School of Computing Science
Simon Fraser University, Canada

Modeling and Caching of P2P Traffic

Mohamed Hefeeda

Osama Saleh

ICNP'06

15 November 2006

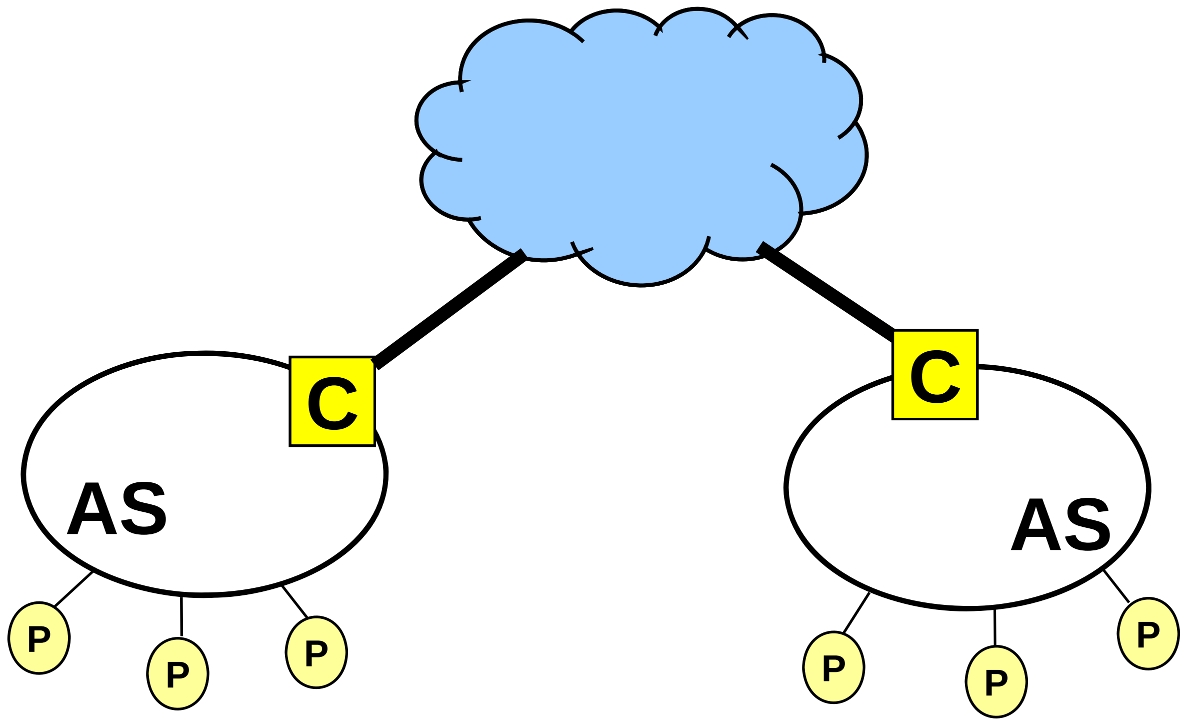
Motivations

- **P2P traffic is a major fraction of Internet traffic**
- **… and it is increasing [Karagiannis 04]**

- **Negative consequences**
  - **increased load on networks ➔**
  - **higher cost on ISPs (and users!), and**
  - **more congestion**
- **Can traffic *caching* help ?**

Our Problem

- Design an effective caching scheme for P2P traffic
- Main objective:
  - Reduce WAN traffic → reduce cost & congestion



Our Solution Approach

- **Measure and model P2P traffic characteristics relevant to caching, i.e.,**
  - **seen by cache deployed in an autonomous systems (AS)**

- **Then, develop a caching algorithm**

Why not use Web/Video Caching Algorithms?

- **Different traffic characteristics:**
  - **P2P vs. Web:** P2P objects are large, immutable and have different popularity models
  - **P2P vs. Video:** P2P objects do not impose any timing constraints

- **Different caching objectives:**
  - **Web:** minimize latency, make users happy
  - **Video:** minimize start-up delay, latency and enhance quality
  - **P2P:** minimize bandwidth consumption

Related Work

- **Several P2P measurement studies, e.g.,**
  - **[Gummadi 03]: Object popularity is not Zipf, but no closed-form model is given, conducted in one network domain**
  - **[Klemm 04]: Query popularity follows mixture of two Zipf distributions, we use popularity of actual object transfers**
  - **[Leibowitz 02] [Karagiannis 04]: highlight potential of caching P2P traffic, no caching algorithms presented**
  - **All provide useful insights, but they were not explicitly designed to study caching P2P traffic**
- **P2P caching algorithms**
  - **[Wierzbicki 04]: proposed two P2P caching algorithms, we compare against the best of them (LSB)**
  - **We also compare against LRU, LFU, and GDS**

