PLAN: A POLICY-AWARE VM MANAGEMENT SCHEME FOR CLOUD DATA CENTERS

PRESENTATION BY: DANIEL HERNANDEZ

OVERVIEW

- *Introduction*
- Problem Modeling
- PLAN Algorithms
- Evaluation
- Related Works
- Conclusion

INTRODUCTION

- Network configuration and management is often overlooked by research that instead focuses improving resource usage efficiency.
- Network policy and Virtual Machine (VM) management have long been researched independently.
- Most VM management schemes require that VM's be consolidated in order to reduce the number of servers being used. Thus, when a VM is migrated to a different server, network policy still requires traffic to traverse a specified order of middleboxes, such as firewalls, intrusion detection and prevention systems, load balancers.
- Since the network policy has not been updated to reflect that the VM has been migrated, the end-to-end traffic flow path will not be a shortest path.

INTRODUCTION (CONTINUED)

- The affected policy must be updated to reflect VM migrations.
- Deployment of applications in Cloud DC without consideration of network policies may lead to up to 91% policy violations.
- When deciding where to migrate a VM, locations of middleboxes have to be taken into consideration. Failing to do so will result in a sub-optimal network with lower performance and possibly cause service disruption.
- Other existing proposals that aim to dynamically manage network policies can be placed in two categories
  - Virtualization and Consolidation
  - SDN-based policy enforcement

OVERVIEW

- Introduction
- ***Problem Modeling***
- PLAN Algorithms
- Evaluation
- Related Works
- Conclusion

PROBLEM MODELING

- Motivating example
  - A common Data Center (DC) Web service application is used as an example to show that migrating VMs without policy consideration will have unexpected results and application performance degradation
  - Topology and Application: The following figure shows a DC network topology that consists of network switches and several distinct types of middleboxes.

