CSx55: Distributed Systems [DHTs]

Shrideep Pallickara
Computer Science
Colorado State University

COMPUTER SCIENCE DEPARTMENT COLORADO STATE UNIVERSITY

1

Frequently asked questions from the previous class survey

- If you keep searching for nodes to populate your table, don't you already know about the network at that point?
- Diversity of routing paths if two (neighboring) peers have nearly identical (off by 1) leafsets?
- How does node **A** discover the conduit node **X** in Pastry?
- Replication in Pastry?

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT PEER-TO-PEER SYSTEMS L17.2

2

Topics covered in this lecture

- Pastry wrap-up
- Tapestry
- Unstructured P2P Systems

3

4

Detection and coping with node failures

- When a node's immediate neighbors (in the GUID space) cannot communicate with it?
  - The node is considered failed
- Necessary to **repair** leaf sets and routing tables that contain the failed GUID
  - Leaf sets are repaired *proactively*
  - Routing tables at the other nodes are updated on a "*when discovered basis*"

Repairing leaf sets

- Node that discovers the failure
  - Looks for a live node close to the failed node, and requests copy of that node's leaf set, L'
  - This should contain GUIDs that partly overlap those in the node that discovered failure
    - Include one that should replace the failed node
- Other neighboring nodes are informed
  - They perform a similar procedure

Locality

- Pastry's routing structure is redundant
  - Multiple routes between pairs of nodes
- Construction of routing tables tries to take advantage of this redundancy
  - Reduce message transmission times by exploiting locality properties of underlying network

Routing table: Exploiting locality. [1/2]

- In the routing table, each row contains 16 entries
  - Entries in the i^{th} row give addresses of 16 nodes with GUIDs with $i-1$ initial hexadecimal digits
  - I^{th} digit takes each of the possible hexadecimal values
- Well-populated Pastry system contains more nodes than can be contained in an individual routing table

Routing table: Exploiting locality. [2/2]

- When routing table is constructed, a choice is made for each position
  - Between multiple candidates
  - Based on *proximity* neighbor selection
- Locality metric
  - IP hops or measured latency

Performance of exploiting locality

- Since the information in the routing table is not comprehensive
  - Mechanism does not produce globally optimal routing
- Simulations show that
  - On average, the routing is 30-50% longer than the optimum

Coping with malicious nodes

- Small degree of *randomness* is introduced into route selection
- Randomized to yield a common prefix that is less than the maximum length
  - With a certain probability
- Routes are taken from an earlier row
  - Less optimal, but different than standard version
  - Client transmission succeed in the presence of small numbers of malicious nodes

11

TAPESTRY

12

Tapestry

- Routes messages to nodes based on GUIDs associated with the resources
  - Uses **prefix routing** in a manner similar to Pastry
- **160-bit** identifiers are used
  - To refer to both objects and nodes that perform routing actions
- For any resource with GUID G, there is a unique root node, with GUID R_G
  - R_G is *numerically closest* to G

Tapestry Routing [Summary]

- Uses local routing tables, which they also call **neighbor maps**, to route messages
- Routing is digit-by-digit
  - 4*** → 42** → 42A* → 42AD
- This longest prefix routing is also used by classless interdomain routing (CIDR)

Tapestry: Routing messages

- Each node maintains a **routing table**
  - Entries include nodeIDs and IP addresses
- This routing table has multiple levels
  - Each level contains links to nodes matching a prefix up to a digit position in the ID
  - The i^{th} entry in the j^{th} level at node **N**?
    - Location of the closest node which begins with the $prefix(\mathbf{N}, j\text{-}1) + i$
    - E.g., 9^{th} entry of the 4^{th} level for node `325AE` is ?
      - 3259

Tapestry Routing

- The router for the n^{th} hop
  - Shares a prefix of length $\geq n$ with the destination ID
  - Looks in its $(n+1)^{th}$ level map for entry matching the next digit in the destination ID
- Guarantees that any node in the system can be reached in at most *log N* logical hops
  - N is the size of the ID space i.e. $N = 2^{160}$

When a digit cannot be matched?

- Looks for a "close" digit in the routing table
- This approach is called **surrogate routing**
  - Results in mapping every identifier **G** to a unique root node G_R

