

# A Task-Based Adaptive-TTL approach for Web Server Load Balancing\*

**Devarshi Chatterjee**

**Zahir Tari**

RMIT University  
School of Computer Science and IT  
Melbourne, Australia  
*[zahirt@cs.rmit.edu.au](mailto:zahirt@cs.rmit.edu.au)*

\* Supported by ARC (ARC Discovery DP 0346545) and SUN Microsystems

# Overview

- **Motivation**
- **Technical Issues**
- **Basics & Limitations of LAN/WAN based approaches**
- **Our approach**
  - WAN-level (with HLW and server classification)
  - LAN-level (with re-direction and queue estimation)
- **Testing results**

# Why Load Balancing?

## ■ Benefits

- Improve resource utilization
- Improve system performance
- Adding extra devices costly

## ■ Architecture

- Independent mirrored-servers
  - Manually selected by users
  - Not scalable as they provide a non-transparent location binding
- Distributed web servers
  - Appear as a single host.
  - Authoritative DNS (A-DNS)
    - Allow to control binding within a cluster, based on specific load balancing policies and information (such as TTL).
    - Bottleneck at the A-DNS (with all the classical problems).

# Load Characteristics

- **Empirical measurements [Crovella 1998] show many short tasks, and fewer large tasks.**
- **E.g. 1998 World Soccer Cup [M. Arlitt et al., HP T.R. 1999]**
  - Traffic measurements
    - 1.3 billions of requests in 3 month, with 90% static HTML, 2% dynamic requests. 0,08% of requests counts for 20% of data.
    - *Request distribution is heavy-tail (i.e.  $P[X>x] \approx x^{-\alpha}$ )*
- **Heavy-tailed distribution (e.g. *Pareto distribution*)**
  - Decreasing failure rate (i.e. the longer it runs, the longer to continue)
  - Infinite variance
  - Few (less than 1%) very large tasks take up more than 50% of the total task sizes

# Task Distribution

- Heavy-tailed

$$\Pr\{X > x\} \sim x^{-\alpha}, \quad 0 < \alpha < 2$$

$\alpha \approx 2$  (moderate variability)

$\alpha \approx 1$  (high variability)

- Simplest Heavy-tailed (**Pareto** distribution)

$$\Pr\{X \leq x\} = 1 - (k/x)^\alpha$$

- Examples

- Process CPU consumption in Unix system,  $\alpha \approx 1$  [Harchol 1997].

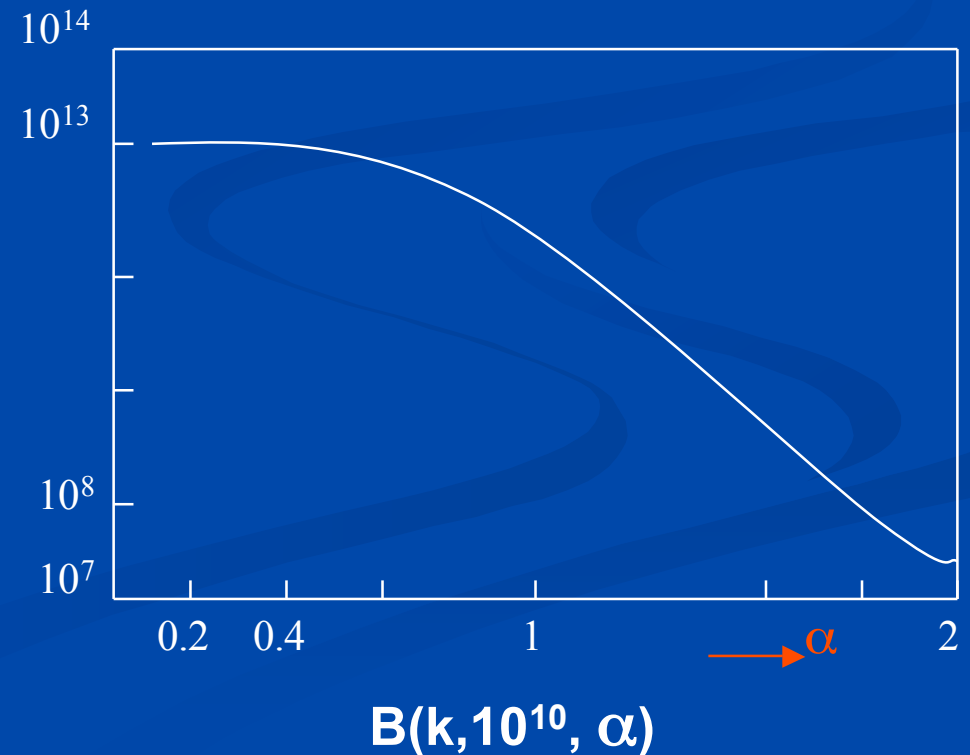
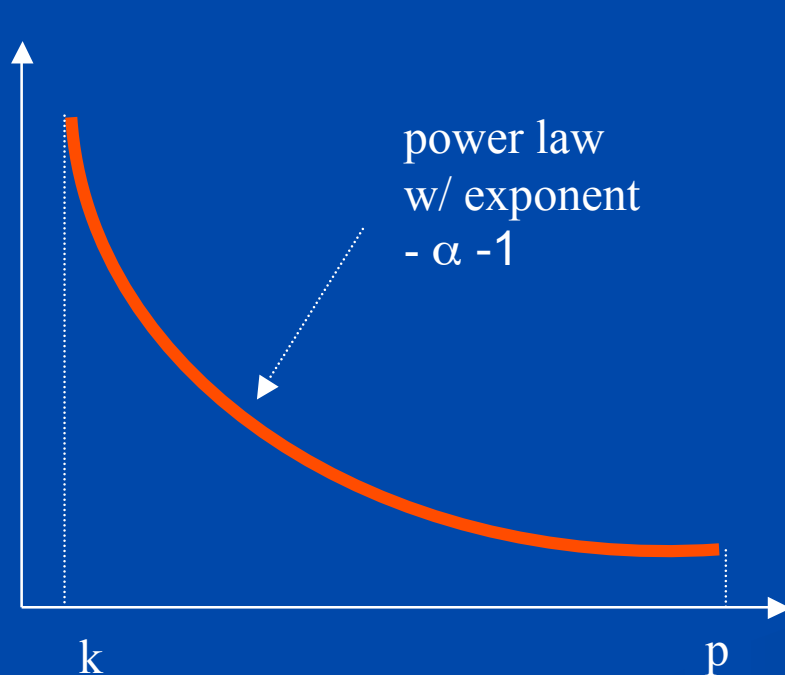
- ✓ File transferred via HTTP,  $1.1 \leq \alpha \leq 1.3$  [Crovella 1997].

