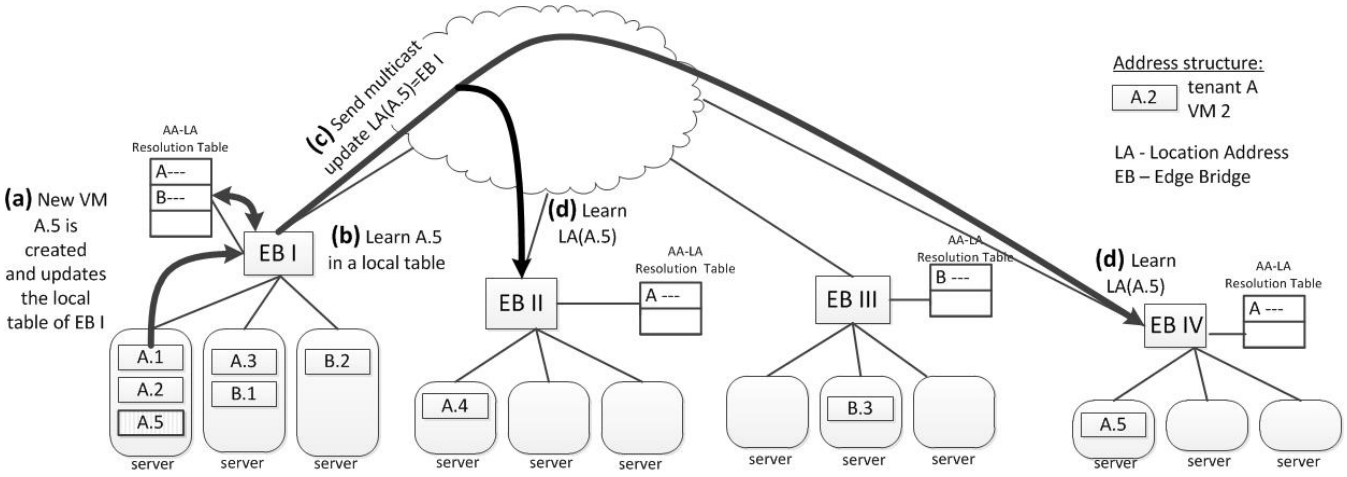


Figure 3: ***Push*** **variant of SAL. A new VM is created and the resolution update process occurs. (a) The new VM A.5 of tenant A is created in a server under EB I. (b) The VM location is updated in the local EB table. (c) EB I multicasts an address update message with the LA of A.5 to the other EBs that serve VMs of tenant A. EB I determines the EBs to multicast using the entries of tenant A VMs in its resolution table. (d) Finally, the multicast EBs insert the new address in their resolution tables.**

5.1 Notations and Assumptions

We would now like to compare the various approaches by formally analyzing their performance. Unfortunately, the performance of each approach is sensitive to many parameters, such as the data center topology, the placement policy, the number of tenants, the distribution of VMs per tenant, the rate of VM creations, migrations and destructions, the burstiness of the application changes, and so on. As a result, to gain some insight, we are reduced to providing a first model in sig-

Table 2: Analysis Notations

N	# of EBs	128
V	# of VMs per EB	640
T	# of tenants	5000
U	# of VMs per tenant ($\equiv VN/T$)	16
C	Active Connections ($\leq VN(U-1)$)	$1.2 \cdot 10^6$
B	EB resolution table capacity	10^5
λ_c	Total VM creation rate (1/sec)	10
λ_m	Total VM migration rate (1/sec)	1
λ_d	Total VM destruction rate (1/sec)	10
λ_u	Total VM location update rate (1/sec) ($\equiv \lambda_c + \lambda_m + \lambda_d$)	21
λ_s	Total resolution request rate (1/sec)	10^5

nificantly simplified settings. In our analysis we compare the following approaches: Central DB, Push with and without SAL, and Pull with and without SAL. We do not analyze DHT-based DB specifically, since in our model, its network load is similar to the network load in the Central DB for the same set of parameters.

Table 2 illustrates the settings for our model. We make several simplifying assumptions. For instance, we assume fixed numbers of VMs per tenant, fixed table capacity at each EB, as well as fixed rates of various VM location update events and VM resolution requests, each following exponentially-distributed inter-event times. We further assume links with infinite capacity and zero propagation time.

In addition, we consider two simple VM placement strategies: *packed* and *round-robin*, similarly to [6]. These two placement strategies are two extremes that typically cause the best- and worst-performance cases. The *best case* typically corresponds to the *packed placement*, in which VMs of a tenant are locally packed under the lowest number of EBs as possible. This placement is typically chosen to minimize the network load. On the other hand, the *worst case* typically corresponds to the *round-robin placement*, in which VMs of a tenant are spread equally among the servers. This placement strategy may be chosen for its fault-tolerance properties.

For each of the placements we impose an additional condition. We distinguish two cases for each of the placements.
For the packed placement: *Case 1a:* If the number of VMs per tenant is small enough to be placed under a single EB ($U \leq V$ or $N \leq T$), no intra-tenant VM communication is passed through EBs.
Case 1b: Otherwise, ($U > V$ or $N > T$), each tenant occupies several EBs.
For the round-robin placement: *Case 2a:* For the *round-robin* placement, if the number of VMs per tenant is smaller than the number of EBs ($U \leq N$ or $V \leq T$), there are no two VMs of any tenant under the same EB.
Case 2b: Otherwise, if the number of VMs per tenant is larger than the number of EBs ($U > N$ or $V > T$), there are VMs of all T tenants under each EB, and each EB serves $\frac{V}{T}$ VMs of a tenant.

In addition, to quantify our models, as shown in Table 2, we assume some typical values (based on [1,2,4,6,12, 13, 15, 17, 20, 26, 28, 30, 32], as well as private talks to industry engineers).