Measurement Study

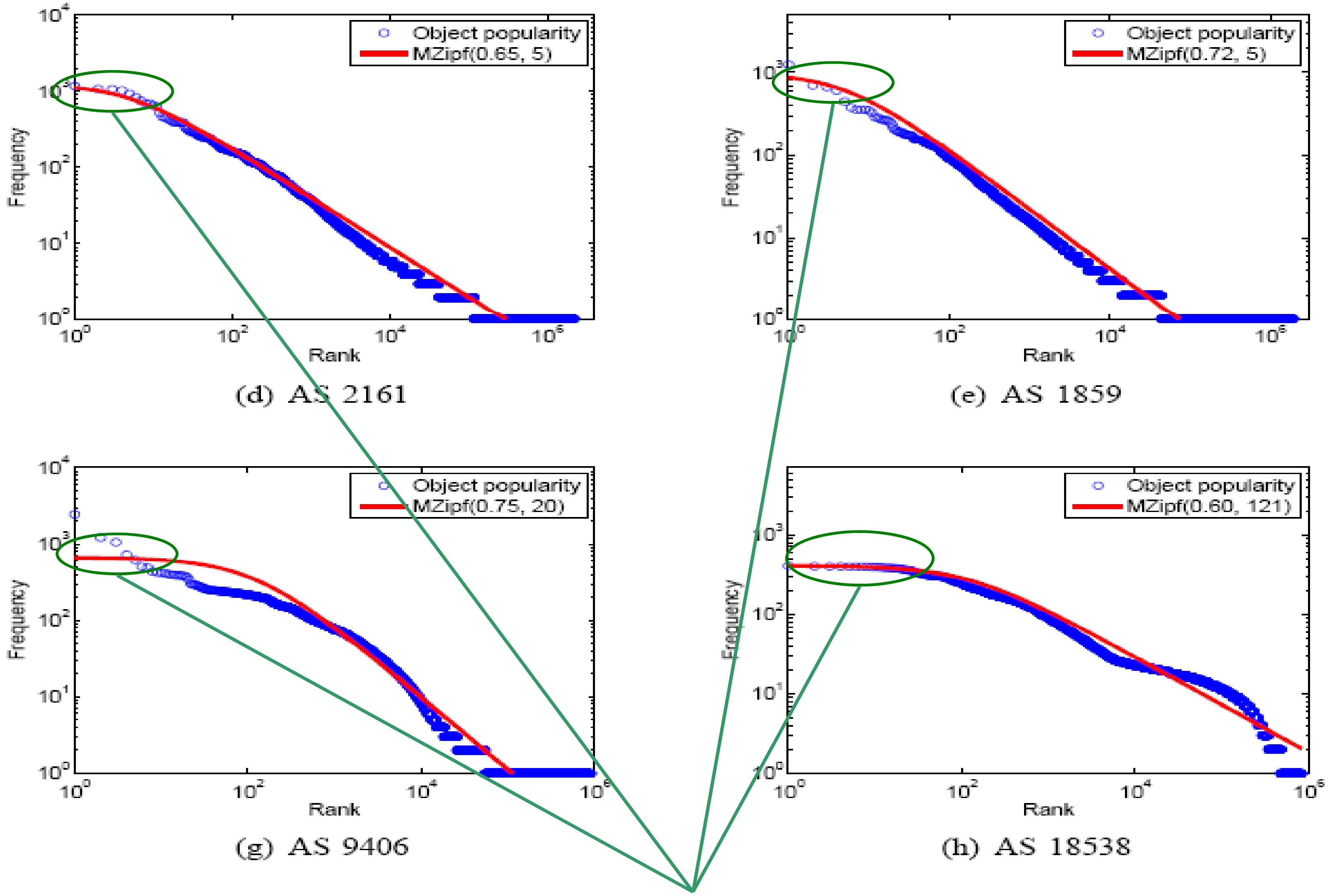
- **Modified Limewire (Gnutella) to:**
  - **Run in super peer mode**
  - **Maintain up to 500 concurrent connections (70% with other super nodes**
  - **Log all *query* and *queryhit* messages**
- **Measure and model**
  - **Object popularity**
  - **Popularity dynamics**
  - **Object sizes**
- **Why Gnutella?**
  - **Supports passive measurements**
  - **Open source: easy to modify**
  - **One of the top-three most popular protocols [Zhao 06]**

Measurement Study: Stats

Measurement Period	Jan 06 – Sep 06
Unique Objects	17 M
Unique IPs	39 M
ASes with more than 100,000 downloads	127
Total traffic volume	6,262 Tera Bytes

- Is it representative for P2P traffic? We believe so.
  - Traffic characteristics are similar in different P2P systems
    - [Gummadi 03]: Non-Zipf traffic in Kazza, same as ours
    - [Saroiu 03]: Napster and Gnutella have similar session duration, host up time, #files shared
    - [Pouwelse 04]: Similar host up time and object properties in BitTorrent

