Distributed Trust Closures in NICE

Seungjoon Lee

Rob Sherwood

Bobby Bhattacharjee

University of Maryland

www.cs.umd.edu/projects/nice

Cooperative Applications

- Applications that allocate some resources for use by other application peers
  - Resources include processing, b/w, and storage
- Examples: on-line media streaming, peer-to-peer applications including file sharing and lookup

Cooperative Application Models

- Centralized trust repository (Mojonation, ebay)
    - requires central registry and authority

Cooperative Application Models

- Centralized trust repository (Mojonation, ebay)
    - requires central registry and authority
- Implicit cooperation (p2p apps)
    - *any* individual user may choose not to cooperate
      but they still get full benefits from the system
    - integrity and correctness depends on *all* users cooperating
    - Current solutions don't work well, e.g. quotas in CFS, or require centralized certificate authorities, e.g. secure routing in Pastry

Cooperative Application Models

- Centralized trust repository (Mojonation, ebay)
  - requires central registry and authority
- Implicit cooperation (p2p apps)
  - *any* individual user may choose not to cooperate
    but they still get full benefits from the system
  - integrity and correctness depends on *all* users cooperating
  - Current solutions don't work well, e.g. quotas in CFS, or require centralized certificate authorities, e.g. secure routing in Pastry
- NICE: decentralized robust cooperation
  - enable open cooperative applications

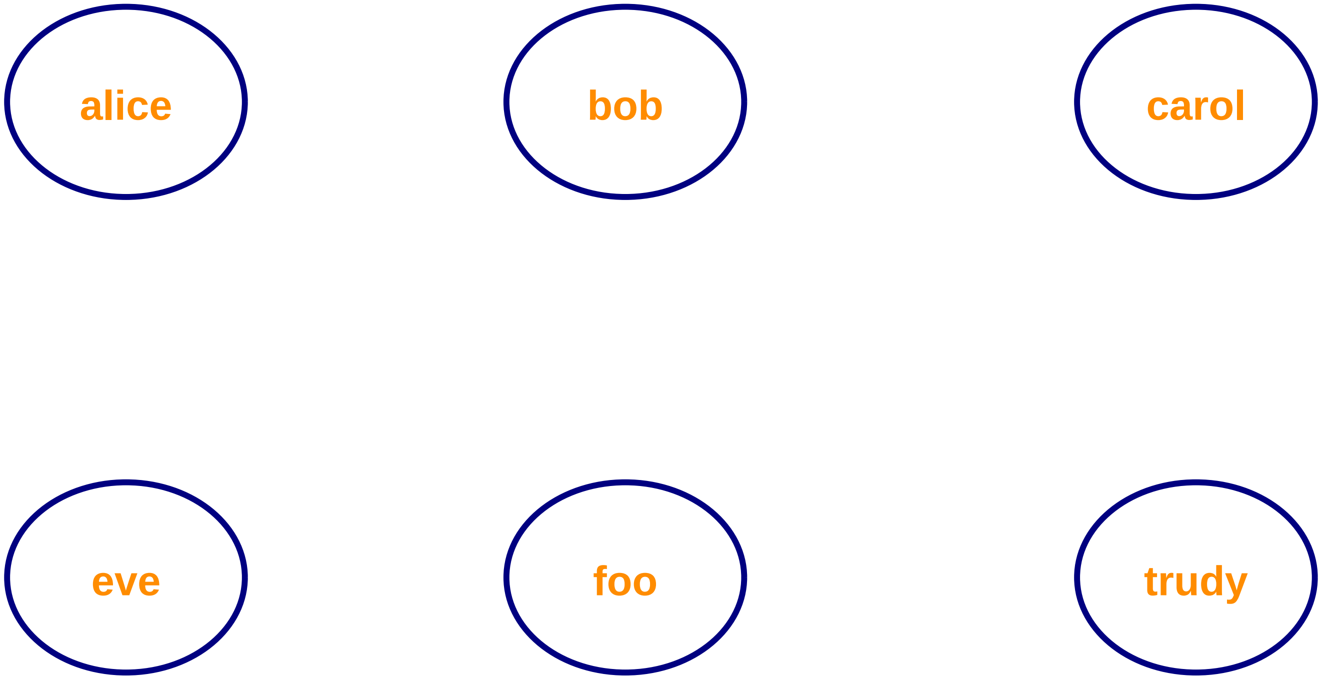
NICE: Services

- Efficient Signaling
- Resource Advertisement and Location
- Secure resource bartering and trading
- Distributed “trust” valuation

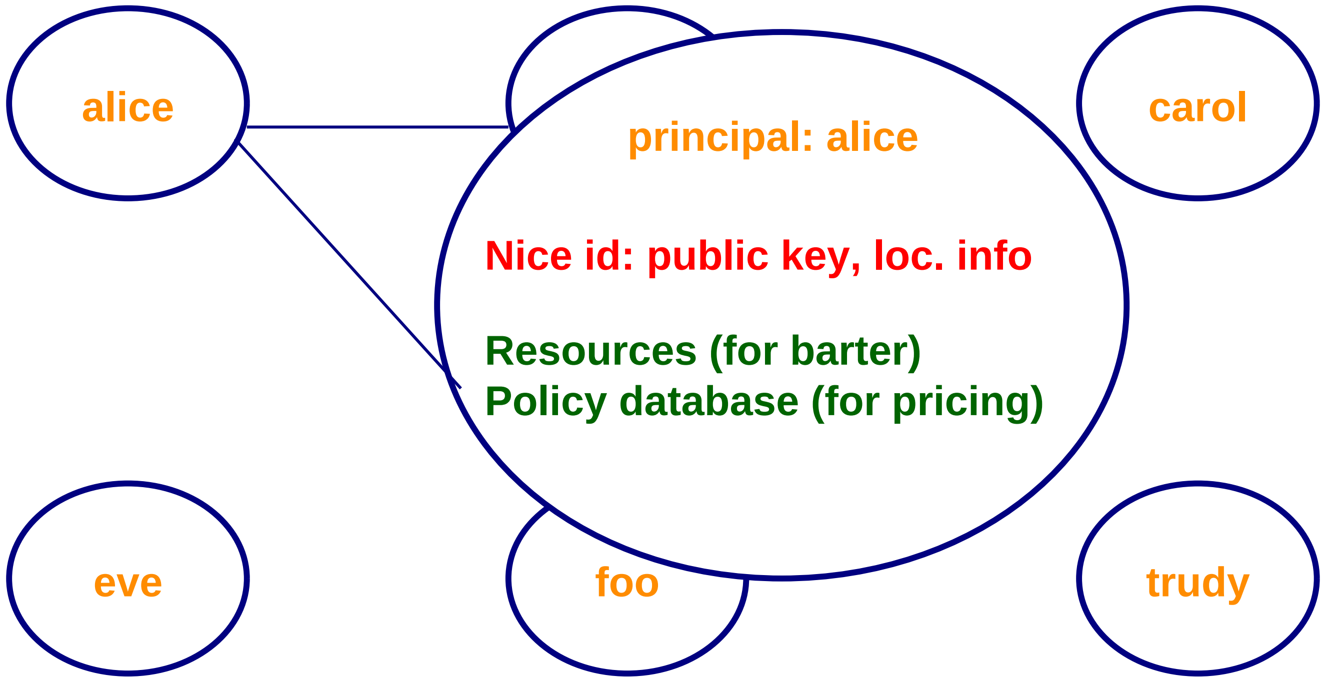
NICE: Preliminaries

- Each user chooses a NICE identifier
  - NICE id contains a public key
  - Key does not need to be published or certified
  - Users may simultaneously use multiple ids
- Each host has a owner
  - Owners set per-host policies
  - Host policies determine resource allocation and pricing

