Caching and Replication on the Internet

Michael Rabinovich

AT&T Labs
misha@research.att.com

Internet Growth

- Traffic doubles every six months [Probe'98]
  - Every 100 days (UUNET)
- Infrastructure can barely keep up
  - Real-time applications will reduce bandwidth for surfing (IP telephony)
- New jump in demand is near
  - Cable modems
  - ISDN
  - DSL
  - New applications (voice, video)
- "The capacity crunch is real and will continue for quite some time"

  Mike O'Dell (Chief Scientist, UUNET)

Addressing Scalability Problem

- "With 25 years of Internet experience, we've learned exactly <u>one</u> way to deal with exponential growth: <u>Caching</u>"

  Van Jacobson

- … and Replication
- $1B market for Web proxy caches by 2000 [Collaborative Research - 98]
- 23 proxy cache vendors
  - http://ircache.nlanr.net/Cache/FAQ/ircache-faq-9.html

Outline

- Background
- Part 1 - Caching
  - Cache levels
  - Issues in proxy caching
  - Cooperative proxy caching
  - Cache as part of network infrastructure
  - Cache consistency
  - Further issues
- Part 2 - Replication
  - Why caching is not enough
  - Fundamental issues
  - Request indirection on the Internet
  - Further issues
- Summary and Conclusion

HTML

- Addressed by a URL
  - < host name, resource name, [parameters] >
- May contain “links” - URLs to other resources
- May embed other resources
- May be stored statically on the server or generated upon request arrival (dynamic resources)

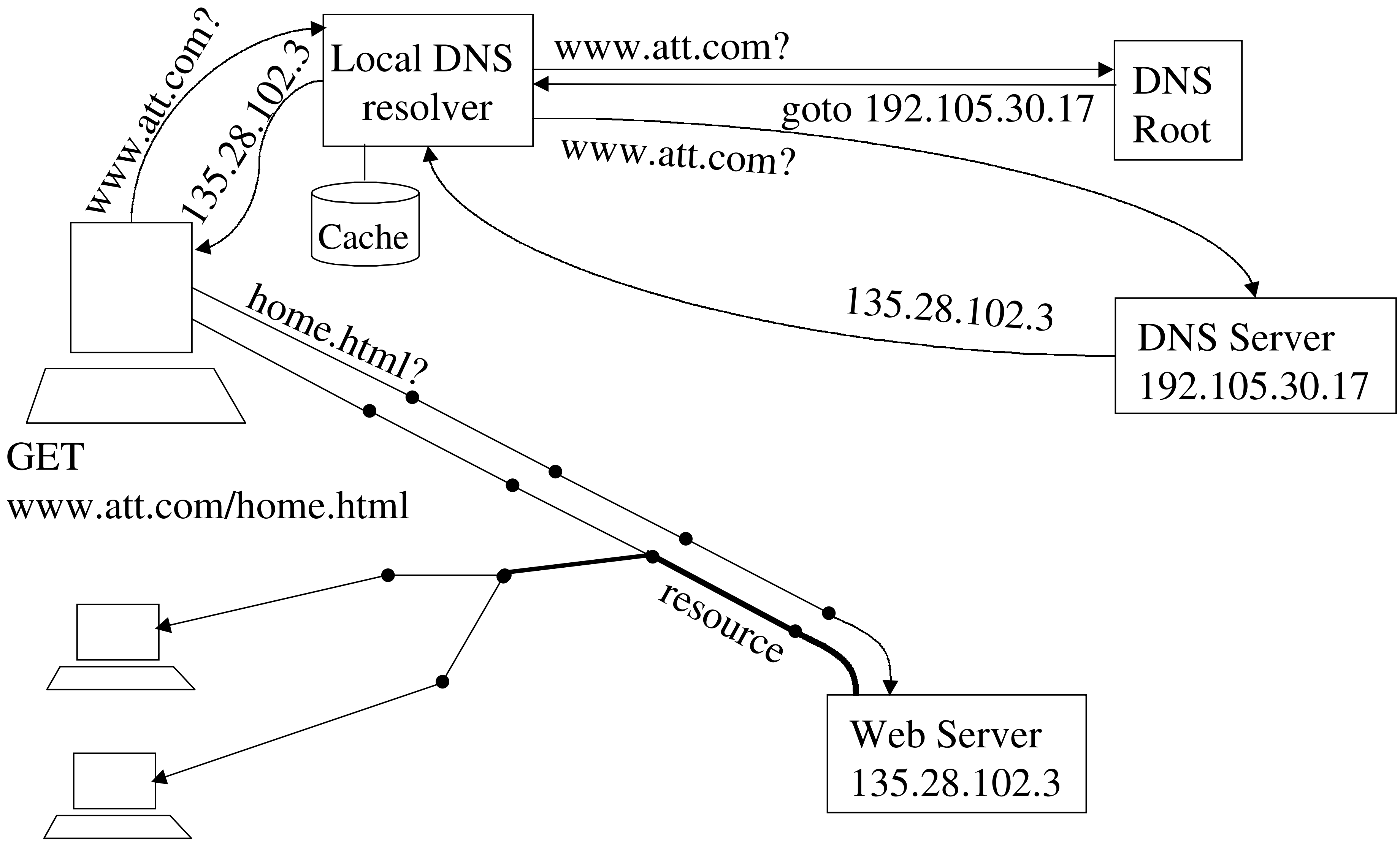
HTTP Protocol

- Request - Response type
- Request
  - Method (What is requested - get object, verify freshness, etc)
  - Resource name
  - Headers
    - "Do not use cache" (Cache control)
    - "Here is information you asked me to store" (Cookie)
    - . . .
- Response
  - Response code ("OK", "Not Found", "Moved", etc)
  - Headers
    - "Do not cache" (Cache control)
    - "Store this information and include with your next request" (Cookie)
    - . . .
  - Resource

Transport

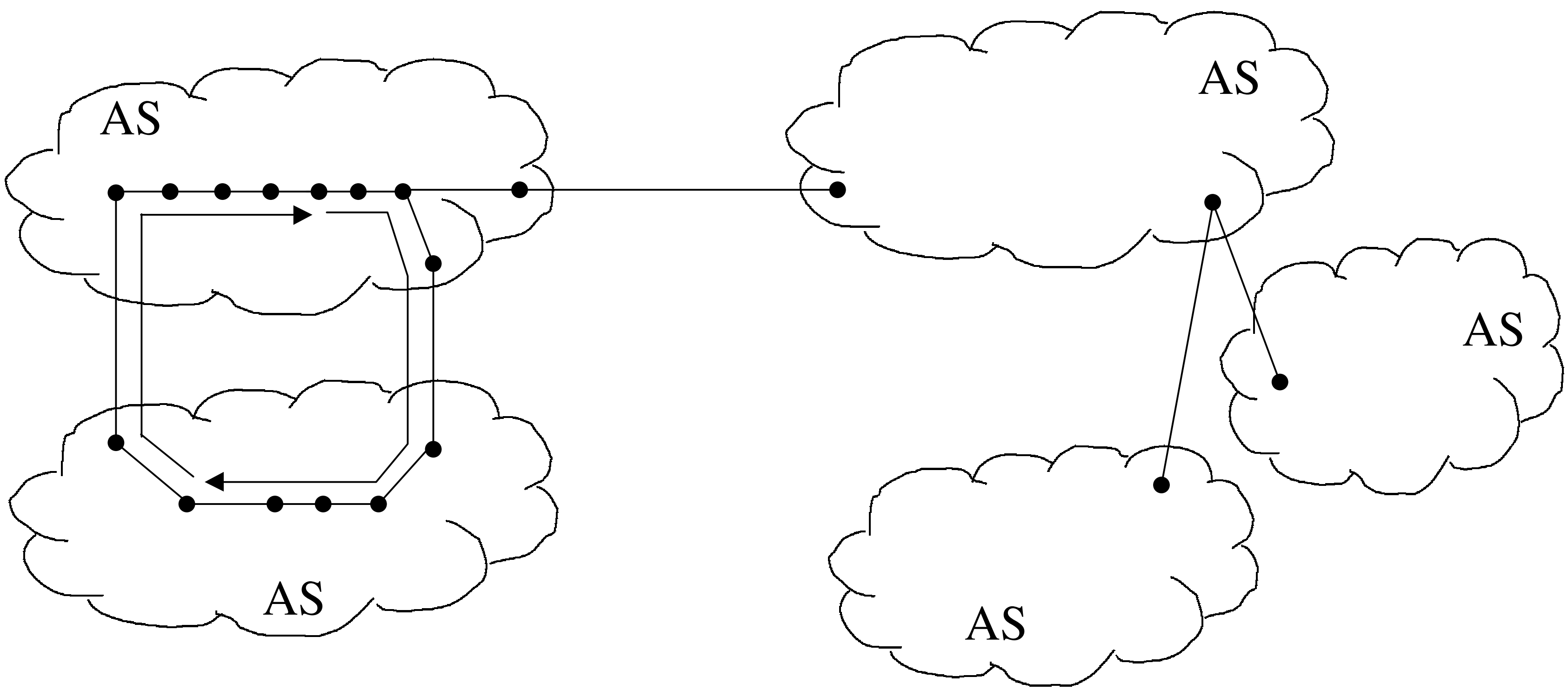
- TCP
  - Connection-oriented; end-to-end
  - Ordered reliable delivery
  - State maintained at both ends
  - Slow start
- Much overhead at start
- Many short connections typical
- Persistent TCP connections (“Keep-Alive” header)

