Rendezvous Points-Based Scalable Content Discovery with Load Balancing

Jun Gao Peter Steenkiste

Computer Science Department
Carnegie Mellon University

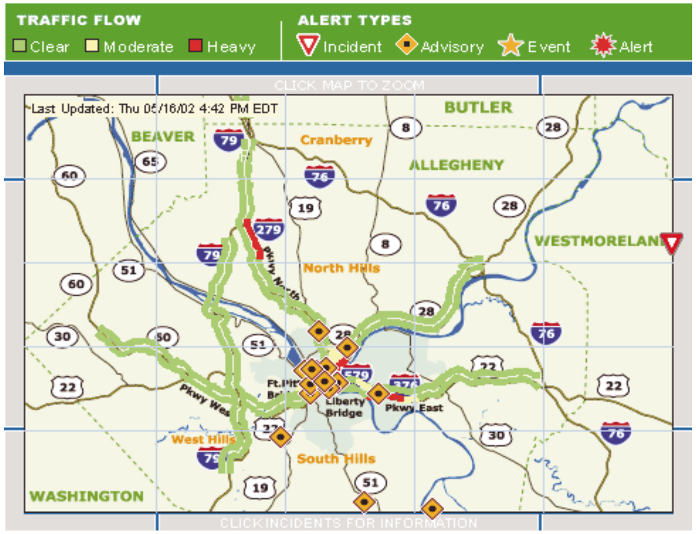
October 24th, 2002
NGC 2002, Boston, USA

Outline

- Content Discovery System (CDS)
- Existing solutions
- CDS system design
- Simulation evaluation
- Conclusions

Content Discovery System (CDS)

- Example: a highway monitoring service
  - Cameras and sensors monitor road and traffic status
  - Users issue flexible queries
- CDS enables content discovery
  - Locate contents that match queries
- Example services
  - Service discovery; P2P; pub/sub; sensor networks



Snapshot from **traffic.com**

Comparison of Existing Solutions

Solutions / Design Goals	Centralized	Distributed			
		Tree-based	Graph-based		
			Registration flooding	Query broadcasting	Hash-based
Searchability	yes	Hierarchical names	yes	yes	Look-up
Robustness	no	no	yes	yes	yes
Scalability	yes?	yes	no	no	yes
Load balancing	yes?	no	no	no	yes?

CDS Design

- Attribute-value pair based naming scheme
  - Enable searchability
- Peer-to-peer system architecture
  - Robust distributed system
- Rendezvous Points-based content discovery
  - Improve scalability
- Load Balancing Matrix
  - Dynamic balance load

Naming Scheme

- Based on Attribute-Value pairs
  - **CN: $\{a_1=v_1, a_2=v_2, \ldots, a_n=v_n\}$**
  - Not necessarily hierarchical
  - Attribute can be dynamic
- Searchable via subset matching
  - $Q \subseteq CN$
  - Number of matches for a CN is large
    - 2^n-1

CN1

Camera ID = 5562 Highway = I-279 Exit = 4 City = Pittsburgh Speed = 25mph Road condition = Icy

