On Balancing Middlebox Set-up Cost and Bandwidth Consumption in NFV

Jie Wu
(PhD student: Yang Chen)
Center for Networked Computing
Temple University, USA
1. Introduction of Middlebox

- **Network Function Virtualization (NFV)**
  - Technology of virtualizing network functions into software building blocks

- **Middlebox**: software implementation of network services
  - Improve the network performance:
    - Web proxy and video transcoder, load balancer, ...
  - Enhance the security:
    - Firewall, IDS/IPS, passive network monitor, ...

- **Examples**
  - Web Proxy
  - Firewall
  - NAT
Multiple middleboxes may/may not have a serving order

- Examples
  - Firewall usually before Proxy
  - Virus scanner either before or after NAT gateway

Categories
- Non-ordered middlebox set
- Totally-ordered middlebox set (service chain)
- Partially-ordered middlebox set

[1] Dynamic Service Function Chaining in SDN-Enabled Networks with Middleboxes (ICNP '16)
Middlebox Traffic Changing Effects

- Middleboxes may change flow rates in different ways
  - Citrix CloudBridge WAN accelerator: 20% (diminishing)
  - BCH(63,48) encoder: 130% (expanding)

Middlebox Placement Overview

Problem
- Placing middleboxes to satisfy all flows’ middlebox service requests

Objectives:
- Minimizing middlebox setup cost \(^{[3]}\)
- Minimizing bandwidth consumption \(^{[2]}\)

Constraints
- Dependency relations
- Traffic-changing effects
- Vertex capacity and middlebox processing volume

A Middlebox Placement Model [4]

- **Cost**
  - Setup cost
  - Communication cost

- **Objective**
  - Minimizing sum of middlebox setup cost and communication cost

- **Two special cases**
  - Facility location problem
    - Single middlebox placement
  - Generalized assignment problem
    - Each middlebox has a limited processing volume
    - Placing middleboxes and assigning to flows

A Service Chain Model \footnote{[2]}

- **Objective**
  - Minimizing the total bandwidth consumption

- **Solutions**
  - Consider traffic-changing effects
  - Place middleboxes for a single flow

\[
\begin{array}{c}
\text{m}_1 & \text{m}_2 & \text{m}_3 \\
\text{Non-ordered} & \text{(Optimal greedy: sort traffic-changing ratios in increasing order)}
\end{array}
\]

\[
\begin{array}{c}
\text{m}_1 & \text{m}_2 & \text{m}_3 \\
\text{Totally-ordered} & \text{(Optimal DP: latter middleboxes must be after front ones)}
\end{array}
\]

\[
\begin{array}{c}
\text{m}_1 & \text{m}_2 & \text{m}_3 \\
\text{Partially-ordered} & \text{(NP-hard: reduced from the Clique Problem)}
\end{array}
\]

\footnote{[2] Traffic Aware Placement of Interdependent NFV Middleboxes (INFOCOM '17)}
2. Our Model

- Problem
  - Placing middleboxes to satisfy all flows' network service requests

- Network service requests
  - Multiple middleboxes
    - Middlebox set with or without dependency relations

- Cost
  - Middlebox setup
    - Sum of middlebox setup cost
  - Bandwidth consumption
    - Sum of each flow's bandwidth consumption cost on each link

- Objective
  - Minimizing total cost of middlebox setup and bandwidth consumption
**A Motivating Example**

Independent middleboxes

Dependent middleboxes: $m'$ before $m$

Red flow with high rate

A flow covered by multiple middleboxes

(Multiple coverage: when additional setup cost is less than the reduced bandwidth consumption cost)
3. Problem Formulation

- **Middlebox setup cost**
  \[ c_1 = \sum_{m \in M} \sum_{v \in V} c_m \]
  - \( c_m \): unit setup cost of middlebox \( m \)

- **Bandwidth consumption cost**
  \[ c_2 = \sum_{f \in F} \sum_{e \in p_f} w(b^e_f) \]
  - \( w(b^e_f) \): bandwidth cost function of flow \( f \) on link \( e \)

  \[ b^e_f = r_f \prod_{m} \lambda_m \]
  - \( r_f \): initial traffic rate of flow \( f \)
  - \( \lambda_m \): traffic-changing ratio of middlebox \( m \)

- **Objective**
  - Minimizing \( c_1 + c_2 \)
Problem Formulation (cont’d)

- Translog bandwidth cost function on each link

\[ w(b^e_f) = \log(b^e_f) = \log(r_f \prod \lambda_m) = \log(r_f) + \sum \log(\lambda_m) \]

- Reasons
  - Widely used in Cisco EIGRP and OSPF protocols
  - Log-linear for easy calculation

- The weight of setup cost and bandwidth consumption
  - Adjusting the traffic-changing ratios and unit setup costs of middleboxes
NP-hard

- Even with no traffic-changing effects
- Even when placing a single middlebox

Proof

- Reduction from set-cover problem
- Use minimum number of middleboxes to “cover” all flows
  - Flows as elements: $F=\{f_1,f_2,\ldots, f_{|F|}\}$
  - Placed middleboxes as sets: $\{S_1, S_2,\ldots\}$
  - $S_1=\{f_1, f_2, f_4\}$, $S_2=\{f_1, f_2\}$, $S_3=\{f_3\}$
In this paper, we focus on tree-structured topologies. Each triangle is mostly a perfect or complete tree.
4. Placing a Single Middlebox

