Architectural styles for software systems

Peer to peer
Peer to peer computing

- a class of applications that takes advantage of resources—storage, cycles, content, humans—available at the edges of the Internet

- Because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses, peer-to-peer nodes must operate outside the DNS system and have autonomy from central servers
The overlay network layer is responsible for implementing an efficient routing algorithm: the nodes in the system are structured in order to decrease the search steps necessary to find the target identifier. Each node maintains a local routing table, which holds the identifiers of other nodes in the system.
Distributed hash tables

A distributed hash table (DHT) provides a lookup service similar to a hash table: (key, value) pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key.
P2P: Overview

- A P2P system is a distributed collection of peer nodes
- Each node is able to provide services, as well as to make requests, to other nodes
  - Each node acts as both a server and a client
- The goal of this style:
  - To share resources and services (data, CPU, disk, …)
Peer-to-peer systems

- File sharing systems based on BitTorrent
- Messaging systems such as Jabber
- Payments systems – Bitcoin and Ethereum
- Databases – Freenet is a decentralized database
- Phone systems – Viber or Skype
- Computation systems - SETI@home
P2P: requirements and drivers

- Typical functional characteristics of P2P systems:
  - File sharing system
  - File storage system
  - Distributed file system
  - Redundant storage
  - Distributed computation

- Typical non-functional requirements:
  - Availability
  - Reliability
  - Performance
  - Scalability
  - Anonymity
<table>
<thead>
<tr>
<th>Class</th>
<th>Peer</th>
</tr>
</thead>
</table>

**Responsibilities**
- Component
- Handles User interaction
- Asks other Peers for searching some data
- Asks some Peers for obtaining some data
P2P: Brief History

- Although they were proposed 30 years ago, they mainly evolved in the last decade
- File sharing systems showed the power of the concept (Napster, 1999; Gnutella, 2000)
- In 2000, the Napster client was downloaded by 50 millions users
  - Traffic peak of 7 TB in a day
- Gnutella followed Napster’s footprint
  - The first release was delivered in 2003
  - In June 2005, Gnutella's population was 1.81 million computers; in 2007, it was the most popular file sharing network with an estimated market share > 40%
  - Host servers are listed at gnutellahost.com
The phases of a P2P application

- A P2P application is organized in three phases:
  - **Boot**: a peer connects to the network and actually performs the connections (remark: P2P boot is rare)
  - **Lookup**: a peer looks for a provider of a given service or information (generally providers are SuperPeers)
  - **Resource sharing**: resources (requested and found) are delivered, usually in several segments
P2P resource sharing example: bitTorrent

BitTorrent tracker identifies the swarm and helps the client software trade pieces of the file you want with other computers.

Computer with BitTorrent client software receives and sends multiple pieces of the file simultaneously.
P2P: Classification

There are three types of P2P architecture, different with respect to the lookup phase:

- **Centralized**
  - Centralized network architecture uses a centralized indexed server to maintain a database of all the content and users at any time
  - The database is updated whenever a peer logs on to the network

- **Decentralized** (Pure P2P)
  - Each peer acts as an index server, searches and holds its own local resources, and as a router, relaying queries between peers

- **Hybrid Architecture**
  - Deploys a hierarchical structure by establishing a backbone network of Super Nodes that take on the characteristics of a central index server
There is a centralized index used to search the information. When peer connects, it informs central server:
- IP address
- Content

File transfer is decentralized, but locating content is highly centralized.

Example: Napster
Centralized: Architecture

Components:

- **Peer**
  - An entity with capabilities similar to other entities in the system
  - Each node is both a server and a client
  - Autonomous: no administrative authority
  - Unreliable: nodes enter and leave the network “frequently”

- **Index Server**
  - An entity with special capabilities:
    - Allow peer to join the system
    - Allow the research of content
    - Maintain a database of all the content and users at any time, which is updated whenever a peer logs on to the network

Connectors:

- **Network protocol**
  - Often specialized for P2P communication
Centralized Index: Component Diagram
Centralized Index: Class Diagram

```
Directory Index
+ Search()
+ Login()

Peer
+ FileTransfer()
```

1 to * relationship between Directory Index and Peer.
Centralized Index: Sequence Diagram
Centralized Index Example: Napster
Centralized Index: Pro & Cons

- **Benefits:**
  - Low per-node state
  - Limited bandwidth usage
  - High success rate
  - Fast search response time
  - Easy to implement and maintain

- **Pitfalls:**
  - Single point of failure
  - Vulnerable to censorship
  - Limited scale
  - Possibly unbalanced load
  - Database might be obsolete
Decentralized P2P organizes the overlay network as a random graph.

Each node knows about a subset of nodes, its “neighbors”:
- Neighbors are chosen in different ways:
  - physically close nodes, nodes that joined at about the same time, etc.

