Lecture 32

- Administration

(1) A. Rowstron and P. Druschel, "*Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001

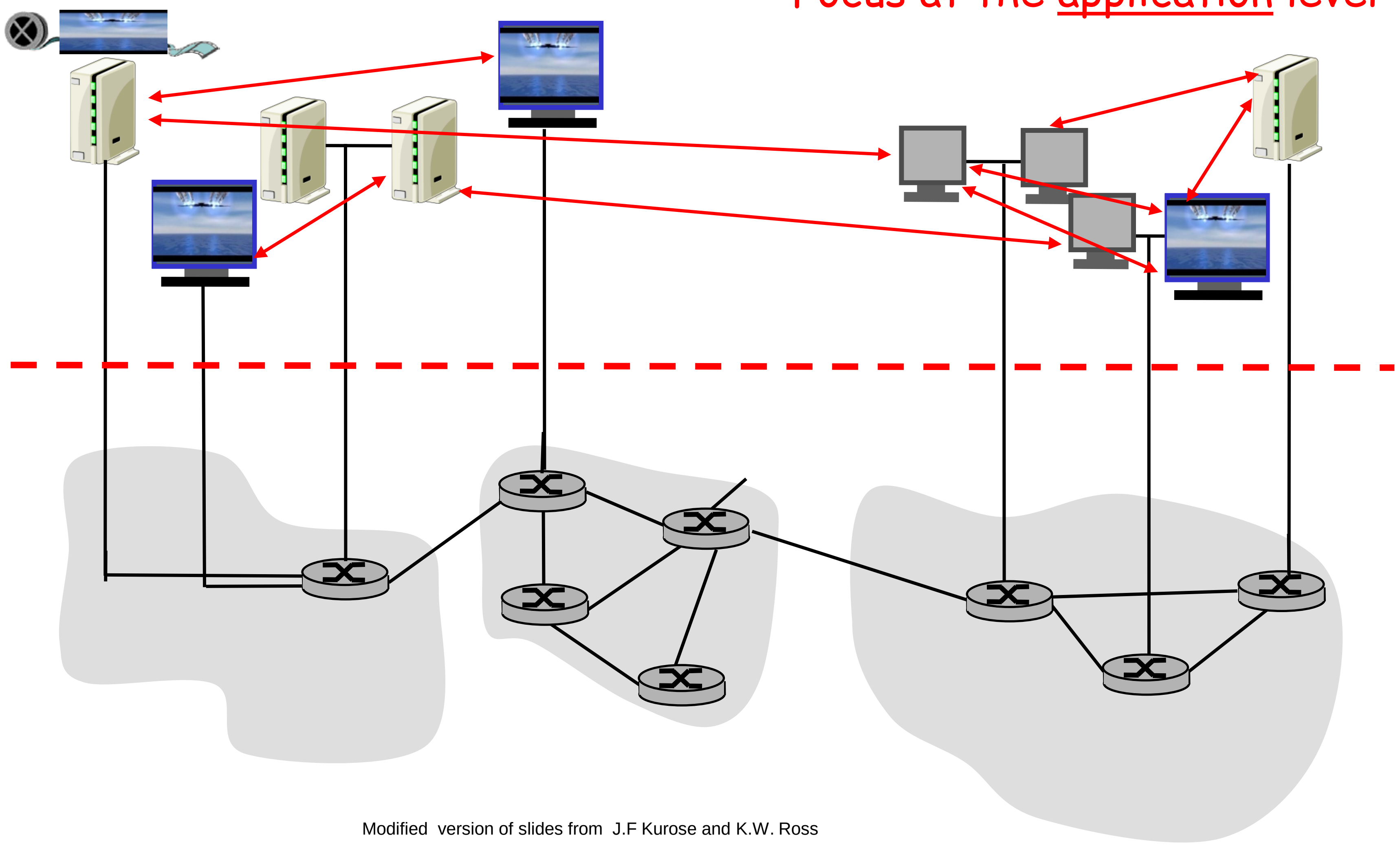
(2) Jeff Odom slides:

http://x1.cs.umd.edu/818/docs/pastry.ppt

Peer-to-Peer

Overlay Networks

Focus at the application level



Introduction: What is Pastry?

- It's a scalable, distributed, decentralized object location and routing substrate

- Serves as a general substrate for building P2P applications: SCRIBE, PAST,...etc.

