THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

Hong Kong Polytechnic University

Department of Computing

A Novel Dynamic Cache Size Tuning Mechanism to Support Fast Response on the Internet with Applications to Mobile and Telemedicine

WU Sui Lun

PhD Thesis Adviser: Dr. WONG Kang Ying Allan

A Thesis Submitted in Partial Fulfillment

of the Requirements for

the Degree of Doctor of Philosophy

August 2005

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_ _____(Signed)

WU SUI LUN (Name of student)

ACKNOWLEDGEMENT

I sincerely thank my supervisor Dr. Allan Wong and co-supervisor Professor Tharam Dillon for their guidance, support and patience throughout my PhD study. Their advice is essential to the qualitative completion of this thesis. I would also like to thank those in the COMP Team who collaborated to provide useful information for my successful investigation into real-time Internet traffic pattern detection, which makes the proposed MACSC framework more accurate under all operation conditions. Last but not least I would like to thank my parents for their support throughout the whole research process.

ABSTRACT

The aim of this thesis is to propose a framework for maintaining a minimum cache hit ratio in a dynamic manner. By doing so the service roundtrip time in a client/server interaction over the Internet is shortened. This is achieved because the dynamic cache size tuner obviates the second leg in the information retrieval process. The final framework proposed in this research after considerable verification is the MACSC (*Model for Adaptive Cache Size Control*). This MACSC research has achieved all the objectives, namely:

a. Proposed the framework that leverages the minimum number of network parameters for accurate dynamic cache size tuning.
b. Proposed compensation measures that help the framework work healthily under extreme conditions.
c. Proposed a RTPD mechanism that is suitable for time-critical applications so that it can be included as part of the self-reconfigurable framework for dynamic cache size tuning on the fly.
d. Verified the generic framework with simulated and live datasets/traces.
e. Verified that the proposed framework can indeed support wired and wireless client/server interactions with similar efficacy.
f. Verified that this efficacy applies to mobile and time-critical applications such as telemedicine.

There are four approaches proposed for realizing the MACSC framework, which leverages the Zipf-like behavior as the conceptual basis. This behavior

represents the relative data object popularity profile from which the popularity ratio is derived on the fly. This ratio determines the size of the dynamic cache size adjustment. The four approaches dictate how the standard deviation and the popularity ratio of the popularity distribution should be computed, and they are as follows:

a. Point estimate (PE): This CLT (*Central Limit Theorem*) based method is sensitive to changes in the ROP profile, but it generates a lot of oscillations for having no feedback system.
b. M^3RT: This CLT based technique is transcribed from another problem domain, namely, Internet End-to-End Performance Measurement (IEPM) [Cottrel99], [Cottrel01]. Previous experience [Ip03] has confirmed that it always yields the mean of any waveform accurately in real-time applications because it has a feedback loop. The good quality of stability of this technique, however, becomes a liability for MACSC application because there is a need to strike a balance between stability and sensitivity, and this led to the proposal of the "fine-tune point estimate (F-PE) technique.
c. F-PE: It combines the merits of PE sensitivity and M^3RT stability and accuracy due to the presence of a feedback loop.
d. Real-time Traffic Pattern Detection (RTPD): It was observed in the early experiments that the Internet traffic patterns have different ill effects on the MACSC accuracy. Therefore, a RTPD mechanism is included in the original MACSC framework to make it into the newer RTPD/MACSC framework. The aim is to let the RTPD mechanism identify the traffic pattern at the time so that the MACSC can "*reconfigure*" itself to neutralize the traffic ill effects.

In the MACSC research context, the RTPD/MACSC(PE) solution is a form of reconfigurable dynamic cache size tuning. As a result four specialized solutions are derived from the general MACSC framework. The specialization is the technique whereby the SD is computed on the fly in each solution.

Therefore, the four novel proposed conceptual solutions, which represents an evolutionary process, are: MACSC(PE), MACSC(M^3RT), MACSC(F-PE), and RTPD/MACSC(PE).

LIST OF REFEREED PUBLICATIONS

The findings of this thesis have contributed to 17 refereed publications so far as follows:

7 Refereed Journal Papers

[1] Richard S. L. Wu, Allan K. Y. Wong and Tharam S. Dillon, $CACHE_{RP}$: A Novel Dynamic Cache Size Tuning Model Working with Relative Object Popularity for Fast Web Information Retrieval, *Journal of Supercomputing*, 2006 (Accepted and will appear in the journal).

[2] Richard S. L. Wu, Allan K. Y. Wong and Tharam S. Dillon, E-MACSC: A Novel Dynamic Cache Tuning Technique to Reduce Information Retrieval Roundtrip Time over the Internet, *Journal of Computer Communications* (Accepted and will appear in the journal).

[3] Wilfred W. K. Lin, Allan K. Y. Wong, Richard S. L. Wu, Applying Fuzzy Logic and Genetic Algorithms to Enhance the Efficacy of the PID Controller in Buffer Overflow Elimination for Better Channel Response Timeliness over the Internet, *Journal of Concurrency: Practice & Experience* (Accepted and will appear in the journal).

[4] Richard S. L. Wu, Allan K. Y. Wong and Tharam S. Dillon, RDCT: A Novel Reconfigurable Dynamic Cache Tuner to Shorten Information Retrieval Time over the Internet, *International Journal of Computer Systems Science & Engineering*, November 2004, 19(6), 363 – 371.

[5] Richard S. L. Wu, Allan K. Y. Wong and Tharam S. Dillon, $CACHE_{RP}$: A Novel Dynamic Cache Size Tuning Model working with Relative Object Popularity for Fast Web Information Retrieval, *Electronic Journal of Lecture Notes in Computer Science*, Springer-Verlag GmbH, 3358, 2004.

[6] Richard S. L. Wu, Allan Kang Ying Wong and Tharam S. Dillon, E-MACSC: A Novel Dynamic Cache Tuning Technique to Maintain the Prescribed Minimum Hit Ratio Consistently for Internet/WWW Applications, *World Scientific and Engineering Academy and Society (WSEAS) Transactions on Computers*, April 2004, 3(2), 424 – 429.

[7] Allan K. Y. Wong, May T. W. Ip, Richard S. L. Wu, A Novel Dynamic Cache Size Adjustment Approach for Better Data Retrieval Performance over the Internet, *Journal of Computer Communications*, September 2003, 26(14), 1709-1720.

2 Book Chapters (Invited)

[8] Richard S. L. Wu, Allan K. Y. Wong and Tharam S. Dillon, E-MACSC: A Novel Dynamic Cache Tuning Technique to Maintain the Hit Ratio Prescribed by the User in Internet Applications, *International Conference on E-Business and Telecommunication Networks (ICETE 2004) Best Paper Book*, Kluwer Academic Publishers, 2004 (***best papers series by ICETE'04)***).

[9] Allan K. Y. Wong, Richard S. L. Wu and Tharam S. Dillon, Dynamic Maintenance of a Given Proxy Cache Hit Ratio by Leveraging the Relative Data Object Popularity Profile to Yield Shorter Service Roundtrip Time, to appear in an edited collection by Nova Science Publishers, Inc., New York.

8 Refereed Conference Papers

[10] Richard S.L. Wu, Wilfred W.K. Lin, and Allan K.Y. Wong, Harnessing Wireless Traffic is an Effective Way to Improve Mobile Internet Performance (Invited Paper), Proceedings of the First IEEE International Conference on Wireless Broadband and Ultra Wideband Communications, Sydney, Australia, 459-464, March, 2006

[11] Wilfred W. K. Lin, Allan K. Y. Wong, Richard S. L. Wu, A Novel Real Time Self Similar Traffic Detector/Filter to Improve the Reliability of a TCP Based End to End Client/Server Interaction Path for Shorter Roundtrip Time, *Proceedings of 2nd International Conference on E-Business and Telecommunication Networks (ICETE 2005)*, Reading, U.K., October 2005, Volume 1, 94-102

[12] Richard S. L. Wu, Tharam S. Dillon and Allan K. Y. Wong, RTPD/MACSC: A Novel Approach for Effective Pervasive Information Retrieval, *Proceedings of the Fourth International Conference on Mobile Business (ICMB 2005)*, Sydney, Australia, July 2005., 514 – 520.

[13] Richard S. L. Wu, Allan K. Y. Wong, Tharam S. Dillon, Using Real-Time Traffic Pattern Detection for Dynamic Cache Size Tuning in Information Retrieval, *Proceedings of the Third Internal Conference on Information Technology and Applications (iCITA' 2005)*, Sydney, Australia, July 2005, Volume 2, 35 – 40.

[14] Richard S. L. Wu, Allan K. Y. Wong, Tharam S. Dillon, $CACHE_{RP}$: A Novel Dynamic Cache Size Tuning Model working with Relative Object Popularity for Fast Web Information Retrieval, *Proceedings of Second International Symposium on Parallel and Distributed Processing and Applications (ISPA 2004)*, Hong Kong, China, December 2004, 410 – 420.

[15] Wilfred W. K. Lin, Richard S. L. Wu, Tharam S. Dillon and Allan K. Y. Wong,

A Novel Real-Time Traffic Pattern Detector for Internet Applications, *Proceedings of the 2004 Australian Telecommunication Networks and Applications Conference (ATNAC 2004)*, December 2004, 224 – 227.

[16] Richard S. L. Wu, Allan K. Y. Wong and Tharam S. Dillon, E-MACSC: A Novel Dynamic Cache Tuning Technique to Maintain the Hit Ratio Prescribed by the User in Internet Applications, *Proceedings of 1st International Conference on E-Business and Telecommunication Networks (ICETE 2004)*, Set al/Portugal, August 2004, 1, 152 – 159.

[17] Richard S. L. Wu, May T. W. Ip and Allan K. Y. Wong, LDC-CM: A Novel Model for Dynamic Cache Size Adjustment, *Proceedings of 2003 International Conference on Internet Computing*, Las Vegas, Nevada, USA, June 2003, 2, 753-758.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER 1

BACKGROUND AND SCOPE

1.1 INTRODUCTION

The Internet and the World Wide Web (WWW) have provided several new opportunities for people to extend their horizon of knowledge, to communicate quickly and effectively, and to set up e-business. In fact, in this era Internet based distributed systems are a key factor in the achievement of economic gains for many companies. Firstly, it is common for company employees to interact with buyers and suppliers via the Internet in field operations. Secondly, business transactions such as buying a birthday present in Hong Kong and having it sent to the receiver in Canada can be conducted quickly and accurately in an electronic manner (i.e. e-business). Thirdly, up-to-date background information can be obtained immediately, via an electronic small-form-factor (SFF) device (e.g. PDA and PMDA) [Patterson03] to back up a time-critical decision such as trading stocks right at the airport minutes before boarding a plane. Fourthly, opportunistic data acquisitions can be conducted in a pervasive manner to prevent disasters proactively [Hightower01]. For example, a pedestrian encounters a terrorist act in progress and videos the scene with her mobile phone (a SFF device). Then, the short video is immediately transmitted to the police so that the latter can assess the situation quickly to save the hostage. The examples above involve nomadic users in a mobile/pervasive computing environment, in which the client (a SFF device of a nomadic user) communicates

through a wireless cell with the wired part of the Internet based PCI (pervasive computing infrastructure).

In the above situations the service response time, which is the interval between sending a request and getting the correct result, is important. The sheer size and heterogeneity of the Internet make it difficult to control and guarantee the prescribed response deadline unless special techniques are employed. This difficulty is aggravated by the fact that the Internet nowadays is basically a “wired and wireless” setup, which involves different incompatible protocols that require the technique of tunneling to link them [Tanenbaum96]. Information retrieval over the Internet is a client/server relationship [Mahanti00], with the "first leg" between the client or requestor and the proxy server, and the "second leg" between the proxy server and the data source (remote web server). It is generally recognized that caching is a technique that can be employed to reduce the service roundtrip time because if the object can be found in the proxy’s local cache then the “second leg” delay can be obviated (to be explained later in this section). The fringe benefit is that finding the data in the cache means less data need to be transferred remotely across the network, and in this sense it frees backbone bandwidth for public sharing for higher network throughput [Aggarwal99]. Therefore, the overall average service roundtrip time (RTT) is made up of the average RTT_1 for the first leg and average RTT_2 for the second; that is $RTT=RTT_1+RTT_2$. As shown in Figure 1.1, if the requested data can be found in the proxy server’s cache then $RTT_2=0$ holds. Locating the requested data object in a remote web server of data source involves the Internet Domain Name Server (DNS) and data transfer over the open Internet, and this usually makes RTT_2 much larger than RTT_1.

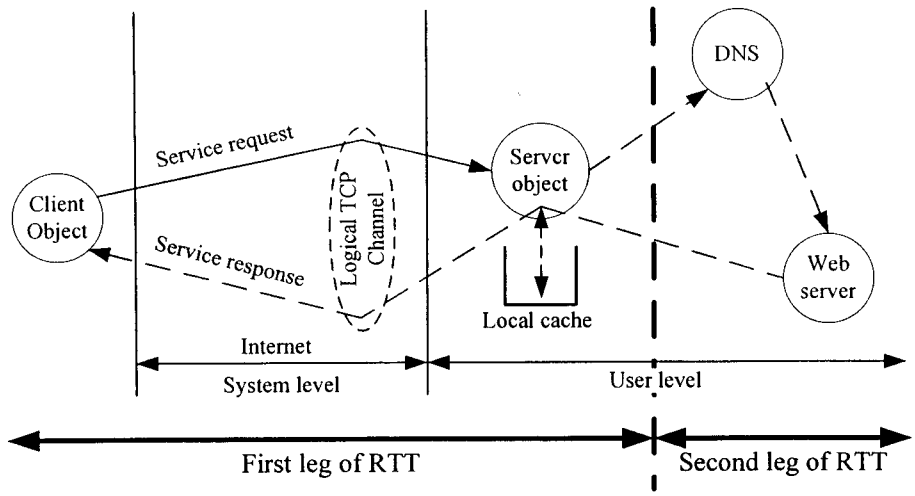


Figure 1.1 The relationship between the client and the proxy server

The hit ratio is the probability of finding the required data locally in the proxy's cache; conversely, the miss ratio is one minus the hit ratio. The hit ratio fluctuates with the clients' shift of preference for certain data items. It is advantageous to have a high proxy cache hit ratio because RTT_2, which involves the Domain Name Server (DNS), is usually very much longer than RTT_1. The DNS helps the proxy locate the required data objects in the right remote data source. The implicit advantage from caching is that less data needs to be transferred across the network. This means more backbone bandwidth available for sharing and a reduction in the chance of network congestion.

Caching is important to different network situations: wired network, wireless network, and "wired and wireless (W&W)" network. In the wired network caching can be used to solve problems of network congestion. In the relatively unreliable wireless network caching can help reduce the number of retransmissions. The important impact of caching will be discussed in chapter 3. In fact, the explicit and implicit advantages from caching have motivated different areas of relevant research. The most popular topic is how to design replacement strategies to effectively keep as many hot data objects as possible in the cache. Almost all the known replacement

strategies work with a static cache size. They aim at yielding a high cache hit ratio but do not necessarily maintain it. For this reason the cache hit ratio fluctuates with respect to the system dynamics and the current data object popularity profile. Maintaining a given cache hit ratio needs dynamic cache size tuning. In this thesis the novel MACSC (*Model for Adaptive Cache Size Control*) framework, which leverages the relative data object popularity profile as the sole parameter for this purpose, is proposed. It represents an important departure from previous work which always postulates static cache size. This new approach leads to significant improvements in cache hit ratios or allows one to maintain a prescribed hit ratio. It computes the tuning solution in a short time to avoid possible deleterious effects by the tuning process.

One may ask why we need dynamic cache size tuning such as the MACSC framework if a very large cache could be used to yield a high hit ratio. The answer is as follows:

1. A very large cache means a higher chance of stale data and thus data incoherence in the hitting process because the data in the cache is not refreshed and updated frequently enough [Breslau99].
2. The MACSC caters especially to small caching systems, which usually cost less than USD1000 [Wessels01]. In these systems memory resources are limited. If too much memory is used for caching, other tasks may be suspended due to lack of recyclable memory, leading to poor system throughput.

1.2 MOTIVATION AND SCOPE OF PROBLEM

The motivation of the thesis is to explore how the cache hit ratio can be maintained under all conditions in a dynamic manner, independent of the changes in the Internet traffic pattern. The Internet follows the power law and its traffic changes anytime, for example from being LRD (long-range dependence) to SRD (short-range dependence) or vice versa. In order to maintain a given hit ratio on the fly the following questions/issues need to be effectively addressed:

1. What are the characteristics of a dynamic cache size tuner that can maintain a given hit ratio under all conditions?
2. What is the underlying principle for a successful tuner as such for real-time or time-critical applications over the Internet?
3. Can such a tuner survive in the dynamic Internet environment in which the traffic pattern changes all the time? It is generally understood that any system (a tuner is no exception) that is designed with a preconceived mathematical model in mind fails easily in the Internet [Paxson95]. This implies that by nature the dynamic cache size tuner(s) should be statistical and work by direct data measurement.
4. How can a specific traffic pattern be detected correctly on the fly?
5. How can the accuracy of the real-time traffic detection (RTPD) mechanism be made independent of traffic patterns?
6. How can the execution time of the RTPD mechanism be shortened enough to avoid any deleterious effects? These effects are undesirable consequences of a long detection time because by the time the computed remedy is available the real problem has already passed, and the remedy ends up correcting a problem

that no longer exists, which itself could be detrimental.

7. Can the proposed dynamic cache size tuning framework operate together with the RTPD mechanism in the sense that it would reconfigure to neutralize the ill effects caused by traffic on its hit ratio while still maintaining efficacy?
8. Can the final “RTPD plus tuner” combination indeed support efficient mobile and time-critical applications such as telemedicine by shortening the service roundtrip time (RTT) through consistent hit ratio maintenance? This should be verified by simulation with live RTT traces collected from different web sites.

The research activities that strive to provide satisfactory solutions to the above issues form the scope of my PhD problem. In this scope the following will be investigated in an orderly manner with the help of an appropriate methodology which, since the problem of dynamic cache size tuning is relatively pristine, must be newly defined. The existing research methodologies may not suffice and new elements that pertain to this line of work have to be identified. The order of the main work items are as follows:

1. Define the principle for dynamic cache size tuning over the Internet, for example, by exploring whether the Zipf-like behavior can provide the necessary basis [Breslau99], [Zipf].
2. Define the compensation measures, if necessary, to neutralize the behavioral deviations by the tuner(s) under extreme operational conditions.
3. Identify different useful statistical methods that help the proposed tuner compute quickly and accurately on the fly. The most important of all these methods should be independent of traffic patterns that embed in a stretch of

inter-arrival times (IAT) among the service requests from the clients to the server.

4. Explore how real-time traffic pattern detection (RTPD) can be achieved.
5. Explore how RTPD can be included in the proposed tuner framework so that the latter can use the detected results to reconfigure itself and ward off any ill effects on its tuning accuracy caused by traffic.
6. Explore the client/server interaction requirements for mobile and time-critical applications using TCM (Traditional Chinese Medicine) based telemedicine as an example. This example is chosen because of my previous related research experience.
7. Verify, at least by simulation, that the final reconfigurable dynamic cache size tuning mechanism(s) can indeed support mobile and time-critical applications.

1.3 THE DRIVING FORCES

The driving forces behind the motivation of this PhD research are as follows:

1. The desire to maintain the given cache hit ratio for information retrieval: There is no such technique in my literature search so far. The importance of this technique is that it shortens the service roundtrip time (RTT) between the client and the server. This makes the client happy and is therefore fundamental to the success of any electronic transactions over the Internet.
2. The advantage of having a shorter service RTT in telemedicine: This saves more lives in contingent cases [Lacroix99], as I had observed in my previous research experience. Some of the telemedicine issues will be

discussed in Chapter 11.

3. The desire to propose a friendly dynamic cache size tuning framework: The aim is to have a “promiscuous” framework, which can be combined with other extant caching techniques for synergistic effects in a friendly manner. The combined framework would consistently yield and maintain a high hit ratio.
4. The desire to leverage the intrinsic Zipf-like behaviour as the sole metric: This behaviour is a formal representation of the relative popularity among different objects in a set. Leveraging this behaviour provides a sound technical basis for dynamic cache size control.

To appreciate the importance of the diving forces above it is useful to understand the relevant issues in Internet caching [Podlipnig03, Wang99, Wu00]. Caching can alleviate network congestion and quickens WWW information retrieval. Its usefulness grows with the size of the WWW page. The danger of congestion is aggravated by the fact that the WWW page size has a monthly growth rate of around 15% but the Internet backbone capacity only increases by 60% yearly [Bharat98]. The massive quantity of information requiring transfer across the network can quickly deplete the amount of sharable bandwidth. Since caching reduces the amount of data to be transmitted across the network, it frees backbone bandwidth for public sharing and thus reduces the chance of network congestion.

The most important performance indicator of a caching system is its ability to yield a high cache hit ratio, even though Internet caching has been used for different purposes such as follows [Podlipnig03]:

1. To reduce network bandwidth usage – Use caching as a means to free

backbone bandwidth for public sharing, and this leads to better system throughput due to less chance of network congestion.

2. To reduce response delays (the focus of this thesis) – Facilitate fast information retrieval to enhance the success of real-time applications over the Internet; for example, a shorter service RTT betters the chance of saving a patient in a emergent case.
3. To reduce loading on the original web server – Split and shift part of the loading of a web server up front to the proxies.

Yet, yielding a high hit ratio (but not necessary maintaining) is the primary aim of all the extant caching strategies. At the moment these strategies, however, cannot produce a high hit ratio and maintain it at the same time because they exclusively use a static cache size [Wu03]. Maintaining a given hit ratio has a very different requirement from yielding a high one because the former need to eliminate the negative impacts on the fly. In fact, one simple way to yield a high hit ratio is to use a very large static cache size conceptually [Breslau99], but this approach can lead to serious data coherence problems.

Although there is no extant caching strategy that can maintain a given hit ratio, from the literature, however, there are some rudimentary adaptive caching (adaptivity) systems. Theoretically these systems could produce a high hit ratio in a relatively continuous manner even though they use a static cache size. They are mainly in the area of adaptive replacement [Podlipnig03], [Wang99] and can be divided into the following categories:

a) The first category invokes different placement strategies that work with a static cache size under different operation conditions. The philosophy is to

use the best strategy to harness the current situation. This category has two problems: a) it is difficult to decide when to invoke a new strategy, and b) it is hard to define which strategy is the best for the time.

b) The second category tunes the parameters of the replacement strategy adaptively to yield a high hit ratio. The category, however, involves many parameters and heavy parameterization means substantial computation delay. By the time the caching remedy is computed the actual problem may have passed. The solution would end up correcting a spurious condition, causing deleterious/undesirable effects.

To fill the void the MACSC (Model for Adaptive Cache Size Control) framework for dynamic cache size tuning is proposed in this PhD thesis. This framework differs from the all the other extant caching systems in two aspects: a) it uses a variable cache size, and b) it maintains the given hit ratio on the fly. Its formal basis is the Zipf-like behaviour [Breslau99]. The MACSC can be combined with other caching strategies, which can neither yield a high cache hit ratio consistently nor maintain it when working alone. The factors that usually contribute to the hit ratio variations include the following [Abrams95], [Yu99]:

a. Seasonal user preference for specific data items – The preference in a rage creates seasonal relative object popularity. Hot data items can have much higher access frequencies than the cold ones.

b. The replacement strategy adopted – There are two basic types of strategies, LRU (least recently used) and LFU (least frequently used). These strategies leverage different parameters and produce very different hit ratios for different conditions [Aggarwal99].

c. The number of parameters being leveraged – Different replacement strategies leverage different number of metrics (e.g. object size, access/reference frequency, and update frequency, object popularity [Jin00a]). Leveraging a metrics involves two delay elements, sampling and computation. Therefore, leveraging many parameters means heavy parameterization that could lead to long computation delay. Besides, the parameters may counteract among themselves in an unpredictable manner.

d. The placement/location of the proxy cache – The cache location in the network can affect the hit ratio. Yet, there is little published experience for this problem.

e. The efficacy of the admission control – Admission control determines which data objects to be cached so that the "one-timers" can be eliminated. It makes the hot data in the cache more concentrate to improve the hit ratio.

The above factors reflect the fact that a successful caching scheme is not easy to build because a success for one operating environment may not apply to another [Wang99]. Deeper analyses in my preliminary PhD investigations indicate that the cache hit ratio is usually affected by different metrics such as follows:

a. Cache size – Although a larger cache size would yield a high hit ratio, risk of data incoherence or inconsistency also increases because stale data is not replaced fast enough [Breslau99]. Therefore, attaining a meaningful high hit ratio is a complex problem. The solution involves a reasonable cache size [Aggarwal99] to be controlled by a powerful replacement strategy, which should also be supported by a sophisticated strategy that ensures data coherence [Shim99].

b. Object popularity – A fixed length cache size yields different hit ratios at different times because of the continuous change in the relative object popularity distribution. An efficacious cache replacement policy should be able to predict the relative popularity of a document and decide proactively whether it is worthwhile to cache it [Jin00a], [Dilley99].

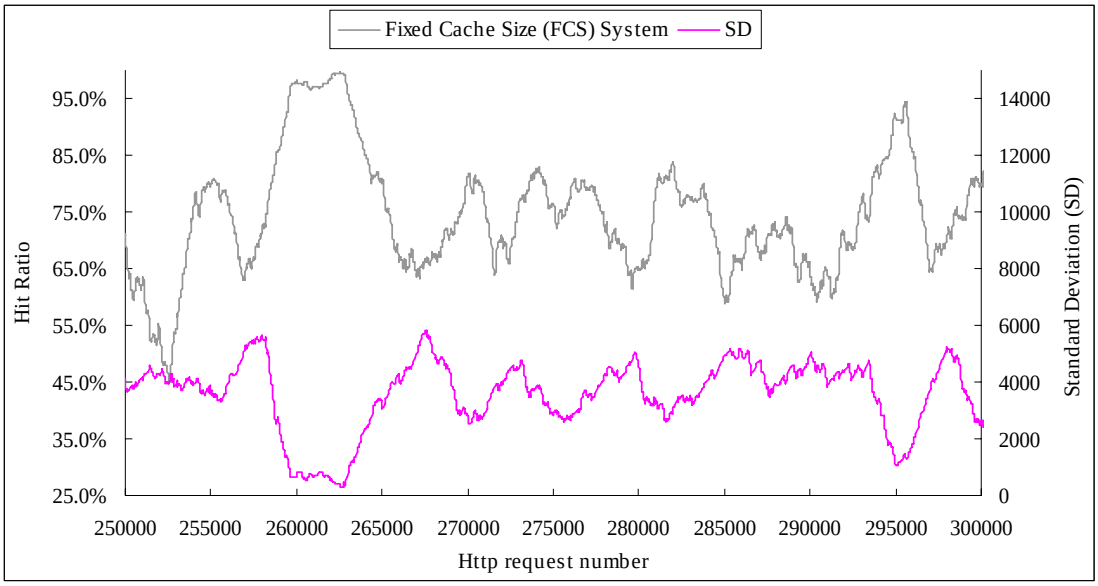


Figure 1.2 Impact of user behavriour on the hit ratio of a wireline fixed cache size system

My preliminary PhD investigations had also revealed a clear correlation between relative data object popularity distribution and cache hit ratio. For example, Figure 1.2 is the investigation for a fixed cache size (FCS) system. It shows how the changes in the standard deviation of the relative data object popularity distribution can affect the cache hit ratio in a wireline environment. Similarly Figure 1.3 is another FCS based result for the wireless environment.

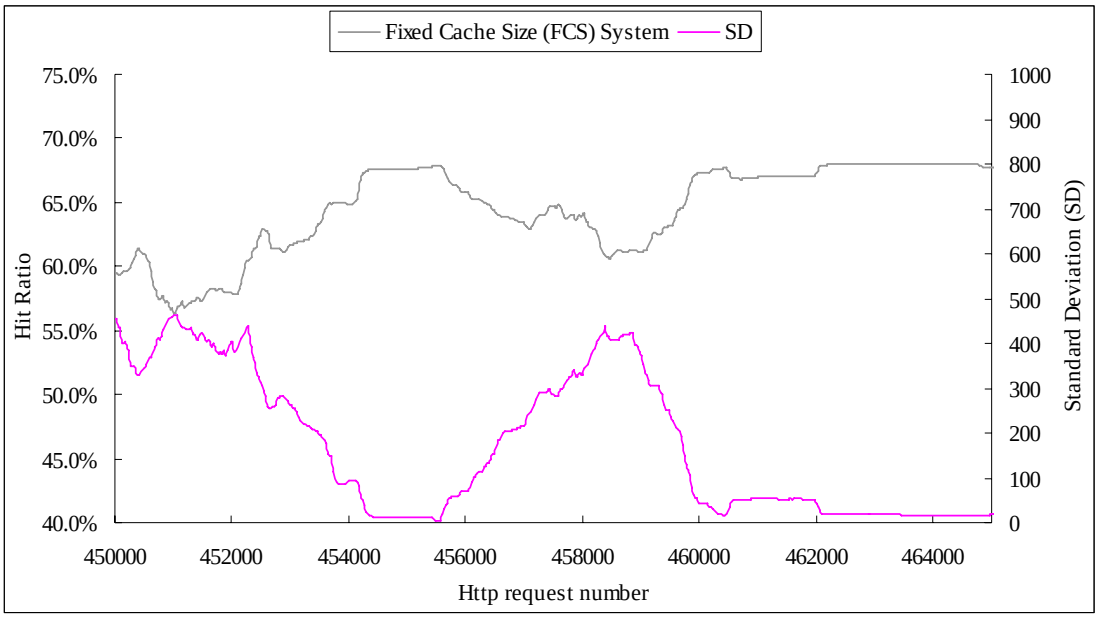


Figure 1.3 Impact of user behavriour on the hit ratio of a wireless fixed cache size system

1.4 POTENTIAL CONTRIBUTIONS TO MOBILE AND TELEMEDICINE APPLICATIONS

Dynamic cache size tuning, as exemplified by the proposed MACSC (*Model for Adaptive Cache Size Control)* conceptual framework, enhances the quality of service (QoS) for mobile and telemedicine applications. In perspective, it reduces the chance of involving the second leg in information retrieval over the mobile Internet as shown in Figure 1.1. As a result it shortens the service roundtrip time (RTT) (also known as response time) to provide the following advantages:

1. Happy customers (clients): Happy customers are return customers that make the e-business a success. Since the Internet today is mobile or pervasive in nature (i.e. “wireless + wireline”) [Garlan02], the intrinsic complexity in the network structure easily lengthens the response time.
2. Conducive to successful time-critical applications: Some applications over the mobile Internet are time-critical or real-time as exemplified by the

telemedicine field. One of the inspirations of this research is my previous experience in telemedicine (Chapter 11 provides more details). From my observation, it is absolutely important for the remote paramedic to get the diagnostic and treatment advice from the "virtual doctor" quickly to save the patients in contingent cases. The MACSC approach can shorten the response time in between and therefore enhances the chance of saving remote patients.

3. Less Internet congestion: If the information to be retrieval is found more often in the proxy cache, then there is less data to be transferred across the network in the second leg of information retrieval (Figure 1.1). This frees the backbone bandwidth and virtually increases the Internet capacity [Bharat98].

Since the Internet nowadays is mobile in nature (i.e. mobile Internet with a "wireless + wireline" infrastructure), the MACSC potentially contributes to make time-critical applications over the Internet a success. From my own pervious experience its contribution to the success of telemedicine is substantial. Figure 1.4 presents a concise telemedicine view about what happens between the client (patient) and the remote sever (virtual doctor). The communication in between is wireless, and the requests by the clients could risk the following:

1. Random discard by the router at the system level: If the channel is congested, the router at the system level discards requests to prevent congestion and this increases the chance of widespread retransmission. If the retransmission error probability is part of the overall channel error

probability error δ, then the average number of trials (ANT) to get a successful transmission is $ANT = \sum_{j=1}^{\infty} j[\delta^{j-1}(1-\delta)] \approx \frac{1}{(1-\delta)}$. Therefore any decrease in δ reduces ANT and shortens the client/server service roundtrip time (RTT) or response time. Dynamic buffer tuning is a powerful way to reduce δ [Wong02a].

2. Uncontrolled buffer overflow at the user level (user buffer level): Any overflow at the server request queue would aggravate the error δ, and therefore different methods were proposed to eliminate this possibility [Ip01], [Wong02a].
3. Delayed response due to a low hit ratio (i.e. high miss ratio): A low hit ratio means more chance for involving the second leg of the information retrieval process. A consistently low hit ratio is translated into a permanently high ANT. Therefore it is important to maintain a given ratio, which represents the user defined acceptable performance, consistently.

It is very difficult if not totally impossible for one to harness the service RTT for the different client/server interactions within the mobile Internet at anytime. The sheer size of the Internet and the high number of different communication protocols, both wireline and wireless, make it impractical to harness the network dynamics at all. The novel MACSC framework contributes to reduce the involvement of the second leg in the information retrieval process. As a result it contributes to shorten the service RTT. In telemedicine, which is intrinsically mobile, this heightens the chance of saving patients in the contingent situations.

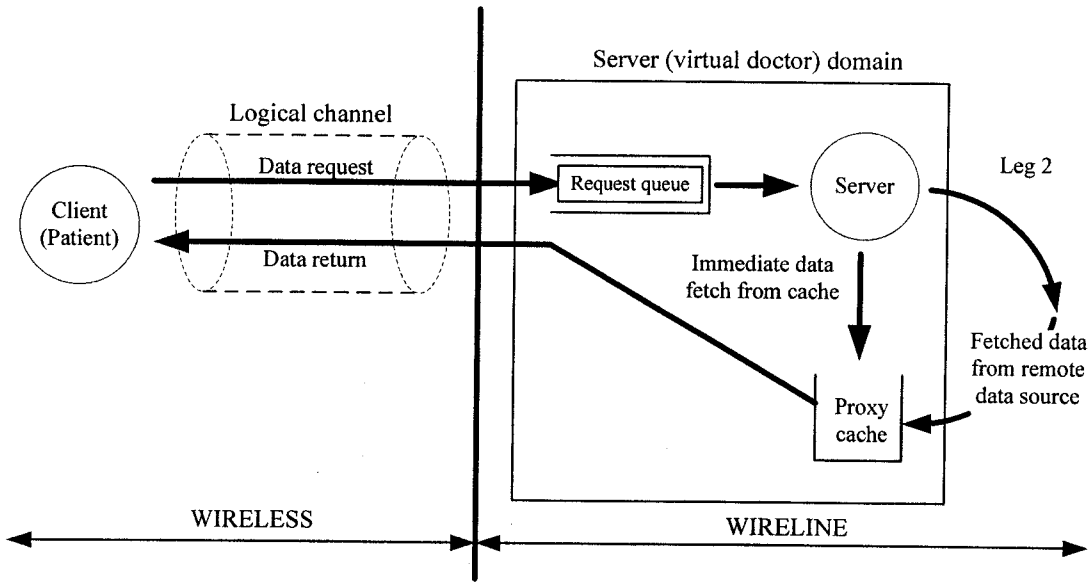


Figure 1.4 A concise telemedicine view

1.5 PLAN OF THESIS

In this thesis, the MACSC (*Model for Adaptive Cache Size Control)* conceptual framework for dynamic cache size tuning, which is supported by four different methods that compute the popularity ratio on the fly in the tuning process, is proposed. These methods led to four different MACSC approaches/solutions: MCSC(PE), MACSC(M^3RT), MACSC(F-PE), RTPD/MACSC. The thesis is organized into twelve chapters as follows: Chapter 1 is the introduction and background information; Chapter 2 describes the classification and the pros and cons of the existing caching techniques; Chapter 3 is the problem statement together with the research methodology; Chapter 4 provides the overview of the four MACSC solutions; Chapter 5 presents the details of the MACSC conceptual framework; Chapters 6, 7 and 8 describe the three different statistical methods that support popularity ratio computation and some verification results; Chapter 9 presents how real-time traffic pattern detection can be carried out and how it can support the RTPD/MACSC approach; Chapter 10 describes how to implement the MACSC

framework and some validation results; Chapter 11 describes how the proposed MACSC conceptual framework can support fast response in mobile and telemedicine applications; and Chapter 12 concludes the thesis.

CHAPTER 2

EVALUATION OF PREVIOUS RESEARCH

2.1 INTRODUCTION

The World Wide Web (WWW) is growing in an exponential rate in size everyday. The volume of pages has a monthly growth rate of around 15% but the Internet backbone capacity increases only by 60% yearly [Bharat98]. The massive amount of information needed to be transferred across the network in browsing and information retrieval can quickly deplete the amount of sharable bandwidth. This situation worsens if retransmissions are involved as a means to recover the information lost owing to different kinds of network faults, which are inevitable due to the sheer size and heterogeneous nature of the Internet. Caching is one good method to alleviate network congestion and speeds up WWW information retrieval. In this chapter, different caching techniques are discussed.

2.2 CLASSIFICATION OF CACHING TECHNIQUES

The importance of caching has motivated different areas of caching research. The aims of these areas include [Wang99]: a) fast information retrieval by a proxy server [Luotonen94], b) system robustness for fail-soft operations, c) caching operation transparency, d) system scalability, e) caching adaptivity in response to changing user demands and network environment, and f) stable collaborative caching by avoiding naive cache routing that introduces Internet perturbations. The existing caching techniques work exclusively with a static cache size.

According to the nature of the current caching techniques, they can be classified into three groups as follows:

Group 1: Infrastructure related

a. Caching architectures
b. Cache routing
c. Load balancing
d. Proxy placement

Group 2: Content related

a. Prefetching
b. Replacement strategies
c. Cache coherency
d. Caching contents

Group 3: Others

a. User access pattern prediction
b. Adaptive caching
c. Web traffic characteristics

The infrastructure related class includes the issues that relate to the overall structure or setup of the proxy servers. The content related class includes the issues that relate to the cache management of the proxy servers. The other issues are included in the group others. Figure 2.1 shows a summary classification of the current caching techniques classification.

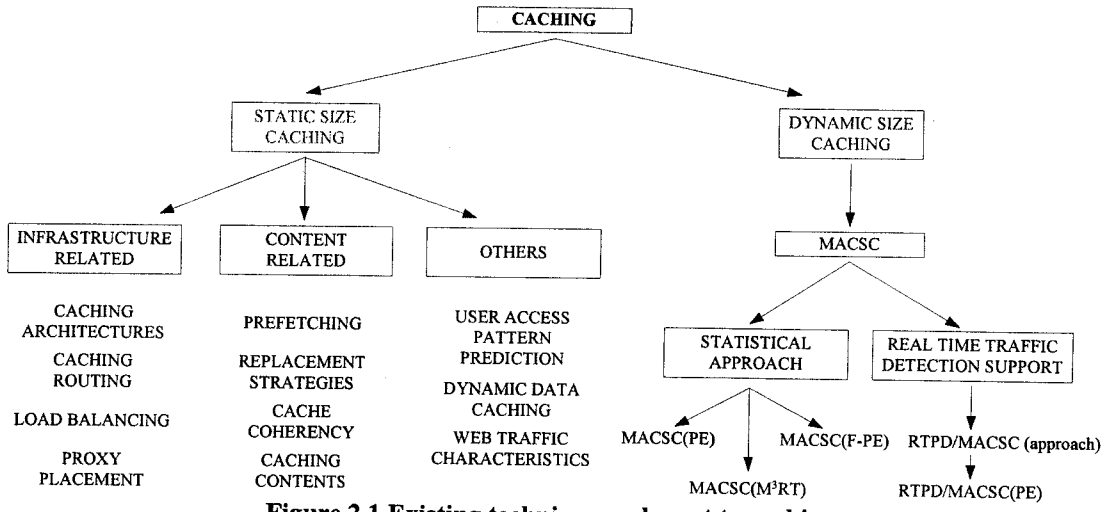


Figure 2.1 Existing techniques relevant to caching

GROUP 1: INFRASTRUCTURE RELATED:

a.) Caching architectures

The hit ratio of a caching system depends on the size and data preference of the client community. If different caching systems work together and share their contents, the overall performance of each of the cooperating caching systems can be improved. Caching architecture, which facilitates such cooperation, accommodates the member caching systems in an organized manner so that the chance of finding the required data object is increased [Chankhunthod96], [Wu00]. The cooperation can exist either in a confined environment or in a distributed manner, as exemplified by the Cache Array Routing Protocol (CARP) system [Valloppillil98]. Caching architectures from the literature can be classified into three different types: i) hierarchical, ii.) distributed, and iii) hybrid which mixes the first two.

The main idea of hierarchical caching, which was first proposed by [Chankhunthod96], has since attracted a substantial amount of later work (e.g. [Rodriguez99], [Michel98] and [Yang]). The aim is to place objects at different levels of a hierarchy, where the root is the master database of all the objects. The

“leaf” caches at the proxy level (i.e. the lowest level) which contains the hot data. If the proxy cannot find the data locally, it will go up one level and this repeats until the master database is reached. The problems of hierarchical caching include: i) possible massive bottlenecks at different levels, ii) data incoherence, and iii) long information retrieval roundtrip (RTT) time. A massive bottleneck is created when all the lower-level proxies try to access a popular high-level proxy. If a proxy is very popular and has very large cache memory, it does not need to update its cache that often. This could lead to incoherence of data between the master level and itself because the cache is not refreshed frequently. Any massive bottleneck would mean long queuing time and thus long RTT.

Another idea within the domain of caching architecture is distributed caching [Povey97], [Tewari98]. This proposes having two levels of cache servers, the bottom level cache server and institutional level. By having two levels only it differs from a formal hierarchical concept, which would have multiple levels. In the distributed case, when a client cannot locate the document in the bottom cache server, it will access the institutional level cache server. Each institutional level cache server contains meta-data information (i.e. a summary directory) of the data objects that other institutional level cache servers store. Then, the directory information of the data objects that help the client locate the data objects can be distributed quickly. Examples include Internet cache protocol (ICP) [Wessels97], Cache Array Routing Protocol (CARP) [Valloppillil98], distributed Internet cache [Povey97], central directory approach (CRISP) [Gadde97], Cachemesh system [Wang97], Summary Cache [Fan98], Cache Digest [Rousskov98], and Relais project [Relais98].

The hybrid architecture combines the hierarchy and distributed caching

techniques to optimize the advantages of both and reduce the possible disadvantages. It, however, may need to be moderated for smoother performance. For example, the hybrid model proposed by Rabinovich tries to limit excessive cooperation among neighbor caches that may lead to unnecessary delay in the information retrieval process [Rabinovich98]

b.) Cache routing

Cache routing investigates how to locate the cache that contains the data objects quickly. The routing table, which indicates what objects are located in particular remote proxy caches, is large. It needs to be updated frequently to avoid data incoherence. There are two main research topics in this area from the literature: i.) cache routing table and ii.) hashing function. The issue of cache routing table [Malpani95], [Wang97] addresses how to improve the speed of finding the needed data object from the routing table, and the examples include: harvest cache system [Chankhunthod96], Adaptive Web caching [Michel98], and manually configured hierarchy [Povey97]. The hashing function helps find the locations of data objects quickly. It involves the following: i.) how to build the summary table, ii.) how to minimize the change in a routing table, and iii.) how to minimize the search time. Hashing function examples include: summary cache [Fan98], and consistent hashing [Karger97].

c.) Load balancing

This intends to resolve the hot spot problem in a collection of collaborating proxy servers. Hot spot means that too many clients are trying to request service

from the same server. The existing systems use mainly the replication strategy to resolve this problem [Chankhunthod96], [Heddaya97], [Malpani95].

d.) Proxy placement

It investigates how to optimize the location of the proxy server so that the objectives of self-organizing, efficient routing, load balancing, and stable operations can be achieved. However, this research topic so far has yielded minimal results because very few researchers have put much effort into this topic [Li99].

GROUP 2: CONTENT RELATED:

a.) Prefetching

The aim is to heighten the hit ratio by predicting what data objects will be requested next by clients. It can be performed for the following purposes: i) between clients and web servers (C&W), ii) between proxy servers and web servers (P&W), and iii) between clients and proxy servers (C&P). The C&W is the earliest approach and its aim is to predict the objects that clients would imminently fetch by using pre-collected traces. The examples include: Prediction-by-Partial-Matching (PPM) [Padmanabhan96], model for speculative dissemination of WWW documents [Bestavros96], and rate-controlled prefetching scheme [Crovella98].

The P&W scheme is exemplified by approaches such as the following:

i. Kroeger et al. [Kroeger97]: They discovered that by combining perfect caching and perfect prefetching at the proxy server level the RTT latency can be reduced up to 60%.
ii. Markatos and Chronaki [Markatos98]: The proposal is to let web servers

regularly push the most popular data objects to the proxy servers. Techniques include those proposed by Cohen et al. [Cohen98], Wcol [Chinen97] and Geographical Push-Caching [Gwetzman94].

The C&P approach is exemplified by the model proposed by Fan et al [Fan99]. Fan relies on the proxy to predict what the clients want next and pushes or pulls the data objects in the idle time between user requests. This can reduce the RTT latency up to 23.4%. Another C&P example is the one by Loon and Bharghavan [Loon97].

b.) Replacement strategies

The aim is to enhance a proxy server's caching efficacy by pushing out the aged data in a fixed-size cache to make room for the new "hot" ones [Arlitt99], [Abrams96]. Data object replacement in a cache is the most researched topic in the area of caching. There are basically two replacement approaches: LRU (least recently used – recency based) and LFU (least frequently used – frequency based). The recency concept associates with *temporal locality*, which states that the chance η of getting a data object is inversely proportional to the elapsed time t since its last access (i.e. $\eta \propto 1/t$) [Jin00b]. The frequency of access of a data object indicates its relative popularity. The log-log plot of access frequencies versus the corresponding ranked data objects is the Zipf-like behavior [Bjarat98], [Nielsen97]. For the same set of data objects the Zipf-like behavior from the proxy point of view is different from that of the Web server (data source). The difference is due to the fact that more hot data in the cache means less remote access to these objects in the data source (i.e. they are relatively cold). A replacement strategy leverages some chosen parameters to compute the cost/index that determines which cold data objects should be evicted

first from the cache. To improve cache hit ratio the technique of "*filtration of one-timers*" can also be used. One-timers are those objects that are rarely accessed [Belloum98], and their removal is necessary to make the hot data objects in the proxy cache more concentrated for a higher hit ratio [Aggarwal99].

c.) Cache coherency

The cause of the data coherence problem (i.e. data incoherence) is that the cached data is not updated frequently enough. In fact, this is a side effect for a very large proxy cache. On one hand this kind of cache would produce a high hit ratio, but on the other hand the stale data causes problems [Breslau99]. This serious open performance problem is still being actively researched.

d.) Caching contents

The main purposes of a proxy are as follows: a) it provides the window/gateway for the protective firewall so that through this window clients within can communicate with the outside world safely [Wu05b] and b) it shortens the service RTT in information retrieval because of its cache hit ratio. The second purpose has inspired tremendous effort in replacement strategies [Aggarawal99]. The aim of obtaining a high hit ratio is to keep as much hot data content in the cache as possible. In light of contents manipulation there are two strategies/techniques that can further reduce the data object access latency. Since these strategies are value-added in nature, they are considered as performing the secondary roles for caching contents. The first strategy is called connection caching [Cáceres98], [Feldmann99], which from previous experience might reduce access latency by up to

40%, and it emphasizes two persistent connections: “between the client and the proxy” and “between the proxy and web server”. The second is called computation caching [Wang99], which aims at making normally non-cacheable dynamic data cacheable. The reason why dynamic data, which are dynamically generated and exist for only a short period, are non-cacheable is that by the time the client wants to access this piece of data, it has already disappeared. The technique is to let the web server pass some of its dynamic data related computation capability to the proxies. Then, the proxies can have a better chance to generate, cache, and maintain the dynamic data concerned. For example, the techniques such as active cache [Cao98] and server accelerator [Levy-Abegnoli99] can achieve such a purpose. Other examples include [Chinen97] and [Challenger99].

GROUP 3: OTHERS:

a.) User Access Pattern Prediction

The objective of the user access pattern prediction is to improve the efficiency of the proxy servers by predicting what users would need imminently. Examples include [Cohen99], and [Yang] that proposed to put the data objects that may be accessed by clients together in the file system based on the access patterns. Other examples, which include [Fan99], [Palpanas99], and [Padmanabhan96], proposed to use the Partial Match model to do the predictions.

b.) Adaptive Caching

The focus of the current research reported in the literature in adaptive caching is to run the replacement dynamically [Podlipnig03]. The researchers proposed to

weigh the parameters in different situations to guide the replacement process [Podlipnig03], and to selectively apply different predefined replacement strategies.

c.) Web traffic characteristics

There are two related issues. The first is the nature of the workload. One of the metrics [Douglis97] in this case is temporal locality, which is related to the LRU approach [Wang99], [Podlipnig03]. The second is the inter-arrival times (IAT), which can affect the stability of the caching system [Paxson95], [Taqqu03], [Willinger03].

2.3 WEAKNESS OF PREVIOUS TECHNIQUES

Caching is generally recognized as a technique to shorten the service RTT in information retrieval over the Internet. This is achieved by improving the proxy cache hit ratio through the use of different methods as summarized in Figure 2.1. However, there are several weaknesses of the existing techniques:

1. Static cache size – the problem of the previous methods is that they can produce a high hit ratio but do not necessarily maintain it. The reason is that they exclusively work with a cache of fixed/static size.
2. Heavy parameterization – the most researched topic in caching is the issue of replacement strategy. The previous strategies can get very complicated by leveraging many parameters such as workload, recency, and relative object popularity [Jin00a]. Heavy parameterization can be counterproductive because of its long computing time. By the time the remedy is computed the real problem has already gone. Using the remedy to correct a spurious

problem can produce deleterious/undesirable effects.

3. Internet traffic – the existing hit-ratio enhancing strategies do not take Internet traffic into account. This can be dangerous because the Internet traffic is changing continuously [Paxson95].

In this thesis, we propose a framework that can solve these problems. It can maintain the given hit ratio by dynamic cache size tuning. It also supports the time-critical applications with short computation time to avoid deleterious effect. Furthermore, it has the ability to ward off the ill effects by different Internet traffic patterns.

2.4 CONNECTIVE SUMMARY

In this chapter different previous caching techniques from the literature are evaluated. These techniques have different strengths and weaknesses. Even though they may generate a high hit ratio, they cannot necessarily maintain it because they work with a static cache size. In addition, these techniques do not compensate for the ill effects caused by the changing Internet traffic pattern, which can be SRD or LRD. Hit ratio maintenance needs the support of dynamic cache size tuning, which naturally works with a variable cache size. This is the basis of my PhD research and the problem statement and methodology will be presented in the next chapter.

CHAPTER 3

PROBLEM STATEMENT AND MEHODOLOGY

3.1 INTRODUCTION

Without caching support the Internet can easily become terribly congested, slow and lose its appeal. The danger of congestion is aggravated by the fact that the World Wide Web (WWW) volume of pages has a monthly growth rate of around 15% but the Internet backbone capacity increases only by 60% yearly [Bharat98]. The massive amount of information needed to be transferred across the network in browsing and information retrieval can quickly deplete the amount of sharable bandwidth. This situation worsens if retransmissions are involved as a means to recover the information lost owing to different kinds of network faults, which are inevitable due to the sheer size and heterogeneous nature of the Internet. As explained earlier, caching alleviates network congestion and speeds up WWW information retrieval by providing two advantages. The explicit advantage is the shortening of the service roundtrip time (*RTT*) for WWW information retrieval. The service *RTT* is the time interval between sending a request by the client and getting the corresponding result from the server correctly. The service RTT in this client/server relationship conceptually consists of two legs. The first leg is for the roundtrip between the client and the proxy server, and the second leg is between the proxy server and the remote data source or web server. If the proxy server finds the data object in its cache, then the second leg is automatically obviated. The hit ratio is

the probability of finding the required data locally in the proxy's cache.

