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 1.1 Basics

Dipl.-Inform. Heiko Niedermayer
Christian Grothoff, PhD
Prof. Dr.-Ing. Georg Carle

Overview

- Motivation
- Basic Terms
  - Peer-to-Peer, Overlay, Graph, Underlay
- Basic Operations
  - Example Peer-to-Peer applications
  - Bootstrapping, Membership, Maintenance

Motivation

- The Internet does not differentiate between client and server.
- Many protocols do.
  - WWW, Mail, etc.

Client/Server

- Server
  - Permanently online, provides a service, administered, provider control
- Client
  - Consumes the service, no special requirements, offline most of the time.

Internet Usage Change around 2000

- Flatrates: Private computers stay online all the time.
- Peer-to-Peer paradigm starts to exploit these resources.

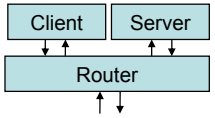
Basic Terms

Basic Terms

Terms: Peer, Peer-to-Peer

Peer

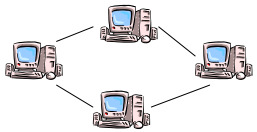
- A host in the network, usually at the edge of a network, in a private home.
- A peer operates as client and server in parallel.
  - Peers are meant to be equal, no pre-defined distiction into client, server or other roles.
  - In reality, not all peers are equal.



Peer-to-Peer

- Paradigm to provide functionality by a set of peers.
- Control is shifted from a server controlled by a single administration to a set of peers, controlled by their users.

Term: Peer-to-Peer System



Peer-to-Peer systems

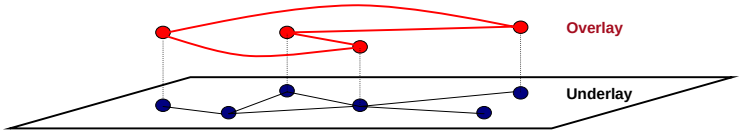
- Distributed systems that consist of equals (peers) with no predefined distinction between client and server and without dedicated servers or central authority.

Common Characteristics

- Peer-to-Peer networks are decentralized and take advantage of resources at the edge of the Internet, in particular the computers of users.
- End systems do not primarily serve the purpose of the Peer-to-Peer system. → Their resources must not be exhausted by the Peer-to-Peer network
- Computers are not always-on. → Environment is less stable and more dynamic than in the traditional client-server case.

Term: Overlay

"'Overlay networks is a term for networks that run on top of an existing infrastructure but provide certain additional functionality."' (Source: www.overlay-networks.info)

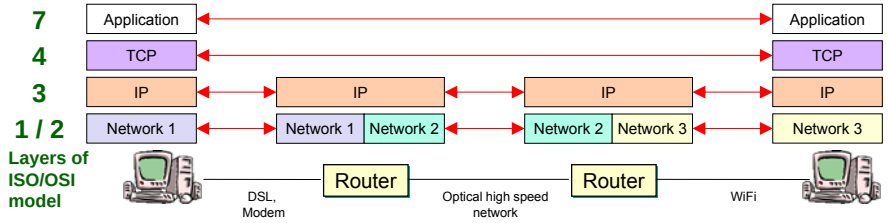


Overlay networks

- add another layer of abstraction in some place in the protocol stack.
- introduce a new structure of who is connected with whom.
- This new structure uses the connectivity of the lower layers for its links. The layer below the overlay is called underlay.
- Overlay networks are usually formed for some kind of reason, usually to provide a desired additional functionality.
  - e.g. send a message to all members of a group (multicast), the overlay represents the group and provides group membership functionality

IP – an overlay in the network stack

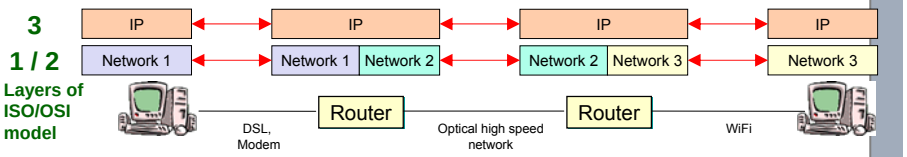
The most well-known overlay is the Internet Protocol (IP) itself.



The Internet Protocol (IP)

- abstracts from the technology and physical location present in the lower layers (Medium access, Physical Layer).
- forms a structure that is optimized to provide connectivity between all connected networks and their entities, no matter where they are and what access technology they use.

