Evaluation of TCP State Replication Methods for High-Availability Firewall Clusters

Yi-Hsuan Feng, Nen-Fu Huang, Senior Member, IEEE, and Yen-Min Wu

Abstract—To provide the reliable connectivity between two endpoints over the Internet, a firewall cluster for stateful high availability removes the single-point failure by replicating and maintaining TCP connection states to a backup firewall node, at the expense of the costs of network and system resources. In this paper, through trace-based simulations on a prototype implementation, we evaluate the overheads of different state replication methods with a tunable time-triggering parameter. Our evaluation results show that the overheads of precise replication are very high, especially for short flows. We find that a compact data structure employing randomization, a small delay on the replication operations, and host-level aggregation yield significant overhead reductions. Typically, the policy of delayed replication reducing 50% and 74.4% of bandwidth costs only excludes 1.9% and 3.4% of the protection on the pass-through traffic, respectively. These schemes and policies are efficient for alleviating peak system load, reducing the replication bandwidth consumption and still protecting the majority of Internet traffic bytes.

Index Terms—firewall, state replication, high availability, failover.

I. INTRODUCTION

From the service-provider standpoint, any service outage can mean lost customers and thus lost revenue. High availability (HA) technologies are widely used by the clustering and chassis-based equipments on the highly-valuable links (e.g., the highest bandwidth link of an enterprise) to increase the service availability and reliability.

A stateless HA cluster can be simply achieved by using the stateless mechanisms for network redundancy (e.g., VRRP [1]) and identical configuration/ruleset. However, without state replication, all legitimate connections (a.k.a. flows), or even worse whole user requests, have to be re-established after a failover due to the loss of flow states in the firewall cluster. On the other hand, in a stateful HA scenario for firewalls, a state replication protocol (SRP) (e.g., pfsync [2] for OpenBSD IP Filter and ct_sync [3] for Linux netfilter) provides replication management and supports reliable connections at cluster-level by switching the active connections to the secondary firewall node transparently in a failover. Namely, an SRP is complementary to stateless failure detection by maintaining state consistency between a firewall node and its backup. Note that both the pfsync and ct_sync protocols adopt the passive replication and rely on explicit messages to replicate three in-order state types (i.e., insertion, modification, and deletion) via multicasting.

Many other solutions have been proposed to provide connection-level reliability such as fault tolerance in TCP [4-9], OS mechanism [10], [11], and new transport protocols [12], [13]. However, although these schemes can be deployed, achieving reliable connectivity remains a challenge. For a user, there is no difference between the service outages due to the networks and due to the servers. Any near-source or near-destination single-point failure still hinders the service quality. When network failures are considered, service availability is often as low as 99%, meaning that a server is out of service for about 15 min a day on average [14]. Furthermore, Boutremans, et al. [15] find that an availability degradation of VoIP service results from the reliability problem of routing equipment. They identify the need for more reliable hardware architecture and fast protection mechanisms against link failures.

Despite a stateful HA cluster complements flow-level protections by removing the single-point failure, an SRP also consume network and system resources to protect a connection. In this paper, we evaluate the costs of different state replication methods for TCP connections (the majority of Internet traffic) and explore the tradeoffs from varying a time-triggering parameter to replicate a connection. The packet traces (see Table I) collected from major Internet backbones and from campus are used to simulate the replication operations on a prototype implementation.

The remainder of this paper is organized as follows. Section II provides an overview of two popular HA modes adopted by firewall clusters. Section III discusses four state replication methods. In Section IV, we discuss that an imprecise replication strategy built on the flow-level traffic characterization might be beneficial. Section V presents the evaluation results driven by the real-world packet traces on a prototype implementation. Finally, Section VI concludes this paper.

II. BACKGROUND: HA MODES

There are various HA techniques for different applications and network infrastructures. In this paper, two distinct HA modes used by firewall clusters are considered as shown in Fig. 1: the active/backup (AB) mode for redundancy and...
For example, many advanced firewalls keep track of the mutable information of a flow from SYN to its completion. Eager replication synchronizes every change on the replicated states to reflect the active flows on the pass-through link.

