POLLING, GREEDY AND HORIZON SERVERS ON A CIRCLE

ARIE HAREL

Rutgers University, Newark, New Jersey

ALAN STULMAN

St. John's University, New York, New York

(Received December 1990; revisions received October 1991, June 1992, May 1993; accepted August 1993)

Service in a loop-based polling system consists of a single server moving around a closed tour, stopping to perform services wherever requests are encountered. There are N stations (unit buffer queues) spaced one unit of distance apart, and the server moves at a unit speed. All queues are identical, and the service time is deterministic. We compare the two well known cyclic polling and greedy servers with a new control policy called the horizon server. The cyclic polling server moves in one direction, even if no requests are waiting, and stops whenever it encounters a request. The greedy server selects the nearest request for its next service. At any station the greedy server can reverse its direction if a new request arrives nearby, and if no requests are waiting the greedy server does not move. The horizon server, with parameter d, ignores all requests for service from a distance farther than d. Within its horizon ($\leq d$) it acts like the greedy server. Analytical solutions for $N = 2$ and 3 and numerical results for $N \leq 6$ show that the horizon server, with the optimum value of d, outperforms the polling and the greedy servers.

Polling systems play a crucial role in controlling access to communication services in telecommunication systems. Polling policies have an impact on response time, throughput and resource allocation in transmission and switching systems. A polling system consists of multiple stations which are served by a single server. In a cyclic polling scheme the server attends these queues by cyclically moving among them and stopping for service wherever necessary. In the original polling model, the case of queues with a single buffer was used for modeling and analyzing the system by Mack, Murphy and Webb (1957). The server visits N stations cyclically around a closed tour, servicing the stations when needed.

The greedy polling server operates as follows: At any station it selects the nearest request for its next service, even if it has to reverse direction. If no requests are waiting, the server remains idling where it served the last request. Note, that in contrast, the cyclic polling server moves in one direction even if no requests are waiting, and stops only whenever it encounters a request.

Applications of polling schemes arise in many telecommunication systems. For example, in local area communication networks the server performs a switching function by choosing which peripheral station gets access to a central facility. In many cases, as demonstrated in the following examples, knowledge of the location of the service requests can be used by the polling algorithm to enhance system performance. In that respect the model can be used to assess the value of collecting and using queue service request information to enhance system performance. (See Gavish 1973 for a discussion of the value of information in queueing systems.)

Consider the following telecommunication facility. A central station uses a directional rotating microwave antenna to transmit and collect information from other stations in the area around it. The antenna is directional and thus at any given time it can transmit effectively only to a narrow cone of (say) v degrees. Communication with a given station is achieved by rotating the antenna, directing it toward the desired station, and transmitting to it (receiving from it) the data. An obvious performance objective in such a system is to maximize the transmission throughput. Since the movement of the antenna is relatively slow compared to the available bandwidth, the service algorithm, used for determining which of the communications requests is to be served next, can be utilized to achieve a higher performance. A similar problem exists in the multibeam antenna in low earth orbit satellite systems in which a single satellite has to serve multiple cells (20–50) on the ground. The antenna has to be switched dynamically between the different cells, based on their traffic intensity, signal strength, satellite trajectory, and revenue stream.

A related application is a communication system in which a central station communicates to peripheral stations by means of laser or infrared transmission; such communication is fully digital and highly directional, requiring a line of sight between the communicating stations, and thus leads to a directional architecture as in the microwave antenna. For distance communication, microwave transmission is widely used as an alternative to cable. On a college campus or industrial complex it is

Subject classification: Queues, cyclic: dynamic scheduling of a server in a polling system.
Area of review: TELECOMMUNICATIONS (SPECIAL ISSUE ON TELECOMMUNICATION SYSTEMS: MODELING, ANALYSIS AND DESIGN).

0030-364X/95/4301-0177 $01.25

inexpensive and easy to put a laser transmitter and receiver on the roof of each building (see, for example, Tanenbaum 1988 p. 65).

The results presented in this paper could be applied to tree-based local access networks (see Gavish 1982 and 1991) in which a central node controls access to a central service facility which is located at the head (root) of the tree. The user nodes are connected to the central node by multidrop lines, with the user nodes thereby sharing the communication links to the central service facility. Access to the communication links is controlled by a multidrop protocol which polls the user nodes in a cyclical order and grants permission to use the shared communication links based on some control policy. Several access control policies have been suggested in the literature starting from a cyclical loop policy where each station is polled in a fixed order to broadcast collision detection and recovery policies. A horizon server polling policy could improve the performance obtained by the earlier policies.

Polling systems have important applications in ring-based local area networks. They play a crucial role in token passing protocols in very high capacity fiber and token ring local area networks. The literature is replete with studies involving SONET (synchronous optical network) systems. One such fiber optics based system involves a dual loop token passing ring. Each node of the token ring is attached to two distinct loops, one of which is transmitting clockwise and the other is transmitting counterclockwise. Such a configuration is referred to as fiber distributed data interface (FDDI). A discussion of such a network can be found in Tanenbaum. He defines a class A station as one which is attached to both rings and thus acts as a bridge between them. Such networks were also discussed by other authors. See, for example, Sosnosky and Wu (1991) and the references therein.

The access of the channel on FDDI is controlled by a token (in some versions multiple tokens; see, e.g., Hammond and O'Reilly 1986). According to this mechanism a station does not transmit until it acquires a free token. It then modifies the token and attaches its data to it. The message is sent around the ring and passed on by each station until the original station receives its own transmission and removes the message from the ring. It then creates and transmits a new free token.

A single token FDDI can be modeled as a polling system with a single server: The free token is modeled by the server. The time of transmitting a free token from one station to the next is made up of two components (propagation delay and station latency) and can be relatively significant. This time of token transmission is modeled by a switchover time in the polling system. The dual loop nature of the system allows the token to be transmitted in both directions (clockwise and counterclockwise) of the loop. In addition, SONET's "embedded overhead channels (EOCs)" (see Davidson and Muller 1992, p. 295) provide the ability to transmit information about the state of the system to each station. The combination of these two properties allows one to adopt a dynamic policy which determines the direction of token movement as a function of the system state. Thus, dynamic policies like the one studied in this paper can be utilized in such a configuration in order to reduce the delay experienced by the messages (in comparison to the simple static policy in which the token moves cyclically in a fixed direction).

