

# Chapter VIII
# Support Infrastructure for Application Layer: Peer-to-Peer (P2P) Overlays



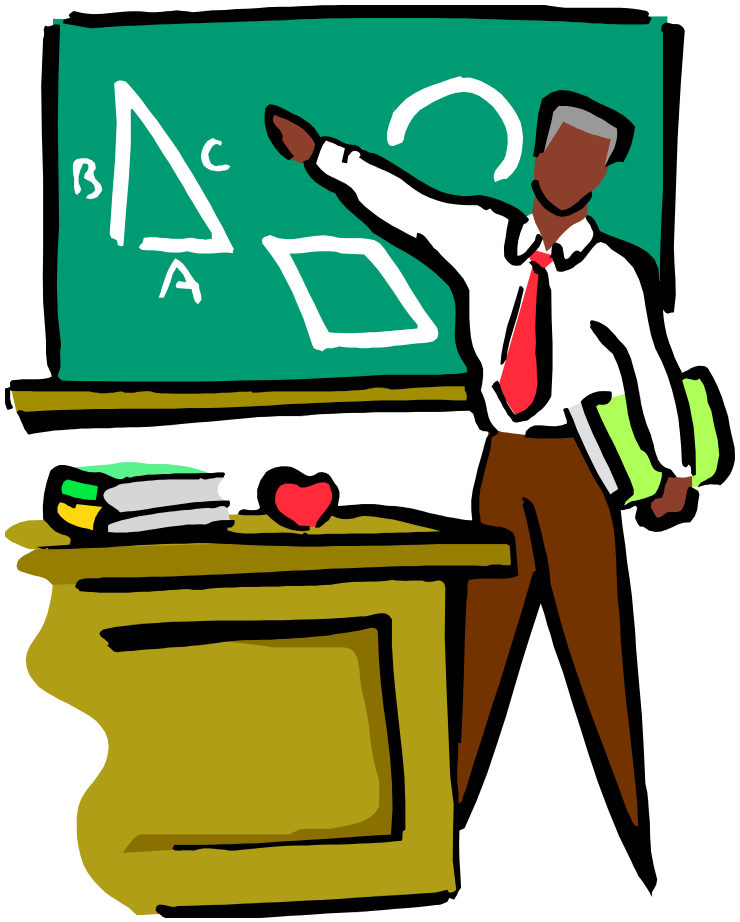
Roch H. Glitho

# Support Infrastructure

Support infrastructure for application layer

- Why?
  - Re-usability across application layer protocols
  - Modularity (i.e. separation between application layer protocol specification / design and infrastructure specification / design)
- Examples discussed in this course
  - Distributed Name System (DNS)
    - Mapping between application layer symbolic addresses and IP addresses
  - Peer to peer overlays
    - Connectivity, routing and messaging between peers for applications such as file sharing, IP telephony

# P2P Overlays




- 1 – Client/server vs. P2P computing
- 2 - Structured P2P Overlays vs. Unstructured P2P Overlays
- 3. Chord, Freenet and Skype
- 4. JXTA: A middleware for P2P applications development

# Client / server vs. P2P computing

- Client / server
  - Essence
    - Single server offering storage and computation
      - Static
      - Updates done solely by server provider
    - Clients access server
      - Passive role
      - No (or little) contribution to storage and computation
  - Underlying assumption
    - Clients have no (or little) storage and computation capabilities

# Client / server vs. P2P computing

- Client / server
  - Inherent issues:
    - Root:
      - server is a computational and network bottleneck
        - » Examples of issues
          - » Scalability
          - » Availability
          - » Efficiency

# Client / server vs. P2P computing

## P2P computing

- Several possible definitions
  - In this course:
    - Computing paradigm that relies on a network of peers (instead of a server) to solve the issues inherent to client / server paradigm, such as:
      - » Scalability
      - » Availability
      - » Efficiency
- Underlying assumption
  - Clients now have more and more storage and processing power that should be used