HTTP Interaction



Internet Routing

- Hierarchical routing (Autonomous systems, OSPF areas)
- Asymmetric and variable routes



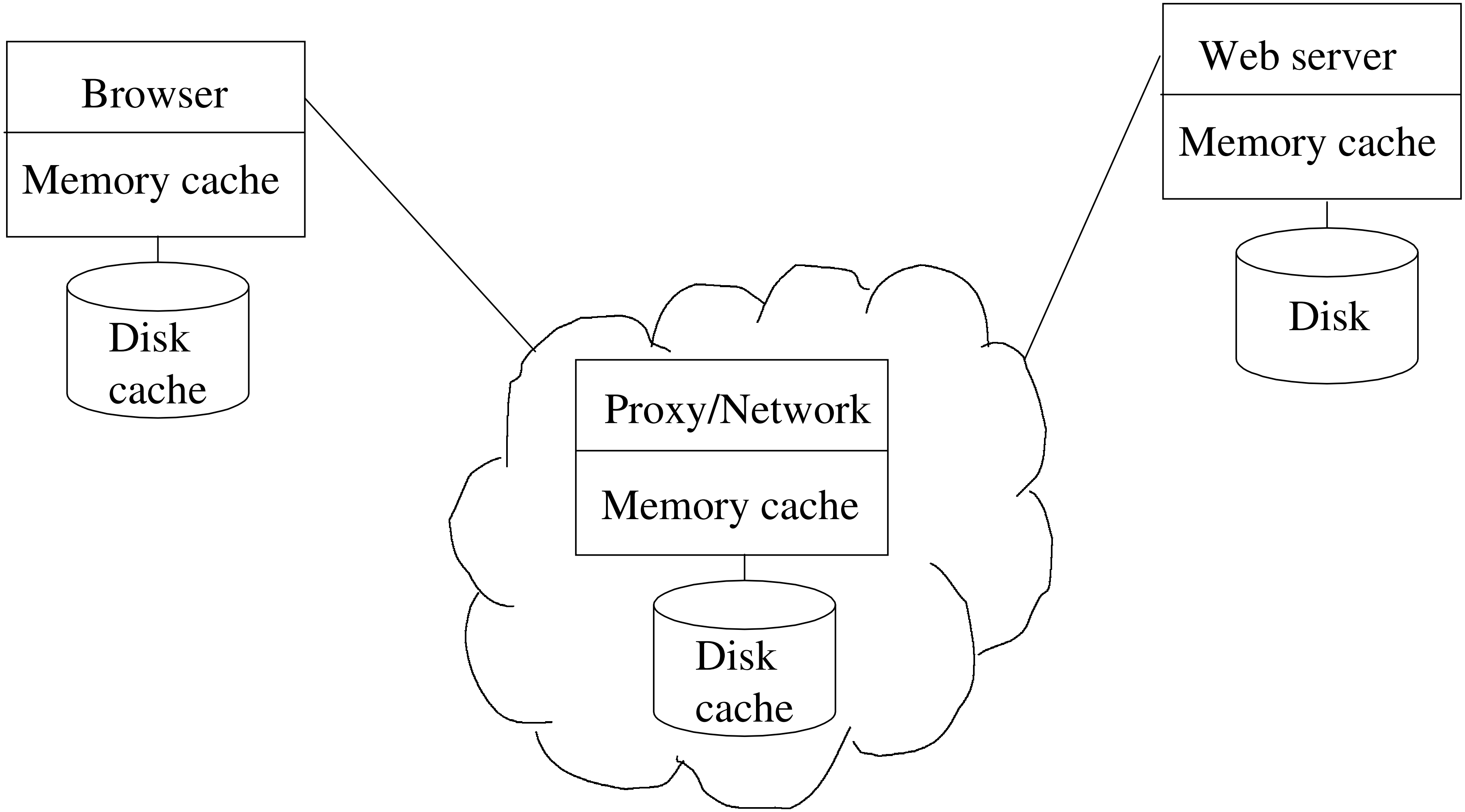
Web Caching vs. Web Replication - a Fuzzy Boundary

- Caching
  - Storing an object at a place that sees the object anyway
  - Examples: browser cache, proxy cache, server main memory cache
  - Cache sees hits and misses
- Replication (“Push Caching”)
  - Storing an object at a place that would not otherwise see the object
  - Example: mirrors
  - Replica sees only hits
- Prefetching (“Push Caching”)
  - Putting an object to the cache before the request

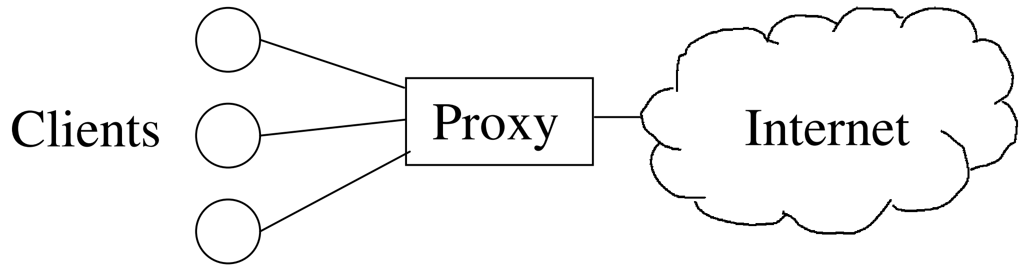
Outline

- Background
- Part 1 - Caching
  - Cache levels
  - Issues in proxy caching
  - Cooperative proxy caching
  - Cache as part of network infrastructure
  - Cache consistency
  - Further issues
- Part 2 - Replication
  - Why caching is not enough
  - Fundamental issues
  - Request indirection on the Internet
  - Further issues
- Summary and Conclusion

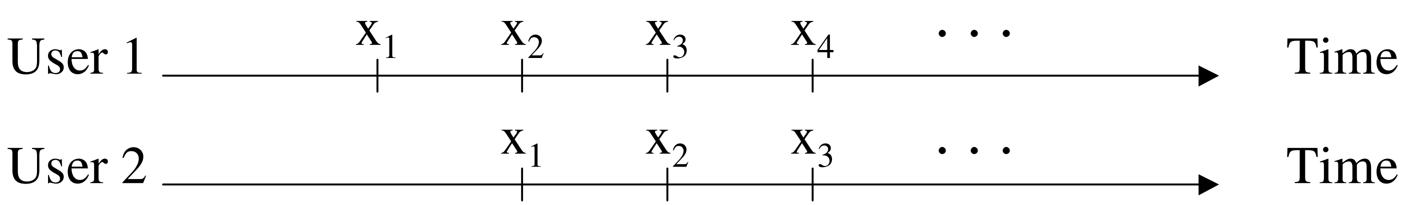
Caching Points



Proxy Caching



- Proxy provides shared cache for many users
- Example:

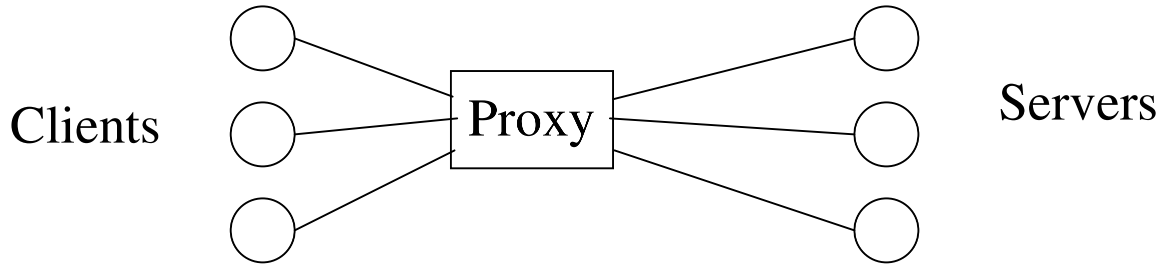


Browser cache hit ratio: 0
Proxy cache hit ratio: 0.5

- Hit ratio grows logarithmically with user population (Gribble & Brewer ‘97)

Proxy Caching: Potential Benefits

- Five proxy traces show 25-43% hit rate with at least 85% of hits due to sharing [Duska et al '97]
- Several ISPs show ≈ 40-50% hit rate
  ≈ 35% byte hit rate
- DEC trace: at most 26% latency reduction [Kroeger et al '97]
  - Hits are for small objects
- Benefits of caching TCP connections [Caceres et al '98]



