

The Anthill Project

Part I: Introduction to Peer-to-Peer

Alberto Montresor

University of Bologna
Department of Computer Science

Prologue

"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 horde instead of a fixed Central Command Post? Yes."

David Gelernter, *The Second Coming: A Manifesto*

© 2001 Alberto Montresor

2

Introduction

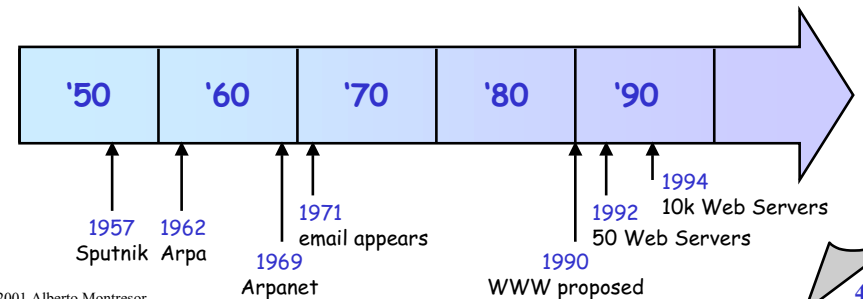
- Recently, the *peer-to-peer* (P2P) paradigm for building distributed applications has gained attention from both industry and the media
- Peer-to-peer: classical definition
 - A P2P system is composed of a distributed collection of *peer* nodes
 - Each node is both a server and a client:
 - may provide services to other peers
 - may consume services from other peers
- Completely different from the client-server model, where:
 - Few specialized servers provide services to a large number of clients

© 2001 Alberto Montresor

3

P2P History: 1969 - 1995

- 1969 – 1995: the origins
 - In the beginnings, all nodes in Arpanet/Internet were peers
 - Every node was capable to
 - perform routing (locate machines)
 - accept ftp connections (file sharing)
 - accept telnet connections (distributed computation)



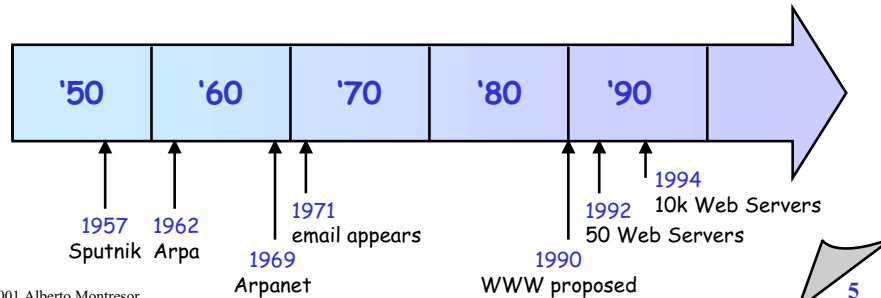
© 2001 Alberto Montresor

4

P2P History: 1995 - 1999

1995 – 1999: the Internet explosion

- The original “state of grace” was lost
- Current Internet is organized hierarchically (client/server)
 - Relatively few servers provide services
 - Client machines are second-class Internet citizens (cut off from the DNS system, dynamic address)



5

P2P History: 1999 - today

1999 – today: the advent of Napster

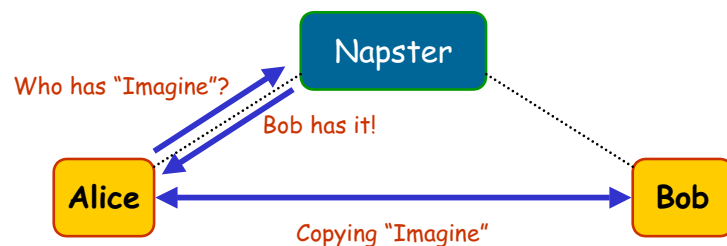
- Jan 1999: the first version of *Napster* is released by Shawn Fanning, student at Northeastern University
- Jul 1999: Napster, Inc. founded
- **In short time, Napster gains an enormous success, enabling millions of end-users to establish a file-sharing network for the exchange of music files**
 - Jan 2000: Napster unique users > 1.000.000
 - Nov 2000: Napster unique users > 23.000.000
 - Feb 2001: Napster unique users > 50.000.000

6

Napster

The Napster architecture:

- Napster works by operating a central server which directs traffic between individual registered users
- Each time a user submits a request for a song, the central server creates a list of users who are currently connected to Napster whose collections include the specified song



7

Napster is not alone

Following the success of Napster, other file-sharing systems started to appear, such as:

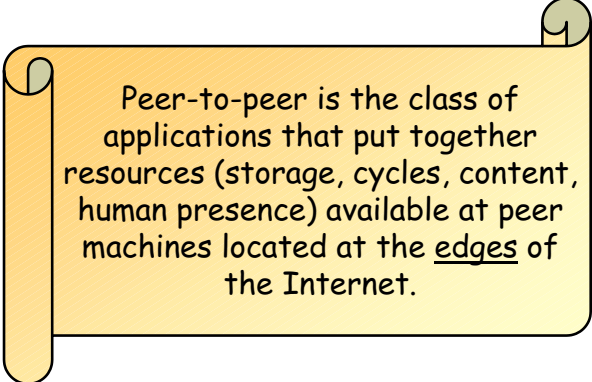
- Gnutella gnutella.wego.com
- Freenet freenet.sourceforge.net
- Moreover, other applications appeared, capable to establish communities comprising millions of cooperating nodes:
 - Distributed Computing
 - Seti @ Home setiathome.ssl.berkeley.edu
 - Distributed.net www.distributed.net
 - Messaging and collaborative tools
 - Groove www.groove.net

8

P2P History: 1999 - today

- **1999 - today: the new peer-to-peer revolution**
 - Since 1999, the IT community started to search a label to define the new distributed model suggested by Napster and these other applications
 - By July 2000, this label was found: *peer-to-peer*
 - The label, however, didn't clarified things
 - Following the classical definition of peer-to-peer, Napster is not peer-to-peer (centralized server)
 - But Napster is what originated the discussion!
- **A new definition for peer-to-peer was needed**

Alternative Definition of Peer-to-Peer



Peer-to-peer is the class of applications that put together resources (storage, cycles, content, human presence) available at peer machines located at the edges of the Internet.