Measurement Study: Object Popularity



(d) AS 2161 (e) AS 1859 (g) AS 9406 (h) AS 18538

Notice the flattened head, unlike Zipf

Modeling Object Popularity

- **We propose a Mandelbrot-Zipf (MZipf) model for P2P object popularity:**

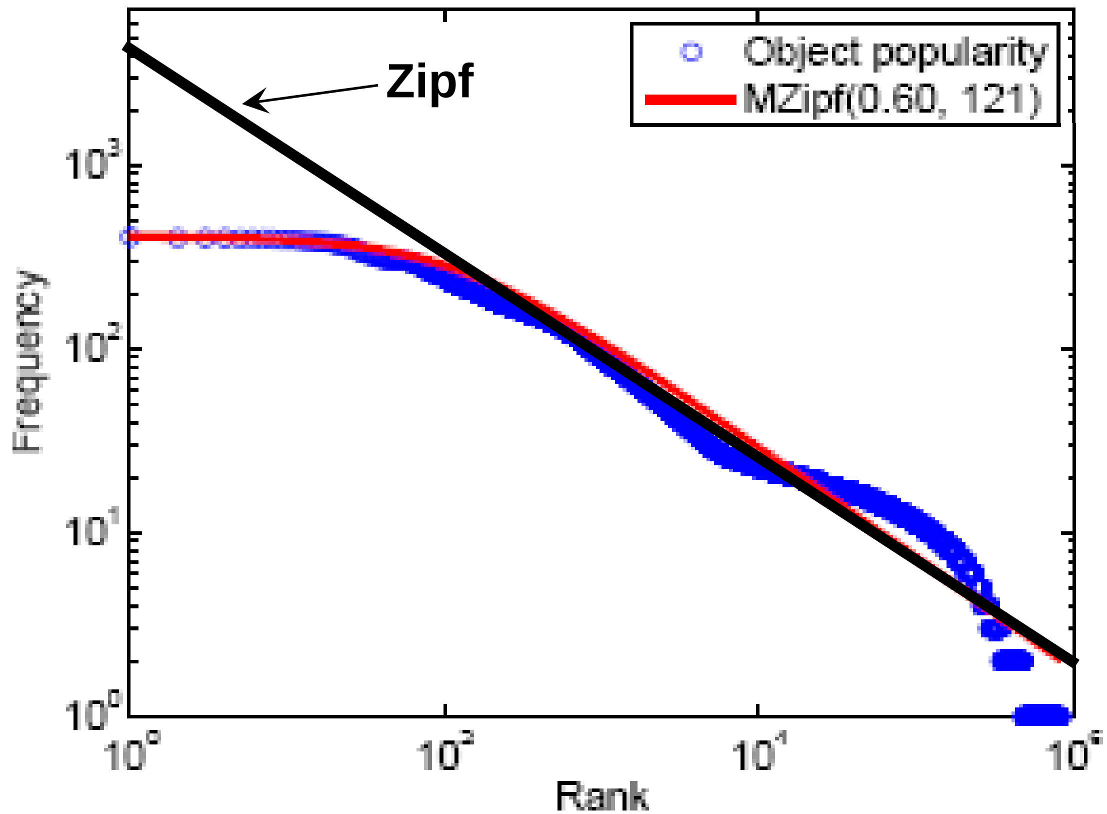
$$p(i) = \frac{K}{(i+q)^{\alpha}}$$

  - **α: skewness factor, same as Zipf-like distributions**
  - ***q:* plateau factor, controls the plateau shape (flattened head) near the lowest ranked objects**
  - **Larger q values → more flattened head**
- **Validation across top 20 ASes (in terms of traffic)**
  - **Sample in previous slide**

Zipf vs. Mandelbrot-Zipf

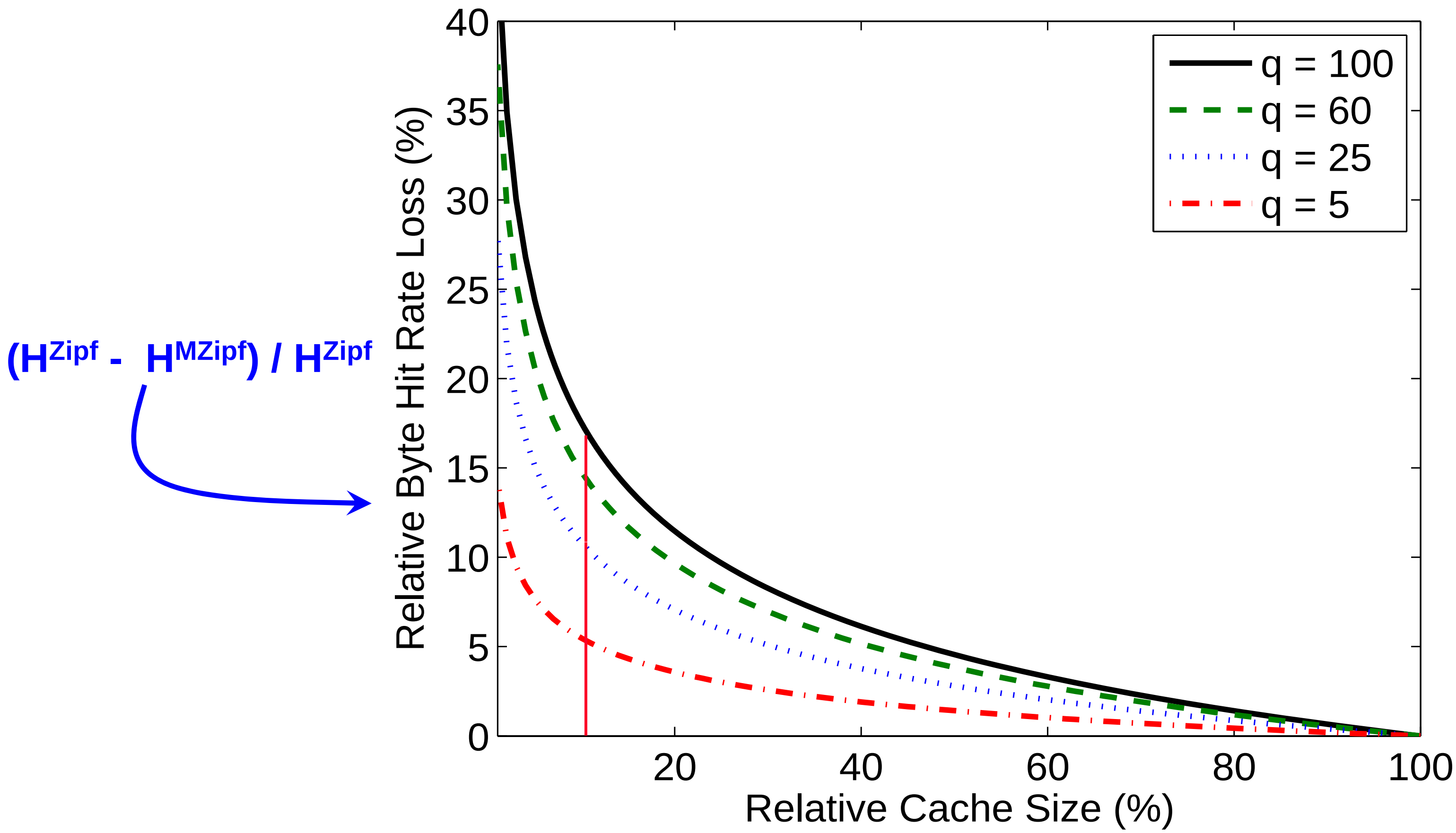
$$\text{Zipf}\ : p(i) = \frac{K}{(i)^{\alpha}} \qquad \text{MZipf}\ : p(i) = \frac{K}{(i+q)^{\alpha}}$$