The importance of caching for the Internet can be separated into two main aspects, namely, from the view of a wired network, and from that of a "wired and wireless" infrastructure:

1. Wired network: The main concern here is how to use caching to alleviate network congestion, which can be caused by many factors, including unpredictable router bottlenecks [Braden98] that can cause sizeable fluctuations in traffic throughput in the network, resulting in system-level and user-level buffer overflows [Wong02a]. This is compounded by the ever growing data object size [Bjarat98], [Duska97]. Caching as a technique can resolve the problem of ever growing data object sizes. Nowadays data objects which were mostly textual in the past, are more complex and frequently tend to be multi-media. As mentioned above, the different empirical studies (e.g. [Bjarat98], [Duska97]) have found that web page size increases by 15% monthly but the Internet backbone capacity improves by only 60% yearly. If this continues, the Internet backbone bandwidth will be unable to sustain the huge number of large web objects to be transferred across the Internet. The result would inevitably be massive network congestion that would make the Internet impractical. The congestion problem is real because of the following reasons: a) WWW is relatively inexpensive and faster than other means, and b) WWW provides a wide range of popular information such as daily news, entertainment programs, weather reports, transportation schedules, financial news, and e-shopping.
2. Wireless network: Wireless communication is basically unreliable and has a

high loss rate. In contrast to the wired network, in which it is possible to ask the sender to reduce transmission to alleviate congestion (i.e. throttling), the wireless network improves reliability by fast retransmissions. Therefore, wired and wireless networks have contradictory requirements to deal with data losses due to network congestion or other faults that include traffic ill effects [Paxson95]. If the proxy server does not need to search for the required data objects from remote data sources again for retransmission, then it supports fast retransmission for the data object that is immediately available in its cache. It implies that the cache always keeps the hot/popular data objects [Tanenbaum96].

3. "Wired and wireless (W&W)" situations: In the context of this research these situations are called the mobile distributed systems (MDS), which will be discussed in the next section. The presence of caching with a "surrogate" server balances the needs of both the wired and wireless sides and resolves the contradiction. On the one hand the surrogate cache keeps as much hot data as possible so that fast retransmissions on the wireless side can be supported. However, the same cache of hot data can prevent the surrogate from conducting excessive remote searches for the required data sources by cyber foraging [Garland02].

3.1.1 MOBILE DISTRIBUTED SYSTEMS (MDS)

These systems are all of the "W&W" nature, and the wireless side can be mobile or pervasive. Examples such as those mentioned above, involve many different cutting edge technologies for support, including location-aware means to

determine the whereabouts of the SFF clients, time-critical communication and information retrieval techniques, trustworthy wireless and wired operations, and advanced SFF devices with a reasonable battery life. It is also useful to have a system that addresses the issue of time-critical information retrieval. It improves the serviceability of the server in a client/server interaction or asymmetric rendezvous [Lewandowski98]. All the *pervasive* "wireless-wired" (or simply W&W) computing setups, however complex, are *mobile distributed systems* (MDS). The main obstacle today that prevents confident MDS deployments is the *dependability* problem [Avizienis04], [Laprie95] that includes many attributes such as reliability, availability, maintainability, and serviceability. The serviceability attribute is only loosely defined in [Laamanen99] as the "*ability of a service to be obtained*". For the research in this thesis it is redefined as the "*ability to obtain the required service at the appropriate quality-of-service (QoS) level within a defined period*". For example, obtaining a service within a deadline is a *QoS* requirement.

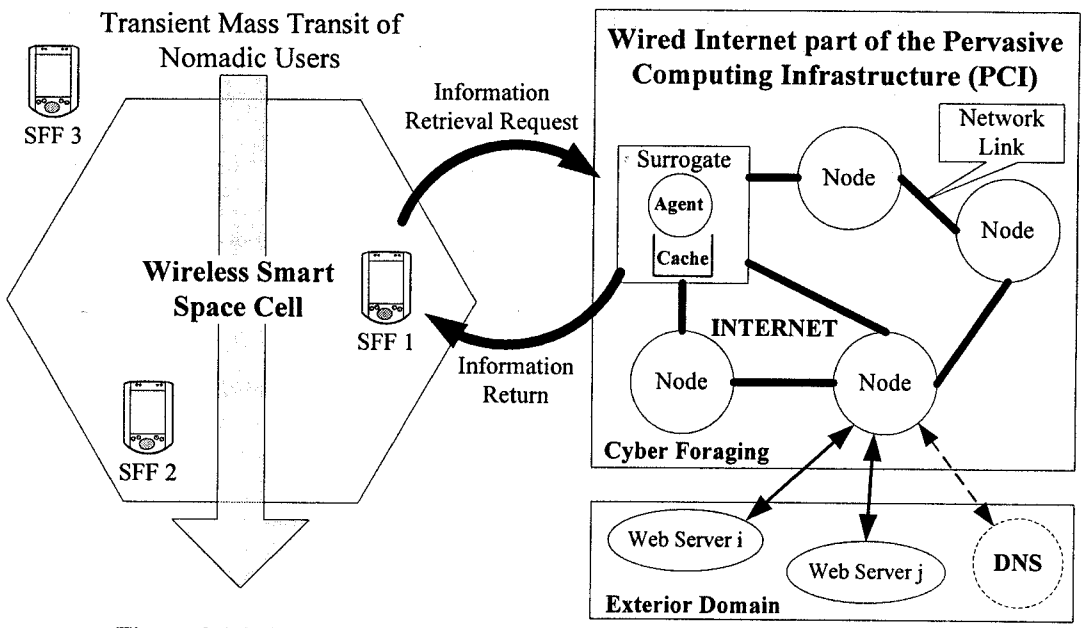


Figure 3.1 Information retrieval by a SFF client in the W&W architecture

Figure 3.1 illustrates the information retrieval actions in a MDS, which has a wireless smart space [Garlan02], [Weiser91] and a supporting wired PCI (Pervasive Computer Infrastructure). The nomadic users follow the transient mass transit and pass through the wireless smart space of the W&W system. In the light of information retrieval "on the run", they use their SFF devices to make requests. Therefore, the number of retrieval requests is tied to the mass transit traffic pattern [Malla03]. The SFF device interacts with the assigned surrogate node in the PCI. The surrogate, which is temporarily assigned to provide assistance, is a gateway to other PCI nodes. It houses different logical servers or agents. If a surrogate can enlist help from the other PCI nodes, the result is *cyber foraging* [Garlan02]. For example, if an agent, which is a proxy server, cannot find a data object requested by a SFF client locally, it may ask for help from other PCI nodes through its surrogate. This is easily achieved if the PCI operates with the peer-to-peer content distribution concept [Androutsellis-Theotokis04], [Wang99], as demonstrated by the Gnutella framework [Gnutella]. This framework allows direct sharing of computer resources without a central server. The peer-to-peer architecture scales and self-organizes freely in response to sudden increases in the number of network nodes and partial failures.

If the data object to be retrieved is not in the PCI domain, then the latter enlists the DNS (*Domain Name Server*) to help locate it in one of the remote data sources (web servers). Therefore, data retrieval in the W&W architecture is similar to the wired network by having two "*legs*" in the information retrieval process. The first leg is the average RTT delay between the SFF client and the agent server, which finds the information locally. The second is the average RTT required to find the information object involving other PCI nodes and possibly the DNS. The importance

for the agent server to have a local cache is shown by the speedup calculation as given in equation (3.1).

$$S_{cache} = (RTT_{1eg1} + RTT_{leg2}) \Big/ (RTT_{leg1} + [1-\psi] * RTT_{leg2}) \quad (3.1)$$

In this equation, the average RTT for the first leg is RTT_{leg1} and that for the second leg is RTT_{leg2}. If the average hit ratio ψ of the agent's cache is 60% or 0.6 (i.e. miss ratio is (1- ψ or 40%), for $RTT_{leg1=}1$, *and* $RTT_{leg2}=10$ then the speedup for the retrieval operation is 2.2 fold. The agent's 60% cache hit ratio is the chance of obviating RTT_{leg2} to produce this speedup. The calculation of S_{cache} in this case is as follows:

$$S_{cache} = (1+10) \Big/ (1+[1-0.6]*10) = 11 \Big/ 5 = 2.2 \quad (3.2)$$

The S_{cache} speedup benefit provided by caching is essential for sharing WWW information and data efficiently. This benefit has inspired different areas of research in caching as discussed in chapter 2. All the strategies and algorithms from the literature are aimed at producing a high S_{cache} value but not necessarily maintaining it. These approaches use a fixed/static cache size and leverage different parameters. In fact, too many parameters could be counterproductive because heavy parameterization leads to long execution time and deleterious effects. A long execution implies that by the time the caching solution is computed the actual problem may have already passed. The solution ends up correcting a spurious problem and leads to undesirable or deleterious consequences. Maintaining a given hit ratio requires dynamic cache size tuning. It is also important any solutions developed in this thesis should utilize light parameterization.

In fact, the loop of "information retrieval request and information return" in the

Figure 3.1 is a high-level view. If this view is put into the e-diagnostic process such as in telemedicine applications, then it is the sending of vital signs to the IMS (Intelligent Medical Server) or virtual doctor, which will return the diagnostic results in the fastest possible time to save lives. This will be explained in more detail later in chapter 11. With respect to Figure 3.1 the sender is simply a SFF device and the IMS is a specialized logical server in one of the surrogate.

In chapter 1, we noted the importance of web caching in reducing RTT time. Figure 1.1 has illustrated this, and Figure 3.1 has further illustrated this by the "information retrieval request and information return" cycle. In all the networks the average RTT is a factor for measuring performance. In time-critical applications a shorter RTT is always essential in order to meet the scheduled deadline. For example, in Figure 3.1 the performance in term of speedup can be represented by equation (3.1). The second leg for the RTT in this case involves the DNS and therefore can be excessively long. In fact, it generalizes the similar impact by RTT on the performance of wired, wireless and W&W situations. An important feature is to provide not only reduced RTT but also same predictability in RTT. It is important to develop a solution that gives a minimum hit ratio. In this chapter, we will give a description that is necessary to more precisely formulate the problem definition. In chapter 2, the review of the literature describes the need to produce a framework that maintains a given minimum hit ratio as the best approach to caching.

3.2 PROBLEM DEFINITION

The aim of the thesis is to develop an efficacious dynamic cache size tuner that can shorten the client/server service RTT by maintaining the given cache hit ratio

under all operation conditions. This is achieved by obviating the second-leg delay in information retrieval to produce the S_{cache} speedup shown in equation (3.1). Dynamic cache size tuning is a relatively pristine area with little experience published in literature. It is important to understand what kind of characteristics are there so that the proposed conceptual framework for dynamic cache size tuning is suitable for real-time applications. It has become known that the Internet traffic, which follows the power law [Medina00], can assume different patterns at different times. These patterns can cause system failure if the system is designed with a preconceived mathematic model such as Poisson [Paxson95], [Lewis96]. In order to prevent the final proposed conceptual framework from falling into similar pitfalls, they should be statistical and work by direct data measurement. The dynamic cache size tuning models are different realizations of a generic framework, which is based on a well defined principle, for example, the "Zipf Law" [Breslau99], [Glassman94]. Yet, this generic model should be evaluated with respect to some applications, for example mobile and telemedicine applications. Since I have done some research in TCM (Traditional Chinese Medicine) data mining in my previous MPhil thesis [Wu02], the dynamic cache size tuning model(s) proposed in this research will be evaluated in this direction. The aim is to ensure that these model(s) would yield the same efficacy in wired and "W&W" situations.

3.3 DEFINITION OF TERMS

In order to make the problem statement for my PhD research more understandable and precise the following terms are defined:

a. Asymmetric rendezvous: It is the client/server interaction relationship, in

which the server is simultaneously serving different clients.

b. Cache: It is a local memory where the proxy server keeps all the popular/hot data.

c. Caching parameters: Every placement algorithm leverages different metrics to ensure enough hot data is in a cache of static size. These metrics include “least frequently used objects”, least recently used objects” and others.

d. Client: It is the service user.

e. Compensation measure: A compensation measure is to bring any deviation back from the norm of system operation.

f. Deleterious effect: It is caused by the fact that the time needed for computing the remedy is longer than the problem duration. The computed remedy ends up correcting a spurious problem can cause undesirable side effects.

g. Dynamic cache size tuning: It is a technique to maintain a given hit ratio under all operation conditions by leveraging the chosen network parameter(s). It differs from yielding a high hit ratio because it involves a variable cache size, which is adjusted on the fly. Existing placement techniques [Bresalu99], [Aggarwal99] leverage different parameters to yield a high hit ratio but do not maintain it because they all work with a static cache size.

h. Hit ratio: It is the chance of finding a requested data in the proxy cache, and the chance of not finding it is the miss ratio. If the data is not found, then the proxy has to search the data object via the DNS (Domain Name Server) of the Internet form the remote data source(s) or web server(s).

i. Inter-arrival time (IAT): The time between the arrival of the $object_i$ and $object_{i-1}$.

j. Internet traffic and traffic pattern: The Internet follows the power/hyperbolic law and its traffic pattern changes over time, for example, from SRD (short-range dependence) to LRD (long-range dependence) or vice versa. This kind of change can lower system stability or cause failure.

k. Long range dependence (LRD): It is the characteristic of the stationary stochastic processes with slow decay of correlations such as self-similar and heavy-tailed.

l. Minimum given hit ratio: A dynamic cache size tuner can yield two effects. The first is to try to maintain the given hit ratio but with oscillations that sometimes bring the final hit ratio below the given value. The second is to upkeep the hit ratio so that it is consistently above the given value. A tuner that yields the given hit ratio as the minimum is definitely more efficacious than the one that produces the first effect and provides more performance advantage.

m. Parameter tuning: Some placement algorithms would tune the parametric values to produce a high hit ratio, but they cannot maintain it because it works with a static cache size. Besides, work with too many parameters would lead to heavy parameterization, which needs long execution time to compute the hit ratio and causes deleterious effects.

n. Real-time traffic pattern detection (RTPD): This is the technique to determine the traffic pattern anytime on the fly. This technique is novel because the existing techniques are exclusively for off-line or postmortem analysis with pre-collected traces.

o. Relative object popularity (ROP): This is a distribution of access frequencies

versus the specific objects over the period of interest. The standard deviation of the distribution changes if the users shift their preference for some objects in the period.

p. Roundtrip time (RTT): It is the interval between the sending of a request by a client and the correct reception of the corresponding answer. This interval is affected by the delays caused by retransmission; the more transmissions the longer the RTT.

q. Proxy server: It is the gateway for the clients within a protecting firewall to communicate with the outside world.

r. Server: It is the service provider.

s. Service response time: It is the duration between sending a request and receiving the correct service result from the server by a client.

t. Short range dependence (SRD): It includes Markovian traffic patterns such as Poisson.

u. Stochastic process - A process that works with operation paths and any path is traced from one end to the other by events at the intermediate hops. It is discrete over the Internet.

v. Zipf-like behavior: Every sizeable caching system has a specific Zipf-like behavior defined as $y(r) \propto (1/r)^{-\beta}$, where r is the ranked position of the object with respect to its access frequency relative to others. The β parameter, which is within the $0 < \beta \leq 1$ range denotes this relativity, and for $\beta = 1$ it is called the Zipf Law.

3.3.1 PROBLEM STATEMENT

The aim of the research is to develop a technique for a dynamic cache size tuning framework that can maintain the give cache hit ratio consistently under all Internet conditions. This conceptual framework should support the following salient features:

a. The framework should leverage as few network parameters as possible so that its computation time is short enough for real-time applications and it does not cause serious deleterious effects at the same time.
b. The cache size adjustment computation on the fly should be accurate and quick, and independent of Internet traffic patterns. This requires efficient statistical sampling techniques, which are likely to be based on the *Central Limit Theorem* (CLT) [Chis92], [Jain91].
c. The final framework should accommodate a real-time traffic detection (RTPD) system/mechanism and use the detected result to self-reconfigure to compensate for the traffic ill effects on the tuning accuracy.
d. The RTPD mechanism should be able to work accurately with sample aggregates of a discrete stochastic process [Leland94], [Taqqu03] and [Willinger03].

Therefore the objectives of the project include the following:

a. Propose the conceptual framework that leverages the minimum number of network parameters for accurate dynamic cache size tuning.
b. Propose compensation measures that help the framework work healthily under extreme conditions.
c. Propose a RTPD mechanism that is suitable for time-critical applications so

that it can be included as part of the framework, which with RTPD support can reconfigure itself for more efficacious dynamic cache size tuning on the fly.

d. Verify the generic framework with simulated and live datasets/traces.
e. Verify that the proposed framework can indeed support wired and wireless client/server interactions with similar efficacy.
f. Verify that this efficacy applies to mobile and time-critical applications such as telemedicine.

3.4 METHODOLOGY

In order to make research a success methodology in strategy and management is very important [Ketchen04]. Usually a research process can be divided into many phases, for example, the eight-step model by Kumar [Kumar96]. The steps are: a) formulating the problem, b) conceptualizing the design, c) constructing an instrument for data collection, d) selecting the sample type for testing, e) writing the research proposal, f) collecting the selected data type, and g) analyzing and processing the sampled data. The research strategy and methodologies must suit the problem domain for good results. In fact, there are many existing strategies and methods in general to cover different problem domains. In the area of computing research three basic types of research can be identified, namely [Philips87]:

1. Exploratory research: This type tackles a little known problem for which the research details cannot be formulated very well at the beginning. The result of this kind of research usually pushes out the knowledge frontiers and leads to discovery of new knowledge.

2. Testing-out research: This type is to find the limits of previous generalizations.
3. Problem-solving research: This type usually starts with a specific real-world problem of well-defined characteristics and then brings all the available intellectual resources together for a reasonable solution.

Once a suitable research methodology is decided, then one can adopt one of the following basic strategies:

1. Top Down: The objectives are defined and realized step by step. The typical examples include: 1) Waterfall Model for software engineering that does not encourage user intervention, and 2) the Fast Prototyping that encourages repetitive user input until the system is finally accepted.
2. *Bottom Up*: The coordination framework as a connective is first proposed so that what is available (commodities and/or intellectual resources) are interconnected into a single system. The complexity of the system increases with time.

The Top Down approach is suitable for testing-out research, and the Bottom Up approach is natural for the problem-solving type.

By nature this PhD research is exploratory because the topic of dynamic cache size tuning has little published experience. Even though this would produce a prototype for testing, which will support further research at the end, the process is naturally top-down because the course of research includes literature search, problem statement, proposed solutions, and data collection. It is, however, difficult to apply the Top Down approach in a strict sense because early exploratory investigations that produce unpredictable results are necessary. That is, the whole investigation would

involve repetitive backtracking and cross-referencing to gain the necessary insight for the next step. This means a need to find a research methodology that can cater to the repetitive and exploratory nature of the research. After a careful consideration, I decided to customize the original methodology proposed for my previous exploratory MPhil research [Wu02], namely, "*investigate & experiment & proceed with possible backtracking, cross referencing and looping* (IEP)" approach for the PhD investigation.

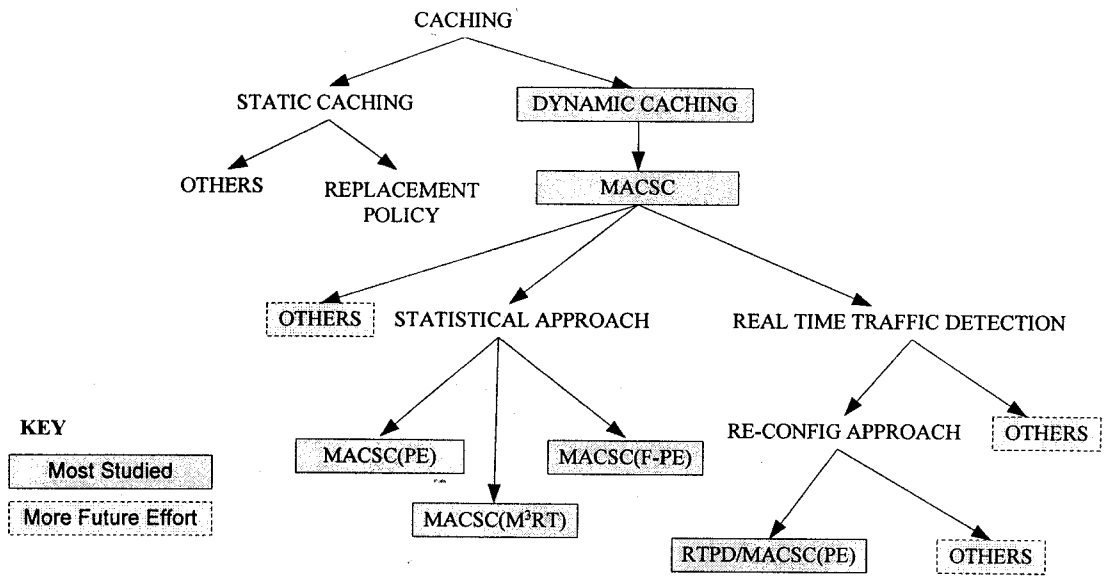


Figure 3.2 The roadmap for project management

The main difference between the IEP and a strict top-down approach is that in the IEP approach explorations and investigations start from the root through the branches to a leaf and then back again. The traversals up and down the branches and leaves represent a heuristic process, and these traversals may repeat many times before enough material, data and insight can be consolidated for the next stage of the research. In fact, the detailed IEP approach is illustrated by the roadmap in Figure 3.2, which is a roadmap that identifies what should be explored and achieved for completing the thesis. For example, an IEP traversal may be the following path:

Dynamic cache size tuning → Propose suitable model(s) → Statistical computation method(s)→ Real-time traffic pattern detection(RTPD)→ Combine RTPD and the proposed tuner model(s) to ward off the ill effects of traffic patterns on tuning accuracy→ Implementation issues→ Verify the effectiveness of the proposed tuner(s) to mobile and telemedicine applications. This path, however, represents one of the many possible "*operation*" paths in the course of the project because traversals back and forth are necessary for cross-reference, data refinement and/or comparison. Those items that should be investigated in the beginning of the research are in "*solid-line boxes*" and those "*dotted-line boxes*" should be investigated later (e.g. the second phase). The research is separated into two different phases because the results from the first phase would orient (or re-orient) the direction of the second.

The traversals may involve the following sequences with backtracking:

1. Understanding the rationale of caching in general,
2. Studying some general caching approaches and statistical approaches,
3. To propose a conceptual dynamic caching framework, MACSC (*Model for Adaptive Cache Size Control*),
4. Implement the MACSC with different statistical approaches,
5. Looking for a stable mobile-agent platform for testing purposes,
6. Refining the MACSC frameworks for better data collection and analysis, and
7. Demonstrating how the MACSC framework can be implemented in the real environment.

3.5 CONNECTIVE SUMMARY

In this chapter the importance of effective web caching over the Internet to the

wired and wireless networks is discussed. A good caching system improves the performance of the network by reducing the RTT in a client/server interaction. This is achieved because if the required data is found in the local cache then the delay that involves the DNS is obviated. Caching can make use of a static cache size or a variable one. The latter case, which emphasizes maintenance of a given cache hit ratio on the fly, is called dynamic cache size tuning. Developing a novel conceptual framework for such a purpose is the problem to be achieved in this PhD thesis and the overview of the proposed solutions is provided in the next chapter. The roadmap for the research and project management is depicted in Figure 3.2.

CHAPTER 4

OVERVIEW OF SOLUTIONS

4.1 INTRODUCTION

In chapter 3, we defined the problem tackled in this thesis, namely the development of a dynamic cache size tuner to maintain a given hit ratio on the fly. In this chapter, the conceptual framework for dynamic cache size tuning, which is known as the MACSC (*Model for Adaptive Cache Size Control*), is proposed. This framework is based on the concept of the Zipf-like behavior, which is apparent in large caching systems [Glassman94], [Breslau99]. The MACSC would maintain a given hit ratio on the fly by leveraging a single parameter, namely, the relative object popularity (ROP) profile. In reality this profile changes with time due to the change of user preference for particular data objects for the period. The meaning of the ROP profile depends on the following views:

1. General view: This is the distribution of access frequencies versus the "unsorted objects".
2. Zipf-like behavior: This is the log-log plot of $y(r) = (1/r)^{-\beta}$, where *y(r)* is the "logged" access frequency and *r* is the log value of the ranked position of the object.
3. Popularity distribution (PD): This is formed by mapping *y(r)* into a bell-shaped curve. The mapping mechanism is *bell(r)=map(y(r))+e* (i.e. equation (5.1) in chapter 5), where *e* denotes the possible mapping error.

The MACSC framework is based on *y(r)* and *bell(r)* above. The argument is that if the PD profile changes, then this change is reflected immediately by the current standard deviation SD. The key issue is then how to measure the SD value quickly and accurately on the fly in a statistical manner because the popularity ratio (PR), namely, (SD_i/SD_{i-1}) or $(SD_i/SD_{i-1})^2$ can be calculated from two successive SD values, where *i* denotes the computation cycle. With the popularity ratio the cache size adjustment for maintaining the given hit ratio can be computed. The computation time must be short so as not to produce any deleterious effect, and this is the reason why the MACSC leverages only a single metric, namely, the ROP profile. In this way the MACSC is suitable for time-critical application because of its short execution time.

Computing the SD value of the PD profile on the fly accurately and quickly is no trivial matter because of the following requirements:

1. Waveform dependence: The accuracy should be independent of the waveform type because any method based on a preconceived mathematical model would lead to failure [Paxson95]. The SD value computation involves two waveforms or distributions, namely, the PD profile and the inter-arrival times (IAT) among the data object access. The negative impact of IAT may be serious. If the average IAT is much shorter than the MACSC execution time, then many data items could be missed in the sampling process, making the PR value inaccurate and the given hit ratio ineffectively maintained. Therefore the criterion for choosing a technique to compute the SD value of the PD quickly and accurately is the *Central Limit Theorem* (CLT); that is, the computation is statistical and by direct

data measurement. The MACSC computation time is its limit for time-critical or real-time applications. The execution time is, however, not fixed in terms of physical time but fixed by the number of clock cycles required as indicated by the Intel® VTune™ Performance Analyzer [VTune] (more details in the later sections).

2. Sensitivity: The power of the MACSC relies on the sensitivity of the SD leveraging/computation mechanism. The SD value is crucial for remedying the cache maintenance process correctly in a short interval. The sampling process involved must have enough sensitivity to dynamically follow the contour of changes in the ROP profile.

In this research four different methods, which represent an evolutionary process, were proposed as follows:

1. Point estimate (PE): This CLT based method is sensitive to changes in the ROP profile, but it generates a lot of oscillations due to the lack of a feedback system.
2. M^3RT: This CLT based technique is transcribed from another problem domain, namely, Internet End-to-End Performance Measurement (IEPM) [Cottrel99], [Cottrel01]. Previous experience [Ip03] has confirmed that it always yields the mean of any waveform accurately in real-time applications because it has a feedback loop. The good quality of stability of this technique, however, becomes a liability for MACSC application because there is a need to strike a balance between stability and sensitivity, and this led to the proposal of the “fine-tune point estimate (F-PE) technique.

3. F-PE: It combines the merits of PE sensitivity and M^{3}RT stability and accuracy due to the presence of a feedback loop. The goal is to decide how much feedback should be allowed and incorporated into the PE process.
4. RTPD: It was observed in the early experiments that the Internet traffic patterns have different ill effects on the MACSC accuracy. Therefore, a RTPD (real-time traffic pattern detection) mechanism is included in the original MACSC framework to make the newer RTPD/MACSC approach. The aim is to let the RTPD mechanism identify the traffic pattern at the time so that the MACSC can "*reconfigure*" itself to neutralize the traffic ill effects on the fly. In the MACSC research context, the RTPD/MACSC approach is a form of reconfigurable dynamic cache size tuning.

As a result there are in total four specialized solutions derived from the general MACSC framework. The specialization is determined by the technique whereby the PD's SD is computed on the fly in each solution. The four solutions are: MACSC(PE), MACSC(M^{3}RT), MACSC(F-PE), and RTPD/MACSC(PE) and they will be introduced in the next section.

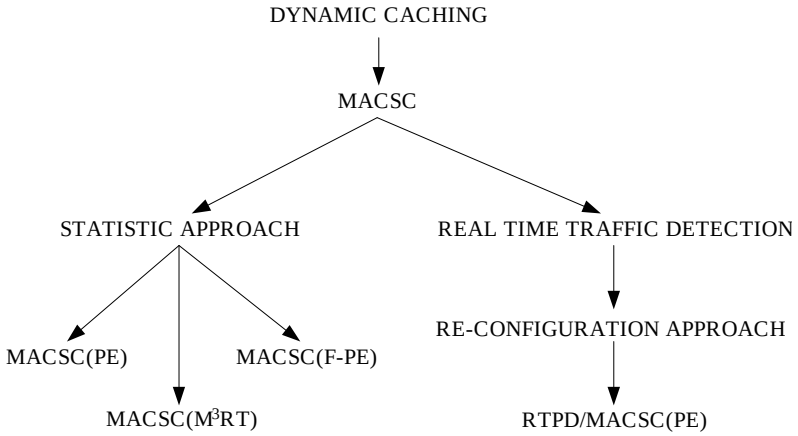


Figure 4.1 A summary of the research roadmap

The concise representation of the roadmap for project management is shown in the Figure 4.1. In this hierarchy my research effort is represented by the right branch, namely, dynamic caching, which leads to the proposal of the MACSC dynamic cache size tuning framework. The two major factors that affect the stability and efficacy of the framework are: a) using the correct statistical approach and b) real-time traffic pattern detection, which addresses the issue of MACSC reconfiguration.

4.2 OVERVIEW OF SOLUTIONS

4.2.1 MACSC FRAMEWORK

The MACSC (*Model for Adaptive Cache Size Control*) conceptual framework is novel and original because it is the only known approach that addresses the issue of dynamic cache size tuning [Wong03]. It maintains the given cache hit ratio adaptively and consistently by tuning the cache size according to the current popularity ratio currently computed on the fly. The hit ratio maintenance not only shortens the client/server roundtrip time (RTT) consistently but also reduces the need for long-haul data transfer. As a result more backbone network bandwidth is freed for public sharing leading to better network throughput. The MACSC rationale is "*reasonable memory usage to maintain the given cache hit ratio*". The cache size has to be timely increased/decreased to accommodate enough hot data objects to satisfy the given hit ratio, without excessively consuming the system memory resources.

The MACSC tunes the cache size by leveraging the *relative object popularity* (ROP) of data objects as the sole parameter. The framework is based on the Zipf-like behavior that is intrinsic to large traces of cached data objects [Breslau99], [Zipf].

MACSC transforms the Zipf-like curve of the data objects into the bell curve conceptually. This bell curve is called the popularity distribution, which provides the basis for the popularity ratio computation to determine the size of the cache adjustment. In the research it was found that some data traces did deviate from the Zipf-like behavior. To compensate a deviation behavior compensation measure is proposed for the MACSC. The detailed discussion of MACSC framework will be presented in chapter 5.

4.2.2 MACSC(PE)

In the MACSC framework the cache size is adjusted according to the popularity ratio computed for the data object access pattern. The popularity ratio computation accuracy is affected by two factors. The first factor is the speed at which the popularity profile changes its shape. Any change as such, in effect, is a reflection of the change of user preference for particular data objects and this requires the computation method to be distribution/waveform independent. The second factor is the inter-arrival times (IAT) among the data object retrieval requests.

The statistical *point-estimate* (PE) method was selected for use in the first MACSC implementation. The PE computation accuracy is inherently independent of the shape of the waveform because it is based on the *Central Limit Theorem*. The PE based MACSC framework is called the MACSC(PE). The detail of the MACSC(PE) is discussed in the chapter 6.

4.2.3 MACSC(M^3RT)

Although the PE method can estimate the standard deviation of the data objects

accurately in most of the cases, it has two shortcomings: unpredictable computation time and serious hit ratio oscillations. The unpredictable computation time problem is due to the impact of the inter-arrival times (IAT) lengths. For example, in a burst mode the IAT may be shorter than the sampling ability of the controller per sampling cycle time. As a result the controller misses data items repeatedly and inadvertently makes wrong calculations that lead to undesirable or deleterious effects. For these reasons, an alternative to the PE approach has to be explored, and this leads to the adoption of the Convergence Algorithm (CA), M^3RT (*Micro Mean Message Response Time*) technique, which is also based on the *Central Limit Theorem* and therefore its accuracy is waveform/distribution independent. This technique was originally proposed for IEPM (*Internet End-to-End Performance Measurement*[Cotrell99]) applications and the aim is to estimate the mean message response time accurately and quickly. The M^3RT can eliminate the problem of unpredictable computation time because it works with a chosen number of data items in every prediction cycle. This number is called the *flush limit f* and the range of $9 \leq f \leq 16$ always yields the quickest convergence in IEPM applications The M^3RT based MACSC framework is called the MACSC(M^3RT). The MACSC(M^3RT) approach provides several benefits that make its computation time more predictable, and those are as follows: a) it maintains the prescribed hit ratio efficaciously, b) it lessens cache size oscillation, and c) it uses a fixed number of data samples. The solution is unique because: a) it utilizes the relative popularity of the data objects as the sole control parameter and b) it tunes the cache size adaptively by direct data measurement with the CA support. The details of the MACSC(M^3RT) approach will be discussed in chapter 7.

4.2.4 MACSC(F-PE)

Although the MACSC(M^3RT) approach can resolve the computation time and oscillation problems of the MACSC(PE), it uses more memory. Therefore, the third approach, namely, MACSC(F-PE) is proposed, where "F-PE" stands for *fine-tuned point estimate*. This approach combines the advantages of the PE sensitivity and the M^3RT feedback stability. It uses the PE technique to estimate the standard deviation of the data objects, but with the support of a feedback system that uses history to moderate the PE process. The details of the MACSC(F-PE) approach will be discussed in chapter 8.

4.2.5 RTPD/MACSC(PE)

The Internet follows the power law [Medina00] and its traffic assumes different patterns over time, for example, long-range dependence (LRD) and short-range dependence (SRD) [Molnár99]. Continued studies of different traffic patterns led to the conclusion by Paxson [Paxson95] that any system, which is designed with a preconceived mathematical model in mind (e.g. Poisson), would fail over the Internet. In the different experiments with the MACSC(PE) prototype, it was observed that different IAT traffic patterns can affect the dynamic cache size tuner's accuracy. The new framework, namely, RTPD/MACSC(PE) is proposed to resolve this problem. The *real-time traffic detection/detector* (RTPD) capability is incorporated into the MACSC framework. In the RTPD/MACSC(PE) approach the MACSC mechanism uses the RTPD capability to detect and identify the specific traffic pattern embedded in the IAT traffic so that it can reconfigure itself to deal with traffic ill effects.

4.2.6 RTPD/MACSC(F-PE)

Logically this solution should be useful because the RTPD capability could neutralize the traffic ill effects as for MACSC(PE). The experimental results for the MACSC(F-PE) solution by itself, however, has revealed that its performance is traffic independent. This means that adding the RTPD capability to the MACSC(F-PE) solution provides no advantage. For this reason the possibility of having a RTPD/MACSC(F-PE) solution was not investigated further. The details of the RTPD/MACSC(PE) and other issues will be discussed in chapter 9.

4.3 CONNECTIVE SUMMARY

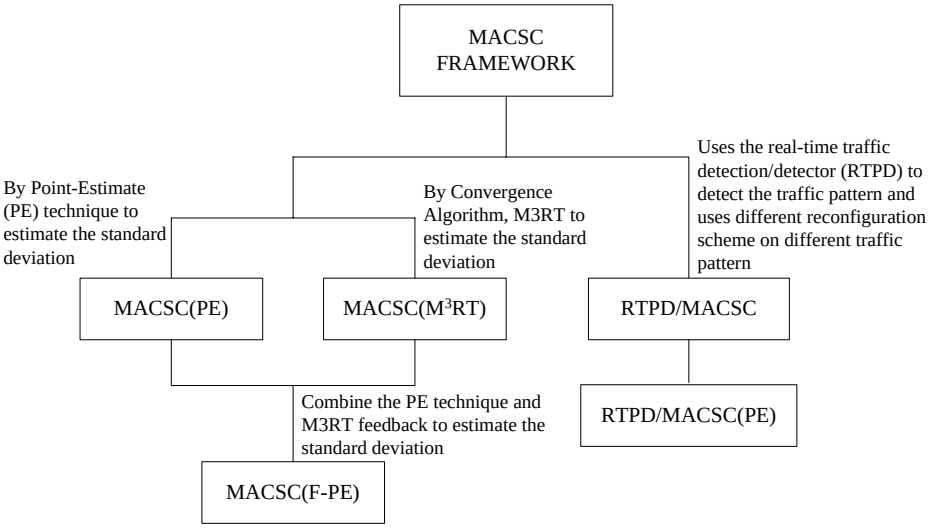


Figure 4.2 Overview of the solutions

This chapter provides the overview of four MACSC solutions: MACSC(PE), MACSC(M^3RT), MACSC(F-PE), and RTPD/MACSC(PE). Figure 4.2 summarizes the relationship among these four solutions, and the details of which will be elaborated in subsequent chapters. The MACSC(F-PE) result indicates that the RTPD/MACSC(F-PE) solution is redundant because the former is already traffic independent.

CHAPTER 5

THE MACSC CONCEPTUAL FRAMEWORK

5.1 INTRODUCTION

In chapter 4, we gave an overview of the MACSC (*Model for Adaptive Cache Size Control*) conceptual framework, which is based on the Zipf-like behavior that reflects the relative popularity data objects in a set. In this framework the dynamic cache size tuning mechanism leverages this relative popularity profile as the sole parameter. The key of leveraging is to compute the current standard deviation of the popularity distribution (PD), which a bell curve obtained by transforming the Zipf-like behavior (as shown by equation (5.1)). The focal issue that follows is how to compute the PD standard deviation correctly and quickly on the fly and per dynamic cache size tuning cycle. Addressing this issue led to the proposals of different solution strategies in this thesis. In this chapter, the discussion is focused on the MACSC conceptual framework. As introduced in chapter 4, the aim of the MACSC framework is dynamic cache size tuning, which maintains the given cache hit ratio adaptively and consistently by adjusting the cache size according to the popularity ratio computed from two successive PD standard deviations. It contributes to shortening the server roundtrip time (RTT) in a client/server interaction. For e-business this is good news because a short RTT keeps customers happy. The ability of a proxy server to maintain the prescribed hit ratio for the local cache reduces its need to access remote data sources (e.g. web servers) for those data

objects requested by clients. This kind of long-haul information retrieval operation can congest the network because of the vast amount of data often needed to be transferred. This inevitably consumes the network bandwidth excessively and leads to sluggish performance. In contrast, dynamic cache size tuning, which keeps more hot data objects in the server's local cache, reduces the need for long-haul data transfer. As a result more backbone network bandwidth is freed for public sharing leading to better network throughput.

The MACSC framework belongs to the area of *caching adaptivity*, which also addresses other issues such as: a) re-configurable caching hierarchies/architectures [Michel98], and b) adaptive models for optimizing cache performance with operation history [Reddy98], [Bolot96]. It focuses on supporting small caching systems of limited memory resources. It is especially suitable for small caching systems of limited memory resources because it strives to maintain the given cache hit ratio as a minimum, without excessive cache memory consumption. These inexpensive systems, which usually cost less than US$1,000, are popular in the field [Wessels01]. For these small systems poor caching would lead to excessive cache memory consumption and poor system performance. This results in frequent task suspensions due to lack of recyclable memory. The MACSC rationale is "*reasonable memory usage to maintain the given cache hit ratio*".

The potential benefits from caching such as those described have inspired different relevant areas of research. The most researched topic so far, however, is how to devise efficacious replacement algorithms. The goal is to push out as many data objects from the cache as possible to make room for hotter newcomers so that a high cache hit ratio can be attained [Aggarwal99]. Replacement algorithms from the

literature, however, work exclusively with a fixed-size cache. As a result they may produce a high cache hit ratio but not necessarily keep it because maintaining a prescribed hit ratio requires dynamic cache size tuning. The cache size has to be timely increased/decreased to accommodate enough hot data objects to satisfy the given hit ratio, without excessively consuming the system memory resources. The only known example that addresses this issue is the MACSC framework [Wong03].

5.2 MACSC

The MACSC mechanism carries out dynamic cache size tuning by leveraging the *relative object popularity* (ROP) of data objects as the sole parameter. Using ROP to produce a high cache hit ratio is a relatively recent concept. For example, it is leveraged as an additional parameter in the "*Popularity-Aware Greedy Dual-Size Web Proxy Caching Algorithms*" [Jin00a], [Cao97], [Young91]. The only framework that leverages ROP as the sole parameter is the MACSC. If the ROP is used as an additional parameter, its potential benefit is easily offset by the long execution time caused by the heavy parameterization of the algorithm.

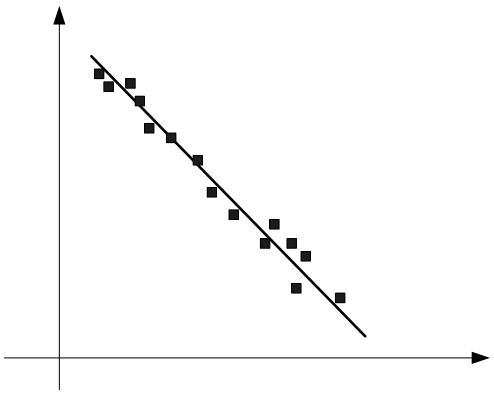


Figure 5.1 Zipf-like distribution (log-log plot)

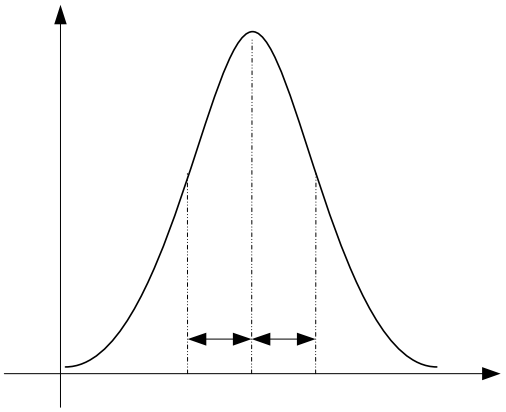


Figure 5.2 Bell shape distribution

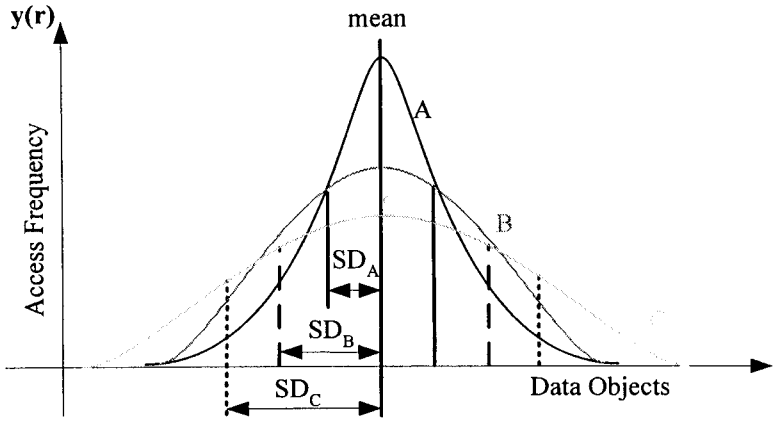


Figure 5.3 PD changes over time and reflects the change in user preference

The MACSC framework is based on the Zipf-like behavior that is intrinsic to large traces of cached data objects [Breslau99], [Zipf]. The behavior is represented by the log-log linear regression plot in Figure 5.1. This plot indicates that the chance for the r^{th} popular object in the *ranked list* (X-axis) to be accessed is proportional to $(1/r)^{-\beta}$, for $0 < \beta \leq 1$. It is the "$y(r)=f_{highest} - y(r-1)$" expression, where γ is a curve fitting parameter, r the ranked position of the object, and $f_{highest}$ the highest access frequency for the "*ranked-first*" object in the data set. The log-log linear regression can be mapped into the bell curve in Figure 5.2 conceptually. The mapping operation is represented by the *bell(r)* equation (5.1).

$$bell(r) = map(y(r)) + e \tag{5.1}$$

The parameter e is the mapping error, which can be ignored if the final dynamic cache size tuning process realized from the mapping process indeed maintains the given hit ratio consistently. Although mapping by equation (5.1) indicates the presence of a bell curve, for implementation purposes the expression in equation (5.2) is used instead. The x variable represents a ranked by "anti-log" data object position. The *bell(x)* curve has a much larger scale relatively than *bell(r)* and thus yields better accuracy.

$$bell(x) = map(y(x)) + e \tag{5.2}$$

The bell curve in figure 5.2 produced by equation (5.1) is called the *popularity distribution* (PD) that represents the changing relative popularity profile of the data objects. The central region of this curve includes the more popular objects, and $f_{highest}$ is the "*mean of the PD distribution*" in the MACSC context [Wong03]. The shape of the PD changes over time due to changes in the user preference towards specific data objects. The changes are immediately reflected by the corresponding PD standard deviation (SD) values. For example, the three curves: A, B and C in Figure 5.3 represent different PD shapes at different time points. The running MACSC mechanism continuously monitors the SD changes and uses them to tune the cache size adaptively. This tuning process strives to maintain the prescribed hit ratio as a minimum, and the *adjusted cache size* (ACS) is computed using either equation (5.3) or equation (5.4). The given cache hit ratio is usually expressed in terms of the number of standard deviations σ. For example, σ=1 means 68.3% and σ =2 for 95.4%.

$$ACS_{SR} = CacheSize_{old} * \left(\frac{SD_{current}}{SD_{last}} \right) \tag{5.3}$$

$$ACS_{VR} = CacheSize_{old} * \left(\frac{SD_{current}}{SD_{last}} \right)^2 \tag{5.4}$$

The popularity ratios for equation (5.3) and equation (5.4) are the *standard deviation ratio* (SR) (i.e. current PD standard deviation over the last or $SD_{current}/SD_{last}$) and the *variance ratio* (VR) (i.e. $(SD_{current}/SD_{last})^2$) respectively.

The following plots show the transformation operation of the MACSC. Figure 5.4 is the pre-collected data trace, EPA-HTTP (From US Environmental Protection

Agency) [SIGCOMM], and Figure 5.5 is the plot of access frequencies versus corresponding *ranked* objects. From Figure 5.5, the log-log plot in Figure 5.6 is produced, and this plot barely exhibits the Zipf-like behavior for β =1.0014. The detail of this will be explained in the section 5.3.2, which discusses deviation behavior compensation. Figure 5.7 is the bell curve transformed from Figure 5.6. The ACS will be calculated based on the information of the Figure 5.7.

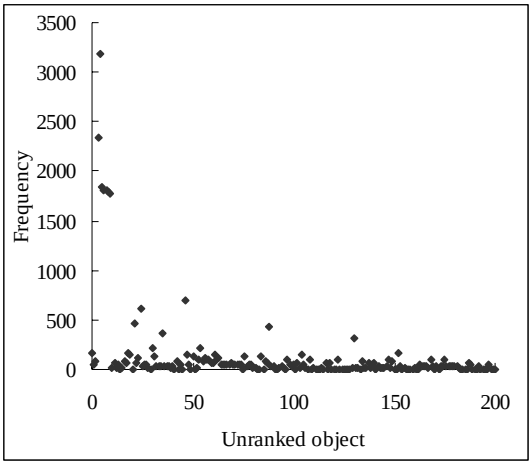


Figure 5.4 The plot of frequency against unranked object ID of EPA-HTTP

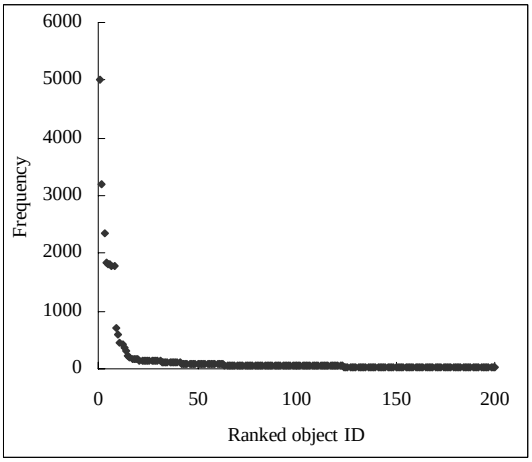


Figure 5.5 The plot of frequency against ranked ID of EPA-HTTP

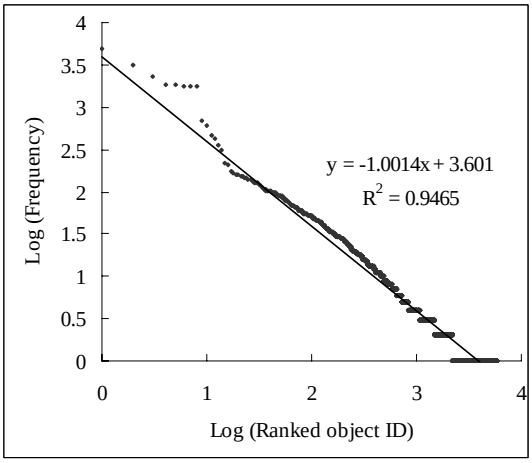


Figure 5.6 The log-log plot of frequency against ranking of EPA-HTTP

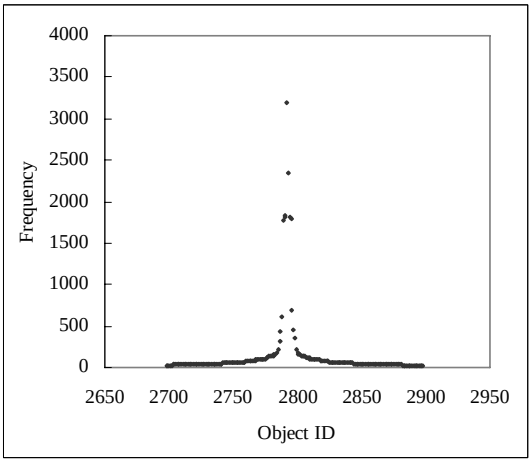


Figure 5.7 Plot of frequency against ranked ID in bell-shape of EPA-HTTP

5.3 COMPENSATION MEASURES FOR MACSC

5.3.1 INITIALIZATION COMPENSATION

The MACSC efficacy hinges upon the accurate measurement of the PD standard deviation changes. Equation (5.3) stipulates that the cache size should change adaptively with respect to the current (SD_j/SD_{j-1}) ratio. It adjusts the cache size by using two consecutively measured PD standard deviations (e.g. in the $(j-1)^{th}$ and j^{th} cycles). The correct implementation of equation (5.3), however, depends on the accurate estimation of the initial cache size before the MACSC starts. Any initialization error will propagate throughout the whole dynamic cache size tuning process. To eliminate this error propagation problem, equation (5.3) should be changed to the form in equation (5.5) for implementation. The ∇ symbol is the number of standard deviations (i.e. $\nabla = \sigma$) that specify the given hit ratio to be maintained (e.g. $\nabla = 1$ for 68.3% and $\nabla = 2$ for 95.4%). It is actually part of the numerator and the denominator (but they cancel out) in the (SD_j/SD_{j-1}) expression. The $OS_{average}$ value in equation (5.5) is the mean object size that MACSC is dealing with. Before MACSC runs (i.e. at time $t=0$ and $j=1$) the initial cache size $CAS_{SR_{t=0}}^{j=1}$ is set to $CAS_{SR_{t=0}}^{j=1} = 2 * \nabla * SD_{t=0}^{j=1} * OS_{average}$. The $SD_{t=0}^{j=1}$ value is the standard deviation computed from the past performance data. Equation (5.5) stipulates that for $j > 1$ the cache size tuning process should not depend on $SD_{t=0}^{j=1}$ but the current SD_j value measured in the j^{th} cycle. Similarly the implementation of equation (5.4) should take the form of equation (5.6). The MACSC cache size initialization is considered as the 0^{th} cycle or $j=0$. The initial cache size of $ACS_{VR_{t=0}}^{j=0} = 2 * \nabla * SD_{t=0}^{j=0} * OS_{average}$ becomes the "*seed value*" for $j \geq 1$. In the subsequent dynamic cache tuning cycles

this seed value is replaced by $2 * \nabla * SD_j * OS_{average}$ and the factor "2" arises from considering one ∇ on both sides of the mean. This scheme eliminates the propagation of the initialization error due to inaccurate $ACS_{VR_{t=0}}^{j=0}$ estimation. The VR based tuning process is, hence, tied to the latest computed SD_j value for $j \geq 1$.

$$ACS_{SR}^{j} = 2 * \nabla * SD_j * OS_{average} \tag{5.5}$$

$$ACS_{VR}^{j} = 2 * \nabla * (SD_j)^2 * OS_{average} \tag{5.6}$$

5.3.2 DEVIATION BEHAVIOR COMPENSATION

The MACSC core consists of two popularity ratios, namely SR and VR. These ratios conceptually come from the popularity distribution shown in Figure 5.2. Firstly, it has been empirically found that large caching systems exhibit Zipf-like behavior [Breslau99], [Zipf]. This behavior for every system is unique and is characterized by a log-log linear regression of the $(1/r)^{-\beta}$ form of a specific β value in the $0 < \beta \leq 1$ range, where r marks the r^{th} most popular object ranked in descending order (Figure 5.8). For a trace of insufficient number of requests the β value may be outside the $0 < \beta \leq 1$ range. In the Figure 5.8, the β value of the BU-Web-client is equal to 0.9817. However, the β value of the NASA-HTTP-client in figure 5.9 is equal to 1.1. It is because the number of requests contained in the NASA-HTTP-client trace is not large enough that the β value is outside the $0 < \beta \leq 1$ range. This deviation from the formal Zipf-like expectation is the *deviation behavior* in the MACSC context. The *deviation behavior* phenomenon is compensated in two steps. The first is to assess the quality of the log-log linear

regression that produces β (i.e. $y(r) = -\beta * r + C$). The quality is reflected by the *coefficient of determination* or R^2, a higher R^2 value for better quality. The second step is to choose an appropriate threshold Th_{R^2} and reject the linear regression for $Th_{R^2} > \beta$. Rejection means continuing to use the last popularity ratio until the $R^2 > Th_{R^2}$ condition is satisfied.

