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

Madhumita Chatterjee
(04429802)

Advisors:
Prof G. Sivakumar.
Prof Menezes Bernard

1/28/2009
Outlay of Presentation

- Context – Motivation
- Status of Work

- System Model
- Dynamic policies in Prolog
- Modelling F/OSS
- Experiments
- Future Plan
Context

- Collaborative P2P groups
  - Self-organising
  - E.g. Multiproject software ecosystem F/OSS

- Adaptive

- Evolving

- Security
  - Multi-level dynamic access control
  - Policy based integrated with Reputation
# Status of work

## Motivation and Context
- Collaborative peer groups-Admission control, voting, authentication.
- Multi-level and multiple roles of peers
- Dynamic Policy based and reputation based access control
- Adaptive and evolving trust for self-organising groups
- Tool to be able to select optimum policy for a specific application

## Work Done
- Framework and protocols for secure communication in **collaborative peer groups**
- **Dynamic Policy based** model for multi-level access control integrated with trust.
- Implemented framework in Peersim and policies in Prolog and interfaced the two with Interprolog. (**ACK, dual degree student Ashish Arya**)
- Performed several experiments to validate the framework.
- Working on improving the system model with an **adaptive trust mechanism** capable of handling **malicious peers** in self-organising groups.
- Plan to include **collaborative re-inforcement** learning for peers in the model

## Publications
- **Integrated Framework for Authentication and Access control in peer to peer groups**
  - **WISA 2007** (Korea)
  - Status- published
- **Dynamic Policy based model for trust based access control in peer to peer applications**
  - **ICC 2009** (Germany)
  - Status - Accepted and under revision
- **A ranked conference by May 2009**
- **Journal paper by September 2009**
P2P Groups Integrated Framework

- Authentication
- Access Control
- Key Mgmt.

Global meta policies for multilevel access control

Trust Engine

- Weighted Context
  - Specific Parameter
- Peer Credibility

Self Trust

Feedback by Direct interaction

Indirect Recommendation

Incentives for rating

Trust Computation

Context based policy

- E-commerce Domain
- Open Source Software Development
- Online gaming

- Policy 1
- Policy 2
- Policy 3

Evolution of P2P Groups

- Dynamic Policies
  - Adaptive Trust
  - Malicious Peers

Peersim Simulator
System Model

• Collaborative peer groups
  ■ Peer has unique user id
  ■ Self-proclaimed rating
  ■ Intrinsic capability
  ■ Maximum potential

• Peers belong to different levels (competency, reputation earned) and perform different roles
Peer Roles

- Peer roles depend on functionality performed by peer
- Peer can either be a service provider only or it can be a service provider and do additional functionality like rating other peers, voting etc.
  - Member peer
  - Admission peer
  - Control peer
- Role of a peer is independent of the level.
Peer Behavior Quality

• Peers can be **Good, Average, Ok** or **Dynamic** in the role as service providers

• Good peers
  ■ expert peers who always provide excellent service.

• Average peers and OK peers
  ■ service quality gradually deteriorates.

• Dynamic peers
  ■ provide the services with one of the three qualities uniformly i.e 33% of its actual services are good.
Peer Quality.....cont

• As a rater, peer quality can be Good, Bad or Honest.

• A Good(Bad) peer would send absolutely correct(wrong) ratings each time based on truth.
  - Difficult without global view of system

• Honest raters provide best-effort truth based on their own views.
Rating Dissemination

- **Push mode**
  - Whenever a peer rating exceeds previous threshold it is propagated to k neighbor peers

- **Pull mode**
  - A peer queries k neighbor peers with whom it had successful interactions for ratings.

- **Both push pull method propagate the bad reputation to all peers in neighborhood.**
Reputation Index Window

- 3 Win (sliding window approach) for storing reputation values
- A node's history is divided into 3 performance windows named as Reputation Index Windows (RIWs) numbered 1 to 3.
- \( RI = \alpha*RIW_1 + \beta*RIW_2 + \gamma*RIW_3 \)
Global and Domain Specific Policies

- Group Charter defines global policies needed to frame the group, describes the roles permitted in the group and the rules associated with each role.
- Policies for new peer join, peer update, minimum trust/reputation levels etc
- Signed Hash value of policies stored.
- Domain specific policies for access rights, authentication, trust and privacy.
Task Allocation

- Group could have different tasks at various difficulty levels
- Tasks could be sequential or concurrent with or without interdependency.
- Scheduling of tasks is done to peers based on availability and peer capability
Group behavior measure

- Measure of group behavior could be in terms of:
  - average reputation of group
  - task completion of group with respect to time
  - group composition i.e no of expert peers in highest layer of group.
Major Features

- Multiple policies apply to group
  - Global and Domain Specific
- System can switch between policies over time
- Policies change based on group composition and trust level
- Group policy can dynamically prioritize requests for join or update
System model as state behavior
State Behavior

- The set of initial states of the peers’ comprises the system state. At time $t$ it is denoted as $s_t = (s_t^i)_{i \in N_t} e S$
  
  - $N_t$ is the set of peers at time $t$.
  - $s_t^i$ are the initial states of each peer.
  - $S$ is the set if all possible system states.