Managing a dynamic environment

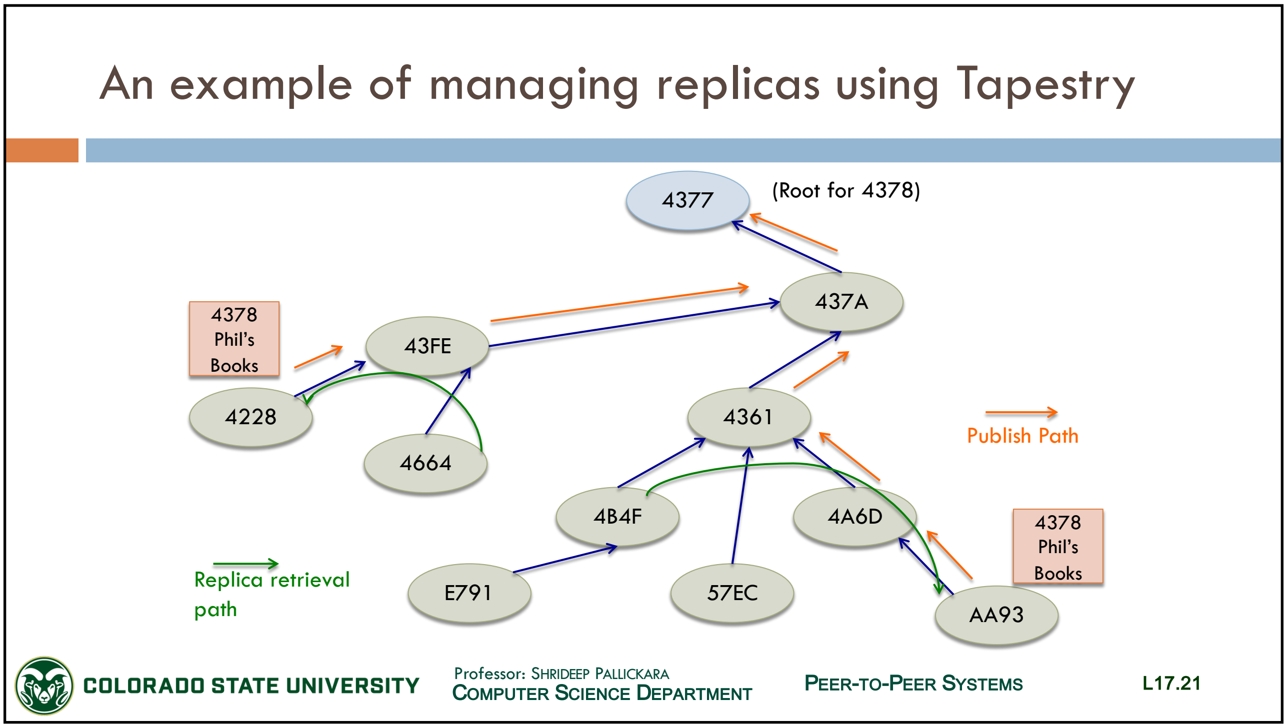
- Route reliably even when intermediate links are changing or faulty
- Exploit network **path diversity**
  - Via *redundant* routing paths
- Primary links are augmented by **backup-links**
  - Each sharing the same prefix

Managing multiple copies of the resource

- Hosts H holding replicas of **G** periodically invoke $publish(G)$
  - Ensures that newly arrived hosts become aware of the existence of **G**
- On each invocation of $publish(G)$
  - Message is routed from invoker towards node R_G
  - On receipt of a publish message R_G enters (G, IP_H)
    - The mapping between **G** and IP address of H
  - Each node in the publication path caches the same mapping

Managing multiple copies of the resource

- When nodes hold multiple (**G**, IP) mappings for the same GUID?
  - They are **sorted** by network distance to the IP address
- Results in *selection of nearest* available replica of the object


An example of managing replicas using Tapestry
4377
(Root for 4378)
437A
4378
Phil's
Books
43FE
4228
4664
4361
Publish Path
4B4F
4A6D
4378
Phil's
Books
Replica retrieval
path
E791
57EC
AA93
COLORADO STATE UNIVERSITY
Professor: SHRIDEEP PALLICKARA
COMPUTER SCIENCE DEPARTMENT
PEER-TO-PEER SYSTEMS
L17.21

21


UNSTRUCTURED P2P SYSTEMS
L8.22

22

Structured P2P systems [Summary]

- Overall **global policy** governing
  - Topology of the network
  - Placements of objects
  - Routing functions to locate objects
- There is a specific **distributed data structure** that underpins
  - Associated Overlay
  - Algorithms that operate on it to route messages

Structured P2P systems [Summary]

- Because of the structure, algorithms are
  - Efficient
  - Offer *time-bounds* on object location
- BUT involve **costly maintenance** of underlying structures
  - In highly dynamic environments

Unstructured P2P systems [1/2]

- Target the maintenance argument
- No overall control on
  - Topology
  - Placements of objects within the network
- Overlay is created in an ***ad hoc*** manner
  - Each node joins network by following simple, local rules to establish connectivity

Unstructured P2P systems [2/2]

- A new joining node will establish contact with a set of *neighbor* nodes
  - These neighbors will be connected to further neighbors, etc.
- The network is fundamentally **decentralized** and self-organizing
  - Resilient to failures

Locating objects in unstructured P2P systems

- Requires a search of the resultant network topology
- **No guarantees** of being able to find the object
  - Performance will also be unpredictable
  - There is a risk of generating *excessive message traffic* to locate objects

Pros and Cons

	Structured P2P	Unstructured P2P systems
Advantages	Guaranteed to locate objects with bounds on this operation. Low message overhead	Self-organizing and naturally resilient to failures
Disadvantages	Maintain complex overlay structures that are difficult and costly in dynamic settings	Probabilistic Cannot offer absolute guarantees on locating objects