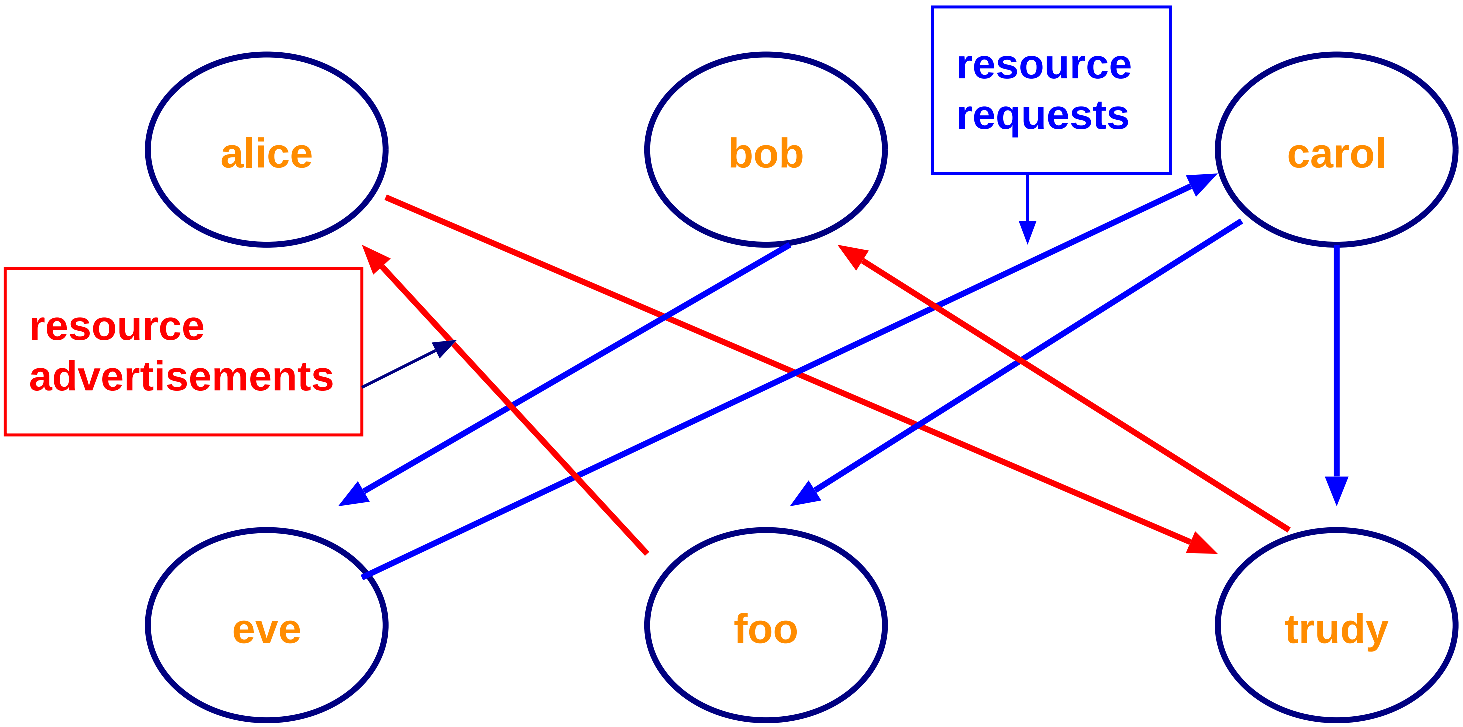
NICE in Operation



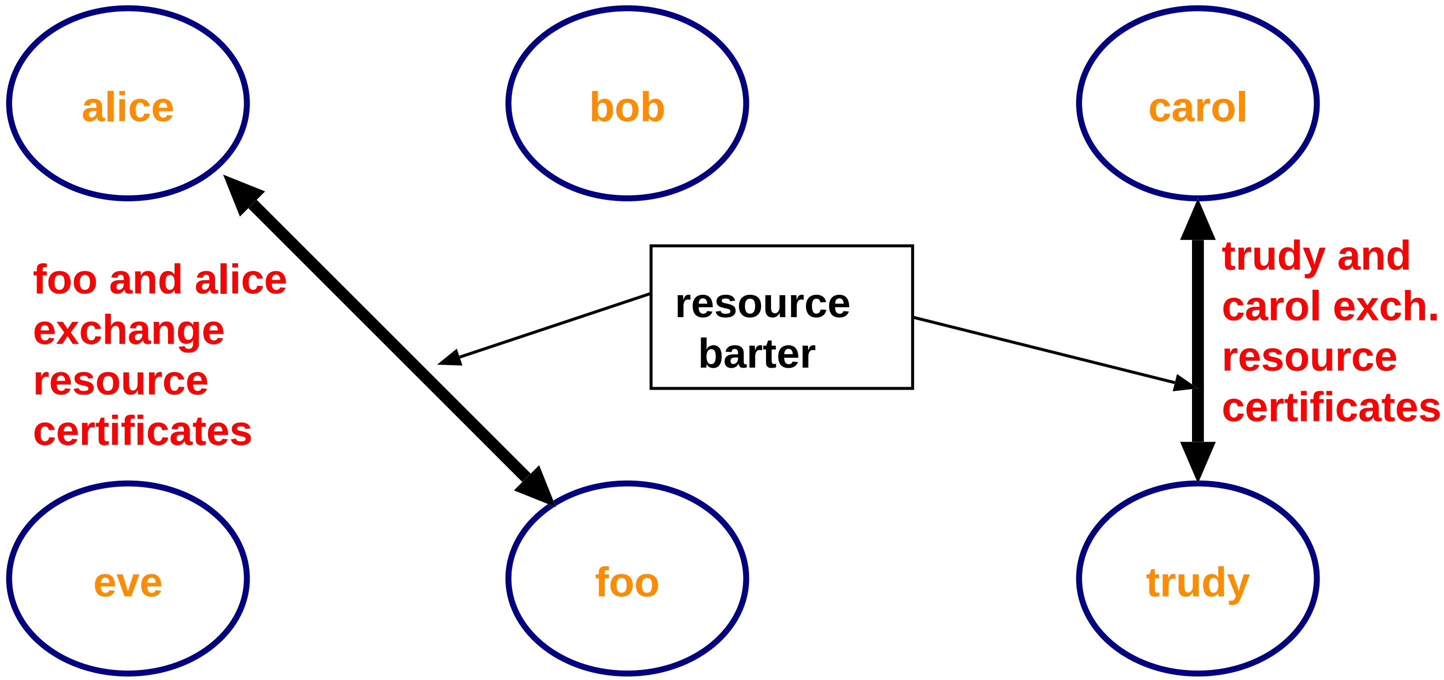
NICE in Operation



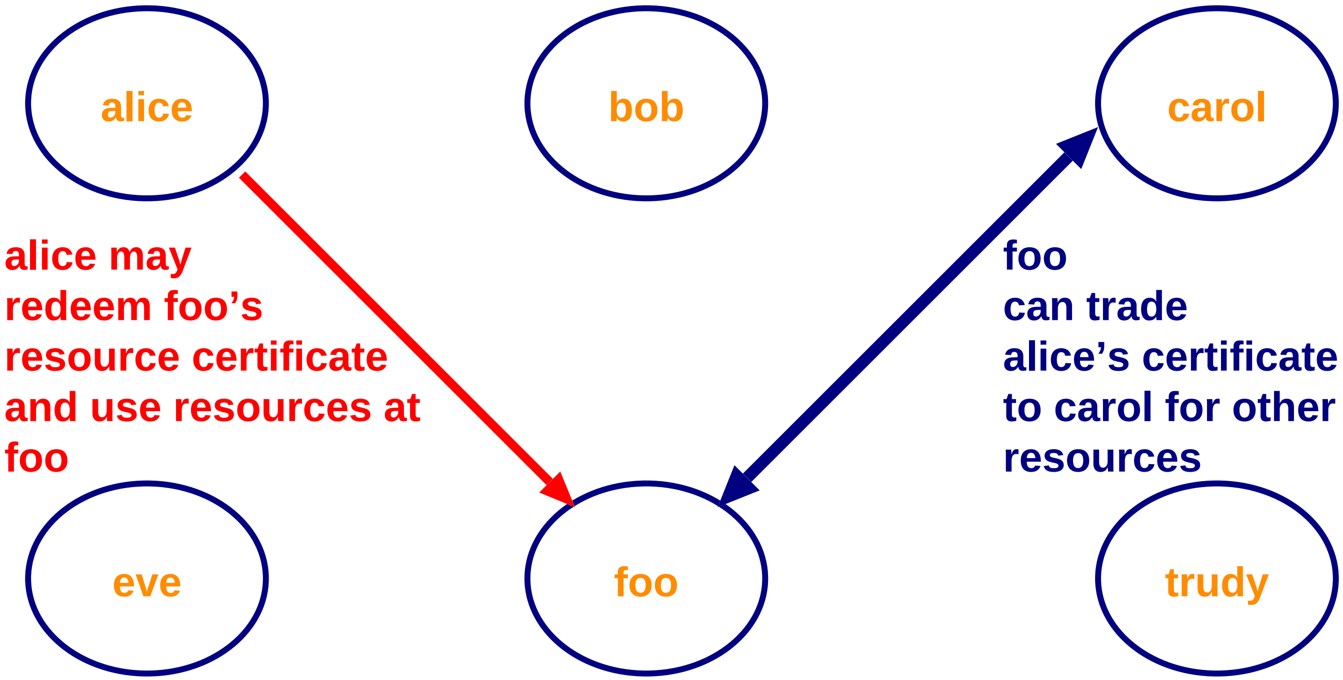
NICE in Operation



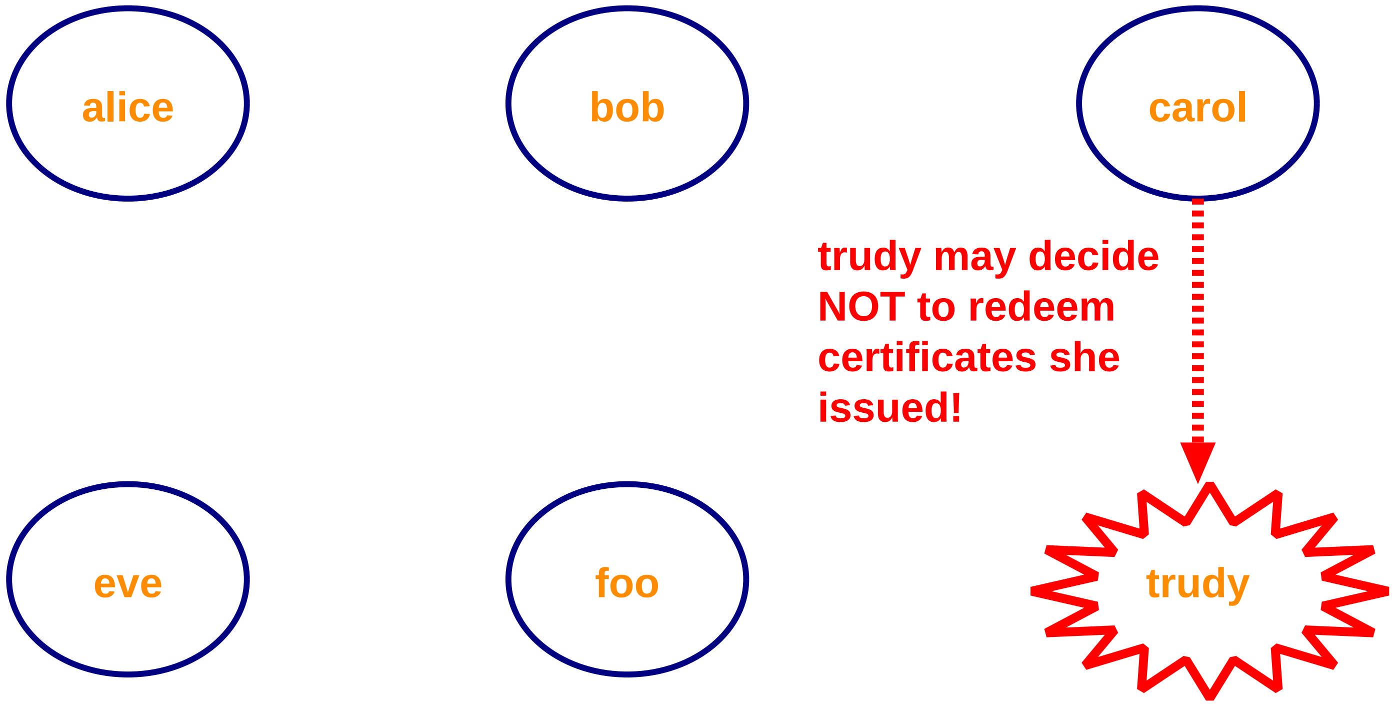
NICE in Operation



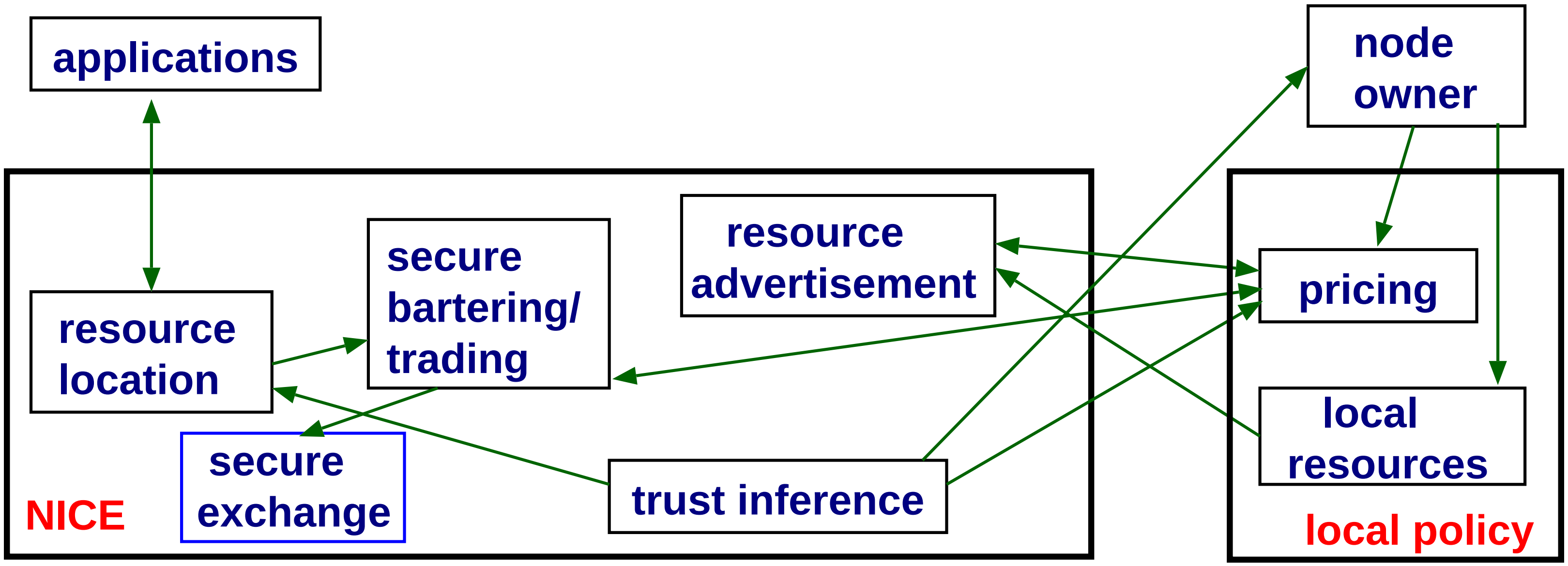
NICE in Operation



NICE in Operation



NICE: Component Architecture



- Users set local resource and pricing policy
- Applications request specific resources
- NICE locates appropriate resources and securely exchanges/trades *resource certificates*
- Resources certificates are redeemed for named resources

Resource Pricing

- Two main objective of default policies:
  - Form robust cooperative groups