P2P and Piracy

- **Most of the material exchanged through P2P file-sharing systems is copyrighted**
- **Some of the P2P projects have *anonymity* among their goals:**
 - Freenet
 - Freehaven
- **This has resulted in the following equation:**
P2P = subversion of intellectual property

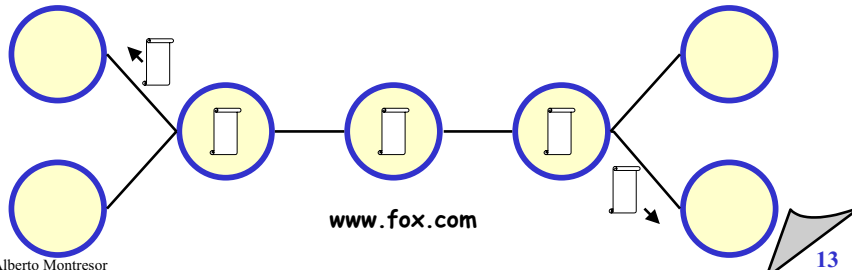
Why P2P?

- **Despite its poor reputation, P2P is extremely interesting from a technical point of view:**
 - Its completely decentralized model enables the development of applications with
 - *high-availability*
 - *fault-tolerance*
 - *scalability*characteristics previously unseen in Internet
 - It exploits what has been defined the “dark matter” of Internet
 - Moreover, P2P is not limited to file-sharing, but it can be applied to distributed computing and collaboration tools

P2P Examples

Example: avoiding the “Slashdot effect”

- “The more popular a piece of information is, the less available it becomes”
- On the contrary, the number of replicas of a document in Freenet increases proportionally to its popularity
- Can be applied to the distribution of movie trailers



© 2001 Alberto Montresor

13

P2P Examples

Example: CareScience Care Data Exchange

- File-sharing network that enables the exchange of information between healthcare organizations:
 - clinical results
 - patient demographics
 - medical records
- Aimed at giving medical personnel easy access to crucial health-related data
- In the United States, 50.000 deaths per year are caused by the lack of information

© 2001 Alberto Montresor

14

P2P Examples

Example: the Flu Vaccine Project

- The development of more effective influenza vaccines requires the simulation of a large number of test cases
- Healthcare organizations do not possess the required computing power
- Popular Power is a platform for distributed computing
 - By downloading the app, you can contribute cycles to a distributed computing project.
 - One of this projects is the Flu Vaccine Project

© 2001 Alberto Montresor

15

P2P Services

Areas of applicability of P2P

- sharing of content
 - distributed web servers, distributed media repository
- sharing of storage
 - distributed file system, distributed search engine
- sharing of CPU time
 - parallel computing
- sharing of human presence
 - the “P” in P2P is “Person”

© 2001 Alberto Montresor

16

P2P Definitions

- **Content/Storage Sharing Applications:**
 - Authors: entities which create documents
 - Publishers: the entities that place documents in the system
 - Readers: the entities that retrieve documents from the system
 - Servers/Clients/Servents/Nodes: the peer entities which actually store data
- **CPU Sharing Applications:**
 - Requester: the entity that requests the execution of a computation
 - Clients/Servents/Nodes: the entities that perform the computation

Content/Storage Sharing Applications Goals

- **Efficiency:**
 - Reader search, reader download, publisher upload
- **Integrity:**
 - An authorized reader must receive the same document as originally placed into the system by the publisher
- **Persistence:**
 - The system must guarantee to store a document for a given amount of time (possibly forever)
- **Resistance to denial-of-service:**
 - Attacker detection and isolation

Content/Storage Sharing Applications Goals

- **Fault-tolerance**
 - No central point of failure
 - Failure localization
- **Anonymity and Partial Anonymity:**
 - Author Anonymity
 - Authors cannot be linked to documents
 - Publisher Anonymity
 - Publishers cannot be linked to documents
 - Server Anonymity
 - Servers cannot be linked to documents
 - Query Anonymity
 - Readers cannot be linked to queries

CPU Sharing Applications Goals

- **Efficiency**
 - Maximization of computations carried out
- **Security**
 - Avoiding malign code to be executed on peer machines
- **Fault-tolerance**
 - Completing computations in spite of failures
- **Resistance to denial-of-service**
 - Attacker detection and isolation

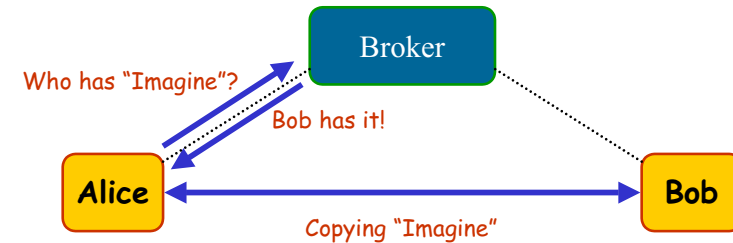
Current Peer-to-Peer Models

▪ Broker-Mediated Content Sharing

- Global index held by central authority (broker)
 - single point of failure
- Users
 - register content with the broker
 - use broker to find content to copy
- Also called “hybrid peer-to-peer”
- Contact:
 - For searches: mediated by the broker
 - For download: direct between requestors and providers
- e.g. Napster

Current Peer-to-Peer Models

▪ Broker-Mediated Content Sharing



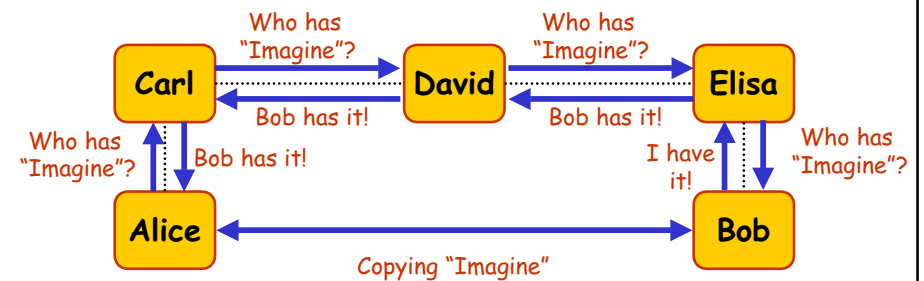
Current Peer-to-Peer Models

▪ (Pure) Peer-to-Peer Content Sharing

- No global index – local knowledge only
 - approximate answers
- Users
 - register content with network neighbors
 - search across the network to find content to copy
- Contact:
 - For searches: mediated by chain of intermediaries
 - For download: directed or through intermediaries
- e.g. Gnutella and Freenet

