ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS

İpek ABASIKELEŞ

PERFORMANCE IMPROVEMENT OF THE TWO DIMENSIONAL BROADCAST-BASED DISTRIBUTED SHARED MEMORY MULTIPROCESSOR SYSTEMS BY MESSAGE OPERATIONS

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

ADANA, 2013

ABSTRACT

PhD THESIS

PERFORMANCE IMPROVEMENT OF THE TWO DIMENSIONAL BROADCAST-BASED DISTRIBUTED SHARED MEMORY MULTIPROCESSOR SYSTEMS BY MESSAGE OPERATIONS

İpek ABASIKELEŞ

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Supervisor : Asst. Prof. Dr. M.Fatih AKAY
Year: 2013, Pages: 139
Jury : Asst. Prof. Dr. M.Fatih AKAY
: Prof. Dr. Hüseyin EKİZ
: Assoc. Prof. Dr. Ulus ÇEVİK
: Assoc. Prof. Dr. Mustafa GÖK
: Assoc. Prof. Dr. S. Ayşe ÖZEL

Distributed shared memory (DSM) systems present strong advantages in terms of scalability, programmability and cost effectiveness. However, obtaining high performance out of these systems turns out to be much more cumbersome than expected due to the increasing remote memory access latency as the number of processors increase. The underlying interconnection network has taken on an increasingly important role in overall performance of DSM systems. Two-dimensional (2D) broadcast-based optical architectures are a popular solution. In this thesis, DSM on 2D broadcast-based optical architecture is simulated by using OPNET Modeler and the performance of the system is improved by message priority and combining techniques under different traffic patterns, including uniform, hot region and bit permutations. The simulated system contains 64 nodes. Each node has a server that processes directory, cache and processor messages, two output channels and two input queues, one for horizontal and the other for vertical dimension. Scenarios are run to measure average network response time, average input waiting time and average channel waiting time for varying values of DSM parameters such as the ratio of the mean channel service time to mean thread run time, probability of a cache block being in modified state and the fraction of write misses. The results show that combining reduces average channel waiting time between 13% and 29%, while priority reduces average input waiting time between 7% and 43% on average. With the application of message priority and combining, between 7% and 40% performance increase for average network response time has gained.

Key Words: Multiprocessor architectures, Distributed shared memory, Message combining, Message priority, Performance evaluation

ÖZ

DOKTORA TEZİ

İKİ BOYUTLU YAYIM TABANLI DAĞITIK ORTAK HAFIZALI MİKROİŞLEMCİ SİSTEMLERİNİN PERFORMANSININ MESAJ OPERASYONLARI İLE ARTIRILMASI

İpek ABASIKELEŞ

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

Danışman : Yrd. Doç. Dr. M.Fatih AKAY
Yıl: 2013, Sayfa: 139
Jüri : Yrd. Doç. Dr. M.Fatih AKAY
: Prof. Dr. Hüseyin EKİZ
: Doç.Dr. Ulus ÇEVİK
: Doç.Dr. Mustafa GÖK
: Doç.Dr. S. Ayşe ÖZEL

Dağıtık ortak hafıza (DOH) sistemleri ölçeklenebilirlik, programlanabilirlik ve maliyet efektifliği açısından güçlü avantajlar sunsa da, bu sistemlerden yüksek performans elde etmek, sistemdeki işlemci sayısı artışı ile artan bellek erişim gecikmeleri nedeniyle beklenenden çok daha zordur. Ara bağlantı ağı mimarisi DOH sistemlerinin performansını belirlemede gittikçe daha da önemli bir hal almaktadır. İki boyutlu yayım tabanlı optik mimariler iyi bir çözüm sunmaktadır. Bu tezde, iki boyutlu optik tabanlı ağ üzerinde DOH sistemi OPNET Modeler benzetim programı kullanılarak modellenmiş ve sistem performansı mesaj öncelik ve birleştirme işlemleriyle normal, yoğun ve bit permütasyon trafik modelleri altında artırılmıştır. Benzetimi yapılan sistem, her biri işlemci, ön bellek ve dizin modüllerine hizmet eden bir sunucu, iki çıkış kanalı ve iki giriş kuyruğu içeren toplam 64 düğüm içerir. Yaratılan senaryolar; ortalama servis süresinin ortalama kanal bekleme süresine oranı, bir ön bellek bloğunun güncellenmiş olma olasılığı ve yazma isteklerinin tüm isteklere oranı gibi çeşitli DOH parametreleri altında ortalama ağ gecikmesini, ortalama çıkış kanalı bekleme süresini ve ortalama giriş kuyrukları bekleme süresini ölçer. Elde edilen sonuçlar, mesaj birleştirme işleminin ortalama çıkış kanalı bekleme süresini %13 ile %29 oranı arasında, mesaj öncelik işlemininse ortalama giriş kuyrukları bekleme süresini %7 ile %43 oranı arasında düşürdüğünü göstermiştir. Her iki tekniğin birlikte uygulandığı sistem üzerinde ise %7 ile %40 arasında ortalama ağ gecikmesi iyileştirmesi elde edilmiştir.

Anahtar Kelimeler: Çoklu mikroişlemcili mimariler, Dağıtık ortak hafıza, Mesaj birleştirme, Mesaj önceliği, Performans analizi

ACKNOWLEDGMENT

Foremost, I would like to express my sincere gratitude to my advisor Assist. Prof. Dr. M. Fatih AKAY, for his supervision guidance, encouragements, patience, motivation, useful suggestions and his valuable time for this work.

I would like to thank Prof.Dr. Hüseyin EKİZ, Assoc. Prof. Dr. Ulus ÇEVİK, Assoc. Prof. Dr. Mustafa GÖK and Assoc. Prof. Dr. S. Ayşe ÖZEL for serving on my committee.

Special thanks to OPNET Technologies, Inc. for letting me use the OPNET Modeler under the University Program and Cukurova University Scientific Research Projects Center for supporting this work (Project no: MMF2011D10).

I would also like to thank Dr. Constantine Katsinis for letting me use the information and figures of the SOME-Bus architecture.

Last but not the least, I would like to thank my family: my parents Lale, Gani and my brother Efe Abasıkeleş, for their endless support and encouragements for my life and career.

CONTENTS **PAGE**

ABSTRACT I
ÖZ II
ACKNOWLEDGMENTS III
CONTENTS IV
LIST OF TABLES VI
LIST OF FIGURES XII
l. INTRODUCTION 1
1.1. Distributed Shared Memory Multiprocessors 2
1.1.1. The Cache Coherence 4
1.1.2. The Memory Consistency 7
1.2. Interconnection Networks 8
1.2.1. Electronic vs Optical Interconnection Networks 9
1.2.2. 1D vs 2D Interconnection Networks 11
1.2.3. Broadcast-based Communication 11
1.2.4. The SOME-Bus Architecture 13
1.2.4.1. The 2D SOME-Bus Architecture 15
1.3. Contributions of the Thesis 18
2. RELATED WORK 21
3. EVALUATION METHODOLOGY 31
3.1. Synthetic Traffic Workload 31
3.1.1. The Temporal Distribution 32
3.1.2. The Spatial Traffic Pattern 32
3.2. Simulation Framework 33
3.2.1. Simulation Tools and System Architecture 33
3.2.2. Message Types 39
3.2.3. Simulation Parameters 43
3.3. The Proposed Algorithms 45
3.3.1. The Message Combining Algorithm 45
3.3.2. The Message Priority Algorithm 48

4. RESULTS AND DISCUSSION 51
4.1. Theoretical Analysis 51
4.2. Simulation Results 53
4.2.1. The Base System 55
4.2.2. The Base System with the Message Combining Algorithm 58
4.2.3. The Base System with the Message Priority Algorithm 63
4.2.4. The Base System with the Message Priority and the Message Combining Algorithm 68
5. CONCLUSION 81
REFERENCES 85
BIOGRAPHY 99
APPENDICES 100

LIST OF TABLES **PAGE**

Table 3.1. Message types in the simulation 39

Table 3.2. The directory protocol messages.....41

Table 3.3. Parameters of the simulated 2D SOME-Bus system.....43

Table 3.4. Estimation of the size and the decreasing rate of the resulting message for the two messages combined under the first scenario 46

Table 3.5. Estimation of the size and the decreasing rate of the resulting message for the two messages combined under the second scenario ...47

Table 3.6. Average of the decreasing rates of the first and the second scenario for two messages combined 47

Table 3.7. The selected messages according to the priority type 49

Table 4.1. Maximum number of messages in the input queues of the base system with message combining algorithm, for different values of traffic pattern, P(M) and P(W) 118

Table 4.2. Maximum number of messages in the horizontal output channels of the base system with message combining algorithm, for different values of traffic pattern, P(M) and P(W) 118

Table 4.3. Maximum number of messages in the vertical output channels of the base system with message combining algorithm, for different values of traffic pattern, P(M) and P(W) 119

Table 4.4 Maximum number of messages in the input queues of the base system with message priority algorithm, for different values of traffic pattern, P(M) and P(W) 119

Table 4.5. Maximum number of messages in the horizontal output channels of the base system with message priority algorithm, for different values of traffic pattern, P(M) and P(W) 120

Table 4.6. Maximum number of messages in the vertical output channels of the base system with message priority algorithm, for different values of traffic pattern, P(M) and P(W) 120

Table 4.7. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.2 under UN traffic 128
Table 4.8. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.2 under UN traffic 128
Table 4.9. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.2 under UN traffic 128
Table 4.10. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.4 under UN traffic 129
Table 4.11. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.4 under UN traffic 129
Table 4.12. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.4 under UN traffic 129
Table 4.13. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.6 under UN traffic 130
Table 4.14. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.6 under UN traffic 130
Table 4.15. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.6 under UN traffic 130
Table 4.16. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.2 under HR traffic 131

Table 4.17. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.2 under HR traffic 131
Table 4.18. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.2 under HR traffic 131
Table 4.19. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.4 under HR traffic 132
Table 4.20. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.4 under HR traffic 132
Table 4.21. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.4 under HR traffic 132
Table 4.22. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.6 under HR traffic 133
Table 4.23. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.6 under HR traffic 133
Table 4.24. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.6 under HR traffic 133
Table 4.25. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.2 under BR traffic 134
Table 4.26. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.2 under BR traffic 134

Table 4.27. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.2 under BR traffic 134
Table 4.28. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.4 under BR traffic 135
Table 4.29. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.4 under BR traffic 135
Table 4.30. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.4 under BR traffic 135
Table 4.31. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.6 under BR traffic 136
Table 4.32. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.6 under BR traffic 136
Table 4.33. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.6 under BR traffic 136
Table 4.34. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.2 under PS traffic 137
Table 4.35. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.2 under PS traffic 137
Table 4.36. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.2 under PS traffic 137

Table 4.37. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.4 under PS traffic 138
Table 4.38. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.4 under PS traffic 138
Table 4.39. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.4 under PS traffic 138
Table 4.40. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.6 under PS traffic 139
Table 4.41. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.6 under PS traffic 139
Table 4.42. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.6 under PS traffic 139

LIST OF FIGURES **PAGE**

Figure 1.1. Tradeoff between latency and bandwidth 6
Figure 1.2. The receiver array. 14
Figure 1.3. The 2D SOME-Bus Architecture. 16
Figure 1.4. Node block diagram. 17
Figure 3.1. State diagram of the acb_fifo model in OPNET Modeler..... 35
Figure 3.2. A 2x2 DSM model for the 2D SOME-Bus architecture 36
Figure 3.3. Reply forwarding directory protocol diagram of a read/write operation to a Modified cache state block..... 38
Figure 3.4. Reply forwarding directory protocol diagram of a write operation to a Shared cache state block 38
Figure 3.5. Average of the decreasing rates according to the combining messages types for three messages combined..... 102
Figure 3.6. Average of the decreasing rates according to the combining messages types for four messages combined..... 103
Figure 4.1. Message path of the reply forwarding protocol for a read request to a (a) Shared / Invalid and (b) Modified block..... 52
Figure 4.2. Message path of the reply forwarding protocol for a write request to an (a) Invalid, (b) Modified and (c) Shared block 53
Figure 4.3. Average network response time of the base system for different values of T/R, for (a) P(W) = 0.2, (b) P(W) = 0.4, (c) P(W) = 0.6 and P(M) = 0.2, 0.4 and 0.6 under UN traffic..... 104
Figure 4.4. Average network response time of the base system for different values of T/R, for (a) P(W) = 0.2, (b) P(W) = 0.4, (c) P(W) = 0.6 and P(M) = 0.2, 0.4 and 0.6 under HR traffic..... 105
Figure 4.5. Average network response time of the base system for different values of T/R, for (a) P(W) = 0.2, (b) P(W) = 0.4, (c) P(W) = 0.6 and P(M) = 0.2, 0.4 and 0.6 under BR traffic 106

Figure 4.6. Average network response time of the base system for different values of T/R, for (a) P(W) = 0.2, (b) P(W) = 0.4, (c) P(W) = 0.6 and P(M) = 0.2, 0.4 and 0.6 under PS traffic 107
Figure 4.7. Average channel waiting time of the base system (vertical axis) for different values of T/R (horizontal axis), for (a) P(M) = 0.2, (b) P(M) = 0.4, (c) P(M) = 0.6 and P(W) = 0.2, 0.4 and 0.6 under UN traffic 108
Figure 4.8. Average channel waiting time of the base system (vertical axis) for different values of T/R (horizontal axis), for (a) P(M) = 0.2, (b) P(M) = 0.4, (c) P(M) = 0.6 and P(W) = 0.2, 0.4 and 0.6 under HR traffic 108
Figure 4.9. Average channel waiting time of the base system (vertical axis) for different values of T/R (horizontal axis), for (a) P(M) = 0.2, (b) P(M) = 0.4, (c) P(M) = 0.6 and P(W) = 0.2, 0.4 and 0.6 under BR traffic 109
Figure 4.10. Average channel waiting time of the base system (vertical axis) for different values of T/R (horizontal axis), for (a) P(M) = 0.2, (b) P(M) = 0.4, (c) P(M) = 0.6 and P(W) = 0.2, 0.4 and 0.6 under PS traffic 109
Figure 4.11. Average network response time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic..... 110
Figure 4.12. Average network response time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under HR traffic..... 111
Figure 4.13. Average network response time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under BR traffic 112
Figure 4.14. Average network response time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under PS traffic 113

Figure 4.15. Average channel waiting time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic 114
Figure 4.16. Average channel waiting time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under HR traffic 115
Figure 4.17. Average channel waiting time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under BR traffic 116
Figure 4.18. Average channel waiting time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under PS traffic 117
Figure 4.19. Average network response time of the base system with the message priority algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic..... 121
Figure 4.20. Average network response time of the base system with the message priority algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under HR traffic..... 122
Figure 4.21. Average network response time of the base system with the message priority algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under BR traffic 123
Figure 4.22. Average network response time of the base system with the message priority algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under PS traffic 124
Figure 4.23. Average channel waiting time of the base system with the message priority algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic..... 125
Figure 4.24. Average input waiting time of the base system with the cache priority (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic 126

Figure 4.25. Average input waiting time of the base system with the directory priority (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic 126
Figure 4.26. Average input waiting time of the base system with the directory & cache priority (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic 127
Figure 4.27. Average input waiting time of the base system with the critical path priority (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic 127

1. INTRODUCTION

The general trend in microprocessor design for improving the performance is embedding an increasing number of cores in processor chips rather than increasing the clock frequency, which is a traditional approach (Olukotun and Hammond, 2005). This is due to the diminishing power efficiency of high frequency processors that results in the increasing power consumption problem of modern processors. Since further frequency scaling can no longer yield significant performance improvements without unreasonably increasing power consumption, the concept of single processor on a chip is likely to wane in the very near future (Burns and Gaudiot, 2005). Instead, multiprocessors or multithreaded architectures will likely be the baseline architectures of future network processors (Yi et al., 2013).

Due to the scaling difficulties of uniprocessor architectures, chip multiprocessors (CMPs) have become the dominant design approach (Sanchez et al.,2010). Thanks to the increasing transistor budget provided by Moore' s Law, it is expected that the number of cores per chip will grow accordingly. CMPs with tens of cores are already being manufactured (Bell et al. 2008; Tremblay and Chaudhry 2008), including prototypes (Lodde et al., 2012) (such as the 48-core Single-chip Cloud Computer developed by Intel (Rattner, 2013) and real products (such as the Tilera' s 100-core TILE-Gx100 processor (Doud, 2011)) while chips with hundreds of cores will be available in the near future (Intel, 2008) (Sanchez et al.,2010).

High performance system on chip and CMPs entail an increase in the number of required processor elements. However, the communication performance between these processor elements is degraded because of the bus-system limitations. Shared buses become bottlenecks for system communication as the number of processor elements increases (Alhussien et al., 2011). As a general trend, on-chip computing platforms are evolving from single-core bus-based systems to multi-core network based systems (Horowitz and Dally, 2004; Borkar, 2007), which are referred to as multi-core Network-on-Chips (NoCs). NoC is a new technology developed to overcome bus limitations and feature more scalability. For instance, in 2007, Intel researchers announced their research prototype multi-core NoC architecture. It

contains 80 tiles arranged as a 10 x 8 2D mesh network (Vangal et al., 2007). Another trend exhibits that the embedded memory content in System-on-Chips (SoCs) increases from 20% 10 years ago to 85% of the chip area today and will continue to increase in the future (Marinissen et al., 2005). Technology advances, such as high density embedded memory and 3D integration, make integrating distributed memories on a chip become feasible and preferable. For instance, in 2008, Loh proposed a 3D-stacked memory architecture where each core has its own memory bank (Loh, 2008); in 2010, a Japanese group presented a technique which can integrate 64 NAND Flash memory dies in a 3D-stacked chip (Sugimori et al., 2010). Combining these two trends, it is a question to organize and manage so many on-chip storage resources. In multi-core NoCs, especially for medium and large scale system size, a distributed memory organization is more suitable to the multi-core architecture and the general on-chip communication network, since the centralized memory organization has already become the bottleneck of performance, power and cost. In the distributed memory organization, memories are distributed in all nodes, featuring good scalability and fair contention and delay of memory accesses. One critical question of the distributed memory organization is whether the memory space is shared or not. Keeping memory only local and private is an elegant architecture solution but ignores rather than solves the issues of legacy code and programming convenience. A lot of legacy software assumes a shared memory model. Most programmers find it easier to express themselves within a shared memory programming model rather than a model that is strictly based on explicit message passing. Consequently, it is believed that it' s essential to support DSM on multi-core NoCs (Chen et al., 2013).

1.1. Distributed Shared Memory Multiprocessors

Shared memory is an attractive programming model for interprocess communication and synchronization in multiprocessor computations. In the past decade, a popular research topic has been the design of systems to provide a shared memory abstraction of physically distributed memory machines. This abstraction,

known as Distributed Shared Memory (DSM), has been implemented both in software (e.g., to provide the shared memory programming model on networks of workstations) and in hardware (e.g., using cache coherence protocols to support shared memory across physically distributed main memories) (Pang et al., 2007).

DSM systems represent a successful hybrid of shared memory multiprocessors and distributed computer systems. They provide the shared memory abstraction in systems with physically distributed memories, and consequently combine the advantages of both approaches (Yahya and Bader, 2007). This approach hides the underlying communication mechanism from the programmer, which is an advantage as the programming model followed can be much simpler (Ramesh and Varadarajan, 2007), so the ease of programming and the portability typical for shared memory systems, as well as the scalability and cost effectiveness of distributed memory systems, can be achieved with less engineering effort (Chennareddy and Deka, 2006). Roughly speaking, it consists of using read and write operations for inter-process communication, thus hiding the particular communication technique employed by the programmers to avoid the need of involvement in the management of messages (Jimenez et al.,2008).

DSM multiprocessors are converging to a family of architectures that resemble generic system architecture. A typical DSM system consists of a number of processing nodes connected by some kind of interconnection network. A node can be a uniprocesor or a small bus based multiprocessor. Each processor is equipped with private primary and usually secondary cache memories. Each node also contains a physically local memory module which is partially or entirely mapped to the DSM global address space (Toncev et al.,2004) and a communication controller that is responsible for managing the communication both within and between nodes (Chaudhuri et al., 2003). On each access to shared space, hardware must determine if the requested data is in the local memory, and if not, the data must be copied from remote memory. Actions are also needed when data is written in shared space to preserve the coherence of shared data (Katsinis, 2004).

The key strength of DSM systems is that communication occurs implicitly as a result of conventional memory access instruction (i.e. loads and stores) which

makes them easier to program. The programmer is given the illusion of a large global address space encompassing all available memory, eliminating the task of explicitly moving data between processes located on separate machines (Jiang and Chaudhary, 2002). Because of these advantages, DSM is one of the most attractive solutions for building large-scale, high-performance multiprocessor systems.

Although DSM systems present strong advantages in terms of scalability, programmability and cost effectiveness, obtaining high performance out of these systems turns out to be much more cumbersome than expected. One of the fundamental communication problems in DSM systems that significantly affect scalability is the increase in remote memory access latency as the number of processors increase in the system (Kodi and Louri, 2004), which in turn is usually the bottleneck of the performance of these systems (Hung and Liu ,2007; Kodi and Louri, 2004). Naive exploitation of shared memory can cause performance problems, because the performance of DSM multiprocessors is often limited by the amount of time spent waiting for remote memory accesses to be satisfied. When the overhead associated with accessing remote memory impacts performance, programmers are forced to spend significant effort managing data placement, migration, and replication – the very problems that shared memory is designed to hide. Thus, the value of DSM multiprocessor architectures is directly related to the extent to which observable remote memory latency can be reduced to an acceptable level (Carter et al., 1998).

1.1.1. The Cache Coherence

Shared multiprocessor systems, which are widely used in modern commercial and scientific computing infrastructures nowadays (Liu, 2007; Phillips, 2007; Sinharoy, 2009), are comprised of several processors that share a global memory (Cuesta et al., 2012). In a multicore processor architecture, the performance of the memory hierarchy is more important than ever, since multicore processors share memory resources (Yi et al., 2013). Caches have been widely used in these systems to improve the performance of the system. Since each processor holds a private

cache, they require a cache coherence protocol to consistently view the global memory (Cuesta et al., 2012).

Cache coherence protocols manage reads and writes to shared memory locations. Given a memory location, a protocol should deliver read values that can be interleaved into a total order of writes to that same location (Vantrease et al., 2011). Simply coherence can be defined as retrieving always the most recent value for any data. Maintaining this feature solely by the software makes the programmer' s task extremely difficult. Modern multiprocessors solve the cache coherence problem in hardware by implementing cache coherence protocols (Nikolov and Lerato, 2010). The performance and scalability of multiprocessors depend to a large extent on the performance and scalability of the cache coherence protocol. Therefore, in order to provide fast and scalable multiprocessors, cache coherence protocols must incur low latency and have high scalability (Cuesta et al., 2012).

There are two main classes of hardware protocols, snoopy and directory based protocols. Snoopy protocols use broadcast medium and hence apply to a smaller-scale bus-based multiprocessors (Nikolov and Lerato, 2010). A simple and elegant solution to cache coherence arises from the very nature of a bus. The bus is a single set of wires connecting several devices, each of which can observe every bus transaction. When a processor issues a request to its cache, the cache controller examines the state of the cache and takes suitable action, which may include generating bus transactions to access memory. Coherence is maintained by having all cache controllers " snoop" on the bus and monitor the transactions (Chandramouli and Iyer, 2004).

Directory cache coherence protocols represent the most scalable alternative. Unlike broadcast based protocols, traditional directories keep track of every memory block in the system, which enables the protocol to easily locate the cached copies without generating large amounts of network traffic (Cuesta et al, 2013). On a miss, a processor sends a coherence message over an interconnect to a directory, which often forwards message(s) to processor(s) currently caching the block. These processors may forward data or acknowledgments to the requesting processor and/or directory (Mukherjee and Hill, 1998). The two major performance goals of a directory

protocol are: (i) to reduce the number of network transactions generated per memory operation, which reduces the bandwidth demands placed on the network and the communication assists; and (ii) to reduce the number of actions, especially network transactions, that are on the critical path of the processor, thus reducing uncontended latency. Network latency is important because memory access time depends on that latency (Culler et al., 1999).

Small scale multiprocessors frequently use a snoopy cache coherence protocol (Chandramouli and Iyer, 2004), while large scale multiprocessors use a directory-based cache coherence protocol (Laudon and Lenoski, 1997). Directory-based cache coherence protocols offer a scalable performance path beyond snooping based ones by allowing a larger number of processors to share a single global address space over physically distributed memory (Acacio et al., 2002a). Although directory protocols can scale to a large number of processing nodes, these systems are slower than their snoopy counterparts (Chandramouli and Iyer, 2004). In brief, snooping protocols consume bandwidth while directory protocols suffer from remote memory access latency. This latency/bandwidth tradeoff is especially important for the commercial workloads that dominate the current use of multiprocessor servers, since many of these workloads exhibit high cache miss rates and a large fraction of cache-to-cache misses (Alameldeen et al., 2002). As illustrated in Figure 1.1., system designers must choose between the high bandwidth use of snooping protocols and the high sharing latency of directory protocols. An ideal approach is combining the low latency of broadcast snooping with the bandwidth efficiency of directory protocol (Martin et al., 2003a).

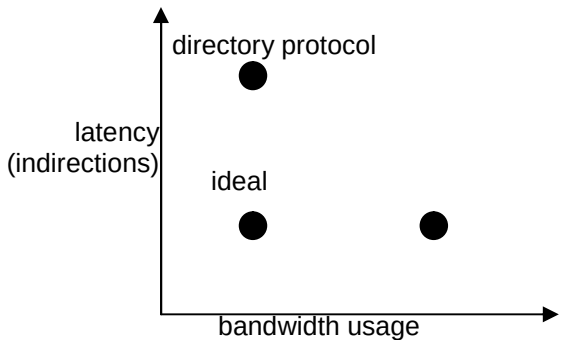


Figure 1.1. Tradeoff between latency and bandwidth

1.1.2. The Memory Consistency

Optimizing the performance of computing systems has always been a concern for hardware and software technology, and therefore, given that the latency of memory operations is one of the key performance factors, it is not surprising that techniques have been developed very early to minimize the cost of these operations in program execution. As a typical optimization technique, DSM multiprocessors use caches, thus benefiting from a parallel architecture where memory operations run concurrently with other instructions. These techniques, as well as various optimizations performed by compilers, are naturally intended to preserve the semantics of sequential code. However, problems arise when executing concurrent code sharing the memory. In particular, many algorithms designed for synchronization purposes (mutual exclusion, producer-consumer) fail to achieve their goal in an optimized implementation setting (Boudol and Petri, 2009).

The memory consistency model is a contract between the programmer and parallel system. The programmer follows the rules that are guaranteed by the parallel system to get the predictable results of the shared memory operations. Shared memory access latency can be reduced by the hardware and software optimizations in the system architecture. These performance optimizations can reorder the shared memory operations and the system may give unexpected results. For the expected results, these reordering should be controlled carefully. Different memory consistency models enforce different ordering constraints on the shared memory operations. The sequential consistency enforces strict ordering constraints on the shared memory operations. The IBM 370, total store ordering and processor consistency models eliminate the ordering constraints between write followed by a read to a different location. The partial store ordering model also in addition removes the ordering constraints among writes to the different locations. The weak consistency, release consistency, Alpha, relaxed memory order and PowerPC models relax the program order requirement among all the shared memory operations to the different locations (Naeem et al., 2011). Due to the minimizing the coherence traffic (Gelado et al., 2010), many contemporary shared-memory architectures use relaxed

memory ordering models that allows the processors to reorders loads and stores by the same thread if they target different addresses (Burckhardt et al., 2007).

1.2. Interconnection Networks

Over the last four decades, progress in high performance computing (HPC) systems has been dominated by remarkable advances in semiconductor technologies, namely improved fabrication technologies, circuit design techniques, and processor microarchitectures. These advances, manifested in Moore' s law, have led to the extremely high performance presented by today' s CMOS-based microprocessors. Large scale distributed systems, while benefiting from this progress; face a severe problem, exacerbated with every generation: the interconnection network, whose performance is crucial to the overall system performance, does not keep up with Moore' s law. The physical laws governing the propagation of signals in electrical transmission lines are beginning to limit the performance of communication systems at high data rates (several Gb/s and higher). Dielectric losses and losses caused by the skin effect limit the transmission distance, requiring large power and dedicated circuits to overcome inter-symbol interference (ISI) (Miller, 2000). The resulting systems are expensive, power hungry, and suffer from a large latency that impacts the overall system performance (Dally and Towles, 2004).

Three critical parameters determine the performance of an HPC interconnection network: latency, bandwidth, and power dissipation. The latency is dominated by the propagation velocity across the transmission lines (time of flight) and by the queueing latency, which is inevitable for systems with large port counts. As systems scale in port counts and clock speeds, the latency grows in absolute terms and even more so in terms of processor clock speeds (Dally and Towles, 2004). Many HPC systems are currently using latency hiding and masking techniques to overcome this problem (Hennessy and Patterson, 2002), but these techniques incur performance and power costs and cannot be used in every application.

Modern processors further challenge network designers by demanding ever-growing off-chip bandwidth requirements (e.g., 512 Gb/s in the IBM Cell Broadband

Engine processor (Kistler et al., 2006)). Providing this bandwidth for remote memory accesses and for interprocessor communication become extremely challenging and power-consuming.

Finally, the power consumed by electronic interconnects in HPC systems to meet the bandwidth and latency requirements is becoming the most critical design constraint. The power expended in HPC systems on computation and communications grows very quickly with the data rates (Mudge, 2001), and the associated cost and heat dissipation problems have become limiting factors in the design and deployment of many such systems (Barroso et al., 2003). The power dissipated by the interconnection network is a large fraction of the total system power. As systems become spatially larger, a larger amount of power is required to overcome the losses in transmission lines. This is done either by periodic regeneration or by sophisticated signal processing techniques such as pre-emphasis and equalization (Liu and Lin, 2004). Additional concerns such as cabling density, bending radii, and cooling airflow also present important and growing challenges to HPC interconnection networks designers (Shacham and Bergman, 2009).

1.2.1. Electronic vs Optical Interconnection Networks

In DSM multiprocessor machines all the memory of the system is distributed among its nodes. Nodes can access memory on other nodes in a software transparent way. The interconnection network is thus part of the memory hierarchy, and therefore high network latencies cause a significant performance bottleneck in the program execution. Interprocessor communication is already one of the main bottlenecks in current multiprocessors. This situation will become worse in the future, a result of increasing clock speeds, the rapid growth in instruction level parallelism and the use of multiple cores per die and multiple threads per core (Krewell, 2005).

Electronic interconnection networks have thus far been capable of coping with the communication demands of today’ s applications, however further scaling of the number of cores and memory requirements of applications may not be able to be

matched with equivalent communication improvements (Miller, 2009). This inability of electronic interconnects to scale in performance is a consequence of the large amount of power that is dissipated by the electronic interconnects and the limited heat-dissipation capabilities of current packaging technology. A previous study has shown that over 50% of the dynamic power dissipated in some high performance microprocessors comes from the interconnects exclusively (Magen et al., 2004). These problems will be further exacerbated as the requirements of CMP systems continue to grow, exemplifying the need for a new interconnect technology that can deliver energy-efficient high-bandwidth communications (Chan et al., 2011).

Optics is a great candidate to introduce fast interconnection networks in the architecture of the multiprocessor systems. Using optical interconnects at the scale of link lengths found in multiprocessor machines (up to a few meters), an increase in connectivity and higher communication bandwidths can be expected, whereas the design of conventional electrical interconnects is limited by the tradeoff between interconnection length and bit rate. The high operating frequency of light tends to virtually eliminate frequency dependent cross-talk, and the inherent voltage isolation is also a highly demanded characteristic that will improve the signal integrity of the high speed communication channels. Finally, a very important aspect of optical interconnects is their inherent ability to switch the light paths easily in a data transparent way. This is paving the way towards adaptable network topologies (Artundo et al., 2006).

Optical interconnects have fundamental advantages compared to electrical interconnects, particularly in delay and potential bandwidth and significant progress in the technology has been made in recent years (Shacham and Bergman, 2007; Xue et al., 2010). It is now believed that optical interconnection has more potential of shortening delay than electric interconnection. Optical interconnection is also expected to provide much larger bandwidth density for buses and networks than electrical interconnection (Shacham et al., 2007). Wavelength-division multiplexing (WDM) technology provides a way to further increase data capacity (Haurylau et al., 2006). On the other hand, successfully fabricating commercial-production on chip optical interconnects requires considerable improvement in lower energy

consumption in both electro-optical and opto-electric signal conversions. The high cost for assembling optical parts for integration has been another serious issue. However, recent advances in silicon photonics have been making cost-effective implementation of optical elements on a silicon chip possible (Ohashi et al.,2009).

1.2.2. 1D vs 2D Interconnection Networks

Ever since industry has turned to parallelism instead of frequency scaling to improve processor performance, multicore processors have continued to scale to larger and larger numbers of cores. It is believed that multi cores will have 1000 cores or more by the middle of the next decade. However, their promise of increased performance will only be reached if their inherent scaling challenges are overcome. Choosing a two dimensional (2D) interconnection network rather than a one dimensional (1D) equivalence increases the system scalability.

1D interconnection networks transport all the data in a plane rather than a volume. The advantage of moving to two dimensional data arrays can be quantified thus: in a real system, there will be a limit to the number of channels which can be accommodated which could come from the maximum size of the chip or from the distance across which the optics can image faithfully; if this limit is N for a 1D array, it will be N2 for a two dimensional network. Equivalently, a 2D network is much more compact than the 1D version having the same number of channels (Taylor and Midwinter, 1990).

1.2.3. Broadcast-based Communication

In a multicomputer network, processors often need to communicate with each other for various reasons, such as data exchange or event synchronization. Efficient communication has been recognized to be critical for HPC systems. One essential communication operator is the one-to-all broadcast, where a source node needs to send a message to every other node in the network. Broadcast has many applications, such as linear algebra algorithms, barrier synchronization, parallel graph algorithms,

parallel matrix algorithms, distributed table-lookup, fast Fourier transformation, and cache coherence (Tseng et al., 1999). Besides, broadcasting is an important collective communication operation in many applications which use parallel computing (Chen and Wang, 2005). Much research has given attention to the problems associated with achieving high performance computing in broadcasting and personalized communication on a variety of interconnection networks (Chen et al., 1996; Tseng, 1997).

Current state-of-the-art on-chip networks provide efficiency, high throughput, and low latency for one-to-one (unicast) traffic. The presence of one-to-many (multicast) or one-to-all (broadcast) traffic can significantly degrade the performance of these designs, since they rely on multiple unicasts to provide one-to-many communication. This results in a burst of packets from a single source and is a very inefficient way of performing multicast and broadcast communication. This inefficiency is compounded by the proliferation of architectures and coherence protocols that require multicast and broadcast communication. In the nature of coherency protocol, when a shared block is invalidated, an invalidate message must be multicast to all sharers. For unicast interconnects, the multicast is translated into several unicast messages. These unicast messages cause network congestion and may harm performance (Jerger et al., 2008). For avoiding redundant unicast invalidates, a broadcast bus can be augmented (Vantrease et al, 2008). There are many studies in literature that use broadcast-based coherence protocols, including Token Coherence (Martin et al., 2003), Intel' s next-generation QPI protocol (Kanter, 2007), AMD Opteron Protocol (Conway and Hughes, 2007) and Uncorq (Strauss et al., 2007). The TokenB protocol requires broadcasting of tokens that maintain ordering amongst requests, while QPI protocol supports unordered broadcasting between subsets of nodes. In AMD Opteron Protocol, order is maintained by communicating requests to an ordering point (memory controller) and then broadcasting from the ordering point.

A network which offers an alternative to switch-based networks relies on one-to-all broadcast, where each processor can directly communicate with any other processor. From the point of view of any processor, all other processors appear the same. Such a network allows data and operations to be structured in the application

code to better reflect the inherent parallelism and reduce message latencies. The most useful properties of such a network of workstations are high bandwidth (scaling directly with the number of workstations), low latency, no arbitration delay, and non-blocking communication. Such a network can be constructed using optoelectronic devices, relying on sources, modulators and arrays of detectors, all being coupled to local electronic processors. One implementation of such architecture is the Simultaneous Optical Multiprocessor Exchange Bus (SOME-Bus) (Katsinis, 2008). The Simultaneous Optical Multiprocessor Exchange Bus (SOME-Bus) (Katsinis, 2001) is a low-latency high-bandwidth optical interconnect with a unique feature that each processor is connected to the rest of the processors with a dedicated channel. Such properties distinguish it from other networks examined in the past (Katsinis and Hecht, 2003)

1.2.4. The SOME-Bus Architecture

The most useful properties of a parallel processor interconnection network are high bandwidth (scaling directly with the number of processors), low latency, no arbitration delay, and non-blocking communication. SOME-Bus directly connects each processor node to all other nodes and is capable of accommodating over one hundred nodes without suffering from bottlenecks.

It uses Bragg gratings within a fiber as narrow-band, inexpensive outcouplers and amorphous silicon detectors built as superstructures on CMOS devices. By using wavelength-division-multiplexing (WDM), it utilizes the higher bandwidth capabilities of optics while optimizing distribution of optical power.

One of its key features is that each node has a dedicated broadcast channel, realized by a specific group of wavelengths in a specific fiber, and an input channel interface based on an array of receivers which simultaneously monitors all channels, resulting in an effectively fully-connected network. Figure 1.2. shows the receiver array. A separate wavelength is used to insert the clock signal which is common to all data streams on a fiber. A very modest implementation of a system with 128 nodes using CMOS receivers, amorphous Si detectors, 32 fibers with 17 wavelengths

(4 per node plus common clock) and a clock rate of 155MHz can provide 77 MB/sec of bandwidth per channel. Using a more aggressive technology such as gallium arsenide lift-off with a clock rate of 1 GHz per wavelength, the same 32-fiber SOME-Bus interconnect can provide up to 500 MB/sec of bandwidth per node.

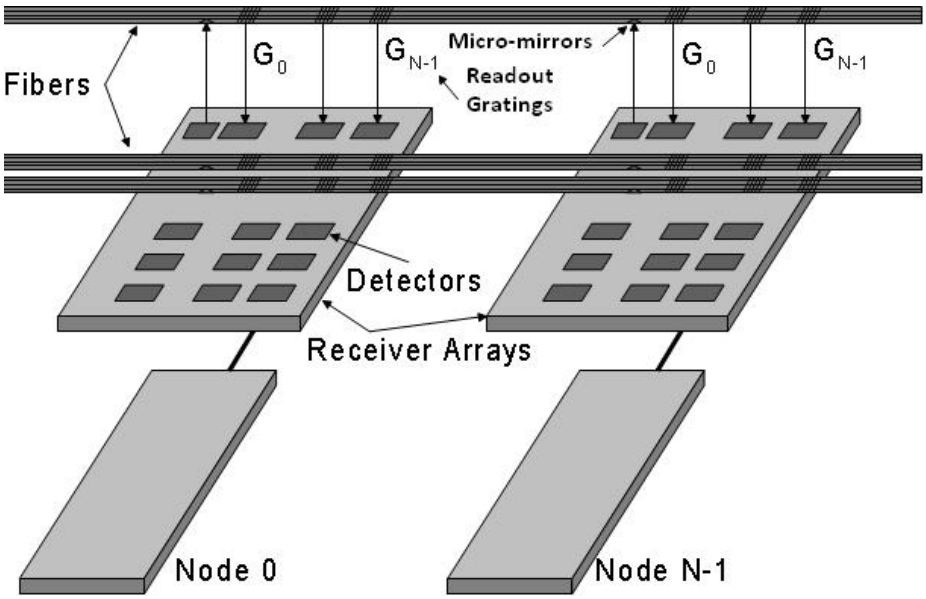


Figure 1.2. The receiver array

Messages exchanged between nodes contain a header field with information on the message type, length and destination address. The input channel interface performs address filtering, barrier processing, length monitoring and type decoding, and supports the arrival of messages from any number of nodes simultaneously, buffering them in a queue until the local processor is ready to remove them. One queue is associated with each input channel, allowing an arbitrary number of messages to arrive simultaneously. Synchronization messages are collected and processed at the receiver. In addition to recognizing its own individual address, the receiver logic can recognize multicast group addresses as well as broadcast addresses.

The SOME-Bus can most readily support a CC-NUMA system where the shared virtual address space is distributed across local memories which can be accessed both by the local processor and by processors from remote nodes, with different access latencies. Traffic on the interconnection network consists of data

messages, due to a cache miss on one (local) node, directed to the memory of another (remote) node, and additional messages which allow the caches to maintain data consistency. Although the SOME-bus can utilize software techniques for implementing cache coherence, it allows strong integration of the transmitter, receiver and cache controller hardware to produce a highly integrated system-wide cache coherence mechanism. This hardware-oriented system uses cache blocks to enforce coherence, resulting in reduced probability of false sharing and thrashing, compared to software-oriented systems which use larger blocks such as virtual memory pages (Katsinis, 2003).

1.2.4.1. The 2D SOME-Bus Architecture

In 2D SOME-Bus architecture (as shown in Figure 1.3.), each node is connected to one horizontal and one vertical SOME-Bus. At each node, an electro-optical converter consisting of a dual receiver and transmitter pair allows messages broadcast on one bus to be forwarded and broadcast (in a cut-through manner) on the bus in the other dimension.

If a node N_{ij} (connected to vertical bus V_i and horizontal bus H_j) sends a message to node N_{mn}, and either $i = m$ or $j = n$, then only one bus (vertical or horizontal, respectively) is used. The message header includes the identification of the destination node and the message is broadcast on the proper bus. If both $i=m$ and $j=n$, then the message is broadcast first on horizontal bus H_j to node N_{mj} with the header containing an indication that node N_{mj} broadcast the message on vertical bus V_m so it can be delivered to its final destination, node N_{mn}. By symmetry, the source node may choose to broadcast the message first on vertical bus V_i to intermediate node N_{in} for rebroadcast on horizontal bus H_n. The design of the header allows for the specification of multiple destinations so that the full capabilities of the architecture may be exploited. In the simplest configuration, the message header contains a list of final destinations. The message is broadcast first on the horizontal bus at the source node and is delivered to all required nodes on that horizontal bus. Each of these nodes examines the header to determine if it is specified as a

destination of the message and/or if it must rebroadcast the message on its own vertical bus. If no rebroadcast from one dimension to the other is necessary, then the system operation is equivalent to the operation of the 1D SOME-Bus (Katsinis and Nabet, 2004).

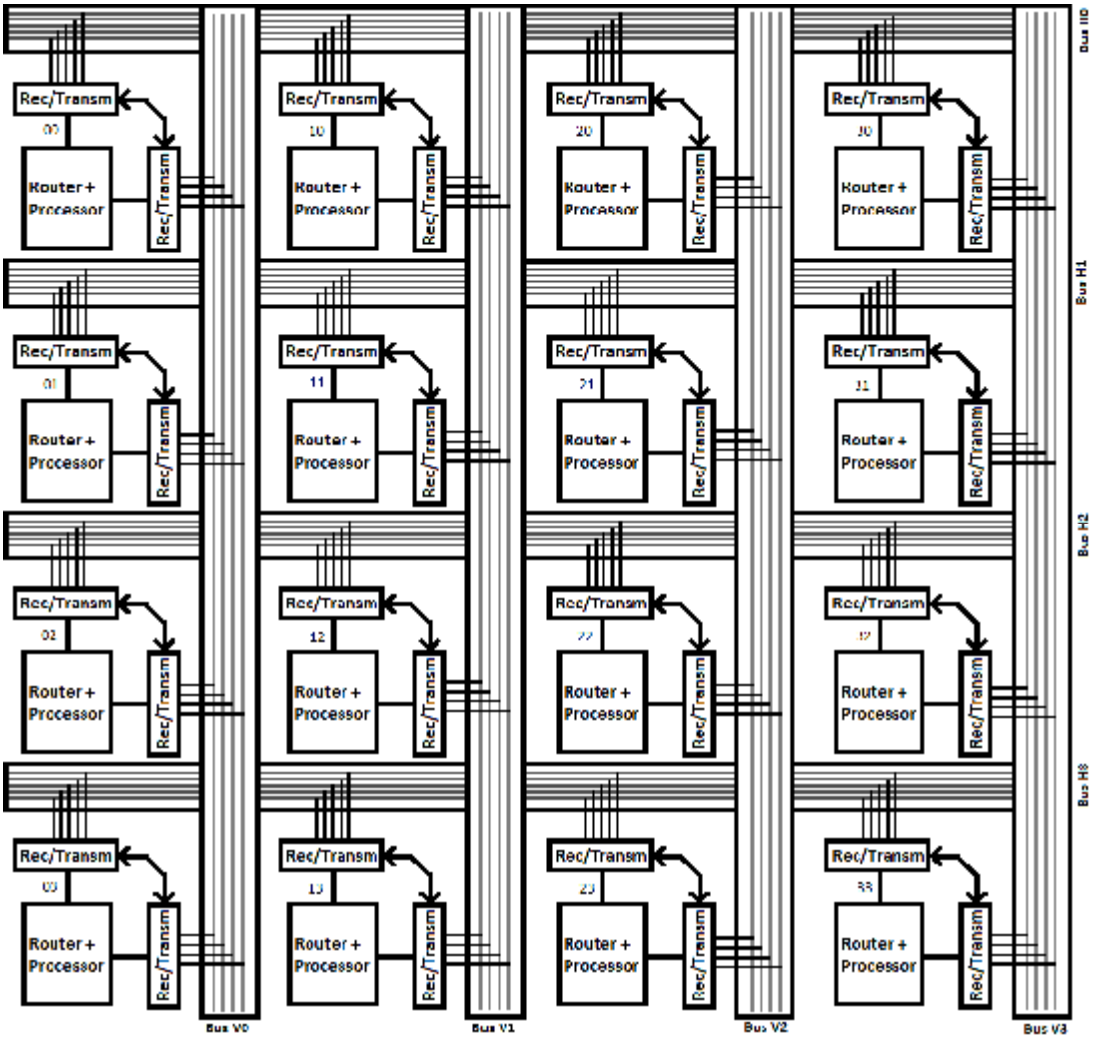


Figure 1.3. The 2D SOME-Bus Architecture

The 2D SOME-Bus bears a superficial resemblance to regular mesh or hypercube architectures augmented with spanning buses. Such architectures have been proposed and studied in the past, and include augmented mesh architectures (Chung, 1995), multiple hierarchical buses connecting meshes (Pan et al., 2001), an r-dimensional hyperbus for a SIMDarchitecture (Tsai et al., 1997), hypercubes

connected through hyperchannels (Louri and Weech, 1998) a ring architecture augmented with regular buses (Dighe et al., 1996) , and hypermeshes using crossbars (Szymanski, 1995). A common characteristic of these architectures is that although multiple wavelengths are used, multiple senders use the same wavelength, requiring contention resolution. Only one processor may transmit on a bus and therefore the architecture must contain additional arbitration hardware or be targeted for applications with sufficient regularity where the bus transactions may be predicted and scheduled at compile time. In contrast, the SOME-Bus architecture allows several (and even all) processors to transmit at the same time.

