Methods for Policy Conflict Detection and Resolution in Pervasive Computing Environments

Evi Syukur
Evi.syukur@csse.monash.edu.au

Seng Wai Loke
Swloke@csse.monash.edu.au

Peter Stanski
Peter.stanski@stanski.com

Policy Management for Web Workshop in Conjunction with
WWW2005 Conference, 2005, Chiba, Japan.

Overview

- Background
- Challenges
- Related Work
- Overview of MHS
- Research Objective
- Conflict Sources and Types
- Conflict Detection
- Conflict Resolution
- Prototype Implementation
- Performance Evaluation of MHS
- Lessons Learnt
- Future Work

Background

- Pervasive Computing
  - Anytime, anywhere of accessing information or utilizing services
- Policy
  - Definition: A set of rules to limit and control the behaviours of entities.
  - Role:
    - To control the entities’ behaviours in accessing mobile services in particular contexts
    - To specify behaviours which the system performs automatically

Challenges

- Have a simple policy language
- Detecting a conflict
- Managing a conflict
- Resolving a conflict

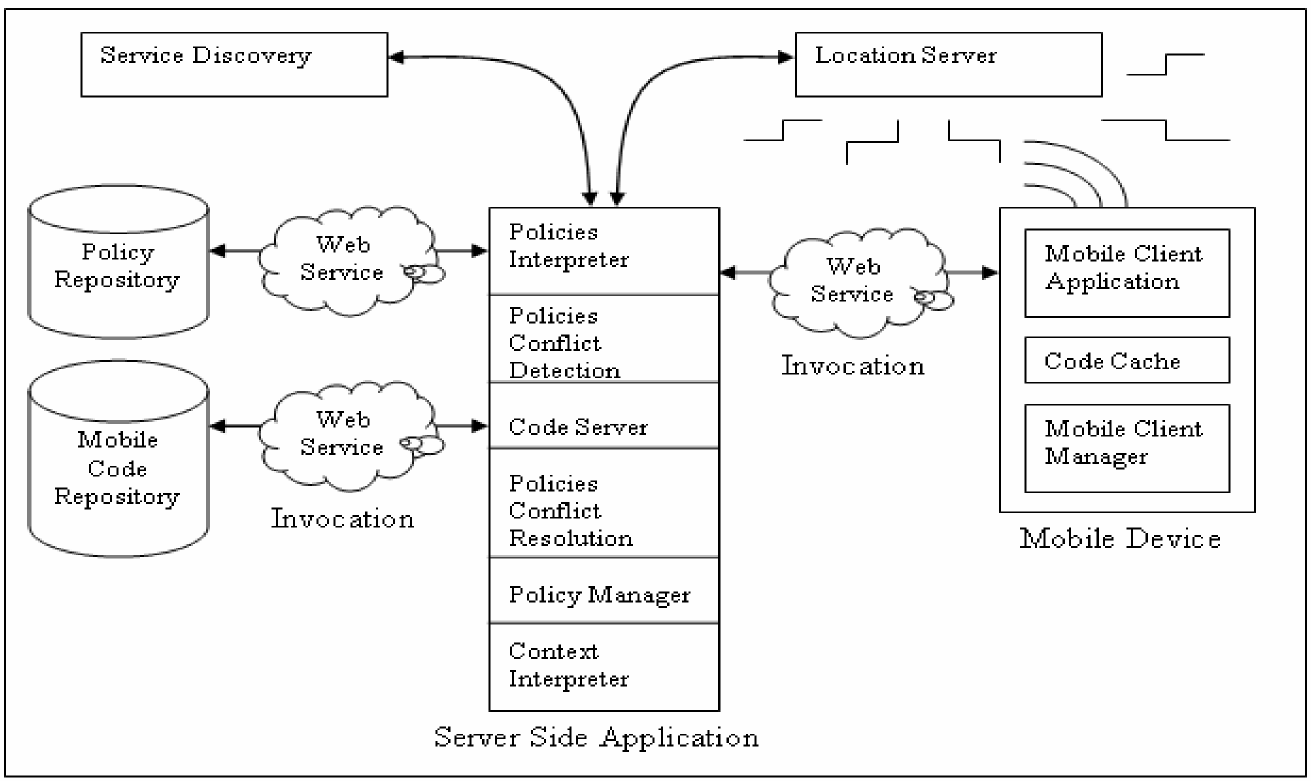
Related Work

- Policy work in pervasive systems
  - Rei (2003)
  - Spatial Policies (2003)