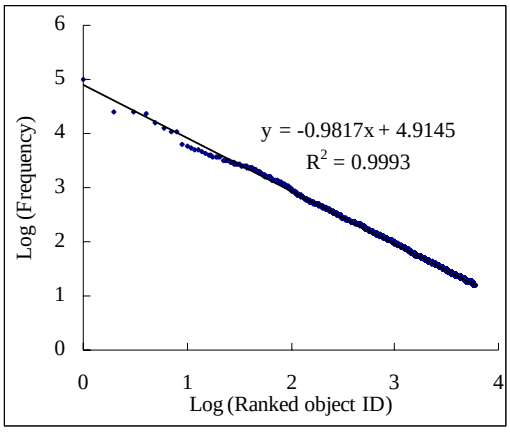


Figure 5.8 The log-log plot of BU-Web-client (169 days)

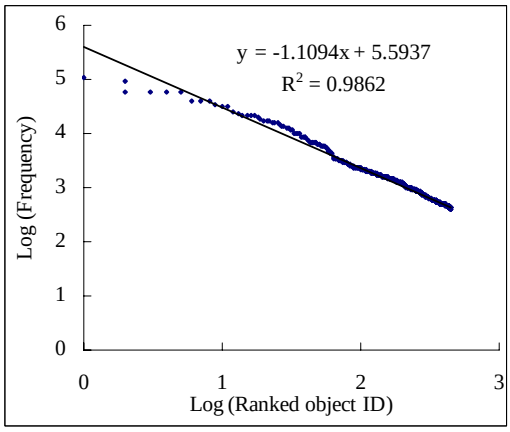


Figure 5.9 The log-log plot of NASA-HTTP-client (31 days)

Both the *bell(r)* and *bell(x)* curves are called the *popularity distributions* (PD), which quantify the same relative data object popularity profile. *bell(r)* is the conceptual popularity distribution and *bell(x)* is its "*anti-log*" version for implementation. Figure 5.1, in effect, depicts the changes of the PD shape over time due to shift of user preferences for particular data objects. The Y-axis of the PD records the access frequencies (or probabilities) for the corresponding objects on the X-axis. The PD variability (spread) is characterized by the standard deviation (SD) that measures the popularity deviations of different data objects from the "*mean value*": $y(1)=f_{highest}$. The object with the highest access frequency in the Zipf-like correlation has virtually become the "*mean value*" of the PD by mapping. Once the SD of the current PD is computed from the live data sampled statistically (e.g. SD_A for curve A in Figure 5.3), the *popularity ratio* (PR) (as shown in equation (5.3) or

(5.4)) can be estimated for tuning the cache size adaptively on the fly to maintain the given cache hit ratio.

It is necessary to empirically verify that *bell(x)* indeed produces the same effect as *bell(r)* for PR computation. In the MACSC context, this verification is called the "*kurtosis and skewness* (KS)" test. In the KS test, the skewness is computed by equation (5.7) and kurtosis is computed by the equation (5.8), where $\bar{x}$ and SD are the statistically measured mean and standard deviation respectively for the aggregate of *m* data items sampled. Skewness measures the symmetry of a bell curve. A positive value indicates that the bell curve embedded in the aggregate skews right (i.e. right tail is heavier than the left). Kurtosis measures whether the bell curve is peaked (positive value) or flat (negative value). In the normal distribution, kurtosis is equal to 3 and skewness is equal to 0.

$$\frac{\sum_{i=1}^{m}(x_i - \bar{x})^3}{(m-1)SD^3} \tag{5.7}$$

$$\frac{\sum_{i=1}^{m}(x_i - \bar{x})^4}{(m-1)SD^4} \tag{5.8}$$

The KS tests indicate that all the traces of data object accesses in our experiments are basically bell curves with reasonable Kurtosis and Skewness values. In the KS tests it also shows that even when the β values of some linear regressions were not in the $0 < \beta \leq 1$ range, the linear regression is still conceptually valid for producing *bell(x)*, which forms the basis for calculating the popularity ratios for dynamic cache size tuning. The condition for validity is having reasonable kurtosis and skewness values as compared to the normal distribution. This KS test will be demonstrated by the experiment result with different traces.

5.4 Connective Summary

In this chapter, the MACSC framework is investigated for dynamic cache size tuning. It maintains the given hit ratio adaptively and consistently by tuning the cache size dynamically according to the popularity ratio of the data object requests. The MACSC framework is conceptually based on the transformation of the Zipf-like curve into the popularity distribution in order to calculate the popularity ratio. In the next chapter, we will discuss the use of the statistical point-estimate (PE) approach to estimate the standard deviation of the data object requests in order to calculate the popularity ratio to tune the cache size on the fly.

CHAPTER 6

THE POINT-ESTEMATE APPROACH

6.1 INTRODUCTION

In the MACSC framework the cache size is adjusted according to the popularity ratio computed for the data objects. The basic concept is that the distribution of the access frequencies against the corresponding data objects at any time represents the real-time relative popularity profile. The popularity-ratio computation accuracy is, however, affected by two factors. The first factor is the speed at which the popularity profile distribution changes its shape. Any shape change, in effect, is a reflection of change of preference for particular data objects by the users. Therefore, any computation method, which caters to a particular distribution/waveform shape such as Poisson, could lead to failure [Paxson95]. The second factor is the inter-arrival times (IAT) among the data object retrieval requests. The IAT distribution affects computation accuracy in two ways. In the first way, if the IAT (in a burst mode) is shorter than the execution time of the dynamic cache size controller, then computation inaccuracy would appear because of incomplete data sampling. In the second way the IAT traffic waveform/pattern over the Internet, which follows the power law [Medina00], changes over time, may change suddenly, for example, from SRD (short-range dependence such as Markovian traffic) to LRD (long- range dependence such as heavy-tailed, self-similar, and multi-fractal). This kind of change can affect the accuracy of the computation method adopted [Paxson95]. To free the

MACSC mechanism from the ill effects of the above two factors, the popularity ratio calculation should be statistical and based on direct measurement. That is, the standard deviation of the relative popularity profile of data objects is freshly and directly measured in each cycle.

The statistical *point-estimate* (PE) approach is used in the first MACSC implementation. The PE computation accuracy is inherently independent of the shape of the waveform (i.e. heavy-tailed, Poisson, self-similar, or multi-fractal) because it is derived from the *Central Limit Theorem*. The computation accuracy, however, could still be affected by the IAT lengths. For example, in a burst mode the IAT may be shorter than the sampling ability of the controller in light of its sampling cycle time. As a result the controller misses data items repeatedly and inadvertently makes wrong calculations that lead to undesirable or deleterious effects. The PE approach provides the minimum capability for the MACSC controller to alleviate the deleterious effects. The PE based MACSC framework is called the MACSC(PE).

6.2 THE MACSC(PE) APPROACH

In the MACSC(PE) approach the cache size is dynamically adjusted according to the popularity ratio, which is calculated with the *point-estimate* (PE) approach. The PE approach is derived from the *Central Limit Theorem*, and the details of which will be provided later. The statistics $\bar{x}$ and s_x, which are the mean and standard deviation values of a data sample of size n for $n > 10$, are called the *point estimates* of the true mean λ and standard deviation δ_x of the population of x. The *Central Limit Theorem* provides the relationship between point estimates and the population/true mean and standard deviation values.

6.2.1 CENTRAL LIMIT THEOREM

This theorem is characterized by the following [Chis92]:

1. The means (i.e. $\bar{x}$) of a series of *x* samples of size *n* from any distribution (Figure 6.1(a)) will form a normal distribution of mean *m* (Figure 6.1(b)). The variability (spread) indicated by $(\delta_{\bar{x}})^2$, where $\delta_{\bar{x}}$ is the standard deviation of this bell curve or standard error plot [Jain91], is smaller than that of the distribution formed by the individual variables themselves; that is, $(\delta_x)^2 > (\delta_{\bar{x}})^2$.

2. The larger the sample size *n* the smaller will be $\delta_{\bar{x}}$, and this implies that *m* will is closer to the true/population mean λ.

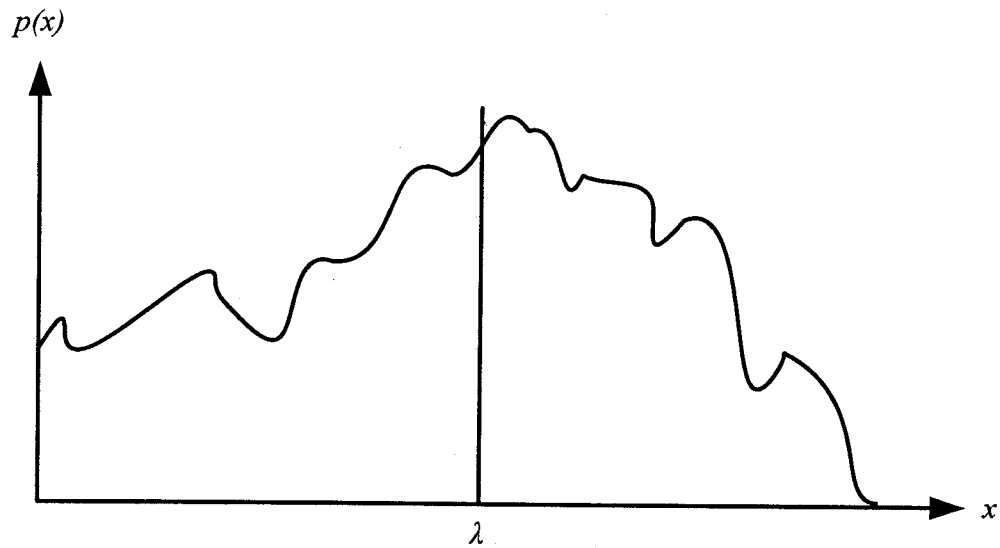
The essence of the *Central Limit Theorem*, which is $\delta_{\bar{x}}^{\,2} = \dfrac{\delta_x^{\,2}}{n}$ for $n > 10$, is captured by Figure 6.1. The figure shows a narrower bell curve for $n = N_2$.

In the estimation of the population mean the concept of a confidence interval is needed, and this implies probabilistic bounds of the following form:

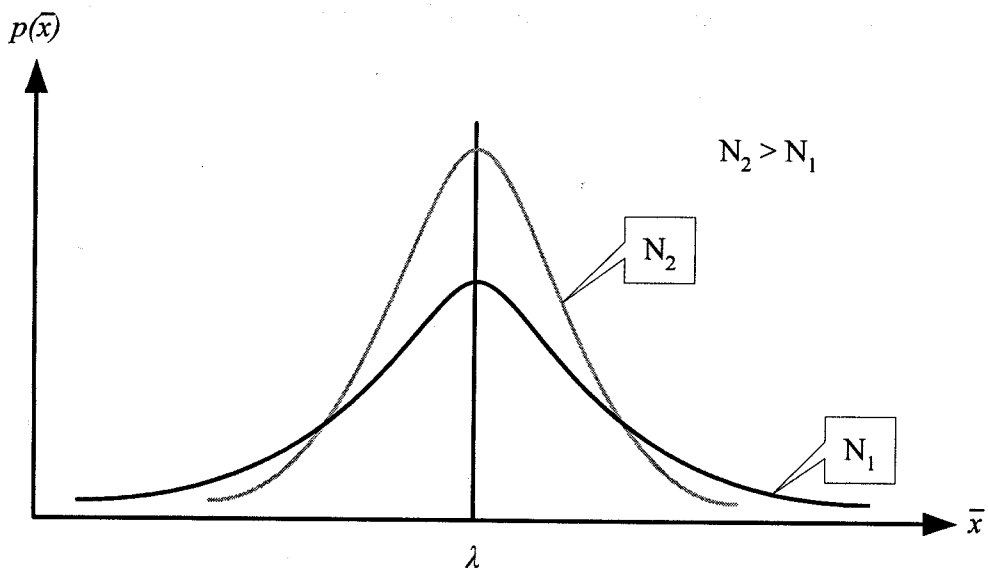
$$\text{Probability } \{c_1 \le \lambda \le c_2\} = 1 - \alpha$$

The (c_1, c_2) is the confidence interval for the population mean and α is the significance level. 100(1-α)% is the confidence level and 1-α is the confidence coefficient. The confidence level is always expressed as a percentage and is close to 100% (e.g. 90% or 95%). Significance level α is set near to zero (e.g. 0.1 or 0.05). In the *Central Limit theorem* applications the 100(1-α)% confidence interval for the population mean is the expression, $\left(x - z_{1-\alpha/2}s/\sqrt{n}\,, x + z_{1-\alpha/2}s/\sqrt{n}\right)$. The $z_{1-\alpha/2}$ value is the *(1-α/2)*-quantile of a unit normal variate. For example, if the mean $\bar{x} = 1.8$,

standard deviation $\delta_{\bar{x}}$ =0.6 and n=20, then the confidence interval will be "$1.8 \mp (1.65)(0.6)/\sqrt{20} = (1.58, 2.02)$", which says that 90% (confidence interval) of the population is included within the range between 1.58 and 2.02. That is, the chance of error allowed is 10%.

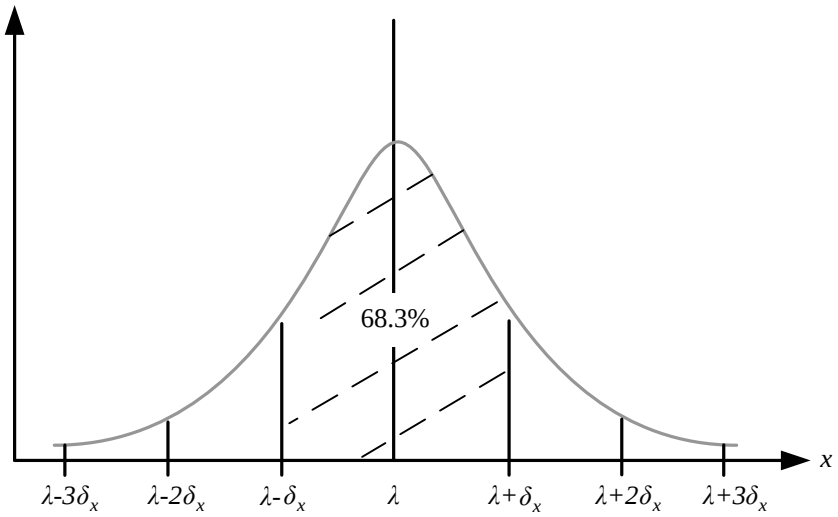


(a) Original distribtion

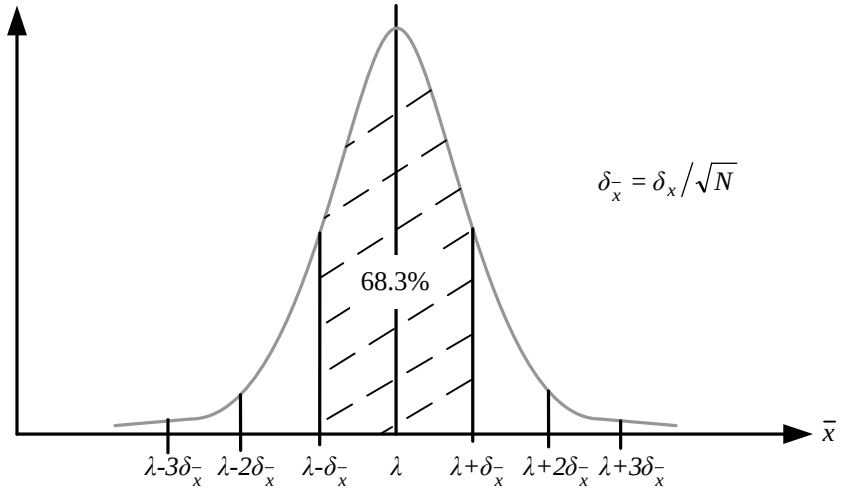


(b) Distribution of sample means from (a)
(Sample size = N)

Figure 6.1 Central Limit Theorem; N_1 and N_2 are two different sample sizes (i.e. n)



(a) Any distribution of (x)



(b) Normal distribution of means ($\bar{x}$) sampled from (a)

Figure 6.2 Normal distribution of means ($\bar{x}$) from any distribution (ideally m = λ)

6.2.2 POINT-ESTIMATE DETAILS

The point-estimate (PE) approach is an estimation technique to estimate the parameters of the x population from a series of data sample of size n for $n > 10$. When the formal expression, $\delta_{\bar{x}}^2 = \frac{\delta_x^2}{n}$ is combined with the following concepts, namely, standard error, significance level, equation (6.1) can be derived. The parameters are defined as follows:

1. *Fractional error tolerance* (*E*): It is the percentage error between the *true mean* λ and *m* (the mean of the bell curve or standard error (SE) plotted/estimated from a series of *sample means* of sample size *n≥10* on the fly).
2. *SD tolerance* (*k*): It is the number of standard deviations (SD) that *m* is away from the *true mean* λ but is still tolerated (*k* same tolerance connotation as E). It is the confidence level about λ.
3. *Predicted standard deviation* ($\delta_{\bar{x}}$) of the SE plot: Theoretically *m* can be estimated from the same series of samples that yield the different $\bar{x}$ values for *n≥10*. By the *Central Limit Theorem* $\delta_{\bar{x}} = \delta_x / \sqrt{n}$ holds, where δ_x is the population/true SD.
4. *Minimum value N*: From equation (6.1) we note that a *minimum sample size N* is needed in practice to estimate the approximate λ and δ_x values that satisfy the given *k* and *E* error tolerances, which, in effect, have the same connotation. The equation (6.1) can be re-arranged to become equation (6.2), which indicates how *N* can be computed. The *N* value, however, is innately unpredictable because it depends on the data profile in the interval of interest. Therefore, it should be found by a process of trial repetitions with progressively larger number of samples until convergence. For this reason, λ and δ_x in equation (6.2) should be replaced by $\bar{x}$ and s_x to become equation (6.3) for implementation purposes. s_x is the standard deviation for the data sample of mean $\bar{x}$. In the repetitive process of *N* estimation, if the first trial has a data sample of size *n*, then every repetition requires more data items. For example, if a fresh sample of size *n* is added in every repetition cycle, then the number of data items used in

the current repetition is (1+*R*)**n*, where R is the number of incremental repetitions for R = 1,2…etc. The repetition stops when the criterion $n \geq N$ is satisfied. In every repetition the PE computes $\bar{x}$ statistically from the n data items first and then $s_x = \sqrt{\frac{\sum_{i=1}^{N}\left(x_i - \bar{x}\right)^2}{N-1}}$, where x_i marks the i^{th} data item in the sample.

$$E\lambda = k\delta_{\bar{x}} = k(\delta_x / \sqrt{N}) \tag{6.1}$$

$$N = (\frac{k\delta_x}{E\lambda})^2 \tag{6.2}$$

$$N = (\frac{ks_x}{E\bar{x}})^2 \tag{6.3}$$

Equation (6.3) is known as the $\sqrt{N} - equation$ in the MACSC context.

The following is an example which shows how the PE process satisfies the $n \geq N$ criterion of the $\sqrt{N} - equation$:

1. It is assumed that the initial 60 data samples (i.e. sample size is $n=60$) have yielded 15 and 9 for $\bar{x}$ and s_x respectively.
2. The given SD tolerance is 2 (i.e. $k=2$ or 95.4%), and the fractional tolerance E is therefore equal to 4.6% (E=0.046); both E and k connote the same error.
3. The N value then should be $N = (\frac{2*9}{0.046*15})^2 \approx 680$.

The value of $N \approx 680$ indicates that the initial sample size $n=60$ is incorrect. To rectify the problem one of the following two methods can be used:

1. The first is to collect (680 – 60) or 620 more data samples and re-calculate $\bar{x}$ and s_x. There is no guarantee, however, the new estimation by this approach

would converge to $n \geq N$. The same process would therefore have to be repeated.

2. The second, which is adopted by the MACSC, is to collect another 60 samples and re-calculate $\bar{x}$ and s_x from the total of 120 samples (i.e. *n=120* for the 2nd trial). The PE is actually the (1+*R*)**n* repetitive process, which stops when the criterion $n \geq N$ is satisfied.

Previous practical experience shows that the second method converges much faster to $n \geq N$. Usually the $\bar{x}$ and s_x values stabilize in the second or third trial [Chis92]. The PE operation in MACSC adopts the second method because previous practical experience consistently shows that the second method converges much faster to $n \geq N$. Usually $\bar{x}$ and s_x stabilize in the second or third trial [Chis92].

In the calculation of the adjusted cache size or ACS, the equation (5.5) becomes equation (6.4) for real MACSC(M^3RT) applications (s_x is the computed standard deviation for j^{th} cycle as explained before).

$$ACS_{SR}^{j} = 2 * \nabla * s_x^j * OS_{average} \qquad (6.4)$$

6.3 MACSC(PE) VERIFICATION

The MACSC(PE) implementation will be discussed in CHAPTER 10. The aim of this section is to show the validity of the PE approach as well as its possible shortcomings. The desire to get rid of the shortcomings leads to new MACSC approaches that will be discussed in the later chapters. In the MACSC(PE) approach, the PE technique is used to estimate the mean and standard deviation of the data objects in each cycle. The MACSC(PE) mechanism cycles through the following:

1. The system collects a number of data objects with the size *n*.

2. It calculates the mean and standard deviation according the sampled data objects.
3. It uses the $\sqrt{N}-equation$ to determine the reasonable sample size N for the system to calculate the acceptable statistical mean and standard deviation.
4. If the sample size n is larger than or equal to N, the system will go to step 5. Otherwise, the system goes back to step 1 to collect more data objects for further calculation based on the $(1+R)*n$ criterion.
5. The system adjusts the cache size according to popularity ratio computed by PE technique.
6. The system goes back to step 1 to collect the additional n new data objects.

Figure 6.3 summarizes the MACSC(PE) operation.

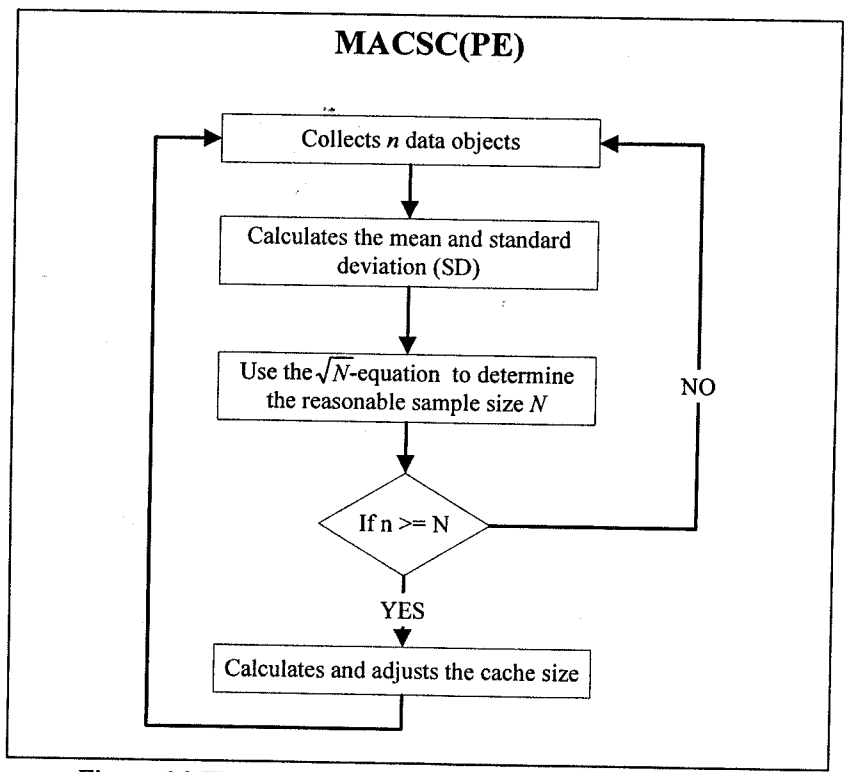


Figure 6.3 The summary flow of the MACSC(PE) approach

6.3.1 SETUP AND ENVIRONMENT

Many simulation experiments were carried out with the MACSC(PE) prototype implemented in Java over the controlled Internet environment. The objectives of the experiments are to verify the following:

a. The MACSC(PE) can indeed determine the popularity ratio of the data objects.
b. It is able to maintain the given hit ratio.

The experimental setup is illustrated in Figure 6.4. In the experiments, the "*fixed cache size* (FCS)" system (i.e. static cache size) was used for the comparison purposes. The simulations were carried out on the Java-based Aglets mobile agent platform [Aridor98], which is chosen for its stability, rich user experience, and scalability. The Aglets platform is designed for Internet applications, and this makes the experimental results scalable and repeatable for the open Internet. The replacement strategy used in the simulations is the basic LRU (*Least Recently Used*) approach with the "*twin cache system* (TCS)" [Aggarawal99]. The TCS has been used successfully in previous investigations of replacement strategies and its main function is to filter the "*one-timers*" that are considered as caching "*noise*". The filtration makes the hot data in the cache more concentrated for more meaningful results. One-timers are unpopular data objects that are accessed only once over a long period. The driver and the MACSC(PE) tuner for the proxy server are *aglets* (*agile applets*) that interact in a client/server relationship. The MACSC(PE) prototype is SR based (equation (5.5)). The cache size is first initialized to meet the prescribed hit ratio in terms of the number (i.e. σ) of standard deviations. For example, if the given hit ratio is one PD (*popularity distribution*) standard deviation

or 68.3% (i.e. $\sigma=1$), the initial cache size in this case is 0.683*40,000*5k bytes (or 136MB), where 5k is the average data object size. If the given hit ratio is 68.3%, the cache size Z should be initialized to $Z \approx Q_{SD} * Sz$, where Sz is the average object size and Q_{SD} equal to the number of objects that represents one standard deviation (i.e. 68.3%).

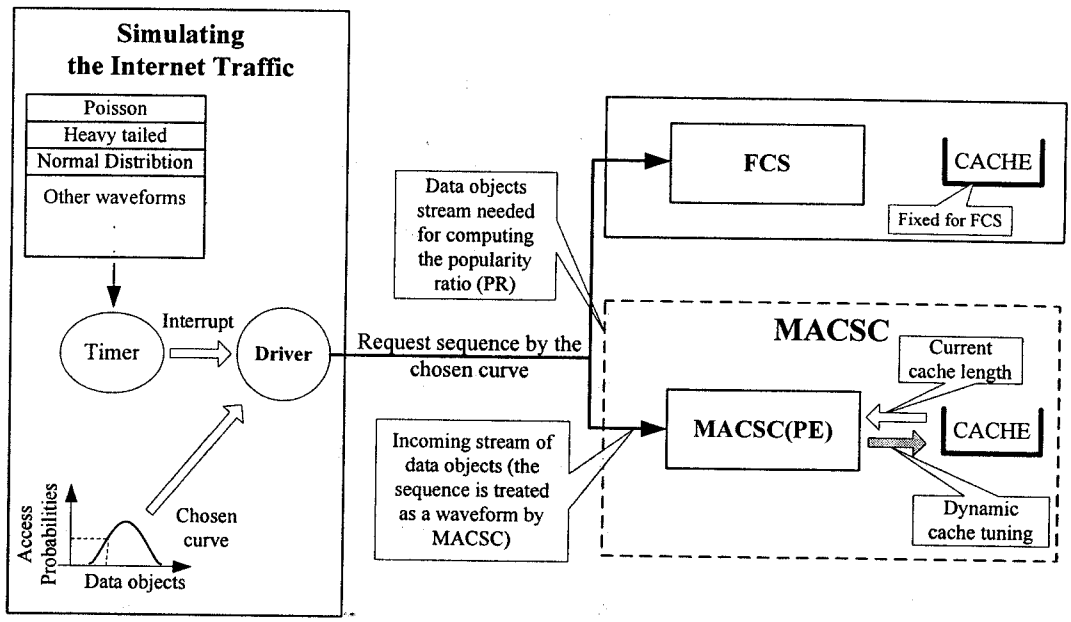


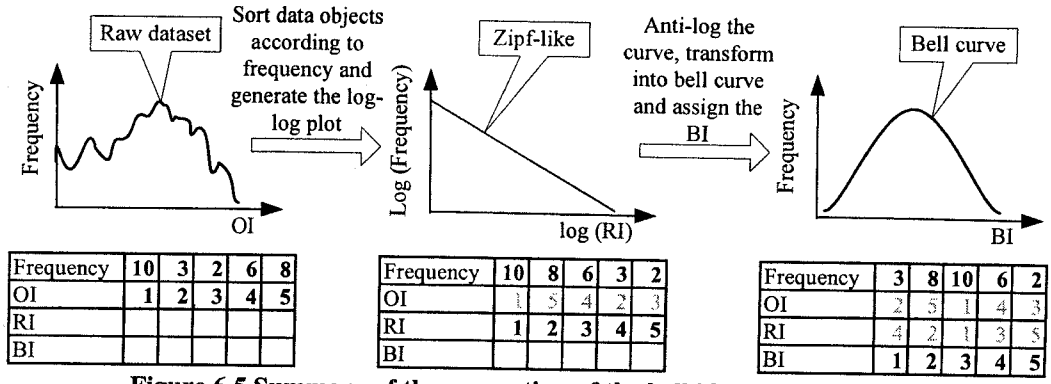
Figure 6.4 Verification set up for the MACSC(PE) tuner (for the proxy) by simulation

In the verification experiments, the driver generates the request traffic profile for the operation. The request traffic can be generated by one of the following methods: a) choosing a known distribution from the table, b) simulating a request pattern by interleaving different bell curves of known standard deviations [Wong03] and c) using a pre-collected real data trace. The simulation results presented in this thesis are produced with the second and third methods. In the first method the chosen curve (in Figure 6.4) mimics the wanted PD distribution. The PD changes over time are mimicked by interleaving different bell curves. The waveform to simulate the IAT among the requests is picked from the table. This drives the

interrupt timer. When an interrupt occurs a random number is generated as the access probability, which is one of the values on the Y axis of the chosen PD curve. On the X-axis of this curve lie the unique integer object identifiers (OI) of the data objects. Each position on the x-axis is also identified by a “bell” identifier (BI), which initially matches the object identifier (OI) in a one-to-one correspondence relationship. The OI position is changed by the ranking process over time as shown in Figure 6.5. These bell identifiers are generated in the beginning of an experiment. The procedures for the experiment are as follows:

1. Each data object in a raw dataset is assigned a unique object identifier (OI).
2. The access frequencies of every data object are counted continuously.
3. The access frequencies of the data objects are sorted in the Zipf-like curve generated by a log-log plot. In this plot, each data object has its own rank identifier (RI) for its ranked position.
4. The Zipf-like curve is transformed into a bell curve called the PD (*popularity distribution*). Then, “bell” identifiers (BI) are assigned.
5. The calculation of the popularity ratio is based on the BI values.

Figure 6.5 shows the summary of the generation of the BI values, and their relationship with RI and OI over time in the continuous operation of the MACSC framework.



Frequency	10	3	2	6	8
OI	1	2	3	4	5
RI					
BI					

Frequency	10	8	6	3	2
OI	1	3	4	2	3
RI	1	2	3	4	5
BI					

Frequency	3	8	10	6	2
OI	2	3	1	4	3
RI	4	2	1	3	5
BI	1	2	3	4	5

Figure 6.5 Summary of the generation of the bell identifier (BI) value

In the MACSC(PE) operation a random number is generated so that the corresponding OI is found by interpolation from the PD and sent as a request to become part of the request traffic (RT). On the server side a concurrent agent called the "*ranker process/task*" is always present (Figure 6.6) to perform two levels of mapping operations. The first level, which is for verification purposes, maps every ranked BI position with the corresponding OI. The OI value, which uniquely marks a data object throughout its service life expectancy, would facilitate the validation experiments later. The second level, which addresses the issue of deployment, maps the OI with the actual object name (e.g. URL). The aim is to facilitate the users in case the actual URL needs to be retrieved. For example, the data object "*OBJ*" may be identified by *OI=10*, but currently it is the 4th relatively hottest object in the data set (i.e. *r=4*). In the MACSC this data object can be referred to as "*OBJ*", *OI=10*, or *r=4* depending on the current phase of the dynamic cache tuning operation.

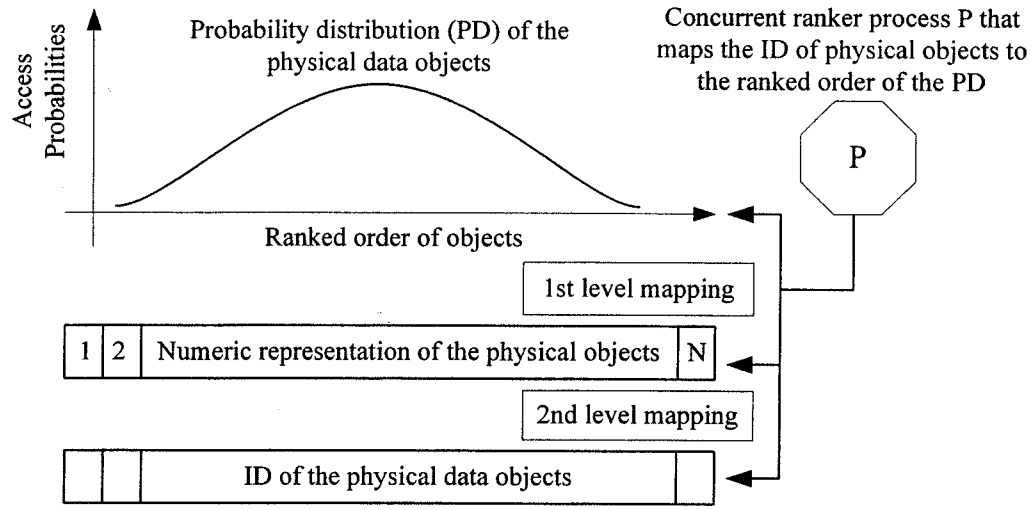


Figure 6.6 A background concurrent "ranker" maps OI with ranked objects

6.3.2 VERIFICATION WITH SIMULATED DATASETS

In the verification experiments with the simulated datasets, a large number of data objects are used. The average size of the data objects is 5k bytes. One million data retrieval transactions are generated by the driver. The simulated datasets are generated by the combining different bell curves with different SD (standard deviation) values. Each dataset has different interleaved SD sequence combination in order to demonstrate the robustness of the dynamic cache size tuning capability of the MACSC(PE) solution clearly. Table 6.1 shows the summary of the simulated datasets that were used in the verification experiments.

Simulated dataset 1	
No. of transactions	200,000
No. of objects	5,000
Average object size	5k byte
Interleaved SD sequence	1k→2.5k→1.5k→2k

Simulated dataset 2	
No. of transactions	400,000
No. of objects	10,000
Average object size	5k byte
Interleaved SD sequence	1k→2k→1.5k→3k

Table 6.1 Summary of the simulated datasets (different interleaved SD sequences)

The simulation results shown in Figure 6.7 were run with the simulated dataset with interleaved SD sequence 1k→2.5k→1.5k→2k. The SD sequence, 1k→2.5k→1.5k→2k was generated by interleaving four bell curves with standard deviations of the 1k, 2.5k, 1.5k and 2k objects, where *k=1000*. The MACSC(PE) maintains the average hit ratio at 52.2% while the "*fixed cache size* (FCS)" system maintains the hit ratio at 39.4% only. The MACSC(PE) tuner yields the highest hit ratio as compared to the FCS. The Figure 6.8 shows that the novel MACSC(PE) tuner produces a 32% higher hit ratio than the FCS. In fact, the MACSC(PE) always yields the highest hit ratio in all the simulations. Figure 6.9 shows the results from another simulation. Similarly, the interleaved SD sequence, 1k→2k→1.5k→3k was generated by interleaving three different bell curves with the standard deviations: 1k, 2k, 1.5k and 3k objects. In this case the MACSC(PE) tuner has 23% higher hit ratio than the FCS (Figure 6.10).

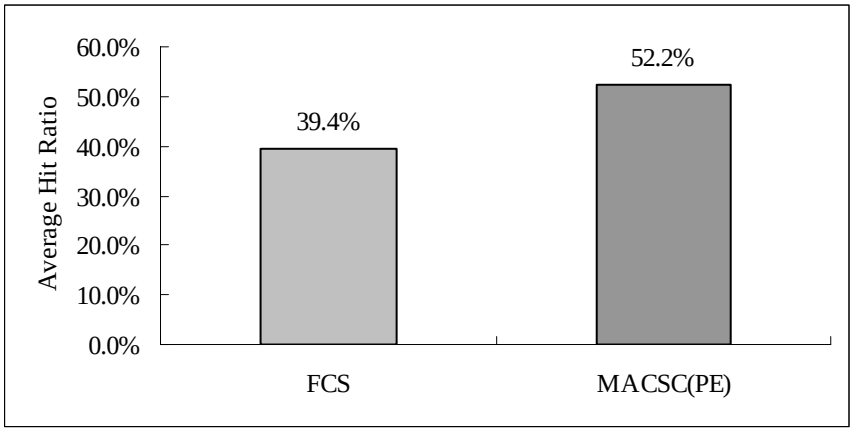


Figure 6.7 The comparison of the hit ratio between MACSC(PE) and FCS (Interleaved SD sequence: 1k→2.5k→1.5k→2k)

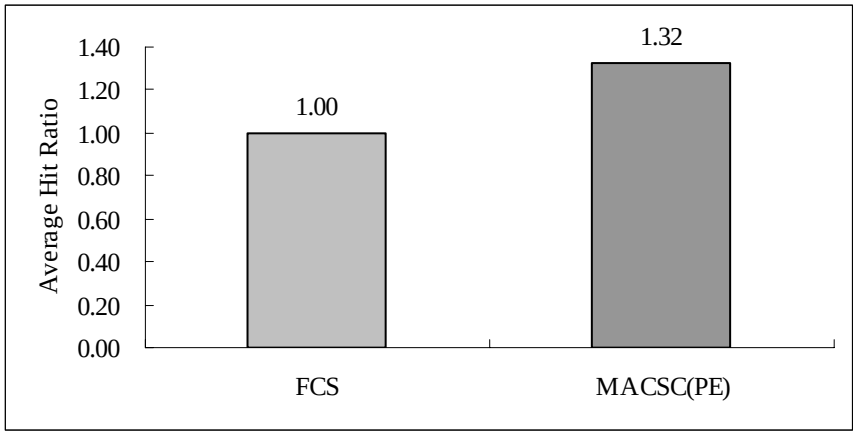


Figure 6.8 The hit ratio improvement over FCS (Interleaved SD sequence: 1k→2.5k→1.5k→2k)

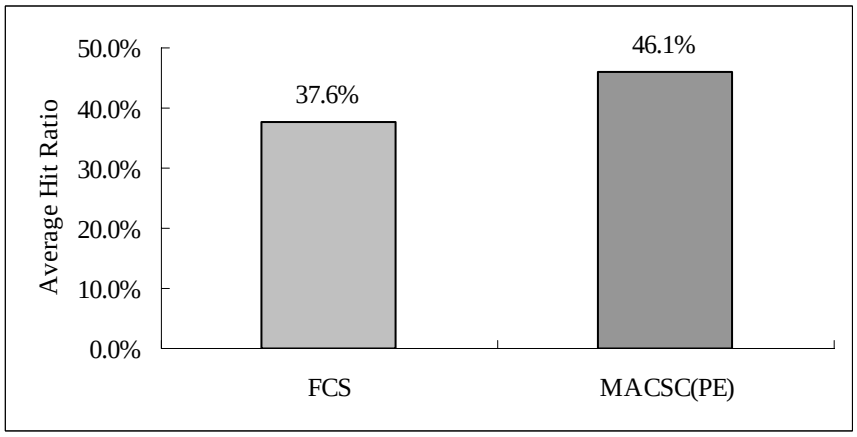


Figure 6.9 The comparison of the hit ratio between MACSC(PE) and FCS (Interleaved SD sequence: 1k→2k→1.5k→3k)

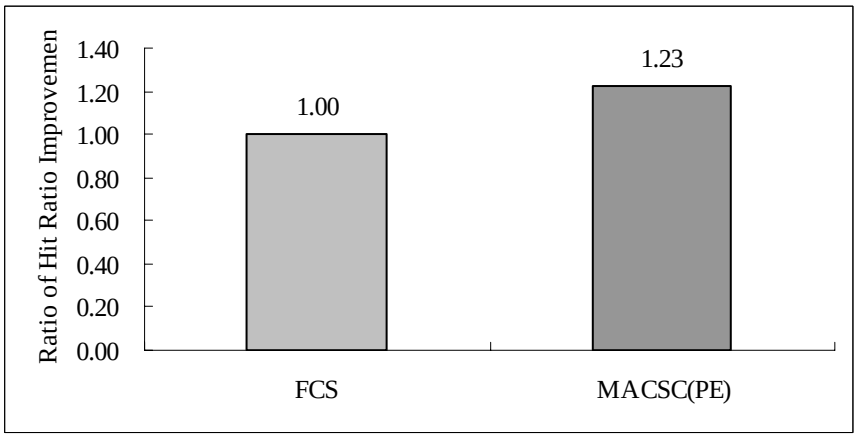


Figure 6.10 The hit ratio improvement over FCS (Interleaved SD sequence: 1k→2k→1.5k→3k)

The Figure 6.11 and Figure 6.12 show the changes of the hit ratio by MACSC(PE) and FCS respectively. In the figures, it can be observed that there is a big impact of the popularity ratio on the performance of the proxy system. When the standard deviation of the data objects rises, the hit ratio of the FCS will drop. The margin of the changes will be in accordance with the change of the standard deviation of the data objects. According to these figures, it can be concluded that the MACSC framework can maintain the given hit ratio by the dynamic cache size tuning. And the PE approach can determine the standard deviation of the data objects successfully for the MACSC(PE) solution.

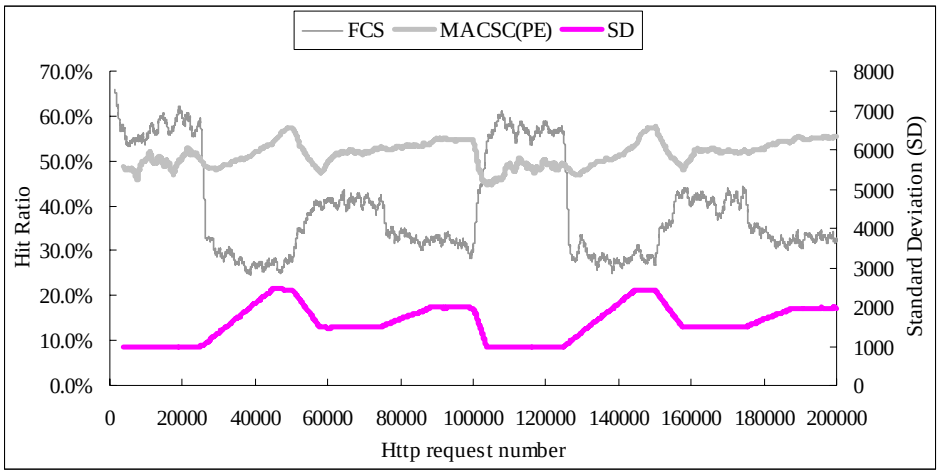


Figure 6.11 Changes of hit ratios by MACSC(PE) and FCS (Interleaved SD sequence: 1k→2.5k→1.5k→2k)

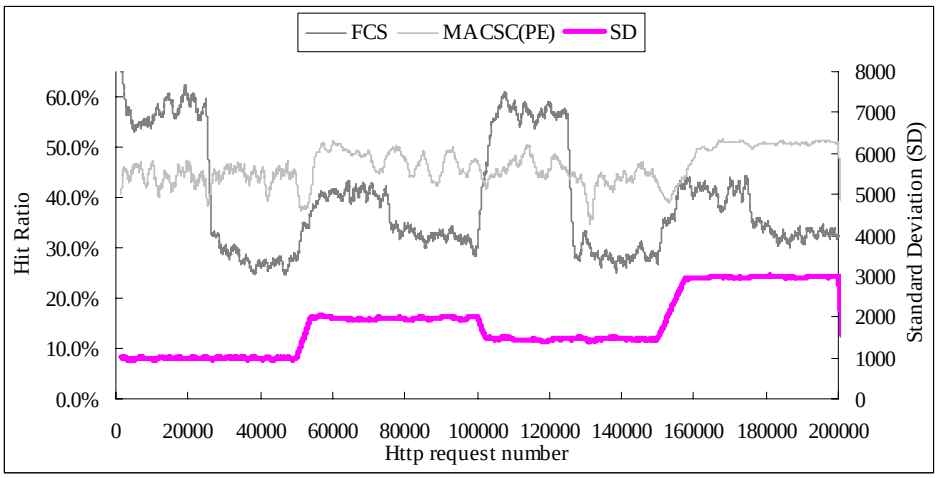


Figure 6.12 Changes of hit ratios by MACSC(PE) and FCS (Interleaved SD sequence: 1k→2k→1.5k→3k)

6.3.3 VERIFICATION WITH PRE-COLLECTED DATA TRACES

In order to verify that the MACSC(PE) indeed has the capability to work in real environments verification experiments were conducted with pre-collected real data traces. The results with the following traces, EPA-HTTP (EPA WWW server located at Research Triangle Park, NC, USA), SDSC-HTTP (San Diego Supercomputer Center), and Calgary-HTTP (University of Calgary, Alberta Canada) [SIGCOMM] are shown here for demonstration. Table 6.2 is the summary of these three data traces. The experiment setup is similar to that for the simulated datasets (Figure 6.4) except that pre-collected data traces were used in this case.

EPA-HTTP	
No. of transactions	42,438
No. of objects	5,584
Duration	24 hours

SDSC-HTTP	
No. of transactions	28,338
No. of objects	1,661
Duration	24 hours

Calgary-HTTP	
No. of transactions	722,982
No. of objects	11,799
Duration	353 days

Table 6.2 Summary of the three pre-collected real data traces

Figure 6.13 compares the experimental results for the FCS and MACSC(PE) with different pre-collected data traces. It shows that the MACSC(PE) can consistently maintain the cache hit ratios at the given 68.3% (one standard deviation) required. In fact, all the experiments conducted so far indicate that the MACSC(PE) performs better than the FCS. Figure 6.14 shows the change of the hit ratio of FCS and MACSC(PE) in the simulation with the EPA-HTTP. It shows that the popularity

ratio of the data objects in real situations indeed varies with time. The MACSC(PE) can maintain the hit ratio much better than the FCS. Figure 6.15 and Figure 6.16 also shows similar results.

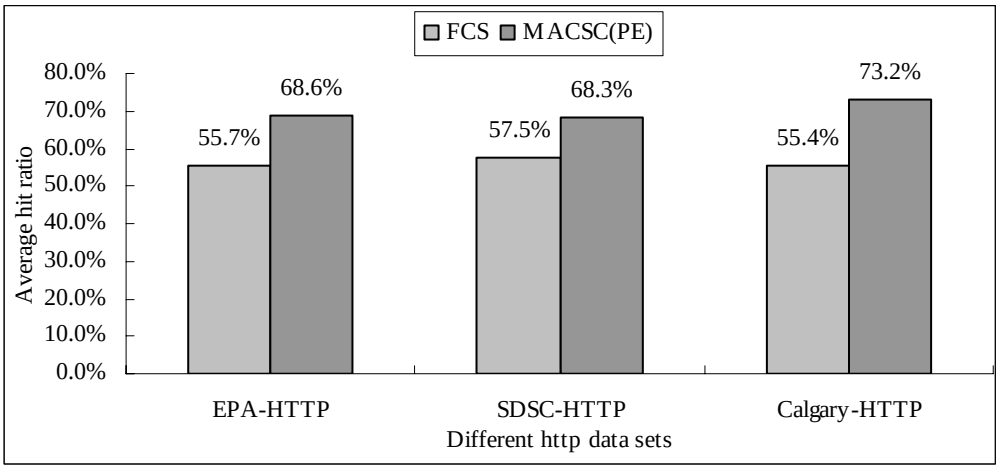


Figure 6.13 Performance comparison of FCS and MACSC(PE) for different data traces

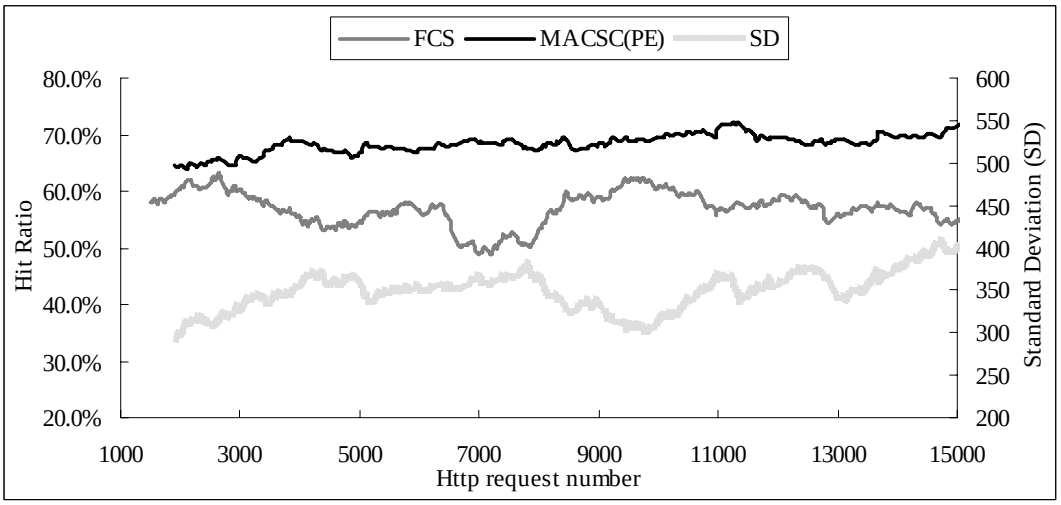


Figure 6.14 The Magnified view of the change of hit ratios by FCS and MACSC(PE) with the EPA-HTTP data trace

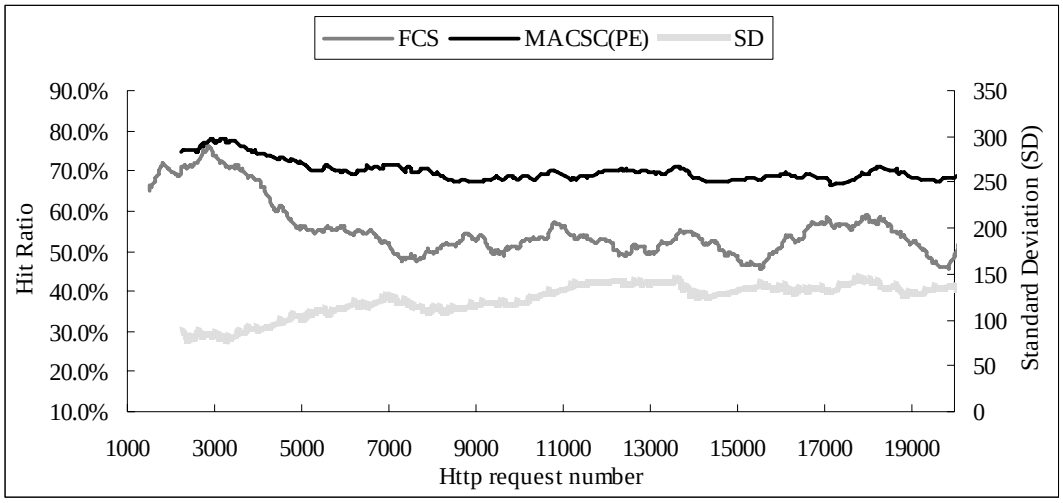


Figure 6.15 The Magnified view of the change of hit ratios by FCS and MACSC(PE) with the SDSC-HTTP data trace

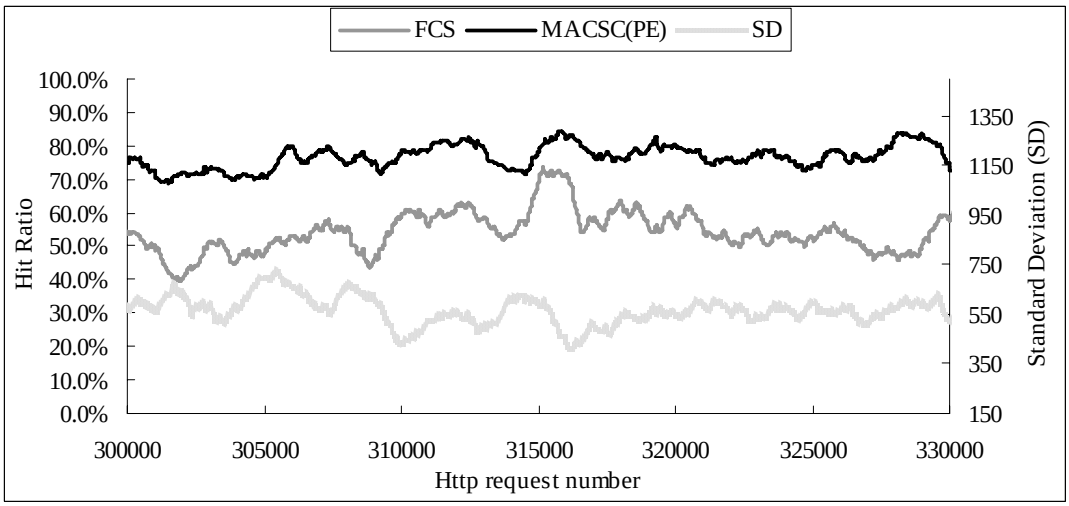


Figure 6.16 The magnified view of changes in hit ratios by FCS and MACSC(PE) with the Calgary-HTTP data trace

6.3.4 SHORTCOMINGS OF THE MACSC(PE)

Although the verification results show that the MACSC(PE) mechanism can indeed maintain the given hit ratio successfully, they have also revealed two PE shortcomings: unpredictable computation time, and serious hit ratio oscillations.

