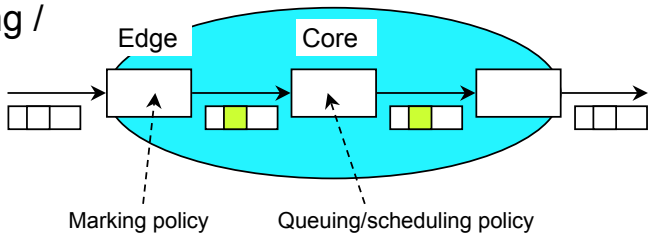
Taxonomy and Description of Policy Combination Methods

Yasusi Kanada
Hitachi Ltd., IP Network Research Center

What is a policy combination?

- **Policies may be mutually dependent.**
  - Negative dependence is called *conflicts*, and widely studied.
  - Polisitive depencence (e.g., cooperation) also exists.
- **A policy combination is**
  - An explicit specification of *positive* relationship between policies.
    - Definition in the paper: Combination of mutually dependent policies for a specific purpose.
- **An example in Diffserv** (Differentiated Services)
  - Edge routers mark a DSCP on packets, and the behavior (PHB) of core routers depend on the DSCP. (DSCP = Diffserv Codepoint)
  - Marking and queuing/scheduling may be controlled by policies.
  - A marking policy and a queuing / scheduling policy cooperate.
  - These policies are connected by DSCP.

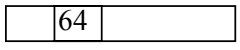


Two architectures for combining policies

- **Policies are rule-based programs in both architectures.**
  - They are programs because they control the network/node behavior.
- **Label-connection architecture**
  - A direct extension of policies are used.
  - They consists of if-then (condition-action) rules.
- **Pipe-connection architecture**
  - Resolution-based semantics is used (similar to “parallel logic languages”, such as Parlog, Concurrent Prolog, or GHC).
- **Label-connection architecture is currently more practical [Kanada 99]**
  - Several advantages.
  - The only implementable architecture by using currently available technology.
- **This talk focuses on label-connection architecture.**

Passing information between policies

- **Tags**
  - Pieces of information transferred between two policies.
- **Tags are classified into Real tags and virtual tags.**
  - Real tags
    - Tags that exist inside a packet.
    - E.g., DSCP
  - Virtual tags
    - Tags that exist outside a packet.
    - E.g., GMPLS label may be outside a packet.



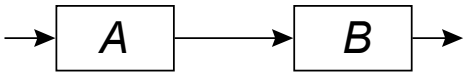
- **Tas are classified into Labels and attributes.**
  - Labels
    - Tags that are used for selecting a rule from a policy.
    - E.g., A DSCP may be used as a label.
  - Attributes
    - Tags that are not used for program control.
    - Used only for specifying actions.
    - E.g., queue priority

Types of policy combination — Local relationship

- **Four types**
  - Concatenation (sequential application)
  - Parallel application
  - Selection
  - Repetition
- **These types are similar to types of control structures in procedural programs.**
- **Why similar?**
  - Data dependences caused by tags are similar to those caused by variables in procedural programs.

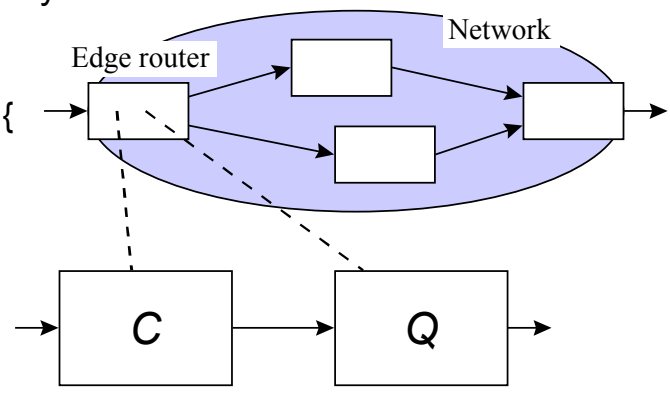
Concatenation

- **Two policies are sequentially applied:**



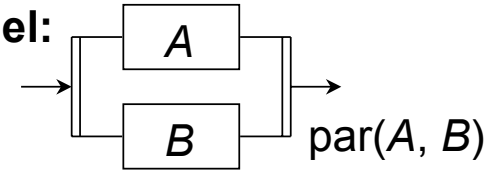
seq(*A*, *B*)