Q1

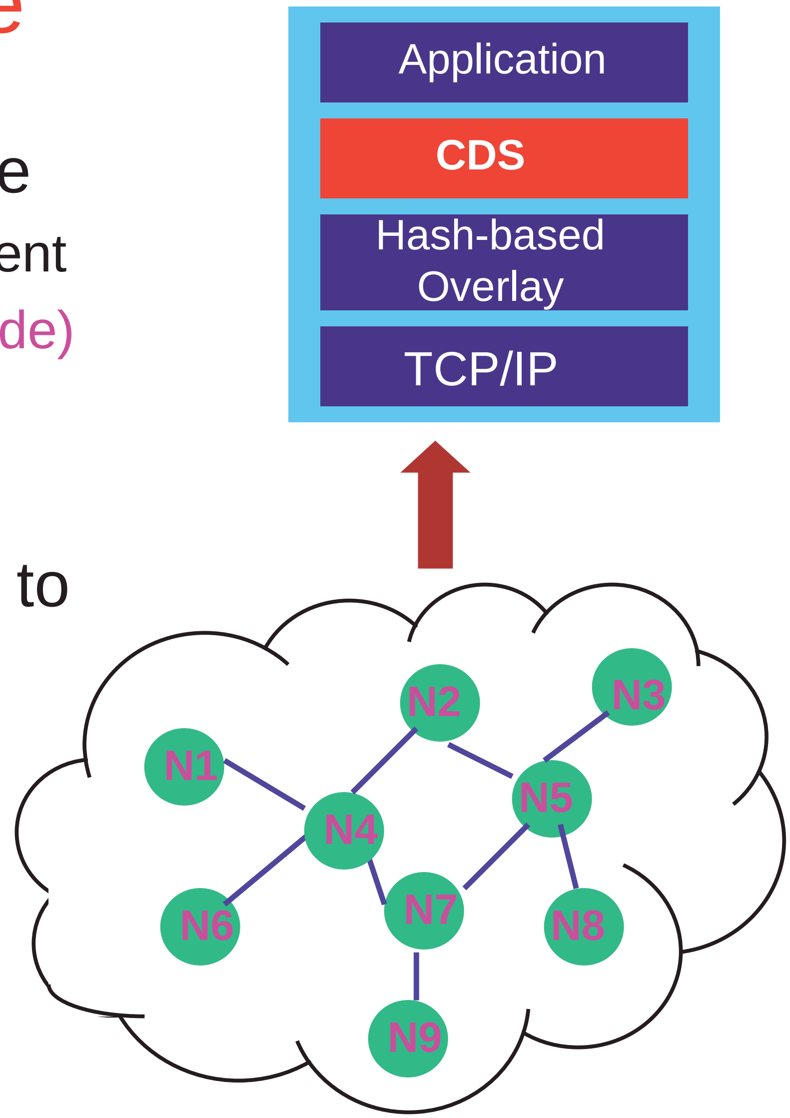
Highway = I-279 Exit = 4 City = Pittsburgh

Q2

City = Pittsburgh Speed = 25mph

Distributed Infrastructure

- Hash-based overlay substrate
  - Routing, forwarding, management
  - Node ID → Hash function *H*(node)
- Application layer publishes contents or issues queries
- CDS layer determines where to register contents and send queries
  - Centralized and network-wide flooding are not scalable
  - Idea: use a small set of nodes as Rendezvous Points



