The Versatility of TTL Caches: Service Differentiation and Pricing

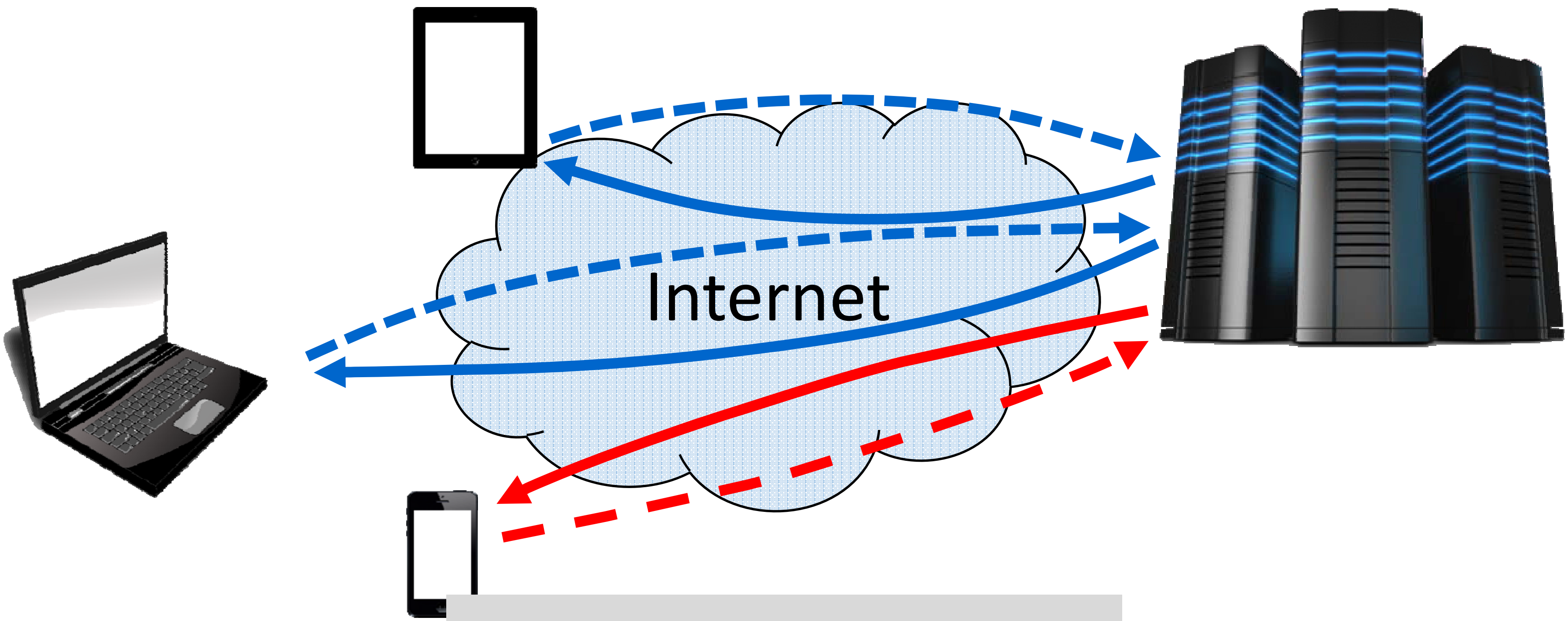
Don Towsley
CICS
Umass - Amherst

Collaborators:

W. Chu, M. Dehghan, R. Ma,
L. Massoulie, D. Menasche,
Y. C. Tay

Internet today

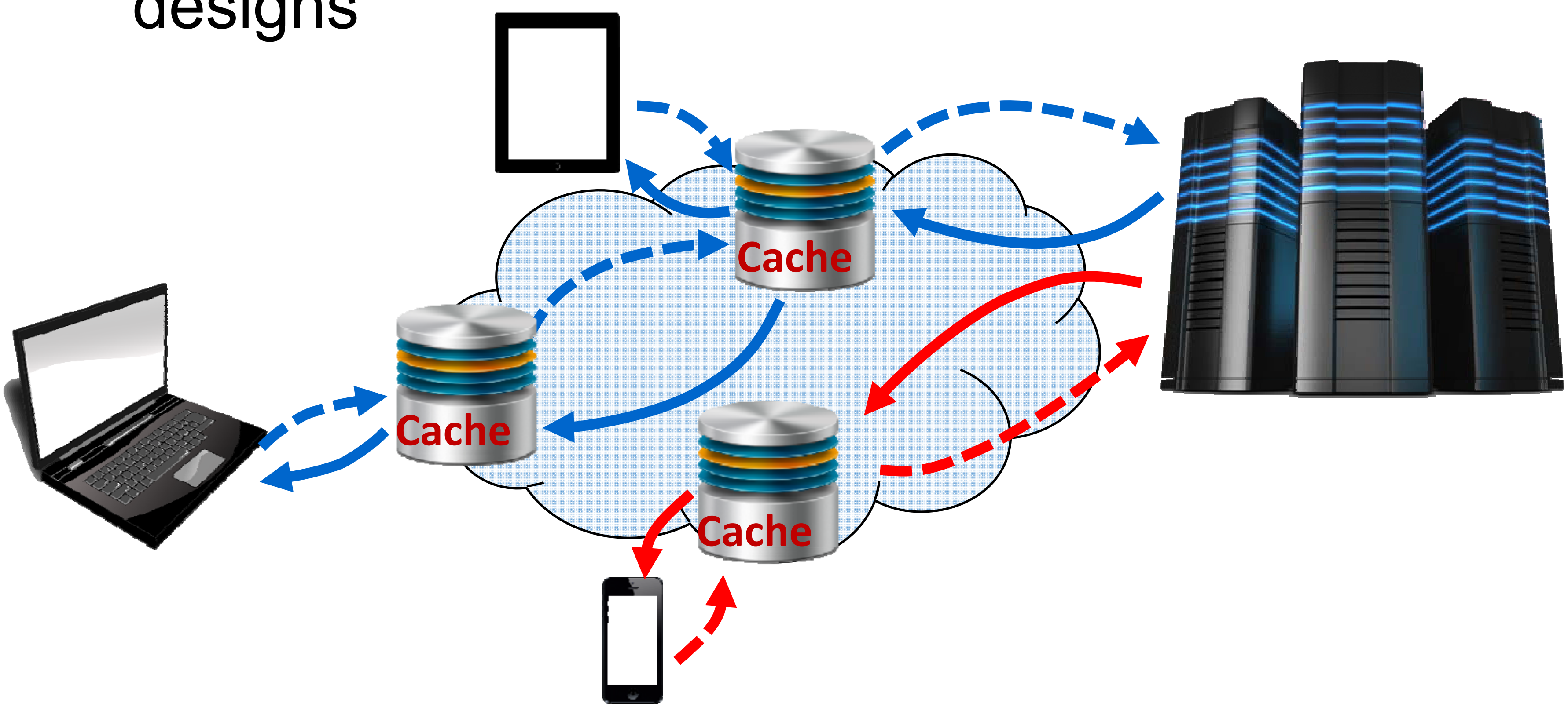
- primary use of Internet – content delivery
- point-to-point communication - users know *where* content is located



Does not scale!

New paradigm: content centric networks

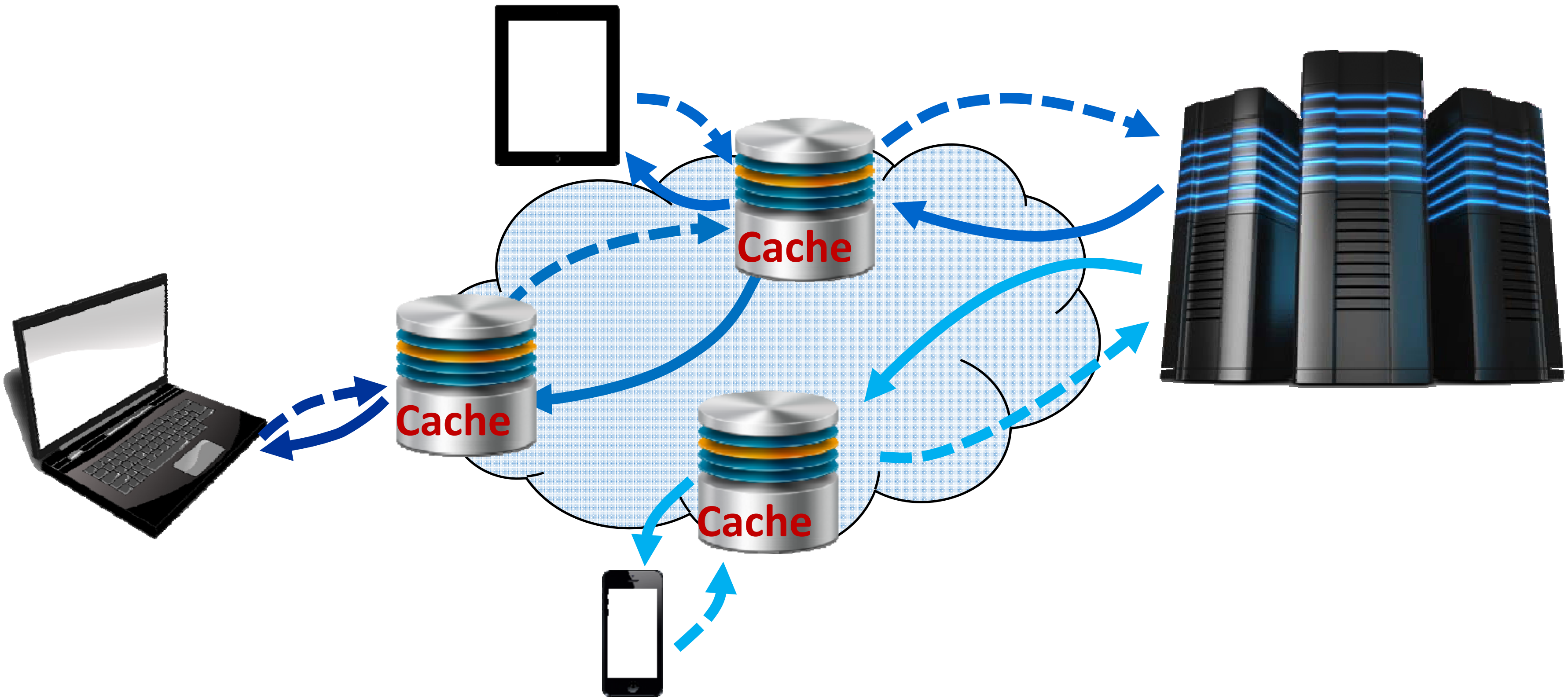
- users request what they want
- content stored at edge of network, in network
- diversity of users, content → driving CCN designs