PROBLEM MODELING

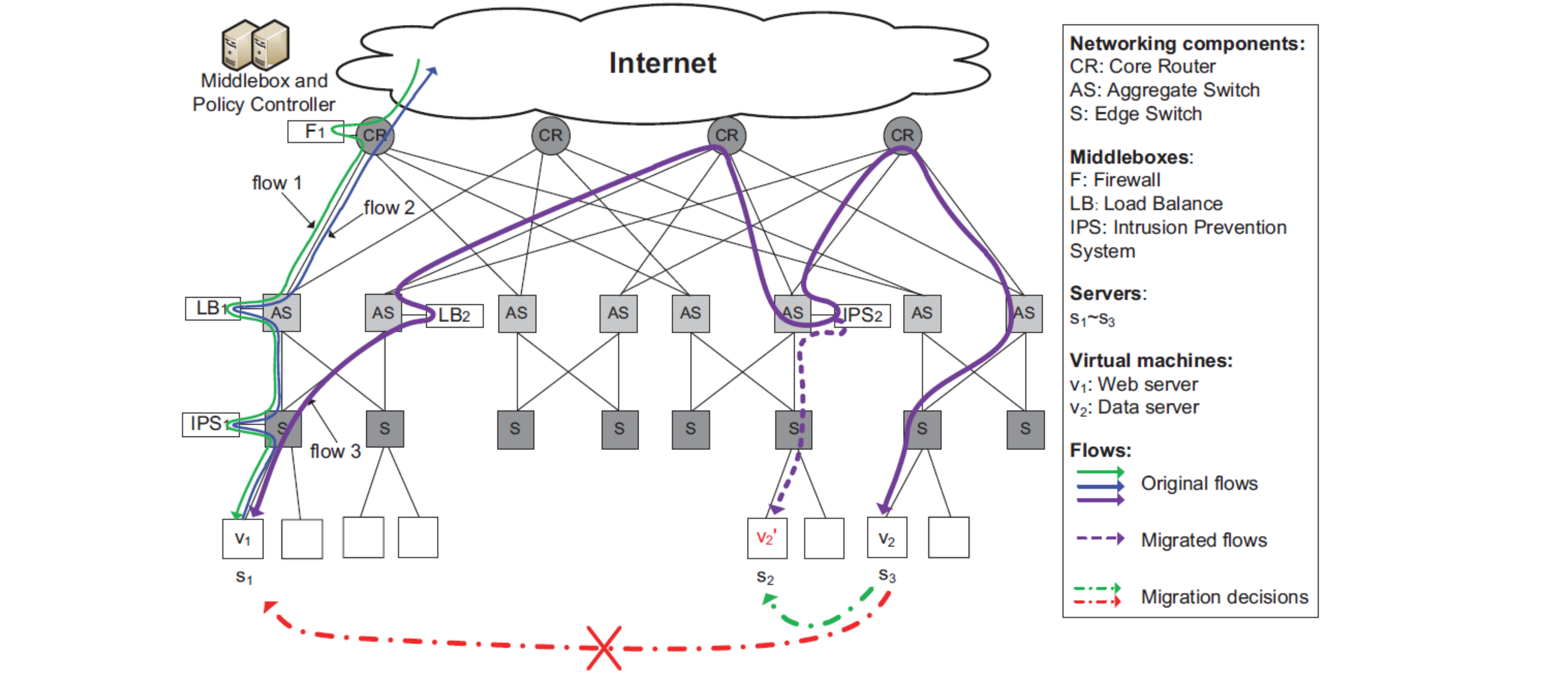


Figure 1: Flows traversing different sequences of middleboxes in DC networks. Without policy-awareness, v_2 will be migrated to s_1, resulting in longer paths for flow 1 and wasting network resources.

PROBLEM MODELING

- In the figure, Firewall F_1 filters unwanted or malicious traffic from the Internet.
- Intrusion Prevention Systems IPS_1 and IPS_2 are configured with a ruleset and monitor the network for malicious activity. It logs such activity and stops/blocks it.
- Load Balancer LB_1 provides a point of entry for the web service. It forwards traffic flow to one or more hosts in the network that provide the service.
- V_1 is a web service that accepts HTTP requests from a client u. After a request is received, V_1 will query a data server V_2, then V_1 performs a computation based on data retrieved from V_2. Then V_1 sends the results to the client.

PROBLEM MODELING

- Policy Configurations: Policies are identified by a 5-tuple and a list of middleboxes.
- The following policies below are configured through the Policy Controller that are used in the motivating example and figure 1 from the previous slides.

- $p_1 = \{u, LB_1, *, 80, HTTP\} \rightarrow \{F_1, LB_1\}$
- $p_2 = \{u, v_1, *, 80, HTTP\} \rightarrow \{IPS_1\}$
- $p_3 = \{v_1, v_2, 1001, 1002, TCP\} \rightarrow \{LB_2, IPS_2\}$
- $p_4 = \{v_2, v_1, 1002, 1001, TCP\} \rightarrow \{IPS_2, LB_2\}$
- $p_5 = \{v_1, u, 80, *, HTTP\} \rightarrow \{IPS_1, LB_1\}$
- $p_6 = \{LB_1, u, 80, *, HTTP\} \rightarrow \{\}$

PROBLEM MODELING

- Policy P_1: Internet client sends HTTP request to public IP of LB_1. Internet traffic must traverse firewall F_1.
- Policy P_2: LB_1 will load balance among different web servers and change the destination of the request to web server V_1. Traffic will traverse IPS_1, which protects the web servers in the DC network.
- Policy P_3 & P_4: Server V_1 will contact a data server to retrieve data. That data server is protected by IPS_2. The response traffic coming from V_2 will need to traverse IPS_2 and LB_2.
- Policy P_5 & P_6: After getting the data from the data server, the web server will send computed results back to the internet client. The response traffic will traverse IPS_1 and LB_1. Then LB_1 forwards the traffic to the internet client. Since the traffic from V_1 is destined for the Internet, it does not need to traverse firewall F_1.

PROBLEM MODELING

- Migration Rule: In order to reduce congestion in the core layers of the DC network, VM management schemes cluster (consolidate) VMs.
- This confines traffic in the lower layers of the network so that as much possible traffic is routed over the edge layer.
- Middleboxes are often collocated so that traffic is kept within the edge layer boundaries.
- In the example and figure 1, we consider the migration of V_2 from S_2.
- Lots of traffic data is exchanged between V_1 and V_2.
- Without policy considerations, V_2 may be migrated to S_1, so that the VMs are close to each other. This will increase the route length of flow 3 in the figure and waste bandwidth.
- Considering policy configurations and traffic patterns in the example and figure, V_2 should be migrated to S_2, to reduce the cost generated between V_2 and IPS_2.

