HybridFlow: Achieving Load Balancing in Software-Defined WANs with Scalable Routing

Songshi Dou
Supervisor: Zehua Guo

School of Automation
Beijing Institute of Technology

January 16, 2021

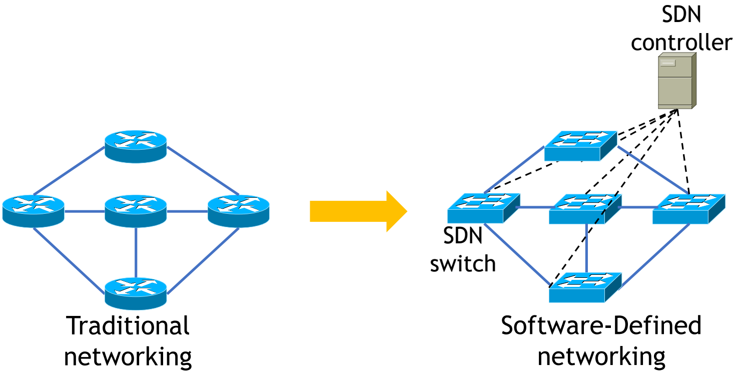
Outline

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

Outline

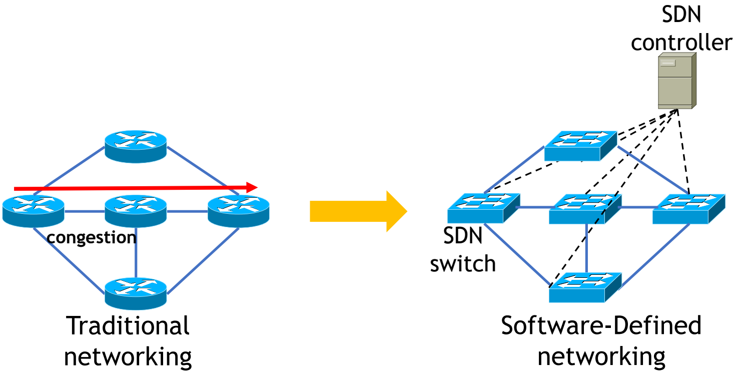
北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

Software-Defined Networking (SDN)



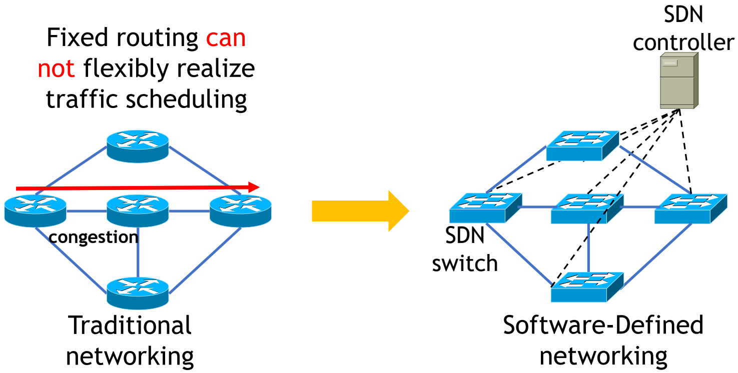
An example of the advantages of SDN.

Software-Defined Networking (SDN)



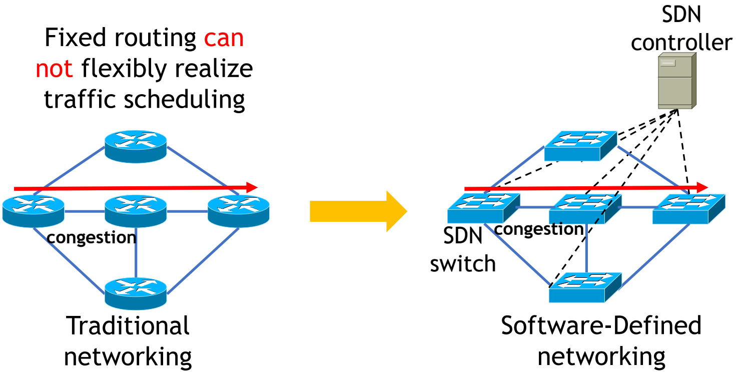
An example of the advantages of SDN.

Software-Defined Networking (SDN)



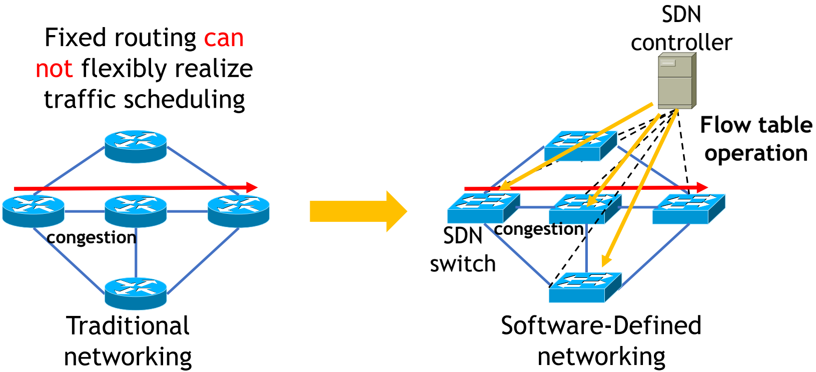
An example of the advantages of SDN.

Software-Defined Networking (SDN)



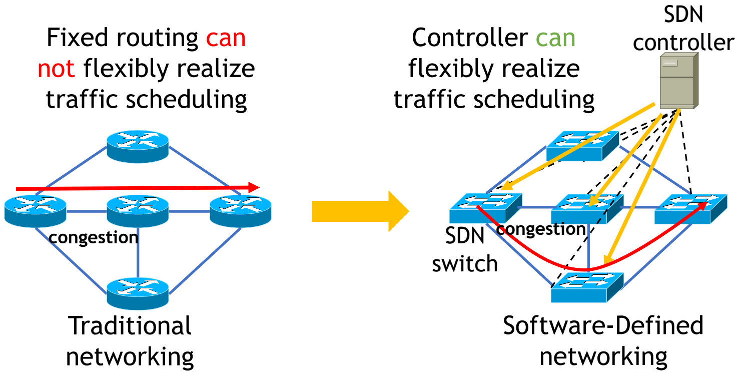
An example of the advantages of SDN.

Software-Defined Networking (SDN)



An example of the advantages of SDN.

Software-Defined Networking (SDN)



An example of the advantages of SDN.

Outline

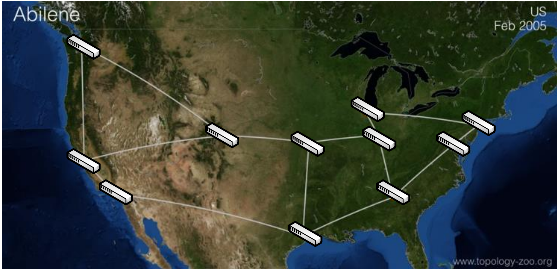
北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

Software-Defined Wide Area Networks (SD-WANs)

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Advantages of deploying SDN into WANs

- High-quality
- Low-latency
- Resilient
- Customized



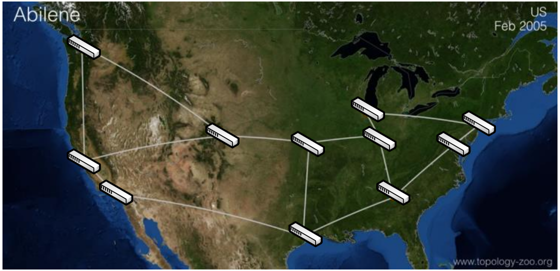
The topology of Abilene.

Software-Defined Wide Area Networks (SD-WANs)

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

Advantages of deploying SDN into WANs

- High-quality
- Low-latency
- Resilient
- Customized



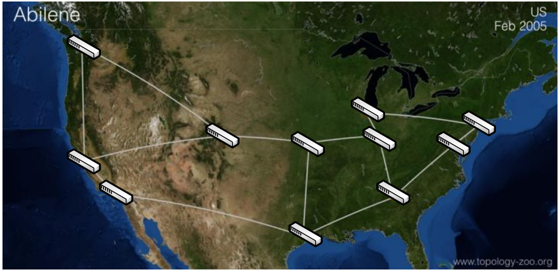
The topology of Abilene.