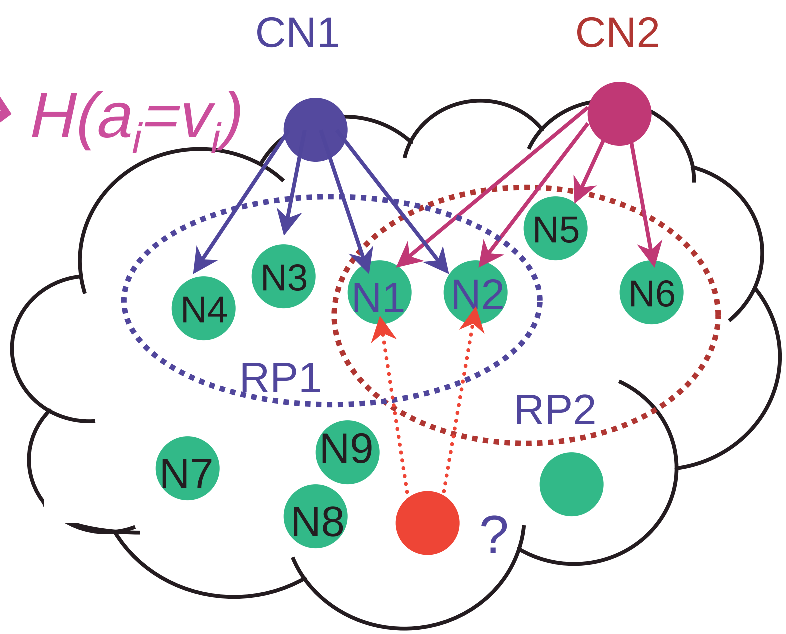
RP-based Scheme

- Hash each AV-pair to get a set of RPs
  - | RP | = n
- RP node stores names that share the same pair
  - Maintain with soft state
- Query is sent directly to an RP node
  - Use the least loaded RP
  - RP node fully resolves locally

CN1: {a1=v1, a2=v2, a3=v3, a4=v4}

CN2: {a1=v1, a2=v2, a5=v5, a6=v6}

$N_i \rightarrow H(a_i=v_i)$



Q:{a1=v1, a2=v2}

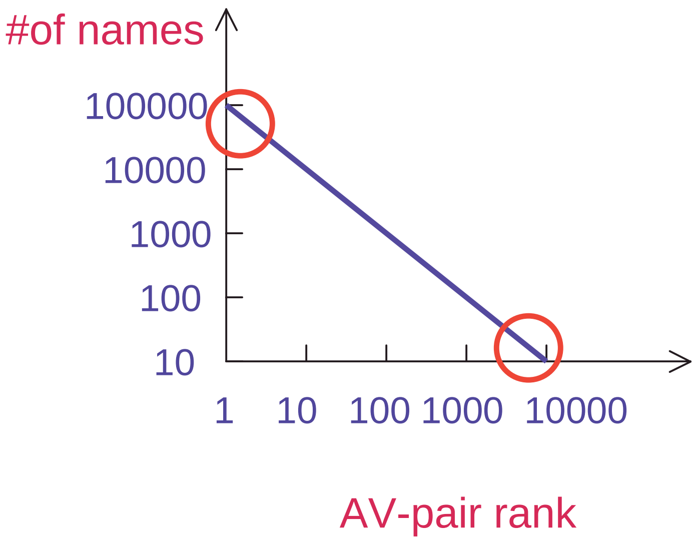
System Properties

- Efficient registration and query
  - O(n) registration messages; n small
  - O(m) messages for query with probing
- Hashing AV-pair individually ensures subset matching
