Queueing Performance under Network Coding

Omer H. Abdelrahman and Erol Gelenbe
Department of Electrical and Electronic Engineering
Imperial College London, UK
Email:{oha06,e.gelenbe}@imperial.ac.uk

***Abstract*—We develop analytical and simulation models to evaluate the additional delay at intermediate nodes of a store and forward packet network when Network Coding (NC) is applied. The approach is based on the analysis of queueing systems with specific service processes that capture the effect of NC. The analytical results are compared with simulations.**

I. Introduction

Network coding (NC) was initially introduced in [1], [2] where its utility for multicast networks was shown. NC allows the algebraic combination of packets at nodes of a multi-hop network, for example by a bit-by-bit XOR operation of two packets, before forwarding them towards their destination. NC can reduce the maximum bandwidth needed for specific links provided that redundant data is sent over alternate paths so that destination nodes may reconstruct the original packet flows. Thus NC can reduce traffic rates on links while distributing traffic on a larger number of paths. It also offers a simple form of encryption for the data that is being transmitted since none of the streams taken singly can be completely decoded by themselves. Although NC can improve network throughput, the performance of NC from a queueing and delay perspective has not received much attention [3]–[6]. Thus in Sections II and III, we present queueing models for a single encoding node with synchronous and asynchronous encoding, and compare the results with simulations.

II. Queueing Model for Synchronous NC (SNC)

Consider a network node that receives packets from a set of distinct flows or connections. The simplest and also most naive approach to NC would require that each packet in each flow be encoded with a packet of *each of the other flows* that pass through that encoding node, with the encoding being carried in sequence for each flow. We can consider that the encoding process acts as a server which waits for the arrival of one packet from each flow before it can start encoding, and is idle when its input buffer does not contain at least one packet from each class. Related models were studied [7] for manufacturing systems, and it was shown that this queueing system is intrinsically unstable so that the queueing delay per flow tends to infinity when the input buffer is unlimited. In other words the waiting time process cannot converge in distribution to a non-defective limit. Most subsequent work on assembly processes has avoided the instability problem by assuming limited capacity buffers [8]–[10] or by controlling the input flow of items [11]. Exact analysis of a finite capacity assembly station with two input streams is presented in [10], where the author concluded that the model becomes intractable as the number of flows increases. The taxicab problem [12] where taxis and customers can only leave the system together is also closely related to this scheme. Because the exact analysis of NC nodes appears to be very difficult, in this section we propose solutions based on the "decoupling approximations" along the lines of [13].

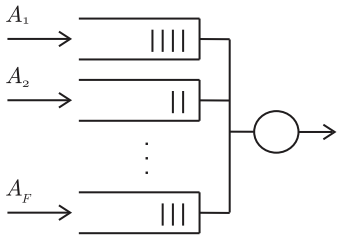

Fig. 1. A schematic representation of an encoding node with F input traffic streams with general inter-arrival times $A_1, \ldots, A_F$

Consider a node which receives F distinct independent flows of packets with general inter-arrival time distribution $A_i(x) = Prob[A_i \leq x]$ for the $i-th$ flow, which queue up in distinct buffers of finite capacity B_i for $i = 1, \ldots, F$. Assume that packet lengths in each stream are independent random variables, and that they are mutually independent between flows, with general distribution $S(x) = Prob[S \leq x]$, where S is the random variable representing packet length. We assume that the node transmission time is directly proportional to packet length with a constant of proportionality of 1.

The node encodes all the flows together, packet by packet and in sequence, and then forwards one encoded packet for each F packets of the distinct flows. We assume that if all the input buffers contain at least one packet, the server will pack the shorter packets with $zero$-bits to reach the length of the longest packet, and encode the resulting packet bit by bit, so that its length will be equal to the largest of the F packet lengths. The transmission time of a packet is then assumed to be proportional to the length of the largest packet. We assume that during the synchronization phase, the next packet to be encoded from each flow waits at the server until all required packets are available. This ensures that the capacity of each queue is not reduced by 1.

