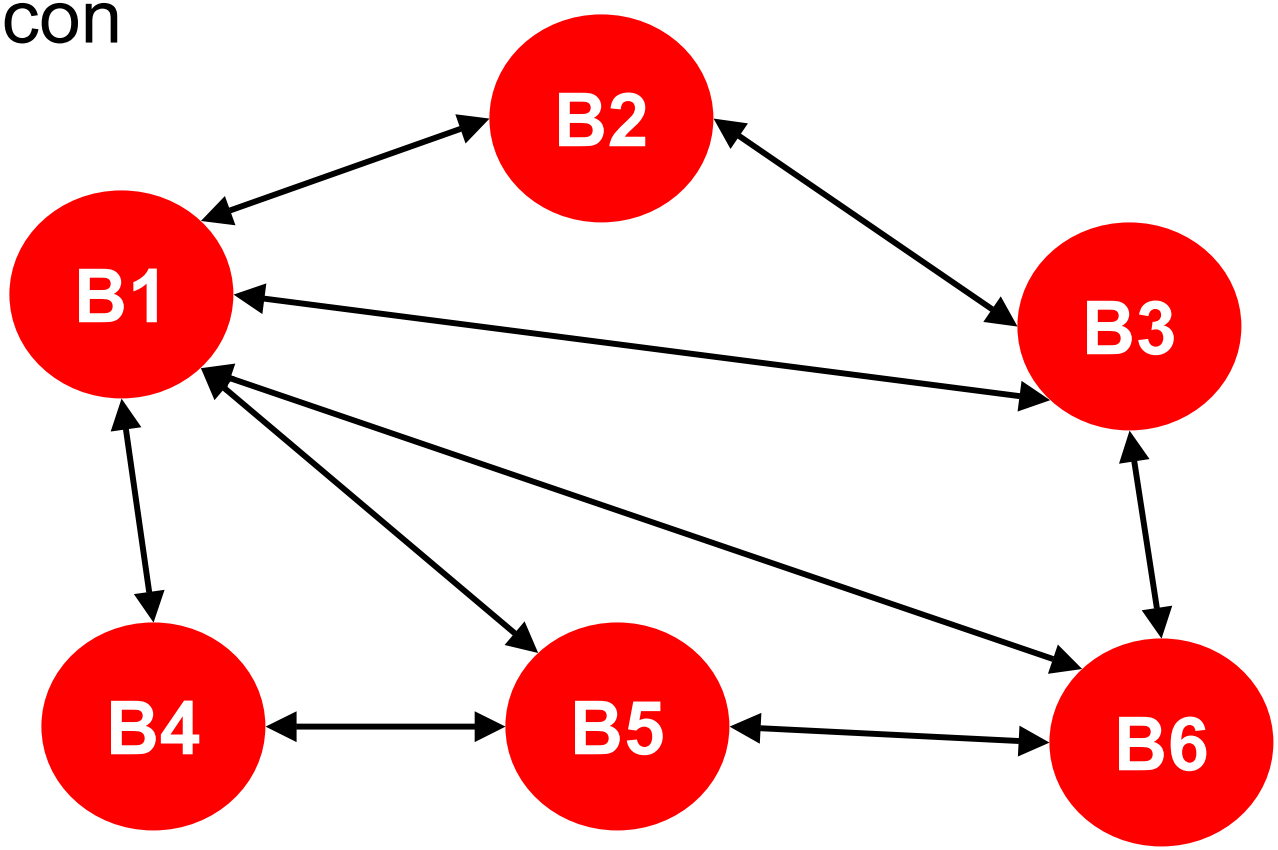
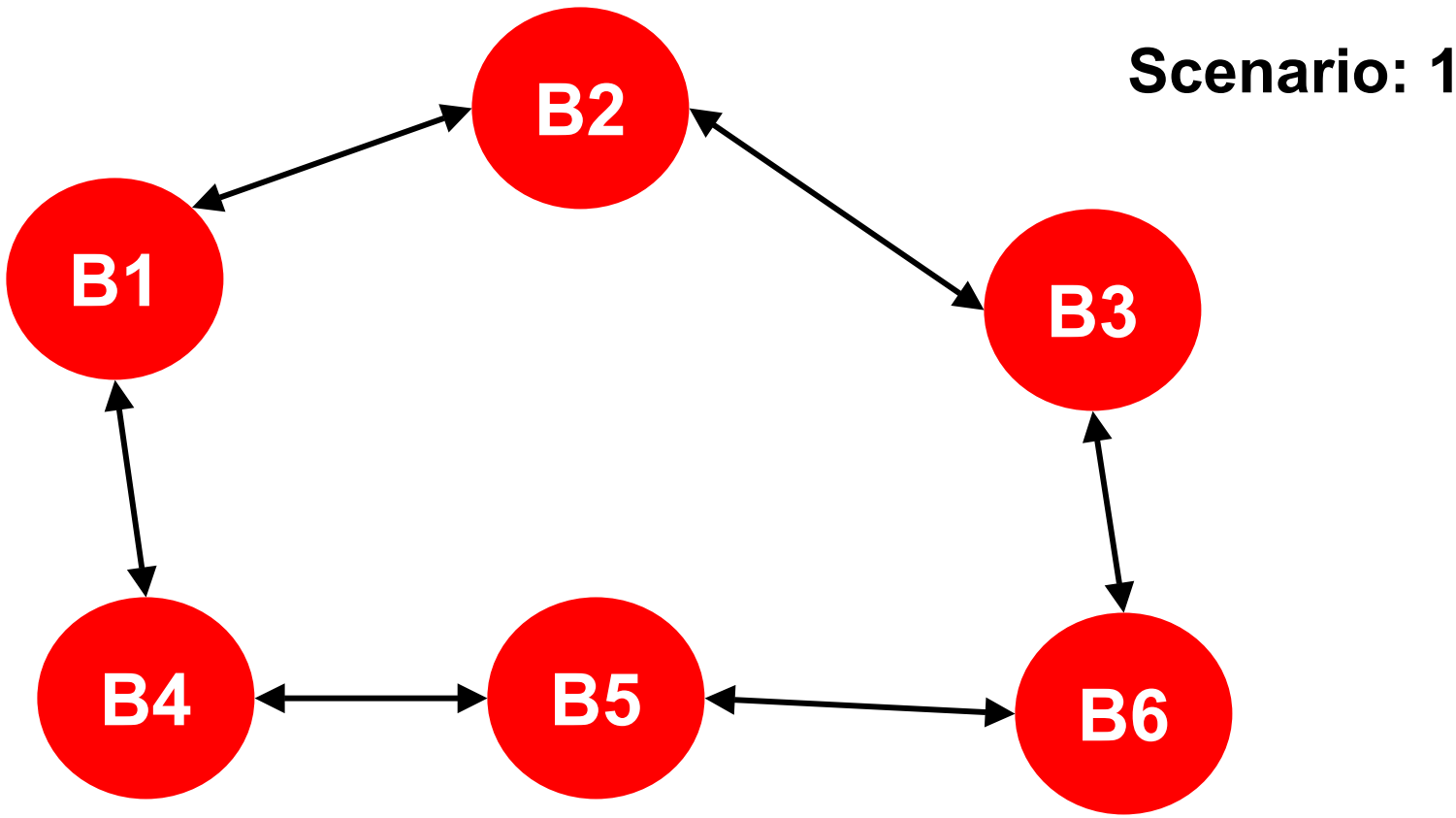
HWP2 – Distributed search

