Understanding and Mitigating the Effects of Denial-of-Service Attacks on Secure Gossip-Based Multicast

Amir Sasson

Understanding and Mitigating the Effects of Denial-of-Service Attacks on Secure Gossip-Based Multicast

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Amir Sasson

Submitted to the Senate of
the Technion — Israel Institute of Technology

Tevet 5764 Haifa January 2004

The research was done under the supervision of Dr. Idit Keidar in the department of Electrical Engineering

I wish to express my gratitude to my supervisor Dr. Idit Keidar for her guidance, support and help.
I would like to thank Gal Badishi and Aran Bergman for their help.

THE GENEROUS FINANICAL HELP OF THE TECHNION IS GRATEFULLY ACKNOWLEDGED

Contents

List of Figures

Abstract

One of the most devastating security threats faced by a distributed system is a *denial of service* (DoS) attack, in which an attacker makes a system unresponsive by forcing it to handle bogus requests that consume all available resources. In a *distributed denial of service* (DDoS) attack, the attacker utilizes several computers as the source of a DoS attack, in order to increase the attack strength. In 2003, approximately 42% of U.S. organizations, including government agencies, financial institutions, medical institutions and universities, were faced with DoS attacks [6]. That year, DoS attacks were the second most financially damaging attacks (65 million USD), only short of theft of proprietary information (70 million USD), and far above other attacks (0.07-27 million USD) [6]. Therefore, coping with DoS attacks is essential when deploying services in a hostile environment such as the Internet [28].

In this thesis, we propose a framework and methodology for quantifying the effect of DoS attacks on a distributed system. We present a systematic study of the resistance of gossip-based multicast protocols to DoS attacks. We show that even distributed and randomized gossip-based protocols, which eliminate single points of failure, do not necessarily eliminate vulnerabilities to DoS attacks. We propose Drum (DoS-Resistant Unforgeable Multicast) – a simple gossip-based multicast protocol that eliminates such vulnerabilities. We show, using asymptotic closed-form analysis ,simulations, and detailed analysis, that Drum survives severe DoS attacks, while traditional gossip-based protocols collapse even under mild attacks.

Chapter 1

Introduction

One of the most devastating security threats faced by a distributed system is a *denial of service* (DoS) attack, in which an attacker makes a system unresponsive by forcing it to handle bogus requests that consume all available resources. In a *distributed denial of service* (DDoS) attack, the attacker utilizes several computers as the source of a DoS attack, in order to increase the attack strength. In 2003, approximately 42% of U.S. organizations, including government agencies, financial institutions, medical institutions and universities, were faced with DoS attacks [6]. That year, DoS attacks were the second most financially damaging attacks (65 million USD), only short of theft of proprietary information (70 million USD), and far above other attacks (0.07-27 million USD) [6]. Therefore, coping with DoS attacks is essential when deploying services in a hostile environment such as the Internet [28].

As a first defense, one may protect a system against DoS attacks by using network-level mechanisms [4, 5, 11, 15, 16, 17, 32, 33]. However, network-level filters cannot detect DoS attacks at the application level, when the traffic seems legitimate. Therefore, even if means are in place to protect against network-level DoS, an attack can still be performed at the application level, as the bandwidth needed to perform such an attack is usually lower. This is especially true if the application performs intensive computations for each message, as occurs, e.g., with secure protocols based on digital signatures. In this thesis, we are concerned with DoS attacks on *secure* gossip-based multicast protocols (such as, e.g., Spinglass [3]), focusing only on the multicast protocol layer.

A DoS attack that targets every process in the system inevitably causes performance degradation, but also requires vast resources. In order to be effective even with limited resources, attackers target vulnerable parts of the system. For example, consider a tree-based multicast protocol; by targeting a single inner node in the tree, an attacker can effectively partition the tree and break up the multicast group. Hence, eliminating single points of failure is an essential step in constructing protocols that are less vulnerable to DoS attacks. We therefore focus on gossip-based (epidemic) multicast protocols [8, 2, 9, 12, 20, 21], which eliminate single points of failure using random choices. Such protocols are robust and have been shown to provide graceful degradation in the face of amounting failures [14]. One may expect that such a system will not suffer from vulnerabilities to DoS attacks, since it can continue to be effective even when many processes fail. Surprisingly, we show that traditional gossip-based multicast protocols are extremely vulnerable to DoS attacks targeted at a small subset of the processes in the system. We show this by quantifying the effects of various DoS attacks on the expected performance of such protocols, first using asymptotic analysis, and then verifying the asymptotic analysis through simulations and detailed mathematical analysis. Our results indicate that extra measures must be taken in order to eliminate common vulnerabilities to DoS attacks.

Having observed the vulnerabilities of traditional gossip-based multicast protocols, we turn to search for a protocol that will eliminate these vulnerabilities. We are familiar with only one previous work, by Minsky and Schneider [27], that deals with DoS attacks on a gossip-based protocol. However, the problem definition in [27] differs from ours in a way that renders their approach inapplicable to our setting (see Chapter 2), and moreover, they only deal with limited attack strengths.

We present a new protocol, *Drum* (DoS-Resistant Unforgeable Multicast), which, using a few simple ideas, eliminates the vulnerabilities of previous protocols, and withstands severe DoS attacks. We analyze and simulate Drum's performance, and compare it to that of traditional gossip-based protocols. We show that in contrast to existing gossip-based protocols, an attacker cannot substantially affect Drum's performance by targeting a small subset of the processes. E.g., under a DoS attack that focuses on 10% of the group members, the number of gossip rounds it takes to distribute a message to all the correct processes grows linearly with the strength of the attack in traditional protocols, whereas in Drum it remains constant.

More generally, we show that, asymptotically, when an adversary has a large sending capacity, its most effective attack against Drum is an all-out attack that distributes the attacking power as broadly as possible. We concentrate on heavy attacks since they are most damaging, and one can expect them to happen in actual scenarios [30]. Obviously, performance degradation due to a broad all-out DDoS attack is unavoidable for any multicast protocol, and indeed all the tested protocols exhibit the same performance degradation under such a broad attack. But while Drum performs much better under targeted attacks, the other protocols are significantly more vulnerable to focused DoS attacks with the same strength, and perform substantially worse under such attacks.

This thesis makes the following contributions:

- It presents a new framework and methodology for quantifying the effects of DoS attacks.
- It uses the new methodology to conduct the first systematic study of the effect of DoS attacks on multicast protocols. This study exposes vulnerabilities of traditional gossip-based multicast protocols.
- It presents Drum, a new gossip-based multicast protocol that eliminates these vulnerabilities.
- It provides asymptotic closed-form analysis as well as simulations and detailed formal analysis of the performance of several gossip-based multicast protocols under DoS attacks varying in strength and extent. Our detailed mathematical analysis and simulation results fully match each other, thus validating our simulation methodology. Our closed-form asymptotic analysis explains the trends observed in the simulations.

The rest of the thesis is organized as follows: Chapter 2 gives background on gossip-based multicast and related work. Chapter 3 overviews Drum's architecture and system model. Chapter 4 describes our DoS-resistant gossip-based multicast protocol. Chapter 5 presents our evaluation methodology and considered attack models. The following three chapters evaluate Drum and compare it to traditional gossip-based protocols using various tools: Chapter 6 gives closed-form asymptotic bounds; Chapter 7 provides a thorough evaluation using simulations; and Chapter 8 gives a detailed mathematical analysis. Chapter 9 concludes. The appendix contains several derivations for the asymptotic analysis.

Chapter 2

Background and Related Work

In recent years, gossip-based (or epidemic) dissemination [8] has emerged as a leading approach in the design of scalable reliable multicast protocols, e.g., [2, 9, 12, 20, 21], as well as other scalable services such as failure detection [31], group membership [7, 10], leader election [13], resource location [19], and aggregate data dissemination [14]. Gossip-based protocols are attractive for a number of reasons: (1) they scale well to thousands of processes; (2) they are reliable, and their reliability degrades gracefully as failures amount [22]; (3) they are adaptable, in that they support dynamic addition and removal of group members; and (4) they are easy to implement and deploy.

The first gossip-based protocol was presented by Demers et al. [8]. Birman et al. [2] offered a Scalable and reliable gossip-based multicast, under the assumption of an underlying multicast layer (such as IP multicast), and an external membership service. Our protocol is modeled after a gossip-based protocol that does not rely on external multicast protocols and that internally supports dynamic group membership, like the *Light-weight Probabilistic Broadcast (lpbcast)* protocol [9].

Gossip-based dissemination works roughly as follows: each process locally divides its time into *gossip rounds*. In each gossip round, the process randomly selects a small number of processes to gossip with, and exchanges information with them. The gossip rounds are not synchronized, and no acknowledgments are sent or awaited. Each piece of information is gossiped for a number of rounds. It has been shown that the propagation time of gossip protocols increases logarithmically with the number of processes [29, 18]. In this context, we identify two methods used for information dissemination:

1. *push-based*, in which the process sends messages to selected processes.

2. *pull-based*, in which the process requests messages from selected processes.

Both Methods are vulnerable to DoS attacks: Attacking the incoming push channels of a process may prevent it from receiving valid messages. Conversely, attacking the process's incoming pull channels may prevent it from sending messages to valid targets. Our new protocol, Drum utilizes both push and pull operations (as previously done in [8, 18]). But in contrast to these previous protocols, Drum independently allocates a bounded amount of resources for each operation (push and pull), in order to confine the impact of a DoS attack to one operation.

Secure gossip-based dissemination protocols that do not incorporate cryptographic primitives were suggested by Malkhi et al. [24, 25, 26]. However, they did not deal with DoS attacks. Follow-up work by Minsky and Schneider [27] suggested a pull-based protocol that can endure limited DoS attacks and Byzantine failures. However, the problem and model considered in [24, 25, 26, 27] differ from ours in two substantial ways: First, these papers solve the *diffusion* problem, in which each message simultaneously originates at more than t correct processes, where t is the number of processes that may suffer Byzantine failures. In contrast, we consider a multicast system where a message originates at one source. Hence, using a pull-based solution as suggested in [27] does not help in withstanding DoS attacks. Second, they do not use digital signatures, but rather assume only pair-wise authenticated channels.

Network-level DoS mitigation has been extensively dealt with both by the academic community [4, 11, 11, 15, 16, 17, 32, 33] and by the industry [5], but DoS-resistance at the secure multicast service layer has gotten little attention. Limiting resource usage for a given operation is an effective mechanism in defending against DoS attacks. For example, Minsky and Schneider [27] suggest to bound the number of accepted requests per round in a pull-based protocol in order to resist DoS attacks. However, their work focuses on load rather than DoS attacks; it includes only a brief analysis of DoS attacks, under the assumption that no more than t processes perform the attack, and that each of them generates a single message per round (the reception bound is also assumed to be one message per round). In contrast, we focus on substantially more severe attacks. Separating resources so as not to affect the performance of orthogonal operations under a DoS attack is also used in COCA [35], for the sake of overcoming DoS attacks on authentication servers.

DoS can also be caused by churn, where processes rapidly join and leave [23], thus reducing availability. In Drum, as in other gossip-based protocols, churn has little effect on availability: even when as many as half the processes fail, such protocols can continue to deliver messages reliably and with good quality of service [22]. A DoS attack of another form can be caused by process perturbations, whereby some processes are intermittently unresponsive. The effect of perturbations is analyzed in [2], where it is shown that probabilistic protocols, e.g., gossip-based protocols, solve this problem. This thesis focuses on DoS attacks in which the attacker sends fabricated application messages. We note that our work is the first that we know of that conducts a systematic study of the effect of DoS attacks on message latency.

Chapter 3

System Model and Architecture

We assume that the underlying network is fully-connected, meaning that each process can directly communicate with every other process in the system. There are no bounds on message delays. Hence, the communication is asynchronous. The communication channels are insecure, meaning that the source of an incoming message cannot be verified. The loss rate on the communication links is bounded and uniform.

Drum supports reliable multicast among processes that are members of a group. Each message is created by exactly one group member (its *source*). The processes use asymmetric cryptography in order to sign messages (e.g., as was used for secure multicast in Spinglass [3]). The processes' public keys (PKs) are certified by a collection of servers, a threshold of which may suffer Byzantine failures, similar to the COCA certification authority [35]. The servers do not participate in the multicast operation among the group members. Servers and processes have clocks that are synchronized within some known bound. The servers are described in detail in [1], and their discussion is beyond the scope of this thesis.

Digital signatures are computationally unforgeable without some knowledge of the respective private key. An adversary is capable of generating fabricated messages and snooping on messages. However, these operations requires the adversary to utilize resources, thus limiting the adversary's capacity for performing malicious operations.

3.1 Drum's Dynamic Membership Protocol

Drum incorporates a dynamic membership protocol, in order to allow processes to join and leave the group at will. The membership protocol is layered on top of Drum's multicast protocol, similar to the way lpbcast's membership protocol is layered on top of lpbcast's multicast protocol [9]. Thus, the membership protocol inherits the properties of Drum's multicast protocol, e.g., scalability, reliability, data integrity and DoS resistance.

In order to join the group, a process must be authorized by the servers. Once the servers authorize the process according to its credentials, they use threshold cryptography to collectively generate a certificate to attest to the authenticity and validity of that process' public key. This public key will be used by the other processes to verify digital signatures of that process. The servers provide the newcomer with an initial (not necessarily complete) list of PKs of the other processes in the group, and also propagate, using gossip, a *log-in* message to the other processes, containing the newly issued certificate.

Whenever a process wishes to log out of the group, it sends a *log-out* message to the servers, which in turn revoke that process' certificate and forward the *log-out* message to the other processes in the group. The servers may also revoke a process' certificate due to suspicion of malbehavior. In this case, an appropriate message is sent to the processes, in order to remove the process from the group.

The certificates expire after a certain amount of time. When a process' certificate is about to expire, the process must request a new certificate from the servers, or be effectively removed from the group (here we are using the bounded clock drift assumption).

Additionally, processes may suffer benign failures, or long delays that affect their responsiveness. In order to prevent situations in which processes try to spread their data through unresponsive processes, a failure detection mechanism can be used. From time to time, each process tests the responsiveness of the other processes it communicates with. If a failure is detected, the process stops communicating with the failed process, but does not propagate this information to other processes. Note, however, that this does not affect the status of the slow/failed process as a group member, as it is possible that the process can communicate with other processes.

Each process piggybacks its certificate on top of its data message if it has recently joined the system, or if it has not created messages for a relatively long period. This way, processes that do not have a complete membership information database will be able to authenticate new messages and complete the database.

For the rest of this thesis we will assume that processes can verify signatures of all incoming valid messages.

Chapter 4

DoS Resistant Gossip Algorithm

Drum is a simple gossip protocol, which achieves DoS-resistance using a combination of pull and push operations, separate resource bounds for different operations, and the use of random ports in order to reduce the chance of a port being attacked. Each process, p, locally divides its time into rounds. The rounds are not synchronized among the processes. A round is typically in the order of a second, and its duration may vary according to local random choices. Process p holds a list of other processes in the group (denoted *MembList*), and their corresponding public keys (this list is-maintained by the membership service). Every round, p chooses two small (constant size) random sets of processes from this list, $view_{push}$ and $view_{pull}$, and gossips with them. E.g., when these views consist of two processes each, this corresponds to a combined fan-out of four. In addition, p maintains an application messages buffer (denoted *MsgBuff*), contains application messages created by p's local application or received from other processes. All application messages are signed by their source (creator) using its own private key and are delivered to the application without any other signatures.