- ✓ File transferred via FTP,  $0.9 \leq \alpha \leq 1.1$  [Paxson 1995].

# Bounded Pareto distribution $B(k,p,\alpha)$

$$f(x) = (\alpha k^\alpha / (1-(k/p)^\alpha)) x^{-\alpha-1} \quad k \leq x \leq p$$

$k$ , the shortest possible job  
 $p$ , the largest possible job



# Technical problems

## ■ Estimation of the load

- [LAN] Periodic updates, baysian models, stale-based approaches [Dahlin 2002]
- [WAN] Difficult to do as DNS controls only a very *tiny* amount of requests. Server-based feedback crucial!

## ■ Task assignment

- [LAN] “request size distribution”, arrival rate, known/unknown size, divisible/indivisible task
- [WAN] “request size distribution”, TTL=0 problem, client/server information

## ■ Task migration/re-direction

# Metrics

## ■ LAN

WaitingTime( $T_k$ )

FlowTime ( $T_k$ ) = WaitingTime( $T_k$ ) +  
ProcessingTime( $T_k$ )

SlowDown ( $T_k$ ) = WaitingTime( $T_k$ )/TaskSize( $T_k$ )

## ■ WAN

*Cumulative Function (CF)*

⇒ maximum cluster utilization (e.g. the probability that the maximum cluster utilization is below a certain threshold).

# WAN-based Approaches

## ■ Facts

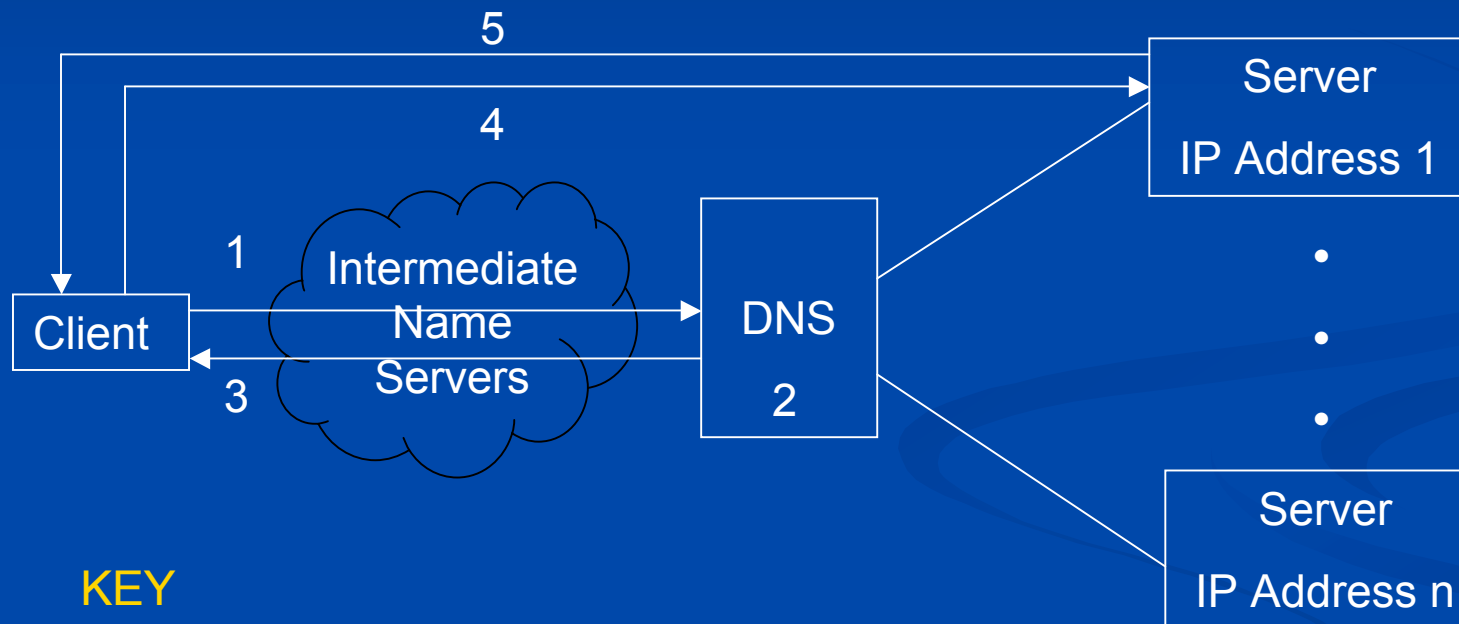
- De-centralized model (for routing requests)
- Caching at I-DNS (bypassing A-DNS)