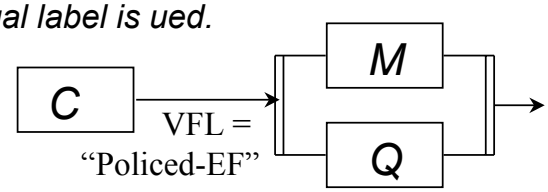
- **An example in Diffserv**
  - Classification and marking policy *C*
    - if (Source_IP is *x.x.x.x*) {
      DSCP = “EF”; }
    - else if (Source_IP is *y.y.y.y*) {
      DSCP = 0; /* DF */ }
  - Queuing policy *Q*
    - if (DSCP is “EF”) {
      Scheduling_Priority = 6;
      Enqueue; }
    - else {
      Scheduling_Priority = 1;
      Enqueue; }
  - The DSCP is used as a real label.



Parallel application

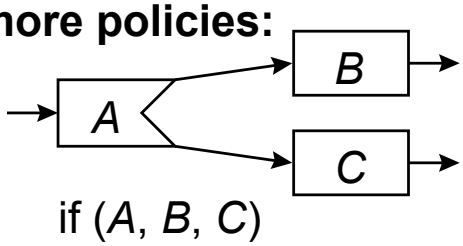
- **Two policies are applied in parallel:**
- **An example in Diffserv**
  - Classification policy *C*
    - if (Source_IP is *x.x.x.x*) {
      VFL = "Policed-EF"; } # *Virtual (flow) label is defined.*
      else {
      VFL = ""; }
  - Marking policy *M*
    - if (VFL is "Policed-EF") { # *Virtual label is ued.*
      DSCP = "EF"; }
    - else {
      DSCP = "DF"; }
  - Queuing policy *Q*
    - if (VFL is "Policed-EF") { # *Virtual label is ued.*
      Scheduling_Priority = 6;
      Enqueue; }
    - else {
      Scheduling_Priority = 1;
      Enqueue; }

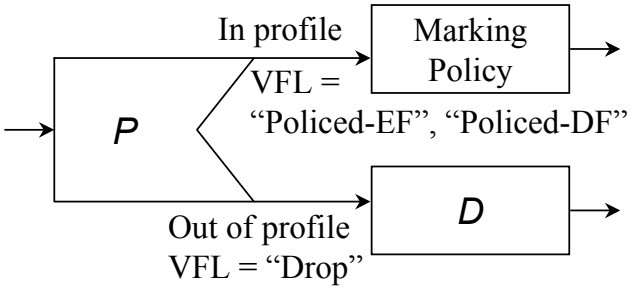




Selection

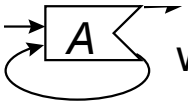
- **A relationship between three or more policies:**
  - Policy *A* outputs two types of results.
  - Policy *B* handles one of them.
  - Policy *C* handles the other.
- **An example in Diffserv**
  - Policing policy *P*
    - if (DSCP is "EF" && Information_Rate <= 2 Mbps) {
      VFL = "Policed-EF"; }
    - else if (DSCP is "EF") {
      VFL = "Drop"; }
    - else {
      VFL = "Policed-DF"; }
  - Dropping policy *D*
    - if (VFL is "Drop") {
      Absolute_Drop; }



