**ÇUKUROVA UNIVERSITY**
**INSTITUTE OF NATURAL AND APPLIED SCIENCES**

**MSc THESIS**

**İpek ABASIKELEŞ**

# PERFORMANCE EVALUATION OF DIRECTORY-BASED CACHE COHERENCE PROTOCOLS ON AN OPTICAL INTERCONNECTION NETWORK SUPPORTING DISTRIBUTED SHARED MEMORY

**DEPARTMENT OF COMPUTER ENGINEERING**

**ADANA, 2009**

**ÇUKUROVA ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

# PERFORMANCE EVALUATION OF DIRECTORY-BASED CACHE COHERENCE PROTOCOLS ON AN OPTICAL INTERCONNECTION NETWORK SUPPORTING DISTRIBUTED SHARED MEMORY

**İpek ABASIKELEŞ**

**YÜKSEK LİSANS TEZİ**
**BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİMDALI**

Bu tez ..../...../…... tarihinde aşağıdaki jüri üyeleri tarafından oybirliği/oyçokluğu ile kabul edilmiştir.

| İmza…….. | İmza…….. | İmza…….. |
|---|---|---|
| Yrd.Doç.Dr.M.Fatih AKAY<br>DANIŞMAN | Yrd.Doç.Dr.Ramazan ÇOBAN<br>ÜYE | Yrd.Doç.Dr.Murat AKSOY<br>ÜYE |

Bu tez Enstitümüz Bilgisayar Mühendisliği Anabilim Dalında hazırlanmıştır.
Kod No:

Prof. Dr. Aziz ERTUNÇ
Enstitü Müdürü
İmza ve Mühür

**Bu Çalışma Çukurova Üniversitesi Bilimsel Araştırma Projeleri Birimi Tarafından Desteklenmiştir.**

**Proje No: MMF2009YL4**

**Not**: Bu tezde kullanılan özgün ve başka kaynaktan yapılan bildirişlerin, çizelge, şekil ve fotoğrafların kaynak gösterilmeden kullanımı, 5846 sayılı Fikir ve Sanat Eserleri Kanunundaki hükümlere tabidir.

# ÖZ

## YÜKSEK LİSANS TEZİ

## DAĞITIK ORTAK HAFIZA DESTEKLİ OPTİK İNTERKONEKSİYON AĞI ÜZERİNDE DİZİN TABANLI ÖN BELLEK TUTARLILIK PROTOKOLLERİNİN PERFORMANS ANALİZİ

**İpek ABASIKELEŞ**

**ÇUKUROVA ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**
**BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**

Danışman: Yrd.Doç.Dr. M.Fatih AKAY
Yıl: 2009, Sayfa: 67
Jüri: Yrd.Doç.Dr. M.Fatih AKAY
Yrd.Doç.Dr. Ramazan ÇOBAN
Yrd.Doç.Dr. Murat AKSOY

Bu çalışmada, simülasyon tekniği kullanılarak katı istek-yanıt, aracılı iletim ve cevap iletimi olmak üzere üç farklı dizin tabanlı ön bellek tutarlılık protokolünün dağıtık ortak hafıza (DOH) destekleyen, düşük gecikmeli, yüksek bant genişliğine sahip yayın tabanlı fiber optik bir ara bağlantı ağı olan SOME-Bus üzerinde performans analizi yapılmıştır. Literatürde bu protokoller sadece okuma durumu için tasarlandığından, performans ölçümleri yapılmadan önce protokoller yazma durumu için yeniden düzenlenmiştir. Simüle edilen sistem, her biri bir işlemci, ön bellek denetleyicisi, dizin denetleyicisi ve çıkış kanalı içeren 64 düğümden oluşmaktadır. Simülasyonlar, her bir protokol için ortalama işlemci kullanımını ve ortalama ağ gecikmesini; ortalama kanal servis süresinin ortalama iplik çalışma süresine oranı (T/R), ön bellek bloğunun değiştirilmiş olma olasılığı {P(M)}, yazma ön bellek eksiklerinin (miss) oranı {P(W)} gibi önemli DOH parametrelerinin değişken değerleri için, farklı ağ trafik modelleri altında gerçeklenmiştir. Elde edilen sonuçlar, tüm protokollerinin performansının, tüm ağ trafik modelleri altında, P(W), P(M) veya T/R oranının artması ile düştüğünü göstermiştir. P(M)'deki artış, P(W)'nin protokol performansları üzerindeki etkisini kırmaktadır. Cevap iletimi protokolü yüksek P(M) değerlerinde, aracılı iletim protokolü ise yüksek P(W) ile düşük P(M)'nin bir arada bulunduğu değerlerde en iyi performanslarını sergilerler. Katı istek-yanıt protokolü ise yoğun bölge trafiği altında en yüksek performansı gösteren protokoldür.

**Anahtar Kelimeler**: dizin protokolü, dağıtık ortak hafıza, çoklu mikroişlemciler, optik ara bağlantı.

## ABSTRACT

## MSc THESIS

**PERFORMANCE EVALUATION OF DIRECTORY-BASED CACHE COHERENCE PROTOCOLS ON AN OPTICAL INTERCONNECTION NETWORK SUPPORTING DISTRIBUTED SHARED MEMORY**

**İpek ABASIKELEŞ**

**COMPUTER ENGINEERING**
**INSTITUTE OF NATURAL AND APPLIED SCIENCES**
**UNIVERSITY OF ÇUKUROVA**

Supervisor: Assist.Prof.Dr. M.Fatih AKAY
Year: 2009, Pages: 67
Jury: Assist.Prof.Dr. M.Fatih AKAY
Assist.Prof.Dr. Ramazan ÇOBAN
Assist.Prof.Dr. Murat AKSOY

In this work, the performance of three directory-based cache-coherence protocols; strict request-response, intervention forwarding and reply forwarding are evaluated via simulation on the Simultaneous Optical Multiprocessor Exchange Bus (SOME-Bus), which is a low-latency and high-bandwidth broadcast-based fiber-optic interconnection network supporting distributed shared memory (DSM). As these protocols appear only for read requests in literature, the protocols are modified for write requests before evaluating their performances. The simulated system contains 64 nodes, each of which has a processor, a cache controller, a directory controller and an output channel. Simulations have been conducted for each protocol to measure average processor utilization and average network latency for varying values of the important DSM parameters such as the ratio of the mean channel service time to mean thread run time (T/R), probability of a cache block being in modified state {P(M)}, the fraction of write misses {P(W)} and under different traffic patterns. The results reveal that the performance of all protocols decreases under all traffic patterns as P(W), P(M) or T/R increases. The effect of P(W) on the performance of the protocols reduces as P(M) increases. Reply forwarding protocol performs the best for high P(M) values, intervention forwarding protocol yields the best performance for low P(M) and high P(W) values and strict request-response is the best protocol under hot-region (HR) traffic.



**Keywords**: directory protocols, distributed shared memory, multiprocessors, optical interconnection network.

**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 members of MSc thesis jury, Assist.Prof.Dr. Ramazan ÇOBAN and Assist.Prof.Dr. Murat AKSOY, for their suggestions and corrections.

I would like to thank OPNET Technologies, Inc. for letting me use the OPNET Modeler under the University Program.

I would also like to thank Cukurova University Scientific Research Projects Center for supporting this work (Project no: MMF2009YL4).

My sincere thanks also goes to 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**

ÖZ ..... I
ABSTRACT ..... II
ACKNOWLEDGMENT ..... III
CONTENTS ..... IV
LIST OF TABLES ..... V
LIST OF FIGURES ..... VI
1. INTRODUCTION ..... 1
1.1. Distributed Shared Memory Multiprocessors (DSM) ..... 1
1.2. Cache Coherence ..... 3
1.2.1. Directory-based Cache Coherence Protocols ..... 5
1.3. Interconnection Networks ..... 8
1.3.1. The SOME-Bus Architecture ..... 9
1.4. Contributions of the Thesis ..... 12
2. RELATED WORK ..... 13
3. EVALUATION METHODOLOGY ..... 18
3.1. Synthetic Traffic Workloads ..... 18
3.1.1. Temporal ..... 18
3.1.2. Spatial ..... 19
3.1.3. Packet Size ..... 19
3.2. Simulation Framework ..... 20
3.2.1. Simulation Tool and System Architecture ..... 20
3.2.2. System parameters ..... 25
4. RESULTS AND DISCUSSION ..... 29
4.1. Theoretical Analysis ..... 29
4.2. Simulation Results ..... 32
5. CONCLUSION ..... 38
REFERENCES ..... 40
BIOGRAPHY ..... 49
APPENDICES ..... 50

**LIST OF TABLES** **PAGE**

Table 3.1. System Parameters ..... 25
Table 3.2. The types of packets in the system ..... 27
Table 4.1. The increase rate of average network latency for each directory protocol under each traffic pattern ..... 66
Table 4.2. The decrease rate of average processor utilization for each directory protocol under each traffic pattern ..... 67

**LIST OF FIGURES** **PAGE**

Figure 1.1. Tradeoff Between Latency & Bandwidth ..... 4
Figure 1.2. Directory protocol diagrams of a read operation to a Modified cache state for (a) strict request-response (b) intervention forwarding (c) reply forwarding. ..... 6
Figure 1.3. Directory protocol diagrams of a write operation to a Shared cache state for (a) strict request-response (b) intervention forwarding (c) reply forwarding. ..... 7
Figure 1.4. Receiver Array ..... 10
Figure 3.1. Node model of a 4-node DSM-based SOME-Bus system in OPNET Modeler ..... 22
Figure 3.2. State diagram of acb_fifo model in OPNET Modeler ..... 23
Figure 4.1. Message path of strict request response protocol for (request/block) (a) read / Shared & Invalid and write / Invalid, (b) read & write / Modified and (c) write / Shared ..... 30
Figure 4.2. Message path of intervention forwarding protocol for (request/block) (a) read / Shared & Invalid and write / Invalid, (b) read & write / Modified and (c) write / Shared ..... 31
Figure 4.3. Message path of reply forwarding protocol for (request/block) (a) read / Shared & Invalid and write / Invalid, (b) read & write / Modified and (c) write / Shared ..... 31
Figure 4.4. Average processor utilization and average network latency of the protocols under bit reverse traffic for P(M) = 0.2 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 50
Figure 4.5. Average processor utilization and average network latency of the protocols under perfect shuffle traffic for P(M) = 0.2 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 51
Figure 4.6. Average processor utilization and average network latency of the protocols under hot region traffic for P(M) = 0.2 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 52

Figure 4.7. Average processor utilization and average network latency of the protocols under uniform traffic for P(M) = 0.2 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 53
Figure 4.8. Average processor utilization and average network latency of the protocols under bit reverse traffic for P(M) = 0.4 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 54
Figure 4.9. Average processor utilization and average network latency of the protocols under perfect shuffle traffic for P(M) = 0.4 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 55
Figure 4.10. Average processor utilization and average network latency of the protocols under hot region traffic for P(M) = 0.4 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 56
Figure 4.11. Average processor utilization and average network latency of the protocols under uniform traffic for P(M) = 0.4 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 57
Figure 4.12. Average processor utilization and average network latency of the protocols under bit reverse traffic for P(M) = 0.6 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 58
Figure 4.13. Average processor utilization and average network latency of the protocols under perfect shuffle traffic for P(M) = 0.6 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 59
Figure 4.14. Average processor utilization and average network latency of the protocols under hot region traffic for P(M) = 0.6 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 60
Figure 4.15. Average processor utilization and average network latency of the protocols under uniform traffic for P(M) = 0.6 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 61
Figure 4.16. Average processor utilization and average network latency of the protocols under bit reverse traffic for P(M) = 0.8 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 62

Figure 4.17. Average processor utilization and average network latency of the protocols under perfect shuffle traffic for P(M) = 0.8 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 63
Figure 4.18. Average processor utilization and average network latency of the protocols under hot region traffic for P(M) = 0.8 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 64
Figure 4.19. Average processor utilization and average network latency of the protocols under uniform traffic for P(M) = 0.8 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6. ..... 65

# 1. INTRODUCTION

## 1.1. Distributed Shared Memory Multiprocessors (DSM)

As we slowly approach the physical limits of processor and memory speed, it is becoming more attractive to use multiprocessors to increase computing power. Two kinds of parallel processors have become popular: tightly coupled shared-memory multiprocessors and distributed-memory multiprocessors (Nitzberg and Lo,1991). Shared memory multiprocessors provide an attractive programming model to develop parallel applications. However, as the number of processors increases, the memory access operations saturate the interconnection medium and degrade the performance, hence severely limiting the scalability of the entire system. Distributed memory multiprocessors, on the other hand, scale well but are usually awkward to program because all the data movement between memories must be done explicitly in the application (Eskicioglu et al., 1999). Briefly, shared memory multiprocessors offer programming advantage, while distributed memory multiprocessors provide cost advantage (Jiang and Chaudhary, 2002). DSM systems are a successful hybrid of these two parallel computer classes. They provide the shared memory abstraction in systems with physically distributed memories and consequently combine the advantages of both approaches (Scott, 2005; Yahya and Bader, 2007). DSM multiprocessors are converging to a family of architectures that resemble a generic system architecture. This architecture consists of a set of nodes or clusters, with local memories, communicating over an interconnection network. Every node contains a processor, its cache subsystem, and a portion of the total main memory on the machine. It also contains 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

instructions (i.e. loads and stores), which makes them easier to program (Kodi and Louri, 2004). 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; Roy and Chaudhary, 1999; Speight and Bennett, 1997). Also, these systems are claimed to be scalable because they can achieve an almost linear performance increase with growing system size and investment. Because of these advantages, DSM is one of the most attractive solutions for building large-scale, high-performance multiprocessor systems (Grujic et al., 1996).

Latency is an ever-increasing component of data access cost, which in turn is usually the bottleneck for modern high performance systems (Dai and Panda, 1997; Hung and Liu, 2007; Yang et al., 2003). One of the fundamental communication problems in DSM systems that significantly affects scalability, is the increase in remote memory access latency as the number of processors increase in the system (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).

Caches play a vital role in reducing latency by maintaining coherent copies of shared data locally (Ha and Pinkston, 1997). Large-scale shared memory multiprocessors use caches to reduce average memory latency and increase effective network bandwidth. Unfortunately, the existence of multiple caches in a shared memory environment can lead to data inconsistencies in the system. This is referred to as the cache-coherence problem. Any system that allows multiple copies of a data item to exist at the same time must solve this problem (Parthasarathy, 1992).

## 1.2. Cache Coherence

An important factor that determines the performance of cached DSM multiprocessors is the scheme by which caches are kept coherent. A cache coherence protocol manages the read and write permissions of data in the caches to ensure all processors observe a consistent view of shared memory (Martin et al., 2003b). The protocol also defines particular states that the system is allowed to be in. The particular actions occur on a memory reference depend on the existing system state. Since the protocol directly affects the memory latencies experienced by a processor, it is a critical issue in the design of a high performance large scale shared memory machine (Parthasarathy, 1992). The two most common types of cache coherence protocols are snooping and directory-based protocols (Ha and Pinkston, 1997).

A snoopy cache coherence protocol relies on all caches monitoring the common bus that connects processors to memory. 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 protocols maintain a directory entry per memory block that records which processor(s) currently cache the block. 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; Barroso et al., 1998; Kunkel et al., 1999; Ranganathan et al., 1998). 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).

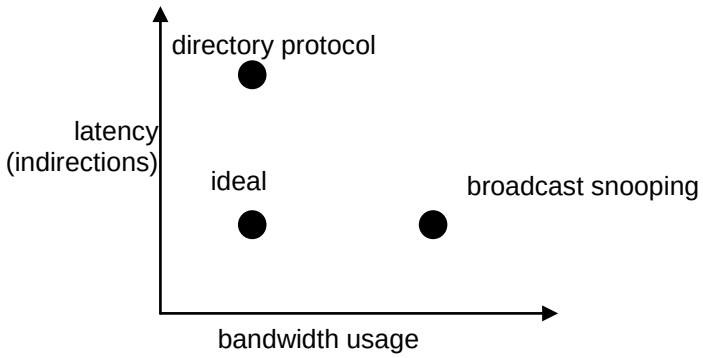
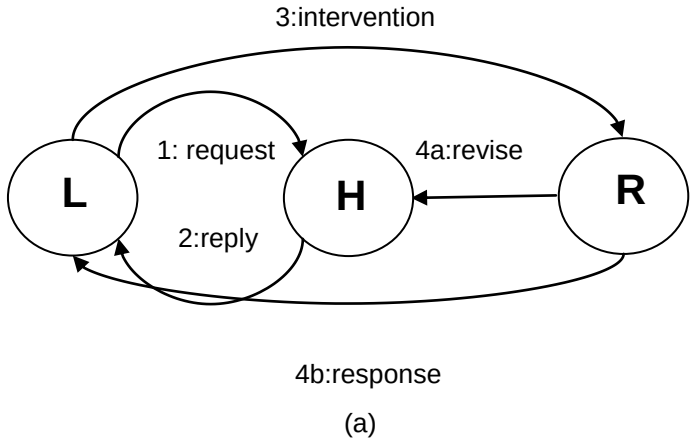


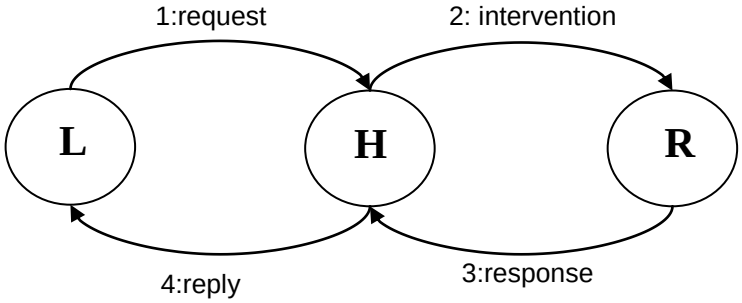

Figure 1.1. Tradeoff Between Latency & Bandwidth

### 1.2.1. Directory-based Cache Coherence Protocols

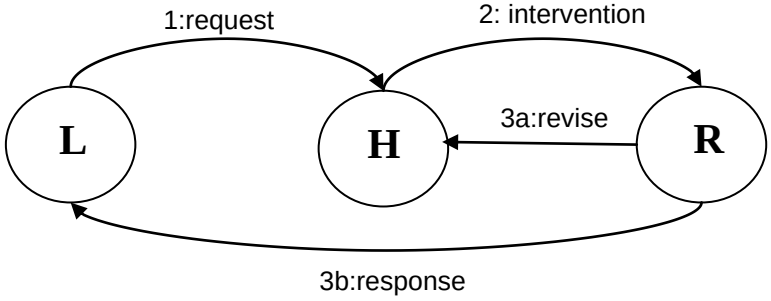
Directory protocol diagrams of a read operation to a Modified cache state for strict request-response, intervention forwarding and reply forwarding protocols are given in Figure 1.2 (Culler et al., 1999). The protocol diagrams of a write operation to a Shared block for the three protocols is shown in Figure 1.3. “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. R1 and R2 are remote sharers of the block. In strict request-response protocol, the local node initially gets the address of the owner node from the home node and then communicates with the owner node. In intervention forwarding protocol, the home node transmits the request to the owner node and waits for a reply. After getting the reply, the home node forwards it to the local node In reply forwarding protocol, 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).




