A Framework for Cost-Effective Peer-to-Peer Content Distribution

Mohamed Hefeeda
Ph.D. Candidate

Advisor: Bharat Bhargava

Department of Computer Sciences
Purdue University
Motivation

- Lots of underutilized end systems (peers) connected to the Internet
- Success of Peer-to-Peer (P2P) paradigm
  - Kazaa, Gnutella, SETI@HOME, …
- Fairly high cost for distributing digital contents (large videos)
- Why not share? Everybody benefits!
Motivation (cont’d)

▪ Our contribution …
  - Collaborative P2P Framework for Content Distribution
    • Peer contributes little, but we have too many of them!

▪ Two settings for the framework
  - Infrastructure
    • Content provider employs peers’ resources to disseminate contents
  - Cooperative resource sharing
    • Peers cooperate/coordinate to serve requests
Motivation (cont’d)

- **What we gain …**
  - Cost-effectiveness (for supplier & client)
  - Ease of deployment (on end systems)
  - Availability (large degree of redundancy)
  - Scalability (more peers ➔ more resources)
  - ..... 

- **What we need to do …**
  - Address research challenges
Motivation (cont’d)

- **Research problems**
  - Select and match multiple supplying peers with a requesting peer
  - Aggregate and coordinate contributions from peers
  - Adapt to peer failures and network conditions
  - Disseminate contents into the system
  - Consider peer *rationality* (self-interest) into protocols
  - Assess and incorporate peer *trustworthiness*
Outline

- Cooperative environment  [ACM MM’03]
  - CollectCast
  - PROMISE
  - Evaluation: simulation and implementation (PlanetLab)

- Infrastructure [J. Computer Networks, To appear 03]
  - Hybrid (super peer) architecture
  - Peer clustering and organization
  - Searching and dispersion algorithms
  - Evaluation: simulation

- Current/Future work
  - Rationality (Economics) [Tech Reports 03]
  - Trustworthiness (Security)
CollectCast

- CollectCast is a new P2P service
  - Middleware layer between a P2P lookup substrate and applications

- Previous work either
  - Assume one sender, e.g., [Tran, et al. 03, Bawa, et al. 02]
    - Ignores peer limited capacity
  - Or, multiple senders but no careful selection, e.g., [Padmanabhan, et al. 02]
    - Ignores peer diversity and network conditions
CollectCast (cont’d)

- Functions
  - Infer and label topology
  - Select best sending peers for each session
  - Coordinate contributions from peers
  - Adapt to peer failures and network conditions
CollectCast: Peer Selection

- **Considers**
  - $R_p$, $A_p(t)$
  - e2e available bandwidth and loss rate
  - Shared path segments

- **Problem formulation:**
  - Find $P^{actv}$ that

\[
\text{Maximizes } E \left[ \sum_{p \in P^{actv}} G_p R_p \right]
\]

Subject to $R_l \leq \sum_{p \in P^{actv}} R_p \leq R_u$
PROMISE and Experiments on PlanetLab

- PROMISE is a P2P media streaming system built on top of CollectCast
- Tested on PlanetLab test bed
- Extended Pastry to support multiple peer look up
- Used several MPGE-4 movie traces
- Select peers using topology-aware (the one used in CollectCast) and end-to-end
CollectCast: Performance

Packet-level performance
- Smoother aggregated rate achieved by CollectCast

Frame-level performance
- Much fewer frames miss their deadlines with CollectCast
- CollectCast requires smaller initial buffering time to ensure all frames meet their deadlines
Outline

- **Cooperative environment** [ACM MM’03]
  - CollectCast
  - PROMISE
  - Evaluation: simulation and implementation (PlanetLab)

- **Infrastructure** [J. Computer Networks, To appear 03]
  - Hybrid (super peer) architecture
  - Peer clustering and organization
  - Searching and dispersion algorithms
  - Evaluation: simulation

- **Current/Future work**
  - Rationality (Economics) [Tech Reports 03]
  - Trustworthiness (Security)
P2P Infrastructure

- **Target environments**
  - Streaming (on-demand) videos to many clients
    - Distance learning, media center, corporate streaming, ...

- **Current approaches**
  - Unicast
    - Centralized
    - Proxy caches
    - CDN (third-party)
  - Multicast
    - Network layer
    - Application layer

- **Proposed approach (Peer-to-Peer)**
Unicast Streaming

- Easy to deploy, administer
- Limited scalability
- Reliability concerns
- Load on the backbone
- High deployment cost $$$....$

Centralized

CDN

- Good performance
- Suitable for web pages with moderate-size objects
- Co$t$: CDN charges for every megabyte served! [Raczkowski 02]
- For one-hour streamed to 1,000 clients, content provider pays $264 to CDN (at 0.5 cents/MByte)
Multicast Streaming

- Efficient!
- Asynchronous client: Patching, skyscraper, ... [Mahanti, et al. 03]
- Tune to multiple channels ➔ Require inbound bandwidth 2R+
- Scalability of routers
- Not widely deployed!!