29

Sharing in unstructured P2P networks

- All nodes in the network offer files to the greater environment
- Problem of locating a file?
  - Maps onto a *search of the whole network*
- CAVEAT:
  - If implemented naively, could result in **flooding** the network with requests

COLORADO STATE UNIVERSITY　Professor: SHRIDEEP PALLICKARA　COMPUTER SCIENCE DEPARTMENT　PEER-TO-PEER SYSTEMS　L17.30

30

Refinements for search in unstructured P2P systems

- Expanded ring search
- Random walks
- Gossiping
- Replication

Refinements for search in unstructured P2P systems: Expanded Ring Search

- Initiating node carries out a series of searches with *increasing values* in the TTL field
- A significant number of searches will likely be satisfied locally (proximate peers)
  - Expand the scope of search only if requests fail in the neighborhood

Refinements for search in unstructured P2P systems: Random Walks

- Initiating node sets of a number of **walkers**
- Walkers follow *random pathways* through the interconnected graph
  - Over the unstructured network

Refinements for search in unstructured P2P systems: Gossiping [1/2]

- Node sends request to a neighbor with a certain probability
- Requests propagate through the network in a manner that is similar to **viral propagations**
  - Such gossip protocols are also referred to as *epidemic protocols*

Refinements for search in unstructured P2P systems: Gossiping [2/2]

- Probabilities may either be
  - **Fixed** for a given network
  - Computed **dynamically** based on:
    - Past experience
    - Current context

Refinements for search in unstructured P2P systems: Replication

- **Replicate** content across a number of peers
- *Probability* of efficient discovery for particular files is enhanced
- Replications can be for
  - The entire file
  - Fragments thereof

GNUTELLA

COLORADO STATE UNIVERSITY

37

Gnutella

- Launched in 2000
- One of the most dominant and influential peer-to-peer file sharing applications

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT PEER-TO-PEER SYSTEMS L17.38

38

Gnutella: Early Versions (0.4)

- Every node forwarded a request to **each** of its neighbors
- Neighbors, in turn, passed this on to their neighbors
  - Until a match was found
- This is **flooding**

Gnutella: Early Versions (0.4)

- Search was **constrained** with a *time-to-live* (TTL) field limiting the number of hops
- At the time of Version 0.4, average peer connectivity was 5 neighbors per-node

GNUTELLA VERSION 0.6 AND LATER

Addressing deficiencies in scaling: Hybrid Architecture [1/2]

- Move away from classic P2P where all nodes are equal
- Some nodes are elected as **ultrapeers**
  - Form the heart of the network
- Other nodes take on the role of **leaf nodes**
- Peers still cooperate to offer service

Addressing deficiencies in scaling: Hybrid Architecture [2/2]

- Leaves connect to a small number of **ultrapeers**
- Ultrapeers are *densely connected* to other ultrapeers
- Effect?
  - Dramatically *reduces* the <u>maximum number of hops</u> for exhaustive search

Query Routing Protocol [1/2]

- Designed to **reduce** the number of queries issued by nodes
- **Exchange information** about files contained on nodes
- **Forward queries** down paths where the system thinks there will be a positive outcome

Query Routing Protocol [2/2]

- Does not share information about files directly
- Protocol produces **set of numbers**
  - By *hashing on individual words* in a file-name
  - For e.g., "`Gone with the wind`" will be represented as <`36, 789, 452, 132`>
- Each node produces a **Query Routing Table**
  - Contains hash values representing *each of the files* contained on that node
  - Sends it to all its associated ultrapeers

Query Routing Protocol: Ultrapeers

- Ultrapeers produce their own Query Routing Table
  - **Union** of all entries *from all connected leaves*; together with entries for files at that ultrapeer
- The ultrapeer then **exchanges** its Query Routing Table with other ultrapeers

Implications of exchanging the Query Routing Table

- Ultrapeers can determine which paths offer a **valid route** for a given query
  - Significantly reduces amount of unnecessary traffic
- Ultrapeer **forwards** a query to a node *only if a match is found*
  - Match indicates that the node has the file
  - Same check performed before forwarding query to another ultrapeer

Avoid overloading the ultrapeers

- Nodes send query to *one* ultrapeer at a time
  - Wait for a specified time period
- ***Avoid reverse traversal*** of messages through the graph
  - Queries in Gnutella contain network address of the initiating ultrapeer
  - File sent directly (using UDP) to that ultrapeer

The contents of this slide-set are based on the following references

- *Distributed Systems: Principles and Paradigms. Andrew S. Tanenbaum and Maarten Van der Steen. 2nd Edition. Prentice Hall. ISBN: 0132392275/978-0132392273.* [Chapter 5]
- *Distributed Systems: Concepts and Design. George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair. 5th Edition. Addison Wesley. ISBN: 978-0132143011.* [Chapter 10]