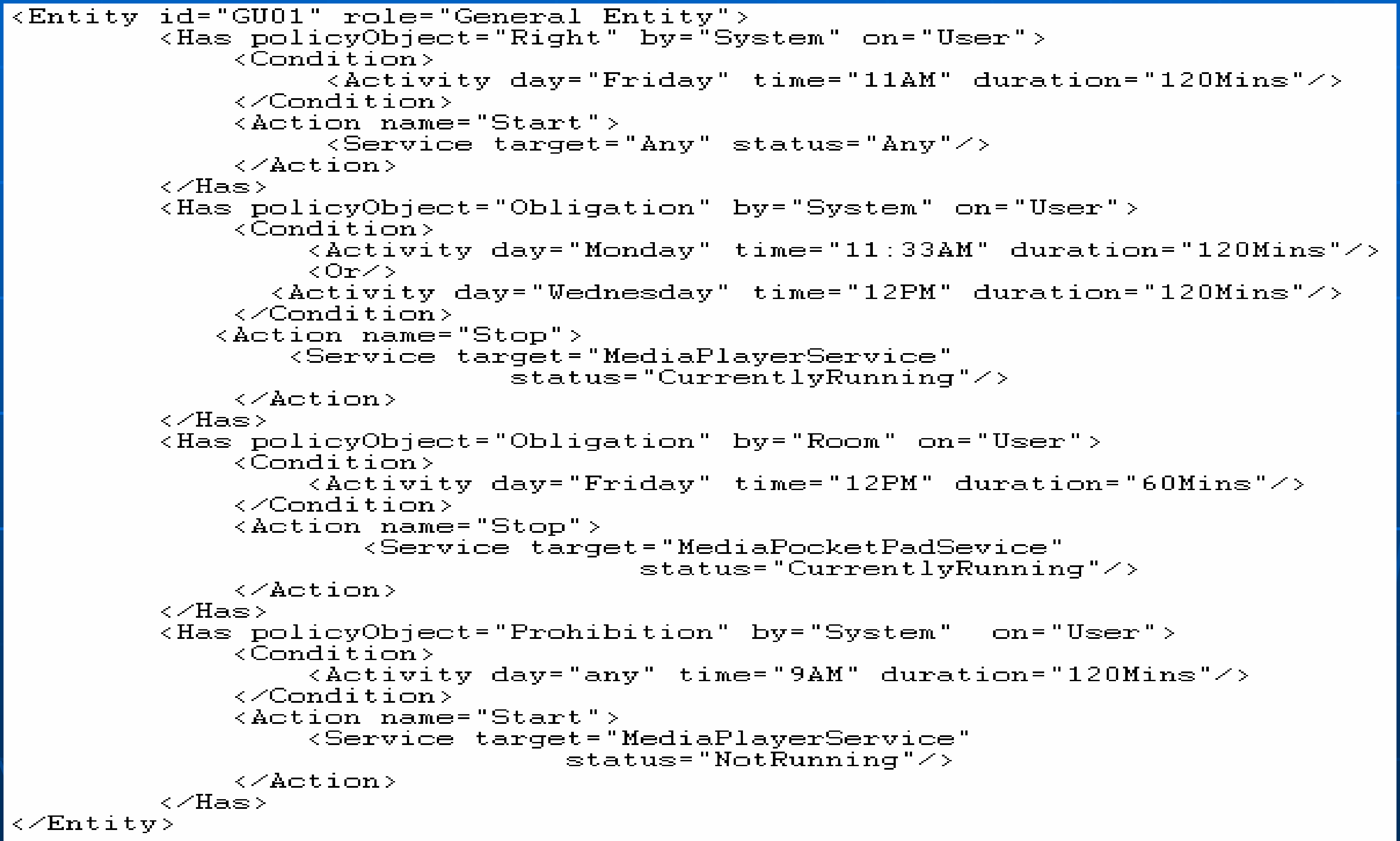
- Conflict detection and resolution work
  - Dynamic policy model by Nicole Dunlop
    - Focuses on enterprise and management policy based systems