Current Peer-to-Peer Models

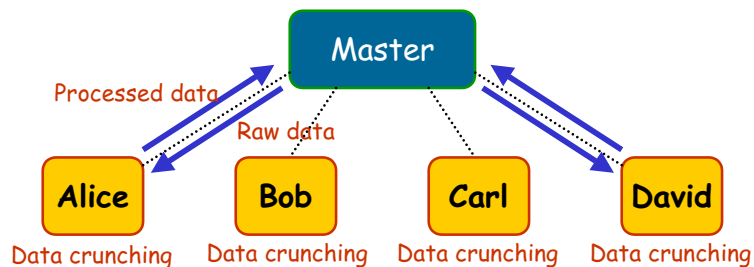
▪ (Pure) Peer-to-Peer Content Sharing



Current Peer-to-Peer Models

Master-Slave Cycle Sharing

- Master send chunks of data are sent to clients
- Data is processed by clients when client is not in use and returned to the master
- Adopted by: SETI@ home, distributed.net, etc.



Pure Peer-to-Peer: Key Questions

- **Does it work?**
 - can we find the data?
 - query success rates
 - length of query paths
- **Does it scale?**
 - logarithmic / linear / polynomial
- **Is it robust?**
 - participants are unreliable
 - different failure modes possible

Peer-to-Peer Systems

Pure Peer-to-Peer Content Sharing Systems

- Gnutella
- Freenet

Master-Slave Cycle Sharing Systems

- Seti@Home
- distributed.net

History of Gnutella

Definition:

Gnutella is a *protocol* for distributed search

History:

- Gnutella was started by Nullsoft, a division of AOL (3/2000)
- When AOL realized the Gnutella potential use for copyright infringement, it closed the project
- The original Nullsoft protocol was reverse-engineered
- Currently, Gnutella is an open-source project
 - The development of the protocol is open-source
 - Many "*servent*" implementation exist
 - Limeware
 - Bearshare

Gnutella

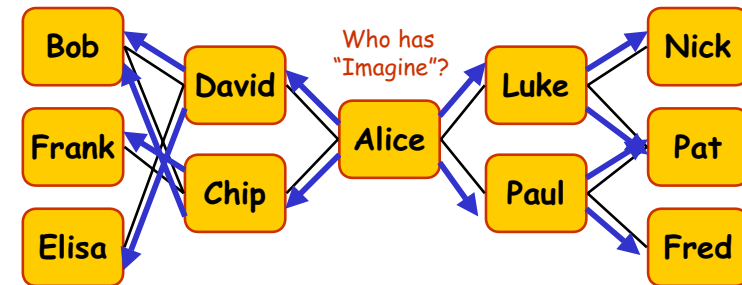
- **The Gnutella protocol consists of:**
 - a set of message types representing the ways in which servents communicate over the network
 - a set of rules governing the inter-servent exchange of messages
- **How to connect to a Gnutella network**
 - A Gnutella servent connects to the network by establishing a connection with another servent currently on the network
 - The acquisition of another servent's address is not part of the protocol definition
 - "Out-of-band" methods

29

© 2001 Alberto Montresor

Gnutella Basics

- **Gnutella is based on broadcasting:**
 - Each message received by a servent A is broadcast to every other servent connected to A
 - This poses serious problems of scalability
 - Mechanisms are used to reduce the number of messages



30

© 2001 Alberto Montresor

Gnutella Messages

- **Each message is composed of:**
 - A message type field
 - PING, PONG
 - QUERY, QUERYHIT
 - PUSH
 - A Time-To-Live (TTL) Field
 - The number of times the message will be forwarded by servents before it is removed from the network
 - Decrement at each hop, lowered if needed
 - A 16-byte ID field uniquely identifying the message on the network
 - Randomly generated
 - Not related to the address of the requester (anonymity)

31

© 2001 Alberto Montresor

Gnutella Message Types

- **PING**
 - essentially, an "are you there" message directed to a host
 - a servent receiving a PING is expected to respond with a PONG message
 - no recommendation as to the frequency of PING messages
- **PONG**
 - the response to a PING
 - has the same ID of the corresponding PING message
 - contains:
 - address of connected Gnutella servent
 - total size and total number of files shared by this servent

32

© 2001 Alberto Montresor

Gnutella Message Types

▪ QUERY

- The primary mechanism for searching the distributed network
- Contains:
 - a null-terminated query string
 - a field specifying the minimum download speed
- A servent is expected to respond with a QUERYHIT message if a match is found against its local data set

Gnutella Message Types

▪ QUERYHIT

- the response to a query
- has the same ID of the corresponding QUERY message
- contains enough information to acquire the data matching the corresponding query
 - IP Address
 - Port number for incoming connections
 - List of file names

▪ PUSH

- A mechanism that allows servents behind firewalls to serve files

Gnutella Message Forwarding Rules

- **A servent receiving a message must forward it to each of the other servents to which it is connected**
- **Exceptions:**
 - TTL is zero
 - PONG messages must be routed along the same path of the incoming PING
 - QUERYHIT messages must be routed along the same path of incoming QUERY messages
 - Messages with duplicated ID should be discarded
- **A cache of message IDs, along with sender, is needed**

Interpretations of Query Strings

▪ Gnutella is a simple protocol

- Defines only how a query string is passed from one site to another
- Uses Http to effectively download the data

▪ How queries are interpreted?

- Different implementations may interpret a string in different ways
 - Ex: by searching the string in set of filenames
 - Ex: by running grep on a set of files
- Flexibility:
 - each site may contribute to a distributed search in a complex way
 - different gnutella networks may solve distinct problems

Problems of Gnutella

Limited horizon

- The number of reachable nodes is limited by *TTL* and the number *N* of concurrent connections
 - Example: tree network, $TTL=5$, $N=4$, nodes = $4^5 = 1024$
- Solution: increase *TTL* and *N*!