- **Zipf over-estimates popularity of objects at lowest ranks**
- **Which are the good candidates for caching**



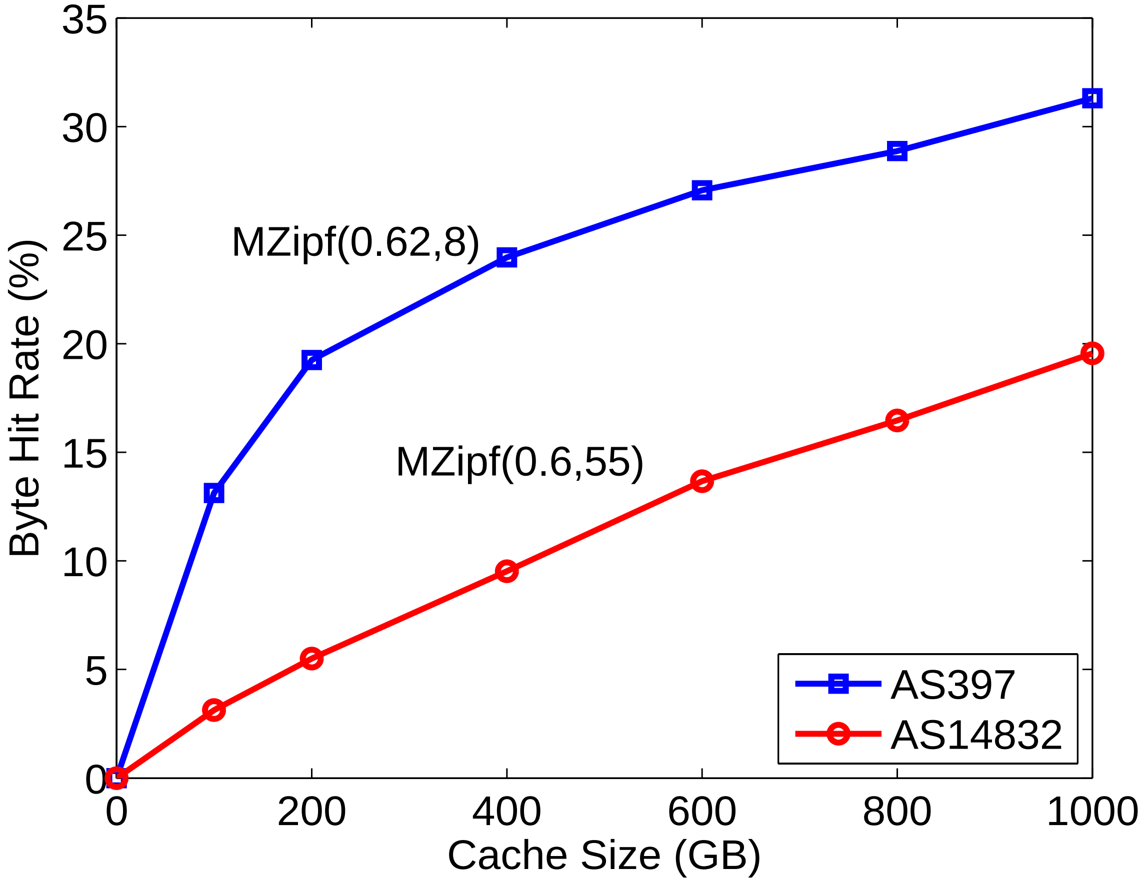
(h) AS 18538

Effect of MZipf on Caching



- **Simple analysis using LFU policy**
- **Significant bye hit rate loss at realistic cache sizes (e.g., 10%)**