(a)

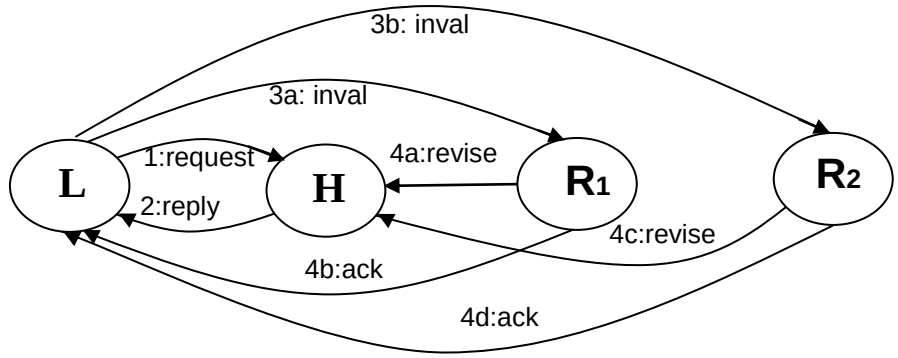



(b)

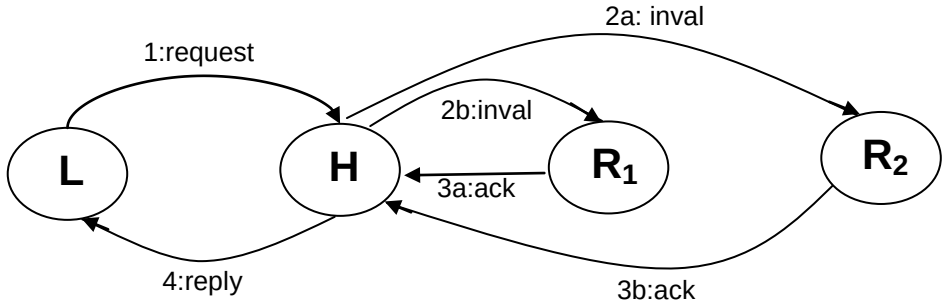



(c)

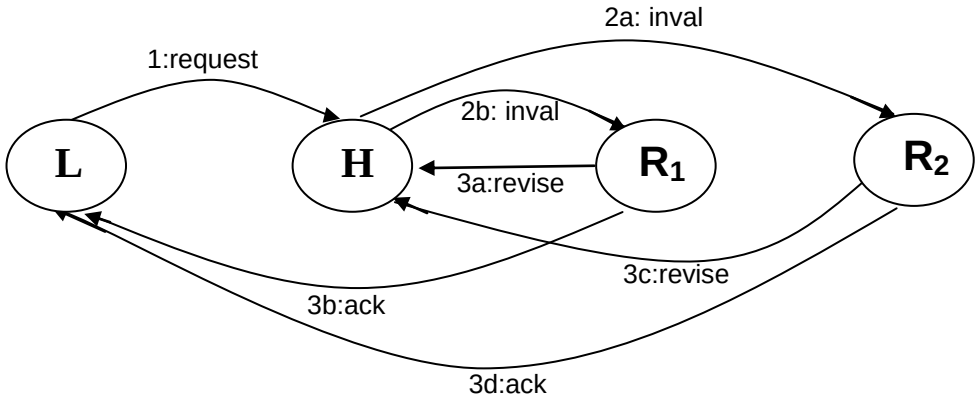
Figure 1.2. Directory protocol diagrams of a read operation to a Modified cache state for (a) strict request-response (b) intervention forwarding (c) reply forwarding




(a)




(b)




(c)

Figure 1.3. Directory protocol diagrams of a write operation to a Shared cache state for (a) strict request-response (b) intervention forwarding (c) reply forwarding

## 1.3. Interconnection Networks

Future high performance DSM systems will utilize commercial off-the-shelf processors that require aggregate computational and communication bandwidths on the order of 4 to 40 terabits per second. Thus, lack of sufficient bandwidth in the interconnection network will be the fundamental obstacle to future scalable DSM systems (Duato et al., 1997). The electrical interconnection networks connecting different processors and memory modules in a modern large scale DSM multiprocessor are running into several physical limitations. In shared memory machines, where the network is the part of the memory hierarchy, the ability to overlap memory access times with useful computation is severely limited by inter-instruction dependencies. Hence, a network with high latencies causes a significant performance bottleneck (Heirman et al., 2005). The need for an interconnect solution providing low latencies and high bandwidths while maintaining minimal power dissipation can be met by leveraging the unique advantages provided by optical interconnection networks (Heirman et al., 2008; Gunawan and Fard, 2007; Sharma et al., 2007; Wang et al., 2008). Optics could solve many other physical problems of electrical interconnects, including precise clock distribution, system synchronization (allowing larger synchronous zones, both on-chip and between chips), bandwidth and density of long interconnections, and reduction of power dissipation (Miller, 2000). Optical fibers exhibit extremely high bandwidth and can be multiplexed to provide a large number of independent communication channels. These characteristics can be exploited to improve the performance of a multiprocessor by simply replacing its traditional, scalable network with an optical equivalent. Massively parallel inter-chip optical interconnects are already making the transition from lab-settings to commercial products (Chamberlain et al., 2002; Heirman et al., 2008; Panigrahi et al., 2006). However, optical technology can usually be exploited more effectively than this. In shared memory multiprocessors, the broadcasting capability of optical fibers can be exploited to simplify the cache coherence hardware and protocol (Bianchini and Carrera, 2001).

### 1.3.1. The SOME-Bus Architecture

According to Katsinis (2003), 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. The Simultaneous Optical Multiprocessor Exchange Bus (SOME-Bus) (Lindquist et al., 1997) 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 1.4 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.

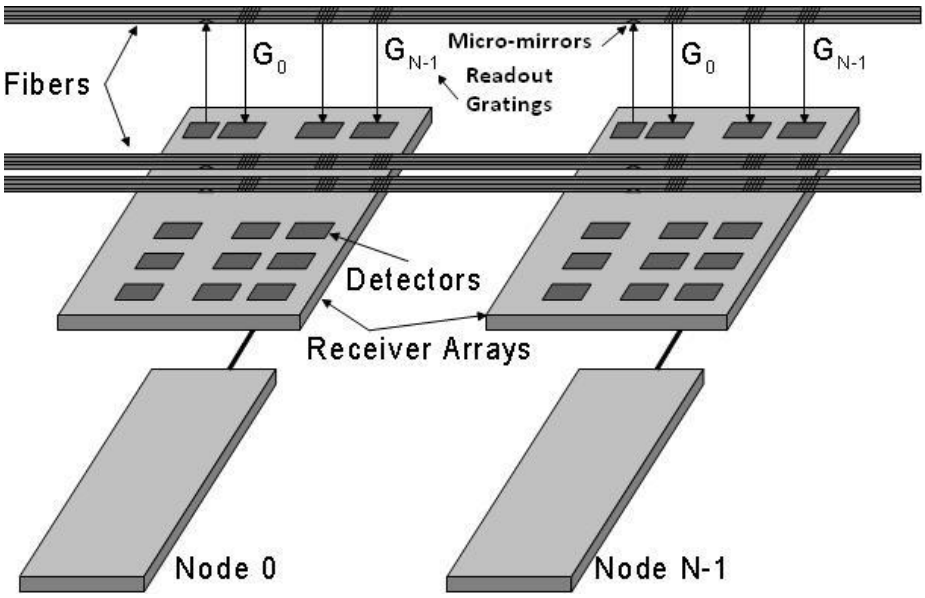



Figure 1.4. Receiver Array

The input channel interface performs address filtering, 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 may appear to be equivalent to a crossbar but it has much more functionality. A major consequence of this architecture is that, due to the multiple broadcast capability, no node is ever blocked from transmitting by another transmitter, no arbitration is required and network bandwidth scales directly with the number of nodes. No communication is ever blocked through contention for shared switching logic. With N nodes, the diameter of the SOME-bus is 1, the time needed for all-to-all communication with distinct messages is O(N) and the time needed for synchronization is O(1). Unlike a fully-connected point-to-point network, where the number of transmitters and channels increases $O(N^2)$, the number of transmitters and channels of the SOME-Bus is O(N), quite smaller than the number required in other

popular architectures, such as the hypercube or the torus. The number of receivers is $N^2$, which is larger than the number required in other architectures. They are arranged so that N receivers are fabricated as amorphous silicon structures constructed as a thin film directly on the surface of a digital CMOS device, with no lithography required. Because of the low conductivity of the amorphous silicon layer, no subsequent patterning is required, and therefore the yield and cost of the receiver is determined by the yield and cost of the CMOS device itself. Since the receiver does not need to perform any routing, its hardware complexity (including detector, logic, and packet memory storage) is small keeping their cost small, too. The full receiver array can be implemented on a single chip even for large values of N ($N > 128$). Therefore, the total receiver cost is approximately $O(N)$ instead of $O(N^2)$.

The SOME-Bus can most readily support a CC-NUMA (Cache-coherent Non-Uniform Memory Access) 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.

A SOME-Bus based system can take advantage of directory-based techniques which notify only those remote caches with affected data blocks. This can be simply accomplished by including a list of destinations in the invalidation message header. Messages are still broadcast over the sending node's output channel, but the decision to accept or reject an input message is performed at the receiver input rather than the cache controller of each remote node.

The receiver array which is integrated on a single chip can appear in the processor memory bus, so that the processor may access it as part of its memory. In addition, the receiver array in connected to the cache, and directory controllers for efficient protocol implementation.

## 1.4. Contributions of the Thesis

In this thesis, the performance of the three directory protocols called strict request-response, intervention forwarding and reply forwarding are investigated. The directory protocol diagrams of the three protocols are only defined for read operations, as given in (Culler,1999). In this thesis, the directory diagrams of a write operation for the three protocols are developed.

Although the performance of some directory-based cache coherence protocols are investigated, the performance evaluation of the directory protocols used in this thesis does not exist in literature. Also, the studies in literature mostly use electrical-based interconnection networks. However, electrical interconnection networks connecting the different processors and memory modules, are running into several physical limitations. The need for an interconnect solution providing low latencies and high bandwidths while maintaining minimal power dissipation can be met by leveraging the unique advantages provided by optical interconnection networks (Hawkins and Wills, 2006; Gunawan and Fard, 2007; Miller, 2000; Wang et al., 2008).

In this thesis, the performances of the three protocols are investigated on a broadcast-based optical interconnection network. For handling long latencies arising from directory protocols and benefiting from broadcast capability of snooping protocols, a broadcast based optical interconnection network called SOME-Bus is used. The SOME-Bus is a low-latency, high-bandwidth, fiber-optic network that directly connects each processing node to all other nodes without contention (Hecht, 2003). Its properties distinguish it from other optical networks examined in the past (Gravenstreter and Melhem, 1998; Narravula, 2003; Ould-Khaoua, 1996; Rajasekaran and Sahni, 1997; Szymanski, 1995; Zhu et al., 2004).

# 2. RELATED WORK

As access to main memory is slow compared to the processor speed, hardware caches are necessary for acceptable performance (Lawrance, 1998). However, cache coherence problems can arise in DSM systems when more than one processor cache holds a copy of a data item (Chandramouli and Iyer, 2004). Most large shared-memory multiprocessors use directory protocols to keep per-processor caches coherent. (Lawrance, 1998; Mukherjee and Hill, 1998). The two major performance goals of the optimizations implemented on protocol level are reducing the number of network transactions generated per memory operation and reducing the number of actions, especially network transactions, that are on the critical path of the processor (Culler, 1999).

In literature, several studies on directory protocols focus on the organization of the directory. Ros et al. (2005) have presented a lightweight directory architecture aimed at facing two important problems: (i) the directory memory overhead; and (ii) the long L2 miss latencies due to the indirection introduced by the accesses to directory information, usually stored in main memory. The lightweight directory architecture removes the directory structure from main memory and it stores directory information in the L2 cache avoiding in most cases the access to main memory. The proposed architecture has been evaluated based on extensive execution-driven simulations of a 32-node cc-NUMA multiprocessor and the results have demonstrated that the lightweight directory architecture achieves better performance than a non-scalable full-map directory, with a very significant reduction on directory memory overhead.

By simulating with parallel applications from the SPLASH benchmark suite, Parthasarathy (1992) have compared the performance, scalability and resource usage of a single level scheme with those of a hierarchical protocol. The results have shown that performance depends on the data access patterns and degree of synchronization in the program. The full map scheme performs adequately with a low processor count or for programs with low synchronization. However, the hierarchical schemes scale better and perform well as the number of processor and the synchronization

overhead increases. Furthermore, good locality in a program makes a hierarchical protocol more attractive, reducing miss latencies and required network bandwidth.

Acacio et al. (2005) have presented a novel two-level directory architecture to eliminate the penalty caused by compressed directories in general. The proposed organization consists of a small full-map first-level directory and a compressed second-level directory. The proposals have been evaluated based on extensive execution-driven simulations (using RSIM) of a 64-node cc-NUMA multiprocessor. Results have demonstrated that a system with a two-level directory architecture achieves the same performance as a multiprocessor with a big and nonscalable full-map directory, with a very significant reduction of the memory overhead.

Li and John (2001) have proposed a new directory scheme, dubbed associative full map directory (*ADir$_p$NB*) which reduces the directory storage requirement. The proposed *ADir$_p$NB* uses one directory entry to maintain the sharing information for a set of exclusively cached memory blocks in a centralized linked list style. Their analysis has indicated that, on a typical architectural paradigm, *ADir$_p$NB* has reduced memory overhead of a traditional full map directory by up to 70-80 percent. In addition to the low memory overhead, they have shown that the proposed scheme can be implemented with appropriate protocol modification and hardware addition. Simulation studies have indicated that *ADir$_p$NB* can achieve a competitive performance with the *Dir$_p$NB*. Compared with limited directory schemes, *ADir$_p$NB* shows more stable and robust performance results on applications across a spectrum of memory sharing and access patterns due to the elimination of directory overflows.

Grahn (1995) have identified three problems: the number of cache misses, implementation cost of the directory and high design complexity in a directory-based protocol, and evaluated implementation and performance aspects of some design alternatives. His evaluation methodology has been based on program-driven simulation. To reduce the number of cache misses, a competitive-update policy has been proposed. The competitive-update policy has been shown to reduce both the read stall and execution times as compared to write-invalidate under a relaxed memory consistency model. However, update-based policies need more buffering and hardware support in the caches. To reduce the implementation cost of the

directory, which is proportional to the number of caches, an alternative directory organization has been proposed which distributes the directory information among the caches sharing the same memory block. To achieve a low write latency, the caches sharing a block have been organized in a tree. The caches have been linked into the tree in parallel with application execution to achieve a low read latency. Grahn has also evaluated a design alternative where the controller has been implemented using software handlers executed on the compute processor. This design alternative has been shown to be competitive with the baseline protocol. However, the performance of this alternative is more sensitive to other design choices, e.g., block size and latency tolerating techniques, than the baseline protocol.

Rhee and Lee (1999) have proposed a new directory cache coherence scheme named broadcast directory protocol, which can be used for large-scale mesh-connected multiprocessors using a deterministic routing algorithm and the wormhole routing. Several parallel programs of the SPLASH-2 suite have been executed on the simulator. For widely shared memory blocks, the network traffic has been significantly reduced due to the cache coherence transaction using only two worms for a set of sharers.

Kong et.al (2005) have proposed using physical address mapping on directories to significantly reduce directory size needed. They have shown that the reduction in the directory size using a well conceived address interleaving scheme could be quite substantial. It allows the directory size to be scaled according to the cache memory size instead of the main memory size and the system size, which is very attractive to large-scale systems. Their simulations have shown that the impact of the different address interleaving schemes on the system performance is usually minimal. The performance is, on the other hand, somewhat sensitive to the directory management overhead.

Kim et al.(2006) have proposed a directory-cache embedded switch architecture with distributed shared cache and distributed shared memory. It is able to reduce the number of home node cache accesses, which results in a reduction in the inter cache transfer time and the total execution time. Simulation results have

verified that the proposed methodology can improve performance substantially over a design in which directory caches are not embedded in the switches.

Several studies on the performance evaluation of the directory protocols take place in literature. For instance, Kong et al. (1999) have presented a non-blocking directory-based cache coherence protocol on an electrical based interconnection network to improve the performance of large-scale distributed shared-memory multiprocessors. In the proposed non-blocking directory protocol, all subsequent request can proceed without being blocked at the directory. The critical path of a data access and the amount of network transactions needed to complete a memory request (including the transactions needed to maintain coherence) have also been reduced. To support the non-blocking directory protocol, the history of data accesses has been maintained with a small history table at the directory. Using detailed simulations, they have been evaluate the performance of the protocol compared with other blocking directory protocols.

The research of Heinrich et al. (1999) has provided an implementation-based, quantitative analysis of the performance, scalability, and robustness of four scalable cache coherence protocols running on top of a single architecture, Stanford FLASH multiprocessor. The four coherence protocols examined are bit-vector/coarse-vector, dynamic pointer allocation, SCI, and COMA. In addition to measurements of the characteristics of protocol execution (e.g., memory overhead, protocol execution time, and message count) and of overall performance, they have examined the effects of scaling the processor count from 1 to 128 processors. The results have shown that, the optimal protocol changes for different applications and can change with processor count even within the same application.

Chaudhuri and Heinrich (2004) have claimed that Negative ACKnowledgments (NACKs) and subsequent retries, used to resolve races and to enforce a total order among shared memory accesses in DSM multiprocessors, not only introduce extra network traffic and contention, but also increase node controller occupancy, especially at the home. They have presented possible protocol optimizations to minimize these retries and offer a thorough study of the performance effects of these messages on six scalable scientific applications running

on 64-node systems and larger. The six programs have selected from the SPLASH-2 benchmark suite. There have been three complete applications (Barnes-Hut, Ocean, and Water) and three computational kernels (FFT, LU, and Radix-Sort). To eliminate NACKs, they have presented a mechanism to queue pending requests at the main memory of the home node and augment it with a novel technique of combining pending read requests, thereby accelerating the parallel execution for 64 nodes by as much as 41 percent (a speedup of 1.41) compared to a modified version of the SGI Origin 2000 protocol. They have further designed and evaluated a protocol by combining this mechanism with a technique that they call write string forwarding, used in the AlphaServer GS320 and Piranha systems. They have found that without careful design considerations, especially regarding atomic read-modify-write operations, this aggressive write forwarding can hurt performance. They have identified and evaluated the necessary micro-architectural support to solve this problem. They have compared the performance of these novel NACK-free protocols with a base bitvector protocol, a modified version of the SGI Origin 2000 protocol, and a NACK-free protocol that uses dirty sharing and write string forwarding as in the Piranha system.

