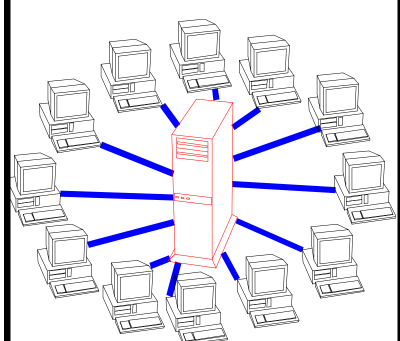
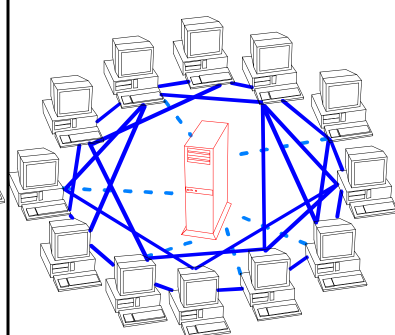
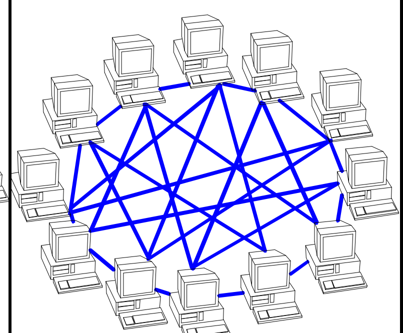
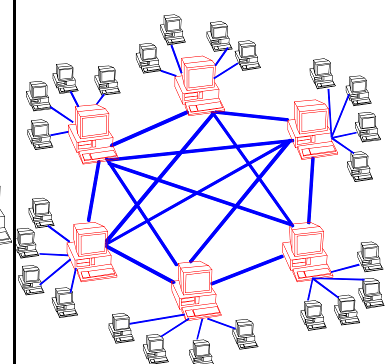
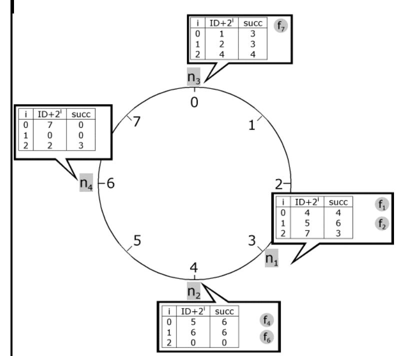
Distributed Hash Tables (DHT)

Jukka K. Nurminen

*Adapted from slides provided by Stefan Götz and Klaus Wehrle (University of Tübingen)

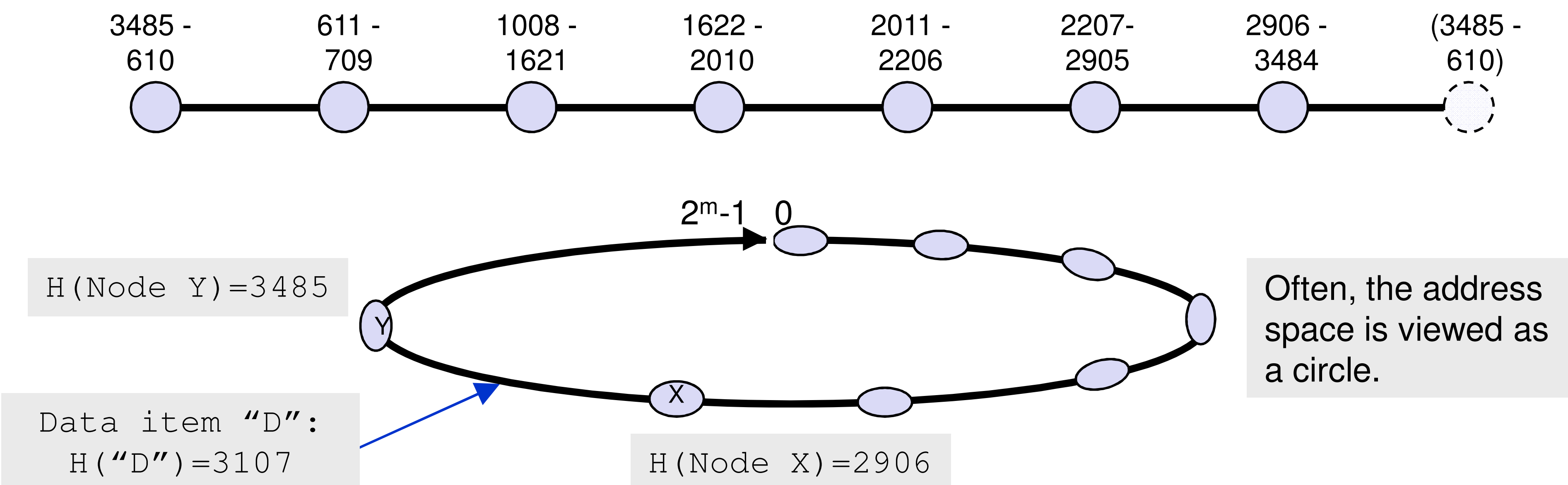
The Architectures of 1st and 2nd Gen. P2P

