A Network Version of The Pump

Myong H. Kang, Ira S. Moskowitz, and Daniel C. Lee
Naval Research Laboratory
Washington, D.C. 20375*

Abstract

A designer of reliable MLS networks must consider covert channels and denial of service attacks in addition to traditional network performance measures such as throughput, fairness, and reliability. In this paper we show how to extend the NRL data Pump to a certain MLS network architecture in order to balance the requirements of congestion control, fairness, good performance, and reliability against those of minimal threats from covert channels and denial of service attacks. We back up our claims with simulation results.

1 Introduction

In a MLS system, a low subject (Low) should be able to send information to a high subject (High), but High should not be able to send information to Low. On the other hand, acknowledgements (ACK) to Low that High has received its messages are necessary for reliability and performance. This is especially true for distributed systems in which the communication channels may not always be reliable. High, however, can manipulate the times that ACKs arrive in order to covertly send unauthorized messages to Low.

The Pump, developed at NRL [3, 4], solves the dilemma of simultaneously assuring reliability, performance and security. We will refer to this as the basic Pump. The basic Pump allows High to send ACKs to Low, but requires that Low receive them at probabilistic time intervals. The basic Pump bases these probabilistic times on past High activity, and moderates the ACK times through the use of a communication buffer.

Since computer systems are becoming more open and interconnected, denial of service problems are receiving more attention [12]. Hence, security devices should also provide protection against such attacks.

*Respective addresses of the authors are (mail code 5540, mail code 5540, mail code 8140) Naval Research Laboratory, Washington, D.C. 20375. Respective e-mail addresses are *mkang@itd.nrl.navy.mil, moskowit@itd.nrl.navy.mil, lee@kingcrab.nrl.navy.mil*

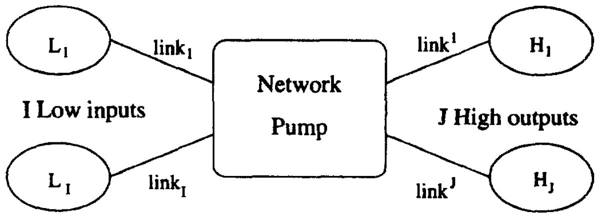


Figure 1: The Pump in a network environment.

This paper addresses how to adapt the basic Pump for use in a network environment, where we have multiple Lows and multiple Highs. We will refer to this as the "network Pump." As we move from a dedicated data and copper world to the B-ISDN[1] world of ATM[2], the issues of congestion control, fairness, and reliability become extremely important and extremely complicated. The network community itself has not worked all of these issues out yet. Our problem is even more complex because we are coupling security (i.e., covert channels and denial of service) with the above.

1.1 Assumptions and Terminology

The network environment that is considered is shown in figure 1.

There are many Lows and Highs, and they are untrusted processes. The network Pump is a trusted process which mediates traffic from Lows to Highs. Each message that will be routed from a Low to a High has a message number, and input and output addresses associated with it. For simplicity, we assume that all messages have the same length. We do not consider multicasting in this paper. Lows (Highs) do not communicate among themselves.

A session is a communication channel between any Low and any High. In figure 1 there are $I \times J$ distinct sessions. During each session_{ij}, a message leaves Low_i, travels over link_i, goes into the network Pump, and

[1]Broadband Integrated Services Digital Network.
[2]Asynchronous Transfer Mode.

1081-6011/95 $04.00 © 1995 IEEE

after processing leaves the network Pump over link^j, and arrives at High$_j$. We assume that all propagation delays are zero for conceptual simplicity[3]. The minimal processing time of the network Pump is a set overhead value O_v, which is small enough so that the network Pump itself never becomes a performance bottleneck.

The input rate λ_{ij} is the rate of inputs from Low$_i$ destined for High$_j$. Hence, $\frac{1}{\lambda_{ij}}$ is the mean interarrival time of inputs. Each High$_j$ behaves as a server with service rate μ_{ij}. This is the inverse of the mean time of service by High$_j$ for messages from Low$_i$.

1.2 Objectives

Most network resources are dynamically shared for efficiency reasons. If this dynamic sharing is not carefully controlled then inefficiency and delays occur [1]. The main functions of congestion control in a network are:

- To prevent inputs from sending messages faster than the outputs can handle them.
- To prevent throughput degradation and loss of efficiency from overloading the network.
- To prevent unfair allocation of network resources from competing inputs.

Since the network Pump is a shared resource among many sessions, it should provide the congestion control mechanism. Let us discuss the specific objectives required of the network Pump.

Reliability / Handshaking

The reliability requirement can be simply stated as *no loss of messages and no duplication of messages.* To satisfy this requirement ACKs and message numbers (ID) are necessary. The network Pump has a reliability protocol that works as follows:

> If a Low has not received ACK by `time_out` after sending a message, it will retransmit the same message. If a High receives the same message then it will keep only one copy.

Further, a Low does not send the next message to a specific High until its previous message to that High has been ACKed (handshake protocol).

[3]This assumption can be easily relaxed.