## ■ Depends on the focus

- **Client**: uses Hidden Load Weight (HLW) [Colajanni et al., IEEE Computer 1999].
- **Server**: similar to LAN-based, however less advance (e.g. sending load asynchronously [Yu et al., IEEE TPDS 1998]).
- **DNS**: uses A-DNS as a dispatcher [Colajanni et al., ACM Computing Survey 2002]. Much more scalable and efficient for geographically distributed web servers!

**Best solution is the INTEGRATION of the first two approaches within DNS-based approach.**

# DNS-based Architecture



## KEY

1. Address request
2. Web Server selection (Address 1)
3. Address Mapping (URL → Address 1)
4. Document Request (Address 1)
5. Document Response (Address 1)

# DNS-based Load Balancing

## ■ TTL (Time To Live)

- Period for which a resource record (containing name to address translation) is cached.
- When it expires a new translation is sought.

## ■ Algorithms

### ■ Static

- All requests have the same TTL. High risk overloading servers.
- #types: stateless, client or sever based, combined

### ■ Dynamic

- Varies depending on the requests, domains etc.

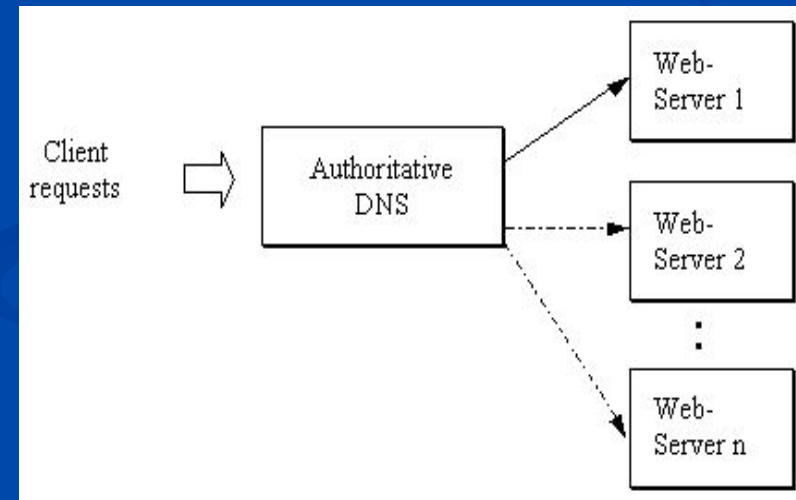
# Stateless

## ■ Various

- **Random** [Colajanni et al., Internet Computing, 1999]
- **RR** [Internet Software Consortium BIND]: BIND finds the next IP from a list of associated IP addresses.

## ■ Limitations

- Does not take into server load and server availability
- Only deals with a small fractions of requests
- Ignores high variability in client's requests (i.e. small # of very large tasks)



# Static based Approaches

## ■ Client-based

- Geographical location
  - E.g. topological proximity [CISCO DistributedDirectory, 1997], client-server link latency, round trip delays [Beck et al., WWW 1998]
- # of requests, using *Hidden Load Weight (HLW)*
  - **Basic:** RR2 [Yu et al., IEEE TPDS 1998] maintains two lists of server pools to balance
  - **Load-based:** DAL (Dynamic Accumulated Load) [Cardellini et al., COMPSAC 1998], where A-DNS
    - Kind of *Load Index* based on HLW
    - *bin@server* (as cumulated HLW from various domains)
    - *lowest bin*, and increases the value by the HLW of the requests

# Static based Approaches

## ■ Server-based

- Compute load (in a certain way?!?)
- Send it *synchronously* (i.e. periodically, approx 8 or 16s) or *asynchronously* (with 2 alarms) to A-DNS, excluding overloaded servers.
- Requests routed in *LLF-like* at the same frequency the load is sent to A-DNS
- E.g. *lbmnamed* [Schemers, LISA 1995], however is limited as sets TTL to zero to avoid caching.

## ■ Client & Server based

- Combination of HLW and alarms from overload servers
- E.g. CISCO's **DistributedDirectory** uses *proximity* with *server availability*.