A firewall cluster in AA mode is composed of two or more active nodes (ANs) sharing the pass-through loads, both playing as the message sender and receiver in replication management. The configuration in AA mode removes a possible throughput bottleneck and is superior in the areas of load balancing capability, better resource utilization, and a lower throughput penalty in the case of failure. In general, an AN/PN can be backed up by a number of cluster nodes. However, in this paper, we consider only the single backup configuration which is often used in practice.

In Fig. 1, on the boundary of clustering pair, there are two load-balancing switches (LBSes) which distribute traffic loads evenly across two pass-through strings as well as ensure that a connection passes through the same string in both bi-direction while in AA mode. On the other hand, in AB mode, LBSes direct the traffic to a dedicated string. If a string failure is detected, LBSes redirect the traffic to the other string.

III. STATE REPLICATION METHODS

A. Eager and selective replication

In a firewall device, a state entry is used to store the information derived by TCP stateful tracking from the bi-directional traffic of a TCP flow. As a flow is initialized and terminated, the corresponding entry is inserted to and removed from the flow table. Each entry contains two types of flow sub-states: immutable and mutable. An immutable sub-state flowID, i.e., four-tuple (DstIP, SrcIP, DstPort, SrcPort), remains constant and is used to identify a connection. Mutable sub-states may be changed very frequently, such as the latest packet arrival time, sequence and acknowledgement numbers, window advertisement and total flow bytes.

Two replication methods are first considered: eager replication and selective replication. Besides the immutable information, eager replication synchronizes every change on the mutable information of a flow from SYN to its completion. For example, many advanced firewalls keep track of the sequence and acknowledgement numbers and TCP flags continuously to ensure the active flows are compliant with the TCP specification in all aspects. To meet these criteria after a failover, eager replication must be adopted by a stateful firewall cluster for synchronizing the mutable information.

Though the message sizes of the pfsync and ct_sync both exceed 100 bytes, an explicit 32-byte-long representation is used to update the following data for investigating the costs of eager replication.

- flowID
- Sequence and acknowledgement numbers
- Segment size, window scale, and TCP flag
- Timestamp
- Operation and direction flag

Another method, selective replication, synchronizes three state changes (i.e., SYN_SENT, ESTABLISHED, and flow completion) only. Actually, both the pfsync and ct_sync use a strategy similar to selective replication to optimize state copying operations. In [6], a selective mechanism is used to save the processing time of the backup server. Furthermore, note that in our evaluation, a 16-byte-long message (only flowID and operation flags) is used by selective replication to evaluate the overheads.

B. Flow Digest

The scheme Flow Digest (FD) [16] improves the procedures of state replication through two factors: 1) an architectural improvement prevents the flow table from the access by replication traffic before a failover, and 2) a compact data structure employing randomization (i.e., Bloom filters) is designed to reflect the active flows.

For PN and AN, all established flows are collected into a terse set representation and synchronized to the backup node by sending a Bloom filter or incremental messages. The scheme shifts to the recovery phase after a failover and the flow table is reconstructed in a packet-driven fashion by querying the backup Bloom filter.

The memory requirement of FD can be kept small, while still achieving high accuracy. For example, for supporting 1,000,000 connections in maximum, using 10,000,000 elements, four bits per element, and four hash functions yields a false positive rate of 1.2% and requires a 7,500-KB memory space; not a concern in today’s equipments. Though false positives are possible, FD never rejects active flows after a failover under accurate state replication.

We use the incremental updates to evaluate the overheads of the FD scheme. By large-size Bloom filters, FD can be viewed as a 2-state replication method to synchronize established flows and their terminations, where the message size is 32-bit.
C. Host-to-host aggregation

By ignoring the port number at the two endpoints, small replication operations at the host-level can protect the packets from different TCP flows between the same host pair. Two observations from Fig. 2 with IPLS-3 provide an initial indication of the potential benefits of host-level aggregation. The distributions of IPLS-1 and AUCK-4 are not shown because they are similar with the results of IPLS-3.