4.1 Operations occurring in a round

Process p performs the following operations in each round:

- *Push* operation – p randomly picks messages from its buffer, and sends them to the processes in its $view_{push}$. There is a well-known port on which the target process listens for push messages.

- *Pull-request* operation – p sends a digest of the messages it has received to the processes in its $view_{pull}$, requesting missing messages. Pull-request messages are sent to a well-known port. Each pull-request specifies a randomly selected port on which responses will be awaited. Once the request is sent, p is listening on the chosen port for a few rounds.

- *Pull-reply* operation – in response to pull-request messages arriving on the well-known port, p randomly selects messages that it has and are missing from the received digests, and sends them to the destinations indicated in the requests.

The random ports transmitted during the pull operation are encrypted using the recipient's public key, in order to prevent an adversary from discovering this random port. Thus, $|view_{pull}|$ encryptions are performed each time these ports are changed (every round).

Upon receiving a new data message, either by push or in response to a pull-request, p first performs some sanity checks. If the message passes these checks, p validates the source's signature. If the signature is valid, p delivers the message to the application and saves it in its *MsgBuff* for a number of rounds; after a while, the message is discarded.

4.2 Resource allocation and bounds

In order to achieve balance in the use of resources, p's capability for sending in each round is equally divided between sending push messages and sending messages in response to pull requests. Likewise, if more data messages than p can handle arrive, then p's capability for processing incoming data messages is equally divided between messages that arrive on the push port and those arriving on the open pull–request ports.

At the end of each round, p discards all unread messages from its incoming message buffers. This is important, especially in the presence of DoS attacks, as an attacker can send more messages than p can handle in a round. Since rounds are locally controlled and randomly vary in duration, the attacker cannot “aim” its messages for the beginning of a round. Thus, a bogus message has an equal likelihood of being discarded at the end of the round as an authentic messages does.

4.3 Mitigating DoS attacks

We now explain how the combination of push, pull, random ports, and resource bounds achieves resistance to DoS attacks. Since there is a bound on the number of messages accepted on each port in a round, the probability of receiving a valid message from a specific host in a given round is inversely proportional to the total number of messages arriving to the same port in that round. An attacking strategy might then be to flood the port with as many messages as possible, in order to decrease the probability of a valid message being accepted at that port.

In order to prevent a process from sending its messages using a push operation, one must attack the intended push recipients. However, the push destinations are randomly chosen in each round and the attacker has no way of predicting these choices.

In order to prevent a process from receiving messages during a pull operation, a DoS attack needs to target the destination of the pull-request or the port on which pull-replies arrive. However, the destinations are randomly chosen, and the random port used for data reception is encrypted, and thus, an attacker cannot effectively prevent a process from receiving messages via the pull operation. Thus, using the push operation, Drum achieves resilience to targeted attacks aimed at preventing a process from *sending* messages, and using the pull operation, Drum can withstand targeted attacks whose purpose is to prevent a process from *receiving* messages. By using a combination of these operations, Drum is able to efficiently propagate messages in the group, even under very severe DoS attacks.

Chapter 5

Evaluation Methodology

The next three sections present an evaluation of the following protocols:

- *Drum*.
- *Push*, which uses only the push operations.
- *Pull*, which uses only the pull operations.

Push and Pull are implemented the same way Drum is, with the important measures of bounding the number of messages accepted in each round, and in Pull, also using random ports. Thus, in comparing the three protocols, we study the effectiveness of combining push and pull operations under the assumption that these other measures are used.

The protocols use a constant fan-out, F. That is, in Push each process sends push messages to F processes ($view_{push} = F$) and accepts push messages from at most F processes in a round, and in Pull, each process sends pull-request messages to F processes ($view_{pull} = F$) and accepts pull-request messages from at most F processes. In Drum, the fan-out is equally divided between the push and pull operations, hence, $view_{push} = view_{pull} = \frac{F}{2}$. In Chapter 6, we assume that Drum's bounds on the number of processes to receive messages from in a round are tested jointly for push and pull operations. That is, a processes discards incoming messages only if it receives messages (push and pull-request together) from more than F processes in a round. In chapters 7 and 8, we give a more strict assumption, where Drum's bounds for each operation are tested separately. That is, each process accepts push messages from at most $\frac{F}{2}$ processes and pull-request messages from at most $\frac{F}{2}$ processes in a round.

As in [2, 9], we consider a system composed of a static set of *n* processes, and the propagation of a single message *M*, initiated by a single process referred to as the *message source*. Note that processes send push and pull-request messages in every round, regardless of whether or not they have *M*. Hence, in Push (and in Drum) a process sends push messages even if it does not have *M*, in Pull (and in Drum) a process sends pull-request messages even if it already received *M*. The adversary attacks a fraction α of the processes ($0 < \alpha \leq 1$), where each process is attacked with x fabricated messages per round (in Drum this means that each process is attacked with $\frac{x}{2}$ push messages and $\frac{x}{2}$ pull-requests). We denote the total attack strength by $B = x \cdot \alpha n$. We assume that the message source is being attacked (this has no impact on the results of the push protocol). We quantify the protocols' performance by measuring their *propagation time* in various scenarios, where the propagation time of a given protocol in a given setting is defined to be the expected number of rounds it takes that protocol to propagate *M* to all (in the closed-form analysis) or to 99% of the correct processes in the system (in the simulations and detailed analysis). We chose a threshold of 99% since under DoS attacks the random algorithm does not guarantee that *M* will reach all the correct processes. Note that correct processes can be either attacked or non-attacked. In both cases, they should be able to send and receive messages.

We assume that the protocol progresses in synchronous rounds, and that the message-delivery latency is smaller than half the gossip period. The latter assumption simplifies the analysis of the pull operation, since this way a process that sends a pull-request message receives the pull-reply message in the same round. We also assume that every process has complete knowledge of all the other processes in the system. All of these assumptions were used in previous analyses of gossip-based protocols, e.g., [2, 9, 24, 27].

We examine all three protocols under various DoS attacks. We evaluate the protocols under *targeted attacks*, where the adversary attacks αn processes ($0 < \alpha < 1$) uniformly. We consider either attacks of *fixed strength*, where B is fixed and α increases (thus, x decreases); or attacks of *increasing strength*, where either x is fixed and α increases, or vice versa (in both cases, B increases). Examining fixed strength attacks allow us to identify protocol vulnerabilities, e.g., if an adversary can benefit from targeting a subset of the processes. Increasing strength attacks enable us to evaluate the performance degradation due to increasing attack intensity. We also consider *broad attacks*, where the adversary attacks all correct processes uniformly..

In the simulations and detailed analysis, we also evaluate the protocols under a *silent attack*, where a subset of the processes are faulty (silent) in the sense that they do not send valid any messages, but also do not engage in a DoS attack.

Chapter 6

Asymptotic Closed-Form Analysis

In this section, we assume that all the processes are correct and the DoS attack is launched from outside the system.

We denote by p_u the probability of a non-attacked process to accept a valid push or pull-request message sent to it. Similarly, We denote by p_a the probability of an attacked process to accept a valid push or pull-request message sent to it. Obviously, p_u is independent of the attack strength. In Appendix A, we give detailed formulas for p_a and p_u, and show that $p_u > 0.6$ for all $F \geq 3$. Since an attacked process is sent at least x messages in a round, and accepts at most F of them, we get the following coarse bound: $p_a < \frac{F}{x}$.

6.1 Drum

We define the *effective expected fan-in* (I) to be the average number of valid data messages a process accepts in a round. (If the same data message is received from k processes, we count this as k messages.) Likewise, the *effective expected fan-out* (O) is the average number of messages that a process sends and are accepted by their targets in a round.

Let us examine the effect of a DoS attack on O and I, with respect to the push operation (O_{push} and I_{push}, resp.). The probability of an attacked process to accept a valid push message sent to it is p_a. The probability of a non-attacked process to accept a valid push message sent to it is p_u. The expected number of processes that choose a given process is F. Therefore, the effective fan-ins I^a_{push} and I^u_{push} of an attacked and non-attacked process (resp.) are:

$$I^a_{push} = F \cdot p_a \quad \text{and} \quad I^u_{push} = F \cdot p_u \tag{6.1}$$

When αn processes are attacked, the effective fan-outs are:

$$O^a_{push} = O^u_{push} = F \cdot (\alpha \cdot p_a + (1 - \alpha) \cdot p_u) \tag{6.2}$$

A similar analysis for the pull operation, under the assumption that the pull-reply's random ports are resilient to DoS attacks, yields the following effective fan-ins and fan-outs:

$$I^a_{pull} = \quad I^u_{pull} = \quad F \cdot (\alpha \cdot p_a + (1 - \alpha) \cdot p_u) \tag{6.3}$$

$$O^a_{pull} = \quad F \cdot p_a \quad \text{and} \quad O^u_{pull} = F \cdot p_u \tag{6.4}$$

In Drum, $O = \frac{1}{2}(O_{push} + O_{pull})$ and $I = \frac{1}{2}(I_{push} + I_{pull})$. Therefore:

$$O^a = I^a = \quad \tfrac{F}{2} \cdot (\alpha \cdot p_a + (1 - \alpha)p_u + p_a) \quad = F \cdot (\frac{\alpha + 1}{2} \cdot p_a + \frac{1 - \alpha}{2} \cdot p_u) \tag{6.5}$$

$$O^u = I^u = \quad \tfrac{F}{2} \cdot (\alpha \cdot p_a + (1 - \alpha)p_u + p_u) \quad = F \cdot (\frac{\alpha}{2} \cdot p_a + \frac{2 - \alpha}{2} \cdot p_u) \tag{6.6}$$

6.1.1 Targeted attacks

Lemma 1. *Fix α (α ¡ 1) and n. Drum's expected propagation time is bounded from above by a constant independent of x.*

Proof. From Equations (6.5) and (6.6) we get that for all x, $O^a = I^a > F \cdot \frac{1-\alpha}{2} \cdot p_u$, and $O^u = I^u > F \cdot \frac{2-\alpha}{2} \cdot p_u$. Since p_u is independent of x, the effective fan-ins and fan-outs of *all* the processes are bounded from below by a constant independent of x. Therefore, the propagation time is inevitably bounded from above by a constant independent of x (In [29], Pittel has shown that for a fixed fan–out of F, the propagation time of a gossip-based protocol is $ln(N)(\frac{1}{F} + \frac{1}{ln(F+1)})$). □

Figure 7.4(a) in Chapter 7.2.1 illustrates this quality of Drum.

Note that I^a_{push} and O^a_{pull} cannot be bounded from below from a positive constant, since they go to 0 when x goes to infinity. Therefore, we cannot use the argument above to show that the propagation times of Push and Pull are bounded independent of x. Indeed, in the next two sections we will show that the propagation times of both Push and Pull increase linearly with x.

6.1.2 Adversary Strategies

Suppose the adversary has a given attack strength. There are many ways for it to divide its strength among the correct processes: it can attack every process in every round with fairly few messages, or attack fairly few processes with a large number of messages. But which strategy will cause the most damage? In order to answer this question, we evaluate Drum under attacks where the adversary has a fixed attacking power. We denote by $c = \frac{B}{F \cdot n}$ the attack strength divided by the total system capacity. We assume that the attack strength $B = x \cdot \alpha n$ is fixed, and consider different choices of x and α.

Lemma 2. *For $c > \frac{3}{p_u}$, Drum's expected propagation time is monotonically increasing with α.*

Proof. We will show that all the processes' effective fan-ins and fan-outs are monotonically decreasing with α. That is, we want to prove that: $\frac{\mathrm{d}O^a}{\mathrm{d}\alpha} < 0$ and $\frac{\mathrm{d}O^u}{\mathrm{d}\alpha} < 0$. By deriving equation 6.5 according to α, we see that we require the following:

$$\frac{\mathrm{d}O^a}{\mathrm{d}\alpha} = \frac{F}{2} \cdot \left(p_a + \alpha \frac{\mathrm{d}p_a}{\mathrm{d}\alpha} + \frac{\mathrm{d}p_a}{\mathrm{d}\alpha} - p_u\right) < 0$$
$$p_a + (\alpha + 1)\frac{\mathrm{d}p_a}{\mathrm{d}\alpha} < p_u$$

Note that p_a depends on α, because p_a depends on x, which for a fixed attack strength decreases as α increases. The probability p_u does not depend on α. In Appendix A we show that $\frac{\mathrm{d}p_a}{\mathrm{d}\alpha} < \frac{F}{\alpha x}$. Recall that $\frac{F}{\alpha x} = \frac{F \cdot n}{B} = \frac{1}{c}$ and $p_a < \frac{F}{x}$. Bounding the left side of the inequality, we get:

$$p_a + (\alpha + 1)\frac{\mathrm{d}p_a}{\mathrm{d}\alpha} < \frac{F}{x} + (\alpha + 1)\frac{F}{\alpha x} = \frac{F}{\alpha x} \cdot (\alpha + \alpha + 1) = \frac{2\alpha + 1}{c} \leq \frac{3}{c}$$

Thus, our condition holds when $\frac{3}{c} < p_u$, that is, when $c > \frac{3}{p_u}$. Similarly, by deriving equation 6.6 we get the condition:

$$\frac{\mathrm{d}O^u}{\mathrm{d}\alpha} = \frac{F}{2} \cdot \left(p_a + \alpha \frac{\mathrm{d}p_a}{\mathrm{d}\alpha} - p_u\right) < 0$$

$$p_a + \alpha \frac{\mathrm{d}p_a}{\mathrm{d}\alpha} < p_u$$

Bounding the left side of the inequality, we get:

$$p_a + \alpha \frac{\mathrm{d}p_a}{\mathrm{d}\alpha} < \frac{F}{x} + \alpha \frac{F}{\alpha x} = \frac{F}{\alpha x} \cdot (\alpha + \alpha) = \frac{2\alpha}{c} \leq \frac{2}{c}$$

Thus, we require that $\frac{2}{c} < p_u$, or that $c > \frac{2}{p_u}$. This is already inferred from our previous result. □

Corollary 1. *For $c > 5$, The most effective attack against Drum is a broad one.*

Proof. Since $p_u > 0.6$ (see Appendix A), for $c > \frac{3}{p_u} > 5$, Drum's expected propagation time is monotonically increasing with α. □

This behavior is validated in the simulations. Moreover, the simulations show that even for much smaller values of c (ranging from 0.25 to 2), Drum's propagation time either increases with α or is virtually unaffected by the choice of the adversary strategy (see figures 7.10–7.11).

6.2 Push

In this section we prove that unlike Drum, Push's propagation time under targeted attacks increases linearly with x. We first prove the following simple lemma.

Lemma 3. $\forall a > 0 \quad a < \frac{1}{ln(1+\frac{1}{a})} < a + 1$.

