Outline

- Introduction
- Background
- Distributed DBMS Architecture
- Distributed Database Design
- Distributed Query Processing
- Distributed Transaction Management
- Building Distributed Database Systems (RAID)
- Mobile Database Systems
- Privacy, Trust, and Authentication
- Peer to Peer Systems

Useful References

- Y. Lu, W. Wang, D. Xu, and B. Bhargava, *Trust-Based Privacy Preservation for Peer-to-peer*, in the 1st NSF/NSA/AFRL workshop on secure knowledge management (SKM), Buffalo, NY, Sep. 2004.

Problem statement

- Privacy in peer-to-peer systems is different from the anonymity problem
- Preserve privacy of requester
- A mechanism is needed to remove the association between the identity of the requester and the data needed

Proposed solution

- A mechanism is proposed that allows the peers to acquire data through trusted proxies to preserve privacy of requester
  - The data request is handled through the peer's proxies
  - The proxy can become a supplier later and mask the original requester

Related work

- Trust in privacy preservation
  - Authorization based on evidence and trust
  - Developing pervasive trust
- Hiding the subject in a crowd
  - K-anonymity
  - Broadcast and multicast

Related work (2)

- Fixed servers and proxies
  - Publius
- Building a multi-hop path to hide the real source and destination
  - FreeNet
  - Crowds
  - Onion routing

Related work (3)

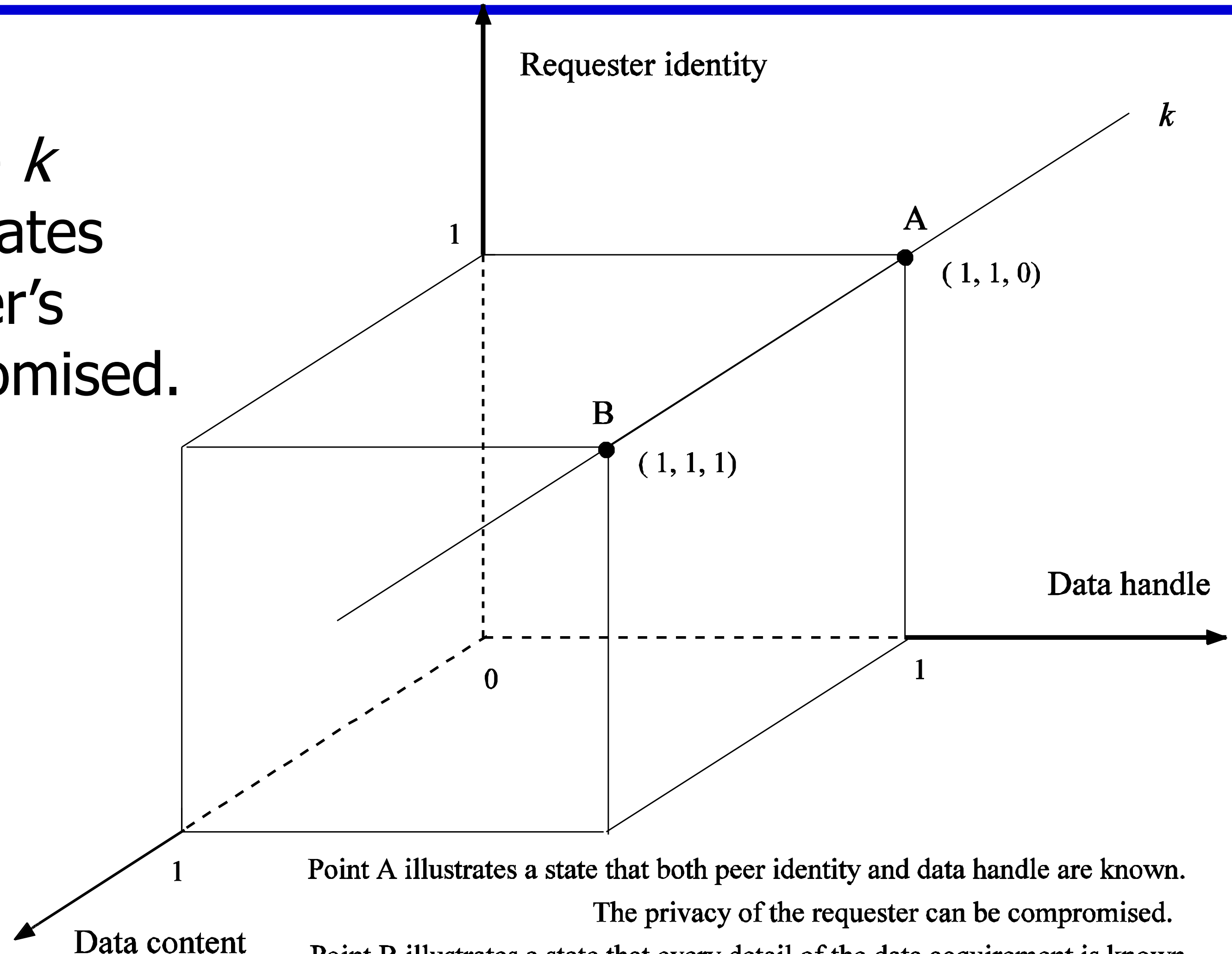
- p^5
  - p^5 provides sender-receiver anonymity by transmitting packets to a broadcast group
- Herbivore
  - Provides provable anonymity in peer-to-peer communication systems by adopting dining cryptographer networks