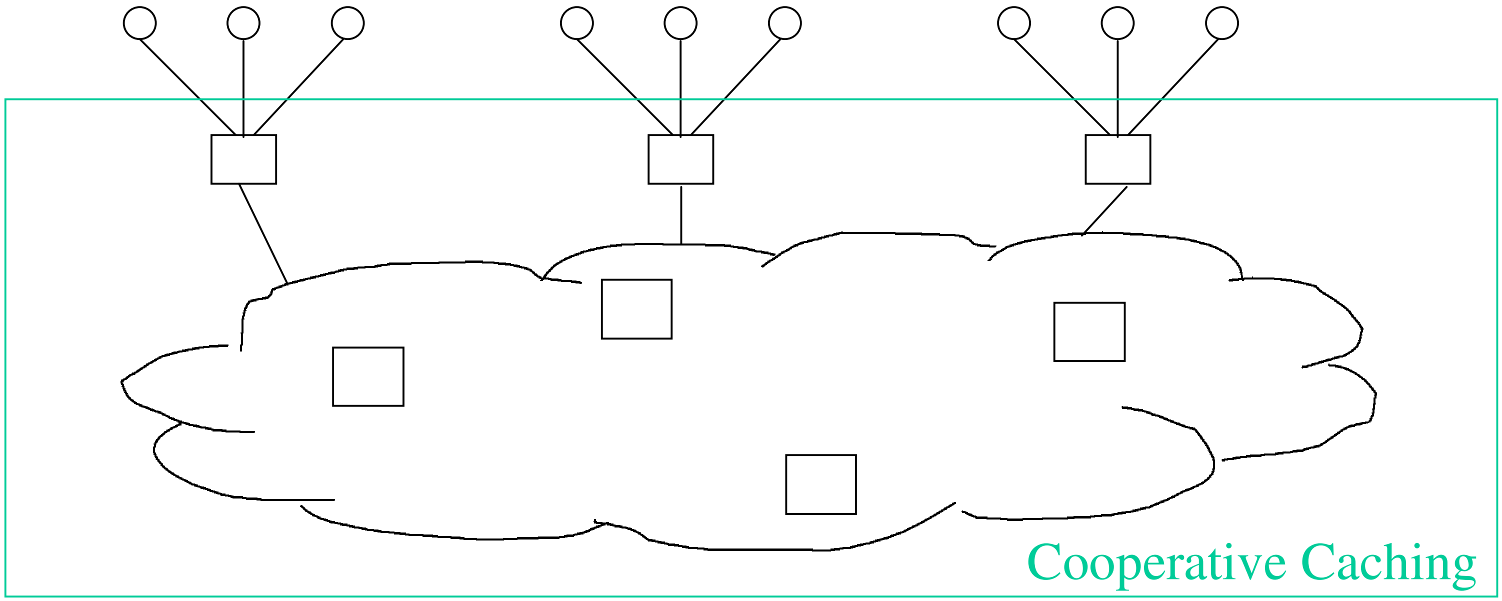
Proxy Caching: Issues and Non-issues

- Non-issue: cache replacement policy for proxies
  - Proxy servers have terabytes of disk space
  - DEC traces: 17K users during 21 days consumed 88GB of cache
- Issues
  - Scalability
  - Cache consistency
  - Document cacheability
  - TCP connection management

Proxy Scalability

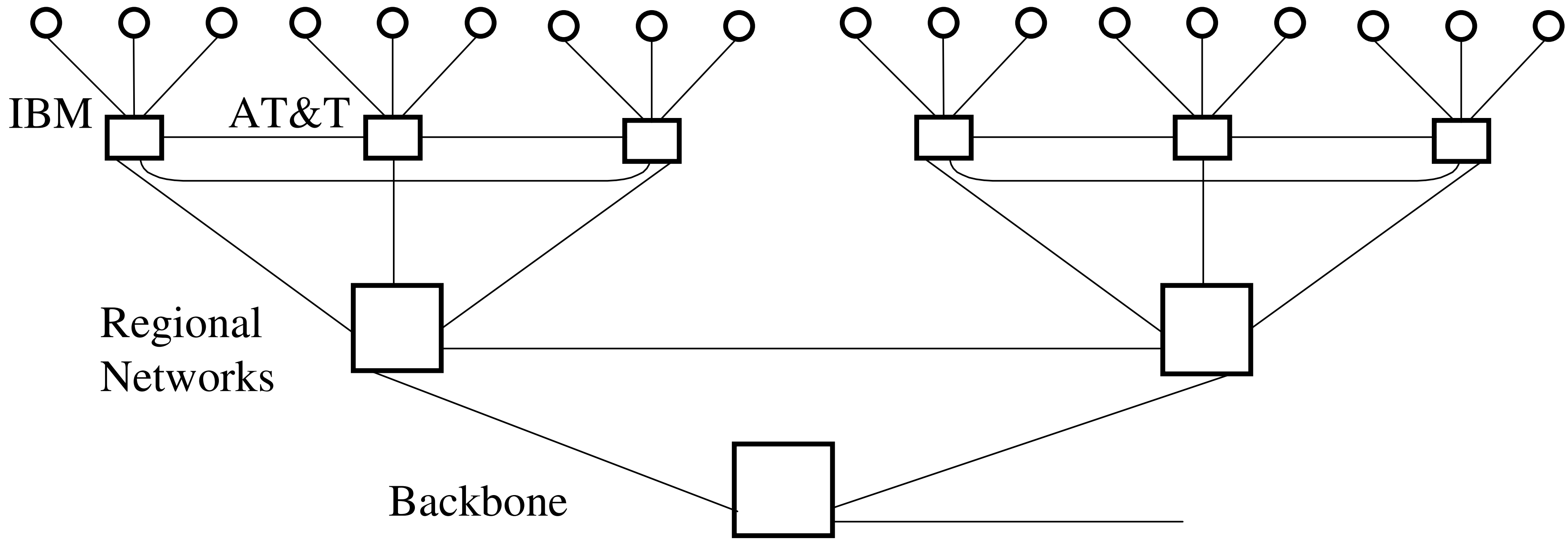
- Large user population is desirable
  - Proxy a bottleneck
  - Population is geographically dispersed
  - May defeat the purpose
- Architectural focus on parallel and distributed systems
  - Geographical scalability vs. capacity scalability
- Market focus on ISPs
  - Lots of users
  - Interested in bandwidth savings
  - Part of network infrastructure

Distributed Proxy Cache



- Issues:
  - Finding out where objects are (location management)
    - Broadcast queries
    - URL hashing
    - Directory service
  - Making sure getting an object from a cache improves performance

Broadcast Queries [Harvest/Squid]



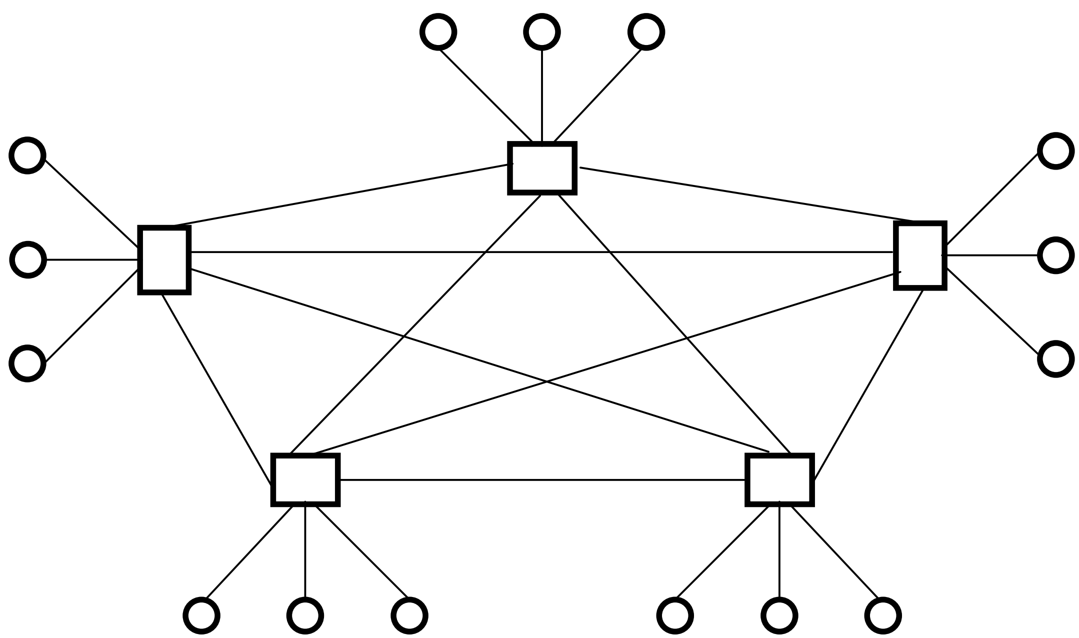
- Send a query to all siblings, parents, and end-server
- Get an object from the first hit
- If all caches missed, send request to the first parent to return a miss

Harvest/Squid Cooperation Model

- Pros:
  - Retrieves an object from fastest cache
  - Little agreement to join hierarchy
  - Simple protocol (Internet Cache Protocol)
    - Cooperation among heterogeneous caches
  - Hierarchy reflects the network topology
- Cons:
  - Does not scale to many siblings/parents
  - Parents may become bottlenecks
  - Deep hierarchies are of limited benefits
    - Impose higher latency (fetches pass through multiple levels)
    - May reduce hit ratio since "cousins" are not queried

