The UNIVERSITY *of* EDINBURGH
SCHOOL *of* COMPUTER SCIENCE

CS4/MSc

Distributed Systems

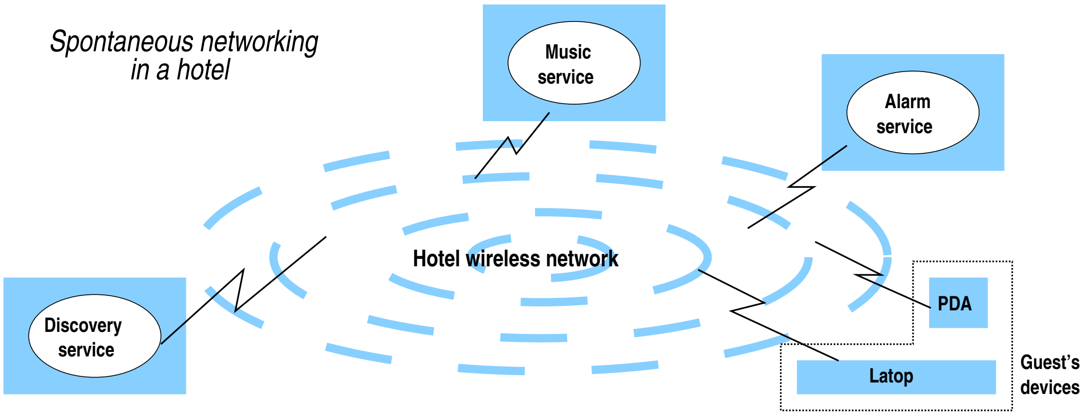
Jane Hillston

jeh@dcs.ed.ac.uk

Room 1604

(Lecture 18: Emerging Technologies, 7th March 2002)

Spontaneous Networking (1)



The growth in the number of mobile devices is moving the emphasis of networking away from fixed long-term networks to more flexible, dynamic and ad hoc arrangements. This is supported by the use of wireless media. Fixed and mobile devices can establish connections to form a **spontaneous network**.

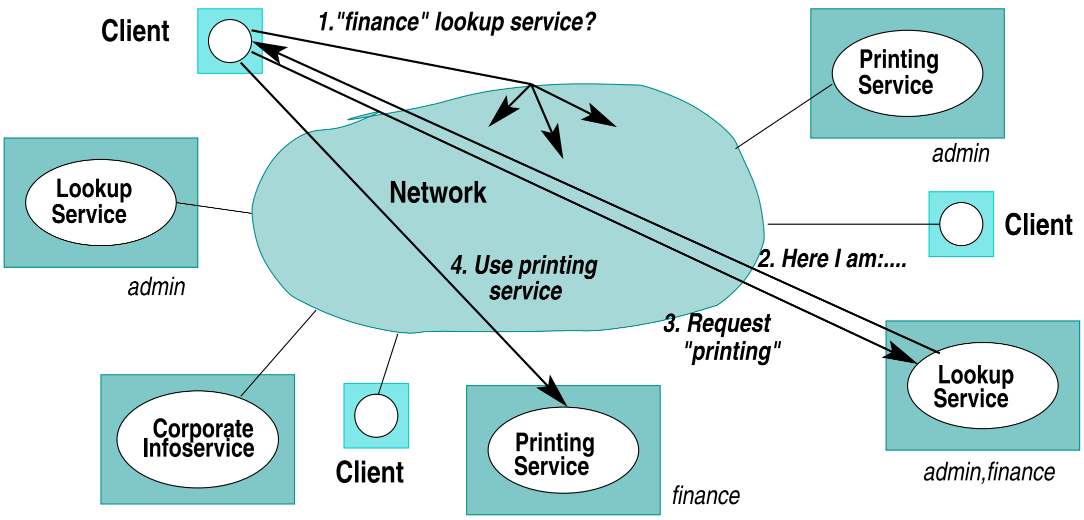
Spontaneous Networking (2)

- Fixed devices may provide services to users and other devices in the same environment.
- A portable device can gain access to these services as well as global services such as the Web.
- A device brought into a new network environment is transparently reconfigured to obtain connectivity: the user should not have to type in the names or addresses of local services to achieve this.
- Devices that find themselves inserted into (or moving between) existing networks of devices discover automatically what services are provided there without special configuration actions by the user. This assumes that the network supports some form of **discovery service**.
- There are issues of **security** and **privacy** related to the use of spontaneous networks which are yet to be resolved.

