Background
Pervasive Communication

• personal communication is increasing:
  • more pervasive and invasive
  • anywhere, any time, any how
  • land line, cell phone, email, voicemail, text messaging, instant messaging, social networks, ...

• the ACCENT initiative is empowering users to control their communications:
  • caller, callee
  • time, location
  • topic, role
  • technology, media
  • cost, quality ...
Telephony Features

• the traditional telephony solution is a feature:
  • call forwarding, call waiting, automatic call-back, ...
  • defined and supported by the network operator

• however, features lack flexibility:
  • low-level control
  • limited customisation
  • network-oriented, not user-oriented

• Internet telephony is growing (e.g. SIP, Skype):
  • the network deals only with signalling and transmission
  • advanced functionality can be placed in the endpoints
  • user customisation is much easier
Goals and Policies for Telephony

- goals/policies are written in the APPEL language
- goals are high-level objectives:
  - persistent, user-oriented
  - declarative rather than executable
  - can be realised through refinement to policies
- policies are high-level rules:
  - higher-level than features
  - infinitely customisable
  - executed in endpoints (customer equipment, servers)
  - typically in ECA form (Event, Condition, Action)
- hierarchical levels: goal → policy → feature
Policy Structure

- policy document:
  - ≥ 1 policies, variables, prototypes, resolutions, goals
- policy:
  - id, owner, target domain, ...
  - optional preference (must, should, prefer, must not, ...)  
  - ≥ 1 policy rules (combined with sequential, parallel, ...)
- policy rule:
  - ≥ 0 triggers (combined with and, or)
  - ≥ 0 conditions (combined with and, or, not)
  - ≥ 1 actions (combined with and, or, else, ...)
- internally stored as XML (abbreviated in this talk)
Notify Arrival Policy

<policy id='Tell me when Bob arrives'
  owner='ken@stir.ac.uk' ...>
<policy_rule>
  <trigger arg1='bob@stir.ac.uk'>present(arg1)
  <action arg1='ken@stir.ac.uk'
    arg2='Bob has arrived'>send_message(arg1,arg2)
No Emergency Forwarding Policy

<policy id='Never forward emergency calls' owner='ken@stir.ac.uk' ...
<preference>must_not
<policy_rule>
<trigger>connect
<condition>
<parameter>call_type
<operator>eq
[value]emergency
<action arg1=''forward to(arg1)
Policy Variables

• variables can be defined and used in policies:

```xml
<variable id='home' owner='ken@stir.ac.uk' value='1234567890' .../>
```

```xml
<policy ...>
    <action arg1=':home'>forward_to(arg1)
```

Goals and Policies for Telephony (C) K. J. Turner, University of Stirling
Prototype Policies

• prototype policies are like regular policies:
  • used to realise goals dynamically
  • high-level effects contribute to goals
  • optimal parameter values can be determined
Add Video Prototype

<prototype id='Add video' owner='ken@stir.ac.uk' effect='call_bandwidth += 512' ...>
<policy_rule>
<trigger>connect_incoming
<condition>
<parameter>medium
<operator>eq
<value>audio
<action arg1='video'>add_medium(arg1)
Resolution Policies

• resolution policies resemble regular policies:
  • used to detect and resolve policy (action) conflicts
  • triggers are actions of regular policies
  • conditions and actions can use the preferences and variables of conflicting policies
  • operator in for preferences means ‘in keeping with’ (i.e. in a similar sense)

• resolution actions:
  • specific, like a regular policy
  • generic, choosing among conflicting policies
<resolution id='Forward-reject' owner='ken@stir.ac.uk'>
  <policy_rule>
    <triggers>
      <and/>
      <trigger arg1='variable0'>forward_to(arg1)
      <trigger arg1='variable1'>reject_call(arg1)
    </triggers>
    <condition>
      <parameter>preference0</parameter>
      <operator>in</operator>
      <value>preference1</value>
      <action>apply_stronger</action>
  </condition>
</resolution>
Goals
Goal Approach

• goal refinement is handled through optimisation:
  • a numerical approach gives greater flexibility
  • individual and overall goals are evaluated numerically
• goals resemble normal policies:
  • goal achievement is assessed through some measure
  • no trigger as goals are persistent
  • one action to maximise/minimise the goal measure
• normally there are multiple goals:
  • an overall evaluation function combines goal measures
  • goals are realised through combinations of prototypes
  • where there is conflict, an optimum selection is made
Maximise Network Use Goal