Client-Server	***Peer-to-Peer***			
1. Server is the central entity and only provider of service and content. → Network managed by the Server 2. Server as the higher performance system. 3. Clients as the lower performance system Example: WWW	1. Resources are shared between the peers 2. Resources can be accessed directly from other peers 3. Peer is provider and requestor (Servent concept)			
	Unstructured P2P			***Structured P2P***
	Centralized P2P	*Pure P2P*	*Hybrid P2P*	*DHT-Based*
	1. All features of Peer-to-Peer included 2. Central entity is necessary to provide the service 3. Central entity is some kind of index/group database Example: Napster	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → No central entities Examples: Gnutella 0.4, Freenet	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → dynamic central entities Example: Gnutella 0.6, JXTA	1. All features of Peer-to-Peer included 2. Any terminal entity can be removed without loss of functionality 3. → No central entities 4. Connections in the overlay are "fixed" Examples: Chord, CAN
				

1st Gen. (Centralized P2P, Pure P2P) — 2nd Gen. (Hybrid P2P, DHT-Based)

Addressing in Distributed Hash Tables

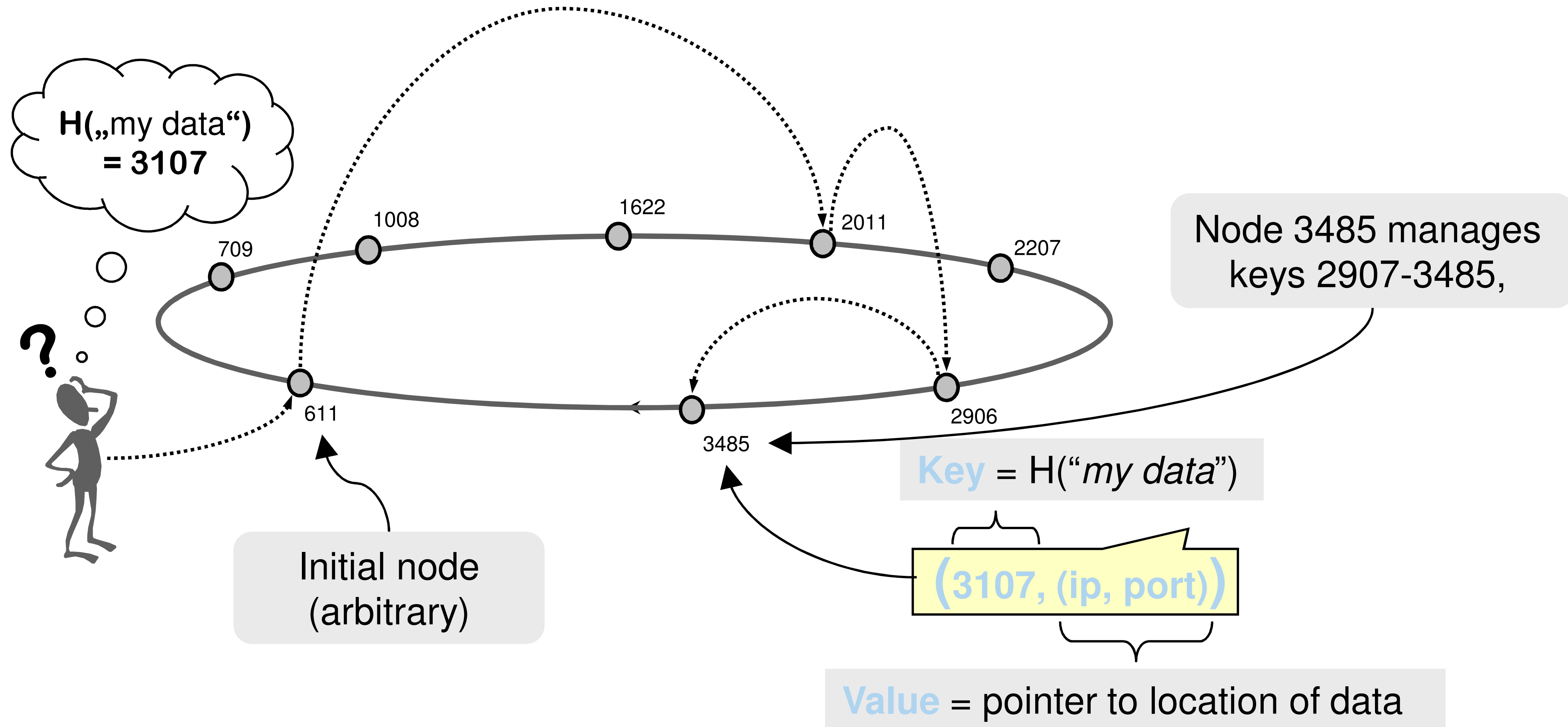
- Step 1: Mapping of content/nodes into linear space
  - Usually: 0, …, 2^m-1 >> number of objects to be stored
  - Mapping of data and nodes into an address space (with hash function)
    - E.g., Hash(*String*) mod 2^m: H(„*my data*") → 2313
  - Association of parts of address space to DHT nodes