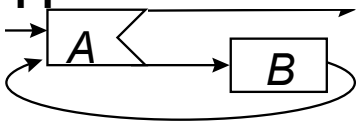


Repetition

- **The policies are repeatedly applied until a condition is met:**

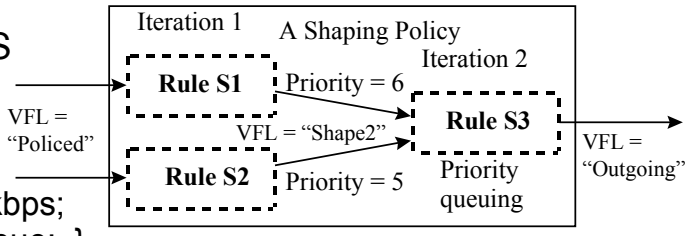


while(*A*) or

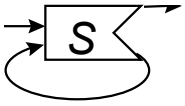


while(*A*, *B*)

- **An example in Diffserv**
  - Hierarchical shaping policy S
    - if (VFL is "Policed" &&
      DSCP is "EF") {
      Scheduling_Priority = 6;
      Maximum_Rate = 700 kbps;
      VFL = "Shape2"; Enqueue; }



    - else if (VFL is "Policed") { # except EF
      Scheduling_Priority = 5;
      Maximum _Rate = 500 kbps; # shaping rate
      VFL = "Shape2"; Enqueue; }



while(*S*)

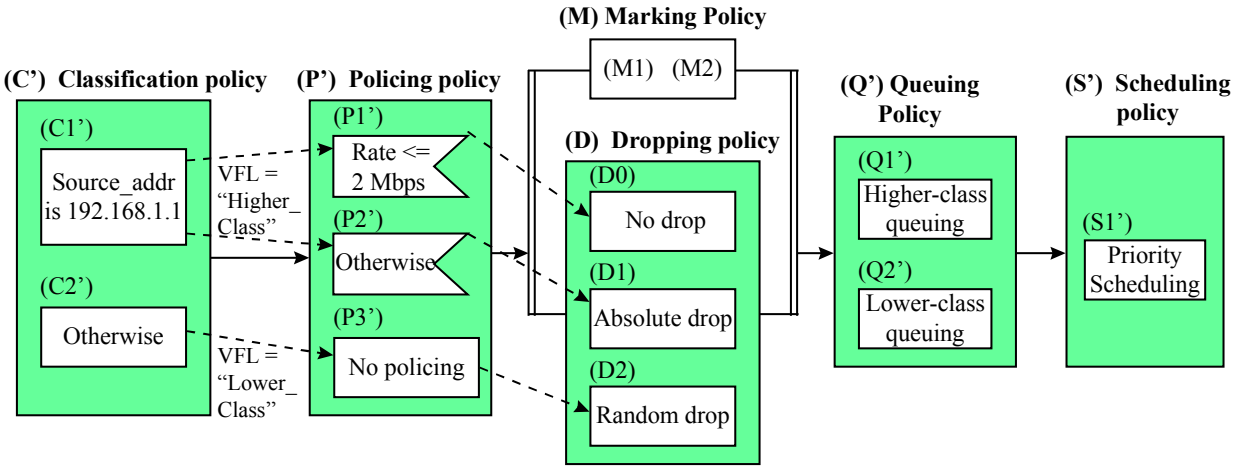
    - else if (VFL is "Shape2") {
      Scheduing_Algorithm = Priority_Queuing;
      Maximum_Rate = 1 Mbps; # shaping rate
      # 200 kbps (700 k + 500 k – 1 M) or less traffic may be dropped here.
      VFL = "Outgoing"; Enqueue; }

Methods of policy organization — Global structure

- **Homogeneous organization**
  - No compound policies are used.
  - The policies are organized such that all rules in a policy have the same type of conditions and the same type of actions.
- **Heterogeneous organization**
  - Other than homogeneous organization.

