

Asymptotically-Optimal Incentive-Based En-route Caching Scheme

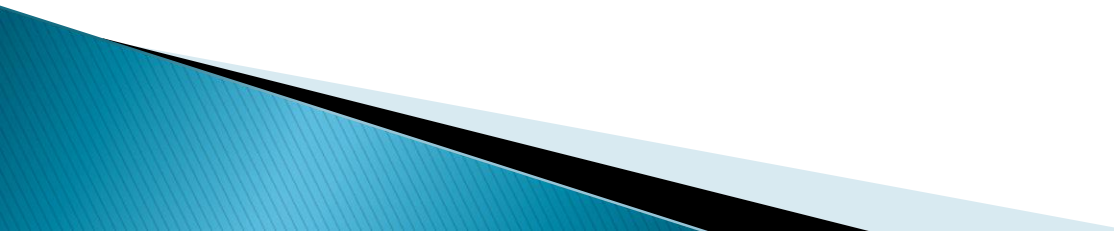
Ammar Gharaibeh*, Abdallah Khreishah*, Issa Khalil†, Jie Wu‡

*New Jersey Institute of Technology

†Qatar Computing Research Institute

‡Temple University

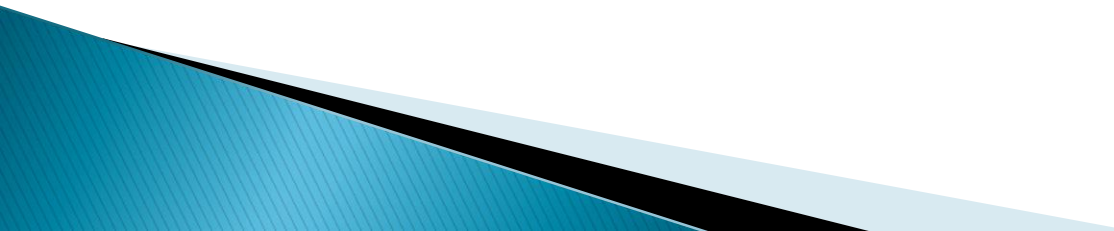
Introduction

- ▶ The Internet traffic is dominated by content retrieval services.
 - ▶ Content Replication schemes are used by CDN
 - Shorter delivery times.
 - Better scalability.
 - Better performance.
 - Better energy efficiency.
 - No caching at intermediate routers
 - Replication is done offline.
- 

Introduction

- ▶ Several caching techniques have emerged.
- ▶ Example: CCN
 - content naming
 - caching at intermediate nodes.
- Requires changes to the TCP/IP protocol stack.
- Caching decision is not designed to optimize network performance
- Requires broadcasting of *Interest* packets.

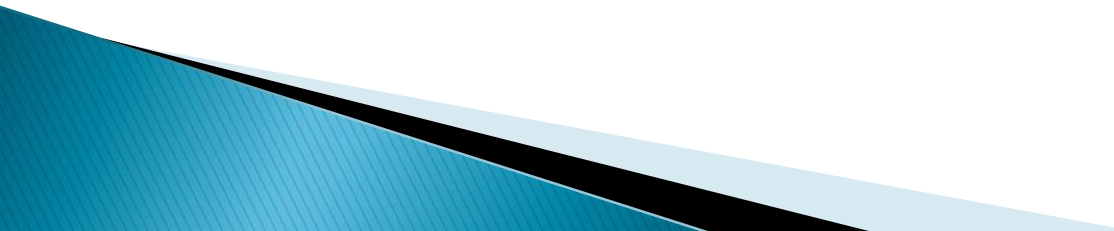
Introduction

- ▶ What are the factors that affect achieving the maximum traffic savings?
 - ▶ Which contents are to be cached in order to achieve the same objective?
- 

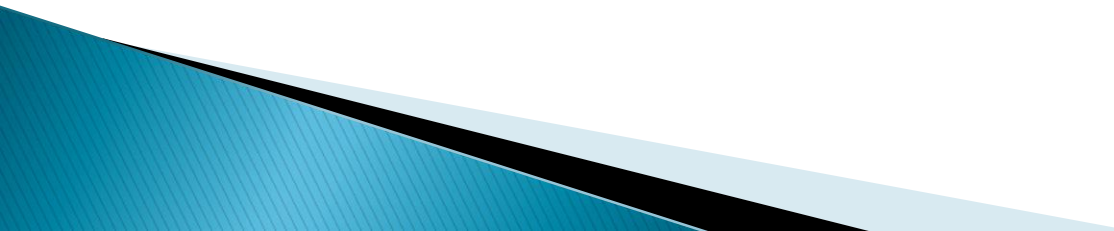
Related Work

- ▶ Different Objectives
- ▶ Such objectives include:
 - Optimizing joint latency-traffic problem [Guan *et. al.* '13].
 - Reduce cache redundancy [Psaras *et. al.* '12].
 - Maximizing cache cooperation through dynamic request routing [Dai *et. al.* '12].

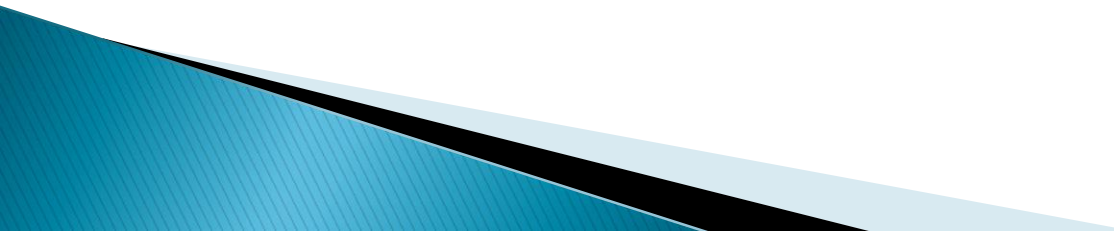
Objectives

- ▶ Optimality.
 - ▶ Providing incentive for caching nodes.
 - ▶ Easily integrated in the TCP/IP suite
- 

