Modeling Web Caching Hierarchy Schemes

Cho-Yu Chiang, Mikihiro Ueno[1], Ming T. Liu, Mervin E. Muller

Department of Computer and Information Science

The Ohio State University

Columbus, Ohio 43210-1277

E-mail: {chiang, ueno, liu, muller-m}@cis.ohio-state.edu

July 3rd, 1999

Abstract -- A mathematical model used to estimate the response times for different Web caching hierarchy schemes is discussed in this paper. Four caching hierarchy schemes -- simple hierarchy, ICP, Cache Digest, and Caching Neighborhood Protocol (CNP) -- were modeled for simulation study. Under some reasonable assumptions, CNP was found to outperform the other approaches in terms of providing the shortest response time for using the target document of size 2K bytes. Such a target was chosen because it approximated the median size for the requested Web documents[17][27] . We also discuss issues relating to the response times for the Web caching infrastructure and why CNP reduces the response times when compared to the other Web caching schemes.

---

1. The author is affliated with the Science and Technical Research Laboratories at NHK (Japan Broadcasting Inc.). This report is done while the author is a visiting scholar at the Ohio State University.

I. Introduction

A. Background

Since the introduction to proxy servers (in short, proxies) [4][12][20], Web caching [14] has been generally considered as an effective approach to alleviate problems such as slow response times and network congestions by increasing the document availability and enabling download sharings among Web clients. There were some research works focusing on building hierarchies of proxy servers to enhance the performance of Web caching. Among the most notables, the pioneer approach was CERN's [5] proxy servers whose infrastructure was based on the simple hierarchy. It was followed by *Harvest* [2] in which the *Internet Cache Protocol* (*ICP*) [7] was introduced to organize the proxy servers and achieve better inter-proxy cooperation. After *Harvest* was commercialized, its public-domain successor, *Squid* [21], made several improvements on *ICP.* The most noteworthy one was *Cache Digest* (*CD*) [18] that utilized bloom filters [19] to enhance the performance of *ICP.* A similar idea, *Summary Cache*, was also presented in [23]. All the approaches mentioned above can be classified as *client-oriented* because the caching behaviors of proxy servers are generally determined by the requests of the clients with little instruction from the origin servers. Besides, these hierarchical approaches could also be deemed *static* as opposed to *dynamic* where the sets of proxy servers that collaboratively handle the requests may change for every single request and thus the request handling paths may vary.

In [1], we presented a caching hierarchy scheme of dynamic nature based on the proposed *Caching Neighborhood Protocol* (*CNP*). By incorporating ideas from [8][10][11][15][22] along with observations from the proxy trace logs [16], we derived a promising new approach. Such an approach is regarded *server-oriented* because the caching behaviors of proxy servers are primarily influenced by the origin servers. The major goals of proposing *CNP* are to *(i)* deal with the cache coherency problem [3][6][9] existing in the client-oriented caching hierarchy schemes, *and (ii)* reduce the network traffic and response times for document requests. It was elaborated in [1] why both goals were not too ambitious to achieve for the *CNP* approach.

B. Aim and Approach of the Research

In this paper, we focus on the issues regarding the response times in the aforementioned goal *(ii)*.

The achievement of goal *(i)* is discussed in [1][24]. Intuitively Web caching could reduce the network traffic via the download sharings. Nevertheless, it is not clear if Web caching also reduces the response times due to the overheads incurred in the processings of the proxy servers in the caching hierarchy. To set up a model for the study, we analyzed and decomposed the Web traffic to individual components inspired by the work of Rousskov [17]. The model was employed to study four different hierarchical approaches -- simple hierarchy without using any inter-proxy communication protocol, *ICP*, *CD*, and *CNP*. As a result of the modeling, simulation experiments were performed to enable us to understand the factors relating to the response times for different Web caching hierarchy schemes. Also, we were able to compare performances for the different approaches.

C. Contributions

Our research contributions are presented in the following aspects: *(i)* To understand the effect of establishing hierarchies for the Web proxy servers. A number of approaches were modeled in which major components that affect the response times were identified. The purpose was to approximate the real networking behaviors of the proxy servers and examine the components of the response times for each approach so as to facilitate discussing the deployment of different caching hierarchy schemes. *(ii)* To suggest a method and lay down the foundation for evaluating the potential of a new Web caching infrastucture in case building a real testbed environment has obstacles and is time-consuming. *(iii)* To perform the simulation analysis that identifies the key issues to designing a low latency Web caching hierarchy scheme. *(iv)* To show the potential benefit of using *CNP* as the underlying infrastructure for the Web caching.

The remainder of the paper is organized as follows. Section II briefly describes the fundamentals of *CNP*. Section III discusses the basic assumptions and the implementation of the simulation model. Section IV details how the simulation was performed and presents the outcome of the analysis of the simulation experiments. Section V is devoted to the discussion of the simulation and scrutinize why *CNP* could outperform the other Web caching hierarchy schemes.

II. CACHING NEIGHBORHOOD PROTOCOL (CNP)

A. Motivation

In [1], we started from examining a five-day proxy server access logs [16] and observed that the top 20% of the Web sites (in terms of frequency of accesses) provide 93% of the most frequently accessed top 20% documents and such top 20% documents account for 94% of the documents that have been accessed more than once. One of the direct implications that could be drawn from the above statement is that the adoption of partially replicate servers for frequently accessed Web origin servers would benefit the performance for the Web caching. Such reasoning provides the justification for *CNP*.

B. Sketch of CNP

CNP introduces the idea of *caching representatives*, which essentially are partially replicate servers [13] and mainly devoted to sharing the loads of the frequently accessed origin servers. In *CNP*, each origin server can build its own *caching neighborhood* with participating proxy servers as the neighborhood members. The proxy members, referred as *caching representatives (C-Reps)*, cache only the most frequently accessed documents of the origin server it represents and handle the requests on behalf of the origin servers. Every response returned by the caching representatives of a caching neighborhood can be regarded as directly returned by the origin server of the neighborhood.

The essentials to the success of *CNP* are request forwarding [1][25] and cache coherency control [1][24] of the caching neighborhood. Basically the request forwarding relies upon the knowledge of the representing status of proxy servers, which can be exchanged by proxy servers in a manner similar to the exchanges of routing tables among routers in the Internet. The coherency of the cached documents is enforced by the proposed *X-tolerance coherency requirement* where the origin server adjusts the value of *X* to decide the coherency level of the cached documents on its *C-Reps*. The origin server takes an active role in invalidating the outdated documents and providing the most frequently accessed documents to the *C-Reps* once it detects changes made to the documents already cached by its *C-Reps*.

CNP requires a proxy server not in the neighborhood of the origin server to always forward the requests to one of the members in the caching neighborhood, either the origin server or a *C-Rep*. Therefore, the cache coherency level is maintained and hopefully the closest caching neighborhood member would deliver the documents. The search for the “closest” caching neighborhood member by a client-side proxy server can be determined by applying the prior experience of the proxy server or some other heuristics. The proximity is measured by using the estimated round-trip turnaround time between the proxy server and the caching neighborhood members. *CNP* will be modeled in Section III.

III. Modeling of Web Caching Hierarchy Schemes

A. Basic Assumptions

In this section, we present our model used for estimating the response times. The rendering of one single document of size 2K bytes is used as the evaluating target. Such a target was chosen because it approximated the median size for the requested Web documents [17][27]. Four caching hierarchy schemes -- simple hierarchy without using any inter-proxy communication protocol, *ICP*, *CD*, and *CNP* -- were taken into consideration. Since the HTTP traffic was analyzed and decomposed into individual components, the estimated values of some of the components were derived from [17] while the others were chosen using reasonable assumptions. All the chosen values will be discussed as appropriate. In this paper, we only consider the *Get* requests that retrieves the static documents. The processings of the *Get-IMS* requests are also modeled in this section, though, the percentage of such requests are small nowadays and skipped for a clearer presentation.

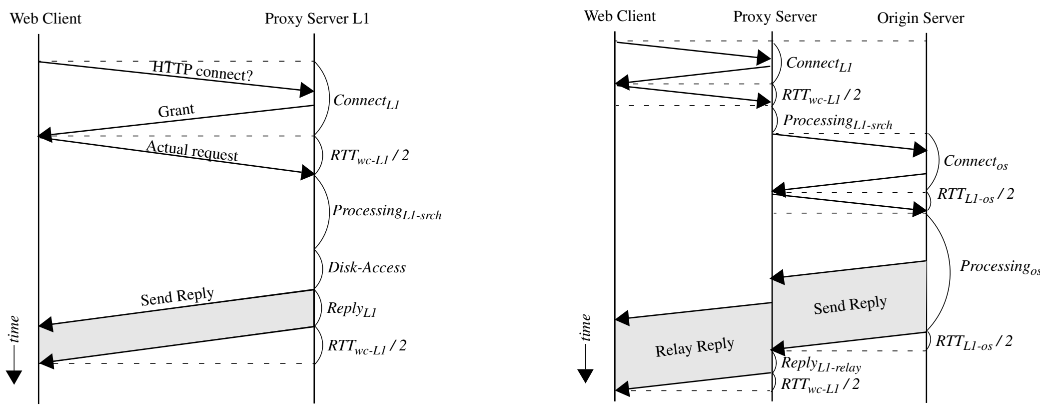
B. Modeling

In this section, a plethora of flowcharts and mathematical formulae derived from the flowcharts that constitute the foundation of this work are presented. The request processing routes were modeled up to three levels of hierarchy to derive the generic formulae for the response times for the different Web caching hierarchy schemes.

1) Simple Hierarchy

For the case of the simple hierarchy, no inter-proxy communication protocol is used. If a cache at hierarchy Level 1 (closest to the Web clients) cannot satisfy the request, it forwards the request to a cache at Level 2. The same procedure will be repeated either until the request is satisfied by a proxy server in the higher level of the hierarchy, or until it is forwarded to the origin server.

Fig. 1 illustrates how the modeling was realized. Basically the time charts capture the HTTP behaviors of the proxy server and the origin server. The parameters shown by the processing steps denote the time needed for that step. The parameters, their estimated values, their descriptions in detail, and the estimation basis for the parameters are summarized in Table 1. These are the parameters that were common for all the caching hierarchy schemes modeled in this paper.



(a) Timing illustration as a proxy server handles a request without contacting the origin server.

(b) Timing illustration as a proxy server forwards a request to the origin server and relays the response.

Figure 1: The timing diagram illustrations of the possible processing scenarios of an HTTP Get request in the one-level simple caching hierarchy.

Table 1: General Terminolgies*

Terms / Values	Descriptions	Source / Misc.
Access$_{os}$ *90,95,100[%]*	Percent of requests gaining access to the origin server before timeout	Tested values
Connect$_{os}$ *350[ms]*	The time it takes a client between sending a HTTP connection setup request to an origin server and receiving of the connection grant message	[17]-p11 Fig.10 Root & Top level
Connect$_{Lx}$ *30/100/200/350[ms]*	The time it takes a client between sending a HTTP connection setup request to a proxy server on the hierarchy level Lx and receiving the connection grant message	[17]-p13 Fig.14
Disk-Access *100[ms]*	The time it takes a proxy server to retrieve a document from the disk to its memory	[17]-p17
h$_{Lx}$ *see Section IV*	Hit rate of the Get requests for the proxy servers on the hierarchy level Lx	[28][29]
h$_{Lx\ \text{-GIMS}}$ *see Section IV*	Hit rate of the GIMS requests for the proxy servers on the hierarchy level Lx	[28][29]
HTTP$_{timeout}$ *15000, 45000[ms]*	The time it takes a proxy server to abort an outgoing HTTP connection setup request for the origin server	[5], tested values
Prob$_{Lx\text{-GIMS-fresh}}$ *50/50/50/50[%]*	Positive confirmation rate of the GIMS requests for a origin server on the hierarchy level Lx	[17]-p.10 Fig.8 ims hits
Prob$_{os\text{-GIMS-fresh}}$ *60[%]*	Positive confirmation rate of the GIMS requests for a proxy server if the request is from an ordinary proxy server	Assumed value > Prob$_{Lx\text{-GIMS-fresh}}$
Processing$_{Lx\text{-hit}}$ *400/500/500/500[ms]*	The time it takes a proxy server on the hierarchy level Lx between receiving a request and returning the last byte of the requested document from the cache (Processing$_{Lx\text{-hit}}$ = Processing$_{Lx\text{-srch}}$ + Disk-Access + Reply$_{Lx}$)	[17]-p12 Fig.11
Processing$_{Lx\text{-srch}}$ *180/250/250/250[ms]*	The time it takes a proxy server on the hierarchy level Lx between receiving a request and finding out if the requested document is in the cache or not	From Processing$_{Lx\text{-hit}}$
Processing$_{os}$ *500[ms]*	The time it takes an origin server between receiving a Get request and returning the last byte of the requested document	Based on Processing$_{Lx\text{-hit}}$

Table 1: General Terminolgies*

Terms / Values	Descriptions	Source / Misc.
$Processing_{os\text{-}GIMS}$ *350[ms]*	The time it takes an origin server between receiving a GIMS request and returning the last byte of the requested document	
$Reply_{Lx}$ *120/150/150/150[ms]*	The time it takes a proxy server on the hierarchy level Lx to reply a document in memory to the requesting party	[17]-p14, Fig.16 200 hits
$Reply_{Lx\text{-}304}$ *25/50/50/50[ms]*	The time it takes a proxy server on the hierarchy level Lx to return a “304: not modified” message	[17]-p14, Fig.16 304 hits
$Reply_{Lx\text{-}relay}$ *50/50/50/50[ms]*	The time it takes a proxy server on the hierarchy level Lx to relay a repsonse to the requesting party	Assumed values
$RTT_{A\text{-}B}$	Round-trip turnaround time between A and B. $RTT_{wc\text{-}L1}$: 20[ms], $RTT_{Lx\text{-}Lx}$: 80/160/200[ms], $RTT_{Lx\text{-}os}$: 240/240/240/240[ms], $RTT_{Lx\text{-}sibling}$: 80/100/120/140[ms]	[17], assumed values

*: If four values are specified and separated with slash (/), the values are for L1, L2, L3 and L4, respectively. The values for L4 are assumed based on the values for L3.

The modeling can also be viewed from Fig. 2. The flowchart depicts how a *Get* request is processed in a one-level simple hierarchy. Fig. 2-formulae shows the formulae derived for estimating the response times.

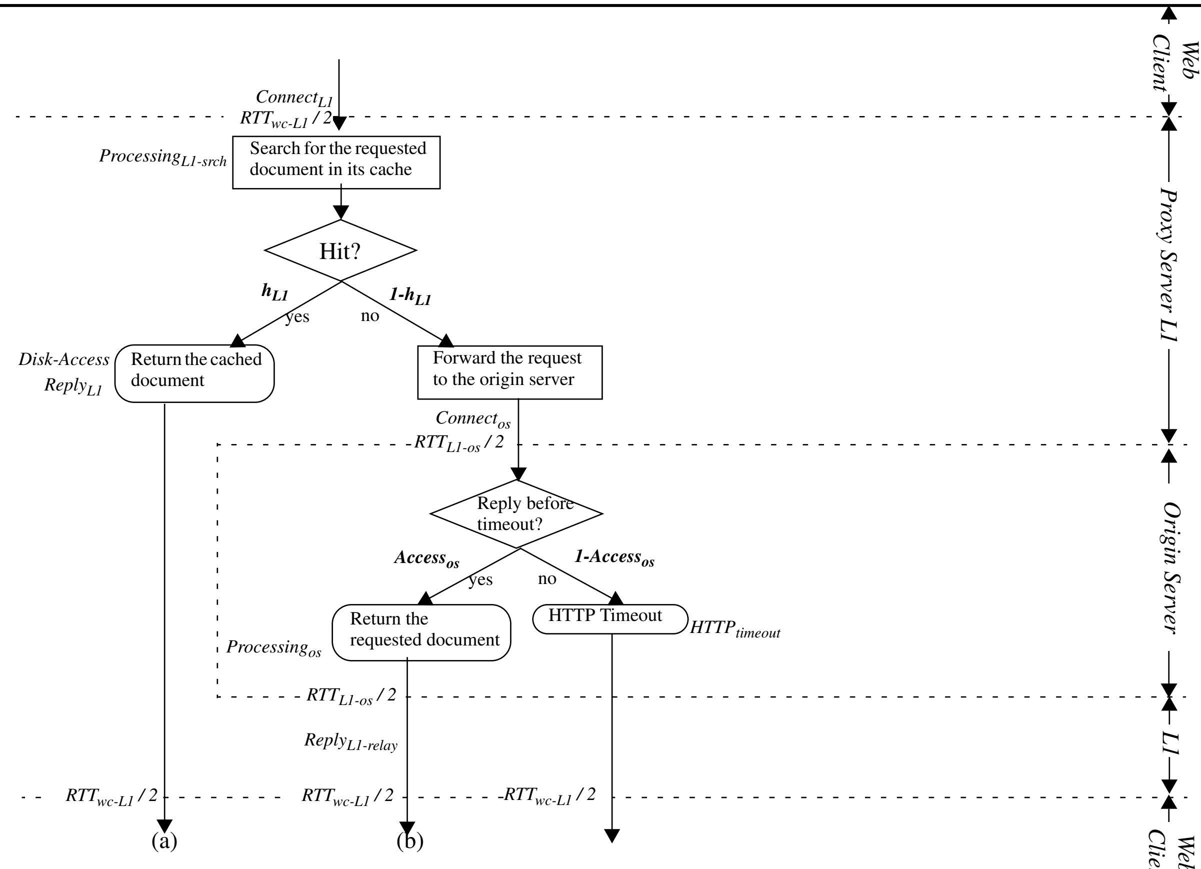


Figure 2: Simple one-level hierarchy flowchart for the Get requests

Response time_{Get} = $\text{Connect}_{L1} + \text{RTT}_{wc\text{-}L1} + h_{L1} * \text{Processing}_{L1\text{-}hit} + (1 - h_{L1}) * (\text{Access}_{os} * \text{Origin Server Hit Time}_{Get} + (1 - \text{Access}_{os}) * \text{HTTP}_{timeout})$

Origin Server Hit $\text{Time}_{Get} = \text{Processing}_{L1\text{-}srch} + \text{Connect}_{os} + \text{RTT}_{L1\text{-}os} + \text{Processing}_{os} + \text{Reply}_{L1\text{-}relay}$

Figure 2-formulae: Formulae for Simple one-level hierarchy for the Get requests

The processings of a GIMS(Get-If-Modified-Sice) request in a one-level simple hierarchy is shown in Fig. 3. Fig. 3-formulae shows the formulae derived for estimating the response times.

.

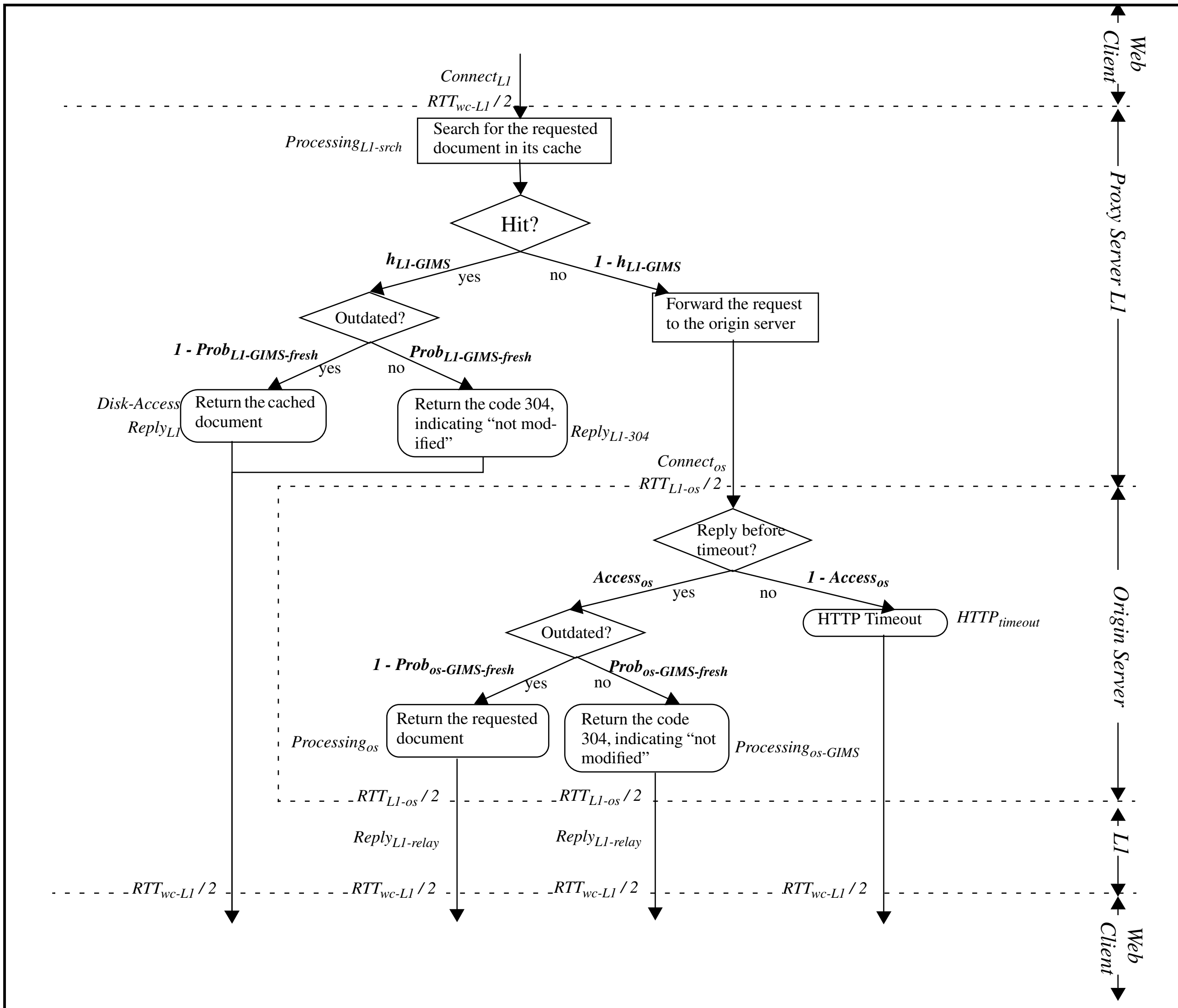


Figure 3: Simple one-level hierarchy flowchart for the GIMS requests

$\text{Response time}_{GIMS} = \text{Connect}_{L1} + \text{RTT}_{wc\text{-}L1} + h_{L1\text{-}GIMS} * \text{Proxy Server L1 Hit Time}_{GIMS} + (1 - h_{L1\text{-}GIMS}) * (\text{Access}_{os} * \text{Origin Server Hit Time}_{GIMS} + (1 - \text{Access}_{os}) * \text{HTTP}_{timeout})$

$\text{Proxy Server L1 Hit Time}_{GIMS} = \text{Prob}_{L1\text{-}GIMS\text{-}fresh} * (\text{Processing}_{L1\text{-}srch} + \text{Reply}_{L1\text{-}304}) + (1 - \text{Prob}_{L1\text{-}GIMS\text{-}fresh}) * \text{Processing}_{L1\text{-}hit}$

$\text{Origin Server Hit Time}_{GIMS} = \text{Processing}_{L1\text{-}srch} + \text{Connect}_{os} + \text{RTT}_{L1\text{-}os} + \text{Prob}_{os\text{-}GIMS\text{-}fresh} * \text{Processing}_{os\text{-}GIMS} + (1 - \text{Prob}_{os\text{-}GIMS\text{-}fresh}) * \text{Processing}_{os} + \text{Reply}_{L1\text{-}relay}$

Figure 3-formulae: Formulae for Simple one-level hierarchy for the GIMS requests

In the same way, Fig. 4 through Fig. 7 display the modeling flowcharts to depict how the Get and the GIMS requests are processed in a two-level and three-level simple hierarchy.

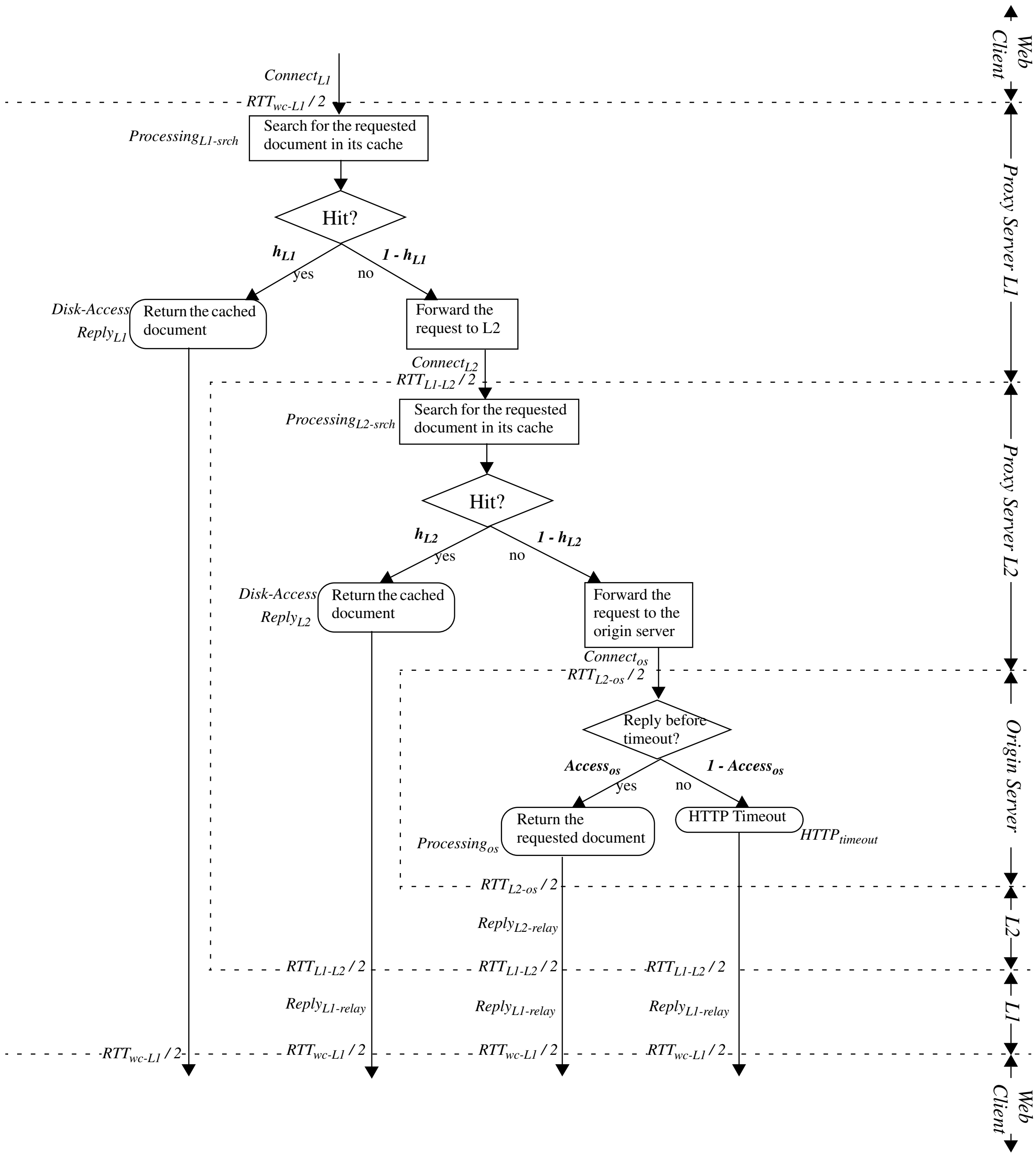


Figure 4: Simple two-level hierarchy flowchart for the Get requests

$\text{Response time}_{Get} = \text{Connect}_{L1} + \text{RTT}_{wc\text{-}L1} + h_{L1} * \text{Processing}_{L1\text{-}hit} + (1 - h_{L1}) * (h_{L2} * \text{Proxy Server L2 Hit Time}_{Get} + (1 - h_{L2}) * (\text{Access}_{os} * \text{Origin Server Hit Time}_{Get} + (1 - \text{Access}_{os}) * \text{HTTP}_{timeout}))$

$\text{Proxy Server L2 Hit Time}_{Get} = \text{Processing}_{L1\text{-}srch} + \text{Connect}_{L2} + \text{RTT}_{L1\text{-}L2} + \text{Processing}_{L2\text{-}hit} + \text{Reply}_{L1\text{-}relay}$

$\text{Origin Server Hit Time}_{Get} = \text{Processing}_{L1\text{-}srch} + \text{Connect}_{L2} + \text{RTT}_{L1\text{-}L2} + \text{Processing}_{L2\text{-}srch} + \text{Connect}_{os} + \text{RTT}_{L2\text{-}os} + \text{Processing}_{os} + \text{Reply}_{L2\text{-}relay} + \text{Reply}_{L1\text{-}relay}$

Figure 4-formulae: Formulae for Simple two-level hierarchy for the Get requests

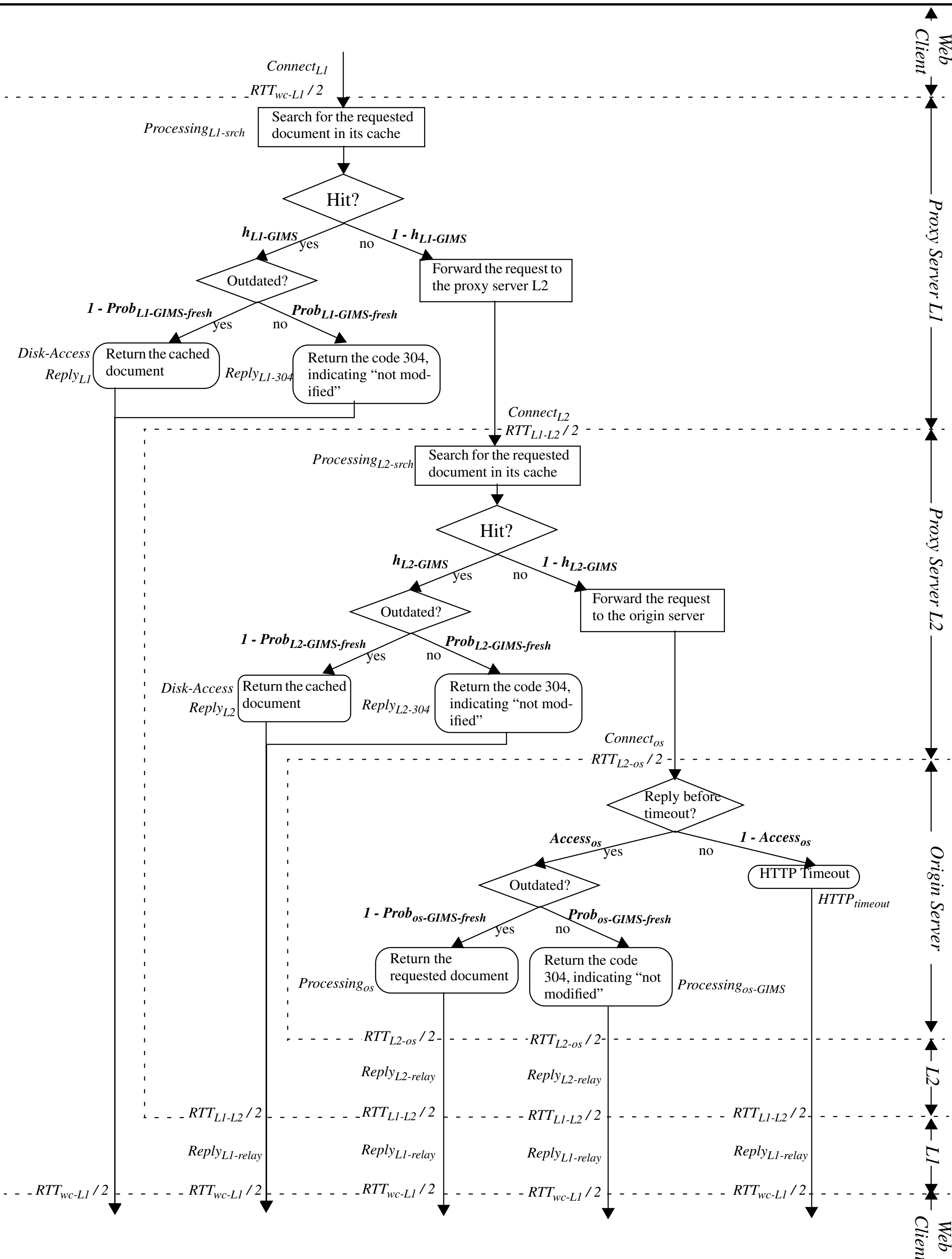


Figure 5: Simple two-level hierarchy flowchart for the GIMS requests

Response $time_{GIMS}$ = $Connect_{L1}$ + $RTT_{wc\text{-}L1}$ + $h_{L1\text{-}GIMS}$ * Proxy Server L1 Hit $Time_{GIMS}$ + (1 - $h_{L1\text{-}GIMS}$) * ($h_{L2\text{-}GIMS}$ * Proxy Server L2 Hit $Time_{GIMS}$ + (1 - $h_{L2\text{-}GIMS}$) * ($Access_{os}$ * Origin Server Hit $Time_{GIMS}$ + (1 - $Access_{os}$) * $HTTP_{timeout}$))

Proxy Server L1 Hit $Time_{GIMS}$ = $Prob_{L1\text{-}GIMS\text{-}fresh}$ * ($Processing_{L1\text{-}srch}$ + $Reply_{L1\text{-}304}$) + (1 - $Prob_{L1\text{-}GIMS\text{-}fresh}$) * $Processing_{L1\text{-}hit}$

Proxy Server L2 Hit $Time_{GIMS}$ = $Processing_{L1\text{-}srch}$ + $Connect_{L2}$ + $RTT_{L1\text{-}L2}$ + $Prob_{L2\text{-}GIMS\text{-}fresh}$ * ($Processing_{L2\text{-}srch}$ + $Reply_{L2\text{-}304}$) + (1 - $Prob_{L2\text{-}GIMS\text{-}fresh}$) * $Processing_{L2\text{-}hit}$ + $Reply_{L1\text{-}relay}$

Origin Server Hit $Time_{GIMS}$ = $Processing_{L1\text{-}srch}$ + $Connect_{L2}$ + $RTT_{L1\text{-}L2}$ + $Processing_{L2\text{-}srch}$ + $Connect_{os}$ + $RTT_{L2\text{-}os}$ + $Prob_{os\text{-}GIMS\text{-}fresh}$ * $Processing_{os\text{-}GIMS}$ + (1 - $Prob_{os\text{-}GIMS\text{-}fresh}$) * $Processing_{os}$ + $Reply_{L2\text{-}relay}$ + $Reply_{L1\text{-}relay}$

Figure 5-formulae: Formulae for Simple two-level hierarchy for the GIMS requests

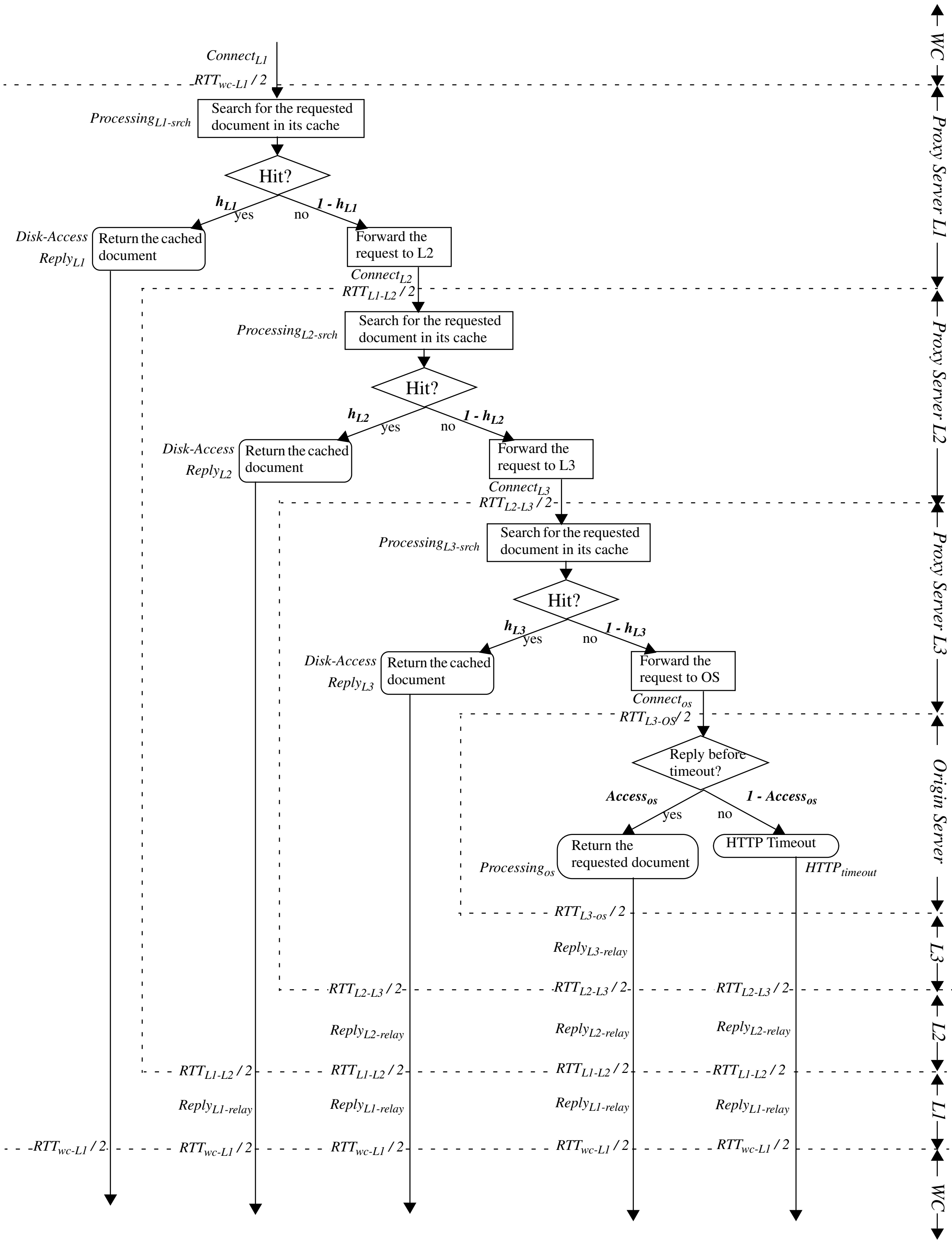


Figure 6: Simple three-level hierarchy flowchart for the Get requests

$\text{Response time}_{Get} = \text{Connect}_{L1} + \text{RTT}_{wc\text{-}L1} + h_{L1} * \text{Processing}_{L1\text{-}hit} + (1 - h_{L1}) * (h_{L2} * \text{Proxy Server L2 Hit Time}_{Get} + (1 - h_{L2}) * (h_{L3} * \text{Proxy Server L3 Hit Time}_{Get} + (1 - h_{L3}) * (\text{Access}_{os} * \text{Origin Server Hit Time}_{Get} + (1 - \text{Access}_{os}) * \text{HTTP}_{timeout})))$

$\text{Proxy Server L2 Hit Time}_{Get} = \text{Processing}_{L1\text{-}srch} + \text{Connect}_{L2} + \text{RTT}_{L1\text{-}L2} + \text{Processing}_{L2\text{-}hit} + \text{Reply}_{L1\text{-}relay}$

$\text{Proxy Server L3 Hit Time}_{Get} = \text{Processing}_{L1\text{-}srch} + \text{Connect}_{L2} + \text{RTT}_{L1\text{-}L2} + \text{Processing}_{L2\text{-}srch} + \text{Connect}_{L3} + \text{RTT}_{L2\text{-}L3} + \text{Processing}_{L3\text{-}hit} + \text{Reply}_{L2\text{-}relay} + \text{Reply}_{L1\text{-}relay}$

$\text{Origin Server Hit Time}_{Get} = \text{Processing}_{L1\text{-}srch} + \text{Connect}_{L2} + \text{RTT}_{L1\text{-}L2} + \text{Processing}_{L2\text{-}srch} + \text{Connect}_{L3} + \text{RTT}_{L2\text{-}L3} + \text{Processing}_{L3\text{-}srch} + \text{Connect}_{os} + \text{RTT}_{L3\text{-}os} + \text{Processing}_{os} + \text{Reply}_{L3\text{-}relay} + \text{Reply}_{L2\text{-}relay} + \text{Reply}_{L1\text{-}relay}$

Figure 6-formulae: Formulae for Simple three-level hierarchy for the Get requests

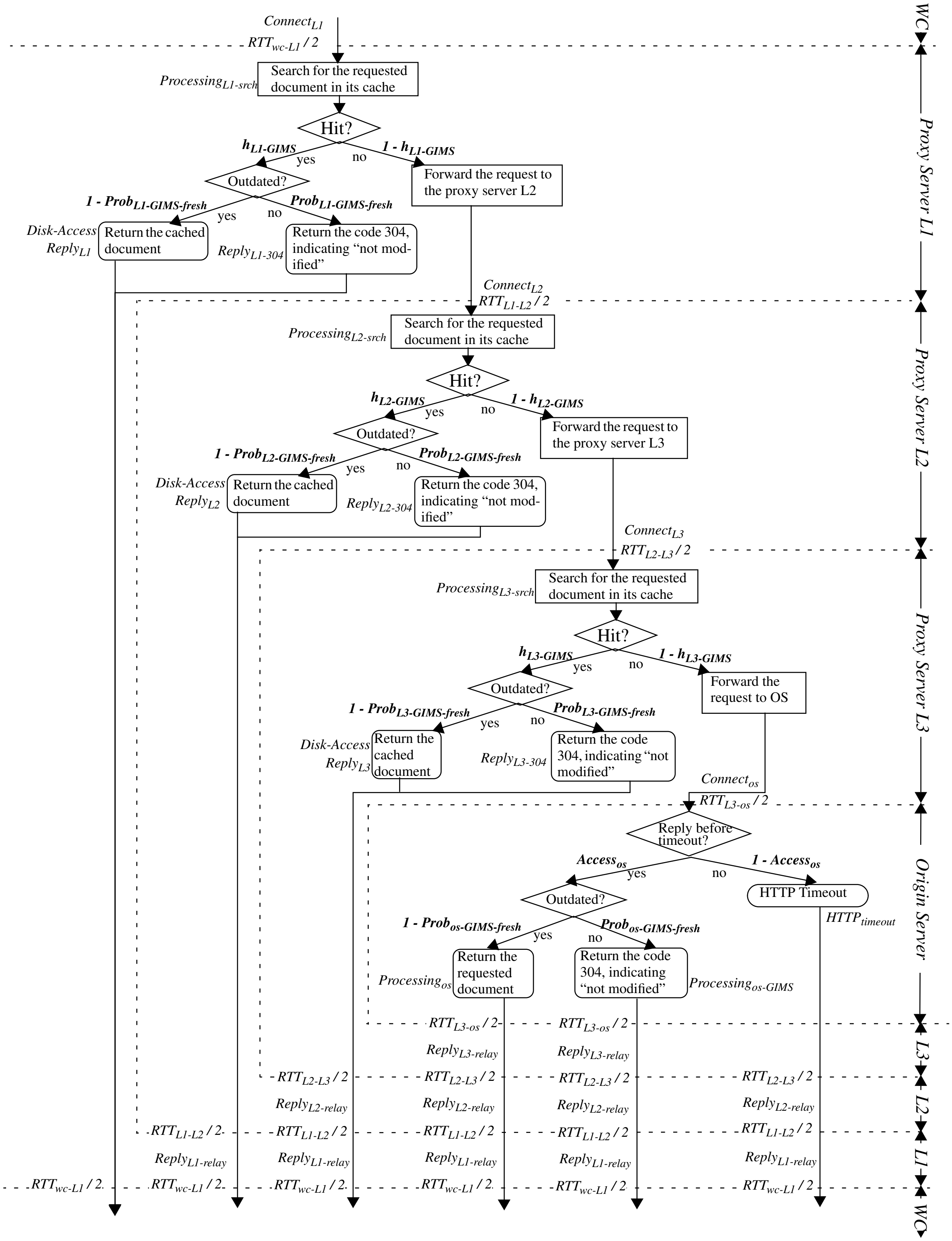


Figure 7: Simple three-level hierarchy flowchart for the GIMS requests

Response $time_{GIMS}$ = $Connect_{L1}$ + $RTT_{wc\text{-}L1}$ + $h_{L1\text{-}GIMS}$ * Proxy Server L1 Hit $Time_{GIMS}$ + (1 - $h_{L1\text{-}GIMS}$) * ($h_{L2\text{-}GIMS}$ * Proxy Server L2 Hit $Time_{GIMS}$ + (1 - $h_{L2\text{-}GIMS}$) * ($h_{L3\text{-}GIMS}$ * Proxy Server L3 Hit $Time_{GIMS}$ + (1 - $h_{L3\text{-}GIMS}$) * ($Access_{os}$ * Origin Server Hit $Time_{GIMS}$ + (1 - $Access_{os}$) * $HTTP_{timeout}$)))

Proxy Server L1 Hit $Time_{GIMS}$ = $Prob_{L1\text{-}GIMS\text{-}fresh}$ * ($Processing_{L1\text{-}srch}$ + $Reply_{L1\text{-}304}$) + (1 - $Prob_{L1\text{-}GIMS\text{-}fresh}$) * $Processing_{L1\text{-}hit}$

Proxy Server L2 Hit $Time_{GIMS}$ = $Processing_{L1\text{-}srch}$ + $Connect_{L2}$ + $RTT_{L1\text{-}L2}$ + $Prob_{L2\text{-}GIMS\text{-}fresh}$ * ($Processing_{L2\text{-}srch}$ + $Reply_{L2\text{-}304}$) + (1 - $Prob_{L2\text{-}GIMS\text{-}fresh}$) * $Processing_{L2\text{-}hit}$ + $Reply_{L1\text{-}relay}$

Proxy Server L3 Hit $Time_{GIMS}$ = $Processing_{L1\text{-}srch}$ + $Connect_{L2}$ + $RTT_{L1\text{-}L2}$ + $Processing_{L2\text{-}srch}$ + $Connect_{L3}$ + $RTT_{L2\text{-}L3}$ + $Prob_{L3\text{-}GIMS\text{-}fresh}$ * ($Processing_{L3\text{-}srch}$ + $Reply_{L3\text{-}304}$) + (1 - $Prob_{L3\text{-}GIMS\text{-}fresh}$) * $Processing_{L3\text{-}hit}$ + $Reply_{L2\text{-}relay}$ + $Reply_{L1\text{-}relay}$

Origin Server Hit $Time_{GIMS}$ = $Processing_{L1\text{-}srch}$ + $Connect_{L2}$ + $RTT_{L1\text{-}L2}$ + $Processing_{L2\text{-}srch}$ + $Connect_{L3}$ + $RTT_{L2\text{-}L3}$ + $Processing_{L3\text{-}srch}$ + $Connect_{os}$ + $RTT_{L3\text{-}os}$ + $Prob_{os\text{-}GIMS\text{-}fresh}$ * $Processing_{os\text{-}GIMS}$ + (1 - $Prob_{os\text{-}GIMS\text{-}fresh}$) * $Processing_{os}$ + $Reply_{L3\text{-}relay}$ + $Reply_{L2\text{-}relay}$ + $Reply_{L1\text{-}relay}$

Figure 7-formulae: Formulae for Simple three-level hierarchy for the GIMS requests

2) ICP

For *ICP,* the inter-proxy communication protocol, has been replaced by *CD* in the current *Squid* implementation of *NLANR* [21], the modelings of the one-level and two-level hierarchy are shown in from Figure 8 to Figure 11. The major difference between *ICP* and simple hierarchy is that *ICP* employs sibling proxy servers nearby to achieve a higher hit rate at a certain level of the hierarchy. Hence, the parameters such as $h_{Lx\text{-}sibling\text{-}Get}$, $RTT\text{-}Hit_{Lx\text{-}sibling}$, and $RTT\text{-}Miss_{Lx\text{-}sibling}$ that portray the sibling accessing characteristics are introduced in Table 2. Table 2 also contains the parameters exclusively used for *CD.*

Table 2: Terminologies for ICP and CD

Terms / Values	Descriptions	Source / Misc.
$\text{Digest-Search}_{\text{Lx}}$ *50/50/50/50*[ms]	The time it takes a proxy server on the hierarchy level Lx to look up the cache digests of its siblings	Assumed values. We suppose the Digest search operations are computationally efficient.
$\text{h}_{\text{Lx-sibling-Get}}$ see Section IV	Positive confirmation rate of ICP GIMS requests for proxy servers to their siblings on the hierarchy level Lx	The hit rates for siblings on any level of the hierarchy is supposed to be low because of the overlap of popular documents in the caches.
$\text{h}_{\text{Lx-sibling-GIMS}}$ see Section IV	Hit rate of ICP Get requests for proxy servers to their siblings on the hierarchy level Lx	Same reason as $\text{h}_{\text{Lx-sibling-Get}}$
$\text{RTT-Hit}_{\text{Lx-sibling}}$ *60/80/100/120*[ms]	Round-trip turnaround time for a proxy server to receive the first ICP-hit message from one of its siblings	[18]
$\text{RTT-Miss}_{\text{Lx-sibling}}$ *120/160/200/240*[ms]	Round-trip turnaround time for a proxy server to receive all ICP-miss messages from its siblings	Assumed values based on [18]
$\text{h}_{\text{Lx-sibling-false-hit}}$ *14.5*[%]	The false hit rate of a Get request for a sibling of a proxy server on the hierarchy level Lx	[18]-p13 table2
$\text{h}_{\text{Lx-sibling-false-GIMS}}$ *14.5*[%]	The false hit rate of a GIMS request for a sibling of a proxy server on the hierarchy level Lx	Assumed values based on $\text{h}_{\text{Lx-sibling-false-hit}}$
$\text{Reply}_{\text{Lx-sibling}}$ *120/150/150/150*[ms]	The time it takes a sibling of a proxy server on the hierarchy level Lx to reply a document in memory to the requesting party	$= \text{Reply}_{\text{Lx}}$

The modelings of the one-level and two-level ICP hierarchy are shown in Fig. 8 through Fig. 11. The corresponding formulae derived for estimating the response times are shown in Fig. 8-formulae through Fig. 11-formulae.