Network layer

Patch stream

Application layer

- Deployable
- E.g., [Narada 02, NICE 02, Zigzag 03, ...]
- Assume end systems can support (outbound bandwidth) multiple folds of the streaming rate
P2P Approach: Key Ideas

- Make use of underutilized peers’ resources
- Make use of heterogeneity
- Multiple peers serve a requesting peer
- Network-aware peer organization

Super Peers play special roles ➔

Hybrid Architecture
Hybrid Architecture: Issues

- **Peer Organization**
  - Two-level peer clustering
  - Join, leave, failure, overhead, super peer selection

- **Cluster-Based Searching**
  - Find *nearby* suppliers

- **Cluster-Based Dispersion**
  - Efficiently disseminate new media files
Peer Organization

- **Previous client clustering techniques**
  - Too coarse [Barford, et al. 02]
    - Few large clusters
    - Good for cache placement
  - Too fine [Bestavros, et al. 01] [Krishnamurthy, et al. 00]
    - Many small clusters

- **Our approach**
  - Balanced, suitable for P2P environments
  - Use BGP tables [RouteViews, Univ. of Oregon]
  - Validated by Internet Statistics
Peer Organization: Two-Level Clustering

- **Two-level clustering**
  - **Network cluster**
    - Peers sharing same network prefix
  - **AS cluster**
    - All network clusters within an Autonomous System

### Snapshot of a BGP routing table

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Next Hop</th>
<th>AS Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.9.0.0/20</td>
<td>207.246.129.6</td>
<td>11608 2914 668 7170 1455</td>
</tr>
<tr>
<td>18.0.0.0/8</td>
<td>207.246.129.6</td>
<td>7018 1 3</td>
</tr>
<tr>
<td>128.2.0.0/16</td>
<td>207.246.129.6</td>
<td>11608 2914 5050 9</td>
</tr>
<tr>
<td>128.10.0.0/16</td>
<td>207.246.129.6</td>
<td>11608 2914 3356 1 19782 17</td>
</tr>
</tbody>
</table>
Dispersion (cont'd)

- **Dispersion Algorithm (basic idea)**
  - /* Upon getting a request from \( P_y \) to cache \( N_y \) segments */
  - \( C \leftarrow \text{getCluster} \left( P_y \right) \)
  - Compute available \( (A) \) and required \( (D) \) capacities in cluster \( C \)
  - If \( A < D \)
    - \( P_y \) caches \( N_y \) segments in a *cluster-wide round robin* fashion (CWRR)

- All values are *smoothed* averages
- Average available capacity in \( C \):
  \[
  A_C = \frac{1}{T} \sum_{P_x \in C} \frac{R_x}{R} \frac{N_x}{N} u_x
  \]
- CWRR Example: (10-segment file)
  - \( P_1 \) caches 4 segments: 1,2,3,4
  - \( P_2 \) then caches 7 segments: 5,6,7,8,9,10,1
Performance Under Flash Crowd Arrivals

Client Arrival Pattern

- Flash crowd ≡ *sudden* increase in client arrivals
System Under Flash Crowd (cont'd)

System capacity

Load on seeding peer

---

The role of the seeding peer is just *seeding*

- During the peak, we have 400 concurrent clients (26.7 times original capacity) and *none* of them is served by the seeding server (50% caching)
Dispersion Algorithm

- **Avg. number of hops**
  - 8.05 hops (random), 6.82 hops (ours) → 15.3% savings

- **For a domain with a 6-hop diameter**
  - Random: 23% of the traffic was kept inside the domain
  - Cluster-based: 44% of the traffic was kept inside the domain
Outline

- **Cooperative environment** [ACM MM’03]
  - CollectCast
  - PROMISE
  - Evaluation: simulation and implementation (PlanetLab)

- **Infrastructure** [J. Computer Networks, To appear 03]
  - Hybrid (super peer) architecture
  - Peer clustering and organization
  - Searching and dispersion algorithms
  - Evaluation: simulation

- **Current/Future work**
  - Rationality (Economics) [Tech Reports 03]
  - Trustworthiness (Security)
Rationality in P2P Systems

- **Rationality**: self-interest ≡ maximize one’s own utility

- **Rationality is a growing concern in P2P systems**
  
  [Shneidman & Parkes 03] [Adar & Huberman 00] [Golle, et al. 01]
  
  - Rational nodes consume more than they contribute → Jeopardizes growth of P2P systems

- **Infrastructure**
  
  - Provider motivates peers to contribute resources
  
  - Revenue sharing mechanism [Hefeeda, et al., Tech Report 03]

- **Cooperative**
  
  - Enforce fair contribution/consumption of resources
  
  - Develop distributed incentive mechanisms [in progress]
Dealing with Rationality

- **Ideas from economic theory ported to computer science**
- **Mechanism Design (MD)** [e.g., Fudenburg & Triole 91]
  - Inverse game theory: how to design a game to yield the desired outcomes in equilibrium
  - Define strategies, “rules of the game”, for selfish agents
- **Algorithmic Mechanism Design (AMD)** [Nissan & Ronen 99]
  - MD + computational complexity considerations
- **Distributed Algorithm Mechanism Design (DAMD)** [Feigenbaum & Shenker 02] [Feigenbaum et al. 02]
  - AMD + Distributed
Cooperative: Incentive Mechanism

- Peers are agents with costs (storage and BW)
- Want to replicate (provision) objects into the network to optimize a system-wide function (e.g. minimize the Expected Search Size (ESS))

Problem
- Design an incentive mechanism that yields optimal replication strategy, given that nodes are rational

Related Work
- Optimal replication with obedient nodes [Cohen & Shenker 02]
  - Replicating objects in proportion to the square root of their rates results in minimum ESS
Incentives??