Software-Defined Wide Area Networks (SD-WANs)

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Advantages of deploying SDN into WANs

- High-quality
- Low-latency
- Resilient
- Customized



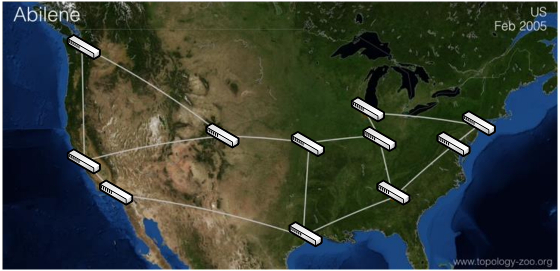
The topology of Abilene.

Software-Defined Wide Area Networks (SD-WANs)

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

Advantages of deploying SDN into WANs

- High-quality
- Low-latency
- Resilient
- Customized



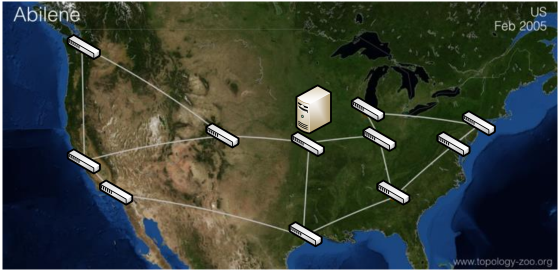
The topology of Abilene.

Software-Defined Wide Area Networks (SD-WANs)

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

The single-controller control plane

- One controller controls the whole network
- Easy to deploy and maintain the consistent network view



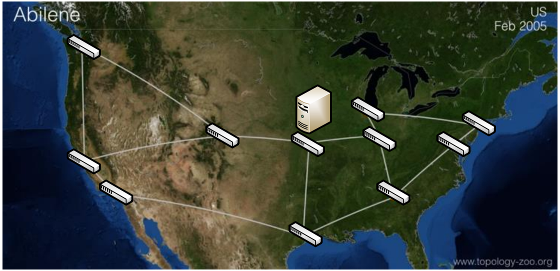
The single-controller control plane.

Software-Defined Wide Area Networks (SD-WANs)

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

The single-controller control plane

- One controller controls the whole network
- Easy to deploy and maintain the consistent network view

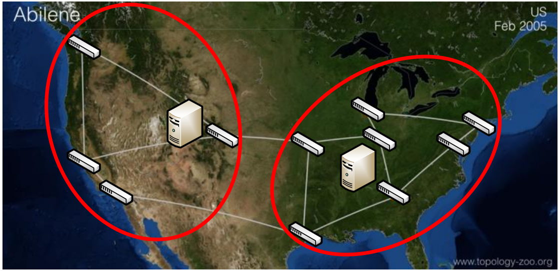


The single-controller control plane.

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

The multi-controller control plane

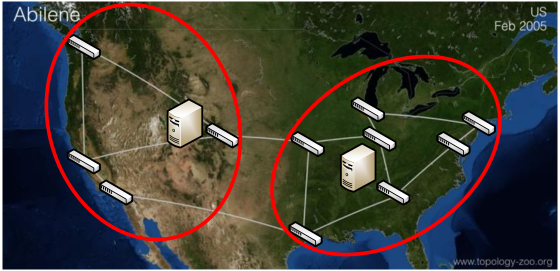
- Partitioning the network into domains
- Physically distributed but logically centralized control plane



The multi-controller control plane.

The multi-controller control plane

- Partitioning the network into domains
- Physically distributed but logically centralized control plane



The multi-controller control plane.

Outline

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Outline

北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

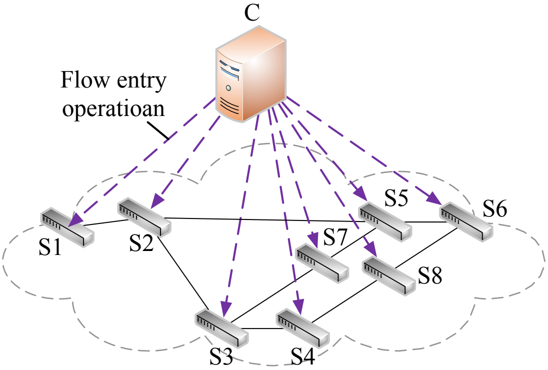
Limitation of existing routing schemes

With the single controller

- Path calculation (using a routing algorithm)
- Path deployment (deploying flow entries to install, update, or delete paths for flows)

Limitations

- The calculation and establishment of per-flow paths for many flows consume large processing ability of controller
- The single-controller control plane suffers from limited processing ability of controller



Single controller with per-flow routing.

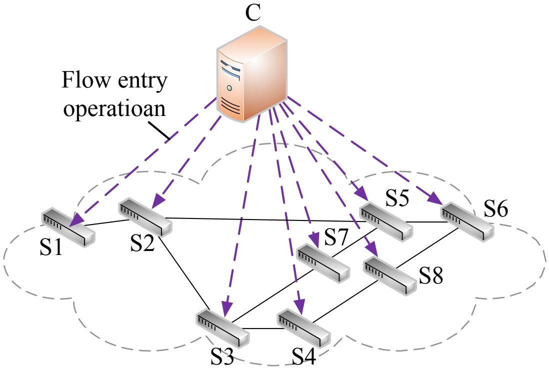
Limitation of existing routing schemes

With the single controller

- Path calculation (using a routing algorithm)
- Path deployment (deploying flow entries to install, update, or delete paths for flows)

Limitations

- The calculation and establishment of per-flow paths for many flows consume large processing ability of controller
- The single-controller control plane suffers from limited processing ability of controller



Single controller with per-flow routing.

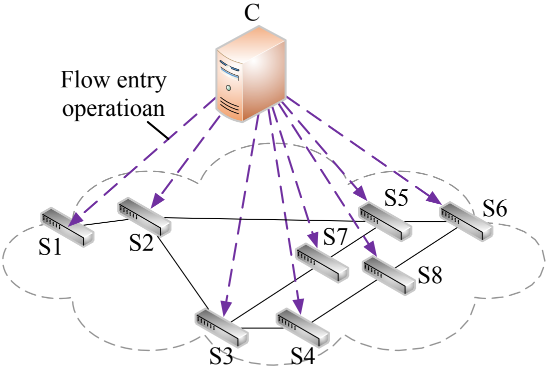
Limitation of existing routing schemes

With the single controller

- Path calculation (using a routing algorithm)
- Path deployment (deploying flow entries to install, update, or delete paths for flows)

Limitations

- The calculation and establishment of per-flow paths for many flows consume large processing ability of controller
- The single-controller control plane suffers from limited processing ability of controller



Single controller with per-flow routing.

Limitation of existing routing schemes

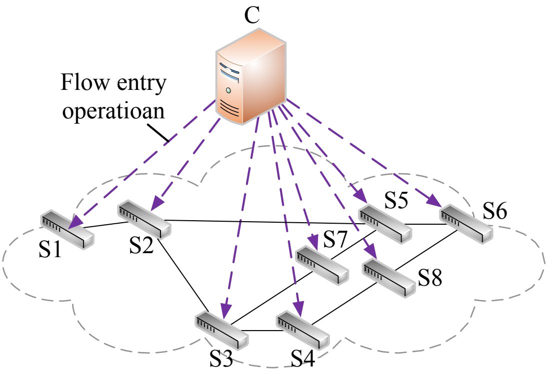
北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

With the single controller

- Path calculation (using a routing algorithm)
- Path deployment (deploying flow entries to install, update, or delete paths for flows)

Limitations

- The calculation and establishment of per-flow paths for many flows consume large processing ability of controller
- The single-controller control plane suffers from limited processing ability of controller



Single controller with per-flow routing.