Caching for content delivery

Decreases

- ❑ delays
- ❑ banwidth consumption
- ❑ server loads



New Challenges

Content Providers

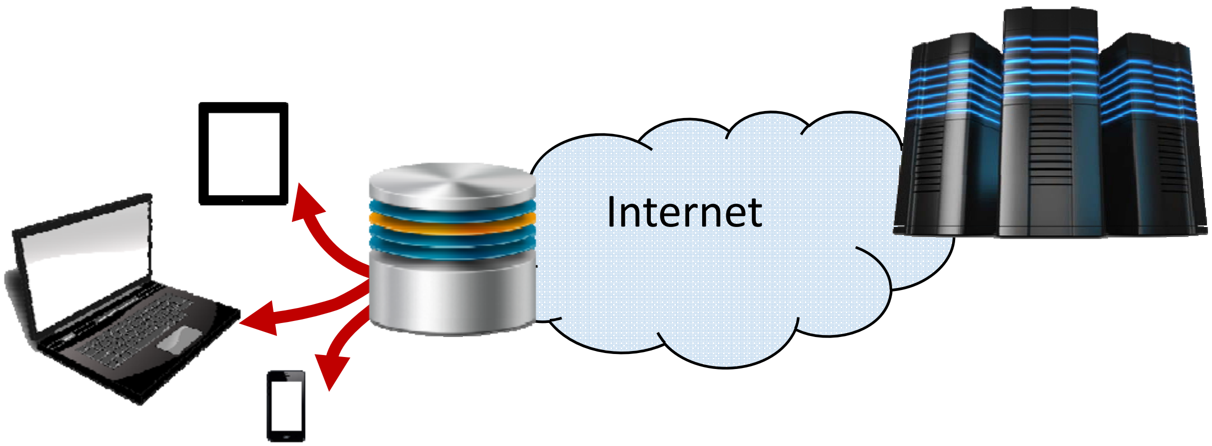
hulu
YouTube
NETFLIX
HBO GO
amazon

Content Distribution Networks

Akamai
Azion technologies
Limelight NETWORKS
fastly
Level(3) COMMUNICATIONS

Users

Service differentiation



- not all content equally important to providers/users
- content providers have different service demands
- economic incentives for CDNs
- current cache policies (mostly) oblivious to service requirements