Privacy measurement

- A tuple <requester ID, data handle, data content> is defined to describe a data acquirement.
- For each element, “0” means that the peer knows nothing, while “1” means that it knows everything.
- A state in which the requester’s privacy is compromised can be represented as a vector <1, 1, y>, (y Є [0,1]) from which one can link the ID of the requester to the data that it is interested in.

Privacy measurement (2)

For example, line *k* represents the states that the requester's privacy is compromised.



Point A illustrates a state that both peer identity and data handle are known. The privacy of the requester can be compromised.

Point B illustrates a state that every detail of the data acquirement is known.

Mitigating collusion

- An operation "*" is defined as:

$$< c_1, c_2, c_3 >=< a_1, a_2, a_3 > * < b_1, b_2, b_3 >$$

$$c_i = \begin{cases} \max(a_i, b_i), & a_i \neq 0 \;\; and \;\; b_i \neq 0; \\ 0, & otherwise. \end{cases}$$

- This operation describes the revealed information after a collusion of two peers when each peer knows a part of the "secret".
- The number of collusions required to compromise the secret can be used to evaluate the achieved privacy

Trust based privacy preservation scheme

- The requester asks one proxy to look up the data on its behalf. Once the supplier is located, the proxy will get the data and deliver it to the requester
  - Advantage: other peers, including the supplier, do not know the real requester
  - Disadvantage: The privacy solely depends on the trustworthiness and reliability of the proxy