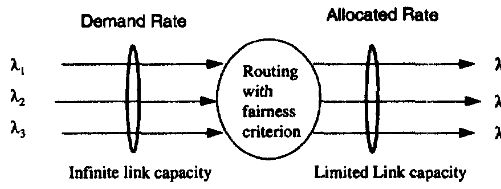


Figure 2: fairness

Performance

We desire good performance. The network Pump is designed to achieve this by exercising congestion control. The network Pump controls the rates into itself (input rates) by attempting to slave the input rates to the average rates out (service rates) of the network Pump by moderating the ACK rate to a Low, since this Low will not send a new message until it receives an ACK from the previous (session) message.

If the service rates are greater than the input rates, then the network Pump should not hurt performance. If service rates are less than the input rates, then the outputs cannot handle their inputs. Therefore, the network Pump by slowing the input rate, alleviates congestion, and at the worst, does not lessen total throughput.

Fairness

Bandwidth of communication links, transmission speed, and processing speed are all limited. Therefore, if the load of data traffic offered to the network Pump exceeds its capability, some of the load must be cut. The load must be cut fairly for all the sessions that share the network Pump. The idea is shown in figure 2 where the output limitation is due to limited output link capacity.

Fairness can be defined in different ways. One fairness policy is *max-min fairness.* This policy says all sessions should get bandwidth according to the following criterion — the smallest allocated rate is as large as possible and, given this, the second-smallest allocated rate is as large as possible, etc.[2]. For example, if there are three sessions whose demand rates are 0.4, 0.5, 0.6 and the output capacity equals 1 then all three sessions will be allocated rates of 1/3, 1/3, 1/3 under max-min fairness. If one session demands less than what it can get, the leftover bandwidth will be equally shared among the rest of the sessions. For example, if there are three sessions whose demand rates are 0.2, 0.5, 0.6 and the output capacity equals 1 then those sessions will be allocated 0.2, 0.4, 0.4, respectively under max-min fairness.

The advantages of this policy are (1) there is a simple way to implement this policy (i.e., round-robin scheduling [2]) and (2) the scheduling scheme does not need to know the demand rates of sessions which may not always be known. Since this policy gives preference to sessions that have lower demand rate, it does not allow a session to take the entire bandwidth if there is more than one session. One disadvantage of this policy is that a heavily demanded session is penalized more than a lightly demanded session (i.e., not sensitive to demand rates).

There are other fairness policies such as the proportional policy [11]. This policy allocates bandwidth in proportion to each input demand. The network community does not have a "best" fairness policy. The network Pump uses max-min fairness because of the above advantages.

Covert Channels

It is well known that the ACK stream that is required to satisfy the reliability requirement introduces covert channels. This was the motivation for developing the basic Pump over the conventional store and forward buffer type of communication. We will show in section 3.2, as we did in [3, 4] that the capacity of the covert channels can be made negligible.

Denial of Service

We interpret the denial of service attack in a broad sense in the network environment:

> If a session cannot achieve its intended throughput due to the misbehavior of other sessions then the session is under a denial of service attack.

Since the network Pump is a shared resource among several sessions, services for other sessions can be potentially disrupted if too much resource is allocated to one particular session. The design of the network Pump should prevent such a situation.

2 Background — The Basic Pump

In the basic Pump our concern is sending messages from (one) Low to (one) High. In [3, 4], we reviewed why traditional communication protocols (including *read-down* and *blind write-up*) cannot satisfy the needs for reliability, performance, and security simultaneously. As a solution, the basic Pump was introduced as shown in figure 3.

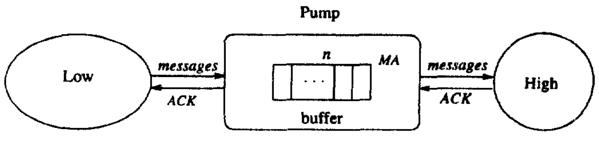


Figure 3: The Basic Pump

The basic Pump [3] places a buffer (size n) between Low and High, and gives ACKs at probabilistic times to Low based upon a moving average (MA) of the past m High ACK times. A High ACK time is the time from when the buffer sends a message to High to the time when High sends an ACK back. This has the double benefit of keeping the buffer from filling up and having a minimal negative impact upon performance. The actual ACK time to Low is, if there was space on the buffer when Low sent the message, an exponential random variable with mean equal to MA, shifted by an amount of time equal to the minimum processing time of a message. When Low must wait for space on the buffer the shift is equal to *max*(wait time for space, minimum processing time). (In [4] we have slightly modified this over the first exposition of the basic Pump [3] to introduce extra noise. However, for the network Pump we stay with the simpler formulation.)

At present, the basic Pump has been built by HFSI to run on a XTS-300 platform. Also the basic Pump is being built as a device by the prototype laboratory of NRL's Center for High Assurance Computer Systems. Early results, along with the simulation results of Kang and Moskowitz [4], show a proof of concept for the basic Pump. Based upon this and the need for a secure network congestion mediator we feel the extension to the network environment is warranted.