6.3.4.1 UNPREDICTABLE COMPUTATION TIME

It is observed that the requirement to satisfy $n \geq N$ criterion in real-life

applications is unpredictable. This involves the following problems: a) the unpredictable number of data items needed by the statistical PE approach to satisfy the N value of the criterion $n \geq N$, and b) the unpredictable inter-arrival times (IAT) among these items. For example, the Table 6.3 shows the range of the sampling size of the MACSC(PE) with different pre-collected data traces. The collecting of 722850 samples in the aforementioned example may take seconds, hours or even days. The unpredictable sampling number and the IAT interval between any two data items in a sample may reduce the tuning precision of the MACSC.

	EPA-HTTP	**SDSC-HTTP**	**Calgary-HTTP**
Minimum	30	30	30
Maximum	42300	25410	722850
Average	69	69	37

Table 6.3 The range of the sample size of the MACSC(PE) with different pre-collected data traces

6.3.4.2 SERIOUS HIT RATIO OSCILLATIONS

The MACSC(PE) cache hit ratio can seriously oscillate in the steady state because of the large oscillation of the standard deviation of the data objects. This large oscillation make the MACSC(PE) adjustment of the cache size also very large and so reduces the overall performance of the framework. The Figure 6.17 shows a magnified view of the changes of hit ratio by MACSC(PE) with the simulated dataset. It shows that the hit ratio oscillates heavily even when the large trend of the standard deviation doesn't change too much. This leads to the rapid changes of the cache size of the system in Figure 6.18. The result is the decrease in the overall performance of the system. The same situation occurs in the pre-collected data traces. Figure 6.19 and Figure 6.20 show the magnified view of the change of the cache size and memory usage of the MACSC(PE) with the EPA-HTTP dataset respectively.

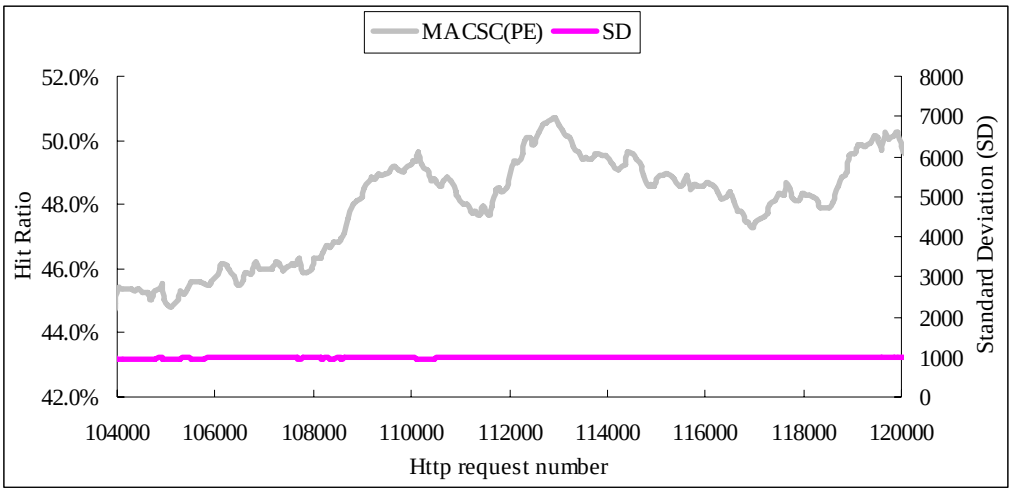


Figure 6.17 The Magnified view of changes of hit ratios by MACSC(PE) (Interleaved SD sequence: 1k→2k→1.5k→3k)

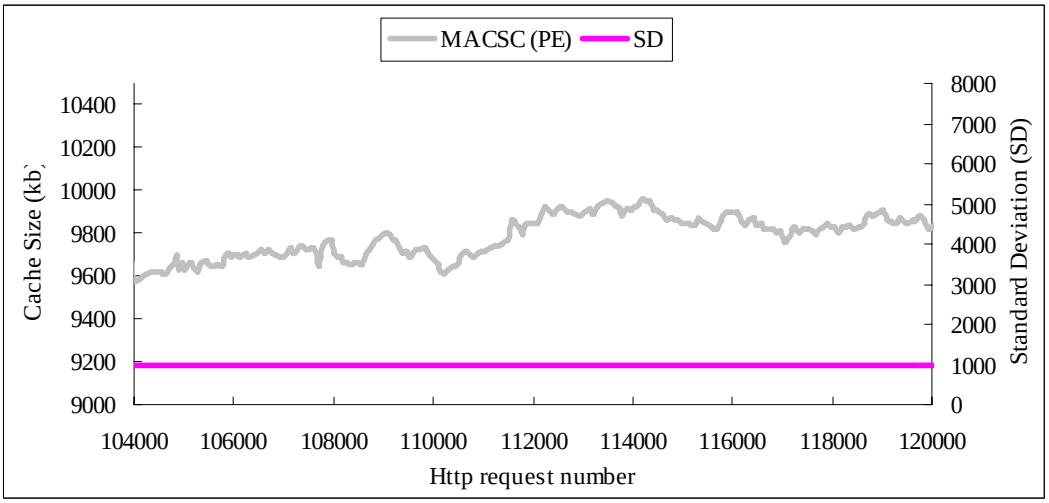


Figure 6.18 The Magnified view of changes in memory usage by MACSC(PE) (Interleaved SD sequence: 1k→2k→1.5k→3k)

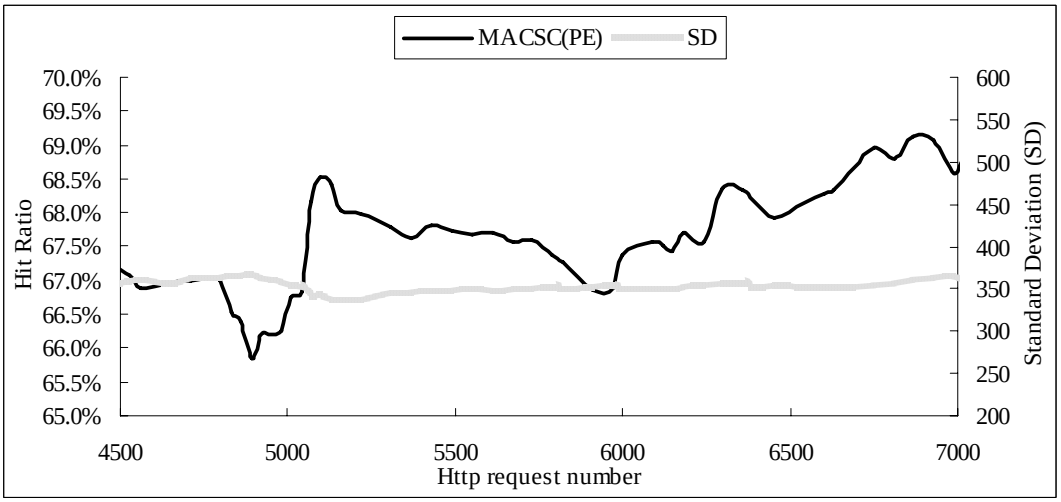


Figure 6.19 The Magnified view of changes of hit ratios by MACSC(PE) with the EPA-HTTP data trace

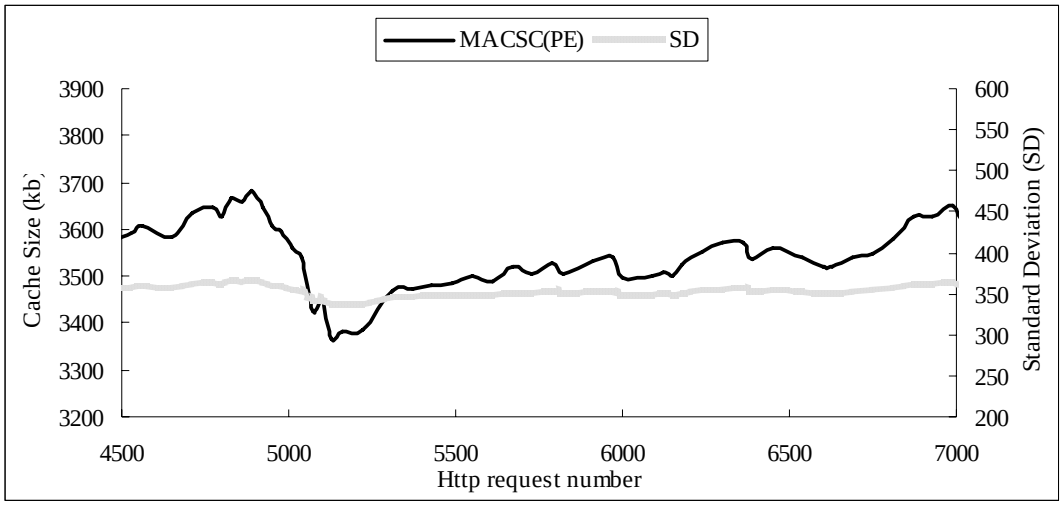


Figure 6.20 The Magnified view of changes in memory usage by MACSC(PE) with the EPA-HTTP data trace

6.4 Connective Summary

In this chapter the theoretical foundation for the MACSC(PE) solution is presented. The core of the statistical computation power for attaining the popularity ratio for dynamic cache size tuning is the *point-estimate* (PE) approach. The verification experiments show that the MACSC(PE) solution can indeed maintain the given cache hit ratio consistently in a dynamic manner. The PE technique, however, has two shortcomings: unpredictable computation time, and serious hit ratio oscillations. For these reasons, an alternative to the PE approach was explored, and this led to the adoption of the M^3RT (*Micro Mean Message Response Time*) technique, which is also based on the *Central Limit Theorem* and therefore its accuracy is waveform/distribution independent. This technique was originally proposed for IEPM (*Internet End-to-End Performance Measurement* [Cottrell99]) applications and the aim is to estimate the mean message response time accurately and quickly on the fly. Since it treats any profile (e.g. data and/or traffic profiles) as a waveform, it has the potential to replace the PE approach. The M^3RT can eliminate the problem of unpredictable computation time because it works with a chosen

number of data items in every prediction cycle. This number is called the *flush limit f* and the optimal range of $9 \leq f \leq 16$ always yields the quickest convergence in IEPM applications. This optimal range, however, may change when the M^{3}RT is transcribed for dynamic cache size tuning, which is a domain that differs in nature from the previous IEPM perspective.

CHAPTER 7

ADAPTION OF CONVERGENCE ALGORITHM

7.1 INTRODUCTION

The point-estimate or PE technique, as observed from the experimental results, has two shortcomings, namely, unpredictable computation time, and possible serious hit ratio oscillations. For these reasons, an alternative to the PE approach has to be explored. This led to the adoption of the Convergence Algorithm (CA), which is a theoretical algorithm for quick and accurate prediction of the mean roundtrip time (RTT) of a communication channel in the interval of interest. The CA was originally proposed for the IEPM (*Internet End-to-End Performance Measurement* [Cottrell99]) applications. It is similar to the PE mechanism in two respects: a) it is derived from the *Central Limit Theorem,* and b) it is suitable for real-time application [Wong02a]. Therefore, the CA could be adapted for supporting the MACSC mechanism in dynamic cache size tuning. The accuracy of prediction/estimation by the CA is waveform/distribution independent. The M^3RT, which is the CA implementation will replace the PE in the MACSC operation. This replacement is used to enhance the MACSC(PE) to the MACSC(M^3RT) version, which is introduced in this chapter.

Similar to its MACSC(PE) predecessor, the MACSC(M^3RT) mechanism consistently maintains the given cache hit ratio. It is, however, difficult to estimate the MACSC(PE) convergence time because the following are unpredictable: a) the number of data samples needed by the PE process to achieve convergence and b) the

inter-arrival times among these data samples. In the MACSC(M^3RT) approach this unpredictability problem is resolved by replacing PE with the M^3RT mechanism, which works with f (i.e. *flush limit*) number of data items sampled on the fly. Previous IEPM experience shows that the region for fast M^3RT convergence is $9 \leq f \leq 16$. The MACSC(M^3RT) approach provides several benefits including: a) it maintains the prescribed hit ratio efficaciously, b) it lessens cache size oscillation, and c) it uses a fixed number of data samples and this makes its computation time more predictable. It is unique because: a) it utilizes the relative popularity of the data objects as the sole control parameter and b) it tunes the cache size adaptively by direct data measurement with the CA support. The relative popularity profile of data objects is called *popularity distribution* (PD) in the MACSC(M^3RT) context. Any change in the PD's standard deviation indicates a shift of user preference for particular data objects. Monitoring and leveraging this change is the basis for MACSC(M^3RT) to find a meaningful *popularity ratio* for deciding how the cache size should be tuned in a dynamic manner.

7.2 THE MACSC(M^3RT) APPROACH

The problem of the *point-estimate* (PE) approach in MACSC(PE) is that it cannot predict the sampling size in each cycle. In order to resolve this problem, the PE mechanism is replaced by the M^3RT, creating the new MACSC(M^3RT) version. The M^3RT works with a fixed number of data items, and this fixed number is called the *flush limit* or simply *f*.

7.2.1 CONVERGE ALGORITHM – M^{3}RT

The Convergence Algorithm (CA) was proposed to predict the *mean message response time* of a communication channel in a sizeable network such as the Internet quickly and accurately [Wong01b]. It treats any type of data distribution as a waveform. In this sense the CA is generic, and it was confirmed that the CA indeed has the capability to predict the mean of any waveform accurately and quickly [Wong01a]. The CA was proposed because calculating the mean of a waveform over time by addition and division can lead to memory overflow. Equation (7.1) shows how the mean, namely, M_i of a distribution of i number of sampled data items can be calculated simply by addition and division. This form has a memory overflow problem for very large i values. For example, one may run into this problem when trying to measure the mean roundtrip time (RTT) of an Internet TCP channel over a long period. The source of memory overflow is the summing operation: $\sum_{j=1}^{i} m_j = \sum_{j=1}^{i} RTT_j$. This problem, however, can be prevented by transforming equation (7.1) to equation (7.2), based on the concept of the *Central Limit Theorem*.

$$M_i = \frac{\sum_{j=1}^{i} m_j}{i}; where\ \ i, j \geq 1 \tag{7.1}$$

$$M_i = \frac{P * M_{i-1} + \sum_{j=1}^{j=f} m_j^i}{P + f}; i > 1, \ \ M_0 = m_{j=0}^{i=1} \tag{7.2}$$

The transformation involves the following parameters:

1. M_i is the mean estimated in the i^{th} prediction cycle from the fixed f (*flush limit*) number of data items (i.e. m_j^i)

2. P is the damping factor to reduce oscillation in the M_i convergence process

3. M_{i-1} is the last estimated mean as feedback to the current prediction cycle.

4. M_0 is the first sample when CA had started running.

The choice of f is important for the M^{3}RT convergence, and it was confirmed that the best f range is $9 \leq f \leq 16$ [Wong01a], [Wong01b], [Ip03] by previous IEPM experience. The M^{3}RT (*Micro Mean Message Response Time*) tool is a *micro* CA implementation in the form of a Java API. It is *micro* because it operates as a logical entity that provides the M_i prediction service anywhere and anytime by message passing. The M^{3}RT always converges to the true mean of any given waveform. Figure 7.1 shows how it converges to M_i mean values of the different IAT (inter-arrival times) segments of size *f=14* extracted from the EPA-HTTP trace [SIGCOMM]. Every data item sampled in this trace is made up of two components, the requested/named data object and the relevant IAT.

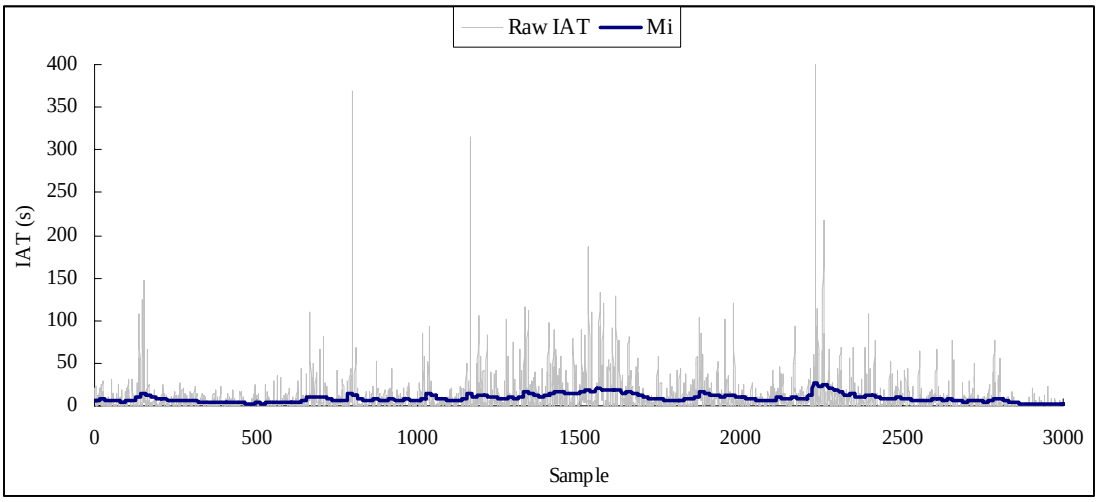


Figure 7.1 M_i calculations by M^{3}RT for the IAT series in the EPA-HTTP trace

7.2.2 DETAIL EXPLAINATION OF MACSC(M^{3}RT)

The PE approach is now replaced by the M^{3}RT *micro* IEPM mechanism, which

uses *f* (*flush limit*) number of data samples to compute s_x and $\bar{x}$. By doing so the following objectives are achieved:

1. *Improved ACS execution time predictability*: Only a fixed *f* number of data samples are used to satisfy the $n \geq N$ criterion.
2. *Independence of traffic patterns*: The M^{3}RT accuracy is independent of the traffic waveform because its *Convergence Algorithm* basis is derived from the *Central Limit Theorem* [Wong01a], [Wong01b].
3. *High tuning accuracy*: The M^{3}RT mechanism helps yield s_x and $\bar{x}$ to satisfy $n \geq N$ quickly and accurately. With pre-collected data traces it needs only 211 clock cycles on average to satisfy $n \geq N$ [Wong03]. This execution time is arisen from the fact that the data items in the trace are immediately usable without any delay. If M^{3}RT has to sample live data on the fly, the actual convergence time depends on the average IAT for the *f* data items in the sample [Wong02b], [Ip03].

The M^{3}RT treats the access frequencies of a collection of data objects as a waveform. Although this waveform represents the relative object popularity profile in the MACSC(M^{3}RT) context, the M^{3}RT treats it in a similar manner to service RTT values [Cottrel99], [Cottrel01], [Wong01a], [Matthews00], [Paxson97], [Ip03]. To summarize, the M^{3}RT implementation has the following salient features that are beneficial to the MACSC(M^{3}RT) approach:

1. Waveform independence: This comes from its *Central Limit Theorem* basis.
2. Predictable execution time: Timing analysis with Intel® VTune™ Performance Analyzer [VTune] confirms that it needs an average of 211 clock cycles for intrinsic convergence when pre-collected data traces are used.

It is intrinsic because the data items in the trace are immediately usable. In on-line applications the actual M^{3}RT execution time depends on the average IAT (inter-arrival time) delay among the data items to be collected on the fly. This makes the actual convergence time much higher than 211 clock cycles. The clock cycles are neutral and can be easily converted into physical time for any platform of interest. For example, if a platform operates at 850 MHz, then the physical time for 211 clock cycles is $T = (1/850 * 10^6) * 211 \approx 0.25 * 10^{-6}$ seconds (i.e. 0.25 microseconds).

3. Ever-increasing accuracy: The CA model is integrative because of the M_{i-1} feedback. As a result, the longer M^{3}RT runs the more accurate its prediction becomes [Wong02b].

The MACSC(M^{3}RT) approach has two parallel components: "*MACSC model without PE support + M^{3}RT*". The total execution latency, T_{MACSC} for the previous MACSC framework is determined by three elements: a) the unpredictable sampling delay T_{Sample} for collecting enough data to satisfy the $n \geq N$ criterion of the $\sqrt{N} - equation$, b) the *point-estimate* computation time, T_{PE} that includes all the iterations until $n \geq N$ is satisfied, and c) the time $T_{PR\&Adjustment}$ for computing PR and carrying out the actual cache size adjustment; that is, $T_{MACSC}=T_{Sample}+T_{PE}+T_{PR\&Adjustment}$. The T_{Sample} delay is unpredictable because it depends on the average IAT of the live samples. The T_{PE} latency is equal to $R*L_{PE}$, where R is the number of iterations, and L_{PE} is the average time for an iteration pass/cycle to compute $(\frac{ks_x}{\bar{x}})^2$. Previous observations show that the R range is $1 \leq R \leq 5$ most of the time. The $T_{PR\&Adjustment}$ latency is fixed because the PR

computation and the physical cache size adjustment involve no unpredictable elements. Working with different pre-collected traces (no actual IAT delays) where samples are immediately usable, the average T_{PE} and $T_{PR\&Adjustment}$ latencies for the MACSC Java prototype were approximated as 712,000 and 43,000 clock cycles respectively. The average T_{Sample} latency is equal to the M^{3}RT execution time of 211 clock cycles. The maximum intrinsic speedup (S) by MACSC(M^{3}RT) compared to MACSC for pre-collected traces is S=(T_{PE}+$T_{PR\&Adjustment}$)/(211+$T_{PR\&Adjustment}$), which is approximately, 712,000/43,211≈16.5 or 1,650%. In real-time operations, however, the M^{3}RT component in the MACSC(M^{3}RT) must sample live items one by one on the fly. The actual unpredictable IAT delays among data samples would make the 1,650% intrinsic speedup difficult to achieve.

Although the M^{3}RT mechanism, which is derived from the *Central Limit Theorem*, is a potential candidate to replace the PE mechanism, it lacks the necessary sensitivity to follow the PD changes closely. On the one hand the M^{3}RT always predicts the mean of a waveform accurately with data collected on the fly, whilst on the other hand the last M_{i-1} feedback to the current M_i prediction (equation (7.2) to yield stability for the convergence process becomes a liability for applying M^{3}RT directly to the MACSC(M^{3}RT) approach. The liability is that the M^{3}RT may not be responsive enough for accurate popularity ratio calculation to reflect the actual current situation. To resolve this liability problem equation (7.2) is transformed through equation (7.3) to become equation (7.4). By arranging $\alpha = \frac{p}{p+f}$ and $(1-\alpha) = \frac{f}{p+f}$ the new equation (7.5) is obtained for the actual M^{3}RT based operation.

$$M_i = \frac{p}{p+f} * M_{i-1} + \frac{1}{p+f}\left(\sum_{j=1}^{j=f} m_i^j\right) \tag{7.3}$$

$$M_i = \frac{p}{p+f} * M_{i-1} + \frac{1}{p+f}(f)\left(\frac{1}{f}\right)\left(\sum_{j=1}^{j=f} m_i^j\right); \tag{7.4}$$

$$\because \frac{p}{p+f} + \frac{f}{p+f} = 1$$

$$M_i = \alpha * M_{i-1} + (1-\alpha) * \left(\frac{\sum_{j=1}^{j=f} m_i^j}{f}\right) \tag{7.5}$$

By adopting the same integrative principle of the M_i prediction, the $\delta_{\bar{x}}$ computation has changed to φ_i estimation by equation (7.6). Then, equation (5.5) becomes equation (7.7) for real MACSC(M^{3}RT) applications, where α and β weight the feedback value (i.e. history) in the M_i and φ_i computation respectively.

$$\varphi_i = \beta * \varphi_{i-1} + (1-\beta) * \sqrt{\frac{\sum_{j=1}^{j=f}\left(m_i^j - M_i\right)^2}{f}} \tag{7.6}$$

$$ACS_{SR}^{i} = 2 * \nabla * \varphi_i * OS_{average} \tag{7.7}$$

7.3 MACSC(M^{3}RT) VERIFICATION

In the MACSC(M^{3}RT), the framework use the converge algorithm (equation (7.5) and (7.6)) to calculate the mean M_i and standard deviation φ_i in *i* cycle. The values will then feedback as M_{i-1} and φ_{i-1} for the next cycle computation. The α and β are the weight values for the ratio of the history and current value. The MACSC(M^{3}RT) mechanism cycles through the following steps:

1. The system collects f data objects
2. It calculates the M_i by equation (7.5) and φ_i by equation (7.6) with the feedback M_{i-1} and φ_{i-1} with weight α and β respectively.
3. The system adjusts the cache size according to the popularity ratio computed by equation (7.7) and feedback the values as M_{i-1} and φ_{i-1} for the next cycle computation.
4. Go back to step 1 for next cycle.

Figure 7.2 summarizes the MACSC(M^3RT) operation.

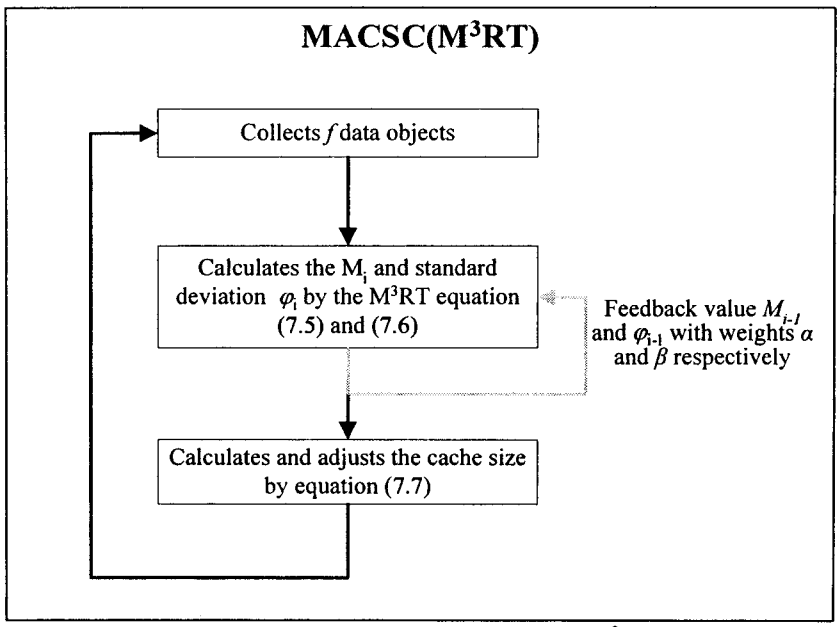


Figure 7.2 The summary flow of the MACSC(M^3RT) approach

7.3.1 SETUP AND ENVIRONMENT

Many simulation experiments were carried out to verify the MACSC(M^3RT) approach and compare it with MACSC(PE) and FCS setups. The objective is to confirm that M^3RT based MACSC(M^3RT) approach is indeed stable and more efficient than its PE based MACSC predecessor, MACSC(PE). The experimental setup is illustrated in Figure 7.3, and the experiments were conducted in a similar

environment to the previous MACSC(PE) ones. The MACSC(M^3RT) experiments are separated into two different parts: a) verification with simulated datasets and b) verification with pre-collected real-life data traces.

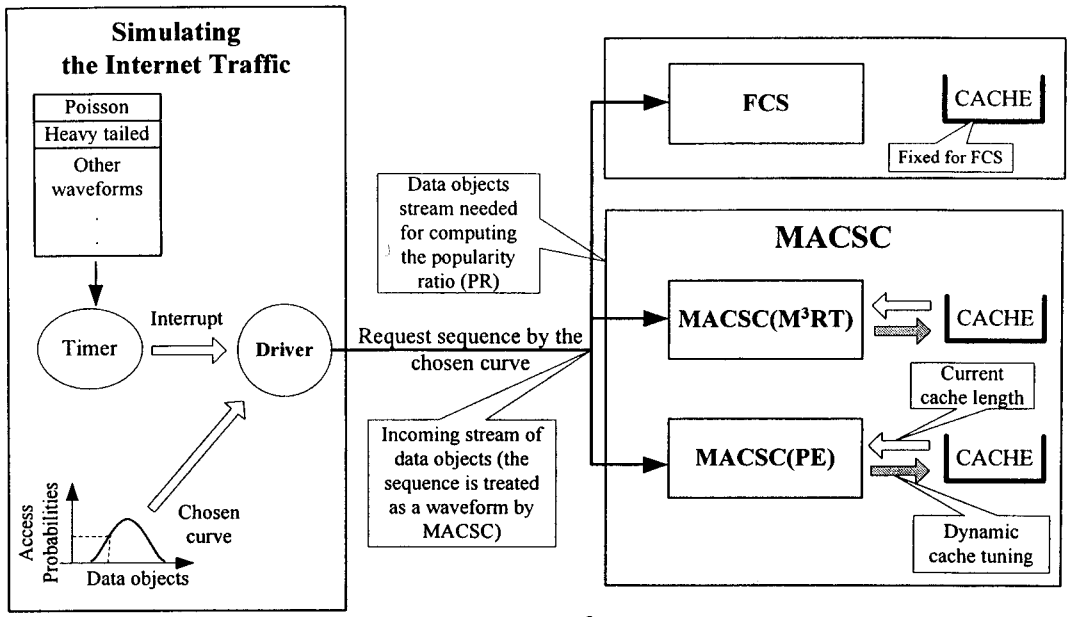


Figure 7.3 Verification set up for the MACSC(M^3RT) tuner (for the proxy) by simulation

7.3.2 VERIFICATION WITH SIMULATED DATASETS

In these experiments, a large number of simulated data objects are used. The same simulated datasets that are used in the previous experiments (Table 6.1). The cache size is first initialized to meet the prescribed hit ratio of one standard deviation. The MACSC(M^3RT) parameter flush limit f and weight parameter α are set to 19 and 0.999 respectively. Figure 7.4 (Interleaved SD sequence, 1k→2.5k→1.5k→2k) shows that the MACSC(M^3RT) maintains the given hit ratio better than the MACSC(PE) and FCS. Figure 7.5 shows the hit ratio improvement of different frameworks over FCS. Figure 7.6 and Figure 7.7 show the results of another simulation (Interleaved SD sequence, 1k→2k→1.5k→3k). It also shows that the MACSC(M^3RT) maintains the given hit ratio better than MACSC(PE) and FCS.

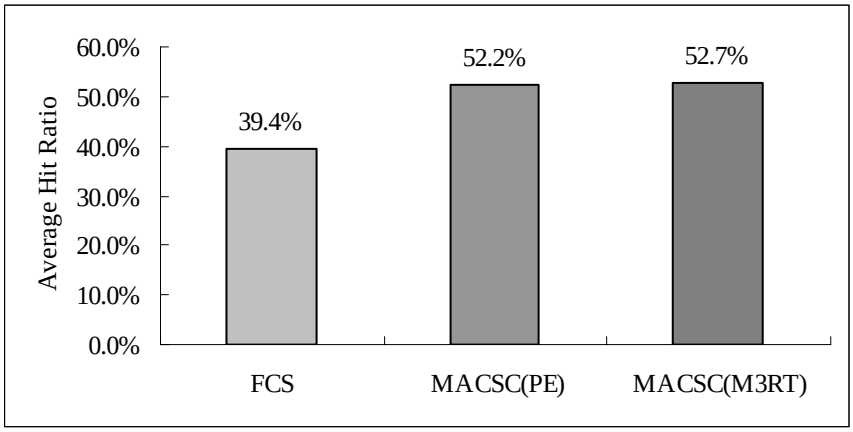


Figure 7.4 The comparison of the hit ratio between different frameworks (Interleaved SD sequence: 1k→2.5k→1.5k→2k with f=19 and α=0.999)

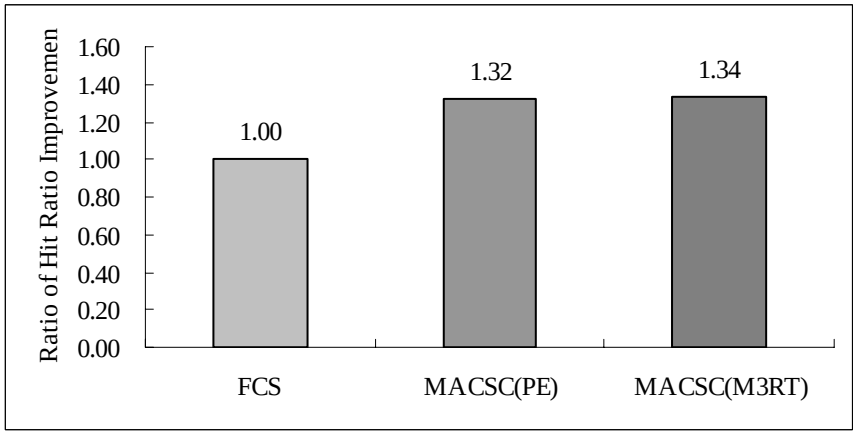


Figure 7.5 The hit ratio improvement over FCS (Interleaved SD sequence: 1k→2.5k→1.5k→2k with f=19 and α=0.999)

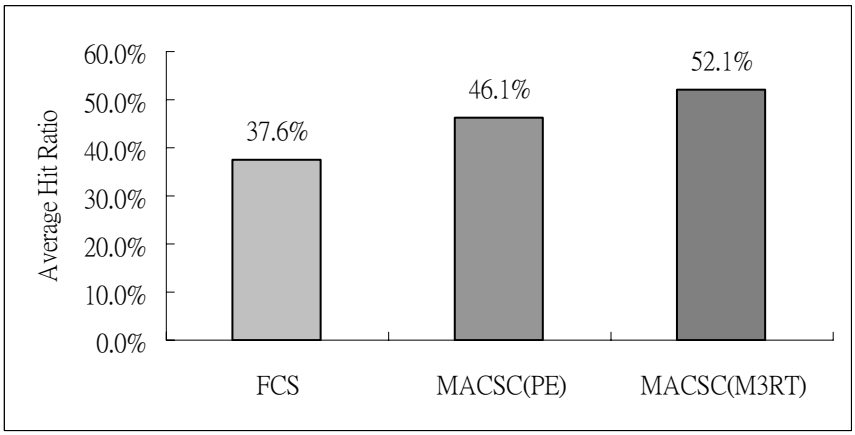


Figure 7.6 The comparison of the hit ratio between different frameworks (Interleaved SD sequence: 1k→2k→1.5k→3k with f=19 and α=0.999)

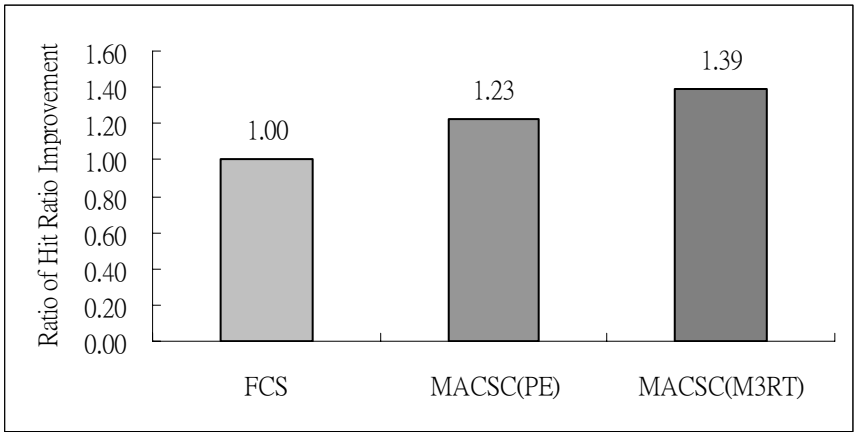


Figure 7.7 The hit ratio improvement over FCS
(Interleaved SD sequence: 1k→2k→1.5k→3k)

The Figure 7.8 and Figure 7.9 show the changes of the hit ratio by FCS, MACSC(PE) and MACSC(M^3RT) respectively. In the figures, it can be observed that the M^3RT approach is better than the PE approach because the hit ratio of the MACSC(M^3RT) oscillates less than the MACSC(PE). The MACSC(M^3RT) can maintain the hit ratio more stable throughout the experiments while the MACSC(PE) drops a little bit sometime. The feedback system of the M^3RT reduces the impact of the small oscillations of the data objects and so it makes the MACSC(M^3RT) more stable compared with the MACSC(PE).

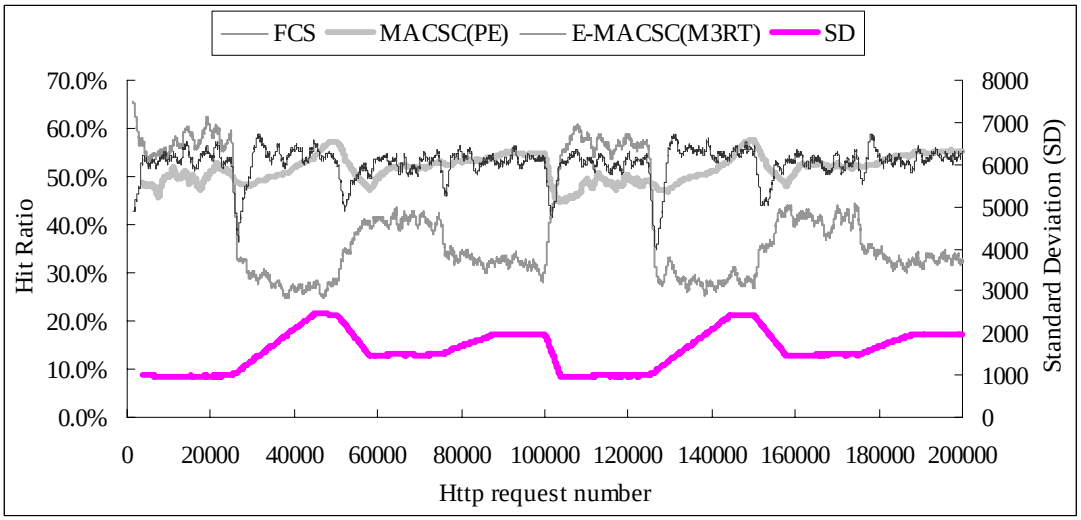


Figure 7.8 Changes of hit ratios by different frameworks
(Interleaved SD sequence: 1k→2.5k→1.5k→2k)

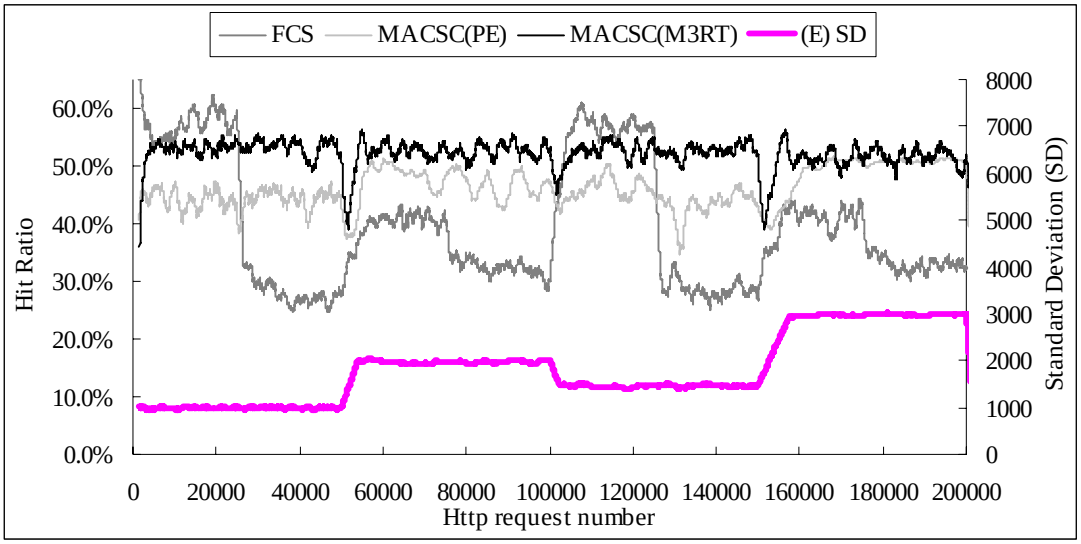


Figure 7.9 Changes of hit ratios by different frameworks (Interleaved SD sequence: 1k→2k→1.5k→3k)

To confirm that the PE approach indeed needs more data samples to be collected on the fly to satisfy the $E\lambda = k\delta_{\bar{x}}$ criterion than the M^{3}RT mechanism, some of the above MACSC(M^{3}RT) and MACSC(PE) simulations were repeated under the same conditions. The average number of data samples needed by each tuner from accumulated experience is listed in Table 7.1. Consistently, the MACSC(PE) tuner needs an average of 155 data samples to attain $n \geq N$ convergence, but the MACSC(M^{3}RT) tuner needs only 19 on average. That is, the MACSC(PE) uses (155/19) ≈ 8.15 times more samples on average. But, the intrinsic computation overhead of MACSC(PE) is 16 times higher than the MACSC(M^{3}RT), (0.96ms/0.06ms) = 16. The timing analysis was carried out with the Intel® VTune™ Performance Analyzer [VTune] on the platform operating at the speed of 1.5 GHz (G for Giga). The result indicates that if the average IAT is getting shorter (e.g. IAT→0), the speedup can get up to 16 times (shown previously). Yet, this is difficult to achieve in real-life MACSC(M^{3}RT) applications because the data items have to be

sampled one by one on the fly. Sometimes the IAT delay between two samples can be significant.

	Range of data sampling	**Average number of data sampling per cycle**	**Physical computation time in each cycle on the platform that operates at 1.5GHz**
MACSC(PE)	30 ~ 450 (To satisfy $n \geq N$)	155	0.96 ms
MACSC(M^3RT)	Any choice from the range: 16 ~ 20 (f value)	19	0.06 ms

Table 7.1 Comparing of MACSC(PE) and MACSC(M^3RT)

Figure 7.10 shows the impact of different α values (i.e. equation (4.3) an (4.4)) when working with f = 19 to produce fast $n \geq N$ convergence. The RT cyclical sequence and the given hit ratio are: 2k→6k→4k and 68.3% (one standard deviation about the "f_{highest} *mean*") respectively. The MACSC(M^3RT) always maintains the prescribed hit ratio consistently for $\alpha \leq 0.999$. For any α value larger than 0.999, the hit ratio drops steeply together with memory consumption. The cause is the sudden loss of PR sensitivity because the emphasis is now on the past performance represented by α rather the current changes, namely, the (1 - α) factor as shown in equation (4.4). Figure 7.11 shows the impact of different *flush limits* on the hit ratio with $\alpha \leq 0.999$. The *flush limit* range that yields the highest hit ratio had shifted to the new range $17 \leq f \leq 22$ from the original $9 \leq f \leq 16$ for M_i prediction by M^3RT [Wu04]. This shift is caused by the integrative property of the φ_i component in equation (4.4).

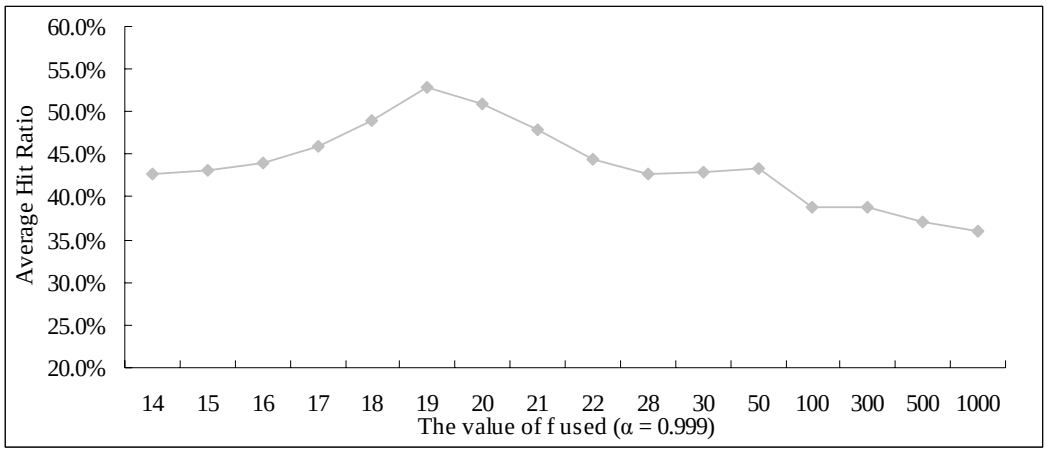


Figure 7.10 Correlation among hit ratio, cache size and α with f=19

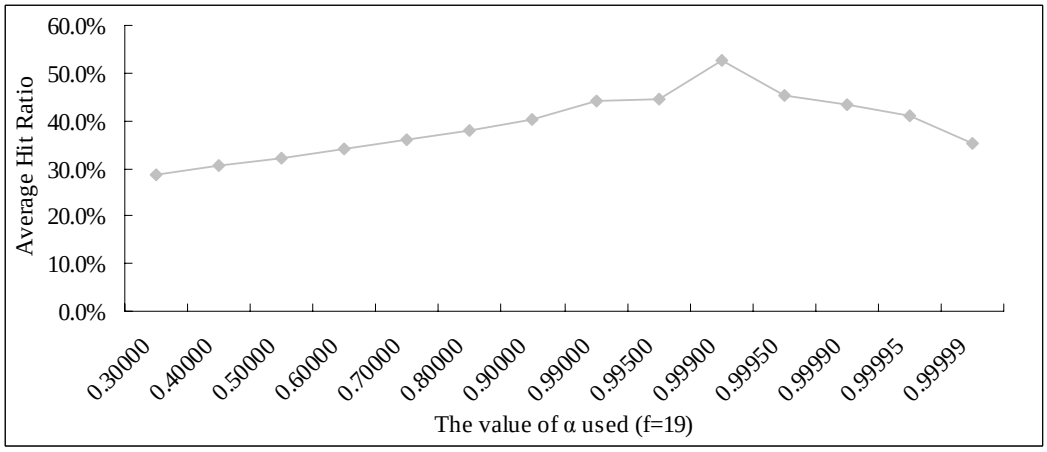


Figure 7.11 Correlation among hit ratio, cache size and f values with α= 0.999

7.3.3 VERIFICATION WITH PRE-COLLECTED DATA TRACES

In these simulation experiments, different pre-collected data traces are used to verify the MACSC(M^3RT) in real environments. The same pre-collected data traces in previous experiments are used (Table 6.2). Figure 7.12 compares the experimental results for the FCS, MACSC(PE) and MACSC(M^3RT) with different pre-collected data traces. It shows that the MACSC(M^3RT) can consistently maintain a better hit ratio compare with the MACSC(PE) and FCS. Figure 7.13 shows the change of the hit ratio of different frameworks in the simulation with the EPA-HTTP. It shows that the MACSC(M^3RT) can maintain a better cache hit ratio than MACSC(PE). Figure 7.14 and figure 7.15 also show similar results.