MHS: a campus based policy system

- Architectural design



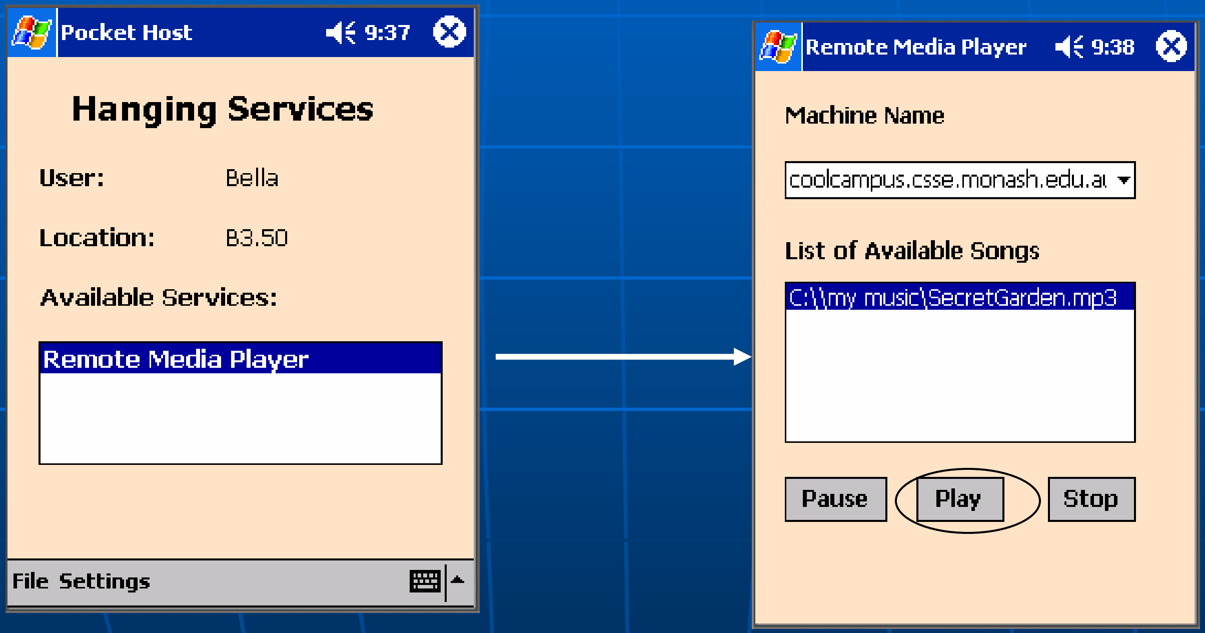
Policy in our system



```
<Entity id="GU01" role="General Entity">
        <Has policyObject="Right" by="System" on="User">
            <Condition>
                 <Activity day="Friday" time="11AM" duration="120Mins"/>
            </Condition>
            <Action name="Start">
                 <Service target="Any" status="Any"/>
            </Action>
        </Has>
        <Has policyObject="Obligation" by="System" on="User">
            <Condition>
                <Activity day="Monday" time="11:33AM" duration="120Mins"/>
                <Or/>
              <Activity day="Wednesday" time="12PM" duration="120Mins"/>
            </Condition>
           <Action name="Stop">
               <Service target="MediaPlayerService"
                           status="CurrentlyRunning"/>
            </Action>
        </Has>
        <Has policyObject="Obligation" by="Room" on="User">
            <Condition>
                <Activity day="Friday" time="12PM" duration="60Mins"/>
            </Condition>
            <Action name="Stop">
                   <Service target="MediaPocketPadSevice"
                                  status="CurrentlyRunning"/>
            </Action>
        </Has>
        <Has policyObject="Prohibition" by="System"  on="User">
            <Condition>
                <Activity day="any" time="9AM" duration="120Mins"/>
            </Condition>
            <Action name="Start">
                <Service target="MediaPlayerService"
                              status="NotRunning"/>
            </Action>
        </Has>
</Entity>
```

