Karlsruher Institut für Technologie

Fast but economical: A simulative comparison of structured peer-to-peer systems

Ingmar Baumgart, Bernhard Heep

Institute of Telematics, Karlsruhe Institute of Technology (KIT), Germany

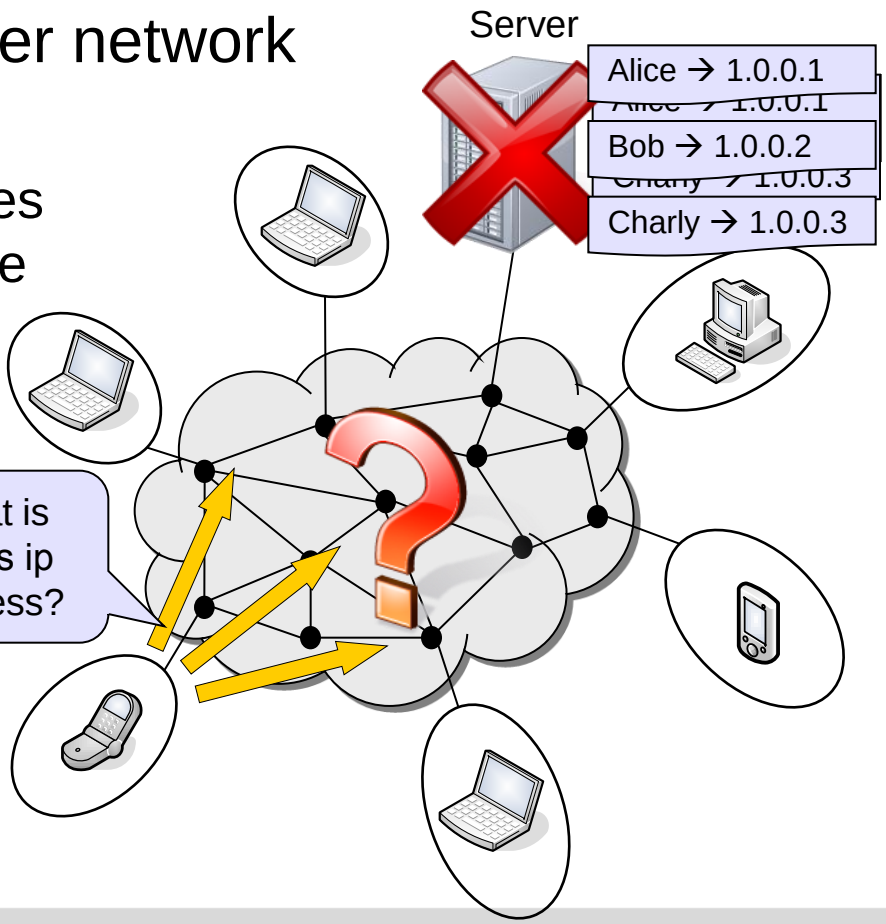
Institute of Telematics, Department of Informatics

Motivation: peer-to-peer systems

- Peer-to-peer systems are ready for production use
  - E.g. BitTorrent, Skype, Sopcast
- Benefits of using a peer-to-peer network
  - No costs for operating servers
  - Each additional users contributes his own resources to support the network (scalability)
  - Decentralized architecture increases reliability (no single point of failure)

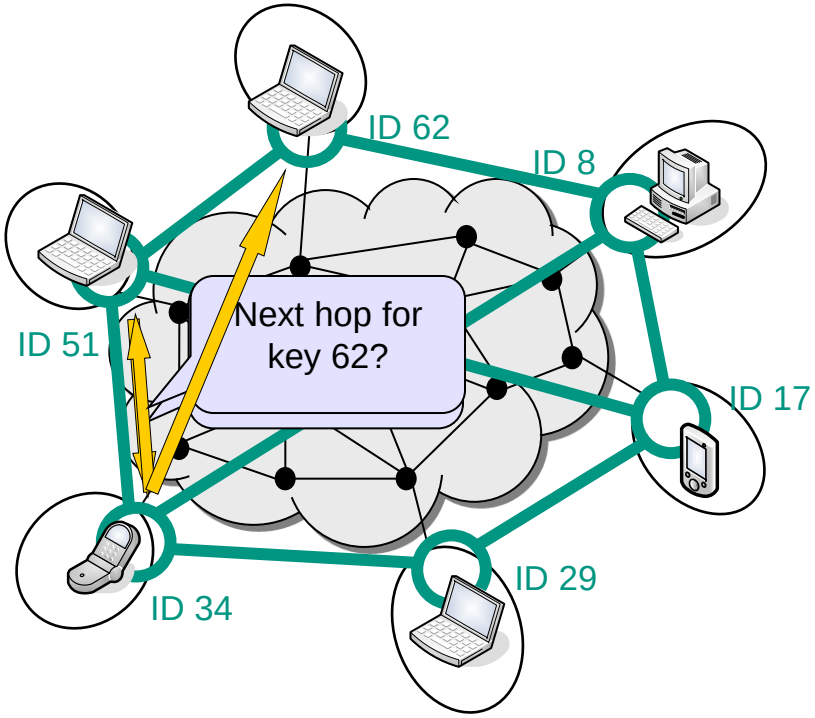
Motivation: peer-to-peer systems

- Peer-to-peer systems are ready for production use
  - E.g. BitTorrent, Skype, Sopcast
- Benefits of using a peer-to-peer network
  - No costs for operating servers
  - Each additional users contributes his own resources to support the network (scalability)
  - Decentralized architecture increases reliability (no single point of failure)
- Current example: IETF is about to specify a protocol (RELOAD) for peer-to-peer voice over IP



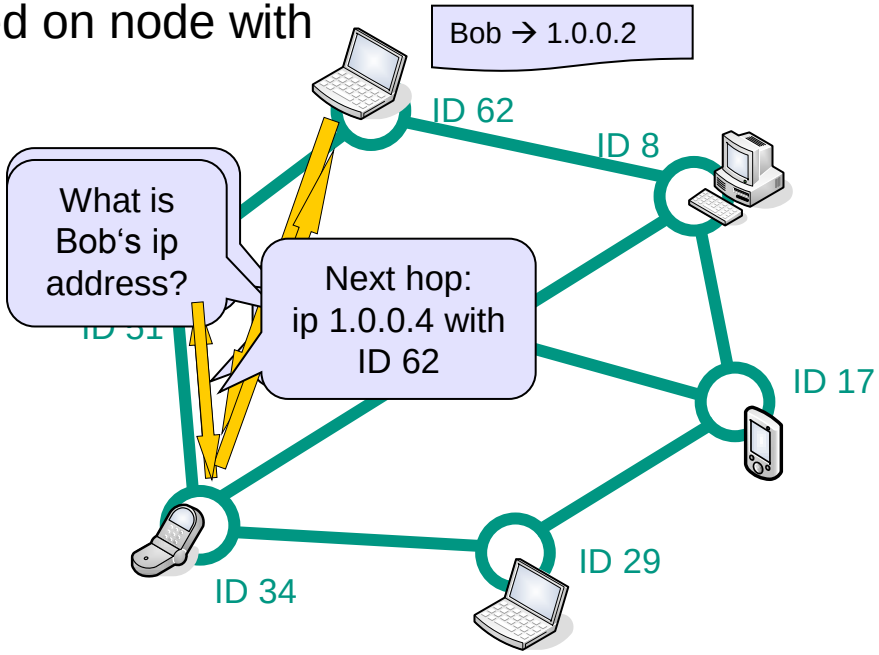
Structured peer-to-peer networks

- Distributed data storage usually done with structured peer-to-peer networks :
  - Basic service: key-based routing (KBR)
    - Logical overlay topology on top of a physical underlay network (e.g. the Internet)
    - Overlay routing table containing neighbors in overlay topology
    - Each node has a unique overlay address (*nodeId*)
    - Efficient lookup of nodes on basis of their nodeIds



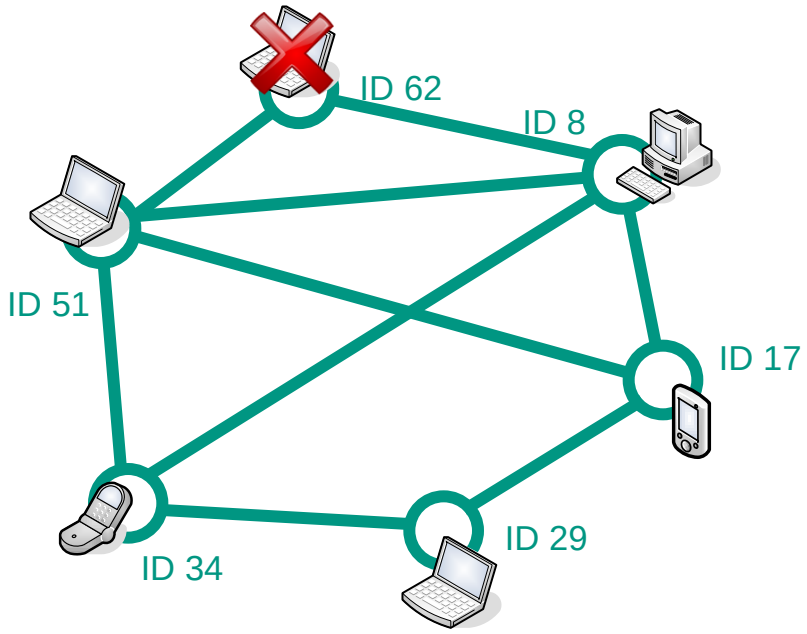
Structured peer-to-peer networks

- Distributed data storage usually done with structured peer-to-peer networks :
  - Basic service: key-based routing (KBR)
  - Date storage on top of KBR: distributed hash table (DHT)
    - Data item *(key, value)* is stored on node with closest nodeId to *h(key)*
    - Example
      - *h*(„Bob“) = 61
      - KBR lookup for key 61 leads to responsible node with nodeId 62