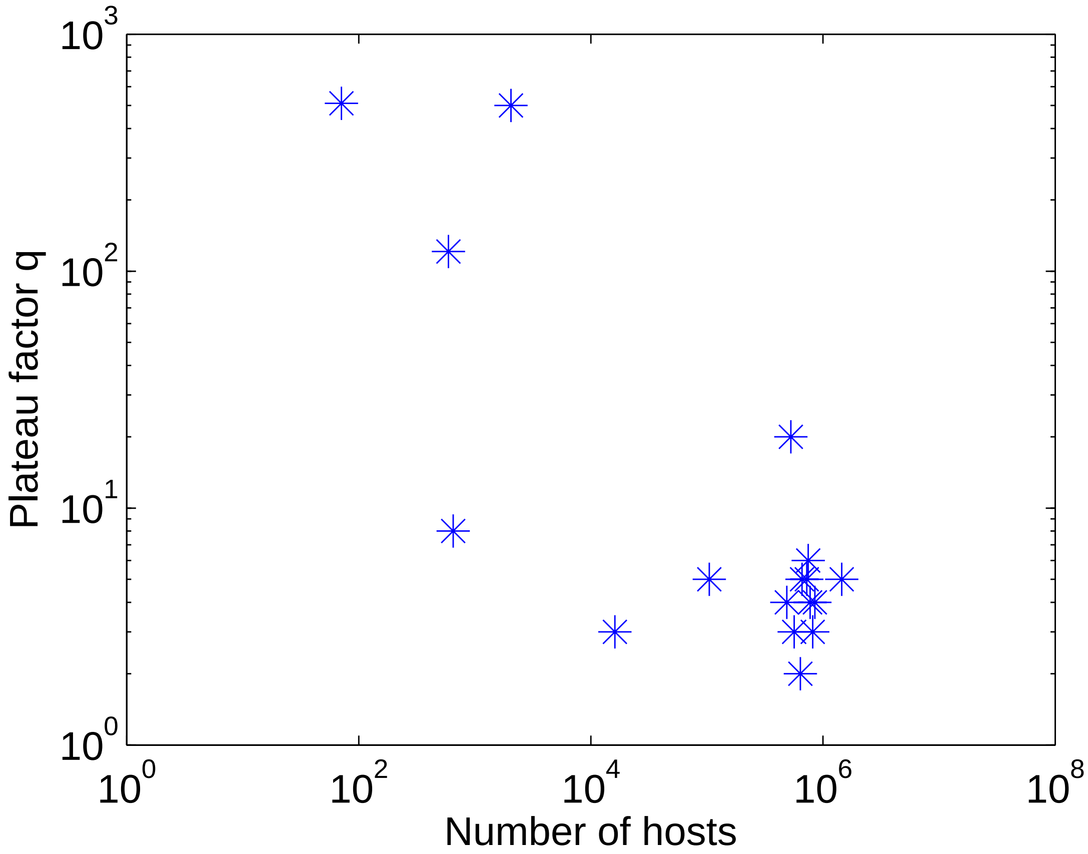
Effect of MZipf on Caching (cont'd)



- **Trace-based simulation using Optimal policy in two ASes**
- **larger q (more flattened head) ➔ smaller byte hit rate**

When is q large?

- **In ASes with small number of hosts**
  - **Immutable objects ➔ download at most once behavior ➔**
  - **Object popularity bounded by number of hosts ➔ large *q***



q values for 20 ASes

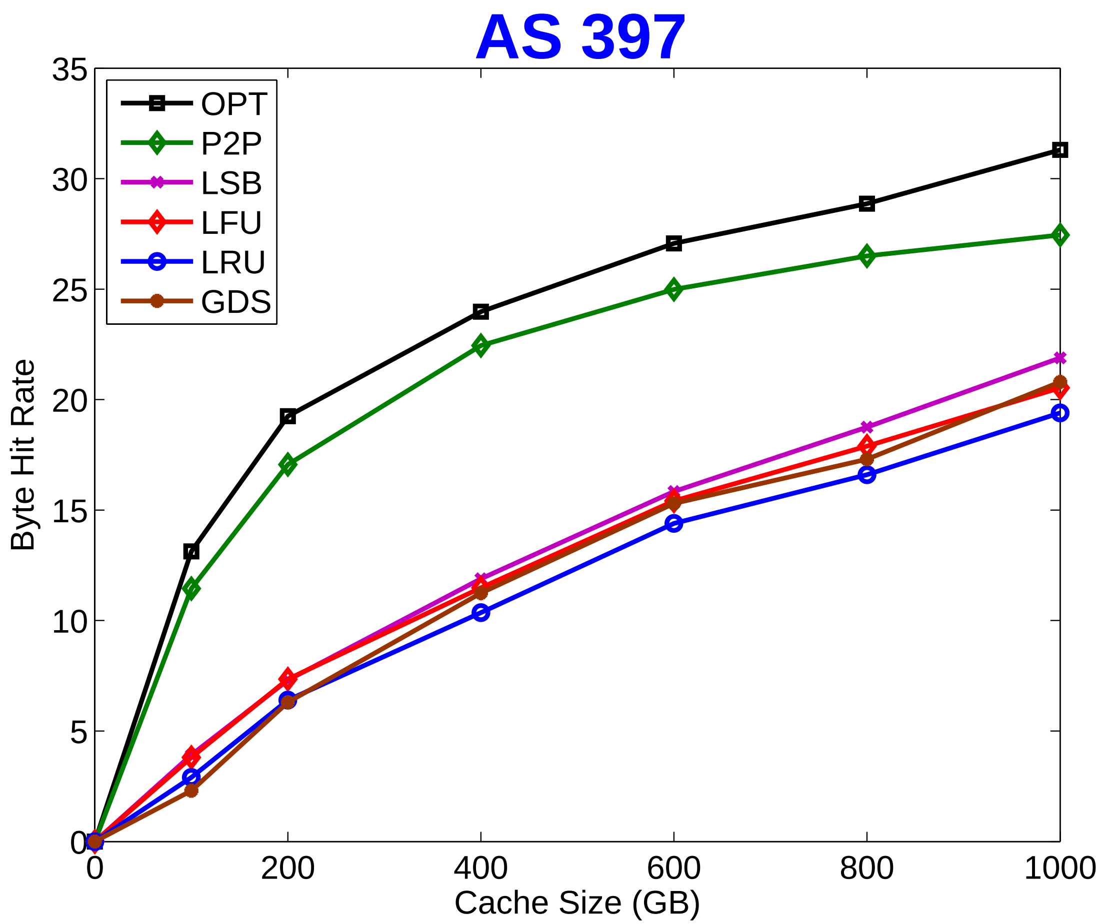
P2P Caching Algorithm: Basic Idea

- Proportional Partial Caching
  - Cache fraction of the object proportional to its popularity
  - Motivated by the Mandelbrot-Zipf popularity model
  - Minimizes the effect of caching large unpopular objects
- Segmentation
  - Divide objects into segments of different sizes
  - Motivated by the existence of multiple workloads
- Replacement
  - Replace segments of the object with the least number of served bytes normalized by its cached fraction