A service example

- Media Player service



Pocket Host 9:37

Hanging Services

User: Bella

Location: B3.50

Available Services:

Remote Media Player

Remote Media Player 9:38

Machine Name

coolcampus.csse.monash.edu.au

List of Available Songs

C:\\my music\SecretGarden.mp3

Our Objective

- To detect and resolve the conflict efficiently
  - System performance
  - Implementation
  - Does it accommodate all conflicts that might happen in the future?

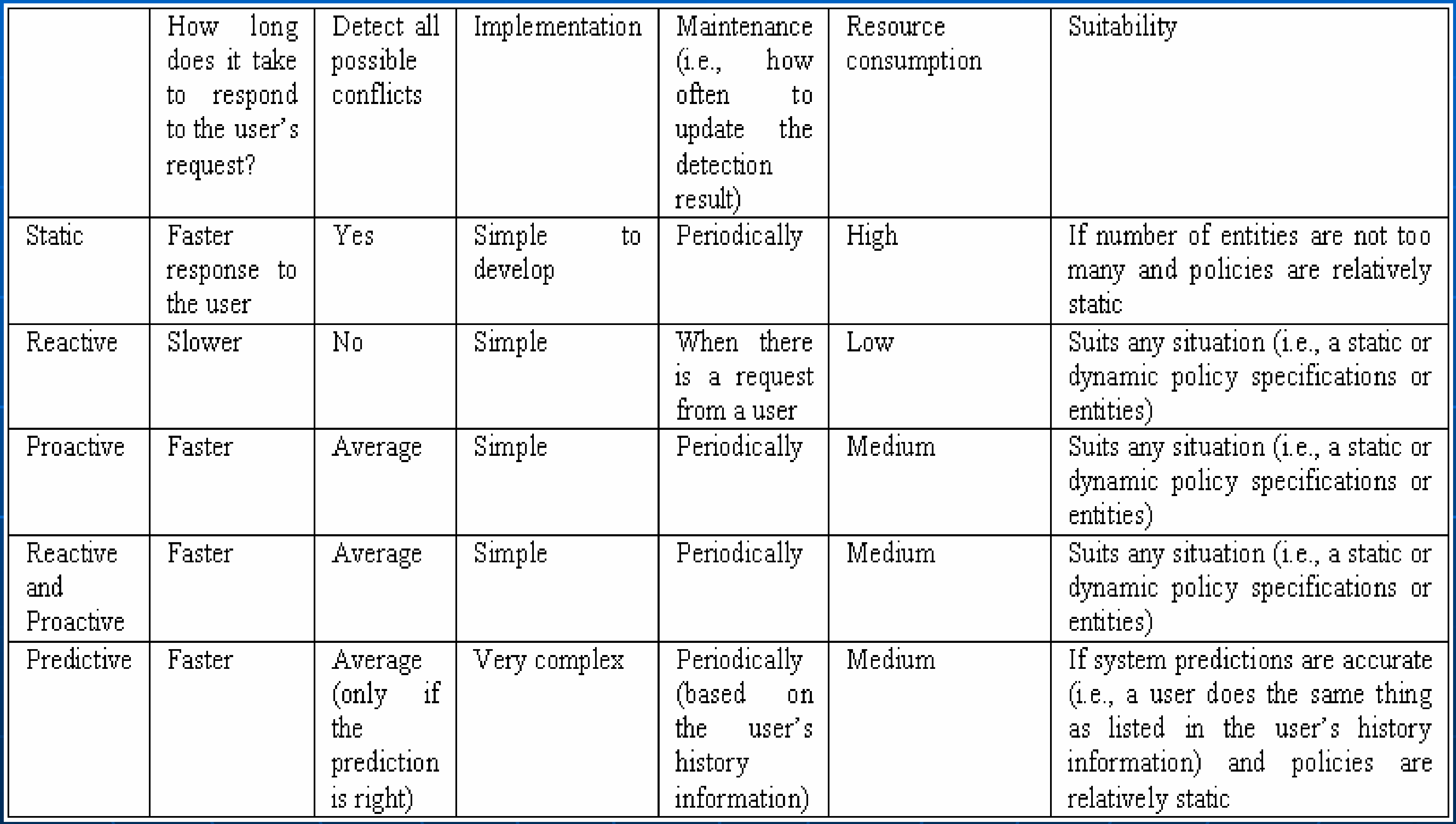
Conflict Sources and Types

- Conflict sources
  - Policy space modality conflict
    - Conflict between a system and a room
  - Role conflict
    - Conflict between a user with higher and lower role
  - Entity conflict
    - Conflict between users
- Conflict types
  - Potential conflict = not yet a conflict, as the contexts for a conflict to happen have not been satisfied
  - Actual conflict

Conflict Detection

- Aims
  - To investigate several possible sources of conflicts and types that may occur within a pervasive system
- Strategy
  - Static conflict detection
  - Dynamic conflict detection