Discovery Services

- The purpose of a **discovery service** is to accept and store details of services that are available on the network (**registration service**) and to respond to queries from clients about them (**lookup service**).
- Services are registered as ⟨**name, attributes**⟩ pairs and matching of requests to services will be done on the basis of the attributes. One of the attributes is a location which can be returned in response to a client request, allowing the client to subsequently to communicate with the service directly.
- The extent of a network covered by a discovery service is called its **scope** and this is often defined in terms of connectivity to a particular network. This is generally appropriate because clients in a spontaneous network usually want to access services which are physically close by.
- **Jini** is a Java-based discovery system designed to be used for spontaneous networking. It assumes that JVMs run in all the hosts and communication is by means of RMI.

Jini Discovery Service



A Jini client discovering and using a printing service. The required service is within the **group** “finance”. To find the appropriate lookup service the client **multicasts** to the network. The lookup service responds with its unicast address for accepting requests.

Peer-to-peer Systems

- "**Peer-to-peer**" is a buzz-phrase of the moment and is used to mean different things to different people.
- In its purest sense it is used to define decentralised systems, which offer a service without the requirement for there to be a single **server**. Each process is a **peer** and may act as client or server. An example of such a system is the distributed information system **Freenet**. (This definition would exclude systems such as **Napster**.)
- However the term is also used to describe distributed processing systems, such as **SETI@Home**, which aim to use processing power which is widely distributed across the Internet to tackle a single computational problem.
- Peer-to-peer can be regarded as the class of technologies which aim to take advantage of the **resources** (storage capacity, CPU cycles, information content) at the "**edges**" of the Internet.

On the Edge

What counts as the edge of the Internet? The edges of the Internet are the user machines. Some of these, such as home PCs may have only periodic connection to the Internet, for example, via modems.

Where do the "resources" come from? As PC hardware and software have improved there is typically spare capacity in these machines for at least part of the time.

How can they be exploited? At the edge of Internet, resources may be only intermittently connected and connection can be using unpredictable IP addresses. This means that the usual mechanisms for locating resources (DNS) cannot be used. Currently users need to explicitly donate their resources, either because they intend to benefit from the distributed service, or from some altruistic motivation.

Benefits of decentralisation? Removal of the reliance on a single server improves performance, scalability, and availability. It can also lead to anonymity and distributed costs and administration.

Freenet

- Freenet is a distributed, anonymous information storage and retrieval system, particularly aimed at combatting censorship.
- It aims to protect the anonymity of both authors and readers and avoid removal of access to sensitive data by denial of service attacks.
- Freenet nodes work collectively, pooling their storage resources and cooperating to route requests to the most likely physical location of the data.
- Files are referred to in a location-independent way (avoiding DNS) based on **keys**.
- Availability is maintained because files are replicated close to requests for them, and the more popular files will be replicated to a greater degree.
- Fault-tolerance is provided because there is no single point of failure.

Freenet: basic behaviour

- Each node maintains is own local datastore which it makes available to the network for reading and writing.
- Each node also keeps a dynamic routing table containing addresses of other nodes and the keys that they are thought to hold.
- Requests for keys are passed from node to node through a chain of proxy requests: each node makes a local decision about where to send the request next.
- The route chosen will depend on the key.
- Keys are assigned to files based on a hash function. Different options are given for deriving keys, essentially offering differing levels of security.
- Any node can join the network once it knows the address of one (or more) node which is already a member.