Limitation of existing routing schemes

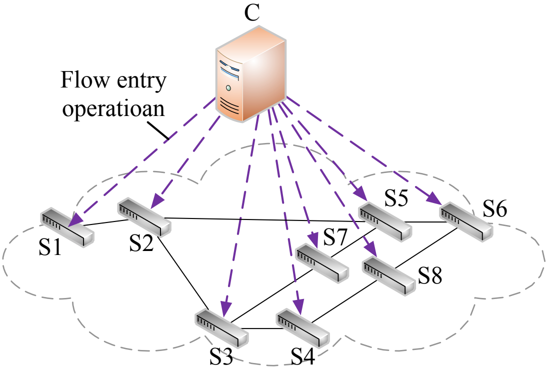
北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

With the single controller

- Path calculation (using a routing algorithm)
- Path deployment (deploying flow entries to install, update, or delete paths for flows)

Limitations

- The calculation and establishment of per-flow paths for many flows consume large processing ability of controller
- The single-controller control plane suffers from limited processing ability of controller



Single controller with per-flow routing.

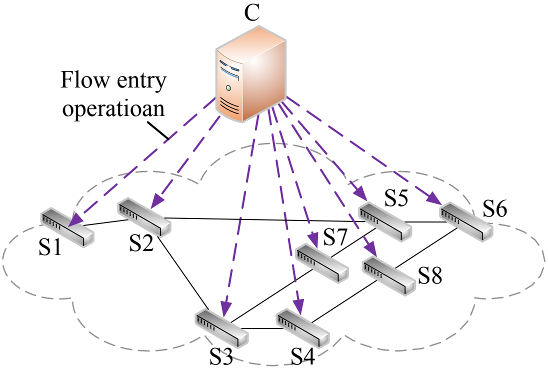
Limitation of existing routing schemes

With the single controller

- Path calculation (using a routing algorithm)
- Path deployment (deploying flow entries to install, update, or delete paths for flows)

Limitations

- The calculation and establishment of per-flow paths for many flows consume large processing ability of controller
- The single-controller control plane suffers from limited processing ability of controller



Single controller with per-flow routing.

Limitation of existing routing schemes

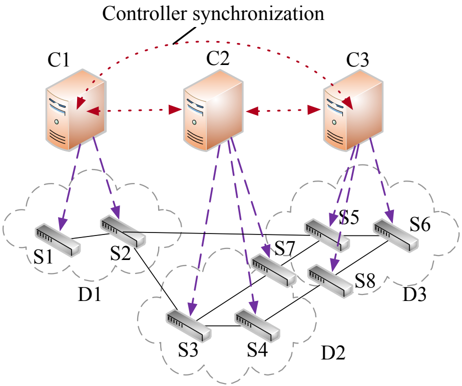
With the multiple controllers

- Synchronizing among controllers to maintain consistent network view
- Avoiding the limited processing ability of a single controller

Limitations

Network performance is affected by synchronization

- When synchronized network information arrive controllers at different time
- When a flow's path traverses multiple domains
- Extra resource consumption



Multiple controllers with per-flow routing.

Limitation of existing routing schemes

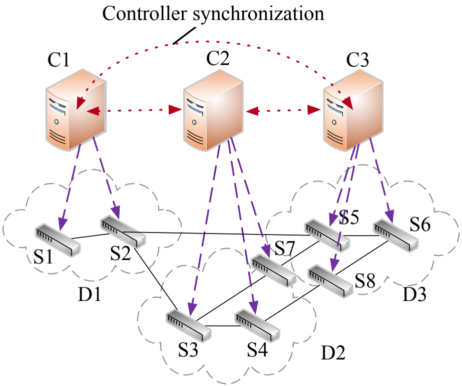
With the multiple controllers

- Synchronizing among controllers to maintain consistent network view
- Avoiding the limited processing ability of a single controller

Limitations

Network performance is affected by synchronization

- When synchronized network information arrive controllers at different time
- When a flow's path traverses multiple domains
- Extra resource consumption



Multiple controllers with per-flow routing.

Limitation of existing routing schemes

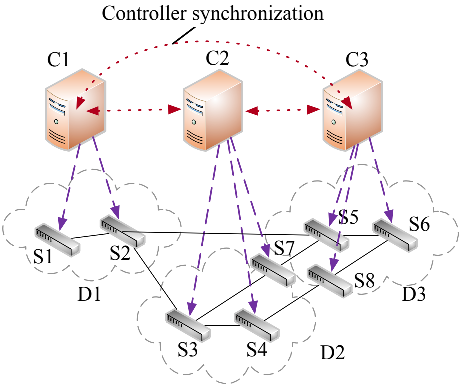
With the multiple controllers

- Synchronizing among controllers to maintain consistent network view
- Avoiding the limited processing ability of a single controller

Limitations

Network performance is affected by synchronization

- When synchronized network information arrive controllers at different time
- When a flow's path traverses multiple domains
- Extra resource consumption



Multiple controllers with per-flow routing.

Limitation of existing routing schemes

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

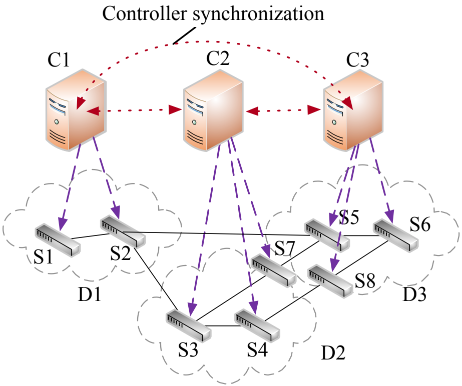
With the multiple controllers

- Synchronizing among controllers to maintain consistent network view
- Avoiding the limited processing ability of a single controller

Limitations

Network performance is affected by synchronization

- When synchronized network information arrive controllers at different time
- When a flow's path traverses multiple domains
- Extra resource consumption



Multiple controllers with per-flow routing.

Limitation of existing routing schemes

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

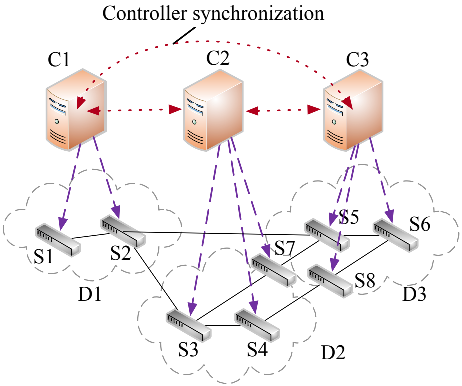
With the multiple controllers

- Synchronizing among controllers to maintain consistent network view
- Avoiding the limited processing ability of a single controller

Limitations

Network performance is affected by synchronization

- When synchronized network information arrive controllers at different time
- When a flow's path traverses multiple domains
- Extra resource consumption



Multiple controllers with per-flow routing.

Limitation of existing routing schemes

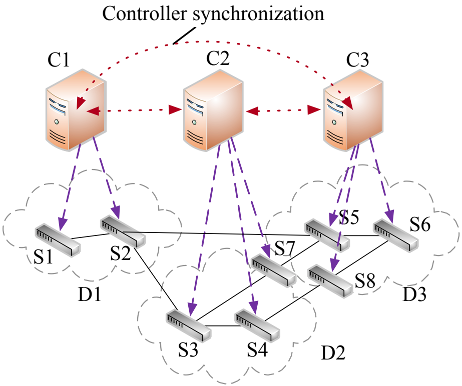
With the multiple controllers

- Synchronizing among controllers to maintain consistent network view
- Avoiding the limited processing ability of a single controller

Limitations

Network performance is affected by synchronization

- When synchronized network information arrive controllers at different time
- When a flow's path traverses multiple domains
- Extra resource consumption



Multiple controllers with per-flow routing.

Limitation of existing routing schemes

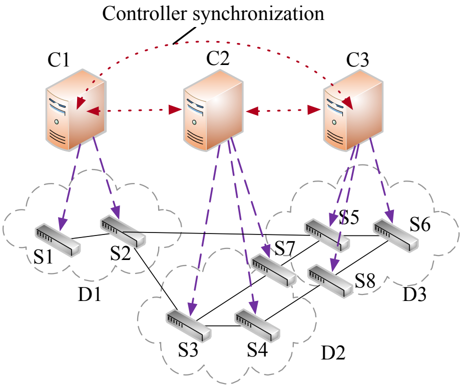
With the multiple controllers

- Synchronizing among controllers to maintain consistent network view
- Avoiding the limited processing ability of a single controller

Limitations

Network performance is affected by synchronization

- When synchronized network information arrive controllers at different time
- When a flow's path traverses multiple domains
- Extra resource consumption



Multiple controllers with per-flow routing.

