Qualifying Examination

A Decentralized Location and Routing Infrastructure for Fault-tolerant Wide-area Applications

John Kubiatowicz (Chair) Satish Rao
Anthony Joseph John Chuang

Outline

- Motivation
- Tapestry Overview
- Examining Alternatives
- Preliminary Evaluation
- Research Plan

New Challenges in Wide-Area

- Trends:
  - Moore's Law growth in CPU, b/w, storage
  - Network expanding in reach and bandwidth
  - Applications poised to leverage network growth
- Scalability: # of users, requests, traffic
- Expect failures everywhere
  - 10^6's of components → MTBF decreases geometrically
- Self-management
  - Intermittent resources → single centralized management policy not enough
- Proposal: solve these issues at infrastructure level so applications can inherit properties transparently

Clustering for Scale

- Pros:
  - Easy monitor of faults
  - LAN communication
    - Low latency
    - High bandwidth
  - Shared state
  - Simple load-balancing
- Cons:
  - Centralized failure:
    - Single outgoing link
    - Single power source
    - Geographic locality
  - Limited scalability
    - Outgoing bandwidth
    - Power
    - Space / ventilation

Global Computation Model

- Leverage proliferation of cheap computing resources: cpu's, storage, b/w
- Global self-adaptive system
  - Utilize resources wherever possible
  - Localize effects of single failures
  - No single point of vulnerability
- Robust, adaptive, persistent

Global Applications?

- Fully distributed share of resources
  - Storage: OceanStore, Freenet
  - Computation: SETI, Entropia
  - Network bandwidth: multicast, content distribution
- Deployment: application-level protocol
- Redundancy at every level
  - Storage
  - Network bandwidth
  - Computation

Key: Routing and Location

- Network scale → stress on location / routing layer
- Wide-area decentralized location and routing on an overlay
- Properties abstracted in such a layer
  - Scalability: million nodes, billion objects
  - Availability: survive routine faults
  - Dynamic Operation: self-configuring, adaptive
  - Locality: minimize system-wide operations
  - Load balanced operation

Research Issues

- Tradeoffs in performance vs. overhead costs
  - Overlay routing efficiency vs. routing pointer storage
  - Location locality vs. location pointer storage
  - Fault-tolerance and availability vs. storage, bandwidth used
- Performance stability via redundancy
- Not:
  - Application consistency issues
  - Application level load partitioning

Outline

- Motivation
- Tapestry Overview
- Examining Alternatives
- Preliminary Evaluation
- Research Plan

What is Tapestry

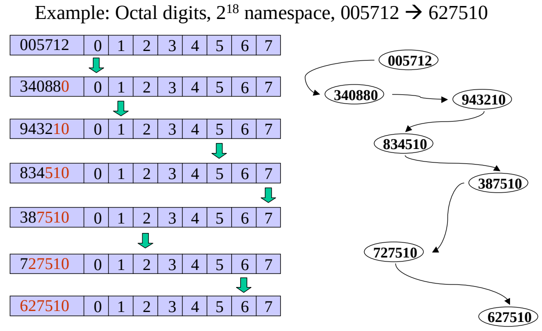
- A prototype of a dynamic, scalable, fault-tolerant location and routing infrastructure
- Suffix-based hypercube routing
  - Core system inspired by PRR97
- Publish by reference
- Core API:
  - *publishObject(ObjectID, [serverID])*
  - *msgToObject(ObjectID)*
  - *msgToNode(NodeID)*

Plaxton, Rajamaran, Richa '97

- Overlay network with randomly distributed IDs
  - Server (where objects are stored)
  - Client (which want to search/contact objects)
  - Router (which forwards messages from other nodes)
- Combined location and routing
  - Servers "publish / advertise" objects they maintain
  - Messages route to nearest server given object ID
- Assume global network knowledge