  - Query may contain only 1 AV-pair
- No inter-RP node communication for query resolution
  - Tradeoff between CPU and Bandwidth
- Load is spread across nodes
  - Different names use different RP set

Load Concentration Problem

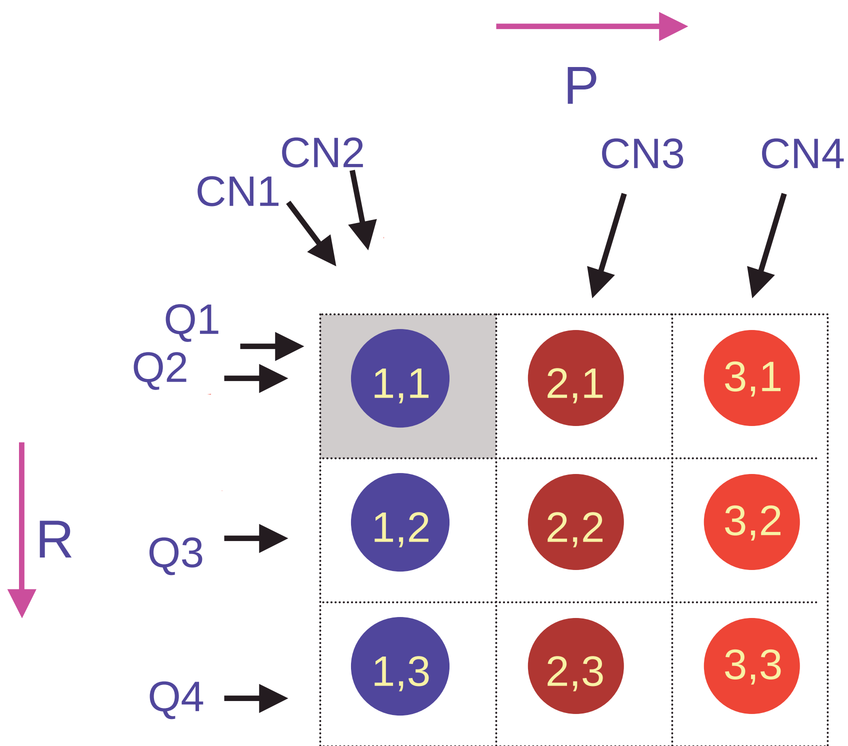
- RP node may be overloaded
  - Some AV-pairs more popular than others
    - Speed=55mph vs. Speed=95mph
    - P2P keyword follows Zipf distribution
  - However, many nodes are underutilized
- Intuition: use a set of nodes to share load caused by popular pairs
- Challenge: accomplish load balancing in a distributed and self-adaptive fashion