PROBLEM MODELING

- Communication Cost with Policies: Let V be the set of VM's in the DC network, hosted by the set of servers S. $\lambda_k(v_i, v_j)$ denotes the traffic load (rate) between the two VMs following policy p_k, which is in the set of policies P. MB = $\{mb_1, mb_2, \ldots\}$ is a group of middleboxes.
- A Middlebox Controller configures middleboxes and monitors them and can inform switches of addition or failure of a middlebox.
- A Policy Controller can be used by network admins to specify and update policies, and distribute them to the corresponding switches.

PROBLEM MODELING

- Communication cost with Policies (Continued): Each policy p_i in the set of policies P is of the form {flow -> sequence}. Flow is a 5-tuple defined as {$source_{ip}$, $destination_{ip}$, $source_{port}$, $destination_{port}$, protocol}.
- Sequence is defined as a list of middleboxes that all traffic flow that matches policy p_i must traverse in order.
- $P(v_i, v_j)$ is the set of all policies defined for traffic from v_i to v_j. $P(v_i, v_j) = \{p_k \mid p_k.source = v_i, p_k.destination = v_j\}$.
- $L(n_i, n_j)$ is the routing path between nodes n_i and n_j. Link l is an element of $L(n_i, n_j)$ if the link is on the path.
- If a flow from two VMs matches a policy, the routing path is:

$$\begin{aligned} L_k(v_i, v_j) = \ & L(v_i, p_k^{in}) \\ & + \sum_{mb_s^k \neq p_k^{out}} L(mb_s^k, mb_{s+1}^k) \\ & + L(p_k^{out}, v_j) \end{aligned}$$

PROBLEM MODELING

- Not all DC links are equal. Utilization of lower cost switches are preferable to the more expensive router links. This keeps investment cost low for providers.
- The Communication Cost of all traffic from VM v_i to v_j is shown below:

$$
\begin{aligned}
C(v_i, v_j) &= \sum_{p_k \in P(v_i, v_j)} \lambda_k(v_i, v_j) \sum_{l_s \in L_k(v_i, v_j)} c_s \\
&= \sum_{p_k \in P(v_i, v_j)} (C_k(v_i, p_k^{in}) + C_k(p_k^{in}, p_k^{out}) \\
&\quad + C_k(p_k^{out}, v_j))
\end{aligned}
$$

PROBLEM MODELING

- Policy-Aware VM Allocation Problem: Each server is connected to an edge switch, and each edge switch can retrieve a global graph of all middleboxes from the Policy Controller.
- In order to preserve policy requirements, the acceptable servers that a VM vi can migrate to are: $S(v_i) = \bigcap_{mb_k \in MB^{in}(v_i) \cup MB^{out}(v_i)} S(mb_k)$
- $S(v_i)$ is all servers that can be reached by v_i, so these are possible destinations where v_i can be migrated to.
- The vector R_i denotes the physical resource requirements of VM v_i

PROBLEM MODELING

- The amount of physical resource provisioning by host server s_i is given by vector H_i.
- A is an allocation of all VMs. $A(v_i)$ is the server which hosts v_i in A, and $A(s_i)$ is the set of VMs hosted by s_i.
- Considering a migration for VM vi from is current allocated server to another server, the feasible space of candidate servers for vi is characterized by:

$$\mathcal{S}_i = \{\hat{s} | (\sum_{v_k \in A(\hat{s})} R_k + R_i) \preceq H_j, \hat{s} \in S(v_i)\}$$

PROBLEM MODELING

- Let $C_i(s_j)$ be the total communication cost induced by v_i between s_j and $MB^{in}(v_i)$ U $MB^{out}(V_i)$.

$$\mathcal{C}_i(s_j) = \sum_{p_k \in P(v_i,*)} C_k(v_i, p_k^{in}) + \sum_{p_k \in P(*,v_i)} C_k(v_i, p_k^{out})$$

- Migrating a VM generates network traffic between the source and destination hosts because it involves copying the in-memory state and the content of the CPU registers between the hypervisors.
- There are three phases to pre-copy: pre-copy phase, pre-copy termination phase, and stop-and-copy phase.
- The estimated migration cost is:

$$C_m(v_i) = M \cdot \frac{1-(R/L)^{n+1}}{1-(R/L)}$$

$$\text{where } n = \min(\lceil \log_{R/L} \frac{T \cdot L}{M} \rceil, \lceil \log_{R/L} \frac{X \cdot R}{M \cdot (L-R)} \rceil)$$

