Stochastic Models of Pull-Based Data Replication in P2P Systems

Xiaoyong Li and Dmitri Loguinov

Presented by Zhongmei Yao

Internet Research Lab
Department of Computer Science and Engineering
Texas A&M University

September 11, 2014

Agenda

- Motivation
- Single-Hop Replication
- Cascaded Replication
- Cooperative Caching
- Redundant Querying
- Conclusion

Motivation

- File replication is necessary in P2P networks to handle peer overload
- Certain P2P applications sustain periodic content updates at the source
  - Online auctions
  - Decentralized collaboration
  - Online games
- Replicas need to continuously synchronize against the source or (possibly) other replicas
  - Ensures reliability of service
  - Delivers fresh content to consumers

Motivation

- Push-based policy
  - Sources send each update to every replica
  - Structured P2P networks: source must track the status and location of each replica, which has high maintenance overhead, especially when the network structure is volatile
  - Unstructured P2P networks: replica management is achieved by message spreading, which generates large amounts of redundant traffic
- Pull-based policy
  - Replicas retrieve content when they decide so
  - This improves both scalability of the system (due to lower overhead) and availability of the data
  - However, results in possibility of staleness at the replica

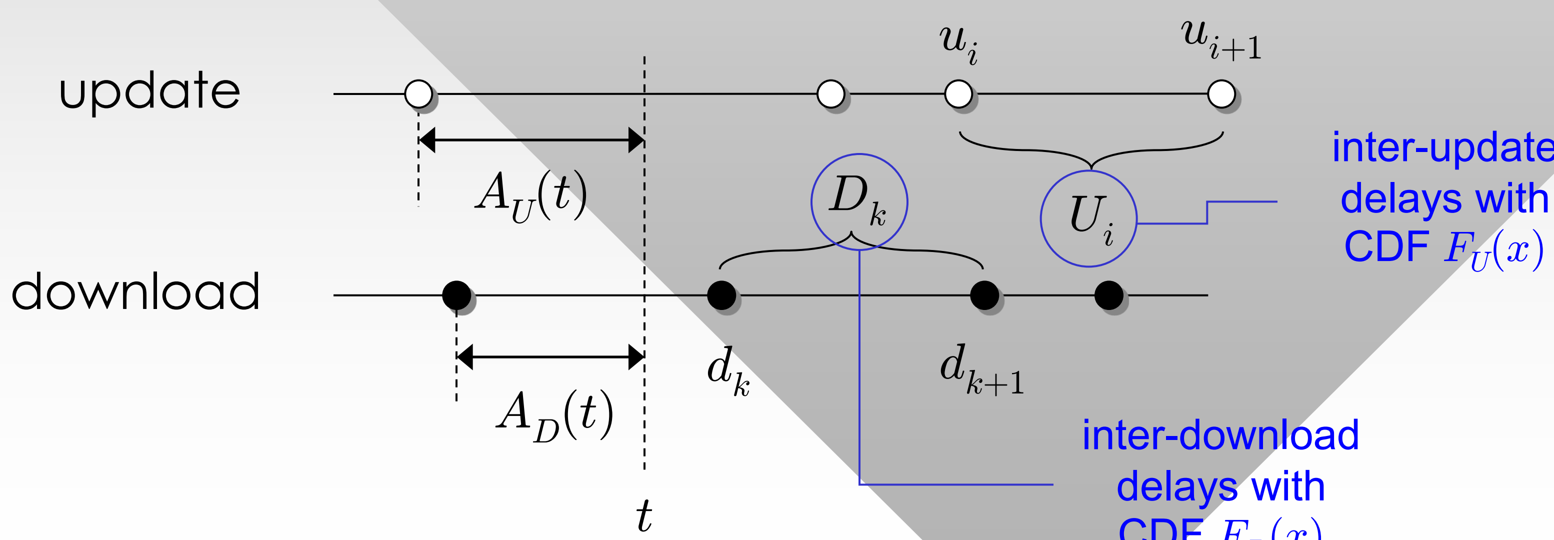
Agenda

- Motivation
- Single-Hop Replication
- Cascaded Replication
- Cooperative Caching
- Redundant Querying
- Conclusion

Single-Hop Replication (Model)