Example: Gnutella, Bitcoin

*Copyright © Tore Mørkved, Peer-to-Peer Programming with Wireless Devices*
Decentralized : Class Diagram
Decentralized: Architecture

- **Components:**
  - **Peer**
    - An entity with capabilities similar to other entities in the system
    - Each node is both a server and a client
    - Autonomous: no administrative authority
    - Unreliable: nodes enter and leave the network “frequently”
    - Local knowledge: nodes only know a small set of other nodes

- **Connectors:**
  - **Network protocol**
    - Often specialized for P2P communication
Decentralized Example: Gnutella
Decentralized: Component Diagram

![Decentralized: Component Diagram](image)

**Fig. 2.** Typical component-based structure of a Gnutella servant

Cornelli et al., Implementing a Reputation-Aware Gnutella Servent, Proc. Networking 2002
Decentralized: Sequence Diagram (1)

Fig. 3. UML Sequence diagram of a search session
Fig. 4. UML Sequence diagram of a reply session
Decentralized: Pro & Cons

- **Benefits:**
  - Limited per-node state
  - Fault tolerant

- **Pitfalls:**
  - High bandwidth usage
  - Long time to locate item
  - No guarantee on success rate
  - Possibly unbalanced load
P2P: Hybrid Architecture

- Deploys a hierarchical structure by establishing a backbone network of Super Nodes that take on the characteristics of a central index server.
- When a client logs on to the network, it makes a direct connection to a single Super Peer.
- Example: Skype
Example: Skype

Skype Architecture

- Registration & Authentication
- STUN/TURN/TCP used as required
- Authentication Layer
- Super Node Layer
- Skype Client Layer

Call between public clients
Call between firewalled clients

Skype login server
Message exchange with the login server during login

Ordinary host (SC)
Super node (SN)
Neighbor relationships in the Skype network
Hybrid: Architecture

- Components:
  - **Peer**
    - An entity with capabilities similar to other entities in the system
    - Each node is both a server and a client
    - Autonomous: no administrative authority
    - Unreliable: nodes enter and leave the network “frequently”
  - **Super Peer**
    - Gathers and stores information about peer and content available for sharing
    - Act as servers to regular peer nodes, peers to other super Peers
    - Maintain indexes to some or all nodes in the system

- Connectors:
  - **Network protocol**
    - Often specialized for P2P communication
Hybrid Architecture: Component Diagram
Hybrid Architecture: Class Diagram

Super Peer

Search()
Login()
DisseminateQuery()

Simple Peer
Hybrid Architecture: Sequence Diagram
Hybrid Architecture Example: Skype (1)
A mixed client-server and peer-to-peer architecture addresses the discovery problem.

Replication and distribution of the directories, in the form of supernodes, addresses the scalability problem and robustness problem encountered in Napster.

Promotion of ordinary peers to supernodes based upon network and processing capabilities addresses another aspect of system performance:
- “not just any peer” is relied upon for important services
Hybrid Architecture Example: Skype (3)

- A proprietary protocol employing encryption provides privacy for calls that are relayed through supernode intermediaries.
- Restriction of participants to clients issued by Skype, and making those clients highly resistant to inspection or modification, prevents malicious clients from entering the network.
Hybrid Architecture: Pro & Cons

**Benefits:**
- Manageable per-node state
- Manageable bandwidth usage and time to locate item
- Guaranteed success

**Pitfalls:**
- Possibly unbalanced load
- Harder to support fault tolerance
Bitcoins

- Bitcoins are based on the idea of avoiding to let to spend twice the same digital coin using a chain of transactions recorded in a shared ledger.
- The Bitcoin system is peer-to-peer, and transactions take place between anonymous users directly, without an intermediary.
- These transactions are verified by network nodes and recorded in a public distributed ledger called a blockchain.
Decentralized: Bitcoin

Bitcoin node: main functions

Reference Client (Bitcoin Core)
Contains a Wallet, Miner, full Blockchain database, and Network routing node on the bitcoin P2P network.

Full Block Chain Node
Contains a full Blockchain database, and Network routing node on the bitcoin P2P network.

Solo Miner
Contains a mining function with a full copy of the blockchain and a bitcoin P2P network routing node.

Lightweight (SPV) wallet
Contains a Wallet and a Network node on the bitcoin P2P protocol, without a blockchain.

Pool Protocol Servers
Gateway routers connecting the bitcoin P2P network to nodes running other protocols such as pool mining nodes or Stratum nodes.

Mining Nodes
Contain a mining function, without a blockchain, with the Stratum protocol node (S) or other pool (P) mining protocol node.

Lightweight (SPV) Stratum wallet
Contains a Wallet and a Network node on the Stratum protocol, without a blockchain.
Anonimity

- Bitcoin addresses are not tied to people
- Transactions are not tied to people
- Transaction data is transmitted to a random subset of nodes

- However, there are some (expensive) methods to de-anonymize a user, so Bitcoin is not perfectly anonymous
Blockhain
Moving bitcoins between wallets
Bitcoin

https://bulldozer00.com/2015/10/25/bitcoin-in-uml/
Implementing bitcoins: blockchain

- Each bitcoin (BTC) node retains a copy of the global, publicly shared Blockchain.
- The Blockchain has 380K+ Blocks.
- Each Block has one or more validated BTC Transactions embedded within it.
- Via the interface facilities provided by a BTC Node, a User composes a Transaction and submits it to the network for validation and execution.
- Each instance of a BTC Transaction contains a source address, destination address, the BTC amount to be transacted, and the source address owner’s signature.
Conclusions

- P2P networks are quite old: the Internet is a P2P
- Several new applications are implemented as p2p
- Blockchains are a powerful architecture for innovative financial applications
Self test

- What is the relationship between p2p and C/S?
- What is a hybrid p2p network?
- What is a blockchain?
References

- Grolimund, A Pattern Language for Overlay Networks in Peer-to-Peer Systems, EuroPloP 2006
- Ripeanu, Peer-to-peer architecture case study: Gnutella network, 2001
- Wang, Skype VoIP service-architecture and comparison, 2005
- Amoretti e Zanichelli, P2P-PL: A Pattern Language to Design Efficient and Robust Peer-to-Peer Systems, 2016
Questions?