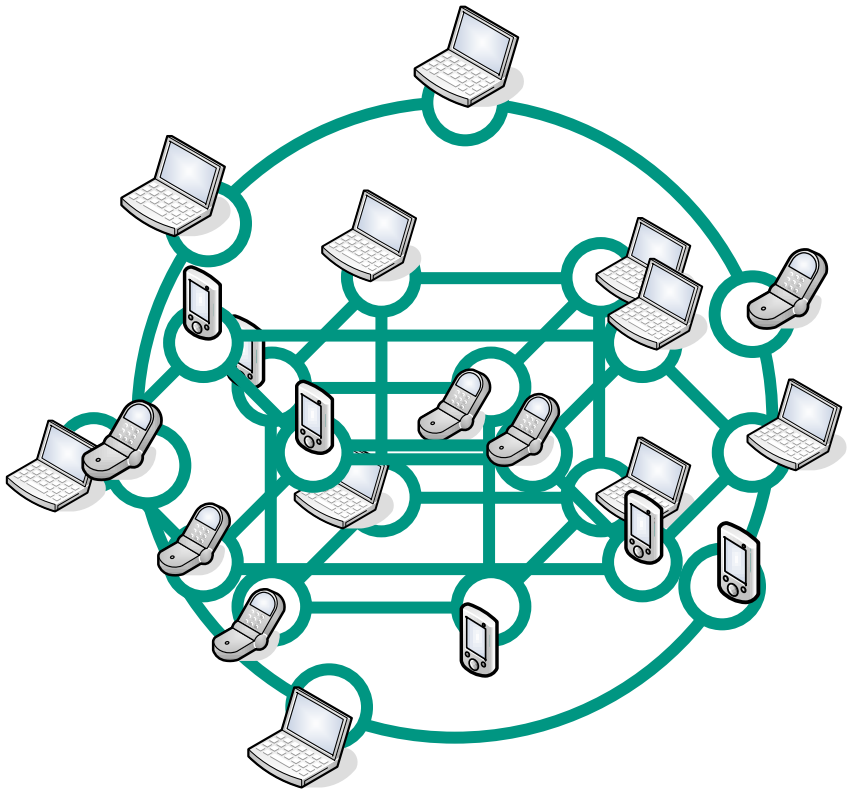
Challenges with peer-to-peer networks

- How to achieve a reliable and efficient service with peers continuously joining and failing („churn“)?



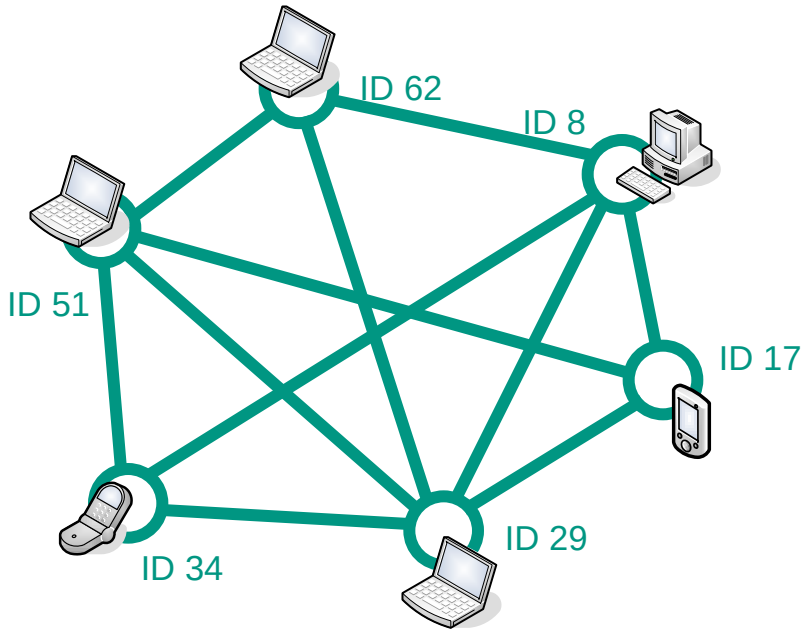
Challenges with peer-to-peer networks

- How to achieve a reliable and efficient service with peers continuously joining and failing („churn“)?
- Many proposals for KBR protocols during the last 10 years (e.g. *Chord, Pastry, …*) – how to choose the “right” protocol?
  - Differ e.g. in overlay topology and stabilization mechanisms



Challenges with peer-to-peer networks

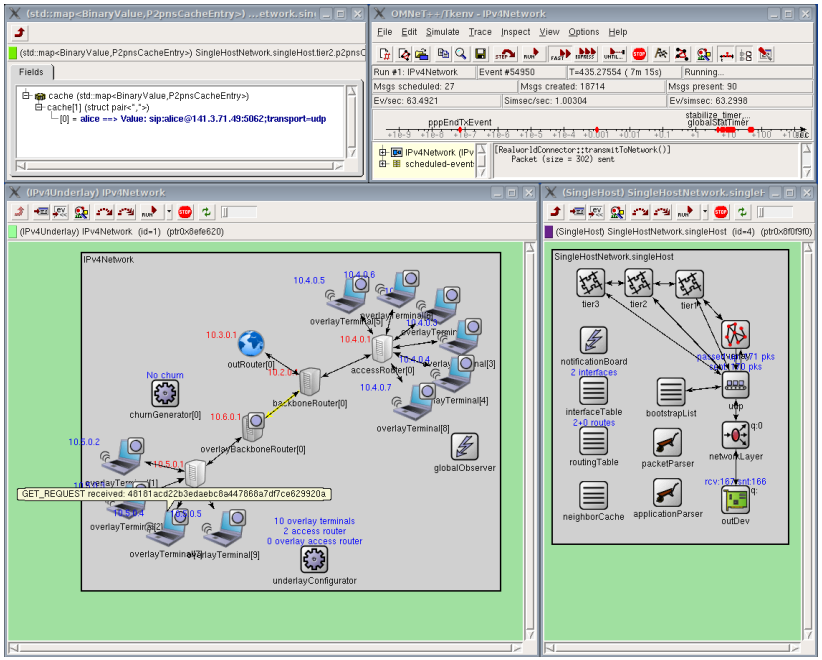
- How to achieve a reliable and efficient service with peers continuously joining and failing („churn“)?
- Many proposals for KBR protocols during the last 10 years (e.g. *Chord, Pastry, …*) – how to choose the “right” protocol?
  - Differ e.g. in overlay topology and stabilization mechanisms
- How to choose KBR parameters?
  - Many protocol parameters that influence each other
    - Size of routing table
    - Stabilize intervals
  - Trade off between delivery ratio, latency and communication costs