- Information requests indicated by arrows:

- Model
  - Source experiences random updates via process N_U
  - Replica periodically downloads the content via process N_D



Single-hop Replication (Result)

- Equilibrium ages of update and download processes are given by A_U and A_D with the following distributions:

update rate $1/E[U_i]$

$$G_U(x) := P(A_U < x) = \mu \int_0^x (1 - F_U(y))dy$$

download rate $1/E[D_k]$

$$G_D(x) := P(A_D < x) = \lambda \int_0^x (1 - F_D(y))dy$$

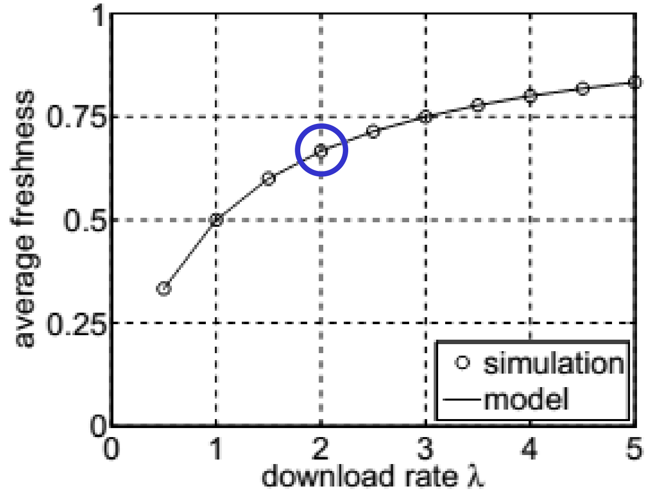
- Theorem 1: Freshness probability is given by:

density $dG_D(x)/dx$

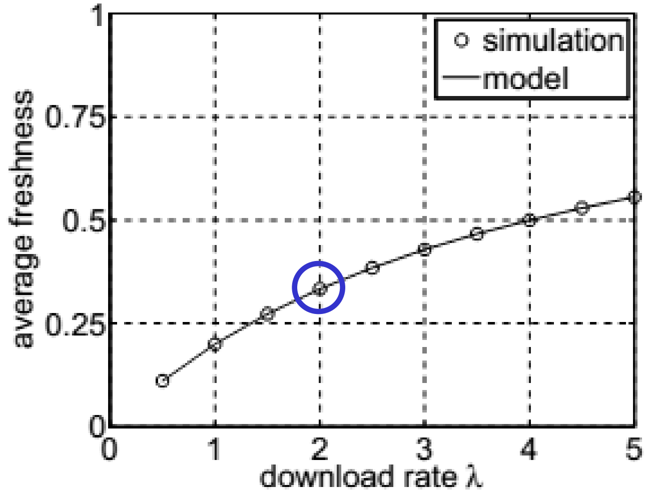
$$p = E[\bar{G}_U(A_D)] = \int_0^\infty \bar{G}_U(x) g_D(x) dx$$

complementary CDF $1 - G_U(x)$

Single-hop Replication (Simulations)



(a) Pareto U, constant D



(b) constant U, Pareto D

Fig. 3. Simulation results of (8) under $\mu = 2$.

- Observe that
  - The model matches simulations well
  - Constant download intervals perform significantly better against Pareto update cycles in (a) than the other way around in (b)

Best Download Strategy

- With the same download rate λ, what distribution of synchronization intervals $F_D(x)$ is best?
- Definition 1: Variable X is stochastically larger than Y in second order, i.e., $X \geq_{st}^{2} Y$, if

$$\int_0^x \bar{F}_X(y)dy \geq \int_0^x \bar{F}_Y(y)dy \text{ for all } x \geq 0$$

- Theorem 2: For a given download rate λ and fixed update process N_U, freshness increases if download delays become stochastically larger in second order
  - Similarly, for a given update rate μ and fixed download process N_D, freshness increases if inter-update delays become stochastically smaller in second order

Best Download Strategy

- Lemma 3: For a given mean, a constant stochastically dominates all other random variables in second order
- Theorem 3: For a fixed download rate λ, constant inter-download delays are optimal under all N_U
- To understand the discussion that follows, we need more terminology
- Definition 2: A random variable X is NWU (new worse than used) if

$$P(X > x + y | X > y) \geq P(X > x)$$

  - If the inequality is reversed, it is NBU (new better than used)

