Brocade: Landmark Routing on Peer to Peer Networks

Ben Y. Zhao
Yitao Duan, Ling Huang, Anthony Joseph, John Kubiatowicz

IPTPS, March 2002

State of the Art Routing

- High dimensionality and coordinate-based P2P routing
  - Tapestry, Pastry, Chord, CAN, etc…
  - Sub-linear storage and # of overlay hops per route
  - Properties dependent on random name distribution
  - Optimized for uniform mesh style networks

Reality

- Transit-stub topology, disparate resources per node
- Result: Inefficient inter-domain routing (b/w, latency)

Talk Outline

- Motivation
- Brocade Architecture
- Brocade Routing
- Evaluation
- Summary / Open Questions

Brocade: Landmark Routing

- Goals
  - Eliminate unnecessary wide-area hops for inter-domain messages
  - Eliminate traffic going through high latency, congested stub links
  - Reduce wide-area bandwidth utilization
  - Maintain interface: RouteToID (globally unique ID)

Brocade Architecture
Mechanisms
- Intuition: route quickly to destination domain
  - Organize group of supernodes into secondary overlay
  - Sender (S) sends message to local supernode SN1
  - SN1 finds and routes message to supernode SN2 near receiver R
    - SN1 uses Tapestry object location to find SN2
    - SN2 sends message to R via normal routing

Classifying Traffic
- Brocade not useful for intra-domain messages
  - P2P layer should exploit some locality (Tapestry)
  - Undesirable processing overhead
  - Classifying traffic by destination
    - Proximity caches:
      - Every node keeps list of nodes it knows to be local
      - Need not be optimal, worst case: 1 relay through SN
    - Cover set:
      - Supernode keeps list of all nodes in its domain.
      - Acts as authority on local vs. distant traffic

Entering the Brocade
- Route: Sender → Supernode (Sender)?
  - IP Snooping brocade
    - Supernode listens on P2P headers and redirects
    - +: Transparent to sender  – : may touch local nodes
  - Directed brocade
    - Sender sends message directly to supernode
    - Sender locates supernode via DNS resolution:
      - nslookup supernode.cs.berkeley.edu
    +: maximum performance  – : state maintenance

Inter-supernode Routing
- Route: Supernode (sender) → Supernode (receiver)
  - Locate receiver’s supernode given destination nodeID
  - Use Tapestry object location
  - Routing mesh w/ built in proximity metrics
  - Location exploits locality (finds closer objects faster)
  - Finding supernodes
    - Supernode “publishes” cover set on brocade layer as locally stored objects
    - To route to node N, locate server on brocade storing N

Feasibility Analysis
- Some numbers
  - Internet: ~ 220M hosts, 20K AS’s, ~10K nodes/AS
  - Java implementation of Tapestry on PIII 800: ~1000 msgs/second
  - State maintenance
    - AS of 10K nodes, assume 10% enter/leave every minute
    - Only ~1.75% of CPU spent processing publish on Brocade
  - If inter-supernode traffic takes X ms, Publishing takes 5 X
  - Bandwidth: 1Kmsg * 1Kmsg/min = 1MB/min = 160kb/s
  - Storage requirement of Tapestry
    - 20K AS’s, Octal Tapestry, \( \log_2(20K) \) = 10 digits
    - 10K objects (Tapestry GUIDs) published per supernode
    - Tapestry GUID = 160 bits = 20B
    - Expected storage per SN: 10 * 10K * 20B = 2MB

Evaluation: Routing RDP
- Local proximity cache on, inter-domain intra-domain = 3:1
- Packet simulator, GT-ITM, 4096 T, 16 SN, CPU overhead = 1
**Evaluation: Bandwidth Usage**

- Brocade Aggregate Bandwidth Usage

**Brocade Summary**

- **P2P systems assume uniformity**
  - Extraneous hops through backbone to domains
  - Routing across congested stubs links

- **Constrain inter-domain routing**
  - Remove unnecessary routing through stubs
  - Reduce expected inter-domain hops
  - Limit misdirection in less congested backbone

- **Result:** lower latency, less bandwidth utilization

**Ongoing Questions**

- Performance at what cost?
  - Keep virtualization and level of indirection, named routing
  - May lose some fault-tolerance (how much?)

- Making P2P real
  - Deployment issues?
  - Impact of BGP routing policies on performance?

- Future/ongoing work
  - Fault-tolerant supernodes
  - Finer-grain node differentiation?
  - Brocade as replacement for BGP?

[Email/Website]