Figure 1.4. shows the block diagram of a node. There are two sets of receivers and transmitters, one associated with a vertical and a horizontal Bus. A router connects the receiver/transmitter pairs and the local processor. The router allows the processor to access the queue in each transmitter, and one queue in each receiver. The circuitry in the router allows access to the queues either in random- access or in a polling fashion. The router also allows messages to be forwarded from one receiver to the opposite transmitter for rebroadcast on the other dimension. The router moves messages in electronic form (16 or 32-bit words) and does not need to operate at photonic speeds.

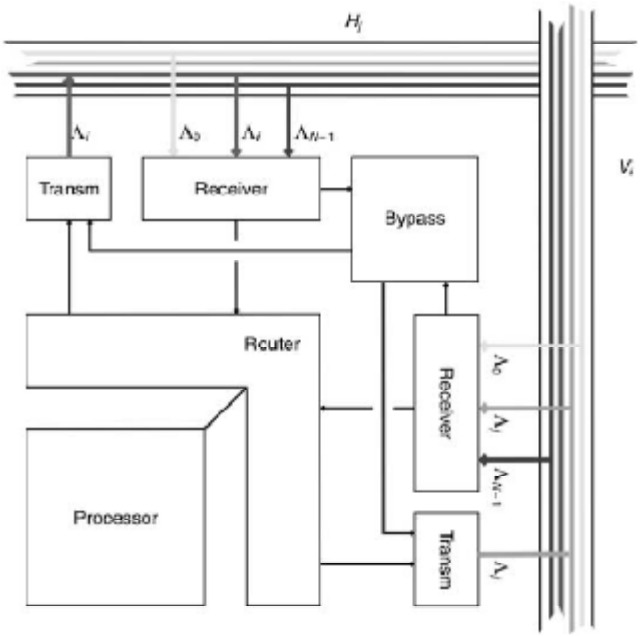


Figure 1.4. Node block diagram

1.3. Contributions of the Thesis

Improving the performance of shared memory architectures and reducing the coherence traffic have been attracting the researchers' attention over a decade. Some of them focuses on the underlying interconnection network architecture (Cheng et al., 2006; Flores et al., 2010; Rutgers et al., 2012), while the others propose cache coherence protocol optimizations (Eisley et al., 2006; Ros et al., 2010; Pascual et al., 2008; Deshpande et al., 2010; Abasıkeleş and Akay, 2010). Other architectural improvements, including cache organizations, are also gain a place in literature (Kim et al., 2009; Chishti et al., 2005; Chen et al., 2013).

With the high cost of remote memory access, DSM systems achieve higher performance by reordering (giving priority) or overlapping (combining) the servicing of multiple memory or communication operations from a processor (Culler et al., 1999). In order to achieving low latencies, messages can be combined within the interconnect (Protic et al., 1996). Priorities may be used to differentiate memory transactions, decreasing the response time to higher priority messages, as block invalidation (Chaves et al., 2011).

Message priority has been extensively used for NoCs. Some of the studies gives higher priority to low latency and low bandwidth messages (Beigne et al., 2005; Carara et al., 2009), while the others give priorities according to rate, deadline, laxity (Lu et al., 2005) and message criticality (Manolache et al., 2005). Priority technique is also used for reducing the coherence traffic by giving higher priority to short packets (Bolotin et al., 2007; Chaves et al., 2011), forwarded requests and all replies (Barroso et al., 2000). Moreover, Karlsson and Brorsson (2003) have used critical message priority on software DSM systems. The common point of these studies is using a switch-based network as underline architecture.

Message combining has been used for distributed memory systems (Shafer and Ghose, 1998), one dimensional arrays (Souravlasa and Roumeliotisb, 2008) and multistage interconnection networks (Hanawa et al., 1996) as well as for shared memory systems both on switch-based (Ma et al., 2012; Dvorak, 2008; Dvorak and Joras, 2010; Dai and Panda, 1996) and broadcast-based architectures (Katsinis and

Hecht, 2000; Katsinis and Hecht, 2004). Message combining is used for 1D broadcast-based optical interconnection network with multiple server models and its performance is evaluated under UN and hot spot traffic patterns (Katsinis and Hecht, 2004).

As distinct from the previous studies in literature, after modeling DSM on 2D broadcast-based architecture with one server model, we have used both message combining and message priority techniques together for reducing coherence traffic. We have implemented critical path messages priority, which is extensively used in literature, as well as cache, directory and cache with directory messages priority, as distinct from the previous studies in literature. Four types of message priority (cache priority, directory priority, cache & directory priority and critical path priority) and three types of message combining (two messages, three messages and four messages combining) techniques are implemented on the base system both individually and jointly. Messages are prioritized in the input queues and combined in the output channels. Both for priority and combining operations, memory consistency model and cache coherence are preserved.

The system is simulated by using OPNET Modeler simulation program using a directory-based cache coherence protocol called reply forwarding protocol and a relaxed memory consistency model, which allows both read and write operations to cut in line previous reads and writes without violating cache coherence. The simulated system contains 8×8 nodes. Each node has a server that processes directory, cache and processor messages, two output channels and two input queues, one for horizontal and the other for vertical dimension. Scenarios are run to measure average network response time, average input waiting time and average channel waiting time. Average network response time is the time interval between the instant when a cache miss causes a message to be enqueued in the output channel until the instant when the corresponding data or acknowledge message arrives at the input queue. Average input waiting time is the time interval between a message is enqueued in the input queue of a node until it gets serviced while average channel waiting time is the time interval between the message is enqueued in the output channel queue until it gets serviced. Wherefore the underlying architecture is 2D,

average channel waiting time is calculated as the average of average horizontal and average vertical channel waiting times.

The results show that the combining mechanism reduces average channel waiting time up to 29%, while the priority mechanism reduces average input waiting time up to 43%. The performance increasing rate of average network response time due to the message combining is higher for the higher T/R values than the lower-to-middle T/R values. The message combining algorithm yields the best performance for four messages combining for all traffic patterns and performs the best under bit permutation traffic models. The performance of the message priority algorithm on average network response is higher for the lower T/R values regardless of the traffic patterns, the other DSM parameter values and the combining types and decreases as T/R increases.

With the application of the message priority and the message combining mechanisms together up to 40% performance increase for average network response time has been gained. The message priority algorithm is the most dominant factor on the performance of average network response time for the lower values of T/R, while the message combining flows through on the performance of average network response time for the higher T/R values. When the priority and the combining algorithms are implemented on the base system together, the obtaining performance is higher than the individual algorithm performances. The directory priority performs better under bit permutation traffic models, while the critical path priority usually performs the best through the entire priority algorithms. The more number of messages combining leads to the more performance increase for all traffic patterns regardless of the other DSM parameters. Bit permutation traffic models get an edge over the other traffic patterns as P(W) increases for the entire combination types.

2. RELATED WORK

Modern DSM machines face an ever-growing disparity between processor cycle times and interconnect latencies (Somogyi et al., 2004). The widening gap between processor and memory performance is the main bottleneck for modern computer systems to achieve high processor utilization (Xue et al., 2010). Semiconductor fabrication advances and circuit innovations have led to dramatic increases in operating frequencies, and recent architectural developments—such as chip multiprocessors and simultaneous multithreading—place an even greater load on memory subsystems. DSMs face all the challenges associated with uniprocessor designs, as well as coherence requirements that tax the interconnect (Somogyi et al., 2004).

To improve the performance of the shared memory architectures and reducing the coherence traffic, a variety of studies, most of which uses simulation technique for the performance evaluation, have been proposed in literature. Some of them focuses on the underlying interconnection network architecture (Cheng et al., 2006; Flores et al., 2010; Rutgers et al., 2012), while the others proposes cache protocol optimizations (Eisley et al., 2006; Ros et al., 2010; Pascual et al., 2008; Deshpande et al., 2010). Other architectural improvements are also gain a place in literature (Kim et al., 2009; Chishti et al., 2005; Chen et al., 2013).

Cheng et al. (2006) propose an interconnect composed of wires with varying latency, bandwidth, and energy characteristics, and advocate intelligently mapping coherence operations to the appropriate wires. They present a comprehensive list of techniques that allow coherence protocols to exploit a heterogeneous interconnect and evaluate a subset of these techniques to show their performance and power-efficiency potential. Most of the proposed techniques can be implemented with a minimum complexity overhead.

Flores et al. (2010) show how messages can be efficiently managed in tiled CMP, from the point of view of both performance and energy, by combining both address compression with a heterogeneous interconnect. In particular, their proposal is based on applying an address compression scheme that dynamically compresses

the addresses within coherence messages allowing for a significant area slack. The arising area is exploited for wire latency improvement by using a heterogeneous interconnection network comprised of a small set of very-low-latency wires for critical short-messages in addition to baseline wires. Detailed simulations of a 16-core CMP show that their proposal obtains average improvements of 10% in execution time and 38% in the energy-delay2 product of the interconnect. Additionally, the sensitivity analysis shows that their proposal performs well when either OoO cores or caches with higher latencies are considered.

Rutgers et al. (2012) claim that experimental evidence is provided, showing that replacing a connection-oriented NoC by a connectionless one in a distributed shared memory system reduces the hardware costs and improves the performance. They observed that their FPGA could only support an 8-core system with a connection-oriented NoC. They exchanged the NoC with their tree-shaped, connectionless network and a ring, allowing a 32-core system in the same FPGA, mainly because of a reduced number of physical connections. Although the analytical worstcase performance slightly decreased, measurements show that the latency of latency-critical memory reads was reduced by 52% on average.

Eisley et al. (2006) propose an implementation of the cache coherence protocol within the network, embedding directories within each router node that manage and steer requests towards nearby data copies, enabling in-transit optimization of memory access delay. Their simulation results across a range of SPLASH-2 benchmarks demonstrate significant performance improvement and good system scalability, with up to 44.5% and 56% savings in average memory access latency for 16 and 64-node systems, respectively, when compared against the baseline cache coherence protocol. Detailed microarchitecture and implementation characterization affirms the low area and delay impact of in-network coherence.

Ros et al. (2010) present DiCo-CMP, a novel cache coherence protocol especially suited to future many-core tiled CMP architectures. In DiCo-CMP, the task of storing up-to-date sharing information and ensuring ordered accesses for every memory block is assigned to the cache that must provide the block on a miss. Therefore, DiCo-CMP reduces the miss latency compared to a directory protocol by

sending requests directly to the cache that provides the block in a cache miss. These latency reductions result in improvements in execution time of up to 6 percent, on average, over a directory protocol. In comparison with Token-CMP, their protocol only sends one request message for each cache miss, as such is able to reduce network traffic by 43 percent.

Pascual et al. (2008) present a coherence protocol aimed at dealing with transient failures that affect the interconnection network of a CMP, thus assuming that the network is no longer reliable. In particular, their proposal extends a token-based cache coherence protocol so that no data can be lost and no deadlock can occur due to any dropped message. Using the GEMS full-system simulator, they compare their proposal against a similar protocol without fault tolerance (TOKENCMP). They show that in the absence of failures, their proposal does not introduce overhead in terms of increased execution time over TOKENCMP. Additionally, their protocol can tolerate message loss rates much higher than those likely to be found in the real world, without increasing the execution time by more than 15 percent.

Deshpande et al. (2010) present cache coherence problem and solution on it, in multiprocessor architectures with various protocols of cache coherence. They compare and discuss benefits and limitations of protocols. Their results show that in general, the directory based protocol is more used for larger systems to enhance their performance; while snooping protocol is used for smaller systems.

Kim et al. (2009) propose a novel on-chip L2 cache organization for chip multiprocessors with private L2 caches. The proposed approach, called reusability-aware cache sharing (RACS), combines the advantages of both a private L2 cache and a shared L2 cache. Since a private L2 cache organization has a short access latency, the RACS scheme employs a private L2 cache organization. However, when a cache block in a private L2 cache is selected for eviction, RACS first evaluates its reusability. If the block is likely to be reused in the near future, it may be saved to a peer L2 cache which has space available. In this way, the RACS scheme effectively simulates the larger capacity of a shared L2 cache. Simulation results show that RACS reduced the number of off-chip memory accesses by 24% compared to a pure

private L2 cache organization on average for the SPLASH 2 multi-threaded benchmarks, and by 16% for multi-programmed benchmarks.

Chishti et al. (2005) change the latency-capacity tradeoff in two significant ways. They propose three novel ideas to exploit the changes: (1) Though placing copies close to requestors allows fast access for read-only sharing, the copies also reduce the effective on-chip capacity in CMPs. They propose controlled replication which avoids extra copies in some cases, and obtains the data from an already-existing on-chip copy at the cost of some extra latency. Because the copy is on-chip, the latency penalty is small and is offset easily by the reduction in off-chip misses due to reduced capacity pressure. (2) Inter-processor communication induced by read-write sharing is on-chip in CMPs and off-chip in DSMs. Because on-chip communication is faster than off-chip communication, there is a new opportunity to optimize read-write sharing in CMPs. Rather than incur slow access to read-write shared data through coherence misses as do DSMs, they propose in-situ communication which provides fast access to the data without making copies (via controlled replication) or incurring coherence misses. (3) DSMs migrate private data (in the case of no sharing) close to the requesting cores to allow fast access. While such migration is useful for CMPs as well, it may result in inefficient use of the on-chip capacity. They also propose capacity stealing which enables a core to migrate its less-frequently-accessed data to unused frames in neighboring caches with less capacity demand. Thus, capacity stealing dynamically customizes allocation of on-chip capacity. Because neighboring cores are on-chip in a CMP, accessing neighbors' caches is not expensive as is the case in DSMs.

According to Chen et al. (2013), DSM preferably uses virtual addressing in order to hide the physical locations of the memories in NoC based multi-core platforms. However, this incurs performance penalty due to the Virtual-to-Physical (V2P) address translation overhead for all memory accesses. Based on the data property which can be either private or shared, this paper proposes a hybrid DSM which partitions a local memory into a private and a shared part. The private part is accessed directly using physical addressing and the shared part using virtual addressing. In particular, the partitioning boundary can be configured statically at

design time and dynamically at runtime. The dynamic configuration further removes the V2P address translation overhead for those data with changeable property when they become private at runtime. In the experiments with three applications (matrix multiplication, 2D FFT, and H.264/AVC encoding), compared with the conventional DSM, their techniques show performance improvement up to 37.89%.

With the high cost of remote memory access, DSM systems achieve higher performance by reordering (giving priority) or overlapping (combining) the servicing of multiple memory or communication operations from a processor (Culler et al., 1999). In order to achieving low latencies, messages can be combined within the interconnect (Protic et al., 1996). Priorities may be used to differentiate memory transactions, decreasing the response time to higher priority messages, as block invalidation (Chaves et al., 2011).

Priority technique has been extensively studied for NoC architectures. Beigne et al. (2005) propose a new Asynchronous NOC architecture, which provides low latency transfers. For improving efficiency and to guarantee low latency, two virtual channels are implemented in each node. The first one is dedicated to real-time, low latency packets, and the other one for best-effort traffic. The first channel VC0 has the highest priority and can suspend the path of the second channel VC1. A given packet cannot be suspended by a packet of the same or lower priority, it may only be suspended when a packet with a higher priority requests the same network link. In that case, the suspended packet is stalled and stored in previous nodes.

Li and Mutka (1996) developed a range of flow control schemes for real-time messages concerning priority mapping strategies, priority adjustment methods, and arbitration functions. In (Song et al., 1999), a flit-level preemption flow control is developed to resolve the priority inversion problem, i.e., a higher priority message is blocked by a lower priority message occupying shared resources. Lu et al. (2005) assume a real-time (RT) message delivery model without a complicated router architecture and without a special service. All messages are globally prioritized (priority ties are resolved arbitrarily). This model arbitrates shared lanes and link bandwidth by priority. The priority, which may be assigned according to rate, deadline or laxity takes a small number of flits.

In the work made by Carara et al. (2009), the implemented QoS support relies on fixed priorities mechanism. Two prioritized best effort (BE) traffic classes are distinguished inside the NoC: (i) high priority packets and (ii) low priority packets. One physical channel (channel 0) is reserved to transmit exclusively high priority packets, whereas channel 1 may transmit both packet classes. Sharing one of the two physical channels provides higher support to high priority traffics, since two high priority flows can share common paths. The priorities mechanism provides a soft guaranteed service (latency and bandwidth) to high priority traffics through a virtual resource reservation.

Manolache et al. (2005) approach can automatically assign message priorities according to the message criticality. Cota et al. (2003) use a packet switched network model on 2d torus architecture using message passing. They states that the cores with larger number of packets and larger size of packets have priority to use shorter paths to reduce test time.

For speeding up the recovery process, Grecu et al. (2007) consider a second retransmission-based recovery scheme, where ARQ and retransmitted messages have higher priority than the regular messages. The advantage of priority based recovery is more apparent at higher injection rates, where the ARQ and retransmitted messages are able to travel faster than the regular messages waiting in queue buffers for available resources.

Message priority technique has also been extensively studied for switch-based architectures. Bolotin et al. (2007) state that their basic low cost mechanism is based on priority support embedded in the NoC, which differentiates between short control signals and long data messages to achieve a major reduction in cache access delay. They attribute different priorities to packets transmitted by the cache coherence protocol. Operations such as read and exclusivity request (short data packets) are transmitted using high priority. Long packets, such as packets containing data to be written in the memory or a block just read from the memory are transmitted using low priority. They show that low cost Priority-based NoC is extremely useful for increasing performance of almost any other CMP transaction (i.e. uncached and cache-coherenet R/W, search in DNUCA, isolating low priority traffic,

synchronization and mutual exclusion support). Priority-based NoC along with the discussed NoC interfaces are evaluated in detail using cycleaccurate CMP-NoC simulations across several SPLASH-2 benchmarks and static web content serving benchmarks showing substantial L2 cache access delay reduction and overall program speedup.

Barroso et al. (2000) describes the Piranha system, a research prototype being developed at Compaq that aggressively exploits chip multiprocessing by integrating eight simple Alpha processor cores along with a two-level cache hierarchy onto a single chip. In addition to exploiting chip multiprocessing, the Piranha prototype incorporates several other unique design choices including a shared second-level cache with no inclusion, a highly optimized cache coherence protocol, and a novel I/O architecture. They propose optimizations to an invalidation-based directory protocol which exploits priority to differentiate packets. A low priority lane is used by request sent to a home node, while the high priority lane is used by forwarded requests and all replies.

Chaves and Moraes (2011) goal of their work is to optimize a directory-based cache coherence protocol exploiting specific NoC services, as multicast and priorities. They use multicast to reduce traffic generated by the cache coherence protocol, and priority to increase the system performance. To demonstrate their proposal, an MPSoC described at the RTL level is used, enabling accurate performance and energy evaluation. Results show a reduction of 17% in the number of clock cycles and a reduction up to 86% (average reduction: 39%) in energy consumption for some memory transactions.

Priority is also used in (Karlsson and Brorsson, 2003) for software DSM systems. They claim that many applications exhibit a good performance on DSM systems but it has been shown that performance critical messages can be delayed behind less important messages because ofthe enqueuing behavior in the communication libraries used in current systems. They present a new portable communication library that supports priorities to remedy this situation. They describe an implementation of the communication library and a quantitative model that is used to estimate the performance impact of priorities for a typical situation. Using

the model, they show that the use of high-priority communication reduces the latency of performance critical messages substantially over a wide range of network design parameters. The latency is reduced with up to 10-25% for each delaying low priority message in the queue ahead.

Message combining technique has been used for distributed memory system in (Shafer and Ghose, 1998), for one dimensional arrays in (Souravlasa and Roumeliotisb, 2008), for multistage interconnection networks in (Hanawa et al., 1996) and for a ring-based network and 2D-mesh in (Dvorak, 2008; Dvorak and Joras, 2010; Ma et al., 2012).

Shafer and Ghose (1998) reduce the execution time of a parallel program by merging messages (also called message combining, or message consolidation) after the program has already been partitioned and scheduled onto the processing elements. The data from two (or more) messages are combined and sent in a single communication by locating sections of the system where a merge will affect the overall execution time, and determining before the merge takes place whether it will positively or negatively affect the system.

Souravlasa and Roumeliotisb (1998) present a message combining approach for scheduling runtime array redistribution of one-dimensional arrays. The important contribution of the proposed scheme is that it eliminates the need for local data reorganization, the blocks destined for each processor are combined in a series of messages exchanged between neighbouring nodes, so that the receiving processors do not need to reorganize the incoming data blocks before storing them to memory locations. Local data reorganization is of great importance, especially in networks where there is no direct communication between all nodes (like tori, meshes, and trees). Thus, a block must travel through a number of relays before reaching the target processor. This requires a higher number of messages generated, therefore, a higher number of data permutations within the memory of each target processor should be made to assure correct data order. The strategy is based on a relation between groups of communicating processor pairs called superclasses. They have implemented two sets of simulations on a multiprocessor system, where the number of nodes varied from 8 to 128 to test its performance. These experiments lead them

to conclude that the message combining approach performs better when they redistribute relatively small sized messages on a large number of processors.

Hanawa et al. (1996) states that Simple Serial Synchronized (SSS) Multistage Interconnection Network (MIN) is a novel MIN architecture for connecting processors and memory modules in multiprocessors. Synchronized bit-serial communication simplifies the structure/control, and permits the bit-serial message combining mechanism. From the theoretical analysis and empirical results, it appears that the influence of the hot spot contention is not catastrophic in the small size SSS-MIN (16×16) when some parallel programs from SPLASH benchmark run. In this situation, the effect of the message combining is small. However, in a large system (256 processors), the pass through ratio improves by 11% when a parallel program with relatively large access ratio (r=0.4) runs. Since the additional hardware for the message combining is only 20% in the SSS-MIN, the message combining is useful in large systems

Dvorak (2008) investigates an impact of collective communication overhead that may be critical for performance of paralel applications. Two potential topologies of networks on chip are investigated, a ring-based network and 2D-mesh, due to their easy manufacturability on a chip. The wormhole switching, full duplex links and 1-port non-combining as well as combining nodes are considered. As a future work of this study, Dvorak and Joras (2010) investigate an impact of direct and combining collective communications models that may be critical for performance of parallel applications. Analysis provided for any given start-up time and message transfer time reveals the fastest collective communication mode in relation to the number of processing elements in 2D meshes and fat tree networks on a chip.

Ma et al. (2012) explore support for reduction or many-to-one communication operations. As a case study, they focus on acknowledgement messages (ACK) that must be collected in a directory protocol before a cache line may be upgraded to or installed in the modified state. They make two primary contributions: an efficient framework to support the reduction of ACK packets and a novel Balanced, Adaptive Multicast (BAM) routing algorithm. The proposed message combination framework complements several multicast algorithms. By combining ACK packets during

transmission, the framework not only reduces packet latency by 14.1% for low-to-medium network loads, but also improves the network saturation throughput by 9.6% with little overhead.

Message combining for DSM systems have been studied commonly for switch-based architectures. Dai and Panda (1996) present an approach to reduce cache invalidation overheads. They introduce a set of multidestination-based reservation and gather worms to distribute invalidation requests and collect acknowledgments. Six different grouping schemes are investigated in generating such multidestination worms in order to implement fully-mapped directory-based cache coherence on networks using deterministic (e-cube) or adaptive (turn-model) wormhole routing. These schemes are evaluated for different applications and 2D mesh configurations through execution-driven simulation. The interplay among overall execution time, the number of invalidation messages, total network traffic, routing adaptivity is studied. The results indicate that up to 15% reduction in overall execution time can be achieved by using multidestination messages. It is observed that turn-model routing with a density-dependent column grouping leads to maximum benefits. The results demonstrate that current and future DSM systems can take advantage of these schemes to deliver better performance.

Message combining under broadcast-based architecture has been studied in (Katsinis and Hecht, 2000) for message passing programming model. Katsinis and Hecht (2000) assumed that each message (B) combined with an already existing message (A) causes the size of message A to increase by half the size of message B and as a consequence the latency is reduced by approximately 25%.

Message combining has been studied for broadcast-based DSM model in (Katsinis and Hecht, 2004). However, simulations have been run only for hot-spot and uniform traffic pattern and the designed architecture has a 1D interconnection network. Besides, the decreasing rate of combined message size is constant for every combining operation and is 10% for different destination addresses and 15% for the same destination addresses.

3. EVALUATION METHODOLOGY

The evaluation of high performance parallel systems is a delicate issue due to the difficulty of generating workloads that represent those that will run on actual systems (Navaridas and Miguel-Alonso, 2009). Most performance studies on interconnection networks in literature are carried out using synthetic traffic workloads, which stands for running a simulator for a large number of cycles (simulated time) to get performance results of the network in steady state, for many practical reasons (Miguel-Alonso et al., 2008). Since the synthetic traffic from independent sources allows the programmers to rapidly assess raw performance, it provides a good approach to evaluate an interconnection network system (Navaridas and Miguel-Alonso, 2009). Although this is not a realistic approach, it is considered that the obtained results are the indicators of the minimum level of the system performance that the network could provide under real traffic (Miguel-Alonso et al., 2008). Synthetic loads capture the representative aspects of application-driven workloads and are easier to design and manipulate. Furthermore, the programmers do not use traces or real traffic as it becomes impracticable with large network sizes (Camara et al., 2007).

3.1. Synthetic Traffic Workload

Synthetic traffic workload is characterized by the temporal distribution, the spatial traffic pattern and the size of the messages (Miguel-Alanso et al., 2008). The spatial traffic pattern, which is explained in Section 3.1.1., determines the distribution of the destinations for each source node. The temporal distribution designates when a message is generated, while the message size distribution specifies the size of each message, as is explained in Section 3.2.2.

3.1.1. The Temporal Distribution

In literature, a range of options can be found for the temporal distribution of the message generation. These options can be classified in two groups that are called independent and nonindependent traffic sources. In independent traffic sources, nodes are " programmed" to generate messages using some probability distribution including Poisson, Bernoulli (that are smooth over large time intervals) and on-off models that better characterize the self similarity of traffic in some applications (Shin and Pinkston, 2003). Each node progresses independently of the others. Most simulation-based studies of interconnection networks follow this approach. In this thesis, exponential distribution is used for the message generation. The messages keep walking around the network until the simulation ends.

3.1.2. The Spatial Traffic Pattern

In this thesis, the spatial traffic pattern is evaluated into two categories: the distribution of the home nodes and the distribution of the owner and the sharer nodes.

Uniform distribution (UN) is used for selecting the owner and the sharer nodes. In uniform traffic, the source and the destination nodes are chosen via a uniform random process. The load is balanced because the source and the destination coordinates are uniformly distributed, although the random process that generates the coordinates may cause transient hotspots (Gratz and Keckler, 2010). Uniform traffic is useful for detecting network bottlenecks as it reflects the topological properties of the network under study (Camara et al., 2007).

For the distribution of the home nodes, bit reverse (BR) and perfect shuffle (PS), which are a collection of well-known bit permutations, are used. Permutation patterns, in which clusters of nodes communicate among themselves for extended intervals, are common in multiprocessor applications (Gratz et al., 2008). Bit permutations traffic pattern includes a uniform traffic pattern, in which each source sends an equal amount of traffic to each destination. Bit reverse and perfect shuffle are those traffic patterns in which the destination address in bit is reversed and

shuffled of the source bit address (Paliwal et al., 2009). For instance, if the source bit address is (s3, s2, s1, s0), the destination for a BS traffic pattern is (s0, s1, s2, s3) while for a PS traffic pattern is (s2, s1, s0, s3). In contrast with uniform traffic, bit permutations are sensitive to the order in which nodes are labeled as message destinations are selected depending on their label (Camara et al., 2007).

Uniform traffic, and hot region traffic (HR (Blumrich et al., 2003)) are also included in the simulations for the distribution of the home nodes. Blumrich et al. (2003) create hot region traffic by selecting the destinations of 25% of the messages randomly within a small " hot" contiguous sub-mesh region consisting of 12.5% of the machine. The destinations of the remaining 75% of the messages are chosen uniformly over the entire machine.

Each of these traffic patterns used in this thesis represents a worst-case scenario: uniform has no locality, permutations make an uneven use of resources, and hot-spot models nodes that receive a higher proportion of the traffic (Izu et al., 2005a). These traffic patterns provide insight into the relative strengths and weaknesses of the different combining and priority techniques proposed in this thesis.

3.2. Simulation Framework

3.2.1. Simulation Tools and System Architecture

OPNET Modeler (OPNET Technologies Inc., 2013) is an environment for network modeling and simulation, which can also be used for designing and studying interconnection networks and protocols. It is based on a series of hierarchical editors, project editor, node editor and process editor, which directly parallel the structure of interconnection networks and protocols. The node editor captures the architecture of a system by depicting the flow of the messages between the functional elements, called modules. Each module can generate, send and receive messages from the other modules to perform its function within the node. Modules typically represent physical resources such as buffers, ports, queues and buses. Modules are assigned

process models, developed in the process editor, to achieve any required behavior. The process editor uses a finite state machine approach to support specification of protocols, algorithms and queuing policies. States (the condition of a module) and transitions (a change of state) graphically define the progression of a process in response to events. States have " enter executives" (code that is executed when the module moves into a state) and " exit executives" (code that is executed when the module leaves a state), and there is " transition executive" (code that is executed in response to a specific event). There are two kinds of states: An unforced (red) state is the one that returns control of the simulation to the simulation kernel after executing its enter executives. A forced (green) state is one that does not return control, but instead immediately executes the exit executives and transitions to another state.

Each SOME-Bus node can be represented by a set of queues through which messages of different types flow. Each node contains four major components: The processor handles all activities related to the scheduling of the threads. The arrival of data and ownership acknowledge messages causes threads to become ready for execution and therefore, affects the processor operation. The cache controller fills requests for data from the threads. The directory controller maintains the directory information for the portion of main memory that is located at its node and receives and processes data and ownership requests from the processor. The channel controller receives messages from the processor, cache or directory controllers and delivers them to the destination node. If the source and destination nodes of the message are different, the message is considered to be remote and is placed on the output queue associated with the output channel of the source node. When the channel becomes available, the message is transmitted and arrives at the input queue at the destination node. Messages that are broadcast or multicast arrive simultaneously at the destination input queues.

A 64 node 2D DSM SOME-Bus system is simulated by using OPNET Modeler. Each simulated node has a server, which processes the directory, the cache and the processor messages, two output channels and two input queues, one for the horizontal and the other for the vertical dimension.

The underlying process model that controls queue modules' behaviour is OPNET' s built-in acb_fifo model, which can be seen in Figure 3.1. OPNET' s built-in acb_fifo model has its own server and can concentrate multiple incoming packets streams into its single internal queuing resource. It also supports the First-in-First-out service ordering discipline and a way to control service times. The " init" state is used to initialize the process and setting the appropriate variables. If a packet arrives when the process is in " init" state, the process transitions to the " arrival" state, else it transitions to the " idle" state where it waits for packet arrival. The " arrival" state is used for receiving packets and starting service. In the " arrival" state, if the server is not busy then the process moves into the " svc_start" state, which in turn transitions to the " idle" state, where it waits either for packet arrival or service completion. While in the " idle" state, if the processing of a packet is completed, the process moves into the " svc_compl" state. While in the " svc_compl" state, if the queue is not empty, the process moves into the " svc_start" state.

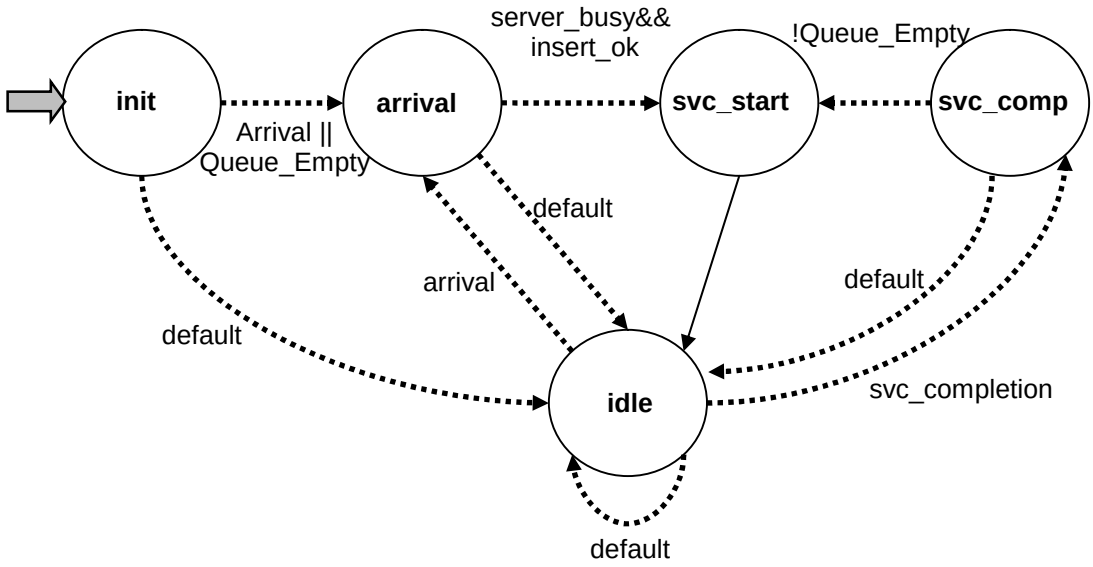


Figure 3.1. State diagram of the acb_fifo model in OPNET Modeler

A 2x2 node DSM model for the 2D SOME-Bus architecture can be seen in Figure 3.2. S represents the server. O_h represents the horizontal output channel and O_v represents the vertical output channel, while I_h represents the horizontal input queue and I_v represents the vertical input queue. Bus_h_0/Bus_h_1 represents horizontal

SOME-Bus of number zero and one, respectively while Bus_v_0/Bus_v_1 represents vertical SOME-Bus of number zero and one, respectively.

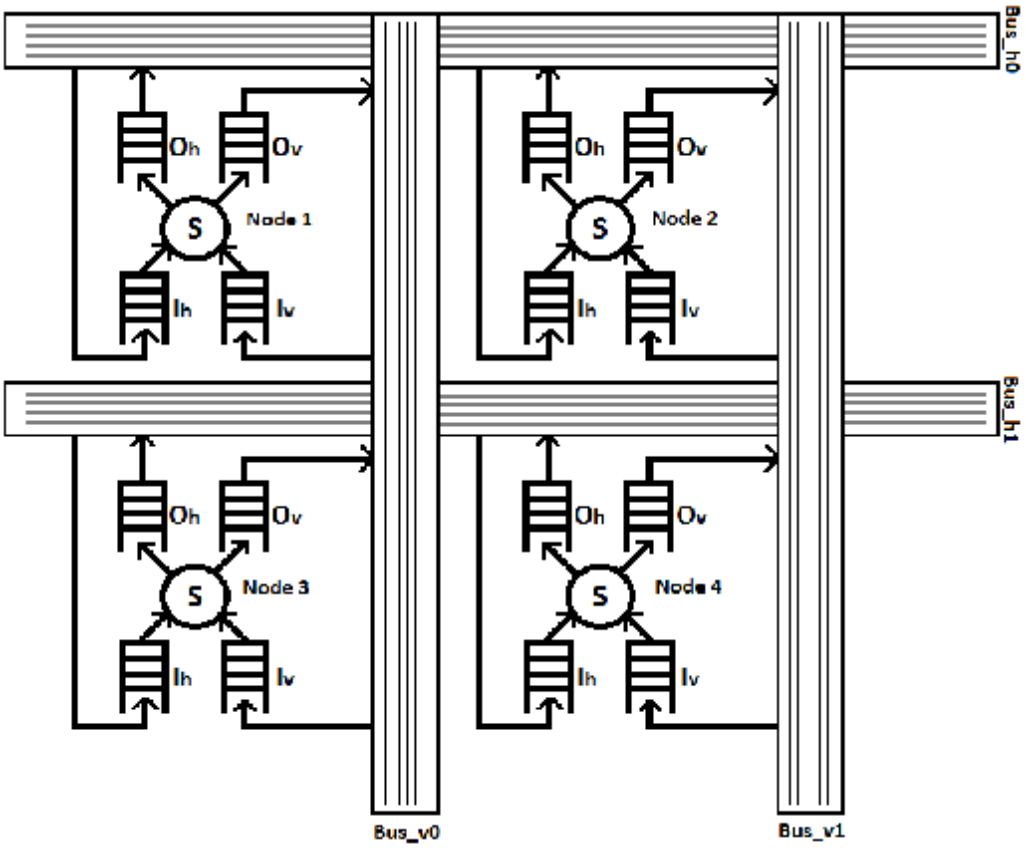


Figure 3.2. A 2x2 DSM model for the 2D SOME-Bus architecture

If a source node sends a message to a destination node, which is positioned in the same dimension, the system operation is equivalent to the operation of the 1D SOME-Bus. Otherwise, rebroadcast operation over an intermediate node is needed. In the simulation experiments described in this thesis, if the source and the destination nodes are not at the same dimension, messages are always routed on the horizontal bus first and then on the vertical bus to avoid deadlock. For instance, if Node 1 sends a message to Node 2, initially the message is inserted into O_{h1}, which is the horizontal output channel of Node 1, since the destination node is in the same horizontal dimension. The message is then transmitted on Bus_h_0 and placed into the horizontal input queue of Node 2 (I_{h2}). Similarly, if Node 1 sends a message to Node 3, which is in the same vertical dimension, $O_{v1,}$ which is the vertical output channel

of Node 1, is used instead of O_{h1}. The message is transmitted on Bus_v_0 and placed into the vertical input queue of Node 3 (I_{v3}). However, when Node 1 sends a message to Node 4, the message is firstly routed on the horizontal bus, i.e. it follows the path of O_{h1} and Bus_h_0, respectively. The node, which is in the same dimension with the source and the destination node is the intermediate node. In this example, Node 2 is in the same horizontal dimension with Node 1, while it is in the same vertical dimension with Node 4. Thus, the message is enqueued in I_{h2}. Node 2 displaces the message from its horizontal input queue to its vertical output queue (O_{v2}) in order to get the message off to its destination. Then, the message is transmitted on Bus_v_1 and eventually arrives to its destination, the vertical input queue of Node 4 (I_{v4}).

The memory consistency model used in this thesis is a type of relaxed consistency model that allows reads or writes to bypass previous reads or writes if and only if they transact to the different memory blocks. The directory based cache coherence protocol used in the system is reply forwarding protocol, which shows better performance (lower average network response time and higher average processor utilization) for most cases than the equivalent protocols (Abasikeles and Akay, 2010) and the state of a cache block (i.e. cache line) is determined according to the Modified Exclusive Shared Invalid (MESI) protocol. Each cache line is either Modified, i.e. the local cache has the only copy of the cached data in the system and it is dirty; Exclusive, i.e. only one cache has a copy of the block and it has not been modified; Shared, i.e. the local cache contains a valid, read-only copy of the data, and furthermore other caches may also have a read-only copy; or Invalid, i.e. the local cache does not have a valid copy of the data. Directory entries can be in the state Unowned, i.e. no cached copies in the system; Shared, i.e. zero or more read-only cached copies or Modified, i.e. one read-write cached copy in the system and the block may be in either dirty or (clean) exclusive state in the cache (Eisley et al., 2006). Each directory entry is associated with a bit vector (the copy set) that identifies the processors with a copy of the data block corresponding to that entry.

The reply forwarding directory protocol diagram of a read/write operation to a Modified cache state block is given in Figure 3.3., while that of a write operation to a Shared cache state block is shown in Figure 3.4. (Abasikeles and Akay, 2010). In

the Figures, " L" represents the local node which requests the block, " H" represents the home node, which has the requested block in its local memory, " R" represents the remote owner node, which has the block in modified state in its cache. " R_1" and " R_2" are remote sharers of the block.

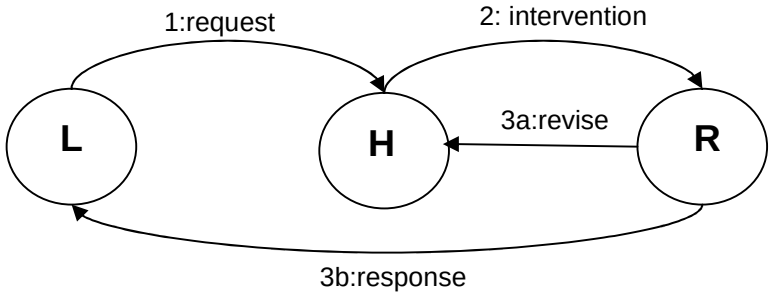


Figure 3.3. Reply forwarding directory protocol diagram of a read/write operation to a Modified cache state block

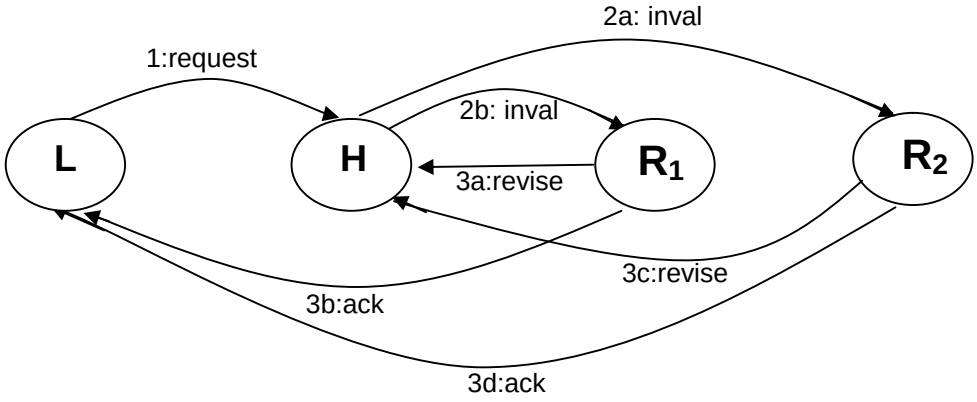


Figure 3.4. Reply forwarding directory protocol diagram of a write operation to a Shared cache state block

Both for a read and a write request to a block in Modified cache state, initially the local node requests the block from the home node. The home node transmits the request to the owner node and the owner node directly communicates with the local node (i.e. sends reply to the local node), while it sends a revise message to the home node. If the local node generates a write request to a shared block, the home node transmits the request to the entire sharer nodes in the system and the sharers directly communicate with the local node, while they send a revise message to the home node. For the remainder request and block pairs, the home node provides the block from its local memory.

3.2.2. Message Types

The simulated system includes five types of messages, which are shown in Table 3.1. Each message includes a source address of two bytes, a destination address of two bytes and other header information including error bits, message types etc. of two bytes. The first type of messages (*Type1*) does not carry data. The second type of messages (*Type2*) carries data of 18 bytes. The third type of messages (*Type3*) carries data with a sharer list, while the fourth type of the messages (*Type4*) carries a sharer list without data. The last type of messages (*Type5*) carries only the requestor address. Since the address of a node takes a space of two bytes in a message and the sharer count in the system is four, the entire size of the field reserved for the sharer addresses is eight bytes. While the entire size of the field reserved for the requestor address is solely two bytes, as is seen in Table 3.1.

Table 3.1. Message types in the simulation

Message part/byte	*Type1*	*Type2*	*Type3*	*Type4*	*Type5*
Source address	2	2	2	2	2
Destination address	2	2	2	2	2
Other header info	2	2	2	2	2
Data	-	18	18	-	-
Sharers	-	-	8	8	-
Requestor	-	-	-	-	2
Total bytes	**6**	**24**	**32**	**14**	**8**

The channel service time of a message that does not carry data (*Type1*) is a quarter of that of a message that carries data (*Type2*). This ratio is consistent with the message sizes used in (Ye et al., 2004) and (Lee et al., 2000). In (Ye et al., 2004), short messages are the messages with no payload (for instance, memory access request messages), while long messages are the messages with payloads (for instance, data fetch messages). The size of a short message is a quarter of that of a long message. Lee et al. simulated a 64 node DSM system, in which they used 32 bytes for a cache line (payload in message) and 12 bytes for an invalidation message for conventional unicast, which corresponds to the *Type1* message in our simulation. Therefore, the size of a message with data used in (Lee et al., 2000) is 44 bytes, which is approximately four times more than a message without data.

For a message carrying a data with a sharer list (*Type3*), the channel service time is five times more than that of a message that does not carry a data (*Type1*). *Type3* can be seen as a combination of the payload and the header part of *Type4*. The header size of *Type4* is two times more than the header size of *Type1*. The payload is three times more than the size of *Type1*, because the size of *Type2* is four times more than the size of *Type1*. Therefore, the size of *Type3* is five times more than the size of *Type1*.

The messages created in the system according to the reply forwarding protocol can be seen in Table 3.2. The table includes the names, the definitions, the source and destination node elements and the types of the messages.

The processor at each node is assumed to be executing a program with multiple threads. After a thread finishes running, a cache miss occurs, which causes a request message to be sent to the home directory node of the requested block. This message is either a read request or a write request. A read request is defined as a data request and symbolized with *DATA_REQ*. A *DATA_REQ* is sent from the requesting processor node to the home directory node. *OWN_REQ* and *UPGRADE_OWN_REQ* are write request messages, which are described as owner requests. If a node has the requested block in its local cache, but does not have write permission on that block, it sends *UPGRADE_OWN_REQ* message to the home directory, whereas it forwards *OWN_REQ* if it has neither the block nor permission. All request messages regardless of being read or write have the message type of *Type1*.

According to the reply forwarding protocol, for a data request to a block in a shared or an unowned directory state, the block is supplied from the home directory node memory. The home directory node sends a data acknowledgment message (*DATA_ACK*) with the data to the requesting node. Since this message includes the data, is has type of *Type2*. However, if the block is in modified directory state, after determining the owner cache node among the remote nodes with uniform distribution, a downgrade request message (*DOWN_GR_REQ*) is sent to that owner cache by the home directory. This message also includes the address of the requesting processor node. Therefore, it has a message type of *Type5*. When a cache gets a message of *DOWN_GR_REQ*, it sends *DATA_ACK* message to the requesting

node and downgrade write back acknowledge (*DOWN_GR_WB_ACK)* message to the home directory node. Since the purpose of the *DOWN_GR_WB_ACK* message is to inform the home directory about the operation only, it has the type of *Type1*.

