Adaptive Probabilistic Search for Peer-to-Peer Networks

Dimitrios Tsoumakos, Nick Roussopoulos

Department of Computer Science,

University of Maryland, College Park

{dtsouma, nick}@cs.umd.edu

Third IEEE International Conference on Peer-to-Peer Computing

Linköping, Sweden, 1-3 Sept 2003

Presentation Outline

- Short introduction to P2P technology
- Object location in *unstructured* P2P networks
- The APS algorithm
- Simulation results
- Related work
- Conclusions

The notion of P2P

- *"Sharing of resources available at the edges of the Internet"*
- Resources could be content, storage, CPU-cycles, bandwidth, etc.
- Peers operate both as clients and servers
- P2P paradigm has many plausible characteristics:
  - Scalability
  - No centralized authority, robustness
  - Cooperation, sharing
  - Anonymity, etc

What can P2P be used for?

- According to a (conservative) estimate:
  - 10 billion MHz & 10,000 TB not utilized at the edges of the Internet [openP2P.com]
- The size of the networks and the complexity/ requirements from the protocols steadily increase
- On the other hand:
  - Bandwidth consumption attributed to popular file-sharing applications reaches 60% of the total Internet traffic [15]
- Must be able to locate the resources efficiently

The Problem of Object Location in P2P

- We focus on *unstructured* P2P networks
  - Network does not control replica placement
  - No guarantees for a search
- Each peer obtains a set of objects, makes requests for others (no caching)
- In such networks, peers arrive and depart in an ad-hoc manner

Object Location Schemes for P2P

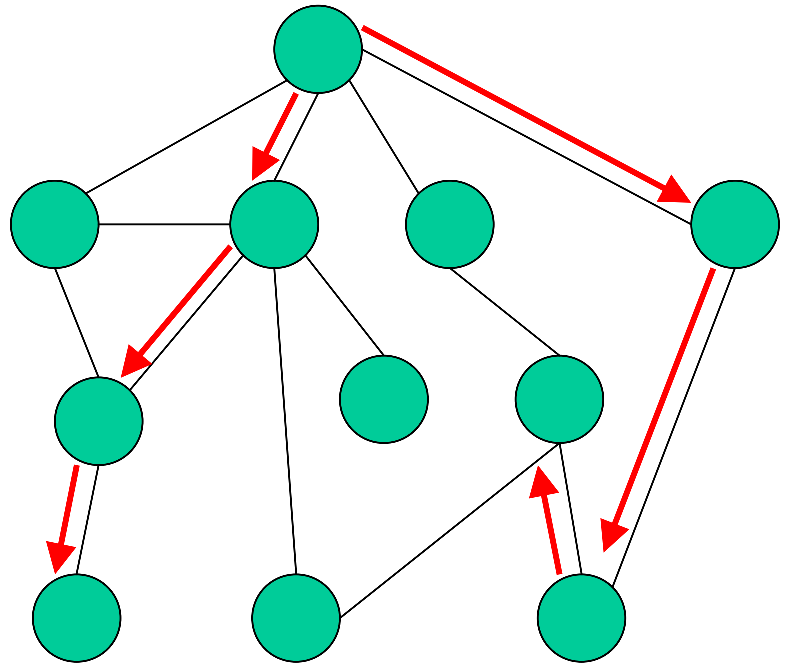
- Napster [11] utilized a central directory for the location of the music files

- Current search schemes present two basic problems:
  - Search in a blind manner $\Rightarrow$ use flooding (or its variations)
  - Utilize indices too expensive to maintain

The Random Walks Approach [9]

- Deployment of *k walkers* for object discovery
- Random forwarding

- Vast message reduction
- Local load-balancing
- Varying performance
- Cannot adapt to different workloads



Desired Characteristics

- Bandwidth-efficient
- Effective object discovery
- Adaptation to different workloads
- Robustness in dynamic environments/failures

The Adaptive Probabilistic Search scheme (APS)

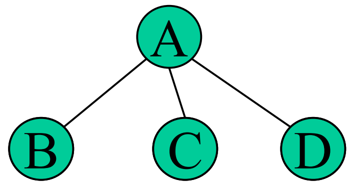
- Deploy *k* walkers
- Probabilistic forwarding using indices
- Peers keep indices regarding only their neighbors
- Indices are updated according to walker success/ failure
- Two index update policies