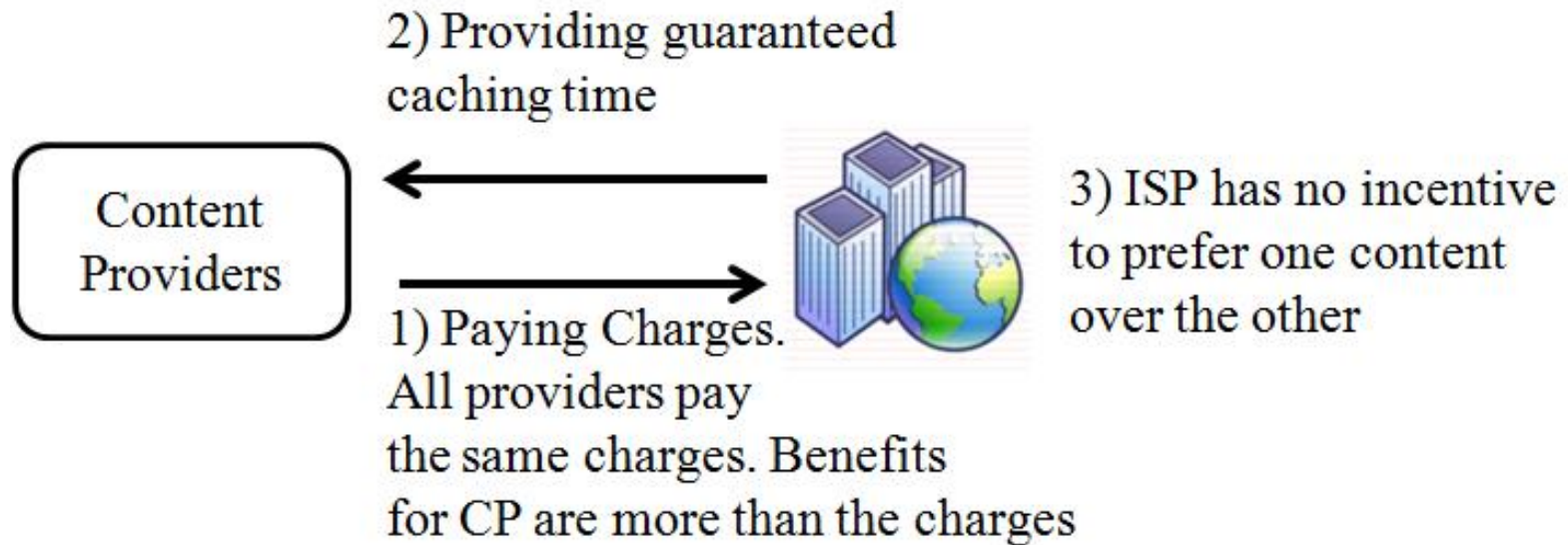
Contributions

- ▶ Asymptotically (in terms of number of nodes) Optimal (in terms of traffic savings) online caching algorithm under common cases.
 - ▶ Incentives for the caching nodes.
 - ▶ Low complexity.
 - ▶ Easily implemented in a distributed fashion.
 - ▶ Easily integrated in TCP/IP suite.
 - ▶ Reactive caching
- 

En-Route Caching

- ▶ A request for a content is only forwarded along the path to the content's source.
 - ▶ Reduces the amount of forwarded *Interest* packets as opposed to CCN.
 - ▶ Easily implemented.
- 

Settings

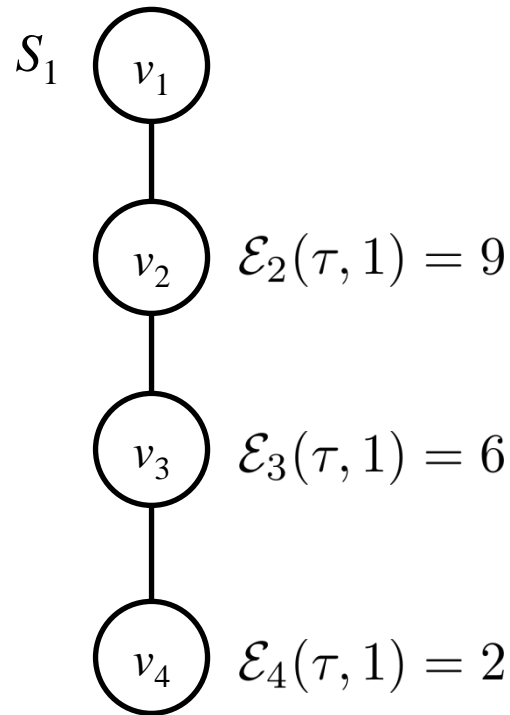


Settings

- ▶ Total Traffic Savings of caching in the interval $[0, t]$ is defined as:

$$\sum_{\tau=0}^t \sum_{i=1}^n \sum_{j=1}^m d_i(\tau_0, j) \mathcal{E}_i(\tau_0, j) I(a_i(\tau, j))$$

Traffic Savings Example



- ▶ Caching β_1 at v_3 alone for a single time slot will yield a saving of

$$\mathcal{E}_3(\tau, 1) \times d_3(\tau, 1) = 6 \times 2 = 12$$

Definitions

- ▶ Offline vs. Online Algorithms:
 - Offline algorithm has the complete knowledge of the future.
 - Offline algorithm knows *when, where, and how many times* a content will be requested.
 - This knowledge leads to the optimal performance.
 - Online algorithm does not possess such knowledge.
 - Online algorithm has to make a caching decision based on the available information at the time of the content arrival.

Offline vs. Online Example

S_X S_Y

v_1	
-------	--

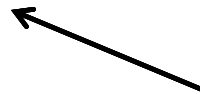
v_2	Y
-------	-----

$W_3(\tau, X) = 1$
 $W_3(\tau, Y) = 1$

v_3	✗
-------	---

$W_4(\tau, X) = 1$
 $W_4(\tau, Y) = 10$

v_4	X
-------	-----



Second request for X

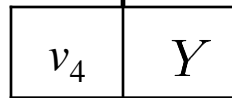
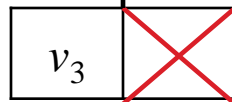
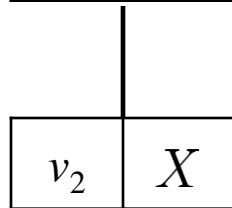
- ▶ v_3 has a full cache
- ▶ v_4 and v_2 can cache one content at most.
- ▶ Contents X and Y will be requested twice by v_4

Savings = 3

Offline vs. Online Example

S_X S_Y

v_1	
-------	--

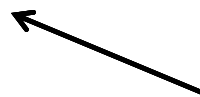


$W_3(\tau, X) = 1$
 $W_3(\tau, Y) = 1$

$W_4(\tau, X) = 1$
 $W_4(\tau, Y) = 10$

- ▶ v_3 has a full cache
- ▶ v_4 and v_2 can cache one content at most.
- ▶ Contents X and Y will be requested twice by v_4

Savings = 30



Second request for X

Offline vs. Online Example

- ▶ The offline algorithm knows in advance the order in which the contents will be requested, and can decide to cache X at v_2 and Y at v_4 , to achieve traffic savings of 31.