Table 3.2. The directory protocol messages

Name	Definition	From	To	Type
DATA_REQ	Data request to a block	Processor	Directory	Type1
DOWN_GR_REQ	Data request to modified cache	Directory	Cache	Type5
DOWN_GR_WB_ACK	Response of DOWN_GR_REQ	Cache	Directory	Type1
DATA_ACK	Response of DATA_REQ	Directory/ Cache	Processor	Type2
OWN_REQ	Owner request to a block	Processor	Directory	Type1
OWN_ACK	Response of OWN_REQ	Directory/ Cache	Processor	Type2/ Type1
INV_REQ	Invalidation request to sharers	Directory	Cache	Type5
INV_WB_ACK	Response of INV_WB_REQ	Cache	Directory	Type1
INV_ACK	Response of INV_REQ	Cache	Processor	Type1
BLOCK	Data block	Directory	Cache	Type1
BLOCK_ACK	Response of BLOCK	Cache	Directory	Type1/ Type2
UPGRADE_OWN_REQ	Upgrade owner request to a block	Processor	Directory	Type1
INV_WB_REQ	Invalidation request to modified cache	Directory	Cache	Type5
RESPONSE_HOME	Response to home directory	Directory	Directory	Type1
SHARER_LIST	The list of sharers for a block	Directory	Directory	Type3/ Type4

For an ownership request (*OWN_REQ*) to a block in an unowned directory state, the home directory sends an ownership acknowledge message (*OWN_ACK*) with the requested block to the requesting node. Since this message includes data, is has type of *Type2*. If the block is in modified directory state, after determining the owner cache node among the remote nodes with uniform distribution, an invalidation write back request message (*INV_WB_REQ*) is sent to that owner cache by the home directory. This message also includes the address of the requesting processor node. Therefore, it has message type of *Type5*. When a cache gets a message of *INV_WB_REQ*, it sends *OWN_ACK* message to the requesting node and invalidation write back acknowledge (*INV_WB_ACK)* message to the home directory node. Since the purpose of the *INV_WB_ACK* message is to inform the home directory about the operation, it has type of *Type1*. It can be easily observed that the network transactions of an ownership request to a block in modified directory state are the same with that of a data request. The only difference is the type of the messages.

However, if the block is in shared state, the path of the messages is quite different than the remainder states. Since the invalidation acknowledgment messages are forwarded to the requestor node by the remote caches, the home directory initially sends the addresses of sharer caches to the requestor node' s directory, which is demonstrated with *SHARER_LIST* message in Table 3.2. and then the home directory node sends the invalidation request messages (*INV_REQ*) to the entire sharer cache nodes, which are determined among the whole nodes with uniform distribution. If the requested message is an ownership request (*OWN_REQ*), the *SHARER_LIST* message also includes the requested block because the requestor does not have the data, as explained above. Besides, the type of this message is *Type3*. If the requested message is an upgrade ownership request (*UPGRADE_OWN_REQ*), sending the data block is not required since the requestor node already has it in its local cache and the type of this message is *Type4*. When a requesting directory gets a *SHARER_LIST* message, it sends an informative response message that has the type of *Type1* to the home directory. When the sharer' s cache receives an invalidation request message, it sends the invalidation acknowledge message to the requesting directory (*INV_ACK*). When all invalidation acknowledgement messages are received by the requestor directory, it sends an owner acknowledge message to the local processor and the data to the local cache if needed. Since the sharer cache nodes send their *INV_ACK* message directly to the requestor, they need to know the address of the requesting node that results in the *INV_REQ* message to have the type of *Type5*.

Another type of the message in the system is generated when a cache gets a block but has no empty space to put it in, as illustrated with the *BLOCK* message in Table 3.2. At this time, the cache has to remove a random block, and notify the home directory of the block about this operation. This information message is called *BLOCK_ACK* and if the removed block is in modified directory state, the cache also have to send the data block to the home directory that causes the *BLOCK_ACK* message to have a type of *Type2* (otherwise it is *Type1*).

3.2.3. Simulation Parameters

The traffic generation method used in this thesis is extensively described in Section 3.1. For making the experiments reproducible, the rest of the parameters used in the simulations must be described. Other major parameters of the simulation, which can be seen in Table 3.3., are the distribution of the thread run time (R), number of threads in each node, the fraction of write messages, the number of invalidation messages sent with every request for ownership message, the mean channel service time (T) for different types of messages, probability of a cache being full and probability of a block being in various states.

Table 3.3. The parameters of the simulated 2D SOME-Bus system

Parameter	Value
Thread number in each node	10
Mean thread run time	exponentially distributed with a mean of 100
Probability of write (ownership) request	0.2 , 0.4 , 0.6
Probability of upgrade ownership request	0.2
Prob. of a block being in modified state	0.2 , 0.4 , 0.6
Mean channel service time – Type1	varies between 1.25 and 25
Mean channel service time – Type2	varies between 5 and 100
Mean channel service time – Type3	varies between 6.25 and 125
Mean channel service time – Type4	varies between 2.5 and 50
Mean channel service time – Type5	varies between 1.875 and 37.5
Home node selection	UN, HR, BR, PS
Owner node selection	UN
Probability of a cache being full	0.15
Sharer count	4

The processor at each node is assumed to be executing a program with ten threads. In a real application execution, a large fraction of time will be spent by the processors doing calculations. At certain instants, these calculations need data in external memory and a remote memory access is performed. An important parameter in this respect is the computation to communication ratio, which tells us whether the execution of a certain application is dominated by useful computation, versus waiting for remote memory accesses (Heirman et al., 2007). In this simulation, the time between subsequent requests from the same node (called thread run time) has an exponential distribution with a mean of 100. In the context of communication networks this time is also referred to as the think time, during which the processor or

user ‘ thinks’ about what request he will make next. The requesting message is an ownership request message with a probability of P(W), or a data request message with a probability of 1 – P(W). P(W) has the values 0.2, 0.4 and 0.6 while the probability that a block is found in modified state, P(M), also takes the values 0.2, 0.4 and 0.6. These numbers are consistent with commonly observed memory reference patterns of real parallel applications and benchmarks. For instance, Acacio et al. (2002b) have experimented with five different parallel applications (i.e. EM3D, FFT, MP3D, Ocean and Unstructured) and they observed that write cycles constitute 25% to 68% of all memory cycles. In (Hu and John, 2006), the write miss percentage of the SPEC CPU INT 2000 benchmarks was reported to change from 13% to 52.74%. It was also reported in the same study that 20% to 55% of overall misses were to a modified cache block. The number of invalidation messages sent with every ownership request message is four.

The ratio T/R varies between 0.05 and 1. This range of the ratio is sufficient to capture the system behavior under most common configurations and cache behavior. Specifically, let m be the miss rate and F the number of instructions per second performed by the processor at each node. In addition, let S be the mean message size in bytes and C the channel bandwidth (in bytes per second). Then, the ratio of the mean thread run time to the mean message transfer time T/R = mSF/C. In current high performance architectures, the ratio of F/C is in the range of 0.5 – 1. For example, in Cray XT3 (Alam et al., 2008), $F = 4.8 \times 109$, and the links have a peak bandwidth of 7.6 GB/s. With small cache blocks and miss rate in the neighborhood of 10% or less (due to the fact that programmers are going to target and distribute their applications for maximum locality, thus most accesses on well behaved applications are going to fall in cache), the resulting ratio T/R is in the range of 0.05 to 1.

There are several applications for which upgrade misses account for an important fraction of the cache misses (Acacio et al., 2002b). Upgrade misses are caused by a store instruction that finds a read-only copy of the data in the cache. For this kind of misses, the cache already has the valid data and only needs exclusive ownership. The directory must invalidate all the copies of the data but the one held

by the requesting processor. The effect of upgrade misses is taken into account in the simulation by setting the probability of having an upgrade ownership request message to 0.2, which is consistent with the numbers given in (Acacio et al., 2002b).

The value of the last parameter, the probability of a cache being full, is 15%. When a cache gets a block but has no empty space to put it in (full), it removes a random block and notifies the home directory of the block about this operation.

3.3. The Proposed Algorithms

3.3.1. The Message Combining Algorithm

Owing to the broadcast capabilities of the SOME-Bus architecture, we have combined two, three and four messages waiting in both the horizontal and the vertical output channel queues of each node in the system, regardless of the message destinations and create a new message to be transmitted. When a message arrives at the output channel queue and finds that the queue is nonempty, it is combined with the message or the messages at the tail of the queue.

The size of the resulting message differs with the sizes of the combining messages. Two scenarios are considered for each combining type. The first one is the worst case scenario that the destination addresses, the requestor addresses and the entire sharer addresses are different in the combining messages. The second scenario is the best case in the sense that the destination addresses, the requestor addresses and the sharers are the same for all combining messages. The estimation of the size and the decreasing rate of the resulting message for the two messages combining under the first and the second scenarios are given in Table 3.4. and Table 3.5., respectively. The headers of the tables are comprised of the parts of the combining messages (Source address, Destination address, Other header info, Data, Sharers, Requestor), total size of the resulting message after the combining operation (Total Combined), total size of the messages without combining operation (Total Separate) and the decreasing rate in the size of the resulting message according to the total size of the messages without combining (Decreasing Rate), while the rows of the tables

include the types of the combining messages, which are given in Table 3.1. Average of the decreasing rates of the first and the second scenario is used in the simulations, as is seen in Table 3.6. for two messages combining.

Table 3.4. The estimation of the size and the decreasing rate of the resulting message for the two messages combining under the first scenario

	Source address	Destination address	Other header info	Data	Sharers	Requestor	Total Combined	Total Separate	Decreasing Rate (%)
Type0-Type0	2	2 + 2	2	-	-	-	8	12	33.3
Type0-Type1	2	2 + 2	2	18	-	-	26	30	13.3
Type0-Type2	2	2 + 2	2	18	8	-	34	38	10.5
Type0-Type3	2	2 + 2	2	-	8	-	16	20	20
Type0-Type4	2	2 + 2	2	-	-	2	10	14	28.5
Type1-Type1	2	2 + 2	2	18+18	-	-	44	48	8.3
Type1-Type2	2	2 + 2	2	18+18	8	-	52	56	7.1
Type1-Type3	2	2 + 2	2	18	8	-	34	38	10.5
Type1-Type4	2	2 + 2	2	18	-	2	28	32	12.5
Type2-Type2	2	2 + 2	2	18+18	8+8	-	60	64	6.25
Type2-Type3	2	2 + 2	2	18	8+8	-	42	46	8.7
Type2-Type4	2	2 + 2	2	18	8	2	36	40	10
Type3-Type3	2	2 + 2	2	-	8+8	-	24	28	14.3
Type3-Type4	2	2 + 2	2	-	8	2	18	22	18.2
Type4-Type4	2	2 + 2	2	-	-	2+2	12	16	25

The source addresses and the header information of the messages are the same in the resulting message, regardless of the scenario, owing to the fact that the messages are combined in the nodes' own output channels. Therefore, each one takes two bytes in the resulting message. However, since the data of the combining messages are different, the resulting message does not gain profit in size from the data, regardless of the scenario. The size of the sharer addresses, the destination addresses and the requestor addresses differs with the scenario. For the worst case scenario, since the values of each address are different in the combining messages, each of them occupies a place of the multiplication of their own sizes with the number of the combining messages in the resulting message, as is seen in Table 3.4. However, since each of these addresses are equal in the combining messages, they put in an appearance only once in the resulting message, regardless of the number of the combining messages, for the best case scenario, as is seen in Table 3.5.

Table 3.5. The estimation of the size and the decreasing rate of the resulting message for the two messages combining under the second scenario

	Source address	Destination address	Other header info	Data	Sharers	Requestor	Total Combined	Total Separate	Decreasing Rate (%)
Type0-Type0	2	2	2	-	-	-	6	12	50
Type0-Type1	2	2	2	18	-	-	24	30	20
Type0-Type2	2	2	2	18	8	-	32	38	15.8
Type0-Type3	2	2	2	-	8	-	14	20	30
Type0-Type4	2	2	2	-	-	2	8	14	42.8
Type1-Type1	2	2	2	18+18	-	-	42	48	12.5
Type1-Type2	2	2	2	18+18	8	-	50	56	10.7
Type1-Type3	2	2	2	18	8	-	32	38	15.8
Type1-Type4	2	2	2	18	-	2	26	32	18.7
Type2-Type2	2	2	2	18+18	8	-	50	64	21.9
Type2-Type3	2	2	2	18	8	-	32	46	30.4
Type2-Type4	2	2	2	18	8	2	34	40	15
Type3-Type3	2	2	2	-	8	-	14	28	50
Type3-Type4	2	2	2	-	8	2	16	22	27.3
Type4-Type4	2	2	2	-	-	2	8	16	50

Table 3.6. Average of the decreasing rates of the first and the second scenario for two messages combining

	Decreasing Rate(Scenario 1)	**Decreasing Rate(Scenario 2)**	**Average Decreasing Rate**
Type0-Type0	33.3	50	**41.65**
Type0-Type1	13.3	20	**16.65**
Type0-Type2	10.5	15.8	**13.15**
Type0-Type3	20	30	**25**
Type0-Type4	28.5	42.8	**35.65**
Type1-Type1	8.3	12.5	**10.4**
Type1-Type2	7.1	10.7	**8.9**
Type1-Type3	10.5	15.8	**13.15**
Type1-Type4	12.5	18.7	**15.6**
Type2-Type2	6.25	21.9	**14.075**
Type2-Type3	8.7	30.4	**19.55**
Type2-Type4	10	15	**12.5**
Type3-Type3	14.3	50	**32.15**
Type3-Type4	18.2	27.3	**22.75**
Type4-Type4	25	50	**37.5**

Average of the decreasing rates in the size of the resulting message according to the total size of the messages without combining is used in the simulations for all combining types. These rates are given in Table 3.6. for two messages combining. Since the estimation methods for three and four messages combining, which occupy a pretty much space, are the same with two messages combining, only average of the decreasing rates are given for three and four messages combining, as is seen in Figure 3.5. and Figure 3.6., respectively. In the Figures, the vertical axis states the type of the combining messages, while the horizontal axis expresses average

decreasing rate of the first and the second scenario. For instance, " 0-0-0-0" in Figure 3.6. denotes the combining operation of four *Type0* messages.

3.3.2. The Message Priority Algorithm

A message in the input queues of a node is waiting to be serviced for either of the cache, the directory or the processor. We give different types of priorities to the messages waiting in the input queues to increase the performance of the system. There are four types of priorities that include the cache priority, the directory priority, the cache & directory priority and the critical path priority. Table 3.7. demonstrates which priority type gives the priority to which messages in the system. The cache priority, as the name suggests, provides the cache messages (*DOWN_GR_REQ*, *INV_REQ*, *INV_WB_REQ* and *BLOCK*) to forge ahead, while the directory priority brings the directory messages (*DATA_REQ*, *DOWN_GR_WB_ACK*, *OWN_REQ*, *INV_WB_ACK*, *BLOCK_ACK*, *SHARER_LIST*, *RESPONSE_HOME* and *UPGRADE_OWN_REQ*) forward in the input queues of the nodes. The cache & directory priority gives precedence to both the cache and the directory messages (the entire system messages except *DATA_ACK*, *INV_ACK* and *OWN_ACK*), while the critical path priority puts only the messages (*DATA_REQ*, *DOWN_GR_REQ*, *OWN_REQ*, *INV_REQ*, *UPGRADE_OWN_REQ*, *INV_WB_REQ* and *SHARER_LIST*) that are directly in the response path of a request message (i.e. on the critical path) first.

The point to be taken into consideration that the processor messages are not given a priority in the input queues. Since the messages waiting to be serviced by the processor are consist of the acknowledgement messages, i.e. *DATA_ACK*, *INV_ACK* and *OWN_ACK*, attaching no priority to these kinds of messages can be prima facie considered as preposterous. However, it should be kept in mind that the priority operation is conducted in the input queues of the nodes. Accordingly, an acknowledgement message waiting to be serviced in the input queue represents that its response has already been returned and it is waiting for the purpose of creating a new request message. Therefore, these kinds of messages are not in the return path of

a response of a request message that has been released to the network and do not contribute to the network response time.

Table 3.7. The prioritized system messages according to the priority type

	The cache priority	The directory priority	The cache & directory priority	The critical path priority
DATA_REQ	-	X	X	X
DOWN_GR_REQ	X	-	X	X
DOWN_GR_WB_ACK	-	X	X	-
DATA_ACK	-	-	-	-
OWN_REQ	-	X	X	X
OWN_ACK	-	-	-	-
INV_REQ	X	-	X	X
INV_WB_ACK	-	X	X	-
INV_ACK	-	-	-	-
BLOCK	X	-	X	-
BLOCK_ACK	-	X	X	-
UPGRADE_OWN_REQ	-	X	X	X
INV_WB_REQ	X	-	X	X
RESPONSE_HOME	-	X	X	-
SHARER_LIST	-	X	X	X

The memory consistency model and the cache coherence are protected during the priority operation is implemented on the messages. The memory consistency model used in this thesis allows the messages to counteract the others if and only if they deal with different memory blocks. The cache coherence in the system is conducted by the home directories. The home directory decides whether a new request on a memory block to be released to the network or not due to the coherence rules if a request has already been processed on that memory block. Therefore, even though a directory message is prioritized in the input queue, if it is necessary to keep the message waiting according to the coherence rules, the message stands by until the previous request has been completed. This situation is not go for the cache messages because the cache messages waiting in the input queues of a node do not have a vitiating effect on the cache coherence.

4. RESULTS AND DISCUSSION

Simulations have been run for 10 hours and performance measures such as average network response time, average input waiting time and average channel waiting time are collected under different traffic patterns for different values of T/R before applying the message priority and the combining mechanisms (base system), after applying the message combining mechanism individually on the base system (base + combining), after applying the message priority mechanism individually on the base system (base + priority), and after applying the message priority and the combining mechanisms (base + priority + combining) together on the base system. Average network response time is the time interval between the instant when a cache miss causes a message to be enqueued in the output channel until the instant when the corresponding data or acknowledge message arrives at the input queue. Average input waiting time is the time interval between a message is enqueued in the input queue of a node until it gets serviced while average channel waiting time is the time interval between the message is enqueued in the output channel queue until it gets serviced. Wherefore the underlying architecture is 2D, average channel waiting time is calculated as the average of average horizontal and average vertical channel waiting times.

4. 1. Theoretical Analysis

Before investigating the performance of the message combining and the message priority algorithms, it would be beneficial to perform simple theoretical analysis to validate the simulation results reported in Section 4.2. Figure 4.1 and Figure 4.2 show the network path of a message for reply forwarding protocol for a read and a write request to different types of cache blocks, respectively. It is assumed that the size of Type1 packet is x. Thus, the sizes of Type2, Type3 and Type4 packets are 4x, 5x and 2x, respectively. In the Figure, for a write request to a shared block, ax / bx denotes that the size of the packet is ax if the request is an ownership request and is bx if the request is an upgrade ownership request.

As is seen in Figures, the network path of a packet, the number of transactions and the load on the network (i.e. total size of the packets transported on the network per request) is the same for both read and a write request to an invalid state block and it has the minimum critical path through all block states with the value of 5x. Besides, a read request to a modified state block is identical with a write request and the critical path is 6x.

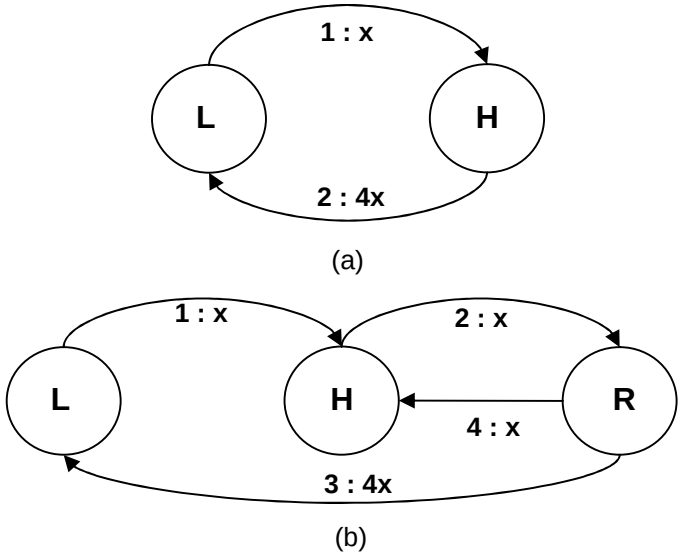


Figure 4.1. Message path of reply forwarding protocol for a read request to a (a) Shared / Invalid and (b) Modified block

A read request to a shared block has the minimum critical path just like an invalid block. However, since a write request to a shared block requires invalidation messages as the number of sharers, the critical path is the longest among all request types to all types of block states with the value of 8x. Although the down grade and invalidation write back messages (i.e. The fourth message in Figure 4.1 (b) and Figure 4.2 (b); the fifth message in Figure 4.2 (c)), which are transmitted from the caches to the home directories for a purpose of respond do not affect the critical path of the requests, they give rise to the traffic of the network to get congested.

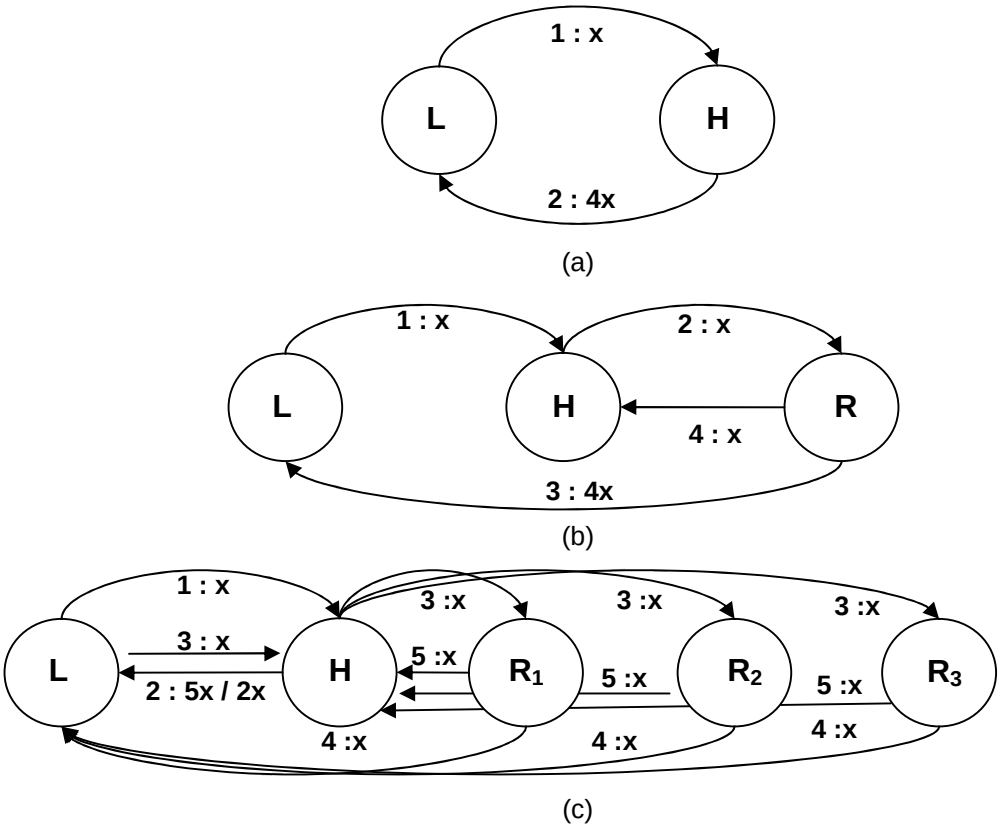


Figure 4.2. Message path of reply forwarding protocol for a write request to an (a) Invalid, (b) Modified and (c) Shared block

4. 2. Simulation Results

The simulation results for the base and the base + combining systems are demonstrated with average network response time and average channel waiting time, because the combining operation is implemented on output channels and the message combining algorithm performance on average network response time can be evaluated by the decreasing in average channel waiting time. However, using the results of average input waiting time is conformable to investigate the performance of the message priority algorithm on average network response time as the priority operation is applied on input queues. Therefore, in conjunction with average network response time and average channel waiting time results, average input waiting time are also included for the evaluation of the base + priority performance.

Considering that the combined message number has a direct effect on average channel waiting time, the exchange in average channel waiting time originated from the message combining algorithm for all combining types can be observed in a single chart. However, the circumstance is slightly different for the message priority algorithm. Giving a priority o a definite message type brings about other types of messages to wait much more in the input queues. Accordingly, the performance analysis through average input waiting time of all types of messages can lead up to an incorrect interpretation of the priority algorithm. On this account, it would be beneficial to figure average input waiting time results before and after the message priority algorithm for each priority types, individually. Since average input waiting time and average channel waiting time demonstrate similar tendency under different traffic patterns, only the figures for UN traffic is included as an illustration in this thesis to avoid the figure accumulation. Only average network response time figures are included for other traffic patterns to annotate the priority algorithm performance. Correspondingly, average network response time is sufficed for the evaluation of the performance of the message combining and the message priority algorithms together on the base system.

Besides the performance measures, such as average network response time, average channel waiting time and average input waiting time, maximum number of messages waiting both in the input queues and the output channels through the simulation are included on the purpose of forming an opinion about the necessary buffer sizes and observing the implemented priority and combining algorithm performances on the message traffic flow of the input queues and the output channels.

The vertical axis of the all figures presented in this section shows the performance measures in terms of clock cycle, while the horizontal axis shows the ratio of T/R. In all Figures, “ M” represents P(M) and “ W” represents P(W), while “ R“ represents the mean of thread run time and “ T” represents the channel transfer time, which is proportional to the size of packets. The service time for a packet with data changes from 5 to 100 (because the rate T/R changes from 0.05 to 1).

4.2.1. The Base System

Figure 4.3 through Figure 4.6 show average network response time and Figure 4.7 through Figure 4.10 show average channel waiting time of the base system for different values of T/R, P(M) and P(W) under UN, HR, BR and PS traffic, respectively.

As is seen in Figure 4.3 through Figure 4.6, average network response time under a traffic pattern significantly differs from the others for the higher values of T/R and has the highest value (the lowest performance of the system) under BR traffic by landslide, while HR, PS and UN traffic patterns follow it, respectively. The difference of the performance between the latter three traffic patterns is quite smaller than that of BR.

As is seen in Figure 4.7 through Figure 4.10, average channel waiting time under a traffic pattern significantly differs from the others for the higher values of T/R and has the highest value under BR traffic, just as average network response time. The difference is that average channel waiting time under PS traffic run a close second to the BR traffic, while HR follows BR for average network response time. Average channel waiting time under HR and UN traffic is smaller than that of other traffic patterns and their values are closer to each other.

The home directory node is predetermined, constant and individual for each requestor node under BR traffic. Unfortunately, the nature of 2D architecture usually leads to the requestor and the home nodes to be in different horizontal and vertical buses, which eventuates in the usage of intermediate nodes. Accordingly, an intensive traffic occurs continuously. As a consequence, both average network response time and average channel waiting time has the highest value under BR among all traffic patterns, as is seen in Figure 4.5 and Figure 4.9.

As explained in Section 3.1.2, some definite nodes are selected over all nodes as a home node under HR traffic, which gives rise to the congestion to be formed around these nodes. Therefore, average network response time under HR is higher than that of UN traffic, while lower than that of BR, as is seen in Figure 4.4.

An individual home directory node is selected by requestor nodes under PS traffic just as BR. Accordingly, average channel waiting time has higher values than the remainder two traffic patterns (UN and HR), as BR (Figure 4.10). However, the system under PS does not use intervention nodes as much as it does under BR. Thus, average network response time under PS has lower values than BR and slightly higher values than UN, as is seen in Figure 4.6.

Owing to the increase in the service time of a message as T/R increases, both average network response time and average channel waiting time increase as T/R increases for all traffic patterns, P(M) and P(W) values, as is seen in Figure 4.3 through Figure 4.10. Besides, average network response time increases as P(W) increases and the increasing rate decreases with the increase in P(M) for all traffic patterns (Figure 4.3 through Figure 4.6). Thus, the increase in P(W) decreases the system performance much more for the lower values of P(M) than the higher values of P(M). When the theoretical analyses in Section 4.1 are investigated, it can be clearly observed that the number and the weight of the network transactions on the critical path produced for a modified and an unowned block are the same for a read request with a write request. However, a block in shared state causes more transaction (both in number and weight) to be produced for a write request than a read request. Therefore, the increase in P(W) increases the response time for all situations and with the increase in P(M), the system behavior approaches to a read request that decreases the increasing rate inclined from P(W).

As is seen in Figure 4.7 through Figure 4.10, average channel waiting time decreases as P(W) increases and the decreasing rate diminishes as P(M) increases for all traffic patterns. Although the channel waiting time is in direct proportion to the response time, it can be conveniently said from the results that the behavior of average channel waiting time is exactly the opposite of average network response time. This situation can be seen unreasonable at first appearance. However, the answer is hidden in the theoretical analyses. As is explained above, the increase of P(W) increases the number and the weight of the transactions on the critical path that causes higher average network response time to be raised. However, it distributes the traffic and the congestion of the network, which brings on the channel waiting times

of the messages to be lowered. In other words, the messages wait less on the channels but they have to pass more transactions to complete the critical path with the increase of P(W). Similarly, since the transaction number generated for a write request to a shared block is more than that of a modified block, average network response time decreases, whereas average channel waiting time increases as P(M) increases.

The increase in P(M) for the lower values of P(W) decreases the performance of the system with increasing average network response time for all values of T/R under UN traffic, as is seen in Figure 4.3. The same effect can be observed slightly for the middle values of P(W). However, for the higher values of P(W), the system is adversely affected from the increase of P(M). The increase in P(M) increases the performance of the system with decreasing average network response time. The reasons of this situation can be clearly obtained from the theoretical analyses. Lower write probability corresponds to higher read probability. Under this situation, if a block is in modified state, extra network transactions (6x) are created on the critical path than it is in shared or unowned state (5x). Therefore, the increase in average network response time is an expected result. However, a block in shared state causes extra network transactions to be created on the critical path than a modified state block for the higher values of P(W). Thus, under this situation the increase in P(M) increases the system performance.

HR is the traffic pattern that extra traffic on some definite home nodes is created. When the effect of P(M) is investigated for the lower values of P(W), it can be said that when P(M) increases, although average channel waiting time decreases due to the canalizing of the traffic to the other nodes, average network response time increases due to the extra transaction on the return path, as explained above for UN traffic. The same tendency can also be seen under HR traffic for the lower values of T/R (Figure 4.4). However, an exact opposite effect is observed for the middle-to-higher T/R values under HR. The reason of this situation is that the congestion in the nature of HR traffic becomes more intense for the lower values of P(M) and the distributor effect of P(M) can yield higher performance especially for the higher values of T/R. Although more transaction is needed for the return path, each

transaction is completed on a lighter traffic that results in a better performance (lower average network response time). The distributing effect of increasing P(M) can precisely seen under BR traffic (especially with the increase in P(W)), where an intensive traffic is continuously generated, as is seen in Figure 4.5.

Average network response time for the lower values of P(W) increases as P(M) increases for the lower values of T/R, while stands almost the same for the higher values of T/R under PS traffic (Figure 4.6). The difference of the system behavior under the middle values of P(W) from the lower values of P(W) is that average network response time decreases for the higher values of T/R. Ultimately, for the higher values of P(W), average network response time decreases for all situations and the decreasing rate increases as T/R increases. The distributing effect of increasing P(M) rises to the occasion under PS just as HR and BR. However, due to less often intervention node usage, the nature of PS is less congested than BR, which causes the distributing effect to be occurred less under PS than BR, just as HR.

As is seen in Figure 4.8, average channel waiting time under HR traffic shows similar tendency to UN, as other traffic patterns. The difference is that the decreasing rate on average channel waiting time with increasing P(W) values is higher for all situations. For instance when P(M) is 0.2 under UN traffic (HR), the decreasing rate in average channel waiting time passing from P(W) = 0.2 through P(W) = 0.4 is 16% (26%), while from P(W) = 0.2 through P(W) = 0.6 is 28% (42%). Since extra congestion on some definite home directory nodes is created by the nature of HR traffic, the traffic distributing effect of the increase in P(W) yields better performance.

4.2.2. The Base System with the Message Combining Algorithm

Figure 4.11 through Figure 4.14 show average network response time and Figure 4.15 through Figure 4.18 show average channel waiting time of the base system (as symbolized with “ no_comb” in Figures) with the message combining algorithms, including two messages combining (as symbolized with “ 2_comb” in

Figures), three messages combining (as symbolized with " 3_comb" in Figures) and four messages combining (as symbolized with " 4_comb" in Figures), for different values of T/R, P(M) and P(W) under UN, HR, BR and PS traffic, respectively.

As is seen in Figure 4.11 through Figure 4.14, the message combining algorithm increases the performance of the system by decreasing average network response time regardless of the number of combined messages, for all P(M), P(W) and T/R values. The increasing rate is higher for the higher T/R values than the lower-to-middle T/R values. Since the increasing in T/R values makes the channels to be denser for all traffic patterns, more messages are tend to wait in the channel queues which results in the message combining algorithm to perform better, as is seen in Figure 4.15 through Figure 4.18.

The message combining algorithm yields the best performance for four messages combining for all traffic patterns and three messages combining follows it. The performance of the message combining algorithm can obviously be seen in the higher T/R values. Average decreasing rate in average network response time over all T/R values is between 6% and 12% under UN (Figure 4.11), 7.3% and 12.5 % under HR (Figure 4.12), 8.3% and 19% under BR (Figure 4.13), %8.4 and 14.4% under PS traffic patterns (Figure 4.14), while maximum decreasing rate in average network response time over all T/R values is between 13.75% and 22.7% under UN (Figure 4.11), 12.9% and 24% under HR (Figure 4.12), 12.3% and 26.8% under BR (Figure 4.13), 13.3% and 26.2% under PS traffic patterns (Figure 4.14). Besides, average (maximum) decreasing rate in average channel waiting time over all T/R values reaches up to 22.6% (32.45%) under UN (Figure 4.14), 21.3% (32.8%) under HR (Figure 4.15), 29.8% (74.8%) under BR (Figure 4.16) and 28.14% (43.2%) under PS traffic pattern (Figure 4.17).

The message combining algorithm performs the best under bit permutation traffic models, especially under BR, where average channel waiting time is the highest, the network traffic is the most congested and the channels are the most intense. The decreasing in average channel waiting time reflects to average network response time, as expected. PS traffic pattern follows BR, while the message combining algorithm performs better under HR traffic than it does under UN due to

its reducing effect on the congestion of the home directory nodes, which is the bottleneck of the network traffic. Consequently, the message combining algorithm yields better performance under the traffic patterns, which uses the channels intensively.

The effect of P(M) and P(W) on the message combining algorithm differ greatly under different traffic patterns. When the effect of P(M) under UN traffic is investigated, it can be said that the increase in P(M) for the lower values of P(W) increases the decreasing rate of average channel waiting time (thus yields even more performance) both on average and on maximum over all T/R values for all combining types because a modified block causes more transactions to be generated and uses the channels much more frequently than the other states, as is seen in Figure 4.15. The frequent usage of the output channels brings on more messages to be combined. However, this performance increase does not reflect to average network response time. The performance increase of message combining algorithm on average network response time decreases as P(M) increases, as is seen in Figure 4.11. For instance, the performance increasing rate is 12% on average and 22.7% on maximum when P(M) is 0.2, whereas it reaches up to 9.8% on average when P(M) is 0.6. Although the difference is inconsiderable, it is originated from the intense traffic on the channels for the lower P(M) values that results in the message combining algorithm to perform better on average network response time. For middle-to-higher values of P(W) under UN traffic, the change of P(M) values does not significantly affect neither average network response time nor average channel waiting time. Moreover, the performance of the base system for different values of P(M) does not considerably differentiate, as is explained above.

The analysis of the base system variance over P(W) values demonstrate that the increase in P(W) increases the message combining performance on average channel waiting time for all combining types. The higher P(W) values stir more transaction and traffic up to be created that results in much more success of the message combining algorithm on the channels. However, since the performance of the base system over average network response time decreases as P(W) increases, the message combining performance achievement on average channel waiting time does

not reflect to average network response time. Therefore, the more P(W) leads to the less increase in performance of the message combining algorithm on the base system.

The increase of P(M) values for the lower values of P(W) under HR traffic yields more performance increase of the message combining algorithm on average network response time (Figure 4.12) for all combining types and on average channel waiting time for two message combining (Figure 4.16). There is not a considerable effect of P(M) values for the lower values of P(W) on average channel waiting time for three and four messages combining. The major factor, which has an impact on the performance of the system under HR is the home directory nodes that an extra traffic is created on. The alteration of P(M) for middle values of P(W) under HR traffic does not significantly affect the message combining algorithm performance neither on average network response time nor on average channel waiting time as UN. Besides, the more P(M) for the higher values of P(W) leads to the less increase in performance of the message combining algorithm both on average network response time and on average channel waiting time. Although the increase in P(M) for the lower values of P(W) disentangles the traffic by canalizing it to the other nodes, the increase in the probability of a block to being in shared state undertakes this task for the higher values of P(W). Accordingly, the increase in P(M) has an attenuator impact on the performance increase, which grows out of the message combining algorithm. The rise in the value of P(M) provides the message combining algorithm to perform better on average channel waiting time regardless of the number of combined messages for lower-to-middle values of P(W), while engenders a downfall in the message combining algorithm performance increase for the higher values of P(W). This circumstance passes through to average network response time as an inconsiderable change for lower-to-middle P(W) values and a decreasing on the increase of performance for the higher P(W) values.

Both the increase in P(W) and P(M) raises the performance of all message combining algorithm types on average network response time under BR traffic, as is seen in Figure 4.13. The enhancement of the message combining algorithm on average channel waiting time increases as P(M) increases for the higher P(W) values

and as P(W) increases for all P(M) values, as is seen in Figure 4.17. BR traffic is the model that an intense traffic is continuously generated. Therefore, both the increase in P(M) and P(W) enhances the performance of the message combining algorithm by forwarding more messages to the output channels. The higher P(W) or P(M) values causes more messages to be directed to output channels, thus the increasing in P(M) can demonstrate its performance enhancing effect on average channel waiting time.

The exchange in both P(M) and P(W) values scarcely affect the performance of the message combining algorithm on average network response time under PS traffic, as is seen in Figure 4.14. When the performance of the algorithm on average channel waiting time under PS is investigated it can be said that the change in P(M) values don' t not have a considerable effect while the increase in P(W) values increases the performance, as is seen in Figure 4.18. Every node generates its block request to a dedicated home directory node under PS traffic as BR. Therefore, average channel waiting time of the base system has higher values under PS than HR and UN traffic, as explained above. However, the usage of intermediate nodes is not as frequent as BR traffic. Thus, the increase in P(W) has an increasing effect, while P(M) does not.

Table 4.1 through Table 4.3 show maximum number of messages in the input queues, the horizontal output channels and the vertical output channels of the base system with message combining algorithm, for different values of traffic pattern, P(M) and P(W), respectively. As is seen from Table 4.2 and Table 4.3, maximum number of messages both in the horizontal and the vertical output channels decreases with message combining, as expected. Owing to the home node selection rule, maximum number of messages waiting in the horizontal output channels is less under bit permutation traffic patterns than the others, as is seen in Table 4.2. Therefore, a buffer with a smaller size is sufficient for bit permutation traffic patterns. The sublimit of the horizontal output channel buffer size is 95 and is equal to maximum number of messages waiting in the horizontal output channels under UN traffic pattern. The frequent intermediate node usage under BR traffic pattern causes maximum number of messages in the vertical output channels to be considerably higher under BR than the other traffic patterns. Besides, the sublimit of

the vertical output channel buffer size is 193 (Table 4.3). Since the home nodes of 25% is selected randomly within a small " hot" contiguous sub-mesh region consisting of 12.5% of the nodes, maximum number of messages in the input queues are considerably higher under HR than the other traffic patterns and the necessary input buffer size is minimum 372.

4.2.3. The Base System with the Message Priority Algorithm

Figure 4.19 through Figure 4.22 show average network response time of the base system (as symbolized with " no_pr" in Figures) with the message priority algorithms, including the cache messages priority (as symbolized with " cache" in Figures), the directory messages priority (as symbolized with " dir" in Figures), the cache and directory messages priority (as symbolized with " dircache" in Figures) and the critical path messages priority (as symbolized with " c_path" in Figures), for different values of T/R, P(M) and P(W) under UN, HR, BR and PS traffic, respectively. Average channel waiting time of the base system with the message priority algorithms under UN traffic is shown in Figure 4.23 while average input waiting time of the base system with the cache, the directory, the cache & directory and the critical path priority under UN traffic are illustrated in Figure 4.24 through Figure 4.27, respectively.

As is seen in Figure 4.19 through Figure 4.22, the performance of the message priority algorithm on average network response is higher for the lower T/R values regardless of the traffic patterns, P(W), P(M) values and the combining types and decreases as T/R increases. Although, exceptional circumstances can be scarcely encountered in patches for some higher T/R values, it can be observed that the message priority algorithm increases the performance of the system by decreasing average network response time. This success is owed to the decrease in average input waiting time. As is seen in Figure 4.24 through Figure 4.27, the message priority algorithm yields better performance on average input waiting time for the lower values of T/R, regardless of the priority types. The aim of the message priority algorithm is performing a decreasing effect on the response time of a message by

canalizing the messages from the input queues to the output channels. This is the reason lied behind the high performance of the message priority algorithm for the lower T/R values. As the messages are mandatorily wait in the output channel queues instead of the input queues for the higher T/R values, the message combining algorithm does not demonstrate a good performance on average network response time.

The critical path priority algorithm yields the best performance over all priority types on average network response time under UN traffic as is seen in Figure 4.19. The cache & directory priority follows it, while the cache priority performs better than the directory priority for the higher P(W) values. The directory priority mostly has the lowest performance over all priority types. If average input waiting time results are investigated from Figure 4.24 through Figure 4.27, it can be observed that the performance ordering of the priority types on average network response time clarified above is identical with average input waiting time. The performance increase of average input waiting time originated from the critical path priority reaches up to 66.5%, whereas the highest performance increase takes its source from the cache & directory priority (the cache priority) is 51.2% (36.5%).

As P(W) increases, the performance increase on average network response time decreases for all priority types regardless of the other DSM parameters under UN traffic. For instance, the performance increase of average network response time while P(W) is 0.2 is 27% on average and 61% on maximum of all T/R values. When P(W) is 0.4 (0.6), the same rates drop down to 14% (8%) on average and 34% (15%) on maximum. The key consideration that determines the effect of the message priority algorithms on average network response time is the algorithm performance on average input waiting time. The message priority algorithm engenders a recessive exchange on average channel waiting time while it influence average input waiting time appreciably. The tendency of average input waiting time resulted from the implementation of the message priority algorithm exactly passes through to average network response time. As is seen from the example presented above, the increasing performance rate of the critical path priority on average network response time decreases approximately 50% as P(W) increases. Correspondingly, the performance

increase of average input waiting time while P(W) is 0.2 is 42% on average and 66% on maximum of all T/R values. When P(W) is 0.4 (0.6), the same rates drop down to 22% (13%) on average and 42% (23%) on maximum. This is due to the fact that the message priority algorithm decreases average input waiting time while increases average channel waiting time, as is seen in Figure 4.23, by canalizing the traffic of the messages from input queues to output channels. As the nature of increasing P(W) values brings back extra traffic to be generated on the output channels, it has a decreasing effect on the message priority algorithm performance. Average network response time decreasing effect of the cache priority algorithm both on average and on maximum over T/R values increases as P(M) increases for lower-to-middle P(W) values. A read request to a modified state block gives occasion to an additional usage of the cache messages than both shared and unowned states, while a write request to a modified state block brings about an extra usage of the cache messages than unowned state but less usage of cache messages than shared state, as is seen from theoretical analyses in Section 4.1. The more cache messages bring back the more performance increase of the cache priority algorithm. Therefore, the exchange in P(M) values for the higher values of P(W) does not have a steady effect on the performance of the cache priority algorithm on average network response time. The leaning of the directory priority algorithm performance on average network response time with the increasing P(M) values under UN traffic is exactly the opposite of the cache priority algorithm. In other words, the system performance increasing effect of the cache priority algorithm both on average and on maximum over T/R values decreases as P(M) increases for lower-to-middle P(W) values. Wherefore this situation accompanies intensively usage of the caches instead of the directories, as explained above, it affects the performance of the cache priority algorithm positively, while it takes quite a toll on the performance of the directory priority algorithm. The cache and directory priority algorithm effect on average network response time is similar to that of the directory priority algorithm. This tendency can also be observed in average input waiting time results. Accordingly, this situation is originated from the performance attenuator effect of the cache & directory priority algorithm on average input waiting time. As is explained above, while the increasing P(M) values

for lower-to-middle P(W) values increases the performance of the cache priority algorithm, it decreases the performance of the directory priority algorithm on average network response time. The behavior of the directory algorithm surpasses on the cache & directory priority algorithm, which leads to the decreasing of the performance increase. The impact of increasing P(M) values on the performance of the critical path priority algorithm on average network response time is investigated, it can be clearly said that it is grounded on the behavior of the algorithm on average input waiting time and the performance increase decreases for the lower values of P(W) while increases for the higher values of P(W). The results show that the critical path priority algorithm performs better for the states, which have shorter critical paths on the base system.

As P(W) increases, the performance increase on average network response time decreases for all priority types regardless of the other DSM parameters under HR traffic, as UN traffic, as is seen in Figure 4.20. The average network response time decreasing (i.e. performance increasing) rates grow out of the message priority algorithms are lower under HR than that of under UN for all priority types regardless of the P(M) or P(W) values. The difference of the system behavior under HR from UN is that the best performed algorithm is the cache & directory priority instead of the critical path priority. The critical path priority only gives preference to the messages that directly hold a place on the path of the response of a request (i.e. on the critical path). However, the system generates some special messages between the directories and the caches like BLOCK messages, which do not have a direct effect on the network response time, whereas give occasion the traffic of the network to get crowded. Giving a priority to this kind of messages provides the priority algorithm to perform well under HR traffic, where an extra intensity is generated on some definite home directory nodes. The cache priority algorithm shows similar behavior under HR and UN traffic. The difference is that the increasing rates are lower under HR than that of UN. Owing to the extra traffic on the output channels of some definite nodes created by the nature of HR, these output channels get more congested with the forwarding of the messages from the inputs by the priority algorithms that results in the priority algorithm performance under HR to be lower than that of under UN.