  - Not lose large amounts of resources to malicious users
- Policies:
  - Trust-based pricing
  - Trust-based trading limits
- Default policies curb difficult DoS attacks

Trust Evaluation in NICE

- Integrity of entire NICE platform depends on trust computations
- A $\longrightarrow$ B trust is a local measure (at A) of how likely A believes a transaction with B will be successful
- Users can use past experience to assign trust values
  - Trust can also be *inferred* through other trusted users

Goals of Trust Inference Schemes

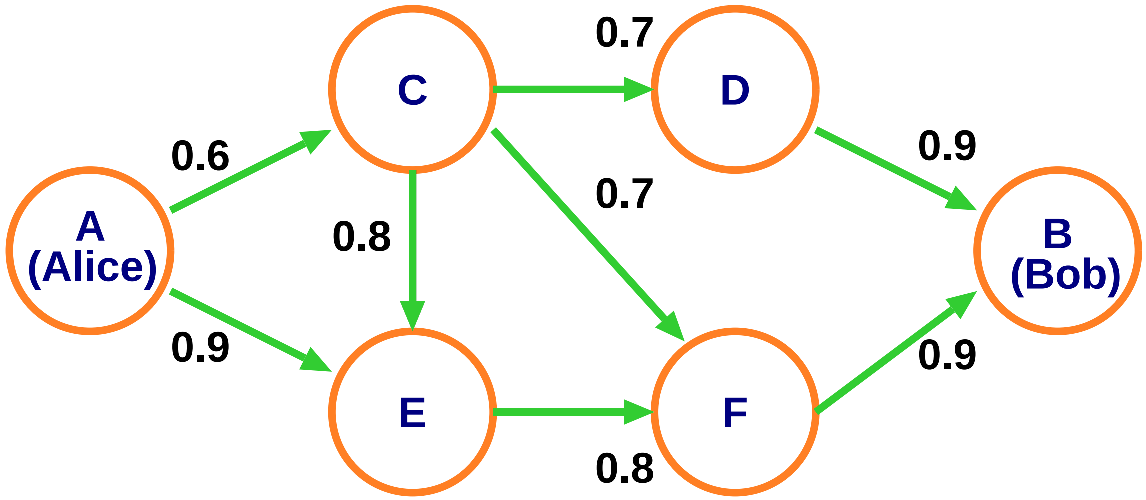
- Users should be able to use local policy to assign trust
- Good nodes should *find* other good nodes efficiently . . .