PROBLEM MODELING

- The utility of a migration is defined as: $U(A(v_i) \to \hat{s}) = \mathcal{C}_i(A(v_i)) - \mathcal{C}_i(\hat{s}) - C_m(v_i)$
- The utility is 0 if no migration takes place.
- The total utility is the summation of utilities for all migrated VMs.
- The Policy-Aware VM Management (PLAN) problem is defined below:

Definition 1. *Given the set of VMs $\mathbb{V}$, servers $\mathbb{S}$, policies $\mathbb{P}$, and an initial allocation A, we need to find a new allocation $\hat{A}$ that maximizes the total utility:*

$$\begin{aligned} &\max \mathcal{U}_{A \to \hat{A}} \\ &s.t.\ \mathcal{U}_{A \to \hat{A}} > 0 \\ &\hat{A}(v_i) \in S_i, \forall v_i \in \mathbb{V} \end{aligned} \tag{8}$$

PROBLEM MODELING

The following theorem proves that the PLAN problem is NP-Hard.

Theorem 1. *The* PLAN *problem is NP-Hard.*

Proof: To show the non-polynomial complexity of *PLAN*, we will show that the Multiple Knapsack Problem (MKP) [19], whose decision version has already been proven to be strongly NP-complete, can be reduced to this problem in polynomial time.

Consider a special case of allocation A_0, in which all VMs are allocated to one server s_0, then the *PLAN* problem is to find a new allocation $\hat{A}$ for migrating VMs that maximizes the total utility $\mathcal{U}_{A_0 \to \hat{A}}$. We denote $S' = \mathbb{S} \setminus \{s_0\}$ to be the set of destination servers for migration. For a VM v_i, suppose the computed communication cost induced by v_i on all candidate servers is the same, i.e., $C_i(\hat{s}) = \delta_i, \forall \hat{s} \in S'$, where δ_i is a constant. Consider each VM to be an item with size R_i and profit $U(A(v_i) \to \hat{s}) = \mathcal{C}_i(A(v_i)) - \delta_i - C_m(v_i)$, each server $s_j \in S'$ to be knapsack with capacity H_j. The *PLAN* problem becomes finding a feasible subset of VMs to be migrated to servers S', maximizing the total profit. Therefore, the MKP problem is reducible to the *PLAN* problem in polynomial time, and hence the *PLAN* problem is NP-hard.

■

OVERVIEW

- Introduction
- Problem Modeling
- ***PLAN Algorithms***
- Evaluation
- Related Works
- Conclusion

PLAN ALGORITHMS

- Policy-Aware Migration Algorithms: Server hypervisors (or SDN controller) will monitor traffic load for each collocated VM vi. A migration decision phase is triggered periodically during which vi will compute the appropriate destination server for migration. If no migration is needed, then the utility is 0.
- Otherwise, the total utility is increased after migration when: $A(v_i) \neq \hat{s}.$

PLAN ALGORITHMS

- Algorithm 1 PLAN-VM is only triggered for a migration decision every $T_m + \tau$ time, where τ is a random value
- PLAN VM will be suppressed for T_s time period if vi is migrated to a new server. The value of T_s depends on traffic patterns.
- Algorithm 2 PLAN-Server is designed for hypervisors on servers which can accept requests from VMs based on the residual resources of the corresponding server and prepare for migration of remote VMs.
- Several control messages will be exchanged for both PLAN-VM and PLAN-Server.

PLAN ALGORITHMS

- sendMsg(type, destination, resource) sends a control message to the destination.
- getMsg() reads these messages when received.
- The request message is a probe from VM to a destination server for migration. A server can respond by sending back accept or reject message
- If the server accepts the request from distant VM, a migrate message will be sent back as confirmation
- For each VM, the algorithm starts checking feasible servers for improving utility by calling Decision-Migration()
- This function will find a potential destination server for the VM to perform migration
- A blacklist L is maintained to avoid repeat requests to servers that have already rejected the VM.
- If a feasible server accepts the VM's request, it will be migrated to that server.