The APS scheme (1)

- Requesting peers deploy *k* walkers
- A walker can be:
  - Successful (finds a replica of the object)
  - Unsuccessful (travels TTL hops or cannot travel further or completes a circle)
- At each step, the search packet maintains the query path
- Peers maintain soft state – avoid duplicates

The APS scheme (2)

- Each peer maintains one index per neighbor per requested object
- Index values represent the probability of finding that object at (or through) each neighbor
- Example (indices at node A):
  - A chooses B with Pr = 0.3
  - A chooses C with Pr = 0.5
  - A chooses D with Pr = 0.2



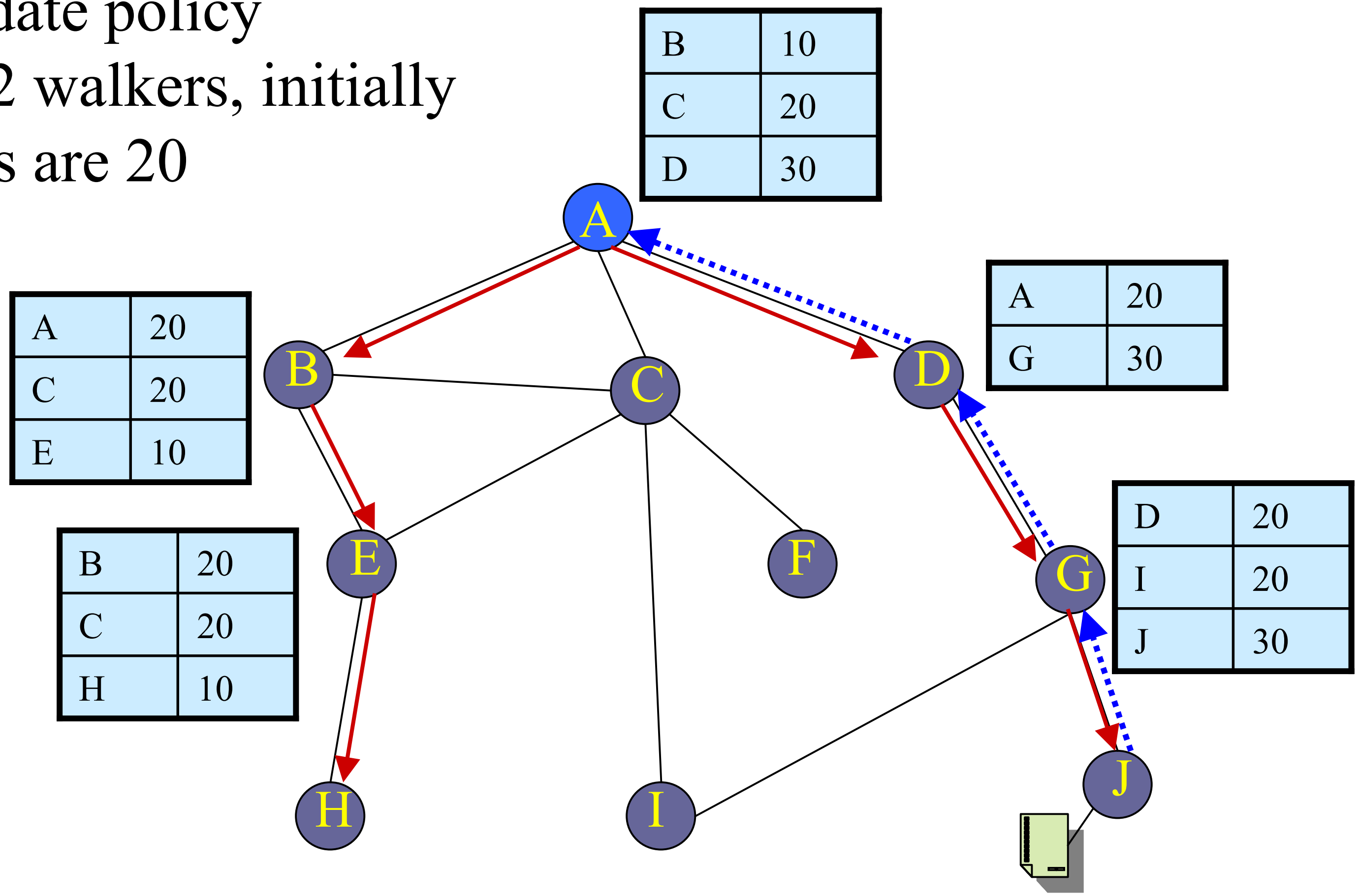
B	30
C	50
D	20

The APS scheme (3)