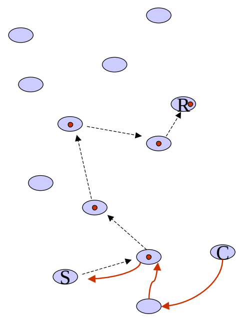
Basic Routing

Example: Octal digits, 2^{18} namespace, 005712 → 627510



Publish / Location

- Each object has associated root node, e.g. identity *f( )*
- Root keeps a pointer to object's location
- Object *O* stored at server *S*
  - *S* routes to Root(*O*)
  - Each hop keeps <*O*,*S*> in index database
- Client routes to Root(*O*), route to *S* when <*O*,*S*> found



What's New

PRR97

- Benefits:
  - Simple fault-handling
  - Scalable: state: $bLog_b(N)$, hops: $Log_b(N)$
    b=digit base, N= |namespace|
  - Exploits locality
  - Proportional route distance
- Limitations
  - Global knowledge algorithms
  - Root node vulnerability
  - Lack of adaptability

Tapestry

- Inherited:
  - Scalability
  - Exploits locality
  - Proportional route distance
- New:
  - Distributed algorithms
  - Redundancy for fault-tolerance
  - Redundancy for performance
  - Self-configuring / adaptive

Fault-resilience

- Minimized soft-state vs. explicit fault-recovery
- Routing
  - Redundant backup routing pointers
  - Soft-state neighbor probe packets
- Location
  - Soft-state periodic republish
    - 50 **million** files/node, daily republish, $b = 16$, $N = 2^{160}$, 40B/msg, worst case update traffic → 156 kb/s,
    - expected traffic for network w/ 2^{40} nodes → 39 kb/s
  - Hash objectIDs for multiple roots
    - P(findingReference w/ partition) = $1 - (1/2)^n$ where n = # of roots

Dynamic "Surrogate" Routing

- Real networks much smaller than namespace
  - sparseness in the network
- Routing to non-existent node
  (or, defining $f: (N) \rightarrow (n)$, where
  N = namespace, n = set of nodes in network)
- Example:
  Routing to root node of object *O*
  Need mapping from $N \rightarrow n$

PRR97 Approach to $f(N_i)$

- Given desired ID N_i,
  - Find set *S* of nodes in existing network nodes *n* matching most # of suffix digits with N_i
  - Choose S_i = node in *S* with highest valued ID
- Issues:
  - Mapping must be generated statically using global knowledge
  - Must be kept as hard state in order to operate in changing environment
  - Mapping is not well distributed, many nodes in *n* get no mappings

Tapestry Approach to $f(N_i)$

- Globally consistent distributed algorithm:
  - Attempt to route to desired ID N_i
  - Whenever null entry encountered, choose next "higher" non-null pointer entry
  - If current node *S* is only non-null pointer in rest of route map, terminate route, $f(N_i) = S$
- Assumes:
  - Routing maps across network are up to date
  - Null/non-null properties identical at all nodes sharing same suffix

Analysis of Tapestry Algorithm

Globally consistent deterministic mapping

- Null entry → no node in network with suffix
- $\therefore$consistent map → identical null entries across same route maps of nodes w/ same suffix

Additional hops compared to PRR solution:

- Reduce to coupon collector problem
  Assuming random distribution
- With $n * ln(n) + cn$ entries, P(all coupons)= $1\text{-}e^{-c}$
- For $n=b$, $c=b\text{-}ln(b)$, P(b^2 nodes left) = $1\text{-}b/e^b = 1.8 * 10^{-6}$
- # of additional hops $\cong Log_b(b^2) = 2$

Distributed algorithm with minimal additional hops

Properties of Overlay

- Logical hops through overlay per route
- Routing state per overlay node
- Overlay routing distance vs. underlying network
  - Relative Delay Penalty (RDP)
- Messages for insertion
- Load balancing

Alternatives: P2P Indices

- Current Solutions:
  - DNS server redirection, DNS peering
- Content Addressable Networks
  - Ratnasamy et al., ACIRI / UCB
- Chord
  - Stoica, Morris, Karger, Kaashoek, Balakrishnan MIT
- Pastry
  - Druschel and Rowstron Microsoft Research