Outline

北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

Design challenges

Existing solution with single controller

- Suffering from limited processing ability of a single controller

Existing solution with multiple controllers

- Suffering from degradation of network performance due to synchronization

An ideal solution

- A simple but scalable solution that employs the single controller to maintain good performance with low processing load

What to consider when realizing the ideal solution

- When the controller reroutes flows
- Which flows should be rerouted by the controller
- Which new paths should be used for the rerouted flows
- How the controller updates the flows' paths

Design challenges

北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

Existing solution with single controller

- Suffering from limited processing ability of a single controller

Existing solution with multiple controllers

- Suffering from degradation of network performance due to synchronization

An ideal solution

- A simple but scalable solution that employs the single controller to maintain good performance with low processing load

What to consider when realizing the ideal solution

- When the controller reroutes flows
- Which flows should be rerouted by the controller
- Which new paths should be used for the rerouted flows
- How the controller updates the flows' paths

Design challenges

Existing solution with single controller

- Suffering from limited processing ability of a single controller

Existing solution with multiple controllers

- Suffering from degradation of network performance due to synchronization

An ideal solution

- A simple but scalable solution that employs the single controller to maintain good performance with low processing load

What to consider when realizing the ideal solution

- When the controller reroutes flows
- Which flows should be rerouted by the controller
- Which new paths should be used for the rerouted flows
- How the controller updates the flows' paths

Design challenges

Existing solution with single controller

- Suffering from limited processing ability of a single controller

Existing solution with multiple controllers

- Suffering from degradation of network performance due to synchronization

An ideal solution

- A simple but scalable solution that employs the single controller to maintain good performance with low processing load

What to consider when realizing the ideal solution

- When the controller reroutes flows
- Which flows should be rerouted by the controller
- Which new paths should be used for the rerouted flows
- How the controller updates the flows' paths

Design challenges

Existing solution with single controller

- Suffering from limited processing ability of a single controller

Existing solution with multiple controllers

- Suffering from degradation of network performance due to synchronization

An ideal solution

- A simple but scalable solution that employs the single controller to maintain good performance with low processing load

What to consider when realizing the ideal solution

- When the controller reroutes flows
- Which flows should be rerouted by the controller
- Which new paths should be used for the rerouted flows
- How the controller updates the flows' paths

Design challenges

Existing solution with single controller

- Suffering from limited processing ability of a single controller

Existing solution with multiple controllers

- Suffering from degradation of network performance due to synchronization

An ideal solution

- A simple but scalable solution that employs the single controller to maintain good performance with low processing load

What to consider when realizing the ideal solution

- When the controller reroutes flows
- Which flows should be rerouted by the controller
- Which new paths should be used for the rerouted flows
- How the controller updates the flows' paths

Design challenges

Existing solution with single controller

- Suffering from limited processing ability of a single controller

Existing solution with multiple controllers

- Suffering from degradation of network performance due to synchronization

An ideal solution

- A simple but scalable solution that employs the single controller to maintain good performance with low processing load

What to consider when realizing the ideal solution

- When the controller reroutes flows
- Which flows should be rerouted by the controller
- Which new paths should be used for the rerouted flows
- How the controller updates the flows' paths

Design challenges

Existing solution with single controller

- Suffering from limited processing ability of a single controller

Existing solution with multiple controllers

- Suffering from degradation of network performance due to synchronization

An ideal solution

- A simple but scalable solution that employs the single controller to maintain good performance with low processing load

What to consider when realizing the ideal solution

- When the controller reroutes flows
- Which flows should be rerouted by the controller
- Which new paths should be used for the rerouted flows
- How the controller updates the flows' paths

Outline

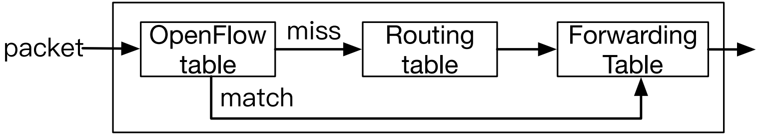
Outline

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

Opportunity

Hybrid Routing

- If a matched entry is found, the packet is processed based on the actions under the entry in OpenFlow routing mode
- If a matched entry is not found, the packet is further forward to the traditional routing table under OSPF routing mode
- With the hybrid routing, we can address the above concern (4) by reducing the number of the controller's operation for path establishment/update/deletion

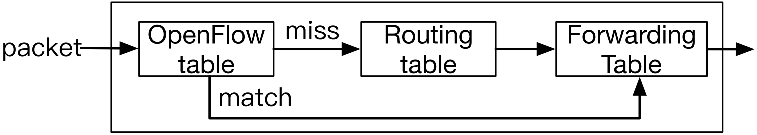


Hybrid OpenFlow/OSPF routing mode in a commercial SDN switch.

Opportunity

Hybrid Routing

- If a matched entry is found, the packet is processed based on the actions under the entry in OpenFlow routing mode
- If a matched entry is not found, the packet is further forward to the traditional routing table under OSPF routing mode
- With the hybrid routing, we can address the above concern (4) by reducing the number of the controller's operation for path establishment/update/deletion

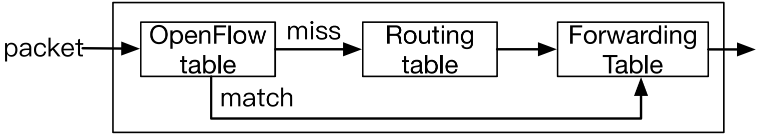


Hybrid OpenFlow/OSPF routing mode in a commercial SDN switch.

Opportunity

Hybrid Routing

- If a matched entry is found, the packet is processed based on the actions under the entry in OpenFlow routing mode
- If a matched entry is not found, the packet is further forward to the traditional routing table under OSPF routing mode
- With the hybrid routing, we can address the above concern (4) by reducing the number of the controller's operation for path establishment/update/deletion



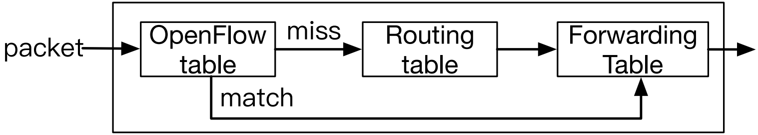
Hybrid OpenFlow/OSPF routing mode in a commercial SDN switch.

Opportunity

BEIJING INSTITUTE OF TECHNOLOGY

Hybrid Routing

- If a matched entry is found, the packet is processed based on the actions under the entry in OpenFlow routing mode
- If a matched entry is not found, the packet is further forward to the traditional routing table under OSPF routing mode
- With the hybrid routing, we can address the above concern (4) by reducing the number of the controller's operation for path establishment/update/deletion



Hybrid OpenFlow/OSPF routing mode in a commercial SDN switch.