# Dynamic based Approaches

## ■ Idea

- Variability of TTL based on domain popularity
- Results in low TTL for clients making frequent queries and vice versa [Colajanni et al., WWW 1999]

## ■ Limitations

- I-DNS may cache name-address mapping
- TTL=0 problem (I-DNS discard it, A-DNS bottleneck)
- Task size is not considered (Pareto distribution)
- TTL does not work on web browser caching

# Proposed Model

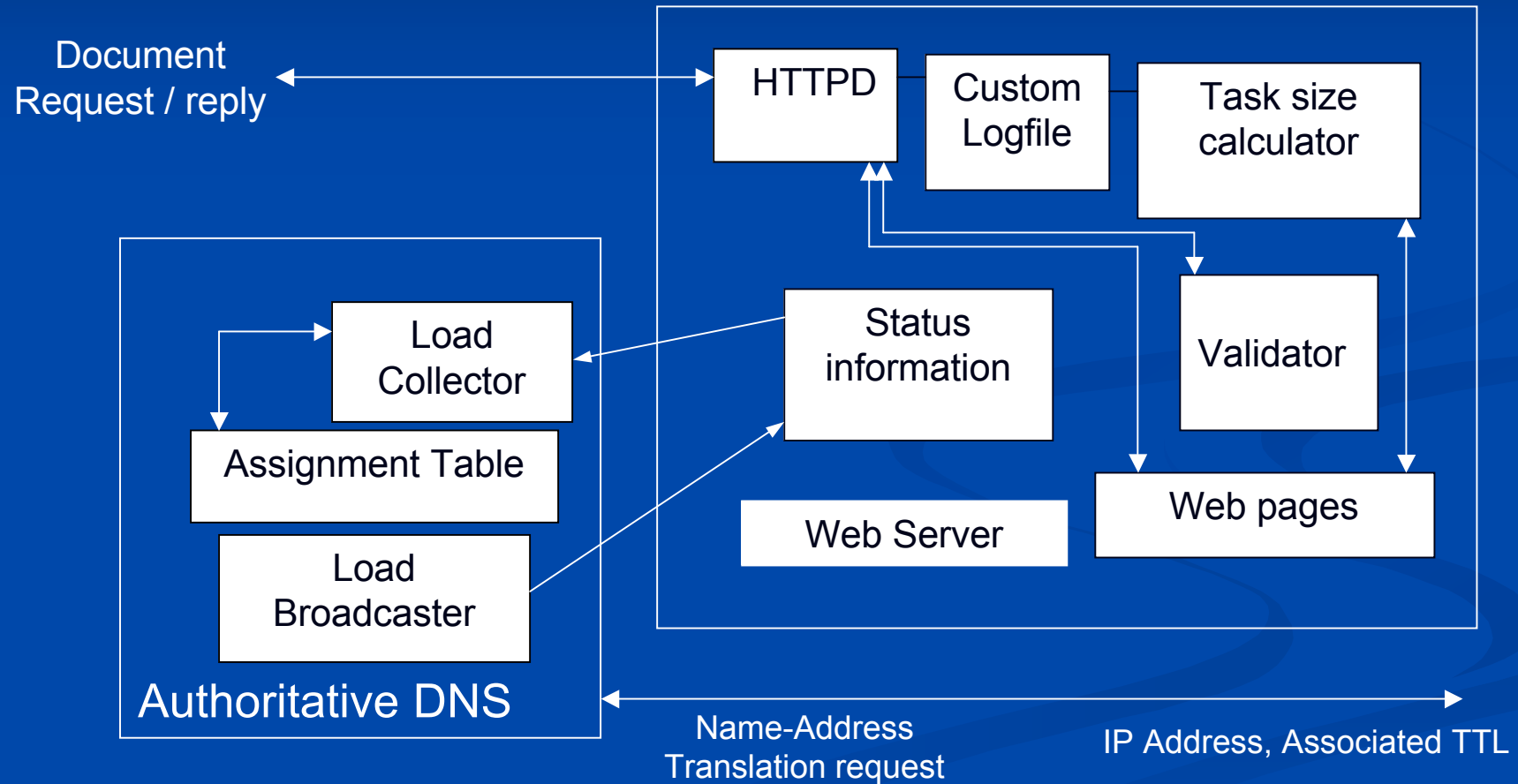
## ■ Principle

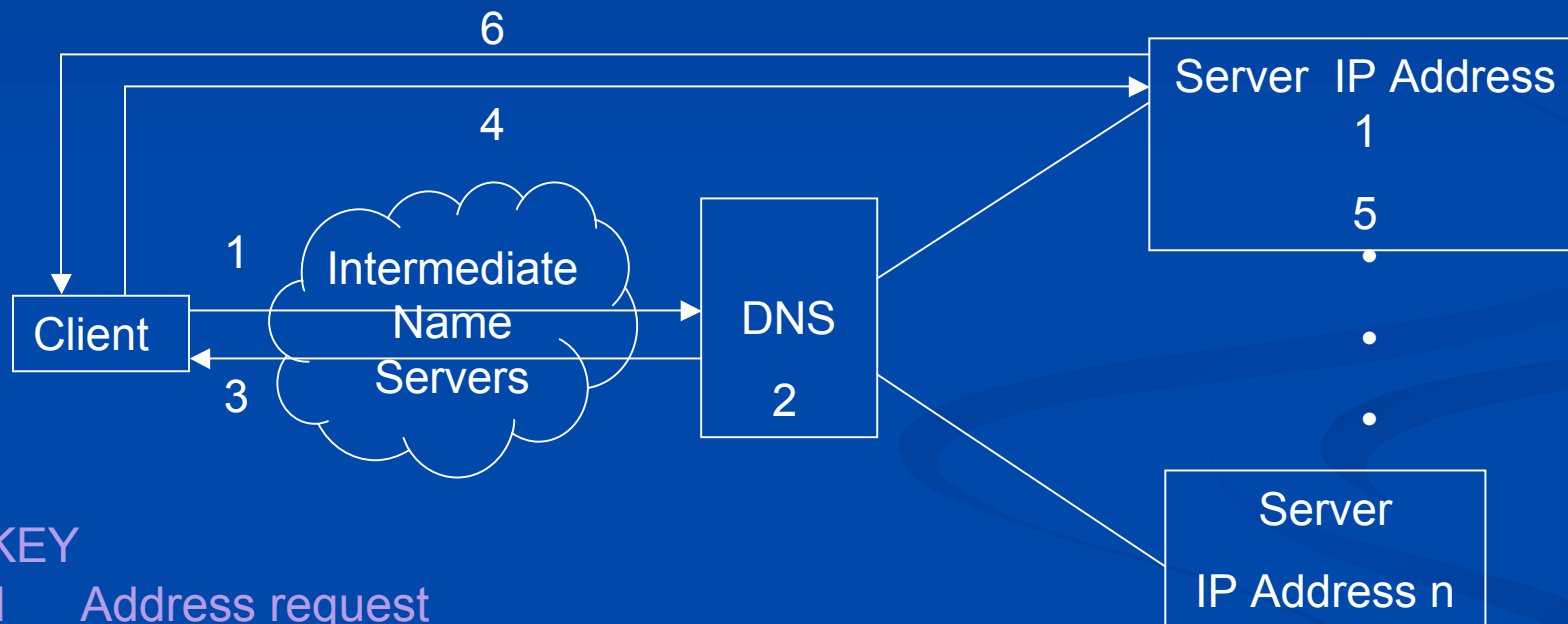
- Take into account client's parameters
  - Dynamic TTL assigned based on domain popularity