Proof. We show that $\forall y > 0 \quad \frac{1}{y} < \frac{1}{ln(1+y)} < \frac{1}{y} + 1$.
Define $h(y) = ln(1+y) - \frac{y}{1+y}$ and $g(y) = ln(1+y) - y$. By taking derivatives we get:

$$h'(y) = \frac{1}{1+y} - \left(\frac{1}{1+y} - \frac{y}{(y+1)^2}\right) = \frac{y}{(y+1)^2} > 0, \quad \forall y > 0,$$

$$g'(y) = \frac{1}{1+y} - 1 < 0, \quad \forall y > 0.$$

Since $h(0) = g(0) = 0$, $y > ln(1 + y) > \frac{y}{(y+1)}$. Therefore, $\frac{1}{y} < \frac{1}{ln(1+y)} < \frac{1}{y} + 1$. $\square$

We proceed to show that Push's propagation time is linear in x.

Lemma 4. *The expected propagation time to all processes in Push is bounded from below by:*

$$k_u + \frac{ln\left(\frac{n}{(1+F((1-\alpha)p_u+\alpha p_a)^{k_u}}\right)}{ln(1+F\alpha p_a)}, \text{ where } k_u = \frac{ln(1+\frac{n}{p_u}((1-\alpha)p_u+\alpha p_a))}{ln(1+F((1-\alpha)p_u+\alpha p_a))}$$

Proof. Let $M(k)$ denote the expected number of processes that have M at the beginning of round k. Let $U(k)$ and $A(k)$ denote the expected number of non-attacked and attacked processes (resp.) that have M at the beginning of round k. Thus, $M(k) = A(k) + U(k)$. In round k, each process that has M sends it to F other processes. On average, $F \cdot \alpha$ (resp., $F \cdot (1-\alpha)$) of those are attacked (resp., non-attacked), and each attacked process receives the message with probability p_a. Thus, we get the following recursive bounds:

$$U(k+1) \leq U(k) + M(k) \cdot F \cdot (1-\alpha) \cdot p_u \tag{6.7}$$

$$A(k+1) \leq A(k) + M(k) \cdot F \cdot \alpha \cdot p_a \tag{6.8}$$

Since the source is attacked, $U(0) = 0$ and $A(0) = 1$. Moreover, $U(k) \leq (1-\alpha)n$ and $A(k) \leq \alpha n$. We now solve the above set of bounded difference equations. We denote $F_u = F \cdot (1-\alpha) \cdot p_u$, and $F_a = F \cdot \alpha \cdot p_a$. By adding Equations (6.7) and (6.8) we get:

$$M(k) \leq (1 + F_u + F_a)^k$$

By substituting this expression for $M(k)$ in Equations (6.7) and (6.8) we get:

$$U(k+1) \leq U(k) + F_u(1 + F_u + F_a)^k, \quad \text{and}$$

$$A(k+1) \leq A(k) + F_a(1 + F_u + F_a)^k$$

We first solve these recursive equations without any bounds, and get :

$$U(k) \leq U(0) + F_u \sum_{i=0}^{k-1} (1 + F_u + F_a)^i, \quad \text{and}$$

$$A(k) \leq A(0) + F_a \sum_{i=0}^{k-1} (1 + F_u + F_a)^i$$

Next, we compute the smallest k's that maintain the bounds, and get:

$$k_u \geq \frac{ln(1 + \frac{n(1-\alpha)}{F_u}(F_u + F_a))}{ln(1 + F_u + F_a)}, \quad \text{and}$$

$$k_a \geq \frac{ln(1 + \frac{(\alpha n - 1)(F_u + F_a)}{F_a})}{ln(1 + F_u + F_a)}$$

$k_u < k_a$ for all $\alpha > \frac{p_u}{n(p_u - p_a)}$. Since p_a decreases linearly with x and p_u is constant, asymptoticly $\frac{p_u}{n(p_u - p_a)} \approx \frac{1}{n}$. Hence, $U(k)$ hits the bound first. Therefore, at that time, $U(k_u) = (1 - \alpha)n$ and $M(k_u)$ is at most $(1 + F_u + F_a)^{k_u}$. Since we give a lower bound on the propagation time, we consider $M(k_u) = (1 + F_u + F_a)^{k_u}$. Next, we compute the expected number of rounds after k_u to propagate M to all the processes, by solving the following difference equations:

$$M(k^{'} + 1) \leq M(k^{'}) + M(k^{'})F_a, \quad \text{and}$$

$$M(0) = (1 + F_u + F_a)^{k_u}$$

For $M(k^{'}) = n$, we have that

$$k^{'} \geq \frac{ln\left(\frac{n}{(1+F_u+F_a)^{k_u}}\right)}{ln(1 + F_a)}$$

Thus, the propagation time to all processes is

$$k \geq k_u + k^{'} = k_u + \frac{ln\left(\frac{n}{(1+F_u+F_a)^{k_u}}\right)}{ln(1 + F_a)}, \quad \text{where } k_u = \frac{ln(1 + \frac{n(1-\alpha)}{F_u}(F_u + F_a))}{ln(1 + F_u + F_a)}$$

$\square$

Corollary 2. *Fix α and n. The propagation time of Push increases at least linearly with x.*

Proof. For a fixed $y > 0$ and $b \geq 1$, $p_a \to 0$ as $x \to \infty$, and hence $ln(b + y \cdot p_a) = \Theta(1)$. Therefore, k_u and $ln\left(\frac{n}{(1+F((1-\alpha)p_u+\alpha p_a)^{k_u}}\right)$ are constant with x. Since $p_a < \frac{F}{x}$, it holds that $F \cdot \alpha \cdot p_a = O(\frac{1}{x})$. The corollary follows since, by Lemma 3, $\frac{1}{ln(1+\frac{1}{x})} = \theta(x)$. $\square$

The above corollary explains the trend exhibited by Push in Figure 7.4(a).

6.3 Pull

In this section we prove that under targeted attacks, Pull's propagation time increases linearly with x.

Lemma 5. *Fix α and n. The propagation time of Pull grows at least linearly with x.*

Proof. Let $\widetilde{p}$ denote the probability of propagating a message beyond the source in a round. In Appendix B, we give the formula for computing $\widetilde{p}$. Let Y be the number of correct processes that choose the source in a round. Y is binomially distributed, with $N = n-1$ and $p = \frac{F}{n-1}$. According to Chernoff Bound: for $X \sim B(N,p)$, $\Pr(X \geq m) \leq (\frac{Np}{m})^m \cdot e^{m-Np}$, for any $m > Np$. With $N = n - 1$, and $M = 10F + 1$, we get that $\Pr(Y \geq 10F + 1) \leq (\frac{F}{10F+1})^{10F+1} e^{9F+1}$. Note that $\Pr(Y \geq 10F + 1)$ decreases with F. For $F \geq 4$, the probability that more than $10F$ other correct processes choose the source in a round is less than $4.26{\cdot}10^{-26}$, hence, this probability can be neglected. Here, we give a very gross over-estimate of $\widetilde{p}$ by assuming that exactly $10F$ other correct processes choose the source in every round. (When fewer processes choose the source, *M* is *less* likely to leave the source.) Since $p_a < \frac{F}{x}$, $\widetilde{p} < (1 - (\frac{x-F}{x})^{10F})$. The number of rounds it takes to propagate a message beyond the message source is geometrically distributed with $\widetilde{p}$. Therefore, its expectation is $\frac{1}{\widetilde{p}} > \frac{x^{10F}}{x^{10F}-(x-F)^{10F}}$. In Lemma 6, we show that $\frac{x^{10F}}{x^{10F}-(x-F)^{10F}} = \Theta(x)$. Therefore, $\frac{1}{\widetilde{p}} = \Omega(x)$. $\square$

Figure 7.4(a) illustrates this behavior of Pull.

Lemma 6. $\frac{x^{10F}}{x^{10F}-(x-F)^{10F}} = \Theta(x)$.

Proof. We first show that $\frac{a-1}{b} < \frac{a^b}{a^b-(a-1)^b} \leq \frac{a-1}{b} + 1$ for every $a > 1$, $b \in N$.

In order to prove the left inequality, we prove by induction on b that $\frac{b}{a-1} > \frac{a^b-(a-1)^b}{a^b}$.

For b=1, $\frac{1}{a-1} > \frac{1}{a}$ for every $a > 1$.

Induction Hypothesis: $\frac{b}{a-1} > \frac{a^b-(a-1)^b}{a^b}$.

The inductive Step:

$$\frac{a^{b+1}-(a-1)^{b+1}}{a^{b+1}} = \frac{a(a)^b-(a-1)(a-1)^b}{a(a)^b} =$$

$$\frac{a^b}{a(a)^b} + \frac{a-1}{a}\frac{a^b-(a-1)^b}{a^b} <$$

$$\frac{1}{a} + \frac{a-1}{a}\frac{b}{a-1} = \frac{1}{a} + \frac{b}{a} = \frac{b+1}{a} < \frac{b+1}{a-1}.$$

Next, we prove that $\frac{a^b}{a^b-(a-1)^b} \leq \frac{a-1}{b} + 1$, i.e., $\frac{(a-1)^b}{a^b-(a-1)^b} \leq \frac{a-1}{b}$. From the inequality $(a)^b \leq (a-1)^b + b(a-1)^{(b-1)}$, we get that $a^b - (a-1)^b \leq b(a-1)^{(b-1)}$. Hence, $\frac{(a-1)^b}{a^b-(a-1)^b} \leq \frac{(a-1)^b}{b(a-1)^{b-1}} = \frac{a-1}{b}$.

Therefore, $\frac{a-1}{b} < \frac{a^b}{a^b-(a-1)^b} \leq \frac{a-1}{b} + 1$. Now, by substituting $\frac{x}{F}$ for a and $10F$ for b in the latter inequality, we get that $\frac{x-F}{10F^2} < \frac{x^{10F}}{x^{10F}-(F+x)^{10F}} \leq \frac{x-F}{10F^2} + 1$ for every $x > F$. Therefore, $\frac{x^{10F}}{x^{10F}-(x-F)^{10F}} = \Theta(x)$. $\square$

Chapter 7

Simulation Results

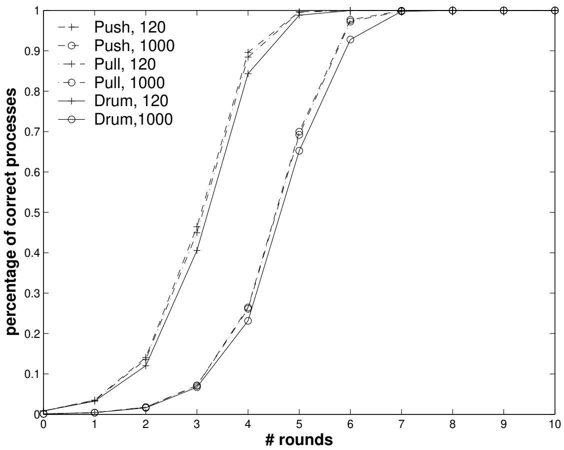
This chapter presents a simulation of the three protocols. We first simulate the protocols in the absence of DoS attacks and show that the measures taken in Drum have little impact when there are no DoS attack. We then simulate the protocols under various DoS attack scenarios. In the simulation the system size was varied from 120 to 1000 processes. The simulation is conducted in *MATLAB*, and the entire system is simulated in a single process. In this chapter we consider a loss rate of 0.01 on all links. We fix $F = 4$. In the next chapter we present a formal detailed analysis of the protocols, and show that our analysis results match our simulation results, thus validating our methodology. Each data point in the simulation is averaged over 1000 runs.

7.1 Without DoS Attacks

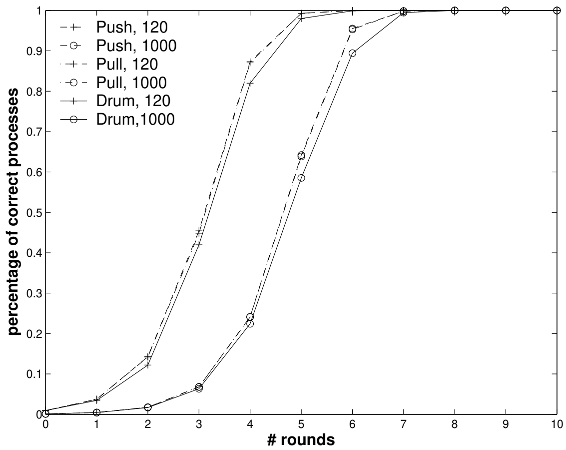
In this section we consider two scenarios:

- *No Attack*, in which there are no faulty processes.
- *Silent Attack*, where a subset of the processes are faulty (silent) in the sense that they do not send any messages.

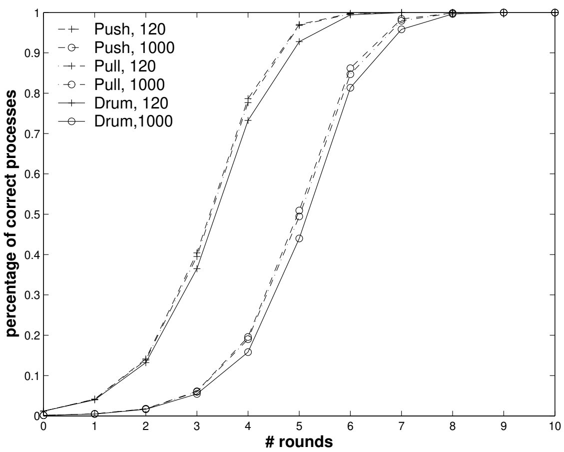
We examine the cumulative distribution function (*CDF*) of the percentage of correct processes that receive *M* by a given round. Figure 7.1 illustrates the CDF, under no attack (Figure 7.1 (a)) and silent attacks with 10%–50% faulty processes (Figure 7.1 (b)–(d)), in systems composed of 120 and 1000 processes ($n = 120$, and $n = 1000$, respectively). Figure 7.2 shows the average propagation time to 99% of the correct processes with a log–scale x–axis under the same attacks, for different system sizes. Both figures show that the behavior of the three protocols is almost identical. That is, Drum performs as well as the other protocols in the absence of DoS attacks. Figure 7.2 shows that the protocols' propagation times are logarithmic in the number of processes. This confirms previous analyses of gossip protocols [29, 18], which have shown that gossip protocols' propagation times increase logarithmically with the number of processes.



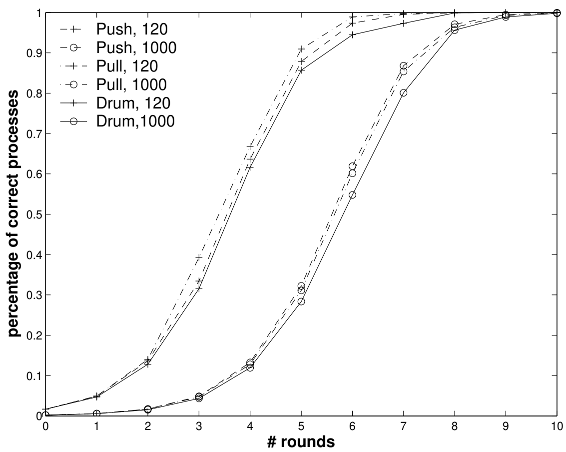
(a) No attack.



(b) Silent attack, 10% are silent.