Step 2: Routing to a Data Item

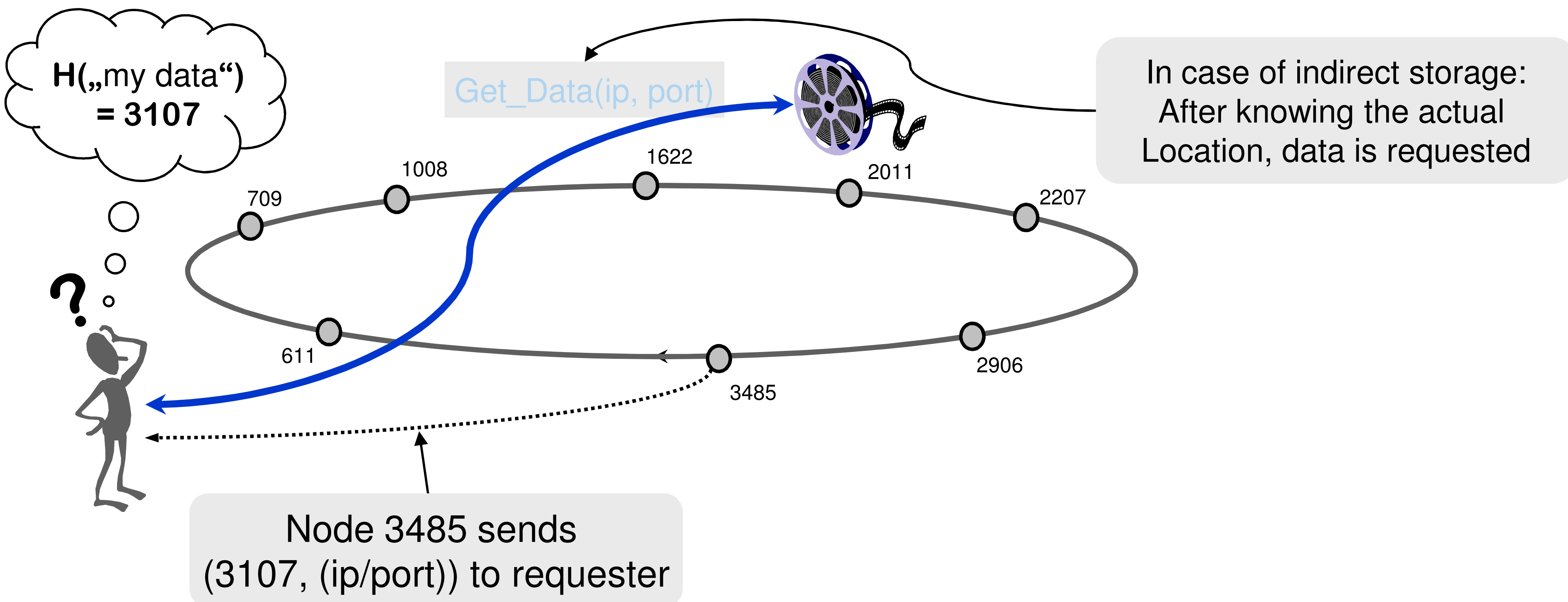
put (key, value)
value = get (key)