Hash-partitioning of URLs (CARP protocol) [Ross'97]

- All proxies agree on a hash function: URL$\rightarrow$proxy
- A given URL is always cached at the same proxy, and other proxies will fetch the URL from that proxy
- Supported by Netscape and Microsoft proxies

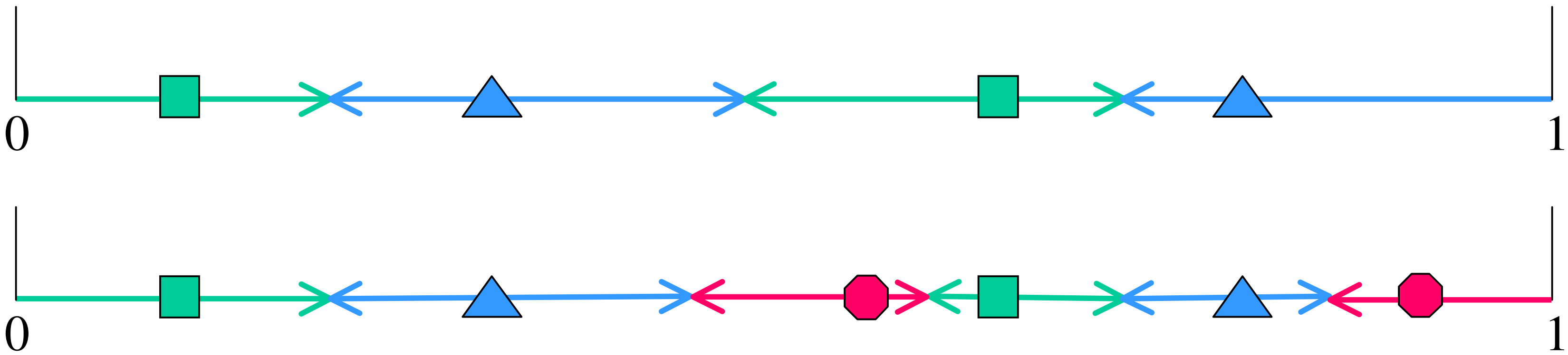


Fault-tolerance

- When a proxy fails or is added, bucket assignment may change drastically
  - For modulo-N functions, almost 100% disruption possible [Thaler & Ravishankar '97]
  - For equally-divided interval, 50% disruption
- Hash disruption reduces hit ratio
- Solution: consistent hashing [Karger et al '97]

A Consistent Hash Function

- Throw some number C “bucket points” per proxy to [0, 1]
- To map URL→ proxy:
  - Map URL to a point x in [0, 1] and find the closest bucket point
  - Take the proxy to which the bucket point belongs

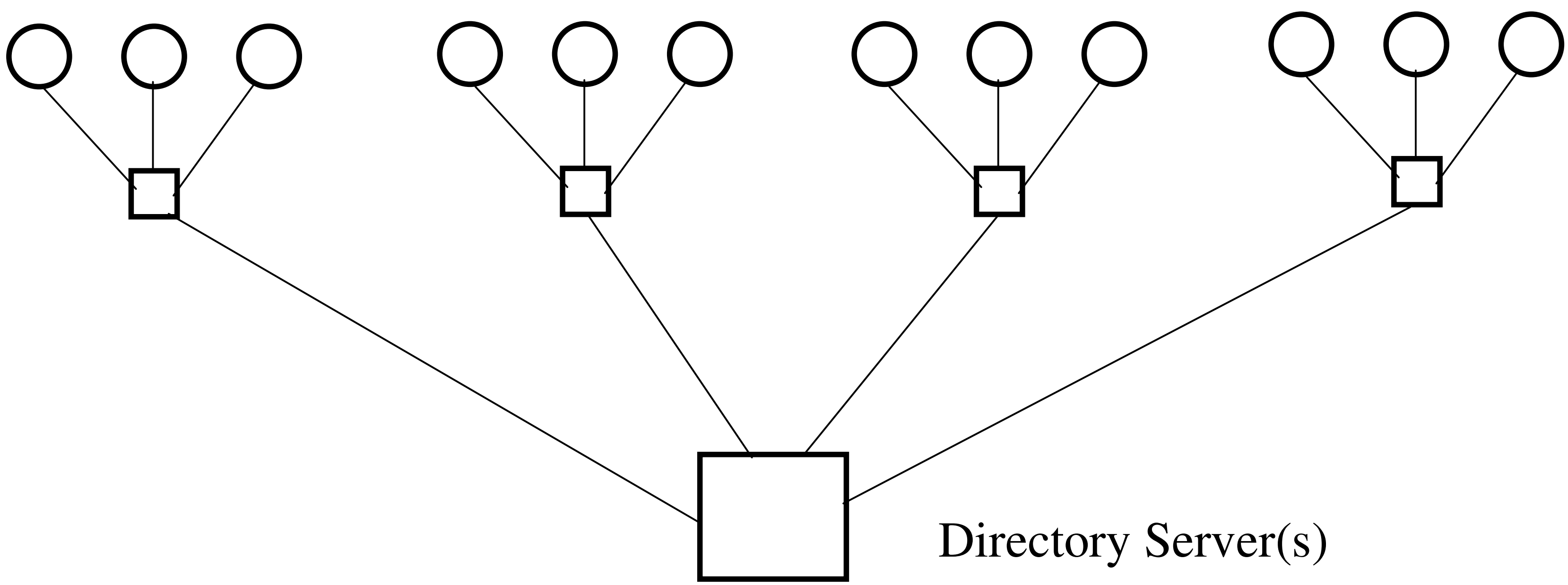


Hash-partitioning of URLs

- Pros:
  - Capacity-scalable
  - Low-overhead location management
  - Efficient use of cache space (no duplicates)
- Cons:
  - Limited geo-scalability
  - Tightly coupled architecture (agreement on hash functions among nodes)
  - Relies on load-uniform hashing
- Well suited for:
  - Intranets
  - Scalable implementation of individual proxy nodes in global cooperation caching schemes

Directory Service

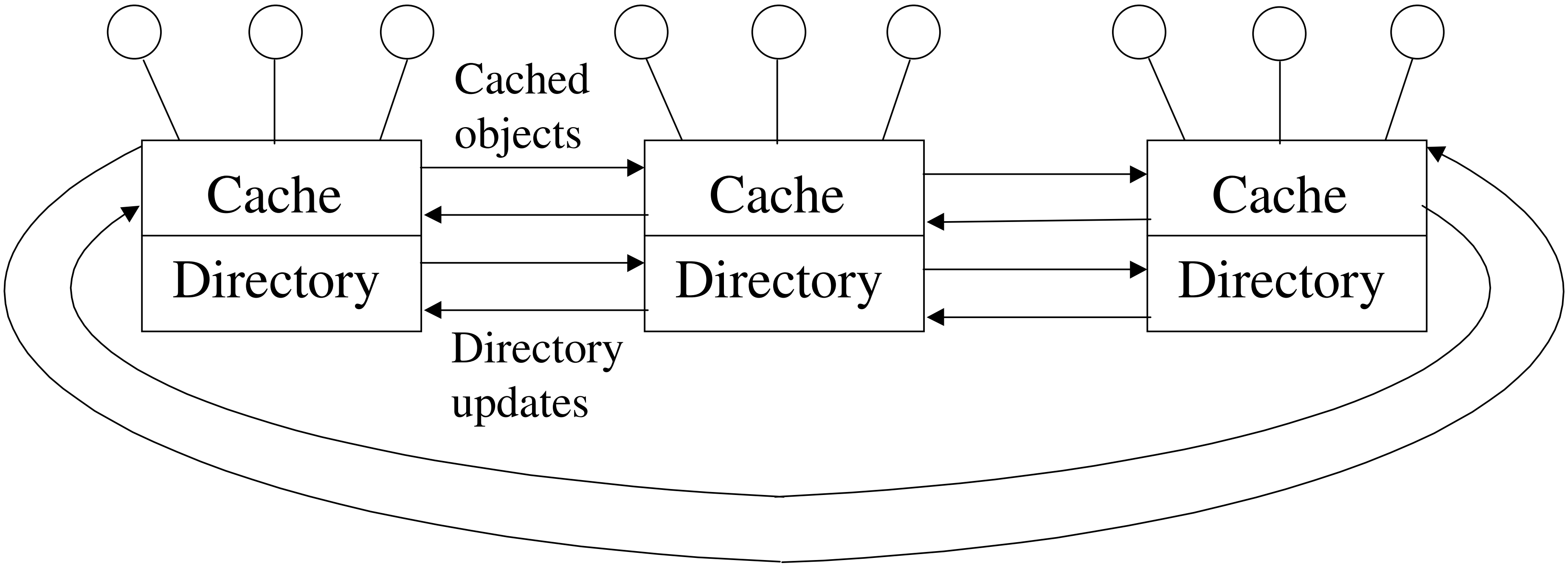
- Using a separate service to track object location (CRISP [Gadde et al '97])
- Directory is notified of each cache's additions and deletions
- Directory is consulted on a local miss



Directory Service

- Pros:
  - Capacity-scalable
    - 266 MHz AlphaStation supports over 85K users with 3ms delay
    - URLs can be partitioned among multiple directories
  - Fault-tolerance
    - Cache failures result in repeat directory queries
    - Directory failures result in going to Internet
    - Hot stand-by for directory is cheap (few directory servers)
  - Loosely coupled
    - Easy to add/remove proxies
- Cons:
  - Limited geo-scalability
  - Location management overhead higher than in the hash scheme

Replicated Directory Service [Relais]



- Eliminate directory query from critical path
- Batched directory updates provide good consistency [Tewari et al ‘98]
- But what about traffic overhead for updates and space overhead for directories?