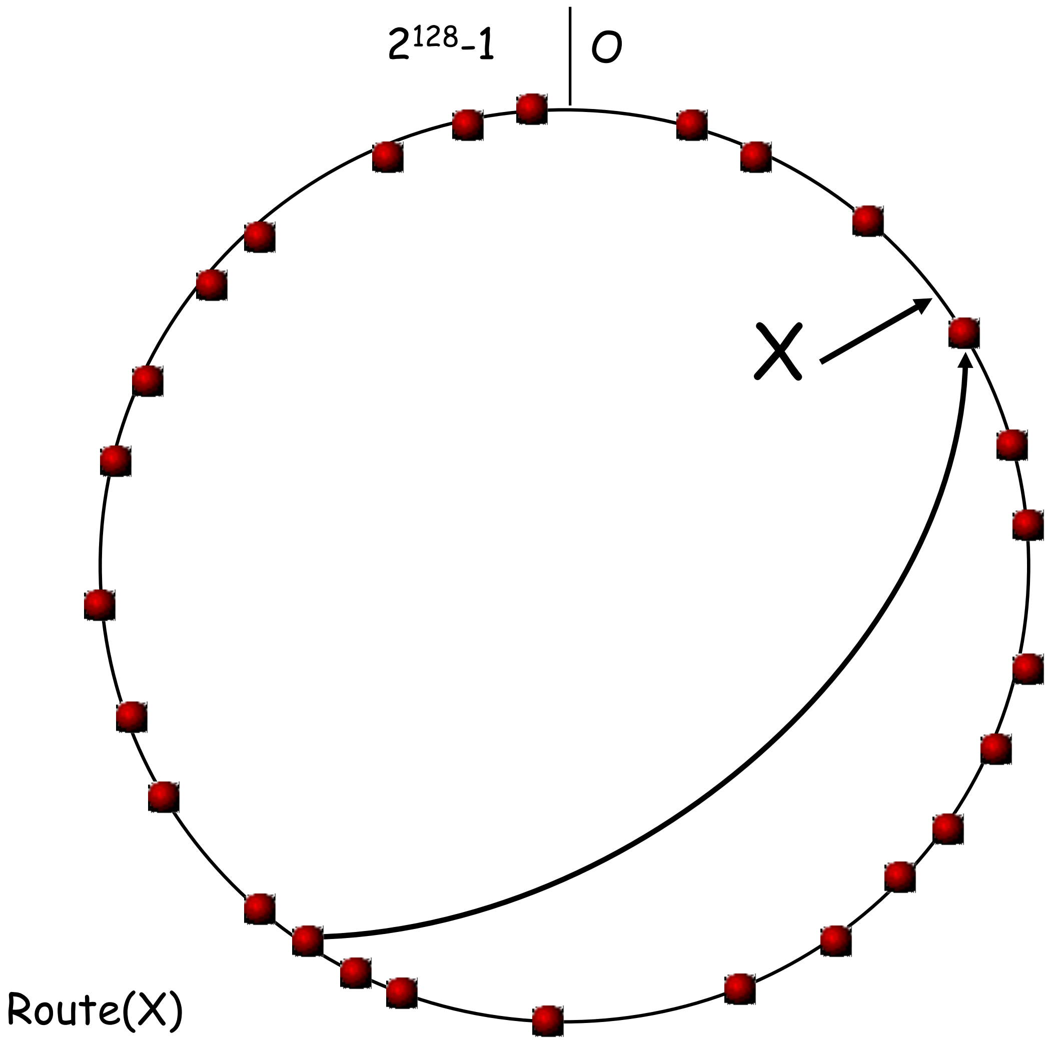
- Seeks to minimize distance messages travel

Pastry

- Decentralized
- Fault-resilient
- Scalable
- Reliable
- Good route locality properties

- Examples:
  - Replicas – geographically separate
  - Pub/sub – nodes used as rendezvous points, keeps subscriber info from node to topicID (key). Reverse spanning tree.