Star-based local area networks with an active hub are another application area for the results presented in this paper. In active hub-based star networks such as STARLAN or LANSTAR, the hub is the center of the star and each user station is connected to it by a dedicated line. The hub scans the lines using a polling policy and serves each station in turn, transferring messages between stations. In LANSTAR the active hub consists of a high capacity (40 to 100 megabits per second) central loop, which has a very short diameter (a few feet). User stations are connected to the central loop by dedicated lines, each with a capacity of 2.3 megabits per second. Access to the loop is regulated by an appropriate polling control policy.

A number of authors have dealt with a series of related problems. An excellent review of the literature on the subject of polling systems is provided by Takagi (1986, 1988 and 1990). Bundy and Mack (1973) describe the problem of the server operating on a line. Coffman and Hofri (1982) analyze a scanning (or polling) model on the interval, and show how to find the performance measures by reducing the problem to a linear system. Abdel-Malek and Li (1990) introduce models to determine the sequence that minimizes the execution time of a robotic travel scheme.

In an update of the literature on the subject of polling models, Takagi (1990) reviewed over 400 articles on the subject, mostly from the last few years, with most of the applications in the area of telecommunication. Application of polling models to computers and telecommunications is the subject of several survey articles and chapters in books. Takagi (1991) reviewed applications of polling models to the following communication networks: the half-duplex transmission, polling data link control, explicit and implicit token passing protocols, and more. Levy and Sidi (1990), in their review, described many important applications to communications. Their basic model is token passing systems in local area networks. Extensions of the basic model include the modeling of acknowledgments and distributed algorithms in token ring networks, fixed and random polling order. Grillo (1990) described polling system models that have been successfully employed in communication networks. Several books devote a full chapter each for polling models and their applications in communication systems (see, for example, Hayes 1984, Hammond and O'Reilly 1986, Takagi 1986, and Schwartz 1987.

Most studies of scheduling rules for the server have been done by simulations, e.g., Teorey and Pinkerton (1972), Hofri (1980), and Geist and Daniel (1983). Geist and Daniel propose a modified version of the greedy server in which the reversal in direction is limited.

The Model

In this paper, we assume that N identical stations are evenly spaced around a closed tour. Without loss of generality we take the positive switchover time (the time needed to switch from one station to the next) to be the unit time. We assume that after a transmission the time until the next request from the same station is exponentially distributed with parameter lambda, and the transmission time is a constant. Thus, we assume a Poisson (lambda) arrival at each station (queue), and each queue is a pure loss system with a single buffer. An occupied buffer is available for a new arrival only upon the completion of the occupier's service request. The productivity of this system is measured by the throughput of the system, the mean number of stations served per time unit.

We examine the question of scheduling the movement of the server by assuming that at each station the server has full information about the state of all stations. Using this information the server can reverse his direction or can stop and wait. Maximizing the throughput can be formulated as a Markov decision process where the optimal solution for a small size problem can then be found. Unfortunately, there are 2^N states in this system, and the optimal policy does not have a simple structure. Thus, an on-line implementation of this solution requires very fast look-up tables at any station, which is not practical in most cases. Therefore, our objective is to examine a suboptimal policy which is simple enough to be practical, yet still out-performs the polling and the greedy servers.

We name our decision rule the *horizon server* with parameter d (for distance). The horizon server selects the nearest request for its next service, provided the request is within its horizon. If the location of the nearest request for service is more than d stations apart, the horizon server just ignores it. The motion of the horizon server is dynamic in the sense that its destination can change at any station if a new request arrives nearby. The horizon server with $d = 0$ is a server dedicated to one station only. Whenever $d \geq 1$ the horizon server will serve all stations in the long run. If $d \geq N/2$ the horizon server operates like the greedy server.

To get some insight into why the horizon server should work well, consider the case $\lambda > 1$. Here the optimal value of d is zero because the switchover time is one and the mean waiting time of a dedicated server is less than 1. A dedicated server alternates between service and idle state. The unit buffer assumption implies that at the end of each service cycle there must be an idle cycle. Maximizing the throughput is equivalent to minimizing the mean idle time, and ignoring requests which are too far apart seems to coincide with this objective. The above argument does not hold if the stations are not identical, or if the buffer is larger than one unit.

For $N = 2$ and $N = 3$ we obtain explicit formulas for the throughput of the horizon and greedy servers. For $N \geq 4$ we cannot get a closed-form solution, and to get the throughput we solve a set of $O(2^N)$ linear equations. For small N (i.e., $N \leq 6$) we provide enough numerical evidence to indicate that the horizon server, with the optimal value of d, outperforms the polling and greedy servers.

For λ sufficiently close to 1 we show, in Corollary 1, that the horizon server with $d = 1$ outperforms the polling server for all N. We also approximate the optimal value of d for the horizon server.

Our analysis is the first to provide an exact measure for the greedy server. For a small number of stations our scheduling rule demonstrates that the additional location information, when used properly, can improve the effectiveness of the system.

In Sections 2 and 3, we investigate the cases $N = 2$ and $N = 3$ and in Section 4 we analyze the general case. In Section 5, we develop an approximation for the optimal value of d. In Section 6, we provide some simulation results to validate the approximation. In Section 7, we conclude with a few final remarks.

1. NOTATION AND PREVIOUS RESULTS

There are N identical stations. They are spaced one unit of distance apart, on a closed tour, and the server moves at a unit speed. Each station (queue) is a pure loss system of a single buffer that has Poisson arrivals with parameter λ. The duration of the deterministic service time is S.

The throughput of the polling server is given by:

$$P = \left[S + \frac{1 + \sum_{n=1}^{N} \binom{N}{n} \prod_{j=0}^{n-1} [e^{\lambda(N+jS)} - 1]}{\sum_{n=0}^{N-1} \binom{N}{n} \prod_{j=0}^{n} [e^{\lambda(N+jS)} - 1]} \right]^{-1}. \quad (1)$$

cf., Mack, Murphy and Webb; see also, Takagi (1988), Section 2.