The performance of the directory priority algorithm on average network response time under HR is lower than that of UN and is not significantly affected from the exchange of P(M) values. Giving a priority to the directory messages in the input queues causes more directory messages to be forwarded to the output channels, where an extra intensity is already generated by HR. Therefore, the performance of the algorithm under HR is lower than that of UN.

The increase in P(M) for all P(W) values increases the performance of the message priority algorithm on average network response time regardless of the priority type under BR traffic, differently from UN and HR traffic patterns, as is seen in Figure 4.21. As explained in Section 4.2.1., owing to the distributing effect of increasing P(M) values on the intensive traffic of the output channels, which is created by BR traffic, average network response time decreases as P(M) increases. Therefore, the message priority performance on average network response time also increases as P(M) increases. As P(W) increases, the performance increase on average network response time decreases for all priority types regardless of the other DSM parameters under BR traffic, as UN and HR traffic. The performance of the cache priority algorithm on average network response time has the lowest value under BR traffic over all traffic patterns, while the critical path priority has the highest performance on average network response time over all priority types under BR. The directory priority under BR demonstrates higher performance than that of UN and HR traffic for middle-to-higher P(W) values.

PS is the traffic pattern that the directory priority algorithm performs the best on average network response time over all traffic patterns for all DSM parameters because each requestor node communicates with its individual home directory node and the flow of the traffic has a regular intense, as is seen in Figure 22. Although the tremendous usage of intervention nodes under BR traffic prevents the directory priority algorithm to totally show its performance on average network response time, the algorithm demonstrates its highest performance under PS traffic. The critical path priority algorithm yields the best performance over all priority types on average network response time under PS as UN traffic. Moreover, its performance is the highest under PS over all traffic patterns for the middle-to-higher P(W) values. Both

the performance of the critical path and the cache & directory priority algorithm on average network response time increases as P(M) increases for all values of P(W). The performance of the directory priority algorithm decreases as P(M) increases for the lower-to-middle values of P(W) and increases as P(M) increases for the higher values of P(W). The cache priority algorithm performance under PS is lower than that of UN and HR, whereas higher than that of BR. The results show that the directory priority algorithm performs better under the bit permutation traffic models, i.e. BR and PS, while the cache priority algorithm performs worse.

Table 4.4 through Table 4.6 show maximum number of messages in the input queues, the horizontal output channels and the vertical output channels of the base system with message priority algorithm, for different values of traffic pattern, P(M) and P(W), respectively. As is seen from Table 4.4, maximum number of messages in the input queues decreases with message priority, as expected. Since the home directories are responsible from providing the cache coherence, the decreasing rate is the lowest for directory priority over all priority types. The sublimit of the vertical output channel buffer size is 196 (Table 4.6), while the sublimit of the horizontal output channel buffer size is 95 (Table 4.5). Besides, the necessary input buffer size is minimum 364, as is seen in Table 4.4.

4.2.4. The Base System with the Message Priority and the Message Combining Algorithm

Since the performance evaluation of the combination of three kinds of message combining and four kinds of message priority algorithms on average network response time is fairly difficult to observe in a single chart, the decreasing rate on average network response time of the base system resulted from implementing the combinations is shown in tables for different values of T/R, P(M) and P(W). The average and the maximum values of the rates are also included in the tables to take advantage of easy comparison and interpretation. The decreasing percentage rate on average network response time of the base system according to the different combinations of the message combining and the message priority

algorithms for different values of P(M), P(W) and T/R under UN traffic is shown in Table 4.7 through Table 4.15; under HR traffic is shown in Table 4.16 through Table 4.24; under BR traffic is shown in Table 4.25 through Table 4.33 and under PS traffic is shown in Table 4.34 through Table 4.42.

As is seen in Table 4.7 through Table 4.15, the message priority algorithm is the most dominant factor on the performance of average network response time for the lower values of T/R, while the message combining flows through on the performance of average network response time for the higher T/R values regardless of P(M) and P(W) values under UN traffic. When the system performance for P(M) is 0.2 and P(W) is 0.2 is investigated it can be said that the performance difference between the message priority groups is higher than that of the message combining groups, as is seen in Table 4.7. For instance, when T/R is 0.05, the performance increasing rate originated from the critical path priority group gets through to 60%, the cache & directory priority group reaches up to 48%, the directory priority group goes down to 32% and the cache priority group gets through to 13%, whereas when T/R is 1, the performance increasing rate, which takes root from two combining group reaches up to 16%, three combining group goes down to 21% and four combining group gets through to 23%. Critical path priority with four messages combining has the highest performance on average network response time for the lower values of P(M) and P(W) (Table 4.7), where the critical path priority with three messages combining follows it. Although, the maximum performance of the cache & directory priority with four messages combining is lower than that of the critical path priority with two messages combining, it is higher on the average performance. The directory priority has higher performance increase than the cache priority regardless of the combining number. The performance increasing effect of the critical path or the cache & directory algorithm with all types of combining on average network response time for the lower values of T/R is higher than that of for the higher values of T/R. Therefore, the performance of the critical path priority and the cache & directory priority group on average network response time decreases as T/R increases. Although, the same situation is slightly observed under the directory priority with the lower values of combining, the system performance does not

decrease with increasing T/R under the directory priority with four messages combining. The cache priority shows exactly the opposite performance with the increase in T/R. The performance of the cache priority on average network response time increases as T/R increases. The results show that the most performance elevator parameter for the lower values of P(M) and P(W) is the critical path priority. The cache & directory priority stands a close second, while four messages combining and the directory priority are hard on the heels of it. The performance of the message combining on average network response time is higher than that of the cache priority for all types of combining. The cache priority affects the system performance at least over all priority and combining parameters. If a general deliberation of the performance of the system for the lower values of P(M) and P(W) is conducted, it can be said that the performance order of the priority algorithms from higher to lower is the critical path, the cache & directory, the directory and the cache priority, respectively. Correspondingly, four messages combining has the highest performance on average network response time over all messages combining. Three messages combining follows it, while two messages combining have the lowest performance.

The increase in P(M) from the value of 0.2. to 0.4 for the lower values of P(W) decreases the performance of the combining and the priority algorithm combinations except the cache priority groups both on the average and the maximum values of average network response time through T/R, as is seen in Table 4.8. The decreasing rate is recessive for the critical path priority, while is considerably high for the cache & directory and the directory priority groups. However, the performance increases as P(M) increases for the cache priority regardless of the combined message numbers. The performance order of the message combining for P(M) is 0.4 and P(W) is 0.2 is similar with the order for P(M) is 0.2 and P(W) is 0.2 (i.e. four messages combining yields the best performance, whereas two combining has the lowest). However, the performance order of the message priority for P(M) is 0.4 and P(W) is 0.2 is different than the order for P(M) is 0.2 and P(W) is 0.2. The critical path priority is still the best performed priority type and the cache & directory follows it. Nevertheless, the cache priority shows better performance than

the directory priority especially for middle-to-higher number of messages combining. The increase in T/R decreases the performance of the critical path priority and the cache & directory priority, while does not sorely affect the performance of the remainder two priority types. This is because the performance increasing effect of the critical path priority and the cache & directory priority on average network response time is higher than that of combining types.

As is seen in Table 4.9, the increase in P(M) (from 0.4 to 0.6) for the lower values of P(W) does not affect the performance of the cache priority groups much, whereas has an decreasing effect on the performance of the other types of combinations. Especially the directory priority is the most influenced with the decreasing rate of 50%. The highest two priority performance remains the same (the critical path and the cache & directory) for all P(M) values for the lower values of P(W), while the performance difference increases between the cache priority and the directory priority groups. The combining performance increases regardless of the priority types as P(M) increases from 0.4 to 0.6 for P(W) is 0.2 and is higher than the directory priority performance for all situations, while has approximate value with the cache priority.

If the system for the lower P(W) values is briefly investigated, it can be observed that the system benefits from the priority performance for the lower T/R values and the combining performance for the higher T/R values and the performance increase under the combination of the priority and the combining algorithms together is higher than that of both the combining and the priority performances, implemented individually on the base system, because the priority and the combining algorithms have adjuvant effect to each other. The messages that are forwarded from the input queues to the output channels by the priority algorithms are combined in the output channels by the combining algorithms. Since the critical path priority gives the priority to the messages that directly affects the network response time and four messages combining causes high numbers of messages to be combined, they have the highest effect on the performance of the system over all priority and combining types, respectively. The performance decreasing effect of the increasing P(M) values for the lower values of P(W) takes root from the lengthening

effect on the return path of a message. However, since the return path surmounts the caches more frequently, the cache priority performance increases, whereas the directory priority decreases as P(M) increases.

If the system behavior passing from the lower values of P(W) to the middle values of P(W) is investigated, it can be observed that the increase in P(W) inclines a decrease in the performance of all combinations both on the average and the maximum of average network response time over all T/R values, as is seen in Table 4.10 through Table 4.12. The decreasing rate, which diminishes as P(M) increases, is 27% on the average and 36% on the maximum under the critical path priority groups, while the same rates are 22% and 30% under the cache & directory priority groups; 23% and 45% under the directory priority groups, respectively. The increase in P(M), which smoothes over the decreasing effect of P(W), shows its maximum performance under the directory priority. The cache priority is the least effected from the increase in P(W) over all priority groups. Besides, it can be conveniently said that the increase in P(W) does not create a quite difference on the performance of the cache priority groups for the lower values of P(M). The effect of the increase in P(M) on the cache priority groups is exactly the opposite of that of the directory priority groups. Nevertheless, the maximum decreasing rate is 10%. The best performed combining algorithm is still four messages combining and three messages combining follows it as for the lower values of P(W). Although the critical path priority shows the highest performance over all priority groups, the performance of the cache & directory priority is within striking distance for the lower values of P(M), as is seen in Table 4.10. Even though the critical path priority draws away from the cache & directory priority as P(M) increases, the maximum performance difference solely comes up to 12,5% (Table 4.12). The directory priority groups demonstrate lower performance than that of the cache priority groups for all situations. It can be concisely said that the performance of the combining algorithms on average network response time for middle values of P(W) is better than that of the directory priority algorithms for all combining types and is better than that of the cache priority algorithms for higher combining types, whereas worse than that of the critical path priority and the cache & directory priority for all situations.

The increase in P(W) brings on a decrease in the performance of all combinations both on the average and the maximum of average network response time over all T/R values when passing from the middle values of P(W) to the higher values of P(W), as is seen in Table 4.13 through Table 4.15. The decreasing rate is 18% on the average and 30% on the maximum under the critical path priority groups, while the same rates are 10% and 26% under the cache & directory priority groups; 21% and 30% under the directory priority groups; 12% and 17% under the cache priority groups, which is affected at least from the increase in P(W) over all priority groups, respectively. The decreasing performance rate passing from the middle values of P(W) to the higher values of P(W) is higher than that of passing from the lower values of P(W) to the middle values of P(W) under the critical path, the cache & directory and directory priority groups, while it is lower than that of under the cache priority groups. Since the increasing P(W) values strikes through the usage of the caches, its performance decreasing affect bears itself the minimum under the cache priority groups, early on. However, the more increasing in P(W) values brings back the more congested traffic on the network that results in the more decreasing affect on the cache priority groups, whereas it engenders the more usage of the directories that leads up to the less decreasing effect on the directory priority groups. The directory priority tendency overbears the cache priority for the higher values of P(W) on the performance of the critical path and the cache & directory priorities, as explained above. Accordingly, the decreasing effect of the increasing P(W) decreases under the critical path and the cache & directory priorities, as the directory priority. Although four messages combining shows the highest performance over all combining groups, the performance of three messages combining is within striking distance. Furthermore, the performance of two messages combining is closer to four and three messages combining than it is for lower values of P(W).

For the higher values of P(W), the performance of all combining algorithms on average network response time is higher than that of all priority algorithms for the lower values of P(M), as is seen in Table 4.13. The performance difference increases as the number of combined messages increases. Three and four messages combining algorithms yield better performance than all priority types for the middle values of

P(M) (Table 4.14) and show similar performance with the critical path and cache & directory priority for the higher values of P(M) (Table 4.15). Even though the performance of the cache & directory priority approaches to the performance of the critical path priority for the higher values of P(W), a slight difference can be observed in patches. The cache priority is be hard on the heels of them, while the directory priority has relatively lower performance even a negative effect on average network response time in some cases.

As is seen from the results that the increase in P(W) under UN traffic provides the combining algorithms to perform better than the priority algorithms on average network response time. Owing to the congested traffic created by the nature of write requests, combining the messages waiting in the output queues disentangles the traffic of the network rather than forwarding them from input queues to output channels.

The message priority algorithm is the most dominant factor on the performance of average network response time for the lower values of T/R, while the message combining flows through on the performance of average network response time for the higher T/R values regardless of P(M) and P(W) values under HR traffic, as UN traffic, as is seen in Table 4.16 through Table 4.24. For the lower values of P(W), the performances of all combinations under HR traffic are lower than that of under UN (Table 4.16 through Table 4.18). Average decreasing rate of different combining types for the lower values of P(W) is 38% for the lower values of P(M) (Table 4.16); 36% for the middle values of P(M) (Table 4.17) and 21% for the higher values of P(M) (Table 4.18) for the critical path priority groups. The same rates are 31%, 27% and 17% for the cache & directory priority groups; 35%, 21% and 2% for the directory priority groups and 15%, 14% and 8% for the cache priority groups, respectively. The results show that the increase in P(M) values for the lower values of P(W) negatives the performance decreasing effect of HR traffic for all priority types due to the distributing effect on the congested traffic of the network. This effect is observed for the directory priority groups at the most and for the cache priority groups at the least. Although the cache & directory priority shows the highest performance over all priority groups, the performance of the critical path

priority is within striking distance for the lower values of P(M). However, the critical path priority forges ahead with the increasing P(M) values and eventually has higher performance than the cache & directory priority groups for the higher values of P(M). Nevertheless, the difference of the performances does not exceed 15%. The cache priority performs better than the directory priority and both have lower performance than the remainder two priority groups. The performance ordering of the combining types for the lower values of P(W) under HR traffic does not differ from that of under UN traffic, i.e. four messages combining has the highest performance, whereas two messages combining has the lowest. Even though, the increase in P(M) has an increasing effect on the performance of the cache priority groups, it does not differ the performances of the remainder priority groups much. However, the increase in P(M) increases the combining performances. The performance of the combining algorithms on average network response time is better than that of the directory priority, whereas worse than that of the critical path priority and the cache & directory priority. The performance ordering between the cache priority and the combining groups differentiates according to the combined number of messages and P(M) value.

The increase in P(W) from the value of 0.2. to 0.4 decreases both on the average and the maximum through T/R values of all combining and priority group performances on average network response time under HR traffic, as is seen in Table 4.19 through Table 4.21. The decreasing rate is the minimum for the cache priority groups and even is close to zero for the lower values of P(M). Although, as P(M) increases, the decreasing effect of the increasing P(W) values demonstrates its performance, the maximum rate entirely reaches to 9%. Average decreasing rate originated from the increase in P(W) is between 11% and 14% for the cache & directory priority groups; 18% and 25% for the critical path priority groups; 18% and 29% for the directory priority groups, which is the most affected priority type. Even though the cache & directory priority shows better performance than the critical path priority for the lower values of P(M), the increase in P(M) makes up the difference and eventually the performances are equalized for the higher values of P(M). The performance ordering of the combining types still does not change for the middle

values of P(W). The cache priority performs better than the directory priority and both have lower performance than the remainder two priority groups for the middle values of P(W) as for the lower values of P(W).

The increase in P(W) brings on a decrease in the performance of all combinations both on the average and the maximum of average network response time over all T/R values when passing from the middle values of P(W) to the higher values of P(W) under HR traffic, as is seen in Table 4.22 through Table 4.24. The decreasing rate is the minimum for the cache priority groups and even is close to zero for the lower values of P(M). Even though, the decreasing effect of the increasing P(W) values demonstrates its performance as P(M) increases, the maximum rate just rises to 14%. Average decreasing rate originated from the increase in P(W) is between 10% and 14% for the cache & directory priority groups; 15% and 19% for the critical path priority groups; 16% and 22% for the directory priority groups, which is the most affected priority type. The cache & directory priority performs better than the critical path priority for all situations. However, the difference between their performances is lowered as P(M) increases. The behavior of the remainder priority and combining types are the same for the higher values of P(W) with the middle values of P(W).

The cache & directory priority performs relatively better than the remainder priority types under HR traffic, differently from the UN traffic, because it distributes the congested traffic generated by the nature of HR. Owing to the priorities given to the extra messages between the directory and the caches, which does not directly affect the response time but increases the traffic of the network, the cache & directory priority performs better than the critical path priority, which only gives the priorities to the messages that directly impose on the response time. Since the nature of the increasing P(W) leads to a more congested traffic and the nature of increasing P(M) distributes the congestion of the traffic, the advantage of the cache & directory priority, explained above, increases as P(W) increases, whereas decreases as P(M) increases. The extra traffic generated under HR causes the performances of all combinations to be lower than UN. However, the system under HR is less affected from increase in P(W) than the system under UN.

The message priority algorithm is the most dominant factor on the performance of average network response time for the lower values of T/R, while the message combining flows through on the performance of average network response time for the higher T/R values regardless of P(M) and P(W) values under bit permutation traffic patterns, as the other traffic patterns, as is seen in Table 4.25 through Table 4.42. Four messages combining groups performs the best through all combining groups, while three messages combining follows it under both BR and PS. Besides, the lowest performance belongs to two messages combining.

As P(W) increases for the lower-to-middle values of P(M) under BR, all combining performances increase. The critical path priority groups have the highest performance over all priority groups under BR. The increase in P(W) regardless of P(M) values increases the cache priority groups performance, whereas decreases the performance of the remainder priority groups under BR.

For the lower values of P(W) under BR, the second highest performed priority is the directory priority, as is seen in Table 4.25 through Table 4.27. The cache & directory priority ensue the directory priority, while the cache priority has considerably lower performance (even quarter of the directory priority performance). The cache priority performance increases while the directory priority performance decreases as P(M) increases. Although, the performance of the cache & directory priority also decreases as P(M) increases, due to the higher decreasing rate of the directory priority, it counteracts the directory priority as P(M) increases. For this situation, the cache priority only goes fast enough to join the performance difference.

The cache & directory priority performance is higher than that of the directory priority performance for the middle values of P(W), as is seen in Table 4.28 through Table 4.30. Even though the performance difference among the priority groups for the middle values of P(W) is lower than the lower values of P(W), the directory priority still outperforms than the cache priority. The increase in P(M) for the middle values of P(W) increases the performances of all priority and combining groups. Although, the performance order of the priority groups for the higher values of P(W) is the cache & directory priority, the directory priority and the cache priority, respectively, the difference is quite a little. As is seen in Table 4.31 through

Table 4.33, the increase in P(M) for the higher values of P(W) also increases the performances of all priority groups. However, this effect is more apparent when P(M) value passes from 0.4 to 0.6 for the cache priority and the directory priority groups.

If a comparison between BR and the other traffic patterns for the lower values of P(W) is carried out, it can be easily said that both the performance of the critical path priority and the cache & directory priority groups is lower (even the one-half) under BR traffic than that of under UN. Besides, the same performance decreasing rate reaches up to 40% for the cache priority. Although, the performance of the directory priority under BR is lower than that of under UN for the lower values of P(M), the rate is not as much as the remainder traffic patterns. Moreover, its performance under BR gets higher for the middle values of P(M) with the higher combining types and for the higher values of P(M) with all combining types. If BR traffic is measured up against HR for the lower values of P(W), it can be observed that the critical path priority groups under BR performs worse for the lower values of P(M), whereas better for the higher values of P(M) than HR traffic. The performance of the cache & directory priority groups under BR is merely higher for the higher values of P(M) with three or four messages combining, while the cache priority groups solely shows better performance under BR when P(M) is 0.6 with four messages combining. However, the performance of the directory priority groups under BR is always higher than HR.

When the performance differences between BR and the other two traffic patterns is investigated for the middle values of P(W), it can be observed that although the critical path priority and the cache & directory priority groups under BR show lower performance than UN for the lower-to-middle values of P(M), both the decreasing rates are not such high as they are for the lower values of P(W) and even the performances draw ahead for the higher values of P(M). The directory priority performance is higher under BR traffic than the remainder two traffic patterns for all values of P(M) and P(W), whereas the cache priority under BR only yields better performance than UN for the higher values of P(M) with middle-to-higher number of combined messages. The critical path priority under BR yields better performance

for all P(M) and P(W) values; the cache & directory priority performs better for the lower-to-middle values of P(M) and the cache priority has higher performance for the higher values of P(M) than HR. Ultimately, all priority groups yields higher performance under BR traffic than UN and HR for the higher values of P(W), even the ratio achieves to 100% for the directory priority.

The results show that the directory priority groups have their highest performance under BR. Although, the other priority groups under BR do not perform as high as other traffic patterns for the lower-to-middle values of P(W), it can be clearly said that their performance forge ahead under BR as P(W) increases and eventually their performances become the highest under BR. As is explained before, the traffic with a regular intensity around the home directory nodes that is generated under BR stirs up the directory priority and the combining algorithms to perform better. Since this traffic gets more congested as P(W) increases, the improving rates under BR are higher than UN. Besides, the system performance under BR increases as the combining number of messages and P(M) increases.

Finally, if the combination of the priority and the combining algorithms performances under PS traffic is investigated, it can clearly be said that the combining performances increases as P(W) increases. The critical path priority is the best performed priority, while the cache & directory priority stands a close second. Although, the directory priority yields better performance than the cache priority for lower-to-middle values of P(W), its performance become worse for the higher values of P(W). The increase in P(M) increases the critical path priority and the cache & directory priority performances. However, the increasing rate in the critical path priority is more than that in the the cache & directory priority.

When the performance differences between PS and the other traffic patterns for the lower values of P(W) are investigated, it can explicitly be observed that the directory priority under PS traffic has the highest performance over all traffic patterns, while the critical path, the cache and the cache & directory priority have better performance under PS than either HR or BR for all situations. Although the cache priority performs worse under PS than UN for all situations, the critical path and the cache & directory performs better for the higher values of P(M).

The performance of the entire combining and priority combinations under PS is higher than HR for the middle-to-higher values of P(W) and higher than UN for the higher values of P(W). The critical path and the cache & directory priority always performs better under PS than BR for the middle-to-higher values of P(W). However, for the higher values of P(W), the directory priority yields higher performance under BR than PS.

The results show that when the priority and the combining algorithm are implemented on the base system together, the obtaining performance is higher than the individual algorithm performances. The directory priority performs better under bit permutation traffic models, while the critical path priority usually performs the best through the entire priority algorithms. The more number of messages combining leads to the more performance increase for all traffic patterns regardless of the other DSM parameters. Bit permutation traffic models get an edge over the other traffic patterns as P(W) increases for the entire combination types. Although the more congested traffic causes the more performance increase of the algorithms, a regular intensity of the created congested traffic is a must for this advantage.

5. CONCLUSION

DSM allows ease of programming in large-scale parallel computer systems by hiding the message passing mechanism. However, mapping the shared logical address space on local memories and keeping it consistent requires significant traffic on the interconnection network. Improving the performance of these systems and reducing remote memory access latency have been attracting the researchers' attention over a decade.

There is an extensive research about reducing the latency of shared memory systems including improving the underlying interconnection network architecture, optimizing the cache coherence traffic and conducting architectural innovations.

With the high cost of remote memory access, DSM systems achieve higher performance by giving priority or combining the servicing of multiple memory or communication operations from a processor. Both message priority and message combining techniques have been extensively used in literature. Priorities are given according to bandwidth, rate, deadline, laxity, criticality and length of the messages. Combining technique is mostly conducted on switch-based architectures and rarely implemented on 1D broadcast-based architectures. Besides, the studies that use combining and priority techniques together are narrow-scoped, i.e. combining is implemented in the form of multicasting and priorities are given solely by the length of the messages. Therefore, there is a lack of research about using different kinds of message combining and priority algorithms together on a 2D broadcast-based optical architecture supporting DSM and conducting a detailed evaluation on the system for observing its performance.

This thesis includes, via simulation, modeling of 2D DSM on the SOME-Bus multiprocessor architecture with one server model and evaluating the performance improvement of the system due to four types of message priority, including cache, directory, cache with directory and critical path priority, and three types of combining algorithms, including two, three and four messages combining, under different traffic patterns, such as UN, HR, BR and PS and for varying values of important DSM parameters such as T/R, P(M) and P(W). Messages are prioritized in

the input queues and combined in the output channels regardless of their destinations. Both for priority and combining operations, memory consistency model and cache coherence are preserved.

The message priority and combining techniques are implemented on the base system both individually and jointly for observing their performance in detail. The system is simulated by using OPNET Modeler with reply forwarding protocol and a relaxed memory consistency model, which allows both read and write operations to cut in line previous reads and writes without violating cache coherence. The simulated system contains 8×8 nodes. Each node has a server that processes directory, cache and processor messages, two output channels and two input queues, one for horizontal and the other for vertical dimension. Scenarios are run to measure average network response time, average input waiting time and average channel waiting time. Average network response time is the time interval between the instant when a cache miss causes a message to be enqueued in the output channel until the instant when the corresponding data or acknowledge message arrives at the input queue. Average input waiting time is the time interval between a message is enqueued in the input queue of a node until it gets serviced while average channel waiting time is the time interval between the message is enqueued in the output channel queue until it gets serviced. Wherefore the underlying architecture is 2D, average channel waiting time is calculated as the average of average horizontal and average vertical channel waiting times.

The results reveal that the message priority algorithm is the most dominant factor on the performance of average network response time for the lower values of T/R, while the message combining flows through on the performance of average network response time for the higher T/R values. The combining reduces average channel waiting time between 13% and 29% on average, while the priority reduces average input waiting time between 7% and 43% on average. Four messages combining has the highest performance on average network response time over all messages combining, while the critical path priority performs the best over the entire priority types in most cases. With the application of message priority and combining together, the obtaining performance is higher than the individual algorithm

performances and between 7% and 40% performance increase for average network response time has gained.

The directory priority performs better under bit permutation traffic models, while the critical path priority usually performs the best through the entire priority algorithms. The more number of messages combining leads to the more performance increase for all traffic patterns regardless of the other DSM parameters. Bit permutation traffic models get an edge over the other traffic patterns as P(W) increases for the entire combination types.

REFERENCES

ABASIKELEŞ, İ., AKAY, M. F., 2010. Performance evaluation of directory protocols on an optical broadcast-based distributed shared memory multiprocessor. Computers & electrical engineering, 36(1):114-131.

ACACIO, M.E., GONZÁLEZ, J., GARCÍA, J.M., DUATO, J., 2002a. Owner prediction for accelerating cache-to-cache transfer misses in a cc-NUMA architecture. Proc. Conference on High Performance Networking and Computing, 1-12.

ACACIO, M.E., GONZÁLEZ, J., GARCÍA, J.M. AND DUATO, J., 2002b. The use of prediction for accelerating upgrade misses in cc-NUMA multiprocessors. Proc 11th International Conference on Parallel Architectures and Compilation Techniques (PACT'02), 155.

ALAM, S.R., BARRETT, R.F., FAHEY, M.R., KUEHN, J.A., MESSER, O.E.B., MILLS, R.T., ROTH, P.C., VETTER, J.S. and WORLEY, P.H., 2008. An Evaluation of the Oak Ridge National Laboratory Cray XT3. International Journal of High Performance Computing Applications, 22:52-80.

ALAMELDEEN, A.R., MAUER, C.J., XU, M., HARPER, P.J., MARTIN, M.M., SORIN, D.J. et al., 2002. Evaluating Non-deterministic Multi-threaded Commercial Workloads. Proceedings of the Fifth Workshop on Computer Architecture Evaluation Using Commercial Workloads.

ALHUSSIEN, A., WANG, C., and BAGHERZADEH, N., 2011. Design and evaluation of a high throughput robust router for network-on-chip. Published in IET Computers & Digital Techniques, 6(3):173-179

ARTUNDO, I., DESMET, L., HEIRMAN, W., DEBAES, C., DAMBRE, J., Van CAMPENHOUT, J.M., THIENPONT, H., 2006. IEEE Journal Of Selected Topics In Quantum Electronics, 12(4): 828 – 837.

BARROSO, L. A., DEAN, J., and HÖLZLE, U., 2003. Web search for a planet: The Google cluster architecture. IEEE Micro, 23(2):22-28.

BARROSO, L.A., GHARACHORLOO, K., MCNAMARA, R., NOWATZYK, A., QADEER, S., SANO, B., SMITH, S., STETS, R., and VERGHESE, B., 2000. Piranha: a scalable architecture based on single-chip multiprocessing. In Proceedings of the 27th annual international symposium on Computer architecture (ISCA '00), 282-29.

BEIGNÉ, E., CLERMIDY, F., VIVET, P., CLOUARD, A., RENAUDIN, M. ,2005. An Asynchronous NOC Architecture Providing Low Latency Service and its Multi-level Design Framework. Asynchronous Circuits and Systems, ASYNC 2005, Proceedings, 11th IEEE International Symposium on, 54-63.

BELL, S., EDWARDS, B., AMANN, J., CONLIN, R., JOYCE, K., LEUNG, V., MACKAY, J., REIF, M., BAO, L., BROWN, J., et al., 2008. TILE64 processor: A 64-core SoC with mesh interconnect. In Proceedings of the International Solid-State Circuits Conference, 88-598.

BLUMRICH, M., CHEN, D., COTEUS, P., GARA, A., GIAMPAPA, M., HEIDELBERGER, P., SINGH, S., STEINMACHER-BUROW, B., et al., 2003. Design and analysis of the bluegene/L torus interconnection network. IBM Research Report RC23025 (W0312-022).

BOLOTIN, E., GUZ, Z., CIDON, I., GINOSAR, R., and KOLODNY, A., 2007. The Power of Priority: NoC Based Distributed Cache Coherency. In International Symposium on Networks-on-Chip (NOCS'07), 117-126.

BORKAR, S., 2007. Thousand core chips: a technology perspective. In: Proceedings of the Design Automation Conference, 746– 749.

BOUDOL, G., and PETRI, G., 2009. Relaxed memory models: an operational approach. Proceedings of the 36th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages, 392-403.

BURCKHARDT, S., ALUR, R., and MARTİN, M.MK., 2007. Checkfence: Checking Consistency of Concurrent Data Types on Relaxed Memory Models. Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation,12-21.

BURNS, J., and GAUDIOT, J.-L., 2005. Area and System Clock Effects on SMT/CMP Throughput. IEEE Trans. Computers, 54(2):141-152.

CARARA, E., CALAZANS, N., MORAES, F., 2009. Managing QoS flows at task level in NoC-based MPSoCs. 17th IFIP International Conference on Very Large Scale Integration (VLSI-SoC), 133-138.

CARTER, J., KUO, C.C., KURAMKOTE, R., and SWANSON, M., 1998. Design Alternatives for Shared Memory Multiprocessors. 5th International Conference On High Performance Computing, 41-50.

CAMARA, J.M., MORETÓ, M., VALLEJO, E., BEIVIDE, R., MIGUEL-ALONSO, J., MARTINEZ, C., NAVARIDAS, J., 2007. Mixed-radix Twisted Torus Interconnection Networks. IEEE International Parallel and Distributed Processing Symposium, IPDPS 2007, 1-10.

CHAN, J., HENDRY, G., BERGMAN, K., and CARLONI, L.P., 2011. Physical-Layer Modeling and System-Level Design of Chip-Scale Photonic Interconnection Networks. IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, 30(10): 1507 - 1520.

CHANDRAMOULI, B., IYER, S., 2004. A performance study of snoopy and directory based cache-coherence protocols. Department Of Computer Science, Duke University.

CHAUDHURI, M., HEINRICH, M., HOLT, C., SINGH, J.P., ROTHBERG, E., and HENNESSY, J., 2003. Latency, Occupancy, and Bandwidth in DSM Multiprocessors: A Performance Evaluation. IEEE Transactions On Computers, 52(7):862-880.

CHAVES, T.M., CARARA, E.A., and MORAES, F.G., 2011. Energy-efficient Cache Coherence Protocol for NoC-based MPSoCs. SBCCI '11 Proceedings of the 24th symposium on Integrated circuits and systems design, 215-220.

CHEN, T., and WANG, N., 2005. Optimal broadcasting on incomplete star graph interconnection networks. Journal of Systems Architecture, 51:143– 150.

CHEN, T., TSENG, Y., and SHEU, J., 1996. Balanced spanning trees in complete and incomplete star graphs. Transactions on Parallel and Distributed Systems, 7(7): 717– 723.

CHEN, X., LU, Z., JANTSCH, A., CHEN S., and GU, H., 2013. Reducing Virtual-to-Physical address translation overhead in Distributed Shared Memory based multi-core Network-on-Chips according to data property. Journal Computers and Electrical Engineering archive, 39(2):596-612.

CHENNAREDDY, V., and DEKA, J.K., 2006. Formally Verifying the Distributed Shared Memory Weak Consistency Models. International Conference on Advanced Computing and Communications, 455-460.

CHENG, L., MURALIMANOHAR, N., RAMANI, K., BALASUBRAMONIAN, R., and CARTER, J.B., 2006. Interconnect-Aware Coherence Protocols for Chip Multiprocessors. ISCA '06 Proceedings of the 33rd annual international symposium on Computer Architecture, 339-351.

CHISHTI, Z., POWELL, M.D., and VIJAYKUMAR, T.N., 2005. Optimizing Replication, Communication, and Capacity Allocation in CMPs. 32nd International Symposium on Computer Architecture ISCA '05, 357-368.

CHUNG, K. L., 1995. Prefix Computations on a Generalized Mesh-Connected Computer with Multiple Buses. IEEE Trans. Parallel and Distributed Systems, 6(2):196-199.

CONWAY, P., and HUGHES, B., 2007. The AMD Opteron Northbridge architecture, present and future. IEEE Micro, 27(2):10-21.

COTA, E., KREUTZ, M., ZEFERINO, C.A., CARRO, L., LUBASZEWSKI, M., SUSIN, A., 2003. VLSI Test Symposium, Proceedings 21st, 128-133.

CUESTA, B., ROBLES, A., and DUATO, J., 2012. Switch-based packing technique to reduce traffic and latency in token coherence. Journal of Parallel and Distributed Computing, 72(3):409-423.

CUESTA, B. ROS, A., GÓMEZ, M.E., ROBLES, A., and DUATO, J., 2013. Increasing the Effectiveness of Directory Caches by Avoiding the Tracking of Noncoherent Memory Blocks. IEEE Transactions On Computers, 62(3):482-495.

CULLER, D.E., SINGH, J.P., GUPTA, A., 1999. Parallel Computer Architecture: A Hardware/Software Approach. Morgan Kaufmann Publishers, San Fransisco, USA, 880p.

DAI, D., and PANDA, D.K., 1996. Reducing Cache Invalidation Overheads in Wormhole Routed DSMs Using Multidestination Message Passing. Proceedings of the 1996 International Conference on Parallel Processing, Vol.3. Software, 138– 145.

DALLY, W.J., and TOWLES, B., 2004. Principles and Practices on Interconnection Networks. Morgan Kaufmann, 550p.

DESHPANDE, S., RAVALE, P., AND APTE, S., 2010. Cache coherence in centralized shared memory and distributed shared memory architectures. International Journal on Computer Science and Engineering (IJCSE), 39-44.

DIGHE, O. M., VAIDYANATHAN, R., ZHENG, S. Q., 1996. The Bus-Connected Ringed Tree: A Versatile Interconnection Network. J. Parallel and Distributed Computing, 33:189-196.

DOUD, B., 2011. The TILE-Gx Processor: Enabling HPC through Massive-Scale Manycore. http://www.ll.mit.edu/HPEC/agendas/proc11/Day2/Posters/B-9_Doud.pdf

DVORAK, V., 2008. Communication Performance of Mesh- and Ring-Based NoCs. Networking, ICN 2008, Seventh International Conference on, 156-161.

DVORAK, V., and JAROS, J., 2010. Optimizing Collective Communications on 2D-Mesh and Fat Tree NoC. 2010 Ninth International Conference on Networks, 22 – 27.

EISLEY, N., PEH, L.S. and SHANG, L., 2006. In-Network Cache Coherence. IEEE Computer Architecture Letters, 5(1):34-37.

FLORES, A., ACACIO, M.E., and ARAGÓN, J.L., 2010. Exploiting address compression and heterogeneous interconnects for efficient message management in tiled CMPs. Journal of Systems Architecture, 56(9):429–441.

GELADO, I., CABEZAS, J., NAVARRO, N., STONE, J.E., PATEL, S., and HWU, W.W., 2010. An Asymmetric Distributed Shared Memory Model for Heterogeneous Parallel Systems. Proceedings of the 15th edition of ASPLOS on Architectural support for programming languages and operating systems, 347-358.

GRATZ, P. V., and KECKLER, S. W., 2010. Realistic workload characterization and analysis for networks-on-chip design. In The 4th Workshop on Chip Multiprocessor Memory Systems and Interconnects (CMP-MSI).

GRATZ, P., GROT, B., KECKLER, S. W., 2008. Regional congestion awareness for load balance in networks-on-chip. In: IEEE 14th International Symposium on High Performance Computer Architecture, HPCA 2008, 203-214.

GRECU, C., ANGHEL, L., PANDE, P.P., IVANOV, A., SALEH, R., 2007. Essential Fault-Tolerance Metrics for NoC Infrastructures. On-Line Testing Symposium, IOLTS 07, 13th IEEE International. 37-42.

HANAWA, T., FUJIWARA, T., and AMANO, H., 1996. Hot spot contention and message combining in the Simple Serial Synchronized Multistage Interconnection Network. Eighth IEEE Symposium on Parallel and Distributed Processing, 298- 305.

HAURYLAU, M., CHEN , G., CHEN , H., ZHANG , J., NELSON, N. A., ALBONESI, D. H., FRIEDMAN, E. G., FAUCHET, P. M., 2006. On-chip optical interconnect roadmap: Challenges and critical directions. IEEE Journal of Selected Topics in Quantum Electronics, 12(6): 1699 – 1705.

HEIRMAN, W., DAMBRE, J., VAN CAMPENHOUT, J., 2007. Synthetic Traffic Generation as a Tool for Dynamic Interconnect Evaluation. ACM Press, 65-72.

HENNESSY, J. L., and PATTERSON, D. A., 2002. Computer Architecture: A Quantitative Approach. 3rd ed. San Francisco, CA: Morgan Kaufmann.

HOROWITZ, M., DALLY, W., 2004. How scaling will change processor architecture. In: Proceedings of the International Solid-State Circuits Conference (ISSCC' 04), Digest of Technical Papers, 132-133.

HU, S. and JOHN, L., 2006. Avoiding store misses to fully modified cache blocks. Proc. 25th IEEE Int. Performance, Computing, and Communications Conference (IPCCC' 2006), 286-296.

HUNG, S.S., and LIU, D.S.M., 2007. Efficient Reduction of Access Latency through Object Correlations in Virtual Environments. EURASIP Journal on Advances in Signal Processing, 1(1):178-178.

INTEL., 2008. Intel Tera-scale Computing Research Program. http://www.intel.com/content/www/us/en/research/intel-labs-tera-scale-research-paper.html.

IZU, C., MIGUEL-ALONSO, J., GREGORIO, J.A., 2005a. Evaluation of interconnection network performance under heavy non-uniform loads. Lecture Notes in Computer Science, ICA3PP 2005, 3719:396– 405.

JERGER, N. E., PEH, L.-S., and LIPASTI, M. H., 2008. Virtual Circuit Tree Multicasting: A Case for On-Chip Hardware Multicast Support. 35th International Symposium on Computer Architecture, 229– 240.

JIANG, H., CHAUDHARY, V., 2002. MigThread: Thread Migration in DSM Systems., International Conference on Parallel Processing Workshops, 581-588.

JIME´NEZ, E., FERNA´NDEZ, A., and CHOLVI, V., 2008. A parametrized algorithm that implements sequential, causal, and cache memory consistencies. The Journal of Systems and Software, 81(1):120-13.

KANTER, D., 2007. The common system interface: Intel' s future interconnect. http://www.realworldtech.com/page.cfm? ArticleID=RWT082807020032

KARLSSON, S., and BRORSSON, M., 2003. Priority Based Messaging for Software Distributed Shared Memory. Journal Cluster Computing, 161-169.

KATSINIS, C., 2008. A comparison of Broadcast-based and Switch-based Networks of Workstations.

KATSINIS, C., 2004. Merging, sorting and matrix operations on the SOME-Bus multiprocessor architecture. Future Generation Computer Systems, 20:643–661.

KATSINIS, C., 2003. Models of Distributed-shared-memory on An Interconnection Network for Broadcast Communication. Journal of Interconnection Networks, 4(1):77-102.

KATSINIS, C., 2001. Performance analysis of the simultaneous optical multiprocessor exchange bus, Parallel Computing, 27(8):1079– 1115.

KATSINIS, C., and HECHT, D., 2004. Fault-tolerant DSM on the SOME-Bus Multiprocessor Architecture with Message Combining. 18th International Parallel and Distributed Processing Symposium,12:210.

KATSINIS, C., and HECHT, D., 2000. Hot-Spots and Message Combining in An Interconnection Network for Broadcast Communication. The 2000 Int. Conf. on Parallel and Distributed Processing Techniques and Applications.

KATSINIS, C., and HECHT, D., 2003. Some-bus-now: A network of workstations with broadcast. in:Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications.

KATSINIS, C., and NABET, B., 2004. A scalable interconnection network architecture for petaflops computing, the Journal of Supercomputing. 27:103-128.

KIM, H., YOUN, S., and KIM, J., 2009. Reusability-aware cache memory sharing for chip multiprocessors with private L2 caches. Journal of Systems Architecture, 55 (10-12):446– 456.

KISTLER, M., PERRONE, M., and PETRINI, F., 2006. Cell multiprocessor communication network: Built for speed. IEEE Micro, 26(3):10-23.

KODI, A.K., and LOURI A., 2004. Design of a High-Speed Optical Interconnect for Scalable Shared Memory Multiprocessors, Proceedings on 12th Annual IEEE Symposium on High Performance Interconnects, 92-97.

KREWELL, K., 2005. Best servers of 2004: Where multicore is the norm. Microprocessor Rep., www.mdronline.com/mpr/h/2005/0118/190301.html

LAUDON, J., LENOSKI, D., 1997. The SGI Origin: A ccNUMA Highly Scalable Server. Proc. 24th Annual lnternational Symposium on Cornputer Architecture, 241-251.

LEE, C.K., CHOI, J.H., PARK, K.H. and KIM, B.W., 2000. Fast & Cost Effective Cache Invalidation in DSM. Proc. of the 7th International Conference on Parallel and Distributed Systems, 492.

LI, J.-P., and MUTKA, M. W., 1996. Real-time virtual channel flow control. Journal of Parallel and Distributed Computing, 32(1):49– 65.

LIU, J., and LIN, X., 2004. Equalization in high-speed communication systems. IEEE Circuits Syst. Mag., 4(2):4-17.

LIU, O., 2007. AMD technology: power, performance and the future. 3rd Workshop on High performance computing in China (ATIP' s), 89-94.

LODDE, M., FLICH, J., and ACACIO, M.E., 2012. Heterogeneous NoC Design for Efficient Broadcast-based Coherence Protocol Support. 2012 Sixth IEEE/ACM International Symposium on Networks-on-Chip, 59-66.

LOH, G., 2008. 3D-stacked memory architectures for multi-core processors. In: Proceedings of the International Symposium on Computer Architecture (ISCA' 08), 453– 464.

LOURI, A. ,WEECH, B., 1998. A Spanning Multichannel Linked Hypercube: A Gradually Scalable Optical Interconnection Network for Massively Parallel Computing. IEEE Trans.on Parallel and Distributed Systems, 9(5): 497-512.

LU, Z., JANTSCH, A., SANDER, I., 2005. Feasibility Analysis of Messages for On-chip Networks Using Wormhole Routing. Design Automation Conference, 2: 960 – 964

MA, S., JERGER, N. E., and WANG, Z., 2012. Supporting Efficient Collective Communication in NoCs. In IEEE 18th International Symposium on High Performance Computer Architecture (HPCA), 165-176.

MAGEN, N., KOLODNY, A., WEISER, U., and SHAMIR, N., 2004. Interconnect-power dissipation in a microprocessor. in Proc. Int. Workshop SLIP, 7– 13.

MANOLACHE, S., ELES, P., PENG, Z., 2005. Fault and energy-aware communication mapping with guaranteed latency for applications implemented on NoC. Design Automation Conference, 266-269.

MARINISSEN, E., PRINCE, B., KELTEL-SCHULZ, D., and ZORIAN, Y., Challenges in embedded memory design and test. In: Proceedings of the conference on Design, Automation and Test in Europe, 722-727.

MARTIN, M.M.K., HARPER, P.J., SORIN, D.J., HILL, M.D., WOOD, D.A., 2003a. Using destination-Set Prediction to Improve the Latency/Bandwidth Tradeoff in Shared Memory Multiprocessors. 30th Annual International Symposium on Computer Architecture (ISCA 2003), 206-217.

MARTIN, M.M.K., HILL, M.D., and WOOD, D.A., 2003b. Token Coherence: A New Framework for Shared-Memory Multiprocessors. IEEE Micro, 23(6):108-116.

MIGUEL-ALONSO, J., IZU, C., GREGORIO, J.A., 2008. Improving the Performance of Large Interconnection Networks using Congestion-Control Mechanisms. Performance Evaluation, 65(3– 4):203– 211.

MILLER, D.A.B., 2000. Rationale and challenges for optical interconnects to electronic chips. Proc IEEE, 88:728-749.

MILLER, D., 2009. Device requirements for optical interconnects to silicon chips. Proc. IEEE, 97(7):1166– 1185.

MUDGE, T., 2001. Power: A first-class architectural design constraint. IEEE Computer, 34(4):52-58.

MUKHERJEE, S.S., HILL, M.D., 1998. Using Prediction to Accelerate Coherence Protocols. Proc. International Symposium on Computer Architecture, 179-190.