5.2 Resolution Table Length

In this section we evaluate for each of the compared architectures *the resolution table length* (or occupancy), i.e. the required number of resolution entries in the consistent resolution database tables for minimal resolution network load. Specifically, in the *pull* variant, the table length in an EB is the number of resolution table entries needed to support the active connections outgoing from the EB. In the *push* variant, the table length in an EB is the number of VM addresses that are stored in the resolution table. In other words, for a general push architecture it is simply the number of VMs in the data center, while for the push architecture with SAL it is the number of VMs of the tenants that have some VMs under the EB. The local VMs of the EB are not counted. For the central DB, it is simply the number of VMs in the data center.

5.2.1 Central DB

The Central DB architecture maintains a central resolution data base that provides resolution for all the VMs, thus the final resolution table length is simply the number of VMs in the data center, which is VN.

5.2.2 Push

In the general push architecture each EB table stores the resolution entries for all the VMs in the data center, except the local VMs under EB, thus the resolution table length is $V(N-1)$.

In the push architecture with SAL, the number of resolution entries depends on the placement of VMs, as discussed in Section 5.1.

For the *packed* placement, in *Case 1a*, no intra-tenant VM communication is passed through EBs, thus the resolution tables are empty. Otherwise, in *Case 1b*, each resolution table stores entries for one tenant only, of all VMs of a tenant besides the ones that are located under the EB, i.e. a total of $U - V = \frac{V(N-T)}{T}$ entries.

For the *round-robin* placement, in *Case 2a*, there are no two VMs of any tenant under the same EB, thus each resolution table needs to store entries for all other U VMs of a tenant, for each of its V VMs, except for a single local VM, i.e. a total of $V(U-1)$ entries. Otherwise, in *Case 2b*, there are VMs of all T tenants under each EB, and each EB serves $\frac{V}{T}$ VMs of a tenant. Thus, the

resolution table, for each of the tenants, stores entries of VMs under other EBs, i.e. total of $T(U-\frac{V}{T}) = V(N-1)$ entries.

5.2.3 Pull

In the Pull and SAL-Pull architectures the consistency in the resolution tables is kept only for the entries used in the active connections C. Therefore, each EB resolution table stores consistent entries for the connection between its VMs and VMs under other EBs.

Each tenant has an average of $\frac{C}{T}$ connections between its VMs, out of the $U(U-1)$ possible connections between its U VMs. Therefore, given a pair of VMs, the probability that there is a connection between them is $P_{connect} = \frac{C/T}{U(U-1)} \approx \frac{CT}{V^2N^2}$.

Next, for the evaluation, the previously defined types of placements are considered.

For the *packed* placement, we again observe the two cases. In *Case 1a*, there is no connection between the EBs, thus the resolution tables are empty. Otherwise, in *Case 1b*, the V VMs under EB communicating with other $U-V$ VMs of the tenant outside the EB. Thus, the possible number of VMs to connect to outside of the EB is $V(U-V)$. The probability that an resolution entry for any VM X outside the EB is needed, is the probability that exists any VM from the V VMs under EB that connecting with this VM X. It is equal to $1-(1-P_{connect})^V$. Then, the number of resolution entries in the EB is the product of the number of all resolution entries for the tenant $U-V$ by the probability that this entry is needed: $(U-V)(1-(1-P_{connect})^V) = (U-V)(1-(1-\frac{CT}{V^2N^2})^V)$.

For the *round-robin* placement, in *Case 2a*, there are $U-1$ potential connections for each for the V VMs under EB. Therefore the number of active connections through the EB is $\frac{CT}{V^2N^2} \cdot V(U-1) \approx \frac{C}{N}$. Otherwise, in *Case 2b*, each of the T tenants has $\frac{V}{T}$ VMs under the EB. For each tenant, the possible number of VMs to connect to outside of the EB is $U-\frac{V}{T}$, and the probability that the entry for VM is needed is $1-(1-P_{connect})^{\frac{V}{T}}$. Thus, the number of connections out of each EB is $T(U-\frac{V}{T})(1-(1-P_{connect})^{\frac{V}{T}}) = T(U-\frac{V}{T})(1-(1-\frac{CT}{V^2N^2})^{\frac{V}{T}}) = V(N-1)(1-(1-\frac{CT}{V^2N^2})^{\frac{V}{T}})$.

5.2.4 Summary

Table 3 summarizes the expression for the resolution table length in each of the compared methods. In addition, the numerical estimations are based on the values in Table 2.

Figure 4 plots the resolution table lengths as a function of the number N of EBs, based on Table 3. Figures 4(a) and 4(b) show the resolution table length as a function of the number of EBs for the packed and for the round-robin placement, respectively. The values for SAL and for Pull in Figure 4(a) are equal to 0 for $N < T$, therefore they are not seen in the left side of the graphs. Similarly, Figures 4(c) and 4(d) plot the same table lengths, but assuming that the number of tenants T is scaled such that the ratio $\frac{N}{T}$ is kept fixed. The values for SAL and for Pull in Figure 4(a) are equal to 0 for all N, therefore they are not seen in the graphs.

5.3 Network Load

Next we evaluate the network load of the address resolution management packets as a function of VM location updates and address resolution requests rates. The network load is expressed as the rate of address resolution packets. For ease of an evaluation, a single multicast or broadcast packet to k destinations is counted as k packets.

5.3.1 Preliminary Notations

Before we begin with the analysis of the network load, we define several probability notations.

First, we denote the P_{miss} as the probability that the resolution entry is unknown in the table. We assume a uniform probability of each entry to store resolution of any VM. For general Pull or Push method, the P_{miss} is approximated as $P_{\text{miss}} = 1-\min\{1, \frac{B}{VN}\}$, since B is the table length and VN is the total number of VMs to store. Similarly, for SAL-Push approach P_{miss} is approximated as $P_{\text{miss}} = 1-\min\{1, \frac{B}{U\cdot\text{tenants per EB}}\}$, because the table stores entries for VMs of the served tenants only. We consider two types of placement. For the packed placement, the number of tenants per EB is approximated as $\min\{1, \frac{V}{U}\} = \min\{1, \frac{T}{N}\}$ ($P_{\text{miss}} = 1-\min\{1, \frac{B}{U\cdot\min\{1,\frac{V}{U}\}}\}$). For the round-robin placement, it is $\min\{V,T\}$ ($P_{\text{miss}} = 1-\min\{1, \frac{B}{U\cdot\min\{V,T\}}\}$).