First, Fig. 2 illustrates that parallel connections are observed for all lifetime distributions, especially parallel degrees less than 5 connections. One probable reason is that parallel TCP connections are widely used by kinds of applications, like web transfers, multi-stream applications, and P2P. For example, web browsers open parallel connections at the same time to request various objects of a web page. In [17], the analysis of web traffic shows that nearly all web clients open 4 or fewer simultaneous TCP connections to transfer the inline contents. In Firefox 2.0 browser, the default setting for maximum parallel connections per server is increased to 8. The study also points out that as clients transfer more objects, the likelihood of using concurrent connections increases. On the other hand, the studies [18], [19] on the web workloads show that both the number of objects per web page and the number of distinct server delivering content per page are increasing over the years.

Second, as the host-level lifetime decreases, especially less than 5 sec, we observe a clear increase of the number of maximum parallel connections. Our flow analysis shows that many endpoints establish high-degree parallel connections at almost the same time. This implies the overheads of port-level replication for short flows are much higher than that of long flows due to the short burst and parallelism. By aggregating replication operations per source-destination pair, an HA cluster can counteract these potential overheads.

To evaluate the overheads of host-level aggregation, only the first establishment event and the last deletion event between the two endpoints are replicated to the backup node. The message format/size is identical to that of selective replication.

IV. FLOW LIFETIME VS STATE REPLICATION OVERHEADS

Thus far, in order to guarantee consistency, state changes in the primary firewall node are forced to be synchronized precisely. However, this approach is expensive. Measurements of the Internet traffic have shown that most TCP flows are short-running [20] and that long flows (e.g., less than 20%) carry a high proportion (e.g., 85%) of the total traffic bytes [20], [21]. Furthermore, long HTTP sessions of purchases are more profitable for web sites [22] and should be protected for successful completion. For web traffic [18], 15% of the TCP connections are persistent. However, these persistent connections deliver approximately 40% of the transferred bytes for web objects.

The lifetime and size distributions of the Internet traffic add another dimension to state replication. The above studies imply that the major costs come from short flows when doing replication, but focusing on long flows protects the majority of network traffic (in bytes) and profits. For instance, the efficiency of replicating a flow less than 50 ms is extremely low, particularly when this service is not critical for users. Because of the increased memory accesses and network operations, the cluster performance is likely to be impacted negatively due to the heavy loads from an SRP. Clearly, there is a tradeoff between good pass-through performance in failure-free duration and minimal recovery overheads after a failover.

A lazy threshold $t_{threshold}$ is used as a time-triggering parameter which refers to how long a flow has persisted before a replication operation is performed for that flow. We explore the effects of varying this parameter on the costs of four replication methods mentioned above. A $t_{threshold} = 0$ represents the precise replication which indicates the flow replication is dependent on the state replication method only.

V. EVALUATIONS

A. Implementation and packet trace data

The experiments were performed on a testbed consisting of two identical 3-port machines (Intel Pentium-4 2.0GHz and 512MB RAM) as the cluster nodes. Two nodes are connected with a 100Mbps LAN (the replication link). As the base-line implementation, both machines run Linux 2.4.20 with our kernel module and patch installed. A tasklet implements the stateful tracking subsystem, four replication schemes, and lazy threshold in the kernel space. The flow table is implemented by a hash table. The inactivity timeouts for the idle entries in SYN_SENT, ESTABLISHED, and FIN_WAIT states are 20, 60, and 20 sec, respectively. Two kernel threads are used to send and receive the packets via UDP multicasting on the replication link.