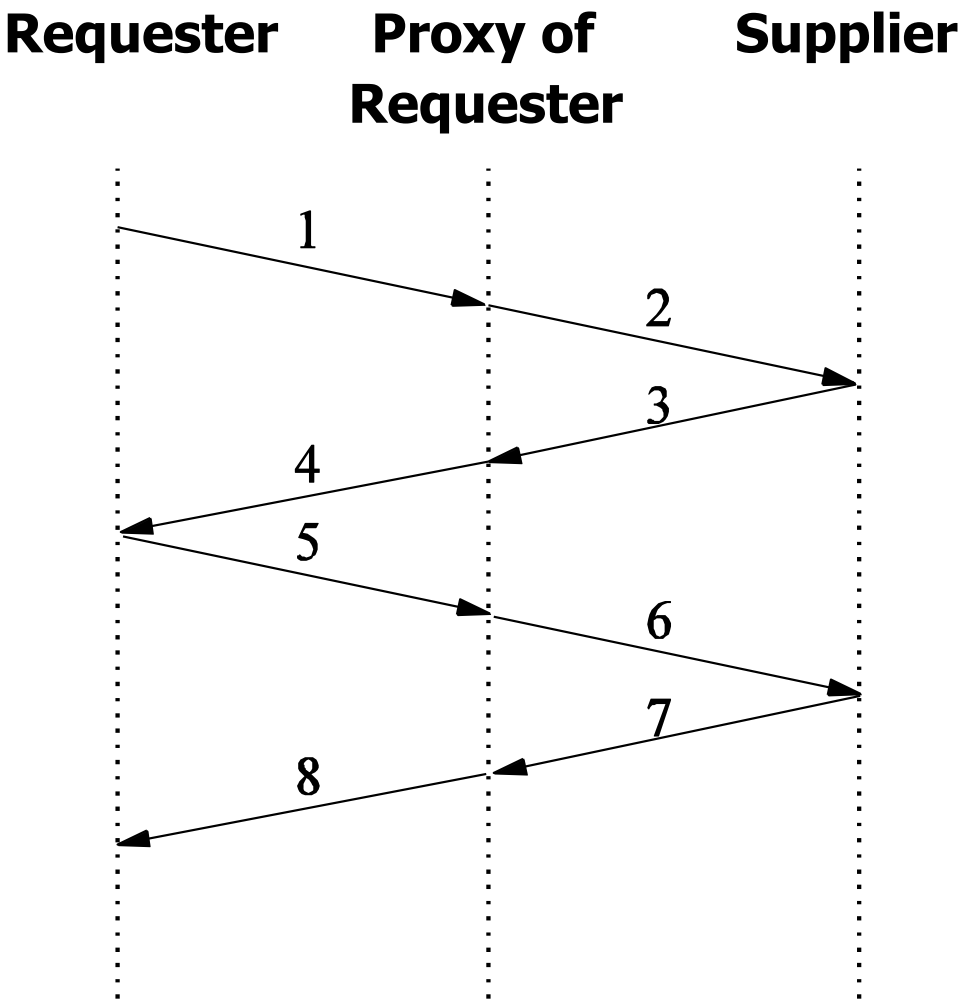
Trust based scheme – Improvement 1

- To avoid specifying the data handle in plain text, the requester calculates the hash code and only reveals a part of it to the proxy.
- The proxy sends it to possible suppliers.
- Receiving the partial hash code, the supplier compares it to the hash codes of the data handles that it holds. Depending on the revealed part, multiple matches may be found.
- The suppliers then construct a bloom filter based on the remaining parts of the matched hash codes and send it back. They also send back their public key certificates.

Trust based scheme – Improvement 1

- Examining the filters, the requester can eliminate some candidate suppliers and finds some who may have the data.
- It then encrypts the full data handle and a data transfer key k_{data} with the public key.
- The supplier sends the data back using k_{data} through the proxy
- Advantages:
  - It is difficult to infer the data handle through the partial hash code
  - The proxy alone cannot compromise the privacy
  - Through adjusting the revealed hash code, the allowable error of the bloom filter can be determined

Data transfer procedure after improvement 1



R: requester *S*: supplier

Step 1, 2: *R* sends out the partial hash code of the data handle

Step 3, 4: *S* sends the bloom filter of the handles and the public key certificates

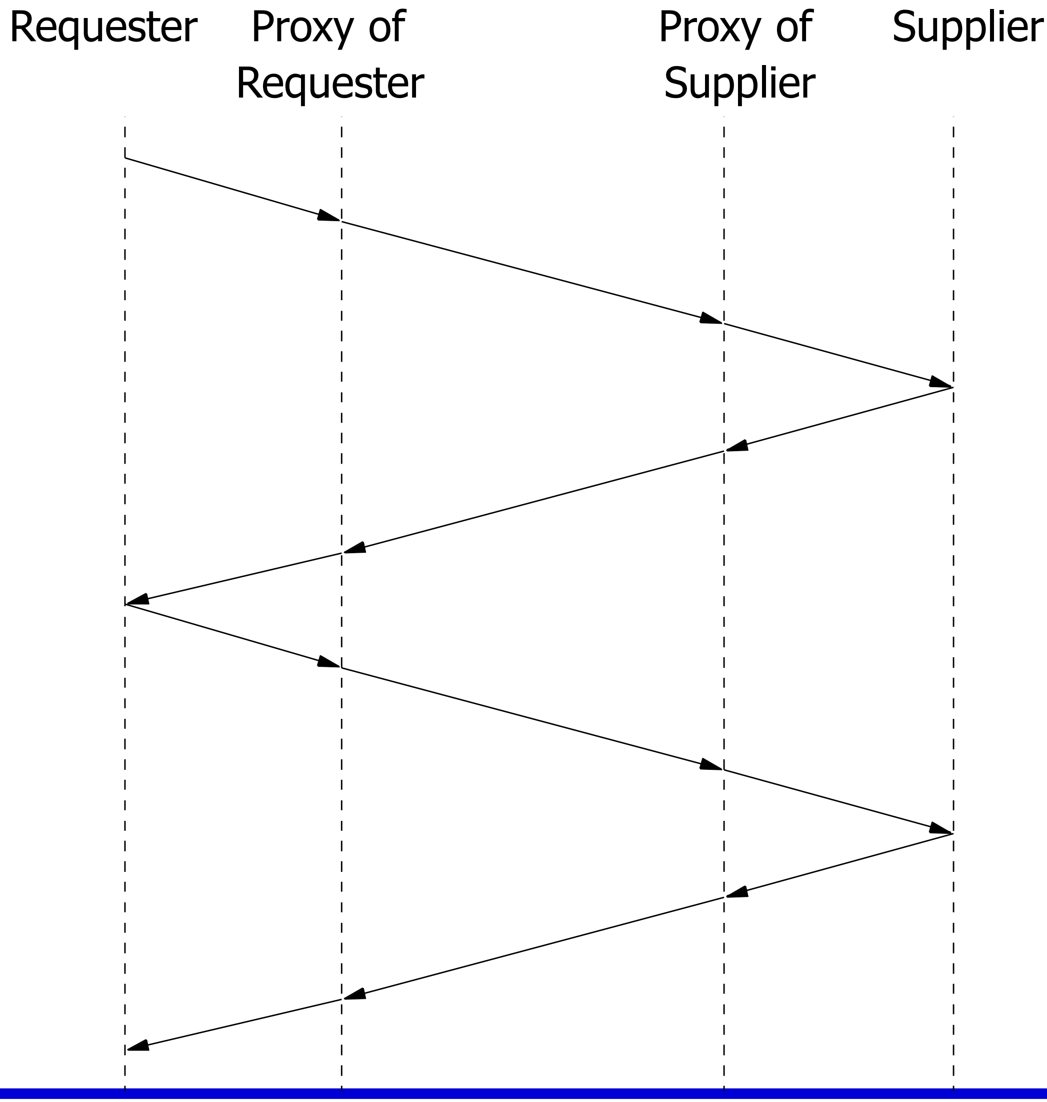
Step 5, 6: *R* sends the data handle and k_{Data} encrypted by the public key

Step 7, 8: *S* sends the required data encrypted by k_{Data}

Trust based scheme – Improvement 2

- The above scheme does not protect the privacy of the supplier
- To address this problem, the supplier can respond to a request via its own proxy

Trust based scheme – Improvement 2



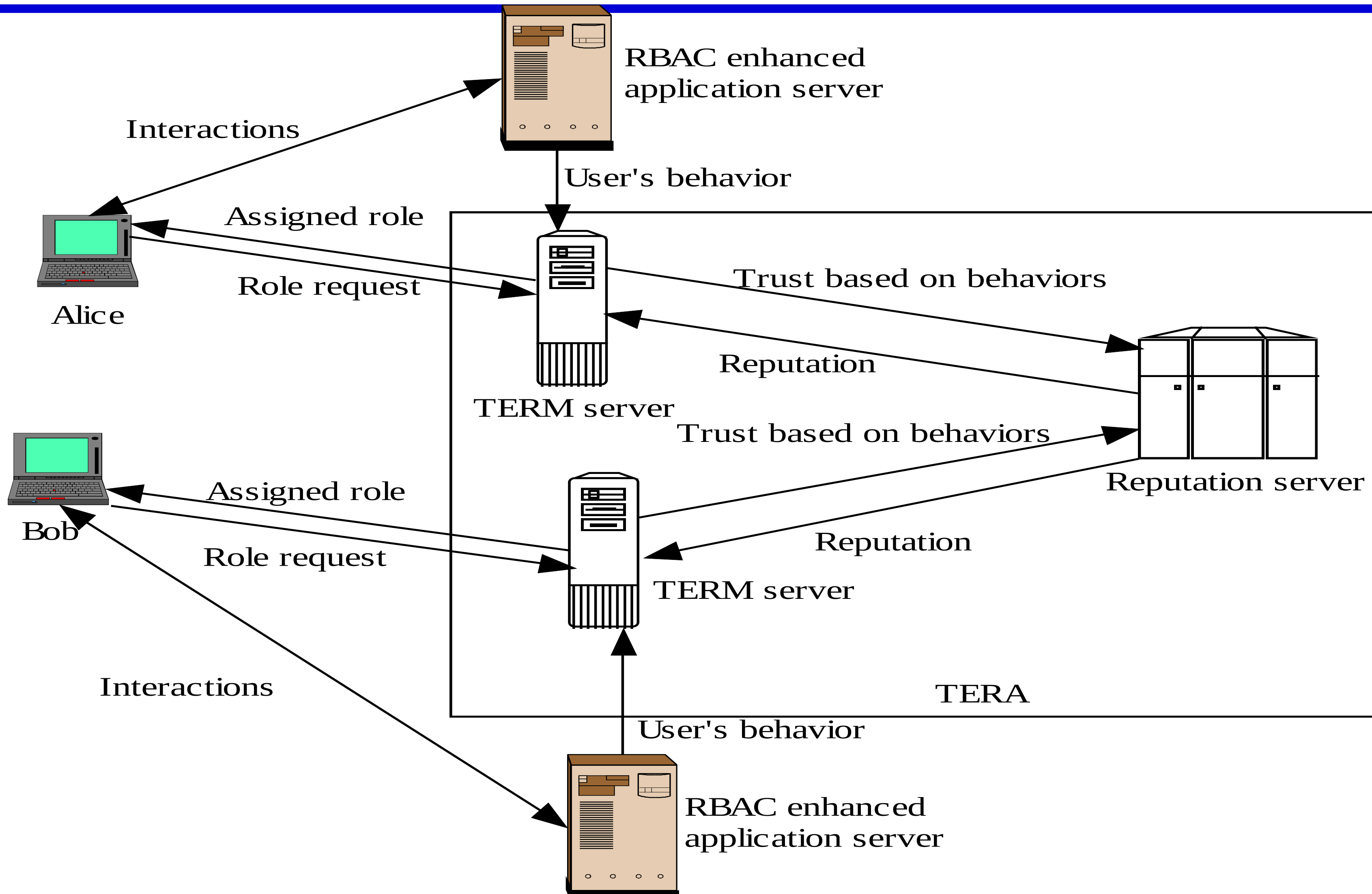
Trustworthiness of peers

- The trust value of a proxy is assessed based on its behaviors and other peers' recommendations
- Using Kalman filtering, the trust model can be built as a multivariate, time-varying state vector