NAEEM, A., CHEN, X., LU, Z., and JANTSCH, A., 2011. Realization and Performance Comparison of Sequential and Weak Memory Consistency Models in Network-on-Chip based Multi-core Systems. 16th Asia and South Pacific, Design Automation Conference (ASP-DAC), 154-159.

NAVARIDAS, J., MIGUEL-ALONSO, J., 2009. Realistic Evaluation of Interconnection Networks Using Synthetic Traffic. Eighth International Symposium on Parallel and Distributed Computing, ISPDC '09, 249 – 252.

NIKOLOV, A.V., and LERATO, L., 2010. Comparison of the Performance of Two Service Disciplines for a Shared Bus Multiprocessor with Private Caches. IJCSI International Journal of Computer Science Issues, 7(2).

OHASHI, K., NISHI, K., SHIMIZU, T., NAKADA, M., FUJIKATA, J., USHIDA, J., TORII, S., NOSE, K., MIZUNO, M., YUKAWA, H. et al., 2009. On-Chip Optical Interconnect. Proceedings of the IEEE, 97(7).

OLUKOTUN, K., and HAMMOND, L., 2005. The Future of Microprocessors. ACM Queue, 3(7):26-29.

OPNET Technologies, Inc., 2013. OPNET Modeler. OPNET University program, http://www.opnet.com/university_program.

PALIWAL, K. K., JANYANI, V., GAUR, M. S., and LAXMI, V., 2009. Impact of Faulty Links on Quality-of-Service in Network-on-Chip under Different Traffic Patterns. IJCSNS, 9(3):108.

PAN, Y., ZHENG, S.Q., LI,K., SHEN, H., 2001. An improved generalization of mesh-connected computers with multiple buses. IEEE Trans. on Parallel and Distributed Systems, 12(3):293-305.

PANG, J., FOKKINK, W., HOFMANB, R., and VELDEMA R., 2007. Model checking a cache coherence protocol of a Java DSM implementation. The Journal of Logic and Algebraic Programming, 71(1):1-43.

PASCUAL, R.F., GARCI´A, J.M., ACACIO, M.E., and DUATO, J., 2008. Extending the TokenCMP Cache Coherence Protocol for Low Overhead Fault Tolerance in CMP Architectures. IEEE Transactions on Parallel and Distributed Systems, 19(8): 1044-1056.

PHILLIPS, S., 2007. Victoriafalls: scaling highly-threaded processor cores, 19th Annual IEEE Symposium on High Performance Chips (Hot Chips).

PROTIC, J., TOMASEVIC, M., and MILUTINOVIC, V., 1996. Distributed shared memory: concepts and systems. Parallel & Distributed Technology: Systems & Applications, 4(2):63-71.

RAMESH, B., and VARADARAJAN, S., 2007. User-level DSM System for Modern High-Performance Interconnection Networks. CoRR Tech. Rep. abs/cs/0703112.

RATTNER, J., 2013. Single-chip cloud computer: An experimental many-core processor from intel labs. http://download.intel.com/pressroom/pdf/rockcreek/SCC_Announcement_JustinRattner.pdf

ROS, A., ACACIO, M.E., and GARCI´A, J.M., 2010. A Direct Coherence Protocol for Many-Core Chip Multiprocessors. IEEE Transactions on Parallel and Distributed Systems, 21(12):1779-1792.

RUTGERS, J.H., BEKOOIJ, M.J.G., and SMIT, G.J.M., 2012. Evaluation of a Connectionless NoC for a Real-Time Distributed Shared Memory Many-Core System. 15th Euromicro Conference on Digital System Design (DSD), 727-730.

SANCHEZ, D., MICHELOGIANNAKIS, G., and KOZYRAKIS, C., 2010. An Analysis of On-Chip Interconnection Networks for Large-Scale Chip Multiprocessors. ACM Transactions on Architecture and Code Optimization, 7(1).

SHACHAM, A., and BERGMAN, K., 2009. An Experimental Validation of a Wavelength-Striped,Packet Switched. Journal Of Lightwave Technology, 27(7): 841 - 850.

SHACHAM, A., and BERGMAN, K., 2007. Building Ultralow-Latency Interconnection Networks Using Photonic Integration. IEEE Micro, 27(4):6– 20.

SHACHAM, A., BERGMAN, K., and CARLONI, L.P., 2007. On the design of a photonic network-on-chip. Int.Symp. Networks-on-Chips, 53 – 64.

SHAFER, S.R., and GHOSE, K., 1998. Parallel program execution time with message consolidation. United States Patent, Patent 5781787

SHIN, J. and PINKSTON, T.M., 2003. The Performance of Routing Algorithms under Bursty Traffic Loads. Proc. Int'l Conf. Parallel and Distributed Processing Techniques and Applications (PDPTA '03), 737-743.

SINHAROY, B., 2009. POWER7 multi-core processor design. 42nd Annual IEEE/ACM International Symposium on Microarchitecture,1.

SOMOGYI, S., WENISCH, T. F., HARDAVELLAS, N., KIM, J., AILAMAKI, A., and FALSAFI, B., 2004. Memory Coherence Activity Prediction in Commercial Workloads. Proceedings of the 3rd workshop on Memory performance issues: in conjunction with the 31st international symposium on computer architecture, 37-45.

SONG, H., KWON, B., and YOON, H., 1999. Throttle and preempt: a flow control policy for real-time traffic in wormhole networks. Journal of Systems Architecture, 45(8).

SOURAVLASA, S., and ROUMELIOTISB, M., 2008. A message combining approach for efficient array redistribution in non-all-to-all communication networks. International Journal of Computer Mathematics. 85(11):1609–1619.

SUGIMORI, Y., KOHAMA, Y., SAITO, M., YOSHIDA, Y., MIURA, N., ISHIKURO, H., SAKURAI, T., and KURODA, T., 2009. A 2Gb/s 15pJ/b/chip inductive-coupling programmable bus for NAND flash memory stacking. In: Proceedings of the International Solid-State Circuits Conference (ISSCC' 09), Digest of Technical Papers, 244-245.

STRAUSS, K., SHEN, X., and TORRELLAS, J., 2007. Uncorq: Unconstrained snoop request delivery in embedded-ring multiprocessors. 40th Annual IEEE/ACM International Symposium on Microarchitecture, 327-342.

SZYMANSKI, T., 1995. Hypermeshes Optical Interconnection Network for Parallel Computing. Journal Parallel Distributed Computing, 26(1).

TAYLOR, M.G., MIDWINTER, J.E., 1990. Multistage interconnection networks using optical interconnects. IEE Colloquium on Optical Connection and Switching Networks for Communication and Computing, 3(1):3-4.

TONCEV, M., TOMASEVIC, M., DJORDJEVI, J., and ALEKSIC, M., 2004. Improving performance of a DSM system by the communication controller optimizations. Electrical and Computer Engineering, 1035-1038.

TREMBLAY, M., and CHAUDHRY, S., 2008. A third-generation 65nm 16-core 32-thread plus 32-scoutthread CMT SPARC processor. In Proceedings of the International Solid-State Circuits Conference, IEEE, Los Alamitos, CA.

TSAI, H., HORNG, S., TSAI, S., KAO, T., LEE, S., 1997. Solving an algebraic path problem and some related graph problems on a hyper-bus broadcast network. IEEE Trans. on Parallel and Distributed Systems, 8(12): 1226 - 1235.

TSENG, Y., 1997. Multi-node broadcasting in hypercubes and star gaphs. in: Proceedings of the Third International Conference on Algorithms and Architectures for Parallel Processing, 51– 58.

TSENG, Y., WANG, S., HO, C., 1999. Efficient Broadcasting in Wormhole-Routed Multicomputers: A Network-Partitioning Approach. IEEE Transactions on Parallel and Distributed Systems, 10(1):44 – 61.

VANGAL, S., HOWARD, J., RUHL, G., DIGHE, S., WILSON, H., TSCHANZ, J., FINAN, D., IYER, P., SINGH, A., JACOB., T, et al., 2007. An 80-tile 1.28 tflops network-on-chip in 65nm CMOS. In: Proceedings of the International Solid-State Circuits Conference (ISSCC' 07), Digest of Technical Papers, 98-589.

VANTREASE, D., LIPASTI, M.H., and BINKERT, N., 2011. Atomic Coherence: Leveraging Nanophotonics to Build Race-Free Cache Coherence Protocols. 17th International Symposium on High Performance Computer Architecture (HPCA), 132-143.

VANTREASE, D., SCHREIBER, R., MONCHIERO, M., MCLAREN, M., FIORENTINO, M. et al., 2008. Corona: System Implications of Emerging Nanophotonic Technology. Proceedings of the 35th Annual International Symposium on Computer Architecture, 153-164.

XUE, C. J., HU, J., SHAO, Z., and SHA, E., 2010. Iterational retiming with partitioning: Loop scheduling with complete memory latency hiding. ACM Transactions on Embedded Computing Systems (TECS) TECS Homepage archive, 9(3).

XUE, J., GARG, A., CIFTCIOGLU, B., WANG, S., HU, J. et al., 2010. An Intra-Chip Free-Space Optical Interconnect. Proceedings of the 37th annual international symposium on Computer architecture, 94-105.

YAHYA, A.A., and BADER, R.M.I., 2007. Distributed Shared Memory Consistency Object-based Model. Journal of Computer Science, 3(1):57-61.

YE, T.T., BENINI, L. and MICHELI, G.D., 2004. Packetization and routing analysis of on-chip multiprocessor networks. Journal of Systems Architecture, 50:81– 104.

YI, K., RO, W.W., and GAUDIOT, J.-L., 2013. Importance of Coherence Protocols with Network Applications on Multicore Processors. IEEE Transactions On Computers, 62(1):6-15.

BIOGRAPHY

She was born in Adana, in 1984. She has completed her elementary education at Celalettin Seyhan Primary Education School. She went to ÇEAŞ Seyhan Anatolian High School for her high school education and graduated from there with 1st degree in 1999. Then she deserved to educate in Adana Science School. She has completed her university education at the department of Computer Engineering of Istanbul Technical University (ITU) with a degree of *magna cum laude* in 2007. She has received her MSc in Computer Engineering from Çukurova University in 2009.

She currently has a book, 3 SCI journal papers, 2 national and 4 international conference papers.

She worked as an expert computer engineer at Presidency of Çukurova University in Adana from 2007 through 2012. Since 2012, she has been working as a Research Assistant at Adana Science and Technology University.

Her interest areas are parallel processing, evolutionary computation, simulation and modeling of multiprocessors with distributed shared memory.

APPENDICES

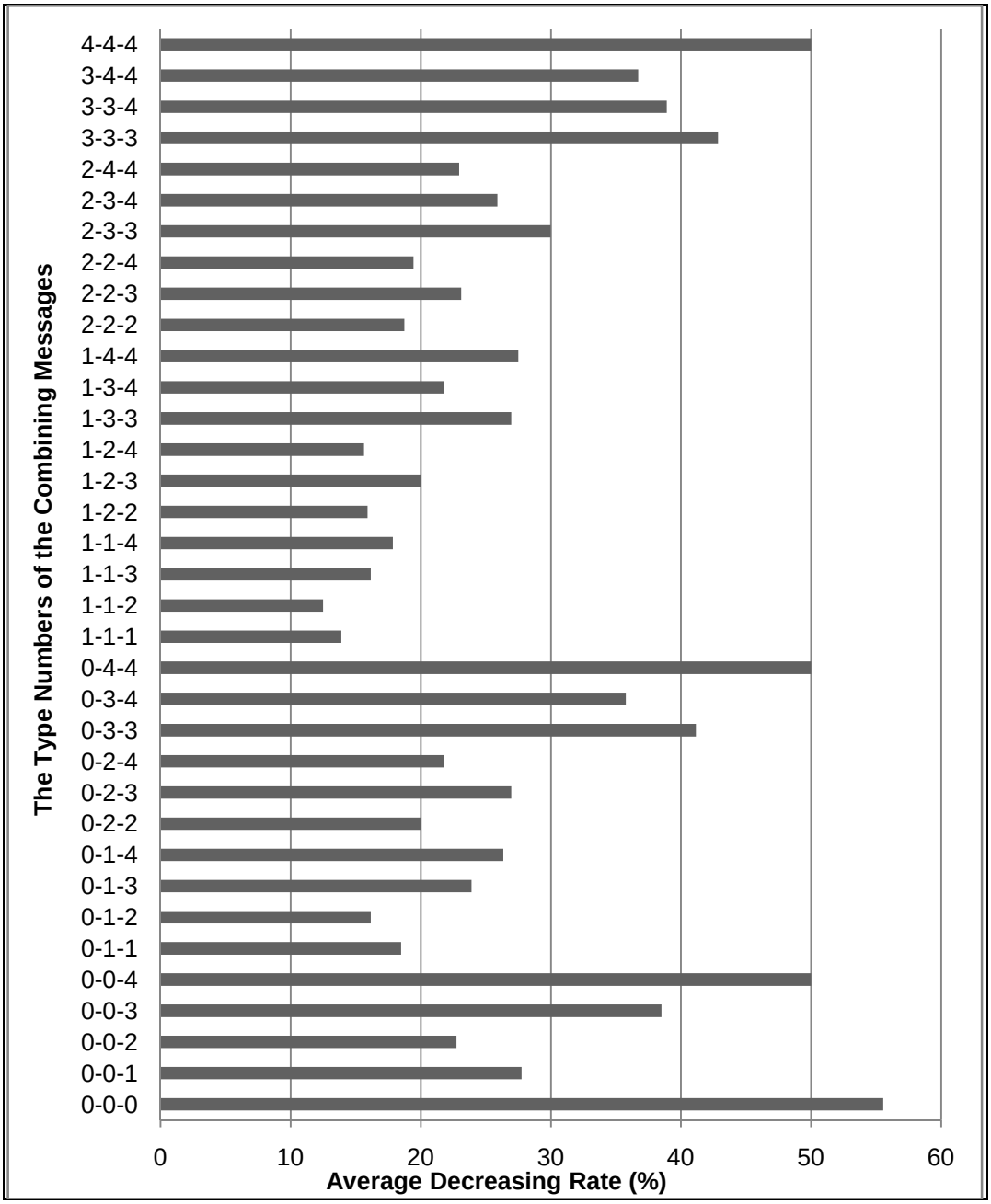


Figure 3.5. Average of the decreasing rates according to the combining messages types for three messages combined

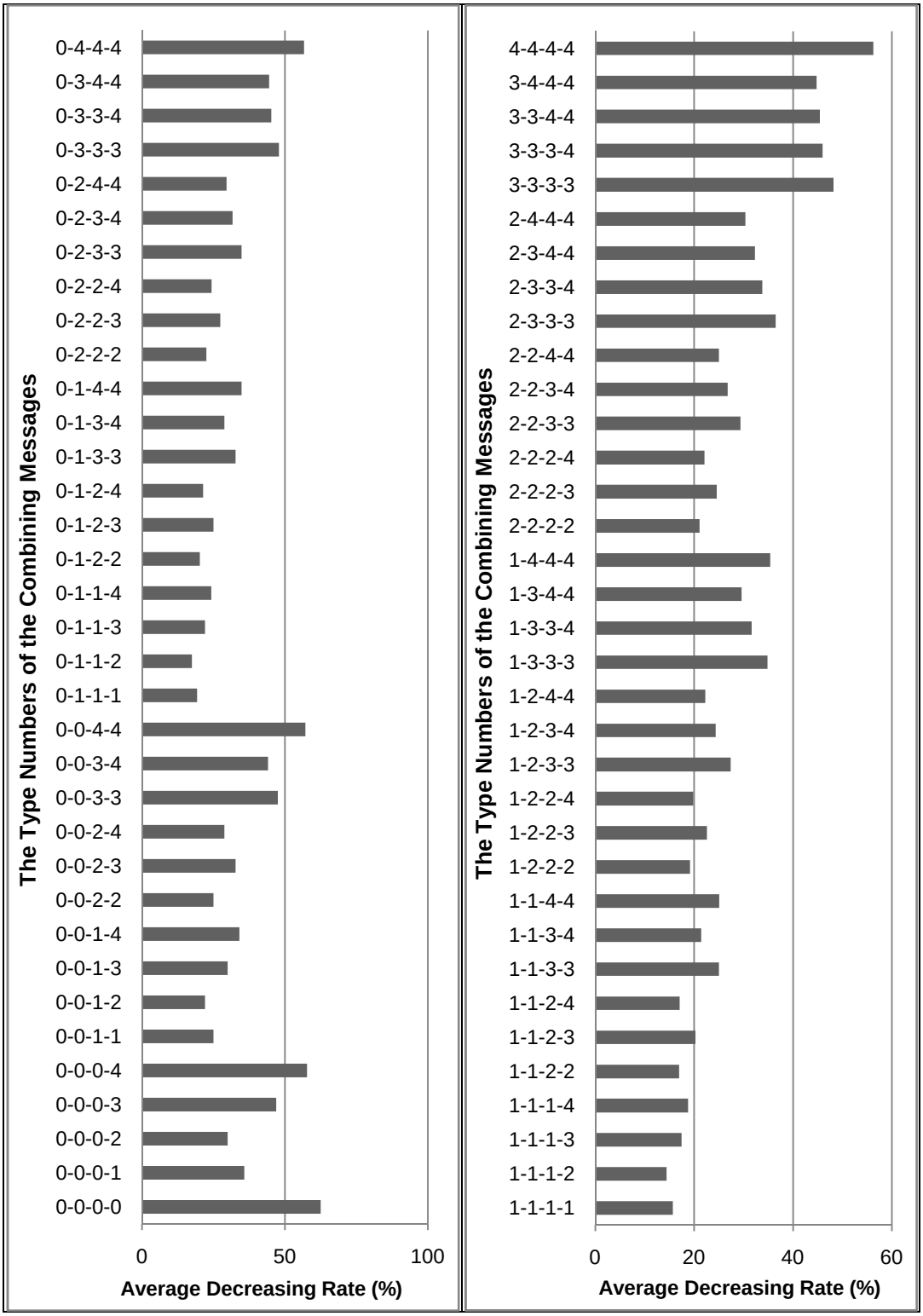


Figure 3.6. Average of the decreasing rates according to the combining messages types for four messages combined

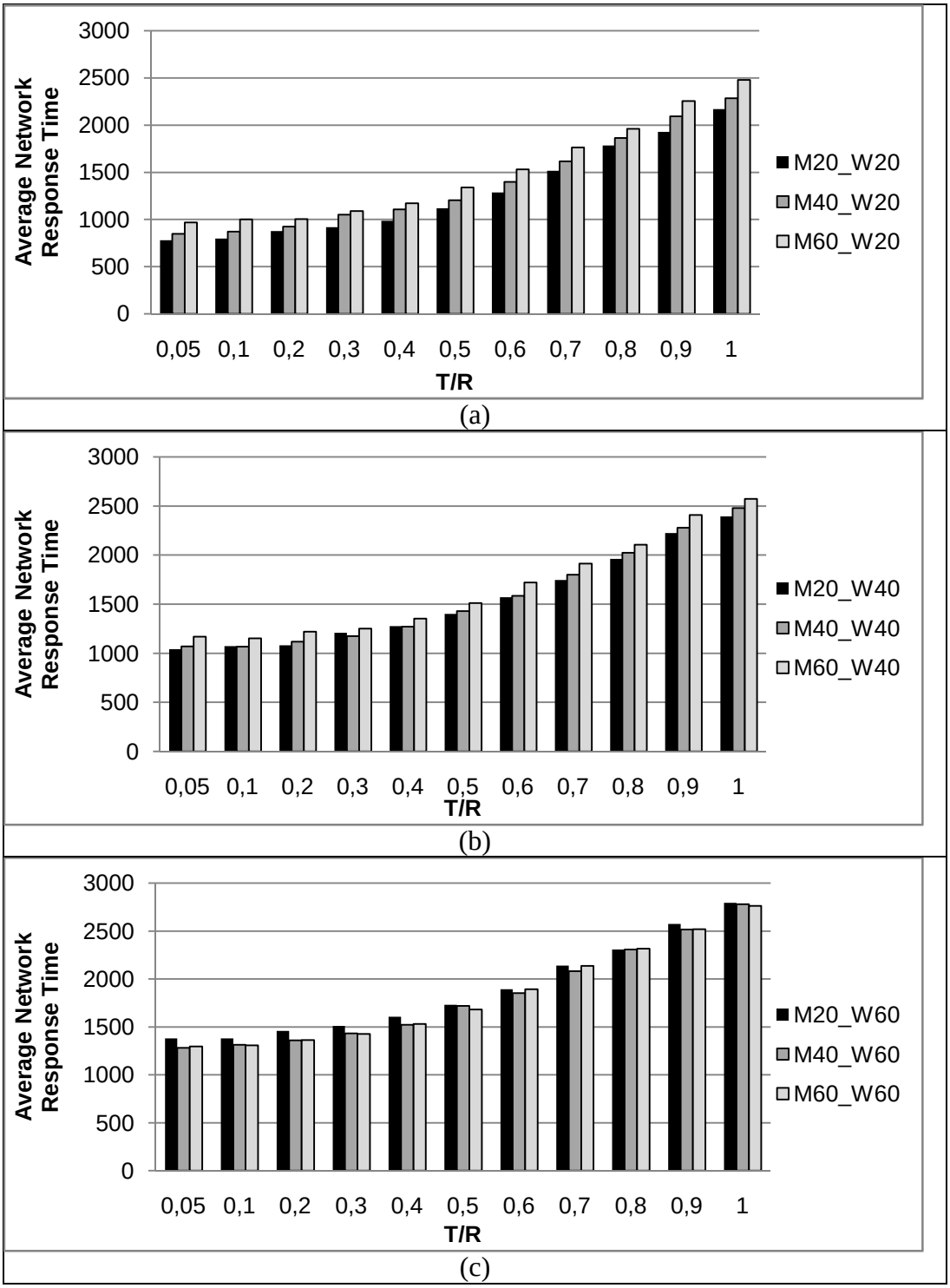


Figure 4.3. Average network response time of the base system for different values of T/R, for (a) P(W) = 0.2, (b) P(W) = 0.4, (c) P(W) = 0.6 and P(M) = 0.2, 0.4 and 0.6 under UN traffic

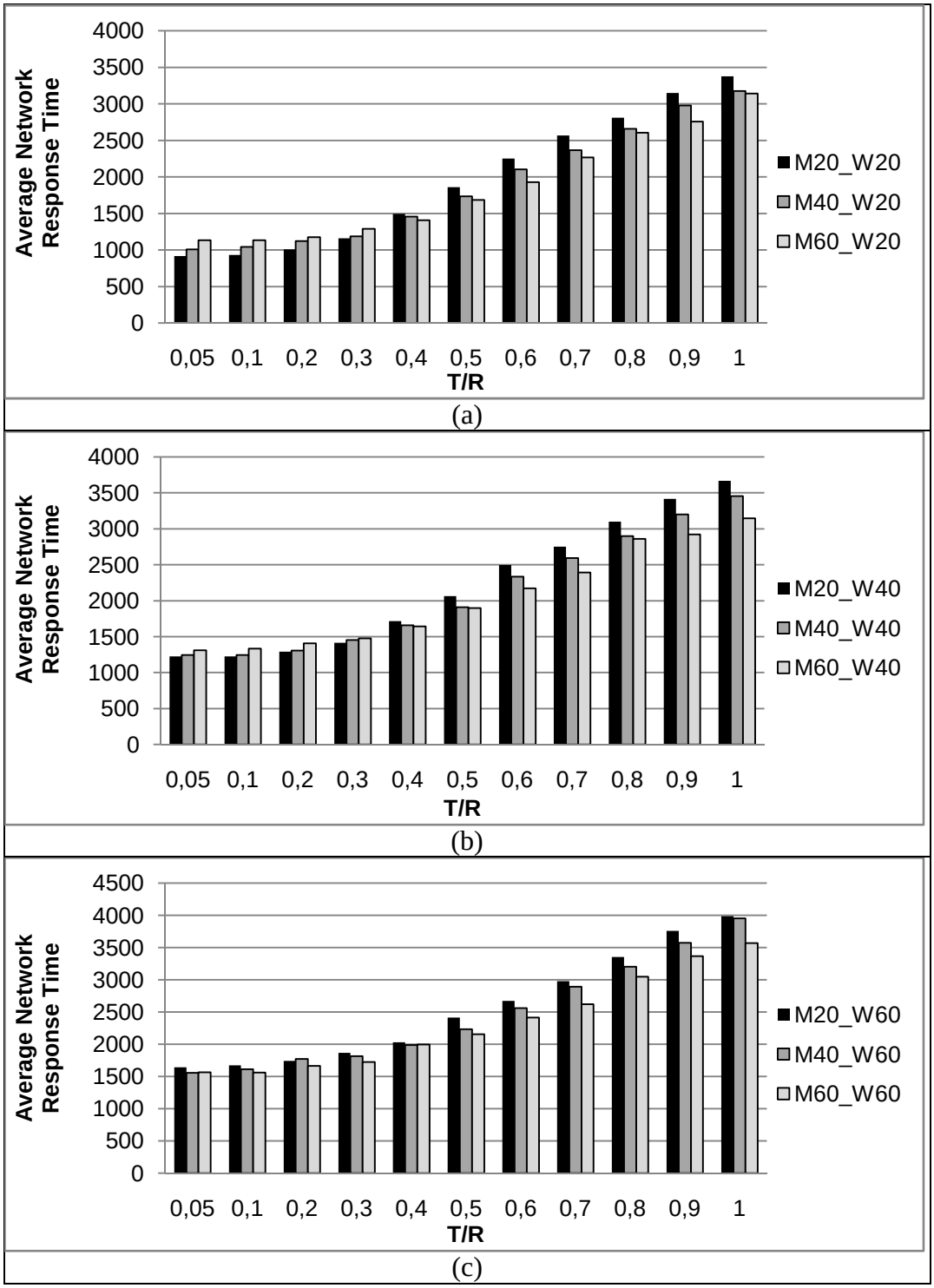


Figure 4.4. Average network response time of the base system for different values of T/R, for (a) P(W) = 0.2, (b) P(W) = 0.4, (c) P(W) = 0.6 and P(M) = 0.2, 0.4 and 0.6 under HR traffic

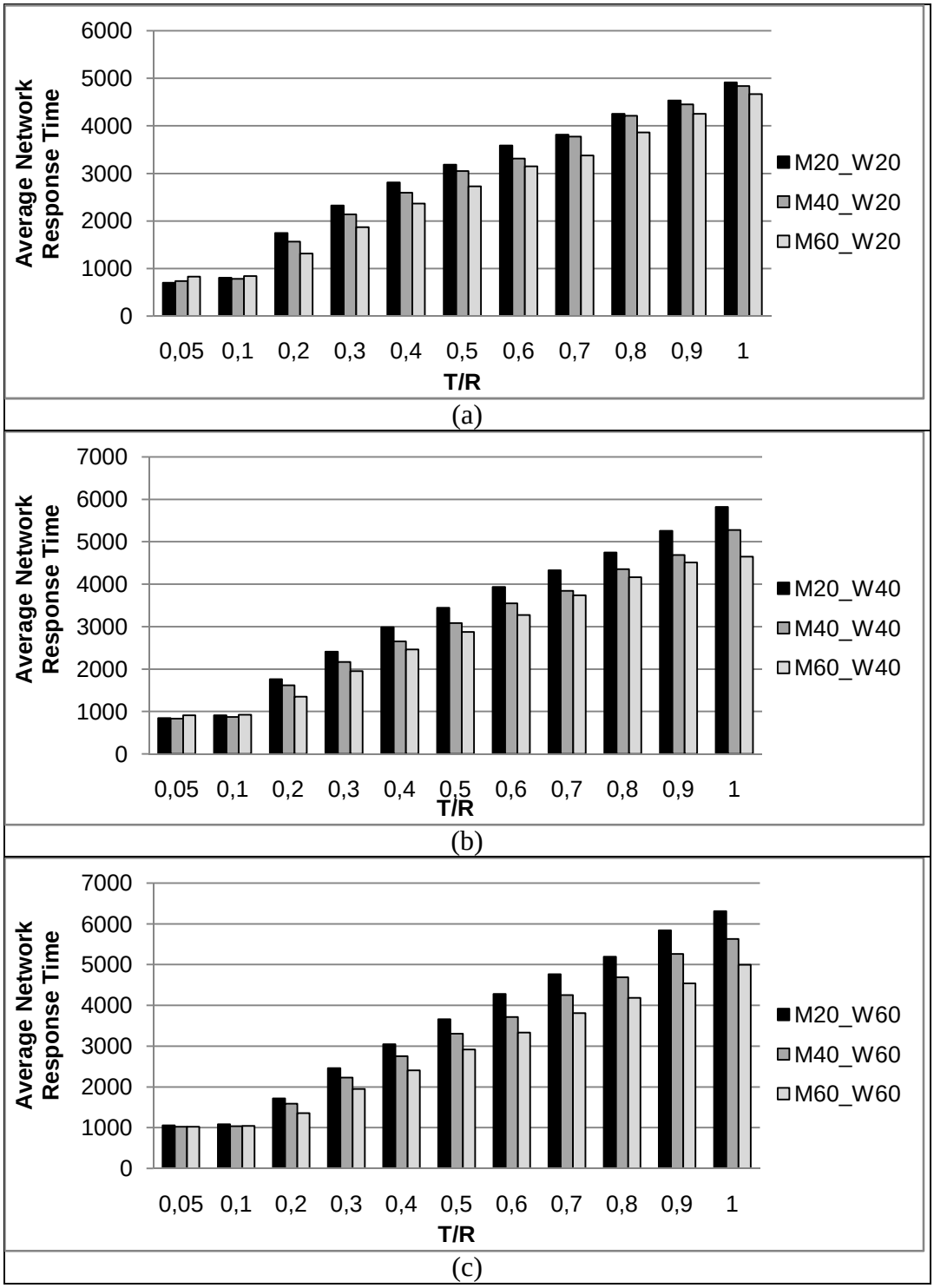


Figure 4.5. Average network response time of the base system for different values of T/R, for (a) P(W) = 0.2, (b) P(W) = 0.4, (c) P(W) = 0.6 and P(M) = 0.2, 0.4 and 0.6 under BR traffic

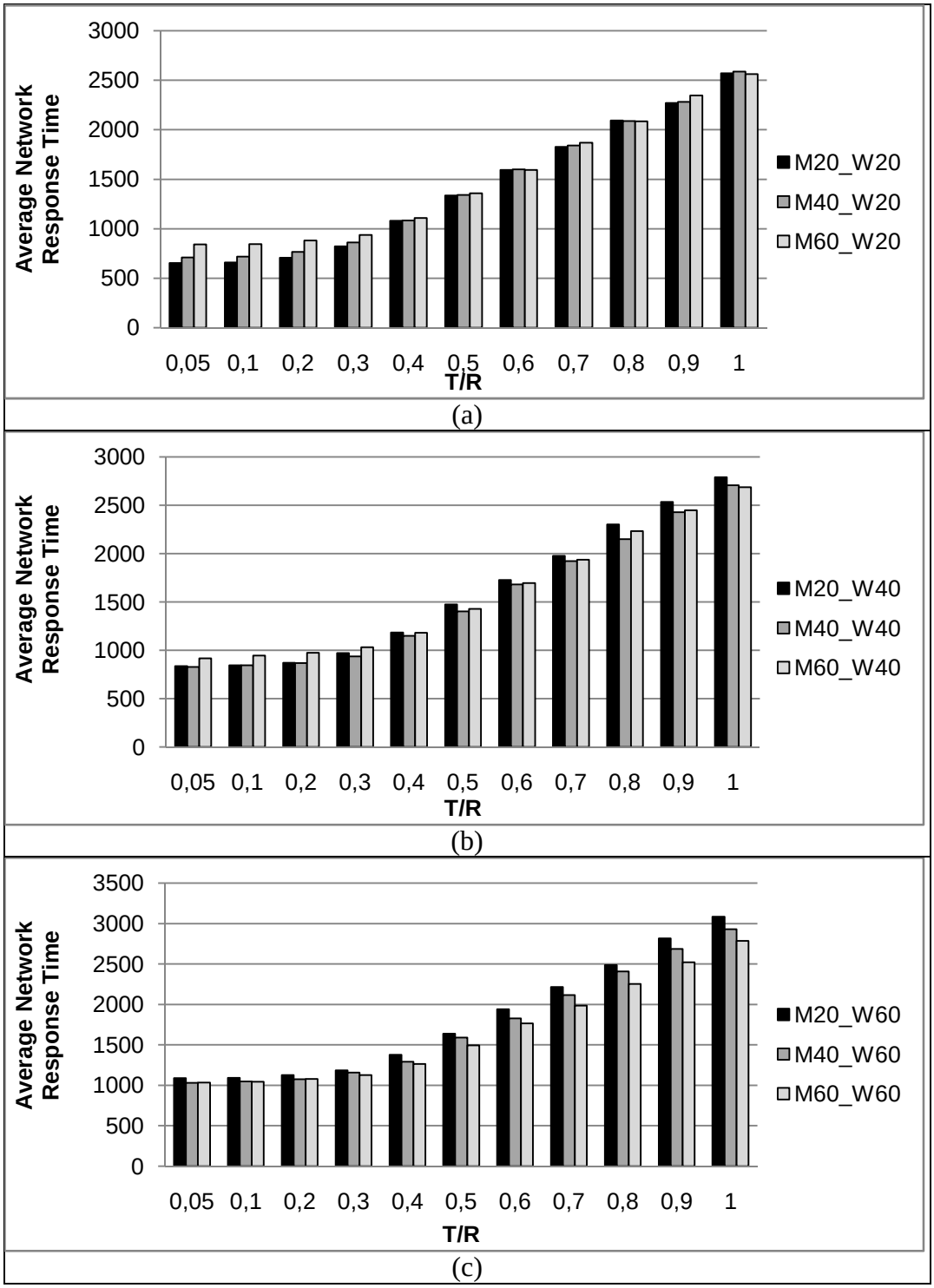


Figure 4.6. Average network response time of the base system for different values of T/R, for (a) P(W) = 0.2, (b) P(W) = 0.4, (c) P(W) = 0.6 and P(M) = 0.2, 0.4 and 0.6 under PS traffic

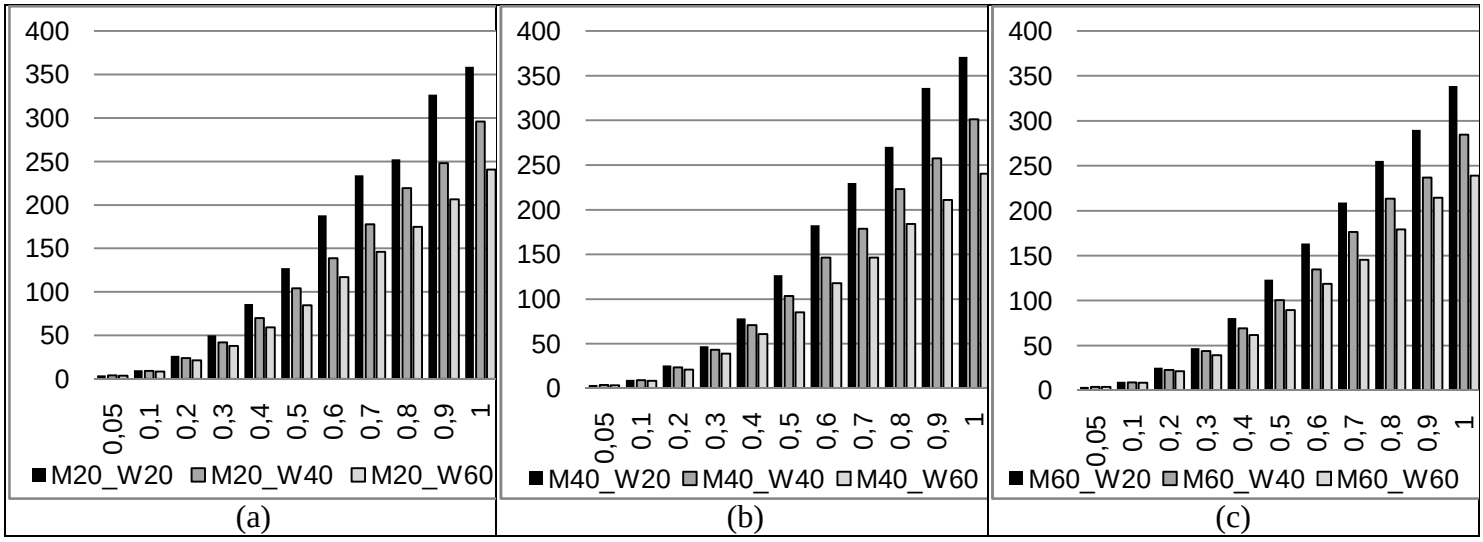


Figure 4.7. Average channel waiting time of the base system (vertical axis) for different values of T/R (horizontal axis), for (a) P(M) = 0.2, (b) P(M) = 0.4, (c) P(M) = 0.6 and P(W) = 0.2, 0.4 and 0.6 under UN traffic

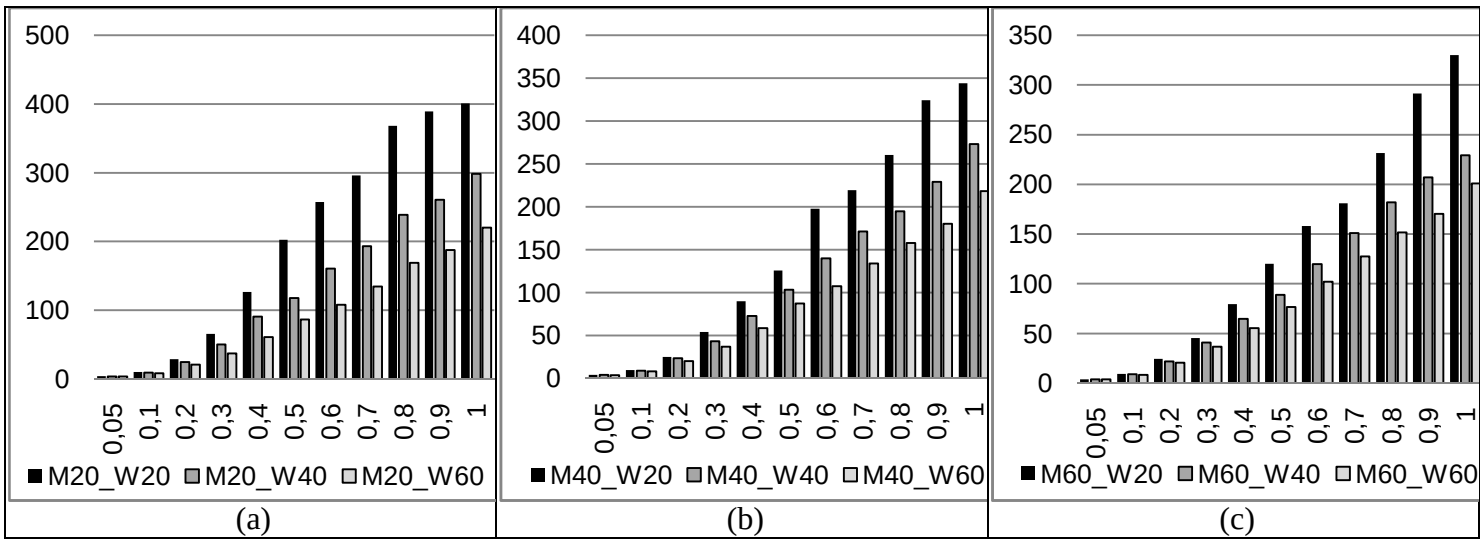
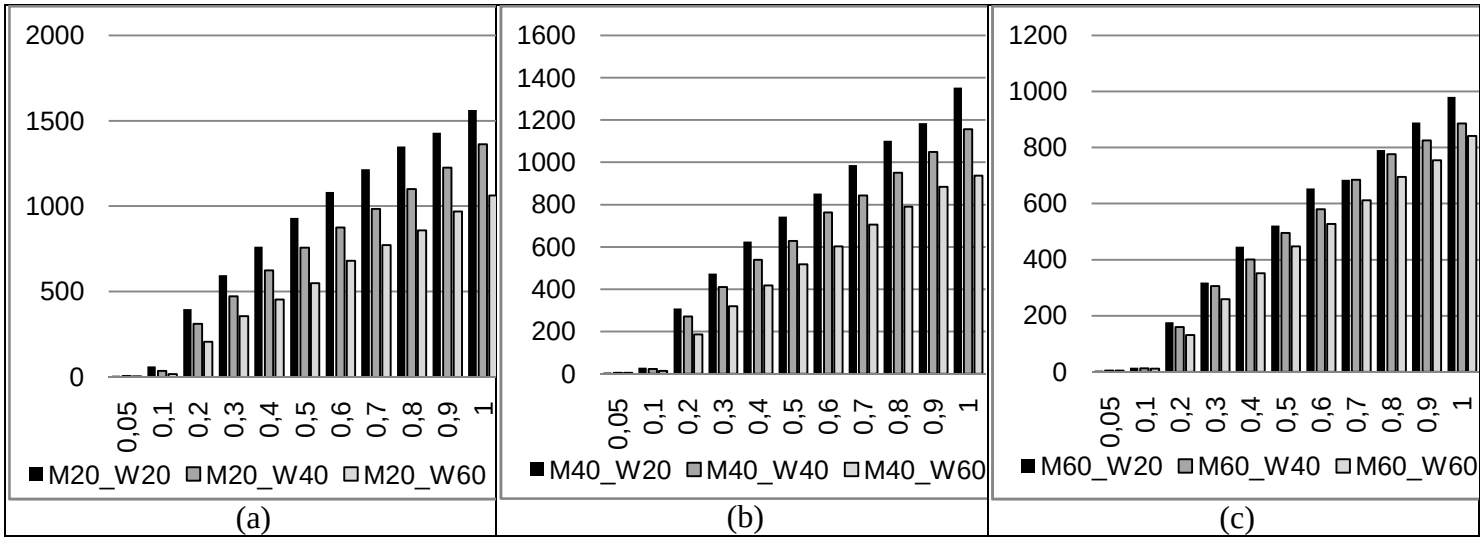
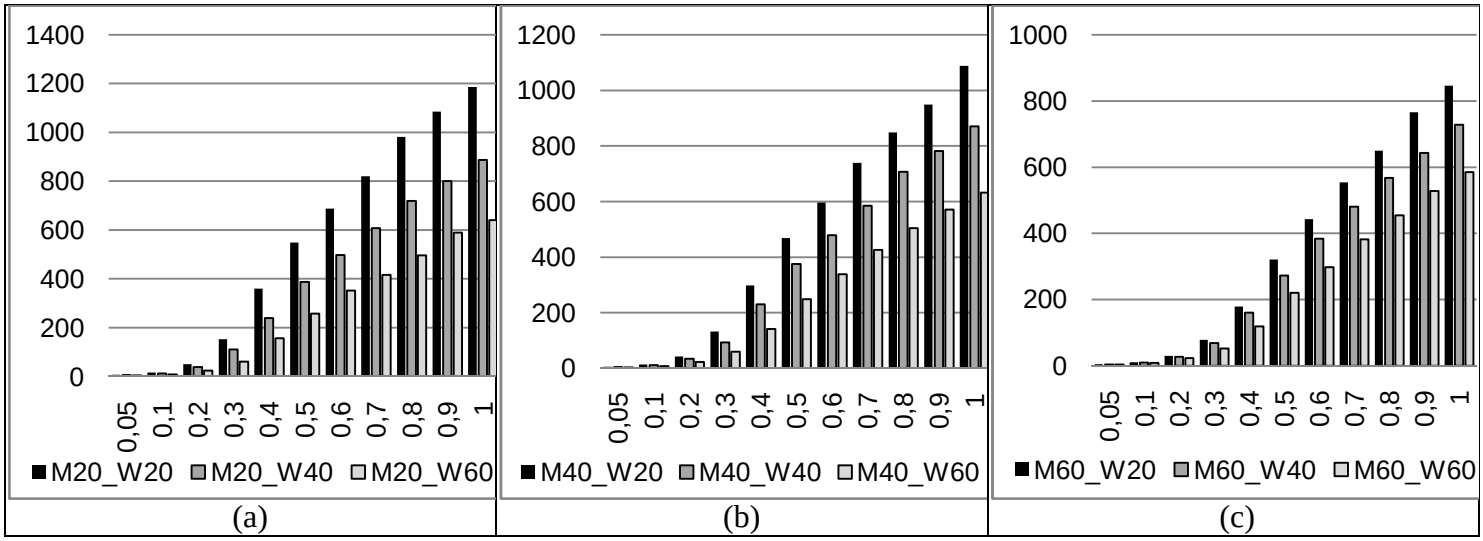


Figure 4.8. Average channel waiting time of the base system (vertical axis) for different values of T/R (horizontal axis), for (a) P(M) = 0.2, (b) P(M) = 0.4, (c) P(M) = 0.6 and P(W) = 0.2, 0.4 and 0.6 under HR traffic



(a) (b) (c)

Figure 4.9. Average channel waiting time of the base system (vertical axis) for different values of T/R (horizontal axis), for (a) P(M) = 0.2, (b) P(M) = 0.4, (c) P(M) = 0.6 and P(W) = 0.2, 0.4 and 0.6 under BR traffic



(a) (b) (c)

Figure 4.10. Average channel waiting time of the base system (vertical axis) for different values of T/R (horizontal axis), for (a) P(M) = 0.2, (b) P(M) = 0.4, (c) P(M) = 0.6 and P(W) = 0.2, 0.4 and 0.6 under PS traffic