# Client / server vs. P2P computing

- P2P computing
  - Clients federate via a P2P network to offer storage and computational capabilities required by applications
    - Clients are called peers
      - Each peer may contribute according to its capabilities
      - More powerful peers sometimes called super – peers may contribute more
    - Pure P2P vs. hybrid P2P
      - Pure P2P:
        - » Fully decentralized architecture
          - » Not that common (e.g. Freenet)
      - Hybrid P2P
        - » Some level of centralization
          - » More common (e.g. Skype)

# Client / server vs. P2P computing

- P2P computing
  - Some examples of technical challenges
    - Self organization
      - Peers may join or leave the network anytime
    - Storage and look up
      - Where to store?
        - » Items may be stored on any set of peers
      - Efficiency of lookups (guarantee vs. performance)
    - Fault tolerance
      - Voluntary departures vs. un-voluntary departures
        - » What to do if a peer leaves?
        - » What to do if a peer goes down?

# Structured P2P overlays vs. unstructured P2P overlays

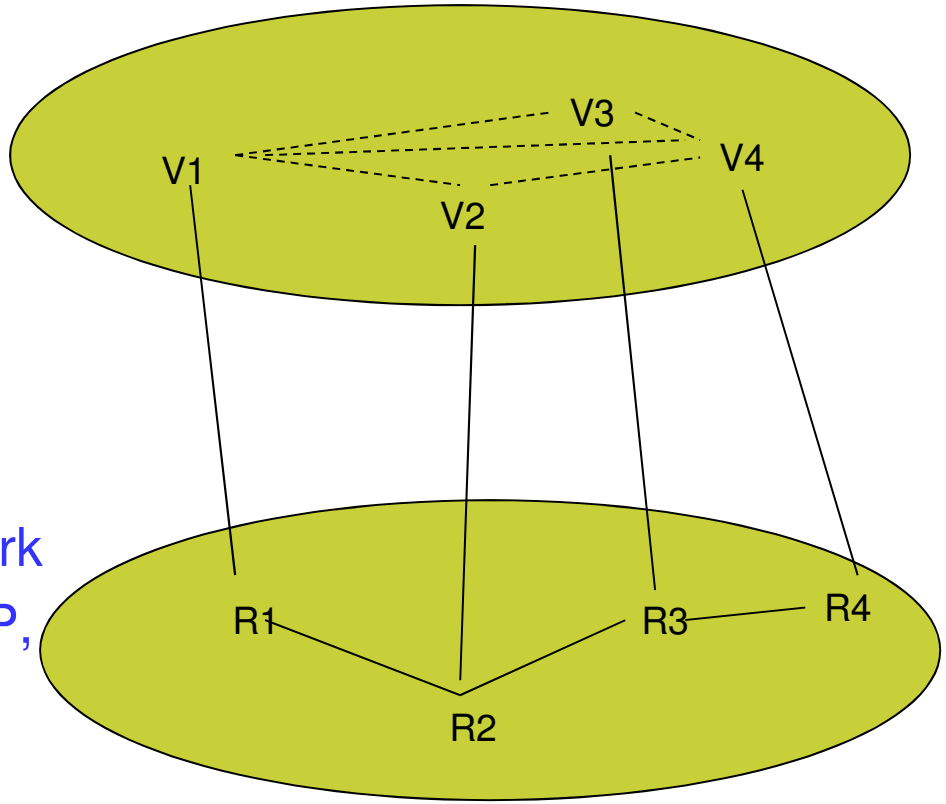
- P2P overlay
  - Current way of implementing P2P computing
    - Application layer virtual networks that provide storage, processing, connectivity and routing
      - Network built by peers that federate to offer storage and processing capabilities to applications
        - » Built on top of existing networks, thus the name of overlay
          - » Applications running on top of transport protocols of real network
          - » Real network nodes become virtual nodes in the overlay

# Structured P2P overlays vs. unstructured P2P overlays

## P2P overlay

Overlay (Above Transport)

Real network (PHY,link,IP, Transport)