Pastry: Object insertion/lookup



Msg with key *X* is routed to live node with nodeId closest to X

Problem: complete routing table not feasible

Pastry Node

- Represented by 128-bit randomly chosen nodeId (Hash of IP or public key)
- NodeId is in base 2^b (b is a configuration parameter; b typical value 2 or 4)
- Evenly distributed nodeIds along the circular namespace ($0-2^{128} - 1$ space).
- Routes a message in O(log N) steps to destination
  - N: size of network
- Node state contains:
  - Leaf Set (**L**)
  - Routing table (**R**)
  - Neighborhood Set (**M**)

Design of Pastry: *Node state*

- Leaf set: L
  Numerically closest nodes (L is a configuration parameter = 16, 32 typically)
- Routing Table (Prefix-based)
- Neighborhood Set: M physically closest nodes

NodeId 10233102

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Routing table

-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

commonPrefix - nextDigit - restofID

Pastry node state (Leaf Set)

Leaf set	SMALLER	LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

- Serves as a fall back for routing table and contains:
  - L/2 numerically closest and larger nodeIds
  - L/2 numerically closest and smaller nodIds
- Size of L is typically 2^b or 2×2^b
- Nodes in L are numerically close (could be geographically diverse)

Pastry node state: *Neighborhood set (M)*

- Contains the IP addresses and nodeIds of closest nodes according to proximity metric
- Size of |M| is typically 2^b or 2×2^b
- Not used in routing, but instead for maintaining locality properties

Neighborhood set			
13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

Node state: *Routing Table*

- Matrix of **$\log_{2^b}$ N** rows and **2^b – 1** columns (N is the number of nodes in the network)
- Entries in row *n* match the first *n* digits of current nodeId AND
- Column number follows matched digits: Format: matched digits–column number–rest of ID
- **$\log_{2^b}$ N** populated on average

Node10233102 $^{(2)}$, ($b = 2$, $l = 8$)

0	1	2	3
02212102		22301203	31203203
	11301233	**1**2230203	**1**3021022
10031203	**10**132102		**10**323302
10200230	**102**11302	**102**2302	
10230322	**1023**1000	**1023**2121	
10233001		**10233**232	
		10233120	

Routing$^{(2)}$:

- If message with key D is within range of leaf set, forward to numerically closest leaf
- Else forward to node that shares at least one more digit with D in its prefix than current nodeId
- If no such node exists, forward to node that shares at least as many digits with D as current nodeId but numerically nearer than current nodeId

Routing Messages

if $(L_{-\lfloor |L|/2 \rfloor} \leq D \leq L_{\lfloor |L|/2 \rfloor})$ {

    // D is within range of our leaf set

    forward to L_i, s.th. $|D - L_i|$ is minimal;

} else {

    // use the routing table

    Let $l = shl(D, A)$;

    if $(R_l^{D_l} \neq null)$ {

        forward to $R_l^{D_l}$;

    }

    else {

        // rare case

        forward to $T \in L \cup R \cup M$, s.th.