- During the search:
  - Peers increase the index value(s) of the next-hop(s) they choose (*optimistic* approach)
  - Or, they decrease them (*pessimistic* approach)
- If a walker is successful (unsuccessful) in the optimistic (pessimistic) case, there is nothing to be done
- Otherwise, correct indices along the *reverse* path
  - Increase/decrease by more than the initial amount

An example of APS

Node J holds the requested object
Pessimistic update policy
Nodes deploy 2 walkers, initially
all index values are 20
TTL = 3



Characteristics of APS

- No message exchange after node arrivals/ departures or object updates
- Utilize positive & negative feedback from walkers
- Increased performance with more queries – knowledge-sharing
- The two update policies

Improving APS

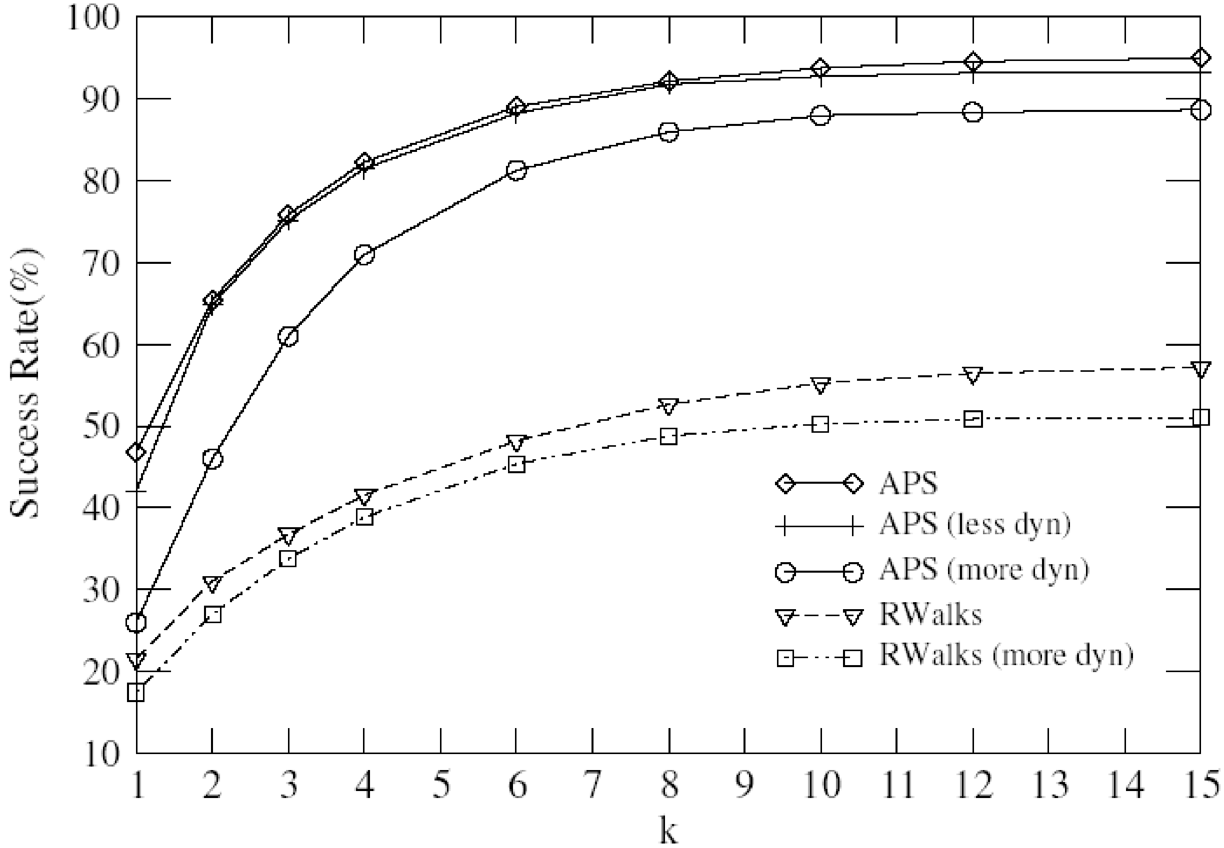
- In *swapping-APS (s-APS),* peers monitor the ratio of successful walkers to choose a policy
  - Reduced message production