Conflict Detection



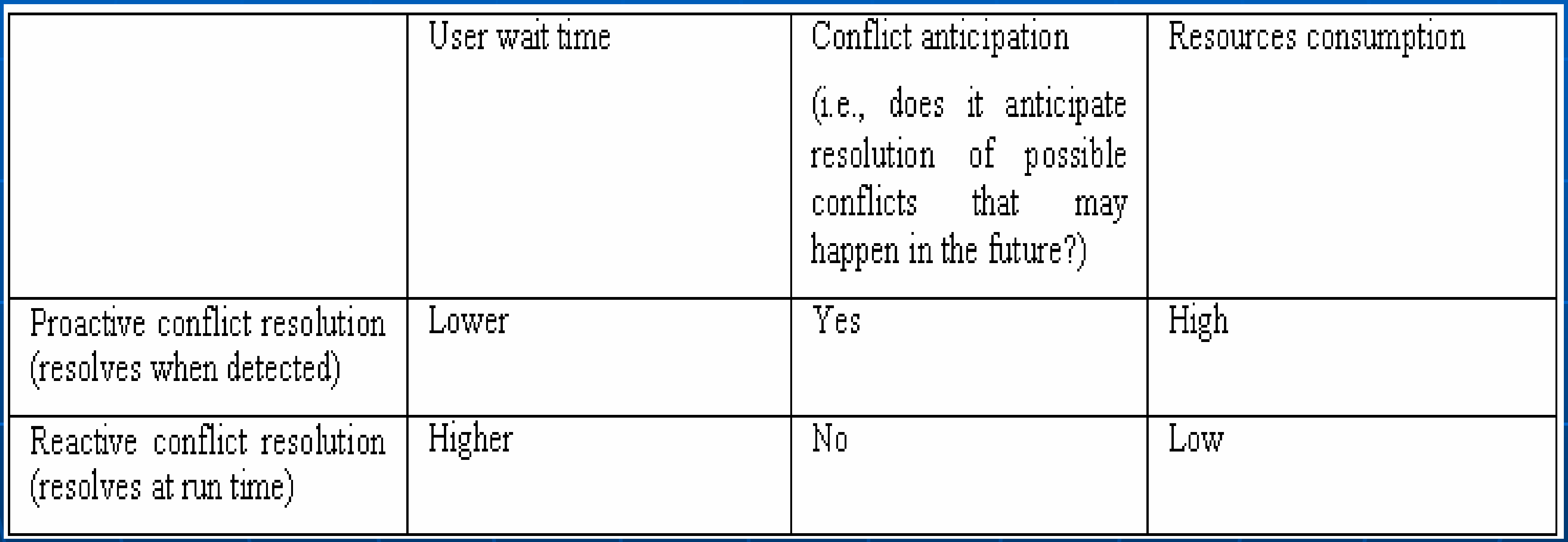
	How long does it take to respond to the user's request?	Detect all possible conflicts	Implementation	Maintenance (i.e., how often to update the detection result)	Resource consumption	Suitability
Static	Faster response to the user	Yes	Simple to develop	Periodically	High	If number of entities are not too many and policies are relatively static
Reactive	Slower	No	Simple	When there is a request from a user	Low	Suits any situation (i.e., a static or dynamic policy specifications or entities)
Proactive	Faster	Average	Simple	Periodically	Medium	Suits any situation (i.e., a static or dynamic policy specifications or entities)
Reactive and Proactive	Faster	Average	Simple	Periodically	Medium	Suits any situation (i.e., a static or dynamic policy specifications or entities)
Predictive	Faster	Average (only if the prediction is right)	Very complex	Periodically (based on the user's history information)	Medium	If system predictions are accurate (i.e., a user does the same thing as listed in the user's history information) and policies are relatively static