We validate four replication methods by the trace-based simulation, which gives us the imitation of the activities of known backbone/campus networks and a practical picture of benefits and drawbacks of the given methods. The replication schemes are applied to the bi-directional 10-min traces of Abilene-I and Abilene-III (denoted as IPLS-1 and IPLS-3) which were collected from OC-48c and OC-192 links in 2002 and 2004 [24]. Another bi-directional 24-hour packet trace files (denoted as AUCK-4) from NLANR [24] were captured at the University of Auckland in 2001. All inbound and outbound packets with the corresponding metadata are read from the trace.
files and then sent to the kernel space sequentially as the input packets. At the end of reading trace data, all active flows are forced to complete and then passed to flow analysis.

To enable a fair comparison, we ignore purposely the replication for the flows whose SYN packets were not captured in the trace files, though this leads to an underestimation of the pass-through traffic (especially long flows) and replication costs. Furthermore, due to the fact that routes may be asymmetric at the backbone, there is a minor tuning in stateful tracking.

For setting FD parameters, the maximum number of the allocated state entries in three traces is 159,394. Therefore, we set the maximum concurrent connections supported by a cluster node as 200K and set the Bloom filter size as 2,000K elements. The MD- is used as the hashing functions and the hash number is 4.

B. Trace-based evaluation results

To understand the effects of the imprecise replication, the reductions of the protected pass-through and replication traffic are studied by tuning the parameter $t_{\text{threshold}}$ from 0 to 20,000 ms. Note that the active flows whose states are already replicated are referred to as the protected flows. On the other hand, the overhead of a replication method is measured by its bandwidth costs and the protected traffic bytes (TCP payloads). Let $N_{\text{replication}}$ and $N_{\text{protection}}$ be the total bytes of transmitted replication messages and pass-through packets whose states already have been replicated, respectively. Then, the overhead is defined as $(N_{\text{replication}})/(N_{\text{protection}})$. Figs. 3–5 illustrate the evaluation results in AB mode.

First, it is observed that eager and selective methods are vulnerable to one-way flows and malicious SYN packets. For example, in IPLS-1 and IPLS-3, 9.9% and 39.2% of the recycled state entries get stuck in SYN_SENT state. In the case of IPLS-3, the flow analysis shows that 87.1% of the recycled SYN_SENT entries are allocated by one-way flows (almost sending only 1 to 3 packets), and the remaining 11% are the two-packet flows (SYN and RST). In a cluster using eager/selective replication, a short one-way flow allocates two state entries (in PN/AN and its backup node), which are recycled and deleted immediately after an inactivity timeout (20 sec in our simulations). Thus, these one-way flows significantly aggravate the contention on the system resources of two cluster nodes and bandwidth consumptions on the replication link. By contrast, because FD and host-level aggregation only replicate the established flows (namely, from ESTABLISHED state), the number of the deletion events activated by the SYN_SENT timeouts are much less than those of eager and selective methods. The measurement of one-way and two-way flows has been the subject of research in [25].

In Figs. 3(a)–5(a), when $t_{\text{threshold}}=50$ ms, due to the savings of replicating one-way flows and malicious SYN packets, the reduction ratios of selective replication on the bandwidth costs are as high as 24.8%, 27.8%, and 16.7% in IPLS-1, IPLS-3, and AUCK-4. By contrast, the cost reduction ratios of FD and host-level aggregation are only 0.05%–6.3% by the same $t_{\text{threshold}}$. On the other hand, because most TCP flows of the Internet traffic are short-lasting, they dominate the state replication costs. Except for eager replication, a very clear rise on the replication traffic reductions is observed when $t_{\text{threshold}}<1$ sec in three packet traces.

About the reductions on the protected pass-through bytes, Figs. 3(a)–5(a) show that only 0.09–1.6% of total pass-through bytes are not protected by $t_{\text{threshold}}=50$ ms, but up to 27.8% of the replication traffic are saved for selective replication. For FD in IPLS-3, reducing 50% ($t_{\text{threshold}}=320$ ms), 74.4% ($t_{\text{threshold}}=500$ ms), and 88.9% ($t_{\text{threshold}}=2,000$ ms) of the replication traffic excludes only 1.9%, 3.4%, and 11.8% of the pass-through bytes. The host-level aggregation has a very
similar behavior. Remember that the resource savings come from the reduced replication operations. The reductions on the replication costs and protected bytes mirror the cumulative flow lifetime and size distributions in packet traces. Obviously, the efficiency of replicating flows longer than 500 ms is much higher than the precise replication. A small \( \tau \) threshold can be useful for alleviating peak system load, reducing bandwidth consumption, and protecting the majority of Internet traffic bytes.