Example Zipf distribution



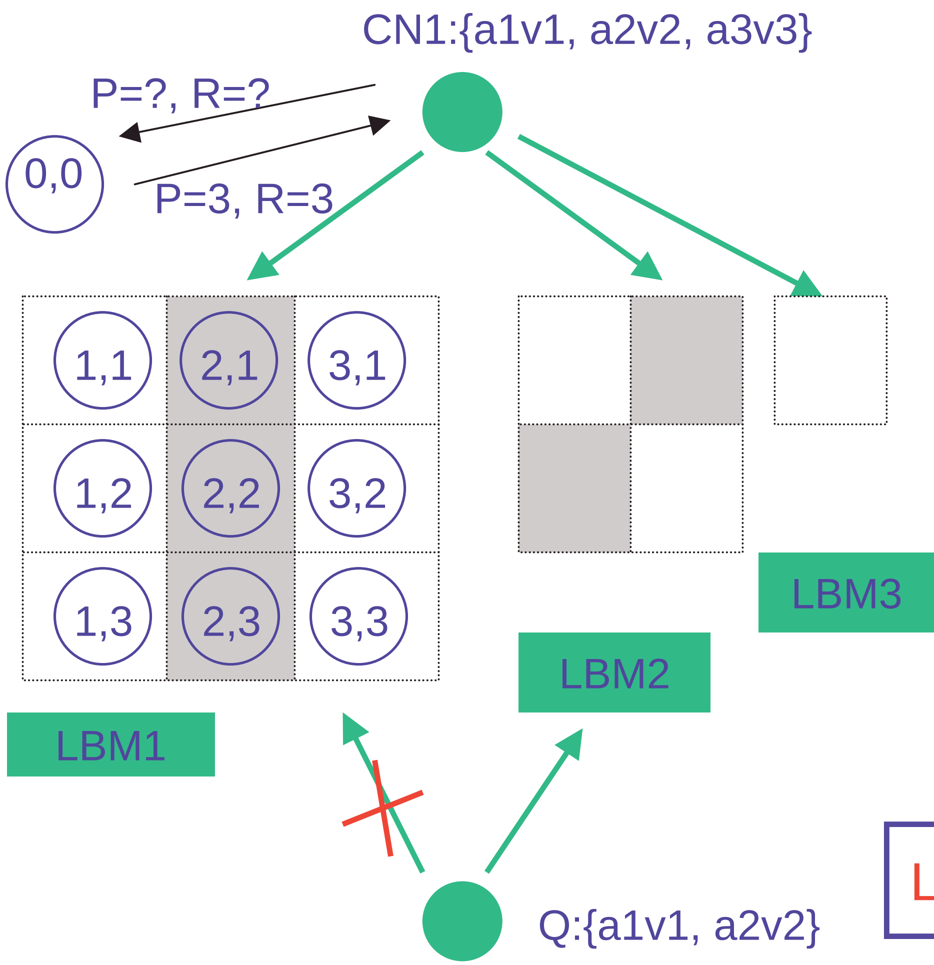
Load Balancing Matrix (LBM)