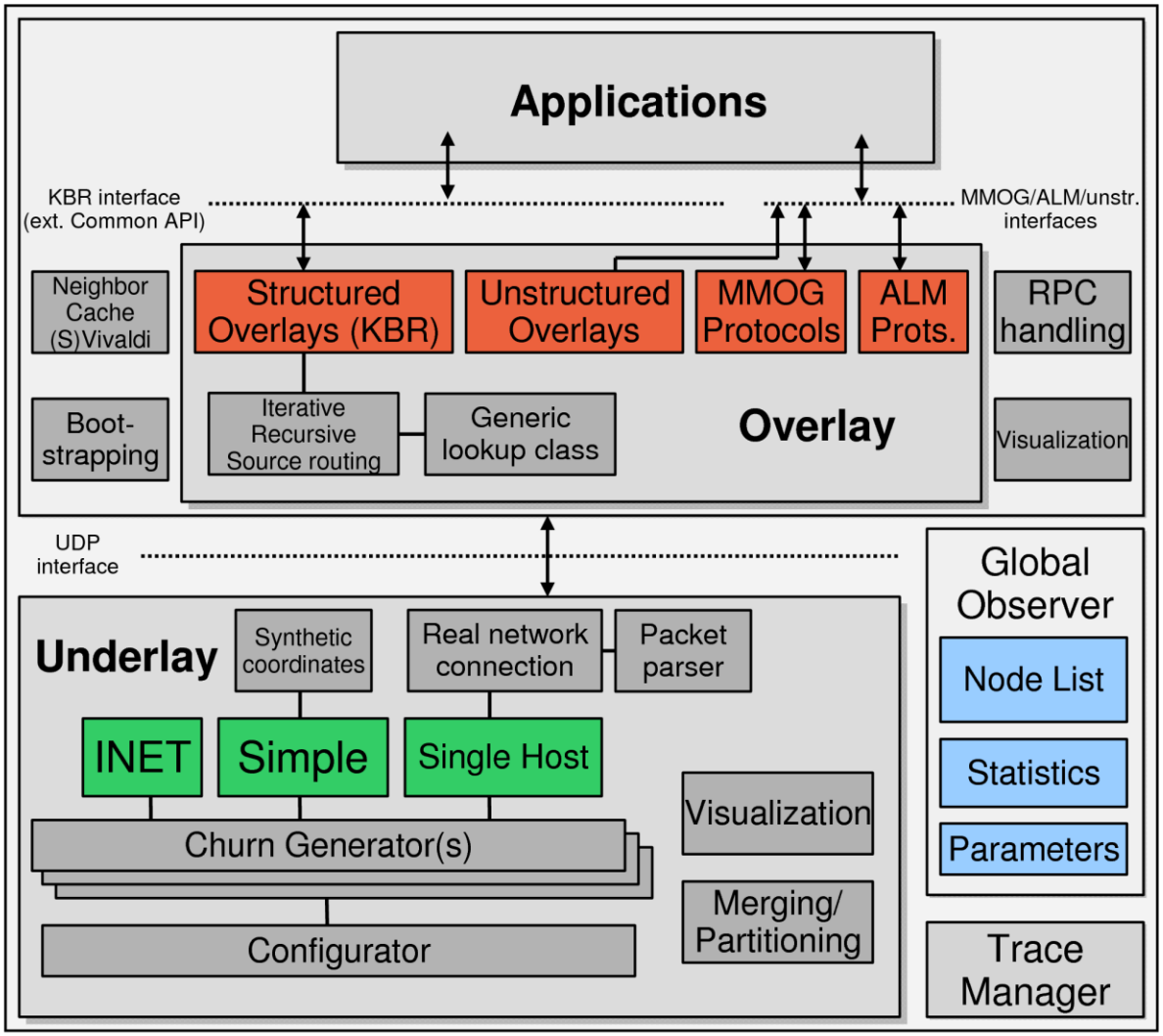
Overlay-Framework OverSim

Our overlay frameworks **OverSim** based on OMNeT++

- Supports simulation as well as emulation of overlay protocols
- Scalable (>100,000 nodes) and flexible my modular architecture
- Graphical user interface
- Large number of protocols and applications already implemented
- Open source project actively used by research community
- Separation of common functions to support fair comparison

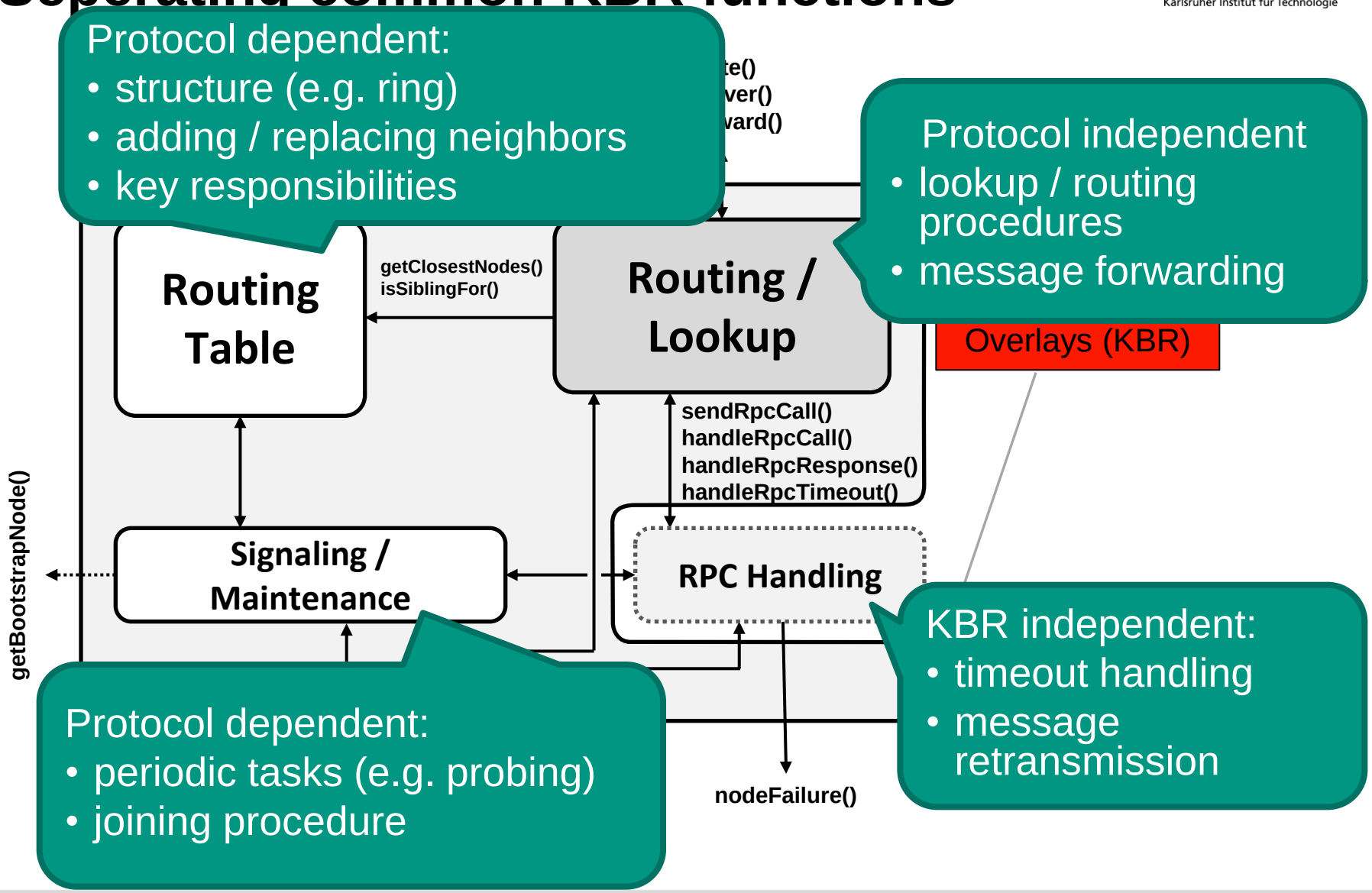


OverSim: modular architecture



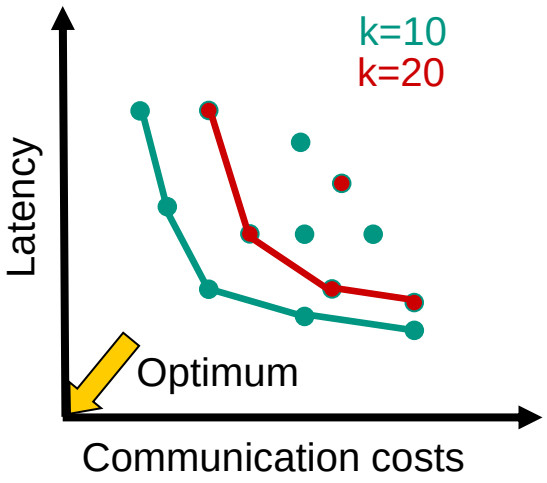
MMOG:	Massively Multiplayer Online Gaming
ALM:	Application Layer Multicast
KBR:	Key Based Routing
RPC:	Remote Procedure Call

Seperating common KBR functions



How to choose KBR parameters?

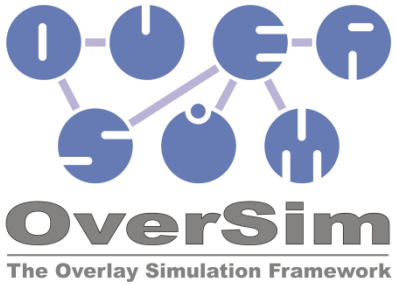
- Trade-off between
  - Costs: Bandwidth per node
  - Performance: Delivery ratio and latency
  - → Multiple-criteria optimization problem with penalty function:
    - For failed lookups we add a penalty of 10s and optimize for latency only
- Several simulation runs with varied parameter values
  - Each parameter combination leads to a single data point in the plot
  - Convex hull shows meaningful parameters
- How to compare KBR protocols?
  - Same methodology



[1] J. Li et al, "A performance vs. cost framework for evaluating DHT design tradeoffs under churn", Infocom 2005