Let $H(d)$ denote the throughput of the horizon server with parameter d. As mentioned, the horizon server with $d = 0$ is a server dedicated to one station only. This station is indeed the $M/G/1/1$ queue with mean service time S. Thus, from the Erlang loss formula, with one server, it is easy to verify that for all N

$$H(0) = \lambda/(1 + \lambda S). \quad (2)$$

We assume that at each station the server has full information about the state of the system. The state of the system can be described by the vector

$$Y = (y_1, y_2, \cdots, y_N), \quad (3)$$

where y_i is 1 if buffer i is full, or 0 if buffer i is empty. The index i is the relative position of a station with respect to the server. That is, we define the station where the server is currently located to be number 1. The stations to the left (clockwise) are assigned successively higher index numbers, such that index N is assigned to the station immediately to the right of the server. It may help to think about the server as fixed at location 1 and the stations on a rotating circle (or wheel). This notation is very compact. Indeed, one can also translate the binary vector Y into the equivalent decimal representation.

If the horizon server is idle, then each of the K (K = Min$\{2d + 1; N\}$) stations in his domain are empty and thus the time to the next decision epoch (the next arrival to the domain) has an exponential distribution with parameter $K\lambda$. A decision epoch is whenever the server can commence a movement, reverse direction, or start the next service.

It is useful to relate the throughput of the system to the user's performance measures. Let B be the fraction of requests lost, which is also the probability that a station is occupied. Let TH be the throughput of the system regardless of the type of server, but excluding the dedicated server. Since the throughput of each station is TH/N, we get

$$B = 1 - TH/(\lambda N). \tag{4}$$

Finally, to get the response time let W_q be the mean request response time for those requests that are not lost upon arrival; then

$$W_q = \frac{N}{TH} - \frac{1}{\lambda}. \tag{5}$$

2. THE CASE $N = 2$

From (1) it is easy to verify that for $N = 2$ the throughput of the polling server is

$$P = e^{\lambda S}(e^{2\lambda} - 1)/[1 + e^{\lambda S}(e^{2\lambda} - 1)(1 + S)]. \tag{6}$$

The next result gives the throughput of the horizon server with $d = 1$ and also the throughput of the greedy server for $N = 2$.

Proposition 1. For $N = 2$ and $d = 1$ the throughput of the horizon server is

$$H(1) = \frac{2e^{\lambda S} + e^{-\lambda} - 1}{e^{-\lambda}\left(\frac{1}{\lambda} + S\right) + (2e^{\lambda S} - 1)(1 + S)}. \tag{7}$$

Proof. This model can be analyzed as a semi-Markov process where the state space is the binary vector $Y = (y_1, y_2)$ or the equivalent integer between zero and three. The transition probability matrix is given by the matrix in Table I.

Table I
Transition Probability Matrix for $N = 2$

	(0, 0)	(0, 1)	(1, 0)	(1, 1)
0 = (0, 0)	0	1/2	1/2	0
1 = (0, 1)	0	0	$e^{-\lambda}$	$1 - e^{-\lambda}$
2 = (1, 0)	$e^{-\lambda S}$	$1 - e^{-\lambda S}$	0	0
3 = (1, 1)	0	1	0	0

The duration of state 0 is exponentially distributed with parameter 2λ. In state 1 the server moves from one station to the next and the duration is one unit of time. The duration of states 2 and 3 is S, which is the service time. The transition probabilities follow easily from the exponential assumption and from the definition of the horizon server.

Let X_i be the limiting probabilities for the above Markov chain. This yields:

$$X_0 = 2e^{-\lambda}/(3e^{-\lambda} - 2 + 4e^{\lambda S}), \tag{8}$$

$$X_1 = (2e^{\lambda S} - 1)/(3e^{-\lambda} - 2 + 4e^{\lambda S}), \tag{9}$$

$$X_2 = 2e^{-\lambda}e^{\lambda S}/(3e^{-\lambda} - 2 + 4e^{\lambda S}), \tag{10}$$

$$X_3 = (1 - e^{-\lambda})(2e^{\lambda S} - 1)/(3e^{-\lambda} - 2 + 4e^{\lambda S}). \tag{11}$$

Each visit to states 2 or 3 is a service cycle. Thus, the expected number of requests per unit time, the throughput, is given by

$$H(1) = (X_2 + X_3)\Big/\left(\frac{1}{2\lambda}X_0 + X_1 + SX_2 + SX_3\right). \tag{12}$$

By substituting (8), (9), (10) and (11) in (12) and then doing some calculations, we obtain (7).

From (2) and (7) it is easy to verify that for $\lambda < (>)\ 1$, $H(1) > (<)\ H(0)$, and for $\lambda = 1$, $H(1) = H(0)$.

The next proposition shows that the horizon server outperforms the polling server.

Proposition 2. $H(1) > P$ for all $S \geq 0$ and $\lambda > 0$.

Proof. We show that $H(1) - P > 0$.

$$[1 + e^{\lambda S}(e^{2\lambda} - 1)(1 + S)] \cdot \left[e^{-\lambda}\left(\frac{1}{\lambda} + S\right) + (2e^{\lambda S} - 1)(1 + S)\right][H(1) - P]$$

$$= 2e^{\lambda S} + e^{-\lambda} - 1 + (e^{2\lambda} - 1)\left(1 - \frac{1}{\lambda}\right)e^{\lambda(S-1)}$$

$$= e^{\lambda S}\left[2 - e^{-\lambda S}(1 - e^{-\lambda}) + (e^{2\lambda} - 1)\left(1 - \frac{1}{\lambda}\right)e^{-\lambda}\right]$$

$$> e^{\lambda S}\left[2 - (1 - e^{-\lambda}) + (e^{2\lambda} - 1)\left(1 - \frac{1}{\lambda}\right)e^{-\lambda}\right]$$

$$= \frac{e^{\lambda S}}{\lambda}[\lambda + \lambda e^{\lambda} - e^{\lambda} + e^{-\lambda}]$$

$$> 0,$$

because the derivative of the last expression is positive for all $\lambda > 0$, and equal to zero for $\lambda = 0$.