Summary Cache [Fan et al ‘98]

- Represents directory of a cache as Bloom filter
- Uses m-bit vectors and k hash functions h:
  URL → {1, …, m}

001 000001 0001 000001	URL_1
h_1 h_2 h_4 h_3	
010 000001 0000 010001	URL_2
h_4 h_1 h_2 h_3	
011 000001 0001 010001	Bloom filter
000 100001 0001 000001	True negative
010 000001 0001 000001	False hit

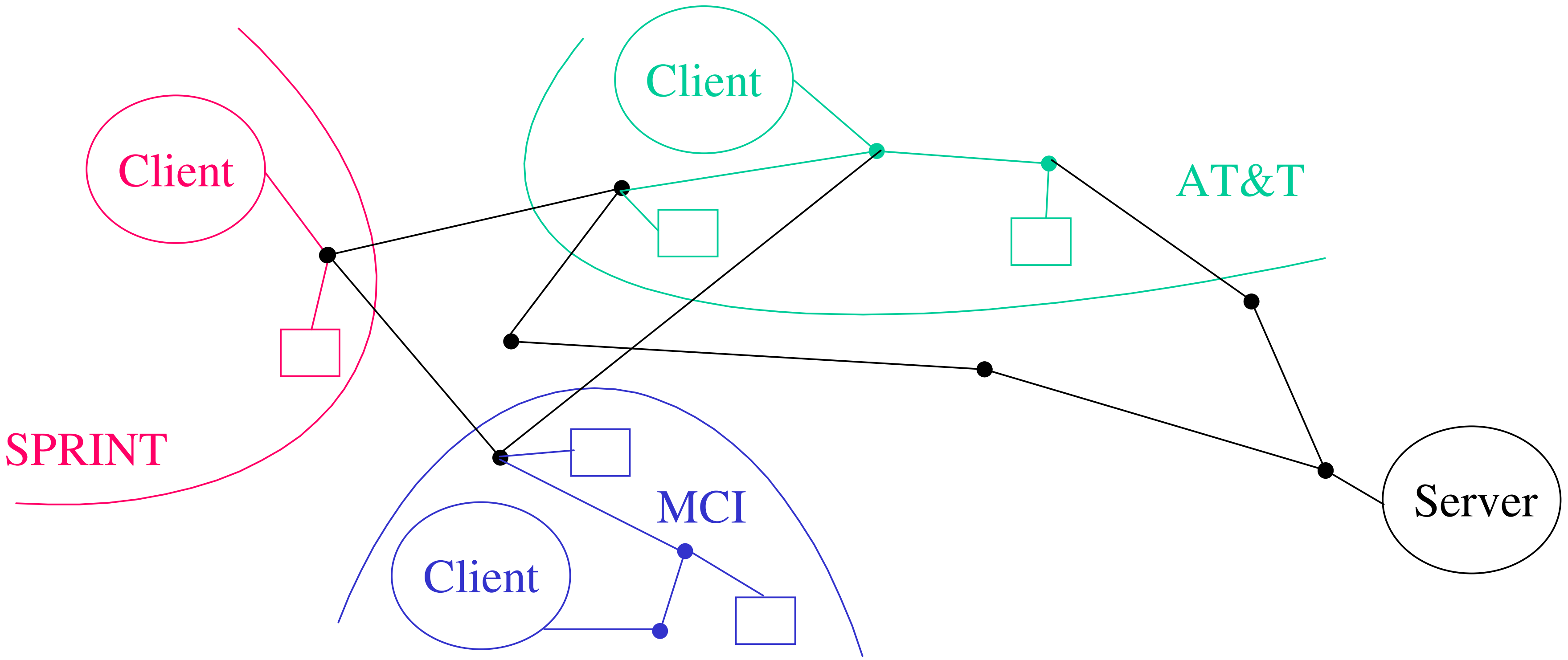
Bloom Filter-based Cache Cooperation

- Each proxy maintains Bloom filters for itself and others
- Periodic exchanges of updates to Bloom filters
- To locate an object, check its bits against all Bloom filters
- $m = 16 \cdot (number\_of\_cached\_URLs); \quad k = 4$

Cache Meshes

- Directory-based and hash-based cooperation do not scale geographically
- Harvest/Squid hierarchies were designed to stop at backbone entry
- What about cooperating (inter) backbone caches?
  - Mirror Image and NLANR efforts are underway
  - Distant caches can be worse than the origin server [Rousskov & Soloviev '98]
  - "Distant" is different for different objects
  - Domain-base distinction is too coarse

Global Cache Mesh



Cache Routing

- Build routes [Michel et al ‘98]
- Snip routes from routers [Grimm et al ‘98]
- Pros:
  - Moves a missed request closer towards the origin server
    - Has to go in this direction anyway
    - No bad decisions for bandwidth
- Cons:
  - Can miss a nearby cache off the path
  - Handing off requests between caches is slow
- Trades latency for bandwidth

Neighborhood Caches
[Rabinovich et al ‘98]

- A proxy clusters web sites into sets based on “cost” of accessing the origin server
- A neighborhood for the cluster contains proxies with lower access cost
- A proxy discards entries for objects cached outside the corresponding neighborhood
- For a missed request, a proxy fetches the object from a remote proxy if listed in its directory, or from the origin server otherwise

Neighborhood caches - Scalability

- Capacity
  - Directory size and request rate are limited by the neighborhood
- Geography
  - No long-haul directory update propagation
  - No synchronous directory queries

Outline

- Background
- Part 1 - Caching
  - Cache levels
  - Issues in proxy caching
  - Cooperative proxy caching
  - Cache as part of network infrastructure
  - Cache consistency
  - Further issues
- Part 2 - Replication
  - Why caching is not enough
  - Fundamental issues
  - Request indirection on the Internet
  - Further issues
- Summary and Conclusion

Cache Deployment in ISPs

Explicit

- Requires browser configuration
- Flexible proxy placement in the network
- Potentially more performance gains

Transparent

- No configuration at the client
- Better behavior under failures
- Requires careful proxy placement