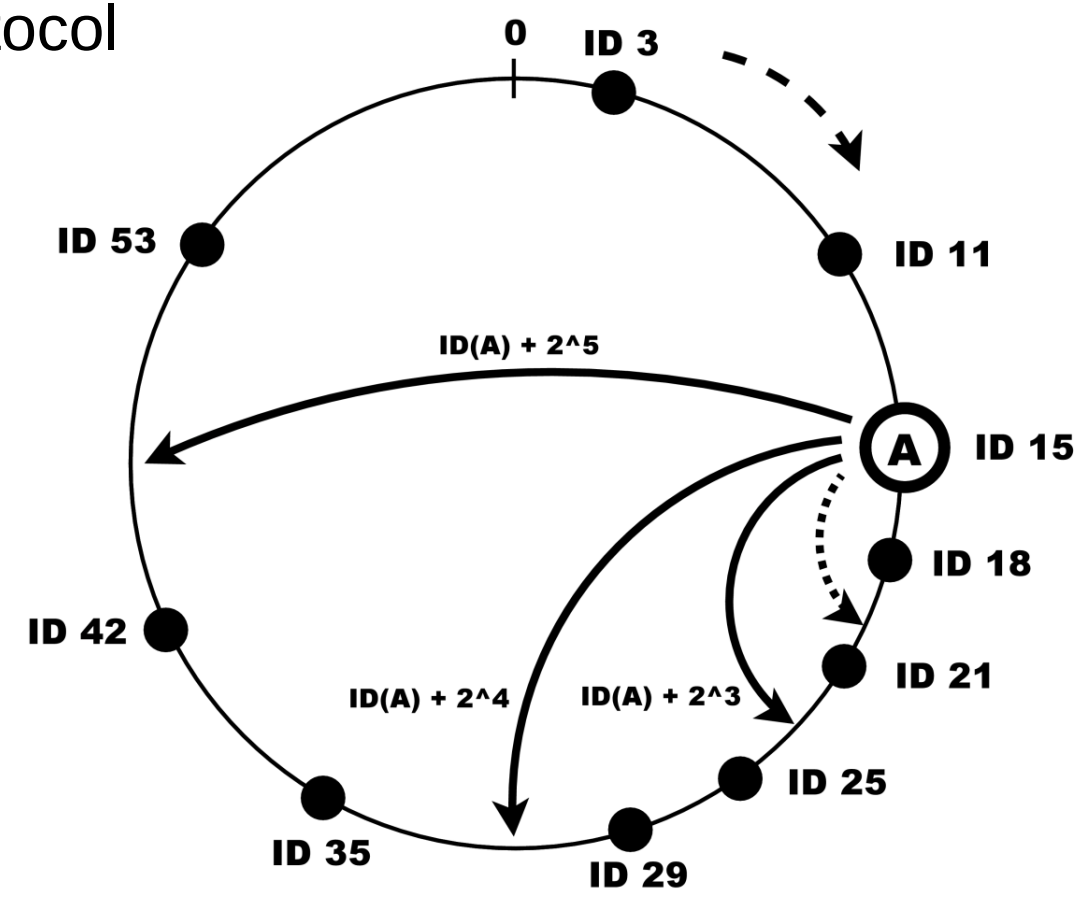
Simulation setup

- Standard scenario
  - 10,000 nodes
  - Churn: Weibull distributed lifetime of 166 min mean and *k*=0,5 (observed file sharing networks)
  - Underlay model using typical Internet latencies (calculated using synthetic coordinates)
  - Overlay protocols: Chord, Koorde, Pastry, Bamboo, Kademlia and Broose all with 160 bit nodeIds
  - Test application on each node performs periodic lookups
- Additional scenarios with different
  - Number of nodes
  - Churn rates

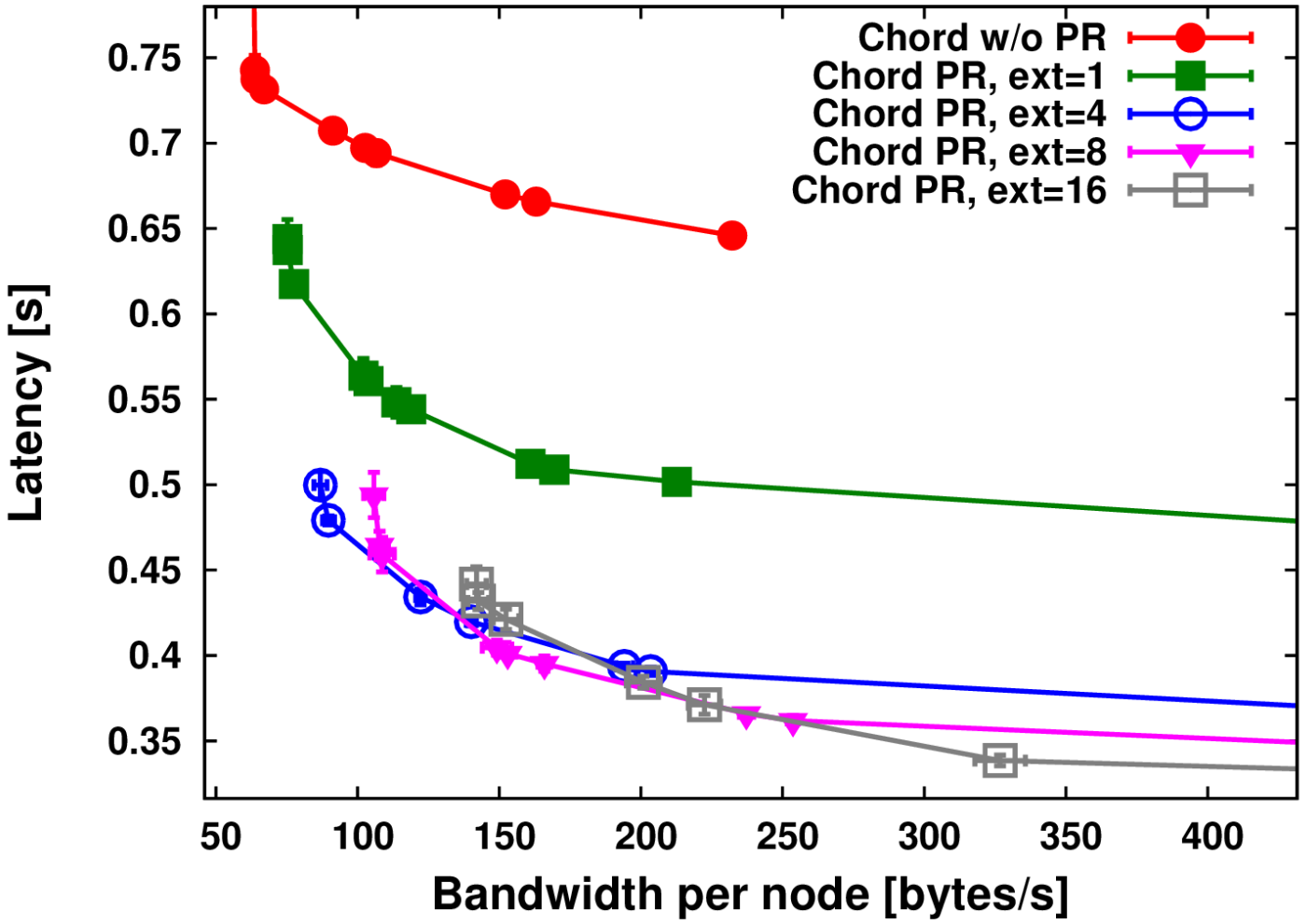


Example: Chord

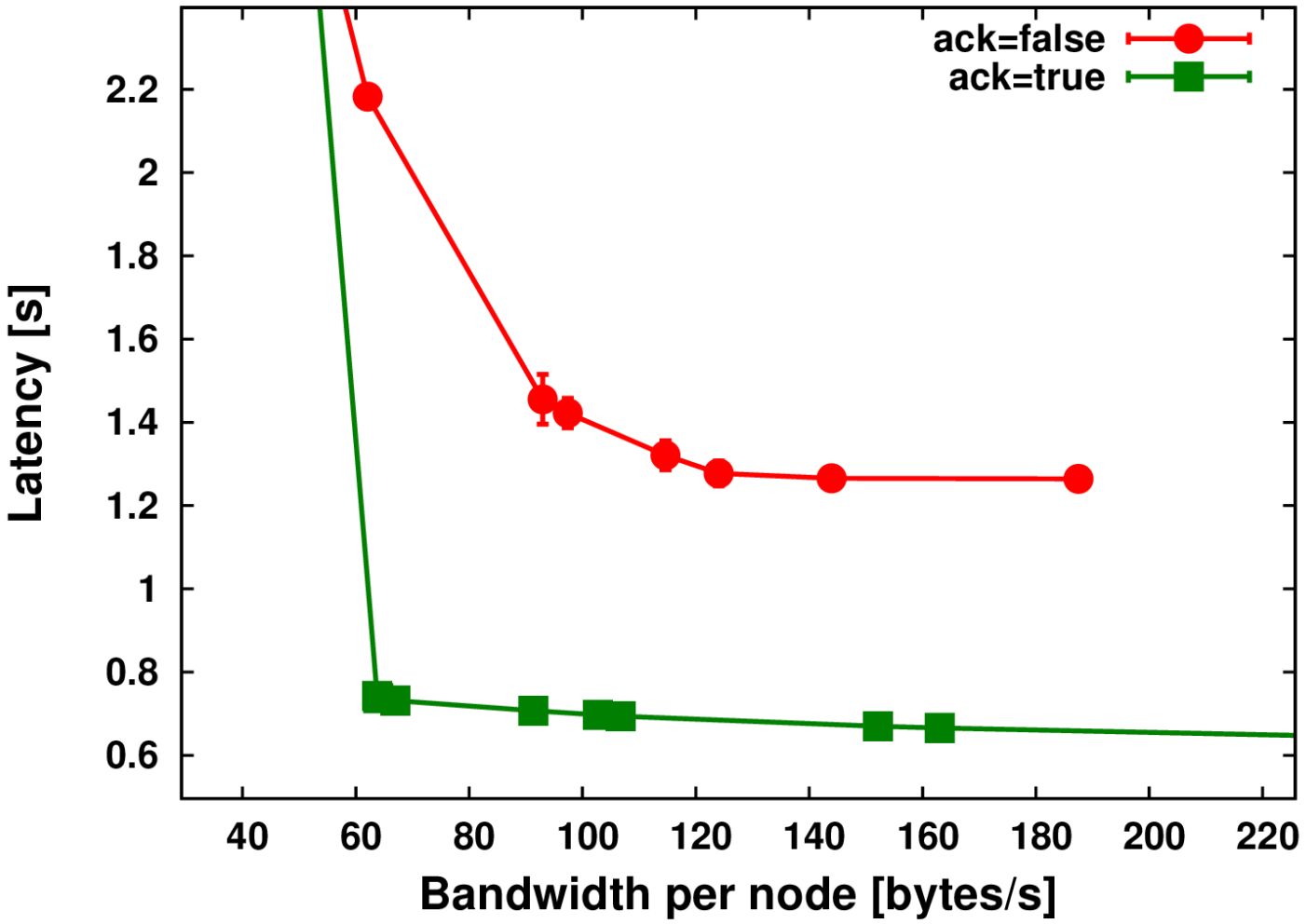
- Very popular KBR protocol
- Logical ring
  - Sucessor list
  - Predecessor
- Finger table
  - log(N) fingers
  - Optional: Maintain more than one node for each finger
- Periodic stabilization
  - Notify / stabilize