Trace-based Performance Evaluation

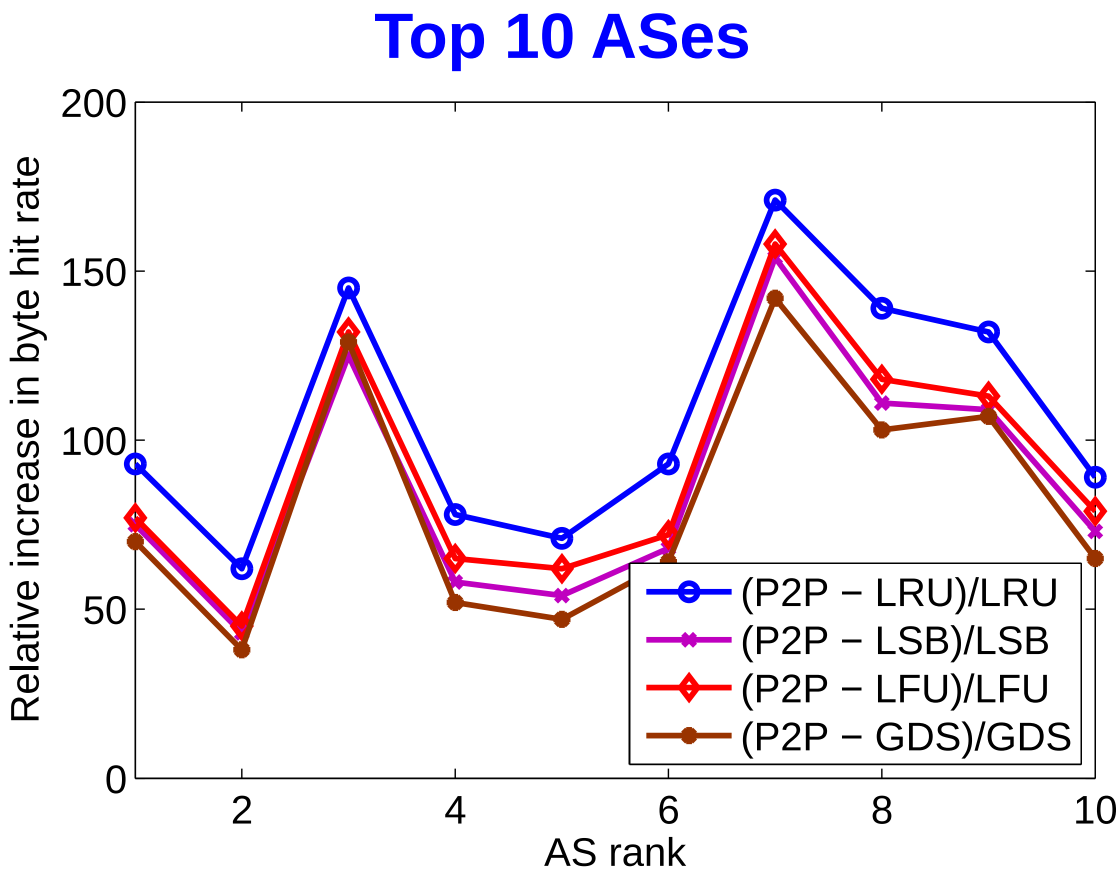
- **Algorithms Implemented**
  - Web policies: LRU, LFU, Greedy-Dual Size (GDS)
  - P2P policies: Least Sent Bytes (LSB) [Wierzbicki 04]
  - Offline Optimal Policy (OPT): looks at entire trace, caches objects that maximize byte hit rate
- **Scenarios**
  - With and without aborted downloads
  - Various degrees of temporal locality (popularity, temporal correlation)
- **Performance**
  - Byte Hit Rate (BHR) in top 10 ASes
  - Importance of partial caching
  - Sensitivity of our algorithm to: segment size, plateau and skewness factors

Byte Hit Rate: No Aborted Downloads



- **BHR of our algorithm is close to the optimal, much better than LRU, LFU, GDS, LSB**