# Structured P2P overlays vs. unstructured P2P overlays

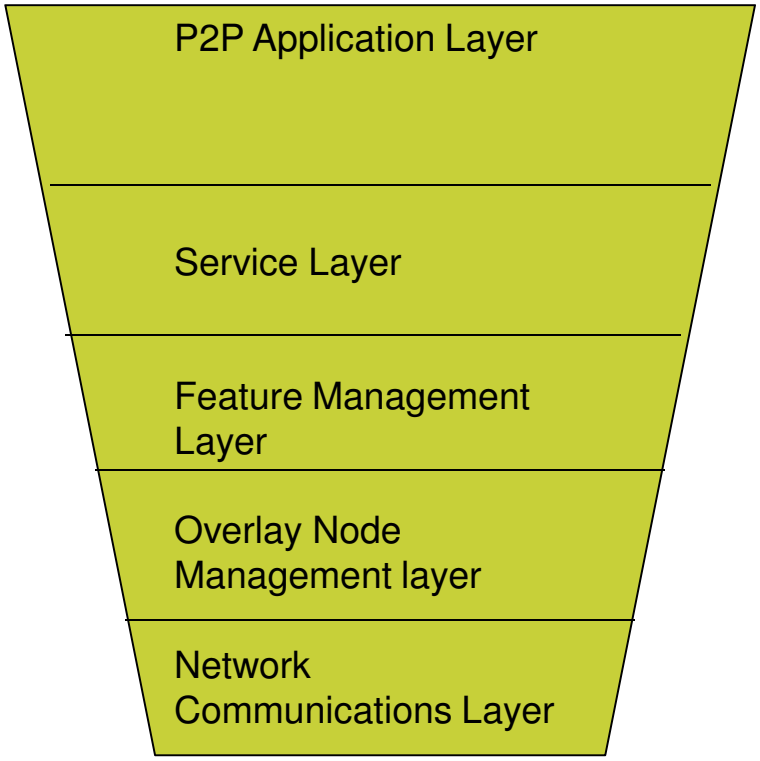
## P2P overlay

- Characteristics
  - own topology that may be different from the topology of the real network
  - Own protocols that may be different from the protocols used in the real network
  - May come with an application embedded in it (e.g. Skype) or as an infrastructure that can be used by other applications (e.g. CHORD)
  - APIs, toolkits are provided when the application is not embedded in the overlay

# Structured P2P overlays vs. unstructured P2P overlays

## P2P overlay

### Simplified abstract view (All this is above transport)



# Structured P2P overlays vs. unstructured P2P overlays

- Simplified abstract view
  - Application layer
    - Actual P2P applications (e.g. file sharing, IP telephony)
      - » Maybe either embedded in the P2P infrastructure or built by developers using APIs depending on the P2P overlay
  - Service layer
    - Services or building blocks used by developers to build applications
      - » Maybe or may not be visible to third party developers
  - Feature management
    - Features used by all applications (e.g. security, fault resilience)

# Structured P2P overlays vs. unstructured P2P overlays

- Overlay nodes management
  - Routing, resources discovery, location look up
- Service layer
  - Services or building blocks used by developers to build applications
    - » Maybe or may not be visible to third party developers
- Network communication layer
  - Interface to the real network
    - On top of a real transport network

# Structured P2P overlays vs. unstructured P2P overlays

## Structured P2P overlays

- Tightly controlled topology
  - Content placed at very specific locations
    - Efficient subsequent queries
    - Technique used: Distributed Hash Table (DHT)
      - Generation of a key
        - » Put (Key, value)
        - » Value = Get (key)
      - Each peer has a small routing table of neighbouring peers
        - » Node ID
        - » IP address
      - Messages routed progressively using Node ID that are closer to the key

# Structured P2P overlays vs. unstructured P2P overlays

## Structured P2P overlays

- Some examples
  - Content Addressable Network (CAN)
  - Chord
  - Tapestry
  - Pastry
  - Kademlia
  - Viceroy

# Structured P2P overlays vs. unstructured P2P overlays

Un-structured P2P overlays

- Loosely controlled topology
  - Content placed at random locations
    - Flooding techniques
      - » Efficient for highly replicated content
      - » Inefficient for rare content

