Peer-to-peer networks – (due till May 6, 2009)

Exercise 8.1: Routing in Pastry

a) In Pasty, the node closest to the key 5734 is to be found. We assume that the initiator of the query is the „first node“ which has no common prefix with the key. The first nodes makes a lookup in its routing table and forwards the query to a second node which is already closer but not yet responsible for the four digit key 5734. Only the „third node“ should be the final node where the query ends. Fill in the complete routing tables of the three nodes and find appropriate values for their leave sets. We assume that the network is dense enough to populate every cell in the tables.

```
Key: 5734

first node (ID 1322)

routing table
0: 0567 2334 3900 4251 5322 6824 7324
1: 1011 1120 1244 1478 1522 1659 1723
2: 1303 1312 1334 1340 1352 1362 1377
3: 1320 1321 1323 1324 1325 1326 1327

leave set
 : 1259 1320 1321 1323 1324 1325 1326
```

```
second node (ID 5322)

routing table
0: 0063 1537 2250 3915 4333 6730 7206
1: 5011 5112 5203 5480 5519 5636 5734
2: 5301 5310 5332 5347 5358 5365 5370
3: 5320 5321 5323 5324 5325 5326 5327

leave set
 : 5112 5203 5320 5321 5323 5324 5325
```

```
third node (ID 5734)

routing table
0: 0243 1548 2610 3187 4397 6144 7567
1: 5080 5133 5221 5330 5419 5536 5614
2: 5705 5712 5725 5741 5757 5763 5775
3: 5730 5731 5732 5733 5735 5736 5737

leave set
 : 5731 5732 5733 5735 5736 5737 5741
```

Peer-to-peer networks

Exercise 8.1: Routing in Pastry

b) The „second node“ points to another node in (1, 1) (row 1, column 1 / marked with (brackets)). The particular nodes (1, 1) has vanished. How do you fill in the gap? What are you doing if your first, second, ... seventh attempt to fill in the gap fails?

```
second node (ID 5322)

routing table
0: 0063 1537 2250 3915 4333 6730 7206
1: 5011 5112 5203 5480 5519 5636 5734
2: 5301 5310 5332 5347 5358 5365 5370
3: 5320 5321 5323 5324 5325 5326 5327

leave set
 : 5112 5203 5320 5321 5323 5324 5325
```

Solution:

First, the node should ask all other nodes from the same row #1 for their own „one-digit prefix“ row. All their entries are suitable for node 5322 as well since they have the same prefix „5“. If none of them are available, the nodes should start asking the next row #2 for their prefixes in row #1 because their first-digit prefix suits as well. Theoretically, all nodes stored in the following row could be asked as well. In contrast, the previous rows are not necessarily suitable, because they may have different prefixes.

Peer-to-peer networks

Exercise 8.2: Kademlia

a) Kademlia defines the distance between two IDs by applying the XOR operator and interpreting the result as an integer. Show that this actually defines a metrics

Show that:

$$d(x,y) > 0 \; \forall \; x \neq y$$
$$d(x,y) = 0 \Rightarrow x = y$$
$$d(x,y) = d(y,x)$$
$$d(x,y) \leq d(x,z) + d(z,y)$$

$d(x,y) > 0 \; \forall \; x \neq y$: Prove by contradiction

d(x, y) = x XOR y = 0 means that no bit must be set.
bit_1 XOR bit_2 = 0 is only true for 0 XOR 0 and 1 XOR 1.

0 XOR 0 = 0 / 0 XOR 1 = 1
1 XOR 1 = 0 / 1 XOR 0 = 1

Thus, x != y is not possible by definition of XOR.

Peer-to-peer networks

Exercise 8.2: Kademlia

Show that: $d(x,y) > 0 \; \forall \; x \neq y$
$d(x,y) = 0 \Leftrightarrow x = y$
$d(x,y) = d(y,x)$
$d(x,y) \leq d(x,z) + d(z,y)$

d(x,y)=0

For every bit is true that 0 can result only from 0 XOR 0 or 1 XOR 1. What is true bitwise is true for a sequence of bits as well.

d(x,y)=d(y,x) <=> x XOR y = y XOR x

In fact, XOR is commutative as well by definition (see table on last page).