In Figs. 3(b)–5(b), we show the overheads of four replication methods. The overheads of eager replication do not decrease significantly as \( \tau \) threshold increases. This confirms the high costs of keeping all mutable information consistent between the cluster nodes. In Fig. 3(b), when \( \tau \) threshold =0 ms in IPLS-1, the overheads of selective method, FD, and host-level aggregation are only 7.9%, 2.4%, and 3.3% of the overhead of eager replication, respectively.

Figs. 3(b)–5(b) show that the overheads of the FD scheme are much less than those of eager and selective replication. For example, in IPLS-3 at \( \tau \) threshold=500 ms, FD reduces 99.5%, 79.8%, and 22.9% overheads when compared to eager, selective, and host-level aggregation methods. Furthermore, though host-level aggregation avoids the operations for parallel connections, except for \( \tau \) threshold < 1 sec in AUCK-4, the overheads of host-level aggregation are slightly higher than FD. This is because the message size of the FD incremental update is 32 bits and the size of host-level aggregation is 16 bytes.

Another important metric is the number of replicated entries in the backup node. Though this metric may be not critical to an active/backup cluster, the valuable state entries of an active/active cluster are allocated both by pass-through flows and replicated flows. Thus, we perform simulations in AB mode to investigate the effects of \( \tau \) threshold on the maximum number of replicated entries in the backup node. Note that, in FD no replicated entry is required in the flow table of the backup node.

For eager and selective methods, Fig. 6 illustrates that varying \( \tau \) threshold from 0 to 50 ms reduces the maximum replicated entry number from 128,852 to 89,425 (a 30% decrease) due to the effects of one-way flows. A \( \tau \) threshold=2,000 ms reduces 45% of the maximum number of eager/selective replication (from 128,852 to 69,897), while host-level aggregation reduces it by 44.4% (from 12,873 to 7,152) at the same \( \tau \) threshold. Fig. 6 also illustrates that parallel connections exist for all lifetimes and parallelism degree increases as the lifetime decreases. When \( \tau \) threshold=50 ms, the entry number of host-level aggregation is 12,818; only 14.3% of the requirements of eager/selective replication.

VI. CONCLUSION

To improve service availability and reliability, the stateful HA firewall clusters are deployed to remove network single-point failures. In this paper, we perform the simulation tests by real backbone/campus packet traces to evaluate the costs of four state replication methods as the possible solutions for firewall clusters with a tunable time-triggering parameter. To the best of our knowledge, there have been no cost evaluation results of the flow-level state replication methods over HA clusters available. We believe that our results also give a practical view to other technologies using TCP state replication, like transparent TCP-connection migration.

We find that the precise replication overheads for short flows are high, because most TCP flows are short-running and the short flows are likely to have high-degree parallel connections. Thus, a small time delay can yield significant reductions on the bandwidth costs and cluster resources. Typically, reducing 50% and 74.4% of bandwidth costs only excludes 1.9% and 3.4% of the protection on the pass-through traffic. Moreover, the overheads of the FD scheme are lowest in nearly all the tests we ran. For the active/active clusters, an important metric, the maximum replicated entry number in the backup node, is investigated. The results show that both the host-level aggregation and a time trigger larger than 5 sec reduce effectively the number of replicated state entries. In summary, our investigation highlights the benefits of the imprecise state replication, including the scheme employing randomization, the host-level aggregation and the time-delay policy.

Besides above results, we further suggest that an SRP should
replicate a TCP flow from its ESTABLISHED state. This strategy avoids the high costs, such as a high entry-recycling rate and unnecessary bandwidth consumptions, from very short one-way flows and malicious SYN packets.

REFERENCES