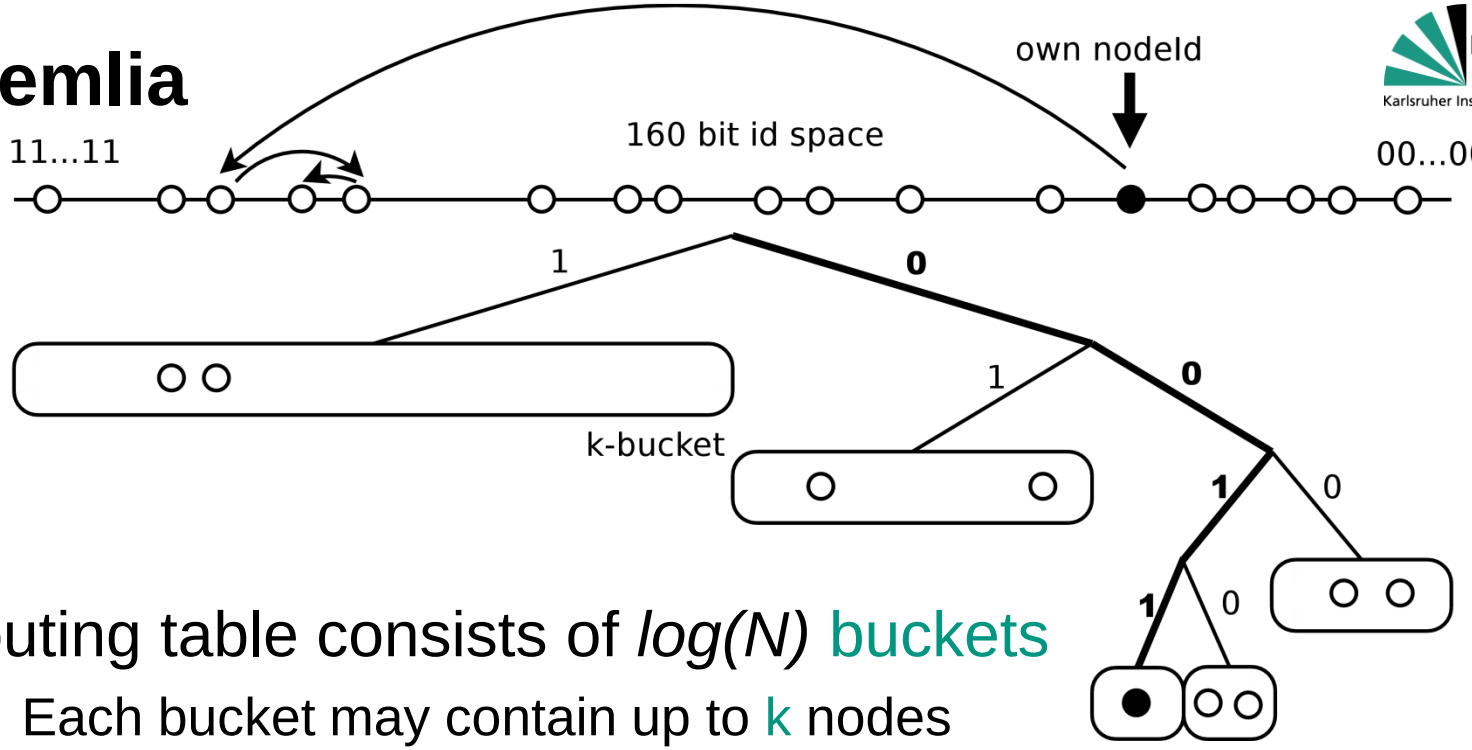
Chord: Extended fingertable



Chord: Acknowledgements (recursive)

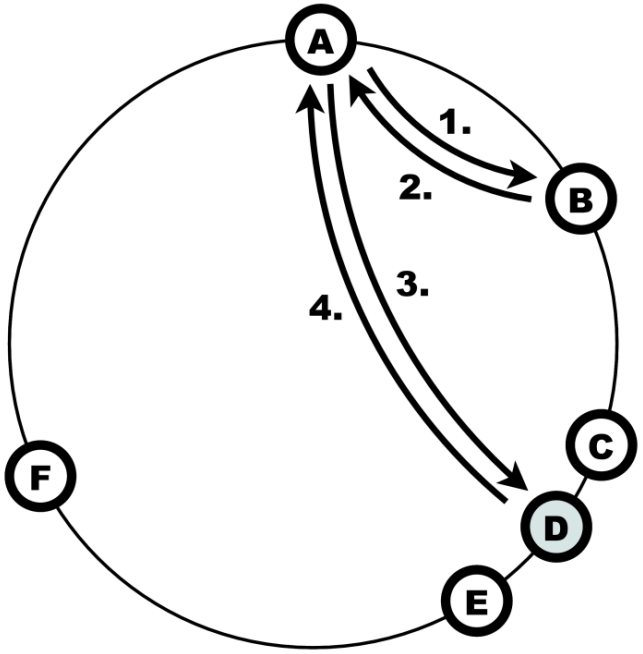


Kademlia

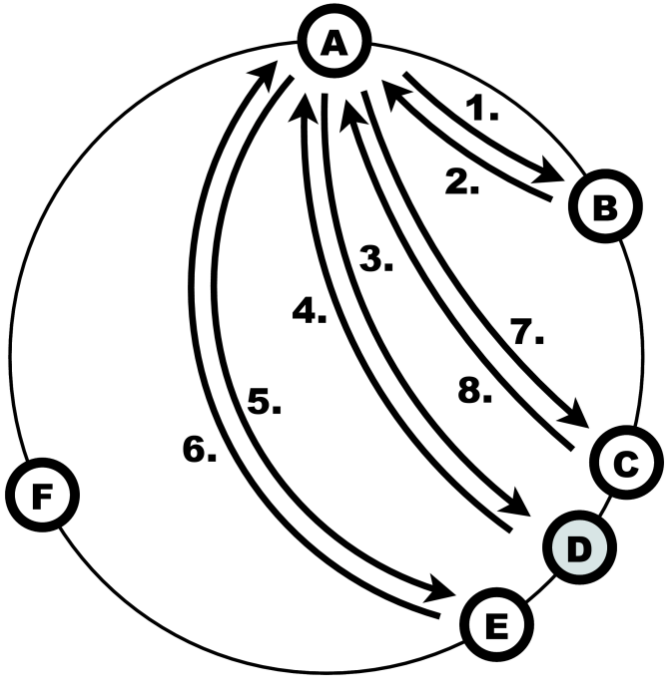


- Routing table consists of *log(N)* buckets
  - Each bucket may contain up to k nodes
- New nodes are learned passively from application lookups
- Exhaustive iterative lookups
  - Originator queries closest nodes for their *k* closest nodes
  - May be done in parallel
  - Terminates, if all *k* closest nodes have been queried and lookup process has stalled