Algorithm 1 PLAN-VM for v_i

```
/* Triggered every T_m + τ period */
L = ∅
DECISION-MIGRATION(v_i, L)
loop
    msg ← getMsg()
    switch msg.type do
        case reject
            L = L ∪ {msg.sender}
            DECISION-MIGRATION(v_i, L)
        case accept
            sendMsg(migrate, msg.sender, R_i)
            perform migration: v_i → s
    end switch
end loop

function DECISION-MIGRATION(v_i, L)
    s_0 ← A(v_i)
    S_i ← feasible servers in Equation (4)
    X ← arg max_{x∈S_i\L} U(A(v_i) → x)
    if X ≠ ∅ && s_0 ∉ X then
        s ← the one with most residual resources in X
        sendMsg(request, s, R_i)
    else
        exit        ▷ exit whole algorithm if no migration
    end if
end function
```

PLAN ALGORITHMS

- For each server, PLAN-Server keeps listening for incoming migration requests from VMs.
- For a request from VM v_i, server s_j will check its residual resources and send back an accept message if it has enough resources to host v_i.
- Otherwise, it will send a reject message to the VM v_i.

Algorithm 2 PLAN-Server for s_j

```
loop
    msg ← getMsg()
    switch msg.type do
        case request
            v_i = msg.sender
            R_i = msg.resouce
            if Σ_{v_k ∈ A(s_j)} R_k + R_i ≤ H_j then
                sendMsg(accept, v_i)
            else
                sendMsg(reject, v_i)
            end if
        case migrate
            if Σ_{v_k ∈ A(s_j)} R_k + R_i ≤ H_j then
                provisionally resource reservation etc.
            else
                sendMsg(reject, v_i)
            end if
    end switch
end loop
```

PLAN ALGORITHMS

- The PLAN Scheme in Algorithms 1 and 2 can decrease the total communication cost and will eventually converge to a stable state.

Theorem 2. *Algorithms 1 and 2 will converge after a finite number of iterations.*

Proof: The cost of each VM v_i is determined by its hosting server and related ingress/egress middleboxes in $MB^{in}(v_i)$ and $MB^{out}(v_i)$. Hence, under the policy scheme described in the previous section, the migrations of different VMs are independent. Furthermore, each time a migration occurs in *line 11* of Algorithms 1, say, $A(v_i) \rightarrow s$, the utility gained from the migration is always larger than zero, i.e., $U(A(v_i) \rightarrow s) > 0$. Thus, the total induced communication cost, which is always a positive value, is strictly decreasing while VMs are migrating among servers. So, the two algorithms will converge after a finite number of steps. ■

PLAN ALGORITHMS

- Initial Placement: Policy-aware initial placement of VMs is also critical for new VMs in DC networks.
- When a VM instance is initialized, the DC network controller needs to find a suitable server to host the VM. Predefined application-specific policies should be known to the VM. Along with the resource requirements and all servers' residual resources, the feasible decision space can be obtained.
- Even though traffic load might not be available for the VM, a best server can still be chosen by considering traffic of all policies for the VM equally, $\lambda_k = 1, \forall p_k \in P(v_i, *) \cup P(*, v_i)$
- The migration cost is 0 during initial placement.
- The destined server to host v_i is $\arg\max_{s \in \mathcal{S}_i} \mathcal{C}_i(s)$

OVERVIEW

- Introduction
- Problem Modeling
- PLAN Algorithms
- ***Evaluation***
- Related Works
- Conclusion

EVALUATION

- Experimental Setup: PLAN is implemented and evaluated under a fat-tree DC topology. A single VM is modeled as a collection of socket applications communicating with one or more other VMs in the DC network. For each server, a VM hypervisor application is implemented to manage all collocated VMs on the server.
- It supports VM migration among the different servers in the DC network.
- There are limited CPU and memory resources to model a typical DC server's capability.
- In the simulation, 2320 VMs were created on 250 servers. Each VM has an average 10 random outgoing socket connections, with a randomly generated rate.
- A VM migration will only be possible when the target host has sufficient resources and bandwidth, which is considered to be a feasible server.

