HWP2 – Distributed search

HWP1 – Each beacon knows about every other beacon
HWP2 – Distributed search

• `searchget(token, searchkey, hopcount)`
  – Restrict each beacon to maintain information about two other beacons
HWP2 – Distributed search

- searchget(token, searchkey, hopcount)
  - Restrict each beacon to maintain information about two other beacons

Scenario: 2

B1 → B2 → B3
B4 → B2 → B6
B5 → B2 → B3
**HWP2 – Distributed search**

- `searchget(token, searchkey, hopcount)`
  - Restrict each beacon to maintain information about two other beacons

**Scenario: 1**

![Diagram](image)

- SRC
- Hop count = 3
- B1 → B2 → B3 → B6 → DST
- B2
- B3
- B4
- B5
- B6
- DST
HWP2 – Distributed search

- `searchget(token, searchkey, hopcount)`
  - Restrict each beacon to maintain information about two other beacons

Scenario: 1

- SRC
- Hop count=3
- B1
  - HC=2
- B2
  - HC=2
- B3
  - HC=1
- B4
  - HC=2
- B5
- B6
  - HC=0
- DST
HWP2 – Distributed search

- searchget(token, searchkey, hopcount)
  - Restrict each beacon to maintain information about two other beacons

Scenario: 2

SRC
Hop count=3
• You will use controlled flooding to search for key
Reference HWP1 solution

• C source code will be available in the course webpage (home works section)
• Three threads
  – locatePeersSend
    • Continuously sends identification every BEAT seconds on multicast port
    • Garbage collects clients that you haven’t heard in 3*BEAT seconds
  – locatePeersRecv
    • Receives multicast packets and adds to internal table
  – serviceRequest
    • Services telnet clients for tuple service
Course project proposal?
Outline

Naming – brief intro.

- DNS translates from machine names to IP addresses
  - greenhouse.cs.uga.edu ⇒ 128.192.152.206
  - static translation (usually valid for a few days)

- DNS round-robin used to dynamically match name to host (based on machine load, for example)
  - www.cnn.com ⇒ 207.25.71.23 or 207.25.71.24 or ...

- Network level switch (for e.g. CISCO director)
  - Automatically forward network packets to some server
Naming Intent

Naming Intent

• The URL is a hint to get the UGA news from some server that is “appropriate” (in terms of locality and access costs) and that fits in my display.

• The exact host names and paths can be remapped to the appropriate forms.

• My view of UGA Today depends on my device, my advertisement preferences etc. Your view of UGA Today may be different from mine.
Active Names Goals

• Server selection – select appropriate server from replicas spread across the Internet

• Client customization – Customize the page for the present client (e.g. Transcoding, customizable portals [e.g. my.yahoo.com])

• Server customization – Advertisements customized for the user, collecting statistics (e.g. hit counts, ad rotation etc.)
Naming today

1. Name

2. host

3. URL - redirect

4. URL

5. Name

Client

DNS Server

Proxy

HTTP Server1

HTTP Server2

Database Server

RESULTS
Active name system

- Clients generate active names (domain:name) and name of a namespace program to resolve it.
- Clients hand them to a resolver.

- Name space program locates next program to run and then transports data to that program.

- Each program acts as a filter that transports and transforms its input to its output.
Active name system

• Active name resolver determines domain-specific program
  – These programs are location independent and can run anywhere
  – Application specific, the name is resolved in domain-specific manner

• Domain specific code (e.g. ad rotation)

• After methods are associated with each active name
  – After methods are a list of programs guaranteed to be called
  – They can perform client-specific transformation of data
Multi-way RPC for efficiency

Traditional RPC
• Results passed down the nodes.
• Adds latency

Client → Proxy → Proxy → Proxy → Server
Multi-way RPC for efficiency

Multi-way RPC
• Results sent directly
• Minimal latency
Multi-way RPC

- Security implications – someone else responds to your queries
- Use capability certificates to authenticate response
- Resource consumption limit should be managed. Resolvers use compute resources on foreign hosts.
- Hierarchical namespaces.
Applications

• Extensibility
  – Compared DNS round robin (next server), distributed director (closest server as measured by hop count) and Active names (number of hops; biased by a decaying histogram of previous performance)
  – Average latency follows Distributed Director at low load and DNS round robin at high load
Transcoding

- Using transcoding

- Grayscale 85KB
- Original 116 KB
- Low JPEG Quality 10 KB
- Crop 40KB

Foggy road with fall foliage canopy

Transliterate few bytes

Thumbnail 2KB
Decision on where to transcode

• Can transcode either on the server or proxy

• Proxy is closer to client

• Wide area network from server to proxy could be congested. Tradeoff between sending a smaller image across the country vs performing transcoding at the proxy

• Active name migrates based on the current load
Composability

• Use server-side include to update page based on the current request
• Banner ad rotation
• Logs cookies

• Implemented using server side and active namespaces.
• Active namespaces are shown to be composable