Scalability

- The number of messages exchanged increases exponentially with the increase of *TTL* and *N*
 - Example: tree network, $TTL=5$, $N=4$, messages = $4^5 = 1024$
- Solution: decrease *TTL* and *N*!
- Solution: caching policies based on query popularity
 - over 50% cache hits with 5-minute caching

Problems of Gnutella

Denial of Service attacks

- Flooding the system with requests
 - Strange traffic observed in Gnutella
- Solution: keep statistics about frequency of requests and close connections with offending nodes

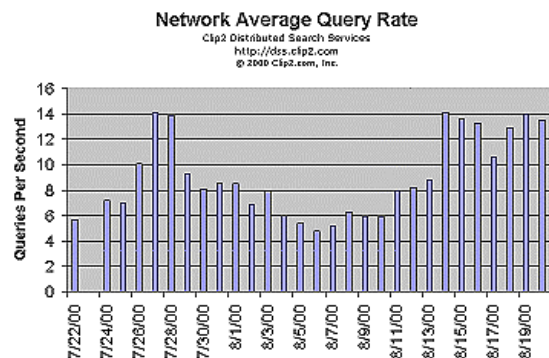
Privacy attacks

- A site advertised file names that appeared to offer child pornography
- It logged the IP address and domain name of every download request (included in HTTP)
- Solutions: none at present

Scalability

Important dates:

- July 27, 2000: Napster Flood
- August 14, 2000: Gnutella Hits the first scalability barrier



Gnutella Hits the Wall

August 2000:

- average Gnutella network traffic began to regularly exceed 10 query/sec per link
- user reported responses to their searches were fewer in number and slower to arrive than in the past

The first scalability barrier:

- For modem users: 10 query/sec
- Modem users ceased to be able to participate as *peers*

Other scalability barriers exist:

- DSL

Current Status

- **Current Gnutella population:**
 - 10,000-30,000 user per day (estimated)
- **How this population is organized?**
 - Fragmented in multiple dynamically changing responsive and unresponsive segments
 - Finding and remaining connected to a responsive segment is a matter of chance
- **gnutellahosts.com list service**
 - provides addresses included in the largest identifiable responsive segment
 - servents connect to gnutellahosts, obtain addresses, disconnect

© 2001 Alberto Montresor

41

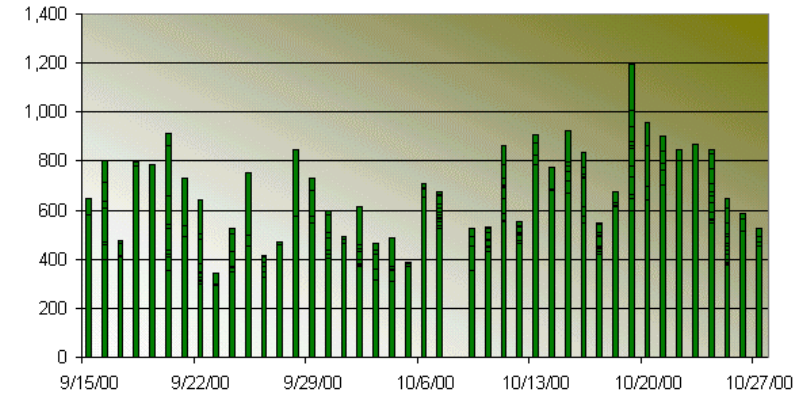
Number of Hosts in Largest Responsive Segments

Number of Hosts in Largest Responsive Network Segment

Clip2 Distributed Search Solutions

<http://dss.clip2.com>

© 2000 Clip2.com, Inc.



© 2001 Alberto Montresor

42

Gnutella is not Stable

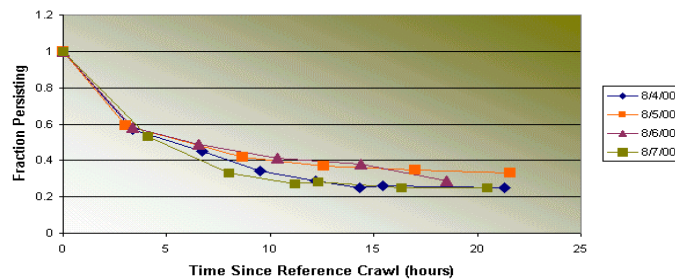
- **Note: the "Largest Responsive Segment" is a moving target**
 - The structure of a Gnutella network is in a continuous state of flux

Gnutella Network Host Persistence

Clip2 Distributed Search Services

<http://dss.clip2.com>

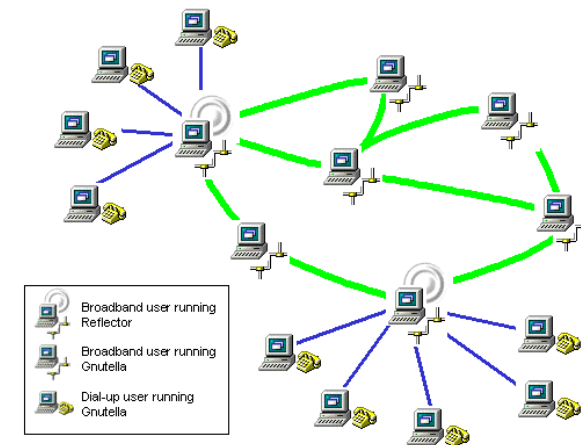
© 2000 Clip2.com, Inc.



© 2001 Alberto Montresor

43

Reflectors



Clip2 Distributed Search Solutions

<http://dss.clip2.com>

© 2000 Clip2.com, Inc.

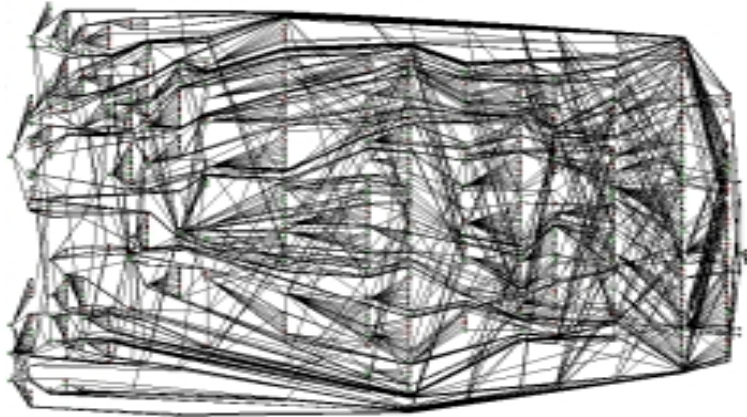
© 2001 Alberto Montresor

44

Gnutella Map

Partial Map of Gnutella Network - 7/27/00

Clip2 Distributed Search Services
<http://das.clip2.com>
©2000 Clip2.com, Inc.