Best Download Strategy

- Suppose X is NWU (e.g., Pareto), Y is memoryless (exponential), and Z is NBU (e.g., constant) such that $E[X] = E[Y] = E[Z]$. Then, $Z \geq_{st}^2 Y \geq_{st}^2 X$

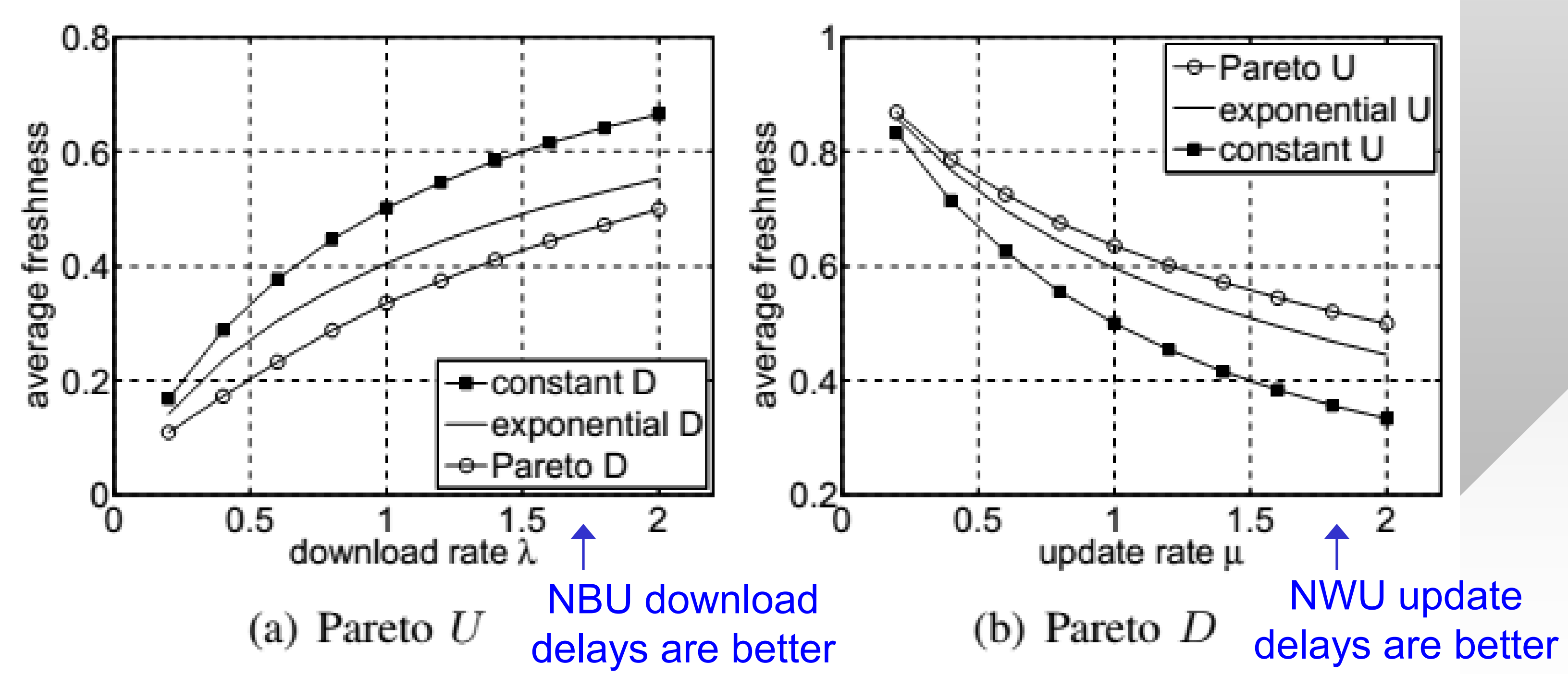


Fig. 4. Ordering of freshness under different families of distributions.