L4/L7-switches

- Open
- Requires extra switch

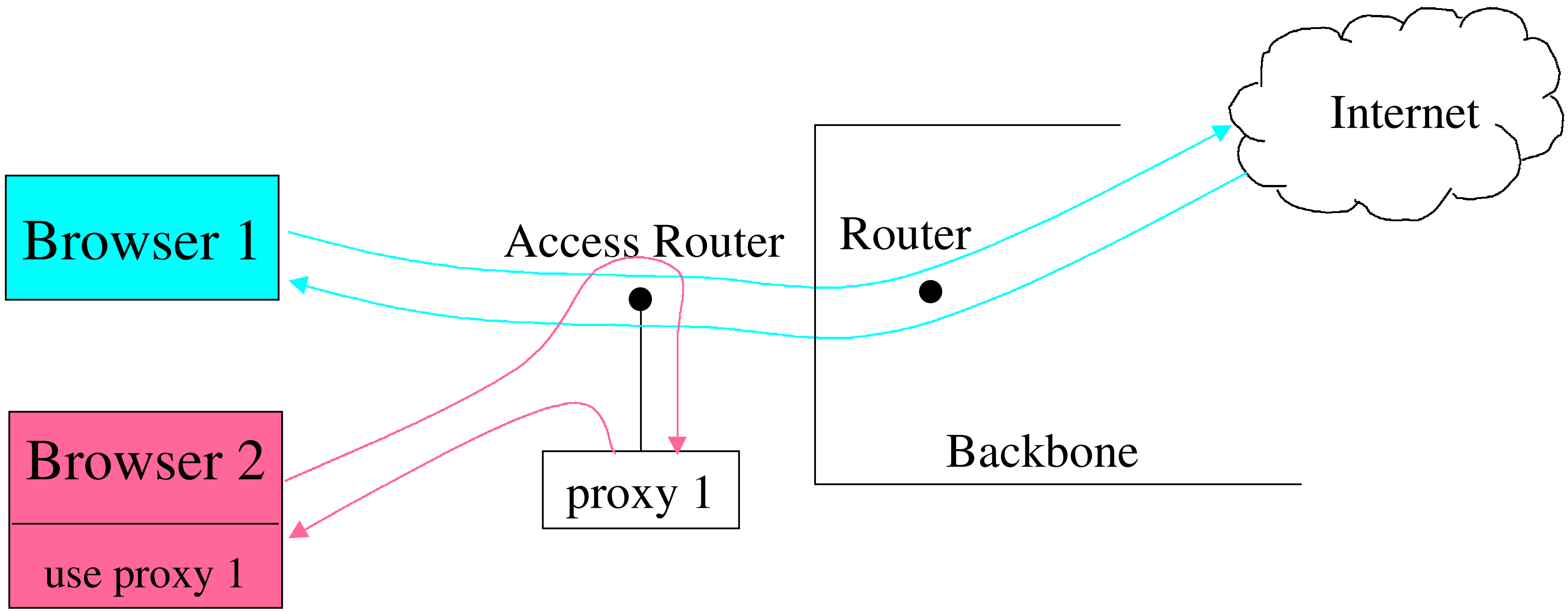
CISCO

- Proprietary
- Uses existing routers

InfoLibria

- Snooping link

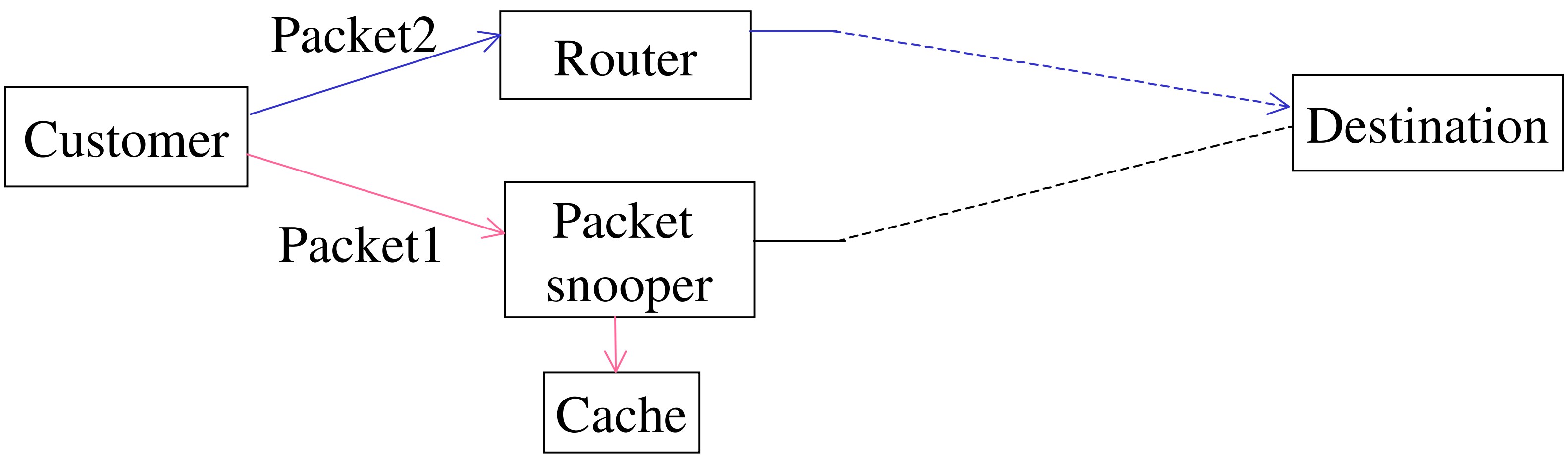
Explicit Proxy



- Can be used by any customer
- Can be placed anywhere in the network

Transparent Proxy

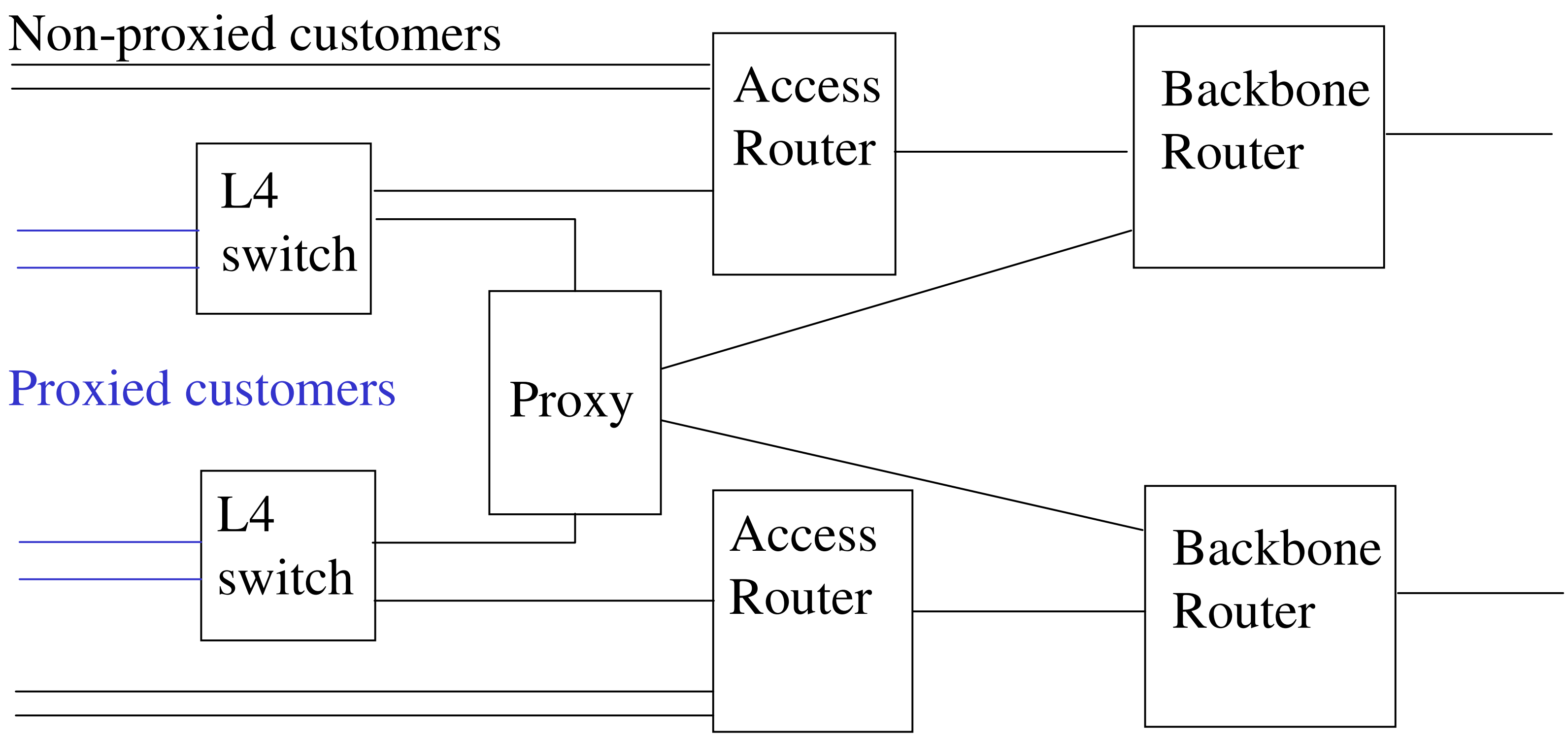
- Router snoops on packets, diverts all TCP port 80 traffic to proxy
- Proxy impersonates all destinations



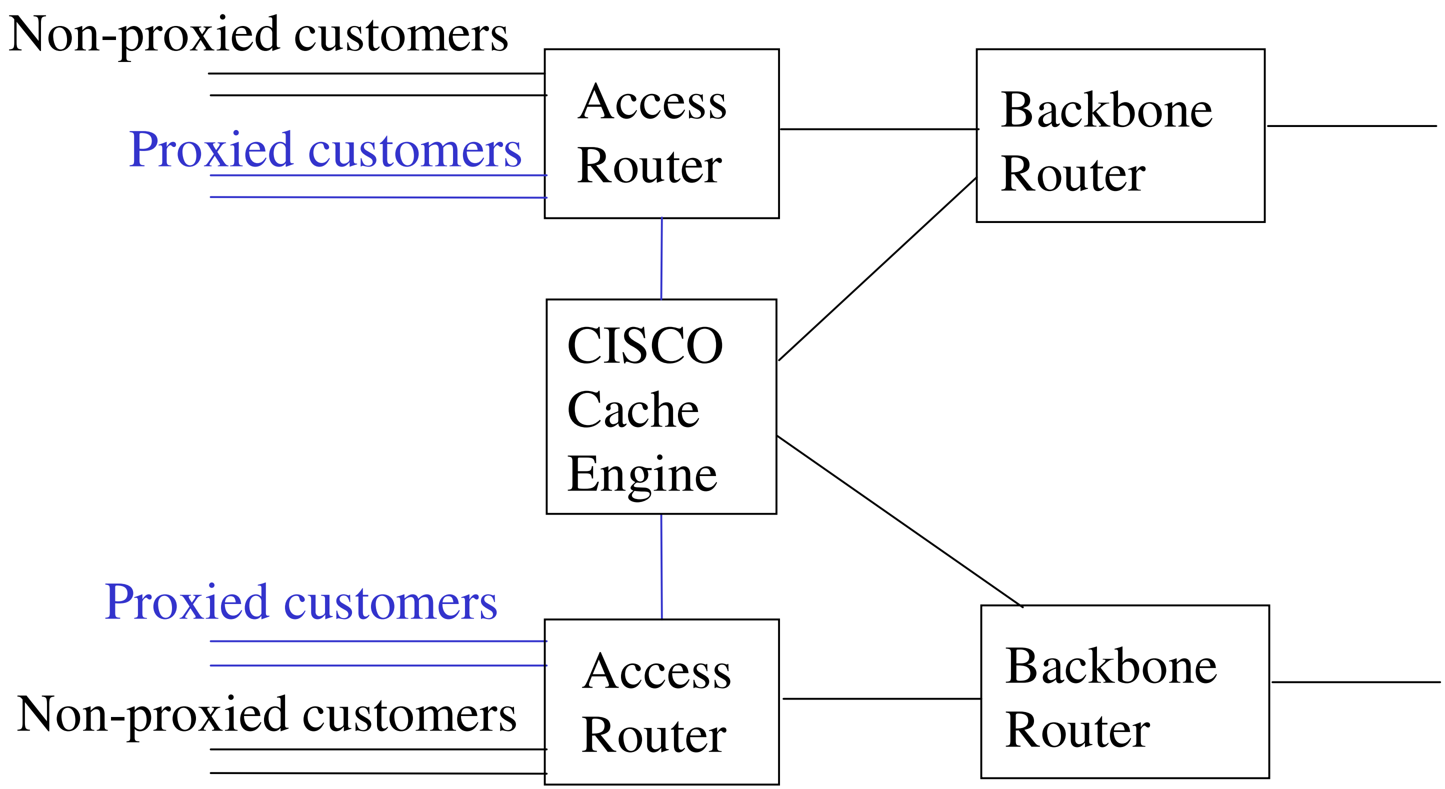
  - Snooping router must see all packets from a client
  - Inappropriate for customers with multiple ISPs