- Maximum time taken to download a document
- Processing capacity of servers
- HTTP redirection used as server side load balancing

## ■ Implemented

- Assignment Table containing server' info (computation of task size, Web server processing capability, Web server utilization)
- Updated every 15 seconds

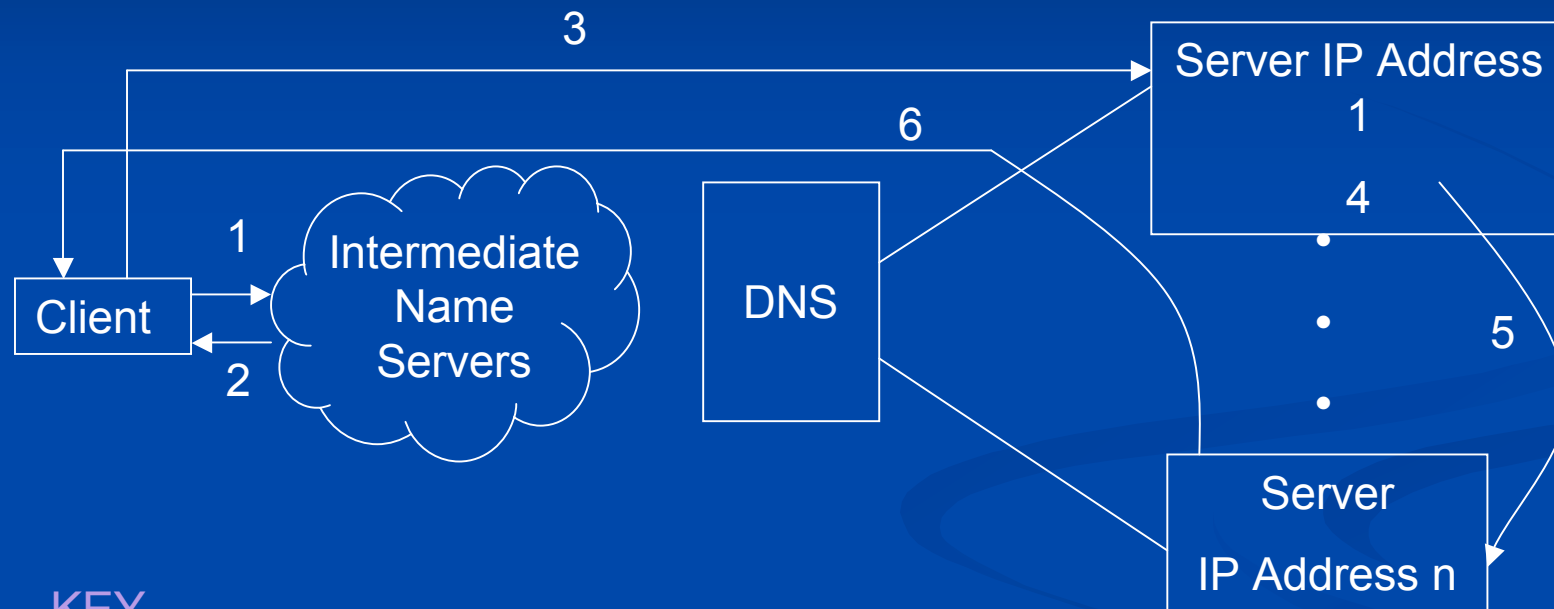
# Architecture





### KEY

- 1 Address request
- 2 Web Server selection (Address 1)
- 3 Address Mapping (URL => Address 1)
- 4 Document Request (Address 1)
- 5 Server side check (cached?)
- 6 Document Response (Address 1)