# 3. EVALUATION METHODOLOGY

For investigating the performance of interconnection network systems, synthetic traffic has been extensively used in the technical literature. A simulation using synthetic traffic signifies running a simulator for a large number of cycles (simulated time) to get performance results with the network in steady state. Although this methodology is not a completely realistic assumption, it can be useful in order to stress the communication capabilities and it is also very desirable to observe the behavior of the network as a part of a whole system executing real loads (Puente et al., 2001). The results obtained with synthetic traffic are expected to indicate the minimum level of performance the network could provide under actual traffic (Miguel-alonso et al., 2008). For some applications such as Radix or LU (part of the widely-known SPLASH benchmark suite (Singh et al., 1992)), it has been shown to be a reasonable expectation (Puente et al., 2001).

## 3.1. Synthetic Traffic Workloads

Synthetic traffic is characterized by three distributions: temporal (that determines the packet inter-arrival times, i.e. bernoulli, constant bursts, Markov etc.), spatial (destination of packets, i.e. uniform, hot region, zipf, constant (perfect shuffle, transpose etc.)) and packet-size (i.e. constant, uniform, and polynomial etc.) (Ridruejo and Miguel-alonso, 2005).

### 3.1.1. Temporal

In literature, a range of options can be found for the temporal distribution of packet 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 packets 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 work, exponential distribution is used for packet generation. The packets keep walking around the network until the simulation ends.

### 3.1.2. Spatial

In this work, spatial traffic pattern is evaluated into two categories: distribution of home nodes and distribution of owner and sharer nodes. Uniform distribution is used for selecting owner and sharer nodes. Each node selects its destinations randomly in a packet-by packet basis (Izu et al., 2005b). For the distribution of home nodes, bit-reversal (BR) and perfect-shuffle (SH), which are a collection of well-known bit permutations, are used. Bit permutations are a subset of permutations in which the destination address is computed by permuting and selectively complementing the bits of the source address (Dally and Towles, 2004). For example if the source bit address is ($s_3$, $s_2$, $s_1$, $s_0$), the destination for a bit-reversal traffic pattern is ($s_0$, $s_1$, $s_2$, $s_3$) and for a perfect shuffle traffic pattern is ($s_2$, $s_1$, $s_0$, $s_3$). Uniform traffic (UN), and hot region traffic (HR (Blumrich et al., 2003)) are also included in the simulation. Blumrich et al. (2003) create hot region traffic by selecting the destinations of 25% of the packets randomly within a small “hot” contiguous sub-mesh region consisting of 12.5% of the machine. The destinations of the remaining 75% of the packets 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).

### 3.1.3. Packet Size

Networks often impose a maximum packet size and messages have to be segmented to fit in several of those packets (Izu et al., 2005a). For this reason, most studies are done with fixed-size messages of 8-32 bits [Callahan and Goldstein,

1995; Puente et al., 2001; Shin and Pinkston, 2003]. However, in general, real applications running on parallel computers do not use fixed-size messages. In this study, five types of variable sized packets are created to walk around the network. The sizes of the packets are determined according to the ratio T/R, while the types of the packets are determined according to the parameters (i.e. data, address) that the packet includes. Both the sizes and types of the packets are extensively described in Section 3.2.2.

## 3.2. Simulation Framework

### 3.2.1. Simulation Tool and System Architecture

OPNET Modeler (OPNET Technologies Inc., 2007) 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 messages between functional elements, called modules. Each module can generate, send and receive packets from 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.

Initially, a 4-node DSM-based SOME-Bus system is designed by using OPNET Modeler as shown in Figure 3.1. After testing the system and ensuring that it works correctly, it has been expanded to 64 nodes. The processor, cache controller, directory controller and channel controller are represented by queue modules with the symbols “pr”, “cac”, “dir” and “ch”, respectively. The function of the “hub” is to receive data and coherence messages from the channel module and send them to the other queue modules. The underlying process model that controls queue modules’ behavior is OPNET’s built-in acb_fifo model, which can be seen in Figure 3.2. 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.

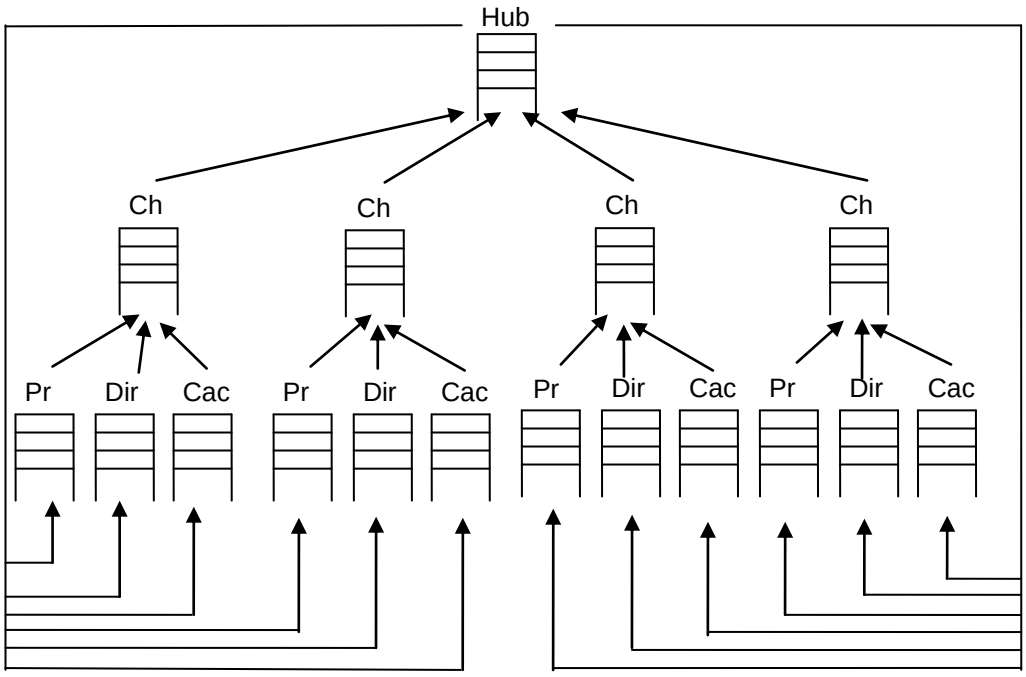



Figure 3.1. Node model of a 4-node DSM-based SOME-Bus system in OPNET Modeler

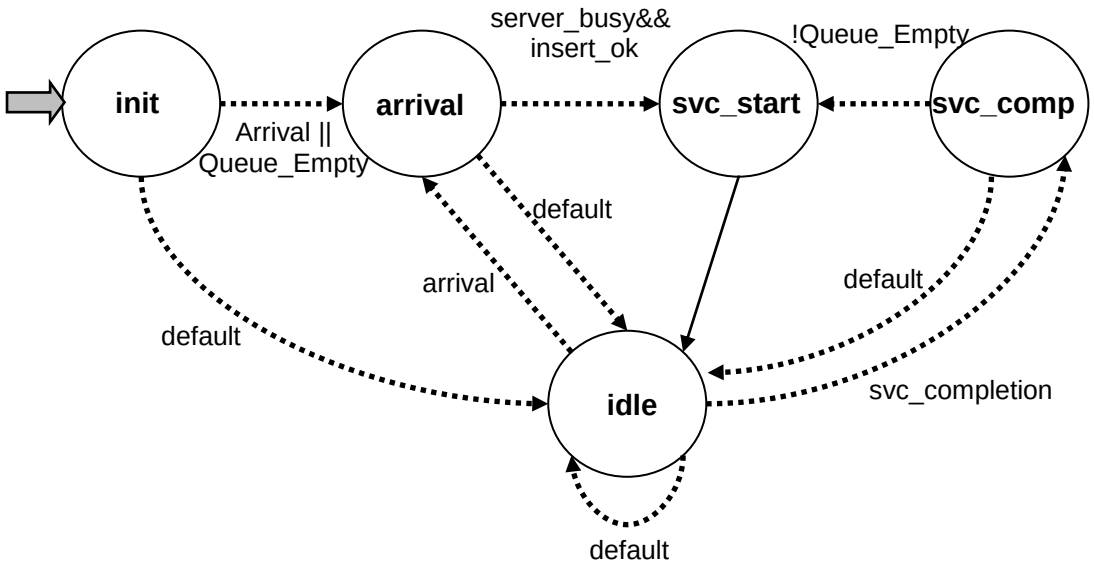



Figure 3.2. State diagram of acb_fifo model in OPNET Modeler

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.

For a data request to a block in shared or unowned directory state, the block is supplied from the home node memory. The home node sends data acknowledge message with data to the requesting node. If the block is in exclusive directory state, the owner node is determined among remote nodes with uniform distribution by the home directory. In intervention forwarding and reply forwarding protocols, the home directory sends downgrade write back request message to the owner node, which has

the modified block in its local cache. However, in strict request-response protocol, the home directory sends the address of the owner node to the requestor node. Then, the requester node's directory sends downgrade write back request message to the owner node. When the owner node's cache receives downgrade write back request message and if the protocol is reply forwarding or strict request-response, the owner node directly sends downgrade write back acknowledge message with data to the requesting directory and sends a revision message to the home directory. If the protocol is intervention forwarding, the owner node sends downgrade write back acknowledge message with data to the home directory.

For an ownership request to a block in unowned directory state, the home directory sends ownership acknowledge message with the requested block to the requesting node. The network transactions of an ownership request to a block in exclusive directory state is the same with the transactions of a data request to a block in exclusive directory state. The only difference is the type of the messages. However, if the block is in shared state; in intervention forwarding protocol, the home directory sends invalidation messages to the sharer nodes and waits for acknowledgments from them. In reply forwarding protocol, the home directory initially sends the addresses of the sharers to the requestor node's directory and then sends invalidation messages to the sharer nodes. In strict request-response protocol, the home directory sends the addresses of the sharers to the requestor node's directory and then the requesting directory sends invalidation messages to the sharer nodes. When the sharer's cache receives an invalidation message, it sends the invalidation acknowledge message to the home directory if the protocol is intervention forwarding or it sends the invalidation acknowledge message to the requesting directory if the protocol is reply forwarding or strict-request response. In intervention forwarding protocol, when all invalidation acknowledge messages are received by the home directory, the home directory sends ownership acknowledge message with data if needed, to the requestor node. In reply forwarding and strict request-response protocol, when all invalidation acknowledge messages are received by the requestor directory, the home directory sends owner acknowledge message to

local processor and data to local cache if needed, and then, it sends a revision message to the home directory.

Another type of message in the system is generated when a cache gets a block but has no empty space to put it in. At this time, the cache has to remove a random block, and notify the home directory of the block about this operation.

### 3.2.2. System parameters

The traffic generation method used in this work is extensively described in Section 3.1. For making the experiments reproducible, the rest of the parameters used in this simulation must be described. Other major parameters of the simulation, which can be seen in Table 3.1, 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 packets, probability of a cache being full and probability of a block being in various states.

Table 3.1. System Parameters

| Parameter | Value |
|---|---|
| Thread number in each node | 4 |
| Mean thread run time | exponentially distributed with a mean of 100 clock cycles |
| Mean channel service time for a Type-1 packet | varies between 1.25 and 25 |
| Mean channel service time for a Type-2 packet | varies between 5 and 100 |
| Mean channel service time for a Type-3 packet | varies between 6.25 and 125 |
| Mean channel service time for a Type-4 packet | varies between 2.5 and 50 |
| Probability of write (ownership) request – P(W) | 0.2 , 0.4 , 0.6 |
| Probability of upgrade ownership request | 0.2 |
| Probability of a block being in modified state – P(M) | 0.2 , 0.4 , 0.6 , 0.8 |
| Probability of a block being in unowned state | 0.1 |
| Probability of the requestor being the only sharer | 0.15 |
| Owner node selection | Uniformly distributed |
| Probability of a cache being full | 0.15 |
| Sharer count | 3 |

The processor at each node is assumed to be executing a program with four 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 whereas the probability that a block is found in modified state, P(M), takes the values 0.2, 0.4, 0.6, and 0.8. 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 three.

A packet consists of two parts. The header contains the destination address, the source address, the requested operation type (read, write, invalidate, etc.) and the payload contains the transported data as indicated in (Ye et al., 2004). The channel service time of a packet varies according to the size (type) of the packet. As is seen in Table 3.2, the simulated system includes four types of packets.

Table 3.2. The types of packets in the system

| **Packet Type** | # of destination addresses in header | Payload |
|---|---|---|
| Type 1 (packet without data) | 1 | - |
| Type 2 (packet with data) | 1 | + |
| Type 3 (packet with sharer list&without data) | Sharer number | - |
| Type 4(packet with sharer list & data) | Sharer number | + |

The channel service time of a packet that does not carry data (Type-1) is a quarter of that of a packet that carries data (Type-2). This ratio is consistent with the packet sizes used in (Ye et al., 2004) and (Lee et al., 2000). In (Ye et al., 2004), short packets are the packets with no payload (for instance, memory access request packets), while long packets are the packets with payloads (for instance, data fetch packets). Short packets have a size of 16 bytes and for estimating the average delay on the network, they used 64 bytes as the size of a long packet. Lee et al. (2000) simulated a 64 node DSM system, in which they used 32 bytes for a cache line (payload in packet) and 12 bytes for an invalidation message for conventional unicast, which corresponds to the Type 1 packet in the simulation of this thesis. Therefore, the size of a packet with data used in (Lee et al., 2000) is 44 bytes, which is approximately four times more than a packet without data.

For a packet carrying a sharer list without data (Type-3), the channel service time is two times more than a packet that does not carry data. (Dai and Panda, 1996a) and (Dai and Panda, 1996b) assert that a unicast control message (Type-1) have a size of 4 bytes and a multicast control message (Type-3) have a size of 6 bytes, so the ratio of Type-3 message to Type-1 message is 1.5. In the simulation of this thesis, this ratio is round to 2.

For a packet carrying data with a sharer list (Type-4), the channel service time is five times more than a packet that does not carry data (Type-1). Type-4 can be seen as a combination of the payload and the header part of Type-3. The header size of Type-3 is two times more than the header size of Type-1. The payload is three times more than the size of Type-1, because the size of Type-2 is four times more than the size of Type-1. Therefore, the size of Type-4 is five times more than the size of Type-1.

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 × 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.

# 4. RESULTS AND DISCUSSION

Three different scenarios in OPNET Modeler are created to investigate the performance of strict request-response, intervention forwarding and reply forwarding protocols. For each protocol, simulation has been run for 1 hour, and performance measures such as average processor utilization and average network latency are collected for different spatial patterns and for different values of T/R, P(M) and P(W). Average processor utilization refers to average fraction of time that threads are executing and average network latency 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.

## 4.1. Theoretical Analysis

Before investigating the performance of the protocols, it would be benefical to perform simple theoretical analysis to validate the simulation results reported in Section 4.2. Figure 4.1 to Figure 4.3 show the network path of a packet for strict request-response, intervention forwarding and reply forwarding protocols for different types of requests to different types of cache blocks (shown as request/block). It is assumed that the size of Type 1 packet is *x*. Thus, the sizes of Type 2, Type 3 and Type 4 packets are *4x*, *2x* and *5x*, respectively. In the Figures, 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 Figure 4.1a, Figure 4.2a and Figure 4.3a, for a read request to a shared or invalid cache block or for a write request to an invalid cache block, 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 all protocols. Reply forwarding protocol has the minimum number of transactions and the load on the network over all protocols for both read and write requests to a Modified cache block (Figure 4.3b). For this situation, although the number of

network transactions is the same for strict requst response and intervention forwarding protocols, the load on the network of strict request response protocol is less than that of intervention forwarding protocol (Figure 4.1b and Figure 4.2b). As is seen in Figure 4.1 c, Figure 4.2 c and Figure 4.3c for a write request to a Shared block, the number of total network transactions is the same for all protocols. The load on the network is equal for reply forwarding and strict request response protocols while it is the minimum for intervention forwarding protocol. Intervention forwarding protocol utilizes the home node at most, whereas strict request response protocol directs the tasks to the requestor node. The home node of reply forwarding protocol provides direct communication from the owner node(s) to the requestor node. In such a case, the busyness rate of the home node for the protocols from the highest to lowest is intervention forwarding, reply forwarding and strict request response, respectively.