            $shl(T, D) \geq l$,

            $|T - D| < |A - D|$

    }

}

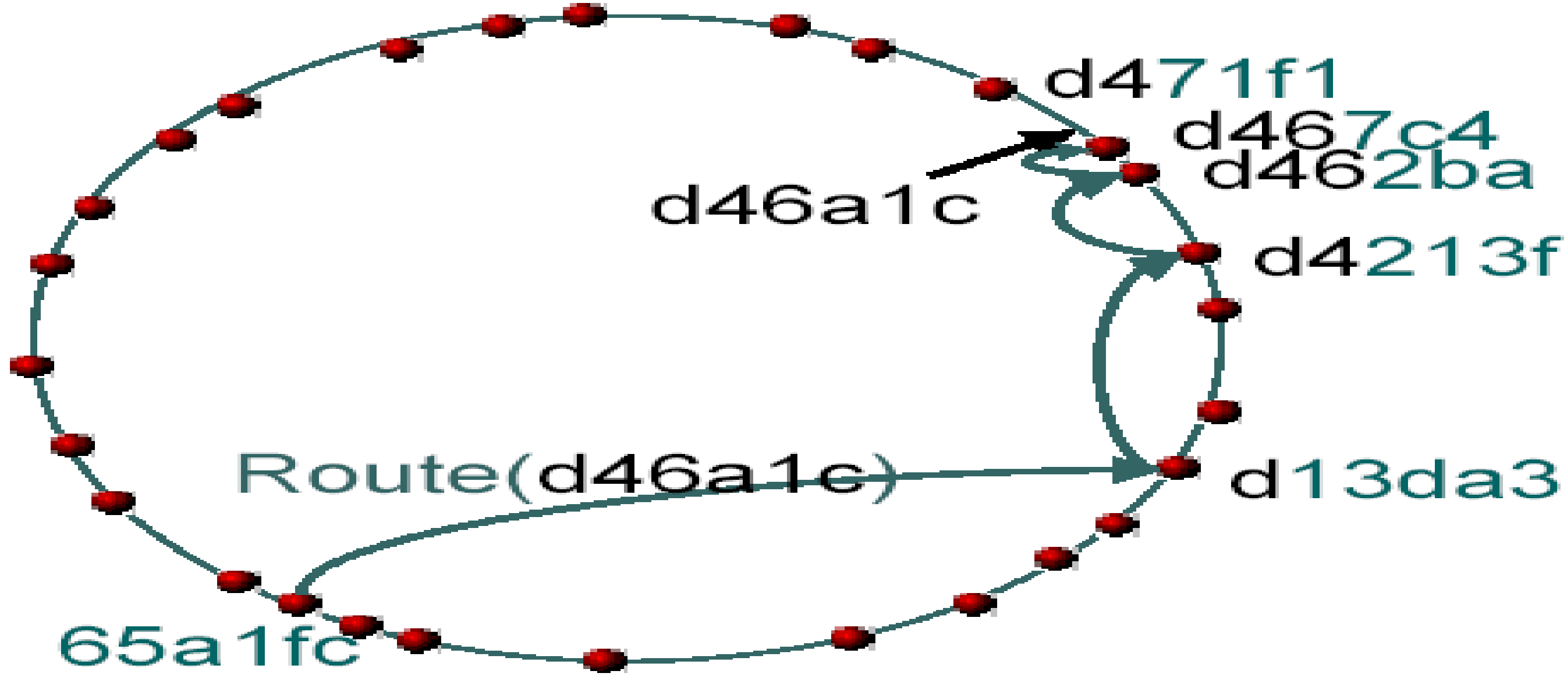
(1) Node is in the leaf set

(2) Forward message to a closer node (Better match)

(3) Forward towards numerically Closer node (not a better match)

D: Message Key
L_i: i^{th} closest NodeId in leaf set
shl(A, B): Length of prefix shared by nodes A and B
R^i_j: $(j, i)^{th}$ entry of routing table

Routing Example:



Routing Performance:

- (1) If key is within leaf set:
  - target one hop away
- (2) If key has to be routed:
  - Number of nodes with longer prefix decreases by 2^b
- (3) Key is not covered by the leaf set (i.e., failed)
  - With high probability, one more hop needed
- Thus: Number of routing steps needed $\lceil \log_{2^b} N \rceil$