IP – an overlay in the network stack



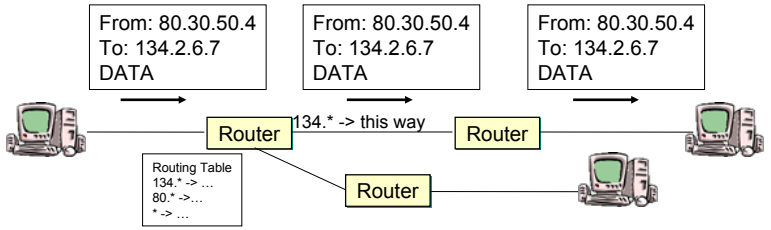
- The problem to route a packet to its destination is usually separated into the *routing* and the *forwarding*.

Routing

- Operation to determine optimal paths and structure in a network.
- Method: Routers run a routing protocol and a shortest path algorithm to determine the best paths to all reachable destinations in the network.
- Result: Routing table with entries (destination, next hop towards the destination) The next hop is a neighbor that can be reached with the local layer 2 protocol (e.g. ethernet).

destination	next hop
131.*.*.*	115.3.1.2
80.12.*.*	114.3.1.2
…	…

IP – an overlay in the network stack



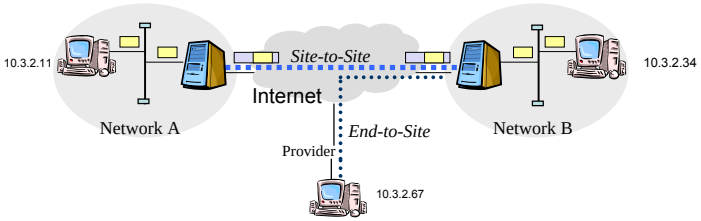
Forwarding

- Operation of the router when it sends a packet to its next hop according to routing table.
- Situation: a packet arrives at a router.
- Response: router checks its routing table for entries that match with the destination given in the packet.
  - Use the best next hop specified in the corresponding entry of the routing table.
  - If the target does not exist, some router on the path will recognize this and return an error message (ICMP protocol).
- Necessary: For any packet (IP datagram) with any arbitrary target any system (router) on the way can efficiently decide where it has to go.

Examples for Overlay Networks

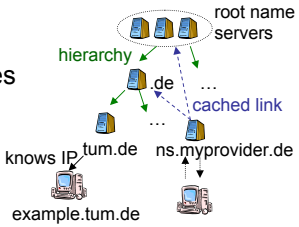
Virtual Private Networks (VPNs)

- used to connect local networks that are located in different places using other networks, e.g. the Internet or phone lines.



Domain Name System (DNS)

- used to map names of resources to IP adresses
- DNS servers internally form an overlay
  - Hierarchy with a group of root servers to more and more specialized servers for subdomains.



Peer-to-Peer or not Peer-to-Peer

Auctions / eBay

- Peer-to-Peer
  - Money and goods exchange (nothing to do with the network)
- Not Peer-to-Peer
  - The platform itself (Auctions, Accounts, Information transfer) and its Information Management

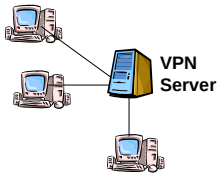
Skype

- Peer-to-Peer
  - Lookup, User Interaction, Data Exchange
- Not Peer-to-Peer
  - Login, Account Management

Many Peer-to-Peer systems are not purely Peer-to-Peer.

Overlay vs. Peer-to-Peer

- Overlay and not Peer-to-Peer ?
  - Virtual Private Networks with VPN servers.

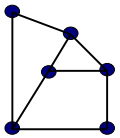


- Peer-to-Peer system and no Overlay ?
  - Usually Peer-to-Peer systems create a new structure (overlay) on top of an underlying network.
    - No perfect examples without overlay.
  - Possible examples
    - Peers in an ad-hoc or sensor network may not add a new structure with new identities.
    - Peers in a LAN playing a P2P game use IP.
    - Students in a lecture organize where they sit, etc. However, again no new adressing or communication structure.
    - …

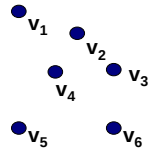
Spontaneous meeting, no central coordination

Some terms from Graph Theory

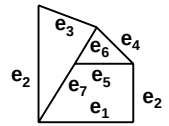
- Graph G=(V,E)
- Vertex set V = {v_1, v_2, …v_n}
  - We usally say **nodes.**
  - n = |V|
- Edge set E = {e_1, e_2, …e_m}
  - We usually say **links.**
  - m = |E|
  - Can have attributes like distance, etc.