We also define P_{wrong} as the probability that the resolution entry in the table is inconsistent. It equals $P_{\text{wrong}} = \frac{\lambda_u}{\lambda_u+\frac{\lambda_s}{N}}$ for Pull. It is equal to 0 for Push, since the entries in Push are consistent.

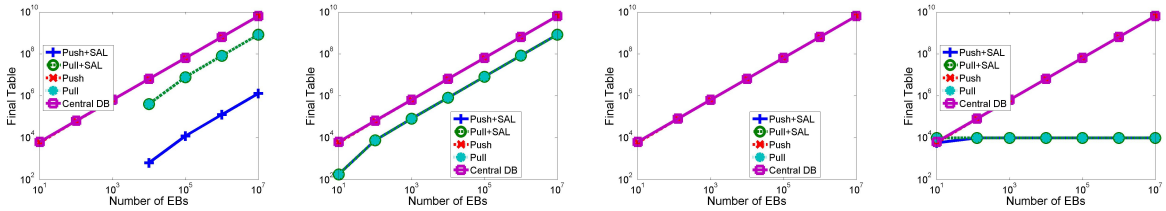
We also define the probability $P_{\text{in other EB}}$ that the resolution is to another EB and it is calculated as follows. In the packed placement, each tenant occupies $\lceil\frac{U}{V}\rceil = \lceil\frac{N}{T}\rceil$ EBs, so the probability to find the resolution destination under another EB is the complimentary to a probability of finding the destination under current EB, and is equal to $P_{\text{in other EB}} = 1-\frac{1}{\lceil\frac{U}{V}\rceil}$. In the round-robin placement, each tenant has $\lceil\frac{U}{N}\rceil = \lceil\frac{V}{T}\rceil$ VMs under EB. Therefore, for each tenant there are up to $\lceil\frac{V}{T}\rceil$ VMs in a specific EB out of its all U VMs, so the probability to find the connection destination in another EB is the complimentary to a probability of finding the destination under a specific EB, thus it is equal to $P_{\text{in other EB}} = 1-\min\{1, \frac{\lceil\frac{V}{T}\rceil-1}{U}\}$.

5.3.2 Central DB

In the Central DB architecture each new connection

Table 3: Resolution Table Lengths (entries).

Architecture	Packed Placement	Typical Estimation	Round-robin Placement	Typical Estimation
Central DB	VN	$8.2 \cdot 10^4$	VN	$8.2 \cdot 10^4$
Push	$V(N-1)$	$8.2 \cdot 10^4$	$V(N-1)$	$8.2 \cdot 10^4$
SAL-Push	$\frac{V \max\{0,(N-T)\}}{T}$	0	$V(\min\{U,N\}-1)$	$9.8 \cdot 10^3$
Pull, SAL-Pull	$\max\{0,U-V\}(1-(1-\frac{CT}{V^2N^2})^V)$	0	$\frac{C}{N}$, if $U \leq N$; $V(N-1)(1-(1-\frac{CT}{V^2N^2})^{\frac{V}{T}})$, if $U > N$	$9.8 \cdot 10^3$



(a) Packed Placement. Fixed number of tenants. (b) Round-robin Placement. Fixed number of tenants. (c) Packed Placement. Number of tenants is scaled with N. (d) Round-robin Placement. Number of tenants is scaled with N.

Figure 4: Model. Resolution Table Length as a Function of Number N of EBs.

retrieves the resolution from central DB. Each new connection with unknown resolution requires two messages: one for the request to the DB and one for the reply back. For the wrong inconsistent resolution entry in the EB table, two additional messages are required: one for sending to a wrong destination and another one for error reply.

Therefore the network load for Central DB is:

$$\begin{aligned} &NL_c \approxeq \\ &=2\lambda_s P_{\text{in other EB}}(P_{\text{miss}} + 2(1-P_{\text{miss}})P_{\text{wrong}}). \end{aligned} \quad (1)$$

For the packed placement it is:

$$\begin{aligned} &NL_{c-packed} \approxeq \\ &2\lambda_s(1-\frac{1}{\lceil \frac{U}{V} \rceil}) \cdot (1-\min\{1,\frac{B}{VN}\} \\ &+ 2(\min\{1,\frac{B}{VN}\})\frac{\lambda_u}{\lambda_u + \frac{\lambda_s}{N}}). \end{aligned} \quad (2)$$

and for the round-robin placement it is:

$$\begin{aligned} &NL_{c-roundrobin} \approxeq \\ &2\lambda_s(1-\min\{1,\frac{\lceil \frac{V}{T} \rceil - 1}{U}\}) \cdot (1-\min\{1,\frac{B}{VN}\} \\ &+ 2(\min\{1,\frac{B}{VN}\})\frac{\lambda_u}{\lambda_u + \frac{\lambda_s}{N}}). \end{aligned} \quad (3)$$

In DHT-based DB architecture the address resolution is done by the resolver switches, thus it is similar to Central DB with the exception that the resolution requests are to the resolution switches in which the DHT is located. Therefore the network load of the resolution packets is similar to the load in Central DB architecture.