### KEY

1. Address request (Intermediate Name Server)
2. Address Mapping (URL => Address 1)
3. Document Request (Cached - Address 1)
4. Web Server Selection
5. HTTP-Redirectation
6. Document Response (Address 1)

# Testing

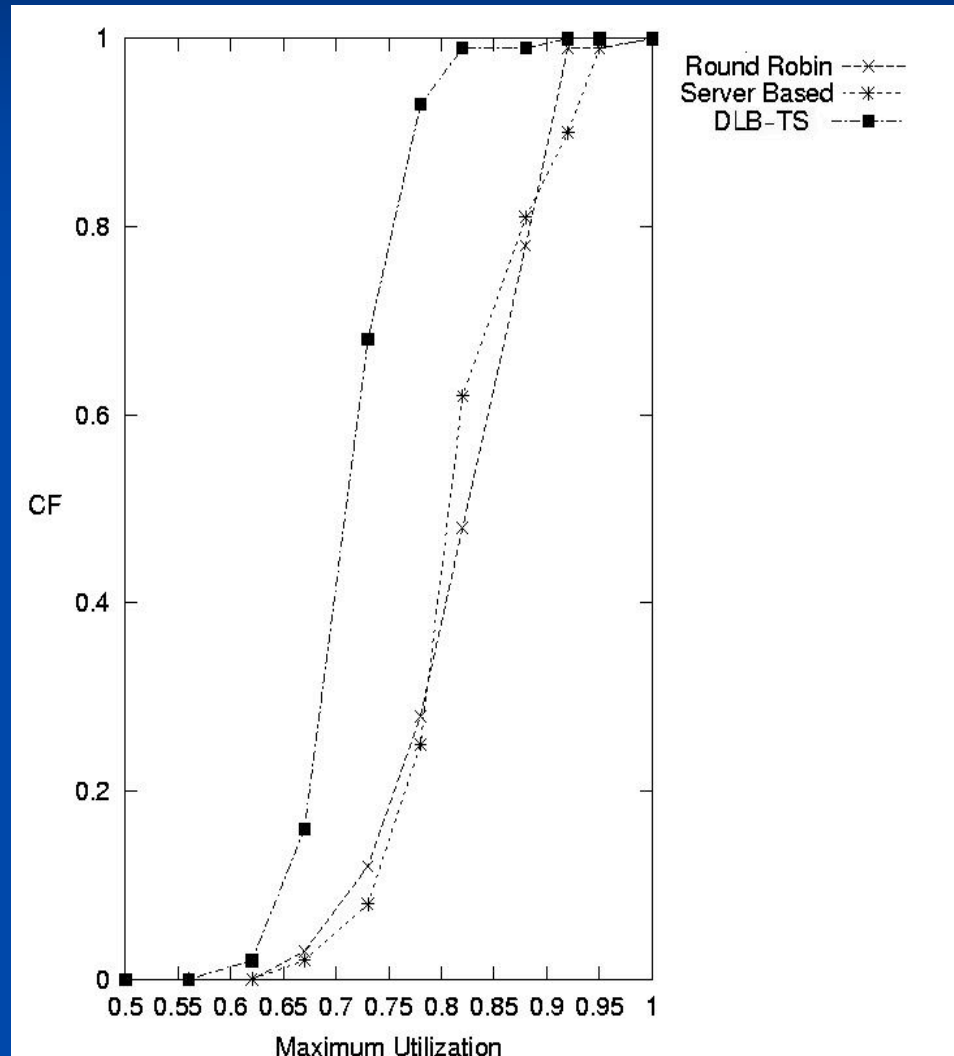
## ■ Various

- TTL variation
- Response time
- Utilization of the cluster
- Sensitivity of update interval to Assignment table