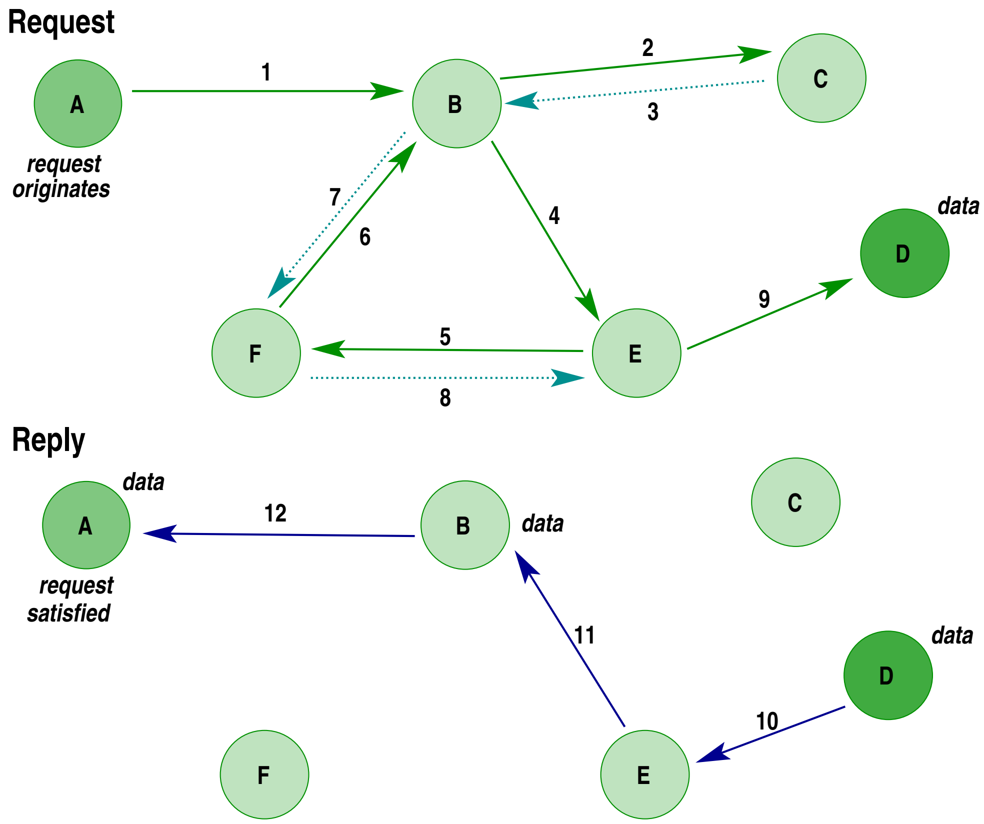
Freenet: storing data

- When a user wishes to store a file she calculates a key for it.
- Then she sends an **insert** message to her own node specifying the proposed key value and a hops-to-live value (this determines the number of nodes it will be stored on).
- A node receiving an insert message checks that the key is not already assigned to a file in its datastore. If it is, it returns the file to the sender of the message. The user will need to try again with another key.
- If the key is not found, the node forwards the insert message to the node in its routing table which has the closest key.
- If the hops-to-live limit is reached without a key collision being detected an **all clear** message is propagated back to the user.
- The user then sends the data which is propagated along the path established by the insert message and stored at each node along the path. The routing tables of each node along the path will be updated with the user's node matching the key.

Freenet: retrieving data (1)

- To retrieve a file a user must first obtain or calculate its binary file key.
- She then sends a **request** to her own node requesting the file matching the key.
- When a node receives a request it first checks its own store. If the file is there, is it returned together with a tag indicating that it was this node that satisfied the request; if it is not, the node looks up the closest key in its routing table and forwards the request to the corresponding node.
- Each request is given a **hops-to-live** which is decremented at each node visited. This prevents infinite chains.
- If the request is subsequently successful and returns with the file, the node passes the file to whoever requested it. It also caches a copy of the file in its own datastore and creates a new entry in its routing table, associating the first node that satisfied the request with the key.

Freenet: retrieving data (2)



Freenet: retrieving data (3)

- If forwarding a request according to the algorithm above would create a loop (as in step 6 in the diagram above), the request is refused. The node then looks for the key in its routing table which is second closest and forwards the request in that direction.
- If a node or a link are unavailable (down) the request is forwarded to the second closest node according to the routing table.
- This approach is followed until the complete key table has been tried at which point the node will report failure to the requester. That node will then try its next choice etc.
- This is a *steepest-ascent hill-climbing search with backtracking*.
- As a file is returned back along the path which the request followed (ignoring unsuccessful branches) by default each node copies the file and updates its routing table. However, as a security measure, a node may arbitrarily decide to change the reply message to report itself, or any other node, to be the source of the file.

SETI@Home

- SETI@home is a scientific experiment that uses Internet-connected computers in the Search for Extraterrestrial Intelligence (SETI).
- Users can participate by running a free program that downloads and analyses radio telescope data. The telescope in Arecibo, Puerto Rico, records 35 Gbytes of data per day.
- Users join the project by registering at a web site.
- The application runs as a screen-saver, using the computer when it would otherwise be idle.
- The distributed approach works well because the data analysis task can be easily broken into small independent chunks (0.25 Mbyte chunks termed **work-units**). Each work unit is replicated.
- Connection is only made to transfer fresh or completed work-units, and this can be under user control.
- The system is centralised in the sense that control is maintained in Berkeley, where a large database keeps track of the work-units and their current status.

Internet-Scale Operating Systems

- In order to ease the development of Internet-scale applications such as SETI@Home, proposals are being made for Internet-scale operating systems (ISOS).
- Just as with a conventional operating system the objective of the ISOS would be to manage resources transparently to the applications, creating a virtual computing platform on which programs can operate as if they were in sole possession of the system.
- It is software layer would have responsibility for allocating and scheduling resources for each application, and handling communication between hosts.
- It might also be responsible for accounting information since it is unlikely that the current philanthropic attitude towards large distributed computation will continue.
- Some notion of **quality of service** (QoS) guarantee may also be incorporated allowing different priorities to be used in resource allocation and scheduling (potentially with economic consequences).