5.3.3 Push

In the Push architecture each location update involves broadcasting update messages to all other $N-1$ edge bridges. Also, in the absence of the requested entry from the table, the EB needs to broadcast the resolution request to $N-1$ other EBs and receive one reply. Therefore, the network load in the Push architecture is:

$$NL_{push} \approxeq \lambda_u(N-1) + \lambda_s N P_{\text{in other EB}} P_{\text{miss}}. \quad (4)$$

For the packed placement it equals:

$$\begin{aligned} &NL_{push-packed} \approxeq \\ &\approxeq \lambda_u(N-1) + \lambda_s N(1-\frac{1}{\lceil \frac{U}{V} \rceil})(1-\min\{1,\frac{B}{VN}\}). \end{aligned} \quad (5)$$

and for the round-robin placement it equals:

$$\begin{aligned} &NL_{push-roundrobin} \approxeq \lambda_u(N-1) + \\ &+ \lambda_s N(1-\min\{1,\frac{\lceil \frac{V}{T} \rceil - 1}{U}\})(1-\min\{1,\frac{B}{VN}\}). \end{aligned} \quad (6)$$

5.3.4 SAL-Push

SAL-Push variant is similar to general Push with the difference that the update messages are sent only to the selected EBs. In the packed placement, the average number of EBs under which the VMs of a single tenant

VMs are stored is $\lfloor \frac{U}{V} \rfloor$. In *Case 1a* it is equal to 0, and no update messages are required. In *Case 1b* the network load equals:

$$\begin{aligned} &NL_{S-push-packed} \\ &\approxeq \lambda_u(\lfloor \frac{U}{V} \rfloor - 1) + \lambda_s N P_{\text{in other EB}} P_{\text{miss}} = \\ &= \lambda_u(\lfloor \frac{U}{V} \rfloor - 1) + \\ &+ \lambda_s N (1 - \frac{1}{\lceil \frac{U}{V} \rceil})(1 - \min\{1, \frac{B}{U \cdot \min\{1, \frac{V}{U}\}}\}). \end{aligned} \tag{7}$$

In the round-robin placement, in *Case 2a*, each location update requires $U-1$ messages, one for each other VM of a tenant. Therefore, the network load equals:

$$\begin{aligned} &NL_{S-push-roundrobin-Case3} \\ &\approxeq \lambda_u(U-1) + \lambda_s N P_{\text{in other EB}} P_{\text{miss}} = \\ &= \lambda_u(U-1) + \\ &+ \lambda_s N (1 - \min\{1, \frac{\lceil \frac{V}{T} \rceil - 1}{U}\})(1 - \min\{1, \frac{B}{U \cdot \min\{V, T\}}\}). \end{aligned} \tag{8}$$

Otherwise, in *Case 2b*, there is a VM of each tenant on each EB, and location update requires a message to every other EB. Therefore, the network load is:

$$\begin{aligned} &NL_{S-push-roundrobin-Case4} \\ &\approxeq \lambda_u(N-1) + \lambda_s N P_{\text{in other EB}} P_{\text{miss}} = \\ &= \lambda_u(N-1) + \\ &+ \lambda_s N (1 - \min\{1, \frac{\lceil \frac{V}{T} \rceil - 1}{U}\})(1 - \min\{1, \frac{B}{U \cdot \min\{V, T\}}\}). \end{aligned} \tag{9}$$

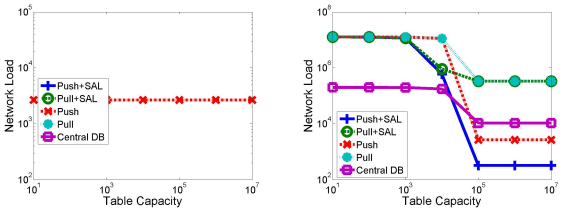
5.3.5 Pull

In Pull architectures the network load consists of the cost of broadcasting resolution request messages to all other EBs. The broadcasting happens when the requested entry is not in the table, or when the entry in the table, but holds the wrong resolution. The last case can happen if the requested VM has moved since its last entry update in the table. The network load for Pull architecture equals:

$$\begin{aligned} &NL_{pull} = NL_{S-pull} \\ &\approxeq \lambda_s N P_{\text{in other EB}}(P_{\text{miss}} + (1 - P_{\text{miss}}) P_{\text{wrong}}). \end{aligned} \tag{10}$$

Using the notations defined in Section 5.3.1, with packed placement, the network load for general Pull equals:

$$\begin{aligned} &NL_{Pull-packed} \approxeq \lambda_s N (1 - \frac{1}{\lceil \frac{U}{V} \rceil}) \cdot \\ &\cdot ((1 - \min\{1, \frac{B}{VN}\}) + (\min\{1, \frac{B}{VN}\}) \frac{\lambda_u}{\lambda_u + \frac{\lambda_s}{N}}). \end{aligned} \tag{11}$$



(a) Packed Placement. (b) Round-robin Placement.

Figure 5: Model. Network Load as a Function of Table Capacity B.

and with round-robin placement, the network load for general Pull equals:

$$\begin{aligned} &NL_{Pull-roundrobin} \approxeq \lambda_s N (1 - \frac{\lceil V/T \rceil - 1}{U}) \cdot \\ &\cdot ((1 - \min\{1, \frac{B}{VN}\}) + \min\{1, \frac{B}{VN}\} \frac{\lambda_u}{\lambda_u + \frac{\lambda_s}{N}}). \end{aligned} \tag{12}$$

5.3.6 SAL-Pull

In continue to Equation 10 the network load of SAL-Pull with packed placement equals:

$$\begin{aligned} &NL_{S-pull-packed} \\ &\lambda_s N (1 - \frac{1}{\lceil \frac{U}{V} \rceil}) \cdot ((1 - \min\{1, \frac{B}{U \cdot \min\{1, \frac{V}{U}\}}\}) + \\ &+ (\min\{1, \frac{B}{U \cdot \min\{1, \frac{V}{U}\}}\}) \frac{\lambda_u}{\lambda_u + \frac{\lambda_s}{N}}). \end{aligned} \tag{13}$$

and for the round-robin placement:

$$\begin{aligned} &NL_{S-Pull-roundrobin} \\ &\lambda_s N (1 - \frac{\lceil V/T \rceil - 1}{U})((1 - \min\{1, \frac{B}{U \cdot \min\{V, T\}}\}) + \\ &+ (\min\{1, \frac{B}{U \cdot \min\{V, T\}}\}) \frac{\lambda_u}{\lambda_u + \frac{\lambda_s}{N}}). \end{aligned} \tag{14}$$