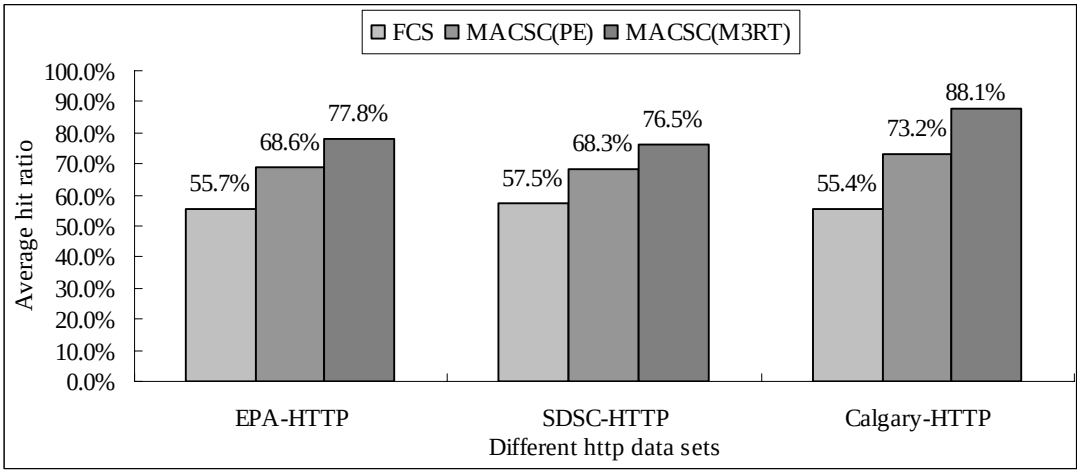


Figure 7.12 Performance comparison of different frameworks for different data traces

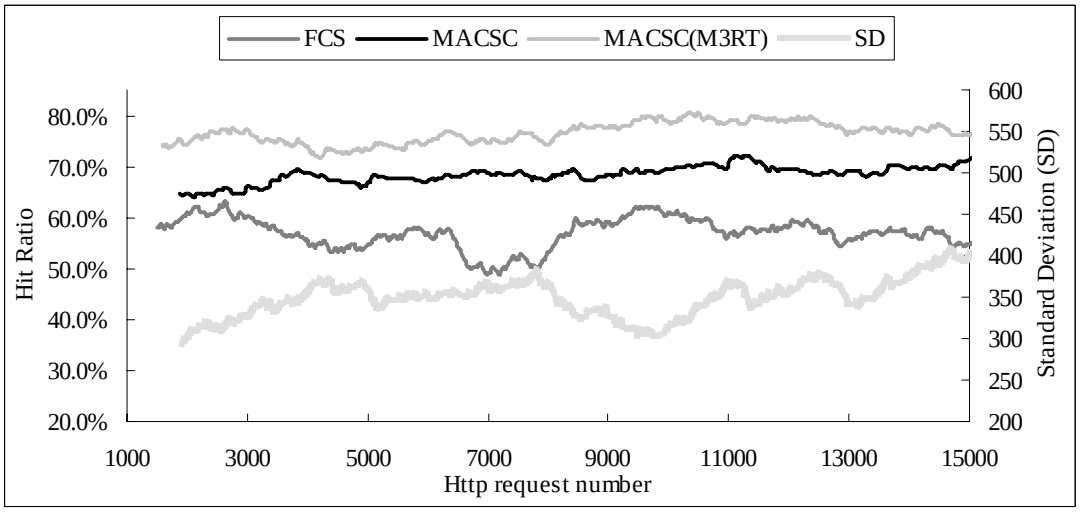


Figure 7.13 The Magnified view of the change of hit ratios by different frameworks with the EPA-HTTP data trace

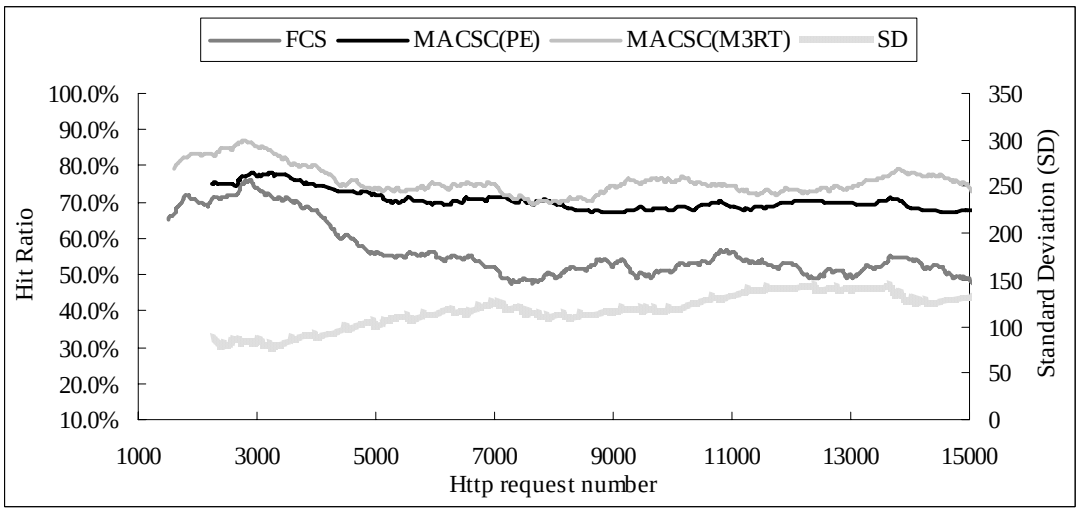


Figure 7.14 The Magnified view of the change of hit ratios by different frameworks with the SDSC-HTTP data trace

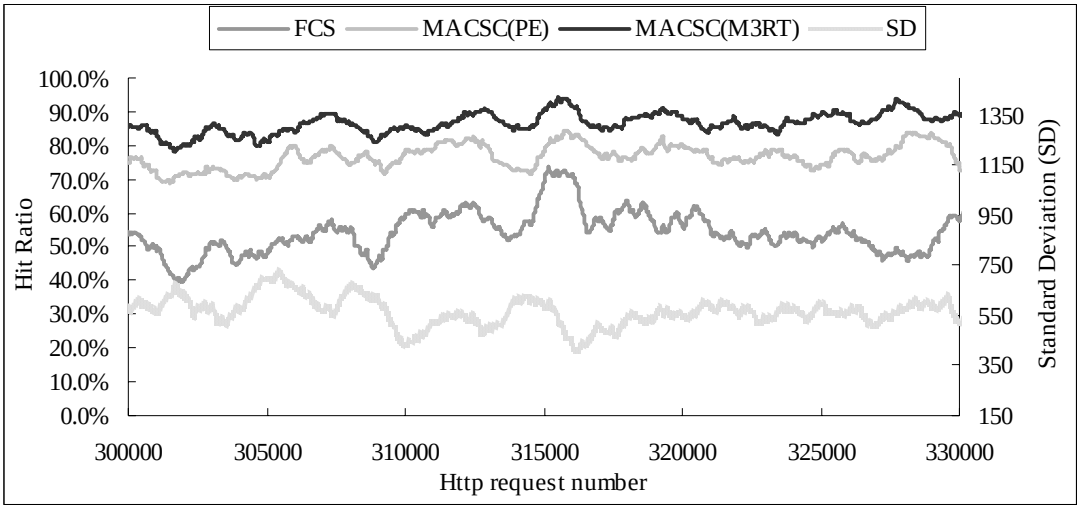


Figure 7.15 The magnified view of changes in hit ratios by different frameworks with the Calgary-HTTP data trace

In Figure 7.16, the memory usage by different frameworks with different data traces is shown. It shows that although the MACSC(M^3RT) maintains a higher hit ratio than the MACSC(PE) approach, it also uses more memory than the MACSC(PE).

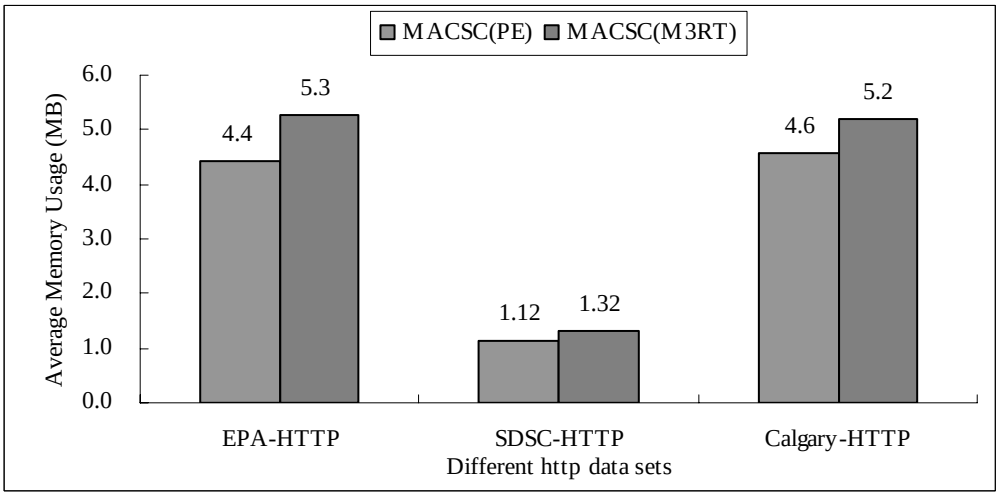


Figure 7.16 The memory usage by different frameworks with different data traces

7.4 CONNCECTIVE SUMMARY

In this chapter the novel MACSC(M^3RT) approach is introduced. In this framework the M^3RT mechanism has replaced the PE approach. The verification

experiments show that the M^3RT approach has the capability to maintain a high cache hit ratio like PE but with a smaller sample size (17 to 22 samples in each prediction cycle). The physical computation time of the M^3RT is consistently lower than the PE. The feedback system of the M^3RT approach reduces the hit-ratio oscillation problem. More experimental results will be presented in CHAPTER 10. Memory usage by the MACSC(M^3RT) is, however, higher than that by MACSC(PE). To resolve this shortcoming, another approach, namely, MACSC(F-PE) is explored. In the MACSC(F-PE) approach: the PE sensitivity and the M^3RT feedback stability are combined.

CHAPTER 8

OPTIMAL DYNAMIC CACHING SIZE TUNING

8.1 INTRODUCTION

In this chapter, the optimal dynamic cache size tuning framework, MACSC(F-PE) will be introduced. The previous MACSC(PE) approach had two shortcomings: a) unpredictable computation time, and b) serious hit ratio oscillations. The interim MACSC(M^3RT) approach was proposed to resolve these problems. The MACSC(F-PE) utilizes the stability advantage from the feedback loop of the M^3RT technique to reduce hit ratio oscillations. The verification experiment shows that the MACSC(F-PE) approach indeed achieves the objective successfully. It is also found that the MACSC(F-PE) maintains the cache hit ratio more efficiently than both the MACSC(PE) and the MACSC(M^3RT) tuners.

8.2 THE MACSC(F-PE) APPROACH

The MACSC(F-PE) approach alleviates hit ratio oscillations as shown by the MACSC(PE) approach. These oscillations are caused by inaccurate calculation of the standard deviation of the relative data object profile. This reduces the overall performance of the dynamic cache size tuning mechanism. “F-PE” stands for *fine-tuned point estimate*, which is based on the successful MACSC(M^3RT) experience of using history information of the data objects as feedback for stability. The objective of the feedback system is to resolve the problem of heavy oscillation

in the MACSC(PE). It involves the following steps: 1) compute $\bar{x}$ from the N samples by using the PE approach, namely, the $\sqrt{N}-equation$, and then the standard deviation s_x. 2) the value is fine-tuned and changes to s_x^z by equations (8.1) with feedback s_x^{z-1} and weight β, where z is the operation cycle. 3) the adjusted cache size or ACS is conceptually determined by equation (8.2) and its implementation is represented by equation (8.3) to avoid propagation of any cache size initialization error.

$$s_x^z = \beta s_x^{z-1} + (1-\beta)s_x^z \tag{8.1}$$

$$CacheSize_z = CacheSize_{z-1} * \left(\frac{s_x^z}{s_x^{z-1}} \right) \tag{8.2}$$

$$CacheSize_z = 2 * \nabla * s_x^z * OS_{average} \tag{8.3}$$

The MACSC(F-PE) mechanism cycles through the following steps:

1. The system collects a number of data objects with the size n.
2. It calculates the mean and standard deviation according the sample data objects.
3. It uses the $\sqrt{N}-equation$ to determine the reasonable size N that the system should collect in order to have an acceptable mean and standard deviation.
4. If the sample size n is larger than or equal to N, the system will go to step 5. Otherwise, the system goes back to step 1 to collect more data objects for further calculation.
5. The system calculate the s_x^z value by equation 8.2 feedback s_x^{z-1} and weight β
6. The system adjusts the cache size according to popularity ratio based on equation 8.3 and the s_x^z value and feedback the value as s_x^{z-1} for the next

cycle computation.

7. The system goes back to step 1 to collect n new data objects for next cycle.

Figure 8.1 summarizes the MACSC(F-PE) operation.

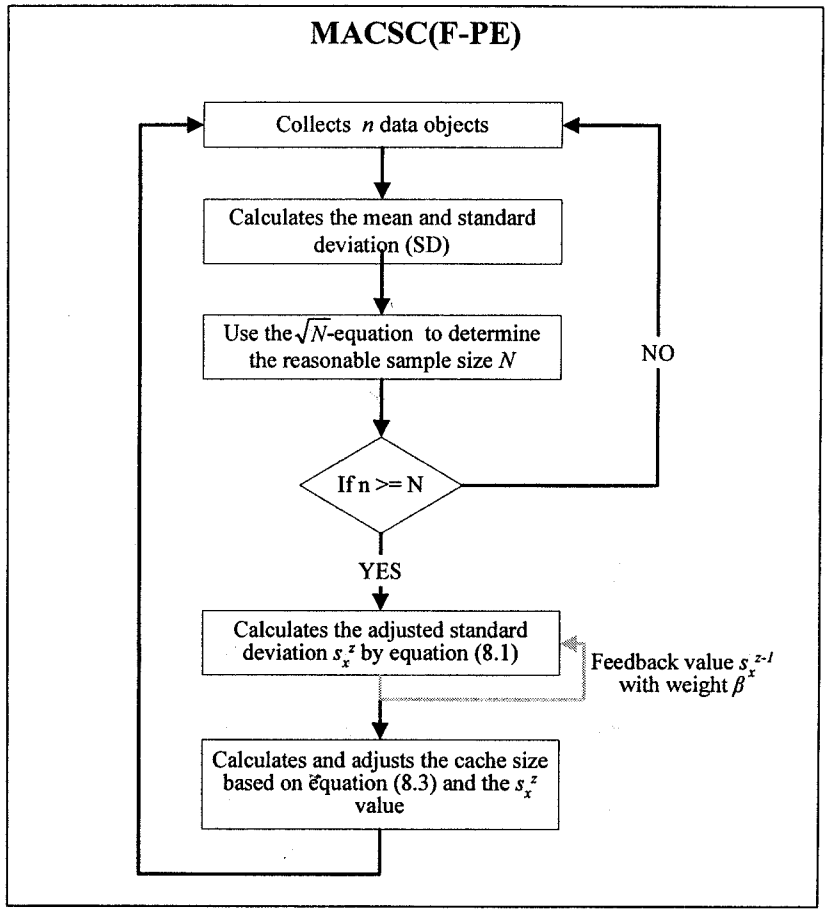


Figure 8.1 The summary flow of the MACSC(F-PE) approach

8.3 MACSC(F-PE) VERIFICATION

8.3.1 SETUP AND ENVIRONMENT

Many simulation experiments were carried out to verify the MACSC(F-PE) approach and compare it with the MACSC(M^3RT), MACSC(PE) and FCS dynamic cache size tuners. The objective is to confirm that the MACSC(F-PE) approach can indeed reduce the oscillation problem of the PE by using a fine-tuned feedback system. The experimental setup is illustrated in Figure 8.2, which is similar to that

for verifying the previous frameworks. The MACSC(F-PE) experiments are basically separated into two different types: a) verification with simulated datasets and b) verification with pre-collected real-life data traces. The aim of the second type is to ensure that the MACSC(F-PE) approach can indeed work well with real-world operations.

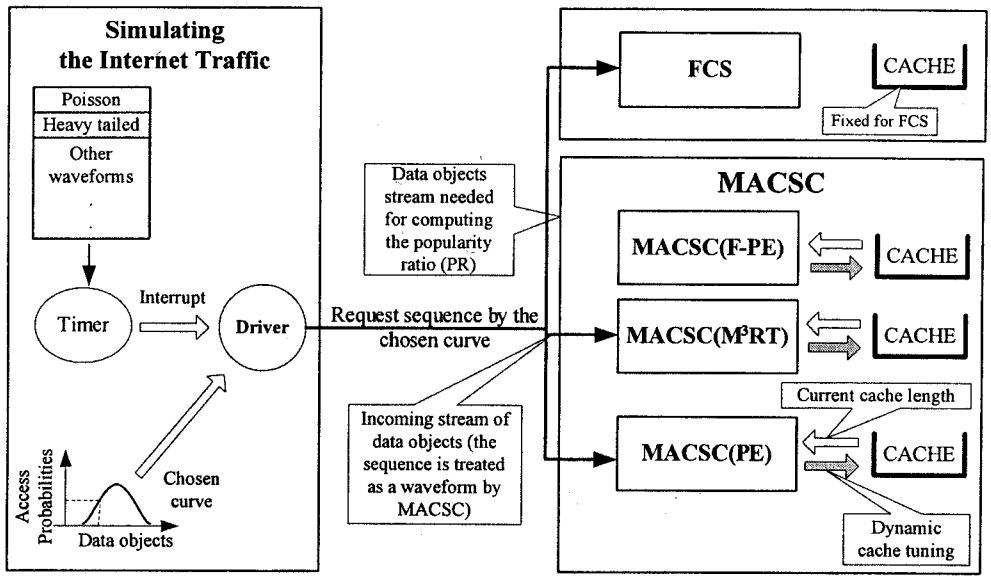


Figure 8.2 Verification set up for the MACSC(F-PE) tuner (for the proxy) by simulation

8.3.2 VERIFICATION WITH SIMULATED DATASETS

In these experiments, a large number of simulated relative data object popularity profiles (or datasets) are used. These datasets are the same as those for verifying the MACSC(PE) and MACSC(M^3RT) prototypes (Table 6.1). The cache size is first initialized to meet the prescribed hit ratio of one standard deviation. Then, the MACSC(F-PE) weight β is set to different values, for example, 0.9.

Figure 8.3 (simulated standard deviation (SD) sequence, 1k→2.5k→1.5k→2k) shows that the MACSC(F-PE) maintains the given hit ratio better than the MACSC(M^3RT), MACSC(PE) and FCS. Figure 8.4 shows the hit ratio improvement of different frameworks over FCS. Figure 8.5 and 8.6 shows the results of another

simulation (simulated standard deviation (SD) sequence, 1k→2k→1.5k→3k). The MACSC(F-PE) consistently outperformed the other tuners in the experiments by maintaining the highest cache hit ratio.

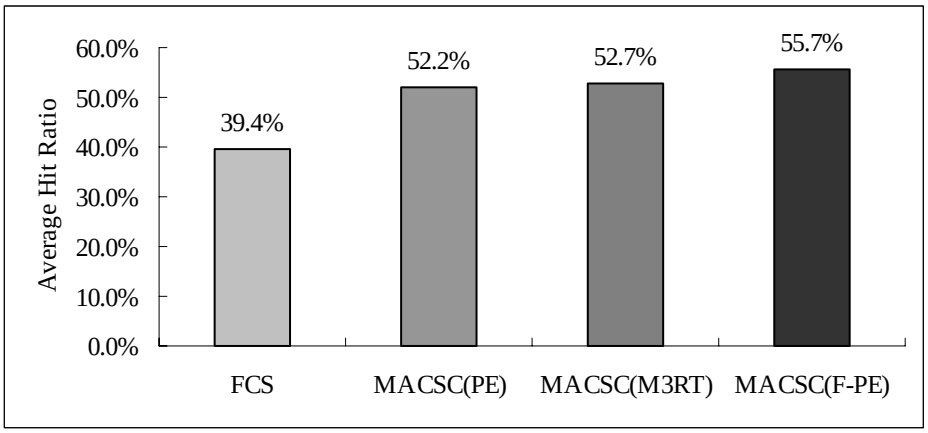


Figure 8.3 The comparison of the hit ratio between different frameworks (Interleaved SD sequence: 1k→2.5k→1.5k→2k)

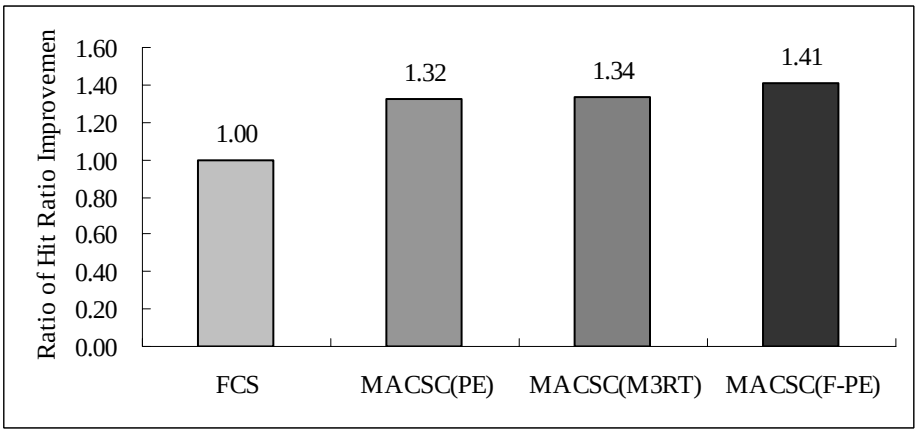


Figure 8.4 The hit ratio improvement over FCS (Interleaved SD sequence: 1k→2.5k→1.5k→2k)

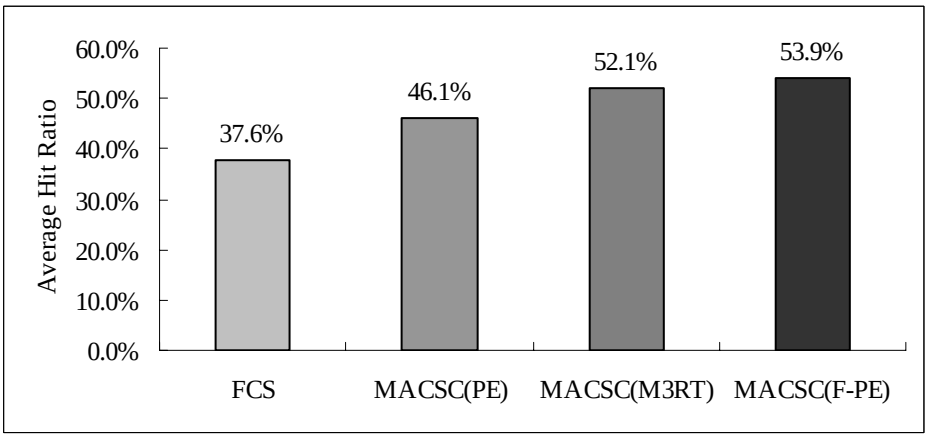


Figure 8.5 The comparison of the hit ratio between different frameworks (Interleaved SD sequence: 1k→2k→1.5k→3k)

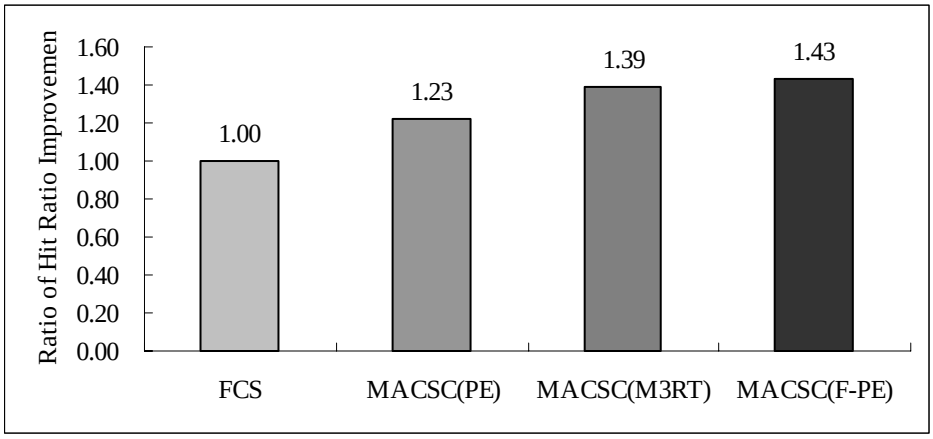


Figure 8.6 The hit ratio improvement over FCS
(Interleaved SD sequence: 1k→2k→1.5k→3k)

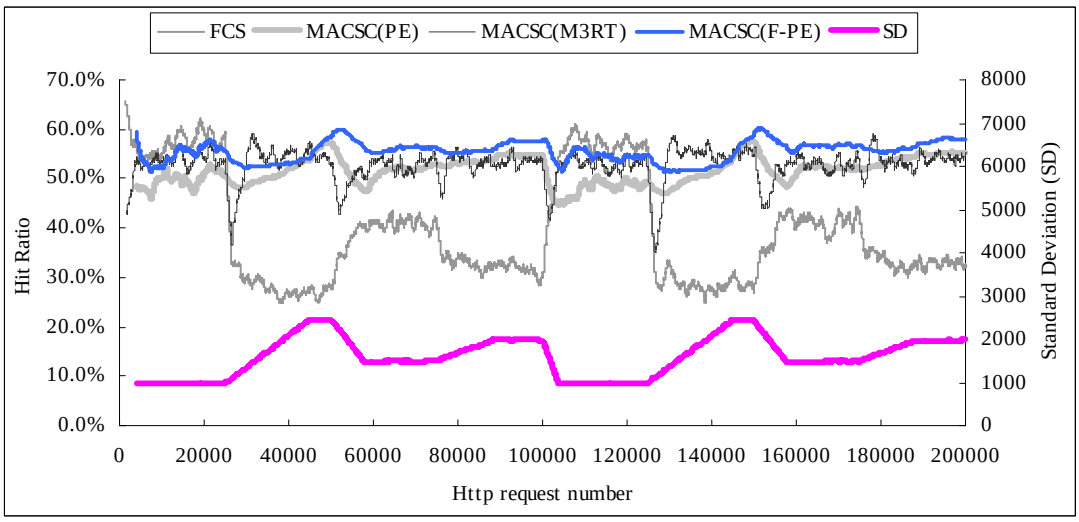


Figure 8.7 Changes of hit ratios by different frameworks
(Interleaved SD sequence: 1k→2.5k→1.5k→2k)

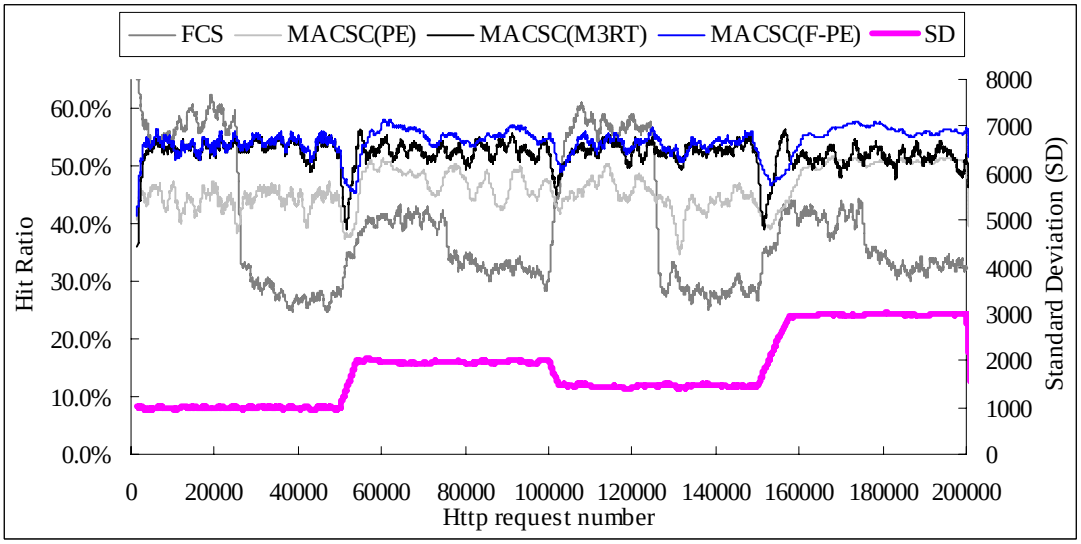


Figure 8.8 Changes of hit ratios by different frameworks
(Interleaved SD sequence: 1k→2k→1.5k→3k)

Figure 8.7 and Figure 8.8 show the changes of the hit ratio by FCS, MACSC(PE), MACSC(M^3RT) and MACSC(F-PE) respectively over time. In these figures, it can be observed that the MACSC(F-PE) maintains a higher cache hit ratio and is more stable than the M^3RT and PE based approaches. The MACSC(F-PE) maintains the highest hit ratio of them all, and its feedback system eliminates most of the oscillations compared to the PE approach.

The Figure 8.9 shows the memory usage by the different approaches with different simulated datasets. It can be observed that the memory usage of the MACSC(F-PE) approach is nearly equal to that by the MACSC(PE). The MACSC(M^3RT) use more memory compared with them. However, the performance of hit ratio maintenance by the MACSC(F-PE) is the best.

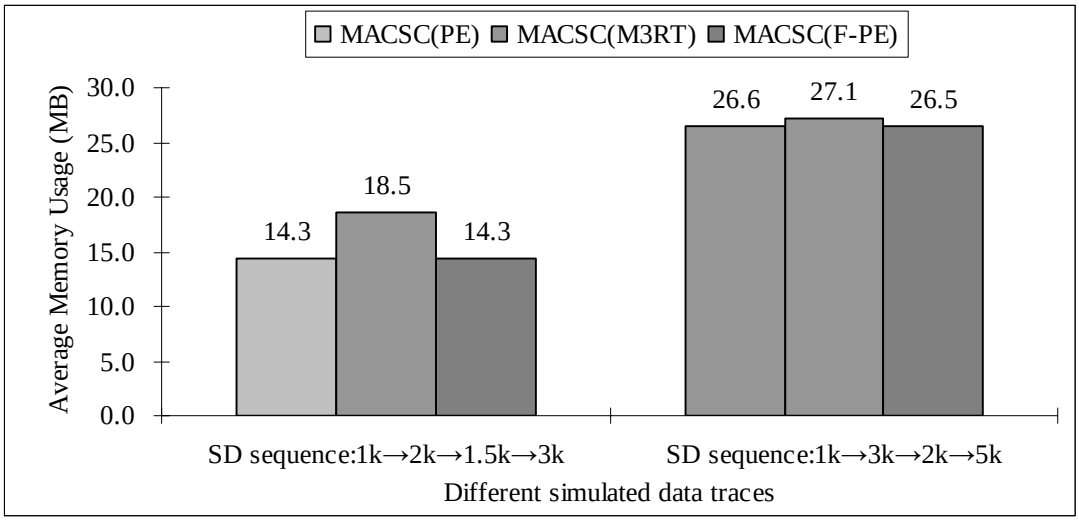


Figure 8.9 The memory usage by different frameworks with different simulated data traces

8.3.3 VERIFICATION WITH PRE-COLLECTED DATA TRACES

In these simulation experiments, different pre-collected data traces from real-life operations are used to verify the MACSC(F-PE) (Table 6.2). Figure 8.10 compares the experimental results of the FCS, MACSC(PE), MACSC(M^3RT) and

MACSC(F-PE) approaches. It indicates that the MACSC(F-PE) indeed consistently maintains the highest cache hit ratio. Figure 8.11 shows the changes of the hit ratio of the different tuners with the EPA-HTTP trace. The MACSC(F-PE) yields the highest hit ratio, and this phenomenon is also observed in Figure 8.12 and Figure 8.13.

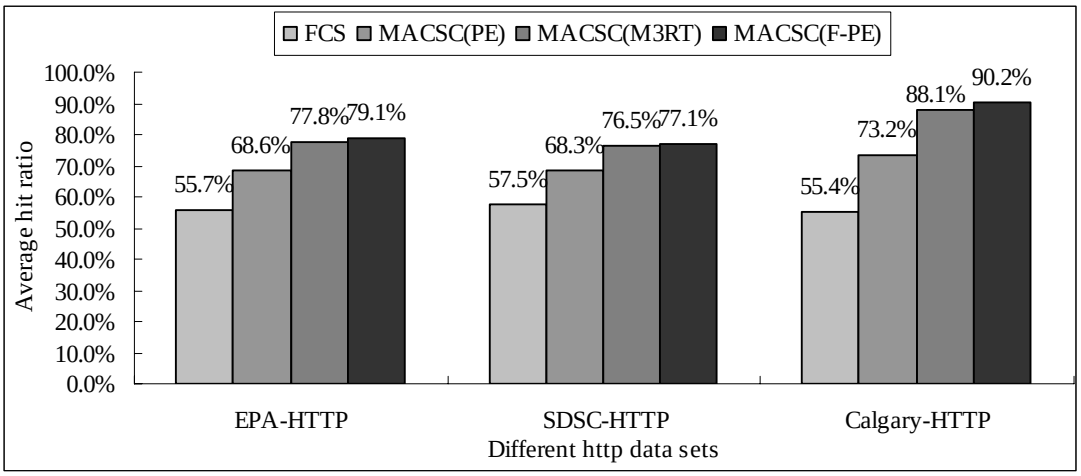


Figure 8.10 Performance comparison of different frameworks for different data traces

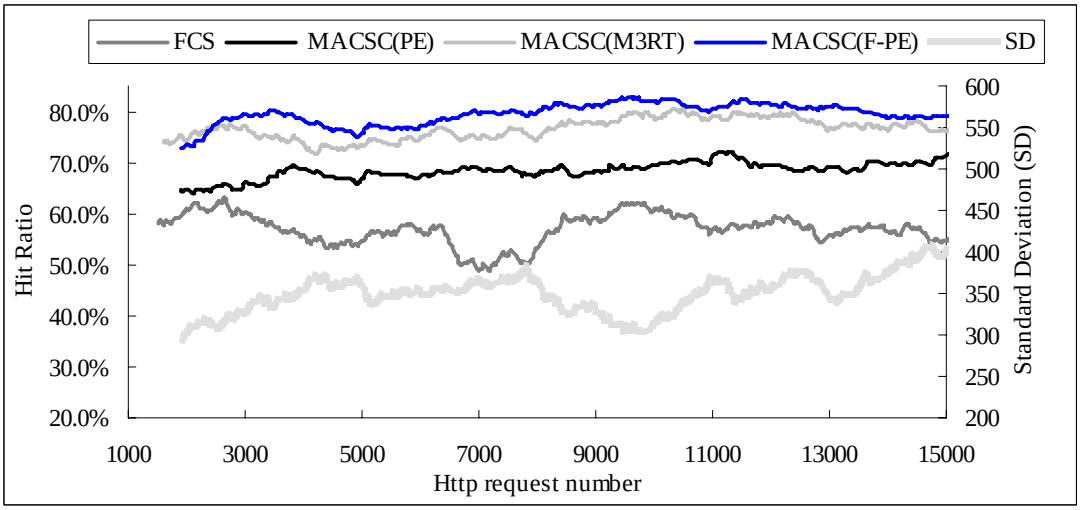


Figure 8.11 The Magnified view of the change of hit ratios by different frameworks with the EPA-HTTP data trace

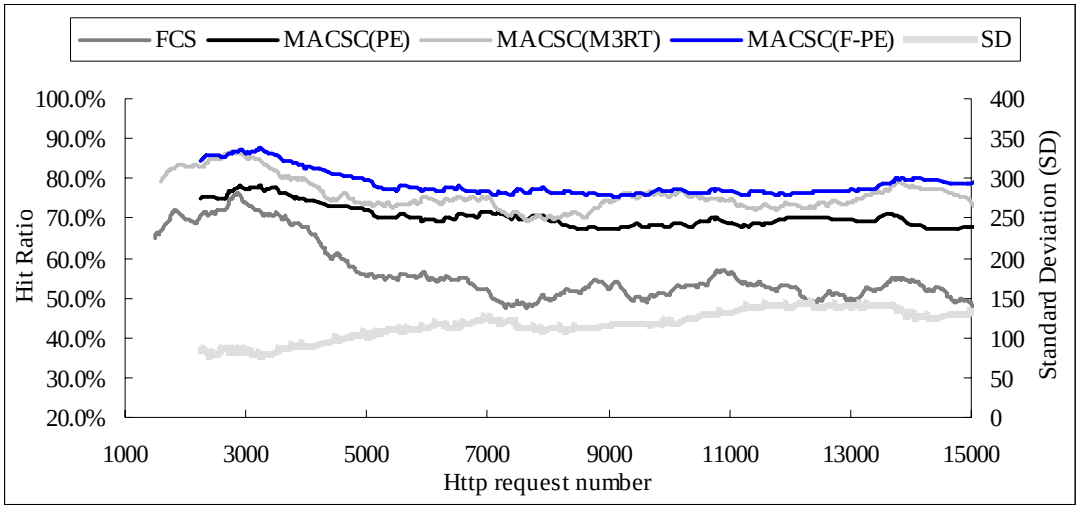


Figure 8.12 The Magnified view of the change of hit ratios by different frameworks with the SDSC-HTTP data trace

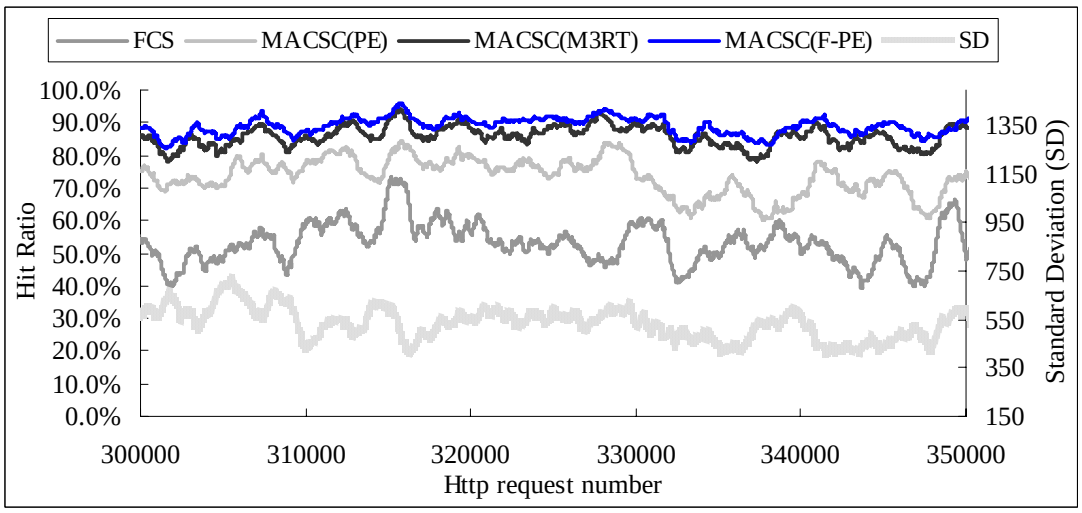


Figure 8.13 The magnified view of changes in hit ratios by different frameworks with the Calgary-HTTP data trace

Figure 8.14 shows the memory usage by the different tuners. Despite the fact that the MACSC(F-PE) is the most efficient tuner in hit ratio maintenance, it uses less memory than the MACSC(PE).

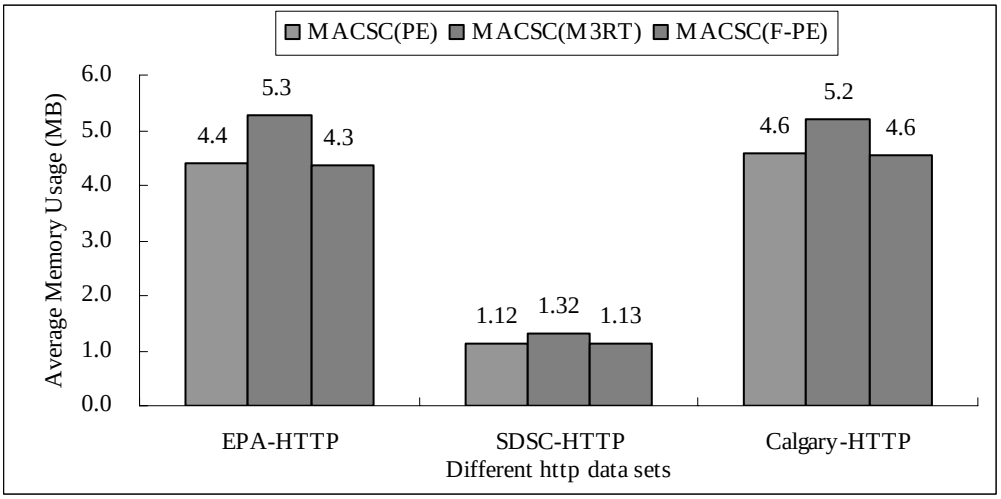


Figure 8.14 The memory usage by different frameworks with different data traces

8.4 CONNECTIVE SUMMARY

This chapter introduces the novel MACSC(F-PE) approach, which combines the PE technique with a feedback mechanism. This mechanism is inspired by the MACSC(M^3RT) experience of using history information in the current hit ratio maintenance operation. The experiment results confirm that the feedback approach has greatly reduced the oscillation problem of the pure PE technique. Furthermore, the MACSC(F-PE) is the most efficient compared with the MACSC(PE) and MACSC(M^3RT) approaches. In the next chapter the impact of different Internet traffic patterns to the MACSC operation in general will be discussed.

CHAPTER 9

REAL-TIME TRAFFIC PATTERN DETECTION

9.1 INTRODUCTION

The Internet follows the power law [Medina00] and its traffic can assume different forms over time, for example, long-range dependence (LRD) and short-range dependence (SRD) [Molnár99]. LRD traffic includes heavy-tailed and self-similar patterns, and SRD includes Markovian traffic patterns such as Poisson. Continued studies of different traffic patterns led to the conclusion by Paxson [Paxson95] that any system, which is designed with a preconceived mathematical model in mind, for example, Poisson, would fail over the Internet. This implies that any systems running over the Internet should be adaptive in nature. These systems should adapt their processes with respect to the changes in the operation environments anytime. To support dynamic adaptation such the system should be able to achieve the following: a) to sample the indicative performance parameters (such as average service roundtrip time (RTT)) to gauge changes in the environment, b) to compute the necessary adjustment or reconfiguration with the sampled statistics quickly and accurately to avoid deleterious effects, and c) to administer the computed adjustment on the fly. Deleterious effects are the undesirable effects that arise from the correction of a spurious problem with a computed result. This happens because the computation time of the corrective solution is longer than the duration of the problem. By the time the solution is ready, the problem has long gone. The

deleterious problem can be detrimental to time-critical applications, which usually have at least one of the following requirements [Stankovic98]:

a. Hard nature: The result must be obtained within the deadline for safe operations.
b. Soft nature: The result is acceptable even those with occasional slippage of the deadline, or some minor slippage of it.
c. Firm nature: The result after the deadline is meaningless. That is why in smart systems they would proceed with an operation only if it is certain that the result is produced before that deadline.

In the different experiments with the MACSC(PE) prototype, it was observed that different IAT traffic patterns can affect the dynamic cache size tuner's accuracy. Different timing analyses of the prototype were carried out with different pre-collected traces of http requests in a postmortem manner by using the Intel® VTune™ Performance Analyzer [VTune]. It was found that the prototype intrinsically needs an average of 1,673,100 clock cycles to execute [Wu03]. It is intrinsic because the data items are immediately available from the trace without actual delays. In real-life cases, the tuner has to collect the data items one by one with IAT (inter-arrival time) delays between them. Therefore the actual tuner execution time is much longer than the intrinsic one. For a node that operates at the speed of 143 mega hertz, for example, the physical time is $1{,}673{,}100*(143*10^{-6})$ or 11.7ms. If the IAT among the data items in real-life applications is consistently less than 11.7ms, then the MACSC(PE) mechanism would miss sampling many data items, and this leads to erroneous cache ratio maintenance. Furthermore, different traffic patterns will have different impacts on the loss of the data items. To

demonstrate this phenomenon, experimental results with the EPA-HTTP (Environmental Protection Agency) trace [SIGCOMM] are shown here. Figure 9.1 shows the IAT trace of 42,438 http requests of the LRD nature. Figure 9.2 shows the trace of Poisson IAT for comparison. The R/S plot in Figure 9.3, which is a postmortem traffic analysis technique to be described in detail later, confirms the Hurst value for this trace is 0.761. Figure 9.4 shows that the R/S plot of Poisson IAT, which has a Hurst value of 0.491. Figure 9.5 shows data loss (i.e. missed data items) versus IAT with different traffic patterns. (i.e. SRD (EPA-HTTP) and LRD (Poisson)) Figure 9.6 shows how the cache hit ratio deteriorates. It shows that different traffic pattern will lead to different impact on the MACSC framework.

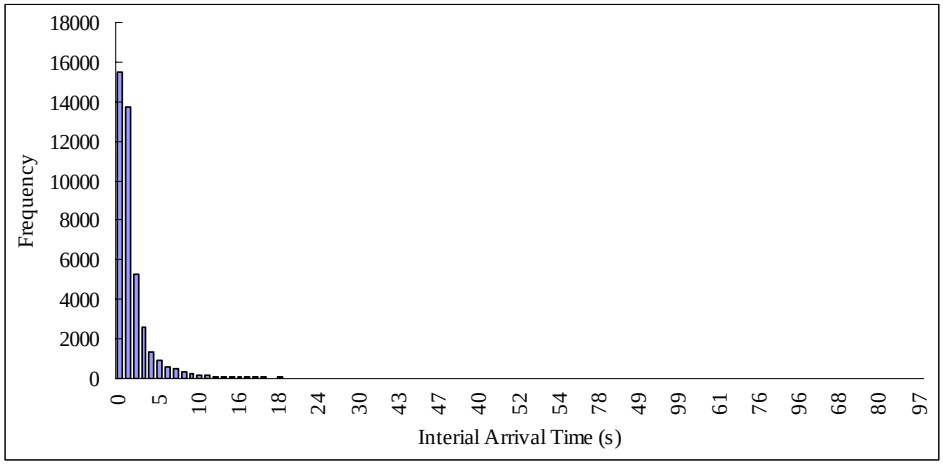


Figure 9.1 LRD frequency distribution of http request to the EPA dataset, mean IAT is 2

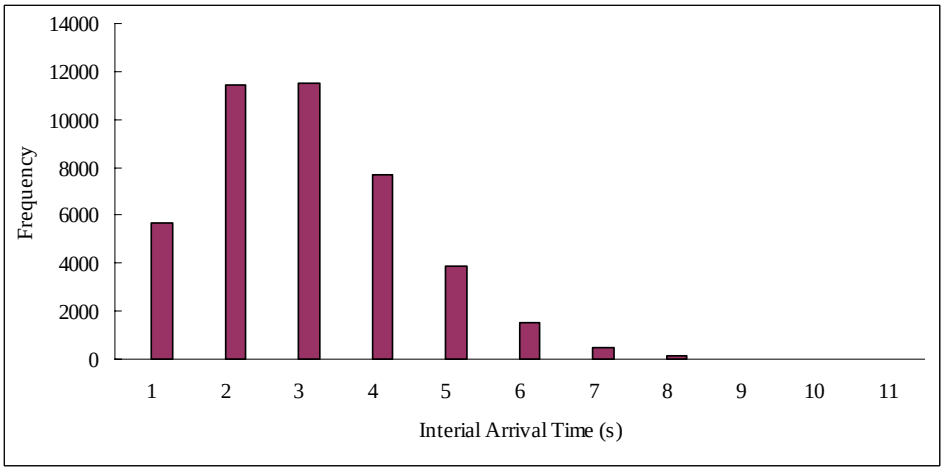


Figure 9.2 SRD frequency distribution of the Poisson IAT, mean IAT is 2

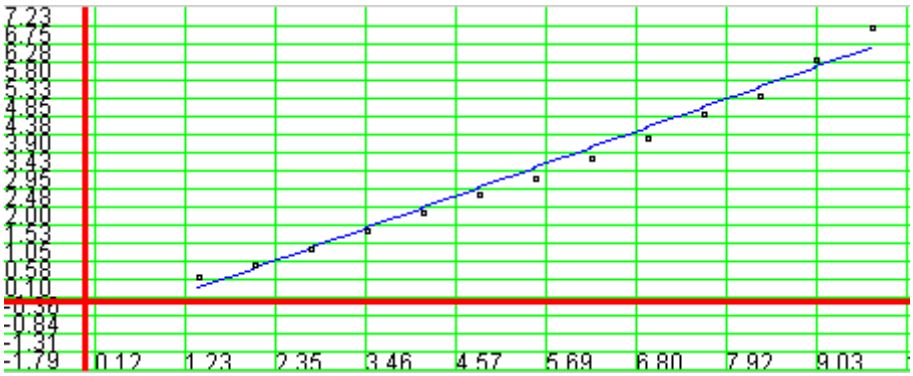


Figure 9.3 R/S plot confirms LRD nature for EPA trace, H=0.761 with 99.3% confidence

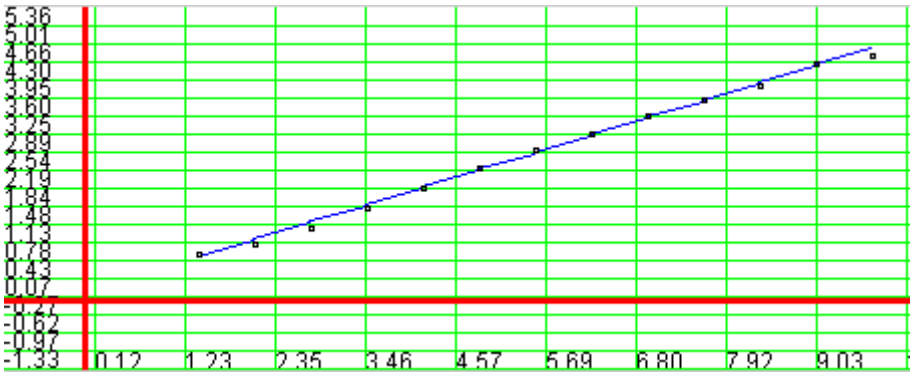


Figure 9.4 R/S plot confirms SRD nature for Poisson trace, H=0.491 with 99.87% confidence

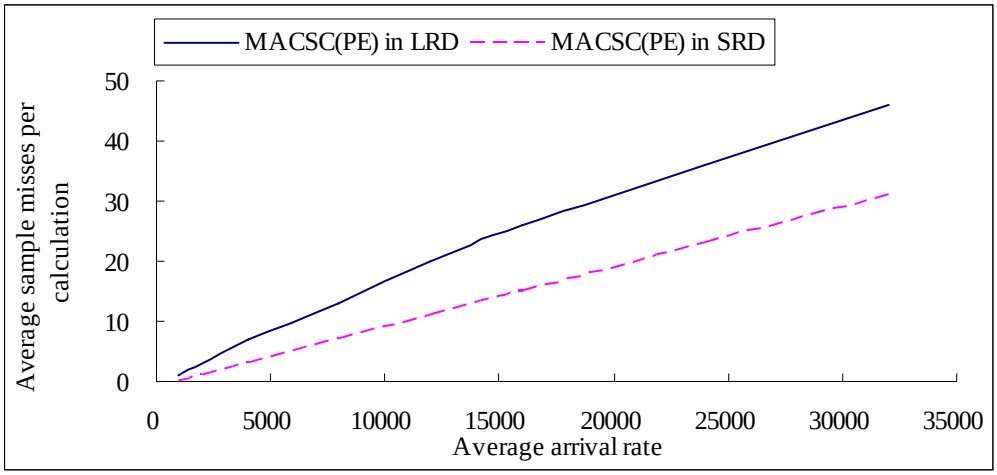


Figure 9.5 Data loss (missed data) of MACSC(PE) versus IAT for the LRD and SRD traces

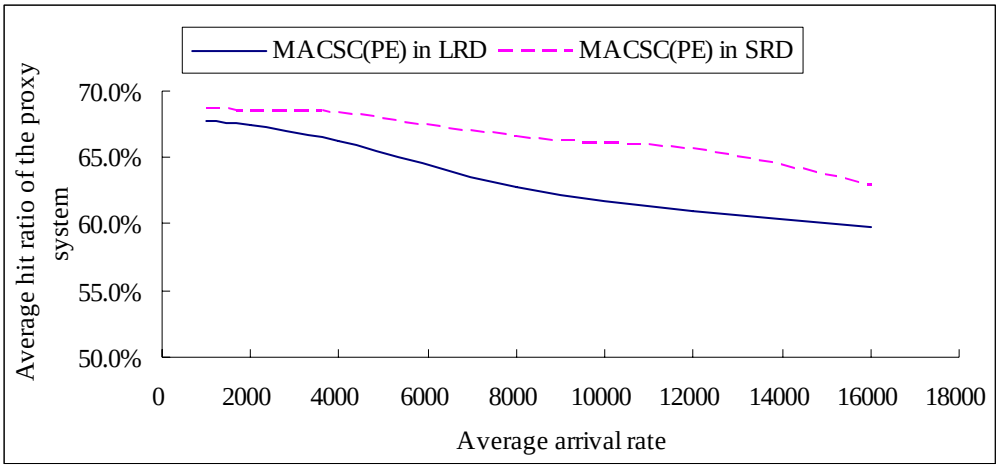


Figure 9.6 Hit ratio of MACSC(PE) versus IAT for the LRD and SRD traces

The experimental results with the MACSC(F-PE) prototype shows that its performance is traffic independent. Figure 9.7 and 9.8 shows how data loss and cache hit ratio correlate with IAT for SRD (EPA-HTTP) and LRD (Poisson) traffic patterns. Clearly the traffic patterns have no negative impact on the MACSC(F-PE) performance because of the presence of the feedback loop in the solution.. The β value of the MACSC(F-PE) for the experiments that produce the effects in Figure 9.7 and 9.8 is set to 0.9.