Definitions

- ▶ **Competitive Ratio:**

- Defined as the ratio of the performance achieved by the best offline algorithm to the performance achieved by the online algorithm.

$$\sup_t \sup_{\substack{\text{all input} \\ \text{sequences in } [0,t]}} \frac{P_{off}}{P_{on}}.$$

- ▶ We show that the best online algorithm has a competitive ratio which is lower bounded by $\Omega(\log n)$

Cost-Reward Caching (CRC) Algorithm

- ▶ Exponential caching cost function.

$$C_i(\tau, j) = D_i(\mu^{\lambda_i(\tau, j)} - 1)$$

where μ is a constant that depends on the number of the nodes in the network.

The Algorithm

Algorithm 2 Cost-Reward Caching (CRC)

New request for β_j arriving at node i at time t_0

$\forall \tau \in \{t_0, \dots, t_0 + T_j(t_0)\}$, Compute $\lambda_i(\tau, j)$, $C_i(\tau, j)$

if $\sum_{\tau=t_0}^{t_0+T_j(t_0)} \mathcal{E}_i(\tau, j) d_i(t_0, j) \geq \sum_{\tau=t_0}^{t_0+T_j(t_0)} \frac{r_j}{D_i} C_i(\tau, j)$ **then**

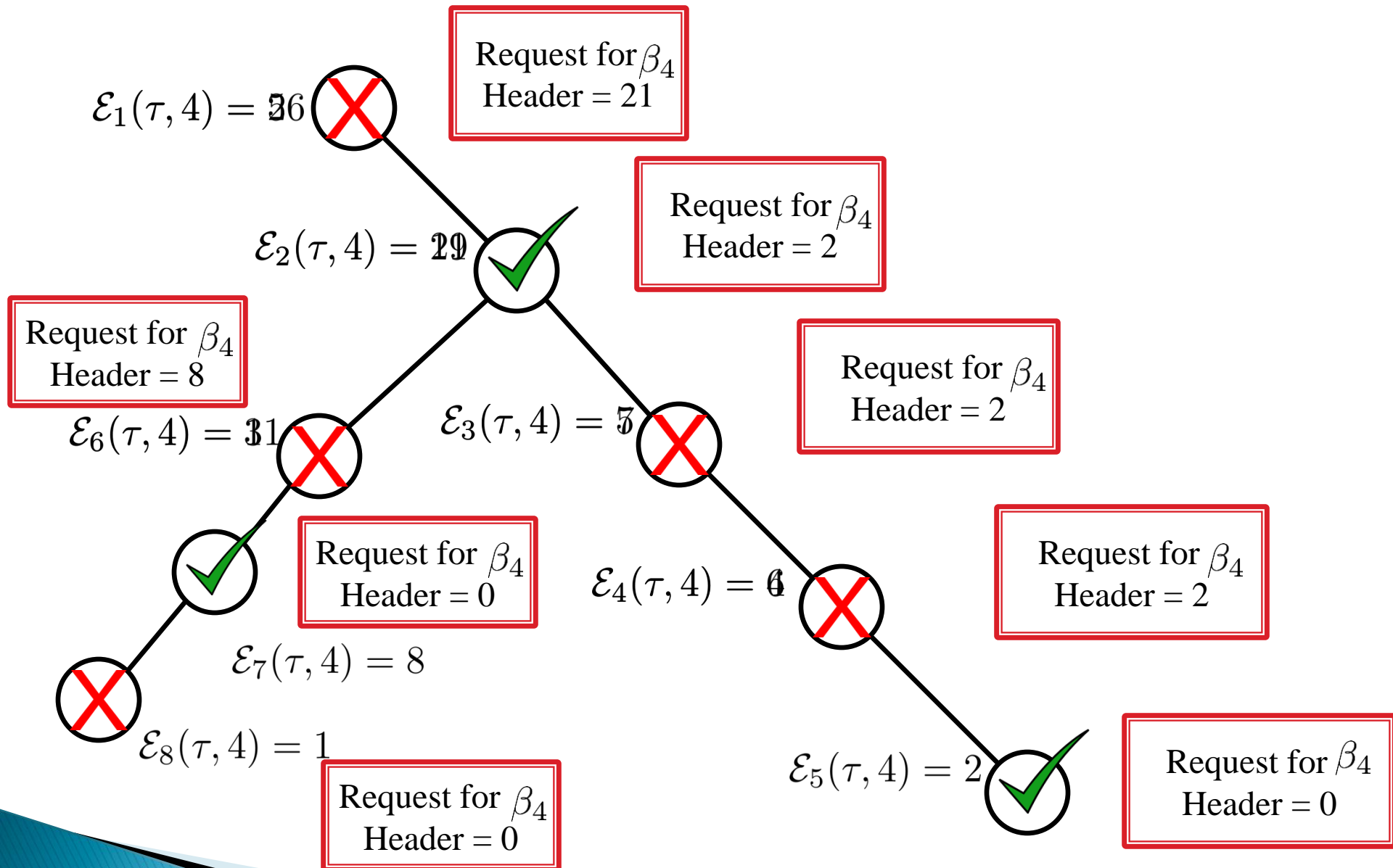
Cache β_j on node i

$\tau_0(i, j) = t_0(i, j)$

$\forall \tau \in \{t_0, \dots, t_0 + T_j(t_0)\}$, $\lambda_i(\tau, j+1) = \lambda_i(\tau, j) + \frac{r_j}{D_i}$