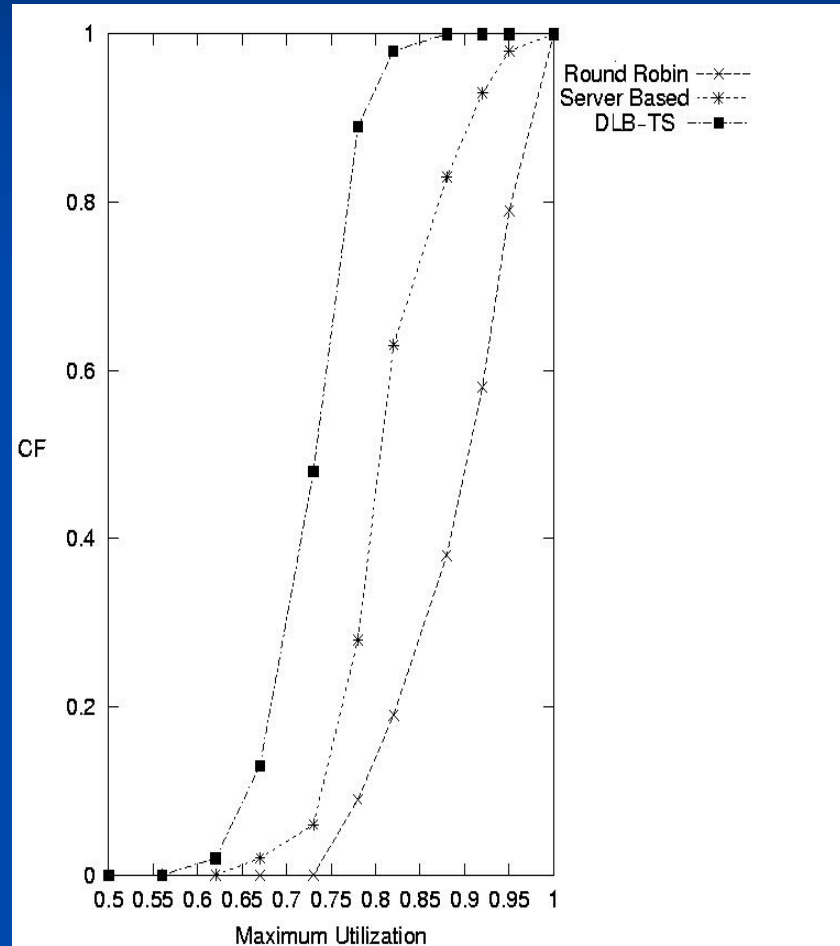
## ■ Effect of TTL

- TTL set to 15s, 0.99 probability of not overloading the cluster
- Cumulative function (e.g. probability of the maximum utilization is below a threshold)  
[Colajanni et al., WWW 1999]