- Routing to a K/V-pair
  - Start lookup at arbitrary node of DHT
  - Routing to requested data item (key)

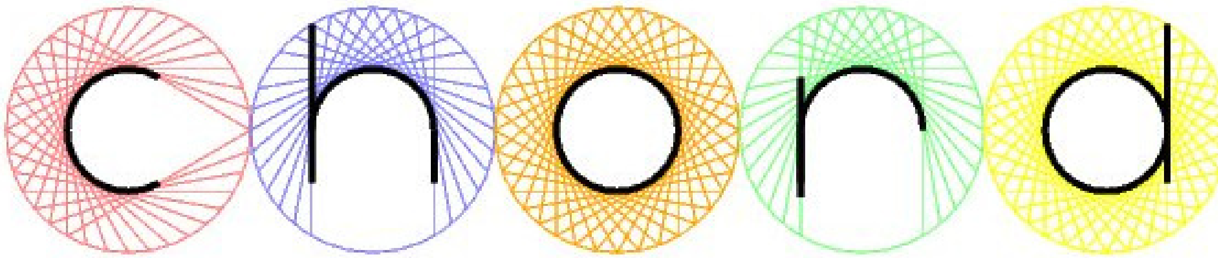


Step 2: Routing to a Data Item

- Getting the content
  - K/V-pair is delivered to requester
  - Requester analyzes K/V-tuple
    (and downloads data from actual location – in case of indirect storage)

