

Faster Content Access in KAD

Mortiz Steiner, Damiano Carra, and
Ernst Biersack

Eurécom
Sophia-Antipolis, France
September 2008

Outline

- What is a DHT
- What is KAD
- How KAD works
 - ◆ Routing
 - ◆ Publishing
 - ◆ Searching
- Conclusion

KAD

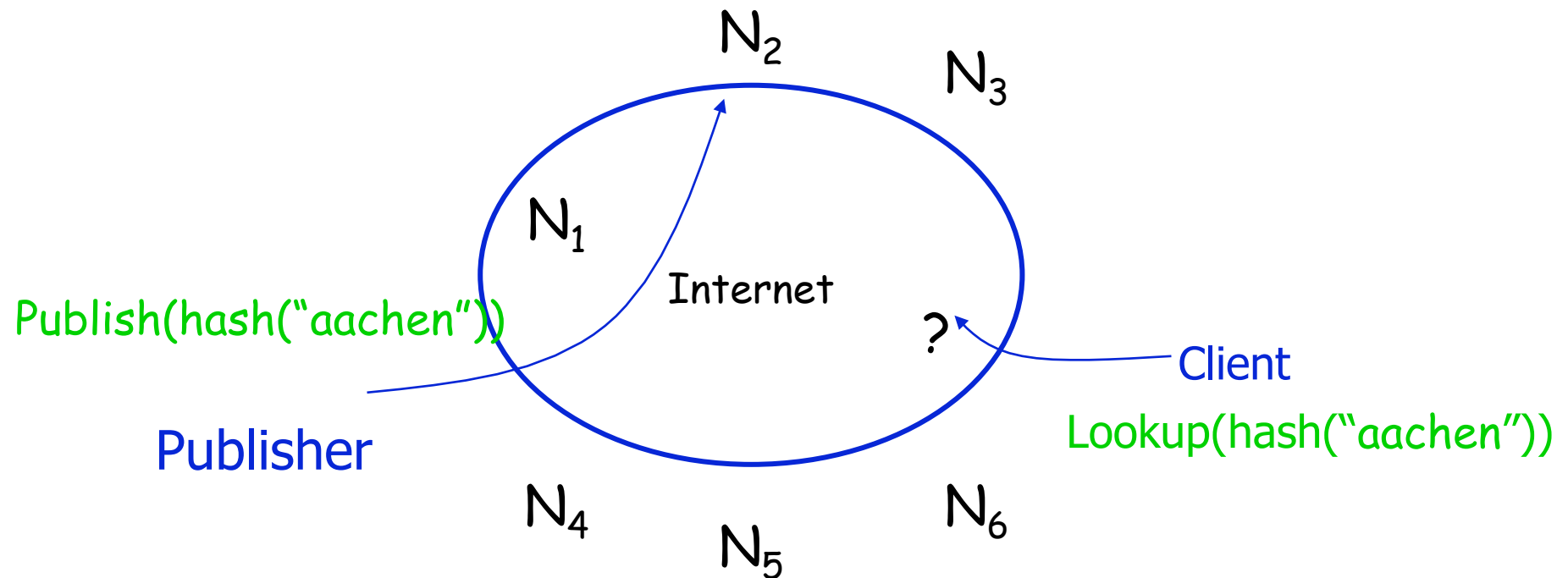
- Why interesting?
 - ◆ Only real “production” DHT
 - ◆ Very popular, since part of
 - ✦ eMule
 - ✦ aMule
 - ✦ (Azureus)

- What is KAD?
 - ◆ DHT derived from Kademia
 - ◆ DHT with largest user base:
 - ✦ At any point of time more than 1.5 Mio peers connected

What is a DHT

- What is a DHT
 - ◆ **A distributed database for publishing and searching information**
 - ◆ Consists of many peers, each one is responsible for storing part of the database
 - ◆ How to partition content of DB
 - ✦ Use key of the object to decide on which peer to store information
 - ◆ What is a key: unique identifier
 - ✦ Key = hash(IP@), or
 - ✦ Key = hash(string)
 - ◆ Each peer and each object is identified by its key

What is a DHT

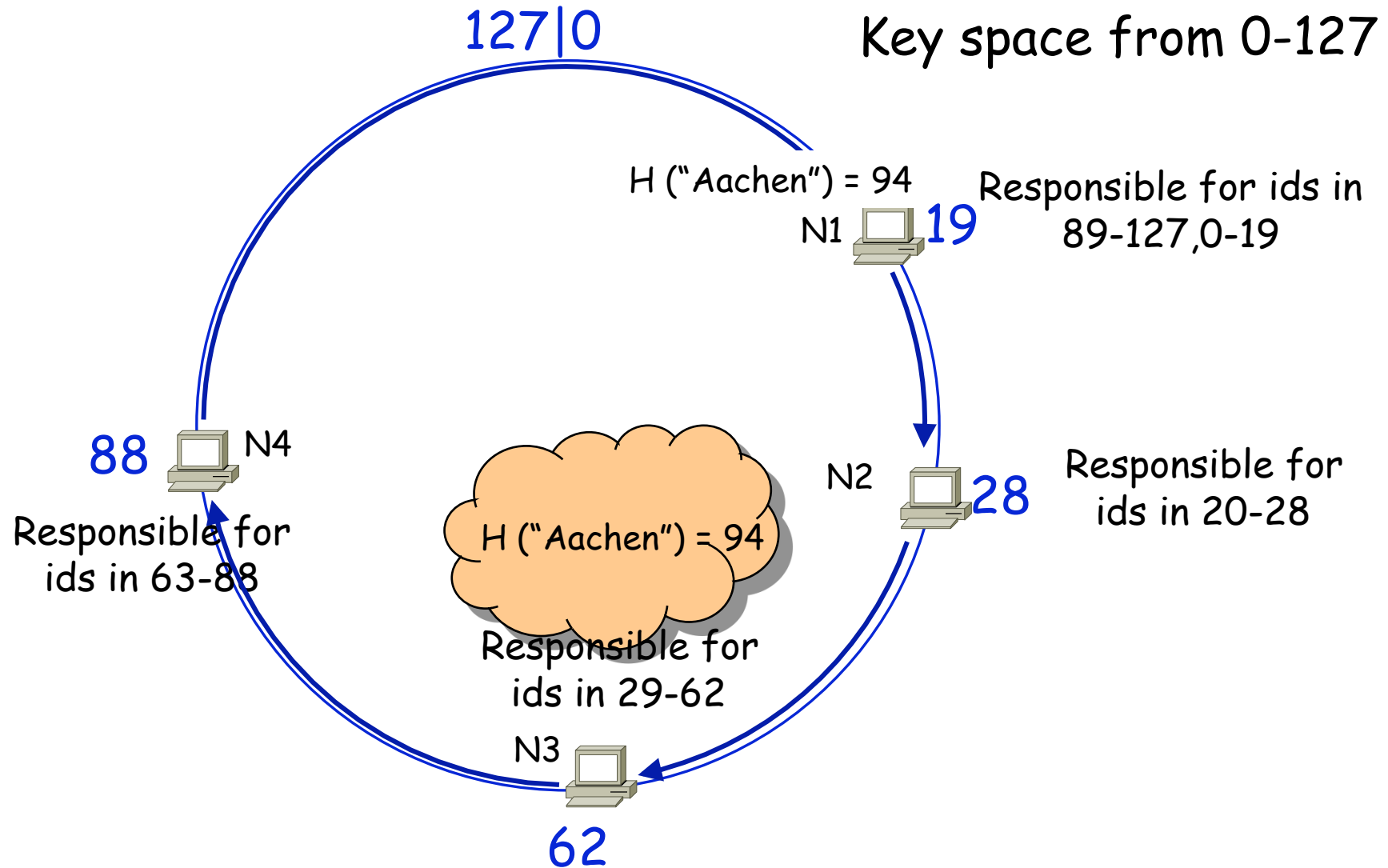


- Important issues
 - ◆ How to partition key space
 - ◆ How to route
 - ◆ How to maintain information under churn (peers joining and leaving all the time)