Experimental platform - TERA

- Trust enhanced role mapping (TERM) server assigns roles to users based on
  - Uncertain & subjective evidences
  - Dynamic trust
- Reputation server
  - Dynamic trust information repository
  - Evaluate reputation from trust information by using algorithms specified by TERM server

Trust enhanced role assignment architecture (TERA)



Conclusion

- A trust based privacy preservation method for peer-to-peer data sharing is proposed
- It adopts the proxy scheme during the data acquirement
- Extensions
  - Solid analysis and experiments on large scale networks are required
  - A security analysis of the proposed mechanism is required

Peer to Peer Systems and Streaming

Useful References

- G. Ding and B. Bhargava, *Peer-to-peer File-sharing over Mobile Ad hoc Networks*, in the First International Workshop on Mobile Peer-to-Peer Computing, Orlando, Florida, March 2004.
- M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, *PROMISE: Peer-to-Peer Media Streaming Using CollectCast*, In Proc. of ACM Multimedia 2003, 45-54, Berkeley, CA, November 2003.

Overview of Peer-to-Peer (P2P) Systems

- Peer
  - Autonomy: no central server
  - Similar power
- Share resources among a large number of peers
- P2P is a distributed system where peers collaborate to accomplish tasks

P2P Applications

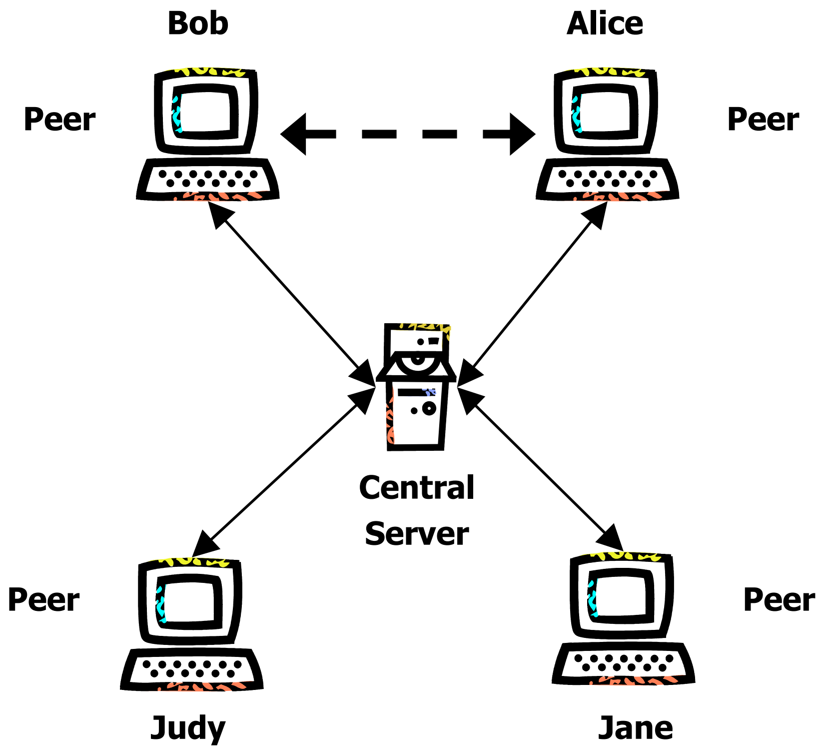
- P2P file-sharing
  - Napster, Gnutella, KaZaA, eDonkey, etc.
- P2P Communication
  - Instant messaging
  - Mobile Ad hoc network
- P2P Computation
  - Seti@home

P2P Searching Algorithms

- Search for file, data, or peer
- Unstructured
  - Napster, Gnutella, KaZaA, eDonkey, etc.
- Structured
  - Chord, Pastry, Tapestry, CAN, etc.