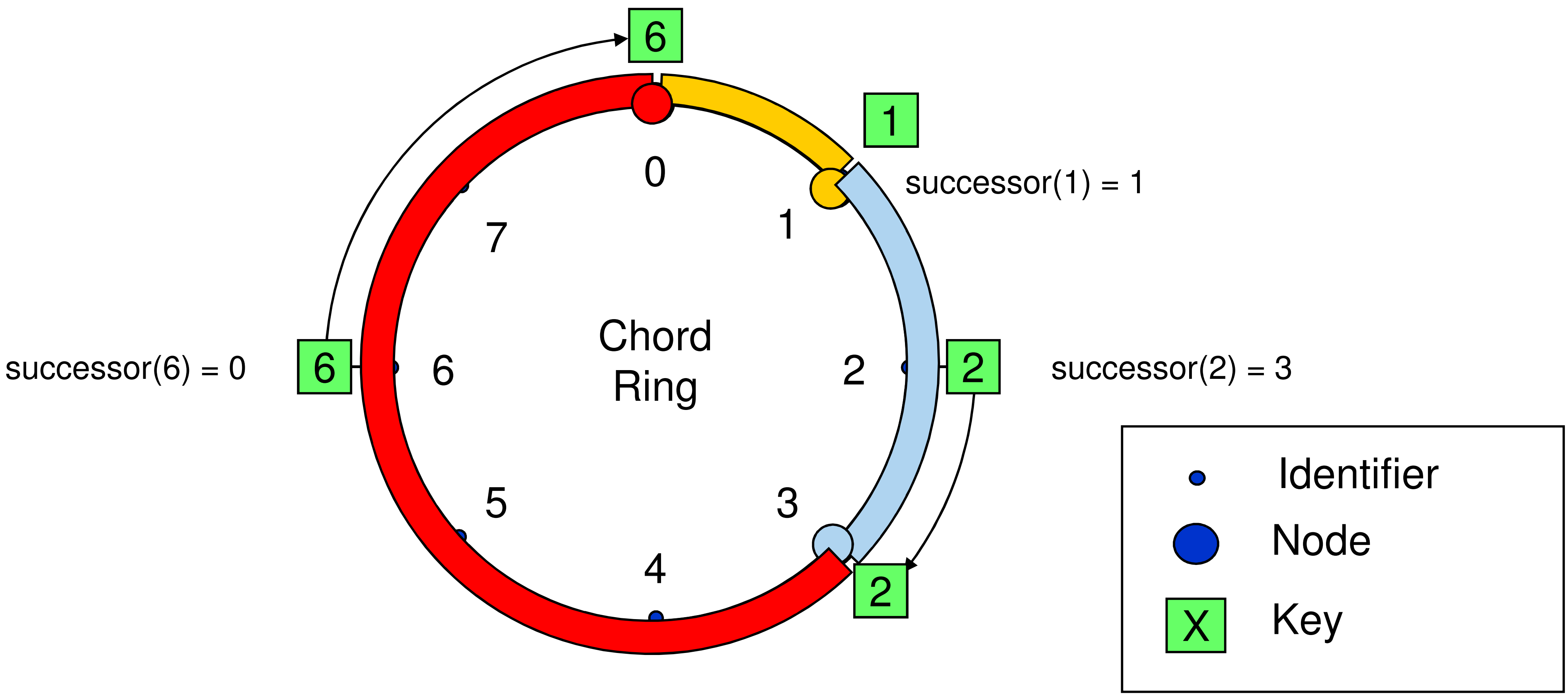


Chord



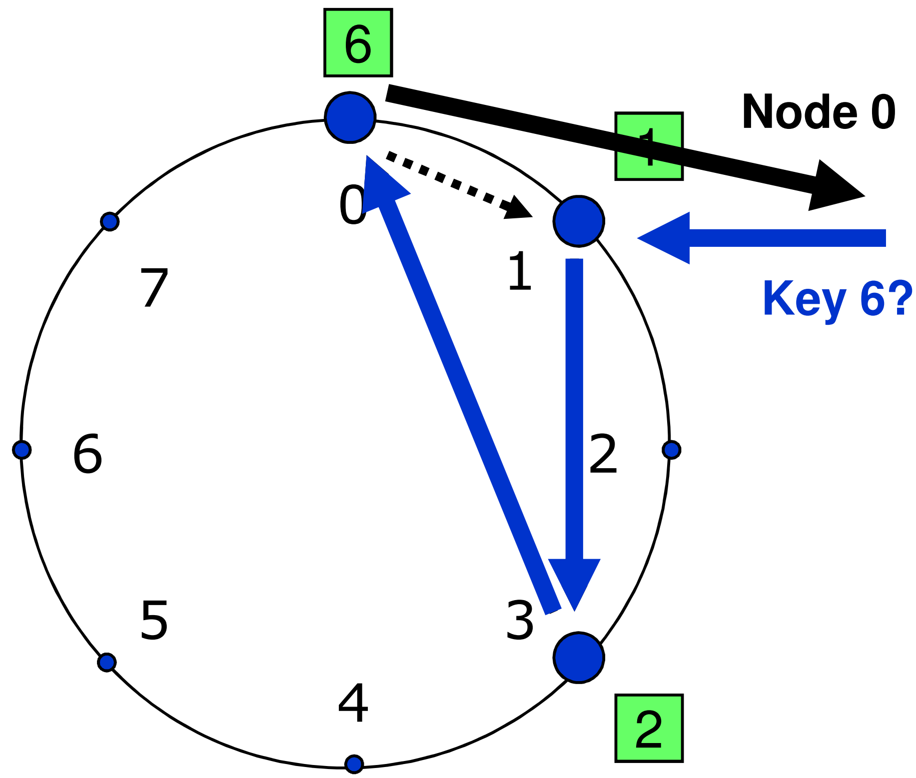
Chord: Topology

- Keys and IDs on ring, i.e., all arithmetic modulo 2^160
- (key, value) pairs managed by clockwise next node: successor