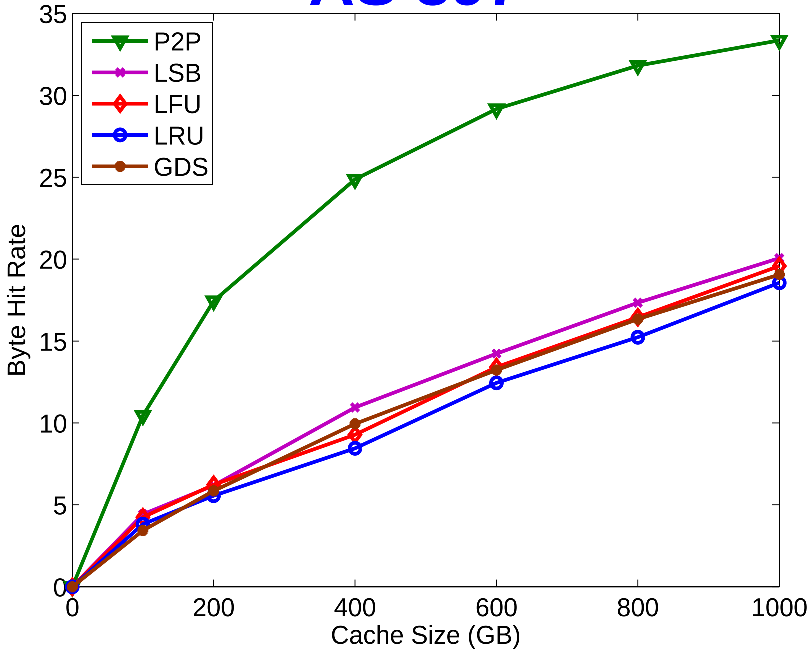
Byte Hit Rate: No Aborted Downloads (cont'd)



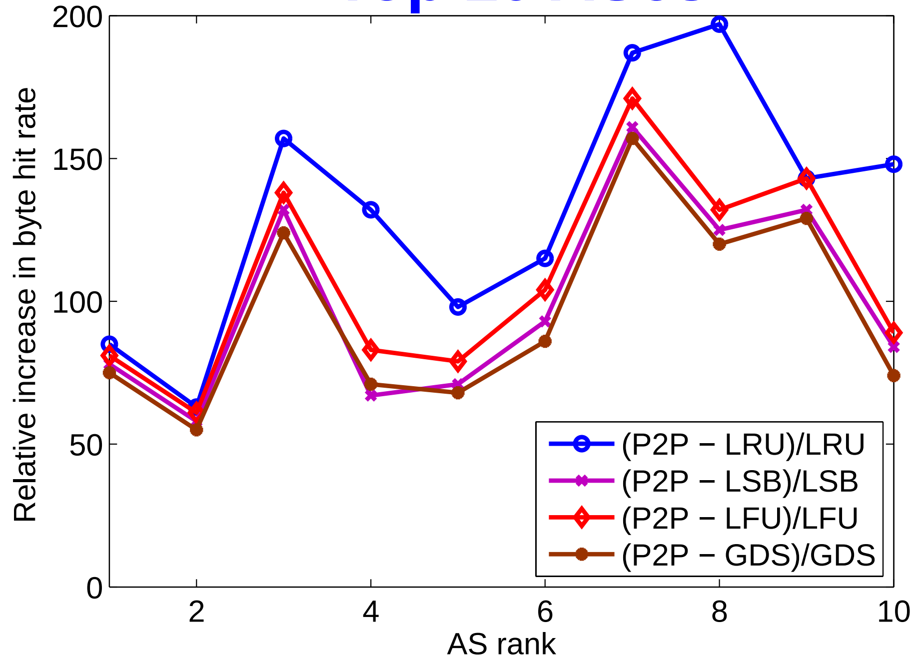
- **Our algorithm consistently outperforms all others in top 10 ASes**