LBM for AV-pair: {a1=v1}



- Organize nodes into a logical matrix
  - Each column holds a partition
  - Rows are replicas of each other
- Node IDs are determined by: $H(a1=v1, p, r) \rightarrow N_1^{(p,r)}$
- Matrix expands itself to accommodate extra load
  - Increase P when registration load reaches threshold
  - Query load ↑ → R ↑

Registration and Query with LBM



Registration

- Determine LBM size
- Register with one random column of each matrix
  - Compute IDs locally

Query

- Query can be sent to the matrix of any AV-pair
  - Use LBM with min(P)
- Sent to one random node in each column

Load is balanced within an LBM

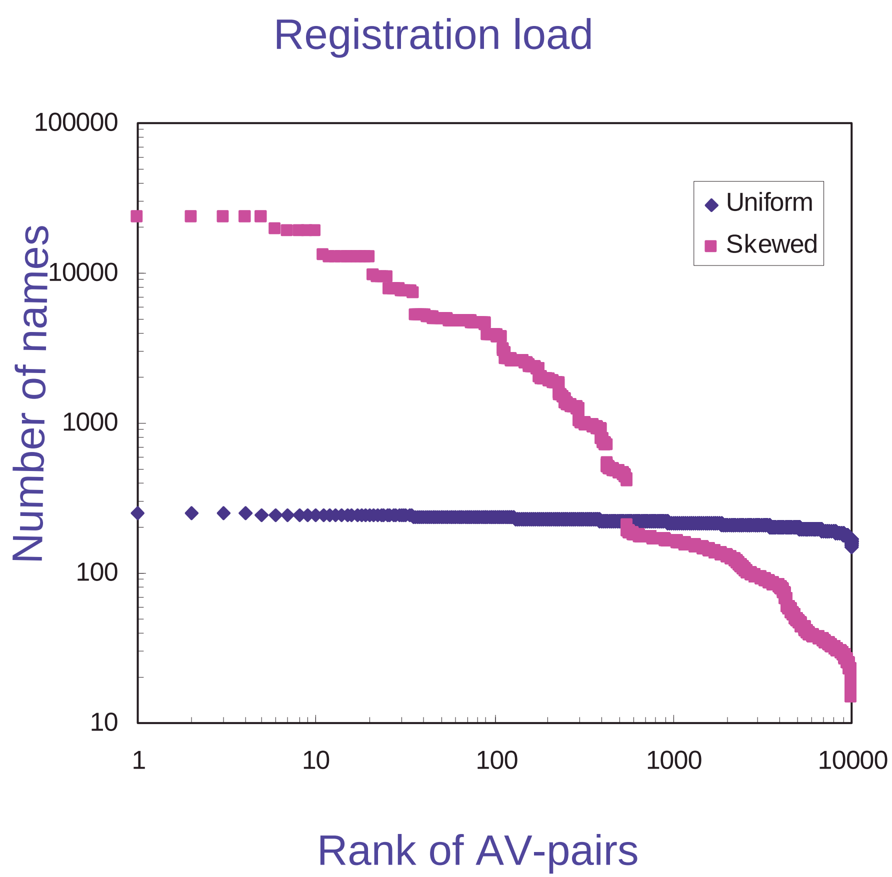
System Properties with LBM

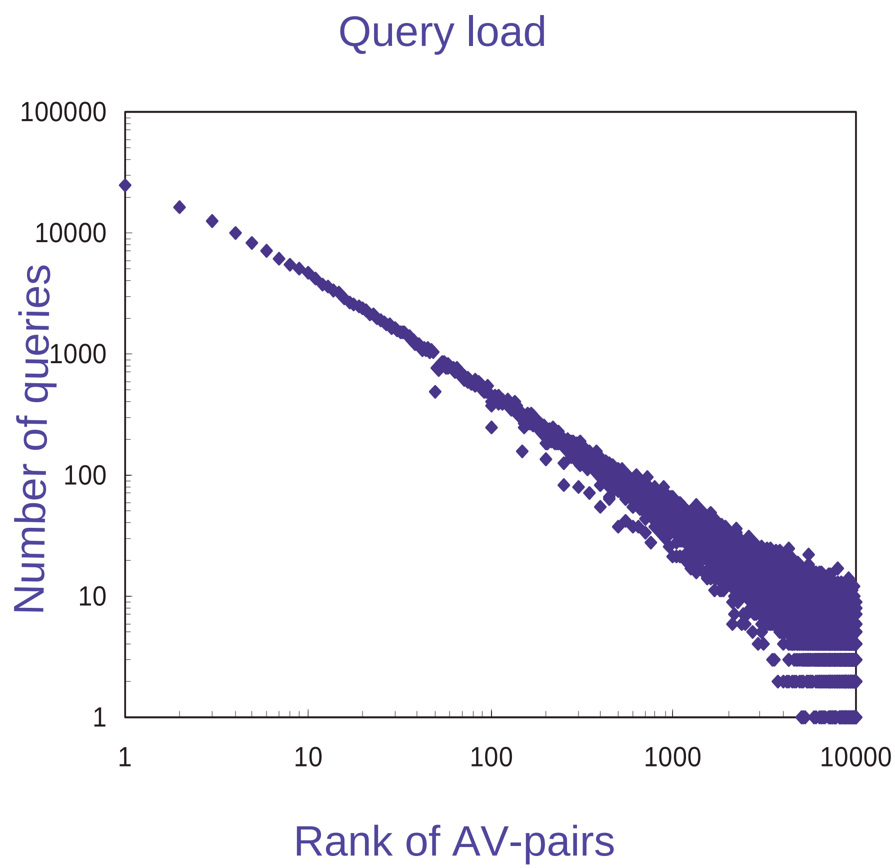
- Registration and query cost for one pair increases
  - O(R) registration messages
  - O(P) query messages
  - Matrix size depends on current load
- LBM must be kept small for efficiency
  - Query optimization helps, e.g., large P → small R
  - Matrix shrinking mechanism
  - E.g., May query a subset of the partitions
- Load on each RP node is upper-bounded
  - Efficient processing
- Underutilized nodes are recruited as LBM expands