L4 Switches

- Alteon's AceSwitch
- ArrowPoint's ContentSmart (also L7)



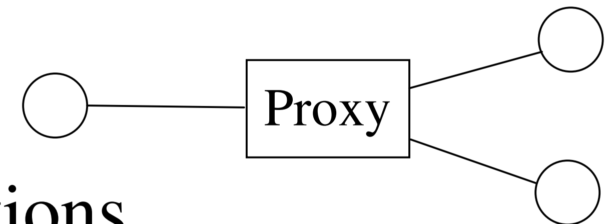
CISCO IOS Routers and Cache Engine



- Proprietary protocol to talk to proxy (can only be used with CISCO cache engine)

Pitfalls of Transparent Proxies

- Unstable routing breaks connections
- Proxy breaks IP address-based authentication
  - Gleim/CyberCash/Digex example
  - Difficult to bypass



- Limited caching of TCP connections
- Reusing a browser → proxy connection for accessing multiple Web sites is impossible
- Using IP address instead of DNS name may reduce hit rate

Cache Consistency

- Cache validation
  - Weak consistency
  - Intended mechanism - *EXPIRES* response header
    - Responses rarely include *EXPIRES*
  - Alex protocol [Cate '92]
    - Uses *Last-Modified* header, inserted automatically
    - TTL = 20% * (NOW - Last_Modified)
    - Validate once a *TTL* interval
    - Under 1% stale hits

Cache Consistency

- Server invalidation [Lui & Cao ‘97]
  - Callback to clients upon page change
  - (Almost) strong consistency
  - Overhead for invalidation lists is a concern
- Piggybacked validation/invalidation [Krishnamurthy & Wills ‘97,98]
  - (Almost) strong consistency

Further Issues

- Caching “uncacheable” pages
  - Template caching [Douglis et al ‘97]
  - Delta encoding [Banga ‘97, Mogul ‘97]
- Proxies as information filters
  - Intel’s QuickWeb
  - “Soft cache” [Kangasharju ‘98]
- Proxy server architecture
- Main memory caching at browser, proxy, end-server
- Pre-fetching
- Multicast and push caching [SkyCache, Almeroth et al’98, Touch ‘98]

Outline

- Background
- Part 1 - Caching
  - Cache levels
  - Issues in proxy caching
  - Cooperative proxy caching
  - Cache as part of network infrastructure
  - Cache consistency
  - Further issues
- Part 2 - Replication
  - Why caching is not enough
  - Fundamental issues
  - Request indirection on the Internet
  - Further issues
- Summary and Conclusion

Web Replication

- Caching is not enough
  - Limited performance gains
  - “Cache busting”
- Two-fold goal of replication:
  - Load balancing
  - Geographical distribution
- Fundamental issues:
  - Distributing requests among replicas
  - Deciding on the number and placement of replica
  - Ensuring replica consistency

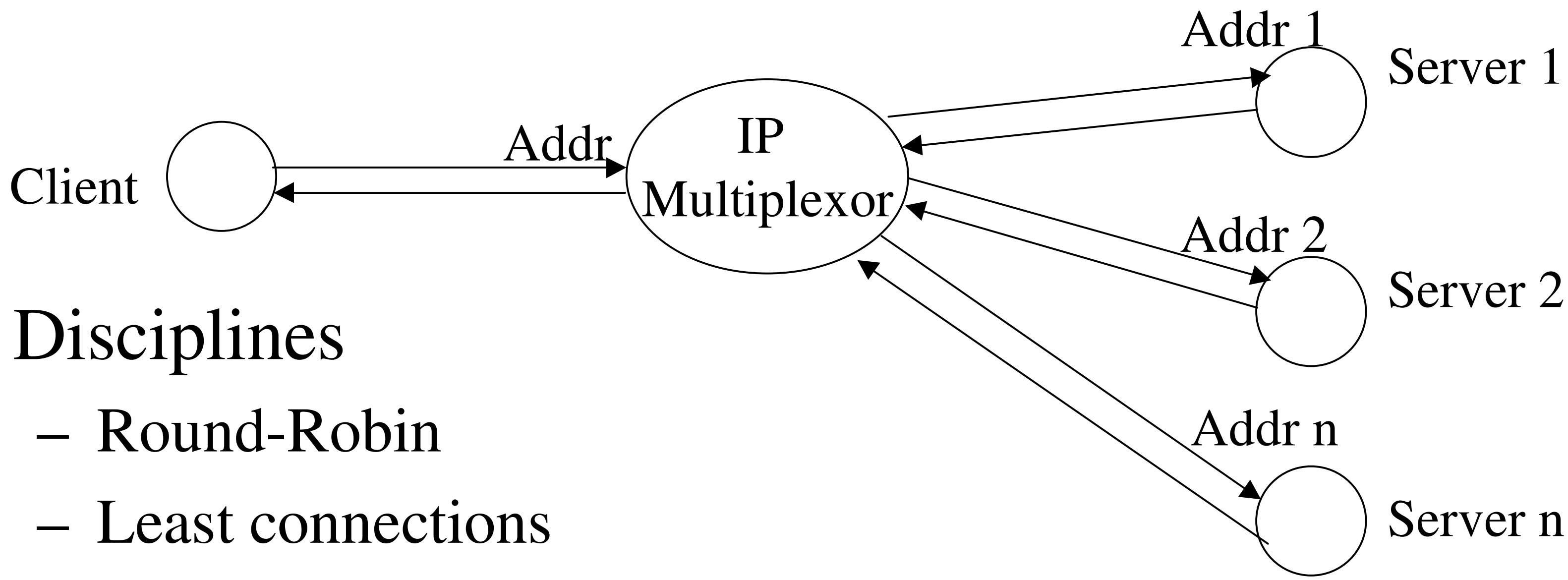
Replica Consistency

- Types of objects
  - Immutable
    - Static
    - Read-only queries
  - Commuting updates
    - Hit metering
    - Some e-commerce
  - True read-write
- Objects on the Web [Manley & Seltzer '97, Mogul et al 97, Caceres et al '98]
  - More than 85% static objects
- Only small fraction of objects require synchronization on access