Comparing the Alternatives

Properties	TAP.	Chord	CAN	Pastry
Parameter	Base b	None	Dimen d	Base b
Logical Path Length	$Log_b(N)$	$Log_2(N)$	$O(d*N^{1/d})$	$Log_b(N)$
Neighbor-state	$bLog_b(N)$	$Log_2(N)$	$O(d)$	$bLog_b(N)+O(b)$
Routing Overhead (RDP)	$O(1)$	→$O(1)$	$O(1)$?	$O(1)$?
Messages to insert	$O(Log_b^2(N))$	$O(Log_2^2(N))$	$O(d*N^{1/d})$	$O(Log_b(N))$
Consistency	App-dep.	Eventual	Epoch	???
Load-balancing	Good	Good	Good	Good

$\underbrace{\text{Chord, CAN, Pastry}}_{\text{Designed for P2P Systems}}$

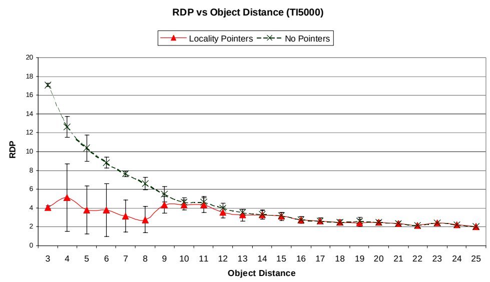
Outline

- Motivation
- Tapestry Overview
- Examining Alternatives
- Preliminary Evaluation
- Research Plan

Evaluation Metrics

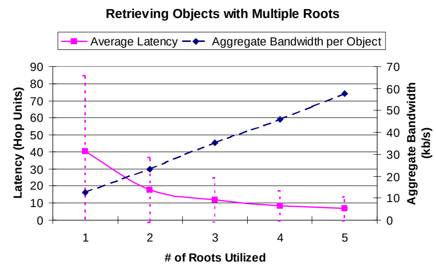
- Routing distance overhead (RDP)
- Routing redundancy → fault-tolerance
  - Availability of objects and references
  - Message delivery under link/router failures
  - Overhead of fault-handling
- Locality vs. storage overhead
- Optimality of dynamic insertion
- Performance stability via redundancy

Results: Location Locality



- Measuring effectiveness of locality pointers (TIERS 5000)

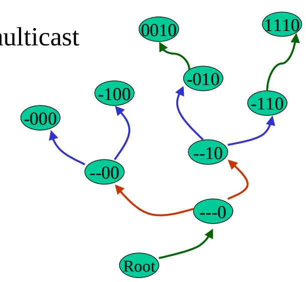
Results: Stability via Redundancy



- Impact of parallel queries on multiple roots on response time and its variability. Aggregate bandwidth measures b/w used for softstate republish 1/day and b/w used by requests at rate of 1/s.

Example Application: Bayeux

- Application-level multicast
  - Leverages Tapestry
    - for scale
    - fault-tolerance
  - Optimizations
    - Self-configuring into sub-trees
    - Group ID clusters for lower b/w



Research Scope

- Effectiveness as application infrastructure
  - Build new novel apps
  - Port existing apps to scale to wide-area
- Use simulations to better understand parameters effects on overall performance
- Explore further stability via statistics
- Understand / map out research space
- Outside scope:
  - DoS resiliency
  - Streaming media, P2P, content-distribution apps

Timeline 0-5 months

- Simulation/analysis of parameters impact on performance
- Quantify approaches to exploit routing redundancy, analyze via simulation
- Finish deployment of real dynamic Tapestry
- Consider alternate mechanisms
  - Learn from consistent hashing

Timeline 5-10 months

- Extend deployment to wide-area networks
  - Nortel, EMC, academic institutions
  - Evaluate real world performance
- Design and implement network-embedded SDS (w/ T. Hodes)
- Optimizing routing by fault prediction
  - Integrate link-characterization work (Konrad)
- Start writing dissertation

Timeline 10-13 months

- Finish writing dissertation
- Travel / Interviews
- Graduate