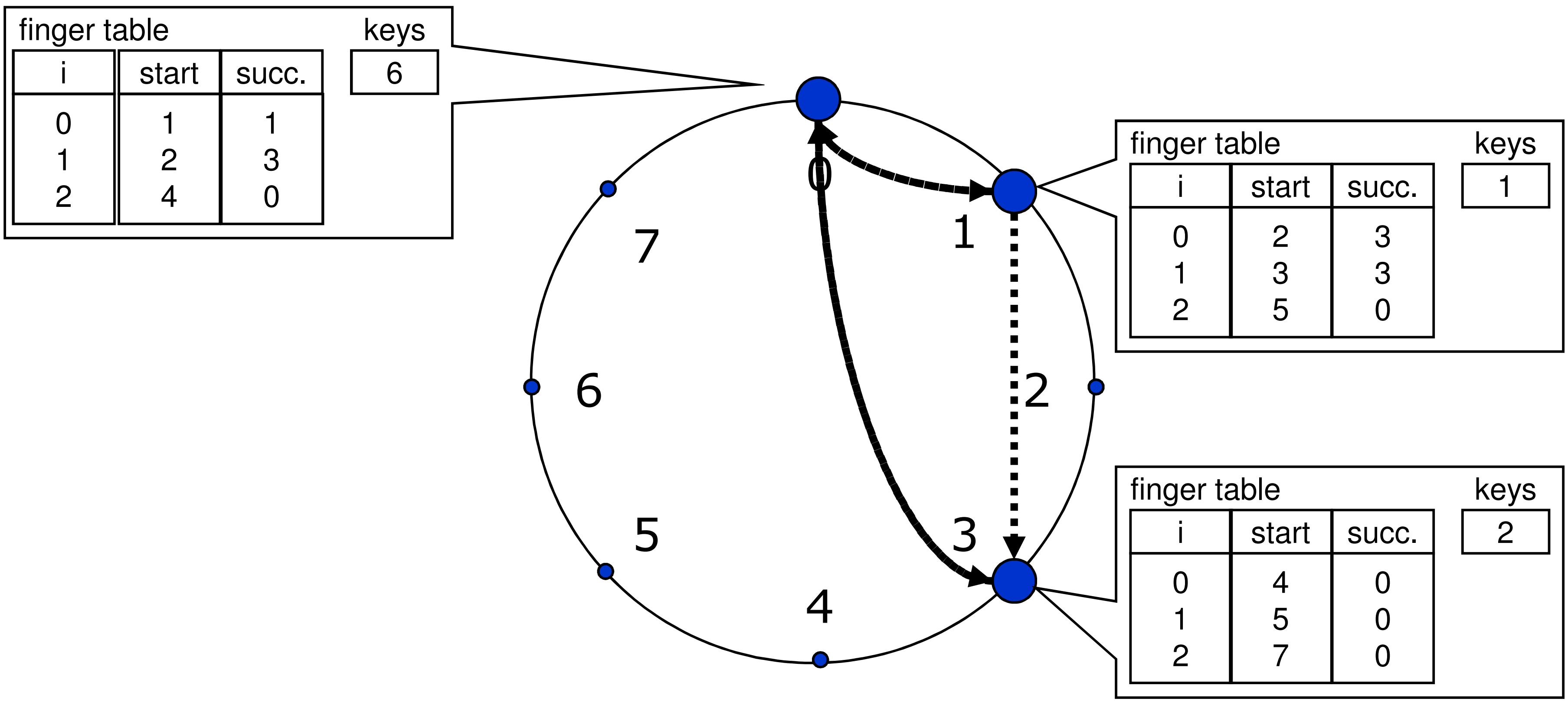
Chord: Primitive Routing

- Primitive routing:
  - Forward query for key x until successor(x) is found
  - Return result to source of query
- Pros:
  - Simple
  - Little node state
- Cons:
  - Poor lookup efficiency:
    O(1/2 * N) hops on average
    (with N nodes)
  - Node failure breaks circle



