Gnutella was one of the first examples of a pure P2P system. It is not run by a single company, and no nodes act only as servers. Gnutella eliminates the directory server, for sharing files the user must connect to the Gnutella network, a loose federation of computers running Gnutella, to find the computer only has to know the address of one other Gnutella machine, e.g., machines published at well-known web sites. At first connection the computer receives hundreds of addresses of machines which may be used at subsequent occasions.

A Gnutella program tries to maintain 3 or 4 connections to other Gnutella machines at any one time. To find a file, send request with file name and current hop count to its neighbors.

- Neighbor has matching file: respond with the location of the file.
- Increment hop count;
- If hop count < maximum hop count, then propagate request to its neighbors.

BitTorrent is a P2P communications protocol for file sharing. The recipients of data also supply data to newer recipients, thus reducing the cost and burden on any given individual source, and reducing dependence upon the original distributor.

eDonkey is a P2P file sharing network used primarily to exchange audio and video files and computer software.

Files identified using compound MD4 hash checksums, which are a function of the bit content of the file.

Gossip-based Approach
Propagate information in the same way as epidemic diseases spread:
- time t₀: suppose I know something new
- time t₁: I pick a friend and tell him; now 2 people know
- time t₂: we each pick a friend and tell them; now 4 people know
- time t₃: .......

Information spreads at exponential rate.
- due to re-infection information spreads at approx. 2ⁿ after k rounds.

Combination of push and pull works best.
- Algorithm is quite robust and scalable
- Information travels on exponentially many paths.
- Difficult to slow down.
- The load of the participating nodes is independent of the system size.
- Information spreads in log(system size) time.
- Network load is linear in system size.

**BitTorrent** is a P2P communications protocol for file sharing:
- the recipients of data also supply data to newer recipients,
- reducing the cost and burden on any given individual source,
- reducing dependence upon the original distributor.

**eDonkey** is a P2P file sharing network
- used primarily to exchange audio and video files and computer software.
- Files identified using compound MD4 hash checksums, which are a function of the bit content of the file.
- Overnet as successor of eDonkey protocol.

**Gossip-based Approach**

---

All processes play a similar role:
- Interacting cooperatively as peers to perform a distributed computation.

There is no distinction between clients and servers.
- Clients talk directly to one another.

**Client-Server vs. Peer-to-Peer**
- Napster
- Gnutella

**Other System Examples**

---

**Information Sharing**
- Message exchange
- Naming entities
- Bidirectional communication
- Producer-consumer interaction
- Client-server model
- Peer-to-peer model

**Group model**
- Taxonomy of communication
  - Message serialization

**Levels of Abstraction**
Paradigms for distributed applications

Information Sharing
Message exchange
Naming entities
Bidirectional communication
Producer-consumer interaction
Client-server model
Peer-to-peer model
Group model

Taxonomy of communication
Message serialization
Levels of Abstraction

Levels of Abstraction

If several senders are involved, the following message ordering schemes may be applied:

1. no serialization.
2. loosely-synchronous.
   There is a loosely synchronized global time which provides a consistent time ordering.
3. virtually-synchronous.
   The message order is determined by causal interdependencies among the messages. For example, a message N has been sent after another message M has been received, i.e. N is potentially dependent on M.
4. totally ordered.
   By token: before a sender can send a message, it must request the send token.
   A selected component (the coordinator) determines the order of message delivery for all recipients.

One sender

There are the following ordering schemes:

- according to the message arrival on the recipient's side; different receivers can have different message arrival sequences.
- according to message sequence number generated by the sender; this approach is sender-dominated.

Several senders

receiver creates a serialization according to its own criteria.
The client-server model implements a sort of **handshaking principle**, i.e., a client invokes a server operation, suspends operation (in most of the implementations), and resumes work once the server has fulfilled the requested service.

**Terms and definitions**

**Concepts for client-server applications**

**Processing of service requests**

**File service**

**Time service**

**Definition:** A **time service** provides a synchronized system-wide time for all nodes in the network.

**Name service**

**LDAP - Lightweight Directory Access Protocol**

**Failure tolerant services**

---

**Client**

**Definition:** A **client** is a process (say, an application) that runs on a client machine and that typically initiates requests for service operations.

**Potential clients are a priori unknown.**

**Service**

**Definition:** A **service** is a piece of software that provides a well-defined set of services. This piece of software may run on one or multiple (server) machines.

**Server**

**Definition:** A **server** is a subsystem that provides a particular service to a set of a priori unknown clients. A server executes a (piece of) service software on a particular server machine. Obviously, a single server machine can host multiple server subsystems.

**A server provides a set of operations (procedures).**
Terms and definitions

Definitions

Client-server interfaces

Multitier architectures

<table>
<thead>
<tr>
<th>Client</th>
<th>Presentation execution</th>
<th>Presentation execution</th>
<th>Presentation execution (with local database)</th>
<th>Presentation execution database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Database</td>
<td>Database</td>
<td>Execution (with local database)</td>
<td>Database</td>
</tr>
</tbody>
</table>
| Case 1     | Case 2                  | Case 2                 | Case 3                                      | Case 3                          | Case 4

Different cases

Case 1: remote data storage, access, for example, via Sun NFS.
Case 2: remote presentation (for example X window system).
Case 3: distributed application

cooperative processing among the individual components of an application.

Case 4: distributed data storage

The information is distributed between client and server; information replication is possible.

Single dedicated server process

Cloning of new server processes

Parallel request processing through threads

This is a variant of the second approach.

Shared address space, i.e., the approach allows shared utilization of variables.
Single dedicated server process

A single dedicated server process is in charge of processing requests for service operations.

- no parallel processing of requests, which results in the following disadvantages:
  - approach may be time consuming.
  - no interruption of the processing of the current request when a higher prioritized request appears in the queue.
  - server becomes bottleneck.

Cloning of new server processes

Every incoming request is handled by a new server process.

- Cloning of new server processes is expensive;
- Synchronization of access to shared persistent data;
- Parallel processing of several applications is possible;

Processing of service requests

Clients and servers have different life spans; servers manage these requests in a queue.

- Single dedicated server process
- Cloning of new server processes
- Parallel request processing through threads
  - This is a variant of the second approach.
  - Shared address space, i.e., the approach allows shared utilization of variables;

Definition: A file service [Svobodova 1984] provides (remote) centralized data storage facilities to clients distributed among a network.

- server deals with bulk data storage, high performance computation, collecting/managing large amounts of data.
- client deals with "attractive" display, quick interaction times.
- use of caching to speed up response time. Use of cache "hints" to facilitate cache management speed up system when hint is correct.
- mechanism to detect wrong hint and seek up-to-date information.

Distinction between stateless and stateful

- Stateless server
- Stateful server
Stateless server do not manage any state information about their clients; the client must supply all necessary parameters to process the request.

Stateful server subsystems manage state information about their clients.

Server tracks its clients and takes actions to keep their cached states up-to-date. Client can trust its cached data → cache is owned by the server.

As a consequence, programming at the client site becomes less complex.

Stateful transactional server architecture: after recovery of server crash an abort message is sent to client.

A crashed server can be restarted without dealing with state reinstalments.