With these assumptions, the queue length process seen by the $i-th$ individual flow includes its own arrival process, and a service time which is proportional to the length of the

978-1-4244-4536-3/09/$25.00 ©2009 IEEE

largest of the F co-encoded packets plus the largest of the residual inter-arrival times for any of the flows which may be empty when the service begins. Thus if any of the other buffers are empty, the resulting service time will include the residual inter-arrival time until the missing packets arrive, followed by the encoding and transmission time.

The decoupling approximation we propose is based on the following heuristic assumption. Let p_{ji} be the probability that in steady state the $j-th$ queue is *empty* when a service begins at the designated $i-th$ queue. We will assume that the steady-state probability that any subset $Z \subseteq Z(i) = \{1, \ ... \ i-1, i+1, \ ... \ F\}$ is empty when a service begins at i is $\prod_{j\in Z} p_{ji} \prod_{j\notin Z}[1-p_{ji}]$ and therefore that the total equivalent service time S_i observed by the $i-th$ queue is the residual time for packets to arrive to *all* currently empty queues followed by the maximum of the transmission times for *all* of the packets:

$$S_i(x) \equiv Prob[S_i \leq x] = \tag{1}$$
$$\sum_{Z\subseteq Z(i)} \prod_{j\in Z} p_{ji} \prod_{j\notin Z}[1-p_{ji}] \int_0^x G_Z(x-y) F S(y)^{F-1} dS(y)$$

where

$$G_Z(y) = \prod_{j\in Z} \hat{A}_j(y) \tag{2}$$

and $\hat{A}_j$ is the forward recurrence time of the inter-arrival time to queue j, which has a distribution:

$$\hat{A}_j(y) = \frac{1}{E[A_j]} \int_0^y [1 - A_j(z)] dz \tag{3}$$

To further simplify matters, we will also assume that p_{ji} does not depend on i, and that we can obtain p_j from the probability that queue j is empty using the solution of a $GI/GI/1/B_j$ queue with service time distribution $S_j(x)$ and inter-arrival time distribution $A_j(x)$. We will base our computations for the $GI/GI/1/B_j$ queue on the diffusion approximation for a finite buffer using the results in [14]. The numerical approach then consists in solving the coupled equations (1) with the appropriate expression for p_j.

For the special case $B_i = 1, \ \forall i$ an exact expression for the distribution of the waiting time for the $i-th$ flow is given by:

$$W_i(x) = \int_0^\infty G_{Z(i)}(y+x) g_i(y) \ dy, \quad x \geq 0 \tag{4}$$

which has a probability mass at 0 when the residual inter-arrival time to the $i-th$ queue is greater than the residual time for packets to arrive to all other queues. In the numerical results, however, we do not consider this case since the total buffering capacity of a typical node would be larger than F.

The solution of this system can be summarized as follows:

Step 1 Assuming that the quantities p_i are known, find the equivalent service time distribution for each queue i as a function of p_j, $\forall j \neq i$ using (1).

Step 2 For each subsystem i: determine the steady state queue length distribution $\pi_i(k)$ (exactly or approximately) as a function of p_j.

Step 3 Solve the system of non-linear equations $\pi_i(0) = p_i$, $i = 1, \ldots, F$.

Step 4 For unbalanced node traffic, suppose $\lambda_i \leq \lambda_j, \ \forall j$. If $|\lambda_j(1-\pi_j(B_j)) - \lambda_i(1-\pi_i(B_i))| > \varepsilon$, then:

$$\hat{\pi}_j(k) = \begin{cases} 1 - \dfrac{\lambda_i}{\lambda_j}(1-\pi_i(B_i)) & k = B_j \\ \dfrac{1-\hat{\pi}_j(B_j)}{1-\pi_j(B_j)} \ \pi_j(k) & k < B_j \end{cases}$$

where ε is a small number.