Simulation Evaluation

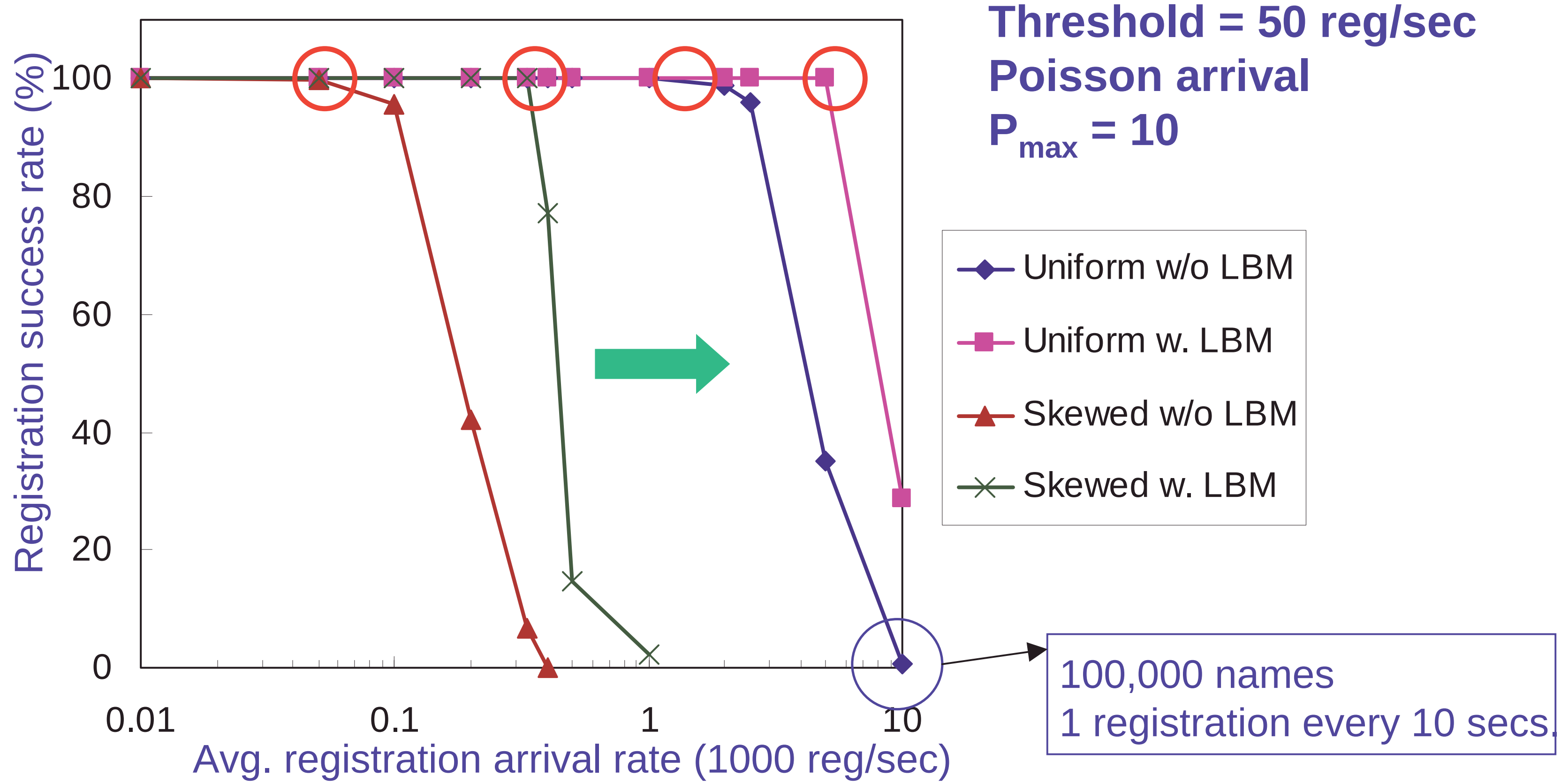
- Implement in an event-driven simulator
  - Each node monitors its registration and query load
  - Assume Chord-like underlying routing mechanism
- Experiment setup
  - 10,000 nodes in the CDS network
  - 10,000 distinct AV-pairs (50 attributes, 200 values/attribute)
  - Use synthetic registration and query workload
- Performance metric: success rate
  - System should maintain high success rate as load increases

Workload


Registration load
100000
10000
1000
100
10
1
10
100
1000
10000
Number of names
Rank of AV-pairs
Uniform
Skewed


Query load
100000
10000
1000
100
10
1
1
10
100
1000
10000
Number of queries
Rank of AV-pairs