# Structured P2P overlays vs. unstructured P2P overlays

## Un-structured P2P overlays

- Some examples
  - Napster
  - Freenet
  - Gnutella
  - KazaA
  - BitTorrent

# Structured P2P overlays vs. unstructured P2P overlays

## Chord, Freenet and Skype

- Chord
  - Structured P2P overlay that can be used to build applications
  - An example of possible applications
    - Time shared storage for nodes with intermittent connection
      - Goal:
        - » Have one's data always available (even when disconnected)
      - Solution
        - » Store the data of other peers when connected and get ones data stored by other peers when disconnected

# Structured P2P overlays vs. unstructured P2P overlays

## Chord, Freenet and Skype

- Chord
  - Key features
    - Load balancing
    - Full decentralization
    - Scalability
    - Availability
    - Flexible naming

# Structured P2P overlays vs. unstructured P2P overlays

## Chord, Freenet and Skype

- Freenet
  - Goal
    - Create an un-censorable and secure global information storage system
      - Unstructured P2P
      - Application embedded in the P2P overlay
        - » Efforts to decouple the application from the P2P overlay were not successful

# Structured P2P overlays vs. unstructured P2P overlays

## Chord, Freenet and Skype

- Freenet
  - Requirements
    - Privacy for information producers, consumers and holders
    - Resistance to information censorship
    - High availability and reliability
    - Efficient, scalable and adaptive storage and routing

# Structured P2P overlays vs. unstructured P2P overlays

## Chord, Freenet and Skype

- Freenet
  - Architectural principles
    - Users use globally unique identifier (GUID) to insert and retrieve files
      - GUIDs are assigned by the system
      - Data may be encrypted before insertion in the network
      - Files are stored on some set of nodes (may migrate or be replicated)
    - Messages travel through node to node chains and each link is individually encrypted
    - Controlled fllooding

# Structured P2P overlays vs. unstructured P2P overlays

## Chord, Freenet and Skype

- Skype
  - Application embedded in the P2P overlay
    - No design information available in the public domain
      - The little that is known is by reverse engineering

# Structured P2P overlays vs. unstructured P2P overlays

## Chord, Freenet and Skype

- Skype
  - Several servers
    - Login server
    - Skype-out server (PC to Public Switched Telephony Network (PSTN) calls)
    - Skype 0 in server (PSTN calls to PC)

# Structured P2P overlays vs. unstructured P2P overlays

## Chord, Freenet and Skype

- Skype
  - Several servers
    - Login server
    - Skype-out server (PC to Public Switched Telephony Network (PSTN) calls)
    - Skype - in server (PSTN calls to PC)

# Structured P2P overlays vs. unstructured P2P overlays

## Chord, Freenet and Skype

- Skype
  - Overlay architecture
    - Hierarchical
      - Ordinary host
      - Super nodes
        - » Every ordinary host is connected to a super node
          - » Routing table contains list of reachable super nodes
        - » Super nodes are interconnected

# Structured P2P overlays vs. unstructured P2P overlays

## Chord, Freenet and Skype

- Skype
  - Signaling
    - Always over TCP
      - May go directly from caller to callee (if callee is in caller busy and both are with public IP)
      - May go via a super node (When for instance callee is behind a NAT)



