Policy Based Framework for Trust Management and Evolution of Peer to Peer Groups.

Madhumita Chatterjee
(04429802)

Advisors:
Prof G. Sivakumar &
Prof Menezes. Bernard
Context

- Challenges of Collaborative P2P groups
  - Decentralized, dynamic, self-organized
    - Different functional roles of peers
  - Need for admission control, access control
  - Risks in interacting with previously unknown peers
    - Need for trust
  - Resource management and task scheduling

- Need for **dynamic policies** to optimise group behavior.
  - Membership control, task allocation, level or role elevation
  - Adaptive trust

- E.g Multiproject software ecosystem F/OSS
Status of work/contributions

- Proposed Framework and protocols for secure communication in collaborative peer groups
  - Integrated Framework for Authentication and Access control in peer to peer groups
    - WISA 2007 (Korea)
  - NO Policy Language defined
- Dynamic Policy based model for multi-level access control. Implemented framework in Peersim and policies in Prolog and interfaced the two with Interprolog. Modeled FOSS in the simulator.
  - Dynamic Policy based model for trust based access control in peer to peer applications (IEEE-ICC 2009 (Germany)
    - Preliminary experiments, No adaptive trust mechanism to handle malicious peers.
- Performed several experiments to validate the framework. Improved upon system model and proposed generic model with an adaptive trust mechanism capable of handling malicious peers in self-organising groups.
  - Dynamic Policy Adaptation for Collaborative Groups. (CNSA 2010 -(Chennai), (Recent Trends in Network Security), Springer
  - Performed several experiments to show that group can dynamically change policies to optimise performance and mitigate the effect of malicious peers. Improved State Model.
Trust Engine

Self Trust
Feedback by Direct interaction
Indirect Recommendation
Incentives for rating

Trust Computation

Weighted Context Specific Parameter
Peer Credibility

Context based policy

E-commerce Domain
Open Source Software Development
Online gaming
Policy 1
Policy 2
Policy 3

P2P Groups Integrated Framework
Authentication
Access Control
Key Mgmt.

Global meta policies for multilevel access control

Evolution of P2P Groups
Dynamic Policies
- Adaptive Trust
- Malicious Peers

Peersim Simulator

Role 1
Role 2
Role 3
L1
L2
L3
P2P Groups Integrated Framework

**Trust Engine**

- Self Trust
- Feedback by Direct interaction
- Indirect Recommendation
- Incentives for rating
- Weighted Context Specific Parameter

**Trust Computation**

- Peer Credibility

**Context based policy**

<table>
<thead>
<tr>
<th>E-commerce Domain</th>
<th>Open Source Software Development</th>
<th>Online gaming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy 1</td>
<td>Policy 2</td>
<td>Policy 3</td>
</tr>
</tbody>
</table>

**Global meta polices for multilevel access control**

- **Evolution of P2P Groups**
  - Dynamic Policies
    - Adaptive Trust
    - Malicious Peers

**Peersim Simulator**
Focus of work

- Generic Model for Collaborative peer groups
- Declarative language for dynamic policies
- Context sensitive trust model
- Simulator framework for experimental analysis
- Interesting results for case studies
  - Optimization with dynamic policies for join, leave, update, task allocation, etc.
  - Mitigating effect of malicious peers
Today’s Talk Outline

- System Model
- Malicious Peer model
- Simulator Framework
- Interesting Results
F/OSS Model

Tasks in the group
- development of software modules of different difficulty levels
- content distribution
- resource sharing
- publish/subscribe
- postings news-group.

Role_Set = \{GR U DR\}
GR = \{MP,AP,CP\}
DR = \{Developer, Reviewer, Moderator\}

Attribute_Set for Rating
- P: Coding cost (no of peers)
- T: Completion time
- L: Lines of Code
- Q: Quality

Events: Join, Leave, Update, Rating, task scheduling,

Skill-Module
- Concurrent
- Sequential
- Job_Diff = \{easy, moderate, complex\}
Group Activities and Behavior

- New Peer
- Existing peer
- Join
- Leave
- Update
- Rating
- Access control
- Role_Set
- Concurrent sequential
- Task Scheduling
- P2P Group
- Peer Behavior
- Group Metrics
- Dynamic Policies
Formal Model of Self Adaptation