To investigate the quality of the proposed approximation we have compared the results for delay and packet loss probability from the model and the discrete event simulation. Buffers are all assumed to have the same capacity and the traffic load is varied from very light to saturation. Fig. 2 shows numerical results for 2 traffic streams with Erlang-2 inter-arrival times distribution and Erlang-3 packet lengths. In general, we find that the model is relatively insensitive to the number of encoded flows, the packet length distribution and the capacity of the input buffers for the range of values tested $(5-20)$ since the model becomes unstable for very large capacities.

Having obtained an approximate solution technique that provides reasonably good accuracy, we can compare the packet delay and loss of SNC with a peer non-coding scheme under similar conditions. The latter uses a technique known as *store-and-forward* in which packets received from all traffic streams are merged into a single queue and transmitted in a first-come-first-served order. For fair comparisons, we provide the same per node total buffering capacity, total incoming traffic rate and packet length distribution.

From Fig. 2, it is clear that SNC results in a much larger delay and packet loss due to the need to wait for the arrival of at least one packet from each distinct flow before forming an encoded packet. This result is only to be expected, since in SNC the slowest traffic stream dominates the waiting time process. However, interestingly enough, SNC offers better performance when the input queues are saturated. Overall, we can conclude that SNC is not suitable for practical implementation unless it is used in a fully synchronized or highly saturated network. This leads us to the idea that network nodes could use SNC adaptively whenever the input traffic rates saturate the network.

Next we evaluate the bandwidth efficiency of SNC against that of the traditional technique. We define as a figure of merit the ratio of the output rate from the store-and-forward node to the output rate from the synchronized node. This is evaluated against the total traffic load normalized to the average service rate of un-encoded packets, as depicted in Fig. 3. In the figure, we observe that the ratio of the output rates is always greater than 1 which indicates that SNC reduces the amount of traffic coming out from the node. However, one should note that the bandwidth savings provided by the scheme do not always imply better performance. In fact, when the traffic rates do not saturate the store-and-forward node, the average number of packets served per time unit with SNC is less than that served without coding, because of the large synchronization

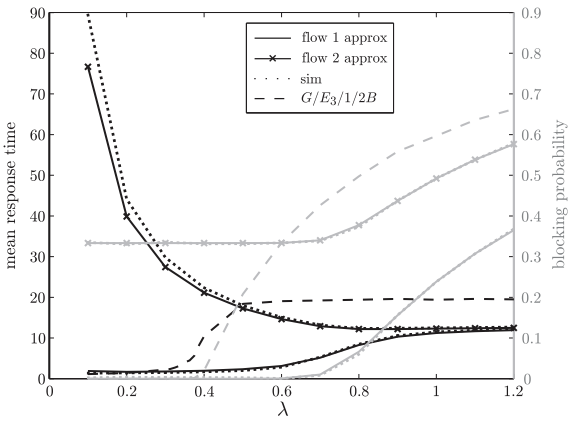

Fig. 2. Mean response time (dark) and blocking probability (light) for SNC with parameters: $F = 2$, $B_i = 10$, Erlang-2 inter-arrival times distribution with rates $\underline{\lambda} = \lambda[1 \quad 1.5]$ and Erlang-3 packet length distribution with mean 1. These are compared with results for a $G/E3/1/2B$ system where G is a mixture of two Erlang-2 arrival processes of rates $\underline{\lambda}$.

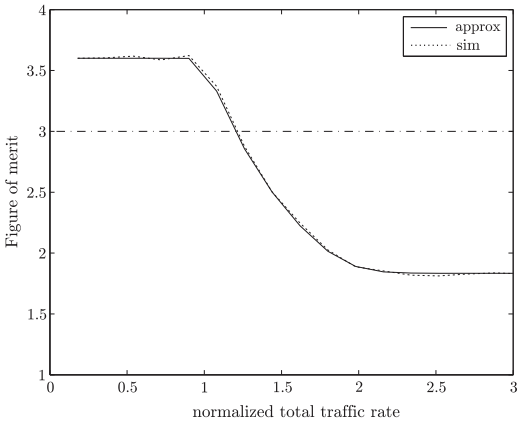

Fig. 3. Figure of merit for SNC with parameters: $F = 3$, $B_i = 20$, Poisson arrivals of rates $\underline{\lambda} = \lambda[1 \quad 1.2 \quad 1.4]$ and exponential packet length distribution with mean 1. In the area below the horizontal line, SNC node serves more packets per time unit for less bandwidth, whereas the bandwidth savings above the horizontal line are due to the large synchronization delays.