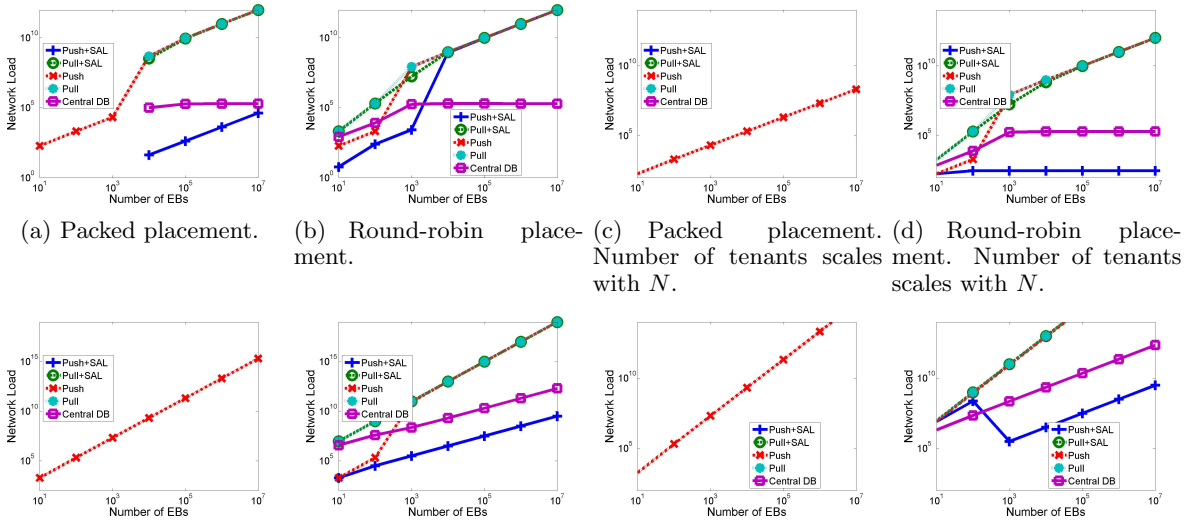
5.3.7 Summary

The network load estimation of the resolution architectures for the packed and round-robin placements is summarized in Table 4. The expressions are next evaluated in Figures 5 and 6.

Figure 5 shows the network load of the resolution packets as a function of the table capacity B in the packed placement and the round-robin placement. The packed placement result is trivial, as resolution packets are sent in the Push architecture only. The round-robin placement result is explained next. We can see that for small table capacities, the hit rate is low in the EBs under all architecture, therefore the Central DB architecture has lower network load, because the resolution requests are sent in unicast and not flooded as in other architectures. In large table capacities, the Push ar-

Table 4: Network Load.

Architecture	Packed Placement	Round-robin Placement
Central DB	$2\lambda_s(1 - \frac{1}{\lceil \frac{U}{V} \rceil})(1 - \min\{1, \frac{B}{VN}\} + 2(\min\{1, \frac{B}{VN}\})\frac{\lambda_u}{\lambda_u + \frac{\lambda_s}{N}})$	$2\lambda_s(1 - \min\{1, \frac{\lceil \frac{V}{T} \rceil - 1}{U}\})(1 - \min\{1, \frac{B}{VN}\} + 2(\min\{1, \frac{B}{VN}\})\frac{\lambda_u}{\lambda_u + \frac{\lambda_s}{N}})$
Push	$\lambda_u(N-1) + \lambda_s N(1 - \frac{1}{\lceil \frac{U}{V} \rceil})(1 - \min\{1, \frac{B}{VN}\})$	$\lambda_u(N-1) + \lambda_s N(1 - \min\{1, \frac{\lceil \frac{V}{T} \rceil - 1}{U}\})(1 - \min\{1, \frac{B}{VN}\})$
SAL-Push	$\lambda_u(\lfloor \frac{U}{V} \rfloor - 1) + \lambda_s(N-1)(1 - \frac{1}{\lceil \frac{U}{V} \rceil})(1 - \min\{1, \frac{B}{U \cdot \min\{U,V\}}\})$	$\lambda_u(\min\{U, N\} - 1) + \lambda_s(N-1)(1 - \min\{1, \frac{\lceil \frac{V}{T} \rceil - 1}{U}\})(1 - \min\{1, \frac{B}{U \cdot \min\{V,T\}}\})$
Pull	$\lambda_s N(1 - \frac{1}{\lceil \frac{U}{V} \rceil})((1 - \min\{1, \frac{B}{VN}\}) + (\min\{1, \frac{B}{VN}\})\frac{\lambda_u}{\lambda_u + \frac{\lambda_s}{N}})$	$\lambda_s N(1 - \frac{\lceil V/T \rceil - 1}{U})((1 - \min\{1, \frac{B}{VN}\}) + \min\{1, \frac{B}{VN}\}\frac{\lambda_u}{\lambda_u + \frac{\lambda_s}{N}})$
SAL-Pull	$\lambda_s N(1 - \frac{1}{\lceil \frac{U}{V} \rceil})((1 - \min\{1, \frac{B}{\min\{U,V\}}\}) + (\min\{1, \frac{B}{U \cdot \min\{U,V\}}\})\frac{\lambda_u}{\lambda_u + \frac{\lambda_s}{N}})$	$\lambda_s N(1 - \frac{\lceil V/T \rceil - 1}{U})((1 - \min\{1, \frac{B}{U \cdot \min\{V,T\}}\}) + (\min\{1, \frac{B}{U \cdot \min\{V,T\}}\})\frac{\lambda_u}{\lambda_u + \frac{\lambda_s}{N}})$



(a) Packed placement. (b) Round-robin placement. (c) Packed placement. Number of tenants scales with N. (d) Round-robin placement. Number of tenants scales with N.

(e) Packed placement. Number of tenants and rates scale with N. (f) Round-robin placement. Number of tenants and rates scale with N. (g) Packed placement. Number of tenants, table capacity and rates scale with N. (h) Round-robin placement. Number of tenants, table capacity and rates scale with N.

Figure 6: Model. Network Load vs Number of EBs N.

chitectures has lower network load than in other architectures, because the tables are large enough to store the resolution for all the VMs, the information is updated instantly and the multiple request broadcasts are avoided. For Pull, the SAL approach improves the network load under middle table capacities. For small table capacities, the difference in the gain of a slightly better hit rate in the tables is negligible considering the miss cost, and for the large table capacities, the tables are large enough. Both in Pull with and without SAL the table hit rate is not bounded by the table capacity.

