Peer-to-peer systems (P2P)

Expansion in client-server is by providing more hosts. Cost of
- deployment
- administration
- recovery
P2P: enables large scale sharing of data and resources by eliminating any requirement for separately-managed servers and associated infrastructure.
P2P $ \neq $ client-server

Characteristics of P2P

- All user contributes to system
- all nodes have same functionality & capability
- the correct operation of the system is not dependent on existence of central server
- they can be designed to give anonymity to user
- use algorithms for placement of data to ensure balanced workload

Three generations of P2P systems.

Identifying resources

Globally Unique IDentifiers (GUIDs)
- Secure hash from resource’s state
- hence resources are self-identifying
- guarded against tampering (use hash to authenticate)
- Down side: resources must be static
Hence P2P most suitable for files (music, books, …)
- use trusted server for dynamic resources

How does Napster work?

Napster Architecture

1 Napster Use

1 File

1 Napster Server

1 Index

1 Napster Architecture
Limitations

Napster demonstrated possibility of sharing resources in large scale
Load balancing and locality is via central server
Limitations:
- unified index (but replicated)
- consistency of replicates
- no guarantee of availability

not a serious issue for music !!

2nd generation P2P: partition and distribute the index

P2P middleware

to simplify the construction of services that are implemented across many hosts in a widely distributed network

Requirements:
- client locates and accesses any available service
- ability to add and remove new resources
- add and remove host
- provision of simple interface for the use of programmers

Non-functional requirements:

• Global Scalability
• Load balancing
• Optimisation of local interaction between neighbouring peers
• Accommodating highly dynamic host availability
• Security of data in an environment with heterogeneous trust
• Anonymity, deniability and resistance to censorship

Routing Overlay

Distributed algorithm for locating nodes & objects

ensures any node can access any object
routing each request through sequence of nodes
RO maintain knowledge of multiple copes of objects (ensures availability)
finds nearest “live” node

main task of Routing Overlay

1. client wishing to invoke an operation on an object submits a request including object’s GUID to RO, which routes the request to a node at which a replica of the object resides
2. Node wishing to make a new object available to a P2P service computes GUID for the object and announces it to RO, which then ensures that the object is reachable by all clients

main task of RO (continue)

3. When clients request the removal of objects from the service the routing overlay must make them unavailable
4. Nodes may join and leave the service. When a node joins the service, the routing overlay arranges for it to assume some of the responsibilities of other nodes. When a node leaves (either voluntarily or as a result of a system or network fault), its responsibilities are distributed amongst the other nodes.
GUID

GUID is created via hash functions from part or all states of the object
Hash $H(M) = h$:
1- easy to compute $h$
2- difficult to compute $M$ from $h$
3- high probability of uniqueness; if $H(M) = H(M')$, then $M = M'$
Notice the change in hash value
SHA1("Behzad")= 30d0b8a93dbf8769c344e6c2094cd719c88a6f943
SHA1("Behzad")=8d6de5dc5c5c0bedcd886748ed6a229f196e6

GUID (continue)

Two major method of using GUID:
- Distributed Hash Table (DHT)
- Distributed object location and routing (DOLR)

DHT

API:
- put(GUID, data) replicate data to all nodes responsible for GUID
- remove (GUID) delete all references to GUID and its data
- value = get(GUID) accessor for GUID
multiple replica of data is stored in nodes which have numerically close GUID
replication ensures high probability of access

DOLR

Object can be installed everywhere and RO provides mapping between GUID and address node containing replicate
API
- publish(GUID) computes GUID from obj.
- unpublish(GUID) makes obj. not accessible
- sendToObj(msg,GUID, [n]): invocation sent to an object to access it, by opening TCP connection … Delivery of message to n replicate

Human interaction

GUID is not human readable
Human readable indexing is in used indexes can be centralised or P2P
(P2P better to overcome weaknesses of Napster)
For music it is often a webpage:
Example: BitTorrent has an index of GUID and URL of a host that holds up-to-date copy of the network address of providers

Comparing IP routing and RO

Scale
- IPv4 $2^{32}$ addresses, IPv6 $2^{128}$ addresses.
  But, hierarchically structured
- RO addresses are $> 2^{128}$ and can be fully occupied
Load balancing
- IP’s load is determined by topology and traffic
- RO’s load can controlled; random location
Network dynamic
- IP tables updated asynchronously every x hour
- RO tables can be updated both async and sync in a few seconds

Fault tolerance
- IP: ensure router is available or replicate (costly)
- Resources are replicated in RO

Target identification
- IP’s addresses map into unique node
- RO’s address target nearest replica

Security and anonymity
- IP: addressing secure if all nodes trusted.
  Anonymity of owner not possible
- RO: security even in untrusted environment, limited anonymity

Summary
Differences between P2P and client-server
Three generations of P2P systems
requirements of P2P middleware
Role of GUID and hashing
Routing overlay and comparison with IP overlay

Further reading: chapter 10

Exercises:
10.1 Early file-sharing applications such as Napster were restricted in their scalability by the need to maintain a central index of resources and the hosts that hold them. What other solutions to the indexing problem can you identify?

10.2 The problem of maintaining indexes of available resources is application-dependent. Consider the suitability of each of your answers to Exercise 10.1 for (a) music and media file sharing, (b) long-term storage of archived material such as journal or newspaper content, (c) network storage of general-purpose read-write files.

Exercises:
10.3 What are the main guarantees that users eXpect conventional servers (e.g. web servers or file servers) to offer?

10.4 The guarantees offered by conventional servers may be violated as a result of: a) physical damage to the host; b) Errors or inconsistencies by system administrators or their managers; c) successful attacks on the security of the system software; d) hardware or software errors.

Give two examples of possible incidents for each type of violation. Which of them could be described as a breach of trust or a criminal act? Would they be breaches of trust if they occurred on a personal computer that was contributing some resources to a peer-to-peer service? Why is this relevant for peer-to-peer systems?