This has given rise to alphabet soup

LACS GDSF ASA

PSC k-LRU RPAC

PRR RAND LFU DWCS

LARC LRU LFRU PBSTA

RMARK

ARC FIFO SAIU

LRV

2Q CAR SIZE LIRS

TOW DSCA

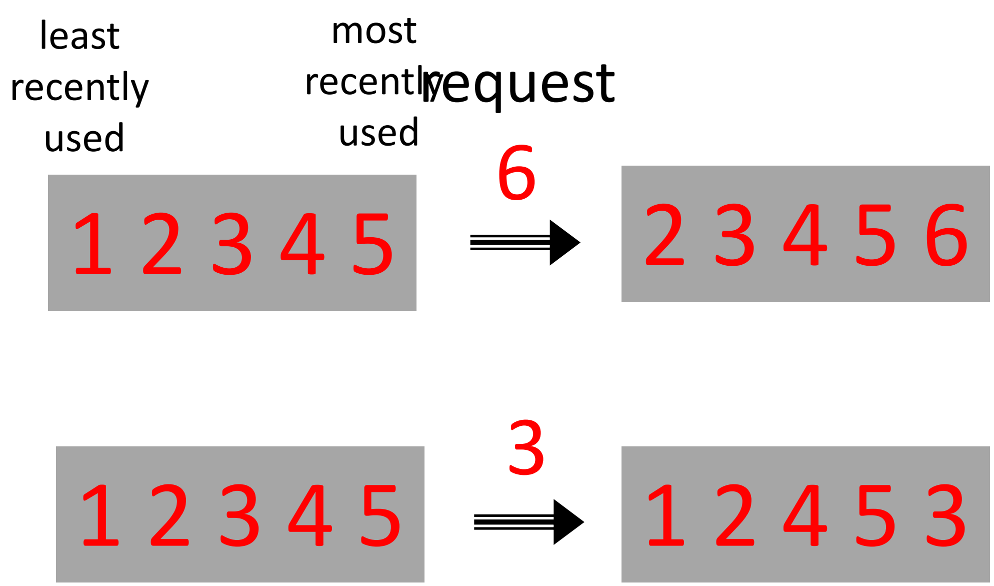
CLOCK

This has given rise to

LACS GDSF ASA
PSC k-LRU RPAC
PRR RAND LFU DWCS
LARC LRU PBSTA
LFRU
RMARK
SAIU
ARC FIFO
2Q LRV
CAR SIZE LIRS
TOW DSCA
CLOCK

LRU (least recently used)

- classic cache management policy
- contents ordered by recency of usage
- miss – remove least recently used content
- hit – move content to front



Challenges

- how to provide differentiated services
  - to users
  - to content providers
- how to make sense of universe of caches
- how to design cache policies

Answer: Time-to-Live (TTL) caches

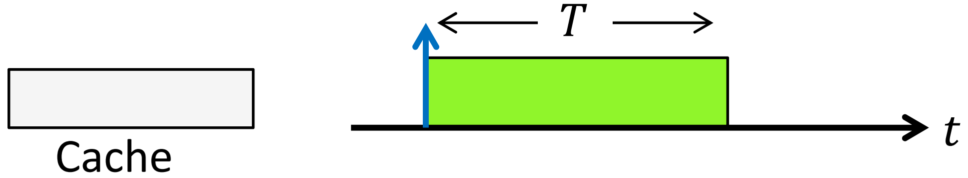
Outline

- introduction
- TTL caches
- differentiated services: utility driven caching
  - per content
  - per provider
- incentivizing caches
- conclusions, future directions

Time-to-live (TTL) caches

TTL cache

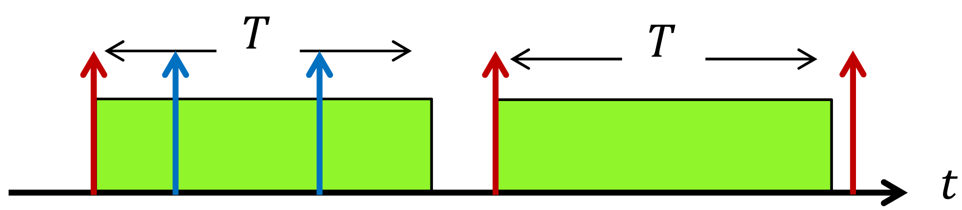
- associate timer with every content
  - set on miss
  - remove content when timer expires



- versatile tool for modeling caches
- versatile mechanism for cache design/configuration
- two types of TTL caches
  - reset, non-reset

Non-reset TTL cache

- timer set on cache miss



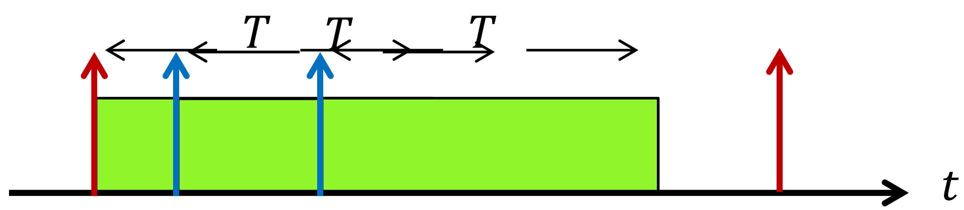
- TTL non-reset hit probability (content i):

$$h_i = 1 - \frac{1}{1 + \lambda_i T_i}$$

λ_i - request rate (Poisson)

Reset TTL cache

- timer reset at every request



- TTL reset hit probability (content i):

$$h_i = 1 - e^{-\lambda_i T_i}$$

Characteristic time approximation
(Fagin, 77)

Cache size B; request rate $\lambda_i, i = 1, \dots, N$

- LRU – model as reset TTL cache

$$\sum_i \left(1 - e^{-\lambda_i T}\right) = B$$

- FIFO – model as non-reset cache

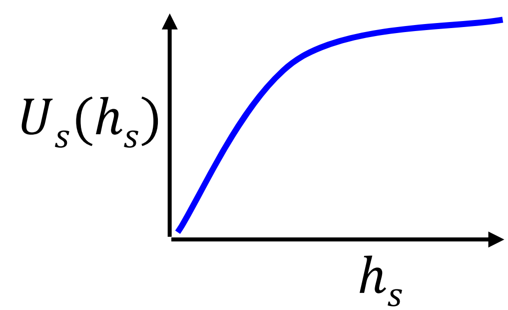
$$\sum_i \left(1 - \frac{1}{1 + \lambda_i T}\right) = B$$

