Introduction to Policy-based Networking

■ What is policy-based Networking?
  ◆ Network node configurations are vendor- and/or device-specific.
  ◆ Policy-based networking replaces such configuration methods by a unified (and standard-based) method.

■ What is a policy?
  ◆ Policy rule: a condition-action rule
    ■ if condition then action
  ◆ Policy: a list of policy rules.
    ■ \{ rule_1, rule_2, ..., rule_n \}
Problem: Complexity of Policy Transformation

- Human operators handle high-level policies.
- “Low-level policies” must be deployed to network nodes.
- Transformation from high- to low-level policies may be complex; i.e., it is not necessarily one-to-one.
  - Routers, especially high-performance routers, require specific forms of policies (commands).

Transformation Types: Policy Division and Fusion

- **Policy division**
  - A transformation of a high-level policy into two or more low-level policies.

  ![Diagram of Policy Division]

- **Policy fusion**
  - A transformation of two or more high-level policies into one low-level policy.

  ![Diagram of Policy Fusion]

- **A combination of policy division and fusion**

  ![Diagram of Combined Policy Division and Fusion]
Policy Division: Example

**Input: Marking and priority queuing policy for Diffserv**

- EC2 = { if (Source_IP is 192.168.1.1) {
  DSCP = "EF"; Priority = "High"; },
  if (true) {
    DSCP = "BE"; Priority = "Low"; }
}. 

**Output: Marking policy and queuing policy**

- E2 = { if (Source_IP is 192.168.1.1) { DSCP = "EF"; },
  if (true) { DSCP = "BE"; }
}. 

- C2 = { if (Source_IP is 192.168.1.1) { Priority = "High"; },
  if (true) { Priority = "Low"; }
}. 

Each rule is divided into two rules.

---

Policy Division: Example (cont’d)

**Input: Marking and priority queuing policy for Diffserv**

- EC2 = { if (Source_IP is 192.168.1.1) {
  DSCP = "EF"; Priority = "High"; },
  if (true) {
    DSCP = "BE"; Priority = "Low"; }
}. 

**Output: Marking policy and queuing policy**

- E2 = { if (Source_IP is 192.168.1.1) { DSCP = "EF"; },
  if (true) { DSCP = "BE"; }
}. 

- C2 = { if (Source_IP is 192.168.1.1) { Priority = "High"; },
  if (true) { Priority = "Low"; }
}. 

Conditions are copied.
Policy Division: Example (cont’d)

- **Input**: Marking and priority queuing policy for Diffserv
  - EC2 = { if (Source_IP is 192.168.1.1) {
    - DSCP = "EF"; Priority = "High";
    - if (true) {
        - DSCP = "BE"; Priority = "Low";
    }
  }.

- **Output**: Marking policy and queuing policy
  - E2 = { if (Source_IP is 192.168.1.1) { DSCP = "EF"; },
    - if (true) { DSCP = "BE"; }
  }.
  - C2 = { if (Source_IP is 192.168.1.1) { Priority = "High"; },
    - if (true) { Priority = "Low"; }
  }.

Actions are divided.

How complex?: Restrictions on Policy Division

- **Restrictions on data reference and marking**
  - The naive transformation must be inhibited
    - if rules in the high-level policy refer to field in the packet, and
    - if this rule or another rule writes the same field

Wrong division example

- **Input E’**
  - e1: if (DSCP is 14) DSCP = 10
  - e2: if (…) DSCP = 14

- **Output F’**
  - f1: if (DSCP is 14) DSCP = 10
  - f2: if (…) DSCP = 14

Problem 1: Rule m1 fails to catch this flow
Problem 2: Rule m1 wrongly catches this flow

Reference to a DSCP
Remarking of the DSCP
Elimination of the restrictions by using VFLs

**Introduction of virtual flow labels (VFLs)**
- A VFL is a label attached to a packet or flow.
- A VFL is similar to a DSCP but it exists outside the packet.

**Policy division using VFLs**
- The restrictions can be eliminated by introducing VFLs in a policy division. (See [Kan 01b] for detail.)

```
e1: if (DSCP is 14) DSCP = 10
e2: if (...) DSCP = 14
```

```
f1: if (DSCP is 14) {
    DSCP = 10; VFL = "m1";
}
f2: if (...) {
    DSCP = 14; VFL = "m2";
}
```

```
m1: if (VFL is "m1") …
m2: if (VFL is "m2") …
```

---

Method of Software-Hardware Integration for Policy-based QoS

**Restrictions of policy division can be resolved by a software-hardware integration.**
- Hardware-based VFLs (called flow IDs) are introduced into routers.
- Policy division with VFLs are implemented in policy agents.
Prototype Development for Diffserv Policies

- **Diffserv policies in PolicyXpert™ were implemented for a gigabit router.**
  - PolicyXpert™ is a QoS policy server developed by Hewlett Packard and Hitachi.

- **Diffserv policies in PolicyXpert™ sometimes require policy division and/or fusion.**
  - These transformations enable flexible use of Diffserv policies. (not strictly necessary)

- **The restrictions are going to be eliminated by a software-hardware integration.**
  - VFLs (called flow IDs) were implemented by hardware.
  - A policy agent that uses flow IDs is going to be developed.