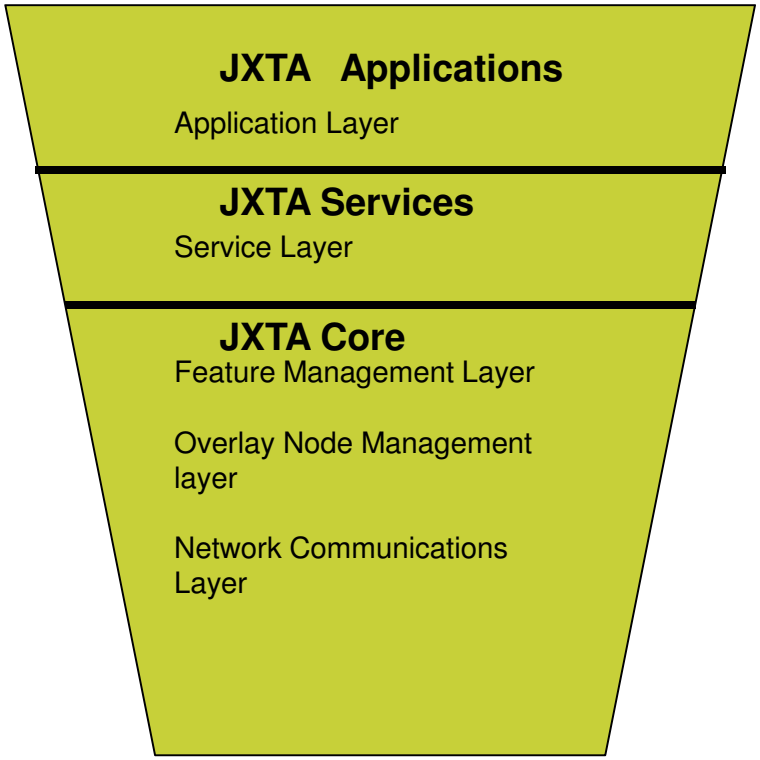
# JXTA: A middleware for P2P application development

## Key objectives

- Interoperability
- Platform (e.g. OS, programming language) independence
- Ubiquity (e.g. possibility to run on a wide range of devices: from laptops to cell phones and wireless sensors)
- Application independence (i.e. possibility of building any p2p application – file sharing, IP telephony)

# JXTA: A middleware for P2P application development

## Architecture



# JXTA: A middleware for P2P application development

## Programming models and APIs

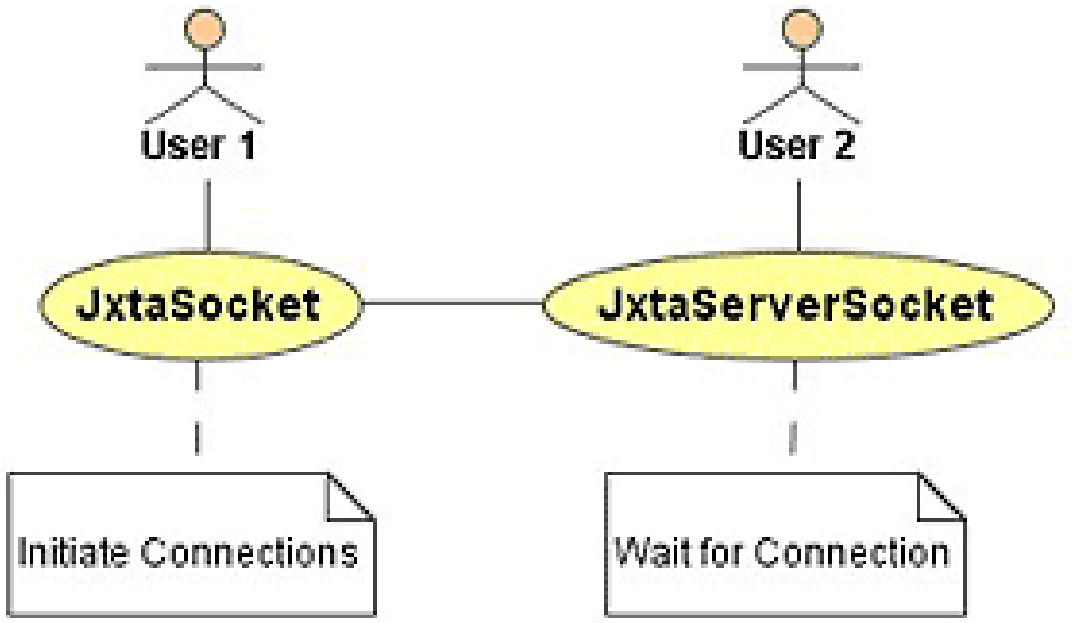
- A very wide range model including a socket programming model / API

# JXTA: A middleware for P2P application development

- A set of protocols
  - Some examples
    - Peer discovery
    - Peer information protocol
    - Peer membership protocol

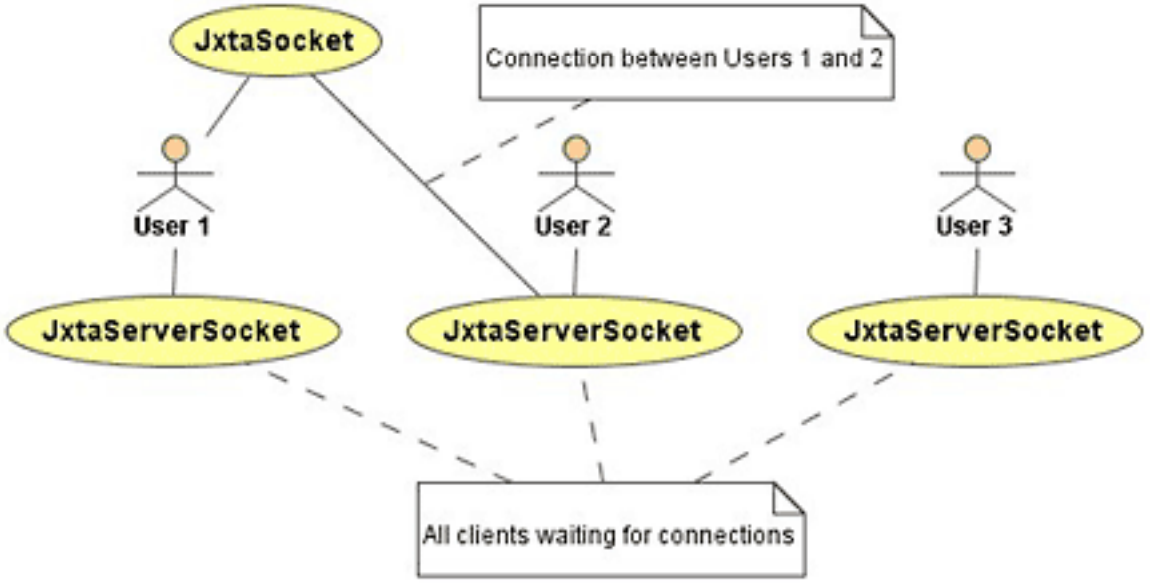
# JXTA: A middleware for P2P application development

P2P chat using sockets (every peer is both a client and a server)



# JXTA: A middleware for P2P application development

P2P chat using sockets (every peer is both a client and a server)



# JXTA: A middleware for P2P application development

## P2P chat using sockets (every peer is both a client and a server) – Joining the default peer group (netgroup)

```
Public void startJXTA(){ 2: netPeerGroup =
   PeerGroupFactory.newNetPeerGroup(); 3: PeerGroupID
   peerGroupBinaryID = MD5ID.createPeerGroupID
   (netPeerGroup.getPeerGroupID(),"hello world application",
   "chat"); 4: applicationPeerGroup =
   PeerGroupTool.createAndJoinPeerGroup (netPeerGroup,
   "Hello World",peerGroupBinaryID); 5: }
```

# References

1, E.K. Lua et al, A Survey and Comparison of Peer-to-Peer Overlay Network Schemes, IEEE Communications Surveys and Tutorials, March 2004

2. I. Stoica et al, Chord: A Scalable Peer to peer Lookup Protocol for Internet Applications, IEEE/ACM Transactions On Networking, 2003
3. I. Clarke et al., Protecting Free Expression On-Line with Freenet, IEEE Internet Computing, January/February 2002
4. S. A. baset and H. Schulzrinne, An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol, IEEE Infocom 2006
5. L. Gong, JXTA: A Network Programming Environment, IEEE Internet Computing 2001
6. The Socket API in JXTA 2.0 http://java.sun.com/developer/technicalArticles/Networking/jxta2.0/index.html