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

<table>
<thead>
<tr>
<th>Design Goals</th>
<th>Solutions</th>
<th>Centralized</th>
<th>Distributed</th>
<th>Graph-based</th>
<th>Hash-based</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tree-based</td>
<td>Registration flooding</td>
<td>Query broadcasting</td>
</tr>
<tr>
<td>Searchability</td>
<td>yes</td>
<td>Hierarchical names</td>
<td>yes</td>
<td>yes</td>
<td>Look-up</td>
</tr>
<tr>
<td>Robustness</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Scalability</td>
<td>yes?</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Load balancing</td>
<td>yes?</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes?</td>
</tr>
</tbody>
</table>
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
    \(2^n - 1\)

---

<table>
<thead>
<tr>
<th>CN1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera ID = 5562</td>
</tr>
<tr>
<td>Highway = I-279</td>
</tr>
<tr>
<td>Exit = 4</td>
</tr>
<tr>
<td>City = Pittsburgh</td>
</tr>
<tr>
<td>Speed = 25mph</td>
</tr>
<tr>
<td>Road condition = Icy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highway = I-279</td>
</tr>
<tr>
<td>Exit = 4</td>
</tr>
<tr>
<td>City = Pittsburgh</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q2</th>
</tr>
</thead>
<tbody>
<tr>
<td>City = Pittsburgh</td>
</tr>
<tr>
<td>Speed = 25mph</td>
</tr>
</tbody>
</table>
Distributed Infrastructure

- Hash-based overlay substrate
  - Routing, forwarding, management
  - Node ID $\rightarrow$ Hash function $H(\text{node})$

- 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\}

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
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 \( \uparrow \rightarrow R \uparrow \)
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 $P \rightarrow$ 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**

**Query load**

- **Number of names**
- **Number of queries**

- **Rank of AV-pairs**

- Uniform
- Skewed

---

Jun Gao
Carnegie Mellon University
Registration Success Rate Comparison

Threshold = 50 reg/sec
Poisson arrival
$P_{max} = 10$

100,000 names
1 registration every 10 secs.
Query Success Rate Comparison

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

1 million users,
1 query every 10 secs.
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