HWP1 – Each beacon knows about every other beacon



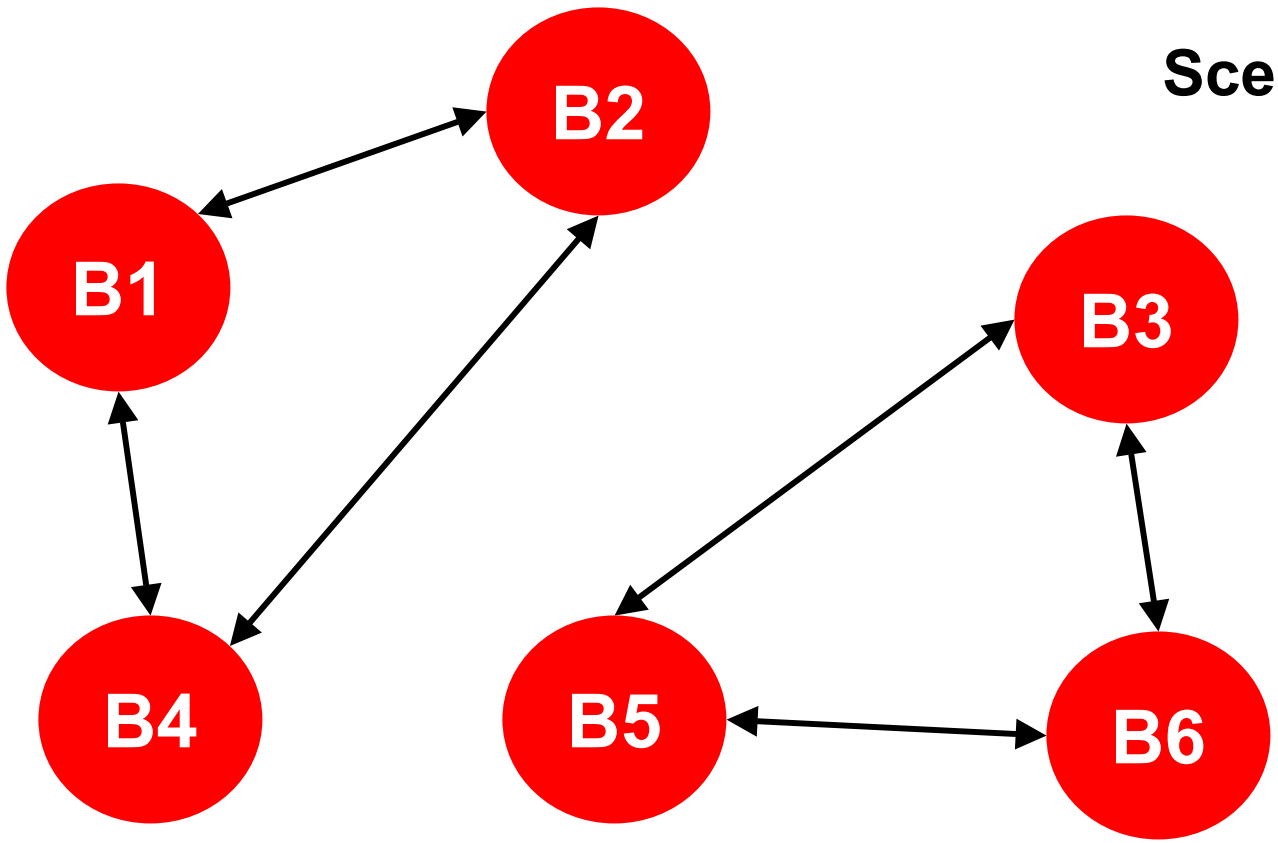
HWP2 – Distributed search

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



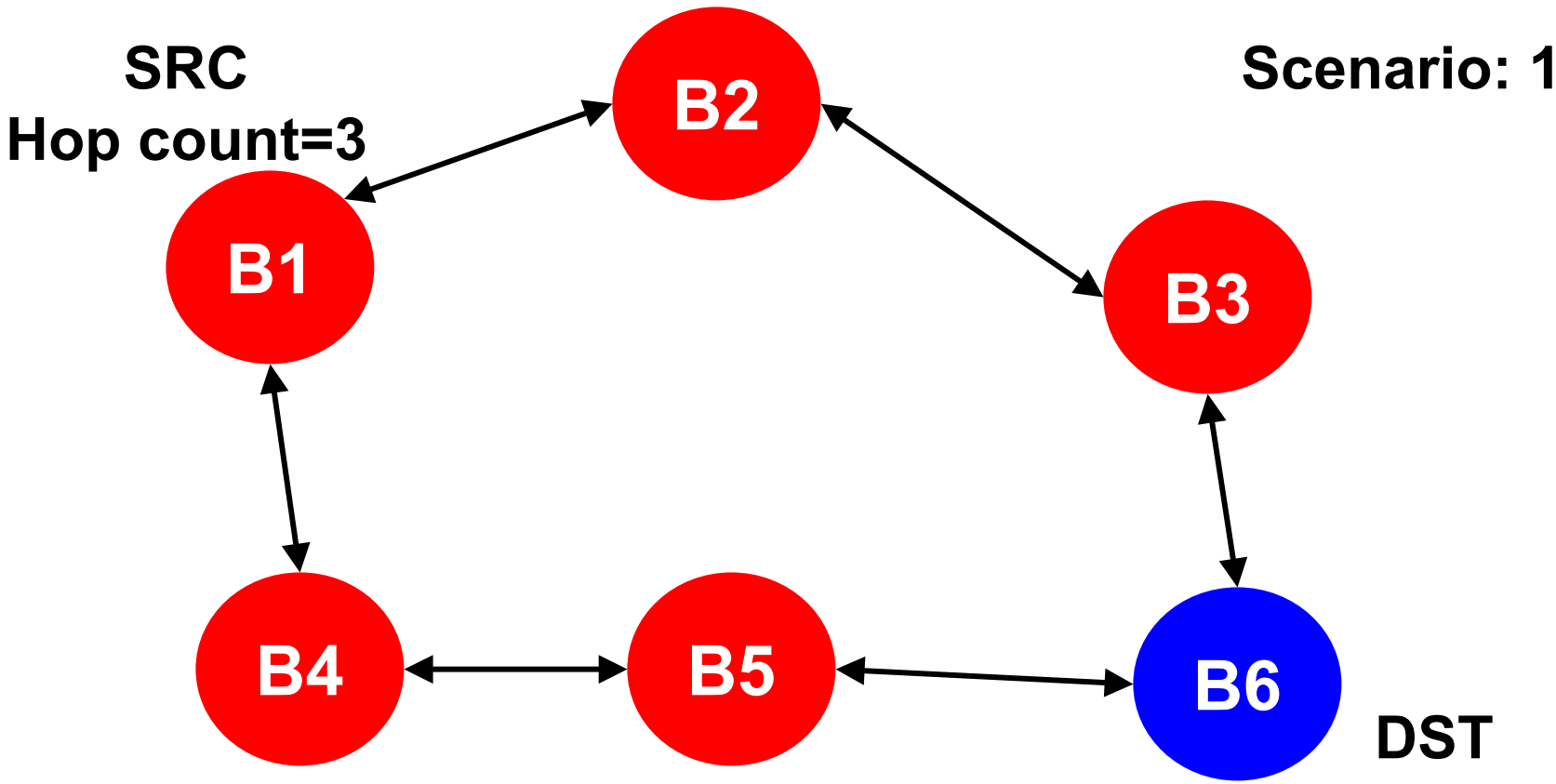
HWP2 – Distributed search

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



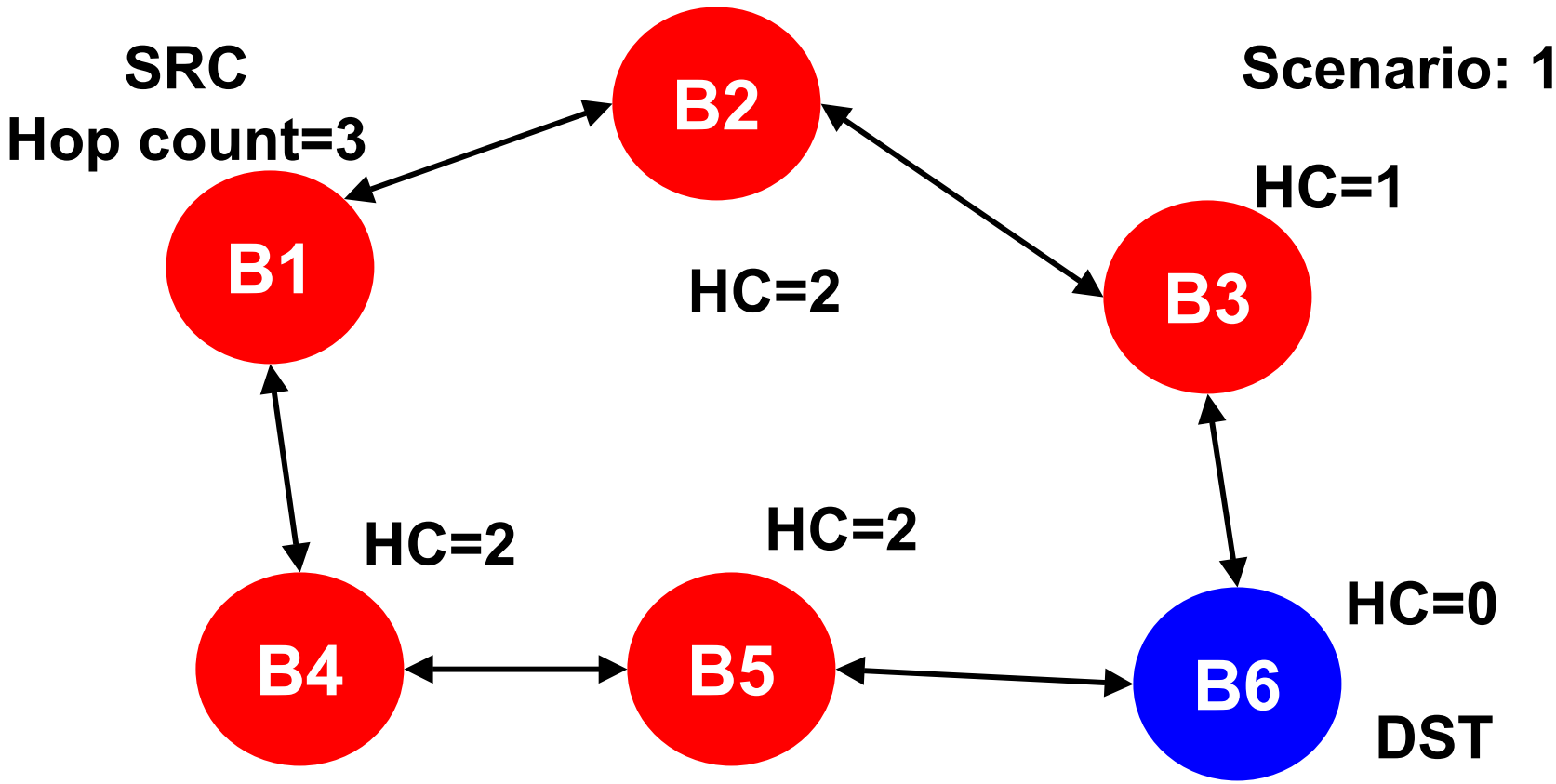