- Self organising group should adapt to the environment
- At time t
  \[ S_t = (S_t^i)_{i \in N_t} \in S \]

- System behavior is modelled as a sequence of pairs of system state and total environment state.
- A system state transition is caused by transitions of individual states
  \[ (s_1, e_1) \rightarrow (s_2, e_2) \rightarrow (s_3, e_3) \ldots \]

Where e1, e2, ...en represent environment state
State Model

State Parameters---Group Metrics
- Avg reputation of group — c1
- Skill set of peers -- c2
- Job completion rate — c3
- Group composition — c4

Peer Characteristics
- SPR
- MIP
State Model

State Parameters---Group Metrics

- Avg reputation of group – c1
- Skill set of peers -- c2
- Job completion rate — c3
- Group composition — c4

Peer Characteristics

- SPR
- MIP

Group Health

State S₁

P₁

State S₂

P₂

State S₃

P₃

State Sₖ

Pₖ
Conditions that trigger Policy change

- Combination of group metrics /predicates
- \( c_1 \land c_2 \)
- Case
  
  if \( c_1 < t_1 \& c_2 > t_2 \), then \( P_1 \rightarrow P_2 \)

- *Group Health* measures the Group Performance Index for a particular state of the system and the system’s environment
Optimal Group Behavior

- **Abstract Function Group_Health**

\[ S^*(e) = \{ s \in S : \text{Group\_Health}(s, e) = \max \text{Group\_Health}(s', e) \} \].

So at time \( t \), set of optimal system states is

\[ S^*_t = S^*_{(et)}. \]

- Control peers periodically monitor Group_Health and trigger request for policy change
  - consensus with other control peers
- Group which is in state \( S_i \) with Policy \( P_i \) moves to state \( S_j \) with policy \( P_j \)
Generic Model for collaborative groups.

P2P Collaborative Groups

- Set of Peers with common interest
  - Service Provider
  - Rater

- Group Charter
  - Group Description
  - Group Activities
  - Meta Rules for membership
  - Group Composition—Role_Set = {GR U DR}

- Set of Activities
  - Group Management Activities
  - Task Oriented Activities
  - Social Tasks
  - Malicious or Destructive Activities

- Set of Events
  - Internal
  - External
    - Schedule
      - Periodic
      - Event Driven
Group Charter

- Defines Global and domain specific policies
- Global
  - roles permitted in the group and the rules associated with each role.
  - policies for new peer join, peer update, minimum trust/reputation levels, discarding malicious peers etc
  - E.g. For a new policy to be framed at least 75% of existing peers with trust value > some threshold x must collaborate. This ensures that malicious peers cannot frame new policies.

- Rules for creating destroying a group
  - Signed Hash value of policies stored
Domain Specific Policies

- Rules for updating role or level of a peer
- Rules to decide optimum no of peers needed at each level
- Task allocation policies
- Adaptive trust policies
  - Group works with trust model TF1 until it reaches a particular state and then switches to model TF2
Set of Activities

- Admission Control
- Access Control
- Publishing of Group Existence/Charter
- Creation of Groups
- Removal of Groups
- Resource Management
- Scheduling Tasks

Task Oriented Activities
- Procedural (function/task of Groups i.e Domain Specific)
- Information/Resource request
- Opinion/Reputation request (Trust)
- Opinion/Reputation response
- Record Maintenance (of Group activities)
- Maintenance of certificates, keys, ratings.
- Voting
- Evaluator/critic (measures group activities against some Group Performance Index GPI)

Social Activities
- Reward actions
- Punish actions
- Observer actions i.e maintaining records of those group activities that help in providing feedback.

Malicious Activities
- Aggressor behavior i.e peers who lower the rating of other peers.
- Recognition seeker i.e colluding peers who call attention to self by providing very good behavior for some time and increasing each others ratings.
Set of Events

- **external**
  - Join
  - Leave

- **internal**
  - Update
  - Task Allocation and Scheduling
  - Policy change
Modeling policies using Prolog

- We define Prolog Facts as entities, levels, roles and allowed operations.
- **Entity Set** = \{UUID, PK\}
- **Role Set** = \{GR U DR\}
  - **GR** = \{MP, AP, CP\}
  - **DR** = \{developer, reviewer, moderator\}
- **Operation Set** = \{Read, Write\} or
  \{submit, review, moderate\}…. 
Rules for Join

- \text{verify}(N\text{peer}, Request, Rl, Level, Rate, SPR) : - \\
  \text{join}(Request), \text{member}(Rl), \text{Level} =:= 1, SPR >= 40, \text{assert(belong}(N\text{peer}, \text{Level})).