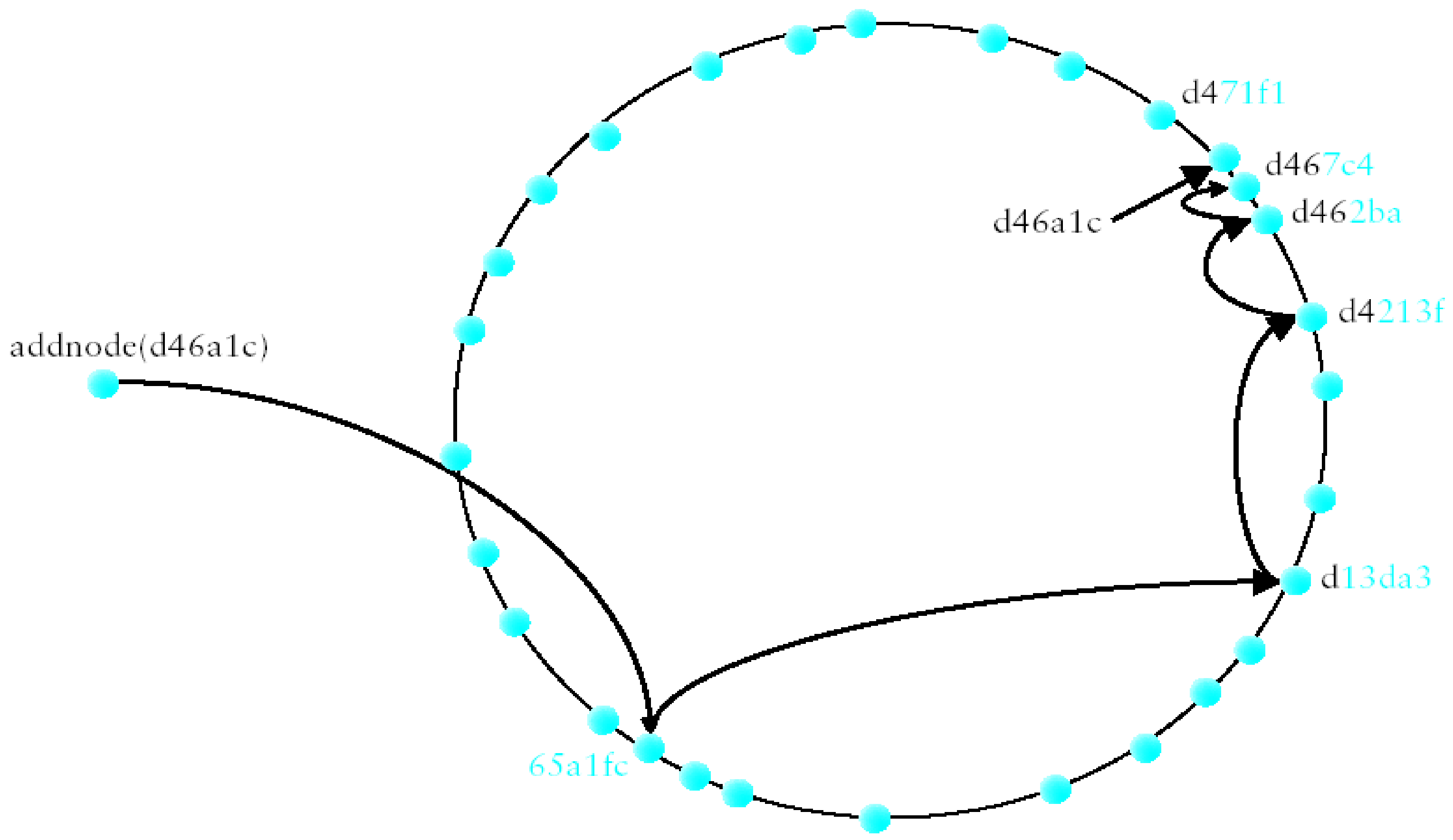
Pastry node join

- X = new node, Z = numerically closest node, A = bootstrap (A is close in proximity space to X)
- X sends a join message to A with target nodeId X
- A forwards to B →C→…
- Stops at Z, numerically closest to X's nodeId
- In process, A,B,…,Z send their state tables to X

Node Join

- X's neighborhood set (NS) = A's NS
- X's Leaf Set = Z's leaf set
- X's routing table is filled as follows:
  - X's Row 0 = A's row 0 ($X_0 = A_0$)
  - X's Row 1 = B's row 1 ($X_1 = B_1$)
  - ...etc.
- X sends its state to every node in its state tables (Leaf set, neighborhood set, and routing table)

Node Join: Example



Node departure (2)

- Invalid nodes in leaf set: detected by heartbeat monitor
  - Repair by inserting node from another leaf's LS
- Heartbeat for neighborhood set (NS)
  - Query all NS members for their NS tables, choose replacement according to proximity metric
- Invalid routing entries detected when attempting to route
  - Query nodes in row for replacement entry, if failed
  - Query successive rows until success

Node failure in routing table: example

If node in red fails

0x	1x	2x	3x	4x	5x		7x	8x	9x	ax	bx	cx	dx	ex	fx
60x	61x	62x	63x	64x		66x	67x	68x	69x	6ax	6bx	6cx	6dx	6ex	6fx
650x	651x	652x	653x	654x	655x	656x	657x	658x	659x		65bx	65cx	65dx	65ex	65fx
65a0x		65a2x	65a3x	65a4x	65a5x	65a6x	65a7x	65a8x	65a9x	65aax	65abx	65acx	65adx	65aex	65afx

Try asking some neighbor in the same row for its 655x entry

If it doesn't have one, try asking some neighbor in the row below, etc.

Locality in Pastry

- Based on proximity metric (i.e., No. of IP hops, geographic distance)
- Proximity space is assumed to be Euclidean
- The route chosen for a message is likely to be "**good**" with respect to the proximity metric
- We will discuss locality regarding:
  - Routing table locality
  - Route locality
  - Locating the nearest among *k* nodes

Summary

- Pastry is a generic P2P object location and routing substrate
- Distributed, and scales well
- Used in developing applications like file storage, global file sharing,...etc.
- Considers locality when routing messeges