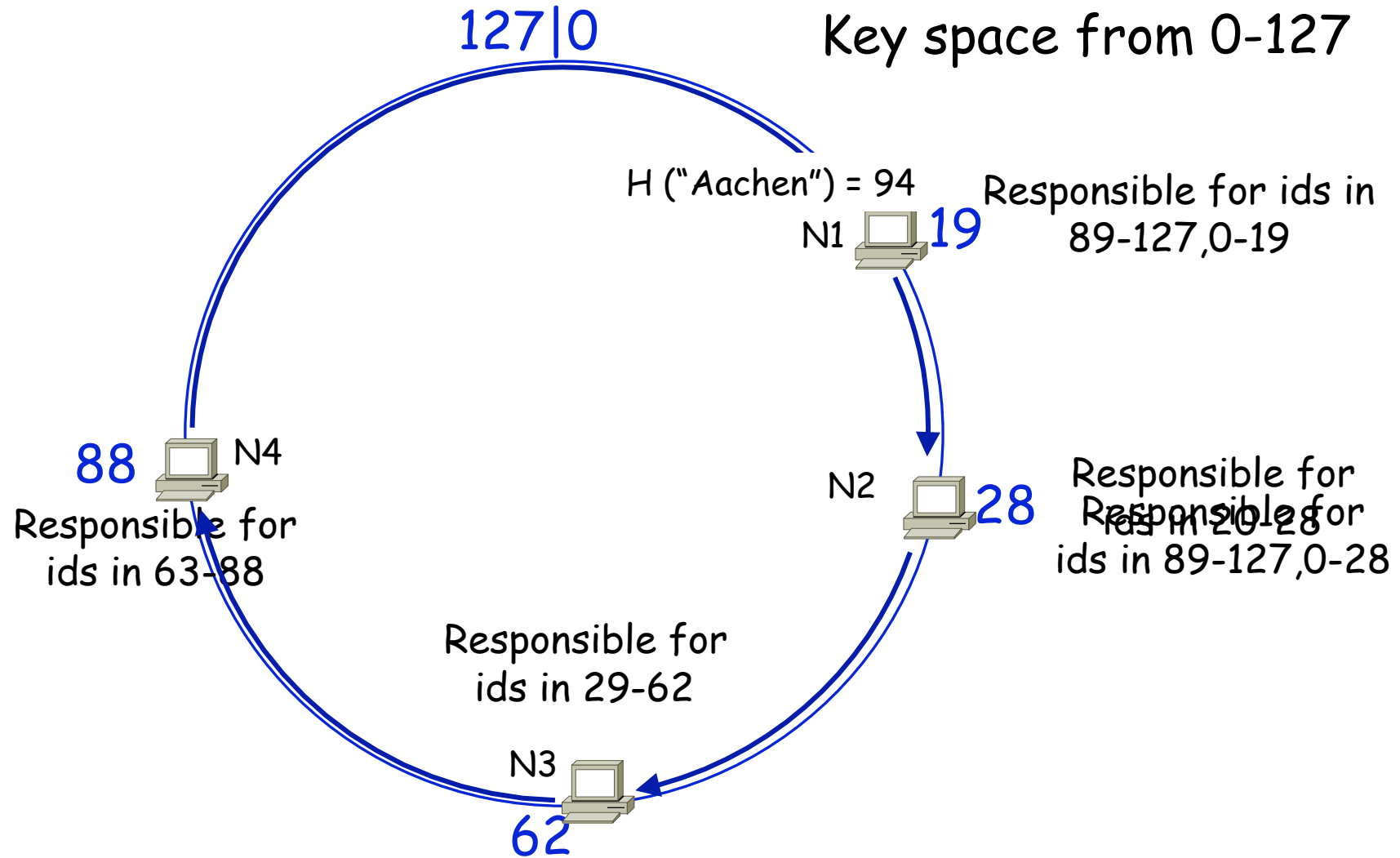
P2P System to hold a Distributed DB

- Assume the following problem
 - ◆ We have the content of the phone book
 - ✦ Entries: **name, phone#**
 - ◆ Want to partition the content over all peers so that each peer is uniquely responsible for a small subset of the entries
 - ✦ Q: which entries should be stored at which peer?
 - ✦ A: HASHING: function $H(\text{string}) \rightarrow \text{ID} \in (0, 127)$
 - $H(\text{"finger"}) = 105 \in (0, 127)$
 - $H(\text{"steiner"}) = 43 \in (0, 127)$
 - ◆ Everything has an $\text{ID} \in (0, 127)$
 - ✦ Content
 - ✦ Nodes $H(\text{"random number"}) = 19 \in (0, 127)$
- → Data structure is called **Distributed Hash Table (DHT)**

Vanilla DHT: Principles



Vanilla DHT: Node Failure



Key Look-Up in Kademlia

- Each key k is a bit string of length $m = 160$ bits:

- XOR metric:

- Let node $j = j_{m-1}j_{30} \dots j_0$ and $k = k_{m-1}k_{30} \dots k_0$:

- Note that closest ID is **unique**:

- $d(j,k) = d(j',k) \Leftrightarrow j = j'$

$$d(j,k) = \sum_{i=0}^{31} |j_i - k_i| \cdot 2^i$$

- Refinement of longest-prefix match

- Example (8 bits)

$k = 10010110$

$j = 10110110$

$j' = 10001001$

$d(j,k) = 2^5 = 32$

$d(j',k) = 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 31$

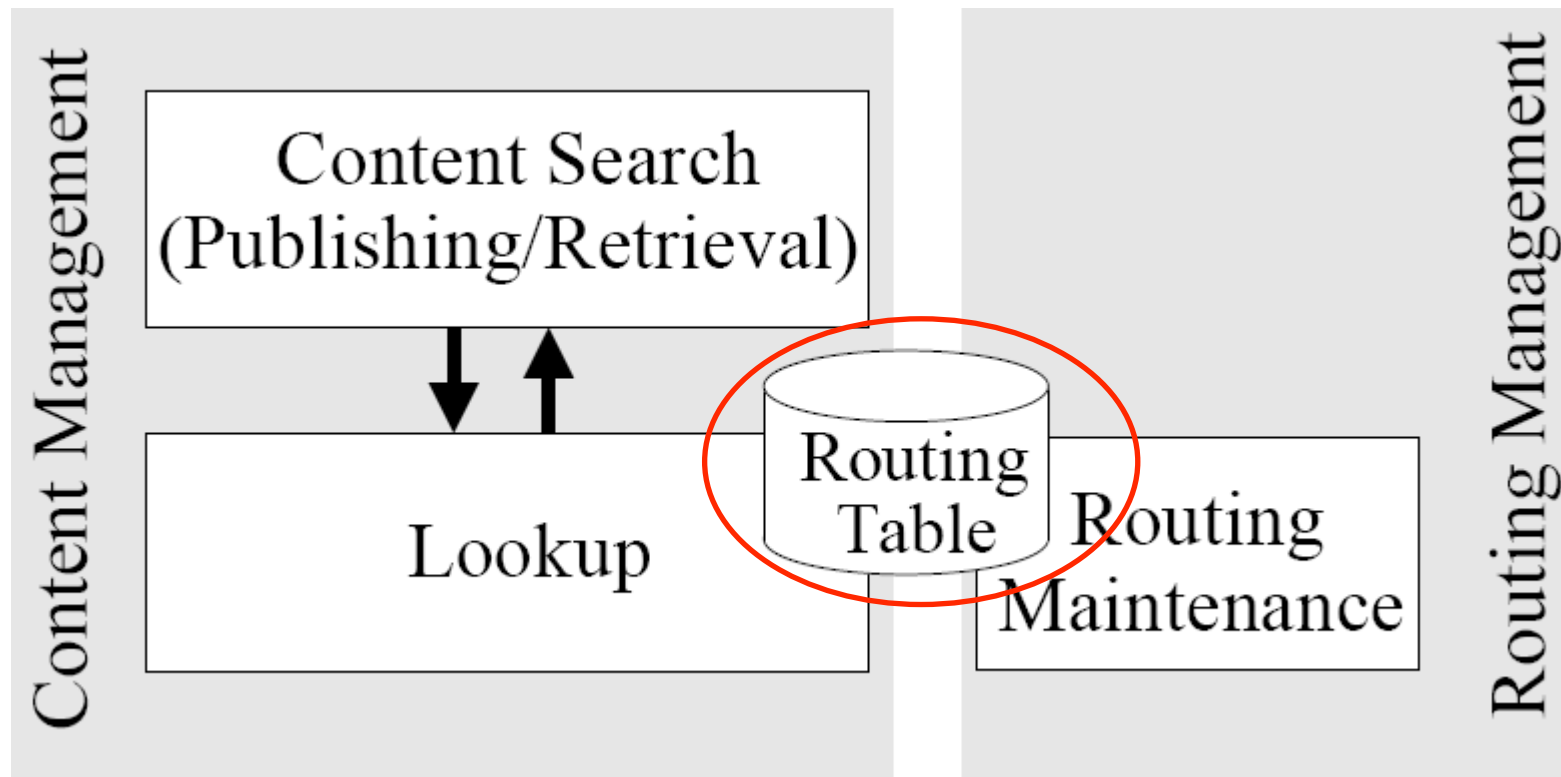
- Advantages:

- Easy to compute

- Symmetrical: If node j is close to node i then node i is also close to j

- Useful to learn new contacts (routing table entries) simply by receiving communication requests from other peers

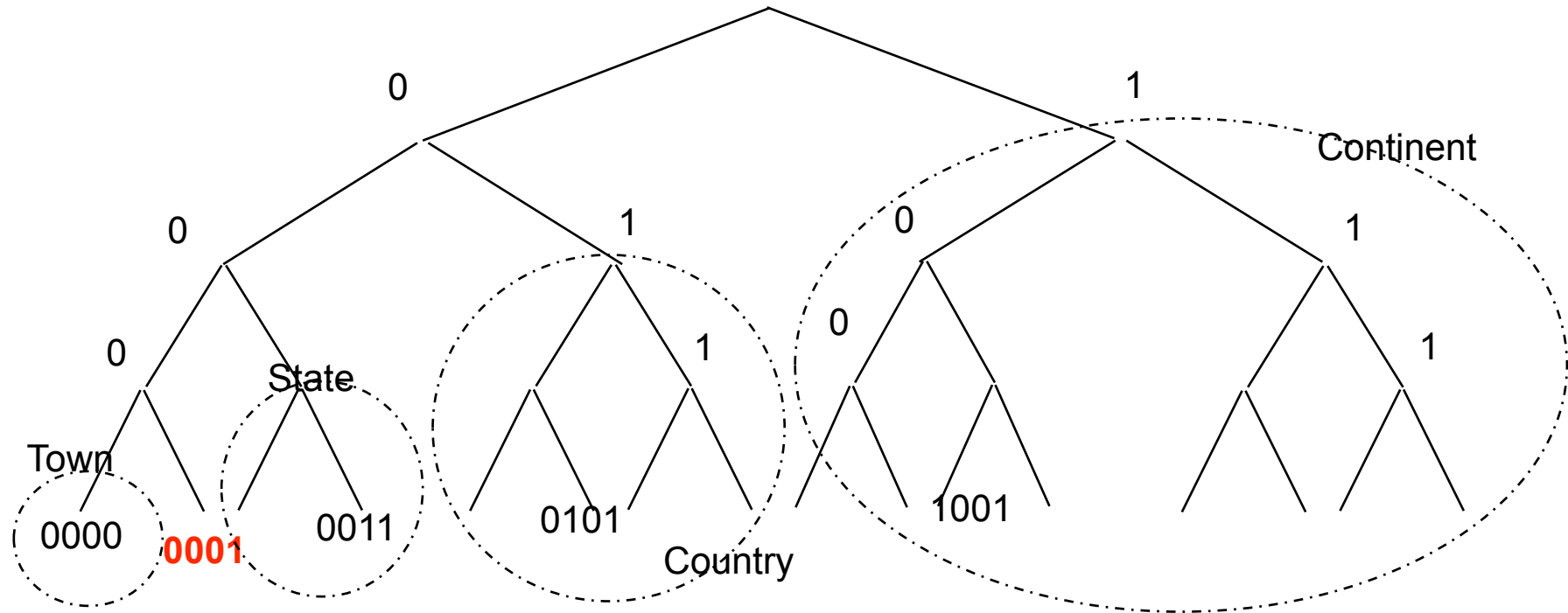
KAD Architecture



Note: Lookup module is used by both, the Publishing and Search module

Our discussion refers to KAD as implemented in version 2.1.3 of aMule

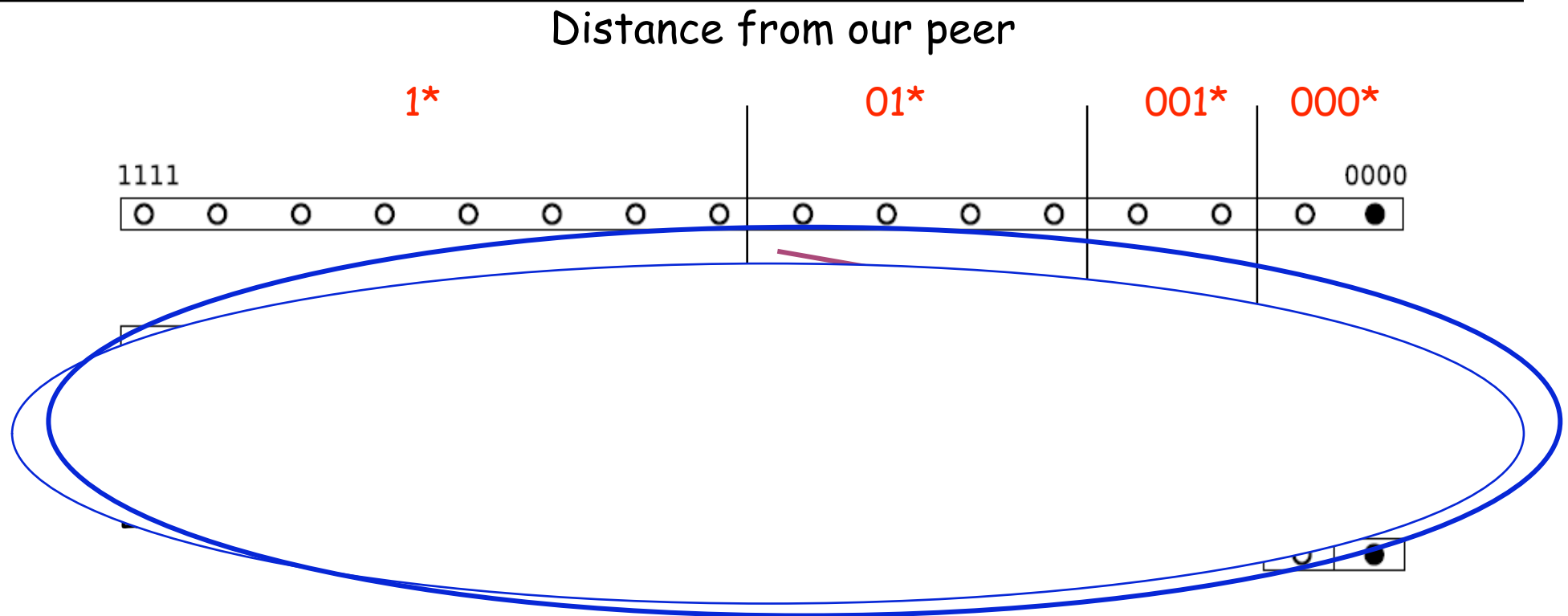
KAD: Routing Table



Peer 0001		
i	Contacts	Distance
1	1001	1000
2	0101	0100
3	0011	0010
4	0000	0001

- Organize peers in tree:
- How does routing table of a peer look like
 - ◆ Entries r_i
 - ✦ r_i has the **same** (i-1) first bits and bit i **inverted**
 - ◆ Consider peer with ID 0001
 - ✦ Has entries to 1001, 0101, 0011, 0000

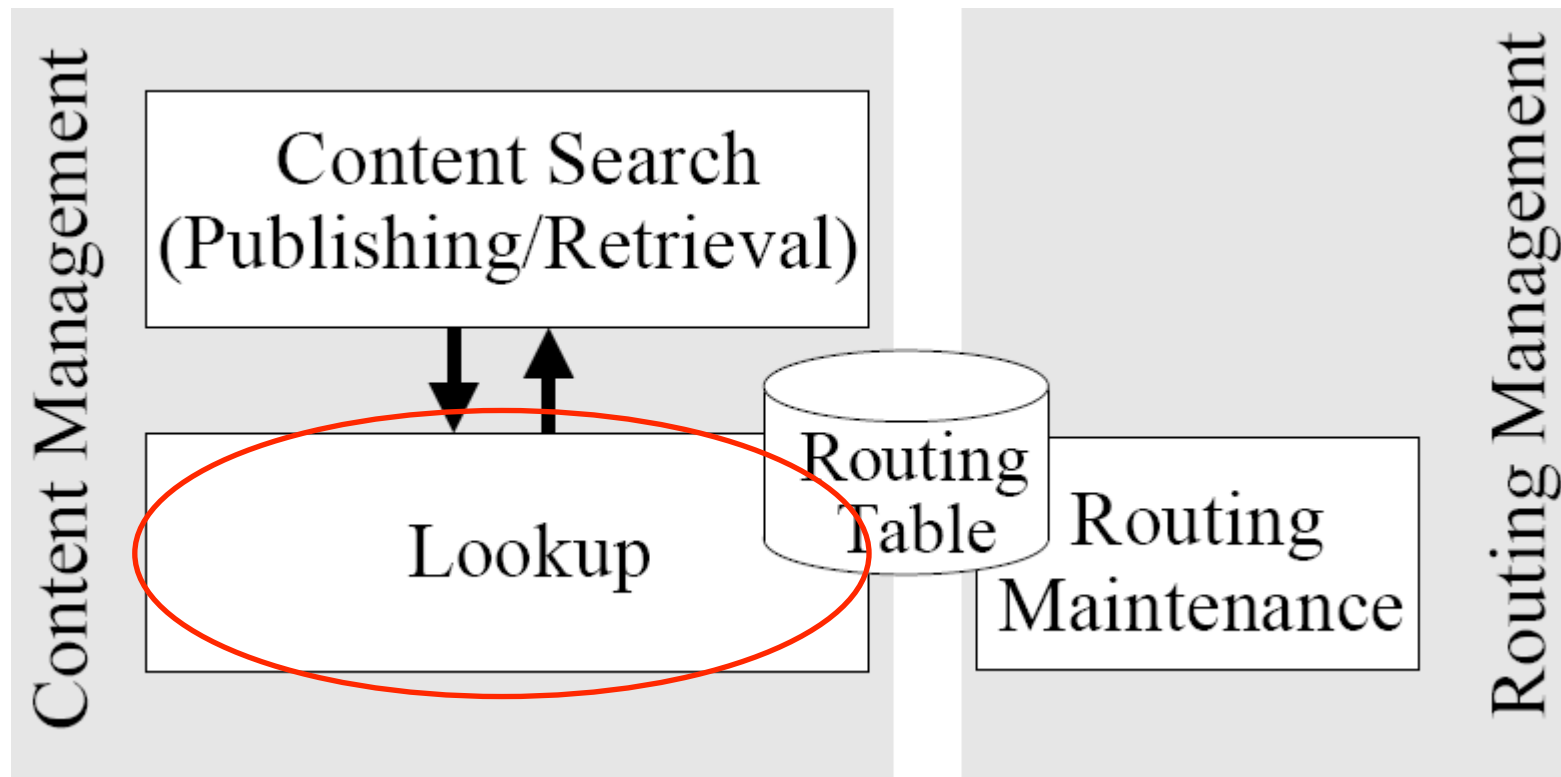
KAD- Routing Table



- **How to route, i.e find next hop peer?**
 - ◆ Compute XOR distance *Dist* between ID of node and ID of target
 - ◆ Use *Dist* value bit-by-bit to traverse routing table until reaching a leaf (bucket)
 - ◆ Take one of the bucket entries as next hop
 - ◆ Ex.: Dist = **0010**



KAD Architecture



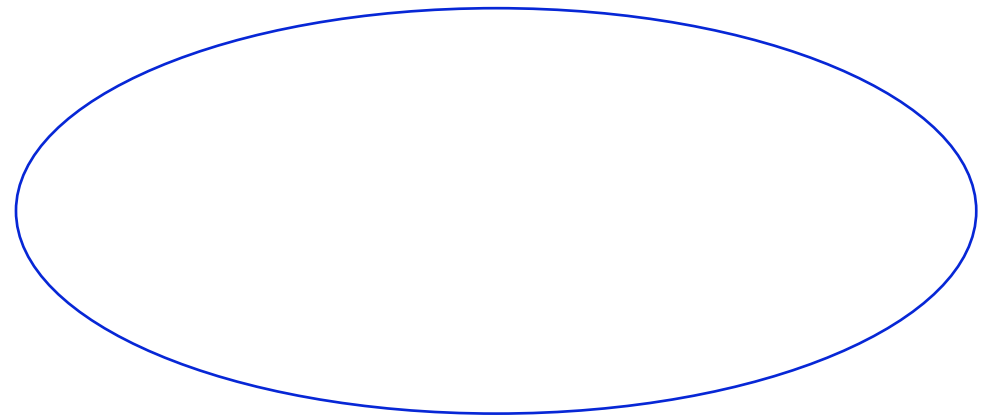
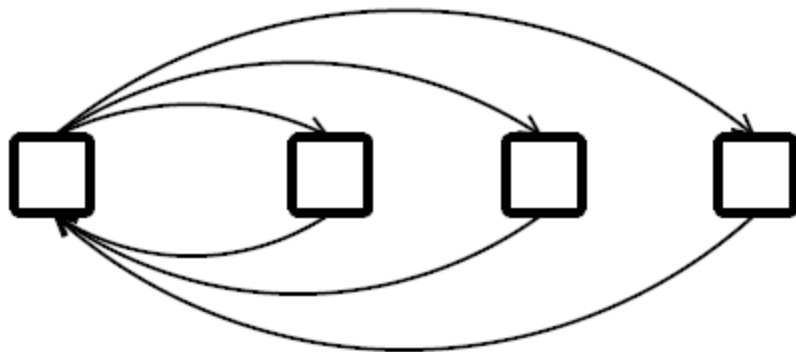
Note: Lookup module is used by both, the Publishing and Search module

Our discussion refers to KAD as implemented in version 2.1.3 of aMule

KAD- Lookup : Iterative

- KAD uses **iterative routing**
 - ◆ Source is responsible for entire lookup process
 - ◆ At each step, source send lookup request to the next hop and waits for reply
- Adv of iterative routing:
 - ◆ Lookup messages cannot be lost due to the departure of an intermediate peer holding the lookup request
 - ◆ Iterative routing is easier to debug since information at each step is reported back to source

Iteration:



KAD: Lookup process

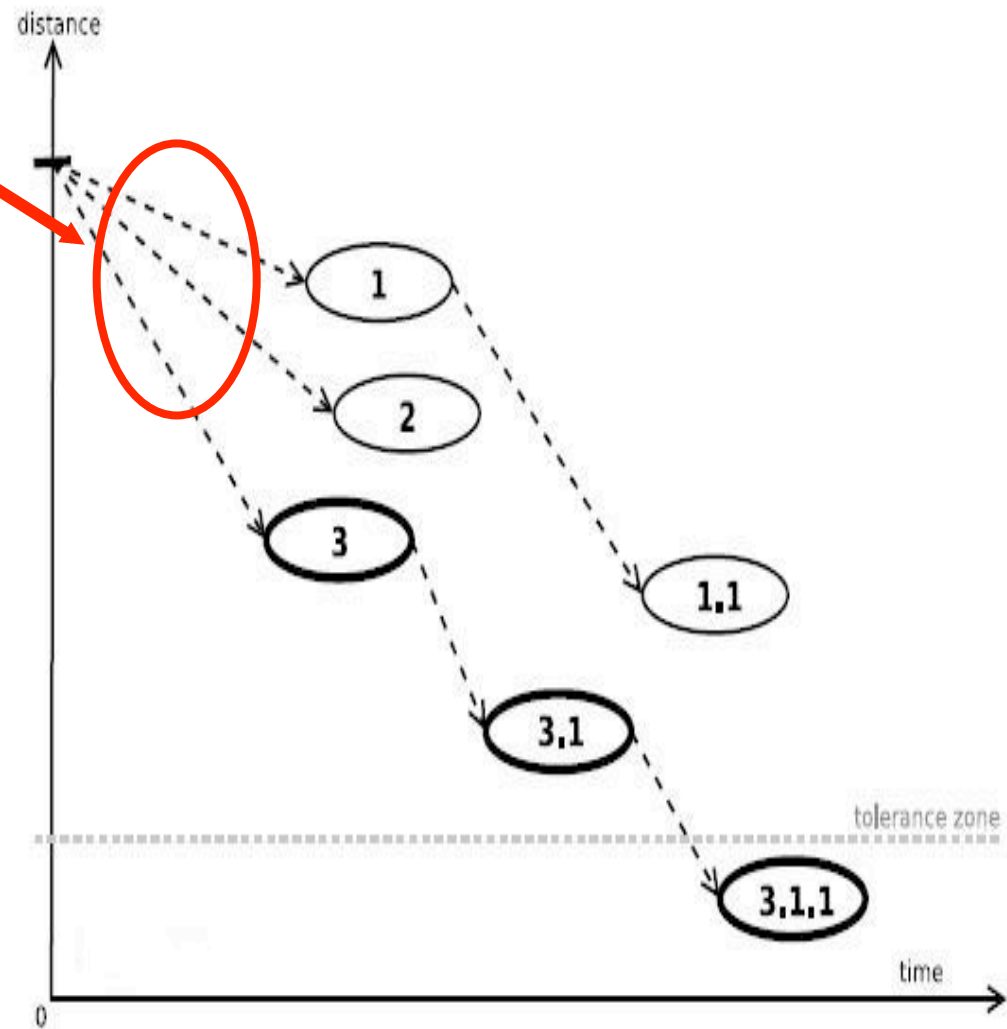
- Lookup starts by

asking 3 peers

- Each peer answers

by providing 2 closer peers

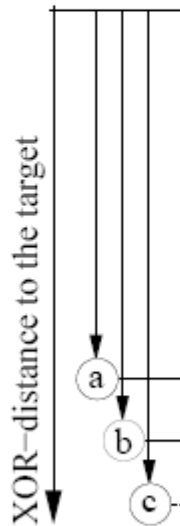
- Lookup is iterative



KAD- Lookup : Parallel lookup

Top α
closest

	s	r
a	✓	
b	✓	
c	✓	
d		



β = # number of nodes in response

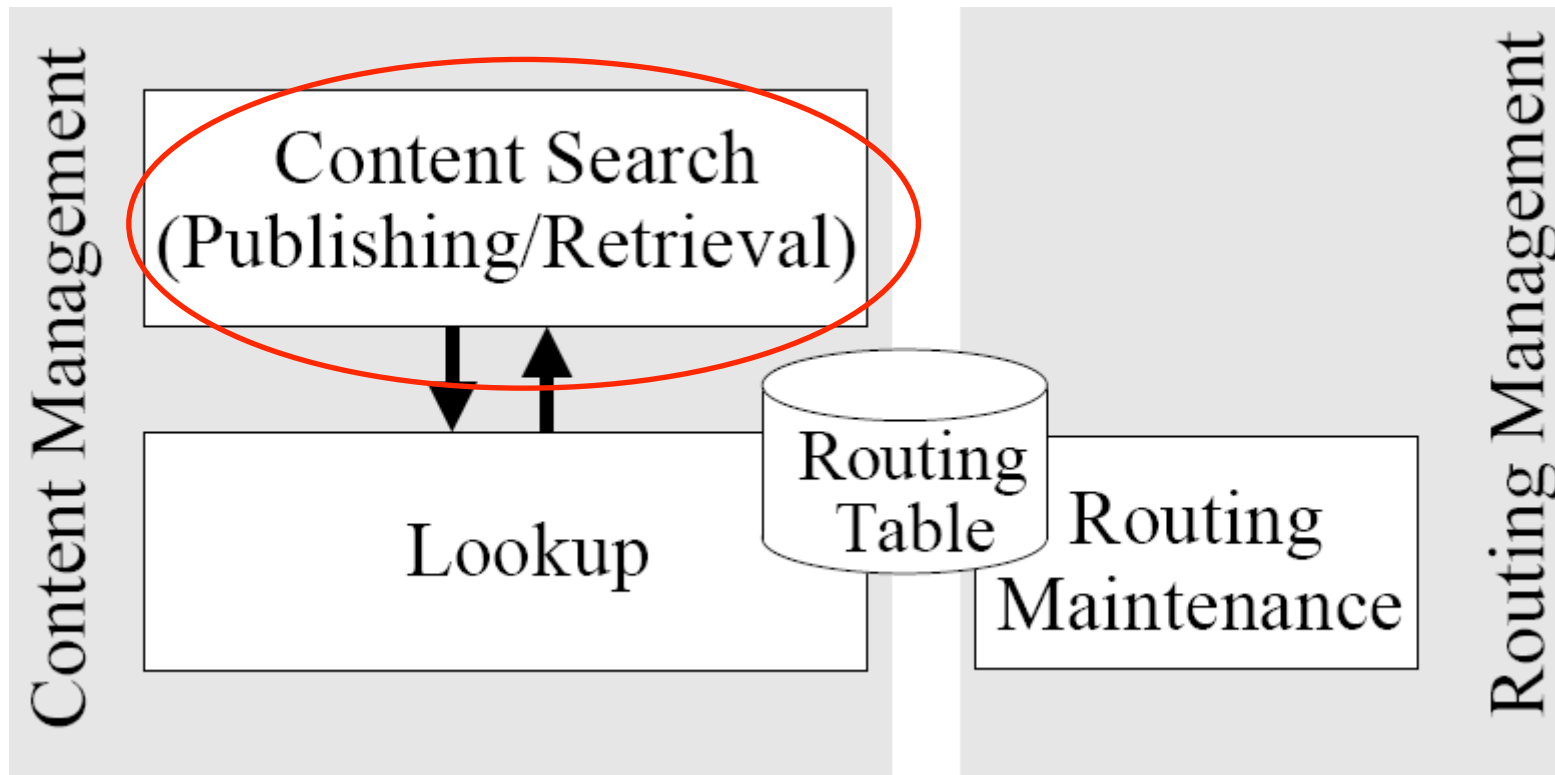
Figure 2. Example of lookup ($\alpha = 3$; $\beta = 2$).

KAD- Lookup : Parallel lookup to improve Lookup-Speed

- **Parallel** lookup:
 - ◆ Source simultaneously issues multiple lookup requests to different peers and continues the lookup process based on the information obtained from all requests
- **Adv.:**
Reducing the problem of hitting stale contacts and improves lookup performance
- **Disadv.:**
Greater control traffic overhead i.e., a larger number of requests per lookup



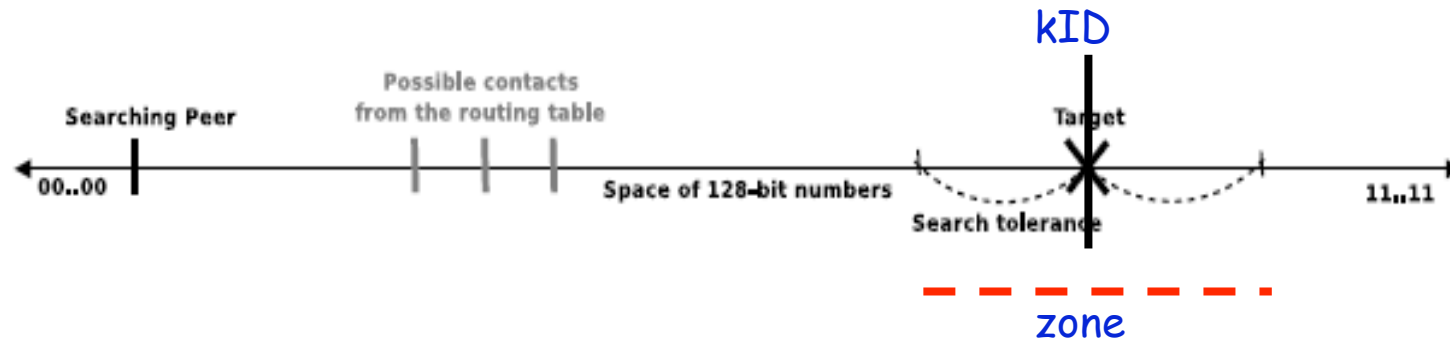
KAD Architecture



Note: Lookup module is used by both, the Publishing and Search module

KAD Publish: How and Where

- Where to **publish** the information for a given a key **kID**?
 - ◆ In KAD: On 11 nodes, who's first 8-bits are the same as **kID**
 - ✦ i.e. there is a **zone** and all peers in this zone are potential candidates for storing the key