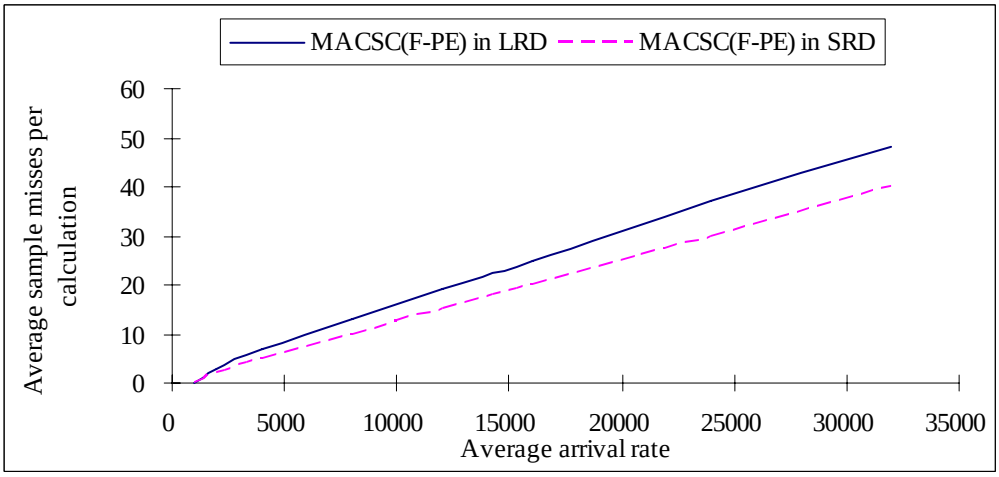


Figure 9.7 Data loss (missed data) of MACSC(F-PE) versus IAT for the LRD and SRD traces

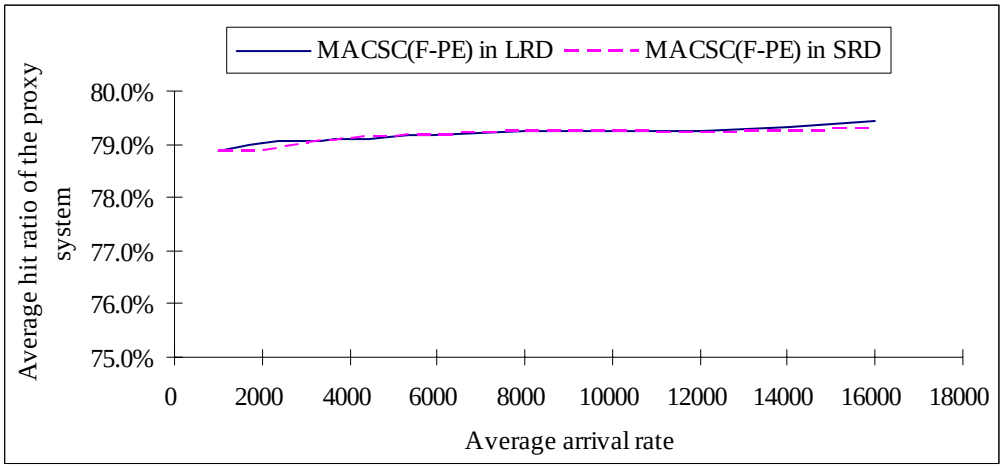


Figure 9.8 Hit ratio of MACSC(PE) versus IAT for the LRD and SRD traces

9.2 DIFFERENT TYPES OF TRAFFIC PATTERNS

The Internet follows the power law [Resnick97] and its traffic pattern can change suddenly from LRD and SRD. This phenomenon was investigated by different researchers [Molnár99]. The important findings in Internet traffic research include the following:

1. One could use Guassianity test (Gaussian means normal distribution of kurtosis and skewness values equal to 3 and 0 respectively) to indicate that an aggregate X_l^m in a stochastic process X is stationary. The parameter *m* is the block size of the aggregate and *l* is the lag, for *l*=1,2,3…*n*. A process is stationary if it has independent increments [Leland94], [Taqqu03], [Willinger03].
2. One could use the Hurst value measurement to determine if an aggregate is LRD or SRD, where $0<H<0.5$ indicates SRD and $0.5<H<1$ for LRD traffic.
3. There are different estimators/filters proposed for identifying specific traffic patterns from pre-collected traces/datasets (i.e. postmortem analysis), for example, Poisson and heavy-tailed. The details of this filter will be elaborated later.

SRD/LRD differentiation	Heavy-tailed traffic	Self-similar traffic
R/S (Rescaled Statistics) plot, Periodogram [Molnár99]	Modified QQ-plot [Embrechts97], De Haan's Moment [Resnick97]	Whittle, Variance-time plot [Molnár99]

Table 9.1 Summary of different techniques for postmortem traffic analysis

Table 9.1 summarizes the different estimators that can be applied for traffic pattern detections. In fact, these techniques were implemented and put together in the Selfis tool [Karagiannis02], [Karagiannis03]. For example, Figure 9.9 shows how the R/S (Rescaled Statistics) plot mechanism of the Selfis identifies the LRD character from an IAT trace. In this case, the H value is 0.706 with 97.89% confidence for the LRD identification. The Periodogram of the Selfis also confirms the LRD character for the same trace as shown in Figure 9.10.

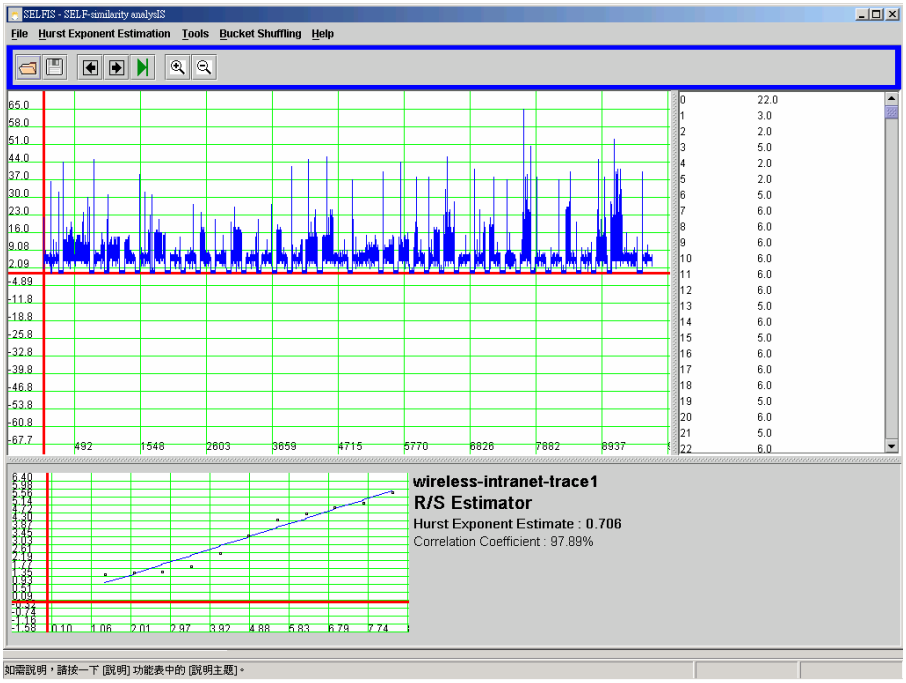


Figure 9.9 LRD identification by Selfis's R/S estimator

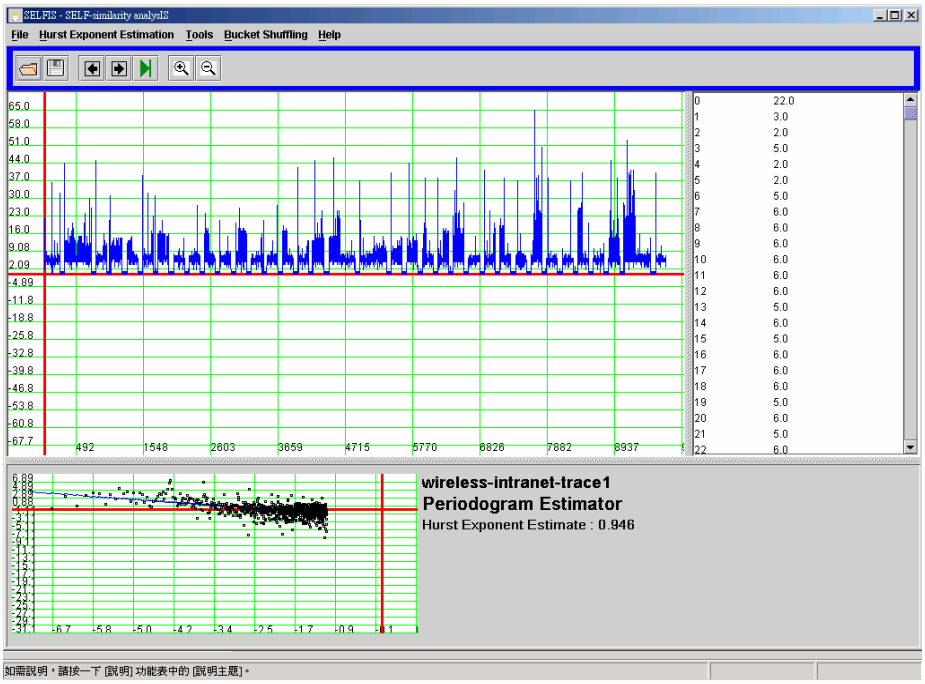


Figure 9.10 LRD confirmation by the Selfis's Periodogram estimator

Postmortem traffic analyses with traces are of the "lump nature" at the present moment as concluded by the COMP Team because of the following reasons:

1. *Composite results*: Techniques or estimators such those shown by Table 9.1 or assembled by the Selfis tool produce the results for the given traces. The results as such are composite in nature because a trace may embed different traffic patterns in different segments. This kind of lump analysis is misleading and unsuitable for real-time applications unless there exists a technique/tool that can confirm the stationarity of an aggregate before the analysis starts. That is why the COMP Team has proposed the CAB (Continuous Aggregate Based) approach that can identify the stationarity of an aggregate.
2. Theoretical discrepancy: In the Internet traffic literature Gaussinaity is equated to stationarity. While this may be true for continuous stochastic processes, there is room for errors for discrete processes. Gaussianity is based on the concept of

a perfect normal distribution of kurtosis and skewness values equal to 3 and 0 respectively. Yet, the Gaussian distribution can provide a good approximation for the binomial and Poisson distributions only under certain conditions [Jain91]. The Poisson distribution is a good approximation of the binomial distribution only for rare events; Poisson and binomial distributions are memoryless. The concepts of Gaussianity, Poisson, and binomial are lumped together in most literature [Leland94]. For example, the basic theory for postmortem self-similar traffic analysis in the field is the fractional Brownian motion, which has stationary increments. The incremental process is known as fractional Gaussian noise of the LRD character [Leland94]. This conceptually but vaguely links Gaussianity to a stationary process, whether discrete or not.

9.3 REAL-TIME TRAFFIC ANALYSIS

There is little experience in real-time traffic analysis until the COMP Team (or simply the Team) has proposed how it could be done [Lin04]. The argument put forth by the Team includes the following:

1. SRD/LRD differentiation: The Team found that the R/S plot is very effective in differentiating SRD from LRD. The existing form, however, should be adapted to suit real-time application. The details will be provided later.
2. Java API: It is logical to convert the different traffic estimators into Java API so that the *real-time traffic detection/detector* (RTPD) mechanism can invoke the appropriate filter anytime and anywhere as a logical object to provide the necessary service.
3. CAB mechanism: The basic argument for the CAB mechanism is to have a

variable block size *m* for the l^{th} aggregate X_l^m of a stochastic discrete process X. The aim is to increase *m* until the property of stationarity has appeared. Only then the R/S plot, which is supported by the M^3RT [Wong02b], which is a micro IEPM (Internet End-to-End Performance Measurement [Cottrel01]) tool, is invoked to differentiate SRD from LRD. Once the LRD character is identified the filtration process invokes the appropriate filter to identify whether it is heavy-tailed or self-similar. The presence of stationarity in X_l^m is indicated by the "kurtosis-skewness (KS)" test. The indication is obtained by comparing the kurtosis and skewness values with that of the normal distribution (kurtosis = 3 and skewness = 0). If the difference is within a pre-defined limit, then stationarity is assumed to have appeared. For example, the workable kurtosis and skewness limits found by experimentation by the Team are 9 and 100 respectively.

The Team's RTPD concept is basically the combination: "R/S plot + M^3RT + filtration".

9.3.1 THE KS TEST

Skewness is shown by equation (5.7), where $\bar{x}$ and *SD* are the measured mean and standard deviation respectively for the aggregate of *m* data items. For a real-time aggregate X_l^m, where *l* is the lag. The skewness value measures the symmetry of a bell curve embedded in the sample of size *m*. A positive value means that the bell curve skews right (i.e. right tail is heavier). Kurtosis is shown in equation (5.8), which decides if the bell curve is *peaked* (positive value) or *flat* (negative value)

compared to the normal distribution (Gaussian) with kurtosis=3 and skewness=0. For example, the EPA-HTTP trace from [SIGCOMM] has skewness and kurtosis values of 0.00076 and 8.47 respectively. These values indicate a symmetric bell curve that is more peaked than the normal distribution.

9.3.2 THE RTPD

Since the efficacy of the MACSC(PE) in dynamic cache size maintenance is affected by the IAT traffic pattern, it is necessary to compensate for any ill effects introduced by the traffic. To achieve this the MACSC(PE) should be supported by *real-time traffic detection/detector* (RTPD) capability. Then, the compensation mechanism involves the following:

1. Calibrate the ill effects with respect to different traffic patterns such as Poisson, heavy-tailed, and self-similar.
2. Incorporate RTPD capability so that traffic pattern changes can be monitored and detected.
3. Devise a reconfiguration scheme so that the MACSC(PE) mechanism would reconfigure on the fly with respect to the traffic pattern detected.

With agreement, I would make use of the Team's accumulated RTPD experience. In return I would contribute a novel R/S approach suitable for real-time applications to the Team and also as part of my PhD contribution.

The traditional R/S (*rescaled adjusted statistics*) computation is represented by the expression: $R/S = \frac{\max\{W_i : i = 1,2,....,k\} - \min\{W_i : i = 1,2,...,k\}}{\sqrt{\text{var}(X)}}$. The parameter

W_i is defined as $W_i = \sum_{m=1}^{i}(X_m - \overline{X})$ for $i = 1,2,...k$, where $\overline{X}$ is the mean computed by $\overline{X} = 1/k \sum_{i=1}^{k} X_i$. The best value for *k* has to be found by trial and error. This is the drawback for the traditional R/S method because the R/S of accuracy and speed depends on *k*. The R/S ratio is the rescaled range of the stochastic process X over a time interval *k*, where X is defined in discrete time (X_i: i =1,2…*k*). The most useful feature of the R/S plot is the relationship for large *k*: $R/S = (k/2)^H$. The H (Hurst) effect/value is also the slope of the *log-log* plot of *log(R/S) versus log(k)* [Molnár99]. For a stationary process the H effect 0.5<H<1 indicates LRD traffic behavior. For example, the LRD behavior of the trace for Figure 9.11 is indicated by the R/S plot shown in Figure 9.12 with H=0.7674.

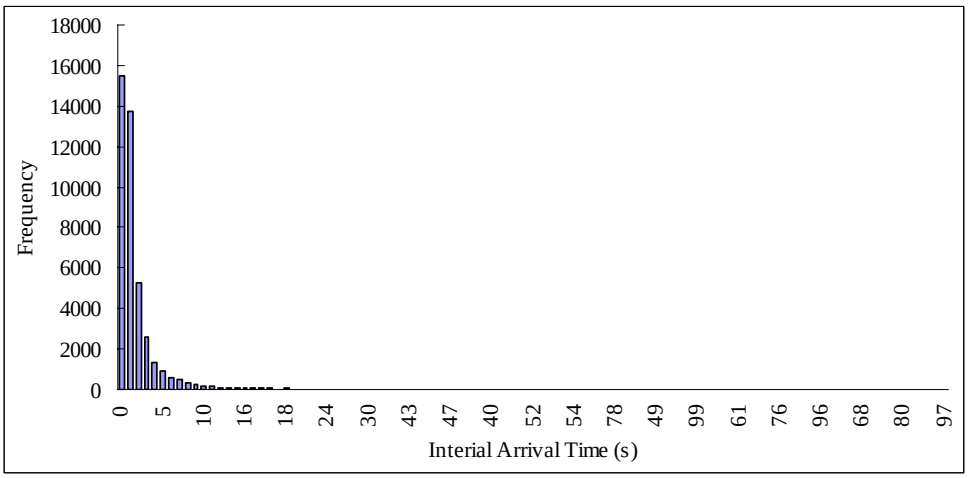


Figure 9.11 Frequency plot of the real-life EPA-HTTP IAT (inter-arrival time) trace

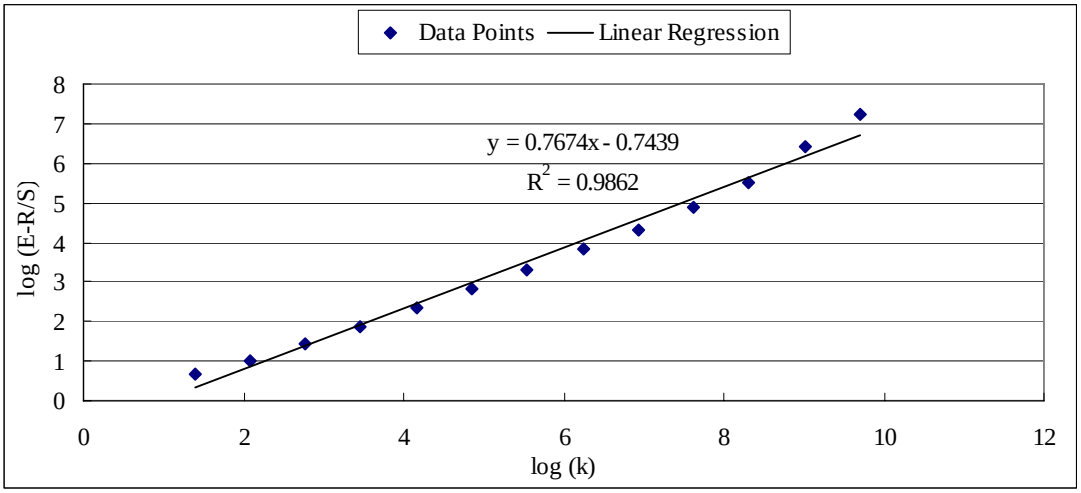


Figure 9.12 R/S plot for the trace in [SIGCOMM] (Figure 9.11) confirms its LRD behavior

RTPD is conceptually the "M^{3}RT + R/S" combination. The key point in my work is to shorten the R/S execution time by using the M^{3}RT mechanism, which can be inhibited or activated by the user. If the M^{3}RT is inhibited, the RTPD runs the traditional R/S computation. If it is activated, then it runs in parallel as an independent entity and makes the R/S estimator into an enhanced version (i.e. E-R/S) that is more suitable for real-time applications. It is "*enhanced*" because now $\overline{X}$ is replaced by the more accurate M_i. The E-R/S execution time becomes predictable because M_i is computed with a known f (*flush limit*) number of samples. The M^{3}RT predicts the mean of the waveform from the current aggregate X_l^m on the fly, independent of the E-R/S main body represented by $W_i = \sum_{m=1}^{i}(X_m - M_i)$, instead of the traditional $W_i = \sum_{m=1}^{i}(X_m - \overline{X})$. If the E-R/S main body needs the current M_i to compute the Hurst value for identifying the LRD or SRD character, it fetches it directly from M^{3}RT. That is, the M^{3}RT presence means $\overline{X} = M_i$ instead of

$\overline{X} = 1/k \sum_{i=1}^{k} X_i$. The filtration process identifies the traffic pattern, for example, heavy-tailed or self-similar for the LRD type. In action, the filtration process consists of a collection of logical objects (i.e. traffic filters/estimators) that specialize in identifying particular patterns. For example, the *modified QQ-plot filter* identifies heavy-tailed waveforms. These filters were proposed by the Team, for example, the self-similarity (S^2) traffic filter.

In each M_i prediction cycle only *f=14* data items are used to obtain the fastest convergence. Timing analysis with Intel® VTune™ Performance Analyzer shows that by itself the M^{3}RT needs 200 intrinsic clock cycles on average to execute [Wong02a], [VTune] provided that the *f* data items are immediately available (i.e. intrinsic) as part of a trace. In real applications the M^{3}RT execution time will be much longer because data has to be sampled on-line with unpredictable inter-arrival times (IAT) among them. For example, if the mean IAT for 14 samples were also 200 clock cycles, then the execution time would be =(14*200)+200=3,000 clock cycles. For a platform operating at the speed of 100 mega hertz (MHz), the physical time of 200 clock cycles is 200/(100*10^{-6}) seconds or 2 micro seconds.

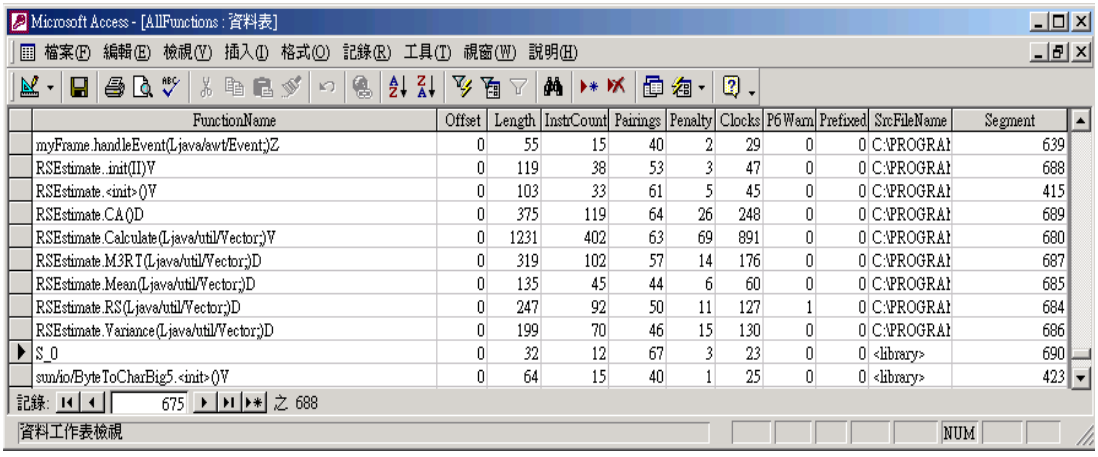


Figure 9.13 RTPD (E-R/S) execution time (891 clock cycles) by Intel® VTune™ Performance Analyzer

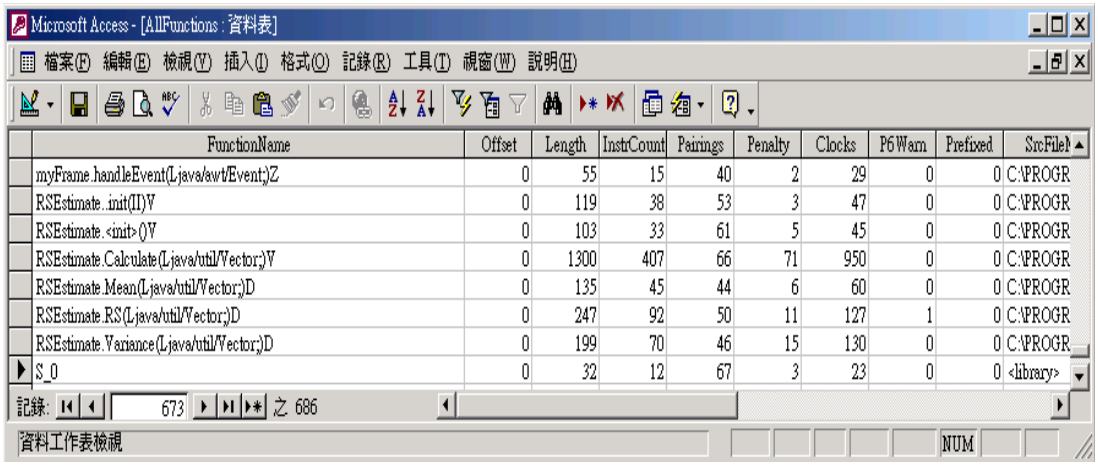


Figure 9.14 R/S execution time (950 clock cycles) by Intel® VTune™ Performance Analyzer (without M^{3}RT support)

Timing analyses of the Java-based RTPD prototype by the Intel® VTune™ Performance Analyzer in Figure 9.13 and Figure 9.14 show the following: a) the RTPD with M^{3}RT support (i.e. E-R/S) needs 891 intrinsic clock cycles to execute, and b) the version without M^{3}RT (i.e. traditional R/S) needs 950 intrinsic clock cycles for its execution. For 14 immediately usable data samples (i.e. f =14) in intrinsic cases the E-R/S is about 7% faster than the traditional R/S approach. In real-life applications, the E-R/S should be more efficient because more data and thus more delay in calculating $\overline{X} = 1/k \sum_{i=1}^{k} X_i$ will occur.

Since the traditional R/S estimator has been verified by different researchers already in the post-mortem manner [Molnár99], the experiments in this research focus on making sure that the E-R/S is faster than and as accurate as the traditional R/S approach. In fact, the drawback of the traditional R/S estimator is to estimate $\overline{X}$ by trial and error. The E-R/S does not have this problem because it always uses 14 data items to compute M_i. On the contrary, if more than 14 samples are used to compute $\overline{X}$, then the computation time by the traditional R/S would be much longer than the E-R/S's. In the light of this, the E-R/S execution time is more

predictable, and M_i is more accurate than $\overline{X}$ because it has a feedback loop in its computation, namely, M_{i-1}. In contrast the $\overline{X}$ computation in the traditional R/S estimator uses only the current data items without any feedback and this means more perturbations.

9.4 CONNECTIVE SUMMARY

In this chapter, the impact of the real-time traffic pattern on the MACSC framework is discussed. It shows that different traffic patterns will have different impacts on the dynamic cache size tuner's accuracy. So the *real-time traffic pattern detection/detector* (RTPD) is proposed to determine the traffic pattern of the IAT in order to have a different reconfiguration scheme for the MACSC framework to reduce the impact. It was found that the MACSC(F-PE) performance in dynamic cache size tuning is immune to the ill effects of different traffic patterns. The reason is the presence of a feedback loop, which is based on the MACSC(M^3RT) approach, in the solution. In the next chapter, the implementation of the MACSC framework, RTPD/MACSC(PE), in the real environment and the reconfiguration scheme by RTPD will be discussed.

CHAPTER 10

VALIDATION OF THE MACSC FRAMEWORK

10.1 INTRODUCTION

In the previous chapters, the three MACSC approaches, MACSC(PE), MACSC(M^3RT) and MACSC(F-PE) were discussed and verified. This chapter discusses the implementation and validation issues of these dynamic cache size tuners in light of real-life deployments over the Internet. The purpose of verification as shown in the previous chapters is to make sure that a framework is logically correct as it was intended set out to do. The design or implemented prototype is verified by simulation or pre-collected traces. The verified designed/prototype may not be able to meet the constraints imposed by the real environment and this requires proper modifications. The MACSC framework is intended for time-critical operations over the Internet and real data has to be sampled and used per dynamic cache size tuning cycle. To meet the time-critical requirement it becomes impractical to collect too many data items in order to compute the necessary cache size adjustment for maintaining the given hit ratio. However, there should be enough data items so that the Zipf-like behavior can be determined from the sampled data for the cycle so that the popularity ratio can be computed. This chapter proposes a solution for validating the MACSC solutions. The main difference between verification and validation experiments is that the former uses large pre-collected traces or simulated data sets and the latter relies on the limited number of real-time data items sampled

on the fly. In real-life applications the pre-collected traces used in the MACSC verification experiments normally do not exist. That is, the MACSC deployment relies on using the data items sampled per dynamic cache size tuning cycle. At the end of this chapter, the performance of the RTPD/MACSC(PE) as compared with different Internet traffic patterns will be presented.

10.2 PROPOSED IMPLEMENTATION SOLUTION

In the real environment, users send the URL requests to the proxy server continually. In the previous verification experiments the URLs are represented by unique integer object identifiers (OI), which provides the basis for the Zipf-like log-log plot. From this log-log plot the corresponding bell curve (i.e. the popularity distribution (PD)) is generated for calculating the popularity ratio. In order to facilitate the calculation of the PD standard deviation, the "bell" identifiers (BI) are assigned in ascending order. In real applications the BI values are generated anew in every dynamic cache size tuning cycle, which involves the following steps as depicted in Figure 10.1:

1. The system collects *n* (MACSC(PE) and MACSC(F-PE)) or *f* (MACSC(M^3RT)) data objects and counts the frequencies of the data objects identified in the current cycle.
2. From these data objects the ranker generates the Zip-like log-log plot.
3. The system transforms the Zip-like curve into bell curve and assigns the BI to the data objects.
4. The system calculates the popularity ratio of the data objects based on the BI.
5. The system adjusts the cache size on the fly according to the popularity ratio

derived from the current two standard deviations of the “BI” distribution.

6. The ranker prepares for the next new cycle (i.e. loop back to step 1).

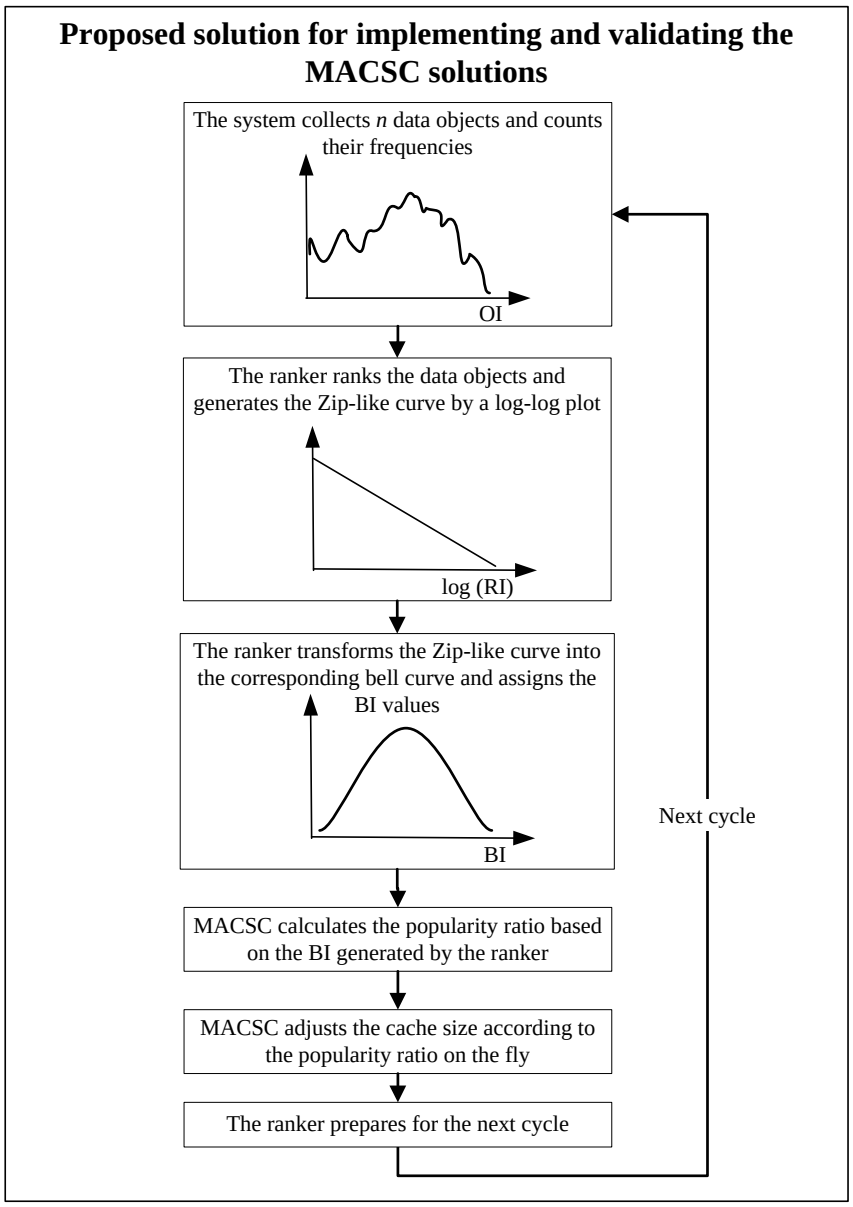


Figure 10.1 Proposed approach for validating the MACSC solutions

10.3 VALIDATION DETAILS

Any computer system design and implementation should involve verification and validation process. The design should be verified to make sure that it is logically correct and complies with the functionality. The verification is usually limited in

scope and can be carried out in different ways. For example, one can put the design into a Petri net [Tan94] and verify its logical and constraints correctness. One of the ways to achieve both objectives is using a “time Petri net” tool such as AlphaSim. Another way is to implement the design into a prototype so that it can be verified with chosen samples or datasets of its functionality. This empirical approach has its advantage because the same prototype can be validated later in controlled environments before the production phase. Therefore implementation and validation are inseparable. It is appropriate to run a system over a real Internet environment and collect the data for analyses and conclusion. Yet, another way is to use real-life data traces to drive the prototype and analyze its behavior. This is a flexible approach because traces of all kinds can be downloaded from well-known web sites [SIGCOMM]. Besides, the experience gained from the verification exercise becomes useful because the verification and validation results can be compared and evaluated. The validation of the MACSC solutions, namely, MASCS(PE), MACSC(M^3RT) and MACSC(F-PE) follows the second approach. That is, traces downloaded from international web sites are used to drive the MACSC mechanism but the decision of dynamic cache size tuning is based on successive small data samples as it happens in real-life applications.

10.3.1 SETUP AND ENVIRONMENT

Many experiments were carried out with different pre-collected data traces. The setup and environment of the experiments are similar to that for the verification exercise (e.g. Figure 6.4 discussed in chapter 6). Wherever it is possible the same pre-collected data traces are used to make sure that the same behavior occurs, for

example, EPA-HTTP (EPA WWW server located at Research Triangle Park, NC.), SDSC-HTTP (San Diego Supercomputer Center), and Calgary-HTTP (University of Calgary, Alberta Canada) [SIGCOMM]. The ranker in each model ranks the collected URL in each cycle in order to provide the necessary information for the popularity ratio calculations.

10.3.2 EXPERIMENTAL RESULTS

The results are shown in the following figures for demonstration purposes. In the Figure 10.2, a comparison is given with the results of the FCS, MACSC(PE), MACSC(M^3RT) and MACSC(F-PE) algorithms. It shows that both the MACSC(PE) algorithm and the MACSC(F-PE) algorithm can maintain the given hit ratio. The MACSC(F-PE) has slightly better performance than MACSC(PE), but the MACSC(M^3RT) performance is worse (even worse than the FCS, which is used as a control for comparison purposes). The reason is the sampling size problem. The M^3RT uses *f* samples to calculate the mean and standard deviation of the data objects. The range of *f* is between 9 and 16. Since the ranker needs to re-calculate the mean and standard deviation of data objects popularity profile in each cycle, the small range of *f* does not provide sufficient information to make any impact. As a result it cannot return accurate predictions for the MACSC to compute correct cache adjustment.

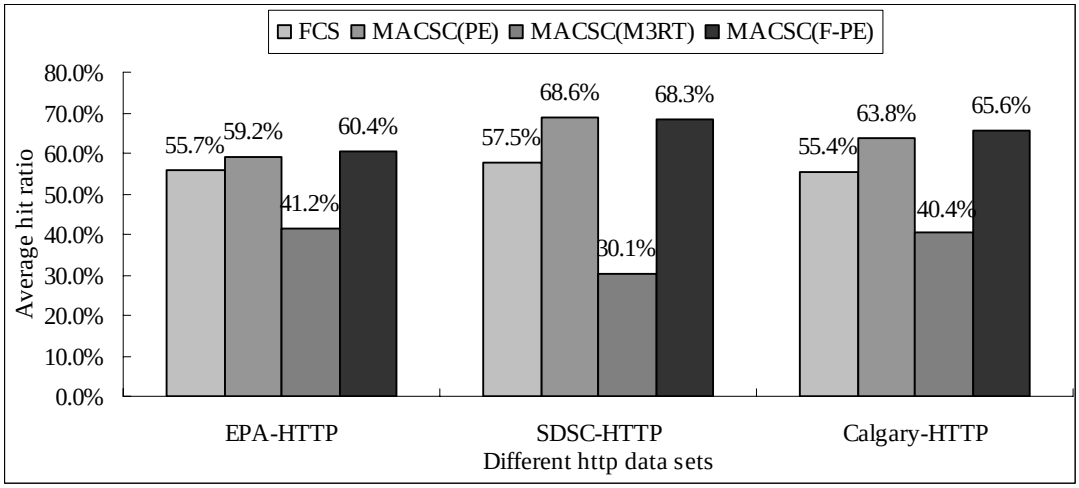


Figure 10.2. The Performance comparison of different algorithms for different data traces

Figure 10.3 and 10.4 show the changes of hit ratio and the magnified view of the changes of hit ratio with the EPA-HTTP trace respectively. The figures show that the MACSC(PE) and MACSC(F-PE) maintain the given hit ratio consistently independent of the standard deviation perturbations in the data object popularity profile, but the FCS and MACSC(M^3RT) tuners cannot. Figure 10.5 provides some evidence for this phenomenon by showing the memory usage by the different tuners. It shows that the MACSC(PE) and MACSC(F-PE) adjust the cache size dynamically according to the standard deviation at the time while the MACSC(M^3RT) does not. This is due to the fact that the calculation of the standard deviation by the M^3RT technique cannot satisfy the requirements for the ranker mechanism. For example, Figure 10.5 shows how the standard deviation calculated by M^3RT is nearly equal throughout the whole experiment. Figure 10.6, 10.7, 10.8 and 10.9 are the results for the experiment with the SDSC-HTTP and Calgary-HTTP traces respectively.

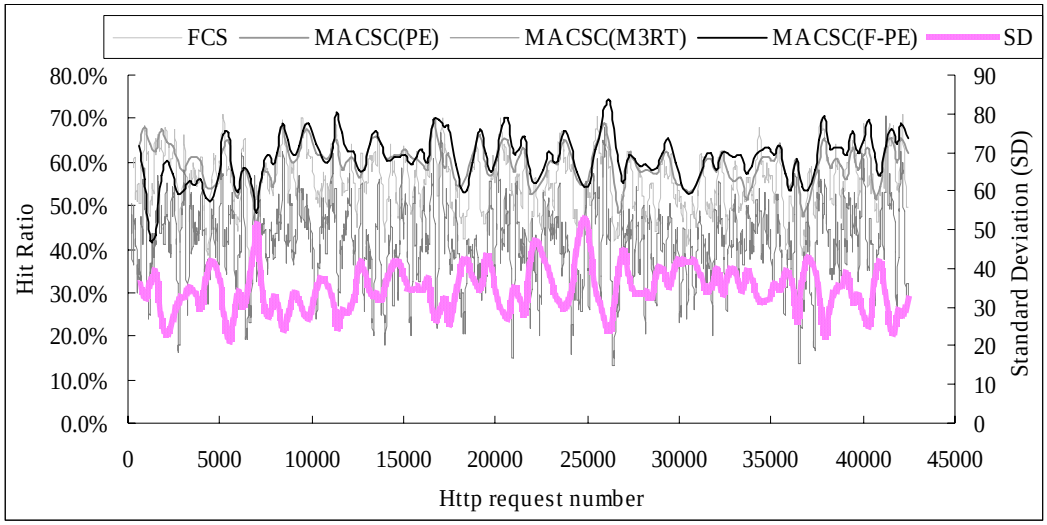


Figure 10.3 The change of hit ratios by different algorithms with the EPA-HTTP data trace

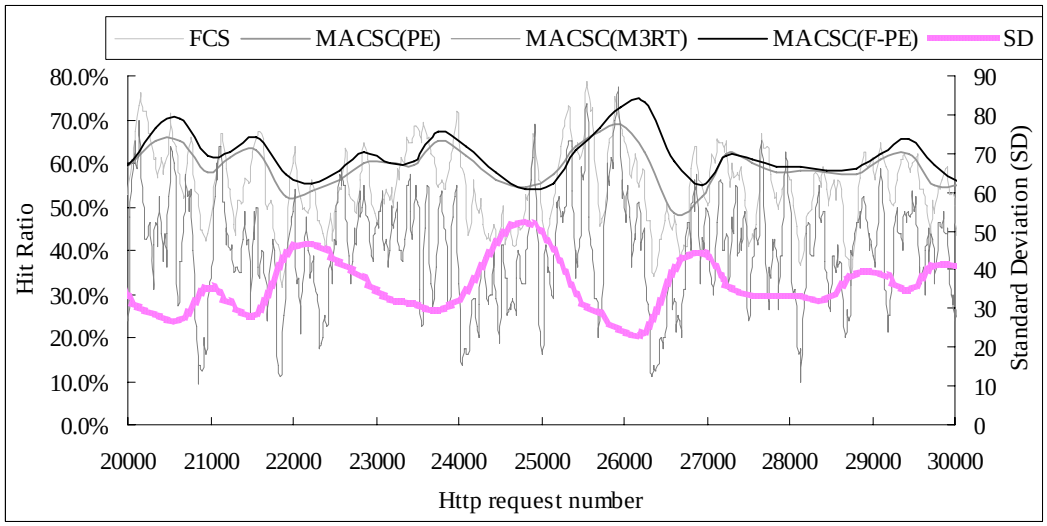


Figure 10.4 The magnified view of the change of hit ratios by different algorithms with the EPA-HTTP data trace

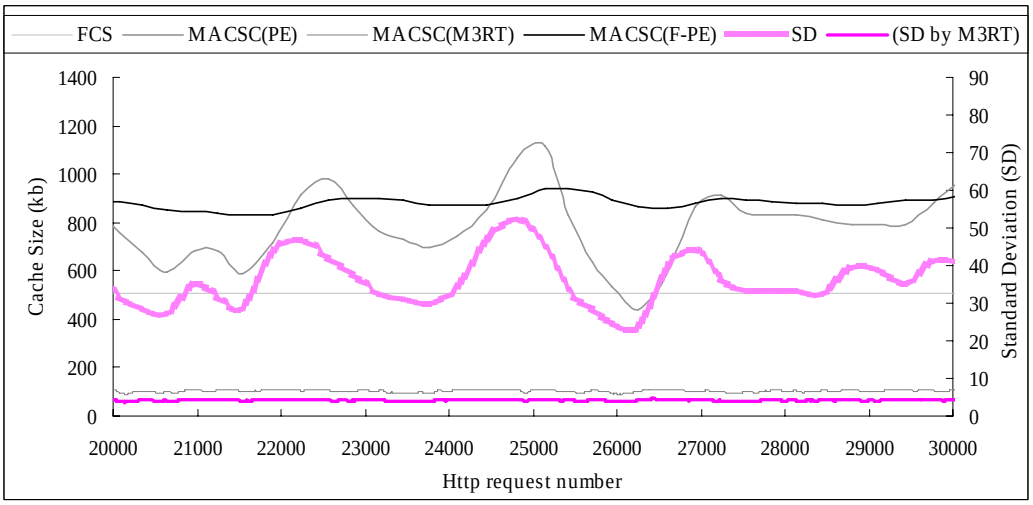


Figure 10.5 The change of the memory usage by different algorithm with the EPA-HTTP data trace

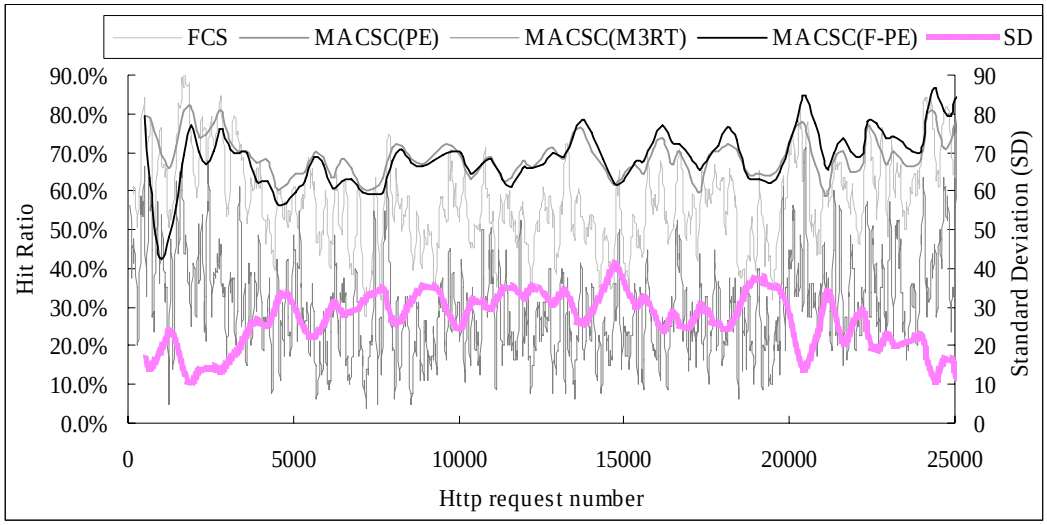


Figure 10.6 The change of hit ratios by different algorithms with the SDSC-HTTP data trace

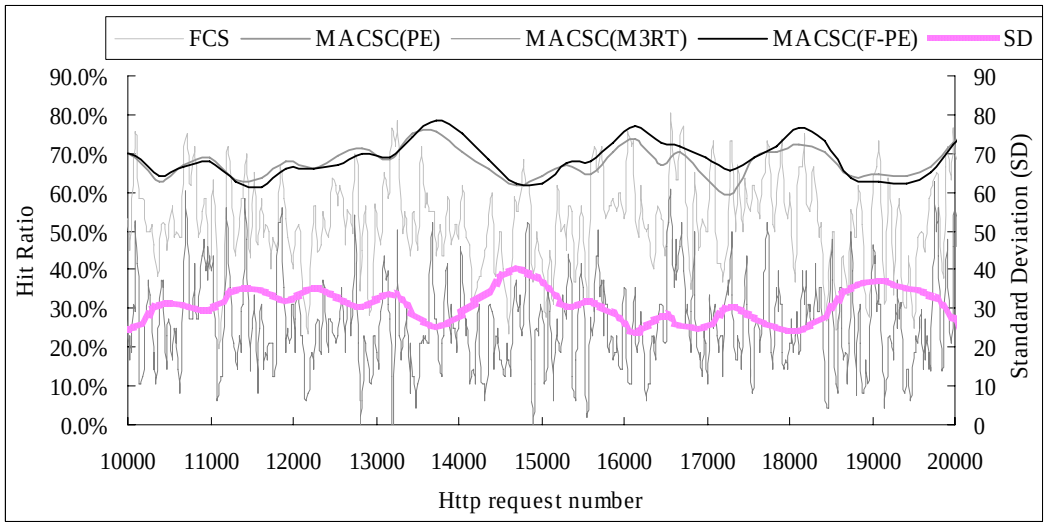


Figure 10.7 The magnified view of the change of hit ratios by different algorithms with the SDSC-HTTP data trace

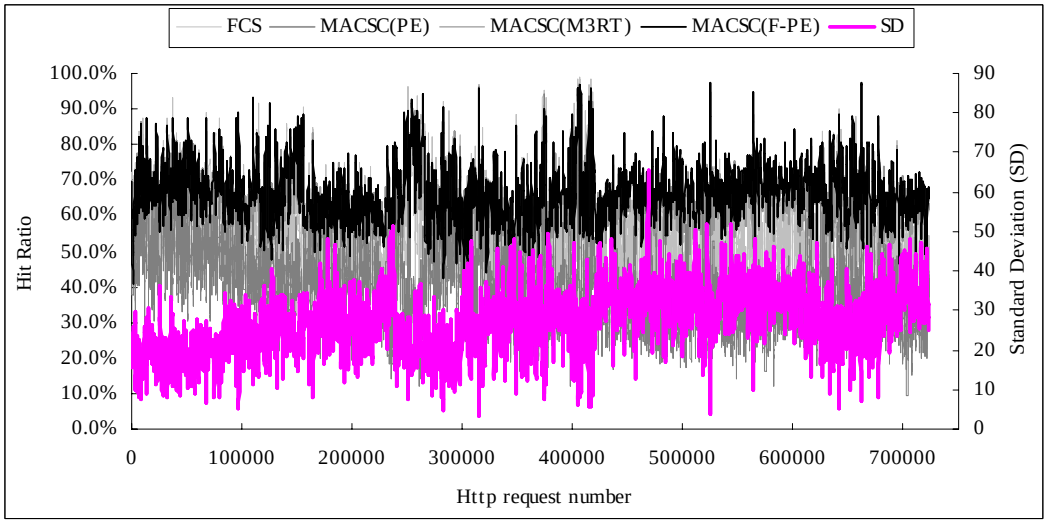


Figure 10.8 The changes in hit ratios by different algorithms with the Calgary-HTTP data trace

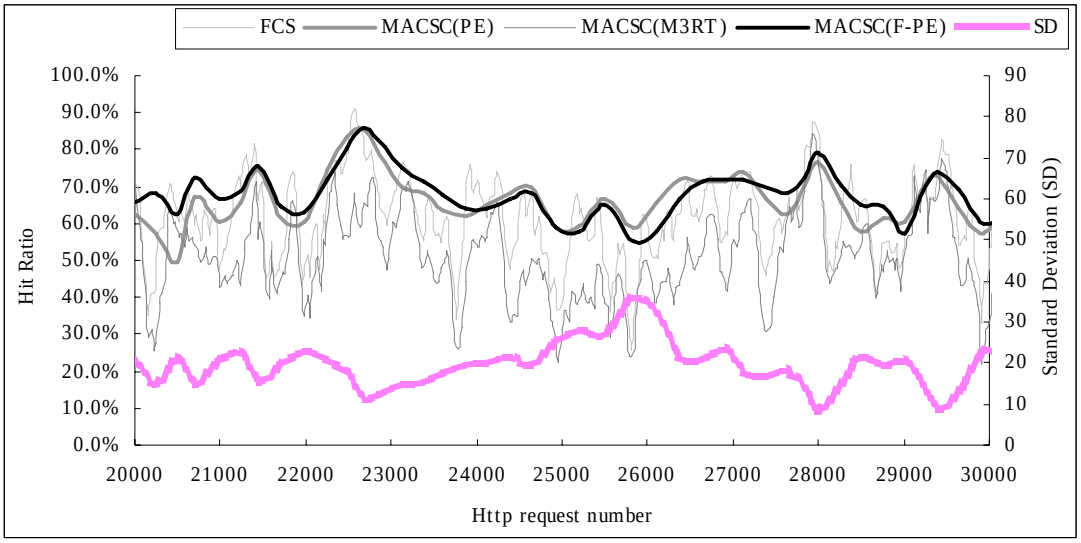


Figure 10.9 The magnified view of changes in hit ratios by different algorithms with the Calgary-HTTP data trace

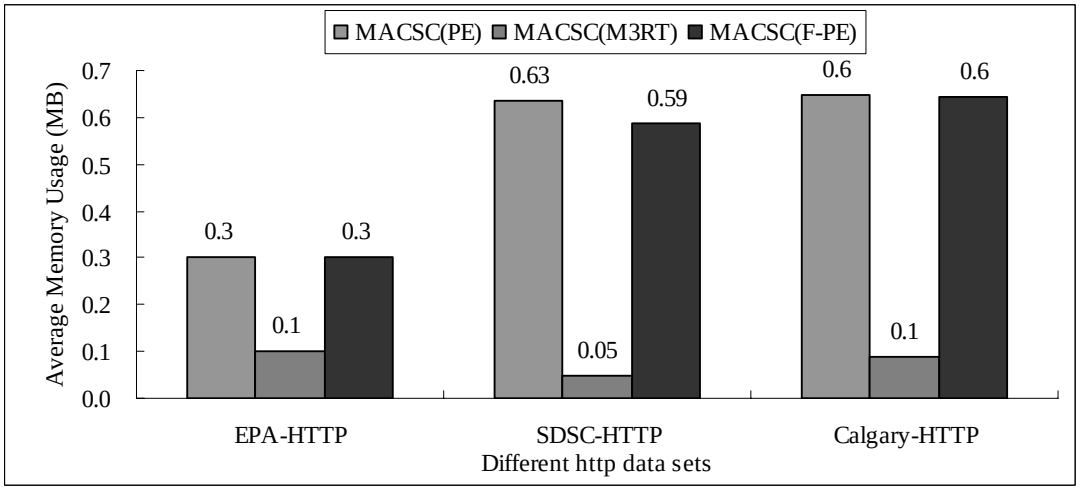


Figure 10.10 The memory usage by different algorithms with different data traces

Figure 10.10 shows the memory usage by different tuners with different pre-collected data traces. Memory usage by MACSC(F-PE) is similar to but lower than MACSC(PE). The capability of hit-ratio maintenance of MACSC(F-PE) is better than the MACSC(PE) in general. It can be concluded form the validation data that MACSC(F-PE) is the best MACSC implementation followed by MACSC(PE).