Routing modes: iterative

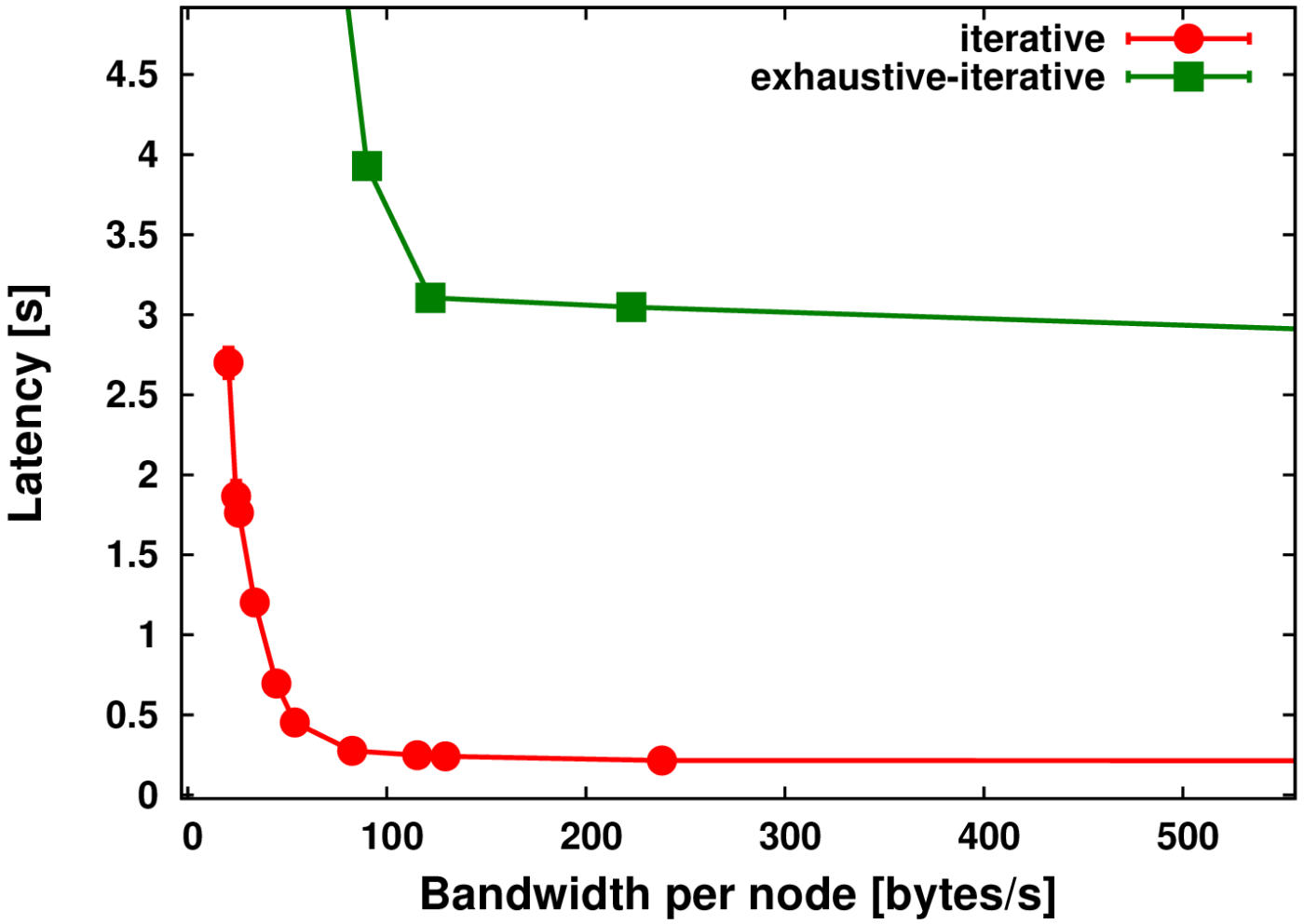


iterative

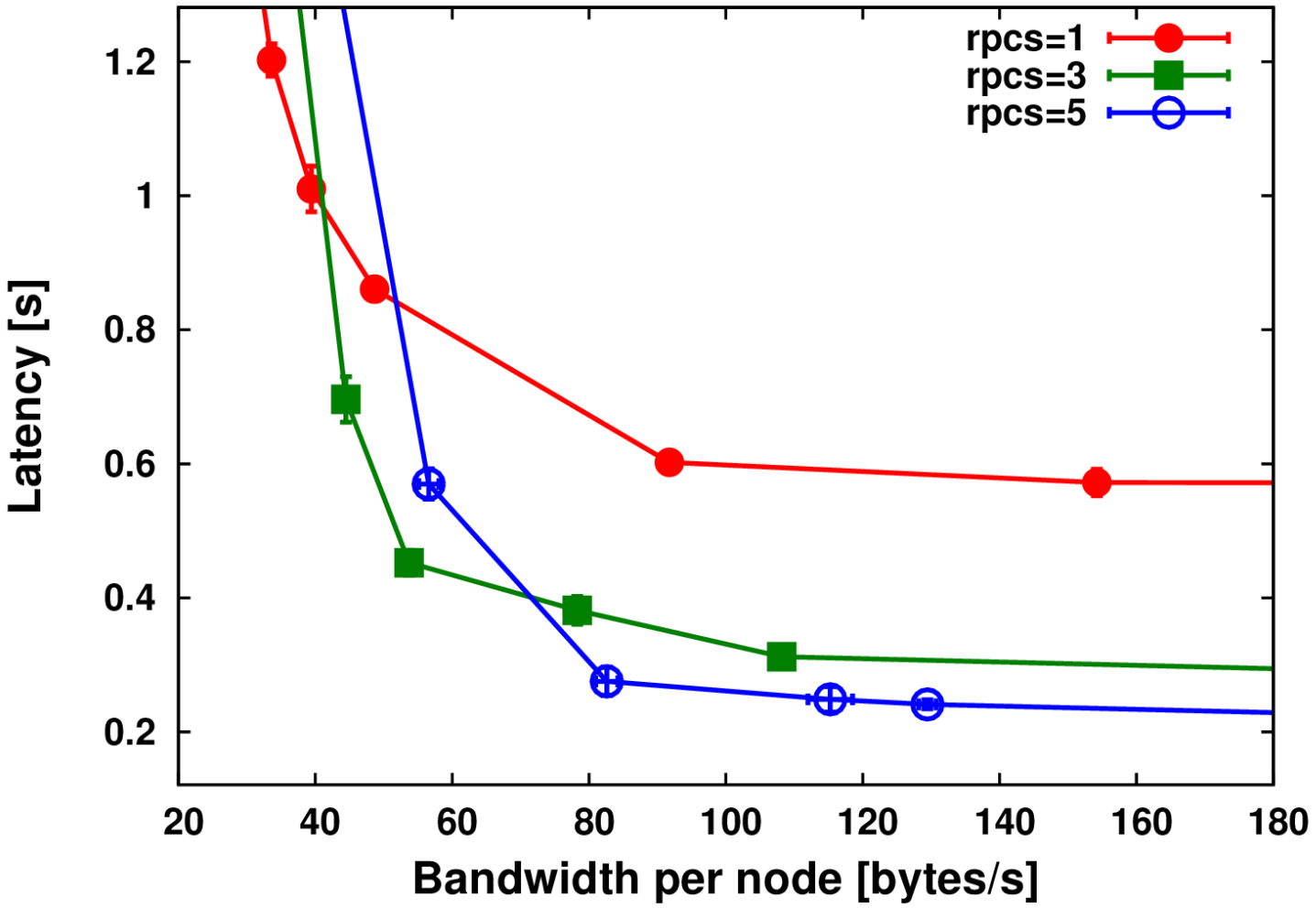


Exhaustive-iterative
(Kademlia publication)

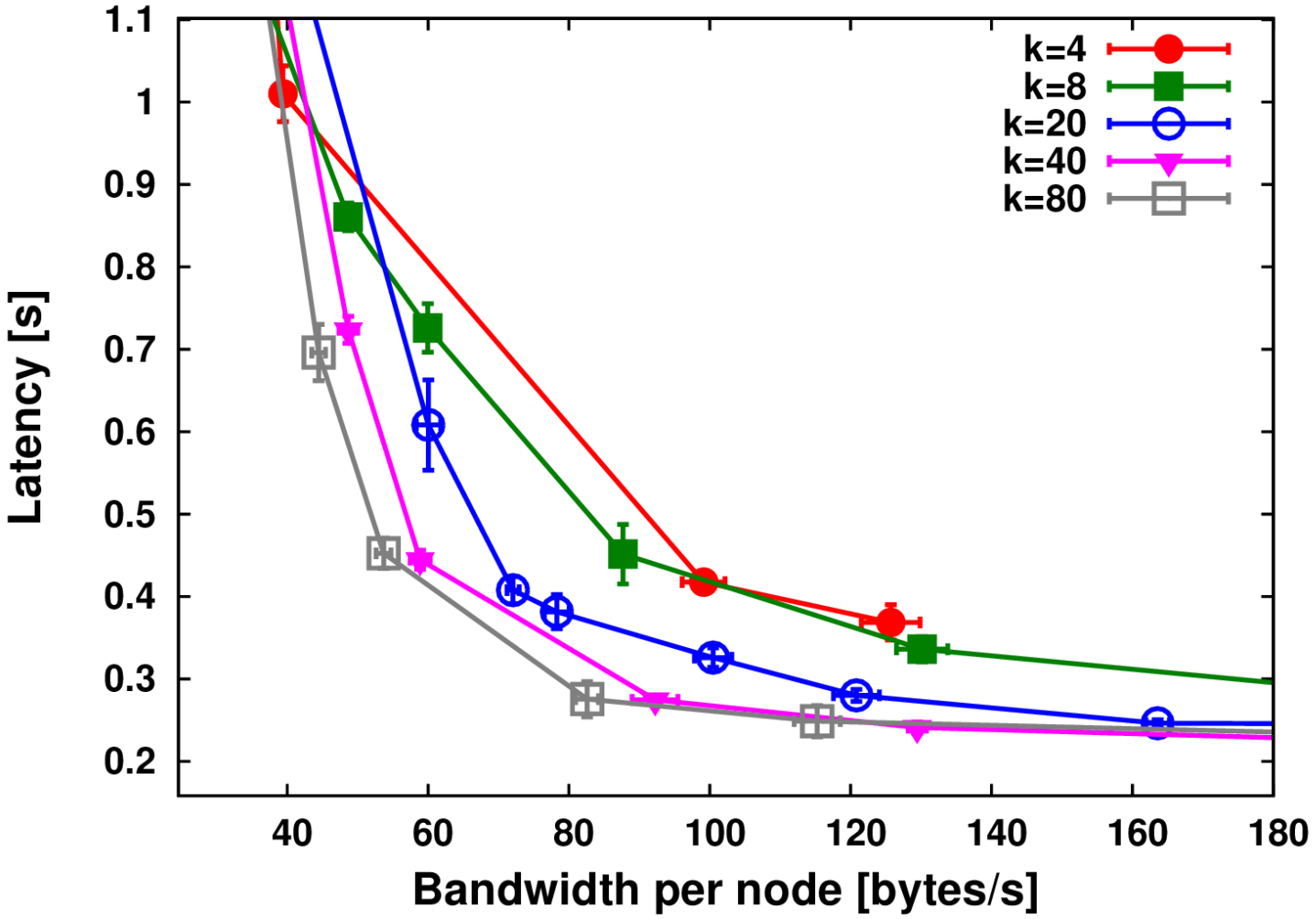
Kademlia: Don‘t use exhaustive-iterative



