

Peer-to-peer models

Client-server computing

- servers provide special services to clients
- clients request service from a server

Pure peer-peer computing

- all systems have equivalent capability and responsibility
- symmetric communication

Hybrid

 peer-to-peer where servers facilitate interaction between peers

Evolution of the Internet (services)

First generation

- multiple smaller webs
 - telnet, ftp, gopher, WAIS

Second generation

- Mosaic browser
 - retrieval process hidden from user
 - merge all webs into a world-wide-web

Third generation

- peer-to-peer (?)
- distributed services; distribution hidden from user

Peer-to-peer networking

"If a million people use a web site simultaneously, doesn't that mean that we must have a heavy-duty remote server to keep them all happy?

No; we could move the site onto a million desktops and use the Internet for coordination.

Could amazon.com be an itinerant hoarde instead of a fixed central command post? Yes."

- David Gelernter The Second Coming: A Manifesto

Triggers

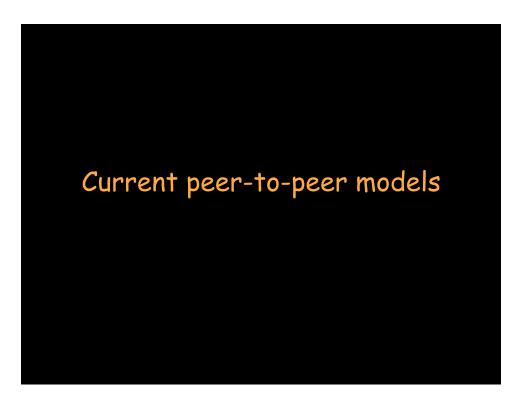
- Mail, ftp, rtalk, telnet served as triggers to the 1st generation of the Internet.
- Mosaic served as a trigger to the 2nd generation of the Internet
- Services like napster and gnutella served as triggers to Internet-based peer-to-peer computing

Clients are generally untapped

• Large business client layer might have:

2000 clients × 50 GB/client = 100 TB spare storage

2000 clients × 300 MHz/client × 9 ops/cycle = 5.4 trillion ops/second spare computing



Distributed file caching

• Akamai

- Buy thousands of servers and distribute them around the world
- Cache pages that don't change a lot
- Users annotate content on their web sites to point to akamai servers

Advantages

- Higher availability
- Better performance
 - Most references in the same network as yours.
- Rapid expansion is easy for an organization

Directory server mediated file sharing

- Users register files in a directory for sharing
- Search in the directory to find files to copy
- · Central directory, distributed contents

Napster

- Started by 19-year-old college dropout Shawn Fanning
- Stirred up legal battles with \$15B recording industry
- Before it was shut down:
 - 2.2M users/day, 28 TB data, 122 servers
 - $\boldsymbol{\cdot}$ Access to contents could be slow or unreliable

Peer-to-peer file sharing

- Users register files with network neighbors
- Search across the network to find files to copy
- Does not require a centralized directory server
- Use time-to-live to limit hop count

Gnutella

- Created by author of WinAMP
 - (AOL shut down the project)
- Anonymous: you don't know if the request you're getting is from the originator or the forwarder

KaZaA

- Supernodes: maintain partial uploaded directories and lists of other supernodes

Peer-to-peer file sharing

BitTorrent

To distribute a file:

- .torrent file: name, size, hash of each block, address of a tracker server.
- Start a seed node (seeder): initial copy of the full file

To get a file:

- Get a .torrent file
- Contact tracker tracker manages uploading & downloading of the archive:
 - get list of nodes with portions of the file
 - Tracker will also announce you
- Contact a random node for a list of block numbers
 - request a random block of the file

Example: The Pirate Bay

- Torrent tracker (indexing site)
- > 12 million peers
- About 50% seeders, 50% leechers
- Risk: indexing sites can be shut down

Cycle sharing

aka Grid Computing

aggregate autonomous computing resources dynamically based on availability, capability, performance, cost.