- In *weighted-APS (w-APS),* indices are modified according to the object's distance from a peer
  - Preference to objects "near" the requesters

Simulations

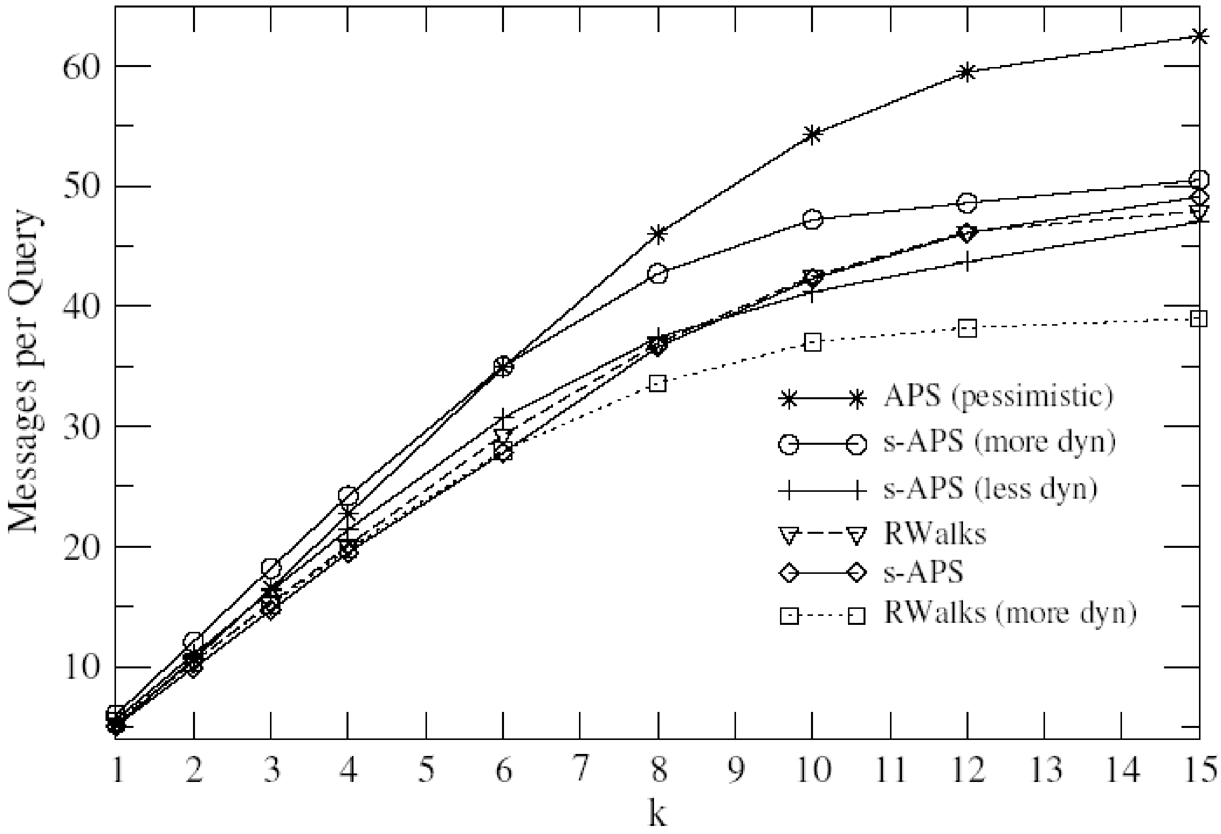
- Pure and hybrid P2P models
- Random and power-law topologies
- 100 objects of varying popularity
- Various query and replication strategies
- 3 settings of increasingly dynamic behavior
- 3 important metrics:
  - Success rate
  - Messages per query
  - Hits per query

Comparison with RWalks (1)