HWP2 – Distributed search

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



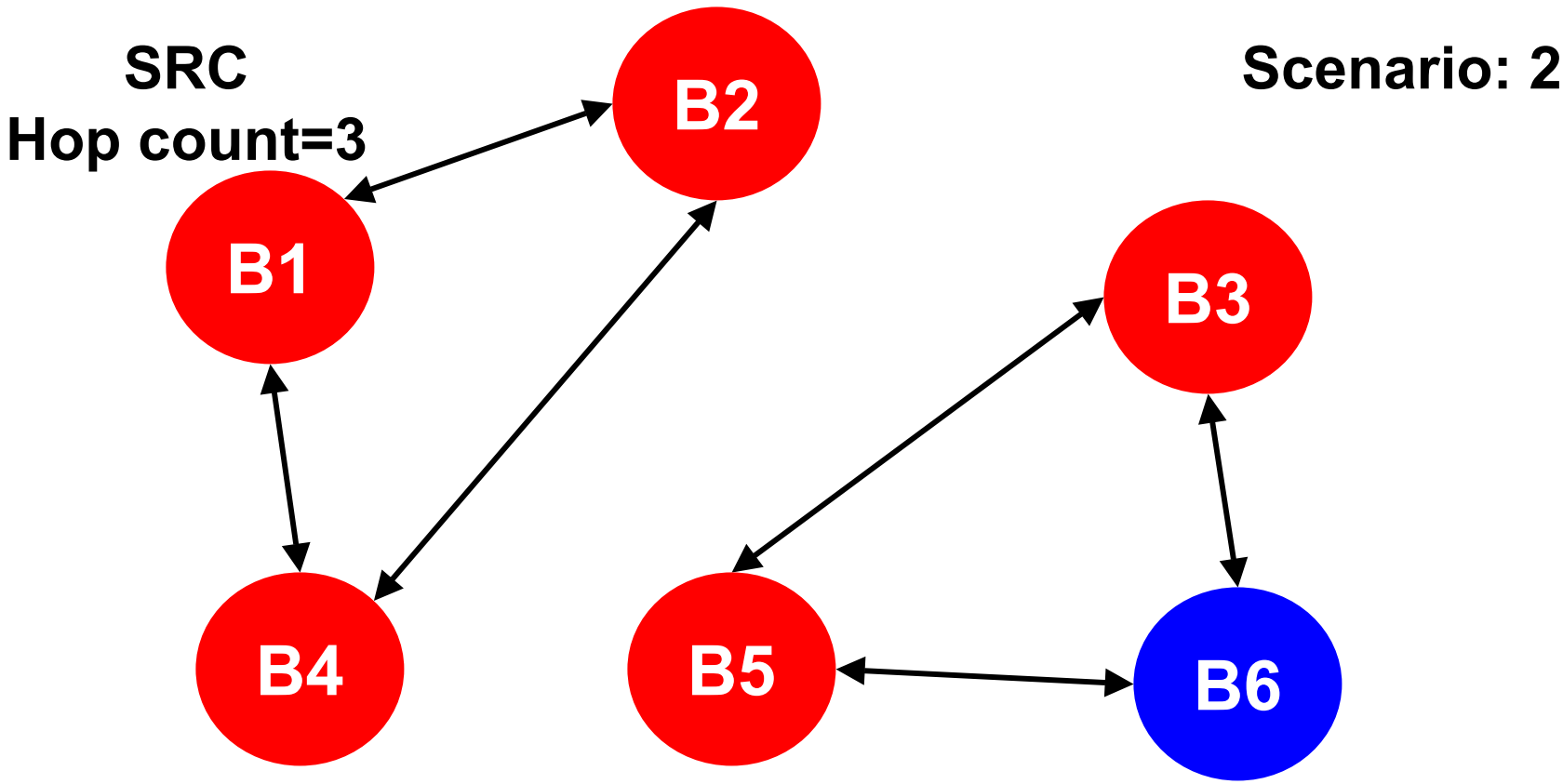
HWP2 – Distributed search

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



HWP2 – Distributed search

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



- You will use controlled flooding to search for key