Example: Intel NetBatch

- >70% workstations idle, 50% servers idle
- Developed NetBatch c.1990
- Stopped buying mainframes in 1992
- 1990: 100 machines
- 2000: >10K machines across ~20 sites
- 2.7 million jobs/month

Cycle sharing

Example: SETI@home

- Scan radio telescope images
- Chunks of data sent to client in suspend mode (runs as screensaver)
- Data processed by clients when not in use and results returned to server

| | Total | Last 24 hours |
|--------------------------------------|-------------------------|---|
| Users | 5,405,452 | 647 |
| Results received | 1,843,726,685 | 1,311,140 |
| Total CPU time | 2,273,326.688 years | 877 years |
| Floating Point Operations | 6.77×10 ²¹ | 5.11x10 ¹⁸ (59.18 TeraFLOPs/sec) |
| Average CPU time per work unit | 10 hr 48 min 4.0 sec | 5 hr 51 min 34.4 sec |

SETI@home (4/28/8)

- Total hosts: 1,887,363
- Users: 811,755
- 252 countries

Cycle sharing

Example: distributed.net code breaking

RC5: 72 bits

total keys tested: 2.315×10¹⁹ (19.35 quintillion) total to search: 4.722×10²¹ overall rate: 1.36×10¹¹ keys per second % complete: 0.490% 1,973 days

RC5-64 challenge:

total keys tested: 15.27×10¹⁸ total to search: 18.45×10¹⁸ overall rate: 1.024×10¹¹ keys per second % complete: 82.77 1,726 days

Tons of distributed efforts

- Berkeley Open Infrastructure for Network Computing (BOINC): boinc.berkeley.edu
- Choose projects
- Download software
 - BOINC Manager coordinates projects on your PC
 - When to run: location, battery/AC power, in use, range of hours, max % CPU

http://boinc.netsoft-online.com/

Tons of distributed efforts

- SETI@home
- Climateprediction.net
- Einstein@home
- Predictor@home
- Rosetta@home
- BBC Climate Change Experiment
- LHC@home
- World Community Grid
- · SIMAP
- SZTAKI Desktop Grid
- PrimeGrid
- uFluids
- MalariaControl
- and lots more...

http://boinc.netsoft-online.com/

File servers

- Central servers
 - Point of congestion, single point of failure
- Alleviate somewhat with replication and client caching
 - E.g., Coda
 - Limited replication can lead to congestion
 - Separate set of machines to administer
- But ... user systems have LOTS of disk space
 - 350 GB is common on most systems
 - 500 GB 7200 RPM Samsung SpinPoint T Series: \$99
- Berkeley xFS serverless file system

Amazon S3 (Simple Storage Service)

Web services interface for storing & retrieving data

- Read, write, delete objects (1 byte 5 GB each)
- Unlimited number of objects
- REST & SOAP interfaces
- Download data via HTTP or BitTorrent

Fees

- \$0.15 per GB/month
- \$0.13 \$0.18 per GB transfer out
- \$0.01 per 1,000 PUT/LIST requests
- \$0.01 per 10,000 GET requests

Google File System

- Component failures are the norm
 - Thousands of storage machines
 - Some are not functional at any given time
- · Built from inexpensive commodity components
- Datasets of many terabytes with billions of objects
- GFS cluster
 - Multiple chunkservers
 - Data storage: fixed-size chunks
 - Chunks replicated on several systems (3 replicas)
 - One master
 - File system metadata
 - Mapping of files to chunks

Google File System usage needs

- Stores modest number of large files
 - Files are huge by traditional standards
 Multi-gigabyte common
 - Don't optimize for small files
- Workload:
 - Large streaming reads
 - Small random reads
 - Most files are modified by appending
 - Access is mostly read-only, sequential
- Support concurrent appends
- High sustained BW more important than latency
- Optimize FS API for application
 - E.g., atomic **append** operation

Google file system

- GFS cluster
 - Multiple chunkservers
 - Data storage: fixed-size chunks
 - Chunks replicated on several systems (3 replicas)
 - One master
 - File system metadata
 - Mapping of files to chunks
- Clients ask master to lookup file
 - Get (and cache) chunkserver/chunk ID for file offset
- Master replication
 - Periodic logs and replicas

