Chair for Network Architectures and Services
Department of Informatics
TU München – Prof. Carle

Peer-to-Peer Systems and Security IN2194

Chapter 1
Peer-to-Peer Systems
1.5 General aspects of P2P

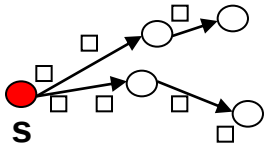
Dr. Heiko Niedermayer
Prof. Dr.-Ing. Georg Carle

Basic Operations

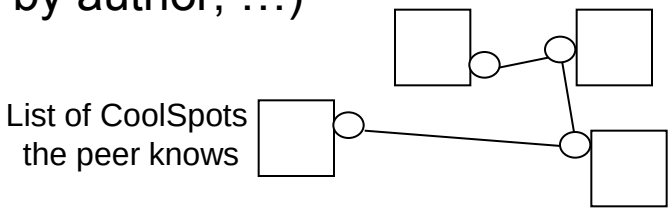
Example Applications

- The examples on this slide do not correspond to real networks, but they are used to illustrate possible solutions through the P2P part of this lecture. So, we will build them differently from section to section.

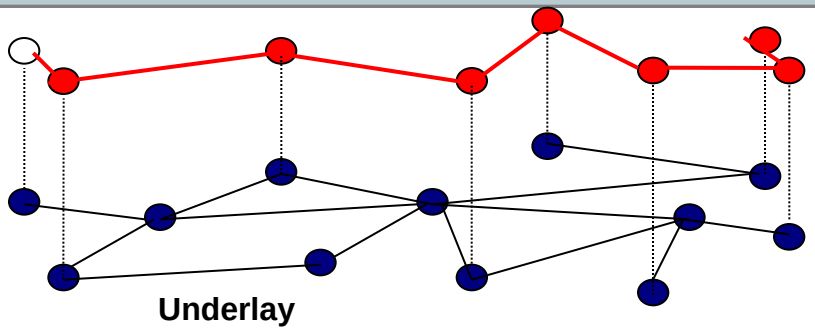
- A Multicast P2P Network
  - Multicast = 1 sender, many receivers
  - A source node s sends out a stream of messages.
  - A set of other nodes wants to receives the messages.



- CoolSpots Munich P2P Network
  - A Peer-to-Peer network that provides information about interesting objects (GPS coordinate, name of object, description of object, keywords for object, author, object rating by author, …)



Application Requirements



Multicast from the white game server as source to its peers.
Is this a good graph for fast delivery?
→ No, a balanced tree allows O(logn) diameter.

Application

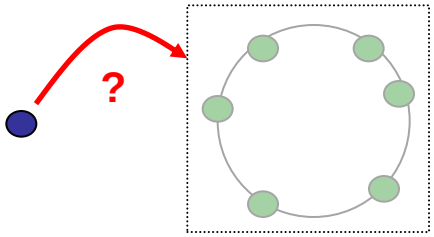
- Peer-to-Peer networks are usually created for an application or application scenario.
  - Filesharing
  - File Distribution
  - Instant Messaging and Voice-over-IP
  - Multicast
  - Peer-to-Peer Video Streaming
  - Anonymous communication and services
  - …
- The application is the purpose of the Peer-to-Peer network.
- The application and its requirements determine if a given graph is a good or a bad choice.

Basic operations

- Life of a P2P network
  - Bootstrapping
  - Membership / Identity / Roles
  - Dynamics / Structure / Maintenance
  - Operations

Bootstrapping

How to find the Peer-to-Peer overlay?



Problem

- How do I contact a node in the network?
- How can this be done without a server or a single server?

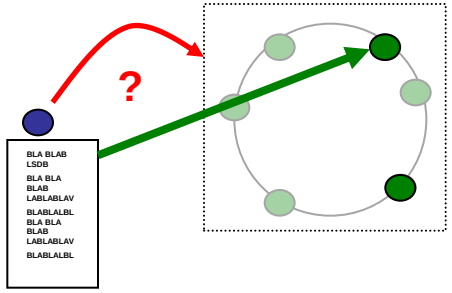
Necessary

- Some knowledge
  - direct knowledge („You already know it“)
  - indirection via some rendezvous point („Ask someone“)
- Important
  - *We cannot find a service we do not know anything of!*
  - However, more semantically-oriented rendezvous mechanisms might exist in the future (e.g. „give me a news-exchange network for Munich“), but they are not common today.