delays. On the other hand, the synchronized node serves more packets per time unit and uses less bandwidth to do so when the node is saturated. The decay in the figure of merit graph suggests that the equivalent service rate of the synchronized node increases as the traffic rates increase. Finally, we remark that the effect of packet size distribution on the performance of SNC is negligible when the node is light to moderately loaded.

III. Asynchronous Partial NC (APNC)

An asynchronous partial NC (APNC) scheme opportunistically encodes packets from *distinct* flows with any packets present, except for packets belonging to the same flow. Thus after the encoding node forwards a packet, the next packet forwarded will simply be the encoded version of the packets from distinct flows that are present in the queues. If only one of the flows has a packet present, then just that one unencoded packet will be forwarded, while if more than one flow has at least one packet present in their queues, then the encoded packet will include the head-of-the-line packet from each of those flows. Again, we study a queue, say the $j-th$, in isolation from the others and consider that the queues interact with each other via the steady-state probabilities, which in this case we call q_j for the $j-th$ queue, that the $j-th$ server does not participate in encoding a packet. In this case this may occur either because that queue is idle (no customers), or because the queue is busy but the server is idle because there is another currently ongoing encoding and transmission which started while the $j-th$ queue was empty.

To analyze the system we have just described, we propose an approximation based on constructing an "equivalent" $G/G/1$ model for any of the F individual queues, having the "server with vacations" property [15], [16]. Note that in this case the assumption of finite buffers is not needed because (contrary to the SNC case) the system with unlimited buffer size is not always unstable. The server with vacation is a queueing system in which, after each service ends, if the queue is empty then the server will "go off" for a vacation time V, and the process repeats itself if the server finds the queue empty at the end of the vacation time. Service starts again when, at the end of a vacation time, there is at least one customer in queue. Such models usually assume that each successive vacation time is an independent and identically distributed random variable. The key result about a queue with vacations was obtained nearly thirty years ago:

Result (Gelenbe-Iasnogorodski 1980 [16]) Let W be the random variable representing the steady-state distribution of the customer waiting time for a queue (with unlimited buffer size) and a server with vacations, for a single server queue with service time S and vacation time V, which are assumed to be mutually independent random variables, while both the successive service times and the successive vacation times are sequences of independent and identically distributed random variables. We assume that the inter-arrival times of customers to the queue are i.i.d. (but the arrival process need not be Poisson). Let U be the waiting time for exactly the same queue, but *without* vacations (i.e. when $V = 0$). Then the following equality holds in distribution:

$$W = U + \hat{V} \tag{5}$$

where $\hat{V}$ is a random variable whose distribution is given by:

$$\hat{V}(x) = \frac{1}{E[V]} \int_0^x [1 - V(y)]\, dy \tag{6}$$

Thus the result summarized in (5) allows us to map all properties of interest of a queue with vacations in steady state to those of a system without vacations using the probability distribution of the vacation time V. In particular we can see that only U depends on the arrival process, and therefore the stability condition for the queue with vacations is *not* affected by the vacation time distribution, and is identical to the stability conditions for the corresponding *ordinary* queue.