When $\lambda > 1$ we obviously have $H(0) > H(1) > P$.

3. THE CASE $N = 3$

From (1), one can verify that for $N = 3$ the throughput of the polling server is

$$P = \left[S + \frac{a^6b^2 + (b^3 - a^3)(2a^3b - a^3 + b^2)}{(b^3 - a^3)(a^3b - a^3 + b^2)} \right]^{-1}, \quad (13)$$

where

$$a = e^{-\lambda(1+S)} \quad (14)$$

and

$$b = e^{-\lambda S}. \quad (15)$$

The next result gives the throughput of the horizon server with $d = 1$ and also the throughput of the greedy server for $N = 3$.

Proposition 3. For $N = 3$ and $d = 1$ the throughput of the horizon server is

$$H(1) = \frac{a^3 + h}{\left(\frac{1}{\lambda} + S\right)a^3 + (S + 1)h}, \quad (16)$$

where

$$h = a(3 - b^2) + a^2(1 - b)^2 + (1 - a)^2(3 - b^2). \quad (17)$$

Proof. This model can be analyzed as a semi-Markov process where the state space is the binary vector $Y = (y_1, y_2, y_3)$ or the equivalent integer between zero and seven. The transition probability matrix is given in Table II.

The duration of state 0 in the table is exponentially distributed with parameter 3λ. In states 1, 2 and 3 the server moves and the duration is one unit of time. The duration of states 4, 5, 6 and 7 is S, which is the service time. The transition probabilities follow from the exponential assumption and the definition of the horizon server.

We find the limiting probabilities for the above Markov chain. This yields:

$$X_0 = 3a^3/2[3 - b^2 - 3a + 4a^2 + 2a^3 + ab^2 + 2a^2b] \quad (18)$$

$$X_1 = X_0[3 - b^2 - 2a + ab]/3a \quad (19)$$

$$X_2 = X_0[3 - b^2 - 3a + 2a^2 + ab^2 - a^2b]/3a^2 \quad (20)$$

$$X_3 = X_0[3 - b^2 - 6a + 4a^2 + 2ab^2 - 2a^2b]/3a^3 \quad (21)$$

$$X_4 = X_0/b^2 \quad (22)$$

$$X_5 = X_0(b - a)(3 - b^2)/3ab^2 \quad (23)$$

$$X_6 = X_0[3b - b^3 - 3ab + 4a^2b + ab^3 - a^2b^2 - 3a^2]/3a^2b^2 \quad (24)$$

$$X_7 = X_0(b - a)[3b - b^3 - 3ab + 4a^2b + ab^3 - a^2b^2 - 3a^2]/3a^3b^2. \quad (25)$$

Each visit to states 4, 5, 6, or 7 is a repair cycle. Thus, the expected number of repairs per unit time, the throughput, is given by

$$H(1) = \frac{X_4 + X_5 + X_6 + X_7}{\frac{X_0}{3\lambda} + X_1 + X_2 + X_3 + S(X_4 + X_5 + X_6 + X_7)}. \quad (26)$$

Finally, substituting (18), (19), (20), (21), (22), (23), (24) and (25) in (26) and then doing some calculations, we obtain (16).

The next proposition shows that, for $1 > \lambda > 0$, the horizon server, $H(1)$, outperforms the polling server.

Proposition 4. $H(1) > P$ for all $S \geq 0$ and $1 > \lambda > 0$.

Proof. Suppose that $H(1) = P$. Using (13) and (16) we get

$$\frac{a^6b^2 + (b^3 - a^3)(2a^3b - a^3 + b^2)}{(b^3 - a^3)(a^3b - a^3 + b^2)} = \frac{h + a^3/\lambda}{h + a^3},$$

and this is equivalent to

Table II
Transition Probability Matrix for $N = 3$

	(0, 0, 0) 0	(0, 0, 1) 1	(0, 1, 0) 2	(0, 1, 1) 3	(1, 0, 0) 4	(1, 0, 1) 5	(1, 1, 0) 6	(1, 1, 1) 7
(0, 0, 0) 0	0	1/3	1/3	0	1/3	0	0	0
(0, 0, 1) 1	0	0	0	0	$\left(\frac{a}{b}\right)^2$	$\frac{a}{b} - \left(\frac{a}{b}\right)^2$	$\frac{a}{b} - \left(\frac{a}{b}\right)^2$	$\left(1 - \frac{a}{b}\right)^2$
(0, 1, 0) 2	0	0	0	0	$\left(\frac{a}{b}\right)^2$	$\frac{a}{b} - \left(\frac{a}{b}\right)^2$	$\frac{a}{b} - \left(\frac{a}{b}\right)^2$	$\left(1 - \frac{a}{b}\right)^2$
(0, 1, 1) 3	0	0	0	0	0	0	$\frac{a}{b}$	$1 - \frac{a}{b}$
(1, 0, 0) 4	b^2	$b(1 - b)$	$b(1 - b)$	$(1 - b)^2$	0	0	0	0
(1, 0, 1) 5	0	b	0	$1 - b$	0	0	0	0
(1, 1, 0) 6	0	0	b	$1 - b$	0	0	0	0
(1, 1, 1) 7	0	0	0	1	0	0	0	0

$$\frac{a^3(1-b)}{b^3} = \frac{(b^3-a^3)(-1+1/\lambda)-b^2(h+a^3)}{b(b^3-a^3)(-1+1/\lambda)-b^2(h+a^3)}. \quad (27)$$

For all $0 < \lambda < 1$ the right-hand side of (27) is greater than 1, and the left-hand side is less than 1, contradicting $H(1) = P$. Thus, it suffices to confirm $H(1) > P$ for one arbitrary point to complete the proof. For $S = 0$ and $\lambda = -\ln(1/2)$ we get $H(1) = 0.967 > P = 0.875$.

For $\lambda > 1$, we clearly have $H(0) > H(1)$ and also $H(0) > P$.