Figure 6 shows the network load as function of number of EBs N in the packed placement and round-robin placement.

Figures 6(a) and 6(b) present the network load for the packed and round-robin placements, respectively, keeping other parameters fixed. In the packed placement, the network load for $N \leq T$ is equal to 0 in all the architectures besides Push. For large N, Push with SAL and the Central DB outperform the other approaches. For the round-robin placement, the Central DB approach outperforms the distributed approaches. Figures 6(c) and 6(d) present the network load as function of number of EBs (N) for the packed and round-robin placements, respectively, scaling also the number of tenants (T), such that the ratio N/T is kept fixed. Moreover, Figures 6(e) and 6(f) present the network load as function of number of EBs (N) for the packed and round-robin placements, respectively, scaling also the number of tenants (T) and the rates λ_c, λ_m, λ_d and λ_s with N. Finally, Figures 6(g) and 6(h) present the network load as function of number of EBs (N) for the packed and round-robin placements, respectively, scaling also the number of tenants (T), the rates λ_c, λ_m, λ_d and λ_s, and the table capacities (B) with N. With the packed placement, only the Push architecture has a positive network load, since all the VMs of each tenant are served by a single EB. With round-robin placement, Push with SAL outperforms the other approaches.

6. SIMULATIONS

In this section, we describe a set of simulation results evaluating SAL and comparing it to several existing address resolution methods.

6.1 Simulator

We implemented an event-driven simulation of the data center network address resolution system. The simulation includes *VM location update events*, i.e. creations, migrations and destructions, as well as *VM address resolution events*, which are initiated by VMs and request for a resolution of other VMs.

In the simulation, tenants are defined as disjoint sets of virtual machines. The VMs are assigned to the hosting edge bridges independently of the tenant they belong to. Furthermore, the source and destination VMs of each resolution request are chosen uniformly within the VMs of each tenant.

We implemented the following address resolution schemes: *Central DB*, *Push* with and without SAL, and *Pull* with and without SAL. The Pull scheme consists of three variants besides SAL. On the figures they are marked by *Pull (complete)*, *Pull (connection)* and *Pull (conservative)*. The difference between these three Pull schemes is the way in which the EB learns resolution information from the incoming broadcast resolution requests that are not destined to the EB. In *Pull (complete)*, the EBs stores information of each incoming resolution request message. In *Pull (connection)*, only the entries that already exist in the resolution table are updated, but no new entry is learned. This method is similar to ARP. Lastly, in *Pull (conservative)*, the EB does not learn from any resolution request not destined to it.

In all the schemes, the table lengths are limited by a fixed table capacity. When an entry is added to a full table, the oldest entry is overwritten. In addition, an entry with a wrong information is revealed when it is accessed. The wrong entries and the missing entries are resolved by the broadcast resolution request messages to all the servers — except for the Central DB scheme, where the resolution is done by an access to the central directory.

The output of the simulator includes the number of transmitted resolution messages, as well as the occupancy, the number of updates, and the hit percentage of the resolution tables. For simplicity, we neglect the impact of the network topology. Thus, each unicast message between a pair of VMs is counted as a single message, and a multicast or a broadcast message is counted as the number of recipients. For example, a request broadcast by an EB in a data center with N EBs is counted as $N-1$ messages, since it is sent to $N-1$ EBs; and the unicast reply is counted as a single message. Pulling the address resolution data base in the Central DB architecture is counted as two messages: one for the request and one for the reply. Revealing a wrong entry costs two additional messages: one for sending a packet to a wrong destination, and the second for receiving a reply message indicating that the destination is wrong.

6.2 Synthetic Trace Simulation Results

We start by running simulations with a synthetically-generated trace. We use the typical values from Table 2, and vary the table capacity B from 10 to 10^6 entries. The placement distribution is uniform, such that at every placement decision, the edge bridge for each VM is chosen uniformly. New VMs pick uniformly their tenants. The VM chosen for migration or destruction are

also picked uniformly. At the initial state of the simulations, the data center is full with random VMs up to its capacity ($V \cdot N$). The simulations are run until the steady state.

Figures 7(a), 7(b) and 7(c) show the impact of the resolution table capacity on the mean resolution packet network load, the largest mean update rate of a table, and the mean hit rate, respectively, for each of the architectures. Note that for the Central DB, the shown table capacity is for the tables in EBs and not for the central data base.

Specifically, Figure 7(a) confirms our intuition that as table capacity increases, the miss rate decreases and therefore network load decreases, up to a specific large value of table capacity, beyond which there are no further gains. The result also supports our insight from the model that for larger table sizes, the Push architectures perform better than the Pull architectures, and that the SAL approach for both Push and Pull reduces the network load for some ranges of table capacities, while never increasing the network load. It also seems that for low table sizes, the preferred resolution method is the Central DB. However, it also relies on a large memory storage with a central data base that holds the resolution of all the VMs. This large memory is not reflected in this plot.

Figure 7(b) presents the update rate of a single table. By table update, we define each change of the resolution entry in the table, including the address change in an existing entry, and an old entry overwrite for a different VM. For the Central DB architecture, the updates are counted on the central data base, since it suffers a larger update rate than the EB tables. Since the central data base is updated upon each VM location change only, the shown update rate for Central DB is fixed for any table size in the EB.

Figure 7(c) confirms the intuition that the table hit rate increases with the table capacity, and that for the Push and complete Pull architectures, the hit rate is lower than for the other approaches, since in these architectures the resolution tables store information about VMs that are irrelevant.

6.3 Benchmark Trace

Next, we evaluate the system with a benchmark trace from the IBM Research Compute Cloud (RCCv2), where the customer data was anonymized [3]. The extracted events from the trace are the creation and destruction times for various VMs in the data center, their placement, and their tenant assignment. Furthermore, the address resolutions are randomly added with a ratio of 100 resolution events per VM location update event. The RCCv2 system does not include migration, thus the update events only consist of VM creation and destruction events.