© 2001 Alberto Montresor

45

Gnutella: Conclusions

- **Gnutella pros**
 - Simple architecture, easily implementable, could be profitably used for small groups
- **Gnutella cons**
 - Not scalable
- **Gnutella “future work”**
 - Analysis of its behavior
 - Implementation of caching policies
 - Subdivision in multiple Gnutella networks for scalability
 - Hierarchy?

© 2001 Alberto Montresor

46

Freenet

- **Freenet:**
 - An adaptive peer-to-peer application that permits the publication, replication and retrieval of data while protecting the anonymity of users

- **Philosophy:**

I worry about my child and the Internet all the time, even though she's too young to have logged on yet. Here's what I worry about. I worry that 10 or 15 years from now, she will come to me and say 'Daddy, where were you when they took freedom of the press away from the Internet?'"

Mike Godwin, Electronic Frontier Foundation

© 2001 Alberto Montresor

47

Freenet Goals

- **Socio-political goals of Freenet**
 - Publisher anonymity
 - Reader anonymity
 - Server anonymity
 - Resistance to attempts by attackers to deny access to data
 - Denial-of-service attacks
 - Removal attacks
- **Technical goals of Freenet**
 - Decentralization of all network functions
 - Data replication/distribution without human intervention
 - Efficient dynamic storage and routing of information
 - High availability for popular data

© 2001 Alberto Montresor

48

Freenet Design

- **Freenet is a P2P network of nodes storing data files**
- **Data files:**
 - are named by location-independent *keys*
- **Freenet nodes:**
 - maintain a *datastore* and make it available to the network:
 - for reading
 - for writing
 - maintain a *dynamic routing table* containing
 - addresses of other hosts
 - keys they are thought to hold
 - query one another to store and retrieve data files

Freenet Design

- **Freenet differs from Napster and Gnutella**
 - it is not based on a central server
 - it is not based on broadcasts
- **Freenet is adaptive**
 - responds adaptively to usage patterns
 - transparently moves, replicates, deletes files as needed
- **Freenet and persistency:**
 - it is not intended to guarantee permanent file storage
 - but most files may persist indefinitely, if enough nodes join

Freenet Messages

- **All messages contain:**
 - A randomly-generated 64 bit *Transaction ID*
 - Unique “with high probability”
 - A *TTL* field (Hop-to-Live)
 - A *Depth* field (number of hops performed so far)
- **Messages are forwarded from node to node**
 - TTL *decremented* at each hop
 - Message not discarded when TTL reaches 1
 - Randomly forwarded for other steps (for anonymity)
 - Depth *incremented* at each hop
 - Used in reply messages to set the TTL
 - Does not start at 0 (for anonymity)

Freenet Messages: Handshake

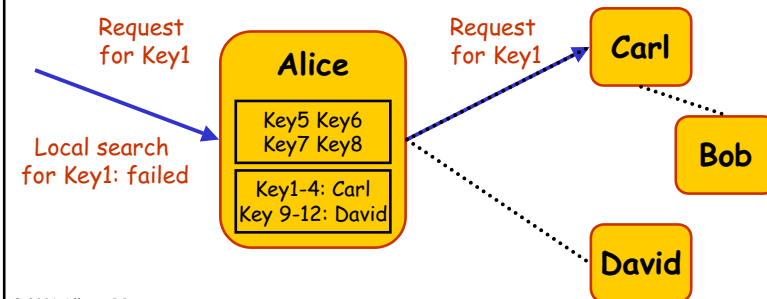
- **Request.Handshake message:**
 - Sent by a node to announce its existence
 - Specifies the desired return address of the sending node
- **Reply.Handshake message:**
 - Contains the protocol version understood
- **Handshakes:**
 - Are remembered for few hours
 - May be forwarded from node to node

Freenet Messages: Request

- **Request.Data message:**
 - Contains a search key
- **Send.Data message:**
 - Sent when a nodes is found containing the data searched
 - Contains
 - the ID of the request
 - the data requested
 - the address of the node hosting the data (may be faked)
- **Reply.NotFound message:**
 - Sent when a node cannot satisfy the request (see later)
 - Contains the ID of the request

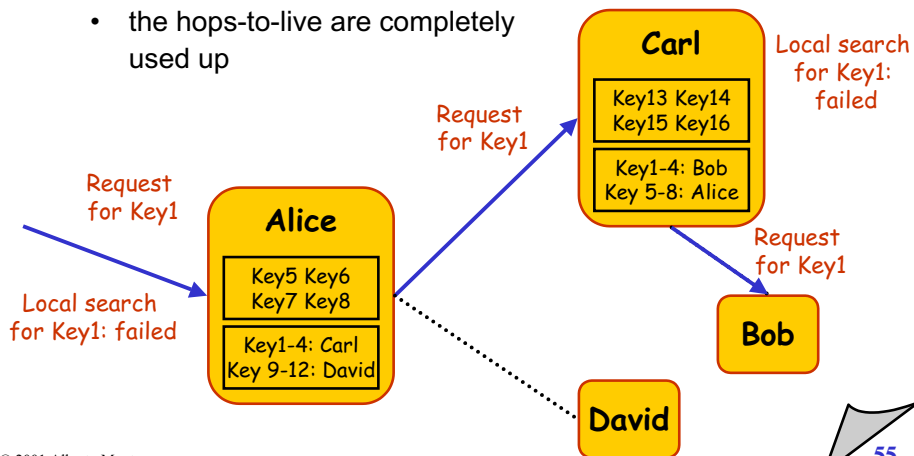
Freenet Algorithm: Request

- **When a node receive a request for a key:**
 - Attempts to retrieve the file locally, if possible
 - Otherwise, forwards the request to another node
 - Which node? Local decision in IP-style
 - Decision depends on the key



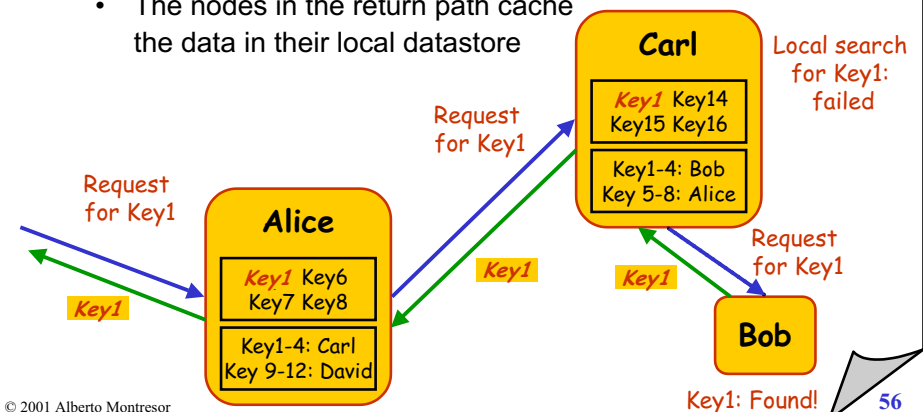
Freenet Algorithm: Request

- **Requests are passed from node to node until**
 - the requested key is found
 - the hops-to-live are completely used up