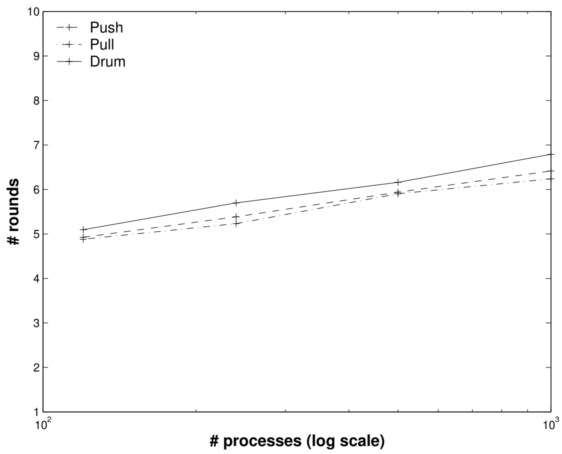


(c) Silent attack, 30% are silent.

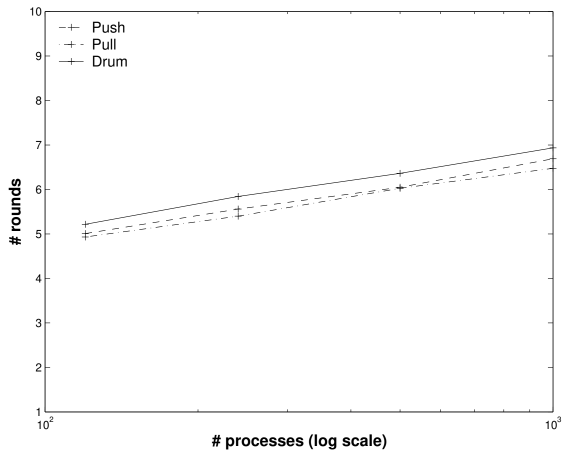


(d) Silent attack, 50% are silent.

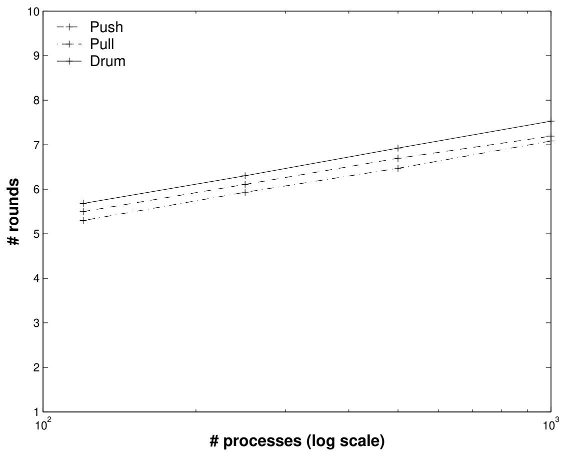
Figure 7.1: CDFs of percentage of correct processes that receive M, $n = 120, 1000$.



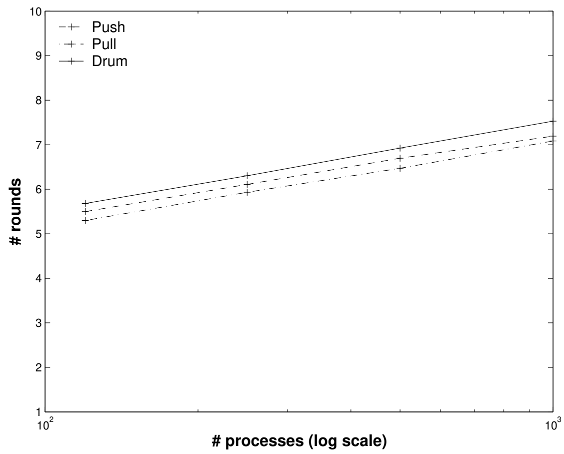
(a) No attack.



(b) Silent attack, 10% are silent.



(c) Silent attack, 30% are silent.



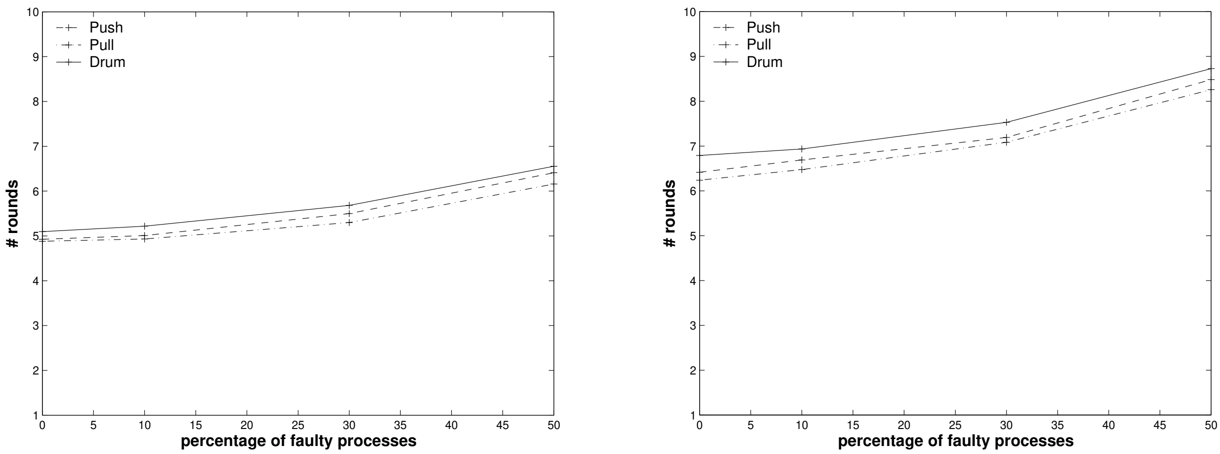
(d) Silent attack, 50% are silent.

Figure 7.2: Average propagation time to 99% of the correct processes, *n*=120, 240, 500, 1000.

Figure 7.3 illustrates the protocols' propagation times as the percentage of faulty (silent) processes increases, for $n = 120$ and $n = 1000$. It shows that the percentage of faulty processes has little effect on the protocols' behavior: even when as many as half of the processes fail, the protocols can continue to deliver messages reliably and with good quality of service.

7.2 DoS Attacks

We now turn our attention to DoS attacks. Here, we assume that the adversary operates from within the system, and that 10% of the processes are controlled by the adversary. As in silent attacks, they

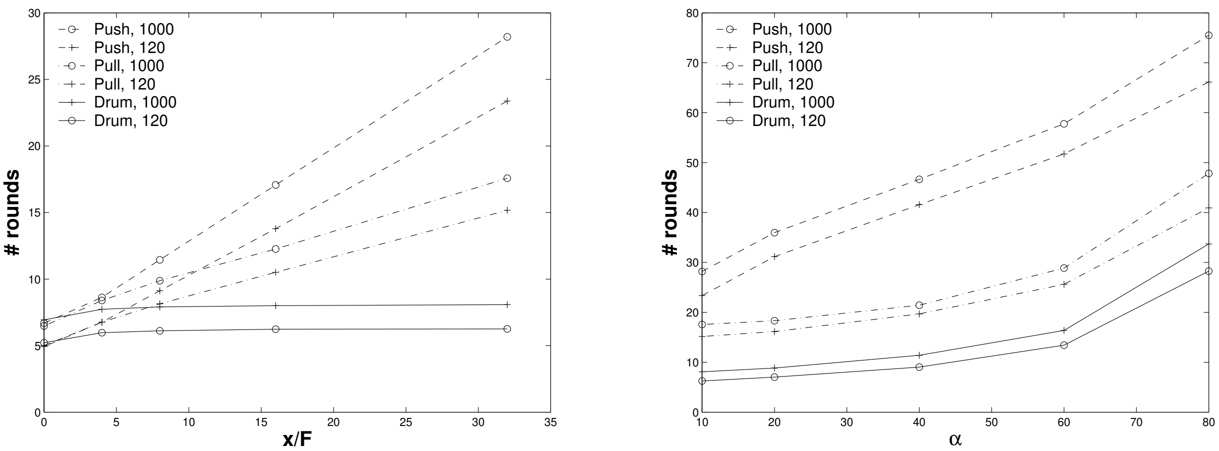


(a) Silent attack, n=120. (b) Silent attack, n=1000.

Figure 7.3: Average propagation time to 99% of the correct processes, for *n*=120, 1000.

do not propagate any valid messages. Note that the controlled (faulty) processes are not being attacked.

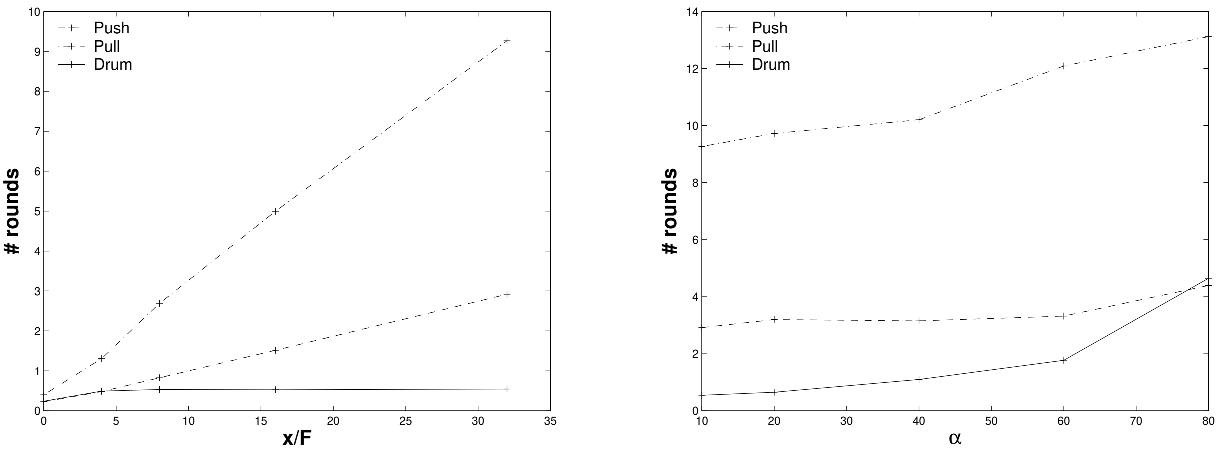
7.2.1 Targeted DoS Attacks



(a) $\alpha = 10\%$. (b) $x = 32F$.

Figure 7.4: Average propagation time to 99% of the correct processes, $n = 120, 1000$.

Figure 7.4 compares the time it takes M to reach 99% of the correct processes for the three protocols under various DoS attacks, with 120 and 1000 processes. Figure 7.4(a) shows that when 10% of the processes are attacked, the propagation time of both Push and Pull increases linearly



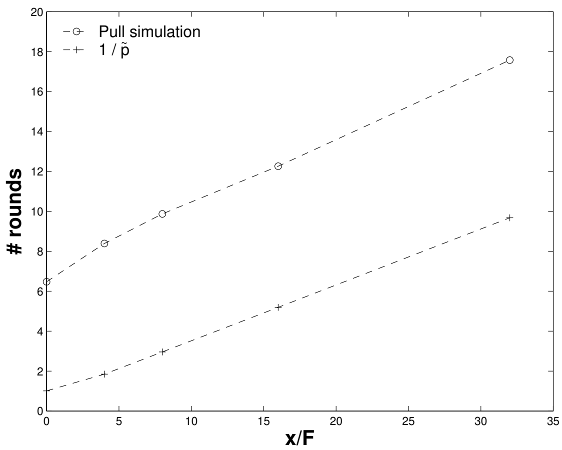
(a) $\alpha = 10\%$. (b) $x = 32F$.

Figure 7.5: STDs of the propagation time to 99% of the correct processes, $n = 1000$.

with the severity of the attack, while Drum's propagation time is unaffected by the attack strength. This is consistent with the prediction of Lemmas 1 and 5 and Corollary 2. Figure 7.4 (b) illustrates the propagation time as the percentage of attacked processes (and thus B) increases. Although the protocols exhibit similar trends, Drum propagates messages much faster than Push and Pull.

Figure 7.5 illustrates the *standard deviation* (STD) of the propagation times presented in Figure 7.4 for $n = 1000$. It shows that for a fixed α, Drum's STD is not affected by the attack strength, whereas the other protocols' STD increases linearly. Furthermore, both Drum and Push exhibit a small STD compared to Pull. E.g., for $\alpha = 10\%$ and $\frac{x}{F} = 32$, the STDs of Drum and Push are 0.5 and 2.9 rounds (resp.), whereas Pull's STD is 9.3 rounds. Thus, the behavior of Drum and Push is more predictable. The high STD of Pull's propagation time is mainly due to the large STD of the number of rounds it takes to propagate *M* beyond the source. The number of rounds it takes to propagate *M* beyond the source is geometrically distributed with $\widetilde{p}$, where $\widetilde{p}$ is the probability to propagate *M* beyond the source in a round (see Lemma 5 in Chapter 6.3). Thus, the STD of the number of rounds it takes to propagate *M* beyond the source is $\frac{\sqrt{1-\widetilde{p}}}{\widetilde{p}}$. A numerical calculation of $\widetilde{p}$ according to the formula in Appendix B with $n = 1000$, $F = 4$, and $\frac{x}{F} = 32$ yields an STD of 9.17 rounds, which explains Pull's measured STD of 9.3 rounds mentioned above.

Figure 7.6 compares $\frac{1}{\widetilde{p}}$ – the expected number of rounds it takes to propagate *M* beyond the



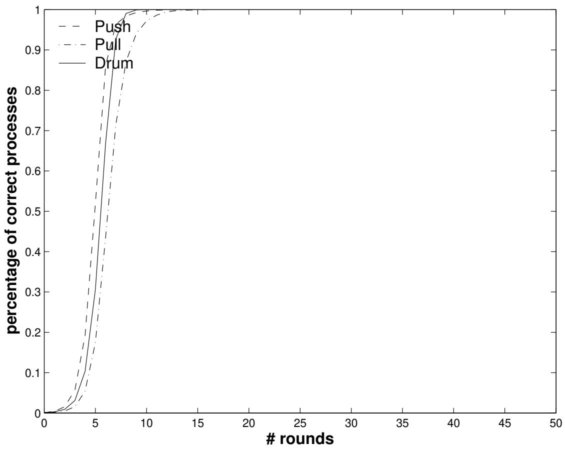
(a) $\alpha = 10\%$.

Figure 7.6: $\frac{1}{\widetilde{p}}$ vs. Pull's average propagation time to 99% of the correct processes, $n = 1000$.

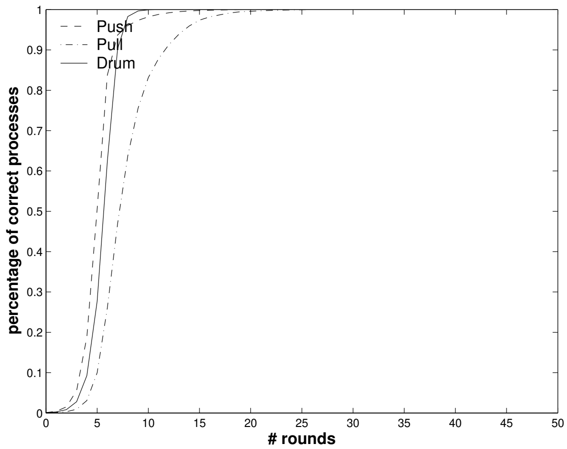
source in Pull (computed using the formula for $\widetilde{p}$ in Appendix B) – to Pull's average propagation time, for $n = 1000$, $\alpha = 10\%$. The figure shows that the linear growth of Pull's propagation time is mainly due to the linear growth in the time it takes to propagate *M* beyond the source.

