Software in the Network:

What Happened and Where to Go

Prof. Eric A. Brewer
UC Berkeley
Inktomi Corporation

The "old" Internet

- Local networks with local names and switches
- IP creates global namespace and links the local networks
- Routers connect IP names worldwide, and move packets
- Big networks cooperate (peering)
- Circa: 1969 to 1998

Traditional Design

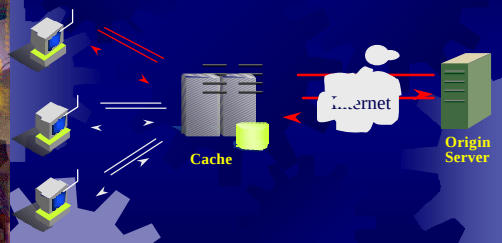
- Build on top of datagrams
  - Flexible
  - Easy to multiplex (efficient)
- State at the edges
  - "fate sharing" => host & state die together
  - Simpler, more reliable core
- No "software" in the middle
  - Really no servers…

Outline

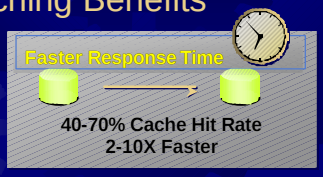
- Arrival of Software
  - The Spread
- Some Analysis
  - Design Principles
- Where to go from here…

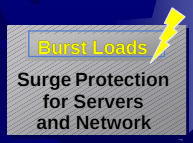
The Arrival of Software (in the network)

1) Basic Caching

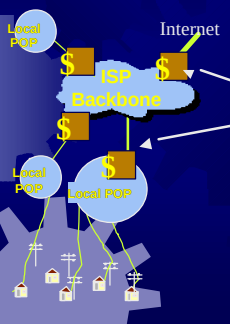


Caching Benefits





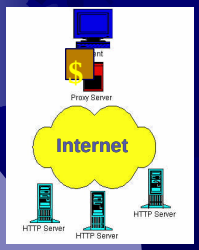
Caching for ISPs



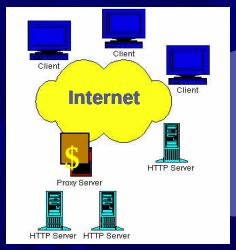
- Moves data closer to consumer
- Backbone caches for saving bandwidth
- Edge caches for QoS
- Does it work?
  - > 10 billion hits/day at AOL
- Even more important for cable/DSL

2) Reverse Caching

Forward Proxy Cache
Cache handles client requests

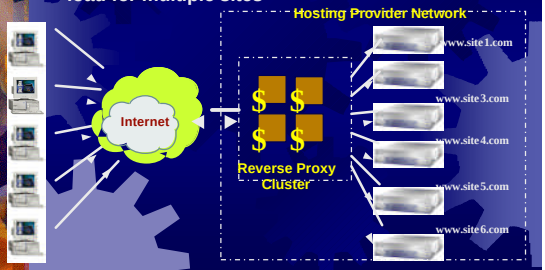


Reverse Proxy Cache
Cache fronts origin server



Surge Protection

A cluster of caches in reverse proxy mode buffers load for multiple sites

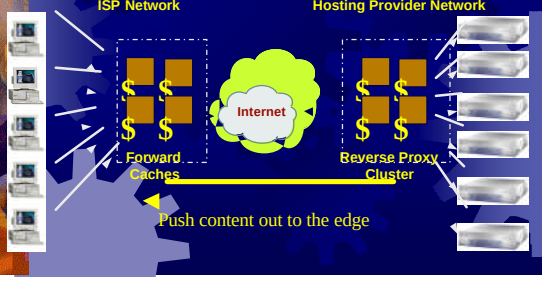


Surge Protection

- "Peak Insurance"
  - Available bandwidth for unanticipated traffic
  - Analogous to "over-draft" protection
- Additional bandwidth for big events
- Guaranteed Availability
  - covers transient server faults
  - Different service level agreements (SLAs) depending on peak protection

3) Content Distribution

We can connect these caches!

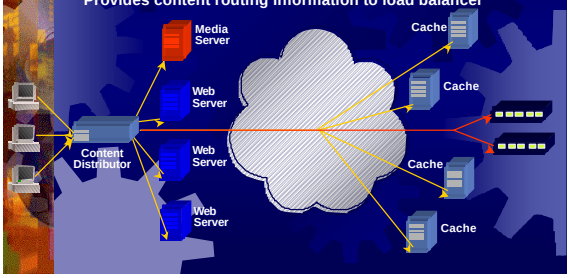


Content Delivery

Retrieves content from development or management source

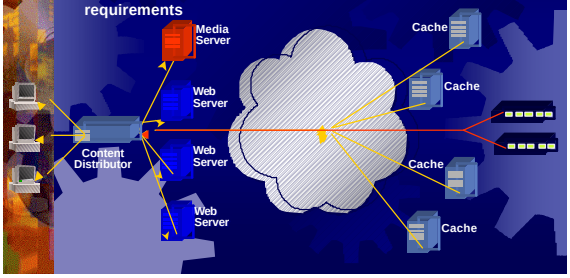
Replicates and distributes to destination servers/caches