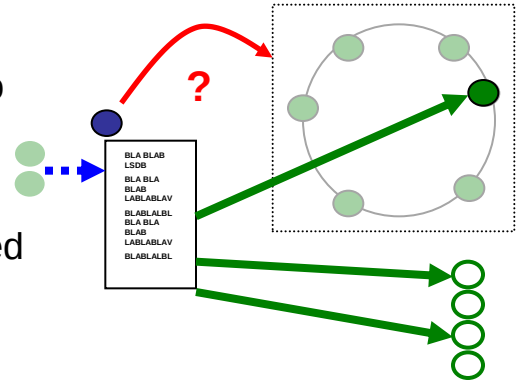
How to find the Peer-to-Peer overlay?

Solutions

- List with fixed (rendezvous) nodes
  - Method
    - Contact one of the nodes on the list
  - Assumption:
    - There exists a fixed set of nodes that are always accessable, now *and* in the future.
  - The approach is efficient and simple.
  - List or parts of the list may be updatable
    - e.g. by the user via the website

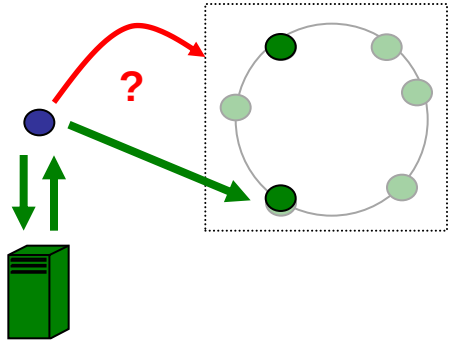


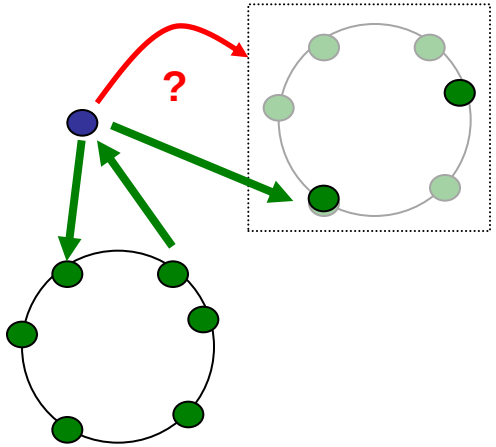
- Cache with nodes
  - Method
    - Store nodes you have seen. Contact some of them in order to join.
  - Assumption
    - Many nodes stay long in the network and can still be contacted the next time.
  - Cannot solve the initial contact problem (→ no cache yet).
  - Many nodes are stored anyway during runtime for robustness. Problem, what are the long-lived nodes?



How to find the Peer-to-Peer overlay?

- Aquire node list via some client/server service like HTTP
  - Mechanism
    - Ask a server for a list with current rendezvous nodes in the network.
  - Assumption
    - There is a server that has a knowledge of currently active (rendezvous) nodes in the systen. It needs to get either updated regularly or scan the P2P network.
  - For large networks no need to scan all nodes.
  - External nodes get to know network status information
    - Privacy issues, useful information for attacks, etc.
- Aquire node list via some _other_ Peer-to-Peer mechanism
  - Mechanism
    - Like above, but using a Peer-to-Peer system.
- Ask user





- While no individual solution is perfect, a combination of the mechnisms presented should work for most cases.

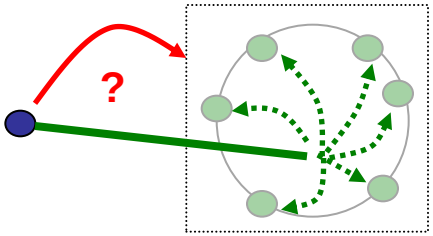
Special case: local Peer-to-Peer networks

Motivation

- Example: Friends sit in a park or classroom. They want to form a Peer-to-Peer network.
- In case of wireless networks such networks are called mobile ad-hoc networks (MANETs). Bluetooth is an example.

Solution

- Use Broadcast or Multicast mechanism of lower layers to find other peers.



Potential Bootstrapping for the example apps

A secret multicast P2P system

- Source of the multicast stream sends its IP address and a secret to invited users via some out-of-band mechanism (e.g. via a messenger)

CoolSpotsMunich P2P system

- The website that offers the CoolSpotsMunich software also operates a webservice that scans the P2P network for active rendezvous-peers.
  - The service provides some peers, users with a good connection may also declare themselve as rendez-vous peers and be added to the list.
- A peer contacts the webservice via HTTP and DNS (http://peers.coolspotsmunich.example)
- A peer contacts one of the peers, if it timeouts, another one.