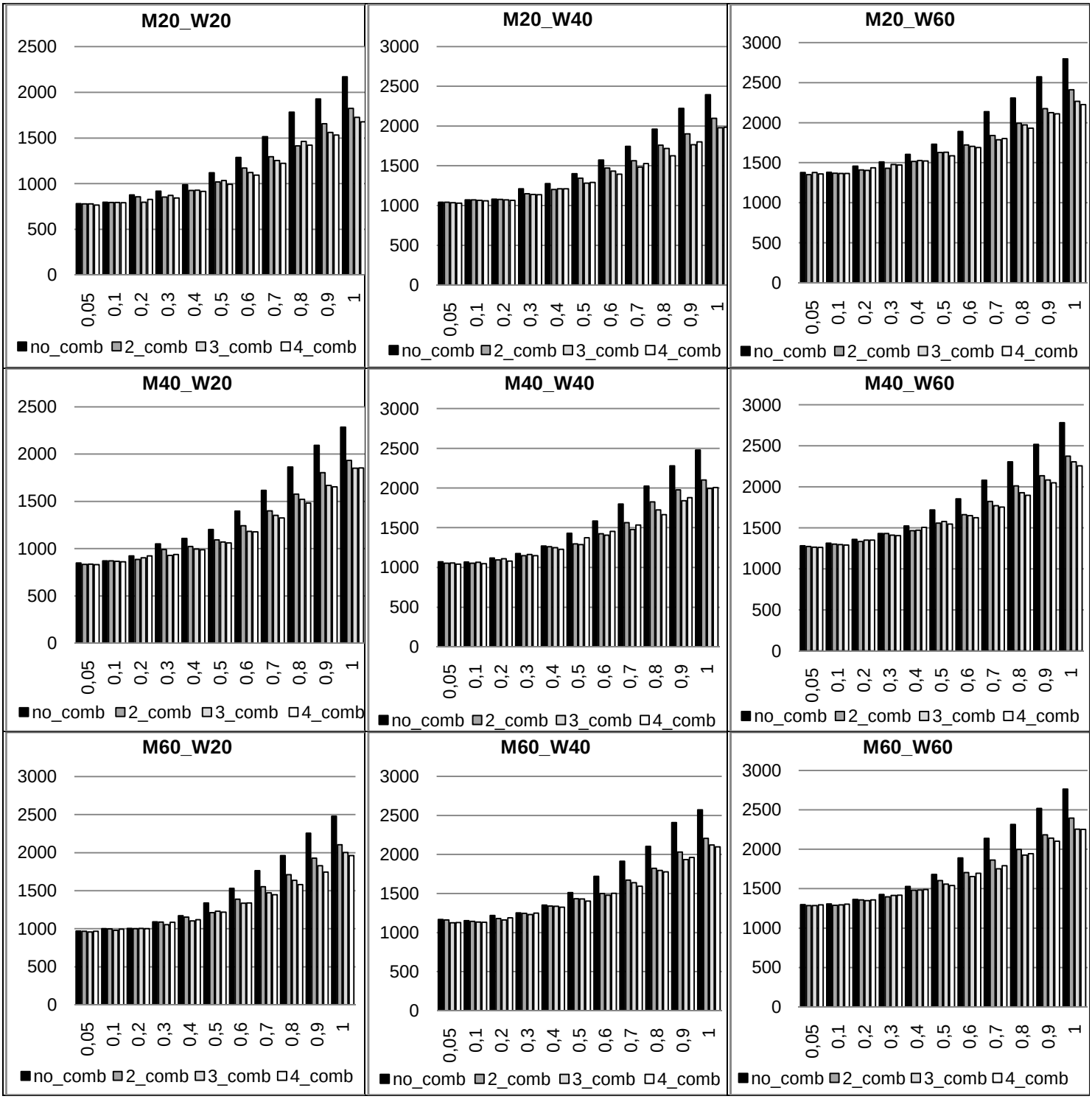


Figure 4.11. Average network response time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic

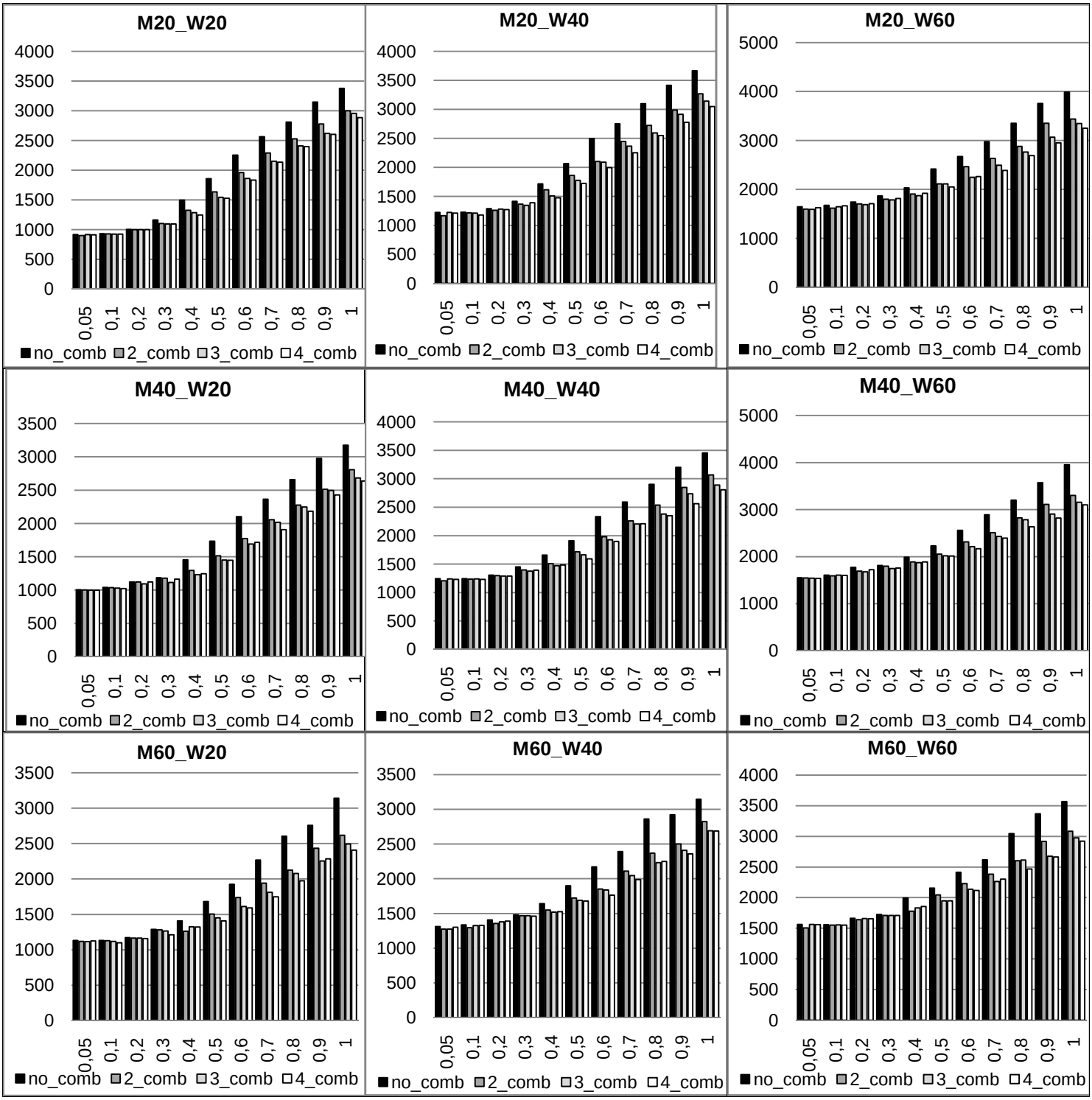


Figure 4.12. Average network response time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under HR traffic

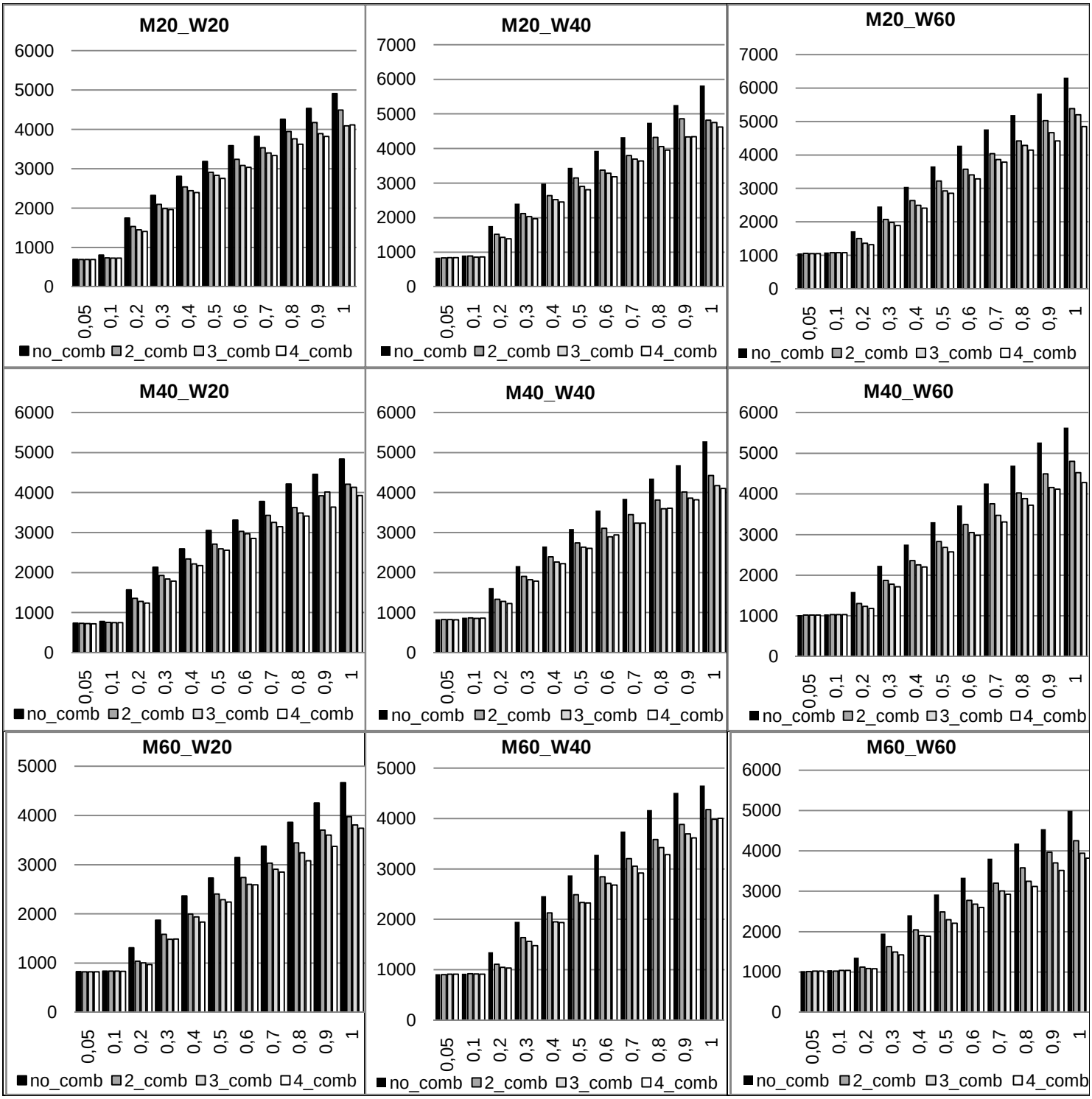


Figure 4.13. Average network response time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under BR traffic

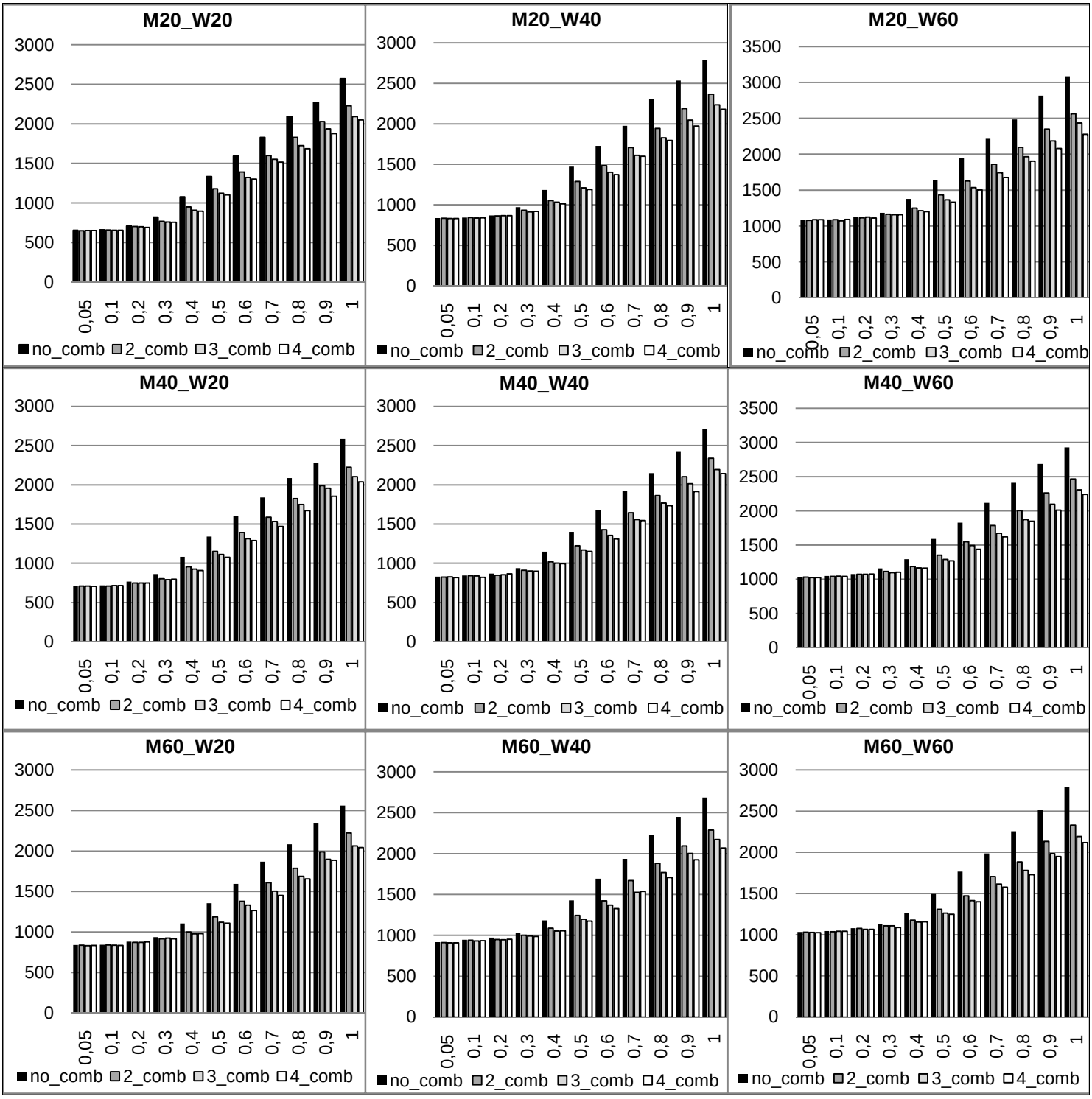


Figure 4.14. Average network response time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under PS traffic

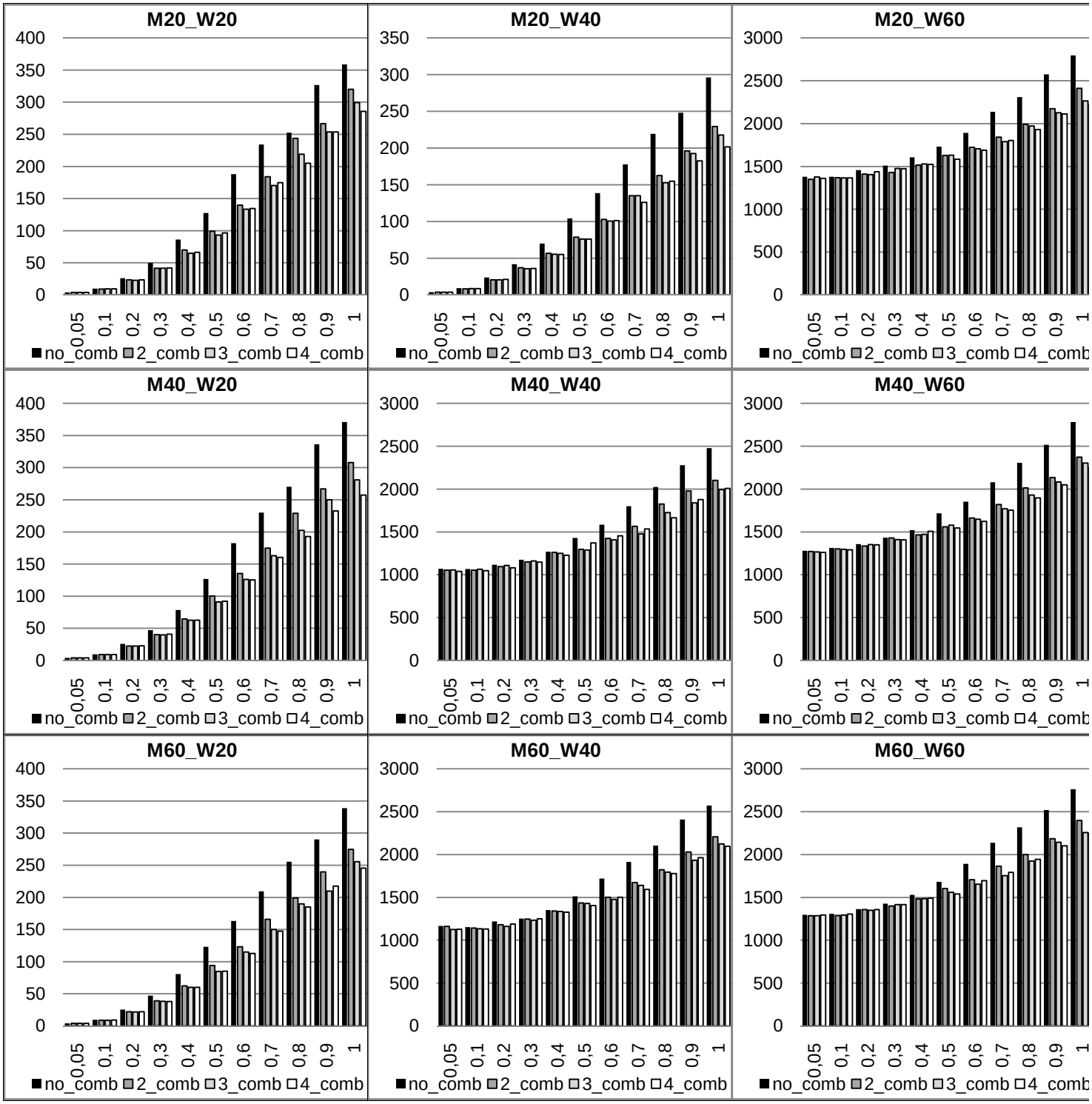


Figure 4.15. Average channel waiting time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic

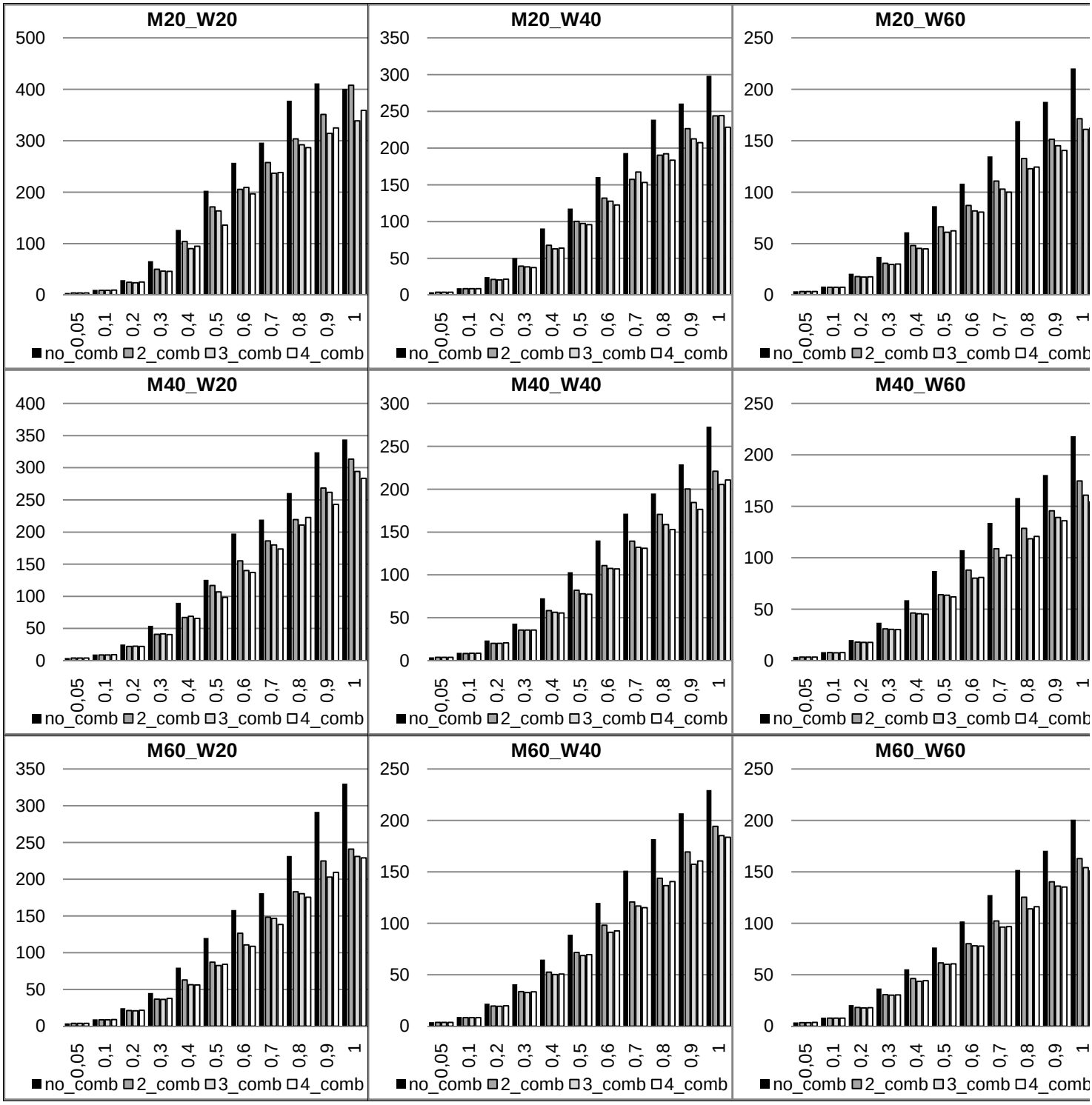


Figure 4.16. Average channel waiting time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under HR traffic

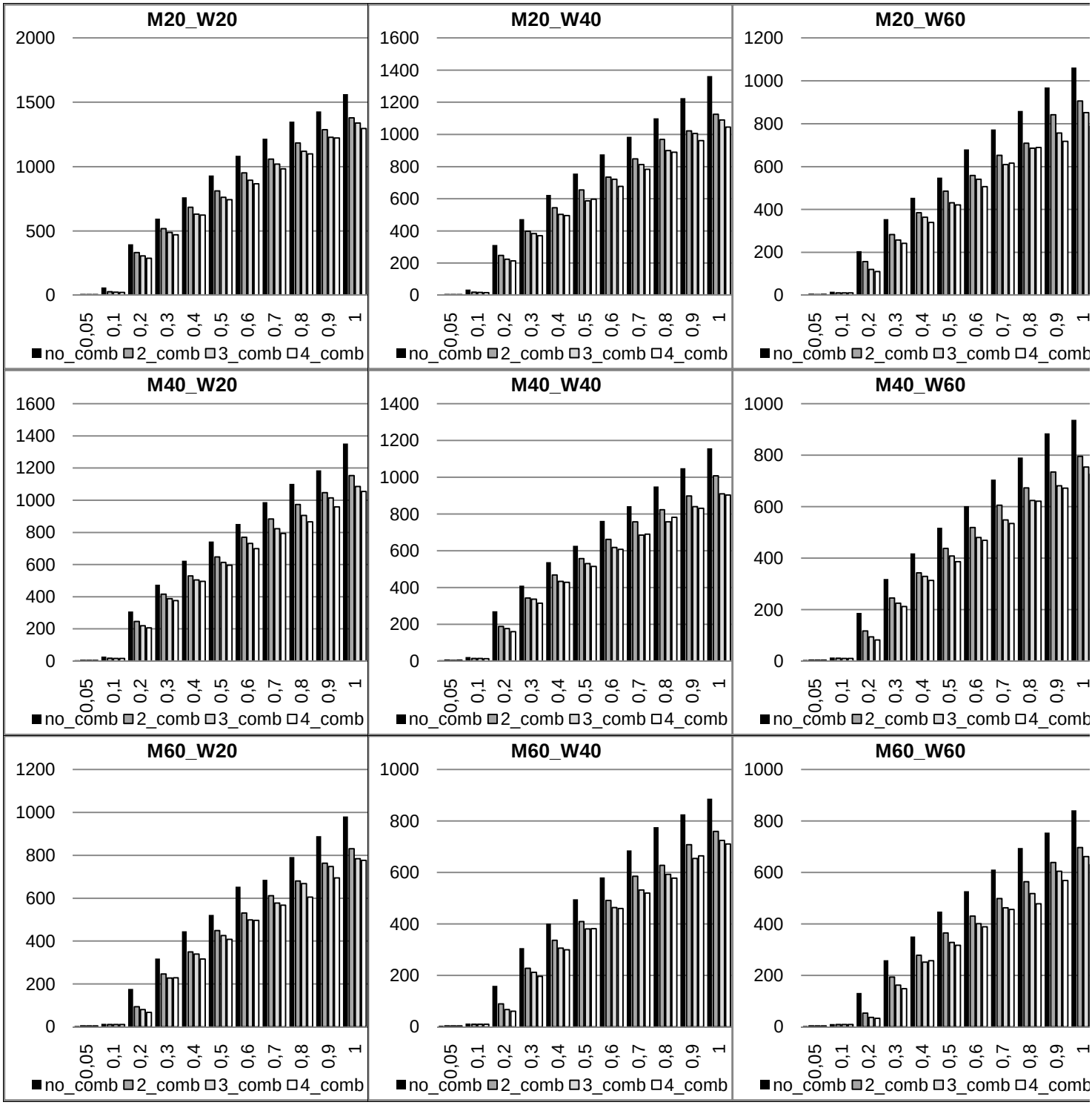


Figure 4.17. Average channel waiting time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under BR traffic

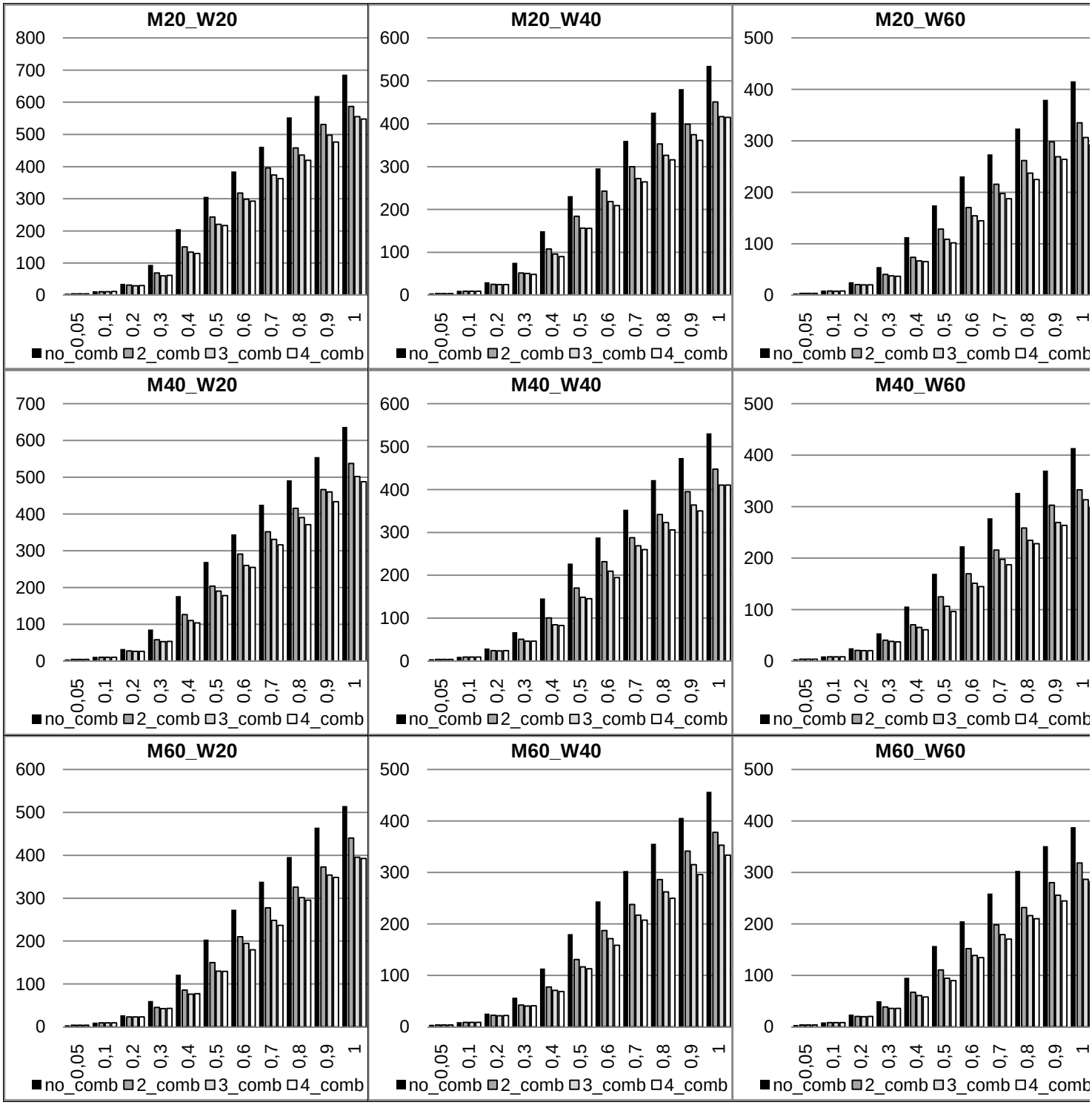


Figure 4.18. Average channel waiting time of the base system with the message combining algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under PS traffic

Table 4.1 Maximum number of messages in the input queues of the base system with message combining algorithm, for different values of traffic pattern, P(M) and P(W)

		MAXIMUM MESSAGE NUMBER						MAXIMUM MESSAGE NUMBER			
		BASE	2_COMB	3_COMB	4_COMB			BASE	2_COMB	3_COMB	4_COMB
UN	M20_W20	120	121	103	108	BR	M20_W20	53	54	58	59
	M20_W40	130	110	99	111		M20_W40	61	57	59	61
	M20_W60	97	142	90	124		M20_W60	71	61	64	61
	M40_W20	114	167	145	133		M40_W20	58	68	65	63
	M40_W40	125	94	111	114		M40_W40	62	63	64	66
	M40_W60	102	101	133	105		M40_W60	64	63	61	67
	M60_W20	102	102	104	102		M60_W20	63	71	66	75
	M60_W40	122	85	118	88		M60_W40	65	68	68	66
	M60_W60	88	88	98	89		M60_W60	70	68	71	68
HR	M20_W20	323	310	309	343	PS	M20_W20	57	55	57	55
	M20_W40	332	363	340	369		M20_W40	61	60	55	55
	M20_W60	284	339	361	344		M20_W60	65	70	58	58
	M40_W20	364	327	365	333		M40_W20	57	62	59	65
	M40_W40	341	372	337	344		M40_W40	66	62	62	62
	M40_W60	260	306	309	310		M40_W60	70	68	61	61
	M60_W20	332	272	271	291		M60_W20	60	70	65	69
	M60_W40	319	262	262	298		M60_W40	65	69	66	67
	M60_W60	266	257	252	255		M60_W60	73	66	69	67

Table 4.2 Maximum number of messages in the horizontal output channels of the base system with message combining algorithm, for different values of traffic pattern, P(M) and P(W)

		MAXIMUM MESSAGE NUMBER						MAXIMUM MESSAGE NUMBER			
		BASE	2_COMB	3_COMB	4_COMB			BASE	2_COMB	3_COMB	4_COMB
UN	M20_W20	95	75	64	76	BR	M20_W20	21	21	19	19
	M20_W40	71	55	61	53		M20_W40	24	18	22	23
	M20_W60	36	36	36	31		M20_W60	30	20	21	25
	M40_W20	57	57	44	48		M40_W20	23	22	23	22
	M40_W40	43	40	31	33		M40_W40	24	24	23	24
	M40_W60	34	33	29	30		M40_W60	30	25	24	22
	M60_W20	53	34	35	38		M60_W20	29	29	29	29
	M60_W40	32	28	32	32		M60_W40	29	29	29	28
	M60_W60	40	33	29	32		M60_W60	29	28	28	26
HR	M20_W20	53	53	53	53	PS	M20_W20	22	22	22	21
	M20_W40	43	38	41	46		M20_W40	27	25	26	27
	M20_W60	40	34	35	37		M20_W60	38	35	36	36
	M40_W20	45	32	43	44		M40_W20	35	29	25	26
	M40_W40	35	34	34	33		M40_W40	35	27	27	31
	M40_W60	31	31	29	31		M40_W60	35	30	30	30
	M60_W20	37	35	36	37		M60_W20	32	32	32	30
	M60_W40	34	34	33	32		M60_W40	34	32	32	34
	M60_W60	34	34	34	31		M60_W60	38	38	34	34

Table 4.3 Maximum number of messages in the vertical output channels of the base system with message combining algorithm, for different values of traffic pattern, P(M) and P(W)

		MAXIMUM MESSAGE NUMBER						MAXIMUM MESSAGE NUMBER			
		BASE	2_COMB	3_COMB	4_COMB			BASE	2_COMB	3_COMB	4_COMB
UN	M20_W20	53	52	52	51	BR	M20_W20	193	187	192	190
	M20_W40	39	39	38	39		M20_W40	193	186	184	187
	M20_W60	36	36	35	35		M20_W60	177	176	177	168
	M40_W20	63	51	41	34		M40_W20	189	188	188	186
	M40_W40	44	42	42	39		M40_W40	169	165	165	168
	M40_W60	37	37	37	37		M40_W60	177	156	159	159
	M60_W20	52	47	36	33		M60_W20	181	180	176	180
	M60_W40	48	34	38	38		M60_W40	164	178	168	177
	M60_W60	41	41	40	40		M60_W60	155	152	155	151
HR	M20_W20	79	77	72	77	PS	M20_W20	74	74	71	70
	M20_W40	66	64	66	62		M20_W40	70	68	69	70
	M20_W60	64	59	61	64		M20_W60	71	65	62	67
	M40_W20	78	68	72	70		M40_W20	84	68	63	73
	M40_W40	63	60	58	63		M40_W40	83	71	64	62
	M40_W60	61	53	52	56		M40_W60	85	70	67	71
	M60_W20	85	48	49	53		M60_W20	82	75	79	65
	M60_W40	65	51	50	52		M60_W40	80	73	66	64
	M60_W60	50	49	50	50		M60_W60	65	61	64	65

Table 4.4 Maximum number of messages in the input queues of the base system with message priority algorithm, for different values of traffic pattern, P(M) and P(W)

		MAXIMUM MESSAGE NUMBER						MAXIMUM MESSAGE NUMBER				
		BASE	CACHE	DIR	CACHE_DIR	CRITICAL		BASE	CACHE	DIR	CACHE_DIR	CRITICAL
UN	M20_W20	120	75	113	89	87	BR	M20_W20	53	50	49	50	52
	M20_W40	130	81	127	96	80		M20_W40	61	48	53	48	56
	M20_W60	97	94	97	96	95		M20_W60	71	51	57	54	66
	M40_W20	114	107	113	111	110		M40_W20	58	53	56	56	56
	M40_W40	125	108	125	115	124		M40_W40	62	53	57	56	59
	M40_W60	102	95	101	101	100		M40_W60	64	54	58	58	64
	M60_W20	102	100	102	95	101		M60_W20	63	58	61	63	63
	M60_W40	122	109	98	92	122		M60_W40	65	58	63	65	65
	M60_W60	88	86	88	87	87		M60_W60	70	65	67	62	61
HR	M20_W20	323	294	320	313	280	PS	M20_W20	57	47	50	49	53
	M20_W40	332	317	331	325	332		M20_W40	61	54	53	51	61
	M20_W60	284	283	284	282	280		M20_W60	65	51	54	52	64
	M40_W20	364	362	343	344	327		M40_W20	57	53	56	55	55
	M40_W40	341	331	337	341	340		M40_W40	66	56	58	58	57
	M40_W60	260	252	255	251	250		M40_W60	70	53	65	56	62
	M60_W20	332	292	266	284	274		M60_W20	60	52	59	56	54
	M60_W40	319	273	291	277	272		M60_W40	65	57	65	60	62
	M60_W60	266	260	266	258	256		M60_W60	73	59	73	71	72

Table 4.5 Maximum number of messages in the horizontal output channels of the base system with message priority algorithm, for different values of traffic pattern, P(M) and P(W)

		MAXIMUM MESSAGE NUMBER							MAXIMUM MESSAGE NUMBER				
		BASE	CACHE	DIR	CACHE_DIR	CRITICAL			BASE	CACHE	DIR	CACHE_DIR	CRITICAL
UN	M20_W20	95	61	67	58	76	BR	M20_W20	21	22	20	22	26
	M20_W40	71	57	51	49	65		M20_W40	24	21	19	25	22
	M20_W60	36	55	33	49	53		M20_W60	30	22	21	24	22
	M40_W20	57	36	61	39	56		M40_W20	23	22	22	20	22
	M40_W40	43	43	44	46	46		M40_W40	24	26	27	24	25
	M40_W60	34	29	31	39	29		M40_W60	30	26	21	25	22
	M60_W20	53	35	32	32	29		M60_W20	29	32	25	24	24
	M60_W40	32	26	32	29	29		M60_W40	29	28	29	29	28
	M60_W60	40	36	38	37	35		M60_W60	29	29	31	28	26
HR	M20_W20	53	59	53	56	65	PS	M20_W20	22	22	24	22	24
	M20_W40	43	62	42	38	83		M20_W40	27	30	29	33	34
	M20_W60	40	38	40	51	40		M20_W60	38	33	37	31	35
	M40_W20	45	39	40	43	56		M40_W20	35	28	25	26	27
	M40_W40	35	31	39	37	34		M40_W40	35	31	34	39	37
	M40_W60	31	33	38	34	32		M40_W60	35	30	34	39	39
	M60_W20	37	40	32	38	35		M60_W20	32	35	36	34	31
	M60_W40	34	41	35	33	31		M60_W40	34	37	37	37	37
	M60_W60	34	40	36	31	32		M60_W60	38	42	41	43	40

Table 4.6 Maximum number of messages in the vertical output channels of the base system with message priority algorithm, for different values of traffic pattern, P(M) and P(W)

		MAXIMUM MESSAGE NUMBER							MAXIMUM MESSAGE NUMBER				
		BASE	CACHE	DIR	CACHE_DIR	CRITICAL			BASE	CACHE	DIR	CACHE_DIR	CRITICAL
UN	M20_W20	53	57	66	55	56	BR	M20_W20	193	196	194	190	192
	M20_W40	39	59	66	69	61		M20_W40	193	186	187	182	187
	M20_W60	36	57	64	59	58		M20_W60	177	176	172	174	172
	M40_W20	63	45	56	48	43		M40_W20	189	188	187	194	189
	M40_W40	44	51	52	46	52		M40_W40	169	173	170	182	175
	M40_W60	37	50	47	48	43		M40_W60	177	158	171	164	168
	M60_W20	52	36	38	37	39		M60_W20	181	178	178	176	173
	M60_W40	48	41	46	36	43		M60_W40	164	174	173	172	176
	M60_W60	41	35	37	32	36		M60_W60	155	153	160	157	159
HR	M20_W20	79	83	98	83	86	PS	M20_W20	74	75	72	78	79
	M20_W40	66	81	79	80	86		M20_W40	70	85	84	78	80
	M20_W60	64	74	66	67	80		M20_W60	71	81	89	75	86
	M40_W20	78	67	64	79	65		M40_W20	84	68	69	71	68
	M40_W40	63	61	66	66	68		M40_W40	83	72	73	71	70
	M40_W60	61	56	77	61	61		M40_W60	85	79	76	69	72
	M60_W20	85	53	56	54	75		M60_W20	82	70	74	78	73
	M60_W40	65	56	54	60	65		M60_W40	80	70	74	74	85
	M60_W60	50	44	42	43	50		M60_W60	65	60	63	62	70

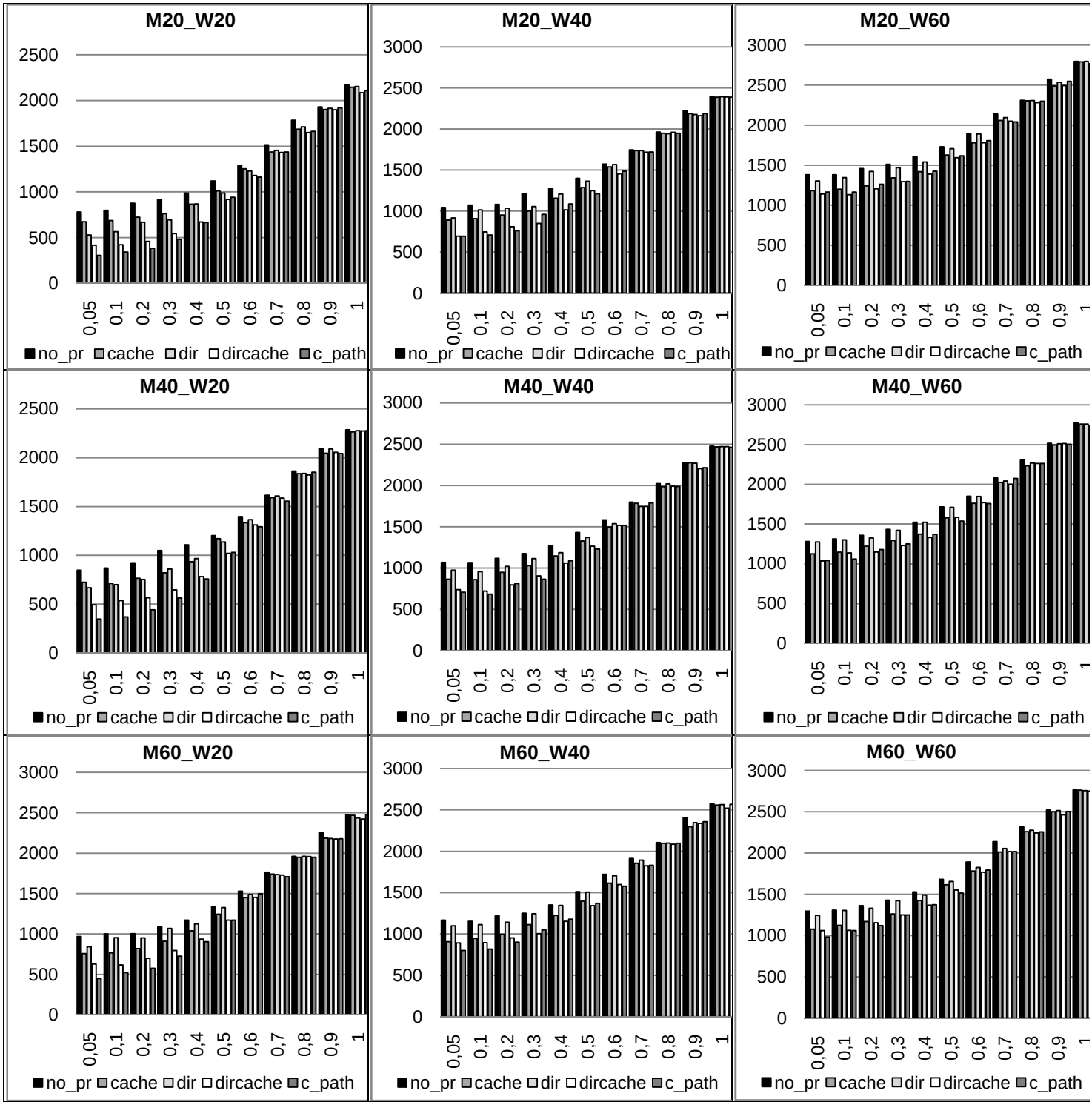


Figure 4.19. Average network response time of the base system with the message priority algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic

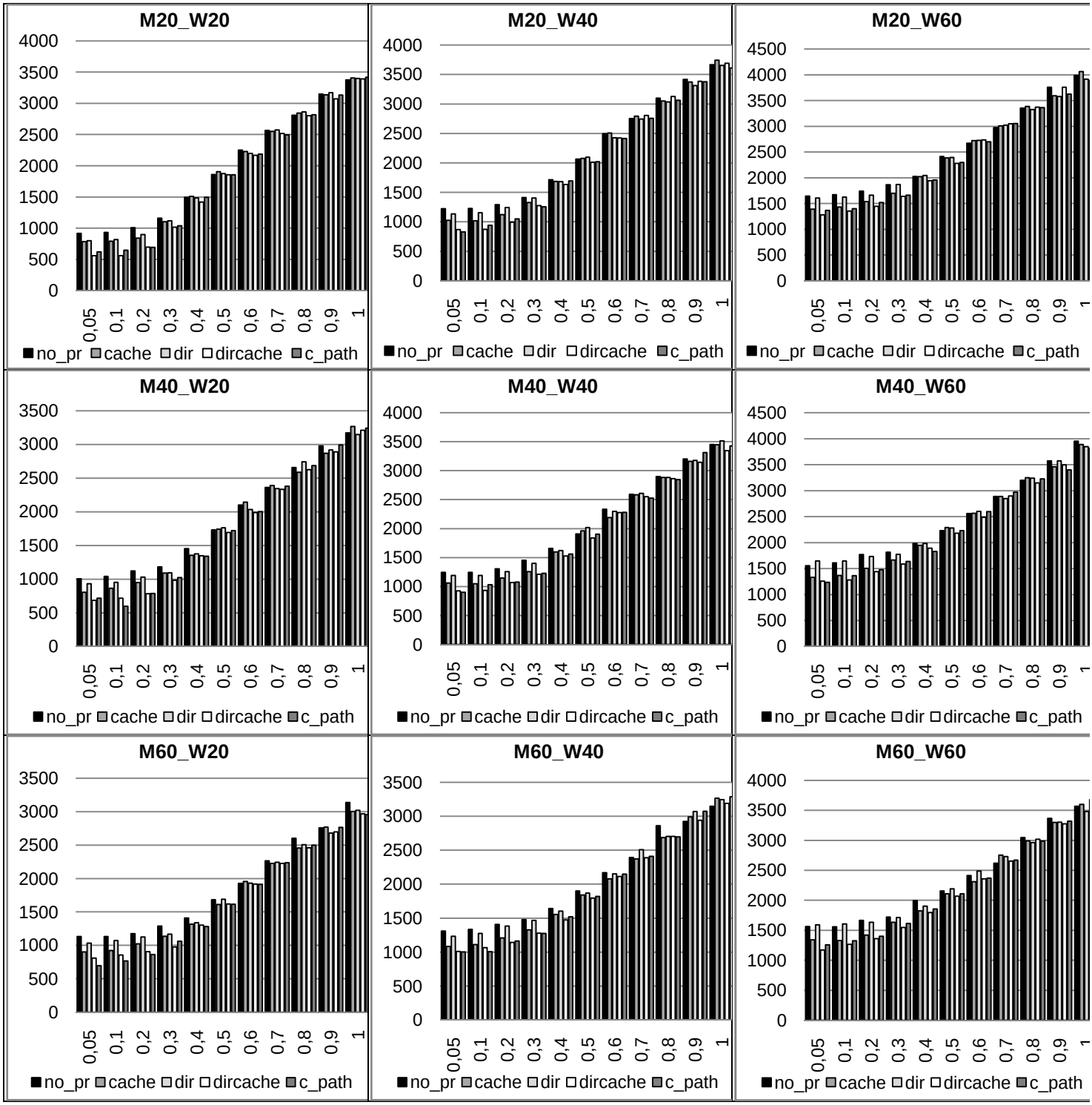


Figure 4.20. Average network response time of the base system with the message priority algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under HR traffic