Opportunity

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

Crucial Flow Rerouting

- Observation: Based on our analysis of real world traffic matrices, there exist some crucial links which have high traffic load and are likely to experience congestion
- We present a metric called Variation Slope to select crucial links, and flows on crucial links are recognized as crucial flows
- With the crucial flow rerouting, we can address the above concerns (1)-(3) by
  - deciding the right time to reroute flows
  - reducing the number of rerouted flows
  - reducing the overhead to calculate the path for the controller

Opportunity

Crucial Flow Rerouting

- Observation: Based on our analysis of real world traffic matrices, there exist some crucial links which have high traffic load and are likely to experience congestion
- We present a metric called Variation Slope to select crucial links, and flows on crucial links are recognized as crucial flows
- With the crucial flow rerouting, we can address the above concerns (1)-(3) by
  - deciding the right time to reroute flows
  - reducing the number of rerouted flows
  - reducing the overhead to calculate the path for the controller

Opportunity

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Crucial Flow Rerouting

- Observation: Based on our analysis of real world traffic matrices, there exist some crucial links which have high traffic load and are likely to experience congestion
- We present a metric called Variation Slope to select crucial links, and flows on crucial links are recognized as crucial flows
- With the crucial flow rerouting, we can address the above concerns (1)-(3) by
  - deciding the right time to reroute flows
  - reducing the number of rerouted flows
  - reducing the overhead to calculate the path for the controller

Opportunity

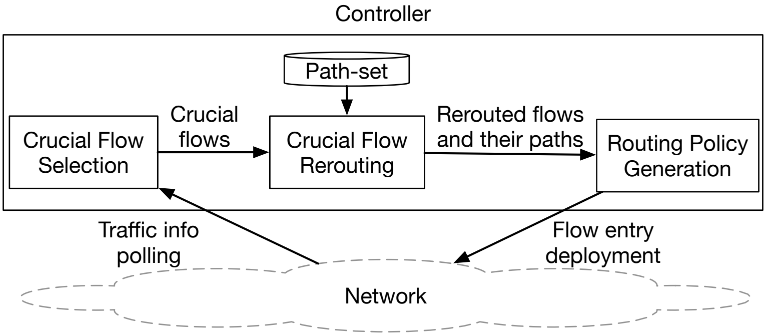
北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Crucial Flow Rerouting

- Observation: Based on our analysis of real world traffic matrices, there exist some crucial links which have high traffic load and are likely to experience congestion
- We present a metric called Variation Slope to select crucial links, and flows on crucial links are recognized as crucial flows
- With the crucial flow rerouting, we can address the above concerns (1)-(3) by
  - deciding the right time to reroute flows
  - reducing the number of rerouted flows
  - reducing the overhead to calculate the path for the controller

Outline

Design overview

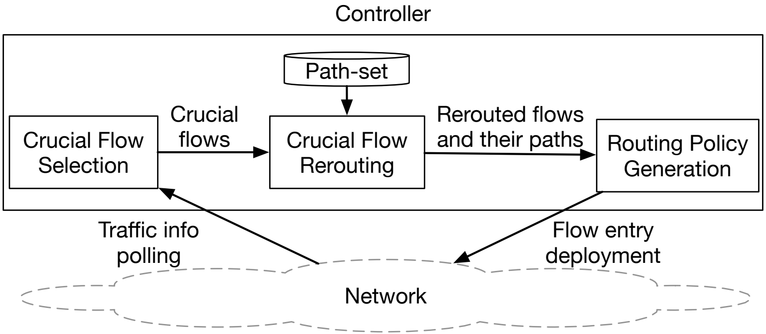


HybridFlow structure.

HybridFlow structure

- Modules of Crucial Flow Selection and Crucial Flow Rerouting realize Crucial Flow Rerouting
- Routing Policy Generation module realizes Hybrid Routing

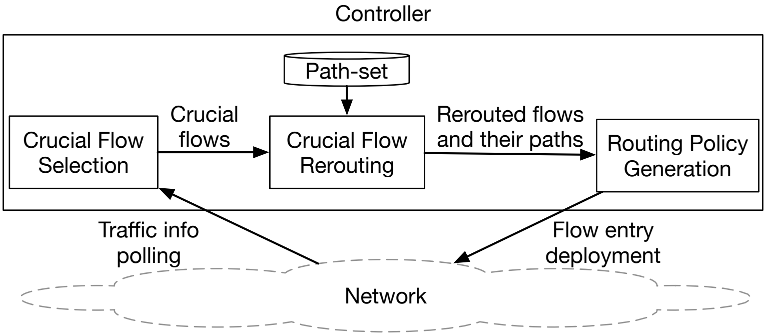
Design overview



HybridFlow structure.

HybridFlow structure

- Modules of Crucial Flow Selection and Crucial Flow Rerouting realize Crucial Flow Rerouting
- Routing Policy Generation module realizes Hybrid Routing

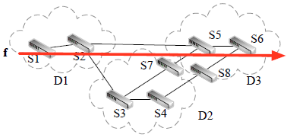


HybridFlow structure.

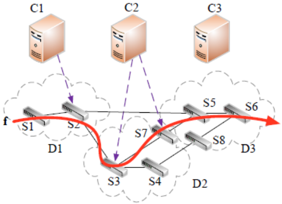
HybridFlow structure

- Modules of Crucial Flow Selection and Crucial Flow Rerouting realize Crucial Flow Rerouting
- Routing Policy Generation module realizes Hybrid Routing

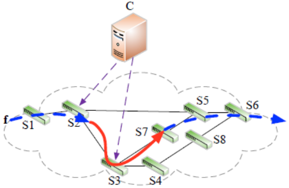
Design overview



(a) Flow f is originally forwarded on its shortest path using OpenFlow.



(b) Rerouting flow f with multiple controllers and per-flow routing.



(c) Rerouting flow f with HybridFlow.

OpenFlow table

Priority	Source	Destination	Actions
high	S1	S6	towards S3
low	*	*	routing table lookup

Routing table

Destination	Next Hop
S6	S5

S2

OpenFlow table

Priority	Source	Destination	Actions
high	S1	S6	towards S7
low	*	*	routing table lookup

Routing table

Destination	Next Hop
S6	S2

S3

(d) Tables of $S2$ and $S3$ in (c).

Outline

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Outline

Crucial Flow Selection Module

Calculating Variation Slopes

- $lc(i)$ is used to denote the combination of i links with maximum amount of unique flows
- A Variation Slope denotes the amount of flows in one unit of traffic load

$$\text{VariationSlope(i+1)} = \frac{lc(i+1)^{flow\ amount} - lc(i)^{flow\ amount}}{lc(i+1)^{load} - lc(i)^{load}}$$

Selecting Crucial Flows

- Finding the local extreme point $(i+1)$ among the neighbor Variation Slopes
- $lc(i+1)$ is the combination of crucial links, and all flows traversing crucial links are crucial flows

Algorithm 1 CrucialFlowSelection()

Input: Link set E, traffic matrix M;
Output: Crucial flows set $F^{crucial}$;

for i = 2 : $|E|$ **do**
  calculate VariationSlope(i-1), calculate VariationSlope(i), calculate VariationSlope(i+1);
  if VariationSlope(i-1)<VariationSlope(i) and VariationSlope(i+1)< VariationSlope(i) **then**
    $F^{crucial} = lc(i)^{flows}$;
    break;
  end if
end for
return $F^{crucial}$;

Crucial Flow Selection Algorithm.

Crucial Flow Selection Module

Calculating Variation Slopes

- $lc(i)$ is used to denote the combination of i links with maximum amount of unique flows
- A Variation Slope denotes the amount of flows in one unit of traffic load

$$\text{VariationSlope(i+1)} = \frac{lc(i+1)^{flow\ amount} - lc(i)^{flow\ amount}}{lc(i+1)^{load} - lc(i)^{load}}$$

Selecting Crucial Flows

- Finding the local extreme point $(i+1)$ among the neighbor Variation Slopes
- $lc(i+1)$ is the combination of crucial links, and all flows traversing crucial links are crucial flows

Algorithm 1 CrucialFlowSelection()

Input: Link set E, traffic matrix $\mathbf{M}$;
Output: Crucial flows set $F^{crucial}$;

```
for i = 2 : |E| do
    calculate VariationSlope(i-1), calculate VariationSlope(i), calculate VariationSlope(i+1);
    if VariationSlope(i-1)<VariationSlope(i) and VariationSlope(i+1)< VariationSlope(i) then
        F^crucial = lc(i)^flows;
        break;
    end if
end for
return F^crucial;
```

Crucial Flow Selection Algorithm.