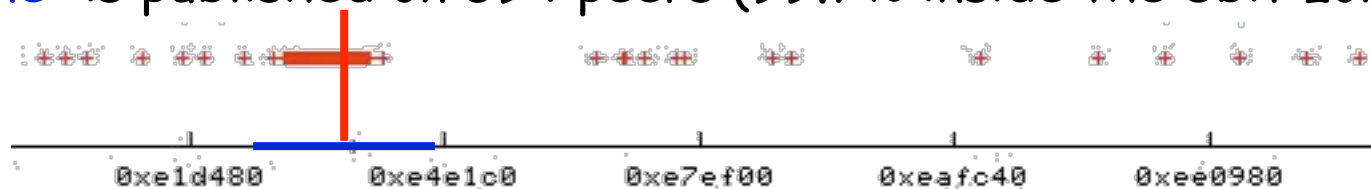


- A zone defined by the first 8 bits is $1/256$ of the entire key space and contains **several thousand peers**

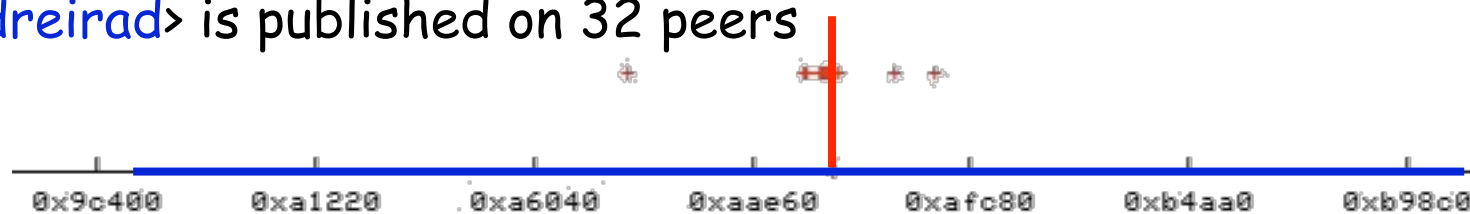
KAD Publish: How and Where: Keyword Hash Distribution from Measurements

- Observations:
 - ◆ An 8-bit zone contains about 8000 nodes
 - ◆ How close the nodes we publish on (w.r.t to key) depends on whether the keyword
 - ✦ Is popular (many different publishers) or
 - ✦ Is rare (single publisher)

- <the> is published on 594 peers (99.7% inside the 8bit zone)



- <dreirad> is published on 32 peers



1. The popular keyword is spread on 30 times wider keyspace

KAD Publish/ Retrieval

- Two steps
 - ◆ 1) Find close by nodes that agree in at least the first 8 bits
 - ◆ 2) Contact 11 of the closest nodes for publish/retrieval

KAD Publish: How and Where

- Where to **publish** the information for a given a key **kID**?
 - ◆ On 11 nodes, who's first 8-bits are the same as **Kid**
 - How to find these nodes in 8-bit zone?
 - ◆ **Do look-up**: Issue iterative **route_req(kID)** to discover nodes in 8-bit zone
 - ✦ Will obtain an ordered (by closeness) list of candidate nodes
 - ✦ Question: When to stop searching for closer nodes???
- Stop when candidate list does not change anymore for **t** seconds
(**t=3 seconds**)
- When candidate list is stable for **t** seconds
 - ◆ Contact 11 closest (to key) candidate nodes and publish content

KAD Retrieval : How and Where

- Where to **retrieve** the information for a given a key **kID**?
 - ◆ On 11 nodes, who's first 8-bits are the same as **Kid**
- How to find these nodes in 8-bit zone?
 - ◆ **Do look-up**: Issue iterative **route_req(kID)** to discover nodes in 8-bit zone
 - ✦ Will obtain an ordered (by closeness) list of candidate nodes
 - ✦ Question: When to stop searching for closer nodes???
- Stop when candidate list does not change anymore for **t** seconds (**t=3 seconds**)
- When candidate list is stable for **t** seconds
 - ◆ Contact 11 closest (to key) candidate nodes with search request
- Is waiting for 3 seconds a good idea??

Results

- How do we obtain the results
 - ◆ Measurement of real system
 - ✦ Instrument aMule peer client code
 - ✦ Modify the key parameters
 - ◆ Analytical modeling to answer what-if questions

KAD Retrieval: Lookup Latency

Total lookup latency

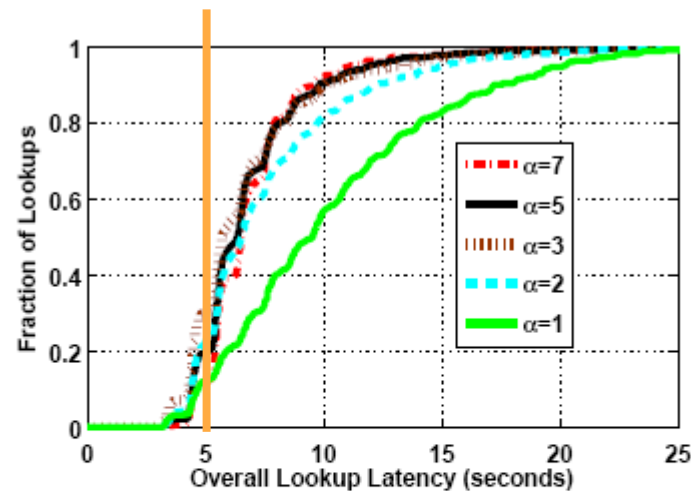


Figure 8. Empirical CDF of the overall lookup latencies as a function of the degree of parallelism α ($\beta = 2; t = 3$).

At least 5 seconds,
(parallelism ($\alpha > 1$) reduces latency quite a bit)

KAD Content Search: Design Parameters

- **Alpha**: Number of parallel route requests
- **Beta**: number of new contacts contained in each answer
- **T**: timeout before contact list is considered stable

KAD Content Search: Alpha

- parallelism ($\alpha > 1$) helps a lot
- $\alpha = 3$ is a reasonable choice

KAD Content Search: Timeout T

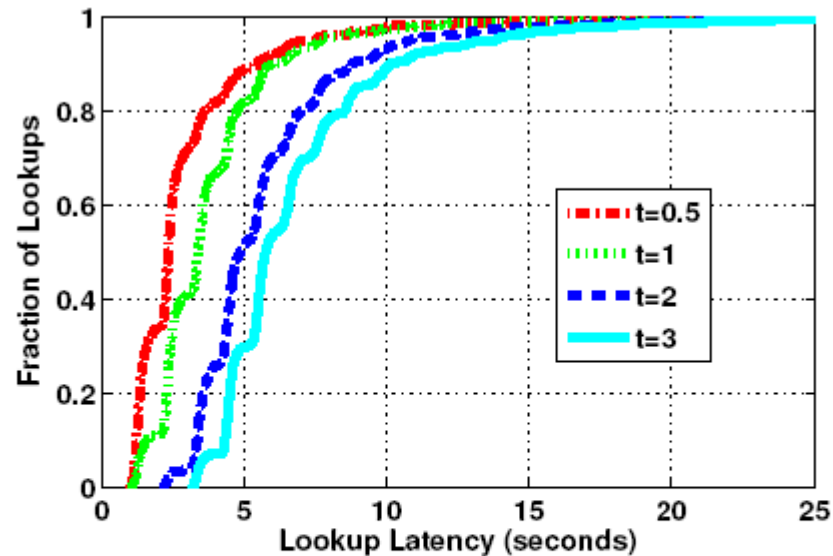


Figure 8. Empirical CDF of the overall lookup latencies as a function of the route request timeout t ($\alpha = 3; \beta = 2$).