- **What may serve as Incentives in P2P??**
  - **Pricing (monetary)**
    - E.g., micro payments
  - **Non-pricing (non-monetary)**
    - Higher priority
    - Peers ranking
    - Peers membership level
    - .....  
  - **Success story**
    - SETI@HOME: “User of the Day”, listing of users
Trustworthy P2P Systems

- As always, Security is a big issue!
- More peers join if they can trust other peers
- How to assess trust? And, how to use it?
- Research Problems
  - Evidence identification
    - What may serve as an evidence?
    - E.g., fraction of time a peer fulfilled its commitment
  - Evidence collection
    - How to collect sufficient instances of this evidence?
    - Complexity: communications, processing
  - Trust models
    - Using evidences, build web of trust among peers
  - Trust-based searching
    - Find me a peer that, with high probability, will fulfill its duties!!!
Conclusions

- A P2P framework for content distribution
- CollectCast and PROMISE
  - Cooperative media streaming environment
- Hybrid Architecture
  - P2P streaming with super peer assisting in searching and dispersion
  - Cost-effective infrastructure for content distribution managed by a provider
- Rationality (Current)
  - Incentive-compatible network provisioning
- Trustworthiness (Future)
  - Trust-based searching schemes
Thank You!

Questions, Suggestions, Comments are appreciated!

- More information and papers at …
  http://www.cs.purdue.edu/homes/mhefeeda
P2P Systems: Basic Definitions

- **Peers** cooperate to achieve desired functions
  - *Cooperate*: share resources (CPU, storage, bandwidth), participate in the protocols (routing, replication, ...)
  - *Functions*: file-sharing, distributed computing, communications, ...

- **Examples**
  - Gnutella, Napster, Freenet, OceanStore, CFS, CoopNet, SpreadIt, SETI@HOME, ...

- **Well, aren’t they just distributed systems?**
  - P2P == distributed systems?
P2P vs. Distributed Systems

- P2P = distributed systems++;
  - Ad-hoc nature
  - Peers are not servers [Saroui, et al. 02]
    - Limited capacity and reliability
  - Much more dynamism
  - Scalability is a more serious issue (millions of nodes)
  - Peers are self-interested (selfish!) entities
    - 70% of Gnutella users share nothing [Adar & Huberman 00]
  - All kind of Security concerns
    - Privacy, anonymity, malicious peers, ... you name it!
P2P Systems: Rough Classification

[Lv et al., ICS’02], [Yang et al., ICDCS’02]

- **Structured (or tightly controlled, DHT)**
  - Files are *rigidly* assigned to specific nodes
  - Efficient search & guarantee of finding
    - Lack of partial name and keyword queries
  - **Ex.:** Chord [Stoica, et al. 01], CAN [Ratnasamy, et al. 01], Pastry [Rowstron & Druschel 01]

- **Unstructured (or loosely controlled)**
  - Files can be anywhere
  - Support of partial name and keyword queries
    - Inefficient search (some heuristics exist) & no guarantee of finding
  - **Ex.:** Gnutella

- **Hybrid (P2P + centralized), super peer notion)**
  - Napster, KazaA
File-sharing vs. Streaming

- **File-sharing**
  - Download the *entire* file first, then use it
  - Small files (few Mbytes) ➞ short download time
  - A file is stored by one peer ➞ one connection
  - No timing constraints

- **Streaming**
  - Consume (playback) as you download
  - Large files (few Gbytes) ➞ long download time
  - A file is stored by multiple peers ➞ several connections
  - Timing is crucial
Streaming Approaches

- **Distributed caches** [e.g., Chen and Tobagi, ToN’01 ]
  - Deploy caches all over the place
  - Yes, increases the scalability
    - Shifts the bottleneck from the server to caches!
  - But, it also multiplies cost
  - What to cache? And where to put caches?

- **Multicast**
  - Mainly for live media broadcast
  - Application level [Narada, NICE, Scattercast, Zigzag, … ]
  - IP level [e.g., Dutta and Schulzriner, ICC’01]
    - Widely deployed?
Streaming Approaches (cont'd)

- **P2P approaches**
  - **SpreadIt** [Deshpande, *et al.*, Stanford TR 01]
    - **Live media**
      - Build application-level multicast distribution tree over peers
  - **CoopNet** [Padmanabhan *et al.* 02]
    - **Live media**
      - Builds application-level multicast distribution tree over peers
    - **On-demand**
      - Server redirects clients to other peers
      - Assumes a peer can (or is willing to) support the full rate
      - CoopNet does not address the issue of quickly disseminating the media file