Membership / Identities / Roles

Membership – when am I in the system?

- Bootstrapping → find first node
- Membership → how does a node become a part of the network?

Membership

- Once at least one link to the node exists, a node is part of the system.
  - Relates to connected components of the graph G of the system.
  - Most common model in Peer-to-Peer systems.
  - Membership management = maintaining and managing edges of the graph.
- Membership = having an ID in the system
  - Separates membership from being part of the current system graph.
  - Option: The node has valid credentials to join the system.
    - e.g. node knows password required to be let into the multicast P2P network
  - Option: Some data in the system refers to the node.
    - e.g. Skype member, but offline

Not all peers are equal – Roles in P2P networks

Possible roles (not complete)

- Peer
  - Standard role.
- Super peer
  - A quite common role. A peer that is „stronger“ and contributes more to the system by taking part in ressource-intensive critical tasks.
- Client peer
  - A peer that primarily uses the system.
- Rendezvous-peer
  - A peer that is available for bootstraps.
- Authority
  - An authority (not necessarily a peer in the system) that is trusted and provides cryptographic material for security operations.
- Initiator
  - Peer that started the system.
- Peer of instance i of P2P system
  - A system may have separated parallel instances running (e.g. various multicast groups in our multicast P2P system).

Role Assignment

Role Assignment

- Self-Assignment (by software or user)
- Assigned by other peers
- Assigned by some authority
- Role may change over time.

Criteria for Assignment

- Preferences
  - Default + User settings
- Lifetime
  - Studies show: Old nodes tend to remain longer in the network and tend to be more reliable. ➔ super peer?
- Strength
  - Enough ressources? ➔ super peer?
- Reachability
  - No firewall / NAT
- Identity
  - Some identity may determine the role, e.g. certificate by authority says super peer.
- Trust
  - A peer behaved good over time and earned reputation, use it for critical tasks?
- Random

Identities in a Peer-to-Peer network

Definition 2.1 (Identity)

- An identity or ID of an entity is an abstract representation of the entity under which it is known by other entities. The ID is usually a number or a string. An ID is expected to be unique, while the entity may have more than one ID.

Identities in networked systems

- Types of identities
  - Personal identities and pseudonyms of the user.
  - The machine has addresses on ISO/OSI layers 2 and 3.
  - The application on a machine is addressed via a layer 4 address, the port on transport layer in Internet terminology.
  - The Peer-to-Peer system on top may reuse any of these identities.
  - Cleanest solution: create system-specific identities.
- Identities are important for addressing.
  - Either `address=ID` or `address = transform (ID, …)`
    - Transform is usually a hash function.
  - Identity may be used to indicate roles and differ per role, layer, etc.
- Any entity can have an identity, also items, services, ….

Identities

Constraints on Identities

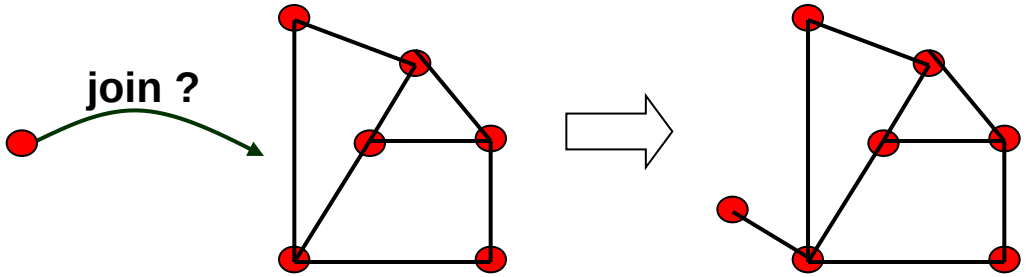
- Identities may not be free from constraints.
- Identities may have semantics.
  - e.g. identity of an item in a video sharing system
    ID = hash(”Perfect −video.mpeg”)
  - Since hashes of names and semantic descriptions are one-way, collision handling may allow multiple semantics to be used.
- There are more restricting constraints like cryptographic identifiers.
  - Such identifiers combine an identity with a cryptographic key (cf. Chapter 2). ➔ node slight adaptations or changes to ID, no other semantic assignment
- Identities in a protocol are often limited by a fixed bitlength.
  - If this limit is small ➔ important to handle collisions
  - If this limit is large and collisions are extremely unlikely ➔ one may skip the collision handling.
    - However, collisions are also a question of how identifiers are assigned. Only strategies like independent uniform random assignment guarantee that collisions are unlikely in large identifier spaces.

