Incorporating P2P Networks in Service Provider Infrastructure

Alberto Leon-Garcia & Shad Sharma
University of Toronto

Why P2P for Service Providers?

- “Virtual distributed servers”
- Autonomous execution of applications on commodity resources
- P2P Innovations & Benefits
  - KaZaA, BitTorrent, Skype
  - Self-organizing, self-managing
  - Reliability
  - Scalability and Performance
  - Cost savings
- P2P Broad Applicability
  - Not limited to rogue operators
- Carrier Class Challenges
  - Reliability, Performance, Security
Introduction

- Overlay Topology
  - Application layer routing
  - Nodes maintain logical neighbours to whom they forward messages

- P2P Applications
  - Content Delivery
  - Lookups and Search
  - Service Virtualization
    - E.g. P2P HTTP server

Distributed Hashing

- Hash table
  - Defines set of buckets that hold objects

- Hash function
  - Distributes objects into buckets
  - Objects distributed “uniformly” among buckets

- Distributed Hash Table
  - Nodes are the buckets that store objects
  - Objects: files/resources/things you want to find/store

- Structured overlays well suited to providing DHT services
  - Predefined positions assigned to peers
  - Peers assigned hash values (buckets)
Introduction

Unstructured Overlays
+ Robust, reliable, fast insertion and removal
  - Broadcast based search
    \( O(m \sqrt[3]{n}) \) search time
    \( O(m \times n) \) search messages

Structured Overlay
+ Fast & efficient DHT search
  \( O(\log_B(n)) \) search time
  \( O(\log_B(n)) \) search messages
Routing table maintenance required
  - Not robust under churn

Hybrid Overlay
+ Fast & efficient DHT search
+ Robust, reliable, fast insertion and removal
+ Resilient to churn

Structured DHT capable overlays
Rigid finger tables

Chord
Structured DHT capable overlays
Rigid finger tables

Kademlia
Loosely consistent DHT overlay
Relaxed finger tables

Newscast
Epidemic protocol based on gossiping

Montressor
Dual layer approach: Newscast substrate

TrebleCast
Peers inserted in order in spiral-like fashion

Spiral - Notion of layers:
  - Provides data redundancy
  - Data stored at each layer

Peers maintain 4 neighbours:
  - In, out, left, right

Successor:
  - Peer responsible for replacing a failed peer
  - Successor moves “inwards” (closer to core)

Layer indicative of peer reliability:
  - Peers closer to core considered more reliable
TrebleCast (2)

- Dual layer approach:
  - Newscast substrate
  - Grid superstructure

- Adaptable to churn:
  - Superstructure repaired through gossip messages exchanged at Newscast substrate

- Fast adaptive search:
  - Search messages exchanged at superstructure layer
  - Lookups under static conditions: $O(\log(n))$
  - Graceful search degradation under increasing churn

- Flexible data storage policy:
  - Choose location of stored data (at core for instance)
  - Permits flexibility allowing data redundancy and load balancing

- Robustness and reliability:
  - Build overlay around core of reliable server-like peers

Implementation

- TrebleCast implemented in Java

- Currently used for SIP virtualization
  - May implement any <key, value> pair storage based mechanism
  - Register, store, retrieve, delete: $O(\log(n))$ time

- TrebleCast simulator implemented in Java

- P2P Monitor implemented in Java
  - Monitors peers in a P2P network
  - Allows basic interaction with peers through virtual console
**Pareto Turnover**

- Reliable peers move to overlay core
- Core “protected” from churn
- Improved search time (less routing table maintenance)

**Fast Adaptive Search**

![Graph showing search time for overlay network over time](image)
Static Search Comparison

Chord Churn Search Comp.

- Aggressive repair mechanism implemented to maintain Chord structure
- Search degrades exponentially as Churn rate increases past 10 peers/sec
TrebleCast Churn Search Comp.

- TrebleCast search degrades under exponential lifetime distribution
- Search remains almost constant under Pareto lifetime distribution
- Note: Storage policy chosen so that a core set of reliable peers are responsible for storage

Conclusions

- Treblecast for service provider setting
- Resilient to churn
- Fast adaptive search: $O(\log(n))$
- Inherent support for data redundancy
- Flexible data storage & retrieval policy