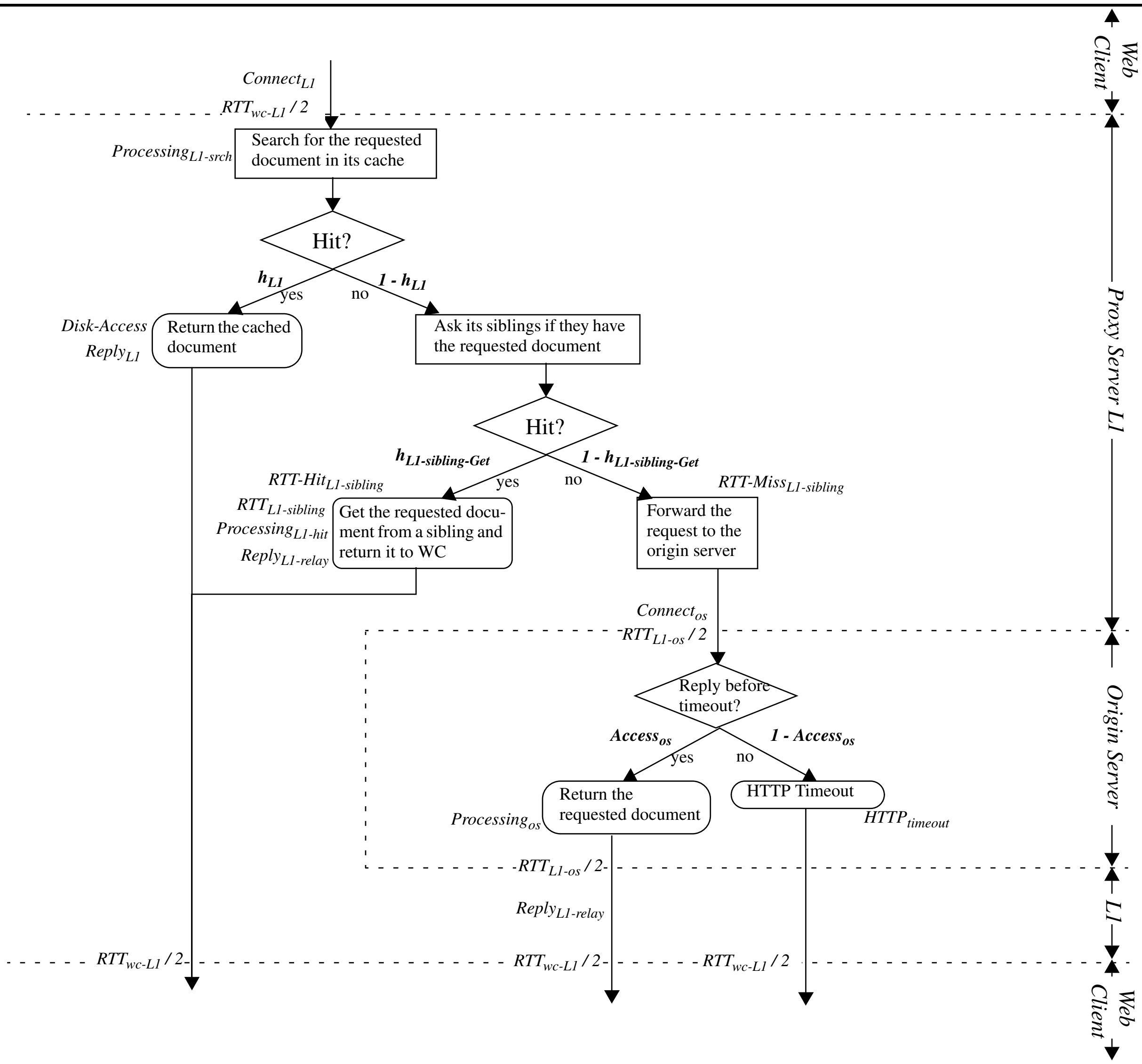


Figure 8: ICP one-level hierarchy flowchart for the Get requests

Response $time_{Get}$ = $Connect_{L1} + RTT_{wc\text{-}L1} + h_{L1} * Processing_{L1\text{-}hit} + (1 - h_{L1}) * (h_{L1\text{-}sibling\text{-}Get}$ * Sibling Proxy Server Hit $Time_{Get} + (1 - h_{L1\text{-}sibling\text{-}Get}) * (Access_{os}$ * Origin Server Hit $Time_{Get} + (1 - Access_{os}) * HTTP_{timeout}))$

Sibling Proxy Server Hit $Time_{Get} = Processing_{L1\text{-}srch} + RTT\text{-}Hit_{L1\text{-}sibling} + RTT_{L1\text{-}sibling} + Processing_{L1\text{-}hit} + Reply_{L1\text{-}relay}$

Origin Server Hit $Time_{Get} = Processing_{L1\text{-}srch} + RTT\text{-}Miss_{L1\text{-}sibling} + Connect_{os} + RTT_{L1\text{-}os} + Processing_{os} + Reply_{L1\text{-}relay}$

Figure 8-formulae: Formulae for ICP one-level hierarchy for the Get requests

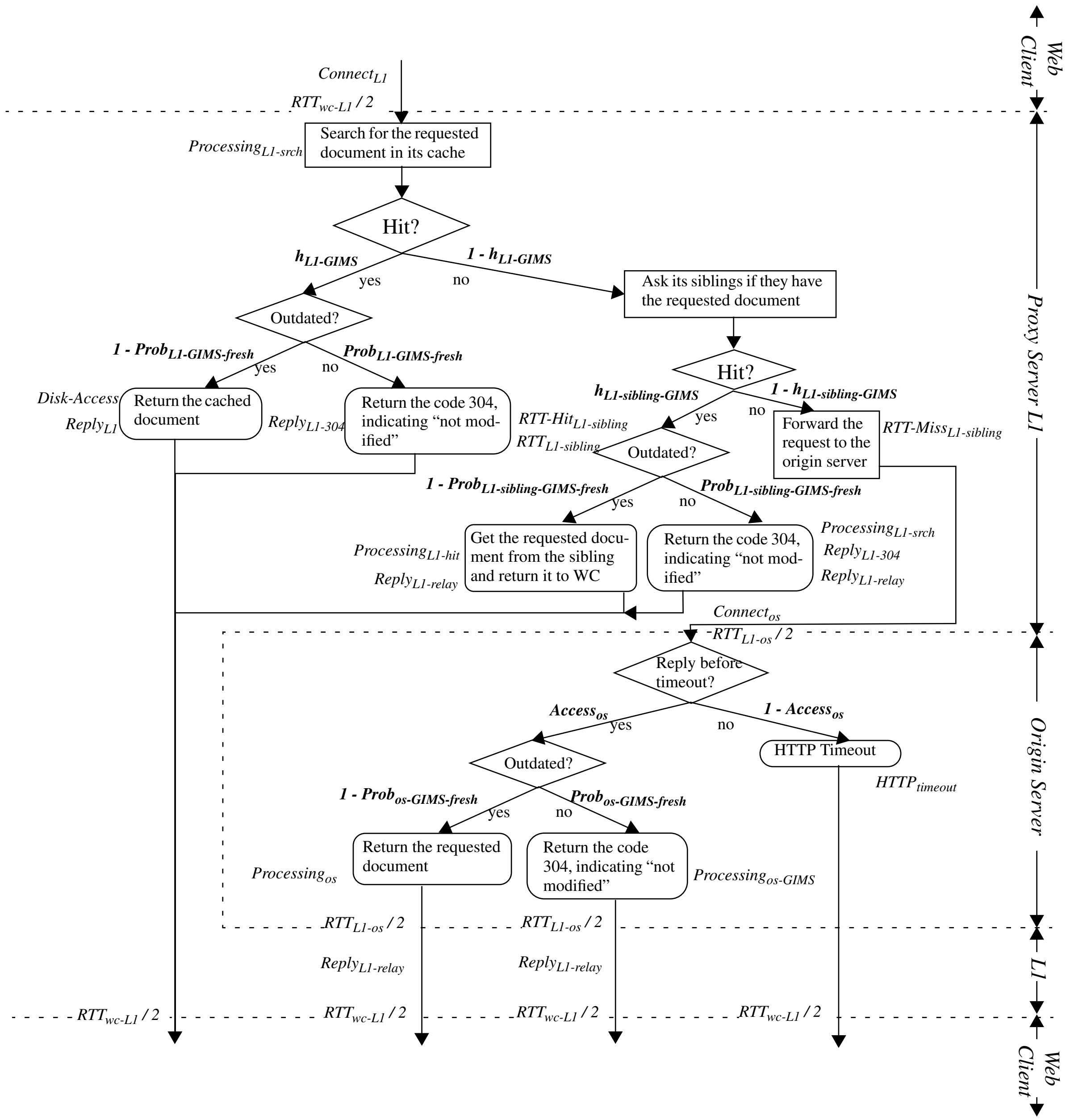


Figure 9: ICP one-level hierarchy flowchart for the GIMS requests

$$\text{Response time}_{GIMS} = \text{Connect}_{L1} + \text{RTT}_{wc\text{-}L1} + h_{L1\text{-}GIMS} * \text{ICP Proxy Server L1 Hit Time}_{GIMS} + (1 - h_{L1\text{-}GIMS}) * (h_{L1\text{-}sibling\text{-}GIMS} * \text{ICP Proxy Server L1-sibling Hit Time}_{GIMS} + (1 - h_{L1\text{-}sibling\text{-}GIMS}) * (\text{Access}_{os} * \text{Origin Server Hit Time}_{GIMS} + (1 - \text{Access}_{os}) * \text{HTTP}_{timeout}))$$

$$\text{ICP Proxy Server L1 Hit Time}_{GIMS} = \text{Prob}_{L1\text{-}GIMS\text{-}fresh} * (\text{Processing}_{L1\text{-}srch} + \text{Reply}_{L1\text{-}304}) + (1 - \text{Prob}_{L1\text{-}GIMS\text{-}fresh}) * \text{Processing}_{L1\text{-}hit}$$

$$\text{ICP Proxy Server L1-sibling Hit Time}_{GIMS} = \text{Processing}_{L1\text{-}srch} + \text{RTT-Hit}_{L1\text{-}sibling} + \text{RTT}_{L1\text{-}sibling} + (1 - \text{Prob}_{L1\text{-}sibling\text{-}GIMS\text{-}fresh}) * \text{Processing}_{L1\text{-}hit} + \text{Prob}_{L1\text{-}sibling\text{-}GIMS\text{-}fresh} * (\text{Processing}_{L1\text{-}srch} + \text{Reply}_{L1\text{-}304}) + \text{Reply}_{L1\text{-}relay}$$

$$\text{Origin Server Hit Time}_{GIMS} = \text{Processing}_{L1\text{-}srch} + \text{RTT-Miss}_{L1\text{-}sibling} + \text{Connect}_{os} + \text{RTT}_{L1\text{-}os} + \text{Prob}_{os\text{-}GIMS\text{-}fresh} * \text{Processing}_{os\text{-}GIMS} + (1 - \text{Prob}_{os\text{-}GIMS\text{-}fresh}) * \text{Processing}_{os} + \text{Reply}_{L1\text{-}relay}$$

Figure 9-formulae: Formulae for ICP one-level hierarchy for the GIMS requests

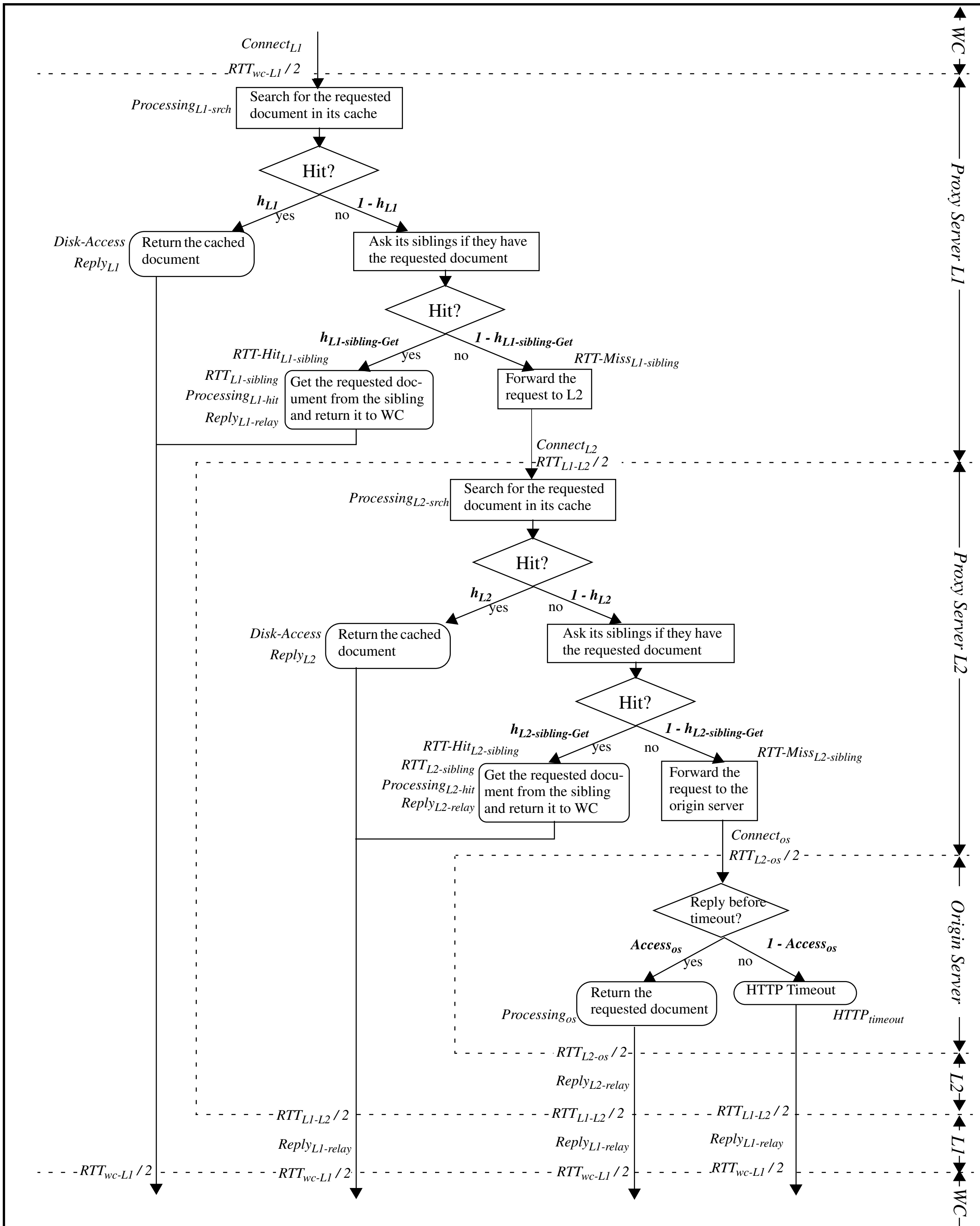


Figure 10: ICP two-level hierarchy flowchart for the Get request

Response $time_{Get}$ = $Connect_{L1}$ + RTT_{wc-L1} + h_{L1} * $Processing_{L1-hit}$ + (1 - h_{L1}) * ($h_{L1-sibling-Get}$ * ICP Proxy Server L1 sibling Hit $Time_{Get}$ + (1 - $h_{L1-sibling-Get}$) * (h_{L2} * $Processing_{L2-hit}$ + (1 - h_{L2}) * ($h_{L2-sibling-Get}$ * ICP Proxy Server L2-sibling Hit TimeGet + (1 - $h_{L2-sibling-Get}$) * ($Access_{os}$ * Origin Server Hit $Time_{Get}$ + (1 - $Access_{os}$) * $HTTP_{timeout}$))))

ICP Proxy Server L1-sibling Hit $Time_{Get}$ = $Processing_{L1-srch}$ + $RTT\text{-}Hit_{L1-sibling}$ + $RTT_{L1-sibling}$ + $Processing_{L1-hit}$ + $Reply_{L1-relay}$

ICP Proxy Server L2-sibling Hit $Time_{Get}$ = $Processing_{L1-srch}$ + $RTT\text{-}Miss_{L1-sibling}$ + $Connect_{L2}$ + RTT_{L1-L2} + $Processing_{L2-srch}$ + $RTT\text{-}Hit_{L2-sibling}$ + $RTT_{L2-sibling}$ + $Processing_{L2-hit}$ + $Reply_{L2-relay}$ + $Reply_{L1-relay}$

Origin Server Hit $Time_{Get}$ = $Processing_{L1-srch}$ + $RTT\text{-}Miss_{L1-sibling}$ + $Connect_{L2}$ + RTT_{L1-L2} + $Processing_{L2-srch}$ + $RTT\text{-}Miss_{L2-sibling}$ + $Connect_{os}$ + RTT_{L2-os} + $Processing_{os}$ + $Reply_{L2-relay}$ + $Reply_{L1-relay}$

Figure 10-formulae: Formulae for ICP two-level hierarchy for the Get requests

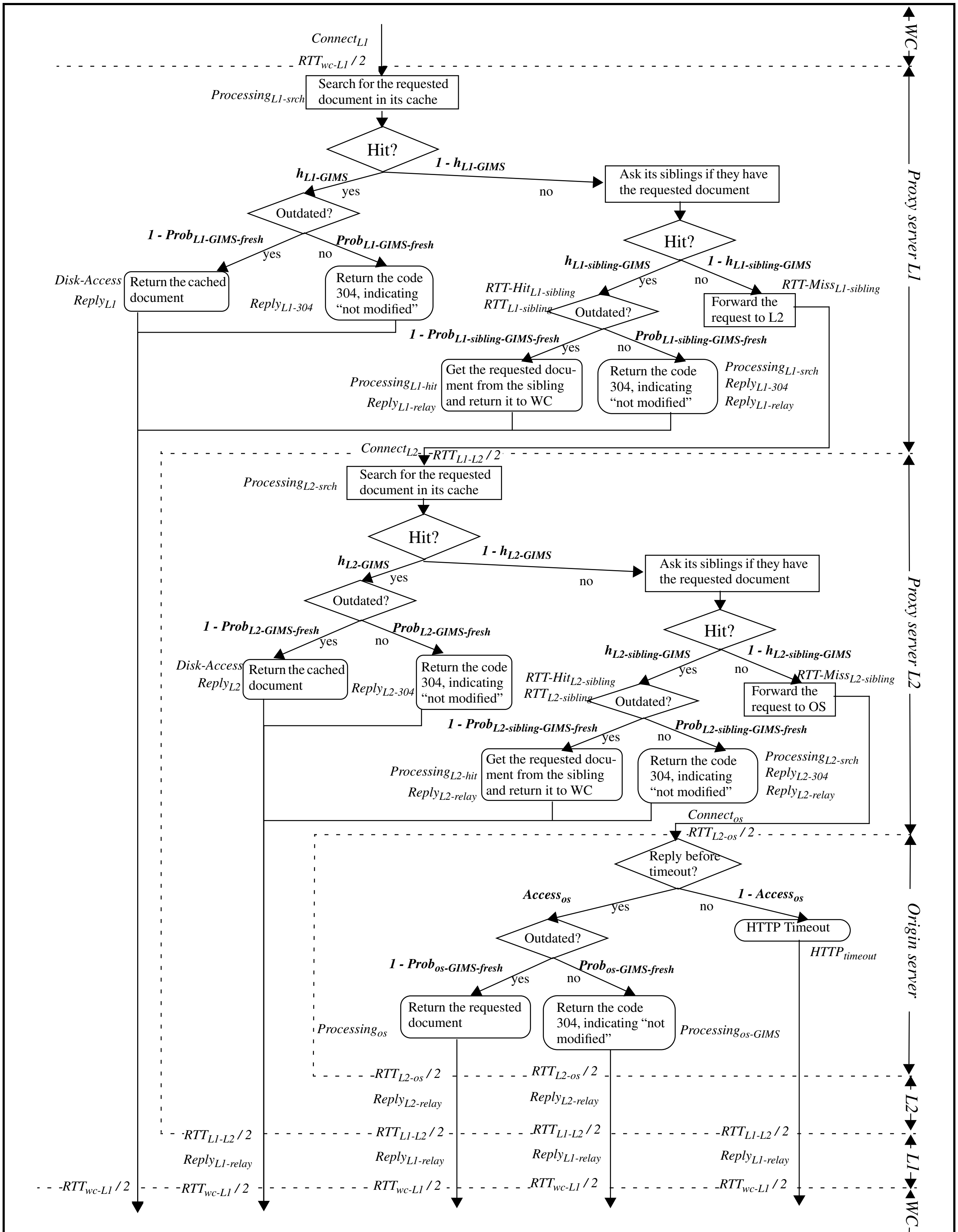


Figure 11: ICP two-level hierarchy flowchart for the GIMS requests

$\text{Response time}_{GIMS} = \text{Connect}_{L1} + \text{RTT}_{wc\text{-}L1} + h_{L1\text{-}GIMS} * \text{ICP Proxy Server L1 Hit Time}_{GIMS} + (1 - h_{L1\text{-}GIMS}) * (h_{L1\text{-}sibling\text{-}GIMS} * \text{ICP Proxy Server L1-sibling Hit Time}_{GIMS} + (1 - h_{L1\text{-}sibling\text{-}GIMS}) * (h_{L2\text{-}GIMS} * \text{ICP Proxy Server L2 Hit Time}_{GIMS} + (1 - h_{L2\text{-}GIMS}) * (h_{L2\text{-}sibling\text{-}GIMS} * \text{ICP Proxy Server L2-sibling Hit Time}_{GIMS} + (1 - h_{L2\text{-}sibling\text{-}GIMS}) * (\text{Access}_{os} * \text{Origin Server Hit Time}_{GIMS} + (1 - \text{Access}_{os}) * \text{HTTP}_{timeout}))))$

$\text{ICP Proxy Server L1 Hit Time}_{GIMS} = \text{Prob}_{L1\text{-}GIMS\text{-}fresh} * (\text{Processing}_{L1\text{-}srch} + \text{Reply}_{L1\text{-}304}) + (1 - \text{Prob}_{L1\text{-}GIMS\text{-}fresh}) * \text{Processing}_{L1\text{-}hit}$

$\text{ICP Proxy Server L1-sibling Hit Time}_{GIMS} = \text{Processing}_{L1\text{-}srch} + \text{RTT-Hit}_{L1\text{-}sibling} + \text{RTT}_{L1\text{-}sibling} + (1 - \text{Prob}_{L1\text{-}sibling\text{-}GIMS\text{-}fresh}) * \text{Processing}_{L1\text{-}hit} + \text{Prob}_{L1\text{-}sibling\text{-}GIMS\text{-}fresh} * (\text{Processing}_{L1\text{-}srch} + \text{Reply}_{L1\text{-}304}) + \text{Reply}_{L1\text{-}relay}$

$\text{ICP Proxy Server L2 Hit Time}_{GIMS} = \text{Processing}_{L1\text{-}srch} + \text{RTT-Miss}_{L1\text{-}sibling} + \text{Connect}_{L2} + \text{RTT}_{L1\text{-}L2} + \text{Prob}_{L2\text{-}GIMS\text{-}fresh} * (\text{Processing}_{L2\text{-}srch} + \text{Reply}_{L2\text{-}304}) + (1 - \text{Prob}_{L2\text{-}GIMS\text{-}fresh}) * \text{Processing}_{L2\text{-}hit} + \text{Reply}_{L1\text{-}relay}$

$\text{ICP Proxy Server L2-sibling Hit Time}_{GIMS} = \text{Processing}_{L1\text{-}srch} + \text{RTT-Miss}_{L1\text{-}sibling} + \text{Connect}_{L2} + \text{RTT}_{L1\text{-}L2} + \text{Processing}_{L2\text{-}srch} + \text{RTT-Hit}_{L2\text{-}sibling} + \text{RTT}_{L2\text{-}sibling} + (1 - \text{Prob}_{L2\text{-}sibling\text{-}GIMS\text{-}fresh}) * \text{Processing}_{L2\text{-}hit} + \text{Prob}_{L2\text{-}sibling\text{-}GIMS\text{-}fresh} * (\text{Processing}_{L2\text{-}srch} + \text{Reply}_{L2\text{-}304}) + \text{Reply}_{L2\text{-}relay} + \text{Reply}_{L1\text{-}relay}$

$\text{Origin Server Hit Time}_{GIMS} = \text{Processing}_{L1\text{-}srch} + \text{RTT-Miss}_{L1\text{-}sibling} + \text{Connect}_{L2} + \text{RTT}_{L1\text{-}L2} + \text{Processing}_{L2\text{-}srch} + \text{RTT-Miss}_{L2\text{-}sibling} + \text{Connect}_{os} + \text{RTT}_{L2\text{-}os} + \text{Prob}_{os\text{-}GIMS\text{-}fresh} * \text{Processing}_{os\text{-}GIMS} + (1 - \text{Prob}_{os\text{-}GIMS\text{-}fresh}) * \text{Processing}_{os} + \text{Reply}_{L2\text{-}relay} + \text{Reply}_{L1\text{-}relay}$

Figure 11-formulae: Formulae for ICP two-level hierarchy for the GIMS requests

3) CD

Like its predecessor *ICP*, *CD* also involves sibling proxy servers in order to achieve a higher hit rate at a certain level of the hierarchy. We assume that the hierarchies for both *ICP* and *CD* are built in a way such that the proxy servers from the bottom level to the top level of the hierarchy follow the physical network hierarchy, i.e., from covering a smaller region to covering a larger region. An extention to the above assumption is that the siblings of a proxy server should reside on the same or similar levels of the hierarchy. Such assumptions vindicate the notion that the siblings should be in the vicinity of the networking neighborhood so as to reduce the response times. The selection of the values of some of the parameters are influenced by such assumptions. The modelings of one-level and two-level CD hierarchies are shown in Fig. 12 through Fig. 15. The corresponding formulae derived for estimating the response times are shown in Fig. 12-formulae through Fig. 15-formulae.

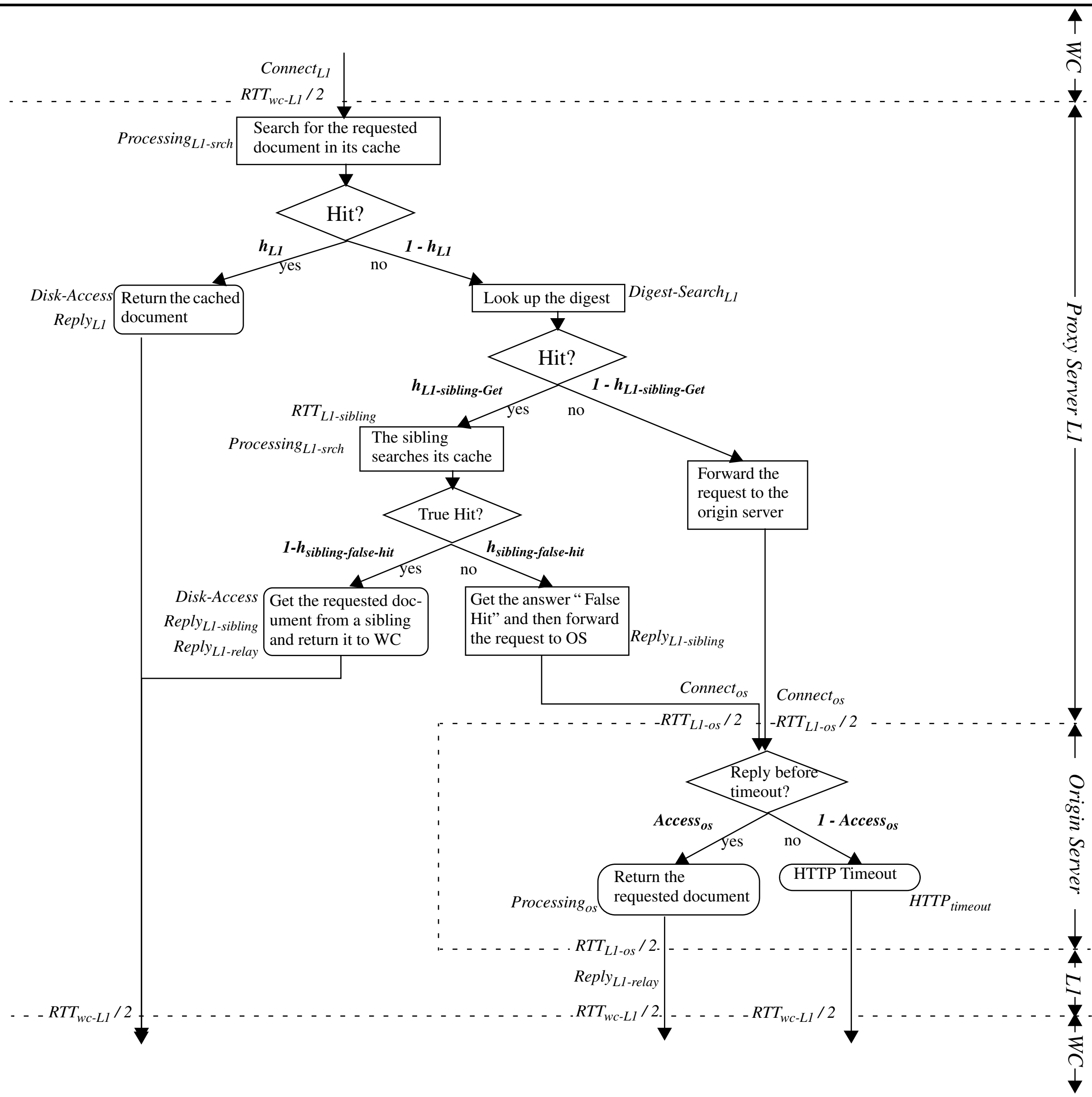


Figure 12: CD one-level hierarchy flowchart for the Get requests

.

$\text{Response time}_{Get} = \text{Connect}_{L1} + RTT_{wc\text{-}L1} + h_{L1} * \text{Processing}_{L1\text{-}hit} + (1 - h_{L1}) * (h_{L1\text{-}sibling\text{-}Get} * ((1 - h_{L1\text{-}sibling\text{-}false\text{-}hit}) * \text{CD Proxy Server L1-sibling True Hit Time}_{Get} + h_{L1\text{-}sibling\text{-}false\text{-}hit} * \text{CD Proxy Server L1-Sibling False Hit Penalty}_{Get}) + (h_{L1\text{-}sibling\text{-}Get} * h_{L1\text{-}sibling\text{-}false\text{-}hit} + (1 - h_{L1\text{-}sibling\text{-}Get})) * (\text{Access}_{os} * \text{Origin Server Hit Time}_{Get} + (1 - \text{Access}_{os}) * \text{HTTP}_{timeout}))$

$\text{CD Proxy Server L1-sibling True Hit Time}_{Get} = \text{Processing}_{L1\text{-}srch} + \text{Digest-Search}_{L1} + RTT_{L1\text{-}sibling} + \text{Processing}_{L1\text{-}hit} + \text{Reply}_{L1\text{-}relay}$

$\text{CD Proxy Server L1-sibling False Hit Penalty}_{Get} = RTT_{L1\text{-}sibling} + \text{Processing}_{L1\text{-}srch} + \text{Reply}_{L1\text{-}sibling}$

$\text{Origin Server Hit Time}_{Get} = \text{Processing}_{L1\text{-}srch} + \text{Digest-Search}_{L1} + \text{Connect}_{os} + RTT_{L1\text{-}os} + \text{Processing}_{os} + \text{Reply}_{L1\text{-}relay}$

Figure 12-formulae: Formulae for CD one-level hierarchy for the Get requests

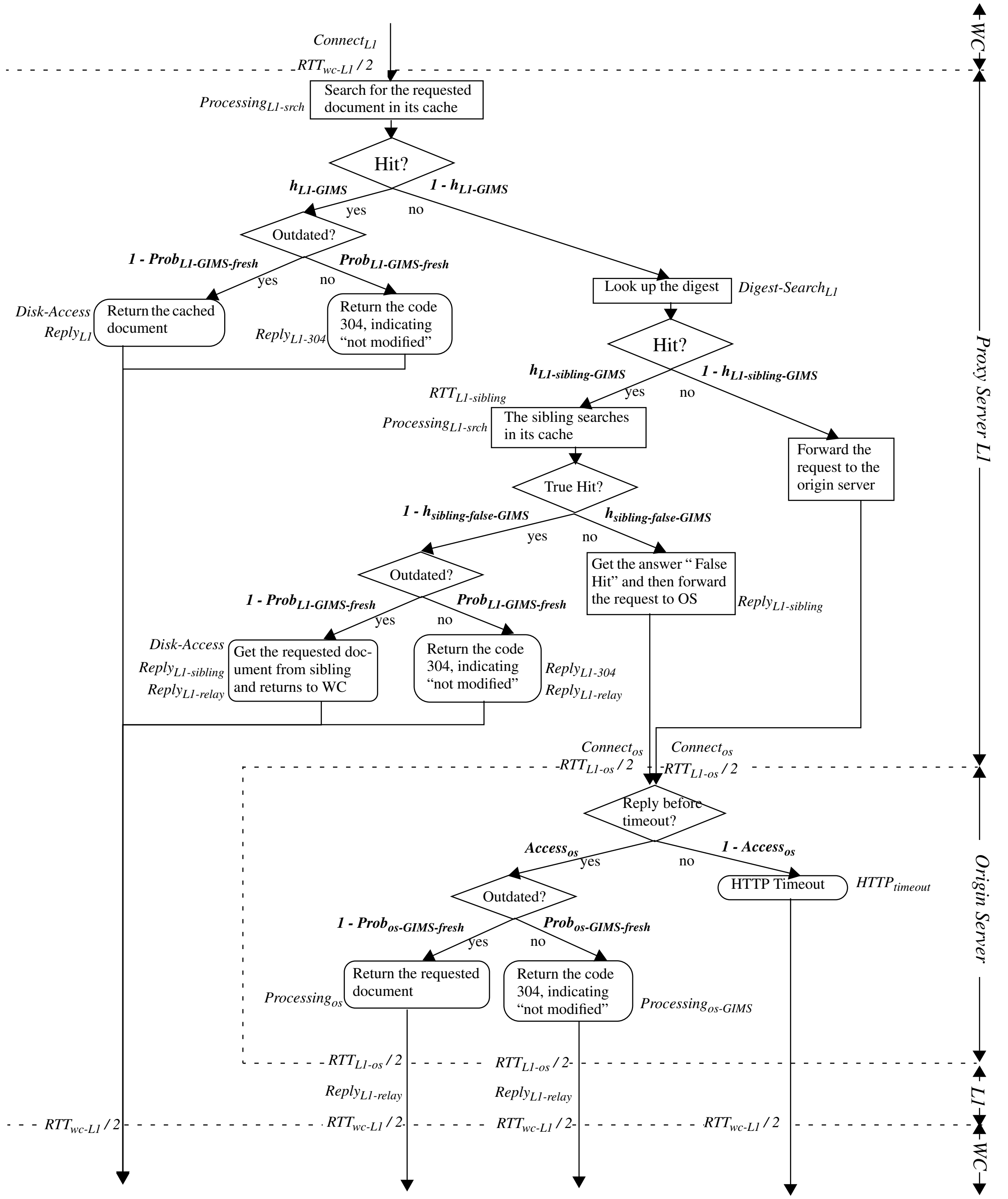


Figure 13: CD one-level hierarchy flowchart for the GIMS requests

$$\text{Response time}_{GIMS} = \text{Connect}_{L1} + \text{RTT}_{wc\text{-}L1} + h_{L1\text{-}GIMS} * \text{CD Proxy Server L1 Hit Time}_{GIMS} + (1 - h_{L1\text{-}GIMS}) * (h_{L1\text{-}sibling\text{-}GIMS} * (1 - h_{L1\text{-}sibling\text{-}false\text{-}GIMS}) * \text{CD Proxy Server L1-sibling True Hit Time}_{GIMS} + h_{L1\text{-}sibling\text{-}GIMS} * h_{L1\text{-}sibling\text{-}false\text{-}GIMS} * \text{CD Proxy Server L1-Sibling False Hit Penalty}_{GIMS} + (h_{L1\text{-}sibling\text{-}GIMS} * h_{L1\text{-}sibling\text{-}false\text{-}GIMS} + (1 - h_{L1\text{-}sibling\text{-}GIMS})) * (\text{Access}_{os} * \text{Origin Server Hit Time}_{GIMS} + (1 - \text{Access}_{os}) * \text{HTTP}_{time\text{-}out}))$$

$$\text{CD Proxy Server L1 Hit Time}_{GIMS} = \text{Prob}_{L1\text{-}GIMS\text{-}fresh} * (\text{Processing}_{L1\text{-}srch} + \text{Reply}_{L1\text{-}304}) + (1 - \text{Prob}_{L1\text{-}GIMS\text{-}fresh}) * \text{Processing}_{L1\text{-}hit}$$

$$\text{CD Proxy Server L1-sibling True Hit Time}_{GIMS} = \text{Processing}_{L1\text{-}srch} + \text{Digest-Search}_{L1} + \text{RTT}_{L1\text{-}sibling} + \text{Prob}_{L1\text{-}GIMS\text{-}fresh} * (\text{Processing}_{L1\text{-}srch} + \text{Reply}_{L1\text{-}304}) + (1 - \text{Prob}_{L1\text{-}GIMS\text{-}fresh}) * \text{Processing}_{L1\text{-}hit} + \text{Reply}_{L1\text{-}relay}$$

$$\text{CD Proxy Server L1-sibling False Hit Penalty}_{GIMS} = \text{RTT}_{L1\text{-}sibling} + \text{Processing}_{L1\text{-}srch} + \text{Reply}_{L1\text{-}sibling}$$

$$\text{Origin Server Hit Time}_{GIMS} = \text{Processing}_{L1\text{-}srch} + \text{Connect}_{os} + \text{RTT}_{L1\text{-}os} + \text{Prob}_{os\text{-}GIMS\text{-}fresh} * \text{Processing}_{os\text{-}GIMS} + (1 - \text{Prob}_{os\text{-}GIMS\text{-}fresh}) * \text{Processing}_{os} + \text{Reply}_{L1\text{-}relay}$$

Figure 13-formula: Formula for CD one-level hierarchy for the GIMS requests

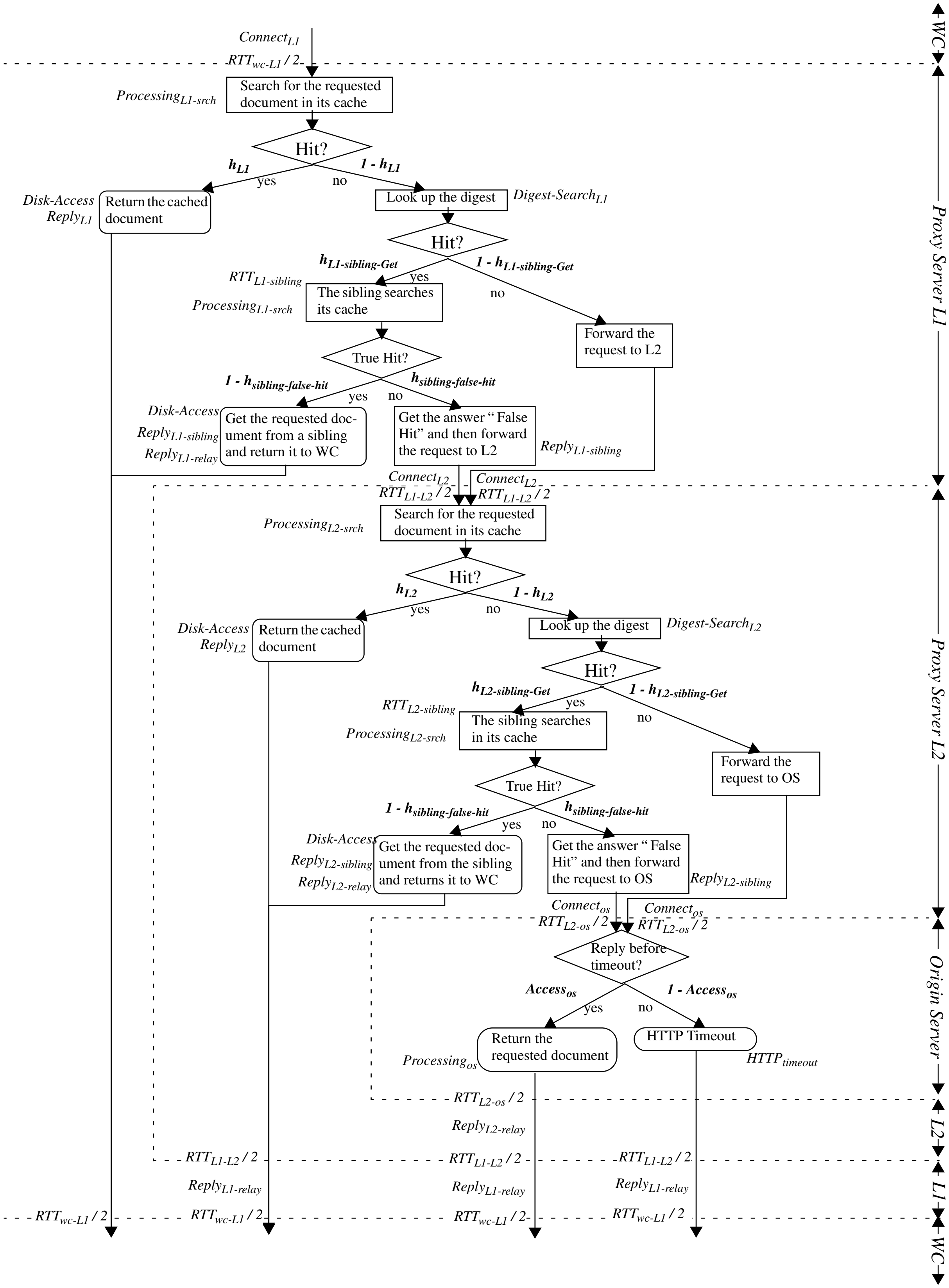


Figure 14: CD two-level hierarchy flowchart for the Get requests

$\text{Response time}_{Get} = \text{Connect}_{L1} + \text{RTT}_{wc\text{-}L1} + h_{L1} * \text{Processing}_{L1\text{-}hit} + (1 - h_{L1}) * (h_{L1\text{-}sibling\text{-}Get} * ((1 - h_{L1\text{-}sibling\text{-}false\text{-}hit}) * \text{CD Proxy Server L1-sibling True Hit Time}_{Get} + h_{L1\text{-}sibling\text{-}false\text{-}hit} * \text{CD Proxy Server L1-sibling False Hit Penalty}_{Get}) + (h_{L1\text{-}sibling\text{-}Get} * h_{L1\text{-}sibling\text{-}false\text{-}hit} + (1 - h_{L1\text{-}sibling\text{-}Get})) * (h_{L2} * \text{CD Proxy Server L2 Hit Time}_{Get} + (1 - h_{L2}) * (h_{L2\text{-}sibling\text{-}Get} * ((1 - h_{L2\text{-}sibling\text{-}false\text{-}hit}) * \text{CD Proxy Server L2-sibling True Hit Time}_{Get} + h_{L2\text{-}sibling\text{-}false\text{-}hit} * \text{CD Proxy Server L2-sibling False Hit Penalty}_{Get}) + (h_{L2\text{-}sibling\text{-}Get} * h_{L2\text{-}sibling\text{-}false\text{-}hit} + (1 - h_{L2\text{-}sibling\text{-}Get})) * (\text{Access}_{os} * \text{Origin Server Hit Time}_{Get} + (1 - \text{Access}_{os}) * \text{HTTP}_{timeout}))))$

$\text{CD Proxy Server L1-sibling True Hit Time}_{Get} = \text{Processing}_{L1\text{-}srch} + \text{Digest-Search}_{L1} + \text{RTT}_{L1\text{-}sibling} + \text{Processing}_{L1\text{-}hit} + \text{Reply}_{L1\text{-}relay}$

$\text{CD Proxy Server Lx-sibling False Hit Penalty}_{Get} = \text{RTT}_{Lx\text{-}sibling} + \text{Processing}_{Lx\text{-}srch} + \text{Reply}_{Lx\text{-}sibling}$ (x = 1 or 2)

$\text{CD Proxy Server L2 Hit Time}_{Get} = \text{Processing}_{L1\text{-}srch} + \text{Digest-Search}_{L1} + \text{Connect}_{L2} + \text{RTT}_{L1\text{-}L2} + \text{Processing}_{L2\text{-}hit} + \text{Reply}_{L1\text{-}relay}$

$\text{CD Proxy Server L2-sibling True Hit Time}_{Get} = \text{Processing}_{L1\text{-}srch} + \text{Digest-Search}_{L1} + \text{Connect}_{L2} + \text{RTT}_{L1\text{-}L2} + \text{Processing}_{L2\text{-}srch} + \text{Digest-Search}_{L2} + \text{RTT}_{L2\text{-}sibling} + \text{Processing}_{L2\text{-}hit} + \text{Reply}_{L2\text{-}relay} + \text{Reply}_{L1\text{-}relay}$

$\text{Origin Server Hit Time}_{Get} = \text{Processing}_{L1\text{-}srch} + \text{Digest-Search}_{L1} + \text{Connect}_{L2} + \text{RTT}_{L1\text{-}L2} + \text{Processing}_{L2\text{-}srch} + \text{Digest-Search}_{L2} + \text{Connect}_{os} + \text{RTT}_{L2\text{-}os} + \text{Processing}_{os} + \text{Reply}_{L2\text{-}relay} + \text{Reply}_{L1\text{-}relay}$

Figure 14-formulae: Formulae for CD two-level hierarchy for the Get requests

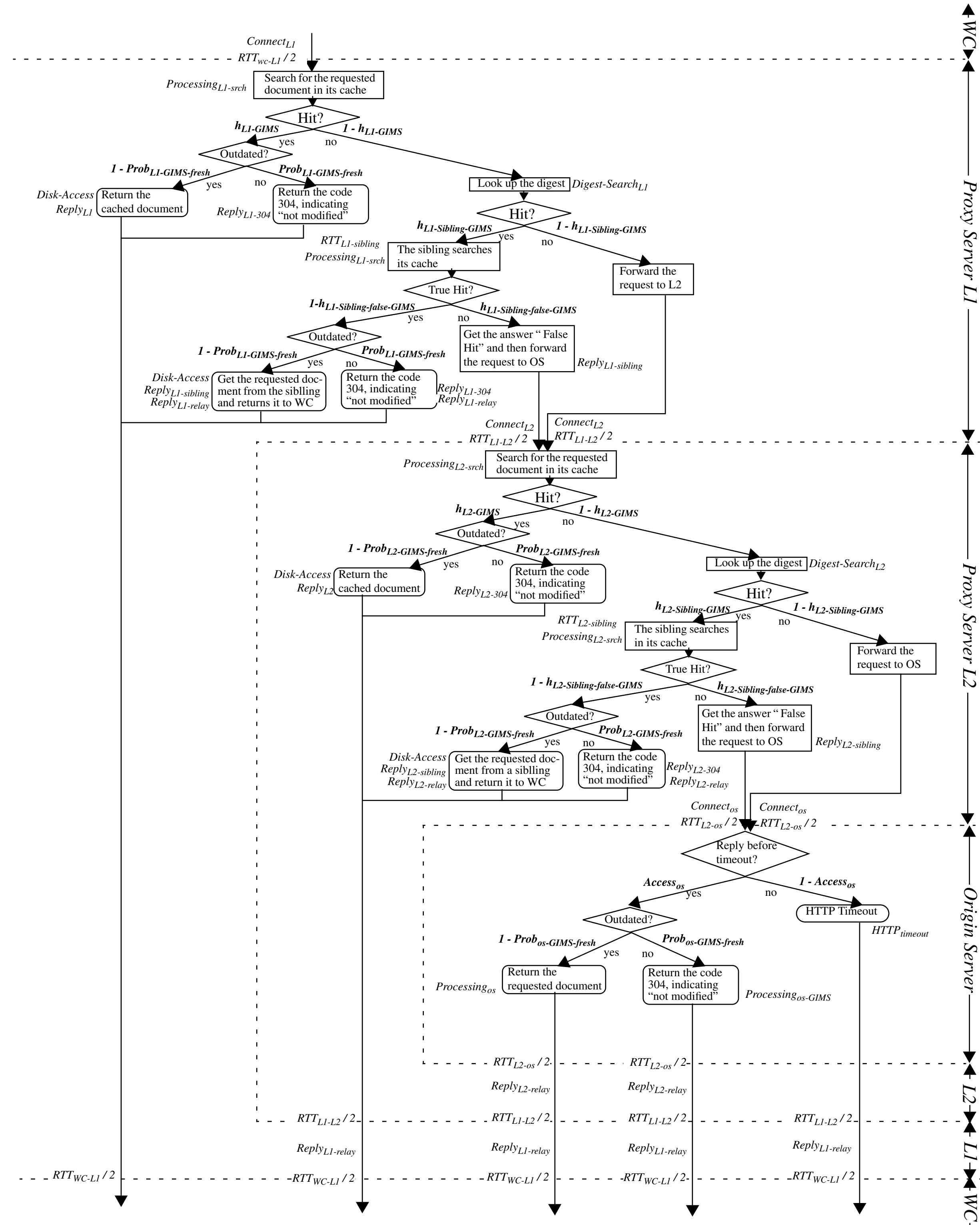


Figure 15: CD two-level hierarchy flowchart for the GIMS requests

Response time_{GIMS} = $\text{Connect}_{L1} + \text{RTT}_{wc\text{-}L1} + h_{L1\text{-}GIMS}$ * CD Proxy Server L1 Hit Time_{GIMS} + $(1 - h_{L1\text{-}GIMS})$ * $(h_{L1\text{-}sibling\text{-}GIMS}$ * $((1 - h_{L1\text{-}sibling\text{-}false\text{-}GIMS})$ * CD Proxy Server L1-sibling True Hit Time_{GIMS} + $h_{L1\text{-}sibling\text{-}false\text{-}GIMS}$ * CD Proxy Server L1-sibling False Hit PenaltyGIMS) + $(h_{L1\text{-}sibling\text{-}GIMS}$ * $h_{L1\text{-}sibling\text{-}false\text{-}hit} + (1 - h_{L1\text{-}sibling\text{-}GIMS}))$ * ($h_{L2\text{-}GIMS}$ * CD Proxy Server L2 Hit Time_{GIMS} + $(1 - h_{L2\text{-}GIMS})$ * $(h_{L2\text{-}sibling\text{-}GIMS}$ * $((1 - h_{L2\text{-}sibling\text{-}false\text{-}GIMS})$ * CD Proxy Server L2-sibling True Hit Time_{GIMS} + $h_{L2\text{-}sibling\text{-}false\text{-}GIMS}$ * CD Proxy Server L2-sibling False Hit PenaltyGIMS) + $(h_{L2\text{-}sibling\text{-}GIMS}$ * $h_{L2\text{-}sibling\text{-}false\text{-}GIMS} + (1 - h_{L2\text{-}sibling\text{-}GIMS}))$ * $(\text{Access}_{os}$ * Origin Server Hit Time_{GIMS} + $(1 - \text{Access}_{os})$ * $\text{HTTP}_{timeout}))))$

CD Proxy Server L1 Hit Time_{GIMS} = $\text{Prob}_{L1\text{-}GIMS\text{-}fresh}$ * $(\text{Processing}_{L1\text{-}srch} + \text{Reply}_{L1\text{-}304}) + (1 - \text{Prob}_{L1\text{-}GIMS\text{-}fresh})$ * $\text{Processing}_{L1\text{-}hit}$

CD Proxy Server L1-sibling True Hit Time_{GIMS} = $\text{Processing}_{L1\text{-}srch}$ + Digest-$\text{Search}_{L1} + \text{RTT}_{L1\text{-}sibling} + \text{Prob}_{L1\text{-}GIMS\text{-}fresh}$ * $(\text{Processing}_{L1\text{-}srch} + \text{Reply}_{L1}) + (1 - \text{Prob}_{L1\text{-}GIMS\text{-}fresh})$ * $\text{Processing}_{L1\text{-}hit} + \text{Reply}_{L1\text{-}relay}$

CD Proxy Server Lx-sibling False Hit Penalty_{GIMS} = $\text{RTT}_{L1\text{-}sibling} + \text{Processing}_{L1\text{-}srch} + \text{Reply}_{L1\text{-}sibling}$ (x = 1 or 2)

CD Proxy Server L2 Hit Time_{GIMS} = $\text{Processing}_{L1\text{-}srch}$ + Digest-$\text{Search}_{L1} + \text{Connect}_{L2} + \text{RTT}_{L1\text{-}L2} + \text{Prob}_{L2\text{-}GIMS\text{-}fresh}$ * $(\text{Processing}_{L2\text{-}srch} + \text{Reply}_{L2\text{-}304}) + (1 - \text{Prob}_{L2\text{-}GIMS\text{-}fresh})$ * $\text{Processing}_{L2\text{-}hit} + \text{Reply}_{L2\text{-}relay} + \text{Reply}_{L1\text{-}relay}$

CD Proxy Server L2-sibling True Hit Time_{GIMS} = $\text{Processing}_{L1\text{-}srch}$ + Digest-$\text{Search}_{L1} + \text{Connect}_{L2} + \text{RTT}_{L1\text{-}L2} + \text{Processing}_{L2\text{-}srch}$ + Digest-$\text{Search}_{L2} + \text{RTT}_{L2\text{-}sibling} + \text{Prob}_{L2\text{-}GIMS\text{-}fresh}$ * $(\text{Processing}_{L2\text{-}srch} + \text{Reply}_{L2\text{-}relay}) + (1 - \text{Prob}_{L2\text{-}GIMS\text{-}fresh})$ * $\text{Processing}_{L2\text{-}hit} + \text{Reply}_{L2\text{-}relay} + \text{Reply}_{L1\text{-}relay}$

Origin Server Hit Time_{GIMS} = $\text{Processing}_{L1\text{-}srch}$ + Digest-$\text{Search}_{L1} + \text{Connect}_{L2} + \text{RTT}_{L1\text{-}L2} + \text{Processing}_{L2\text{-}srch}$ + Digest-$\text{Search}_{L2} + \text{Connect}_{os} + \text{RTT}_{L2\text{-}os} + \text{Prob}_{os\text{-}GIMS\text{-}fresh}$ * $\text{Processing}_{os\text{-}GIMS} + (1 - \text{Prob}_{os\text{-}GIMS\text{-}fresh})$ * $\text{Processing}_{os} + \text{Reply}_{L2\text{-}relay} + \text{Reply}_{L1\text{-}relay}$

Figure 15-formula: Formula for CD two-level hierarchy for the GIMS requests

4) CNP

CNP can be regarded as a two-level dynamic caching hierarchy schemes. A request has to be processed by an authoritative source -- either the origin server or its *C-Reps* -- to receive a response that warrants the currency of the returned document. As a client-side proxy server receives a request for which it is not entitled to respond, it has to forward the request to the closest authoritative source for the document. Fig. 4 depicts such a scenario. The request forwarding paths were numbered to facilitate tracking down the corresponding formulae to the right of the figure. The vague part in the origin server section is used to indicate the processings for the requests received directly from the average proxy servers. Note that the figure does not layout the scenario in case the client-side proxy server happens to be a *C-Rep*.