Conflict Resolution

- Aims
  - to resolve all types of conflicts in minimum amount of time, and so, minimizes the user wait time
- Techniques to resolve conflicts
  - Role hierarchy overrides policy
  - Space holds precedence over visitors
  - Obligation holds precedence over rights

Conflict Resolution

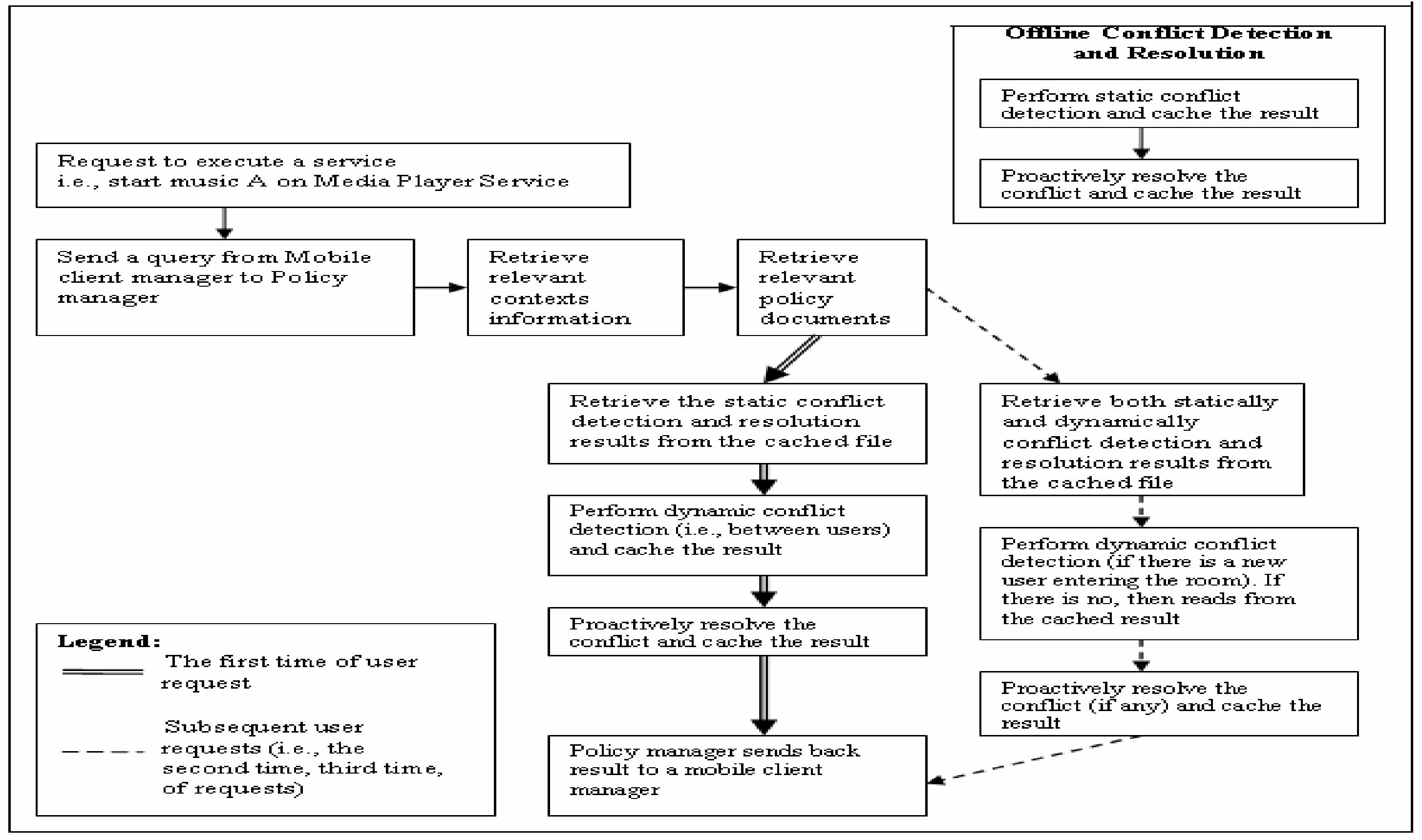
- When to resolve the conflict



	User wait time	Conflict anticipation (i.e., does it anticipate resolution of possible conflicts that may happen in the future?)	Resources consumption
Proactive conflict resolution (resolves when detected)	Lower	Yes	High
Reactive conflict resolution (resolves at run time)	Higher	No	Low

MHS: a campus based policy

- Implementation details



Offline Conflict Detection and Resolution

Perform static conflict detection and cache the result

Proactively resolve the conflict and cache the result