A VFL Function Implemented in Hardware

- **Two filter blocks and flow IDs (VFLs)**

  ![Diagram of VFL Function](image)

  - Input packet
  - Packet and Flow ID (VFL)
  - Router
  - Inbound interface
  - Filter block 1
  - Actions 1
  - Filter classification
  - Filter block 2
  - Actions 2
  - Crossbar switch
  - Filter block 1
  - Output packet
  - Flow ID (VFL)
  - Outbound interface
  - Packet and Flow ID (VFL)
  - Actions 1
  - Filter classification
  - Actions 2
  - Filter classification
  - Output packet
Policy Transformation for the Elimination

- Instead of copying conditions, flow IDs are used.

- Example
  
  ◆ Input
  ```java
  EC2 = { if (Source_IP is 192.168.1.1) {
    DSCP = "EF"; Priority = "High";
  },
  if (true) {
    DSCP = "BE"; Priority = "Low";
  } }
  ```

  ◆ Output
  ```java
  E2' = { if (Source_IP is 192.168.1.1) {
    Flow_ID = "EF_FID"; DSCP = "EF";
  },
  if (true) {
    Flow_ID = "BE_FID"; DSCP = "BE";
  } }
  ```

  - In addition to DSCP, flow IDs are set.

  ```java
  C2' = { if (Flow_ID is "EF_FID") { Priority = "High"; },
  if (Flow_ID is "BE_FID") { Priority = "Low"; }
  }
  ```

  - Instead of copying the conditions, flow-ID conditions are introduced.

Evaluation

- Performance of marking rules was measured.

- Method
  
  Policy F: 100 aggregation rules, the flows hit the 10th, 30th, 50th, 70th and 90th rules.
  Policy S: 5 marking rules.

  Five flow IDs connect rules in F and S.

  - Smartbit 6000B input and output rates were both measured to be 1.42 Mpps, i.e., no performance degradation occurred.
  - This means introduction of VFLs does not degrade the performance.
Conclusion

- We have developed a method of software-hardware integration for resolving the restrictions of policy division.
- We are developing a policy agent and a gigabit router integrated by using this method to support the Diffserv policies of PolicyXpert.
- A preliminary evaluation result shows that both high-performance and flexibility are achieved by this integration.

Policy Division: Example 2

- Input: a rule with flow aggregation
  
  \[ E_3 = \begin{cases} 
  \text{if (Source\_IP is 192.168.1.1 || Source\_IP is 192.168.1.3)} & \\
  \quad \text{if (Information\_Rate <= 1 Mbps)} \quad \{ \\
  \quad \quad \text{DSCP = "EF";} \\
  \quad \quad \} \text{else \{} \quad \text{absolute\_drop;} \quad \}; \quad \} \\
  \end{cases} \]

- Output
  
  \[ E_{31} = \begin{cases} 
  \text{if (Source\_IP is 192.168.1.1)} & \\
  \quad \{ \text{DSCP = "EF";} \}; \ \\
  \text{if (Source\_IP is 192.168.1.3)} & \\
  \quad \{ \text{DSCP = "EF";} \}; \ \}
  \end{cases} \]
  \[ E_{32} = \begin{cases} 
  \text{if (DSCP is "EF")} & \\
  \quad \{ \text{if (Information\_Rate > 1 Mbps)} \quad \{ \\
  \quad \quad \text{absolute\_drop;} \quad \}; \quad \} \\
  \end{cases} \]

This rule aggregates two flows.
Restrictions on Policy Division (cont’d)

Restrictions on flow aggregation

◆ If specific data is used for identifying an aggregated flow, flows that are not caught by any rule in F’ (called default flows) must be inhibited.

Example

\[
E':
\begin{align*}
\text{e: } & \text{ if (C1 OR C2) DSCP = 10} \\
\text{f1: } & \text{ if (C1) DSCP = 10} \\
\text{f2: } & \text{ if (C2) DSCP = 10} \\
\text{m: } & \text{ if (DSCP is 10) …}
\end{align*}
\]

Default flows

Policy Transformation for Resolution (cont’d)

Example 2

◆ Input

\[
E3 = \begin{cases}
\text{if (Source_IP is 192.168.1.1 ||} \\
\text{Source_IP is 192.168.1.3) } \\
\text{if (Information_Rate <= 1 Mbps) } \\
\text{DSCP = "EF";} \\
\text{else } \\
\text{absolute_drop;}
\end{cases}
\]

◆ Output

\[
E31 = \begin{cases}
\text{if (Source_IP is 192.168.1.1) } \\
\text{Flow_ID = “EF_FID”; } \\
\text{if (Source_IP is 192.168.1.3) } \\
\text{Flow_ID = “EF_FID”; }
\end{cases}
\]

- A flow ID is used twice.
- No need to set DSCP here. (This transformation is simpler.)

\[
E32 = \begin{cases}
\text{if (Flow_ID = “EF_FID”) } \\
\text{if (Information_Rate <= 1 Mbps) } \\
\text{DSCP = "EF";} \\
\text{else } \\
\text{absolute_drop;}
\end{cases}
\]