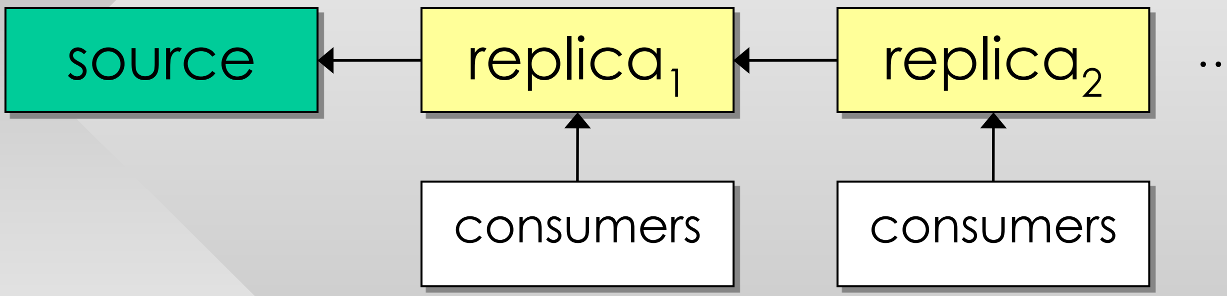
Computer Science, Texas A&M University

Agenda

- Motivation
- Single-hop Replication
- Cascaded Replication
- Cooperative Caching
- Redundant Querying
- Conclusion

Cascaded Replication (Model)

- Replicas are organized into a tree, where each node asks its parent for updates (source is at the root):



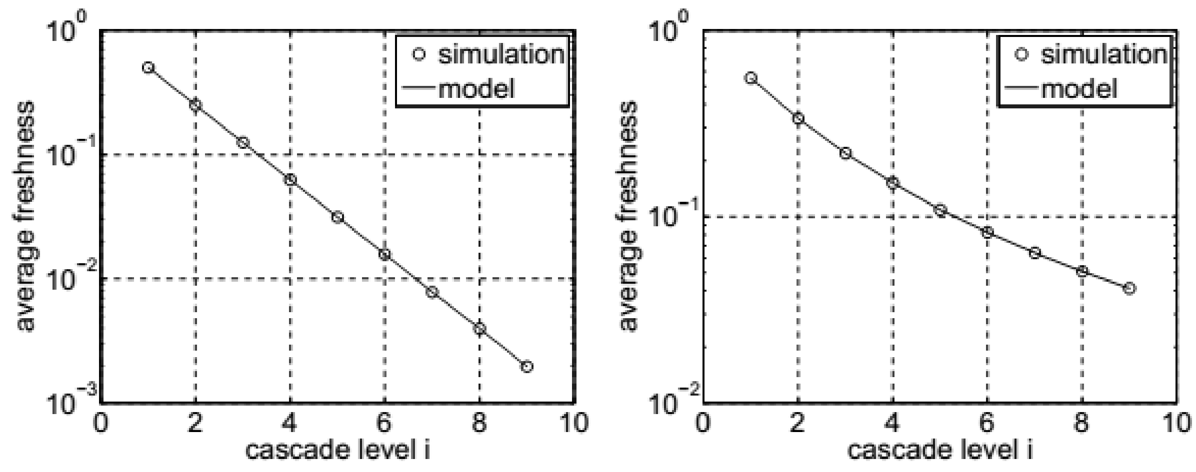
- Nodes at depth i use download delays $D^{(i)} \sim F_D^{(i)}(x)$
- Theorem 4: Freshness probability at depth i is

$$p_i = E[\bar{G}_U(Q_i)], \text{ where } Q_i = A_D^{(1)} + A_D^{(2)} + \cdots + A_D^{(i)}$$

- The order of replicas along each branch has no effect on freshness at the leaves!

Cascaded Replication (Simulation)

- Therefore, one should use slow download rates λ near the root (to avoid overloading the source), faster near the bottom of the tree
- Simulations match the model well
- In (b), freshness decays slower, indicating that Pareto updates U are easier to scale to a large number of users than exponential