else

Do not cache



CCN vs. CRC

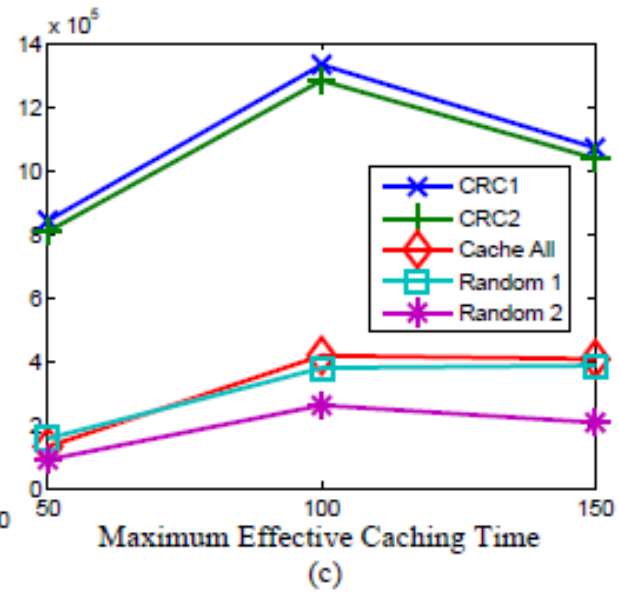
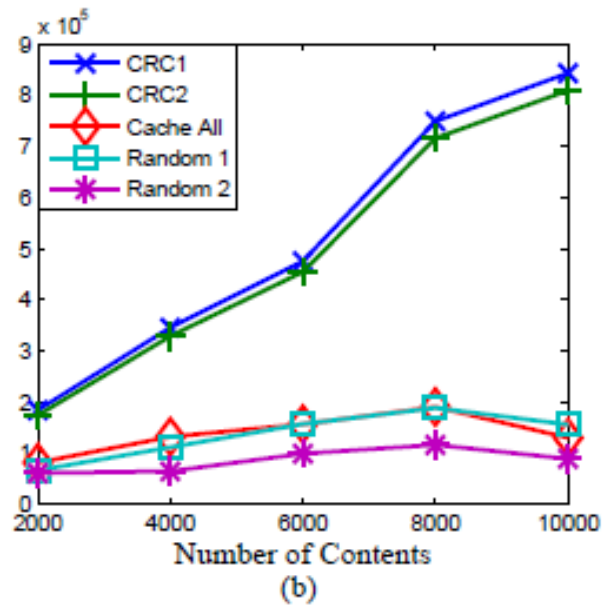
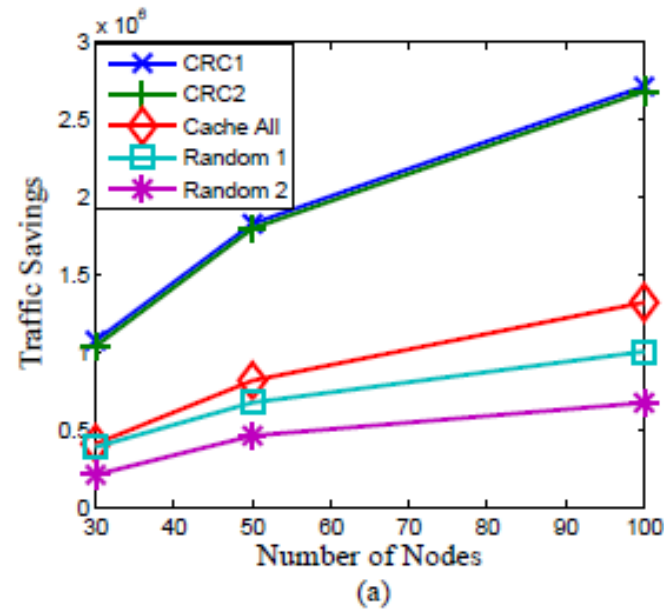
	CCN	CRC
Caching Decision	Not designed to optimize network performance	Designed to maximize the traffic savings
Changes to TCP/IP protocol stack	Requires a radical change to the protocol stack	Does not require changes to the protocol stack
Interest Packet Forwarding	Broadcast	En-route

Performance Analysis

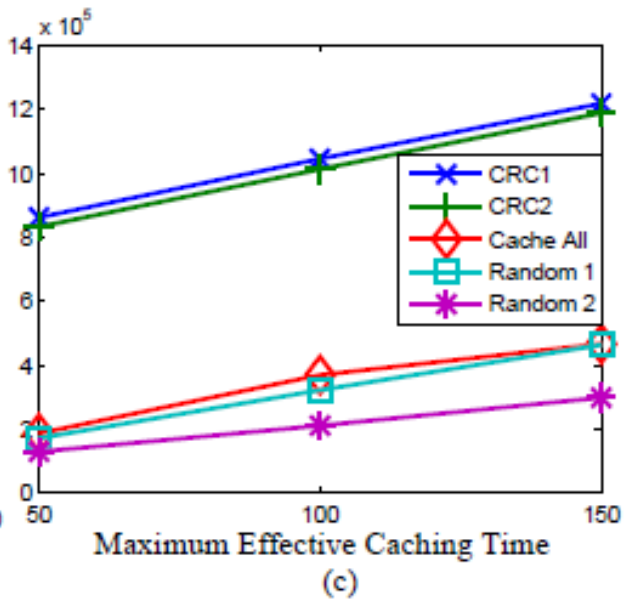
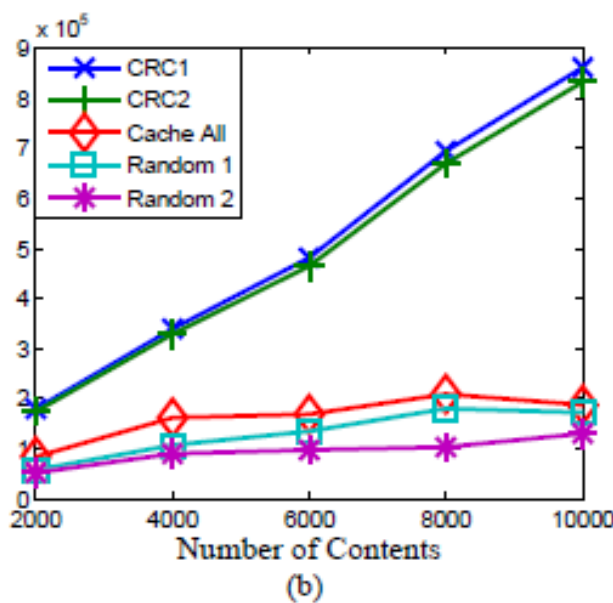
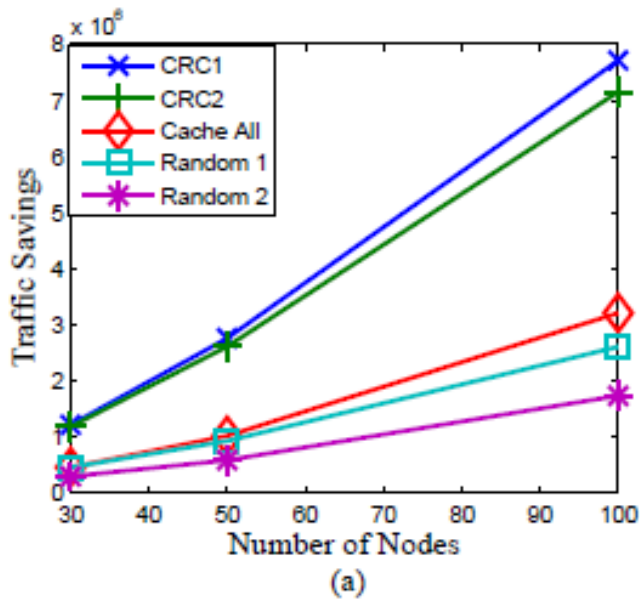
- ▶ Capacity constraints are not violated
- ▶ The algorithm achieves a $\mathcal{O}(\log n)$ competitive ratio.
- ▶ Any online algorithm has a competitive ratio which is lower bounded by $\Omega(\log n)$