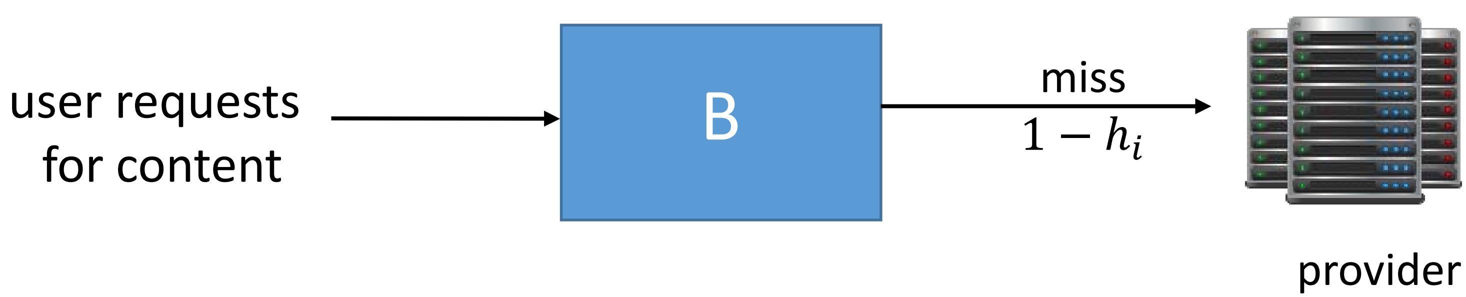
- T – *cache characteristic time*
- asymptotically exact as $N, B \to \infty$; accurate for $B > 100$
- extends to many cache policies

Providing differentiated services

Model

- single cache, size B
- N contents, request rates $\{\lambda_i\}$
- h_i: hit probability of content i
- each content has ***utility***, function of hit probability $U_i(h_i)$
  - concave, increasing





Cache utility maximization

$$\begin{aligned} \underset{h_i}{\text{maximize}} \quad & \sum_{i=1}^{N} U_i(h_i) \\ \text{such that} \quad & \sum_{i=1}^{N} h_i = B \\ & 0 \le h_i \le 1, \quad i = 1, 2, \ldots, N. \end{aligned}$$

Utility-based caching

- cost/value tradeoff
  - $U_i(h_i) = V_i(h_i) - C_i(h_i)$
- fairness implications
  - e.g. Proportionally fair w.r.t. hit probability
- cache markets
  - contract design

Cache utility maximization

$$
\begin{aligned}
\underset{h_i}{\text{maximize}} \quad & \sum_{i=1}^{N} U_i(h_i) \\
\text{such that} \quad & \sum_{i=1}^{N} h_i = B \\
& 0 \le h_i \le 1, \quad i = 1, 2, \ldots, N.
\end{aligned}
$$

Can we use this framework to model existing policies?

Reverse Engineering

Can we obtain same statistical behavior as LRU, FIFO using timers? What utilities?

LRU

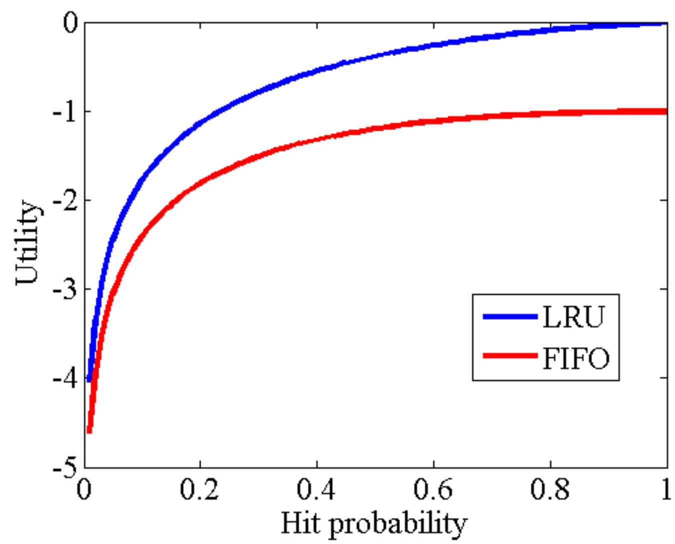
$$U_i(h_i) = \lambda_i \mathrm{li}(1 - h_i)$$

$$\mathrm{li}(x) = \int_0^x \frac{\mathrm{d}t}{\ln t}$$

logarithmic integral

FIFO

$$U_i(h_i) = \lambda_i(\log h_i - h_i)$$



Dual Problem

- Lagrangian function:

$$L(\boldsymbol{h}, \alpha) = \sum_{i=1}^{N} U_i(h_i) - \alpha\left[\sum_{i=1}^{N} h_i - B\right]$$

