Cooperative Peer Groups in NICE

INFOCOM 2003

謝志峰

2003/9/25
Outline

- Introduction
- Cooperative System
- Nice
- Distributed Trust Computation
- Simulations
- Conclusions
- Comment
A distributed scheme for trust inference in peer-to-peer networks.

- NICE system is a platform for implementing cooperative applications over the Internet.

- We describe a technique for efficiently storing user reputation information in a completely decentralized manner.

We present a new decentralized trust inference scheme that can be used to infer across arbitrary levels of trust.
Cooperative System

- We define a cooperative application as one that allocates a subset of its resources, processing, bandwidth, and storage, for use by other peers.

- The goal is develop algorithms that will allow “good” users to identify other “good” users, and thus, enable robust cooperative groups.
Let the “good” nodes find each other quickly and efficiently: Good nodes should be able to locate other good nodes without losing resources interacting with malicious nodes.

Malicious nodes and cliques should not be able to break up cooperating groups by spreading mis-information to good nodes.
NICE

- NICE is a platform for implementing cooperative distributed applications.

- Applications in NICE gain access to remote resources by bartering local resources.

- Transactions in NICE consist of secure exchanges of resource certificates.
NICE provides the following services:
- Resource advertisement and location
- Secure bartering and trading of resources
- Distributed “trust” valuation
NICE

- **Trust-based pricing:**
  - In trust-based pricing, resources are priced proportional to mutually perceived trust.
  - From Alice to Bob is $T_{Alice}(Bob) = 0.5$, and $T_{Bob}(Alice) = 1.0$
  - Alice trades with a principal with lower trust she incurs a greater risk of not receiving services in return.
Distributed Trust Computation

- Each involved user produces a signed statement (called a *cookie*) about the quality of the transaction.

- Consider a successful transaction $t$ between users Alice and Bob in which Alice consumes a set of resources from Bob.

- After the transaction completes, Alice signs a cookie $c$.

- Each transaction creates new cookies which are stored by different users.
Distributed Trust Computation

- **Strongest path:**
  - Alice chooses the *strongest* path, and uses the minimum trust value on the path as the trust value for Bob.
  - The strongest path is $AEBF$, and Alice infers a trust level of 0.8 for Bob.

- **Weighted sum of strongest disjoint paths:**
  - $ACDB$ is the other disjoint path (with strength 0.6), and the inferred trust value from Alice to Bob is 0.72.
Distributed Trust Inference: Basic Algorithm

- Each user stores a set of signed cookies that it receives as a result of previous transactions.

- Suppose Alice wants to use some resources at Bob’s node.
  - Either Alice already has cookies from Bob, or Alice and Bob have not had any transactions yet.

- When Alice has no cookies from Bob.
  - Alice initiates a search for Bob’s cookies at nodes from whom she holds cookies.
  - Suppose Alice has a cookie from Carol, and Carol has a cookie from Bob.
  - Carol gives Alice a copy of her cookie from Bob.
  - Alice presents two cookies to Bob: one from Bob to Carol, and one from Carol to Alice.
Refinements

- Whenever node receives a cookie from some other node, it also receives a digest of all other cookies at the remote node.

- Each node keeps a digest of recently executed searches and uses this digest to suppress duplicate queries.
Negative Cookies

- It follows high trust edges out of Bob and terminates when it reaches a negative cookie for Eve.

- The search returns a list of people whom Bob trusts who have had negative transactions with Eve in the past.

- If Bob discovers a sufficient set of negative cookies for Eve, he can choose to disregard Eve’s credentials, and not go through with her proposed transaction.
Simulations

- We simulate a stable system consisting of only good users.
- We assume that all users implement the entire search protocol correctly.
- Each query starts at a node $s$ chosen uniformly at random and specifies a search for cookies of another node $t$ chosen uniformly at random.
Simulations

- The higher threshold searches have a less possible absolute margin of error, and thus produce the best paths.
- However, very high threshold searches are also more likely to produce no results.
Simulations

- The number of failed transactions are proportional to the number of bad users in the system.
- Bad nodes rapidly fill the preference lists of good nodes, but are quickly identified as malicious.
Conclusions

- A low overhead trust information storage and search algorithm is used in the NICE system to implement a range of trust inference algorithms.

- We have presented a scalability study of our algorithms, and have shown that our technique is robust against a variety of attacks by malicious users.
Comment

- Cache
- QoS
- Routing