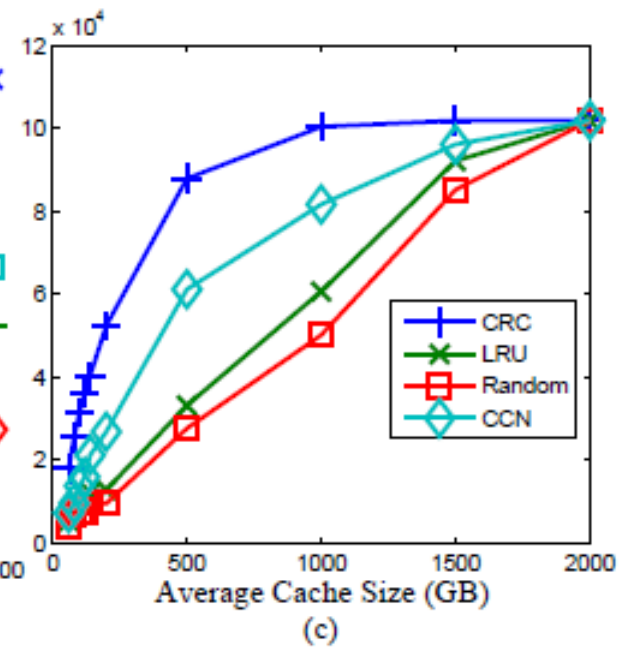
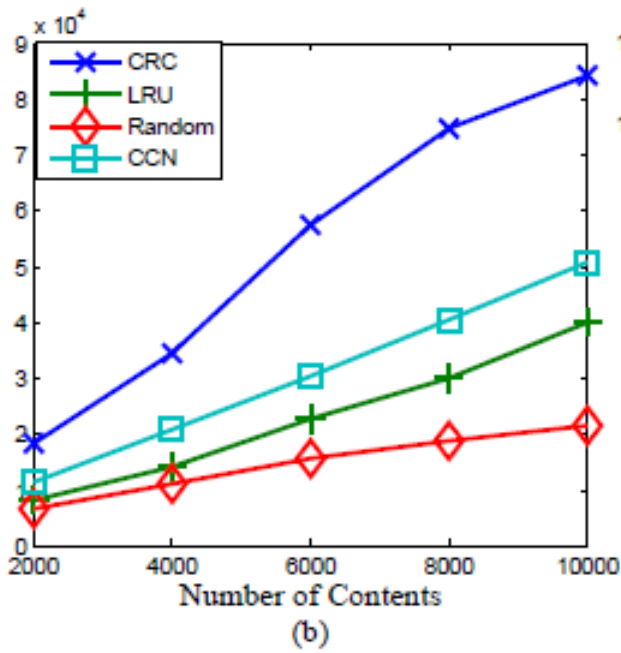
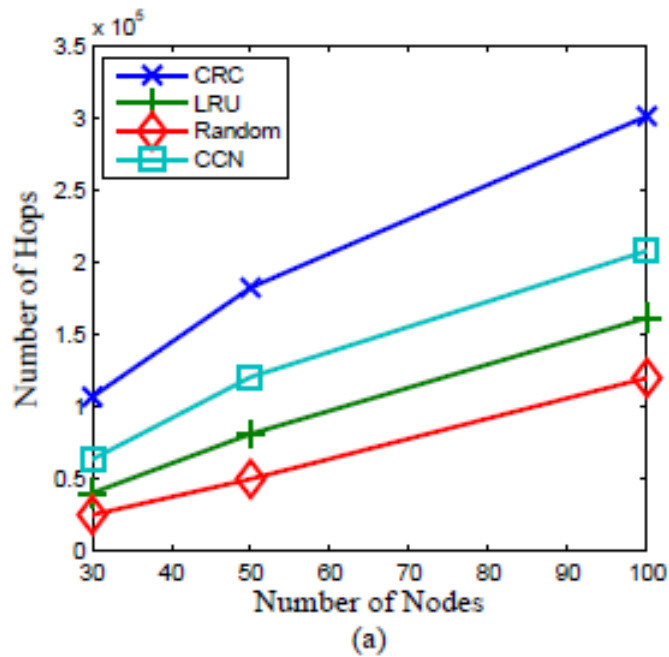
Results on Random Topology