- T: smaller timeout significantly reduces overall lookup latency
 - T=3 is too large for content search

Exploring the parameter Space

α	$\beta = 2; t = 3$		$\beta = 2; t = 0.5$	
	average # of messages $\bar{\rho}_h$	median lookup latency	average # of messages $\bar{\rho}_h$	median lookup latency
1	8.5	9.5	10.4	5.6
2	11.5	6.6	12.8	2.4
3	13.7	5.8	15.2	2.3
4	16.9	6.1	18.0	2.3
5	20.0	6.4	20.6	2.2
6	22.9	6.5	24.0	2.3
7	26.5	6.5	27.7	1.8
8	30.0	6.6	30.4	1.6
9	32.9	6.6	34.0	1.5
10	36.7	6.6	36.8	2.2

- Increasing alpha will result in
 - ◆ More messages
 - ◆ but **not necessary** in lower **lookup latency**

Modeling

- Paper also contains a significant analytical modeling part that allows to
 - ◆ Validate the measurements and our understanding of the operations of KAD
 - ◆ Answer lots of “what-if” questions
 - ◆ Makes for a nice paper to “sell” performance modeling to CS Ph. D. students (don’t let them do simulation:-))

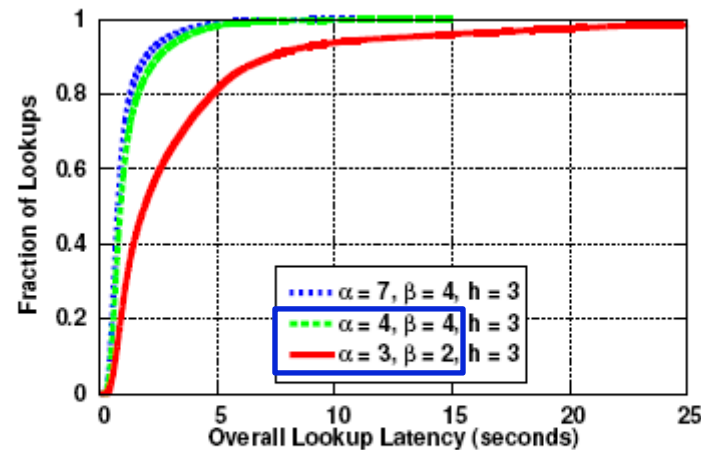


Figure 10. CDF of the Overall Lookup Latency, $F_{ICL}(d)$: qualitative analysis.

Conclusion

- Implementers of KAD needed to make a number of implementation choices fixing some critical parameters such as
 - ◆ alpha, beta, timeout
- Choice of alpha and timeout critical for good performance
 - ◆ alpha =3 is reasonable compromise
- Look-up process is shared by the publish and retrieval module
- Not a good idea
 - ◆ Timeout of t=3 seconds **appropriate** for **publish**
 - ◆ Timeout of t=3 seconds **much to large** for **retrieval**

Bibliography on KAD

- **Faster Content Access in KAD**
Moritz Steiner, Damiano Carra, and Ernst W. Biersack
Proc. of the 8th IEEE Conference on Peer-to-Peer Computing (P2P'08), Aachen, Germany, September 2008.
- **Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm**
Thorsten Holz, Moritz Steiner, Ernst W. Biersack, and Felix Freiling
First Usenix Workshop on Large-scale Exploits and Emergent Threats (LEET), San Francisco, CA, USA, April 2008
- **Building a Reliable P2P System Out of Unreliable P2P Clients: The Case of KAD**
Damiano Carra and Ernst W. Biersack
Proc. of the 3rd International Conference on Emerging Networking Experiments and Technologies (CoNEXT '07), December 2007, New York, USA
- **A global view of KAD**
Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack
Internet Measurement Conference, (IMC 2007), October 2007, San Diego, USA
- **Exploiting KAD: Possible uses and misuses**
Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack
Computer Communications Review, Vol. 37, N°5, October 2007
- **Load Reduction in the KAD Peer-to-Peer System**
Moritz Steiner, Wolfgang Effelsberg, Taoufik En-Najjary, Ernst W. Biersack
Fifth International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P), October 2007, Vienna, Austria

And: **Mortiz Steiner's Ph. D. thesis. Coming soon**

Traces at <http://www.eurecom.fr/~btroup/kadtraces>

Merci

KAD Publish: How and Where

- Where to store the information for a given a key **kID**?
 - ◆ On 11 nodes, who's first 8-bits are the same as **kID**
- How to find these node in 8-bit zone?
 - ◆ Issue iterative `route_req(kID)` to discover nodes in 8-bit zone
 - ✦ Will obtain an ordered (by closeness) list of candidate nodes that are up
 - ✦ Question: When to stop searching for closer nodes???

```
7 When route response is received do
8   timeout = now + t;
9   answered.insert(sender);
10  foreach contact  $\in$  response do
11    if contact not  $\in$  candidates and contact not  $\in$ 
12      queried then
13        candidates.insert(contact);
14        if dist(contact,target) < dist(sender,target)
15          then
16            /* approaching the target */
17            if contact is in the  $\alpha$  closest contacts to the
18              target then
19              send route request(target, $\beta$ ) to
20                contact;
```

KAD Publish: How and Where

- Take nodes from ordered list and send `publish_request` to

```
8      if timeout ≤ now then
      /* candidate list is considered
      stable */
9      for i ← 0 to candidates.size do
10     contact = candidates.get(i);
11     if contact ∈ queried then
12     if contact ∈ answered and contact in
      tolerance zone around target then
      /* the timeout triggers
      the actual content
      search. */
13     send content
      request(TYPE, target) to
      contact;
14     candidate.remove(contact);
```



KAD Search: Node Lookup

- Find nodes in 8-bit zone of key kID?
 - ◆ Issue iterative `route_req(kID)` to discover nodes in 8-bit zone
 - ✦ Will obtain an ordered (by closeness) list of candidate nodes that are up
 - ✦ Question: When to stop searching for closer nodes???

```
7 When route response is received do
8   timeout = now + t;
9   answered.insert(sender);
10  foreach contact  $\in$  response do
11    if contact not  $\in$  candidates and contact not  $\in$ 
12      queried then
13        candidates.insert(contact);
14        if dist(contact,target) < dist(sender,target)
15          then
16            /* approaching the target */
17            if contact is in the  $\alpha$  closest contacts to the
18              target then
19              send route request(target, $\beta$ ) to
20                contact;
```

KAD Search: Content Search

- Take nodes from ordered list and send `search_request` to

```
8      if timeout ≤ now then
      /* candidate list is considered
      stable */
9      for i ← 0 to candidates.size do
10     contact = candidates.get(i);
11     if contact ∈ queried then
12     if contact ∈ answered and contact in
      tolerance zone around target then
      /* the timeout triggers
      the actual content
      search. */
13     send content
      request(TYPE, target) to
      contact;
14     candidate.remove(contact);
```

