ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS

Gülsade KALE

INVESTIGATION OF THE EFFECT OF MESSAGE COMBINING MECHANISM ON THE 2D SOME-BUS ARCHITECTURE

DEPARTMENT OF COMPUTER ENGINEERING

ADANA, 2012

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

INVESTIGATION OF THE EFFECT OF MESSAGE COMBINING MECHANISM ON THE 2D SOME-BUS ARCHITECTURE

Gülsade KALE

MSc THESIS

DEPARTMENT OF COMPUTER ENGINEERING

We certified that the thesis titled above was reviewed and approved for the award of degree of the Master of Science by the board of jury on 20/11/2012

…….....	……	……..……
Asst. Prof. Dr. M. Fatih AKAY	Assoc. Prof. Dr. Zekeriya TÜFEKÇİ	Assoc. Prof. Dr. Ulus ÇEVİK
SUPERVISOR	MEMBER	MEMBER

This MSc Thesis is written at the Department of Institute of Natural And Applied Sciences of Çukurova University.
Registration Number:

Prof. Dr. Selahattin SERİN
Director
Institute of Natural and Applied Sciences

This thesis as financially supported by C.U Research Resource Foundation **MMF2011YL15**

Not: The usage of the presented specific declerations, tables, figures, and photographs either in this thesis or in any other reference withourt citation is subject to "The law of Arts and Intellectual Products" number of 5846 of Turkish Republic.

ABSTRACT

MSc THESIS

INVESTIGATION OF THE EFFECT OF MESSAGE COMBINING MECHANISM ON THE 2D SOME-BUS ARCHITECTURE

Gülsade KALE

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

Supervisor :Asst. Prof. Dr. M. Fatih AKAY
Year: 2012, Pages: 47
Jury :Asst. Prof. Dr. M. Fatih AKAY
:Assoc. Prof. Dr. Zekeriya TÜFEKÇİ
:Assoc. Prof. Dr. Ulus ÇEVİK

To minimize the effect of hot-spots and latencies in broadcast-based architectures, messages are combined in the output channel queue to form a new message containing the payloads of all original messages. In this thesis, the performance of the message combining method is investigated on a 64-node 2D SOME-Bus architecture which is a low-latency and high-bandwidth broadcast-based fiber-optic interconnection network employing the message passing protocol. The performance of the method is evaluated by OPNET Modeler with various synthetic traffic patterns including hot-region, uniform, perfect-shuffle and bit-reverse. Performance measures such as average processor utilization, average channel waiting time and average network response time have been collected before and after applying the combining method in the output channel queue of a node for varying values of the ratio of the mean channel service time to mean thread run time (T/R) and under different traffic patterns. The results show that average network response times decrease by 8.81% to 20.62%, average channel waiting times decrease by 15.39% to 28.02% and average processor utilization increase by 5.12% to 12.71% after applying the combining method.

Keywords: Parallel processing, multiprocessor architectures, optical interconnection networks, message combining.

ÖZ

YÜKSEK LİSANS TEZİ

MESAJ BİRLEŞTİRME MEKANİZMASININ 2D SOME-BUS MİMARİSİ ÜZERİNDEKİ ETKİSİNİN İNCELENMESİ

Gülsade KALE

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Yrd. Doç. Dr. M.Fatih AKAY
Yıl: 2012, Sayfa: 47
Jüri : Yrd. Doç. Dr. M.Fatih AKAY
: Doç. Dr. Zekeriya TÜFEKÇİ
: Doç. Dr. Ulus ÇEVİK

Yayın tabanlı mimarilere sahip ağlardaki gecikme ve darboğazların etkisini azaltmak için kanal çıkış kuyruğundaki mesajlar diğer tüm orijinal mesajların da verilerini içerecek şekilde birleştirilerek yeni bir mesaj formu oluşturulur. Bu tezde mesaj birleştirme mekanizmasının performansı; 64 düğümlü, düşük gecikmeli, yüksek bant genişliğine sahip yayın tabanlı fiber optik bir ara bağlantı ağı olan 2D SOME-Bus ağı üzerinde mesaj geçişi protokolü çalıştırılarak incelenmiştir. Mekanizmanın performansı; yoğun bölge (hot-region), düzenli (uniform), mükemmel karışım (perfect shuffle) ve bit ters çevirme (bit-reverse) gibi birçok trafik modeli altında OPNET Modeler kullanılarak değerlendirilmiştir. Ortalama işlemci verimi, ortalama kanal bekleme süresi ve ortalama ağ cevap süresi gibi performans ölçüleri farklı trafik modelleri altında ve T/R'nin (ortalama kanal servis süresinin ortalama işlem süresine oranı) farklı değerleri için bir düğümün çıkış kanal kuyruğunda uygulanan birleştirme mekanizmasından önce ve sonra toplanmıştır. Sonuçlar, birleştirme mekanizması uygulandıktan sonra ortalama ağ cevap süresinin %8.81 ve %20.62, ortalama kanal bekleme süresinin %15.39 ve %28.02 arasında azaldığını, ortalama işlemci veriminin ise %5.12 ve %12.71 arasında arttığını göstermektedir.

Anahtar Kelimeler: Paralel işleme, çok işlemcili mimariler, optik ara bağlantı ağları, mesaj birleştirme.

ACKNOWLEDGMENTS

First and foremost, I would like to thank and express my sincere gratitude to my advisor Asst.Prof.Dr. M. Fatih AKAY for his supervision, guidance, encouragements, patience, motivation and useful suggestions for this work. I am grateful to him for all the things that I have learnt under his supervision.

I would like to thank members of the MSc thesis jury, Assoc.Prof.Dr. Ulus ÇEVİK and Assoc.Prof.Dr. Zekeriya TÜFEKÇİ 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: MMF2011YL15).

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 İbrahim KALE, Şenel KALE, my sisters Gökçe KALE and Gonca KALE and my brother Cihangir KALE, for their endless support and encouragements for my life and career.

CONTENTS **PAGE**

LIST OF TABLES **PAGE**

LIST OF FIGURES **PAGE**

LIST OF ABBREVIATONS

SOME-Bus : Simultaneous Optical Multiprocessor Exchange Bus

OPNET : OPNET Technologies, Inc.

1. INTRODUCTION

Parallel computing allows solving problems that usually pose high requirements on the memory and processing power i.e. applications that can not be run using a single CPU. Also, some problems are time critical; they need to be solved in a very limited amount of time. Using uni-processors for those applications can not achieve the desired performance and therefore parallel computers are used to solve that sort of problems. The main intention in using parallel systems is to support high execution speeds. The scope of parallelization of an application comes from the identification of multiple tasks of the same kind, which is a major source of speed up achieved by the parallel computers (Ravela, 2010).

Interconnects specify how the processor nodes in a parallel computers are connected. The information about various topologies are defined in (Duncan,1990; Grama, Gupta, Karypis and Kumar, 2003). Few of the most widely used are ring, bus, hypercube and tree based topologies. The type of the interconnection used is affected in terms of the cost and scalability. Some topologies scale well in terms of area and power while increasing the cost whereas some other topologies come at low cost but not well scalable. The choice of the interconnection used is often a trade-off between these two factors. Also, the inefficient way of interconnecting the processor nodes in a multiprocessor leads to performance drop down by introducing high latency in the communications even though the architecture is scalable. The best interconnection model must scale with the number of the processor nodes, with introducing low latency during communications. Often it is suggested that processor nodes should be interconnected in terms of small ring or bus topologies and interconnecting them using mesh (Asanovic and others, 2006).

Present experience reveals difficulties in the design and use of massively parallel systems because the required computers need interconnection networks with high bisection bandwidth and low latency to connect a large number of nodes. Current massively parallel computers use small-degree networks with large diameters. Wormhole routing achieves lower latency when the network is not heavily

loaded. Performance is often poor or moderate with many modern large-scale applications, mostly due to load imbalance, barrier synchronization, and communication patterns, such as all-to-all communication, which place an excessive load on the interconnection network (Katsinis and Nabet, 2004).

The utmost flexibility in exploiting parallelism is afforded by a topology with diameter equal to one, where each processor can directly communicate with any other processor. Such a network allows the user, or the compiler, to structure the data and operations in the application code to better reflect the parallelism inherent in the applications with the resulting benefit that messages experience smaller queuing delays at the source and the destination. 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. Such an interconnection network can be constructed using optoelectronic devices (and multiple-wavelength data representations), relying on sources, modulators and arrays of detectors, all being coupled to local electronic processors (Katsinis and Nabet, 2004).

Optical interconnection networks (OINs) offer a potentially disruptive technology solution that can provide ultra high throughput, minimal access latencies, and low-power dissipation that remains independent of capacity. By realizing the enormous bandwidth capacities and stringent latency requirements of interconnecting the growing number of compute elements in a scalable fashion, OINs can become a key to relieving what is perhaps the most daunting challenge to performance of future computing systems (Bergman, 2008).

The communication models used in the parallel architecture classifies the type of the multiprocessors (Quammen, 2005). Multiple processors that share the global address space are classified under shared-memory multiprocessors. The multi-processors in these systems communicate with each other through global variables stored in a shared address space. Another complete variant architecture for the one mentioned above is known as distributed-memory multiprocessors where each processor has its own memory module and the data at any time instant is private to the processors. These types of systems are constructed by interconnecting each

component with a high-speed communication network. These architectures rely on the send/receive primitives for communication between multiple processors communicate to each other over the network. Parallel programming models that rely on shared memory based multiprocessors are called shared memory based parallel programming models and parallel programming models that rely on distributed memory architectures are called distributed memory based parallel programming models (Ravela, 2010).

The SOME-Bus (Katsinis, 2001) is a low latency, high bandwidth, fiber-optic interconnection network which directly links arbitrary pairs of processor nodes, and can efficiently interconnect over one hundred nodes. It contains a dedicated channel for the data output of each node, eliminating the need for global arbitration and providing bandwidth that scales directly with the number of nodes in the system. Each of N nodes has an array of receivers, with one receiver dedicated to each node output channel. No node is ever blocked from transmitting by another transmitter or due to contention for shared switching logic. The entire N-receiver array can be integrated on a single chip at a comparatively minor cost resulting in O(N) complexity (Abasikeles and Akay, 2010). The details of the SOME-Bus architecture is given in Section 3.

In multiprocessor architectures, hot-spot contention can occur when several processors concurrently request access to one data structure with resulting severe performance degradation. Hot-spot contention causes additional contention for the buffers and links of the interconnection network (Katsinis, 1999). Hot-spot contention becomes a serious problem affecting the whole system's performance. It is imperative that these systems effectively manage or avoid system bottlenecks such as hot spots (Katsinis, 2004).

Message combining is a method that can be used to avoid hot spots in multiprocessor architectures. To minimize the effect of hot-spots and the latencies on the network in broadcast-based architectures, messages can be combined to form a new message containing the payloads of all original messages. It is possible to combine messages even when they are destined for distinct nodes, to spread the transmission overhead costs over larger combined payloads (Katsinis and Hecht,

2000). The resulting combined message needs less time to be transmitted than the sum of the individual message transmission times, with most of the time savings due to reduced header information. When messages are small, these savings can have a significant effect on performance (Katsinis, 1999).

In this thesis, we investigate the performance of message combining mechanism on the 2D SOME-Bus architecture by using OPNET Modeler. The system includes 64 nodes. Each node in the simulated system contains a processor server and a channel server. Initially, we observe the performance of the base system (without message combining) for varying values of T/R and under different traffic patterns, including hot-region, uniform, perfect-shuffle and bit-reverse. Then, we apply message combining mechanism by combining different number of messages waiting in the channel queues of the system. Performance of the message combining mechanism on the 2D SOME-Bus architecture is assessed by measuring the average processor utilization, average channel waiting time and average network response time.

2. LITERATURE REVIEW

(Hanava, Fujiwara and Amano, 1996) used the message combining mechanism to prevent hot spot contention in a simple serial synchronized multistage interconnection network. The authors state that the influence of the hot spot contention is not catastrophic in the small size interconnection network (16x16) when some parallel programs from SPLASH benchmark run. In this situation, the effect of the message combining is small. However, in large systems (256 processors) the message combining shows better performance.

Hardware techniques have been proposed which attempt to avoid hot-spots by combining memory requests from several processors when directed to the same memory module. The IBM RP3 and Ultracomputer (Almasi and Gottlieb, 1994) systems use this method. Each switch in the interconnection network combines memory request messages directed at the same memory location into a single message and keeps a record of the messages combined. When the reply to a combined message reaches the switch that performed combining, the switch generates multiple replies to all combined messages using the information in its wait buffer. The Ultracomputer and IBM RP3 systems combine two messages into one. In the Ultracomputer, the hardware cost of combining was modest: The pin count increases by 50% and the number of gates doubles to enhance the switch to support combining (Katsinis, 2004).

(Lee, Kruskal and Kuck, 1994) suggests that there is a possibility of many processors requesting a shared variable at virtually the same time. This idea has been formalized in the "hot-spot" traffic model, where a fixed fraction of memory requests is for a single shared variable. Under the model, it has been shown that such concurrent requests to shared variable can create contention serious enough to stall large machines. As an effective method of alleviating this type of contention, "combining," in which several requests for the same variable can be combined into a single request, has been suggested. The effectiveness of combining has been studied and it turns out that pair wise combining is slightly too restrictive to handle hot-spots if the machine size becomes very large.

(Dickey, Gottlieb and Liu, 1994) investigates the effect of message combining in a variety of switch architectures for use in multistage interconnection networks. Simulation results are provided for implementation alternatives using switches of varying capabilities in order to compare the effectiveness of message combining. It is shown that the two-and-a-half-way combining switch provides performance equivalent to that of other more expensive designs for systems with up to 1024 processors.

(Sauravlas and Roumeliotis, 2010) presents a message combining approach for scheduling runtime array redistribution of one-dimensional arrays. The important contribution of the proposed scheme is that it eliminates the need for local data reorganization, the blocks destined for each processor are combined in a series of messages exchanged between neighboring nodes, so that the receiving processors do not need to reorganize the incoming data blocks before storing them to memory locations.

Although combining requests at a switch in shared memory architectures has received a lot of attention, the similar problem in cache based distributed shared memory architectures has received much less attention. (Shafer and Ghose, 1994) uses message combining for optimizing inter-processor communication in programs for distributed memory multiprocessors. Their basic approach is to combine messages with the explicit goal of reducing the overall execution time, taking into account direct and indirect dependencies among the concurrent units. They first establish that combining messages between a pair of isolated processors is not necessarily useful in reducing the overall execution time of the program because of complex inter-processor dependencies. The conditions under which message combining is profitable are then established. They then search for such conditions along chains of dependencies that exist across several processors and combine messages that satisfy these conditions.

(Katsinis, 2000) examines the effect of message combining on the performance of a multiprocessor system based on a broadcast interconnection network using simulation and analytical models under the message-passing paradigm and the presence of hot-spots, and they compare it to similar performance

measures on multiprocessor systems based on the crossbar and the torus. The authors conclude that the effectiveness of combining 4 or 8 messages is small, however combining two messages does have a strong effect, resulting in 10% increase of processor utilization as network traffic increases.

(Katsinis and Hecht, 2004) investigates the effect of message combining on the fault-tolerant DSM on the SOME-Bus multiprocessor architecture. Simulation results show that in the SOME-Bus architecture under the DSM paradigm, messages tend to wait at the node output network interface. Consequently, to minimize the effect of increased network traffic, messages can be combined at the node output queue to form a new message containing the payloads of all original messages. They use simulation to examine the effect of such message combining on the performance of SOME-Bus, in the presence of additional traffic due to fault tolerance, and they compare it to similar performance measures of a reduced SOME-Bus network where two nodes share one channel. The authors state that under the DSM paradigm, where messages are relatively small, combining of messages has a strong effect on performance.

(Katsinis, 2004) examines the effect of message combining on the performance of SOME-Bus, in the presence of hotspots, using simulation and analytical models, under the message-passing and distributed shared-memory paradigms, and they compare it to similar performance measures on multiprocessor systems based on the crossbar and the torus. The authors state that message combining has a significant effect even its simplest form of two-way combining, with higher levels of combining almost canceling the effect of the hot spot. Simulation results show that, this is specially true under the DSM paradigm where messages are relatively small. Also, even in larger systems and under a severe hot-spot, message combining is beneficial and succeeds in reducing average latency significantly.

3. OVERVIEW OF THE SOME-BUS ARCHITECTURE

3. 1. 1D SOME-BUS

The SOME-Bus (Katsinis, 2001) incorporates optoelectronic devices into a very-high-performance processing architecture. It is a low-latency, high-bandwidth, fiber-optic interconnection network. One of its key features is that each of N nodes has a dedicated broadcast channel operating at 20–30 GBytes/sec, realized by a group of wavelengths in a specific fiber, and an input channel interface based on an array of N receivers which simultaneously monitors all N channels, resulting in an effectively fully connected network. The receiver array (Figure 3.1) does not need to perform any routing, and consequently its hardware complexity (including detector, logic, and packet memory storage) is small. This organization eliminates the need for global arbitration and provides bandwidth that scales directly with the number of nodes in the system. No node is ever blocked from transmitting by another transmitter or due to contention for shared switching logic (Katsinis and Nabet, 2004).

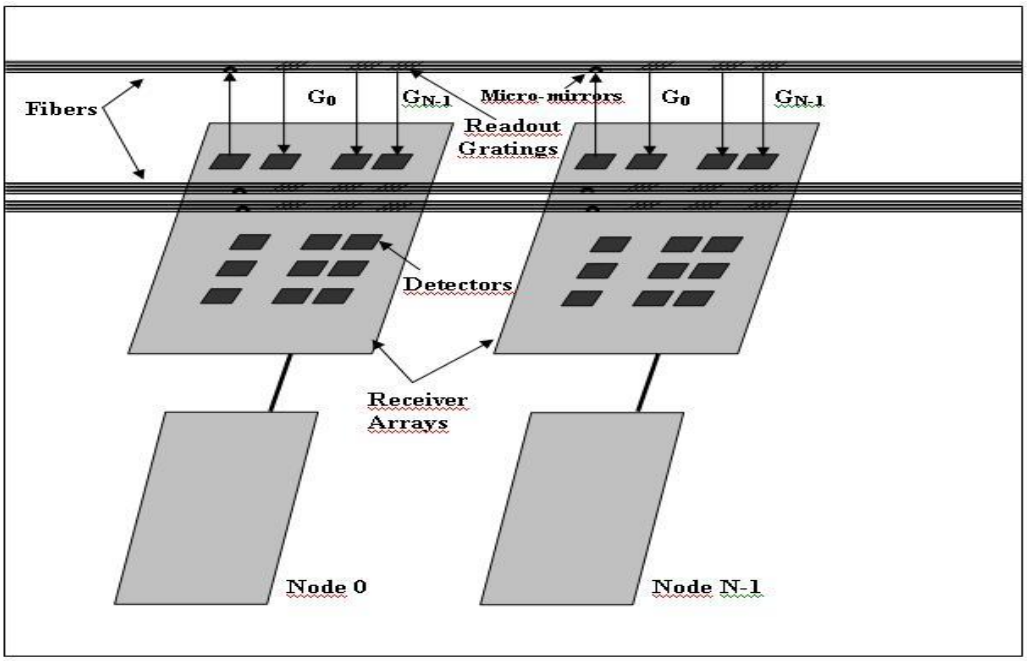

Figure 3.1.Parallel receiver array (Katsinis, 2004)

Messages exchanged between nodes contain a header field with information on the message id, source address, destination address and CRC. Once the logic level signal is restored from the optical data, it is directed to the input channel interface which consists of two parts: the optical interface, shown in Figure 3.2, which includes physical signaling, address filtering, barrier processing, length monitoring and type decoding, and the processor interface, shown in Figure 3.3, which includes a routing network and a queuing system. One queue is associated with each input channel, allowing messages from any number of processors to arrive and be buffered simultaneously, until the local processor is ready to remove them. Synchronization messages are collected and processed at the receiver. Arbitration may be required only locally in a receiver array when multiple input queues contain messages (Katsinis and Nabet, 2004).

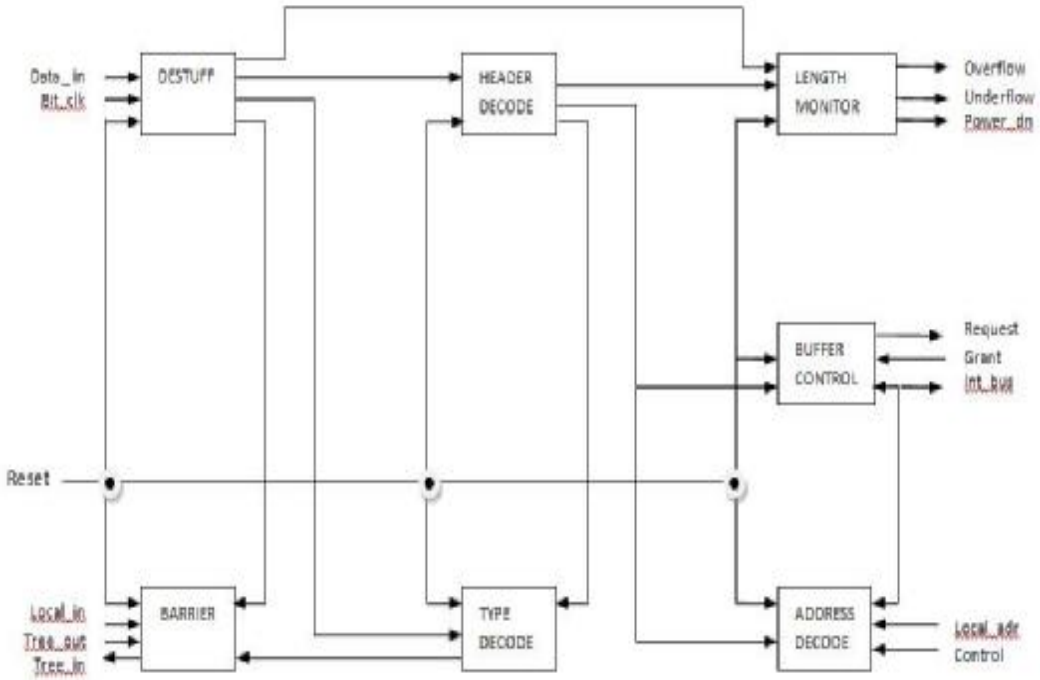

Figure 3.2.Optical interface (Katsinis, 2004)

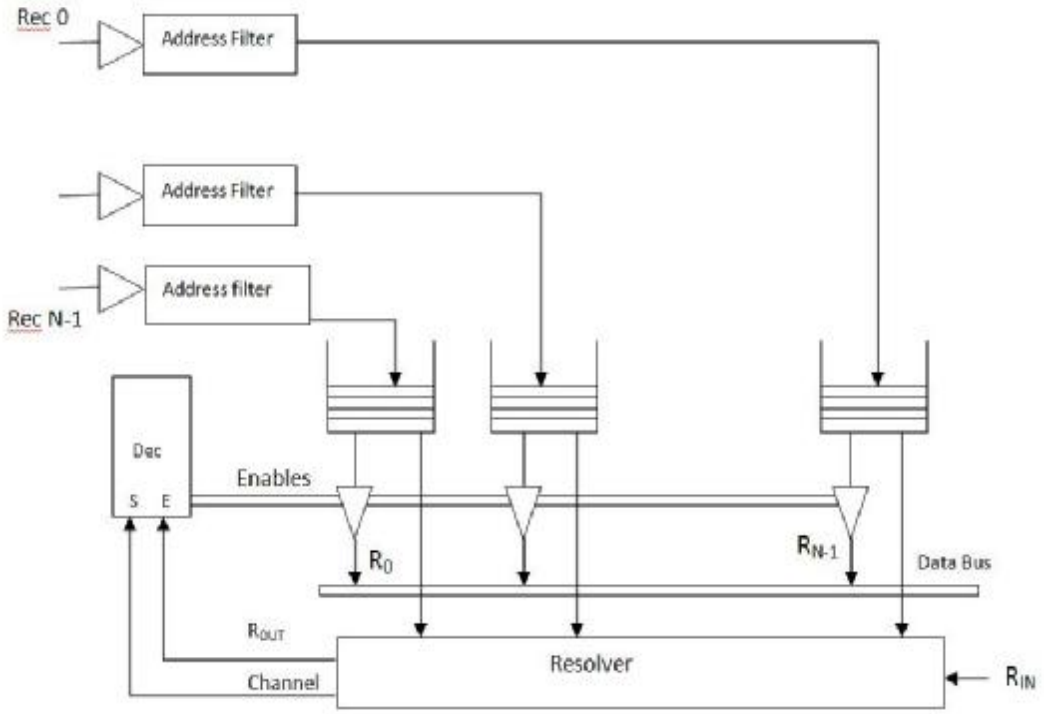

Figure 3.3.Processor interface (Katsinis, 2004)

Each detector generates a bit stream which is examined to detect the start of the packet and the packet header. The header decode circuitry examines the header field, which includes information on the message type, destination address and length, to determine whether the message is a synchronization message or a data message. If the message is a synchronization message, it is handled by the barrier circuitry, otherwise, it is a data message and the destination address is compared to the set of valid addresses contained in the address decode circuitry. In addition to recognizing its own individual address, a processor can recognize multicast group addresses as well as broadcast addresses. Once a valid address has been identified, the data is buffered and placed in the proper queue. If the address does not match, the message is ignored (Katsinis and Nabet, 2004).

3.2. 2D SOME-BUS

The 2D SOME-Bus consists of N horizontal and vertical 1D SOME-Bus networks. Each node is connected to one horizontal and one vertical SOME-Bus. At

each node, an electro-optical converter consisting of a dual receiver and transmitter pair allows messages broadcast on one bus to be forwarded and broadcast on the bus in the other dimension. Figure 3.4 shows the organization of a smaller-scale (4 by 4) network. Each bus has a set of wavelengths which form the communication channels of the nodes attached to that bus. All the buses use the same wavelengths. For simplicity, Figure 3.4 shows one wavelength per channel (Katsinis and Nabet, 2004).

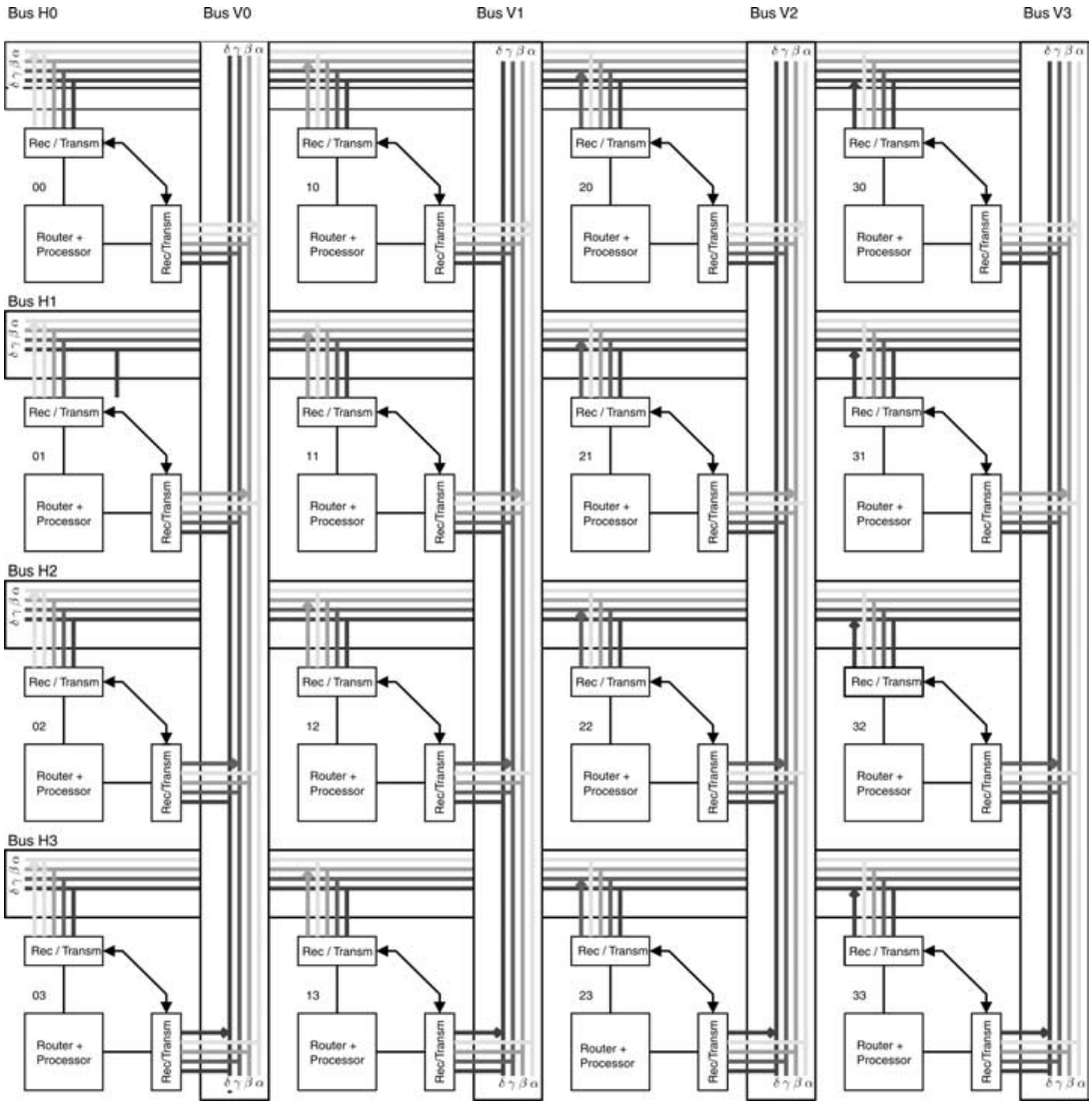

Figure 3.4.2D SOME-Bus (Katsinis, 2004)

If a node N_{ij} (connected to vertical bus V_i and horizontal bus H_j) sends a message to node N_{mn}, and if either i = m or j = n, then only one bus (vertical or

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

If no rebroadcast from one dimension to the other is necessary, then the system operation is equivalent to the operation of the 1D SOME-Bus. When a message must be rebroadcast from one dimension to the other, a message is enqueued locally at the intermediate node. At some later time, the message is broadcast on the other dimension in the same way as locally generated messages are transmitted.

Figure 3.5 shows the block diagram of a node. There are two sets of receivers and transmitters, one associated with a vertical and a horizontal bus. A router connects the receiver/transmitter pairs and the local processor. The router allows the processor to access the queue in each transmitter, and one queue in each receiver. The router also allows messages to be forwarded from one receiver to the opposite transmitter for rebroadcast on the other dimension. The router moves messages in electronic form (16- or 32-bit words) and does not need to operate at photonic speeds (Katsinis and Nabet, 2004).

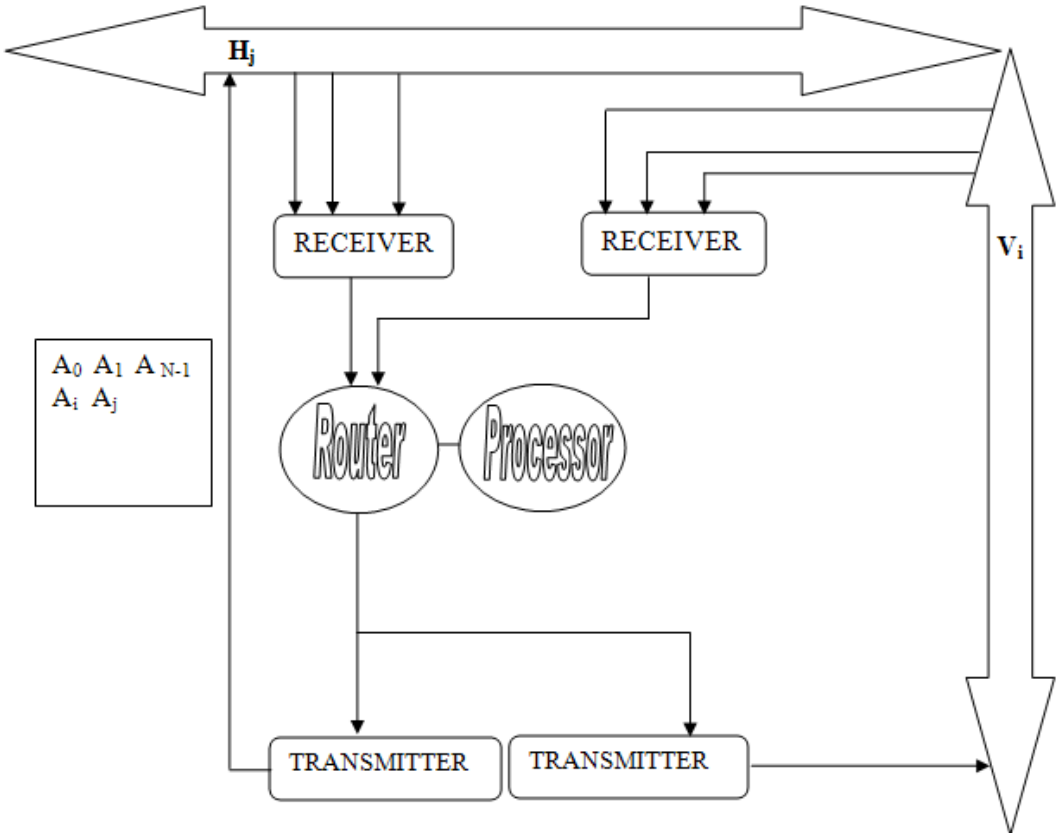

Figure 3.5.Node block diagram

4. MATERIAL AND METHOD

This section presents the simulation framework used for investigating the performance of the message combining mechanism on the 2D SOME-Bus architecture.

The 2D SOME-Bus with N^2 nodes, contains N 1D SOME-Bus networks and in addition there are N additional 1D SOME-Bus networks interconnecting the nodes in the other dimension. In the current version of the architecture, each node has two distinct interfaces to two 1D SOME-Bus networks.

When the 2D SOME-Bus is used under message passing, individual message communication is still point-to-point: a message originates at one node and is received by a single node. Due to multiple broadcast capabilities of the network, many or even all nodes may transmit a message at the same time to different destination nodes or even the same destination node. In the 2D SOME-Bus, if the destination of a message is on not on the same 1D SOME-Bus as the source, then the message is sent to an intermediate node for retransmission. At that intermediate node the message may be queued.

We use OPNET Modeler, which is an environment for network modeling and simulation, and 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 (Shin and Pinkston, 2003).

It is assumed that the processor at each node extracts a data message from an input queue, processes it for a period of time and when that period expires, it generates an output data message. A processor becomes idle only when all its input queues are empty. Messages are distinguished by their type; data or acknowledgment.

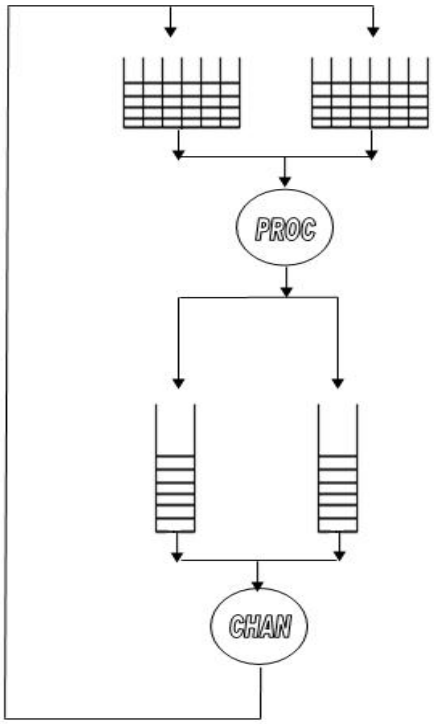

Figure 4.1 Node model

The underlying process model that controls queue models’ behaviour is OPNET’s built-in acb_fifo model, which is given in Figure 4.2 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 strating 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 “svc_comp” state. While in the “svc_comp” state, if the queue is not empty, the process moves into the “svc_start” state.

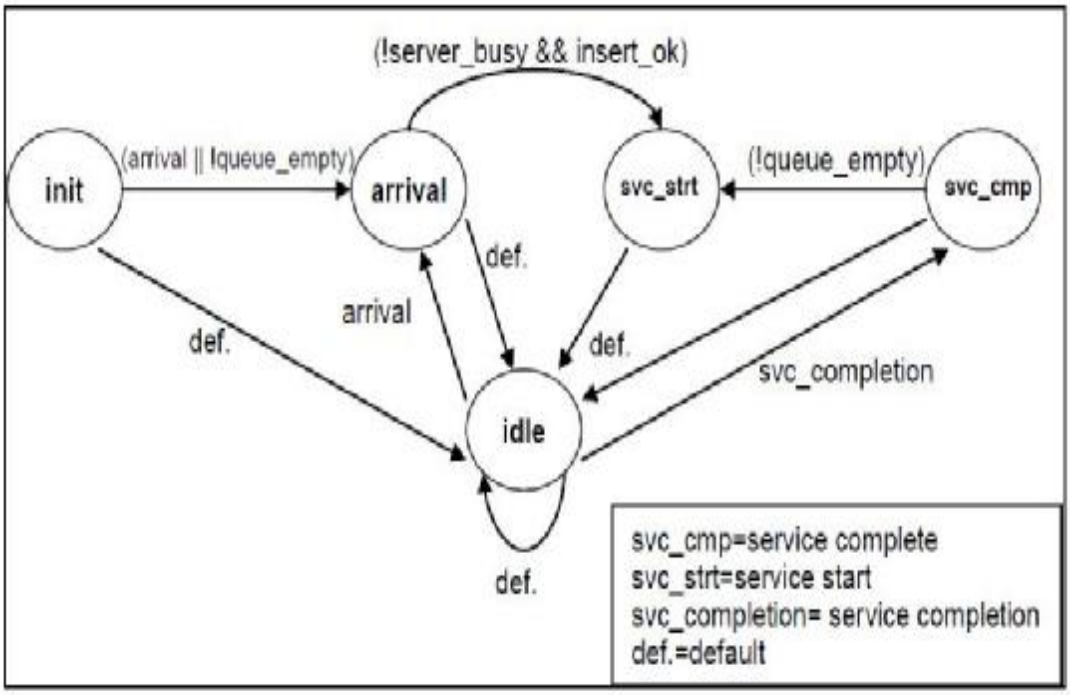

Figure 4.2 Process model of queue modules

Messages exchanged between nodes contain a header field with information on the message type (data or acknowledgment), length, destination address and CRC. 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.

The system includes 64 nodes. Each node in the simulated system contains a processor server and two channel servers. Initially, we observe the performance of the base system (without message combining) for varying values of T/R and under different traffic patterns. Then, we apply message combining mechanism by combining different number of messages waiting in the channel queues of the system. Performance of the message combining mechanism on the 2D SOME-Bus architecture is assessed by measuring the average processor utilization, average channel waiting time and average network response time.

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.) [9].

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 non-independent 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 application [20]. 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 times. The packets keep walking around the network until the simulation ends.

Regarding spatial distributions, we use a collection of well-known permutations: uniform, bit-reverse, perfect shuffle and hot region. Uniform traffic pattern can be represented by a traffic matrix, where each matrix element $\lambda_{s,d}$ gives the fraction of traffic sent from node *s* destined to node *d*. In the uniform traffic, the destination node is selected using uniform distribution with mean in range from 1 to N. Bit permutations such as bit-reverse and perfect shuffle are those in which each bit d_i of the b-bit destination address is a function of the one bit of the source address (Dally and Towels, 2004). In the hot region pattern, the destinations of the 25% of the packets are chosen randomly within a small hot-region consisting of 12.5% of the

nodes (Blumrich, 2003) Table 4.2 lists the destination node selection for these traffic patterns.

Table 4.1 Synthetic traffic patterns

Name	**Pattern**
Uniform (UN)	$\lambda_{s,d} = 1 / N$
Bit-Reverse (BR)	$d_i = b_i + 1$
Perfect Shuffle (PS)	$d_i = s_{i-1} \bmod b$
Hot-Region (HR)	25% of the packets are sent to 12.5% of the node group

The packet format used in the simulation is shown in Table 4.1. Each packet includes the source address, destination address, message ID and CRC in its header. Acknowledgment packets contain only header information whereas packets that carry data contain header information and data. In general, there are 3 types of packets, which differ in the number of bytes used for destination addresses.

Table 4.2.Packet format

	Packet 1	Packet 2	Packet 3
Source Address	2 bytes	2 bytes	2 bytes
Destination Address	2 bytes	4 bytes	6 bytes
Message ID	2 bytes	2 bytes	2 bytes
CRC	2 bytes	2 bytes	2 bytes
Total Header	8 bytes	10 bytes	12 bytes
Total (Header + Payload)	8+10=40 bytes	10+32=42 bytes	12+32=44 bytes

The other major parameters of the simulation are the number of nodes (N), the distribution of processing times (with mean R), the distribution of message transfer times (with mean T), and the number of messages (K) in the system.

Simulation has been run for two traffic models, client-server and asynchronous. In client-server model, the acknowledgment is sent from the receiver node back to the sender node whereas in asynchronous model, there is no acknowledgment message.

5. RESULTS AND DISCUSSION

Fig.5.1 through Fig.5.8 show average channel waiting times for different traffic patterns under client-server and asynchronous models. Fig.5.9 through Fig.5.16 show average network response times for different traffic patterns under client-server and asynchronous models. Fig.5.17 through Fig.5.24 show average processor utilizations for different traffic patterns under client-server and asynchronous models. Fig.5.25 through Fig.5.28 show average rate of decrements for channel waiting times under client-server and asynchronous models. Fig.5.29 through Fig.5.32 show average rate of decrements for network response times under client-server and asynchronous models. Fig.5.33 through Fig.5.36 show average rate of increments for processor utilizations under client-server and asynchronous models.

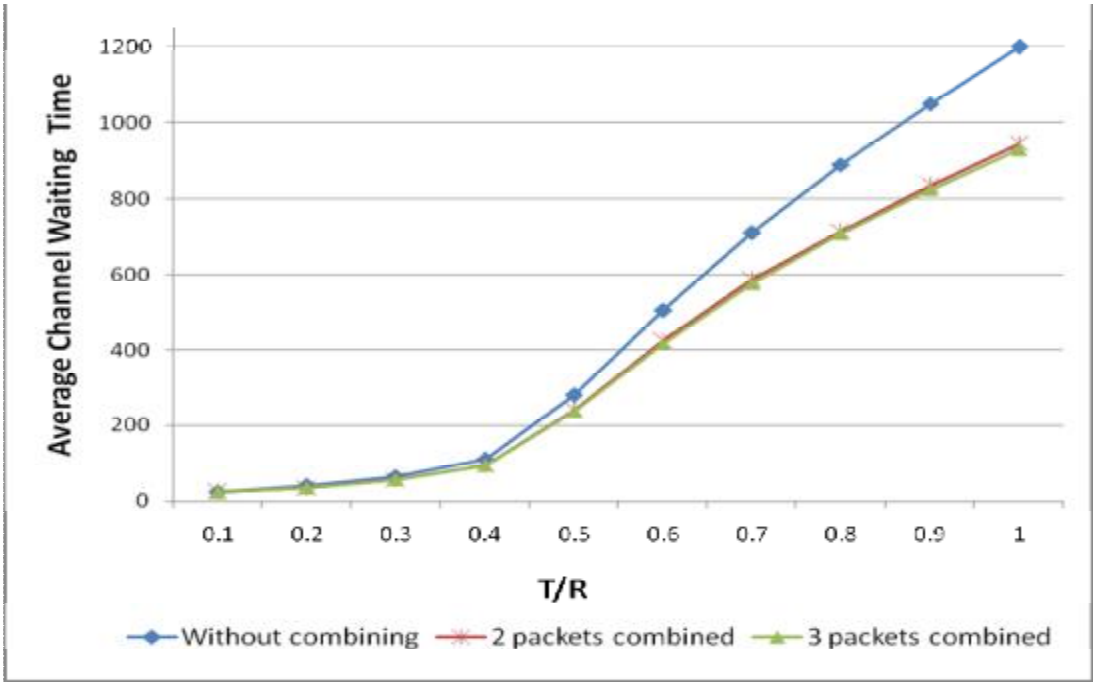

Figure 5.1. Average channel waiting time for uniform pattern under client-server traffic model

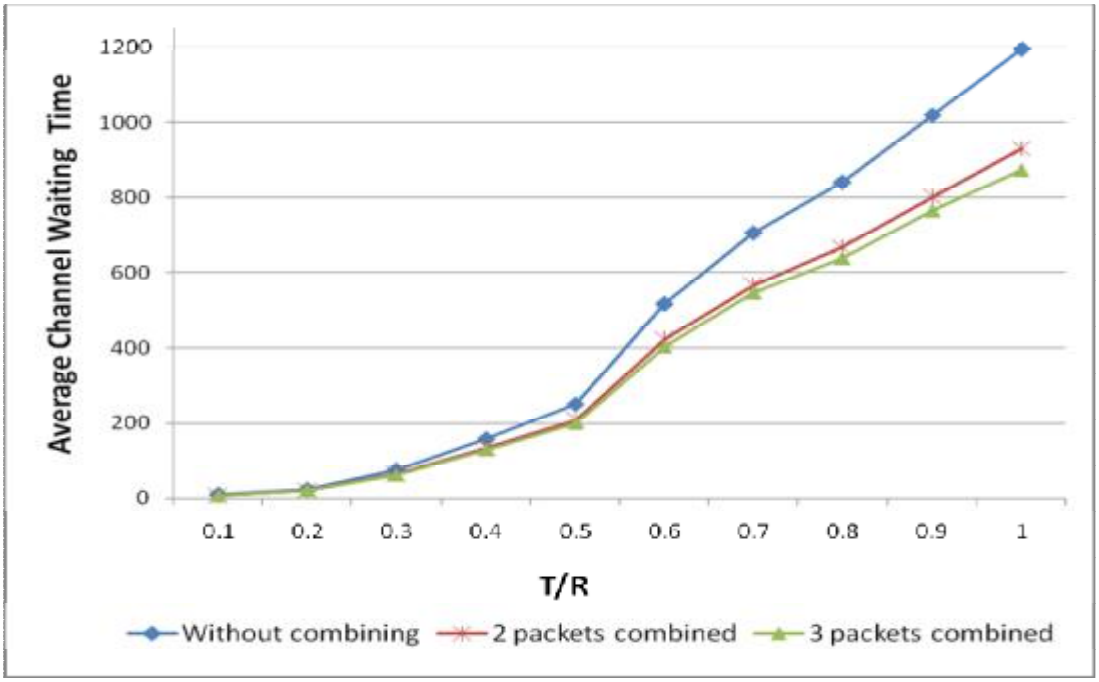

Figure 5.2. Average channel waiting time for bit-reverse pattern under client-server traffic model

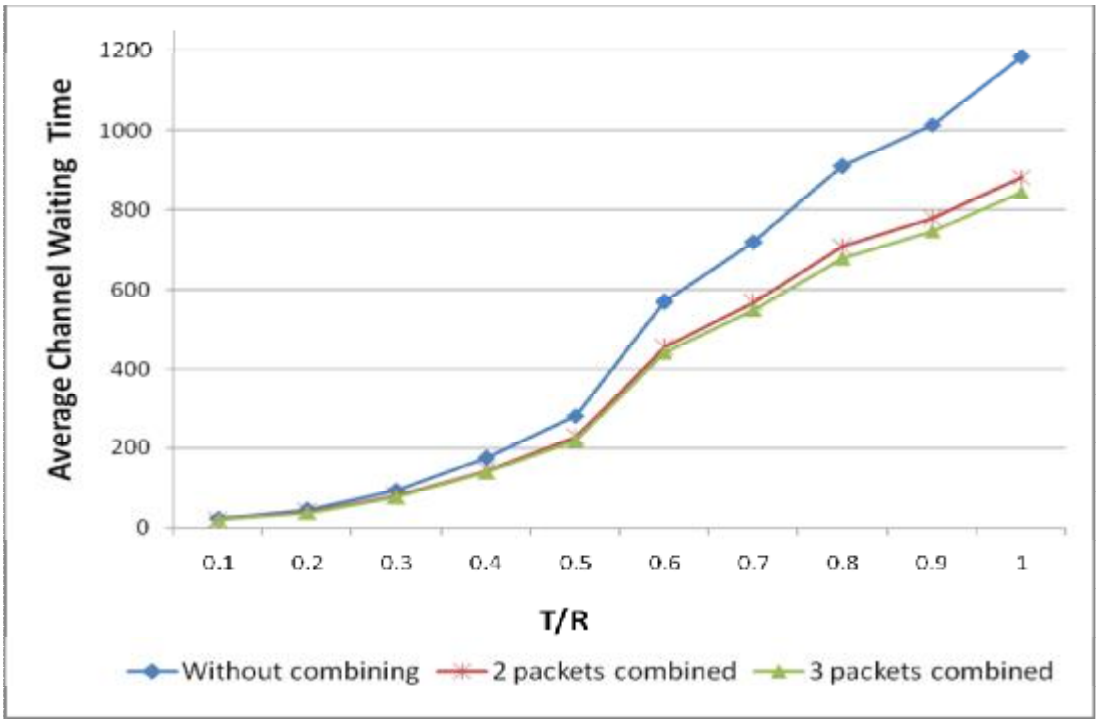

Figure 5.3. Average channel waiting time for perfect shuffle pattern under client-server traffic model

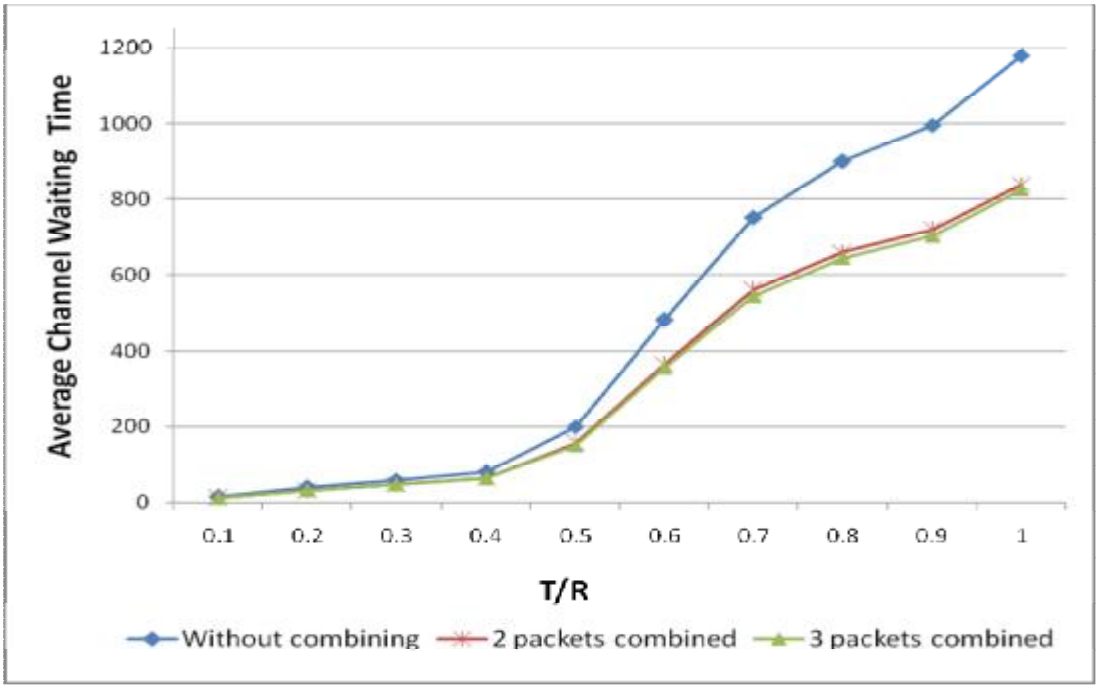

Figure 5.4. Average channel waiting time for hot-region pattern under client-server traffic model

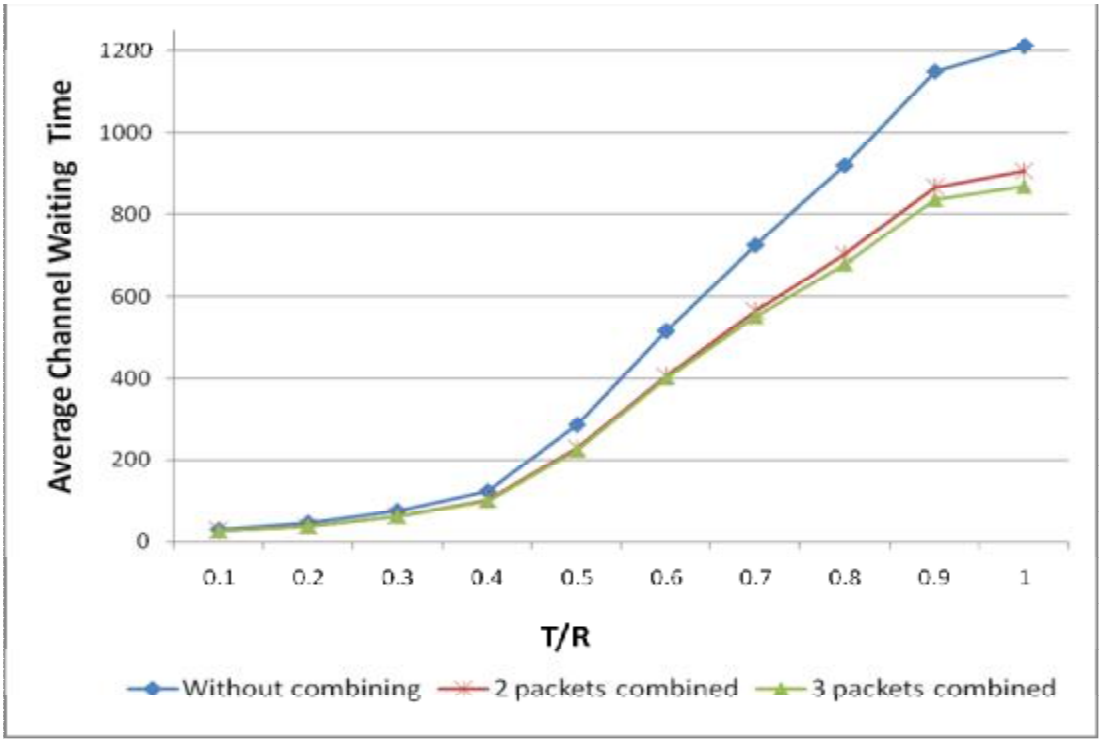

Figure 5.5. Average channel waiting time for uniform pattern under asynchronous traffic model

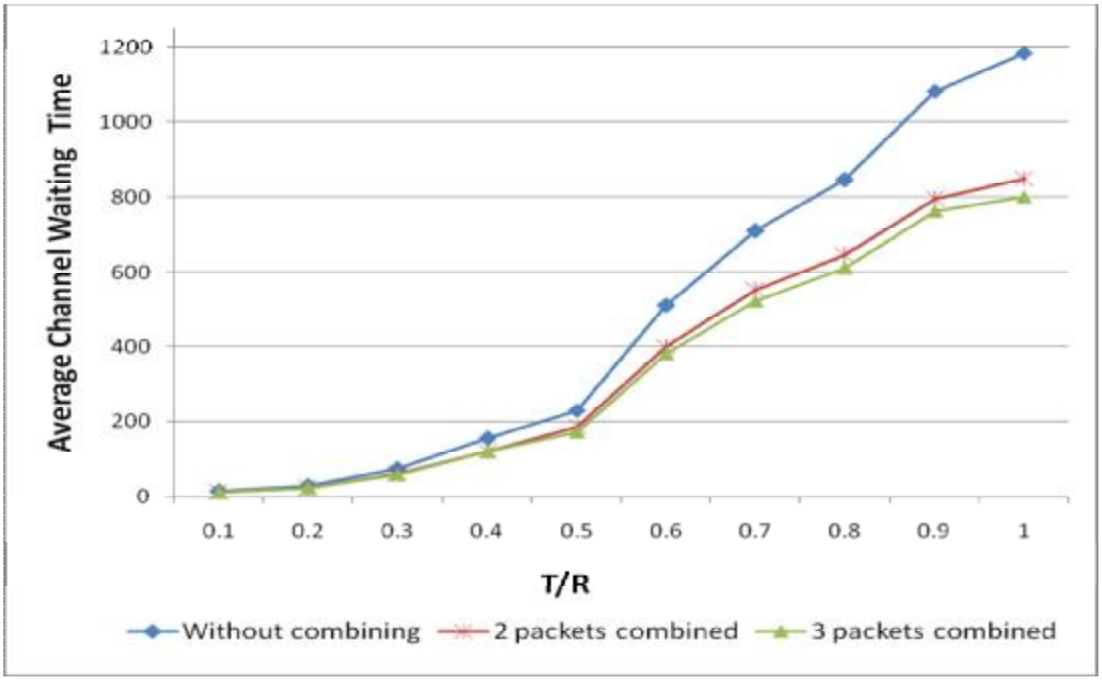

Figure 5.6. Average channel waiting time for bit-reverse pattern under asynchronous traffic model

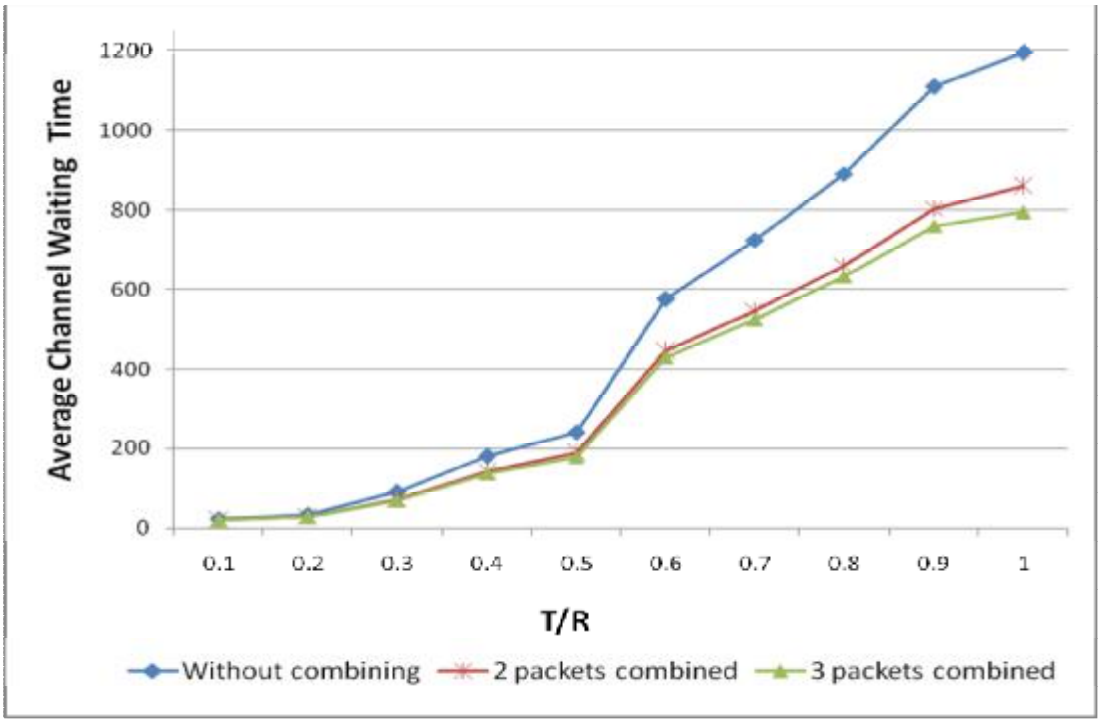

Figure 5.7. Average channel waiting time for perfect shuffle pattern under asynchronous traffic model

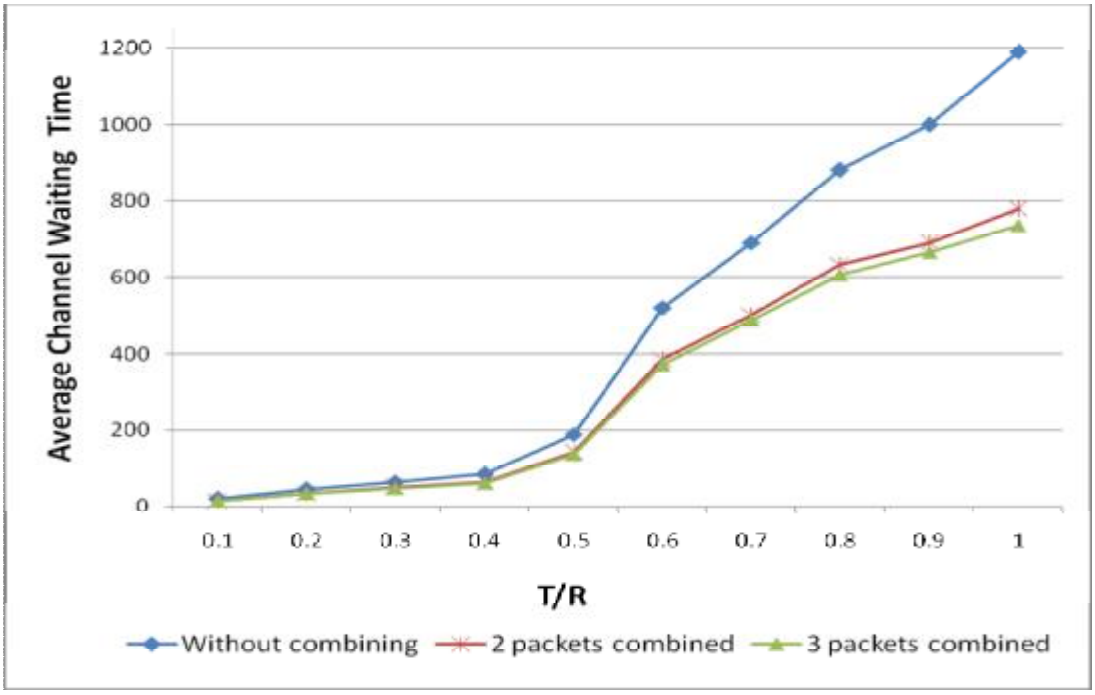

Figure 5.8. Average channel waiting time for hot-region pattern under asynchronous traffic model

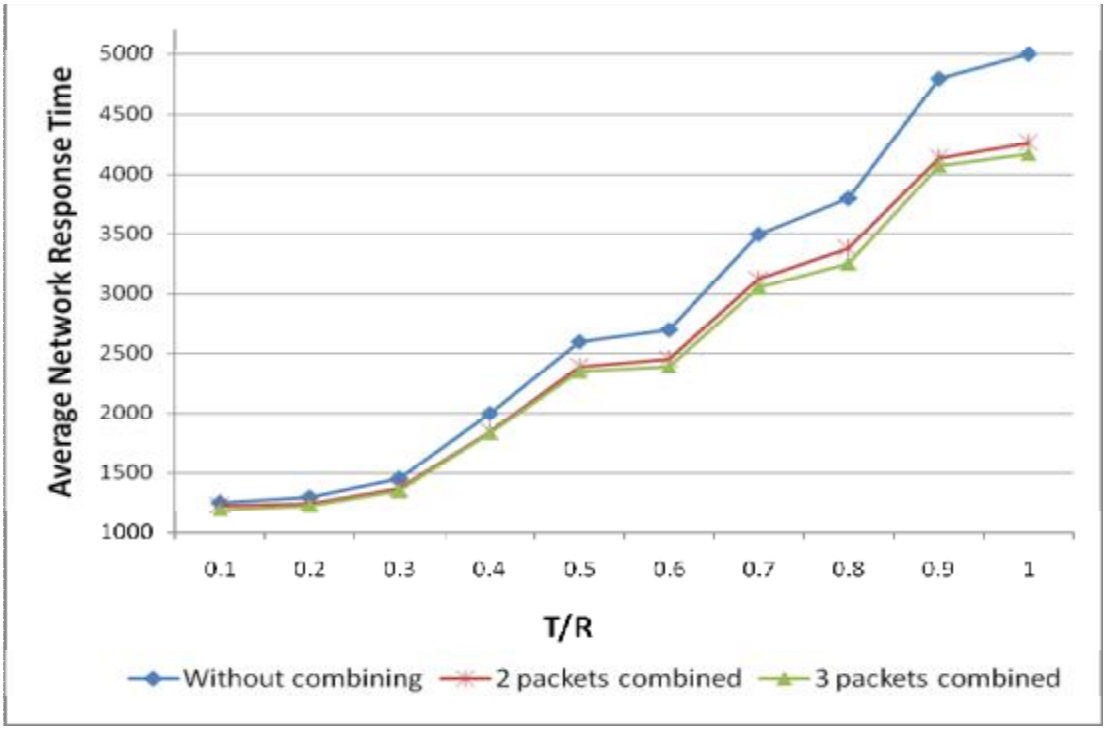

Figure 5.9. Average network response time for uniform pattern under client-server traffic model

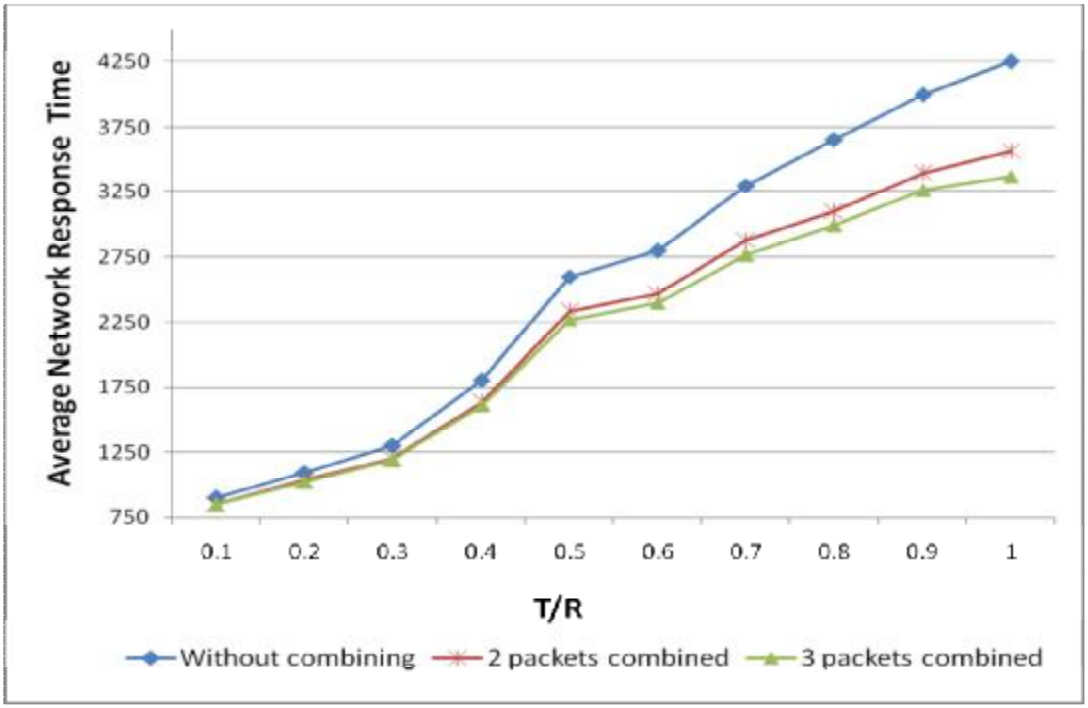

Figure 5.10. Average network response time for bit-reverse pattern under client-server traffic model

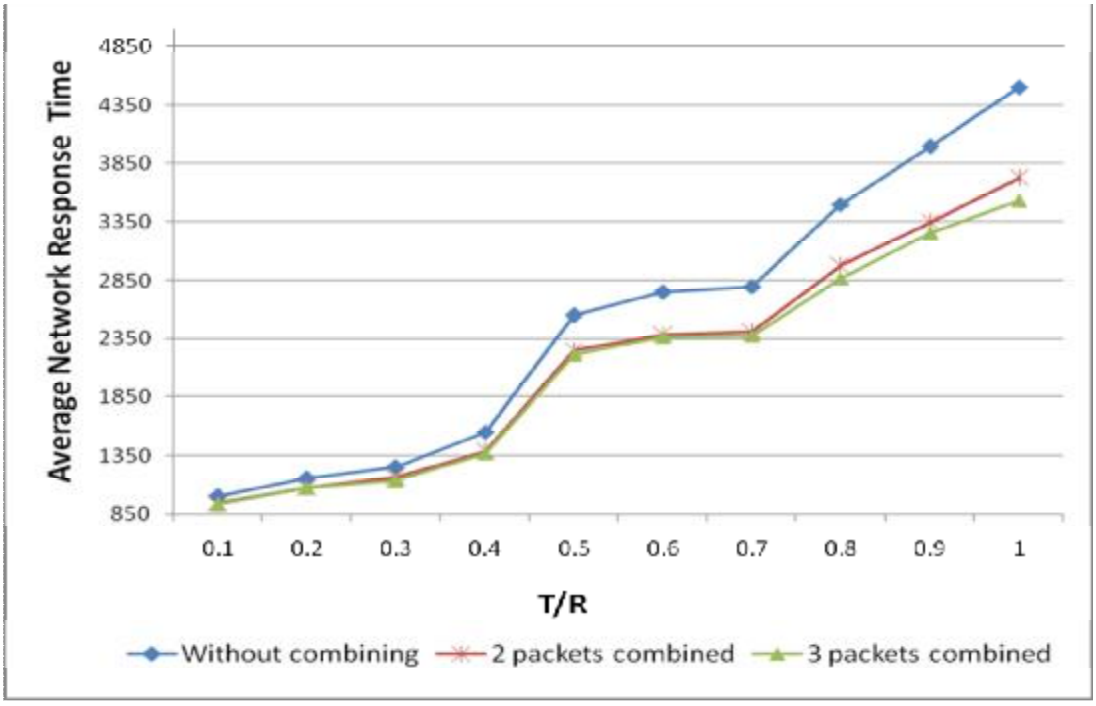

Figure 5.11. Average network response time for perfect shuffle pattern under client-server traffic model

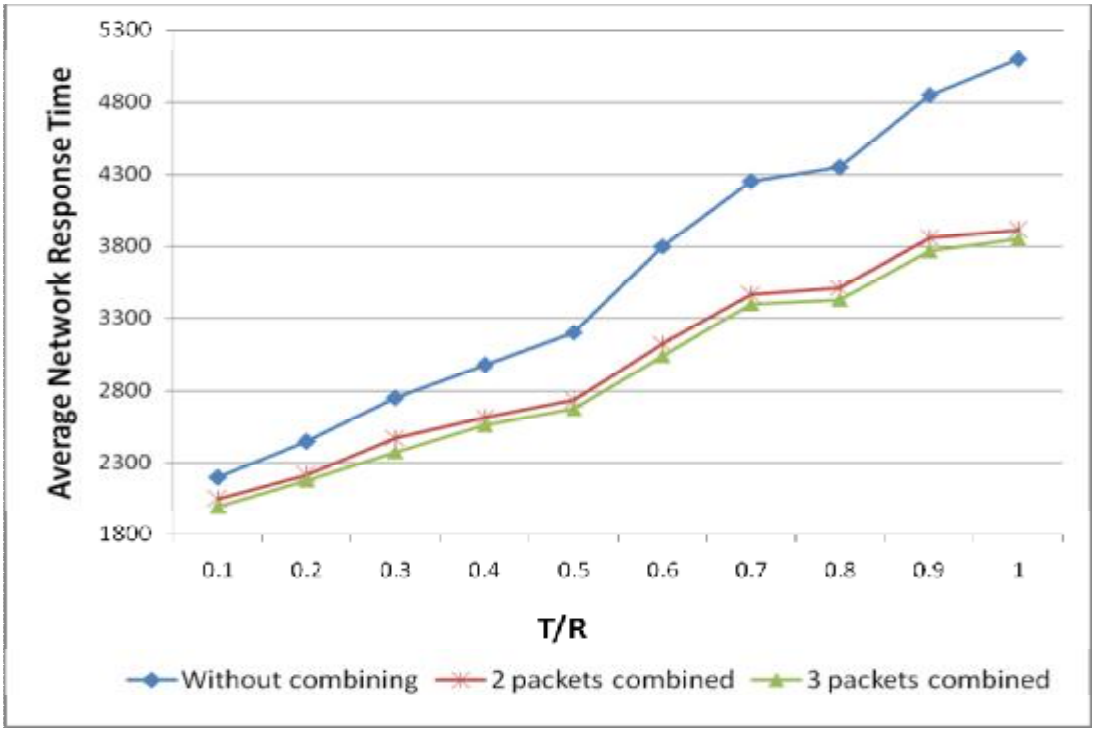

Figure 5.12. Average network response time for hot-region pattern under client-server traffic model

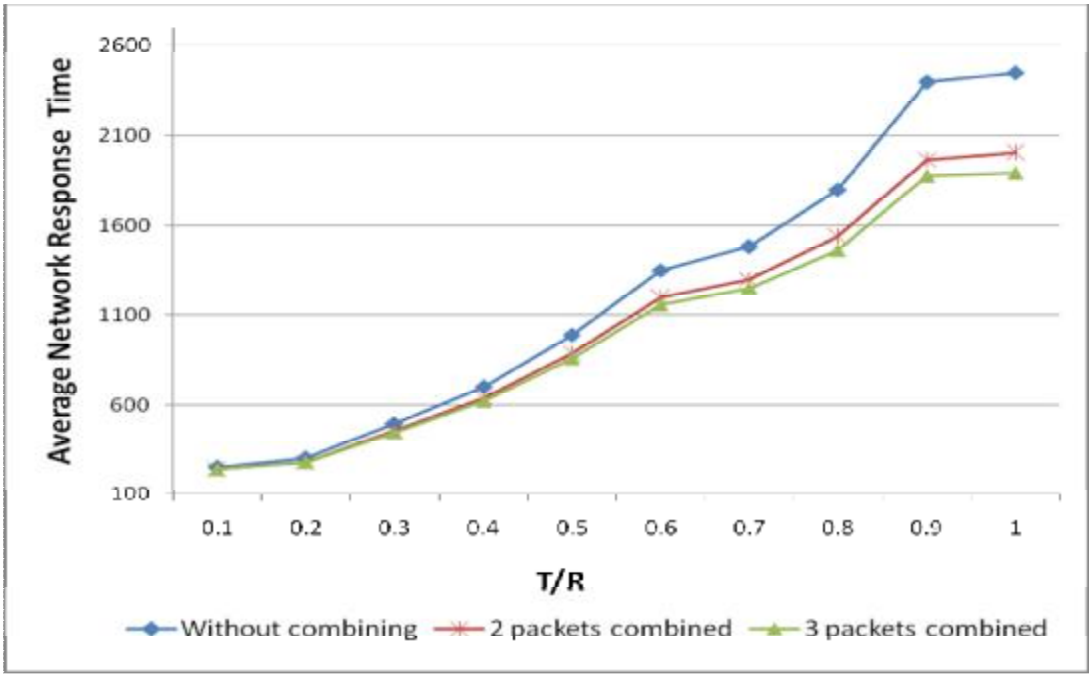

Figure 5.13. Average network response time for uniform pattern under asynchronous traffic model

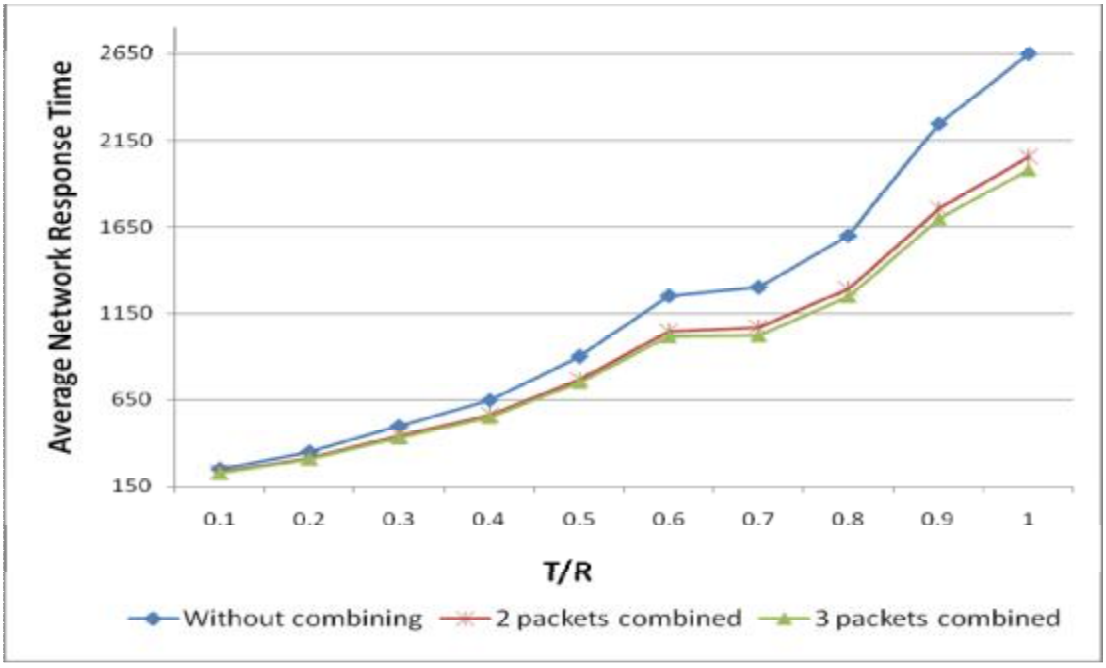

Figure 5.14. Average network response time for bit-reverse pattern under asynchronous traffic model

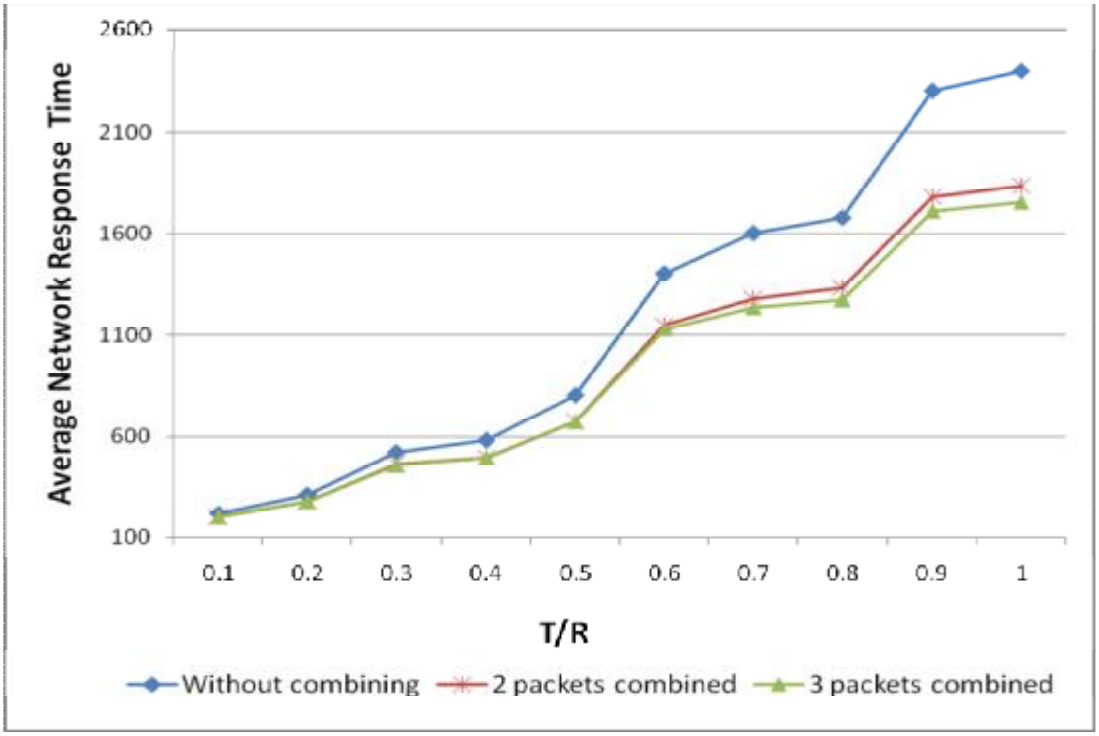

Figure 5.15. Average network response time for perfect shuffle pattern under asynchronous traffic model

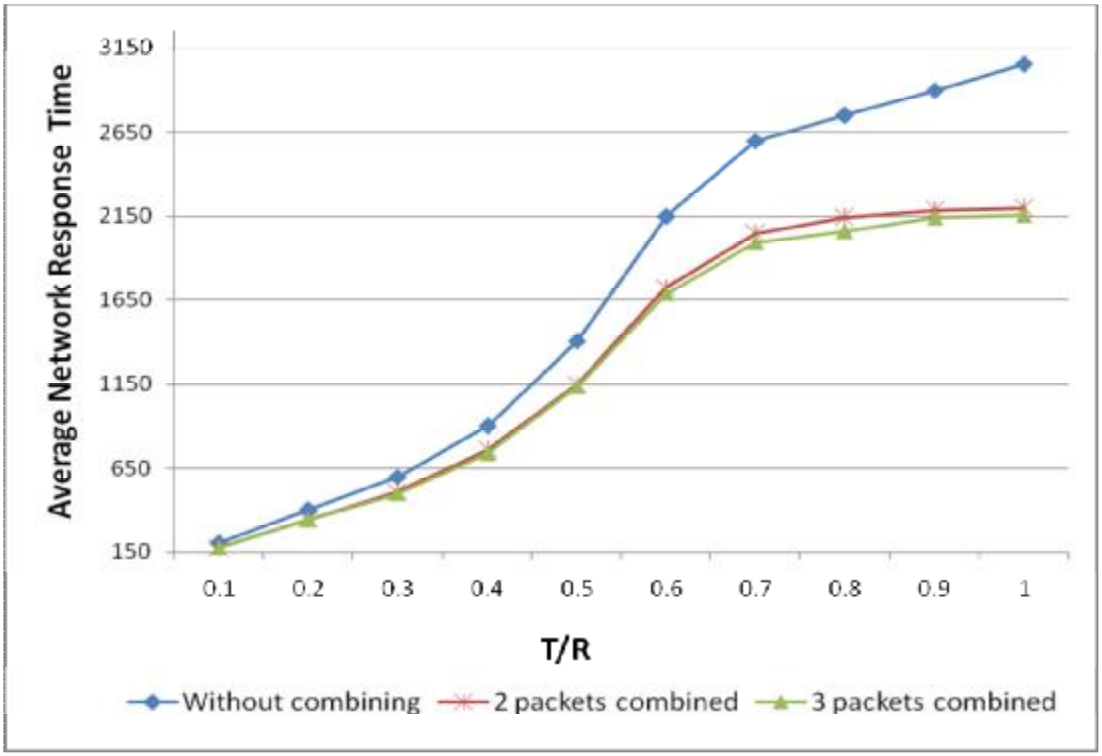

Figure 5.16. Average network response time for hot-region pattern under asynchronous traffic model

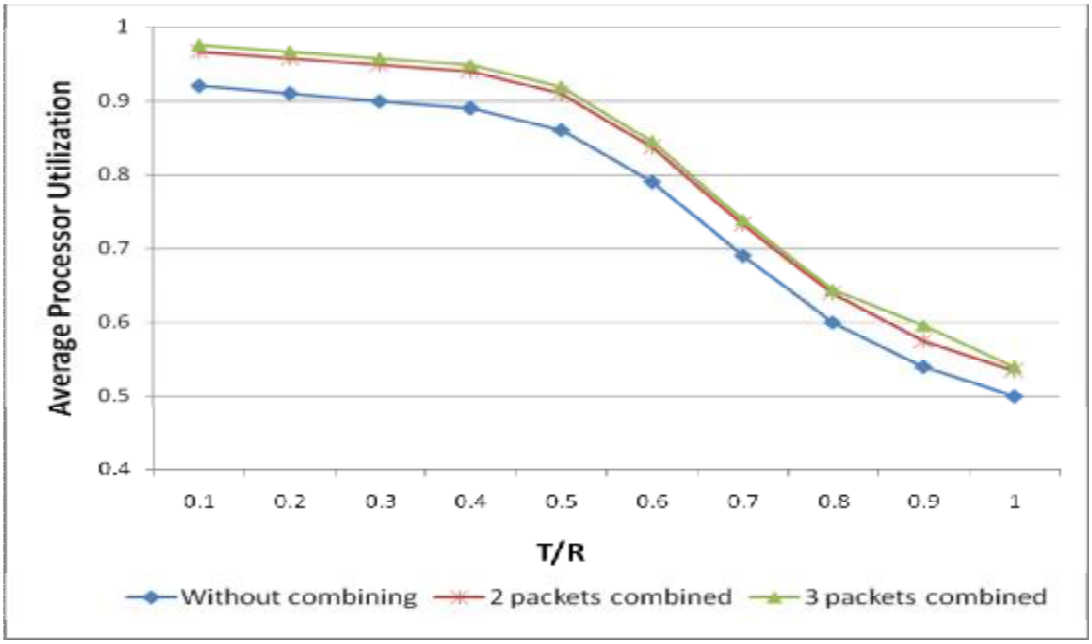

Figure 5.17. Average processor utilization for uniform pattern under client-server traffic model

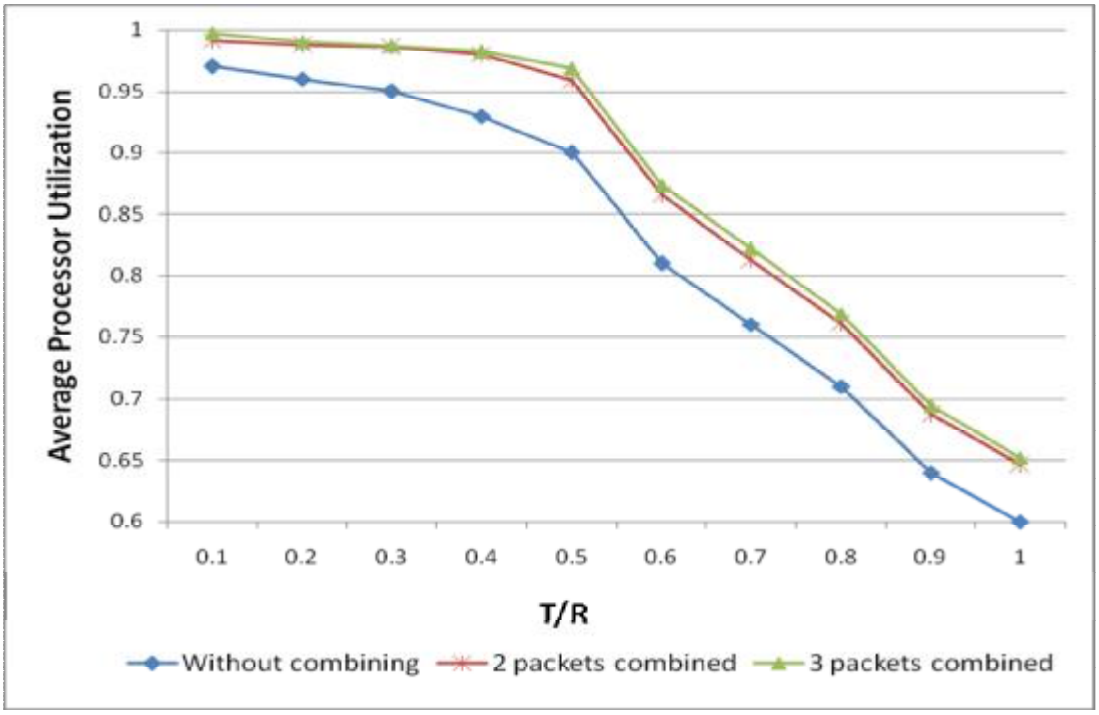

Figure 5.18. Average processor utilization for bit-reverse pattern under client-server traffic model

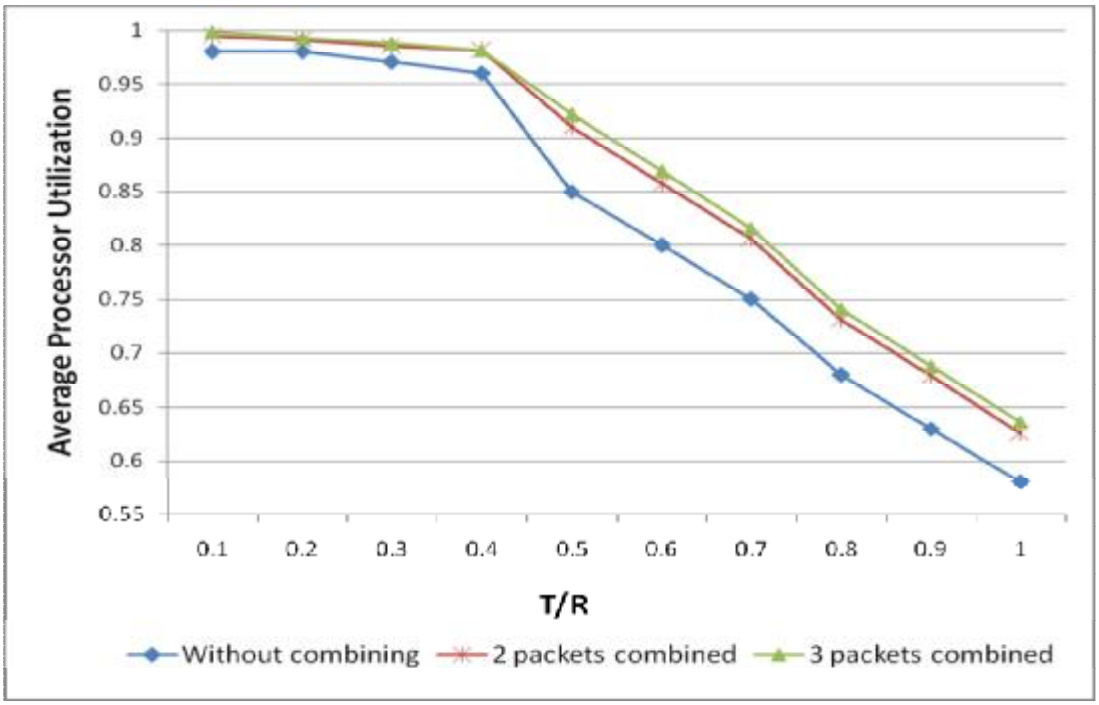

Figure 5.19. Average processor utilization for perfect shuffle pattern under client-server traffic model

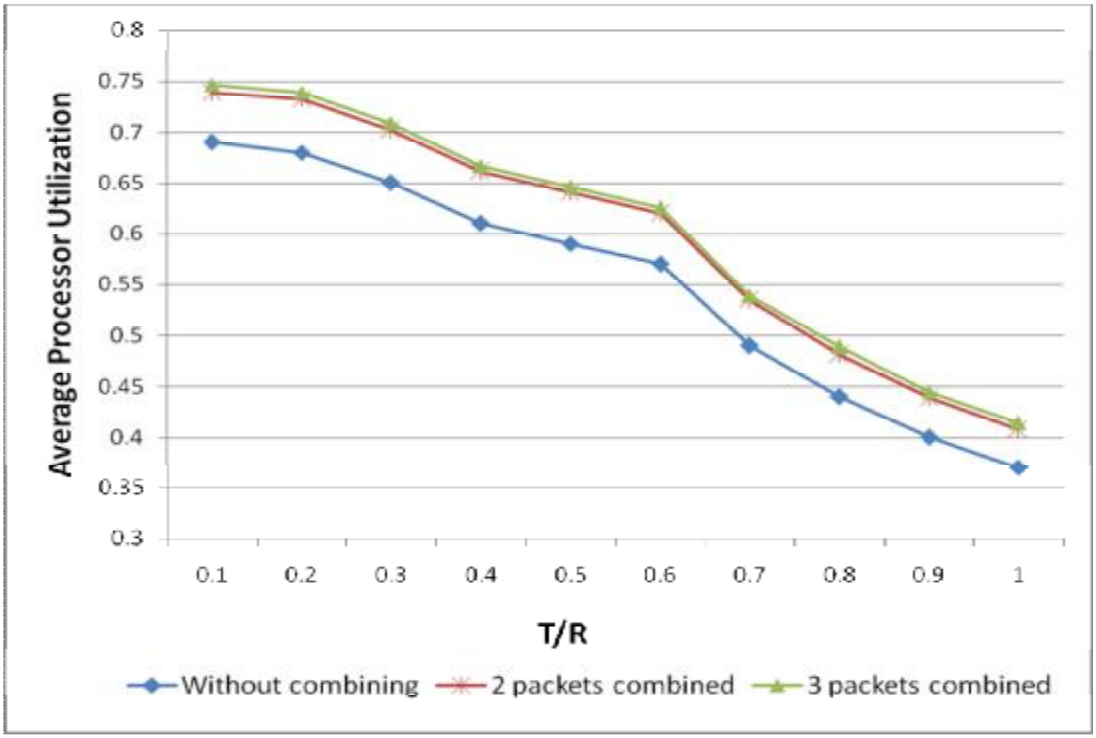

Figure 5.20. Average processor utilization for hot-region pattern under client-server traffic model

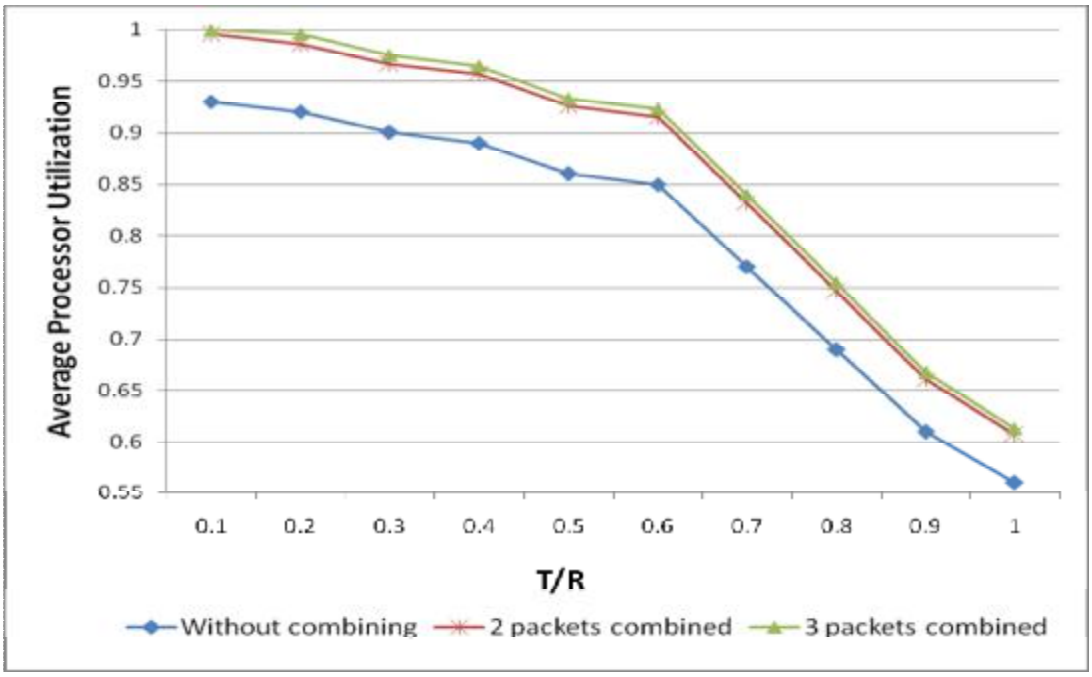

Figure 5.21. Average processor utilization for uniform pattern under asynchronous traffic model

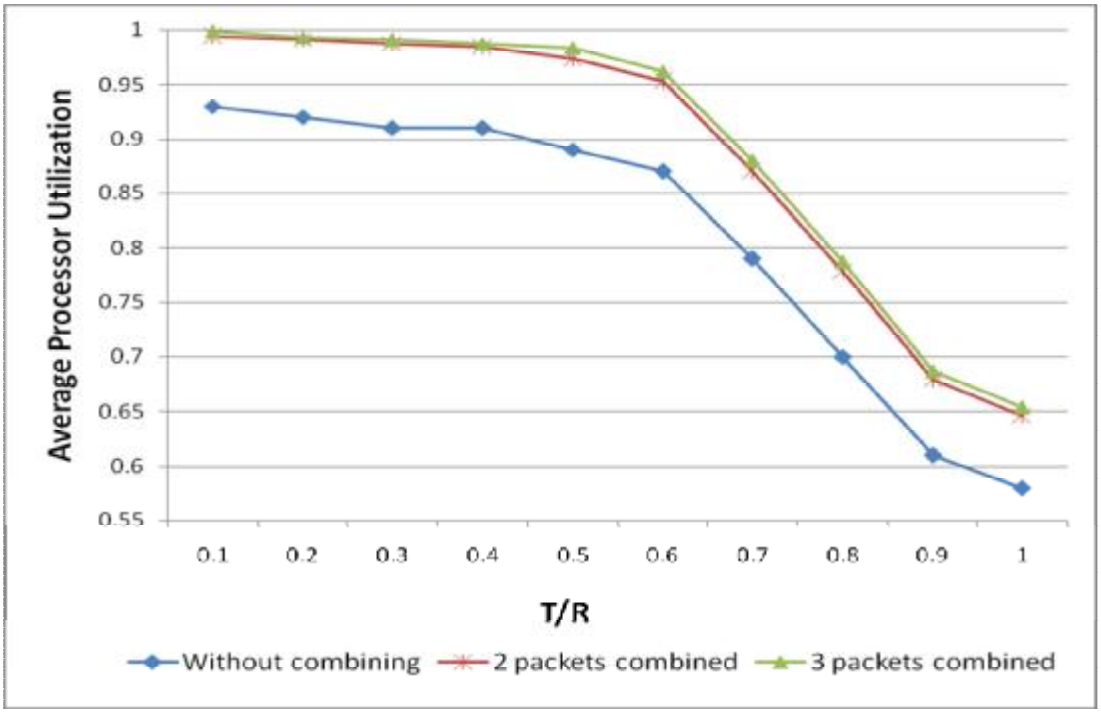

Figure 5.22. Average processor utilization for bit-reverse pattern under asynchronous traffic model

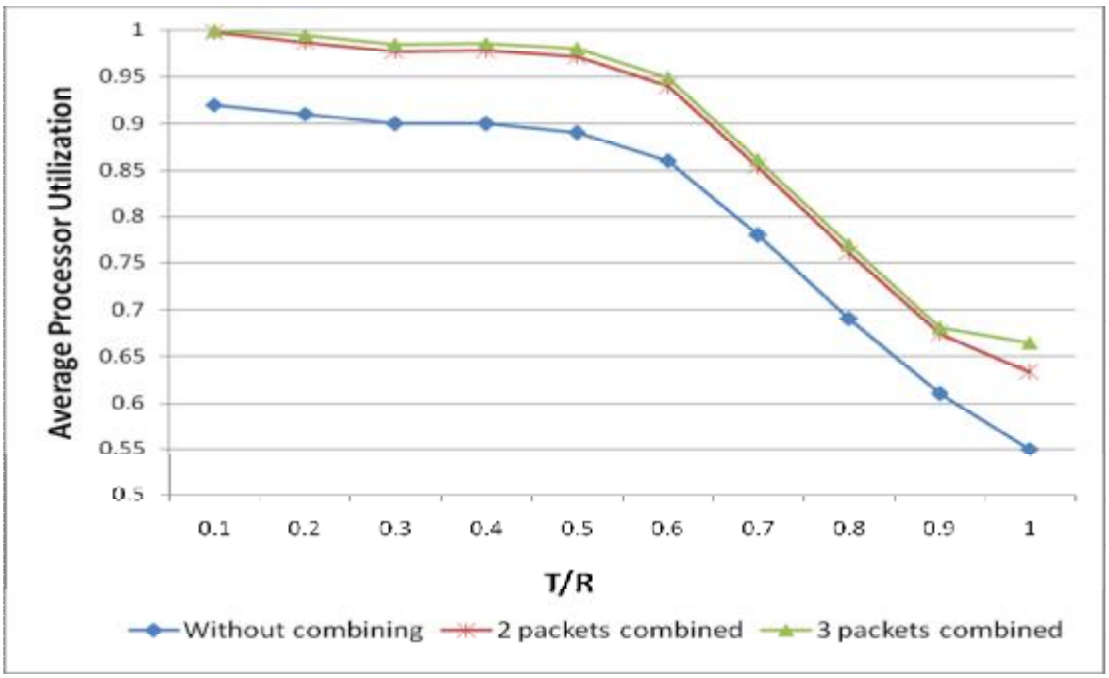

Figure 5.23. Average processor utilization for perfect shuffle pattern under asynchronous traffic model

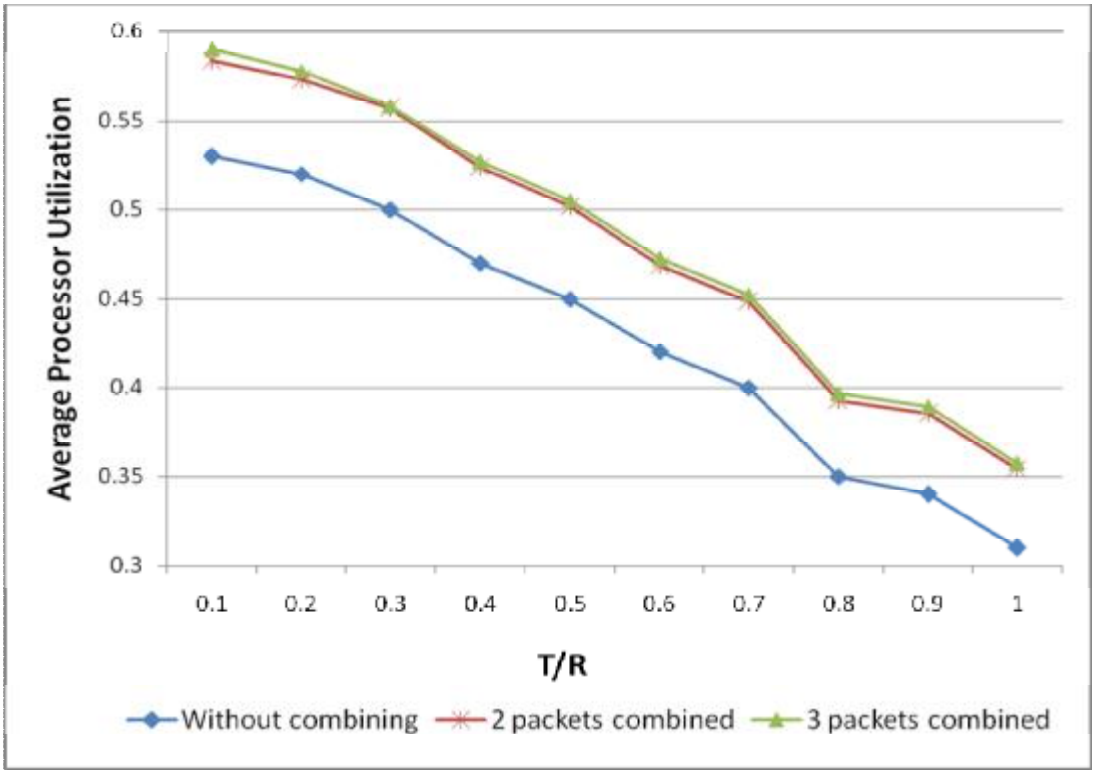

Figure 5.24. Average processor utilization for hot-region pattern under asynchronous traffic model

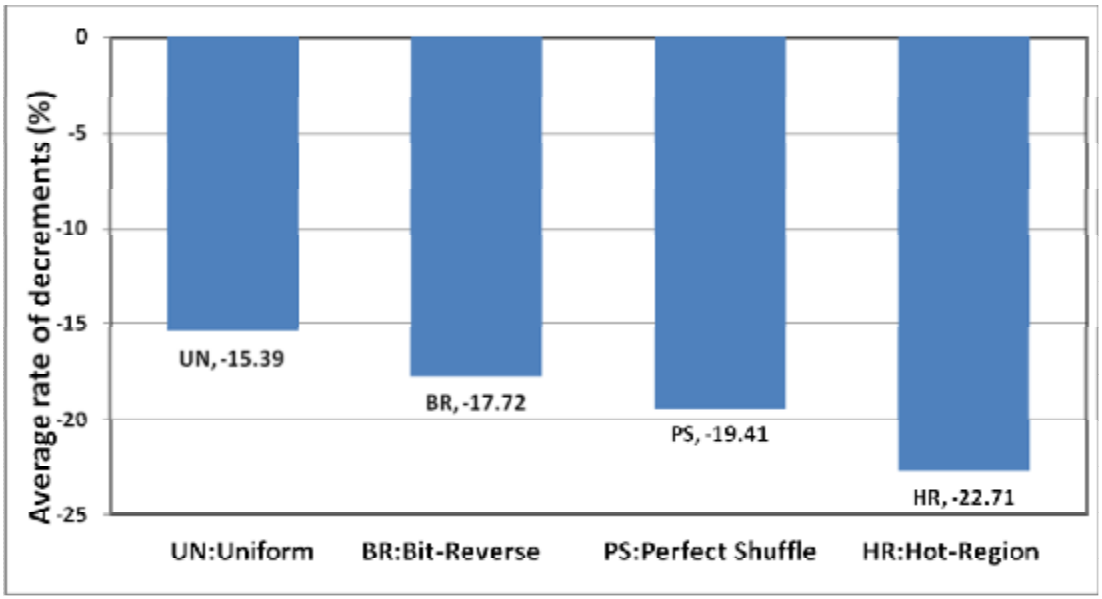

Figure 5.25. Average rate of decrements for channel waiting time under client-server traffic model with two message combining

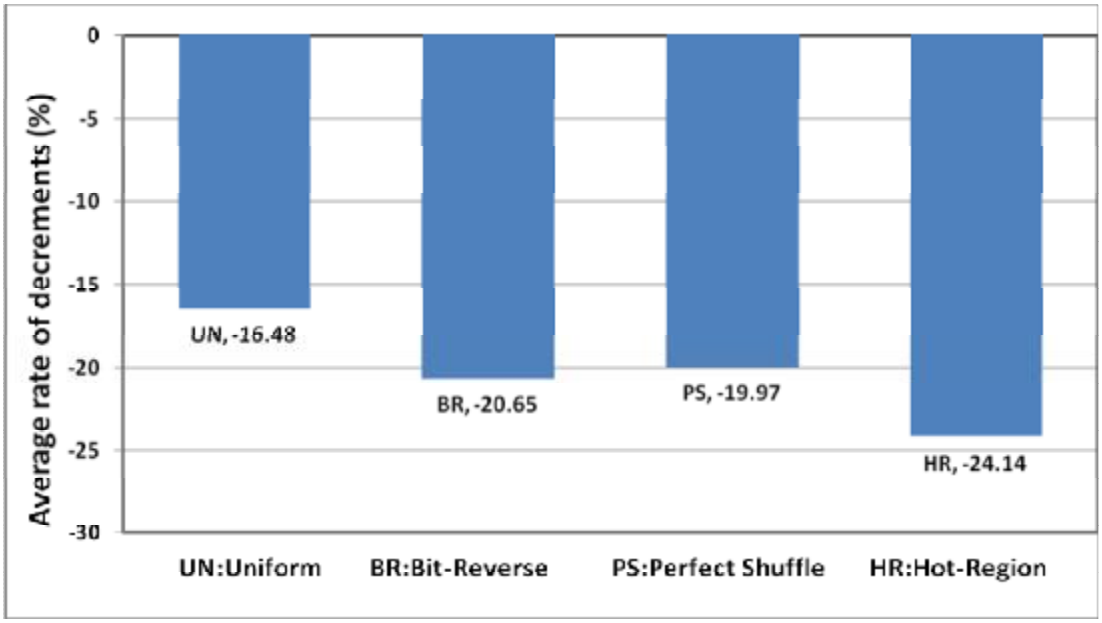

Figure 5.26. Average rate of decrements for channel waiting time under client-server traffic model with three message combining

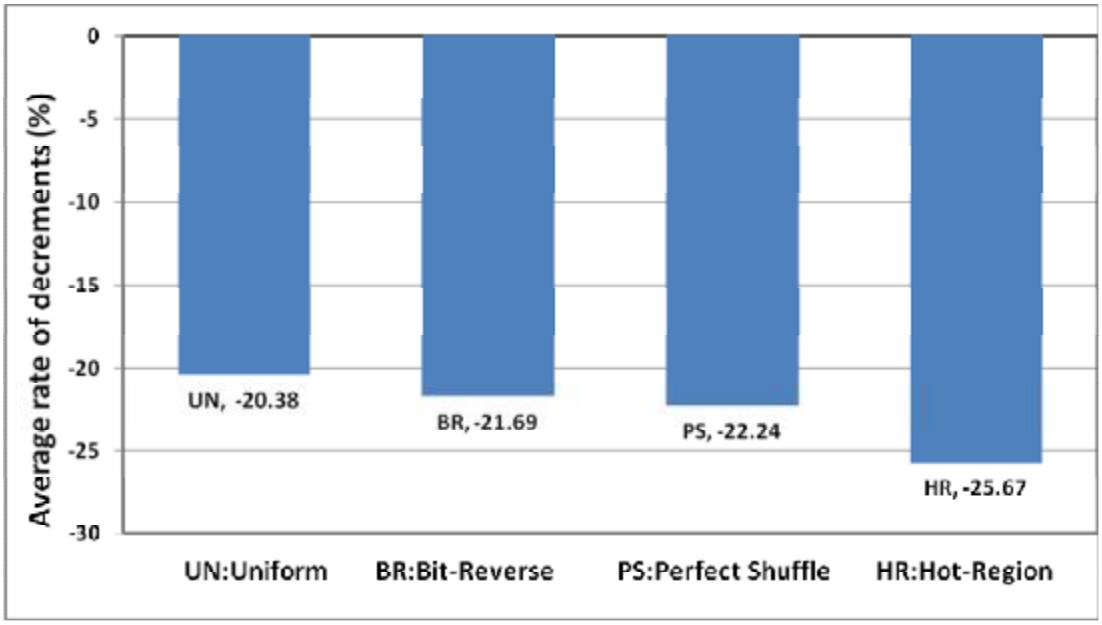

Figure 5.27. Average rate of decrements for channel waiting time under asynchronous traffic model with two message combining

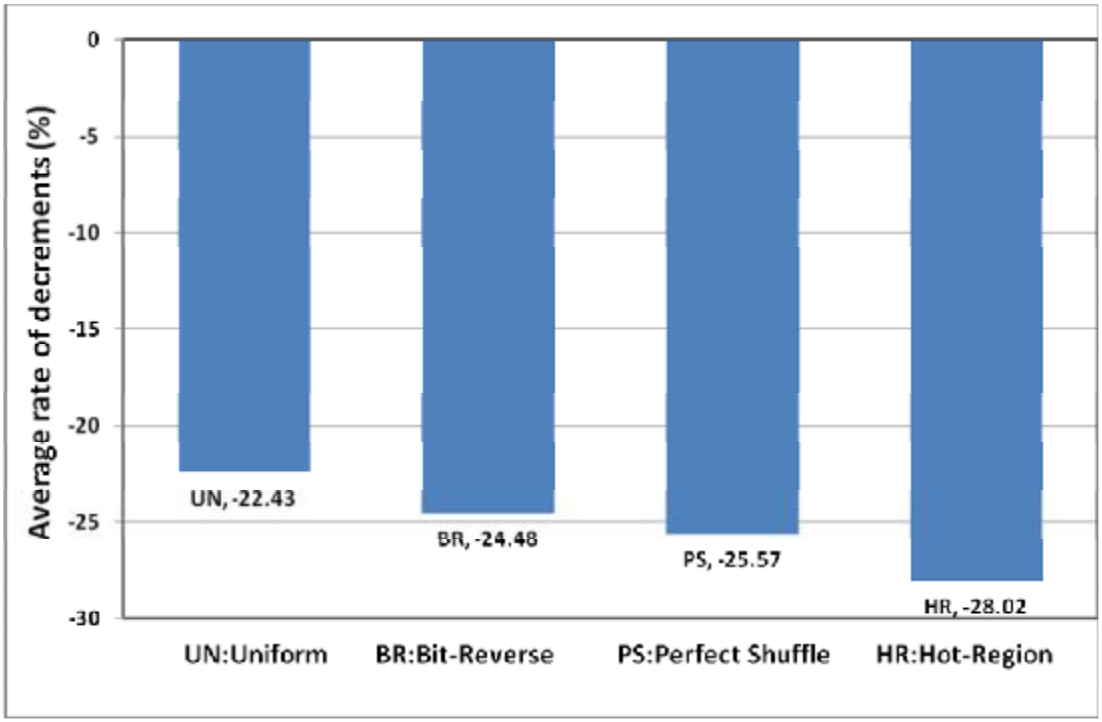

Figure 5.28. Average rate of decrements for channel waiting time under asynchronous traffic model with three message combining

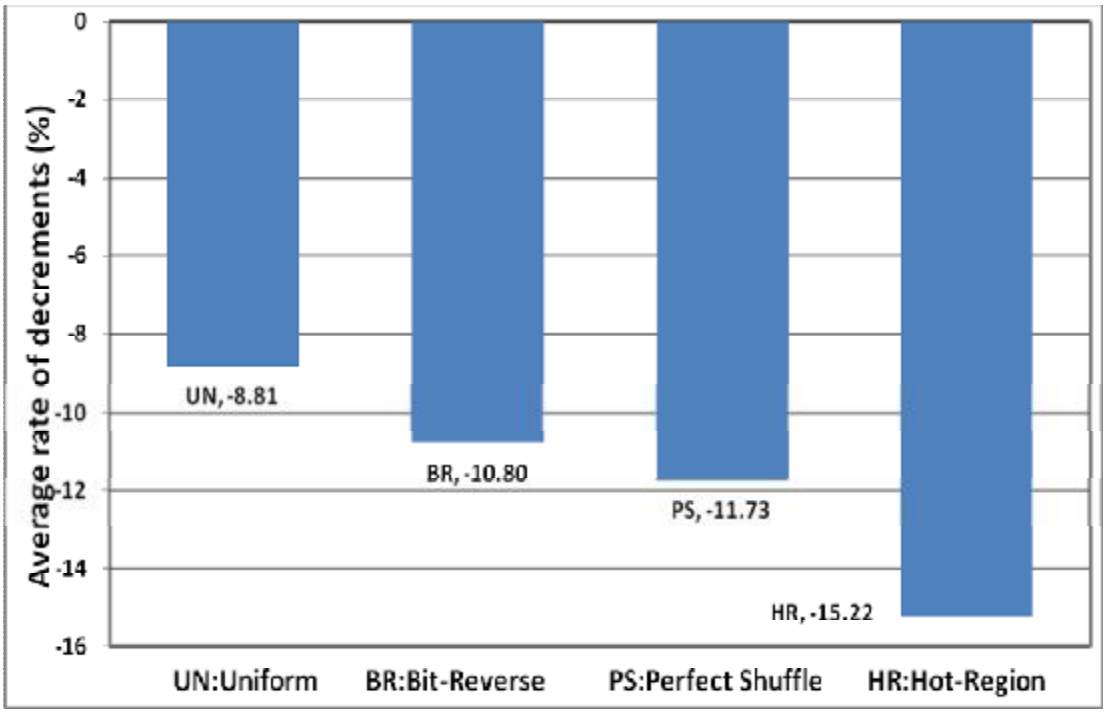

Figure 5.29. Average rate of decrements for network response time under client-server traffic model with two message combining

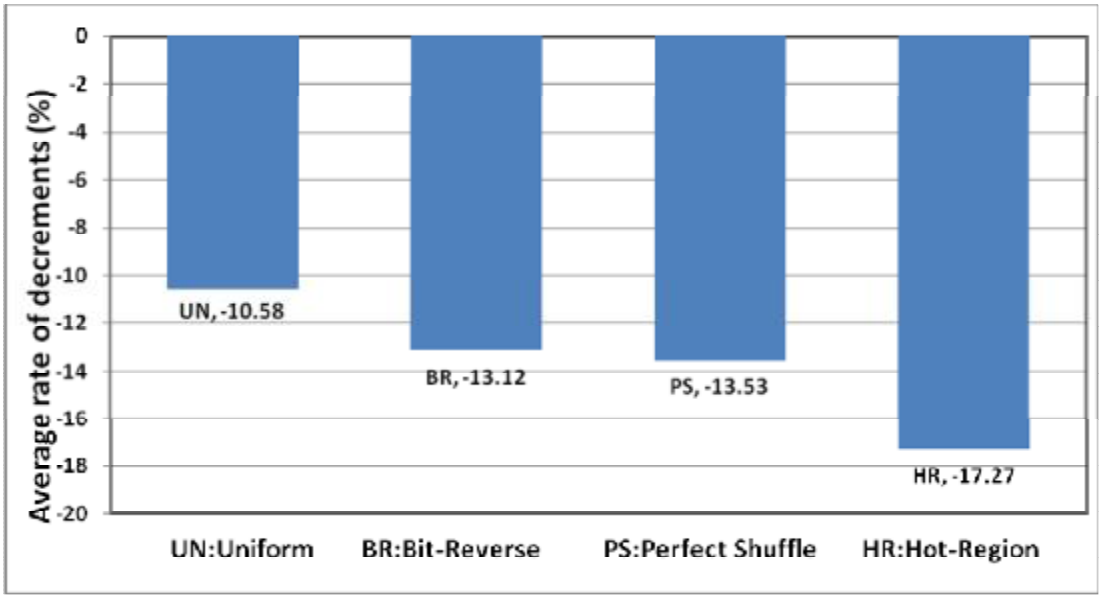

Figure 5.30. Average rate of decrements for network response time under client-server traffic model with three message combining

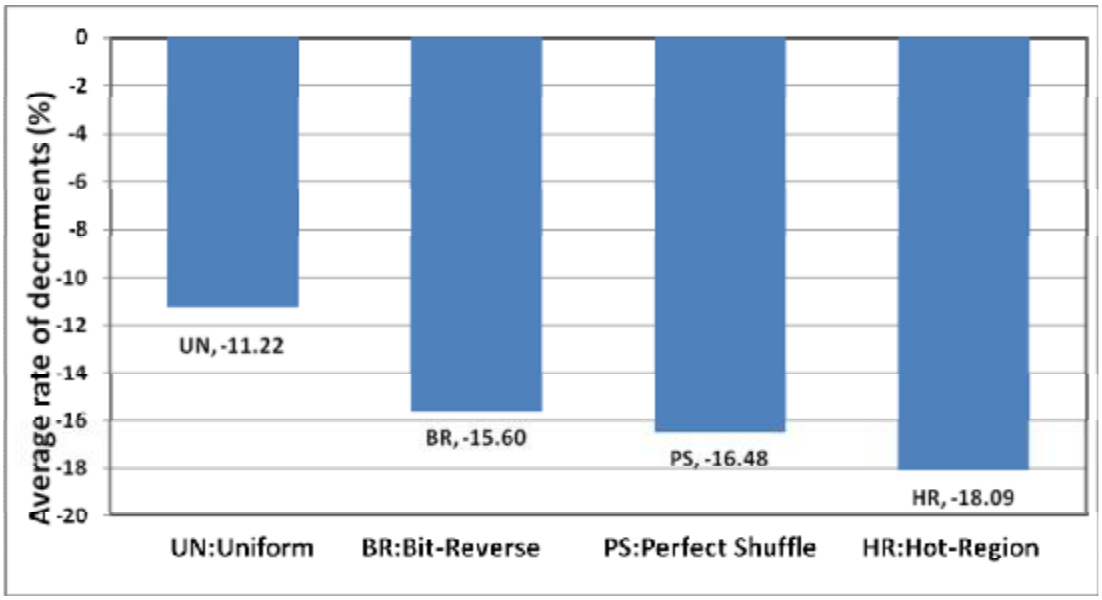

Figure 5.31. Average rate of decrements for network response time under asynchronous traffic model with two message combining

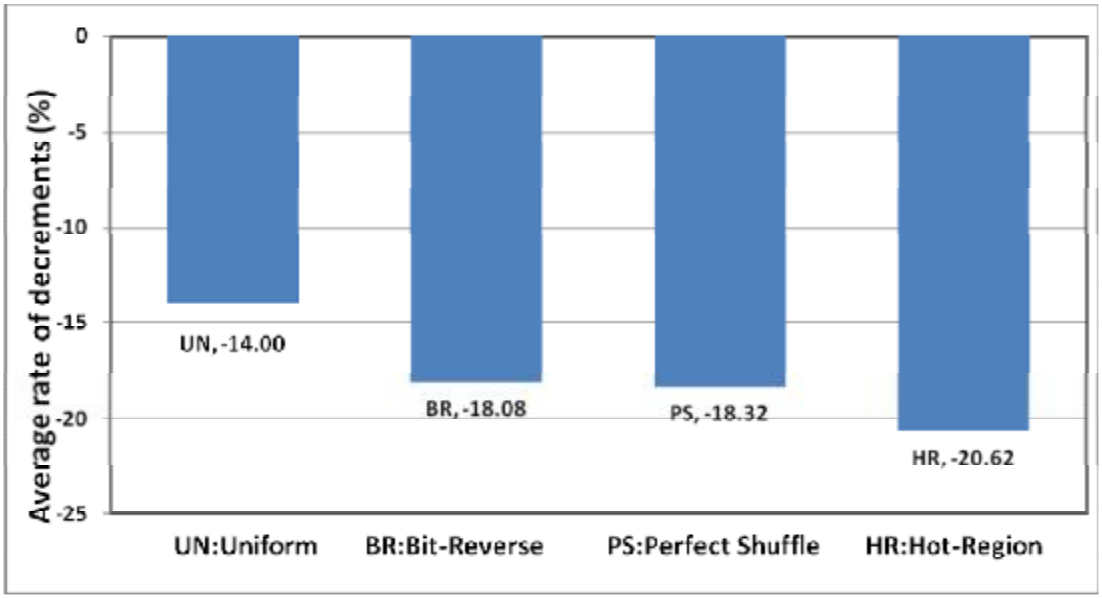

Figure 5.32. Average rate of decrements for network response time under asynchronous traffic model with three message combining

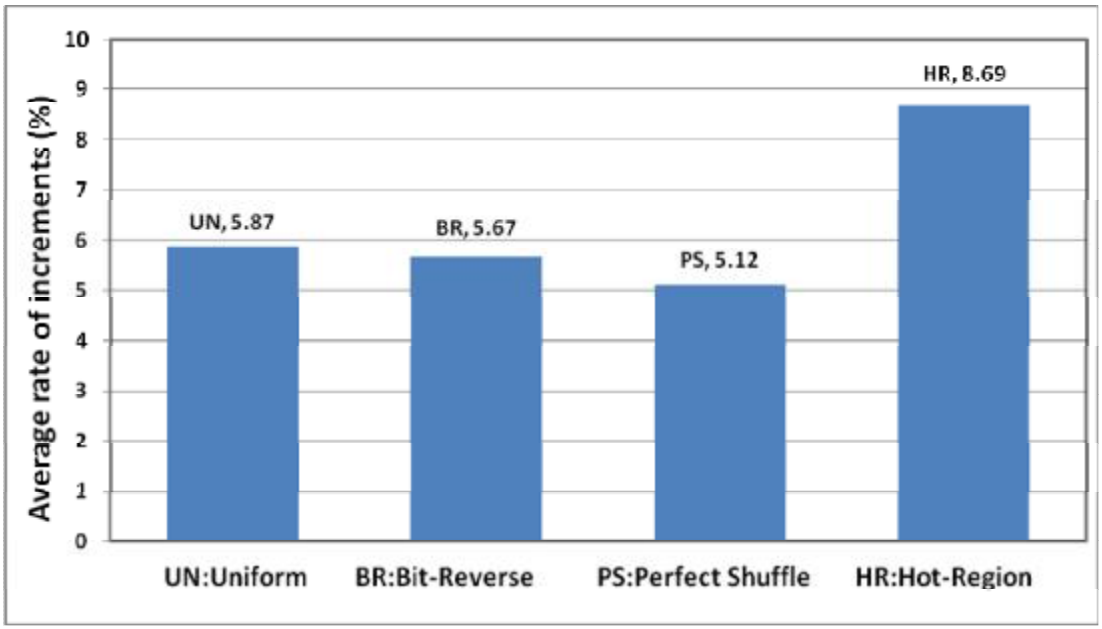

Figure 5.33. Average rate of increments for processor utilization under client-server traffic model with two message combining

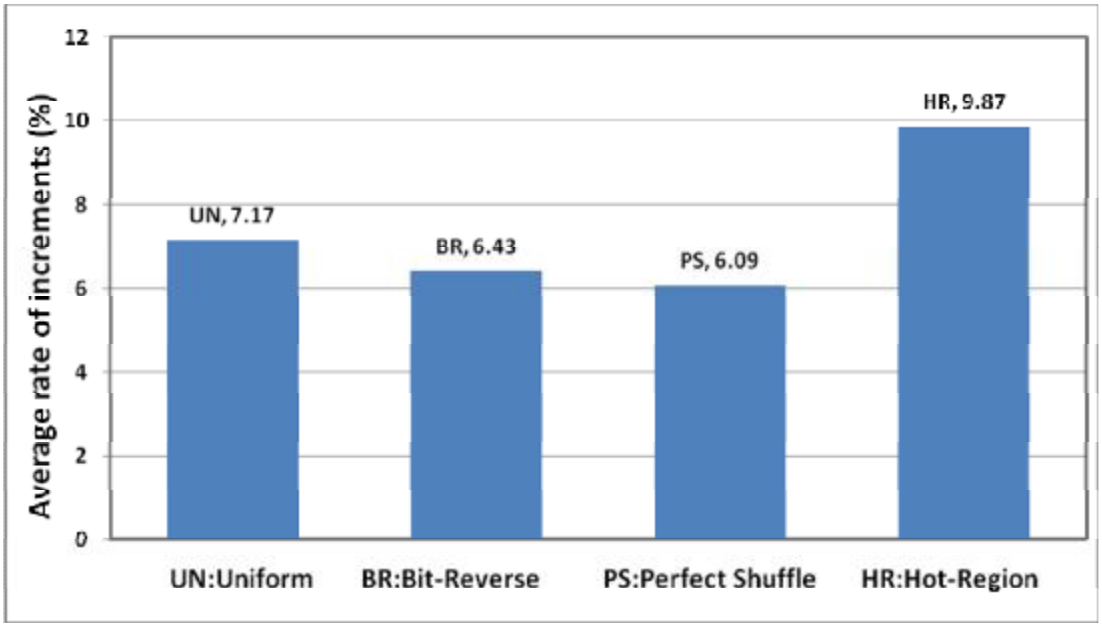

Figure 5.34. Average rate of increments for processor utilization under client-server traffic model with three message combining

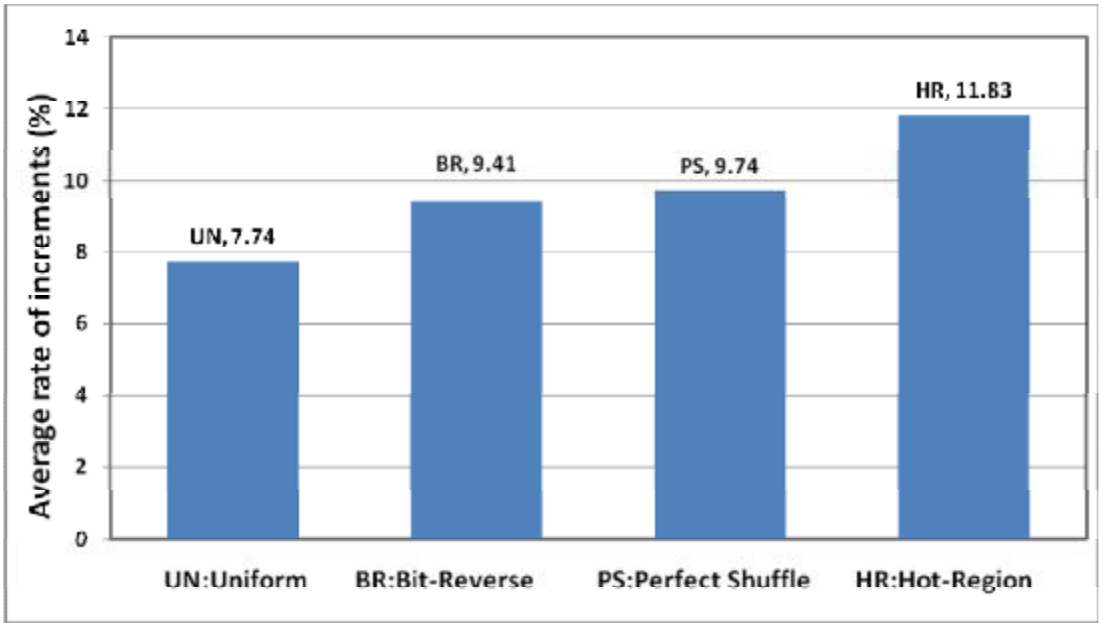

Figure 5.35. Average rate of increments for processor utilization under asynchronous traffic model with two message combining

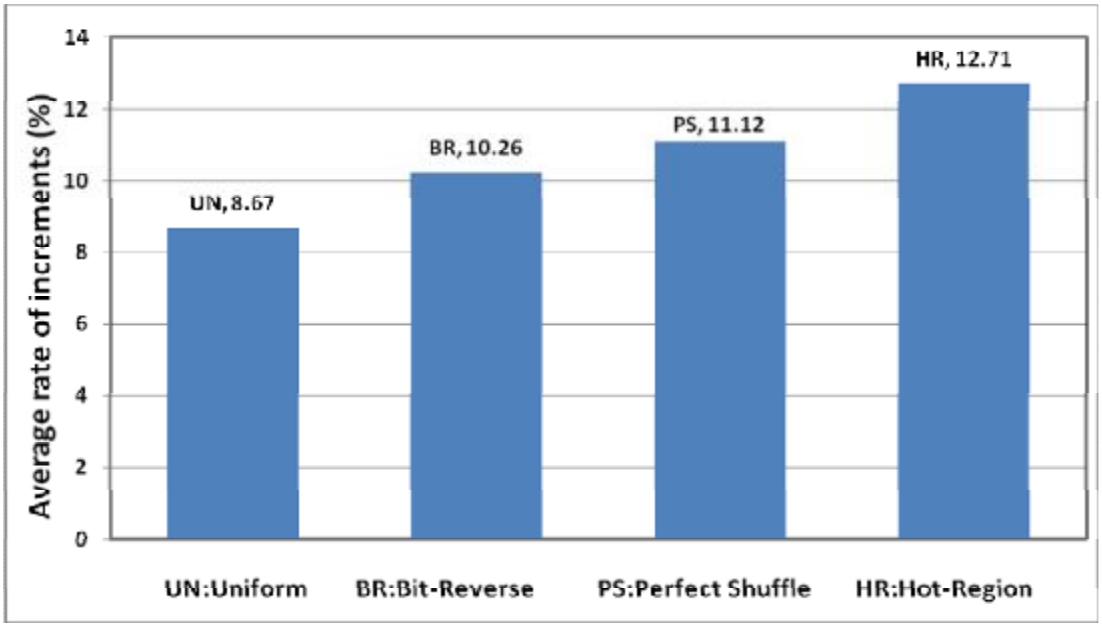

Figure 5.36. Average rate of increments for processor utilization under asynchronous traffic model with three message combining

1. The best performance for all traffic patterns is observed when the number of combined messages is 3. The performance of the architecture decreases as the number of combined messages goes beyond 3.
2. The average rate of decrements in network response times under client server model for uniform, bit reverse, perfect shuffle and hot region traffic patterns are 9.7%, 11.96%, 12.63% and 16.24%, respectively.
3. The average rate of decrements in channel waiting times under client server model for uniform, bit reverse, perfect shuffle and hot region traffic patterns are 15.93%, 19.19%, 19.16% and 23.43%, respectively.
4. The average rate of increments in processor utilizations under client server model for uniform, bit reverse, perfect shuffle and hot region traffic patterns are 6.52%, 6.05%, 5.61% and 9.28%, respectively.
5. The average rate of decrements in network response times under asynchronous model for uniform, bit reverse, perfect shuffle and hot region traffic patterns are 12.61%, 16.84%, 17.4% and 19.76%, respectively.

6. The average rate of decrements in channel waiting times under asynchronous model for uniform, bit reverse, perfect shuffle and hot region traffic patterns are 21.24%, 23.09%, 23.91% and 26.85%, respectively.
7. The average rate of increments in processor utilizations under asynchronous model for uniform, bit reverse, perfect shuffle and hot region traffic patterns are 8.21%, 9.84%, 10.43% and 12.27%, respectively.
8. The performance of the combining method on the 2D SOME-Bus architecture is better for the asynchronous traffic pattern.
9. Both for client server and asynchronous models, the highest rate of decrement in average network times and average channel waiting times is observed for the hot region traffic pattern.
10. Both for client server and asynchronous models, the highest rate of increment in processor utilizations is observed for the hot region traffic pattern.
11. Both for client server and asynchronous models, the lowest rate of decrement in average network times is observed for the uniform traffic pattern.
12. Both for client server and asynchronous models, the lowest rate of increment in processor utilizations is observed for the uniform traffic pattern.
13. Increasing the number of combined messages from 2 to 3 yields 2.92% decrement in average channel waiting times.
14. Increasing the number of combined messages from 2 to 3 yields 2.44% decrement in average network response times.
15. Increasing the number of combined messages from 2 to 3 yields 0.93% increment in average processor utilizations.

6. CONCLUSION

Because parallel architectures evolve towards general-purpose systems with increasing size; hot-spot contention becomes a serious problem affecting the whole system's performance. It is imperative that these systems effectively manage or avoid system bottlenecks such as hot-spots. To minimize the effect of hot-spots in broadcast architectures, messages are combined to form a new message containing the payloads of all original messages even when they are destined for distinct nodes.

The performance of the message combining method is evaluated via simulation using synthetic traffic patterns (hot-region, uniform, perfect shuffle and bit-reverse) on a 64 node 2D SOME-Bus multiprocessor architecture. Initially, we observe the performance of the base system (without message combining) for varying values of T/R. Then, we apply message combining method by combining different number of messages waiting in the channel queues of the system. Performance of the message combining method on the 2D SOME-Bus architecture is assessed by measuring the average network response time, average channel waiting time and average processor utilization before and after applying the message combining method.

The results show that the combining method is able to decrease the average network response time, average channel waiting time and increase the average processor utilization. The message combining method performs the best under hot region traffic. Combining three messages yields better performance than combining two messages. Under asynchronous traffic model, the decrement rate of the average network response time and the average channel waiting time is higher than that of the average network response time and the average channel waiting time under client-server traffic model after the application of the message combining method. Also, the increment rate of the processor utilization under asynchronous traffic model is higher than that of the processor utilization under client-server traffic model.

The results reveal that average network response times decrease by 8.81% to 20.62%, average channel waiting times decrease by 15.39% to 28.02% and average processor utilization, increase by 5.12% to 12.71% after applying the combining

method. In conclusion, there is significant performance increase after applying the message combining method.

REFERENCES

ABASIKELES, I., AKAY, M.F., 2010. Performance Evaluation Of Directory Protocols On An Optical Broadcast-Based Distributed Shared Memory Multiprocessor. Journal Of Computer And Electrical Engineering, 36:114-131.

ALMASI, G.S., and GOTTLİEB, A.,1994. Highly Parallel Computing. Second Edition, Benjamin Cummings, Redwood City, California.

ASANOVİC, K., BODİK, R., CATANZARO, B.C., GEBİS, J.J., HUSBANDS, P., KEUTZER, K., PATTERSON, D.A., PLİSHKER, W.L., SHALF, J., WİLLİAMS, S.W., and YELİCK, K.A.,2006. The Landscape Of Parallel Computing Research: A View From Berkeley. EECS Department, University Of California At Berkeley, Technical Report No UCB/EECS-2006-183, California, 56p.

BERGMAN, K., 2008. Optical Interconnection Networksin Advanced Computing Systems. Department Of Electrical Engineering, Columbia University, Newyork, Ny,Usa, pp. 765-801.

BLUMRICH, M. ET AL., 2003. Design And Analysisof The Bluegene/L Torus Interconnection Network, Ibm Research Report RC23025, 1-9.

DALLY, W.J., TOWELS, B., 2004. Principles And Practices Of Interconnection Networks, Morgan-Kaufmann, San Francisco, 49-50.

DİCKEY S., GOTTLİEB A. AND LİU Y. S., "Interconnection Network Switch Architectures And Combining Strategies", Ultracomputer Research Laboratory, Courant Institute Of Mathematical Sciences, New York University, 1994.

DUNCAN, R.,1990. A Survey Of Parallel Computer Architectures. Computer, 23(2): 5-16.

GRAMA, A., GUPTA, A., KARYPİS, G., and KUMAR, V.,2003. Introduction To Parallel Computing. Second Edition, Addison Wesley, 856p.

HANAVA T., FUJİWARA T., and AMANO H., 1996. Hot Spot Contention And Message Combining İn The Simple Serial Synchronized Multistage Interconnection Network. Eighth Ieee Symposium On Parallel And Distributed Processing, pp. 298-305.

KATSINIS C., 1999. Performance Models Of An Interconnection Network For Broadcast Communication. PDPTA'99 International Conference On Parallel And Distributed Processing Techniques And Applications, Las Vegas, Nevada, USA, pp. 2231-2237.

KATSINIS C. and HECHT D.,2000. Hot-Spots And Message Combining İn An Interconnection Network For Broadcast Communication. The 2000 Int. Conf. On Parallel And Distributed Processing Techniques And Applications, 2, June.

KATSINIS, C.,2001. Performance Analysis Of The Simultaneous Optical Multiprocessor Exchange Bus. Journal of Parallel Computing, 27:1079–1115.

KATSINIS C., 2004. Hot-Spots On An Interconnection Network For Broadcast Communication. Computer Systems Science And Engineering, 19: 49-56.

KATSINIS C. and HECHT D.,2004. Fault-Tolerant Dsm On The Some-Bus Multiprocessor Architecture With Message Combining. Proceedings Of The 18th International Parallel And Distributed Processing Symposium (IPDPS'04), 12:210.

KATSINIS C. and NABET B., 2004. A Scalable Interconnection Network Architecture For Petaflops Computing", The Journal Of Supercomputing, 27:103–128.

LEE G. G., KRUSKAL C.P. AND KUCK D.J., 1994. On The Effectiveness Of Combining In Resolving Hot Spot Contention. Journal Of Parallel And Distributed Computing, 20: 136-144.

QUAMMEN, C.W.,2005. Introduction To Programming Shared-Memory And Distributed-Memory Parallel Computers, Acm Crossroads, NY, USA, 12(4):3–9.

RAVELA, S. C., 2010. Comparison Of Shared Memory Based Parallel Programming Models. M.S. Thesis, Computer Science, School Of Computing, Blekinge Instıtute Of Technology, Sweden, 68p.

SAURAVLAS S. and ROUMELİOTİS M., 2010. A Message Combining Approach For Efficient Array Redistribution In Non-All-To-All Communication Networks. International Journal Of Computer Mathematics,Greece.

SHAFER S. AND GHOSE K., 1994. Improving Parallel Program Execution Time With Message Consolidation. 8th Parallel Processing Symposiıum, Mexico.

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.

BIOGRAPHY

Ms. Gülsade KALE was born in Gaziantep, in 1978. She completed her elemantary school in Germany. She graduated from Fitnat Nuri Tekerekoğlu Anatolian High School in 1996. She got her BSc degree from the Department of Computer Engineering of Eastern Mediterranean University (EMU) in 2003. She received her MSc degree in Computer Education and Instructional Technology Department of Ege University in 2009. Since 2009, she has been serving as a department head and working as an instructor at the Computer Technologies Department of Kilis 7 Aralık University.

Her research interests are parallel computing and multiprocessor architectures.