Figures 7.7 and 7.8 illustrate the protocols' CDFs under different DoS attacks. As expected, Push propagates *M* to the non-attacked processes very quickly, but takes much longer to propagate it to the attacked processes. Again, we see that Drum significantly outperforms both Push and Pull when a strict subset of the system is attacked. Figure 7.8(d) shows that Drum succeeds in propagating *M* to 99% of the correct processes even under a severe attack, where 80% of the processes are attacked with 128 fabricated messages each. Note that since 10% of the processes are faulty, only 10% of the processes in the system are both correct and non-attacked in this case. Thus, even if only 10% of the processes in the system are "healthy", they can still propagate messages effectively to and from all other correct processes.

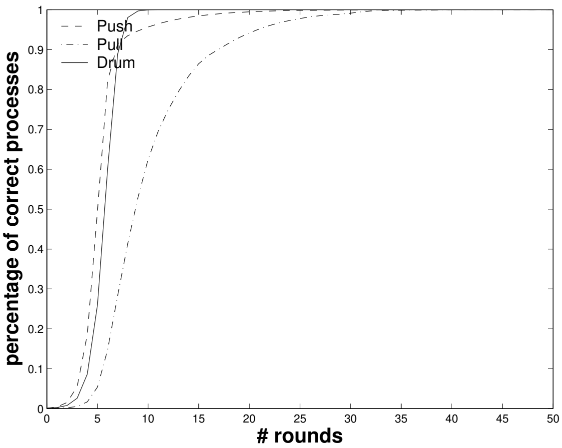
Interestingly, on average, Push propagates *M* to more processes per round than Pull does (see figures 7.7 and 7.8), although the average number of rounds Pull takes to propagate M to 99% of the correct processes is smaller than that of Push (see Figure 7.4). This paradox occurs since, with Pull, there is a non-negligible probability that *M* is delayed at the source for a long time. With $n = 1000$, $F = 4$, and $\frac{x}{F} = 32$, the probability for *M* not being propagated beyond the source



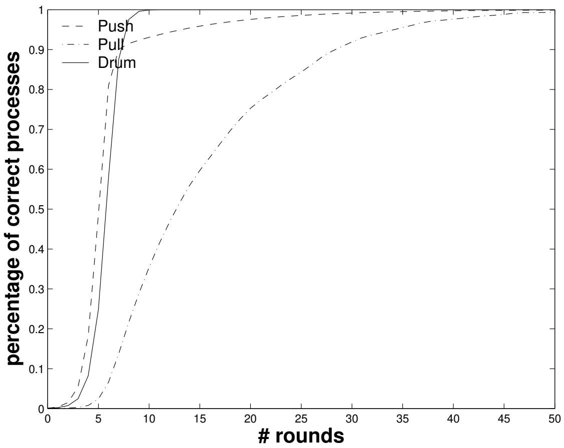
(a) $\alpha = 10\%, x = 4F$.



(b) $\alpha = 10\%, x = 8F$.



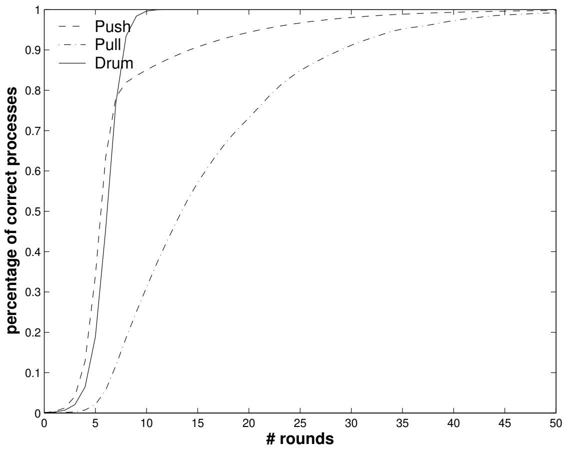
(c) $\alpha = 10\%, x = 16F$.



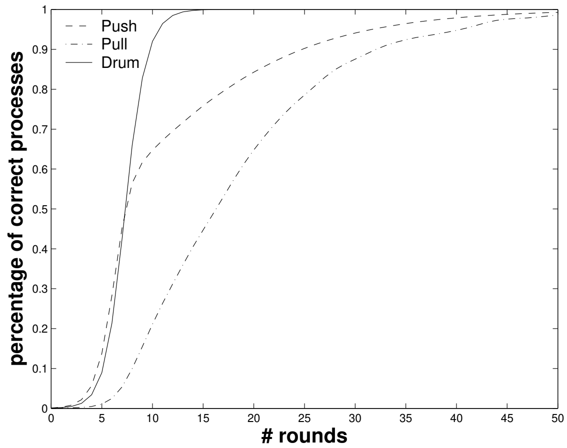
(d) $\alpha = 10\%, x = 32F$.

Figure 7.7: CDFs of percentage of correct processes that receive *M*, $n = 1000$.

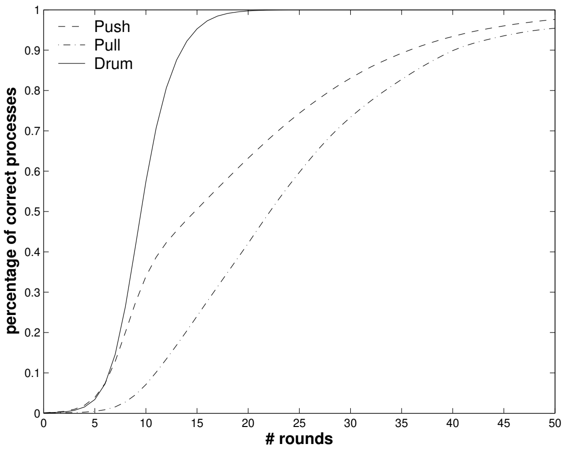
in 5, 10, and 15 rounds is 0.58, 0.34, and 0.19 resp. (as computed using the formula for $\widetilde{p}$ in Appendix B). Once M reaches one non-attacked process, it quickly propagates to the rest of the processes. Therefore, even if by a certain round k, in most runs, a large percentage of the processes have M, there is still a non-negligible number of runs in which Pull does not reach *any* process (other than the source) by round k. This large difference in the percentage of processes reached has a large impact on the average depicted in figures 7.7 and 7.8. In contrast, Push, which reaches all the non-attacked processes quickly in all runs, does not have runs with such low percentages factoring into this average. Nevertheless, Push's average propagation time to 99% of the correct processes is much higher than Pull's, because Push has to propagate M to *all* the attacked processes, whereas



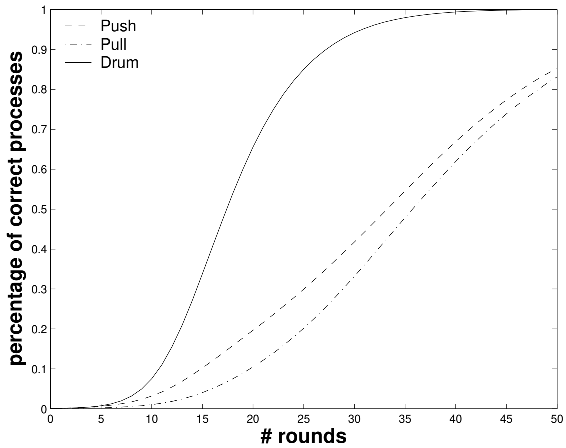
(a) $\alpha = 20\%, x = 32F$.



(b) $\alpha = 40\%, x = 32F$.



(c) $\alpha = 60\%, x = 32F$.



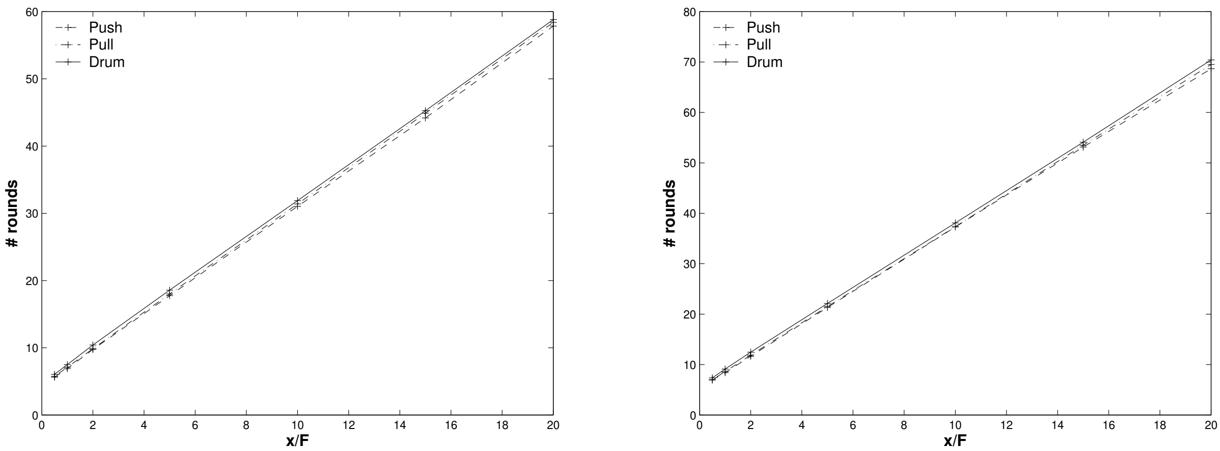
(d) $\alpha = 80\%, x = 32F$.

Figure 7.8: CDFs of percentage of correct processes that receive M, $n = 1000$.

Pull has to propagate M only out of one attacked process.

7.2.2 Broad Attacks

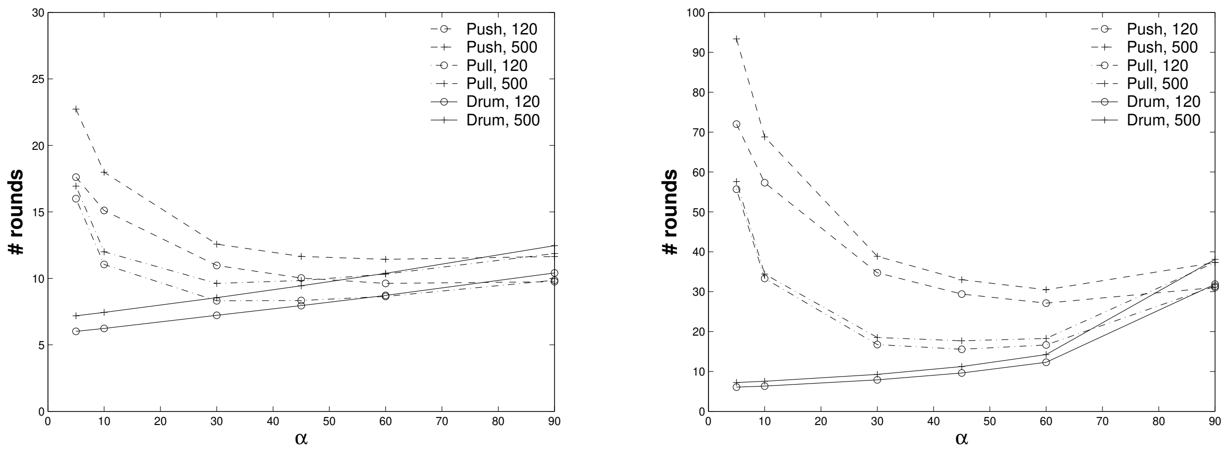
We now consider broad attacks, i.e., the case that all correct processes are attacked. Since 90% are correct, this corresponds to $\alpha = 0.9$. Figure 7.9 compares the average propagation time to 99% of the correct processes with the three protocols under broad attacks with increasing attack strengths (B). As Figure 7.9 shows, under broad attacks, the behavior of the three protocols is almost identical. Moreover, their propagation time increases linearly with the attack strength.



$n = 120$ $n = 500$

Figure 7.9: Average propagation time to 99% of the correct processes, broad attacks.

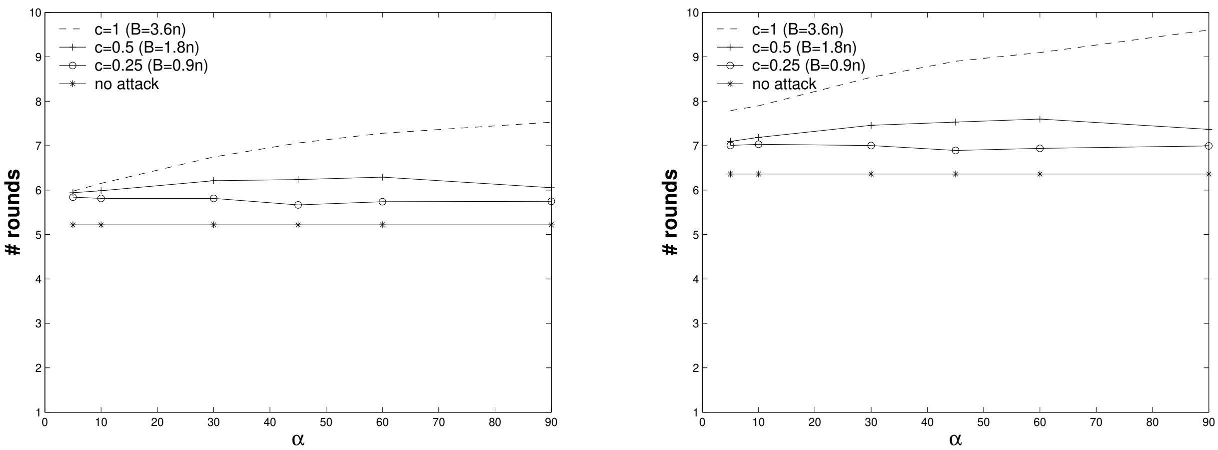
7.2.3 Adversary Strategies



$B = 7.2n(c = 2)$ $B = 36n(c = 10)$

Figure 7.10: Average propagation time to 99% of the correct processes.

We now evaluate the protocols under a range of adversary strategies with fixed adversary strengths. First, we consider severe attacks with $B = 7.2n$ and $B = 36n$ (corresponding to $c = 2$, and $c = 10$, respectively) fabricated messages per round. If the adversary chooses to attack all correct processes, it can send 8 (respectively, 40) fabricated messages to each of them in each round because 90% of the processes are correct. If the adversary instead focuses on 10% of the processes, it can send 72 (respectively, 360) fabricated messages per round to each. Figure 7.10 illustrates the protocols' average propagation times with different percentages of attacked processes, for system sizes of 120 and 500. It validates the prediction of Lemma 2, and shows that Drum's propagation time increases with α. Thus, an adversary cannot "benefit" from focusing its capacity on a small subset of the processes. In contrast, the performance of the other two protocols is seriously hampered when a small subset of the processes is targeted. Not surprisingly, the three protocols perform equally when all the correct processes are targeted.



n=120 n=500

Figure 7.11: Drum, average propagation time to 99% of correct processes.