10.4 PERFORMACE OF THE RTPD/MACSC APPROACH

The core of MACSC(PE) is the "$\sqrt{N}-equation$", which can be expressed in the form of $E\mu = k\delta_{\bar{x}} = k(\delta_x/\sqrt{N})$. Another more useful form is $N = (\frac{k\delta_x}{E\mu})^2$ because it can be changed to the $N = (\frac{ks_x}{Ex})^2$ form, where $\bar{x}$ and s_x are the mean and standard deviation of a sample of size n. It is a fact that n may not be equal to N, which is governed by the tolerance criteria, namely, E and k that connote the same error [Chis92]. The repetitive process that expands n until it satisfies the $n \geq N$ condition is called the point-estimate (PE) technique. This statistical technique is independent of the waveform because it is based on the *Central Limit Theorem*. If one can evaluate the n value in every PE cycle correctly, the convergence to N can be quickened tremendously. One way of doing this is to make use of the RTPD mechanism so that the MACSC operation can make use of the detected result by the RTPD to estimate n anew in every PE cycle. This capability of using different n values under different conditions is called "reconfiguration" in the MACSC context, as demonstrated by the RTPD/MACSC(PE) solution. The MACSC reconfiguration process on the fly involves the following:

1. Calibration: There should a calibration of traffic pattern versus n as shown by Figure 10.11.
2. Real-time traffic pattern detection (RTPD): The mechanism should be able to identify a traffic pattern (e.g. heavy-tailed) quickly.
3. Reconfiguration on the fly: The MACSC adjusts the n value with respect to the detected traffic pattern and proactively quickens the convergence to N.

Figure 10.12 and Figure 10.13 demonstrate how the RTPD/MACSC(PE) solution

performs better than the MACSC(PE) working alone for the EPA-HTTP trace:

1. The RTPD/MACSC(PE) consistently yields a higher hit ratio.

2. The presence of the RTPD mechanism does not require more memory usage.

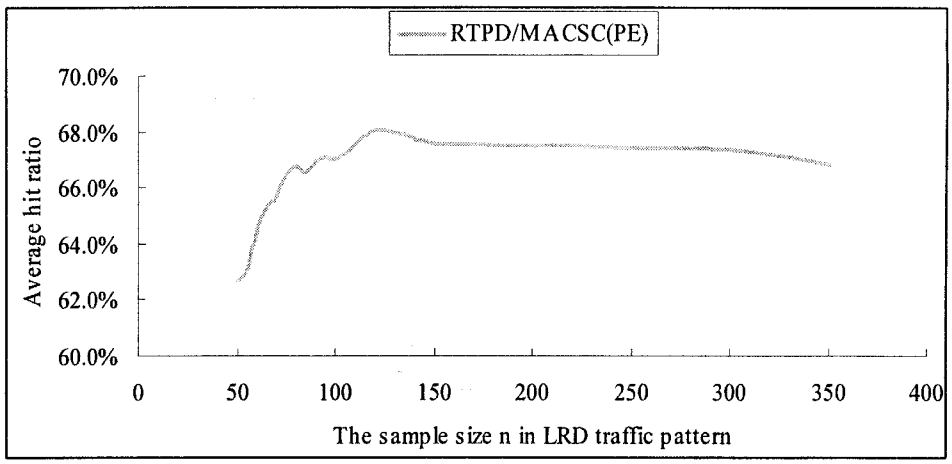


Figure 10.11 Calibration of the *n* value in LRD traffic pattern

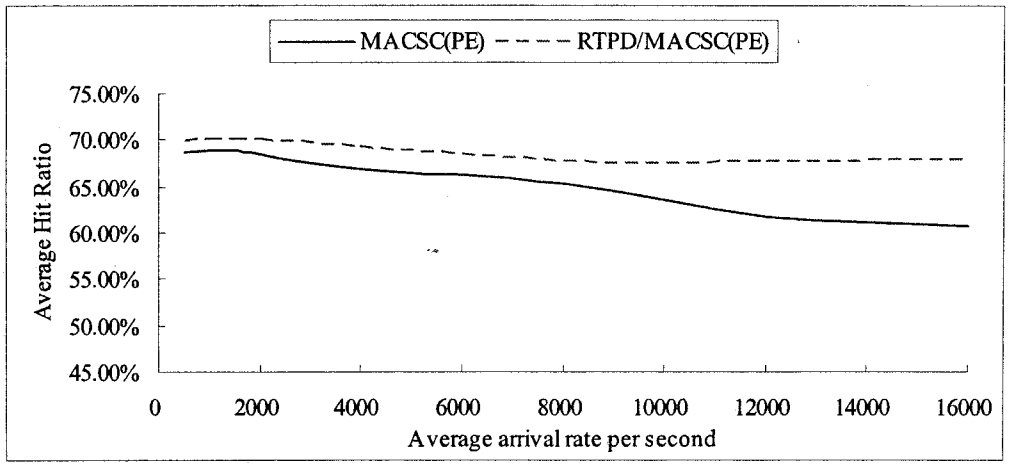


Figure 10.12 RTPD/MACSC(PE) yields higher hit ratio in general

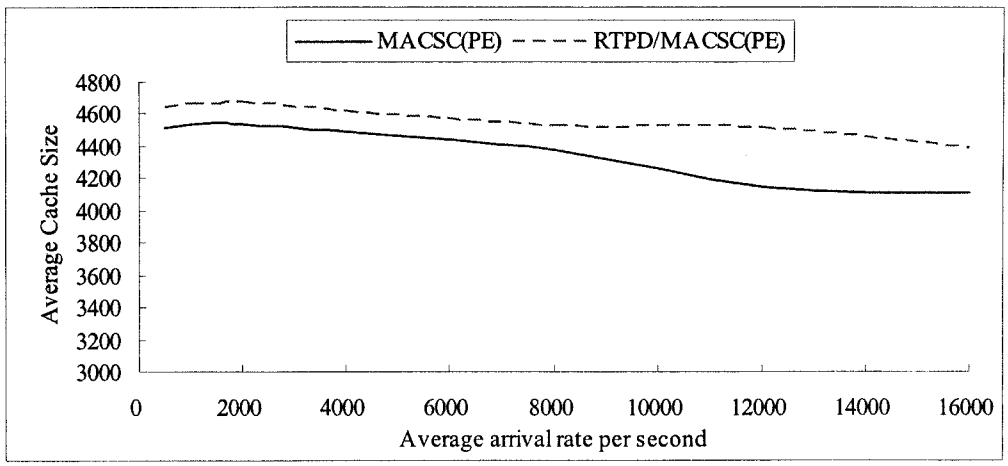


Figure 10.13 RTPD/MACSC(PE) does not consume excessive caching memory

10.5 CONNECTIVE SUMMARY

In this chapter the issue of validation of the three tuners, MACSC(PE), MACSC(M^3RT), and MACSC(F-PE) are addressed. From the experimental results the following are observed: a) it is difficult to implement the MACSC(M^3RT) solution because there is not enough sensitivity for the CA mechanism to detect any significant changes in standard deviation in the data object popularity profile. The reason is that the CA uses only the small *flush limit* number of live data items sampled per decision cycle. The MACSC(F-PE), which makes use of the feedback advantage shown by the MACSC(M^3RT) approach, produces a higher hit ratio than the MACSC(PE) in a consistent manner. It was found that the RTPD mechanism is useful because in the RTPD/MACSC(PE) approach the MACSC main body can use the detected traffic pattern to reconfigure itself to ward off traffic pattern ill effects on the dynamic cache size tuning accuracy. In the next chapter, we discuss how the MACSC framework could contribute to mobile and time-critical applications such as telemedicine will be presented. It is important that the MACSC framework should perform equally well in both wired and wireless conditions. That is, the presence of a "wireless communication" in the information retrieval process over the Internet would not cause performance problems for the dynamic cache size tuning process.

CHAPTER 11

CONTRIBUTION TO MOBILE AND TIME-CRITICAL APPLICATIONS

The three novel dynamic cache size tuners proposed in my PhD research are an outcome of my preliminary work at the beginning of the research. The scope of the work at the outset is trying to address issues related to mobile and time-critical applications that are related [Lacroix99]. In my research telemedicine was chosen to demonstrate that the proposed MACSC conceptual framework for dynamic cache size tuning does indeed contribute to time-critical applications. The choice was made based on my previous MPhil experience. Therefore, the related issues include the following:

a. Propose a reasonable and verifiable framework to support e-diagnosis [Maciocia87], [M94].
b. Explore how the diagnostic roundtrip time (RTT) can be shortened to give more precious time for saving lives.
c. Explore how media data can be sent from a remote medic in the field accurately and quickly in a pervasive manner.
d. Investigate how medical data from a very large database can be mined for building diagnostic descriptors (DD).

e. Investigate how the "virtual doctor" or intelligent medical server (IMS) should be designed and its support.

Telemedicine and pervasive computing are related and the basis for pervasive computing considered is mobile computing. The difference is that pervasive computing should track the user's intent [Weiser91], [Hightower01] but not mobile computing, which involves both the "wired and wireless (W&W)" parts of the Internet. Since the field of telemedicine and pervasive/mobile e-diagnosis is relatively pristine, there are inevitably many issues to be explored and addressed, especially with respect to the above five issues. Since preliminary analysis in my PhD work concluded that the time required to explore and address the above five issues is beyond the time available in a single PhD thesis, I felt that focusing on the second issue (i.e. b) would be more productive within the appropriate time span. The judgment was largely based on the result of my MPhil thesis [Wu02], which addressed some issues in TCM (Traditional Chinese Medicine) data mining. At that time TCM was thought to consist of three basic areas/domains, depicted as (A), (B) and (C) in Figure 11.1. Domain (A) addressed the issue of communicating diagnostic information in a mobile or pervasive manner; (B) was concerned with dynamic caching of diagnostic information for fast response; and (C) was the background task of data-mining diagnostic information, known as the *diagnostic descriptors* (DD). These domains covered a wide range of issues and problems, and therefore the present PhD research focuses on (B).

The rationale of my MPhil research was that every TCM patient record should contain enough diagnostic information, such as image, text and prescribed drugs. The data mining exploration in domain (C) was based on only a single metric, for

example, the tongue image. The intelligent miners would associate the metric with different diseases/sicknesses. These associations are encoded into the corresponding DD, which is stored in the repertoire of popular records (RPR). This repertoire represents some prefetching operations to help the IMS because proxy B in the IMS domain (i.e. (B) in Figure 11.1) fetches part of the DD to replenish the hot data in its local cache. For example, during the "SARS (*Severe Acute Respiratory Syndrome*)" period in Hong Kong and China the DD in the RPR are mostly SARS related (provided that the IMS is located in this area), and only the "*hottest*" DD in the popular records repertoire in (C) to be fetched by proxy B. In the MPhil research, however, it was found that the virtual doctor is more useful if the diagnostic information can be returned in time to save emergency cases [Wu02]. Even for ordinary cases consistent delayed return of e-diagnostic results would not help the medic or the doctor in the field either. The proposed MACSC framework, which uses dynamic cache size tuning to maintain a given hit ratio, can shorten the e-diagnosis roundtrip time.

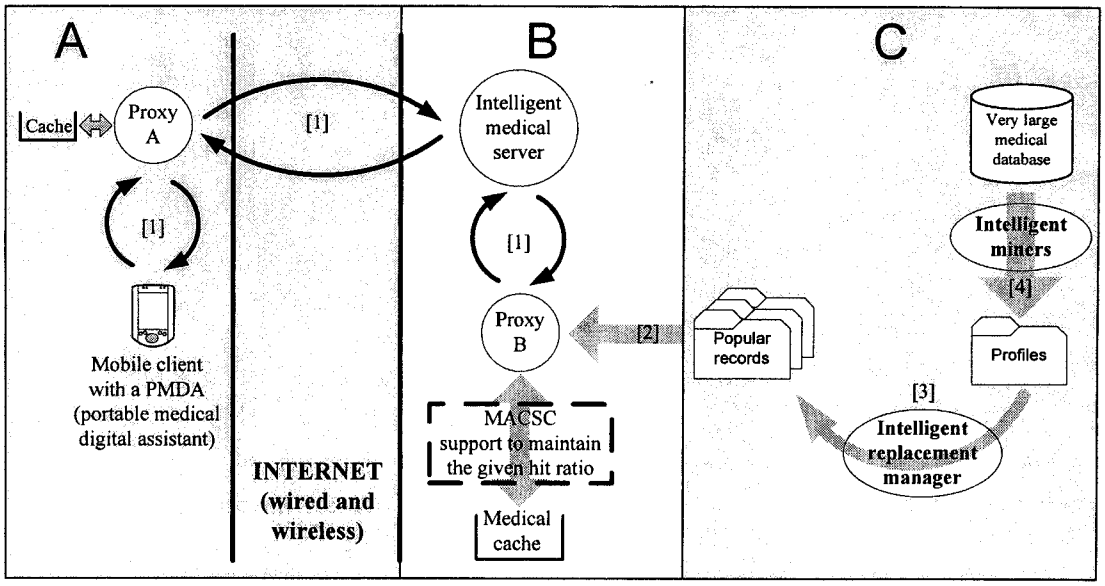


Figure 11.1 TCM Domains

11.1 COMMUNICATING DIAGNOSTIC INFORMATION OVER THE INTERNET PERVASIVELY

The area (A) in Figure 11.1 includes the following issues:

a. A proposed protocol indicated by the path marked (1) to support pervasive communication between the mobile PMDA (Pervasive Medical Digital Assistant) and the remote *intelligent medical server* (IMS) or virtual doctor in area (B).

b. An infrastructure to support effective pervasive communication, which may involve technology such as the *Blue-tooth* [Burkhardt02] and IPv6 as well as base-station technology.

c. The interface for the mobile PMDA sends relevant information to the IMS for a suggested diagnosis. This interface may be similar to Figure 11.2 in GUI design, which was used by the telemedicine research team of Dr. Allan Wong, for test and demonstration purposes. I participated in this GUI development to evaluate how effective diagnostic communication can be carried out. The roundtrip time problem in the e-diagnosis process that I observed has inspired my research in the area of dynamic cache size tuning to maintain a given hit ratio. This maintenance would shorten the e-diagnosis RTT.

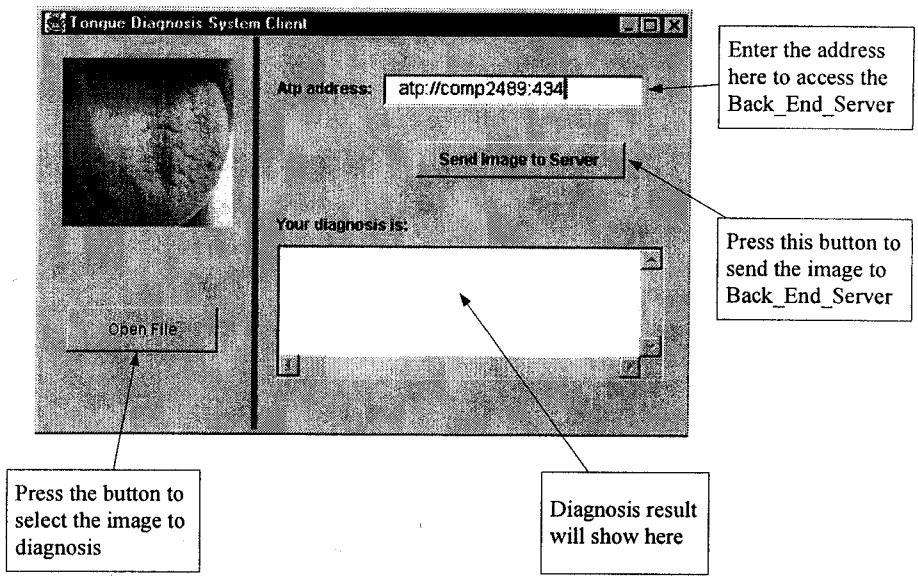


Figure 11.2 A GUI example for PMDA development and communication evaluation

11.2 CACHING DIAGNOSTIC INFORMATION FOR FASTER RESPONSE

The communication between the PMDA and IMS is end-to-end, as shown in Figure 11.3. The holder carrying the mobile PMDA could perform a diagnosis with the help of the IMS anytime and anywhere ubiquitously, especially in emergency situations. If the diagnosis is received in time, then the holder can save lives effectively. The end-to-end channel over the wireless and/or wired Internet may not be "*pure*" TCP/IP, but it would use AQM mechanisms in the relaying routers over the channel. The router AQM purpose is to detect the possibility of incipient overflow in its local buffer and to feed this possibility back to a client such as the PMDA. With feedback the client may react by cutting the transmission rate so as to prevent router overflow, and this process is called throttling [Tanenbaum96].

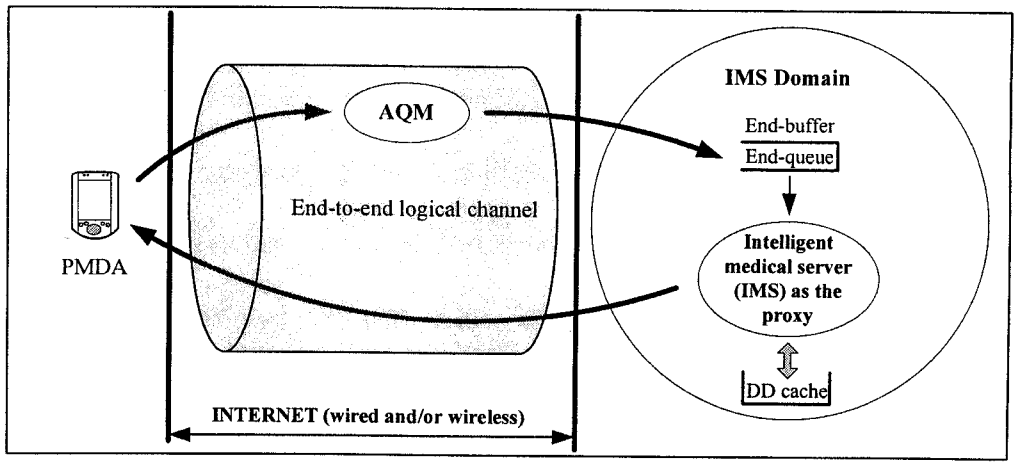


Figure 11.3 Model of PMDA –IMS communication

The AQM mechanism compensates for the deficiencies of the inherent TCP/IP congestion avoidance ability [Lakeshman96], [Lakeshman97] and is still at the exploratory stage [Ren02], [Firoiu00]. For example, the RED model, which is recommended as a candidate for Internet end-to-end overflow prevention in RFC2309 [Braden98], does not perform in a stable manner. This has led to a continuing effort to find better models (e.g. [Chris00], [Ramakrishnan99]). If the end-buffer in the IMS domain deals and receives requests from more than one PMDA, then the *end-queue* traffic may surge exponentially due to the *many-to-one* (IMS) relationship. The surge would inevitably lead to overflow, message loss and retransmission. It is common for any logical channel to have an error probability [Paxson97], for example, δ_A, which also includes the chance of overflow for the end-buffer δ_O. The average number of transmissions (ANT) to get success at the j^{th} trial can be represented conceptually by $ANT = \sum_{j=1}^{k\to\infty} j(\delta_A)^{j-1}(1-\delta_A) \approx 1/(1-\delta_A)$.

That is, elimination of the δ_O component from δ_A would make a shorter *roundtrip time* (RTT) for path (1) depicted in Figure 11.1. In fact, end-buffer overflow

elimination is a relatively new area of research and there are only a few of such experimental models in existence (e.g. [Wong02a], [Ip01]). The empirical performance of some of these models indicates that they indeed effectively stifle end-buffer overflow in a consistent manner. Yet, the publications for these models so far have not analytically differentiated the end-buffer (or end-queue) behavior with respect to different traffic patterns. After some rigorous study, it was concluded that the prevention of end-buffer overflow in the IMS domain can rely on techniques proposed by others (e.g. the PID controller [Ip01]). In this study, however, I have gained some insight into how controllers may react to different traffic patterns. The experience also provides some idea of how to identify the appropriate methodologies for finding the patterns embedded in the traffic traces.

The issue of having a high hit ratio for proxy B in Figure 11.1 is important because a high hit ratio saves I/O time and shortens the service RTT in the e-diagnosis. For example, if the average hit ratio is 0.7, then it is only 30% or 0.3 chance for Proxy B to involve the intelligent replacement manager that controls the costly search such as path (3) and (4) in Figure 11.1. If the collective average search time for the two paths were T, the speedup would be $T/(0.3T+0.7*0)$ or close to 333% (i.e. three fold).

11.3 RELATED ISSUES IN COMMUNICATING DIAGNOSTIC INFORMATION PERVASIVELY

Pervasive interaction requires special support because the interaction can switch back and forth between the wired and unwired environments [Burkhardt02]. The architecture support should come from a mixture of different but relevant

technologies. To communicate diagnostic information correctly and efficiently retransmission should be reduced. The average number of trails (ANT) to get a successful transmission as explained previously is $ANT \approx 1/(1-\sigma_A)$.

11.3.1 IMPACT OF INTERNET TRAFFIC

The Internet traffic can affect the stability and performance of the PMDA-IMS communication, which therefore should be adaptive in the sense that it would adjust its behavior in a dynamic manner as required by the traffic circumstances. From different sources of research experience, it has become clear that the end-to-end channel dynamics can affect successful Internet applications seriously [Cottrel99], [Sanghi93], [Downey99]. This is especially so when dealing with the question of how to provide the required RTT timeliness in Internet-based time-critical application cases. The main issues of RTT timeliness control in the research are:

a. Accurate interpretation/prediction of end-to-end behavior. This problem belongs to the area of Internet End-to-End Performance Measurement (IEPM) [Cotrell99].

b. Detection/interpretation of the traffic pattern anytime from the aggregate X_l^m data of a discrete stochastic process X, where m is the block size of the aggregate and l the lag, for l=1,2,3,...n in the time series [Willinger98].

For the IEPM problem an extensive study was performed for the purpose of this research, and in particular the M^{3}RT *micro* IEPM technique [Wong02b], which is an object-based version derived from the *Convergence Algorithm* (CA) [Wong01b]. The experience from the critical evaluation of the M^{3}RT technique has contributed to the improvement of the MACSC (*Model for Adaptive Cache Size Control*) for dynamic

cache control [Wong03]. The details of the improved version, which is also christened the DCA (dynamic cache adjustment) model, are published in [Wu03] and [Wu05b]; [Wu03] is a refereed conference paper. [Wu05b], and [Wong03] are a refereed journal papers.

11.3.2 PMDA (PORTABLE/MOBILE MEDICAL DIGITAL ASSISTANT)

This is a conceptual hand-held gadget and its final form is still an issue to be explored but not in the scope of this thesis. It, however, would support capturing and organizing multimedia data for diagnostic purposes. The organized data is sent to the IMS, which returns the maximum likelihood of the health problem(s) and the possible treatment suggestions. Under emergency conditions in a remote area, this helps save lives because the person, who is carrying the PMDA, might not necessarily be a highly trained medic. Some preliminary investigation has been done in the beginning of my PhD research to explore the issue of the type of multimedia data suitable for the PMDA operation. From the diagnosis point of view there are usually five categories [Maciocia87] of information suitable for use in the process, namely, visual (visual features and images (e.g. X-ray image), "smell + sound", illness history, pulsation, and laboratory report. For remote diagnosis, which is a part of telemedicine [Wong97], image has received more attention than other categories. In fact, in my MPhil thesis, which deals with fast data mining for telemedicine, the image is the key medium. The final choice of my PhD investigation, however, is how to help effective PMDA-IMS communication. This led to the proposal of the generic MACSC conceptual framework that maintains a given hit ratio by the technique of dynamic cache size tuning.

Figure 11.4 summarizes the framework of the architecture needed to support pervasive PMDA-IMS interaction. The wireless communication needs the support of base stations, which are the front-ends of the mobile support system (MSS). In Figure 11.4 the client can be the PMDA functional unit and the supporting components in the architecture include: socket, TCP/IP, IP and PPP (Shiva point-to-point protocol). In the PhD research, however, the verification and validation of the proposed MACSC framework is by simulation with different pre-collected real-le traces such as EPA-HTTP, SDSC-HTTP...etc.[Wu04], [Wu05a].

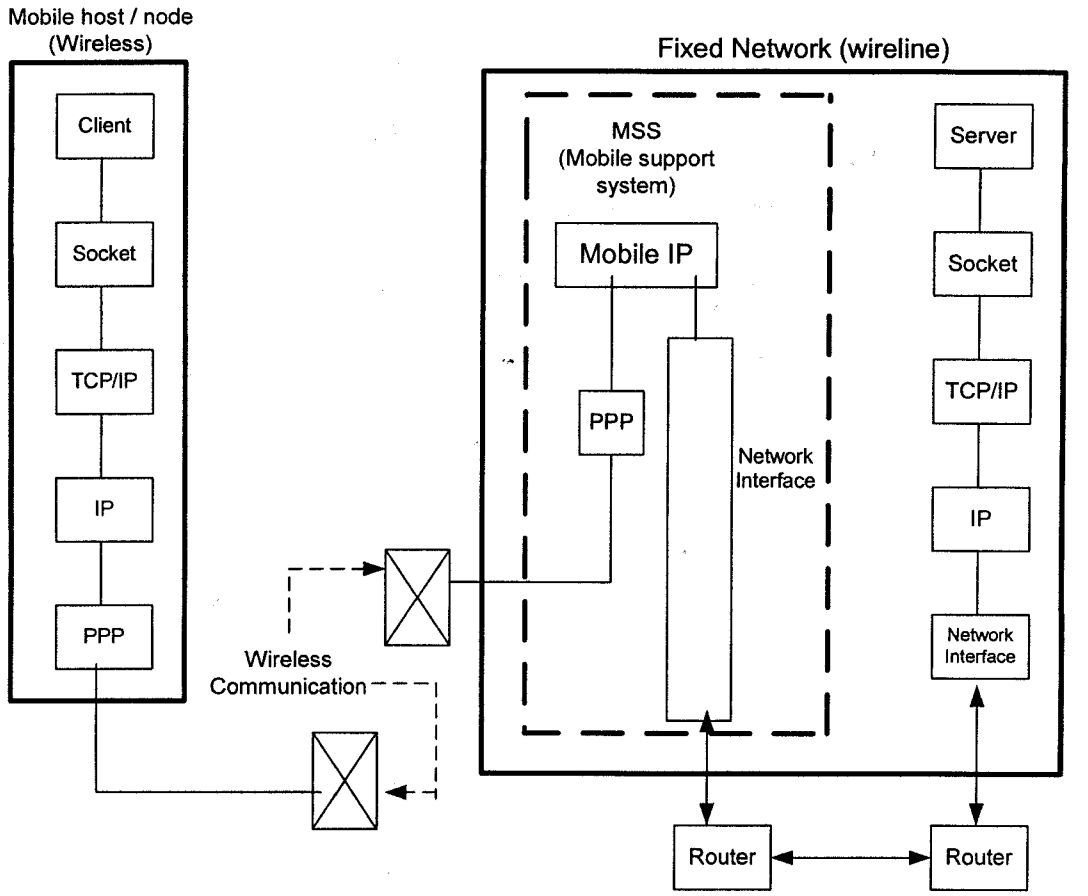


Figure 11.4 Architectural supports for pervasive PMDA-IMS interaction overview

11.4 APPLYING THE RTPD/MACSC APPROACH TO SHORTEN THE ROUNDTRIP TIME FOR MOBILE DIAGNOSTIC INFORMATION RETRIEVAL IN TELEMEDICNE

Mobility and pervasiveness are important attributes for contemporary distributed computing [Garlan02]. With mobility the client, which is a small-form-factor (SFF) device (e.g. PDA, mobile phone, PMDA), can be powered off after its user has made the request to its surrogate. Usually a surrogate or gateway is a node in the wired part of the Internet that is assigned to temporarily assist a client. For example, the (B) entity in Figure 11.1 can run inside a surrogate. When the SFF device is powered on again it can receive the service result for the previous request. A SFF client moves from one wireless cell to another, and with the help of the local base station it may continuously interact with the same remote server (e.g. the (B) entity in Figure 11.1). Client mobility plus the *smart space* [Weiser91], creates the basis for the pervasive/mobile computing environment needed in telemedicine. A smart space is conceptually a very well supported wireless cell. The supporting infrastructure may even track the user's intent. For example, after a client has made a service request to a remote server it might power off and move to another smart space. The supporting infrastructure then tracks user's movement (i.e. intent). When the user has arrived at a new smart space the service result, which has already been rendered immediately usable, is already waiting there. The tracking capability depends on location-sensing or location-aware technology, which finds the whereabouts of a client (and user) in a ubiquitous manner.

Figure 11.5 shows how a client (SFF device) carried by a human user in a smart space interacts with the high-speed wired Internet. The location-aware capability

helps the client hook up with a surrogate. If the surrogate in the wired part of the high-speed Internet is a proxy server that supports information retrieval, then it will handle the data object requests by the client first. If the data object cannot be found in its local cache, then *cyber foraging* is necessary. In cyber foraging the proxy (e.g. the virtual doctor) enlists help from the other entities (e.g. intelligent miners in Figure 11.5), which may reside in other nodes in the infrastructure, until the required data object is found and retrieved. In this thesis the average service roundtrip time (RTT) between the client and the proxy is called the *first leg* (i.e.RTT_1) and that between the proxy and the remote data source the *second* (i.e.RTT_2). The hit ratio for the proxy cache is important for reasonable pervasive information retrieval time. This can be demonstrated by assuming the following: hit ratio σ =0.6 (i.e. 60%) and RTT_2=*10* RTT_1. Then, the speedup of the service response due to caching for this case is represented by $S = \frac{(RTT_1 + RTT_2)}{(RTT_1 + 0.4RTT_2)} = \frac{11}{(1+0.4*10)} = \frac{11}{5} = 2.2$. A hit ratio σ =0.6 (i.e. a miss ratio of (1- σ)) obviates 60% of the delay by the second leg (i.e. RTT_2). Clearly, a high σ value is always desired for efficient wireless pervasive information retrieval to make a client/user happy. The proposed MACSC framework obviates the second-leg delay by maintaining a high hit cache ratio. In this case, the cache should hold the popular/hot diagnostic information.

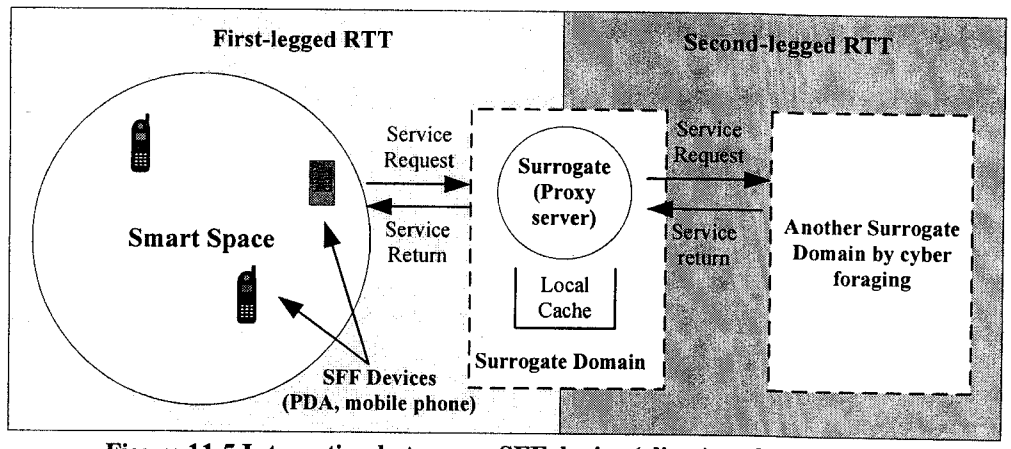


Figure 11.5 Interaction between a SFF device (client) and a surrogate

In fact, a surrogate (e.g. virtual doctor or intelligent medical server) may interact with different SFF clients simultaneously. The number of clients is usually tied to the *mass transit traffic* pattern [Malla03] through the smart space. These clients may seasonally request some data objects more than others. For example, during the SARS epidemic period in Hong Kong the diagnostic information may exclusively be SARS related). The preference of data object/information over an interval by users is a Zipf-like behavior [Breslau99]. Therefore, the cache hit ratio at any time is determined instantaneously by two driving forces: a) the information retrieval request traffic and b) the users' preference for certain data objects. In this thesis we propose to use the MACSC (*Model for Adaptive Cache Size Control*) concept [Wong03] as the basis to maintain the given hit ratio on the fly. This deals with the ever-changing user preference, which shifts the relative popularity profile for a given set of data objects. To neutralize the ill effects caused by the traffic pattern changes due to mass transit the RTPD capability is necessary. Figure 11.6, which is reproduced from the [Lin04], [Wong03] experience, clearly shows how the hit ratio by the MACSC working alone (without RTPD support) is affected by different traffic patterns. The hit ratio drops seriously for the LRD (long-range

dependence) traffic pattern as the average arrival rate increases, as compared to that of the SRD (short-range dependence) pattern [Molnár99]. With RTPD support the MACSC self-tunes to compensate for the traffic pattern changes. The "MACSC plus RTPD" combination is called the RTPD/MACSC approach. Figure 11.7 depicts how the MACSC framework helps shorten the RTT between a mobile client and a proxy server in the surrogate node.

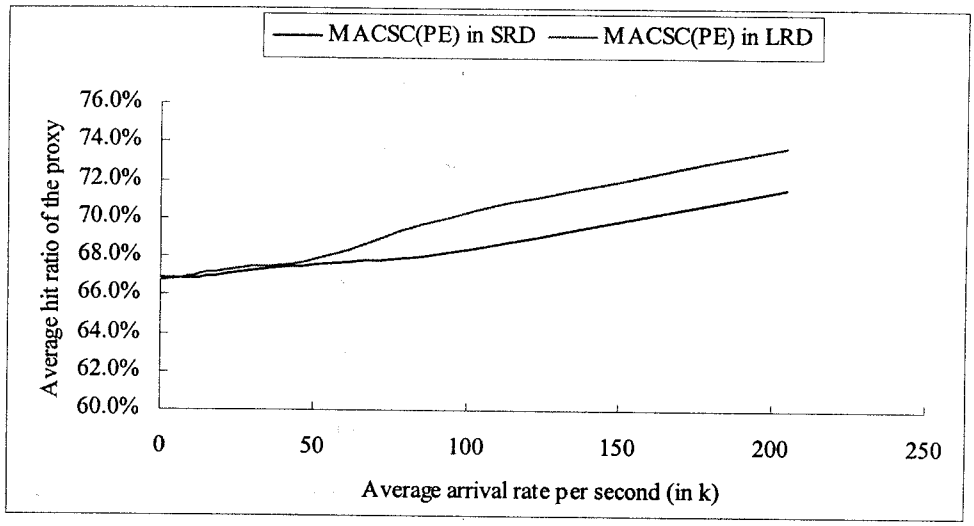


Figure 11.6 MACSC(PE) hit ratios for the LRD and SRD traffic patterns

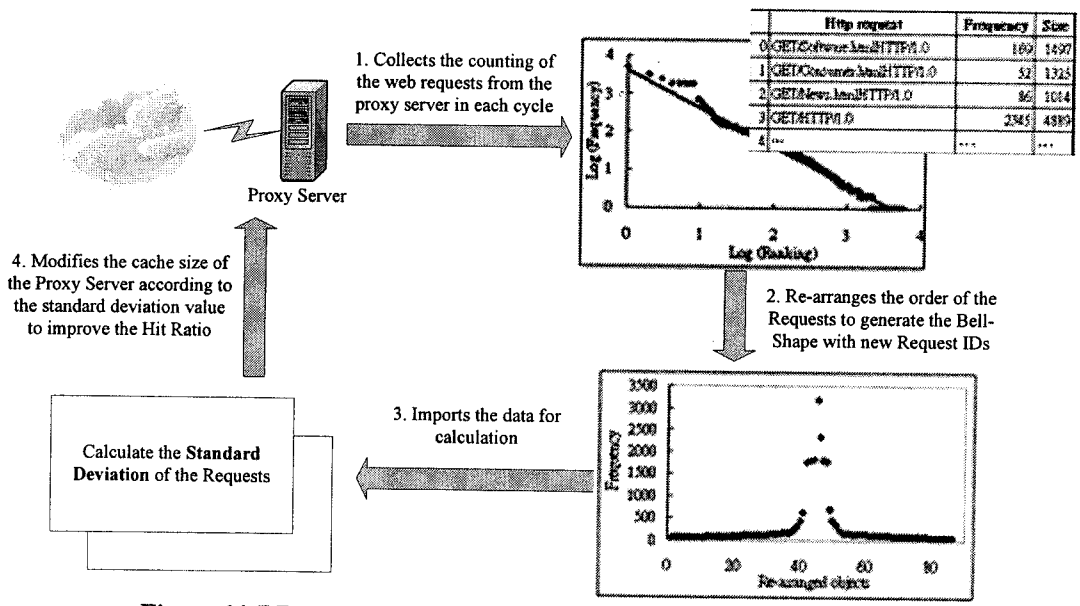


Figure 11.7 Real MACSC implementation scenario for a proxy server

Figure 11.8 shows how the RTPD/MACSC approach shortens the information retrieval roundtrip time in pervasive computing. The e-diagnosis process in the telemedicine environment shown in Figure 11.1 is essentially similar. The SFF client (i.e. PDMA) makes its request in the wireless smart space to the surrogate (i.e. virtual doctor or intelligent medical server), which is also serving similar requests from the wired Internet domain. All request streams merge at the surrogate service access point, and the merged traffic embeds different patterns (e.g. SRD and LRD) at different times. The traffic pattern depends on many factors, such as the mass transit traffic (MTT) through the smart space (e.g. a large train station or airport). The number of SFF clients in the smart space may increase or decrease suddenly in proportion to MTT. In the wired Internet infrastructure the surrogate is just another proxy server. During peak-demand seasons the average *inter-arrival times* (IAT) among the different requests in the merged traffic can be very short. The combined effect of the merged traffic and the request demand bursts of small standard deviations can affect the cache hit ratio of the server in the surrogate node in various ways. The MACSC supported by the RTPD capability tunes the cache size to compensate for problems due to the combined effect so that the given hit ratio is consistently maintained as the minimum in a dynamic manner.

From the perspective of the SFF-1 (i.e. small-form-device no. 1) client the first leg of the service roundtrip time is between itself and the surrogate. The second leg of the service RTT is between the surrogate and the remote server that returns the data objects found through cyber foraging. If the RTPD/MACSC approach could maintain the given hit ratio, then the second leg would be obviated to consistently yield a much shorter information retrieval roundtrip time. The present strategy of hit

ratio maintenance by the RTPD/MACSC approach is achieved by choosing the most appropriate initial sample size n for the core point-estimate or PE calculation in the tuning process. The meaning of n is illustrated in the following PE iterative process that strives to satisfy the $n \geq N$ criterion:

1. The initial n=60 data retrieval requests have yielded 15 and 9 for $\bar{x}$ and s_x respectively for the PD.
2. The given tolerance is 2 standard deviations (i.e. k=2 or 95.4%), and the fractional error E is therefore equal to 4.6% (E=0.046); both E and k connote the same error.
3. The minimum N estimated from $N = (\frac{k\delta_x}{E\lambda})^2$ is $N = (\frac{2*9}{0.046*15})^2 \approx 680$.

The value $N \approx 680$ implies that the initial sample size n = 60 is insufficient. To rectify the problem it is necessary for the RTPD/MACSC approach to collect 60 more IAT samples and re-calculate $\bar{x}$ and s_x from the total of 120 samples (i.e. n=120 for the 2nd trial). The process repeats continuously with 60 new additional samples each time until $n \geq N$ is satisfied. The work in [Lin04] shows the n values for different traffic patterns can be calibrated. This means that the RTPD/MACSC approach can choose the n value appropriate for the traffic pattern detected by the RTPD entity. The present research follows this previous experience, and in the context of the MACSC framework, this is real-time system reconfiguration.

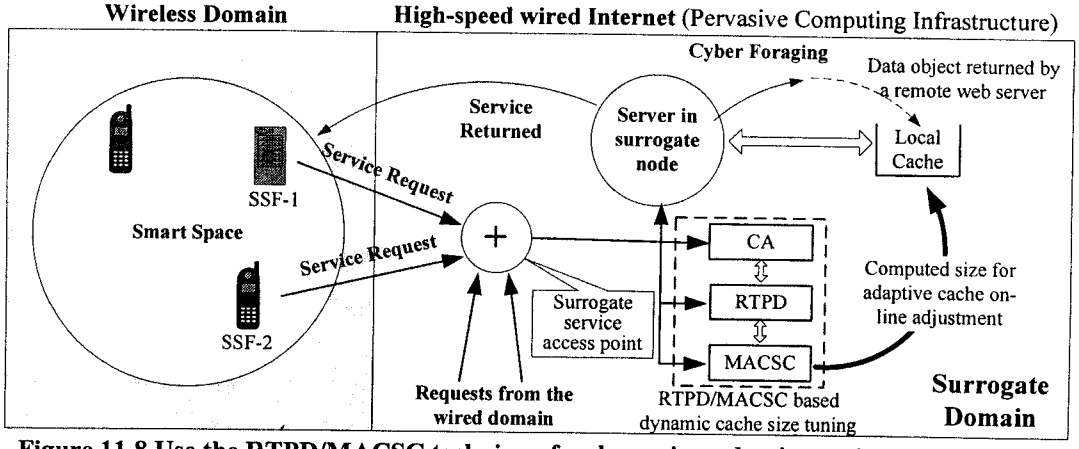


Figure 11.8 Use the RTPD/MACSC technique for dynamic cache size tuning to shorten the information retrieval RTT in a pervasive computing environment

11.4.1 SIMULATIONS

The aim of the simulations is to verify that the RTPD/MACSC approach can indeed maintain the given hit ratio (e.g. one standard deviation or 68.3%) for the surrogate under different traffic conditions. This is essentially a need in e-diagnosis. The verification by simulation is carried out with the Aglets mobile agent platform [Aridor98], which is chosen for its time-proving stability and rich user experience [Wong03]. The setup for the experiment is illustrated in Figure 11.9. The driver agent (agile applet) and the surrogate proxy agent represent the interacting SFF client, namely the PDMA and the virtual doctor in the surrogate respectively. The clock uses the chosen traffic pattern, which is either a chosen distribution or a downloaded wireless trace to simulate the IAT of the merged traffic for the surrogate proxy. The requested data objects, however, are interpolated from the chosen PD. The X axis of the PD represents the integer identifiers of the data objects, and the Y axis represents the probabilities of the integer identifiers. The sequencer generates a random number with which the identifier is interpolated from the X axis. Therefore, every request contributes two attributes, namely, the data object identifier and the

IAT. The surrogate cache is implemented as a TCS (*twin cache system*) [Aggarwal99], which is designed to filter the "*one-timers*" to make the hot data in the cache more concentrated. The execution times between the MACSC working alone and the RTPD/MACSC approach are measured by the Intel® VTune™ Performance Analyzer [VTune] and compared. Conceptually the execution times of the two MACSC versions should be comparable. The additions of the CA and RTPD entities/capabilities to the MACSC process (i.e. the RTPD/MACSC approach) should not lengthen the latter's execution time significantly for two reasons: a) the execution time of the MACSC working alone is longer than those for the CA and RTPD actions, and b) the MACSC, the CA and the RTPD entities are running in parallel. The repertoire of the 40,000 data objects ("data source" indicated in Figure 11.9) simulates the effect of cyber foraging.

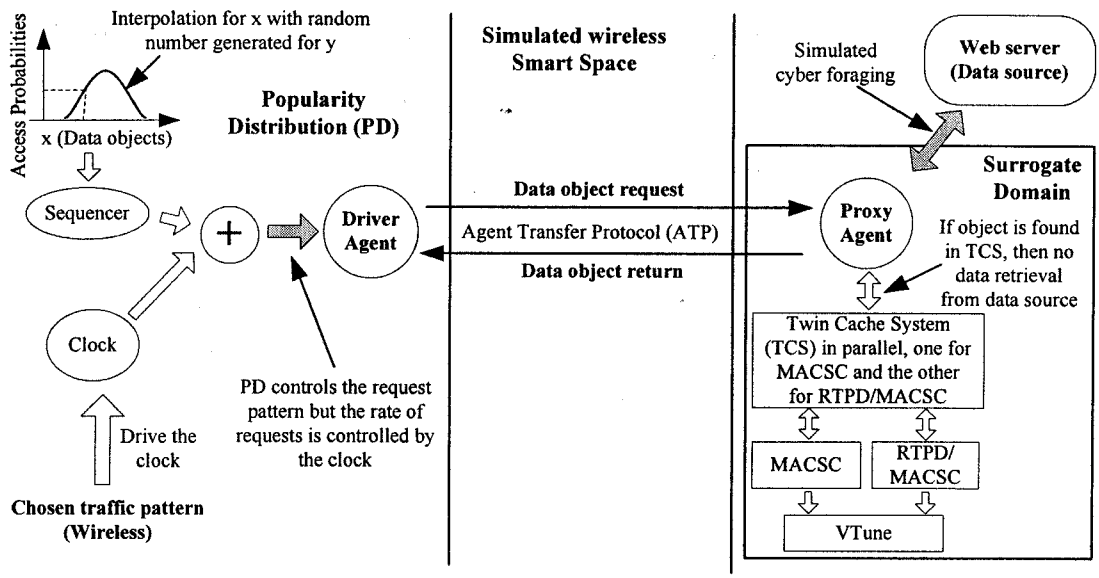


Figure 11.9 Setup to verify the RTPD-based MACSC (i.e. MACSC/RTPD) for maintaining the given hit ratio for the surrogate's cache in e-diagnosis in mobile telemedicine

Many experiments with different wireless traces were carried out, and they all indicate that both the RTPD/MACSC approach and the MACSC(PE) tuner (without

[blank page]

RTPD support) indeed maintain the given hit ratio of 68.3% (one standard deviation) equally well under various conditions. The RTPD/MACSC approach, however, consistently uses less memory than the MACSC(PE) tuner and has less chance to cause deleterious effects. The results presented in this thesis are produced with the wireless LAN trace from ACM SIGCOMM'01 [Balachandran02]. This trace, which is widely-used used by different researchers, records the wireless traffic in the ACM SIGCOMM'01 conference over three days at U.C. San Diego in August 2001. The trace captures roughly 300,000 cases of traffic flows from the 195 users that consumed 4.6 GB of bandwidth. The wireless LAN was an IEEE 802.11b [IEEE99] network installed in a large auditorium for different conference sessions. Table 11.1 summarizes the trace, and AP means access point. The experiment that yielded the result for Figure 11.9 made use of the http data retrieval frequencies. About 57.5% of the total bytes (2.645 GB) or 45.8% of the total traffic are related to http data requests. The experiment that yielded the results presented in Figure 11.10 and Figure 11.11 used the IAT in the trace as the basis.

Environment parameter	Values
Number of wireless users	195
Maximum users at an AP	32
Total hours of trace	52
Total bytes transmitted	4.6 GB
Total flow	298995
Peak throughput at an AP	3.2 Mbps

Table 11.1 Summary of the ACM SIGCOMM'01 trace

The experimental results in Figure 11.10 show the changes in the hit ratios by MACSC(PE) and RTPD/MACSC(PE) respectively over time. The standard deviation (SD) fluctuations indicate the periods in which small clusters of objects became more popular than others. A high standard deviation means that most of the data objects have similar popularity. The changes in the Hurst value imply that the

traffic patterns in the trace are mixed. The traffic pattern changes from LRD (for H>0.5) to SRD (for H<0.5) and vice versa abruptly. Specifically the MACSC intended to yield a higher hit ratio than the given minimum of 68.3% and RTPD/MACSC(PE). Our careful analysis, however, reveals that this is the result of inaccurate PD estimation by the MACSC's PE calculation. The inaccuracy produces a spuriously higher PR ratio and thus a corresponding cache adjustment size to yield a higher hit ratio. There is more PE accuracy for the RTPD/MACSC(PE) because the RTPD detects the traffic pattern with which the tuner adaptively adjusts the PE calculation accordingly. More PE accuracy makes the RTPD/MACSC(PE) use less cache memory and comparatively yields a lower hit ratio than the MACSC working alone. Since the MACSC uses more memory than necessary, it may lead to suspensions of some tasks in the proxy server and poor system throughput as a deleterious effect.

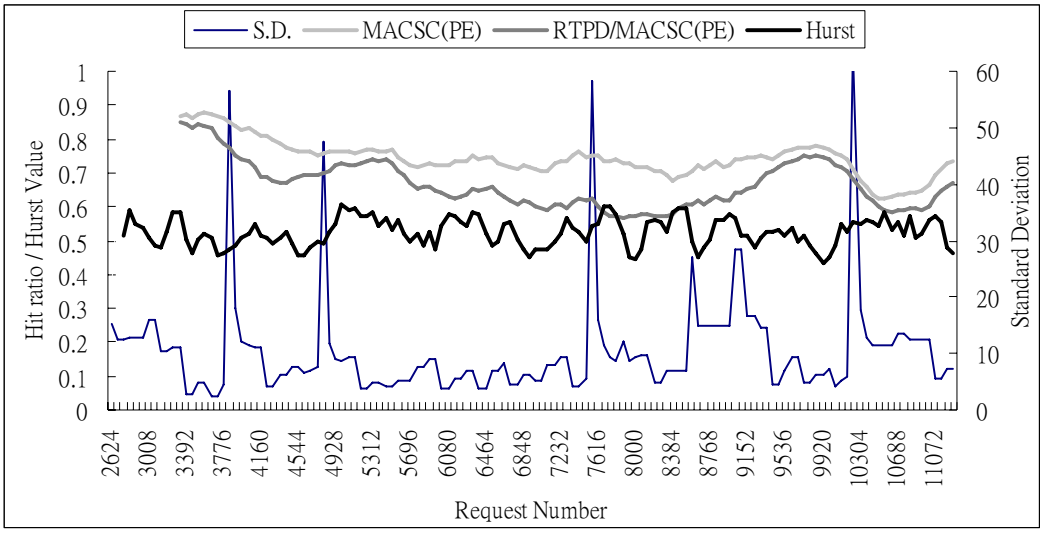


Figure 11.10 Hit ratios by MACSC(PE) and RTPD/MACSC(PE), SD, and H value are compared for the wireless ACM SIGCOMM'01 trace (58% LRD and 42% SRD)

The experimental data in Figure 11.11 was obtained by changing the IAT values in the ACM SIGCOMM'01 trace. The aim is to empirically show that the PE

inaccuracy indeed worsens when the average IAT rate rises. This is natural because a high average IAT rate can cause more request loss. The MACSC hit ratio can superficially improve with the increased IAT rate and PE inaccuracy. This is shown by Figure 11.11 in which both the MACSC hit ratio and the cache memory usage go up with the IAT rates. For example, the MACSC produced roughly a 5% better hit ratio better than RTPD/MACSC with the IAT rate of 1200k/s. But, the memory usage surges by 50% compared with the RTPD/MACSC with the same IAT rate. This indicates that the RTPD/MACSC is more efficient than the MACSC working alone because it can maintain the given hit ratio with less memory usage in mobile diagnostic information retrievals by SFF/PDMA clients.