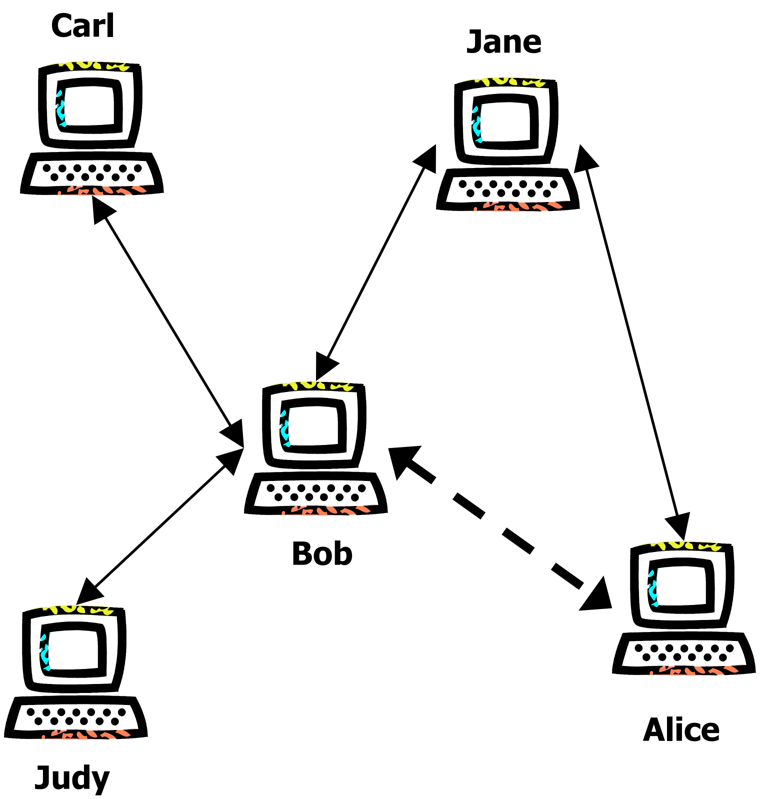
Napster: Central Directory Server

- Bob wants to contact Alice, he must go through the central server
- Benefits:
  - Efficient search
  - Limited bandwidth usage
  - No per-node state
- Drawbacks:
  - Central point of failure
  - Limited scale
  - Copyrights



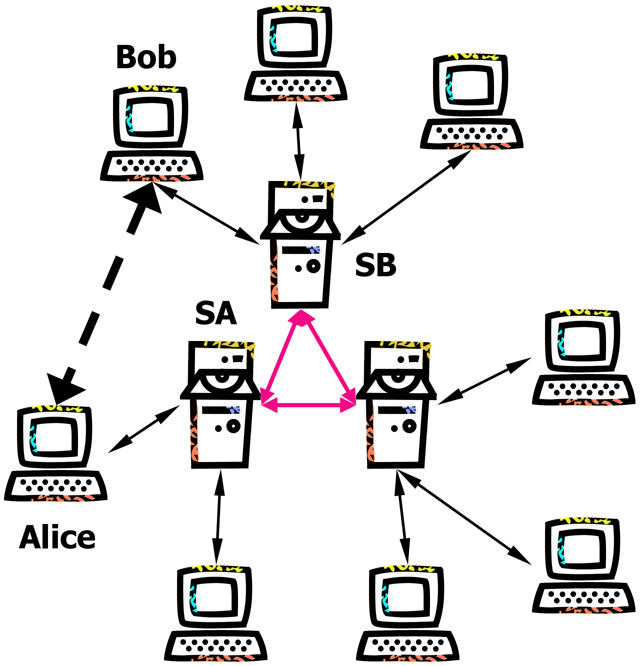
Gnutella: Distributed Flooding

- Bob wants to talk to Alice, he must broadcast request and get information from Jane
- Benefits:
  - No central point of failure
  - Limited per-node state
- Drawbacks:
  - Slow searches
  - Bandwidth intensive
  - Scalability



KaZaA: Hierarchical Searching

- Bob talks to Alice via Server B and Server A.
- Popularity:
  - More than 3 M peers
  - Over 3,000 Terabytes
  - >50% Internet traffic ?
- Benefits:
  - Only super-nodes do searching
  - Parallel downloading
  - Recovery
- Drawbacks:
  - Copyrights