Next, we evaluate Drum under attacks with relatively small adversary powers of $B = 0.9n$, $B = 1.8n$, and $B = 3.6n$ (corresponding to $c = 0.25$, $c = 0.5$ and $c = 1$, respectively) and also without an attack (as a baseline). As Figure 7.11 shows, under such attacks, Drum's propagation time either increases with α or is virtually unaffected by the choice of the adversary strategy.

Chapter 8

Detailed Analysis

In this chapter we give a formal analysis of the three protocols in the absence of DoS attacks (no attack and silent attack) as well as under Dos attacks. In Section 8.1, we describe the parameters and notations used in our analysis formulas. In Section 8.2, we detail our analysis formulas. In Section 8.3, we use these formulas to compute the expected percentage of correct processes that receive *M* for a given round, and compare these results with the simulation results of the previous chapter.

8.1 Definitions

Parameters

- b – number of faulty processes.
- ϵ_{loss} – the probability of a message loss. We assume that its distribution is uniform and independent of any other factor.
- $F_{in-push}$, $F_{in-pull}$ – bounds on the number of processes to receive messages from in a round, in a push or a pull (respectively) operation. We fix $F_{in-push} = F_{in-pull} = \frac{F}{2}$ in Drum, $F_{in-push} = F$ in Push , and $F_{in-pull} = F$ in Pull.

- x_{push}, x_{pull} – number of fabricated push or pull (respectively) messages sent to an attacked process in each round. In Drum, $x_{push} = x_{pull} = \frac{x}{2}$, in Push, $x_{push} = x$, and in Pull, $x_{pull} = x$.

Notation

- *p* – the probability that a given correct process, *target*, will receive a gossip message *M* from another given correct process, *sender*, in a certain round. We denote $q = 1 - p$.
- p_{push}, p_{pull} – similar to *p*, but as a result of a push or a pull (respectively) operation.
- d_{push} – the probability that the target will discard the sender's incoming push message due to the bound $F_{in-push}$ on push messages accepted during each round, given that the sender's message reached the target.
- d_{pull} – the probability that the sender will discard the target's incoming pull-request message due to the bound $F_{in-pull}$ on pull-request messages accepted during each round, given that the target's message reached the sender.

  Note that since $F_{in-push}$ and $F_{in-pull}$ are smaller in Drum than in Push and Pull, these evaluate to different values for different protocols.
- S_r – the number of correct processes that have *M* at the beginning of round *r*, $S_r \in [1 \ldots n - b]$.

The probability d is similar to $1 - p_u$ computed in Appendix A for the asymptotic analysis, expect for the following: (i) $\epsilon_{loss} > 0$, (ii) there are faulty processes ($b > 0$), and (iii) in Drum, the bounds $F_{in-push}$ and $F_{in-pull}$ are tested separately. That is, each processes accepts push messages from at most $F_{in-push}$ processes, and pull-request messages from at most $F_{in-pull}$ processes.

8.2 Formulas

In Section 8.2.1, we present the formulas for the case without DoS attacks. In Section 8.2.2, we add DoS attacks. In the next section we present the results obtained from both analyses.

8.2.1 Without DoS Attacks

Here, the b faulty processes are silent in the sense that they do not send any messages (when $b = 0$ there is no attack). Our formulas are based on the analysis of a push-based protocol presented in [9].

We first compute d_{push} and d_{pull}. Where d_{push} is the probability that the target will discard the sender's incoming push message, given that the sender's message reached the target, and d_{pull} is the probability that the sender will discard the target's incoming pull-request message, given that the target's message reached the sender. Note that d_{push} (respectively, d_{pull}) depends on the number of messages that the target (respectively, sender) receives in a gossip round, since a process will accept only the first $F_{in-push}$ (respectively, $F_{in-pull}$) push (respectively, pull-request) messages in each round. The computation of d_{push} is as follows:
Denote the event "sender s chooses target t and the sender's message is not lost" by R_{s-t}. Let Y be the number of valid messages received by the target in a single round, and Z be the number of processes the choose the target in a single round, then:

$$\Pr(Y = y\ ,\ Z = z \mid R_{s-t}) \quad = \quad \binom{n-b-2}{z-1}\left(\frac{|view_{push}|}{n-1}\right)^{z-1}\left(1-\frac{|view_{push}|}{n-1}\right)^{n-b-1-z}\binom{z-1}{y-1}(1-\epsilon_{loss})^{y-1}{\epsilon_{loss}}^{z-y}$$

Where, $0 < y \leq z < n - b$. Therefore,

$$\begin{aligned}\Pr(Y = y \mid R_{s-t}) \quad &= \quad \sum_z \Pr(Y = y\ ,\ Z = z \mid R_{s-t}) = \\ &\sum_{z=y}^{n-b-1}\binom{n-b-2}{z-1}\left(\frac{|view_{push}|}{n-1}\right)^{z-1}\left(1-\frac{|view_{push}|}{n-1}\right)^{n-b-1-z}\binom{z-1}{y-1}(1-\epsilon_{loss})^{y-1}{\epsilon_{loss}}^{z-y}\end{aligned}$$

Let q_Y be the probability that the target discards the message sent by the sender, given R_{s-t}, then:

$$q_Y = \begin{cases} 0 & Y \leq F_{in-push} \\ \frac{Y-1}{Y}\cdot\frac{Y-2}{Y-1}\cdots\frac{Y-F}{Y-F+1} = \frac{Y-F}{Y} & Y > F_{in-push} \end{cases}$$

Calculating d_{push} gives:

$$d_{push} = \sum_y q_y \cdot Pr(Y = y \mid R_{s-t}) =$$

$$\sum_{y=F_{in-push}+1}^{n-b-1} \frac{y - F_{in-push}}{y} \cdot$$

$$\sum_{z=y}^{n-b-1} \binom{n-b-2}{z-1} \left(\frac{|view_{push}|}{n-1}\right)^{z-1} \left(1 - \frac{|view_{push}|}{n-1}\right)^{n-b-1-z} \binom{z-1}{y-1} (1-\epsilon_{loss})^{y-1} (\epsilon_{loss})^{z-y}$$

We compute d_{pull} similarly to the computation of d_{push}, and get the following formula:

$$d_{pull} = \sum_y q_y \cdot Pr(Y = y \mid R_{t-s}) =$$

$$\sum_{y=F_{in-pull}+1}^{n-b-1} \frac{y - F_{in-pull}}{y} \cdot$$

$$\sum_{z=y}^{n-b-1} \binom{n-b-2}{z-1} \left(\frac{|view_{pull}|}{n-1}\right)^{z-1} \left(1 - \frac{|view_{pull}|}{n-1}\right)^{n-b-1-z} \binom{z-1}{y-1} (1-\epsilon_{loss})^{y-1} (\epsilon_{loss})^{z-y}$$

We now compute the probabilities p_{push} and p_{pull}. Where p_{push} (respectively, p_{pull}) is the probability that the target will successfully receive a gossip message *M* from the sender, in a certain round, as a result of a push (respectively, pull) operation. The formula for p_{push} includes dependent probabilities. That is,

$$p_{push} = \Pr\left(\begin{array}{c} sender\ chooses \\ target \end{array}, \begin{array}{c} the\ msg\ is \\ not\ lost \end{array}, \begin{array}{c} target\ does\ not \\ drop\ the\ msg \end{array}\right) =$$

$$\Pr\left(\begin{array}{c} sender\ chooses \\ target \end{array}\right) \Pr\left(\begin{array}{c} the\ msg\ is \\ not\ lost \end{array}\right) \Pr\left(\begin{array}{c} target\ does\ not \\ drop\ the\ msg \end{array} \mid \begin{array}{c} sender\ chooses \\ target \end{array}, \begin{array}{c} the\ msg\ is \\ not\ lost \end{array}\right) =$$

$$\left(\frac{|view_{push}|}{n-1}\right)(1-\epsilon_{loss})(1-d_{push})$$

The probability p_{pull} is calculated similarly:

$$p_{pull} = \left(\frac{|view_{pull}|}{n-1}\right)(1-\epsilon_{loss})^2(1-d_{pull})$$

In Push $p = p_{push}$, in Pull $p = p_{pull}$, and in Drum $p = 1 - (1-p_{push})(1-p_{pull})$.

Given that *i* correct processes have *M* at the beginning of a round, we define the probability p_{ij} that exactly *j* ($j \geq i$) correct processes have *M* at the beginning of the next round:

$$p_{ij} \triangleq P_r(S_{r+1} = j | S_r = i)$$

We now approximate p_{ij} as follows [1]:

$$p_{ij} \approx \binom{n-b-i}{j-i}(1-q^i)^{j-i}(q^i)^{n-b-j}$$

S_r is computed recursively as follows:

$$P_r(S_0 = 1) = 1,$$

$$P_r(S_{r+1} = j) = \sum_{i \leq j} P_r(S_r = i)p_{ij}$$

The expected number of correct processes that have *M* at the beginning of each gossip round is then as follows:

$$E(S_r) = \sum_{1 \leq j \leq n-b} P_r(S_r = j)j$$

8.2.2 DoS Attacks

We now add DoS attacks into the mix. For a probability P, we denote by P^a the probability of P when the process is under an attack, and by P^u the case that the process is not under an attack. E.g., d^a_{push} is the probability that an attacked target will discard the sender's incoming push message, and d^u_{push} is the probability that a non-attacked target will discard the sender's incoming push message.

Under a DoS attack, d^u_{push} (respectively d^u_{pull}) is equal to d_{push} (respectively, d_{pull}) as calculated above, whereas the formulas for d^a_{push} and d^a_{pull} also depend on the number of fabricated push or pull-request (respectively) messages sent to an attacked correct process in each round, i.e., x_{push}

[1] This formula gives an over-estimate of p_{ij}, since some of the counted events reflect a situation where a process sends and receives more messages than allowed by the F_{in} and F_{out} bounds. However, the probabilities of these events can be neglected (see [2]).

and x_{pull}. The computation of d^a_{push} is now as follows:

Let $\acute{X}_{push}$ be the number of fabricated push messages that the target receives in a single round, then:

$$\Pr(\acute{X}_{push} = \acute{x}_{push}) = \binom{x_{push}}{\acute{x}_{push}}(1-\epsilon_{loss})^{\acute{x}_{push}}{\epsilon_{loss}}^{x_{push}-\acute{x}_{push}}$$

and

$$q_Y = \begin{cases} 0 & Y + \acute{X}_{push} \leq F_{in-push} \\ \frac{Y+\acute{X}_{push}-F}{Y+\acute{X}_{push}} & Y + \acute{X}_{push} > F_{in-push} \end{cases}$$

Calculating d^a_{push} gives:

$$d^a_{push} = \sum_{y}\sum_{\acute{x}_{push}} q_y \cdot \Pr(\acute{X}_{push} = \acute{x}_{push}) \cdot Pr(Y = y \mid R_{s-t}) =$$

$$\sum_{y=1}^{n-b-1}\sum_{\acute{x}_{push}=0}^{x_{push}} \max\left\{0, \frac{y+\acute{x}_{push}-F_{in-push}}{y+\acute{x}_{push}}\right\} \cdot \binom{x_{push}}{\acute{x}_{push}}(1-\epsilon_{loss})^{\acute{x}_{push}}(\epsilon_{loss})^{x_{push}-\acute{x}_{push}} \cdot$$
$$\sum_{z=y}^{n-b-1}\binom{n-b-2}{z-1}\left(\frac{|view_{push}|}{n-1}\right)^{z-1}\left(1-\frac{|view_{push}|}{n-1}\right)^{n-b-1-z}\binom{z-1}{y-1}(1-\epsilon_{loss})^{y-1}(\epsilon_{loss})^{z-y}$$

We compute d^a_{pull} similarly to the computation of d^a_{push}, and get the following formula:

$$d^a_{pull} = \sum_{y}\sum_{\acute{x}_{pull}} q_y \cdot \Pr(\acute{X}_{pull} = \acute{x}_{pull}) \cdot Pr(Y = y \mid R_{t-s}) =$$

$$\sum_{y=1}^{n-b-1}\sum_{\acute{x}_{pull}=0}^{x_{pull}} \max\left\{0, \frac{y+\acute{x}_{pull}-F_{in-pull}}{y+\acute{x}_{pull}}\right\} \cdot \binom{x_{pull}}{\acute{x}_{pull}}(1-\epsilon_{loss})^{\acute{x}_{pull}}(\epsilon_{loss})^{x_{pull}-\acute{x}_{pull}} \cdot$$
$$\sum_{z=y}^{n-b-1}\binom{n-b-2}{z-1}\left(\frac{|view_{pull}|}{n-1}\right)^{z-1}\left(1-\frac{|view_{pull}|}{n-1}\right)^{n-b-1-z}\binom{z-1}{y-1}(1-\epsilon_{loss})^{y-1}(\epsilon_{loss})^{z-y}$$

The probabilities p^u_{push} (p^u_{pull}) and p^a_{push} (p^a_{pull}) are computed as follows:

$$p^u_{push} = (\frac{|view_{push}|}{n-1})(1-\epsilon_{loss})(1-d^u_{push}), \quad p^u_{pull} = (\frac{|view_{pull}|}{n-1})(1-\epsilon_{loss})^2(1-d^u_{pull})$$

$$p^a_{push} = (\frac{|view_{push}|}{n-1})(1-\epsilon_{loss})(1-d^a_{push}), \quad p^a_{pull} = (\frac{|view_{pull}|}{n-1})(1-\epsilon_{loss})^2(1-d^a_{pull})$$

Given that i_u and i_a non-attacked and attacked (respectively) correct processes have *M* at the beginning of a round, we can compute the probability q^*_u (respectively q^*_a), that none of these i_u and i_a processes successfully propagate *M* to any other non-attacked (attacked) correct process:

Push:

$$q^*_u = (1-p^u_{push})^{i_u+i_a}$$

$$q^*_a = (1-p^a_{push})^{i_u+i_a}$$

Pull:

$$q^*_u = q^*_a = (1-p^u_{pull})^{i_u} \cdot (1-p^a_{pull})^{i_a}$$

Drum:

$$q^*_u = (1-p^u_{push})^{i_u+i_a} \cdot (1-p^u_{pull})^{i_u} \cdot (1-p^a_{pull})^{i_a}$$

$$q^*_a = (1-p^a_{push})^{i_u+i_a} \cdot (1-p^u_{pull})^{i_u} \cdot (1-p^a_{pull})^{i_a}$$

Now we can compute the joint probability $p_{i_u i_a j_u j_a}$ that exactly j_u and j_a ($j_u \geq i_u, j_a \geq i_a$) correct processes have *M* at the beginning of the next round:

$$p_{i_u i_a j_u j_a} \triangleq P_r(S^u_{r+1} = j_u, S^a_{r+1} = j_a | S^u_r = i_u, S^a_r = i_a) =$$

$$P_r(S^u_{r+1} = j_u | S^u_r = i_u, S^a_r = i_a) \cdot P_r(S^a_{r+1} = j_u | S^u_r = i_u, S^a_r = i_a)$$