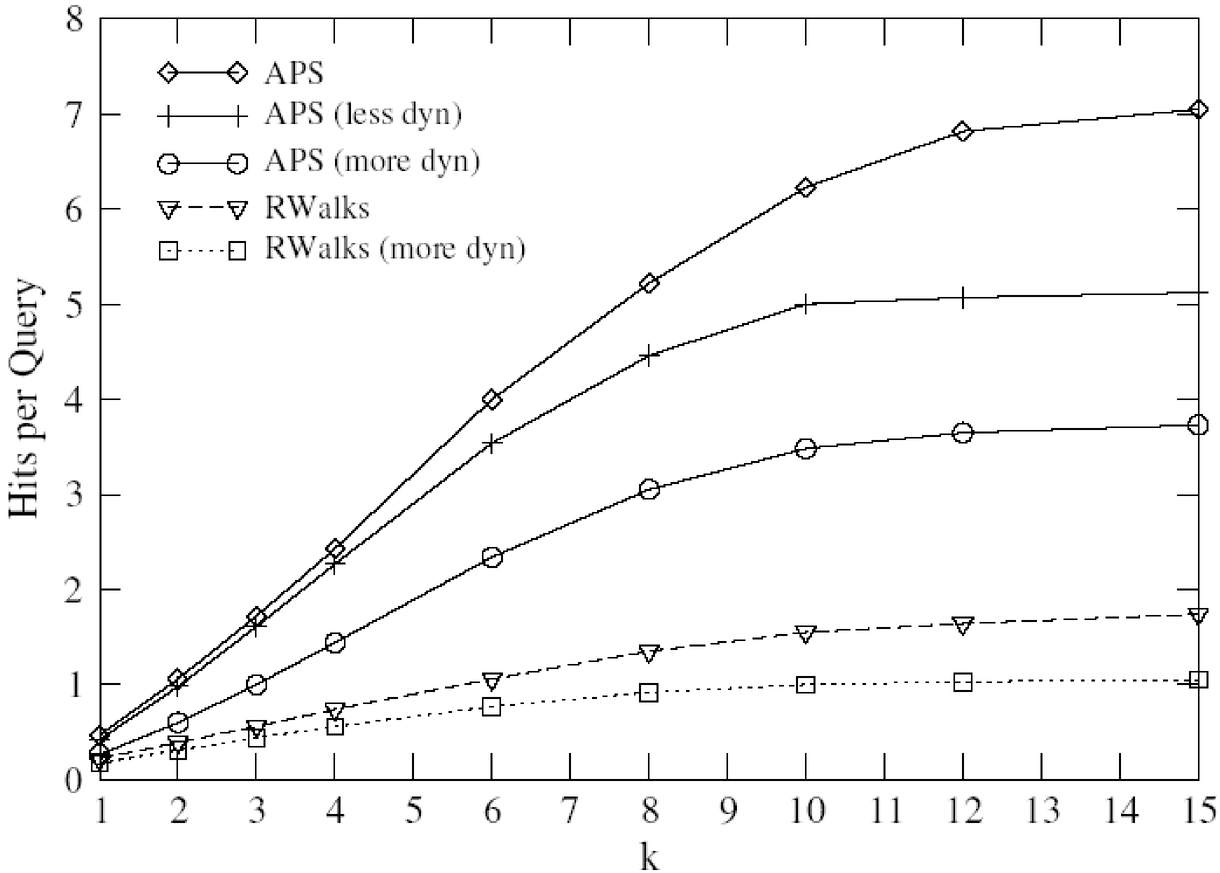
- About 40% more accurate
- < 10% decrease in the most dynamic setting

Comparison with RWalks (2)