- \text{verify}(Request, Rl, Level, Rate, SPR, Expert, Tot ) : - \text{join}(Request), \text{admission}(Rl), \text{Level} =:= 2, SPR> 50, \text{Expert}> 20, \text{Tot} < 200.
Dynamic Rules

- `update engine(Levels, MPs, APs, CPs, Tot) :- Tot > 100, CPs > 20, retract(old_join_rule), assert(new_join_rule)`

- `update engine(Levels, MPs, APs, CPs, Tot, AvgT) :- AvgT > t, CPs > 20, retract(old_update_rule), assert(new_update_rule)`
Domain Specific Policies

- `update_level(CodesSubmitted, PercentAccept, Level, Role):`-member(Role), CodesSubmitted > 6, PercentAccept > 50, Level =:= 1.

- `update_level(CodesSubmitted, PercentAccept, Level, Role):`-admission(Role), CodesSubmitted > 20, PercentAccept > 50, Level =:= 1.

- `introduce_level(Role, Level, CPs, NPeers, High_Level, Vote):`-maximal(Role), CPs > 40, Npeers > 40, Level =:= High_Level, Vote =:= 100.
Layered and context sensitive reputation model

- Self Trust
- Feedback by Direct interaction
- Indirect Recommendation
- Incentives for rating

Weighted Context Specific Parameter
Peer Credibility

Trust Computation
Modeling malicious peers

- Simple malicious peer.
  - Under-performing peers
- Hypocritical peers.
  - random
  - structured.
- Colluding Peers.
Handling malicious peers

- Each peer builds a knowledge base represented by three list structures viz:
  - Doubted List, Black List and Trusted List.
- Peers with sudden deviation in perf are put in doubted list
  - Credibility reduced but given second chance
- Trust policy does not permit interactions with peers in black list
  - Purged according to an ageing policy
Global Trust value

- $T$ is an aggregation of the direct experiences of every peer about peer $x$ as well as the recommendations received about peer $x$, and the context factor.
  $$\frac{F_x}{I_{x,y}}$$

$$T = \alpha \times T_D(x, y) + \beta \times T_{ID}(x, y) + \gamma \times Context\, factor$$

- $\alpha$ is the weight associated with direct experience,
- $\beta$ with indirect reputation
- $\gamma$ is a fine tuning constant to control the amount of reputation gained by rating others.
Adaptive Reinforcement Learning for Trust

- **Generation of Reinforcement Value**
  - $R = I_{\text{max}} \times (\text{Th} - \text{opErr})$, $\text{opErr} < \text{Th}$
  - $= P_{\text{max}} \times (\text{opErr} - \text{Th})$, otherwise

- $R = \text{Reinforcement value}$
- $\text{Th} = \text{threshold for separating rewards and punishments}$
- $\text{opErr} = \text{error associated with opinion}$
- $I_{\text{max}} = \text{maximum incentive}$
- $P_{\text{max}} = \text{maximum punishment}$

**Goal** = maximise $R$
Task allocation policies

- Allocate jobs to resources depending on their trust values/roles without considering the complexity of jobs.
- Allocate jobs depending on the complexity level.
- Allocate jobs depending on deadline to finish.
- Allocate jobs depending on security policy, i.e., whether it needs encryption or not.
PeerSim Simulator

- PeerSim is a library written in the Java language which consists of different components or classes which help in constructing and initializing the underlying network.
- Every component can be dynamically loaded through a configuration file.
PeerSim Architecture
Any peersim component can be defined and configured.

Any component can be fine scheduled.

The network size, the length of the simulation (in cycles).

The Node class type can be redefined.

Simulation details (e.g., shuffling, seed,...).