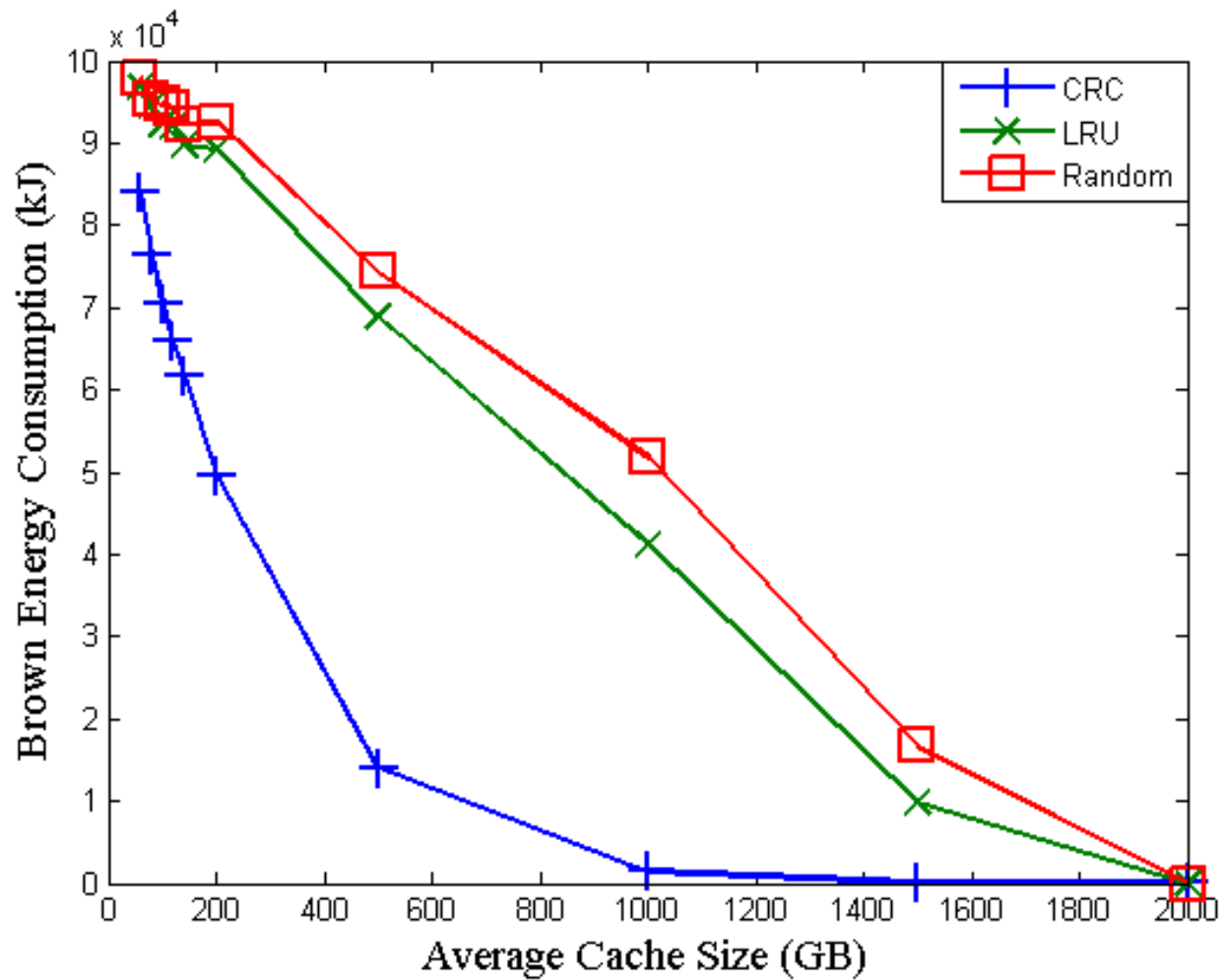
Results on Small World Topology



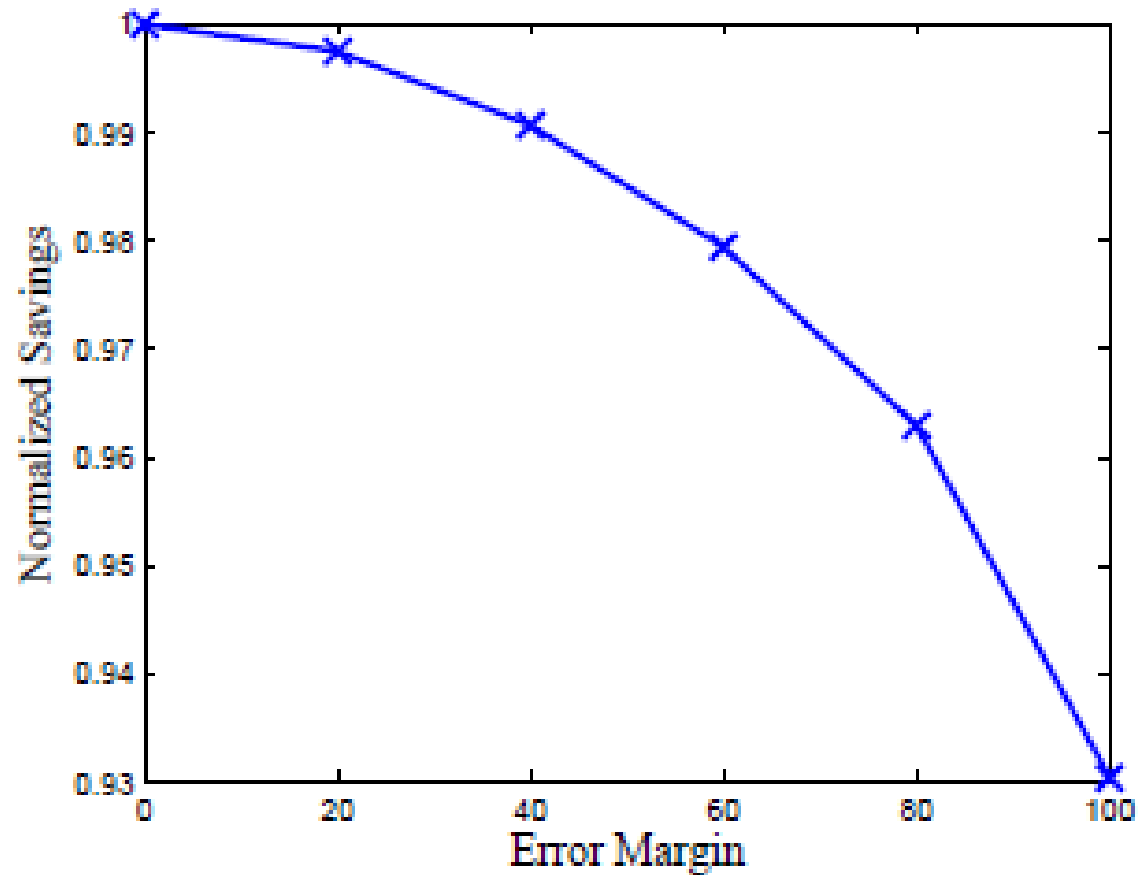
Replacement CRC



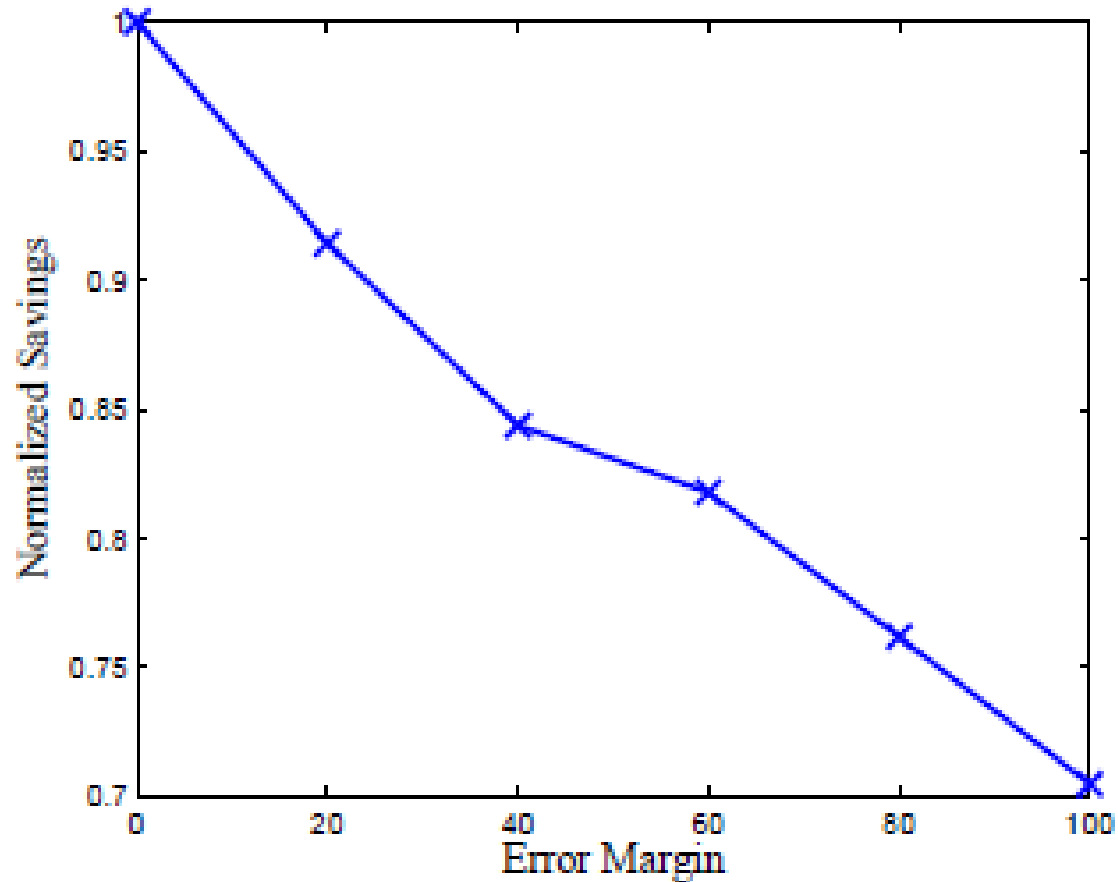
Energy CRC



CRC Performance vs. Expectation Errors



CRC Performance vs. Errors in Effective Caching Time



Future Work

- ▶ Theoretical analysis of the CRC extensions
- ▶ Incorporating learning techniques to estimate the values of content popularity expectations.

Thank You

Appendix: Performance Analysis

► Assumptions:

$$1 \leq \frac{1}{n} \cdot \frac{\mathcal{E}_i(\tau, j) b_i(j)}{r_j T_j(\tau)} \leq F \quad \forall j, \forall i \neq S_j, \forall \tau$$

$$r_j \leq \frac{\min D_i}{\log(\mu)} \quad \forall j$$

Appendix: Performance Analysis

- ▶ The CRC algorithm does not violate the capacity constraints.
- ▶ Proof: Let β_j be the first content that caused a violation in the capacity constraints at node i . Using the assumptions from the previous slide, along with the definition of the cost function, we reach

$$\frac{r_j}{D_i} C_i(\tau, j) \geq \mathcal{E}_i(\tau, j) d_i(t_0, j)$$

- ▶ From the definition of the algorithm, β_j should not be cached at node i

Appendix: Performance Analysis

- ▶ CRC algorithm achieves $\mathcal{O}(\log n)$ competitive ratio under common cases.
- ▶ Lemma: Traffic savings gained by CRC algorithm is lower bounded by the sum of the caching costs.
- ▶ Lemma: The additional traffic savings of the offline algorithm is upper bounded by the sum of the caching costs.
- ▶ Using these lemmas, the traffic savings of the CRC is within $\mathcal{O}(\log n)$ of the traffic savings achieved by the offline algorithm under common cases.

Appendix: Performance Analysis

- ▶ Proposition: Any online algorithm has a competitive ratio which is lower bounded by $\Omega(\log n)$