Reference HWP1 solution

- C source code will be available in the course web page (home works section)
- Three threads
  - locatePeersSend
    - Continously 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

Active Names: Flexible Location and Transport of Wide-Area Resources. Amin Vahdat, Michael Dahlin, Thomas Anderson, and Amit Aggarwal. In Proceedings of the Second USENIX Symposium on Internet Technologies and Systems, October 1999

Naming – brief intro.

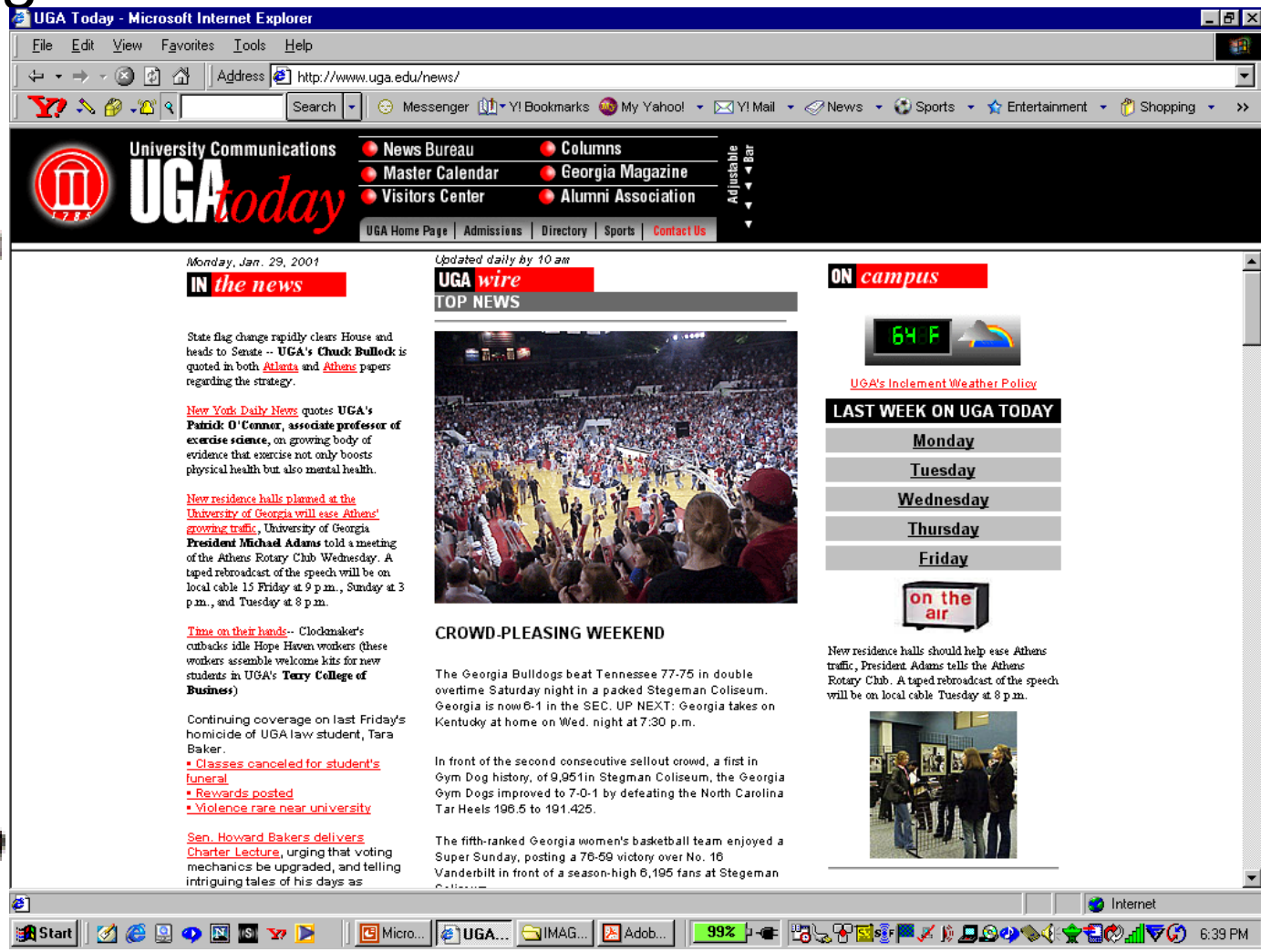
- DNS translates from machine names to IP addresses
  - greenhouse.cs.uga.edu ⇒ 128.192.152.206
  - static translation (usually valid for a few days)