4. SOME RESULTS FOR $N \geq 2$

The polling, greedy or horizon server alternates between service and nonservice cycles. The duration of the service cycle is the deterministic service time S, and the nonservice cycle is the random time between two consecutive service cycles. The nonservice cycle may include waiting and/or moving periods. To compare between two different control policies, it is sufficient to compare the expected nonservice cycle times because the service cycle is the same for all policies. For the polling server the expectation of the nonservice cycle is greater than 1 and therefore

$$H(0) > P, \quad \lambda \geq 1. \quad (28)$$

In the next proposition we relate $H(1)$ to $H(0)$.

Proposition 5. For all $N \geq 2$

$$H(1) > H(0), \quad 0 < \lambda < 1. \quad (29)$$

Proof. For $H(0)$ the expected nonservice cycle is $1/\lambda$. For $N = 2$, (29) follows from (2) and (7). For the horizon server with $d = 1$ and $N \geq 3$ let Q be the probability that the nonservice cycle is 1. This is the case when at the completion of the service cycle a new request for service is already waiting. Let NS be the duration of the nonservice cycle. If no request is waiting, the waiting time for a new request is exponential with mean $1/3\lambda$. Therefore, the expectation of NS for the horizon server with $d = 1$ is given by (30).

$$E(NS) = Q + (1-Q)[\tfrac{1}{3}(1/3\lambda) + \tfrac{2}{3}(1 + 1/3\lambda)] \quad (30)$$

$$= Q + (1-Q)\left(\frac{2}{3} + \frac{1}{3\lambda}\right) < 1,$$

$$0 < \lambda < 1,\ 0 < P < 1.$$

The result follows, because for the horizon server with $d = 0$, $E(NS) = 1/\lambda > 1$ for all $0 < \lambda < 1$.

Corollary 1. *For all N there is some $\lambda_1(N) < 1$ where,*

$$H(1) > P \quad \textit{for all } \lambda_1(N) < \lambda \leq 1. \quad (31)$$

This follows because for $\lambda = 1$, $H(0) = H(1) = 1/(1+S) > P$.

For $N \geq 4$ we cannot get a closed-form solution for the throughput of the horizon server with parameter $d \geq 1$. However, for a given N, d, λ and S, the throughput of the horizon server, $H(d)$, can be found by solving a set of 2^N linear equations. Similar to Table II one can find the transition probability matrix for a larger N.

Tables III–IX provide numerical results for the throughput of the horizon and the polling servers when $N = 4$ and $N = 6$ for different values of S, λ, and d. We have used the True BASIC structured language system with a precision of sixteen digits. These tables provide enough numerical evidence to indicate that for $N \leq 6$ the horizon server, with the optimal value of d, outperforms the polling server. The greedy server is a special case of the horizon server when $d = N/2$ if N is even or $d = (N-1)/2$ if N is odd. Thus, by definition, the horizon server outperforms the greedy server. In some cases the difference in the performance is significant. In Table III when $\lambda = 0.4$ we have $H(2) = 0.861$, $H(1) = 0.872$, and $P = 0.798$. When $\lambda = 0.5$ we get $H(2) = 0.917$, $H(1) = 0.935$, and $P = 0.865$. Indeed, it is possible to show that the throughput of the horizon server can be arbitrarily large compared with the greedy or the polling server. The nonservice cycle for the horizon server with $d = 0$ is $1/\lambda$. For the polling server this cycle is at least 1, and for the greedy server this cycle is at least 2/3 (see 30). Thus, for $S = 0$,

$$\lim_{\lambda \to \infty} H(0)/P = \infty \quad (32)$$

and

$$\lim_{\lambda = \infty} H(0)/H(d) = \infty, \quad d \geq 1. \quad (33)$$

5. APPROXIMATION FOR THE OPTIMAL VALUE OF D

To use our control policy, the horizon server, one needs to find the optimal value of d. This value depends on λ, S, and N and finding it may be computationally prohibitive even for small N. To overcome this obstacle, this section develops an approximation for the optimal value of d. We start with Proposition 6.

Let $EW(k)$ be the expected duration of the moving cycle for the horizon server, given a request from a distance k initiated this cycle. This is the total time that the

Table III
The Throughputs for $N = 4$ When $S = 0$

λ	H(2)	H(1)	P
0.1	**0.354160**	0.331164	0.329680
0.2	**0.605842**	0.585127	0.550671
0.3	**0.765551**	0.761313	0.698806
0.32466	**0.793972**	**0.793972**	0.727097
0.4	0.861072	**0.871707**	0.798103
0.5	0.917261	**0.935034**	0.864665
0.6	0.950424	**0.968945**	0.909282
0.7	0.970159	**0.986166**	0.939190
0.8	0.981985	**0.994517**	0.959238
0.9	0.989106	**0.998363**	0.972676
1.0	0.993404	**1.000000**	0.981684

Table IV
The Throughputs for $N = 4$ When $S = 0.1$

λ	H(2)	H(1)	P
0.1	**0.349491**	0.326684	0.325441
0.2	**0.587106**	0.567503	0.535321
0.3	**0.728629**	0.725081	0.669521
0.323	**0.751187**	**0.751190**	0.692755
0.4	0.807800	**0.816850**	0.755197
0.5	0.851646	**0.865476**	0.809987
0.6	0.876197	**0.889548**	0.845123
0.7	0.890131	**0.900894**	0.867723
0.8	0.898118	**0.906020**	0.882298
0.9	0.902724	**0.908221**	0.891717
1.0	0.905392	**0.909091**	0.897815

server is in a continuous movement between stations. The next proposition gives $EW(k)$.