3 An Architecture of the Network Pump

The architecture of a network Pump is shown in figure 4.
Each component of the network Pump works as follows:

Lows and Highs: (Exterior to the network Pump)
Lows (Highs) is the set of inputs (outputs) to the

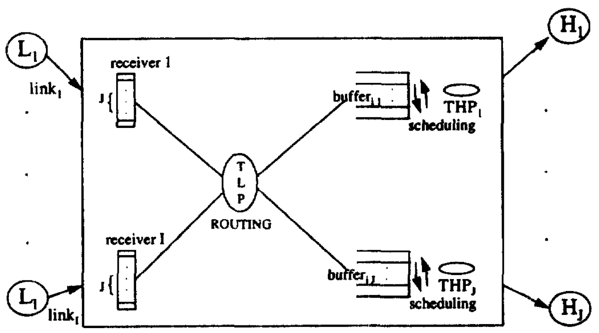


Figure 4: A logical view of the network Pump.

network Pump. Lows (Highs) consists of I (J) non-communicating among themselves processes. Each input can send messages to any output, with various rates as discussed before. Since Low$_i$ is outside of the network Pump we assume the Low$_i$ has some procedure for sending messages over link$_i$, the only constraint being that the rates are λ_{ij}, such that $\sum_j \lambda_{ij} \leq$ capacity of link$_i$. Consider session$_{ij}$ — After Low$_i$ sends a message to High$_j$, it waits for the ACK to that message from the network Pump. Once this ACK arrives, Low$_i$ can send another message to High$_j$. Therefore, each Low can send only one message in each session without receiving the ACK to the previous message from the network Pump (handshake protocol). When High$_j$ receives a message from the network Pump it sends an ACK back after the appropriate service time to the network Pump.

Receivers
There is one receiver$_i$ for Low$_i$. In each receiver, there are J slots; slot$_j$ stores a messages from session$_{ij}$ until it is routed by the TLP.

Trusted Low Process (TLP):
The TLP takes a message from a receiver and routes it to the appropriate output buffer. We denote by T_r the time from when a message is sent from a Low to the time when that message is placed in the appropriate output buffer. (We will also refer to T_r as "routing time.") If there is available space in the output buffer, T_r is equal to the overhead O_v. If there is no space, the message is not placed until there is a space available. Therefore, T_r includes both O_v and the amount of time the message waits until the output buffer is available. After the message is routed to the output buffer, the TLP is ready to send an ACK back to the appropriate Low$_i$. The time this ACK arrives at Low$_i$ depends on the randomization scheme, but is always at least T_r.

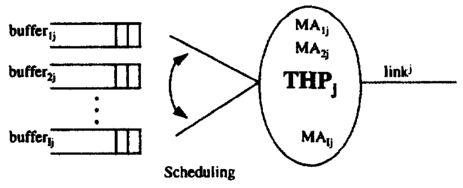


Figure 5: A closer view of a trusted high process.

Output Buffers
There are I logical output buffers for H$_j$, denoted by buffer$_{ij}$. A message from session$_{ij}$ will be stored in buffer$_{ij}$.

Trusted High Processes (THP):
THP$_j$ delivers a message from buffer$_{ij}$ to High$_j$ according to a scheduling scheme. THP$_j$ cannot deliver another message from buffer$_{ij}$ until the prior message from buffer$_{ij}$ is ACKed (by High$_j$).

3.1 A Detailed Design

A detailed design rationale is described in this section.

3.1.1 Trusted High Processes

THP_j plays an important role in scheduling delivery from output buffers to High$_j$ and in computing moving averages. Figure 5 graphically describes the role of a THP$_j$.

Consider THP$_j$ — it has to deliver messages from each buffer$_{ij}$ to High$_j$. Since the capacity of link^j is limited by physical considerations and inputs may send more messages than link^j or High$_j$ can handle, THP$_j$ needs some scheduling scheme. This scheduling scheme determines the fairness among different inputs.

The network Pump uses round-robin scheduling because it is simple and achieves max-min fairness [2]. For example, if THP$_j$ has to serve three output buffers then an opportunity to send a message is given in the order of buffer$_{1j}$, buffer$_{2j}$, buffer$_{3j}$, buffer$_{1j}$, ... If buffer$_{2j}$ does not have any message to send then the opportunity is transferred to buffer$_{3j}$ and the next opportunity is given to buffer$_{1j}$, and so on.

THP$_j$ also maintains and updates moving averages (MA_{1j}, ..., MA_{Ij}). The reason for THP$_j$ to maintain I moving averages (i.e., one per session) instead of one moving average is that High$_j$ may have different service times for messages from different Lows. Throughout this paper we assume the message service time is not a performance bottleneck in the benign case. In other words, the bottleneck is output links, not servers, unless the system is under denial of service or covert channel attacks.

Since there is potentially more than one input, the method of computing moving averages is different from when there is only one input. When there is only one input, the moving average is computed based on the interval from the time the message is sent to the time the ACK arrived from High. However, if there is more than one input then the message is ready to be sent but cannot be sent because the output link is not available. This additional waiting time must be taken into account or else the input messages will flood the output buffers.