- 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

- http://www.uga.edu/news/



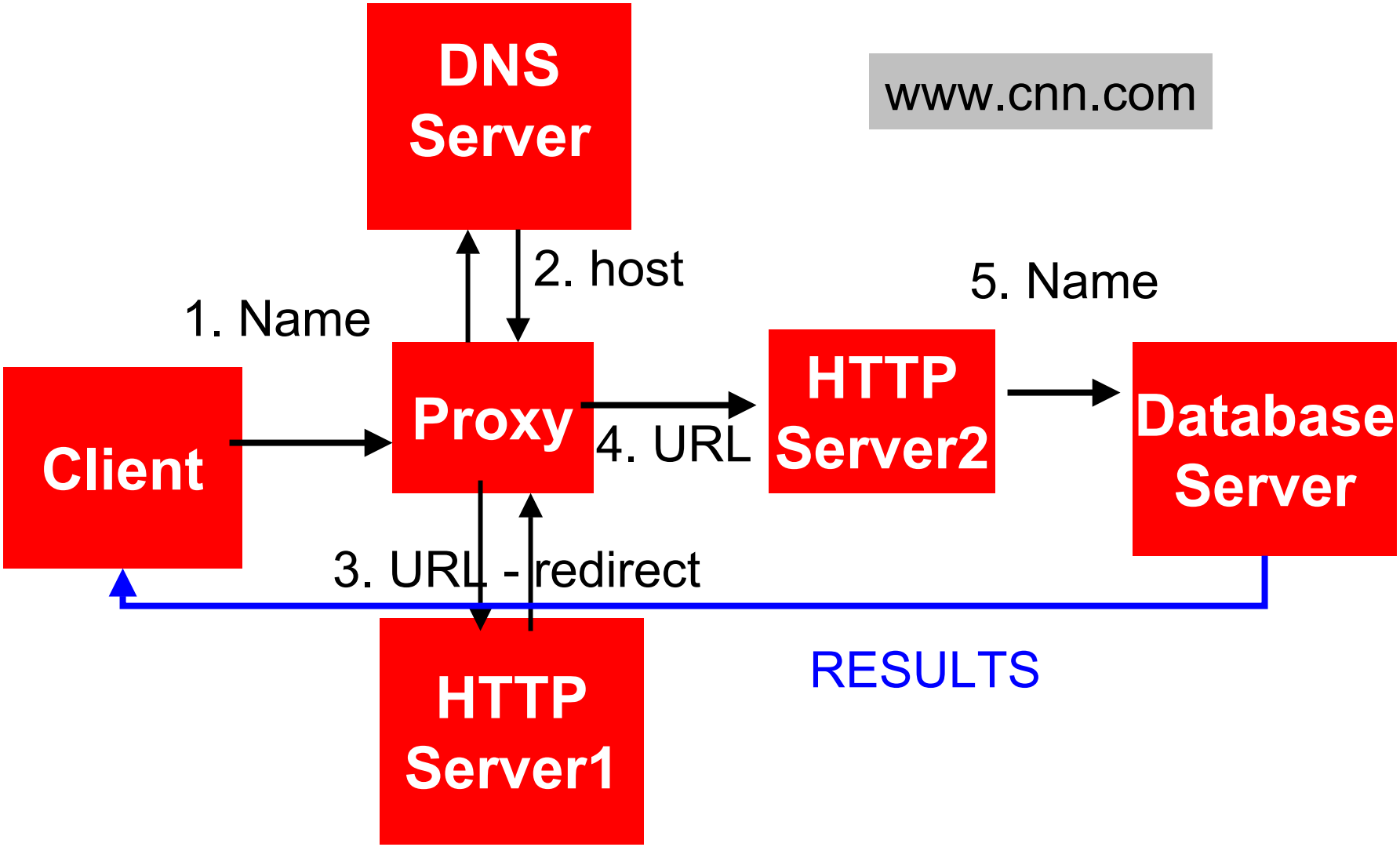
Naming Intent

- 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



Active name system

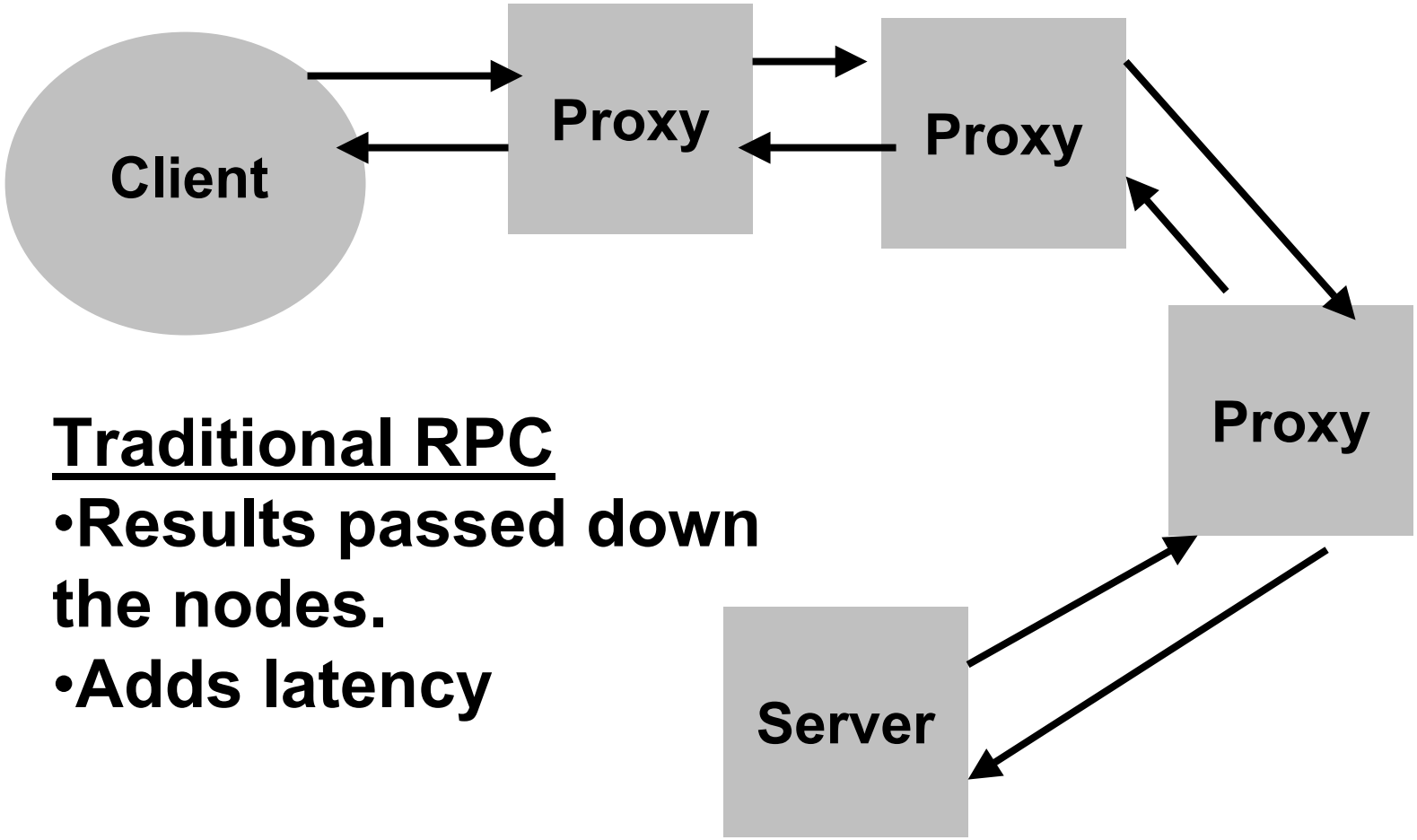
- Clients generate active names (domain:name) and name of a namespace program to resolve it.
- Clients hand them to a resolver.
- Name space program locates next program to run and then transports data to that program
- Each program acts as a filter that transports and transforms its input to its output.