Dual variable

- optimality condition:

$$\frac{\partial L}{\partial h_i} = U_i'(h_i) = \alpha$$

- inverse

$$\boldsymbol{h_i} = \boldsymbol{U_i}'^{-1}(\boldsymbol{\alpha})$$

LRU Utility Function

- optimality condition:

$$h_i = {U_i}'^{-1}(\alpha)$$

- TTL approximation

$$h_i = 1 - e^{-\lambda_i T}$$

- let hit probability decrease in α, increase in T
- let $T = 1/\alpha$

$$U_i(h_i) = \lambda_i \mathrm{li}(1 - h_i)$$

Fairness properties

- weighted proportional fairness

$$U_i(h_i) = \lambda_i \log h_i$$

yields $h_i \propto \lambda_i$

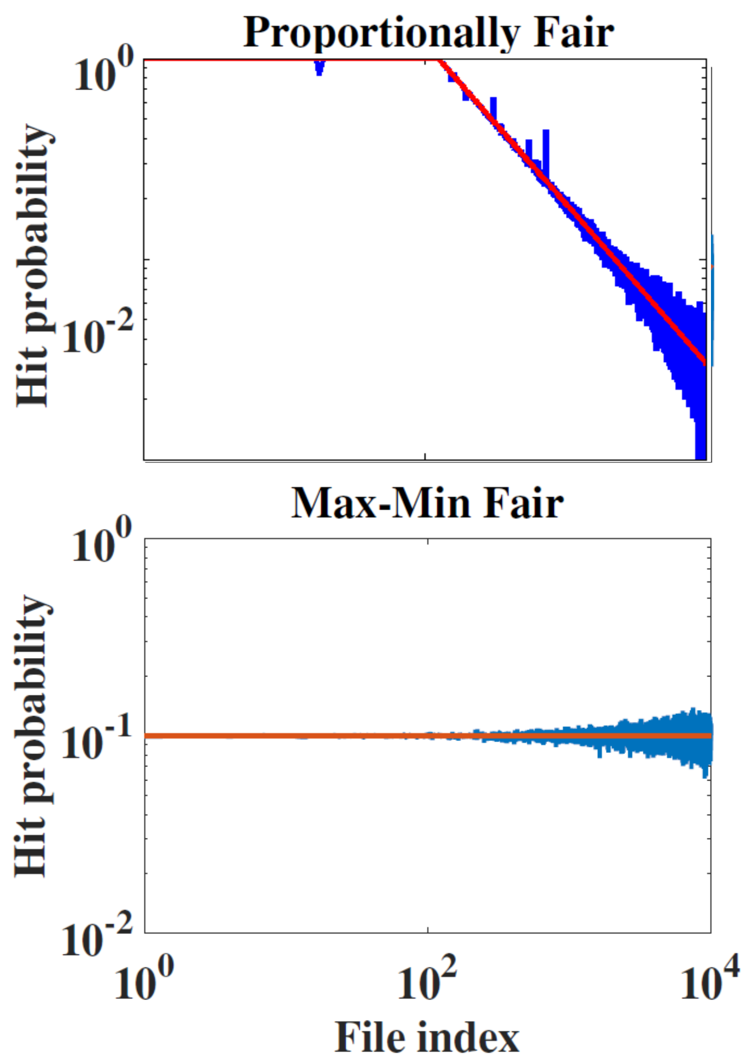
- max-min fairness – limit as $\beta \to \infty$

$$U_i(h_i) = \lim_{\beta \to \infty} \frac{h_i^{1-\beta}}{1-\beta}$$

yields $h_i = B/N$

Evaluation

- 10,000 contents
- cache size 1000
- Zipf popularity, parameter 0.8
- 10^7 requests



Cache utility maximization

$$\begin{aligned} \underset{h_i}{\text{maximize}} \quad & \sum_{i=1}^{N} U_i(h_i) \\ \text{such that} \quad & \sum_{i=1}^{N} h_i = B \\ & 0 \le h_i \le 1, \quad i = 1, 2, \ldots, N. \end{aligned}$$

Q: How do we control hit probabilities?

A: TTL cache; control hit probabilities through timers

Cache utility maximization

$$\max_{T_i} \sum_{i=1}^{N} U_i(h_i(T_i))$$

$$s.t. \sum_{i}^{N} h_i(T_i) \leq B$$

On-line algorithms

- dual algorithm
- primal algorithm
- primal-dual algorithm

Setting timer in dual

- TTL-reset cache:

$$\boldsymbol{h_i = 1 - e^{-\lambda_i T_i}}$$

- optimality condition:

$$\boldsymbol{h_i = {U_i}'^{-1}(\alpha)}$$

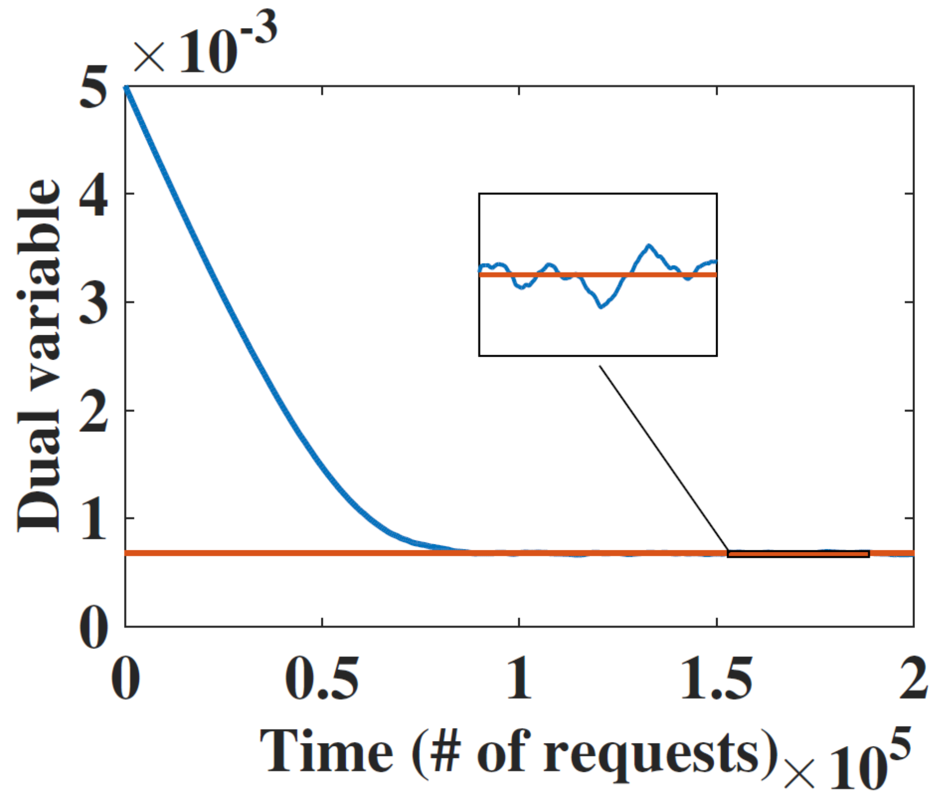
$$\Rightarrow$$

$$\boldsymbol{T_i} = -\frac{\boldsymbol{1}}{\boldsymbol{\lambda_i}}\log\left(1 - \boldsymbol{{U_i}'^{-1}(\alpha)}\right)$$

- find α via gradient descent; update at each request
- estimate $1/\lambda_i$ using sliding window

Convergence: dual algorithm

- 10,000 contents
- cache size 1000
- Zipf popularity, parameter 0.8
- 10^7 requests



Primal algorithm

- primal problem replaces buffer constraint with soft "cost" constraint

$$\max_{T_i} \sum_{i=1}^{N} U_i(T_i) - C\left(\sum_{i=1}^{N} h_i(T_i) - B\right)$$

with convex cost function C

- similar style on-line algorithm

Summary

- utility-based caching enables differentiated services
- TTL cache provides flexible mechanism for deploying differentiated services
- simple online algorithms require no apriori information about:
  - number of contents
  - popularity
- framework captures existing policies
  - e.g. LRU and FIFO