EVALUATION

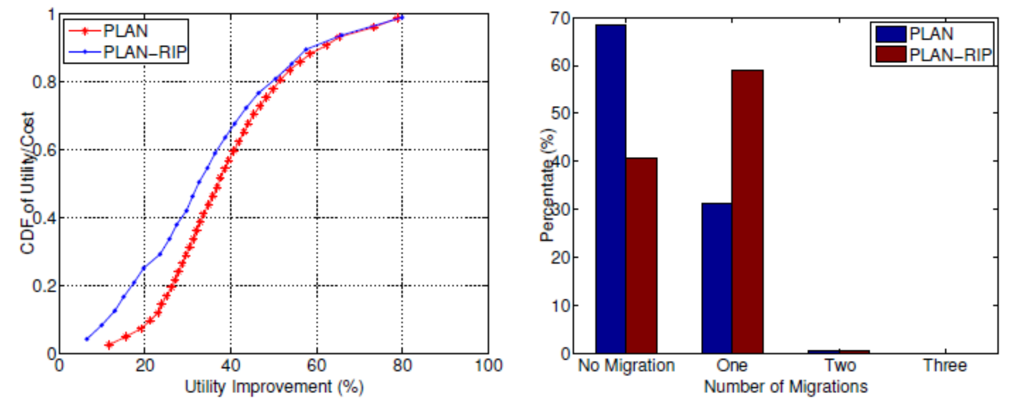
- The policy scheme from PLAN Algorithms has also been implemented.
- 10% of traffic flow will be policy-free, the other 90% of traffic flow will traverse a sequence of middleboxes as required by policies before reaching their destination.
- Each policy constrained traffic flow will traverse 1 to 3 middleboxes.
- To show the benefits of PLAN, it is compared against S-CORE.
- S-CORE is similar to PLAN but it is not a policy aware VM management scheme.
- S-CORE is a live VM migration scheme that reduces communication cost by consolidating VM's, but it does not consider network policies when doing so. A positive utility, communication cost outweighs migration cost, and the server can host the VM, is all that is required for a VM migration to take place under this scheme.

EVALUATION

- PLAN by default is used with the initial placement algorithm described previously.
- S-CORE initially starts with a set of randomly allocated VMs.
- PLAN with Random Initial Placement (PLAN-RIP) does not use the initial placement algorithm.
- The impact of policies on average route length and link utilization is considered.

EVALUATION

- Experimental Results: The following figure demonstrates some unique properties of PLAN in its progress towards convergence in terms of communication cost improvement as well as number of migrations.



(a) CDF: ratio of utility to communication cost

(b) Number of migrations before convergence

Figure 2: Performance of PLAN

EVALUATION

- Part a of the figure shows improvement of individual VM's communication cost after each migration through calculating the ratio of utility to the communication cost of that VM before migration.
- Each migration can reduce communication cost by 39.06% on average for PLAN and 34.19% for PLAN-RIP.
- Nearly 60% of measured migrations can effectively reduce their communication cost by as much as 40%. Improvements are more significant when VMs are allocated randomly at initialization.

EVALUATION

- Part b of the figure shows the number of migrations per VM as PLAN converges to a stable state. In PLAN, only 30% of VMs need to migrate only once to reach a stable state.
- In PLAN-RIP, 60% of VMs need to migrate once when it converges.
- Very few VMs need to migrate twice (less than 1%) and no VM needs to migrate three or more times.
- Low cost, low overhead initial placement can significantly reduce overhead.

EVALUATION

- The following figure shows the snapshot of VM allocations at both the initial and converged states of PLAN.