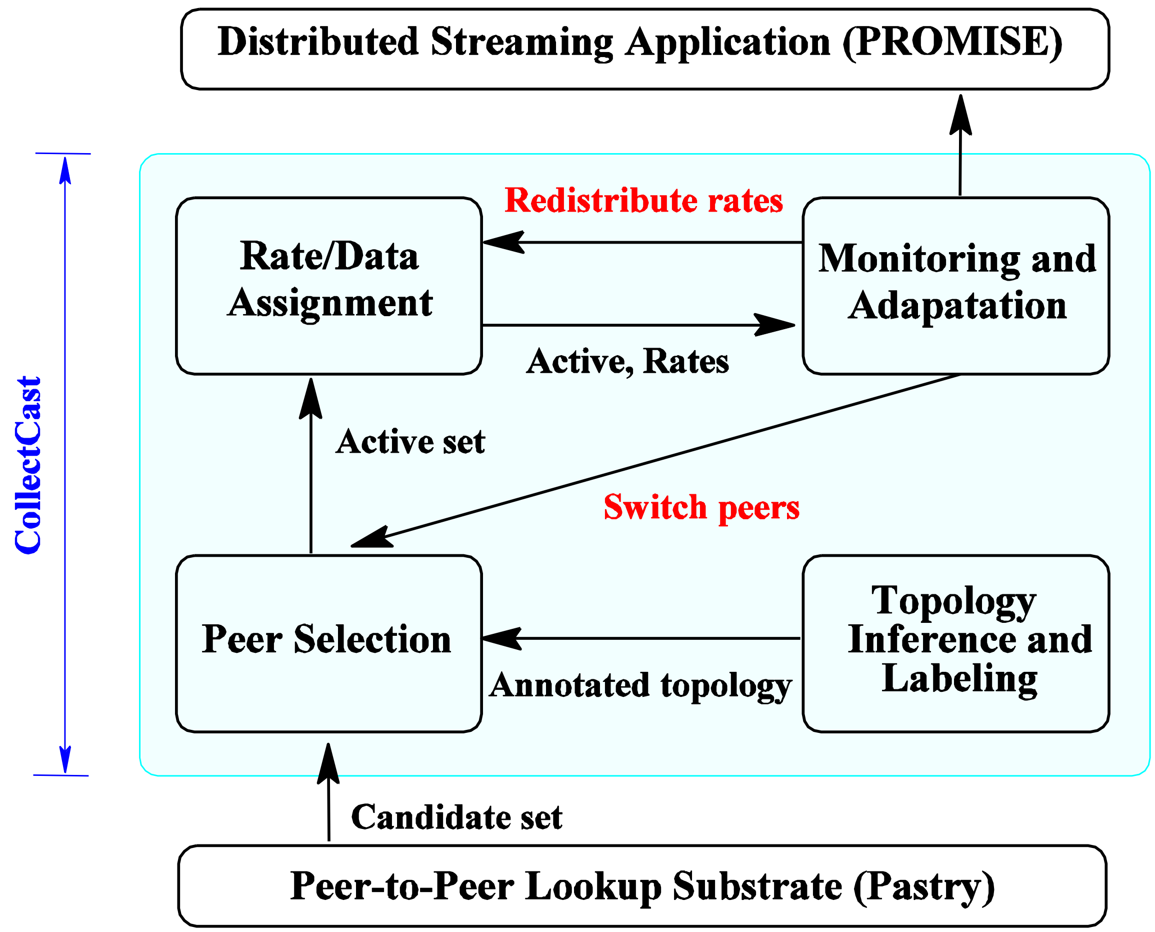
P2P Streaming

- Peers characterized as
  - **Highly diverse**
  - **Dynamic**
  - **Have limited capacity, reliability**
- Problem
  - **How to select and coordinate multiple peers to render the best possible quality streaming?**

CollectCast (Developed at Purdue)

- CollectCast is a new P2P service
  - Middleware layer between a P2P lookup substrate and applications
  - Collects data from multiple senders
- Functions
  - Infer and label topology
  - Select best sending peers for each session
  - Aggregate and coordinate contributions from peers
  - Adapt to peer failures and network conditions

CollectCast (cont'd)



