Integrating Policy-based Management and Adaptive Traffic Engineering for QoS Deployment

Steven Van den Berghe
Filip De Turck
Piet Demeester

Policy 04
Goal

- Service Differentiation
  - Based on transport characteristics (~DiffServ)
  - Based on availability characteristics
  - Based on survivability characteristics
- Flexible: QoS-deployment with ‘classic’ features such as roaming, ...

Varying Demand (mobile users, access services)

Varying Network (QoS after Fast Reroute)
Approach

**QoS Resource Management:**
- Paths (MPLS) & Per Hop (DiffServ)
- Calculation (Dimensioning) and Implementation (provisioning)
- For different service types (session based & connection-less)

**Policy-based configuration methodology**
- PDP-PDP-PEP interaction:
  - Central decision unit for longer term dimensioning
  - Local (edge) decision unit for short term load-balancing
  - Network-wide implementation unit for, well, implementing the service (=RSVP-TE, DiffServ config + monitoring)
- Input = Demand Matrix, Output = DS/MPLS-config
- Optional explicit session activations (e.g. telephone calls)

**Adaptive:** use monitoring to perform load-balancing
SONAR Architecture

Service Overlay Networking using Adaptive Resource management

PDP
- Perform TE-algorithms

SIP proxy
- Manage QoS-sessions

Local (edge) PDP
- Add mappings, manage trunks

PEP: configuration
- Drive RSVP-TE
- Do Traffic Mapping
- Configure & Monitor DiffServ
Outline

- Introduction (done)
- SONAR Architecture: E-PDP

- Admission Control
  - Why we don’t need to invent new technologies/protocols
  - Registration/activation processes

- Conclusion
E-PDP architecture

- SIP
- Session Requests
- Traffic (re)-mapping
- E-PDP
- PDP
- Tunnels + weight
- Monitoring
- Implicit Session Request

(re)-mapping
E-PDP architecture

- PDP
- Tunnels + weight
- SIP
- Session Requests
- E-PDP
- Implicit Session Request
- Monitoring
- Traffic (re)-mapping
Traffic (Re-)Mapping

- PDP gives L-PDP a set of tunnels and weights
- Mapping depends on ingress classification mechanism
  - Flow-based classifier: need to translate into flow mappings
  - Hashing based classifier: can use weight as input *(future work)*
- E-LSP can be in 3 states: GREEN / YELLOW / RED

Monitoring

- Alarm Recvd
- Monitoring
- Alarm Recvd

No Alarm Recvd in $T_{cancel}$

- $alarmCnt--$
- if $alarmCnt < alarmMax$

RED

- $alarmCnt++$
- Monitoring
- Alarm Recvd

YELLOW

- $alarmCnt=1$
- Re-map traffic

GREEN

- Mon. wght > Pr. wght
- Mon. wght $\leq$ Pr. wght

- $alarmCnt=0$
- $alarmCnt=1$
new flow activation algorithm:
- \( \exists \) E-LSP in ‘green’-state: map flow to E-LSP (WRR)
- else \( \exists \) E-LSP in ‘yellow’-state: map flow to E-LSP (WRR)
- else block flow

Can also be running flow whose mapping gets ‘garbage-collected’ (cf. infra)
EPDP architecture

- SIP
- E-PDP
- PDP
- Tunnels + weight
- Session Requests
- Monitoring
- Implicit Session Request
- Traffic (re)-mapping
Implicit flow request

- route caching:
  - hashing-based lookup of <src,dst,DSCP,proto,srcprt,dstprt>
    - If found, use attached LSPid
    - If not found, get new LSPid (⇒ L-PDP must be embedded in slow-path of ingress)
  - garbage collecting on cache

- $T_{gc}$: cache garbage collecting time-out defined per service:
  - main importance: delays effect of re-mapping for running services
  - should be > average service duration for short sessions (e.g. WWW-browsing)
  - should be limited for other services (e.g. maximum of 2 re-mappings during a video-conferencing)
  - should be small enough to allow re-mapping to have effect
EPDP architecture

- SIP
- E-PDP
- PDP
- Tunnels + weight
- Monitoring
- Implicit Session Request
- Traffic (re)-mapping
- Session Requests
EPDP architecture

SIP → Session Requests → E-PDP

PDP → Tunnels + weight

Implicit Session Request

Traffic (re)-mapping

Monitoring
Explicit Service Level CAC

- For some services QoS = better no than bad connection
  - e.g. current telephony service
- Take time for explicit service invocation & CaC
  - Trade-off: invocation time vs. quality

Solution: take underlying LSP states into account
- for example:

\[
p(accepted) = 1 - \frac{\#lsp_{Yellow}}{\#lsp_{Yellow} + \#lsp_{Red}}
\]

\[
p(accepted) = \frac{\#lsp_{Green} + \#lsp_{Yellow}}{\#lsp_{total}}
\]

- Adds extra level of service differentiation !!!
  - Same DS class
  - Different application level CAC
Deployment Example

SLS = set of attributes

- Some discrete (DiffServ Classes $\rightarrow$ User Classes): eg. Residential vs. Business
- Some cont.: e.g. time

Roaming = define translation mechanism for SLSs
Decision = ingress-only
SIP as QoS-session control protocol

- SIP=well-known for IP-telephony
  - Registration of users at certain location
  - Management of sessions: create, end, ...

- But: is actually more generic (similar to http)
- Sufficient for QoS-session setup

- Note: architecture allows multiple mechanisms to be present as long as mechanism is the same
  - RSVP, Web Services, ....
Registration Process

1. **Registration**
2. **Visiting User**
   - Select Possible Services
   - Store User Info
   - OK

3. **ISP Info**
4. **Roaming Possible**
   - Yes
   - Translate Request
   - Translate Attributes between Providers

5. **Accepted?**
   - Yes
     - P1
       - U
       - XSL-T
     - P2
       - U
       - N
       - P2
       - D

Note: The diagram illustrates the process of registering a user, including selection of possible services, storing user information, and translating attributes between providers.
Activation Process

1. Activation

2. Network Status
   - get

3. User Info
   - get

4. Check Admission Variants
   - no

5. Select LSP
   - set

6. Explicit Invocation
   - yes

7. Alarm to User
   - set
Conclusion

SONAR=Adaptive Traffic Engineering Platform
- Multiclass/multipath
- Monitoring Feedback

Policy-based CAC:
- User requirements: attributes
- Network status: environment
- Ingress-only decision, allows roaming

Existing technologies: XSL-T+XACML+SIP
- Possible
- Optimal ???

Future work: further integration (e.g. OSGi)

User Differentiation
Orthogonal
To Traffic Differentiation