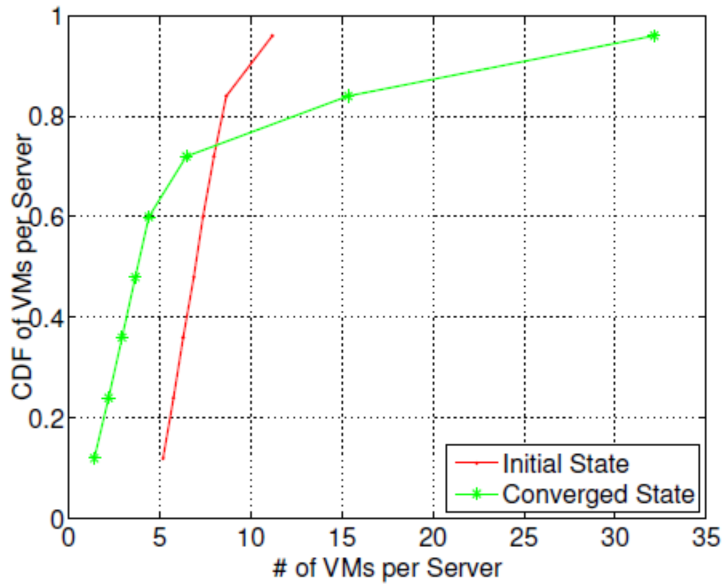


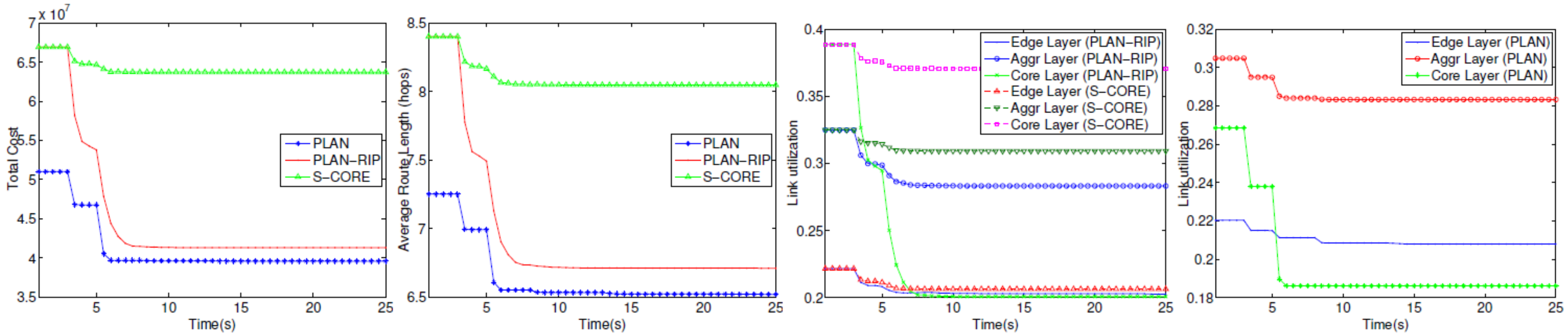
Figure 3: VMs clustering on servers at different states

EVALUATION

- Before PLAN runs, VMs are randomly distributed on servers. Each server hosts between 5 to 12 VMs.
- After PLAN converges, VMs are clustered into several groups of servers.
- Nearly 16% of servers host 56.55% of the total VMs.
- 3.2% of servers are idle when PLAN converges.

EVALUATION

- The following figure shows the overall communication cost reduction (measured in terms of number of bytes using network links), average end-to-end route length, and link utilization for all layers for all three schemes.



(a) Total communication cost (b) Average route length (c) Link utilization (d) Link utilization

Figure 4: Performance comparison of *PLAN* and *S-CORE* VM migration schemes

EVALUATION

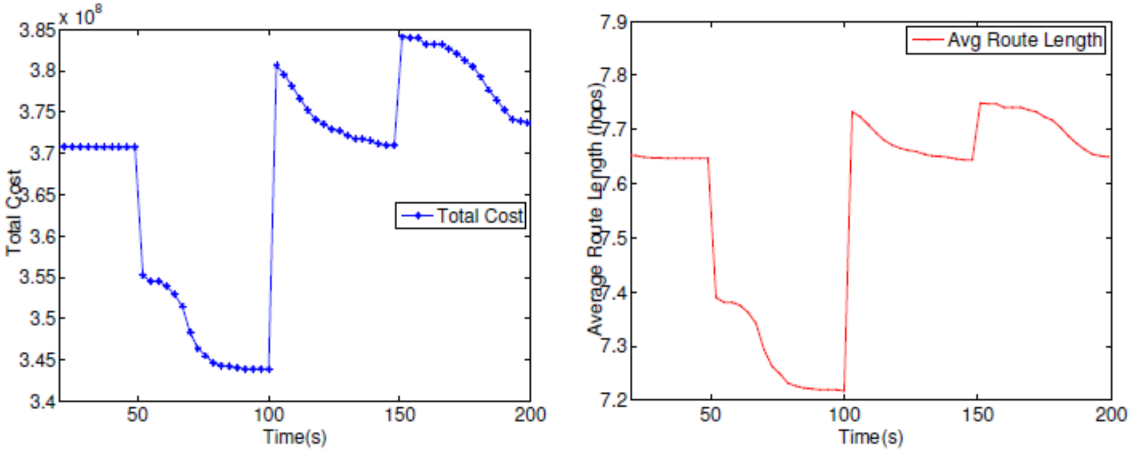
- Part a of the figure shows that PLAN and PLAN-RIP can efficiently converge to a stable allocation.
- PLAN reduces the total communication cost by 22.42% and PLAN-RIP reduces the total communication cost by 38.27%. Much better than S-CORE which only reduces the total communication cost by 4.79%.
- Part b of the figure shows that by migrating VMs, the average route length can be reduced by as much as 20.12% for PLAN-RIP and by as much as 10.08% by PLAN, while S-CORE only reduces the average route length by only 4.22%.
- Both parts a and b in the figure show that PLAN can optimize network-wide communication cost by localizing VMs that frequently communicate with each other, which reduces the length of the end-to-end path.