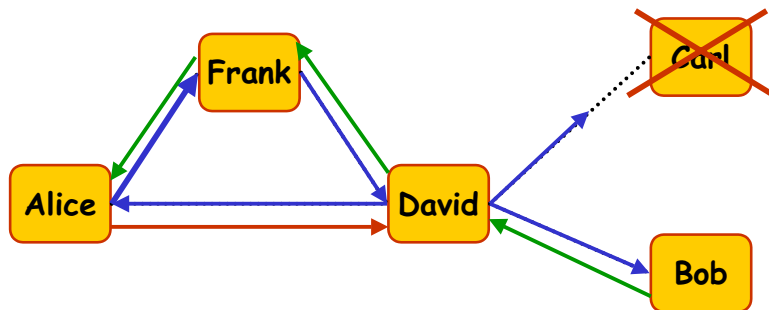
Freenet Algorithm: Request

- **When a request is successful (found in the local datastore)**
 - The data are returned to the requester along the same path of the incoming request
 - The nodes in the return path cache the data in their local datastore



Freenet Algorithm: Request

- A request cannot be forwarded to the selected node
 - If the node is down
 - If the node has been already visited by this request
- In this case, another node among the neighbors is chosen

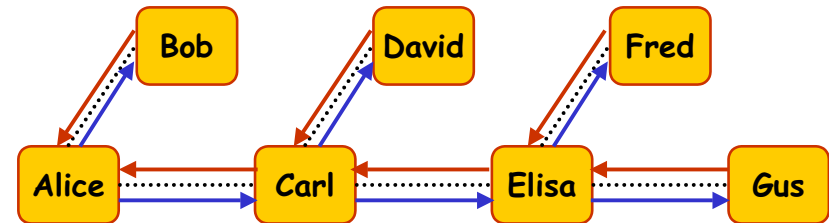


57

© 2001 Alberto Montresor

Freenet Algorithm: Request

- If a node runs out of candidates to try:
 - it reports failure back to its upstream neighbor
 - this can try other nodes, or report failure back
- If hops-to-live are exceeded:
 - a failure is reported back to the original requestor

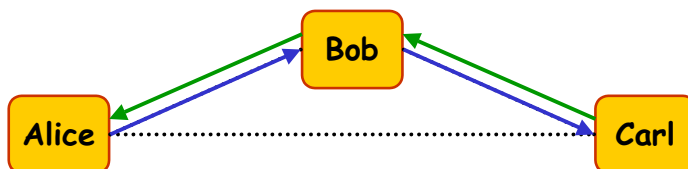


58

© 2001 Alberto Montresor

Freenet Algorithm: Request

- Network structure actively evolves over time:
 - Send.Data messages contain the address of the replying node
 - When receiving a Send.Data messages, the replying node is added to the routing table
 - Note: this field may be changed during the return trip (anonymity)



59

© 2001 Alberto Montresor

Freenet Algorithm: Insert

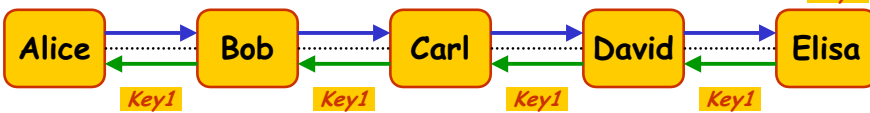
- Inserts follow a parallel strategy to requests:
 - *Request.Insert* message:
 - Sent to verify whether a key is already used in the network
 - Contains the *proposed key*
 - Responses:
 - Send.Data: if the data has been actually found
 - Reply.NotFound: if references to the data (not the actual data) have been found in routing tables
 - Request.Continue: data not found, TTL not run out, *insert OK*
 - Reply.Insert: data not found, TTL run out, *insert OK*
 - Note: All these responses contains the same ID as the original Request.Insert message
 - Send.Insert message:
 - Similar to a Send.Data message

60

© 2001 Alberto Montresor

Freenet Algorithm: Insert

Request.Insert
Key1, TTL = 5

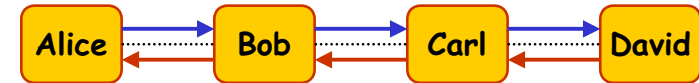


Request.Insert
Key1, TTL = 3

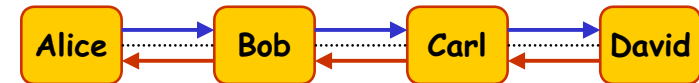


Freenet Algorithm: Insert

Request.Insert
Key1, TTL = 5



Request.Insert
Key1, TTL = 3



Freenet Algorithm: Data Management

▪ Databases in Freenet nodes:

- store file content and key
- have configurable size
- use LRU policy
 - ordered by request time and/or insert time
- when a new file arrive:
 - if enough space is available, store the new file
 - else, delete the last recently used file and store the new file

▪ Routing tables

- use LRU policy in the same way as datastore
- routing entries will be retained longer since they are smaller

Freenet Algorithm: Data Management

▪ Databases are not caches:

- there is no “permanent” copy which is being replicated in a cache
- when all nodes independently decide to drop a file, the file is no longer available in the network
- Freenet goal is not persistent storage (as Freehaven)

▪ Stored files are encrypted:

- node operators cannot know the content of their databases
- the motivations:
 - not for securing files (requestor can read them)
 - node operator can deny any knowledge about its database

Keys

- **Files in Freenet are identified by binary file keys:**
 - 160 bit keys
 - Obtained through SHA-1 hash algorithm
- **Three kind of keys:**
 - Keyword-signed key (KSK) (the original)
 - Signed-subspace key (SSK)
 - Content-hash key (CHK)
- **Keys and routing tables:**
 - A node cannot maintain a pair (key, next hop) for each document in the system
 - Lexicographic closeness of keys is used