Table 3: Terminologies for CNP

Terms / Values	Descriptions	Source / Misc.
$CNP_{os\text{-}Get}$ *250* [ms]	The time it takes an origin server between receiving a Get request from its *C-Rep* and returning the last byte of the requested document	In CNP, the requests from C-Reps have a higher priority to be processed by the originserver compared to the requests from other proxy servers. The process queueing time is thus reduced
$CNP_{os\text{-}GIMS}$ *150* [ms]	The time it takes an origin server between receiving a Get-If-Modified-Since request from its *C-Rep* and returning the last byte of the requested document	
$Connect_{CSP}$ *30* [ms]	The time it takes a client-side proxy server between accepting a HTTP connection setup request and the actual receiving of the entire request	$= Connect_{L1}$
$h_{CRep\text{-}Get}$ *{90, 95}* [%]	The hit rate of a *C-Rep* processing a Get request for the document provided by the origin server that it represents	See Section IV
$h_{CRep\text{-}GIMS}$ *{90, 95}* [%]	The hit rate of a *C-Rep* processing a GIMS request for the document provided by the origin server that it represents	Same reason as $h_{CRep\text{-}Get}$
$h_{CSP\text{-}Get}$ *25* [%]	The hit rate of a Client-Side Proxy server processing a Get request	$= h_{L1}$
$h_{CSP\text{-}GIMS}$ *25* [%]	The hit rate of a Client-Side Proxy server processing a GIMS request	$= h_{L1\text{-}GIMS} = h_{L1}$
$Prob_{CNP\text{-}GIMS\text{-}fresh}$ *{10, 50}* [%]	Positive confirmation rate of the Get-IMS requests for an origin server if the requests are received from one of its *C-Reps*	see Section IV
$Prob_{CSP\text{-}Get}$ *100* [%]	The probability of a Client-Side Proxy server making a Get request on behalf of the requesting Web client	We focus on only the Get requests and ignore the Get-IMS requests in this paper
$Prob_{Get\text{-}CRep}$ *{50, 80}* [%]	The probability of a Client-Side Proxy server sending a Get request to a *C-Rep* of the origin server for the document	see Section IV
$Prob_{GIMS\text{-}CRep}$ *{50, 80}* [%]	The probability of a Client-Side Proxy server sending a GIMS request to a *C-Rep* of the origin server for the document	Same as $Prob_{Get\text{-}CRep}$
$Processing_{CSP\text{-}srch}$ *180* [ms]	The time it takes a Client-Side Proxy server to search its caching database in memory	$= Processing_{L1\text{-}srch}$
$Processing_{CRep\text{-}hit}$	The time it takes a *C-Rep* between receiving a HTTP connect request and returning the last byte of the requested document	$= Processing_{CRep\text{-}srch} + Disk\text{-}Access + Reply_{CRep}$
$Processing_{CRep\text{-}srch}$ *250* [ms]	The time it takes a *C-Rep* to search its caching database in memory	$= Processing_{L2\text{-}srch}$

Table 3: Terminologies for CNP

Terms / Values	Descriptions	Source / Misc.
$Processing_{os\text{-}CNP\text{-}Get}$	The time it takes an origin server to process a Get request from someone other than its *C-Reps*	= $Processing_{os}$
500 [ms]		
$Processing_{os\text{-}CNP\text{-}GIMS}$	The time it takes an origin server to process a Get-IMS request from someone other than its caching representatives	= $Processing_{os\text{-}GIMS}$
350 [ms]		
$Reply_{x\text{-}relay}$	The time it takes *x* to relay a repsonse to the requesting party, x is CSP or CRep.	
50 [ms]	$Reply_{CSP\text{-}relay} = Reply_{CRep\text{-}relay} = 50$[ms]	= $Reply_{Lx\text{-}relay}$
$RTT_{A\text{-}B}$	Round-trip turnaround time between A and B.	$RTT_{CSP\text{-}CRep} + RTT_{CRep\text{-}os}$ should be > $RTT_{CSP\text{-}os}$, but not much. The reason is that we assume that the *C-Rep* should be in about the same direction of the origin server for the client-side proxy server.
	$RTT_{wc\text{-}CSP} = RTT_{wc\text{-}L1} = 20$[ms]; $RTT_{CSP\text{-}os} = RTT_{Lx\text{-}os} = 240$[ms]; $RTT_{CRep\text{-}os} = 200[ms]^{*}$; $RTT_{CSP\text{-}CRep}$: $100[ms]^{*}$; (*: see Source / Misc.)	

In Fig. 4, as the client-side proxy server receives a request for which it is not a *C-Rep*, a *Get-IMS* request will be made even if the cache has a copy of the document. Otherwise, a *Get* request will be sent to either a *C-Rep* or the origin server, depending on the estimated proximity. In case the request is being forwarded to a *C-Rep*, the *C-Rep* will check if it does have the needed document. It also needs to verify if the document is still considered current before returning the document to the request forwarding party. Because invalidation is used in *CNP* and most of the popular documents on the origin server are cached, the probability that a *C-Rep* can return a current document without contacting the origin server is supposed to be high as assumed.

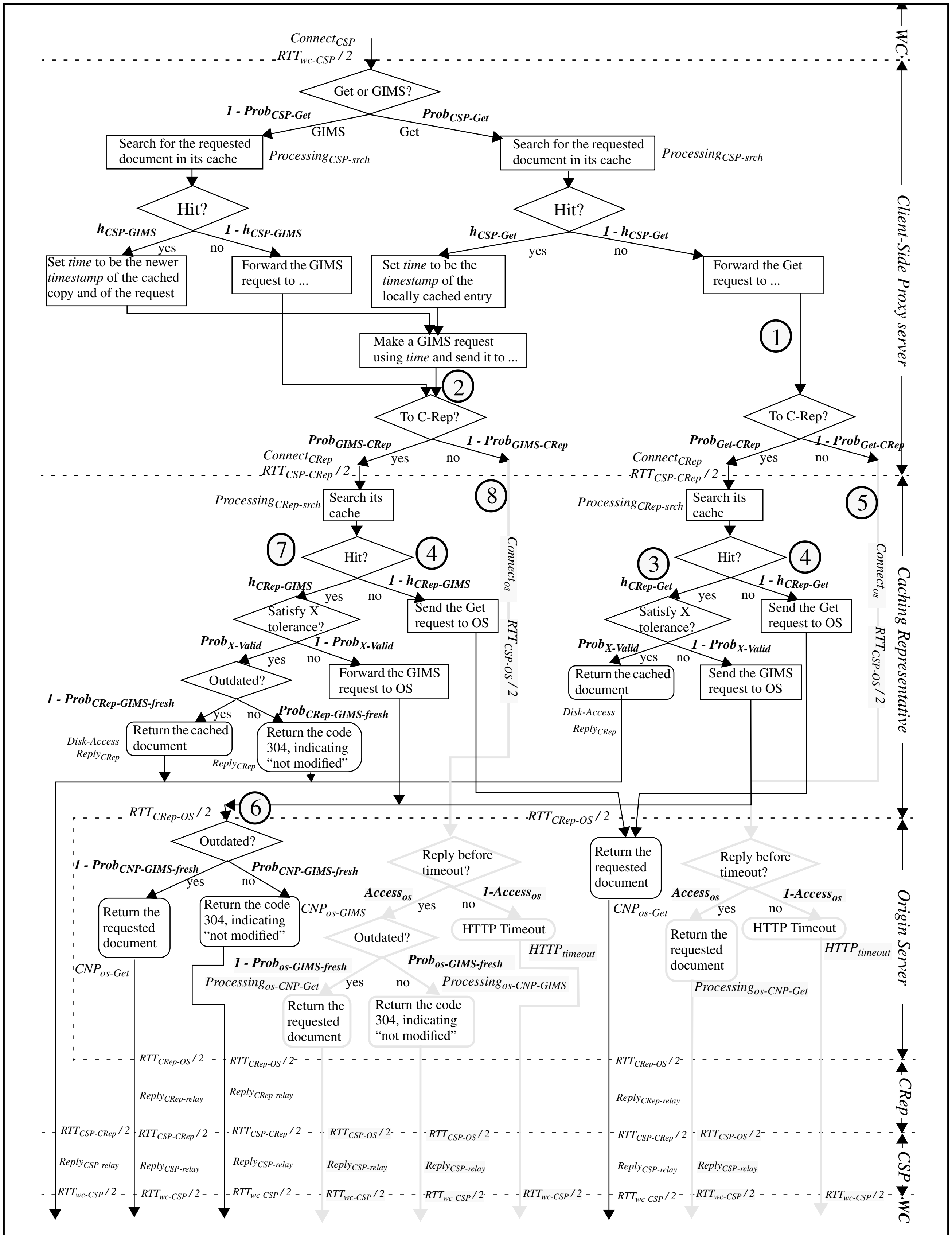


Figure 16: CNP : Client-Side Proxy server is not a Caching Representative

Response time = $Connect_{CSP} + RTT_{wc\text{-}CSP} + Processing_{CSP\text{-}srch} + Prob_{CSP\text{-}Get} * (1 - h_{CSP}) * {}^{\{1\}}$Client-Side Proxy Server Request $Time_{Get} + (Prob_{CSP\text{-}Get} * h_{CSP} + (1 - Prob_{CSP\text{-}Get})) * {}^{\{2\}}$Client-Side Proxy Server Request $Time_{GIMS} + Reply_{CSP\text{-}relay}$

${}^{\{1\}}$Client-Side Proxy Server Request $Time_{Get} = Prob_{Get\text{-}CRep} * (Connect_{CRep} + RTT_{CSP\text{-}CRep} + h_{CRep\text{-}Get} * {}^{\{3\}}$C-Rep CNP Hit $Time_{Get} + (1 - h_{CRep\text{-}Get}) * {}^{\{4\}}$Origin Server through C-Rep Hit $Time_{Get}) + (1 - Prob_{Get\text{-}CRep}) * {}^{\{5\}}$Origin Server Direct Hit $Time_{Get}$

${}^{\{2\}}$Client-Side Proxy Server Request $Time_{GIMS} = Prob_{GIMS\text{-}CRep} * (Connect_{CRep} + RTT_{CSP\text{-}CRep} + h_{CRep\text{-}GIMS} * {}^{\{7\}}$C-Rep CNP Hit $Time_{GIMS} + (1 - h_{CRep\text{-}GIMS}) * {}^{\{4\}}$Origin Server through C-Rep Hit $Time_{Get}) + (1 - Prob_{GIMS\text{-}CRep}) * {}^{\{8\}}$Origin Server Direct Hit $Time_{GIMS}$

${}^{\{3\}}$C-Rep CNP Hit $Time_{Get} = Prob_{X\text{-}valid} * Processing_{CRep\text{-}hit} + (1 - Prob_{X\text{-}valid}) * {}^{\{6\}}$Origin Server through C-Rep Hit $Time_{GIMS}$

${}^{\{4\}}$Origin Server through C-Rep Hit $Time_{Get} = Processing_{CRep\text{-}srch} + RTT_{CRep\text{-}os} + CNP_{os\text{-}Get} + Reply_{CRep\text{-}relay}$

${}^{\{5\}}$Origin Server Direct Hit $Time_{Get} = Connect_{os} + RTT_{CSP\text{-}os} + Access_{os} * Processing_{os\text{-}CNP} + (1 - Access_{os}) * HTTP_{timeout}$

${}^{\{6\}}$Origin Server through C-Rep Hit $Time_{GIMS} = Processing_{CRep\text{-}srch} + RTT_{CRep\text{-}os} + Prob_{CNP\text{-}GIMS\text{-}fresh} * CNP_{os\text{-}GIMS} + (1 - Prob_{CNP\text{-}GIMS\text{-}fresh}) * CNP_{os\text{-}Get} + Reply_{CRep\text{-}relay}$

${}^{\{7\}}$C-Rep CNP Hit $Time_{GIMS} = Prob_{X\text{-}valid} * ((1 - Prob_{CRep\text{-}GIMS\text{-}fresh}) * Processing_{CRep\text{-}hit} + Prob_{CRep\text{-}GIMS\text{-}fresh} * (Processing_{CRep\text{-}srch} + Reply_{CRep})) + (1 - Prob_{X\text{-}valid}) * {}^{\{6\}}$Origin Server CNP Processing $Time_{GIMS})$

${}^{\{8\}}$Origin Server Direct Hit $Time_{GIMS} = Connect_{os} + RTT_{CSP\text{-}os} + Access_{os} * ((1 - Prob_{os\text{-}GIMS\text{-}fresh}) * Processing_{os} + Prob_{os\text{-}GIMS\text{-}fresh} * Processing_{os\text{-}GIMS}) + (1 - Access_{os}) * HTTP_{timeout}$

Figure 16-formulae: Formulae for CNP as Client-Side Proxy server is not a Caching Representative

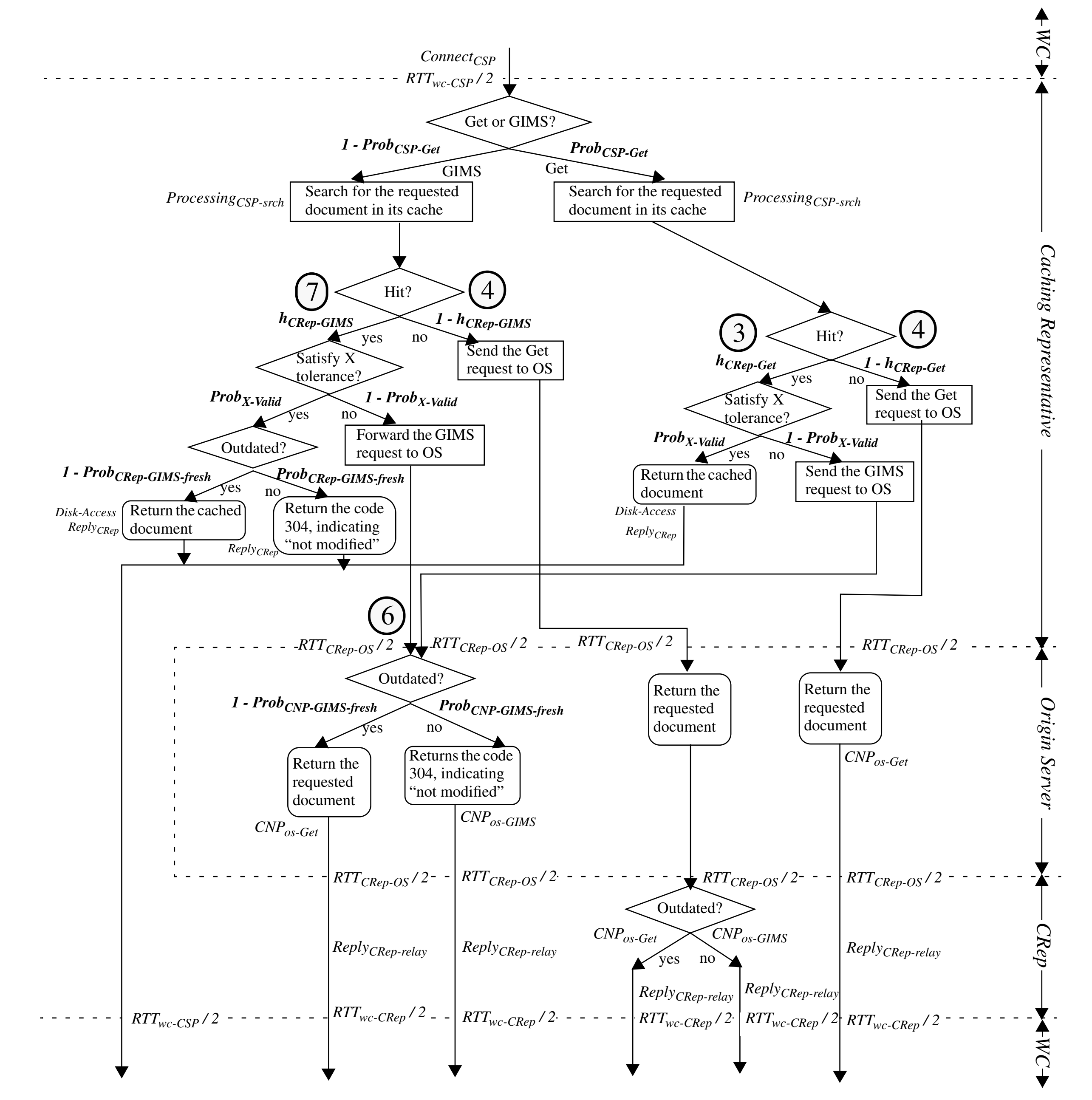


Figure 17: CNP : Client-Side Proxy server is a Caching Representative

Response time = $\text{Connect}_{CSP} + \text{RTT}_{wc\text{-}CSP} + \text{Prob}_{CSP\text{-}Get} * (h_{CRep\text{-}GIMS} * {}^{\{7\}}\text{C-Rep CNP Hit Time}_{GIMS} + (1 - h_{CRep\text{-}GIMS}) * {}^{\{4\}}\text{Origin Server through C-Rep Hit Time}_{Get}) + (1 - \text{Prob}_{CSP\text{-}Get)} * (h_{CRep\text{-}Get} * {}^{\{3\}}\text{C-Rep CNP Hit Time}_{Get} + (1 - h_{CRep\text{-}Get}) * {}^{\{4\}}\text{Origin Server through C-Rep Hit Time}_{Get})$

${}^{\{3\}}\text{C-Rep CNP Hit Time}_{Get} = \text{Prob}_{X\text{-}valid} * \text{Processing}_{CRep\text{-}hit} + (1 - \text{Prob}_{X\text{-}valid}) * {}^{\{6\}}\text{Origin Server through C-Rep Hit Time}_{GIMS}$

${}^{\{4\}}\text{Origin Server through C-Rep Hit Time}_{Get} = \text{Processing}_{CRep\text{-}srch} + \text{RTT}_{CRep\text{-}os} + \text{CNP}_{os\text{-}Get} + \text{Reply}_{CRep\text{-}relay}$

${}^{\{6\}}\text{Origin Server through C-Rep Hit Time}_{GIMS} = \text{Processing}_{CRep\text{-}srch} + \text{RTT}_{CRep\text{-}os} + \text{Prob}_{CNP\text{-}GIMS\text{-}fresh} * \text{CNP}_{os\text{-}GIMS} + (1 - \text{Prob}_{CNP\text{-}GIMS\text{-}fresh}) * \text{CNP}_{os\text{-}Get} + \text{Reply}_{CRep\text{-}relay}$

${}^{\{7\}}\text{C-Rep CNP Hit Time}_{GIMS} = \text{Prob}_{X\text{-}valid} * ((1 - \text{Prob}_{CRep\text{-}GIMS\text{-}fresh)} * \text{Processing}_{CRep\text{-}hit} + \text{Prob}_{CRep\text{-}GIMS\text{-}fresh} * (\text{Processing}_{CRep\text{-}srch} + \text{Reply}_{CRep})) + (1 - \text{Prob}_{X\text{-}valid}) * {}^{\{6\}}\text{Origin Server CNP Processing Time}_{GIMS})$

Figure 17-formulae: Formulae for CNP as Client-Side Proxy server is a Caching Representative

IV. SIMULATION AND ANALYSIS

To use the modeling results for the simulation, we made some general assumptions in the following aspects: *(i)* Selecting values for the parameters, *(ii)* Major parameters to be controlled and studied, and *(iii)* Miscellaneous assumptions about the simulation setup.

(i) Most of the values we selected for parameters were obtained from summarizing the research in [17]. Based on the previous assumption about how the caching hierarchies were built and the recommended proxy server setup guidelines in [28], we tuned the values slightly as appropriate. Hence, the data missing from the collected statistics have to be presumed from the known values. For example, $RTT\text{-}Hit_{Lx\text{-}sibling}$ is a parameter that we can obtain estimated values from [18]. However, they are the statistics for the proxy servers on the top level of the hierarchy with high bandwidth Internet backbone connections. By taking all the factors into account, we estimated the values for $RTT\text{-}Hit_{Lx\text{-}sibling}$.

(ii) The parameters of interest were the traffic load, the server demand, the server acceessibility, and the number of levels in the caching hierarchies. For traffic load and server demand, we estimated the ratio of the value at 90 percentile to the value of the median of the distribution to represent the heavy load and high demand. The ratio of the value at 10 percentile to the median of the distribution was used to represent the light load and low demand. According to the estimated values of light (low), median, and heavy (high) load/demand summarized in Table 4, we selected 50%, 100%, and 500% of the median values in the experiments to approximate the values for the heavy/high, average, and light/low loads/demands. Because there is a big gap between the median load/demand and the heavy load/demand, we inserted the ratio 300% to observe the performance variation closely. The values of the

Table 4: Ratio of the value of the median of the distribution

percentile	$Processing_{LX\text{-}srch}$			$Connect_{os}$	$Connect_{LX}$			$Reply_{LX}$			$RTT_{A\text{-}B}$
	L1	L2	L3		L1	L2	L3	L1	L2	L3	
90	2.67	5.80	5.00	5.71	1.50	3.50	5.00	1.17	3.00	3.33	1.17
50	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
10	0.17	0.20	0.20	0.29	0.33	0.40	0.35	0.25	0.33	0.47	0.17

parameters that in nature are server demand oriented or traffic load oriented are adjusted in proportion as the load/demand changes. The server accessibility is introduced in the parameter *Access_os* to investigate how the response times will change as an origin server is not able to process all the requests before timeouts while experiencing high demand. We assumed that the majority of the requests would be fulfilled and only 90%, 95%, and 100% were used as the testing values in the study.

(iii) We assume that the proxy servers in the hierarchy are always available. The demands on the proxy servers and the origin servers are independent. One of the most important assumptions that we made was based on [1], which found that about 33% of the accesses were made to the documents which were only accessed once during a five-day period, let alone the fact that the first access to a document would not get a hit. Nevertheless, we used 70% as the ultimately attainable hit rate for simple hierarchy, *ICP*, and *CD* by being optimistic. Besides, the current HTTP standard states that 45 seconds is the default timeout value [5]. However, Web users are inclined to interrupt the connections if the document transmissions are slower than expected. Therefore, 15 seconds was adopted as the primary test value which assumed the median of the actual timeout values after taking the human factor into account. Such a value will affect the response times only as the value of *Access_os* is less than 100%. The configurations used to study the performance for simple hierarchy, ICP, and CD are given in Table 5.

TABLE 5 Configurations used for the study

<table>
<tr><th>Configurations</th><th>Timeout (sec)</th><th>Collective Hit rate*</th><th>Hit rates for L1 - L4 (%)
{Lx(Lx$_{sibling}$), ...}</th></tr>
<tr><td>Simple-1</td><td>15</td><td>54%</td><td>{25%, 15%, 10%, 20%}</td></tr>
<tr><td>Simple-2</td><td>15</td><td>64%</td><td>{25%, 15%, 20%, 30%}</td></tr>
<tr><td>Simple-3</td><td>15</td><td>64%</td><td>{30%, 25%, 20%, 15%}</td></tr>
<tr><td>ICP-1</td><td>15</td><td rowspan="2">44(41)%,
59.5(57.3)%,
63(61)%,
69.3(67.3)%</td><td rowspan="2">{25%(25%)},
{25%(10%), 25%(20%)},
{25%(10%), 15%(5%), 20%(15%)}
{25%(10%), 15%(5%), 10%(5%), 20%(15%)}</td></tr>
<tr><td>CD-1</td><td>15</td></tr>
<tr><td>ICP-2</td><td>15</td><td rowspan="2">44.8(43.3)%
60(57.5)%
63.7(61.3)%
69(66.5)%</td><td rowspan="2">{35%(15%)},
{35%(15%), 15%(15%)},
{35%(15%), 10%(10%), 5%(5%)}
{35%(15%), 10%(10%), 5%(5%), 5%(10%)}</td></tr>
<tr><td>CD-2</td><td>15</td></tr>
<tr><td>CD-3</td><td>45</td><td colspan="2">same as ICP-1 and CD-1</td></tr>
</table>

*The collective hit rate indicates the percentage of requests that can be satisfied by a cache in the hierarchy. We believe that the sibling selection criteria hinges on the number of levels used in the caching hierarchy. That explains why we have four different settings for {L1}, {L1, L2}, {L1, L2, L3}, and {L1, L2, L3, L4} respectively in two cells of the last column.

A. Simple Hierarchy

We use *Simple-1* as an illustrative configuration because both *Simple-2* and *Simple-3* possessed similar characteristics (see APPENDIX). The rational to set the hit rates for L1 to L3 as decreasing was that the frequently accessed documents tend to generate hits in the lower levels of the hierarchy, as confirmed in [28]. Except *Simple-3*, the hit rate for L4 was set higher than L3 because we supposed that the top-level proxy servers might contain more documents like what are currently deployed by *NLANR* [29].

Based on the assumption that the proxy servers at all hierarchies are operated at the median response times, the findings are summarized as follows.

1. Hierarchy does not improve the response times for all combinations of the controlled parameters, as shown in Fig. 18a, Fig. 18b, and Fig. 18c.
2. In most figures the cases using the light traffic loads are skipped because their performances are close to those of the cases using the average traffic loads, as demonstrated in Fig. 18b.
3. As the number of levels in the hierarchy increases, the response time is more sensitive to the traffic load, as shown in Fig. 18b.
4. When the traffic load is light or average, the number of levels in the hierarchy has little impact on the response times regardless of the other parameters. (Fig. 18c)
5. Under the same server demand, the variance of the response times is reduced with respect to *Access_os* as the number of levels in the hierarchy increases, as can be seen in Fig. 18c. The implication is that the deeper hierarchy will reduce the response time if the accessibility of the origin server is not 100%.

We were surprised that most of the characteristics stated above, except the first one, were also valid for *ICP* and *CD*. The reasonable explanation is that both *ICP* and *CD* do not deviate from the simple hierarchy scheme in practice. They enhance the hit rate of the proxy servers at a certain level in the hierarchy with assumed low cost. Therefore, we will only discuss the characteristics specific to *ICP* and *CD* in the next section.

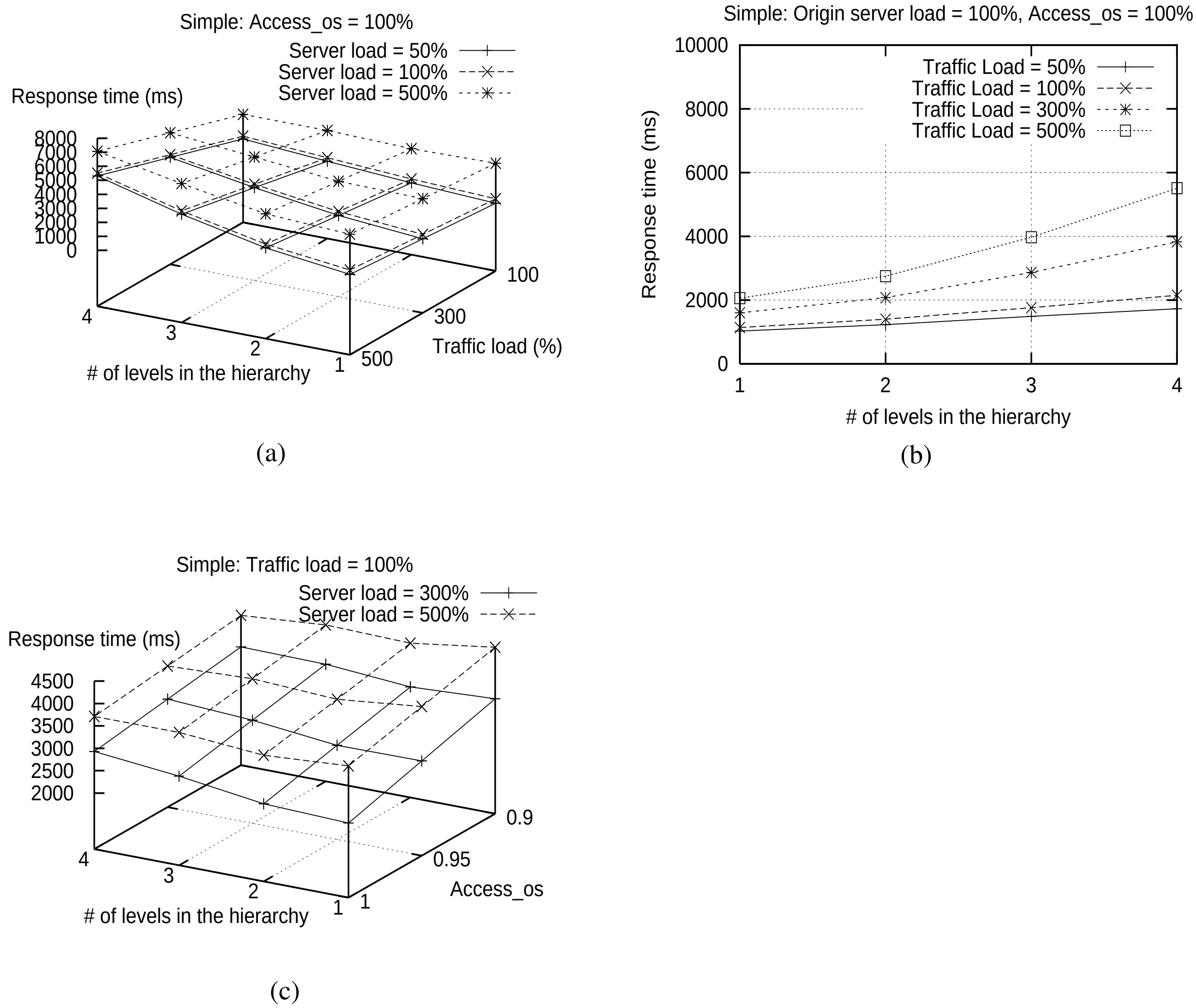


Figure 18: The simulation outcome for the simple Web caching hierarchy scheme

B. ICP and CD

Since *ICP* is the predecessor of *CD*, we combine the simulation results for both of them for easy comparisons. Several configurations listed in Table 5 were set to study the performance differences within an approach and between the approaches. For the columns listing the collective hit rates, the numbers enclosed in parentheses are the actual hit rates for *CD* after taking the false hit ratio into account. For *CD* alone, we adopted 15 and 45 seconds as the values for HTTP timeout to evaluate the impact on the response times. The selected experimental statistics are summarized in Table 6.

i) A comparison of the three approaches discussed this far can be drawn from Fig. 18b, Fig. 19a, and Fig. 19b. When the traffic load is light or average, the performances are close for all. Nevertheless, as the traffic load becomes heavy, the simple hierarchy surprisingly outperforms *ICP*, though *CD* is still the best of the three. Fig. 19c takes a closer look at the performance difference between *ICP* and *CD* under high server demand and various traffic loads.

ii) An interesting phenomenon is shown in Fig. 19d. We found that the two-level hierarchy configuration outperformed all the others (also observable in Fig. 19e). Such a feature is especially obvious when the HTTP timeout value is set to 45 seconds. *Access_os* is not a factor in the case where HTTP timeout is set to 15 seconds, but plays an important role in affecting the performance where HTTP timeout value equals 45 seconds.

iii) We also examine the effect of the number of levels in the hierarchy for all combinations of server demands and traffic loads in Fig. 19f. It shows that one-level and two-level hierarchies turn in very similar response times while the four-level case adversely affects the response times. Such an effect is more prominent as the traffic load increases.

TABLE 6 Response times for Simple hierarchy, ICP, & CD (Access_os = 1)

Server demand	**Avg**		**Avg**		**Hvy**		**Hvy**	
Traffic load	**Avg**	**%***	**Hvy**	**%***	**Avg**	**%***	**Hvy**	**%***
Simple-1 (2-level)	1399	100	2751	100	3567	100	4919	100
ICP-1 (2-level)	1376	98	3149	114	2753	77	4526	92
ICP-2 (2-level)	1287	92	2894	105	2644	74	4251	86
CD-1 (2-level)	1309	94	2477	90	2759	77	3927	80
CD-2 (2-level)	1230	88	2301	84	2657	74	3729	76
CD-3 + (2-level)	2228	N/A	3360	N/A	3605	101	4738	96
Simple-1 (4-level)	2149	154	5510	200	3710	104	7070	144
ICP-1 (4-level)	2255	161	6547	238	3332	93	7625	155
CD-1 (4-level)	2059	147	5016	182	3199	90	6156	125

*The percentage calculation is relative to the values for Simple-1.

+We assume that the server will not be always available only when it is under high demand. The values provided are based on Access_os = 0.95. Otherwise, the values will be identical to those of CD-1.

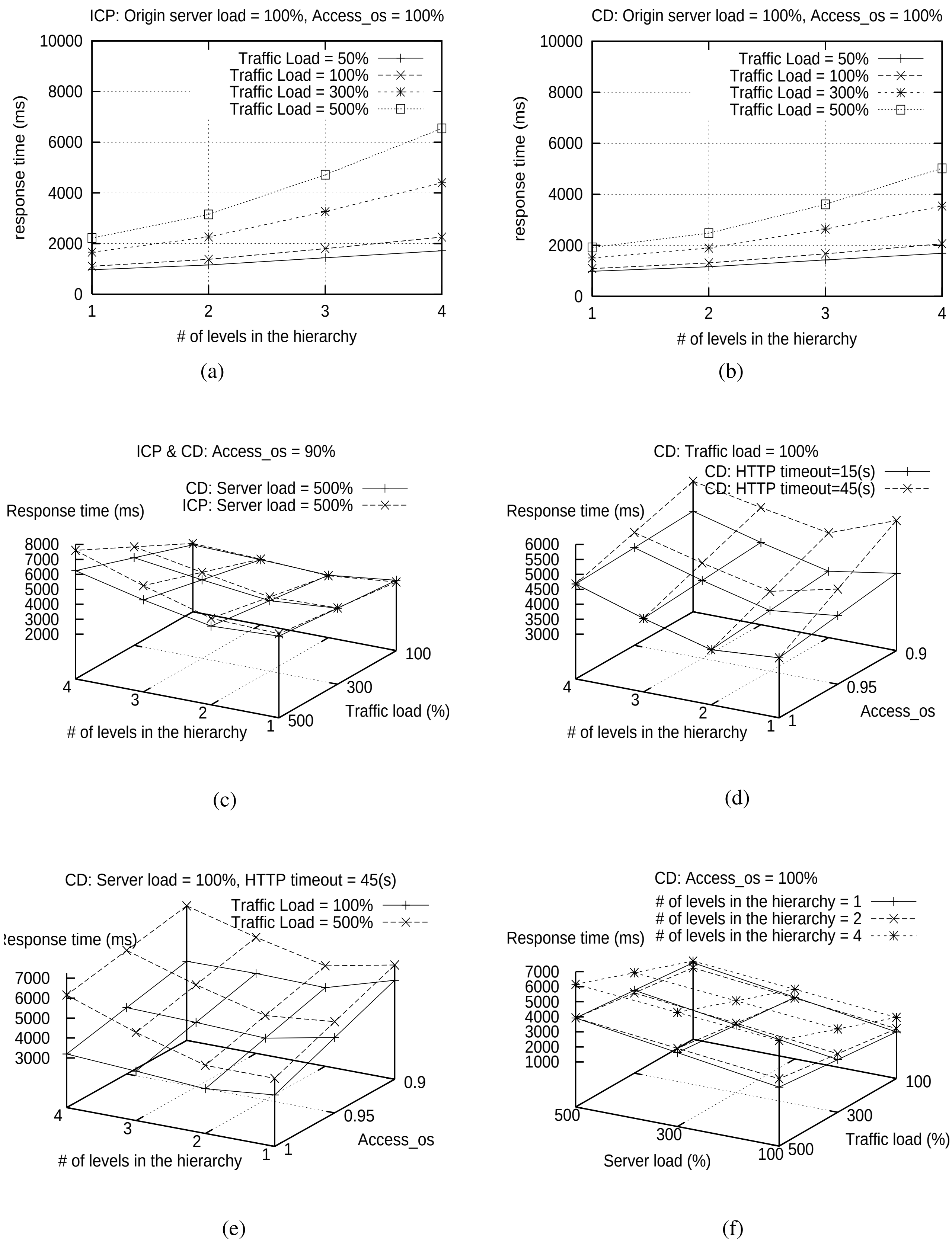


Figure 19: The simulation outcome for ICP and CD Web caching hierarchy scheme

C. CNP

Since the employment of *CNP* requires a large-scale testbed and it is not available yet, we tried to carefully select values for the parameters from the available statistics of the proxy servers and origin servers. In the following, we will first discuss how the values for some key parameters were chosen for *CNP* and then present the results of the simulation.

1) Selecting values for the parameters used for CNP

First, we set $Prob_{CSP\text{-}Get}$ to 1 to exclude the *Get-IMS* requests and focus only on the *Get* requests. The parameter $h_{CSP\text{-}Get}$ is set to 0.25 since this value was used as the hit rate for the first level proxy server in the other approaches. The parameter $Prob_{Get\text{-}CRep}$ decides the percentage of the documents that will be forwarded to a *C-Rep* rather than to the origin server. For the simulation purpose, we assume that *CNP* is widely accepted by most of the frequently accessed origin servers in the *Web* and the request forwarding information is properly propagated as in the case of routing information in the Internet. Thus, the range {0.5, 0.8} is selected for $Prob_{Get\text{-}CRep}$. The upperbound 0.8 was chosen primarily because the most frequently accessed 10% of the origin servers alone have accounted for up to 83% of the accesses [1]. The lower bound is just a pessimistic estimate for the value. That's because in *CNP*, the origin server can piggyback the representing status of its caching neighborhood information to the proxy servers in case the requests it receives are not from its *C-Reps*. Note that the values are chosen conservatively in order to be impartial with respect to *CNP*.

Due to the argument stated above, we also need to select a critical value for $Prob_{CSP\text{-}is\text{-}CRep}$ (on the top of Fig. 16, Fig. 17). This parameter dictates the probability that a client-side proxy server happens to be a *C-Rep* for a request. We learned in [1] that if a client-side proxy server opts to represent only the top 5% of the most frequently accessed origin servers for the local Web community it services, more than 75% of the requests can be handled by the proxy server without being forwarded. However, the range for $Prob_{CSP\text{-}is\text{-}CRep}$ was set as {0.05, 0.25}. The reason for setting it in such a low range was based on the philosophy that the origin server would not over-expand its caching neighborhood to overburden itself. If a proxy server was granted the privilege of being a *C-Rep* in a certain region, no other proxy server in the vicinity would be included as a caching neighborhood member. Such a design philosophy ensures that the origin servers will not be overloaded.

Once a request reaches the *C-Rep*, the first parameter it encounters is $h_{CRep\text{-}Get}$. The range {0.9, 0.95} for $h_{CRep\text{-}Get}$ is chosen because of the observation about the concentration of Web accesses mentioned at the beginning of Section II and the assumption that a *C-Rep* will cache at least the top 20% of the most frequently accessed documents of the origin servers it represents.

Even in case the needed document is in the cache, the *C-Rep* still needs to validate if the document conforms to the *X-tolerance coherence requirement* imposed by the origin server of the caching neighborhood. Since the invalidation is used in the caching neighborhood maintenance, the value of $Prob_{X\text{-}Valid}$ is set to 0.9.

The last parameter that is worthy of consideration is $Prob_{CRep\text{-}GIMS\text{-}fresh}$, which is used in the computation for the case where the client-side proxy server has the document in its cache but has to send a *Get-IMS* request to the *C-Rep*. The parameter represents the probability that the cached document at the client-side proxy server is still current. Not having enough information for a conjecture, the range for $Prob_{CNP\text{-}GIMS\text{-}fresh}$ is assumed to be {0.1, 0.5} to accomodate most possibilities.

In addition to the parameters discussed above, we also set *Access_os* to 1 for the processing routes passing through the *C-Reps* because the *C-Reps* learn if the origin server is available via the regular invalidation messages from the origin server. We also believe that the origin servers would not be overloaded as much as the other schemes are because the demands are disseminated to the *C-Reps*. Hence, the highest server demand that we think an origin server could experience may be well below 500%. However, we still use 500% as the possible high demand in the experiments for fair comparisons with the other schemes. Persistent HTTP connections are assumed to always exist among the *C-Reps* and the origin servers so that there should be no connection setup time per request. One enhancement to *CNP* also taken into account is that the origin server gives the processing priority to the requests from the *C-Reps* over the requests from the other proxy servers. This influences the values chosen for the parameters such as $CNP_{os\text{-}Get}$, $CNP_{os\text{-}GIMS}$ and $RTT_{CSP\text{-}CRep}$.

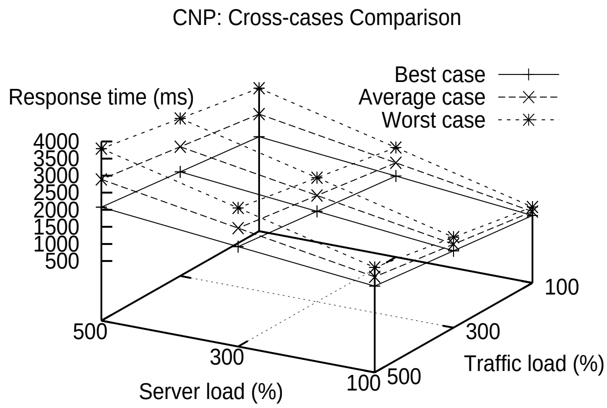
TABLE 7 Response times for CD & CNP

Server demand	Avg		Avg		Hvy		Hvy	
Traffic load	**Avg**	%*	**Hvy**	%*	**Avg**	%*	**Hvy**	%*
No hierarchy used	1090	83	3010	121	3510	127	5450	139
CD-1 (15)	1309	100	2477	100	2759	100	3927	100
CD-2 (15)	1230	94	2301	93	2657	96	3729	95
CNP's worst case	974	74	1829	74	2937	106	3793	97
CNP's average case	840	64	1551	63	2180	79	2891	74
CNP's best case	722	55	1282	52	1516	55	2076	53

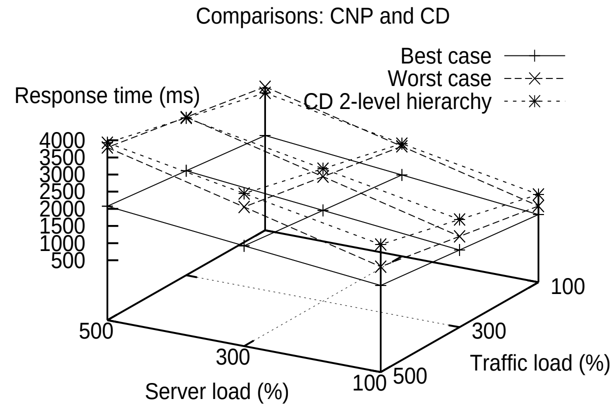
*The percentage calculation is relative to the values for CD-1. Note that the difference in response times will increase if Access_os < 1. Besides, CNP's performance is even better if we use 300% as the ratio for high server demand.

2) *Simulation outcome for the CNP*

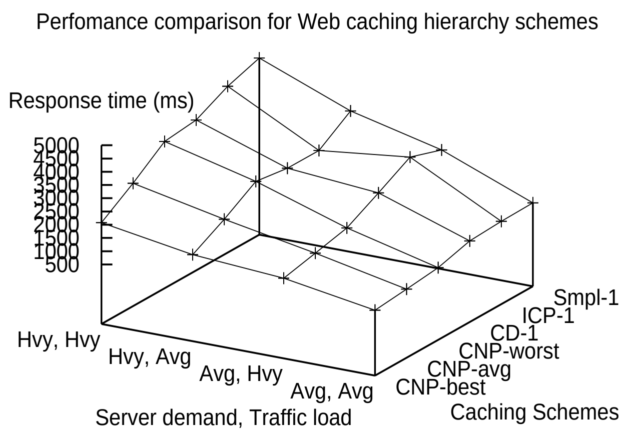
To understand the performance fluctuation when *CNP* is used, we set up the experiments to monitor the best, the worst, and the average cases by setting the values of the previously discussed parameters at the high, low, and middle of their ranges, respectively. Fig. 20a displays the performance comparison when using these three different settings. We found that *CNP* was very insensitive to the change of the traffic load when the demand of the origin server was average. As expected, the server demand is the dominating factor for the response times for *CNP*. Fig. 20b shows the performance comparisons among *CD-1*, *CNP*'s best case, and *CNP*'s worst case. We can see that *CNP's* best case is far superior to *CD*-1 while *the CNP's* worst case also outperforms *CD*-1 when the server demand is not very high. It strengthens the likelihood that *CNP* is a better approach in terms of providing the shortest response time for a median-sized document. We also conducted the comparison for six Web caching configurations under combinations of server demands and traffic loads in Fig. 20c. Clearly, *CNP* is superior to all the other approaches.



(a)



(b)



(c)

Figure 20: The simulation outcome for CNP and the comparison of four different Web caching hierarchy schemes

V. DISCUSSION

1. Concluding from all the simulation results presented in this paper, we can see that only *CNP* can improve the response times in all tested cases when compared to the case where no hierarchy is used (Table 7). Note that all the other approaches are not able to match up with the no-hierarchy case when the load and demand are average. It is critical because such a case is typical for all possible load/demand combinations.

2. From the perspective of the response times, the number of levels in the hierarchy should be low. However, the variance of the response times are reduced as the number of levels in the hierarchy increases. Along with the consideration that the caching hierarchy can reduce the netwrok traffic, the facts suggest that two-level hierarchy may be the best configuration for the Web caching hierarchy. We also found that the HTTP timeout value needs to be much lower than the specified 45 seconds to improve the response times for the Web caching.

3. In this research, *CNP* was found to outperform the other Web caching schemes, not to mention that CNP handles the cache coherency problem while the others do not. It is not a surprise to us because *CNP* takes advantage of the Web accessing characteristics and sets up the *C-Reps* to enhance the hit rate to be higher than 90%. Such a high hit rate could not be achieved by the other schemes compared in this study because the overlap of the frequently accessed documents in the cache and the waste of considerable cache storage space on documents which are rarely accessed. We envisage that such a problem can only be alleviated by adopting the server-oriented approach instead of the client-oriented approach.

4. In our approach, we pay attention to the user response times because we believe that for the Web caching to be successful, it needs to benefit the end users directly. What the users care most is the reduction of the response times for the document requests. If the user can obtain documents directly from the origin servers much faster than from the proxy server, definitely it will hinder the use of the proxy servers. Therefore, in our opinion, the response time should be the foremost perfomance measuring metric for the Web caching research.

VI. SUMMARY

In this paper, we presented the mathematical modeling results for four Web caching hierarchy schemes: simple hierarchy, *ICP*, *CD*, and *CNP*. Based on the result of the modeling, the statistics collected in [17], and the knowledge derived from the trace log analysis in [1], we set up the values for the parameters in the models and performed simulation experiments to understand the factors on the response times for the schemes under different origin server demands and traffic loads. The complete simulation result containing hundreds of 2-D figures is available in [17]. To save space, 3-D figures were used to illustrate the important properties of the response times for the Web caching hierarchies. The comparison of the response times for different caching schemes was presented and discussed. *CNP* appears to outperform the other schemes even at the assumed worst-case scenario.

REFERENCES

[1] C.-Y. Chiang, M. Liu and M. Muller, "*Caching Neighborhood Protocol: A Foundation for Building Dynamic Caching Hierarchies with WWW Proxy Servers*," to appear in ICPP'99, Aizu, Japan, September 1999. (*http://www.cis.ohio-state.edu/~chiang/cnp/icpp99-submission.ps*)

[2] A. Chankhunthod, P. B. Danzig, C. Neerdals, M. F. Schwartz and K. J. Worrell, "*A Hierarchical Internet Object Cache,*", in Proceedings of the USENIX Technical Conference, San Diego, CA, January 1996.

[3] A. Dingle and T. Partl, "*Web Cache Coherence*," Computer Network and ISDN Systems, 28(1996), pp. 907-920.

[4] A. Luotonen, K. Altis, "*World-Wide Web Proxies,*" Computer Networks and ISDN Systems 27(1994) pp. 147-154.

[5] T. Berners-Lee, R. Calliau, A. Luotonen, H. Nielsen, and A. Secret, "*The World-Wide Web*," Communications of the ACM, 37(8), pp.76-82, 1994.

[6] C. Liu and P. Cao, "*Maintaining Strong Cache Consistency in the World-Wide Web,*" in Proceedings of the International Conference on Distributed Computing Systems, 1997.

[7] D. Wessels and K. Claffy, "*Internet Cache Protocol (ICP), version 2*," RFC2186, 1998.

[8] J. Gwertzman and M. Seltzer, "*The Case for Geographical Push Caching,*" in Proceedings of the 1995 Workshop on Hot Operating Systems, 1995.

[9] J. Gwertzman and M. Seltzer, "*World Wide Web Cache Consistency,*" in Proceedings of the 1996 USENIX Technical Conference, San Diego, January 1996.

[10] K. J. Worrell, "*Invalidation in Large Scale Network object Caches,*" Master Thesis, Dept. of Computer Science, 1994.

[11] L. Zhang, S. Floyd, and V. Jacobson, "*Adaptive Web caching,*" in the 2nd Web Caching Workshop, Boulder, CO, June 1997.