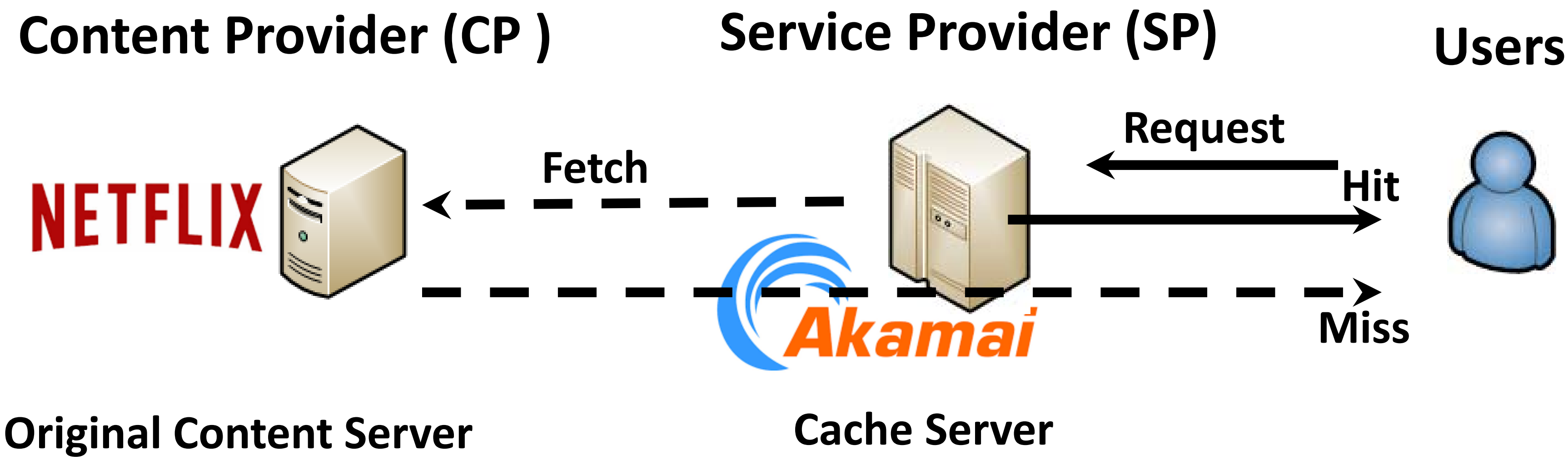
Other issues

- provider-based service differentiation
- monetizing caching

Differentiated monetization of content

- focused on
  - user/content differentiation
  - CP differentiation
- how can SPs make money?
  - contract structure?
  - effect of popularity?

Per request cost and benefit



- benefit per request hit b
- cost per request miss c

Key: how should SP manage cache?

Formulating as utility optimization

$$U(h) = \lambda b h \; - \lambda c(1-h) - P(h)$$

$P(h)$ - payment to cache provider

Q: how should SP manage cache? pricing schemes?

Service contracts

Contracts specify pricing $P(h)$ per content

- nonrenewable contracts
- renewable contracts
  - occupancy-based
  - usage-based

Non-renewable contracts

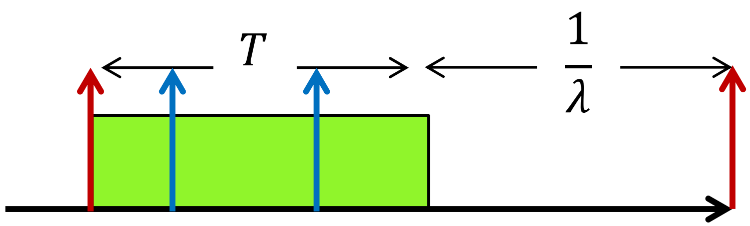
- on-demand contract upon cache miss
  - no content pre-fetching
  - contract for time T linear price pT
    - proportional to TTL, per-unit time charge p
- potential inefficiency
  - content evicted upon TTL timer expiration

  $\Longrightarrow$ miss for subsequent request
- how long to cache content?

Non-renewable contracts

- value accrual rate to content provider

$$U = \lambda \frac{b\lambda T + c}{\lambda T + 1}$$



- payment rate to cache provider

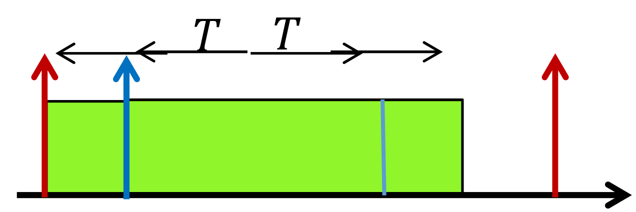
$$P = p \frac{T}{T + 1/\lambda}$$

Rule: cache if $\lambda(b + c) > p$; $T^* = \infty$

otherwise not

Occupancy-based renewable contracts

- on-demand contract on every cache request
  - o pre-fetching
  - o at request, pay
    - pT if miss
    - px, if time since last request $x < T$



- CP pays for time content in cache

Rule: cache if $\lambda(b + c) > p$; $T^* = \infty$

otherwise not

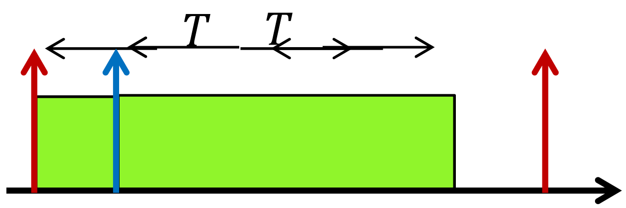
same as non-renewable contract

Observations

- both contracts occupancy based; pay for time in cache
- renewable contract more flexible
    - allows contract renegotiation
- results generalize to renewal request processes

Usage-based renewable contracts

- on-demand contract on every cache request
  - o no pre-fetching
  - o at request, always pay pT



- price - pT per request

Rule: cache if $\lambda(b+c) > p$

$$T^* = \frac{1}{\lambda} \ln \frac{\lambda(b+c)}{p}$$

otherwise not

Observations

Usage-based pricing

- provides better cache utilization than occupancy-based pricing
  - T^* decreasing function of λ, p; increasing function of b, c
- better incentivizes cache provider

Summary

- TTL cache versatile construct for
  - modeling/analysis
  - design/configuration
  - adaptive control
  - pricing
- TTL combined with utility-based optimization
  - provides differentiated cache services
  - shares caches between content providers
  - provides incentives for cache providers

Future directions

- differentiated services in a multi-cache setting
  - presence of router caches
  - multiple edge caches
- relaxation of assumptions
  - Poisson, renewal → stationary
  - arbitrary size content
- pricing
  - non-linear pricing
  - market competition among cache providers
- unified cache, bandwidth, processor allocation framework

Thank you