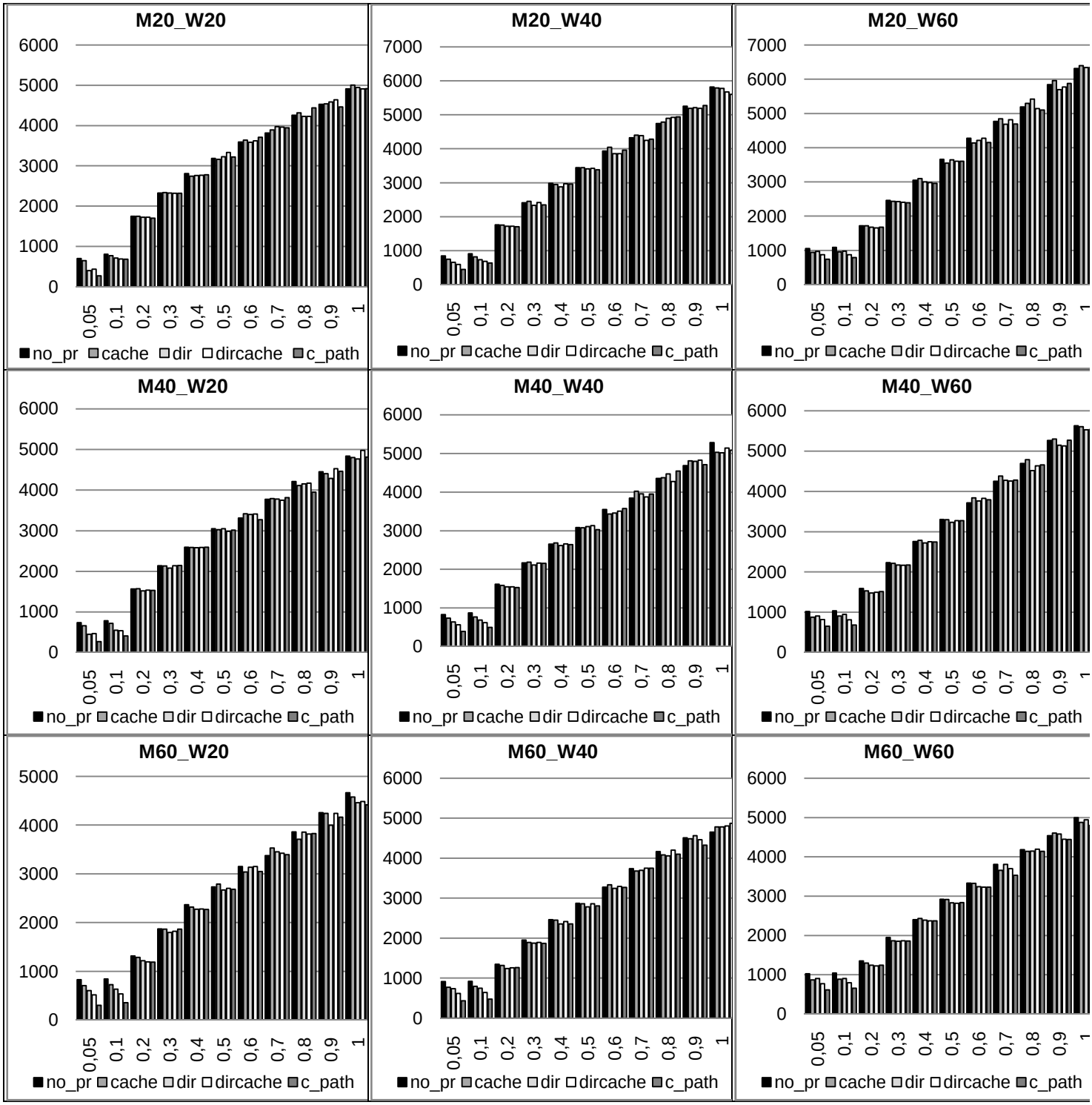


Figure 4.21. Average network response time of the base system with the message priority algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under BR traffic

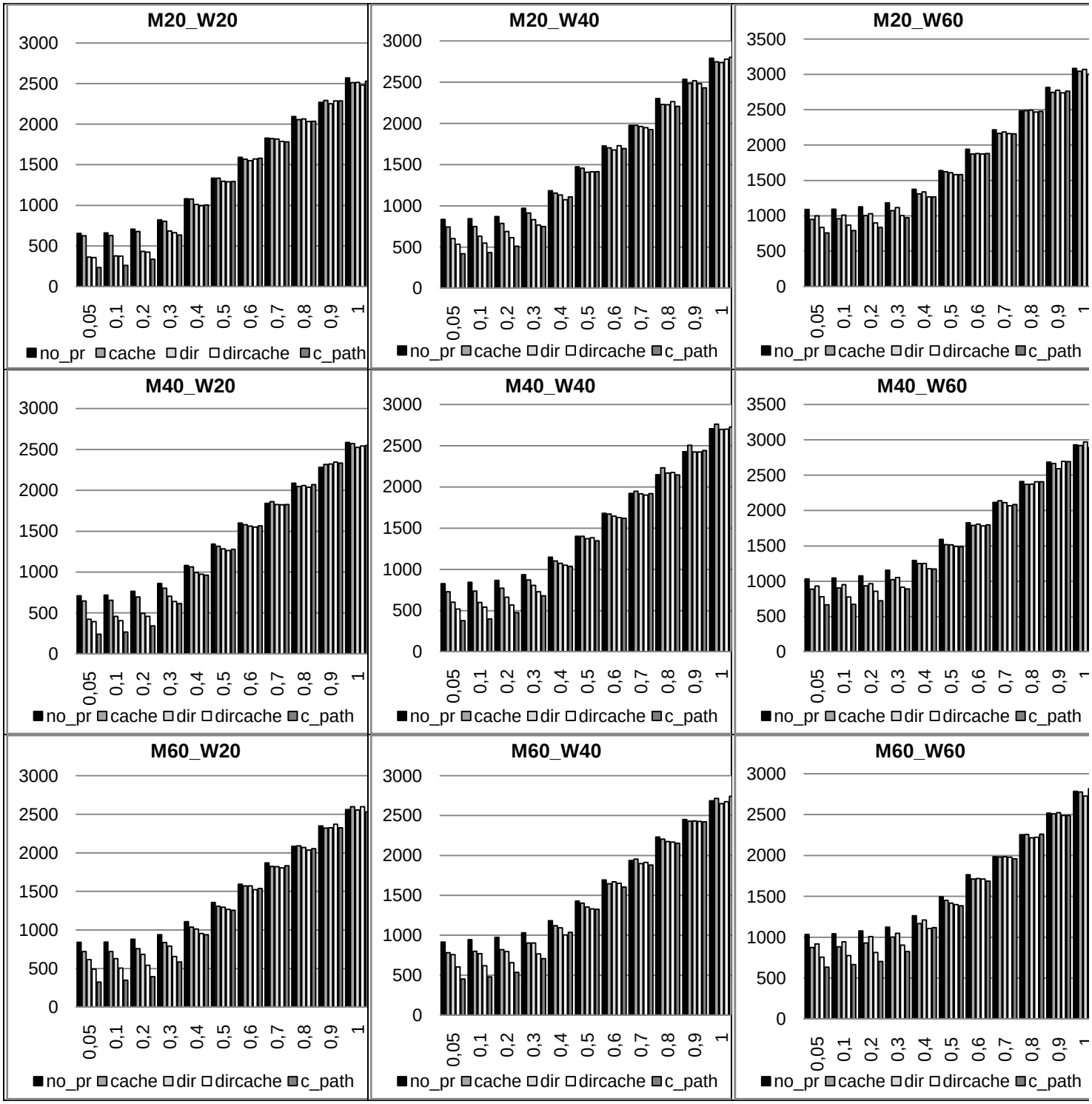


Figure 4.22. Average network response time of the base system with the message priority algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under PS traffic

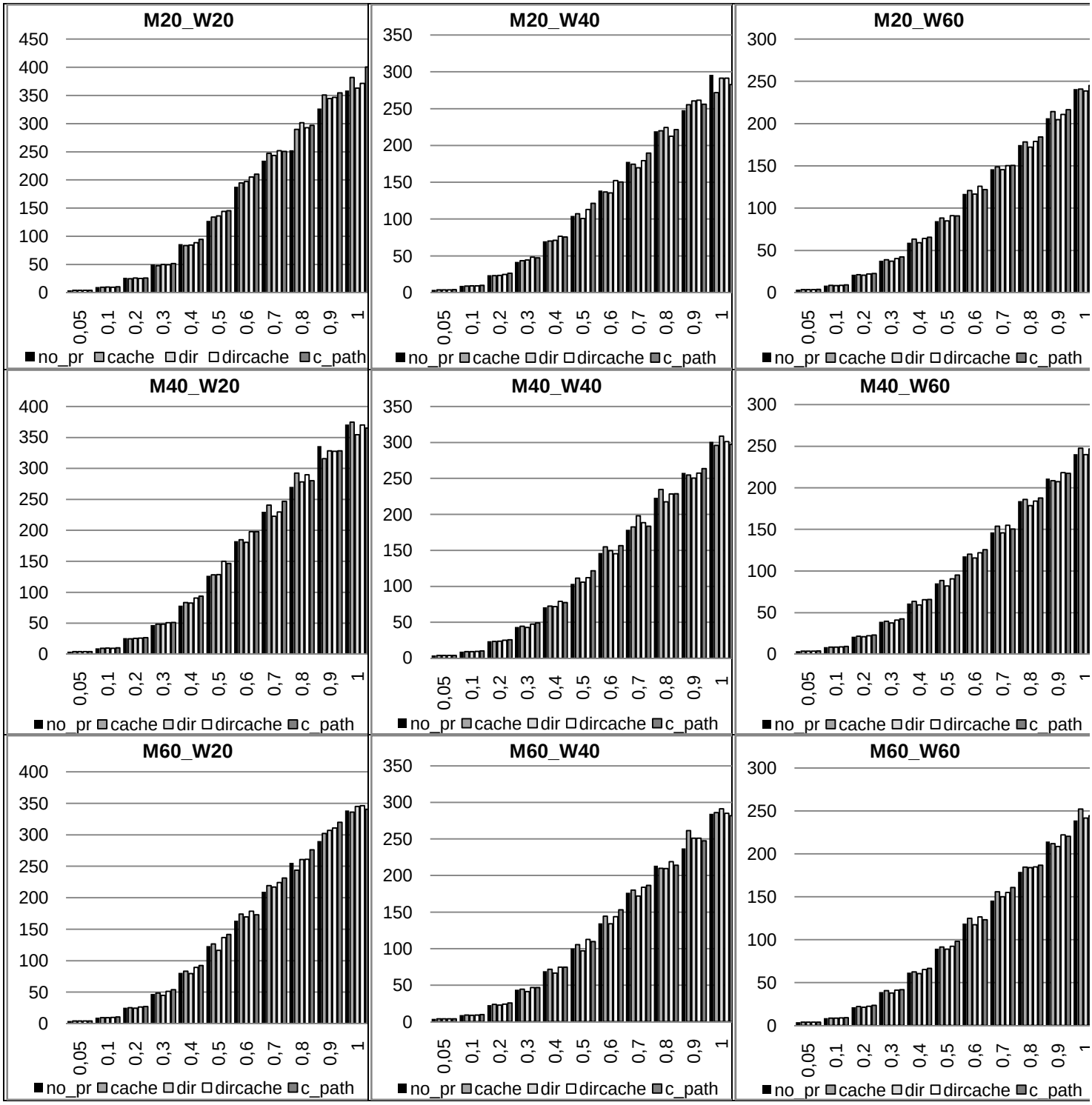


Figure 4.23. Average channel waiting time of the base system with the message priority algorithm (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic

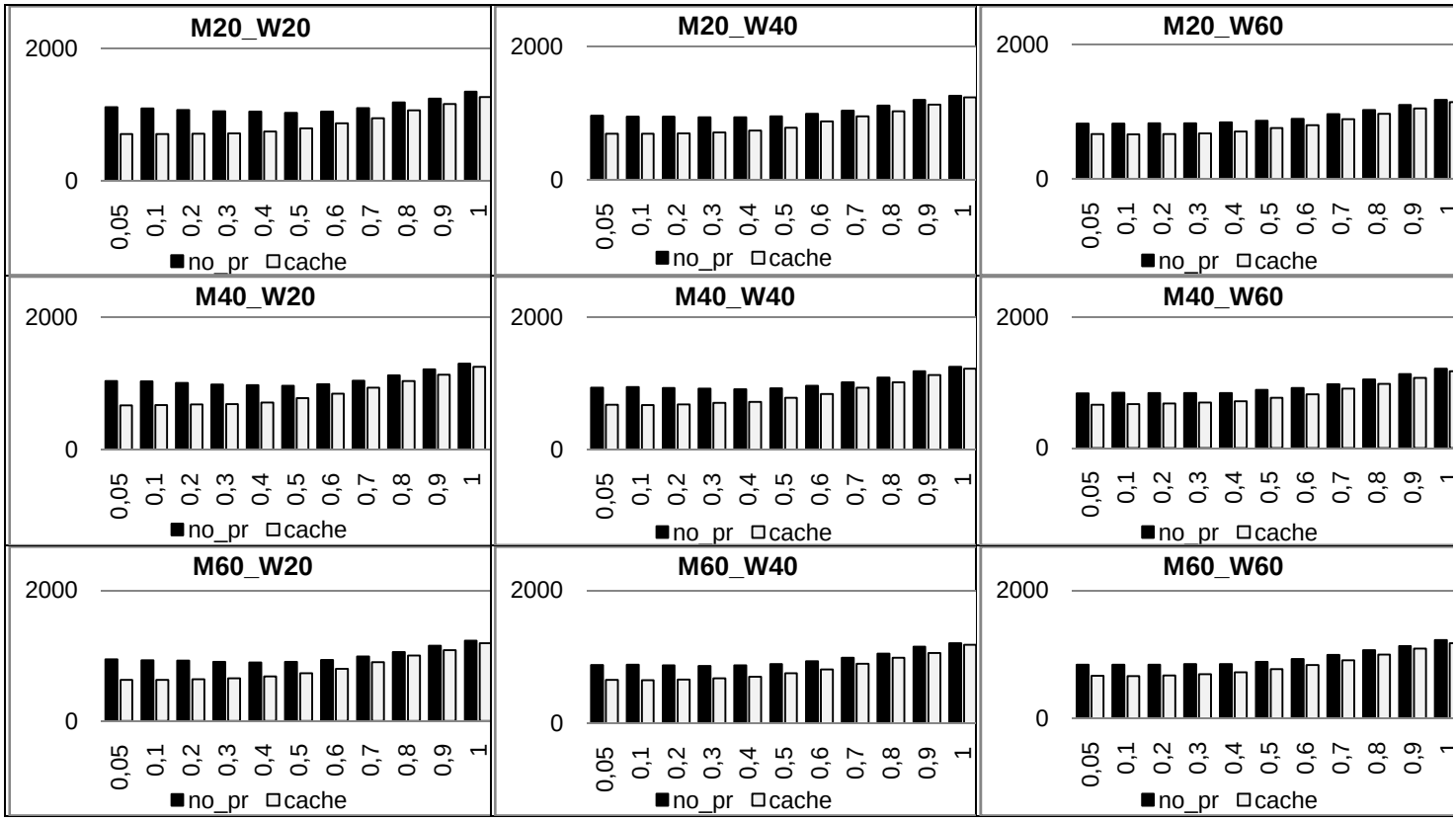


Figure 4.24. Average input waiting time of the base system with the cache priority (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic

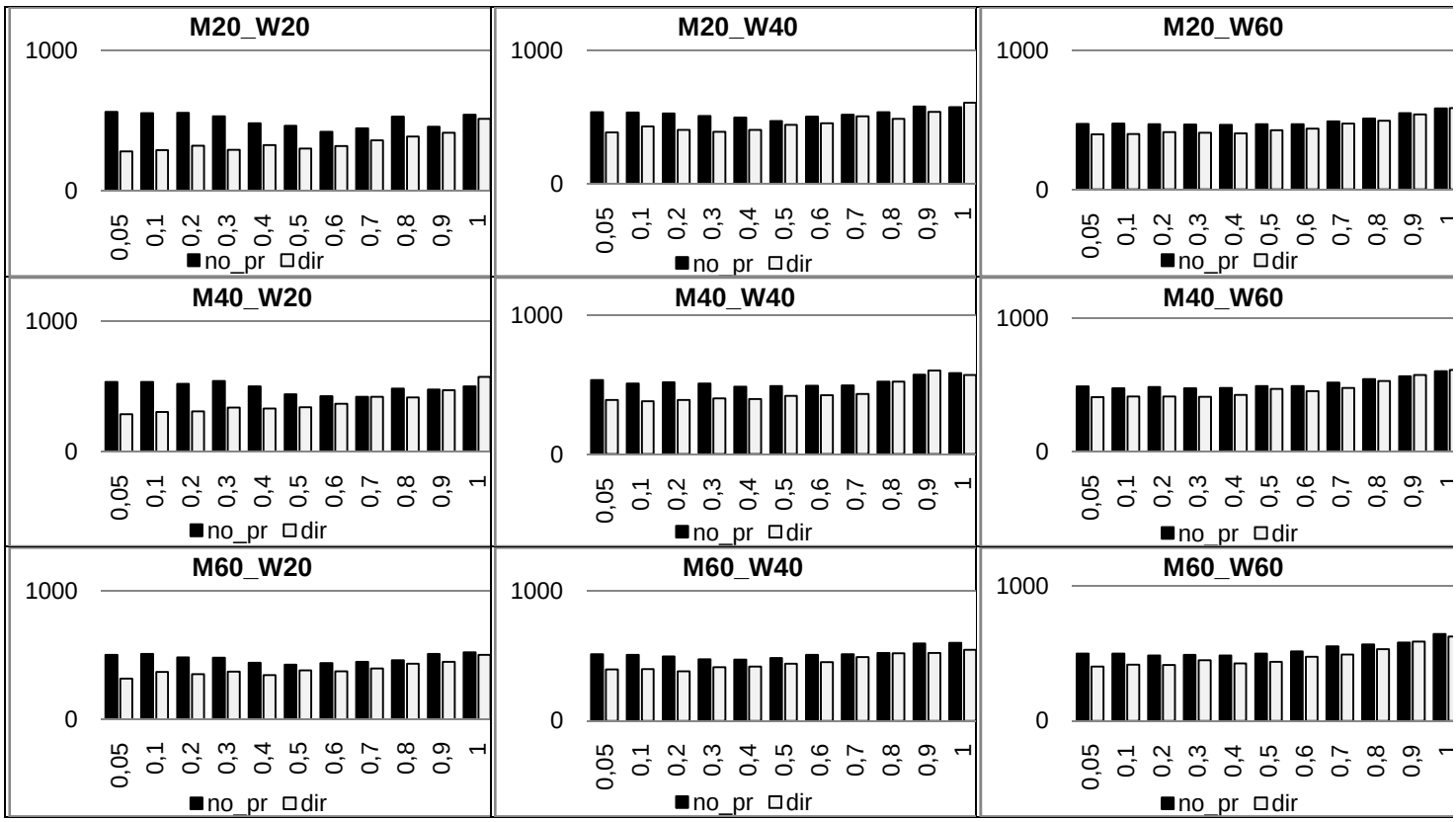


Figure 4.25. Average input waiting time of the base system with the directory priority (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic

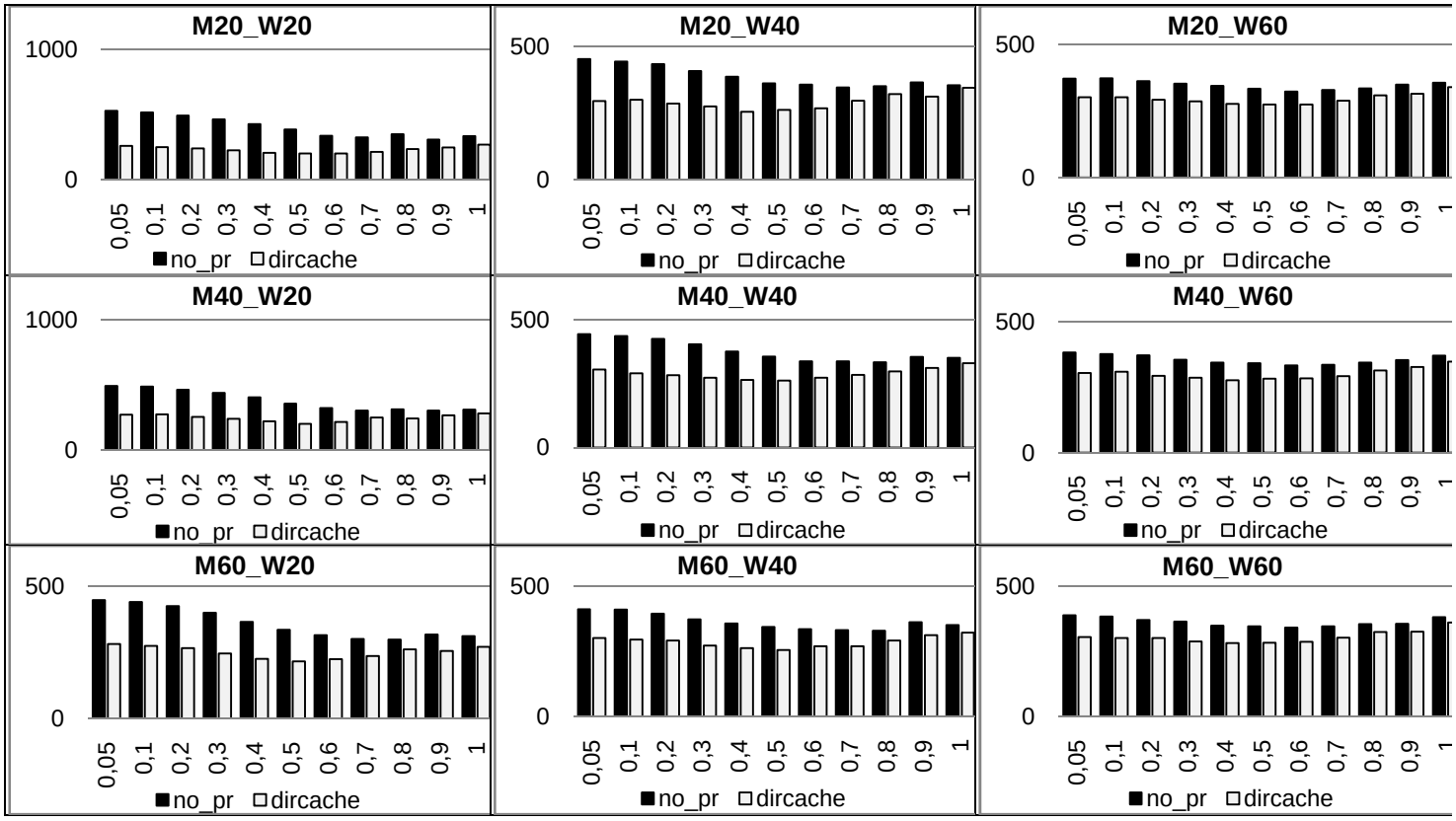


Figure 4.26. Average input waiting time of the base system with the directory & cache priority (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic

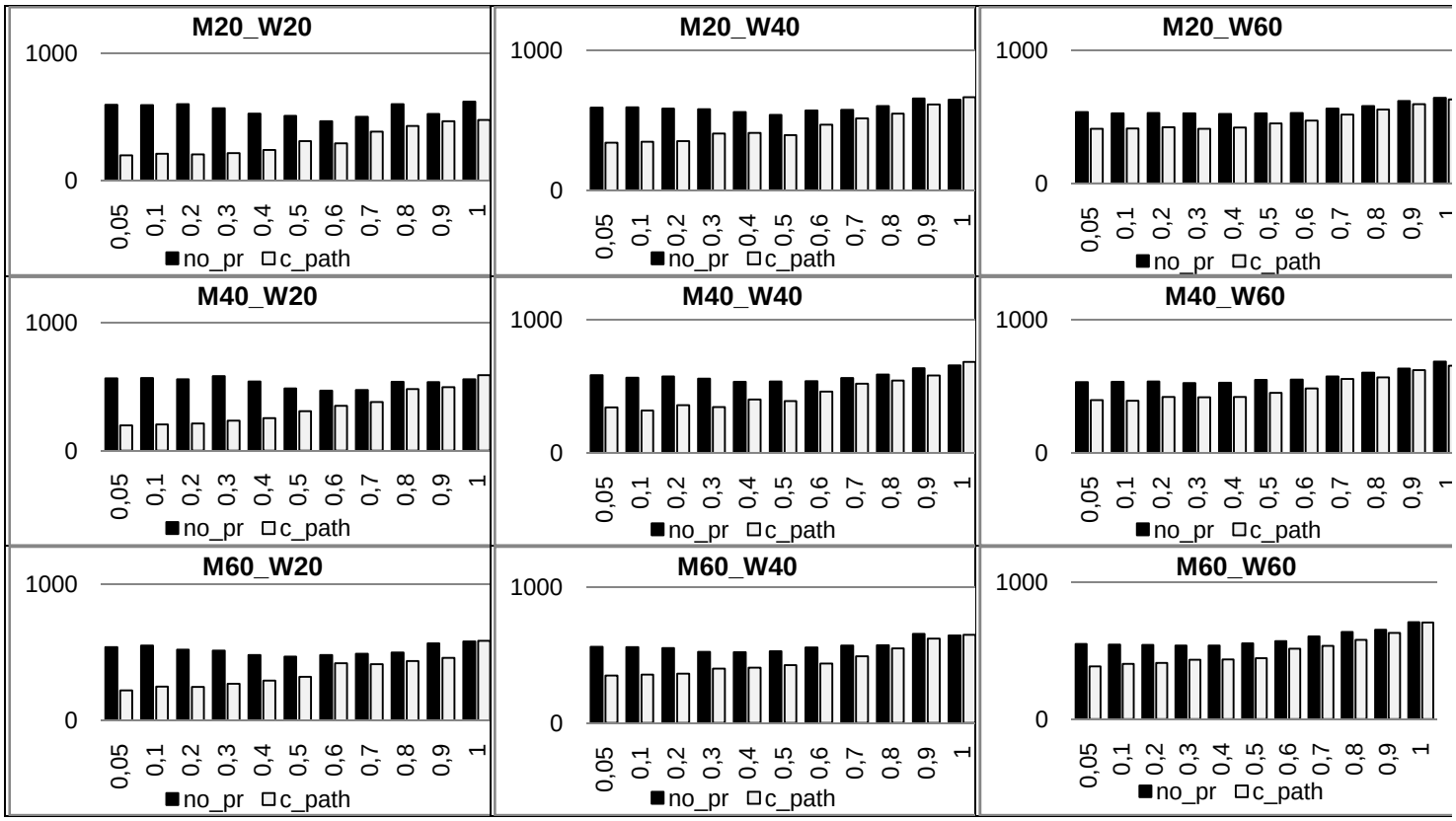


Figure 4.27. Average input waiting time of the base system with the critical path priority (vertical axis) for different values of T/R (horizontal axis), P(M) and P(W) under UN traffic

Table 4.7. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.2 under UN traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	60,52	60,77	60,89	46,21	48,53	47,96	32,83	29,30	26,63	12,39	13,43	13,67
0.1	56,02	55,76	58,79	47,41	47,27	48,80	27,04	28,50	30,81	12,90	14,52	13,37
0.2	54,74	53,06	55,04	48,19	48,08	47,59	26,94	25,29	26,87	17,92	17,73	18,64
0.3	49,52	52,77	51,41	43,71	42,62	42,57	20,23	26,49	21,13	19,53	18,03	17,87
0.4	39,13	43,80	41,66	41,27	37,08	39,65	21,28	24,48	24,15	16,26	16,78	16,25
0.5	28,78	34,78	37,18	34,14	33,86	36,23	24,62	23,25	23,86	15,89	18,94	19,58
0.6	26,05	27,61	32,41	21,82	29,16	31,67	16,63	19,04	23,10	18,58	20,21	21,35
0.7	20,38	25,77	27,72	22,92	27,12	27,81	18,74	21,93	21,68	18,36	19,85	24,18
0.8	20,37	26,79	28,55	23,72	25,51	28,24	18,99	20,54	26,48	20,08	23,97	25,27
0.9	19,58	21,14	25,29	18,97	22,54	26,20	16,13	21,22	21,01	15,52	22,11	22,12
1	17,00	20,02	23,61	16,24	21,73	23,74	15,97	19,99	22,88	15,96	20,13	21,20
average	**35,65**	**38,39**	**40,23**	**33,15**	**34,86**	**36,41**	**21,76**	**23,64**	**24,42**	**16,67**	**18,70**	**19,41**
maximum	**60,52**	**60,77**	**60,89**	**48,19**	**48,53**	**48,80**	**32,83**	**29,30**	**30,81**	**20,08**	**23,97**	**25,27**

Table 4.8. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.2 under UN traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	59,93	57,53	58,09	41,75	41,00	42,81	18,68	12,56	19,88	16,53	16,59	16,08
0.1	56,69	57,02	57,33	41,92	40,64	41,11	18,65	17,82	19,03	16,65	18,45	17,51
0.2	53,02	51,92	49,88	38,56	40,35	41,54	16,73	12,98	20,64	19,06	19,31	19,03
0.3	50,37	49,96	52,13	43,76	42,69	41,18	24,54	21,91	22,96	24,09	25,14	24,59
0.4	41,92	42,04	39,91	36,61	37,46	40,46	16,42	18,24	19,48	21,80	22,71	23,76
0.5	31,47	34,23	32,19	27,22	32,29	31,97	17,77	19,73	16,77	16,20	21,35	21,96
0.6	22,59	28,79	28,22	23,70	28,71	28,90	15,47	16,88	19,20	17,44	20,56	22,89
0.7	20,58	24,12	27,70	20,04	22,57	26,93	17,63	21,39	22,83	15,45	21,51	24,18
0.8	18,60	24,09	27,28	16,23	24,07	27,56	15,20	20,13	22,10	16,68	22,70	22,21
0.9	16,82	22,09	24,10	16,28	22,48	25,53	17,42	20,53	22,73	15,00	21,37	23,13
1	14,87	20,33	22,87	13,24	20,05	23,59	15,77	18,99	21,15	14,26	19,48	21,44
average	**35,17**	**37,47**	**38,15**	**29,03**	**32,03**	**33,78**	**17,66**	**18,29**	**20,62**	**17,56**	**20,83**	**21,53**
maximum	**59,93**	**57,53**	**58,09**	**43,76**	**42,69**	**42,81**	**24,54**	**21,91**	**22,96**	**24,09**	**25,14**	**24,59**

Table 4.9. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.2 under UN traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	52,23	48,60	47,81	35,95	34,54	29,15	12,38	6,52	10,53	19,92	20,18	20,95
0.1	52,70	48,22	54,55	36,78	33,08	34,56	9,81	12,46	14,98	21,82	22,61	22,30
0.2	49,00	44,31	42,41	31,87	31,49	32,33	5,69	9,66	11,21	16,72	16,93	17,92
0.3	41,13	36,57	42,56	32,32	33,92	32,85	11,74	12,16	11,13	19,51	20,91	20,14
0.4	34,44	38,72	37,34	25,89	27,36	26,26	11,38	7,36	6,05	14,49	19,44	20,22
0.5	27,99	32,14	32,01	24,92	27,59	29,43	11,01	16,16	16,45	20,52	21,49	22,13
0.6	23,46	22,99	25,76	20,64	25,65	27,02	11,27	15,69	21,05	17,01	21,81	22,47
0.7	19,43	22,47	27,22	18,38	23,53	25,89	16,21	19,72	19,07	16,66	23,26	20,11
0.8	16,40	22,59	24,22	16,86	22,99	22,83	15,40	19,74	21,80	14,86	20,13	21,68
0.9	17,60	22,21	25,85	18,01	21,92	25,57	16,70	21,39	22,84	18,79	22,37	23,91
1	17,18	24,08	22,92	17,25	23,58	25,20	15,66	20,45	23,89	14,71	20,41	23,06
average	**31,96**	**32,99**	**34,79**	**25,35**	**27,79**	**28,28**	**12,48**	**14,66**	**16,27**	**17,73**	**20,87**	**21,35**
maximum	**52,70**	**48,60**	**54,55**	**36,78**	**34,54**	**34,56**	**16,70**	**21,39**	**23,89**	**21,82**	**23,26**	**23,91**

Table 4.10. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.4 under UN traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	35,27	35,42	31,13	33,52	30,69	33,29	5,99	9,71	8,97	16,78	20,09	18,40
0.1	32,77	28,46	31,97	31,14	31,81	31,38	8,72	9,73	8,51	20,15	18,93	17,82
0.2	25,70	29,47	30,39	26,00	24,21	27,80	7,89	4,34	5,77	15,68	16,14	13,25
0.3	24,85	31,05	25,97	29,89	29,96	31,53	12,76	12,07	13,16	16,16	19,41	20,37
0.4	24,82	25,35	30,14	23,78	23,92	26,30	9,41	7,16	10,32	19,71	21,64	22,86
0.5	20,48	24,38	22,24	23,98	24,23	22,11	10,51	12,05	14,19	19,67	20,64	19,16
0.6	18,59	21,37	28,04	18,92	21,01	21,00	12,43	13,46	9,71	15,32	21,97	17,89
0.7	15,58	18,83	22,31	16,87	20,91	19,42	12,71	11,91	17,87	15,69	17,97	19,46
0.8	15,23	21,44	19,94	15,80	19,13	24,01	12,62	15,60	18,64	11,86	16,90	18,47
0.9	16,34	21,09	22,45	18,67	20,55	20,77	14,99	19,99	21,67	16,55	18,95	23,07
1	13,39	18,67	20,02	11,07	19,24	21,10	12,31	16,69	19,77	12,79	16,91	19,79
average	**22,09**	**25,05**	**25,87**	**22,69**	**24,15**	**25,34**	**10,94**	**12,06**	**13,51**	**16,40**	**19,05**	**19,14**
maximum	**35,27**	**35,42**	**31,97**	**33,52**	**31,81**	**33,29**	**14,99**	**19,99**	**21,67**	**20,15**	**21,97**	**23,07**

Table 4.11. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.4 under UN traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	37,31	36,72	36,57	28,69	33,16	29,11	8,78	8,31	10,92	18,92	21,55	19,06
0.1	34,92	35,89	33,97	29,10	30,91	28,45	7,01	6,59	8,04	16,05	15,56	18,70
0.2	30,74	32,53	31,73	28,54	28,92	25,78	8,45	4,33	4,97	20,47	19,38	19,29
0.3	29,52	29,48	31,01	28,19	23,84	27,83	9,19	9,00	10,13	16,57	19,56	17,33
0.4	22,29	26,47	26,54	24,24	23,21	22,45	7,55	10,09	9,91	17,89	15,58	17,25
0.5	20,20	27,12	25,69	21,57	23,23	23,77	11,26	13,30	11,48	19,38	18,99	16,25
0.6	15,60	21,82	22,07	18,24	17,97	24,86	12,16	14,75	16,78	15,48	17,68	21,42
0.7	17,14	24,41	22,26	17,12	20,27	22,20	14,27	14,60	18,46	17,73	20,17	18,88
0.8	16,23	20,85	21,42	16,85	21,81	21,95	15,27	17,07	18,80	17,04	20,50	19,99
0.9	15,40	20,72	23,43	17,37	20,04	23,37	17,25	19,89	19,43	15,54	21,79	22,91
1	14,55	22,75	20,01	17,51	19,34	20,01	14,39	19,58	20,96	15,65	18,96	21,77
average	**23,08**	**27,16**	**26,79**	**22,49**	**23,88**	**24,53**	**11,42**	**12,50**	**13,63**	**17,34**	**19,07**	**19,35**
maximum	**37,31**	**36,72**	**36,57**	**29,10**	**33,16**	**29,11**	**17,25**	**19,89**	**20,96**	**20,47**	**21,79**	**22,91**

Table 4.12. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.4 under UN traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	31,18	33,91	30,62	25,99	26,38	25,32	6,10	4,94	11,17	21,21	20,32	19,82
0.1	29,66	29,14	28,90	20,34	24,49	23,01	6,42	0,70	6,94	18,16	16,05	17,58
0.2	30,23	30,03	31,67	23,58	17,88	23,83	7,37	5,85	2,63	18,36	17,16	16,98
0.3	21,53	27,42	24,09	22,28	19,97	20,89	7,74	5,45	5,70	14,57	14,23	16,91
0.4	22,00	23,37	24,88	17,62	20,38	23,88	6,95	2,33	4,61	16,78	16,81	16,27
0.5	18,98	21,34	21,92	19,17	20,27	20,94	7,53	10,37	12,23	15,47	16,83	18,68
0.6	18,13	24,24	22,63	19,36	25,31	24,63	14,05	14,52	16,28	15,68	18,95	21,14
0.7	14,65	21,65	23,99	16,39	20,93	22,85	13,45	14,32	17,90	16,52	19,21	20,24
0.8	15,49	19,65	22,25	16,53	18,81	20,39	13,31	16,03	17,86	14,93	18,36	18,44
0.9	17,63	21,55	23,45	16,63	22,17	24,23	16,40	20,19	21,63	17,10	22,19	21,41
1	15,49	19,29	21,18	13,39	19,56	20,73	13,72	20,19	19,30	15,84	18,71	22,70
average	**21,36**	**24,69**	**25,05**	**19,21**	**21,47**	**22,79**	**10,28**	**10,44**	**12,39**	**16,79**	**18,07**	**19,11**
maximum	**31,18**	**33,91**	**31,67**	**25,99**	**26,38**	**25,32**	**16,40**	**20,19**	**21,63**	**21,21**	**22,19**	**22,70**

Table 4.13. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.6 under UN traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	17,25	14,44	17,98	18,33	19,09	18,88	4,23	2,64	4,49	13,11	14,55	14,60
0.1	15,55	15,03	13,11	17,97	14,94	16,13	1,40	0,53	2,89	13,02	14,41	12,71
0.2	16,33	19,49	18,35	18,36	19,84	18,31	2,92	3,95	5,37	13,85	14,77	12,96
0.3	15,53	16,30	14,79	13,21	18,83	18,95	3,85	4,58	4,51	12,26	14,08	14,89
0.4	15,40	18,69	18,09	14,82	17,23	18,46	5,93	3,43	6,74	14,15	15,23	15,13
0.5	14,43	16,63	16,28	17,21	18,17	18,54	3,79	9,92	6,77	15,94	15,43	15,85
0.6	13,86	18,30	19,81	16,27	18,58	19,35	10,69	12,24	11,58	15,34	16,50	19,33
0.7	16,43	20,31	21,40	18,85	20,58	21,21	12,78	15,10	16,46	16,44	19,17	19,74
0.8	16,28	19,93	21,40	15,08	19,81	19,65	12,84	16,94	16,00	15,17	19,84	20,95
0.9	18,97	22,81	21,62	19,53	20,22	21,40	17,57	18,41	17,93	15,03	21,43	20,88
1	17,59	20,03	23,00	16,65	21,91	24,18	15,65	19,80	21,21	15,49	20,93	21,59
average	**16,15**	**18,36**	**18,71**	**16,93**	**19,02**	**19,55**	**8,33**	**9,78**	**10,36**	**14,53**	**16,94**	**17,15**
maximum	**18,97**	**22,81**	**23,00**	**19,53**	**21,91**	**24,18**	**17,57**	**19,80**	**21,21**	**16,44**	**21,43**	**21,59**

Table 4.14. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.6 under UN traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	18,10	18,30	16,05	17,81	18,48	17,23	-1,51	-0,09	1,96	11,75	12,87	13,80
0.1	17,18	16,21	17,50	15,88	17,96	19,81	3,38	0,28	0,63	14,76	11,83	13,01
0.2	13,64	11,32	18,62	16,14	18,37	15,61	1,72	0,41	-0,07	11,17	12,62	10,84
0.3	13,04	17,15	14,61	16,23	18,37	16,95	1,26	4,74	2,42	13,81	11,35	13,36
0.4	15,55	16,94	16,70	15,97	12,48	17,20	1,66	3,72	5,15	9,02	14,10	14,29
0.5	17,08	20,49	18,23	19,75	20,09	19,85	8,64	9,39	7,88	15,82	18,08	17,22
0.6	15,06	19,29	21,98	17,05	17,99	19,70	12,92	10,34	13,84	13,51	17,55	17,08
0.7	16,42	20,53	18,46	17,88	20,87	19,12	12,17	16,57	16,10	17,16	19,15	20,56
0.8	17,96	18,64	19,95	17,87	22,25	21,51	14,79	18,27	18,08	15,70	18,35	20,52
0.9	15,81	20,37	21,88	17,70	20,01	21,60	15,44	17,93	18,40	16,37	19,59	20,34
1	16,40	22,18	21,21	16,79	23,27	23,12	15,36	17,90	19,78	16,11	20,49	21,78
average	**16,02**	**18,31**	**18,65**	**17,19**	**19,11**	**19,25**	**7,80**	**9,04**	**9,47**	**14,11**	**16,00**	**16,62**
maximum	**18,10**	**22,18**	**21,98**	**19,75**	**23,27**	**23,12**	**15,44**	**18,27**	**19,78**	**17,16**	**20,49**	**21,78**

Table 4.15. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.6 under UN traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	23,00	22,19	20,70	17,92	19,22	18,76	-0,25	1,98	4,56	15,72	13,73	15,37
0.1	20,90	20,93	21,12	18,12	18,31	19,76	2,00	2,95	0,51	18,05	15,32	15,10
0.2	21,96	19,08	20,99	17,84	19,96	17,66	4,54	0,49	1,59	14,29	14,61	14,55
0.3	18,37	14,44	16,79	18,08	17,41	16,07	-0,90	1,41	1,73	12,96	14,07	13,88
0.4	17,81	20,24	19,45	17,14	15,70	17,66	3,82	4,12	3,01	12,73	15,84	11,63
0.5	17,90	15,48	18,82	14,96	18,26	18,81	5,71	5,81	8,42	13,83	14,86	17,13
0.6	16,38	21,18	20,20	17,40	20,45	20,80	12,72	13,64	12,89	13,93	18,06	18,45
0.7	19,73	22,58	22,09	19,34	21,08	21,36	12,89	16,72	16,53	16,67	18,86	19,98
0.8	16,96	21,21	20,83	14,95	20,45	20,61	12,79	16,74	19,18	15,84	18,66	19,76
0.9	15,74	21,10	22,10	17,10	20,17	20,66	13,25	17,60	18,63	14,64	19,15	19,79
1	15,12	20,07	20,81	16,88	19,04	21,14	15,39	17,79	18,93	15,14	18,41	18,98
average	**18,53**	**19,86**	**20,35**	**17,25**	**19,10**	**19,39**	**7,45**	**9,02**	**9,63**	**14,89**	**16,50**	**16,78**
maximum	**23,00**	**22,58**	**22,10**	**19,34**	**21,08**	**21,36**	**15,39**	**17,79**	**19,18**	**18,05**	**19,15**	**19,98**

Table 4.16. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.2 under HR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	38,76	38,15	41,69	40,51	43,41	35,55	4,50	17,62	8,64	13,66	18,06	16,02
0.1	43,06	45,97	35,97	43,07	37,24	44,59	12,55	9,11	11,88	14,18	10,31	16,02
0.2	40,46	39,40	32,30	35,67	39,46	37,12	15,85	16,86	17,47	15,19	14,27	14,31
0.3	28,06	27,93	29,11	28,02	24,47	28,31	12,00	17,23	14,79	18,93	20,56	19,08
0.4	17,01	21,43	21,99	18,08	21,94	23,83	14,18	22,93	19,45	13,48	18,65	19,83
0.5	14,64	19,64	18,75	12,60	18,37	20,83	14,79	19,90	20,58	11,74	15,98	18,17
0.6	14,90	17,85	19,88	14,22	20,32	19,74	15,90	19,03	20,65	13,02	15,77	20,94
0.7	12,13	16,08	17,77	13,92	15,90	20,03	12,88	17,88	19,13	13,27	16,96	18,31
0.8	10,13	14,25	18,34	11,80	15,18	15,50	11,49	15,14	16,92	12,27	13,86	15,88
0.9	10,97	16,27	18,25	11,72	15,62	18,07	12,92	14,34	16,56	12,06	15,48	16,75
1	9,64	13,11	14,94	11,31	14,87	15,36	10,29	14,75	15,30	11,27	13,95	15,25
average	**21,80**	**24,55**	**24,45**	**21,90**	**24,25**	**25,36**	**12,49**	**16,80**	**16,49**	**13,55**	**15,81**	**17,32**
maximum	**43,06**	**45,97**	**41,69**	**43,07**	**43,41**	**44,59**	**15,90**	**22,93**	**20,65**	**18,93**	**20,56**	**20,94**

