Handling Policy Conflicts in Call Control

Lynne Blair and Ken Turner
University of Stirling, Scotland
www.cs.stir.ac.uk/accent
29th June 2005
ACCENT Project Motivation

• increasing user connectivity:
  • any caller, any device
  • any location, any time

• wider definition of 'call':
  • PSTN, Internet
  • voicemail, speech recognition/synthesis
  • email, web service, multimedia session

• users need:
  • greater control over calls
  • greater flexibility in devices and locations
  • their own policies => conflicts!
Policy Handling — One Server

- suppose lynne calls ken on the same server:
  - policies for lynne as caller are retrieved and filtered for applicability
  - policies for ken as callee are retrieved and filtered for applicability
  - domain policies might also apply (e.g. outside hours calls go to voicemail)
- a complete set of policies is accumulated on the policy server
- conflicts are then detected and resolved
Policy Handling — Many Servers

- policies cannot be retrieved and accumulated in one place
- all servers involved write policies to a shared but trusted 'blackboard':
  - a temporary area in some policy store
  - one per call, but not one per network
  - selected by first server with enabled policies
  - resolution triggered by last policy server
- forwarding and forking complicate things:
  - actions like these must be deferred
  - all routes are thus potential in conflict handling
Conflict Detection/Resolution

- resolutions detect and resolve conflicts
- resolution policies:
  - are higher-level (like meta-policies)
  - define what conflict means in terms of conflicting actions (e.g. add vs. remove medium)
  - define what resolution is appropriate
  - are external and localised - not hard-wired into the policy system
  - are inherently multi-party (e.g. users, domains)
  - may be generic - choose which party prevails
  - may be specific - dictate what action to take
Resolution Policies

• resolution policies have almost the same structure as call policies

• resolution triggers are call actions - they define variables, preferences, timestamps

• resolution conditions use this information - in/out = preferences in/out of keeping

• generic resolutions prefer:
  • caller/callee, newer/older
  • negative/positive, superior/inferior, ...

• specific resolutions are call policy actions - may differ from the triggers
Media Conflict Resolution

- prefer not vs. should add video to a call

```xml
<triggers>
  <and/>
  <trigger arg1="var1">add_medium(arg1)
  <trigger arg1="var2">add_medium(arg1)
</triggers>

<conditions>
  <and/>
  <condition>
    <parameter>var1 <operator>eq <parameter>var2
  <condition>
    <parameter>pref1 <operator>out <value>pref2
</action>apply_negative
Call Rejection Conflict

- should vs. must not reject a call

```xml
<triggers>
  <and/>
  <trigger arg1="var1">reject_call(arg1)
  <trigger arg1="var2">reject_call(arg1)
</triggers>

<condition>
  <parameter>pref1</parameter>
  <operator>out</operator>
  <value>pref2</value>
  <action arg1="call conflict">reject_call(arg1)
```
Conclusion

• the policy system is:
  • user-oriented (policy wizard)
  • communications-independent (largely)
  • flexible (infinite variety of policies)
  • extensible (new application domains)

• conflict resolution:
  • external and localised (defined per domain)
  • generic or specific actions
  • currently implemented for single server
  • designed for multiple servers