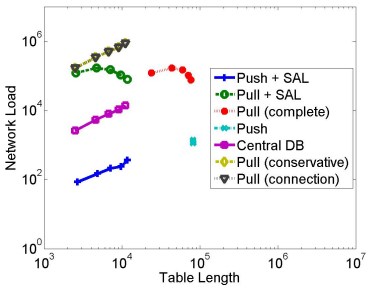


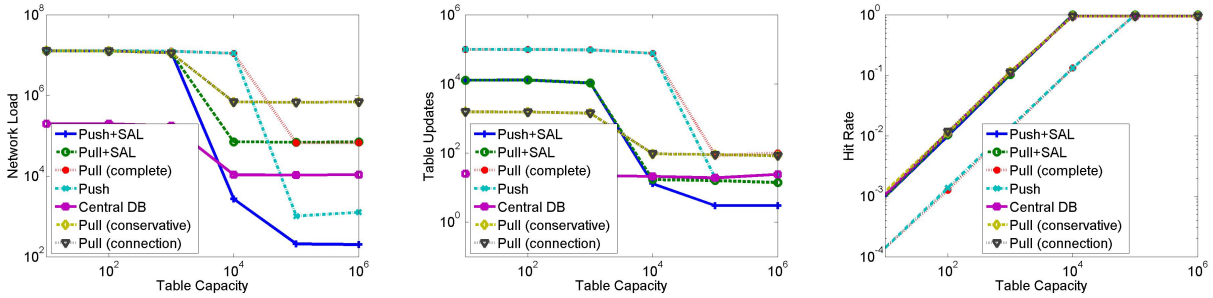
Figure 9: Placement Effect on Network Load and Resolution Table Length. $p = 1$ is the rightmost point for each architecture. $p = 0.2$ is the leftmost point for each architecture. For $p = 0$ the values are equal to 0, thus are not shown.

Figure 8 is analogous to Figure 7 of the synthetic trace simulations. Most algorithms behave similarly. Moreover, since we now use a slightly higher rate of VM location updates compared to the resolution request rate, the Central DB approach presents a lower asymptotic network load than the Push architectures. This is because a higher location update rate requires unnecessary location update messages in the Push architectures, since an update message may be unnecessary in practice when a VM is moved again before its location resolution is requested by the other EBs. Although the Central DB architecture slightly outperforms the Push with SAL approach, it still requires a higher table update rate, as shown in Figure 6.3. Clearly, higher VM update rates have are detrimental for Push architectures, other parameters being equal. Lastly, Figure 8(c) shows that the SAL approach also improves the resolution table hit rate.

6.4 Placement Strategy Effect

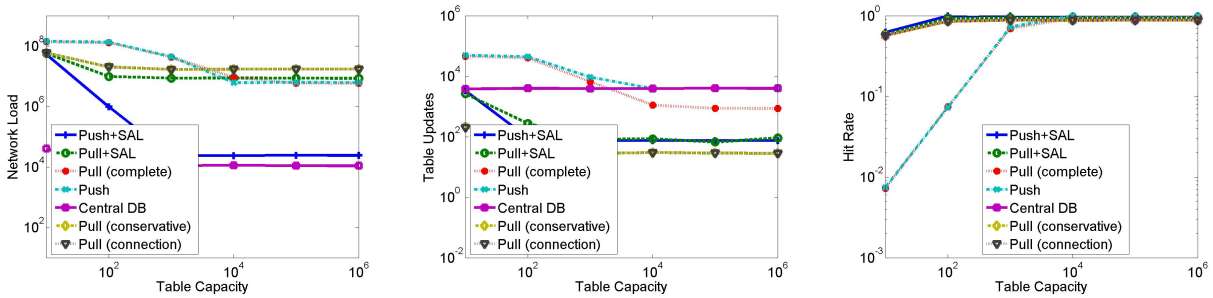
Next we check the effect of the placement strategy on the resolution packets network load and the resolution table length. We already discussed in Section 5.1 the two extreme placement strategies: *packed* and *round-robin*. We simulate *hybrid placement strategies* in which, given a parameter p between 0 and 1, each placement decision picks the *packed* placement strategy with probability p, and the *round-robin* placement strategy with probability $1 - p$. We run simulations with the hybrid placement strategies by varying p from 0 to 1 in steps of 0.2. The resolution table capacities are chosen as infinity large so as to evaluate the resolution table length in an unconstrained manner. Other parameters are chosen based on the values in Table 2.

Figure 9 shows the largest resolution table at the end of the simulation run vs. the cumulative number of resolution packets sent for the synthetic trace. For each type of architecture the results from various hybrid place-



(a) Network Load vs. Resolution Table Capacity. (b) Largest Table Update Rate vs. Resolution Table Capacity. (c) Mean Table Hit Rate vs. Resolution Table Capacity.

Figure 7: Synthetic Event Trace Simulation Results.



(a) Network Load vs. Resolution Table Capacity. (b) Largest Table Update Rate vs. Resolution Table Capacity. (c) Mean Table Hit Rate vs. Resolution Table Capacity.

Figure 8: Benchmark Trace Simulation Results.

ment strategies are connected by a line. The rightmost point for each architecture line is for $p = 1$, and the leftmost point is for $p = 0.2$. For the packed placement ($p = 0$), all the VMs of each of the tenants are packed under a single EB, thus no resolution request is exchanged between the EBs and no updates are pushed in SAL. Therefore, the result for $p = 0$ is omitted, since all the values are equal to 0. The only exception is the Push architecture, in which the updates are still pushed between the EBs and the network load and table sizes are larger than 0. Also, for the Push architecture, all the values (including $p = 0$) are concentrated in the graph, since it is less affected by the placement strategy.

It appears that the relative performance of diverse approaches is relatively insensitive to the placement strategy. Therefore, the main insight is that the impact of the placement strategy is *less significant* than we expectd before running the simulation.