  . . . and not lose large amounts of resources
- Inference should be resilient against cooperating *malicious* groups
  - Malicious users disseminate arbitrary trust information
  - Good node cliques should be immune to such attacks

Centralized Trust Evaluation

- Trust Graph
    - Vertices are unique user identifiers
    - Directed edges represent how much source trusts dest.
- Many different inference algorithms feasible
    - Strongest Path
    - Weighted sum of disjoint strongest paths

Centralized Evaluation Examples



- Strongest Path: (AEFB, 0.8)
  - Inferred trust: 0.8
- Weighted sum of strongest disjoint paths:
  - Two disjoint paths: (AEFB, 0.8; wt. .9), (ACBD, 0.6; wt. .6)
  - Inferred trust: 0.72

Distributed Trust Inference

- Two main problems to distribute
  - Storage of trust information
  - Efficiently locate relevant edges
- If trust graph can be efficiently reproduced, then users can use any centralized algorithm to infer trust

Storing trust values in Cookies

- Suppose Alice redeems a resource certificate signed by Bob
- Alice assigns a [0,1] value to the transaction *quality* and signs a transaction record — called a cookie
- Bob stores the cookie signed by Alice
  - Alice does *not* keep a record of the transaction!
  - The set of Alice's cookies stored by Bob determines the value of the Alice$\longrightarrow$Bob edge

Using cookies: base case

- Suppose Bob later wants to use Alice's resources:
  - Bob presents Alice a cookie(s) signed by Alice herself
  - Alice can verify her own signature . . .

    . . . and use the cookie values to price her resources

Using Cookies: recursive case

- Bob wants to use Carol's resources but does not have cookies from Carol
  - Suppose Alice does have cookies from Carol
- Bob searches for Carol's cookies amongst users from whom he has cookies (Alice)
- Alice gives Bob a copy of the Carol$\longrightarrow$Alice cookies
- Bob presents Carol with a Carol$\longrightarrow$Bob cookie path

  i.e. Bob produces a $\{$Carol$\longrightarrow$Alice, Alice$\longrightarrow$Bob$\}$ cookie set
- Carol can now infer a trust value for Bob

Properties of cookie-based trust storage

- If Bob wants to use resources at Carol, *he* has to initiate a cookie search
  - Guards against a DoS attack
- Bob only stores statements of the form "*X* trusts Bob"
  - Clearly, Bob will discard any low valued cookies he gets
  - Users store cookies most beneficial to their own cause
- Transaction record storage is completely distributed
  - Fabricated transactions don't affect legitimate users

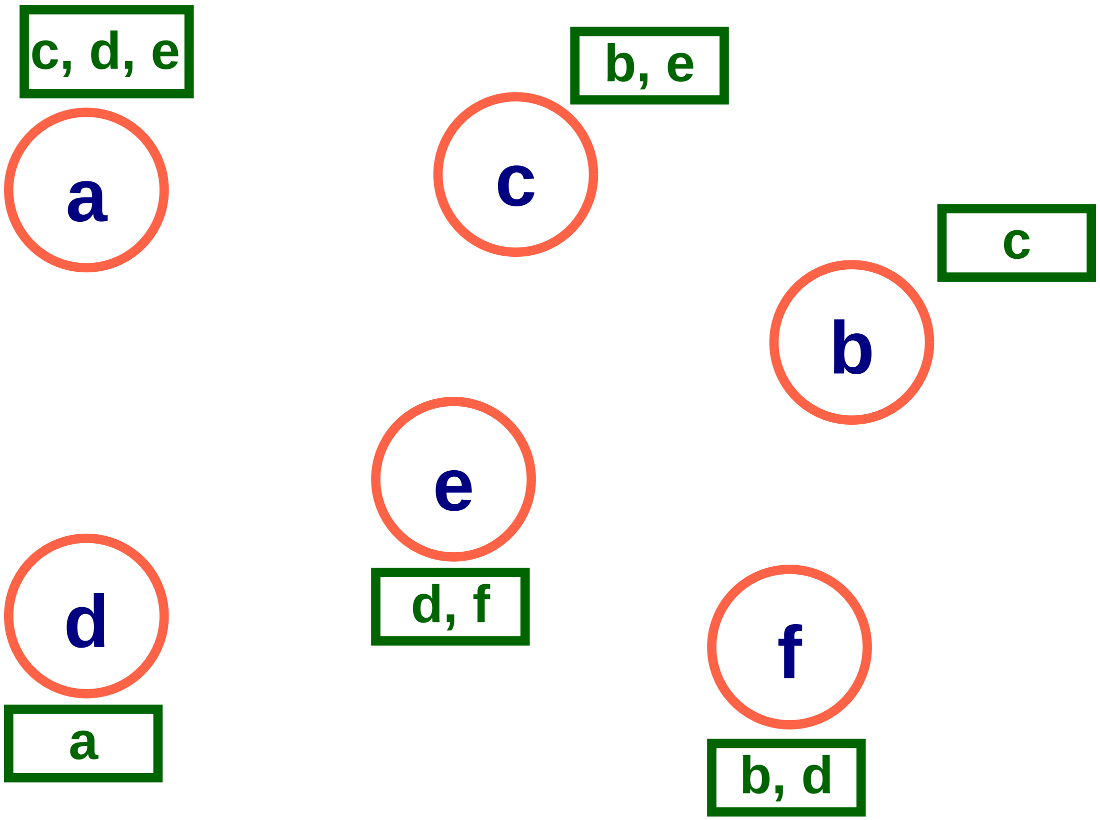
Properties of cookie-based trust storage

- If Bob wants to use resources at Carol, *he* has to initiate a cookie search
  - Guards against a DoS attack
- Bob only stores statements of the form "*X* trusts Bob"
  - Clearly, Bob will discard any low valued cookies he gets
  - Users store cookies most beneficial to their own cause
- Transaction record storage is completely distributed
  - Fabricated transactions don't affect legitimate users

How to locate cookies?

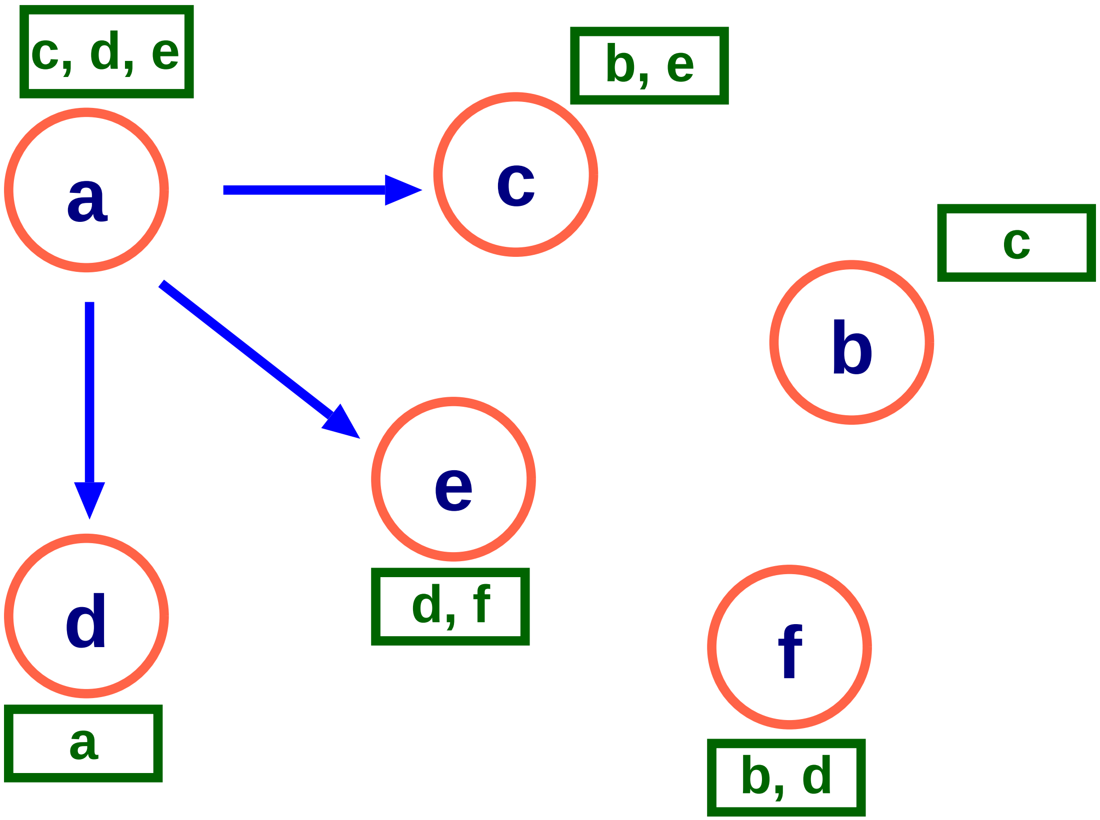
easy answer: flood queries for specific cookies