MA_{ij} of the network Pump is the moving average of the last m High$_j$ ACK times of messages from buffer$_{ij}$. A High$_j$ ACK time is the difference between when High$_j$ ACKs a message from a buffer$_{ij}$ and max(time that message arrived in buffer$_{ij}$, time that the previous message from buffer$_{ij}$ was ACKed by High$_j$). In other words, if the buffer$_{ij}$ is not empty then previous ACK time by High$_j$ is used to compute the moving average. However, if the buffer$_{ij}$ is empty when a new message arrives then we use the arrival time instead of the previous ACK time.

3.1.2 Output Buffers

The number of messages in buffer$_{ij}$ is important to achieve fairness [2] (the bigger the number of messages in buffer$_{ij}$ the fairer). This is because our round-robin scheduler does not take burstiness into account. The way to handle bursts is to have enough messages queued in buffer$_{ij}$ so that times of abundance and starvation (with respect to message arrivals) are balanced out. In fact, it is desirable to keep the queue length in session$_{ij}$ positive so that max-min fairness is preserved. However, if the queue length is too big we have covert channel and denial of service problems. Therefore, it is desirable to keep the queue length at a certain level, which is referred to as the *Fair size*, and which we leave as a design parameter (of course the burstier the input the larger the fair size must be). Figure 6 shows buffer$_{ij}$ where the number of messages in the buffer fluctuates around the Fair size.

Since the network Pump has a built-in mechanism to share output buffers fairly among different sessions (i.e., moving average construction to control input rates which will be discussed in section 3.1.3), all output

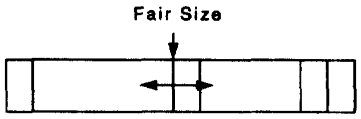


Figure 6: A closer view of buffer$_{ij}$.

buffers are dynamically shared among different sessions.

3.1.3 Trusted Low Process

When routing requests arrive from receivers, the TLP routes messages to the proper output buffers and reads the current moving average value. Once the message is delivered, the TLP is ready to send an ACK. However, this ACK will be delayed depending on the moving average of the session and the randomization scheme. The network Pump uses a similar randomization scheme as the basic Pump whose details are presented in [3, 4]. In simple terms, the TLP of the basic Pump delays ACK based on the exponential random distribution whose mean is the moving average of the session. This ACK rate controls the input rates if the input rate is higher than the service rate due to the handshake protocol.

As we discussed in section 3.1.2 we wish to make sure that the number of messages in an output buffer fluctuates around the Fair size. To achieve this, we modify the ACK scheme from the basic Pump. The way the basic Pump controls the ACK time to Low can be written as follows:

$$ACK\ time = \begin{cases} T_r & \text{if } MA - T_r \le 0 \\ \min(\ T_r + f_r(MA - T_r)\ ,\ \texttt{time\_out}\ ) & \text{otherwise} \end{cases}$$

where T_r is routing time, $f_r(x)$ is a draw from an exponential random distribution with mean x, and MA is the moving average of the ACK time from High to the basic Pump. (Note that there is only one session in the basic Pump.) Recall that T_r is the time between when the a message is sent from Low to when the message is placed in the output buffer. Hence, a random delay is included in the ACK time in addition to the routing time if $MA - T_r > 0$.

We now describe the way the Network Pump controls ACK times to Low$_i$ for a message in each session. Define

$$Q \equiv f_r(MA_{ij} - T_r) + k \cdot (N - Fair\ size)$$

where N is the number of messages in buffer$_{ij}$ at the time the message is placed in buffer$_{ij}$, and k is a design parameter that can be varied. Note that the moving average of the ACK times from High$_j$ to the network

Pump is computed separately for each session. For session_{ij}, the ACK time to Low for each message is:

$$ACK\ time = \begin{cases} T_r & \text{if } MA_{ij} - T_r \leq 0 \text{ or } Q \leq 0 \\ \min(\ T_r + Q,\ \texttt{time\_out}\ ) & \text{otherwise.} \end{cases}$$

We now elaborate the rationale of the extra term $k \cdot (N - Fair\ size)$ in Q. As long as Low_i has messages to send to High_j, the network Pump wants to keep buffer_{ij} nonempty. The reason is to prevent missing the round-robin turn, and thus to give each session throughput close to max-min fairness. Therefore, when the number of messages in buffer_{ij} is less than the Fair size, the network Pump reduces its ACK time to Low in order to accelerate the input rate, as seen in the extra term. On the other hand, if buffer_{ij} is often full, we have covert channel problems (see section 3.2). Hence, the network Pump decreases the input rate by increasing the ACK time to Low when the number of messages in buffer_{ij} is larger than the Fair size. Both k and the Fair size are design parameters that can be chosen.

In the simulation that is described in section 4, we use $k = MA_{ij}/(Fair\ size)$. Thus

$$Q = f_r(MA_{ij} - T_r) + MA_{ij}(\frac{N}{Fair\ size} - 1).$$

Therefore, we have

$$avg(ACK\ time) \approx T_r + avg(Q) = MA_{ij}(\frac{avg(N)}{Fair\ size}).$$

Note that $avg(N)$ is close to the Fair size due to the second term of Q. Thus we have $avg(ACK\ time) \approx MA_{ij}$.