Now turning back to the model for APNC, note that if at the end of a service time the $j-th$ queue is *not empty*, then the subsequent service time distribution $S_j(x)$ will be the maximum of the service times for the set of non-empty queues including the $j-th$ queue. Thus, for $Z(j) = \{1,\ ...\ j-1, j+1,\ ...\ F\}$, the resulting approximation for the service time distribution after a departure that *does not* leave an empty queue at j will be:

$$S_j(x) = S(x) \sum_{Z \subseteq Z(j)} S(x)^{|Z|} \prod_{i \in Z} [1-q_i] \prod_{i \notin Z} q_i \qquad (7)$$

On the other hand if the $j-th$ queue is empty after a service time ends, then when the next arrival to that queue occurs, the arriving packet will have to wait for its service until after any currently ongoing service involving the *other* queues ends.

Thus after the $j-th$ queue becomes empty at the end of a service time, a sequence of vacation times involving the services at *other* queues will take place, and some of these other services will possibly be of zero duration if some of the other queues are empty. These vacation times have a probability distribution similar to $S_j(x)$, *except* that the $j-th$ queue is not involved. We will denote the vacation time distribution for the $j-th$ queue by $V_j(x)$, where:

$$V_j(x) = \sum_{Z \subseteq Z(j)} S(x)^{|Z|} \prod_{i \in Z} [1-q_i] \prod_{i \notin Z} q_i \qquad (8)$$

We can now use the formula for the probability q_j that the $j-th$ queue is empty or that it is busy but that the server is idle due to a vacation time, from the corresponding result for the model with vacations, where $S_j(x)$ and $V_j(x)$ are, respectively, the service time and vacation time distributions.

The QoS parameters of interest for APNC are the mean response time and bandwidth efficiency which are tested for 2, 3 and 4 traffic streams and for various packet length and inter-arrival times distributions. When the arrival processes are Poisson, the mean response times for the equivalent $M/G/1$ subsystems (without vacations) are obtained exactly using the well known *Pollaczek-Khinchin* formula [15]. For general arrival processes, we use the diffusion approximation in [14] to solve the equivalent $G/G/1$ queues, which is likely to introduce additional errors in computations.

Figure 4 shows the mean response times for nodes with balanced Poisson flows. One may notice that the approximation yields very accurate results, which in most instances remain within 2% of the simulation results. Numerical results involving general arrival processes are presented in Fig. 5. The figure shows that the error between the analytical and simulation results is less than 8%. Overall the quality of the approximation appears to be relatively insensitive to the number of encoded flows, the traffic load and the packet length distribution.

Next we evaluate the accuracy of the model in predicting the output rate from the encoding node. Since obtaining an exact expression for the throughput seems to be very difficult, we propose to use the following approximation:

$$\gamma_1 = \sum_{n=1}^{F} \sum_{Z \subseteq \{1,\ldots,F\}:|Z|=n} \prod_{j \in Z} [1-q_j] \prod_{j \notin Z} q_j \left(\int_0^\infty x\ nS(x)^{n-1} dS(x) \right)^{-1} \qquad (9)$$

Alternatively the throughput can be obtained from an approximate expression for the mean inter-departure time as follows:

$$\gamma_2^{-1} = \prod_{j=1}^{F} q_j \left(E[S] + \int_0^\infty x\ dH(x) \right) + \sum_{n=1}^{F} \sum_{\substack{Z \subseteq \{1,\ldots,F\} \\ |Z|=n}} \prod_{j \in Z} [1-q_j] \prod_{j \notin Z} q_j \int_0^\infty x\ nS(x)^{n-1} dS(x) \qquad (10)$$

where

$$H(x) = (1 - \prod_{j=1}^{F} [1 - \hat{A}_j(x)]) \qquad (11)$$

In general, (9) tends to slightly underestimate the throughput whereas (10) overestimates it. The error between the analytical and simulation results is less than 5% in most of the cases tested.

The numerical results demonstrate that APNC outperforms the traditional store-and-forward technique regardless of traffic conditions. The performance gap between the two schemes becomes wider when the arrival rates of the distinct flows are balanced, and when the number of flows increases. In Fig. 4, the coding advantage expressed as the ratio of the maximum stable traffic rate using NC to that without NC is roughly 1.5, 2 and 2.5 for 2, 3 and 4 traffic streams, respectively. Fig. 5 shows that when the system is lightly loaded, the incoming flows experience similar average packet delays although they arrive at different rates. This is likely due to the fact that in such instances, packets from slower traffic streams experience server vacations more frequently thus increasing their mean response times.