- $e_0, e_1, e_2, \ldots, e_n$ represent environment state

- $E_1, E_2, E_3$ represent external events
• System behavior is modelled as a sequence of pairs of system state and total environment state.

• System state transition \((s_1,e_1) \rightarrow (s_2,e_2) \rightarrow (s_3,e_3) \ldots \)

• Behavior \(\text{beav} : SX\xi \rightarrow R\)

• \(S^*(e)\) is the set of optimal states

\[
S^*(e) = \{ s \in S : \text{beav}(s,e) = \max \text{beav}(s',e) \}.
\]
Modelling Policies using Prolog

- Use of logic programming language prolog
- Rule is an expression of the form
- $Ro(uo):-R1(u1),...,Rn(un)$
  - 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).
- \( \text{verify}(N\text{peer}, Request, Rl, Level, Rating, Vote) :\text{-join}(Request), \text{member}(Rl), \text{Level} =: = 1, \text{Vote} >= 40, \text{Rating} >= 3, \text{assert}(\text{belongs}(N\text{peer}, \text{Level})). \)
- \( \text{verify}(N\text{peer}, Request, Rl, Level, Rating, Vote) :\text{-join}(Request), \text{member}(Rl), \text{Level} =: = 2, \text{Rate} >= 5, \text{assert}(\text{belongs}(N\text{peer}, \text{Level})). \)
Dynamic rules

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

```
update_engine(L, M, A, C) :- C > 20, retract((verify(N, R, L, V, R)), update(R), maximal(L), Level =:= 3, Rating >= 7),
assert((verify(N, R, L, V, R)) :- update(R), maximal(L), Level =:= 3, Rating >= 9).
```
Simulation framework
F/OSS Domain

- Represents a group of like minded participants to develop software systems and related artifacts intended to be shared freely.
- F/OSS systems, hyperlinked artifacts and tools and project web sites serve as venues for:
  - socializing, building relationships and trust, sharing and learning with others.
- Software evolution in a multi-project F/OSS ecosystem is a process of co-evolution of:
  - interrelated and inter-dependent projects, people, artifacts, tools, code and project specific processes.
- Tasks in the group:
  - development of software modules of different difficulty levels
  - content distribution
  - resource sharing
  - publish/subscribe
  - postings news-group.
F/OSS Domain

- The attribute set for this domain would consist of parameters like...
  - P → Coding cost (no of peers)
  - T → Completion time
  - L → Lines of Code
  - Q → Quality
Assumptions

- A project in our model consists of different modules of different difficulty levels.
- The group could be working simultaneously on multiple projects.
- Modules could run concurrently, currently we have not assumed any interdependence between modules.
Assumptions ....cont

- A project module consists of following parameters:
  - Starting time
  - Current time (clock of every module is different and works concurrently)
  - Time of completion
  - Expected time of completion
  - Status (Finished, started, working)
  - Bugs (number of bugs)
• No of bugs is a Uniform Distribution
• Module arrival is as per Poisson distribution.
• Average completion time of a module depends on difficulty level.
• Attributes considered while rating the performance of a peer developing some code is
  ■ No of bugs, Time Taken, Lines of Code.
• Module completion is also assumed to be a Poisson distribution
• Each module has an expected time of completion
Peer Behavior

Levels

Coder

OK (Member)

Reviewer

Moderator

Expert

OK

Average

Expert

Bugs [0, 1, 2, 3]

Average

Bugs [0, 1]

Bugs [0, 1, 2]

Expert

[9, 10]

Average

5-10

6-10
Domain Specific Policies

- `update_level(CodesSubmitted, PercentAccept, Level, Role):-member(Role), CodesSubmitted > 6, PercentAccept > 50, Level =:= 1.`
- `update_level(CodesReviewed, PercentAccept, Level, Role):-member(Role), CodesReviewed > 10, PercentAccept > 50, Level =:= 2.`
- `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.`
Goals

1. Controlled Group formation
2. Allowing peers to dynamically and collaborative change group policies results in more efficient group formation.
3. Peers with good behavior rise to higher levels more quickly.
4. Average trust rate of expert peers is the highest.
5. Our decentralized methods of storing data and reputation values ensures that the group performance does not degrade as data is stored only with trusted peers.
6. Peers with bad performance are penalized and moved down the layers.
7. Peers could learn in the group according to a learning curve which then affects the growth curve.
Group efficiency versus time

- Initially 40 peers and 20 total modules
Load on admission peer vs time
Load on Control peer versus time
Average Trust Rate
Future Plan (A) – Evolution of P2P groups

- Testing the flexible policy framework for evolution of P2P groups in different application contexts.
  - User will be able to decide which policy set is best suited for a specific application.
Future Plan (B) Adaptive Trust Module With Learning Incorporated

- How trust decisions can be made adaptive to changes in trust among peers
  - Behavioral and temporal changes
  - No prior knowledge
- Application of Collaborative Reinforcement Learning Algorithms
- Dynamic Adaptive trust policies
  - Group uses $TP_1$ till it reaches a certain state then uses $TP_2$
Other.....

- Need to make the approach more robust against malicious peers.
- Effect of degree of maliciousness and frequency of malicious peers affecting the group
Contributions....


Acknowledgements

- Prof G. Sivakumar
- Prof M. Bernard