$d(x,y) \leq d(x,z) + d(z,y)$

The following table evaluates all combinations:

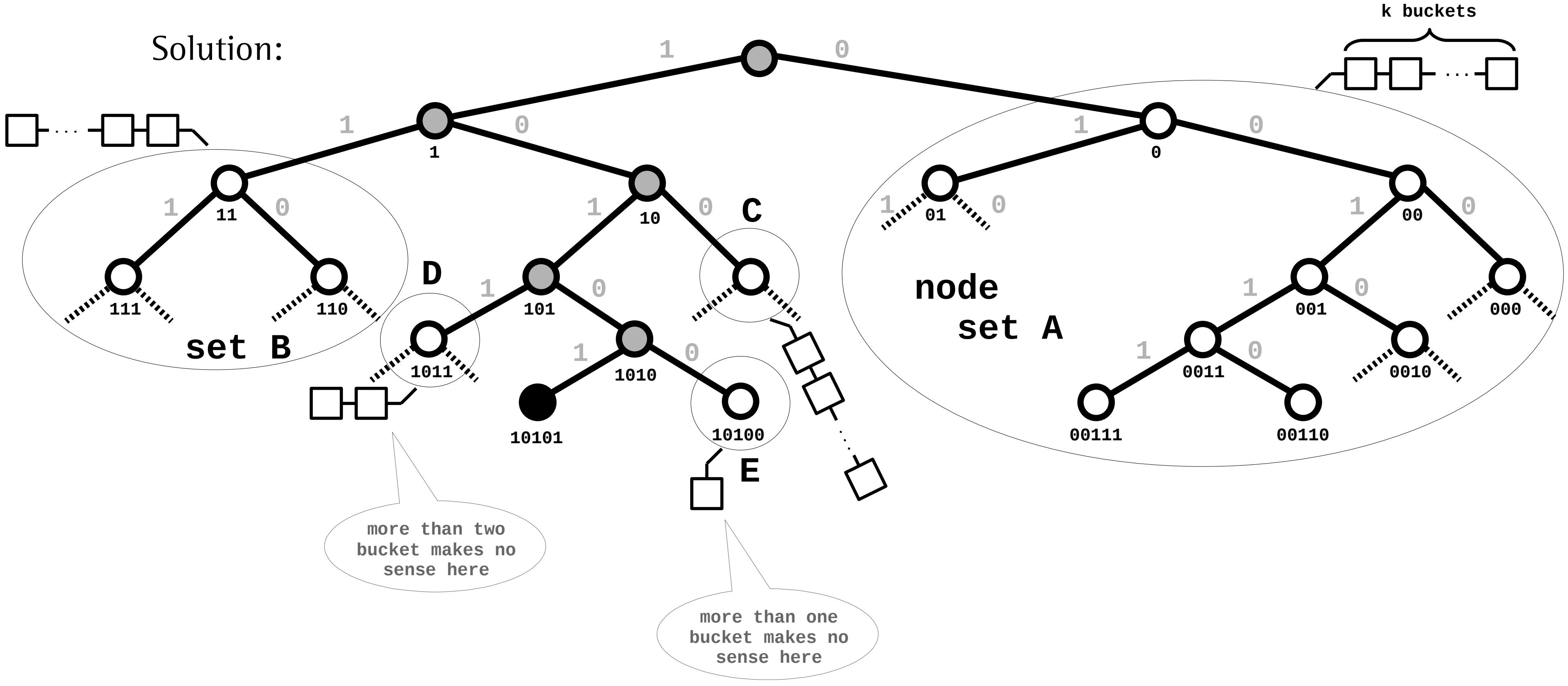
x	z	y	d(x,y)	d(x,z)+d(z,y)
0	0	0	0	0
0	0	1	1	1
0	1	0	0	2
0	1	1	1	1
1	0	0	1	1
1	0	1	0	2
1	1	0	1	1
1	1	1	0	0

Peer-to-peer networks

Exercise 8.2: Kademlia

b) Kademlia's authors states, that a node knows only few neighbors in a far distance and increasingly more neighbors for smaller distances. Explain why.

Solution:



Peer-to-peer networks

Exercise 8.2: Kademlia

Solution (continued):

All nodes in „node set A“ have the largest distance from node 10101 because they differ in the first (and most significant) bit. Even though set A includes half of all nodes on average, it has only k buckets like all other node sets as well. As a consequence, a large number of distant nodes are covered by only few buckets.

The nearer a node set moves to our considered node 10101, the smaller the set. Even though the sets become smaller further down the tree, they are still covered by k buckets. So 10101 has a more detailed knowledge of near nodes as compared to distant nodes.

c) The authors of Kademlia claim that particularly the fact d(x,y)=d(y,x) makes the protocol superior as compared to Chord. Try to find a reason why. (Keep in mind that a node learns from the queries of other nodes.)

Solution:

As we showed in a), the distance between two nodes is a metrics and does not depend on a particular direction. Whenever a node gets a query, it can calculate the distance and store the node in a free bucket of the set the requesting node is included in. Thus, only by listening a node gets to know parts of the network. This is not possible in Chord as only forward routing is considered here. The distance between two nodes depends heavily on their order. Thus, a node can not learn from bypassing massages as easily as Kademlia can.