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

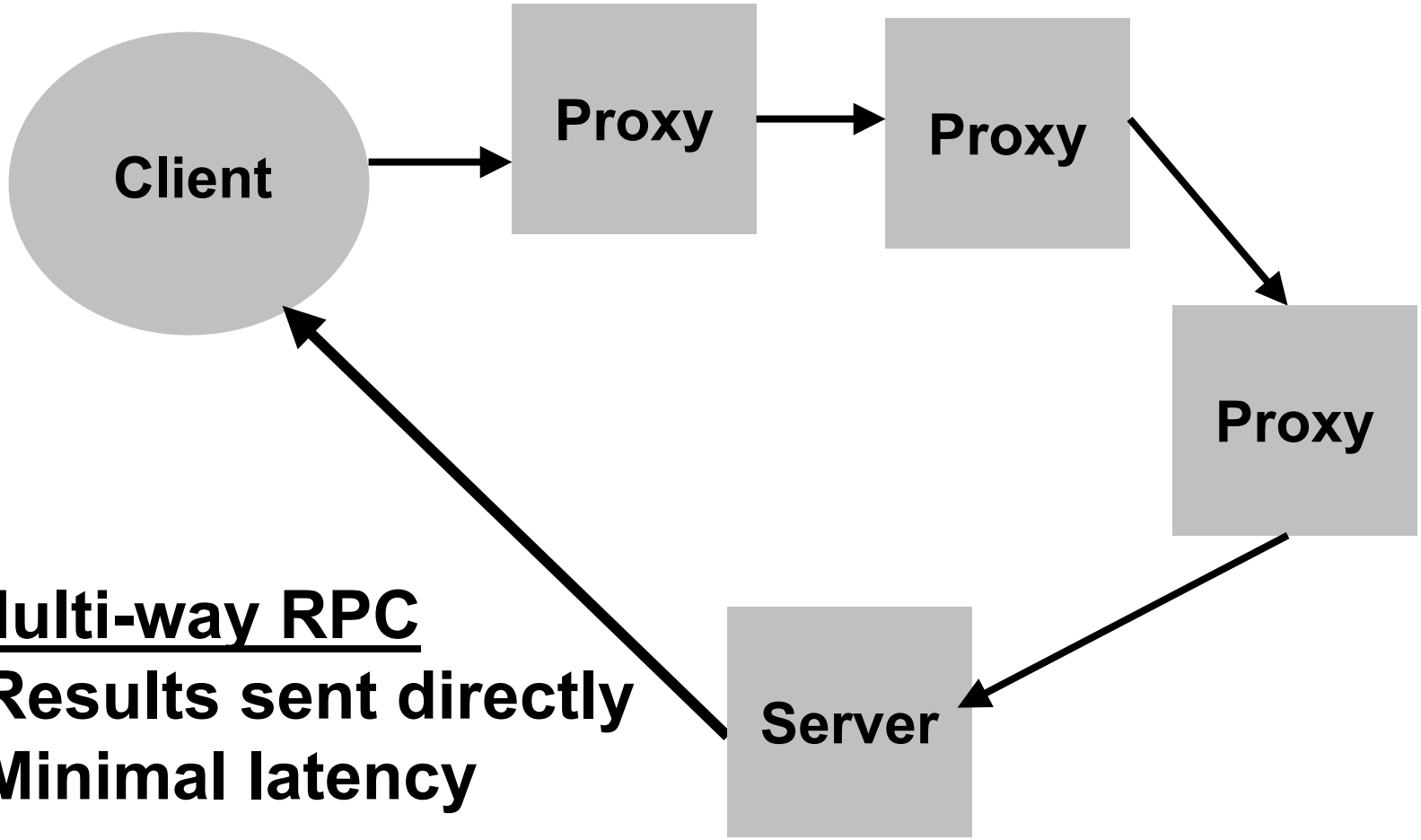
Multi-way RPC for efficiency



<u>Traditional RPC</u>

- **Results passed down the nodes.**
- **Adds latency**

Multi-way RPC for efficiency



<u>Multi-way RPC</u>

- **Results sent directly**
- **Minimal latency**

Multi-way RPC

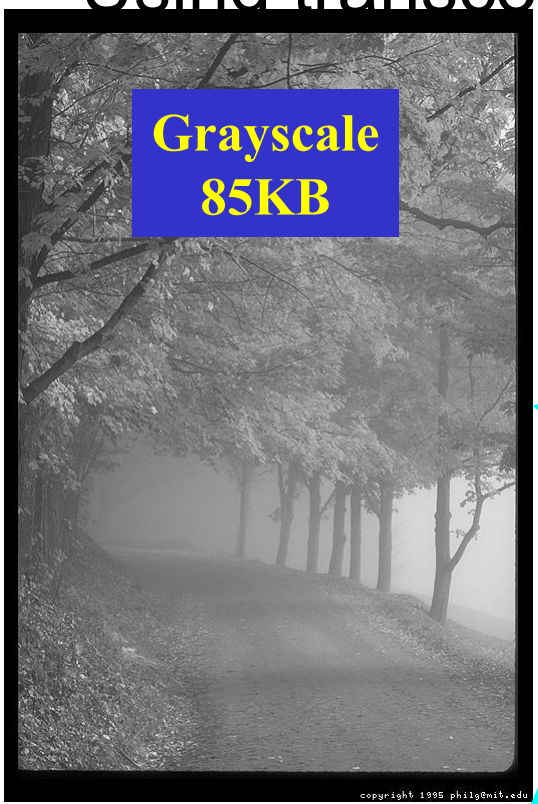
- Security implications – someone else responds to your queries
- Use capability certificates to authenticate response
- Resource consumption limit should be managed. Resolvers use compute resources on foreign hosts.
- Hierarchical namespaces.

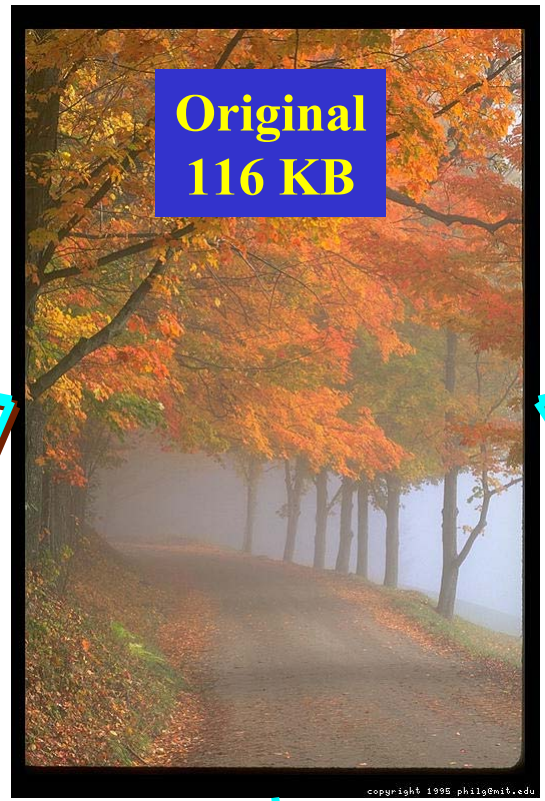
Applications

- 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





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