Provides content routing information to load balancer



Information Backflow

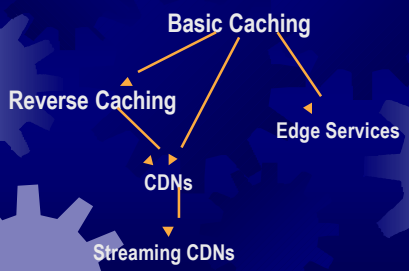
- Aggregates data about content usage and performance
- Tracks whether service level requirements are being met
- Dynamically adjusts content availability to meet those requirements



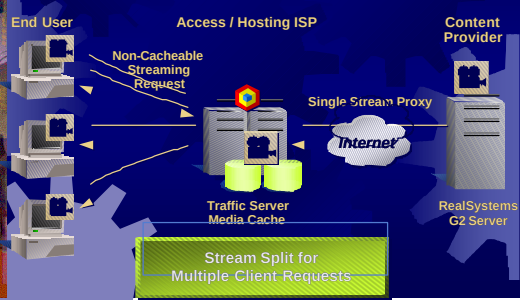
Example: Cable & Wireless

Global Content Distribution

The Spread…



4) Streaming Media CDN



5) Edge-Based Services

- Potentially high local bandwidth
  - Cable or DSL
- Wide-area bandwidth efficiency
- Fast response time
  - … and more predictable
- Integrate localized content
- Associated with *client*, not server
  - client pays (typically their ISP)

Edge-service Examples

- Caching
  - Exploits response time, BW efficiency, and high local BW
- Filtering
  - A form of local content transformation
- A/V
  - Exploits bandwidth efficiency, high local BW, predictable response time

More Examples

- Transformation
  - Change content on the fly for end user
  - Support for WebTV, Palm Pilot, Smart Phones
- Software Rental, Storage
  - Exploits high local BW
- Games
  - Exploits low predictable latency

Some Analysis

A New Internet

- There's a whole new layer on top…
- Redefines naming/routing/multicast
- Stores data in the middle of the network
- Adds new services for fault tolerance
- Adds transformation of data on the fly
- Adds distribution control (push/freshness)

… and it appeared silently almost overnight

What worked?

- Much better *control*
  - *High-level objects, not packets*
- Very flexible
  - High rate of innovation
- Generally secure (with hardware help)
- Sometimes cost effective

The Control Issue

- Managing the network at the datagram level is too hard
  - One solution: flows
  - Better solution: higher-level objects
    - http objects, streams, storage, etc.
- Tasks made easier:
  - Whole-object caching
  - Security (access control, filtering)
  - Streaming (control channel)
  - Network Management

Cost Effective?

- Two reasons to deploy:
  - Cost savings
  - Creation of new revenue (e.g. CDN)
- In general:
  - Cost savings were effective
  - Revenue generation usually was not…
    - problems were rarely technical…

What didn't work?

- Bandwidth limitation of servers?
  - caches really at the edge … not in the core
- Transit networks don't need software…
  - Low-level network by definition
  - No cost savings, no revenue creation
- Peering
- Very hard to write reliable services
  - Extreme performance
  - 24x7 availability, dark POPs
  - Easier if for your own use (e.g. Akamai)
  - Inktomi had "plug in" API for edge – but few users

Peering didn't work (yet)

- Two networks need software to interoperate…
  - E.g. CDNs, streaming
- Solution: some new peering protocol
  - *Politically very tricky!!!*
  - Need to hide some customer details (privacy)
  - Both sides want some advantage (!)
- Actual solution: own the whole thing
  - (no peering)
  - Exception: regional hosters did peer some

Where to Go…

Peer-to-Peer

- Claim: real P2P will need software in the network
  - Don't now only because of legality
    - > 50% traffic is P2P (in some networks)
    - But to handle it is to enable it…
- E.g. storage, legal content swapping
  - Need search, access control, …
  - Need peering?
    - Claim: not for now, but eventually
    - We'll see single owner versions first…

Software Changes the Agenda

- Quality of Service?
  - Actual quality depends on caches, not on bandwidth reservation
- Is bandwidth the problem?
- What about IP multicast, or DNS?
- What about secure end-to-end connections?
  - Does it break the new Internet? (yes)
  - Does it have to? Maybe not…

The Real Solution

Separate control and data network!

- Datagrams for the data
  - High-speed ASICs
  - Operations on packets
- Software for the control
  - Flexible servers, new high-level protocols
  - Operations on objects, flows
  - Security, management, storage, naming
- => "Server-controlled Router"

Conclusions

- Software in the network is real
  - And here to stay…
- Flexibility is enabling
  - But better for control than data
- Still hard to write
  - (but routers are hard too)
- Move towards a more flexible mix:
  - Software present
  - … but not on the data path

End of Talk

Charaterization Summary

- VAS usually orthogonal to web services
  - A/V, filtering, caching applies to all sites
  - Transformation is content tuning for the edge group; different groupings are different services
- Focus on:
  - high-bandwidth services
  - low-response time services
  - integration of private/local data

Toward Smarter Networks

- Automatic replication
- Document routing
  - determination of nearby caches
- Compression & mirroring
- Data transformation

The action is at the "edge" of the network

This is a new Internet

- The **action** is between the caches
- NOT between the routers
- There are new protocols in use:
  - push content to the edge
  - invalidate remote content for freshness
  - collate remote logs into a single log
  - A/V streaming that works