7. CONCLUSIONS

In the paper we proposed Smart Address Learning (SAL), a novel approach that expands the scalability of current address resolution mechanisms in the data centers, for both the network load and the resolution table sizes, which makes it possible to be implemented on faster memory devices. The key property of the approach is to selectively learn the addresses in the resolution tables, based on the fact that the VMs of different tenants do not communicate.

We presented an analytical model of the network load and resolution table sizes for the presented resolution methods. We further used the model and simulations to evaluate the tradeoff of the network load and the resolution table size. Our analysis showed that both the network load and the resolution table sizes can be reduced by orders of magnitude depending on the system parameters.

More generally, to our knowledge, this paper is the first to introduce a model for comparing address resolution methods in data centers, as well as the first to evaluate them using real-life trace simulations. A more advanced analysis of the optimal address resolution tradeoff in data centers is left for future work.

Acknowledgment

The authors would like to thank Orna Agmon Ben-Yehuda and Aran Bergman for their helpful comments, as well as Mariusz Sabath and David Breitgand, IBM WRC, who kindly shared the data of the IBM Research Compute Cloud (RCCv2) traces [3]. This work was partly supported by the Hasso Plattner Institute Research School, the Intel ICRI-CI Center, the Israel Min-

istry of Science and Technology, and European Research Council Starting Grant No. 210389.

8. REFERENCES

[1] Amazon web services LLC. "https://aws.amazon.com".

[2] Microsoft Corporation, an overview of Windows Azure. "http://www.microsoft.com/ downloads/details.aspx?displaylang=en&FamilyID =96d08ded-bbb9-450b-b180-b9d1f04c3b7f.".

[3] G. Ammons et al. RC2: A living lab for cloud computing. *IBM, IBM Research Report RC24947*, 2010.

[4] D. Ármannsson, G. Hjálmtýsson, P. D. Smith, and L. Mathy. Controlling the effects of anomalous arp behaviour on ethernet networks. ACM CoNEXT '05, 2005.

[5] F. Bari, R. Boutaba, R. Esteves, M. Podlesny, G. Rabbani, Q. Zhang, F. Zhani, and L. Granville. Data center network virtualization: A survey. *IEEE Communications Surveys and Tutorials*, 2012.

[6] T. Benson, A. Akella, A. Shaikh, and S. Sahu. CloudNaaS: a cloud networking platform for enterprise applications. SOCC '11, 2011.

[7] P. Bodík, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica. Surviving failures in bandwidth-constrained datacenters. SIGCOMM '12, 2012.

[8] R. Chamarajanagar, P. Hunt, S. Kimble, T. Nguyen, and G. Rashiyamany. Selective passive address resolution learning. US Patent Application 20080144634, 2008.

[9] L. Dunbar, W. Kumari, and I. Gashinsky. Practices for scaling ARP and ND for large data centers. Network Working Group Internet Draft work in progress.

[10] A. Edwards, A. Fischer, and A. Lain. Diverter: a new approach to networking within virtualized infrastructures. ACM WREN '09, 2009.

[11] K. Elmeleegy and A. Cox. Etherproxy: Scaling ethernet by suppressing broadcast traffic. In *IEEE INFOCOM'09*, 2009.

[12] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: a scalable and flexible data center network. ACM SIGCOMM '09, 2009.

[13] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. SecondNet: a data center network virtualization architecture with bandwidth guarantees. ACM Co-NEXT '10, 2010.

[14] G. Hankins. Pushing the limits, a perspective on router architecture challenges. In *North American Network Operators Group, NANOG 53*.

[15] N. Ilyadis. The evolution of next-generation data center networks for high capacity computing. In *VLSI Circuits (VLSIC), 2012*, 2012.

[16] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The nature of data center traffic: measurements and analysis. ACM IMC '09, 2009.

[17] R. Katz. Tech titans building boom. *IEEE Spectrum*, 46(2):40 –54, Feb. 2009.

[18] C. Kim, M. Caesar, and J. Rexford. Floodless in seattle: a scalable ethernet architecture for large enterprises. In *ACM SIGCOMM '08*, 2008.

[19] M. Mahalingam and et al. VXLAN: A framework for overlaying virtualized layer 2 networks over layer 3 networks. In *Network Working Group Internet Draft*, 2011.

[20] J. Metzler, A. Metzler, and et al. The emerging data center LAN. *Webtorials Analyst Division, Cloud Networking Reports 2010 - 2012*.

[21] D. Meyer, L. Zhang, and K. Fall. Report from the IAB workshop on routing and addressing. In *IETF, RFC 4984*, 2007.

[22] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul. SPAIN: COTS data-center Ethernet for multipathing over arbitrary topologies. USENIX NSDI'10, 2010.

[23] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary. NetLord: a scalable multi-tenant network architecture for virtualized datacenters. ACM SIGCOMM '11, 2011.

[24] A. Myers, T. E. Ng, and H. Zhang. Rethinking the service model: Scaling ethernet to a million nodes, 2004.

[25] Y. Nachum, L. Dunbar, I. Yerushalmi, and T. Mizrahi. Scaling the address resolution protocol for large data centers (SARP). INTAREA Working Group Internet Draft (work in progress).

[26] R. Niranjan 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. ACM SIGCOMM '09, 2009.

[27] M. Saluan. Want to Provide Cloud Services? You Need to Understand Multi-Tenancy. `http://mspmentor.net/blog/want-provide-cloud-services-you-need-\understand-multi-tenancy`, 2013.

[28] A. Shieh, S. Kandula, A. Greenberg, and C. Kim. Seawall: performance isolation for cloud datacenter networks. HotCloud'10, 2010.

[29] M. Sridharan and et al. NVGRE: Network virtualization using generic routing encapsulation. In *Network Working Group Internet Draft*, 2011.

[30] B. Stephens, A. Cox, W. Felter, C. Dixon, and J. Carter. PAST: scalable ethernet for data centers. ACM CoNEXT '12, 2012.

[31] B. Stephens, A. L. Cox, S. Rixner, and T. S. E. Ng. A scalability study of enterprise network architectures. ACM/IEEE ANCS '11, 2011.

[32] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 2010.