Request Distribution: Points of Indirection

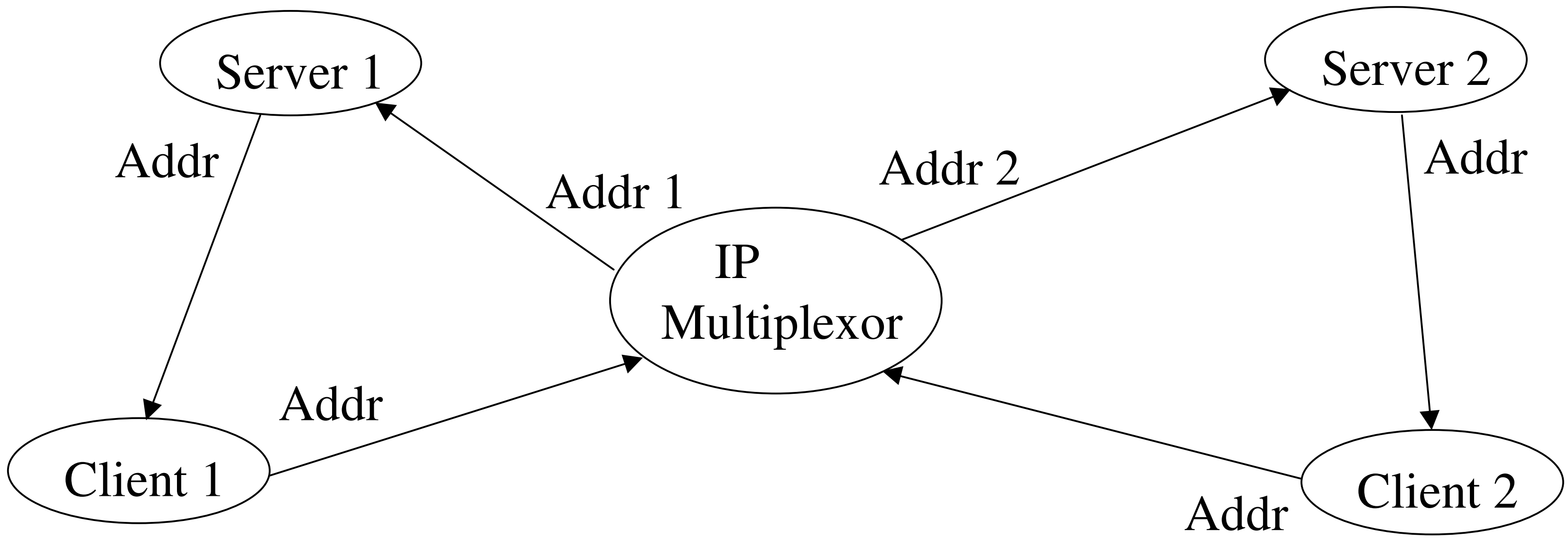
- User (explicit mirrors)
- Client
  - [Fei at al ‘98, Baentsch et al ‘97, Yoshikawa et al ‘97]
- DNS
  - [Katz ‘94, Colajanni ‘97, Dias ‘96, CISCO Distributed Director]
- HTTP
  - [WindDance, Distributed Director, Bestavros ‘95]
- IP
- Anycast
- Combinations of the above

IP-Level Indirection



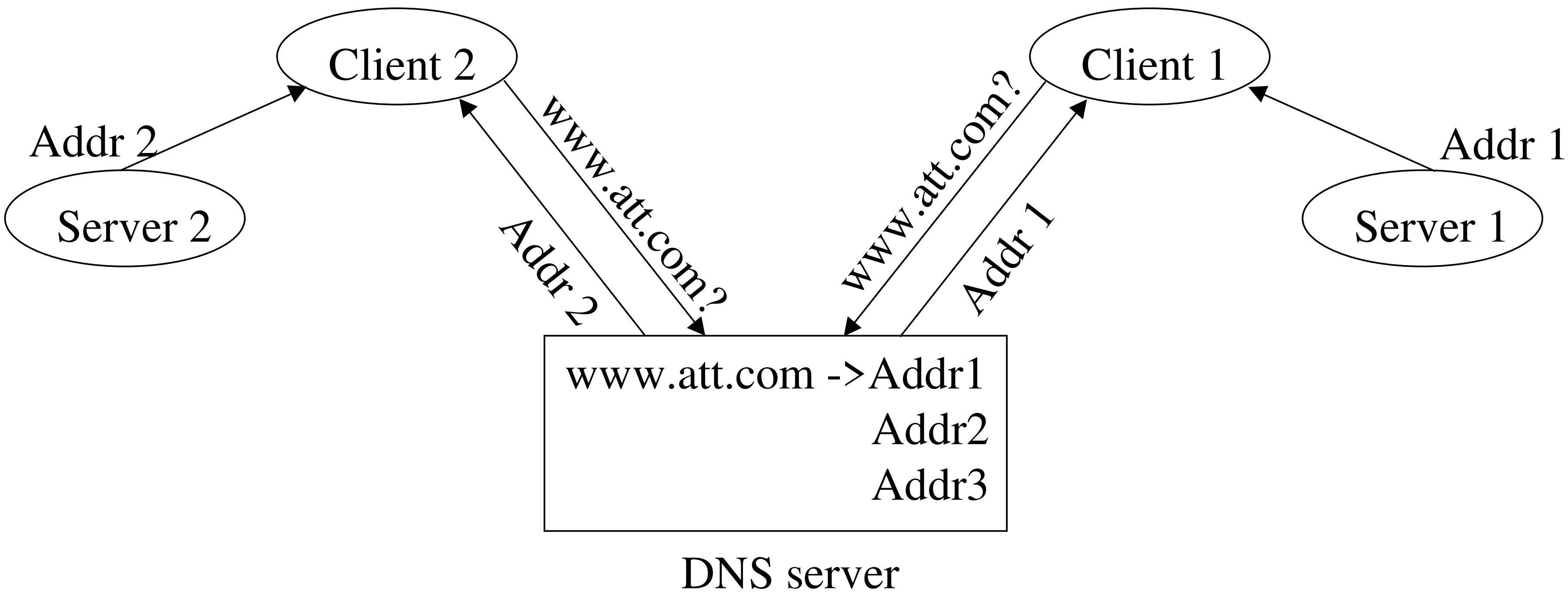
- Disciplines
  - Round-Robin
  - Least connections
  - Fastest
  - Connections threshold
- Many products available
  - CISCO Local Director, HydraWeb, etc
- Review in "Network World" Sept 22, 1997
- Standard DNS server
- Do not scale geographically

IBM's Network Dispatcher



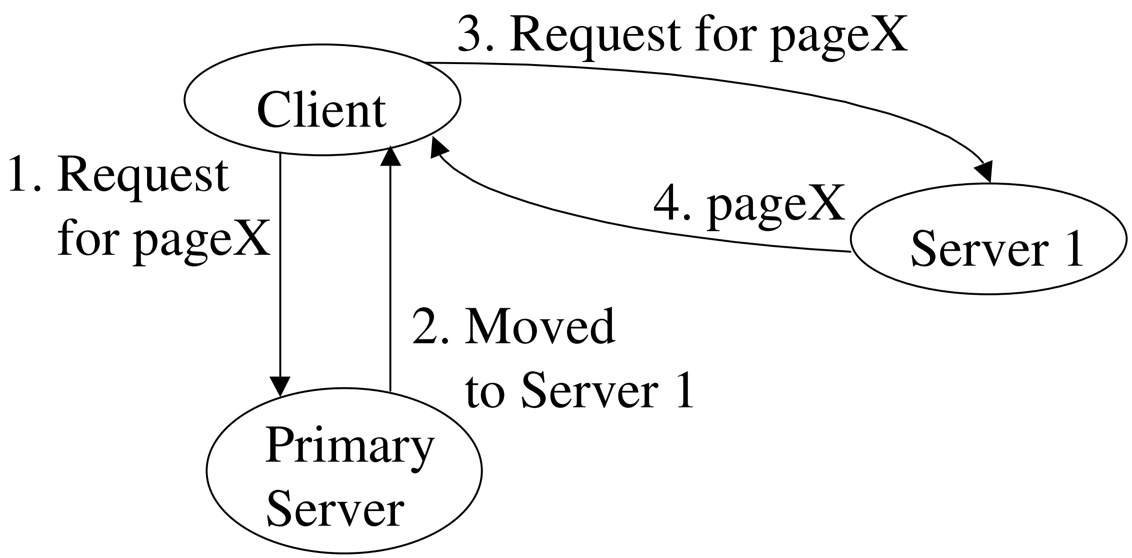
- Uses IP spoofing
- Scales better geographically

DNS-Level Indirection



- A site is a replication unit
- DNS caching complicates control
- Hard to change replica sets
- Small availability gains

HTTP-Level Indirection



- Heavy weight
- Primary server sees all requests

Client-Based Indirection

- Possible points of indirection
  - DNS resolvers [Fei et al ‘98]
  - Client proxy [Baentsch et al ‘97]
  - Applet downloaded from primary server [Yoshikawa et al ‘97]
- Clients know replica sets
  - Security issues
  - Hard to change replica sets
- Clients assign requests to replicas
  - Hard to control request distribution

Further Issues

- Algorithm for request distribution
- Dynamic replica allocation [Heddaya & Mirdad '97, Rabinovich et al '98]
  - Who makes decisions
  - Algorithms

Conclusions

Researchers:

- Mind the reality
  - Asymmetric and unstable routing
  - Multi-packet requests and objects
  - DNS caching
- Beware the scale
  - Danger of toy experimentation
- Know the design space
  - Choose the right problems

Conclusions

<u>Content providers</u>:

- Yes, you can defeat caching, but why?
  Harmful habits:
  - URL aliases
  - Cookies for fun
  - IP-based authentication
- Think performance. Shared infrastructure:
  - Your sloppy design affects everybody
  - You cannot always mask sloppy design by more hardware

Summary

- Caching and replication are primary tools in dealing with Internet scale
- Very rich research area
  - Multiple levels of caching
  - Cooperative caching
  - Replication algorithms
  - Inter-play of caching and replication with Internet/Web protocols and infrastructure
  - . . .
- High impact/visibility area
  - Many start-ups from academia (NetCache, Inktomi, InfoLibria)