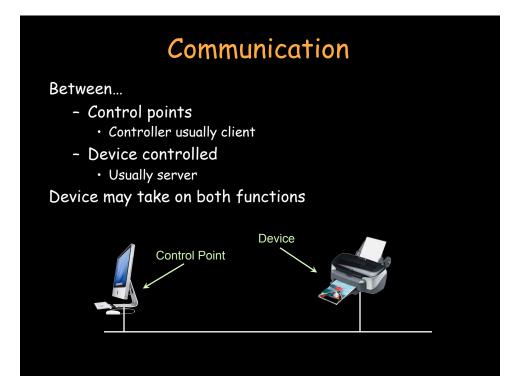


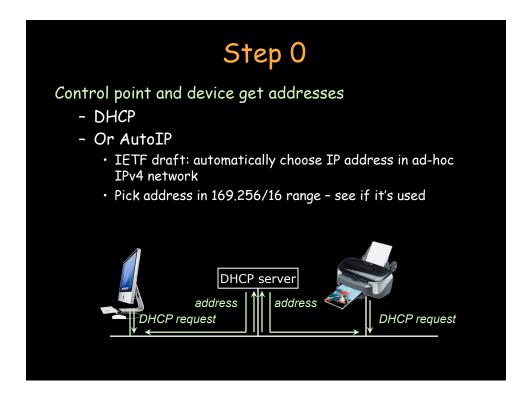
Ad-hoc networking and auto-discovery

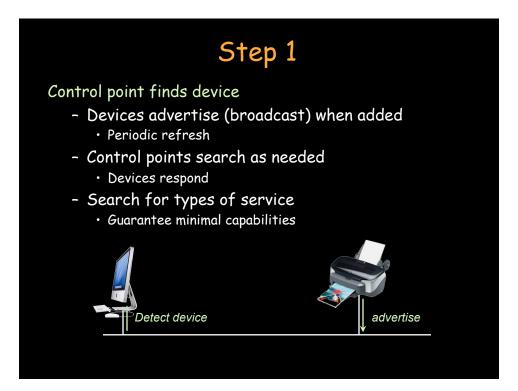
- Device/service discovery and control
 - Sun's JINI
 - Microsoft, Intel: UPnP
- Universal Plug and Play architecture
- http://www.upnp.org
- Networking
 - Unreliable: nodes added/removed unpredictably
 - Programs need to talk to programs (services)

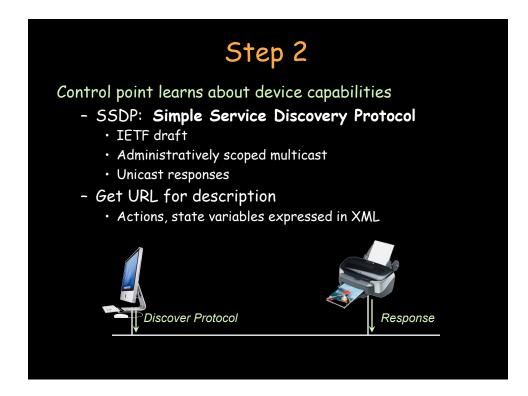
UPnP strategy

- $\boldsymbol{\cdot}$ Send data only over network
 - No executables
- Use standard protocols
- Leverage standards
 - HTTP, XML
- Basic IP network connectivity















Bonjour (Rendezvous)

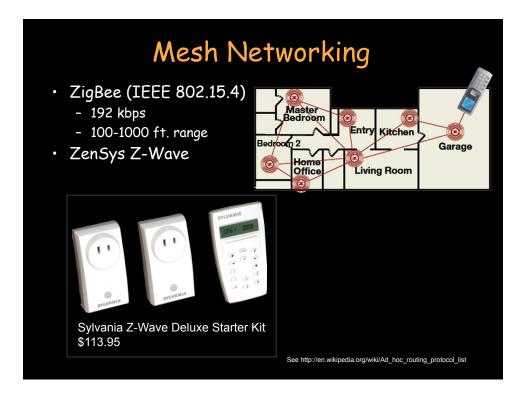
Apple et al.

- allocate addresses without a DHCP server
 Use 169.254/16 zeroconf range
- translate between names and IP addresses without a DNS server
 - Use IP multicast
- locate or advertise services without using a directory server
 - Use DNS
 - Structured Instance Names

Mesh Networking

Mobile Ad-hoc networks, Sensor networks,

- · Hop node-to-node until the destination is reached
 - Nodes can act as repeaters to nearby peers
 - Robust connectivity: find alternate routes
- Dynamic routing
 - Table-based: maintain fresh lists of destinations/routes
 - Reactive: find route on demand
 - Hierarchical
 - Geographical
 - Power-aware
 - Multicast



<section-header><list-item><list-item><list-item><list-item><list-item><list-item><list-item><list-item>

Issues

- Security
 - Protection of content
 - Protection against worms, viruses
 - Privacy
- Predictable connectivity
- Routing
- Fault tolerance
- Naming, resource discovery
- Standards, interoperability