Solution
- Local Greedy Algorithm (LGA)

Steps
- Calculate each total cost of placing middleboxes in a whole level
- Select the level with the minimum total cost
- Iterative implementation
  - From top level to bottom, total costs will decrease and then increase
  - Select the level with the local minimum
4. Placing a Single Middlebox (cont’d)

Time complexity (|V|: #node)
- $O(|V|)$

Optimal for perfect tree topologies
- Symmetry of placement
- No multiple “coverage” situation

Also optimal for complete tree topologies
- Also multiple “coverage” situation
- The most unbalanced traffic distribution: left and right subtrees of root have a depth difference of 1
Illustration

Calculate level by level
5. Placing Multiple Middleboxes

Non-ordered middlebox set placement

Solution
- Combined Local Greedy Algorithm (CLGA)

Insight
- Place each middlebox independently by applying LGA

Time complexity (|V|: #node, |M|: #middlebox)
- \( O(|V||M|) \)

Optimal for complete trees
Totally-ordered Middlebox Set Placement

- Solution: Dynamic Programming (DP)
- Works for infinite and finite vertex capacity
- \( \text{OPT}(i, j) \)
  - Minimum cost of subtree with root \( v_i \) when placing first \( j \) middleboxes in the set
  - \( \infty \) if capacity is not enough
Dynamic Programming Formulation

- **Left triangle**
  
  \[
  \text{OPT}(i, j) = \begin{cases} 
  \min_{0 \leq k \leq j} \{ \text{OPT}(2i, k) + \text{OPT}(2i + 1, k) \\ + \sum_{k < l \leq j} c_l + [\log i] \sum_{k < l \leq j} \lambda_l \}, & 1 \leq i \leq \lfloor \frac{n}{2} \rfloor. \\
  \sum_{0 \leq l \leq j} c_l + [\log i] r_f, & \lfloor \frac{n}{2} \rfloor < i \leq n. \\
  \infty & \text{if not enough node capacity.} \\
  0 & \text{otherwise.}
  \end{cases}
  \]
  
  - **Left subtree**
  - **Right subtree**
  - Bandwidth consumption
  - Newly placed middleboxes

- **Right triangle**
  - Similar to the left triangle's formulation
An Example

- Dependency relations
  - $m_1 \rightarrow m_2 \rightarrow m_3$

- Initial traffic rate
  - $r_1 = r_2 = r_3 = 1$

<table>
<thead>
<tr>
<th></th>
<th>$m_1$</th>
<th>$m_2$</th>
<th>$m_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic-changing ratio</td>
<td>0.5</td>
<td>0.8</td>
<td>1.1</td>
</tr>
<tr>
<td>Setup cost</td>
<td>0.2</td>
<td>0.4</td>
<td>0.3</td>
</tr>
</tbody>
</table>
Totally-ordered Middlebox Set Placement (cont’d)

Insights
- The optimal placement with root \( v_i \) by placing first \( j \) and its two subtrees by placing no more than \( j \) middleboxes

Perfect tree
- Transformed to a line
- Similar to a single flow placement

Complete tree
- No multiple “coverage” situation

Time complexity (\(|V|\): #node, \(|M|\):#middlebox)
- \( O(|V||M|^3) \)
Partially-ordered Middlebox Set Placement

- NP-hard even for a single flow [2]
- One heuristic solution
  - Insight
    - Transform into a totally-ordered middlebox set
  - Steps ($\lambda$: traffic-changing ratio)
    - Treat middleboxes with dependencies as a single middlebox
    - Sort middleboxes in increasing order of $\lambda$
  - Example
    - Middlebox set
      |   | m1 | m2 | m3 | m4 | m5 | m6 |
      |---|----|----|----|----|----|----|
      |$\lambda$| 0.7 | 1.1 | 0.8 | 1.1 | 0.5 | 1.4 |
    - Dependency relationship: $m_2 \rightarrow m_3$, $m_4 \rightarrow m_5 \rightarrow m_6$

Partially-ordered Middlebox Set Placement (cont’d)

- Another heuristic solution
  - Insight
    - Transform into a non-ordered middlebox set
  - Steps
    - Treat middleboxes with dependencies as a single middlebox by a topological order
    - No dependency relations among new middleboxes
  - Example
    - Middlebox set
      - Dependency relationship: \( m_2 \rightarrow m_3, m_4 \rightarrow m_5 \rightarrow m_6 \)
6. Handling Heterogeneous flows for Non-ordered Middlebox Set

- **Group Flows by Initial Bandwidths (GFIB)**
  - Group flows by initial traffic rates ($r_f$: f's traffic rate)
    - $\#\text{group: } \left\lceil \log_2 \frac{\max r_f}{\min r_f} \right\rceil + 1$
    - The traffic rate range of the $i$th group: $2^{i-1} \times \min r_f \leq r_f < 2^i \times \min r_f$
  - Treat flows in each group as homogeneous
  - Apply CLGA for each group