EVALUATION

- Parts c and d of the figure show that PLAN can mitigate link utilization at the core and aggregation layers by 30.55% and 7.01%, respectively.
- For PLAN-RIP, it can reduce link utilization by 42.87% and 12.81%, respectively.
- For S-CORE, the reduction in link utilization is only by 4.6% and 4.8%, respectively.
- The figure also shows that PLAN's initial placement algorithm can improve communication cost, route length, and link utilization.

EVALUATION

- The following figure shows the algorithm’s performance results when policies are changed at different time intervals, 50s, 100s, and 150s and after the algorithm had initially converged.



(a) Change of total communication cost (b) Change of route length

Figure 5: Performance of *PLAN* with dynamic policies.

EVALUATION

- 10% of policies are removed at 50s, which makes the corresponding traffic flow policy free. The DC is now in a non-optimized state, so there is room for optimizing the VM allocations.
- PLAN can promptly adapt to new policy patterns, which reduces the total communication cost, route length, and link utilization.
- The same can be observed for when new policies are added at 100s and then policies are modified at 150s. Disabling some policies produces new policy-free traffic flow so PLAN can localize their hosting VMs, which improves bandwidth.
- Core-layer link utilization is reduced when some policies are disabled at 50s.
- These results show that PLAN is highly adaptive to dynamism in policy configuration.

OVERVIEW

- Introduction
- Problem Modeling
- PLAN Algorithms
- Evaluation
- ***Related Works***
- Conclusion

RELATED WORKS

- Network policy management research has either focused on devising new policy-based routing-switching or using Software-Defined Networking (SDN) to manage network policies in order to guarantee correctness.
- Another proposed scheme was Player, which is a policy-aware switching layer for DCs consisting of inter-connected policy-aware switches (pswitches).
- A proposed middlebox architecture, CoMb, actively consolidates middlebox features and improves middlebox utilization, which reduces the number of required middleboxes.

RELATED WORKS

- Developments in SDN enable more flexible middlebox deployments over the network while ensuring that traffic will traverse the desired set of middleboxes.
- SIMPLE, is a SDN-based policy enforcement scheme to steer DC traffic in accordance to policy requirements.
- FlowTags leverages SDN's global network visibility and guarantees correctness of policy enforcement.
- These proposals do not consider VMs migration, which risks policy violation and reduced performance

RELATED WORKS

- Mvmotion is a metadata based VM migration approach which reduces the amount of transferred data during migration by utilizing memory de-redundant technique between two physical hosts.
- This does not consider network policy in the design.
- The closest work to PLAN is called PACE (Policy-Aware Application Cloud Embedding).
- PACE only considers one-off VM placement in conjunction with network policies. So it does not further improve resource utilization in the face of dynamic workloads.

OVERVIEW

- Introduction
- Problem Modeling
- PLAN Algorithms
- Evaluation
- Related Works
- ***Conclusion***

CONCLUSION

- In multi-tenant Could Data Centers (DCs), network policies are popularly used to provide secure and high performance services.
- We have studied the optimization of DC network resource usage while adhering to policies governing traffic flow routed over the network (infrastructure).
- PLAN, a policy-aware VM management scheme that meets both efficient DC resource management and middleboxes traversal requirements
- An optimization problem of maximizing the utility of VM migration was modeled. This problem is NP-Hard.
- Based on experimental results, PLAN can reduce network-wide communication cost by 38% over diverse aggregate traffic loads and network policies.
- It is adaptive to changing policy and traffic dynamics.