Proposition 6

$$EW(k) = \sum_{i=1}^{k} e^{-\lambda(i-1)^2}. \tag{34}$$

Proof. Index the original position of the server as 0. At time 0 all the queues at locations $0, \pm 1, \pm 2, \ldots, \pm k - 1$ are empty and a new request arrives at queue k and initiates the moving cycle. Suppose that when- ever there is a tie, the server goes to the +. For exam- ple, if at time 1, when the server is at location 1, queues -1 and $+3$ are occupied, the server selects $+3$ for its next service. We show the result by induction. For $k = 1$, clearly, $EW(1) = 1$. Now suppose that (34) holds for $k - 1$; then we only have to show that

$$EW(k) - EW(k-1) = e^{-\lambda(k-1)^2}. \tag{35}$$

Let x be the duration of the moving cycle, which is a nonnegative and integer valued random variable. Therefore,

$$EW(k) = \sum_{i=1}^{k} P(x \geq i). \tag{36}$$

Using (36) we get $EW(k) - EW(k-1) = P(x \geq k)$. But $P(x \geq k) = P(x = k)$ because $x \leq k$. Thus, we only need to show that

Table V
The Throughputs for $N = 4$ When $S = 1$

λ	H(2)	H(1)	P
0.1	**0.307084**	0.287299	0.287612
0.2	**0.437202**	0.427412	0.411910
0.3	**0.478773**	0.478070	0.461807
0.317	**0.482197**	**0.482196**	0.466696
0.4	0.492236	**0.493603**	0.482633
0.5	0.497016	**0.498140**	0.491830
0.6	0.498824	**0.499463**	0.496073
0.7	0.499530	**0.499850**	0.498087
0.8	0.499811	**0.499620**	0.499061
0.9	0.499924	**0.499993**	0.499537
1.0	0.499969	**0.500000**	0.499771

Table VI
The Throughputs for $N = 4$ When $S = 5$

λ	H(2)	H(1)	P
0.1	**0.161546892**	0.158027410	0.158366775
0.2	**0.166445004**	0.166330237	0.165933732
0.3	**0.166650704**	0.166649909	0.166569848
0.3136	**0.166655381**	**0.166655381**	0.166592416
0.4	0.166665393	**0.166665713**	0.166652621
0.5	0.166666563	**0.166666610**	0.166664582
0.6	0.166666658	**0.166666663**	0.166666356
0.7	0.1666666660	**0.1666666665**	0.1666666201
0.8	0.1666666666094	**0.1666666666560**	0.1666666597094
0.9	0.1666666666620	**0.1666666666662**	0.1666666656261
1.0	0.1666666666663	**0.1666666666667**	0.1666666665110

$$P(x = k) = e^{-\lambda(k-1)^2}. \tag{37}$$

Now, $x = k$ if, and only if, the server moves directly to location k. For this event to take place, queue k must remain the nearest request to the server throughout its moving cycle. Thus, queues $-k + 1 - 2i$ and queues $-k + 2 + 2i$, $i = 1, 2, \ldots, k - 2$, should remain empty during the intervals $(0, i]$, $i = 1, 2, \ldots, k - 2$, and queue $k - 1$ must remain empty during $(0, k - 1]$. The result follows because

$$(k-1)^2 = k - 1 + 2\sum_{i=1}^{k-2} i. \tag{38}$$

Let $ENS(d)$ be the expected duration of the nonservice cycle for the horizon server with parameter d, given that no request is waiting when this cycle starts. The next proposition gives $ENS(d)$.

Proposition 7

$$ENS(d) = \frac{1}{\lambda(2d+1)} + \frac{2}{2d+1}\sum_{i=1}^{d}(d+1-i)e^{-\lambda(i-1)^2}. \tag{39}$$

Proof. Each of the $2d + 1$ queues has the same probability to become occupied first. Conditioning on the queue that become occupied first and using (34) we get

Table VII
The Throughputs for $N = 6$ When $S = 0$

λ	H(3)	H(2)	H(1)	P
0.1	**0.485452**	0.479288	0.403894	0.451188
0.1718	**0.687741**	**0.687743**		0.643279
0.2	0.742508	**0.744017**	0.681533	0.698806
0.3	0.868470	**0.870742**	0.846896	0.834701
0.4	0.932622	**0.933737**	0.932615	0.909282
0.4087		**0.937485**	**0.937482**	0.913896
0.5	0.965803	0.966224	**0.972389**	0.950213
0.6	0.982847	0.982990	**0.989385**	0.972676
0.7	0.991487	0.991528	**0.996205**	0.985004
0.8	0.995799	0.995814	**0.998792**	0.991770
0.9	0.997937	0.997941	**0.999710**	0.995483
1.0	0.998989	0.998990	**1.000000**	0.997521

Table VIII
The Throughputs for $N = 6$ When $S = 0.1$

λ	H(3)	H(2)	H(1)	P
0.1	**0.476331**	0.470441	0.396909	0.443181
0.1713	**0.661132**	**0.661134**		0.620781
0.2	0.709962	**0.711237**	0.655583	0.671348
0.3	0.814743	**0.816325**	0.797730	0.787270
0.4	0.864115	**0.864771**	0.864522	0.846327
0.4026		**0.865615**	**0.865615**	0.847396
0.5	0.887825	0.888037	**0.892456**	0.876607
0.6	0.899140	0.899202	**0.903235**	0.892225
0.7	0.904473	0.904490	**0.907162**	0.900316
0.8	0.906958	0.906963	**0.908522**	0.904519
0.9	0.908108	0.908109	**0.908964**	0.906707
1.0	0.908638	0.908638	**0.909091**	0.907847

$$ENS(d) = \frac{1}{2d+1}\left\{\frac{1}{\lambda(2d+1)} + 2\sum_{i=1}^{d}\left[\frac{1}{\lambda(2d+1)} + EW(i)\right]\right\}$$

$$= \frac{1}{2d+1}\left\{\frac{1}{\lambda(2d+1)} + \frac{2d}{\lambda(2d+1)} + 2\sum_{j=1}^{d}\sum_{i=1}^{j} e^{-\lambda(i-1)^2}\right\}$$

$$= \frac{1}{\lambda(2d+1)} + \frac{2}{2d+1}\sum_{i=1}^{d}(d+1-i)e^{-\lambda(i-1)^2}.$$

We can now derive our approximation for d. It is based on minimizing the time until the next service cycle. The horizon server with parameter d ignores all requests from outside its domain, even if no requests are waiting inside the domain, provided that $EW(d+1) \geq ENS(d)$. Equating $EW(d+1)$ to $ENS(d)$ and simplifying the equation yields

$$\lambda\sum_{i=1}^{d+1}(2i-1)e^{-\lambda(i-1)^2} = 1. \tag{40}$$

The numerical solutions $\lambda^*(d)$, for all $d \leq 51$, are given in Table X. With this table one can design the horizon server for all systems with $N \leq 103$, and for all systems (regardless of N) with $\lambda \geq 0.00134187$. If, for a given N, $d \geq N/2$ for N even or $d \geq (N-1)/2$ for N odd, then d should be $N/2$ or $(N-1)/2$, respectively. The values $\lambda^*(d)$ appear to be lower bounds on the exact values and they are asymptotes to the exact values, when S is large.