3.1.4 Receivers

Receivers receive messages from Lows and request routing to the TLP. Each receiver contains J temporary (size one) buffers so that the inputs from one session do not interfere with inputs from other sessions. Messages in the temporary buffers will either be routed or discarded after `time_out` (if there is no output buffer available).

3.2 Design Review

In this section, we review the design of the network Pump and explain how the objectives in section 1.2 are satisfied. We back our claims on performance, fairness, and denial of service by the simulation results presented in section 4.

Reliability

Due to the reliability protocol requirement that was specified in section 1.2 (i.e., ACK, retransmission of the same message after `time_out`, and message ID), the network Pump provides a higher level of reliability than TCP/IP.

Performance

The network Pump does not hurt performance (throughput). Consider the following two cases:

- *Input rate is faster than the service rate*: The network Pump's ACK rate which is tied to the moving average of the server will slow down input to match the servers. However, this will not degrade performance because the throughput will be determined by the service rate which is the performance bottleneck.

- *Input rate is slower than the service rate*: The network Pump's ACK rate will not slow down the input rate in this case. Hence, there is no effect on performance.

Hence, the network Pump does not affect the throughput unless the network Pump itself is the bottleneck.

Fairness

The network Pump uses a round-robin scheduling scheme which enforces max-min fairness at THP_js if all inputs can accumulate enough messages at output buffers. The network Pump's modified moving average construction that was described in section 3.1.3 encourages all inputs to send as many messages as possible up to the Fair size. Hence, the network Pump achieves max-min fairness.

Covert Channel Analysis

In [4] we discussed how ACKs can cause a communication channel from High to Low in the basic Pump. Obviously we have the same concern with the network Pump. Let us review the full buffer channel (FBC) from the basic Pump and see its impact upon the covert covert channel analysis of the network Pump. We then discuss how a Trojan horse might use ACKs to form a statistical channel (exploitation strategy 3 in [3, 4], see also [9]).

The basic Pump was designed as a secure version of a standard store and forward buffer (SAFB). The basic Pump without probabilistic delay is the same as the SAFB. In a SAFB the High service rate can slow down so that it is less than the Low input rate. This will cause the buffer to become full. Low now attempts to insert a message into the buffer and must wait until either a `time_out` or until High finally ACKs a message to the basic Pump — thereby creating space on the buffer so that Low can insert its message and therefore receive an ACK. High and Low can now play this game of High ACKing a message whenever it wants (within the limits of the smallest amounts of manipulable time) and this time being exactly reflected to Low (modulo overhead times). This forms the FBC from High to Low. In [3, 4] we analyzed the capacity of this covert channel. The FBC capacity is simply the logarithm of the root of certain polynomials [7, 8]. However, if we use the basic Pump, the probabilistic arrival times of the ACKs introduce noise into the communication channel. Also, the basic Pump prevents the buffer from becoming/staying full. These two effects, the noise and the fact that the buffer is hardly ever full, severely diminish the capacity[4] of various exploitations of the FBC. Bounds on the capacity reductions are discussed in [3] and exactly given in [4]. In brief, a relationship between the buffer size n and the moving average MA was given with respect to the desired percent reduction of the covert channel capacity.

In the network Pump our Trojan horse scenario is that one particular High$_j$ and one particular Low$_i$ are in cahoots via cooperating Trojan horses (recall that we are looking at the situation where the Lows (Highs) do not communicate among themselves). In the network Pump we have even more noise introduced into the channel by the multiple users. Also, in the network Pump instead of just attempting to keep the buffer from become full we attempt to keep the buffer around the Fair size — this further reduces the usefulness of the FBC. Therefore, we can use the capacity bounds from [4] as a rough upper bound (we can very conservatively replace n by the Fair size). This bound becomes even rougher as I and J increase.

In the basic Pump there is a statistical channel [9] from High to Low, when the buffer is not full, caused by Low attempting to correlate the ACK times to High's actions. As the number of terms making up the moving average grows this correlation, and hence the channel capacity, decrease. The same holds as well for the network Pump and again the multiple users introduce spurious noise which further serves to confound any meaningful interpretation of the ACK times. Also, the Fair size further frustrates correlation attempts by Low$_i$. Therefore, the bounds from the basic Pump again hold.

Finally, the network Pump (as well as the basic Pump) is sensitive to the small message criterion [10]. By this we mean even if one has a channel with small, or even zero, capacity it might still be possible to send small, possibly noisy, messages in relatively quick time. The network Pump is designed to thwart such a covert communication attempt. We will not go into further details here.

[4]Here, unlike the rest of the paper, we use the term *capacity* in Shannon's information theoretic sense [13].

Denial of Service

In the network environment denial of service can occur in the following two cases:

1. A server slows down.
2. An input sends messages faster than the rate that the intended server can handle.

In these cases, the shared resources will be monopolized by this specific session so that other sessions cannot use required resources.

The above cases will not happen if the network Pump mediates between the Lows and Highs because the network Pump monitors the servers' activities and determines service rates. The service rate will be reflected to the ACK rate to Low$_i$ through the moving average construction. Due to the network Pump's handshake protocol and moving average construction, inputs (Lows) cannot send any more than the servers (Highs) can handle.