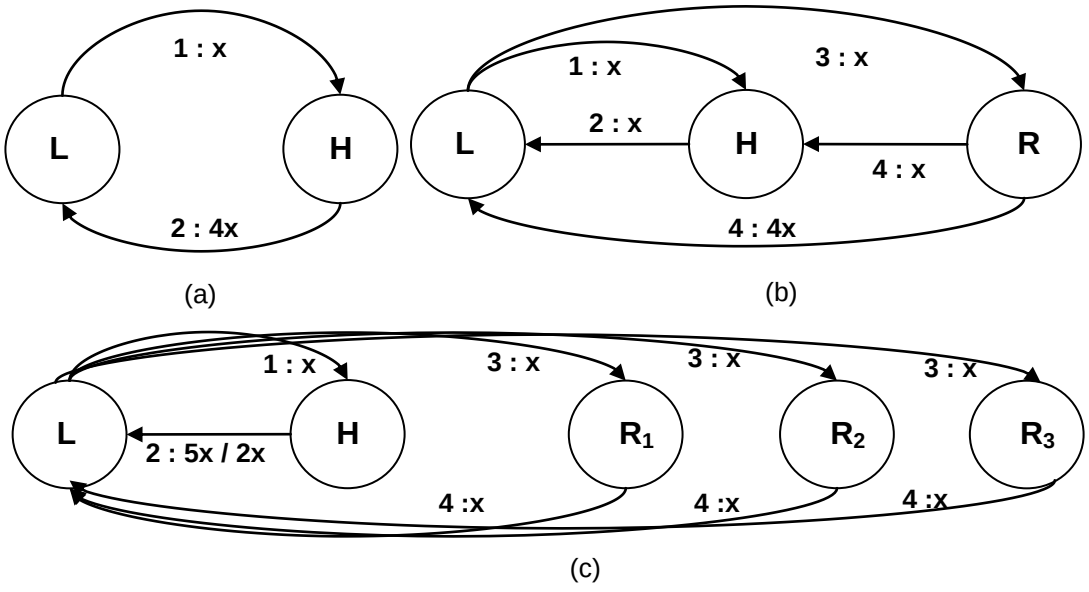



Figure 4.1. Message path of strict request response protocol for (request/block) (a) read / Shared & Invalid and write / Invalid, (b) read & write / Modified and (c) write / Shared

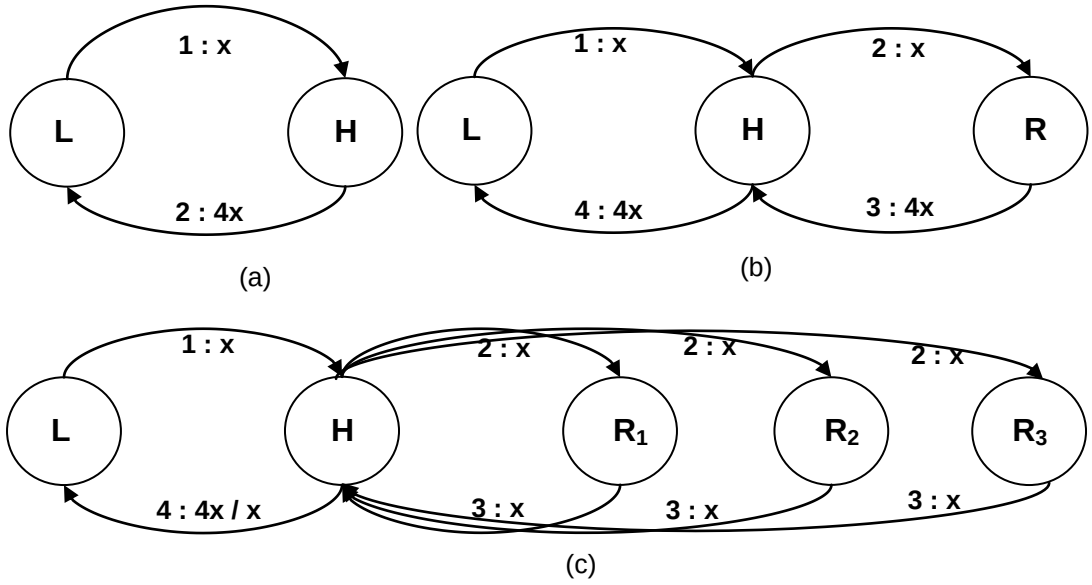



Figure 4.2. Message path of intervention forwarding protocol for (request/block) (a) read / Shared & Invalid and write / Invalid, (b) read & write / Modified and (c) write / Shared

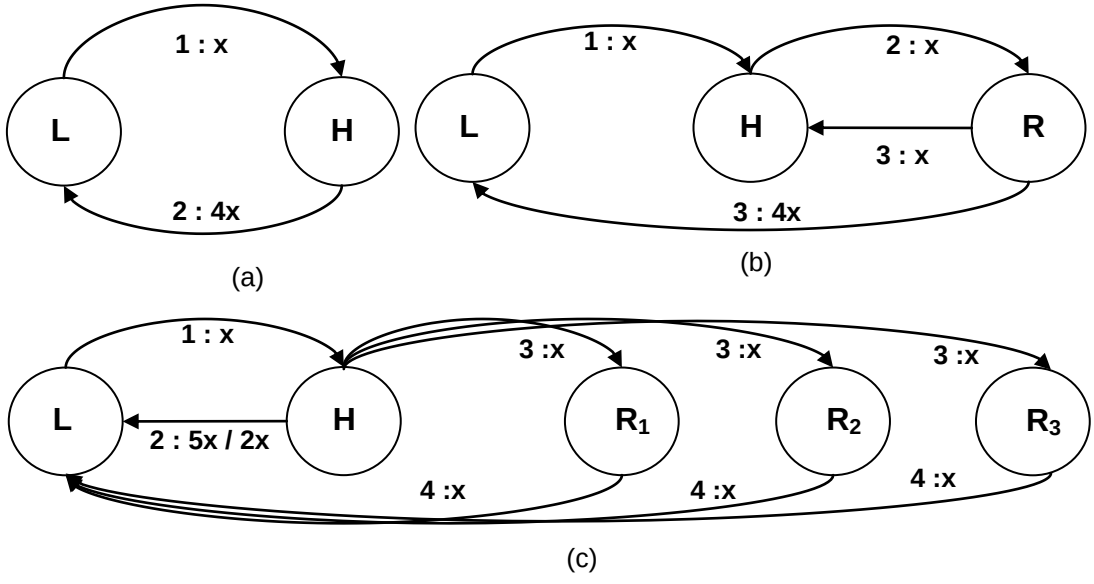



Figure 4.3. Message path of reply forwarding protocol for (request/block) (a) read / Shared & Invalid and write / Invalid, (b) read & write / Modified and (c) write / Shared

## 4.2. Simulation Results

Figure 4.4 through Figure 4.19 show the average processor utilizations and the average network latencies for different spatial patterns and for different values of T/R, P(M) and P(W). In the Figures, "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).

It can be easily seen from Figure 4.4 and Figure 4.5 that for P(M) = 0.2, the performance of the three protocols is almost the same for P(W) = 0.2 and under BR (Figure 4.4) or SH traffic (Figure 4.5). As P(W) increases, intervention forwarding performs better (higher processor utilization and lower network latency) than the other protocols, while strict request-response performs better than that reply forwarding. The gap between the performances of the three protocols increases as P(W) increases. Under HR traffic (Figure 4.6), for P(M) = 0.2, strict request-response performs the best for middle values of T/R and for P(W) = 0.2. Reply forwarding becomes the worst while the other protocols perform similar as P(W) increases. As is seen from Figure 4.7, for P(M) = 0.2, the performance of the three protocols is almost the same for UN. Intervention forwarding becomes the best and reply forwarding becomes the worst as P(W) increases.

When P(M) is considerably low (i.e. P(M) = 0.2), the criterion that creates the performance difference between the three protocols is the write request to a Shared block (see Figure 4.1 to Figure 4.3). Therefore, in the following interpretations, the increase in P(W) implies the increase in write request to a Shared block. For intervention forwarding protocol, although the load on the network is the minimum (Figure 4.2c), the burden on the home node is the maximum over all protocols. The load on the network is the same for reply forwarding and strict request-response protocols. However, strict request-response reduces the burden on the home node. Under BR and SH traffic, the requestor node always selects the same home node. For this reason, the criterion to determine the performance of the best protocol is the load on the network instead of the burden on the home node. Therefore, especially as

P(W) increases, intervention forwarding performs the best over all protocols. Strict request-response performs better than reply forwarding because the home node is less busy for strict request-response protocol. The burden on the home node has the maximum importance for HR. Thus, strict request-response performs the best when P(W) is the minimum. As P(W) increases, owing to the advantage of the load on the network of intervention forwarding, the performance of strict request response and intervention forwarding becomes almost the same. In such a case, reply forwarding protocol performs the worst. When all of the performance benchmarks are taken into account together for UN, there is not a clear difference between the performances of the three protocols for low values of P(W), as expected. As P(W) increases, intervention forwarding protocol, which has the minimum load on the network, performs the best while strict request-response, which distracts the home node at least, performs better than reply forwarding.

As is seen from Figure 4.8, for P(M) = 0.4 and under BR traffic, the performance of the three protocols is almost the same for low-to-mid values of P(W). For higher values of P(W), intervention forwarding yields the best performance over all protocols and strict request-response tends to perform better than reply forwarding. The results for PS is similar to the results for BR. The only difference is that for high values of P(W), the gap between the performances of the three protocols under PS traffic is less than that of BR traffic (Figure 4.9). Under HR traffic (Figure 4.10), for P(M) = 0.4, intervention forwarding is clearly the worst for P(W) = 0.2, while the other protocols yield comparable performances. As P(W) increases, strict request-response becomes the best while reply forwarding performs almost the same with intervention forwarding. It can be easily seen from Figure 4.11 that the three protocols yield comparable performances under UN traffic, for P(M) = 0.4 and for all P(W) values.

When the results for P(M) = 0.4 are compared with the results for P(M) = 0.2, it can easily be said that reply forwarding yields better performance for P(M) = 0.4 than for P(M) = 0.2 under BR and PS traffic. However, the increase in P(W) decreases the performance of reply forwarding more than that of other protocols. Under HR traffic, the increase in P(M) decreases the performance of intervention

forwarding protocol at most because intervention forwarding has the maximum load on the network for the requests to a Modified cache block and burdens the home node more than the other protocols. Strict request-response is the best protocol under HR traffic because the burden on the home node is substantially reduced. Under UN traffic, as under BR and PS traffic, reply forwarding is the most affected by the negative impact of the increase in P(W). That is because the load on the network of reply forwarding protocol is more than that of intervention forwarding for a write request to a Shared block and reply forwarding burdens the home node more than strict request-response.

Under BR and PS traffic (Figure 4.12 and Figure 4.13), for P(M) = 0.6, reply forwarding performs the best for low values of P(M). As P(W) increases, the performance of the three protocols becomes almost the same. As is seen from Figure 4.14, for P(M) = 0.6 and under HR traffic, intervention forwarding performs the worst for all P(W) values while the other protocols yield comparable performances. It can be said from Figure 4.15 that reply forwarding performs the best and intervention forwarding performs the worst under UN traffic, for P(M) = 0.6 and P(W) = 0.2. The gap between the performances of the three protocols decreases as P(W) increases.

The results for P(M) = 0.6 show that as P(M) increases, owing to the advantage of the requests to Modified cache blocks, reply forwarding performs the best under BR, PS and UN traffic. However, the increase in P(W), deprives of this advantage and eventually reply forwarding yields comparable performance with the other protocols for BR and PS while the gap between the performance of reply forwarding and the other protocols decreases for UN. Intervention forwarding that burdens the home node for all situations at most, performs by far worse than the other protocols under HR traffic. Even the increase of P(W) could not prevent intervention forwarding from being the worst. Since high P(M) value rolls back the performance of intervention protocol, which is the most disadvantaged protocol for requests to a Modified cache block state in all respects. This situation also demonstrates that the effect of P(W) on the performance of the protocols is vitiated by the inrease in P(M). Strict request-response yields comparable performance with

reply forwarding under HR. This is due to the fact that strict request-response does not exhaust the home node while reply forwarding is completely the most favorable protocol for the requests to a Modified cache block state.

As is seen from Figure 4.16 and Figure 4.17, for P(M) = 0.8 and under BR and PS traffic, reply forwarding is the best for all P(W) values. Strict request-response follows reply forwarding and intervention forwarding is the worst. The gap between the performance of reply forwarding and the performances of the other protocols slightly decreases as P(W) increases. It can be easily indicated from Figure 4.18 that the performance of intervention forwarding protocol is considerably lower than that of other protocols for all P(W) values while reply forwarding tends to perform better than strict request-response for low P(W) values under HR traffic and for P(M) = 0.8. However, strict request-response yields comparable performance with reply forwarding as P(W) increases. Under UN traffic (Figure 4.19), the performances of the three protocols for P(M) = 0.8 and for all P(W) values from highest to lowest are for reply forwarding, strict request-response and intervention forwarding, respectively.

It can be said from the results for P(M) = 0.8 that intervention forwarding performs the worst while reply forwarding performs the best for all traffic patterns. This conclusion is inevitable for intervention forwarding because of its drawbacks concerning Modified cache block states and the home node congestion explained above. The gap between the performance of intervention forwarding and the performances of the other protocols is maximum under HR traffic, where the congestion on the home is the most essential factor that imposes the performance. Owing to the high value of P(M), reply forwarding that is the best protocol for Modified cache block states achieves to be the best protocol for all situations. Even the increase in P(W) could not prevent reply forwarding from being the best protocol.

The performance of the three protocols decreases as P(M), P(W) or T/R increases regardless of the traffic pattern. For P(M) = 0.2, the three protocols perform the worst under HR traffic and yield comparable performances under other traffic patterns for all P(W) values. As well as the difference between the protocol

performances under UN, BR and PS traffic is so low, the performances from the highest to lowest is under BR, PS and UN, respectively. The gap between the performances of the three protocols under HR and that of under other traffic patterns decreases as P(W) increases. Under all traffic patterns, as P(W) increases, the performances of the three protocols exhibit a considerably high decrease for P(M) = 0.2. As P(M) increases, the gap between the performance of the three protocols decreases for different values of P(W) and for P(M) = 0.8, the performances are the almost the same for all P(W) values. This result also claims that the decrease of the protocol performances originating from P(W) is reduced by the increase of P(M) value. The decrease in the performance of reply forwarding is the minimum as P(M) increases, and is the maximum as P(W) increases. Since strict request-response reduces the burden on the home node, it shows quite good performance under HR traffic. When P(M) has low values, intervention forwarding performs preferable, as P(W) increases.

For supporting the comments made above by using numerical data, Table 4.1 and Table 4.2 are created by using the Figures 4.4 through 4.19. Table 4.1 and Table 4.2 show the increase rate of average network latency and the decrease rate of average processor utilization i.e. the decrease rate of the protocol performance for each directory protocol under each traffic pattern by defining the case P(M) = 0.2 and P(W) = 0.2 as a reference point. “int” represents intervention forwarding, “rep” represents reply forwarding and “st” represents strict request-response protocol.

In literature, the three protocols are investigated only for read requests (Culler et al., 1999) and it has been denoted that reply forwarding performs better than intervention forwarding while strict request-response is the worst. In this thesis, after reorganizing the three protocols for write requests and investigating the system for various parameters, it has been shown that strict request-response yields better performance than expected, especially under HR traffic. The reason is the fact that the size of the packets going through the network is as important as the size of network transaction count. Although the number of the network transactions in strict-request response is higher, the number of packets that carry data in this protocol is less than the number of packets that carry data in intervention forwarding.

In conclusion, a hybrid protocol can exhibit the best performance by using intervention forwarding protocol for low P(M) values with high P(W) values, reply forwarding for high P(M) values and strict request-response for the rest. If the traffic pattern is taken into account, strict request-response is the best protocol under HR traffic.

# 5. CONCLUSION

DSM allows ease of programming in large-scale parallel computer systems by hiding the message passing mechanism. Mapping the shared logical address space on local memories and keeping it consistent requires significant traffic on the interconnection network. There is extensive research in development of directory-based cache-coherence protocols that may reduce the network traffic; however there is a lack of research in performance comparison of different directory protocols on a broadcast-based optical interconnection network supporting DSM.

This thesis evaluates, via simulation, the performance of three directory protocols; strict request-response, intervention forwarding and reply forwarding, on the SOME-Bus multiprocessor architecture under different traffic patterns and for varying values of the important DSM parameters such as the T/R, P(M) and P(W) after reorganizing the three protocols, which are designed only for read requests, for write requests. The results reveal that as P(W), P(M) or T/R increases, the performance of the three protocols decreases for all traffic patterns. The increase in P(M) reduces the effect of P(W) on the performances of the protocols. Intervention forwarding protocol performs the best for low P(M) values and high P(W) values while reply forwarding protocol yields the best performance for high P(M) values. Under HR traffic, strict request-response performs the best over all protocols. The three protocols perform the worst under HR traffic, while yielding comparable performances under other traffic patterns for all P(M) and P(W) values. Also, the gap between the performances of the three protocols under HR and that of under other traffic patterns decreases as P(W) increases.

Although this study is of value to literature by clarifying the performances of the three protocols on an optical interconnection network under varying values of DSM parameters, there are some limitations, which constitutes the future work of this study. The simulation technique is commonly used for performance evaluations in literature. However, the actual results can only be obtained by running real applications. Furthermore, for achieving realistic performance results for the protocols real traffic patterns should be used instead of synthetic traffic patterns.

Another limitation of this study is using only one architecture for evaluating the performances of the protocols. Although the properties of the SOME-Bus architecture distinguish it from other optical networks examined in the past, the performances of the protocols under other architectures should be investigated for conducting a generally accepted research.

## REFERENCES


ACACIO, M.E., GONZÁLEZ, J., GARCÍA, J.M. AND DUATO, J., 2005. A Two-Level Directory Architecture for Highly Scalable cc-NUMA Multiprocessors. IEEE Transactions on Parallel and Distributed Systems, 16(1): 67 - 79.

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.

BARROSO, L.A., GHARACHORLOO, K., BUGNION, E., 1998. Memory System Characterization of Commercial Workloads. Proceedings of the 25th Annual International Symposium on Computer Architecture, 3-14.

BIANCHINI, R., CARRERA, E.V., 2001. Designing and Evaluating a Cost-Effective Optical Network for Multiprocessors. Journal of Parallel and Distributed Computing, 61:178-201.

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

CALLAHAN, T.J., GOLDSTEIN, S.C., 1995. NIFDY: A Low Overhead, High Throughput Network Interface. Proc. of the 22nd Annual International Symposium on Computer Architecture (ISCA '95), Santa Margherita Ligure, Italy, 230-241.

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.

CHAMBERLAIN, R.D., FRANKLIN, M.A., BAW, C.S., 2002. Gemini: An Optical Interconnection Network for Parallel Processing. IEEE Transactions on Parallel and Distributed Systems, 13:1038-1055.

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., 2004. The Impact of Negative Acknowledgments in Shared Memory Scientific Applications. IEEE Transactions on Parallel and Distributed Systems, 15(2): 134-150.

CHAUDHURI, M., HEINRICH, M., HOLT, C., 2003. Latency, Occupancy, and Bandwidth in DSM Multiprocessors: A Performance Evaluation. IEEE Transactions on Computers, 52(7):862-880.

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., 1996a. Building Efficient Limited Directory-Based DSMs: A multidestination Message Passing Based Approach. Technical Report OSU-CISRC-4/96-TR21, Ohio State Univ., Columbus.

DAI, D. and PANDA, D.K., 1996b. Reducing Cache Invalidation Overheads in Wormhole Routed DSMs Using Multidestination Message Passing. Proc. Of the Int'l Conf. on Parallel Processing (ICPP'96), 138–145. (Dai and Panda, 1996b)

DAI, D., PANDA, D.I.S., 1997. How Much Does Network Contention Affect Distributed Shared Memory Performance?. Proceedings of the international Conference on Parallel Processing, 454.

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

DUATO, J., YALAMANCHILI, S., NI, L., 1997. Interconnection Networks: An Engineering Approach. Morgan Kauffman Publishers, San Francisco, USA.

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

ESKICIOGLU, M.R., MARSLAND, T. A., WEIWU, H. and WEISONG S., 1999. Evaluation of the JIAJIA Software DSM System on High Performance Computer Architectures. Proceedings of the 32nd Annual Hawaii International Conference on System Sciences, 8:8012.

GRAHN, H., 1995. Evaluation of Design Alternatives for a Directory-Based Cache Coherence Protocol in Shared-Memory Multiprocessors. Phd Thesis, Department of Computer Engineering, Lund University.

GRAVENSTRETER, G., MELHEM, R., 1998. Realizing Common Communication Patterns in Partitioned Optical Passive Stars (POPS) Networks. IEEE Transactions on Computers, 47(9): 998 - 1013.