Keyword-Signed Key (KSK)

- **Based on a short descriptive text string**
 - Ex.: "*text/philosophy/sun-tzu/art-of-war*"
- **This string is used:**
 - to deterministically generate a private/public key pair
 - public half is hashed to obtain the key
 - private half is used to sign the file
 - Minimal integrity check
 - as key to encrypt the file
 - provide deniability
 - To advertise the file, advertise the descriptive string
- **Problems:**
 - Flat namespace (conflicts)
 - Key squatting (insert junk files under popular descriptions)

Signed-Subspace Key (SSH)

- **Enable user to create personal namespaces**
 - by randomly generate a private/public key pair
 - by selecting short descriptive strings for files
- **These key pair and string are used:**
 - to generate the file key
 - file key = hash(public half + descriptive string)
 - the private half is used to sign the file
 - integrity check
 - the file is encrypted using the descriptive string

Signed-Subspace Key (SSH)

- **To advertise files:**
 - advertise the descriptive string plus the public half
 - advertise the key itself
- **To publish files:**
 - The private key is needed
 - No conflicts are possible (same string but different owners)
- **Owners may manage their own namespaces**
 - Use hierarchical structure
 - Directory-like files containing hypertext pointers
 - to other files
 - to other directories

Content-Hash Key (CHK)

- Useful for implementing updating and splitting
- CHK are obtained:
 - by hashing the contents of the corresponding file
 - files are also encrypted by a randomly-generated encryption key
- To advertise files:
 - advertise the file key and the decryption key
- File splitting:
 - Using content-hash keys for each part
 - Indirect file to point to the individual parts
 - Load balancing: different parts are stored in different nodes

© 2001 Alberto Montresor

69

Freenet Routing

- Assumption:
The quality of routing should improve over time
- Reasons:
 - Nodes should specialize in locating sets of similar keys
 - If a node is listed in routing tables for a particular key, it will tend to receive request for keys similar to that key
 - Nodes should specialize in storing clusters of files having similar keys
 - Forwarding a result will result in gaining a copy of the file itself
- These two effect improve the efficiency of future request:
self-reinforcing cycle

© 2001 Alberto Montresor

70

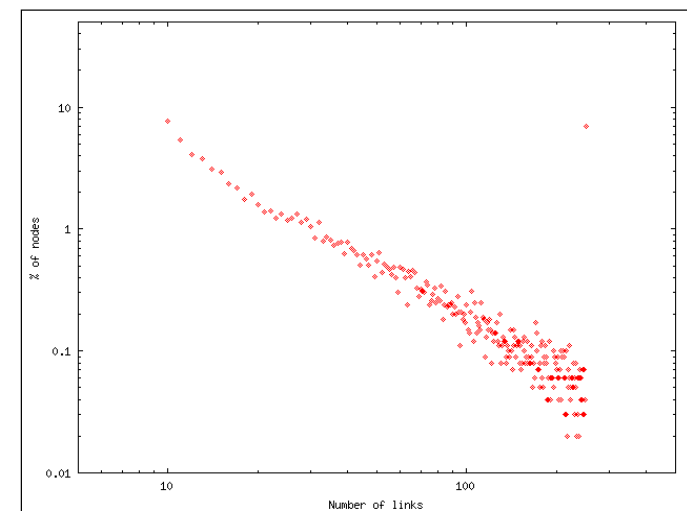
Freenet as a Small World

- The small-world model
 - Milgram: six degrees of separation
 - Watts: between order and randomness
 - short-distance clustering + long-distance shortcuts
- “Scale-free” link distribution
 - $P(n) = 1/n^k$
 - has no term related to the size of the network
 - applies at all scales from small to large
 - most nodes have only a few connections
 - some have a lot of links
 - important for binding disparate regions together

© 2001 Alberto Montresor

71

Freenet as a Small World



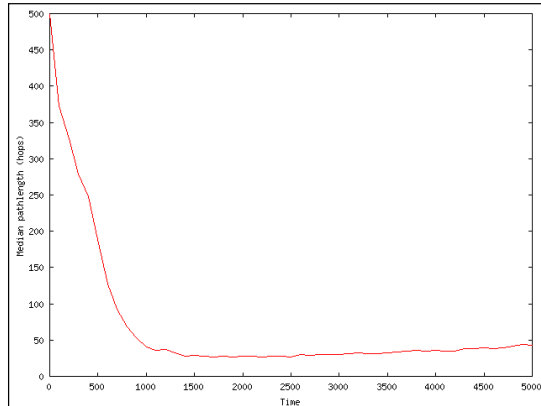
© 2001 Alberto Montresor

Source: freenet.
sourceforge.net

72

The Importance of Routing

- Existence of short paths is not enough:
 - they must be found
 - Adaptivity helps Freenet find good paths



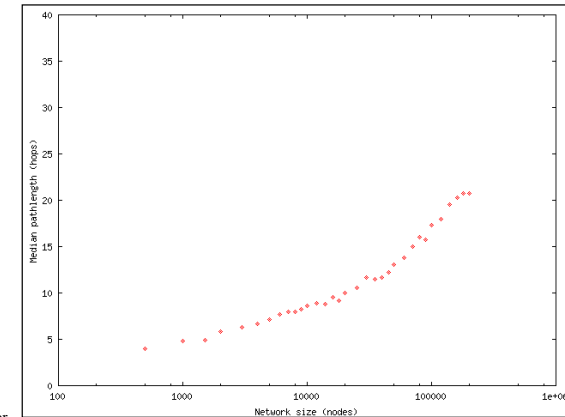
Source: freenet.
sourceforge.net

© 2001 Alberto Montresor

73

Freenet Scalability

- Real-world networks are large
 - nearly 400,000 downloads of Freenet
 - 50 million Napster users