[12] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams and E.A. Fox, "*Caching Proxies: Limitations and Potentials,*" in Proceedings of the Fourth Intl. World Wide Web Conference., Boston, MA, Dec. 1995.

[13] M. Baentsch, L. Baum, G. Molter, S. Rothkugel and P. Sturm, "*Enhancing the Web's Infrastructure: from Caching to Replication*" IEEE Computing, pp. 18-27, March-April, 1997.

[14] M. Baentsch, L. Baum, G. Molter, S. Rothkugel and P. Sturm, "*World Wide Web Caching: The Application-Level View of the Internet,*", IEEE Communications Magazine, June 1997.

[15] M. Nabeshima, "*The Japan Cache Project: An Experiment on Domain Cache,*" in Proceedings of the 6th International World Wide Web Conference, 1997.

[16] National Lab of Applied Network Research, Sanitized access log, Jan. 1999. (*ftp://ir-*

cache.nlanr.net/Traces/)

[17] A. Rousskov, "*On Performance of Caching Proxies,*" SIGMETRICS'98, (The result is obtainable from *http://www.cs.ndsu.nodak.edu/~rousskov/research/cache/squid/profiling/*)

[18] A. Rousskov, D. Wessels, "*Cache Digest,*" Computer Networks and ISDN Systems, Vol. 30, No 22-23, Nov. 1998.

[19] B. Bloom, "*Space/time Trade-offs in Hash Coding with Allowable Errors,*", Communications of the ACM, Vol. 13, pp. 422-426, July 1970.

[20] Steven Glassman, "*A Caching Relay for the World Wide Web,*" in Proceedings of the 1st International Conference on the WWW, 1994.

[21] Squid Internet Object Cache (*http://squid.nlanr.net/Squid/*)

[22] V. N. Padmanabhan and J. C. Mogul, "*Using Predictive Prefetching to Improve World-Wide Web Latency,*" ACM Computer Communication Review, pp.22-36, v.27, n3, July 1996.

[23] L. Fan, P. Cao, J. Almeida and A. Z. Broder, "*Summary Cache: A Scalable Wide-area Cache Sharing Protocol*" in Proceedings of the ACM SIGCOMM'98, October, 1998.

[24] C.-Y. Chiang, M. Ueno, Y. Li, M. Liu and M. Muller, "*Maintaining Cache Coherency for Proxy Servers in Dynamic Web Caching Hierarchies*", unpublished.

[25] C.-Y. Chiang, Y. Li, M. Liu and M. Muller, "*Performance Study on the Request Forwarding for Proxy Servers in Dynamic Web Caching Hierarchies*", unpublished

[26] D. Wessels and K. Claffy, "*ICP and the Squid Web Cache*", IEEE JSAC, April 1998, pp. 345-357.

[27] M. Arlitt and C. L. Williamson, "*Internet Web Servers: Workload Characterization and Performance Implications*", IEEE/ACM Tran. on Networking, Vol. 5, No. 5, October 1997.

[28] Desire Main page. (*http://www.nic.surfnet.nl/surfnet/projects/desire/*

[29] NLANR weekly caching proxy statistics, (*http://www.ircache.net/Cache/Statistics/Vitals/Html/*

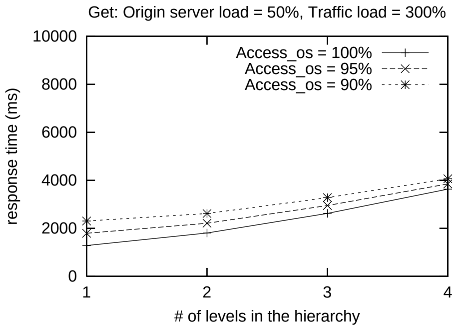
APPENDIX

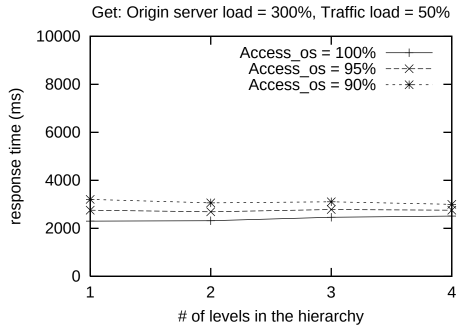
In this section, all the simulation for the configurations in Table 5 are presented in 2-D figures. The title on the upper-left corner of each page shows which parameters are under study for deriving the figures in the page. For example, "Simple-1, access-Level" indicates that the figures describe the simulation outcome of our study for the "Simple-1" configuration, the number of levels in the hierarchy is observed on the X-axis (that's why L is capitalized), and different values of Access_os were used to watch their influences on the response times.

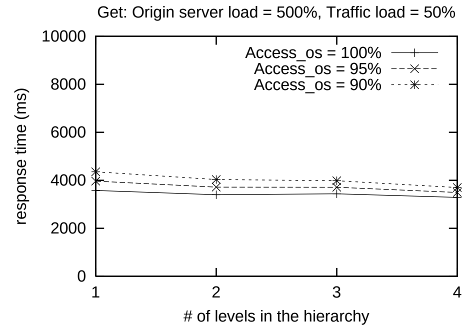
TABLE 8 Terms for the title of each page

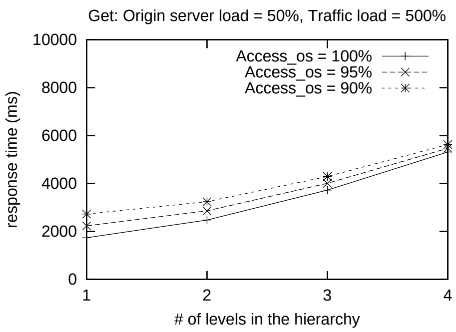
Terms	Descriptions
Access	Access_os (see Table 1)
Level	The number of levels in the hierarchy
Svr	Origin server demand
Tfc	Traffic load

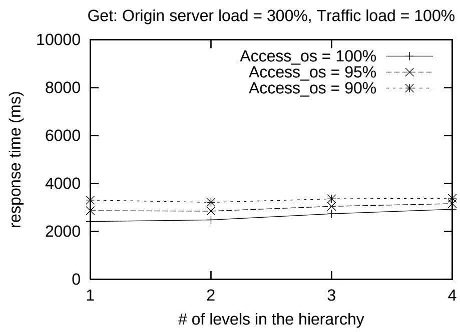
Simple-1, access-Level:

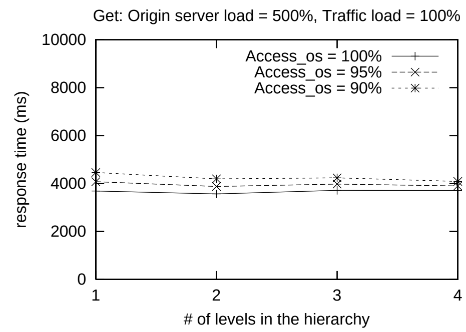


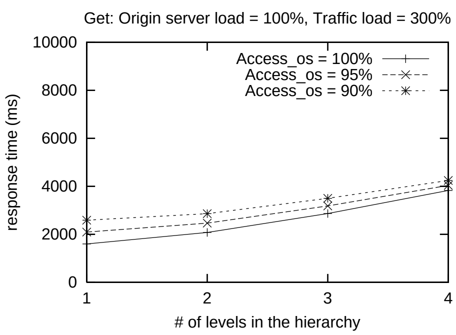


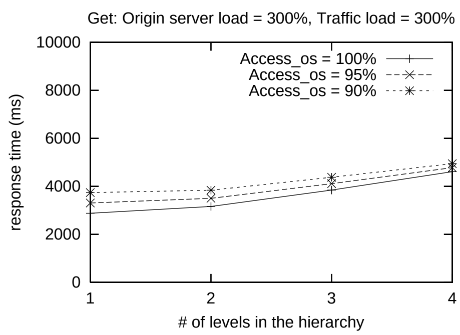


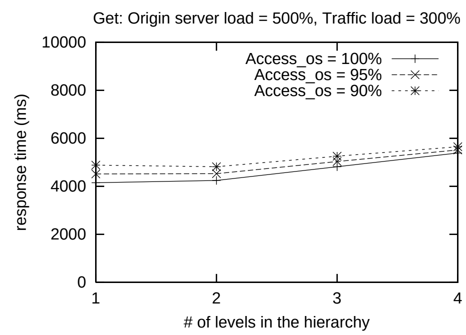


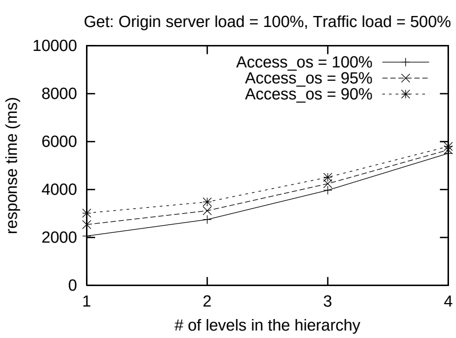


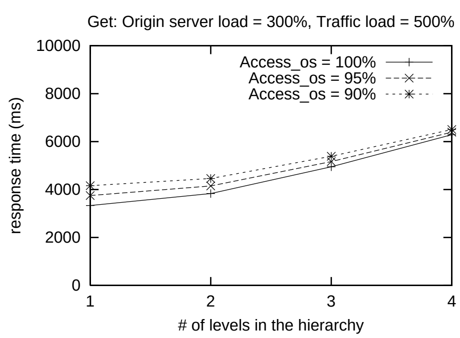


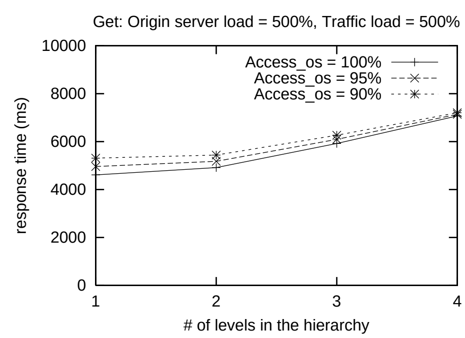




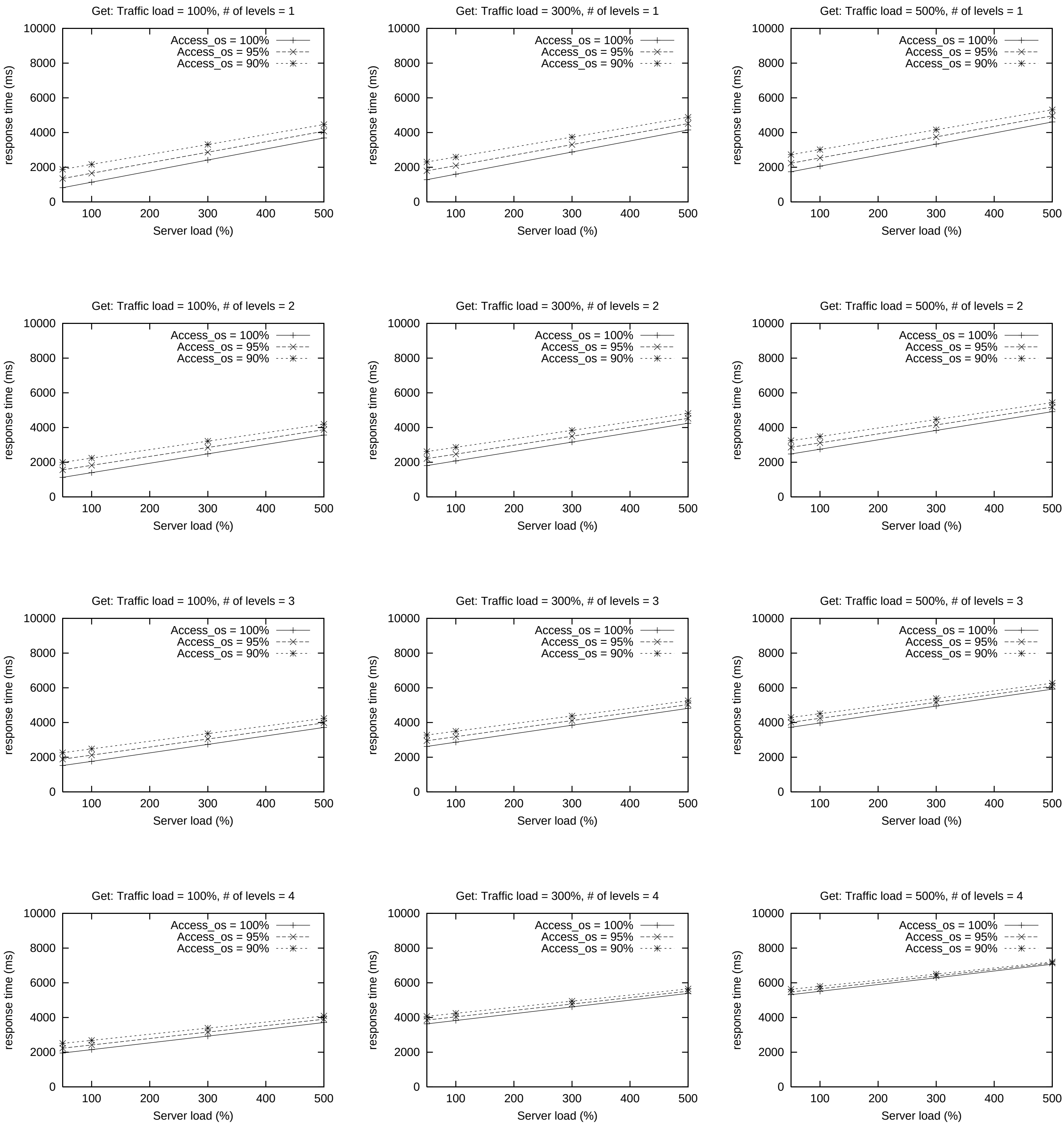




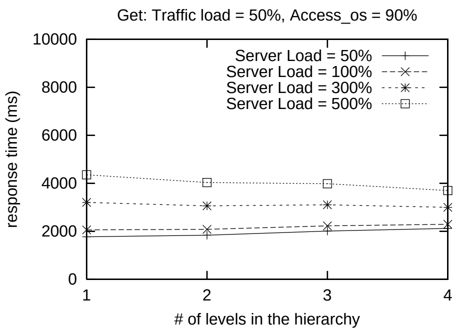


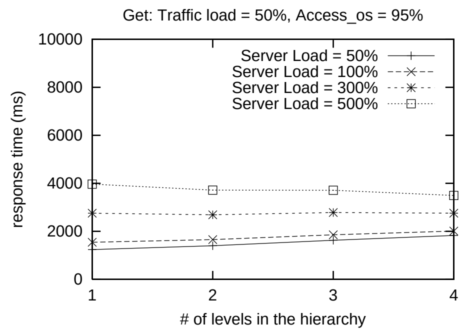


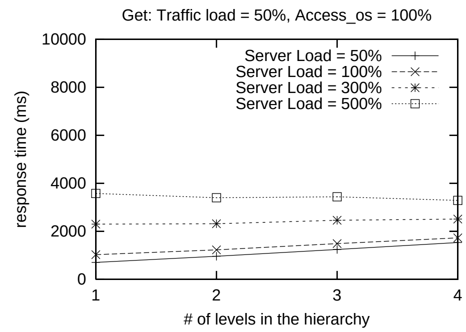
Simple-1, Svr-access:

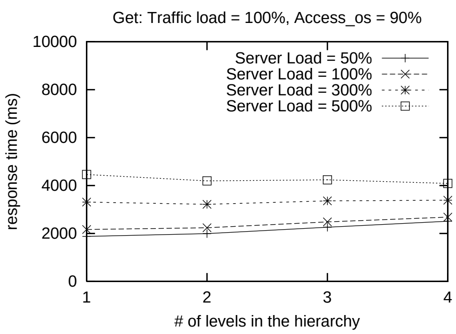


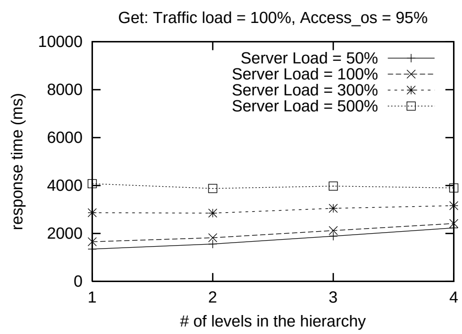
Simple-1, svr-Level:

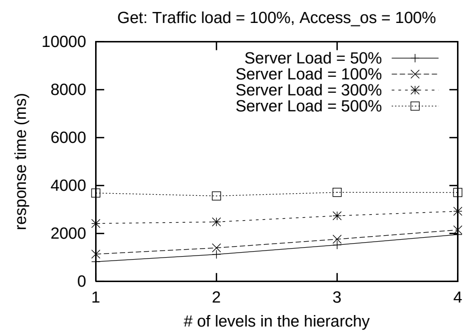


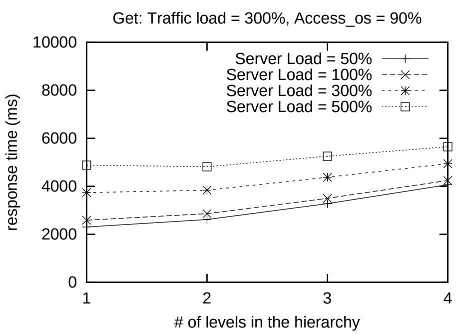


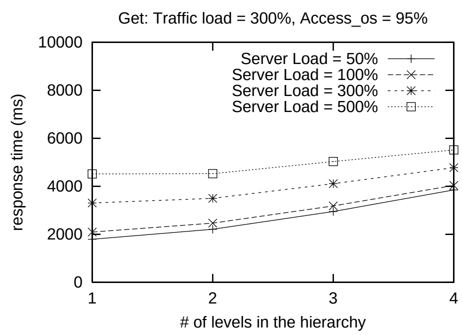


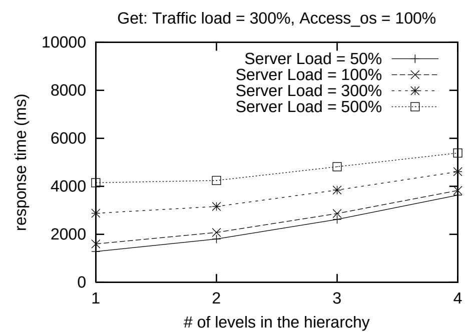


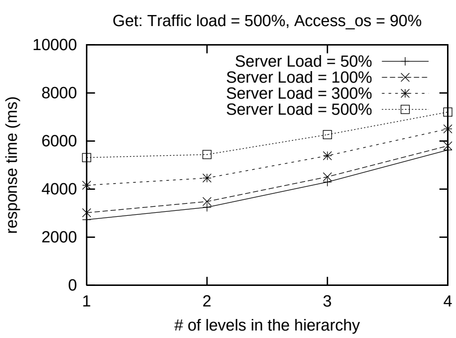


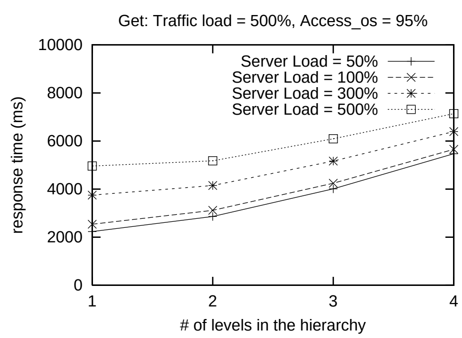


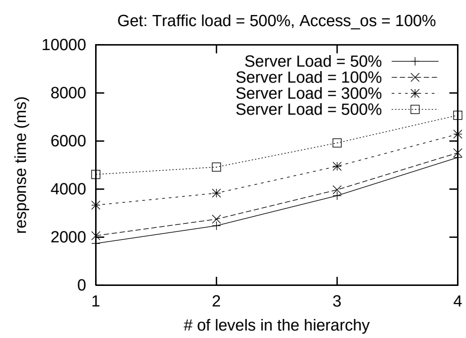




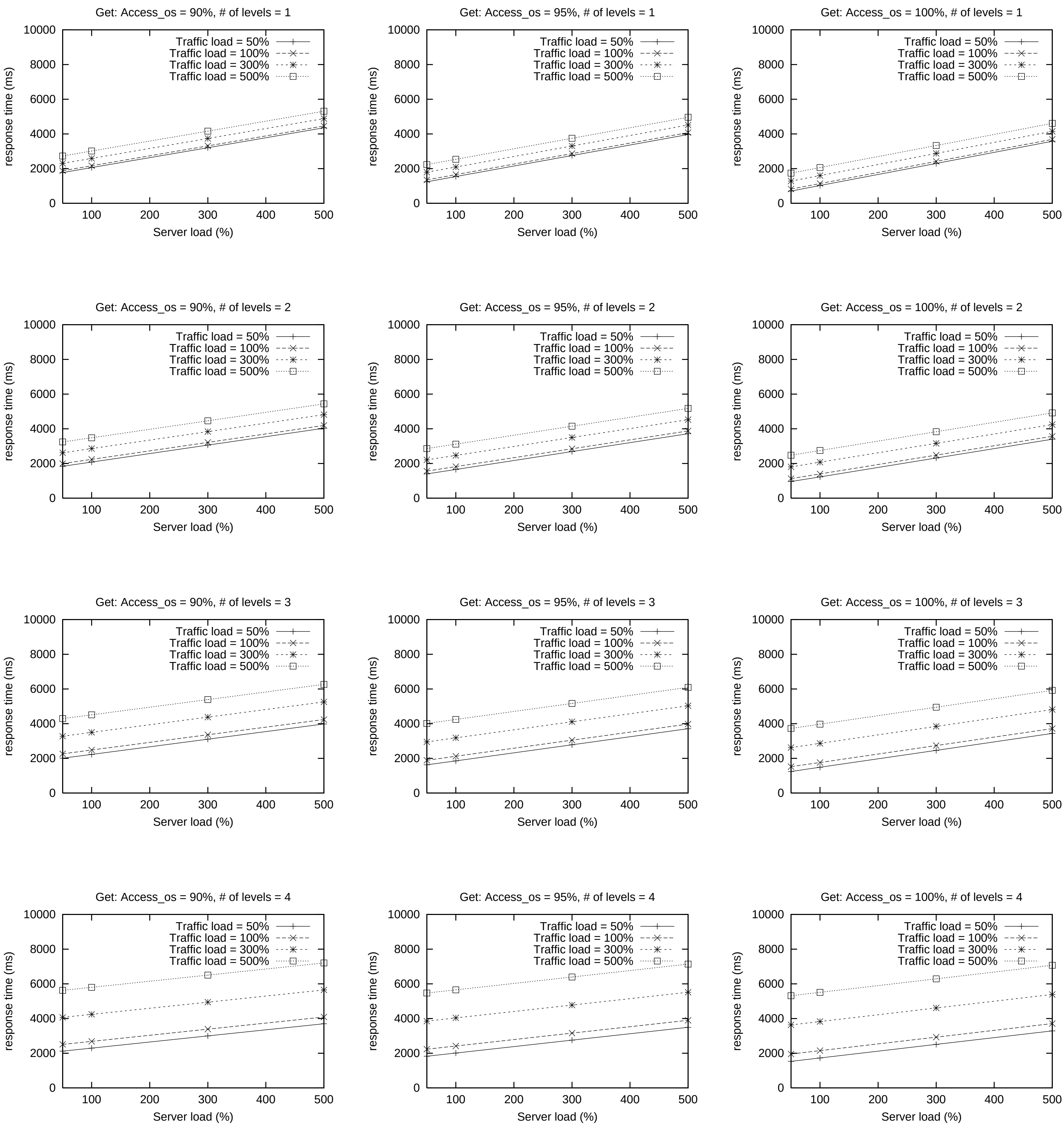




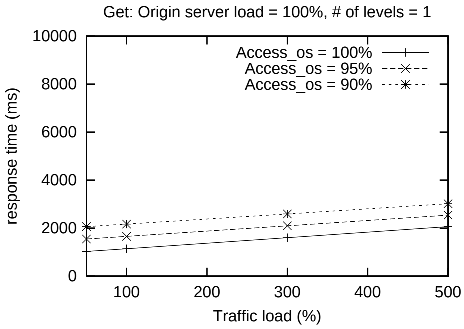


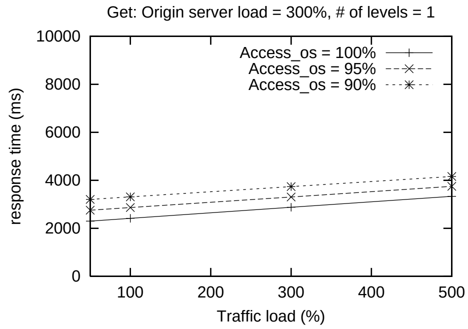


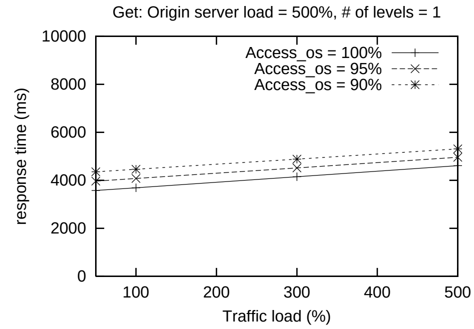
Simple-1, Svr-tfc:

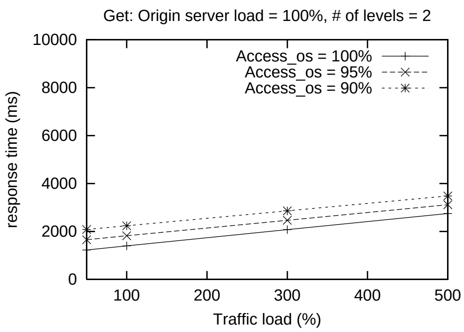


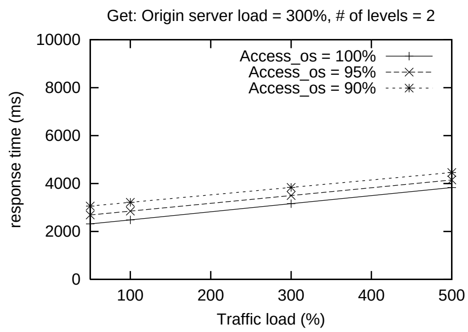
Simple-1, Tfc-access:

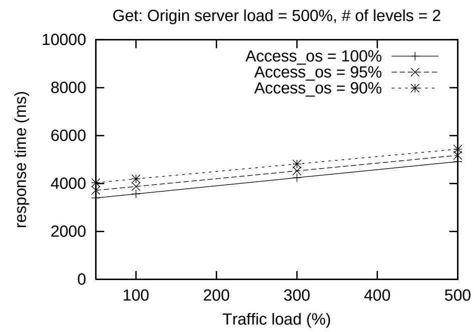


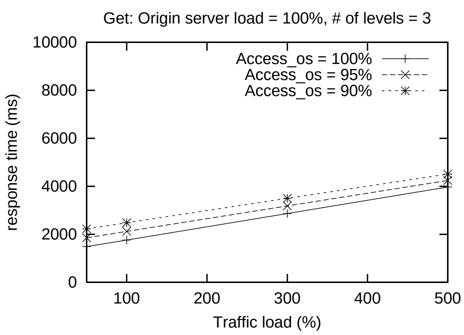


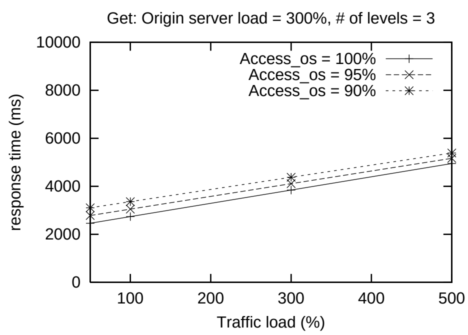


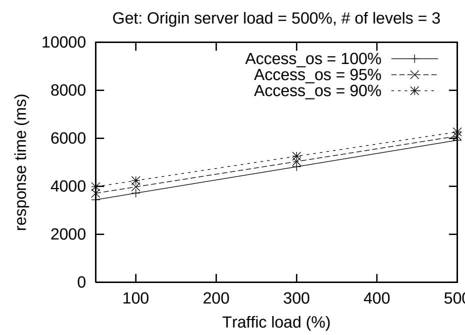


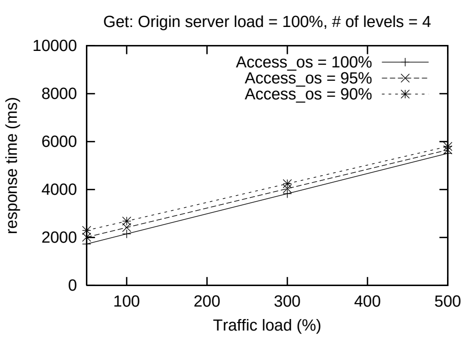


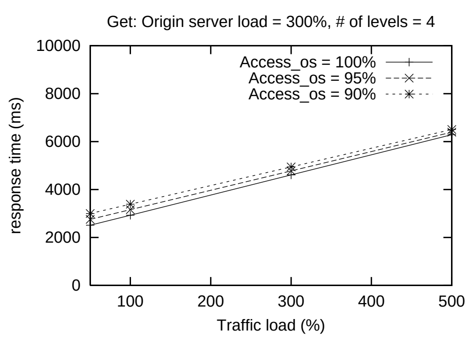


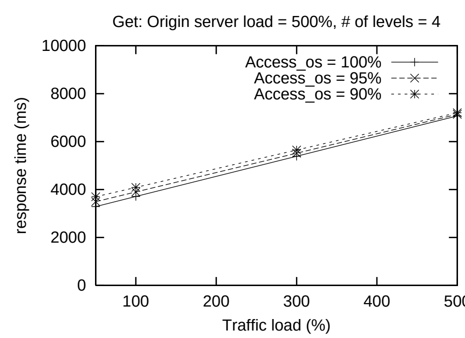




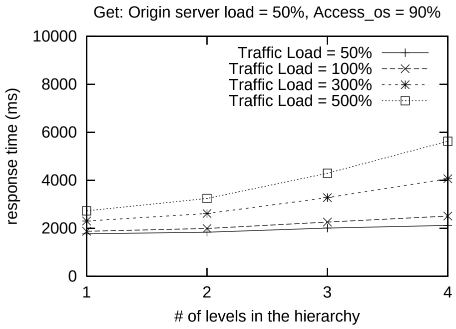


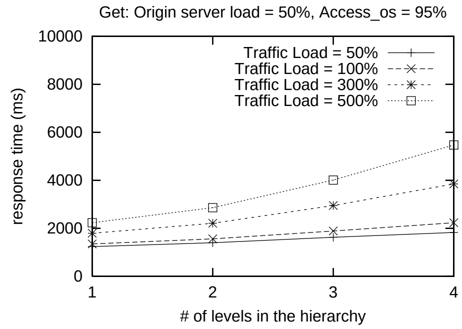


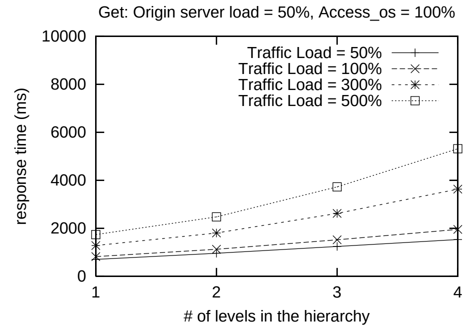


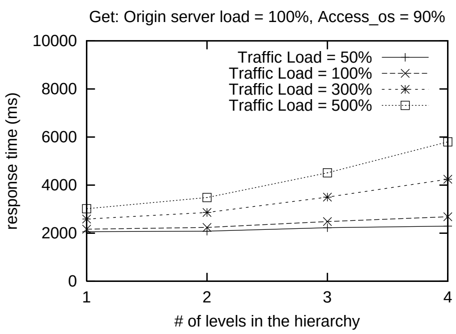


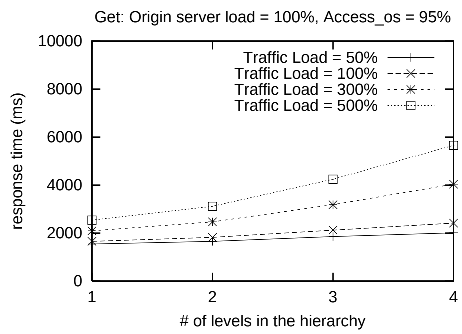
Simple-1, tfc-Level:

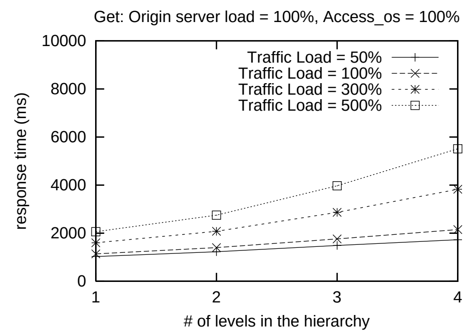


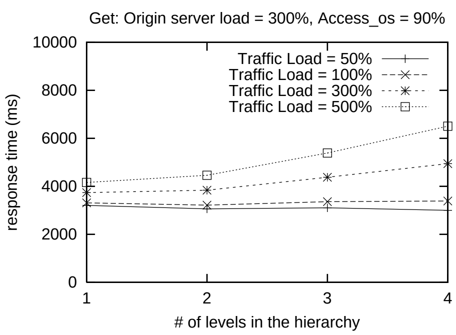


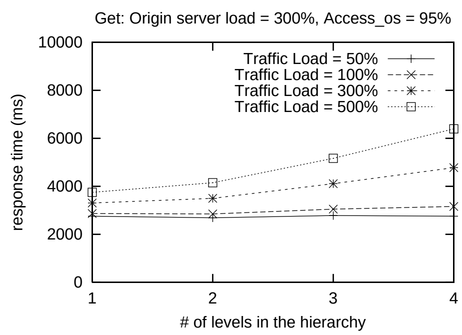


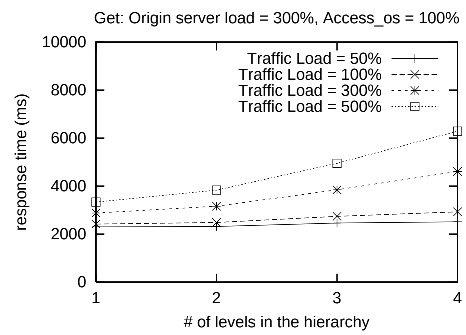


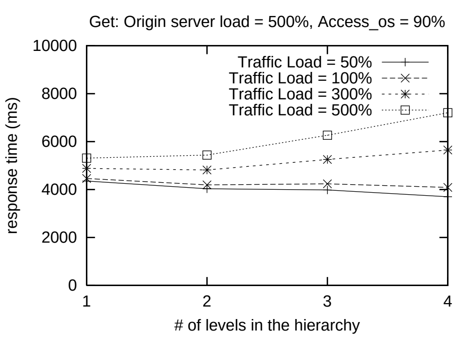


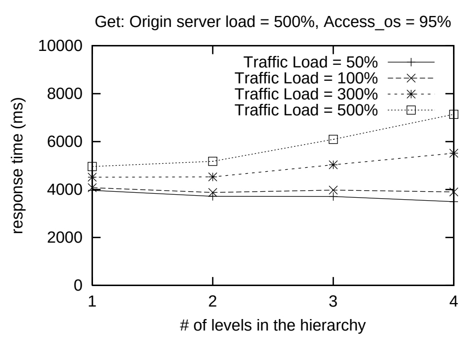


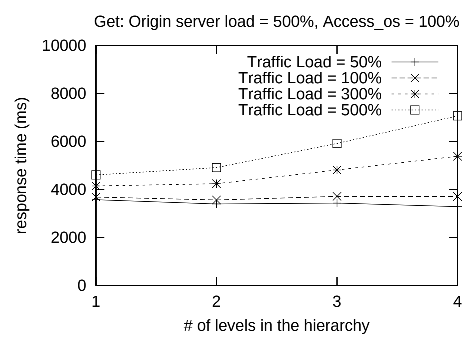




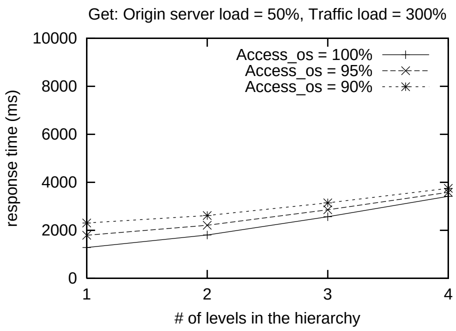


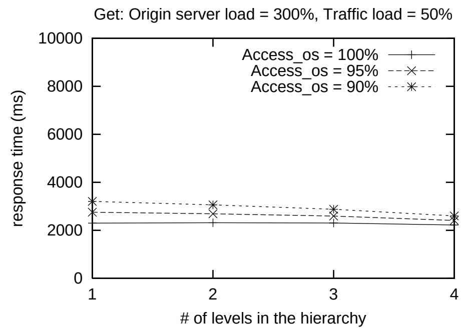


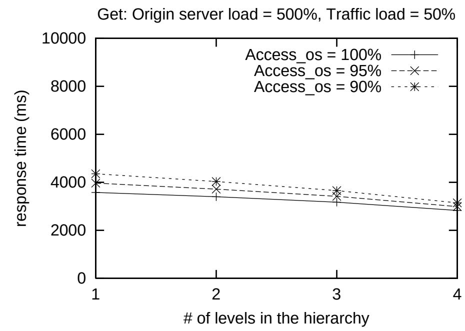


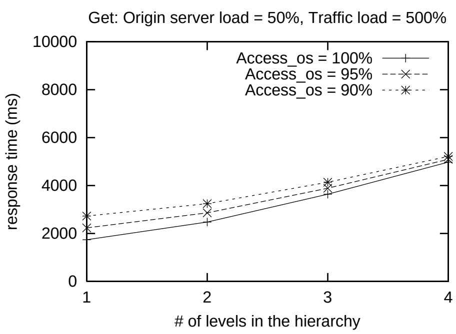


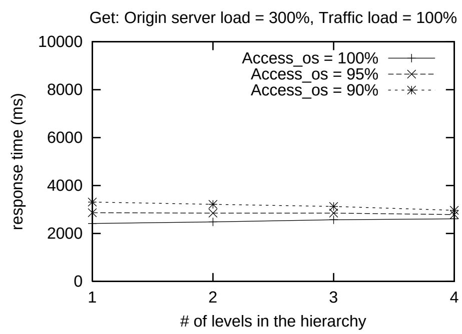
Simple-2, access-Level:

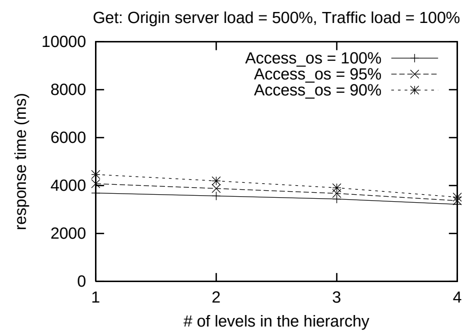


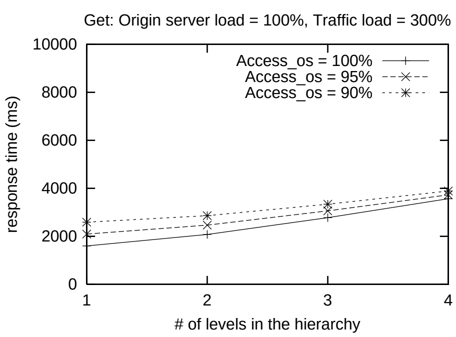


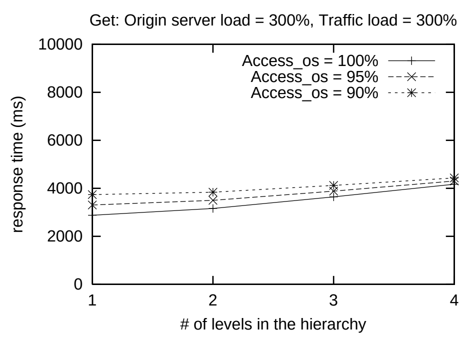


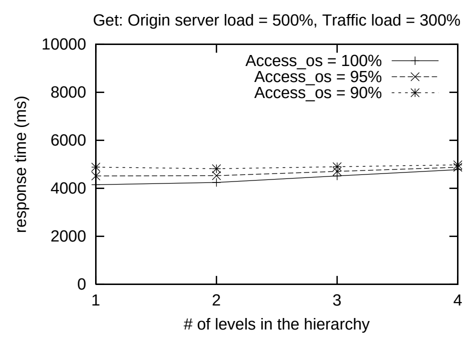


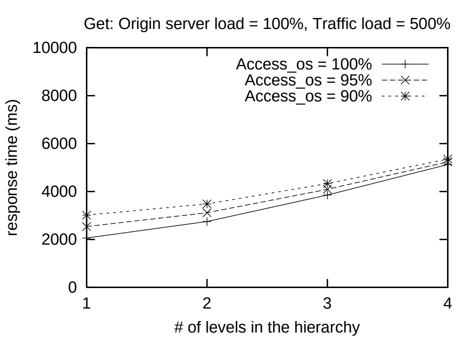


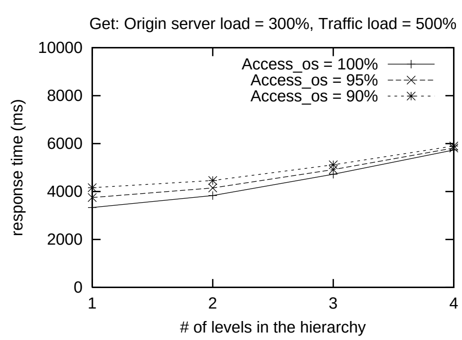


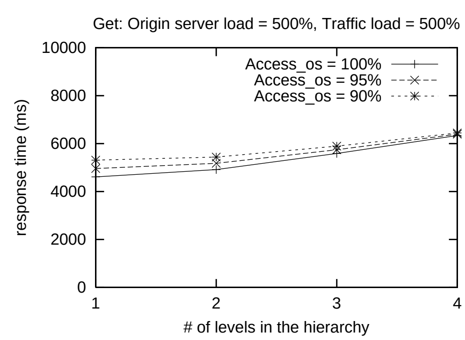




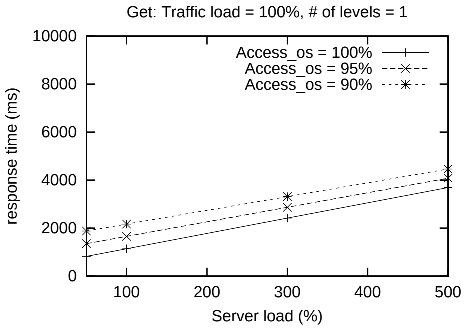


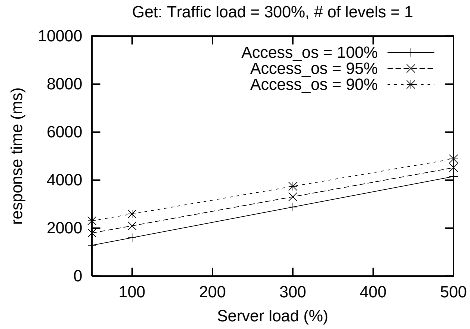


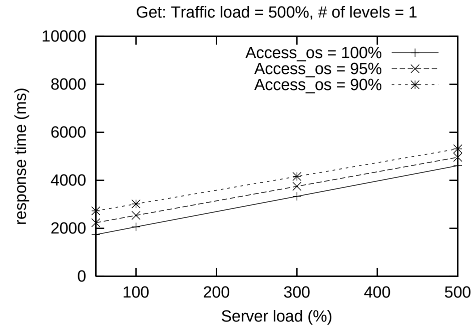


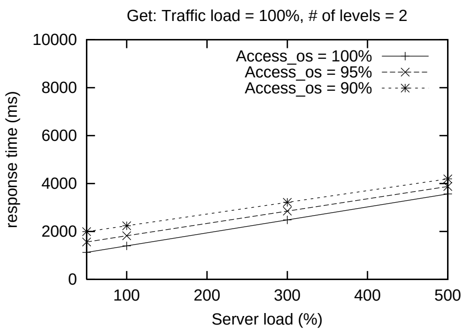


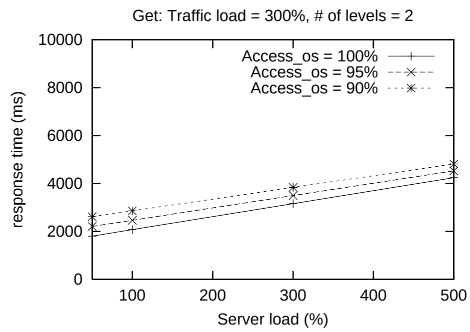
Simple-2, Svr-access:

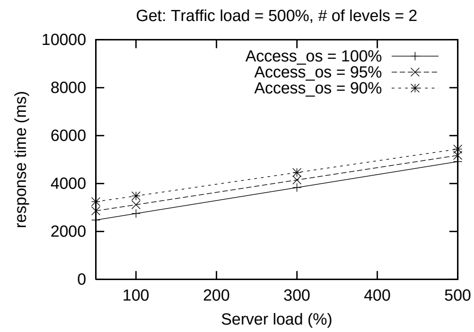


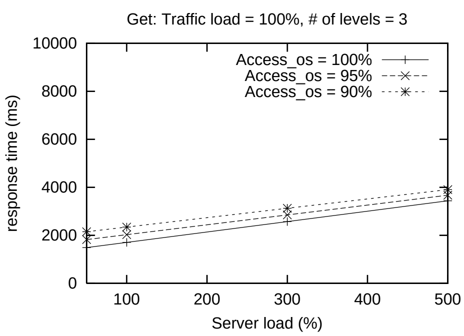


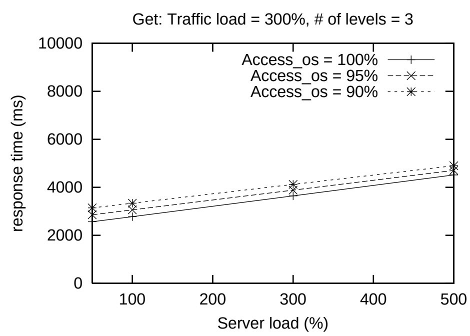


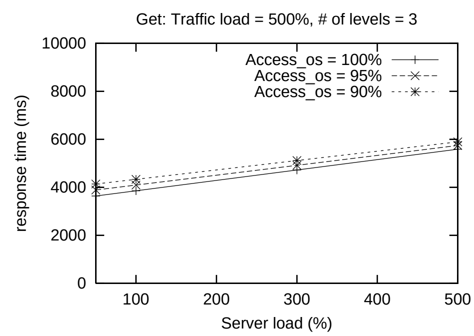


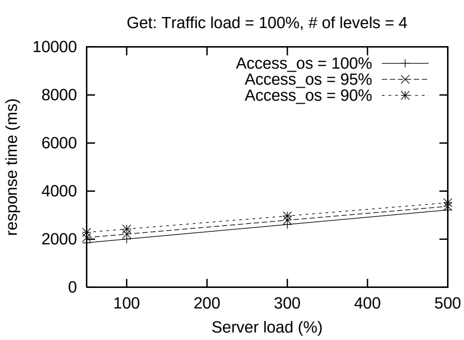


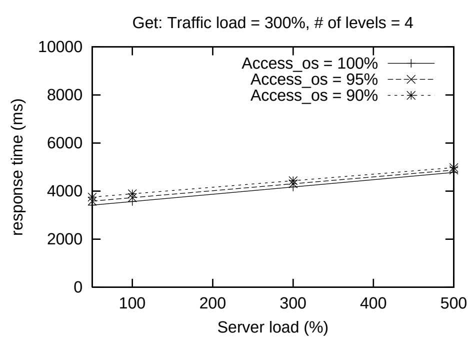


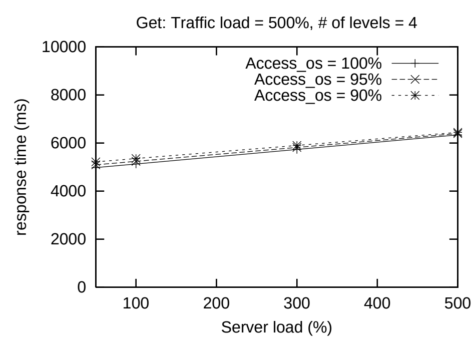




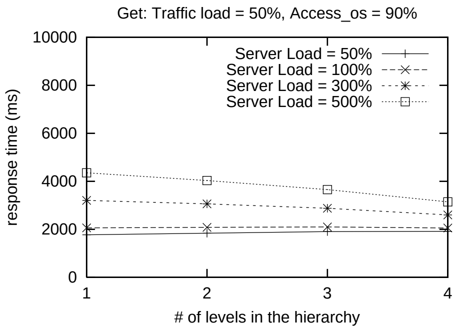


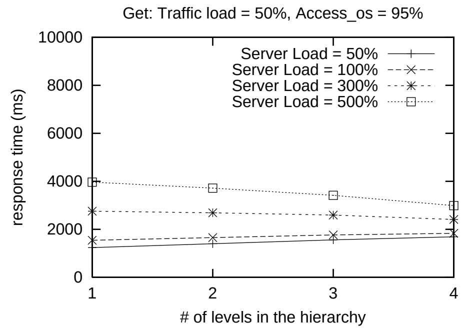


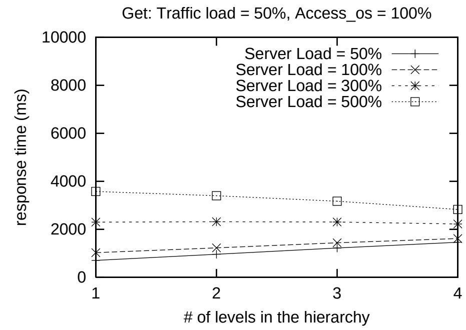


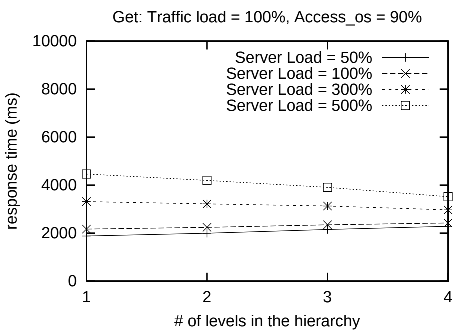


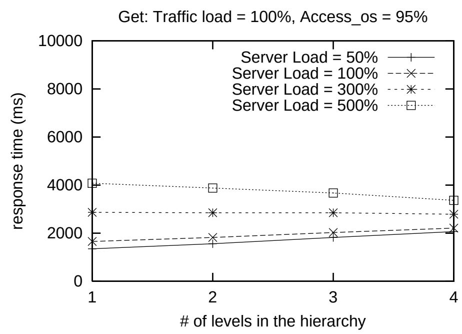
Simple-2, svr-Level:

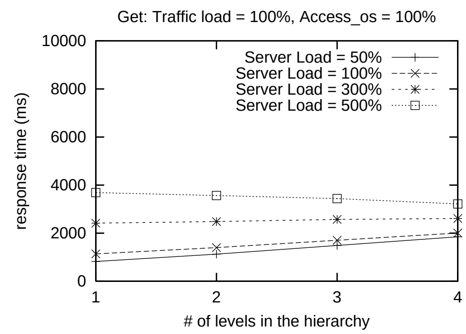


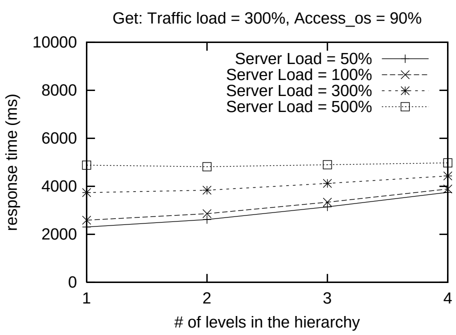


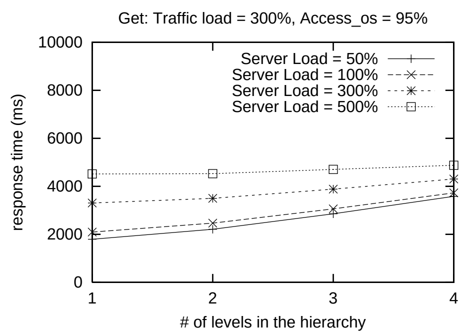


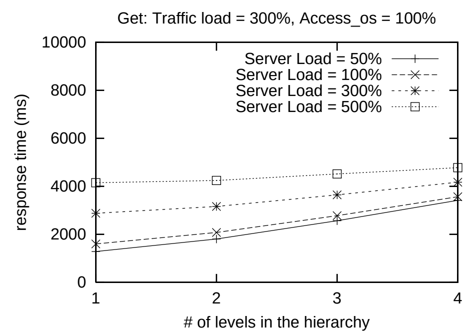


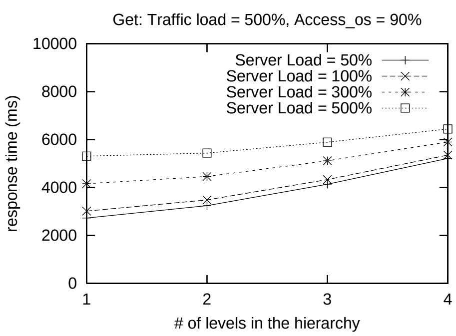


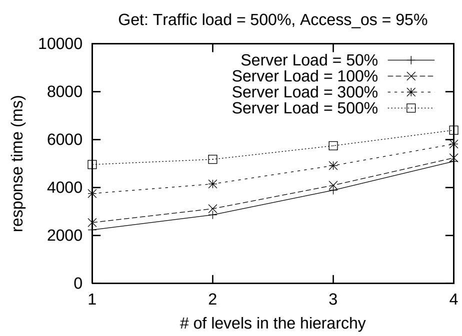


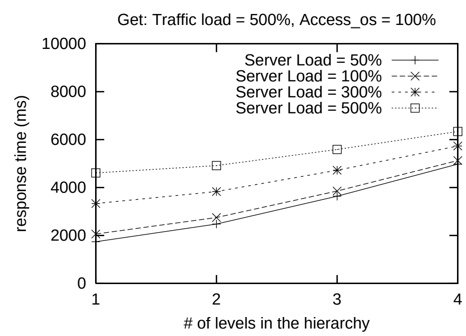






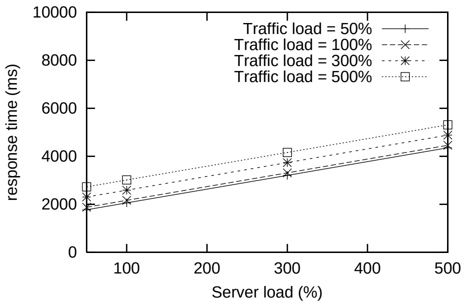




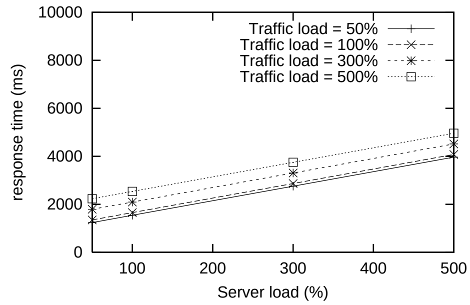


Simple-2, Svr-tfc:

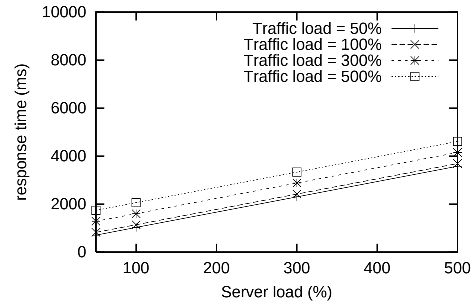
Get: Access_os = 90%, # of levels = 1



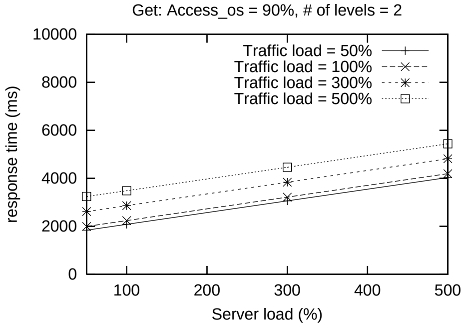
Get: Access_os = 95%, # of levels = 1



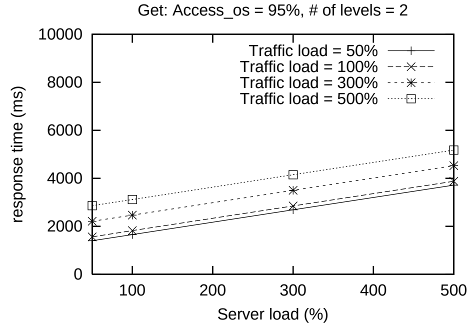
Get: Access_os = 100%, # of levels = 1



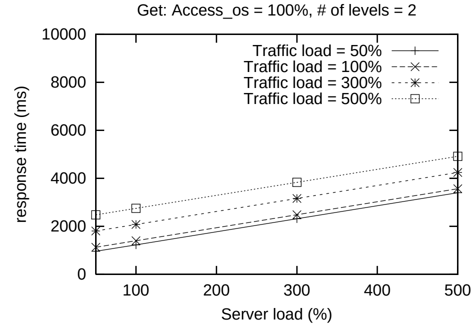
Get: Access_os = 90%, # of levels = 2



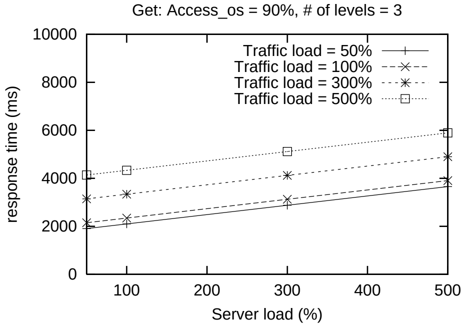
Get: Access_os = 95%, # of levels = 2



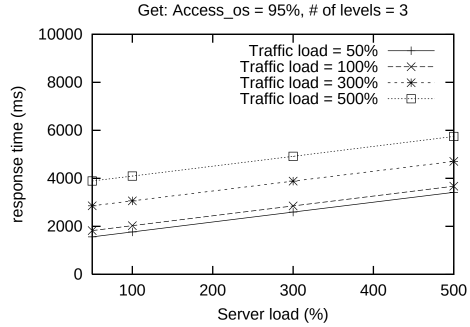
Get: Access_os = 100%, # of levels = 2



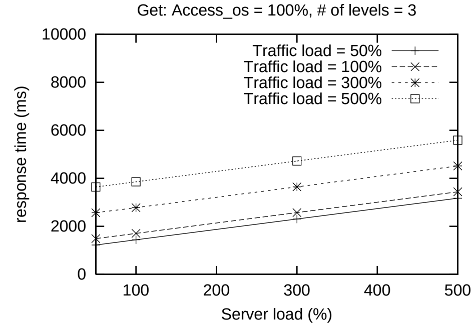
Get: Access_os = 90%, # of levels = 3



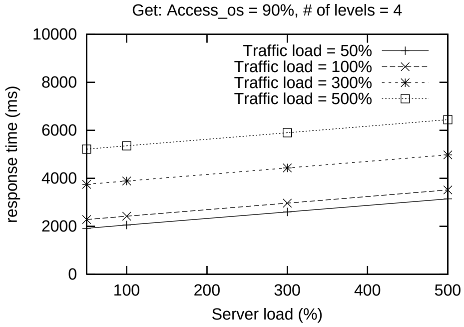
Get: Access_os = 95%, # of levels = 3



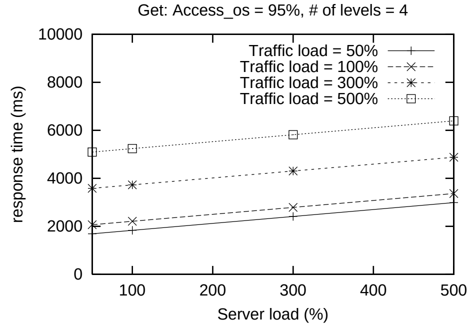
Get: Access_os = 100%, # of levels = 3



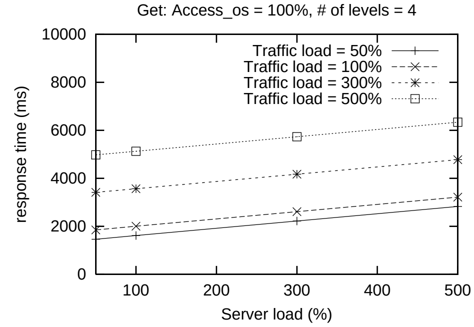
Get: Access_os = 90%, # of levels = 4



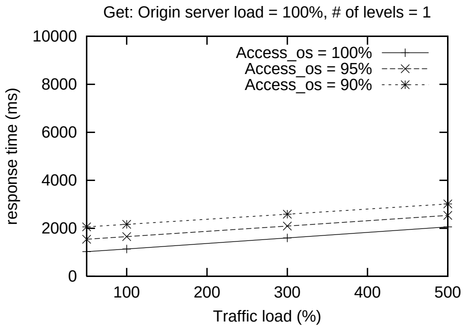
Get: Access_os = 95%, # of levels = 4

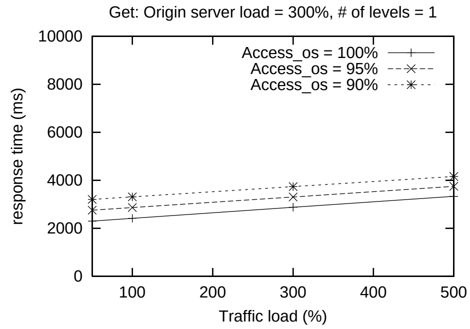


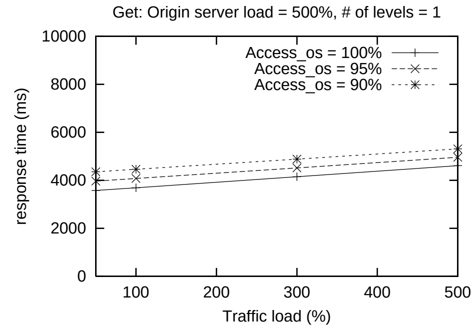
Get: Access_os = 100%, # of levels = 4

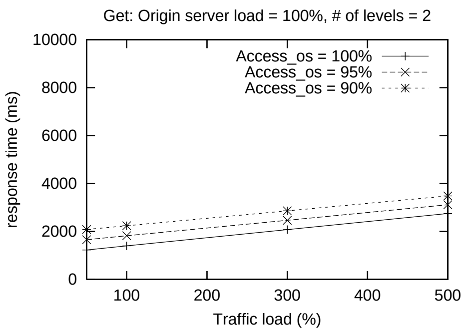


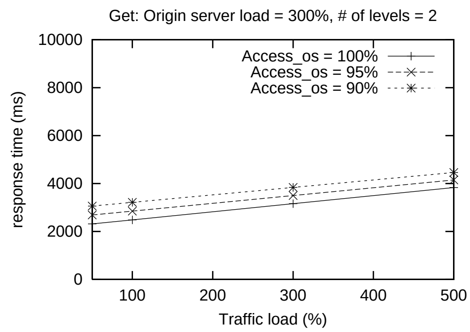
Simple-2, Tfc-access:

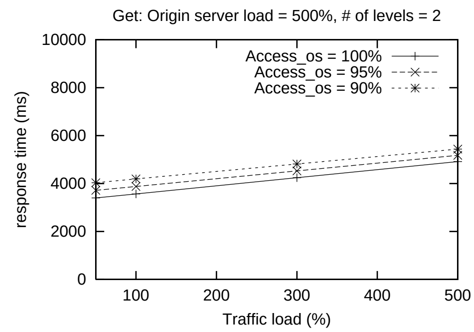


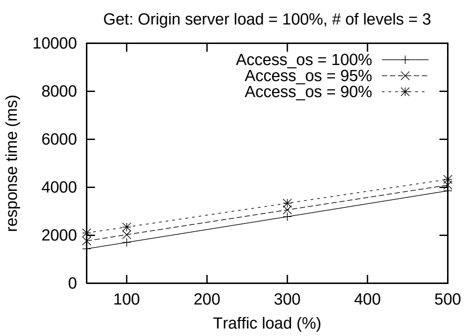


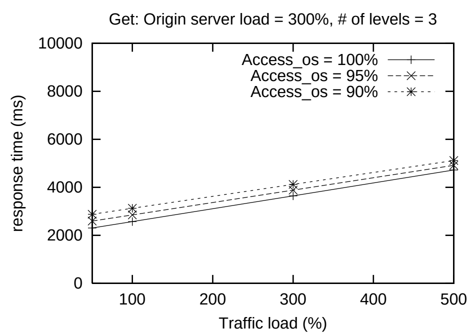


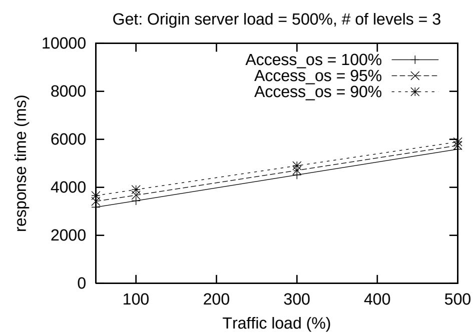


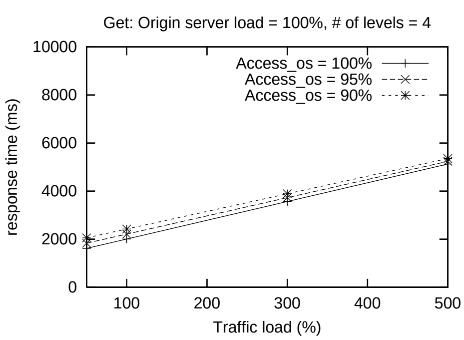


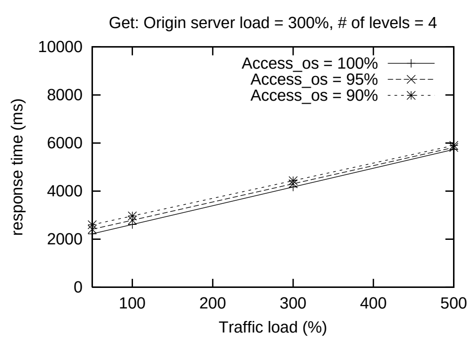


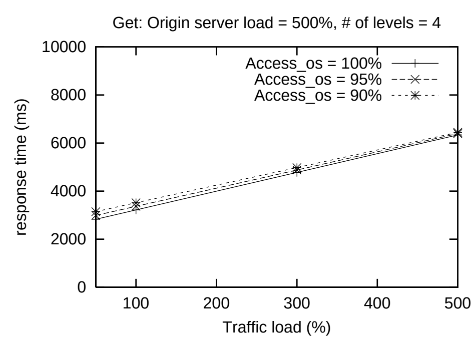




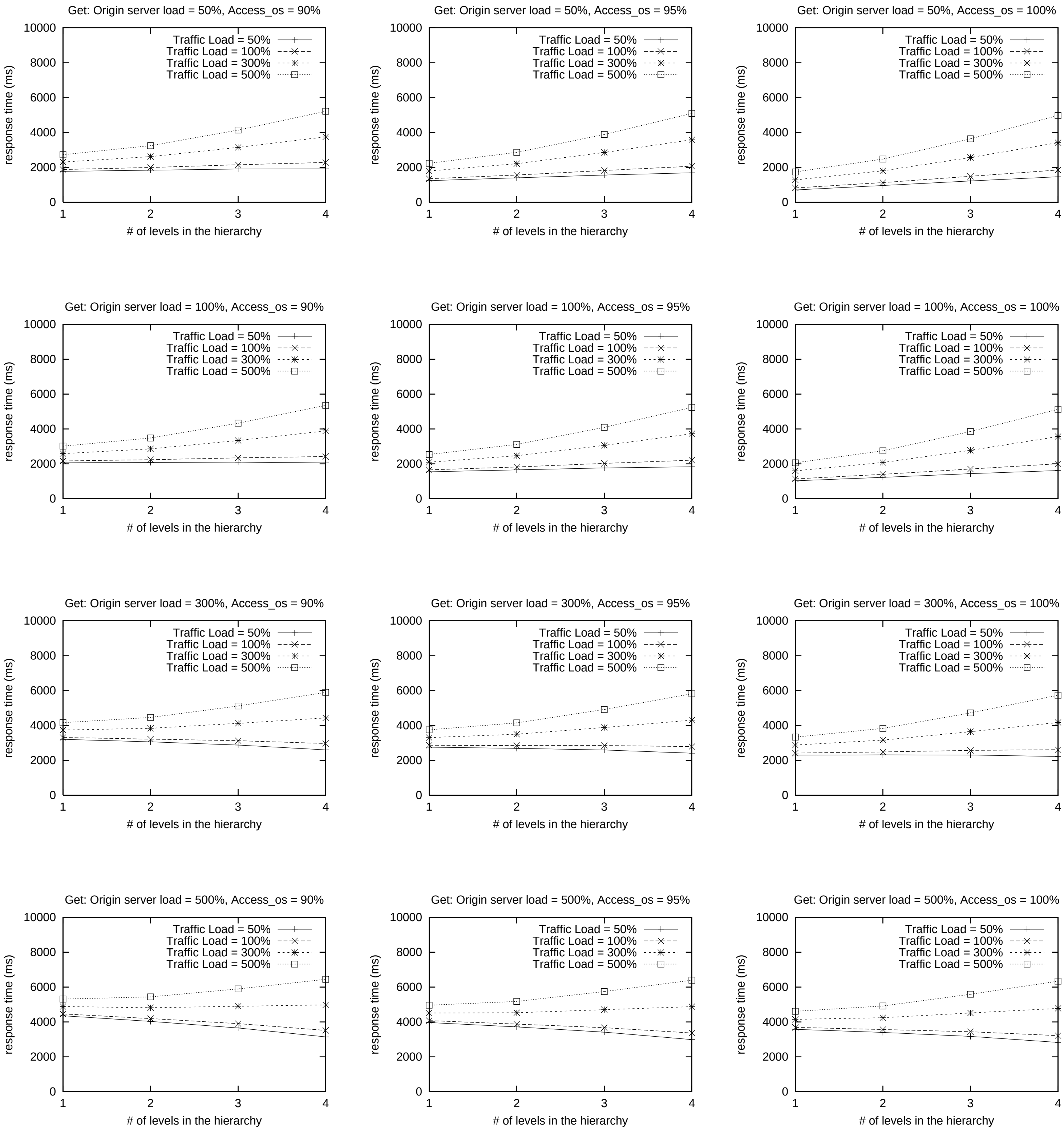




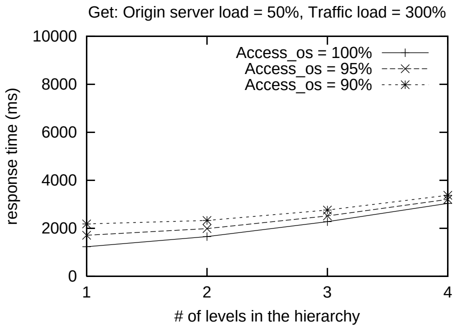


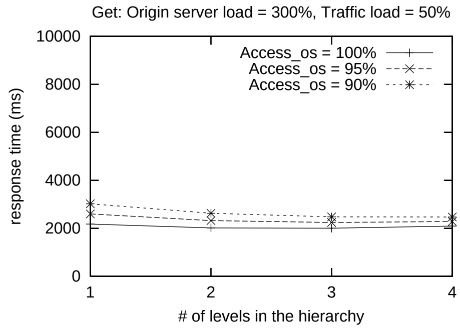


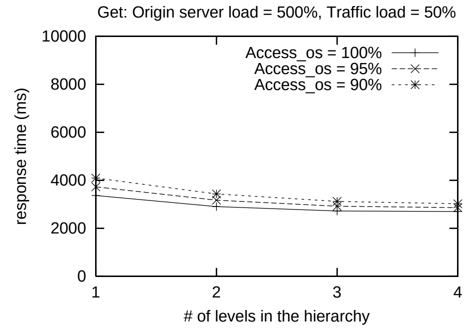
Simple-2, tfc-Level:

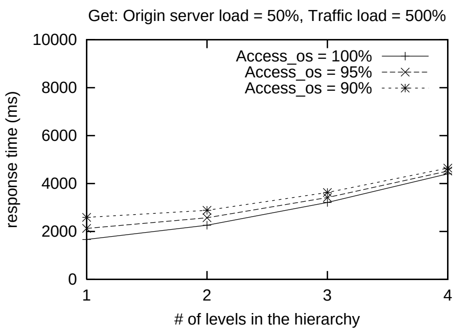


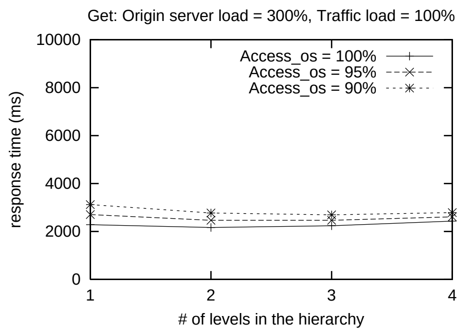
Simple-3, access-Level:

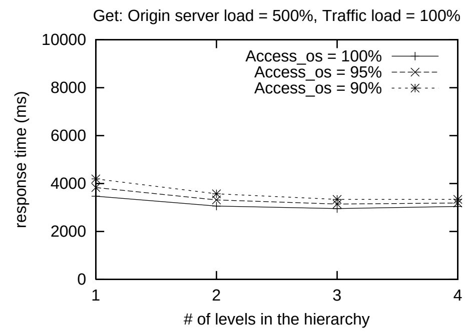


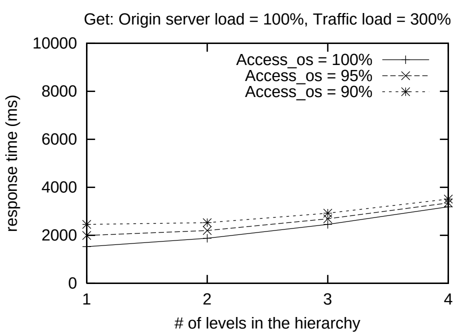


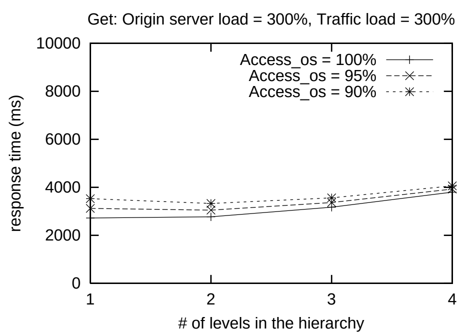


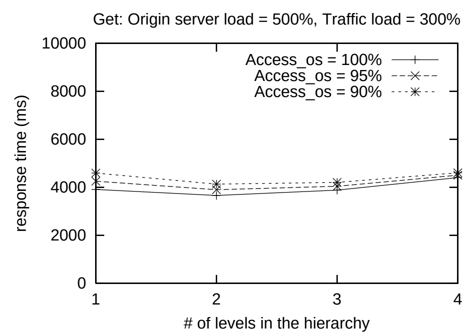


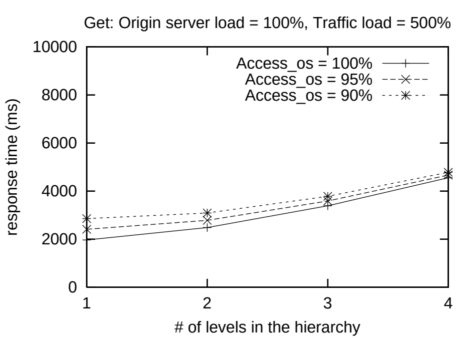


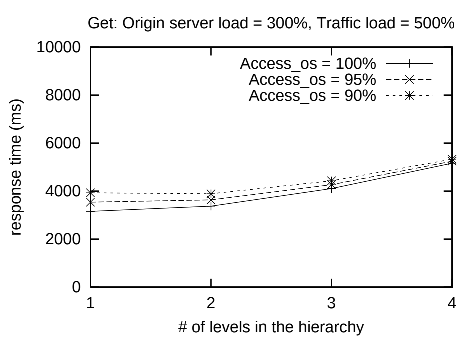


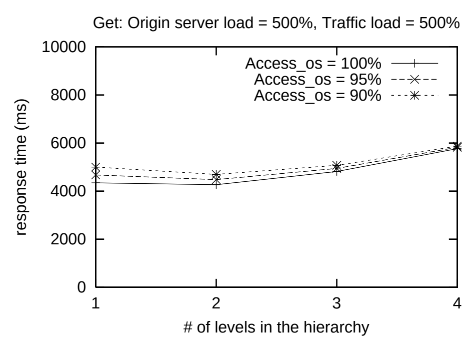




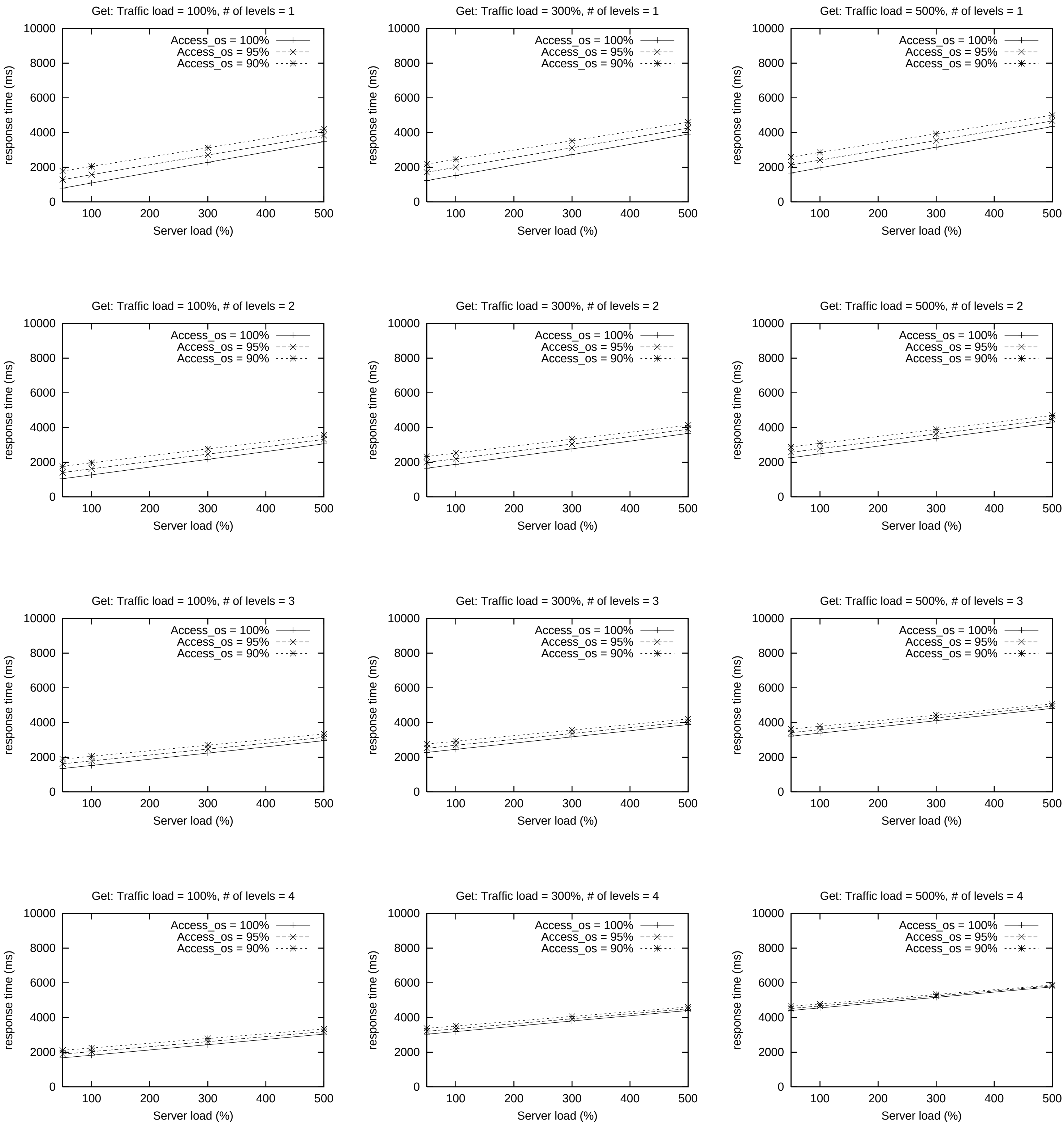




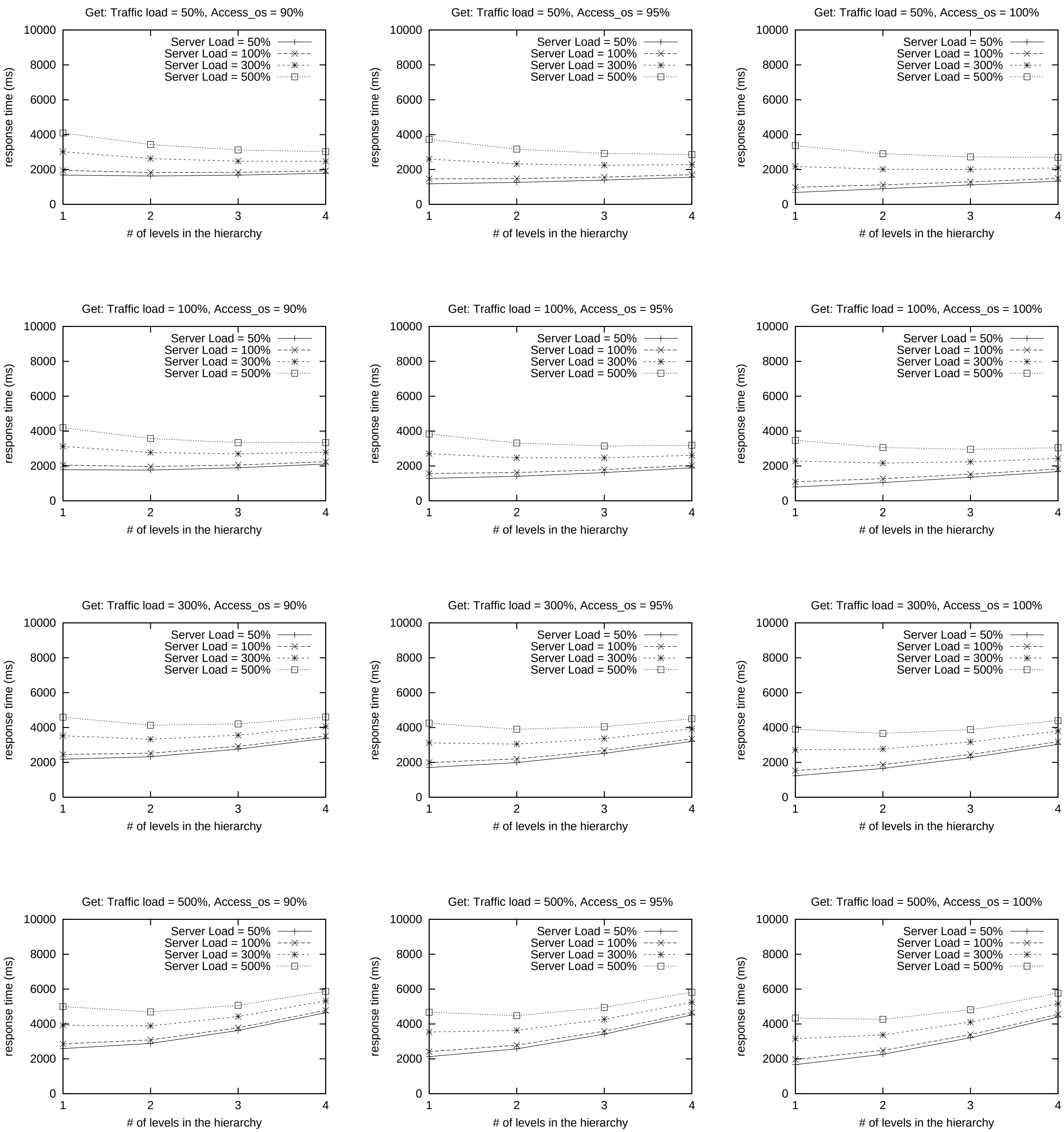




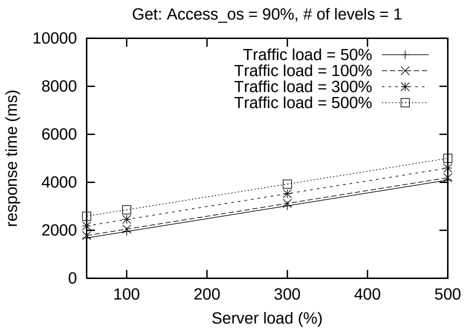
Simple-3, Svr-access:

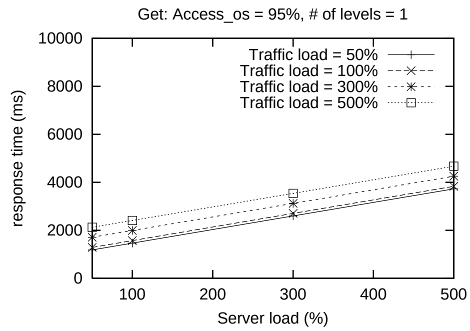


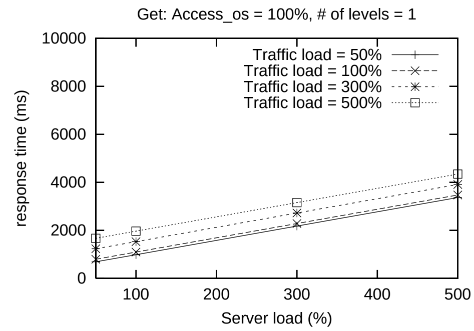
Simple-3, svr-Level:

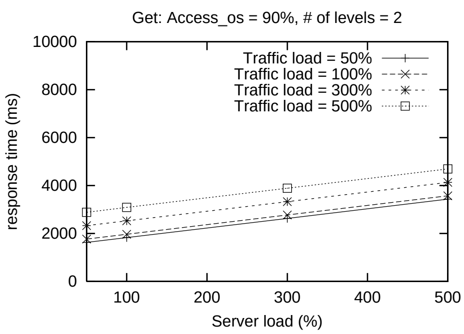


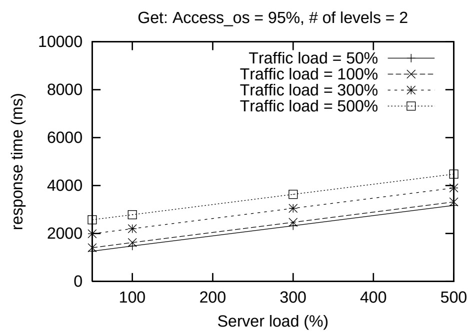
Simple-3, Svr-tfc:

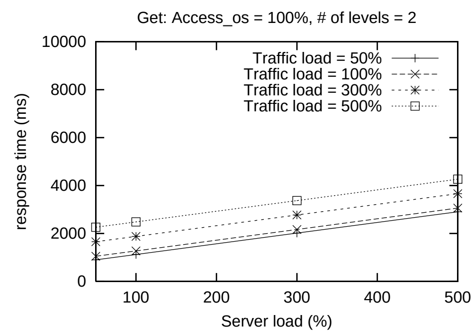


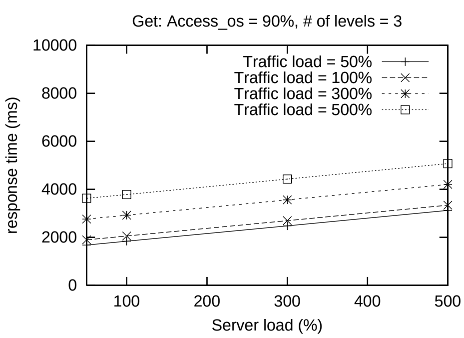


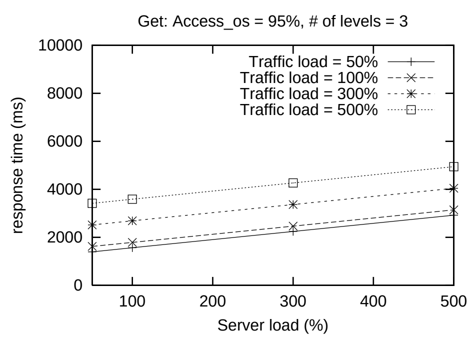


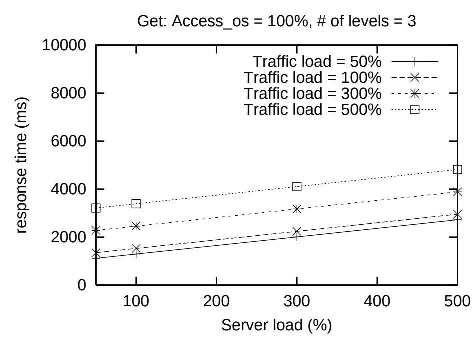


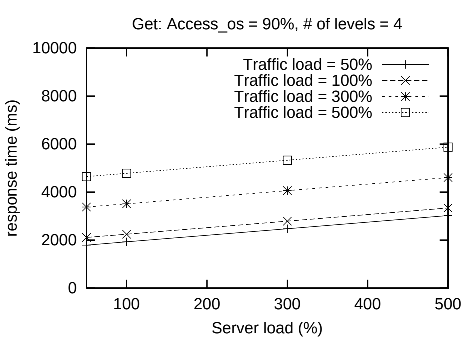


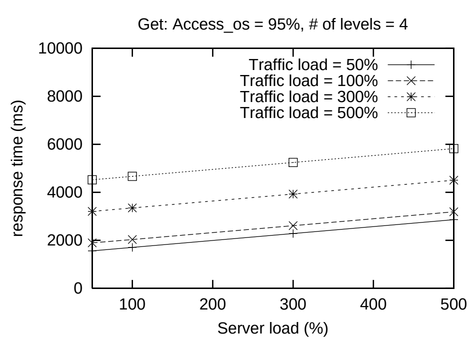


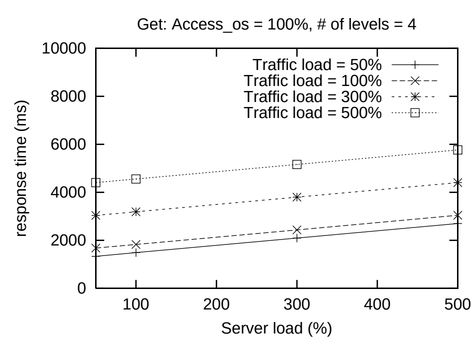




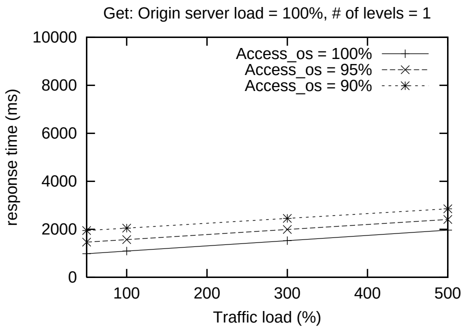


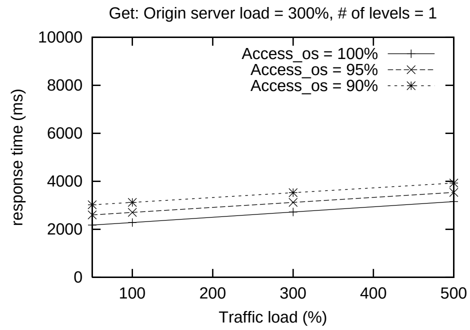


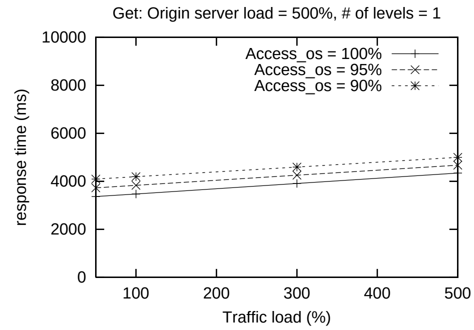


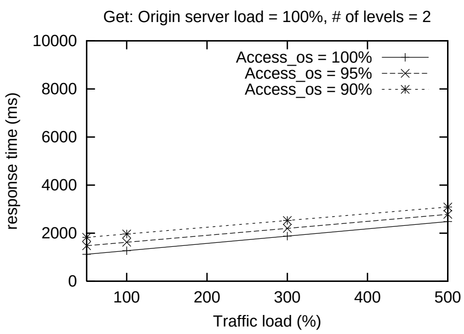


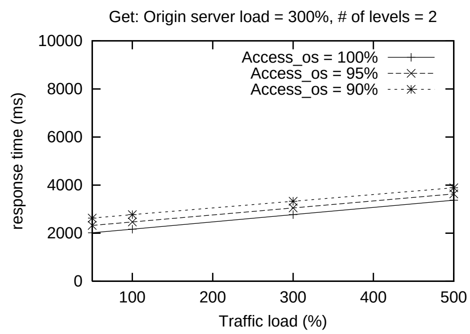
Simple-3, Tfc-access:

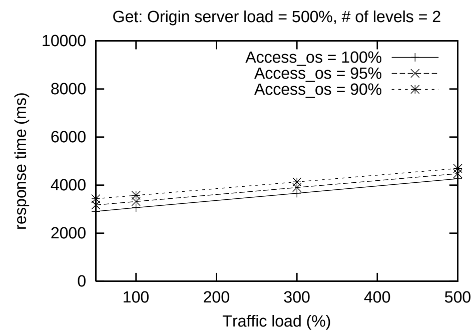


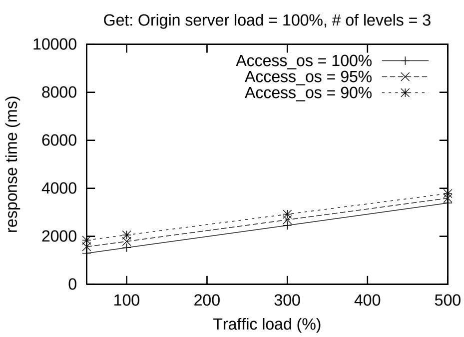


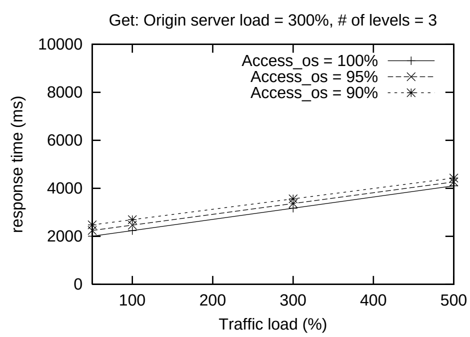


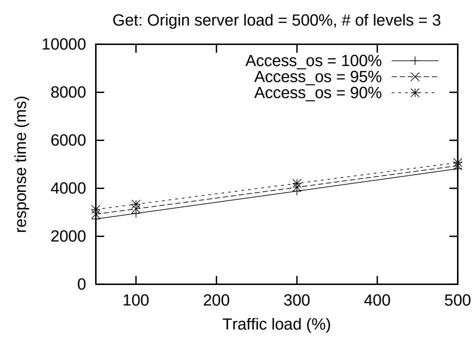


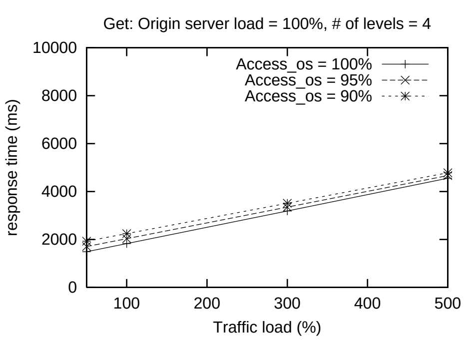


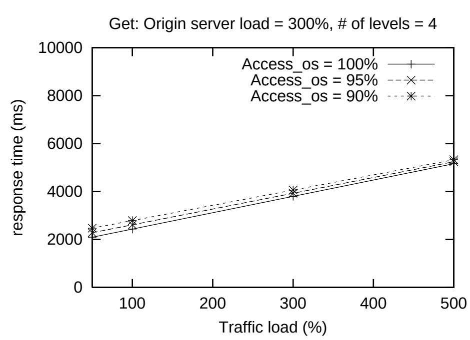


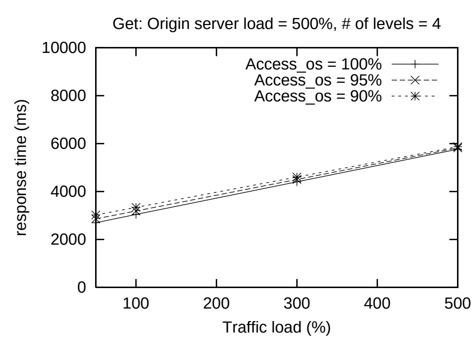




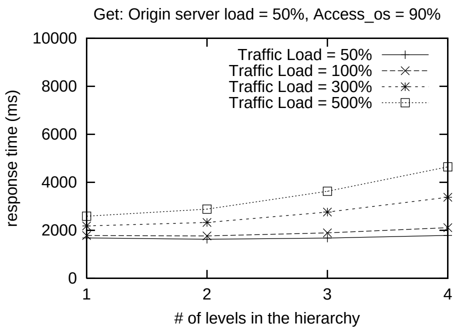


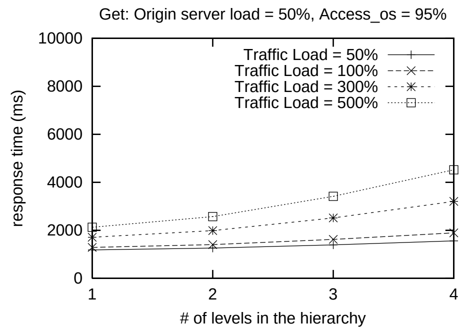


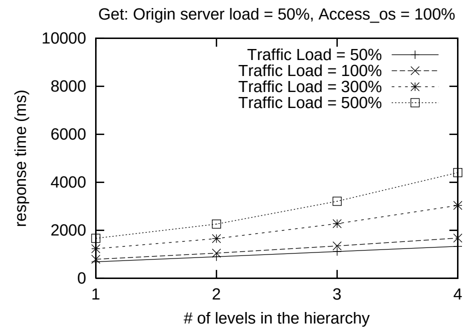


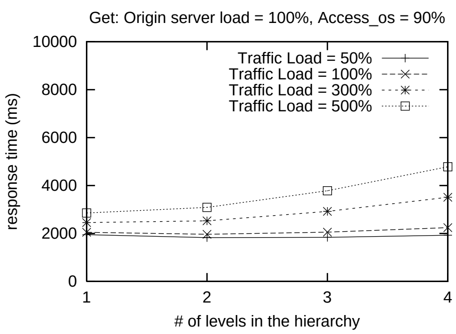


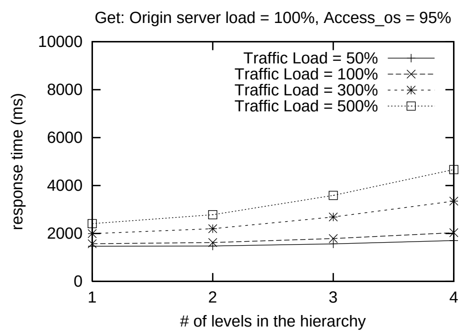
Simple-3, tfc-Level:

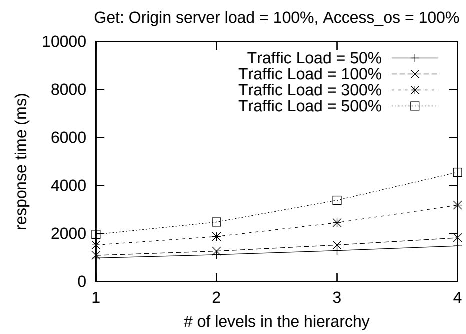


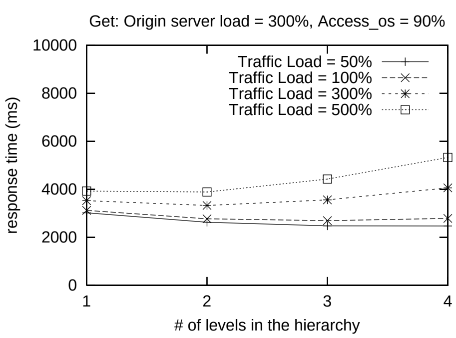


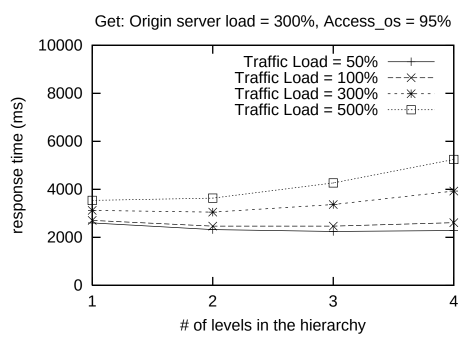


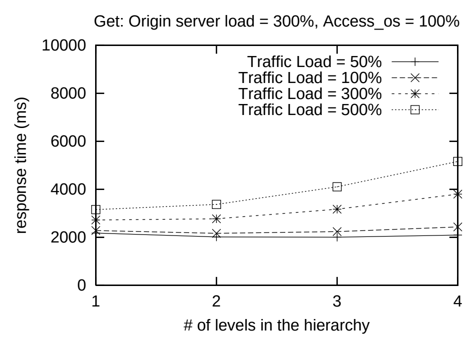


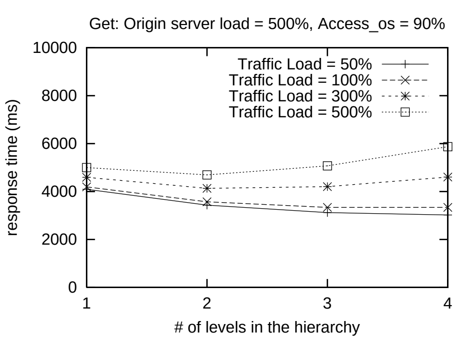


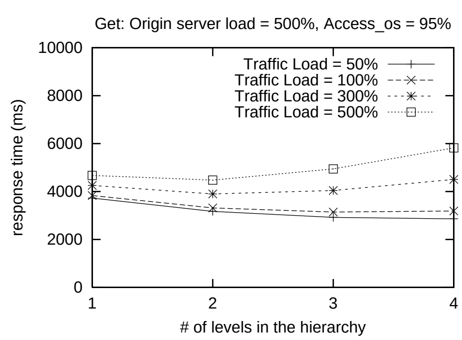


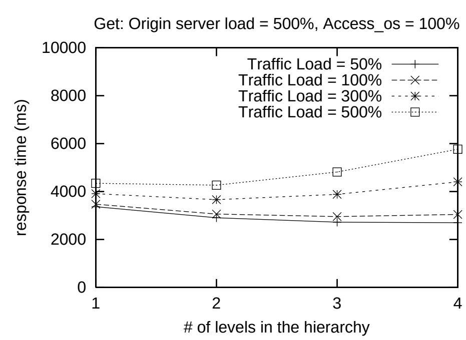




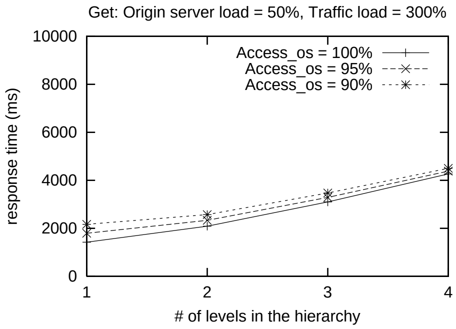


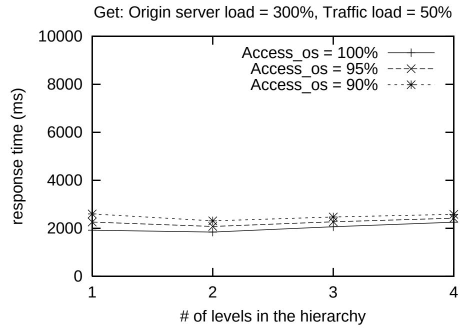


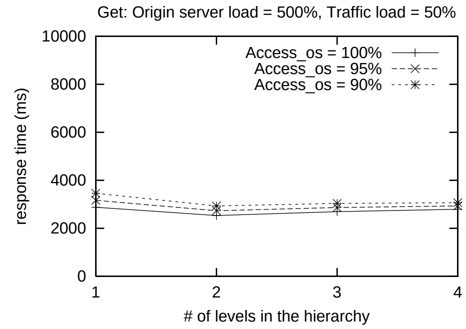


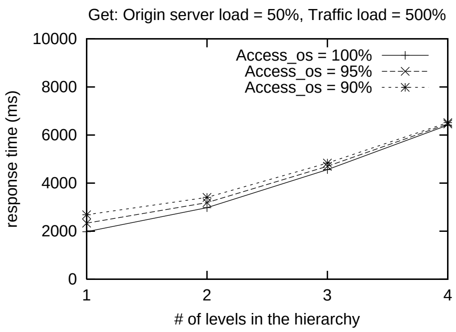


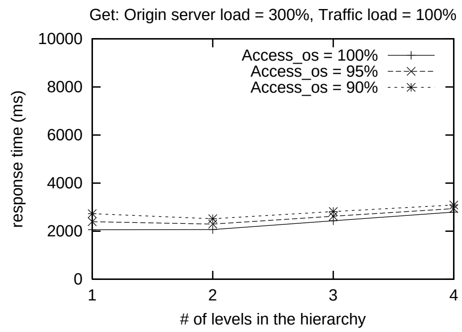
ICP-1, access-Level:

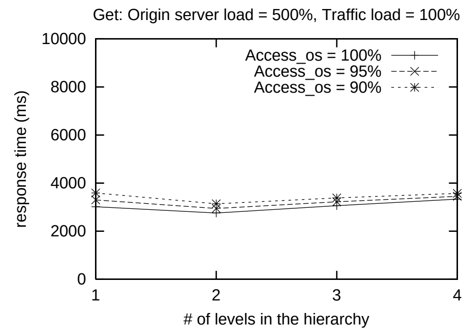


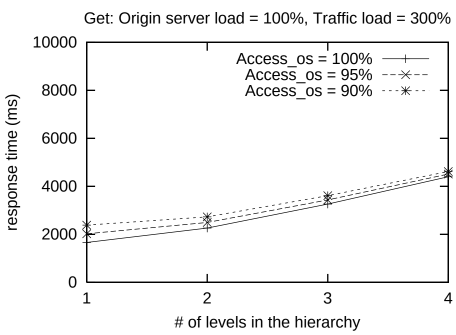


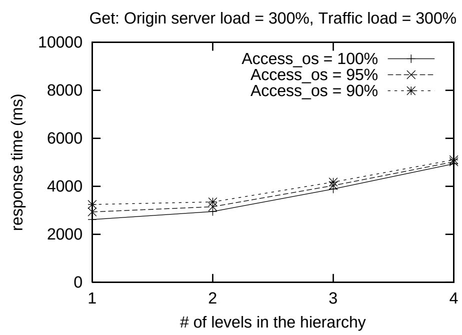


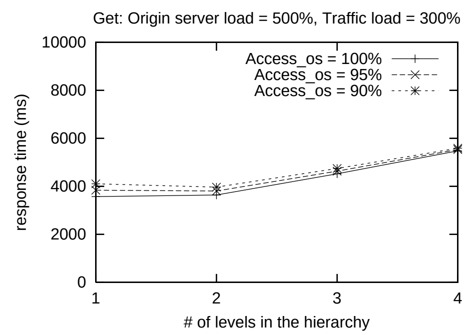


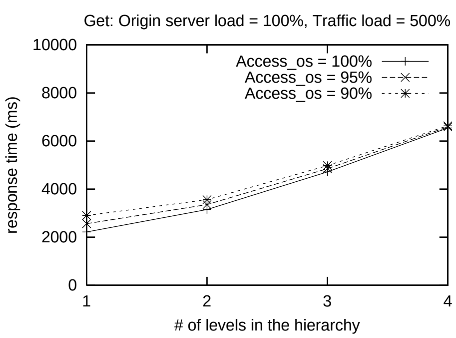


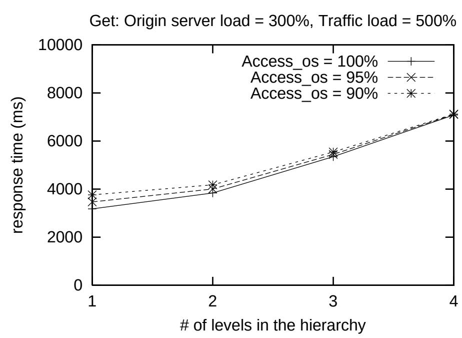


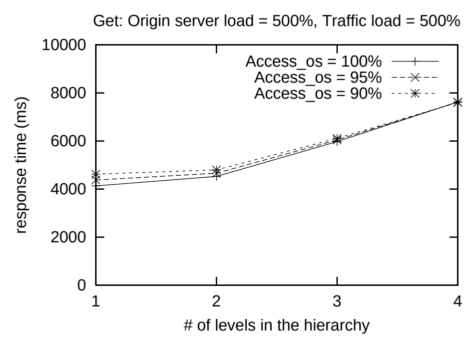




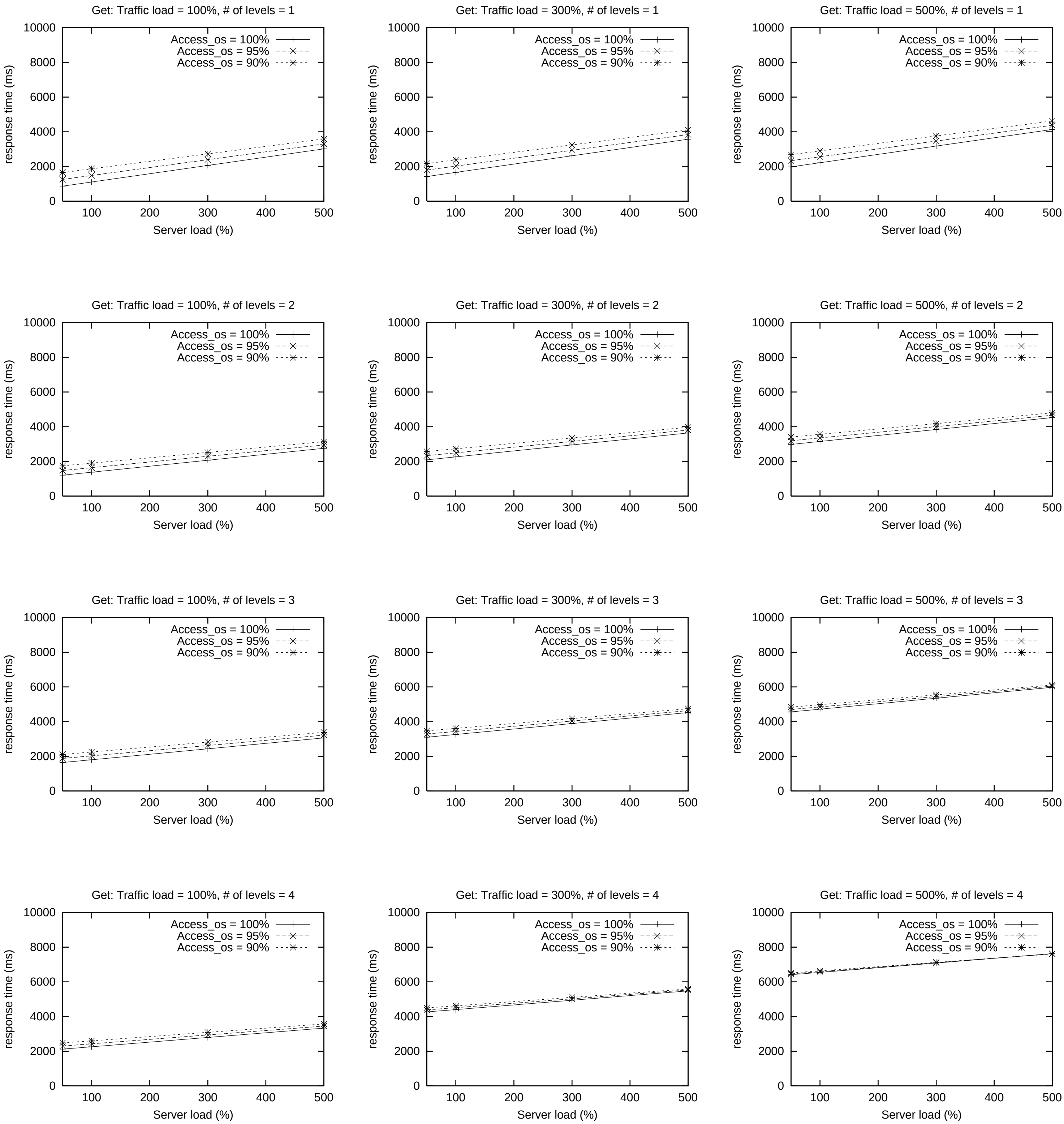




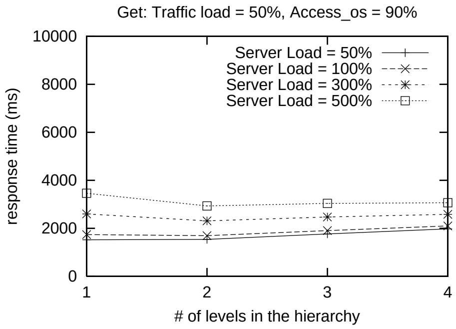


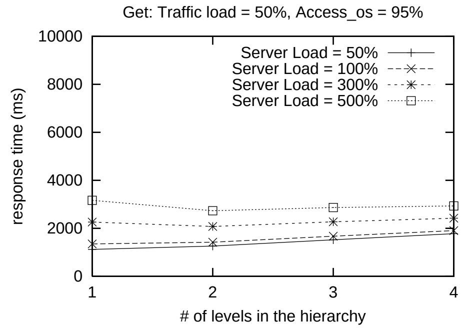


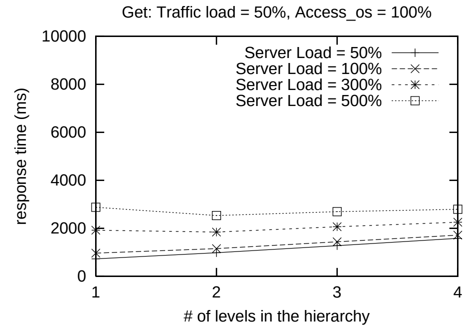
ICP-1, Svr-access:

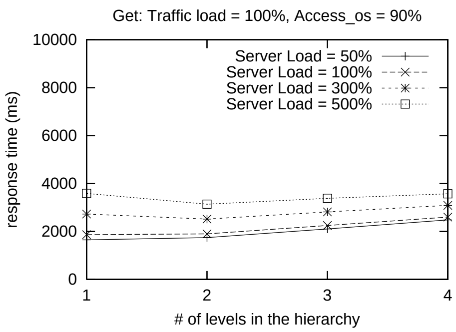


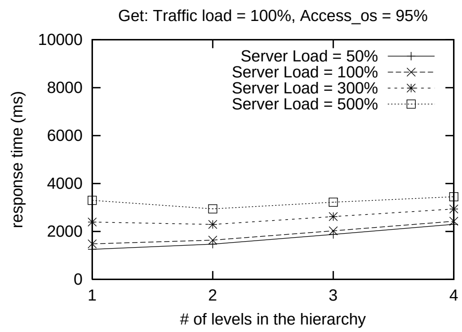
ICP-1, svr-Level:

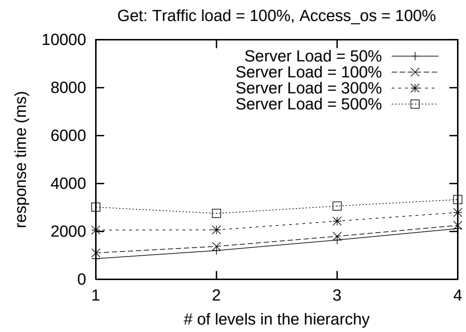


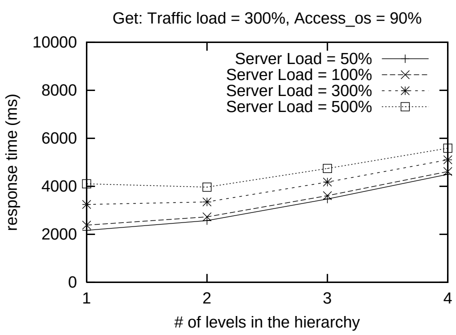


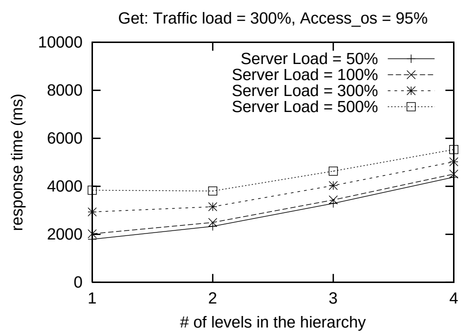


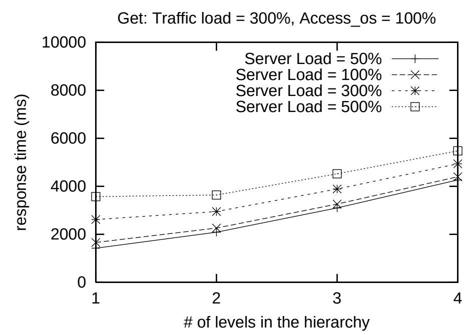


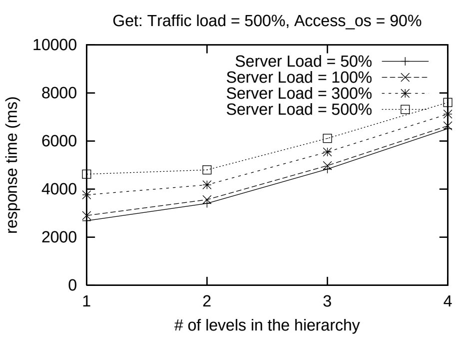


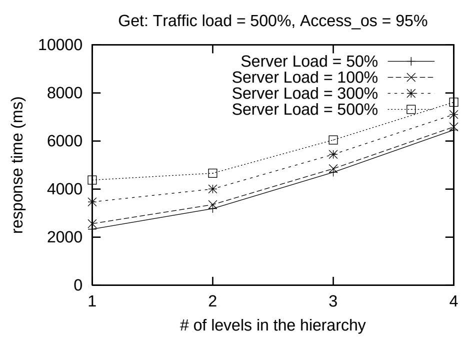


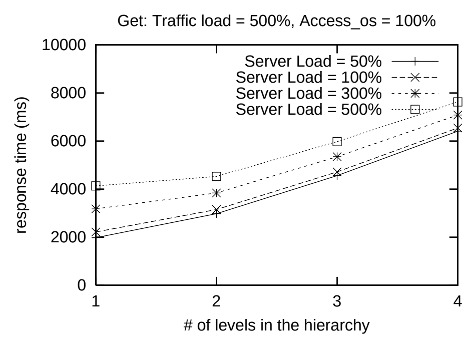




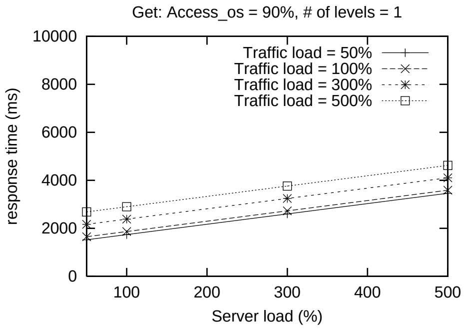


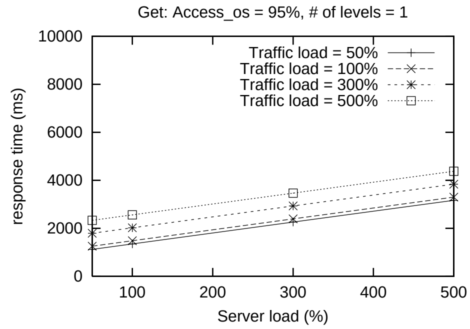


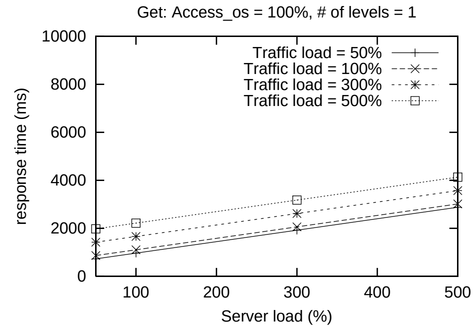


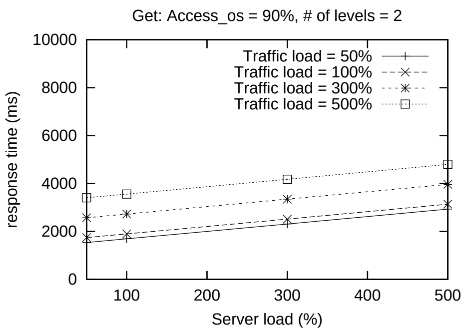


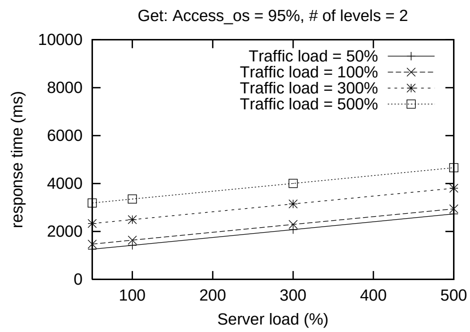
ICP-1, Svr-tfc:

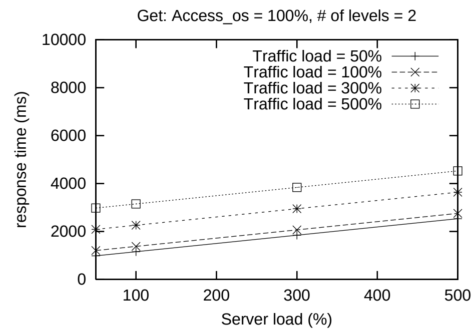

Get: Access_os = 90%, # of levels = 1
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

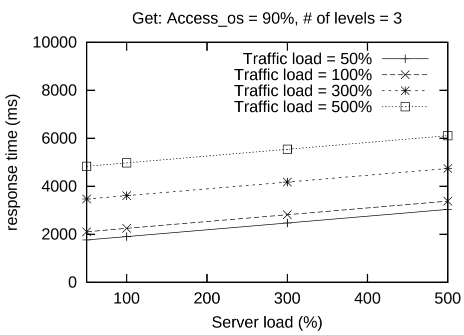

Get: Access_os = 95%, # of levels = 1
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

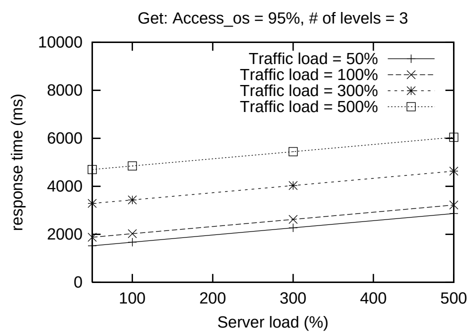

Get: Access_os = 100%, # of levels = 1
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

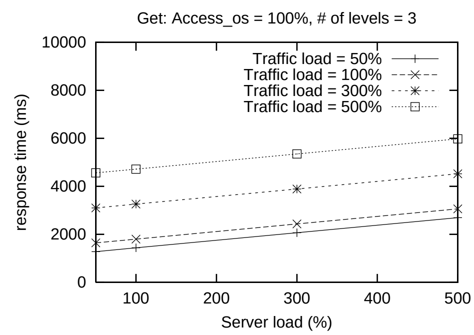

Get: Access_os = 90%, # of levels = 2
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

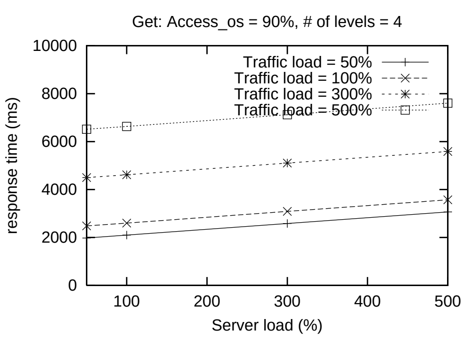

Get: Access_os = 95%, # of levels = 2
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

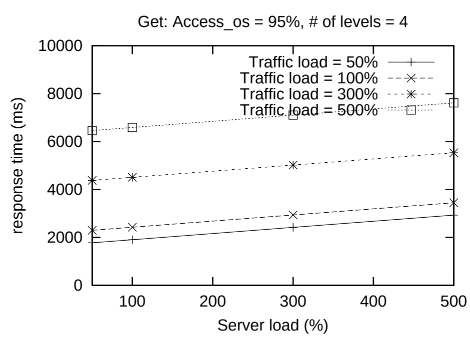

Get: Access_os = 100%, # of levels = 2
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

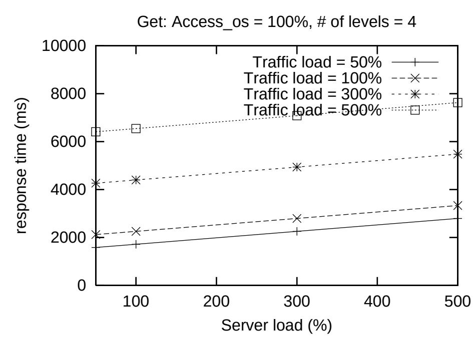

Get: Access_os = 90%, # of levels = 3
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)


Get: Access_os = 95%, # of levels = 3
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

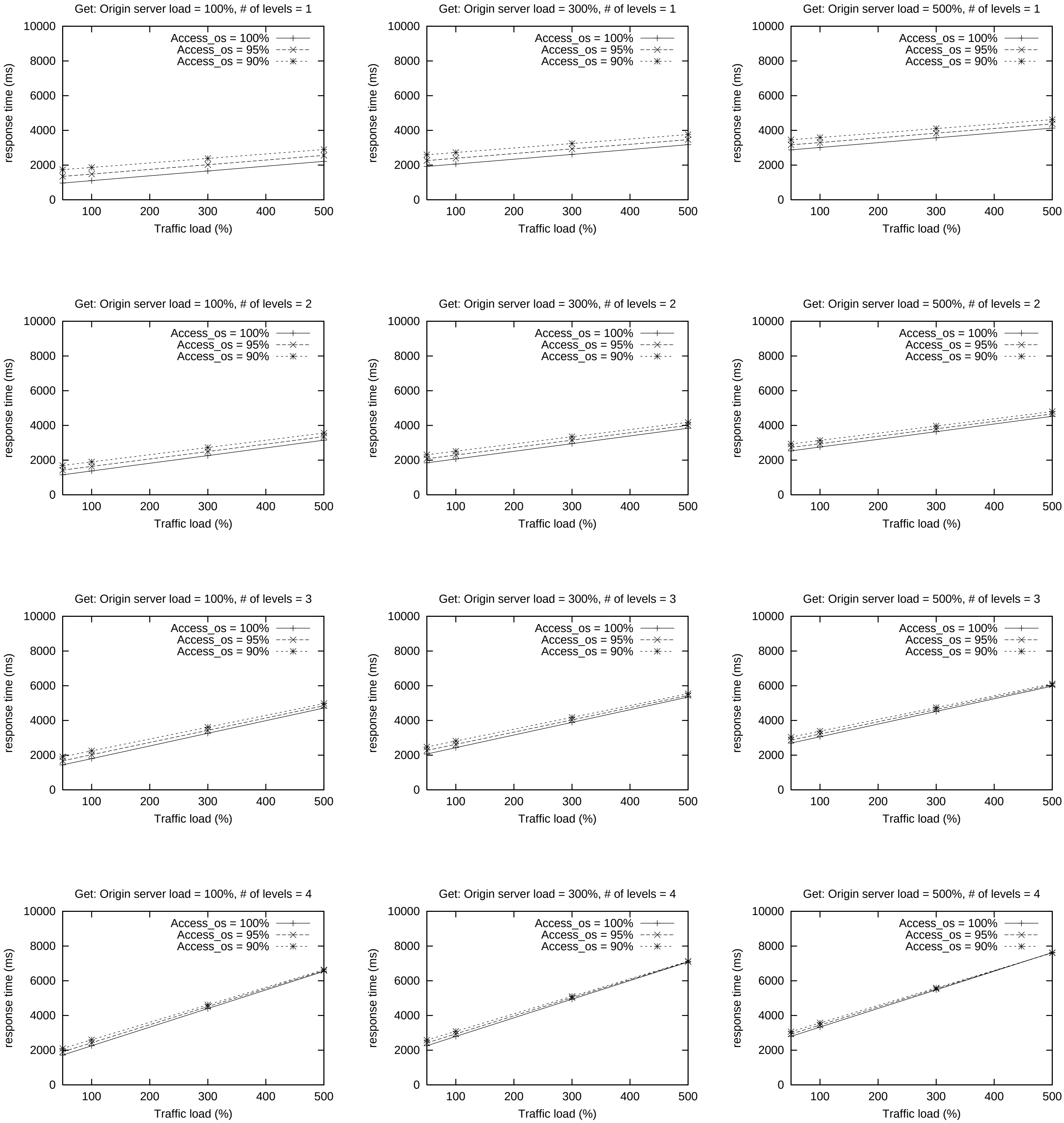

Get: Access_os = 100%, # of levels = 3
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)


Get: Access_os = 90%, # of levels = 4
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

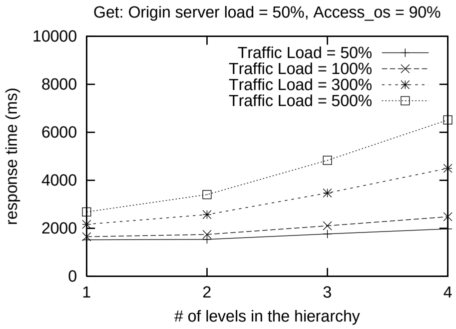

Get: Access_os = 95%, # of levels = 4
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

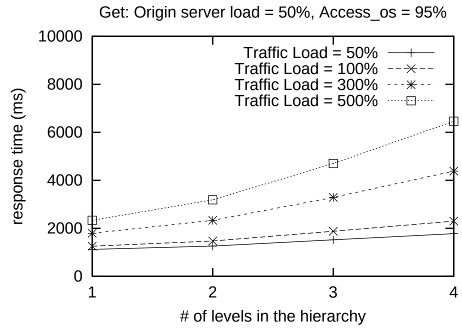

Get: Access_os = 100%, # of levels = 4
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

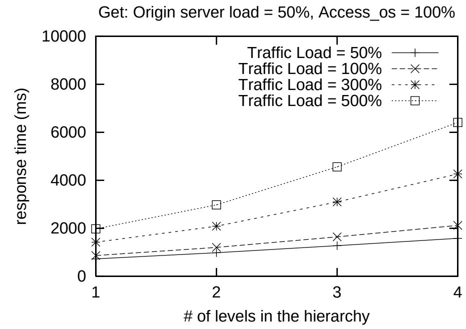
ICP-1, Tfc-access:

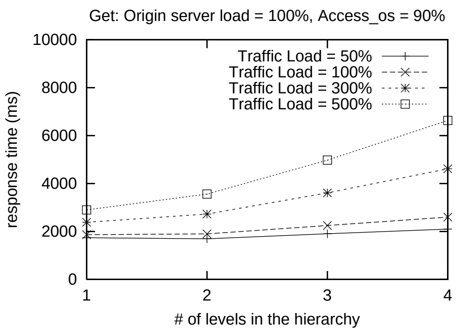


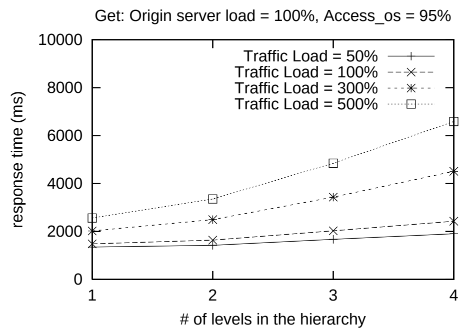
ICP-1, tfc-Level:

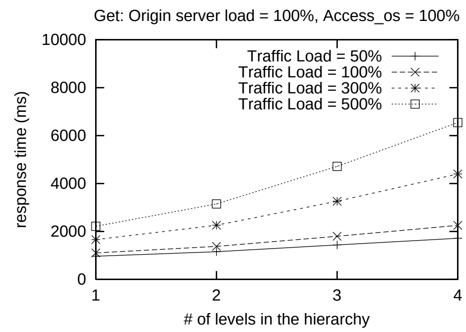


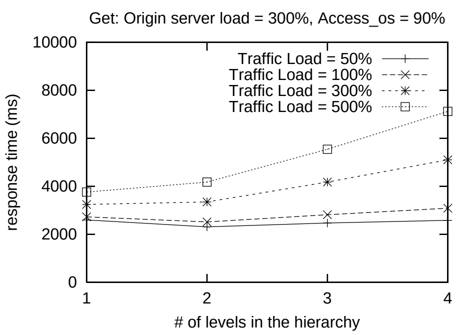


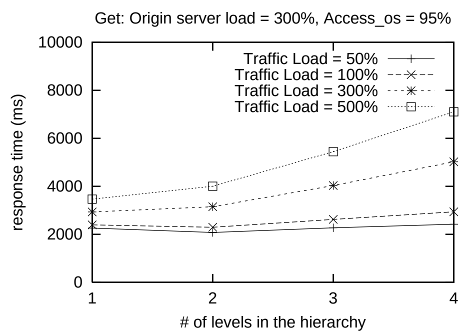


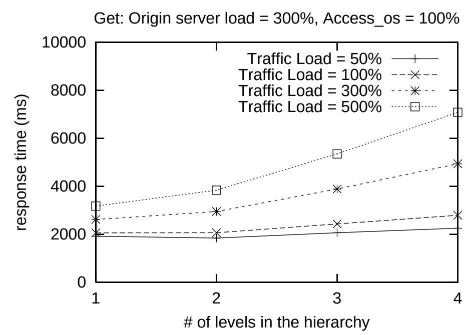


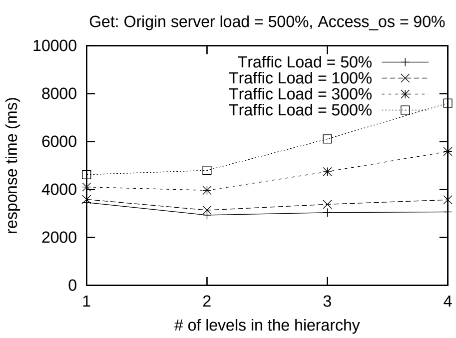


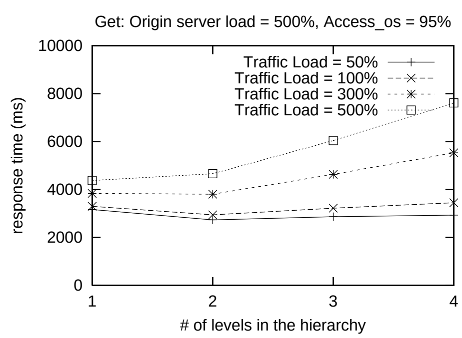


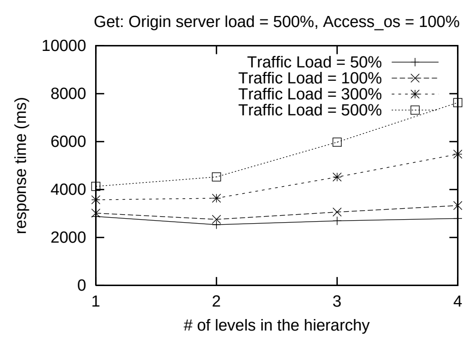




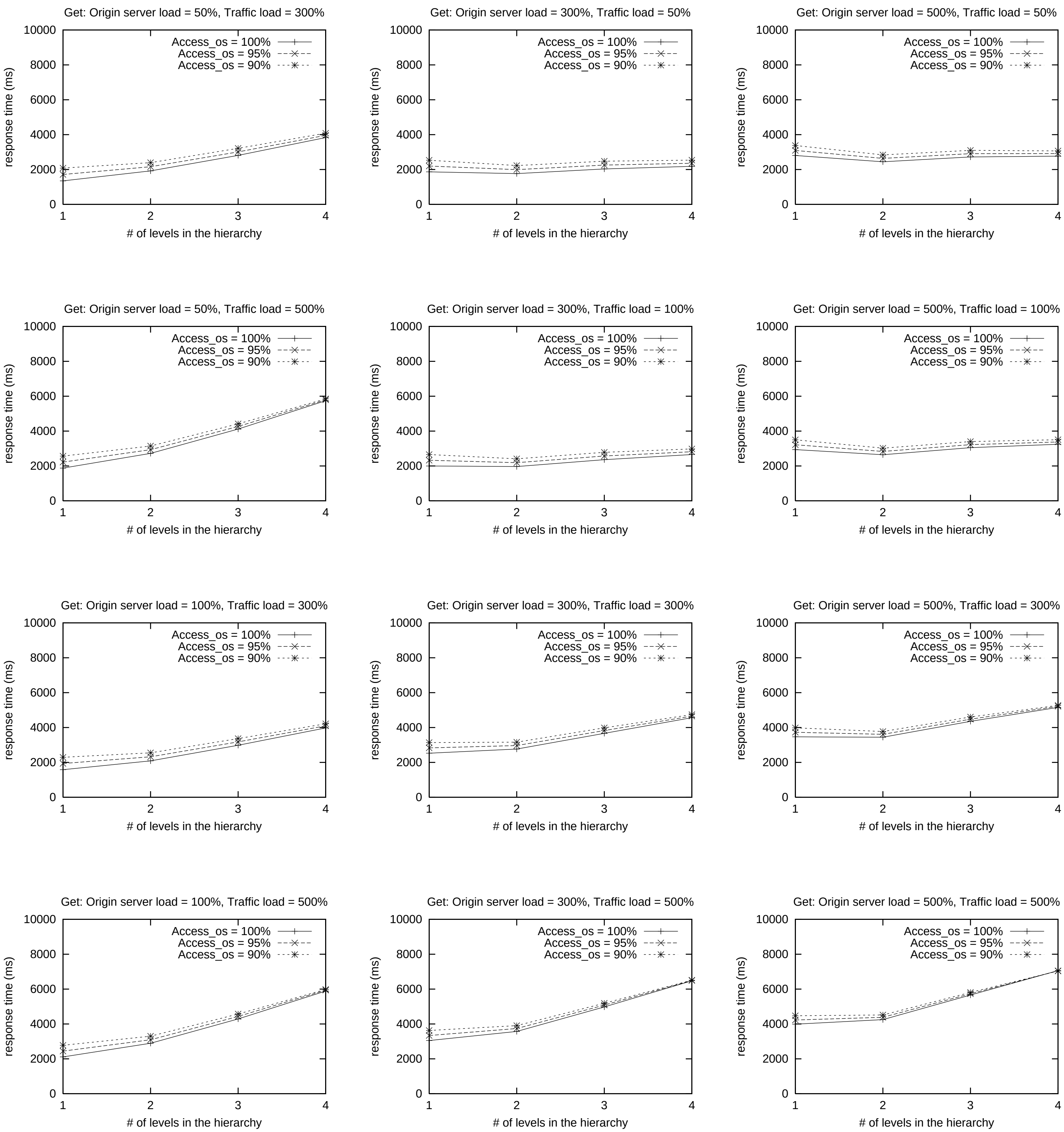




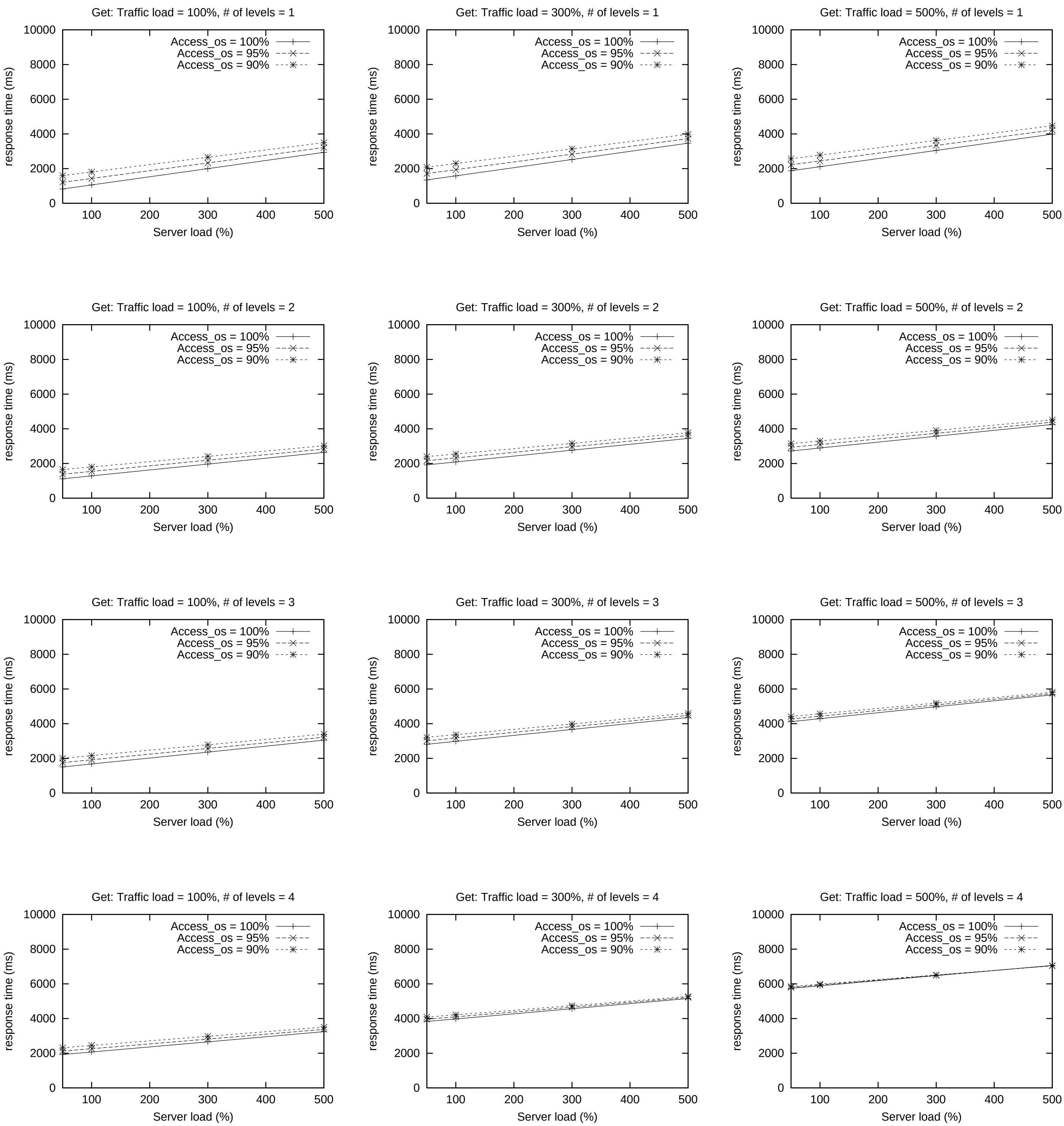




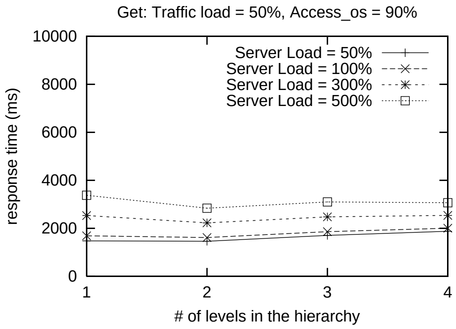
ICP-2, access-Level:

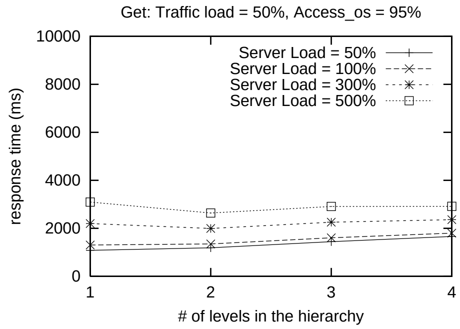


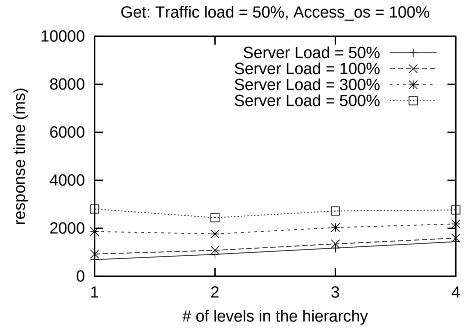
ICP-2, Svr-access:

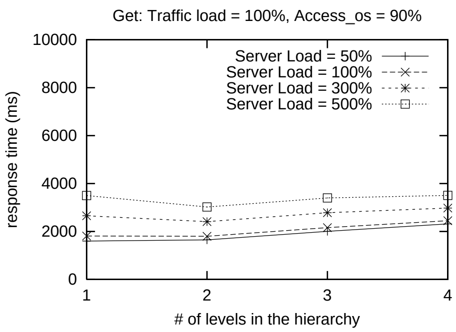


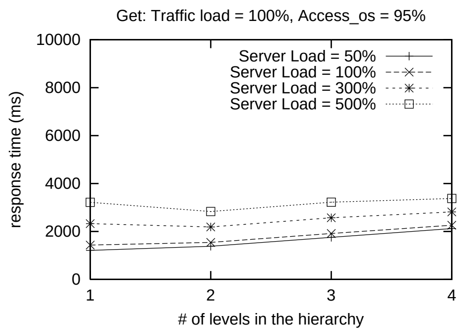
ICP-2, svr-Level:

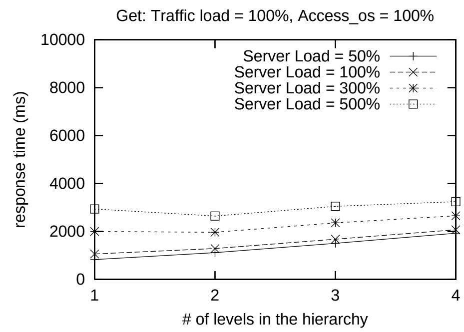


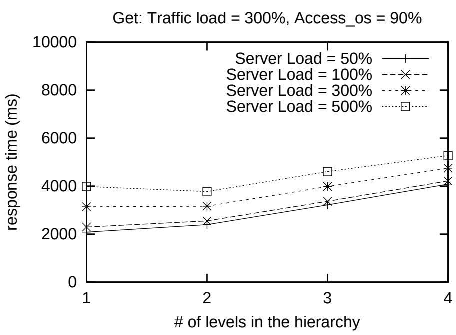


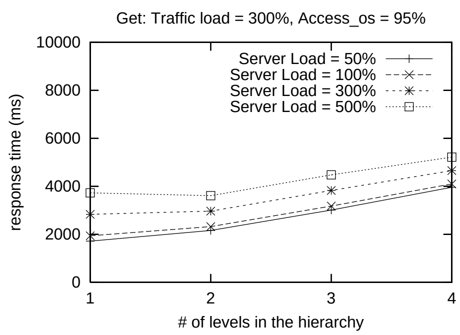


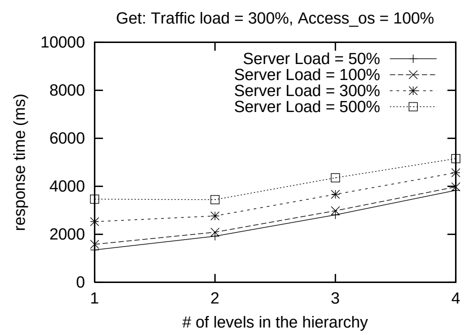


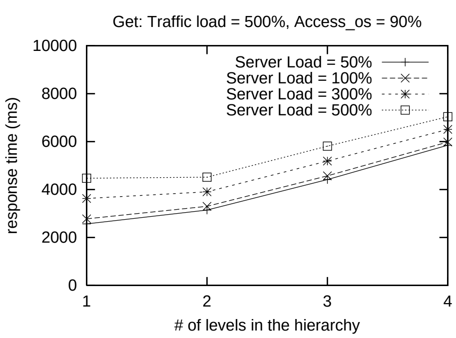


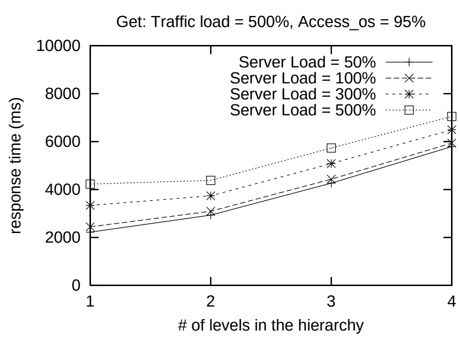


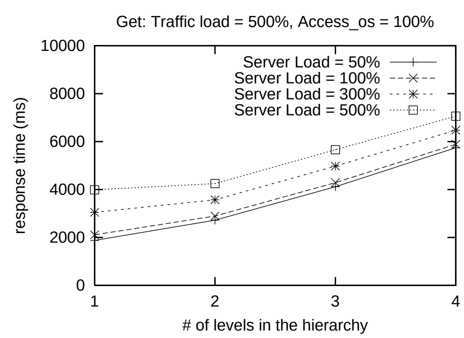




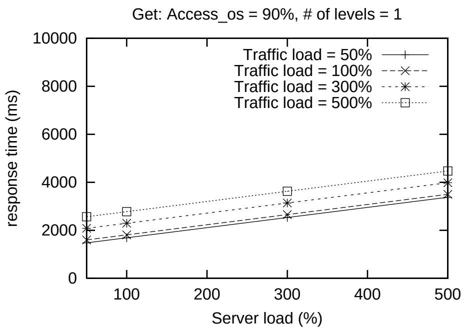


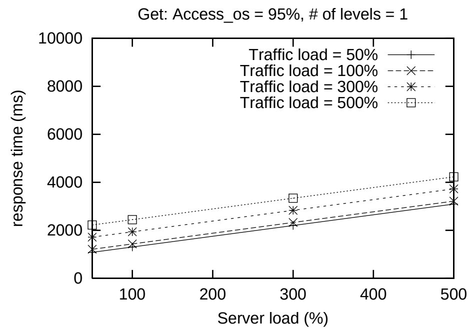


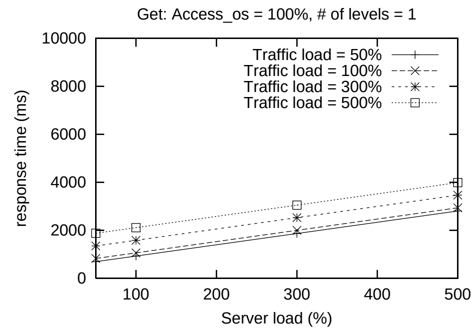


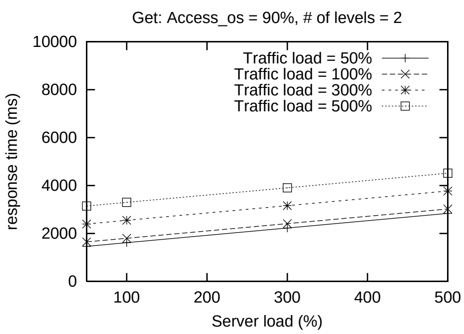


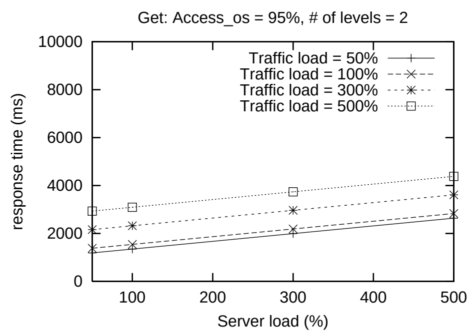
ICP-2, Svr-tfc:

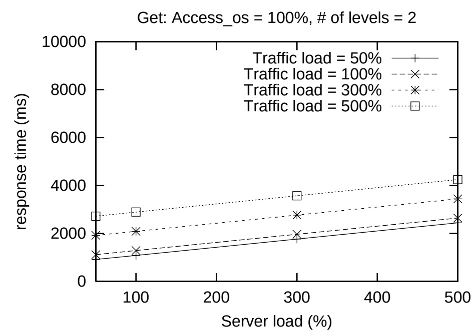

Get: Access_os = 90%, # of levels = 1
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

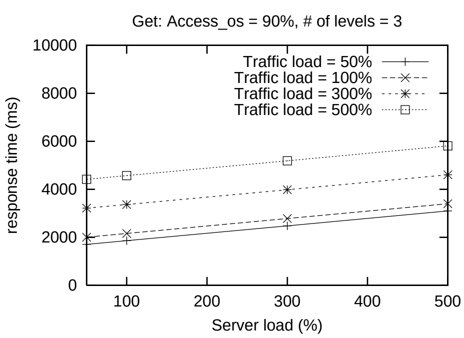

Get: Access_os = 95%, # of levels = 1
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

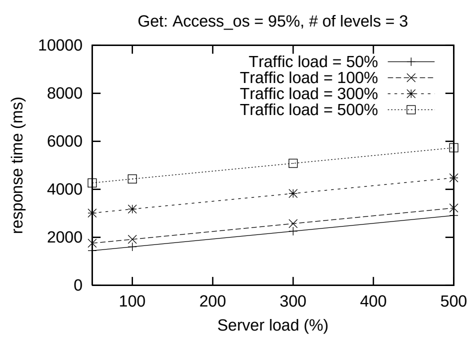

Get: Access_os = 100%, # of levels = 1
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