GRUJIC, A., TOMASEVIC, M., MILUTINOVIC, V., 1996. A simulation study of hardware oriented DSM Approaches. Parallel & Distributed Technology: Systems & Applications, 4:74-83.

GUNAWAN, I., FARD, N.S., 2007. Reliability Evaluation of Optical Multistage Interconnection Networks. Proceedings of the ICTON Mediterranean Winter Conference, 1-4.

HA, J.H., PINKSTON, T.M., 1997. SPEED DMON: Cache Coherence on an Optical Multichannel Interconnect Architecture. Journal of Parallel and Distributed Computing, 41:78-91.

HAWKINS, C. and WILLS, D.S., 2006. Impact of Number of Angles on the Performance of the Data Vortex Optical Interconnection Network. Journal of Lightwave Technology, 24(9): 3288 - 3294.

HECHT, D., 2003. Fault-tolerant distributed shared memory on a broadcast-based interconnection architecture. Ph.D. thesis, Drexel University.

HEINRICH, M., SOUNDARARAJAN, V., HENNESSY, J., GUPTA, A., 1999. A Quantitative Analysis of the Performance and Scalability of Distributed Shared Memory Cache Coherence Protocols. IEEE Transactions on Computers, 48(2):205-217.

HEIRMAN, W., DAMBRE, J., ARTUNDO, I., DEBAES, C., THIENPONT, H., STROOBANDT, D. et al., 2008. Predicting the performance of reconfigurable optical interconnects in distributed shared-memory systems. Photonic Network Communications, 15:25-40.

HEIRMAN, W., DAMBRE, J., STROOBANDT, D., DEBAES, C., THIENPONT, H., CAMPENHOUT, J.V., 2005. Prediction Model for Evaluation of Reconfigurable Interconnects in Distributed SharedMemory Systems. Proceedings of the 2005 international workshop on System level interconnect prediction, 51-58.

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

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

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.

IZU, C., MIGUEL-ALONSO, J., GREGORIO, J.A., 2005b. Traffic Sources and their Influence on the Evaluation of Interconnection Network Performance. Technical report EHU-KAT-IK-03-05. Department of Computer Architecture and Technology, The University of the Basque Country.

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

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

KIM, D., KIM, M., SOBELMAN, G.E., 2006. DCOS: Cache Embedded Switch Architecture for Distributed Shared Memory Multiprocessor SoCs. IEEE International Symposium on Circuits and Systems (ISCAS), 4.

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

KONG, J., YEW, P. and LEE, G., 2005. Minimizing the Directory Size for Large-Scale Shared-Memory Multiprocessors. IEICE - Transactions on Information and Systems, e88–d(11): 2533-2543.

KONG, J., YEW, P. and LEE, G., 1999. Technical Report – A Non blocking directory protocol for Large Scale multiprocessors. University of Minnesota, Computer Science and Engineering Technical Report Abstract.

KUNKEL, S., ARMSTRONG, B., VITALE, P., 1999. System Optimization for OLTP Workloads. IEEE Micro, 56-64.

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

LAWRENCE, R., 1998. A Survey of Cache Coherence Mechanisms in Shared Memory Multiprocessors. Department of Computer Science, University of Manitoba, Manitoba, Canada.

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, T. and JOHN, L.K., 2001. ADirpNB: A Cost-Effective Way to Implement Full Map Directory-Based Cache Coherence Protocols. IEEE Transactions on Computers, 50(9): 921 - 934.

LINDQUIST, R.G., KULICK, J.H., COHEN, W.E., GAEDE, R.K., WELLS, E., ABUSHAGUR, M., et al., 1997. An Optoelectronic Design of the Simultaneous Optical Multiprocessor Exchange Bus (Some-Bus). SPIE Proceedings, 3025:12-14.

NARRAVULA, H., 2003. Performance of Parallel Algorithms on a Broadcast-Based Architecture. PhD thesis, Drexel University, Philadelphia, USA.

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.

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

NITZBERG, B., LO, V., 1991. Distributed Shared Memory: A Survey of Issues and Algorithms. Computer, 21:52-60.

OPNET Technologies, Inc., 2007. OPNET Modeler. OPNET University program, http://www.opnet.com/university_program.

OULD-KHAOUA, M., 1996. Comparative Evaluation of Hypermesh and Multi-Stage Interconnection Network. Computer Journal, 39(3):232.

PANIGRAHI, S.C., PAUL, S., SAHOO, G., 2006. OMTSE – An Optical Interconnection System for Parallel Computing. Proceedings of the International Conference on Advanced Computing and Communications, 626-627.

PARTHASARATHY, M., 1992. Implementing Shared Memory on Large Scale Multiprocessors. M.S. Thesis, University of Illinois.

PUENTE, V., IZU, C., BEIVIDE, R., GREGORIO, J.A., VALLEJO, F. and PRELLEZO, J.M., 2001. The Adaptative Bubble Router. Journal of Parallel and Distributed Computing, 61(9):1180–1208.

RAJASEKARAN, S., SAHNI, S., 1997. Sorting, Selection, and Routing on the Array with Reconfigurable Optical Buses. IEEE Transactions on Parallel and Distributed Systems, 8(11):1123-1132.

RANGANATHAN, P., GHARACHORLOO, K., ADVE, S., BARROSO, L., 1998. Performance of Database Workloads on Shared-Memory Systems with Out-of-Order Processors. Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems, 32(5):307-318.

RHEE, Y. and LEE, J., 1999. A Scalable Cache Coherent Scheme Exploiting Wormhole Routing Networks. Proceedings of the 5th International Symposium On High-Performance Computer Architecture, 223.

RIDRUEJO, F.J., MIGUEL-ALONSO, J., 2005. INSEE: an Interconnection Network Simulation and Evaluation Environment. Lecture Notes in Computer Science, Proc. Euro-Par 2005, 2005:1014 - 1023.

ROS, A., ACACIO, M.E. and GARCÍA, J.M., 2005. A Novel Lightweight Directory Architecture for Scalable Shared-Memory Multiprocessors. 11th International Euro-Par Conference, 582-591.

ROY, S., CHAUDHARY, V., 1999. Design Issues for a High-Performance Distributed Shared Memory on Symmetrical Multiprocessor Clusters. Cluster Computing, 2(3):177-186.

SCOTT, M.L., 2005. Shared memory computing on clusters with symmetric multiprocessors and system area networks. ACM Transactions on Computer Systems, 23:301-335.

SHARMA, N., CHADHA, D., CHANDRA, V., 2007. Performance evaluation of the augmented data vortex switch fabric: An all-optical packet switched interconnection network. Optical Switching and Networking, 4:213–224.

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.

SINGH, J.P., WEBER, W., GUPTA, A., 1992. SPLASH: Stanford parallel applications for shared memory. Computer Architecture News, 20(1):5–44.

SPEIGHT, E., BENNETT, J. K., 1997. Brazos: A Third Generation DSM System. Proc. of the USENIX/WindowsNT Workshop, 95-106.

SZYMANSKI, T., 1995. Hypermeshes Optical Interconnection Network for Parallel Computing. Journal Parallel Distributed Computing, 26(1):1-23.

WANG, H., PETRACCA, M., BIBERMAN, A., LEE, B.G., CARLONI, L.P., BERGMAN, K., 2008. Nanophotonic Optical Interconnection Network Architecture for On-Chip and Off-Chip Communications. Proceedings of the Optical Fiber communication/National Fiber Optic Engineers Conference (OFC 2008), 1-3.

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

YANG, J., WARD, M.O., RUNDENSTEINER, E.A., HUANG, S., 2003. Visual hierarchical dimension reduction for exploration of high dimensional datasets. Proceedings of Symposium on Visualization (VisSym ’03), Grenoble, France, 19–28.

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.

ZHU, M., NARRAVULA, H., KATSINIS, C., HECHT, D., 2004. A Channel Caching Scheme on an Optical Bus-Based Distributed Architecture. 18th International Parallel and Distributed Processing Symposium, 233.

## BIOGRAPHY

I was born in Adana, in 1984. I have completed my elemantary education at Celalettin Seyhan Primary Education School. I went to high school at ÇEAŞ Seyhan Anatolian High School and graduated from there with 1st degree in 1999. Then I deserved to educate in Adana Fen Lisesi. I have completed my university education at department of Computer Engineering of Istanbul Technical University (ITU) in 2007. Since 2007, I have been working as an expert computer engineer at Presidency of Çukurova University in Adana.

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

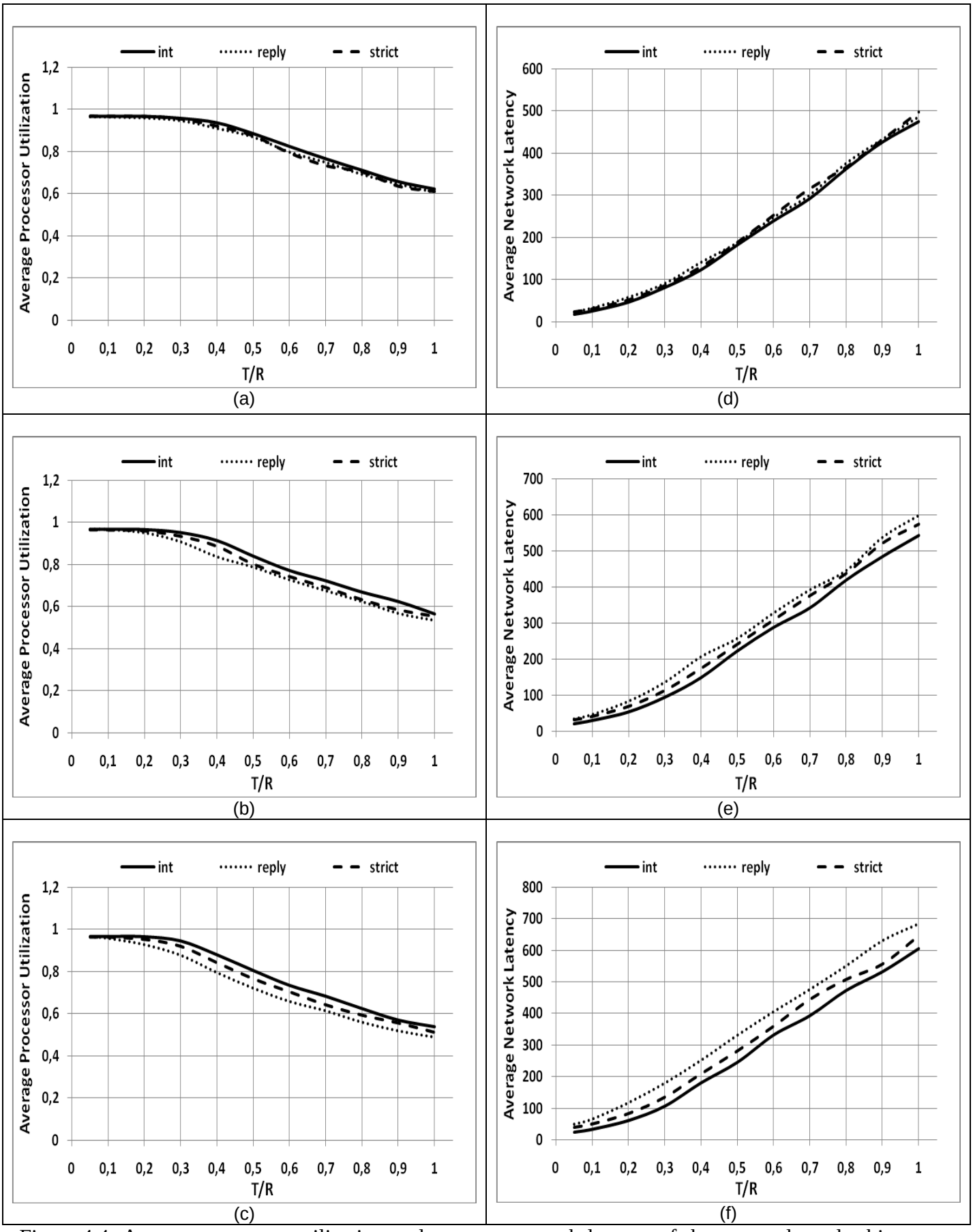



Figure 4.4. Average processor utilization and average network latency of the protocols under bit reverse traffic for P(M) = 0.2 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

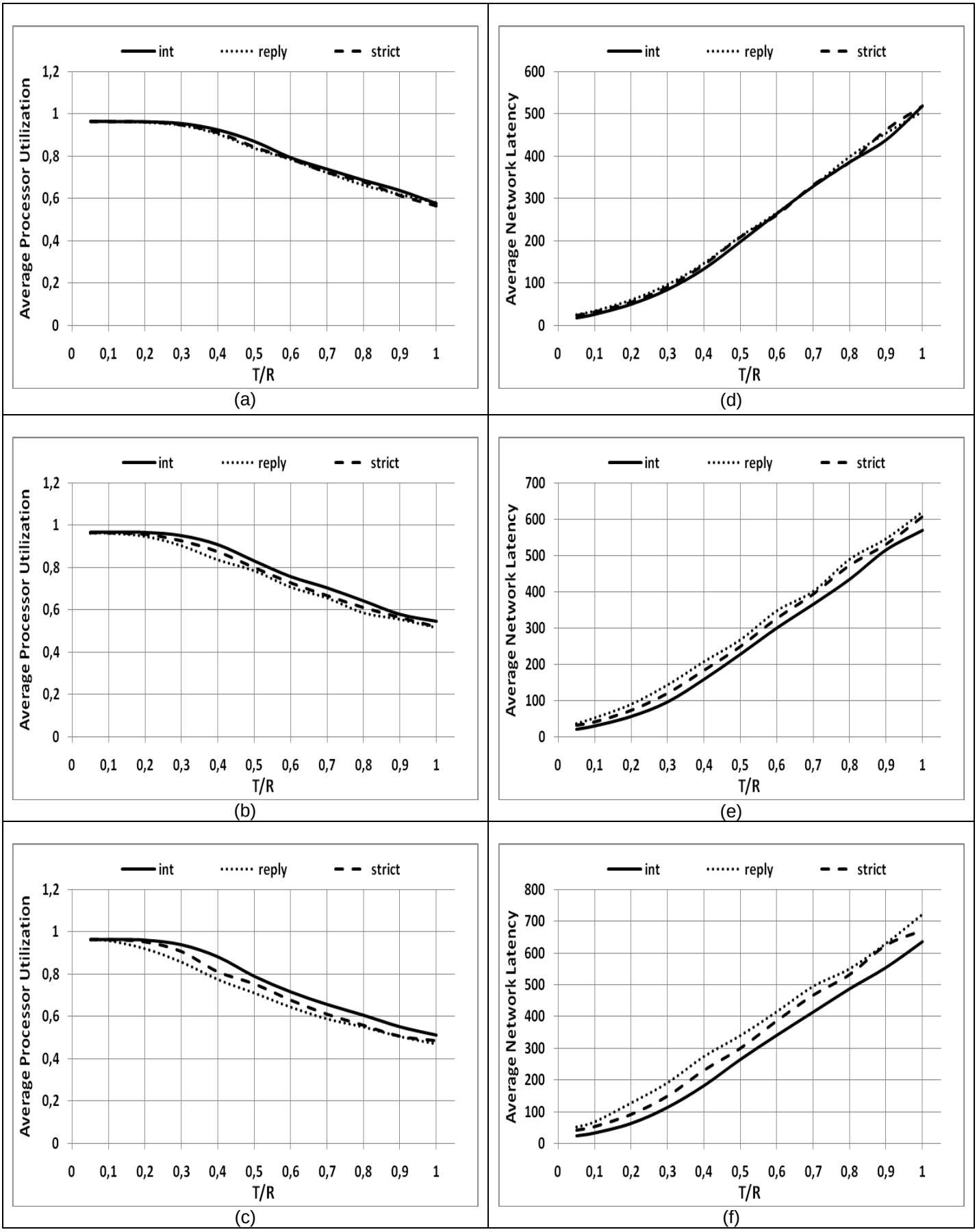



Figure 4.5. Average processor utilization and average network latency of the protocols under perfect shuffle traffic for P(M) = 0.2 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

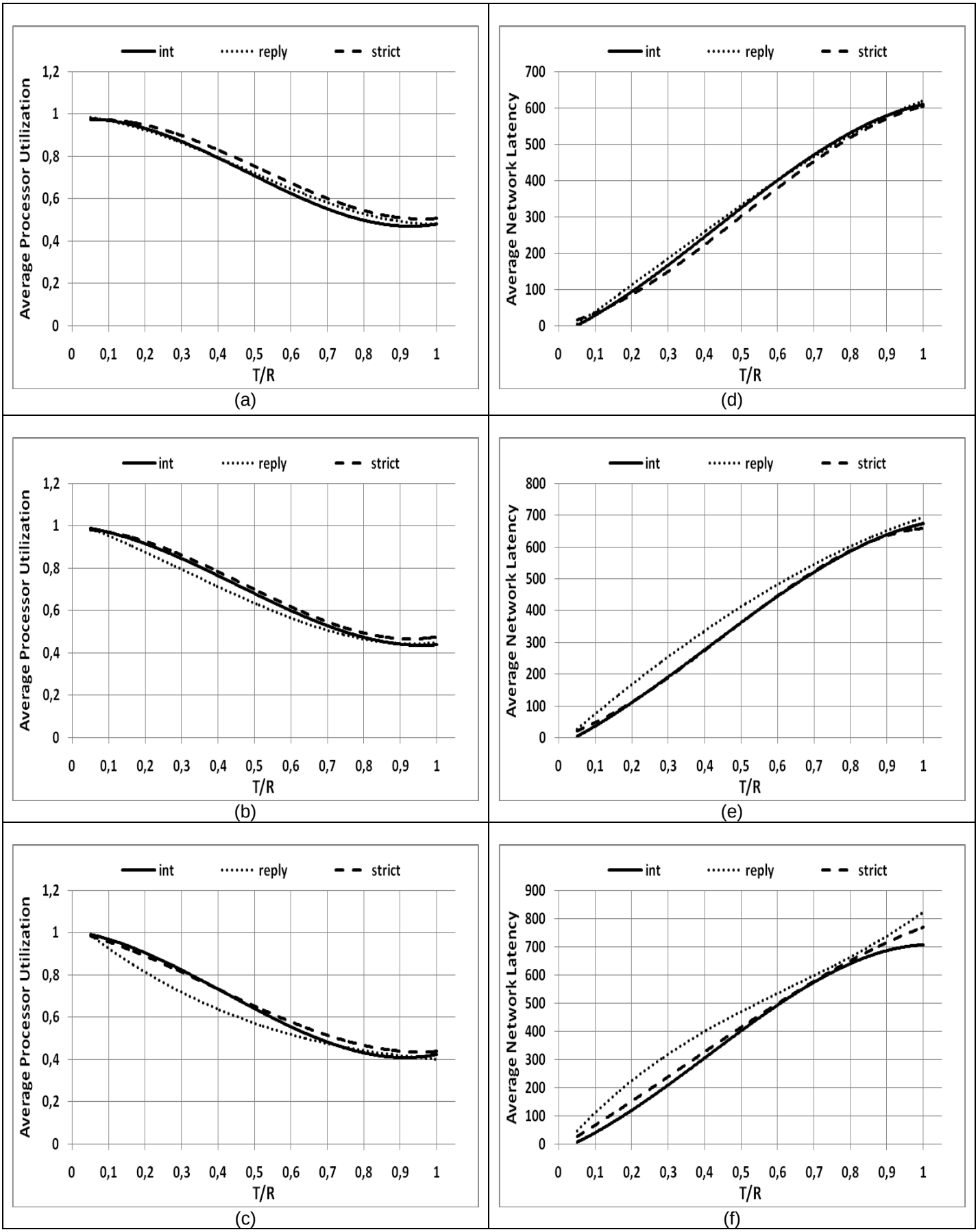



Figure 4.6. Average processor utilization and average network latency of the protocols under hot region traffic for P(M) = 0.2 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