Flooding Example

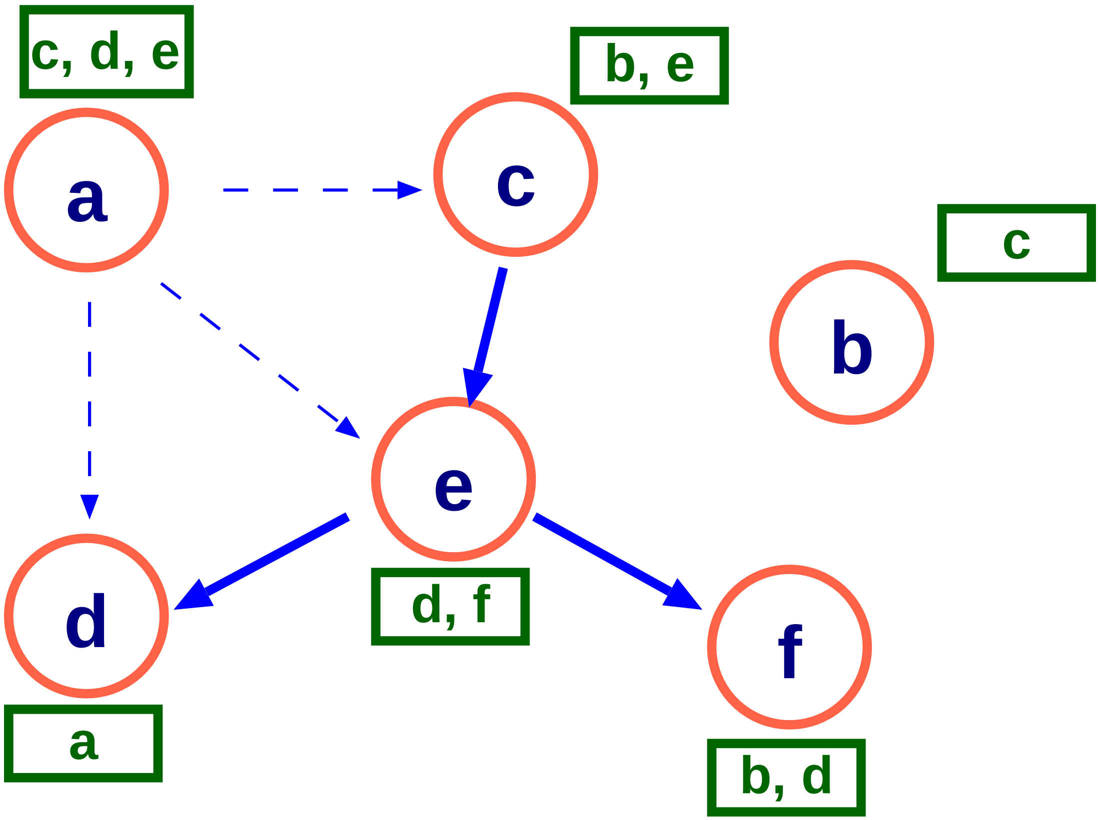


- Initial cookie state

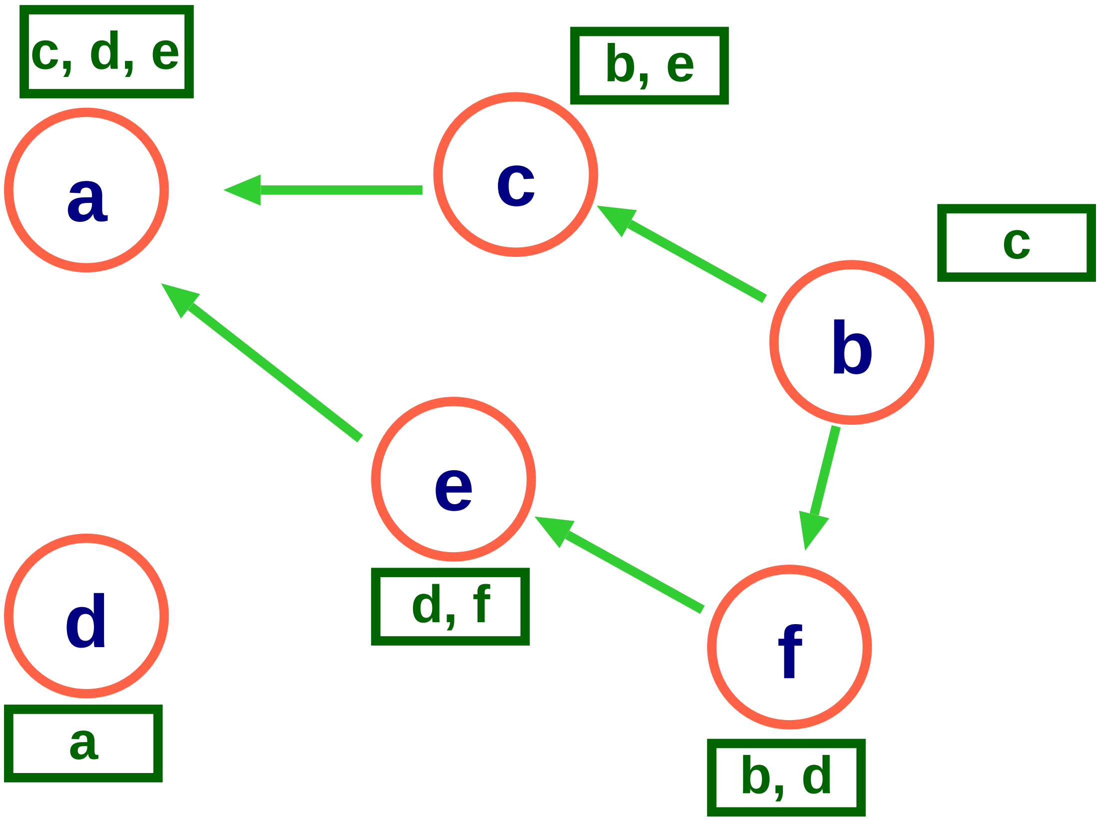
Flooding Example



Flooding Example



Flooding Example



- The b⟶a component of the trust graph is reconstructed via flooding

Analysis

- Flooding is “guaranteed” to reconstruct relevant trust graph component
- Problems:
    - Inefficient
    - Bad nodes can erase information about failed transactions by simply deleting low valued cookies!

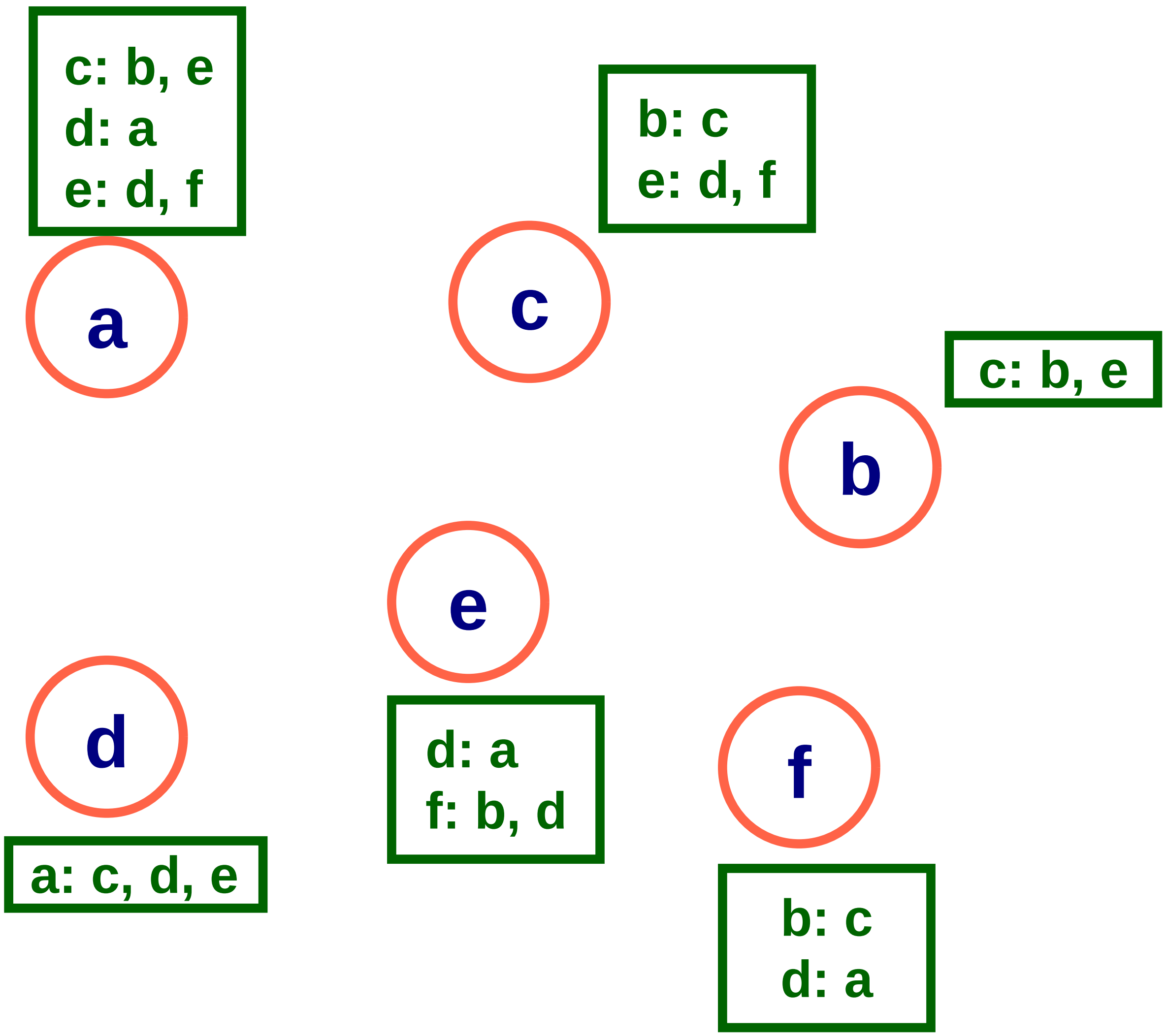
Refinements

- Search efficiency
  - Use cookie-digests to direct searches
  - Limit search outdegree
- Store failed transaction information in “negative cookies”
- Quickly discover good nodes using “preference lists”

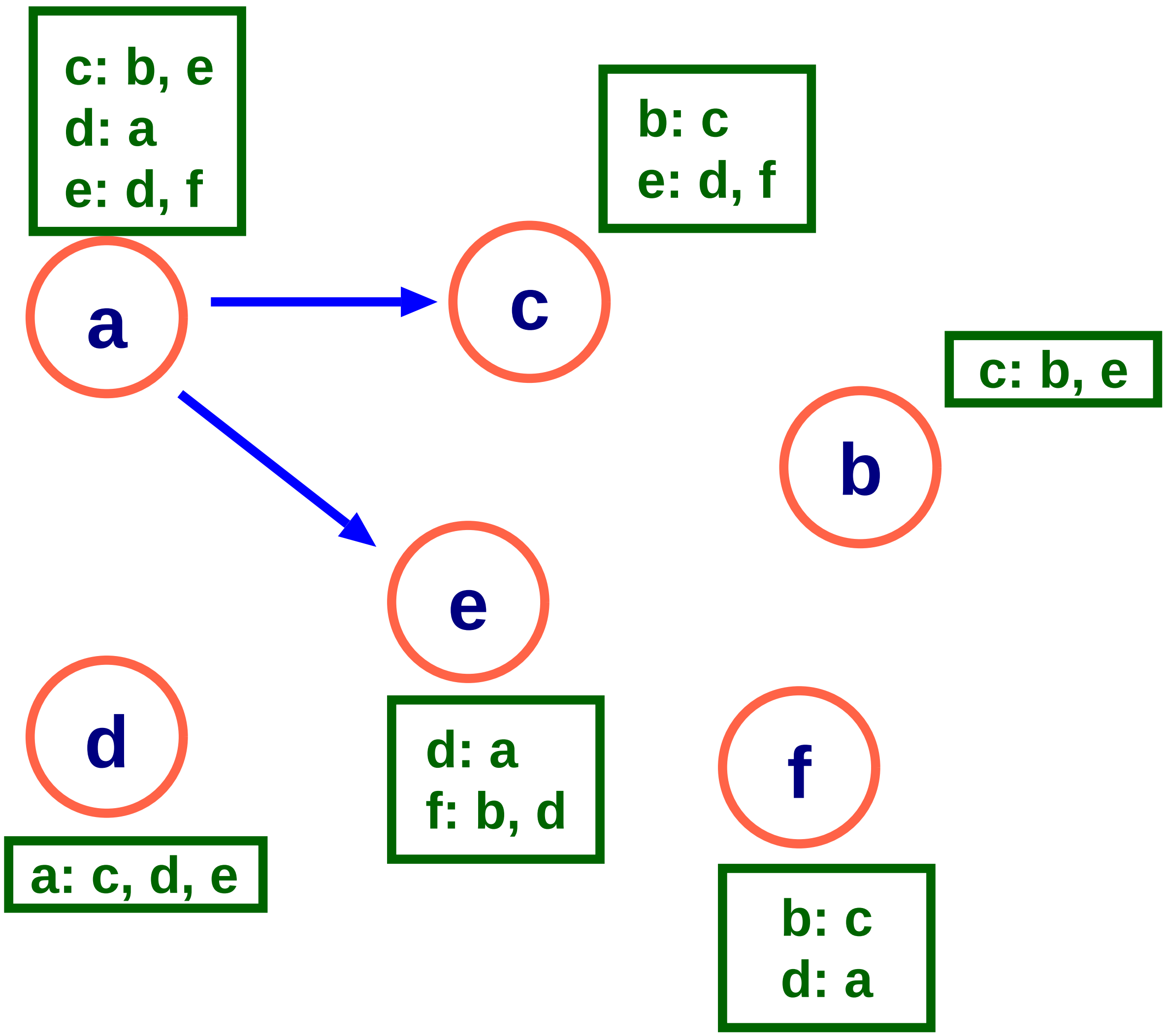
Using digests to speed up search

- Suppose Alice gives a cookie to Bob
- Along with the cookie, Alice also gives Bob a digest of all users from whom *she* has cookies
  - Cookie digests efficiently implemented using Bloom filters
- Searches proceed along random edges only for a hop or two
  - After a random search phase, searches are forwarded only if there is a hit in a cookie digest

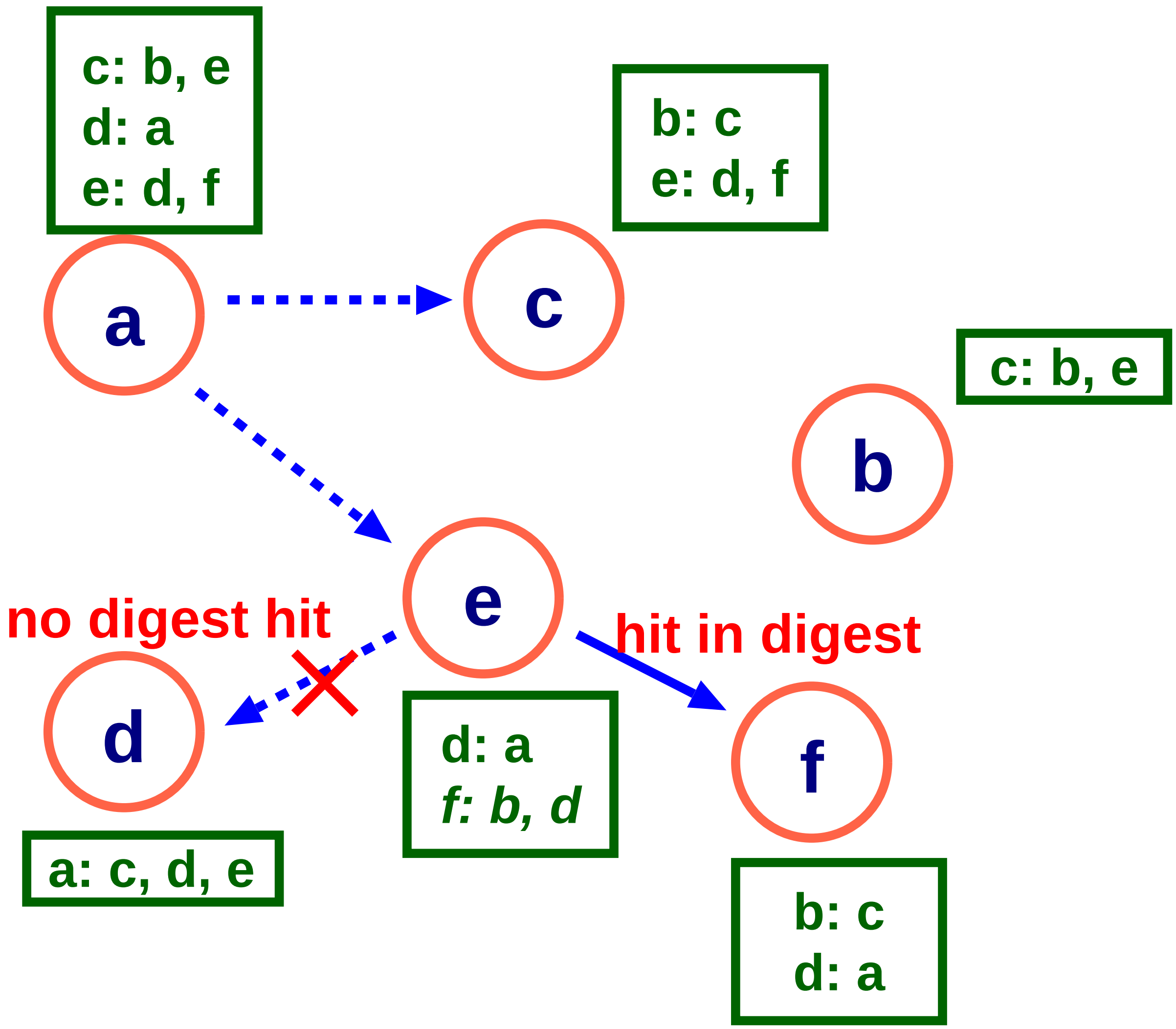
Digest-based search example



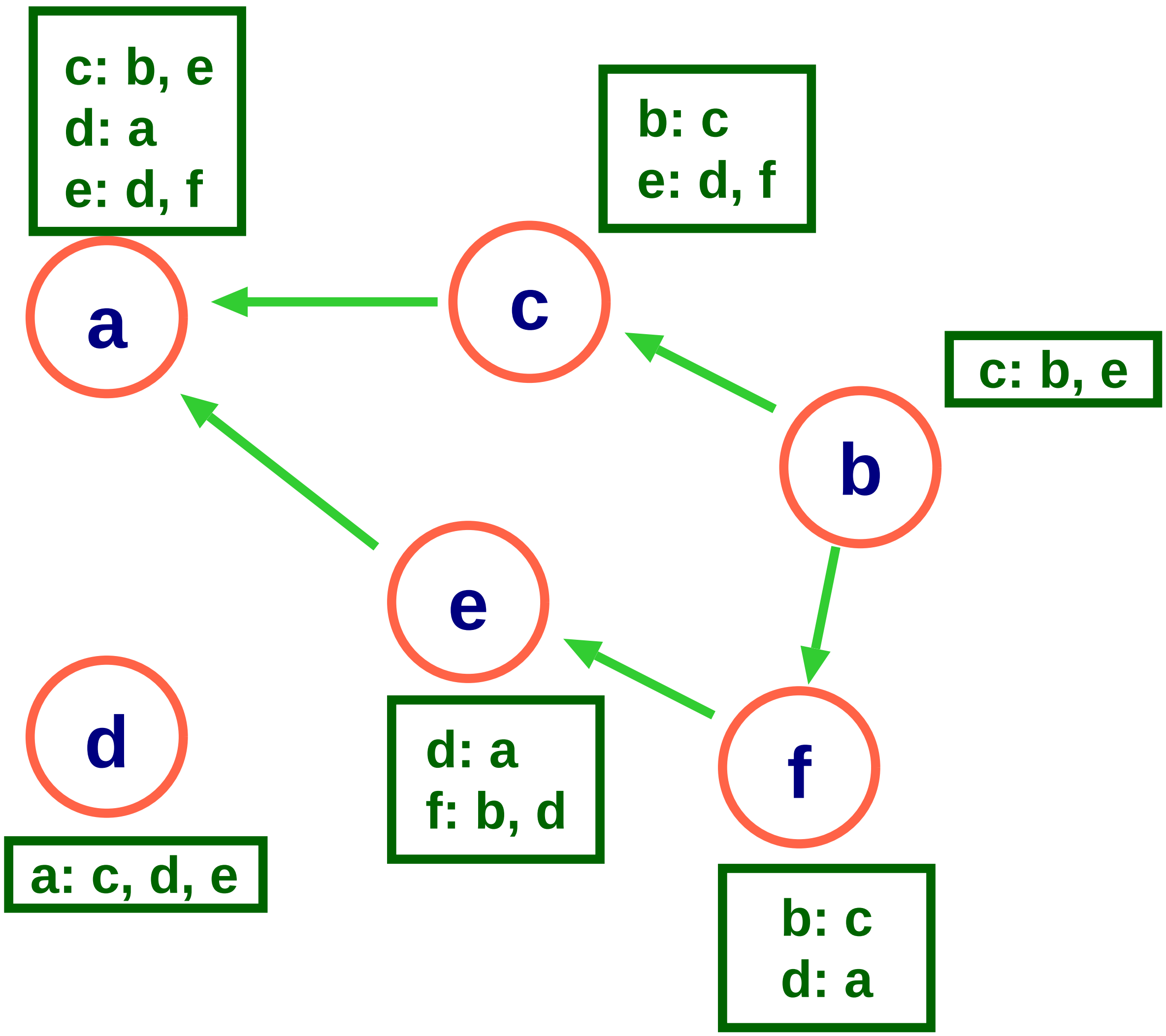
Digest-based search example



Digest-based search example



Digest-based search example



Negative Cookies

- Suppose Eve uses Alice's resources, but does not provide the negotiated resources she promised
- In the original scheme, Eve would receive a low-valued cookie from Alice. . . and promptly discard it
- Instead, Alice stores the low-valued "negative" cookie *herself*
  - Alice won't trust Eve as long as she stores the negative cookie
- The negative cookie can also be used by Bob (who trusts Alice)
  - Before accepting a transaction with Eve, Bob searches for negative cookies for Eve at users he trusts

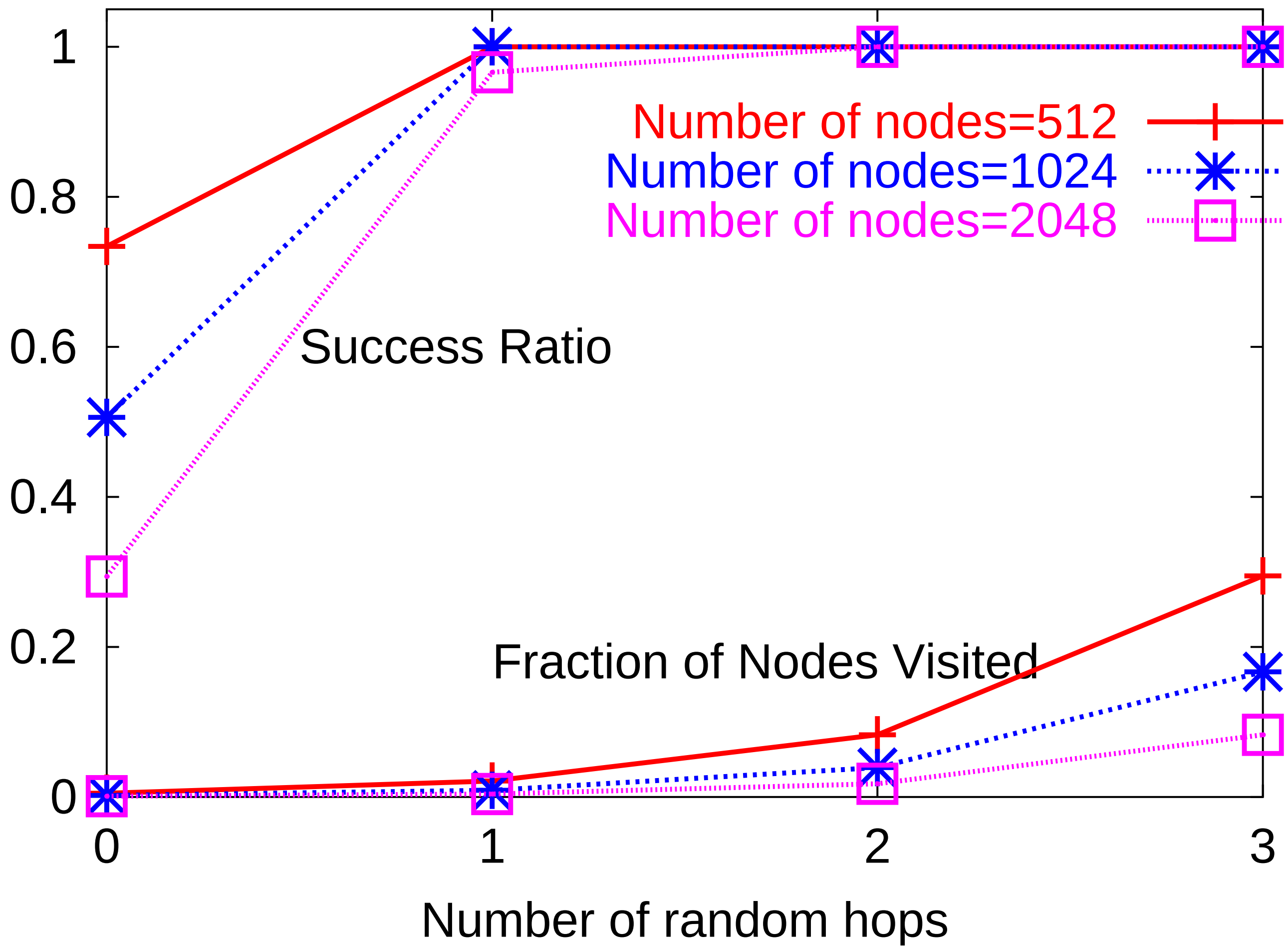
Preference Lists

- To quickly discover other good nodes, each user (say Bob) keeps a preference list
- Bob's preference list contains potential high trust nodes (with whom Bob has not interacted yet)
- Bob interacts with nodes in the preference list with higher probability
- As Bob discovers new nodes during cookie searches, they are included in Bob's preference list iff they have high trust values

Results: Experimental Setup

- Simulations on 64-2K node groups
- Simulations include good and bad users
- Two types of results:
    - Scalability
    - Robustness