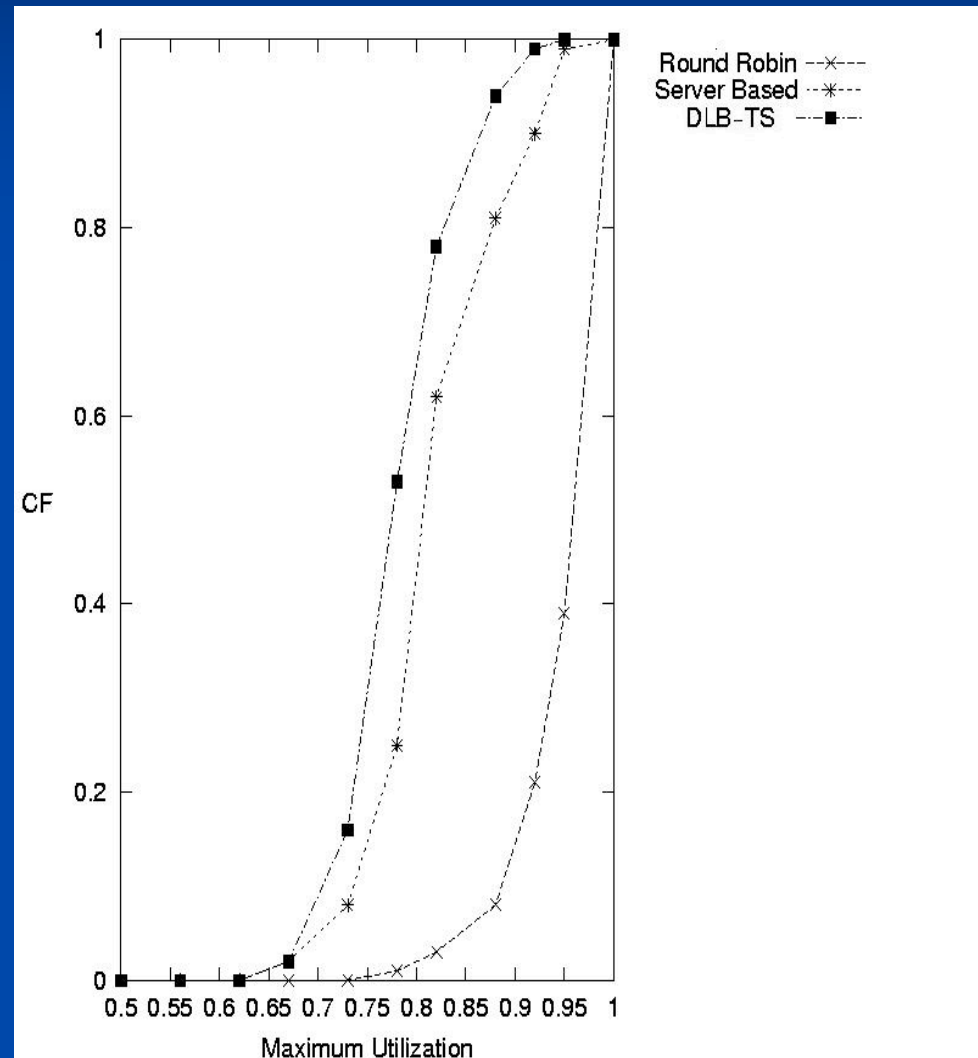
# CF with TTL = 15s



# CF with TTL = 60s

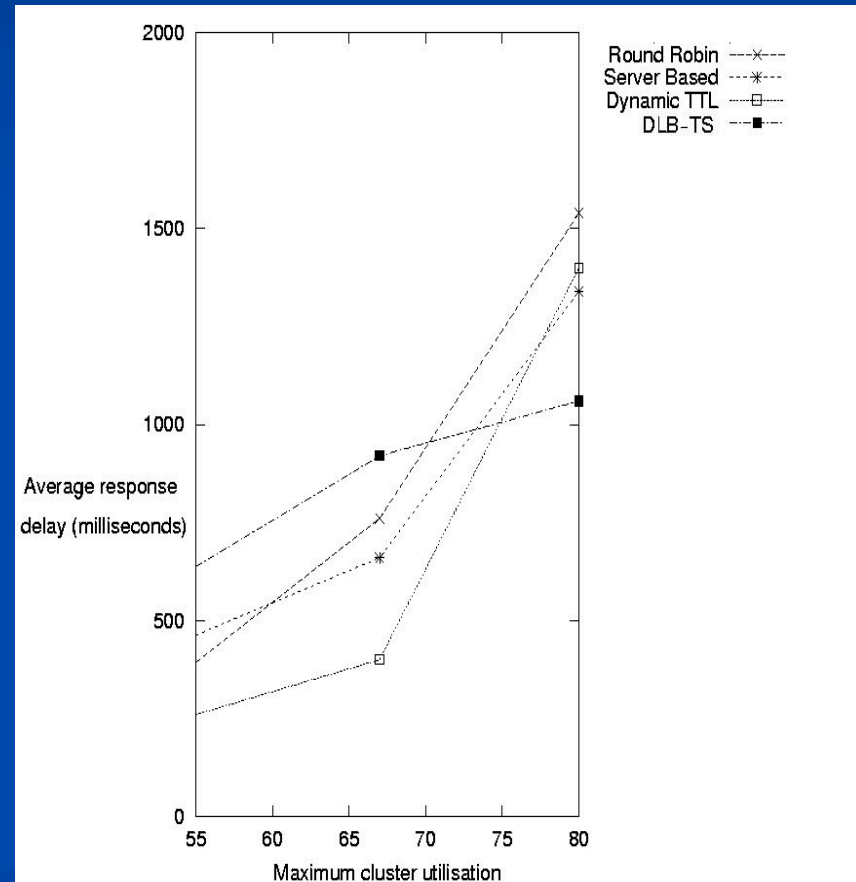


# CF with TTL = 120s



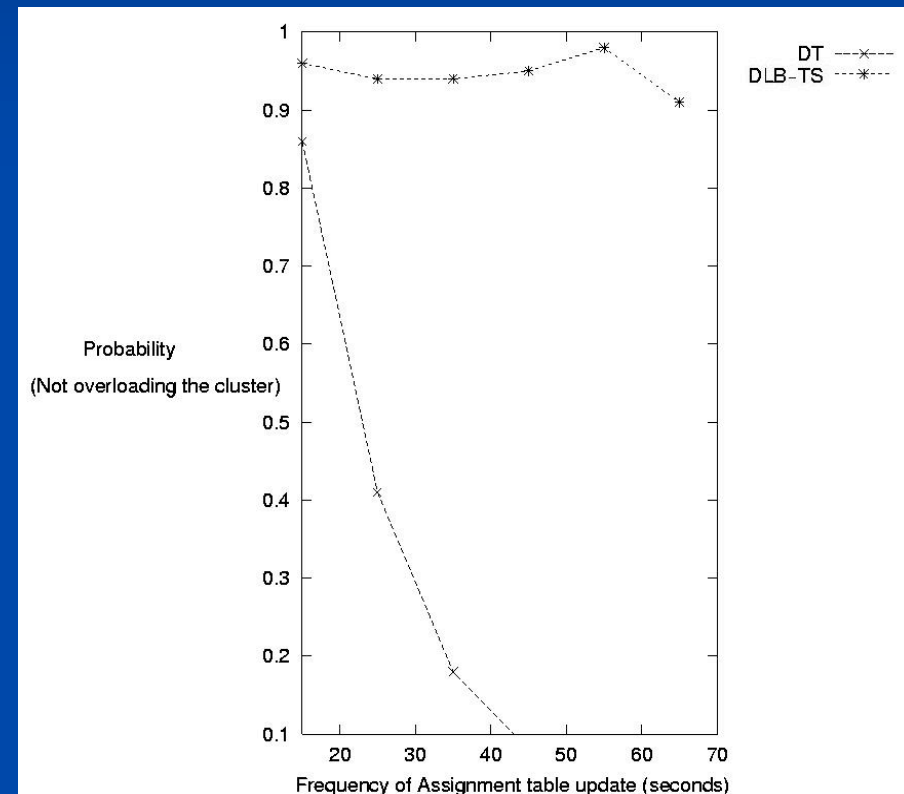
# Client Perceived Delay

- High initial response time, better than the rest at 75% of cluster utilization
- 16% improvement over the rest at 80% cluster utilization



# Sensitivity of Update Interval to Assignment Table

- Not highly dependent like dynamic TTL (DT)
- A high (> 60 seconds) interval caused independence to be affected



# Conclusion

- **Client-server algorithms have better performance**
- **DLB-TS insensitive to the change of TTL, whereas Adaptive-TTL fails at high TTL**
- **DLB-TS lowers the client's perceived delay under high load**