- measures are typically weighted sums:
  - measures use current environment values
  - weights are determined by typical values
  - automated sensitivity analysis checks proposed weights
- measure of network use:
  \[0.0008 \times \text{bandwidth} \times \text{duration} + 6.0 \times \text{handled}\]
- goal definition:
  
  ```xml
  <goal id='Maximise network use'
        owner='ken@stir.ac.uk' ...>
    <policy_rule>
      <action arg1='network_use'>maximise(arg1)</action>
  </policy_rule>
  ```
Minimise Call Cost Goal

• measure of call cost:
  \[ 1.0 \times \text{rate} \times \text{duration} \]

• goal definition:

  \[
  \text{<goal id='Minimise call cost' owner='ken@stir.ac.uk' ...>}
  \text{ <policy_rule>}
  \text{ <conditions>}
  \text{ <and/>}
  \text{ <condition>}
  \text{ <parameter>day <operator>in <value>1..5}
  \text{ <condition>}
  \text{ <parameter>bandwidth <operator>gt <value>128}
  \text{ <action arg1='call_cost'>minimise(arg1)}
  \text{ </policy_rule>}
  \text{ </conditions>}
  \text{ </goal>}
\]
Goal Refinement

- goal refinement has static and dynamic phases
- static analysis (on definition):
  - identify which prototypes contribute to which goals
  - instantiate prototypes as if they were regular policies
- dynamic analysis (on a trigger):
  - receive a trigger from the managed system
  - determine environment values and relevant policies
  - filter goal-related policies
  - choose their optimum combination and parameters
  - detect and resolve conflicts among the selected policies
  - ask the managed system to perform policy actions
Overall Goal Evaluation

• the overall evaluation function might be:
  + network_use
  - call_cost
  + multimedia_use
  - interruption_time

• sample goals/prototypes/conditions might yield:
  • add a lawyer to the call (for contractual calls)
  • add limited video to the call (for moderate bandwidth)
  • set a call limit of 10 minutes (for reduced cost)
  • include a manager in the call (as video is in use)
Tool Support
Managed System

• any system can be managed through policies:
  • an additional policy interface is required
  • notifies significant system events (e.g. call, hang-up)
  • performs policy actions (e.g. forward, reject)
• for telephony, interface modules exist for:
  • SER (SIP Express Router)
  • 7000 ICS (Mitel softswitch)
  • GNU GK (H.323 gatekeeper)
• performance overhead:
  • < 1 second (policies)
  • < 2 seconds (goals and policies)
Policy and Goal Servers

- the policy store is an XML database
- the policy server reacts to system events:
  - retrieves triggered and eligible policies
  - requests goal refinement
  - determines policy actions and resolves conflicts
- the goal server creates an optimal set of policies
Policy Wizard

- policy wizards support user-friendly definition:
  - inherently multi-lingual
  - near-natural language, interactive voice, digital pen

Applicability (label, owner, ...):

- label: Transfer a call to Jean
- owner: ken@stir.ac.uk
- applies to: ken@stir.ac.uk
- valid from: 2008-12-25 09:00
- valid to: 2009-01-06 09:00
- profile: In the office
- status: enabled

Preference (must, prefer, ...):

- prefer

Rules (combinations, triggers, conditions, actions):

- when a call is not answered after 10 seconds
- or
- when I am called
- if the hour is after 13:00
- do forward the call to jean@plc.com
- and then
- do send a message to michael@uni.ac.uk about call to Mark
Offline Conflict Analyser

- policies are analysed offline for conflicts:
  - conflicting effects suggest likely problems
  - results can be manually tuned
  - resolution policies are automatically generated

Goals and Policies for Telephony (C) K. J. Turner, University of Stirling 27
Ontology Server

• definition of the APPEL policy language:
  • core and domain-specific schemas
  • supplemented by core and domain-specific ontologies

• the ontology server allows:
  • tool-independent access to domain information
  • other goal and policy tools to be domain-independent
Review
Application Areas

• core and domain-specific aspects are separated
• the approach is thus generic and extensible
• current applications include:
  • (Internet) telephony
  • home care systems
  • sensor networks
  • wind farms
Conclusion

• the APPEL policy language supports:
  • policies, variables, resolutions
  • prototypes, goals
• the ACCENT toolset provides:
  • policy and goal servers
  • context and ontology servers
  • conflict detection and resolution
  • user-friendly wizards
• the approach is generic, though illustrated here on (Internet) telephony