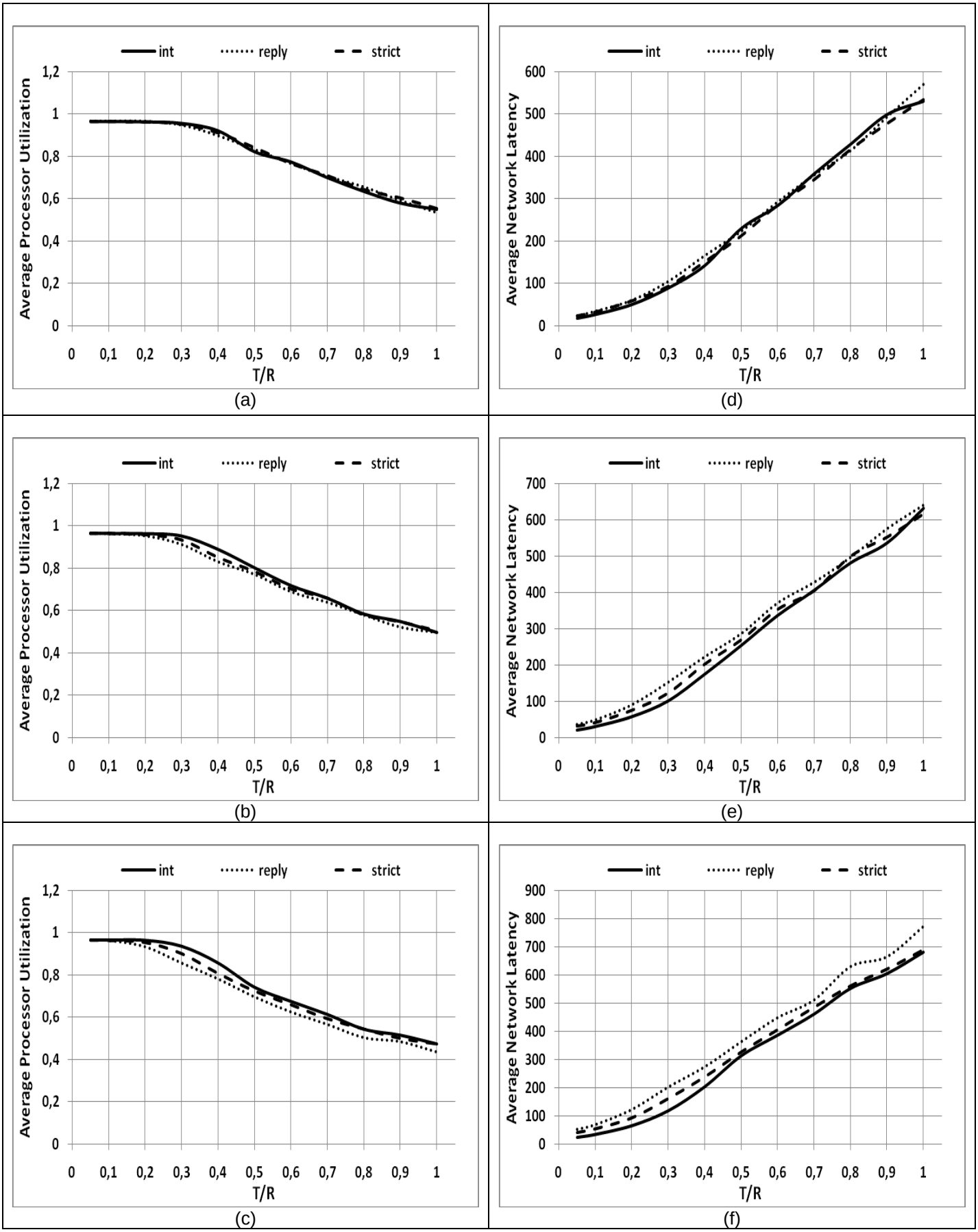



Figure 4.7. Average processor utilization and average network latency of the protocols under uniform traffic for P(M) = 0.2 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

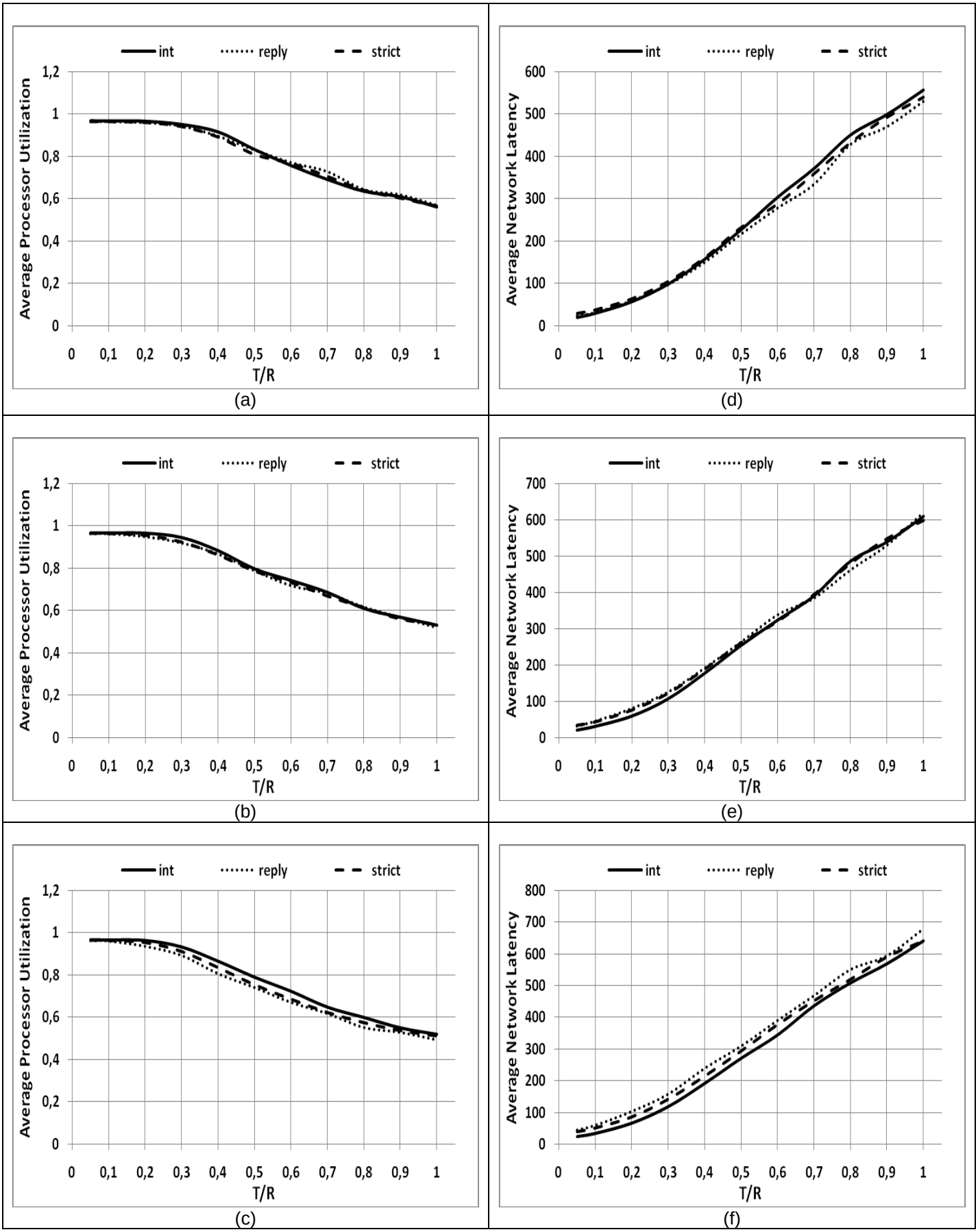



Figure 4.8. Average processor utilization and average network latency of the protocols under bit reverse traffic for P(M) = 0.4 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

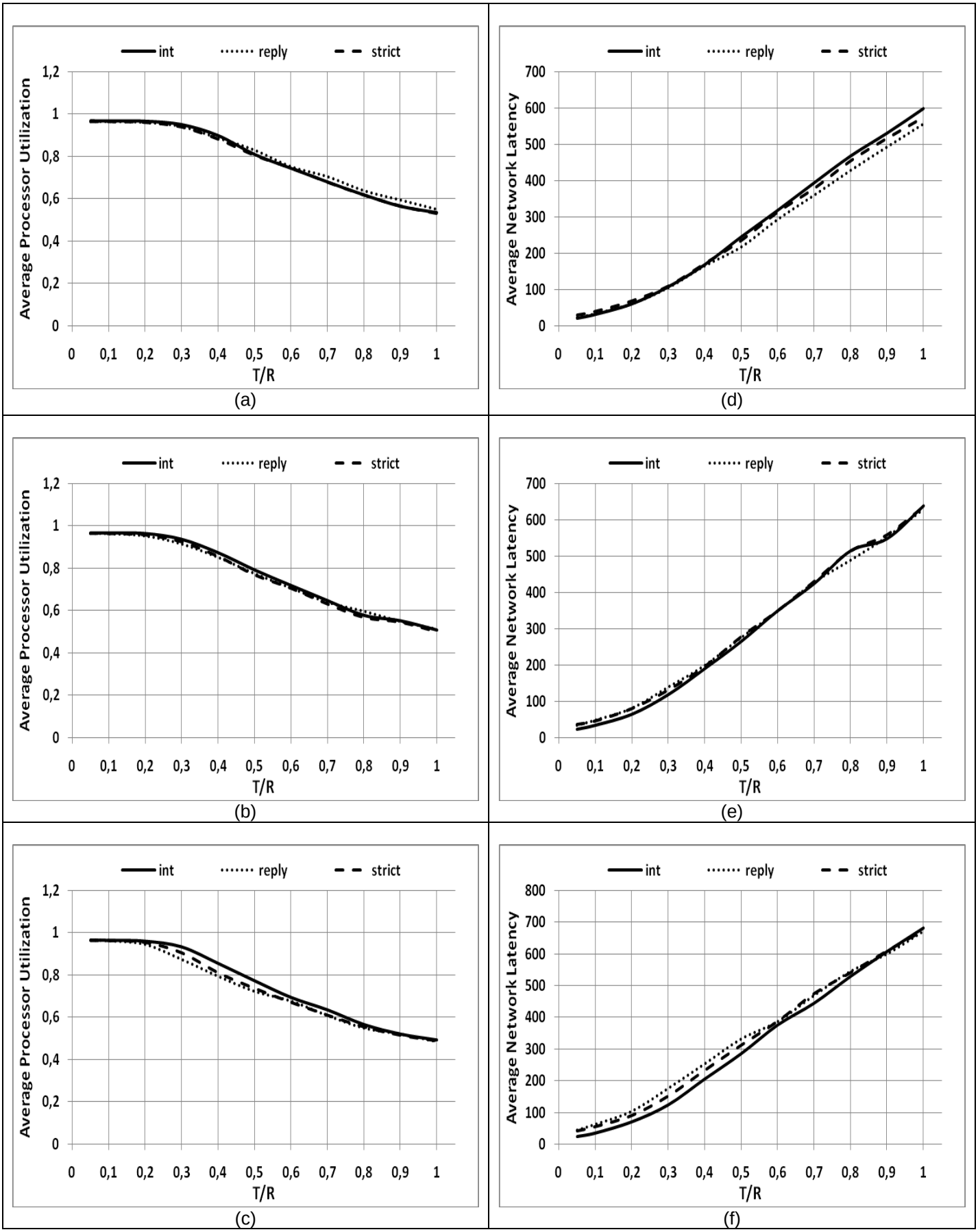



Figure 4.9. Average processor utilization and average network latency of the protocols under perfect shuffle traffic for P(M) = 0.4 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

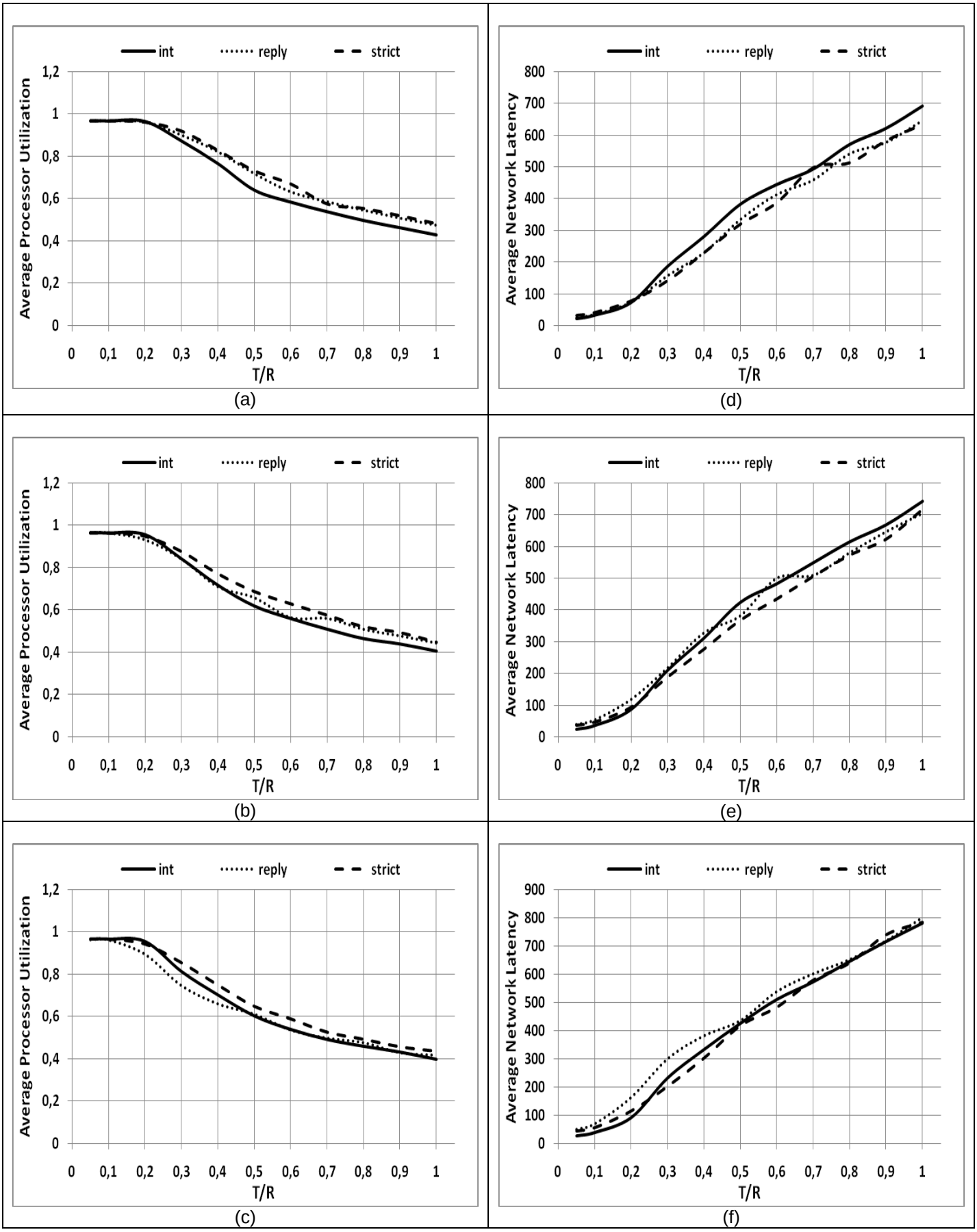



Figure 4.10. Average processor utilization and average network latency of the protocols under hot region traffic for P(M) = 0.4 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

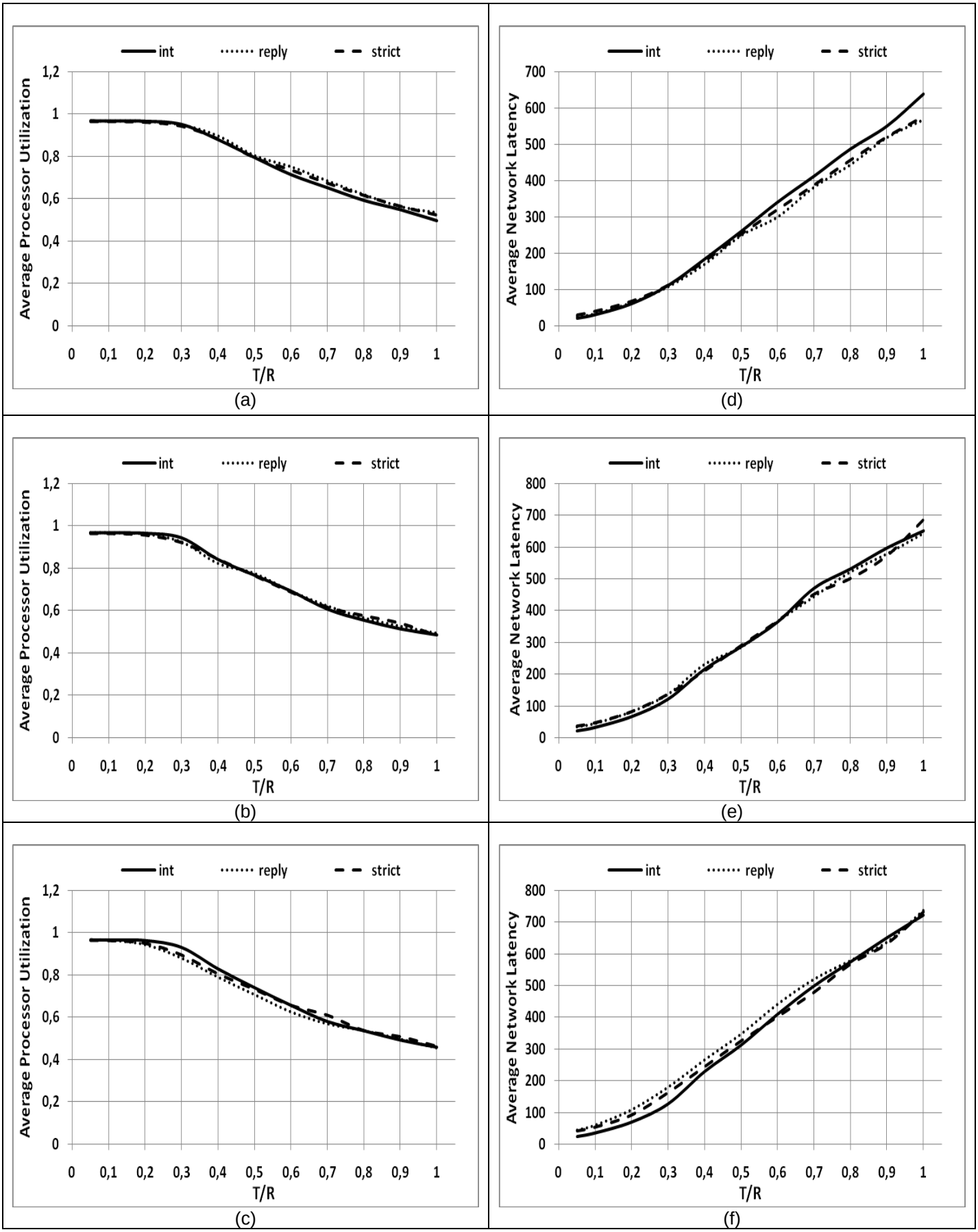



Figure 4.11. Average processor utilization and average network latency of the protocols under uniform traffic for P(M) = 0.4 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