Where:

$$P_r(S^u_{r+1} = j_u | S^u_r = i_u, S^a_r = i_a) = \binom{n-b-\alpha n-i_u}{j_u-i_u}(1-q^*_u)^{j_u-i_u}(q^*_u)^{n-b-\alpha n-j_u}$$

and

$$P_r(S^a_{r+1} = j_a | S^u_r = i_u, S^a_r = i_a) = \binom{\alpha n-i_a}{j_a-i_a}(1-q^*_a)^{j_a-i_a}(q^*_a)^{\alpha n-j_a}$$

S_r^u and S_r^a are computed recursively as follows:

$$P_r(S_0^u = 0, S_0^a = 1) = 1$$

$$P_r(S_{r+1}^u = j_u, S_{r+1}^a = j_a) = \sum_{0 \leq i_u \leq j_u} \sum_{1 \leq i_a \leq j_a} P_r(S_r^u = i_u, S_r^a = i_a) p_{i_u i_a j_u j_a}$$

$$P_r(S_{r+1}^u = j_u) = \sum_{0 \leq j_u \leq n-b-\alpha n} P_r(S_{r+1}^u = j_u, S_{r+1}^a = j_a)$$

$$P_r(S_{r+1}^a = j_a) = \sum_{1 \leq j_a \leq \alpha n} P_r(S_{r+1}^u = j_u, S_{r+1}^a = j_a)$$

The expected number of non-attacked and attacked correct processes that have *M* at the beginning of each round, is calculated as follows:

$$E(S_r^u) = \sum_{0 \leq j_u \leq n-b-\alpha n} P_r(S_r^u = j_u) j_u$$

$$E(S_r^a) = \sum_{1 \leq j_a \leq \alpha n} P_r(S_r^a = j_a) j_a$$

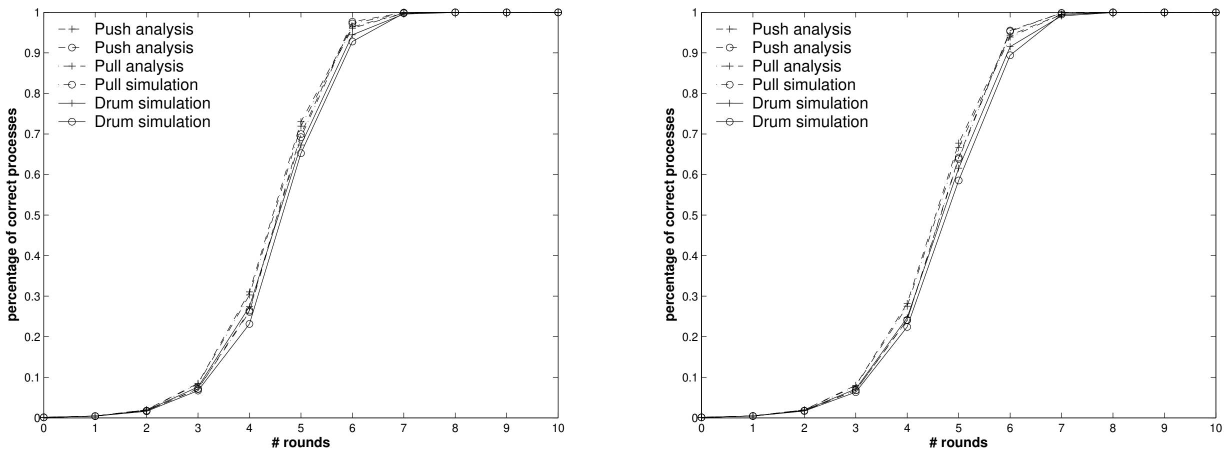
$$E(S_r) = E(S_r^u) + E(S_r^a)$$

8.3 Results

We numerically computed the above formulas using *MATLAB*, with the same parameters used in the simulation presented in the previous chapter. We now compare the analysis results with the results obtained in the simulation.

We show CDFs of the percentage of correct processes that receive *M* by each round. Figure 8.1 shows the CDFs under no attack (Figure 8.1 (a)) and under a silent attack (Figure 8.1 (b)). The analysis and simulation results are almost identical.

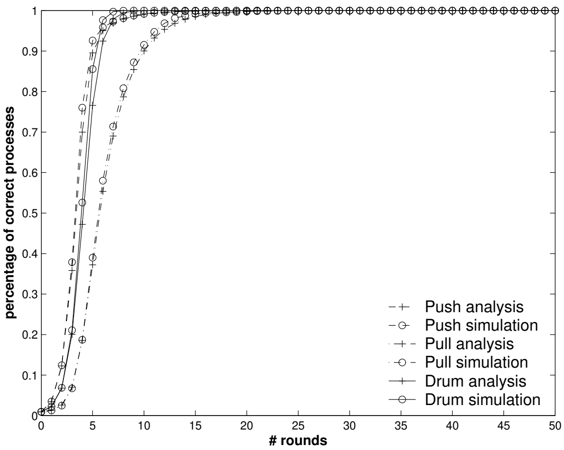
Figure 8.2 compares our analysis results with the simulation results under various DoS attacks for a system with 120 processes. Again, the analysis and simulation results are virtually the same. Thus, the analysis validates our simulations (and vice versa).



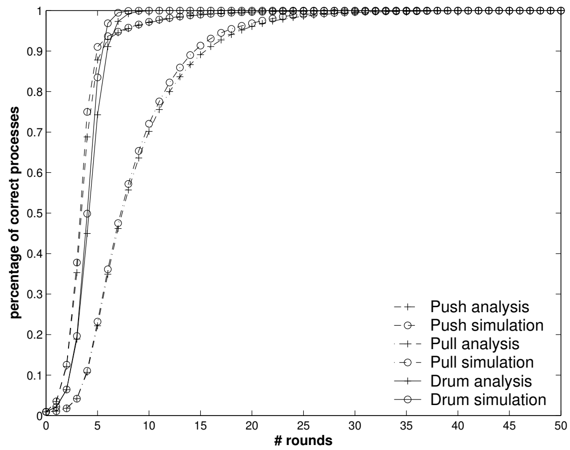
(a) No attack.

(b) Silent attack, 10% are silent.

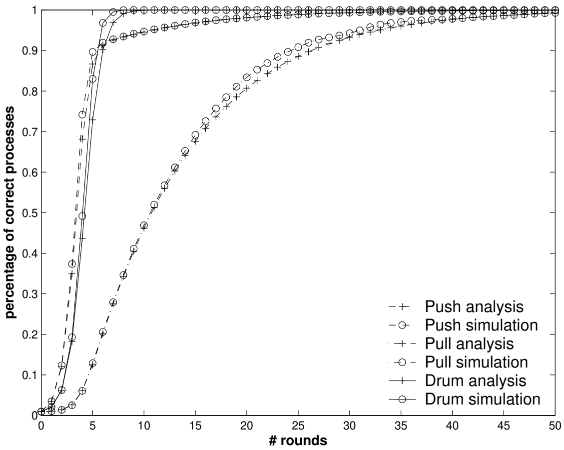
Figure 8.1: Analysis vs. Simulation: CDFs of percentage of correct processes that receive M, $n = 1000$.



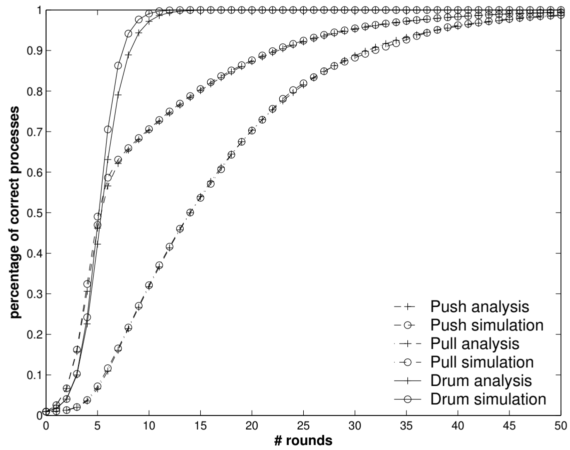
(a) α=10%, $\frac{x}{F}$=8.



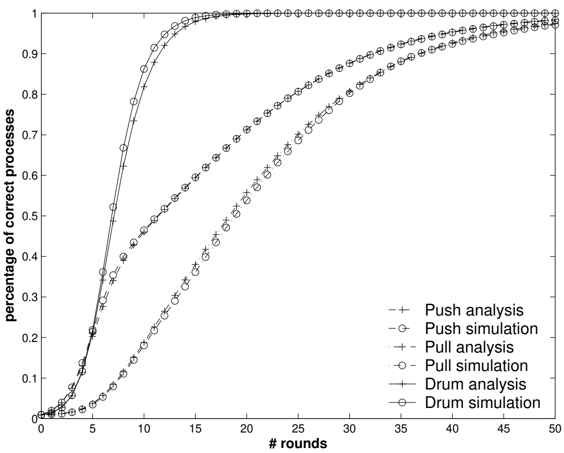
(b) α=10%, $\frac{x}{F}$=16.



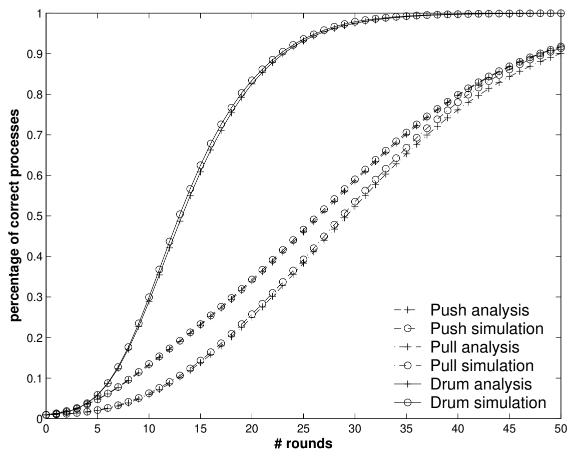
(c) α=10%, $\frac{x}{F}$=32.



(d) α=40%, $\frac{x}{F}$=32.



(e) α=60%, $\frac{x}{F}$=32.



(f) α=80%, $\frac{x}{F}$=32.

Figure 8.2: Analysis vs. Simulation: CDFs of percentage of correct processes that receive M, $n = 120$.

Chapter 9

Conclusions

We have conducted the first systematic study of the impact of DoS attacks on multicast protocols, using asymptotic analysis, simulations, and detailed analysis. Our study has exposed weaknesses of traditional gossip-based multicast protocols: Although such protocols are very robust in the face of process crashes, we have shown that they can be extremely vulnerable to DoS attacks. In particular, an attacker with limited attack strength can cause severe performance degradation by focusing on a small subset of the processes.

We have suggested a few simple measures that one can take in order to improve a system's resilience to DoS attacks: (i) combining pull and push operations; (ii) bounding resources separately for each operation; and (iii) random port selection. We have presented Drum, a simple gossip-based multicast protocol that uses these measures in order to eliminate vulnerabilities to DoS attacks. Our closed-form mathematical analysis, simulations, and detailed analysis have proven that using both push and pull operations goes a long way in fortifying a system against DoS attacks. We have shown that, as the attack strength increases asymptotically, the most effective attack against Drum is one that targets all the correct processes in the system. As expected, the inevitable performance degradation due to such a broad attack is identical for all the studied protocols. However, protocols that use only pull or only push operations perform much worse under more focused attacks, which have little influence on Drum.

Bibliography

[1] G. Badishi, I. Keidar, and A. Sasson. The design and architecture of the Drum secure multicast system. Technical report. `http://www.ee.technion.ac.il/~badishi/papers/DRUM-arch.ps`.

[2] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Trans. Comput. Syst.*, 17(2):41–88, 1999.

[3] K. P. Birman, R. van Renesse, and W. Vogels. Spinglass: Secure and scalable communications tools for mission-critical computing. In *DARPA International Survivability Conference and Exposition (DISCEX)*, June 2001.

[4] R. K. C. Chang. Defending against flooding-based distributed denial-of-service attacks: A tutorial. *IEEE Communications Magazine*, 40:42–51, October 2002.

[5] Cisco Systems. Defining strategies to protect against TCP SYN denial of service attacks. `http://www.cisco.com/warp/public/707/4.html`.

[6] CSI/FBI. Computer crime and security survey. 2003. `http://www.gocsi.com/forms/fbi/pdf.jhtml`.

[7] A. Das, I. Gupta, and A. Motivala. SWIM: Scalable weakly consistent infection-style process group membership protocol. In *The International Conference on Dependable Systems and Networks (DSN)*, pages 303–312, 2002.

[8] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Stuygis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *6th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–12, 1987.

[9] P. T. Eugster, R. Guerraoui, S. B. Handurukande, A. M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. In *The International Conference on Dependable Systems and Networks (DSN)*, 2001.

[10] A. J. Ganesh, A. M. Kermarrec, and L. Massoulie. Scamp: Peer-to-peer lightweight membership service for large-scale group communication. In *3rd International Workshop on Networked Group Communications (NGC)*, November 2001.

[11] X. Geng and A. B. Whinston. Defeating distributed denial of service attacks. *IEEE IT Professional*, pages 46–51, July/August 2000.

[12] I. Gupta, A.-M. Kermarrec, and A. J. Ganesh. Efficient epidemic-style protocols for reliable and scalable multicast. In *21st IEEE International Symposium on Reliable Distributed Systems (SRDS)*, pages 180–189, October 2002.

[13] I. Gupta, R. van Renesse, and K. P. Birman. A probabilistically correct leader election protocol for large groups. In *14th International Symposium on DIStributed Computing (DISC)*, pages 89–103, Oct. 2000.

[14] I. Gupta, R. van Renesse, and K. P. Birman. Scalable fault-tolerant aggregation in large process groups. In *The International Conference on Dependable Systems and Networks (DSN)*, pages 433–442, 2001.

[15] A. Habib, M. Hefeeda, and B. Bhargava. Detecting service violations and dos attacks. In *Proceedings of Network and Distributed System Security Symposium, San Diego*, pages 177–189, February 2003.

[16] A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks. In *Proceedings of ACM SIGCOMM*, August 2003.

[17] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proceedings of Network and Distributed System Security Symposium, San Diego*. The Internet Society, February 2002.

[18] R. M. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *IEEE Symposium on Foundations of Computer Science*, pages 565–574, 2000.

[19] D. Kempe, J. Kleinberg, and A. Demers. Spatial gossip and resource location protocols. In *33rd ACM Symp. on Theory of Computing (STOC)*, 2001.

[20] A.-M. Kermarrec, L. Massouli, and A. J. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3):248–258, March 2003.

[21] M. J. Lin and K. Marzullo. Directional gossip: Gossip in a wide area network. In *European Dependable Computing Conference (EDCC)*, pages 364–379, 1999.

[22] M. J. Lin, K. Marzullo, and S. Masini. Gossip versus deterministically constrained flooding on small networks. In *14th International Symposium on DIStributed Computing (DISC)*, pages 253–267, 2000.

[23] P. Linga, I. Gupta, and K. Birman. A churn-resistant peer-to-peer web caching system. *ACM Workshop on Survivable and Self-Regenerative Systems*, October 2003.