Graph G



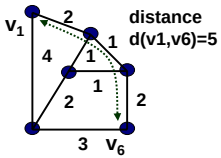
Vertex set V



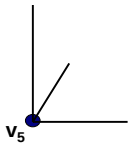
Edge set E

Some terms from Graph Theory

- Distance d(i,j)
  - Shortest path between nodes v_i and v_j
- Diameter D of G
  - Longest distance in graph G
- Degree
  - Node degree = number of edges adjacent to node
  - Degree of a graph = max. node degree
- A graph is connected if there is a path from any node in the graph to any other node in the graph.
- A graph is k-connected if any k-1 nodes can be removed without causing the resulting subgraph to become disconnected.

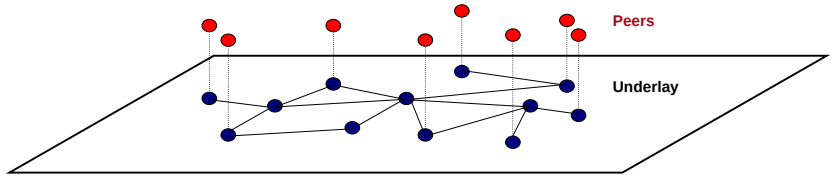


diameter(G)=5



degree (v_5) = 3

Peers and Underlay



Underlay

- Provides connectivity between all peers in the Peer-to-Peer network (overlay).

Peers V = {v_1, v_2, …v_n}

- Peers are the nodes of the graph G.
- Peers may have a name (identities are usually necessary).
- The set of edges E needs to be created by the Peer-to-Peer algorithms.
  - The graph needs to be connected.
  - The structure should be good for the purpose of the Peer-to-Peer system.

Complexity / Performance

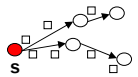
- Let us consider a graph with n nodes.
- Complexity metrics
  - Number of links m = |E| in graph
  - Maximum/Average State of a node
    - Degree of a node, Routing table size, Non-graph-related information
  - Number of Responsibities (e.g. number of files provided to network)
  - Traffic through a node / Number of Requests
  - Maintenance Traffic / Overhead (amout of traffic vs maintenance traffic)
  - …
- Performance metrics
  - Diameter or average distance in Graph
  - Robustness / Connectedness of a graph
  - Performance / Quality of application-specific operations
  - ….

Basic Operations

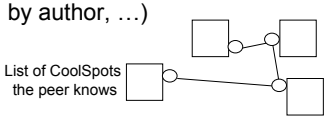
Basic Operations

Example Applications

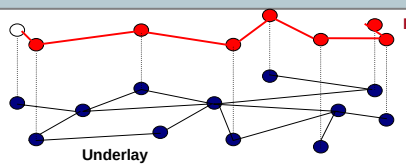
- The examples on this slide do not correspond to real networks, but they are used to illustrate possible solutions through the P2P part of this lecture. So, we will build them differently from section to section.
- A Multicast P2P Network
  - Multicast = 1 sender, many receivers
  - A source node s 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.

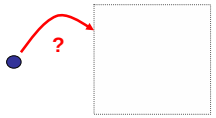
Basic operations

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

Bootstrapping

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?

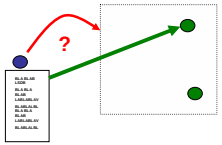
Necessary

- 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.

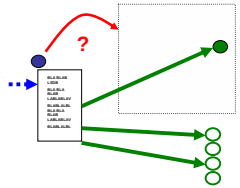
How to find the Peer-to-Peer overlay?