4 Simulation Results

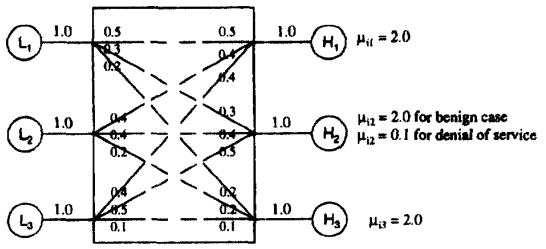
To substantiate our claims on performance, fairness and denial of service, simulation experiments have been conducted.

4.1 Simulation Set Up

In our simulation scenario, there are three Lows (L_1, L_2, L_3) and three Highs (H_1, H_2, H_3); hence 9 sessions. The capacities of all input and output links are 1.0. All inputs have Poisson arrival distributions[5]. Input rates from L_1 are $\lambda_{11} = 0.5, \lambda_{12} = 0.3, \lambda_{13} = 0.2$, the input rates from L_2 are $\lambda_{21} = 0.4, \lambda_{22} = 0.4$, $\lambda_{23} = 0.2$, and the input rates from L_3 are $\lambda_{31} = 0.4, \lambda_{32} = 0.5, \lambda_{33} = 0.1$ (see figure 7).

All Highs have 2-Erlang distributed service rates [5]. For the benign case all service rates are set to 2.0. For denial of service simulation, service rates are $\mu_{i1} = 2.0$, $\mu_{i2} = 0.1$, and $\mu_{i3} = 2.0$ for $i = 1, 2, 3$.

[5]This is an idealized input rate. Since we have congestion control this is not achieved. In our simulation we generate, in a Poisson manner, a certain number of messages which accumulate in a queue. When this queue is filled the generation stops and starts up again when this queue again has space in it.



L1
L2
L3
H1
H2
H3
1.0
0.5
0.3
0.2
0.4
0.1
$\mu_{i1} = 2.0$
$\mu_{i2} = 2.0$ for benign case
$\mu_{i2} = 0.1$ for denial of service
$\mu_{i3} = 2.0$

Figure 7: The simulation scenario.

The performance and fairness of the following two systems and the "ideal" case are compared under different total output buffer sizes.

- The network Pump as described in section 3. The last 30 High ACK times are used to compute the moving average (i.e., $m = 30$) and the Fair size per session was set to $\frac{1}{10}$ of the total output buffer size. Note that there are 9 sessions in our scenario.
- The Nonpump. This is the same as the network Pump (still has the handshake protocol) except that it does not have the moving average or probabilistic construction. (Thus, output buffers are allocated to each session on a first come first serve basis.) In other words, ACKs will be sent to Lows as soon as the message is routed. Hence, input rates will be forcibly adjusted only when there are no available output buffers. The purpose of this system is to demonstrate the importance of congestion control.
- The ideal case is where the max-min fairness rates are achieved over the output links. The max-min rates for H_1 are (1/3, 1/3, 1/3), for H_2 they are (0.3, 0.35, 0.35), and for H_3 they are (0.2, 0.2, 0.1). Hence these are the ideal versions to which we compare the network Pump and Nonpump.

4.2 Simulation Results in the Benign Case

In the benign case (i.e., inputs and outputs behave — no Trojan horses are present), there is not much of a performance difference between the network Pump and Nonpump even though the network Pump performs slightly better than the Nonpump. This slight performance difference comes from the congestion control mechanism. Since the Nonpump has little congestion control, some inputs still send more messages than the intended server can handle. This causes an unfair sharing of resources and degrades performance. This effect will be magnified under the denial of service attack. Figures 8, 9, and 10 show the performance and fairness among sessions that send messages to H_2.

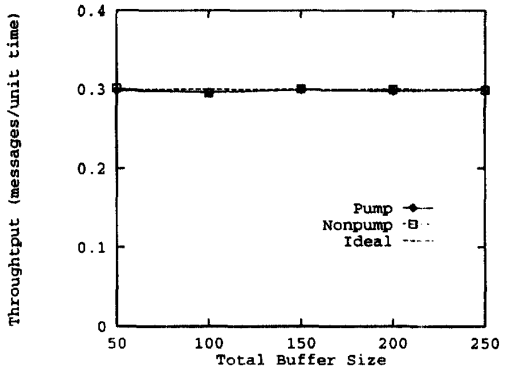


Figure 8: Throughput of session_{12}.

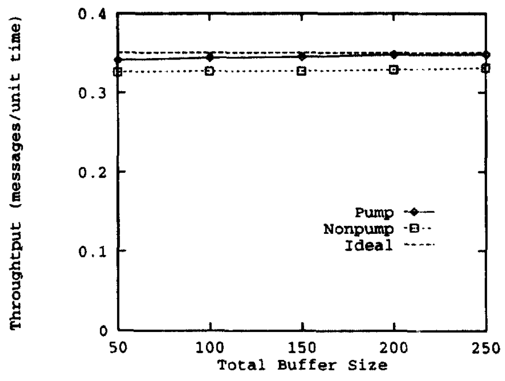


Figure 9: Throughput of session_{22}.