Outline

北京理工大學 BEIJING INSTITUTE OF TECHNOLOGY

Crucial Flow Rerouting Module

Path constraint

- One flow can be only forwarded on one path

$$\sum_{p \in P_f} y_f^p = 1, \forall f \in F^{crucial} \quad (1)$$

Link constraint

- Each link's load should not exceed link capacity C

$$load^e = \sum_{f \in F^{crucial}} \sum_{p \in P_f} (\delta^{p,e} * r_f * y_f^p) + C^e \quad (2)$$

$$load^e \leq C, \forall e \in E \quad (3)$$

Flow table constraint

- Flow table's utilization cannot exceed its flow table capacity T

$$\sum_{f \in F^{crucial}} \sum_{p \in P_f} (\xi^{p,v} * \alpha^{p,v} * y_f^p) + T_v \leq T, v \in V \quad (4)$$

Algorithm 2 CrucialFlowRerouting()

Input:
$E^{crucial}$: the set of crucial links; $F^{crucial}$: the set of flows on crucial links;
P: the path-sets of flows on crucial links;
Output:
$\mathcal{Y}$: the selected paths of rerouted flows $\{f \in F^{crucial} | \mathcal{Y}_f\}$, where $\mathcal{Y}_f = \{p \in P_f | y_f^p = 1\}$;

1: generate vector $\mathcal{Y}^* = \{y_k, k \in [1, \sum_{e \in E^{crucial}} \sum_{f \in F^{crucial}} |P_f|]\}$;
2: **for** $y_k \in \mathcal{Y}^*$ **do**
3: get y_k's flow f, path p, and flow f's old path p_0;
4: **if** $f \in \mathcal{X}$ **then** // test Equation (1)
5: continue;
6: **end if**
7: **for** $e \in p$ **do** // test Equation (3)
8: **if** $C^e + \delta^{p,e} * r_f * y_f^p > C$ **then**
9: go to line 2;
10: **end if**
11: **end for**
12: **for** $v \in p$ **do** // test Equation (4)
13: **if** $T_v + \alpha^{p,v} * y_f^p > T$ **then**
14: go to line 2;
15: **end if**
16: **end for**
17: $\mathcal{Y}_f = \{y_f^p = 1\}, \mathcal{Y} = \mathcal{Y} \cup \mathcal{Y}_f$, update link utilization for $e \in \{p, p_0\}$;
18: update flow table utilization for $v \in \{p, p_0\}$, remove f from $F^{crucial}$;
19: **if** $F^{crucial} == \varnothing$ **then**
20: break;
21: **end if**
22: **end for**
23: return $\mathcal{X}$, $\mathcal{Y}$

Crucial Flow Rerouting Algorithm.

Crucial Flow Rerouting Module

Path constraint

- One flow can be only forwarded on one path

$$\sum_{p \in P_f} y_f^p = 1, \forall f \in F^{crucial} \tag{1}$$

Link constraint

- Each link's load should not exceed link capacity C

$$load^e = \sum_{f \in F^{crucial}} \sum_{p \in P_f} (\delta^{p,e} * r_f * y_f^p) + C^e \tag{2}$$

$$load^e \le C, \forall e \in E \tag{3}$$

Flow table constraint

- Flow table's utilization cannot exceed its flow table capacity T

$$\sum_{f \in F^{crucial}} \sum_{p \in P_f} (\xi^{p,v} * \alpha^{p,v} * y_f^p) + T_v \le T, v \in V \tag{4}$$

Algorithm 2 CrucialFlowRerouting()

Input:
$E^{crucial}$: the set of crucial links; $F^{crucial}$: the set of flows on crucial links;
P: the path-sets of flows on crucial links;
Output:
$\mathcal{Y}$: the selected paths of rerouted flows $\{f \in F^{crucial} | \mathcal{Y}_f\}$, where $\mathcal{Y}_f = \{p \in P_f | y_f^p = 1\}$;

1: generate vector $\mathcal{Y}^* = \{y_k, k \in [1, \sum_{e \in E^{crucial}} \sum_{f \in F^{crucial}} |P_f|]\}$;
2: **for** $y_k \in \mathcal{Y}^*$ **do**
3: get y_k's flow f, path p, and flow f's old path p_0;
4: **if** $f \in \mathcal{X}$ **then** // test Equation (1)
5: continue;
6: **end if**
7: **for** $e \in p$ **do** // test Equation (3)
8: **if** $C^e + \delta^{p,e} * r_f * y_f^p > C$ **then**
9: go to line 2;
10: **end if**
11: **end for**
12: **for** $v \in p$ **do** // test Equation (4)
13: **if** $T_v + \alpha^{p,v} * y_f^p > T$ **then**
14: go to line 2;
15: **end if**
16: **end for**
17: $\mathcal{Y}_f = \{y_f^p = 1\}, \mathcal{Y} = \mathcal{Y} \cup \mathcal{Y}_f$, update link utilization for $e \in \{p, p_0\}$;
18: update flow table utilization for $v \in \{p, p_0\}$, remove f from $F^{crucial}$;
19: **if** $F^{crucial} == \varnothing$ **then**
20: break;
21: **end if**
22: **end for**
23: return $\mathcal{X}$, $\mathcal{Y}$

Crucial Flow Rerouting Algorithm.

Crucial Flow Rerouting Module

北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

Path constraint

- One flow can be only forwarded on one path

$$\sum_{p \in P_f} y_f^p = 1, \forall f \in F^{crucial} \quad (1)$$

Link constraint

- Each link's load should not exceed link capacity C

$$load^e = \sum_{f \in F^{crucial}} \sum_{p \in P_f} (\delta^{p,e} * r_f * y_f^p) + C^e \quad (2)$$

$$load^e \leq C, \forall e \in E \quad (3)$$

Flow table constraint

- Flow table's utilization cannot exceed its flow table capacity T

$$\sum_{f \in F^{crucial}} \sum_{p \in P_f} (\xi^{p,v} * \alpha^{p,v} * y_f^p) + T_v \leq T, v \in V \quad (4)$$

Algorithm 2 CrucialFlowRerouting()

Input:
$E^{crucial}$: the set of crucial links; $F^{crucial}$: the set of flows on crucial links;
P: the path-sets of flows on crucial links;

Output:
$\mathcal{Y}$: the selected paths of rerouted flows $\{f \in F^{crucial} | \mathcal{Y}_f\}$, where $\mathcal{Y}_f = \{p \in P_f | y_f^p = 1\}$;

1: generate vector $\mathcal{Y}^* = \{y_k, k \in [1, \sum_{e \in E^{crucial}} \sum_{f \in F^{crucial}} |P_f|]\}$;
2: **for** $y_k \in \mathcal{Y}^*$ **do**
3: get y_k's flow f, path p, and flow f's old path p_0;
4: **if** $f \in \mathcal{X}$ **then** // test Equation (1)
5: continue;
6: **end if**
7: **for** $e \in p$ **do** // test Equation (3)
8: **if** $C^e + \delta^{p,e} * r_f * y_f^p > C$ **then**
9: go to line 2;
10: **end if**
11: **end for**
12: **for** $v \in p$ **do** // test Equation (4)
13: **if** $T_v + \alpha^{p,v} * y_f^p > T$ **then**
14: go to line 2;
15: **end if**
16: **end for**
17: $\mathcal{Y}_f = \{y_f^p = 1\}, \mathcal{Y} = \mathcal{Y} \cup \mathcal{Y}_f$, update link utilization for $e \in \{p, p_0\}$;
18: update flow table utilization for $v \in \{p, p_0\}$, remove f from $F^{crucial}$;
19: **if** $F^{crucial} == \varnothing$ **then**
20: break;
21: **end if**
22: **end for**
23: return $\mathcal{X}$, $\mathcal{Y}$

Crucial Flow Rerouting Algorithm.