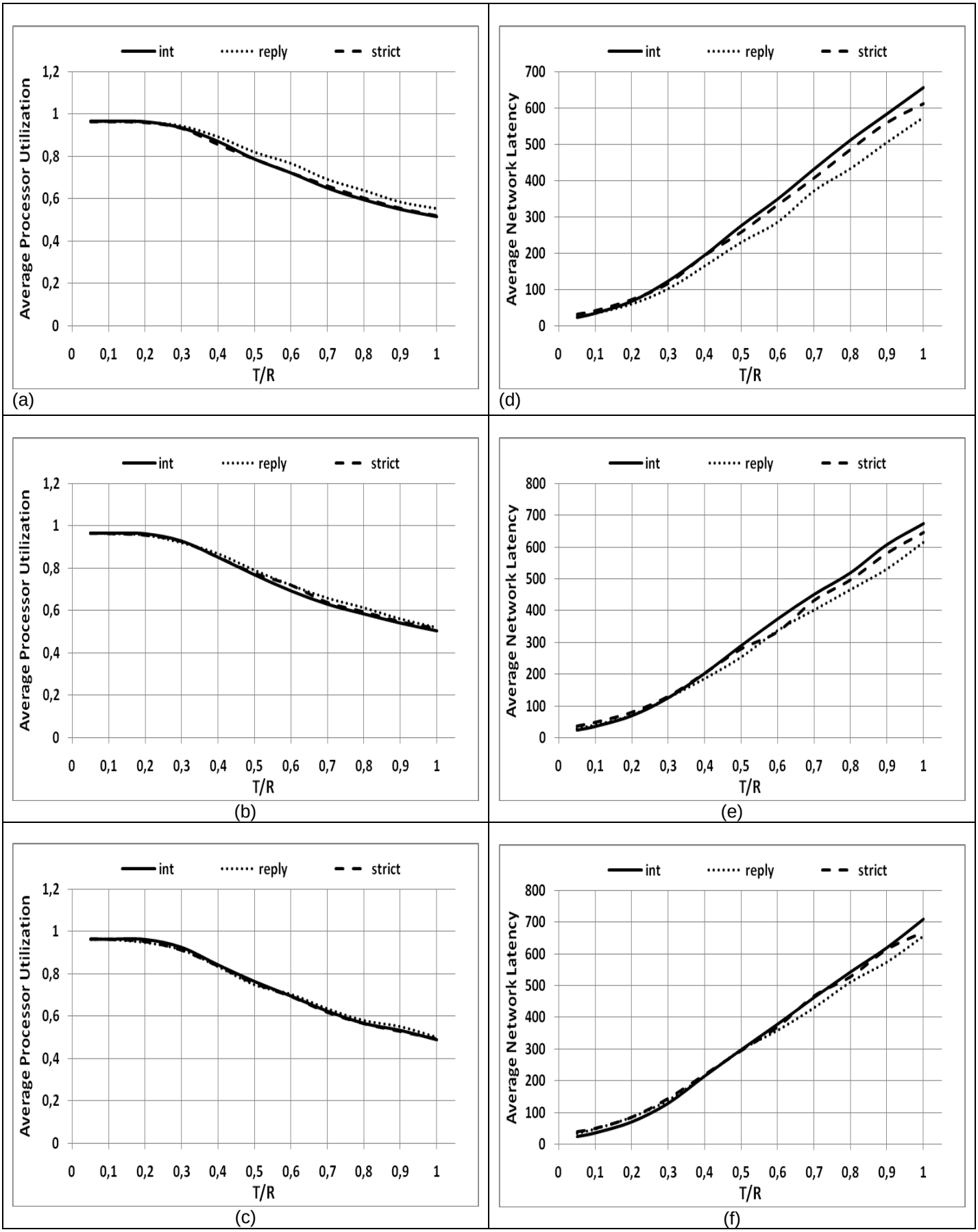



Figure 4.12. Average processor utilization and average network latency of the protocols under bit reverse traffic for P(M) = 0.6 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

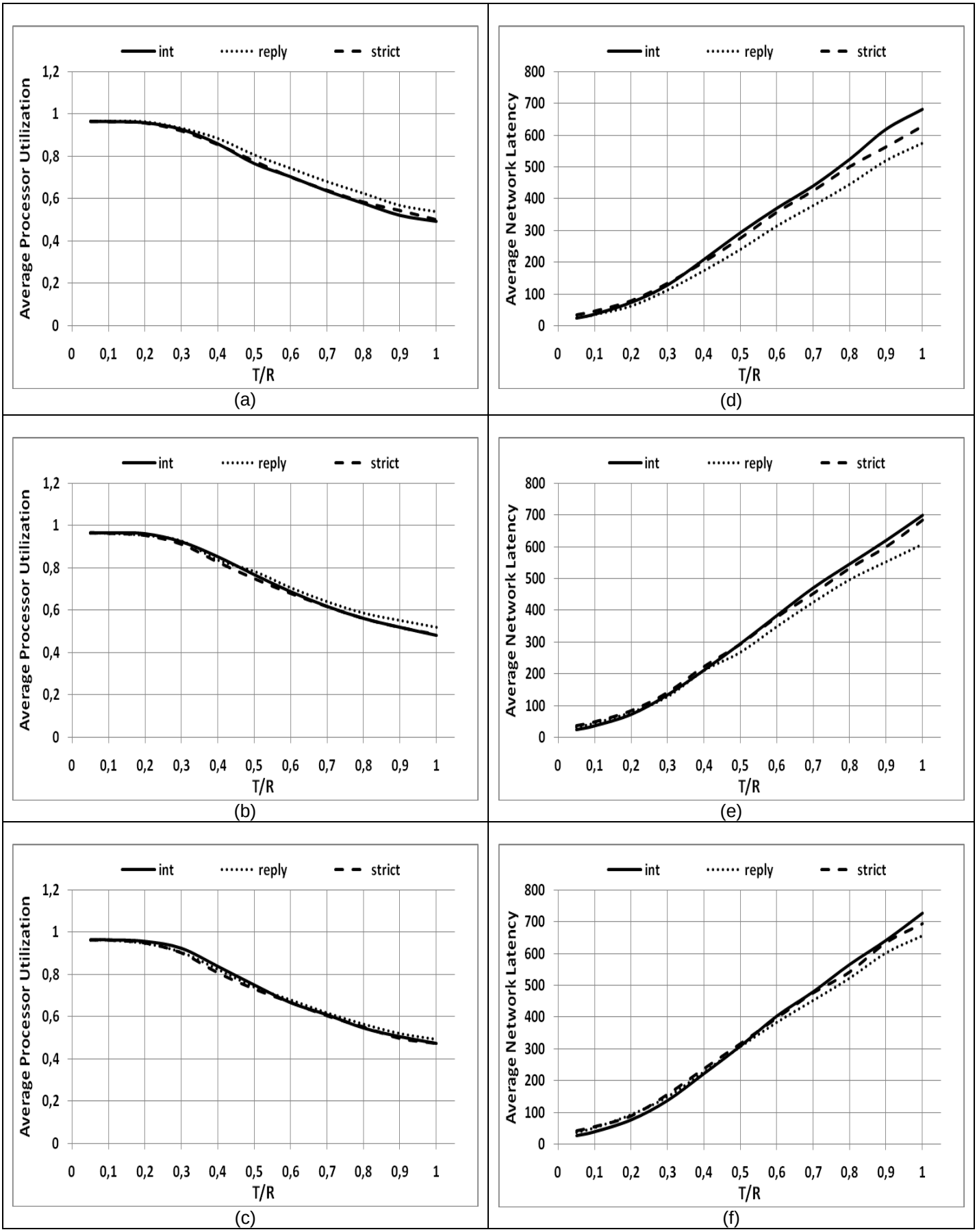



Figure 4.13. Average processor utilization and average network latency of the protocols under perfect shuffle traffic for P(M) = 0.6 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

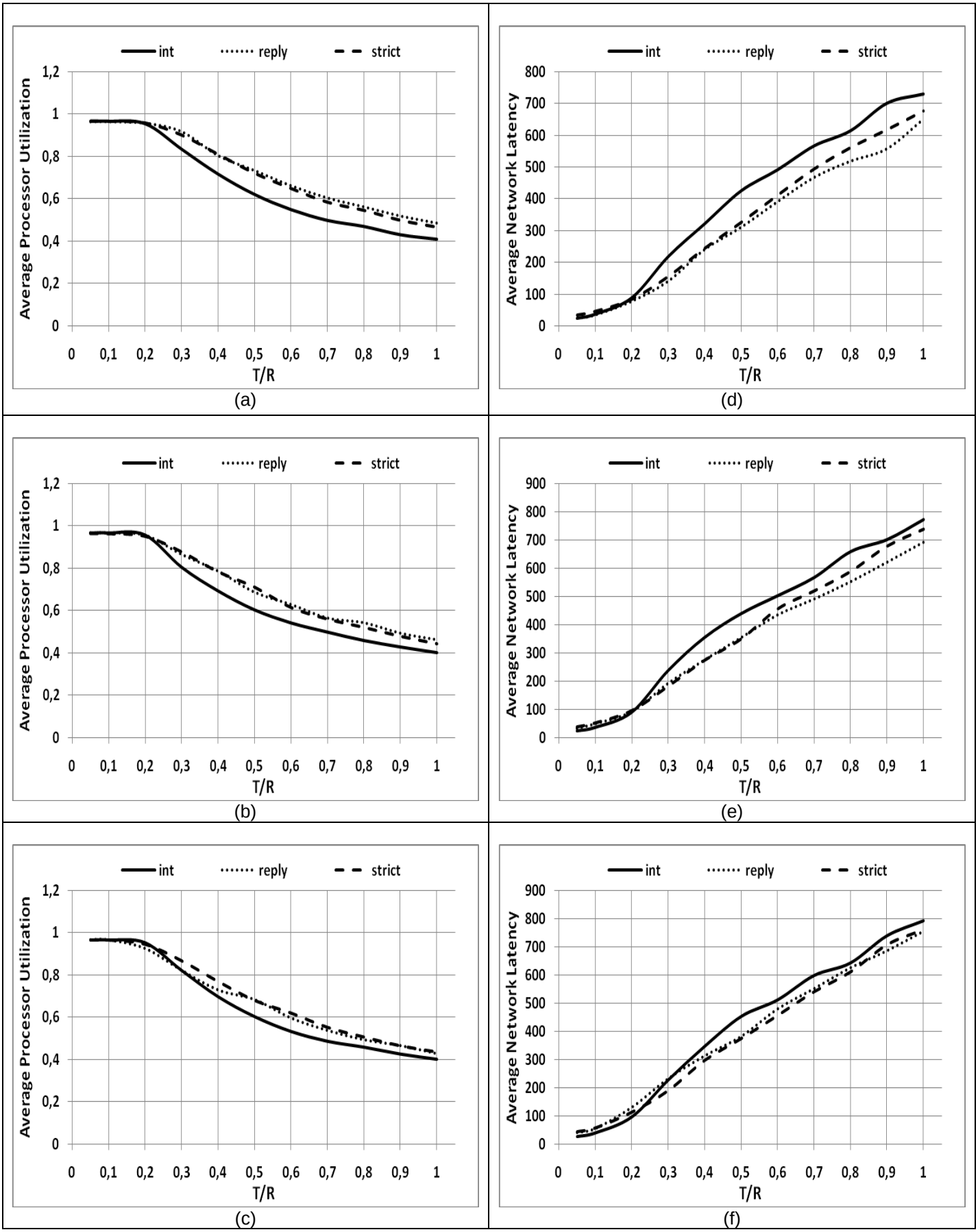



Figure 4.14. Average processor utilization and average network latency of the protocols under hot region traffic for P(M) = 0.6 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

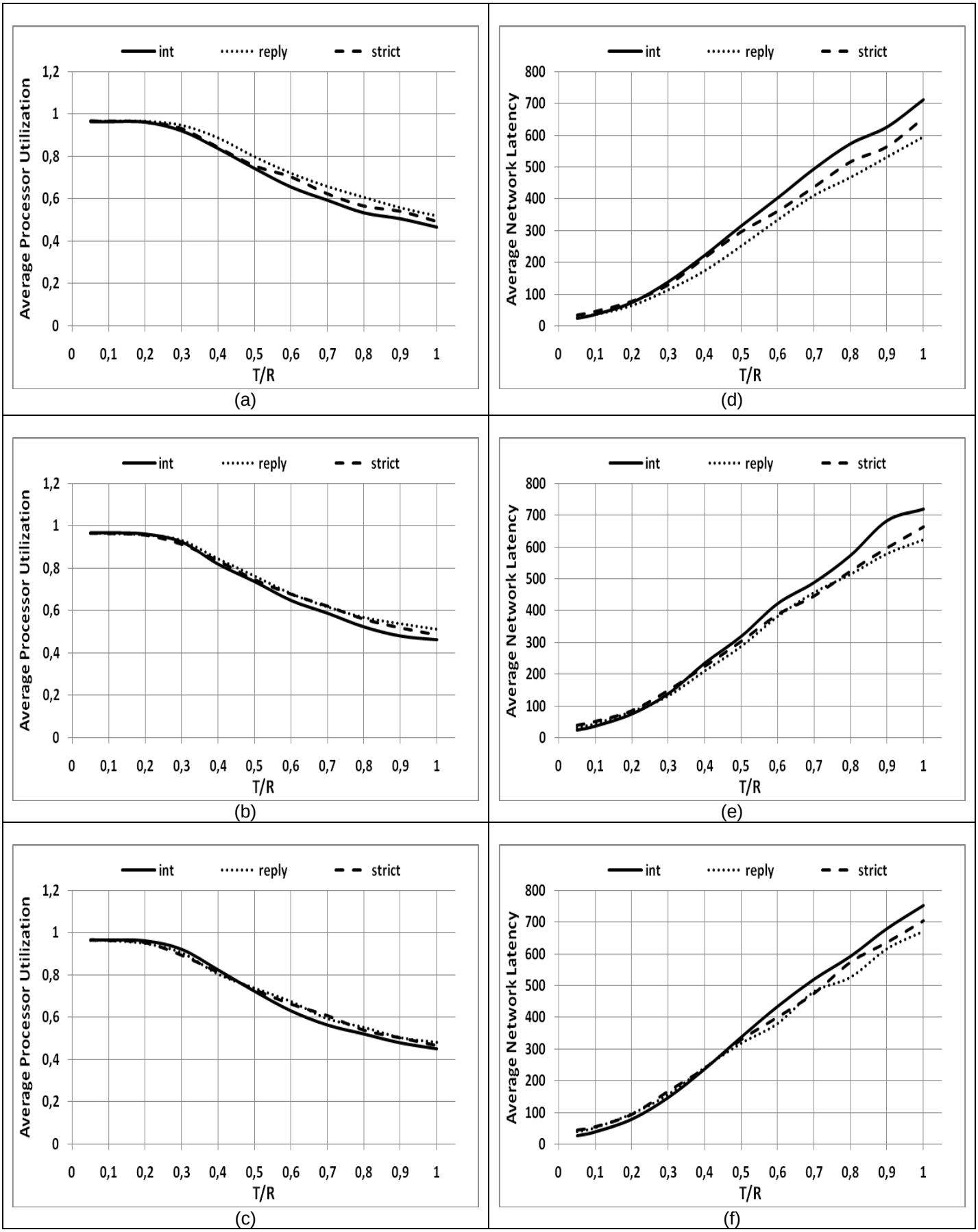



Figure 4.15. Average processor utilization and average network latency of the protocols under uniform traffic for P(M) = 0.6 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