(a) exponential U (b) Pareto U

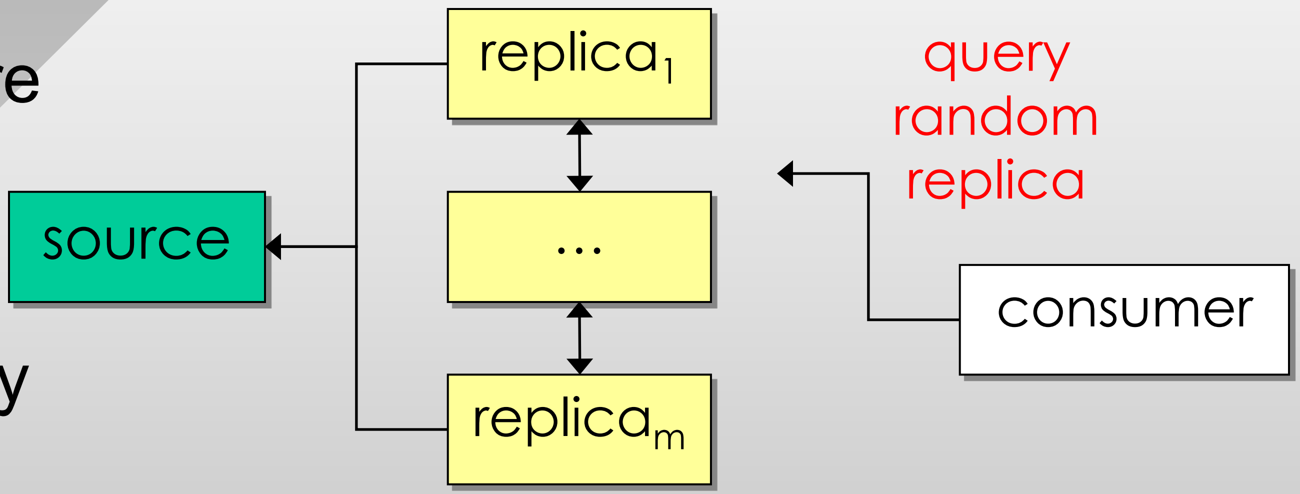
Fig. 9. Cascaded freshness with exponential D and $\lambda = \mu = 2$.

Agenda

- Motivation
- Single-hop Replication
- Cascaded Replication
- Cooperative Caching
- Redundant Querying
- Conclusion

Computer Science, Texas A&M University

Cooperative Caching (Model)

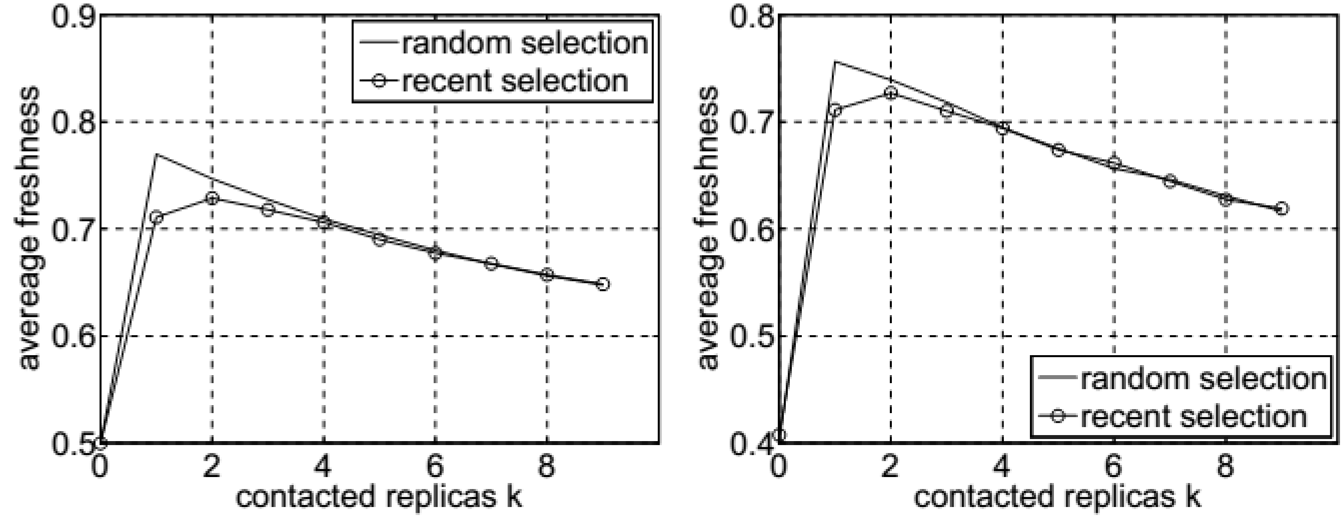


- All nodes are at level 1
- A replica may not only contact the source (using download process N_D), but also ask for updates from other peer caches (using communication process N_C)
- At each random point of N_C, the node contacts k peers concurrently and selects the freshest version
- Consumers query a single node among m available