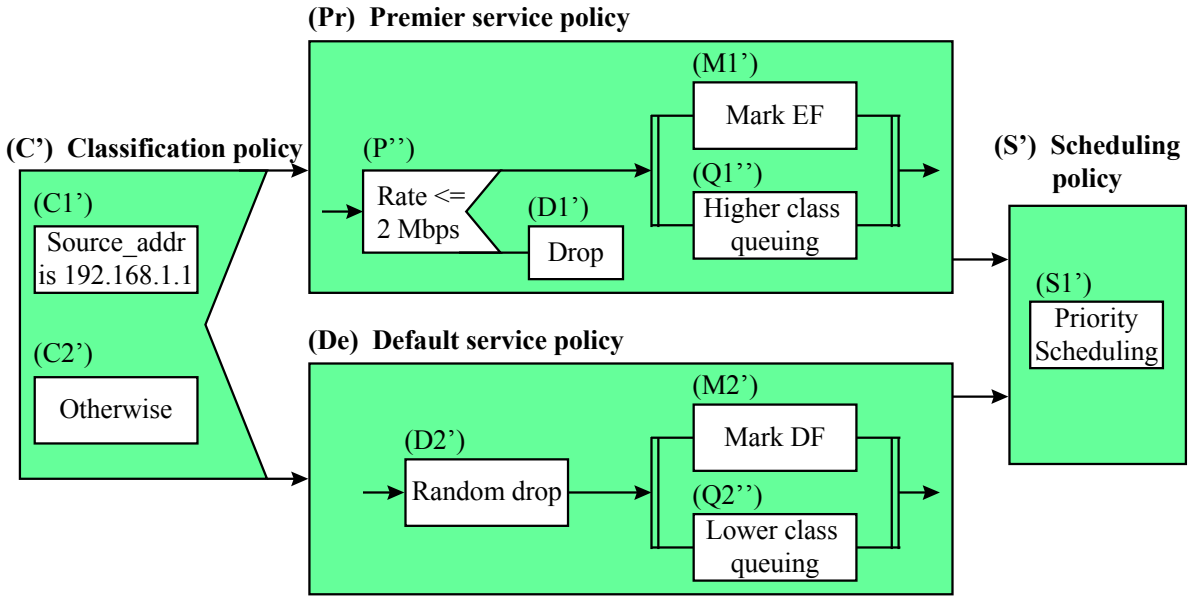
Homogeneous organization

■ Example in Diffserv



Heterogeneous organization

■ Example in Diffserv



Comparison of the policy-organization types

- **Homogeneous organization is more device-oriented.**
  - Because each policy in this organization may be implemented by a specific device function.
  - Each policy may be mapped to pipelined or SIMD packet procesing hardware.
  - Better suited to device control and performance management purposes.
- **Heterogeneous organization is more service-oriented.**
  - Because compound policies usually represent abstract functions.
  - Better suited to service management.

Discussion on policy-combination types

- **Semantics**
  - Policy semantics can be clarified by explicitly specifying policy combinations.
  - If not specified explicitly, a change of the application order may cause errorneous results.
- **General use**
  - If policy combination is not specified, policy usage is more restricted; e.g., the execution order must be predefined.
  - The policy system cannot be general-purpose.
- **Adaptation to devices**
  - If policy combination is specified, the policies may be adapted to a variety of devices.
- **Optimization**
  - Inefficient policies may have to be optimized.
  - If policy combination is specified, the possibility of optimizing policies is improved.

Summary

- **Four types of policy combination (local relationship) are defined:**
  - Concatenation
  - Parallel application
  - Selection
  - Repetition.
- **Advantages of specifying policy combination (global structure)**
  - The system becomes semantically clearer.
  - The system becomes better suited to general use.
  - The range of functionality becomes wider.
  - The possibility of policy optimization becomes improved.
- **Two types of organization**
  - Homogeneous organization is more device-oriented.
  - Heterogeneous organization is more service-oriented.

Future work

- **Implementation — two approaches**
  - To design a new policy language and systems
  - To embed policy combination specification into existing policy systems
- **Development of translation methods**
  - Policy division: Dividing a policy into two.
  - Policy fusion: Merging two or more policies into one.
  - Will be discussed in IM 2001.