Since session_{12} has less input rate than it is entitled to (1/3), it achieves its demand rate as allocation rate. A little jitter in figure 8 is from the probablistic nature of the input rather than any effects from routing devices[6]. This probabilistic jitter slightly affects the throughput of other sessions (figures 9 and 10).

Figures 9 and 10 also show the effect of the scheduler, and the size of the output buffer and the Fair size to the fairness and throughput of each session, since 0.4 and 0.5 are both greater than 0.35. As the size of

[6]The rate of 0.3 is less that 1/3 (the max-min rate). The simulator never exactly generates a rate of 0.3, hence the jitter.

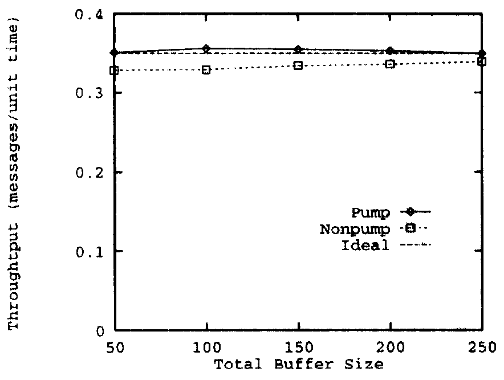


Figure 10: Throughput of $session_{32}$.

output buffer grows the throughput approaches to its ideal fairness rate.

Even though we do not show the performance of other sessions due to space limitations, the network Pump performs very well (basically the same as in figures 11-16).

4.3 Simulation Results under Denial of Service Attack

To show the effect of denial of service attack, we slow down the service rate (i.e., $\mu_{i2} = 0.1$) of one High, namely H_2. Figures 11 through 16 shows the performance and fairness comparison between the network Pump and the Nonpump. The performance of the network Pump is hardly affected by the attack. However, the performance of the Nonpump is greatly affected. The main reason for the degradation of performance is that all output buffers are occupied by sessions that send messages to H_2 so that the rest of sessions have to wait a long time to obtain them.

Figures 11, 12, and 13 show the throughputs of sessions to H_1.

These figures (11, 12, and 13) show no jitter of throughput as the size of buffer increases. This shows that the probablistic nature of inputs are all hidden because all input (demand) rates to H_1 are greater than its allocation rate and messages are always waiting for their turn at the output buffer[7] (the round-robin scheme takes a message from each buffer in turn and does not pass any buffer because they always have a

[7]For example a fluctuation around a rate of 0.5 is not significant when the (effective) allocated rate is actually much less than 0.5 .

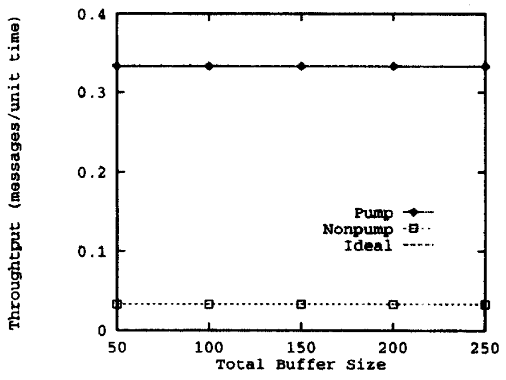


Figure 11: Throughput of $session_{11}$.

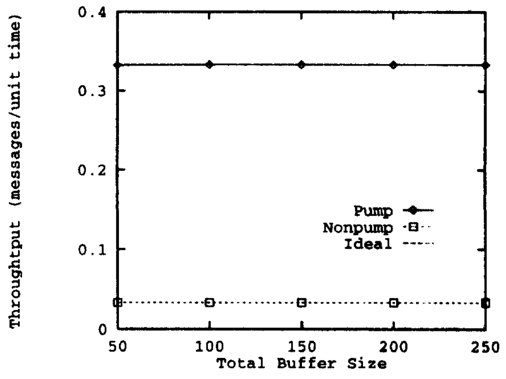


Figure 12: Throughput of $session_{21}$.

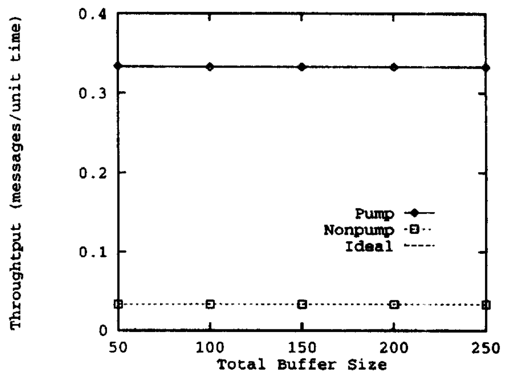


Figure 13: Throughput of $session_{31}$.

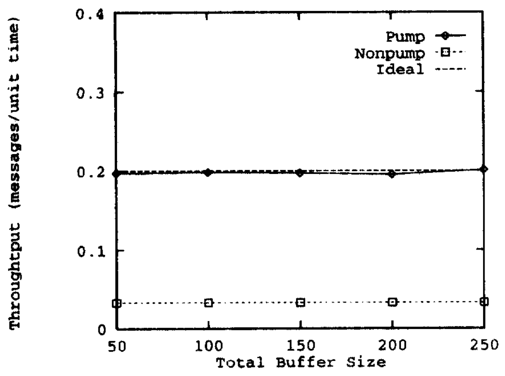


Figure 14: Throughput of session$_{13}$.