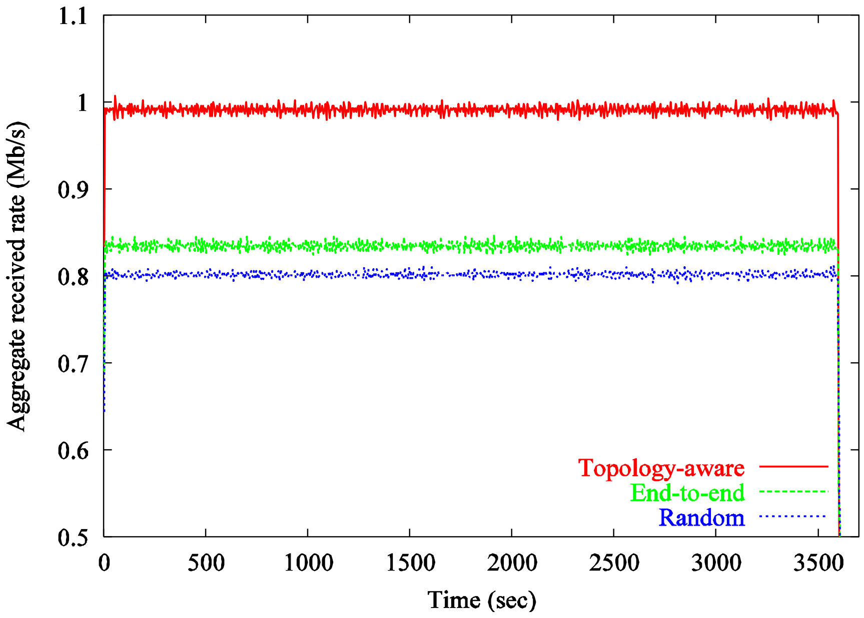
Simulations

- Compare selection techniques in terms of
  - The aggregated received rate, and
  - The aggregated loss rate
  - With and without peer failures
- Impact of peer availability on size of candidate set
- Size of active set
- Load on peers

Simulation: Setup

- Topology
  - On average 600 routers and 1,000 peers
  - Hierarchical (Internet-like)
- Streaming session
  - Rate $R_0 = 1$ Mb/s
  - Duration = 60 minutes
  - Loss tolerance level $\alpha_u = 1.2$
- Peers
  - Offered rate: uniform in $[0.125R_0, 0.5R_0]$
  - Availability: uniform in $[0.1, 0.9]$
  - Diverse P2P community
- Results are averaged over 100 runs with different seeds

Aggregate Rated: No Failures



- Careful selection pays off!

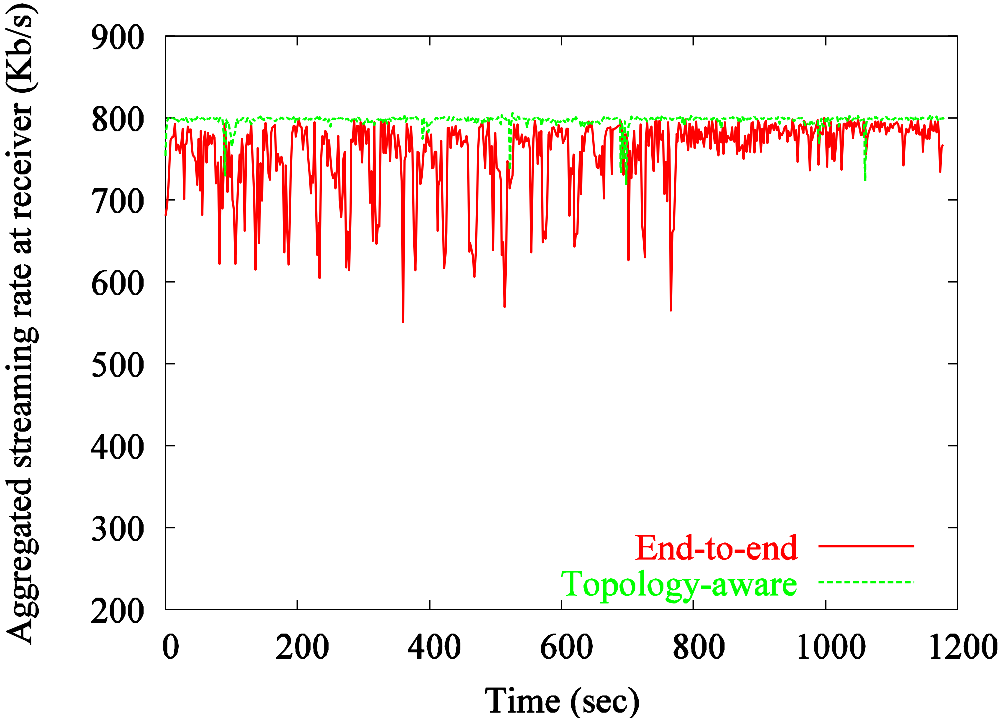
PROMISE and Experiments on PlanetLab (Test-bed at Purdue)

- PROMISE is a P2P media streaming system built on top of CollectCast
- Tested in local and wide area environments
- Extended Pastry to support multiple peer look up

PlanetLab Experiments

- PROMISE is installed on 15 nodes
- Use several MPGE-4 movie traces
- Select peers using topology-aware (the one used in CollectCast) and end-to-end
- Evaluate
  - Packet-level performance
  - Frame-level performance and initial buffering
  - Impact of changing system parameters
  - Peer failure and dynamic switching

Packet-Level: Aggregated Rate



- Smoother aggregated rate achieved by CollectCast

Conclusions

- New service for P2P networks (CollectCast)
  - Infer and leverage network performance information in selecting and coordinating peers
- PROMISE is built on top of CollectCast to demonstrate its merits
- Internet Experiments show proof of concept
  - Streaming from multiple, heterogeneous, failure-prone, peers is indeed feasible
- Extend P2P systems beyond file sharing
- Concrete example of network tomography