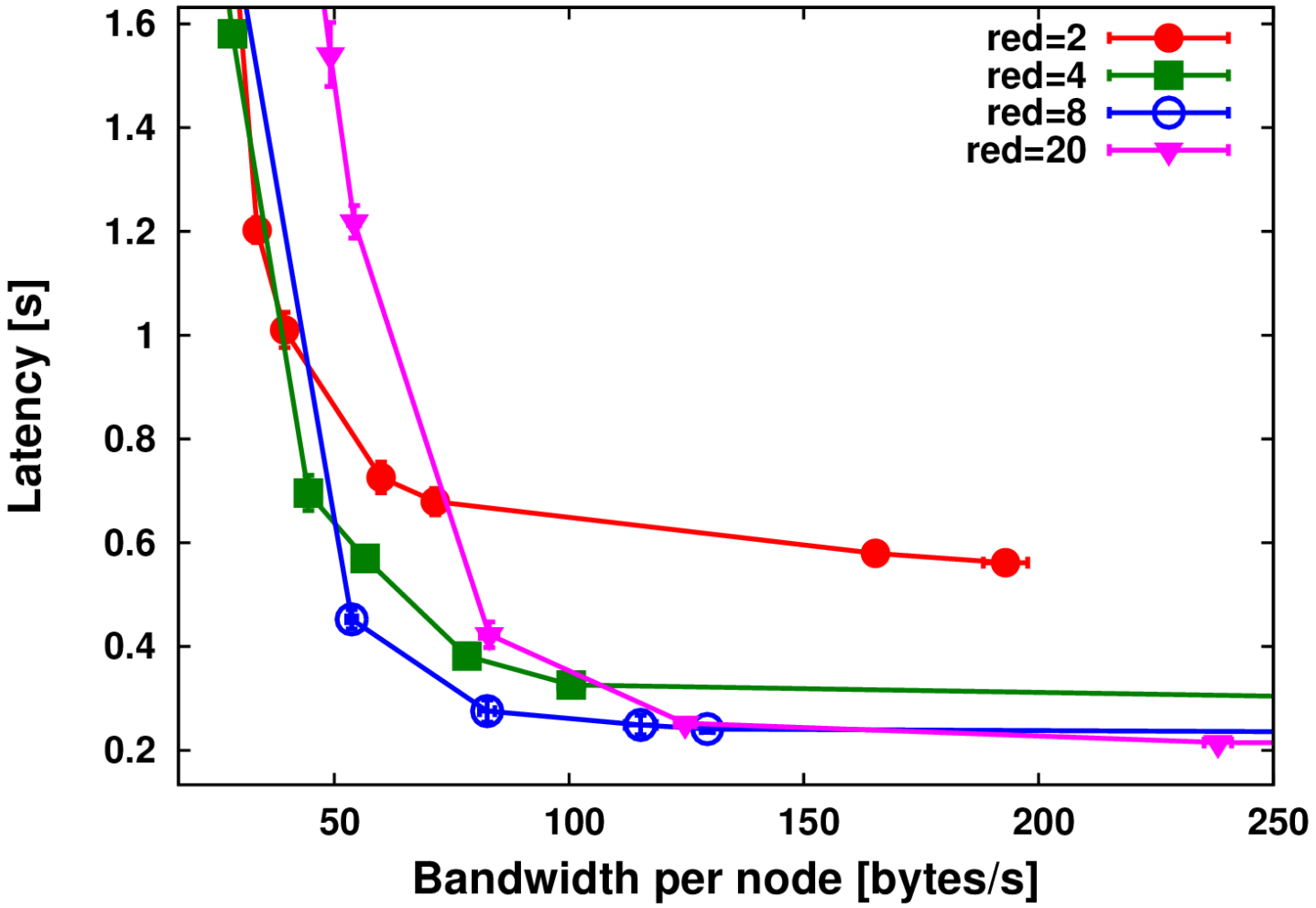
Kademlia: Parallel RPCs



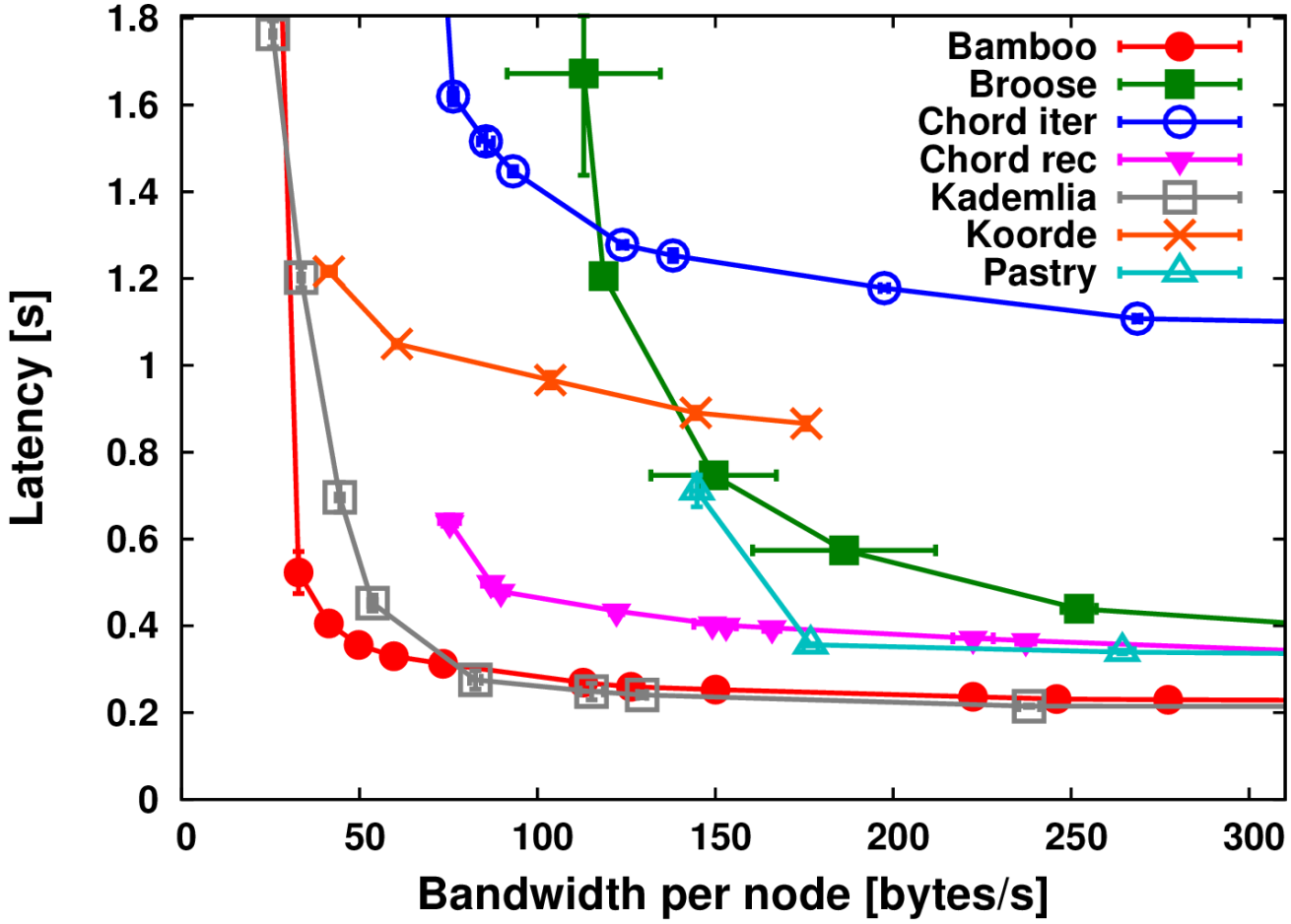
Kademlia: Bucket size *k*



Kademlia: Redundant nodes per hop

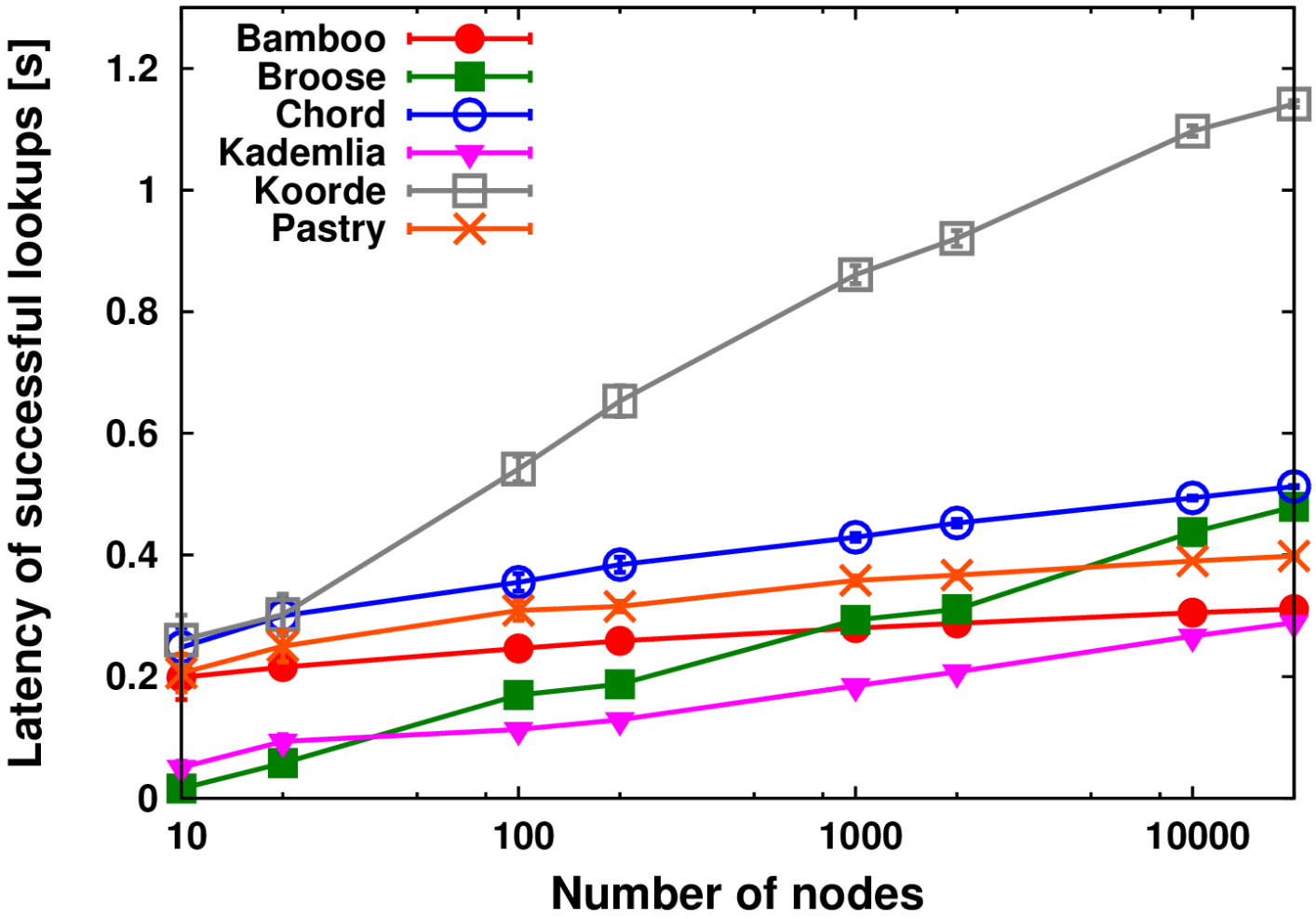


Comparison of KBR protocols

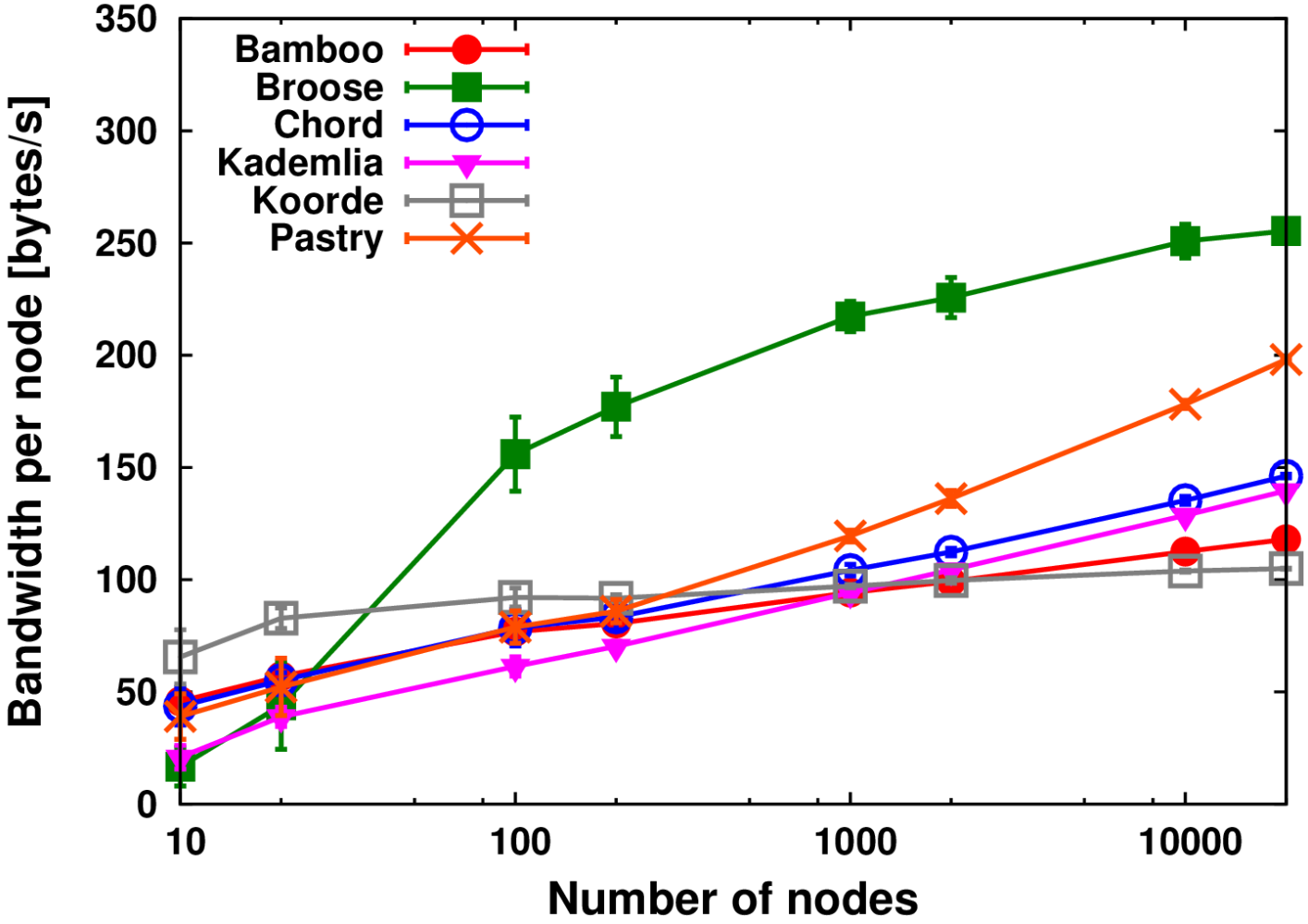


- Bamboo and Kademlia provide an efficient KBR service with low latency and low communication costs

Latency depending on network size



Bandwidth depending on network size



Summary of remarkable results

- Kademlia and Bamboo best regarding lookup latencies and communication costs
- Kademlia
  - Our extensions to separate bucket size k and number of returned nodes r for lookups lead to lower latencies while keeping traffic low
  - Parallel iterative lookups achieve similar effects like proximity neighbor selection (PNS)
- Chord and Pastry perform bad, but are still proposed for current P2P systems (e.g. IETF RELOAD)
- Protocols based on De Bruijn topologies (Koorde, Broose) don’t show any practical benefits
- Recursive routing should only be used with per hop acknowledgements

Conclusion

- OverSim supports fair comparison by separating common functions
- Graph properties often only play minor role – e.g. timeout handling much more critical!
- Bamboo and Kademlia (with modifications) are in general good candidates for KBR protocols

- Future work
  - Try to improve the RELOAD proposal based on our results
  - Compare even more protocols (please contribute!)

→ www.oversim.org