Byte Hit Rate: Aborted Downloads

AS 397



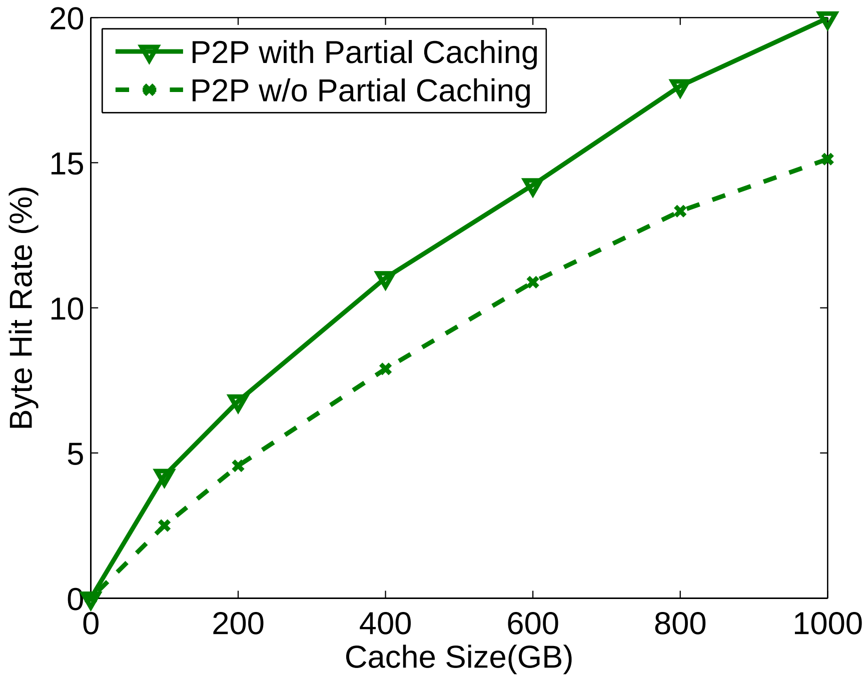
Top 10 ASes

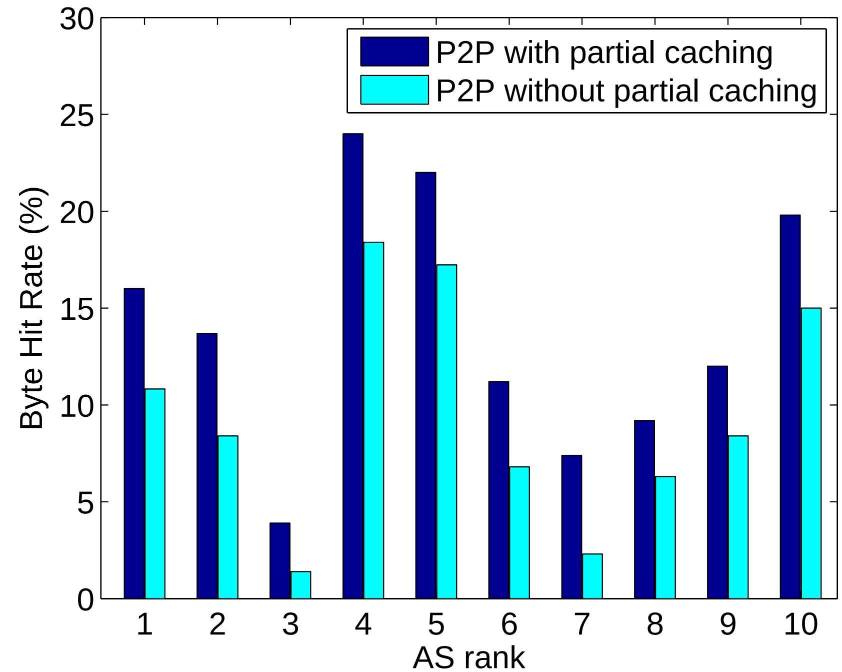


- **Same traces as before, adding 2 partial transactions for every complete transaction [Gummadi 03]**
- **Performance gap is even wider**
  - **BHR is at least 40% more, and**
  - **At most triple the BHR of other algorithms**

Importance of Partial Caching (1)

- **Compare our algorithm with and without partial caching**
  - **Keeping everything else fixed**

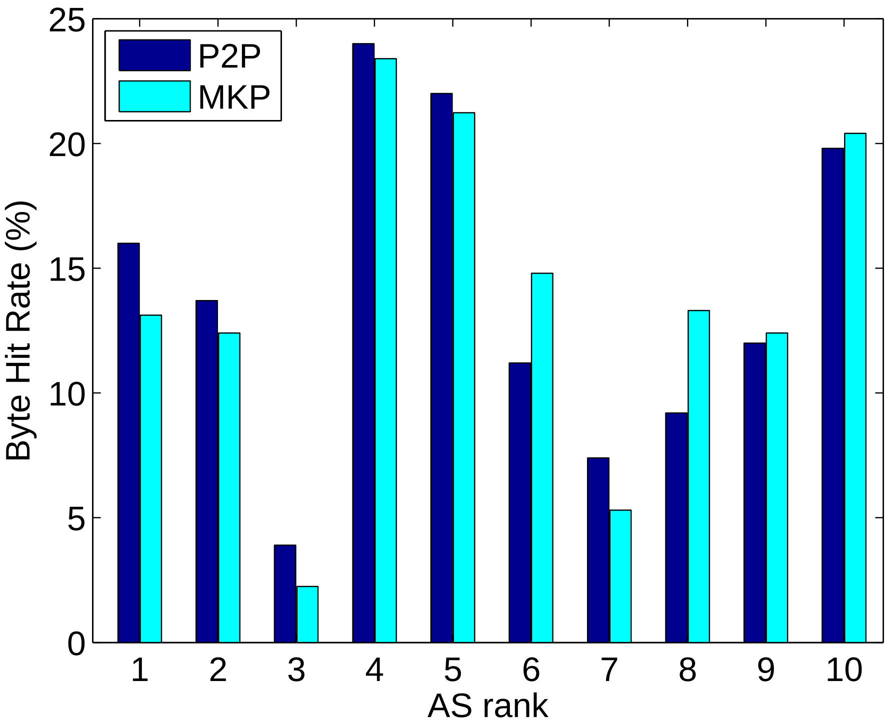




- **Performance of our algorithm degrades without partial caching in all top 10 ASes**

Importance of Partial Caching (2)

- Compare against an optimal policy that **does not** do partial caching
- MKP = store Most K Popular *full* objects that fill the cache
- Our policy outperforms MKP in 6 out of 10 top ASes, and close to it in the others



- MKP: *optimal*, no partial caching
- P2P: *heuristic* with partial caching

Importance of Partial Caching (3)

- **Now, given that our P2P *partial* caching algorithm**
  - **Outperforms LRU, LFU, GDS (all full caching)**
  - **Is close to the offline OPT (maximizes byte hit rate)**
  - **Outperforms the offline MKP (stores most K-popular objects)**
  - **Suffers when we remove partial caching**
- **It is reasonable to *believe* that**

Partial caching is critical in P2P systems,

because of large object sizes and MZipf popularity

Conclusions

- Conducted eight-month study to measure and model P2P traffic characteristics relevant caching
- Found that object popularity can be modeled by Mandelbrot-Zipf distribution (flattened head)
- Proposed a new proportional *partial* caching algorithm for P2P traffic
  - Outperforms other algorithms by wide margins,
  - Robust against different traffic patterns

Thank You!

Questions??

- Some of the presented results are available in the extended version of the paper
- All traces are available:

  http://www.cs.sfu.ca/~mhefeeda

Future Work

- Implement a P2P proxy cache prototype
- Extend measurement study to include other P2P protocols
- Analytically analyze our P2P caching algorithm
- Use cooperative caching between proxy caches at different ASes