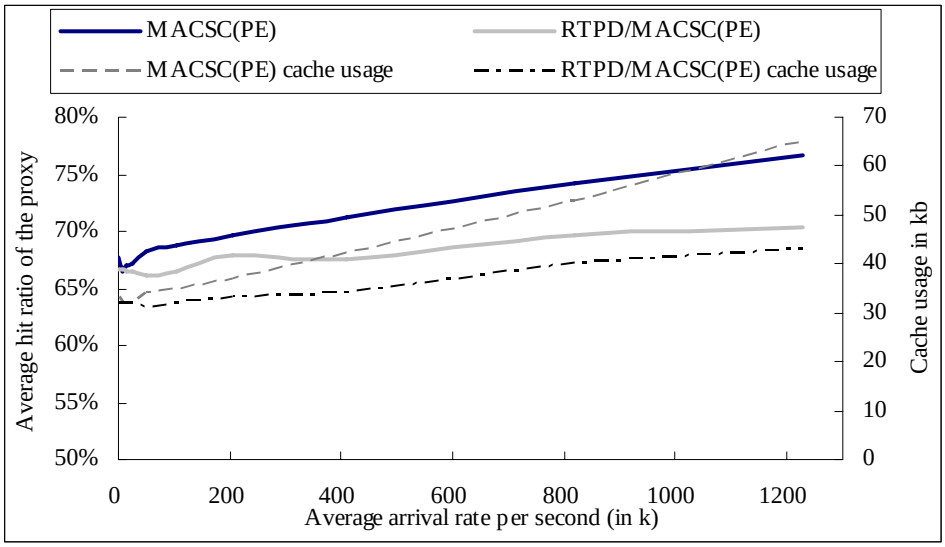


Figure 11.11 Hit ratios and cache usages by MACSC(PE) (hit ratio: 66.7 – 79.7%) and RTPD/MACSC(PE) (hit ratio: 66.7 – 69.7%) with different arrival rates

Environment	Pentium III 930MHz CPU ($933*10^6$ clock cycles per second), 256MB Ram		Celeron® 2.6GHz CPU ($2.6*10^9$ clock cycles per second), 512MB Ram		Pentium 4 2GHz CPU ($2*10^9$ clock cycles per second), 1 GB Ram	
	Total Physical time	Total Physical time	Total Physical time	Physical time for the calculation of 1 mean & S.D.	Total Physical time	Physical time for the calculation of 1 mean & S.D.
MACSC (Total 49 calculations)	1260 ms	1260 ms	1218 ms	≈24.9 ms	815 ms	≈16.6 ms
RTPD/MACSC (Total 42 calculations)	1273 ms	1273 ms	1229 ms	≈29.3 ms	873 ms	≈20.8 ms

Table 11.2 Execution time comparison for MACSC and RTPD/MACSC over three different platforms

Different experiments confirm that the execution times for MACSC and RTPD/MACSC are comparable. Table 11.2 is a comparison for three different computer platforms. For example, with the Intel® VTune™ Performance Analyzer the MACSC was found to take an average of 25 ms to execute on the PIII 930MHz CPU platform. For the three different platforms (as for others) the RTPD/MACSC consistently uses around 20% more execution time on average. Indeed, both the MACSC and RTPD/MACSC prototypes programmed in Java need very little computation power to calculate the cache size adjustment on the fly.

11.4.2 PRELIMINARY CONCLUSION

The novel RTPD/MACSC approach, RTPD/MACSC(PE), for dynamic cache size tuning indeed can help mobile/pervasive diagnostic information retrieval in telemedicine. It combines the MACSC (*Model for Adaptive Cache Size Control*) framework and the real-time traffic pattern detection (RTPD) capability. The experimental results with wireless traces as well as known traffic patterns (LRD and SRD types) confirm that it has no problem in achieving its goal of maintaining a given hit ratio under all traffic conditions. The RTPD/MACSC(PE) has less cache memory usage than the MACSC and thus less potential for deleterious effects. Despite the presence of the complex RTPD mechanism, the RTPD/MACSC(PE) needs only 20% more execution time than the MACSC(PE) tuner working alone. The RTPD/MACSC(PE) solution adapts the initial sample size n for the *point-estimate* computation in every cycle on the fly. The adaptation is done with respect to the traffic pattern detected and identified by the RTPD capability. The verification experiments confirm: a) there is a correlation between the IAT interval

and the request loss rate, which affects the cache hit ratio, and b) tuning the parameter *n* for *point estimate* neutralizes the ill effects on the cache hit ratio by traffic pattern changes that cause higher cache memory usage. The next step of the research should aim at validating the RTPD/MACSC(PE) solution by testing it vigorously in different real wireless environments.

11.5 CONNECTIVE SUMMARY

In this chapter, the contribution of the MACSC framework to the mobile and telemedicine applications is discussed. The experiments with the wireless data show that the MACSC framework can support e-diagnosis or other mobile medical applications. In the next chapter, the conclusion, achievements and future work will be discussed.

CHAPTER 12

CONCLUSION, FUTURE WORK AND ACHIEVEMENTS

In this chapter, the following will be discussed: the overall conclusion, future work and achievements. The comparison of the four novel solutions, namely, MACSC(PE), MACSC(M^3RT), MACSC(F-E) and RTPD/MACSC(PE) is also provided as a recap.

12.1 OVERALL CONCLUSION

Without caching support the Internet can easily become terribly congested and lose its appeal. The danger of congestion is aggravated by the fact that the WWW page size has a monthly growth rate of around 15% but the Internet backbone capacity only increases by 60% yearly. The massive quantity of information requiring transfer across the network in browsing and information retrieval can quickly deplete the amount of sharable bandwidth. The retransmissions needed to recover lost information due to various network faults are inevitable due to the sheer size and heterogeneous nature of the Internet, and this worsens the situation.

Caching alleviates network congestion and hastens WWW information retrieval by providing two advantages. The explicit advantage is the shortening of the service roundtrip time (*RTT*) for retrieval while the implicit advantage from speedup is less data being transferred across the network, thus providing more backbone bandwidth

for sharing and less chance of network congestion.

The service *RTT* is the interval between the client's request and reception of the server's correct corresponding result and in this client/server relationship it conceptually consists of two legs. The first is the roundtrip between the client and the proxy server, and the second, between the proxy server and the remote data source or web server. If the proxy server finds the data object in its cache, then the second leg is automatically obviated. The hit ratio is the chance of finding the required data locally in the proxy's cache. It fluctuates with the clients' shift of preference for certain data items. For a set of data objects this shift quickly changes the relative popularity profile. If σ is the cache hit ratio, and RTT_1 and RTT_2 are the average roundtrip times respectively for the first and second legs, then $S=(RTT_1+RTT_2)/(RTT_1+[1-\sigma]*RTT_2)$ is the information retrieval speedup, where $[1-\sigma]$ is the miss ratio of the proxy cache. With σ = 0.5, RTT_1=10 and RTT_2=40 the speedup is S=(10+40)/(10+0.5*40)=50/30 or 1.67. Having a high proxy cache hit ratio is advantageous because RTT_2, which involves the Domain Name Server (DNS), is usually much longer than RTT_1. The DNS helps the proxy locate the required data objects in the correct remote data source.

The explicit and implicit advantages from caching have motivated different areas of relevant research. The most popular topic is the design of replacement strategies that effectively keep as many hot data objects as possible in the cache. Continual replacement operations, which update the contents of the cache at the same time, are necessary to prevent the cache data from becoming stale (i.e. data incoherence). From this point of view it seems practical to have a very large cache but, this also increases the chance of data coherence. Therefore it is important to

produce a high cache hit ratio and keep data incoherence abated at the same time. Almost all the known replacement strategies work with a static cache size, and they aim to yield a high cache hit ratio but do not necessarily maintain it. For this reason the cache hit ratio fluctuates with respect to the system dynamics and the current relative data object popularity profile. Maintaining a given cache hit ratio needs dynamic cache size tuning which, in contrast, works with a variable cache size.

The original MACSC (*Model for Adaptive Cache Size Control*) conceptual framework for dynamic cache size tuning over the Internet is proposed in this thesis. It strives to maintain the given hit ratio under all conditions by leveraging the relative data object popularity profile as the sole parameter. This leveraging strategy shortens the MACSC execution time so that it computes and administers the tuning solution quickly to avoid possible deleterious effects. The MACSC conceptual framework is based on the Zipf-like behavior, which is the log-log plot of the access frequencies versus the corresponding ranked positions (r) of the data objects in the trace; that is, $y(r) \propto (1/r)^{-\beta}$. The β parameter, which is within the $0 < \beta \leq 1$ range, denotes this relativity, and for β =1 it is called the Zipf Law. The MACSC framework does not work directly with *y(r)* but with the popularity distribution (PD). The PD is a bell curve produced by the mapping *y(r)* into a bell-shaped curve. The mapping mechanism is *bell(r)=map(y(r))+e*, where *e* denotes the possible mapping error. The argument is that if the bell curve indeed produces the expected dynamic cache size tuning result, then *e* can be ignored. The dynamic cache size tuning objective is achieved by measuring the PD standard deviation (SD) quickly and accurately on the fly in a statistical manner. From two successive SD values the popularity ratio (PR), namely, (SD_i/SD_{i-1}) (i.e. standard deviation ratio (SR)) or

$(SD_i/SD_{i-1})^2$ (i.e. variance ratio (VR)) can be calculated, where *i* denotes the computation/tuning cycle. The amount of dynamic cache size tuning to be done in each cycle on the fly is determined by the current PR ratio. The MACSC emphasizes supporting small, inexpensive caching systems, which usually cost less than USD1000 in the field. Therefore, it is important that it works correctly with the SR ratio because the VR involves a large amount of memory in the tuning process and thus may easily deplete the memory resources of small systems and make them perform sluggishly.

In a live caching system the PD shape changes continuously because it reflects the user preference for particular data objects. The MACSC mechanism needs to compute the SD value for the PD at anytime in order to achieve the dynamic cache size tuning and maintain the given hit ratio successfully. Since the Internet dynamics is the result of a discrete stochastic process, the SD value should be computed statistically by direct data measurement. Yet, the Internet follows the power law and its traffic pattern in light IAT (inter-arrival time) pattern changes over time. Therefore, for the MACSC accuracy to be IAT independent, the methods for computing the SD should be based on the *Central Limit Theorem* (CLT). For this reason two basic methods, namely, point estimate (PE) and the M^3RT, which is a micro IEPM (Internet End-to-End Performance Measurement) technique, are adopted. The M^3RT is micro because it exists as a logical object to be invoked for service anytime and anywhere. These two basic methods led to the following solutions: MACSC(PE) (i.e. PE based) and MACSC(M^3RT) (i.e. M^3RT based). The MACSC(PE) solution is oscillatory as observed from different verification experiments because of high PE sensitivity to the profile changes in the relative data

object popularity. In contrast, the MACSC(M^3RT) solution lacks the PE's sensitivity because the feedback loop in the M^3RT mechanism dampens it despite giving the system stability. To combine the merits of the PE and M^3RT mechanisms the third solution, namely, MACSC(F-PE) is proposed. In this solution, which outperforms its predecessors in all the verification experiments in a consistent manner, the PE mechanism is moderated by a feedback loop. Strikingly this feedback loop makes the MACSC(F-PE) dynamic cache size tuning accuracy insensitive to the IAT ill effects. The preliminary experimental results in the MACSC(PE) verification exercise showed that its accuracy depended on the IAT traffic pattern. This inspired the investigation of real-time traffic pattern detection (RTPD) and the possibility of allowing the MACSC(PE) framework to have this capability so that it can use the detected result to reconfigure itself on the fly to ward off traffic ill effects. This led to the proposal of the novel RTPD/MACSC(PE) solutions. This solution is an example of how real-time traffic pattern detection can be incorporated to enhance the efficacy of a time-critical application. The preliminary verification results show that both MACSC(F-E) and RTPD/MACSC(PE) are independent of the IAT traffic. The RTPD/MACSC(PE), however, has the advantage of knowing exactly what IAT traffic condition that it is working with. It may provide a greater degree of freedom for the MACSC framework to immediately deal with unusual situations, which has yet to be explored.

Altogether four solutions are proposed and verified for the MACSC conceptual framework, which is formulated based on the Zipf-like behavior. This behavior is distinctive for caching systems that deal with a large number of data objects, and the four solutions are, namely, MACSC(PE), MACSC((M^3RT), MACSC(F-PE), and

RTPD/MACSC(PE). Since the MACSC(F-PE) verification shows that this solution is independent of IAT patterns, the investigation of having a RTPD/MACSC(F-PE) solution becomes redundant and therefore no further effort at this stage is warranted in this direction. In the verification exercise the experiments made use of the relative popularity profile of all the data objects embedded in reasonably large traces (simulated or real). In this way every data object can be uniquely identified to facilitate the calculation of the popularity ratio. In real-life applications, however, a caching system not only has to deal with a huge amount of different data objects but also unpredictable access preferences. This makes the following impractical for straight MACSC implementation in real-life deployment: a) to identify every data object in the large population, and b) to compute Zipf-like and popularity distributions for the whole population in every dynamic cache size tuning cycle since the long computation delay would make the MACSC solutions unworkable. Therefore, the approach proposed in the thesis for realizing the MACSC framework for real-life deployment is to create the Zipf-like behavior and the PD from the data items sampled for the current dynamic cache size tuning cycle. If there is deviation from the Zipf-like behavior then the "kurtosis and skewness (KS)" test is used to confirm the existence of a valid popularity distribution from which the popularity ratio would be derived. Since the validity confirmation requires a reasonable number of sampled items, the proposed approach for MACSC realization and validation for real-life applications applies to only those PE based solutions (i.e. MACSC(PE), MACSC(F-PE) and RTPD/MACSC(PE) and not the MACSC(M^3RT). The MACSC(M^3RT) is excluded because it uses a small, fixed *flush limit* number of data items per sample. This small sample size does not give the MACSC(M^3RT)

approach the necessary sensitivity.

The thesis has achieved all the stated objectives and demonstrated how the MACSC conceptual framework can support mobile and time-critical applications such as TCM (Traditional Chinese Medicine) based telemedicine. The novel MACSC framework that maintains a given hit ratio by dynamic cache size tuning has been successfully verified with respect to the four solutions: MACSC(PE), MACSC(M^3RT), MACSC(F-PE), and RTPD/MACSC(PE). The proposed approach for implementing the MACSC framework for validation and practical deployment excludes the MACSC(M^3RT) solution.

Table 12.1 summarizes and compares the four novel solutions proposed in this thesis as follows:

MACSC(PE) is the first implemented solution of the MACSC framework. It maintains the hit ratio successfully in spite of its three problems: oscillation in the convergence process, unpredictable sampling time, and being sensitive to Internet traffic patterns.

1. MACSC(M^3RT) eliminates the MACSC(PE) oscillations by incorporating the M^3RT IEPM (Internet End-to-End Performance Measurement) technique. The sampling time of this technique is predictable because it uses the flush limit *f* number of samples in every computation cycle. Yet, MACSC(M^3RT) has two problems: a) it uses more memory than the MACSC(PE), and b) it is sometimes not responsive enough and this produces a low hit ratio.
2. MACSC(F-PE) is proposed to reduce the MASCSC(PE) oscillation by the using of the feedback concept of MACSC(M^3RT). The MACSC(F-PE) performance is better than MACSC(PE) and MACSC(M^3RT) because it yields

the highest hit ratio consistently with less memory than the other two solutions.

3. RTPD/MACSC(PE) is proposed to eliminate the ill effects by Internet traffic patterns. It maintains a better hit ratio than MACSC(PE) working alone with the presence of real-time traffic pattern detection (RTPD) mechanism. This solution works with a reconfiguration scheme, which uses of the results detected by the RTPD mechanism to self-tune so that traffic ill effects are nullified.

These four solutions represent an evolutionary development process in the course of my PhD research.

a. MACSC(PE)	b. MACSC(M^3RT)	c. MACSC(F-PE)	d. RTPD/MACSC(PE)
* Oscillation problem * Unpredictable on-line sampling time * Control accuracy is affected by different Internet traffic patterns	* Eliminates MACSC(PE) oscillation * Predictable sampling time by using the flush limit *f* number of samples * Uses more memory than MACSC(PE) * Fail to respond quickly to traffic pattern changes and therefore fail to yield a high hit ratio	* Reduces MACSC(PE) oscillation by using the feedback concept of MACSC(M^3RT) * Minimizes memory usage * Needs no RTPD because its accuracy is insensitive to Internet traffic patterns	* Includes RTPD to eliminate ill effects by Internet traffic patterns * Maintains the hit ratio better than the MACSC(PE) * Carries out real-time reconfiguration that works with the traffic pattern identified by RTPD currently
From a. to d. above is an evolutionary development process			

Table 12.1 Summary and comparison of four solutions (an evolutionary process)

12.2 RESEARCH METHODOLOGY ADAPTATION

By nature this PhD research is exploratory because the topic of dynamic cache size tuning has little published experience. Despite this, the research produces a prototype for testing and supporting further research as one of its output, the process is naturally top-down because the course of research includes literature search, problem statement, proposed solutions, and data collection. It is, however, difficult to apply the Top Down approach in a strict sense because early exploratory investigations that produce unpredictable results are necessary. That is, the whole investigation would involve repetitive backtracking and cross-referencing to gain the necessary insight for the next step. This means a need to find a research methodology that can cater to the repetitive and exploratory nature of the research. After a careful consideration, I decided to customize the original methodology proposed for my previous exploratory MPhil research [Wu02], namely, "*investigate & experiment & proceed with possible backtracking, cross referencing and looping* (IEP)" approach for the PhD investigation. The research process involves different traversals in the IEP methodology and backtracking. It is summarized as follows:

a. Understanding the rationale of caching in general,
b. Studying some general caching approaches and statistical approaches,
c. To propose a dynamic caching framework, MACSC (*Model for Adaptive Cache Size Control*),
d. Implement the MACSC with different statistical approaches,
e. Looking for a stable mobile-agent platform for testing purposes,
f. Refining the MACSC frameworks for better data collection and analysis, and

g. Demonstrating how the MACSC frameworks can be implemented in the real environment.

12.3 AREAS OF FUTURE WORK

The implementation of any of these conceptual solutions, however, has to overcome the problem of very large data sets in a caching system. Since the Zipf-like behavior is usually apparent only for very large datasets, compensation methods are needed to deal with situations that deviate from the formal Zipf-like behavior, which is governed by $y \propto (1/r)^{-\beta}$ for $0 < \beta \leq 1$. The deviation is usually indicated by the condition of $\beta > 1$. In such cases the bell nature of the distribution is verified by the "kurtosis and skewness" test. The Zipf-like behavior refers to the relative access frequencies of data objects in a very large dataset. It becomes impractical for the "ranker" to re-establish a new Zipf-like distribution and then compute the standard deviation for the popularity ratio (PR) essential for the dynamic cache size adjustment operation. The proposed alternative for the implementation of the four solutions is to compute PR with respect to the set of data object requests sampled for the current tuning cycle. In this way the PR ratio can be computed quickly from the Zipf-like distribution based on the set. Simulations with different traces show that this is a viable solution for practical MACSC deployments. The only problem is that this implementation approach does not work well for the MACSC(M^3RT) solution because it only needs a small number of samples (i.e. equal to the flush limit) to estimate the mean of any distribution accurately. The small flush-limit number of samples, however, makes it difficult to construct a meaningful Zipf-like distribution with high confidence. How to implement the MACSC(M^3RT) effectively is therefore

an important item for future exploration.

The research has uncovered different problems, which should be addressed in the future to make the MACSC framework more deployable for time-critical applications. The more immediate future work items include:

a. Conduct more M^3RT investigation: The aim is to find out how this technique can be efficaciously utilized for accurate estimation of the standard deviation of the popularity distribution on the fly in MACSC application.
b. Perform more precise calibration: The aim is to calibrate traffic patterns versus the initial *n* values for the PE statistical estimation.
c. Test MACSC with different replacement algorithms: The experiments conducted so far worked with the LRU (least recently used) approach. It is worthwhile to compare the performance of different "MACSC + replacement algorithm" combinations because some useful combination may exist.

12.4 ACHIEVEMENTS

This PhD thesis has contributed a novel conceptual framework, namely, the MACSC, for maintaining a given hit ratio by dynamic cache size tuning over the Internet. Four solutions for realizing the framework were proposed and verified, namely, MACSC(PE), MACSC(M^3RT), MACSC(F-PE), and RTPD/MACSC(PE). A practical implementation method, which is based on the data samples collected for the current tuning cycle, is proposed and tested. This method works well for the MACSC(PE), MACSC(F-PE), and RTPD/MACSC(PE) solutions but not the MACSC(M^3RT). The four conceptual solutions provide a solid basis for future

deeper research in the direction of dynamic cache size tuning over the Internet. The verification results indicate that the MACSC framework definitely contributes to shorten the service roundtrip time in web information retrieval. The findings from the research so far have contributed to 17 refereed publications as follows:

8 Refereed Journal Papers

[1] Richard S.L. Wu, Wilfred W.K. Lin and Allan K.Y. Wong, Harnessing Wireless Traffic is an Effective Way to Improve Mobile Internet Performance, *Proceedings of 1st IEEE International Conference on Wireless Broadband and Ultra Wideband Communications AuS Wireless 2006*, March, 2006

[2] Richard S. L. Wu, Allan K. Y. Wong and Tharam S. Dillon, $CACHE_{RP}$: A Novel Dynamic Cache Size Tuning Model Working with Relative Object Popularity for Fast Web Information Retrieval, *Journal of Supercomputing*, 2006 (Accepted and will appear in the journal)

[3] Richard S. L. Wu, Allan K. Y. Wong and Tharam S. Dillon, E-MACSC: A Novel Dynamic Cache Tuning Technique to Reduce Information Retrieval Roundtrip Time over the Internet, *Journal of Computer Communications* (Accepted and will appear in the journal)

[4] Wilfred W. K. Lin, Allan K. Y. Wong, Richard S. L. Wu, Applying Fuzzy Logic and Genetic Algorithms to Enhance the Efficacy of the PID Controller in Buffer Overflow Elimination for Better Channel Response Timeliness over the Internet, *Journal of Concurrency: Practice & Experience* (Accepted and will appear in the journal)

[5] Richard S. L. Wu, Allan K. Y. Wong and Tharam S. Dillon, RDCT: A Novel Reconfigurable Dynamic Cache Tuner to Shorten Information Retrieval Time over the Internet, *International Journal of Computer Systems Science & Engineering*, 19(6), 2004, 363 – 371.

[6] Richard S. L. Wu, Allan K. Y. Wong and Tharam S. Dillon, $CACHE_{RP}$: A Novel Dynamic Cache Size Tuning Model working with Relative Object Popularity for Fast Web Information Retrieval, *Electronic Journal of Lecture Notes in Computer Science*, Springer-Verlag GmbH, 3358, 2004.

[7] Richard S. L. Wu, Allan Kang Ying Wong and Tharam S. Dillon, E-MACSC: A Novel Dynamic Cache Tuning Technique to Maintain the Prescribed Minimum Hit Ratio Consistently for Internet/WWW Applications, *World Scientific and Engineering Academy and Society (WSEAS) Transactions on Computers*, April 2004, 3(2), 424 – 429.

[8] Allan K. Y. Wong, May T. W. Ip, Richard S. L. Wu, A Novel Dynamic Cache Size Adjustment Approach for Better Data Retrieval Performance over the Internet, *Journal of Computer Communications*, September 2003, 26(14), 1709-1720.

2 Book Chapters (Invited)

[9] Richard S. L. Wu, Allan K. Y. Wong and Tharam S. Dillon, E-MACSC: A Novel Dynamic Cache Tuning Technique to Maintain the Hit Ratio Prescribed by the User in Internet Applications, International Conference on E-Business and Telecommunication Networks (ICETE 2004) Best Paper Book, Kluwer Academic Publishers, 2004 (***best papers series by ICETE'04)***).

[10] Allan K. Y. Wong, Richard S. L. Wu and Tharam S. Dillon, Dynamic

Maintenance of a Given Proxy Cache Hit Ratio by Leveraging the Relative Data Object Popularity Profile to Yield Shorter Service Roundtrip Time, to appear in an edited collection by Nova Science Publishers, Inc., New York.

7 Refereed Conference Papers

[11] Wilfred W. K. Lin, Allan K. Y. Wong, Richard S. L. Wu, A Novel Real Time Self Similar Traffic Detector/Filter to Improve the Reliability of a TCP Based End to End Client/Server Interaction Path for Shorter Roundtrip Time, *Proceedings of 2nd International Conference on E-Business and Telecommunication Networks (ICETE 2005)*, Reading, U.K., October 2005

[12] Richard S. L. Wu, Tharam S. Dillon and Allan K. Y. Wong, RTPD/MACSC: A Novel Approach for Effective Pervasive Information Retrieval, *Proceedings of the Fourth International Conference on Mobile Business (ICMB 2005)*, Sydney, Australia, July 2005, 514 – 520.

[13] Richard S. L. Wu, Allan K. Y. Wong, Tharam S. Dillon, Using Real-Time Traffic Pattern Detection for Dynamic Cache Size Tuning in Information Retrieval, *Proceedings of the Third Internal Conference on Information Technology and Applications (iCITA' 2005)*, Sydney, Australia, July 2005, Volume 2, 35 – 40, Volume 2, 35 – 40.

[14] Richard S. L. Wu, Allan K. Y. Wong, Tharam S. Dillon, $CACHE_{RP}$: A Novel Dynamic Cache Size Tuning Model working with Relative Object Popularity for Fast Web Information Retrieval, *Proceedings of Second International Symposium on Parallel and Distributed Processing and Applications (ISPA 2004)*, Hong Kong, China, December 2004, 410 – 420.

[15] Wilfred W. K. Lin, Richard S. L. Wu, Tharam S. Dillon and Allan K. Y. Wong, A Novel Real-Time Traffic Pattern Detector for Internet Applications, *Proceedings of the 2004 Australian Telecommunication Networks and Applications Conference (ATNAC 2004)*, December 2004, 224 – 227.

[16] Richard S. L. Wu, Allan K. Y. Wong and Tharam S. Dillon, E-MACSC: A Novel Dynamic Cache Tuning Technique to Maintain the Hit Ratio Prescribed by the User in Internet Applications, *Proceedings of 1st International Conference on E-Business and Telecommunication Networks (ICETE 2004)*, Set al/Portugal, August 2004, Volume 1, 152 – 159.

[17] Richard S. L. Wu, May T. W. Ip and Allan K. Y. Wong, LDC-CM: A Novel Model for Dynamic Cache Size Adjustment, *Proceedings of 2003 International Conference on Internet Computing*, Las Vegas, Nevada, USA, June 2003, Volume 2, 753-758.

REFERENCES

[Abrams95] M. Abrams, C. Standridge, G. Abdulla, S. Williams and E.A. Fox, Caching proxies: limitations and potentials, *Proceedings of the 4th International World Wide Web Conference*, MA, December 1995.

[Abrams96] M. Abrams, C. R. Standridge, G. Abdulla, E. A. Fox and S. Williams, Removal Policies in Network Caches for WWW Documents, *ACM SIGCOMM Computer Communication Review*, 26(4), 1996.

[Aggarwal99] C. Aggarwal, J. L. Wolf and P. S. Yu, Caching on the Word Wide Web, *IEEE Transactions on Knowledge and Data Engineering*, 11(1), 1999.

[Androutsellis-Theotokis04] Androutsellis-Theotokis and D. Spinnellis, A Survey of Peer-to-Peer Content Distribution Technologies, *ACM Computing Surveys*, 36(4), 2004, 335 – 371.

[Aridor98] Y. Aridor and M. Oshima, Infrastructure for Mobile Agents: Requirements and Design, *Proceedings of 2nd International Workshop on Mobile Agents (MA '98)*, Springer Verlag, September 1998.

[Arlitt99] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich and T. Jin, Evaluating Content Management Techniques for Web Proxy Caches, *Proceedings of the 2nd Workshop on Internet Service Performance*, 1999.

[Avizienis04] A. Avizienis, J.-C. Laprie, B. Randell and C. Landwehr, Basic Concepts and Taxonomy of Dependable and Secure Computing, *IEEE Transactions on Dependable and Secure Computing*, 1(1), 2004, 11-33.

[Balachandran02] A. Balachandran, G.. M. Voelker, P. Bahl and P. V. Rangan,

Characterizing User Behavior and Network Performance in a Public Wireless LAN, *Proceedings ACM SIGMETRICS'02*, Marina Del Rey, June 2002, 30(1), 195 – 205.

[Belloum98] A. Belloum and L. O. Hertzberger, Dealing with One-Timer-Documents in Web caching, *Proceedings of the 24th Conference on EUROMICRO*, 2, 1998.

[Bestavros96] A. Bestavros and C. Cunha, Server-initiated document dissemination for the WWW, *IEEE Data Engineering Bulletin*, September 1996.

[Bharat98] K. Bharat and A. Broder, Estimating the Relative Size and Overlap of Public Web Search Engines, *Proceedings of the 7th International World Wide Web Conference (WWW7)*, April 1998.

[Bjarat98] K. Bjarat and A. Broder, Estimating the Relative Size and Overlap of Public Web Search Engines, *Proceedings of the 7th International World Wide Web Conference (WWW7)*, 1998.

[Bolot96] J-C Bolot and P. Hoschka, Performance Engineering of the World Wide Web: Application to Dimensioning and Cache Design, *Computer Networks*, 28(7), 1996.

[Braden98] B. Braden et. al., Recommendations on Queue Management and Congestion Avoidance in the Internet, *RFC2309*, April 1998

[Breslau99] L. Breslau, P. Cao, F. Li, G. Phillips and S. Shenker, Web Caching and Zipf-like Distributions: Evidence and Implications, *Proceedings of the INFOCOM'99*, Vol.1, 1999.

[Burkhardt02] J. Burkhardt, H. Henn, S. Hepper, K. Rintdorff, T. Schack, Pervasive Computing, Addison-Wesley, 2002

[Cáceres98] R. Cáceres, F. Douglis, A. Feldmann, G.. Glass and M. Rabinovich, Web proxy caching: the devil is in the details, *ACM Performance Evaluation Review*, 26(3): pp. 11-15, December 1998.

[Cao97] P. Cao and S. Irani, Cost-Aware WWW Proxy Caching Algorithms, *Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems*, 1997.

[Cao98] P. Cao, J. Zhang and K. Beach, Active cache: caching dynamic contents on the Web, *Proceedings of IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'98)*, pp. 373 – 388, 1998.

[Challenger99] J. Challenger, A. Iyengar and P. Dantzig, A scalable system for consistently caching dynamic Web data, *Proceedings of Infocom'99*.

[Chankhunthod96] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz and K. J. Worrel, A hierarchical Internet object cache, *Proceedings of Usenix'96*, January 96.

[Chinen97] K. Chinen and S. Yamaguchi, An interactive prefetching proxy server for improvement of WWW latency, *Proceedings of INET'97*, June 1997.

[Chis92] J. A. Chis, Introduction to Simulation and Modeling - GPSS/PC, Prentice Hall, 1992.

[Chris00] M. Christiansen, K. Jeffay, D. Ott and F. D. Smith, Tuning RED for Web Traffic, *ACM SIGCOMM*, August 2000.

[Cohen98] E. Cohen, B. Krishnamurthy and J. Rexford, Improving end-to-end performance of the Web using server volumes and proxy filters, *Proceedings of Sigcomm'98*.

[Cohen99] E. Cohen, B. Krishnamurthy and J. Rexford, Efficient algorithms for

predicting requests to Web servers, *Proceedings of Infocom'99*.

[Cottrel99] L. Cottrel, M. Zekauskas, H. Uijterwaal and T. McGregor, Comparison of Some Internet Active End-to-End Performance Measurement Projects, http://www.slac.stanford.edu/comp/net/wan-mon/iepm-cf.html, July 1999

[Cottrel01] L. Cottrel, Passive vs. Active Monitoring, http://www.slac.stanford.edu/comp/net/wan-mon/passive-vs-active.html, March 2001.

[Crovella98] M. Crovella and P. Batford, The network effects of prefetching, *Proceedings of Infocom'98*.

[Dilley99] J. Dilley and M. Arlitt, Improving Proxy Cache Performance: Analysis of Three Replacement Policies, IEEE Internet Computing, 1999, 44-50.

[Douglis97] F. Douglis, A. Feldmann, B. Krishnamurthy and J. Mogul, Rate of change and other metrics: a live study of the World Wide Web, *Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems (USITS-97)*, December 1997.

[Downey99] A. B. Downey, Using pathchar to Estimate Internet Link Characteristics, *Proceedings of the ACM SIGCOMM'99*, October 1999, 241-250.

[Duska97] B. M. Duska, D. Marwood and M. J. Feeley, The Measured Access Characteristics of World-Wide-Web Client Proxy Caches, *Proceedings of the USENIX Symposium on Internet Technology and Systems*, 1997.

[Embrechts97] P. Embrechts, C. Klüppelberg and T. Mikosh. Modeling Extremal Events for Insurance and Finance, Springer-Verlag, Berlin Heidelberg, 1997.

[Fan98] L. Fan, P. Cao, J. Almeida and A. Z. Broder, Summary cache: a scalable wide-area Web cache sharing protocol, *Proceedings of Sigcomm'98*.

[Fan99] L. Fan, P. Cao, W. Lin and Q. Jacobson, Web prefetching between low-bandwidth clients and proxies: potential and performance, *Proceedings of the Sigmetrics'99*.

[Feldmann99] A. Feldmann, R. Caceres, F. Douglis, G. Glass and M. Rabinovich, Performance of Web proxy caching in heterogeneous bandwidth environments, *Proceedings of Infocom'99*.

[Firoiu00] V. Firoiu and M. Borden, A study of Active Queue Management for Congestion Control, *Proceedings of the INFOCOM 2000*, March 2000.

[Gadde97] S. Gadde, M. Rabinovich and J. Chase, Reduce, reuse, recycle: an approach to building large Internet caches, *Proceedings of the HotOS'97 Workshop*, May 1997.

[Garlan02] D. Garlan, D. P. Siewiorek, A. Smailagic and P. Steenkiste, Project Aura: Toward Distraction-free Pervasive Computing, *IEEE Pervasive Computing*, 1(2), April 2002, 22 – 31.

[Glassman94] S. Glassman, A caching relay for the World Wide Web, *Proceedings of First International Conference on the World Wide Web*, CERN, Geneva, Switzerland, May 1994.

[Gnutella] The Gnutella web site: http://gnutella.wego.com

[Gwetzman94] J. Gwetzman and M. Seltzer, The case for geographical pushing-caching, *HotOS Conference*, 1994.

[Heddaya97] A. Heddaya, S. Mirdrad and D. Yates, Diffusion-based Caching Along Routing Paths, *2nd Intl. Web Caching Workshop*, Baltimore, MD, June 1997.

[Hightower01] J. Hightower and G. Borriello, Location Systems for Ubiquitous Computing, *IEEE Computer*, 34(8), August 2001, 57-66.

[IEEE99] IEEE. 802.11b/d3.0 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, August 1999.

[Ip01] M. T. W. Ip, W. W. K. Lin, A. K. Y. Wong, T. S. Dillon and D. H. Wang, An Adaptive Buffer Management Algorithm for Enhancing Dependability and Performance in Mobile-Object-Based Real-time Computing, *Proceedings of the IEEE ISORC'2001*, Magdenburg, Germany, May 2001, 138-144.

[Ip03] M. T. W. Ip, Development of a Micro IEPM (Internet End-to-End Performance Measurement) Technique for Internet Based Computing, *MPhil Thesis*, Department of Computing, Hong Kong Polytechnic University, Hong Kong SAR, PRC, 2003.

[Jain91] R. Jain, The Art of Computer Systems Performance Analysis, Wiley, 1991.

[Jin00a] S. Jin and A. Bestavros, Popularity-Aware Greedy Dual-Size Web Proxy Caching Algorithms, *Proceedings of the Int'l Conf. on Distributed Computing Systems*, 2000.

[Jin00b] S. Jin and A. Bestavros, Temporal Locality in Web Request Streams: Sources, Characteristics and Caching Implications, *Proceedings of the International Conference on Measurement and Modeling of Computer Systems*, 2000.

[Karagiannis02] T. Karagiannis, M. Faloutsos. SELFIS: A Tool For Self-Similarity and Long-Range Dependence Analysis, *1st Workshop on Fractals and Self-Similarity in Data Mining: Issues and Approaches (in KDD)*, Edmonton, Canada, July 23, 2002

[Karagiannis03] T. Karagiannis, M. Faloutsos, M. Molle, A User-Friendly Self-Similarity Analysis Tool, Special Section on Tools and Technologies for Networking Research and Education, *ACM SIGCOMM Computer Communication Review*, 33(3), 2003.

[Karger97] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin and R. Panigrahy, Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web, *STOC 1997*.

[Ketchen04] D. J. Ketchen Jr., D. D. Bergh Eds., Research Methodology in Strategy and Management, Elsevier 2004

[Kroeger97] T. M. Kroeger, D. D. E. Long and J. C. Mogul, Exploring the bounds of Web latency reduction from caching and prefetching, *Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems*, Monterey, CA, December 1997.

[Kumar96] R. Kumar, Research Methodology, A Step-by-Step Guide for Beginners, SAGE Publications 1999.

[Lacroix99] A. Lacroix, L. Lareng, G. Rossignol, D. Padeken, M. Bracale, Y. Ogushi, R. Wootton, J. Sanders, S. Preost and I. McDonald, G-7 Global Healthcare Applications Sub-project 4, *Telemedicine Journal*, March 1999.

[Lakeshman96] T. V. Lakeshman, A. Neidhardt and T. Ott, The Drop from Front Strategy in TCP over ATM and Its Internetworking with other Control Features, *INFOCOMM 1996*.

[Lakeshman97] T. Lakeshman and U. Madlow, The Performance of TCP/IP for Networks with High Bandwidth –Delay Products and Random Loss, *IEEE/ACM Transactions on Networking*, 5(3), June 1997, 336-350

[Laamanen99] H. Laamanen, T. Alanko and K. Raatikainen, Dependability issues in mobile distributed system, *Proceedings of the Pacific Rim International Symposium on Dependable Computing*, 1999, 7 – 14.

[Laprie95] J.-C. Laprie, Dependable Computing: Concepts, Limits, Challenges,

Proceedings of the IEEE 25th International Symposium on Fault-Tolerant Computing, 1995.

[Leland94] W. Leland, M. Taqqu, W. Willinger and D. Wilson, On the Self-Similar Nature of Ethernet Traffic (Extended Version), *IEEE/ACM Transactions on Networking*, 2(1), February 1994, 1-15

[Lewis96] T. Lewis, The Next 10000_2 Years: Part 1, *IEEE Computer*, April 1996

[Levy-Abegnoli99] E. Levy-Abegnoli, A. Iyengar, J. Song and D. Dias, Design and performance of Web server accelerator, *Proceedings of Infocom'99*.

[Lewandowski98] S. M. Lewandowski, Frameworks for Component-Based Client/Server Computing, *ACM Computer Survey*, March 1998.

[Li99] B. Li, M. J. Golin, G.. F. Italiano, X. Deng and K. Sohraby, On the optimal placement of Web proxies in the Internet, *Proceedings of Infocom'99*.

[Lin04] W. W. K. Lin, R. S. L. Wu, T. S. Dillon and A. K. Y. Wong, A Novel Real-Time Traffic Pattern Detector for Internet Applications, *Proceedings of the 2004 Australian Telecommunication Networks and Applications Conference (ATNAC 2004)*, December 2004, 224 – 227.

[Loon97] T. S. Loon and V. Bharghavan, Alleviating the latency and bandwidth problems in WWW browsing, *Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems (USITS-97)*, December 1997.

[Luotonen94] A. Luotonen and K. Altis, World Wide Web proxies, Computer Networks and ISDN Systems, *First International Conference on WWW*, April 1994.

[M94] 宋天彬，“實用中醫舌診彩色圖譜”，合肥市: 安徽科學技術出版社, 1994.

[Maciocia87] G. Maciocia, Tongue Diagnosis in Chinese Medicine, Eastland Press,

Seattle USA, 1987.

[Mahanti00] A. Mahanti, C. Williamson and D. Eager, Traffic Analysis of Web Proxy Caching Hierarchy, *IEEE Network*, 14(3) 2000.

[Malla03] A. Malla, M. El-Kadi, S. Olariu and P. Todorova, A Fair Resource Allocation Protocol for Multimedia Wireless Networks, *IEEE Transactions on Parallel and Distributed Systems*, 14(1), January 2003, 63 – 71.

[Malpani95] R. Malpani, J. Lorch and D. Berger, Making World Wide Web caching servers cooperate, *Proceedings of the 4th International WWW Conference*, Boston, MA, December 1995.

[Markatos98] E. P. Markatos and C. E. Chronaki, A TOP-10 approach to prefetching on Web, *Proceedings of INET'98*.

[Matthews00] W. Matthews and L. Cottrel, The PingER Project: Active Internet Performance Monitoring for the HENP Community, *IEEE Communications Magazine*, May 2000.

[Medina00] A. Medina, I. Matta and J. Byers, On the Origin of Power Laws in Internet Topologies, *ACM SIGCOMM*, 30(2), 2000.

[Michel98] S. Michel, K. Nguyen, A. Rosentein, L. Zhang, S. Floyd and V. Jacobson, Adaptive Web Caching: Towards a New Global Caching Architecture, *Journal of Computer Networks and ISDN Systems*, 1998.

[Molnár99] S. Molnár, T. D. Dang and A. Vidacs, Heavy-Tailedness, Long-Range Dependence and Self-Similarity in Data Traffic, *Proceedings of 7th International Conference on Telecommunication Systems*, Modelling and Analysis, March 1999, Nashville, USA,18-21.

[Nielsen97] J. Nielsen, Zipf Curves and Website Popularity,

http://www.useit.com/alertbox/zipf.html

[Padmanabhan96] V. N. Padmanabhan and J. C. Mogul, Using predictive prefetching to improve World Wide Web latency, *Proceedings of Sigcomm'96*.

[Palpanas99] T. Palpanas and A. Mendelzon, Web prefetching using partial match prediction, *Proceedings of WCW'99*.

[Patterson03] C. A. Patterson, R. R. Muntz and C. M. Pancake, Challenges in Location-Aware Computing, *IEEE Pervasive Computing*, 2(2), 2003, 80 – 89.

[Paxson95] V. Paxson, S. Floyd, Wide area traffic: The Failure of Poisson Modeling, *IEEE/ACM Transactions on Networking*, 3(3), 1995.

[Paxson97] V. Paxson, End-to-End Internet Packet Dynamics, ACM SIGCOMM, *Computer Communication Review*, 27(4), October 1997, 139-154.

[Philips87] E. M. Philips and D. S. Pugh, How to Get a Ph.D., Open University Press, 1987.

[Podlipnig03] S. Podlipnig and L. Böszörményi, A Survey of Web Cache Replacement Strategies, *ACM Computing Surveys*, 35(4), December 2003, pp. 374 – 398.

[Povey97] D. Povey and J. Harrison, A distributed Internet cache, *Proceedings of the 20th Australian Computer Science Conference*, Sydney, Australia, February 1997.

[Rabinovich98] M. Rabinovich, J. Chase and S. Gadde, Not all hits are created equal: cooperative proxy caching over a wide-area network, *Computer Networks and ISDN System*, 30(22 – 23), November 1998, 2253 – 2259.

[Ramakrishnan99] K. K. Ramakrishnan and S. Floyd, A proposal to Add Explicit Congestion Notification (ECN) to IP, *RCF 2481*, January 1999

[Reddy98] M. Reddy and G.. P. Fletcher, An Adaptive Mechanism for Web Browser

Cache Management, *IEEE Internet Computing*, January-February, 1998

[Relais98] Relais: cooperative caches for the World Wide Web, 1998

[Ren02] F. Ren, Y. Ren and X. Shan, Design of a Fuzzy Controller for Active Queue Management, *Computer Communication*, 25, 2002, 874-883

[Resnick97] S. I. Resnick, Heavy Tail Modeling and Teletraffic Data, *The Annals of Statistics*, 25(5), 1997, 1805-1869

[Rodriguez99] P. Rodriguez, C. Spanner and E. W. Biersack, Web caching architecture: hierarchical and distributed caching, *Proceedings of WCW'99*.

[Rousskov98] A. Rousskov and D. Wessels, Cache Digest, *Proceedings of 3rd International WWW Caching Workshop*, June 1998.

[Sanghi93] D. Sanghi, A. Agrawala and B. N. Jain, Experimental Assessment of End-to-End Behavior on the Internet, *Proceedings of the IEEE Infocom'93*, San Francisco, USA, March 1993, 867-874.

[Shim99] J. Shim, P. Scheuermann and R. Vingralek, Proxy Cache Algorithms: Design, Implementation, and Performance, IEEE Transactions on Knowledgement and Data Engineering, 11(4), January/August 1999, 549-562

[SIGCOMM] The Internet Traffic Archive, *ACM SIGCOMM Special Interest Group on Data Communications*, http://ita.ee.lbl.gov/index.html

[Stankovic98] J. A. Stankovic, M. Spuri, K. Ramamritham, G. C. Buttazzo, Deadline Scheduling for Real-Time Systems, EDF and Related Algorithms, Kluwer Academic Publishers, 1998

[Tan94] G. Y. Tan, G. S. Hura, A Petri-net-based Modeling Assisted Software Environment (MASE) tool, *Proceedings of Computer Software and Applications Conference*, 439 – 444, 1994.

[Tanenbaum96] A. S. Tanenbaum, Computer Networks, 3rd Edition, Prentice Hall, 1996

[Taqqu03] M. S. Taqqu, Fractional Brownian Motion and Long-Range Dependence, in Theory and Applications of Long-Range Dependence, P. Doukhan et al., Eds., Birkhuser 2003, 5-38.

[Tewari98] R. Tewari, M. Dahlin, H. Vin and J. Kay, Beyond hierarchies: design considerations for distributed caching on the Internet, *Technical Report TR98-04*, Department of Computer Science, University of Texas at Austin, February 1998.

[Valloppillil98] V. Valloppillil and K. W. Ross, Cache array routing protocol v1.0, *Internet Draft*, 1998, http://icp.ircache.net/carp.txt

[VTune] Intel Vtune, http://developer.intel.com/software/products/vtune/

[Wang99] J. Wang, A Survey of Web Caching Schemes for the Internet, *ACM Computer Communication Review*, 29(5), 1999, 36 – 46.

[Wang97] Z. Wang, Cachemesh: a distributed cache system for World Wide Web, *Web Cache Workshop*, 1997.

[Weiser91] M. Weiser, The Computer for the Twenty-First Century, *Scientific American*, September 1991, 94-104.

[Wessels97] D. Wessels and K. Claffy, Internet cache protocol (ICP), version 2, *RFC 2186*, 1997.

[Wessels01] D. Wessels, Web Caching, O'Reilly & Associates Inc., 2001.

[Willinger98] W. Willinger, V. Paxson and M.S. Taqqu, Self-similarity and heavy-tails: Structuring Modeling of Network Traffic, A Practical Guide to Heavy Tails: Statistical Techniques and Applications, Hirkhauser, 1998.

[Willinger03] W. Willinger, V. Paxson, R. H. Hiedi and M. S. Taqqu, Long-Range

Dependence and Data Network Traffic, Theory and Applications of Long-Range Dependence, P. Doukhan et al., Eds., Birkhuser 2003, 373-408.

[Wong01a] A. K. Y. Wong, T. S. Dillon, W. W. K. Lin and M. T. W. Ip, M^2RT: A Tool Developed for Predicting the Mean Message Response Time for Internet Channels, *Computer Networks*, vol. 36, 2001.

[Wong01b] A. K. Y. Wong and J. H. C. Wong, A Convergence Algorithm for Enhancing the Performance of Distributed Applications running on Sizeable Networks, *International Journal of Computer Systems Science & Engineering*, 16(4), 2001, 229-236.

[Wong02a] A. K. Y. Wong, W. W. K. Lin, M. T. W. Ip and T. S. Dillon, Genetic Algorithm and PID Control Together for Dynamic Anticipative Marginal Buffer Management: An Effective Approach to Enhance Dependability and Performance for Distributed Mobile Object-Based Real-time Computing over the Internet, *Journal of Parallel and Distributed Computing (JPDC)*, 62, 2002, 1433-1453.

[Wong02b] A. K. Y. Wong, M. T. W. Ip and T. S. Dillon, M^3RT: An Internet End-to-End Performance Measurement Approach for Real-Time Applications with Mobile Agents, *Proceedings of the ISPAN''02*, 2002.

[Wong03] A. K. Y. Wong, M. T. W. Ip and R. S. L. Wu, A Novel Dynamic Cache Size Adjustment Approach for better Retrieval Performance over the Internet, *Journal of Computer Communications*, 26(14), 2003, 1709-1720.

[Wong97] S. T. C. Wong and H. K. Huang, Networked Multimedia for Medical Imaging, *Journal of Multimedia*, April-June 1997.

[Wu00] K. L. Wu and P. S. Yu, Latency-Sensitive Hashing for Collaborative Web Caching, *Computer Networks*, 33(1-6), 633 – 644, 2000.

[Wu02] S. L. Wu, A Framework for Scalable (Mobile Agent Based) Distributed Mining of Association Rules over the Internet, *MPhil Thesis*, Department of Computing, Hong Kong Polytechnic University, Hong Kong SAR, PRC, 2002.

[Wu03] R. S. L. Wu, M. T. W. IP and A. K. Y. Wong, LDC-CM: A Novel Model for Dynamic Cache Size Adjustment, *Proceedings of the PDPTA'03*, Las Vegas, USA, June 2003.

[Wu04] R. S. L. Wu, A. K. Y. Wong and T. S. Dillon, E-MACSC: A Novel Dynamic Cache Tuning Technique to Maintain the Hit Ratio Prescribed by the User in Internet Applications, *Proceedings of the International Conference on E-business and Telecommunication Networks (ICETE2004)*, Portugal, August 2004, 152-159.

[Wu05a] R. S. L. Wu, A. K. Y. Wong and T. S. Dillon, RDCT: A Novel Reconfigurable Dynamic Cache Tuner to Shorten Information Retrieval Time over the Internet, *International Journal of Computer Systems Science & Engineering*, 2005.

[Wu05b] R. S. L. Wu, A. K. Y. Wong and T. S. Dillon, E-MACSC: A Novel Dynamic Cache Tuning Technique to Reduce Information Retrieval Roundtrip Time over the Internet, *Journal of Computer Communications* (Accepted and will appear in the journal).

[Yang] J. Yang, W. Wang, R. Muntz and J. Wang, Access driven Web caching, *UCLA Technical Report #990007*.

[Young91] N. E. Young, On-line Caching as Cache Size Varies, *Proceedings of the Symposium on Discrete Algorithms*, 1991.

[Yu99] Philip S. Yu and Edward A. MacNair, Performance Study of a Collaborative Method for Hierarchical Caching in Proxy Servers, IBM Watson Research Center,

Research Report.

[Zipf] Zipf Curves and Website Popularity, http://www.useit.com/alertbox/zipf.html

APPENDICES

A1 SCHOLARSHIP BY RESEARCH MERITS

THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

THIS IS TO CERTIFY THAT

WU Sui Lun

has been awarded in the academic year 2004/05

Chung Hwa Travel Service Scholarship

茲證明

胡瑞能

於二零零四至二零零五學年榮獲

中華旅行社獎學金

Mrs. Dorinda Fung
Director of Student Affairs
馮陳敏慈
學生事務總監

30 April 2005

A2 REVIEWERS' COMMENTS

The 2nd International Conference on E-business and Telecommunication Networks (ICETE 2004)

This paper describes an interesting piece of work on dynamic cache tuning. Though the work described therein is an enhancement of the authors' previous model MACSC, the suggested enhancement, which leverages on the relative object popularity profile, has led to significant improvement over the original MACSC approach.

The model deserves more detailed studies, particularly how the parameters should be optimized for different types of systems and environments.

The Second International Symposium on Parallel and Distributed Processing and Applications (ISPA'04)

A novel model for dynamic cache size tuning is proposed. The performance evaluation is also provided. It is basically well-written and of interest.

In this article, the authors have given us a thorough discussion about how to keep the hit ratio of web caches at a consistent level. The topic is novel and interesting. Intensive theoretical analysis and extensive experiment verification have been provided to prove the effectives of authors`s cache size tuning model.

The discussion about how to keep a specific hit ratio but not to gain a hit ratio as high as possible is interesting and novel.

The 3rd International Conference on Information Technology and Applications (ICITA'2005)

This paper introduces a model to tune the cache size adaptively. The model proposed is interesting, and show increased performance in information retrieval.

The Fourth International Conference on Mobile Business (mBusiness 2005)

Original research has been reflected and a novel approach is proposed.

This paper proposes a novel approach to shorten the information retrieval time by a small form factor client in a pervasive computing environment. Relative work, its limitations and how this research proposes to address this gap. They are well presented. Future propositions to test this approach are also well taken.

The authors' primary contribution appears to be using their RTPD method in conjunction with MACSC. This could be made clearer in the paper (and even in the paper title), i.e. that the real contribution is adding RTPD.