Our results indicate that APNC is effective in reducing the bandwidth utilization by up to 35% when the node is heavily loaded. The bandwidth savings are more pronounced when the arrival rates of the distinct flows are balanced, since more coding opportunities arise in such instances. This is illustrated in Fig. 6 where we plot the percentage of saved bandwidth against the normalized total traffic rate for a node receiving 2 Poisson flows with exponential packet lengths. We can see that the scheme achieves the maximum bandwidth reductions when the arrival rates are equal. In the figure, we also investigate the effect of increasing the number of balanced flows on the bandwidth utilization performance. As one would expect, APNC offers more bandwidth savings as the number of flows increases.

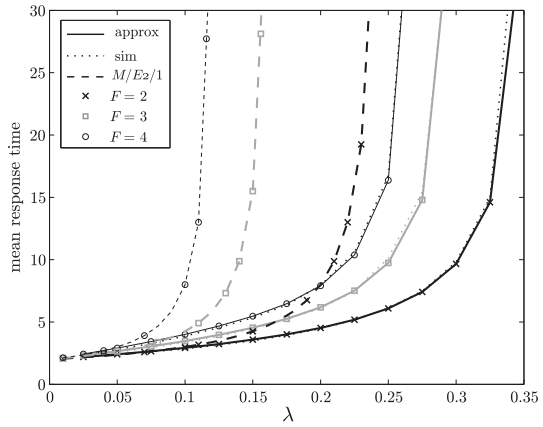

Fig. 4. Mean response times for APNC with $F = 2,\ 3,\ 4$, Poisson arrivals of rates $\lambda_i = \lambda$ and Erlang-2 packet length distribution with mean 0.5, and results for $M/E2/1$ systems with arrival rates $F\lambda$.

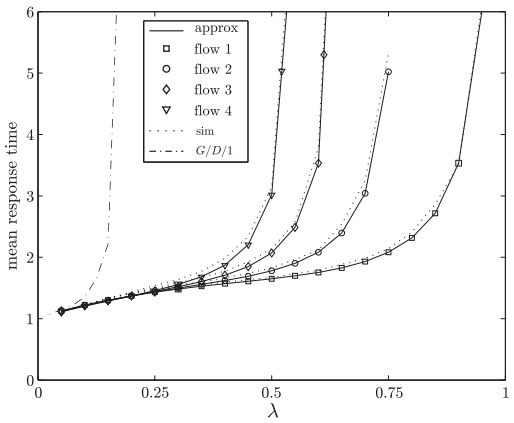

Fig. 5. Mean response time for APNC with $F = 4$, Erlang-2 inter-arrival times distribution with rates $\underline{\lambda} = \lambda[1 \quad 1.25 \quad 1.5 \quad 1.75]$ and constant packet lengths 1, and results for a $G/D/1$ system where G is a mixture of four Erlang-2 arrival processes with rates $\underline{\lambda}$.

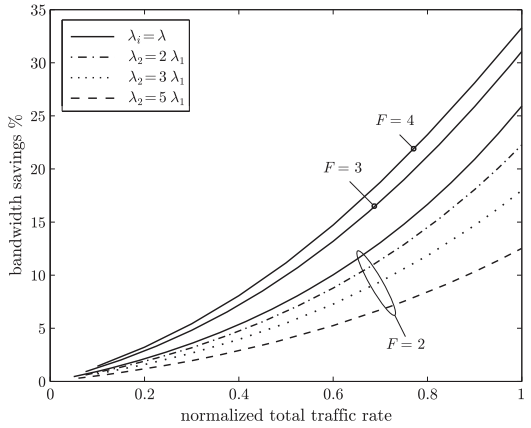

Fig. 6. Analytical results for the percentage of saved bandwidth vs the normalized total traffic rate for APNC with Poisson flows and exponential packet length distribution with mean 1. The scheme offers more bandwidth reductions as the number of flows F increases, especially when the flows are balanced.