Crucial Flow Rerouting Module

Objective function

- The link load balancing performance is measured by the maximum utilization of links in the network

$$obj_1 = \max_{e \in E}(u^e) \tag{5}$$

- The total link load of links in the network

$$obj_2 = \sum_{e \in E} load^e \tag{6}$$

Problem formulation

$$\min_{y}\{\max_{e \in E}(u^e) + \lambda * \sum_{e \in E} load^e\} \tag{P}$$

subject to

Eqs. (1), (3), (4)

Algorithm 2 CrucialFlowRerouting()

Input:
$E^{crucial}$: the set of crucial links; $F^{crucial}$: the set of flows on crucial links;
P: the path-sets of flows on crucial links;
Output:
$\mathcal{Y}$: the selected paths of rerouted flows $\{f \in F^{crucial} | \mathcal{Y}_f\}$, where $\mathcal{Y}_f = \{p \in P_f | y_f^p = 1\}$;

1: generate vector $\mathcal{Y}^* = \{y_k, k \in [1, \sum_{e \in E^{crucial}} \sum_{f \in F^{crucial}} |P_f|]\}$;
2: **for** $y_k \in \mathcal{Y}^*$ **do**
3: get y_k's flow f, path p, and flow f's old path p_0;
4: **if** $f \in \mathcal{X}$ **then** // test Equation (1)
5: continue;
6: **end if**
7: **for** $e \in p$ **do** // test Equation (3)
8: **if** $C^e + \delta^{p,e} * r_f * y_f^p > C$ **then**
9: go to line 2;
10: **end if**
11: **end for**
12: **for** $v \in p$ **do** // test Equation (4)
13: **if** $T_v + \alpha^{p,v} * y_f^p > T$ **then**
14: go to line 2;
15: **end if**
16: **end for**
17: $\mathcal{Y}_f = \{y_f^p = 1\}, \mathcal{Y} = \mathcal{Y} \cup \mathcal{Y}_f$, update link utilization for $e \in \{p, p_0\}$;
18: update flow table utilization for $v \in \{p, p_0\}$, remove f from $F^{crucial}$;
19: **if** $F^{crucial} == \varnothing$ **then**
20: break;
21: **end if**
22: **end for**
23: return $\mathcal{X}$, $\mathcal{Y}$

Crucial Flow Rerouting Algorithm.

Crucial Flow Rerouting Module

Objective function

- The link load balancing performance is measured by the maximum utilization of links in the network

$$obj_1 = \max_{e \in E}(u^e) \tag{5}$$

- The total link load of links in the network

$$obj_2 = \sum_{e \in E} load^e \tag{6}$$

Problem formulation

$$\min_{y}\{\max_{e \in E}(u^e) + \lambda * \sum_{e \in E} load^e\} \tag{P}$$

subject to

Eqs. (1), (3), (4)

Algorithm 2 CrucialFlowRerouting()

Input:
$E^{crucial}$: the set of crucial links; $F^{crucial}$: the set of flows on crucial links;
P: the path-sets of flows on crucial links;
Output:
$\mathcal{Y}$: the selected paths of rerouted flows $\{f \in F^{crucial} | \mathcal{Y}_f\}$, where $\mathcal{Y}_f = \{p \in P_f | y_f^p = 1\}$;

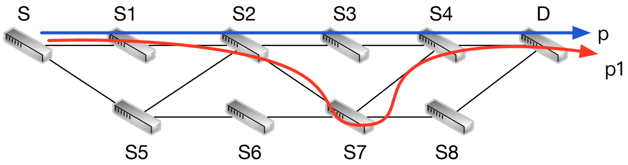
1: generate vector $\mathcal{Y}^* = \{y_k, k \in [1, \sum_{e \in E^{crucial}} \sum_{f \in F^{crucial}} |P_f|]\}$;
2: **for** $y_k \in \mathcal{Y}^*$ **do**
3:     get y_k's flow f, path p, and flow f's old path p_0;
4:     **if** $f \in \mathcal{X}$ **then** // test Equation (1)
5:         continue;
6:     **end if**
7:     **for** $e \in p$ **do** // test Equation (3)
8:         **if** $C^e + \delta^{p,e} * r_f * y_f^p > C$ **then**
9:             go to line 2;
10:         **end if**
11:     **end for**
12:     **for** $v \in p$ **do** // test Equation (4)
13:         **if** $T_v + \alpha^{p,v} * y_f^p > T$ **then**
14:             go to line 2;
15:         **end if**
16:     **end for**
17:     $\mathcal{Y}_f = \{y_f^p = 1\}, \mathcal{Y} = \mathcal{Y} \cup \mathcal{Y}_f$, update link utilization for $e \in \{p, p_0\}$;
18:     update flow table utilization for $v \in \{p, p_0\}$, remove f from $F^{crucial}$;
19:     **if** $F^{crucial} == \varnothing$ **then**
20:         break;
21:     **end if**
22: **end for**
23: return $\mathcal{X}$, $\mathcal{Y}$

Crucial Flow Rerouting Algorithm.

Outline

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

Routing Policy Generation Module



p: S->S1->S2->S3->S4->D, p1:S->S1->S2->S7->S4->D

⇩

overlapped path = p1∩p = {S->S1->S2,S4->D}

OpenFlow path = {S2->S7->S4}

⇩

Flow entry on S2: priority: high, match field: source=S, destination=D, action: towards S7;

Flow entry on S7: priority: high, match field: source=S, destination=D, action: towards S4

An example of routing policy generation module.

Algorithm 3 RoutingPolicyGeneration()

Input:
$\mathcal{Y}$: the selected paths of rerouted flows $\{f \in F^{crucial} | \mathcal{Y}_f\}$;
P^{dest_f}: the set of shortest paths with destination $dest_f$;
Output:
$\mathcal{Z}$: the set of generated flow entries;

1: $\mathcal{Z} = \emptyset$;
2: **for** $y \in \mathcal{Y}$ **do**
3: get y's flow f, $p_f = y$, flow f's source src_f, and destination $dest_f$;
4: **for** $p_i \in P^{dest_f}$ **do**
5: $P_i^{overlap} = p_f \cap p_i = \{p_i^{overlap_1} : v_{s_1} \rightarrow ... \rightarrow v_{d_1}, ..., p_i^{overlap_j} : v_{s_j} \rightarrow ... \rightarrow v_{d_j}\}$;
6: **if** $P_i^{overlap} == p_f$ **then**
7: go to line 1;
8: **else if** $P_i^{overlap} == \emptyset$ **then**
9: continue;
10: **end if**
11: remove paths in $P_i^{overlap}$ from p_f except $v_{d_1}, v_{s_2}, ..., v_{d_{j-1}}, v_{s_j}$;
12: **end for**
13: **for** $\{v_{i1} \rightarrow v_{i2}\} \in p_f$ **do**
14: generate flow entry $z(v_{i1}) =$ {priority: $high$, match field: source=src_f, destination=$dest_f$, action: towards v_{i2}} on v_{i1};
15: $\mathcal{Z} = \mathcal{Z} \cup z(v_{i1})$;
16: **end for**
17: **end for**
18: return $\mathcal{Z}$

Routing Policy Generation Algorithm.

Outline

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

Evaluation

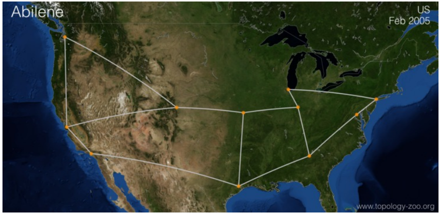
北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

Simulation setup

- Abilene topology with 12 nodes and 30 links
- Time slots: 5 minutes
- 8064 traffic matrices (4 weeks)

Comparison algorithms

- Open Shortest Path First (**OSPF**)
- Equal-Cost Multi-Path (**ECMP**)
- Crucial Flow Hybrid Routing (**CFHR**)
- **HybridFlow**
- **HybridFlow(+1)**
- Maximizing the utilization of links (**MaxU**)
- Multiple Traffic Matrix Load Balancing (**MTMLB**)



Abilene topology.

Evaluation

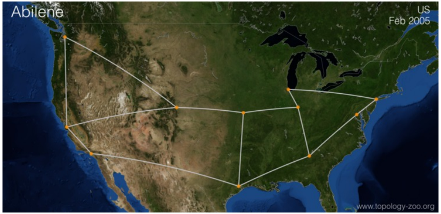
BEIJING INSTITUTE OF TECHNOLOGY

Simulation setup

- Abilene topology with 12 nodes and 30 links
- Time slots: 5 minutes
- 8064 traffic matrices (4 weeks)

Comparison algorithms

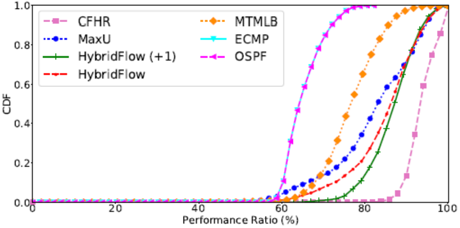
- Open Shortest Path First (**OSPF**)
- Equal-Cost Multi-Path (**ECMP**)
- Crucial Flow Hybrid Routing (**CFHR**)
- **HybridFlow**
- **HybridFlow(+1)**
- Maximizing the utilization of links (**MaxU**)
- Multiple Traffic Matrix Load Balancing (**MTMLB**)



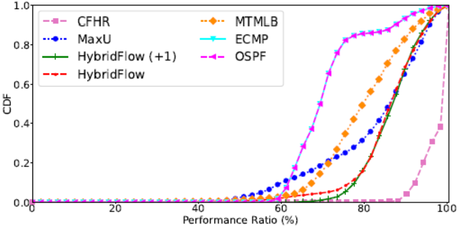
Abilene topology.