Table 4.17. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.2 under HR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	42,69	36,93	34,02	31,64	31,91	29,34	8,43	3,69	6,91	13,62	11,87	16,09
0.1	33,40	44,53	35,01	33,77	31,10	35,27	9,58	7,78	9,61	17,74	16,71	20,25
0.2	29,13	35,46	31,39	34,21	27,77	34,94	11,42	16,13	12,85	12,05	20,21	22,26
0.3	26,97	25,15	26,35	24,10	27,14	25,49	8,91	9,37	8,72	17,31	18,13	12,35
0.4	22,05	25,37	25,88	22,83	24,24	27,13	15,62	19,74	20,94	15,81	17,24	19,91
0.5	17,41	20,10	22,00	11,07	18,95	21,87	15,32	18,82	20,72	13,97	17,77	19,06
0.6	14,91	19,16	22,53	14,83	19,81	24,93	14,97	19,83	21,65	16,30	19,04	22,53
0.7	12,94	18,35	20,60	14,35	18,53	19,19	11,17	18,85	19,45	11,51	17,00	16,69
0.8	13,34	18,75	19,39	13,34	17,90	21,92	14,70	16,98	19,16	12,60	19,62	15,56
0.9	13,55	18,36	19,52	14,36	20,35	20,13	13,41	18,43	19,32	15,07	16,53	18,95
1	10,65	13,78	18,19	10,09	18,53	16,52	12,82	16,04	17,45	10,67	15,09	18,27
average	**21,55**	**25,09**	**24,99**	**20,42**	**23,29**	**25,16**	**12,40**	**15,06**	**16,07**	**14,24**	**17,20**	**18,36**
maximum	**42,69**	**44,53**	**35,01**	**34,21**	**31,91**	**35,27**	**15,62**	**19,83**	**21,65**	**17,74**	**20,21**	**22,53**

Table 4.18. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.2 under HR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	37,89	42,47	40,00	29,86	29,04	30,27	7,20	5,85	7,42	23,48	17,43	16,84
0.1	34,92	40,94	38,80	25,66	25,50	25,93	4,60	7,23	7,26	18,42	21,73	16,86
0.2	27,05	33,58	29,64	25,04	23,94	24,80	8,54	3,14	5,90	14,96	16,42	19,30
0.3	28,75	27,91	24,71	25,71	26,01	21,84	6,94	8,80	5,99	14,60	17,55	18,87
0.4	23,91	24,28	23,24	20,78	20,49	24,31	9,93	13,84	12,42	13,88	12,71	15,65
0.5	20,33	23,64	24,25	19,86	22,24	23,69	18,75	17,20	18,49	18,01	20,14	19,42
0.6	14,66	17,20	21,97	16,02	21,62	23,70	15,77	18,23	19,47	16,75	18,60	20,29
0.7	15,59	22,85	24,72	16,21	22,15	20,88	15,49	23,80	23,47	17,71	17,62	20,00
0.8	18,09	21,51	26,88	15,59	24,42	23,41	17,69	19,84	24,61	16,62	21,51	24,06
0.9	14,72	18,93	22,15	14,48	20,52	22,46	15,36	20,17	24,40	12,80	18,74	20,87
1	18,38	20,78	24,44	17,87	21,17	22,67	16,19	23,65	24,74	15,07	24,50	19,82
average	**23,12**	**26,74**	**27,34**	**20,64**	**23,37**	**24,00**	**12,41**	**14,70**	**15,83**	**16,57**	**18,81**	**19,27**
maximum	**37,89**	**42,47**	**40,00**	**29,86**	**29,04**	**30,27**	**18,75**	**23,80**	**24,74**	**23,48**	**24,50**	**24,06**

Table 4.19. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.4 under HR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	28,61	26,49	23,56	29,31	31,97	30,20	9,72	5,51	5,13	17,43	14,85	20,21
0.1	24,24	20,25	25,54	28,01	26,51	24,21	5,83	8,55	4,41	13,78	17,54	13,99
0.2	22,36	22,57	27,46	23,43	28,44	24,59	6,17	10,25	4,45	16,49	15,33	13,93
0.3	16,55	20,14	20,28	17,74	21,29	22,84	7,02	9,74	8,74	10,91	14,52	13,38
0.4	15,53	19,45	18,24	17,91	18,67	23,57	13,99	17,66	15,25	15,55	14,84	20,03
0.5	11,77	17,71	18,65	14,25	18,17	21,81	11,00	13,87	18,58	12,61	15,82	20,29
0.6	15,51	18,15	22,53	14,59	21,14	24,01	13,42	17,62	22,01	16,22	17,06	18,67
0.7	9,32	17,49	20,74	13,56	17,06	19,84	11,54	15,21	16,58	10,43	14,99	16,20
0.8	13,59	16,22	19,48	12,75	18,38	19,53	11,74	14,41	20,30	12,00	15,59	17,43
0.9	10,88	17,43	18,53	10,47	17,83	18,72	13,11	14,81	18,48	11,22	16,48	17,64
1	12,03	13,61	17,06	9,26	14,76	16,90	11,45	15,87	18,03	9,94	13,66	17,74
average	**16,40**	**19,05**	**21,10**	**17,39**	**21,29**	**22,38**	**10,45**	**13,04**	**13,81**	**13,33**	**15,52**	**17,23**
maximum	**28,61**	**26,49**	**27,46**	**29,31**	**31,97**	**30,20**	**13,99**	**17,66**	**22,01**	**17,43**	**17,54**	**20,29**

Table 4.20. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.4 under HR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	20,66	24,12	25,59	28,41	24,53	27,72	4,12	4,29	4,31	18,36	17,93	15,56
0.1	20,62	25,04	24,32	24,33	21,27	25,05	2,77	1,63	2,68	14,83	10,66	15,43
0.2	19,16	22,05	23,92	24,85	24,74	24,60	3,67	4,78	5,24	14,75	17,83	15,15
0.3	20,96	23,66	20,00	21,21	21,49	23,45	7,26	6,63	8,09	14,26	18,19	16,58
0.4	19,05	20,25	19,99	19,55	23,91	24,05	12,86	11,79	13,91	14,17	15,55	19,06
0.5	12,67	16,38	17,96	14,51	17,23	21,51	11,13	15,52	15,28	9,48	14,82	16,48
0.6	14,93	20,85	23,39	17,79	22,87	24,60	15,54	19,25	18,86	14,07	20,25	21,01
0.7	13,44	15,60	17,83	13,61	18,59	18,39	14,18	17,03	19,70	12,60	16,42	20,32
0.8	14,31	17,41	20,52	13,54	17,82	19,78	15,72	15,97	20,09	14,11	15,45	20,07
0.9	10,23	15,94	20,15	13,98	19,05	16,25	12,13	18,00	19,71	10,84	18,10	20,28
1	13,07	19,56	19,98	14,47	17,46	18,45	13,00	15,06	18,76	11,41	15,32	16,81
average	**16,28**	**20,08**	**21,24**	**18,75**	**20,81**	**22,17**	**10,22**	**11,81**	**13,33**	**13,54**	**16,41**	**17,89**
maximum	**20,96**	**25,04**	**25,59**	**28,41**	**24,74**	**27,72**	**15,72**	**19,25**	**20,09**	**18,36**	**20,25**	**21,01**

Table 4.21. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.4 under HR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	25,39	26,48	22,29	22,18	24,84	24,49	3,60	0,81	3,67	16,56	15,91	14,23
0.1	23,39	24,63	21,81	22,47	24,78	23,86	1,09	0,22	2,55	17,86	17,61	16,20
0.2	22,23	22,13	24,40	21,22	22,10	22,95	3,14	2,59	0,98	16,40	15,23	16,56
0.3	20,17	19,21	18,07	22,63	19,03	19,25	6,33	0,28	1,51	13,57	14,16	14,35
0.4	17,07	17,45	18,66	16,86	16,54	20,63	7,41	8,61	7,36	12,96	15,61	17,45
0.5	17,72	20,18	20,73	19,46	19,89	20,40	13,26	15,86	15,18	15,80	19,80	17,72
0.6	15,92	19,11	21,39	17,69	21,96	20,02	11,92	16,23	19,80	14,92	19,34	18,17
0.7	13,37	19,84	20,07	11,96	18,77	15,39	12,38	15,76	16,30	12,92	19,11	19,72
0.8	19,96	22,60	25,68	18,34	24,20	24,81	17,33	24,03	24,34	18,99	24,15	23,09
0.9	12,72	17,03	19,53	12,73	20,11	18,91	11,50	18,05	18,95	10,49	16,76	19,00
1	9,02	17,39	16,66	10,93	15,62	18,37	8,76	14,55	19,87	9,09	17,05	14,35
average	**17,90**	**20,55**	**20,84**	**17,86**	**20,71**	**20,83**	**8,79**	**10,64**	**11,86**	**14,51**	**17,70**	**17,35**
maximum	**25,39**	**26,48**	**25,68**	**22,63**	**24,84**	**24,81**	**17,33**	**24,03**	**24,34**	**18,99**	**24,15**	**23,09**

Table 4.22. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.6 under HR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	15,52	14,18	19,79	23,64	22,46	18,92	1,24	-4,20	1,18	16,41	15,60	15,53
0.1	16,64	14,98	15,36	23,68	21,74	19,85	5,59	2,57	-0,13	16,99	16,09	16,95
0.2	14,29	15,20	15,72	18,86	18,42	20,06	2,06	2,37	-0,62	11,06	12,62	14,47
0.3	13,18	14,82	15,07	17,07	15,61	18,63	4,58	6,12	8,36	13,18	16,63	16,11
0.4	10,41	13,23	17,23	14,40	16,15	18,35	6,91	6,76	4,56	12,90	13,08	10,06
0.5	17,23	17,93	19,08	15,74	18,10	22,99	13,05	14,87	15,07	15,65	18,18	18,67
0.6	12,25	15,01	16,90	12,79	19,22	19,37	11,42	14,27	15,84	12,21	18,36	18,00
0.7	9,67	16,23	16,90	11,94	16,02	17,28	11,27	14,76	17,06	11,41	14,05	17,69
0.8	11,15	15,53	18,85	12,56	18,84	16,44	13,39	18,15	16,71	12,19	16,30	18,89
0.9	13,49	17,90	22,28	15,93	19,38	22,70	15,03	18,49	20,56	15,40	18,09	20,56
1	12,98	20,24	18,69	15,74	16,73	20,13	13,60	18,64	21,07	11,39	18,63	20,00
average	**13,35**	**15,93**	**17,81**	**16,58**	**18,42**	**19,52**	**8,92**	**10,25**	**10,88**	**13,53**	**16,15**	**16,99**
maximum	**17,23**	**20,24**	**22,28**	**23,68**	**22,46**	**22,99**	**15,03**	**18,64**	**21,07**	**16,99**	**18,63**	**20,56**

Table 4.23. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.6 under HR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	20,19	14,20	16,48	17,91	18,80	21,16	-2,84	0,16	-0,90	15,49	12,84	12,52
0.1	17,89	14,84	16,62	20,13	20,53	19,19	-0,54	1,90	0,03	11,78	15,55	15,50
0.2	21,98	15,83	18,20	23,14	24,34	19,13	8,43	7,68	4,89	14,90	16,05	15,56
0.3	11,94	17,24	13,46	15,13	21,34	19,75	4,33	5,16	1,05	15,11	14,14	13,37
0.4	9,70	15,66	16,44	15,90	15,78	18,11	7,64	7,06	5,23	11,20	13,14	14,94
0.5	15,61	14,75	14,15	11,62	16,74	15,72	8,05	11,18	11,26	7,77	13,58	16,93
0.6	13,35	14,93	15,18	12,89	16,45	20,15	10,87	12,85	16,31	10,90	16,93	15,82
0.7	13,65	16,41	19,14	15,29	14,46	20,16	13,43	18,22	18,41	12,26	15,82	18,89
0.8	10,11	16,56	17,89	12,25	18,28	18,50	13,59	14,51	16,32	12,96	17,02	16,11
0.9	15,83	19,74	20,91	16,33	18,26	22,57	15,27	17,97	20,10	16,40	19,31	19,96
1	14,82	17,53	22,75	16,98	20,31	22,78	16,09	20,28	21,08	17,58	16,40	21,61
average	**15,01**	**16,15**	**17,38**	**16,14**	**18,66**	**19,75**	**8,57**	**10,63**	**10,34**	**13,31**	**15,53**	**16,47**
maximum	**21,98**	**19,74**	**22,75**	**23,14**	**24,34**	**22,78**	**16,09**	**20,28**	**21,08**	**17,58**	**19,31**	**21,61**

Table 4.24. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.6 under HR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	20,46	20,92	21,38	23,93	19,97	19,84	-0,42	-0,02	0,52	16,96	16,21	15,83
0.1	13,74	14,22	18,77	22,66	20,88	17,94	-0,56	-2,21	-0,34	12,24	15,08	14,71
0.2	15,77	15,16	14,77	18,78	22,55	16,77	-2,05	-0,85	2,32	11,51	13,23	14,19
0.3	12,57	14,22	15,72	15,37	15,03	13,84	-0,96	-2,18	-0,30	9,21	10,54	9,22
0.4	14,39	15,47	19,50	17,11	19,59	18,92	8,89	6,71	8,15	14,22	11,91	14,19
0.5	12,10	15,67	12,79	12,18	13,81	17,36	9,73	7,86	10,38	12,10	12,43	12,20
0.6	12,98	16,24	15,06	13,14	17,53	17,99	11,47	12,00	13,40	11,41	15,00	16,55
0.7	13,16	14,55	17,90	13,29	13,57	14,07	9,91	12,11	13,00	8,28	13,24	13,15
0.8	14,42	17,81	21,61	15,08	21,57	19,27	14,47	18,81	17,03	14,48	17,79	19,82
0.9	17,33	19,91	21,32	15,23	20,56	20,62	18,00	18,43	19,05	13,44	18,37	20,61
1	12,73	17,86	19,49	16,92	17,61	21,09	14,31	18,07	18,67	13,23	17,93	19,14
average	**14,51**	**16,55**	**18,03**	**16,70**	**18,42**	**17,97**	**7,53**	**8,07**	**9,26**	**12,46**	**14,70**	**15,42**
maximum	**20,46**	**20,92**	**21,61**	**23,93**	**22,55**	**21,09**	**18,00**	**18,81**	**19,05**	**16,96**	**18,37**	**20,61**

Table 4.25. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.2 under BR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	60,88	60,88	59,79	38,09	39,38	38,28	43,06	42,81	44,76	7,43	8,23	7,44
0.1	51,44	55,66	57,47	40,00	41,68	42,29	40,11	43,41	42,63	13,89	14,64	15,56
0.2	14,51	17,82	21,19	14,45	17,48	20,44	14,11	19,21	21,16	13,27	18,37	19,91
0.3	10,36	14,31	16,35	10,52	15,07	15,85	10,12	13,86	16,60	9,81	13,31	16,35
0.4	11,10	13,03	16,29	10,54	12,39	15,95	7,98	14,13	15,83	11,62	13,07	16,35
0.5	8,09	13,51	15,17	8,26	11,84	13,64	9,50	11,59	12,22	10,44	11,97	13,07
0.6	9,38	13,78	15,36	7,76	13,83	15,45	8,74	13,55	14,93	10,26	11,91	15,90
0.7	5,62	10,02	14,60	8,07	10,28	11,25	6,95	8,61	14,90	8,94	11,17	12,27
0.8	10,37	13,13	14,57	8,79	8,65	12,69	9,23	10,97	16,80	6,03	12,18	14,12
0.9	5,36	14,25	12,22	6,36	10,49	13,35	10,10	9,40	12,85	7,90	13,07	13,04
1	12,45	12,66	12,62	8,46	15,64	15,98	10,15	11,82	12,55	10,75	15,45	16,84
average	**18,14**	**21,73**	**23,24**	**14,66**	**17,88**	**19,56**	**15,46**	**18,12**	**20,48**	**10,03**	**13,03**	**14,62**
maximum	**60,88**	**60,88**	**59,79**	**40,00**	**41,68**	**42,29**	**43,06**	**43,41**	**44,76**	**13,89**	**18,37**	**19,91**

Table 4.26. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.2 under BR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	62,80	64,19	63,76	38,83	40,00	39,50	37,45	37,95	37,48	11,10	10,94	11,30
0.1	57,55	58,79	59,29	38,43	38,83	39,30	34,85	36,65	37,40	12,57	12,21	12,05
0.2	15,40	22,81	24,08	16,79	21,37	25,33	15,81	21,16	23,92	13,25	19,77	22,47
0.3	13,11	17,38	18,38	12,50	15,84	17,26	13,94	16,83	18,73	11,00	13,91	16,43
0.4	12,14	14,58	16,99	12,13	14,43	16,05	11,30	14,78	17,97	9,63	13,65	15,95
0.5	14,34	15,09	18,05	13,59	14,47	18,24	12,16	16,48	19,22	12,02	14,46	16,74
0.6	9,35	11,49	15,79	7,43	12,27	16,04	8,74	15,03	13,15	7,87	12,10	13,70
0.7	10,00	15,74	18,07	10,56	16,48	17,03	12,22	14,70	18,01	11,79	14,41	14,70
0.8	14,63	20,39	19,17	14,37	16,59	16,78	13,79	17,75	19,35	14,55	18,32	19,69
0.9	11,86	15,68	15,04	12,42	12,38	16,35	10,96	14,57	19,07	12,98	16,62	15,49
1	12,30	16,50	18,05	14,02	16,30	17,75	11,63	14,75	16,61	10,50	17,02	15,63
average	**21,23**	**24,79**	**26,06**	**17,37**	**19,91**	**21,78**	**16,62**	**20,06**	**21,90**	**11,57**	**14,85**	**15,83**
maximum	**62,80**	**64,19**	**63,76**	**38,83**	**40,00**	**39,50**	**37,45**	**37,95**	**37,48**	**14,55**	**19,77**	**22,47**

Table 4.27. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.2 under BR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	63,16	62,27	63,00	37,29	37,50	37,62	26,97	28,50	25,48	14,51	14,96	13,94
0.1	59,55	59,98	59,03	35,96	36,69	36,09	26,70	27,64	25,91	13,92	14,26	13,40
0.2	29,75	36,93	43,13	26,99	39,09	41,21	25,30	34,17	38,57	22,45	28,74	33,49
0.3	19,12	23,59	26,83	18,47	24,26	26,78	17,55	23,92	26,31	15,83	20,43	22,45
0.4	16,87	21,56	23,15	16,45	22,28	23,75	18,21	21,83	24,48	14,16	18,86	21,25
0.5	15,00	20,14	20,01	14,50	17,81	21,04	12,08	18,70	21,53	11,21	15,79	19,95
0.6	13,48	18,88	20,07	15,14	19,56	20,90	14,41	17,23	21,05	11,92	18,11	20,43
0.7	13,17	15,08	16,44	10,85	14,65	18,15	9,87	13,84	16,62	9,96	14,80	15,00
0.8	15,10	19,01	20,15	12,48	18,70	21,10	15,17	18,79	19,64	13,03	16,49	20,13
0.9	14,71	19,77	21,10	13,79	20,95	18,95	13,85	20,14	22,68	12,60	16,51	19,52
1	15,43	20,35	20,93	16,29	22,05	22,08	16,59	20,11	24,54	16,40	18,38	19,73
average	**25,03**	**28,87**	**30,35**	**19,84**	**24,87**	**26,15**	**17,88**	**22,26**	**24,25**	**14,18**	**17,94**	**19,94**
maximum	**63,16**	**62,27**	**63,00**	**37,29**	**39,09**	**41,21**	**26,97**	**34,17**	**38,57**	**22,45**	**28,74**	**33,49**

Table 4.28. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.4 under BR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	48,15	48,78	48,91	29,18	31,06	30,22	23,10	23,85	24,61	12,37	12,29	12,58
0.1	44,15	46,80	46,94	31,00	32,63	31,59	23,02	22,96	25,03	14,29	14,96	14,93
0.2	15,98	20,78	23,93	16,04	20,97	23,35	17,22	21,66	23,67	15,09	19,00	21,48
0.3	13,42	18,27	20,14	13,27	17,60	19,00	13,11	16,79	19,04	12,46	17,12	18,51
0.4	12,74	18,69	19,93	12,26	17,47	19,53	14,17	17,98	20,54	13,82	17,23	18,64
0.5	10,54	16,17	18,71	11,20	15,32	18,29	12,34	15,17	19,02	11,18	14,05	16,27
0.6	10,14	17,85	17,70	14,99	16,80	19,49	10,38	16,58	17,53	11,64	15,94	19,10
0.7	12,72	14,57	16,34	13,83	13,17	16,65	12,12	14,29	20,21	8,13	14,88	16,73
0.8	12,79	15,44	18,45	9,71	14,32	15,47	8,60	12,79	15,90	12,50	15,71	15,92
0.9	12,00	17,98	18,22	10,42	16,31	15,20	12,76	15,81	19,65	6,50	15,09	19,16
1	14,35	19,50	21,22	12,41	18,68	20,06	13,26	17,74	17,63	11,93	19,36	21,62
average	**18,82**	**23,17**	**24,59**	**15,85**	**19,48**	**20,80**	**14,55**	**17,78**	**20,26**	**11,81**	**15,97**	**17,72**
maximum	**48,15**	**48,78**	**48,91**	**31,00**	**32,63**	**31,59**	**23,10**	**23,85**	**25,03**	**15,09**	**19,36**	**21,62**

Table 4.29. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.4 under BR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	50,82	52,05	52,51	31,14	31,70	32,49	23,72	21,63	27,14	11,75	11,97	11,70
0.1	48,27	49,90	50,42	31,55	33,13	31,79	24,28	26,26	25,04	13,77	13,91	13,46
0.2	20,63	24,44	28,24	19,53	25,99	27,68	19,27	24,88	27,75	16,97	24,11	25,22
0.3	13,96	17,91	20,31	11,68	18,06	20,30	13,25	18,74	18,89	11,83	16,71	17,67
0.4	12,72	16,99	18,40	10,77	15,64	16,78	12,06	17,09	18,87	9,39	14,80	17,56
0.5	12,78	15,39	19,05	11,56	15,34	17,08	11,19	15,51	17,28	11,34	13,56	16,49
0.6	14,51	17,59	20,22	11,02	15,63	17,92	12,97	18,25	18,49	11,73	14,95	19,13
0.7	11,74	14,53	18,43	8,45	15,20	18,11	14,49	12,85	15,57	12,03	14,41	13,33
0.8	12,74	16,98	21,21	15,99	19,38	19,65	13,78	17,70	20,73	13,92	17,05	19,33
0.9	13,72	15,85	16,46	12,21	12,99	19,41	12,44	17,95	18,28	10,88	13,65	15,82
1	15,21	22,03	20,06	15,51	20,43	20,03	13,86	19,37	21,18	13,33	19,27	19,28
average	**20,65**	**23,97**	**25,94**	**16,31**	**20,32**	**21,93**	**15,57**	**19,11**	**20,84**	**12,45**	**15,85**	**17,18**
maximum	**50,82**	**52,05**	**52,51**	**31,55**	**33,13**	**32,49**	**24,28**	**26,26**	**27,75**	**16,97**	**24,11**	**25,22**

Table 4.30. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.4 under BR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	51,38	51,89	52,20	31,81	32,08	31,00	19,50	18,97	20,67	15,40	15,86	15,58
0.1	50,88	49,21	51,21	29,97	31,02	30,72	18,79	17,71	17,40	15,17	14,61	15,11
0.2	26,12	38,15	42,58	28,29	34,96	37,63	25,87	31,14	34,15	22,85	28,71	31,29
0.3	19,40	24,61	28,44	17,94	23,71	26,65	18,66	23,67	27,71	15,88	21,89	23,21
0.4	17,03	21,87	24,73	16,94	22,81	23,23	15,12	22,03	24,75	15,66	19,66	24,47
0.5	12,95	18,87	23,62	14,97	20,55	21,09	15,27	19,55	22,54	12,55	17,70	21,03
0.6	13,81	18,95	20,86	14,02	19,55	22,49	15,74	18,34	23,29	13,89	17,01	18,93
0.7	14,53	20,81	21,93	14,40	20,46	23,13	15,47	19,88	24,17	13,68	17,81	20,63
0.8	14,66	21,00	22,22	16,24	18,54	22,59	14,23	19,45	21,00	18,39	18,80	22,88
0.9	18,06	18,90	23,21	15,46	20,49	23,10	17,10	20,55	22,40	13,05	18,76	21,32
1	10,64	15,93	15,64	6,10	17,79	20,09	10,41	13,08	15,81	11,10	17,29	18,91
average	**22,68**	**27,29**	**29,69**	**18,74**	**23,81**	**25,61**	**16,92**	**20,40**	**23,08**	**15,24**	**18,92**	**21,22**
maximum	**51,38**	**51,89**	**52,20**	**31,81**	**34,96**	**37,63**	**25,87**	**31,14**	**34,15**	**22,85**	**28,71**	**31,29**

Table 4.31. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.6 under BR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	29,03	30,34	29,22	19,36	17,13	17,94	9,16	8,98	10,25	12,41	11,47	12,11
0.1	29,59	29,05	29,77	20,52	18,96	20,74	9,19	9,24	11,31	13,10	12,67	12,65
0.2	18,41	24,81	29,10	19,16	25,82	29,48	18,85	25,50	28,49	15,37	21,64	24,88
0.3	17,34	22,99	26,37	17,32	22,98	25,24	18,53	23,05	26,23	15,14	22,02	24,24
0.4	16,94	20,41	22,55	14,85	20,57	21,43	15,23	20,71	22,94	14,03	18,98	21,23
0.5	16,69	21,84	23,70	15,07	20,50	23,21	16,99	21,57	22,82	16,30	21,42	23,04
0.6	17,35	22,28	23,97	15,44	21,33	24,23	16,14	22,41	24,06	17,03	19,19	23,51
0.7	13,16	21,42	23,96	15,03	18,05	22,64	14,26	19,79	22,86	15,51	17,66	21,90
0.8	15,07	19,31	23,01	14,50	19,25	21,18	14,62	20,03	19,98	12,95	20,98	18,86
0.9	14,42	22,54	21,41	17,66	21,70	22,58	15,59	20,28	22,03	16,35	18,04	22,08
1	16,19	21,02	20,15	13,93	19,59	22,36	15,90	20,26	22,91	16,52	19,82	22,78
average	**18,56**	**23,27**	**24,84**	**16,62**	**20,54**	**22,82**	**14,95**	**19,26**	**21,26**	**14,97**	**18,53**	**20,66**
maximum	**29,59**	**30,34**	**29,77**	**20,52**	**25,82**	**29,48**	**18,85**	**25,50**	**28,49**	**17,03**	**22,02**	**24,88**

Table 4.32. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.6 under BR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	36,74	34,91	36,58	21,42	21,90	20,92	10,96	12,38	10,85	13,81	12,45	13,74
0.1	36,53	35,20	35,46	22,66	23,48	22,19	11,40	11,59	10,62	13,30	13,24	13,00
0.2	23,92	29,56	33,06	24,58	30,65	34,17	23,79	28,71	31,14	19,17	25,84	28,38
0.3	18,26	23,24	25,16	16,81	24,20	26,49	16,13	23,12	26,67	16,40	20,80	23,47
0.4	14,60	20,84	24,89	14,76	19,44	23,89	14,81	20,60	22,57	13,92	18,90	21,04
0.5	15,84	20,09	24,19	15,48	20,54	22,97	15,67	20,21	23,17	14,24	18,96	21,13
0.6	12,87	18,69	20,06	13,54	16,96	20,51	13,33	18,97	20,27	10,64	17,15	20,51
0.7	14,18	20,05	19,08	14,13	19,76	21,55	15,60	18,38	22,60	14,17	19,30	20,34
0.8	14,69	19,37	22,41	15,10	21,47	21,35	14,73	20,84	22,20	14,44	18,99	21,04
0.9	16,17	22,15	22,78	16,26	20,63	22,00	15,58	22,65	20,19	16,62	21,88	22,88
1	16,88	20,08	21,96	16,14	20,58	22,63	16,08	20,09	21,75	14,42	20,87	21,58
average	**20,06**	**24,02**	**25,97**	**17,35**	**21,78**	**23,52**	**15,28**	**19,78**	**21,09**	**14,65**	**18,94**	**20,65**
maximum	**36,74**	**35,20**	**36,58**	**24,58**	**30,65**	**34,17**	**23,79**	**28,71**	**31,14**	**19,17**	**25,84**	**28,38**

Table 4.33. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.6 under BR traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	40,27	40,85	40,27	23,08	23,88	24,68	12,40	10,96	12,90	16,24	15,52	15,48
0.1	39,23	39,35	39,09	24,20	25,76	25,09	12,50	13,02	10,42	16,08	16,37	16,21
0.2	31,91	39,33	43,21	30,76	36,07	37,65	25,10	26,93	28,43	25,21	28,41	30,32
0.3	23,01	25,70	29,22	21,92	28,32	29,85	20,71	27,08	30,51	18,00	25,54	26,74
0.4	17,75	21,73	25,05	17,45	21,43	24,59	17,89	21,49	25,63	16,45	21,93	23,25
0.5	18,14	23,01	25,36	18,36	21,88	26,54	17,59	22,83	26,36	15,43	20,91	24,39
0.6	14,83	21,52	25,72	18,53	20,86	23,63	16,27	20,86	25,79	15,71	19,96	24,27
0.7	18,66	23,25	25,76	17,96	21,34	23,74	17,38	21,96	26,77	15,83	20,76	24,80
0.8	19,29	24,84	25,42	16,29	24,35	23,08	17,48	21,15	23,11	14,85	20,46	21,95
0.9	14,51	21,23	23,38	13,84	21,90	21,52	17,33	21,50	22,28	14,76	20,08	25,24
1	15,45	22,29	24,82	17,82	20,61	23,65	18,33	26,12	26,82	15,68	20,65	22,36
average	**23,00**	**27,55**	**29,75**	**20,02**	**24,22**	**25,82**	**17,54**	**21,26**	**23,55**	**16,75**	**20,97**	**23,18**
maximum	**40,27**	**40,85**	**43,21**	**30,76**	**36,07**	**37,65**	**25,10**	**27,08**	**30,51**	**25,21**	**28,41**	**30,32**

Table 4.34. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.2 under PS traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	64,10	64,09	63,69	44,72	45,13	44,61	45,91	45,31	47,85	4,67	5,25	5,33
0.1	60,95	61,51	61,34	44,66	44,01	44,35	43,85	45,49	44,56	5,32	6,06	3,72
0.2	55,00	55,94	55,33	40,66	43,48	43,22	40,71	41,15	40,97	5,69	6,39	5,25
0.3	39,53	44,79	46,80	36,86	38,73	40,93	31,93	33,89	35,28	10,36	10,91	11,50
0.4	23,53	28,16	30,06	21,92	29,73	31,74	20,65	26,34	27,98	14,02	17,72	20,08
0.5	16,59	21,26	23,50	16,15	20,76	23,74	16,30	21,69	23,26	12,73	16,48	18,50
0.6	14,82	19,93	21,18	14,23	18,59	21,61	14,78	19,61	20,56	12,91	16,91	19,66
0.7	14,13	17,61	20,52	13,70	17,91	20,07	14,40	18,17	19,82	12,06	16,32	18,90
0.8	14,37	19,34	20,58	15,31	18,66	21,71	15,19	18,41	21,72	12,58	18,51	20,53
0.9	10,89	16,77	19,24	12,05	16,34	19,27	12,92	16,64	18,20	10,92	16,05	17,33
1	15,21	19,55	20,76	15,58	18,47	20,65	15,16	18,23	21,52	13,61	18,80	19,33
average	**29,92**	**33,54**	**34,82**	**25,08**	**28,35**	**30,17**	**24,71**	**27,72**	**29,25**	**10,44**	**13,58**	**14,56**
maximum	**64,10**	**64,09**	**63,69**	**44,72**	**45,13**	**44,61**	**45,91**	**45,49**	**47,85**	**14,02**	**18,80**	**20,53**

Table 4.35. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.2 under PS traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	64,65	64,81	64,97	42,80	43,19	44,19	41,56	38,42	41,58	9,77	8,85	8,97
0.1	62,73	63,47	63,51	42,07	43,65	42,96	38,21	37,06	38,14	8,65	7,80	8,61
0.2	59,21	57,49	58,55	42,05	42,57	42,82	36,26	36,30	36,98	9,37	10,90	9,53
0.3	45,57	48,04	50,28	38,18	39,85	40,29	31,02	32,31	33,51	12,73	13,25	13,71
0.4	26,67	33,12	35,92	25,72	32,17	35,44	23,01	29,35	30,14	14,47	19,43	21,00
0.5	19,57	25,35	26,60	18,52	24,47	26,39	18,15	23,35	25,30	14,09	19,10	20,47
0.6	16,62	21,10	23,32	16,97	21,64	23,68	15,65	20,01	23,67	13,21	18,17	21,24
0.7	15,63	19,20	21,68	13,99	19,44	20,64	15,35	19,76	22,22	13,63	17,05	20,41
0.8	15,49	20,89	21,66	14,32	19,72	21,61	13,27	18,44	21,97	13,82	17,74	19,27
0.9	11,88	18,35	19,56	12,96	16,28	18,96	12,28	17,38	18,32	13,31	16,38	18,58
1	16,38	19,04	20,02	14,17	19,34	20,17	14,58	19,78	21,73	13,76	17,94	20,44
average	**32,22**	**35,53**	**36,92**	**25,61**	**29,30**	**30,65**	**23,58**	**26,56**	**28,51**	**12,44**	**15,15**	**16,57**
maximum	**64,65**	**64,81**	**64,97**	**42,80**	**43,65**	**44,19**	**41,56**	**38,42**	**41,58**	**14,47**	**19,43**	**21,24**

Table 4.36. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.2 under PS traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	61,36	62,22	62,80	40,66	40,23	40,17	24,67	24,53	27,45	14,96	14,80	14,63
0.1	58,43	58,85	58,97	40,30	41,39	37,54	25,26	24,83	24,11	13,96	13,89	14,08
0.2	55,84	55,39	55,20	37,74	39,60	39,26	23,32	24,80	23,84	13,90	14,14	14,20
0.3	48,78	49,38	51,62	37,35	35,37	37,90	20,37	23,94	23,34	14,01	14,45	14,71
0.4	35,11	39,21	40,38	30,73	34,11	35,78	20,04	23,95	25,04	18,61	19,69	19,82
0.5	25,44	30,72	31,96	24,76	31,17	32,15	20,91	24,34	26,22	17,48	22,05	24,74
0.6	18,48	24,73	28,64	18,85	24,73	27,65	17,64	22,10	24,36	16,10	21,26	23,81
0.7	18,35	24,37	25,17	18,76	24,60	25,76	17,73	23,65	24,80	17,50	21,30	24,10
0.8	16,35	22,44	22,98	15,87	21,43	23,78	15,21	21,20	23,01	13,01	20,57	21,45
0.9	16,87	20,60	22,67	16,50	20,26	22,87	15,24	20,53	23,29	15,26	18,51	22,52
1	15,35	18,69	21,97	14,86	19,72	22,34	13,16	19,49	21,24	14,02	18,06	20,53
average	**33,67**	**36,96**	**38,40**	**26,94**	**30,24**	**31,38**	**19,42**	**23,03**	**24,25**	**15,35**	**18,06**	**19,51**
maximum	**61,36**	**62,22**	**62,80**	**40,66**	**41,39**	**40,17**	**25,26**	**24,83**	**27,45**	**18,61**	**22,05**	**24,74**

Table 4.37. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.4 under PS traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	50,44	51,15	51,04	36,10	38,53	37,95	27,40	24,91	23,74	11,95	11,56	11,62
0.1	48,58	49,61	49,48	36,83	36,18	36,19	26,68	24,99	24,73	11,63	11,07	11,63
0.2	43,70	43,69	44,47	33,72	32,70	33,78	23,26	23,29	21,41	10,10	11,08	9,53
0.3	36,17	40,06	39,47	32,12	33,45	34,05	21,63	22,68	22,76	13,24	14,36	14,40
0.4	22,38	29,06	31,42	22,12	28,35	29,72	19,86	22,67	23,74	14,67	18,85	20,20
0.5	18,86	24,74	28,88	19,82	25,12	27,65	18,68	22,71	23,07	16,62	19,93	22,62
0.6	16,57	20,91	24,26	16,77	21,81	22,97	14,90	20,65	23,31	14,75	19,14	21,38
0.7	15,71	20,21	21,25	14,87	20,61	23,34	14,71	19,49	22,29	14,38	18,29	20,30
0.8	16,54	22,70	23,49	18,34	21,84	23,27	16,71	21,83	24,15	16,95	21,14	23,05
0.9	15,88	20,92	23,54	16,25	19,17	22,22	15,32	19,49	22,02	15,25	19,39	21,90
1	16,01	20,56	23,00	16,73	21,13	22,49	15,16	19,16	20,62	15,27	20,30	21,96
average	**27,35**	**31,24**	**32,75**	**23,97**	**27,17**	**28,51**	**19,48**	**21,99**	**22,89**	**14,07**	**16,83**	**18,05**
maximum	**50,44**	**51,15**	**51,04**	**36,83**	**38,53**	**37,95**	**27,40**	**24,99**	**24,73**	**16,95**	**21,14**	**23,05**

Table 4.38. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.4 under PS traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	54,72	53,94	53,81	36,99	36,73	37,81	28,26	26,86	26,21	11,85	11,63	11,43
0.1	51,16	52,95	53,09	36,69	38,52	36,67	27,62	27,42	29,42	13,14	11,60	13,17
0.2	46,44	47,96	48,08	35,91	34,64	34,61	22,91	25,80	23,76	11,67	11,58	11,11
0.3	40,57	42,34	43,24	32,52	32,89	34,52	20,60	22,34	21,45	12,54	12,38	13,50
0.4	24,87	31,93	34,07	27,31	31,97	32,42	20,34	23,26	25,73	17,32	20,03	20,72
0.5	19,33	23,16	27,52	18,91	24,89	28,41	18,51	22,63	24,28	15,99	19,40	21,80
0.6	18,27	22,76	26,01	18,29	23,41	23,56	17,29	21,99	23,11	14,65	20,46	22,25
0.7	15,75	21,34	22,53	16,52	20,88	22,25	15,56	20,07	22,47	14,17	18,64	20,22
0.8	15,53	19,49	20,79	12,94	18,93	20,47	13,41	17,28	21,61	12,95	18,04	20,32
0.9	13,12	20,93	22,54	15,18	19,78	21,36	14,82	21,15	21,63	15,18	18,28	19,60
1	15,43	20,56	22,42	15,93	21,91	21,60	13,42	19,86	21,55	12,57	18,91	21,48
average	**28,65**	**32,49**	**34,01**	**24,29**	**27,69**	**28,52**	**19,34**	**22,61**	**23,75**	**13,82**	**16,45**	**17,78**
maximum	**54,72**	**53,94**	**53,81**	**36,99**	**38,52**	**37,81**	**28,26**	**27,42**	**29,42**	**17,32**	**20,46**	**22,25**

Table 4.39. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.4 under PS traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	49,19	50,01	49,87	34,01	33,98	34,47	18,87	17,17	17,46	14,37	14,66	13,56
0.1	49,33	50,24	48,95	34,31	36,16	35,23	19,75	18,48	18,89	15,59	15,93	15,75
0.2	45,61	46,81	47,44	33,49	34,62	32,97	18,30	14,82	17,16	15,68	15,53	16,04
0.3	40,61	43,25	42,79	31,74	33,91	32,10	15,98	17,38	16,04	15,64	16,26	16,39
0.4	30,13	34,96	35,74	27,48	31,56	30,81	18,97	19,07	22,08	17,46	19,13	20,65
0.5	24,01	28,04	31,03	23,61	27,91	29,02	20,81	22,49	23,11	17,88	22,47	23,40
0.6	19,77	25,53	28,71	21,69	25,29	27,70	18,63	24,42	26,54	18,26	22,59	23,68
0.7	17,89	24,69	25,98	17,21	22,15	26,40	17,48	22,45	24,77	16,74	21,24	22,95
0.8	16,78	24,36	24,61	18,72	24,20	25,94	18,33	22,20	24,41	17,89	23,00	24,78
0.9	15,25	22,41	25,24	16,98	21,62	23,98	16,18	21,23	23,72	16,01	20,30	23,28
1	15,45	21,21	22,95	16,70	20,85	23,57	15,45	20,52	22,93	14,57	20,01	22,75
average	**29,46**	**33,77**	**34,85**	**25,09**	**28,39**	**29,29**	**18,07**	**20,02**	**21,56**	**16,37**	**19,19**	**20,30**
maximum	**49,33**	**50,24**	**49,87**	**34,31**	**36,16**	**35,23**	**20,81**	**24,42**	**26,54**	**18,26**	**23,00**	**24,78**

Table 4.40. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.2 and P(W) = 0.6 under PS traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	29,26	30,90	28,68	22,04	21,80	23,16	11,46	9,94	9,49	13,58	12,81	13,59
0.1	26,55	27,94	29,94	21,92	23,18	21,85	10,24	7,45	7,86	12,74	11,96	12,25
0.2	26,11	29,47	28,03	22,33	21,29	20,89	8,07	9,30	10,65	12,66	13,33	13,31
0.3	25,34	26,94	25,35	20,12	20,15	19,98	9,64	9,72	10,79	13,04	14,17	13,34
0.4	22,90	25,51	27,80	21,97	25,11	25,96	14,19	17,72	16,51	17,30	19,34	19,93
0.5	19,32	24,77	27,46	19,63	24,48	26,62	16,53	19,49	22,35	16,73	21,51	24,27
0.6	19,08	25,20	26,79	20,77	25,70	27,35	18,71	22,37	24,91	18,06	23,31	24,93
0.7	17,76	22,57	27,29	19,56	23,03	27,14	17,01	22,22	25,86	16,78	23,25	24,46
0.8	16,97	23,55	25,20	17,97	22,98	25,60	17,96	22,55	24,73	16,76	22,41	24,87
0.9	18,29	25,03	25,39	18,89	25,18	26,03	19,00	24,03	26,68	18,24	23,67	26,51
1	17,81	24,40	26,12	17,89	23,57	26,11	17,31	22,30	24,56	17,22	23,16	25,89
average	**21,76**	**26,03**	**27,10**	**20,28**	**23,32**	**24,61**	**14,56**	**17,01**	**18,58**	**15,74**	**18,99**	**20,30**
maximum	**29,26**	**30,90**	**29,94**	**22,33**	**25,70**	**27,35**	**19,00**	**24,03**	**26,68**	**18,24**	**23,67**	**26,51**

Table 4.41. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.4 and P(W) = 0.6 under PS traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	34,95	34,81	35,71	23,41	24,41	26,49	10,41	11,04	10,66	12,78	12,60	12,60
0.1	33,34	35,99	34,96	25,25	26,49	24,62	11,51	11,19	11,50	14,02	12,59	13,29
0.2	32,56	33,29	31,94	23,86	23,36	22,82	10,83	11,71	9,91	13,56	13,98	13,53
0.3	30,64	31,48	32,03	23,00	24,70	24,85	12,17	13,56	13,86	15,01	15,54	16,23
0.4	25,10	29,40	29,80	22,03	24,68	25,66	13,92	14,13	16,95	16,32	18,71	19,62
0.5	23,85	28,14	31,20	23,98	27,53	28,94	18,58	23,12	24,14	18,11	24,04	25,76
0.6	18,00	24,05	26,18	19,78	24,33	26,14	17,08	22,21	24,52	15,40	22,44	24,74
0.7	19,37	24,15	26,32	18,18	23,71	26,49	17,77	22,76	25,70	16,88	22,44	24,25
0.8	18,61	24,62	27,02	19,03	24,93	26,13	18,10	23,88	25,36	19,12	23,27	25,01
0.9	17,33	22,68	26,01	18,39	23,46	26,75	17,98	23,60	25,78	17,76	23,72	24,61
1	17,44	21,94	24,95	16,56	22,50	24,45	17,54	22,76	24,57	16,78	21,89	23,49
average	**24,65**	**28,23**	**29,65**	**21,23**	**24,55**	**25,76**	**15,08**	**18,18**	**19,36**	**15,98**	**19,20**	**20,28**
maximum	**34,95**	**35,99**	**35,71**	**25,25**	**27,53**	**28,94**	**18,58**	**23,88**	**25,78**	**19,12**	**24,04**	**25,76**

Table 4.42. The decreasing rate of average network response time using different combinations of message combining and message priority algorithms for P(M) = 0.6 and P(W) = 0.6 under PS traffic

Algorithm / T/R	c_path & 2_comb	c_path & 3_comb	c_path & 4_comb	dircache & 2_comb	dircache & 3_comb	dircache & 4_comb	dir & 2_comb	dir & 3_comb	dir & 4_comb	cache & 2_comb	cache & 3_comb	cache & 4_comb
0.05	37,68	39,11	39,44	27,07	27,69	25,00	9,99	11,05	9,45	14,38	14,84	14,34
0.1	36,70	38,82	37,55	25,30	27,07	25,91	9,68	9,03	10,74	14,94	14,90	15,15
0.2	36,84	35,30	36,19	24,42	25,55	26,74	8,79	11,86	10,67	15,17	16,16	15,19
0.3	32,34	33,19	34,10	23,40	23,68	22,82	9,59	11,68	8,55	15,25	15,07	14,87
0.4	27,02	31,26	31,39	23,58	23,70	26,17	13,75	13,28	15,50	16,28	17,35	19,55
0.5	22,31	25,92	30,42	22,38	26,05	27,73	16,24	19,73	21,11	17,64	21,27	23,18
0.6	21,30	26,85	27,64	20,23	24,73	27,58	18,43	22,97	24,81	18,19	22,15	24,54
0.7	18,32	24,22	25,57	17,98	22,72	25,50	16,32	20,76	24,76	17,68	21,64	24,32
0.8	18,13	24,02	26,89	17,51	24,19	25,41	16,69	22,98	25,30	16,25	22,39	25,23
0.9	17,37	22,83	25,56	17,25	22,54	27,31	15,59	23,82	24,96	16,47	22,59	24,29
1	17,60	23,10	26,07	17,31	22,29	25,55	17,87	23,32	23,67	15,95	22,43	24,94
average	**25,96**	**29,51**	**30,99**	**21,49**	**24,56**	**25,98**	**13,90**	**17,32**	**18,14**	**16,20**	**19,16**	**20,51**
maximum	**37,68**	**39,11**	**39,44**	**27,07**	**27,69**	**27,73**	**18,43**	**23,82**	**25,30**	**18,19**	**22,59**	**25,23**