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

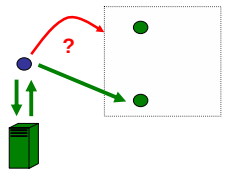


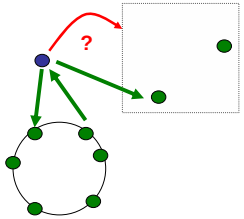
- Cache with nodes
  - Method
    - Store nodes you have seen. Contact some of them in order to join.
  - Assumption
    - Many nodes stay long in the network and can still be contacted the next time.
  - Cannot solve the initial contact problem (→ no cache yet).
  - Many nodes are stored anyway during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

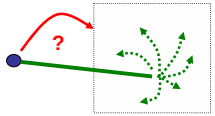
Special case: local Peer-to-Peer networks

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 send its IP address and a secret to invited users out-of-band (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 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.

Bootstrapping examples in real P2P sytems

Edonkey / eMule

- Via node lists (called „server list“, „server.met“ files)
  - Downloaded from somewhere or stored on hard disk
- Via user input / Kad also uses a node cache.

BitTorrent

- Problem is to find a torrent / tracker with the file of interest.
  - No automatic solution.
- Tracker knows active nodes.
  - Tracker known from .torrent file.

Skype

- Host cache: list of cached super peers
- Fixed set of bootstrap super peers (7 in 2004) – in the host cache after installation

Membership / Identities / Roles

- 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.
- Virtual Membership as requirement to participate.
  - 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

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

- Self-Assignment
  - By software or by 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?
- 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.

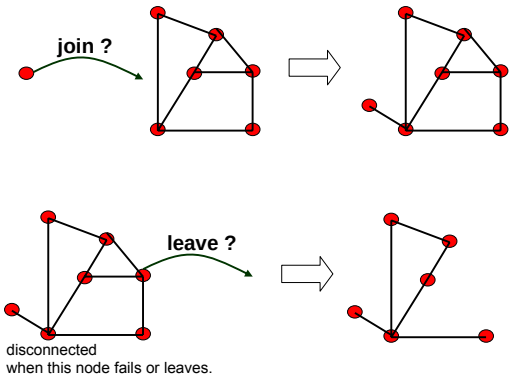
Dynamics / Structure / Maintenance

Dynamics / Structure / Maintenance

Join and Leave

P2P network is not static.

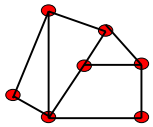
- Node joins
  - Needs to be added to the network
  - Usually via some node in the network already known (rendezvous point, list/cache of nodes)
- Node leaves
  - Important to keep the graph connected
  - Better not rely on a single node that could leave anytime



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.
- Other metric for churn: `Halflife(system, n nodes) = min(time until n new nodes joined, time until n/2 of the nodes left)`

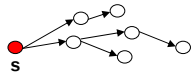
Common Maintenance Operations

- 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 and operation.
- …

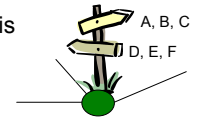
Structure

Structure

- The operations of the P2P system have an impact on the graph of the Peer-to-Peer system.
  - e.g. connect to at least 10 nodes, each from a different IP range.
  - e.g. connect to at least 10 nodes with similar name.
- Structure for application workflow
  - The application requires nodes A, B, C to process messages in that order in order to provide one of its services.
  - e.g. Multicast Distribution Tree



- Structure for routing
  - The graph and local state at each node is used to create a routing table.



Operation

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
- …

… 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

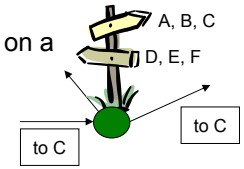
- Nodes store information provided by their users to items (cool spots) they evaluated and entered.
- Find someone with items located near a given GPS coordinate.
- Find all items within a certain distance to a given GPS coordinate.
- Find nodes where user is an expert for bavarian restaurants
  - Ask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

Routing

„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 metric
    - other parameter

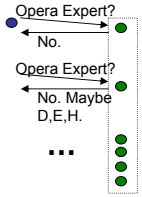


Searching

„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 a best direct way.
  - The requested information can be anywhere.
    - Example: The CoolSpotsMunich network may not have a routing to directly find opera experts.
    - Example: A fast relay node for the corresponding position in the multicast tree (depends on the parent and children in tree, its bandwidth, etc.).
  - Also holds when some nodes could be discarded as unqualified (cannot match the search due to some reason).
    - e.g. no opera expert as in the example
  - Allows local decisions on a matching at the corresponding nodes.



Routing and Searching – Types of Queries

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
  $f_{\text{this-query}}(\text{candidate}) > \text{threshold}_{\text{this-query}}$
- SQL / Database queries