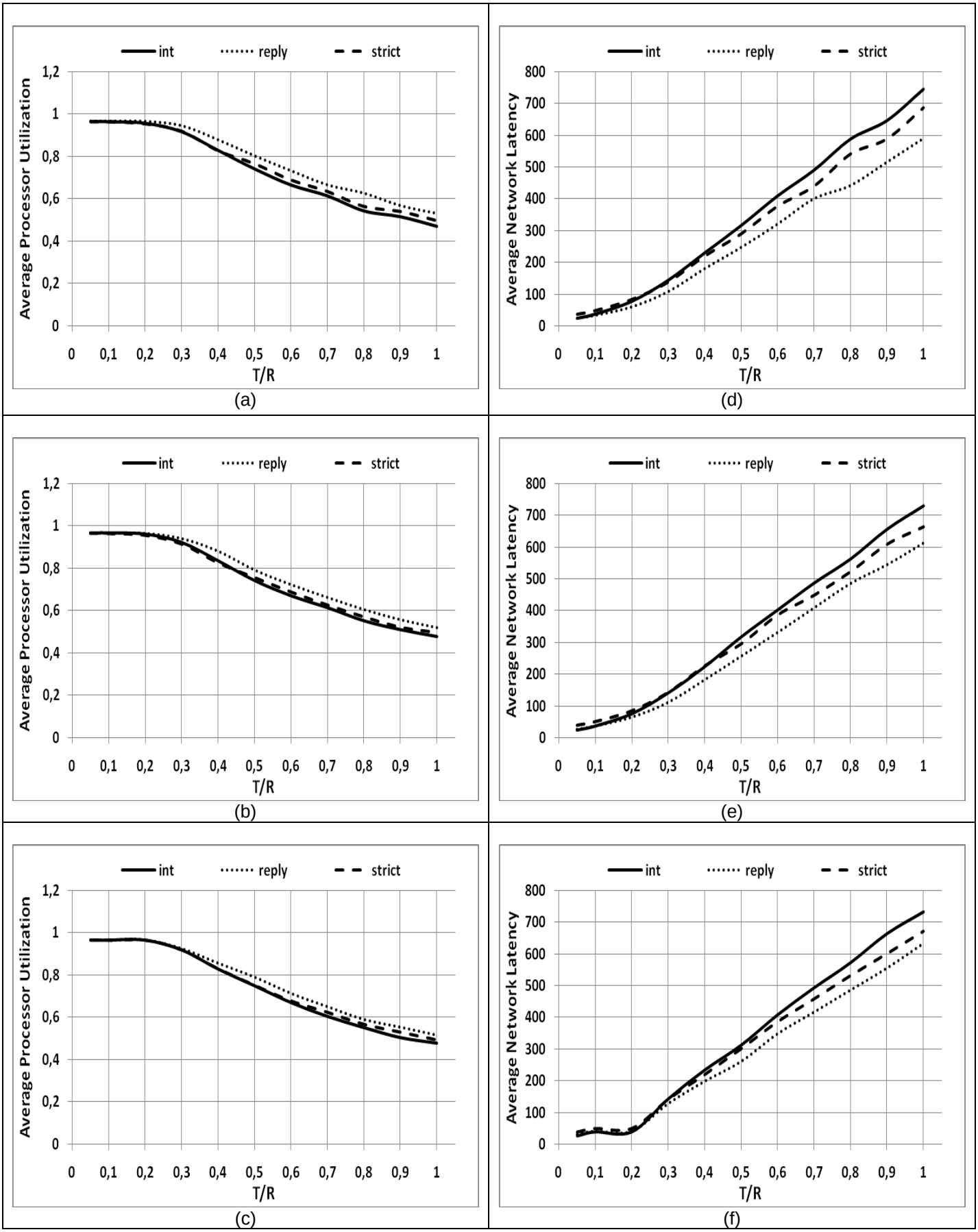



Figure 4.16. Average processor utilization and average network latency of the protocols under bit reverse traffic for P(M) = 0.8 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

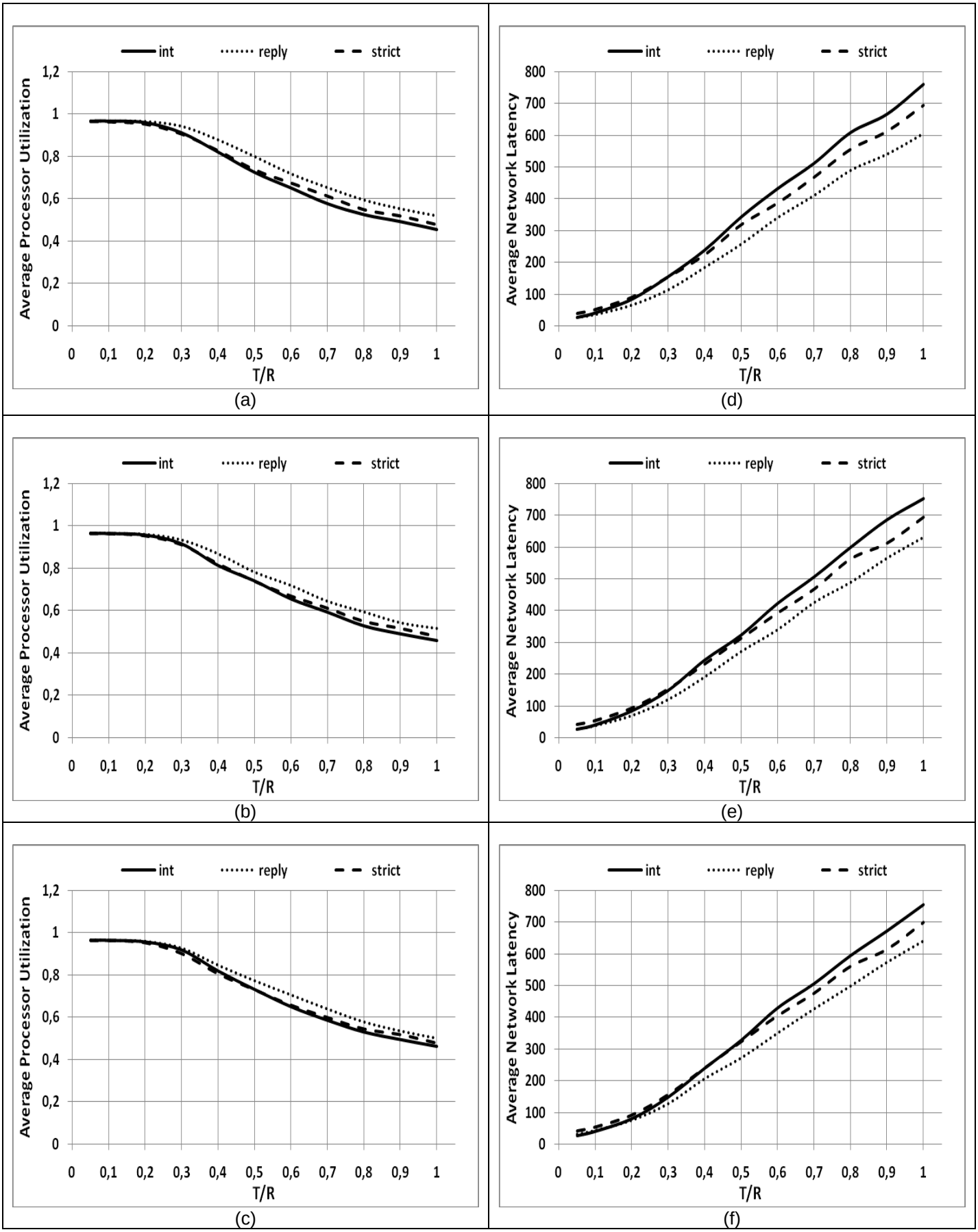



Figure 4.17. Average processor utilization and average network latency of the protocols under perfect shuffle traffic for P(M) = 0.8 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

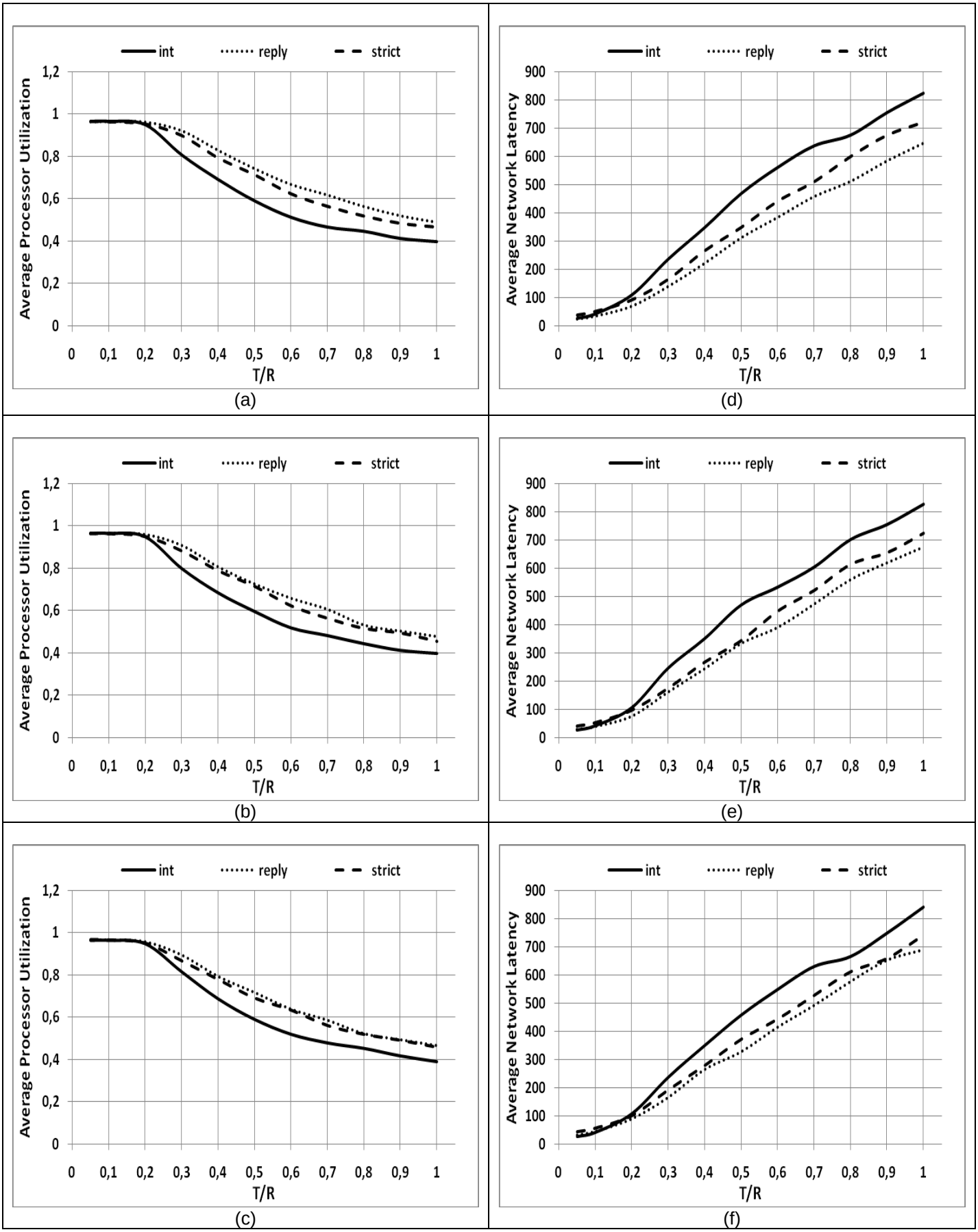



Figure 4.18. Average processor utilization and average network latency of the protocols under hot region traffic for P(M) = 0.8 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

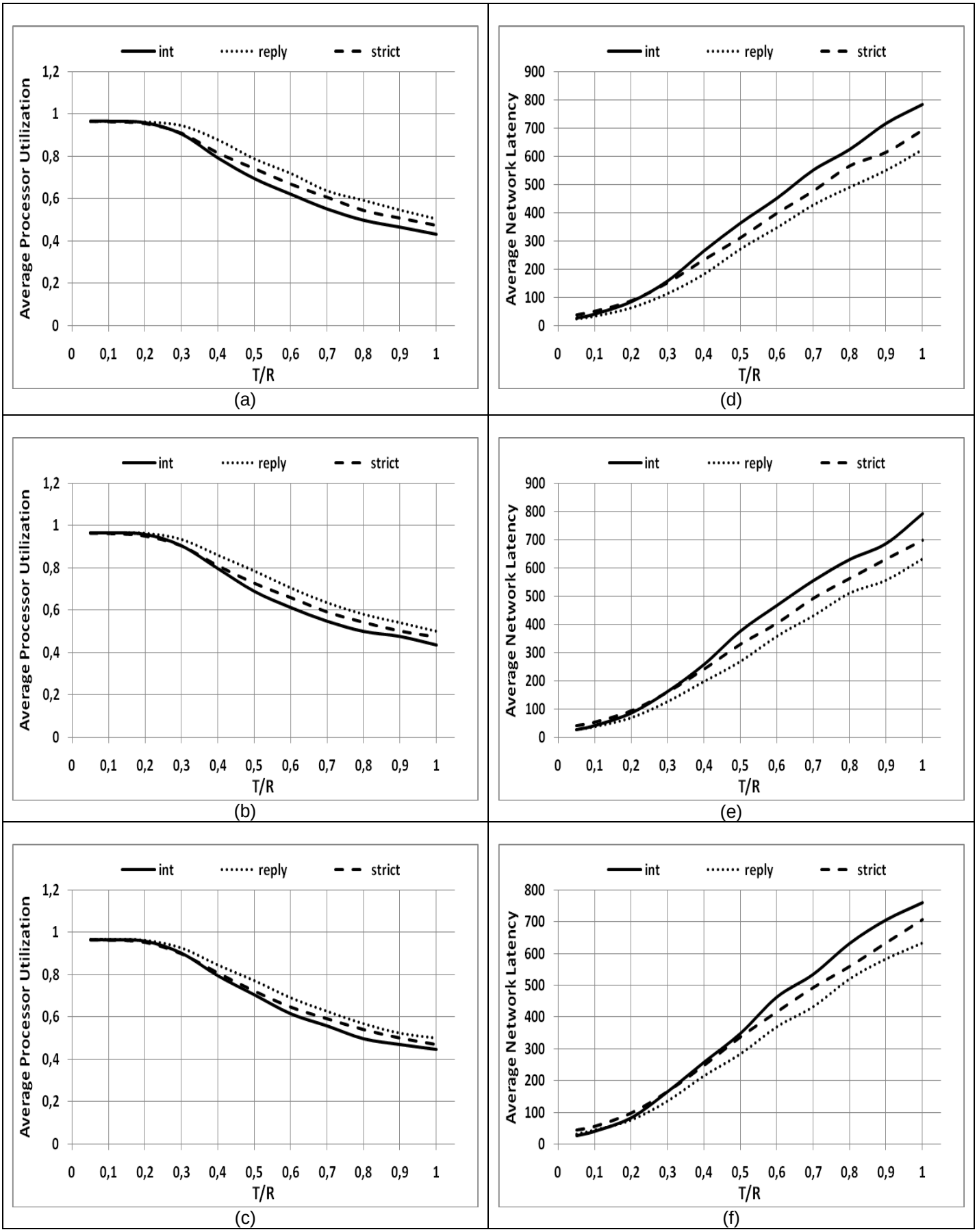



Figure 4.19. Average processor utilization and average network latency of the protocols under uniform traffic for P(M) = 0.8 and (a) – (d) for P(W) = 0.2, (b) – (e) for P(W) = 0.4, (c) – (f) for P(W) = 0.6

Table 4.1. The increase rate of average network latency for each directory protocol under each traffic pattern

| | | | Traffic Pattern | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | BR | | | PS | | | HR | | | UN | | |
| | | P(W) | int | rep | st | int | rep | st | int | rep | st | int | rep | st |
| P(M) | 0.2 | 0.2 | %0 | %0 | %0 | %0 | %0 | %0 | %0 | %0 | %0 | %0 | %0 | %0 |
| | | 0.4 | %17 | %30 | %21 | %14 | %27 | %21 | %12 | %21 | %16 | %15 | %22 | %21 |
| | | 0.6 | %32 | %58 | %39 | %28 | %53 | %41 | %22 | %40 | %33 | %30 | %50 | %40 |
| | 0.4 | 0.2 | %23 | %11 | %16 | %21 | %9 | %15 | %11 | %-2 | %4 | %17 | %5 | %12 |
| | | 0.4 | %33 | %30 | %30 | %30 | %28 | %30 | %21 | %16 | %16 | %27 | %23 | %29 |
| | | 0.6 | %42 | %51 | %44 | %39 | %45 | %42 | %27 | %34 | %31 | %39 | %43 | %43 |
| | 0.6 | 0.2 | %43 | %17 | %32 | %39 | %14 | %29 | %23 | %-4 | %10 | %36 | %10 | %27 |
| | | 0.4 | %49 | %30 | %38 | %43 | %26 | %38 | %28 | %8 | %20 | %41 | %22 | %32 |
| | | 0.6 | %54 | %41 | %47 | %48 | %38 | %45 | %30 | %21 | %25 | %45 | %31 | %42 |
| | 0.8 | 0.2 | %64 | %23 | %46 | %58 | %21 | %43 | %36 | %-5 | %18 | %53 | %15 | %38 |
| | | 0.4 | %61 | %29 | %46 | %57 | %26 | %44 | %36 | %2 | %19 | %54 | %18 | %41 |
| | | 0.6 | %61 | %32 | %46 | %57 | %28 | %46 | %36 | %7 | %21 | %52 | %22 | %43 |

Table 4.2. The decrease rate of average processor utilization for each directory protocol under each traffic pattern

| | | | Traffic Pattern | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | BR | | | PS | | | HR | | | UN | | |
| | | P(W) | int | rep | st | int | rep | st | int | rep | st | int | rep | st |
| P(M) | 0.2 | 0.2 | %0 | %0 | %0 | %0 | %0 | %0 | %0 | %0 | %0 | %0 | %0 | %0 |
| | | 0.4 | %3,3 | %6,3 | %4,4 | %3,1 | %5,8 | %4,5 | %3,4 | %7,4 | %4,7 | %3,2 | %5,8 | %4,8 |
| | | 0.6 | %6,2 | %11,2 | %7,6 | %5,9 | %11,3 | %8,6 | %6,7 | %13,2 | %9 | %6,6 | %11,5 | %8,8 |
| | 0.4 | 0.2 | %4,3 | %2,3 | %3,2 | %4,3 | %1,9 | %3,5 | %2,7 | %-1,3 | %0,6 | %3,7 | %1,4 | %2,9 |
| | | 0.4 | %6,4 | %6 | %5,8 | %6,5 | %5,9 | %6,5 | %5,5 | %4,3 | %3,9 | %6 | %5,9 | %6,2 |
| | | 0.6 | %7,9 | %10,2 | %8,6 | %8 | %9,5 | %8,9 | %7,2 | %9,6 | %7,1 | %8,1 | %10,1 | %9 |
| | 0.6 | 0.2 | %7,9 | %3,4 | %6,2 | %7,9 | %3,2 | %6,4 | %6,2 | %-2,7 | %1,9 | %7,8 | %2,7 | %6,1 |
| | | 0.4 | %9,2 | %6,1 | %7,2 | %8,6 | %5,8 | %8,1 | %7,4 | %0,7 | %4,1 | %8,8 | %5,5 | %7,1 |
| | | 0.6 | %9,8 | %8,3 | %9 | %9,7 | %8,1 | %9,6 | %7,4 | %4,6 | %5,3 | %9,5 | %7,8 | %8,8 |
| | 0.8 | 0.2 | %11,5 | %5,1 | %8,7 | %11,5 | %4,6 | %8,9 | %8,5 | %-3,6 | %3,3 | %11,1 | %3,7 | %8,2 |
| | | 0.4 | %11,2 | %5,8 | %9,1 | %11,1 | %5,2 | %8,9 | %8,5 | %-1,8 | %3,7 | %11,2 | %4,4 | %8,8 |
| | | 0.6 | %11,2 | %6,8 | %9,1 | %11,1 | %6,2 | %9,6 | %8,4 | %-0,5 | %4,1 | %10,7 | %5,4 | %9,2 |