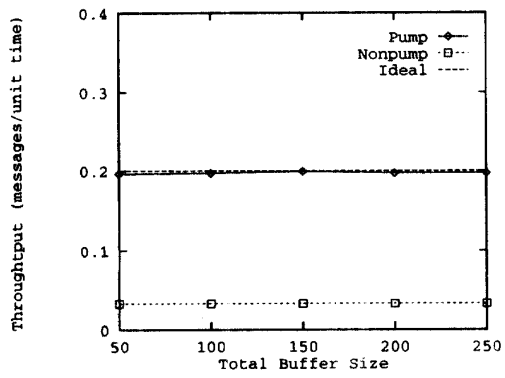


Figure 15: Throughput of session$_{23}$.

message ready to send).

Figures 14, 15, and 16 show the throughputs of different inputs to H_3. Again the jitter of throughput is from the probabilistic nature of inputs rather than the effect of different buffer sizes.

We do not show throughputs of session$_{12}$, session$_{22}$, session$_{32}$ because under the denial of service attack all cases have throughput values around 0.033.

5 Summary

This paper describes the need for a secure device that can route messages from (multiple) Lows to (multiple) Highs. Even though abstract composition problems have been well studied [6], this paper shows that the actual design of such a device is quite complicated. This

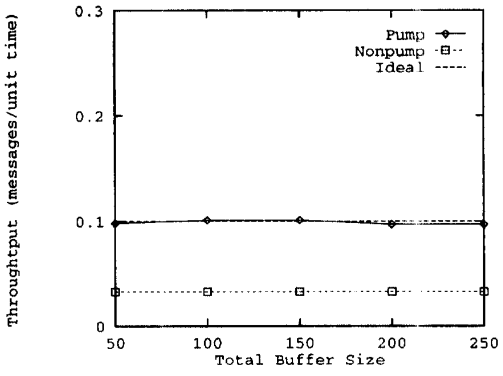


Figure 16: Throughput of session$_{33}$.

secure device should not only meet the requirements of conventional network routers such as performance, reliability, and fairness, but also the requirements of security, such as minimal impact from covert channels and denial of service attacks. The network Pump that was introduced in this paper can balance the above requirements.

This paper emphasizes the design and the rationale behind these design decisions. We also back our claims through the preliminary simulation results.

Our future plan includes designing and building the network Pump on top of the ATM layer.

References

[1] M. Gerla and L. Kleinrock. "Flow Control: A comparative survey," IEEE Trans. Commun., vol. 28, no. 4 pp. 553 - 574, Apr. 1980.

[2] E. L. Hahne. "Round-robin scheduling for max-min fairness in data networks," IEEE J. Select. Areas Commun., vol. 9 no. 7 pp. 1024 - 1039, Sep. 1991.

[3] M. H. Kang and I. S. Moskowitz. "A pump for rapid, reliable, secure communication," Proceedings ACM Conf. Computer & Commun. Security '93, pp. 119 - 129, Fairfax, VA, 1993.

[4] M. H. Kang and I. S. Moskowitz. "A data pump for communication," Submitted for publication.

[5] A. M. Law and W. D. Kelton. Simulation Modeling & Analysis. McGraw Hill, 1991.

[6] J. D. McLean. "A general theory of composition for trace sets closed under selective interleaving functions." Proceedings 1994 IEEE Computer Society Symposium on Research in Security and Privacy, pp. 79 - 93, Oakland, CA, 1994.

[7] A. R. Miller and I. S. Moskowitz. "Reduction of a class of Fox-Wright Psi functions for certain rational parameters," Computers & Mathematics with Applications. To appear.

[8] I. S. Moskowitz and A. R. Miller. "Simple timing channels," Proceedings 1994 IEEE Computer Society Symposium on Research in Security and Privacy, pp. 56 - 64, Oakland, CA, 1994.

[9] I. S. Moskowitz and M. H. Kang. "Discussion of a statistical channel," Proceedings IEEE-IMS Workshop on Information Theory and Statistics, p. 95, Alexandria, VA, 1994.

[10] I. S. Moskowitz and M. H. Kang. "Covert channels — Here to stay?," Proceedings COMPASS '94, pp. 235 - 243, Gaithersburg, MD, 1994.

[11] B. Mukherjee and S. Banerjee. "Alternative strategies for improving the fairness in and an analytical model of DQDB networks," Proceedings IEEE INFOCOM '91, pp. 879 - 888, 1991.

[12] R. M. Needham. "Denial of service: An example," Communications of the ACM, vol. 37 no. 11, pp. 42 - 46, 1994.

[13] C. Shannon and W. Weaver. The mathematical theory of communication. University of Illinois Press, 1949. Also appeared as a series of papers by Shannon in the Bell System Technical Journal, July 1948, October 1948 (A Mathematical Theory of Communication), January 1949 (Communication in the Presence of Noise).