Cooperative Caching (Simple Scenario)

- To aid in cooperation between replicas
  - Source maintains a replica list ordered by the most recent download timestamp
- Strategies to choose k peers
  - Random: uniformly among all m
  - Recent: peers with the largest contact timestamp
- Assume ν is the rate at which process N_C generates points at each node within the replica cluster
- Objective: Given a fixed communication bandwidth $k\nu$, choose such k and ν that provide the highest freshness

Cooperative Caching (Simple Scenario)



(a) exponential D, $\nu = 10/k$ (b) Pareto D, $\nu = 10/k$

- Random selection with $k = 1$ provides best results!
  - Recent selection is biased towards peers that were the most up-to-date at previous download time d_k, but are no longer the freshest by next download instance d_{k+1}

Cooperative Caching (Full System)

- Suppose all peers have the same bandwidth B and let s be the service rate that each replica can offer to clients
- The object is to maximize the combined service rate

$R := ms = m(B - k\nu - \lambda)$ subject to:

$$\begin{cases} m\lambda \leq B \\ p(\lambda, k, \nu) \geq 1 - \epsilon \end{cases}$$

number of replicas

average freshness with cooperative caching

user defined parameter

Cooperative Caching (Full System)

- Paper explains why an optimal cluster size m exists
- With $\epsilon = 0.5$, the improvement in service rate between cooperative and non-cooperative cases reaches a factor of 6.6