Table IX
The Throughputs for $N = 6$ When $S = 1$

λ	H(3)	H(2)	H(1)	P
0.1	**0.391426**	0.388144	0.336616	0.370265
0.1695	**0.462945**	**0.462946**		0.449519
0.2	0.476009	**0.476160**	0.461746	0.465934
0.3	0.493877	**0.493917**	0.492470	0.489918
0.3644		**0.497433**	**0.497433**	0.495220
0.4	0.498401	0.498405	**0.498585**	0.496813
0.5	0.499580	0.499580	**0.499733**	0.498962
0.6	0.499889	0.499889	**0.499950**	0.499658
0.7	0.499970	0.499970	**0.499991**	0.499887
0.8	0.499992	0.499992	**0.499998**	0.499962
0.9	0.499998	0.499998	**0.500000**	0.499987
1.0	0.499999	0.499999	**0.500000**	0.499996

Table X
The Approximate Values of d

d	$\lambda^*(d)$	d	$\lambda^*(d)$	d	$\lambda^*(d)$
1	1.000000	18	0.00819870	35	0.00259630
2	0.313142	19	0.00747294	36	0.00247172
3	0.162242	20	0.00684297	37	0.00235621
4	0.101543	21	0.00629232	38	0.00224890
5	0.0704297	22	0.00580797	39	0.00214901
6	0.0521360	23	0.00537950	40	0.00205586
7	0.0403749	24	0.00499846	41	0.00196884
8	0.0323213	25	0.00465797	42	0.00188743
9	0.0265407	26	0.00435239	43	0.00181114
10	0.0222374	27	0.00407699	44	0.00173954
11	0.0189391	28	0.00382787	45	0.00167225
12	0.0163501	29	0.00360171	46	0.00160893
13	0.0142771	30	0.00339574	47	0.00154926
14	0.0125890	31	0.00320757	48	0.00149296
15	0.0111945	32	0.00303517	49	0.00143977
16	0.0100279	33	0.00287680	50	0.00138948
17	0.00904126	34	0.00273095	51	0.00134187

6. SIMULATION RESULTS

To validate the approximations developed in Section 5, we will present some simulation results. We can get some validation from the numerical results in Section 4. For example, the approximation in Table X gives $\lambda^*(3) = 0.162242$. From Tables VII, VIII, and IX, we see that $H(2) = H(3)$ when $\lambda = 0.1718$, $\lambda = 0.1713$, $\lambda = 0.1695$ for $S = 0$, $S = 0.1$, $S = 1$, which shows that the approximation is accurate.

Tables XI–XIII provide simulation results for the throughput of the horizon server, and exact results for the polling server, when $N = 9$, for different values of S, λ, and d. They show that the first five values of the approximation are quite accurate. For example, Table X illustrates that for $0.0704 < \lambda < 0.1015$ we get $d = 4$, which is in agreement with the simulation results.

Each simulation result is based on 10,000,000 (ten million) service completions. An attempt to get meaningful simulation results for a larger value of N was not successful. The problem is that $\Delta \equiv (H(d+1) - H(d))$ is decreasing in N and, for larger values of N, this Δ is less than the simulation error.

7. FINAL REMARKS

The key factor that affects the performance advantage of the horizon server is the congestion in the system.

Table XI
Simulation Results for $N = 9$ When $S = 0$

λ	$H(4)$	$H(3)$	$H(2)$	$H(1)$	*P–Exact*
0.05	**0.38319**	0.37597	0.34803	0.26500	0.36237
0.1	**0.62175**	0.62002	0.59732	0.47920	0.59343
0.2	0.85044	**0.85072**	0.84801	0.76827	0.83470
0.3	0.94068	0.94077	**0.94088**	0.91334	0.93279
0.4	0.97726	0.97731	**0.97739**	0.97251	0.97268
0.5	0.99161	0.99158	0.99161	**0.99219**	0.98889

When the system is congested, the horizon server is likely to move as the polling server, and to serve most of the stations in every cycle. Under such heavy traffic conditions, the throughput of the system approaches $1/(1 + S)$ and the performance advantage diminishes. It is interesting that, even under such conditions, for $N \leq 6$, the horizon server is marginally better.

In the continuous polling system, as in Coffman and Gilbert (1986, 1987), requests for service arrive at random positions on the route of the server. This model cannot benefit from the horizon server, because only at the stations can the horizon (or the greedy) reverse directions. At each station, the server has full information about the state of each station. The server can update the state of the system only at the stations. Thus, a decision to move to the next station implies "black out" of new information for one unit of time. When the server does not move, this "black out" period does not occur: Information becomes available instantaneously. If, for example, the next request arrives at the station where the server is waiting, the server starts to serve immediately. This is why it pays (sometimes) to wait. The continuous model considers the limit as $N \to \infty$, while the switchover time (or distance) $\to 0$, in contrast to our model. In this way, the total switchover time (the circumference) remains a constant. In the continuous model, there are no "black out" periods, and it does not pay to wait.

Coffman and Gilbert (1987) have shown that, for the continuous model with a constant service time, the polling and greedy servers are roughly, equally effective, except for certain extreme values of the parameters. Our numerical results also show that, in many cases, the performance difference is not significant and may not justify preferring the horizon server over the polling system, because the polling server does not require the additional information about the status of the stations.

The greedy server is a special case of the horizon when $d \geq N/2$. Therefore, the horizon server always outperforms the greedy, regardless of N, the service time distribution, or the distribution of the time that the queue is empty. The greedy server also requires at least as much information about the state of the system. Thus, all applications that are currently using the greedy server can benefit by switching to the horizon server. In some cases, the difference in the performance is significant and can enhance the throughput by more than 5%. The example at the end of Section 3 shows an increase in the throughput of 10%. Note that all computations required for the horizon server are performed before the system is put in place. The real-time implementation does not involve calculation. Finally, we neither claim nor conjecture that the horizon server outperforms the polling server in general when $N \geq 7$.