Request to execute a service i.e., start music A on Media Player Service

Send a query from Mobile client manager to Policy manager

Retrieve relevant contexts information

Retrieve relevant policy documents

Retrieve the static conflict detection and resolution results from the cached file

Perform dynamic conflict detection (i.e., between users) and cache the result

Proactively resolve the conflict and cache the result

Policy manager sends back result to a mobile client manager

Retrieve both statically and dynamically conflict detection and resolution results from the cached file

Perform dynamic conflict detection (if there is a new user entering the room). If there is no, then reads from the cached result

Proactively resolve the conflict (if any) and cache the result

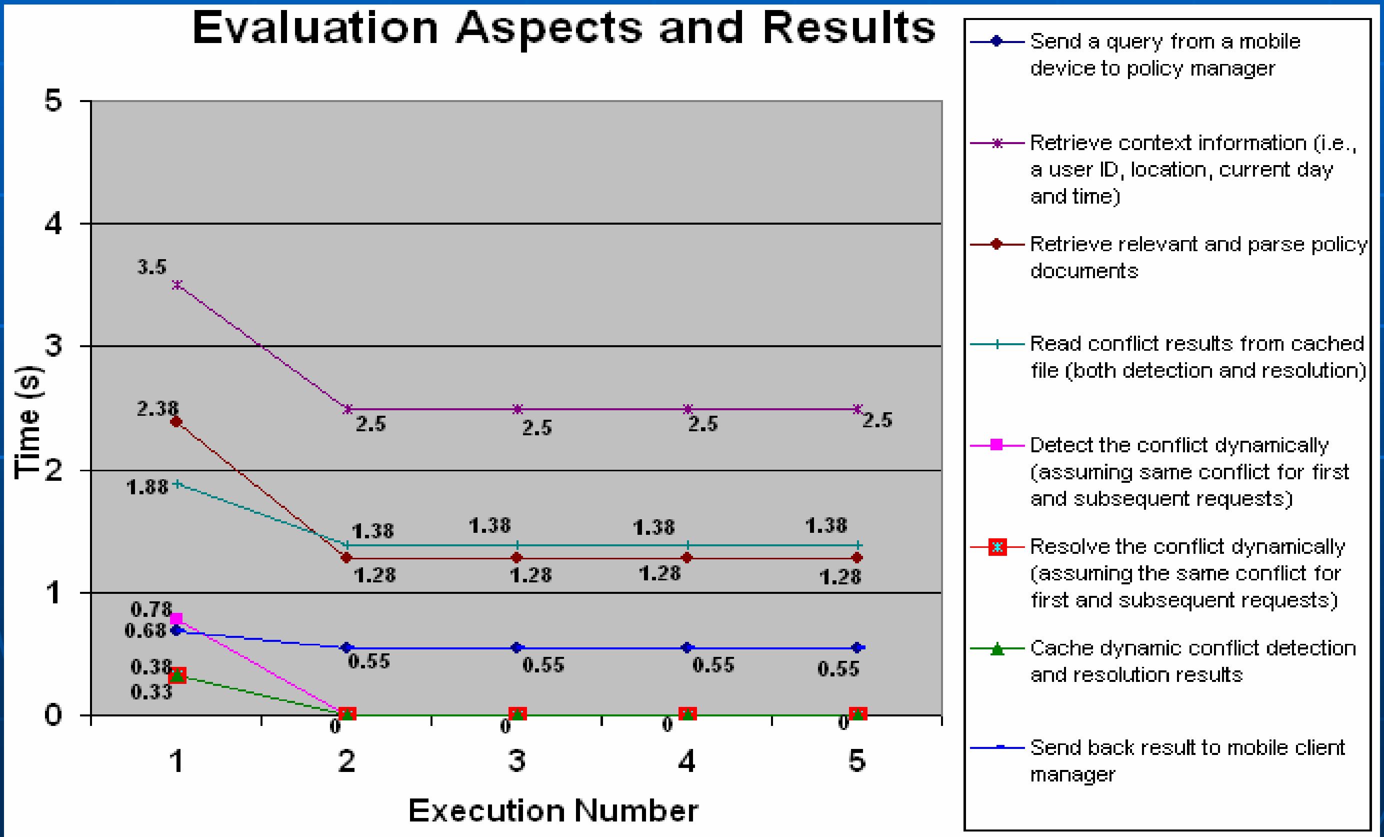
Legend:

═══ The first time of user request

– – – – Subsequent user requests (i.e., the second time, third time, of requests)

MHS: a campus based policy

- Performance results



MHS: a campus based policy

- Tuser wait time(s)

  = Tsend a query from a mobile client to policy manager

  + Tretrieve context information

  + Tretrieve and parse relevant policy documents

  + Tread conflict results from a cached file (both detection and resolution)

  + Tdetect a conflict dynamically (if any)

  + Tresolve a conflict dynamically (if any)

  + Tcache results (if any)

  + Tsend back result to the mobile client manager

- Best case scenario (i.e., minimum a user wait time delay)
  - Any execution which is not the first. It takes 6.26s (=0.55 +2.5 + 1.28 +1.38 + 0 + 0 + 0 + 0.55)
- Worst case scenario
  - The first time of requesting the service. It takes 10.61s ((= 0.68 + 3.5 + 2.38 + 1.88+ 0.78 + 0.33 + 0.38 + 0.68)

Lessons Learnt

- The user wait time can be reduced by
  - Resolving conflicts as soon as they are detected
  - Using a combination of conflict detection strategies
  - Reducing the time it takes to retrieve context information
- Advantage and Disadvantage of having a policy
  - Advantage: Can control and limit the entities' behaviours
  - Disadvantage: The user wait time to execute a service is longer
- The suitability of each conflict detection and resolution depends on the system designs, system goals and types of conflicts that we are dealing with

Future work

- Continue working on a proactive and predictive conflict detection strategies.
- Monitor the probability of potential conflict occurrence
- Study the nature of each conflict found in pervasive systems
- Have a policy conformance
- Apply our policy concepts (i.e., designs, conflict detection and resolution strategies) in different pervasive domains
- Allowing users to modify their policy specifications dynamically at run time
- Use ontology and Semantic Web Language to represent a policy language (hence, it allows reusable of language across different domains i.e., networking, pervasive, etc.)

References

[1]Kagal, L., Finin, T. and Joshi, A., “A Policy Language for a Pervasive Computing Environment, *Proc. of IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, Italy, June 2003.

[2]Scott, D., Beresford, A. and Mycroft, A., “Spatial Policies for Sentient Mobile Applications”, *Proc. of IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, Italy, June 2003.

[3]Dunlop, N., Indulska, J. and Raymond, K., “Dynamic Policy Model for Large Evolving Enterprises”, *Proc. 5th IEEE Enterprise Distributed Object Computing Conference*, Seattle, Sept 2001.

Questions?