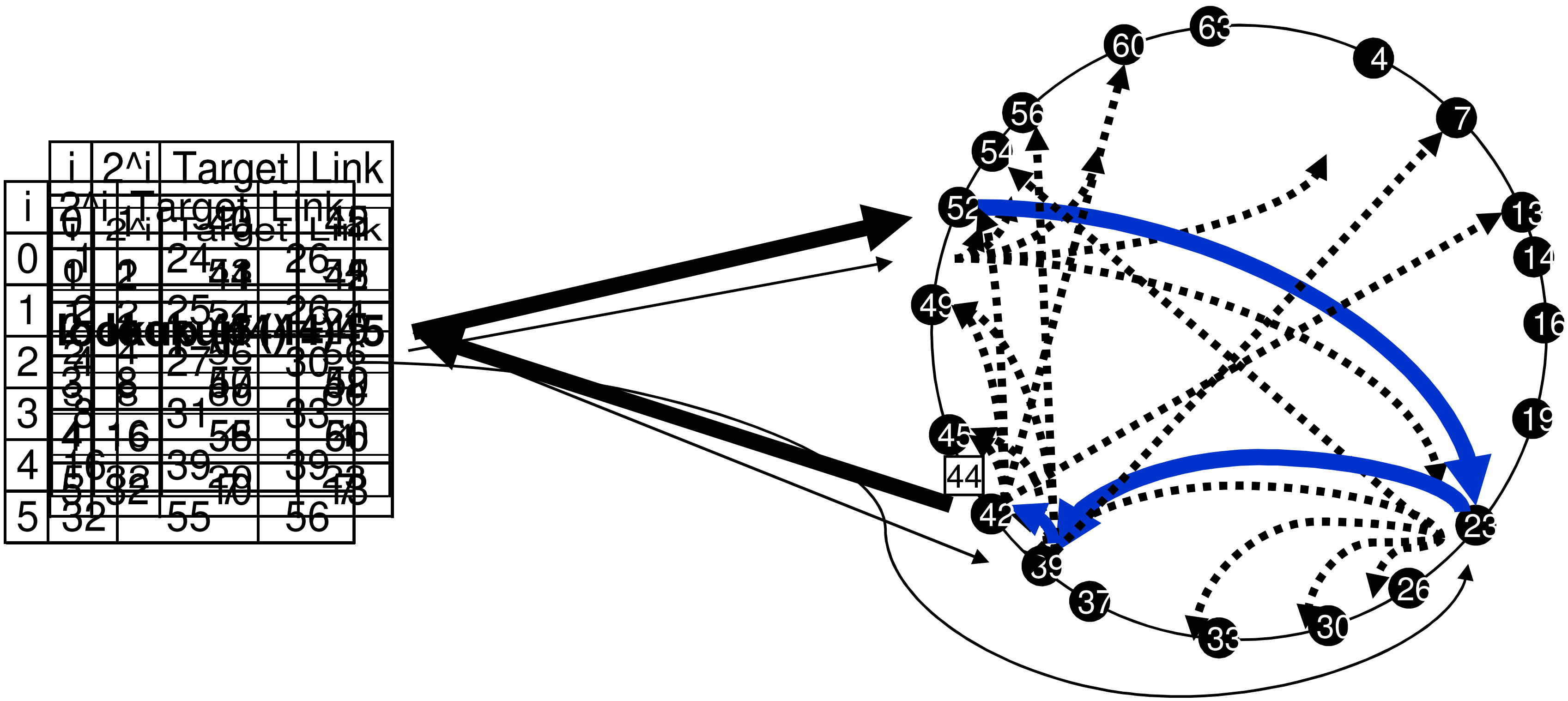
Chord: Routing

- Chord's routing table: *finger table*
  - Stores log(N) links per node
  - Covers exponentially increasing distances:
    - Node n: entry i points to successor(n + 2^i) (*i-th finger*)



Chord: Routing

- Chord's routing algorithm:
  - Each node n forwards query for key k clockwise
    - To farthest finger preceding k
    - Until n = predecessor(k) and successor(n) = successor(k)
    - Return successor(n) to source of query



Comparison of Lookup Concepts

System	Per Node State	Communi-cation Overhead	Fuzzy Queries	No false negatives	Robustness
Central Server	O(N)	O(1)	✓	✓	✗
Flooding Search	O(1)	$O(N^2)$	✓	✗	✓
Distributed Hash Tables	O(log N)	O(log N)	✗	✓	✓

Extra slides

Summary of DHT

- Use of routing information for efficient search for content
- Self-organizing system
- Advantages
  - Theoretical models and proofs about complexity (Lookup and memory O(log N))
  - Simple & flexible
  - Supporting a wide spectrum of applications
    - <Key, value> pairs can represent anything
- Disadvantages
  - No notion of node proximity and proximity-based routing optimizations
  - Chord rings may become disjoint in realistic settings
  - No wildcard or range searches
  - Performance under high churn. Especially handling of node departures
  - Key deletion vs. refresh
- Many improvements published
  - e.g. proximity, bi-directional links, load balancing, etc.

Different kinds of DHTs

- Specific examples of Distributed Hash Tables
  - Chord, UC Berkeley, MIT
  - Pastry, Microsoft Research, Rice University
  - Tapestry, UC Berkeley
  - CAN, UC Berkeley, ICSI
  - P-Grid, EPFL Lausanne
  - Kademlia, Symphony, Viceroy, …
- A number of uses
  - Distributed tracker
  - P2P SIP
  - ePOST