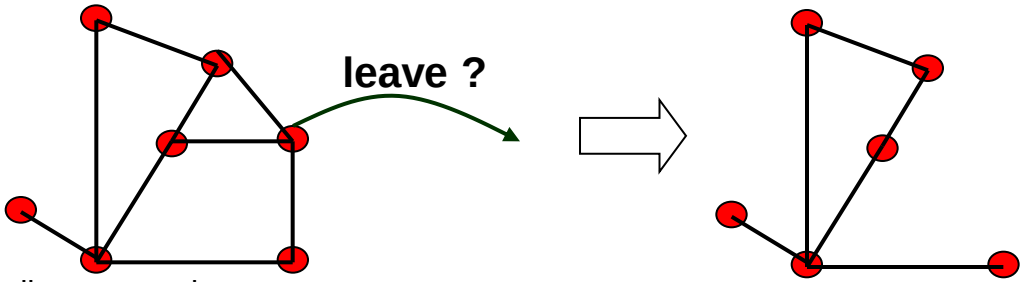
Dynamics / Maintenance

Join and Leave

P2P network is not static.

- Node joins
  - Needs to be added to the network
  - Usually via some node in the network already known (rendezvous point, list/cache of nodes)
- Node leaves
  - Important to keep the graph connected
  - Better not rely on a single node that could leave anytime



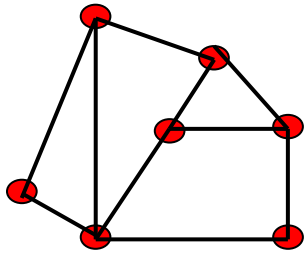


disconnected
when this node fails or leaves.

Churn / Maintenance

How to organize such a network?

- e.g. k-connected graph



2-connected -- each node can be removed without disconnecting the graph

Churn

- Churn is the rate with which nodes join and leave the system.
- `Churn = (Joins + Leaves) / T`

Common Maintenance Operations

- Use more links to overcome failures.
  - k-connected by design; additional alternative links into each direction
- Check if links are still valid.
  - Ping neighbor nodes from time to time.
  - Check and update on access, e.g. when sending message to neighbor.
- Cache nodes that were seen, though not using them atfirst.
- Store data on multiple nodes.
- Multiple nodes should be able to process a request.
- …

Maintenance of Data (how long, …?)

- Option Hardstate
  - Data is stored and maintained in DHT.
    - Requires that DHT maintenance ensures that data is not lost
  - Properties
    - Old data nees to be removed explicitly.
- Option Softstate
  - Data is stored only for a period of time and needs to be refreshed.
    - Requires that nodes refresh their data
    - Data maintenance in DHT optional
      - If refresh period is smaller than the typical time that data gets lost due to churn
  - Properties
    - Old data gets automatically removed.

Maintenance of Data (how much, multiple items,…?)

- General problem of how much data is stored
  - Peers are not trustworthy. Will a peer flood the network with data?
  - Softstate
    - Limits waste to what the peer can store within the refresh period
    - Data removed some time after malicous peer leaves
  - Storage limits hard to enforce if purely Peer-to-Peer.

- What about multiple write requests for one data item?
  - Peer PA writes item A, Peer PB writes item A, what to do?
  - Options
    - Keep all variants
      - In case of a request, the receiver needs to figure out which of the items he requested
    - First author of an item protects his item with passphrase and only nodes with the right passphrase can overwrite.
      - Not very secure
    - Data is simply overwritten
    - No new data is accepted till timeout

Operation

Common operations in P2P systems

- Find someone to
  - get something
  - use a service
  - interact
  - interact for a cooperative service or goal
  - maintain network
- Find something (item, data, information, etc.) to
  - get it
  - set it
- Interact with other nodes to cooperatively
  - provide a service
  - share resources
  - run an algorithm
- …

Programming aspects

- `String result = otherNode.getAnswer(Parameters)`?
  - Problem: Other node is not on local machine, the answer …
    - … is not directly available
    - … takes some time
    - … may not come at all
- Options
  - Wait for an answer = blocking this thread
    - Parts of the program cannot proceed
  - Skip the processing and continue the computation when answer arrives
    - Non-blocking
- Utilizing Events in programming
  - Computations with other nodes are based on events
  - Continue a computation when the appropriate event occurs, e.g.
    - Answer X received in MessageReceivedEvent
    - Then continue with `String result = X.toString()`

Will the answers still come?