[24] D. Malkhi, Y. Mansour, and M. K. Reiter. Diffusion without false rumors: On propagating updates in a Byzantine environment. *Theoretical Computer Science*, 299(1–3):289–306, April 2003.

[25] D. Malkhi, E. Pavlov, and Y. Sella. Optimal unconditional information diffusion. In *15th International Symposium on DIStributed Computing (DISC)*, 2001.

[26] D. Malkhi, M. K. Reiter, O. Rodeh, and Y. Sella. Efficient update diffusion in Byzantine environments. In *20th IEEE International Symposium on Reliable Distributed Systems (SRDS)*, October 2001.

[27] Y. M. Minsky and F. B. Schneider. Tolerating malicious gossip. *Distributed Computing*, 16(1):49–68, February 2003.

[28] D. Moore, G. Voelker, and S. Savage. Inferring Internet denial-of-service activity. In *Proceedings of the 10th USENIX Security Symposium*, pages 9–22, 2001.

[29] B. Pittel. On spreading a rumor. *SIAM Journal on Applied Mathematics*, 47(1):213–223, February 1987.

[30] S. Staniford, V. Paxson, and N. Weaver. How to own the Internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, pages 149–167, August 2002.

[31] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. In *IFIP Intl. Conference on Distributed Systems Platforms and Open Distributed Processing*, 1998.

[32] H. Wang, D. Zhang, and K. Shin. Detecting syn flooding attacks. In *Proceedings of IEEE INFOCOM*, pages 1530 –1539, June 2002.

[33] J. Wang, L. Lu, and A. A. Chien. Tolerating denial-of-service attacks using overlay networks – impact of overlay network topology. *ACM Workshop on Survivable and Self-Regenerative Systems*, October 2003.

[34] E. W. Weisstein. *CRC Concise Encyclopedia of Mathematics, Second Edition*. Chapman and Hall, London, UK, 1999.

[35] L. Zhou, F. B. Schneider, and R. van Renesse. COCA: A secure distributed online certification authority. *ACM Trans. Comput. Syst.*, 20(4):329–368, 2002.

Appendix A

Calculating p_u and p_a

Suppose process p_i sends a message to process p_j, we want to calculate the probability that process p_j accepts this message. Denote the event "process p_i sends a message to process p_j" by S_{ij}.
Let Y be the number of valid messages received by p_j in a single round, then:

$$0 < y < n \quad \Pr(Y = y \mid S_{ij}) \quad = \quad \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(1 - \frac{F}{n-1}\right)^{n-1-y}$$

Let q_Y be the probability that a non-attacked process, p_j, discards the message sent by p_i, given S_{ij}, then:

$$q_Y = \begin{cases} 0 & Y \leq F \\ \frac{Y-1}{Y} \cdot \frac{Y-2}{Y-1} \cdots \frac{Y-F}{Y-F+1} = \frac{Y-F}{Y} & Y > F \end{cases}$$

Calculating p_u gives:

$$
\begin{aligned}
p_u &= 1 - \sum_{y=-\infty}^{\infty} q_y \cdot Pr(Y = y \mid S_{ij}) = \\
& 1 - \sum_{y=F+1}^{n-1} \frac{y-F}{y} \cdot \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y} = \\
& \sum_{y=1}^{F} \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y} + \\
& \sum_{y=F+1}^{n-1} \frac{F}{y} \cdot \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y}
\end{aligned} \tag{A.1}
$$

If p_j is attacked with $x \geq F$ messages, we get:

$$
q_Y = \frac{Y+x-1}{Y+x} \cdot \frac{Y+x-2}{Y+x-1} \cdots \frac{Y+x-F}{Y+x-F+1} = \frac{Y+x-F}{Y+x}
$$

And thus:

$$
\begin{aligned}
p_a &= 1 - \sum_{y=-\infty}^{\infty} q_y \cdot Pr(Y = y \mid S_{ij}) = \\
& 1 - \sum_{y=1}^{n-1} \frac{y+x-F}{y+x} \cdot \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y} = \\
& \sum_{y=1}^{n-1} \frac{F}{y+x} \cdot \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y}
\end{aligned}
$$

We compue the deriviation of p_a:

$$\begin{aligned}
\frac{\mathrm{d}p_a}{\mathrm{d}p_x} &= \sum_{y=1}^{n-1} \frac{\mathrm{d}\frac{F}{y+x}}{\mathrm{d}x} \cdot \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y} = \\
& \sum_{y=1}^{n-1} \frac{-F}{(y+x)^2} \cdot \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y} \\
\frac{\mathrm{d}x}{\mathrm{d}\alpha} &= \frac{\mathrm{d}\frac{B}{\alpha n}}{\mathrm{d}\alpha} = \frac{-B}{\alpha^2 n} \\
\frac{\mathrm{d}p_a}{\mathrm{d}\alpha} &= \frac{\mathrm{d}p_a}{\mathrm{d}x} \cdot \frac{\mathrm{d}x}{\mathrm{d}\alpha} = \\
& \sum_{y=1}^{n-1} \frac{FB}{\alpha^2 n(y+x)^2} \cdot \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y} = \\
& \sum_{y=1}^{n-1} \frac{Fx}{\alpha(y+x)^2} \cdot \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y} < \\
& \sum_{y=1}^{n-1} \frac{Fx}{\alpha x^2} \cdot \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y} = \frac{F}{\alpha x}
\end{aligned}$$

A.1 Bounding p_u

We now give a bound on p_u.

$$\mu \triangleq E[Y \mid S_{ij}] = \textstyle\sum_{y=1}^{n-1} y \cdot \binom{n-2}{y-1} q^{y-1}(1-q)^{n-1-y} = \frac{n-2}{n-1} \cdot F + 1$$

$$E[Y^2 \mid S_{ij}] = \textstyle\sum_{y=1}^{n-1} y^2 \cdot \binom{n-2}{y-1} q^{y-1}(1-q)^{n-1-y} = \frac{(n-2)(n-3)}{(n-1)^2} \cdot F^2 + 3 \cdot \frac{n-2}{n-1} \cdot F + 1$$

$$\sigma^2 \triangleq Var(Y \mid S_{ij}) = \frac{(n-2)(n-3)}{(n-1)^2} \cdot F^2 + 3 \cdot \frac{n-2}{n-1} \cdot F + 1 - \left(\frac{n-2}{n-1} \cdot F + 1\right)^2 = \frac{n-2}{n-1} \cdot F - \frac{n-2}{(n-1)^2} \cdot F^2$$

By [34], for $n \gg 1$ we get that Y given S_{ij} can be approximated using a normal distribution function, with $\mu = F + 1$ and $\sigma^2 = F$. The cumulative distribution function $D(x)$ is thus given by:

$$D(x) = \tfrac{1}{2} \cdot \left(1 + \operatorname{erf}\left(\tfrac{x-\mu}{\sqrt{2}\sigma}\right)\right) = \tfrac{1}{2} \cdot \left(1 + \operatorname{erf}\left(\tfrac{x-F-1}{\sqrt{2F}}\right)\right) \quad \text{where} \quad \operatorname{erf}(z) = 1 - \frac{2}{\sqrt{\pi}} \int_z^{\infty} e^{-t^2} \mathrm{d}t$$

From [34], we get the following:

$$\frac{1}{x+\sqrt{x^2+2}} < e^{x^2}\int_x^\infty e^{-t^2}\mathrm{d}t < \frac{1}{x+\sqrt{x^2+\frac{4}{\pi}}}$$

Concluding that:

$$\mathrm{erf}(z) = 1 - \frac{2}{\sqrt{\pi}}\int_z^\infty e^{-t^2}\mathrm{d}t > 1 - \frac{2}{\sqrt{\pi}}\cdot\frac{e^{-z^2}}{z+\sqrt{z^2+\frac{4}{\pi}}}$$

The first sum in formula A.1 is approximated by $D(F)$. Calculating $D(F)$ gives:

$$\begin{aligned} D(F) &= \frac{1}{2}\cdot\left(1+\mathrm{erf}\left(\frac{-1}{\sqrt{2F}}\right)\right) > \frac{1}{2}+\frac{1}{2}\cdot\left(1-\frac{2}{\sqrt{\pi}}\cdot\frac{e^{-\frac{1}{2F}}}{\sqrt{\frac{1}{2F}+\frac{4}{\pi}}-\frac{1}{\sqrt{2F}}}\right) = \\ & 1-\frac{1}{\sqrt{\pi}}\cdot\frac{e^{-\frac{1}{2F}}}{\frac{\sqrt{\pi+8F}}{\sqrt{2\pi F}}-\frac{1}{\sqrt{2F}}} = 1-\sqrt{2}\cdot\frac{\sqrt{F}\cdot e^{-\frac{1}{2F}}}{\sqrt{\pi+8F}+\sqrt{\pi}} \end{aligned}$$

Define:

$$g(F) = \frac{\sqrt{F}\cdot e^{-\frac{1}{2F}}}{\sqrt{\pi+8F}+\sqrt{\pi}}$$

We want to bound D(x) from above by finding for which values of F, $g'(F) < 0$. The denominator of $g'(F)$ is always positive, so we ignore it when calculating the derivative:

$$\left(\frac{e^{-\frac{1}{2F}}}{2\sqrt{F}}+\frac{\sqrt{F}e^{-\frac{1}{2F}}}{2F^2}\right)\left(\sqrt{\pi+8F}-\sqrt{\pi}\right)-\frac{8\sqrt{F}e^{-\frac{1}{2F}}}{2\sqrt{\pi+8F}} < 0$$

$$\frac{F^{\frac{3}{2}}+F^{\frac{1}{2}}}{2F^2}\left(\sqrt{\pi+8F}-\sqrt{\pi}\right)-\frac{8\sqrt{F}e^{-\frac{1}{2F}}}{2\sqrt{\pi+8F}} < 0$$

$$\frac{\left(F^{\frac{3}{2}}+F^{\frac{1}{2}}\right)\left(\sqrt{\pi+8F}-\sqrt{\pi}\right)\sqrt{\pi+8F}-8F^{\frac{5}{2}}}{2F^2\sqrt{\pi+8F}} < 0$$

Once again, the denominator is positive, and we get:

$$\left(F^{\frac{3}{2}}+F^{\frac{1}{2}}\right)\left(\sqrt{\pi+8F}-\sqrt{\pi}\right)\sqrt{\pi+8F}-8F^{\frac{5}{2}} < 0$$

$$\pi+8F-\sqrt{\pi^2+8\pi F}-8F\cdot\left(1-\frac{1}{F+1}\right) < 0$$

$$\frac{8F}{\sqrt{\pi}(F+1)} < \sqrt{\pi+8F}-\sqrt{\pi}$$

Taking derivatives we get:

$$\begin{aligned}\frac{8}{\sqrt{\pi}(F+1)^2} &\overset{?}{<} \frac{8}{2\sqrt{\pi+8F}}\\ 2\sqrt{\pi+8F} &\overset{?}{<} \sqrt{\pi}(F+1)^2\end{aligned}$$

Clearly, $(F+1)^2$ grows faster than $2\sqrt{\pi+8F}$. Numerically solving for $F = 1$ shows that the inequality holds. Thus, it holds for every $F \in \mathbb{N}$. Consequently, we only need to find the first F for which:

$$\frac{8F}{\sqrt{\pi}(F+1)} < \sqrt{\pi+8F} - \sqrt{\pi}$$

A numerical solution for this inequality shows that it first holds for $F = 3$. Thus, for $F \geq 3$ we get that $g'(F) < 0$, and thus $D(F+1) > D(F)$. Assigning $F = 3$ in our previous bound for D(F), we get that for all $F \geq 3$, $D(F) \geq D(3) > 0.3968 \approx 0.4$. Assuming $F \geq 3$, we get:

$$\sum_{y=1}^{F} \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y} > 0.4$$

Since $D(x)$ is maximal at $x = \mu = F+1$ and symmetric around it, we get the approximation:

$$\sum_{y=F+1}^{2F} \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y} > \sum_{y=1}^{F} \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y}$$

And finally, we conclude that:

$$\begin{aligned} p_u &= \sum_{y=1}^{F} \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y} + \\ & \sum_{y=F+1}^{n-1} \frac{F}{y} \cdot \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y} > \\ & \frac{2}{5} + \sum_{y=F+1}^{2F} \frac{F}{2F} \cdot \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y} > \\ & \frac{2}{5} + \frac{1}{2} \cdot \sum_{y=1}^{F} \binom{n-2}{y-1} \left(\frac{F}{n-1}\right)^{y-1} \left(\frac{n-1-F}{n-1}\right)^{n-1-y} > \\ & \frac{2}{5} + \frac{1}{2} \cdot \frac{2}{5} = \frac{3}{5} \end{aligned}$$

Appendix B

Calculating $\widetilde{p}$

We now compute $\widetilde{p}$, the probability that M propagates from the source in a round. Let Y be the number of valid messages received by the target in a single round, and Z be the number of processes the choose the target in a single round, then:

$$\Pr(Y = y\ ,\ Z = z) \quad = \quad \binom{n-b-1}{z} \left(\frac{|view_{push}|}{n-1}\right)^{z} \left(1 - \frac{|view_{push}|}{n-1}\right)^{n-b-1-z} \binom{z}{y} (1-\epsilon_{loss})^{y} {\epsilon_{loss}}^{z-y}$$

Where, $0 \leq y \leq z < n - b$ (b is the number of malicious processes). Therefore,

$$\Pr(Y = y) \quad = \quad \sum_{z} \Pr(Y = y\ ,\ Z = z\ ) = \sum_{z=y}^{n-b-1} \binom{n-b-1}{z} \left(\frac{|view_{push}|}{n-1}\right)^{z} \left(1 - \frac{|view_{push}|}{n-1}\right)^{n-b-1-z} \binom{z}{y} (1-\epsilon_{loss})^{y} {\epsilon_{loss}}^{z-y}$$

Let $\acute{X}$ be the number of fabricated push messages that the target receives in a single round, then:

$$\Pr(\acute{X} = \acute{x}) = \binom{x}{\acute{x}} (1-\epsilon_{loss})^{\acute{x}} {\epsilon_{loss}}^{x-\acute{x}}$$

Let $p_Y(k)$ be the probability that source accepts exactly k ($k \leq F$) valid pull-request messages, then:

$$p_Y(k) = \begin{cases} 0 & Y < k \\ \frac{\binom{Y}{k}\binom{\acute{X}}{F-k}}{\binom{Y+\acute{X}}{F}} & Y \geq k \text{ and } \acute{X} \geq F-k \\ 1 & Y = k \text{ and } \acute{X} < F-k \\ 0 & Y > k \text{ and } \acute{X} < F-k \end{cases}$$

Calculating $\widetilde{p}$ gives:

$$\widetilde{p} = 1 - \sum_k (\epsilon_{loss})^k \sum_y \sum_{\acute{x}} p_y(k) \cdot \Pr(\acute{X} = \acute{x}) \cdot Pr(Y = y) =$$

$$1 - \sum_{k=0}^{F} (\epsilon_{loss})^k \sum_{y=0}^{n-b-1} \sum_{\acute{x}=0}^{x} p_y(k) \cdot \Pr(\acute{X} = \acute{x}) \cdot Pr(Y = y)$$