IV. Conclusions

In this paper, we have evaluated the queueing performance at intermediate nodes of a store and forward packet network under NC, based on models of coupled queues which are solved approximately. The forms of NC considered are synchronous coding and asynchronous partial coding. The analysis was validated via simulation studies. We have shown that NC can improve performance significantly provided that it is used opportunistically unless implemented in a fully synchronized network. In future work we will consider NC schemes which restrict coding within disjoint subsets of the packet flows. The work presented in this paper will also be extended to incorporate these models in a network setting where performance metrics such as the end-to-end delay including the packet assembly and decoding at the output are also considered.

Acknowledgment

This research was supported by the EU FP6 CASCADAS (IST-027807) and FP7 DIESIS Projects of the EU Future and Emerging Technologies Program.

References

[1] R. Yeung and Z. Zhang, "Distributed source coding for satellite communications," *Information Theory, IEEE Transactions on*, vol. 45, no. 4, pp. 1111–1120, May 1999.

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

[3] D. Lun, P. Pakzad, C. Fragouli, M. Medard, and R. Koetter, "An analysis of finite-memory random linear coding on packet streams," *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2006 4th International Symposium on*, pp. 1–6, 03-06 April 2006.

[4] B. Shrader and A. Ephremides, "A queueing model for random linear coding," *Military Communications Conference, 2007. MILCOM 2007. IEEE*, pp. 1–7, 29-31 Oct. 2007.

[5] Y. Ma, W. Li, P. Fan, and X. Liu, "Queuing model and delay analysis on network coding," *Communications and Information Technology, 2005. ISCIT 2005. IEEE International Symposium on*, vol. 1, pp. 112–115, 12-14 Oct. 2005.

[6] X. He and A. Yener, "On the energy-delay trade-off of a two-way relay network," *Information Sciences and Systems, 2008. CISS 2008. 42nd Annual Conference on*, pp. 865–870, March 2008.

[7] J. M. Harrison, "Assembly-like queues," *Journal of Applied Probability*, vol. 10, no. 2, pp. 354–367, Jun 1973.

[8] E. H. Lipper and B. Sengupta, "Assembly-like queues with finite capacity: bounds, asymptotics and approximations," *Queueing Syst. Theory Appl.*, vol. 1, no. 1, pp. 67–83, 1986.

[9] W. J. Hopp and J. T. Simon, "Bounds and heuristics for assembly-like queues," *Queueing Systems*, vol. 4, no. 2, pp. 137–155, June 1989.

[10] U. N. Bhat, "Finite capacity assembly like queues," *Queueing Syst. Theory Appl.*, vol. 1, no. 1, pp. 85–101, 1986.

[11] F. Bonomi, "An approximate analysis for a class of assembly-like queues," *Queueing Syst. Theory Appl.*, vol. 1, no. 3, pp. 289–309, 1987.

[12] D. G. Kendall, "Some problems in the theory of queues," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 13, no. 2, pp. 151–185, 1951.

[13] E. Gelenbe and A. Ghanwani, "Approximate analysis of coupled queueing in atm networks," *Communications Letters, IEEE*, vol. 3, no. 2, pp. 31–33, Feb 1999.

[14] E. Gelenbe, "On approximate computer system models," *J. ACM*, vol. 22, no. 2, pp. 261–269, April 1975.

[15] E. Gelenbe and I. Mitrani, *Analysis and synthesis of computer systems*. Academic Press, London, 1980.

[16] E. Gelenbe and R. Iasnogorodski, "A queue with server of walking type (autonomous service)," *Annales de l'institut Henri Poincar (B) Probabilits et Statistiques*, vol. 16, no. 1, pp. 63–73, 1980.