- Further Problems
  - Request may get lost
  - Answers may get lost
  - Other node does not answer
  - Other node may not be existing anymore
- What to do?
  - Try again
  - When to stop?
    - A fixed number of tries
    - Timeout = give up when no event occurs in a given time interval
      - Requires Timer and TimeOutEvent
- What then?
  - Abort computation
  - Use other node if possible ➔ algorithms need to deal with this uncertainty

… in our example apps …

P2P Multicast System

- Find source for or node in multicast tree with certain ID (in case of multiple multicast P2P groups in one P2P system)
- Ask source if allowed to join.
- Position yourself / ask for position in distribution tree.
- Balancing of distribution tree.
- Forward incoming messages to children.

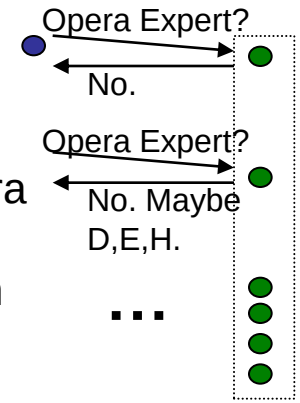
CoolSpotsMunich P2P System

- Nodes store information provided by their users to items (cool spots) they evaluated and entered.
- Find someone with items located near a given GPS coordinate.
- Find all items within a certain distance to a given GPS coordinate.
- Find nodes where user is an expert for bavarian restaurants
  - Ask these nodes about good restaurants near to GPS coordinate.

„How can we find something or someone?“

Searching

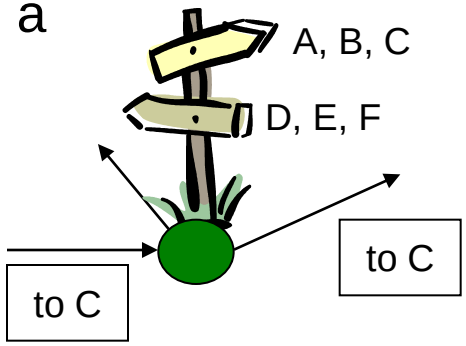
- Look for someone or something.
  - Location of destination or set of destinations is unknown.
  - Breadth-First / Depth-First Searching, Flooding, etc.
- Usually used when no criteria exists that allows to determine an entity that directly knows the answer.
  - E.g.: The requested information can be anywhere.
    - Example: The CoolSpotsMunich network may search for opera experts.
    - Example: Find a fast relay node for the corresponding position in the multicast tree (depends on the parent and children in tree, its bandwidth, etc.).
- Allows local decisions on a matching at the corresponding nodes.
  - E.g.: „Do you have operas on your disk?“ „Yes, these…/ No“



„How can we find something or someone?“

Routing

- Routing = algorithm / protocol to find a path (routing phase).
- Given a destination send a message to the target on a direct way (forwarding phase).
- Needs
  - Addresses (Names) for peers or items
  - Means to determine and know the way
- Usually operates on one parameter („address“) and distance metric on this parameter.
  - Examples for distance metric: euclidian, Hamming distance, etc.
  - Not suitable to find close nodes according to
    - other metrics
    - other parameters
- E.g.: h(Opera) = A, so send message to A for Operas.



Routing and Searching – Types of Queries

Routing

Searching

Types of Queries

- Exact
- Fuzzy Queries
  - Find somthing similar according to a metric.
- Range Queries (Bereichsabfrage)
  - Find everything in given Interval, e.g. [c,pfau] or (2,4]
- String Queries
  - substring, startsWith, endsWith, …
- Complex queries
  Find peers / items where for a node-specific and query-specific function $f_{\text{this-query}}$ holds
  $f_{\texttt{this-query}}(\texttt{candidate}) > \texttt{threshold}_{\texttt{this-query}}$
- SQL / Database queries

„Marienplatz, Munich“

„somewhere in Munich“

„everything on the way from Garching to Munich Marienplatz“

„All restaurants near Marienplatz with good food according to my taste.“