Scalability



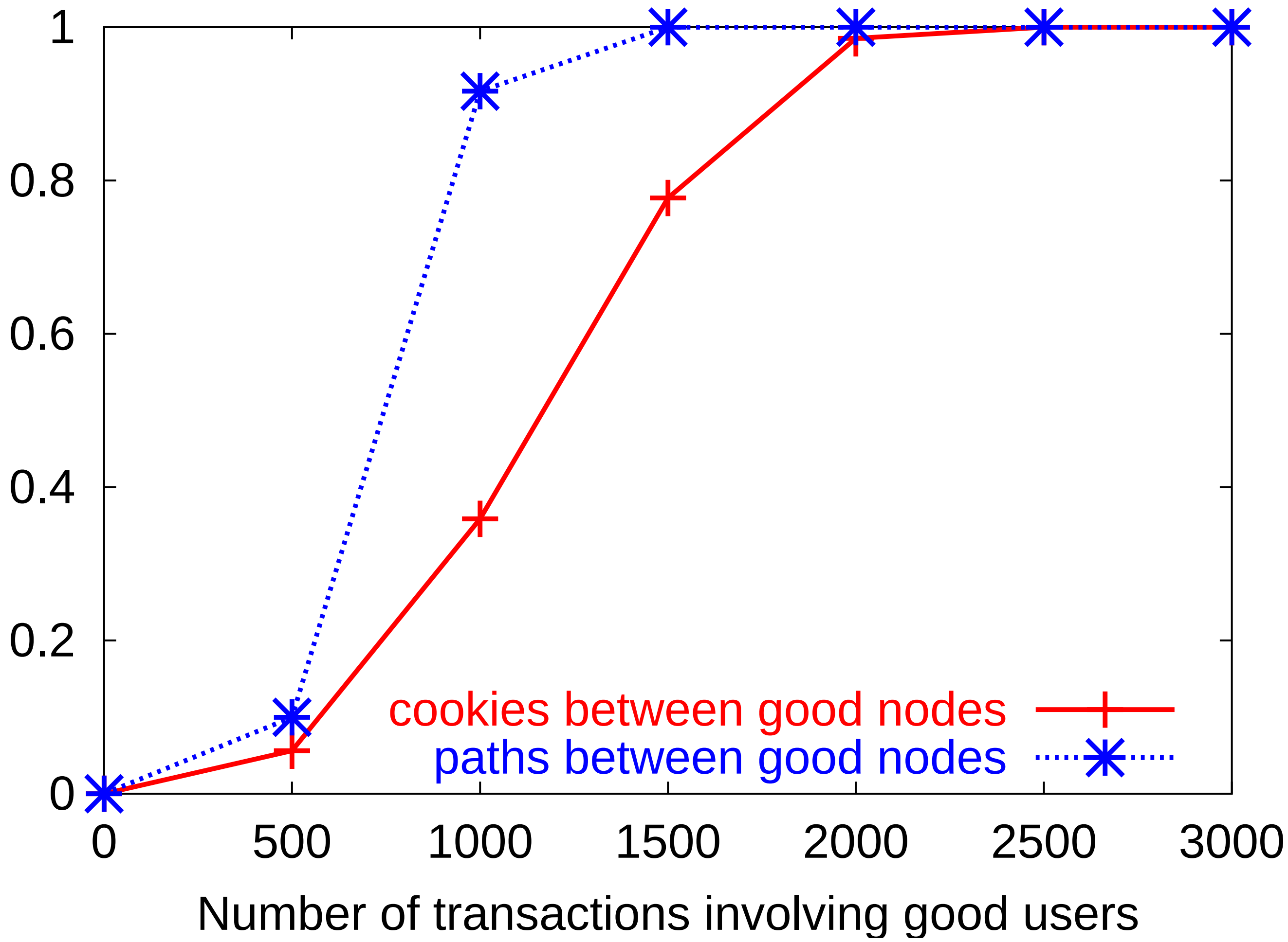
Good, Bad, and Regular users

- Three user models:
  - Good users: implement entire protocol correctly
    - Good-good transactions always result in 1.0 valued cookie
  - Regular users: implement entire protocol correctly
    - Good-regular transactions result in [0,1] range cookies
    - Mean cookie value: 0.7
  - Bad users: form a clique before simulation begins
    - Always report 1.0 value for all bad users
    - Bad-other transactions produce 1.0 cookies with prob. .5 . . .
    - . . . and produce a 0.05 valued cookie with prob. 0.5

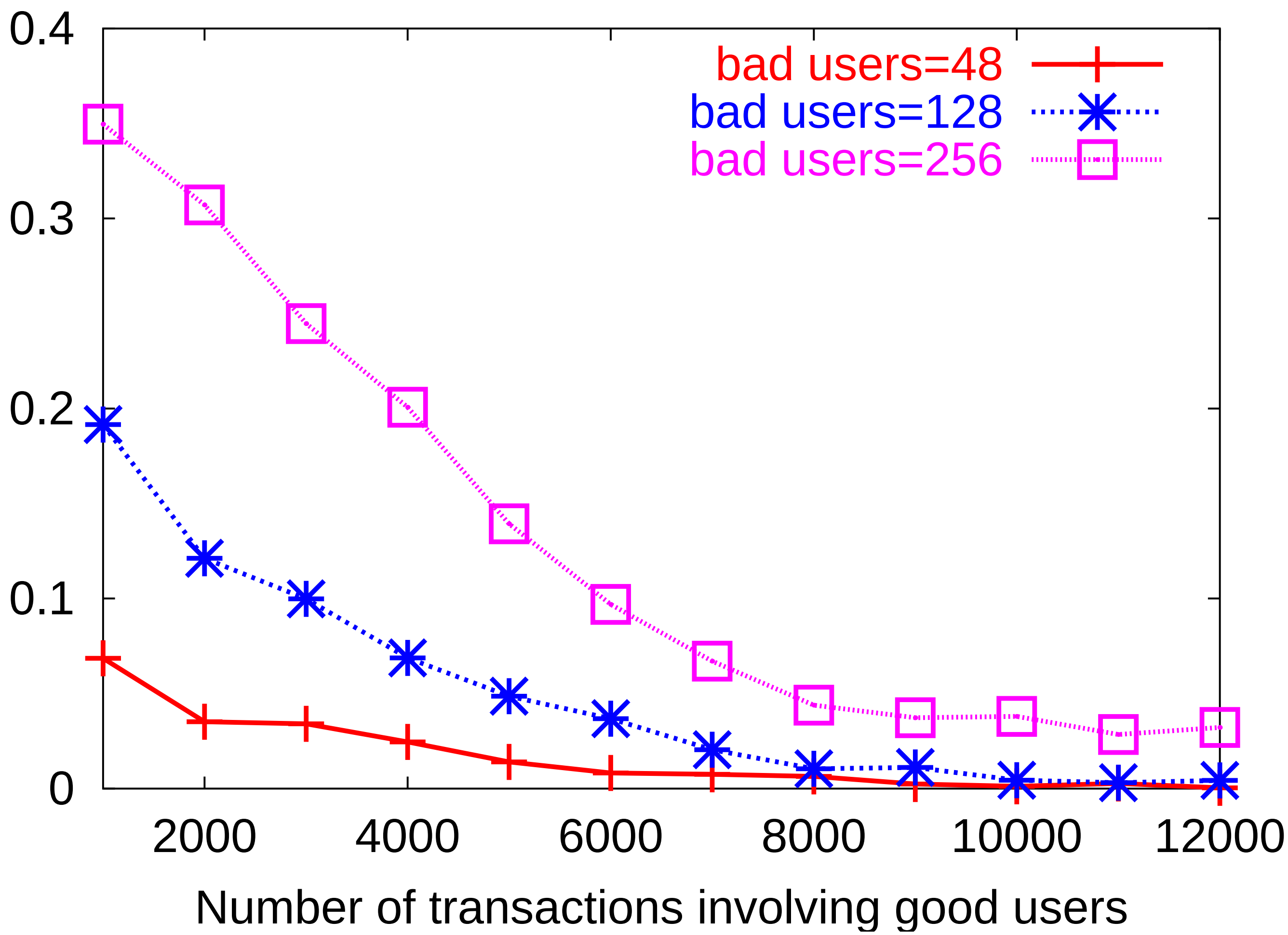
Simulation setup

- Simulation includes preference lists, negative cookies, and digests
- At each time step, a user (Alice) is chosen
- Alice chooses another user (Bob) from her pref. list to start a transaction
- Alice-Bob transaction proceeds if Alice can find a 0.85 Alice-Bob path (and if Bob cannot find a negative cookie for Alice)
- After two unsuccessful tries with different users, the simulator allows a transaction without checking Alice's credentials

Behavior with Regular Users



Failed Transactions with Bad Users



Related Work

- Centralized trust inference
  - Mojonation
  - Intertrust DRM
  - Commercial web sites, e.g. e-bay, avogato. . .
- Distributed trust inference
  - PGP (one hop reference)
- p2p database to store user complaints, Aberer et. al.
- Previous work in centralized trust inference
  - Direct experience & reputation-based inference (Abdul-Rahman et. al.)

Status

- Initial scheme will be presented at INFOCOM 2003.
- Current work on applying scheme to existing systems, specifically quotas in CFS, and robust routing in DHTs
- Prototype implementation underway