Registration Success Rate Comparison



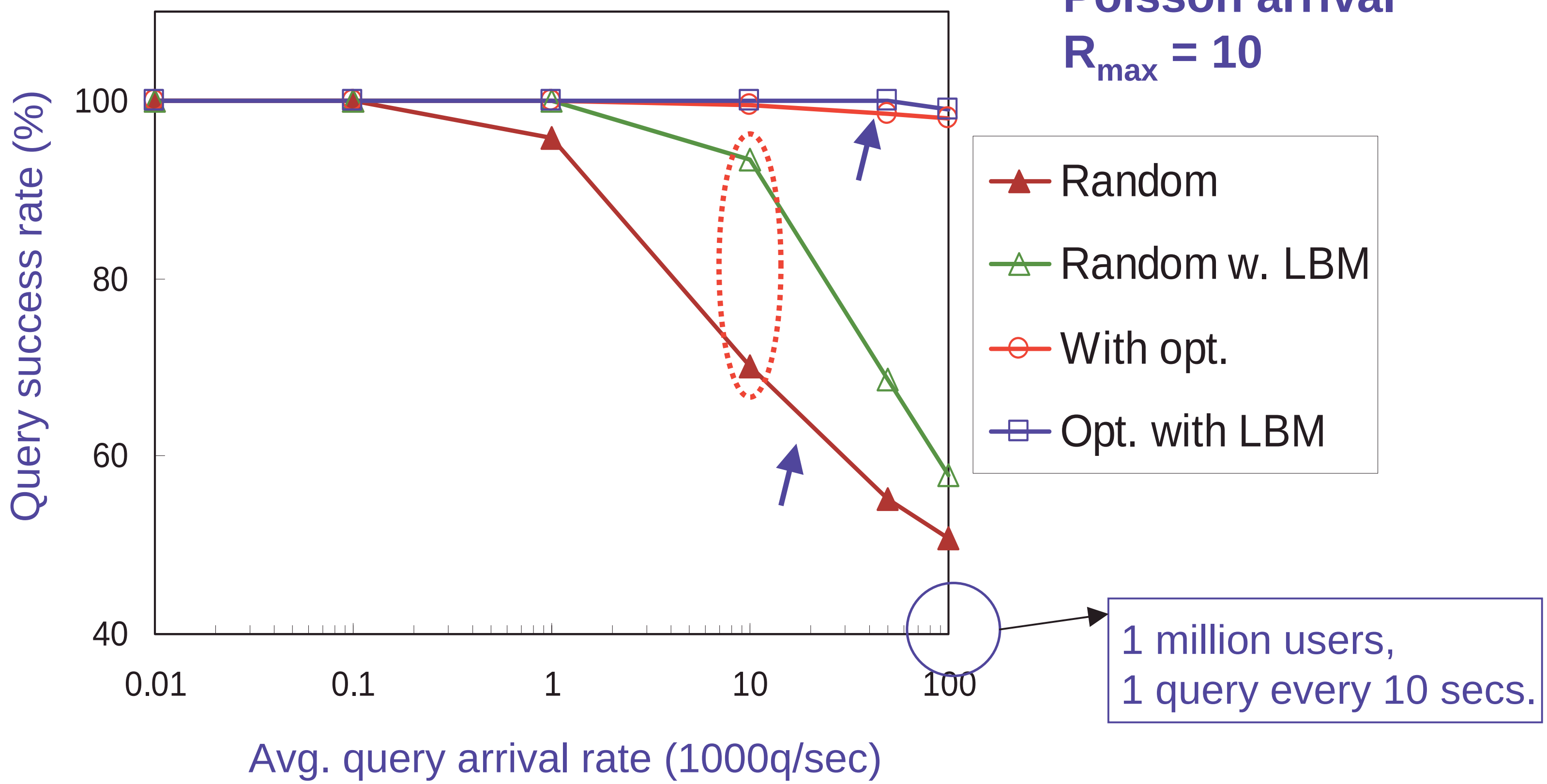
Threshold = 50 reg/sec

Poisson arrival

P_{max} = 10

Query Success Rate Comparison

Threshold = 200 q/sec
Poisson arrival
R_{max} = 10



Conclusions

- Proposed a distributed and scalable design to the content discovery problem
- RP-based approach addresses scalability
  - Avoid flooding
- LBMs improve system throughput
  - Balance load
- Distributed algorithms
  - Decisions are made locally