- ▶ Proof: We show this proposition by giving an example network, such that the best online algorithm competitive ratio is lower bounded by $\Omega(\log n)$

Settings

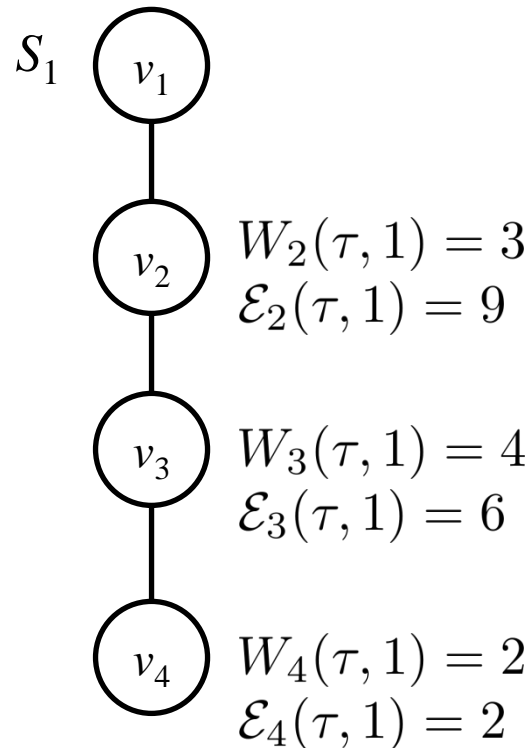
- ▶ A network is represented by a graph $G(V, E)$, where every node $i \in V$ has a caching capacity D_i .
- ▶ For each contents β_j , we define the following:
 - S_j : The source of β_j
 - r_j : The size of β_j
 - $T_j(\tau)$: The effective cache duration for β_j when the first request appears at time τ
- ▶ Slotted time system.

Settings

- ▶ $d_i(\tau, j)$: Number of hops between node i and the first node caching β_j along the path to S_j
- ▶ $\mathcal{E}_i(\tau, j)$: Total number of requests for β_j to be served from the cache of node i at time τ
- ▶ $\tau_0(i, j)$: The time when β_j is cached at node i
- ▶ Total Traffic Savings of caching in the interval $[0, t]$ is defined as:

$$\sum_{\tau=0}^t \sum_{i=1}^n \sum_{j=1}^m d_i(\tau_0, j) \mathcal{E}_i(\tau_0, j) I(a_i(\tau, j))$$

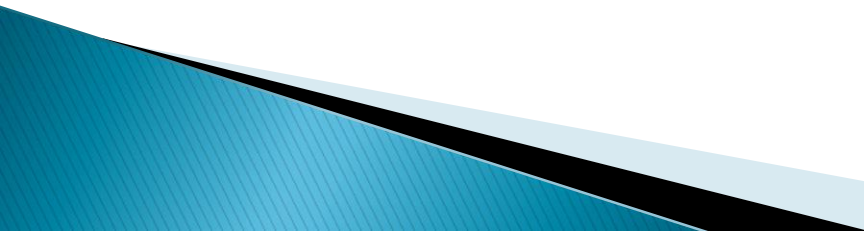
Traffic Savings Example



- ▶ Caching β_1 at v_3 alone for a single time slot will yield a saving of

$$\mathcal{E}_3(\tau, 1) \times d_3(\tau, 1) = 6 \times 2 = 12$$

Settings

- ▶ The caching nodes charge the contents providers in order to cache their contents, thus providing the incentives for the nodes to cache.
 - ▶ The charging policy is assumed to be the same for all contents providers. This prevents favoring one content over the other.
 - ▶ The caching nodes has to provide guarantees for the content providers. To this end, we consider non-preemptive caching scheme.
 - ▶ In the simulations, we provide results for an extension to the basic scheme that considers content replacement.
- 