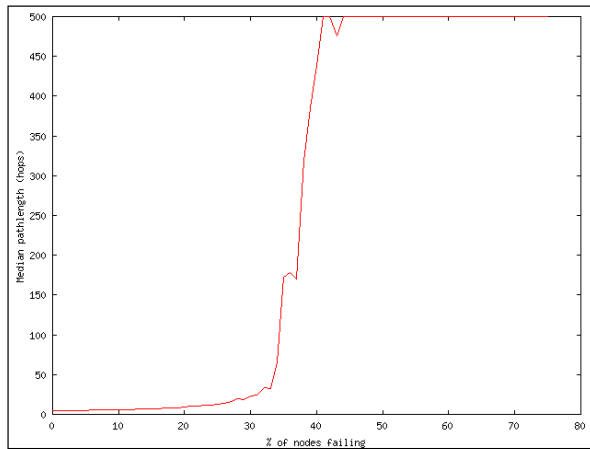
Source: freenet.
sourceforge.net

© 2001 Alberto Montresor

74

Fault Tolerance

- Two types of failure: random failure



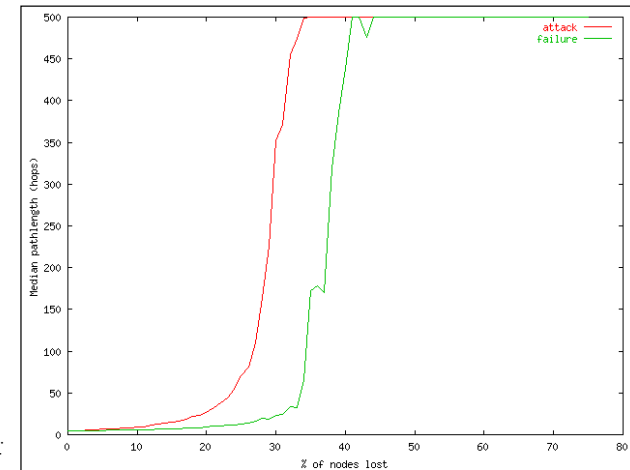
Source: freenet.
sourceforge.net

© 2001 Alberto Montresor

75

Fault Tolerance

- Two types of failure: targeted attack



Source: freenet.
sourceforge.net

© 2001 Alberto Montresor

76

Freenet Pros

- **Avoiding the “slashdot effect”:**
 - The request mechanism will cause popular data to be transparently mirrored close to requestors
 - Example: file inserted in UK but popular in Italy
- **Exploiting adaptively the resources:**
 - Nodes that successfully supply data will gain routing tables entries and be contacted more often than nodes that do not
- **Content-independent fault-tolerance:**
 - Lexicographic closeness of keys does not imply:
 - any closeness of the original descriptive strings
 - any closeness of the subject matter
 - Clustered files will have different content
 - “Similar” files will be scattered across the network

Freenet Pros

- **Removal of unwanted documents:**
 - The LRU policy of the datastore:
 - removes outdated documents
 - removes rarely accessed documents
- **Reader/Publisher Anonymity**
 - A node in a request path cannot tell whether its predecessor in the path initiated the request or not
 - Messages not immediately discarded when TTL=1
 - Depth starting with a value greater than 0
 - Note: Possible use of traffic analysis
- **Server anonymity / Deniability:**
 - Difficult to relate a document to a server
 - Operators can deny to know the content of its datastore

Freenet Cons

- **Search Problem**
 - The keys are Freenet’s current weak point
 - Hashing of keys:
 - Human-Rights.doc and HumanRights.doc have completely different hash values
 - Hashing renders Freenet unusable for random searches
 - Need an “out-of-band” communication of keys
- **A solution:**
 - Gateway to the Web exists under the name of Fproxy
 - Possibility of using hyperlinks

Freenet: Conclusions

- **Future Work**
 - Create a searching mechanism
 - Better client interfaces
 - Metering tools
 - Difficult problem due to anonymity
 - Searches modify the Freenet network

Seti@Home

- **A scientific experiment that uses Internet-connected computers in the**
Search for Extraterrestrial Intelligence (SETI)
- **Based on Master/Slave Cycle Sharing**
 - People download the “client”
 - Seti@Home distributes radio signal chunks to be analyzed to clients
 - Clients compute them and return the result to Seti@Home

Seti@Home

- **Some figures:**
 - Friday March 30, 2001

	Total	Last 24 Hours
Users	2.895.449	2581
Results received	306.441.320	432749
Total CPU time	611.327 years	years
Average CPU time x work unit	17 hr 28' 32"	19 hr 58' 49"
Floating Point Op.	7.793060e+20	1.687721e+18 19.53 TeraFLOPs/sec

Distributed.net

- **Distributed computation of mathematical problems**
 - Des breaking (DES-III)
 - less than 24 hours
 - sponsored by RSA
 - Csc (encryption breaking)
 - RC5 breaking
 - 250 days
 - sponsored by RSA
 - Optimal Golomb Ruler

P2P Initiatives

- **P2P projects share common problems**
 - Multitude of P2P projects appeared recently
 - Common problems: security, scalability, reliability, routing
- **Unfortunately:**
 - a theoretical and technical framework capable of supporting P2P application development is still missing
 - traditional techniques, given the substantial increase in the *complexity* of these systems, are not suitable
- **This lack has been recently recognized by the IT community**

P2P Initiatives: P2P-WG

- **Peer-to-Peer Working Group (P2P-WG)**
 - Founded by Intel in July 2000
 - Aimed at providing new communication standards for the emerging P2P market
 - P2P-WG is modeled as an industry consortium
 - 25.000\$: steering committee member
 - 5.000\$: technical committee member
 - 500\$: guest
 - Note: *The P2P-WG does not use a P2P-approach to design the new standards for P2P!*

P2P Initiatives: JXTA

- **The acronym JXTA comes from “*JuXTApose*”;** according to **Bill Joy (chief scientist at Sun):**

“[Juxtapose] means putting things next to each other, which is really what peer-to-peer is about”
- **JXTA is aimed at providing minimal mechanisms by which people can further innovate:**
 - being able to pipe from one peer to another
 - being able to group a set of peers together, and create groups of groups
 - doing monitoring and metering
 - a security layer

P2P Initiatives: JXTA

- **Sun is working on a first specification and reference implementation**
 - expected for April 2001
- **JXTA is an open source project that will be released under the Apache license**

P2P Initiatives: OpenP2P

- **O'Reilly has created a web site devoted to P2P aimed at:**
 - serving as repository of P2P documentation and links
 - guiding the development of open protocols and standards
- **OpenP2P Web site:**

<http://openp2p.com>

Conclusions

- P2P is thought as the distribution model of the future
- Enormous efforts are being addressed to P2P
 - Intel et al, Sun, O' Reilly
- But:
 - "Push" technology was "the distribution model of the future" in 1997-1998

Discussion

Why P2P in a course about
"Complex Adaptive Systems"?