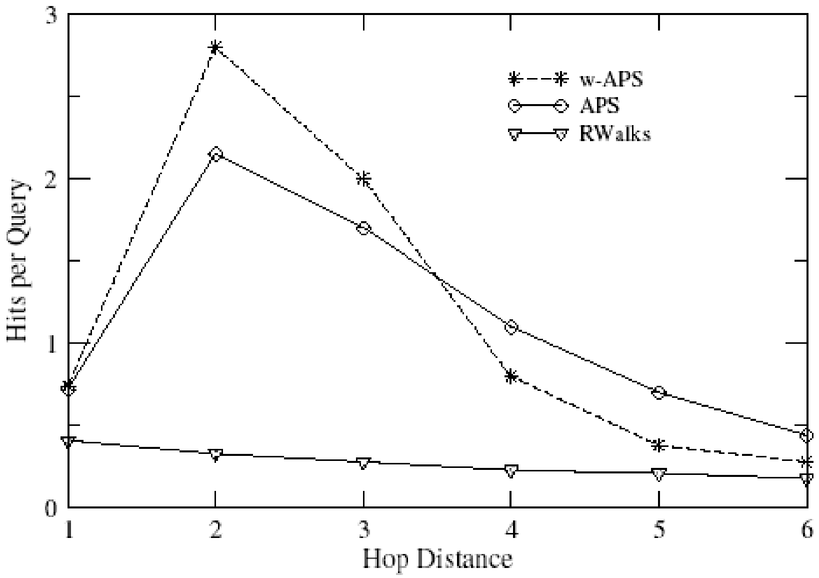
- Very close to Random Walks
- s-APS achieves message reduction

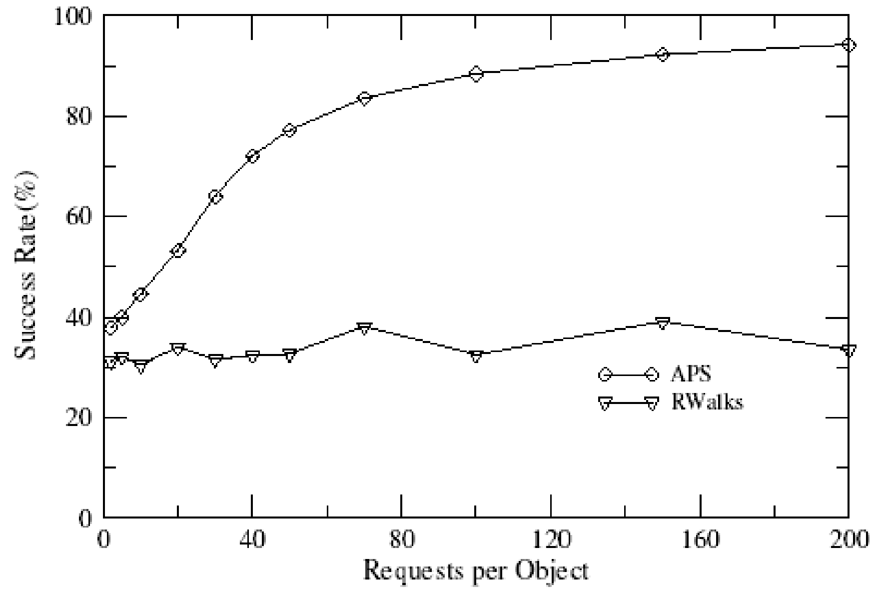
Comparison with RWalks (3)



- About 4 times more hits
- 40% decrease in the most dynamic setting

Comparison with RWalks (4)





•w-APS discovers more objects near the requesters

•APS benefits as more queries are generated

Comparison with GUESS [4]

Metric	*s-APS*			*GUESS*		
	Succ%	Mesg	Hits	*Succ%*	Mesg	Hits
Messages	97.7	**16.3**	5.22	63.9	**16.1**	1.28
	98.6	**22.0**	7.01	65.6	**22.2**	1.87
	99.7	**33.2**	11.39	84.0	**33.1**	2.55
Hits	81.0	3.2	**1.33**	63.9	16.1	**1.28**
	94.6	8.7	**3.42**	86.4	45.0	**3.70**
	97.9	16.5	**5.42**	94.5	65.1	**5.60**

- For similar messages, 4 times more hits
- For similar hits, 4-5 times fewer messages

Related Work

- Various *blind* methods
  - Flood-based (Gnutella, Modified-BFS[8], Iterative Deepening [9, 19])
  - Random Walks
- Gnutella2 [18], GUESS for hybrid networks
- *Informed* approaches:
  - Intelligent-BFS[8], DRLP[10]
  - Routing and Local Indices [3,19]
- Thorough comparison of several methods in *WebDB'03* [5]

Conclusions

- APS algorithm for object location in unstructured P2P networks
- Main features are:
  - Probabilistically directed walkers – low bandwidth consumption
  - Fast, joint learning
  - Adaptation
  - Robustness
- Favors large workloads, has *k* as an upper bound to its hits