The Algorithm (CRC)

▶ Definition:

- The relative load on a caching node i at time τ when a request for β_j arrives is the sum of sizes of currently cached contents divided by the cache capacity and is defined as:

$$\lambda_i(\tau, j) = \sum_{\substack{k:k < j \\ k \in \text{Cache}_i(\tau)}} \frac{r_k}{D_i}$$

Practical Issues

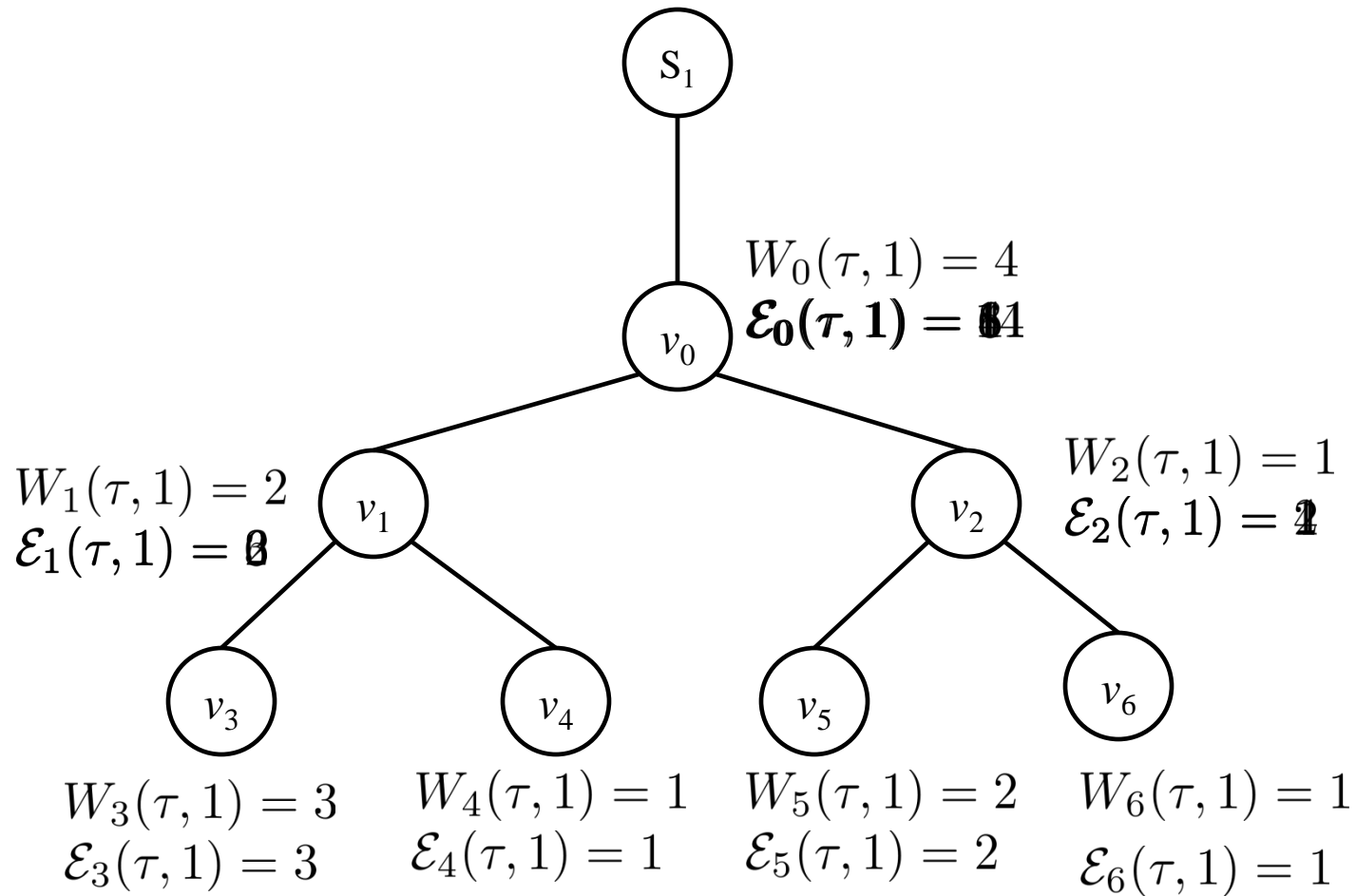
- ▶ Calculating Initial Content Expectation Values:
 - Calculated before any request for any content appears.
 - For each content, a caching tree rooted at the content's source is established.
 - Based on the content popularity at each caching node, number of clients connected to each caching node, and the caching tree, the initial values of contents expectations ($\mathcal{E}_i(\tau, j)$) are calculated.

Practical Issues

Algorithm 3 Initial Content Popularity Expectation Calculation

for each content $\beta_j = \{S_j, r_j, T_j(\tau)\}$ **do**
 $CachingTree(j) \leftarrow$ build the traditional path tree
 rooted at S_j
 for each caching node $i \in CachingTree(j)$ **do**
 Calculate $W_i(\tau, j)$
 Initialize $\mathcal{E}_i(\tau, j) \leftarrow W_i(\tau, j)$
 end for
 for each node $z \in Ancestor(i)$ in $CachingTree(j)$
 do
 $\mathcal{E}_z(\tau, j) = \mathcal{E}_z(\tau, j) + W_i(\tau, j)$
 end for
end for

Practical Issues



Practical Issues

- ▶ Effective Caching Duration:
 - Depends on the content arrival time.
 - Requires the broadcast of the first arrival time to all other nodes.
 - Additional overhead to broadcast the first arrival time is negligible compared to the reduction of the broadcasted *Interest* packets achieved through En-Route caching.

Simulation Results

- ▶ Five schemes are simulated
 - CRC: Represents our basic algorithm
 - CRC Version 2: A variation of CRC, where the content is retrieved from the closest node that has a copy of the content, not necessarily along the path to the content's source.
 - All Cache
 - Random Caching Version 1: The probability of caching a content is proportional to the content's popularity
 - Random Caching Version 2: The probability of caching a content is proportional to the content's popularity and inversely proportional to the relative load of the caching node.

Extensions to CRC Algorithm

- ▶ We introduce three different extensions of the basic CRC algorithm:
 - **Energy CRC:** The traffic savings are measured based on the energy saved on the upstream path from the caching node up to the first node that already has the content in its cache.
 - **Replacement CRC:** In this extension, content replacement is allowed.
 - **Energy CRC with Replacement:** combination of the previous two extensions.
- 