Load balancing performance

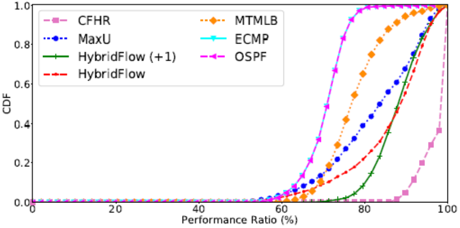
北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY



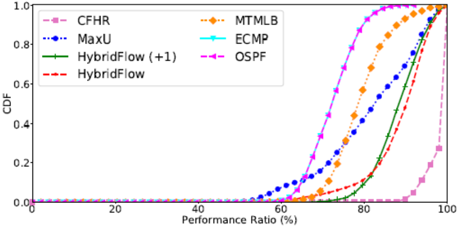
(a) CDF of performance ratio of traffic matrix week one.



(b) CDF of performance ratio of traffic matrix week two.

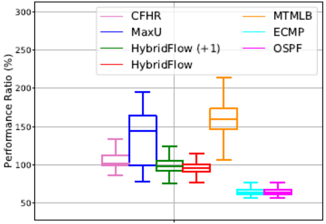


(c) CDF of performance ratio of traffic matrix week three.

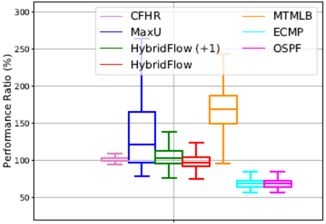


(d) CDF of performance ratio of traffic matrix week four.

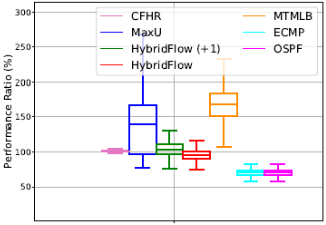
Total link load performance



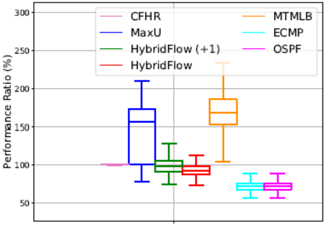
(a) Total link load of traffic matrix week one.



(b) Total link load of traffic matrix week two.

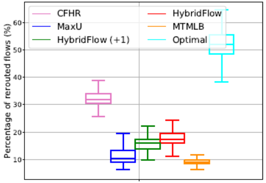


(c) Total link load of traffic matrix week three.

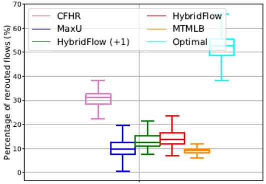


(d) Total link load of traffic matrix week four.

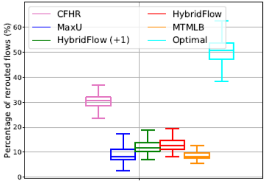
Performance of rerouted flows



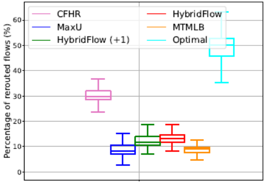
(a) The ratio of the number of rerouted flows to the total number of flows of traffic matrix week one.



(b) The ratio of the number of rerouted flows to the total number of flows of traffic matrix week two.

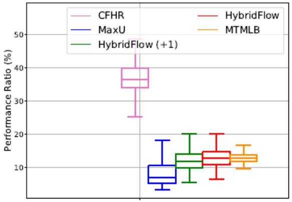


(c) The ratio of the number of rerouted flows to the total number of flows of traffic matrix week three.

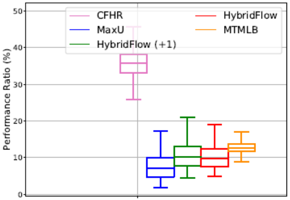


(d) The ratio of the number of rerouted flows to the total number of flows of traffic matrix week four.

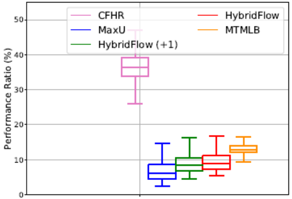
Controller’s processing overhead



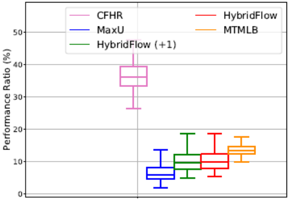
(a) Number of control messages for flow rerouting of traffic matrix week one.



(b) Number of control messages for flow rerouting of traffic matrix week two.



(c) Number of control messages for flow rerouting of traffic matrix week three.



(d) Number of control messages for flow rerouting of traffic matrix week four.

Outline

Summary

New idea

- We propose **HybridFlow**[1] to realize good load balancing performance with low processing overhead by dynamically **identifying crucial flows** and rerouting them on forwarding paths configured with the **hybrid routing**.

New problem and solution

- We formulate the crucial flow rerouting as the **CFHR problem** with the objective of achieving the optimal load balancing performance under given resource constraints and propose a **heuristic algorithm** to efficiently solve the problem.

Good performance

- The simulation based on the real traffic traces and network topologies shows that compared with the optimal solution, HybridFlow can achieve **near optimal load balancing performance** by **rerouting less flows** on average.

[1] Under review for possible publication in **IEEE Transactions on Communications**, Major Revision

Summary

New idea

- We propose **HybridFlow**[1] to realize good load balancing performance with low processing overhead by dynamically **identifying crucial flows** and rerouting them on forwarding paths configured with the **hybrid routing**.

New problem and solution

- We formulate the crucial flow rerouting as the **CFHR problem** with the objective of achieving the optimal load balancing performance under given resource constraints and propose a **heuristic algorithm** to efficiently solve the problem.

Good performance

- The simulation based on the real traffic traces and network topologies shows that compared with the optimal solution, HybridFlow can achieve **near optimal load balancing performance** by **rerouting less flows** on average.

[1] Under review for possible publication in **IEEE Transactions on Communications**, Major Revision

Summary

北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

New idea

- We propose **HybridFlow**[1] to realize good load balancing performance with low processing overhead by dynamically **identifying crucial flows** and rerouting them on forwarding paths configured with the **hybrid routing**.

New problem and solution

- We formulate the crucial flow rerouting as the **CFHR problem** with the objective of achieving the optimal load balancing performance under given resource constraints and propose a **heuristic algorithm** to efficiently solve the problem.

Good performance

- The simulation based on the real traffic traces and network topologies shows that compared with the optimal solution, HybridFlow can achieve **near optimal load balancing performance** by **rerouting less flows** on average.

[1] Under review for possible publication in **IEEE Transactions on Communications**, Major Revision

Summary

BEIJING INSTITUTE OF TECHNOLOGY

New idea

- We propose **HybridFlow**[1] to realize good load balancing performance with low processing overhead by dynamically **identifying crucial flows** and rerouting them on forwarding paths configured with the **hybrid routing**.

New problem and solution

- We formulate the crucial flow rerouting as the **CFHR problem** with the objective of achieving the optimal load balancing performance under given resource constraints and propose a **heuristic algorithm** to efficiently solve the problem.

Good performance

- The simulation based on the real traffic traces and network topologies shows that compared with the optimal solution, HybridFlow can achieve **near optimal load balancing performance** by **rerouting less flows** on average.

[1] Under review for possible publication in **IEEE Transactions on Communications**, Major Revision

Thank you for your attention!

Q&A

songshidou@hotmail.com