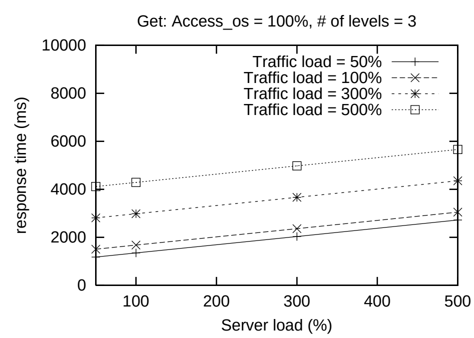

Get: Access_os = 90%, # of levels = 2
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

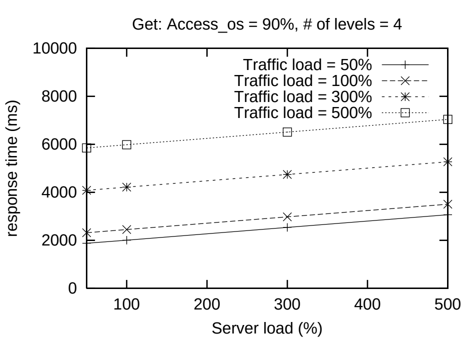

Get: Access_os = 95%, # of levels = 2
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

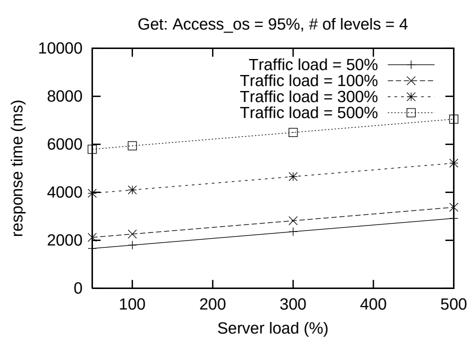

Get: Access_os = 100%, # of levels = 2
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

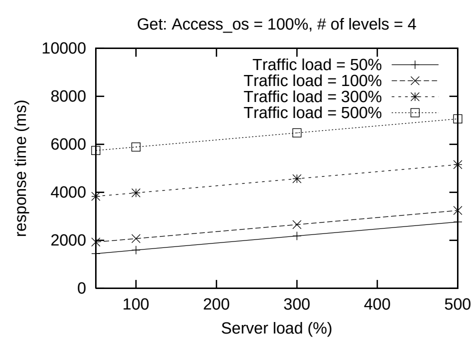

Get: Access_os = 90%, # of levels = 3
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)


Get: Access_os = 95%, # of levels = 3
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

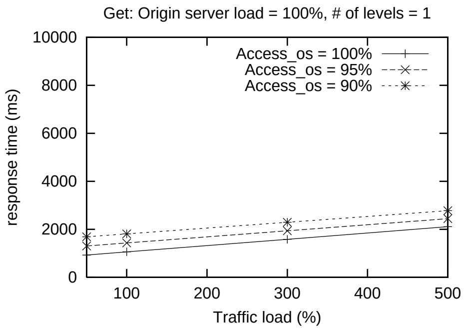

Get: Access_os = 100%, # of levels = 3
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

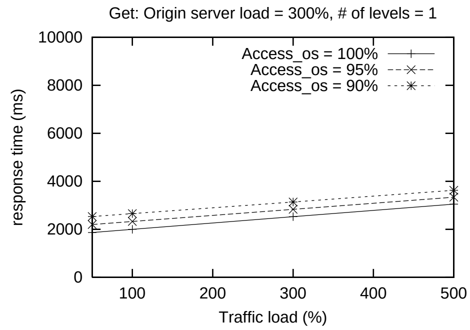

Get: Access_os = 90%, # of levels = 4
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

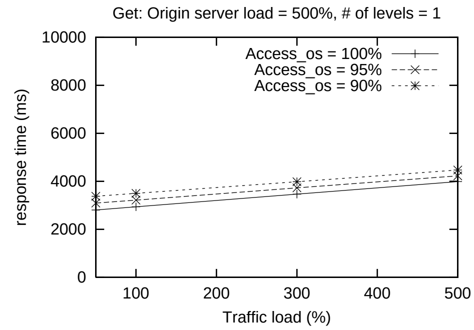

Get: Access_os = 95%, # of levels = 4
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

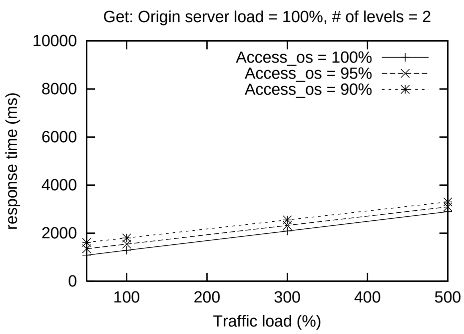

Get: Access_os = 100%, # of levels = 4
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

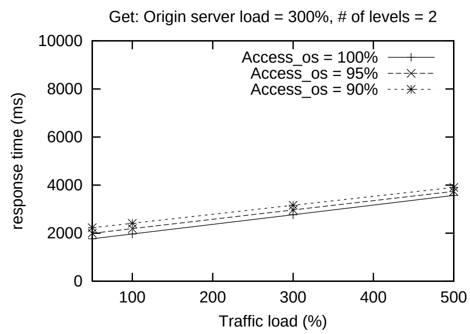
ICP-2, Tfc-access:

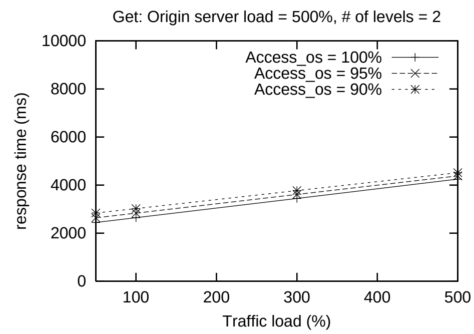


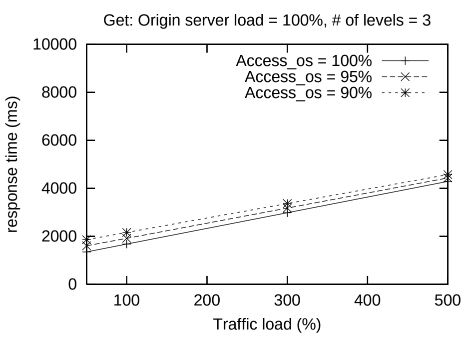


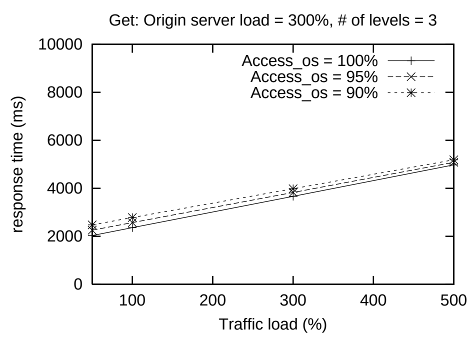


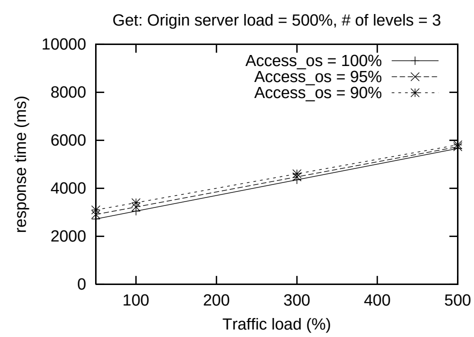


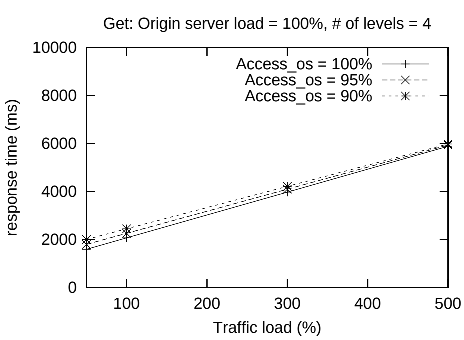


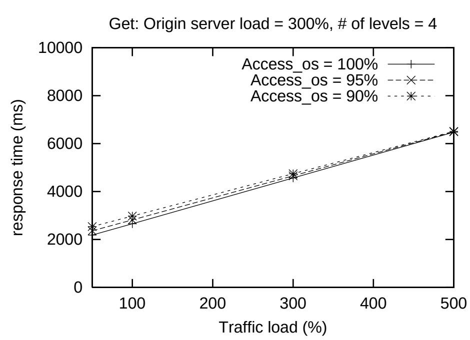


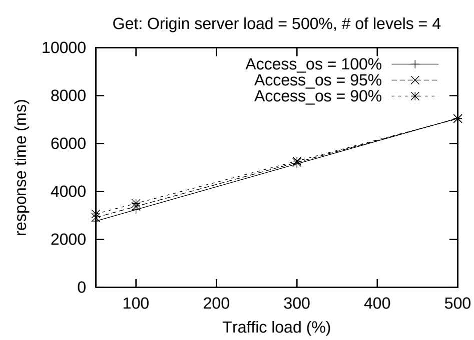




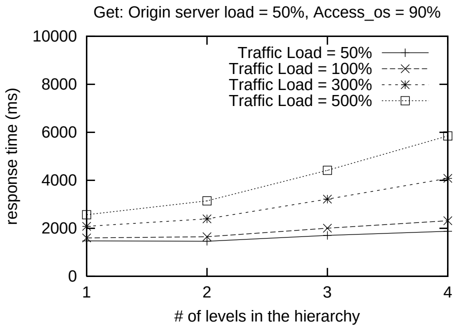


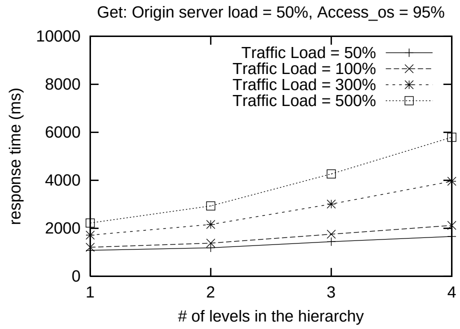


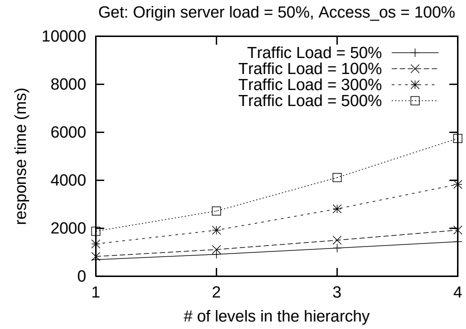


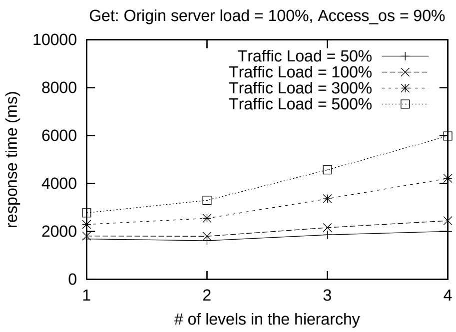


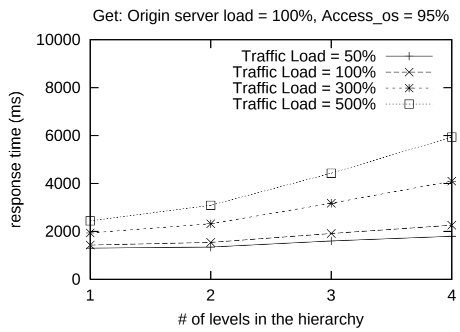
ICP-2, tfc-Level:

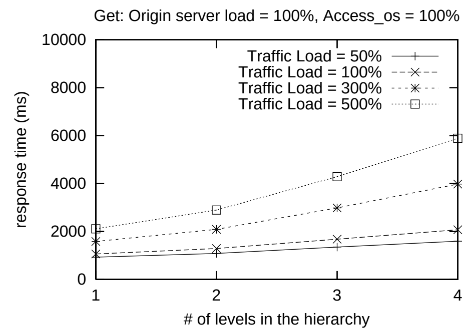


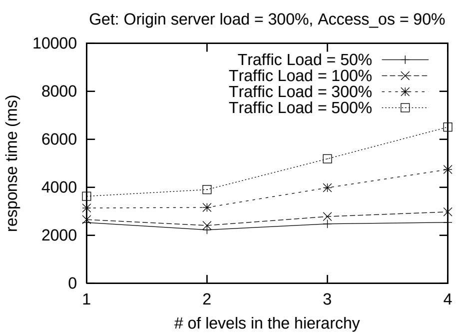


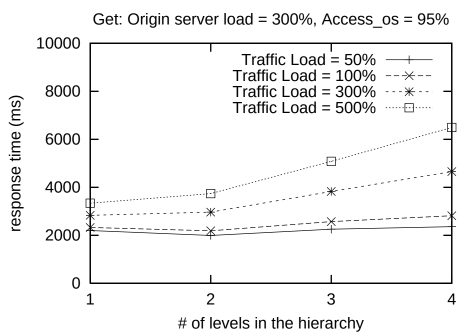


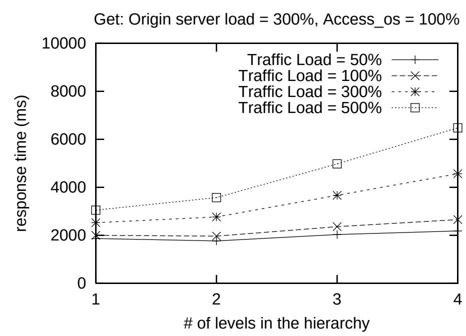


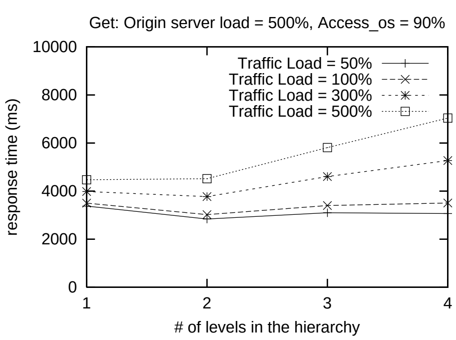


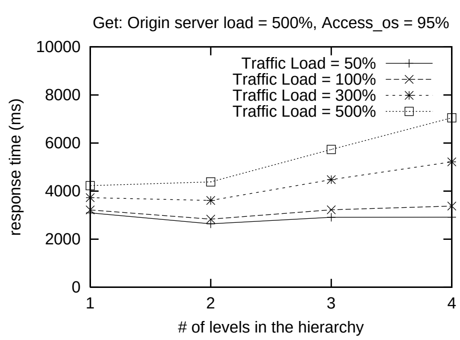


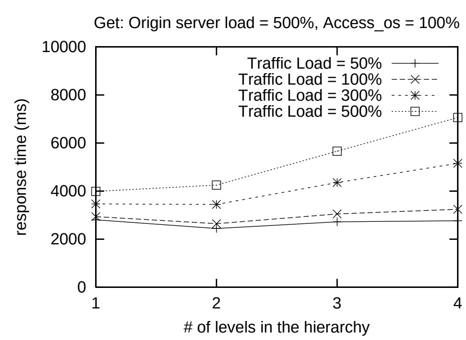




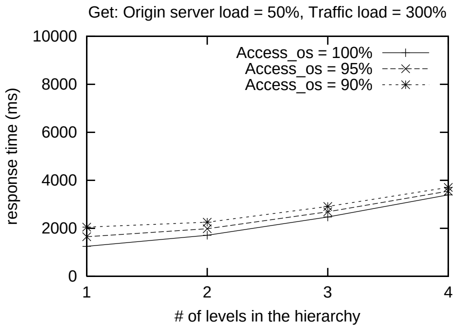


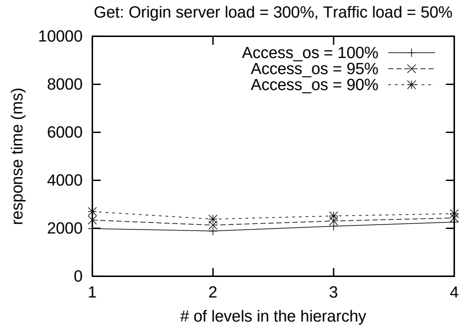


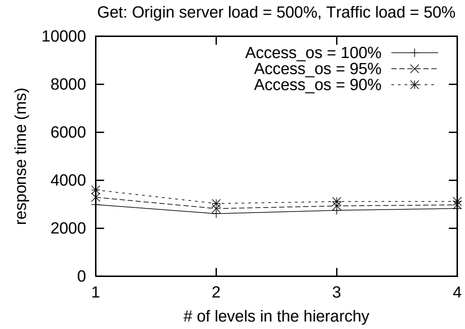


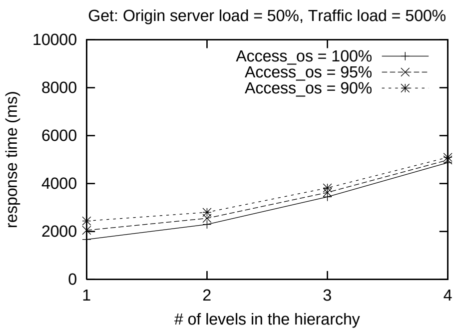


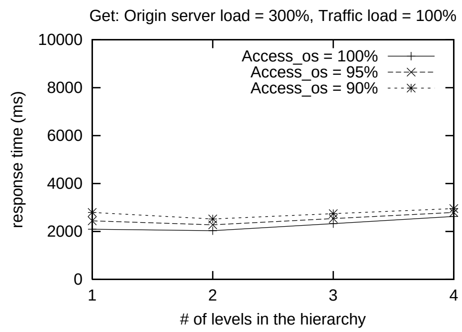
CD-1, access-Level:

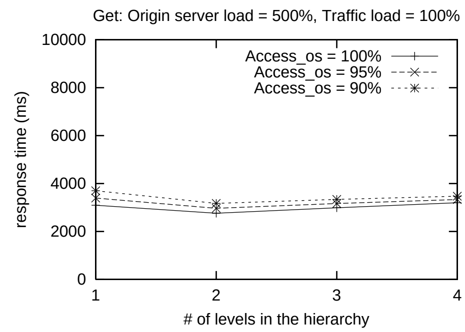


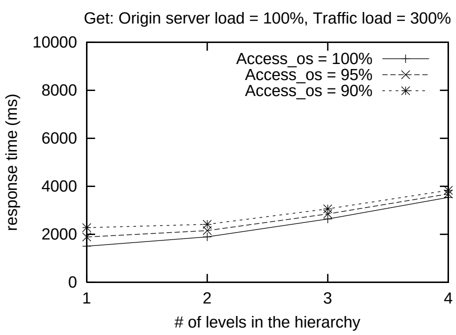


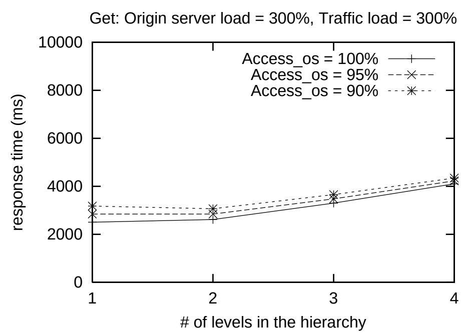


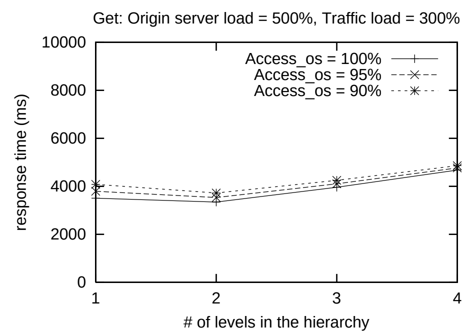


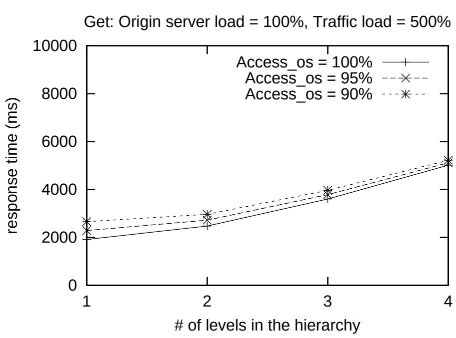


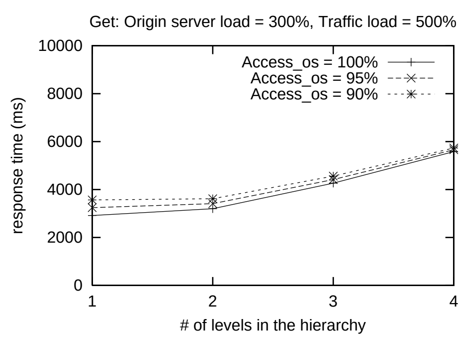


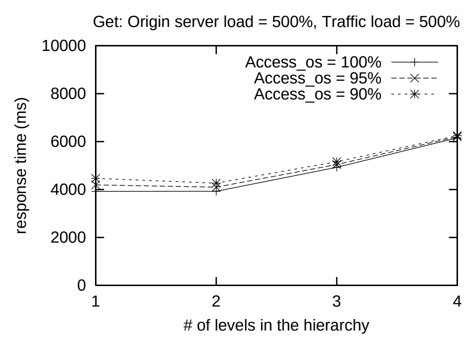




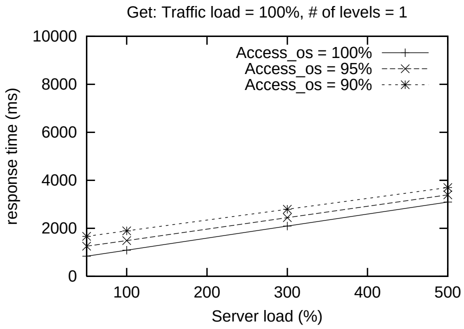


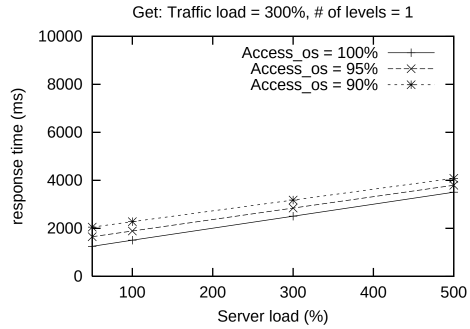


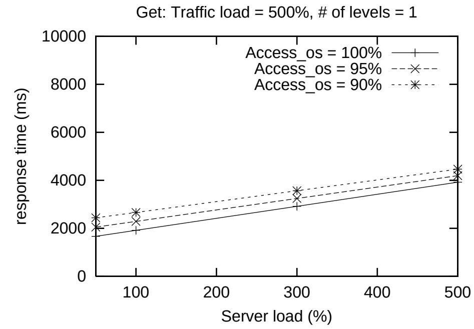


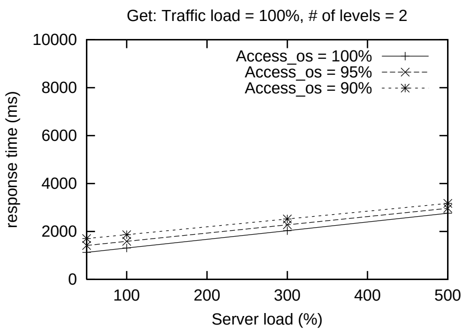


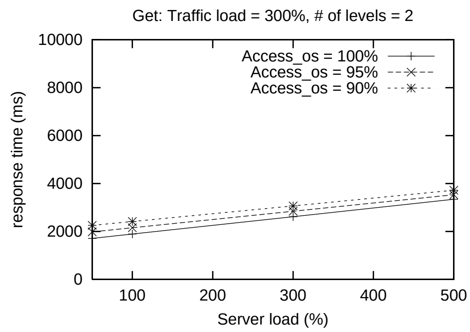
CD-1, Svr-access:

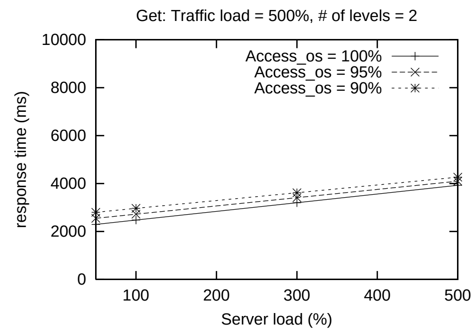


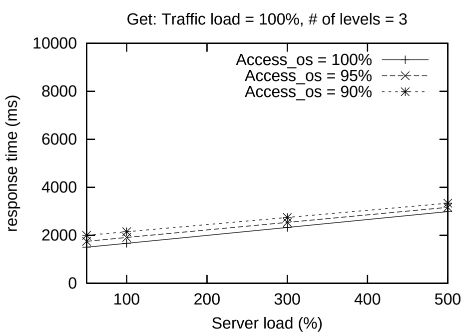


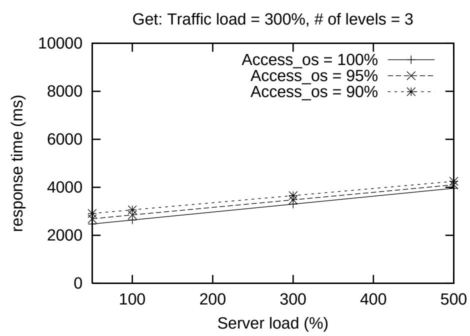


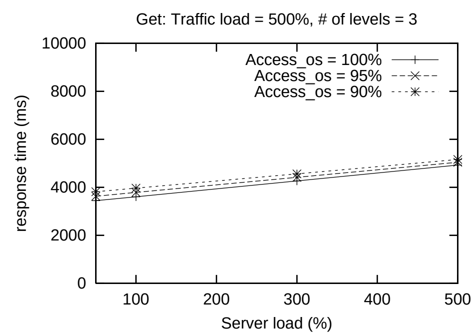


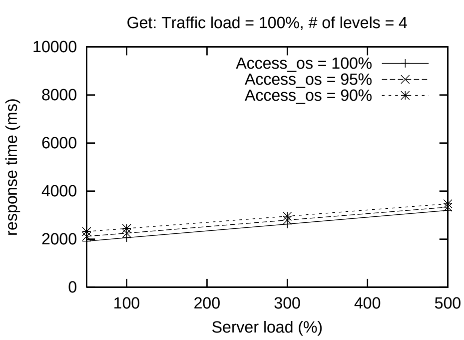


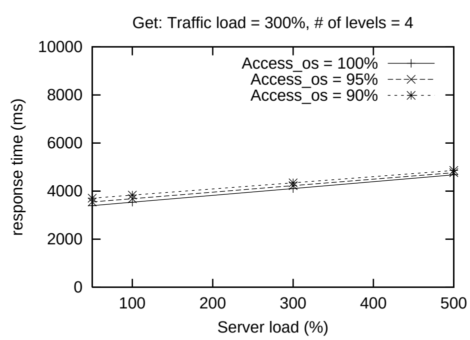


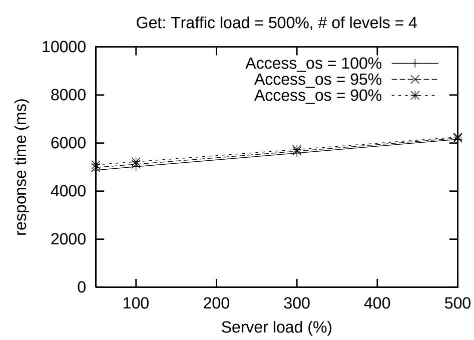




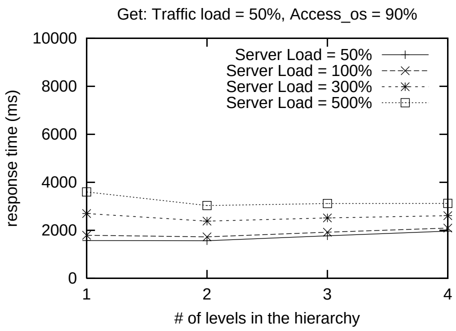


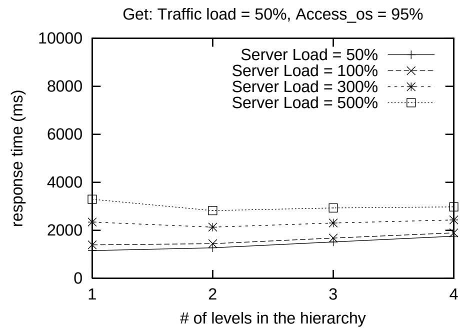


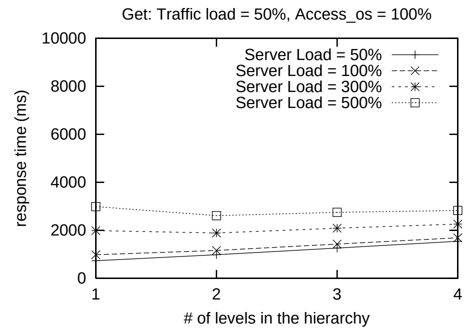


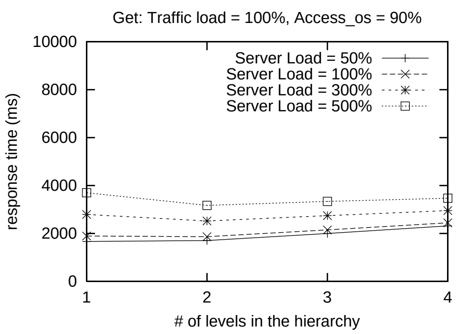


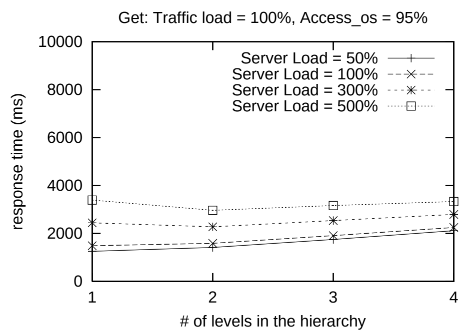
CD-1, svr-Level:

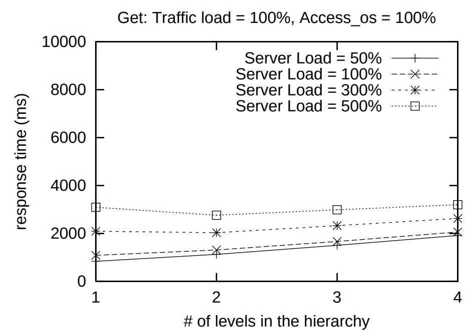


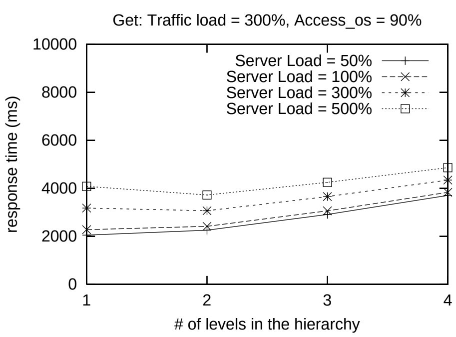


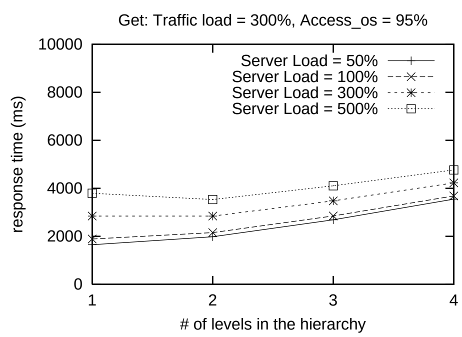


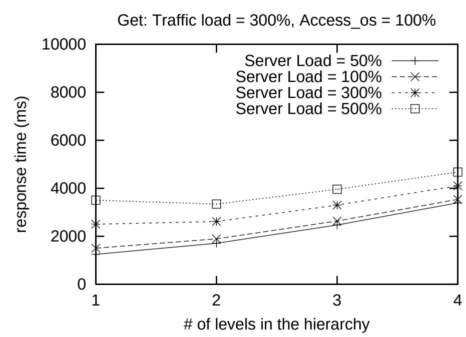


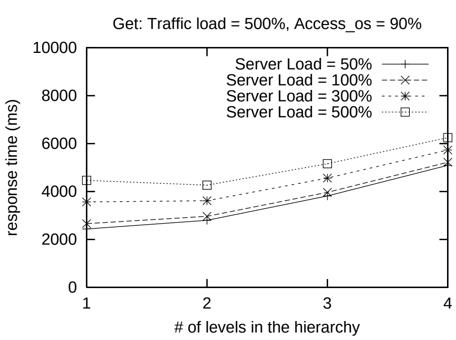


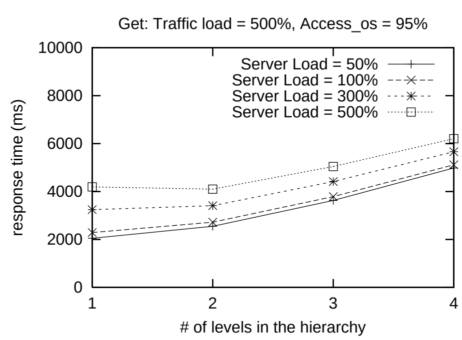


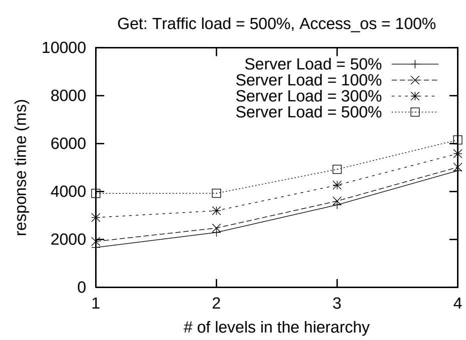




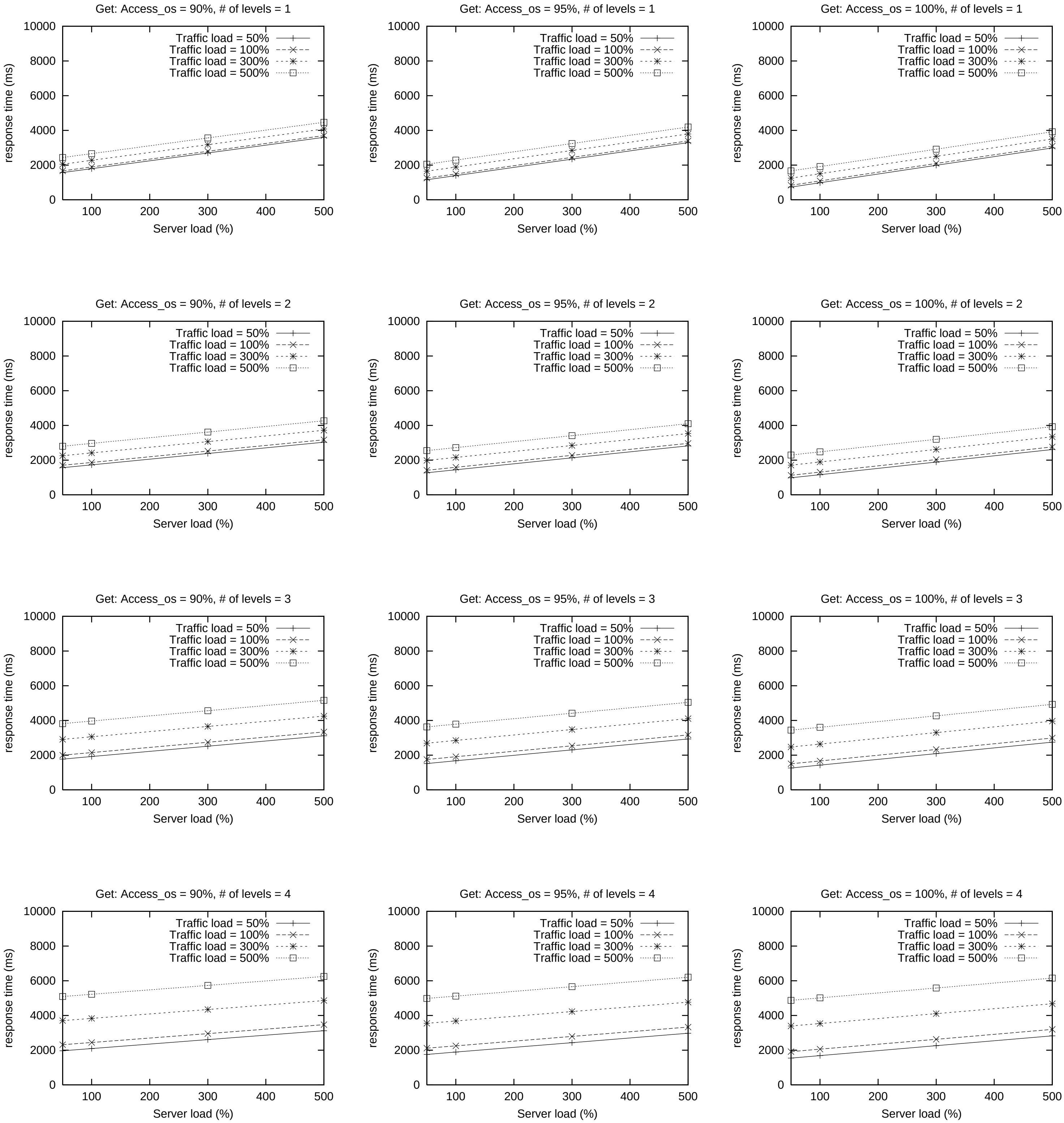




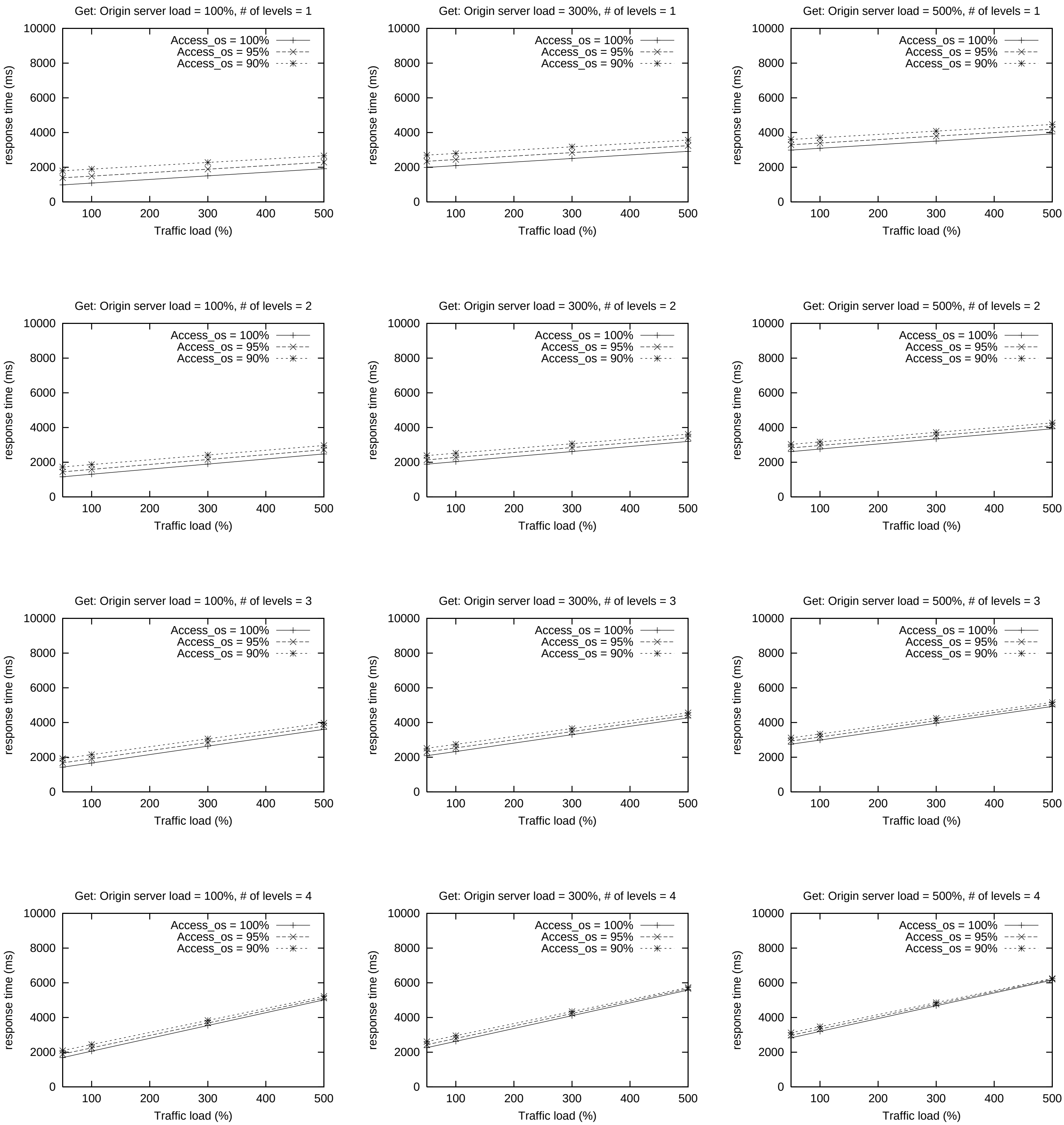




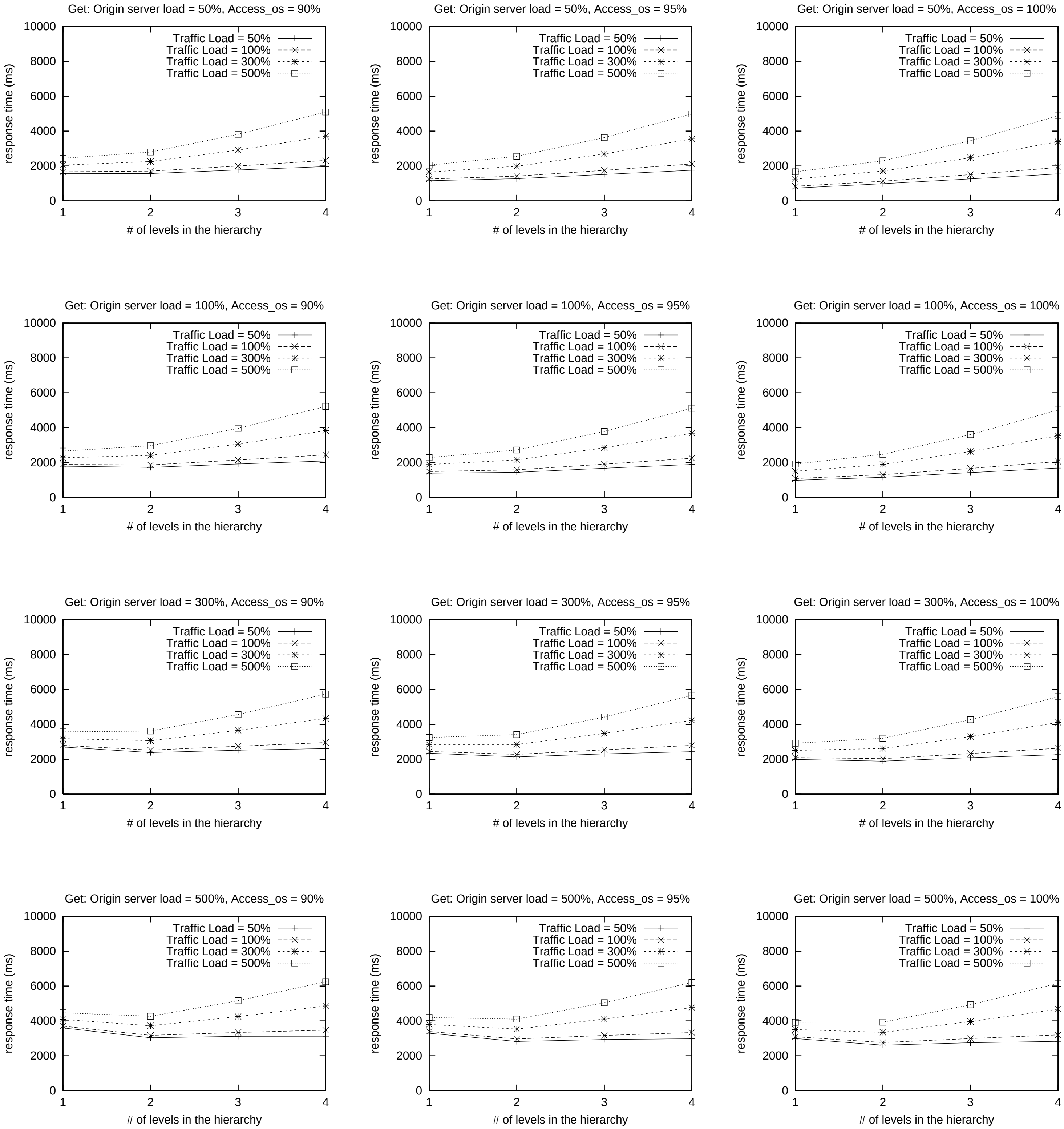
CD-1, Svr-tfc:


Get: Access_os = 90%, # of levels = 1
Get: Access_os = 95%, # of levels = 1
Get: Access_os = 100%, # of levels = 1
Get: Access_os = 90%, # of levels = 2
Get: Access_os = 95%, # of levels = 2
Get: Access_os = 100%, # of levels = 2
Get: Access_os = 90%, # of levels = 3
Get: Access_os = 95%, # of levels = 3
Get: Access_os = 100%, # of levels = 3
Get: Access_os = 90%, # of levels = 4
Get: Access_os = 95%, # of levels = 4
Get: Access_os = 100%, # of levels = 4
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

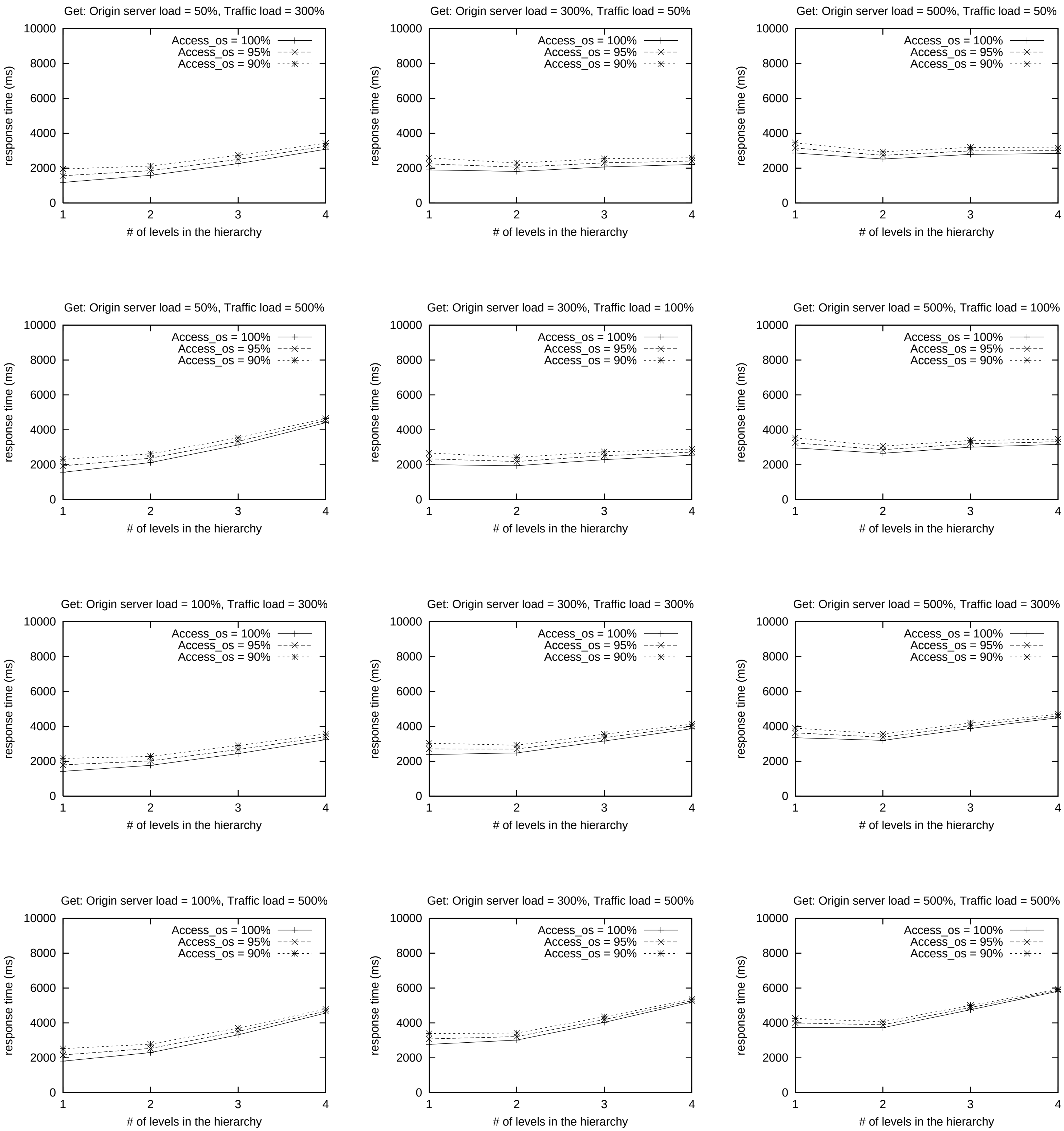
CD-1, Tfc-access:



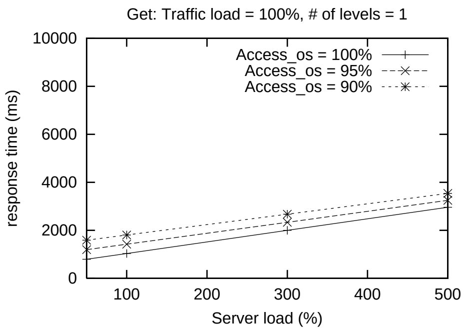
CD-1, tfc-Level:

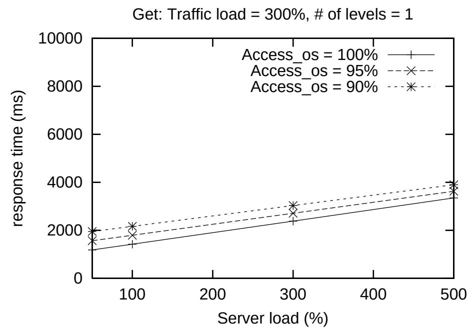


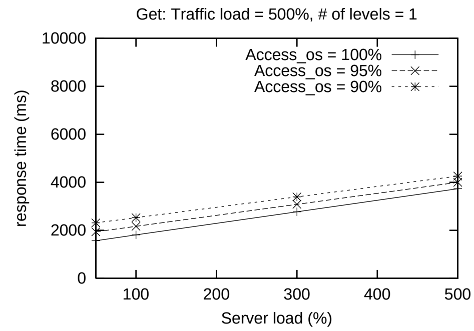
CD-2, access-Level:

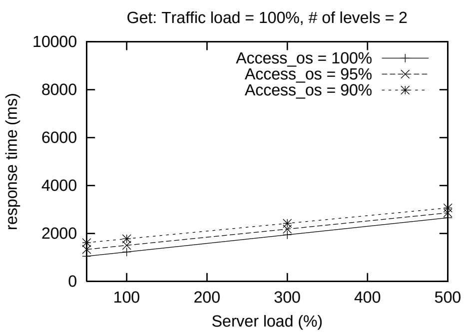


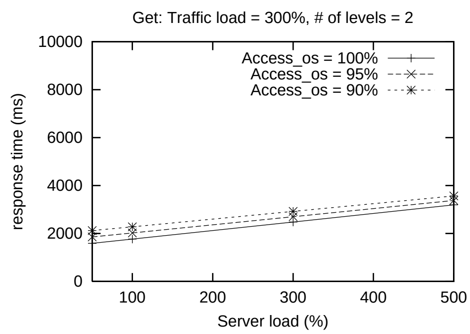
CD-2, Svr-access:

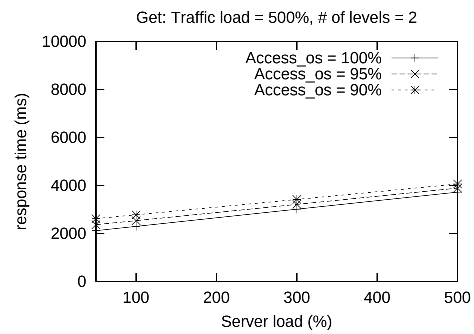


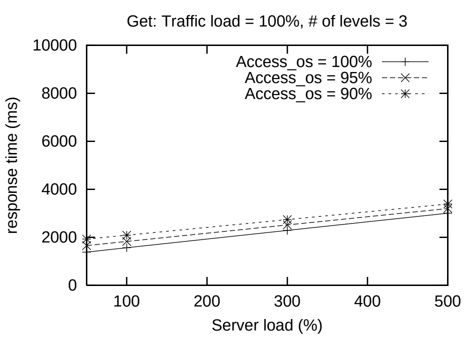


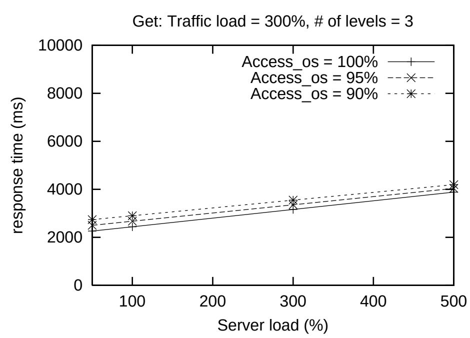


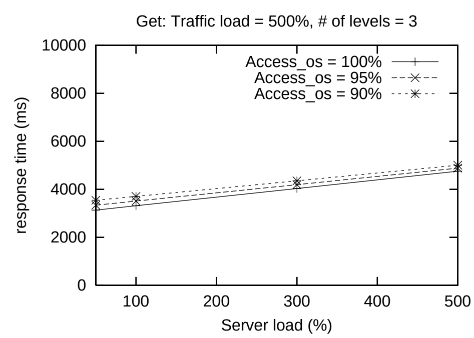


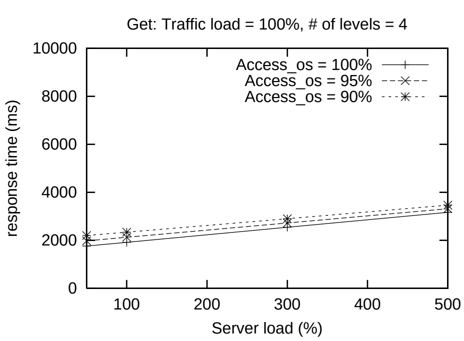


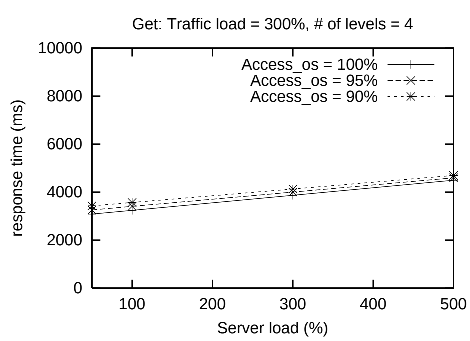


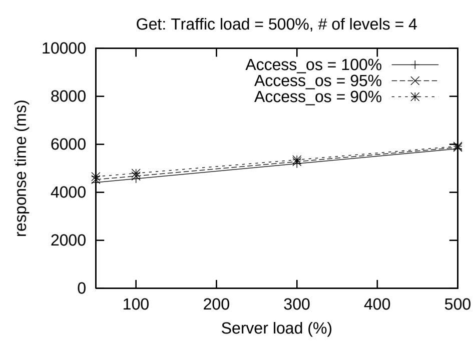




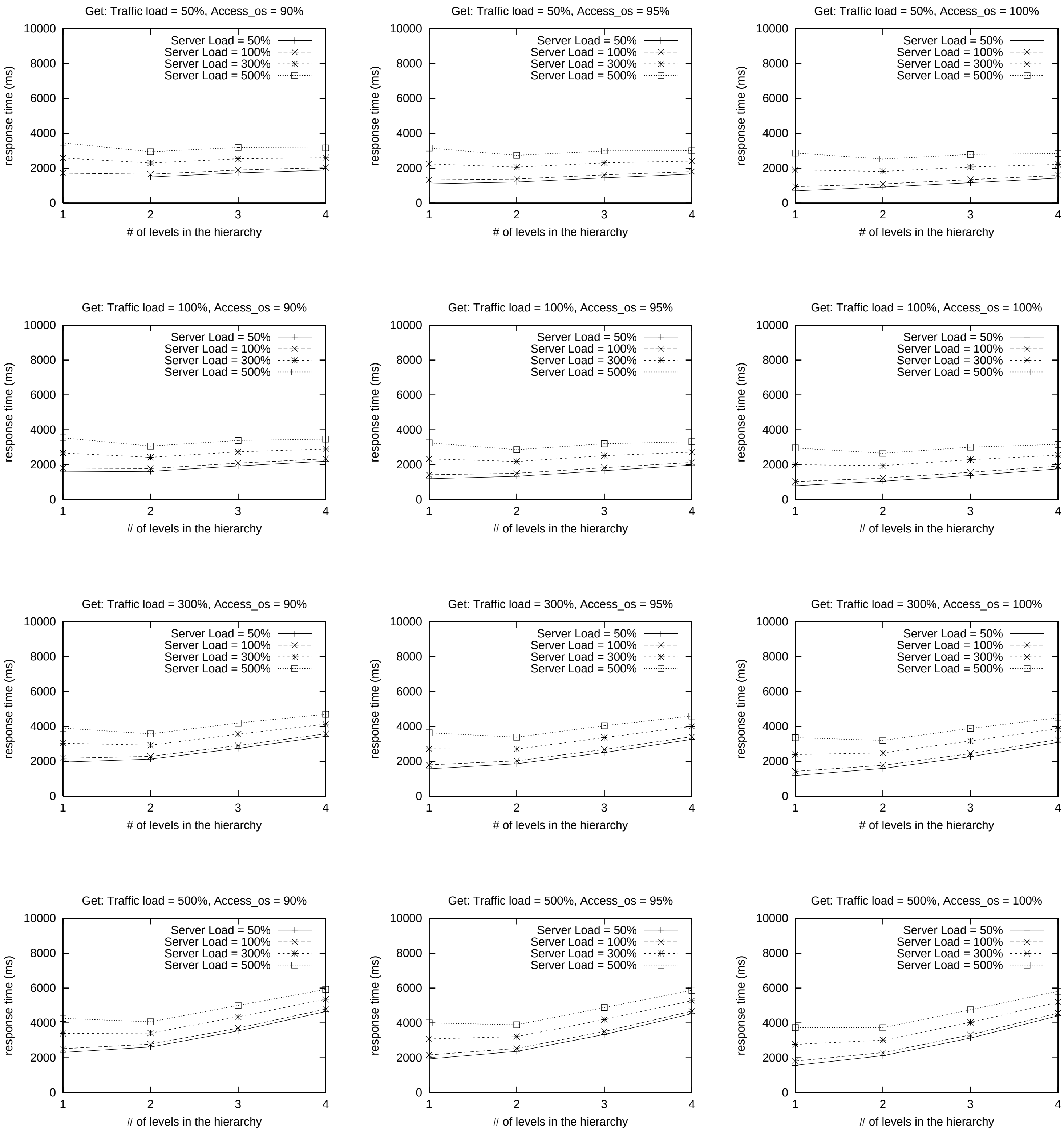




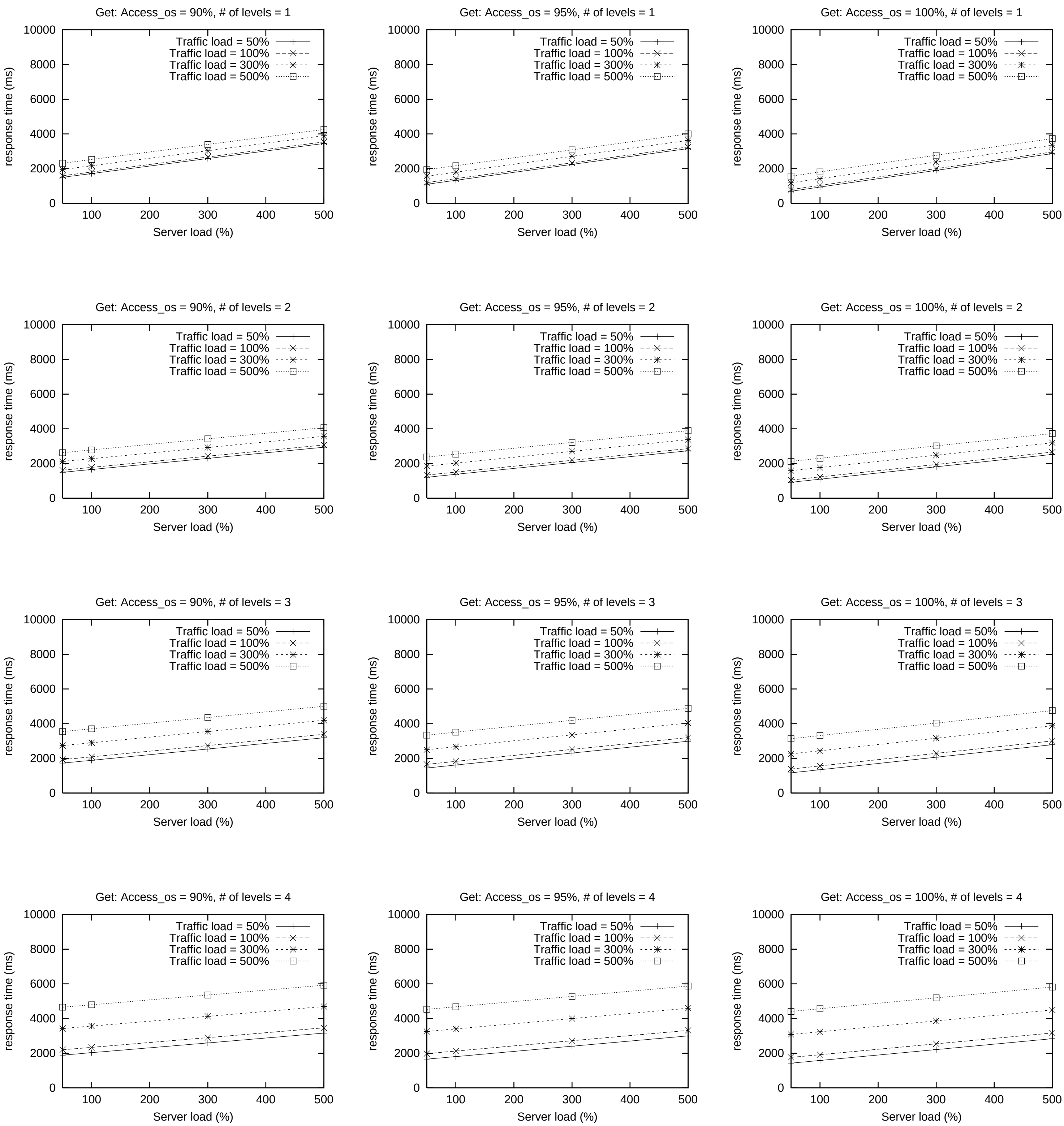




CD-2, svr-Level:

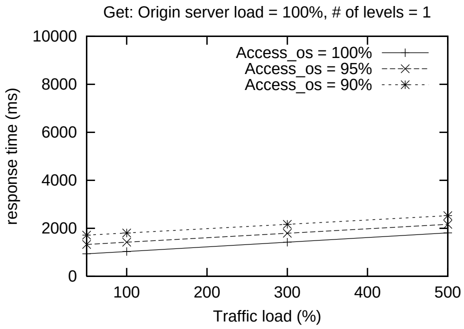


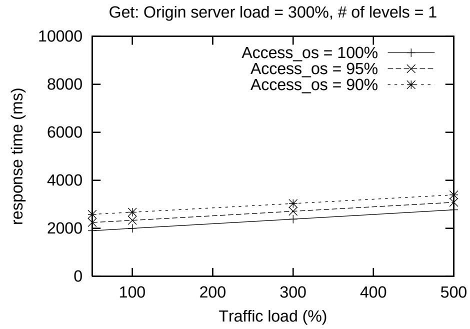
CD-2, Svr-tfc:

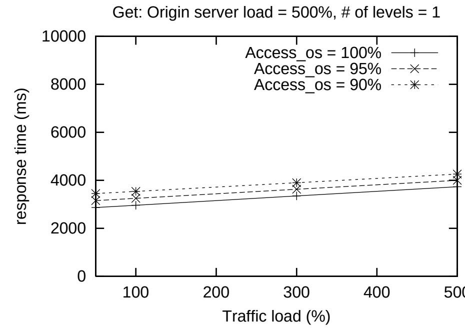


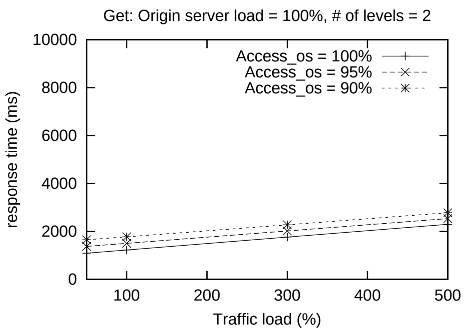
Get: Access_os = 90%, # of levels = 1
Get: Access_os = 95%, # of levels = 1
Get: Access_os = 100%, # of levels = 1
Get: Access_os = 90%, # of levels = 2
Get: Access_os = 95%, # of levels = 2
Get: Access_os = 100%, # of levels = 2
Get: Access_os = 90%, # of levels = 3
Get: Access_os = 95%, # of levels = 3
Get: Access_os = 100%, # of levels = 3
Get: Access_os = 90%, # of levels = 4
Get: Access_os = 95%, # of levels = 4
Get: Access_os = 100%, # of levels = 4
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

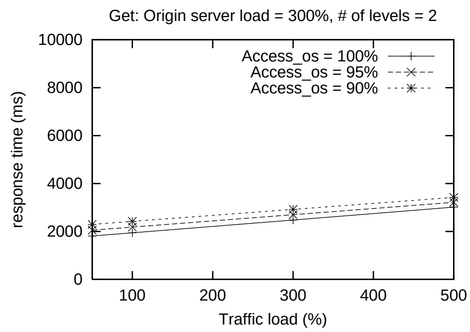
CD-2, Tfc-access:

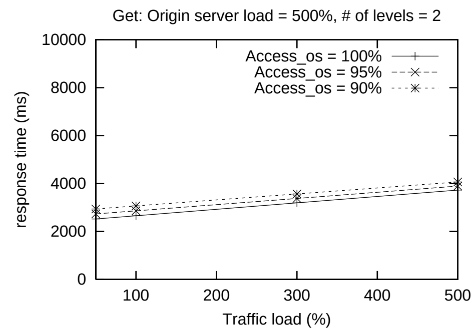


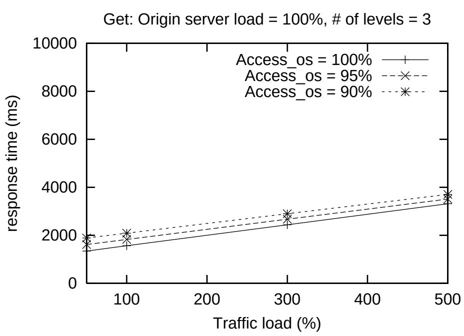


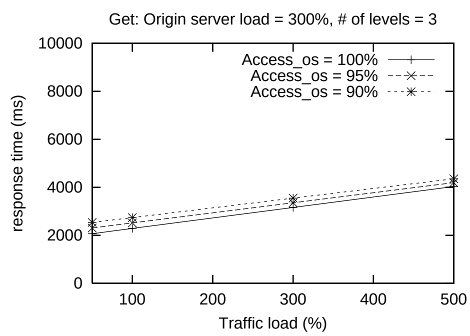


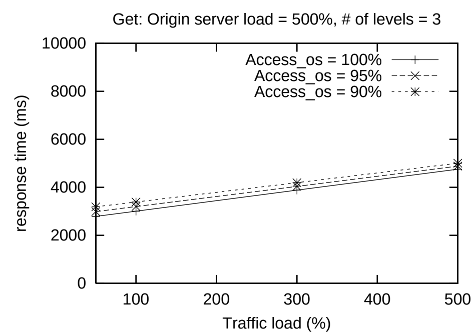


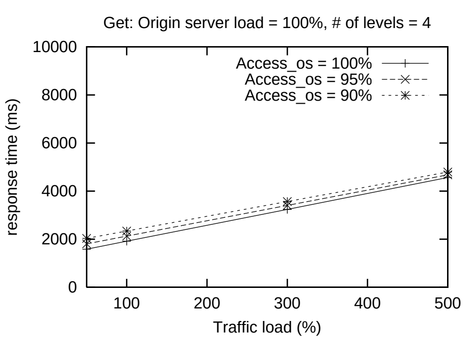


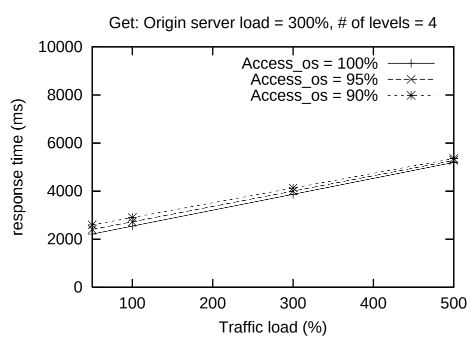


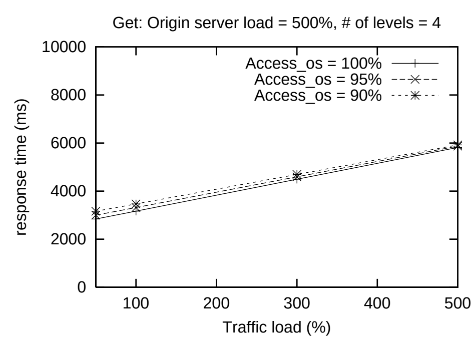




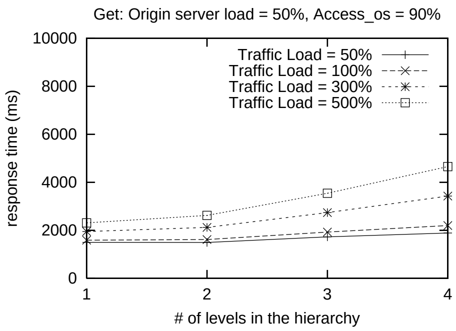


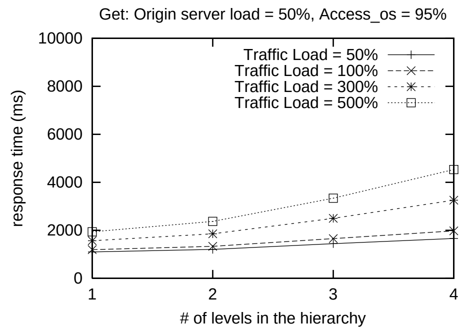


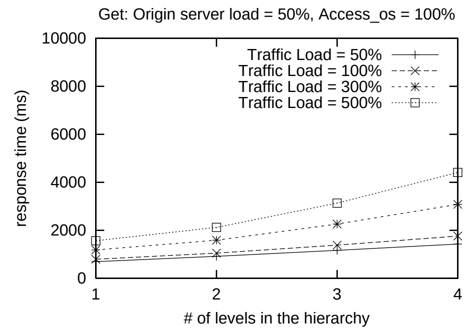


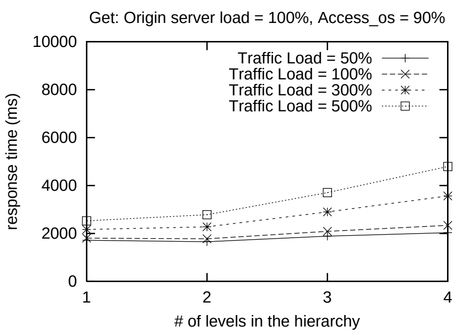


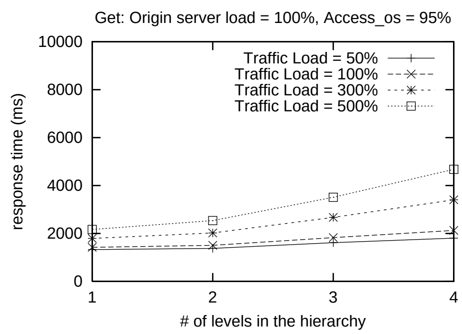
CD-2, tfc-Level:

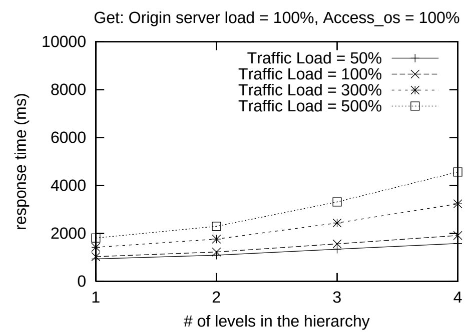


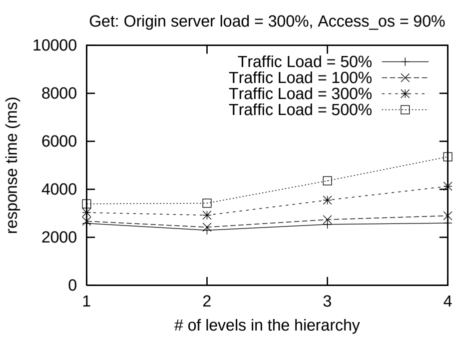


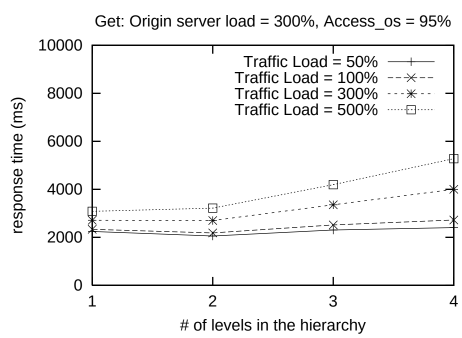


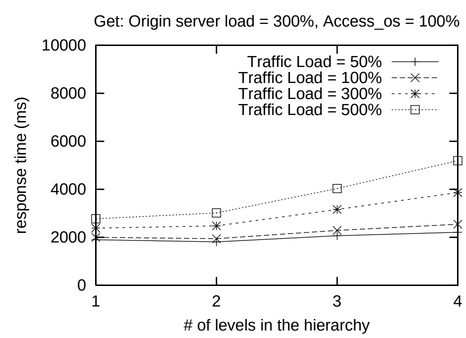


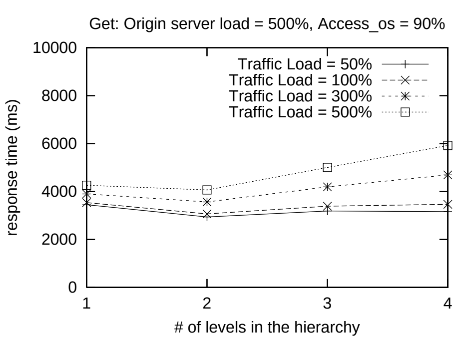


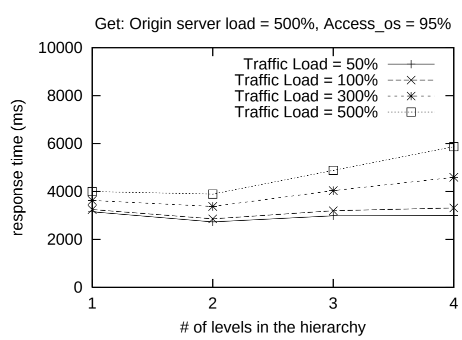


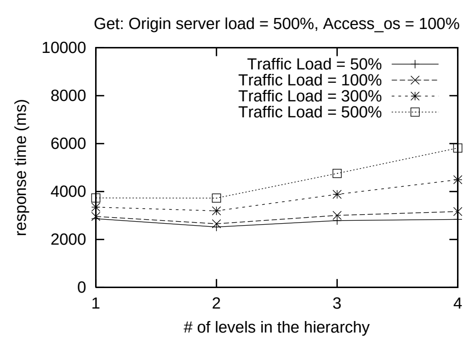




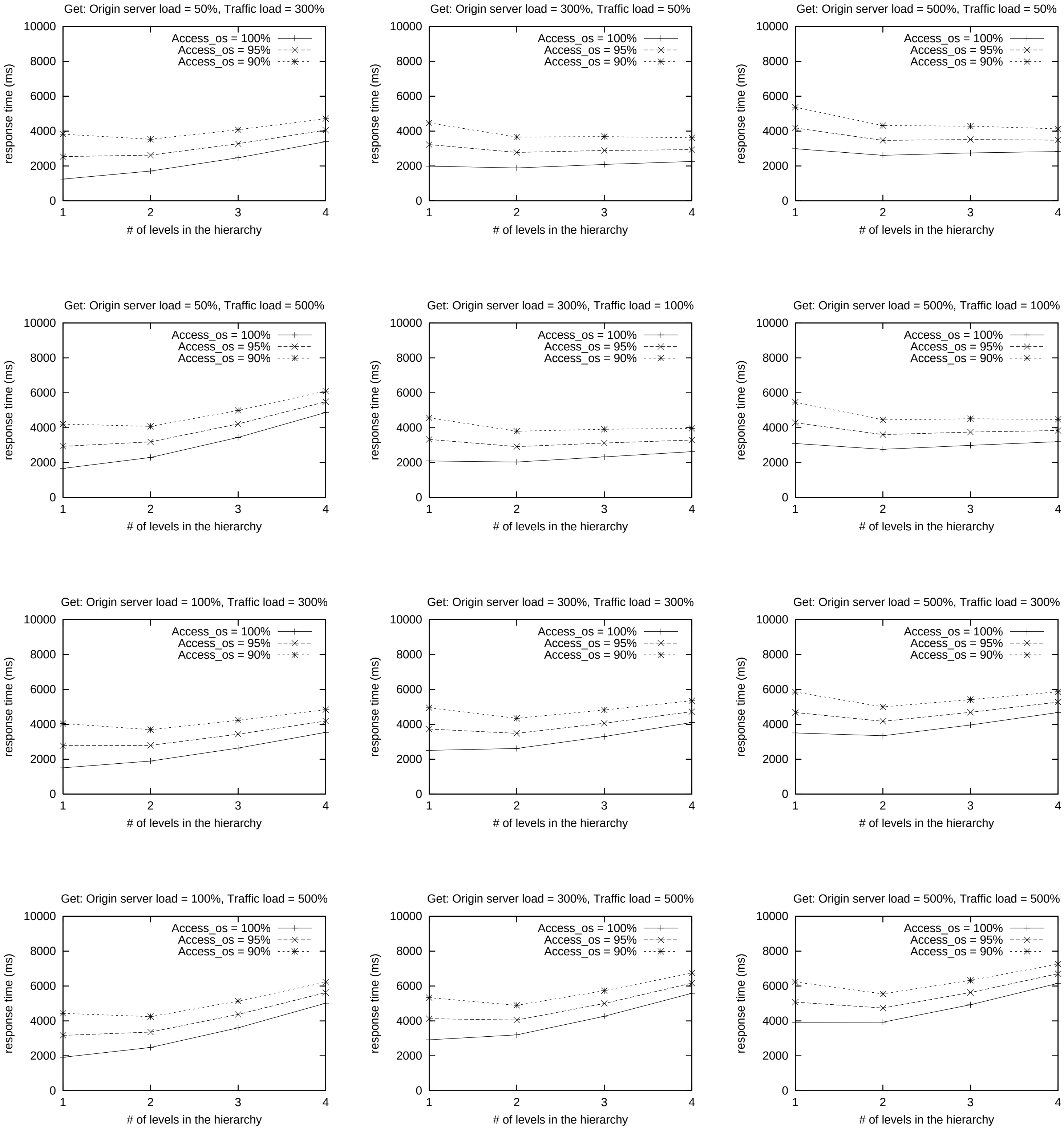




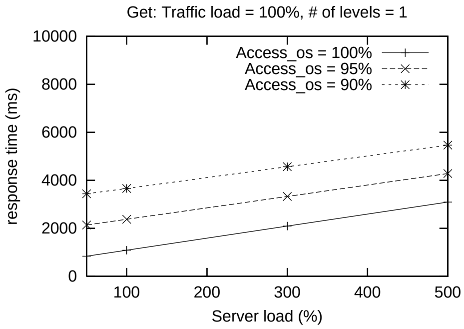


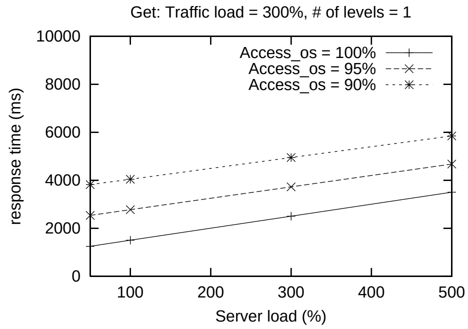


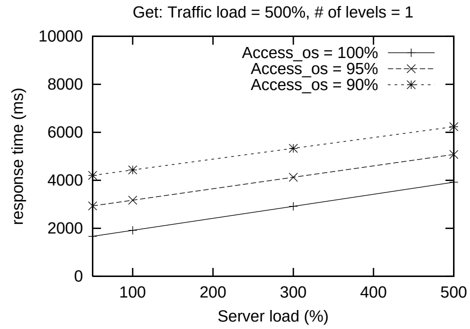
CD-3, access-Level:

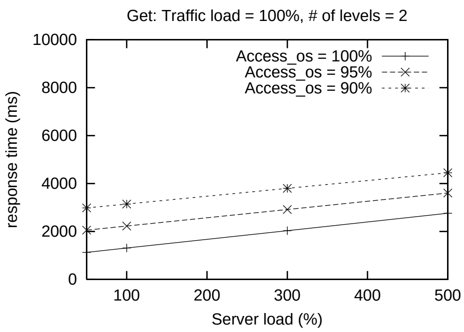


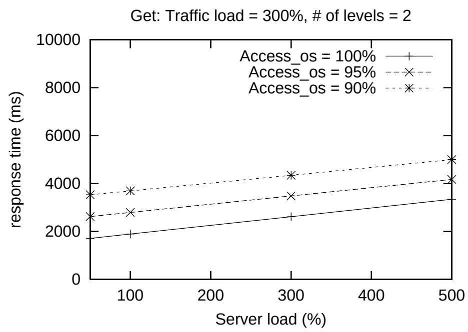
CD-3, Svr-access:

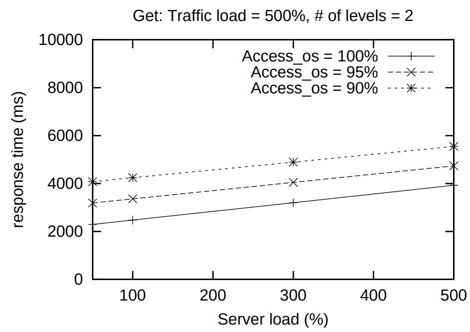


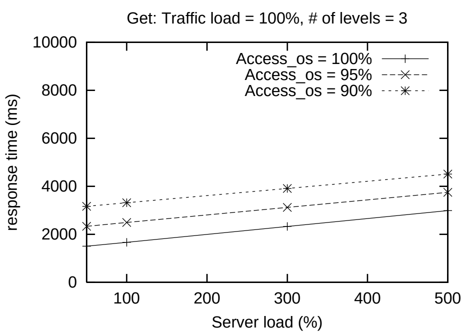


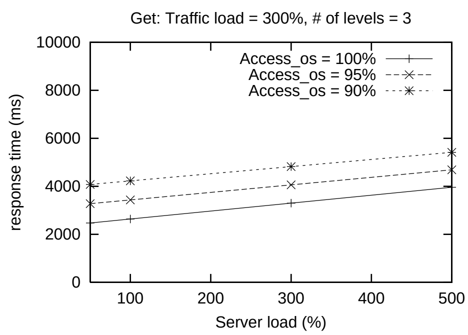


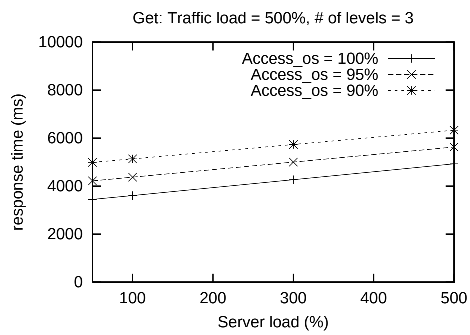


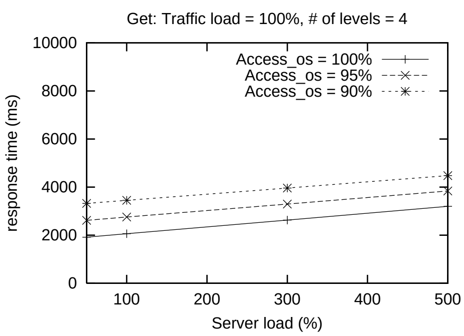


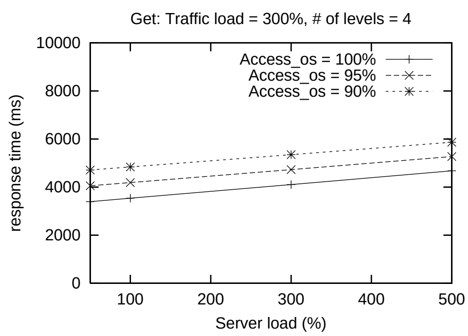


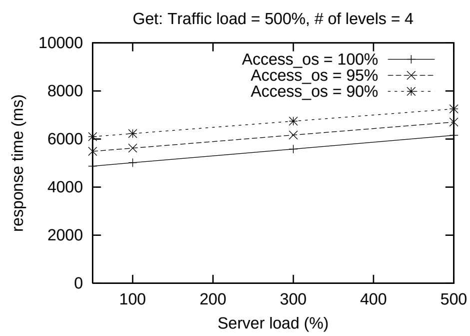




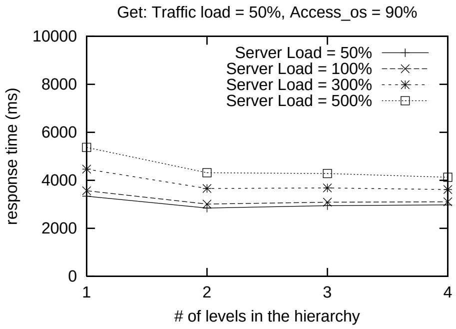


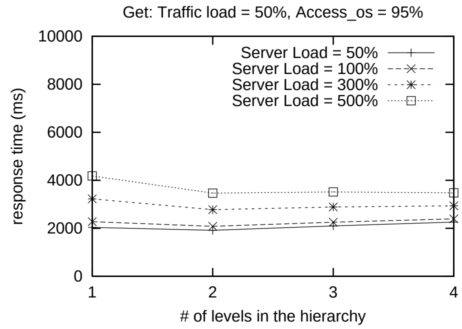


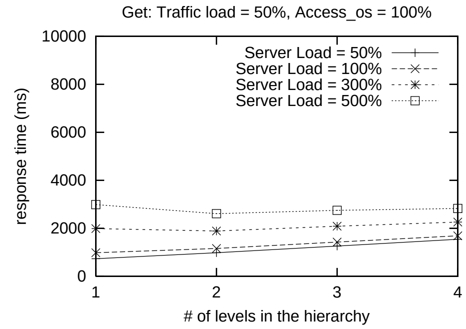


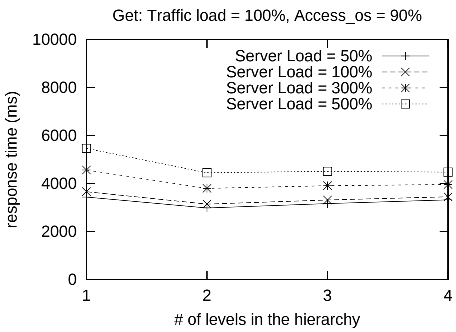


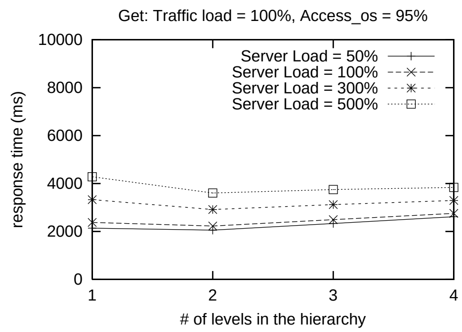
CD-3, svr-Level:

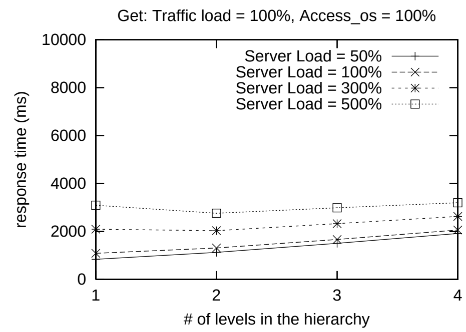


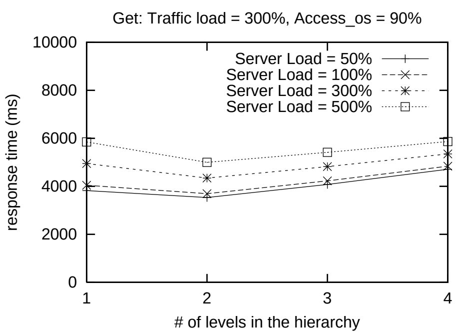


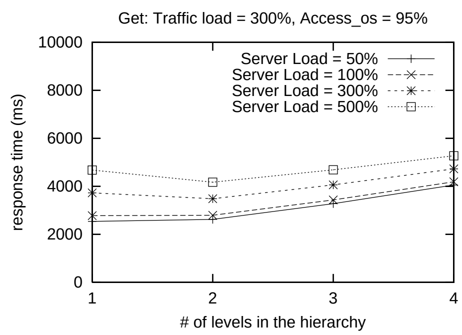


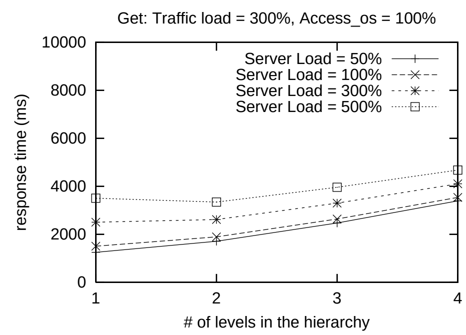


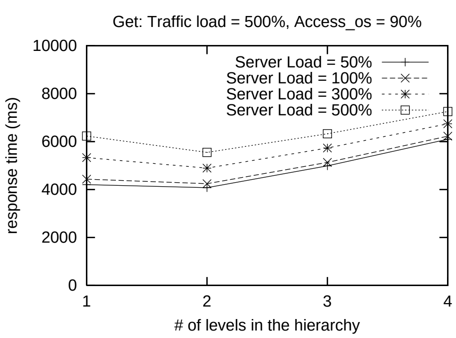


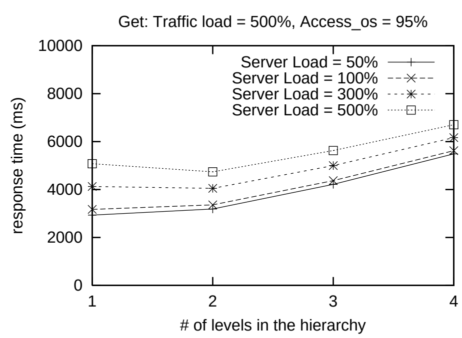


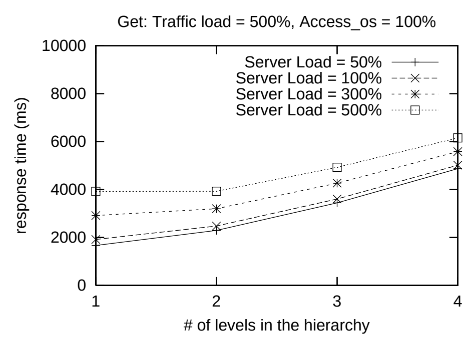




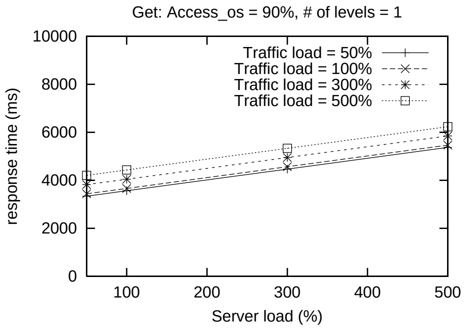




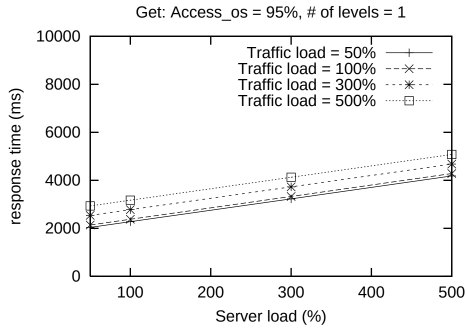




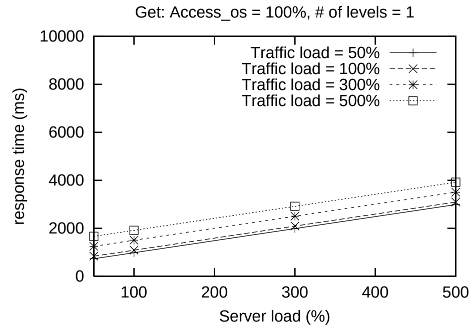
CD-3, Svr-tfc:



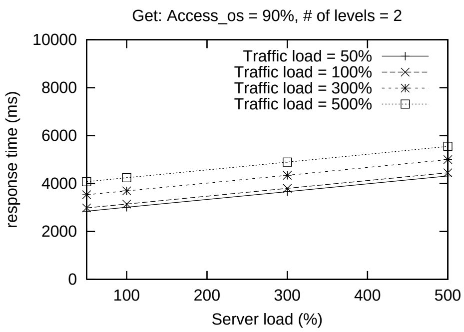
Get: Access_os = 90%, # of levels = 1
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)



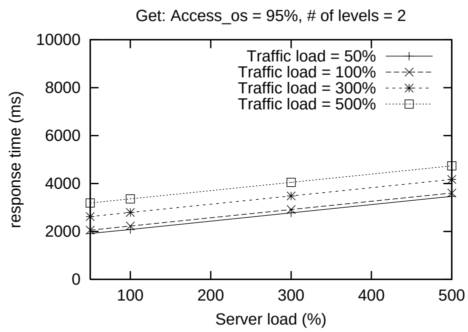
Get: Access_os = 95%, # of levels = 1
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)



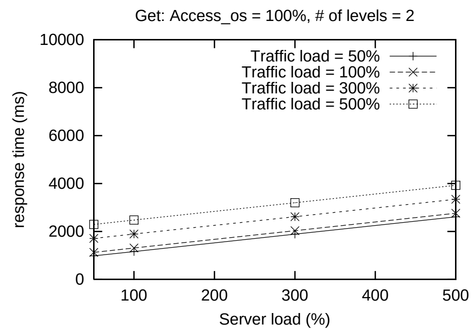
Get: Access_os = 100%, # of levels = 1
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)



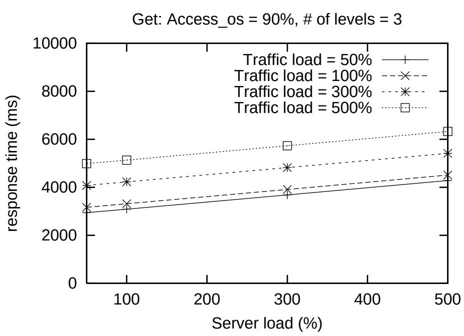
Get: Access_os = 90%, # of levels = 2
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)



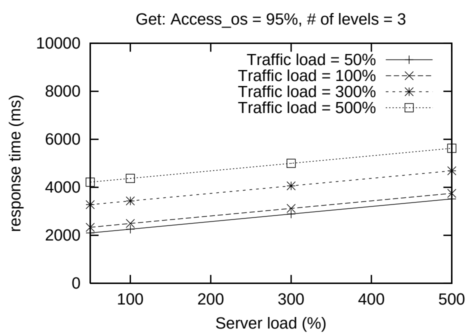
Get: Access_os = 95%, # of levels = 2
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)



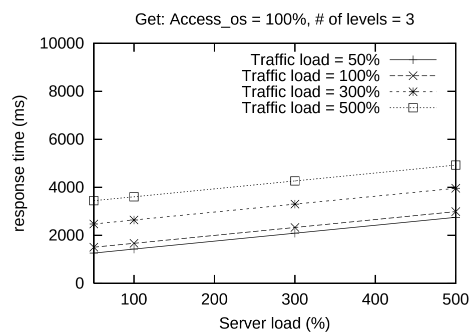
Get: Access_os = 100%, # of levels = 2
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)



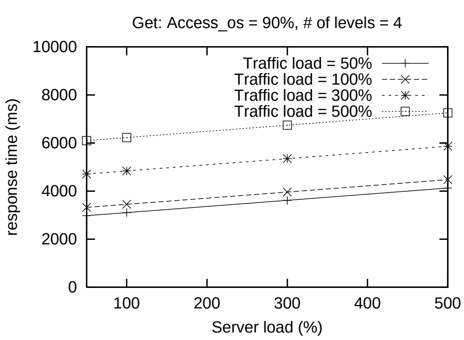
Get: Access_os = 90%, # of levels = 3
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)



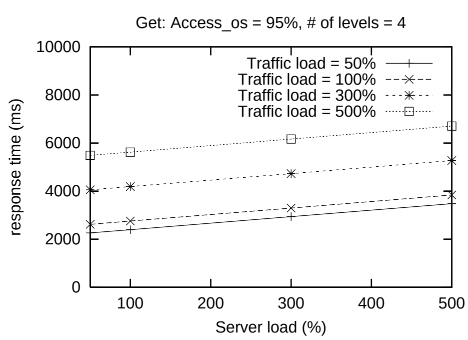
Get: Access_os = 95%, # of levels = 3
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)



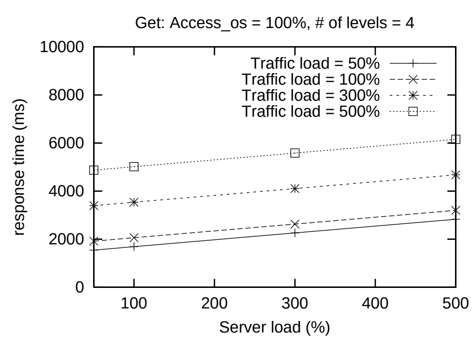
Get: Access_os = 100%, # of levels = 3
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)



Get: Access_os = 90%, # of levels = 4
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

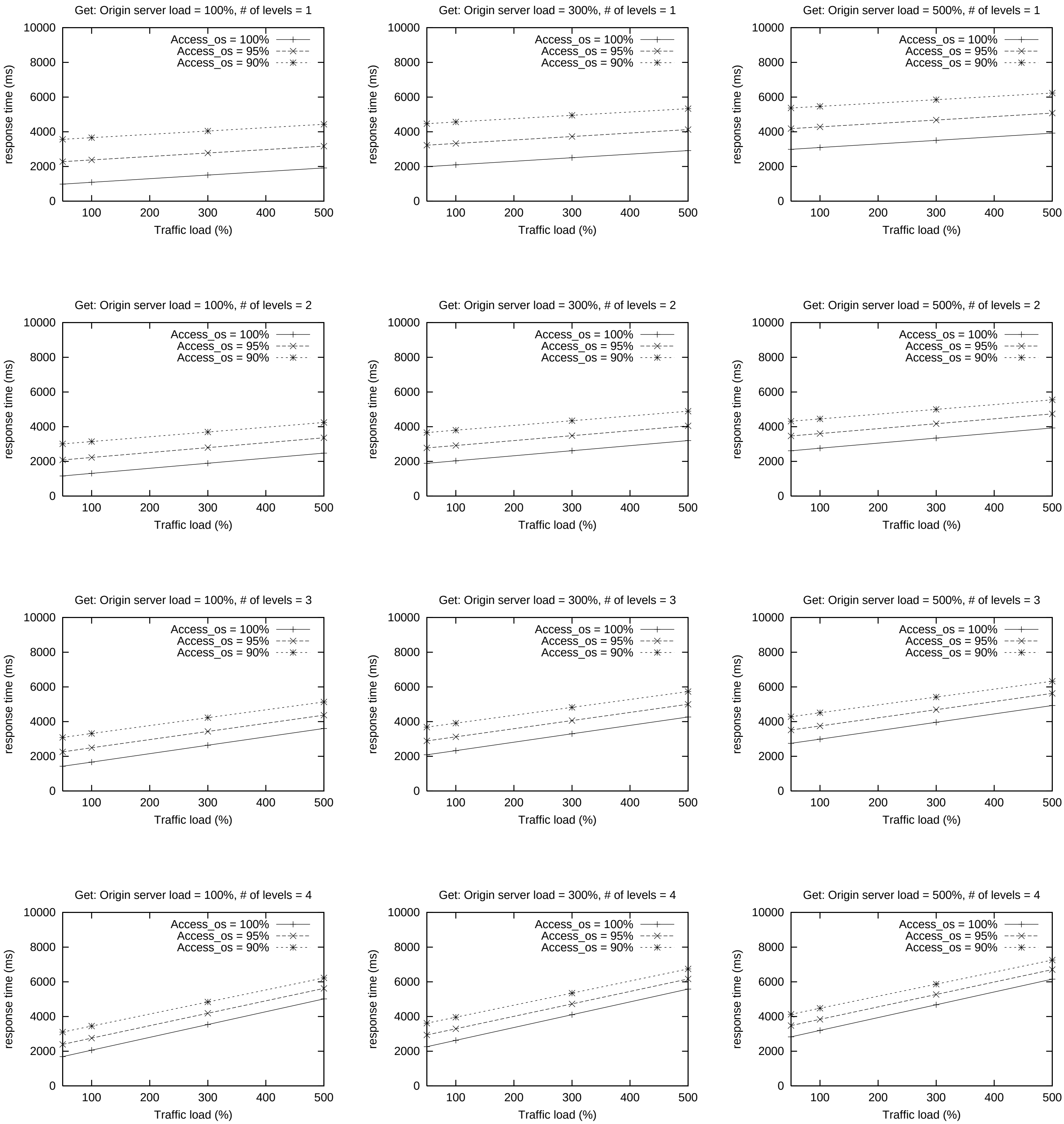


Get: Access_os = 95%, # of levels = 4
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)



Get: Access_os = 100%, # of levels = 4
Traffic load = 50%
Traffic load = 100%
Traffic load = 300%
Traffic load = 500%
response time (ms)
Server load (%)

CD-3, Tfc-access:



CD-3, tfc-Level: