# Throughput Comparison of Automatic Repeat Request Assisted Butterfly Networks

Yang Qin and Lie-Liang Yang
School of ECS, University of Southampton, SO17 1BJ, United Kingdom
Tel: 0044-(0)23-8059 3364, Email: yq06r,lly@ecs.soton.ac.uk; http://www-mobile.ecs.soton.ac.uk

***Abstract*— The maximum achievable throughput and the steady-state throughput of the Butterfly networks are investigated and compared in the context of three types of automatic repeat request (ARQ) schemes of, namely, the stop-and-wait ARQ (SW-ARQ), go-back-$N$ ARQ (GBN-ARQ) and the selective-repeat ARQ (SR-ARQ). Our studies show that, at a given packet error rate (PER), the SW-ARQ scheme yields the lowest throughput while the SR-ARQ scheme can attain the highest throughput, among the three ARQ schemes considered. At a very low PER, the GBN-ARQ may achieve a similar throughput as the SR-ARQ. However, as the PER increases, the throughput achieved by the GBN-ARQ converges to that of the SW-ARQ.**

## I. Introduction

Network coding considers coding over packet networks and it has been recognized as one of the techniques that have the potential to improve the capacity of conventional networks [1, 2]. Therefore, since its invention by Ahlswede, Cai, Li and Yeung [1], network coding has drawn a lot of attention. In literature, performance of communications networks employing network coding has been investigated widely mainly under the assumption that packets are conveyed over the networks without errors [3]. However, in practical wired or wireless communications networks, transmission errors always are unavoidable and, usually, error-detection or error-correction techniques are required in order to ensure reliable communications [2, 4]. Therefore, in this contribution, we motivate to study by simulations the achievable throughput of the Butterfly networks, when they are operated in non-ideal communications environments and are protected by an ARQ data transmission scheme [5]

Specifically, in this paper three types of ARQ schemes are studied, which are the SW-ARQ, GBN-ARQ and the SR-ARQ [5]. Both the maximum achievable throughput and the steady-state throughput of the Butterfly networks are investigated. For the maximum achievable throughput, we assume that the source node of a Butterfly network always has packets prepared, whenever it wants to transmit. By contrast, for the steady-state throughput, we assume that packets arriving at the source node follow a Poisson process. Hence, packets at the source node may not always be prepared and the source node may sometimes need to wait, even it is free to transmit.

Previous researches related to this study are briefly summarized as follows. In [6], network coding involving feedback has first been studied. In [7], the problems of applying ARQ techniques to network coding have been addressed. It has been suggested to add an extra network coding layer into the TCP/IP stack, in order to introduce network coding into the existing Internet infrastructure. Furthermore, in [8], a random network coding framework employing hybrid ARQ scheme has been proposed for real-time media broadcast over single-hop wireless networks.

## II. Butterfly Network and Operations

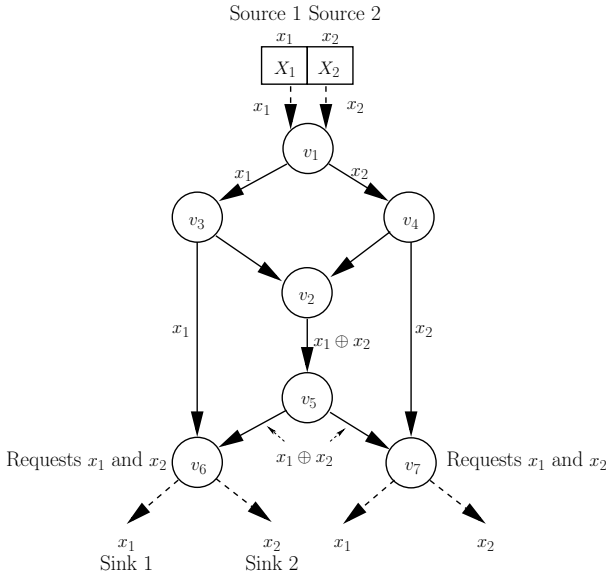


Fig. 1. A butterfly network with one source node $v_1$ multicasting messages $x_1$ and $x_2$ to both the sink nodes $v_6$ and $v_7$ using linear network coding.

The system considered in this contribution is a typical Butterfly network as shown in Fig. 1. Node $v_1$ is a source node residing the two information sources $X_1$ and $X_2$, which generate the packets to be transmitted simultaneously to the sink nodes $v_6$ and $v_7$. Each of the sink nodes is attached with two sinks, for example, Sink 1 and Sink 2 to the sink node $v_6$, as shown in Fig. 1. Node $v_2$ is a two-input-single-output (2ISO) node employing packet-level network coding. Node $v_3$, $v_4$ and $v_5$ are the intermediate nodes, which forward incoming packets to the designated outgoing link(s) without network coding. The function of the intermediate nodes is identical to the conventional "store-and-forward" nodes. Finally, nodes $v_6$ and $v_7$ are two sink nodes expecting information from both sources $X_1$ and $X_2$. For convenience of description, let us define the link set $\mathcal{L} = \{l_{1,3}, l_{1,4}, l_{2,5}, l_{3,2}, l_{3,6}, l_{4,2}, l_{4,7}, l_{5,6}, l_{5,7}\}$, where $l_{i,j}$ is the link transmitting packets from node $v_i$ to node $v_j$. We assume that packets are transmitted over the communication links based on the ARQ strategy. In this paper, three types of

ARQ schemes, namely the SW-ARQ, GBN-ARQ and the SR-ARQ, are studied in the context of the Butterfly networks. In our analysis and simulations, the following assumptions are adopted.

- All the links $\{l_{i,j}|l_{i,j} \in \mathcal{L}\}$ of the Butterfly network employ the same ARQ scheme, SW-ARQ, GBN-ARQ or SR-ARQ.
- The $n$th packet pair simultaneously generated by $X_1$ and $X_2$ are denoted as $x_1(n)$ and $x_2(n)$, respectively, where $n$ is defined as the generation number. Let $x(n) = (x_1(n), x_2(n))$ denote the $n$th packet pair. We assume that $x(n)$ arrives at $v_1$ immediately after it is generated.
- Each of the links in $\mathcal{L}$ is divided into two channels: the forward channel and the feedback channel. The forward channel is assumed to be a binary symmetric channel (BSC). The probability of detectable packet errors of the forward channel is the same for every link, which is denoted as $p_e$ simply referred to as the PER. We assume that the probability of undetectable packet errors is very small and can be ignored. This is usually true, since for most error-control codes adopted in practical communications systems, the probability of undetectable errors is very small, in comparison with the probability of detectable errors. Furthermore, we assume that the link outage rate is zero, implying that no packet is lost. Additionally, the feedback channels are assumed ideal, which do not generate erroneous feedbacks.
- The round trip time (RTT) is denoted by $T$, which represents the time duration between that a node sends a packet and that it receives a confirmation signal. We assume that half of a RTT, i.e., $T/2$, is required for transmitting a packet from a transmit node to a receive node by the corresponding forward channel. Similarly, half a RTT is required for sending a confirmation signal from the receive node to the transmit node using the corresponding feedback channel.
- The duration of packets is assumed to be much shorter than $T$ of the RTT and can be ignored. Furthermore, the processing time of packets can also be ignored. Alternatively, we may view that the average duration of packets as well as the average processing time are included in the RTT.
- Each of the nodes is assumed to have an infinite buffer for storing the packets, whenever necessary.

Based on the above assumptions, the operations of the packets over the Butterfly network of Fig. 1 are carried out as follows.

As shown in Fig. 1, we assume that two unit rate messages (packets) $x_1(n), x_2(n)$ are generated by the information sources $X_1$ and $X_2$, which are attached to the source node $v_1$. We assume for simplicity that both $x_1(n)$ and $x_2(n)$ take values in the field $\mathbb{F} = GF(2)$. As shown in Fig. 1, message $x_1(n)$ is directly sent to the sink node $v_6$ via the route $X_1 \rightarrow v_1 \rightarrow v_3 \rightarrow v_6$. Similarly, message $x_2(n)$ is directly sent to the sink node $v_7$ via the route $X_2 \rightarrow v_1 \rightarrow v_4 \rightarrow v_7$. As shown in Fig. 1, both the messages $x_1(n)$ and $x_2(n)$ are also sent to node $v_2$ via the route $X_1 \rightarrow v_1 \rightarrow v_3 \rightarrow v_2$ for $x_1(n)$ and $X_2 \rightarrow v_1 \rightarrow v_4 \rightarrow v_2$ for $x_2(n)$, respectively. At $v_2$, once $x_1(n)$ and $x_2(n)$ are successfully received, they are encoded to form the message $x_1(n) \oplus x_2(n)$ based on the modulo-2 addition over the field $\mathbb{F} = GF(2)$. The output message $x_1(n) \oplus x_2(n)$ is then sent to the sink nodes $v_6$ and $v_7$ via the route $v_2 \rightarrow v_5 \rightarrow v_6$ and $v_2 \rightarrow v_5 \rightarrow v_7$, respectively. Finally, after the sink nodes $v_6$ and $v_7$ receive the message $x_1(n) \oplus x_2(n)$, the sink node $v_6$ recovers $x_2(n)$ by the operation $x_2(n) = x_1(n) \oplus (x_1(n) \oplus x_2(n))$, while the sink node $v_7$ recovers $x_1(n)$ based on the operation $x_1(n) = x_2(n) \oplus (x_1(n) \oplus x_2(n))$. Consequently, both of the sink nodes $v_6$ and $v_7$ can obtain the messages $x_1(n)$ and $x_2(n)$ generated by the sources $X_1$ and $X_2$, respectively.

From the above description as well as Fig. 1, we can see that, in the Butterfly network, there are two types of packet transmission paths. The first type of transmission paths do not go through the coding node $v_2$, while the second type contain the coding node $v_2$. The data transmission on the paths without involving the coding node is the same as that on the conventional relay links, where each node just stores the packet received from the incoming link and then forwards it to the next node using its outgoing link. By contrast, the paths containing the coding node behave differently from conventional relay links. At the coding node, rather than just forwarding the incoming packets, for each outgoing link, the coding node needs to wait for all the required incoming packets involved in order to form a corresponding outgoing coded packet.

## III. Automatic Repeat Request Schemes

In the Butterfly network as shown in Fig. 1, the packets are conveyed from one node to another protected by an ARQ data transmission scheme. It is well known that ARQ is an efficient error-control method for data transmission, which uses re-transmissions to achieve reliable data transmission over unreliable links [9]. In general, ARQ-based data transmission is operated as follows. After receiving a packet from the transmitter, an feedback is sent by the receiver to the transmitter to inform the status (correct or incorrect) of the received packet. To be more specific, an acknowledgement (ACK) is fed back to acknowledge a correct packet reception, while a negative-acknowledgement (NACK) is fed back for informing an incorrect packet reception. Furthermore, the transmitter may not receive any information from the receiver during some time, which is usually referred to as the timeout. This occurs, for example, when the feedback information gets lost over the feedback channel. Therefore, in the context of the ARQ data transmission, if the transmitter receives an ACK from the feedback channel, it then transmits the next packet. Otherwise, if the transmitter does not receive any feedback before the timeout or when the transmitter receives a NACK from the feedback channel, it then retransmits the packet. This process may continue until the transmitter receives an ACK.

ARQ protocols include many different schemes [5, 9]. In this study, we specifically consider three types of ARQ schemes, which are the stop-and-wait ARQ (SW-ARQ), go-back-$N$ ARQ (GBN-ARQ) and the selective-repeat ARQ (SR-ARQ). Below we briefly describe their principles.

The SW-ARQ is the simplest ARQ method, which is operated as follows. The transmitter transmits a single packet and

then waits for an ACK. When errors occur during the transmission and the errors are detected by the receiver, the receiver then discards the corrupted packet and sends a NACK to the transmitter, in order to inform the transmitter to retransmit the corrupted packet. On the other hand, if a packet is lost during the transmission, the receiver will certainly not respond since it is not aware of the transmission. In this case, if the transmitter does not receive a feedback (ACK or NACK) from the receiver within the timeout period, the packet is then retransmitted by the transmitter.

In the context of the go-back-$N$ ARQ (GBN-ARQ), the transmitter continues to send a number of packets specified by a window size without regarding to the feedbacks from the receiver. With the GBN-ARQ, after each transmission, the transmitter sets an ACK timer for each of the packets transmitted. Once the transmission of a packet is timeout or is confirmed in error by a NACK, the transmitter retransmits the packet as well as all its subsequent packets transmitted. In more detail, the GBN-ARQ carries out the following operations, if packets or feedbacks are not successfully delivered [9]. Here, we assume that node $A$ transmits packet $i$ to node $B$. First, if packet $i$ is corrupted or lost, there are corresponding three types of operations:

- If packet $i$ is corrupted in transmission and $B$ detects the error and discards the corrupted packet $i$, then $B$ sends back an NACK indicating that packet $i$ is rejected. When $A$ receives this NACK, it retransmits this packet and all the subsequent packets transmitted after packet $i$. After the retransmission, $A$ proceeds to transmit the other packets that wait in the queue.
- If packet $i$ is lost during its transmission from $A$ to $B$ and $A$ subsequently transmits packet $(i+1)$, then $B$ receives packet $(i+1)$ and finds that it is out of order. In this case $B$ feeds back a NACK $i$. After receiving the NACK, $A$ then retransmits packet $i$ as well as the following packets.
- If packet $i$ is lost during its transmission and $A$ does not transmits the subsequent packets, then $B$ receives nothing and will hence not respond. Consequently, at $A$ the timeout will be activated to retransmit packet $i$.

When node $A$ transmits the $i$th packet to node $B$, if the corresponding ACK is corrupted or lost, two types of operations may occur:

- If node $B$ soon receives packet $(i+1)$ and feeds back the $(i+1)$th ACK. If this ACK is correctly conveyed, node $A$ believes that packet $(i+1)$ as well as all the previous packets transmitted, including packet $i$, are correct.
- If node $A$ does not receive any feedback before the timeout, it then retransmits packet $i$ and all the subsequent packets that have been transmitted. After the retransmission, node $A$ may proceed to transmit the other packets not transmitted yet.

Finally, when node $A$ transmits packet $i$ to node $B$, if the corresponding NACK is corrupted or lost, node $A$ will eventually become timeout. In this case, node $A$ retransmits packet $i$ and all the subsequent packets transmitted. After the retransmission, node $A$ may proceed to transmit the other packets not transmitted yet.

The GBN-ARQ scheme can outperform the SW-ARQ scheme by avoiding waiting for each individual packet to be confirmed. However, the retransmission of all the subsequent packets following a unsuccessfully transmitted packet may be unnecessary, because some of them may have been correctly received. For this sake, the SR-ARQ transmission scheme [9] has been proposed. Like the GBN-ARQ, the transmitter under the SR-ARQ keeps sending a number of packets specified by a window size, regardless of the feedbacks from the receiver. The difference between the SR-ARQ and the GBN-ARQ is that, only the packets confirmed by NACKs or being timeout are retransmitted. Therefore, the SR-ARQ can reduce the number of retransmissions and provide higher efficiency than the GBN-ARQ. However, the cost for the SR-ARQ is that the receiver must store the previous NACKs until the corresponding packets are correctly received. Moreover, the transmitter and receiver must have more complicated logic in order to ensure that the information delivered by the packets is in the right order.

In this paper, the throughput performance of the Butterfly network is investigated and compared in the context of the above-mentioned three types of ARQ schemes. Note that, since we have assumed that there is no packet loss and that the feedback channels are perfect without transmission errors, therefore, in this contribution, the transmitters do not need to deal with the operations for the event of timeout.

## IV. Performance Results

In this contribution, we evaluate both the maximum steady-state throughput and the (average) steady-state throughput of the Butterfly networks employing various ARQ schemes. The maximum steady-state throughput is achieved under the assumption that the source node $v_1$ always has packets to transmit. In other words, the source node does not need to wait for the packets from sources $X_1$ and $X_2$, whenever it can transmit new packets. By contrast, the steady-state throughput is obtained by assuming that the network is fed with continuous packets generated from homogeneous Poisson processes [10]. In this case, the source node $v_1$ may need to wait for the packets from sources $X_1$ and $X_2$, even it is free to transmit new packets.

In the figures shown in this section, the throughput at the sink node $v_6$ (or $v_7$) is obtained as

$$R_i = \frac{N_i}{T_i}, i = 1, 2 \tag{1}$$

where $N_i$ represents the number of packets successfully received by sink $i$ attached to the sink node $v_6$, while $T_i$ is the total time required to receive these $N_i$ packets.

In order to evaluate the throughput performance, we assume that two packet streams each with $N_{Total} = 1000$ packets are sent from the source node $v_1$ to the sink nodes $v_6$ and $v_7$. However, due to the symmetric property of the Butterfly network, we only need to evaluate the throughput in the context of the sink node $v_6$, which has two sinks, 1 and 2. We assume that the first packet stream is sent from source $X_1$ to Sink 1 and the second packet stream is sent from source $X_2$ to Sink 2. Furthermore, we assume that, in the two packet streams, the packets with the same generation number arrive at $v_1$ at the same time. In practice, this assumption corresponds to the case

that one information source of a given information rate divides the data stream into two sub-streams, each with half of the rate. In our simulations, observation starts after the first 100 packets are received from both the streams, so that the system enters the steady-state. The throughput performance is measured based on the rest 900 packets of each of the streams.

Furthermore, in our simulations, the ARQ schemes are set up as follows. For both the GBN-ARQ and SR-ARQ schemes, we assume that the transmitter is able to send out four packets during one RTT. Since for both the GBN-ARQ and SR-ARQ schemes, larger queueing buffer results in higher throughput [10], hence, for fairness of comparison, the transmission window for both the GBN-ARQ and SR-ARQ schemes is set to a length of eight packets.

The maximum steady-state throughput attained at sinks 1 and 2 of the Butterfly network as shown in Fig. 1 is shown in Fig. 2, when assuming that the source node $v_1$ always has packets to send. The steady-state throughput of the two paths of the Butterfly network is illustrated in Figs. 3-6, when assuming that the packet arrival processes at the sources are modelled by the Poisson processes with different values for the arrival rate, which is denoted by $\lambda_P$.

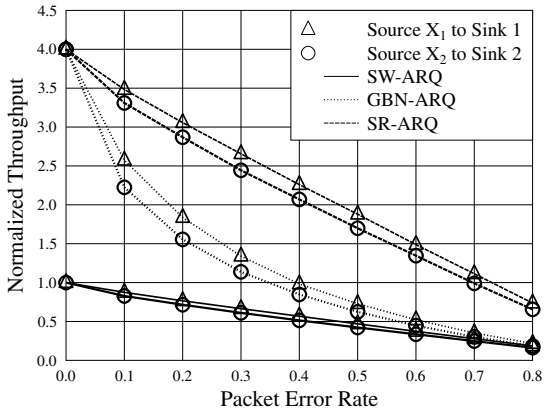


Fig. 2. Maximum throughput versus packet error rate of the Butterfly network, when it is operated in steady-state.

The maximum throughput versus PER for the Butterfly network in the context of the three types of ARQ schemes is shown in Fig. 2, where the throughput was normalized by the RTT. As it can be seen that, in an error-free situation, the system attains the maximum possible throughput that the ARQ scheme can achieve. Specifically, for both the GBN-ARQ and SR-ARQ schemes, the maximum possible throughput can reach is 4 packets per RTT. By contrast, the SW-ARQ scheme can only attain the maximum possible throughput of 1 packet per RTT, since it waits for the ACK/NACK whenever sending a packet. However, when $p_e > 0$, as shown in Fig. 2, the SR-ARQ is capable of attaining the highest throughput among the three ARQ schemes, while the SW-ARQ achieves the lowest throughput. As shown in Fig. 2, when $p_e$ is very low, the GBN-ARQ scheme is capable of achieving a similar throughput as the SR-ARQ scheme. However, as $p_e$ increases, the throughput of the GBN-ARQ converges to that achieved by SW-ARQ scheme.

As shown in Fig. 2, the attainable throughput decreases, as the PER $p_e$ increases. Furthermore, we can see that, for any ARQ scheme, the throughput from source $X_1$ to Sink 1 is higher than that from source $X_2$ to Sink 2. This is because, as seen in Fig. 1, there are only 2 hops from source $X_1$ to Sink 1, but there are 4 hops from source $X_2$ to Sink 2.

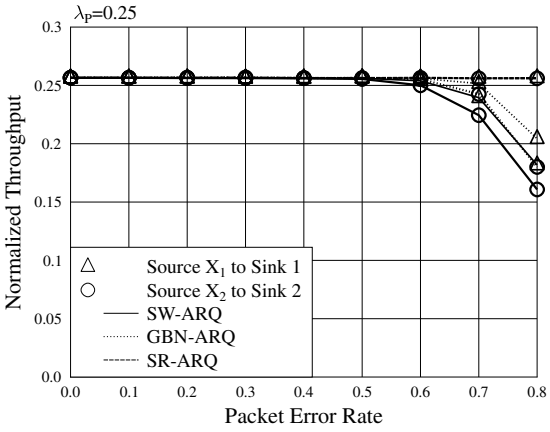


Fig. 3. Steady-state throughput versus packet error rate performance of the Butterfly network, when packets arrive at the sources following the Poisson processes with an arrival rate of $\lambda_P = 0.25$ (packets/RTT).

As the results in Fig. 3 show, when the packet arrival rate is $\lambda_P = 0.25$, all the three ARQ schemes may maintain a similar throughput for both the path from source $X_1$ to Sink 1 and the path from source $X_2$ to Sink 2, provided that the PER is sufficiently low, such as $p_e \leq 0.5$, However, the throughput of the SW-ARQ and GBN-ARQ schemes starts to decrease from the PER $p_e = 0.5$, while the throughput of the SR-ARQ scheme stays constant over the whole range of PER considered. Again, when the throughputs of the two paths are different, as shown in Fig. 3, for both the SW-ARQ and GBN-ARQ schemes, the throughput from source $X_2$ to Sink 2 is lower than that from source $X_1$ to Sink 1, the SW-ARQ attains the lowest throughput among the three schemes considered.

Fig. 4 shows the throughput versus PER performance of the two paths in the Butterfly network, when the packet arrival rate is $\lambda_P = 1$. From the results of Fig. 4, we observe that, for both the GBN-ARQ and SR-ARQ schemes, both the paths can tolerate a small PER without decreasing the throughput. By contrast, the throughput of the SW-ARQ scheme starts decreasing from the PER $p_e = 0$. The throughput of the GBN-ARQ scheme starts decreasing from $p_e \approx 0.2$ for the path from source $X_2$ to Sink 2 and from around $p_e \approx 0.3$ for the path from source $X_1$ to Sink 1. As shown in 4, the throughput of the SR-ARQ scheme starts decreasing from $p_e \approx 0.6$ for both the paths. Finally, when the throughputs of the two paths are different, the throughput of the path from source $X_1$ to Sink 1 is higher than that of the path from source $X_2$ to Sink 2, again, because the first path has less hops than the second path.

Finally, in Fig. 5 and 6 the throughput versus PER performance is shown, when the packet arrival rate is $\lambda_P = 4$ for Fig. 5 and $\lambda_P = 8$ for Fig. 6. Due to the high arrival rate,

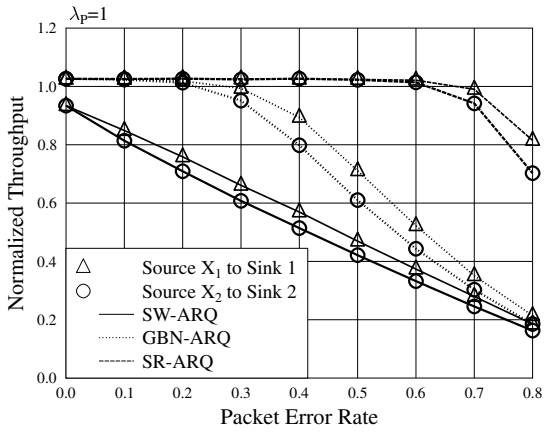


Fig. 4. Steady-state throughput versus packet error rate performance of the Butterfly network, when packets arrive at the sources following the Poisson processes with an arrival rate of $\lambda_P = 1$ (packets/RTT).

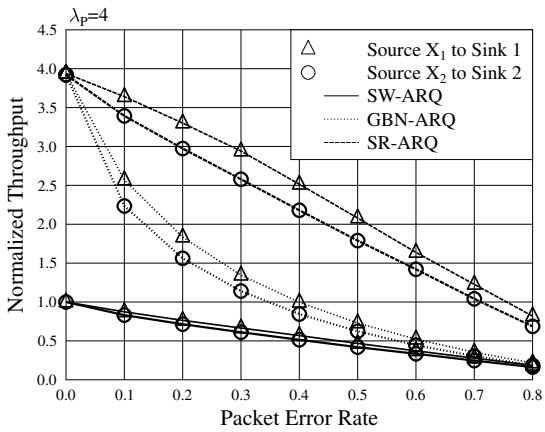


Fig. 5. Steady-state throughput versus packet error rate performance of the Butterfly network, when packets arrive at the sources following the Poisson processes with an arrival rate of $\lambda_P = 4$ (packets/RTT).

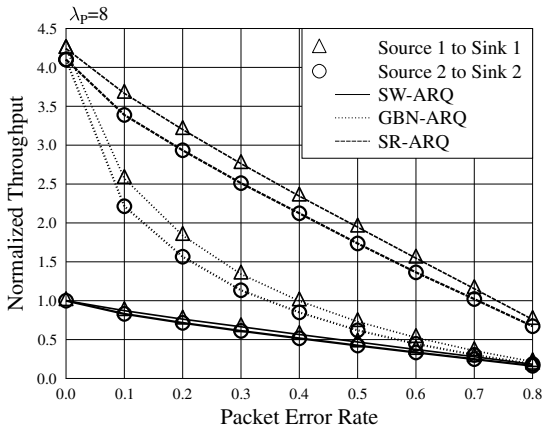


Fig. 6. Steady-state throughput versus packet error rate performance of the Butterfly network, when packets arrive at the sources following the Poisson processes with an arrival rate of $\lambda_P = 8$ (packets/RTT).

the throughput achieved in Figs. 5 and 6 is very similar to the maximum throughput shown in Fig. 2. Specifically, when $p_e = 0$, both the paths with the GBN-ARQ or SR-ARQ scheme is capable of attaining a throughput of $4$ packets per RTT. By contrast, at $p_e = 0$, the SW-ARQ scheme can only achieve its maximum throughput of $1$ packet per RTT. As shown in Figs. 5 and 6, for all the three types of ARQ schemes, the throughput decreases as the PER increases. As the PER starts increasing from $p_e = 0$, the throughput of the GBN-ARQ scheme decreases much faster than that of the SR-ARQ scheme. The throughput of the GBN-ARQ scheme finally converges to that of the SW-ARQ scheme, as the PER becomes very high.

## V. Conclusions

The maximum achievable throughput and the steady-state throughput of the Butterfly network are investigated and compared in the context of the SW-ARQ, GBN-ARQ and the SR-ARQ data transmission schemes. From our studies, we find that, at a given PER, the SW-ARQ scheme attains the lowest throughput while the SR-ARQ scheme is capable of attaining the highest throughput among the three ARQ schemes considered. The GBN-ARQ achieves the throughput close to that of the SR-ARQ, when the PER is very low. However, when the PER increases, the throughput of the GBN-ARQ converges to that of the SW-ARQ. Therefore, for the Butterfly network, the SR-ARQ scheme may be one of the appropriate error protection schemes for supporting data transmission. However, we should realize that the SR-ARQ scheme requires the highest implementation complexity, but the SW-ARQ scheme has the lowest complexity, among the three ARQ schemes considered. Additionally, for a given ARQ scheme at a given PER, we find that the throughput from source $X_1$ to Sink $1$ is usually higher than that from source $X_2$ to Sink $2$.

## References


[1] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[2] D. S. Lun, M. Medard, and M. Effros, "On coding for reliable communication over packet networks," in *Proc. 42nd Annual Allerton Conference on Communication, Control, and Computing*, Sept./Oct. 2004.

[3] S. Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[4] D. S. Lun, M. Medard, R. Koetter, and M. Effros, "Further results on coding for reliable communication over packet networks," in *Proc. International Symposium on Information Theory '05*, 4–9 Sept. 2005, pp. 1848–1852.

[5] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Upper Saddle River, New Jersey: Prentice Hall, 2004.

[6] C. Fragouli, D. Lun, M. Medard, and P. Pakzad, "On feedback for network coding," in *Proc. 41st Annual Conference on Information Sciences and Systems 2007*, 14–16 March 2007, pp. 248–252.

[7] J. Kumar Sundararajan, D. Shah, and M. Medard, "ARQ for network coding," in *Proc. IEEE International Symposium on Information Theory 2008*, 6–11 July 2008, pp. 1651–1655.

[8] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Hybrid ARQ-random network coding for wireless media streaming," in *Proc. Second International Conference on Communications and Electronics 2008*, 4–6 June 2008, pp. 115–120.

[9] W. Stallings, *Data and Computer Communications*, 8th ed. New Jersey: Prentice Hall, 2006.

[10] D. P. Bertsekas and R. G. Gallager, *Data Networks*. Upper Saddle River, New Jersey: Prentice Hall, 1992.