Number of distinct experiments.
Experiments
Job Allocation Policies

<table>
<thead>
<tr>
<th></th>
<th>Easy</th>
<th>Medium</th>
<th>Difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alloc1</td>
<td>0 - 40</td>
<td>40 - 70</td>
<td>70 - 100</td>
</tr>
<tr>
<td>Alloc2</td>
<td>0 - 60</td>
<td>60 - 80</td>
<td>80 - 100</td>
</tr>
<tr>
<td>Alloc3</td>
<td>0 - 80</td>
<td>80 - 90</td>
<td>90 - 100</td>
</tr>
</tbody>
</table>

- In job alloc_1 when a peer enters the system with SPR less than 40 and there is no easy job in the queue then the next level complex job will be allocated and so on.

- In job alloc_2 if there is no easy job when peer with SPR < 40 enters then he is kept idle,..no allocation done

- Job alloc_3:- peer with lower SPR is not allocated a complex job, but peer with higher SPR may be allocated an easy job.
Static vs dynamic join
Varying job allocation with easy jobs
Static vs dynamic job allocation

![Graph showing comparison between static and dynamic job allocation]

- The graph compares the performance of static and dynamic job allocation methods.
- The x-axis represents the percentage completed, while the y-axis shows the time taken.
- Different lines represent various allocation strategies: `job_alloc_1`, `job_alloc_2`, `job_alloc_3`, `dynamic`, and `No_Policy`.
- The performance of static allocation strategies is compared against a dynamic approach to highlight their efficiency.
30 percent malicious peers with varying job allocations
10 percent malicious peers static vs dynamic
Aggregate trust values of peers with 10 percent malicious peers
Aggregate trust values of peers with 10 percent malicious peers
Average Rating for oversmart vs honest peers
Comparison of 10 percent vs 30 percent malicious peers
Static vs dynamic policies in presence of malicious peers
Future Plan
**Thesis Timeline**

- Jan 2006  APS 1
- Jan 2007 APS 2
- Jan 2008  APS 3
- Jan 2009 APS 4
- Aug 2010 APS 5
- Jan 2011 APS 6

**Contributions**

- A -- Proposed Integrated framework for authentication, access control and key management in dynamic peer groups. Implemented in JXTA
- C -- Proposed Generic Model for framework, and adaptive trust model. Performed several experiments to show that group efficiency increases with our dynamic policies.
- C' -- Improved Generic system model. Experimental results with malicious peers.

Integrated A, B, C and C' into a journal paper, submission by March 2011
Publications....


Acknowledgements

- Prof G. Sivakumar
- Prof M. Bernard
References


Prolog

- Use of logic programming language prolog
- Rule is an expression of the form
- \( Ro(uo) \leftarrow R1(u1), \ldots, Rn(un) \)
  - If Prolog knows that body follows from the information in the knowledge base, then Prolog can infer head.
  - The Prolog inference engine provides a mechanism to derive consistent access control decisions at runtime.
  - New facts from independent policy sources can be added to the policy base before decisions are made, ensuring dynamic decisions at runtime.
Global policy for join

- join(join).
- update(update).
- member(member).
- admission(admission_peer).
- maximal(maximal_peer).
- verify(Request, Rl, Level, Rate, SPR, Expert, Tot) :-
  join(Request), admission(Rl), Level =:= 1, SPR > 40.

- verify(Request, Rl, Level, Rate, SPR, Expert, Tot) :-
  update(Request), maximal(Rl), Level =:= 2, Rate >= 6, Expert < 20.
Dynamic rules

- Assert and retract clauses of Prolog can be used to express dynamism

```
update_engine(Loads, MPs, APs, CPs) :-
  CPs > 20, retract((verify(Npeer, Request, RI, Level, Rate, Vote) :- update(Request), maximal(RI), Level =:= 3, Rate >= 7)),
  assert((verify(Npeer, Request, RI, Level, Rate, Vote) :- update(Request), maximal(RI), Level =:= 3, Rate >= 9)).
```
Cumulative contributions of active developers can be defined in terms of the non-linear equation

\[ C(t+1) = C(t) + (\alpha \times P \times R(t)) - (\beta \times C(t)) \]

Where:
- \( C(t) \) = contribution at time \( t \)
- \( P \) = no. of generated proposals/submitted code
- \( R(t) \) = current reputation level of developer
- \( \alpha \) = acceptance rate by reviewer/moderator
- \( \beta \) = overriding of prior contributions by new developments