- **An example**

  ![Diagram](image)

  - $\max r_f = 7$
  - $\min r_f = 1$
  - Group 1: [1,2)
  - Group 2: [2,4)
  - Group 3: [4,8)
6. Handling Heterogeneous Flows for Non-ordered Middlebox Set (cont’d)

- Time complexity

\[
\max\{O(|V| \log |V|), O(|V| (\left\lceil \log_2 \frac{\max r_f}{\min r_f} \right\rceil + 1))\}
\]

- Performance-guaranteed algorithm
  - Approximation ratio \(^5\):
    \[
    \left\lceil \log_2 \frac{\max r_f}{\min r_f} \right\rceil + 1
    \]

---

\(^5\) On Optimal Scheduling of Multiple Mobile Chargers in Wireless Sensor Networks (MSCC ’14)
7. Simulation

- Our algorithms
  - **LGA**
    - Single middlebox
    - Select the level with the minimum cost
  - **CLGA**
    - Non-ordered middlebox set
    - Apply LGA independently
  - **DP**
    - Totally-ordered middlebox set
    - Dynamic programming
  - **GFIB**
    - Heterogeneous flows
    - Group flows by initial traffic rates
    - Combine placement by applying CLGA for each group
7. Simulation

- Comparison algorithms
  - Random-fit
    - Randomly place middleboxes until all flows are satisfied
  - NOSP \(^2\)
    - Place middleboxes in increasing order of traffic-changing effects for each flow from source to destination independently
    - For single middlebox or non-ordered middlebox set
  - TOSP \(^2\)
    - Dynamic programming based algorithm for each flow independently
    - For totally-ordered middlebox set with or without vertex capacity

\(^2\) Traffic aware placement of interdependent NFV middleboxes (INFOCOM ’17)
Settings

- **Topology**
  - Perfect 5-layer binary tree for each triangle

- **Facebook data center traffic trace**
  - Single-flow initial traffic rate: 1~6 Mb

- **Middlebox set**

<table>
<thead>
<tr>
<th></th>
<th>m₁</th>
<th>m₂</th>
<th>m₃</th>
<th>m₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic-changing ratio</td>
<td>0.7</td>
<td>0.8</td>
<td>1.1</td>
<td>1.2</td>
</tr>
<tr>
<td>Setup cost</td>
<td>0.4</td>
<td>0.6</td>
<td>0.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

- **Dependency relationship**
  - m₂ -> m₃ -> m₁ -> m₄
Simulation Results

- **LGA** costs 20.3% less than **NOSP** and 35.1% less than **Random-fit**.
- **CLGA** performs the best even with heavy traffic.
- The performance of **Random-fit** is not steady.
- For heterogeneous flows, **GFIB** saves about 36.9% and 34.0% compared to **NOSP** and **Random-fit**.
We start with homogeneous when placing a single type of middlebox for homogeneous in not only bandwidth consumption, but also in middlebox setup cost. The superiority of CLGA is more obvious in Fig. 4 than that of the combination of the optimal placement of each middlebox. As a non-ordered middlebox set subsection, all only bandwidth consumption, but also middlebox setup cost.

Next, we show the placement of a totally-ordered middlebox set NOSP (a) Total cost value. (b) Middlebox setup cost.

The table shows the total cost value and set-up cost for different middlebox orderings at 3 Mbps (DP). The total cost is larger than the non-ordered middlebox set. Limited vertex capacity increases the minimum cost. The order of a middlebox set matters not only for total cost but also for set-up cost.

<table>
<thead>
<tr>
<th>Totally-ordered middleboxes</th>
<th>Total cost</th>
<th>Set-up cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m_2 \rightarrow m_3 \rightarrow m_1 \rightarrow m_4 )</td>
<td>20.9</td>
<td>10.4</td>
</tr>
<tr>
<td>( m_3 \rightarrow m_1 \rightarrow m_2 \rightarrow m_4 )</td>
<td>23.7</td>
<td>12.0</td>
</tr>
<tr>
<td>( m_1 \rightarrow m_4 \rightarrow m_3 \rightarrow m_2 )</td>
<td>22.8</td>
<td>9.6</td>
</tr>
<tr>
<td>( m_1 \rightarrow m_2 \rightarrow m_3 \rightarrow m_4 )</td>
<td>11.9</td>
<td>4.4</td>
</tr>
<tr>
<td>( m_4 \rightarrow m_3 \rightarrow m_2 \rightarrow m_1 )</td>
<td>24.7</td>
<td>10.2</td>
</tr>
</tbody>
</table>
8. Conclusion and Future Work

- Middlebox constraints
  - Traffic-changing effects
  - Dependency relations
  - Flow sharing

- Middlebox placement
  - Balancing middlebox set-up cost and bandwidth consumption

- Tree-structured topologies
  - Optimal algorithms for homogeneous flows
  - Performance-guaranteed algorithm for heterogeneous flows

- Future work
  - General tree-structures
Minimizing the makespan
Minimizing the average completion time
Q & A