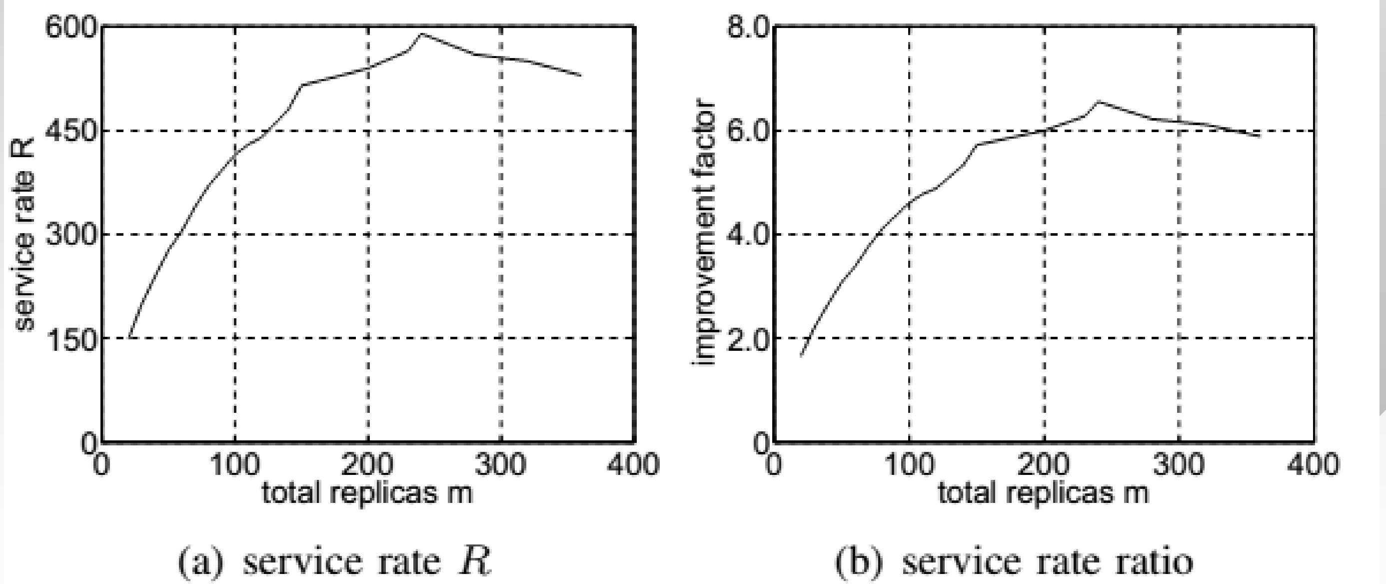


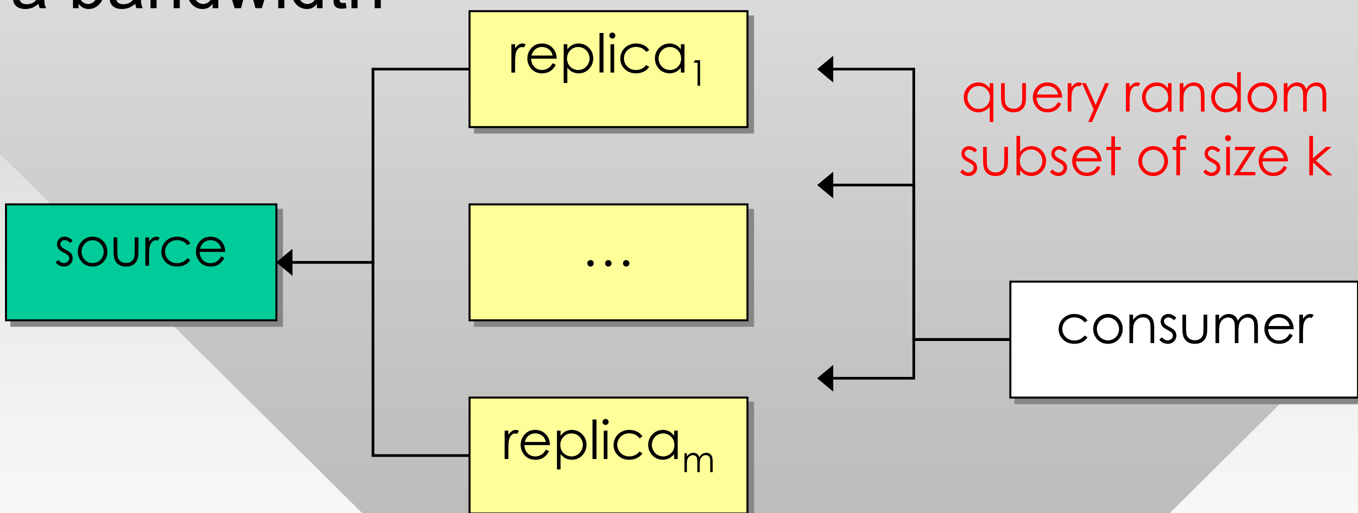
Fig. 12. Effect of m on service rate R ($\epsilon = 0.5, \mu = 1, B = 10$). All distributions are exponential.

Agenda

- Motivation
- Single-hop Replication
- Cascaded Replication
- Cooperative Caching
- Redundant Querying
- Conclusion

Redundant Querying

- All replicas are at depth 1 and do not cooperate, but consumers are allowed to concurrently query k replicas and retrieve the freshest copy, where each cache node is still under a bandwidth constraint B



- This can be reduced to previously studied models by constructing a single download process consisting of a superposition of k processes $\{N_D^{(i)}\}_{i=1}^k$

Redundant Querying

- For sufficiently large k, this superposition tends to a Poisson process for which we have:

target freshness

TABLE II
IMPROVEMENT FROM REDUNDANT QUERYING COMPARED TO NON-REDUNDANT

$1-\epsilon$	Pareto D		exponential D		constant D	
	m^*/m	R^*/R	m^*/m	R^*/R	m^*/m	R^*/R
0.1	21.8	1.1	20	1	18.0	0.90
0.3	26.6	1.3	20	1	14.6	0.73
0.5	32.8	1.6	20	1	12.5	0.63
0.7	42.3	2.1	20	1	11.2	0.56

ratio of optimal cluster size

ratio of service capacity

- For constant D, the redundant case performs worse than the non-redundant!

Conclusion

- We proposed a general framework for modeling lazy synchronization and derived the probability of freshness under general update/download processes
- We then extended these results to cascaded and cooperative replication, finding solutions to a number of optimization problems in those contexts
- Finally, we examined redundant querying and found cases when doing so was detrimental to system performance

Questions?