Table XII
Simulation Results for $N = 9$ When $S = 0.1$

λ	$H(4)$	$H(3)$	$H(2)$	$H(1)$	*P–Exact*
0.05	**0.37895**	0.37189	0.34436	0.26228	0.358443
0.1	**0.60332**	0.60194	0.58132	0.46875	0.577456
0.2	0.80070	**0.80093**	0.79905	0.73256	0.788356
0.3	0.87050	0.87050	**0.87055**	0.85134	0.864610
0.4	0.89583	0.89583	**0.89584**	0.89337	0.892556
0.5	0.90468	0.90467	0.90469	**0.90519**	0.902941

Table XIII
Simulation Results for $N = 9$ When $S = 1$

λ	$H(4)$	$H(3)$	$H(2)$	$H(1)$	*P–Exact*
0.05	**0.33550**	0.33088	0.30927	0.23823	0.319749
0.1	**0.44664**	0.44646	0.44094	0.38115	0.438328
0.2	0.49260	**0.49260**	0.49255	0.48434	0.490952
0.3	0.49895	0.49895	**0.49895**	0.49853	0.498444
0.4	0.49985	0.49985	0.49985	**0.49986**	0.499720
0.5	0.49998	0.49998	0.49998	**0.49999**	0.499950

ACKNOWLEDGMENT

This research was supported in part by a grant from the Council of Business Studies, Rutgers University. The work of the first author was also supported by the Research Resources Committee of the Graduate School of Management, Rutgers University, by the Graduate School Research Award, and by the Research Council of Rutgers University. We wish to thank Dr. H. Takagi and Dr. H. Levy for their comments on an earlier version of this paper. Many thanks are also due to Dr. B. Gavish for his comments which improved the presentation of this paper.

REFERENCES

ABDEL-MALEK, L., AND Z. LI. 1990. The Application of Inverse Kinematics in the Optimum Sequencing of Robot Tasks. *Int. J. Prod. Res.* **28,** 75–90.

BUNDAY, B. D., AND C. MACK. 1973. Efficiency of Bidirectionally Traversed Machines. *J. Royal Statist. Soc.* (Series C), Applied Statistics **23,** 74–81.

COFFMAN, E. G., AND E. N. GILBERT. 1986. A Continuous Polling System With Constant Service Times. *IEEE Trans. Infor. Theory* **IT-32**(4), 584–591.

COFFMAN, E. G., AND E. N. GILBERT. 1987. Polling and Greedy Servers on a Line. *Queue. Syst.* **2**(2), 115–145.

COFFMAN, E. G., AND M. HOFRI. 1982. On the Expected Performance of Scanning Disks. *SIAM J. Comput.* **11**(1), 60–70.

DAVIDSON, R. P., AND N. J. MULLER. 1992. *Internetworking LANs, Operation, Design and Management.* Artech House Inc., Boston.

GAVISH, B. 1982. Topological Design of Centralized Computer Networks—Formulations and Algorithms. *Networks* **12,** 355–377.

GAVISH, B. 1991. Topological Design of Telecommunication Networks—Local Access Network Design Methods. *Anns. Opns. Res.* **33**(1), 17–71.

GAVISH, B., AND P. SCHWEITZER. 1973. Queue Regulation Policies Using Full Information. Technical Report 004, IBM Israel Scientific Center, Haifa, Israel.

GEIST, R., AND S. DANIEL. 1983. V-SCAN: An Adaptive Disk Scheduling Algorithm. *Proc. IEEE Inf. Symp. on Comp. Sys. Org.*, New Orleans.

GRILLO, D. 1990. Polling Mechanism Models in Communication Systems—Some Application Examples. In *Stochastic Analysis of Computer and Communication Systems*, H. Takagi (ed.). Elsevier Science Publishers, Amsterdam, 659–698.

HAMMOND, J. L., AND P. J. P. O'REILLY. 1986. *Performance Analysis of Local Computer Networks*. Addison-Wesley, Reading, Mass.

HAYES, J. F. 1984. *Modeling and Analysis of Computer Communications Networks*. Plenum Press, New York.

HOFRI, M. 1980. Disk Scheduling FCFS vs. SSTF Revisited. *Commun. ACM* **23**(11), 645–653.

LEVY, H., AND M. SIDI. 1990. Polling Systems: Applications, Modeling, and Optimization. *IEEE Trans. Commun.* **38**(10), 1750–1760.

MACK, C., T. MURPHY AND N. L. WEBB. 1957. The Efficiency of N Machines Uni-Directionally Patrolled by One Operative When Walking Times and Repair Times Are Constants. *J. Royal Statist. Soc.* (Series B) **19**, 166–172.

SCHWARTZ, M. 1987. *Telecommunication Networks: Protocols, Modeling and Analysis*. Addison-Wesley, Reading, Mass.

SOSNOSKY, J., AND T. H. WU. 1991. SONET Ring Applications for Survivable Fiber Loop Networks. *IEEE Commun. Mag.* **29**(6), 51–58.

TAKAGI, H. 1986. *Analysis of Polling Systems*. MIT Press, Cambridge, Mass.

TAKAGI, H. 1988. Queueing Analysis of a Polling Model. *ACM Surveys* **20**, 5–28.

TAKAGI, H. 1990. Queueing Analysis of Polling Models: An Update. In *Stochastic Analysis of Computer and Communication Systems*, H. Takagi (ed.). Elsevier Science Publishers, Amsterdam, 267–318.

TAKAGI, H. 1991. Application of Polling Models to Computer Networks. *Computer Network and ISDN Systems* **22**(3), 193–211.

TANENBAUM, A. S. 1988. *Computer Networks*. Prentice-Hall, Englewood Cliffs, New Jersey.

TEOREY, T. J., AND T. B. PINKERTON. 1972. A Comparative Analysis of Disk Scheduling Policies. *Commun. ACM* **15**(3), 177–184.